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

**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

Erstprüfer/in:   
Zweitprüfer/in: .

# Ehrenwörtliche Erklärung

# Kurzfassung

# Abstract

**Keywords:**

# Inhaltsverzeichnis

Ehrenwörtliche Erklärung 2

Kurzfassung 3

Abstract 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 11

3.3 Narrowband IoT 11

3.4 Vorteile und Herausforderungen von IoT-Signalen für die Lokalisierung 12

3.5 Lokalisierungsanwendungen 13

3.6 Lokalisierungsmethoden 15

3.6.1 Datenbank-Matching-Lokalisierungsmethoden 15

3.6.2 Geometrische Lokalisierungsmethoden 16

4 Mikrocontroller- und LTE-Modemeinheiten 18

4.1 Microcontroller Unit (MCU) 18

4.2 Quectel BG96 LTE Modem 18

5 AT Commands 19

6 Serialisierung und Deserialisierung von Daten 20

7 Web Services für IoT Geräte 21

8 Datenvisualisierung 22

9 Vernetzung 23

9.1 Einschalten des Modems 23

9.2 Transmission Control Protocol (TCP) 23

9.3 Secure Socket Layer (SSL) Connection 23

9.4 Message Queue Telemetry Transport (MQTT) 23

10 Standort und GPS-Tracking 24

11 Zusammenfassung und Ausblick 26

Anhang A: Beispiele für die Gliederung von Abschlussarbeiten 27

Anhang B: Formatvorlagen 28

Glossar 29

Quellenverzeichnis 30

Stichwortverzeichnis 31

# Abbildungsverzeichnis

[Abbildung 1: Abdeckungsbereiche und Energieverbrauch von IoT-Signalen (Quelle: Li et al., 2021) 13](#_Toc178295272)

[Abbildung 2: Lokalisierungssystem Architecture (Quelle: Li et al., 2021) 13](#_Toc178295273)

[Abbildung 3:Vergleich von tatsächlichen (rot) und vorhergesagten (blau) Standorten (Quelle: Ray et al., 2016) 16](#_Toc178295274)

[Abbildung 4: Prinzip der geometrischen Lokalisierungsmethoden (Quelle: Li et al. (2021)) 16](#_Toc178295275)

[Abbildung 5: Darstellung der Positionierung basierend auf TDOA-Messungen (Quelle: Yu et al. (2009)) 17](#_Toc178295276)

[Abbildung 6: BG96 mit GNSS und LTE Antenne 18](#_Toc178295277)

[Abbildung 7: DSerial und ATSerial definieren (Arduino Bootloader) 19](#_Toc178295278)

[Abbildung 8: Die sendAndSearch Methode 20](#_Toc178295279)

[Abbildung 9: Die InitModule Methode (Arduino Bootloader) 21](#_Toc178295280)

[Abbildung 10: Flussdiagramm zur Verwendung von TCP/IP-AT-Befehlen (Quelle: quectel\_bg96\_tcpip\_at\_commands\_manual\_v1-1) 23](#_Toc178295281)

[Abbildung 11: Die Methode InitAPN gemäß Flussdiagramm 25](#_Toc178295282)

[Abbildung 12: Debug-Konsole eines Programms, das einen Text an den Quectel Echo-Server sendet 26](#_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

GNSS steht für "Global Navigation Satellite System" und bezeichnet eine Vielzahl von Satellitennavigationssystemen, die weltweit zur Positionsbestimmung, Navigation und Zeitmessung genutzt werden. Diese Systeme nutzen ein Netzwerk von Satelliten, die Signale zur Erde senden, um die Position von Empfängern auf der Erdoberfläche oder in der Nähe zu bestimmen. GNSS ist ein Überbegriff, der mehrere globale und regionale Systeme umfasst.

## LTE und LTE-M

LTE ist ein bekanntes drahtloses und zellulares Netzwerk, das mobile Konnektivität für Mobiltelefone und Nutzer weltweit bereitstellt. LTE markiert den Beginn des zellularen Systems, das als 4G (vierte Generation) bekannt ist. Das 4G-System wurde in einer Reihe von Dokumenten veröffentlicht, die als Releases bekannt sind. Die Organisation, die für die Veröffentlichung dieser Dokumente verantwortlich ist, ist das 3GPP (3rd Generation Partnership Project). (3GPP – The Mobile Broadband Standard, o. D.)

LTE-M steht für LTE-Technologie für Machine-type Communications. Dieser Standard basiert auf 3GPP-Spezifikationen und verwendet Technologiebausteine von 4G und 5G. Das 3rd Generation Partnership Project (3GPP) erstellt weltweit gültige Standards für die Datenübertragung und hat somit auch den LTE-Standard definiert. Im Vergleich zu anderen Low Power Wide Area Networks (LPWAN) eignet sich LTE-M besonders für die Übertragung hoher Datenraten. Es ermöglicht bis zu 7 Mbit/s im Uplink (vom Sensor zum Netzwerk) und bis zu 4 Mbit/s im Download (vom Netzwerk zum Sensor).

## Narrowband IoT

NB-IoT ist eine Lösung für Low Power Wide Area Networks (LPWAN), die in lizenzierten Spektrumbändern arbeitet. Die 3GPP hat diese Technologie als Teil der LTE-Mobilfunknetze integriert, um von dem großen Ökosystem der LTE-Technologie und der Mobilfunkanbieter zu profitieren. NB-IoT verbessert nicht nur bestehende zellulare Anwendungsfälle, sondern erweitert diese auf eine neue Ära von Anwendungsfällen und Szenarien: massives IoT, Smart Homes, Smart Cities, Smart Transportation, Smart Grids, intelligente Versorgungsunternehmen und Zähler, Wearables und entfernte Sensoren, autonome und selbstfahrende Fahrzeuge, Objektverfolgung, mobile virtuelle Realität, Fernsteuerung und Prozessautomatisierung für Luftfahrt und Robotik sowie missionskritische Steuerung. (Liberg et al., 2017)

NB-IoT (Narrowband Internet of Things) ist darauf ausgelegt, eine große Anzahl von Geräten in verschiedenen Anwendungsbereichen zu verbinden und so das sogenannte Internet der Dinge (IoT) zu formen. Die verbundenen Geräte kommunizieren über die zellulare Infrastruktur. NB-IoT-Geräte unterstützen Datenraten von einigen Dutzend Bits pro Sekunde (bps) bis hin zu einigen hundert Kilobits pro Sekunde (Kbps). Es ist geplant, dass NB-IoT fortschrittliche Funktionen für massives IoT einführt, darunter eine hohe Anzahl von NB-IoT-Geräten pro Quadratkilometer, geschätzt auf 1 Million Geräte pro Quadratkilometer.

Es wird erwartet, dass die Anzahl der verbundenen IoT-Geräte in den kommenden Jahren explodieren wird. Zum Beispiel wird bis 2025 prognostiziert, dass mehr als 5 Milliarden Geräte über NB-IoT verbunden sein werden.

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

## Lokalisierungsanwendungen

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

Automatisch generierte Beschreibung

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

# Mikrocontroller- und LTE-Modemeinheiten

## Microcontroller Unit (MCU)

## Quectel BG96

Das BG96-Modem ist ein kompaktes und kosteneffizientes LTE Cat M1/Cat NB1/EGPRS- sowie NB-IoT- und GNSS-Modul gemäß 3GPP Release 13. Es unterstützt globale Frequenzbänder, bietet eine maximale Datenrate von 375 Kbps im Downlink und Uplink, und zeichnet sich durch niedrigen Stromverbrauch und Halbduplex-Betrieb in LTE-Netzwerken aus. Zudem ist es kompatibel mit Quectels EG91/EG95, BC95-G/BG95, UG95/UG96 und M95 Modulen.

Mit den Maßen 26,5 mm × 22,5 mm × 2,3 mm eignet sich das BG96 ideal für automatisierte Fertigungen und Hochvolumenanwendungen. Es verfügt über gängige Schnittstellen wie USB, UART und I2C sowie umfassende Internetprotokolle und unterstützt diverse Betriebssysteme (Windows, Linux, Android). Dadurch eignet sich das Modul für vielfältige IoT-Anwendungen wie drahtlose POS-Systeme, intelligente Messungen und Asset-Tracking.

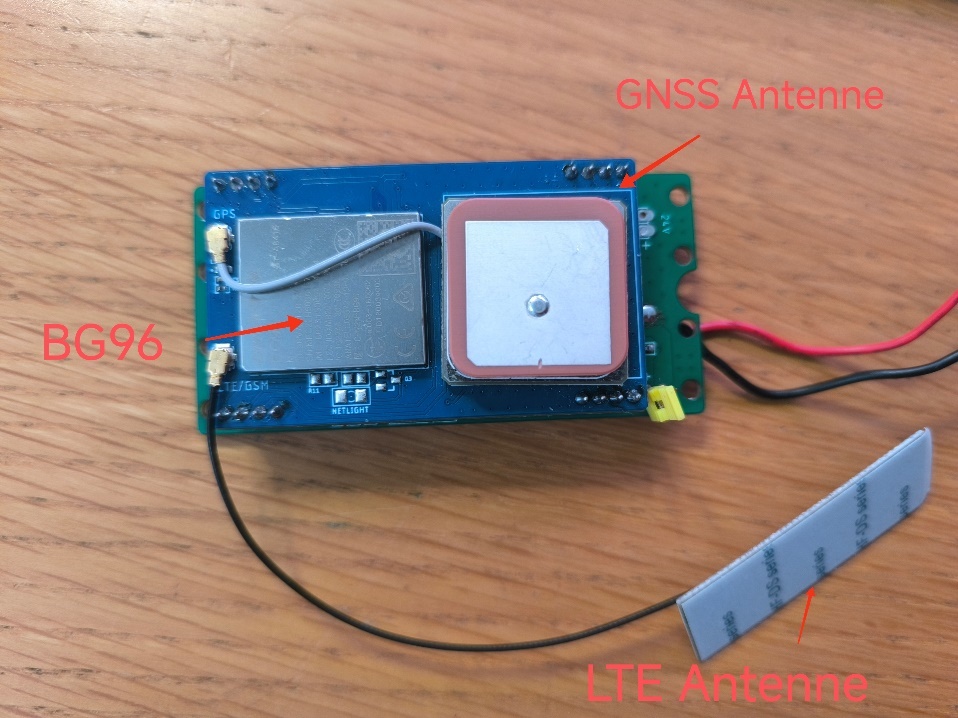


Abbildung : BG96 mit GNSS und LTE Antenne

# Kommunikation

## 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 : 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 : 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\_Commen 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 9: 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 : 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 : 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 : 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;

// Set SSL parameters

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

{

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

return false;

}

// 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 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 : Quellcode InitSSL Methode

Die Methode ‚InitSSL‘ ist verantwortlich für die Initialisierung der Secure Sockets Layer (SSL)-Konfigurationen auf einem BG96-Modul. Sie nimmt mehrere Parameter entgegen, darunter einen Index zur Identifizierung der spezifischen SSL-Instanz sowie Zeiger auf Zeichenarrays für das CA-Zertifikat, das Client-Zertifikat und den Client-Schlüssel. Zusätzlich wird ein Zeiger auf ein Zeichenarray zur Speicherung von Fehlermeldungen oder Statusinformationen übergeben. Zunächst setzt die Methode die SSL-Parameter mittels der Funktion `SetSSLParameters`, wobei die SSL-Version TLS 1.2, die Cipher Suite `TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA` und ein Timeout von 300 Sekunden festgelegt werden. Falls das Setzen der Parameter fehlschlägt, wird eine entsprechende Fehlermeldung in `err\_code` kopiert und die Methode gibt `false` zurück.

Anschließend bewertet die Methode die bereitgestellten Zertifikate, um den geeigneten SSL-Konfigurationsfall zu bestimmen. Wenn keine Zertifikate bereitgestellt werden, versucht die Methode, SSL ohne Zertifikate zu initialisieren, indem sie `SetSSLCertificate` aufruft. Bei Erfolg wird eine Bestätigungsnachricht in `err\_code` gespeichert und die Methode gibt `true` zurück, andernfalls wird eine Fehlermeldung kopiert und `false` zurückgegeben. Wird nur das CA-Zertifikat bereitgestellt, führt die Methode einen Upload des CA-Zertifikats mittels der Funktion `UploadFiles` durch. Dabei implementiert sie einen Wiederholungsmechanismus mit einem Timeout von 30 Sekunden. Sollte die Datei bereits existieren (Fehlercode 407), versucht die Methode, die bestehende Datei zu löschen und den Upload erneut durchzuführen. Nach erfolgreichem Upload setzt die Methode das SSL-Zertifikat ausschließlich mit dem CA-Zertifikat und gibt eine entsprechende Erfolgsmeldung zurück.

Sind sowohl das CA-Zertifikat, das Client-Zertifikat als auch der Client-Schlüssel vorhanden, lädt die Methode nacheinander alle drei Zertifikate hoch. Jede Upload-Operation beinhaltet einen Wiederholungsmechanismus mit einem Timeout von 30 Sekunden und behandelt bestehende Dateien, indem sie versucht, diese zu löschen und den Upload erneut durchzuführen. Nach erfolgreichen Uploads setzt die Methode die SSL-Zertifikate mit sowohl dem CA-Zertifikat als auch den Client-Zertifikaten und dem Client-Schlüssel. Bei jedem Schritt wird auf Fehler und Timeouts geprüft, wobei entsprechende Fehlermeldungen in `err\_code` gespeichert und die Methode bei einem Fehler `false` zurückgibt. Wenn alle Schritte erfolgreich abgeschlossen sind, kopiert die Methode eine Erfolgsmeldung in `err\_code` und gibt `true` zurück, was anzeigt, dass die SSL-Konfiguration erfolgreich initialisiert wurde.

Während des gesamten Initialisierungsprozesses verwendet die Methode Timeout-Prüfungen mittels der Funktion `millis`, um sicherzustellen, dass jede Operation innerhalb der vorgegebenen Zeit abgeschlossen wird. Die robuste Fehlerbehandlung stellt sicher, dass bei Problemen während des Uploads oder der Zertifikatseinstellung klare Fehlermeldungen generiert werden, was die Zuverlässigkeit und Sicherheit der SSL-Konfiguration auf dem BG96-Modul gewährleistet. Insgesamt bietet die Methode `\_BG96\_SSL::InitSSL` einen umfassenden Ansatz zur Einrichtung sicherer SSL-Verbindungen, indem sie verschiedene Authentifizierungsszenarien unterstützt und gleichzeitig eine zuverlässige Fehlerbehandlung sowie Zeitüberwachung implementiert.

## Message Queue Telemetry Transport (MQTT)

# Standort und GPS-Tracking

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

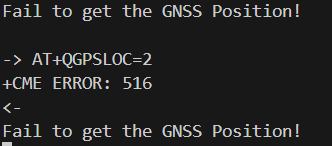
  {

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

  }

  delay(3000);

}

Fehler 516: 

# Zusammenfassung und Ausblick

# Anhang A: Beispiele für die Gliederung von Abschlussarbeiten

# Anhang B: Formatvorlagen

# 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.

# Stichwortverzeichnis