Lokalisierung mit GNSS, LTE und Narrowband IoT:   
Entwicklung von Langzeit- und Echtzeit-Tracking-Software in der Praxis

Erstprüfer/in:   
Zweitprüfer/in:

**Projektarbeit**

im Studiengang  
Informatik

vorgelegt von

**Chahida Raddam Nguyen Lam Bui Marlon May   
Yanik Oberheid**Matr.-Nr.: 18365744 Matr.-Nr.: 18365827 Matr.-Nr.: 18370234  
Matr.-Nr.: 18359570

am tt. Monat Jahr   
an der Hochschule Bochum

# Ehrenwörtliche Erklärung

# Abstract / Kurzfassung

# Inhaltsverzeichnis

Ehrenwörtliche Erklärung 2

Abstract / Kurzfassung 3

Inhaltsverzeichnis 4

Abbildungsverzeichnis 6

Tabellenverzeichnis 6

Abkürzungsverzeichnis 7

Vorwort 8

1 Überblick 9

2 Ziele 10

2.1 Szenario 1: Langzeit-Tracking für Fahrräder 10

2.2 Szenario 2: Echtzeit-Tracking für Fitness-Tracker 10

3 Lokalisierungstechnologien und Applikationen 11

3.1 GNSS 11

3.2 LTE und LTE-M 13

3.3.4. LTE-Programmierung für Mikrocontroller-basierte Tracker 15

3.3 Narrowband IoT 16

3.4 Vorteile und Herausforderungen von IoT-Signalen für die Lokalisierung 17

3.5 Lokalisierungsanwendungen 18

3.6 Lokalisierungsmethoden 20

3.6.1 Datenbank-Matching-Lokalisierungsmethoden 20

3.6.2 Geometrische Lokalisierungsmethoden 21

4 Mikrocontroller- Sensoren und Modems 24

4.1 Microcontroller Unit (MCU) 24

4.2 Quectel BG96 Modem 24

4.3 Bewegunssensor 24

4.4 Temperatursensor 24

5 Kommunikation mit dem MC 25

5.1 Serial Port 25

5.2 AT Command 25

5.3 BG96\_Serial Klasse 26

5.4 BG96\_Common Klasse 27

6 Networking 28

6.1 Einschalten des Modems 28

6.2 Transmission Control Protocol (TCP) 31

6.3 Secure Socket Layer (SSL) Connection 32

6.4 Message Queue Telemetry Transport (MQTT) 37

7 Lokalisierung 38

7.1 GPS 38

7.2 LTE 40

7.3 Narrowband IoT 40

7.4 Auswertung 40

8 Webapp Entwicklung 41

8.1 Anforderungen und Design 41

8.2 Architektur 41

8.3 Implementierung 41

8.4 Testing 41

8.5 Benutzeroberfläche und Interaktivität 41

8.6 Deployment und Hosting 41

9 Zusammenfassung und Ausblick 42

Glossar 43

Quellenverzeichnis 44

9.1 Quellen GPS: 44

9.2 Quellen LTE: 45

9.3 Quellen Narrowband: 45

Stichwortverzeichnis 47

# Abbildungsverzeichnis

[Abbildung 1: Abdeckungsbereiche und Energieverbrauch von IoT-Signalen (Quelle: Li et al., 2021) 18](#_Toc178295272)

[Abbildung 2: Lokalisierungssystem Architecture (Quelle: Li et al., 2021) 18](#_Toc178295273)

[Abbildung 3:Vergleich von tatsächlichen (rot) und vorhergesagten (blau) Standorten (Quelle: Ray et al., 2016) 21](#_Toc178295274)

[Abbildung 4: Prinzip der geometrischen Lokalisierungsmethoden (Quelle: Li et al. (2021)) 22](#_Toc178295275)

[Abbildung 5: Darstellung der Positionierung basierend auf TDOA-Messungen (Quelle: Yu et al. (2009)) 23](#_Toc178295276)

[Abbildung 6: BG96 mit GNSS und LTE Antenne 24](#_Toc178295277)

[Abbildung 7: DSerial und ATSerial definieren (Arduino Bootloader) 25](#_Toc178295278)

[Abbildung 8: Die sendAndSearch Methode 26](#_Toc178295279)

[Abbildung 9: Die InitModule Methode (Arduino Bootloader) 27](#_Toc178295280)

[Abbildung 10: Flussdiagramm zur Verwendung von TCP/IP-AT-Befehlen (Quelle: quectel\_bg96\_tcpip\_at\_commands\_manual\_v1-1) 29](#_Toc178295281)

[Abbildung 11: Die Methode InitAPN gemäß Flussdiagramm 31](#_Toc178295282)

[Abbildung 12: Debug-Konsole eines Programms, das einen Text an den Quectel Echo-Server sendet 32](#_Toc178295283)

# Tabellenverzeichnis

# Abkürzungsverzeichnis

HBI Hochschule für Bibliotheks- und Informationswesen

HdM

DDD

# Vorwort

# Überblick

# Ziele

Das Ziel dieses Softwareprojekts ist die Entwicklung einer umfassenden Lokalisierungslösung, die auf einem Mikrocontroller basiert und in der Lage ist, den Standort eines Objekts je nach Anwendungsszenario zu ermitteln. Es werden zwei Hauptszenarien betrachtet: Langzeit-Tracking für Fahrräder und Echtzeit-Tracking für Fitness-Tracker.

## Szenario 1: Langzeit-Tracking für Fahrräder

In diesem Szenario wird der Mikrocontroller genutzt, um die Position eines Fahrrads über längere Zeiträume hinweg zu überwachen. Dies erfordert eine energiesparende und zuverlässige Lösung, die regelmäßig Standortdaten sammelt und übermittelt, ohne die Batterie schnell zu entladen. Die Hauptanforderungen sind:

Lange Batterielaufzeit: Optimierung der Energiemanagement-Strategien, um die Batterielebensdauer zu maximieren.

Periodische Standortupdates: Standortinformationen werden in regelmäßigen Abständen aufgezeichnet und übermittelt, um den Verlauf der Fahrradbewegungen nachzuvollziehen.

Robuste Kommunikation: Verwendung von LTE-M und Narrowband IoT (NB-IoT) für eine zuverlässige Datenübertragung, auch in Gebieten mit schwacher Netzabdeckung.

## Szenario 2: Echtzeit-Tracking für Fitness-Tracker

Das Echtzeit-Tracking-Szenario zielt darauf ab, die Position eines Fitness-Trackers in nahezu Echtzeit zu ermitteln und zu übermitteln. Dies ist besonders nützlich für Sportler, die ihre Bewegungen und Aktivitäten genau verfolgen möchten. Die Hauptanforderungen in diesem Szenario sind:

Schnelle Standortaktualisierungen: Standortdaten müssen in sehr kurzen Intervallen aktualisiert werden, um eine Echtzeitverfolgung zu ermöglichen.

Hohe Präzision: Genauigkeit der Standortermittlung ist entscheidend, um präzise Bewegungs- und Aktivitätsdaten zu liefern.

Datenmanagement: Effiziente Handhabung der großen Datenmengen, die durch die häufigen Updates generiert werden, und deren sichere Übertragung.

# Lokalisierungstechnologien und Applikationen

## GNSS

Das Global Positioning System (GPS) ist eine satellitengestützte Navigations- und Ortungstechnologie, die ursprünglich vom US-Verteidigungsministerium in den 1970er Jahren entwickelt wurde. GPS wurde entworfen, um eine präzise Positionsbestimmung zu ermöglichen und wird heute weltweit in zivilen, kommerziellen und militärischen Anwendungen genutzt (Kaplan & Hegarty, 2005). Mit der zunehmenden Verbreitung mobiler Geräte und drahtloser Kommunikationstechnologien hat GPS eine Schlüsselrolle in vielen Anwendungen übernommen, von der Navigation über das Flottenmanagement bis hin zum Tracking von Objekten und Personen.

**3.1.1. Funktionsweise**

GPS basiert auf einer Konstellation von mindestens 24 Satelliten, die die Erde in einer Höhe von etwa 20.200 Kilometern umkreisen. Diese Satelliten senden kontinuierlich Funksignale aus, die von GPS-Empfängern auf der Erde empfangen werden. Die Position des Empfängers wird durch die Triangulation dieser Signale berechnet. Jedes Signal enthält Informationen über die Position des Satelliten und die Zeit, zu der das Signal gesendet wurde. Durch die Messung der Zeit, die das Signal benötigt, um den Empfänger zu erreichen, kann die Entfernung zu jedem Satelliten bestimmt werden. Mit Signalen von mindestens vier Satelliten kann der GPS-Empfänger seine Position in drei Dimensionen berechnen: Breite, Länge und Höhe (Misra & Enge, 2011).

GPS verwendet das Prinzip der Laufzeitmessung. Dies bedeutet, dass der Empfänger die Zeitdifferenz zwischen dem Senden und dem Empfangen eines Signals nutzt, um die Entfernung zum Satelliten zu bestimmen. Da die Signale mit Lichtgeschwindigkeit reisen, kann bereits eine winzige Zeitdifferenz zu erheblichen Ungenauigkeiten führen. Aus diesem Grund sind die GPS-Satelliten mit extrem präzisen Atomuhren ausgestattet, die es ermöglichen, die Signale auf wenige Nanosekunden genau zu messen (Hofmann-Wellenhof et al., 2008).

**3.1.2. Genauigkeit und Störungen**

Die Genauigkeit von GPS hängt von mehreren Faktoren ab, einschließlich der Anzahl der sichtbaren Satelliten, der atmosphärischen Bedingungen und der Standortumgebung. Unter idealen Bedingungen, z. B. im Freien ohne Hindernisse, kann die Genauigkeit von GPS bis zu wenigen Metern betragen. In urbanen Gebieten oder Innenräumen kann die Genauigkeit jedoch stark beeinträchtigt werden, da Gebäude oder andere Hindernisse die Signale blockieren oder reflektieren können. Dies wird als Multipath-Effekt bezeichnet, bei dem Signale von Gebäuden oder anderen Objekten abprallen und beim Empfänger eine Verzögerung verursachen (Kaplan & Hegarty, 2005).

Um die Genauigkeit zu verbessern, wurden verschiedene Korrekturmethoden entwickelt. Eine davon ist das Differenzielle GPS (DGPS), bei dem eine Bodenstation die GPS-Signale mit bekannten Koordinaten vergleicht und Korrekturen an die GPS-Empfänger sendet. Eine weitere Methode ist die Verwendung des Globalen Navigationssatellitensystems (GNSS), das andere Satellitensysteme wie das russische GLONASS oder das europäische Galileo in die Berechnungen einbezieht, um die Genauigkeit zu erhöhen und die Verfügbarkeit zu verbessern (Parkinson & Spilker, 1996).

**3.1.3. GPS in der Praxis: Anwendungen und Einschränkungen**

GPS hat zahlreiche Anwendungsgebiete, von der Fahrzeugnavigation und dem Flottenmanagement bis hin zur Überwachung und Sicherheit. In der Landwirtschaft wird GPS verwendet, um die präzise Positionierung von Maschinen zu steuern, während im Bauwesen GPS zur Vermessung und Standortplanung eingesetzt wird. Ein weiteres wichtiges Anwendungsgebiet ist die Ortung und das Tracking von Personen und Objekten in Echtzeit, was insbesondere in der Logistik und im Gesundheitswesen von großer Bedeutung ist (El-Rabbany, 2006).

Ein bedeutender Vorteil von GPS ist seine weltweite Abdeckung und die Fähigkeit, rund um die Uhr Positionsdaten bereitzustellen. Dies hat zur Entwicklung vieler ortsbezogener Dienste geführt, wie z. B. Navigationsanwendungen für Smartphones oder Tracking-Geräte für Tiere oder Fahrzeuge. Ein Nachteil von GPS ist jedoch der relativ hohe Energieverbrauch, insbesondere bei mobilen Geräten, was die Akkulaufzeit erheblich verkürzen kann. Zudem kann GPS unter extremen Bedingungen, wie z. B. in dichten städtischen Umgebungen, unter Wasser oder in unterirdischen Bereichen, nur eingeschränkt genutzt werden (Groves, 2008).

**3.1.4. Implementierung eines GPS-Trackingsystems auf einem Mikrocontroller**

Die Programmierung von GPS-Trackern auf Mikrocontrollern, wie z. B. einem ESP32 oder Arduino, erfolgt in der Regel durch die Integration eines GPS-Moduls (hier BG96) und die Nutzung einer seriellen UART-Schnittstelle. Diese GPS-Module übermitteln ihre Standortdaten in einem standardisierten Format, wie NMEA (National Marine Electronics Association), das die Koordinaten, Geschwindigkeit und Zeit enthält. Um die empfangenen Daten zu verarbeiten, wird oft eine Bibliothek wie TinyGPS++ verwendet, die es ermöglicht, die Rohdaten zu parsen und die relevanten Informationen herauszufiltern (Iannucci & Gariel, 2020).

Die gesammelten GPS-Daten werden zur Positionsverfolgung genutzt, um beispielsweise in regelmäßigen Abständen die Standortdaten an eine zentrale Datenbank zu senden. Eine der wichtigsten Anforderungen bei der GPS-Programmierung für Echtzeit-Tracking-Systeme ist die Energieeffizienz, da viele GPS-basierte Geräte batteriebetrieben sind. Eine typische Strategie besteht darin, das Gerät in regelmäßigen Abständen aufzuwecken, den Standort zu aktualisieren und dann wieder in den Energiesparmodus zu wechseln (Misra & Enge, 2010).

Zur Optimierung der Energieeffizienz setzen viele Mikrocontroller auf stromsparende Betriebsmodi. Tracking-Systeme benötigen eine kontinuierliche Standortüberwachung, aber oft reicht es aus, den Standort nur periodisch zu aktualisieren, um die Lebensdauer der Batterie zu verlängern. Um eine genauere Standortbestimmung zu gewährleisten, verwenden moderne Systeme DGPS (Differential GPS) oder SBAS (Satellite-Based Augmentation Systems) (Groves, 2013).

Ein weiterer wichtiger Aspekt in der GPS-Programmierung ist die Korrektur von Messungen, um die Genauigkeit zu verbessern. Techniken wie DGPS (Differential GPS) oder der Einsatz von zusätzlichen Satellitensystemen wie SBAS (Satellite-Based Augmentation Systems) verbessern die Positionsgenauigkeit, insbesondere in urbanen Gebieten oder bei schwierigen Empfangsbedingungen (Hofmann-Wellenhof et al., 2007).

Erfahrung folglich auch den deutschen Schulzuweisungsprozess beeinflusst und Ungleichverhätnisse

## LTE und LTE-M

LTE (Long-Term Evolution) ist eine Mobilfunktechnologie der vierten Generation (4G), die für mobile Breitbandanwendungen entwickelt wurde. Es bietet hohe Datenraten, geringe Latenzzeiten und eine verbesserte Spektraleffizienz im Vergleich zu früheren Mobilfunkstandards wie 3G. LTE-M (LTE for Machines), auch bekannt als LTE Cat-M1, ist eine spezielle Version von LTE, die für das Internet der Dinge (IoT) optimiert wurde. Es zielt darauf ab, energieeffiziente Verbindungen für IoT-Geräte zu ermöglichen, die keine hohen Datenraten benötigen, aber eine zuverlässige und weitreichende Abdeckung erfordern (Dahlman et al., 2018).

**3.2.1. LTE-Eigenschaften und Anwendungen**

LTE wurde entwickelt, um die Anforderungen moderner Anwendungen zu erfüllen, die hohe Datenraten und eine stabile Verbindung erfordern. Die Technologie bietet Download-Geschwindigkeiten von bis zu 300 Mbit/s und Upload-Geschwindigkeiten von bis zu 75 Mbit/s, abhängig von der Bandbreite und dem genutzten Spektrum (Mogensen et al., 2007). Einer der größten Vorteile von LTE ist die geringe Latenz, die für Echtzeitanwendungen wie Videokonferenzen und Online-Gaming entscheidend ist.

LTE basiert auf dem OFDMA-Verfahren (Orthogonal Frequency-Division Multiple Access) für den Downlink und SC-FDMA (Single Carrier-Frequency Division Multiple Access) für den Uplink. Diese Technologien ermöglichen eine effizientere Nutzung des verfügbaren Funkspektrums und tragen dazu bei, die Netzkapazität zu erhöhen. LTE verwendet auch das MIMO-Verfahren (Multiple Input Multiple Output), das mehrere Antennen zur gleichzeitigen Übertragung und Empfang von Signalen nutzt und so die Datenrate und Zuverlässigkeit verbessert (Holma & Toskala, 2011).

In der Praxis wird LTE in einer Vielzahl von Anwendungen eingesetzt, darunter mobile Kommunikation, Breitbandzugang in ländlichen Gebieten und Fahrzeugkommunikation. Die weite Verbreitung von LTE-fähigen Geräten und die globale Verfügbarkeit von LTE-Netzen machen es zu einer der wichtigsten drahtlosen Technologien für den mobilen Datenverkehr (Andrews et al., 2014).

Trotz seiner vielen Vorteile weist LTE auch einige Einschränkungen auf. Insbesondere ist die Technologie nicht für Anwendungen optimiert, die geringe Datenmengen über längere Zeiträume übertragen, wie es bei vielen IoT-Anwendungen der Fall ist. Außerdem erfordert LTE eine hohe Energieversorgung, was für batteriebetriebene IoT-Geräte suboptimal ist. Dies führte zur Entwicklung von LTE-M und anderen IoT-spezifischen Technologien wie Narrowband IoT (NB-IoT) (Ratasuk et al., 2014).

**3.2.2. LTE-M Eigenschaften und Anwendungen**

LTE-M ist eine Technologie, die speziell entwickelt wurde, um die Anforderungen des IoT zu erfüllen. Es handelt sich um eine Low-Power-Wide-Area (LPWA)-Technologie, die innerhalb des bestehenden LTE-Netzes betrieben wird und dabei dieselbe Infrastruktur nutzt. Im Vergleich zu standardmäßigen LTE-Verbindungen ist LTE-M wesentlich energieeffizienter und ermöglicht IoT-Geräten, über einen längeren Zeitraum mit einer einzigen Batterieladung zu funktionieren (Palattella et al., 2016).

Ein wesentlicher Vorteil von LTE-M ist seine Fähigkeit, bei sehr niedrigen Datenraten zu arbeiten, was für viele IoT-Anwendungen, wie z. B. intelligente Zähler, Gesundheitsüberwachung oder Asset-Tracking, ausreichend ist. LTE-M bietet eine Datenrate von bis zu 1 Mbit/s, was im Vergleich zu herkömmlichen LTE-Diensten gering erscheint, aber für IoT-Anwendungen völlig ausreichend ist. Darüber hinaus unterstützt LTE-M eine tiefere Reichweite als LTE, da es für den Betrieb in schwierigen Umgebungen optimiert wurde, z. B. in Gebäuden oder unterirdischen Bereichen (Ericsson, 2020).

Ein weiteres Merkmal von LTE-M ist die Reduktion der Latenzzeit, was es für Anwendungen, die eine schnelle Reaktion erfordern, wie z. B. Notfallalarmsysteme, attraktiv macht. LTE-M unterstützt außerdem die Funktion „Extended DRX“ (Discontinuous Reception), die es IoT-Geräten ermöglicht, in regelmäßigen Abständen in den Schlafmodus zu wechseln und so den Energieverbrauch weiter zu reduzieren (Ratasuk et al., 2015).

LTE-M eignet sich für eine breite Palette von IoT-Anwendungen, insbesondere in Bereichen, in denen eine zuverlässige, energieeffiziente und kostengünstige Verbindung erforderlich ist. In der Logistik wird LTE-M beispielsweise verwendet, um den Standort und den Status von Gütern während des Transports in Echtzeit zu überwachen. Es wird auch in intelligenten Städten eingesetzt, um vernetzte Infrastrukturen wie intelligente Straßenbeleuchtung oder Verkehrsüberwachungssysteme zu unterstützen (Palattella et al., 2016).

**3.2.3. Vergleich zwischen LTE und LTE-M**

Obwohl LTE und LTE-M auf derselben Grundtechnologie basieren, unterscheiden sie sich in wesentlichen Punkten. Während LTE für den mobilen Breitbandzugang optimiert ist und hohe Datenraten sowie eine niedrige Latenz bietet, liegt der Schwerpunkt von LTE-M auf Energieeffizienz und einer kostengünstigen Verbindung für IoT-Geräte. LTE-M-Geräte sind in der Regel einfacher und benötigen weniger Energie, was sie für batteriebetriebene Anwendungen attraktiver macht.

Ein weiterer Unterschied ist die Netzabdeckung. Während LTE für mobile Endgeräte optimiert ist, die sich häufig zwischen verschiedenen Zellen bewegen, bietet LTE-M eine tiefere Reichweite, was es ideal für IoT-Geräte macht, die in festen Standorten oder abgelegenen Gebieten arbeiten (Dahlman et al., 2018). LTE-M ermöglicht es Netzbetreibern außerdem, bestehende LTE-Infrastrukturen zu nutzen, um IoT-Dienste bereitzustellen, ohne zusätzliche Investitionen in neue Netztechnologien tätigen zu müssen.

### LTE-Programmierung für Mikrocontroller-basierte Tracker

Die Programmierung von Mikrocontrollern zur Nutzung von LTE-Technologie ist entscheidend für moderne Tracking-Anwendungen. Mikrocontroller wie der Arduino oder der ESP32 können in Kombination mit LTE-Modulen wie dem Quectel EC25 oder dem SIM800L eingesetzt werden, um eine zuverlässige Datenübertragung zu gewährleisten.

Für die Programmierung solcher Module werden häufig AT-Befehle verwendet. Ein typisches Beispiel ist der Befehl AT+CGATT=1, der zur Aktivierung der LTE-Verbindung dient. Diese Befehle ermöglichen eine effektive Kommunikation zwischen dem Mikrocontroller und dem LTE-Netzwerk (Quectel, 2020).

Ein wichtiger Aspekt bei der Programmierung ist das Energiemanagement. Viele Tracker sind batteriebetrieben, weshalb Funktionen wie der Power Saving Mode (PSM) und erweiterte Empfangszeiten (eDRX) implementiert werden, um den Stromverbrauch zu minimieren. Diese Funktionen erlauben es dem Modul, in einen Energiesparmodus zu wechseln, wenn es nicht aktiv Daten sendet oder empfängt

## Narrowband IoT

Narrowband IoT (NB-IoT) ist eine LPWAN-Technologie (Low Power Wide Area Network), die entwickelt wurde, um IoT-Anwendungen zu unterstützen, die eine große Reichweite, geringen Stromverbrauch und die Verbindung einer Vielzahl von Geräten erfordern. NB-IoT operiert in lizenzierten Frequenzbändern, was eine stabile und zuverlässige Kommunikation ermöglicht, oft über bestehende LTE-Infrastrukturen. Dies erlaubt Mobilfunknetzbetreibern (MNOs), NB-IoT parallel zu LTE zu implementieren (Raza, Kulkarni & Sooriyabandara, 2017).

NB-IoT wurde speziell für Geräte konzipiert, die nur kleine Datenmengen erzeugen und keine ständige Verbindung benötigen. Es erreicht eine tiefe Durchdringung von Gebäuden und deckt große Entfernungen mit hoher Energieeffizienz ab. Diese ist besonders wichtig für Anwendungsfälle wie Smart Metering, Asset-Tracking, Umweltüberwachung und die industrielle Automatisierung. Die Energieeinsparungen werden durch Funktionen wie Extended Discontinuous Reception (eDRX) und den Power-Saving Mode (PSM) ermöglicht, die es den Geräten erlauben, in den „Schlafmodus“ zu gehen, wenn keine Daten übertragen werden, was die Batterielaufzeit deutlich verlängert (Adelantado et al., 2017).

**3.3.1. Anwendungen und Vorteile**

NB-IoT (Narrowband IoT) zeichnet sich besonders durch seine Fähigkeit aus, Millionen von Geräten in einem begrenzten geografischen Gebiet zu unterstützen. Die Technologie bietet drei Bereitstellungsoptionen, die eine flexible Nutzung des bestehenden LTE-Spektrums ermöglichen: **Standalone**, **In-Band** und **Guard-Band**. Die **Standalone**-Option verwendet ein eigenes Frequenzband, was eine vollständige Trennung vom LTE-Datenverkehr gewährleistet und ideal ist für Regionen, in denen neue Frequenzen zugeteilt werden können. **In-Band**-Betrieb nutzt bestehende LTE-Ressourcenblöcke, was eine einfache Integration in LTE-Netzwerke erlaubt. **Guard-Band** nutzt die Ränder der LTE-Frequenzen, wodurch diese Option Spektrumseffizienz ohne Interferenz mit dem regulären LTE-Datenverkehr ermöglicht. Diese Flexibilität macht NB-IoT besonders geeignet für urbane IoT-Anwendungen wie Smart-City-Implementierungen, in denen sowohl Netzabdeckung als auch Kapazität entscheidend sind (Ratasuk, Mangalvedhe, Zhang, Robert, & Koskinen, 2016).

Obwohl NB-IoT im Vergleich zu anderen LPWAN-Lösungen wie LTE-M niedrigere Datenraten und eine höhere Latenz bietet, kompensiert es dies durch die niedrigen Implementierungskosten der Geräte und die erheblich verlängerte Batterielebensdauer. Diese Eigenschaften machen NB-IoT besonders geeignet für Anwendungsfälle, in denen die Datenübertragung nur gelegentlich erfolgt und die Geräte keine Echtzeitkommunikation erfordern (Raza et al., 2017).

NB-IoT ist eine Schlüsseltechnologie in der rasch wachsenden IoT-Landschaft. Es bietet ein optimales Gleichgewicht zwischen niedrigen Kosten, Energieeffizienz und einer breiten Netzabdeckung. Seine Fähigkeit, mit bestehender LTE-Infrastruktur integriert zu werden, sowie die Unterstützung für Millionen von IoT-Geräten machen NB-IoT zu einer entscheidenden Technologie für zukünftige IoT-Anwendungen.

**3.3.2. Programming Microcontrollers for NB-IoT Trackers**

Beim Integrieren von NB-IoT in ein Mikrocontroller-basiertes Trackingsystem wählen Entwickler typischerweise stromsparende Mikrocontroller (z. B. die STM32-Serie) und kompatible NB-IoT-Module wie das Quectel BC95 oder BC66. Diese Module werden mithilfe von AT-Befehlen programmiert, um Netzwerkregistrierung, Datenübertragung und Energiesparfunktionen zu steuern. Der Mikrocontroller übernimmt die GPS-Datenerfassung und überträgt periodisch die Standortdaten über NB-IoT an das Netzwerk. Effizientes Energiemanagement ist besonders bei batteriebetriebenen Trackern entscheidend und wird durch die Nutzung der stromsparenden Modi von NB-IoT erreicht (Quectel, 2019).

Ein typisches Beispiel für die Implementierung eines NB-IoT-basierten<><<<< Mikrocontrollers ist die Verwendung von Plattformen wie Arduino oder ESP32 in Kombination mit NB-IoT-Modulen wie Quectel BC95 oder u-blox SARA-N2. Diese Module bieten eine einfache Schnittstelle zur Anbindung des Mikrocontrollers an NB-IoT-Netzwerke. Die Programmierung erfolgt in Sprachen wie C oder Python und beinhaltet das Konfigurieren von Netzwerkeinstellungen, das Senden und Empfangen von Daten sowie die Implementierung von Energiesparmodi, um die Batterielebensdauer zu optimieren.

NB-IoT wird voraussichtlich eine entscheidende Rolle im wachsenden IoT-Bereich spielen, da es eine Lösung für energieeffiziente, großflächige Bereitstellungen bietet. Die Integration mit Mikrocontrollern setzt eine durchdachte Verwaltung des Energieverbrauchs sowie effiziente Kommunikationsprotokolle voraus, was NB-IoT zu einer Schlüsseltechnologie für die Zukunft intelligenter Trackingsysteme macht.

## Vorteile und Herausforderungen von IoT-Signalen für die Lokalisierung

Die neueste Kommunikationsinfrastruktur unterstützt zunehmend die Forschung zur Lokalisierung basierend auf IoT-Signalen aus mehreren Gründen. Erstens werden IoT-Signale von den gängigen IoT-Geräten unterstützt und sollen zukünftig auch von mehr intelligenten Verbrauchergeräten genutzt werden. Zweitens können IoT-Systeme bereits verschiedene Lokalisierungssignalmessungen wie Empfangssignalstärke (RSS), Zeitdifferenz der Ankunft (TDoA) und Kanalzustandsinformationen (CSI) liefern. Drittens erhöht die Verbreitung von IoT/5G-Kleinzellen sowie die Möglichkeit, die Kommunikationsfähigkeit von Smart-Home-Geräten wie Lampen, Routern, Lautsprechern und Steckdosen zu aktivieren, die Dichte der Lokalisierungsbasisstationen.

Diese Untersuchung konzentriert sich auf LPWAN-Signale, deckt aber auch andere IoT-Technologien wie Mobilfunknetze und lokale drahtlose Netzwerke (z. B. WiFi, Bluetooth Low Energy (BLE), Zigbee und Radio-Frequency Identification (RFID)) ab. Abbildung 1 zeigt die Abdeckungsbereiche und den Energieverbrauch der wichtigsten IoT-Signale. (Li et al., 2021)

Ein Bild, das Text, Screenshot, Diagramm, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 1: Abdeckungsbereiche und Energieverbrauch von IoT-Signalen (Quelle: Li et al., 2021)

## Lokalisierungsanwendungen

Ein Bild, das Text, Diagramm, Screenshot, Entwurf enthält.

Automatisch generierte Beschreibung

Abbildung 2: Lokalisierungssystem Architecture (Quelle: Li et al., 2021)

Ein Lokalisierungssystem besteht aus vier Komponenten: Knoten (einschließlich Endgeräten), Basisstationen (einschließlich Gateways oder Ankern), Netzwerkservern und Anwendungsservern (Raza, Kulkarni, & Sooriyabandara, 2017). Abbildung 2 zeigt die Architektur eines Lokalisierungssystem. Im Vergleich zu einem gewöhnlichen IoT-System verfügt das Lokalisierungssystem über eine zusätzliche Lokalisierungs-Engine. Abhängig von den Anforderungen an die Rechenlast der Knoten, die Kommunikationslast und die Datensicherheit kann das Lokalisierungsmodul entweder an den Knoten oder an den Netzwerkservern platziert werden. Die Hauptfunktionen der Komponenten sind wie folgt:

* Knoten: Ein Knoten enthält einen Transponder, der Signale überträgt, und optional einen Mikrocontroller mit On-Board-Speicher. Zusätzlich können Anwendungssensoren wie ein GNSS-Empfänger für präzise Positionierung, Inertialsensoren für Bewegungserfassung und Umweltsensoren zur Überwachung von Temperatur, Luftfeuchtigkeit, Rauch, Gas, Licht, Magnetismus und Schall integriert sein. Diese Sensoren können an den Transponderchip angeschlossen oder in diesen integriert sein. Die Knoten können zur statischen Überwachung fest installiert, als Tags auf beweglichen Objekten montiert oder als Benutzergeräte am menschlichen Körper getragen werden. In einigen IoT-Anwendungen senden die Knoten nur häufig Signale aus, anstatt Daten zu verarbeiten, um Energie zu sparen. In anderen Anwendungen erfolgt die Verarbeitung von Bewegungs- oder Lokalisierungsdaten auf den Knoten, um die Kommunikationslast zu reduzieren. Zudem können manche Anwendungen eine ToA-Lokalisierung nutzen, die präzise Timing-Anforderungen an die Knoten stellt, was nur in relativ hochwertigen IoT-Anwendungen erfüllt werden kann.
* Basisstationen (BSs): Die Hauptkommunikationsfunktion der Basisstationen besteht darin, Daten zwischen Knoten und Netzwerkservern zu routen. Die Basisstationen können sich mit Netzwerkservern verbinden und die Daten von Knotensensoren zu den Netzwerkservern und umgekehrt übertragen. Basisstationen haben in der Regel feste und bekannte Standorte sowie global eindeutige Identitäten (IDs, z. B. MAC-Adressen). Für LE-IoT müssen Basisstationen auch Lokalisierungssignale messen, wie Knoten-ID, Basisstation-ID, Empfangszeit der Daten, Kanal, RSS, Nutzlast und Signal-Rausch-Verhältnis (SNR). Zudem kann eine Zeitsynchronisation der Basisstationen für TDoA- oder ToA-basierte Lokalisierung erforderlich sein. Für die AoA-Lokalisierung können außerdem Multi-Array-Antennen und Phasendetektion notwendig sein.
* Netzwerkserver: Ein Netzwerkserver ist verantwortlich für das Dekodieren der Daten von den Basisstationen, das Aufzeichnen der Daten in Datenbanken, die optionale Implementierung von Lokalisierungsberechnungen und die Übermittlung der verarbeiteten Daten an Anwendungsserver. Netzwerkserver können sowohl für die Kommunikation von Sensor zu Anwendung als auch von Anwendung zu Sensor genutzt werden. Für die TDoA-basierte Lokalisierung ist es wichtig, dass die Pakete von verschiedenen Basisstationen bei einem Netzwerkserver ankommen. Außerdem gibt es für Lokalisierungsanwendungen zusätzliche Datenbanken für Lokalisierungssignale auf den Netzwerkservern. Bewegungs- und Lokalisierungsdatenverarbeitungs-Engines befinden sich bei vielen LPWAN-Anwendungen auf Netzwerkservern. Netzwerkserver können entweder Cloud- oder Edge-Server sein.
* Anwendungsserver: Ihre Hauptfunktionen bestehen darin, Daten von Netzwerkservern zu erhalten, zu analysieren und für weitere Anwendungen zu verarbeiten.

## Lokalisierungsmethoden

### Datenbank-Matching-Lokalisierungsmethoden

Datenbank-Matching-Lokalisierungsmethoden, auch als Fingerprinting-Methoden bekannt, sind Techniken zur Positionsbestimmung, bei denen aktuelle Sensordaten mit zuvor erfassten und in einer Datenbank gespeicherten Referenzdaten verglichen werden. Diese Methoden sind besonders nützlich in Umgebungen, in denen GPS-Signale schwach oder unzuverlässig sind, wie z.B. in Innenräumen oder dichten urbanen Gebieten. Die wichtigsten Algorithmen sind:

* Verstecktes Markov-Modell (HMM): Das HMM wird verwendet, um die Bewegung des Mobilgeräts und die beobachteten LTE-Messdaten zu modellieren. Die verborgenen Zustände des HMM sind die Position und die Geschwindigkeit des Mobilgeräts. Die Übergangswahrscheinlichkeiten modellieren die Bewegung des Geräts auf einem Straßennetzwerk.
* Partikelfilter (Particle Filter): Ein Partikelfilter-Algorithmus wird verwendet, um die Zustände des HMMs zu schätzen. Der Partikelfilter hält eine Menge von Partikeln, die jeweils eine Schätzung des Zustands des HMMs und ein Gewicht repräsentieren, das die Wahrscheinlichkeit der Beobachtungen widerspiegelt. Der Algorithmus aktualisiert diese Partikel basierend auf den Übergangswahrscheinlichkeiten und den Beobachtungen.
* Regression basierte Beobachtungslikelihood: Zufallswälder (Random Forests) werden verwendet, um die Wahrscheinlichkeit einer Beobachtung (z.B. gemessene RSRP-Werte) basierend auf einem Standort zu schätzen. Diese Wahrscheinlichkeiten werden dann verwendet, um die Beobachtungslikelihoods im Partikelfilter zu berechnen.
* Support Vector Machine (SVM): Eine SVM wird verwendet, um zu klassifizieren, ob eine LTE-Messung von einem Innen- oder Außenbereich stammt. Die Klassifizierung basiert auf den RSRP- und RSSI-Werten der Messung. Andere Algorithmen wie logistische Regression und Zufallswälder wurden ebenfalls getestet, aber die SVM zeigte die besten Ergebnisse für diese Aufgabe.

Ray et al. (2016) hat einen genannten Algorithmus entwickelt, um eine effektive Lösung für die Lokalisierung zu bieten.

Ein Bild, das Karte, Text, Atlas enthält.

Automatisch generierte Beschreibung

Abbildung 3:Vergleich von tatsächlichen (rot) und vorhergesagten (blau) Standorten (Quelle: Ray et al., 2016)

Ray et al. (2016) haben eine mittlere Genauigkeit von 20 Metern in städtischen Gebieten gezeigt, was eine deutliche Verbesserung gegenüber den mehr als 100 Metern Genauigkeit darstellt, die bei nicht auf maschinellem Lernen basierenden Techniken berichtet wurden. Eine größere Herausforderung besteht darin, Innenstandorte zumindest in Bezug auf Gebäude zu identifizieren.

### Geometrische Lokalisierungsmethoden

Geometrische Lokalisierungsmethoden basieren auf der Nutzung geometrischer Prinzipien zur Bestimmung des Standorts eines Objekts oder Geräts. Diese Methoden verwenden typischerweise die Positionen bekannter Referenzpunkte (wie Basisstationen) und messen Entfernungen oder Winkel zwischen diesen Referenzpunkten und dem zu lokalisierendes Objekt.

Ein Bild, das Text, Diagramm, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 4: Prinzip der geometrischen Lokalisierungsmethoden   
(Quelle: Li et al. (2021))

* Multilateration: Die Multilateration kann verwendet werden, um den Standort eines Knotens zu schätzen, indem die Positionen von mindestens drei Basisstationen (BSs) und ihre Entfernungen zum Knoten verwendet werden. Das Grundprinzip besteht darin, die Schnittpunkte zwischen Kugeln (für die 3D-Lokalisierung) und Kreisen (für die 2D-Lokalisierung) zu berechnen. Diese Methode wird häufig zur Standortbestimmung drahtloser Basisstationen eingesetzt. Gängige Schätztechniken sind das Verfahren der kleinsten Quadrate und der Kalman-Filter (KF). Die Leistung der Multilateration kann durch die Verbesserung der Reichweitenmessgenauigkeit und die Minderung von umgebungs- und empfängerbedingten Fehlern verbessert werden. (Li et al., 2021)
* Hyperbolische Lokalisierung: Die hyperbolische Lokalisierung, die ursprünglich für die Loran-Navigation entwickelt wurde, ist die Hauptmethode für TDoA-Lokalisierung (Time Difference of Arrival). Diese Methode basiert auf den Entfernungsunterschieden zwischen dem zu lokalisierenden Knoten und verschiedenen Basisstationen (BSs). Da bei der hyperbolischen Lokalisierung keine präzise Zeitsynchronisation auf den Knoten erforderlich ist, bietet sie ein großes Potenzial für die Lokalisierung in Low-Power Wide-Area Networks (LPWAN).

Ein Bild, das Entwurf, Diagramm, Design enthält.

Automatisch generierte Beschreibung

Abbildung 5: Darstellung der Positionierung basierend auf TDOA-Messungen  
(Quelle: Yu et al. (2009))

# Mikrocontroller- Sensoren und Modems

## Microcontroller Unit (MCU)

Für unser Projekt wird ein vom Auftraggeber bereitgestellter Mikrocontroller verwendet, dessen Architektur und Pinbelegung mit dem Arduino Zero kompatibel ist. Der Mikrocontroller basiert auf dem SAMD21G18 (ARM® Cortex®-M0+ Chip, 32 Bit). In die Schaltung des Mikrocontrollers wurden ein Temperatur- und Luftfeuchtigkeitssensor (HPP845E131R5), ein Beschleunigungssensor (BMA456) sowie Anschlüsse für ein separates Funkmodul (BG96) integriert. Zusätzlich wurde der Arduino-Bootloader auf den Mikrocontroller geflasht, wodurch dieser wie ein Arduino programmierbar ist.

Ein Bild, das Text, Screenshot, Design enthält.

Automatisch generierte Beschreibung

Abbildung 6: Arduino Zero Pinbelegung

Für die Programmierung des Projekts wird die Programmiersprache C/C++ verwendet, da sie eine hohe Effizienz und ressourcenschonende Implementierung ermöglicht. Zur Entwicklung steht ein Mini-USB-Anschluss zur Verfügung, der als serielle Schnittstelle dient. In der späteren Produktion wird dieser Mini-USB-Anschluss nicht verbaut. Stattdessen erfolgt die Programmierung direkt über ein Atmel-ICE-Debugger-Interface.

Die Stromversorgung des Mikrocontrollers erfolgt über zwei Lithium-Ionen-Batterien mit jeweils 3,6 V.

## Quectel BG96 Modem

Das Quectel BG96 LTE Cat M1/Cat NB1/EGPRS-Modul, das in unserem Projekt verwendet wird, unterstützt eine Vielzahl von Funktechnologien. Es bietet Konnektivität über LTE-M, Narrowband IoT (NB-IoT) und GSM und ermöglicht zudem die Positionsbestimmung mit allen gängigen globalen Satellitensystemen (GPS, GLONASS, Galileo und Beidou).

Das Modul ist physisch mit einer LTE-Antenne und einer GPS-Antenne ausgestattet. Die Kommunikation zwischen dem Mikrocontroller und dem Funkmodul erfolgt über eine serielle Schnittstelle.

Die Steuerung des BG96-Moduls erfolgt mithilfe von AT-Befehlen, die vom Hersteller Quectel bereitgestellt werden. Umfangreiche dokumentierte Anleitungen erleichtern die Implementierung und Konfiguration der gewünschten Funktionen.

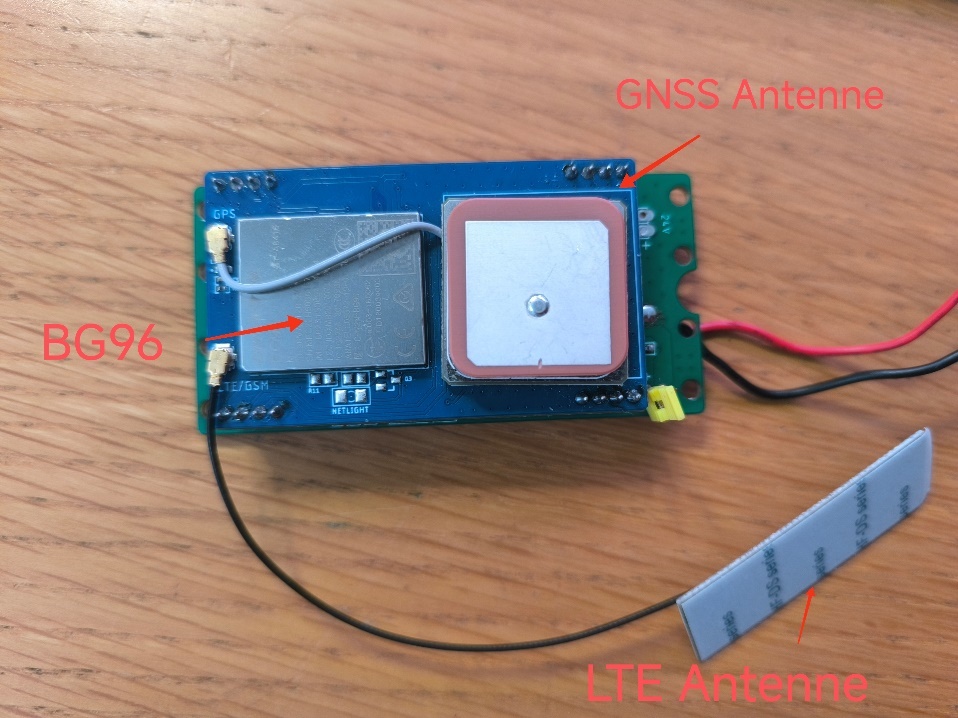


Abbildung 7: BG96 mit GNSS und LTE-Antenne

Für dieses Projekt werden die folgenden AT Command Dokumentationen des Herstellers Quectel verwendet:

• quectel\_bg96\_tcpip\_at\_commands\_manual\_v1-1

• quectel\_bg96\_ssl\_application\_note\_v1-1

• quectel\_bg96\_queccell\_at\_commands\_manual\_v1-0

• quectel\_bg96\_mqtt\_application\_note\_v1-2

• quectel\_bg96\_https\_at\_commands\_manual\_v1-0

• Quectel\_BG96\_GNSS\_AT\_Commands\_Manual\_V1.1

• quectel\_bg96\_file\_at\_commands\_manual\_v1-1

• Quectel\_BG96\_AT\_Commands\_Manual\_V2.3

Diese Dokumentationen bieten detaillierte Informationen und Anleitungen zur Nutzung der AT-Befehle für verschiedene Funktionen und Anwendungen des BG96-Moduls, ein-schließlich TCP/IP, SSL, MQTT, HTTPS, GNSS und Dateiverwaltung.

## Beschleunigungssensor (BMA456)

## Temperatursensor

# Kommunikation mit dem MC

## Serial Port

Der serielle Anschluss dient der Kommunikation zwischen dem Board und einem Computer oder anderen externen Geräten. Über diese Schnittstelle können Daten in Form von seriellen Signalen zuverlässig übertragen und empfangen werden, was eine effiziente und stabile Verbindung ermöglicht. Dieser Anschluss wird häufig für die Programmierung des Boards, das Debugging von Softwareanwendungen sowie die Integration von Peripheriegeräten wie Sensoren, Aktoren, Displays oder externen Speichermedien verwendet. Durch die Nutzung des seriellen Anschlusses können Entwickler einfach und kostengünstig Daten austauschen, Steuerbefehle senden und Informationen empfangen, was die Entwicklung und das Testen von elektronischen Projekten erheblich erleichtert. Zudem unterstützt der serielle Anschluss verschiedene Kommunikationsprotokolle, wodurch eine flexible Anpassung an unterschiedliche Anforderungen und Geräte ermöglicht wird. Insgesamt stellt der serielle Anschluss eine essenzielle Schnittstelle dar, die die Interaktion und Zusammenarbeit zwischen dem Board und einer Vielzahl von Computern sowie anderen elektronischen Geräten effektiv gewährleistet.

#define DSerial SerialUSB

#define ATSerial Serial1

Abbildung 8: DSerial und ATSerial definieren (Arduino Bootloader)

## AT Command

AT-Befehle, offiziell als Attention Commands bekannt, sind Anweisungen, die zur Konfiguration, Steuerung und Kommunikation mit einem Modem verwendet werden. Bei AT-Befehlen beginnt jede Befehlszeile mit „AT“ oder „at“. Das anfängliche „AT“ dient als Präfix und gehört nicht zum eigentlichen Befehlsnamen. Zum Beispiel ist „E“ der tatsächliche AT-Befehl in „ATE<value>“, der das Modem in den Echo-Modus versetzt (es werden alle vom Modem empfangenen AT-Befehle zurückgesendet). Der Platzhalter „<value>“ kann entweder mit 0 oder 1 belegt werden, um das Echo zu deaktivieren bzw. zu aktivieren.

Für dieses Projekt werden die folgenden AT Command Dokumentationen des Herstellers Quectel verwendet:

* quectel\_bg96\_tcpip\_at\_commands\_manual\_v1-1
* quectel\_bg96\_ssl\_application\_note\_v1-1
* quectel\_bg96\_queccell\_at\_commands\_manual\_v1-0
* quectel\_bg96\_mqtt\_application\_note\_v1-2
* quectel\_bg96\_https\_at\_commands\_manual\_v1-0
* Quectel\_BG96\_GNSS\_AT\_Commands\_Manual\_V1.1
* quectel\_bg96\_file\_at\_commands\_manual\_v1-1
* Quectel\_BG96\_AT\_Commands\_Manual\_V2.3

Diese Dokumentationen bieten detaillierte Informationen und Anleitungen zur Nutzung der AT-Befehle für verschiedene Funktionen und Anwendungen des BG96-Moduls, einschließlich TCP/IP, SSL, MQTT, HTTPS, GNSS und Dateiverwaltung.

## BG96\_Serial Klasse

Die Klasse \_BG96\_Serial dient zur Kommunikation mit dem BG96-Modul über serielle Schnittstellen. Sie bietet Methoden zur Konfiguration, Steuerung und Datenübertragung mittels AT-Befehlen.

Cmd\_Response\_t \_BG96\_Serial::sendAndSearch(const char \*command, const char \*test\_str, const char \*e\_test\_str, unsigned int timeout)

{

Cmd\_Response\_t resp\_status = UNKNOWN\_RESPONSE;

for (int i = 0; i < 3; i++){

if(sendATcommand(command)){

resp\_status = readResponseAndSearch(test\_str, e\_test\_str, timeout);

return resp\_status;

}

}

return resp\_status;

}

Abbildung 9: Quellcode der sendAndSearch Methode

Die sendAndSearch() Methoden dienen dazu, einen AT-Befehl an das BG96-Modem zu senden und anschließend die Antwort des Modems auf bestimmte erwartete Zeichenketten oder Zeichen zu überprüfen. Diese Methoden wiederholen den Vorgang bis zu dreimal, falls die gewünschte Antwort nicht innerhalb des festgelegten Timeouts empfangen wird. Die Methode sendATcommand() dient dazu, einen AT-Befehl an das BG96-Modem zu senden. Falls die Debugging-Funktion (UART\_DEBUG) aktiviert ist, werden die gesendeten Daten zur Fehlerbehebung oder Überwachung auf der Debug-Schnittstelle (\_dserial) ausgegeben. Dies umfasst den gesendeten AT-Präfix und den Befehl selbst. Die Methode readResponseAndSearch() der Klasse \_BG96\_Serial dient dazu, die Antwort des BG96-Modems zu lesen und nach bestimmten Zeichenketten zu suchen. rxBuffer ist eine zentrale Variable innerhalb der Klasse \_BG96\_Serial, die dazu dient, eingehende Daten vom BG96-Modem über die serielle Schnittstelle zu speichern. Diese Daten stammen in der Regel als Antwort auf zuvor gesendete AT-Befehle.

## BG96\_Common Klasse

Die Klasse \_BG96\_Common erweitert die Funktionalitäten der Basisklasse \_BG96\_Serial, um erweiterte Funktionen des BG96-Moduls bereitzustellen. Sie bietet Methoden zur Steuerung und Konfiguration des Moduls, wie das Ein- und Ausschalten, das Setzen von Konfigurationsparametern, das Abrufen von Geräteinformationen und Netzstatus sowie das Verwalten von SIM-Karteninformationen.

bool \_BG96\_Common::InitModule()

{

pinMode(ENABLE\_PWR, OUTPUT);

digitalWrite(ENABLE\_PWR, HIGH); // BG96 Modul mit Strom versorgen

delay(800);

pinMode(RESET\_PIN, OUTPUT);

digitalWrite(RESET\_PIN, LOW);

pinMode(POWKEY\_PIN, OUTPUT);

digitalWrite(POWKEY\_PIN, LOW); // Powkey-Pin auf LOW setzen

delay(800);

digitalWrite(POWKEY\_PIN, HIGH); // Powkey-Pin auf HIGH setzen

delay(800);

ResetModule();

return true;

}

Abbildung 10: Quellcode der InitModule Methode (Arduino Bootloader)

Die Methode initialisiert das BG96-Modul, indem sie die notwendigen Pins konfiguriert, das Modul einschaltet und einen Reset durchführt. Sie setzt die Pins ENABLE\_PWR, RESET\_PIN und POWKEY\_PIN entsprechend der Modulanforderungen.

# Networking

In diesem Kapitel wird erläutert, wie ein Arduino-Sketch entwickelt wird, der das Modem aktiviert, es mit dem Mobilfunknetz verbindet und eine Kommunikation mit dem Internet von einem Computer aus ermöglicht.

## Einschalten des Modems

Das Flussdiagramm *Abb. 10* beschreibt den Prozess zur Initialisierung und Verwaltung einer Mobilfunkverbindung mit AT-Befehlen in einem Kommunikationsmodul. Der Ablauf beginnt mit dem Einschalten des Modems, das über den Befehl AT+QPOWD gesteuert wird. Nach einer kurzen Wartezeit wird überprüft, ob das Modul korrekt auf AT-Befehle reagiert. Eine erfolgreiche Rückmeldung in Form eines "OK" zeigt die Einsatzbereitschaft des Moduls an.

Im nächsten Schritt wird der Status der SIM-Karte mittels AT+CPIN? abgefragt. Sollte die SIM-Karte nicht innerhalb von 20 Sekunden erkannt werden, wird das Modul neu gestartet, um eine erneute Abfrage zu ermöglichen. Sobald die SIM-Karte erfolgreich identifiziert wurde, wird die Registrierung im Circuit-Switched (CS) Dienst über AT+CREG? durchgeführt. Erfolgt keine Registrierung innerhalb von 90 Sekunden, wird das Modul erneut gestartet, um den Registrierungsprozess zu wiederholen.

Parallel dazu wird die Registrierung im Packet-Switched (PS) Dienst über AT+CGREG? oder AT+CEREG? geprüft. Auch hier wird erwartet, dass die Registrierung innerhalb von 60 Sekunden erfolgt. Sobald die Netzregistrierung erfolgreich abgeschlossen ist, kann der PDP-Kontext (Packet Data Protocol) konfiguriert werden. Dies erfolgt über AT+QICSGP, wo der Zugangspunktname (APN), Benutzername, Passwort sowie die Authentifizierungsmethode gesetzt werden. Die QoS-Parameter (Quality of Service) werden mit AT+CGQREQ konfiguriert.

Im nächsten Schritt wird der PDP-Kontext mit AT+QIACT aktiviert. Erfolgt keine Aktivierung innerhalb von 40 Sekunden, wird das Modul neu gestartet und der Prozess wiederholt. Nach erfolgreicher Aktivierung wird die Verbindung mittels AT+QIOPEN hergestellt. Sollte die Verbindung nicht innerhalb von 150 Sekunden geöffnet werden, wird der PDP-Kontext neu initialisiert und der Verbindungsaufbau erneut versucht.

Sobald die Verbindung steht, kann der Datenaustausch über AT+QISEND durchgeführt werden. Um die Verbindung aktiv zu halten, wird empfohlen, regelmäßig kleine Datenpakete zu senden. Die erfolgreiche Übertragung wird durch eine Empfangsbestätigung (ACK) signalisiert. Falls keine Bestätigung innerhalb von 90 Sekunden erfolgt, wird die Verbindung geschlossen und neu aufgebaut.

Ein Bild, das Text, Diagramm, parallel, Dokument enthält.

Automatisch generierte Beschreibung

Abbildung 11: Flussdiagramm zur Verwendung von TCP/IP-AT-Befehlen   
(Quelle: quectel\_bg96\_tcpip\_at\_commands\_manual\_v1-1)

bool \_BG96\_TCPIP::InitAPN(unsigned int pdp\_index, const char\* apn, const char\* usr, const char\* pwd, char\* err\_code)

{

// Statusvariablen

Net\_Status\_t i\_status = NOT\_REGISTERED;

Cmd\_Response\_t init\_status;

const char \*e\_str;

char i\_ip[16]; // Buffer für die IP-Adresse

// Schritt 1: SIM-Kartenprüfung

unsigned long start\_time = millis();

while (!DevSimPIN("", READ\_MODE)) {

if (millis() - start\_time >= 20 \* 1000UL) { // Timeout nach 20 Sekunden

e\_str = "\r\nAPN ERROR: No SIM card detected!\r\n";

strcpy(err\_code, e\_str);

ResetModule();

return false;

}

}

// Schritt 2: Netzregistrierung prüfen

start\_time = millis();

while (i\_status != REGISTERED && i\_status != REGISTERED\_ROAMING) {

i\_status = DevNetRegistrationStatus();

if (millis() - start\_time >= 90 \* 1000UL) { // Timeout nach 90 Sekunden

e\_str = "\r\nAPN ERROR: Can't register to the operator network!\r\n";

strcpy(err\_code, e\_str);

ResetModule();

return false;

}

delay(3000); // Warte 3 Sekunden vor dem nächsten Registrierungsversuch

}

// Schritt 3: APN-Konfiguration setzen

if (!SetDevAPNParameters(pdp\_index, IPV4, apn, usr, pwd, PAP\_OR\_CHAP)) {

e\_str = "\r\nAPN ERROR: Failed to set APN parameters!\r\n";

strcpy(err\_code, e\_str);

return false;

}

// Schritt 4: APN aktivieren und IP-Adresse abrufen

start\_time = millis();

while (millis() - start\_time <= 150 \* 1000UL) { // Timeout nach 150 Sekunden

init\_status = ActivateDevAPN(pdp\_index);

if (init\_status == SUCCESS\_RESPONSE) {

if (GetDevAPNIPAddress(pdp\_index, i\_ip)) {

sprintf(err\_code, "\r\nAPN OK: The IP address is %s\r\n", i\_ip);

return true;

} else {

e\_str = "\r\nAPN ERROR: Failed to retrieve IP address!\r\n";

strcpy(err\_code, e\_str);

return false;

}

} else if (init\_status == TIMEOUT\_RESPONSE) {

e\_str = "\r\nAPN ERROR: APN activation timeout. Please reset your device!\r\n";

strcpy(err\_code, e\_str);

ResetModule();

return false;

}

}

// Falls die APN-Aktivierung fehlschlägt

e\_str = "\r\nAPN ERROR: Failed to activate APN!\r\n";

strcpy(err\_code, e\_str);

return false;

}

Abbildung 12: Quellcode der InitAPN Methode gemäß Flussdiagramm

Die Methode `InitAPN` der Klasse „BG96\_TCPIP“ initialisiert den APN des Geräts in mehreren Schritten. Zunächst wird die SIM-Karte überprüft. Erfolgt innerhalb von 20 Sekunden keine Erkennung, wird eine Fehlermeldung ausgegeben und die Methode beendet. Danach wird die Netzregistrierung überprüft, wobei bei einem Timeout von 90 Sekunden ebenfalls eine Fehlermeldung ausgegeben wird. Anschließend werden die APN-Parameter gesetzt. Im letzten Schritt wird der APN aktiviert und die IP-Adresse abgerufen. Bei erfolgreicher Ausführung wird `true` zurückgegeben, andernfalls `false` und ein Fehlercode im `err\_code`-Puffer gespeichert.

## Transmission Control Protocol (TCP)

TCP ist ein verbindungsorientiertes Protokoll, das eine Verbindung zwischen einem Client und einem Server aufbaut und aufrechterhält. Es bestimmt die Übertragung von Anwendungsdaten in Form von Paketen, die das Netzwerk zum Empfänger durchlaufen. TCP kommuniziert mit der Netzwerkschicht, indem es Pakete sendet und empfängt, verwaltet die Flusskontrolle und sorgt für die erneute Übertragung verlorener oder beschädigter Pakete sowie für die Bestätigung aller empfangenen Pakete. Im Open Systems Interconnection (OSI)-Modell ist TCP dem vierten Layer, der Transportschicht, zugeordnet.



Abbildung 13: Debug-Konsole eines Programms, das einen Text an den Quectel Echo-Server sendet

Die Verbindung mit Quectel Echo-Server wird über den Befehl `AT+QIOPEN` initiiert. Erfolgt innerhalb von 150 Sekunden keine erfolgreiche Verbindungsherstellung, wird der PDP-Kontext neu initialisiert und der Verbindungsaufbau wiederholt. Nach erfolgreichem Verbindungsaufbau erfolgt die Datenübertragung mittels `AT+QISEND`. Zur Sicherstellung der Verbindungsstabilität wird empfohlen, regelmäßig Datenpakete zu senden. Bleibt eine Bestätigung (ACK) nach mehreren Versuchen aus, wird die Verbindung geschlossen und neu aufgebaut. Die TCP-Verbindung wird kontinuierlich überwacht. Sollte innerhalb von zwei Minuten keine ACK-Bestätigung empfangen werden, wird die Verbindung als fehlerhaft eingestuft und erneut initiiert.

## Secure Socket Layer (SSL) Connection

Secure Sockets Layer (SSL) ist ein weit verbreitetes kryptografisches Protokoll, das die Sicherheit der Kommunikation über das Internet gewährleistet. SSL ermöglicht die Einrichtung eines sicheren Kanals zwischen zwei Geräten oder zwischen einem Client und einem Server, die entweder über das Internet oder ein internes Netzwerk verbunden sind. Ein typisches Anwendungsbeispiel für SSL ist die Absicherung der Kommunikation zwischen einem NB-IoT-Gerät und einem Server.

**Klasse \_BG96\_SSL**

Die Klasse \_BG96\_SSL stellt eine Abstraktion der SSL-Kommunikationsfunktionen des Quectel BG96 Moduls dar. Sie ermöglicht die Konfiguration, Initialisierung und Verwaltung von SSL-Verbindungen über AT-Befehle. Diese Klasse unterstützt die Konfiguration von SSL-Parametern wie Version und Cipher-Suites, sowie die Handhabung von SSL-Sockets, einschließlich des Sendens und Empfangens von Daten.

bool \_BG96\_SSL::InitSSL(unsigned int ssl\_index, char \*ca\_cert, char \*client\_cert, char \*client\_key, char \*err\_code)

{

unsigned long start\_time;

int f\_err\_code;

// Case 1: No certificates provided (non-secure)

if (strcmp(ca\_cert, "") == 0 && strcmp(client\_cert, "") == 0 && strcmp(client\_key, "") == 0)

{

//Code Case 1

return false;

}

// Case 2: Only CA certificate provided (one-way SSL)

else if (strcmp(ca\_cert, "") != 0 && strcmp(client\_cert, "") == 0 && strcmp(client\_key, "") == 0)

{

//Code Case 2

return false;

}

// Case 3: CA certificate, client certificate, and client key provided (two-way SSL)

else if (strcmp(ca\_cert, "") != 0 && strcmp(client\_cert, "") != 0 && strcmp(client\_key, "") != 0)

{

// Upload CA certificate

start\_time = millis();

while (!UploadFiles((char \*)ssl\_ca\_cert\_name, ca\_cert))

{

if (returnErrorCode(f\_err\_code))

{

if (f\_err\_code == 407) // File already exists

{

unsigned long delete\_start\_time = millis();

while (!DeleteFiles((char \*)ssl\_ca\_cert\_name))

{

if (millis() - delete\_start\_time >= 10 \* 1000UL)

{

strcpy(err\_code, "\r\nSSL ERROR: Unable to delete existing CA certificate file during re-upload.\r\n");

return false;

}

}

}

else

{

sprintf(err\_code, "\r\nSSL ERROR: Error uploading CA certificate file, error code: %d.\r\n", f\_err\_code);

return false;

}

}

if (millis() - start\_time >= 30 \* 1000UL)

{

strcpy(err\_code, "\r\nSSL ERROR: Timeout occurred while uploading CA certificate file.\r\n");

return false;

}

}

// Upload Client Certificate

start\_time = millis();

while (!UploadFiles((char \*)ssl\_client\_cert\_name, client\_cert))

{

if (returnErrorCode(f\_err\_code))

{

if (f\_err\_code == 407) // File already exists

{

unsigned long delete\_start\_time = millis();

while (!DeleteFiles((char \*)ssl\_client\_cert\_name))

{

if (millis() - delete\_start\_time >= 10 \* 1000UL)

{

strcpy(err\_code, "\r\nSSL ERROR: Unable to delete existing client certificate file during re-upload.\r\n");

return false;

}

}

}

else

{

sprintf(err\_code, "\r\nSSL ERROR: Error uploading client certificate file, error code: %d.\r\n", f\_err\_code);

return false;

}

}

if (millis() - start\_time >= 30 \* 1000UL)

{

strcpy(err\_code, "\r\nSSL ERROR: Timeout occurred while uploading client certificate file.\r\n");

return false;

}

}

// Upload Client Key

start\_time = millis();

while (!UploadFiles((char \*)ssl\_client\_key\_name, client\_key))

{

if (returnErrorCode(f\_err\_code))

{

if (f\_err\_code == 407) // File already exists

{

unsigned long delete\_start\_time = millis();

while (!DeleteFiles((char \*)ssl\_client\_key\_name))

{

if (millis() - delete\_start\_time >= 10 \* 1000UL)

{

strcpy(err\_code, "\r\nSSL ERROR: Unable to delete existing client key file during re-upload.\r\n");

return false;

}

}

}

else

{

sprintf(err\_code, "\r\nSSL ERROR: Error uploading client key file, error code: %d.\r\n", f\_err\_code);

return false;

}

}

if (millis() - start\_time >= 30 \* 1000UL)

{

strcpy(err\_code, "\r\nSSL ERROR: Timeout occurred while uploading client key file.\r\n");

return false;

}

}

// Set SSL parameters

// AWS IoT Cores Security policy TLS13\_1\_2\_2022\_10 only supports ECDHE-RSA-AES256-SHA384 for BG96

if (!SetSSLParameters(ssl\_index, TLS\_1\_2, TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA384, 300))

{

strcpy(err\_code, "\r\nSSL ERROR: An error occurred while setting the SSL parameters.\r\n");

return false;

}

// Set SSL Certificates

start\_time = millis();

while (!SetSSLCertificate(ssl\_index, (char \*)ssl\_ca\_cert\_name, (char \*)ssl\_client\_cert\_name, (char \*)ssl\_client\_key\_name, false))

{

if (millis() - start\_time >= 30 \* 1000UL)

{

strcpy(err\_code, "\r\nSSL ERROR: An error occurred while setting the SSL certificates.\r\n");

return false;

}

}

strcpy(err\_code, "\r\nSSL OK: The SSL was successfully initialized with CA certificate, client certificate, and client key.\r\n");

return true;

}

else

{

strcpy(err\_code, "\r\nSSL ERROR: Invalid certificate parameters provided.\r\n");

return false;

}

}

Abbildung 14: Quellcode InitSSL Methode

Die Methode InitSSL() dient der Initialisierung einer sicheren SSL-Verbindung auf dem BG96-Modul, wobei sie Zertifikate hochlädt und SSL-Parameter für die verschlüsselte Kommunikation konfiguriert. Diese Methode spielt eine zentrale Rolle bei der Herstellung sicherer Verbindungen zu Cloud-Diensten wie AWS IoT Core, da sie die Grundlage für die Verschlüsselung der Kommunikation zwischen dem Client (BG96-Modul) und dem Server (AWS IoT) legt.

Die Methode unterscheidet zwischen drei Hauptszenarien, von denen im vorliegenden Anwendungsfall die Zwei-Wege-SSL-Authentifizierung relevant ist. In der ersten Variante, bei der keine Zertifikate bereitgestellt werden, würde eine unsichere Verbindung ohne Verschlüsselung hergestellt werden, was für Anwendungen ohne Sicherheitsanforderungen geeignet wäre, jedoch in der Praxis selten verwendet wird. In der zweiten Variante, bei der nur ein CA-Zertifikat bereitgestellt wird, erfolgt eine einseitige Authentifizierung, bei der lediglich die Identität des Servers überprüft wird. Da für den Anwendungsfall eine gegenseitige Authentifizierung zwischen Client und Server notwendig ist, liegt der Fokus auf der dritten Variante.

Im Fall der Zwei-Wege-SSL-Authentifizierung werden sowohl das CA-Zertifikat als auch das Client-Zertifikat und der Client-Schlüssel verwendet. Diese Zertifikate ermöglichen es dem BG96-Modul, den Server zu authentifizieren und sich selbst gegenüber dem Server zu authentifizieren. Der Prozess beginnt mit dem Hochladen der Zertifikate auf den internen Speicher des Moduls. Hierbei wird überprüft, ob die Datei bereits existiert, und wenn ja, wird sie gelöscht und erneut hochgeladen. Dies stellt sicher, dass immer die aktuellen Zertifikate verwendet werden und alte, möglicherweise ungültige Zertifikate entfernt werden.

Nachdem die Zertifikate erfolgreich hochgeladen wurden, erfolgt die Konfiguration der SSL-Parameter. Für die sichere Kommunikation wird das **TLS 1.2**-Protokoll verwendet, das in Kombination mit dem Verschlüsselungsalgorithmus **ECDHE-RSA-AES256-CBC-SHA384** den Anforderungen von AWS IoT entspricht. Dieser Algorithmus gewährleistet eine starke Verschlüsselung und schützt die Kommunikation vor unbefugtem Zugriff oder Manipulation. Gleichzeitig wird ein Timeout von 300 Sekunden für die Verbindung konfiguriert, um sicherzustellen, dass die Verbindung stabil bleibt und nicht unnötig lange geöffnet bleibt, falls keine Daten übertragen werden.

Im letzten Schritt werden die zuvor hochgeladenen Zertifikate dem SSL-Index des Moduls zugewiesen, wodurch die SSL-Konfiguration abgeschlossen wird. Nach erfolgreicher Initialisierung der SSL-Verbindung kann das BG96-Modul sicher mit dem AWS IoT Core kommunizieren und Daten verschlüsselt senden und empfangen.

## Message Queue Telemetry Transport (MQTT)

# Lokalisierung

## GPS

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Dieser Code steuert ein GNSS-Modul (Global Navigation Satellite System) über ein Arduino-Board. Das Modul verwendet die BG96-Bibliothek, um GNSS-Daten zu initialisieren und zu lesen. Die Kommunikation erfolgt über serielle Schnittstellen.

#include "Arduino.h"

#include "BG96\_GNSS.h"

#define DSerial SerialUSB

#define ATSerial Serial1

#define UART\_DEBUG

GNSS\_Work\_Mode\_t mode = STAND\_ALONE;

\_BG96\_GNSS \_5GNBIoT(ATSerial, DSerial);

void setup()

{

  DSerial.begin(115200);

  while (DSerial.read() >= 0)

    ;

  DSerial.println("This is the \_5GNBIoT Debug Serial!");

  ATSerial.begin(115200);

  while (ATSerial.read() >= 0)

    ;

  delay(1000);

  if (\_5GNBIoT.InitModule())

  {

    DSerial.println("\r\n\_5GNBIoT.InitModule() OK!");

  }

  delay(1000);

  if(\_5GNBIoT.SetGNSSOutputPort(UARTNMEA)){

    DSerial.println("\r\nSet GNSSOutputPort OK!");

  }

  \_5GNBIoT.SetDevCommandEcho(false);

  delay(100);

  if (\_5GNBIoT.TurnOnGNSS(mode, WRITE\_MODE))

  {

    DSerial.println("\r\nOpen the GNSS Function Success!");

  }

  else

  {

    DSerial.println("\r\nFail to open GNSS Function!");

  }

}

void loop()

{

  char gnss\_posi[128];

  if (\_5GNBIoT.GetGNSSPositionInformation(gnss\_posi))

  {

    DSerial.println("\r\nGet the GNSS Position Success!");

    DSerial.println(gnss\_posi);

  }

  else

  {

    DSerial.println("\r\nFail to get the GNSS Position!");

  }

  delay(3000);

}

Fehler 516: Ein Bild, das Text, Schrift, Screenshot enthält.

Automatisch generierte Beschreibung

## LTE

## Narrowband IoT

## Auswertung

# Webapp Entwicklung

## Anforderungen und Design

Welche Anforderungen gibt es?

Welche Funktionen soll die Webapp bieten? (Abstrakt)

## Architektur

Beschreibung der Architektur der Web-App (z.B. Frontend-Backend-Architektur, REST-API, Real-time Updates).

## Implementierung

Vewendete Techologien wie Vue, Maps api etc.

## Testing

## Benutzeroberfläche und Interaktivität

Wie ist die Nutzeroberfläche aufgebaut?

Welche Interaktionsmöglichkeiten gibt es? (Konkret)

## Deployment und Hosting

Wie und wo wird gehostet?

Wie wurde das realisiert?

# Zusammenfassung und Ausblick

# Glossar

# Quellenverzeichnis

**3GPP** – the mobile broadband standard. (o. D.). 3GPP.

<https://www.3gpp.org/>. (Datum des Zugriffs: 18. August 2000).

**Liberg, O., Sundberg, M., Wang, E., Bergman, J. & Sachs, J.** (2017). Cellular Internet of Things: Technologies, Standards, and Performance. Academic Press.

Li, Y., Zhuang, Y., Hu, X., Gao, Z., Hu, J., Chen, L., He, Z., Pei, L., Chen, K., Wang, M., Niu, X., Chen, R., Thompson, J., Ghannouchi, F. M. & El-Sheimy, N. (2021). Toward Location-Enabled IoT (LE-IoT): IoT Positioning Techniques, Error Sources, and Error Mitigation. IEEE Internet Of Things Journal, 8(6), 4035–4062.

**Raza, U., Kulkarni, P., & Sooriyabandara, M.** (2017). Low Power Wide Area Networks: An Overview. IEEE Communications Surveys & Tutorials, 19(2), 855-873.

**Ray, A., Deb, S. & Monogioudis, P.** (2016). Localization of LTE measurement records with missing information. IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference On Computer Communications.

**Yu, K., Sharp, I. & Guo, Y. J.** (2009). Ground‐Based wireless positioning, 152-155.

## Quellen GPS:

* El-Rabbany, A. (2006). Introduction to GPS: The Global Positioning System (2nd ed.). Artech House.
* Groves, P. D. (2008). Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems (2nd ed.). Artech House.
* Hofmann-Wellenhof, B., Lichtenegger, H., & Wasle, E. (2008). GNSS – Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more. Springer.
* Kaplan, E. D., & Hegarty, C. J. (2005). Understanding GPS: Principles and Applications (2nd ed.). Artech House.
* Misra, P., & Enge, P. (2011). Global Positioning System: Signals, Measurements, and Performance (2nd ed.). Ganga-Jamuna Press.
* Parkinson, B. W., & Spilker, J. J. (1996). Global Positioning System: Theory and Applications (Vol. I & II). American Institute of Aeronautics and Astronautics.

## Quellen LTE:

* Andrews, J. G., Buzzi, S., Choi, W., Hanly, S. V., Lozano, A., Soong, A. C. K., & Zhang, J. C. (2014). What Will 5G Be? IEEE Journal on Selected Areas in Communications, 32(6), 1065-1082.
* Dahlman, E., Parkvall, S., & Sköld, J. (2018). 4G, LTE-Advanced Pro and The Road to 5G (3rd ed.). Academic Press.
* **Ericsson. (2020). LTE-M – The technology choice for low power wide area IoT. Retrieved from https://www.ericsson.com/en/reports-and-papers**
* Holma, H., & Toskala, A. (2011). LTE for UMTS: Evolution to LTE-Advanced (2nd ed.). Wiley.
* Mogensen, P., Pajukoski, K., Raaf, B., Irmer, R., & Eichinger, J. (2007). LTE capacity compared to the Shannon bound. IEEE Vehicular Technology Conference, 1234-1238.
* Palattella, M. R., Dohler, M., Grieco, L. A., Rizzo, G., Torsner, J., & Engel, T. (2016). Internet of Things in the 5G Era: Enablers, Architecture, and Business Models. IEEE Journal on Selected Areas in Communications, 34(3), 510-527.
* **Ratasuk, R., Vejlgaard, B., Mangalvedhe, N., & Ghosh, A. (2014). Narrowband LTE-M communication for IoT. IEEE International Conference on Communication, 490-495.**
* Ratasuk, R., Prasad, A., & Ghosh, A. (2015). Overview of LTE enhancements for cellular IoT. IEEE 5G World Forum, 32-38.

## Quellen Narrowband:

* Quectel. (2020). EC25 Mini PCIe LTE Module. Retrieved from Quectel
* Adelantado, F., Vilajosana, X., Tuset-Peiro, P., Martinez, B., Melia-Segui, J., & Watteyne, T. (2017). Understanding the limits of LoRaWAN. *IEEE Communications Magazine*, 55(9), 34-40.
*  Raza, U., Kulkarni, P., & Sooriyabandara, M. (2017). Low power wide area networks: An overview. *IEEE Communications Surveys & Tutorials*, 19(2), 855-873.
*  Quectel. (2019). Quectel BC66-NA Hardware Design. Quectel Wireless Solutions. Retrieved from <https://www.quectel.com>
*  Dahlman, E., Parkvall, S., & Skold, J. (2018). *4G, LTE-Advanced Pro and the Road to 5G* (3rd ed.). Academic Press.
*  Raza, U., Kulkarni, P., & Sooriyabandara, M. (2017). A Survey on Low Power Wide Area Networks for IoT Applications. IEEE Communications Surveys & Tutorials, 19(2), 1273-1290.ra
* Ratasuk, R., Mangalvedhe, N., Zhang, Y., Robert, M., & Koskinen, J.-P. (2016). Overview of Narrowband IoT in LTE Rel-13. In IEEE Conference on Standards for Communications and Networking (CSCN) (pp. 1-7). IEEE. https://doi.org/10.1109/CSCN.2016.7785170

# Stichwortverzeichnis