University of Zurich[UZH]

MASTER THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Secure Position Data Transmission for Object Tracking using LoRaWAN

Matthias Diez
Zurich, Switzerland
Student ID: 10-726-255

Supervisor: Dr. Corinna Schmitt, Sina Rafati
Date of Submission: August 24, 2017

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

CSG
Department of Informatics
Communication Systems Group

Master Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

In this Master thesis, a novel object tracking device based on an Arduino microcontroller is presented. This so called 'tag' uses the low power technology Long Range Wide Area Network (LoRaWAN) to enable communication and a Global Navigation Satellite System (GNSS) module for localization. In order to only access the GNSS module with its high energy consumption in case of suspected movement, an accelerometer is attached to the tag. The Things Network (TTN) is used as free of charge LoRaWAN provider. The TTN and its registered tags are integrated into the existing 'Object Tracking using the Internet of Things' (OTIoT) platform using two application programming interfaces. It is shown that the combination of LoRaWAN and GNSS is well-suited for the object tracking use case and that all features necessary for bi-directional communication with the tag were implemented. At the submission of this Master thesis, the tag is in a prototypical state and is not yet optimized with regards to size, energy consumption, and hardware costs.

Zusammenfassung

In dieser Masterarbeit wird ein neuartiges Gerät zur Ortung von Objekten auf Basis eines Arduino-Mikrocontrollers vorgestellt. Dieser sogenannte 'Tag' nutzt eine energiesparsame Technologie namens Long Range Wide Area Network (LoRaWAN) zur Kommunikation und das Global Navigation Satellite System (GNSS) zur Lokalisation. Um das energieintensive GNSS-Modul nur dann zu nutzen, wenn ein Verdacht auf eine Bewegung des überwachten Objekts vorliegt, ist zusätzlich ein Beschleunigungssensor verbaut. Das The Things Network (TTN) wird als kostenloser LoRaWAN-Anbieter eingesetzt. Über Anwendungsprogrammierschnittstellen des TTN wird das Netzwerk und dessen registrierte Tags in die vorhandene Ortungsplattform namens 'Object Tracking using the Internet of Things' (OTIoT) integriert. Es kann gezeigt werden, dass sich die Technologiekombination von LoRaWAN und GNSS gut für die Ortung von Objekten eignet. Alle Funktionalitäten, welche zur bidirektionalen Kommunikation mit dem Tag nötig sind, konnten implementiert werden. Zum Zeitpunkt der Abgabe dieser Masterarbeit ist der Tag ein Prototyp, welcher im Hinblick auf seine Grösse, seinen Energieverbrauch und seine Hardwarekosten noch nicht optimiert worden ist.

Acknowledgments

First of all, I would like to thank Prof. Dr. Burkhard Stiller for giving me the opportunity to write my Master thesis on the very current and fascinating topic of Low Power Wide Area Network at his Communication Systems Group at the Department of Informatics at the University of Zurich.

I would also like to thank Dr. Corinna Schmitt, who fulfilled her role as advisor for this thesis with great enthusiasm. Her valuable feedback helped to improve both the solution as well as this written document. Also, her positive attitude encouraged me throughout the process to do my best for this thesis.

Furthermore, I would like to thank Gonzalo Casas, the community leader of Zurich's The Things Network community, for his valuable insights into LoRaWAN and The Things Network, as well as for lending me his LoRa gateway early in the development process. Daniel Eichhorn (Netcetera AG) later on lent his LoRa gateway components to me, which helped me to further improve the object tracking device and which I am very thankful for.

Special thanks are also due to my parents, Christoph and Elisabeth Diez, who gave me the opportunity to study at University of Zurich and who always show interest in what I am currently up to at university and in life. Last but not least, I would like to thank my girlfriend Caroline Tanner for her encouragement throughout the process of writing this thesis, for proofreading its text, and for her understanding for my time spent on this thesis.

Table of Content

Abstract **I**

Zusammenfassung **III**

Acknowledgments **V**

1 Introduction **1**

1.1 Description of Work 2

1.2 Thesis Outline 2

2 Background **3**

2.1 Communication Technologies 3

2.2 Localization Methods 5

2.3 Tag Hardware 7

3 Design Decisions **9**

3.1 Taken Decisions 9

3.2 Intended Architecture and Data Flow 11

4 Implementation **13**

4.1 LoRa and LoRaWAN 13

4.1.1 Security and Device Activation 15

4.2 The Things Network 17

4.2.1 Detailed Architecture 17

4.2.2 Usage of The Things Network in OTIoT 20
4.2.3 Limitations of LoRaWAN and The Things Network 24
4.3 OTIoT Tag 25
4.3.1 OTIoT Tag Hardware 25
4.3.2 OTIoT Tag Software 27
4.4 FMLR TrackerTwo 33
4.5 LoRa Gateway 35
4.6 OTIoT Back-End 37
4.6.1 Integration using the The Things Network APIs 37
4.7 OTIoT App 40

5 Evaluation **41**
5.1 Test Environment 41
5.2 Integration Test 42
5.3 Downlink Communication Test 43
5.4 Location Sending Algorithm Test 43
5.5 Findings 46

6 Summary and Conclusions **49**

References **51**

Abbreviations **61**

Glossary **65**

List of Figures **67**

List of Tables **69**

List of Listings **71**

A OTIoT Tag Deployment Manual **73**

B Contents of the CD **75**

Chapter 1

Introduction

It is a very uncomfortable feeling that one gets when an object is lost or stolen. Especially if it concerns a valuable asset that has gone missing. Therefore, it is common user's need to be able to precisely locate an object. For our smartphones and laptops, which have all necessary sensors for communication and localization on board, this is already a reality with applications such as Apple's 'Find my iPhone' [1], Google's 'Find My Device' [2], and the independent 'Prey' application [3]. For other, in particular offline objects (e.g., a bicycle or a backpack), the situation looks different: Today's stand-alone mobile location tracking solutions are cumbersome to use. They require frequent charging of the tracking device, which is particularly tedious if it is attached in a fixed manner to the tracked object. In addition, most of the available tracking solutions force the user to buy a Subscriber Identity Module (SIM) card with a mobile data contract (e.g., 'Tracker.com' [4]), or they offer only limited signal range due to the communication technology used (e.g., 'Tile' [5]). This Master thesis offers an alternative tracking solution using the Internet of Things (IoT) [6], tackling these two drawbacks of existing solutions.

This Master thesis is embedded into the University of Zurich (UZH) Communication Systems Group's research area 'Object Tracking using the Internet of Things' (OTIoT), which was started in late 2016. The overall aim of OTIoT is to provide a platform, which is able to collect location data from many different tracking devices using appropriate interfaces, and which enables the management of the tracking devices as well as the user-friendly display of their location data.
It is Jan Meier's Master thesis [7], submitted in May 2017, which has laid the foundation for OTIoT with the development of a modern, web-based back- and front-end. Figure 1.1 shows the split in responsibility within OTIoT, with Jan Meier's part on the right and the author's part on the left hand side ('Data Gathering').

In the following, the terms 'OTIoT Back-End' for the web server with database and Application Programming Interfaces (APIs), 'OTIoT App' for the web-based front-end serving as user interface, and 'OTIoT Tag' for the location tracking device are used. The term 'OTIoT Platform' encompasses all three of these components, which build a functional system together. Furthermore, 'OTIoT TTN Application' is used as term to describe the application registered for OTIoT in The Things Network. When writing about the research area as such, the term 'OTIoT' is used.

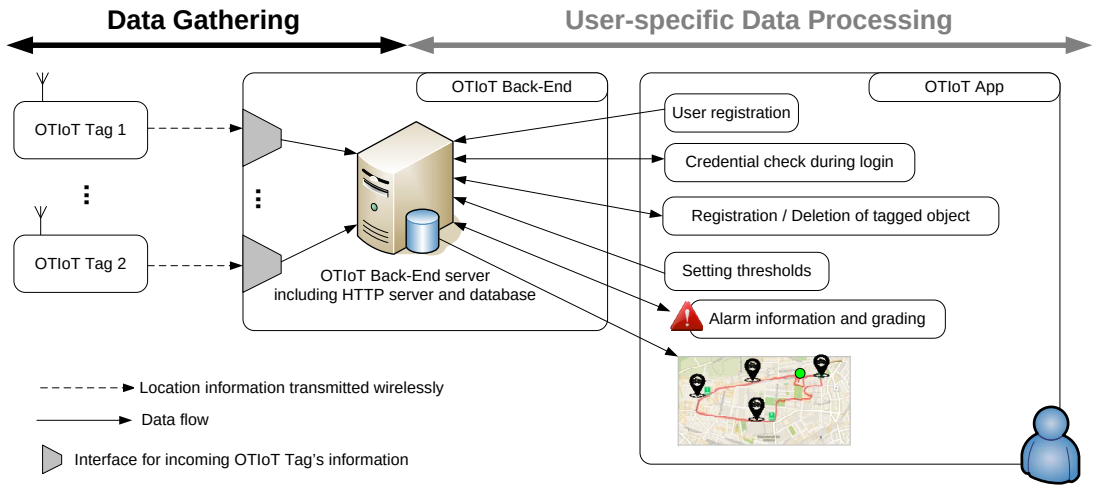

Figure 1.1: OTIoT work package setup

1.1 Description of Work

The focus of this Master thesis is the data gathering part of OTIoT. More specifically, the goal is the hardware selection and assembly as well as the embedded software development of a battery-powered prototypical device, the OTIoT Tag, which is able to send location data of tagged objects to the OTIoT Back-End. Other than the development of the OTIoT Tag, the data integration using multiple APIs (named 'Interfaces' in Figure 1.1) – in order for the OTIoT App to be able to display location data – forms an integral work package. With the finished OTIoT Tag and a self-built gateway, multiple tests are conducted to evaluate the potential of the OTIoT Platform in general, and the OTIoT Tag in particular.

1.2 Thesis Outline

First, different available communication technologies, localization methods, and tag hardware options are presented and discussed in Chapter 2.
This background is then used in Chapter 3 to take and justify the required design decisions regarding the development of the OTIoT Tag.
In Chapter 4, details about chosen technologies are given, the implementation of both the hardware and the software part of the OTIoT Tag is presented, and the integration of the OTIoT Tag into the OTIoT Back-End is explained. In addition, the gateway used and an alternative tracking device is presented.
Next, multiple tests are presented and their findings are discussed in Chapter 5, building the evaluation of the OTIoT Tag and the OTIoT Platform. Following the discussion of the findings, the limitations of the current OTIoT Tag prototype are explained.
In Chapter 6, the content of this thesis is summarized, conclusions are drawn, and pointers to potential future work are given.
In Appendix A, a manual on how to deploy the OTIoT Tag, and in Appendix B, a list of the contents of the compact disk delivered with this Master thesis can be found.

Chapter 2

Background

The goal of this chapter is to give an overview on available communication technologies, localization methods, and tag hardware options. This background knowledge is used to take the appropriate design decisions for the development of the OTIoT Tag in Chapter 3.

2.1 Communication Technologies

In this section, different communication technologies for mobile devices, which could enable data transfer from the OTIoT Tag to the OTIoT Back-End, are examined. Wired communication technologies such as fiber-optic cables or power-line connections are not investigated as they are not relevant for a location tracking system, which inherently consists of mobile nodes.
The differentiation criteria considered for mobile communication technologies are range, power consumption, bandwidth, radio chipset costs, radio subscription costs, number of base stations, transmission latency, and geographical coverage/penetration [8].

Bluetooth Low Energy (BLE) [9], **Radio-Frequency Identification (RFID)** [10], and **Near Field Communication (NFC)** [11] are commonly used for close proximity communication ranging from some centimeters for NFC [12] up to about 10 m for BLE [10], with RFID being situated in between with ranges from 10 cm to 20 m for passive tags [13]. The main advantage of these three technologies is their low power consumption. 'TrackR' [14] is an example of a tracking solution using BLE and leveraging the crowd: Every owner of a TrackR sensor can help to locate other, potentially missing TrackR sensors using their smartphone application when nearby.

Wireless Local Area Network (WLAN) is a popular wireless communication technology option, particularly used for indoor scenarios such as offices. It offers high-speed data rates and a range of up to about 50 m, depending on the topology [10]. Therefore, it is primarily aimed at providing network access in local areas. Thus, there is no country-wide coverage available with WLANs. However, some examples of (inner) city-wide WLAN coverage such as a deployment in Helsinki in Finland [15] and Google WiFi

in Mountain View [16] exist. These deployments are associated with high costs due to the number of access points needed to provide good coverage over a large area.

Using the widely available **cellular network**, based on Global System for Mobile Communications (GSM), is an obvious option, as almost full coverage is offered in developed countries such as Switzerland. Today's used standards for cellular telecommunication are General Packet Radio Service (GPRS), Enhanced Data Rates for GSM Evolution (EDGE), 3G, 4G/Long Term Evolution (LTE) [17].
Long Term Evolution-Machine (LTE-M) [18] and NarrowBand IoT (NB-IoT) [18] are the two main proposals for a cellular IoT standard by 3rd Generation Partnership Project (3GPP) [19]. First LTE-M deployments are available now, such as Verizon's Cat M1 US-nationwide network for IoT [20], which was launched March 31, 2017. As NB-IoT does not operate in the LTE construct [21], it is not yet deployed by many providers. In January 2017, one of the first NB-IoT networks was commercially launched in Spain [22], for example.
An example for traditional, cellular GPRS usage in object tracking is 'Tracker.com' [4], a solution developed for logistics, sporting activities, and object tracking using GPS and GPRS. Tracker.com is cooperating with Swisscom by using their network infrastructure and their SIM cards. While offering great coverage and data rates, the high power consumption of cellular modules and the cost of mobile data contracts are obvious drawbacks of this communication technology.

A relatively new category of communication technologies are **Low Power Wide Area Networks (LPWANs)**. LPWANs aim to combine the low power consumption of technologies such as BLE with the long range of cellular networks. This combination comes at the cost of low data rates. LPWANs mainly cater for small, battery-powered sensors with relatively few data sent per day (e.g., battery-powered sensor nodes, smart meters, or weather stations). [23]
A comparison of the properties of LPWANs with the cellular network (3G/4G/5G) as well as the Wireless Personal Area Network (WPAN) standard ZigBee [24], which is based on the Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 standard[25], is shown in Figure 2.1. It can be seen that LPWANs are superior to ZigBee and the cellular network in terms of range, geographical coverage/penetration, power consumption, and radio chipset costs. However, the bandwidth is many times smaller and the transmission latency is higher than what we are used to from mobile phone usage.
Sigfox [26], a French company founded in 2009 as a startup, has established one of today's available LPWAN standards. Only very small amounts of data (12 Bytes) can be sent very slowly (300 Baud) using binary phase-shift keying [23]. By sending very long and very slow messages, long ranges are possible [23]. Sigfox aims to become a global IoT network operator by either providing the network infrastructure themselves (as in the case of France and USA) or by negotiating exclusive contracts with local telecommunication providers. In Switzerland, there is no Sigfox coverage available as of mid 2017. This is potentially due to a lack of a telecommunication provider as network partner. The patented Sigfox chip technology is licensed by Sigfox to different chip manufacturers. Sigfox chips are available at a relatively low cost [23].
Another LPWAN standard is **Long Range Wide Area Network (LoRaWAN)** [27]. The physical Long Range (LoRa) layer uses chip technology that is patented by US-based, long established chip manufacturer Semtech Corporation [28]. As opposed to Sigfox, LoRa

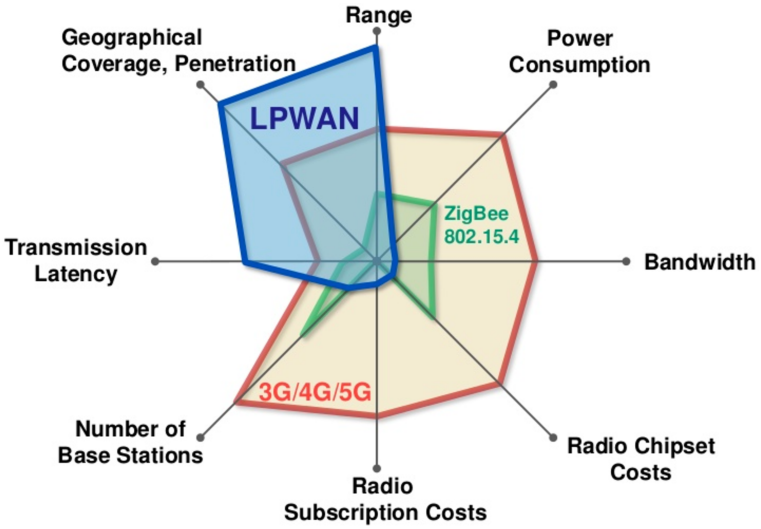

Figure 2.1: Comparison of LPWAN, cellular network, and ZigBee [8]

chips are currently only manufactured by Semtech Corporation themselves and there are no licensing agreements with other chip manufacturers. Using a variant of Chirp Spread Spectrum (CSS) modulation in its radio transmission to achieve strong signals over long distance, data rates up to 50 000 bits per second (bps) and ranges up to 15 km in sub-urban areas [29] can be achieved. CSS modulation was originally developed for robust, low-power radar applications in the 1940s [30]. Another benefit of using LoRaWAN is the availability of low cost hardware.
On the network side, Semtech decided to take a distributed approach to roll out their network by founding the LoRa Alliance [31], a loose conglomerate of LoRaWAN providers and contributors worldwide. Examples for LoRaWAN providers are the The Things Network (TTN) [32], an open infrastructure initiative with local communities world-wide, and the Swisscom Low Power Network (LPN) [33], which is a commercial alternative offered by Switzerland's biggest telecommunication provider, Swisscom. TTN was one of the early LoRaWAN providers when it was founded in Amsterdam in August 2015 [34] and offers free access to the network with the help of crowd-sourced LoRaWAN gateways. Swisscom LPN was launched for public use in October 2016 [35] and offers different paid plans, depending on the required uplink and downlink budget per device.

2.2 Localization Methods

This section presents different localization methods, which are used to determine the absolute location of a tracking device by giving global latitude and longitude information. The presented methods as well as more localization methods can be found in Rainer Mautz' Habilitation thesis on 'Indoor Positioning Technologies' [36], which this section is mainly based on.

The most basic form of localization, offered in principle by all communication technologies, is locating a device using the localized base station it is connected to. This localization method is called **Cell of Origin**. The accuracy of the location increases with decreasing

range of the receiver in this case. This approach is used, for example, to locate parcels equipped with passive RFID tags when they pass through portals in logistics distribution centers. [36]

Multilateration makes use of multiple distance measurements from base stations to a mobile node. Using hyperbolic intersection of the spheres around three or more base stations, the location of the mobile node can be determined. Time of Arrival (TOA), Time Difference of Arrival (TDOA), Round Trip Time (RTT), or Received Signal Strength Indicator (RSSI) are often used as distance measures, which determine the radius of each sphere as shown in Figure 2.2. Multilateration is possible with BLE, active RFID, WLAN, GSM, Sigfox [37], and LoRaWAN [38]. The gateways are required to know their locations for this method to provide an absolute location of the mobile node. In addition, when using TOA as distance measurement, precise synchronization of both transmitter and receiver clocks are necessary for accurate positioning. With TDOA, only the receivers (e.g., gateways) have to synchronize their clocks precisely. This clock synchronization is often done using satellite communication modules. [36]

In contrast to multilateration, **multiangulation** uses multiple Angle of Arrivals (AOAs) instead of distances to calculate a mobile node's location. The main requirement for multiangulation is the use of directionally sensitive antennas, which is the case for some large GSM base stations, for example. [36]

Fingerprinting relies on an offline calibration phase, in which measurements are either collected at known locations ('taking a fingerprint') or computed analytically using a signal propagation model to finally compile a database. The most common measure in fingerprinting databases is RSSI in decibels. In the operation phase, the received RSSI-tuples at the mobile node are matched with tuples from the database to determine the node's location. The Google Geolocation API [39] is an example of a localization service, which makes use of fingerprints from the cellular network as well as from WLANs to estimate a mobile phone's location. OpenCelliD [40] offers a similar service for GSM. Fingerprinting is also a common approach for indoor navigation using BLE or WLAN. [36]

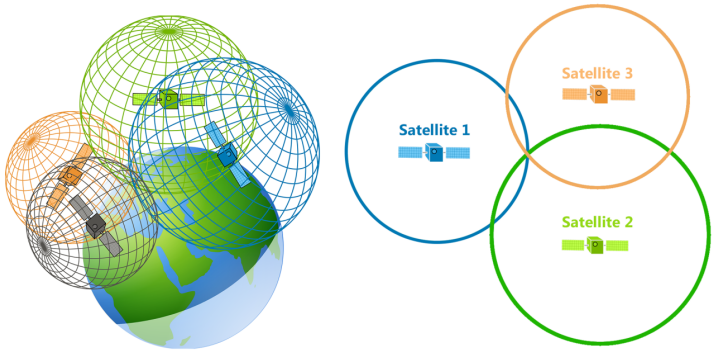

Figure 2.2: Multilateration used by navigational satellites [41]

The **Global Navigation Satellite System (GNSS)** consists of the global satellite systems Global Positioning System (GPS) operated by the United States of America, Globalnaja nawigazionnaja sputnikowaja sistema (GLONASS) by Russia, Galileo by the

European Union, and BeiDou by China. The multilateration approach in GNSS is illustrated in Figure 2.2. While the time-synchronized satellites in any GNSS subsystem also use multilateration to calculate the location of the receiver, the difference in this case is that GNSS cannot be used as the underlying communication technology for a tracking solution at the same time. In order to make the location data available via the Internet, a complimentary communication technology must therefore be chosen. This is also the main drawback of the usage of GNSS, as an extra GNSS module mean more energy consumption and a higher total cost of the tracking device. Another disadvantage is that GNSS localization requires a line of sight to the satellite, which is only possible outdoors. Therefore, GNSS is unsuited for indoor localization. The advantages are a global outdoor coverage as well as a high accuracy of the positioning (typically around 10 m). [36]

2.3 Tag Hardware

The two hardware ecosystems Raspberry Pi [42] and Arduino [43] are now presented to give a basis for the design decisions regarding the OTIoT Tag's hardware taken in Section 3.1. The Arduino Uno and Raspberry Pi 3 Model rB are shown in Figure 2.3.

A **Raspberry Pi** is a general-purpose computer that features a fully functional operating system, usually a Linux distribution. Version 3 of the Raspberry Pi [44] comes with the Broadcom chip BCM2837 that uses a quad-core ARM Cortex A53 (ARMv8) cluster processor with a clock speed of 1.2 GHz [45]. This allows the use of Raspberry Pis for complex and computation-intense applications such as video streaming or hosting a simple web server. The choice of programming language is only restricted by the choice of installed operating system as on any other computer system. [42]

An **Arduino** [43] is a microcontroller motherboard, which is able to run one program at a time. There are different models of Arduinos with different form factors and specifications, which determine their performance. The computation power of an Arduino is much lower than what a Raspberry Pi offers. Popular models are the Arduino Uno Revision 3 [46] using an Atmel ATMega328P with a clock speed of 16 MHz and the Arduino Pro Mini [46] with a 8 or 16 MHz processor. The Arduino ecosystem comes with the Arduino Integrated Development Environment (IDE) [47], which lets users easily program their device with the proprietary Sketch [48] programming language. Sketch is a simplified version of C++ and uses similar syntax. Regular C++ can be used on Arduinos by more advanced users.

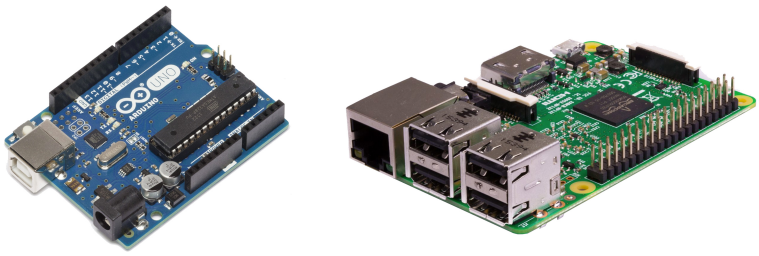

Figure 2.3: Arduino Uno [46] (left) and Raspberry Pi 3 Model B [44] (right)

Chapter 3

Design Decisions

Based on the background research presented in Chapter 2, a suitable communication technology, localization method, and tag hardware has to be chosen for the development of the OTIoT Tag. The taken decisions are justified in this chapter and the intended architecture, which describes how all network, hardware and software components are planned to work together, is presented.

3.1 Taken Decisions

The low energy consumption, long range, and relatively low sensor costs were the main reasons to choose a LPWAN technology for the development of the battery-powered tracking device, the OTIoT Tag. From the two main competitors in the field, **LoRaWAN** was chosen as communication technology, as Sigfox is currently not available in Switzerland [49]. LoRa and LoRaWAN are described in more detail in Section 4.1.
It might be interesting to later combine LoRaWAN with other IoT technologies, available now or in the future, to offer a better coverage and to optimize energy consumption. The IoT development platform FiPy [50], for example, supports five different communication chips (LTE/Cellular, WLAN, Sigfox, LoRaWAN, and BLE). For instance, the LTE module could be used here whenever there is no LoRaWAN coverage available.

Next, a suitable LoRaWAN network provider needs to be selected. **The Things Network** (TTN) was chosen for this role as it is ready to be used and free of charge, offers good outdoor coverage in the city of Zurich and other urban areas of Switzerland, and provides good documentation as well as developer support by means of an active community forum [51] and a Slack channel for Switzerland [52]. More details on TTN can be found in Section 4.2. A map of the TTN coverage in the city of Zurich on August 3, 2017 is shown in Figure 3.1.
Switching to an alternative provider, such as Swisscom LPN, is still an option later on if better coverage or higher quality service is needed and could be done with little effort on the embedded software side. Another argument to decide for TTN over Swisscom LPN is the TTN-related experience gained during the author's participation at the 'MakeZurich' hackathon [53] on February 3-4, 2017, which was hosted by the Open Network Infrastructure Association [54] in collaboration with the the City of Zurich [55].

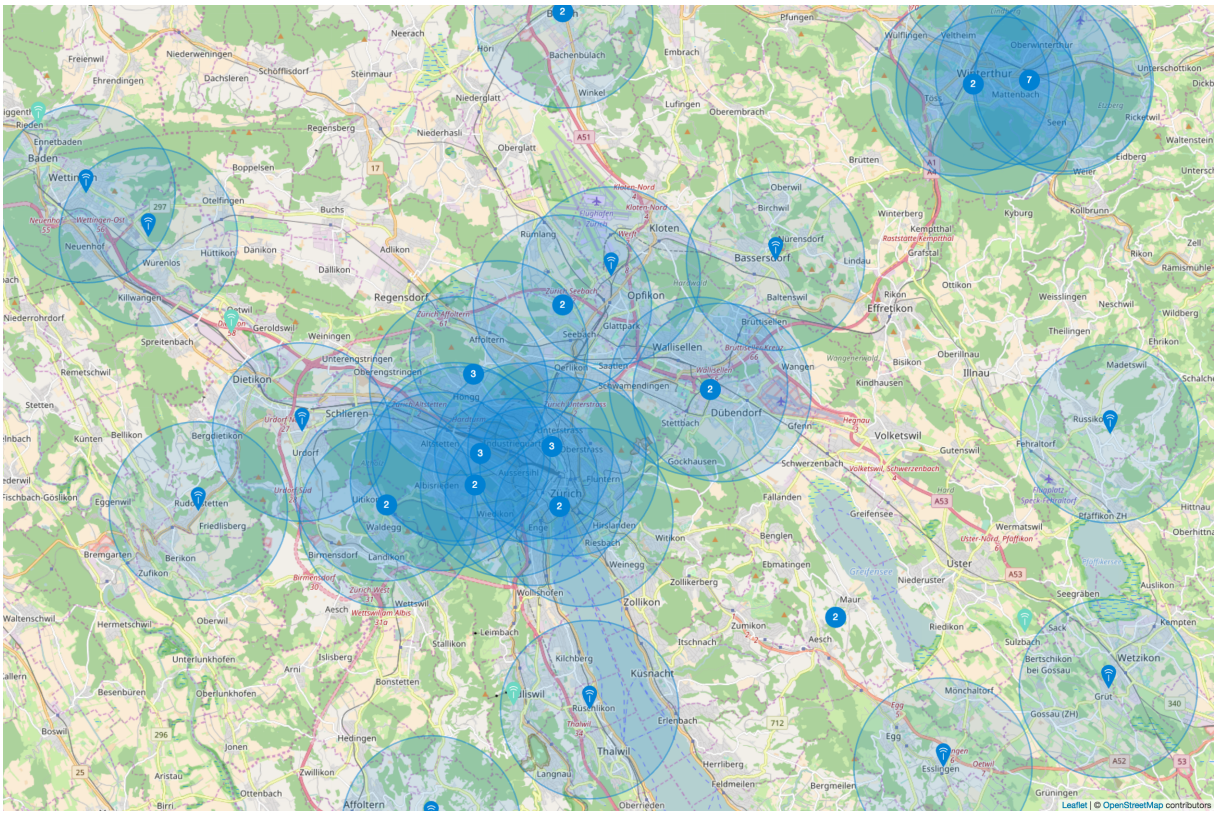

Figure 3.1: TTN's LoRaWAN gateway locations around the city of Zurich [32]

After deciding for a communication technology together with a network provider, the localization method is the next aspect to take a decision for. Since LoRa triangulation is not rolled out yet by neither TTN nor Swisscom LPN as of mid 2017, this method was ruled out early in the process even though first tests were conducted in Neuchâtel, Switzerland in 2016 [56]. Even with the technology available, the predicted accuracies between 20 and 200 m (as presented in a webinar [57] in February 2017) would not be large enough to be suitable for the object tracking use case, as accuracy is a critical factor. Due to the low range of WLAN gateways and a lack of density in more rural areas, WLAN fingerprinting was not deemed a viable option as the primary localization method.
The only remaining examined option is using the **Global Navigation Satellite System**, which offers high accuracy and a wide range of available sensors. Even with the drawbacks of a relatively high energy consumption and a lack of indoor localization capabilities, this method was chosen for the development of the OTIoT Tag due to a lack of high accuracy alternatives. WLAN fingerprinting could, however, later be used as a secondary localization method to decrease energy consumption and improve indoor localization when combined with a GNSS sensor.

Arduino is the tool of choice for building mobile, battery-powered nodes for TTN, which is reflected by the availability of extensive documentation on how to build Arduino-based nodes as well as by the availability of a TTN Software Development Kit (SDK) for Arduino [58]. For LoRa gateways, however, the most common option is to use a Raspberry Pi [59]. The reason being that a LoRa gateway is usually stationary-powered and requires more computational power than a sending node, which is offered by a Raspberry Pi.

Using a Raspberry Pi for a mobile node, however, would drastically shrink its battery life and would be a waste of computational resources for such a simple device.

These taken decisions are reflected in Figure 3.2, which shows how the LoRa modem, the TTN, the GNSS module, and the Arduino motherboard work together to build the OTIoT Tag. An accelerometer is added to reduce the power consumption by only activating the GNSS module when movement of, or tampering with the tag is detected.

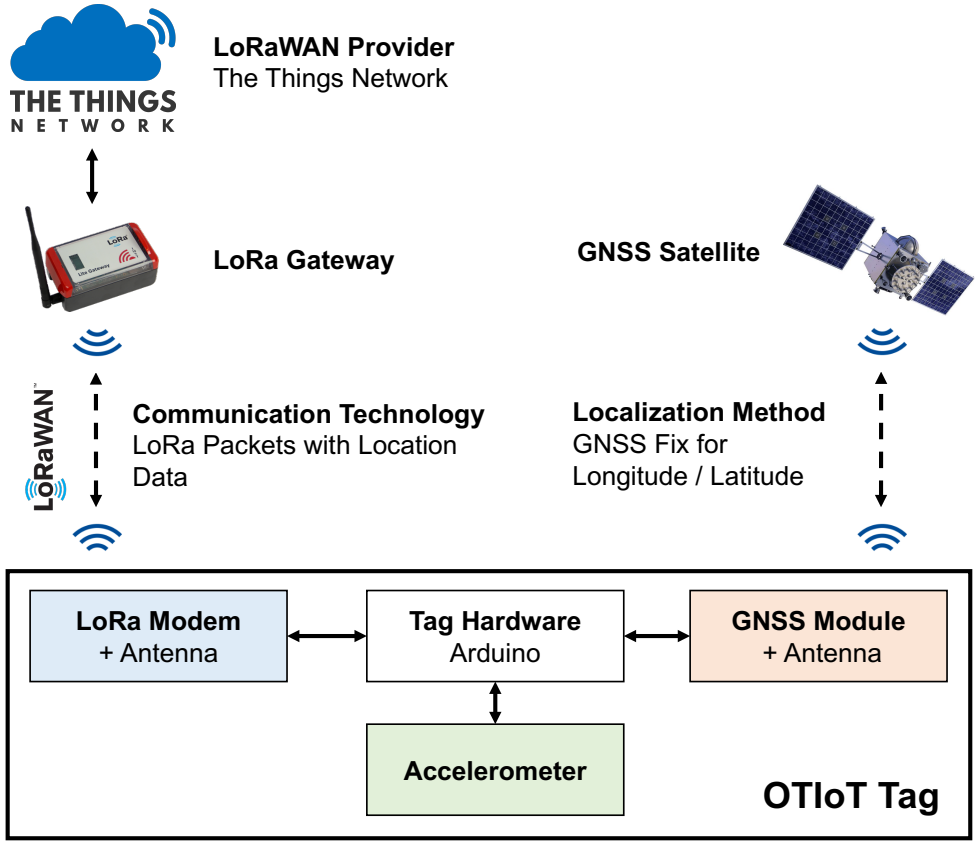

Figure 3.2: Intended schematic hardware layout of the OTIoT Tag

3.2 Intended Architecture and Data Flow

The architecture diagram in Figure 3.3 focuses on the intended data flows between TTN and the OTIoT Platform. The following functional requirements have to be met by the integration architecture: Firstly, uplink location packets need to be sent from the OTIoT Tag via TTN to the OTIoT Back-End in an appropriate format. Furthermore, functionality to send downlink control packets from the OTIoT Back-End via TTN to the OTIoT Tag has to be in place. Lastly, the OTIoT App has to provide the user with means to manage his or her devices.

The data flows are intended to work as follows, with the numbering corresponding to Figure 3.3:

1. The location data is broadcasted by the OTIoT Tag to nearby LoRa gateways via radio link. If a LoRa gateway receives such a LoRa uplink packet, it is forwarded over an Internet backhaul to TTN.
2. The TTN's Hyper Text Transfer Protocol (HTTP) Integration [60] then sends a HTTP POST request with the decrypted and decoded location data to the OTIoT Back-End every time a packet is received.
3. The Data API [61] is intended to be used to send downlink packets to the OTIoT Tag using the Message Queue Telemetry Transport (MQTT) protocol, a connectivity protocol for Machine-to-Machine (M2M)/IoT [62]. This functionality is needed to activate or deactivate a tag, or to set configuration parameters on a tag.
4. The Application Manager API [63] will be used for registering new tags, as well as for deleting and modifying them in the OTIoT TTN Application whenever a user does so in the OTIoT App, syncing the two applications. This ensures that the OTIoT TTN Application is encapsulated from the user and that everything can be controlled via the OTIoT App.

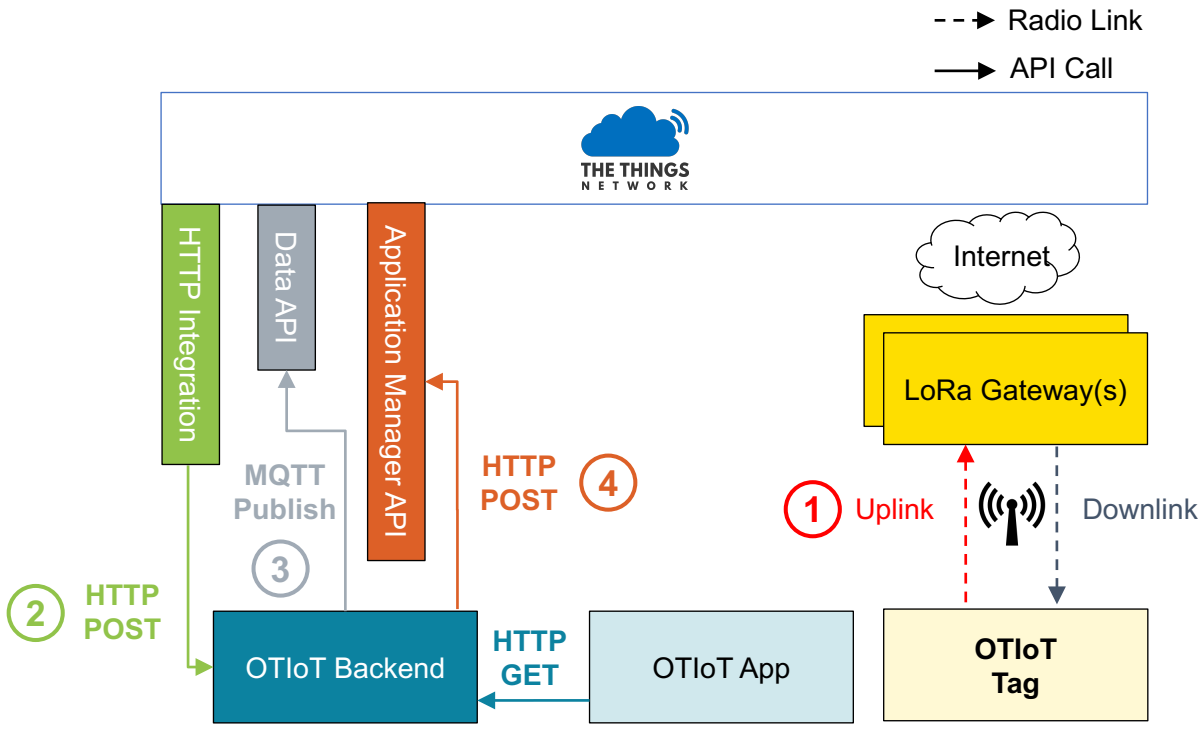

Figure 3.3: Intended architecture and data flows in OTIoT

Details about the TTN APIs used as well as a paragraph about the final architecture can be found in Section 4.6.1.

Chapter 4

Implementation

In this chapter, the different areas involved in the implementation of the data gathering part of the OTIoT Platform are described, assuming the architecture and data flow explained in the previous chapter.
Starting with Section 4.1 about the basics of LoRa and LoRaWAN, the role of TTN in OTIoT is then explained in detail in Section 4.2. The main part of this chapter forms Section 4.3, explaining the functionalities of the OTIoT Tag. An additional tracking node integrated into the OTIoT Platform as well as a self-assembled LoRa gateway are presented in Section 4.4 and Section 4.5, respectively. After that, the integration into the OTIoT Back-End and the OTIoT App is explained in Section 4.6 and Section 4.7.

4.1 LoRa and LoRaWAN

LoRa is a proprietary low power wireless standard, which implements the physical communication layer, corresponding to Layer 1 in the Open Systems Interconnection (OSI) Model. It uses the Industrial, Scientific and Medical (ISM) band between 863 and 870 MHz, and around 433 MHz in Europe, between 902 and 928 MHz in the United States, between 915 and 928 MHz in Australia, and between 779 and 787 MHz as well as between 470 and 810 MHz in China [64]. LoRa modulation is based on chirp spread spectrum technology patented by Semtech Corporation, which is very robust against noise and multipath fading, and enables a longer range at the expense of a lower data rate [64].
For Europe, "LoRaWAN defines ten channels, eight of which are multi data rate channels from 250 bps to 5.5 kbps, a single high data rate LoRa channel at 11 kbps, and a single Gaussian Frequency Shift Keying (GFSK) channel at 50 kbps" [65]. There are three mandatory channels at 868.10, 868.30, and 868.50 MHz for Europe, with each a bandwidth of 125 kHz. For some other European LoRa channels, the bandwidth is 250 kHz [64].

The WLAN bands around 2.4 GHz and 5 GHz are other examples for unlicensed ISM bands. Due to the lower frequency combined with higher ranges, LoRa's ISM frequency band is subject to stricter regulation than the one of WLAN. For Europe, most of LoRa's 125 kHz channels have 1 % or even 0.1 % duty cycles defined in Annex B of the European

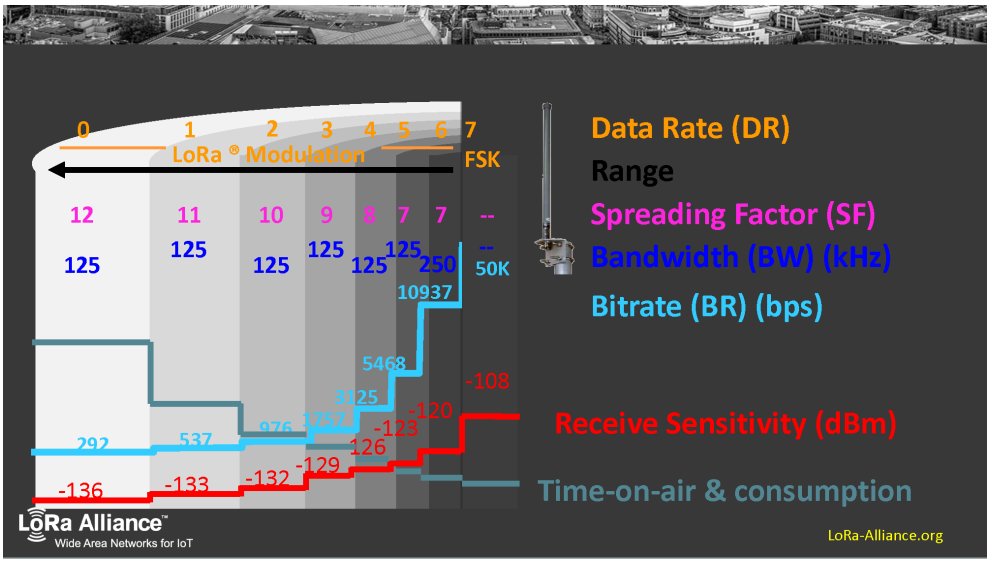

Figure 4.1: Relationship between different LoRa parameters [66]

Telecommunications Standards Institute (ETSI) EN300.220 standard [67]. A 1 % duty cycle means that a device is allowed to occupy a channel only 1 % of the day, resulting in 864 s or 14.4 min of time-on-air per day. The time-on-air of a LoRa packet mainly depends on the payload size and the spreading factor used. At the same time, the spreading factor influences the reachable range of the transmitted packet and its bit rate. Figure 4.1 shows the relationships between data rate, range, spreading factor, bandwidth, bitrate, receive sensitivity and time-on-air. It can be seen that a higher data rate at a lower spreading factor leads to a lower time-on-air, which comes at the cost of a lower receive sensitivity, meaning a lower range of the signal. Accordingly, a user can choose for a LoRa packet to be transmitted over a long time at low speed for a long distance, or for a short time at high speed over a shorter distance. [65]

LoRaWAN builds on top of the physical LoRa layer and offers a Media Access Control (MAC) protocol for low power devices. LoRaWAN networks are typically laid out in a star-of-stars topology, with gateways (in WLAN: access points) relaying messages between end-devices (in WLAN: stations) [68]. The LoRaWAN protocol is defined by the LoRa Alliance [31] in their LoRaWAN Specification [69].

Generally, two types of messages exist: (1) Uplink messages are sent from the end-device to the gateway and (2) downlink messages are sent from the gateway to the end-device [69]. As LoRaWAN is designed for use cases including IoT & M2M, industrial automation, low power applications, battery-operated sensors, smart city, agriculture, metering, and street lighting, the protocol is optimized for many uplink messages and few downlink messages [65]. Uplink messages are used to send sensor information to an application. Downlink messages can be used to send control messages from an application to an end-device, to send MAC commands to it, or to confirm a received uplink, if this was configured by the end-device at the time of transmission. When the ‘Adaptive Data Rate’ feature of LoRaWAN is activated, the network automatically manages the spread factor to optimize for fastest data rate, optimal range, maximum battery life, and maximum network capacity by sending downlink control messages to the end-device [66].

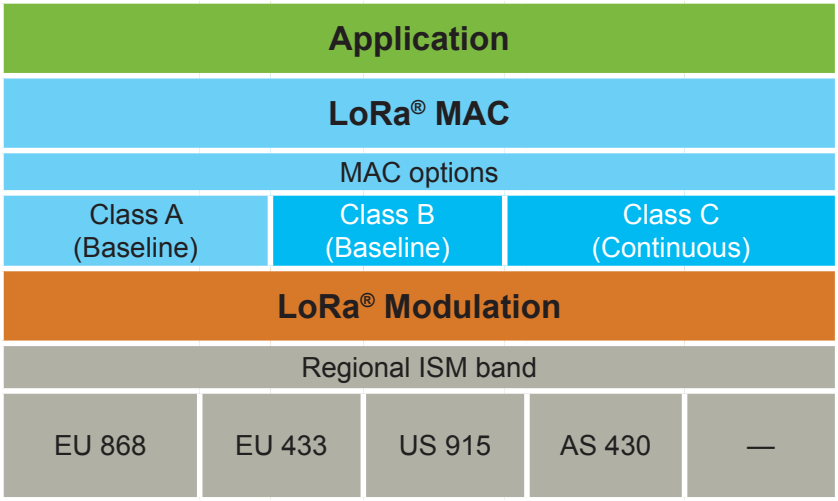

Figure 4.2: LoRaWAN stack [65]

Figure 4.2 shows the official LoRaWAN stack from the LoRa Alliance's technical overview document [65]. The stack's layers are explained in this paragraph.
The 'Regional ISM Bands' are issued by governmental organizations and are listed in the LoRaWAN Regional Parameters document [70]. The first OSI layer, 'LoRa Modulation' uses chirp spread spectrum technology to propagate the signal over long distance. This layer's technology is patented by Semtech Corporation. The LoRaWAN specification [69] defines three different classes of devices: Class A, B, and C devices. Class A ('all') devices offer the most basic functionality of the three types. While uplink messages from the device to the gateway can be transmitted at all times, downlink messages from the gateway to the Class A device can only be received during the two receive windows one and two seconds after sending an uplink message. This is the most power-saving mode for devices. Class B ('beacon') devices extend this functionality by adding additional receive windows in synchronization with the gateways. Class C ('continuous') devices are always able to receive downlink messages except when they are transmitting an uplink message. The continuous listening for incoming downlink messages consumes large amounts of energy and is not suitable for battery-powered end-devices. These devices all use the 'LoRa MAC' protocol, which is often referred to as LoRaWAN. On top of that, applications such as TTN may use the LoRaWAN layer to build their services. The TTN services include device management, payload conversion, and API provisioning.

4.1.1 Security and Device Activation

LoRaWAN's security protocol is based on the IEEE 802.15.4 standard [25], according to the LoRaWAN Specification [69]. It makes use of the Advanced Encryption Standard (AES)-128 as its underlying encryption technology.

Before an end-device is able to communicate using LoRaWAN, it has to be activated. For a device to be activated, the following information is required: Device Address (DevAddr), Network Session Key (NwkSKey), and Application Session Key (AppSKey).
The **DevAddr** is a 32-bit identifier, which is unique within the network. It is present in

each message's data frame to differentiate devices from each other.
The **NwkSKey** is a 128-bit AES encryption key, which is unique per device. It provides communication security as it is used to validate the integrity of each message using the message's Message Integrity Code (MIC). The MIC prevents attackers from intentionally tampering with a message (e.g., changing the payload).
The **AppSKey** is a 128-bit AES encryption key, which is unique per device. The payload of each message is end-to-end encrypted between the application and the end-device (and vice-versa) using the AppSKey.
In addition to these two security keys, the LoRaWAN security protocol makes use of **Frame Counters** to prevent replay attacks. In this type of attack, a previously recorded message is re-transmitted to do harm (e.g., duplicate data or re-execute a command). Messages transmitted to the network are rejected if they have an equal or lower frame counter than the one last seen by the network. [71] [72] [69]

The scope of the two session keys can be seen in Figure 4.3.

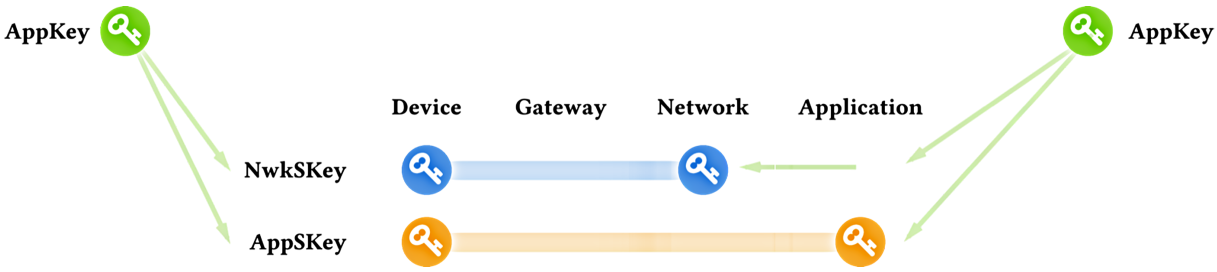

Figure 4.3: Keys and encryption in LoRaWAN [64]

There are two different types of activation possible in LoRaWAN:

Activation by Personalization (ABP) offers the three required keys directly, by hard-coding DevAddr, NwkSKey and AppSKey into the end-device's software. This can be useful for developing as no downlink messages are needed to start sending uplink messages to the network. For production, the use of ABP is discouraged because the device's activation remains unconfirmed. Also, both session keys stay the same over time, which limits the security of the LoRaWAN transmission. [73]

Over-the-Air Activation (OTAA) uses three other, globally unique identifiers to derive the three keys for activation: Device Extended Unique Identifier (DevEUI), Application Extended Unique Identifier (AppEUI), and Application Key (AppKey). In an over-the-air message handshaking process consisting of six steps, the end-device is able to obtain DevAddr, NwkSKey and AppSKey. First, the end-device transmits a 'Join Request' containing DevEUI, AppEUI, and AppKey. The end-device then receives a 'Join Accept' from the network if the credentials are correct. It then authenticates this Join Accept and decrypts it. From the decrypted Join Accept, the DevAddr is extracted and stored. The two security keys – NwkSKey and AppSKey – can then be derived using the AppKey. [66]

4.2 The Things Network

This section gives more details about the LoRaWAN provider TTN and gives insight as to how TTN was used for OTIoT.

The vision of TTN is to offer new data connectivity technologies at a low cost in a crowd-sourced approach. Much like in the early days of traditional Internet in the 1980s, TTN is "created by people that connected their networks to allow traffic from, to and over their servers and cables to pass for free". They use LoRaWAN as their data connectivity technology as it is a "low battery, long range, and low bandwidth" network "without cumbersome WiFi passwords, mobile subscriptions, and zero setup costs". With the long range of every gateway and the lost costs of such a device, a whole city can be covered relatively cheaply. With low costs comes affordability for regular users, who can contribute their gateway's connectivity to the crowd "to enable a network by the users for the users". [32]

Figure 4.4: Gateway locations in TTN's worldwide network [32]

Figure 4.4 shows the TTN global community, consisting of 21 178 users with 1 141 gateways in 89 countries as of August 3, 2017 [32]. Europe is at the center of TTN's network expansion but there are emerging communities in North and South America, Africa, Asia, and Oceania. In Switzerland, nine communities exist as of August 3, 2017: Basel, Bern, Geneva, Lausanne, Luzern, St. Gallen, Winterthur, Zentralschweiz, and Zurich.

4.2.1 Detailed Architecture

TTN's architecture consists of six main components [74]: **Gateways (G)**, **Routers (R)**, **Brokers (B)**, a **Network Server (NS)**, **Handlers (H)**, and **Applications (A)** as shown in Figure 4.5. The architecture is designed for maximum scalability and features a strong separation of concerns between the components.

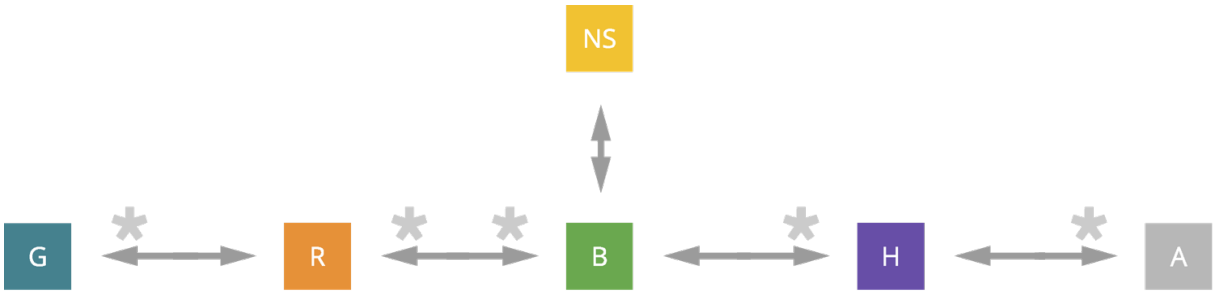

Figure 4.5: TTN high-level core architecture with cardinalities [74]

Arrows show the relationship between the components, where three cases of cardinalities must be distinguished:

1. Relationship 1:1, no star on the arrow (Example: B $\longleftrightarrow$ NS)
2. Relationship 1:n, a star on one side of the arrow (Example: B $\longleftrightarrow^{*}$ H)
3. Relationship n:n, a star on both sides of the arrow (Example: R $^{*}\longleftrightarrow^{*}$ B)

To better understand the role and functionality of each component, the steps taken for receiving an uplink packet (top) and for sending a downlink packet (bottom) are shown in Figure 4.6.

Gateways (G) are responsible for receiving the radio signal transmitted by LoRaWAN end-devices *(Receive)*. Every LoRaWAN packet can be received by multiple gateways, which then encapsulate and forward the received `byte` payload together with metadata such as RSSI and Sound Noise Ratio (SNR) to one specific router over an Internet backhaul *(Forward)*. The router for every gateway can be set in the gateway's configuration. Periodically, the gateway also transmits data about itself such as its GPS coordinates and the number of packets received and transmitted to the router. When a downlink message is forwarded from the router to the gateway, it is responsible to transmit it as a LoRaWAN packet *(Transmit)*. [74]

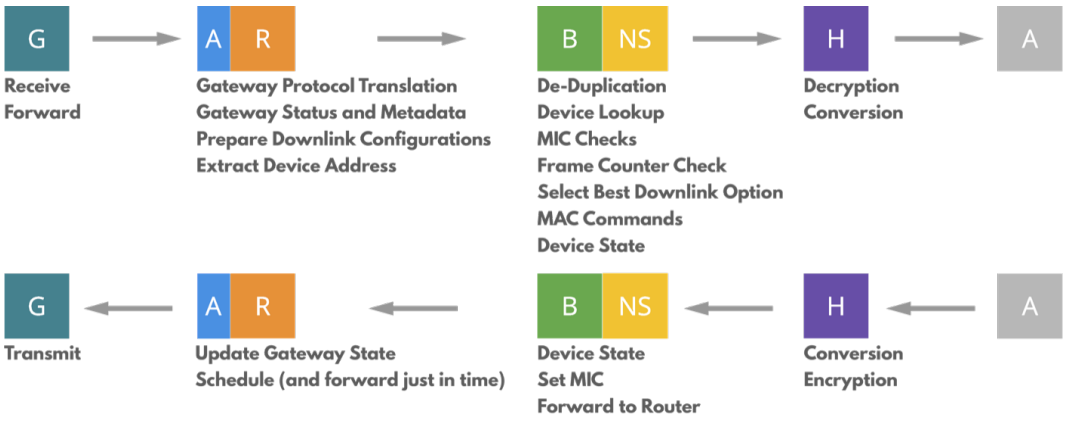

Figure 4.6: Processing flow among TTN's components [74]

The **Router (R)** takes care of all gateway-related functionality and region-specific details. The TTN has developed a number of bridges in the router to translate the vendor-specific gateway protocols into one common format for further processing *(Gateway Protocol Translation)*. Besides translation, the router is also responsible for storing the meta data received by the gateway and for injecting this information into the metadata of each uplink message *(Gateway Status and Metadata)*. For every gateway that received an uplink message, two downlink configurations (one for each receiving window) are calculated by the router *(Preparing Downlink Configurations)*. A scoring algorithm then prepares a score for each downlink configuration according to time-on-air, signal strength, gateway utilization and already scheduled transmissions, while making sure the LoRa duty cycle described in Section 4.1 is enforced. Finally, the router extracts the DevAddr from the uplink message to distribute the traffic to brokers using the device address prefix *(Extract Device Address)*. When a downlink message is sent to the router, it first updates the gateway's state *(Update Gateway State)*. Then, the downlink message is scheduled for the gateway and forwarded at the last moment due to the fact that most gateways only have a downlink buffer of 1 *(Schedule (and forward just in time))*. [74]

The **Brokers (B)** is the main component of the TTN. Each broker is responsible for a range of DevAddrs and forwards their packets to the responsible handler. One of the main tasks of the broker is to combine different received uplink messages into one message, which is done using the payload's Message-Digest Algorithm 5 (md5) sum, the result of a cryptographic hash function. The broker buffers incoming messages for 200 ms, during which the reception-specific meta data of all messages with the same md5 sum are combined *(De-Duplication)*. As the DevAddr is not unique, the device has to be looked up *(Device Lookup)* using cryptographic MIC check *(MIC Checks)*, comparing a list of devices with the same device address using the NwkSKey. By checking the frame counter, the broker then makes sure that the message was not sent in a replay attack. The frame counter is not allowed to be lower than the last known frame count received or be more than 16 384 counts higher than the last known value *(Frame Counter Check)*. Using the pre-calculated scores of the router, the broker can then select the best option for sending a potential downlink message *(Select Best Downlink Option)*. Before forwarding the uplink message to the handler, it is sent to the network server.

The **Network Server (NS)** updates the device's state in its database from the received uplink and downlink messages *(Device State)*. Additionally, the network server creates a downlink template that can later be used by the handler to send a downlink message to the device sending this uplink message. Also, this gives the network server the chance to add MAC commands to the message, such as a command for the device to change its data rate *(MAC Commands)*. It also provisions the list of devices for MIC checks in the broker. For downlink messages, it is responsible to set the message's MIC *(Set MIC)* and to then forward it *(Forward to Router)*. [74]

In the **Handler (H)**, the end-to-end encrypted uplink messages are decrypted using the AppSKey *(Decryption)*. If a user does not want TTN to have access to the decrypted messages, it is possible to host a handler in a private network environment. The decrypted uplink message is then directly forwarded to the TTN **Application (A)** or is further processed by the user-defined Payload Functions, which are described in more detail in Section 4.2.2 *(Conversion)*. The handler also publishes the received uplink messages,

represented as JavaScript Object Notation (JSON), as MQTT topics and as a Google Remote Procedure Call (gRPC) API. After receiving an uplink message, it is the handler's task to decide whether a downlink has to be sent. A downlink is scheduled if (1) the application has a scheduled downlink for the sending device, (2) if the uplink message requires a confirmation, or (3) if the Network Server needs to send MAC commands to the sending device. Before forwarding any downlink message to the broker, it is first processed by Payload Functions *(Conversion)* and encrypted using the AppSKey *(Encryption)*. [74]

4.2.2 Usage of The Things Network in OTIoT

The main role of TTN is the provisioning of the LoRaWAN network infrastructure using the components described in Section 4.2.1. Another role of TTN for OTIoT is offering management and monitoring capabilities to the developer through a user interface called TTN Console. The TTN Console is a web application, which is used to create, manage and delete a registered user's gateways, applications, and devices in TTN. It can also be used to display data received (uplink messages from the device) and sent (downlink messages to the device). For OTIoT, one single application in TTN is used. This also means that all tags use the same AppEUI. An alternative to the TTN Console's Graphical User Interface (GUI) is using the APIs as described in Section 4.6.1, or using the TTN Command Line Interface (CLI) called 'ttncl' [75]. It is the goal of the OTIoT App to encapsulate this functionality from the end-user of the OTIoT Platform.

As of August 3, 2017, the author's local Zurich TTN community [76] consists of 42 gateways (first worldwide) with 90 contributors (second worldwide). The community was founded in October 2015 by Gonzalo Cases, who is still the community leader. Being a very active community, organizing many meetups and even bigger events such as the Make Zurich [53] in February 2017, the Zurich TTN community is rewarded with the label 'Official Community'. The goals needed to be accredited as an official TTN community are listed in [77]. A map of gateway locations in and around the city of Zurich is shown in the previous chapter in Figure 3.1.

Payload Functions in the OTIoT TTN Application

TTN offers conversion of the received `byte` payloads to human-readable text and the other way around. This is achieved using Payload Functions, user-programmable in JavaScript. For uplinks, first a decoder function, then a converter function and lastly a validator function are applied to the received `byte` payload. The behavior of these functions can be differentiated by the incoming port as outlined in Table 4.1. For example, all GNSS location data is sent to port 1 and then processed accordingly, as shown on lines 3 to 14 in Listing 4.1.
For downlinks, an encoder function is used to transform object attributes, scheduled as downlink payload in the TTN Console or received via an API, into a `byte` payload. This `byte` payload has then to be transformed again into meaningful integer, float or string values on the OTIoT Tag as explained in Section 4.3.2.
The used decoder and encoder functions are shown in Listing 4.1 and Listing 4.2.

```
function Decoder(b, port) {
  // Decode uplink messages received as byte payload to object.
  if (port == 1){
    if (b[0] != 0x00 || b[1] != 0x00 || b[2] != 0x00) {
      var lat = ((b[0] & 0x80 ? 0xFFFF << 24 : 0) | b[0] << 16 | b[1]
    << 8 | b[2]) / 10000;
    }
    if (b[3] != 0x00 || b[4] != 0x00 || b[5] != 0x00) {
      var lng = ((b[3] & 0x80 ? 0xFFFF << 24 : 0) | b[3] << 16 | b[4]
    << {}8 | b[5]) / 10000;
    }
    return {
      lat: lat,
      lng: lng
    };
  }
  else if (port == 2){
    return {}; // don't send polling over HTTP Integration
  }
  else if (port == 3){
    return {
      joined: true
    };
  }
  else { // Cayenne LPP: https://github.com/myDevicesIoT/cayenne-docs
   /blob/master/docs/LORA.md
    if (b[0] == 0x01 && b[1] == 0x88) { // Channel: 0x01, Source: GPS
    (0x88)
      if (b[2] != 0x00 || b[3] != 0x00 || b[4] != 0x00) {
        var lat = ((b[2] & 0x80 ? 0xFFFF << 24 : 0) | b[2] << 16 | b
   [3] << 8 | b[4]) / 10000;
      }
      if (b[5] != 0x00 || b[6] != 0x00 || b[7] != 0x00) {
        var lng = ((b[5] & 0x80 ? 0xFFFF << 24 : 0) | b[5] << 16 | b
   [6] << 8 | b[7]) / 10000;
      }
      if (b[8] != 0x00 || b[9] != 0x00 || b[10] != 0x00) {
        var alt = (b[10] | b[9] << 8 | b[8] << 16) / 100;
      }
    }
    if (b[11] == 0x02 && b[12] == 0x02) { // Channel: 0x02, Source:
   Analog Input (0x02)
      if (b[13] != 0x00 || b[14] != 0x00) {
        var battery = (b[14] | b[13] << 8);
      }
    }
    return {
      lat: lat,
      lng: lng,
      alt: alt,
      battery: battery
    };
}}
```

Listing 4.1: Decoder function in the OTIoT TTN Application

The decoder function processing the `byte` payload sent by the OTIoT Tag in Listing 4.1 differentiates between four different port ranges. On port 1, the location information is sent as an integer value with an accuracy of four decimal places. The maximum value is $2^{3*8} = 16\,777\,216$, or a range from $-8\,388\,608$ to $8\,388\,607$, which is enough to map latitude values from $-90.000\,0$ to $90.000\,0$ decimal degrees and longitude values from $-180.000\,0$ to $180.000\,0$ decimal degrees. Therefore, the 3 Bytes for each latitude and longitude have to be combined using bit shift operators and are then divided by 10 000 to reconstruct the original values (lines 4 to 9). On port 2, only polling messages are received, which do not have to be handled by the OTIoT TTN Application. To let the OTIoT Back-End know about new devices, they can send packets on port 3 after joining. On ports greater or equal than 4, the Cayenne Low Power Payload (Cayenne LPP) decoding is implemented, which works similar to the regular location data processing, but additionally offers decoding for altitude (with two decimal places) and battery level (1 to 100). This decoder function in TTN corresponds to the encoding function in the OTIoT Tag in Listing 4.3.

```
function Encoder(object, port) {
  // Encode downlink messages sent as object to an array of bytes.
  var bytes = [];

  if (port === 10) {
    bytes[0] = object.armed ? 1 : 0;
    bytes[1] = object.acc_sensitivity;

    var gps = object.gps_tolerance * 1000000;
    for (var index = 2; index <= 5; index ++) {
        var byte = gps & 0xff;
        bytes[index] = byte;
        gps = (gps - byte) / 256 ;
    }

    var moving_send_delay = object.moving_send_delay;
    for (var index = 6; index <= 9; index ++) {
        var byte = moving_send_delay & 0xff;
        bytes[index] = byte;
        moving_send_delay = (moving_send_delay - byte) / 256 ;
    }
  }

  return bytes;
}
```

Listing 4.2: Encoder function in the OTIoT TTN Application

In the encoder function, which is shown in Listing 4.2, the downlink message `byte` payload for the OTIoT Tag is prepared to be sent on port 10. Four different configuration parameters are transmitted: `armed`, `acc_sensitivity`, `gps_tolerance`, and `moving_send_delay`. The armed state is a simple boolean, which can be transferred as a 8-bit `byte` of 0x00 or 0x01 (hexadecimal representation). The accelerometer sensitivity can range from 0 to 255 and is therefore encoded in the second `byte` with a

capacity of $2^8 = 256$. The GPS tolerance is a strictly positive value and accurate up to six decimal degrees, which fits into 4 Bytes of payload, offering a maximum value of $2^{(4*8)} = 4\,294\,967\,295$. This is enough to map latitude values up to 180.000 000 and longitude up 360.000 000 decimal degrees, with a theoretically maximum value of 4 294.967 295 decimal degrees. A 24-bit integer would only offer a maximum of 16.777 216 decimal degrees. The moving send delay is also encoded using 3 Bytes, offering a maximum value of $2^{(4*8)} = 4\,294\,967\,295$ s, which is enough to configure a send delay of more than one year. The encoder function in TTN corresponds to the decoding function in the OTIoT Tag in Listing 4.4.

For reference, the five port ranges used (four for uplink and one for downlink) in the OTIoT TTN Application's Payload Functions are listed in Table 4.1.

Table 4.1: LoRa ports used in the OTIoT TTN Application

Description	**Port Number**	**Type**
Location Data	1	Uplink
Polling	2	Uplink
Join Message	3	Uplink
Cayenne LPP	≥ 4	Uplink
Configuration	10	Downlink

The TTN Console offers the usage of pre-defined Payload Functions for Cayenne LPP. Cayenne LPP "provides a convenient and easy way to send data over LPWAN networks such as LoRaWAN. The Cayenne LPP is compliant with the payload size restriction, which can be lowered down to 11 Bytes, and allows the device to send multiple sensor data at one time" [78]. Each sensor data must be prefixed with 2 Bytes: one for 'Data Channel' and one for 'Data Type'. The data channel uniquely identifies each sensor in the device across frames (e.g., 'indoor sensor'), while the data type identifies the data type in the frame (e.g., 'temperature'). The identifiers for the data types conform to the IPSO Alliance Smart Objects Guidelines [79], which identifies each data type using an 'Object ID'. For each payload, a number of channels, which hold different kinds of data (e.g., one for each location, temperature,and humidity), is defined.
Cayenne LPP's benefit is the adaptability to different types of payload formats. If for every transmission, it is clear what type of payload is transmitted, 4 Bytes for the data channel and the data type are wasted with every LoRa packet. For the OTIoT Tag, the location data transmitted on port 1 always consists of 3 Bytes for latitude and 3 Bytes for longitude, and therefore using Cayenne LPP would add an unnecessary overhead.

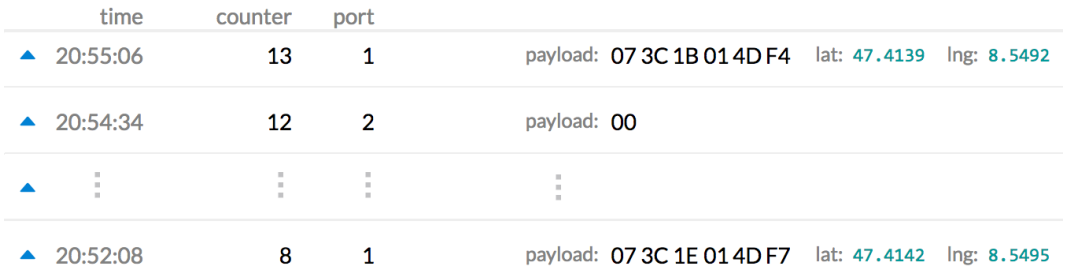

Figure 4.7: Location data sent by the OTIoT Tag, shown in the TTN Console

As shown in Figure 4.7, data can be displayed in both `byte` and human-readable format in real-time in the TTN Console [80]. In addition to the payload, the time of arrival at the TTN handler, the frame counter, the receiving port, the device ID as well as meta data such as the used spreading factor, the receiving gateways and their location, and more technical details can be displayed in the TTN Console.

4.2.3 Limitations of LoRaWAN and The Things Network

While an unlimited number of applications with each an unlimited number of devices is allowed in TTN, strict regulations for the time-on-air usage exist. This so called 'Fair Access Policy' is stricter than the 1 % LoRa duty cycle presented in Section 4.1: It only allows 30 s of uplink time-on-air and only ten downlink messages per day [81]. As of mid 2017, the Fair Access Policy is not enforced by any means and remains a Gentleman's Agreement [82] within the TTN community.

These limitations give the developer an incentive to minimize the payload size of each LoRa uplink packet. In the case of the OTIoT Tag, omitting the send timestamp from the payload saves payload size. This is essential to be allowed to send as many messages as possible, in accordance with TTN's Fair Access Policy. Another way to decrease the payload size is to send the location information (in decimal degrees, coming from the GNSS sensor) in a binary format. A precision of four decimal places was deemed reasonable for the presented object tracking use case. With four decimal places, a maximum accuracy of 11.132 m can be represented. Four decimal places are equal to 3 Bytes of payload, resulting in a payload size of 6 Bytes for both coordinates. At a medium spreading factor of nine (SF9) and adding the LoRaWAN header of 13 Bytes [83], this results in 186 ms time-on-air per LoRa uplink packet according to the LoRa Modem Design Guide [84] and the spreadsheet [85] presented in the TTN forum. Respecting the TTN Fair Access Policy, this leads to a maximum of 161 uplink packets allowed to be sent per day or a packet being sent every 537 s (approx. every 9 min) when sending locations continuously. When only respecting the 1 % duty cycle, a LoRa uplink packet could be sent every 19 s with a SF9. The detailed implementation of the encoding and decoding functions in the OTIoT Tag can be found in Section 4.3.2.

4.3 OTIoT Tag

This section gives an overview on what hardware was used for the OTIoT Tag and what features the embedded software offers. The following features are to be developed:

- Send encoded location information as LoRaWAN uplink packets when the tagged object is moved (only use GNSS if the tag is suspected to be moving).
- Send (empty) polling messages as LoRaWAN uplink packets in order to be able to receive configuration as LoRaWAN downlink packets.
- Send a 'joined' message as LoRaWAN uplink packet to be able to receive a configuration downlink packet on startup.
- Receive and decode configuration sent as LoRaWAN downlink packets in order to configure the behavior of the OTIoT Tag.

4.3.1 OTIoT Tag Hardware

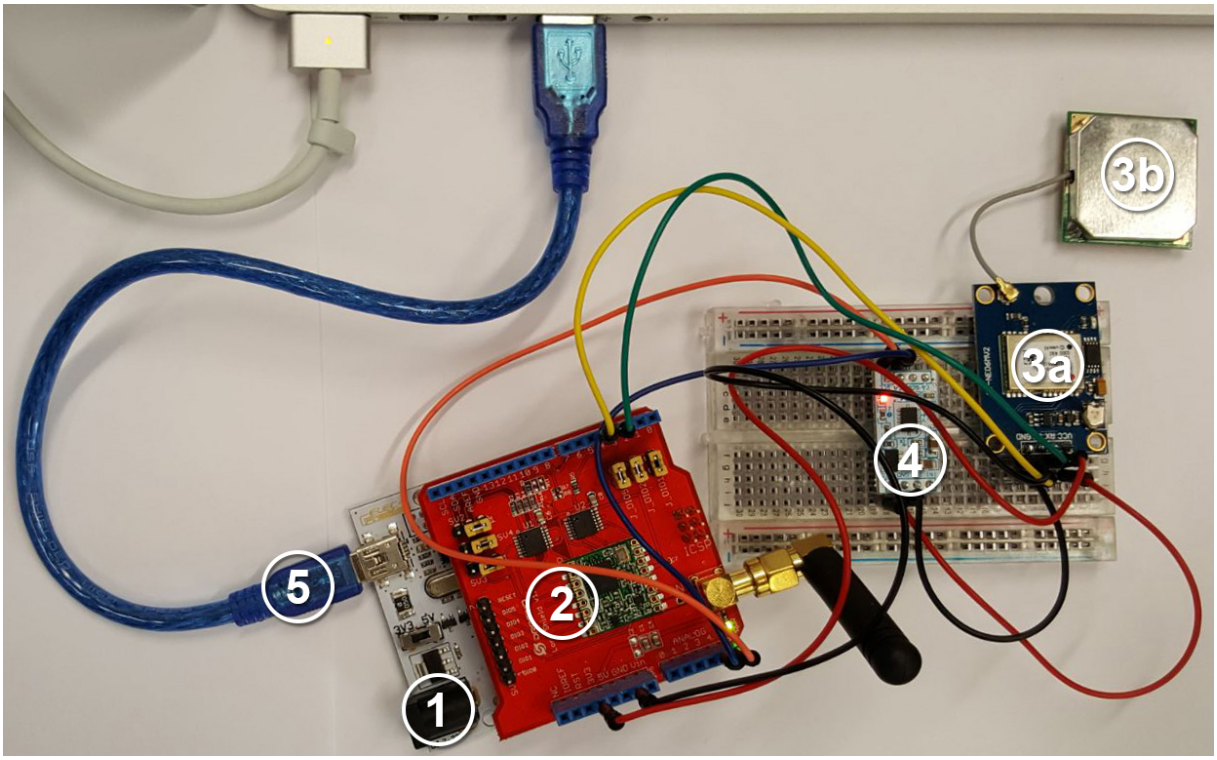

Figure 4.8: OTIoT Tag in unpackaged form

The OTIoT Tag setup is based on Thomas Amberg's beginner-friendly tutorial for LoRaWAN and TTN [86].
The main component is a **(1) Freeduino Uno Rev1.8** [87], which is an Arduino Uno-compatible microcontroller.

As LoRa communication module, a **(2) Dragino LoRa Shield v1.4** [88] is used, which features a HopeRF RFM95W 868/915Mhz RF Transceiver Module [89] – a LoRa modem compatible with the Semtech SX1276 chip [90]. The shield is directly attached to the Freeduino Uno's female header pins. More details on the Dragino LoRa Shield can be found in the Dragino Wiki [91].
To locate the OTIoT Tag, the **(3a) GY-GPS6MV2** breakout board, which gives access to a NEO-6M GPS module [92] by Swiss chip manufacturer u-blox, is connected. Attached to the GPS module is a **(3b) passive porcelain antenna**. The Freeduino Uno connects to the GPS module using two jumper wires for RX (receive) on pin 3 and TX (transmit) on pin 4 and one jumper wire for each 3.3 V current and ground.
An **(4) MMA7455 accelerometer** chip is used for motion detection in the OTIoT Tag. It also has two jumper wires attached for data transmission (SDA on analog pin 4 / SCL on analog pin 5) and one jumper wire for each 3.3 V current and ground. Together with the GPS module, it is attached to the bread board for easier connectivity without soldering.
To supply the OTIoT Tag with power, a laptop or a simple power bank (not in figure) can be used. The power source is connected to the Freeduino Uno's **(5) Mini-USB port**.

Figure 4.8 shows a picture of the assembled but unpackaged OTIoT Tag. A packaged version is shown later in Figure 5.1. A list of the used components of the OTIoT Tag, based on the tutorial [86], is shown in Table 4.2.

Table 4.2: Hardware components of the OTIoT Tag

Component	**Supplier**	**Price**
Freeduino Uno Rev1.8	BOXTEC AG [93]	CHF 23.70
Dragino LoRa Shield - 868MHz v1.4	BLUEBERRY E [94]	CHF 38.45
GY-GPS6MV2 GPS Module + Antenna	BOXTEC AG [95]	CHF 21.80
MMA7455 Accelerometer	Pusterla Elektronik AG (similar to [96])	CHF 26.15
Power Bank	Giveaway (similar to [97])	CHF 15.80
Breadboard Clear - 8.2*5.3cm	BOXTEC AG [98]	CHF 6.20
Breadboard Jumper Wire m-m	BOXTEC AG [99]	CHF 6.80
Total Costs		**CHF 138.90**

4.3.2 OTIoT Tag Software

This section focuses on the implementation details of the embedded software, which runs on the Freeduino Uno. It was programmed in the Sketch language [48] using the Arduino IDE [47].

Arduino Libraries and LoRa Script Template

The software running on the OTIoT Tag is based on a script template by Thomas Telkamp and Matthijs Kooijman [100], which uses OTAA as explained in Section 4.1.1. In the background, it uses the **Arduino-LMIC** library [101] based on the LoraMAC-in-C (LMIC) library, which was open sourced by IBM in 2015 [102]. To control and transform the National Marine Electronics Association (NMEA) 0183 [103] data stream sent by the GPS module into useful variables, the **TinyGPS++** library [104] for Arduino is included. For accessing the x, y, and z axis value of the accelerometer chip, the **MMA7455** library [105] was integrated into the Sketch file. Extensive serial text output is used for debugging purposes in the OTIoT Tag's software. The output can be displayed in the serial monitor of the Arduino IDE using a Baud rate of 57 600.

Location Sending Algorithm and LoRa Communication

With the script template and the libraries in place, the development of the tag's software, with the 'Location Sending Algorithm' at its core, is started. The main goal of this algorithm is to access the GPS module as little as possible and to minimize the LoRaWAN network usage while preserving the OTIoT Tag's capability to report its current location if needed.

The steps of the location sending algorithm can be seen in the flow chart in Figure 4.9 and are explained here in detail, with the required configuration parameters from Table 4.3 in parentheses:

On startup, the Baud rate is configured and the OTIoT Tag immediately tries to join TTN using the hard-coded DevEUI, AppEUI, and AppKey (`DEVEUI, APPEUI, APPKEY`). If no LoRa Join Accept downlink packet is received, a retry is started after some time (`LORA_JOIN_INTERVAL`). After the OTIoT Tag has successfully joined the TTN, the accelerometer is calibrated and the values are saved for later comparison. Also, the GPS sensor tries to obtain the current position, using a certain minimum and maximum amount of retires (`GPS_RETRIES_MIN and GPS_RETRIES_MAX`) and a delay between each retry (`GPS_RETRY_INTERVAL`). The obtained location is then saved in order to be able to compare it later. After that, a LoRa joined uplink packet with empty payload is sent (`LORA_PORT_JOINED`). This makes the OTIoT Back-End schedule a LoRa configuration downlink packet. Besides that, the current location is sent as a LoRa uplink packet on startup (`LORA_PORT_LOCATION`).
When a LoRa downlink packet with configuration information (`ARMED, ACCELEROMETER_SENSITIVITY, GPS_TOLERANCE, MOVING_SEND_DELAY`) is received (`LORA_PORT_CONFIG`), the OTIoT Tag updates its configuration. According to if the tag is then armed or not, different loops are started.

In **'Disarmed'** state, only LoRa uplink poll packets are periodically sent (`LORA_PORT_POLL, LORA_POLL_INTERVAL`) to enable potential downlink packet reception (e.g., activation). When a tag is in disarmed mode, no location data is sent.

In **'Armed'** state, the device periodically (`ACCELEROMETER_CHECK_DELAY`) compares the current accelerometer values with the values saved earlier. If the current x, y, or z value is further away from the saved value than the configured accelerometer sensitivity threshold (`ACCELEROMETER_SENSITIVITY`), the GPS module is activated. The GPS module then tries to obtain the current location for comparison with the last saved location of the device. If the current location is outside of the GPS tolerance (`GPS_TOLERANCE`), 'Moving' (`MOVING`) is set to `true` and the location of the OTIoT Tag is sent as a LoRa uplink packet (`LORA_PORT_LOCATION`).

In **'Moving'** state, the current GPS position is reported to the OTIoT Back-End in regular intervals (`MOVING_SEND_DELAY`) until the tag is found to rest within the specified GPS tolerance. In that case, the OTIoT Tag is set to a non-moving state (`MOVING`) and starts checking the accelerometer again.

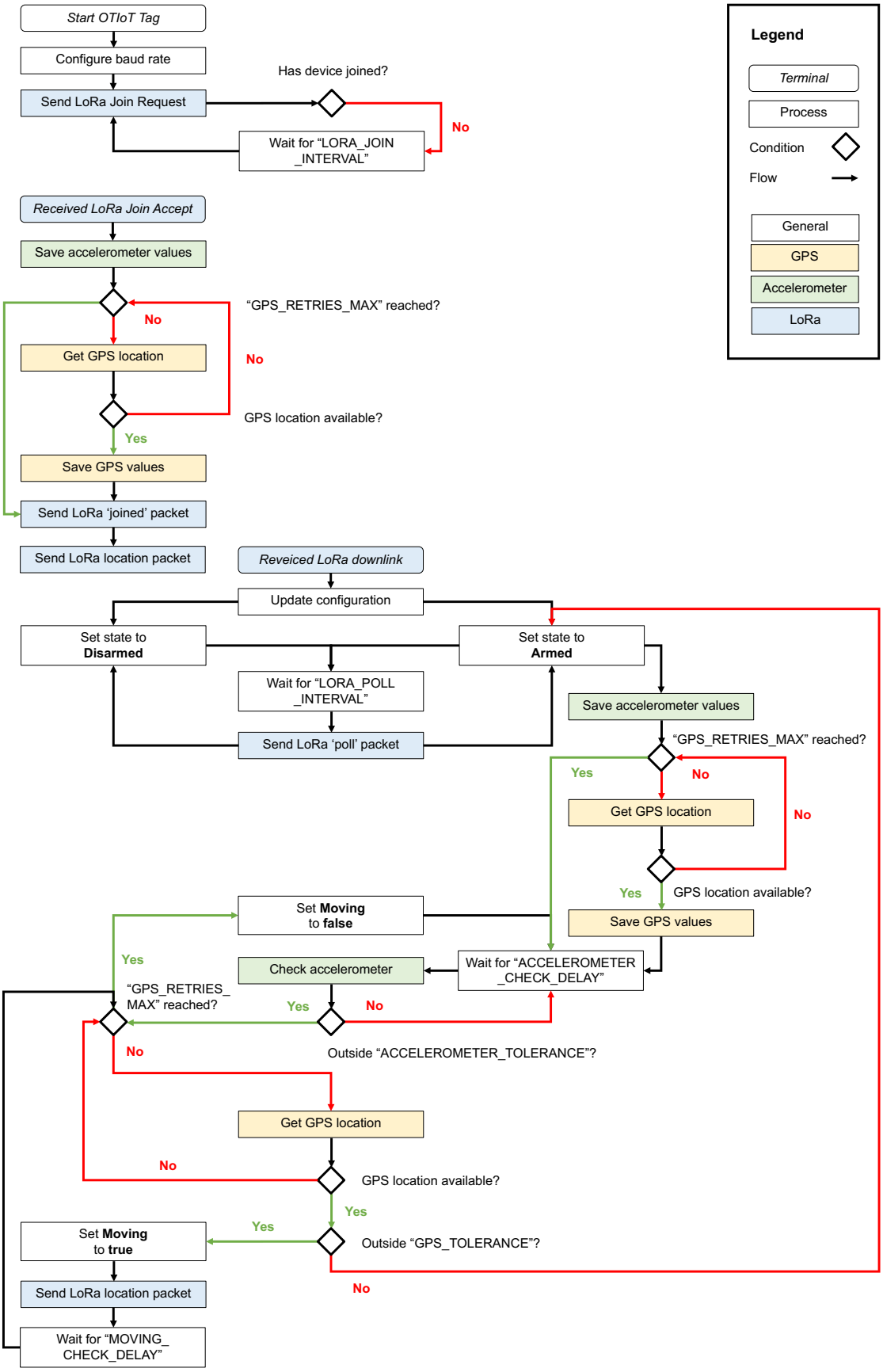

Figure 4.9: Flow chart of the location sending algorithm on the OTIoT Tag

Payload Encoding and Decoding in the OTIoT Tag

The encoding of the location data as `byte` payload before sending it as a LoRaWAN uplink packet is presented in Listing 4.3. As can be seen, the GPS location is multiplied by 10^4 `(GPS_ACCURACY)` to get two integer values, which are then padded into 3 Bytes each (lines 14 to 19) before being sent as LoRa uplink packet.
The OTIoT Tag can switch to using Cayenne LPP by setting `LORA_CAYENNE_LPP` to `true`, which then uses `LORA_PORT_LOCATION_LPP` for sending location uplink packets. Also, `GPS_DEBUG` can be set to `true` if a debug location should be sent in case there is no GPS reception available (e.g., indoors). If confirmed uplink packets should be sent, `LORA_CONFIRMED_UPLINK` hast to be set to `true`.

```
1  void loraGPSSend(float gpsLat, float gpsLng, float gpsAlt) {
     int32_t lat = gpsLat * pow(10, GPS_ACCURACY);
3    int32_t lng = gpsLng * pow(10, GPS_ACCURACY);
     int32_t alt = gpsAlt * 100;
5    byte coordsSize = 0;
     if (LORA_CAYENNE_LPP) { coordsSize = 11; } //2 LPP, 3x lat, lng,
      alt
7    else { coordsSize = 6; } //3 lat, 3 lng
     byte coords[coordsSize];
9    byte cur = 0;
     if (LORA_CAYENNE_LPP) {
11     coords[cur++] = 0x01;
       coords[cur++] = 0x88;
13   }
     coords[cur++] = (lat) >> 16;
15   coords[cur++] = (lat) >> 8;
     coords[cur++] = (lat);
17   coords[cur++] = (lng) >> 16;
     coords[cur++] = (lng) >> 8;
19   coords[cur++] = (lng);
     if (LORA_CAYENNE_LPP) {
21     coords[cur++] = (alt) >> 16;
       coords[cur++] = (alt) >> 8;
23     coords[cur++] = (alt);
       loraSend(coords, sizeof(coords), LORA_PORT_LOCATION_LPP);
25   } else {
       loraSend(coords, sizeof(coords), LORA_PORT_LOCATION);
27   }
     Serial.print(F("LoRa: Packet queued (lat: "));
29   Serial.print(lat / pow(10, GPS_ACCURACY), GPS_ACCURACY);
     Serial.print(F(", lng: "));
31   Serial.print(lng / pow(10, GPS_ACCURACY), GPS_ACCURACY);
     Serial.println(F(")"));
33   os_setTimedCallback(&polljob, os_getTime() + sec2osticks(
      LORA_POLL_INTERVAL), loraPoll); //set new timer for polling
      packet not to obstruct sending
}
```

Listing 4.3: OTIoT Tag's encoding into an uplink message payload of 6 Bytes

When receiving a LoRaWAN downlink packet, the `byte` payload has to be decoded before adapting the OTIoT's configuration as shown in Listing 4.4. The first `byte` determines the armed state, the second one the accelerometer sensitivity. Bytes 3 to 6 and 7 to 10 are bit-shifted into two unsigned 32-bit integers (`uint32_t`) for GPS tolerance and moving send delay. After the configuration is processed and the armed state has been set to `true`, the accelerometer and the GPS module are re-initialized, and the detection loop is started.

```
void loraReceive() {
  byte rxPort = LMIC.frame[LMIC.dataBeg - 1];
  Serial.print(F("Port (receiving): "));
  Serial.println(rxPort);
  if (rxPort == LORA_PORT_CONFIG) {
    Serial.print(F("Size: "));
    Serial.println(sizeof(LMIC.frame));
    armed = LMIC.frame[LMIC.dataBeg];
    Serial.print(F("=> Status: "));
    Serial.println(armed ? "ARMED" : "DISARMED");
    accSensitivity = LMIC.frame[LMIC.dataBeg + 1];
    Serial.print(F("=> Accelerometer Sensitivity: "));
    Serial.println(accSensitivity);

    //Conversion to uint32_t necessary for the bit-shift -> otherwise
     overflow
    uint32_t gpsToleranceRaw = (LMIC.frame[LMIC.dataBeg + 2] | LMIC.
   frame[LMIC.dataBeg + 3] << 8 |  (uint32_t) LMIC.frame[LMIC.
   dataBeg + 4] << 16 |  (uint32_t) LMIC.frame[LMIC.dataBeg + 5] <<
   24);
    gpsTolerance = gpsToleranceRaw / pow(10, GPS_ACCURACY + 2);
    Serial.print(F("=> GPS Tolerance: "));
    Serial.println(gpsTolerance, GPS_ACCURACY + 2);

    movingSendDelay = (LMIC.frame[LMIC.dataBeg + 6] | LMIC.frame[LMIC
   .dataBeg + 7] << 8 | (uint32_t) LMIC.frame[LMIC.dataBeg + 8] <<
   16 | (uint32_t) LMIC.frame[LMIC.dataBeg + 9] << 24);
    Serial.print(F("=> Moving Check Delay: "));
    Serial.println(movingSendDelay);

    if (armed) {
      accInit();
      gpsInit(&gpsjob);
      detectionLoop(&detjob);
    }
  }
}
```

Listing 4.4: OTIoT Tag's decoding of the downlink message's payload of 10 Bytes

Configuration

As described in Section 4.1, LoRa Class A devices are only able to receive downlink packets directly after having sent an uplink packet. As the OTIoT Tag should be, to some extent, remotely user-configurable, downlink packets have to be receivable at some point. This makes it necessary to implement a polling procedure that sends an empty uplink packet (no payload) in a pre-defined interval (`LORA_POLL_INTERVAL`). In accordance with the TTN Fair Access Policy, only ten downlink messages are allowed to be received per device and day. This limit is currently not controlled in this software implementation.

Table 4.3: Configuration parameters of the OTIoT Tag

Name	**Default Value**	**Remote Configurable**
`APPEUI`	None (LSB format)	No
`DEVEUI`	None (LSB format)	No
`APPKEY`	None (MSB format)	No
`LORA_PORT_LOCATION`	1	No
`LORA_PORT_LOCATION_LPP`	4	No
`LORA_PORT_POLL`	2	No
`LORA_PORT_JOINED`	3	No
`LORA_PORT_CONFIG`	10	No
`LORA_CAYENNE_LPP`	True	No
`LORA_JOIN_INTERVAL`	30 (s)	No
`LORA_POLL_INTERVAL`	300 (s)	No
`LORA_CONFIRMED_UPLINK`	0	No
`GPS_DEBUG`	False	No
`GPS_TOLERANCE`	**0.0003** (degrees)	**Yes**
`GPS_RETRIES_MIN`	3	No
`GPS_RETRIES_MAX`	10	No
`GPS_RETRY_INTERVAL`	1000 (ms)	No
`GPS_ACCURACY`	4	No
`MOVING_SEND_DELAY`	**300** (sec)	**Yes**
`ACCELEROMETER_CHECK_DELAY`	10 (s)	No
`ACCELEROMETER_SENSITIVITY`	**15** (of 255)	**Yes**
`ARMED`	**False**	**Yes**
`MOVING`	False	No

In Table 4.3, the available configuration parameters of the OTIoT Tag are listed along with their default values. An indicator if the value is remotely configurable, meaning by use of a LoRa downlink packet, is also included in the table. The remotely configurable parameters are displayed in bold font. The usage of these parameters can be seen in Figure 4.9.

The default value for GPS tolerance in meters is calculated using the earth's circumference divided by 360°: $0.000\,3 * (40\,075\,161.2\,\text{m} \div 360\,\text{degrees}) = 33.396\,\text{m}$ at the equator. The ports used in the OTIoT Tag's software have to match the ports in the decoder and encoder function in the OTIoT TTN Application shown in Section 4.2.2 in order for the OTIoT Platform to work correctly.

Deployment

In order to deploy and authenticate a tag, its software has to be configured according to the device's registration in TTN. For the OTIoT Tag, OTAA was selected as device activation method. The benefits of this type of activation are explained in Section 4.1.1.

The first step to deploy a new tag is to register the tag within the OTIoT App. The DevEUI, AppEUI, and AppKey can then be read from the OTIoT Tag's detail view in the OTIoT App. While DevEUI and AppEUI have to be converted from hexadecimal string format to Least Significant Bit (LSB) `byte` array format, the AppKey has to be converted from hexadecimal string format to Most Significant Bit (MSB) `byte` array format. These formatted EUIs and keys can also be directly copied from the TTN Console. These three values are set as constants in the Arduino Sketch source code in a manual process. The software is then ready to be built and uploaded to the OTIoT Tag using the Freeduino Uno's Mini-Universial Serial Bus (USB) cable. A step-by-step manual of this deployment process can be found in Appendix A.

In contrast to other LoRa modems, the HopeRF RFM95W 868/915MHz RF Transceiver Module [89] of the OTIoT Tag does not offer a built-in DevEUI, which could be used instead of the randomly generated one. As LoRaWAN does not allow communication with non-activated devices for security reasons, there is currently no automatic device activation method available. Refer to Section 4.4 for a deployment method using a CLI over USB to configure the device for deployment.

4.4 FMLR TrackerTwo

In order to demonstrate the simple integration of additional kinds of tracking devices, the author researched for commercial tracking devices in June 2017. Miromico AG[106], a Zurich-based integrated circuit and electronic engineering company, offers the 'FMLR TrackerTwo LoRaWAN GPS Tracker Kit' [107]. It features a multi-band GNSS receiver for GPS, GLONASS, Galileo, BeiDou, and an accelerometer for movement and tampering detection. The application areas are described as "accurate location tracking of humans,

animals and goods" and "detection of tempering and theft" [107]. This set of features makes this tracking device well-suited for a comparison with the OTIoT Tag.
This LoRa end-device is delivered in unassembled form: Before being able to use it, the header pins and the battery case have to be soldered to the Printed Circuit Board (PCB) board. During the ordering process, a tabular file in Comma-separated Values (CSV) format with 'Commissioning Data' is requested in order to specify the DevEUI, AppEUI, the AppKey, the reporting cycle, and the activation method (OTAA in this case).

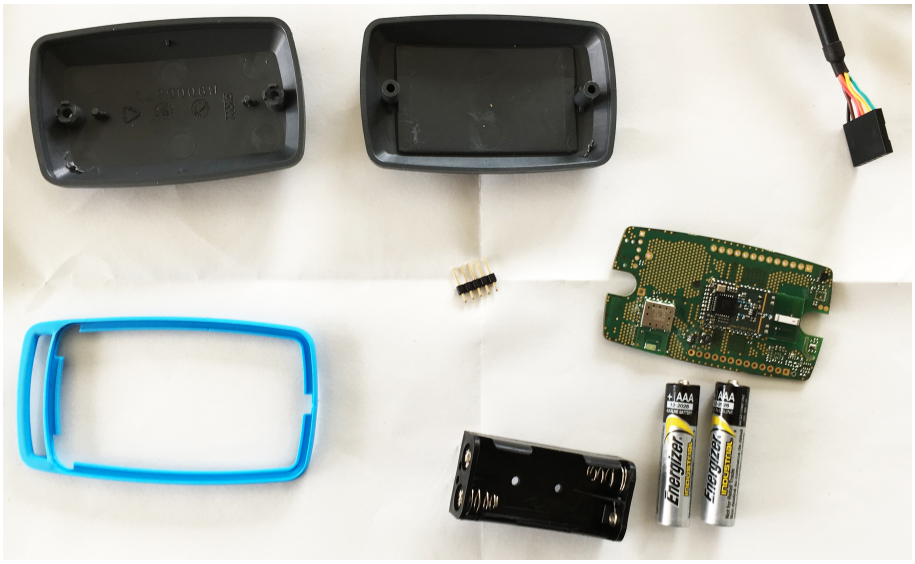

Figure 4.10: FMLR TrackerTwo by Miromico, a compact LoRaWAN tracking device

The FMLR TrackerTwo can be configured via the delivered USB cable (top-right corner in Figure 4.10), using the Hayes command set ('AT commands'). The available AT commands are specified in the FMLR AT Protocol document [108]. On an Unix-based system, the command `screen /dev/tty.usbserial-FT9PB57F 115200,cs8` can be used in the operating system's CLI to initiate communication with the FMLR TrackerTwo and to configure the device. Alternatively, the Arduino IDE's serial monitor with the appropriate port (usbserial-FT9PB57F) and Baud rate (115 200) can be used.
The FMLR TrackerTwo sends its payload in Cayenne LPP format, which was introduced in Section 4.2.2. The following two data types are transmitted: location and battery. The location consists of 3 Bytes each for the longitude, latitude, and altitude value. The battery level is represented using 2 Bytes. This leads to a total payload size of 15 Bytes, with 4 Bytes of Cayenne LPP overhead (two channels, two types).

The FMLR TrackerTwo is configured to send on port 4. However, all ports except 1, 2, and 3 are treated as potential Cayenne LPP ports. Two steps had to be taken in order to integrate the FMLR TrackerTwo into the OTIoT Platform. Firstly, the payload decoder function in the OTIoT TTN Application had to be extended in order to decode the new payload correctly (cf. Listing 4.1). Secondly, the HTTP POST endpoint of the OTIoT Back-End had to be adapted to understand the additional two payload fields `"altitude"` and `"battery"`. The standard Cayenne LPP Payload Functions implementation by TTN could not be used, as they it have overridden the custom payload functions for the OTIoT Tag described in Section 4.2.2. The FMLR TrackerTwo's Cayenne LPP

format is explained in Table 4.4.

Table 4.4: Cayenne LPP payload of the FMLR TrackerTwo (based on [109])

Payload (Hex)	**01 88 07 3C 1E 01 4D FD 00 AD C0 02 02 00 64**	
Data Channel	Data Type	Value
0x01 Sensor	**0x88** = GPS	Latitude: **0x07 3C 1E** = 47.4142
		Longitude: **0x01 4D FD** = 8.5501
		Altitude: **0x00 AD C0** = 444.80 meters
0x02 Sensor	**0x02** = Analog Input	Battery: **0x00 64** = 100% load

4.5 LoRa Gateway

LoRa gateways form the bridges between the radio transmission and the fixed-line network in every LoRaWAN network by offering Internet backhaul capability. A gateway's tasks include the reception and forwarding of uplink packets as well as the transmission of downlink packets to the end-devices. More details about the role of gateways in a LoRaWAN network can be found in Section 4.2.1. A picture of the opened LoRa gateway is shown in Figure 4.11.

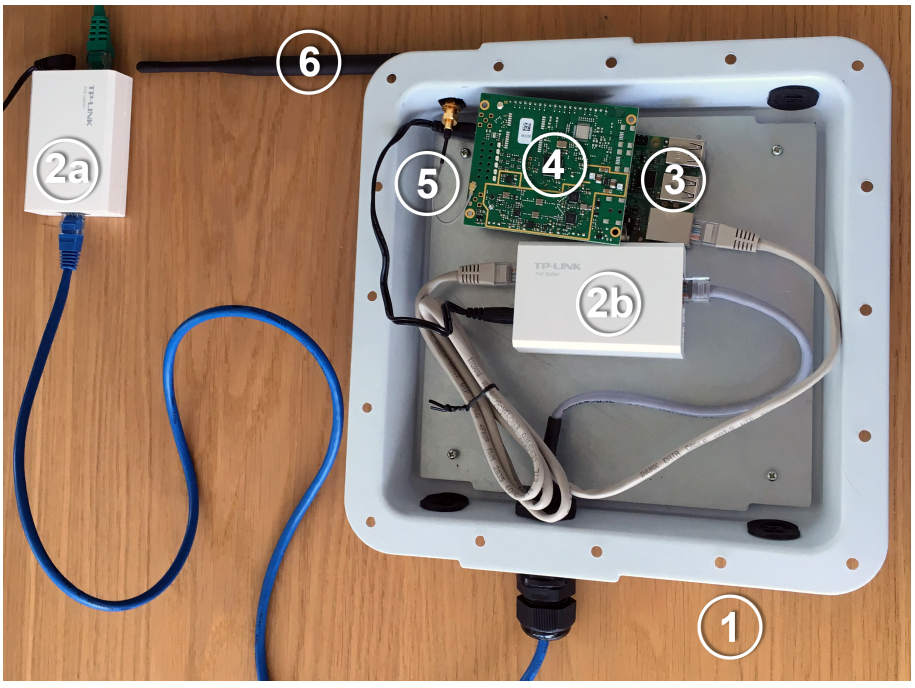

Figure 4.11: LoRa gateway used for developing the OTIoT Tag

As part of this thesis, a LoRa gateway was assembled in order to facilitate the development of the OTIoT Tag by offering indoor coverage at the author's workplace at UZH. The components were selected based on TTN's 'From zero to LoRaWAN in a weekend' manual [59]. In addition, a **(1) steel outdoor enclosure** was ordered in order to prepare the gateway for outdoor deployment. Using a **(2a) Power over Ethernet (PoE) Injector** was suggested by Gonzalo Casas, the community leader of Zurich's TTN community. He uses the same powering option for his own gateway, which this setup is inspired by. A Direct Current (DC) connector with 5 V is used to supply electric current from the **(2b) TP-Link PoE Splitter** to the **IMST iC880a LoRaWAN backplane** (not visible), which powers both the **(3) Raspberry Pi 3 Model B** as well as the **(4) iC880A-SPI - LoRaWAN Concentrator 868MHz** board. The **(5) pigtail for iC880A-SPI**, an adapter from Subminiature A (SMA) (antenna-side) to Ultra Miniature Coaxial Connector (u.FL) (board-side), is used to connect the **(6) SMA Antenna for iC880A-SPI** to the concentrator. When the gateway is connected via Ethernet, it can be administrated via the TTN Console or using its Secure Shell (SSH) interface. All hardware components of the LoRa gateway and their cost are listed in Table 4.5.

Table 4.5: Hardware components of the LoRa gateway

Component	**Supplier**		**Price**
Steel Outdoor Enclosure *1x Ethernet / 4x N-Type holes*	AerialNet [110]	CHF	57.27
TP-Link PoE Injector/Splitter TL-PoE200	BRACK AG [111]	CHF	35.75
Raspberry Pi 3 Model B	BRACK AG [112]	CHF	47.40
Kingston microSDHC Card Class 4 16GB	BRACK AG [113]	CHF	9.70
iC880A-SPI - LoRaWAN Conc. 868MHz	IMST Webshop [114] *EUR 155.-,* 19 % *VAT*	CHF	231.38
Pigtail for iC880A-SPI	IMST Webshop [115] *EUR 6.50,* 19 % *VAT*		
SMA Antenna for iC880A-SPI	IMST Webshop [116] *EUR 6.50,* 19 % *VAT*		
IMST iC880a LoRaWAN backplane *by Gonzalo Casas*	Tindie [117]	CHF	6.00
Total Costs		**CHF**	**387.50**

4.6 OTIoT Back-End

The design and implementation of the OTIoT Back-End was one of the main tasks in Jan Meier's Master thesis [7]. A solid code base consisting of a Python 'Django' [118] web application with a 'PostGIS' database [119], based on a 'nginx' web server [120] with 'Celery' task queue [121] and 'Redis' cache [122] was ready to be extended. This back-end offered a good foundation for the development done as part of this Master thesis. For easier maintainability and portability, the whole back-end is set up using multiple 'Docker' [123] containers.

The main contribution to the OTIoT Back-End of this Master thesis its integration with the TTN APIs.

4.6.1 Integration using the The Things Network APIs

For the interconnection of the OTIoT Platform and TTN, two APIs were integrated into the OTIoT Back-End, which required extending the Python source code.

HTTP Integration

The HTTP Integration [60] activated in the OTIoT TTN Application is used to receive uplink and to send downlink messages. Integrations are publicly offered services that add functionality to a TTN application once they are activated in the TTN Console. As of August 22, 2017, five different integrations are available: Cayenne [124] (for the integration into the IoT dashboard 'myDevices' [125]), Data Storage [126], HTTP Integration [60], IFTT Maker [127], and OpenSensors [128]. The HTTP Integration, in particular, offers a two-way communication channel with the OTIoT TTN Application.

An **external HTTP POST endpoint** can be specified in the HTTP Integration's configuration. This allows the OTIoT Back-End's POST endpoint to be called every time an uplink message is received in the OTIoT TTN Application. The decoded, converted, and validated data is sent to the back-end as POST body of this request. The only information transmitted in the request body's `"payload_fields"` in case of the OTIoT Tag is the location: `"lat"` and `"lng"`. The meta field `"dev_id"` (Device ID) is used to bind the device created in the OTIoT App to the device registered in the OTIoT TTN Application. Besides that, the arrival time of the message can be read from the `"metadata"` field `"time"`. This date and time information is equal to the point in time when the TTN handler has received the uplink message. The small difference between the time the uplink packet was sent, the time the first gateway has received the uplink packet, and the time the TTN handler has received the uplink message was decided to be negligible for the object tracking use case.
An authorization header can be chosen to be sent with the request, which is needed to authenticate as a user in the OTIoT Back-End. The POST request sent to `https://otiot.csg.uzh.ch/sensor-data/ttn/post/` (accessed Aug. 22, 2017) by the HTTP Integration is shown in Listing 4.5.

```
{ "app_id": "otiot",
  "dev_id": "otiot-tag",
  "hardware_serial": "0102030405060708",
  "port": 1,
  "counter": 18,
  "is_retry": false,
  "confirmed": false,
  "payload_raw": "BzwVAU3x",
  "payload_fields": {
    "lat": 47.4133,
    "lng": 8.5489
  },
  "metadata": {
    "time": "2017-08-17T18:57:47.8855633Z",
    "frequency": 868.5,
    "modulation": "LORA",
    "data_rate": "SF9BW125",
    "coding_rate": "4/5",
    "gateways": [{
      "gtw_id": "eui-b827ebfffeb8a0ec",
      "timestamp": 3751389500,
      "time": "2017-08-17T18:57:47.93665Z",
      "channel": 2,
      "rssi": -97,
      "snr": 9,
      "rf_chain": 1,
      "latitude": 47.41431,
      "longitude": 8.54997,
      "altitude": 443
    }]
  }
}
```

Listing 4.5: Uplink message (JSON) sent to the OTIoT Back-End POST endpoint

Besides offering a hook to call an external POST endpoint, the **HTTP Integration's own POST endpoint** can be used to schedule a downlink message to a specific node.

```
{ "dev_id": "otiot-tag",
  "port": 10,
  "confirmed": true,
  "payload_fields": {
    "armed": true,
    "acc_sensitivity": 15,
    "gps_tolerance": 0.0003,
    "moving_send_delay": 300
  }
}
```

Listing 4.6: Downlink message (JSON) sent to the HTTP Integration POST endpoint

If the option `"confirmed"` is set to `true`, the OTIoT TTN Application receives an uplink message to acknowledge the successful transmission of the downlink message to the node. The fields in `"payload_fields"` are encoded using the encoder function presented in Section 4.2.2. Authentication is achieved by adding an Access Key, a string used for authentication that is unrelated to the AppKey), as a GET parameter named 'key'. A POST request with the default configuration values sent to the HTTP Integration's POST endpoint at `https://integrations.thethingsnetwork.org/ttn-eu/api/v2/down/otiot/otiot-integration` (accessed Aug. 22, 2017) is shown in Listing 4.6.

Application Manager API

The second TTN API integrated is the Application Manager API [63], which allows the registration (HTTP POST request), update (HTTP PUT request), and deletion (HTTP DELETE request) of devices in a TTN application. It can also be used to register new applications, to simulate uplink messages, and to create dry-runs for uplink and downlink messages.

In order to register a end-device in the OTIoT TTN Application, the DevEUI, the AppEUI, and the AppKey have to be delivered to the Application Manager API. For authorization, an Access Key is added as 'Authorization' header field of each request, prefixed with 'Key'. In Listing 4.7, an example of JSON data sent to `http://eu.thethings.network:8084/applications/otiot/devices/otiot-tag` as POST request is shown. This request creates a new OTAA end-device with the device ID `"otiot-tag"` within the `"otiot"` application by specifying its three OTAA credentials. The frame counter `("f_cnt")` and `"last_seen"` timestamp are reset. As all tags within the OTIoT Platform are moving devices, no fixed `"altitude"`, `"longitude"`, and `"latitude"` is defined.

```
{ "description": "Location tracking tag for the OTIoT TTN Application
    ",
  "altitude": 0,
  "longitude": 0,
  "latitude": 0,
  "lorawan_device": {
    "app_id": "otiot",
    "dev_id": "otiot-tag",
    "activation_constraints": "otaa",
    "dev_eui": "0102030405060708",
    "app_eui": "0102030405060708",
    "app_key": "01020304050607080102030405060708",
    "disable_f_cnt_check": false,
    "f_cnt_down": 0,
    "f_cnt_up": 0,
    "last_seen": 0,
    "uses32_bit_f_cnt": true
  }
}
```

Listing 4.7: POST request sent to Application Manager API to create a new device

TTN's *Data API* [61] was originally intended to be used for sending downlink messages to the OTIoT Tag using the MQTT protocol, as outlined in Section 3.2. This intention was dropped after it was discovered that the same functionality is offered by the HTTP Integration's POST endpoint. In the future, both usages of the HTTP Integration for receiving uplink data and scheduling downlink messages could be replaced by the Data API's subscribe and publish functionality. Benefits of MQTT over HTTP [129] include faster response times, higher throughput, and low bandwidth usage. Furthermore, MQTT offers a topic-based publish and subscribe model, which suits IoT applications with many data streams.

The final OTIoT architecture therefore remains the same as illustrated in Figure 3.3 – except without the usage of the Data API.

4.7 OTIoT App

The OTIoT App is a web application, which provides users with a front-end based on Vue.js [130] to allow for interaction with the OTIoT Platform. This front-end was developed by Jan Meier as part of his Master thesis [7], where a detailed description of the OTIoT App's features can be found.

In order to allow the integration with the TTN, five fields were added to the OTIoT App: 'DevEUI', 'AppKey', 'Accelerometer Sensitivity', 'GPS Tolerance', and 'Report Interval' (corresponds to the moving send delay). The AppEUI is not user-configurable as it is the same for all tags. Whenever a tag is registered through the dialog shown in Figure 4.12, a device is also registered in the OTIoT TTN Application using the Application Manager API.

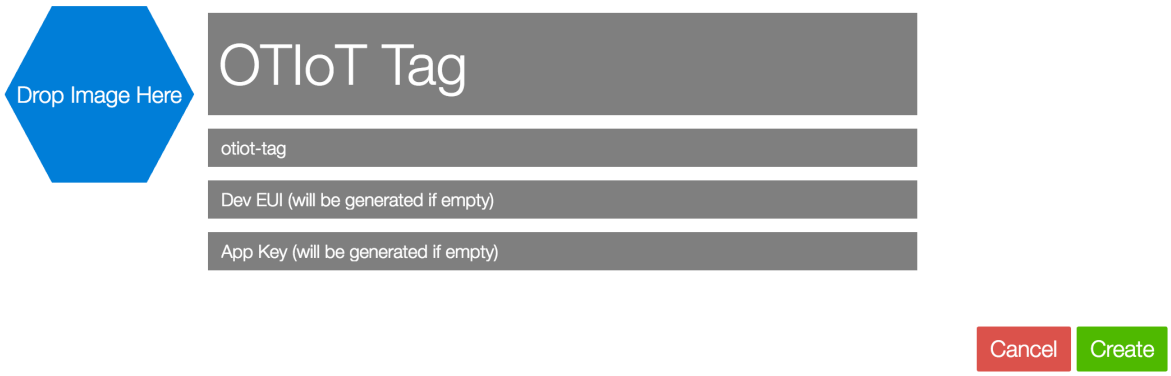

Figure 4.12: Tag registration dialog in the OTIoT App

In case a user changes the configuration of a tag in the OTIoT App's tag detail view, shown in Figure 5.5, a downlink message is scheduled by usage of the HTTP Integration API. The same procedure is called when a user arms a device by clicking the toggle button in the overview or tag detail view.

The current version of the OTIoT App is deployed on `https://otiot.csg.uzh.ch` (accessed Aug. 22, 2017).

Chapter 5

Evaluation

In this chapter, the evaluation of the OTIoT Tag as well as its integration into the OTIoT Platform is conducted as a proof of operability. The test environment as well as the three tests conducted are described in Section 5.1 to Section 5.4. Section 5.5 summarizes the findings of the evaluation at the end of this chapter.

5.1 Test Environment

The test environment consists of the following prerequisites:

1. The **LoRa gateway** described in Section 4.5 is powered using PoE, is connected to the Internet, and is registered in TTN.

2. The **OTIoT TTN Application** has the Payload Functions set up according to Section 4.2.2, and the HTTP Integration is activated and configured according to Section 4.6.1.

3. The **OTIoT Back-End** is deployed on `https://otiot.csg.uzh.ch`, providing a HTTP POST API endpoint for incoming location data.

4. The **OTIoT Tag** is powered using a power bar and packaged for outdoor testing as shown in Figure 5.1.

TTN's LoRaWAN coverage is not part of this evaluation and is taken as given in the following tests. All tests were conducted within approximately 1 500 m of the location of LoRa gateway (cf. Figure 4.11) inside the UZH building in Zurich-Oerlikon, Switzerland. This is because the smallest alert circle configurable in OTIoT App is 1 km.

Figure 5.1: OTIoT Tag in packaged form as testing device

5.2 Integration Test

The 'Integration Test' aims to prove that the integration between the OTIoT Tag, the OTIoT TTN Application, and the OTIoT Platform (back- and front-end) works as intended. The following steps are taken for this test:

1. The user registers and logs in into the OTIoT App.
2. The OTIoT Tag is registered in the OTIoT App manually by the user (cf. Figure 4.12).
3. The OTIoT Tag is automatically registered in the OTIoT TTN Application.
4. The DevEUI, AppEUI, and AppKey are copied from the OTIoT TTN Application to the OTIoT Tag's script manually by the user, using the TTN Console. The software is compiled and uploaded to the OTIoT Tag using the Arduino IDE.
5. Once the upload has finished, the OTIoT Tag tries to join the TTN using OTAA. If no 'Join Accept' packet is received from a nearby LoRa gateway, a retry is started automatically every 30 s.

It is shown in Figure 5.2 in the TTN Console that the OTIoT has successfully joined TTN at 20:47:20.

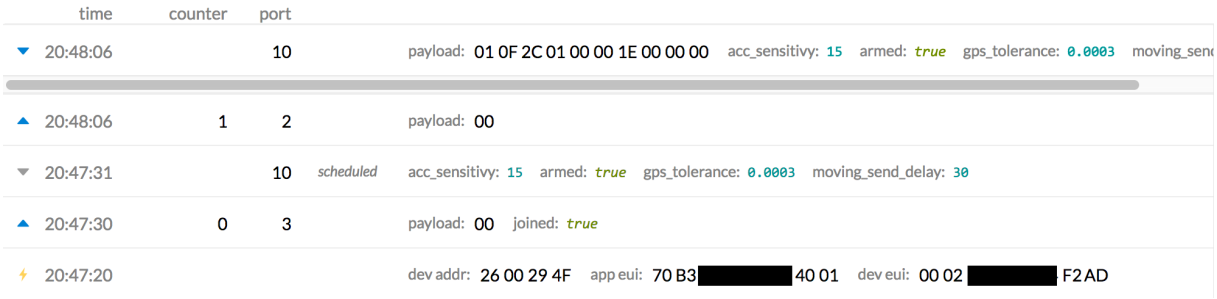

Figure 5.2: The join procedure of the OTIoT Tag in the TTN Console

5.3 Downlink Communication Test

A downlink packet is scheduled for transmission in order to configure the OTIoT Tag once it has joined, and whenever the configuration is changed by the user in the OTIoT App. The following steps are taken in this 'Downlink Communication Test':

1. *Prerequisite: The OTIoT Tag has joined the TTN, as described in Section 5.2.*
2. After joining, a 'joined' uplink packet is sent automatically by the OTIoT Tag (at 20:47:30 in Figure 5.2).
3. When the OTIoT Back-End receives the 'joined' packet, a downlink message with the current tag's configuration is automatically scheduled in the OTIoT TTN Application for transmission to the OTIoT Tag (at 20:47:31).
4. The downlink packet is either sent to the OTIoT Tag directly in the receive windows of 'joined' uplink packet, or after the first polling uplink packet is sent by the OTIoT Tag (at 20:48:06).
5. The OTIoT Tag receives the downlink packet (at 20:48:06) and updates its configuration according to the payload fields `"armed"`, `"acc_sensitivity"`, `"gps_tolerance"`, and `"moving_send_delay"`.

Figure 5.2 shows (from bottom to top) the OTAA device activation, the 'joined' uplink message, the scheduling of the configuration downlink message by the OTIoT Back-End upon reception of the joined message, the first polling uplink packet, and the subsequent transmission of the downlink message.

5.4 Location Sending Algorithm Test

The 'Location Sending Algorithm Test' is used to test the OTIoT Tag's software as presented in Section 4.3.2. Listing 5.1 shows the serial output of the OTIoT Tag while executing the location sending algorithm. The following test steps were conducted:

1. *Prerequisite: The OTIoT Tag is mounted on a bicycle (cf. Figure 5.3).*
2. *Prerequisite: The OTIoT Tag is armed in the OTIoT App and has an alert circle of 1 km from the LoRa gateway configured.*
3. *Prerequisite: The OTIoT Tag has joined the TTN, as described in Section 5.2 (lines 1 to 5).*
4. *Prerequisite: The OTIoT Tag has sent a 'joined' uplink packet and has updated its configuration according to the received downlink packet, as described in Section 5.3 (lines 6 to 19). It is therefore in armed state after startup.*
5. The OTIoT Tag saves the current accelerometer values (line 20).
6. TheOTIoT Tag tries to obtain the current GPS location, sends it as a LoRaWAN location uplink packet, and saves it (lines 21 to 28).
7. The tagged bicycle starts moving away (i.e., is stolen by someone).
8. The OTIoT Tag detects movement using the accelerometer (lines 30 to 32).
9. The OTIoT Tag tries to obtain the current GPS location and compares it to the last saved GPS location (lines 33 to 38).
10. If the current GPS location lays outside of the GPS tolerance, the OTIoT Tag sends a LoRaWAN location uplink packet (lines 37 to 42).
11. The OTIoT Tag is now in 'moving' state and starts sending LoRaWAN location uplink packets periodically (lines 43 to 52).
12. The tagged bicycle leaves the alert circle configured in the OTIoT App.
13. The OTIoT Platform creates and displays an alert to the OTIoT App user, as shown in Figure 5.4.

Figure 5.3: OTIoT Tag mounted on a bicycle

```
LoRa: Start joining...
LMIC 187: EV_JOINING
LMIC 935240: EV_JOINED
LoRa: Joined
LoRa: Setup complete
LoRa: Sending 'joined' packet
LMIC: Packet queued
LoRa: Waiting for configuration downlink...
Accelerometer: Base Values (after calibration)
=> X = 0 Y = 0 Z = 63
LMIC 1439678: EV_TXCOMPLETE (includes waiting for RX windows)
LoRa: Queing uplink polling packet...
LMIC: Packet queued
LMIC 1773124: EV_TXCOMPLETE (includes waiting for RX windows)
Received data, Port (receiving): 10, Size: 64
=> Status: ARMED
=> Accelerometer Sensitivity: 15
=> GPS Tolerance: 0.000300
=> Moving Check Delay: 30
Accelerometer: Base Values => X = 8 Y = 248 Z = 75
GPS: Checking location now
GPS: Trying to get location... ---> Note: executed 3x <---
GPS: Saving new location
GPS: New location found: 47.4144, 8.5491, 495.60
LMIC: Packet queued
LoRa: Packet queued (lat: 47.4143, lng: 8.5490)
GPS: Location Base Values (on init)
=> Latitude: 47.4144, Longitude: 8.5491
---> (Note: Removed some accelerometer checks) <---
Status: ARMED
X = 28, Y = 250, Z = 90
Accelerometer: EVENT - Vibration detected!
GPS: Checking location now
GPS: Trying to get location... ---> Note: executed 3x <---
GPS: Saving new location
GPS: New location found: 47.4142, 8.5486, 471.50
GPS: Diff (lat: 0.0002, lng: 0.0004) is OUTSIDE tolerance
=> Latitude: 47.4142, Longitude: 8.5486
LMIC: Packet queued
LoRa: Packet queued (lat: 47.4141, lng: 8.5485)
Next check in 10 seconds.
LMIC 5607566: EV_TXCOMPLETE (includes waiting for RX windows)
...
Status: ARMED + MOVING
GPS: Checking location now
GPS: Trying to get location... ---> Note: executed 3x <---
GPS: Saving new location
GPS: New location found: 47.4141, 8.5483, 465.00
GPS: Diff (lat: 0.0001, lng: 0.0003) is within tolerance
=> Latitude: 47.4141, Longitude: 8.5483
LoRa: No packet sent
Next check in 30 seconds.
```

Listing 5.1: Serial output of the OTIoT Tag's location sending algorithm

The location data collected during the test is shown in Figure 5.5. A 'Report Interval' of 30 s was defined in the OTIoT App for a higher resolution of data points. For 'Accelerometer Sensitivity' and 'GPS Tolerance', the default values of 15 and 0.000 3 decimal degrees were configured. The alert was correctly triggered after leaving the alert circle, which is shown in Figure 5.4.

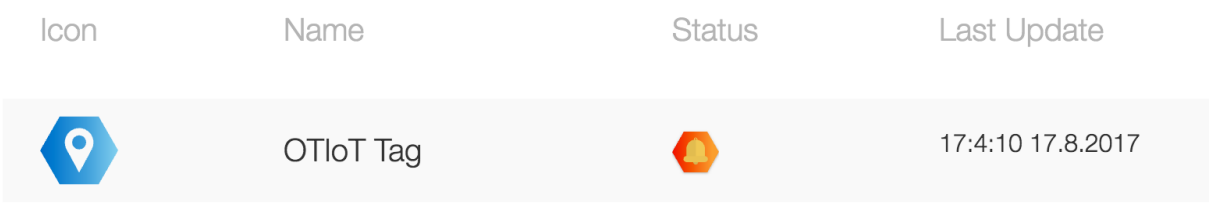

Figure 5.4: Alert shown to the user in the OTIoT App after leaving the configured circle

5.5 Findings

The evaluation of the OTIoT Tag with the aforementioned three tests has shown that the planned features listed in Section 4.3 could all be implemented:

- The OTIoT Tag can send encoded location information as LoRaWAN uplink packets when the tagged object is moved. GNSS is only used if the tag is suspected to be moving. This was proved in the 'Location Sending Algorithm Test'.
- The OTIoT Tag can send (empty) polling messages as LoRaWAN uplink packets in order to be able to receive configuration as LoRaWAN downlink packets. This was proved in the 'Downlink Communication Test'.
- The OTIoT Tag can send a 'joined' message as LoRaWAN uplink packet to be able to receive a configuration downlink packet on startup. This was proved in the 'Downlink Communication Test'.
- The OTIoT Tag can receive and decode configuration sent as LoRaWAN downlink packets in order to configure its behavior. This was proved in the 'Downlink Communication Test'.

A load test was not conducted as it is illegal to use more airtime than the ETSI duty cycles permit. Respecting TTN's Fair Access Policy [81] depends on the user's behavior on the OTIoT Platform: If too many configuration changes are made in the OTIoT App within one day, the policy of sending a maximum of ten downlink messages is violated. Also, if a tagged object goes missing for longer than half a day with the default report interval of 300 s, or if the report interval is set very low by the user, TTN's uplink policy is violated. As calculated in Section 4.2.3, a report interval of 537 s or larger is necessary for continuous location reporting, according to the TTN Fair Access Policy.
Another aspect, which was not tested, is the current TTN coverage in the city of Zurich. Furthermore, no alternative LoRaWAN providers were used and tested as part of this Master thesis.

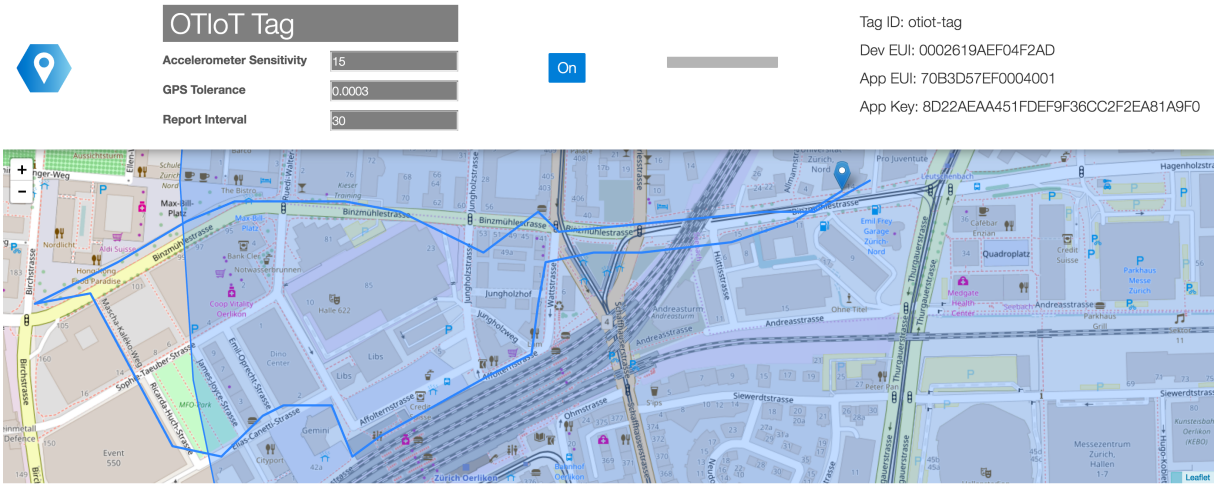

Figure 5.5: Collected location data shown in the OTIoT App's tag detail view

The OTIoT Platform is found to be not yet fully customer-ready as the deployment process still involves the usage of the TTN Console to read out the OTAA credentials and the manual compilation of the adapted Arduino source code.
Additionally, it has to be stated that the OTIoT Tag is still in a prototypical state. The tag's hardware is not optimized in terms of size, energy consumption, and hardware costs. The energy consumption of the OTIoT Tag was therefore not measured and analyzed as part of this evaluation.

Chapter 6

Summary and Conclusions

The goal of this Master thesis is the implementation of a low power and subscription-free object tracking device, as introduced in Chapter 1. Several alternative communication technologies, localization methods, and tag hardware options for developing such a device were examined in Chapter 2. In Chapter 3, the options for communication technologies, localization methods, and tag hardware were discussed and the taken decisions were explained. LoRaWAN's [28] low energy consumption, long range, relatively low sensor costs, and available coverage through TTN [32] were used as arguments to justify its choice as communication technology. As localization method only GNSS was deemed a reasonable option due to its unmatched high accuracy and global coverage. With these two decisions taken, it became evident that using the Arduino hardware ecosystem fits best for the development of the OTIoT Tag. Details and insights about LoRa and LoRaWAN are presented next at the beginning at the beginning of Chapter 4. The TTN's architecture, its usage for OTIoT and its limitations are examined in a next section. The hardware, the software, the location tracking algorithm, and the configuration options of the OTIoT Tag are then described in Section 4.3. Together with Section 4.6, which concerns the integration of the TTN into the OTIoT Platform, this section forms the main part of this thesis. Besides the tag and the integration, also a self-assembled LoRa gateway and the FMLR TrackerTwo [107], a commercial LoRaWAN tracker, are presented in this chapter. Following the implementation chapter, the OTIoT Tag and the OTIoT Platform are tested and evaluated in Chapter 5. The three tests' findings are summarized, and the contributions of this thesis as well as its limitations are pointed out.

To conclude this thesis, it can be stated that the hardware design in Section 3.1 and the architecture in Section 3.2 have proven to work as intended for a LoRaWAN object tracking device. However, the usage of LoRaWAN comes with several limitations. Firstly, only limited downlink functionality exists for LoRaWAN Class A devices, which requires the implementation of a polling function. Secondly, real-time tracking requires relatively large amounts of data transmitted over LoRaWAN. The TTN's Fair Access Policy makes it impossible to have real-time tracking as the minimal allowed sending interval for continuous location transmission is close to 9 min, as shown in Section 4.2.3. Swisscom LPN's [33] largest commercial data package includes only 144 uplink messages per device, which is even less than what TTN allows (161 uplink messages).
Using GNSS currently remains the only option for accurate localization of a tracking

device. The high energy consumption of GNSS modules are a major disadvantage for battery-powered mobile devices such as the OTIoT Tag. Furthermore, GNSS modules are expensive and only work well in line of sight scenarios (i.e., outdoors). The interesting option of LoRa Geolocation [38] could not be evaluated as there is no public availability as of mid 2017. Using an accelerometer to detect movement before accessing the GNSS has proved to be able to reduce access to the GNSS module, as shown in Section 5.4. The alternative localization method of WLAN fingerprinting is not well suited for a LoRaWAN device as it requires to constantly send potentially large uplink messages with the sighted WLAN Service Set Identifiers (SSIDs) and their RSSI, which count towards the allowed duty cycle or policy.

Finally, it was shown that object tracking is possible with a hardware combination of a LoRaWAN modem, a GNSS module, an accelerometer, and an Arduino micro-controller. TTN has been proven to be a reliable and developer-friendly LoRaWAN provider, even though its coverage remains untested.

Future work could include the minimization of the OTIoT Tag's size and its energy consumption, the improvement of the tag's deployment process, and the evaluation of different LoRaWAN network providers and their coverage. Overcoming these limitations of the OTIoT Platform, end-users could be provided with a well-integrated, free of charge object tracking and anti-theft solution. Furthermore, once deployments of new LPWAN technologies such as LTE-M and NB-IoT, which are mentioned in Chapter 2, are available, these technology options should be evaluated for their potential integration into the OTIoT Platform.

References

[1] Apple, Inc.: *Track and find your missing Apple device*, https://support.apple.com/explore/find-my-iphone-ipad-mac-watch, Online; accessed Aug. 22, 2017.

[2] Google, Inc.: *Find My Device*, https://play.google.com/store/apps/details?id=com.google.android.apps.adm, Online; accessed Aug. 22, 2017.

[3] Prey, Inc. Ltd.: *Tracking software: find stolen laptops, phones & tablets*, https://www.preyproject.com, Online; accessed Aug. 22, 2017.

[4] Tracker.ch AG: *Tracker.com, GPS, Tracking, Geofencing, GSM, iPhone App*, http://www.tracker.com/chen/default.aspx, Online; accessed Aug. 22, 2017.

[5] Tile, Inc.: *Tile: Find Your Keys, Wallet & Phone with Tile's App and Bluetooth Tracker Device*, https://www.thetileapp.com, Online; accessed Aug. 22, 2017.

[6] Gartner, Inc.: *Internet of Things*, http://www.gartner.com/it-glossary/internet-of-things/, Online; accessed Aug. 22, 2017.

[7] J. Meier: „Design, Implementation, and Evaluation of an Object Tracking Motion Detection System“, Master's thesis, Department of Informatics, University of Zurich, May 2017. [Online]. Available: https://files.ifi.uzh.ch/CSG/staff/schmitt/Extern/Theses/Jan_Meier_MA.pdf.

[8] P. R. Egli: *LPWAN (Low Power Wide Area Network) - Overview of Emerging Technologies for Low Power Wide Area Networks in Internet of Things and M2M Scenarios*, https://www.slideshare.net/PeterREgli/lpwan, Mar. 13, 2015, Online; accessed Aug. 22, 2017.

[9] Bluetooth Special Interest Group, Inc.: *Bluetooth Technology Website*, https://www.bluetooth.com, Online; accessed Aug. 22, 2017.

[10] J. H. Schiller: *Mobile Communications*, 2nd ed. Pearson Education, 2003.

[11] NFC Forum: *NFC Forum*, https://nfc-forum.org, Online; accessed Aug. 22, 2017.

[12] NFC Forum: *About the Technology*, https://nfc-forum.org/what-is-nfc/about-the-technology/, Online; accessed Aug. 22, 2017.

[13] International RFID Institute, Inc.: *Passive RFID*, http://international-rfid-institute.org/rfid-technologies/passive/, Online; accessed Aug. 22, 2017.

[14] TrackR, Inc.: *Track your phone, wallet, keys & anything else with TrackR!*, https://www.thetrackr.com, Online; accessed Aug. 22, 2017.

[15] VisitHelsinki: *Free WiFi in Helsinki*, http://www.visithelsinki.fi/en/come/welcome-to-helsinki/free-wifi-in-helsinki, Online; accessed Aug. 22, 2017.

[16] City of Mountain View: *Mountain View - Google WiFI*, http://www.mountainview.gov/depts/it/wifi.asp, Online; accessed Aug. 22, 2017.

[17] Qualcomm, Inc.: *The Evolution of Mobile Technologies: 1G 2G 3G 4G LTE*, https://www.qualcomm.com/media/documents/files/the-evolution-of-mobile-technologies-1g-to-2g-to-3g-to-4g-lte.pdf, Jun. 2014, Online; accessed Aug. 22, 2017.

[18] Link Labs, Inc.: *LTE-M & 2 Other 3GPP IoT Technologies To Get Familiar With*, https://www.link-labs.com/blog/lte-iot-technologies, Apr. 12, 2017, Online; accessed Aug. 22, 2017.

[19] 3rd Generation Partnership Project: *3GPP - A Global Initiative*, http://www.3gpp.org, Online; accessed Aug. 22, 2017.

[20] Verizon Communications, Inc.: *Verizon launches industry's first LTE Category M1 (Cat M1) nationwide network for IoT*, http://www.verizon.com/about/news/verizon-launches-industrys-first-lte-category-m1-cat-m1-nationwide-network-iot, Online; accessed Aug. 22, 2017.

[21] Link Labs, Inc.: *An Overview Of Narrowband IoT (NB-IoT)*, https://www.link-labs.com/blog/overview-of-narrowband-iot, Jul. 27, 2016, Online; accessed Aug. 22, 2017.

[22] Vodafone Group, Plc.: *The future in our hands with the commercial launch of NB-IoT in Vodafone Spain*, http://www.vodafone.com/content/index/what/technology-blog/nbiot-commercial-launch-spain.html, Jan. 23, 2017, Online; accessed Aug. 22, 2017.

[23] Link Labs, Inc.: „A Comprehensive Look at Low Power, Wide Area Networks“, Link Labs, Inc., Tech. Rep., Aug. 2016. [Online]. Available: https://www.link-labs.com/lpwan.

[24] ZigBee Alliance: *ZigBee Alliance*, http://www.zigbee.org, Online; accessed Aug. 22, 2017.

[25] IEEE Standards Association: „IEEE Standard for Low-Rate Wireless Networks“, IEEE Computer Society, IEEE Std 802.15.4 - 2015, 2015. [Online]. Available: http://standards.ieee.org/getieee802/download/802.15.4-2015.pdf.

[26] Sigfox S.A.: *Sigfox – The Global Communications Service Provider for the Internet of Things (IoT)*, https://www.sigfox.com, Online; accessed Aug. 22, 2017.

[27] LoRa Alliance: *LoRa Alliance™ – Technology*, https://www.lora-alliance.org/what-is-lora, Online; accessed Aug. 22, 2017.

[28] Semtech Corporation: *What is LoRa?*, http://www.semtech.com/wireless-rf/internet-of-things/what-is-lora/, Online; accessed Aug. 22, 2017.

[29] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melià-Seguí, and T. Watteyne, „Understanding the Limits of LoRaWAN“, *IEEE Communications Magazine*, Feb. 2017 (in printing process). [Online]. Available: https://arxiv.org/pdf/1607.08011.pdf.

[30] Semtech Corporation: „AN1200.22 LoRa™ Modulation Basics“, Semtech Corporation, Tech. Rep. AN1200.22, May 2015. [Online]. Available: http://www.semtech.com/images/datasheet/an1200.22.pdf.

[31] LoRa Alliance: *LoRa Alliance™ – Wide Area Network for IoT*, https://www.lora-alliance.org, Online; accessed Aug. 22, 2017.

[32] The Things Network Foundation: *The Things Network*, https://www.thethingsnetwork.org, Online; accessed Aug. 22, 2017.

[33] Swisscom AG: *Swisscom LPN*, http://lpn.swisscom.ch, Online; accessed Aug. 22, 2017.

[34] W. Giezeman: *The Things Network Launches World's First Crowdfunded Internet of Things Data Network in Amsterdam and The World is Next*, http://thethingsnetwork.pr.co/108437-the-things-network-launches-world-s-first-crowdfunded-internet-of-things-data-network-in-amsterdam-and-the-world-is-next, Aug. 19, 2015, Online; accessed Aug. 22, 2017.

[35] Swisscom AG: *Das schweizweite Netz für das Internet der Dinge ist live*, https://www.swisscom.ch/de/about/medien/infos-und-fakten/2016-das-schweizweite-netz-fuer-das-internet-der-dinge-ist-live.html, Oct. 5, 2016, Online; accessed Aug. 22, 2017.

[36] R. Mautz: *Indoor Positioning Technologies*. Habilitation thesis, Institute of Geodesy and Photogrammetry, Department of Civil, Environmental and Geomatic Engineering, ETH Zurich, Feb. 2012. [Online]. Available: https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/54888/eth-5659-01.pdf?sequence=1&isAllowed=y.

[37] Sigfox S.A.: *Sigfox Geolocation, the simplest and most efficient IoT location service*, https://www.sigfox.com/en/sigfox-geolocation, Online; accessed Aug. 22, 2017.

[38] Semtech Corporation: *LoRa® Geolocation: Unlocking New Value for IoT Solutions*, https://www.semtech.com/wireless-rf/lora-geolocation, Online; accessed Aug. 22, 2017.

[39] Google, Inc.: *Google Geolocation API*, https://developers.google.com/maps/documentation/geolocation/intro, Online; accessed Aug. 22, 2017.

[40] OpenCelliD: *The world's largest Open Database of Cell Towers*, https://opencellid.org, Online; accessed Aug. 22, 2017.

[41] GISGeography.com: *Trilateration vs Triangulation – How GPS Receivers Work*, http://gisgeography.com/trilateration-triangulation-gps/, Mar. 12, 2017, Online; accessed Aug. 22, 2017.

[42] Raspberry Pi Foundation: *Teach, Learn, and Make with Raspberry Pi*, https://www.raspberrypi.org, Online; accessed Aug. 22, 2017.

[43] Arduino: *Open-source electronic prototyping platform enabling users to create interactive electronic objects*, https://www.arduino.cc, Online; accessed Aug. 22, 2017.

[44] Raspberry Pi Foundation: *Raspberry Pi 3 Model B*, https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, Online; accessed Aug. 22, 2017.

[45] Raspberry Pi Foundation: *BCM2837*, https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2837/README.md, Online; accessed Aug. 22, 2017.

[46] Arduino: *Arduino Uno Rev3*, https://store.arduino.cc/arduino-uno-rev3, Online; accessed Aug. 22, 2017.

[47] Arduino: *Arduino Software (IDE)*, https://www.arduino.cc/en/Guide/Environment, Online; accessed Aug. 22, 2017.

[48] Arduino: *Sketch*, https://www.arduino.cc/en/tutorial/sketch, Online; accessed Aug. 22, 2017.

[49] Sigfox S.A.: *Coverage*, https://www.sigfox.com/en/coverage, Online; accessed Aug. 22, 2017.

[50] Pycom, Ltd.: *FiPy*, https://www.pycom.io/product/fipy/, Online; accessed Aug. 22, 2017.

[51] The Things Network Foundation: *Forum*, https://www.thethingsnetwork.org/forum/, Online; accessed Aug. 22, 2017.

[52] The Things Network Switzerland: *Join The Things Network Switzerland on Slack*, https://ttn-ch.herokuapp.com, Online; accessed Aug. 22, 2017.

[53] Open Network Infrastrcuture Association: *Make Zurich 2017*, https://makezurich.ch, Online; accessed Aug. 22, 2017.

[54] Open Network Infrastructure Association: *Open Network Infrastructure Association*, https://opennetworkinfrastructure.org, Online; accessed Aug. 22, 2017.

[55] Stadt Zürich: *Erster MakeZurich Hackathon – 7 Challenges für die Stadt Zürich*, https://www.stadt-zuerich.ch/portal/de/index/ogd/blog/2016/12/makezurich.html, Online; accessed Aug. 22, 2017.

[56] CSEM SA: *GPS-free Positioning for the Internet of Things*, http://www.csem.ch/Page.aspx?pid=42092, Jul. 21, 2016, Online; accessed Aug. 22, 2017.

[57] Semtech Corporation: *Semtech to Highlight LoRa Technology's Geolocation Feature in Mobile World Live Webinar*, http://view6.workcast.net/register?cpak=1146345032859980, Feb. 20, 2017, Online, accessed Aug. 22, 2017.

[58] The Things Network Foundation: *Arduino*, https://www.thethingsnetwork.org/docs/devices/arduino/, Online; accessed Aug. 22, 2017.

[59] The Things Network Switzerland: *From zero to LoRaWAN in a weekend*, https://github.com/ttn-zh/ic880a-gateway/wiki, Online; accessed Aug. 22, 2017.

[60] The Things Network Foundation: *HTTP Integration*, https://www.thethingsnetwork.org/docs/applications/http/, Online; accessed Aug. 22, 2017.

[61] The Things Network Foundation: *MQTT*, https://www.thethingsnetwork.org/docs/applications/mqtt/, Online; accessed Aug. 22, 2017.

[62] MQTT Organization: *MQTT*, http://mqtt.org, Online; accessed Aug. 22, 2017.

[63] The Things Network Foundation: *Application Manager API*, https://www.thethingsnetwork.org/docs/applications/manager/, Online; accessed Aug. 22, 2017.

[64] The Things Network Foundation: *LoRaWAN*, https://www.thethingsnetwork.org/wiki/LoRaWAN/Home, Online; accessed Aug. 22, 2017.

[65] LoRa Alliance – Technical Marketing Workgroup 1.0: „LoRaWAN™: What is it? A technical overview of LoRa® and LoRaWAN™“, LoRa Alliance, Tech. Rep., Nov. 2015. [Online]. Available: https://tinyurl.com/lorawan-techmanual.

[66] LoRa Alliance: *LoRaWAN™ 101 – A Technical Introduction*, https://tinyurl.com/lorawan-101, Online; accessed Aug. 22, 2017.

[67] ETSI: „Short Range Devices (SRD) operating in the frequency range 25 MHz to 1 000 MHz; Part 2: Harmonised Standard covering the essential requirements of article 3.2 of Directive 2014/53/EU for non specific radio equipment“, European Telecommunications Standards Institute (ETSI), Tech. Rep. EN 300 220-2 V3.1.1, Feb. 2017. [Online]. Available: http://www.etsi.org/deliver/etsi_en/300200_300299/30022002/03.01.01_60/en_30022002v030101p.pdf.

[68] LoRa Alliance: *LoRa Alliance™ Technology*, https://www.lora-alliance.org/technology, Online; accessed Aug. 22, 2017.

[69] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent: „LoRaWAN™ Specification 1.0.2“, LoRa Alliance, Inc., Tech. Rep. 1.0.2, Jul. 2016.

[70] LoRa Alliance – Technical committee: „LoRaWAN™ 1.0.2 Regional Parameters“, LoRa Alliance, Inc., Tech. Rep. 1.0.2, Feb. 2017.

[71] The Things Network Foundation: *Security in LoRaWAN and TTN*, https://www.thethingsnetwork.org/wiki/LoRaWAN/Security, Online; accessed Aug. 22, 2017.

[72] The Things Network Foundation: *Address Space in LoRaWAN*, https://www.thethingsnetwork.org/wiki/LoRaWAN/Address-Space, Online; accessed Aug. 22, 2017.

[73] The Things Network Foundation: *Device Registration*, https://www.thethingsnetwork.org/docs/devices/registration.html, Online; accessed Aug. 22, 2017.

[74] The Things Network Foundation: *The Things Network Backend*, https://www.thethingsnetwork.org/wiki/Backend/Home, Online; accessed Aug. 22, 2017.

[75] The Things Network Foundation: *The Things Network CLI*, https://www.thethingsnetwork.org/docs/network/cli/, Online; accessed Aug. 22, 2017.

[76] The Things Network Foundation: *The Things Network Zurich*, https://www.thethingsnetwork.org/community/zurich/, Online; accessed Aug. 22, 2017.

[77] R. Chauhan: *Becoming Official Communities*, https://www.thethingsnetwork.org/article/becoming-official-communities, Dec. 6, 2016, Online; accessed Aug. 22, 2017.

[78] Avanquest North America, Inc.: *myDevices – Cayenne Low Power Payload*, https://mydevices.com/cayenne/docs/lora/#lora-cayenne-low-power-payload, Online; accessed Aug. 22, 2017.

[79] ISPO Alliance: *ISPO Smart Object Guidelines*, https://www.ipso-alliance.org/smart-object-guidelines/, Online; accessed Aug. 22, 2017.

[80] The Things Network Foundation: *The Things Network Console*, https://www.thethingsnetwork.org/docs/network/console/, Online; accessed Aug. 22, 2017.

[81] The Things Network Foundation: *Duty Cycle*, https://www.thethingsnetwork.org/wiki/LoRaWAN/Duty-Cycle, Online; accessed Aug. 22, 2017.

[82] Cambridge University Press: *Meaning of "gentleman's agreement" in the English Dictionary*, http://dictionary.cambridge.org/dictionary/english/gentleman-s-agreement, Online; accessed Aug. 22, 2017.

[83] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, „A Study of LoRa: Long Range & Low Power Networks for the Internet of Things", *Sensors*, vol. 16, no. 9, pp. 1–18, Sep. 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/9/1466.

[84] Semtech Corporation: „SX1272/3/6/7/8: LoRa Modem Designer's Guide", Semtech Corporation, Tech. Rep. AN1200.13, Jul. 2013. [Online]. Available: http://www.semtech.com/images/datasheet/LoraDesignGuide_STD.pdf.

[85] A. van Bentem: *LoRa(WAN) airtime calculator*, https://tinyurl.com/ttn-airtime, Online; accessed Aug. 22, 2017.

[86] T. Amberg: *LoRaWAN IoT with Arduino Uno, Dragino v1.3 & TheThingsNetwork*, Sep. 2016, Online; accessed Aug. 22, 2017. [Online]. Available: http://www.tamberg.org/chopen/2016/LoRaWANIoTWorkshop.pdf.

[87] ElecFreak Co, Ltd: *Freeduino UNO*, http://www.elecfreaks.com/wiki/index.php?title=Freaduino_UNO, Online; accessed Aug. 22, 2017.

[88] Dragino Technology Co., Ltd.: *LoRa Shield*, http://www.dragino.com/products/module/item/102-lora-shield.html, Online; accessed Aug. 22, 2017.

[89] HOPE Microelectronics Co., Ltd.: *RFM95W 868/915Mhz RF Transceiver Module – LoRa module*, http://www.hoperf.com/rf_transceiver/lora/RFM95W.html, Online; accessed Aug. 22, 2017.

[90] Semtech Corporation: *SX1276 137 MHz to 1020 MHz Low Power Long Range Transceiver*, http://www.semtech.com/wireless-rf/rf-transceivers/sx1276/, Online; accessed Aug. 22, 2017.

[91] Dragino Technology Co., Ltd.: *Lora Shield - Wiki for Dragino Project*, http://wiki.dragino.com/index.php?title=Lora_Shield, Online; accessed Aug. 22, 2017.

[92] u-blox AG: *NEO-6 series*, https://www.u-blox.com/en/product/neo-6-series, Online; accessed Aug. 22, 2017.

[93] BOXTEC AG: *Freaduino Uno Rev1.8*, http://shop.boxtec.ch/freaduino-uno-rev18-p-41562.html, Online; accessed Aug. 22, 2017.

[94] BLUEBERRY E GmbH: *Dragino LoRa Shield - support 868M frequency Add-Ons & Module Wireless / Drahtlos*, https://get.blueberrye.io/add-ons/wireless-drahtlos/dragino-lora-shield-support-868m-frequency/a-1690080/, Online; accessed Aug. 22, 2017.

[95] BOXTEC AG: *GY-NEO6MV2 GPS Modul*, http://shop.boxtec.ch/neo6mv2-gps-modul-p-42735.html, Online; accessed Aug. 22, 2017.

[96] PLAY-ZONE GmbH: *MMA7455 Accelerometer / Beschleunigungssensor*, http://www.play-zone.ch/de/mma7455-accelerometer-beschleunigungssensor.html, Online; accessed Aug. 22, 2017.

[97] BRACK.CH AG: *Xtorm Powerbank FS100*, https://www.brack.ch/xtorm-powerbank-fs100-489622, Online; accessed Aug. 22, 2017.

[98] BOXTEC AG: *Breadboard Clear – 8.2*5.3cm*, http://shop.boxtec.ch/breadboard-clear-8253cm-p-40199.html, Online; accessed Aug. 22, 2017.

[99] BOXTEC AG: *Breadboard Jumper Wire m-m (65 Stk) - Mikrokontroller Zubehör Kabel*, http://shop.boxtec.ch/breadboard-jumper-wire-stk-p-41285.html, Online; accessed Aug. 22, 2017.

[100] T. Telkamp and M. Kooijman: *arduino-lmic/examples/ttn-otaa/ttn-otaa.ino*, https://github.com/matthijskooijman/arduino-lmic/blob/master/examples/ttn-otaa/ttn-otaa.ino, Online; accessed Aug. 22, 2017.

[101] M. Kooijman: *Arduino-LMIC library – LoraWAN-in-C library, adapted to run under the Arduino environment*, https://github.com/matthijskooijman/arduino-lmic, Online; accessed Aug. 22, 2017.

[102] IBM Corporation: *Low Power Networking Technology from IBM and Semtech to Help Enable Telcos to Launch New Services for the Internet of Things*, https://www-03.ibm.com/press/us/en/pressrelease/46287.wss, Mar. 11, 2015, Online; accessed Aug. 22, 2017.

[103] National Marine Electronics Association: *NMEA 0183 Standard*, https://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp, Online; accessed Aug. 22, 2017.

[104] M. Hart: *TinyGPSPlus – A new, customizable Arduino NMEA parsing library*, https://github.com/mikalhart/TinyGPSPlus, Online; accessed Aug. 22, 2017.

[105] M. Kemper: *mma-7455-arduino-library*, https://code.google.com/archive/p/mma-7455-arduino-library, Online; accessed Aug. 22, 2017.

[106] Miromico AG: *Miromico – Advanced Engineering*, http://www.miromico.ch, Online; accessed Aug. 22, 2017.

[107] Miromico AG: *FMLR TrackerTwo LoRaWAN GPS Tracker Kit*, http://webshop.miromico.ch/FMLR-TrackerTwo-LoRaWAN-GPS-Tracker-Kit_p_13.html, Online; accessed Aug. 22, 2017.

[108] Miromico AG: „FMLR Minimal AT Interface“, Miromico AG, Tech. Rep. V1.1, Apr. 2017. [Online]. Available: http://www.miromico.ch/tl_files/downloads/flyer/FMLR_AT_Protocol.pdf.

[109] Miromico AG: „Fact Sheet: FMLR TrackerTwo GPS Tracker“, Miromico AG, V1.0.1, Mar. 2017. [Online]. Available: http://www.miromico.ch/tl_files/downloads/flyer/FMLR_TrackerTwo_GPS.pdf.

[110] AerialNet Telecommunications, Ltd.: *Outdoor Enclosure, 1 Ethernet, 4 N-type holes [EZ-SOE01W]*, https://www.aerial.net/shop/product/1102/ezynet-outdoor-enclosure-1-ethernet.html, Online; accessed Aug. 22, 2017.

[111] BRACK.CH AG: *TP-Link PoE Injector + Splitter TL-PoE200*, https://www.brack.ch/tp-link-poe-injector--splitter-124132, Online; accessed Aug. 22, 2017.

[112] BRACK.CH AG: *Raspberry Pi 3 Model B, Quadcore 1,2Ghz, WLAN, BT*, https://www.brack.ch/raspberry-pi-3-model-b-419365, Online; accessed Aug. 22, 2017.

[113] BRACK.CH AG: *Kingston microSDHC Card Class 4 16GB*, https://www.brack.ch/kingston-microsdhc-card-class-99832, Online; accessed Aug. 22, 2017.

[114] IMST GmbH: *iC880A-SPI – LoRaWAN Concentrator 868MHz*, http://webshop.imst.de/ic880a-spi-lorawan-concentrator-868mhz.html, Online; accessed Aug. 22, 2017.

[115] IMST GmbH: *Pigtail for iC880A-SPI*, http://webshop.imst.de/pigtail-for-ic880a-spi-and-ic880a-usb.html, Online; accessed Aug. 22, 2017.

[116] IMST GmbH: *SMA Antenna for iC880A-SPI, WSA01-iM880B and Lite Gateway*, http://webshop.imst.de/antenna-for-ic880a-usb-and-ic880a-spi.html, Online; accessed Aug. 22, 2017.

[117] Tindie, Inc.: *IMST iC880a LoRaWAN backplane*, https://www.tindie.com/products/gnz/imst-ic880a-lorawan-backplane-kit, Online; accessed Aug. 22, 2017.

[118] Django Software Foundation: *The Web framework for perfectionists with deadlines*, https://www.djangoproject.com, Online; accessed Aug. 22, 2017.

[119] PostGIS Project: *Spatial and Geographic objects for PostgreSQL*, http://postgis.net, Online; accessed Aug. 22, 2017.

[120] Nginx, Inc.: *nginx*, https://nginx.org, Online; accessed Aug. 22, 2017.

[121] Celery Project: *Distributed Task Queue*, http://www.celeryproject.org, Online; accessed Aug. 22, 2017.

[122] Redis Project: *Redis*, https://redis.io, Online; accessed Aug. 22, 2017.

[123] Docker, Inc.: *Build, Ship, and Run Any App, Anywhere*, https://www.docker.com, Online; accessed Aug. 22, 2017.

[124] The Things Network Foundation: *myDevices Cayenne*, https://www.thethingsnetwork.org/docs/applications/Cayenne/, Online; accessed Aug. 22, 2017.

[125] Avanquest North America, Inc.: *myDevices – Simplify the Connected World™*, https://www.mydevices.com, Online; accessed Aug. 22, 2017.

[126] The Things Network Foundation: *Storage Integration*, https://www.thethingsnetwork.org/docs/applications/storage/, Online; accessed Aug. 22, 2017.

[127] IFTTT, Inc.: *Do more with Webhooks*, https://maker.ifttt.com, Online; accessed Aug. 22, 2017.

[128] The Things Network Foundation: *OpenSensors Integration*, https://www.thethingsnetwork.org/docs/applications/opensensors, Online; accessed Aug. 22, 2017.

[129] K. Holm: *Using MQTT Protocol Advantages Over HTTP in Mobile Application Development*, https://www.ibm.com/developerworks/community/blogs/sowhatfordevs/entry/using_mqtt_protocol_advantages_over_http_in_mobile_application_development5?lang=en, Oct. 18, 2012, Online; accessed Aug. 22, 2017.

[130] Vue.js Project: *The Progressive JavaScript Framework*, https://vuejs.org, Online; accessed Aug. 22, 2017.

Abbreviations

3GPP	3rd Generation Partnership Project
ABP	Activation by Personalization
AES	Advanced Encryption Standard
AOA	Angle of Arrival
API	Application Programming Interface
AppEUI	Application Extended Unique Identifier
AppKey	Application Key
AppSKey	Application Session Key
BLE	Bluetooth Low Energy
Cayenne LPP	Cayenne Low Power Payload
CLI	Command Line Interface
CSS	Chirp Spread Spectrum
CSV	Comma-separated Values
DC	Direct Current
DevAddr	Device Address
DevEUI	Device Extended Unique Identifier
EDGE	Enhanced Data Rates for GSM Evolution
ETSI	European Telecommunications Standards Institute
GFSK	Gaussian Frequency Shift Keying
GLONASS	Globalnaja nawigazionnaja sputnikowaja sistema
GNSS	Global Navigation Satellite System

GPRS	General Packet Radio Service
GPS	Global Positioning System
gRPC	Google Remote Procedure Call
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
ISM	Industrial, Scientific and Medical
JSON	JavaScript Object Notation
LMIC	LoraMAC-in-C
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LPN	Low Power Network
LPWAN	Low Power Wide Area Network
LSB	Least Significant Bit
LTE	Long Term Evolution
LTE-M	Long Term Evolution-Machine
M2M	Machine-to-Machine
MAC	Media Access Control
md5	Message-Digest Algorithm 5
MIC	Message Integrity Code
MQTT	Message Queue Telemetry Transport
MSB	Most Significant Bit
NB-IoT	NarrowBand IoT
NFC	Near Field Communication

NMEA	National Marine Electronics Association
NwkSKey	Network Session Key
OSI	Open Systems Interconnection
OTAA	Over-the-Air Activation
OTIoT	'Object Tracking using the Internet of Things'
PCB	Printed Circuit Board
PoE	Power over Ethernet
RFID	Radio-Frequency Identification
RSSI	Received Signal Strength Indicator
RTT	Round Trip Time
SDK	Software Development Kit
SIM	Subscriber Identity Module
SMA	Subminiature A
SNR	Sound Noise Ratio
SSH	Secure Shell
SSID	Service Set Identifier
TDOA	Time Difference of Arrival
TOA	Time of Arrival
TTN	The Things Network
u.FL	Ultra Miniature Coaxial Connector
USB	Universial Serial Bus
UZH	University of Zurich
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network

Glossary

Access Key A key used in The Things Network to allow access to certain functionality (devices, messages, settings) via The Things Network's APIs and integrations.

Application EUI A unique 64-bit application identifier (EUI-64 standard) used by LoRaWAN [72].

Application Key A 128-bit application key used by LoRaWAN for encryption [71].

Application Manager API An API offered by The Things Network to manage applications and devices [63].

Application Session Key A 128-bit AES encryption key, which is unique per device, and which is used to encrypt and decrypt payload data [71].

Arduino A family of microcontrollers used for hardware prototyping [43].

BeiDou A global satellite navigation system operated by China.

Data API A MQTT API for data access by The Things Network [61].

Device Address A 32-bit identifier, which is only unique within the network, and which is present in each data frame to differentiate devices from each other [72].

Device EUI A unique 64-bit end-device identifier (EUI-64 standard) used by LoRaWAN [72].

Galileo A global satellite navigation system operated by the European Union.

Gentleman's Agreement "An agreement that is based on trust and is not written down" [82].

GLONASS A global satellite navigation system operated by Russia.

GPS A global satellite navigation system operated by the United States of America. Officially called NAVSTAR GPS.

HTTP Integration A two-way communication channel with the The Things Network, using HTTP requests.

Internet of Things "The network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment" [6].

LoRa A long range wireless radio frequency technology, patented by Semtech Corporation and building the physical layer for LoRaWAN [28].

LoRa Alliance A conglomerate of partners supporting the development and deployment of LoRaWAN [31].

LoRaWAN "A Low Power Wide Area Network (LPWAN) specification intended for wireless battery operated Things in a regional, national or global network" [68].

Open Network Infrastructure Association "A non-profit association working towards a future with networking infrastructure that is more open and accessible to everyone" [54].

OTIoT The name of the research project for building an 'Object Tracking using the Internet of Things' system.

OTIoT App The user-accessible front-end of the OTIoT Platform, forming its presentation layer. URL: `https://otiot.csg.uzh.ch`

OTIoT Back-End The sub-system of the OTIoT Platform, which is responsible for reading, storing, and providing the location data as well all other data required for the functionality of the OTIoT Platform.

OTIoT Platform A web application consisting of the OTIoT Back-End and the OTIoT App. The functionalities include user management, registration and management of devices, and presentation of location data per device.

OTIoT Tag The physical object tracking device used in OTIoT.

OTIoT TTN Application The application for OTIoT registered in the TTN, which can be administrated using the TTN Console and which holds device registrations, Payload Functions and integrations.

Payload Functions The functions used in The Things Network to decode, convert and validate byte payload, or to encode payload.

Raspberry Pi A small and affordable single board computer [42].

Sigfox A French LPWAN IoT solution provider [26].

The Things Network An open, crowd-sourced, decentralized and free IoT network, currently using LoRaWAN [32].

TTN Console A web application provided by The Things Network to let registered users manage their gateways, applications and devices. URL: `https://console.thethingsnetwork.org`

List of Figures

1.1 OTIoT work package setup 2

2.1 Comparison of LPWAN, cellular network, and ZigBee [8] 5
2.2 Multilateration used by navigational satellites [41] 6
2.3 Arduino Uno [46] (left) and Raspberry Pi 3 Model B [44] (right) 7

3.1 TTN's LoRaWAN gateway locations around the city of Zurich [32] 10
3.2 Intended schematic hardware layout of the OTIoT Tag 11
3.3 Intended architecture and data flows in OTIoT 12

4.1 Relationship between different LoRa parameters [66] 14
4.2 LoRaWAN stack [65] 15
4.3 Keys and encryption in LoRaWAN [64] 16
4.4 Gateway locations in TTN's worldwide network [32] 17
4.5 TTN high-level core architecture with cardinalities [74] 18
4.6 Processing flow among TTN's components [74] 18
4.7 Location data sent by the OTIoT Tag, shown in the TTN Console 24
4.8 OTIoT Tag in unpackaged form 25
4.9 Flow chart of the location sending algorithm on the OTIoT Tag 29
4.10 FMLR TrackerTwo by Miromico, a compact LoRaWAN tracking device . . 34
4.11 LoRa gateway used for developing the OTIoT Tag 35
4.12 Tag registration dialog in the OTIoT App 40

5.1 OTIoT Tag in packaged form as testing device 42

5.2 The join procedure of the OTIoT Tag in the TTN Console 43

5.3 OTIoT Tag mounted on a bicycle 44

5.4 Alert shown to the user in the OTIoT App after leaving the configured circle 46

5.5 Collected location data shown in the OTIoT App's tag detail view 47

List of Tables

4.1 LoRa ports used in the OTIoT TTN Application 23
4.2 Hardware components of the OTIoT Tag 26
4.3 Configuration parameters of the OTIoT Tag 32
4.4 Cayenne LPP payload of the FMLR TrackerTwo (based on [109]) 35
4.5 Hardware components of the LoRa gateway 36

List of Listings

4.1 Decoder function in the OTIoT TTN Application 21
4.2 Encoder function in the OTIoT TTN Application 22
4.3 OTIoT Tag's encoding into an uplink message payload of 6 Bytes 30
4.4 OTIoT Tag's decoding of the downlink message's payload of 10 Bytes . . . 31
4.5 Uplink message (JSON) sent to the OTIoT Back-End POST endpoint . . . 38
4.6 Downlink message (JSON) sent to the HTTP Integration POST endpoint . 38
4.7 POST request sent to Application Manager API to create a new device . . 39

5.1 Serial output of the OTIoT Tag's location sending algorithm 45

Appendix A

OTIoT Tag Deployment Manual

The following steps have to be taken in order to deploy a new version of the Arduino source code to the OTIoT Tag.

1. Download the Arduino IDE from `https://www.arduino.cc/en/Main/Software`.
2. Install the Arduino libraries `arduino-lmic-master`, `tinygpsplus`, and `MMA_7455` according to the manual on `https://www.arduino.cc/en/Guide/Libraries`. The libraries' .ZIP files can be found on the CD delivered together with this Master thesis.
3. Open the OTIoT's Tag Arduino source file named `otiot-tag.ino` in the installed Arduino IDE.
4. Connect the Freeduino Uno to your computer using the Mini-USB cable.
5. Select the Freeduino Uno's USB port (marked as 'Arduino/Genuino Uno') using the Arduino IDE's toolbar: Tools -> Port.
6. Click on the right arrow button ('Upload') to compile and upload the Arduino source code to the OTIoT Tag.
7. To see the serial output of the OTIoT Tag, click on the button with the magnifying glass symbol ('Serial Monitor') and configure the Baud rate to 57 600.
8. The code is started whenever the OTIoT Tag is re-connected to power or whenever the code is uploaded to it.

Make sure the DevEUI, AppEUI, and AppKey are the same in the Arduino source code as in the TTN Console. You should see the OTIoT Tag's joining procedure in the 'Data' section of the TTN Console. No messages can be seen if (1) no gateway is within range or if (2) the OTIoT uses incorrect OTAA credentials.

Appendix B

Contents of the CD

The contents of the compact disk, which is delivered with this Master thesis, consists of the following files:

- `Zusfsg.txt`: Plain text version of the German abstract.
- `Abstract.txt`: Plain text version of the English abstract.
- `Masterarbeit.pdf`: PDF file of this Master thesis submission, including all appendices.
- `/latex`: Directory containing the LaTeX source files of this Master thesis.
- `/references`: Directory containing the referenced PDF files.
- `/otiot-tag`: Directory containing the OTIoT Tag Arduino source code file.
- `/otiot-tag/libraries`: Directory containing all required Arduino libraries for the OTIoT Tag, as .ZIP files.
- `/otiot-platform`: Directory containing the OTIoT Platform source code files.