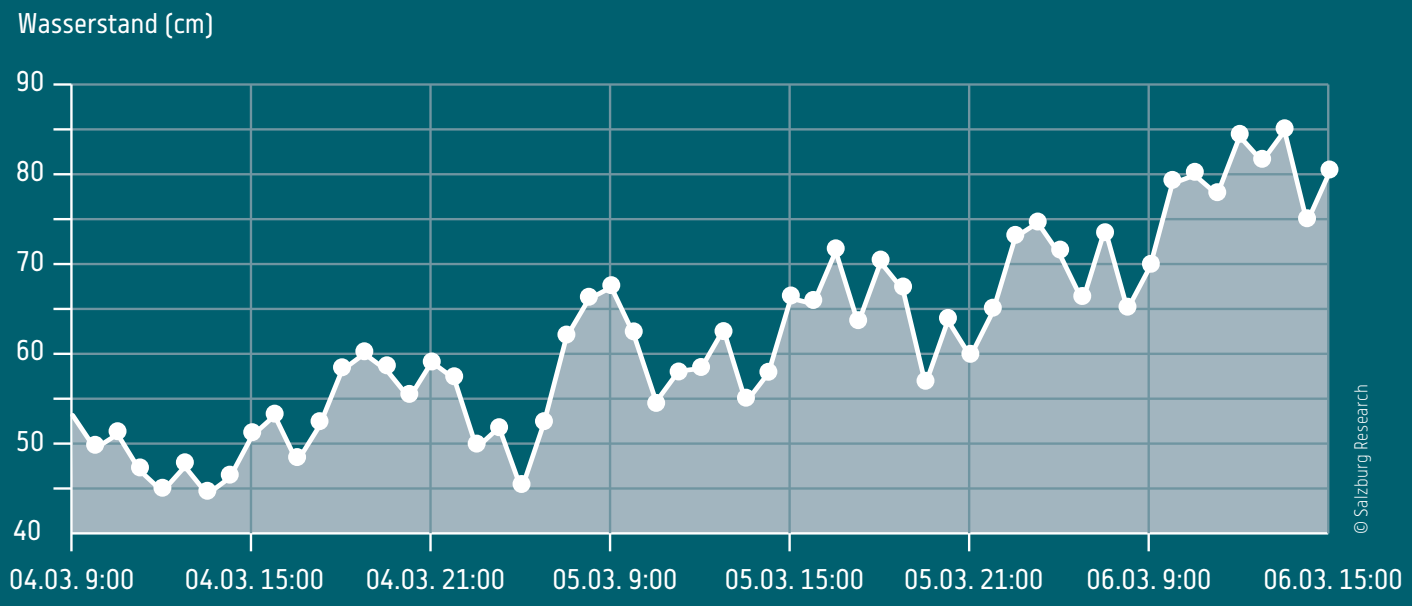
salzburgresearch

Schmalband Maschine-zu-Maschine Kommunikation

Ein Handbuch



Matthias Herlich, Thomas Pfeiffenberger, Ferdinand von Tüllenburg,
Georg Panholzer, Dominik Andexer, Peter Dorfinger, Siegfried Reich

Schmalband Maschine-zu-Maschine Kommunikation – Ein Handbuch

Matthias Herlich
Thomas Pfeiffenberger
Ferdinand von Tüllenburg
Georg Panholzer
Dominik Andexer
Peter Dorfinger
Siegfried Reich

2018-02-28

Salzburg Research Forschungsgesellschaft mbH
Jakob Haringer Str. 5/III
5020 Salzburg
T 0662 2288 200
F 0662 2288 222
M {vorname.nachname}@salzburgresearch.at

M. Herlich, T. Pfeiffenberger, F. Tüllenburg, G. Panholzer, D. Andexer, P. Dorfinger, S. Reich
 Unterstützt vom Land Salzburg - Abteilung 1: Wirtschaft, Tourismus und Gemeinden.

Zusammenfassung

Smart Grid, Smart Home, Smart City und Industrie 4.0 benötigen viele Sensordaten. Die Sensoren, die diese Daten erheben, sind aber zum Teil an mit Kabel schwer erreichbaren Orten oder eine Verkabelung ist mit hohen Kosten verbunden. Die Anbindung der Sensoren durch Funktechnologien ermöglicht die schnelle und einfache Nutzung von Sensoren. Daher bietet das drahtlose Auslesen von Sensoren enorme Vorteile.

Für kabellose Sensoren ist oft nicht die Datenrate der Verbindung, sondern niedriger Energieverbrauch, hohe Zuverlässigkeit und große Reichweite wichtig. Eine mögliche Lösung, die diese Anforderungen erfüllt, ist Schmalbandkommunikation (Narrowband-Kommunikation, NB). Narrowband-Technologien erlauben es effizient kleine Datenmengen zu übertragen. Sie werden hauptsächlich für Maschine-zu-Maschine (M2M) Kommunikation verwendet. Diese M2M Kommunikation ist ein wichtiger Baustein für neue Trends, wie Smart Grid, Smart Home, Smart City und Industrie 4.0.

Es gibt mehrere Technologien im Bereich der Narrowband-Kommunikation, welche wir in diesem Handbuch näher betrachten werden. Vor allem LoRa, Sigfox und NB-IoT (Narrowband - Internet-of-Things) als kommende Standards für die Vernetzung der zahlreichen Geräte im „Internet-of-Things“ sind der Fokus. Der Einsatz dieser neuen Technologien wird mobile Sensoranwendungen stark vereinfachen und somit Unternehmen zu visionären Innovationen führen. Beispiele hierfür sind intelligentes Parkplatzmanagement, automatisches Auslesen von Wasser-/Stromzählern (Smart Meter), Umweltmonitoring und Wildbach-/Wassermonitoring. Viele Unternehmen haben allerdings keine Vorerfahrung mit drahtloser Kommunikation und können ihre Systeme nicht ohne weiteres auf Schmalband-Kommunikation erweitern/umstellen.

Dieses Handbuch gibt Empfehlungen, die es Unternehmen (vor allem kleinen und mittleren Unternehmen) ermöglichen ihre bestehenden Systeme weiter zu entwickeln und innovative Anwendungen auf Basis der Narrowband-Technologien zu entwickeln.

Abstract

Smart Grid, Smart Home, Smart City and Industry 4.0 need a lot of sensor data. However, the sensors, which collect this data, are often at locations, which are hard to reach with cables and cable installation is usually expensive. Connecting the Sensors with wireless technologies allows fast and easy use of the sensors. Therefore, the wireless transmission of sensor values provides enormous advantages.

For wireless sensors the data rate is rarely important. More important are high energy efficiency, reliability, and coverage. A possible solution that provides these features is narrowband (NB) communication. Narrowband-technologies allow the efficient transmission of rare and small data packets (for example, the average temperature transmitted every hour). Such small packets are important for Machine-to-Machine (M2M) communication. This M2M communication is a necessary building block for new trends such as Smart Grid, Smart Home, Smart City and Industry 4.0.

There are many technologies for narrowband communication, of which we will consider the most important ones in this document. LoRa, Sigfox and NB-IoT (Narrowband - Internet-of-Things) are the upcoming standards for the interconnection of many devices in the "Internet-of-things". The use of these technologies will make mobile-sensor applications drastically simpler and, thus, allow visionary innovations. Examples for this are: intelligent parking management, automatic reading of water/power meter (smart meters), environment monitoring (for example, of rivers). However, many companies have no experience with wireless communication and cannot simply build systems based on narrowband communication.

This document gives companies (mostly small and medium sized) companies an overview how to expand their existing systems and develop new applications based on narrowband technologies.

Inhaltsverzeichnis

1 Einleitung

Smart Grids, Smart Home, Smart City und Industrie 4.0 benötigen viele Sensordaten. Beispiele hierfür sind intelligentes Parkplatzmanagement, automatisches Auslesen von Wasser-/Stromzählern (Smart Meter), und Umweltmonitoring wie Überwachen des Wasserstandes von Wildbächen. Die Sensoren, die diese Daten erheben, sind aber zum Teil an mit Kabel schwer erreichbaren Orten oder eine Verkabelung ist mit hohen Kosten verbunden. Die Anbindung der Sensoren durch Funktechnologien ermöglicht die schnelle und einfache Nutzung der Sensoren.

Für kabellose Sensoren stehen oft nicht die Datenrate der Verbindung im Vordergrund, sondern niedriger Energieverbrauch, hohe Zuverlässigkeit und große Reichweite. Eine mögliche Lösung, die diese Anbindung erlaubt, ist Schmalband-Kommunikation. Schmalband-Kommunikation umfasst eine Gruppe von drahtlosen Kommunikations-Technologien, die darauf spezialisiert sind, effizient kleine Datenmengen zu übertragen. Sie werden hauptsächlich für die Kommunikation zwischen Maschinen verwendet und bilden einen wichtigen Baustein für Trends, wie Smart Grid, Smart Home, Smart City und Industrie 4.0.

Es gibt mehrere vielversprechende Schmalband-Technologien. Die bekanntesten sind LoRa, Sigfox und NB-IoT (Narrowband - Internet-of-Things). Die Wahl der richtigen Technologie für einen Einsatzzweck ist aber nicht einfach: Die Technologien unterscheiden sich nicht nur in den technischen Merkmalen der drahtlosen Übertragung, sondern auch in der benötigten Infrastruktur und dem Management.

In diesem Handbuch geben wir zuerst einen Überblick über die Möglichkeiten der Schmalband-Technologien, Vergleichen diese und geben Hinweise welche Technologien sich für welche Einsatzzwecke eignen (Kapitel 2). Anschließend beschreiben wir aufbauend auf dem Überblick welche Innovationen mit Schmalband-Technologien möglich sind (Kapitel 3). Die Kapitel 2 und 3 sind so geschrieben, dass sie für Leser mit allgemeinem technischen Verständnis und Interesse am Thema verständlich sind. Sie geben somit einen guten Einblick in die Möglichkeiten von Schmalband-Kommunikation ohne sich mit den technischen Details zu beschäftigen.

Die folgenden Kapitel beschreiben detailliert wie ein einfaches System zum entfernten Überwachen einer Temperatur mit LoRa(Kapitel 4), Sigfox (Kapitel 5) und NB-IoT (Kapitel 6) umgesetzt werden. Daher erfordern diese Kapitel mehr technisches Wissen als die vorhergehenden Kapitel.

Kapitel 7 beschreibt, wie auf Basis der in den vorherigen Kapiteln beschrieben Infrastruktur andere Daten gemessen werden können. In diesem Beispiel wird eine Position bestimmt und visualisiert. Eine Messumgebung, die die drahtlose Kommunikation überwacht, ist zuletzt in Kapitel 8 beschrieben.

Abbildung 1.1 illustriert die Abhängigkeiten zwischen den Kapiteln. Zusätzlich zeigt sie welche Anforderungen die Kapitel haben.

Warnung

Die in diesem Handbuch beschriebenen Umsetzungen sind **nicht** auf Datensicherheit geprüft. Für ein System, das im produktiven Einsatz genutzt wird, ist es unentbehrlich diese Prüfung durchzuführen! Sollte diese Prüfung nicht durchgeführt werden, können die Daten von Unbekannten beobachtet oder manipuliert werden.

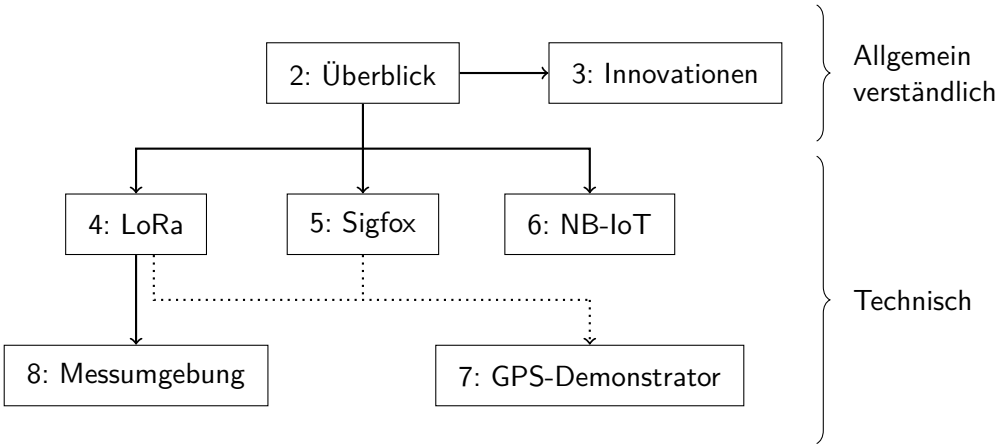


Abbildung 1.1: Abhängigkeiten zwischen den Kapiteln

2 Überblick und Vergleich

Dieses Kapitel gibt einen kurzen Überblick über die bekanntesten Schmalband-Technologien: LoRa, Sigfox und NB-IoT. Wir fokussieren uns hierbei auf die für den Endnutzer wichtigsten Eigenschaften und verweisen für technische Details auf weitere Quellen.

2.1 LoRa

Die Schmalband-Technologie LoRa wird hauptsächlich durch die LoRa Alliance entwickelt.[1] LoRa besteht aus zwei Teilkomponenten: Der Beschreibung der physikalischen Übertragungsschicht (LoRa) und der darauf aufbauenden Architektur und Nachrichten (LoRaWAN).

Die Details der physikalischen LoRa-Schicht sind nicht öffentlich einsehbar, wurden aber zum Teil durch Reverse-Engineering veröffentlicht.[2] Die LoRaWAN Schicht hingegen ist ein offener Standard. Wenn von "LoRa"die Rede ist, ist damit meist LoRaWAN über LoRa gemeint.

Um eine LoRa Übertragung (LoRaWAN über LoRa) durchzuführen ist neben dem Endgerät ein Gateway nötig, dass die LoRa Pakete empfängt. Die Architektur eines LoRa Netzes ist ähnlich der von WLAN: Ein Endgerät kann über ein in der Nähe befindliches Gateway Daten übertragen. Da LoRa und WLAN allerdings unterschiedliche Einsatzzwecke haben, ist bei LoRa die Reichweite höher, die Datenrate geringer und die Energieaufnahme geringer. Da LoRa nicht auf IP-Paketen basiert, kann das Gateway die empfangenen Datenpakete nicht direkt in ein IP Netz weiterleiten. Ein LoRa-Gateway ist für einen LoRa-Server konfiguriert, an den alle empfangenen Daten weitergeleitet werden. Dieser LoRa-Server kann die Daten dann an andere Server weiterleiten, in eine Datenbank speichern und visualisieren. Über die Verarbeitung der Daten nach dem Empfang durch den LoRa-Server umfasst LoRa keine Spezifikation. Diese Auswertung obliegt dem jeweiligen Nutzer.

Die physikalische Übertragung von LoRa findet in Europa im Frequenzband um $868\,\mathrm{MHz}$ statt. Hierzu sind keine Lizenzen nötig, allerdings kann auch nicht ausgeschlossen werden, dass andere Geräte im gleichen Frequenzband senden.

2.1.1 Anbieter

Im Allgemeinen ist jeder Nutzer von LoRa selbst für den Betrieb von LoRa-Gateways verantwortlich. Ähnlich zu Zusammenschlüssen von WLAN Betreibern (z. B. eduroam) gibt es auch Zusammenschlüsse von LoRa-Gateway Betreibern. Diese bieten entweder ihre Gateways kommerziell an oder sind offene Gruppen, die versuchen durch Teilen Ihrer Gateways eine möglichst flächendeckende LoRa-Abdeckung zu erzielen. Beispiele für Zusammenschlüsse/Betreiber von LoRa Netzen sind:

- `https://www.thethingsnetwork.org`
- `http://objenious.com` (französisch)
- `https://loriot.io`
- `https://www.actility.com`
- `https://www.orbiwise.com`

[1] `https://www.lora-alliance.org`
[2] `https://media.ccc.de/v/33c3-7945-decoding_the_lora_phy`

In Österreich hat die ORS (mit Kapsch) bereits Tests mit einem LoRa Netz durchgeführt.[3] Zu deren Ausgang ist allerdings nichts verlässliches bekannt.

2.1.2 Symphony Link

Eine LORA-Implementierung, die im Vergleich zu LoRaWAN einige Erweiterungen insbesondere für industrielle Anwendungen bietet, ist Symphony Link[4] (entwickelt durch LinkLabs). Im Vergleich zu LoRaWAN bietet Symphony Link insbesondere:

- Garantierte Nachrichtenauslieferung basierend auf Acknowledgements und Sendewiederholungen (LoRaWAN bietet dies auch optional)
- Erweiterbarkeit der Gateway-Funktionen um eine verteilte Informationsverarbeitung zu ermöglichen (ähnlich Edge bzw. Fog Computing). Damit kann Basisfunktionalität einer Anwendung auch bei gestörtem Internet-uplink erreicht werden.
- Security durch AES und Over-the-Air Key distribution (zumindest zum Teil auch in LoRaWAN enthalten)
- Repeater Hardware zur Vergrößerung des Netzes ohne (teurere) Gateways

2.2 Sigfox

Sigfox[5] sendet wie LoRa im 868 MHz Band. Im Gegensatz zu LoRa ist es allerdings nicht möglich selbst Gateways zu betreiben. Sigfox sucht sich in jedem Land einen Partner, der in diesem die Infrastruktur bereitstellt. Dies hat den Vorteil, dass der Endnutzer keine eigene Infrastruktur betrieben muss. Dies hat aber auch den Nachteil, dass der Endkunde auf Sigfox als Monopolanbieter angewiesen ist.

Durch die Struktur von Sigfox gibt es in jedem Land höchstens einen Anbieter von Sigfox. In Österreich gibt es bisher noch keinen.[6]

2.3 NB-IoT

NB-IoT ist eine Schmalband-Technologie, welche im Mobilfunkbereich entwickelt wurde. Sie wurde durch die 3GPP in Release 13[7] von LTE spezifiziert. Zur Nutzung von NB-IoT ist zusätzlich zur Client-Hardware ein Vertrag mit einem Mobilfunkanbieter nötig, der NB-IoT anbietet. Ein Vertrag funktioniert ähnlich wie bei einem handelsüblichen Mobilfunkvertrag: Durch Zahlung einer monatlichen Gebühr erhält der Kunde mit einer SIM-Karte Zugriff auf das Netz des Anbieters. Da sich zur Zeit nur Test-Netze im Betrieb befinden ist die Kostenstruktur für die Zukunft noch offen. Die Kosten hängen vom Datenvolumen ab, liegen aber im Moment für etwa 500 KB pro Monat bei etwa 1 € pro Monat für ein Gerät. Für diesen Preis erhält der Kunde die Möglichkeit, über eine Datenverbindung IP-Pakete zu übertragen. Diese können zur Auswertung an einen beliebigen Server gesendet werden.

Alternative Angebote beinhalten einen Server auf der Cloud Infrastruktur des Mobilfunkanbieters. Dort werden die Daten gespeichert und Möglichkeiten zur Visualisierung angeboten. Welche Möglichkeiten diese Auswertung und Visualisierung mit sich bringt, hängt stark vom Anbieter ab.

Anders als Sigfox gibt es bei NB-IoT generell die Möglichkeit den Anbieter zu wechseln, wenn mehrere Betreiber NB-IoT anbieten. Solange die SIM-Karten aber als physische Karten und nicht als

[3] http://www.ors.at/de/presse/pressemitteilungen/loraTM-netzwerk-204/
[4] https://www.link-labs.com/symphony
[5] https://www.sigfox.com
[6] https://www.sigfox.com/en/coverage
[7] http://www.3gpp.org/news-events/3gpp-news/1785-nb_iot_complete

eSIM ausgegeben werden, müssen bei einem Wechsel des Anbieters bei allen Geräten die SIM-Karten getauscht werden.

NB-IoT verwendet eine Kombination aus zeitbasiertem (TDM) und frequenzbasiertem (FDM) Multiplexverfahren für die simultane Signalübertragung über die Funkstrecke. Die Verfahren, die bei NB-IoT eingesetzt werden sind dabei die gleichen, die auch bei LTE-A Anwendung finden, da NB-IoT im Wesentlichen die gleiche Technologie verwendet wie LTE-A und sogar gemeinsam standardisiert wurde. Besonders die für Uplink-Verbindungen verwendete Frequenzmultiplexverfahren SC-FDMA, mit dem binäre Signale in Funkübertragungssignale umgewandelt werden, bietet hohe Effizienz bezüglich Energieverbrauch und gegenseitiger Übertragungsstörungen durch mehrere, gleichzeitig sendende Endgeräte (z. B. Sensoren).

Zunächst ist für SC-FDMA wesentlich, dass im für NB-IoT Kommunikation zur Verfügung stehenden Frequenzband (z. B. im 900 MHz-Bereich) die Unterträger für die parallele Übertragung von Informationen genutzt werden. Diese Unterträger werden im Englischen als Sub-Carrier bezeichnet und sind Träger innerhalb eines Frequenzbands, die beispielsweise für die Übertragung von Zusatzinformationen genutzt werden können. Zum Beispiel wird der Stereoton eines Radiosenders auf einem Unterträger der Frequenz eines Radiosenders übertragen. Die Verteilung der zu übertragenden Information auf mehrere Unterträger wird im Falle von NB-IoT zunächst dazu genutzt, um eine Störungsresistentere (PAPR-Reduktion) aber auch energieeffiziente Informationsübertragung zu erreichen. Dadurch, dass Endgeräte für die Übertragung mit SC-FDMA unterschiedliche Sätze an Unterträgern verwenden, wird gleichzeitig das Potential für wechselseitige Störungen zwischen mehreren, gleichzeitig sendenden Endgeräten minimiert. Die Zuweisung der Unterträger für ein Endgerät erfolgt entweder in Form von direkt benachbarten Subträgern (localized mode) oder durch eine Menge an Subträgern die über das gesamte Frequenzband verteilt sind (distributed mode). Die Auswahl der Subträger, die ein Endgerät benützt wird durch unterschiedliche Koeffizienten erreicht, die jedes Endgerät individuell den verwendeten Modulationsverfahren (Diskrete Fourier Transformation) zuweist.

Quellen:

- `http://literature.cdn.keysight.com/litweb/pdf/5989-8139EN.pdf?id=1431418`
- `https://home.zhaw.ch/kunr/NTM1/literatur/LTE%20in%20a%20Nutshell%20-%20Physical%20Layer.pdf`
- `http://rfmw.em.keysight.com/wireless/helpfiles/89600B/WebHelp/subsystems/lte/content/lte_overview.htm`
- `http://mobilesociety.typepad.com/mobile_life/2007/05/an_introduction.html`
- `https://www.ericsson.com/res/thecompany/docs/publications/ericsson_review/2016/etr-narrowband-iot.pdf`

2.4 Andere (unbekanntere) Technologien

Neben den häufig erwähnten Technologien (LoRa, Sigfox, NB-IoT) gibt es auch weitere Technologien, die ähnliche Anforderungen bedienen. In diesem Abschnitt werden wir kurz einige davon erwähnen. Allgemein funktionieren diese Technologien ähnlich wie die bereits angesprochenen Technologien, sind aber nicht so weit verbreitet oder entwickelt wie die von uns verglichenen Technologien (LoRa, Sigfox und NB-IoT). Daher belassen wir es bei einer kurzen Erwähnung in diesem Kapitel.

- Weightless[8]

[8] `http://www.weightless.org`

- IEEE 802.11ah[9]
- MIOTY[10]
- NWave[11]
- WavIoT[12]
- DASH7[13]
- RPMA[14]

2.5 Vergleich

In diesem Kapitel werden die wichtigsten Narrowband-Technologien verglichen. Der Fokus liegt hierbei nicht auf Detailunterschieden, sondern auf konzeptionellen Unterschieden.

Die wichtigsten Unterscheidungsmerkmale sind:

- Genutzte Frequenz und
- Management der Infrastruktur.

2.5.1 Frequenz

Die verwendete Frequenz beeinflusst einerseits wie gut die Funkwellen Wände durchdringen können und andererseits auch ob es sich um ein lizenziertes Frequenzband handelt.

Allgemein können Funkwellen besser Wände durchdringen je niedriger die Frequenz ist. Technologien wie WLAN, welche im Bereich von 2,4 GHz und 5 GHz senden, eignen sich nur begrenzt, um durch Wände hindurch zu kommunizieren. Die meisten Einsatzzwecke für Narrowband-Technologien müssen durch Wände hindurch kommunizieren. Daher verwenden diese auch Frequenzbänder, welche zumindest in begrenztem Maße Wände durchdringen können. Eine Empfehlung ist zum Beispiel die Verwendung von Frequenzen unter 2 GHz für NB-IoT.

Des Weiteren gibt es noch lizenzierte und unlizenzierte Frequenzbänder. Lizenzierte Frequenzbänder stehen exklusiv einem Nutzer zur Verfügung (dem Lizenzinhaber). Der Lizenzinhaber ist der Einzige, welcher in diesem Frequenzband senden darf. Folglich kann der Betreiber die Interferenz, welche andere Geräte in dem gleichen Frequenzband senden, selbst verwalten. Dies bedeutet nicht, dass Interferenzen ignoriert werden können, aber gibt dem Betreiber die Sicherheit, dass Interferenzen nur von anderen Geräten unter seiner Kontrolle entstehen können. Dadurch kann das Interferenzmanagement vereinfacht werden. NB-IoT verwendet meist lizenzierte Frequenzbänder.

Ein unlizenziertes Frequenzband kann (unter gewissen Bedingungen) von jedem verwendet werden. Dadurch entfallen die (hohen) Kosten zur exklusiven Nutzung des Frequenzbandes, im Gegenzug können aber auch andere Sender das Frequenzband nutzen. Dies hat zur Folge, dass die zur Verfügung stehende Datenrate und Latenz schwer vorher gesagt werden kann, da unbekannt ist, ob und welche anderen Sender das gleiche Frequenzband nutzen. Um diese Effekte eines einzelnen Senders gering zu halten sind Einschränkungen an die Nutzungsdauer in unlizenzierten Frequenzändern üblich. So unterliegt zum Beispiel der Auslastungsgrad für die Frequenzbänder im Bereich 863 MHz bis 870 MHz Beschränkungen zwischen 0,1% und 10%.[3] Das für Narrowband-Kommunikation am häufigsten verwendete Frequenzband ist das bereits erwähnte Band um 868 MHz in Europa und 915 MHz in den USA. Sowohl LoRa als auch Sigfox verwenden diese Frequenzbänder.

[9] https://standards.ieee.org/findstds/standard/802.11ah-2016.html
[10] http://www.iis.fraunhofer.de/mioty
[11] http://www.nwave.io
[12] http://waviot.com
[13] http://www.dash7-alliance.org
[14] https://www.ingenu.com

2.5.2 Infrastruktur

Die Geräte, welche eine Funktechnologie verwenden, können untereinander kommunizieren, aber im Internet der Dinge sind Einsatzzwecke häufiger, bei denen die Geräte mit einem zentralen Server kommunizieren. Um dies zu ermöglichen ist eine Netzinfrastruktur nötig, welche die Signale der Endgeräte empfängt und über ein (meist kabelgebundenes) Netz zu einem Server weiterleitet. Für die Organisation dieses Netzes gibt es verschiedene Möglichkeiten.

Die nötige Infrastruktur kann vom Nutzer der Endgeräte selbst verwaltet werden. Dazu müssen Access Points so aufgestellt werden, dass ein Empfang im benötigten Gebiet gewährleistet ist. Die Access Points müssen dann (über Kabel oder andere drahtlose Technologien) an das Internet angeschlossen werden. Im einfachsten Fall eines Access Points ist der Aufbau ähnlich zu einem privaten WLAN (nur mit größerer Reichweite). LoRa bietet die Möglichkeit selbst eine solche Infrastruktur zu betreiben.

Wenn das Versorgen einer größeren Fläche notwendig ist (Stadt?, Land, Kontinent) ist es nicht mehr sinnvoll diese Infrastruktur selbst zu betreiben. Hierzu gibt es Anbieter, die die Infrastruktur betreiben und (üblicherweise gegen Bezahlung) dem Endnutzer zur Verfügung stellen. Der Endnutzer muss sich in diesem Fall nur um die Endgeräte kümmern, solange er im Empfangsbereich des Netzbetreibers bleibt. Dieses Model ist aus dem Mobilfunk bekannt, bei dem jeder Nutzer einen Vertrag (oder eine Prepaidkarte) bei einem Mobilfunkbetreiber hat. Im Bereich der Narrowband-Kommunikation wird dieses Model in NB-IoT von eben diesen Anbietern umgesetzt. Außerdem gibt es Anbieter die LoRa nutzen, um eine solches Netz zu betreiben[15] und kooperative Initiativen, die gemeinsam ein solches Netz aufbauen und betreiben. Ein Beispiel für eine solche Initiative ist „The things network“[16].

Ein Spezialfall für eine Infrastruktur, die von einem Anbieter betrieben wird, ist ein Anbieter, der ein Monopol auf den Betrieb einer solchen Infrastruktur hat. Die Monopolstruktur erlaubt einen schnellen Aufbau der Technologie und eine einheitliche Nutzung. Allerdings ist im Falle eines Monopols ein Wechsel des Anbieters nicht möglich. Bei mehreren konkurrierenden Betreibern ist ein Wechseln des Anbieters möglich ohne die Technologie zu wechseln. Sigfox ist eine Technologie, bei der der Anbieter eine Monopolstellung hat.

Tabelle 2.1 zeigt eine Einordnung der betrachteten Technologien in die bereits erläuterten Kategorien.

Tabelle 2.1: Einordnung der Schmalband-Technologien

		Frequenzband	
		Lizenziert	Unlizenziert
Architektur	Von Anbieter	NB-IoT	Sigfox
	Selbst betrieben	-	LoRa

2.5.3 Andere Vergleiche

Andere Technologievergleiche haben ihren Fokus auf anderen Schwerpunkten oder Vergleichen andere Technologien miteinander. Wir listen diese hier als Startpunkt um andere Blickwinkel einholen zu können auf:

- `https://www.link-labs.com/blog/nb-iot-vs-lora-vs-sigfox`
- `https://www.iotforall.com/iot-connectivity-comparison-lora-sigfox-rpma-lpwan-technologies/`
- `https://www.link-labs.com/blog/complete-list-iot-network-protocols`

[15] zum Beispiel `https://loriot.io` und `https://www.actility.com`
[16] `https://www.thethingsnetwork.org`

2.6 Handlungsempfehlung

Da die Auswahl einer geeigneten Schmalband-Technologie sich als schwierig gestalten kann, haben wir ein Online-Werkzeug erstellt, dass dabei unterstützen kann. Das Tool besteht aus einem Fragebogen mit Fragen zum geplanten Einsatzzweck von drahtlosen Kommunikationstechnologie. Der kurze Fragebogen hilft bei der Auswahl einer Kommunikationstechnologie, um Sensordaten drahtlos zu übertragen. Er sollte nicht als alleiniges Auswahlkriterium benutzt werden, sondern dient dazu eine erste Übersicht zu erhalten. Der Fragebogen geht davon aus, dass Sensordaten von verteilen Sensoren zu einem zentralen Ort übertragen werden sollen und von dort aus weiterverarbeitet oder visualisiert werden sollen. Wenn Sie andere Anforderungen haben, setzten Sie sich bitte direkt mit uns in Verbindung. Egal ob Sie diese oder ein anderes Kommunikationsproblem lösen möchten, sind wir gerne persönlich bei der Lösung behilflich.

Das Werkzeug ist erreichbar unter:

`https://srfg.at/nb-recommender`

2.6.1 Fragen

Die zu beantwortenden Fragen lauten (in der aktuellen Version):

- Wie oft müssen Daten übertragen werden?
- Wie viele Daten müssen pro Übertragung gesendet werden?
- Welche Verzögerung ist bei der Übertragung von Daten verkraftbar?
- Welche Auswirkungen haben verlorene Pakete?
- Wie lange müssen die Endgeräte ohne Batteriewechsel auskommen?
- Von wo sollen die Sensoren Daten versenden?
- Mit welchen Erschwerungen muss die Kommunikationstechnologie umgehen können?
- Wie viele Sensoren befinden sich höchstens auf einer Fläche von $100\,\mathrm{m}^2$?
- Möchten Sie die benötigte Infrastruktur selbst betreiben?
- Wie wichtig ist Ihnen Unabhängigkeit von anderen Nutzern ähnlicher Technologien in der Nähe?
- Wann möchten Sie die Technologie betreiben?
- Welche laufenden Kosten sind pro Endgerät vertretbar?

2.6.2 Auswertung

Die Auswertung erfolgt in Form einer visuellen Darstellung der Eignung der jeweiligen Technologien (Abbildung 2.1). Diese Ausgabe wird nach der Beantwortung jeder Frage aktualisiert, so dass es möglich ist ein Gefühl dafür zu entwickeln, wie sehr eine gegebene Antwort die Empfehlung für eine Technologie beeinflusst. Die Auswertung sollte als erste Tendenz zur Auswahl einer Technologie genutzt werden und nicht zur alleinigen Entscheidung, da die zugrunde liegenden Inhalte selten pauschale Aussage erlauben.

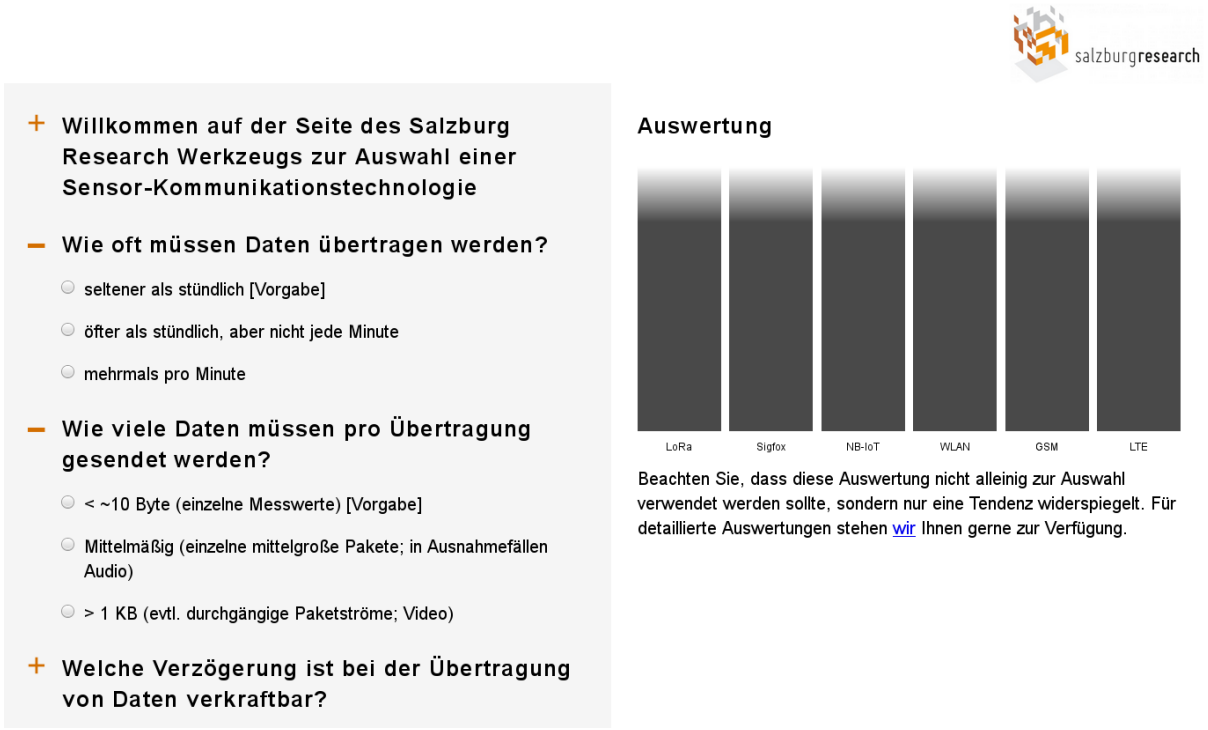


Abbildung 2.1: Das Werkzeug zur Auswertung hilft bei der Auswahl einer Technologie und ist im Internet unter `https://srfg.at/nb-recommender` abrufbar.

3 Innovationen

Datenübertragung mit Schmalband-Technologien versprechen vielfältige innovative Einsatzmöglichkeiten im Bereich der Maschine-zu-Maschine Kommunikation. In diesem Abschnitt werden Anwendungsfälle und Innovationspotentiale beschrieben, die aufgrund verschiedener Faktoren (Wirtschaftsstruktur, geografische Gegebenheiten) für Unternehmen, Verwaltung und Bevölkerung im Bundesland Salzburg von besonderer Bedeutung sind.

Obwohl für einige der Anwendungsfälle eine Schmalband-Technologie besser geeignet sein kann als eine andere (z. B. auf Grund des Wunsches oder der Notwendigkeit eine private Schmalband-Infrastruktur zu betreiben), werden die Technologien im folgenden einheitlich betrachtet, es sei denn eine spezifische Eigenschaft einer Technologie ist entscheidend für die Umsetzung des Anwendungsfalls.

Der nächste Abschnitt gibt generell über die Eigenschaften, Einsatzmöglichkeiten und Voraussetzungen von Schmalband-Datenübertragungen im Kontext des Internets der Dinge und maschineller Kommunikation Auskunft. Anschließend (Abschnitt 3.2) werden innovative Anwendungsbeispiele in unterschiedlichen Bereichen aufgezeigt - mit dem oben genannten Fokus auf im Land Salzburg vorhandene Gegebenheiten und Strukturen. Der schließende Abschnitt 3.3 greift Aspekte auf die im Rahmen von Forschungs- und Entwicklungstätigkeiten die Anwendbarkeit von Schmalband-Technologien weiter ausdehnen und verbessern können. Für einige Anwendungsfelder müssen hier noch wichtige Fragen geklärt werden, um zu bewerten, ob der Einsatz von Schmalband-Technologie möglich oder sinnvoll ist.

3.1 Schmalband-Technologie und das Internet der Dinge

Die Entwicklung von Schmalband-Kommunikationstechnologien wurden hauptsächlich durch die Anforderungen der Maschine-zu-Maschine Kommunikation beeinflusst und getrieben. Maschine-zu-Maschine Kommunikation bezeichnet den automatischen Informationsaustausch zwischen unterschiedlichen Systemen wie Industriemaschinen, Fahrzeugen, Containern, Lagersystemen oder Überwachungssystemen. Zweck ist eine weitgehende Automatisierung von Steuerungs-, Überwachungs-, und Wartungstätigkeiten. Notwendig für diese Entwicklung ist der massive zusätzliche Einsatz von vernetzten Sensoren und Aktoren in und um die Systeme und Maschinen. Diese zunehmende Vernetzung ist damit auch ein Teil des Internet der Dinge.

Allgemein wird bei der Entwicklung der Maschine-zu-Maschine Kommunikation und dem Internet der Dinge angenommen, dass Bandbreitenbedarf und Bandbreitennutzung sich maßgeblich zur bisherigen stark auf menschliche Benutzer ausgelegten Internetnutzung ändern werden. Statt hochauflösende Videodaten und extrem niedrige Latenzen für menschliche Interaktionen werden computerlesbare und komprimierte Binärdaten mit niedrigen Volumen versendet. Weiterhin verlangen die derzeitig berücksichtigten Anwendungsszenarien Nachrichtenübermittlung nur bei Auftreten von bestimmten Events oder in vergleichsweise großen, mehrere Minuten oder gar Stunden umfassenden, zeitlichen Abständen. Aus diesen Anforderungen wurden die Schmalband-Übertragungstechnologien als erheblich kostengünstigere Alternative zu etablierten (breitbandigen) Mobilfunktechnologien wie LTE entwickelt. Die Schmalbandübertragung hat darüber hinaus die Eigenschaft Nutzdaten über große Distanzen übertragen zu können und Wände besser durchdringen zu können. Dies macht Schmalband-Technologie vorteilhaft für die Übertragung von Daten an mobilfunkmäßig sonst schlecht versorgten und unzugänglichen Orten.

Abbildung 3.1 zeigt die Einordnung der schmalbandigen Low-Power-Wide-Area (LPWA) Funk-

übertragungstechnologien bezüglich verfügbarer Bandbreite und Reichweite im Vergleich zu anderen verbreiteten Funktechnologien.

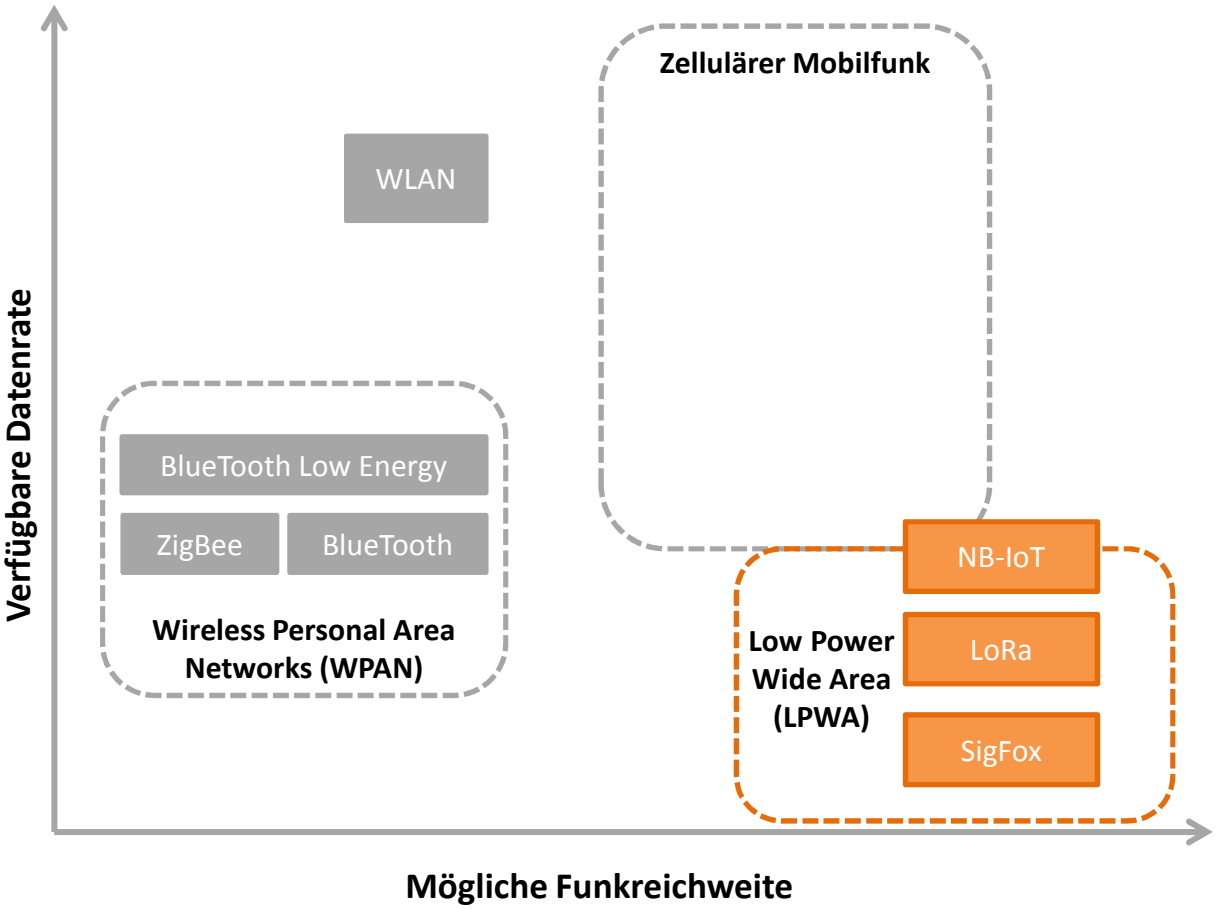


Abbildung 3.1: Einordnung der Schmalband-Technologie im Vergleich zu anderen Datenübertragungsstandards

Da viele der angedachten Anwendungsszenarien nur durch den flächenmäßig und zahlenmäßig großen Einsatz von Sensoren umsetzbar sind, hängt die Umsetzbarkeit dieser auch stark von Kosten, Komplexität und technischer Reife der Sensoren ab. Tabelle 3.1 enthält eine Übersicht verschiedener Sensortypen sowie deren Ausgereiftheit und Skalierbarkeit (d.h. massenhaften Einsatz auf Grund von Größe und Preis). Die Tabelle bezieht sich auf Daten aus dem Jahr 2009 und wird im Detail nicht mehr dem aktuellen Stand der Technik entsprechen, trotzdem kann sie Anhaltspunkte liefern.

3.2 Innovative Anwendungsszenarien für Schmalband-Technologie

3.2.1 Überwachung ausgedehnter Infrastrukturen

Eine spezifische Charakteristik weißt das Land Salzburg durch dessen geologische Einteilung in alpines Gelände und Alpenvorland auf, wobei der alpine Teil der flächenmäßig größere ist. Insbesondere in den alpinen Gebieten ergibt sich daraus eine weitflächig sehr dünne Besiedlung bestehend aus einzelnen Dörfern, Weilern oder Einzelhöfen und einer daraus resultierenden Bevölkerungsdichte von 75 Personen pro km^2. Durch die notwendige Anbindung dieser Bebauungen an öffentliche Infrastrukturen entstehen in Salzburg ausgedehnte Netze (Abwasser, Gas, Strom, Telefon) in teils schwer zugänglichen Lagen.

Besonders Telefonleitungen und Stromleitungen sind dabei oft als potentiell störungsanfälligere

Tabelle 3.1: Klassifzierung von Sensoren bezüglich technischer Reife und Skalierbarkeit (aus: [2])

Kategorie	Parameter	Technische Reife	Skalierbarkeit
Physikalisch	Temperatur	Hoch	Hoch
	Feuchtigkeitsgehalt	Hoch	Hoch
	Flussraten u. Geschwindigkeit	Hoch	Mittel-Hoch
	Druck	Hoch	Hoch
	Lichtdurchlässigkeit	Hoch	Hoch
	Geschwindigkeit u. Richtung	Hoch	Hoch
Chemisch	Gelöster Sauerstoff	Hoch	Hoch
	Elektrische Leitfäigkeit	Hoch	Hoch
	pH	Hoch	Hoch
	Oxydations Reduktionspotential	Mittel	Hoch
	Ionische Hauptgruppen (Cl-, Na+)	Niedrig-Mittel	Hoch
	Nitrate	Niedrig-Mittel	Niedrig-Hoch
	Schwermetalle	Niedrig	Niedrig
	Organische Verbindungen	Niedrig	Niedrig
Biologisch	Mikroorganismen	Niedrig	Niedrig
	Biologische aktive Vereinigungen	Niedrig	Niedrig

Luftkabel bzw. Freileitungen ausgeführt. In Zahlen ausgedrückt bedeutet das für das Land Salzburg eine Gesamtlänge von Freileitungen im Mittel, Niedrig- Hochspannungssegment von ca. 3000 km gegenüber einer Gesamtlänge von Erdkabeln von ca. 13800 km. Dies entspricht einem Anteil von Erdkabeln von 83% (Verkabelungsgrad).[1] Für Telefonleitungen sind exakte Zahlen nicht verfügbar.

Eine weitere großflächige Infrastruktur im Land Salzburg ist durch das Gasverteilungsnetz gegeben, welches eine Gesamtlänge im Hochdruck-, Mitteldruck- und Niederdruck-Segment von etwas weniger als 2000 km aufweist.[2] Ein Großteil der Leitungen im Salzburger Landesgebiet befinden sich im engbesiedelten städtischen Bereich. Es existiert aber auch eine Gasleitung entlang der Salzach bis nach Saalfelden und soll zukünftig durch eine Verbindung nach Hochfilzen bis nach Tirol verlängert werden. Gasleitungen werden üblicherweise unterirdisch geführt und sind schwer erreichbar (insbesondere im städtischen Bereich). Auch bei Wasser und Abwassersystemen handelt es sich um Infrastrukturen größeren Umfangs und schwierigerer Zugangsmöglichkeiten.

Insbesondere die Wartung dieser Infrastrukturen ist durch gegebenenfalls nötige Grabungen, streckenweises Sperren und Begehungen zeit- und kostenintensiv. Der unterstützende Einsatz von Sensoren und Aktoren bildet hier die Grundlage für Automatisierung und Reduktion der Wartungsintervalle. Schmalband-Kommunikationstechnologien mit ihren spezifischen Eigenschaften Robustheit, Reichweite und Langlebigkeit durch Energieeffizienz können viele Anwendungen in diesen Bereichen möglich machen und wurden in verschiedenen Projekten auch bereits umgesetzt. Beispiele dafür sind die Überwachung von Dehnung und Temperatur von Freileitungen im Energieversorgungsnetz, um die Effizienz der Ausnutzung zu steigern. Die aktuelle Last einer Freileitung wird dabei anhand ihrer Temperatur und Dehnung und die Gesamtheit der Temperatur- und Dehnungsdaten eines Freileitungsnetzes kann dazu benutzt werden, die Durchflusssteuerung eines Stromnetzes dahingehend zu optimieren, dass eine gleichmäßige Belastung der Freileitungen im gesamten Netz stattfindet, ohne einzelne Leitungen zu überlasten und ggf. zu beschädigen.[3]

Durch den Einsatz von Sensoren in Gasleitungssystemen können Veränderungen im Gasdruck und Gasströmung erkannt werden, die auf strukturelle Veränderungen wie Leckagen oder Verengungen durch Ablagerungen schließen lassen. Diese Daten können dazu verwendet werden, Wartungs- und

[1] https://www.salzburgnetz.at/de_at/Stromnetz/netzstruktur.html
[2] https://www.salzburgnetz.at/de_at/Erdgasnetz/netzstruktur.html
[3] https://www.microtronics.at/de/referenzen/micca.html

Überprüfungsarbeiten, wie das sogenannte „Molchen“ zielgerichteter und bedarfsgerechter durchzuführen.

Ein weiterer Anwendungsfall besteht bei den Druckregelungen des Gasnetzes: Während Schieber und Reduzierstationen entlang der Hochdruckleitungen (16 bzw. 70 bar) bereits flächendeckend mit automatischen Überwachungsstationen ausgestattet sind, die selbstständig Alarm schlagen, sobald Störungen auftreten, ist dies bei den Hausdruckreglern der Netzkunden nicht der Fall.[4] Diese werden üblicherweise regelmäßig durch Wartungstechniker überprüft. Günstige Sensoren und Kommunikationstechnologien können auch hier für eine bedarfsgerechtere Überprüfung sorgen.

Ähnliche Anwendungsfälle lassen sich schließlich auch für Wasserver- und entsorgungsanlagen finden in dem die dort vorhandenen Pumpen, Rohrleitungen, Schleusen und Ventile mit Sensoren ausgestattet werden, die über Schmalband-Technologien selbstständig Status- und Störungsmeldungen versenden können. Ein Beispiel dafür ist die Übermittlung einer Störungsmeldung durch eine Pumpe, die in einem abgelegenen Trinkwasserbrunnen (Abb. 3.2) eingesetzt wird.

3.2.2 Steuerung und Überwachung industrieller Prozesse

Ähnliche Anwendungsfälle wie bei den ausgedehnten Infrastrukturen lassen sich auch bei der Überwachung, Steuerung und Wartung von industriellen Prozessen finden. Während großflächige Infrastrukturen von großen Reichweiten und niedrigen Energiehaushalt der Schmalband-Technologien profitieren, stehen beim Einsatz in Industrieanalgen mehr die größere Störunanfälligkeit der eingesetzten Funktechnologien sowie im Falle von LoRaWAN, die Möglichkeit zum Aufbau von privaten Kommunikationsinfrastrukturen zum Beispiel auf einem Werksgelände eine Rolle.

Gemessen an Umsatz und Arbeitsplätzen stellen Nahrungsmittelerzeuger (4 Mrd. Euro Umsatz; etwa 6000 Beschäftigte) und Maschinenbauunternehmen (Im Jahr 2013: 1,5 Mrd. Euro Umsatz; etwa 4500 Beschäftigte) die größten Industriezweige Salzburgs dar (Quelle: Salzburger Industriellen Vereinigung, ïv-factsheet 2016"[5]). Diese Bereiche können darüber hinaus auch stark von den Entwicklungen der Maschine-zu-Maschine Kommunikation und damit auch vom Einsatz von Schmalband-Technologien profitieren.

Anwendungsbeispiele in der Lebensmittelindustrie umfassen dabei die Funktionsüberwachung und Steuerung von Pumpen und Ventilen sowie Leckageerkennung in Tanks und Leitungen, die in aller Regel nur schwer von außen erreichbar sind. Insbesondere für den Einsatz von Schmalband Kommunikationstechnologie ergeben sich hier Einsatzmöglichkeiten, wenn Messgeräte beispielsweise ins Innere von Tanks eingebracht werden, um Ablagerungen zu erkennen. Die per Schmalband-Technologie ausgestatteten Messgeräte melden bei Überschreitung eines Grenzwertes diesen an das zentrale Überwachungssystem und geben einen Hinweis auf eine notwendige Tankreinigung. Werden Wartungsarbeiten wie Tankreinigungen turnusmäßig oder erst nachdem eine Qualitätsverschlechterung eines (Teil-)Erzeugnisses festgestellt wurde durchgeführt, ermöglicht das Einbringen eines Sensors bedarfsgerechtere Wartungsintervalle mit geringeren Stillstandszeiten und Produktionsausschuss. Schmalband-Technologie profitiert, wie in diesem Beispiel bei der Datenübertragung aus einem Tank, insbesondere durch die höhere Störungsunempfindlichkeit des Signals und seine Energieeffizienz, welche Batterielaufzeiten eines Senders im Bereich mehrerer Jahre erlauben und damit auch ohne externe Stromzufuhr auskommen.

Innovationen im Maschinenbau geschehen in den letzten Jahren häufig durch bessere Einbindung einzelner Maschinen in die Fertigungsprozesse und engmaschigeren Informationsaustausch zwischen Maschinen untereinander und zwischen Maschinen und Produktionssteuerungssystemen. Ein besonderer Schwerpunkt liegt dabei häufig auf der Verringerung von Stillstandszeiten durch Wartung oder Umrüstung von Maschinen insbesondere durch Optimierung der Produktionsabläufe sowie vorausschauende und reaktive Wartung. Dies trifft in besonderem Maße auf Produktionsmaschinen in Fertigungshallen zu. In diesem Umfeld sind Maschinen üblicherweise in Produktionsnetzen zusammen-

[4] https://www.salzburgnetz.at/de_at/Erdgasnetz/netzstruktur.html
[5] http://www.die-salzburger-industrie.at/wp-content/uploads/2016/09/Salzburg-aktuell-2016.pdf

geschaltet, die im Vergleich zu Schmalband-Technologien hohe Bandbreiten bieten und auch höhere Zuverlässigkeit bieten - wie es für den störungsfreien Fertigungsbetrieb notwendig ist. Nicht zuletzt aus diesem Grund kann dem Einsatz von Schmalband-Technologien innerhalb von Fertigungsanlagen nur eine begrenzte Bedeutung beigemessen werden.

(a) Hochbehälter

(b) Trinkwasserbrunnen

Abbildung 3.2: Trinkwasserversorgung Listsee bei Bad Reichenhall

Ein anderes Bild zeigt sich jedoch insbesondere für Maschinen, die in abgelegenen Gebieten installiert sind, selbst mobil sind oder mobil in ihre Einsatzgebiete verlegt werden können. Insbesondere wenn die Maschinen in mobilfunktechnisch schlecht versorgten Gebieten eingesetzt werden bietet Schmalband-Technologie die Möglichkeiten Betriebs- und Nutzungsdaten einer Maschine sowie Ereignisbenachrichtigungen an Nutzer, Eigentümer oder Hersteller zu übermitteln. Berücksichtigt werden muss hier die beschränkte Bandbreite und die Begrenzung auf wenige Nachrichten pro Tag. Ein konkreter Anwendungsfall wäre die tägliche Übermittlung von Betriebsstunden einer verliehenen Maschine an den Verleiher der Maschine für die nutzungsgenaue Abrechnung.

3.2.3 Umwelt- und Geländeüberwachung; Monitoring landwirtschaftlicher Flächen

Ein weiteres Anwendungsfeld von Schmalband-Technologien existiert im Bereich der Umwelt- und Geländeüberwachung. Insbesondere die fortlaufende Überwachung in bergigem und alpinem Gelände erhöht die Chancen schon die Vorboten von Erdrutschen, Lawinen und Murengängen zu erkennen und entsprechend Verhinderungsmaßnahmen zu treffen. In einem stark integriertem System sind Anwendungen denkbar, die an besonders gefährdeten Stellen nach Erkennung einer Geländebewegung automatisiert Straßensperrungen vornehmen und/oder Alarmierungen auslösen.

Ein Anwendungsbeispiel stellt die Lawinenbeurteilung und -sicherung in gefährdeten Gebieten dar. Üblicherweise werden Lawinenkommissionen eingerichtet, die regelmäßig die Lawinengefährdung durch die Zusammenführung von Wetterdaten (Wind, Temperatur, Niederschlag) sowie Schneedeckenuntersuchungen (z. B. durch Testsprengungen) und Beobachtungen im Gelände (z. B. Verwehungen) feststellt. Auch werden automatische Schnee- und Windstationen als Datenquellen herangezogen. Als Ergänzung zu den etablierten Maßnahmen sowie um eine genauere Datenbasis zu erhalten könnten zukünftig preisgünstige Sensoren vorhandene Messsysteme erweitern, um zeitaufwendige und gefährliche Begehungen oder Helikoptereinsätze einzuschränken. Um die Sensordaten den Untersuchungen zur Verfügung zu stellen wird in diesem Szenario ein Kommunikationssystem benötigt, welches eine größere Anzahl von Sensorknoten in einem bestimmten Gebiet anbinden kann und darüber hinaus robuster gegen Witterungsverhältnisse ist (insbesondere in der Lage ist durch Schneedecken hindurch zu kommunizieren).

Hochwasserschutz bietet Potential mit Hilfe von entsprechender Sensorik und Schmalband-Technologie Prognoseverfahren zu optimieren und den Hochwasserverlauf besser abzuschätzen. Hierbei können die vorhandenen Pegelwerte mit zusätzlichen, ggf. mobilen und autark arbeitenden Messeinheiten ergänzt werden um auf diese Weise Niederschlagsdaten oder Zuflussmengen zu erfassen.

Innovationen durch Geländeüberwachung werden auch im Bereich der Landwirtschaft gesehen. Sensoren in Boden oder an Pflanzen geben Auskunft über Bodenbeschaffenheit, Feuchtigkeit oder Reifegrad der Pflanzen. Ziel ist die Bewirtschaftung landwirtschaftlicher Flächen beginnend bei Aussaht, Düngung und Bewässerung bis zur Ernte zu optimieren. Der Digitalisierung der Landwirtschaft wird in Studien ein Wachstum von jährlich 12% (Quelle: Deutscher Bauernverband nach Studie von Roland Berger, Situationsbericht 2015/2016[6]). In diesem Zusammenhang wird die Anzahl von vernetzten Komponenten in der Landwirtschaft ebenfalls stark anwachsen.

Die Eigenschaften der Schmalband-Technologie kommen diesen Anforderungen entgegen. Der Fokus von Schmalband-Technologie auf Energieeffizienz ermöglicht es dabei Lösungen zu entwickeln, die über mehrere Jahre aktiv bleiben. In der Gelände- und Umweltüberwachung spielen zusätzlich die Kosten für Sensoren und Kommunikation eine wichtige Rolle. Sensorknoten sind im Allgemeinen unbewacht und harschen Umweltbedingungen ausgesetzt, die zu Verlust oder Zerstörung der Knoten führen können. Weiterhin kommt hinzu, dass die Sensoren in den genannten Anwendungsfällen auch abseits mobilfunktechnisch gut versorgter Gebiete eingesetzt werden und damit große Reichweiten erforderlich machen. Sind Sensorknoten im Boden vergraben, überschwemmt oder von einer Schneedecke verdeckt wirken sich die verbesserten Durchdringungseigenschaften (z. B. im Vergleich zu 4G) der Schmalband-Technologie positiv aus. Für die konkrete Umsetzung solcher Anwendungsszenarien sind dennoch wichtige Fragen ungeklärt – beispielsweise nach den tatsächlichen Einflüssen verschiedener Witterungsverhältnisse auf die Kommunikationsqualität oder der optimalen Verteilung der Basisstationen im Messgebiet.

3.2.4 Lokalisierung und Bewegungsüberwachung von transportablen Gegenständen

Im Anwendungsfeld Lokalisierung werden Sensorknoten eingesetzt um den gegenwärtigen Standort von Geräten und Gegenständen zu bestimmen. Die Sensorknoten bestehen aus einem zum Beispiel satellitengestützten Lokalisierungssensor (GPS) und einer Kommunikationsschnittstelle um in regelmäßigen Abständen oder bei Erkennung einer Bewegung die Position an einen Server zu übermitteln. Konkrete Anwendungen in diesem Bereich umfassen die Lokalisierung von Gerätschaften auf einem Werksgelände wie der Position eines transportablen Notfallgenerators. In erweiterten Szenarien können zusammen mit der Lokalisierungsinformation auch Zustandsinformationen wie der Tankfüllstand des Generators übermittelt werden.

Der Einsatz von Schmalband-Technologie ermöglicht es insbesondere eine kosteneffiziente Lösung zur Lokalisierung in großflächigen Gebieten wie Umschlagplätzen für Container oder Güterbahnhöfen zu erreichen. Da die Schmalband-Technologie jedoch keine ausgesprochen hohe Zuverlässigkeit der Datenübertragung bieten kann - d. h. Pakete häufig verloren gehen können und zum Beispiel auf Grund von Störungen auch längere Zeit eine Datenübermittlung erfolglos bleibt - und zusätzlich üblicherweise nur wenige Nachrichten pro Tag und Sensorknoten versendet werden können, müssen die Randbedingungen solcher Anwendungen genau bestimmt werden. Beispielsweise machen Anwendungen zur Lokalisierung von Gegenständen nur dann Sinn, wenn die Abfrage der Position eines Gegenstandes nur selten (bis zu einigen Malen täglich), beispielsweise bei der Verladung eines Containers durchgeführt wird. Um die Ortung eines Gegenstands zu erlauben, muss zudem sichergestellt werden, dass der Gegenstand nicht in ein Gebiet außerhalb der Reichweite des Schmalband-Kommunikationssystems gelangt.

Während sich die Schmalband-Technologie also eher für die Lokalisierung von Gegenständen eignet, die nur selten ihre Position ändern bzw. deren aktuelle Position nur selten benötigt wird, eignet sich die Technologie keinesfalls für Anwendungen für das dauerhafte Überwachen von Bewegungen in Echtzeit. Bewegungsüberwachung von Gegenständen kann sinnvoll nur dann geschehen, wenn diese auf dem Sensorknoten zwischengespeichert werden um dann zu einem späteren Zeitpunkt gesammelt übertragen zu werden. In Anwendungen können diese Daten dann dazu verwendet werden lückenlos nachvollziehbar die Einhaltung von SLAs für Transport und Lagerung nachzuweisen – zum Beispiel

[6] `http://www.bauernverband.de/36-digitalisierung-in-der-landwirtschaft`

durch Schockerkennung; auch in Kombination mit Zustandsdaten wie Temperatur und Feuchtigkeit). Ein prinzipiell ähnlicher Anwendungsfall dafür wurde für das Bewegungstracking von Nutz- und Wildtieren durchgeführt, indem Bewegungsdaten der Tiere verwendet wurden, um Rückschlüsse auf die gesundheitliche Verfassung der Tiere zu ziehen[7]

Lokalisierung und Bewegungserkennung erlauben Anwendungen bei denen geografische Bereiche durch Sensoren überwacht werden (genannt Geo-fencing). Geo-fencing-Anwendungen lassen sich dabei in zwei unterschiedlichen Varianten umsetzen. Einerseits in dem Bewegungssensoren im überwachten Gebiet fest installiert werden, Bewegungen aufzeichnen und schließlich die gemessenen Bewegungsdaten übermitteln und andererseits indem Lokalisierungssensoren selbst bewegt werden, ihre Position selbst mitverfolgen und das Erreichen einer Position oder Grenze selbst feststellen. Anwendungsbeispiele dazu sind die Zugangsüberwachung, bei der durch Bewegungssensoren der Zeitpunkt aufgezeichnet wird zu dem Personen ein überwachtes Gebiet betreten oder die Sicherstellung bzw. Überprüfung, dass eine Maschine nur in einem festgelegten Bereich verwendet wird. Auch kann automatisiert überprüft werden, ob Mietfahrzeuge zeitgerecht wieder zurückgebracht wurden – zum Beispiel bei vereinbarter Mietwagenrückgabe außerhalb der Öffnungszeiten der Verleihstation.

Der Einsatz von Schmalband-Technologie eignet sich für Geo-fencing-Anwendungen insbesondere dann, wenn keine ausreichende Stromversorgung vorhanden ist und eine mehrjährig wartungsfreie Betriebsdauer des Systems gefordert ist, große Kommunikationsreichweite notwendig ist und die Kosten der Datenkommunikation gering sein müssen. Darüber hinaus kann Schmalband-Technologie für konkrete Anwendungsfälle nur dann verwendet werden, wenn eine Alarmierung in Echtzeit nicht notwendig ist und Bewegungsinformationen verloren gehen dürfen oder es ausreichend ist, diese auf dem Sensorknoten zumindest so lange zu speichern bis die Informationen bestätigt übertragen worden sind.

3.2.5 Anwendungsszenarios Gebäudeautomatisierung und Smart City

Ein weiteres innovatives Anwendungsfeld für Schmalband-Technologie findet sich in den Bereichen der Gebäudeautomatisierung und Smart City. Anwendungen in diesen Bereichen erlauben insbesondere effizientere Organisation, bedarfsgerechte Maßnahmen und verbesserte Sicherheit.

Im Bereich der Gebäudeautomatisierung erlaubt der Einsatz von Aktoren und Sensoren die automatische Steuerung von Klimageräten oder der Gebäudebeleuchtung. Dies wird häufig angewendet um den Energieverbrauch der Gebäude zu reduzieren. Üblicherweise erfordern diese Anwendungen die Kommunikation der Sensoren, Aktoren und entsprechender Geräte mit einem zentralen Gebäudemanagementsystem. In Anwendungsszenarien verwendet ein Gebäudemanagementsystem Raumtemperatursensoren und Sonneneinstrahlungssensoren auf der Gebäudeaußenseite um die Innenraumtemperatur durch die Klimaanlage und die Sonnenschutz-Jalousien zu steuern. Die Innenraumtemperatur wird also energieeffizient geregelt.

Die Verwendung von Schmalband-Kommunikation bietet sich insbesondere deshalb an, weil die Funksignalausbreitung in den verwendeten Funkbändern innerhalb von Gebäuden weniger störanfällig ist als beispielsweise WLAN – Wände und Decken werden von den Signalen leichter durchdrungen. Zusätzlich sind die Verbindungskosten bei Schmalband-Technologien geringer, was insbesondere die nachträgliche Ausrüstung von Gebäuden kosteneffizient ermöglicht. Jedoch existieren für den Bereich der Gebäudeautomatisierung eine Reihe alternativer Kommunikationstechnologien wie beispielsweise Powerline Communication (PLC). Die Datenübertragung über das Stromnetz eines Gebäudes hat dabei den Vorteil, dass die Kommunikation genau auf das Gebäude begrenzt ist – Funksignale können dagegen auch im Umfeld des Gebäudes empfangen und abgehört werden. Weiterhin besteht der Vorteil, dass PLC höhere Bandbreiten bietet.

Deutlich größeren Mehrwert bietet Schmalband-Kommunikation im Innovationsbereich Smart City. Die Anwendungsszenarien bei der Automatisierung urbaner Umwelten sind vielfältig. Eines der am häufigsten genannten Anwendungsfälle ist das Smart Metering bei dem Wasser-, Gas-, Heizungs-

[7] https://www.computerwoche.de/a/wie-digitalisierung-die-landwirtschaft-veraendert,3094201

und Stromzähler automatisiert ausgelesen werden (im Takt von beispielsweise 15 Minuten) und die Werte dann täglich gesammelt den Versorgungsunternehmen übermittelt werden. Dadurch werden Verbrauchsplanung und Rechnungslegung für Versorgungsunternehmen vereinfacht.

Eine besondere Herausforderung von Seiten der Kommunikation besteht dabei darin, dass die Zähler in der Regel in Kellern oder tiefen Winkeln innerhalb von Wohnungen installiert sind. Aktuelle Kommunikationstechnologien wie 4G bieten in diesen Bereichen keinen ausreichenden Empfang. Durch andere Übertragungsverfahren von Schmalband-Technologie ist eine höhere Durchdringung möglich.

Weitere Anwendungsszenarien im Bereich Smart City beschreiben die Füllstandsüberwachung von Mülltonnen als Wegbereiter bedarfsgerechter Müllbewirtschaftung. Mülltonnen werden dann entleert, wenn sie voll sind, die Tourenplanung der Müllfahrzeuge wird dementsprechend optimiert.[8]

Intelligente Parkraumüberwachung ermöglicht Autofahrern das einfache Auffinden von Parkplätzen über einen mobil erreichbaren Informationsdienst. Parkplätze werden mit Sensoren und Kommunikationstechnologie ausgestattet, um ihren aktuellen Belegungszustand dem Informationssystem mitzuteilen – der Belegungszustand kann über das Internet von Autofahrern abgerufen werden. Dies dient der Verringerung des Verkehrsaufkommens durch Parkplatzsuche in besonders stark frequentierten Gebieten.

Im Bereich vorausschauender und reaktiver Wartung im Bereich Smart City wird das Management der Straßenbeleuchtung und Ampelsysteme gesehen. Straßenlaternen bzw. Ampeln melden durch Sensoren den Zustand der Leuchtkörper und können so rechtzeitig bereits vor einem tatsächlichen Ausfall gewartet werden.

Ein weiterer innovativer Anwendungsfall stellt die Überwachung historischer Bauwerke dar. Erschütterungssensoren oder Feuchtigkeitssensoren die nachträglich zum Beispiel in Brücken eingebracht werden erlauben die fortlaufende Zustandsüberwachung der Bauwerke und die Einleitung rechtzeitiger Wartungsmaßnahmen.

Die oben genannten Anwendungsszenarien sind besonders für den Einsatz von Schmalband-Technologie geeignet. Zunächst werden durch geringe Kommunikationskosten und verhältnismäßig einfache Installation die Nach- und Ausrüstungskosten gering gehalten. Weiterhin sind die anfallenden Datenmengen in der Regel gering genug um mit der zur Verfügung stehenden Bandbreite übertragen zu werden. Außerdem sind die Messdaten selbst unkritisch genug, so dass vereinzelt verlorengegangene Pakete keine großen Auswirkungen haben. Des Weiteren werden hohe Reichweiten und Signaldurchdringungseigenschaften für die nötige Abdeckung der weit verteilten Sensornetze benötigt. [9]

3.3 Forschungsrelevante Fragestellungen bezüglich Narrowband-basierter Anwendungen

Dieser Abschnitt behandelt einige wissenschaftliche Fragestellungen, die Fähigkeiten und Einsetzbarkeit von Schmalband-Technologien zu erweitern um neue Anwendungsfelder erschließen zu können.

3.3.1 Peer-to-Peer Kommunikation zwischen Endknoten

Narrowband-basierte Kommunikationsinfrastrukturen operieren üblicherweise in Nabe-Speiche Architekturen. Sensorknoten senden ihre Daten an einen (oder auch mehrere) zentrale Gatewayknoten, die über einen breitbandigen Uplink verfügen und die Daten den Steuerungs- bzw. Monitoringsystemen zur Verfügung stellen. So wird die LoRa Funkübertragungstechnolgie üblicherweise in einem LoRaWAN Umfeld (das die eigentlich Nabe-Speiche Architektur beschreibt) betrieben.

Daneben kann LoRa Funkübertragung auch dazu verwendet werden, andere Netztopologien aufzubauen bei denen LoRa-Knoten direkt (Peer-to-Peer) miteinander kommunizieren ohne einen Ga-

[8] https://www.t-systems.com/at/de/newsroom/blog/cloud-computing/cloud-computing/mit-der-cloud-in-eine-neue-und-smarte-welt-310240

[9] https://www.linkedin.com/pulse/building-smart-cities-lora-hussain-fakhruddin

tewayknoten einzubeziehen. In einfachen Anwendungsfällen wurde auf diese Weise ein Sensor direkt an einen Alarmgeber angebunden.[10] Werden Messdaten, wie in diesem Fall, nur von einem einzigen Empfänger benötigt, erlaubt der Verzicht auf den Gatewayknoten große Kosteneinsparungen.

Grundsätzlich ist die P2P-Kommunikation zwischen LoRa Knoten nachteilig für die Batterielaufzeit. Grund hierfür ist, dass mindestens ein LoRa Knoten einen größeren Zeitanteil in Empfangsbereitschaft sein muss um Pakete eines Nachbarknotens zu empfangen. Andererseits lassen sich auf diese Weise auch Anwendungsszenarien realisieren, die in klassischen Nabe-Speiche-Topologien aufwendiger und teurer umsetzbar sind. Ein Anwendungsszenario ist die Verwendung von günstigen LoRa Transceivern um die Kommunikationsqualität und Zuverlässigkeit in Randbereichen der Versorgung durch LoRa Gateways zu verbessern. Beispielsweise durch Verwendung eines LoRa Transceiver, der die Daten eines schlecht erreichbaren Smart Meters an das Gateway weiterleitet. Auch mobile Sensorknoten können von Peer-to-Peer Kommunikation profitieren, wenn zum Beispiel Leihfahrzeuge außerhalb der Reichweite eines Gateways geraten. Das Szenario ist in Abbildung 3.3 dargestellt.

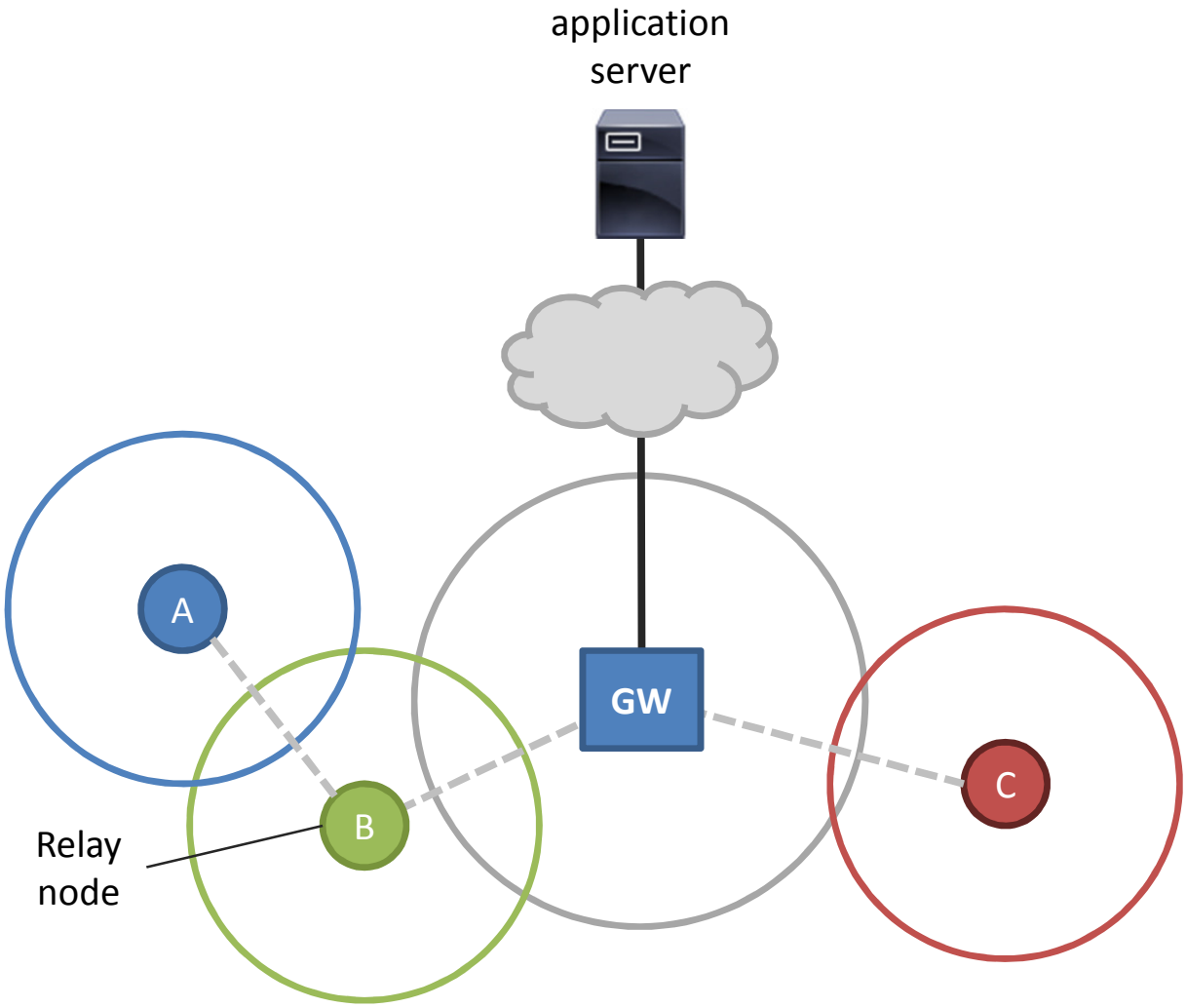


Abbildung 3.3: Datenübertragung über Relay Knoten zur Erweiterung der Abdeckung eines Schmalband-Netzwerks.

Forschungsrelevante Fragestellungen ergeben sich besonders zur (gegebenenfalls automatischen) Konfiguration solcher Peer-to-Peer Netze - zum Beispiel wie die Notwendigkeit der Peer-to-Peer Kommunikation überhaupt festgestellt werden kann und wie optimiert Relay-Knoten ausgewählt werden

[10] http://www.softhard.io/lorawan-solutions/lora-peer-peer/

(z. B. bezüglich Energieverbrauch und Lebensdauer in Anbetracht unterschiedlicher Kommunikationsszenarien). Grundlagen zu diesen Fragestellungen finden sich insbesondere in der wissenschaftlichen Diskussion über Wireless-Sensor-Networks (WSNs).

3.3.2 Multi-Technologie Kommunikationsknoten

Unter Betrachtung liegen Anwendungsmöglichkeiten und Optimierungen im Zusammenhang mit Endknoten oder Kommunikationsgateways die über mehrere Kommunikationstechnologien – beispielsweise LoRa und NB-IoT – verfügen. Im Gegenzug zu höheren Kosten für die Kommunikation und eventuell etwas höherem Energiebedarf können Multi-Technologieknoten die Effektivität von Anwendungen steigern. Je nach Anwendungsszenario, der Netzverfügbarkeit sowie der Netzauslastung verwenden Sensorknoten die besser geeignete Übertragungstechnologie.

Konkrete Fragestellungen ergeben sich nach Einflussfaktoren auf die Kommunikationsqualität von Narrowband-Technologien sowie deren Gewichtung. Wie kann die Entscheidung zur Verwendung einer Kommunikationstechnologie zum Beispiel unter Beibehaltung größtmöglicher Energieeffizienz und Lebensdauer durch Zielfunktionen ausgedrückt werden? Wie können die relevanten Einflussgrößen im operativen Betrieb im Feld bestimmt werden?

3.3.3 Rekonfiguration von Endknoten

Schmalband-Kommunikationstechnologie wird in den meisten Fällen in Anwendungsszenarien eingesetzt, die mehrjährige Laufzeiten vorsehen. Ein daraus entstehendes Problem ist, wie mit der Wartung und Rekonfiguration von Software auf Sensoren oder Aktoren umgegangen werden kann, wenn diese sich an schwer oder gar unerreichbaren Stellen befinden. Sollten während der langen Nutzungsdauer der Systeme Anforderungsänderungen bezüglich erfasster Daten (z. B. Messzeitpunkte, Messauflösung) oder auch Fehlerkorrekturen an der Software notwendig werden sind unter Umständen größere Datenübertragungen zu den Endsystemen nötig. Narrowband-Technologie bietet dafür, bedingt durch begrenzte Bandbreite und upstream-orientierte Datenübertragungen, nur unzulängliche Unterstützung. Wissenschaftliche Fragestellungen ergeben sich bei der Erforschung von Möglichkeiten sowie der Entwicklung innovativer Ansätzen (z. B. auf Basis von Network Coding), mit denen es gelingen kann die gegebenen Rahmenbedingungen optimal auszunutzen.

3.3.4 Zeitsynchronisierung von Endknoten

Zeitsynchronisierung bezeichnet die Herstellung einer gemeinsamen Zeit (in beliebig genauer Auflösung) auf allen Endknoten einer Anwendung. Narrowband-basierte Kommunikationslösungen unterstützen zunächst keine zeitliche Synchronisation direkt auf den Endknoten. Üblicherweise werden erfasste Messdaten an das Gateway übertragen und erst dort mit einem Zeitstempel versehen, bevor sie an zentrale Steuerungs- und Kontrollsysteme weitergeleitet werden. Die Folge ist, dass je nachdem wann ein Endknoten einen Messwert erfolgreich zum Gateway übertragen hat, der Messwert bereits einige Minuten alt sein kann. Für manche Anwendungen mag diese zeitliche Auflösung aber zu gering sein. Außerdem kann die Reihenfolge in der gesuchte Ereignisse an verschiedenen Sensorknoten auftreten nicht zuverlässig erfasst werden – die Ereignismeldungen der unterschiedlichen Sensorknoten können sich überholen. Sollen beispielsweise Bewegungen oder kausale Zusammenhänge verschiedener Events untersucht werden, kann die Erfassung solcher Reihenfolgen notwendig werden.

Im Falle der LoRa-Funkübertragung kann Zeitsynchronisierung der Endknoten auch für die effizientere Nutzung der zur Verfügung stehenden Bandbreite genutzt werden. Da LoRa bzw. LoRaWAN kein Medienzugriffsprotokoll für den Zugriff mehrerer Sender auf ein gemeinsames Medium unterstützt – im Gegensatz zu CSMA/CA bei WLAN oder SC-FDMA in LTE bzw. NB-IoT – entstehen Ineffizienzen bei der Bandbreitennutzung. Endknoten beginnen mit dem Senden eines Datenpakets ohne Rücksicht auf andere Sender in ihrer Reichweite – Kollisionen des eigenen Sendevorgangs mit dem

Sendevorgang anderer Knoten werden nicht erkannt (siehe Abbildung 3.4). Der Verzicht auf ein Medienzugriffsprotokoll wie SC-FDMA ist dem sonst notwendigeren, höheren Energiebedarf geschuldet sowie der Tatsache, dass Endknoten komplexer und damit teurer werden würden.

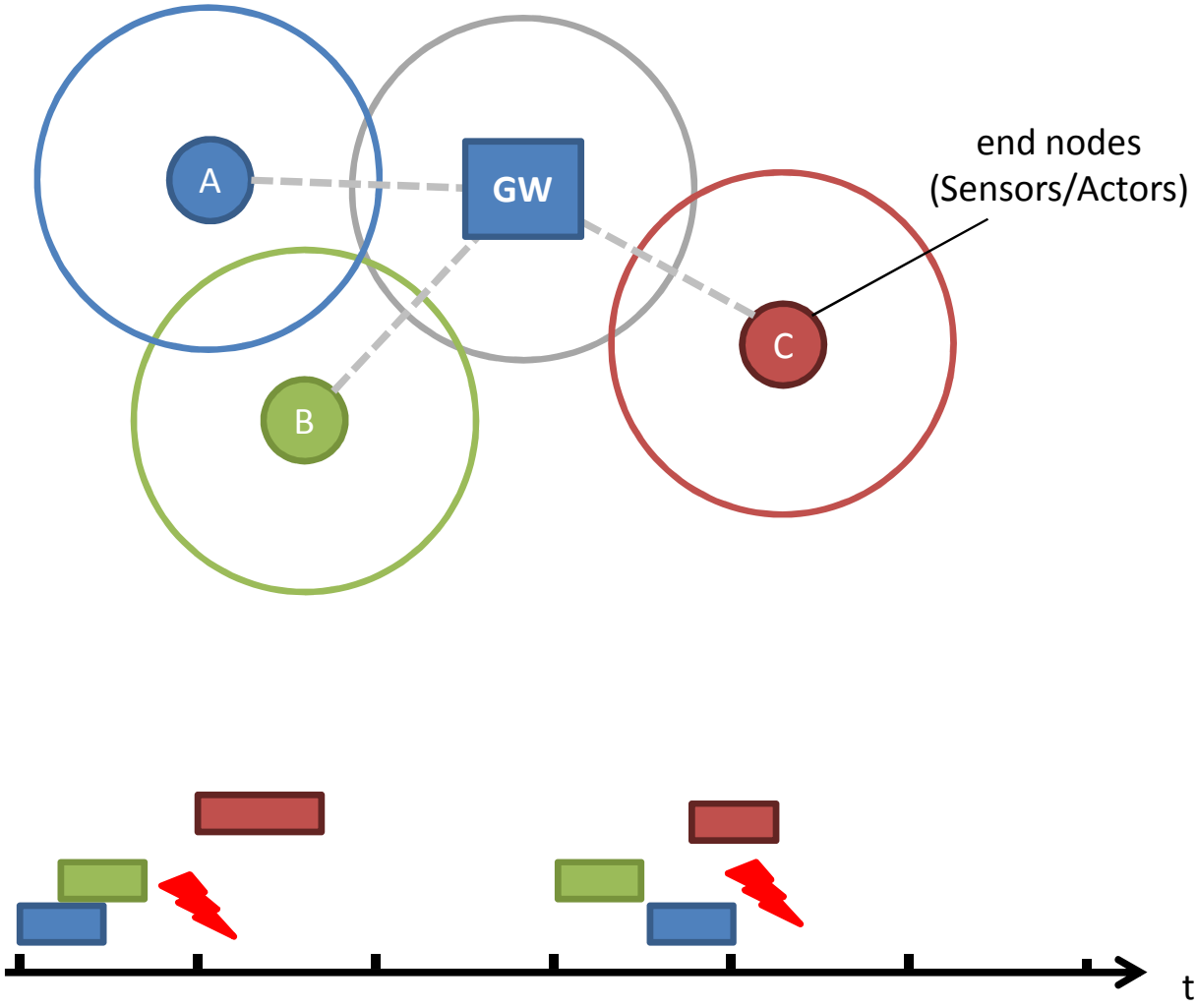


Abbildung 3.4: Kollisionen bei der Datenübertragung über Funkverbindungen

Die Einführung von Zeitsynchronisierung der Endknoten kann dazu verwendet werden, nach dem Prinzip Time Division Multiple Access (TDMA) Sendezeitpunkte für verschiedene Knoten festzulegen mit dem Ziel Kollisionen zu verhindern. Weiterhin kann Zeitsynchronisierung zum Beispiel im Zusammenhang mit Peer-to-Peer Kommunikation nützlich sein, um die Batteriekapazitäten der Relay-Knoten zu schonen. Die Effektivität dieses Vorgehens hängt jedoch stark von Randbedingungen ab wie dem Vorhandensein und der Anzahl nicht-synchronisierter Knoten, die im gleichen Frequenzband senden (Die für LoRa definierten Frequenzbänder können auch von anderen Technologien genutzt werden). Weiterhin bringt der TDMA-Ansatz selbst Ineffizienzen in der Bandbreitennutzung ein. Einerseits wird Bandbreite bei der Nicht- oder nur unvollständigen Nutzung der Kapazität einzelner Zeitslots verschwendet. Andererseits entstehen Ineffizienzen, falls einzelne Zeitslots für die Übertragung eines Pakets nicht ausreichen (z. B. durch mehrmaliges Senden des Paketheaders). Diese Möglichkeiten sind in Abbildung 3.5 abgebildet.

Wissenschaftliche Fragestellungen betreffen diesbezüglich Entwurf und Implementierung eines geeigneten Zeitsynchronisierungsverfahrens, sowie die Erforschung der Rahmenbedingungen unter denen Zeitsynchronisierung tatsächlich einen Vorteil für die effektive Kanalnutzung bedeuten.

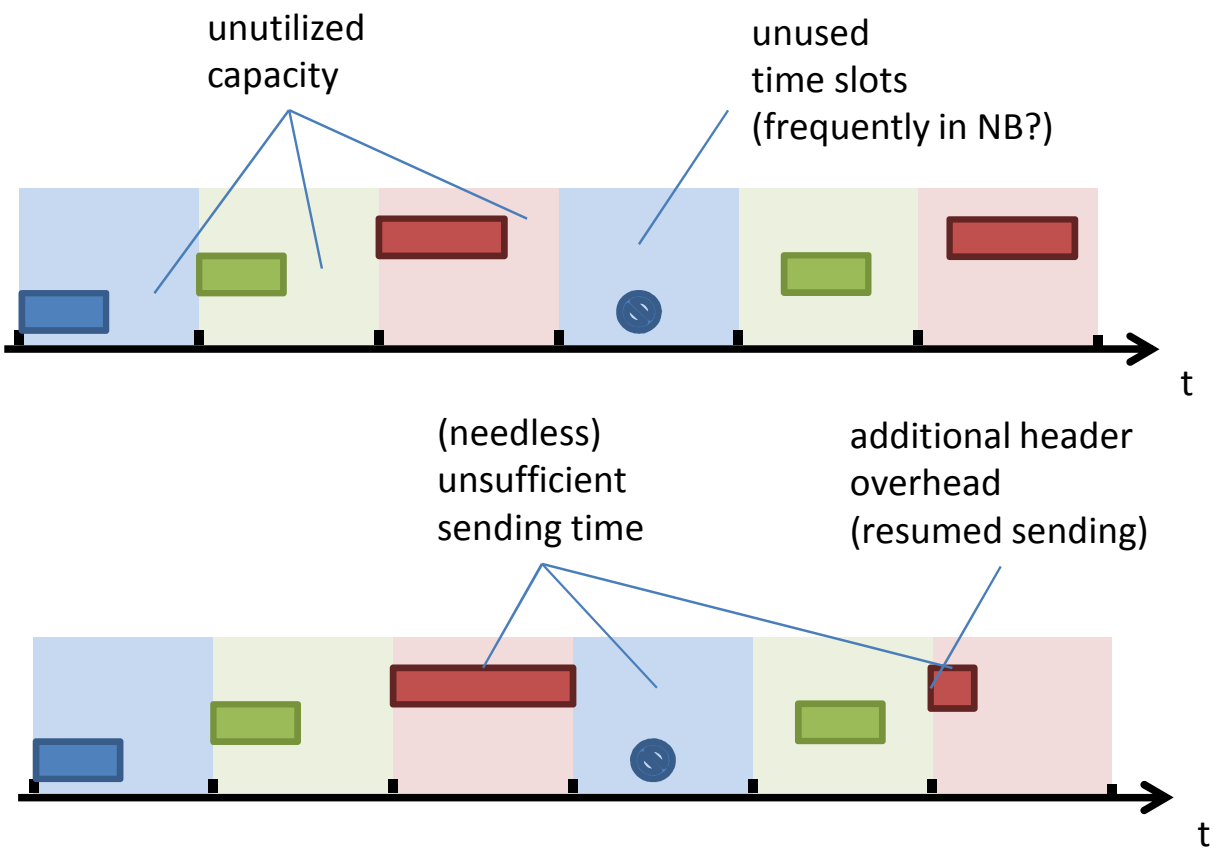


Abbildung 3.5: Ineffiziente Bandbreitennutzung bei TDMA-basierter Kanalaufteilung

3.3.5 Optimiertes Deployment der Kommunikationsknoten

Das Deployment der Gateways und Endknoten hat einen großen Einfluss auf die Qualität der Kommunikation. Grundsätzlich sind Schmalband-Technologien insbesondere darauf ausgelegt große Gebiete und eine Vielzahl von Knoten abzudecken und gleichzeitig eine gute Durchdringung von Gebäuden und anderen Hindernissen zu gewährleisten. Beim Deployment der Kommunikationsknoten gibt es dennoch Randbedingungen, die Effizienz, Zuverlässigkeit und Langlebigkeit des Systems beeinflussen. Wissenschaftliche Fragestellungen fokussieren sich darauf diese Randbedingungen zu erkennen und zu bewerten, um im Rahmen der Möglichkeiten eine Optimierung zu erreichen.

Diese Fragestellungen betreffen zum Beispiel wie Umweltbedingungen (z. B. Geröll, Schnee, Wasser, Gase, Strahlung) und Geländeeigenschaften (urbaner und ländlicher Raum, flach/hügelig/bergig), sowie Dichte der Kommunikationsknoten (Auslastung des Netzes) aber auch Sendehäufigkeiten der Endknoten die Kommunikationsqualität insgesamt beeinflussen. Aus diesen Erkenntnissen können schließlich Fragestellungen beantwortet werden, wie Knoten im Messgebiet zu verteilen sind, um ein optimales Ergebnis zu erreichen. Weiterhin stellt sich die Frage nach Messverfahren, die für die Planung von Schmalband-Netzen relevante Informationen liefern können. Auch die gegebenenfalls selbst-adaptive Verfahren zur Optimierung der Kommunikationsqualität stehen in der Betrachtung. Grundlagen zur Beantwortung dieser Fragen sind im Forschungsgebiet der Wireless-Sensor-Netze und im Bereich der Mobilfunknetzplanung zu finden.

3.3.6 Kritische Kommunikation

Datenübertragung per Schmalband-Technologie ist im Grundsatz auf "best effort"Nutzung ausgelegt. Garantien in Bezug auf zuverlässige Informationsübertragung oder maximale Latenzen sind nicht vorgesehen. Dennoch kann Schmalband-Technologie nützlich für die Verbreitung hoch-kritischer Infor-

mationen sein – zum Beispiel im Kontext von multi-modaler Informationsverteilung. In solchen Anwendungsszenarien werden Informationen – auf Grund der Tatsache das keine Informationstechnologie zu jedem beliebigen Zeitpunkt immer hundertprozentig zuverlässig funktionieren kann - gleichzeitig über unterschiedliche Kommunikationskanäle übertragen. Diese Kanäle umfassen dabei verschiedene Übertragungstechnologien bis hin zu unterschiedlichen Informationsdiensten wie E-Mail oder sozialen Netzwerken.

Insbesondere in Situationen in denen primäre Kommunikationssysteme zum Beispiel durch die Folgen einer Umweltkatastrophe in Mitleidenschaft genommen werden und zusätzlich durch ein erhöhtes Kommunikationsbedürfnis der Bevölkerung belastet sind, können Schmalband-basierte Übertragungsverfahren gegebenenfalls eine wichtige Informationsquelle für Hilfs- und Rettungskräfte darstellen. Aus wissenschaftlicher Sicht ist es jedoch notwendig die Möglichkeiten, die Schmalband-Technologie hier bietet zu ermitteln und zu bewerten.

3.4 Links

Für weitere Informationen zum Thema Innovationen auf Basis von Schmalbandkommunikation bieten sich folgende Informationsquellen an:

- https://www.microtronics.at/de/m2m/m2m_in_use.html
- https://iot.ieee.org/iot-scenarios.html
- https://www.sigfox.com/en/solutions/iot-use-cases
- https://www.cooking-hacks.com/documentation/tutorials/lorawan-for-arduino-raspberry-pi-waspmote-868-900-915-433-mhz
- https://diyprojects.io/wireless-technology-build-diy-iot-project/
- http://cds.linear.com/docs/en/article/Micro%20Jour%200812%20WSNs.pdf
- http://www.paredox.com/foswiki/pub/Luichart/RFIDandWirelessSensorNetworks/RFID_and_Wireless_Sensor_Networks.pdf
- http://files.wimaxforum.org/Document/Download/WiMAX_IEEE_802.16m_Air_Interface_Standard
- https://www.fcm.fraunhofer.de/en/beispiele11/drahtlose_sensornetzeinderland-undforstwirtschaft.html
- http://www.academia.edu/1747732/Presentation_file_about_Wireless_Sensor_Network_Applications_and_Challenges
- http://wireless.ictp.it/wp-content/uploads/2012/02/WSN-Applications.pdf
- http://www.die-salzburger-industrie.at/
- https://www.computerwoche.de/a/ueber-machine-to-machine-und-internet-der-dinge-zur-industrie-4-0,3068010
- http://defl.ch/wp-content/uploads/2017/01/Christian_Ruckstuhl_innet_DEFL.pdf
- https://www.open-homeautomation.com/
- https://gdpr.report/news/2017/05/26/message-future-critical-communications-iot/

- http://iot.sys-con.com/node/3344766
- https://www.microtronics.at/de/m2m/m2m_in_use.html
- http://iot.ieee.org/iot-scenarios.html
- https://www.sigfox.com/en/solutions/iot-use-cases
- http://www.honeywellanalytics.com/en/products/ToxiPro

4 LoRa Einsatz

Dieses Kapitel beschreibt den Aufbau und die Inbetriebnahme eines Sensors mit LoRa Technologie. Es wird die Temperatur mit einem Sensor gemessen, diese über LoRa an einen LoRa Gateway übertragen, welcher den Messwert an einen Lora-Server weiterleitet. Dieser führt LoRa-spezifische Überprüfungen durch (z. B. Verschlüsselung) und leitet den Messwert dann an eine Influx Datenbank weiter. Über Grafana können diese Messwerte dann visualisiert werden. Abbildung 4.1 illustriert die Verkettung der Systeme.

Diese Kapitel geht nicht auf die Details der LoRaWAN Spezifikation [1] ein, sondern zeigt die Schritte auf, welche durchgeführt werden müssen, um ein lauffähiges System auf Basis von LoRa zu erhalten. Bevor ein solches System im produktiven Betrieb genutzt werden kann, muss es allerdings noch einer Sicherheitsprüfung unterzogen werden.

4.1 Hardware

Es gibt diverse Anbieter für LoRa Hardware; sowohl für die Endgeräte als auch für die Gateways. Als Gateway verwenden wir ein Multi-tech Conduit in der Variante mit Linux Betriebssystem[1]. Zur Ergänzung der LoRa-Fuktionalität verwenden wir das zugehörige Einsteckmodul mit der 868 MHz Variante für Europa[2] mit passender Antenne.[3]

Aufgrund der einfachen Programmierbarkeit mit Micropython (Python für Microcontroller) verwenden wir als Endgeräte LoPy von pycom im Paket[4] mit Gehäuse.[5] Da das Expansionboard keine Sensoren enthält, benötigen wir später das Pysense-Board[6] um die Temperatur zu messen.

Geräte, die dem Microcontroller zusätzliche Funktionen hinzufügen, wie das Expansion Board oder das Pysense, werden auch Shields genannt.

4.2 Pycom Engeräte

Information

Die von uns verwendeten Software-Versionen sind nur als Referenz angegeben, wir empfehlen ausdrücklich die jeweils aktuelle Version von der Software zu verwenden und nicht die von uns genutzte. Da viele der verwendeten Projekte in Entwicklung sind, ist zu erwarten, dass diese häufig aktualisiert werden und auch größere Änderungen vorgenommen werden.

Zum Einrichten der LoPy Endgeräte verwenden wir einen Laptop mit OpenSuse Leap 42.2 (oder später 42.3), aber andere Linux-Betriebssysteme sollten ähnlich funktionieren. Anleitungen für Windows sind ebenfalls in der PyCom Dokumentation oder im Kpaitel zu Sigfox (Kapitel refcah:sigfox) zu finden.

[1] https://www.digikey.at/product-detail/de/multi-tech-systems-inc/MTCDT-H5-210L-US-EU-GB/881-1236-ND/5246365

[2] https://www.digikey.at/product-detail/de/multi-tech-systems-inc/MTAC-LORA-868/881-1243-ND/5322991

[3] https://www.digikey.at/product-detail/de/multi-tech-systems-inc/AN868-915A-10HRA/881-1242-ND/5246371

[4] https://www.pycom.io/product/lopy-super-twin-pack/

[5] https://www.pycom.io/product/pycase-clear/

[6] https://www.pycom.io/product/pysense/

Abbildung 4.1: Um die gemessene Temperatur vom Sensor zur Visualisierung in Grafana eine Vielzahl an Hard- und Software- Komponenten nötig.

Einen guten Einstieg im Umgang mit der pycom Hardware gibt es im "Getting started"Guide.[7]

Als Erstes haben wir folgend dem Guide mit dem firmeware updater (Version 1.1.1.b2) die Firmware aller LoPy's aktualisiert (auf Version 1.6.8.b2). Hierzu ist es nötig evtl. beim Transport abgefallene Jumper wieder zu befestigen und den Updater mit root-Rechten auszuführen (evtl. reicht es auch dem Nutzer USB Zugriffsrechte zu geben). Mehr Details was das Update zum Scheitern bringen kann finden sich in der Upgrade Checkliste.[8]

4.2.1 Verbindung zum LoPy

Pycom bietet eine integrierte Entwicklungsumgebung auf Basis von Atom an. Da diese aber noch nicht verfügbar war, als wir LoRa getestet haben, haben wir sie nicht verwendet. Details zur Umgebung befinden sich im Kapitel über Sigfox (siehe Kapitel 5 auf Seite 45)

Unter Linux ist die einfachste Möglichkeit eine direkte Verbindung zum LoPy herzustellen über USB mit dem Kommando:

```
screen /dev/ttyUSB0 115200
```

Die funktioniert als root oder nachdem der aktuelle Nutzer entsprechende Rechte erhalten hat (und sich aus und eingeloggt hat):

```
sudo usermod -a -G dialout $USER
```

4.2.2 Deaktivieren von WLAN

Wir haben aus Stromspar- uns Sicherheitsgründen WLAN deaktiviert.[9]

```
from network import WLAN
wlan = WLAN()
wlan.deinit()
```

Seit einer neueren Version ist es auch möglich WLAN gar nicht erst zu starten.[10]

4.3 LoRaWAN

Für eine Übertragung, die nur den physikalischen LoRa-Layer verwendet, reichen zwei LoPy aus.[11] Um LoRaWAN zu betreiben ist ein Gateway nötig, dass die gesendeten Pakete empfängt.[12]

4.4 LoRaWAN Gateway

Als LoRaWAN Gateway verwenden wir ein MultiTech Conduit mLinux (L Model).[13] Die einfachste Möglichkeit sich mit dem Conduit zu verbinden ist über LAN per SSH. Ab Werk ist das Conduit wie

[7] https://docs.pycom.io/pycom_esp32/pycom_esp32/getstarted.html
[8] https://forum.pycom.io/topic/763/firmware-upgrade-troubleshooting-checklist-procedure/19
[9] https://forum.pycom.io/topic/563/disabling-wifi-on-lopy
[10] https://docs.pycom.io/chapter/firmwareapi/pycom/pycom.html
[11] https://docs.pycom.io/pycom_esp32/pycom_esp32/tutorial/includes/lora-mac.html
[12] https://docs.pycom.io/pycom_esp32/pycom_esp32/tutorial/includes/lora-otaa.html
[13] http://www.multitech.net/developer/software/mlinux/

folgt konfiguriert[14]:

- IP: 192.168.2.1 (DHCP ist deaktiviert)
- username: root
- password: root

Die aktuelle Version des Betriebssystems lässt sich, wenn auf dem Conduit eingeloggt, wie folgt bestimmen:

```
cat /etc/mlinux-version
```

Die aktuelle Version (3.2.0) des Betriebssystems war bereits installiert und musste somit nicht aktualisiert werden[15].

Anhand der Anleitung haben wir nun die installierte Software aktualisiert. Die Abfrage der installierten Versionen mit

```
opkg info lora*
```

Aktualisierung auf Versionen:

- `lora-network-server` auf 1.0.13
- `lora-packet-forwarder` ist bei 1.4.2-r9.0
- `lora-query` ist bei 1.0.2-r1.0

Einbau der LoRa Karte[16] und Erste Schritte mit LoRa und Conduit (L Model) sind auf der Multitech Webseite dokumentiert.[17] Im Anschluss haben wir auch das root Passwort geändert.

DNS server wurde nicht automatisch gesetzt. Wir haben diesen dann indirekt selbst [18] in `/etc/network/interfaces` gesetzt

```
post-up echo "nameserver 9.9.9.9" > /etc/resolv.conf
```

und in `/etc/default/ntpdate` den Synchronisationspool `pool.ntp.org` ergänzt, um Zeitsynchronisation zu erreichen.

Es gibt zwei Möglichkeiten LoRa am Conduit zu betreiben:

- LoRa-Server im Conduit
- Conduit als Forwarder und mit externem LoRa Server

4.5 LoRa-Server im Conduit

Wir erläutern kurz die Variante, in der das Conduit den LoRa Server stellt, verwenden aber später die externe, da diese mehr Flexibilität bietet.

Da das Conduit in seiner Initial-Konfiguration nur, MultiTech LoRa-Hardware akzeptiert, muss andere Hardware erst aktiviert werden.[19]

Die Konfigurationsdatei `/var/config/lora/lora-network-server.conf` existiert nicht, aber kann so erzeugt werden:

[14] http://www.multitech.net/developer/software/mlinux/getting-started-with-conduit-mlinux/
[15] http://www.multitech.net/developer/software/mlinux/using-mlinux/flashing-mlinux-firmware-for-conduit/
[16] http://www.multitech.net/developer/products/accessory-cards/installing-an-accessory-card/
[17] http://www.multitech.net/developer/software/lora/getting-started-with-lora-conduit-mlinux/
[18] http://www.multitech.net/developer/forums/topic/dns-configuration-on-mlinux/
[19] http://www.multitech.net/developer/software/lora/conduit-mlinux-lora-communication/conduit-mlinux-lora-use-third-party-devices/

```
mkdir /var/config/lora
cp /opt/lora/lora-network-server.conf.sample /var/config/lora/
   lora-network-server.conf
```

Und folgende Änderungen durchgeführt:

- lora:frequency band auf 868
- network:public auf true
- name und passphrase entfernt

.

Außerdem ist es nötig in dieser Datei eine EUI und einen KEY eingetragen werden, die eindeutig sind. Wir haben diese zufällig erzeugt, durch (in Python auf einem beliebigem PC):

```
import os,binascii
binascii.b2a_hex(os.urandom(16))
```

In unserem Fall hat dies ergeben:

- eui: "ff42d196516f3c5b"
- key: "8c48122fe46378c5341e72021b693bc7"

Nächster Schritt: Erzeugen von Paketen auf dem LoPy.[20]
Der LoPy kann durch folgende Kommandos mit dem LoRa Netz verbunden werden:

```
from network import LoRa
import socket
import time
import binascii

# Initialize LoRa in LORAWAN mode.
lora = LoRa(mode=LoRa.LORAWAN, region=LoRa.EU868)

# create an OTAA authentication parameters
app_eui = binascii.unhexlify('ff42d196516f3c5b')
app_key = binascii.unhexlify('8c48122fe46378c5341e72021b693bc7')

# join a network using OTAA (Over the Air Activation)
lora.join(activation=LoRa.OTAA, auth=(app_eui, app_key), timeout
   =0)

lora.has_joined()
```

letztes Kommando sagt True und

```
tail -f /var/log/lora-network-server.log
```

auf dem Conduit empfängt ein Paket.
Überprüfung weiterer Details auf dem Conduit[21]:

```
root@mtcdt:~# lora-query -n
Net Addr      Dev EUI                    Class  Joined
                   Seq Num        Up      Down       1st       2nd
   Dropped  RSSI min   max   avg  SNR min   max   avg
```

[20] https://docs.pycom.io/pycom_esp32/pycom_esp32/tutorial/includes/lora-otaa.html
[21] http://www.multitech.net/developer/software/lora/lora-network-server/

```
06:00:00:01  70-b3-d5-49-95-b7-af-71  A      2017-03-24T13:51:37Z
           2         2         3         3         0         0
   -3    -1    -2      7.2   8.5   7.8
```

zeigt RSSI und SNR der Übertragung.[22]

Per MQTT client (z.B. mosquitto) können die empfangen Daten ausgelesen werden. Dies kann auf dem Conduit, aber auch auf einem anderen PC geschehen.

```
root@mtcdt:~# mosquitto_sub -t lora/+/+ -v
lora/70-b3-d5-49-95-b7-af-71/packet_recv {"chan":0,"codr":"4/5","
   data":"QAEAAAYAAwACjSHfI3wnQg==","datr":"SF7BW125","freq
   ":868.10000000000002,"lsnr":9.8000000000000007,"modu":"LORA","
   rfch":0,"rssi":-5,"size":16,"stat":1,"time":"2017-03-24T14
   :02:01.447281Z","tmst":760202795}
lora/70-b3-d5-49-95-b7-af-71/up {"chan":0,"cls":0,"codr":"4/5","
   data":"AQIE","datr":"SF7BW125","freq":"868.1","lsnr":"9.8","
   mhdr":"4001000006000300","modu":"LORA","opts":"","port":2,"rfch
   ":0,"rssi":-5,"seqn":3,"size":4,"timestamp":"2017-03-24T14
   :02:01.447281Z","tmst":760202795}
```

4.6 LoRa-Server in VM

Zuerst muss das Conduit als Paket Forwarder konfiguriert werden.[23] Wir verzichten darauf die Anleitung hier wiederzugeben, da diese stark von der verwendeten Version des Conduit abhängt und sich unter Umständen in der Zukunft ändern wird. Die Version auf der Webseite des Herstellers ist also jeder (möglicherweise veralteten) anderen Dokumentation vorzuziehen. Im folgenden werden wir allerdings unsere Kommentare zu der Umkonfiguration anbringen.

Information

In der Beschreibung zur Konvertierung des Conduit zum forwarder[a] ist die neuste Version nicht die neuste Verfügbare (es Beschreibt die Umstellung für bisher noch nicht verfügbare Versionen).

[a] http://www.multitech.net/developer/software/lora/conduit-mlinux-convert-to-basic-packet-forwarder/

Die Konvertierung in einen Packet forwarder hat bei uns nicht auf Anhieb funktioniert. Folgende zusätzliche Schritte sind eventuell hilfreich:

- Schritt 4: Editieren von `/etc/defaults/lora-packet-forwarder`: Kopieren von `lora-network-server` zu `lora-packet-forwarder`.
- Schritt 6:

  ```
  cp /run/lora/1/global_conf.json /var/config/lora
  ```

- Server Adresse manuell in `global\_conf.json` auf host machine (192.168.2.3 mit Port 1700) gesetzt.
- Manuelles Starten durch

  ```
  /opt/lora/basic_pkt_fwd -c /var/config/lora
  ```

[22] http://www.multitech.net/developer/software/lora/conduit-mlinux-lora-communication/

[23] http://www.multitech.net/developer/software/lora/conduit-mlinux-convert-to-basic-packet-forwarder/

- Firewall auf Host deaktivieren

Nachdem der Conduit in einen forwarder umkonfiguriert wurden, wird ein LoRa-Server benötigt. Wir verwenden den quell-offenen Lora-Server. [24] Er kann durch Vagrant aufgesetzt werden[25] und hat einen Einstiegsguide.[26] Allgemein ist auch vagrant gut dokumentiert.[27]

Nach dem Ausführen des Vagrant Skripts kann der Server durch Zugriff auf `localhost:8080` per Browser erreicht werden (der Port wurde automatisch vom Host auf dem Gast weitergeleitet). Da wir kein extern signiertes Zertifikat verwenden, muss hier (temporär) das Zertifikat manuell akzeptiert werden. Der Login ist standardmäßig mit `admin/admin` durchzuführen. Es gibt auch ein REST interface für automatisierte Änderungen und Abfragen.

Erinnerung

Standardlogins, wie auf dem LoRa-Server, sollten in produktiven Systemen schnellstmöglich geändert werden, da dies in späterer Folge sonst oft vergessen wird.

Information

Wenn die vagrant Maschine fertig eingerichtet und gestartet wurde ist es möglich sich durch `vagrant ssh` mit dieser zu verbinden. Falls die Maschine später auf einen anderen Host migriert wird (z.B. weil die fertige Maschine auf einem Server im Dauereinsatz laufen soll), ist es möglich sich ohne vagrant mit folgendem Kommando trotzdem mit der Maschine zu verbinden:

```
ssh ubuntu@127.0.0.1 -p2222 -i /<directory>/loraserver-setup/.
    vagrant/machines/vagrant/virtualbox/private_key
```

Als nächstes müssen die LoPys auf dem loraserver eingerichtet werden. Dies kann z.B. mit einem Browser über `localhost:8080` gemacht werden. Hierzu wird die DevEUI des Gerätes benötigt. Die `DevEUI` kann weder im LoPy noch auf dem Server geändert werden. Es ist also nötig die Knoten auf dem Server direkt korrekt zu einzurichten. Die `DevEUI` kann folgendermaßen auf dem LoPy ausgelesen werden:

```
from network import LoRa
import binascii
lora = LoRa(mode=LoRa.LORAWAN)
print(binascii.hexlify(lora.mac()).upper().decode('utf-8'))
```

In unserem Fall hat dies folgende Werte ergeben:

- DemoNode1: 70b3d54995b7af71
- DemoNode2: 70b3d5499cb5ac42

Information

Die vagrant Maschine kann mit folgendem Befehl angehalten werden: `vagrant halt`.

[24] https://docs.loraserver.io/loraserver/
[25] https://github.com/brocaar/loraserver-setup
[26] https://docs.loraserver.io/loraserver/getting-started/
[27] https://www.vagrantup.com/intro/getting-started/index.html

4.6.1 Verlegen der VM auf Server

Wir haben den LoRa-Server zum Testen zuerst auf einem Desktop-Rechner installiert. Um den Server nun im Dauerbetrieb zu nutzen, soll er auf einen Server verschoben werden. Wer den LoRa-Server direkt auf einem Server installiert hat, kann diesen Abschnitt Überspringen und direkt zu Kapitel 4.6.2 springen.

Als Erstes sollte die Netzwerk-Konfiguration auf die neue Umgebung angepasst werden. Bei uns heißt das, dass DHCP eingeschaltet werden muss[28]:

```
auto eth0
iface eth0 inet dhcp
```

Warnung

Wenn direkt in `/etc/network/interfaces` gesetzt, scheint DNS nicht zu funktionieren. Wir haben dies umgangen[a] indem wir in `/etc/network/interfaces` folgendes ergänzt haben:

```
post-up echo "nameserver 9.9.9.9" > /etc/resolv.conf
```

[a] http://www.multitech.net/developer/forums/topic/dns-configuration-on-mlinux/

Information

Am Ende der `pkt-fwd` Zeile in `/etc/init.d/lora-network-server` befand sich "/1", welches dort nicht hingehört.

Information

In der aktuellen Version hat `loraserver-setup` ein Problem. Es ist nötig manuell den Postgresql server zu setzen (von `localhost` auf `loraserver:verysecret@localhost`) und Rollen für den loraserver zu erstellen:

```
sudo -u postgres psql

-- create the loraserver_ns user
create role loraserver_ns with login password 'dbpassword';

-- create the loraserver_ns database
create database loraserver_ns with owner loraserver_ns;

-- exit
\q
```

In einer späteren Version ist dies hoffentlich wieder behoben.

4.6.2 Daten Empfangen

Die empfangenen Daten können (auf dem Server) durch folgendes Kommando angezeigt werden:

```
mosquitto_sub -t "application/#" -v
```

Die Daten, welche noch in Base64 codiert sind, können in einer beliebigen Python Umgebung dekodiert werden:

[28] http://www.multitech.net/developer/software/mlinux/getting-started-with-conduit-mlinux/

```
import base64
base64.b64decode('AQID')
```

Information

Ampy[a] ist ein hilfreiches Werkzeug um mit LoPys zu arbeiten da es erlaubt direkt über USB Dateien zu übertragen. Damit der Port nicht bei jedem übertragen angegeben werden muss, kann er fest gesetzt werden:

```
export AMPY_PORT=/dev/ttyUSB0
```

[a] https://github.com/adafruit/ampy

Allgemein nützliche Kommandos zur Ausführung auf LoPys sind:

- LoRa initialisieren:

```
from network import LoRa
import socket
import time
import binascii
lora = LoRa(mode=LoRa.LORAWAN)
app_eui = binascii.unhexlify('ff42d196516f3c5b')
app_key = binascii.unhexlify('8c48122fe46378c5341e72021b693bc7
    ')
lora.join(activation=LoRa.OTAA, auth=(app_eui, app_key),
    timeout=0)
```

- Daten senden:

```
s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
s.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)
s.setblocking(True)
s.send(bytes([0x01, 0x02, 0x03]))
```

- Daten empfangen:

```
s.setblocking(False)
data = s.recv(64)
print(data)
```

- WLAN deaktivieren:

```
from network import WLAN
wlan = WLAN()
wlan.deinit()
```

- Heartbeat deaktivieren:

```
import pycom
pycom.heartbeat(False)
pycom.rgbled(0x000000)
```

- Regelmäßigen Timer setzen:

```
from machine import Timer

def alarm(alarm):
    print("Alarm")

Timer.Alarm(alarm, 1, periodic=False)
```

Vereint man dieser Code-Blöcke zu einem lässt sich darüber ein LoPy programmieren der einen Wert pro Stunde versendet (Demo.py):

```
# Disable WiFi
from network import WLAN
wlan = WLAN()
wlan.deinit()

# Disable Heartbeat
import pycom
pycom.heartbeat(False)
pycom.rgbled(0x000000)

# Connect to SRFG Network
from network import LoRa
import socket
import time
import binascii
lora = LoRa(mode=LoRa.LORAWAN)
app_eui = binascii.unhexlify('ff42d196516f3c5b')
app_key = binascii.unhexlify('8c48122fe46378c5341e72021b693bc7')

lora.join(activation=LoRa.OTAA, auth=(app_eui, app_key), timeout
   =0)

# Send a Message every 3600s = 1h
from machine import Timer

def alarm(alarm):
    s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
    s.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)
    s.setblocking(True)
    s.send(bytes([0x01, 0x02, 0x03]))

Timer.Alarm(alarm, 3600, periodic=True)
```

Warnung

Dieser Code nutzt nicht die Deep-Sleep Funktionalität[a] und ist daher nur für den Betrieb an einer ständigen Stromquelle geeignet. Für einen Batteriebetrieb sollte der Deep-Sleep Modus implementiert werden. Die von Pycom bereitgestellte deepsleep-Library funktioniert mit Stand Februar 2018 noch nicht einwandfrei.

[a] https://docs.pycom.io/chapter/datasheets/boards/deepsleep/api.html

Warnung

Das Kommando `ampy reset` scheint nicht das gleiche zu tun wie ein power cycle. Um das Gerät zuverlässig neuzustarten sollte ein power cycle durchgeführt werden.

Von einem beliebigen Gerät können nun die übertragenen Daten, aber auch sich anmeldende Sensoren angezeigt werden:

```
mosquitto_sub -t '#' -v -h <host>
```

4.7 Pysense

Da das LoPy keine Sensoren beinhaltet, ersetzen wir das Expansion Board durch ein Pysense[29] mit diversen Sensoren.

Das Expansion Board passt genau in das Clearcase, aber nicht in das IP67 Gehäuse. Die Pysense und Pytrack Shields passen genau in das IP67 Gehäuse. Für das IP67 Case[30] gibt es ein eigenes Antennenkabel[31] für wasserdichte Konstruktionen. Für die Antenne muss selbst ein Loch gebohrt werden.[32]

Dokumentation der Pysense-Software ist online verfügbar.[33]

Information

In der von unser verwendeten Firmware-Version konnten wir von einem Linux-Rechner aus keine Verbindung zum LoPy über USB herstellen, wenn es in einem Pysense steckt. Eine Verbindung über das Expansion Board oder mit einem Windows PC war allerdings möglich.

Information

Temperatursensor war Anfangs nicht wirklich zuverlässig nutzbar[a]. Nach dem Aktualisieren der Treiberbibliothek funktioniert das Auslesen allerdings fehlerfrei.

[a] https://forum.pycom.io/topic/1373/pysense-lots-of-invalid-temp-hum-values

Der Code zum Senden der Temperatur lautet (`Temp.py`):

```
# Disable WiFi
from network import WLAN
wlan = WLAN()
wlan.deinit()

# Disable Heartbeat
import pycom
pycom.heartbeat(False)
pycom.rgbled(0x000000)

# Connect to SRFG Network
from network import LoRa
import socket
```

[29] https://www.pycom.io/product/pysense/
[30] https://pycom.io/product/ip67-case/
[31] https://pycom.io/product/ip67-antenna-cable/
[32] https://forum.pycom.io/topic/1348/casing-for-the-pytrack-and-pysense-together-with-lopy
[33] https://docs.pycom.io/pycom_esp32/pycom_esp32/tutorial/includes/pysense-start.html

```
import time
import binascii
lora = LoRa(mode=LoRa.LORAWAN)
app_eui = binascii.unhexlify('ff42d196516f3c5b')
app_key = binascii.unhexlify('8c48122fe46378c5341e72021b693bc7')

lora.join(activation=LoRa.OTAA, auth=(app_eui, app_key), timeout
   =0)

# Send a Message every 3600s = 1h
from machine import Timer

# Needed for random number
import uos

# Needed for temp sensor (one of two possible drivers)
from SI7006A20 import SI7006A20
sen = SI7006A20()

def alarm(alarm):
    s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
    s.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)
    s.setblocking(True)

    # Send a random byte
    #s.send(uos.urandom(1))

    # Send temperature
    temp = int(sen.temperature())
    if temp > 255:
        temp = 255
    if temp < 0:
        temp = 0
    s.send(bytes([temp]))

Timer.Alarm(alarm, 3600, periodic=True)
```

4.8 Graflux zur Visualisierung

Um die Daten an der Server-Seite zu speichern und zu visualisieren verwenden wir InfluxDB[34] und Grafana[35]. hierzu ist es nötig eine virtuelle Maschine zu erstellen in der InfluxDB und Grafana ausgeführt werden. Die hierzu nötigen Schritte können selbst durchgeführt werden (sieh InfluxDB[36] und Grafana[37] Dokumentation). Alternativ ist eine vorkonfigurierte virtuelle Maschine auf Anfrage bei Salzburg Research verfügbar.

Nachdem sowohl InfluxDB und Grafana eingerichtet sind, ist es sinnvoll für die Speicherung eine neue Datenbank anzulegen. Dies geschieht auf der virtuellen Maschine über:

[34] https://www.influxdata.com
[35] https://grafana.com
[36] https://docs.influxdata.com/influxdb/v1.5/introduction/installation
[37] http://docs.grafana.org/installation/rpm/

```
influx
CREATE DATABASE lora
```

Information

Das Anzeigen des Inhalts der Datenbank ist folgender Massen möglich:

```
USE lora
SELECT * FROM data
```

Für weitere Details über Influx siehe Dokumentation[a]

[a] http://docs.influxdata.com/influxdb/v1.2/introduction/getting_started/

Zum Konvertieren der MQTT Daten vom Loraserver für Influx verwenden wir ein eigenes Skript: MQTT2Influx[38][39]. Die nötigen Bibliotheken sind wie folgt zu installieren:

```
sudo apt-get install python3-pip

sudo pip3 install influxdb
sudo pip3 install paho-mqtt
```

Folgendes Python Skript über nimmt die Konvertierung von MQTT zu Influx (`MQTT2Influx`):

```
import re
import base64

import paho.mqtt.client as mqtt

from influxdb import InfluxDBClient

# The callback for when the client receives a CONNACK response
   from the server.
def on_connect(mqtt, userdata, flags, rc):
    print("Connected␣with␣result␣code␣"+str(rc))

    # Subscribing in on_connect() means that if we lose the
       connection and
    # reconnect then subscriptions will be renewed.
    mqtt.subscribe("#")

# The callback for when a PUBLISH message is received from the
   server.
def on_message(mqtt, userdata, msg):
    print(msg.topic+"␣"+str(msg.payload))
    dataMatch = dataPattern.match(str(msg.payload))
    timeMatch = timePattern.match(str(msg.payload))
    appNameMatch = appNamePattern.match(str(msg.payload))
    nodeNameMatch = nodeNamePattern.match(str(msg.payload))
    rssiMatch = rssiPattern.match(str(msg.payload))
    loraSNRMatch = loraSNRPattern.match(str(msg.payload))

    if dataMatch and appNameMatch:
```

[38] https://pypi.python.org/pypi/paho-mqtt
[39] http://influxdb-python.readthedocs.io/en/latest/examples.html

```
        json_body = [
            {
                "measurement": "data",
                "tags": {
                    "appName": appNameMatch.group(1),
                    "nodeName": nodeNameMatch.group(1)
                },
                "time": timeMatch.group(1),
                "fields": {
                    #Add first byte as temp
                    "temp": int.from_bytes(base64.b64decode(
                       dataMatch.group(1)), byteorder='big')
                    ,
                    "rssi": rssiMatch.group(1),
                    "loraSNR": loraSNRMatch.group(1)
                }
            }
        ]
        influx.write_points(json_body)

# Is this secure? It might match these Strings if they are
   included in some of the data!
timePattern = re.compile('.*"time":"([\d\-.:TZ]*)",.*')
appNamePattern = re.compile('.*"applicationName":"(SRFG-Demo-App)
   ",.*')
nodeNamePattern = re.compile('.*"nodeName":"([^"]*)",.*')
rssiPattern = re.compile('.*"rssi":([\d\-]*),.*')
loraSNRPattern = re.compile('.*"loRaSNR":([\d\-\.]*),.*')
dataPattern = re.compile('.*\"data\":\"(.*)\"}.*')

# Connect to influx
influx = InfluxDBClient("localhost", 8086, "", "", "lora")

mqtt = mqtt.Client()
mqtt.on_connect = on_connect
mqtt.on_message = on_message

mqtt.connect("192.168.40.180", 1883, 60)

# Start collecting data from mqtt
mqtt.loop_forever()
```

Information

Ein gezielteres Auslesen der Daten über MQTT statt Pattern wäre auch möglich und würde wahrscheinlich die Auslastung des Systems verringern.

Warnung

Die im Code verwendeten Pattern sind nicht auf Sicherheitslücken geprüft. Vor dem Einsatz in einem produktiven System sollte dies nachgeholt werden oder die Daten direkt aus der MQTT Schnittstelle ausgelesen werden.

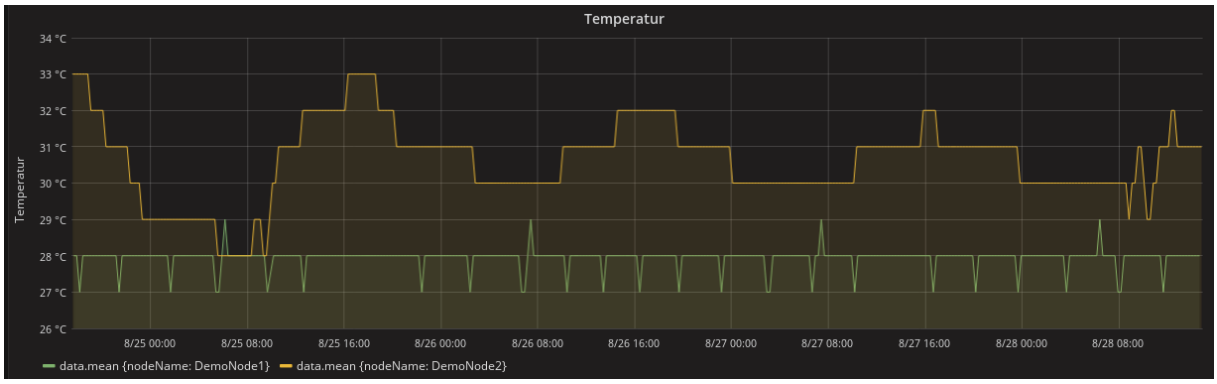


Abbildung 4.2: Die Temperatur von mehreren Sensoren kann über beliebige Zeiträume dargestellt werden.

Wir haben das Skript in `/home/vagrant/MQTT2Influx.py` gespeichert und durch `/etc/init/MQTT2Influx.conf` ausgeführt:

```
#
# This task is run on startup to make sure all from MQTT is
   stored in Influx.

description     "MQTT2Influx"

start on startup
start on started networking

task
exec python3 /home/vagrant/MQTT2Influx.py
```

Warnung

In der aktuellen Version wird das Skript als `root` ausgeführt. In einem produktiven System sollte das Skript von einem Nutzer mit weniger Rechten ausgeführte werden, um mögliche Angriffe auf das System zu erschweren.

Das Anzeigen der Daten in Grafana ist analog zur Grafana Dokumentation[40] durchzuführen. Abbildung 4.2 zeigt eine Beispieldarstellung der Temperatur.

4.9 Alternative Aufbaumöglichkeiten

Andere Möglichkeiten LoRa zu verwenden (zum Teil oben bereits erklärt) umfassen, die Verwendung des Conduit als:

- Forwarder (unsere Hauptnutzung),
- als Server (kurz erläutert) oder
- als App-Server.

Die verwendete open-source Lösung kann auch separat als LoRa-Server und LoRa-App-Server verwendet werden.

[40] `http://docs.grafana.org/features/panels/graph/`

5 Sigfox Einsatz

In diesem Kapitel wird der Aufbau und die Inbetriebnahme der Sigfox-Technologie beschrieben. Ziel ist es, einen Temperatursensor auszulesen und die gemessene Temperatur über das Sigfox-Netzwerk zu übertragen. Die Daten sollen anschließend in eine Datenbank gespeichert und visualisiert werden.

Sigfox[1] ist ein französisches Unternehmen, das ein globales Schmalbandkommunikationsnetzwerk für das Internet der Dinge aufbaut. Ein Sigfox-Gerät soll laut Hersteller folgende Kriterien erfüllen:

- niedriger Energieverbrauch
- geringe Kosten (in der Größenordnung von 1€/Jahr - 1€/Monat pro Gerät) [2])
- Kompatibilität mit anderen Kommunikationslösungen

Das Netzwerk ist derzeit in ca. 40 Ländern verfügbar[3] (Stand Februar 2018) und soll in Zukunft global verfügbar werden.

In Österreich gibt es derzeit keine Möglichkeit, Sigfox regulär zu betreiben, daher haben wir unsere Tests im benachbarten Bayern durchgeführt. Bei einer Anfrage an Sigfox, ob es eine Möglichkeit gibt, das Netz in Salzburg zu betreiben, haben wir keine Antwort erhalten. Es gibt jedoch einen Emulator, mit dem man den Empfang simulieren kann. Mehr dazu in Kapitel 5.9 auf Seite 57.

Das Unternehmen sucht in jedem Land Partner, mit dem das Netzwerk betrieben wird. In Österreich gab es bisher Pilotprojekte mit ORS[4][5] und Evolaris.[6]

5.1 Hardware

Verwendet wird das SiPy-Modul[7] der Firma Pycom. Das SiPy ist ein 55 x 20 mm großer Chip und unterstützt Sigfox, WiFi sowie Bluetooth. Programmiert wird das Modul mit Micropython (Python für Microcontroller). Der Chip kann auf ein so genannten Shield gesteckt werden, um ihn mit weiteren Features zu erweitern:

- Zur Durchführung des Firmware-Updates verwenden wir das Expansion Board v2.1A[8]
- Zur Messung der Temperatur verwenden wir das Pysense[9]

Das SiPy-Modul wird mit dem RGB-LED nach oben zum Micro-USB-Anschluss in das Pysense gesteckt. Zur Ergänzung verwenden wir ein transparentes Gehäuse[10] mit passender Antenne.[11] Zusammengebaut sieht das Ganze wie in Abbildung 5.1 aus:

[1] www.sigfox.com
[2] https://ask.sigfox.com/questions/574/subscription-price.html
[3] https://www.sigfox.com/en/coverage
[4] http://www.ors.at/de/presse/pressemitteilungen/pilotprojekt-mit-sigfox-netzwerk-in-oesterreich-167/
[5] https://futurezone.at/digital-life/orf-tochter-ors-steigt-ins-internet-der-dinge-business-ein/122.633.858
[6] https://www.evolaris.net/de/press/pilotprojekt-mit-sigfox-netzwerk-in-oesterreich-vor-dem-start/
[7] https://www.pycom.io/product/sipy/
[8] https://pycom.io/hardware/expansion-board-2-0-specs
[9] https://www.pycom.io/product/pysense/
[10] https://pycom.io/product/pycase-clear/
[11] https://pycom.io/product/lora-antenna-kit/

Abbildung 5.1: SiPy im Gehäuse mit Antenne

Information

Für das IP67 Case[a] gibt es ein eigenes Antennenkabel[b] für wasserdichte Konstruktionen. Für die Antenne muss selbst ein Loch gebohrt werden.[c]

[a] https://pycom.io/product/ip67-case/
[b] https://pycom.io/product/ip67-antenna-cable/
[c] https://forum.pycom.io/topic/1348/casing-for-the-pytrack-and-pysense-together-with-lopy

Zum Einrichten der SiPy-Hardware verwenden wir Windows 10 und orientieren uns an dem Getting started-Guide von Pycom[12] und Sigfox.[13]

5.2 Inbetriebnahme

Dieses Kapitel beschreibt die einmalig nötigen Schritte, um den Sensor in Betrieb zu nehmen.

5.2.1 Firmware Upgrade

Wir empfehlen, vor Verwendung ein Firmware-Update durchzuführen, da laufend Verbesserungen und neue Features entwickelt werden. Dazu werden für alle gängigen Betriebssysteme Softwaretools von

[12] https://docs.pycom.io/pycom_esp32/pycom_esp32/getstarted.html
[13] http://makers.sigfox.com/getting-started/

Pycom zur Verfügung gestellt.[14]

Information

Die aktuelle Firmware-Version des SiPy kann mit folgendem Befehl ausgelesen werden (zur Verbindung mit dem SiPy siehe unten):

```
os.uname()
```

Information

Es ist darauf zu achten, dass keine Jumper beim Transport abgefallen sind.

Information

Bei der Installation des Windows-Softwaretools wird die ausführbare Datei nur im Startmenü erstellt, nicht jedoch im Installationsordner.

Auch beim Pysense sollte die Firmware aktualisiert werden. Dazu stellt Pycom ein weiteres Softwaretool zur Verfügung.[15]

5.2.2 Kommunikation mit SiPy

Es gibt mehrere Möglichkeiten zur Kommunikation mit dem SiPy:

- Verbindung über USB-Kabel[16]
- Verbindung über WLAN (per FTP[17] oder Telnet[18])

Für die Kommunikation über USB und Telnet kann die Software PuTTy[19] oder der Text- und source-code-Editor Atom[20] verwendet werden. Die von Pycom empfohlene Variante mit Atom können auch wir empfohlen, da man damit das CLI verwenden und die Dateien am Flashspeicher per USB und WLAN synchronisieren kann. Das Plugin Pymakr kann außerdem mit Visual Studios verwendet werden, diese Variante wurde von uns jedoch nicht getestet.

Will man Daten über FTP synchronisieren, benötigt man ein geeignetes FTP-Programm wie beispielsweise Filezilla oder WinSCP.

Information

Standard-Zugangsdaten für SiPy zusammengefasst:

- SSID: sipy-wlan-xxxx
- IP-Adresse: 192.168.4.1
- WLAN Passwort: www.pycom.io
- Benutzername: micro
- Passwort: python

[14] https://docs.pycom.io/chapter/gettingstarted/installation/firmwaretool.html
[15] https://docs.pycom.io/chapter/pytrackpysense/installation/firmware.html
[16] https://docs.pycom.io/chapter/toolsandfeatures/repl/serial.html
[17] https://docs.pycom.io/chapter/toolsandfeatures/FTP.html
[18] https://docs.pycom.io/chapter/toolsandfeatures/repl/telnet.html
[19] http://www.putty.org/
[20] https://docs.pycom.io/chapter/pymakr/installation/atom.html

Falls das standardmäßig aktivierte WLAN nicht benötigt wird oder aus Sicherheits- oder Energiespargründen deaktiviert werden soll, gibt es dazu folgende Befehle:

```
from network import WLAN
wlan = WLAN()
wlan.deinit()
```

Eine Möglichkeit zur Änderung des WLAN-Passwortes ist uns nicht bekannt.

Bei Verwendung von PuTTY zum Aufbau einer USB-Verbindung benötigt man die Portnummer des Anschlusses (z. B. COM3). Diese kann im Geräte-Manager unter Anschlüsse oder im Atom Editor unter "More → Get serial Portsäusgelesen werden.

Information

Damit der Verbindungsaufbau in PuTTY funktioniert, muss die Baudrate (Speed) von 9 600 auf 115 200 geändert werden.[a]

[a] https://forum.pycom.io/topic/1543/solved-cannot-connect/4

5.3 Sensoren auslesen

Um die Sensoren des Pysense verwenden zu können, müssen zuerst die Bibliotheken [21] in das lib-Verzeichnis des Flashspeichers per FTP o.ä. hochgeladen werden. Die Bibliotheken erhält man hier: https://github.com/pycom/pycom-libraries

Information

Mit dem Pysense können neben Temperatur auch die Luftfeuchtigkeit, Beleuchtungsstärke, Luftdruck/Höhe und Beschleunigung gemessen werden.

Die Verwendung der Bibliotheken und das Auslesen der Sensoren wird im nachfolgenden Listing dargestellt. Hier werden die Messergebnisse mit print ausgegeben. Grundsätzlich muss man die Bibliotheken der Sensoren mit ïmportëinbinden und kann anschließend mit dem gleichnamigen Befehl die Messwerte auslesen.

```
# Library fuer Pysense
from pysense import Pysense
py = Pysense()

# Library fuer Temperatur und Luftfeuchtigkeit
from SI7006A20 import SI7006A20
si = SI7006A20(py)
print(si.temperature()) # Temperatur in Grad Celsius
print(si.humidity()) # Luftdruck in %

# Library fuer Lichtsensor
from LTR329ALS01 import LTR329ALS01
lt = LTR329ALS01(py)
print(lt.light())

# Library fuer Luftdruck- und Hoehensensor
from MPL3115A2 import MPL3115A2,ALTITUDE,PRESSURE
mp = MPL3115A2(py,mode=ALTITUDE) # Seehoehe in Meter
```

[21] https://docs.pycom.io/chapter/pytrackpysense/installation/libraries.html

```
print(mp.altitude())

mpp = MPL3115A2(py,mode=PRESSURE) # Luftdruck in Pascal
print(mpp.pressure())

# Library fuer 3-Achsen Beschleunigkeitsmesser
from LIS2HH12 import LIS2HH12
li = LIS2HH12(py)
print(li.acceleration())
print(li.roll())
print(li.pitch())
```

5.4 Registrierung bei Sigfox

Um die Sigfox Technologie nutzen zu können, muss jedes Modul mit seiner Device ID und der PAC Nummer bei Sigfox registriert werden.[22]

Die für die Registrierung benötigte Device ID und PAC Nummer sind laut Sigfox auf der Verpackung des gekauften Modules ersichtlich. Das können wir nicht bestätigen, da weder auf der Verpackung noch am Chip eine ID ersichtlich war. Die Daten können jedoch mit Hilfe folgender Kommandos ausgelesen werden:

```
from network import Sigfox
import binascii

# Wenn in der main.py noch nicht enthalten: Sigfox fuer RCZ1 (=
   Europa) initialisieren
sigfox = Sigfox(mode=Sigfox.SIGFOX, rcz=Sigfox.RCZ1)

# Sigfox Device ID anzeigen
print(binascii.hexlify(sigfox.id()))

# Sigfox PAC Nummer anzeigen
print(binascii.hexlify(sigfox.pac()))
```

Die registrierten Geräte können anschließend über eine zentrale Plattform, dem Sigfox Backend[23] verwaltet werden.

5.5 Übertragung von Daten über Sigfox

Warnung

Bevor man Daten über Sigfox sendet, muss auf eine korrekt angeschlossene Antenne geachtet werden, um Schäden an der Hardware auszuschließen.

Grundsätzlich unterscheidet Sigfox zwischen der Übertragung über das Sigfox-Netzwerk und der Device to Device-Verbindung, was im Programmcode definiert werden muss:

```
# Mode SIGFOX RCZ1 => Sigfox Europe (868 MHz)
sigfox = Sigfox(mode=Sigfox.SIGFOX, rcz=Sigfox.RCZ1)
```

[22] https://docs.pycom.io/chapter/tutorials/sipy/register.html
[23] https://backend.sigfox.com/auth/login

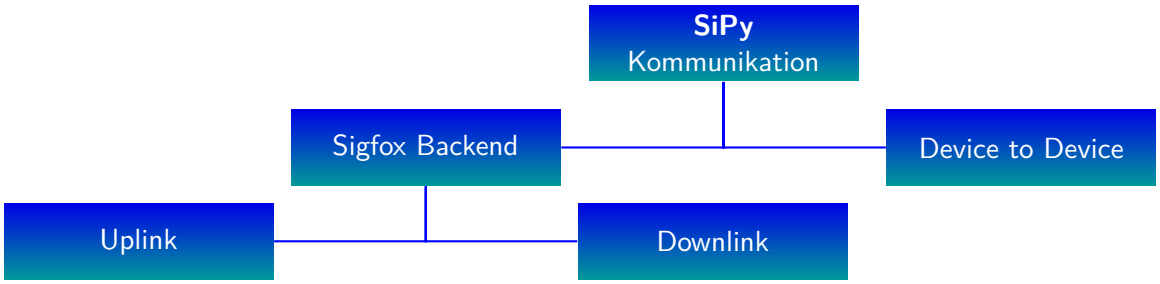


Abbildung 5.2: Zusammenfassung der Möglichkeiten zur Kommunikation über Sigfox.

```
# Mode FSK => device to device (912 MHz)
sigfox = Sigfox(mode=Sigfox.FSK, frequency=912000000)
```

Die tägliche Nachrichtenanzahl und Datenmenge ist begrenzt und kann je nach Land und Vertragspartner unterschiedlich sein. Sie beträgt aber maximal 12 Bytes und ist auf 140 Nachrichten pro Gerät und Tag beschränkt.[24]

Das Sigfox-Protokoll unterstützt bidirektionale Verbindungen. Das heißt, die Daten können in beide Richtungen übertragen, also gesendet (uplink) und empfangen (downlink) werden. Im Programmcode muss dies bei den Socket-Optionen eingestellt werden:

```
# Uplink => setsockopt = False
s.setsockopt(socket.SOL_SIGFOX, socket.SO_RX, False)

# Downlink => setsockopt = True
s.setsockopt(socket.SOL_SIGFOX, socket.SO_RX, True)
```

Weitere Programmcode-Beispiele werden auf der Pycom-Website ausführlich beschrieben.[25]

Es gibt mehrere Möglichkeiten die über Sigfox gesendeten Daten zu verarbeiten und weiter zu leiten. Die erste Möglichkeit ist, Daten mittels Downlink an einen anderen SiPy zu senden. Die zweite Möglichkeit ist es, Daten via Uplink an das Sigfox-Backend und anschließend an einen Server weiterzuleiten. Alternativ können die Daten vom im Sigfox-Backend als csv-Datei exportiert werden. Außerdem ist eine Device-to-Device-Kommunikation zwischen mehreren Sigfox-Endgeräten möglich. Abbildung 5.2 gruppiert diese Kommunikationsmöglichkeiten nochmals grafisch.

Die für uns entscheidende Variante ist der Uplink, da damit Temperaturdaten an einen Server oder zum Testen per E-Mail weitergeleitet werden können. Wir werden daher nur auf diesen Punkt genauer eingehen.

5.6 Das Sigfox-Backend

Was mit den übertragenen Daten passiert, muss im Sigfox-Backend unter dem Menüpunkt Callback definiert werden. Eine gute Erklärung zur Handhabung von Callbacks und weiteren Einstellungen im Backend liefert Sigfox im Backend selbst.[26]

Es können fünf unterschiedlich Callback-Varianten ausgewählt werden. Wir verwenden die Standardvariante „Custom Callback“, mit der beispielsweise Daten von der Sigfox-Cloud auf einen eigenen Server übertragen werden.

[24] https://ask.sigfox.com/questions/842/daily-data-rate.html
[25] https://docs.pycom.io/chapter/firmwareapi/pycom/network/sigfox.html
[26] https://resources.sigfox.com/ und https://makers.sigfox.com

Information

Zusätzlich zu den normalen Callbacks können in der Event Configuration[a] bestimmte Ereignisse eingestellt werden, die einen Callback auslösen, zum Beispiel bei Übertragungsfehlern, die durch eine fehlerhafte Sequenznummer im Sigfox Protokoll erkannt werden.[b]

[a] https://backend.sigfox.com/apidocs/event-callback?configSource=2
[b] https://confluence.sigfox.com/plugins/servlet/mobile#content/view/6132014

Im Callback-Menü können bestehende Standardvariablen wie {device} oder {time} verwendet werden. Diese wurden beim Testmail eingebunden, um deren Funktionsweise zu testen. Bei der Zeitvariable wird die Unixzeit[27] zurückgegeben und muss gegebenenfalls in eine lesbare Uhrzeit umgerechnet werden. Die GPS-Daten (lat, lng) sind sehr ungenau und stellen vermutlich nur den ungefähren Ort einer Basisstation dar.

Unter dem Punkt „Custom payload config“ kann eingestellt werden, wie die Sensordaten verarbeitet werden. In dem Feld wird der Name, die Länge in Byte und der Datentyp definiert. Bei unserer Temperaturübertragung werden beispielsweise 8 Bit übertragen und der Datentyp `int` verwendet.

In der Nachricht können die zur Verfügung stehenden Standardvariablen und die selbst definierten Variablen verwendet werden. Ein Beispiel für eine Testnachricht ist:

```
Testnachricht von Geraet {device}: Uhrzeit: {time}.
Die aktuelle Temperatur betraegt: {customData#temp} Grad Celsius.
```

Will man mehrere Werte gleichzeitig übertragen, erweitert man die Konfiguration einfach um eine weitere Variable, getrennt durch Leerzeichen. Das Feld zur Übertragung von Temperatur und Luftfeuchtigkeit könnte so aussehen:

```
temp::int:8 humidity::int:8
```

5.7 Test in Deutschland

Für den ersten Testversuch in Bayern haben wir zwei SiPys registriert und ein Callback via E-Mail definiert (siehe Abb. 5.3). Das heißt, dass jedes Mal, wenn ein Gerät Daten übermittelt, sendet die Sigfox-Cloud eine Verständigung per E-Mail an den/die angegebenen Empfänger gesendet.

Die Temperatur wurde mit folgendem Programmcode an die Sigfox-Cloud gesendet:

```
# Disable Heartbeat
import pycom
pycom.heartbeat(False)
pycom.rgbled(0x000000)

from network import Sigfox
import socket
import time

# Sensor for temperature
from SI7006A20 import SI7006A20
sen = SI7006A20()

# init Sigfox for RCZ1 (Europe)
sigfox = Sigfox(mode=Sigfox.SIGFOX, rcz=Sigfox.RCZ1)
# create a Sigfox socket
s = socket.socket(socket.AF_SIGFOX, socket.SOCK_RAW)
```

[27] http://www.unixtime.de/

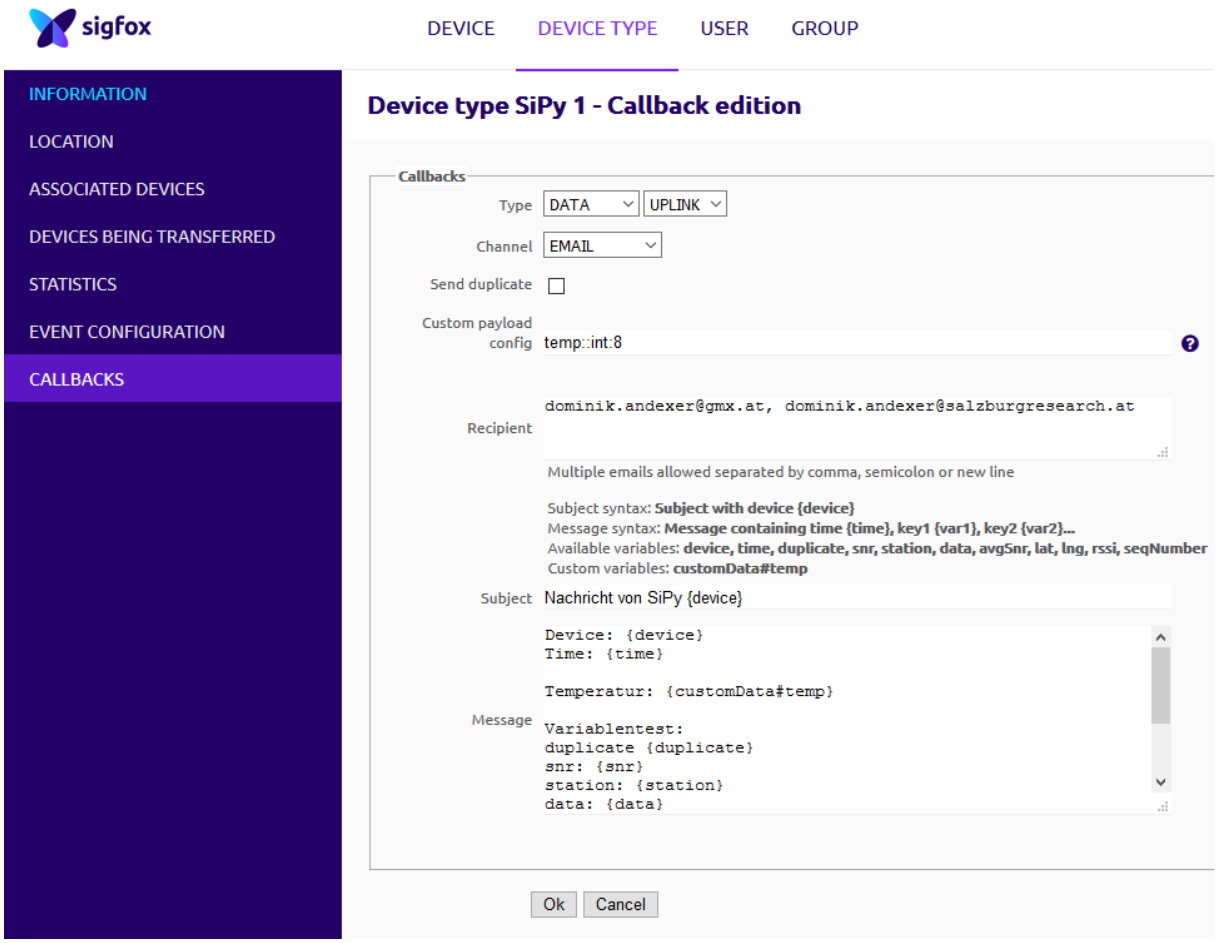


Abbildung 5.3: Einstellungen eines Callbacks via E-Mail

```
# make the socket blocking
s.setblocking(True)
# configure it as uplink only (Mode = False)
s.setsockopt(socket.SOL_SIGFOX, socket.SO_RX, False)

# Send temperature
temp = int(sen.temperature())

# Filter bad data
if temp > 255:
    temp = 255
if temp < 0:
    temp = 0
s.send(bytes([temp]))

# Blink blue when it sends data
pycom.rgbled(0xff3)
time.sleep(1)
pycom.rgbled(0x0)
```

Der Test war erfolgreich, die Antenne musste jedoch ins Freie beziehungsweise auf das Autodach gestellt werden. Die vordefinierten Callbacks per E-Mail wurden versendet und auch die Zusatzinfor-

mationen im Backend wie Average Rssi, SNR und Last seen im Backend wurden erstmalig angezeigt, wie in Abbildung 5.4 ersichtlich ist.

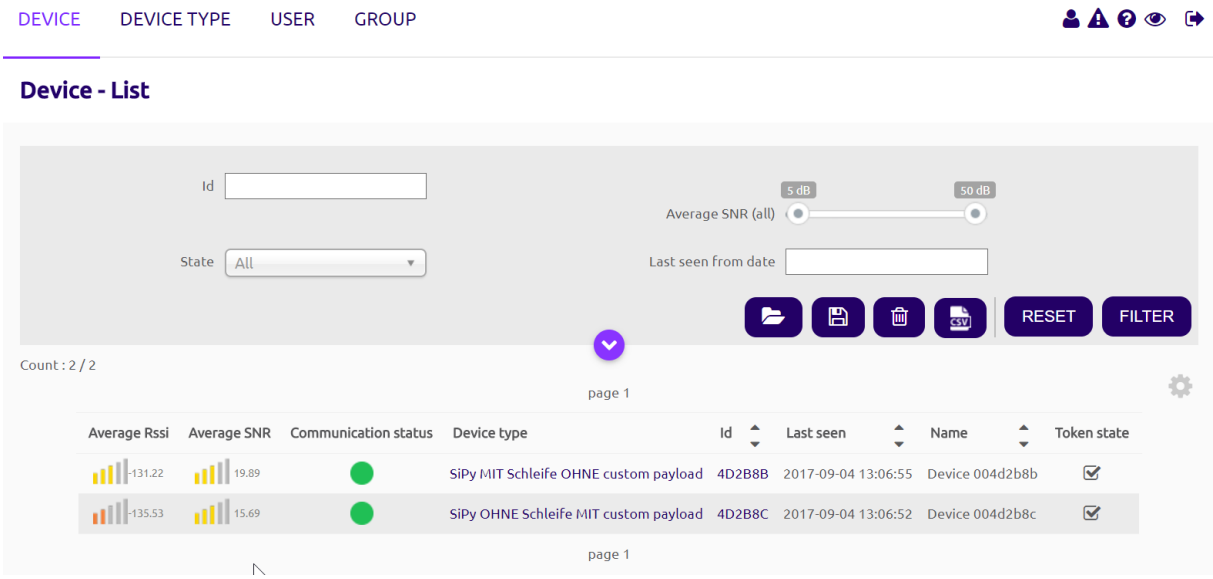


Abbildung 5.4: Geräteliste nach erfolgreichem Test

Information

Der Communication status bleibt grau bis in den Geräteoptionen (Klick auf das jeweilige Gerät und rechts oben auf Edit) ein Intervall eingestellt wird (keep-alive period).[a] Der keep-alive-Status prüft, ob zumindest eine Nachricht im angegebenen Intervall empfangen wird.

[a] https://confluence.sigfox.com/display/OPSDB/Set+a+keep+alive+delay

5.8 Visualisierung der Messwerte

Ziel ist es, die gemessenen Temperaturdaten in eine Datenbank (Influx) zu speichern und anschließend grafisch darzustellen. Dazu haben wir einen Python Flask Server[28] eingerichtet, der HTTP requests (eingehende POST requests vom Sigfox Backend) verarbeiten kann. Als Installationshilfe gibt es ein Youtube-Video[29] und ein Tutorial[30]. Abbildung 5.5 stellt den Ablauf dar.

Abbildung 5.5: Interaktion der Kommunikation eines Messwertes vom Sensor bis zur Visualisierung

Information

Um Daten erzeugen zu können, ohne in Sigfox-Reichweite zu sein, verwenden wir das Firefox Addon „HttpRequester“ in Version 2.1.1[a].

[a] https://addons.mozilla.org/de/firefox/addon/httprequester/versions/

[28] https://github.com/Bucknalla/sigfox-python-flask
[29] https://www.youtube.com/watch?v=98JY6MvumVs
[30] http://flask.pocoo.org/docs/0.12/installation/

Information

Die gesamte Software wurde sowohl am Bürorechner mit Windows 10 als auch am Labor-Server mit Linux erfolgreich getestet. Bei Verwendung von Windows muss Python zuerst heruntergeladen und installiert werden[a] Wir verwenden die Version 3.6.2. In Linux-Distributionen wie beispielsweise Ubuntu ist Python bereits enthalten.

[a] https://www.python.org/

Die Installation der benötigten Pakete `influx`, `flask` sowie `virtualenv` zur Einrichtung einer unabhängigen Python-Umgebung erfolgt durch:

```
$ pip install influxdb
$ pip install flask
$ pip install virtualenv
```

Installation der benötigten Programme unter Linux erfolgt durch:

```
sudo apt-get install flask
sudo apt-get install python-virutalenv
sudo apt-get install influxdb
sudo wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux
   -amd64.zip
sudo unzip ngrok-stable-linux-amd64.zip
sudo rm ngrok-stable-linux-amd64.zip
```

Danach wird ein Ordner und die virtuelle Python-Umgebung flask erstellt. Die Umgebung wird durch Ausführen der Datei `activate` im Ordner Scripts (unter Linux bin) gestartet. Anschließend kann der nachfolgend angeführte Programmcode in flask2influx.py ausgeführt werden.

```
C:\>mkdir myproject
C:\>cd myproject
C:\myproject>virtualenv flask
C:\myproject>cd flask\Scripts # unter Linux "bin"
C:\myproject\flask\Scripts>activate
C:\myproject>python flask2influx.py
```

Programmcode der flask2influx.py:

```
# Flask initialisieren
from flask import Flask, request, jsonify, render_template
import datetime

# Influx initialisieren
from influxdb import InfluxDBClient

app = Flask(__name__)

@app.route('/data/<sensor>', methods=['POST']) # sensor 1 =
   temperature
def add_message(sensor):
        # grab the json data from the POST request and write in
           variables
        content = request.json
        device = str(content['device'])
        time = int(content['time'])
        time = # convert unix time in human readable time
```

```
        datetime.datetime.fromtimestamp(time).strftime('%Y-%m-%d␣
           %H:%M:%S')
        temp = int(content['temp'])
        rssi = int(float(content['rssi']))
        snr = int(float(content['snr']))
        # localhost oder al40-182
        influx= InfluxDBClient("localhost", 8086, "", "", "sigfox
           ")


    json_body = [
            {
                "measurement": "data",
                "tags": {
                    "appName": device,
                    "nodeName": "",
                },
                "time": time,
                "fields": {
                    "temp": temp,
                    "rssi": rssi,
                    "SNR": snr
                }
            }
        ]

        influx.write_points(json_body) # write json body in
           Influx-DB
        print(json_body)
        return('', 200) # return http Code 200 (= OK)

if __name__ == '__main__':
    app.run(host= '0.0.0.0',debug=True)
```

Da unser Laborrechner keine öffentliche IP-Adresse besitzt und durch eine Firewall geschützt ist, kann die Sigfox-Cloud keine Daten an ihn übermitteln. Wir benötigen daher ein weiteres Tool namens ngrok[31]. Es ermöglicht dem Sigfox-Backend, Daten an den Python Flask Server zu senden, selbst wenn dieser keine öffentliche IP-Adresse besitzt oder hinter einer Firewall steht. Dabei stellt unser Laborrechner eine Verbindung mit dem ngrok-Cloud-Service her, der den Datenverkehr von Sigfox akzeptiert und an die lokale Adresse (localhost) weiterleitet.

Öffnen der ngrok-Konsole und Eingabe von folgendem Befehl:

```
$ ./ngrok http 5000
```

Dieser Befehl legt fest, dass localhost über Port 5000 öffentlich durch eine Tunnelverbindung erreicht werden kann. Die angegebene Adresse unter "forwarding"muss in den Callback-Einstellungen im Sigfox-Backend eingegeben werden. In unserem Fall wird eine Subadresse /data/1 (für Sensor 1) festgelegt, wie in Abbildung 5.6 gezeigt.

Erst wenn ngrok und der flask-Server laufen, kann ein http POST request durchgeführt werden. Die im json-body übertragenen Werte werden in die influx-Datenbank gespeichert und dann mit Graflux virtualisiert.

[31] https://ngrok.com/download

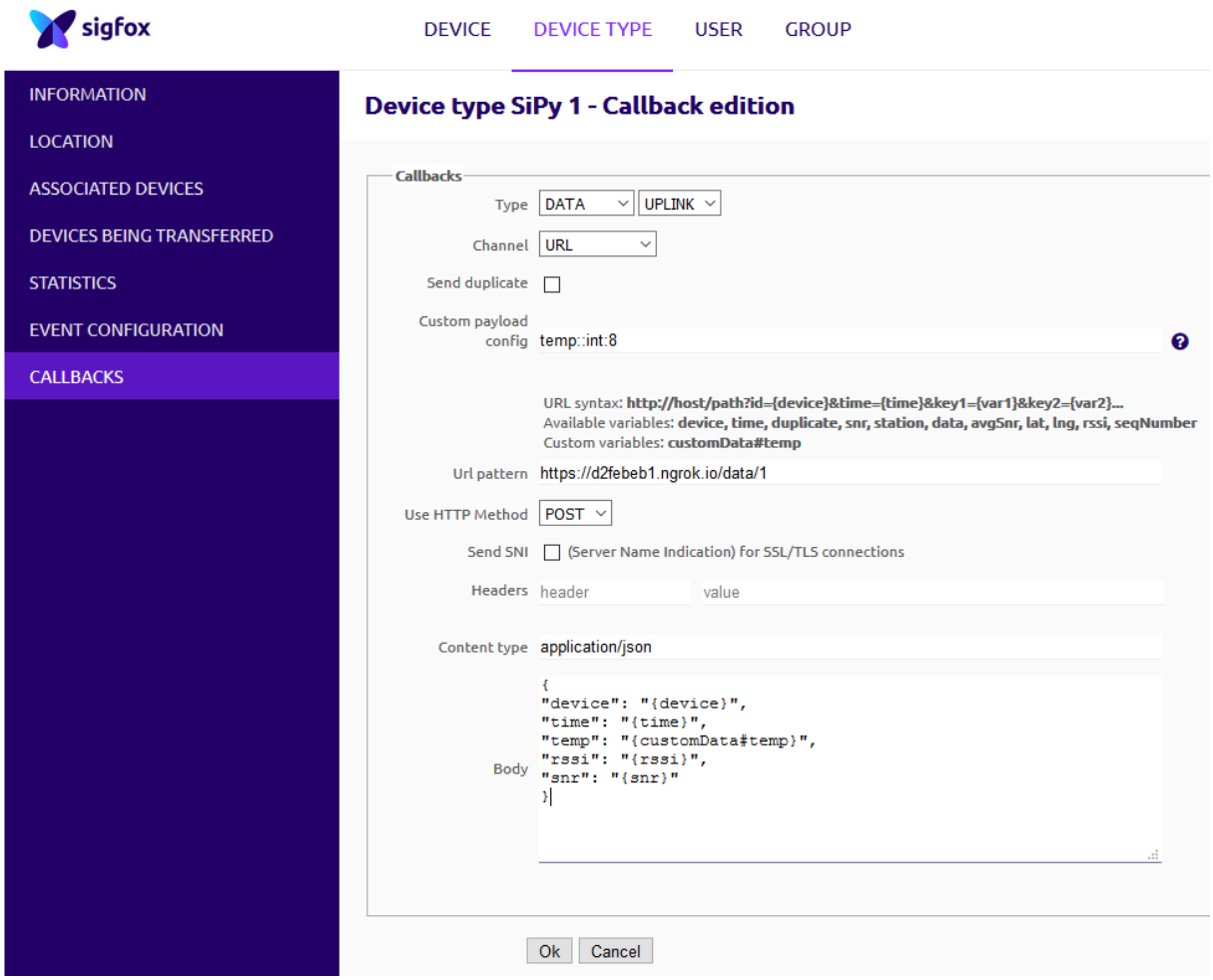


Abbildung 5.6: Einstellungen im Sigfox Backend (http POST request)

Damit der Python Flask Server beim Start und aufrechter Netzwerkverbindung automatisch gestartet wird, haben wir ein Script geschrieben und in `/etc/init/Flask.conf` gespeichert:

```
# This task is run the python flask server to receive messages
   from sigfox backend

start on startup
start on started networking

task
# activate virtualenv flask
exec . /home/flask/bin/activate
# start flask server
exec sudo python /home/flask/flask2influx.py
```

Warnung

Entgegen der Konfiguration in diesem Beispiel sollte der Code nicht als `root`, sondern als Nutzer mit weniger Rechten ausgeführt werden. Wie dies im Detail umgesetzt werden sollte, hängt davon ab, wie die Daten weiterverarbeitet werden sollen.

Information

Bei der freeware-Version von ngrok ist die URL variabel. Deshalb müsste die Adresse im Sigfox-Backend bei jedem Neustart händisch geändert werden, was die Abwicklung kompliziert macht.

Mit unserem Prototyp haben wir erfolgreiche Übertragungstests in Bayern durchgeführt. Dabei wurde die Temperatur mit dem Pysense gemessen und über Sigfox übertragen. Die Daten wurden anschließend vom Sigfox-Backend an unseren Server übertragen, in eine Influx-Datenbank geschrieben und mit Grafana visualisiert.

5.9 Alternative für Tests außerhalb der Sigfox-Netzabdeckung

Derzeit ist es nicht möglich, eine eigene Sigfox Basisstation zu betreiben oder Sigfox zu testen, wenn keine Netzabdeckung im Testgebiet vorhanden ist. Außerdem gibt es derzeit keine Möglichkeit, einen Callback ohne Sigfox-Empfang auszulösen. In solchen Fällen könnte das Sigfox Network Emulator Kit (SNEK)[32] [33] interessant sein. Das Paket besteht aus einem USB-Gerät mit dazugehöriger Software und simuliert das Sigfox Netzwerk. Es kann von Sigfox durch Registrierung und Projektbeschreibung erworben werden. Wir haben das SNEK nicht selbst getestet, da wir einfach reale Tests in Bayern durchführen konnten.

[32] `https://github.com/sigfox/sigfox-snek-howto`
[33] `https://www.sigfox.com/en/support/download`

6 NB-IoT Einsatz

NB-IoT ist eine Schmalband-Technologie, die von der 3GPP[1] standardisiert wurde und auf dem Mobilfunkstandard Long Term Evolution (LTE, Release 13) aufbaut.[2]

NB-IoT erlaubt die Übertragung von IP Paketen über das Netz eines Mobilfunkanbieters. Hierzu ist ein Vertrag mit dem Mobilfunkanbieter notwendig. Für den Vertrag fallen Kosten an (im aktuellen Testbetrieb in der Größenordnung von 1 €/Monat/Gerät), dafür stellt der Mobilfunkanbieter seine Kommunikationsinfrastruktur zur Verfügung. Dadurch muss die Infrastruktur nicht selbst betrieben werden und ist überall dort verfügbar, wo der Mobilfunkanbieter sie anbietet. Im Regelfall heißt dies landesweit und mit möglichem Roaming auch im Ausland (sofern dort NB-IoT aktiv ist).

Information

Es gibt in NB-IoT auch einen energieeffizienteren Modus, bei dem keine vollständigen IP Pakete versendet werden, dafür muss allerdings das Netz konfiguriert werden. Ab wann dies von den Mobilfunkanbietern unterstützt wird, ist noch unklar.

6.1 Infrastruktur

Im Folgenden zeigen wir eine kurze Übersicht des Standes zur NB-IoT Verfügbarkeit in Österreich.

6.1.1 A1 Telekom Austria

Die Telekom Austria testet NB-IoT im Live-Betrieb.[3] Wann dieser allerdings für die Öffentlichkeit geöffnet wird, ist uns nicht bekannt. Mit Ende 2017 bestimmt die Telekom Austria einen Ort für eine NB-IoT Testgegend.

6.1.2 T-Mobile Austria

Die Deutsche Telekom bietet ihren IoT-Service auch in Kombination mit ihrer „Cloud der Dinge“ an. Hier werden die Daten in der Cloud gespeichert und können über ein Webinterface (und REST-Schnittstelle) visualisiert werden[4]. Mit Juli 2017 war dieses Angebot nur in begrenzten Testregionen in Deutschland verfügbar.

Laut NB-IoT Workshop am 2017-11-23 in St. Pölten wird der Ausbau des NB-IoT Netzes in Österreich der T-Mobile im Jahr 2018 stattfinden. Es soll laut plan im Spätsommer/Herbst 2018 abgeschlossen sein.

6.1.3 Hutchison Drei Austria

Zu Plänen von Hutchison Drei in Österreich NB-IoT anzubieten, liegen uns keine Information vor.

[1] http://www.3gpp.org
[2] http://www.3gpp.org/news-events/3gpp-news/1733-niot
[3] https://www.telekomaustria.com/de/newsroom/2016-11-16-telekom-austria-group-testet-als-erster-netzbetreiber-in-osterreich-nb-iot-im-lte-live
[4] https://m2m.telekom.com/de/unser-angebot/narrowband-iot/

6.1.4 Test-Möglichkeiten in Bayern

Um NB-IoT zu testen, besteht in naher Zukunft eventuell die Möglichkeit diese Tests in Bayern durchzuführen, da dort unter Umständen die Bereitstellung von NB-IoT eher stattfindet als in Österreich. Tekmodul verwaltet eine Liste mit deutschen Städten, in denen NB-IoT Live geschaltet ist.[5] Mit Ende 2017 ist allerdings keine Möglichkeit zum Testen in der näheren Umgebung gegeben.

6.2 Hardware

Da die Anbieter der Infrastruktur das Netz noch nicht vollständig aktiviert haben, ist die Hardwareauswahl für NB-IoT zur Zeit beschränkt. Hier eine kleine Auswahl der Modulhersteller zu welchen wir Kontakt haben:

6.2.1 TekModul

TekModul vertreibt Entwicklerkits für NB-IoT.[6] Diese basieren auf Quectel chips.[7] Es gibt diese Kits basieren auf einem GSM oder auf einem LTE Basiskit. Wir haben beide Kits zur Demonstration vor Ort.

6.2.2 Pycom

PyCom bietet NB-IoT Module an; diese sind aber noch nicht verfügbar. Hierbei handelt es sich um ein kombiniertes NB-IoT/WLAN/Bluetooth Modul[8] und ein kombiniertes NB-IoT/LoRa/Sigfox/WLAN/Bluetooth Modul.[9] Wir haben zur Demonstration ein FiPy vor Ort. Für eine Datenübertragung fehlt nur die Abdeckung des Raumes Salzburg durch einen Telekommunikationsanbieter.

6.2.3 Weitere Module

Weitere NB-IoT Module sind zum Beispiel:

- Arduino NB-IoT Shield[10]
- ublox SARA-N2 series[11]
- Quectel BC95[12]

6.3 Demonstrator

Wir können die Funktionsweise von NB-IoT am Beispiel von zwei Entwicklerkits[13] von Tekmodul demonstrieren. Zu deren Einsatz haben wir uns unter anderem auf den NB-IoT Workshops in Berlin und St. Pölten mit anderen Entwicklern ausgetauscht. Dort war für Testzwecke auch eine NB-IoT Basisstation verfügbar und wir konnten NB-IoT erfolgreich testen.

Da NB-IoT lizenzierte Frequenzen verwendet, können nur die jeweiligen Frequenzinhaber Basisstationen basierend auf NB-IoT betreiben. Wir sind daher auf den Betrieb dieser Basisstationen von den Telekommunikationsanbietern angewiesen. Sobald der erste Anbieter in Salzburg sein NB-IoT

[5] https://www.tekmodul.de/liveschaltung-lte-cat-nb1/
[6] https://www.tekmodul.de/lte-cat-nb1-live-sonderaktion-von-tekmodul/
[7] http://www.quectel.com/product/gsmevb.htm
[8] https://pycom.io/product/gpy/
[9] https://pycom.io/product/fipy/
[10] http://iot-university.org/nb-iot-shield-arduino/
[11] https://www.u-blox.com/en/product/sara-n2-series
[12] http://www.quectel.com/product/bc95.htm
[13] https://www.quectel.com/product/gsmevb.htm und https://www.quectel.com/product/umtsevb.htm

Netz aktiviert (und uns eine SIM-Karte dafür zur Verfügung stellt) können wir die Übertragung von Daten über NB-IoT auch praktisch in Salzburg demonstrieren.

7 GPS Demonstrator

Dieses Kapitel beschreibt den Aufbau und die Inbetriebnahme eines GPS-Sensors. Wir bauen diesen Prototypen auf Basis von LoRa, da wir einen eigenen LoRa Gateway betreiben und somit Tests im Labor durchführen können. Alternative könnte der in diesem Kapitel beschrieben Prototyp auch auf Basis von Sigfox gebaut werden.

7.1 Hardware

Zur Messung der GPS-Daten verwenden wir das Shield Pytrack[1] der Firma Pycom. Der Chip enthält neben dem GPS-Sensor einen 3-Achsen 12-bit Beschleunigungssensor.

Zusätzlich verwenden wir folgende Hardware:

- LoPy[2]
- Staub/Wasser-geschütztes Gehäuse IP67[3]
- IP67-Antennenkabel[4]
- LiPo-Akku mit 3,7V und 2000mAh[5]

Das Staub- und Wassergeschützte IP67 Gehäuse besitzt kein Loch für den Antennenanschluss. Aus diesem Grund haben wir selbst eine Bohrung durchgeführt und die Komponenten, wie in Abbildung 7.1 ersichtlich, zusammengebaut. Ein 65 mm x 40 mm Akku passt exakt in das Gehäuse.

7.2 Inbetriebnahme

Zuerst wird die Firmware des LoPys[6] und Pytracks[7] auf den neuesten Stand gebracht.

Die Kommunikation mit dem Pycom-Gerät wurde in Kapitel 5.2.2 beschrieben.

7.3 Sensoren auslesen

Zur Verwendung der Sensoren müssen die Programmbibliotheken des Beschleunigungssensors (LIS2HH12) und des GPS-Sensors (L76GNSS) in den lib-Ordner am flash-Speicher des Pytracks geladen werden.[8] Bei der Verwendung der Sensoren orientieren wir uns an der pycom API-Dokumentation[9] Das nachstehende Programmbeispiel demonstriert die Verwendung des Beschleunigungs- und GPS-Sensors:

[1] https://pycom.io/product/pytrack/
[2] https://pycom.io/hardware/lopy-specs/
[3] https://pycom.io/product/ip67-antenna-cable/
[4] https://pycom.io/product/ip67-antenna-cable/
[5] https://eckstein-shop.de/LiPo-Akku-Lithium-Ion-Polymer-Batterie-37V-2000mAh-JST-PH-Connector
[6] https://docs.pycom.io/chapter/gettingstarted/installation/firmwaretool.html
[7] https://docs.pycom.io/chapter/pytrackpysense/installation/firmware.html
[8] https://github.com/pycom/pycom-libraries/tree/master/pytrack/lib
[9] https://docs.pycom.io/chapter/pytrackpysense/apireference/pytrack.html

Abbildung 7.1: LoPy im Staub/Wasser-geschützten IP67 Gehäuse

```
# Library fuer 3-Achsen Beschleunigungssensor
from LIS2HH12 import LIS2HH12
sen = LIS2HH12()

print(sen.acceleration()) # Returns a tuple with the 3 values of
    acceleration
print(sen.roll()) # Returns a float in degrees between -180 and
    180
print(sen.pitch()) # Returns a float in degrees in the range -90
    to 90

# Library fuer GPS-Sensor
from L76GNSS import L76GNSS

gps = L76GNSS(timeout=10)
print(gps.coordinates()) # Returns a tuple with the longitude and
     latitude
```

Wir setzen beim GPS-Sensor das Timeout auf 10 Sekunden, damit auch bei fehlendem GPS-Empfang ein leeres Tuple zurückgegeben wird. Für einen möglichst energiesparenden Betrieb deaktivieren wir das RGB-LED und WLAN. Die GPS-Daten werden im 5-Minuten-Intervall gesendet.

Außerdem versetzen wir das Gerät für jeweils 5 Minuten in den Deep-Sleep-Modus[10], um den Energieverbrauch weiter zu senken.

Der Code der `main.py` am Pytrack zum Senden der GPS-Daten wird nachfolgend dargestellt.

```
import pycom
import machine
import socket
import time
import binascii
import struct
from network import LoRa
from machine import Timer
from L76GNSS import L76GNSS
from LIS2HH12 import LIS2HH12

# Disable WiFi
from network import WLAN
wlan = WLAN()
wlan.init()

# Disable Heartbeat
pycom.heartbeat(False)
pycom.rgbled(0x000000)

lora = LoRa(mode=LoRa.LORAWAN)
app_eui = binascii.unhexlify('9a7b46dcf93bae50')
app_key = binascii.unhexlify('196a565bd44e6de730c65cde7030cd98')

lora.join(activation=LoRa.OTAA, auth=(app_eui, app_key), timeout
   =0)

time.sleep(10)

s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
s.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)
s.setblocking(True)
# Read gps
gps = L76GNSS(timeout=10)

# Send gps data
coord = gps.coordinates() # returns a tuple with 2 values
if coord != (None, None):
    lat, lon = coord;
    print(lat,lon)
    coords_b = bytearray(struct.pack("2f", lat, lon)) # convert
       to bytearray
    s.send(coords_b) # Send last known coordinates
    print("message␣sent!")
else:
    print("No␣coordinates␣found!")
```

[10] https://docs.pycom.io/chapter/datasheets/boards/deepsleep/api.html

```
machine.deepsleep(300000) # deep-sleep mode for 5 minute
```

Warnung

Die in diesem Code genutzte Deep-Sleep Funktionalität[a] funktioniert mit Satand Februar 2018 noch nicht einwandfrei. In Zukunft sollte diese durch Updates von pycom funktionieren. Im Moment ist er Code daher nur für den Betrieb an einer ständigen Stromquelle geeignet.

[a] https://docs.pycom.io/chapter/datasheets/boards/deepsleep/api.html

Information

Der GPS-Sensor gibt ein Tuple mit zwei Float-Werten zurück (Längen- und Breitengrad). Wenn GPS-Daten gefunden werden, wandelt das Programm die beiden Float-Werte in ein Bytearray um. Die Umwandlung ist notwendig, um die Messergebnisse an das LoRa-Gateway übertragen zu können.

7.4 Visualisierung der GPS-Daten

Die Aufgabe des Skriptes `mqtt2text.py` ist es, aus der empfangenen Nachricht mit Hilfe von Regular Expressions die benötigten Informationen wie Timestamp und die Koordinaten herauszufiltern und in eine Textdatei zu schreiben. Beispielhaft sieht eine solche Datei wie folgt aus:

```
1517837803    2018-02-05T13:36:43.801534Z   47.823532
   13.040256
1517837901    2018-02-05T13:38:41.313531Z   47.823095
   13.040528
1517838202    2018-02-05T13:41:13.327499Z   47.823464
   13.039891
1518341645    2018-02-05T15:40:45.345293Z   47.822894
   13.040881
1517841799    2018-02-05T15:43:19.235821Z   47.822621
   13.040604
1517841976    2018-02-05T15:46:16.947123Z   47.822915
   13.041391
1517842096    2018-02-05T15:48:16.293193Z   47.822713
   13.040247
```

Das Format der Textdatei ist wie folgt definiert:

1. Spalte: Unix-Zeitstempel
2. Spalte: lesbares Datum/Zeit
3. Spalte: Längengrad
4. Spalte: Breitengrad

Der Code zum Filtern der Daten und zum Erstellen der oben angeführten Textdatei lautet:

```
import re
import base64
import struct
import time
import paho.mqtt.client as mqtt
```

```
# The callback for when the client receives a CONNACK response
   from the server.
def on_connect(mqtt, userdata, flags, rc):
    print("Connected␣with␣result␣code␣"+str(rc))

    # Subscribing in on_connect() means that if we lose the
       connection and
    # reconnect then subscriptions will be renewed.
    mqtt.subscribe("#")

# The callback for when a PUBLISH message is received from the
   server.
def on_message(mqtt, userdata, msg):
    print(msg.topic+"␣"+str(msg.payload))
    dataMatch = dataPattern.match(str(msg.payload))
    timeMatch = timePattern.match(str(msg.payload))
    appNameMatch = appNamePattern.match(str(msg.payload))
    nodeNameMatch = nodeNamePattern.match(str(msg.payload))
    rssiMatch = rssiPattern.match(str(msg.payload))
    loraSNRMatch = loraSNRPattern.match(str(msg.payload))

    # write data in textfile
    if dataMatch and appNameMatch:
        coords = base64.b64decode(dataMatch.group(1)) # decode
           base64
        lat, lon = struct.unpack("2f", coords) # convert
           bytearray to 2 float
        line = str(int(time.time())) + "\t" + str(timeMatch.group
           (1)) +
        "\t" + str(lat) + "\t" + str(lon) + "\n"

        f = open('/home/results/' + str(nodeNameMatch.group(1)) +
            '-position.txt', 'a')
        f.write(line)
        f.close()

# Regular Expression Pattern
timePattern = re.compile('.*"time":"([\d\-.:TZ]*)",.*')
appNamePattern = re.compile('.*"applicationName":"(GPS-App)",.*')
nodeNamePattern = re.compile('.*"nodeName":"([^"]*)",.*')
rssiPattern = re.compile('.*"rssi":([\d\-]*),.*')
loraSNRPattern = re.compile('.*"loRaSNR":([\d\-\.]*),.*')
dataPattern = re.compile('.*\"data\":\"(.*)\"}.*')

mqtt = mqtt.Client()
mqtt.on_connect = on_connect
mqtt.on_message = on_message

mqtt.connect("192.168.40.180", 1883, 60)
```

```
mqtt.loop_forever()
```

Die empfangenen Daten sind base64 codiert. Diese werden zuerst dekodiert und anschließend das Bytearray wieder in zwei Float-Werte umgewandelt.

Wir haben das Skript in `/home/mqtt2txt.py` gespeichert und durch `/etc/init/mqtt2txt.conf` sichergestellt, dass es beim Start des Servers automatisch ausgeführt wird:

```
# This task is to run mqtt2txt.py on startup

description     "mqtt2txt"

start on startup
start on started networking

task
exec python3 /home/vagrant/mqtt2txt.py
```

Warnung

Entgegen der Konfiguration in diesem Beispiel sollte der Code nicht als `root`, sondern als Nutzer mit weniger Rechten ausgeführt werden. Wie dies im Detail umgesetzt werden sollte, hängt davon ab, wie die Daten weiterverarbeitet werden sollen.

Die Visualisierung dar Daten ist in Abbildung 7.2 ersichtlich und erfolgt über einen Browser. Die Markierungen auf der Landkarte symbolisieren die zuletzt bekannten Positionen des Pytracks.

Die Datei unter `/home/ubuntu/code/Visualization/SimpleMapServer.py` überprüft Änderungen der Textfiles im angegebenen Pfad und aktualisiert somit die Karte, wenn neue GPS-Positionen in der Textdatei hinzugefügt werden.

Die Initialisierung erfolgt mit:

```
screen -r
python3 ./SimpleMapServer.py /home/results
# Beenden mit Strg+A -> D
# Textdateien befinden sich im Ordner /home/results
```

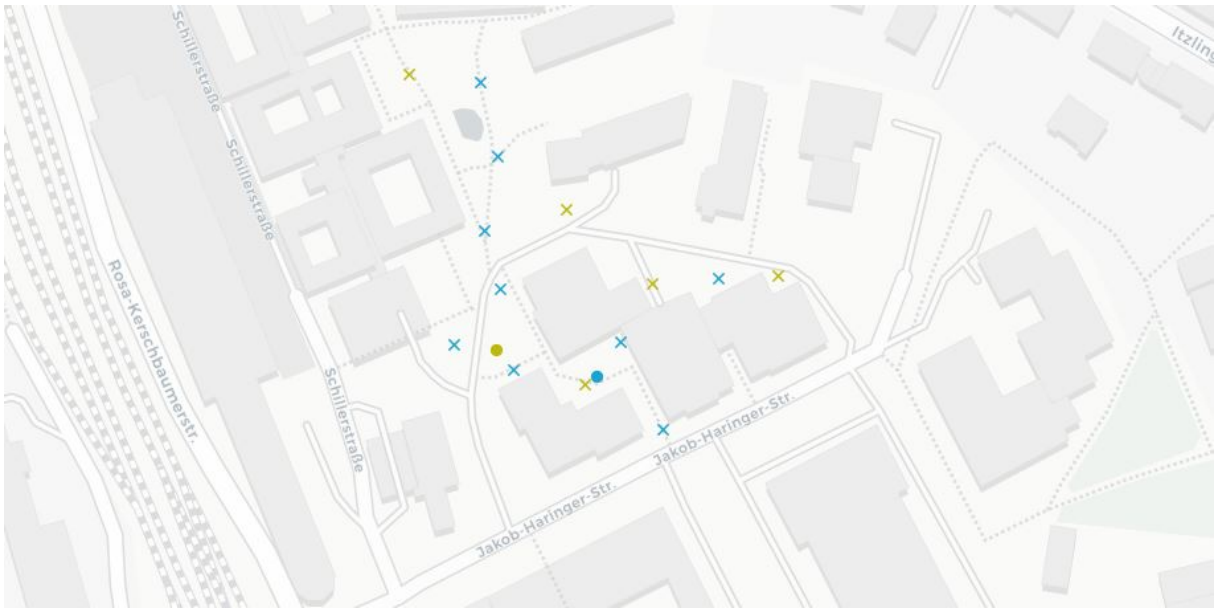


Abbildung 7.2: Visualisierung der GPS-Daten. Alte Positionen werden mit einem X dargestellt, die letzte bekannte Position wird mit einem Kreis

8 Messumgebung

In diesem Kapitel wird aufbauend auf bestehenden Werkzeugen der Salzburg Research eine Test- und Messumgebung für NB-M2M Kommunikation beschrieben. Es wird die dazu notwendige Messhardware erläutert und in Betrieb genommen. Die Messszenarien werden erstellt. Die Messabläufe implementiert und Messungen durchgeführt. Die Messumgebung erlaubt es anforderungsbezogene Untersuchungen der Qualitätseigenschaften von Schmalbandkommunikation durchzuführen. In diesem Task wurde auch eine Masterprojektgruppe (Binna, Katzinger und Ruetz) an der Fachhochschule Salzburg betreut.

8.1 Hardware und Infrastruktur

Dieser Abschnitt beschreibt die verwendete Hardware sowie die Infrastruktur und die Einrichtung der Messumgebung. Für die mobilen Messgeräte wird folgende Hardware verwendet:

- 2x Evaluation Boards mit EMB-LR1272E Modulen [5] [6]
  - Basierend auf Atmel SAMD20
  - Sendeleistung bis zu +19 dBm
  - Empfängersensitivität bis zu -137 dBm
  - Sehr energieeffizient ($\leq 3\,\mu A$ mit RTC, $\leq 1\,\mu A$ im XLP Sleep Mode)
  - RF-Chip: Semtech sx1272
- 2x Antennen EMB-AN868-BB24 [4]
  - Bi-band Antenne
  - Frequenz band: 868 MHz / 2,4 - 2,5 GHz
  - Impedanz: $50\,\Omega$
  - Antennengewinn: 2,1 dBi
  - Polarization: Linear
  - Abstrahlcharakteristik: Omnidirectional
- 2x Raspberry Pi 3
- 2x UMTS-Modul

Die Geräte Raspberry Pi kommunizieren mit den Evaluierungskarten über serielle USB-Verbindungen. Die Evaluation-Boards enthalten einen proprietären LoRa-Chip, der das Senden über die Antennen übernimmt. Zum Ausführen von Tests führen die Raspberry Pi mehrere Python-Skripts aus und sind mit einem VPN verbunden, um den Verwaltungszugriff aufrechtzuerhalten. Das VPN basiert auf OpenVPN und ermöglicht es dem Bediener, sich von entfernter Stelle mit den Messgeräten zu verbinden, und diese zu verwalten. Die Messgeräte verbinden sich auch über das VPN mit dem Datenbankserver, der alle Daten der Messungen sammelt (siehe Abbildung 8.1). Der Datenbankserver führt eine InfluxDB-Instanz aus und ist mit mehreren zusätzlichen Feldern erweitert, die für die Analyse der Daten erforderlich sind. Um eine Remoteverbindung zur Testumgebung herzustellen, muss eine etablierte OpenVPN-Verbindung vom Client-Computer zum VPN-Gateway bestehen.

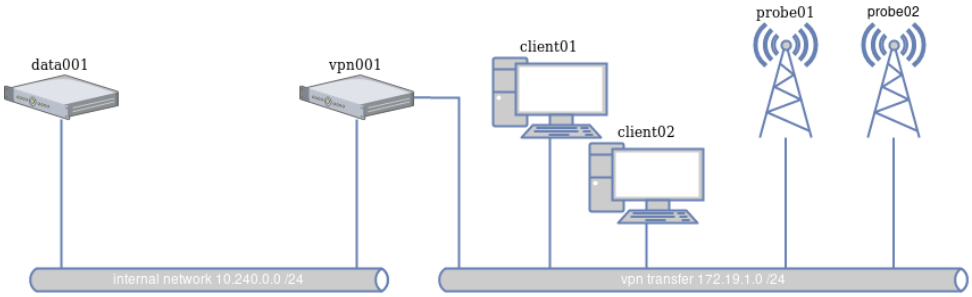


Abbildung 8.1: VPN infrastructure

Nach dem Verbinden mit dem VPN kann über SSH auf jede Maschine zugegriffen werden. Einzige Voraussetzung: Der SSH-Schlüssel muss in der Umgebung bereitgestellt werden. Abbildung 8.1 illustriert den Aufbau des VPN.

Der Datenbankserver und das VPN-Gateway werden in einer virtualisierten Umgebung ausgeführt. Auf das VPN-Gateway kann über eine öffentliche IP-Adresse zugegriffen werden. Um die Verbindung zu den Messgeräten aufrechtzuerhalten, wurden die Raspberry Pis mit einem UMTS Modul erweitert. Beim Verlassen der Ethernet-Verbindung werden die Raspberry Pis so konfiguriert, dass die VPN-Verbindung über die UMTS-Verbindung wiederhergestellt wird.

8.2 Software

In diesem Abschnitt werden die Softwarekomponenten der Messumgebung beschrieben. Die grundlegenden Softwarekomponenten werden in der Abbildung 8.2 gezeigt.

Die gesamte Software, wurde in Python geschrieben.[11] Die Datenbankkomponente, das Webinterface und das EBI LoRa-Gerät sind geschützte Komponenten, die nicht vom Projektteam entwickelt wurden. Alle anderen angezeigten Komponenten in der Abbildung 8.2 wurden vom Projektteam implementiert.

8.3 EBI-LoRa-Gerät

Diese Komponente basiert auf dem proprietären LoRa Evaluation Kit von Embit. Die Komponente ist über eine serielle USB-Verbindung mit den Raspberry Pis verbunden und die EBI Library stellt die Verbindung zwischen der Sender- und Empfängersoftware und dem LoRa Evaluation Kit her.

8.4 EBI-Bibliothek

Diese Bibliothek ist in Python geschrieben und verwendet die im Embit Binary Interface[7] definierten Befehle. Sie stellt die Klasse `Terminal` zur Verfügung, die bei der Installation die serielle Verbindung startet. Die Klasse stellt auch verschiedene Methoden zum Senden von Nachrichten über LoRa oder zum Ändern wesentlicher Einstellungen für das Testen von LoRa bereit, wie zum Beispiel Sendeleistung oder Codierrate.

8.5 Empfänger und Sender

Die Empfänger- und Senderskripts akzeptieren Parameter, die die verschiedenen LoRa-Einstellungen definieren. Der Sender verwendet diese Parameter, um den LoRa-Sender zu konfigurieren. Das Empfängerskript benötigt diese Parameter, um die Einstellungen in die Datenausgabe zu schreiben, sodass keine Testeinstellungen verloren gehen. Die Skripte haben ein Parameter-Flag, um die Testdaten entweder direkt in die Datenbank zu übertragen oder die Daten lokal in einer Textdatei zu speichern

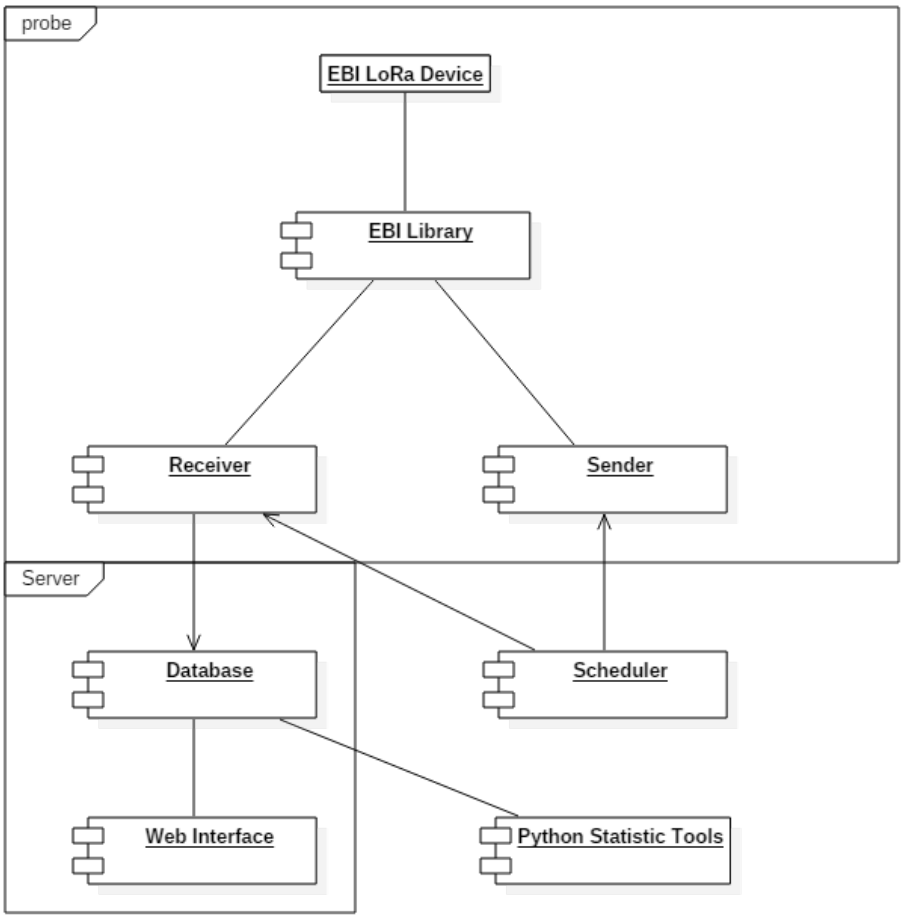


Abbildung 8.2: Softwarekomponenten des Testbeds

und später manuell in der Datenbank zu speichern. Um die Messungen manuell in der Datenbank zu speichern, gibt es ein eigenständiges Python-Skript, das ausgeführt werden kann (`syncData.py`). Jeder Test beginnt mit dem Start des Empfängerskripts. Sobald dies ausgeführt wird und die korrekten Einstellungen eingehalten werden, kann das Senderskript gestartet werden. Wenn das Senderskript gestartet wird, beginnt die Übertragung von einfachen LoRa-Paketen, die eine ganze Zahl enthalten, die pro Paket um 1 inkrementiert wird. Das Empfängerskript empfängt die übertragenen Pakete und validiert die gesendete ganze Zahl mit der zuvor empfangenen ganzen Zahl, um nach Paketverlust zu suchen. Das EBI LoRa Device gibt auch die Information des RSSI weiter, die mit jedem empfangenen Paket in die Datenbank gespeichert wird.

8.6 Datenbank

Die Datenbank wird auf einem virtualisierten CentOS 7-Server ausgeführt. Die verwendete Software für die Datenbank ist InfluxDB.[9] Diese Datenbanksoftware wurde speziell für Metriken, Ereignisse und Echtzeitanalysen entwickelt. Daher wurde die Datenbank für die Anforderungen solcher Messungen konfiguriert. Tabelle 8.1

Die Rohdaten werden in dieser Datenbank gespeichert. Jeder Test erhält eine UUID, um die Daten jedes Tests leicht zuzuordnen. Um weitere Informationen zu den Tests zu erhalten, werden die Metadaten einschließlich des Tests UUID in eine Google-Tabelle geschrieben. Zur Unterscheidung zwischen verschiedenen Standorten erhält jeder Test an einem Ort dieselbe Haupt-UUID und eine

Tabelle 8.1: Die implementierten Felder für jedes Ereignis / Eintrags

Schlüssel	Typ	Beschreibung
Paket_loss	<integer>	Paketverlust
rssi	<integer>	RSSI
rcv_lat	<float>	Empfänger GPS Breitengrad
rcv_long	<float>	Empfänger GPS Längengrad
run_time	<float>	Empfangszeit
snd_lat	<float>	Sender GPS Breitengrad
snd_long	<float>	Sender GPS Längengrad

eindeutige Sub-Test UUID. Dies erleichtert die Auswertung, da die Messungen leichter gruppiert werden können.

8.7 Web-Schnittstelle

Das Webinterface läuft ebenfalls auf dem Datenbankserver und bietet eine Webseite mit Grafana[8], einem webbasierten Grafiktool, das die Daten von InfluxDB direkt visualisieren kann. Dort können die Daten entweder live-graphisch oder nach der Messung visualisiert werden.

8.8 Scheduler

Der Scheduler ist auch in Python geschrieben und muss mit einer Konfigurationsdatei eingestellt werden, die gelesen wird, wenn der Scheduler startet und die vorkonfigurierten Tests ausführt.

Tabelle 8.2: Die benötigten Werte einer Konfigurationsdatei

Schlüssel	Typ	Beispiel
Lage	<string>	Salzburg
Dauer (in Sekunden)	<integer>	600
Entfernung	<string>	10m
Beschreibung	<string>	Erster Test
TX Power	<integer>	14
Spread Factor	<integer>	7
Antennenhöhe	<string>	5m

Der Scheduler kann auf jedem Rechner innerhalb vom VPN gestartet werden. Es liest die Konfigurationsdatei (Tabelle 8.2 und startet den ersten konfigurierten Test in der Liste. Dazu verbindet sich der Scheduler via SSH mit dem ersten Raspberry Pi und startet eine tmux-Session[10] und startet in dieser Sitzung das Empfängerskript. Nachdem das Empfängerskript ausgeführt wurde, verbindet sich der Scheduler via SSH mit dem zweiten Raspberry Pi und startet das Absendeskript in einer anderen Sitzung von tmux. Der Scheduler prüft, wenn ein neuer Test gestartet wird, ob noch ein Test läuft und wenn ja, wird der Test beendet, wenn die Zeit abgelaufen ist. Nachdem der andere Test beendet wurde, wird der neue Test gestartet. Wenn kein Test ausgeführt wird, wird der neue Test gestartet.

8.9 Python Statistik Werkzeuge

Um einige automatische Diagramme zur besseren Analyse der empfangenen Testdaten zu erstellen, haben wir einige Python-Skripte erstellt, die automatisch Boxplots, Histogramme und GPS-Karten und Höhendiagramme erstellen. Die Skripte können von der Kommandozeile mit nur dem Test UUID als Parameter ausgeführt werden und das Skript erstellt die angeforderten Diagramme in einem Unterordner.

9 Zusammenfassung

Schmalband-Technologien erlauben es kleine Datenpakete effizient von einem Sensor an einen zentralen Server zu übertragen. Die Effizient zeigt sich vor allem in Form einer langen Batterielaufzeit und bei hohen Reichweiten.

In der Praxis sind zur Zeit vor allem drei Technologien relevant: LoRa, Sigfox und NB-IoT. Diese unterscheiden sich sowohl in der den technischen Eigenschaften als auch in der Administration. Die Details der technischen Unterschiede sind für die meisten Nutzer der Technologie allerdings wenig relevant; die Unterschiede in der Administration allerdings schon: LoRa und Sigfox nutzen ein freies Lizenzspektrum und können daher weniger Aussagen über die Verfügbarkeit der Technologie an gegebenen Orten machen als NB-IoT. Bei NB-IoT und Sigfox ist der Nutzer auf einen Anbieter der Technologie angewiesen, der das Funknetz betreibt. Es ist möglich ein eigenes LoRa Netz zu betreiben oder auch den Service von einem Anbieter zu erwerben. Sigfox soll (wenn es voll ausgebaut ist), seinen Dienst anbieten können. Für LoRa ist ein solcher Betrieb nicht abzusehen. NB-IoT bietet zwar theoretisch weltweites Roaming an, die Details über dessen Nutzung sind allerdings noch offen.

Für verschiedene Einsatzzwecke eigenen sich also unterschiedliche Technologien. Um diese Auswahl einfacher zu machen, haben wir ein Online-Tool erstellt, das bei dabei unterstützen kann. Es ist frei unter der Adresse `srfg.at/nb-recommender` zu erreichen. So ergeben sich auch viele Einsatzzwecke der Technologien: Von Infrastrukturüberwachung über Lokalisierung von portablen Gegenständen bis zu Gebäudeautomatisierung.

Bei dem Einsatz ähneln sich die drei Technologien LoRa, Sigfox und NB-IoT sehr: Die Messdaten werden mit einem Sensor erhoben und über die ausgewählte Schmalband-Technologie versendet. Diese werden von einem Gateway empfangen und zur Speicherung an einen Server weitergeleitet. Von diesem Server aus können die Daten auf beliebigen Endgeräten darstellt werden. Am Beispiel einer Temperaturmessung haben wir verdeutlicht, wie diese mit unterschiedlichen Technologien umgesetzt werden kann. Außerdem haben wir am Beispiel ein GPS Lokalisierung gezeigt, wie der Sensor und die Visualisierung ausgetauscht werden können.

Um die Qualität einer Schmalband-Verbindung zu bewerten haben wir eine Messumgebung erstellt. Diese erlaubt es automatisiert Messungen durchzuführen, zentral zu speichern und auszuwerten. Eine solche Bewertung der Qualität ist hilfreich um abzuschätzen, ob die Nutzung einer Schmalband-Technologie an einem gegebenen Ort möglich ist, bzw. ob ein Ausbau der Infrastruktur nötig ist.

Dieses Handbuch gibt einen für jeden verständlichen Überblick und Möglichkeiten zur Nutzung von Schmalband-Technologien. Mit den in diesem Handbuch vermittelten Informationen und Referenzen ist es einer technisch versierten Nutzerin möglich eine Anwendung auf Basis von Schmalband-Technologien aufzubauen.

Literaturverzeichnis

[1] LoRa Alliance. Lorawan™ specification. *LoRa Alliance*, 2015.

[2] Antoine B. Bagula, Gordon Inggs, Simon Scott, and Marco Zennaro. On the Relevance of Using OpenWireless Sensor Networks in Environment Monitoring. *Sensors*, 9(6):4845–4868, June 2009.

[3] European Radiocommunications Committee et al. Erc recommendation 70-03 relating to the use of short range devices, 2002.

[4] Embit. EMB-AN868-BB24 Antenne Datasheet. Online version available at `http://www.embit.eu/wp-content/datasheets/EMB-AN868-BB24%20Datasheet.pdf` (24.02.2017).

[5] Embit. EMB-LR1272-EVK Evaluation Kit Datasheet. Online version available at `http://www.embit.eu/wp-content/datasheets/EMB-LR1272-EVK_Short_rev1.0.pdf` (24.02.2017).

[6] Embit. EMB-LR1272 Sendemodul Datasheet. Online version available at `http://www.embit.eu/wp-content/datasheets/EMB-LR1272-datasheet_rev1.0.pdf` (24.02.2017).

[7] Embit. Embit Binary Interface - LoRa-specific Documentation. Online version available at `http://www.embit.eu/wp-content/docs/ebi-LoRa_rev1.0.1.pdf` (08.10.2017).

[8] Grafana Labs. Grafana - The open platform for analytics and monitoring. Online version available at `https://grafana.com/` (08.10.2017).

[9] InfluxData. GitHub - nfluxdata/influxdb: Scalable datastore for metrics, events, and real-time analytics. Online version available at `https://github.com/influxdata/influxdb` (08.10.2017).

[10] Kooothor, Oldgaro. tmux - ArchWiki. Online version available at `https://wiki.archlinux.org/index.php/tmux` (08.10.2017).

[11] Python Software Foundation. Welcome to Python.org. Online version available at `https://www.python.org` (08.10.2017).