Technische Hochschule Mittelhessen

Fachbereich Informationstechnik - Elektrotechnik - Mechatronik

LoRa based IoT endpoint and gateway

Bachelorarbeit

von

Silvere Sacker Ngoufack

Betreuer:

Erster Prüfer Prof. Dr.-Ing. Hartmut Weber
Zweiter Prüfer Prof. Dr.-Ing. Martin Gräfe

Friedberg, den 26. Juni 2020

Inhaltsverzeichnis

Danksagung

Ich möchte mich an dieser Stelle bei allen derjenigen bedanken, die mich während der Anfertigung dieser Bachelorarbeit unterstützt und motiviert haben.

Zuerst bedanke ich mich herzlich bei John Martial Madieu mein Chef. Für seine hilfreichen Anregungen und seine konstruktive Kritiken bei der Entwicklung dieser Arbeit möchte ich mich bedanken.

Ich bedanke mich bei meiner Eltern, dafür das sie mich immer unterstützt haben.

Ein besonderer Danke gilt meiner Freundin und meinen Freunden, die durch ihre Fragen und Anmerkungen mein Wissen erweitert haben.

Selbständigkeitserklärung

Hiermit versichere ich, Silvere Sacker Ngoufack, die vorliegende Arbeit selbstständig und ausschließlich unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst zu haben. Inhalte dieser Arbeit, die wörtlich oder sinngemäß aus den Quellen entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit oder Teile daraus wurden in dieser oder vergleichbarer Form noch keinem anderem Prüfungsgremium vorgelegt und auch nicht veröffentlicht.

Fiedberg, den 03. August 2020

SILVERE SACKER NGOUFACK

Abbildungsverzeichnis

Tabellenverzeichnis

Abstract

1 Einleitung

Dieses Kapitel soll den Leser auf den Inhalt der Arbeit aufmerksam machen, ihn mit der Aufgabestellung vertraut machen und über die Strukturierung und Zielsetzung der Arbeit Auskunft geben.

1.1 Motivation

Objekte werden in der heutigen Tagen immer mehr mit Elektronik und Intelligenz versehen. Die Leute wollen aufgrund dieser Entwicklung, dass Prozesse oder bestimmte Aufgaben ohne menschliches Eingreifen erledigt und miteinander vernetzt werden. Das System soll nur überwacht werden und die Ergebnisse zu bestimmten Zwecken benutzt werden.

Das Internet der Dinge (in Englisch ***internet of things***, Kurzform **IoT**) wird dazu benutzt, um die Interaktion zwischen Menschen und vernetzten elektronischen Systemen zu vereinfachen.

1.2 Aufgabenstellung und Zielsetzung

Diese Bachelorarbeit beschäftigt sich mit der Entwicklung eines vernetzten Systems bestehend aus einem 3D-Beschleunigungssensor, einem 3D-Gyroskop sowie einem Temperatur- und Feuchtigkeitssensor. Die Sensoren messen Daten und übergeben diese an den STM32L475 Mikrocontroller.

Der Mikrocontroller soll die Daten verarbeiten und mit Hilfe von einem LoRa-Modul[16] drahtlos zu einem Server übertragen. Bevor die Übertragung erfolgt, muss das LoRa-Modul Zugang zu dem Server durch ein Gateway bekommen. Abbildung 1.1 gibt einen Überblick über den Aufbau des gesamten Systems.

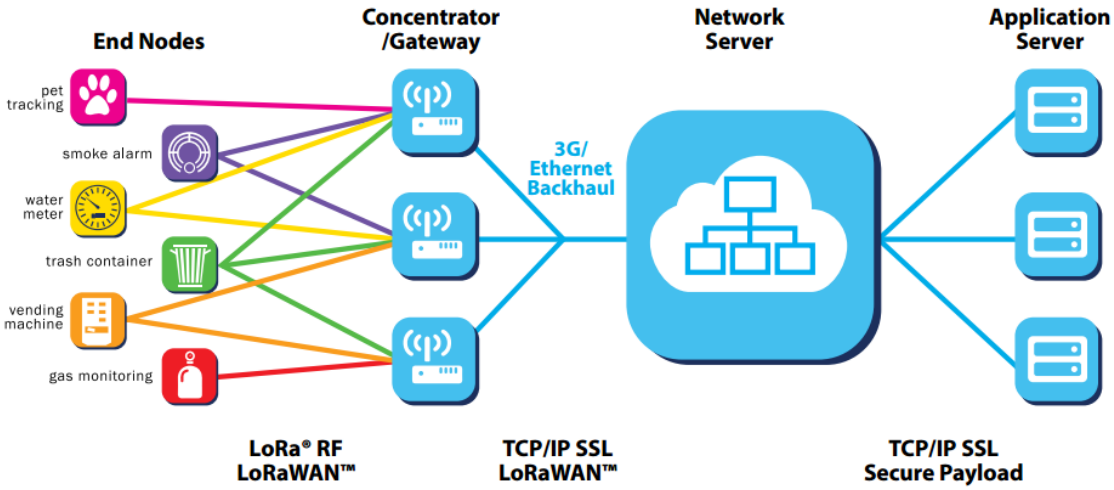

Abbildung 1.1: LoRaWAN Netzwerkarchitektur [1]

1.3 Gliederung der Arbeit

Das Kapitel 2 gibt einen detaillierten Überblick über allen Hardware-Komponenten, die bei der Durchführung dieser Arbeit verwendet werden. Hinzu kommt die Erläuterung der Software-Entwicklung. Als Erstes wird auf die Eigenschaften von des benutzten STM32-Nucleo Board eingegangen. Diesem Kapitel ist auch zu entnehmen, warum genau dieses Board gewählt wurde.

Als nächstes wird das LoRa Node und das LoRaWAN-Protokoll beschrieben. Dieser LoRa-Node wird dazu verwendet, um die gesammelten Daten dem Server drahtlos zu übertragen.

Das Kapitel 3 beschreibt wie der LoRaWAN-Server und das Gateway funktionieren und zeigt die Konfigurationen, die zu übernehmen sind, um eine Verbindung mit einem LoRa-Node herzustellen.

Anschließend im Kapitel 5 wird eine Zusammenfassung und einen kleinen Ausblick der Arbeit gegeben.

2 Hardware Komponenten

In diesem Kapitel werden die für das gesamtsystem benutzten Har hinsichtlicht ihrer funktionsweise und Ansteuerung in Einzelnen erläutert.

2.1 STM32L4 Discovery Kit

Das STM32L4 Discovery Kit ist ein IoT Knoten, womit ein Benutzer Anwendungen mit direkter Verbindung zu einem oder mehreren Cloud-Servern entwickeln kann. Dieses Discovery Kit ermöglicht eine Vielzahl von Anwendungen, indem es eine Multikink-Kommunikation (Bluetooth Low Energie) mit geringem Stromverbrauch Multiway-Erkennung der Umwelt ermöglicht (Siehe Abbildung 2.1).

Das STM32L4 hat einen eingebetteten ST-LINK Debugger/Programmierer, eingebettete Sensoren und viele andere Features, die in dem Datenblatt [13] erläutert zu finden sind. Genau wegen der Vielfalt an Eingenschaften wurde dieses Board ausgewählt. Man braucht kein Breadbord im Vergleich zu dem Arduino oder dem Raspberry-Pi, um die Sensoren mit den Schnittstellen (USART, SPI oder I2C) des Mikrocontrollers zu verbinden. Noch dazu eignet sich dieses Discovery Kit für ein LoRa-Modul von STMicroelectronics, da Arduino-Verbinder vorhanden sind. Dazu ist nur das LoRa-Modul in dieser Verbinder stecken.

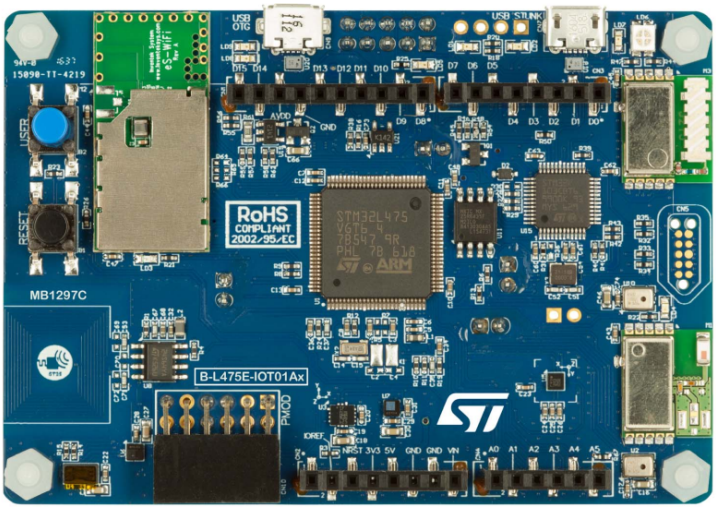

Abbildung 2.1: B-L475E-IOT01A Discovery kit [13]

Für diese Arbeit werden wir uns auf zwei Sensoren beschränken. Erstmal den HTS221[11] Temperatur- und Feuchtigkeitssensor und dann den LSM6DSK 3D-Gyroscope und 3D-Beschleunigungssensor [12]. Diese Daten werden erfasst und drahtlos an den LoRaWAN-Server übertragen. Die folgenden Unterkapitel beschreiben, wie diese Sensoren funktionieren und erklären, wie sie anzusteuern sind, damit die erhaltenen Daten im Rahmen der geforderten Toleranzen der Realität entsprechen. Laut dem Datenblatt ist mit einer Temperaturgenauigkeit von ±0.5°c und einer Feuchtigkeitsgenauigkeit von ±3.5% zu rechnen.

2.1.1 HTS221 Temperatursensor- und Feuchtigkeitssensor

In diesem Unterkapitel wird der HTS221 Temperatur- und Feuchtigkeitssensor beschrieben und erklärt wie die Temperatur und die Feuchtigkeit zu ermitteln sind.

Der HTS221 Sensor misst die relative Feuchtigkeit(H) und die Temperatur (T) und speichert die Daten (16-Bits von Datentyp Integer) als Zweierkomplement. Diese Daten können über I2C- oder SPI-Schnittstelle gelesen werden. Die gespeicherte Daten sind Rohdaten, die am Ausgang von dem Analog-Digital-Converter (ADC) verfügbar sind (Siehe Abbildung 2.2). Um die Temperatur in °c und die relative Feuchtigkeit in % zu bekommen, muss man die Daten aus den Registern auslesen und mit Hilfe der Formel 2.1.1 und 2.1.1 die richtigen Werten herausfinden.

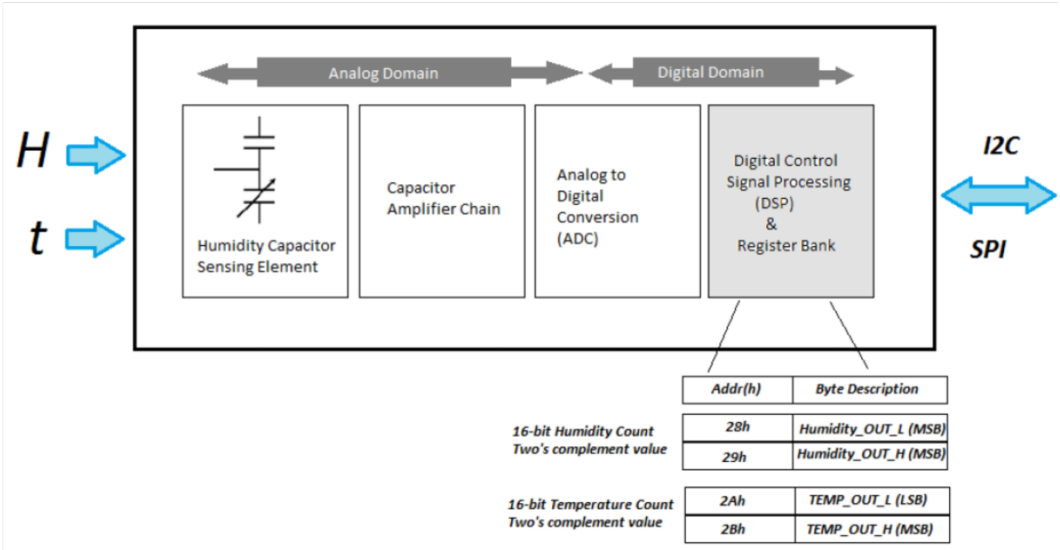

Abbildung 2.2: Humidity sensor analog-to-digital flow [11]

Feuchtigkeit ermitteln

An dieser Stelle wird erklärt wie die Feuchtigkeit von dem Sensor ermittelt wird. Der HTS221 Sensor speichert den Feuchtigkeitswert in Rohzälungen in zwei 8-Bit-Registern:

- H_OUT_H (0x29) (Höchstwertige Bits)
- H_OUT_L (0x28) (Niedrigswertige Bits)

Die zwei Bytes werden verkettet, um ein Zweierkomplement dargestelltes 16-Bit Wort zu bilden. Der relative Feuchtigkeitswert muss durch lineare Interpolation der Register (HUMIDITY_OUT_H & HUMIDITY_OUT_L) mit den Kalibrierregistern berechnet werden.

Der HTS221 Sensor ist bei der Herstellung schon kalibriert und die erforderlichen Koeffizienten sind ADC 16-Bit-Werte, die in den internen Register des Sensors zu lesen sind. Eine weitere Kalibrierung durch den Benutzer ist nicht erforderlich.

Die Tabelle 2.1 stellt die Register dar, in dem die Kalibrierwerte zur Ermittlung der relativen Feuchtigkeit gespeichert sind.

Variable	**Adresse**	**Format**[1]
H0_rH_x2	0x30	u(8)
H1_rH_x2	0x31	u(8)
H0_TO_OUT_H	0x36	s(16)
H0_TO_OUT_L	0x37	s(16)
H1_TO_OUT_H	0x3A	s(16)
H1_TO_OUT_L	0x3B	s(16)

Tabelle 2.1: Kalibrierregister für relative Feuchtigkeit

Nun wissen wir welche Register zu lesen sind, damit die relative Feuchtigkeit mithilfe der Interpolation berechnet wird. Die folgenden Schritten müssen vor der Berechnung durchgeführt werden:

- Werte von H0_rH_x2 und H1_rH_x2 aus Registern 0x30 und 0x31 lesen
- H0_rH_x2 und H1_rH_x2 durch zwei teilen
- Werte von H0_TO_OUT aus Registern 0x36 und 0x37 lesen
- Werte von H1_TO_OUT aus Registern 0x3A und 0x3B lesen
- Rohdate von H_T_OUT aus Registern 0x28 und 0x29 lesen

Nachdem diese Register gelesen wurden, kann nun die Berechnung der relative Feuchtigkeit erfolgen.

[1](u8) 16Bit-Wert ohne Vorzeichen, (s16) 16Bit-Wert mit Vorzeichen

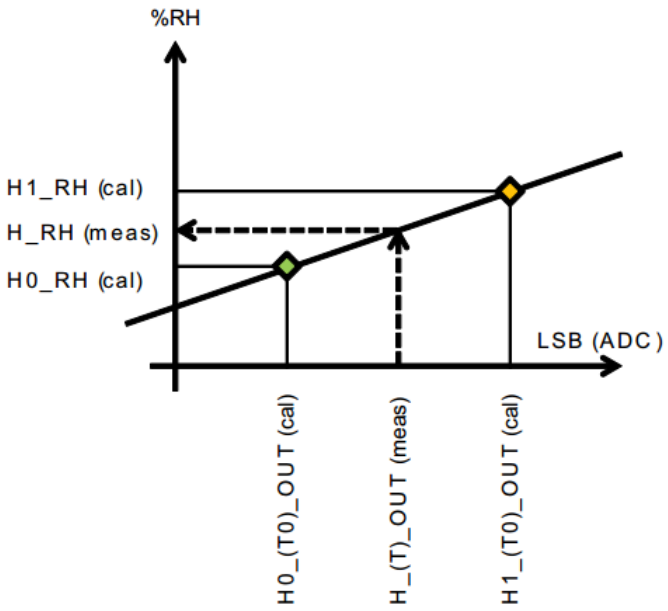

Abbildung 2.3: Linear interpolation to convert LSB to %RH [11]

Aus Abbildung 2.3 bekommt man mit Interpolation die folgende Formel [11]:

$$RH\% = \frac{((H1\_rH - H0\_rH).(H\_T\_OUT - H0\_T0\_OUT))}{(H1\_T0\_OUT - H1\_T0\_OUT)} + H0\_rH$$

Temperatur ermitteln

Der HTS221 Sensor speichert die Temperaturwerten in Rohzälungen in zwei 8-Bit-Registern:

- T_OUT_H (0x2A) (Höchstwertige Bits)
- T_OUT_L (0x2B) (Niedrigswertige Bits)

Die zwei Bytes werden verkettet, um ein Zweierkomplement dargestelltes 16-Bit Wort zu bilden. Die Polarität wird duch den höchstwertigen Bit vom T_OUT_H Register bekannt gegeben.

- Ist dieser Bit 0, die gelesene Temperatur ist positiv.
- Ist dieser Bit 1, die gelesene temperatur ist negativ. In diesem fall ist der Zweierkomplement des gesamten Wort zu bilden, um den richtigen Wert zu bekommen.

Auch hier ist die Temperatur durch lineare Interpolation von den Kalibrierregistern und den Registern T_OUT_H und T_OUT_H in Zweierkomplement zu berechnen.

Die Tabelle 2.2 stellt diese Kalibrierregister dar.

Variable	Adresse	Format
T0_degC_x8	0x32	u(8)
T1_degC_x8	0x33	u(8)
T1/TOmsb	0x35	(u2),(u2)
T0_OUT_H	0x3D	s(16)
T0_OUT_L	0x3C	s(16)
T1_OUT_H	0x3F	s(16)
T1_OUT_L	0x3E	s(16)

Tabelle 2.2: Kalibrierregister zur Temperaturermitllung

Da die Kalibrierregister vom Hersteller mit den korrekten Werten versehen werden, werden wir nun diese Register lesen und mithilfe der gelesenen Werten die Temperatur ermitteln. Bevor man die Temperatur mit linearer Interpolation berechnet, sind die folgenden Schritten erstmal erforderlich.

- Die Koeffiziente T0_degC_x8 und T1_degC_x8 aus den Registern 0x32 und 0x33 lesen
- Die Werte von T0_degC_x8 und T1_degC_x8 durch 8 divisiren, um die Koeffiziente T0_degC und T1_degC zu bekommen.
- Die höchstwertige Bits von T1_degC(T1.9 und T1.8) und T0_degC(T0.9 und T0.8) aus dem Register 0x35 lesen. Diese Werte an den im Schritt 2 ermittelten Werten verketten, damit T0_degC und T1_degC vollständig werden.
- Der Wert von T0_OUT aus den Registern 0x3C und 0x3D lesen.
- Der Wert von T1_OUT aus den Registern 0x3E und 0x3F lesen.
- Der Wert von T_OUT aus den Registern 0x2A und 0x2B lesen.

Nachdem diese Kallibrierregister gelesen wurden, kann man mittels linearer Interpolation die Temperatur in °c berechnen.

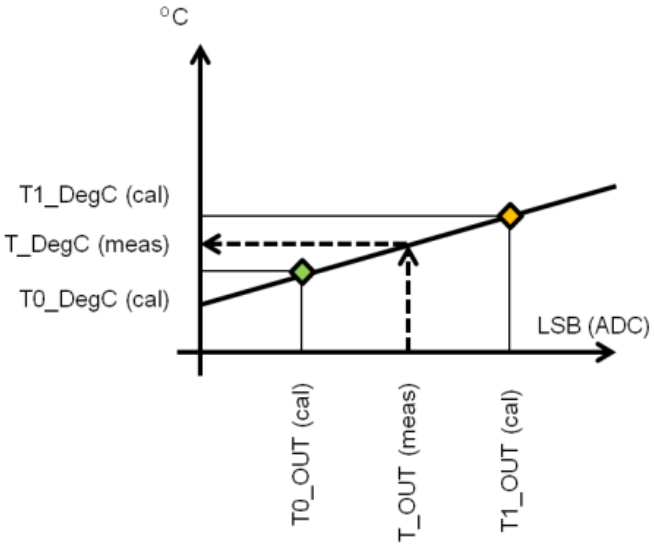

Abbildung 2.4: Linear interpolation to convert LSB to °c [11]

Abbildung 2.4 zeigt der Graph woraus die lineare Interpolation stammt. Die folgende Formel wurde sogar daraus hergeleitet.

$$T[c] = \frac{((T1\_degC - T0\_degC).(T\_OUT - T0\_OUT))}{(T1\_OUT - T0\_OUT)} + T0\_degC$$

Da die Kalibrierwerten zur Berechnung der Temperatur und der relativen Feuchtigkeit bei der Herstellung des Bausteins vorab festgesetzt sind, soll man die Kalibrierregister bei der Programmierung nur ein mal auslesen. Dies erspart den Rechenaufwand des Mikrocontrollers.

2.1.2 LSM6DSL 3D Gyroskope und 3D Beschleunigungssensor

Dieses Unterkapitel berichtet von dem LSM6DSL 3D-Gyroskope und 3D-Beschleunigungssensor. Hier ist zu entnehmen, wie die X-,Y-, und Z-Koordinaten des Sensoren zu ermitteln sind und wie der Sensor abhängig vom Zweck skaliert werden kann.

Der LSM6DSL ist ein digitaler 3D-Beschleunigungsmesser und ein 3D-Gyroskopsystem mit einer digitalen seriellen I2C/SPI Schnittstelle mit einer leistung von 0.65mA im kombinierten Hochleistungsmodus. Das Gerät verfügt über einen von Benutzer wählbaren dynamischen Beschleunigungsbereich von ±2 | ±4 | ±8 | ±16g (g is gleich 9,81m/s) und einen Winkelgeschwindigkeitsbereich von ±125 | ±250 | ±500 | ±1000 | ±2000dps (Degrees per second).

Das extrem geringe Größe und das geringe Gewicht des SMD-Packets machen den

LSM6DSL zu einer idealen Wahl für tragbare Anwendungen wie Smartphones, IoT-verbundene Geräte und andere Anwendungen, bei der reduzierte Paketgröße und -gewicht erforderlich sind.

Der LSM6DSL bietet drei mögliche Betriebskonfiguration:

- nur Beschleunigungsmesser aktiv und Gyroskope inaktiv
- nur Gyroskope aktiv und Beschleunigungsmesser inaktiv
- beide aktiv mit unabhängigem Output Data Rate (ODR)

Der Beschleunigungsmesser und der Gyroskop können unabhängig voneinander konfiguriert werden unteranderem: Power-down, Low-Power, Normal- und High- performance Modus. Um den Stromverbrauch des Sensors zu reduzieren, kann der Gyroskop in einem Ruhestand gesetzt werden.

Sobald das Gerät mit Strom versorgt wird, werden die Kalibriekoeffizienten vom eingebettetem Flash-Speicher zu den internen Registers. Dieser Vorgang dauert ungefähr 15 milisekunden. Nach dieser Zeit fallen der Beschleunigungsmesser und der Gyroskop im Power-Down Modus. Durch der CTRL1_XL bzw. CTRL2_G register können die Geräte geweckt werben, indem man das Betriebmodus auswählt.

Wenn die Daten verfügbar sind, wird eine Unterbrechung (Interrupt demnächst) ausgelöst, wenn der entsprechende Byte vom Beschleunigungsmesser bzw. vom Gyroskop in dem INT1_CTRL Register geschrieben wurde. Das vorhandensein der Daten kann nun mit Hilfe des Statusregister abgefragt werden. Das XLDA-Bit wird auf 1 gesetzt, wenn am Ausgang des Beschleunigungsmesser ein neuer Datensatz verfügbar ist. Das GDS-Bit wird suf 1 gesetzt, wenn am Gyroskopausgang ein neuer Datensatz verfügbar ist.

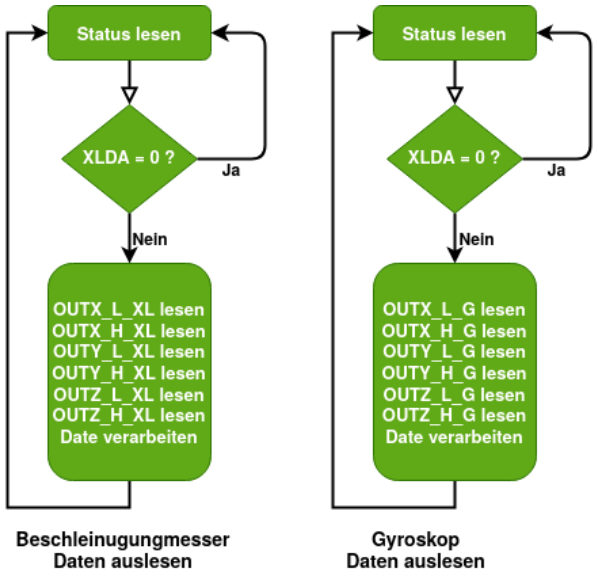

Abbildung 2.5: Flußdiagram zur Datenermitlung

Das Abbildung 2.5 ist der Flußdiagram zur ermitllung der Achsen- und Winkelveränderungen des Beschleunigungsmesser und des Gyroskop.

Wie oben schon erläutert, kann das Gerät so eingestellet werden, dass ein neuer Satz von Messdaten durch ein Signal erkennbar wird. Das XLDA-Bit von dem STATUS_REG-Register wird auf 1 gesetzt, wenn Daten aus dem Beschleunigungsmesser zum Lesen verfügbar sind. Das Signal kann durch den INT1-Pin angesteuert werden, indem das INT1_DRDY_XL-Bit vom INT1_CTRL-Register auf 1 gestezt wird.

Für den Gyroskopsensor wird das GDA-Bit auf 1 gesetzt, wenn die Daten zum lesen verfügbar sind. Das Signal kann durch den INT1-Pin angesteuert werden, indem das INT1_DRDY_G-Bit vom INT1_CTRL-Register auf 1 gestezt wird. Die gemessenen Beschleunigungsdaten werden in OUTX_H_XL-, OUTX_L_XL-, OUTY_H_XL-, OUTY_L_-XL-, OUTZ_H_XL-, OUTZ_L_XL-Register gesendet. Die gemessenen Winkelgeschwindigkeitsdaten werden dagegen in OUTX_H_G-, OUTX_L_G-, OUTY_H_G-, OUTY_L_G-, OUTZ_H_G-, OUTZ_L_G-Register gesendet. Die vollständigen Ausgangsdaten für die X-,Y- und Z-Achsen sind durch die Verkettung von OUTX_H_XL(G) und OUTX_L_-XL(G), OUTY_H_XL(G) und OUTY_L_XL(G), OUTZ_H_XL(G) und OUTZ_L_XL zu erhalten, wobei die Beschleunigungsdaten und die Winkelgeschwindigkeitsdaten als 16-Bit Werte dargestellt sind.

Mit dem LSM6DSL kann man den Inhalt des unteren und oberen Teils der Ausgangsdatenregister vertauschen, sodass die Darstellung entweder Big-Endian oder Little-Endian entspricht. Dies ist möglich, sofern man das BLE-Bit von dem CTRL3_C-Register auf 0 (Little-Endian Standartmäßig) oder auf 1 (für Big-Endian) Big-Endian bedeutet, dass der höchstwertige Byte des Datensatzes in der niedrigsten Speicherstelle gespeichert wird. Little-Endian bedeutet, dass der niedrigswertige Byte des Datensatzes in der niedrigsten Speicherstelle gespeichert wird.

Im Unterkapitel 4.2 weden die Funktionen zur Datenermittlung in der Programmiersprache C sowohl für den HTS221 (Temperatur- und Feuchtigkeitssensor) als auch für den LSM6DSL (3D-Beschleunigungsensor und 3D-Gyroskop) dargestellt und erklärt wie die Kommunicationsschnitstelle (hier I2C) zu benutzen ist.

2.2 LoRa Node: i-nucleo-lrwan1

Die im Kapitel 2.1.1 ermittelten Sensordaten sollen laut der Aufgabestellung mit Hilfe eines drahtloses Protokol an einem Server versand werden. Um diese Daten drahtlos und auf eine lange Strecke zu übertragen, haben wir uns auf das LoRaWAN-Protokol entschieden. Die Gründe warum genau dieses Protokol ausgewählt wurde, werden in diesem Kapitel genannt. Noch dazu wird nicht nur auf die Eigenschaften des benutzten LoRa-Modul eingegangen sondern auch auf den Unterschied von diesem Modul gegenüber anderen Modulen, die auf dem Markt zu finden sind.

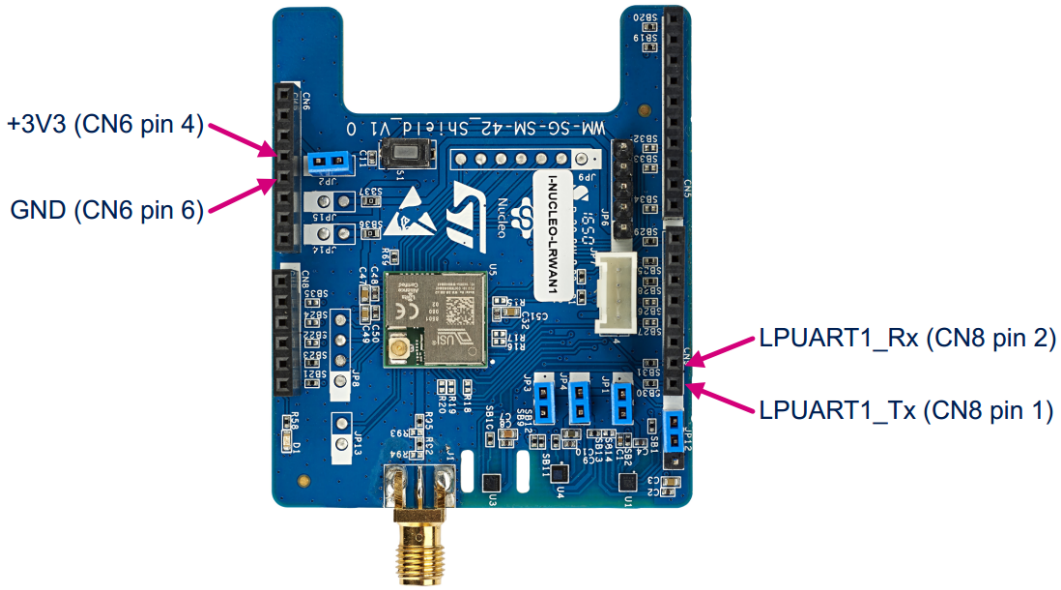

Abbildung 2.6: I-Nucleo-LRWAN1 [15]

Abbildung 2.6 zeigt das LoRa-Modul, das zur Datenübertragung verwendet ist. Diese Platine mit Arduino-Connectoren und mehr ist eine integrierte Lösung, die jedem ermöglich Anwendungen mit der LoRa-Technologie zu entwickeln. Das I-Nucleo-LRWAN1 verfügt über den USI® LoRaWAN™ Technologiemodul für konstengünstiges und stromsparendes Weitverkehrsnetz (LPWAN), welches mit einem eingebetteten Stapel von AT-Befehle (AT-Befehl beschreiben) mitgeliefert wird. Dieses Board wurde ausgewählt, weil es durch ein externes Board wie das Nucleo-L053 oder user B-L475E-IOT01A Discovery kit 2.1 über mehrere Schnittstellen wie LPUART, SPI oder I2C angesteuert werden kann. Noch dazu verfügt das I-Nucleo-LRWAN1 über die folgenden eingebetteten Sensoren.

- ST Beschleunigungs- und Magnetosensor (LSM303AGR)
- ST relative Feuchtigkeits- und Temperatursensor (HTS221)
- ST Drucksensor (LPS22HB)

Im Vergleich zu anderen LoRa-Modulen, worauf keine Sensoren vorhanden sind, braucht man keine zusätzliche Sensoren kaufen. Die Kommunikation mit einem anderen Mikrocontroller erfolgt einfach durch UART, man braucht nicht auf das integriete Radio-Modul zum Senden oder Empfangen von Daten und Befehle zu kümmern. Das Bild 2.7 zeigt, dass das I-Nucleo-LRWAN1 mit einem STM32L0-Mikrocontroller versehen ist, der dazu zuständig ist, die Kommunikation zwischen dem I-Nucleo-LRWAN1 und einem externen Mikrocontroller zu vereinfachen. Das SX1272-Chip ist das eigentliche LoRa-Radio-Modul, das die Daten oder die Befehle per Funk durch die Antenne an entweder einem Gateway (Router) oder einem anderen LoRa-Node sendet.

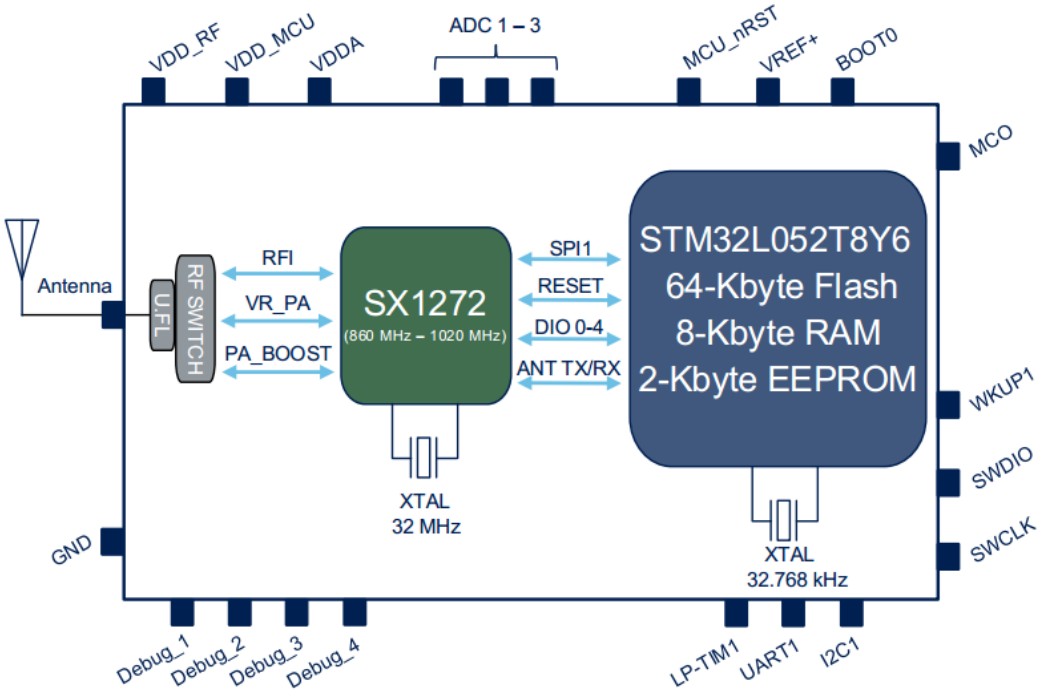

Abbildung 2.7: I-Nucleo-LRWAN1 Architektur [15]

2.2.1 LoRaWAN Protokol

Aktivierung durch OTAA

Aktivierung durch ABP

2.2.2 AT Commandos

3 Gateway und LoRaWAN-Server

3.1 Gateway

3.1.1 Was ist das Lora packet forwarder

3.2 Einstellung des LoRaWAN-Servers

3.3 MQTT brocker zur Datenübertragung

4 Software-Entwicklung

4.1 Entwicklungsumgebung

4.1.1 Eclipse

4.1.2 Libopencm3 Bibliotek installieren

4.2 Sensoren Auslesen

4.3 LoRa Commandos senden

5 Fazit

5.1 Zusammenfassung

5.2 Ausblick

Literaturverzeichnis

[1] LoRa Alliance. What is lorawan. 2020. https://lora-alliance.org/resource-hub/what-lorawanr.

[2] Warren Gay. *Beginning STM32. Developing with FreeRTOS, libopencm3 and GCC*. 2018. https://doi.org/10.1007/978-1-4842-3623-9.

[3] Petr Gotthard. Compact server for private lorawan networks. 2016. https://github.com/gotthardp/lorawan-server/tree/master/doc.

[4] Brian Gough. *An Introduction to GCC for the GNU Compilers gcc and g++*. Network Theory Limited, Bristol UK, 2005. http://www.network-theory.co.uk/gcc/intro/.

[5] libopencm3. libopencm3 lowlevel open-source library for arm cortex mcus. 2013. https://github.com/libopencm3.

[6] Muhammad Ali Mazidi, Shujen Chen, and Eshragh Ghaemi. *STM32 ARM Programming for Embedded Systems. Using C language with STM32F4 ARM*. 2014.

[7] Richard Reese. *Understandidng and using C pointers*. O'Reilly Media, Sebastopol, 2013.

[8] Mickael Remond. *Erlang Programmation*. 2003.

[9] saleae. Debug hardware like the pros with the logic analyzer you will love. 2020. https://www.saleae.com/.

[10] Semtech. Lora network packet forwarder project. 2013. https://github.com/Lora-net/packet_forwarder.

[11] STMicroelectronics. Capacitive digital sensor for relative humidity and temperature. STMicroelectronics, 2018. https://www.st.com/en/mems-and-sensors/hts221.html#resource.

[12] STMicroelectronics. inemo inertial module: always-on 3d accelerometer and 3d gyroscope. 2018. https://www.st.com/en/mems-and-sensors/lsm6dsl.html#resource.

[13] STMicroelectronics. Discovery kit for iot node, multi-channel communication with stm32l4. 2019. https://www.st.com/en/evaluation-tools/b-l475e-iot01a.html#resource.

[14] STMicroelectronics. dm00083560-stm32l47xxx-stm32l48xxx-stm32l49xxx-and-stm32l4axxx-advanced-armbased-32bit-mcus-stmicroelectronics. 2019. https://www.st.com/en/evaluation-tools/b-l475e-iot01a.html#resource.

[15] STMicroelectronics. Usi lora expansion baord for stm32 nucleo. 2019. https://www.st.com/en/evaluation-tools/i-nucleo-lrwan1.html#resource.

[16] USI. Wm-sg-sm-42 at command reference manual. 04 2018. https://github.com/USIWP1Module/USI_I-NUCLEO-LRWAN1/tree/master/preloaded_firmware.