NarrowBand IoT

Testing of early deployment

Henning Håkonsen

Thesis submitted for the degree of
Master in Network and system administration
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2018

NarrowBand IoT

Testing of early deployment

Henning Håkonsen

NarrowBand IoT

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Contents

Listings

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisors, Ola Martin Lykkja and Yan Zhang, for their support. In addition, I would like to thank Q-Free for giving me the opportunity to work with Ola Martin, as well as supplying me with the necessary equipment to fulfill my work.

A special thank you to Telia which let me perform tests on their network. This gave the thesis an additional dimension and improved the results.

I would also like to thank my family, friends and especially my girlfriend, for supporting me through the last two years completing my master's degree.

Abstract

Today there are approximately 30 billion smart devices in the world. Analysis predict exponential growth going forward with 50 billion smart devices by 2020. There is a demand for a new technology which enables communication with sensors and other low powered devices. This thesis is a product of the work and research on a set of technologies suited for this communication, with an emphasis on NB IoT.

Reading notes

Please visit henninghaakonsen.me for the latest copy of the thesis, along with the related research and results. The web application is provided to support my thesis with extra reading material as well as being a tool for my tests. All occurences of the term 'coverage' in all figures related to this thesis should be considered as 'signal power'.

Chapter 1

Introduction

1.1 Goal

The goal of this thesis is to introduce you to Low Power Wide Area Network (LPWAN) technologies and explore the specifications related to one of them, NarrowBand IoT (NB IoT). There are several claims to NB IoT - battery lifetime over 10 years, indoor and underground coverage and low cost. Through a cooperation with Q-Free, Telenor and Telia, I was able to do hands-on tests of NB IoT. The interesting part of this thesis is that I was one of the first to test NB IoT in Norway and it lead me to a motivational process, but also at times frustrating. There are several papers on how NB IoT has great power saving features, referring to the 3rd Generation Partnership Project (3GPP) specification, but there are few discussing how they can be achieved in practice. Several factors will influence the power usage, such as environmental changes, downtime in the network, network configuration and software complexity. I will show you how these factors impact the power consumption and give an overview of the best practice guidelines.

1.2 Motivation

Easily accessed information, connected to all parts of society is a huge motivator for Internet of Things (IoT) and LPWAN. In the future all things will be connected to the Internet, enhancing applications. The definition of IoT is a group of physical devices, for example, vehicles, monitoring systems, watches and so forth, forming a network. The possibilities this network introduces are incredible, but the path towards this goal has not been easy. There have been attempts of similar networks, but they are usually too fixed or expensive and the competition has been more of an obstacle than an advantage. With several emerging technologies suited for low powered devices, Cisco has made an illustration of the expected growth of devices[1.1 on the following page].

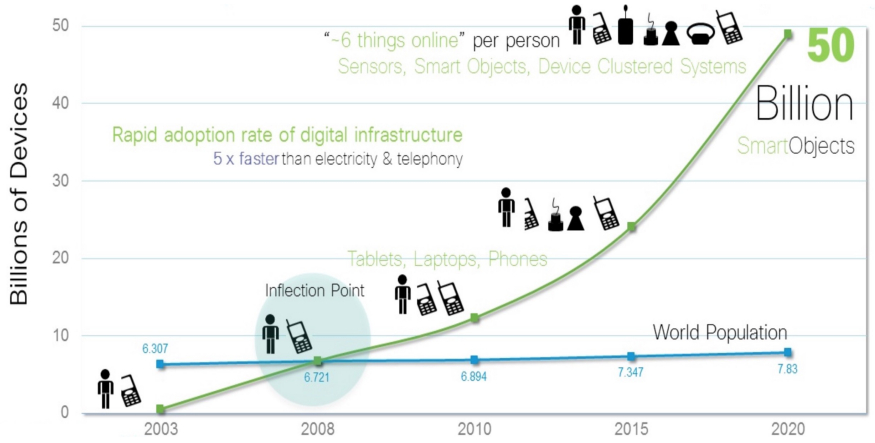
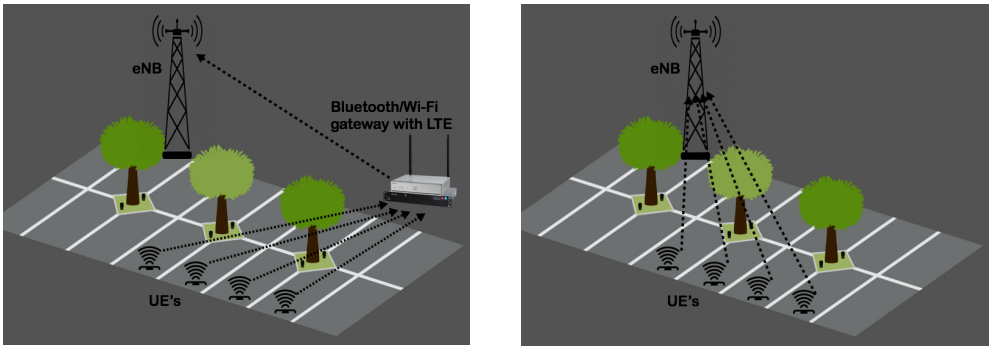


Figure 1.1: Cisco's expected growth of devices based on an article anticipating the growth of the population by The United Nations [13][12]

The payload sent from IoT devices is usually small, typically 100 bytes or less. However, by using Long-term Evolution (LTE) for Machine-To-Machine (M2M) communication on the current infrastructure leads to high load with all the signaling required. Hence, a simplified technology would allow less signaling and load on the telecoms infrastructure, while also lowering monthly fees as a consequence. Figure 1.2 on the next page, presents two setups. Up until now, IoT devices have communicated directly or indirectly with Internet like alternative A. The typical scenario has been that the User Equipment (UE) communicate with a more complex device, often a wireless unit of some kind and this device, or an additional device, communicates with the Evolved Node B (eNB) or via a wired connection. While this is an improvement and a step towards a future of connected things - the current mobile technology does not support the expected growth in this area. In alternative b however, the UE's can communicate directly with the eNB which reduces complexity, cost and maintenance.

Price is always an important factor when choosing a new product. The typical cost of a mobile subscription with LTE and 5-10GB of data per month cost between 20€ and 50€, supporting 50-100 devices. The cost of a subscription over NB IoT will according to Telenor and Q-Free be around 0.2€ per UE. In addition, hardware and installation costs are reduced with NB IoT. The aim is that an NB IoT enabled device will cost around 60€. A similar LTE implementation would require a component that does LTE for embedded devices, which costs around 500€. The calculation 1.1 on the facing page, of the cost of 300 devices from a parking sensor use case shows that NB IoT outperforms the current solution. On the left-hand side, you see LTE specific additions related to installation and monthly fees from the LTE embedded device. The monthly fees and maintenance come from the rent of the hardware position. When using



(a) Setup with private base station (b) Setup with NB IoT

Figure 1.2: Setups pre and post NB IoT time[20]

the LTE embedded device Q-Free needs to rent a space for it and pay for maintenance and power for that particular place. Note that this is just an outline of one example and certain parameters may change depending on hardware, installation company, and network provider.

Sensors per base station (LTE)	100
Subsciption cost - monthly (LTE)	€ 50
Installation cost - one time (LTE)	€ 1 500
Montly fees and maintenance (LTE)	€ 250
Number of years	3
Number of sensors	300

	Setup with private base station	Setup with NB IoT
One time cost		
Cost of sensor	50	60
Installation cost per sensor	5.00	0
Montly cost		
Subscribtion	0.50	0.2
Subscribtion over 3 years	18.00	7.2
Fees and maintenance	0.83	0.3
Fees and maintenance over 3 years	30.00	10.8
Total cost per sensor	€ 103	€ 78
Total cost 300 sensors	€ 30 900	€ 23 400

Table 1.1: Cost comparison between NB IoT and LTE with of 300 devices.

In addition to lower cost and lower complexity, there is a demand for increased coverage. Vehicles, monitoring sensors positioned several meters underground and devices located in packed cities are dependent of extreme coverage. NB IoT will add 20 dB to the link margin, which will enable such devices to operate while holding the power usage to a minimum.

Another motivator for IoT is the expected easiness of the setup. With today's technology, you have to use sensors communicating over Bluetooth or Wi-Fi to a common gateway which is connected to the Internet. With a NB IoT device you will be able to connect directly to Internet, which will encourage many people to engage in this technology, perhaps people without development experience as well. This will be an additional cause of growth in the number of IoT devices and yet another reason why there is a demand to properly implement LPWAN.

The idea of some of the new LPWAN is to use the same hardware as LTE and if the customer wants it they will also support software to enable

an IoT platform. The most promising technology for M2M communication is NB IoT and is enabled with a LTE core network. I will discuss how to deploy NB IoT in section 2.2 on page 18. This new technology supports an extreme amount of devices as well as them being in a secure environment. Security is a big concern when discussing general network related topics and I will introduce you to some security measurements provided by NB IoT.

1.3 Current status

1.3.1 Internet

The main idea of the Internet today is quite similar to the past, but the scale and reach has outgrown what anyone thought could be achieved. In 2015 Google's data centers achieved high speed transfers up to 1 Petabit. "According to Vahdat, that is enough bandwidth for more than 100.000 servers to exchange data at 10 Gbps each, or transmit all scanned contents of the Library of Congress in under one-tenth of a second"[43]. The reason for these data centers is the use of online resources. Offices, as well as home users require more bandwidth and reliability of the services provided from e.g. Google, entertainment, data storage and so forth.

1.3.2 Mobile networks

Norway's GSM network came online in 1993 [24] and Norway has pushed the technology to reach higher standards from this time. The evolution of wireless radio technology has usually been focused on making it faster, while not prioritizing battery lifetime and simplicity. However, because this technology is used in mobile devices, it is for most users good news. LTE offers high Up Link (UL) and Down Link (DL) speeds and good coverage. One problem with LTE and older wireless technologies is that it consumes a lot of power. People are experiencing high speed transfers to their mobile devices, but at a cost draining their batteries. In the future, power efficiency has to be one of the main features of LPWAN established. This is especially the case for sensor networks as these devices need to be connected to a mobile network. Such sensors are often established in remote locations without steady power and using LTE would give the devices short lifetime.

1.3.3 A closer look at LTE

A LTE network consist of some particular nodes for the network to connect to UE's and Internet. This architecture is called Evolved Packet Core (EPC) [46] and this section will give you a get a better look at some of

the attributes of LTE and the parts which make the EPC network. LTE was introduced in release 8 of the 3GPP in 2008. In a summation by Ericsson, they state some facts about LTE from the initial release. LTE from release 8 supported 100 Mbps DL and 50 Mbps UL (Not peak speeds), reduced latency (down to 10ms) and was cost-effective to set up[18]. As of release 10 by 3GPP (year 2011) the speeds were at astonishing 1 Gbps DL and 500 Mbps UL, however because of overhead in the network it is not suited for low powered devices running on batteries.

1.3.4 Introduction to key features of Evolved Packet Core

LTE is composed by 5 components which interact with each other. eNB, MME, HSS, S-GW and P-GW constitutes EPC. See outline of LTE topology in figure 1.3.

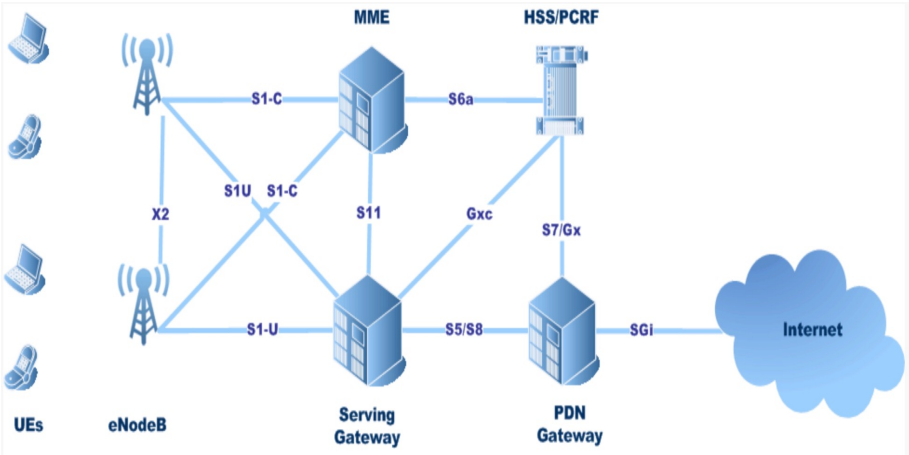


Figure 1.3: Evolved Packet Core overview [35]

eNB

The Evolved Node B (eNB) is what most people refer to as the base station of a mobile network. It is responsible for distributing signal to the users and deal with UL and DL data. The eNB can be set up to suit different scenarios by adding frequency bands. For LTE, most network providers use 800, 900, 1800 and 2100 Mhz, while for NB IoT most companies will use 800 Mhz because at this frequency the signal will reach longer.

MME

Mobile Management Entity (MME) is the main provider of signaling in LTE. MME is connected to eNB, HSS and S-GW, through S1-MME, S6 and S11 interfaces. It is in charge of authentication in cooperation with HSS and terminal-to-network negotiation.

HSS

Home Subscriber Server (HSS) is the main database for information in the network. It is a joint service originating from Home Location Register (HLR) and Authentication Center (AuC). HSS keeps information about subscriber information, that being user identification, addressing and profiles for a subscriber. Profiles describe how the network should perform for this special user and may include parameters like Quality of Service (QoS) and special modes, or states for a subscriber. HSS also holds information about authentication between the network and UEs.

S-GW and P-GW

Serving Gateway (S-GW) and Packet Data Network Gateway (P-GW) deal with the user data plane and transports IP packets from EPC to external networks. The S-GW maintains paths from eNBs to P-GWs.

1.4 Future mobile networks

For the past ten years there has been a rise in sensor activity. In the evolution of IoT devices, some has ported to LPWAN. However, as stated in the motivation, IoT devices rely on good coverage and stability. Many people think of home electronics when discussing IoT, but this is not the main goal of LPWAN devices. Oslo's public transport operator (Ruter) has payment cards communicating with ticket readers over RFID, which in turn communicate with Internet. The Norwegian transport agency uses Dedicated short range communication (DSRC) in their tolling stations to communicate with cars, which is a standard developed by CEN TC278 WG1[16]. The following sections gives examples of applications related to future mobile networks.

1.4.1 Applications

Already back in 2009, a forum called Organisation for Economic Co-operation and Development (OECD) discussed ways to take advantage of sensors to support a green growth [37]. This forum has members from all over the world, including Norway. The idea is that sensors can surveil and monitor emission numbers as well as act upon them. OECD came up with a group of applications which were important to investigate. Some of them are smart grids and energy control systems, smart buildings, transport and logistics, agriculture and other general industrial applications. Together these fields combine most of the energy use today and with smarter systems, power consumption can be reduced.

With this in mind, it is clear that many of the fields OECD discussed has been implemented, or is in the deployment stage. However, the current status of industry activities related to these fields are running over 2G/3G/LTE networks. These networks provide average coverage, cost and power consumption, hence with billions of devices enrolling, this is not a sustainable model.

GSM Association (GSMA) has made an overview of the most applicable areas of use. Figure 1.4, presents some bullet points for each area, and the next section will introduce you to some of the areas.

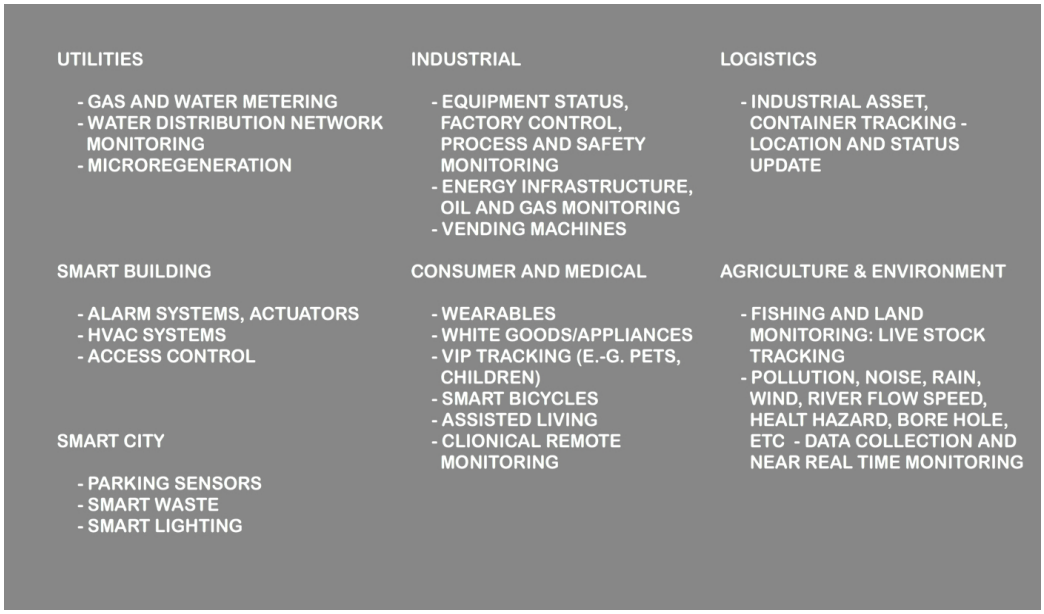


Figure 1.4: LPWAN applications [23]

Utilities and smart cities

This domain resolves to what is investigated in this thesis. Cities are getting smarter by the day and LPWAN can be the answer to a large sensor network for monitoring and managing cities. Intelligent Transportation Systems (ITS) is an initiative to control and manage traffic. The system provides communication between cars, trucks and sensors. The sensors are mounted in traffic lights, signs and other objects known in the transport field. The connected devices can adopt to the environment and help society in several ways. Emergency units can set a route to an accident and the overlaying ITS system will clear the way remotely to enable the emergency unit to quickly arrive at the location.

Examples of other usage areas of sensors in cities are waste areas, power monitoring and CO_2 monitoring. Operations can heavily benefit from more information, as well as automated procedures. There are many use cases where monitoring can be difficult and costly. With simple and

cheap devices one could monitor hundreds of different things. As a result the work effort could decrease and the processes can be more efficient. A good example is garbage disposal. If garbage containers were fitted with measuring sensors, the control system can be notified when it is time to empty the garbage. The use case for this will probably be cities, bigger companies and industrial sites, however it might be used for private residents as well.

These devices need to be cost efficient since they will be deployed in large scale. In 2014 a power supplier in Norway called Lyse began installing smart power meters in over 140 000 households [30]. These devices communicate with the mobile network in the area over a proprietary standard. While it is important to explore new solutions, this is one of many examples where standardized solutions will greatly outperform proprietary solutions.

Parking - a realistic use case

Big cities are investing in smart parking even though cars may be banned from the city centres in some countries. People will still prefer traveling with car and facilitating for efficient parking at a nearby location in these cases is crucial. These parking lots can be managed with sensors communicating over LPWAN. For Q-Free's next generation parking sensor, they are mainly focusing on NB IoT. The sensor is housed in by a small plastic case which will be mounted in flush with the ground. It withstands extreme forces and detects cars by magnetometer and pulsed Doppler radar. The housing is the most expensive part of the sensor, being able to withstand rough conditions for over 10 years. The idea is that the sensor is drilled into the ground, and will communicate over NB IoT. The UE will send status of the occupancy of the parking spot with a fixed interval, as well as reporting parking events when they occur. See 1.5 for a photo of the parking sensor from Q-Free.

As of 2016 the parking sensors communicated to a common gateway (LTE embedded device), a device which communicates with the sensors using a proprietary narrowband communication technology in the ISM band, and to the Internet over LTE. This works fine, but as you might have realized, the cost of the devices themselves and managing them are lowered significantly with LPWAN like NB IoT. Q-Free has chosen to go with NB IoT since this is the most promising technology providing the necessary requirements for their application. The communication between the sensor and the server is outlined in figure 1.6 on the facing page. The sensor sends data to the eNB which in turn sends the data to the IoT platform. If a server has been authenticated within the platform the message is sent to the server. The server receives the packet and can display the message however necessary.

Figure 1.5: Parking sensor from Q-Free[38]

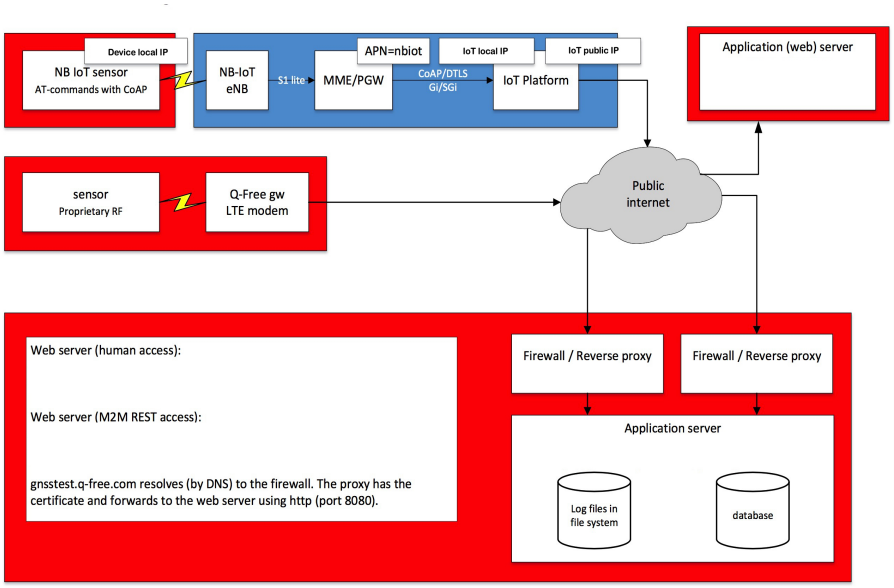


Figure 1.6: Outline of sensor communication [38]

Logistics

Today parcel tracking is a huge success and some companies are using sensors in their trucks for real time tracking. Using LPWAN networks will reduce cost of the devices and with long battery lifetime the sensors can be reused. In addition, the coverage for tracing the parcels will have to be great. If you want real time, or near real time tracking, the sensor

would have to send position data often and this will drain the battery substantially. There are some pros to use LPWAN networks for this kind of information collection, but this is probably not the biggest IoT area.

Industrial

Industry is a wide area and covers sites like gas stations, construction sites and factories. GSMA also includes vending machines in this category [23]. With sensors communicating over LPWAN industry stations can be more automated and it is easier to surveil the systems. Downtime on such systems means lost revenue and efficiency. Low cost sensors which are wireless will hopefully have good coverage and uptime so that these systems run better. This is probably one of the areas where these sensors will be used more frequently since these systems needs continuous monitoring. LPWAN will also solve coverage issues related to industry use. As mentioned good coverage would prevent downtime, but it is also important to notice where some of these sensors will be placed. Factories and underground are some of the locations where industry is often located and therefor coverage is especially important for industry usage.

Consumer

Many people think of consumer products when talking about IoT. There is growth in smart watches and smart wearables, and the biggest problem with these devices today is the battery life. This is mainly due to the extreme cost of communicating either with your mobile telephone over bluetooth or over LTE. It will be interesting when these watches will use LPWAN. Some wearables, such as your watch may use a standard with higher bandwidth than e.g. NB IoT. For normal use, one would probably prefer CAT-M which offers bandwidth up to 1Mbit/s and could possibly provide music playback and phone calls.

Another interesting use case is smart homes with better alarms, water and gas metering, light control and so forth. While this usually has communicated over Wi-Fi and proprietary Industrial, Scientific and Medical (ISM) solutions, there is a potential market where tradition Wi-Fi is not possible or preferable, hence LPWAN can be used.

Medical

A potential step into a healthier society is for medical clinics to be able to monitor and give realtime consulting to their patients. The patients device could automatically uploaded real time data to their doctor, enabling them to uncover symptoms or deceases at an early stage and by this increase the average lifetime. One can also collect a lot of data for patients with a special decease and research for a cure. For this to be realistic one really

needs LPWAN since these sensors will have to be cheap and the stability has to be good.

We are going into an era where the number of elderly will increase heavily worldwide and should seize the opportunity to use this technology to take care of the elderly. A person which needs attention, but can live alone, would prefer an electronical pill dispenser and a system to communicate with the care center. The person could also raise an alarm with this system to alert the care takers. In this way the person will probably have a better life and the cost will stay lower.

1.5 Developing wireless technologies

1.5.1 The general idea

As mentioned in the motivation the general idea of new wireless technologies is to reuse the current LTE infrastructure. One may also use GSM, as the current advantage of this network is that coverage in some areas are better at the moment. However, GSM and other 2G technologies are being phased out and in a couple of years LTE will be deployed at all locations which GSM covers, meaning better speeds, UE density and coverage. Coverage and power consumption are key features of developing wireless technologies and in combination with ease of use it is the reason why some technologies stick while others fade away. Ease of use is a requirement for M2M communication since the amount of data per packet is relatively small. For some of the technologies, guarantied delivery is not important, but it is important to acknowledge that fast delivery will be a requirement for some applications which uses real time monitoring. NB IoT will not suffice for these applications and it is necessary to consider using technologies in parallel or speed up the transmission of e.g. NB IoT. In the future this may not be a problem considering the pace of wireless evolution, but nevertheless it is important to not suppress the issue.

The following subsections introduces the current contenders in the LPWAN market.

1.5.2 eMTC/LTE-M1 and NB IoT

enhanced Machine Type Communication (eMTC) was standardized in the 12th release of 3GPP and updated with NB IoT in the 13th release. It is meant as an high bandwidth alternative to NB IoT and is often referred to as LTE-M1 or CAT-M.

Both standards are implemented using the current LTE infrastructure, but they differ in signal power, bandwidth and the number of bands used in an eNB. Figure 1.8 on page 13 shows that LTE-M1 provides bandwidth

up to 1Mbit/s while NB IoT peakes at around 200Kbit/s (rates from the release they were standardized). The coupling loss of LTE-M1 is said to be 12dBm better than LTE, and NB IoT 20dBm. The details about how NB IoT uses the current infrastructure will be coverd in section 2 on page 15.

1.5.3 LoRa

LoRa [11] is the most popular proprietary solution for enabling IoT devices. It is currently deployed in many cities and uses a proprietary solution. The devices which has a LoRa chip uses RF or WiFi to communicate with a common gateway, typically a "cell" tower. The current range is up to 15km, but in dense areas the range only covers 2-5km. The bandwidth is low, but in light of the messages being passed to the gateway that is fine. Figure 1.7 presents the overlaying infrastructure of a LoRa network. It uses many of the same features as mentioned in the motivation where the devices communicate with a middle box (concentrator/gateway), which in turn communicates with Internet over 3G or ethernet.

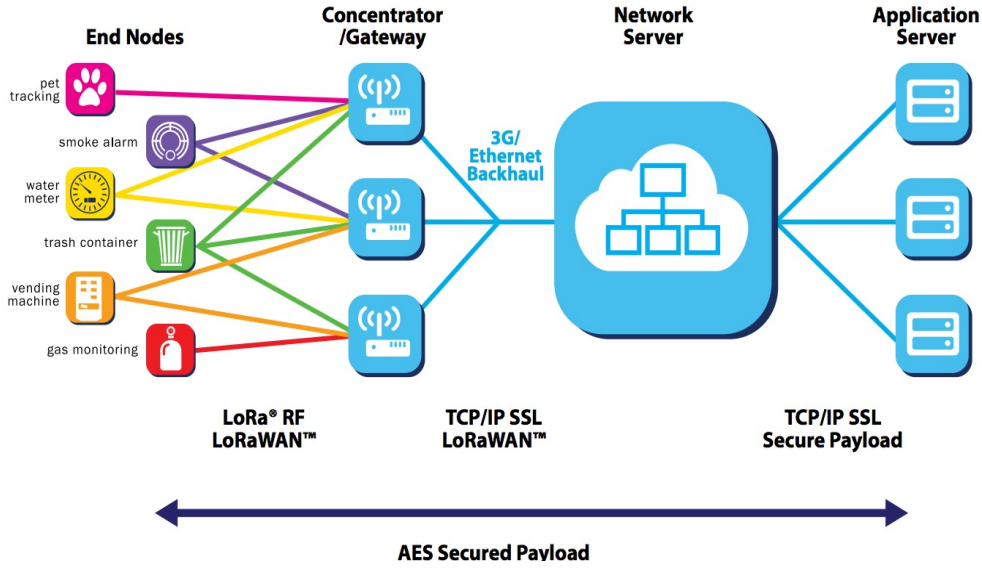


Figure 1.7: LoRa overview [11]

The LoRa solution provides LPWAN at relatively low cost and gives average coverage compared to NB IoT and LTE-M1. However, LTE is being deployed globally and if the coverage is good enough it will be more costly to provide LoRa networks as well as NB IoT or LTE-M1 networks.

	LoRa	GSM (Rel.8)	EC-GSM-IoT (Rel.13)	LTE (Rel.8)	eMTC (Rel.13)	NB-IoT (Rel.13)
LTE user equipment category	N/A	N/A	N/A	Cat.1	Cat.M1	Cat.NB1
Range Max. coupling loss	<15km 155dB	<35km 144dB	<35km 164dB	<100km 144dB	<100km 156dB	<35km 164dB
Spectrum	Unlicensed <1GHz	Licensed GSM bands	Licensed GSM bands	Licensed LTE bands In-band	Licensed LTE bands in-band	Licensed LTE in-band guard-band stand-alone
Bandwidth	<500kHz	200kHz	200kHz	LTE carrier bandwidth (1.4 – 20MHz)	1.08MHz (1.4MHz carrier bandwidth)	180kHz (200kHz carrier bandwidth)
Max. data rate*	<50kbps (DL/UL)	<500kbps (DL/UL)	<140kbps (DL/UL)	<10Mbps(DL) <5Mbps(UL)	<1Mbps (DL/UL)	< 170kbps (DL) < 250kbps (UL)

*Max data rates provided are instantaneous peak rates.

Figure 1.8: LPWAN technologies [36]

Chapter 2

A dive into NarrowBand IoT

I will focus on a general approach towards NB IoT as technology, and test both Telenor and Telia's solution. Section 5 on page 53, discuss how and where the tests were conducted as well as examples and results.

Telenor started their NB IoT network test late 2016, with hardware mainly from Huawei. The goal of the test was to set up a NB IoT network and test the communication with Q-Free's parking sensors and run own tests before releasing the network to the public. At the same time, Telia had just announced that they had deployed their NB IoT network, but it was not yet production ready. At the time of delivering this thesis, neither Telenor or Telia had NB IoT in production. Without any reference to a specific date, they only said that the network would be deployed within a short period of time which indicates that the networks were not in production at the time of the tests, but at a stable state.

NB IoT is the most promising LPWAN solution and the focusing technology in this thesis. The standard is made by 3GPP and European Telecommunications Standards Institute (ETSI), and was launched late summer of 2016 with the 13th release of The Mobile Broadband Standard. The technology takes advantage of the current LTE infrastructure in a very efficient way. The deployment requires no extra hardware, so the software necessary can be implemented into current hardware. The following sections introduces the structure of NB IoT and the design requirements. I have not emphasized security in this thesis and it will only be mentioned in the appropriate sections.

2.1 System design

In a system design process it is important to analyze the use of the new technology. It is crucial to understand how people and the industry will use NB IoT. A good example of a standard which is causing problems today is IPv4. When this standard was introduced, no one could predict the growth of online devices and for this reason NB IoT has some specific

system design features which hopefully will cover its use for foreseeable years ahead.

- Link budget improvement
  The sensors and devices in mind when deploying NB IoT need better signal power than the average phone or LTE modem. As mentioned these sensors will be used underground, indoors and areas normally without cell reception. It is therefor important that the signal is suited for this use. The goal is to reach 20dB extended signal power compared to LTE. Not only does this extend the range that these devices can be deployed, but they can emit transmissions with lower power.

- UE density
  Cisco predicts that each each household has on average 4 devices[12]. According to Oslo council there are approximately 332 568 households per 1.1.2016 [27], meaning that the device pool would be around 1.3 million devices only for house holds. Taking into account that these devices will be broadly used by the industry as well, the device density will be very high. To put this into perspective we can investigate the device density of one GSM channel deployed in the 900 mHz frequency band. For each GSM cannel (200 KHz) there are 8 time slots, giving 8 concurrently connected devices. In one cell tower there are approximately 8 channels, and they are often split between mobile providers. Assuming that all of them belong to one provider, one cell tower can support up to 64 devices concurrently. One GSM cell tower will provide coverage for the surrounding 1km, and since cell towers to close to each other will cause interference only one cell tower can provide coverage for 1km in radius. LTE has much more resources and can provide coverage for more devices, using time and wave division multiplexing.

  There is however a presumption to why NB IoT can support so many devices per carrier. In the specifications one NB IoT channel can support more than 50 thousand devices and keep in mind that NB IoT can be deployed in one GSM channel, which only supported 8 devices. The way this is supposed to work is due to the infrequent updates from the devices. One device might send a message every second hour and another might even send messages only once every day. One scenario where this might cause problems is if every programmer/company sends updates on specific times. Typically one would want updates at the hour marker(e.g. 14:00, or 15:00). If thousands of devices do this at the same time, congestion and latency will definitively affect the performance for the devices.

- Low complexity
  A key feature for the devices supporting NB IoT is low complexity

level. The new standard is less intricate than ordinary LTE which means that the devices can be less complex, resulting in cheaper hardware. This is an important factor for success since these devices need to be cheap to be broadly deployed. The complexity in this technology resides in the core mobile network for it to support many devices. Other than this, the technology supports average speed and latency, perfect for monitoring purposes.

- Low power
  The power consumption of the communication needed for LPWAN need to be low for the estimated battery lifetime of a device to be kept. The goal is over 10 years of lifetime, and since these devices probably will be very cheap, it might be cheaper to change the device instead of changing the battery.

  Low complexity and better coverage will help the device to consume less power, but it will not enable the device to operate for 10 years. The system design includes a special operation mode, called Power Saving Mode. I will elaborate how this key feature works in section 2.4.6 on page 24.

- Latency
  For most applications applied to this new technology, latency is not a key feature. One would like the device to send frequent, or infrequent messages to a server or another device, but the time used is not important. However, what is actually a long time? A ping request on a normal wired connection today uses around 5-100ms, and even less for fiber connections. On mobile networks like LTE, the usual ping time is on average higher than on a wired connection, but usually it does not exceed 100ms. NB IoT aims at a peak latency of 10 seconds. In comparison that is 100 times longer than a normal WAN connection, but it should suffice for most applications using LPWAN. The latency could probably be lowered, but it might interfere with other design goals, such as device density.

- Security
  A big issue in Internet today is security, specially DDoS attacks. These attacks are more frequent and is an attack form which enables thousands of machines to continually spam one or several IP addresses, which in practice disables the application. In the fall of 2016, the DNS service provider "Dyn" was down because of a DDoS attack. Being a big DNS server, Dyn's downtime impacted major Internet platforms in the U.S. and Europe[54]. Since an attack like this needs a huge device pool it is likely to think that IoT devices are a potential tool for hackers. If hackers could access millions of devices with Internet access, they could easily perform global DDoS

attacks which would halt our infrastructure, in turn disabling many of us to do our jobs.

With this in mind, the engineers and designers of NB IoT needed to consider security issues like this. The concept 'IoT platform' is introduced in NB IoT, and this platform can be used to contribute to security measures in cooperation with MME and HSS. The system requirement of NB IoT is that no one should be able to directly access a NB IoT device. One additional security feature in general with mobile networks is SIM cards. Like cell phones, NB IoT devices will be fitted with a SIM card, which will enable the core network to authenticate the device, and visa versa. The SIM card has a subscription connected to the service provider, hence making a strong authentication scheme.

The current design is a general approach to the requirements of low-powered devices. Section 1.4.1 on page 6, presented several applications related to LPWAN and [10] has investigated traffic related issues to M2M communications over LTE. The following list is a summary of the most essential traffic patterns in M2M communications[10].

- UL traffic dominated, because most applications today are focusing on data gathering rather than data control.
- Traffic generated more uniformly through the day by M2M communication, compared to normal mobile communication.
- Traffic can be periodic, which is related to meatering applications which generate data at specific intervals.
- Some applications react to events and generates larger volums of data, hence traffic can be bursty.
- The mobility of M2M devices is lower than conventional mobile devices. UE's will for the most of the time remain at the installation site. However, for some applications like wearables this is not true.
- For many applications QoS will differ. Alarm systems requires higher uptime, better security and increased TSR, while wearables requires excellent handovers between different eNB's and reliable DL data control.

2.2 Deployment

There are numerous articles describing the physical layer of NB IoT, so it is not emphasized in this thesis. For further details about the physical

layer, I recommend reading "A Primer on 3GPP Narrowband Internet of Things"[9].

There are currently three ways to deploy NB IoT and the available deployment modes are standalone, in-band and guard-band - see figure 2.1 for illustrative definition. Standalone operation uses a dedicated GSM/LTE channel, which allows NB IoT to utilize previously used GSM frequencies with their channel bandwidth of 200 kHz, thus allowing for 10 kHz guard bands below and above the NB IoT channel.

In-band operation utilizes bandwidth within one LTE carrier, and allocates one Physical Resouce Block (PRB) of a LTE channel. This is the most technical and advanced solution and requires more work by the network provider. The reason for this is because of interference between the LTE and NB IoT sections in the carrier.

Guard-band operation makes use of the bands not in use by LTE due to interference between LTE carriers. This in-between section is called a guard-band and guards the carries for interfering with each other. This band can be used by NB IoT and is a solution where one would want to utilize all resources of a cell tower.

A NB IoT UE is not required to be configured specifically for either deployment schemes. The UE searches for a carrier on a 100kHz raster, which implies that when deploying in-band mode the anchor carrier can only be placed in certain PRB's[9].

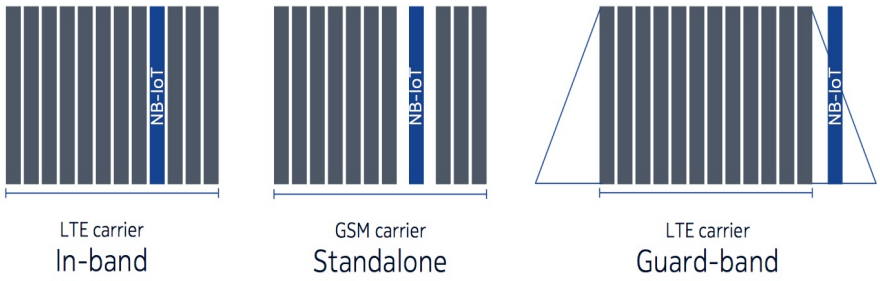


Figure 2.1: NB IoT deployment overview [36]

The choice between in-band, guard-band and standalone operation is based on the network providers frequency budget in the given area and prioritisation of traffic. The following paragraphs introduces which operation mode Telenor and Telia has used.

Telenor has deployed their NB IoT network in stand-alone operation mode, which means that they are using one dedicated LTE channel[38], hence the network does not suffer from noise from LTE and vice versa. The results showed that the stability of the network was good and performed close to what the specification dictates.

Telia has deployed their NB IoT network in in-band operation mode which means they occupy one PRB in an existing LTE channel. As discussed earlier this results in potential frequent interference between NB IoT and LTE devices. Using Telia's network revealed a slightly more unstable behavior which might be related to the fact that they are using in-band operation mode. On the other side, this setup is closer to what one expect the situation to be in a production network with other users, hence the performance difference is arguably logical.

2.3 IoT platform

IoT platform can be rewarding for both customer and network providers. It may enhance the security of the communication. However, in order to utilize the full potential of an IoT platform the network provider will have to open the content of the data and do a repacking of the data. In my opinion, this is not as secure as an end to end transmission. Another motivation for a network provider to promote the IoT platform is that if they have access to the customer's data they can perform analysis on it and give customer based subscriptions or potentially sell your information. A good example of usage of these kinds of data is the possibility to stream music for free, which was enabled by Telenor and Telia in 2018. The networks has to view the data to know that the data is related to music, i.e. Spotify, Tidal, to give the customer this data for free. Likewise the network providers will have access to the data in the IoT platform which is not preferable for many customers. However, if your application sends non-confidential information and you want the application up and running quickly the IoT platform might enhance your application. There are always tradeoffs to using an abstract implementation, like high order programming languages or something like an IoT platform.

2.4 Introduction to energy saving

Power saving is an important feature of M2M communication. Several techniques are used to reduce power consumption, and Discontinuous Reception (DRX) and Power Saving Mode (PSM) are two of them. DRX has existed for 20 years and is implemented in LTE, while PSM was introduced with the development of LPWAN. These two modes, together with paging introduces an improved network, especially suited for sensors. In figure 2.2 on the facing page you can see an outline of the communication process for a device. Radio Resource Connection (RRC) connected mode is the mode where the device actually can communicate with the network. The following procedures shows a possible time convergence graph for a device. The next sections explain the techniques

used, as well as the specific time periods.

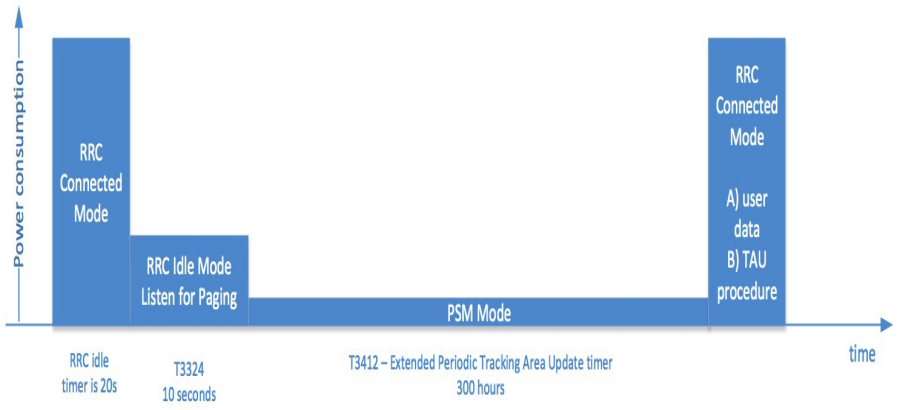


Figure 2.2: NB IoT transmit overview [38]

2.4.1 Prerequisites

To get a good understanding of the results, it is important to look closely at the outcome of the results and understand the meaning. The following sections describe details about how NB IoT works and what the specifications are saying. As stated, I will mostly focus the tests on signal power and power consumption, which is closely related. Signal power is measured in Decibel / referenced to milliwatts (dBm) and provides a relation between the distance between the eNB and the UE, and the transmit power of the UE. In theory, a device will transmit with the relative strength according to the signal power, so that the signal is received at the eNB with the same level.

I will use mW throughout this thesis and the equation, 2.1, shows the relation between Decibel / referenced to milliwatts (dBm), Milliampere (mA) and Milliwatts (mW).

$$\begin{aligned} X \quad dBm &=> 10^{X \quad dBm/10} \quad mW \\ I \quad mA &=> (I \quad mA * V) \quad mW \end{aligned} \tag{2.1}$$

NB IoT operates between $-40dBm$ and $+23dBm$, where theoretically $-40dBm$ equals $10^{-4} = 0.0001mW$ and $+23dBm$ equals $10^{2.3} = 200mW$. The chip's manual states that $-40dBm$ equals $74mA * 3.3V = 244mW$, while $+23dBm$ equals $220mA * 3.3V = 726mW$.

2.4.2 RRC connected mode

This mode is used when the device wants to communicate with the network, either UL or DL. The time spent in RRC connected mode can

vary, but the default NB IoT timer is 20 seconds. In this mode the device will use a lot more power, hence moving away from this mode as soon as the transmit or receive process is finished is important. When the chip is in active/connected mode it uses $6mA * 3.3V = 0.0198mW$ [52].

When the transmit operation is finished the chip will stay in connected mode until the time period ends. There is an optional flag, Release Assistance Indicator (RAI), in the uplink datatransfer frame which puts the device into PSM directly after the transmit operation. The network needs to support this action to give the UE permission to sleep. It is the network that decides what the UE should do according to the network and the device's status. Given that the network supports RAI and this is used the battery life will be extended, since the UE only sends data for a short period rather than waiting for downlink data which in many applications is not needed.

If RAI is not set the device will stay in RRC connected mode until it is notified otherwise from the network. This process happens after the RRC timer and requires two-way communication which is presented in the tests. The device acknowledges the release and goes into RRC idle mode.

2.4.3 RRC idle mode

After ending RRC connected mode the device enters RRC idle mode listening for paging, using approximately 1mA. The default Active Time (T3324) timer is 10 seconds and in this mode the device can receive data from the network with a paging request, meaning that there is data for the device in the network. If there is more data in the network the device will re enter RRC connected mode and try to receive data from the network. If there is no more data for the device it can enter PSM or eDRX. The tests showed the device was only in RRC idle mode for approximately one second before entering eDRX, followed by PSM mode.

2.4.4 Extended Discontinuous Reception (eDRX)

Discontinuous Reception (DRX) is a feature in LTE to keep the device connected, but reducing the power usage by only checking for new data (paging) on time slots. After a sequence of communication the device goes into a DRX state where it periodically checks if the network has new information for the device. If there is information for the device, the device enters RRC connected mode where it can receive and send data.

When designing NB IoT, DRX had to be improved. However, if the device always checks for paging the battery level would decrease at a high rate. eDRX is therefor an outcome of the idea of DRX in the special case of sensor networks. It is a new way to handle communication for sensors

which needs to wake up to certain events or messages from the network. I have included the figure 2.3, illustrating eDRX from an article about eDRX and PSM for LTE-M1. Even though this is for LTE-M1 it applies for NB IoT as well. After a device has communicated, either sending or receiving packets over the network, it is desired that the UE use as little power as possible. By default, the device will enter DRX mode for a network-specified time and if configured go into PSM mode after the given time.

By enabling eDRX, the device will enter eDRX after the RRC idle period. This means that for most of the time the device will sleep, but periodically it will check for new information (paging). The period where the device sleeps is configurable in the network. Since the internal clocks in the devices are unreliable, and the time between these syncs may be long (typically 1-2 hours), the UE and network need to first sync the clock and information between them. This operation is called sync guard and is needed due to communication based on time (time slots/Time division multiple access). After this process the device will operate in normal DRX mode for a certain period (~10 seconds) and go to sleep after that period. This is a very power efficient way to keep the device more synchronized with the network.

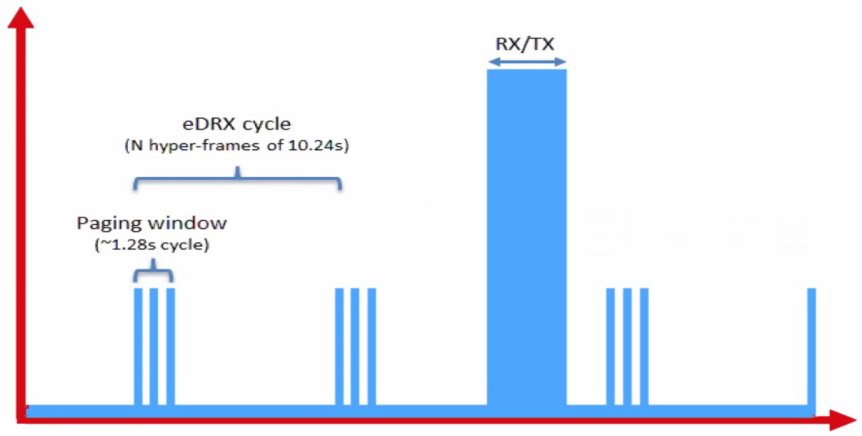


Figure 2.3: eDRX operation [29]

eDRX has a set of configurable parameters which define how the UE behave when entering eDRX mode. In short terms this means specifying eDRX cycle length and paging time window. The cycle length is the length of an eDRX cycle. This means the time from start of a period to the start of the next period. The cycle length can vary from 5 to X seconds. The paging time window can be set to 0 to 20 seconds, where the value 0 means that paging is not in use.

2.4.5 Paging

General paging is a procedure used for communication between UE and eNB. It is the first step for initiating communication and if paging is enabled the UE will listen for data from the network at given time intervals within a RRC or eDRX period. Today paging is allocated on what is called the anchor PRB, hence this resource can be heavily loaded. The paper, "Investigation About the Paging Resource Allocation in NB-IoT"[25], has tried to improve paging for NB IoT and they propose to utilize several resource blocks, and their simulations shows an 30.5% improvement in resource utilization and around 80% reduce in power consumption.

2.4.6 Power Saving Mode (PSM)

PSM is the core of power saving in NB IoT, as well as many other LPWAN. The idea of almost all LPWAN is that the devices send information periodically, while sleeping the rest of the time. After a typical connected period the UE want's to enter sleeping mode where very little power is drawn, only by an internal clock. This mode is called PSM and enables the device to sleep for a network configured time period. In the test case this time (Extended TAU Timer (T3412)) was set to 300 hours, almost two weeks. That means that the device will be removed from all metadata in the MME after 300 hours if there is no communication. The way it works is that the device goes from connected mode, to DRX mode listening for information from the network. After this the device enters PSM mode if activated. When in PSM mode the only thing going on in the device is an internal clock which can start the system at a specific time. This is necessary since the device has to wake up and transmit uplink data at certain user defined intervals.

In PSM the power usage will be as low as 3μA, which is extremely low and the main reason why one can expect these types of sensors to achieve 10 years of battery lifetime.

2.4.7 Coverage Enhancement Level

The UEs will enter different coverage modes based on their signal strength towards the eNB. There are three values of Coverage Enhancement Level (ECL). At which signal power level devices switch over to the different levels is decided by the network, e.g. $-105dBm$ for ECL 1 and $-115dBm$ for ECL 2, signal power better than this resolves to ECL 0.

ECL is not important by itself, but an application fetching the ECL of the device will gain a lot of information about the reception, hence it might reconfigure the setup of the transmit process for example. Based on the signal strength the device will retransmit data to improve Transmit Success Rate (TSR) towards the application server. In ECL 0 the device will

transmit the data once, and in level 1 and 2 the UE will retransmit the data to ensure delivery of the packet. Section 5.1.4 on page 60, gives insight into the transmit process of two transmits, one with ECL 0 and one with ECL 2. The difference is big and will definitely affect the battery lifetime of the device.

2.4.8 The transmit procedure

We have looked at several techniques for NB IoT and the specifications looks very good when the UE has good coverage. In good coverage areas devices will transmit data with a transmit power of $-40dBm$, with an average actual transmit time of ~200ms. Using the calculated values from section 2.4.1 on page 21, this resolves to a power usage of $((244mW/60/60) * 0.2s = 0.0135mWh$. However, in worst case situations, the sensor will boost the signal up to $+23dBm$, and as stated, a sensor with bad reception will try to retransmit data. Depending on the signal power the device will enter different coverage modes, as previously described in section 2.4.7 on the preceding page. Given that the sensor has bad coverage, and is operating in ECL 2 the transmit time might be as long as five seconds for a single data transmit. The actual power usage increases rapidly and the chip would use 726mW for 5 seconds, resulting in a power usage of $((726mWh/60/60) * 5s = 1.01mWh$, which is 74 times increase in power usage compared to optimal conditions. However, as revealed in the tests, $-40dBm$ is normally not possible to achieve.

The following figure 2.4 on the following page, presents an outline of how the device chooses ECL mode dependent on the signal quality. In theory the device should lower its transmit power equal to the signal strength. The graph represents the theory behind the adjustment of the transmit power and you will see how this actually works in section 5 on page 53.

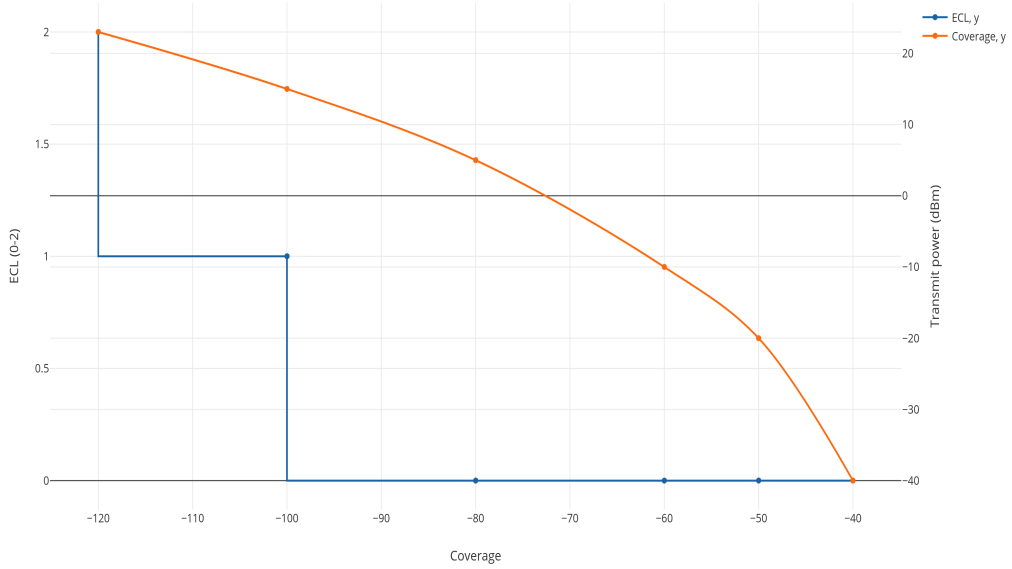


Figure 2.4: Relation between ECL and transmit power

2.5 The current market and deployment strategies

I have presented several applications which suit LPWAN and there is a demand for chips which implement NB IoT or LTE-M1. This section gives a brief introduction to the key firms involved with LPWAN and how the market looks like in early 2018.

2.5.1 Manufacturers and stakeholders

There are many companies which manufacture hardware for the mobile industry. Historically Neul (UK), HiSilicon (China) and Qualcomm (USA) has been the main chipset producers. Neul and HiSilicon are owned by Huawei which makes them one of the biggest producers at the moment. In addition, there are three companies, Ublox, Telit and Quecktel, which have been using "Huawei" chipsets, only adding their custom software. At the moment the agreement towards Huawei has expired and the three software companies will need to source hardware from another manufacturer, which will most likely be Qualcomm.

Currently, there is a problem surfacing in USA which stresses the market. In Europe, GSM/GPRS has been the only alternative, while in the U.S. there has been three network technologies, GSM/GPRS, IS95 and TDMA. Because of the fragmented market, the process of developing

chipsets has been difficult and they have now decided to take down IS95 and TDMA. Most users have gone over to LTE, but there are many companies which use IS95 and TDMA for M2M communication which now has no good alternative. The solution is to move over to another LPWAN, which for most companies will resolve to LTE-M1 because of the opportunities and spread of applications this technology provides. Because of the stress in the market and the increased demand for general LPWAN, all chipset manufacturers began their work on NB IoT and LTE-M1 chipsets. Most of the companies did however reuse their LTE chips to produce LPWAN chipsets. The chip used for my tests is produced by Neul, with software from Ublox, and suffers from the process of reusing hardware originally developed for LTE. Section 3.2.2 on page 32, gives a detailed introduction to how I used the chip, and it is worth noting that the chip was not production ready.

Other companies have seen the development of this market and started to take up the fight against the biggest companies. Nordic Semiconductor is one of these companies, and originally has produced chips for other technologies like Bluetooth. According to a talk they gave, they have used 1.000.000 working hours on chips with LTE-M1 and NB IoT, which is developed from scratch focusing on LPWAN technology. They have seen the stressed market in the U.S. and will begin production of LTE-M1 chips this year, followed by NB IoT in 2019 [38].

Based on the research done in this thesis it will be interesting to see the next generation NB IoT chipsets. These will hopefully be more stable with an even bigger emphasis on power management. The tests performed revealed very promising results given the prerequisites.

2.6 Challenges

With the kind of bandwidth we have today there is less concern with the amount of data transmitted. NB IoT introduces developers to a new kind of applications which should be provident and fully operate for potentially ten years - this introduces difficult programmatic problems. I stumbled upon some of these issues and have included a set of best practice guidelines based on the experience revealed from the tests in section 7.1 on page 89.

The following part of the thesis will cover the project related to the thesis - the testing phase. This part was obviously the most demanding and I have tried to stress the network in ways not intended. Moreover, because of the pre-release of hardware and software[2.5.1] I need to state that the results, especially for the long-term test, should be considered as first results and will change as hardware and software evolves. I will test both Telenor and Telia as they provide NB IoT in Oslo, and it is important to note that I am trying to give an objective statement about NB IoT and

not compare the two network providers.

Chapter 3

Where, why and how?

To get accurate and meaningful results there was a long planning phase. First of all I needed to decide how and what to test, and decided to set up a server which can receive data from a NB IoT UE. In addition, this server will be able to analyze the data and display the results in a meaningful way. I was so lucky to borrow a development kit for NB IoT from Q-Free which enabled me to connect to the network and transmit uplink packets towards the server. I received SIM-cards from Telenor and Telia, which ment that I could do some comparison between the two network providers. With this in mind, the next sections describe and introduce the different parts of the setup and show how they are tied together.

3.1 Testing environment

The project was performed in Oslo and the tests were executed at several locations - UiO, my apartment at Lambertseter (Avstikkeren 7) and at Q-Free's office by Solli Plass. With three locations and two network providers I could collect more information, hence the results got better. Not all base stations in these areas are NB IoT enabled, but you will see that different signal power levels will affect the results. The signal power of the chip is related to the supplied antenna and the direction of this. With the NB IoT development kit there was an antenna and I could position it the direction with best result. See picture 3.1 on the following page, to see the setup at UiO. In section 3.1.1 on the next page, you can see the distances between the different test sites. Unfortunately I was not been able to get the location of Telenor's base station at UiO, but it is likely that it is located at the same place since there are many cells at the location of Telia's cells.

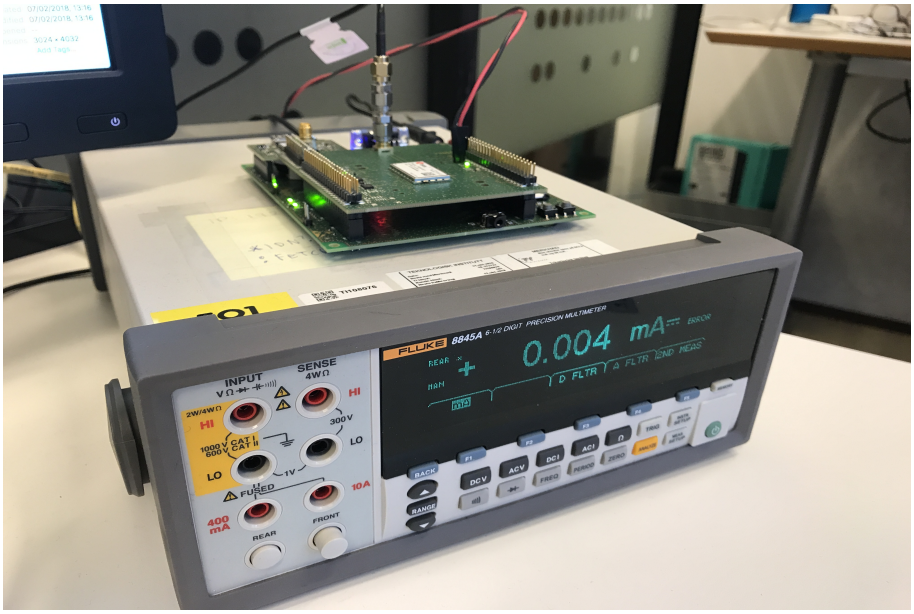


Figure 3.1: NB IoT lab setup

3.1.1 Maps and distances

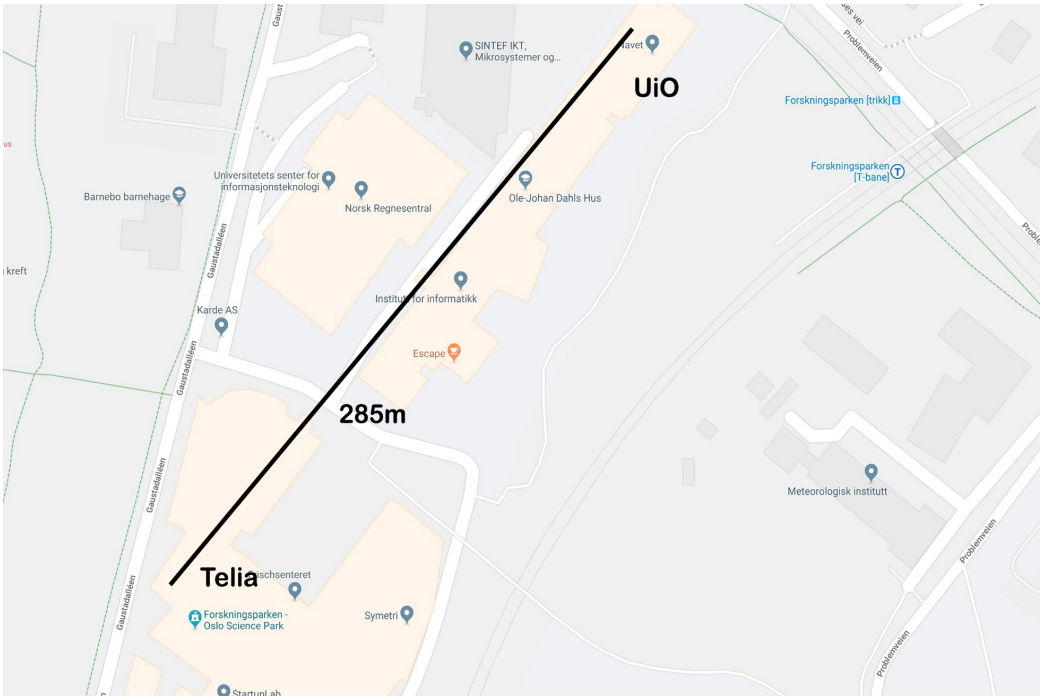


Figure 3.2: Device distance map over IFI, UiO. Referring to www.finnsenderen.no[21], I believe Telenor's cells are located close to Telia's, but have not been able to confirm this by Telenor.

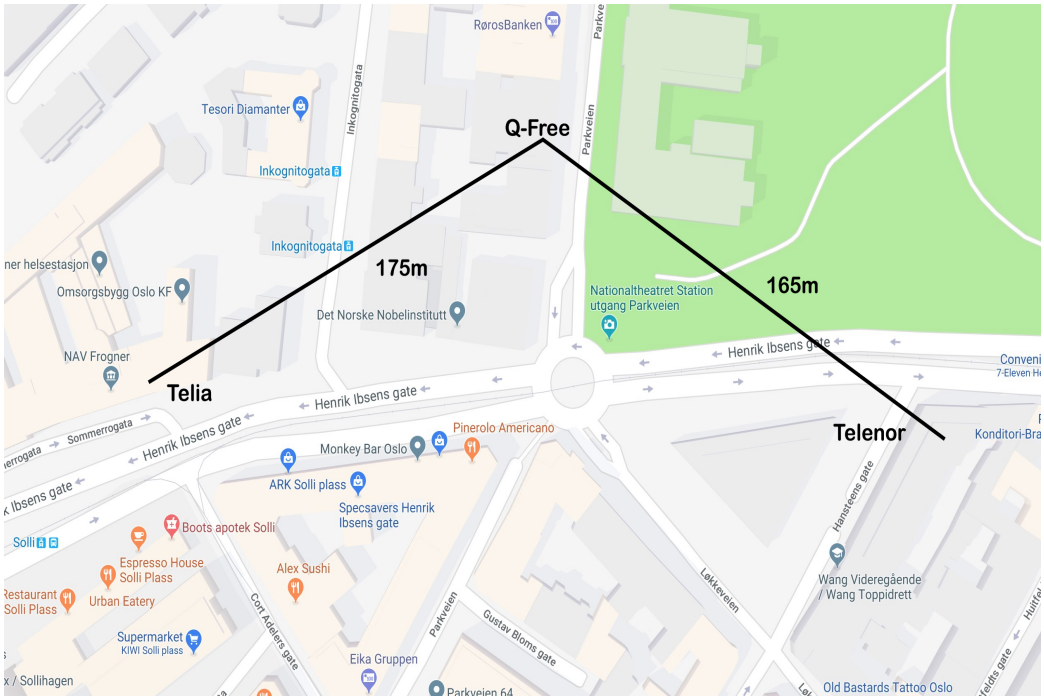


Figure 3.3: Device distance map over Q-Free. The distances are similar, but there are more obstacles between Q-Free and Telia's eNB which affects the reception.

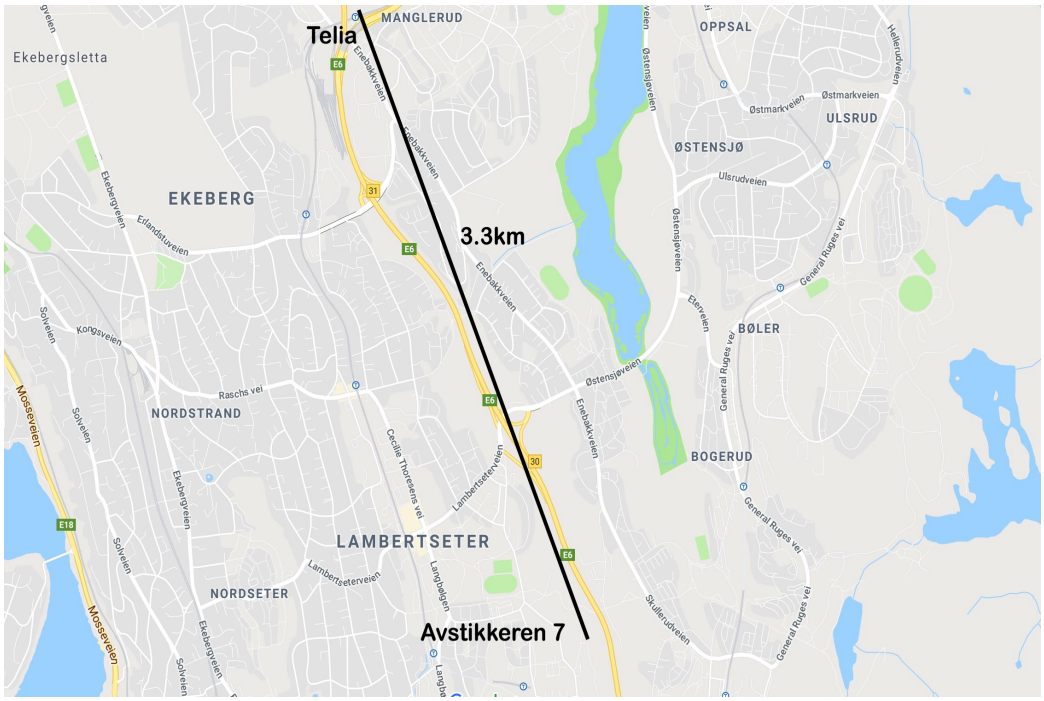


Figure 3.4: Device distance map over Lambertseter, Avstikkeren. The device did not manage to connect to Telenor's network at this location.

3.2 Devices

I have used a set of devices to achieve the results and the following subsections will introduce you to the devices used and why they were necessary for my tests.

3.2.1 Server

To host the web application I applied for a server at UH-IaaS, which hosts servers for university projects. I received a well suited server with the necessary specifications located in Bergen. The server was at times unstable due to power outages, but no results were damaged due to the downtime. Section 4 on page 43, will give a detailed description of the web application.

3.2.2 NB IoT development kit

Early in 2018, Q-Free received a development kit for NB IoT from Ublox. The kit is equipped with a NB IoT chip called SARA N211 from Neul and software from Ublox (see picture 3.5 on the next page). The development kit enabled me to easily develop software for the tests which was crucial. The development kit was connected to a computer over USB and allowed data through the serial port of the chip. I used this setup for all tests, which includes packet size, signal power, latency and general behavior of the network. I also used this setup for longer tests which sends messages continuously with a more practical frequency to the server. The computer will run python programs to test the different features. In many cases the programs will also log test runs locally for detailed graphs of the behavior of the chip. The following subsection gives a detailed introduction to the chip used.

Sara N211 Ublox chip

Section 1.4.1 on page 8, presented Q-Free's parking sensor, which houses an identical NB IoT Ublox chip which is used for my tests. This section will focus on the most essential modem commands for the NB IoT chip. Modem commands are referred to as AT-commands and I will use this notation throughout this paper. As previously mentioned in section 2.4 on page 20, there are configurable timers in the network, as well as flags to keep in mind while developing software for low powered sensors. These parameters can be configured with AT-commands and has proven to give good results when taking power usage into consideration[1]. In addition I

[1] I have used the default timers in this paper

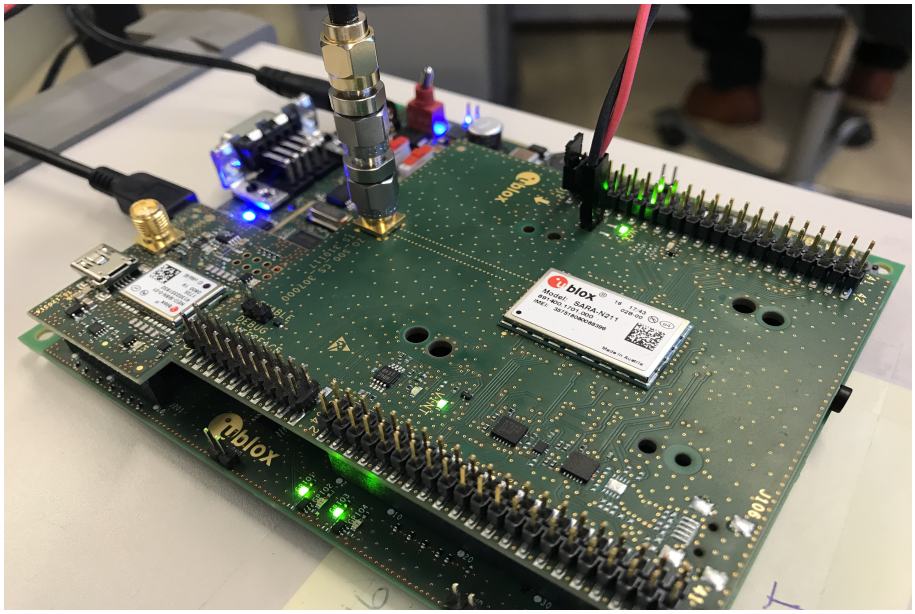


Figure 3.5: Ublox NB IoT development kit. The picture is taken with both attenuators attached, which is a total of 30dBm signal loss.

will show you the AT-command for sending data, and the structure of the data being sent.

Monitoring - NUESTATS

Monitoring is an important part of developing and a way to verify that software is running as optimal as possible. I wanted a way to monitor the performance of the NB IoT chip, hence I choose to take advantage of the precise statistics given from the command **AT+NUESTATS**. According to the specifications of NB IoT, there are a number of interesting thresholds and **AT+NUESTATS** will give me most of this information. In table 3.1 on the following page, you can see the output of the command.

Property	Description
Signal power	NB-IoT signal power expressed in tenth of dBm
Total power	Total power within received bandwidth expressed in tenth of dBm
Tx power	Transmit power expressed in tenth of dBm
Tx time	Elapsed transmit time since last power on event expressed in milliseconds
Rx time	Elapsed receive time since last power on event expressed in milliseconds
Cell id	Physical ID of the eNB providing service to the UE
ECL	Last ECL value
SNR	Last Signal to noise ratio (SNR) value
EARFCN	Last Evolved Absolute Radio Frequency Channel Number (EARFCN) value
PCI	Last Physical Cell ID (PCI) value
RSRQ	Last Reference Signal Received Quality (RSRQ) value

Table 3.1: **NUESTATS** command

I believe that using **AT+NUESTATS** will give me accurate results, but I did stumple upon some problems which will be covered in section, 5.1.3 on page 58 and 6 on page 87.

Time - CCLK

Time is another tool one often use for developing software. One would want to know how long a process endures, and in my case I wanted to monitor the latency of a send operation from the UE to the server. The AT-command **AT+CCLK?** can read the time from the chip and this time is kept in sync with the network and stored in the Mobile Terminal (MT). The read operation returns a string with the time in the following format "yy/MM/dd,hh:mm:ss+TZ", where the characters represent, year, month, day, hours, minutes, seconds and time zone. However, I soon realised that this internal clock was not very good and after the device had been operational for a day the internal clock would be skewed and the latency results were not accurate. Section 6 on page 87, will include a summary of this problem, but my suspicion is that the clock is not updated very often and with an imprecise internal clock the time will be wrong. As I were perfoming all tests with a computer connected to the development kit I started using the clock of the machine connected to the kit and send this timestamp to the server. This did not only provide a more updated clock, but the timestamp also included milliseconds which gave the result higher precision.

eDRX settings - CEDRXS

Section 2.4.4 on page 22, explained how eDRX works and how one can configure the UE. The AT-command **AT+CEDRXS** can be used to define how eDRX is performed on the UE. One can for example issue this AT-command, **AT+CEDRXS: 1, 5, "0101"**, to put the UE into eDRX mode with cycle length of 163.84 seconds[2].

PSM settings - CPSMS

One can also define how PSM is used by the UE. One can enable PSM, and set the timer Extended TAU Timer (T3412)[3] with this command. If PSM is enabled, the device will enter deep sleep after expiry of the timer Active Time (T3324)[4].

Create socket - NSOCR

The command opens a port on the UE, which enables data transfer on the given port. My application opens an UDP socket on a port, and the port used can be a number between 0-65535, except for 5683 since this is used by the chip to send CoAP messages. The command returns a socket number which will be used for subsequent send (NSOSTF) commands.

Send command with flags - NSOSTF

This AT-command is used to send UDP packets on a given port with a specified flag. The command takes a set of parameters and the data in hexadecimal format. The parameters are, socket number (retrieved from the AT-command NSOCR), remote IP address, remote port, flag, data length and actual data. The flag can be used to specify the behavior of the UE after the data has been transmitted, see table 3.2 for details.

Mode	Description
0	No flags are set
1	Send message with high priority
2	Indicates RAI. This mode will put the UE into PSM mode directly after transmitting data. Meaning less time in power heavy modes.
3	Indicate release after next message has been replied to.

Table 3.2: Transmit flag options

[2]This is the NB IoT default value for eDRX.
[3]The NB IoT default value of T3412 is 54 m.[53]
[4]The NB IoT default value of T3324 is 60 s.

The data transmitted towards the server adds an additional CoAP header followed by the actual data. I decided to do this to reduce logic on the server and create the environment similar to a real world application.

Signaling connection status - CSCON

One can verify the connection status of the UE with the command **AT+CSCON?**. If the reply is 1 this means that the UE is in RRC connected mode and is able to send and receive packets. If the reply is 0 this means that the UE is in idle mode, hence the device is inactive. Note that this does not mean that the device is in PSM mode. This only occours after the configured T3324 timer expires.

Network registration status - CEREG

The command AT+CEREG can verify the network registration status of the UE. This command is a very useful for any application which wants to check if the UE is still connected to the network.

PSM status - NPSMR

Gives the status of PSM on the device. It will return **1,1** if the device is in PSM mode which means that the power usage is very low.

3.2.3 Fluke precision multimeter

Q-Free owns a precision multimeter which I used to measure the current through the NB IoT chip. With the multimeter I managed to pinpoint different stages of the chip and by adding corresponding statistics from **AT+NUESTATS** the performance of the results increased. As sketched in figure 3.6 on the facing page, the multimeter was connected to the computer using an ethernet cable, enabling readings from the device. Depending on the selected precision of the measurements the device collected between 30-300 samples per second. The most detailed tests required high sample rate, while the longer tests used lower precision measurements. The device responded on commands much like a Read-Eval-Print-Loop (REPL), where you issue a request with a command and the device will respond with an answer. In table 3.3 on the next page, you can see the commands I used and a short description related to them.

Command	Description
:INIT	Clears old readings
:FETCH?	Fetches all readings from the device. The response is a string with numbers divided with ","

Table 3.3: Fluke commands

3.3 Software

I developed many test programs with a specific setup (see figure 3.6, for an overview) as well as a web application to give me the knowledge and experience to give a precise report about the status of NB IoT the spring of 2018. A thing I learned during the testing phase is that the output format often changed. Since the programs produces results based on data which is already processed it meant that some tests had to be reperformed. In hindsight it would be better to capture the raw data and develop a program which generated graphs based on the data. However, I am satisfied with the end result and you will see many of the generated graphs in section 5 on page 53.

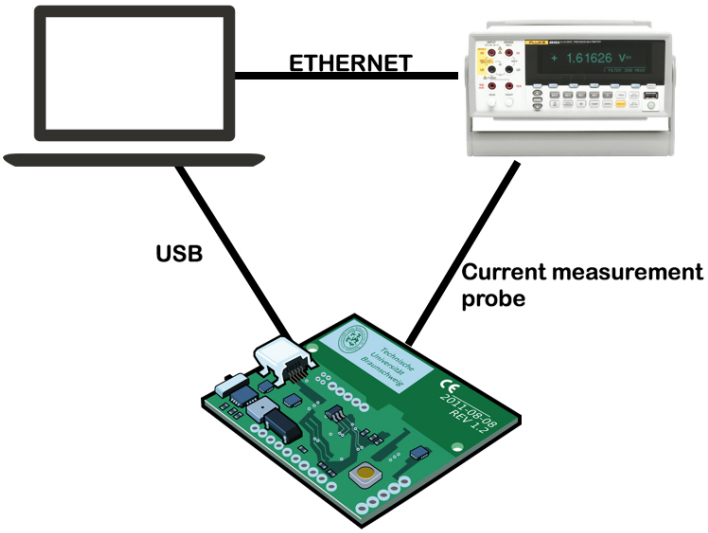


Figure 3.6: Lab setup [40] [22] [26]

I started experimenting with different AT commands and created small programs to test the functionality of the chip and the connection towards

the server. I created a simple program which sent **AT+NUESTATS** to the application server and went on transforming this program to helper methods which would be included in later test programs. A sample packet with a CoAP header would look something like the example 3.1, where the payload is converted to hexadecimal. I used some time creating the correct CoAP packet for the server to accept it as a CoAP packet. If you don't want to use CoAP you can simply fill the payload with data without the CoAP header, but this means that you are missing the features of CoAP which could be beneficial for your application. Dropping the CoAP header means that you are sending a pure UDP packet through the network and it is harder to verify the packet when it is received on an application server.

Listing 3.1 NB IoT sample transmit

```
1 AT+NSOSTF=0,"158.39.77.97",5683,0x0,88,"500230
2 30B131112AFF2D313039365F2D313034325F3233305F39
3 3732315F37363838305F333434333934353325F315F3132
4 335F363235325F39385F2D3130385F31382F30322F3139
5 2C31363A31363A35362B30305F31325F"
```

I proceeded working on communication towards the multimeter, which was quite easy. The multimeter was given an IP address which was related to the IP address of the connected machine. Creating a socket to this IP address in python and connecting to it was simple. The test programs starts off by clearing all buffers with **:INIT** and then fetch data on a regular interval with **:FETCH?; :INIT**. Each reading was converted into a list of numbers which was appended to a global list of readings. Fetching for new data every 5th second was a good tradeoff between fetching all the time and very rare. These functions were added to the helper file **nbiot_labtest_helpers.py**, and forward I went on writing a couple of programs to monitor the behavior of the chip. The following subsection includes documentation on the most essential test programs.

3.3.1 Test programs

at_command_test.py is a program well suited for testing different AT commands which is necessary to explore and research how the device handles different features. The program is like a REPL and requests an optional message and a command. If a message is supplied the program assumes that the request is a send command. If not, the command is handled and the respons is printed in the terminal. Note that this program is written in python V2 while the other programs expect python V3 to execute. See github[47] for complete code.

nbiot_labtest.py continuously transmits packets over NB IoT - either status packets towards the server or random data of different sizes. It enabled me to test different scenarios and log the results in a representable way with graphs. The program takes in a list of parameters described in table 3.4.

Parameter	Description
-id	ID which you want to POST towards the server. Default: 0
-gn	Output graph name. The supplied name will lead the heading, followed by a list of the rest of the parameters.
-d	Set the desired delay between each iteration. Default: 5
-i	Set the desired iterations. Default: -1 which means that the program will loop until the user presses ctrl+c
-b	Set the desired length of the payload in bytes. If 0 is requested, statistics from **NUESTATS** will be sent to server. Default: 100
-r	Set RAI. Default: false
-l	Set device logging. Default: false
-rb	Set test to reboot NB IoT chip directly after execution. Default: false

Table 3.4: **nbiot_labtest.py** parameters. See NB IoT labtest[49] for complete code

To get more stable tests and to automate the process I added logic to perform a set of tests when starting the program. If byte size is not given to the program the execution will perform 4 tests on 50, 100, 200 and 512 bytes. Those 4 tests are a combination of logging and RAI.

For each iteration the program sends a message towards the server, followed by a period of logging. If logging is enabled the program fetches information from the chip with **NUESTATS** and three other commands - **AT+NPSMR?**, **AT+CSCON?** and **AT+CEREG?**. From **NUESTATS** the program retrieves information about transmit and receive time, transmit power, ECL and signal power. I choose these parameters because they give good information by themselves, as well as they combined gives a good indication of what the chip is doing at any time. These results were logged approximately four times per second - limited to the speed of the serial bus between the computer and the development kit. Every fifth second it also fetched stored information from the precision multimeter. Depending on the configured precision of the multimeter, 50-400 samples were logged per second which was correlated to the logged information from the chip.

With all results the program produces a set of graphs which can be used for analysis of the network and the chip.

Because there were no way to retrieve the timestamp of each measurement of the multimeter I had to flatten the result and apply a timestamp to each measurement. This means that I assume that the interval is equal. When looking at the figures in section 5.1 on page 53, the power graphs are mostly alligned with the results created from **NUESTATS**, but often one or two seconds in front of the other graphs.

Graphs generated:

- Receive: Shows the percentage of how much time the device were receiving compared to the actual interval.
- Transmit: Shows the percentage of how much time the device were transmitting compared to the actual interval. Both receive and transmit scales use 1 as 100%.
- Transmit power: Shows the transmit power of the device.
- Signal power: Shows the signal power of the device.
- ECL: Shows which signal power level the device is currently in.
- PSM: Shows the status of PSM.
- RRC state: Shows the RRC state.
- Registration status: Shows the registration status of the chip. 1 is connected, 4 is unknown and 5 is searching for network.
- Power usage: Shows the current consumption of the chip in mA.

Based on the average current of one test the power usage can be calculated. The average current can be converted to mWh and convert this to mWs and multiply the result with the duration of the test. This is an example from webapp [42]: $((9.2089mA * 3.3V)/60/60) * 60 = 0.5064mWh$. With this information it is possible to show the difference in power usage with different scenarios and use the results to outline the lifetime of a device under different kinds of environments. The constant UE logging with **NUESTATS** consumes a lot of power, hence the program can be started without **NUESTATS** (do not use '-l') logging, which gives a better overview of the actual power usage. Most of the tests are performed both with and without the '-l' parameter.

power_calculator.py calculates the power consumption over a given period from one of the test results from the short-term tests. This tool was important in the process of figuring out how much power specific parts of the transmit uses. It extracts all power measurements in mA, between two timestamps and calculates the average current, multiplying this with the time delta between the two timestamps supplied. The program takes in a list of parameters described in table 3.5.

Parameter	Description
-f	File name. Must be from short-term test and have .html format
-s	Start time
-e	End time

Table 3.5: **power_calculator.py** parameters. See power calculator[50] for complete code

nbiot_labtest_details.py tests different packet sizes. The program transmits 6 packets with different packet sizes[5] a given number of times. After the given iterations are processed the program produces two graphs, one with all recordings and one with normalized data[6]. The program takes in a list of parameters described in table 3.6.

Parameter	Description
-gn	Output graph name. The supplied name will lead the heading, followed by a list of the rest of the parameters.
-d	Set the desired delay between each iteration. Default: 5
-i	Set the desired iterations.
-r	Set RAI. Default: false

Table 3.6: **nbiot_labtest_details.py** parameters. See NB IoT details[48] for complete code

[5] 25, 50, 100, 200, 400 and 512 bytes

[6] I have choosen to include average, sum, min and max values for each packet size

Chapter 4

The web application

The long-term tests generated a lot of data, and the simple solution would be to collect the data and import it into Excel for analysis. However, there are approximately eighteen thousand elements in the database, thus maintaining and keeping track of this data in Excel would not suffice for an effective analyzation of the data after the test. I decided to implement a web app to display information about the sensors. This web app can display latency, signal power and other power related data about each sensor. Most of the practical knowledge was gained from results coming into the web application and I would not perform the kind of short-term tests which gave the most interesting results without the web application. I have included some results from the long-term tests in the paper, but if you would like to explore more results I recommend looking at Q-Free, Telenor 09-10 March 2018 and Avstikkeren, Telia 13-15 March 2018[1].

The following sections discusses the implementation of the server and the web app.

4.1 Backend: Node.js

I decided to use node.js for what would become the backend of the server. Node.js is a new and modern way to create simple REST APIs. A REST API follows a set of rules to make it robust and stateless. It also should include all CRUD operations (create, read, update and delete[2]) and should use the appropriate request method, such as POST, GET, PUT and DELETE. A big difference from standard backend programming is that Node.js uses javascript as programming language. Even though javascript is not the cleanest programming language, it is efficient and the codebase is kept low. In addition node.js takes advantage of using Node Package

[1]Here you can see that the latency is affected by the clock bug we discussed in section 3.2.2 on page 34

[2]The server only implements create and read

Manager (NPM), which offers packages for many kinds of applications, both frontend and backend.

The server hosts multiple API endpoints and the most important are collecting data and retrieving data. The database contains a lot of data collected from different tests and it would require a lot of rendering to do analysis on the client side. I decided to analyze the data at specific intervals so that the client would simply display the data. The analysis and data is described in section 4.3.2 on page 51.

4.1.1 The database

There are multiple ways to store data and there is always a trade off when choosing the platform. PostgreSQL is a great example of a database which is fast and reliable, but it can be complicated to set up and it is SQL driven which means all data has to fit into a pre-defined configuration. Since this project is small and the data could potentially change during the test phase, I chose to use a NoSQL database. There are many good tools for storing data in a NoSQL database. MongoDB is the most commonly used and works well with Node.js, hence I decided to use it for data storage. Subsection 4.2.4 on page 47, explains how mongoDB was used in my implementation.

4.2 JavaScript packages for the server

The web app uses different packages from NPM to include certain features. By including quality packages the server logic and size was kept low and I could focus on the data. The following subsections gives a short introduction to the most important packages used.

4.2.1 Express

Express is a fast and robust package for handling requests and offers nice APIs for listening on specific ports and so forth[3]. The code example 4.1, shows the base setup for making a server handle requests. The port is set with a combination of the processes port and user specific port selection. The reason why this might be useful is if your server is hosted on a payed server which might be deployed on different IP and port for each deploy. With this implementation you won't need to consider this. In the example you can see a GET for a specific path, however you can also redirect all routes to a separate file for cleaner code. The last thing one need to do is to listen for incoming requests on the specific port. Now express will handle all request with a single threaded event loop.

[3]See [19] for documentation

Listing 4.1 Base express setup

```
// express setup
const server = express();
const port = process.env.PORT || 8020;

server.get('/', function (req, res) {
  res.send('Hello World')
})

server.listen(port, () => {
  console.log(`app started on`, port);
});
```

4.2.2 CoAP

Constrained Application Protocol (CoAP) is a simple network protocol and offers requests like HTTP, but without all the overhead. CoAP is designed for low powered devices and will be used for testing. I have used node-coap package for including CoAP support on the server[4]. Since the protocol is simple, the requests needs to be handled a bit different. Code example 4.2, shows a simple node-coap setup.

Listing 4.2 Base CoAP setup

```
var coap = require('coap')
var coap_server = coap.createServer()

coap_server.listen(port, () => {
    logger.log("info", `Worker ${process.pid}
       started coap server on ` + port);
})

// All request is handled by this function. It is
   not possible to request a specific url path
coap_server.on('request', function(req, res) {
    // Payload of the request is a byte stream.
       Parse the stream and handle the data
    var data = JSON.parse(req.payload.toString());
})
```

[4]See [31] for documentation

4.2.3 Cluster

With express all request will be handled by one thread as Node.js is single threaded. Since the server basically puts everything on hold while analyzing the data, there was a need for a way to handle request at the same time. The package cluster can handle several requests asynchronous and is based on web workers which are an abstraction for processes in Node.js[28].

The code example 4.3, shows how to enable multiple processes to handle requests to improve efficiency. The cluster package offers a set of functions, so the first thing to do is to verify which worker is master. The master then forks a number of workers which will be divided into processing one specific task, either analysis or requesting HTTP/CoAP requests. Since the server will run over a long time it is important to handle workers which die. The master process will pick up dead workers and respawn the worker so that the server can continue processing requests. It is very important that the worker is forked with the same id since the worker depends on the id to select which task to perform.

Listing 4.3 Express setup with cluster

```
1 const server = express();
2 const coap_server = coap.createServer()

4 const cluster = require('cluster');
5 const numCPUs = require('os').cpus().length;

7 if (cluster.isMaster) {
8     // Fork workers.
9     for (let i = 0; i < numCPUs; i++) {
10      const worker = cluster.fork();
11      notify worker with id
12    }

14    // Respawn workers on exit
15    cluster.on('exit', (worker, code, signal) => {
16      const new_worker = cluster.fork();
17      notify worker with id

19      // Remove old entry from map
20      delete worker_map[worker.id]
21    });
22  } else {
23    // Send message to master process to get
        appropriate id
24    process.send({ msgFromWorker: '' })
```

```
26      // Receive message from the master process.
27      process.on('message', function (msg) {
28        let id = msg.id;

30        if (id == 1 and numCPUs > 1) {
31          // Start analysis task
32        } else if (id == 2) {
33          server.listen(port, () => {
34          });
35        } else {
36          coap_server.listen(() => {
37            // Start coap server
38          })
39        }
40      });
41    }
```

4.2.4 MongoDB

Storing data is key in any server and the server use the MongoDB package. The package includes a nice API for MongoDB storage and is simple, and highly effective[33]. MongoDB can consists of several databases, and each database can contain several collections. An entry within a collection is called a document, and the document can contain data with any size and structure.

One has to install MongoDB on the particular instance and on linux it is as simple as "sudo apt get mongodb". The install process creates a service, mongodb.service, and this service starts at boot time and is ready for incoming connections.

The code example 4.4, shows how to connect to the db and insert a simple document in a collection with an API call.

Listing 4.4 MongoDB setup and insertion

```
1 const MongoClient = require('mongodb').MongoClient
    ;

3 MongoClient.connect('mongodb://localhost:27017/db'
    , (err, db) => {
4     if (err) return err

6     server.post(api + '/nodes', (req, res) => {
7         const data = req.body;
```

```
8           db.collection('nodes').insert(data, (err,
               result) => {
9             if (err) {
10              logger.log("error", "insert failed: "
                   + err);
11              res.send({ 'error': 'An error has
                   occurred' });
12            } else {
13              res.send(result.ops[0]);
14            }
15          });
16      });
17  }
```

4.2.5 Moment

Storing data with MongoDB is easy, but deciding the format of the contents can be difficult. A normal problem on server applications is server time, versus client/sensor time. A common approach is to use UTC time everywhere and the package moment is a great tool to keep track of time[5]. The code 4.5, takes in a timestamp and creates a moment object in UTC time. This original timestamp is not converted, so the sensor also needs to send the timestamp as UTC time. The conversion of timestamps happens in the web application.

Listing 4.5 Simple moment example

```
1 server.post(api + '/nodes/:id', (req, res) => {
2     let data = req.body;
3     let timestamp = moment.utc(data.timestamp);
4 }
```

[5]See [32] for documentation

4.3 Frontend

The web app is used for hosting data related to long-term tests and how the device performs on a more abstract level than with the detailed tests in section 5.1 on page 53. The home page contains a short paragraph about the thesis and links to the latests version of the thesis, source code related to the project and results from the detailed tests, see screenshot 4.1. Further more you can select nodes on the left which includes data from different sites and network providers, see screenshot 4.2 on the following page and 4.3 on the next page.

With a solid backend in Node.js I choose React for the web app. I will not go in detail on the different packages used for the web app, since this is not relevant to the thesis. However, the following section gives a brief introduction on how to use the app and what kind of analysis you can view with it.

4.3.1 Layout

By default the graphs will display data one day backwards from the latest data entry of the selected node. This means that if the day you visit the page is 20.07.18, and the latest data entry is 02.03.18, you will see data from 1-2 March of 2018. You can also inspect certain areas by selecting an area of one of the graphs and the other graphs will adjust.

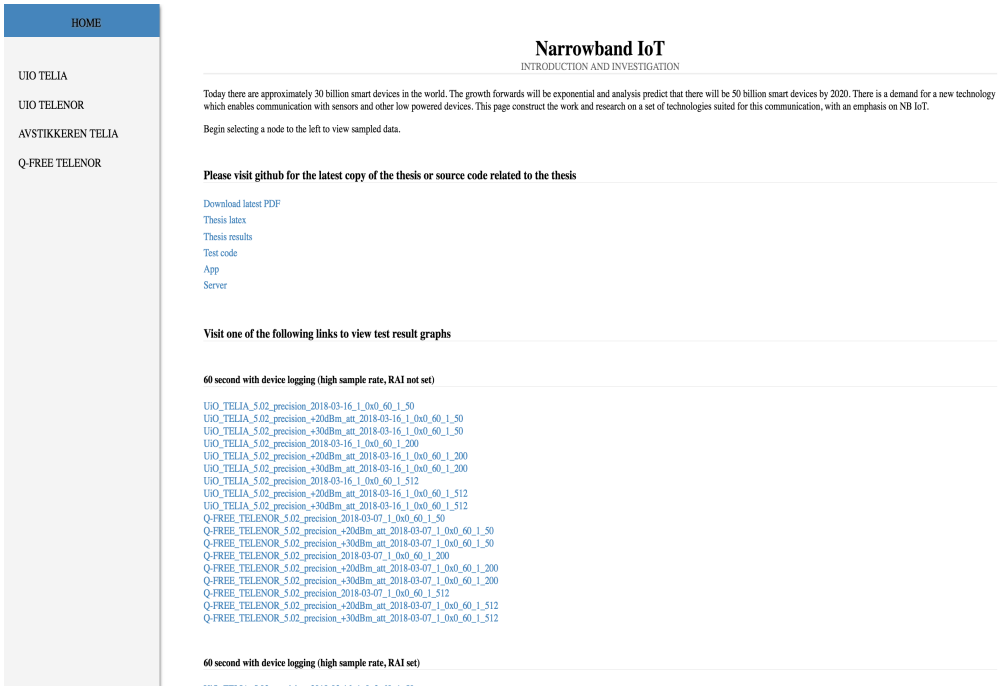


Figure 4.1: SensorApp homepage. Additional material are linked at this page, and you can select a node on the left to explore the long-term results.

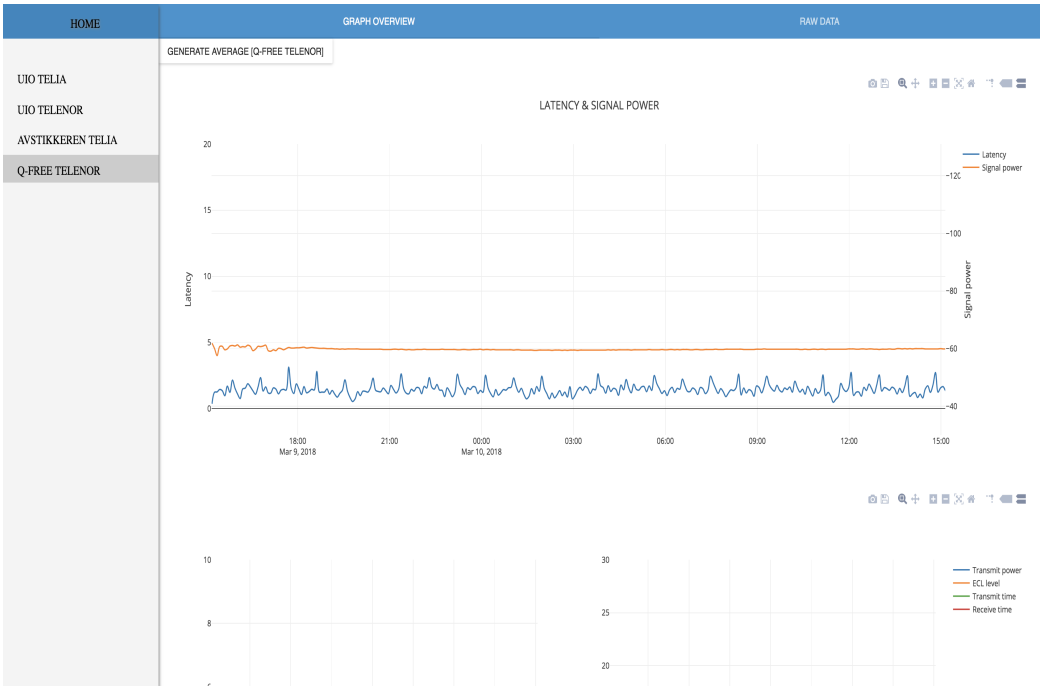


Figure 4.2: SensorApp nodepage part 1. The figure displays the signal power and latency graph from Q-Free, Telenor 09-10 March, with very good results.

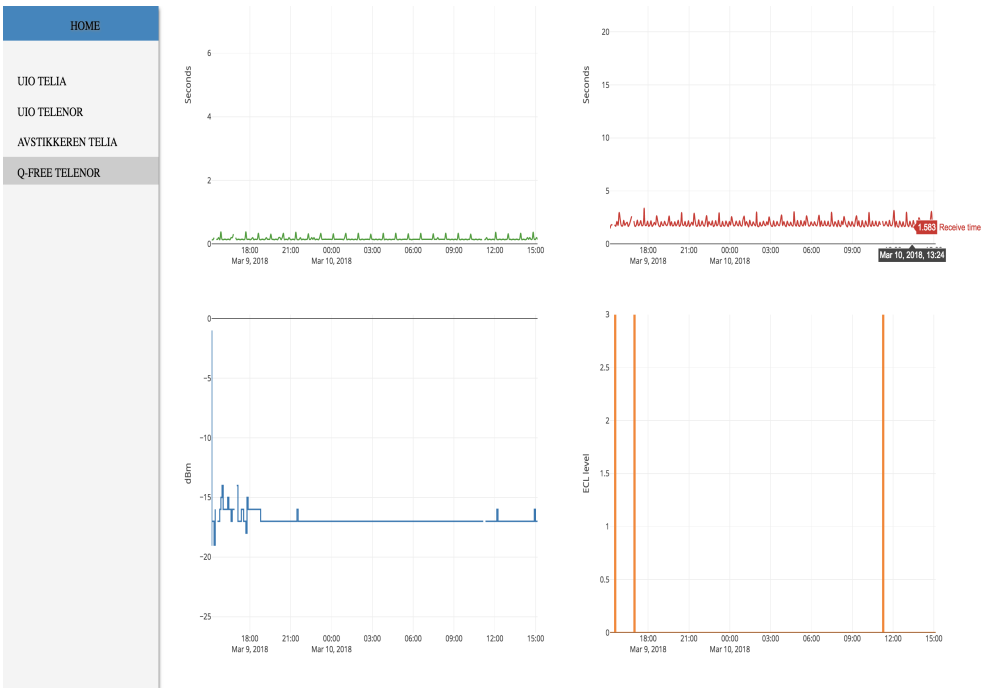


Figure 4.3: SensorApp nodepage part 2. The figure displays the general statistics from Q-Free, Telenor 09-10 March, with very good results.

4.3.2 Analysis

When having selected a node you can explore the different graphs, which is a good way to display statistical data. The graphs data represents the output of the analysis on the server and the following sections describe what the graphs present and why I choose the different aspects of the behavior of the network. Remember that the data is mostly produced by **AT+NUESTATS**, so read section 6 on page 87, for any deviations.

Signal power and Latency

One of the main features of NB IoT is coverage, hence displaying signal power statistics over time was important. One interesting point of signal power is that it is closely related to the power usage of the device. The latency of NB IoT is not a key feature, but the desired latency is under ten seconds according to the specifications[52] and by displaying the latency in the same graph as signal power it is clear how the different levels inflict latency in the network. The output of signal power and latency is directly from the output of **AT+NUESTATS**. To give a clearer understanding of signal power I have classified the values into categories, see table 4.1.

Signal power (dBm)	**Category**
-40 to -60	Excellent signal
-60 to -80	Good signal
-80 to -100	Quite good signal
-100 to -110	Bad signal
-110 to -130	Very bad signal

Table 4.1: Signal power categories

Recieve and transmit time

The output from **AT+NUESTATS** gives a receive and transmit counter in milliseconds. The value between each entry was used to calculate actual receive and transmit time. The analysis produces a value which represents how much time the device was in receive or transmit mode in one interval. This is also closely related to the signal power of the device and also the ECL value which indicates the number of retransmits per packet sent. The device will mostly use power in these periods, so if the device is active for longer periods in either receive or transmit mode the expected lifetime is degraded. Most of the time the active percentage will be low due to usage of Release Assistance Indicator (RAI) and PSM mode, however if the transmit frequency is low the device will more frequently be in some kind of active state, hence the transmit and receive time will increase.

ECL

The signal power level is also closely related to the power usage of the device and an important factor to monitor. You can clearly see that the power usage increases when the device enters ECL 1 and 2 in the testing section 5 on the facing page. The output of ECL is taken directly from the output of **AT+NUESTATS**.

Transmit power

The amount of power used for transmit is decided by software and is key to power usage. The output retrieved from **AT+NUESTATS** show the latest transmit power level of the device. Sometimes the device will send a negative-acknowledgement (NACK), which is always sent with $+23dBm$ signal strength and is what **AT+NUESTATS** sometimes picks up as the latest transmit power level[34]. In the long-term tests with graphs from the web app the logging of transmit power might not be correct related to what transmit power level the device actually used for transmitting the packet. However, in the section on detailed tests 5.1 on the next page, I will log the status of the device up to ten times per second which will give a better understanding of what is happening with the transmit power level. In addition the actual power usage will be monitored with a multimeter, giving more accurate test results.

Chapter 5

Testing

I have tried to include as many tests as possible and this chapter discusses and analyses the results. The test phase began early January and was completed late March. The reason for the late test phase was due to a number of things. First of all, the hardware and software related to NB IoT were not available on a stable platform until late 2017, and Telenor and Telia had limited NB IoT enabled base stations. Even at the beginning of 2018, the devices used were not production ready, but at that stage the error rate was very low. Because of the early adaption I encountered some interesting results, giving an indication of the state of the technology. Given the hypothesis I needed to test the chip and the networks with good equipment and at several locations. Chapter 3 on page 29, gave you an overview of the testing premises and the devices used.

5.1 Short-term tests

I will present how the device handles normal usage and edge cases where non-default behavior is activated. I define normal operation when ECL is 0 and signal power is better than -100dBm. When the UE has bad reception it may retransmit packets and operation like this over longer periods is not sustainable. With normal operation it is also expected that the UE transmit packets with RAI set and PSM activated.

Short-term tests is defined as a logging sequence over a short period with frequent data points. The duration of the tests are under two minutes and the sample rate is high, over 300 samples per second for some tests. These tests will show you how the device performs on a detailed level and will give a good understanding of the transmit process. The reason for including these tests are related to the actual power usage over a short period. By defining power usage statistics for different kinds of transmits it is possible to normalize the data and give an estimation of expected lifetime based on the tests. The detailed tests also will show if the network providers follow the specifications. It is very interesting to follow PSM,

connection status, signal power and power usage over a short period and see how the device handles different scenarios.

The results are sectioned into categories and I will refer to a subset of the results in each section with a reference to a more detailed version in each caption. In this way you will be presented with one problem at a time which will give you more in-depth knowledge.

5.1.1 General

Most of the short-term tests were performed with an emphasis on power usage, rather than coverage and latency. It is however clear that high power usage is a side effect of bad coverage. All tests show that better reception gives lower latency and power usage. In one of the test cases with Telia at Q-Free I saw that the device was struggling even with good reception. Figure 5.1 on the next page, shows that the device is not entering PSM mode even though the expired 120 seconds has elapsed. The device does not leave RRC connected mode, hence the power consumption stays at $6mA$. The signal power of the device at this location was good and the SNR values were much like Telenor's at the same location. This test did not use RAI, so the expected behavior is transmit at the beginning of the program, following a period of 20 seconds in RRC connected mode. After this the device should enter eDRX mode, only receiving data on a set interval, normally 2-25 seconds, with a number of receive transactions within each eDRX period. After this period the device should enter PSM mode if it is enabled by software. However, in this test it looks like the device is performing DRX and polling for data from the network very frequent. Looking at the network connection graph you see that the device does not leave this state until it suddenly indicates loss of connection and rapidly reestablishing connection and transmits the last packet. Directly before the last transmit there is a short period of frequent receive transactions. This might be related to the loss of connection, presuming that the ECL value is delayed until connection is restored. An alternative explanation is that due to the bad configuration the time of the paging window negotiation is longer than expected, resulting in a period of constantly trying to receive data from the network. Continue looking at the time of the last transmit, something interesting is happening with the transmit power. After renegotiating the connection towards the network the transmit power is adjusted from around $+6dBm$ to $+23dBm$ even though the signal power is kept at the same level. At this location most of the tests showed some kind of flaw in the network, either related to packets being dropped, long transmit times or trouble following the normal transmit procedure. It would be interesting to investigate what happened of the network side of this setup. The network becomes a black box and when the network performs out of the ordinary it is hard to

debug.

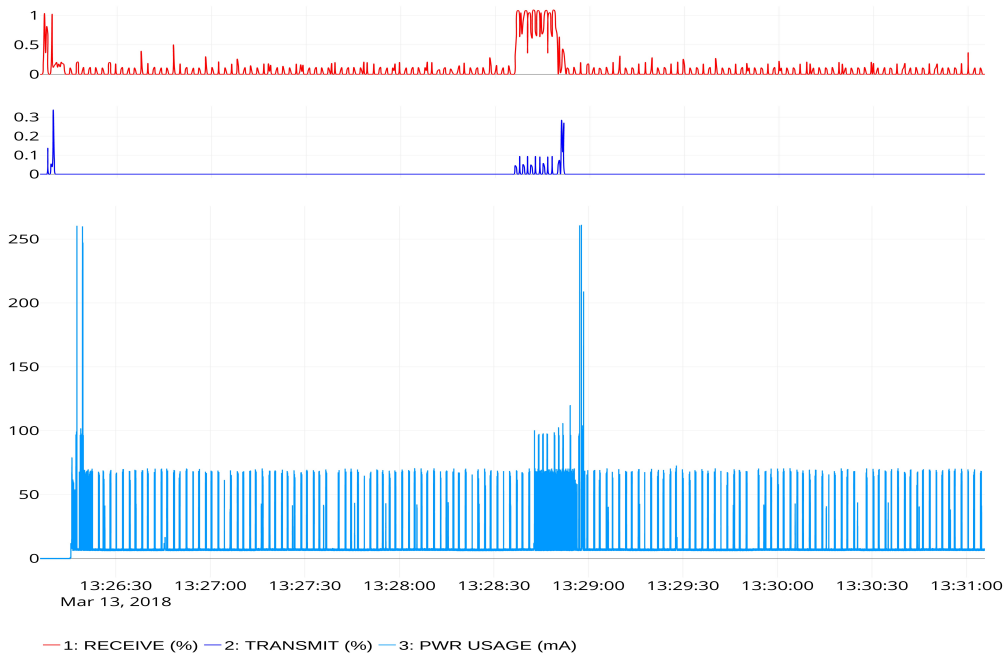


Figure 5.1: The figure displays one test from Telia which behaved out of the normal. See figure 5.15 on page 71, or visit webapp [7], for more details.

With this in mind I will present a proper transmit procedure from Telia and Telenor. Looking at figure 5.2 on the following page and 5.3 on page 57, the process is totally different and is looking more like a graph representation of the specifications. Looking first at the graph from UiO with Telia there is two transmit periods of around 20 seconds, following a period of eDRX and then PSM. RAI was not used in this test either, hence the RRC period of 20 seconds. Notice that the device is not pulling for data all the time within the RRC period. It usually receives a lot of data when initiating the uplink transmit, but after the transmit it only pulls for data around 20% of the time, hence reducing the power usage. When asking Telia about this they state that this is most likely related to DRX process which kicks in because there is not any data to receive[17]. However, I did not manage to figure out why the device performed this way when connected to Telia and not Telenor. It is also worth noting that when the device is not in RRC connected mode the statistics from **NUESTATS** become stale. This is an indication of deep sleep mode, and in eDRX the device should sleep between each eDRX period. After two eDRX periods the device enters PSM with permission from the network.

Moving focus over to Telenor there are some distinct differences from Telia's transmit process. The first thing to notice is that Telenor is pulling

for data close to 100% of the time spent in RRC connected mode. This results in an increase of 5 in terms of pure power usage compared to Telia's solution. The current at this stage is comparable at around $60 - 70mA$, a little higher than Ublox specification on page 16[52]. If you look at the graphs from the website you are able to zoom in on these periods, revealing that Telia's RRC period is mostly using $6 - 7mA$, only flickering up to $60 - 70mA$ when actually receiving. Telenor's RRC period however is mostly at $50 - 60mA$ and it is likely that there are different configurations in the networks resulting in actual receive time. A part from this, the graphs have the same characteristics. One thing worth noticing is the adjusted transmit power because of the excellent reception at Q-Free. The device has $-55dBm$ signal power resulting in $-24dBm$ transmit power, almost minimum. This is clearly reflected in the power usage graph where Telenor only uses at most $70mA$ when transmitting, while Telia at $+9dBm$ transmit power is reaching $110mA$ at transmit time. Notice that this is close to what Ublox state in their specifications. It is however lower than calculated and this will be discussed in section 6 on page 87, about deviations.

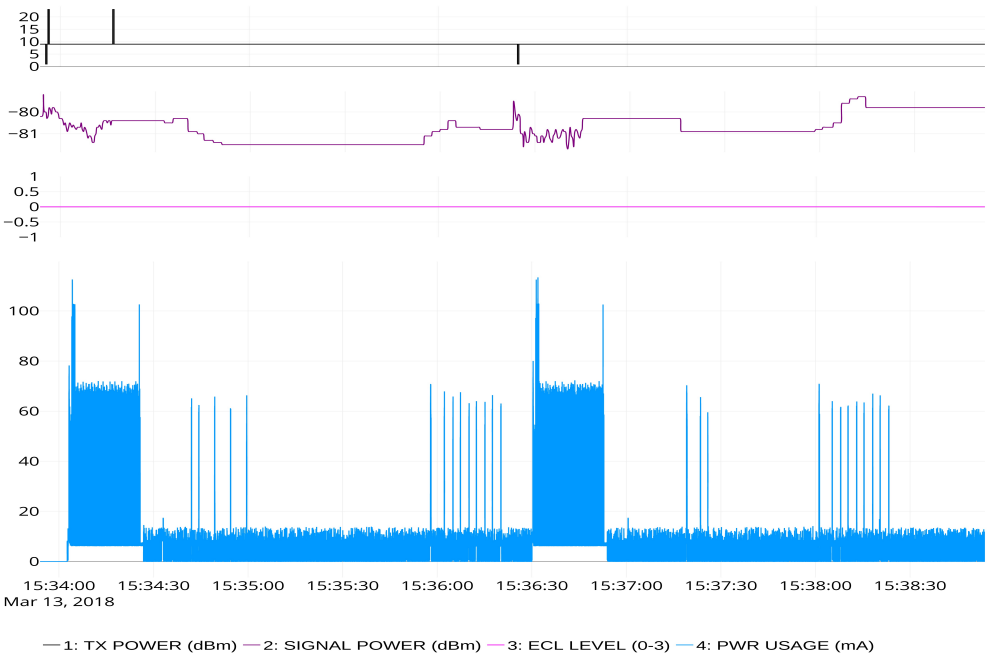


Figure 5.2: The figure displays normal behavior with two transmits over 5 minutes at UiO with Telia. See figure 5.16 on page 72, or visit webapp [8], for more details.

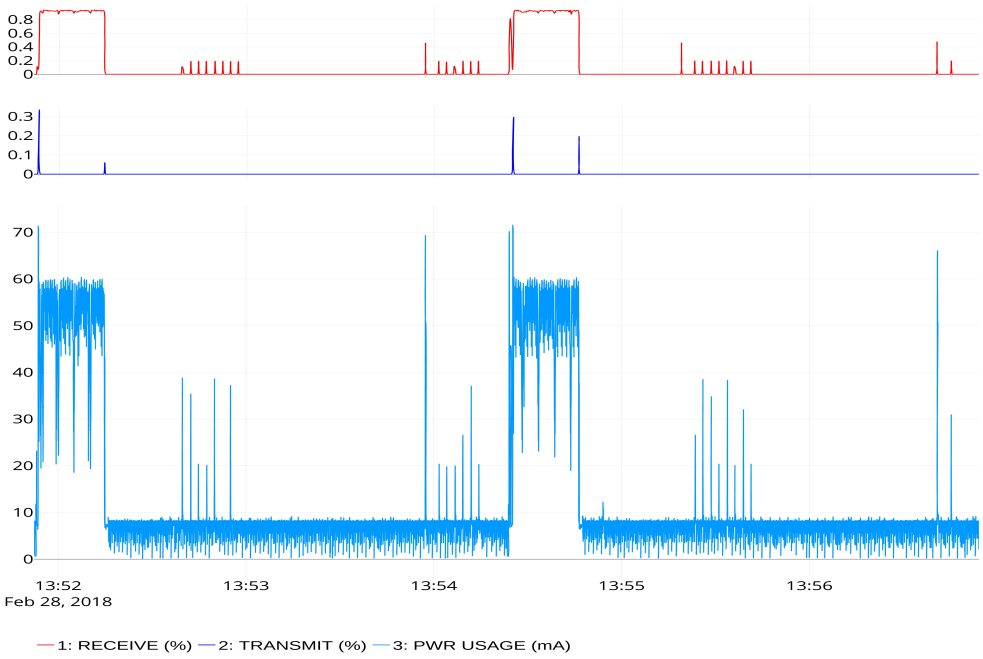


Figure 5.3: The figure displays normal behavior with two transmits over 5 minutes at Q-Free with Telenor. See figure 5.17 on page 73, or visit webapp [6], for more details.

5.1.2 Transmit power spike

Disregarding signal power level, one thing which was revealed after performing the tests was that there was often a spike of power usage up to $230mA$. Not all graphs are included in this paper, but visit henninghaakonsen.me and view results tagged with 60x1 for Telenor for example. You will see this spike in many of the graphs and it is an interesting finding thanks to the precision multimeter. The context of the occurrences is when the device is in RRC connected mode, hence in active communication with the network - either UL or DL. As explained in paragraph 4.3.2 on page 52, a NACK is sent with $+23dBm$ transmit power, hence it is likely that this is what is happening at these spikes. This behavior is not mentioned in the specification, and at the moment of the tests there were more instances of this behavior than without. This behavior was also present in Telia's network, so it is possible that this is normal procedure in a transmit process. It is however somewhat strange that this happens with transmits when the signal power is excellent, since one would presume that this practice would only be in case of bad reception and signaling faults.

5.1.3 Loss of connection prior to transmit

Some tests was performed in Telenor's network at Q-Free with interval at 40 seconds with and without RAI. At many of these transmits I noticed that the device indicated network disconnection directly before transmitting the packet. This resulted in a period with ECL at 255 and higher power usage for a short period where the device reconnected to the network. The two figures 5.4 on the facing page and 5.5 on the next page, represents this behavior. In these tests, there is a period of reconnection to the network, followed by the actual transmit process. This pre-procedure of reconnecting to the network operates at a current of $45 - 55mA$ and is comparable to a RRC period or transmitting with very good reception. This behavior is regardless of the RAI flag, hence the power usage is a lot higher compared to the actual transmit process when using the RAI flag. This is a root problem users and developers would have a hard time finding, and could cause battery lifetime issues. Using **power_calculator.py** I have calculated the power usage over the pre transmit period, see result in equation 5.1. This is actually $0.1764/0.034 = 5.2$ times higher than a normal transmit(see webapp [41]) with RAI set and excellent signal power. For each occurence of this extra procedure, it shortens the battery lifetime with approximately 5 hours, given that the application transmits every hour.

$$\begin{aligned} From &: 2018-03-07 \quad 13:36:14.415420 \\ To &: 2018-03-07 \quad 13:36:19.022218 \\ ((41.793363329583805mA * 3.3V/60/60) * 4.606798S) &= 0.176489mWh \end{aligned} \tag{5.1}$$

Figure 5.4: The figure displays one transmit over 40 seconds at Q-Free with Telenor, with device logging. See figure 5.19 on page 75, or visit webapp [1], for more details.

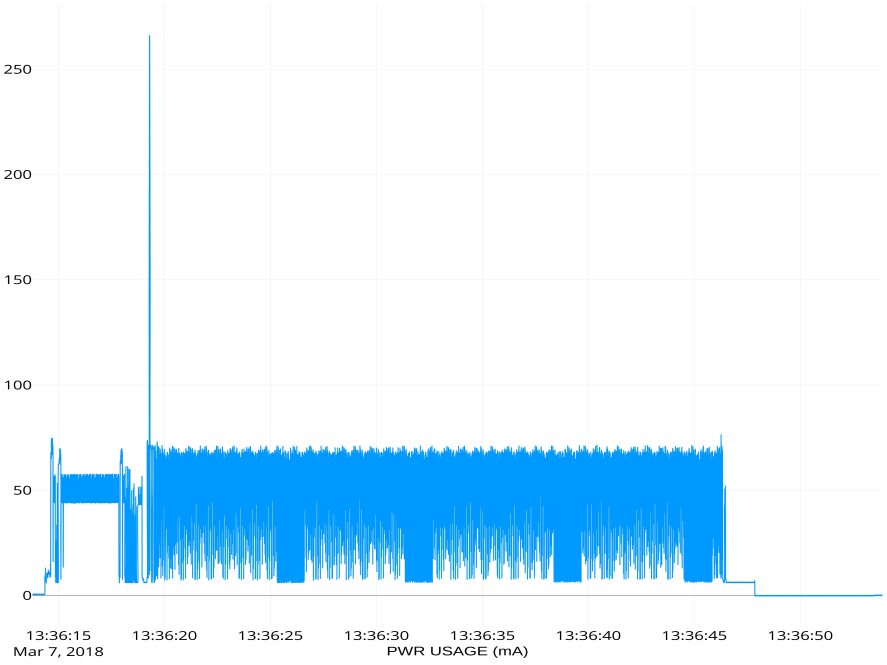


Figure 5.5: The figure displays one transmit over 40 seconds at Q-Free with Telenor, without device logging. Visit webapp [2], for more details.

Because of this bug, there were some problems related to the long-term tests so further investigation was required. The device would indicate network disconnection and reconnection, resulting in ECL being 255 and receive/transmit counters were reset. Looking into the issue by monitoring the transmit process towards the server it became clear that most of the time the device did not try to reconnect to the network. The behavior is not expected and is most likely related to the hardware and software of the development kit. The conclusion is that the bug does introduce reconnection some times, but it is UE related and not network related. This is also based on what was described in section 2.5 on page 26, that the chips produced at the point of the tests were not production ready, but good enough to test the networks.

5.1.4 Signal power and ECL

Depending on the devices signal power the ECL value should adjust accordingly. As discussed in section 2.4.7 on page 24, the network provider configures thresholds for ECL and the results have mostly followed these thresholds as expected. The two figures 5.6 on the next page and 5.7 on page 62, shows results from one transmit of 50 bytes over a period of 60 seconds with RAI set and describes the behavior of different ECL values.

You should focus on the two tops at the beginning of the figures[1]. The first figure indicates good signal power, but at transmit time the transmit power is bumped up from $+9dBm$ to $+23dBm$, maybe because of a NACK. The transmit graph shows that the transmit process is relatively short, only covering approximately 1.5 seconds. In addition the device was actually only active for 15-20% of that time, resulting in actual transmit time of ~0.3 seconds. The following equation 5.2, shows the power usage of the transmit with ECL 0 using the program **power_calculator.py**.

$$\begin{aligned} From &: 2018-03-1612:35:55.90 \\ To &: 2018-03-1612:35:57.54 \\ ((24.77mA * 3.3V/60/60) * 1.64s) &= 0.037mWh \end{aligned} \tag{5.2}$$

This next figure is showing a transmit where the signal power is poor, hence the ECL is 2. Not surprisingly the transmit power is at $+23dBm$ and there is a clear difference compared to the previous result. First of all the transmit process endures longer, around 5 seconds, hence the device is retransmitting the packet to ensure that it will be received at the eNB. The following equation 5.3 on the facing page, shows the power usage of the transmit with ECL 2.

[1]Please visit the web app (link is supplied in the figures) to zoom into the specific periods

$$\begin{aligned} From &: 2018-03-1613:13:42.28 \\ To &: 2018-03-1613:13:47.91 \\ ((80.84mA * 3.3V/60/60) * 5.63s) &= 0.417mWh \end{aligned} \tag{5.3}$$

By comparing the two transmit operations we see that the second transmit uses ~11 times as much power as the first. In my experience the first transmit is the most accurate towards what actually is going on most of the time, but if the UE has bad signal power the battery lifetime will be heavily affected. In section 2.4.8 on page 25, I described the theoretical disadvantage of different ECL values, however I think that this comparison is more accurate towards normal behavior. The results from Telia was included for this observation since the results from the two networks were very similar.

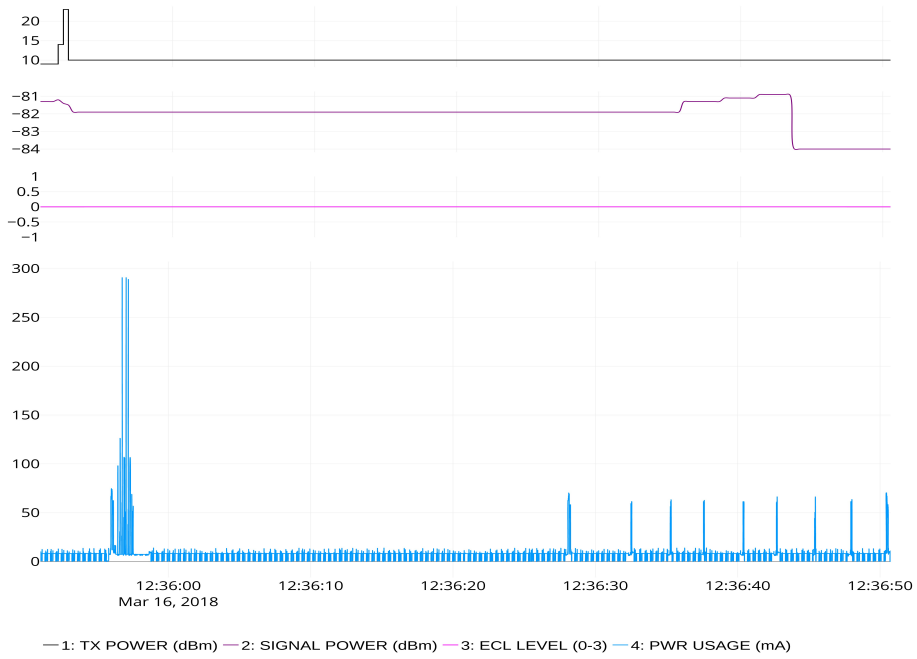


Figure 5.6: The figure displays one transmit over 60 seconds at UiO with Telia, with ECL 0. See figure 5.20 on page 76, or visit webapp [3], for more details.

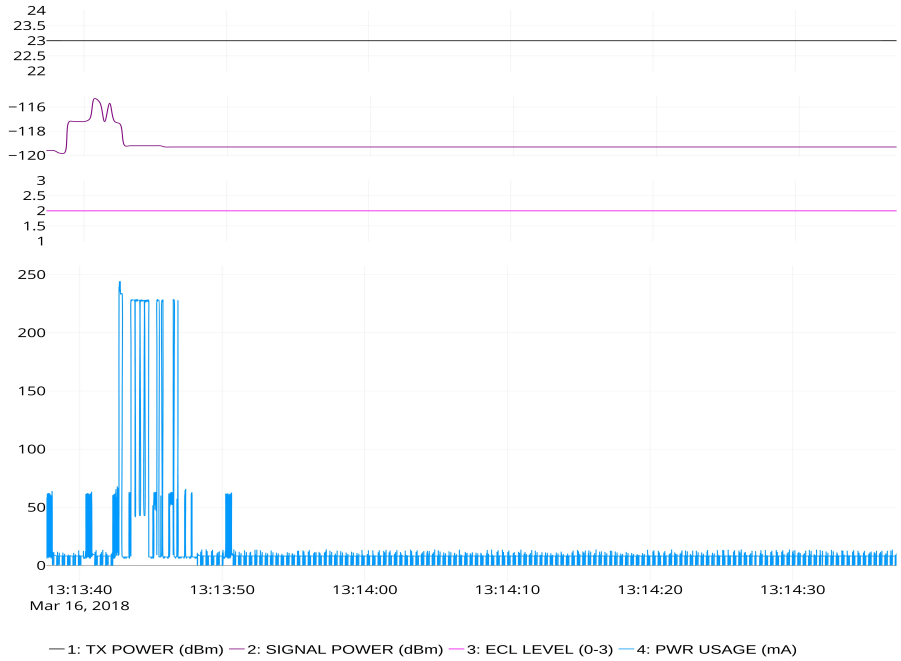


Figure 5.7: The figure displays one transmit over 60 seconds at UiO with Telia, with ECL 2. See figure 5.21 on page 77, or visit webapp [4], for more details.

5.1.5 Transmit comparison

In an attempt to compare the two networks I did many similar tests with both networks. All transmits used the same setup and the signal power was altered using attenuators. One thing to note is that since the tests were performed at Q-Free where the signal power of Telenor's network is exceptional, the device never entered ECL 1 or 2 with Telenor's network. There was however an interesting result and we will begin looking at figure 5.8 on the facing page, from the transmits without RAI set.

There are a few things to notice in this chart. First of all there are two groupings, one with Telenor's transmits and another with Telia's. As we have seen before Telia's transmits use less time in RRC connected mode, hence lowering the power usage. Telia uses around 50% less power at the first two transmits, regardless of byte size. It is not until the last transmit where Telia's transmits uses more power, and this is because the device has entered ECL 2 and retransmits occurs. The next section will focus on packet size, but there are indications that packet sizes does influence the power usage to some degree.

Moving over to figure 5.9 on the next page, with transmits with RAI set, the results are more similar between the two network providers when the device is in ECL 0. By looking closely at the first transmits from Telia there is a bigger gap between packet sizes, and this is mainly because the

RRC period is not influencing the results, thus gaining more information about the actual transmit process. You can view the actual transmit graphs at henninghaakonsen.me where they are tagged with **60_1**.

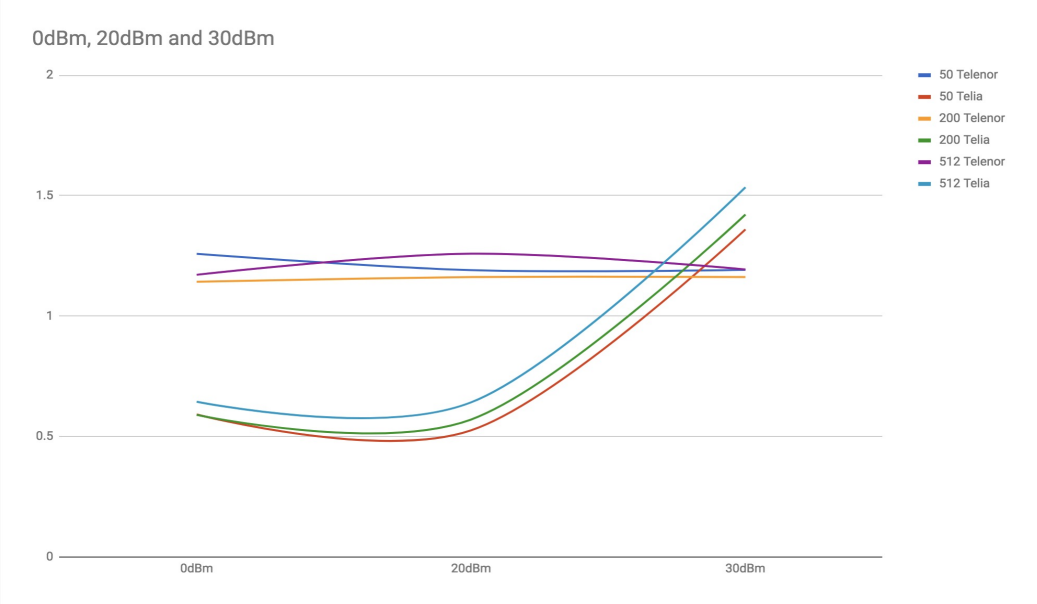


Figure 5.8: The figure compares transmits without RAI. The y-axis shows how much power was used, in mWh.

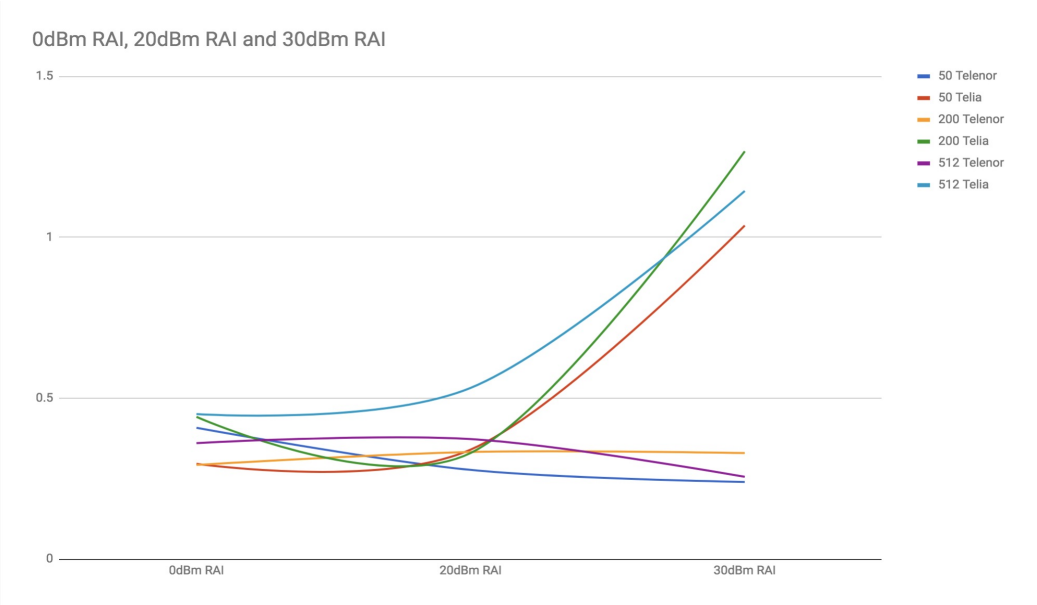


Figure 5.9: The figure compares transmits with RAI. The y-axis shows how much power was used, in mWh.

5.1.6 Packet size

The typical packet size for an IoT device is often small, typically under 100 bytes, but for some cases bigger packets might be required and I have performed a specific test to see what the packet size does with a transmit. The test is performed with the program **nbiot_labtest_details.py** and the figures, 5.10 on the facing page and 5.11 on the next page, displays two executions of the program - one with RAI and one without. It was important to test both transmit modes to see how much the transmit process with different packet sizes costs in terms of power usage.

Looking at the first figure, transmits without RAI, you see that the whole transmit process without RAI endures approximately 20-30 seconds, while the actual transmit process only takes 0.5-2 seconds depending on the signal power level. Investigation all aggregated lines it is clear that the bigger part of the power consumption comes from overhead of the RRC process. There is a slight increase in power usage while increasing the packet size, but if your application transmits packets without RAI flag the packet size is not crucial to the power usage.

Moving over to the next graph there is a noticeable difference. Not surprisingly there is a general decrease in power usage when using RAI. The average aggregated value of packets transmitted with 25 bytes, shows a decrease of 240%. This results in less overhead related to the transmit, hence the power usage from the transmit has bigger influence on the graphs. Looking at the average line in pink there is a steady increase in power usage related to the packet size. Comparing 25 and 512 bytes, there is a power increase of 217% which sounds logical since a bigger payload requires several data packets. It is interesting to look at the aggregated sum of each byte size category. According to [5], the Maximum Packet Size (MPS) is 680 bits, which is 85 bytes. Looking at the aggregated sum line two spikes appear which is likely related to the MPS. There is a spike at 100 bytes, which is directly after the MPS. There is another spike from 400 to 512 bytes, which may be related to the MPS of the chip which is 512 bytes.

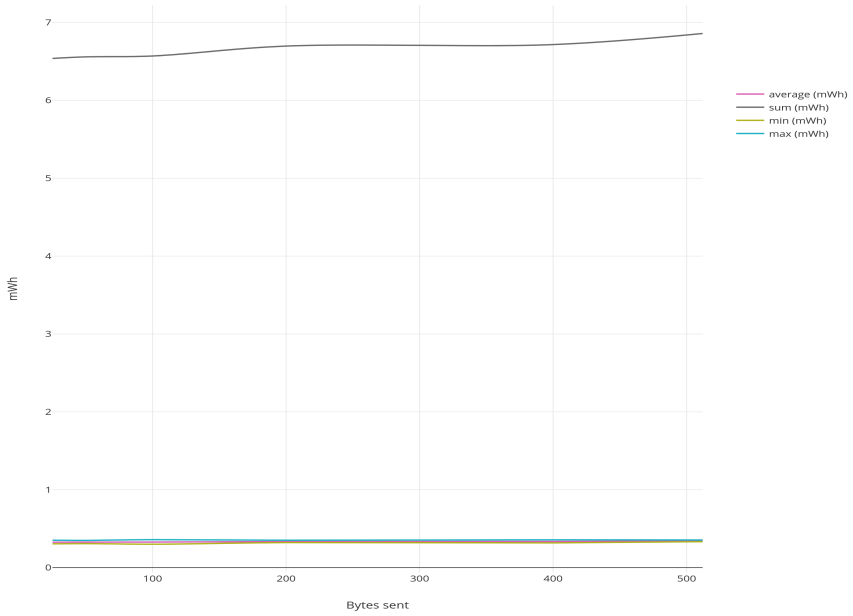


Figure 5.10: The figure compares different packet sizes without RAI set. Visit webapp [15], for more details.

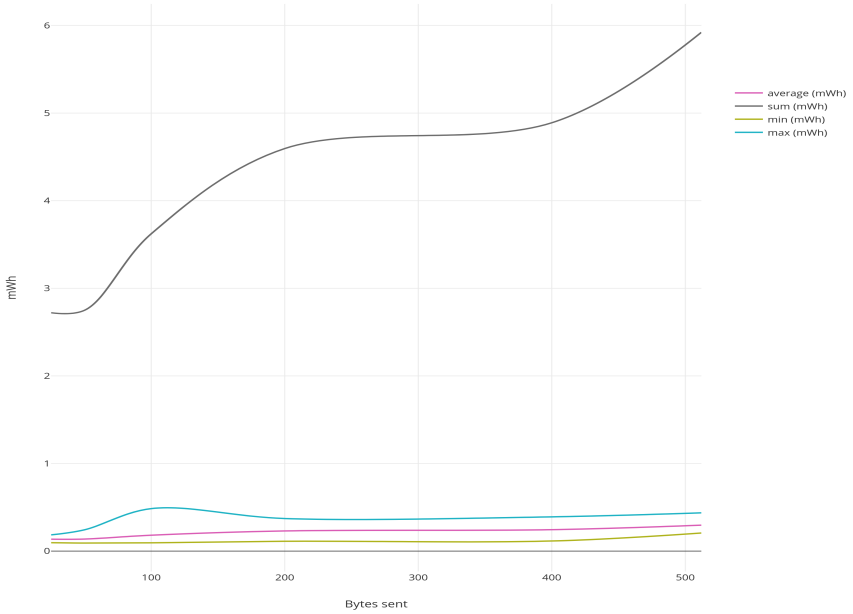


Figure 5.11: The figure compares different packet sizes with RAI set. Visit webapp [14], for more details.

5.1.7 Sensor reboot

There are a number of reasons why the device would perform a reboot, especially if the application is operating over several years. One reason for a reboot is because of some software or hardware fault, which could occur even though it should not happen. Another reason is that the developer reboots the device if it has stalled or some logic kicks in. The main reason to investigate what happens in a reboot is to see how long to network connection takes. If this process is long and cumbersome it will affect the power usage, hence the battery lifetime decreases. I will elaborate why and when your application should consider rebooting the device in section 7.1 on page 89.

I performed several reboot tests with both network providers and different signal power levels. I have included two results, one from Telenor and one from Telia. Since the device reboots and the program looses the connection to the chip it is not possible to log the status of the device with **NUESTATS** so you will only be able to see the power usage in this test.

Figure 5.12 on the next page, presents the first example which is in Telenor's network without any attenuators. The graph displays 120 seconds which includes the whole process of rebooting and connecting to the network. The first thing to notice is the long period of what looks like RRC connected mode. Following is what looks like the actual connection process about half way into the graph and ending with a RRC period until the timer runs out and the device enters PSM mode. The whole process uses $5.1mWh$, which is the same as around 28 transmits with 100 bytes payload with RAI set using the results from the test of packet size. Assuming one transmit every hour, this reboot process uses approximately the same amount of power as a days worth of uptime.

Moving over to Telia's result without any attenuators, see figure 5.13 on page 68, you might notice that this result only covers 60 seconds. This is because the reboot process using Telia's network was much shorter and since the reboot process is the focus, the log duration is kept to the minimum. Directly after reboot the device performs a set of transmits, followed by a period with RRC connected mode. After this process the device enters PSM and the device has connected to the network. The whole process uses $0.55mWh$, which is around ten times less than with Telenor's network. It is interesting to see the differences the network provider introduces to the network connection process, since this is mainly due to network configurations. One explanation of the lower power consumption is likely related to the power consumption during the RRC period. Section 5.1.1 on page 54, showed that Telia uses around 60% less time in RRC connected mode, hence reducing the power consumption drastically. In addition, the duration of the connection process is twice as long with Telenor. The results are consistent throughout all reboot tests

and gives an indication that Telia has used the configuration to reduce the power consumption on reboot and in these RRC periods.

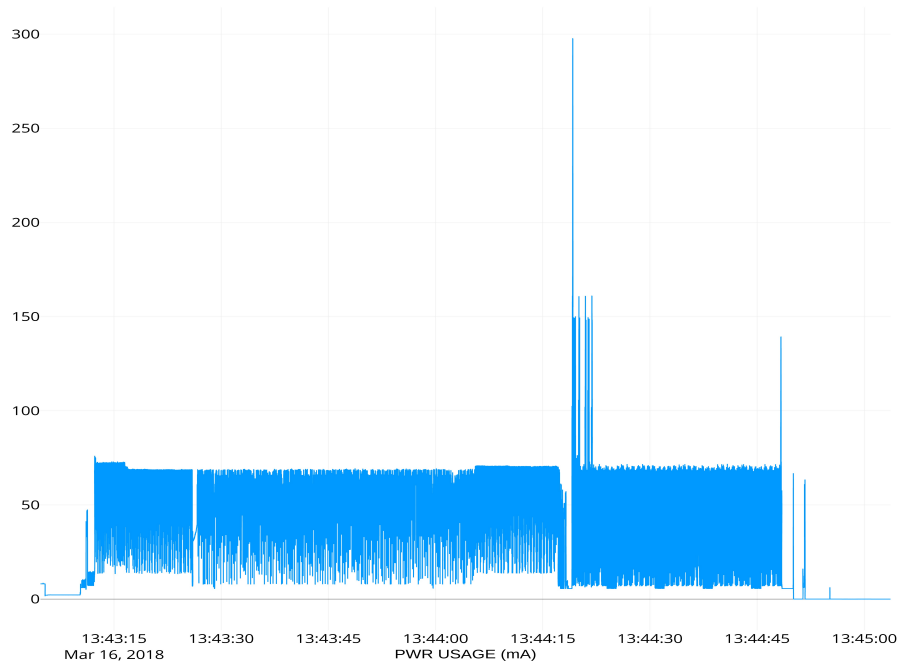


Figure 5.12: The figure shows a reboot of the development kit with Telenor. Visit webapp [44], for more details.

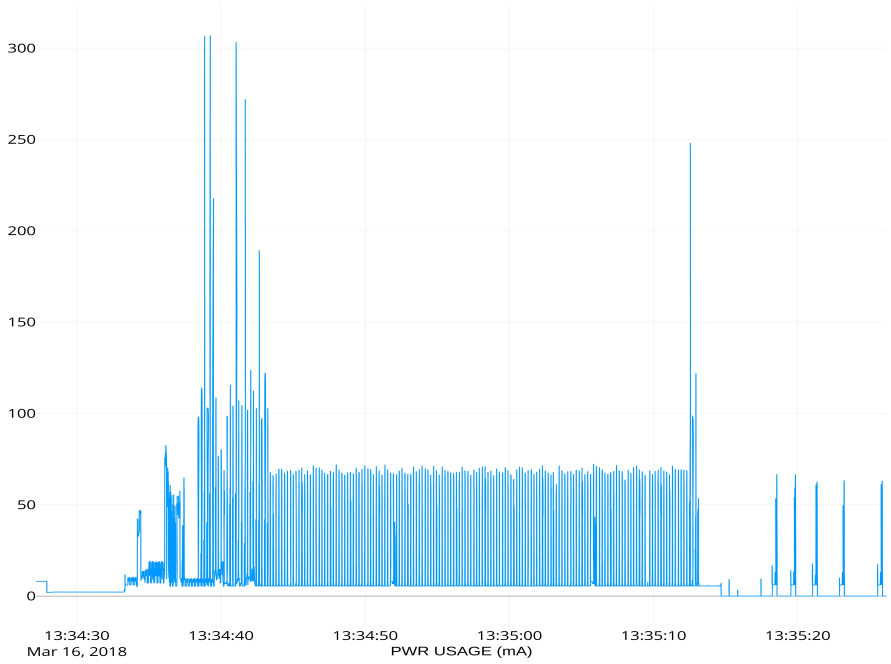


Figure 5.13: The figure shows a reboot of the development kit with Telia. Visit webapp [45], for more details.

5.1.8 Downtime test

If the network is down because of a power outage or similar breakdowns the UE might be disconnected from the network if there are no other eNB's in the close proximity. There was a period in the long-term tests where the device actually lost connection to the network, even though the signal strength showed otherwise. In figure 5.14 on the facing page, you can see a longer monitoring sequence. At first the device transmits data, but after a short while it is disconnected and the subsequent results gives an indication of what happens if the device looses connection. The receive/transmit timers does not change, hence it looks like the device does not use any effort trying to reconnect to the network. I will discuss different approaches to this problem in section 7.1 on page 89.

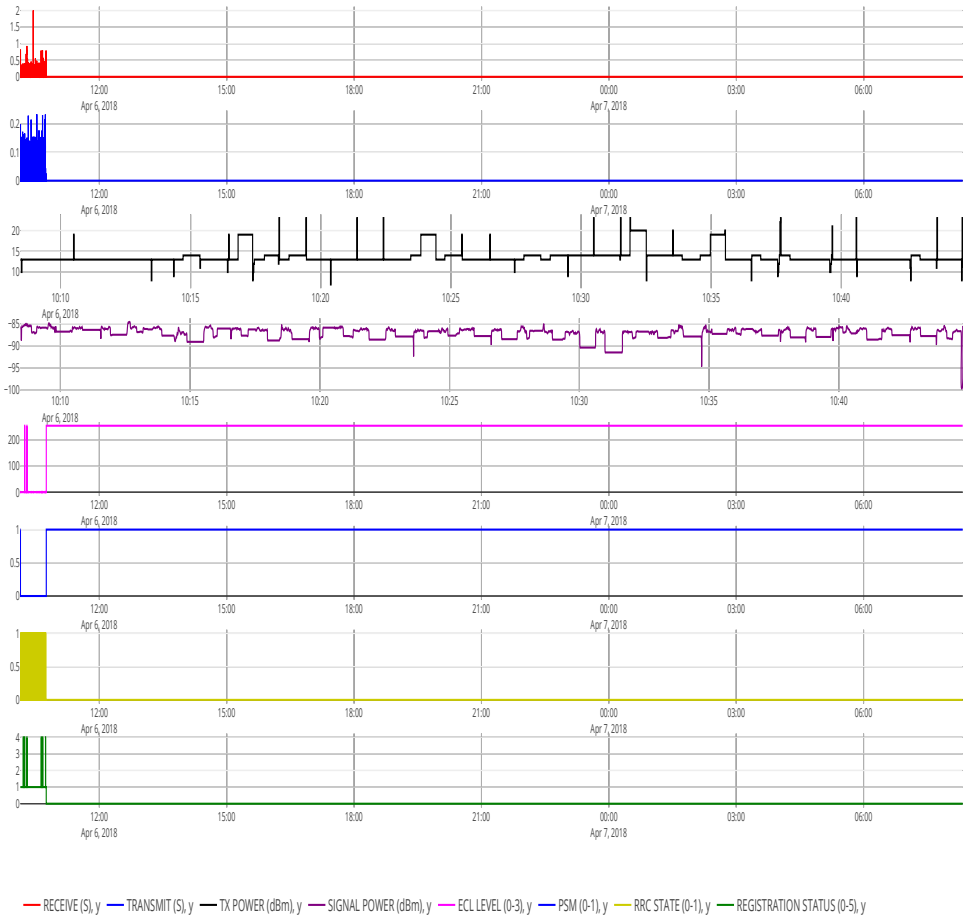


Figure 5.14: Example of disconnection from the network

5.2 Short-term figures

5.2.1 General

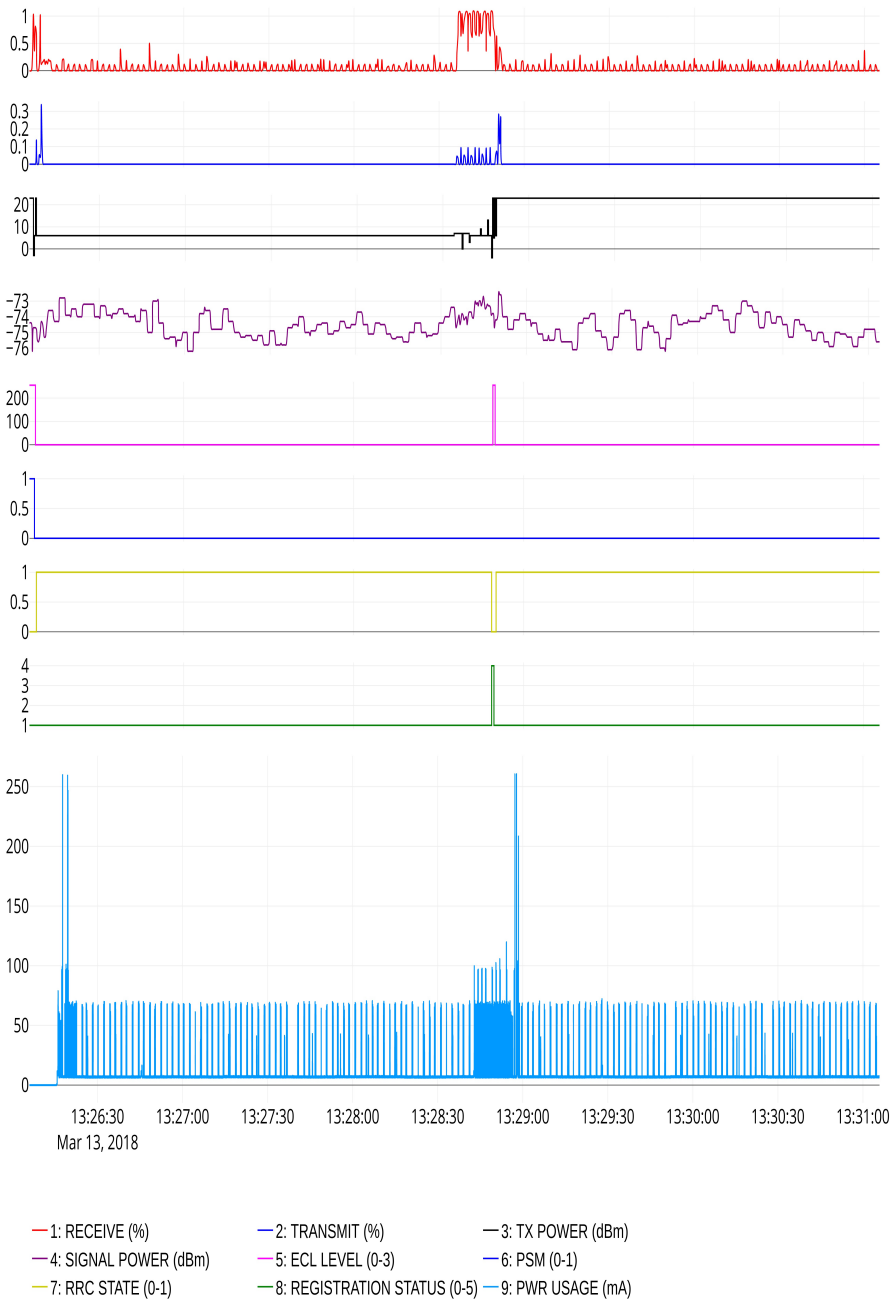


Figure 5.15: Short-term figure - unusual behavior, Telia

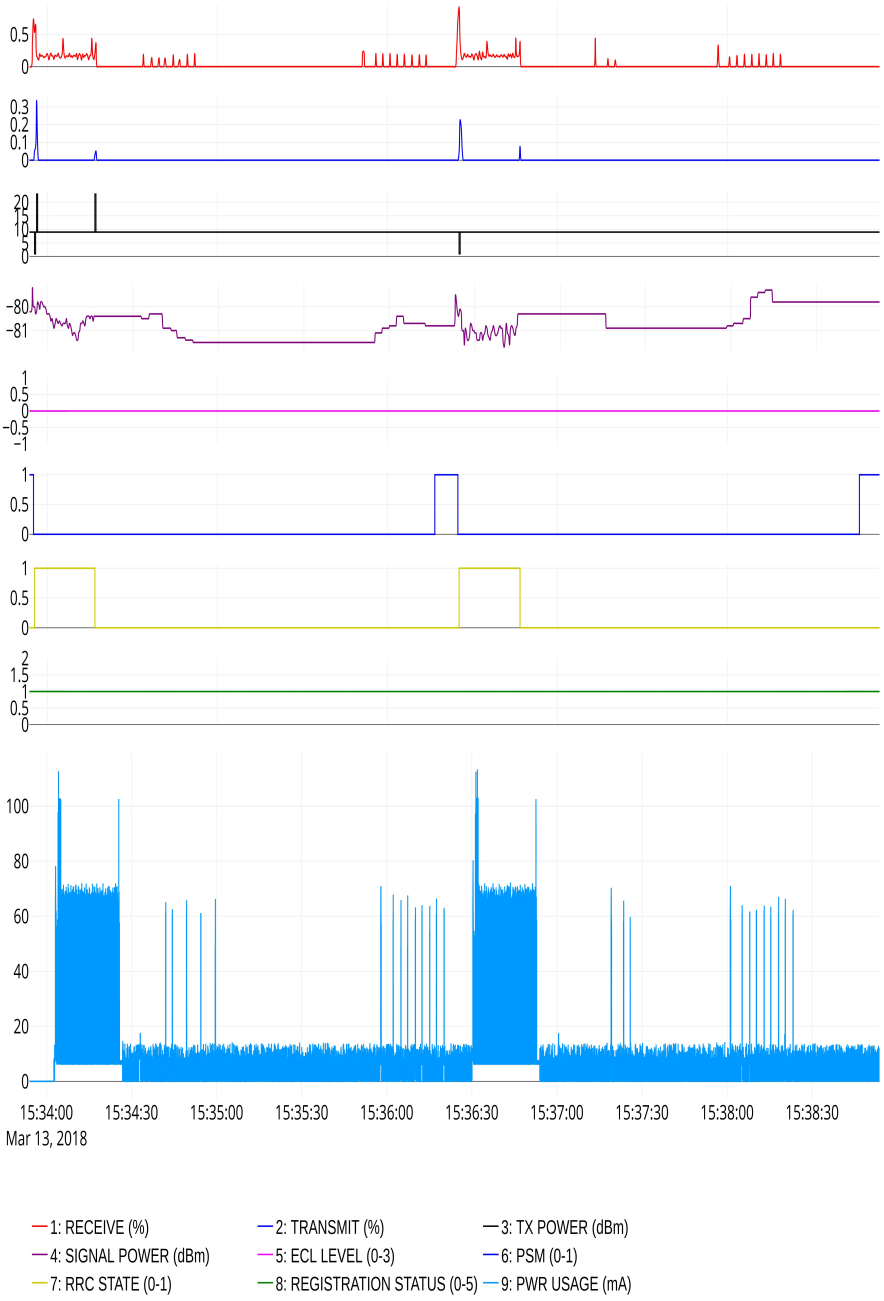


Figure 5.16: Short-term figure - normal behavior, Telia

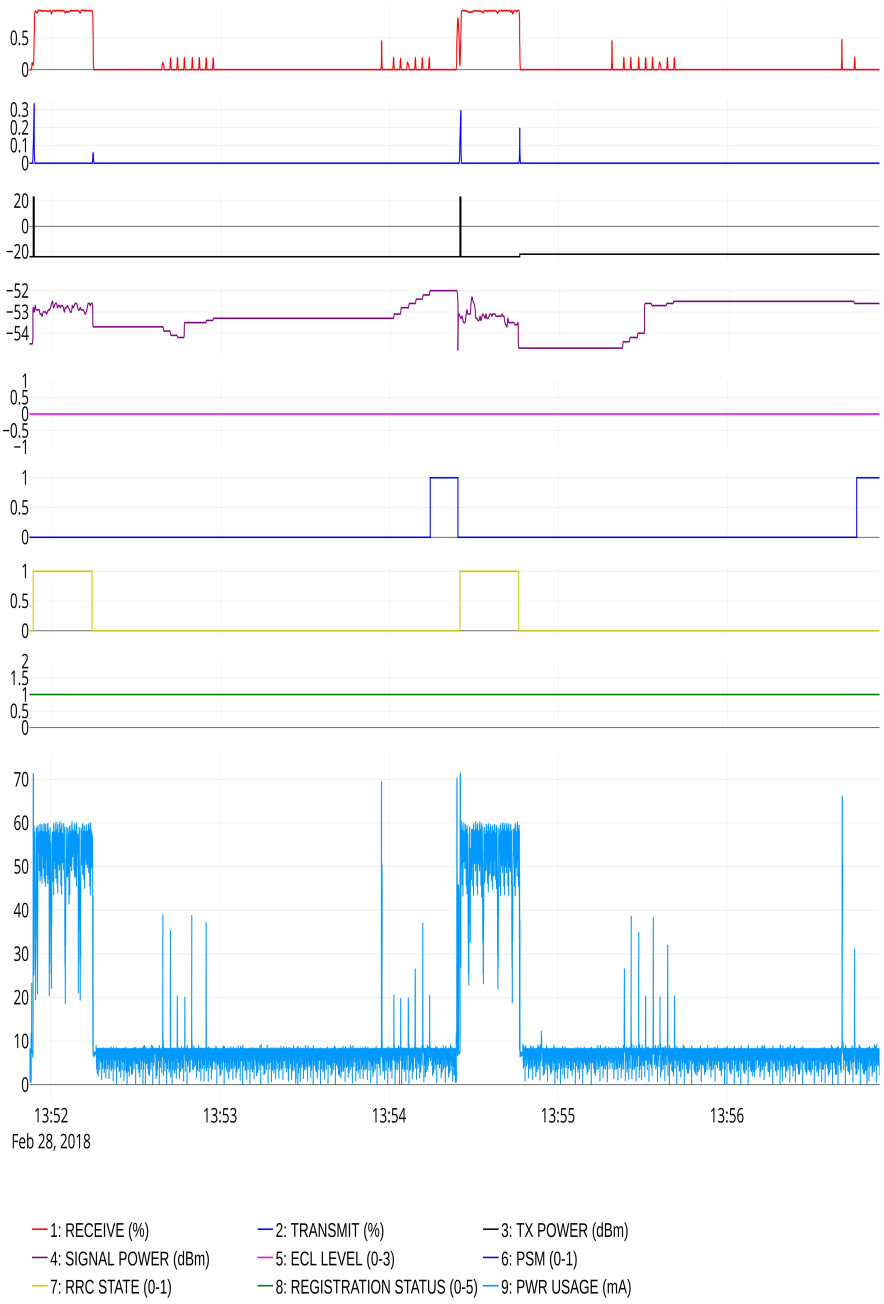


Figure 5.17: Short-term figure - normal behavior, Telenor

5.2.2 Downtime prior to transmit

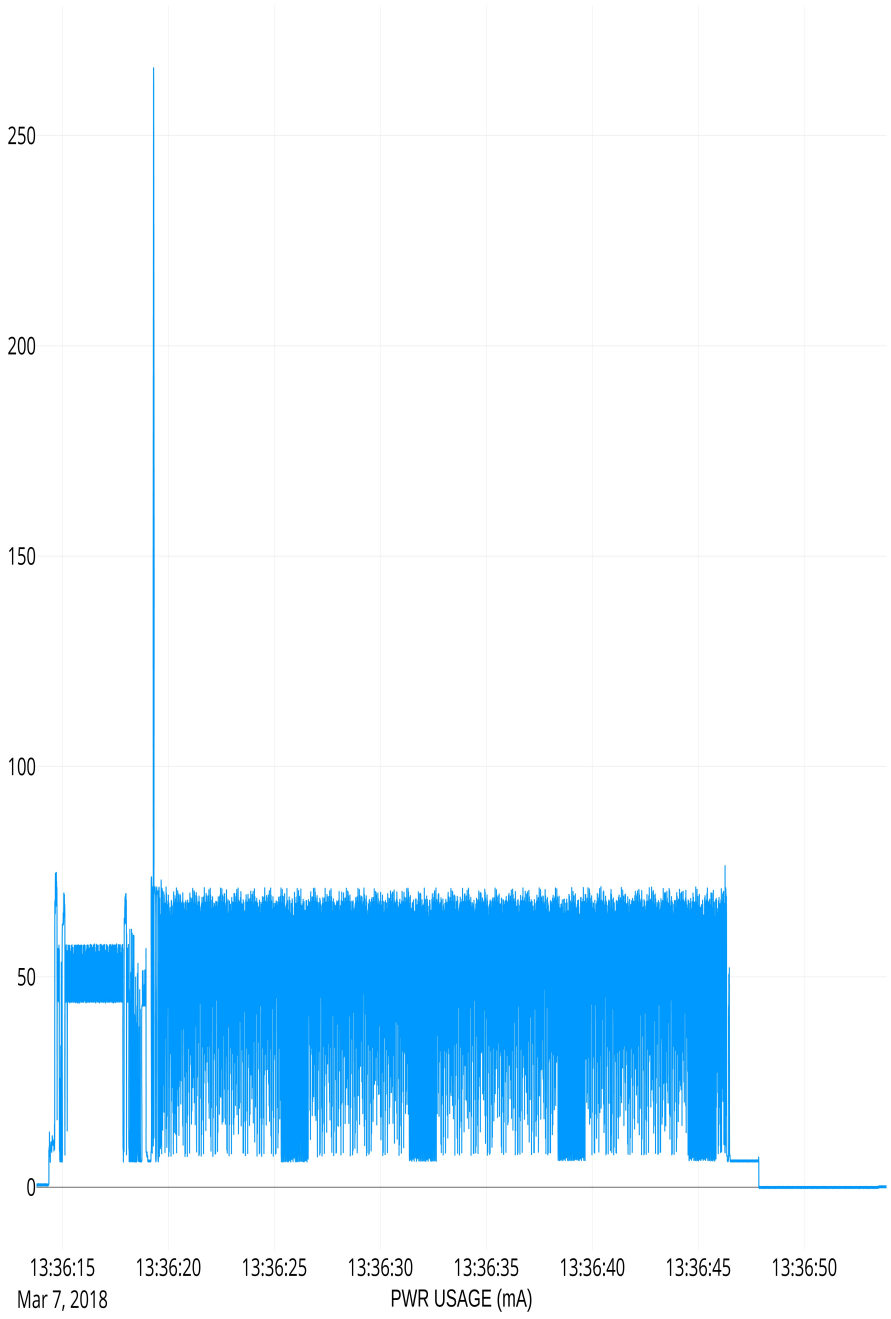


Figure 5.18: Short-term figure - loss of connection, without device logging

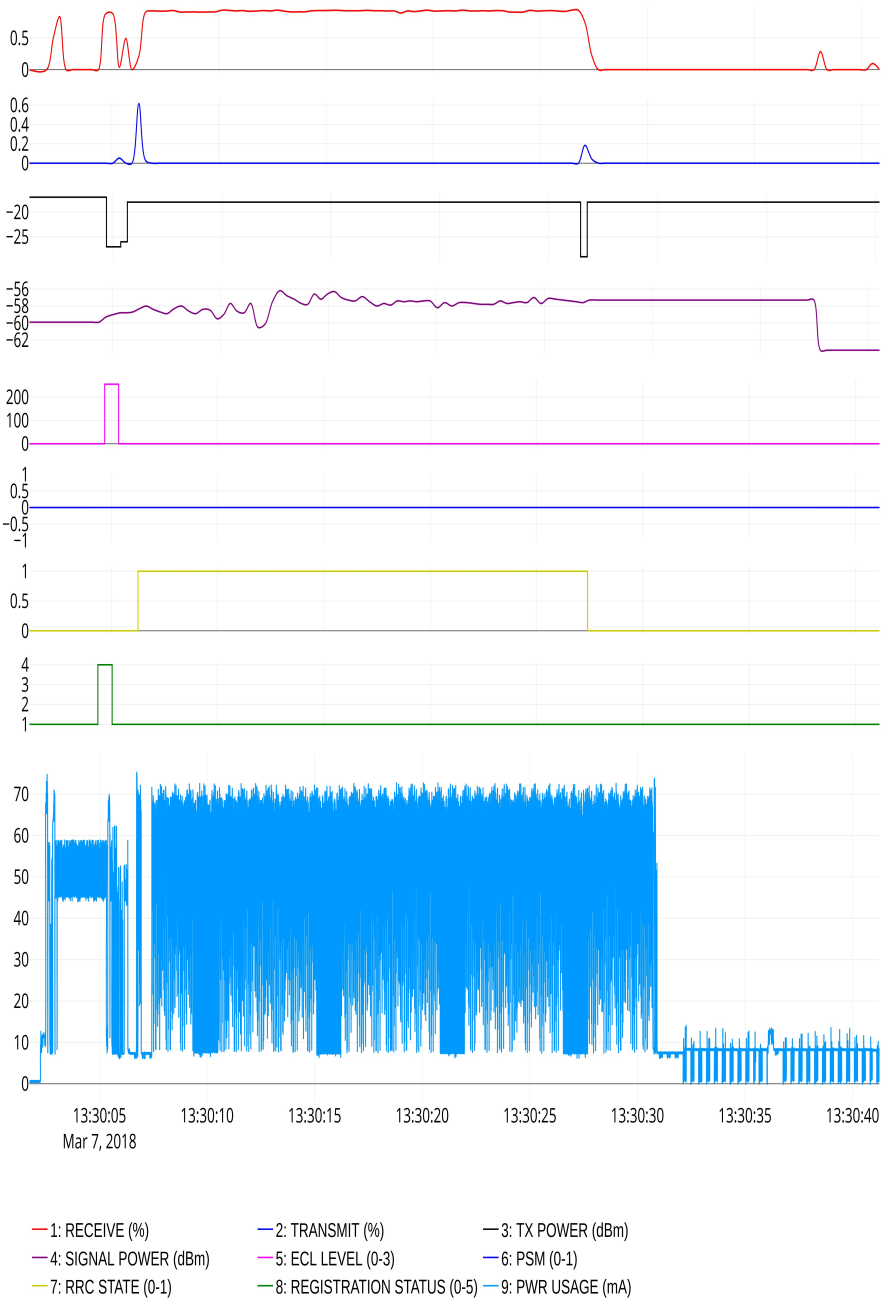


Figure 5.19: Short-term figure - loss of connection, with device logging

5.2.3 ECL

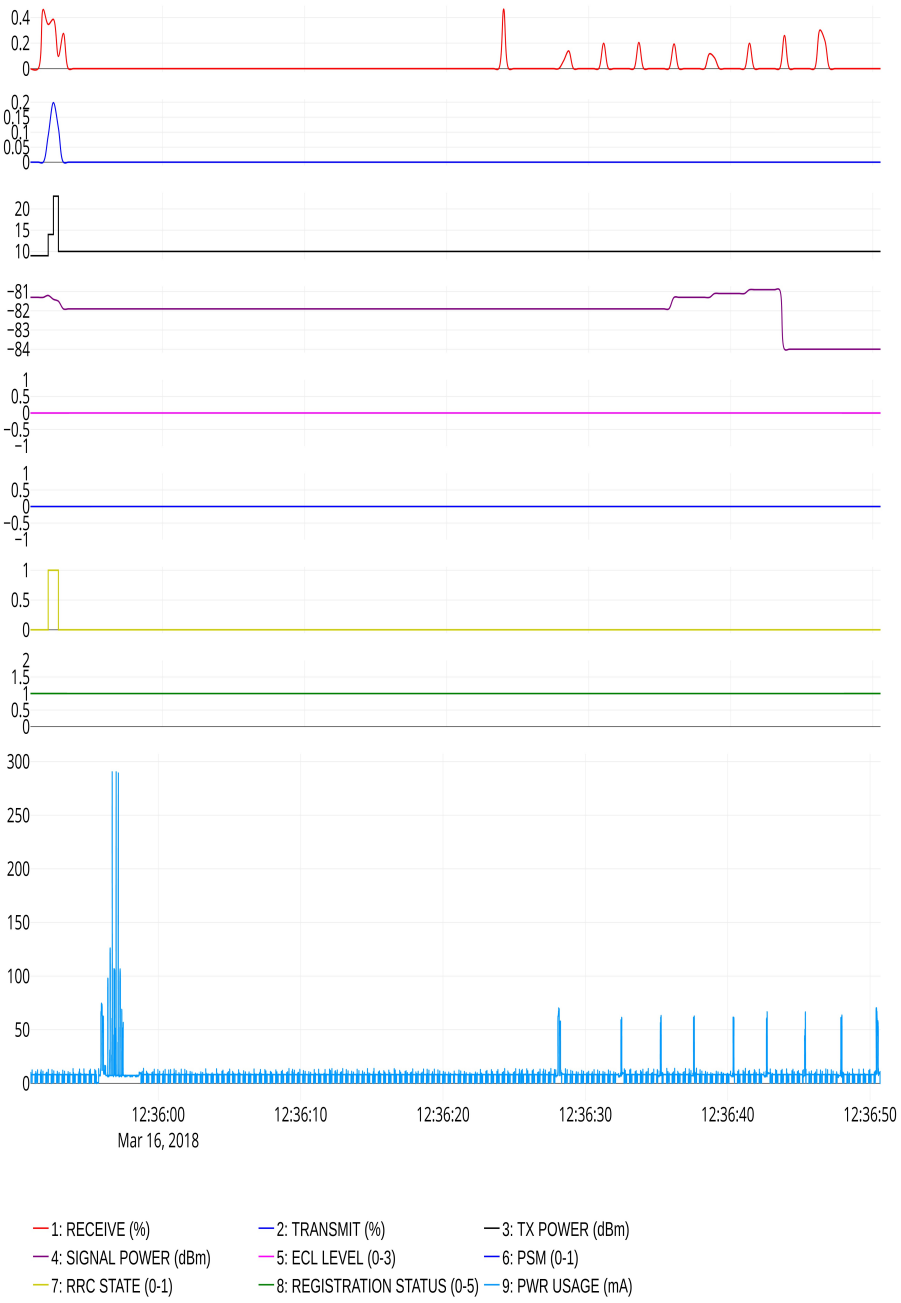


Figure 5.20: Short-term figure - ECL 0

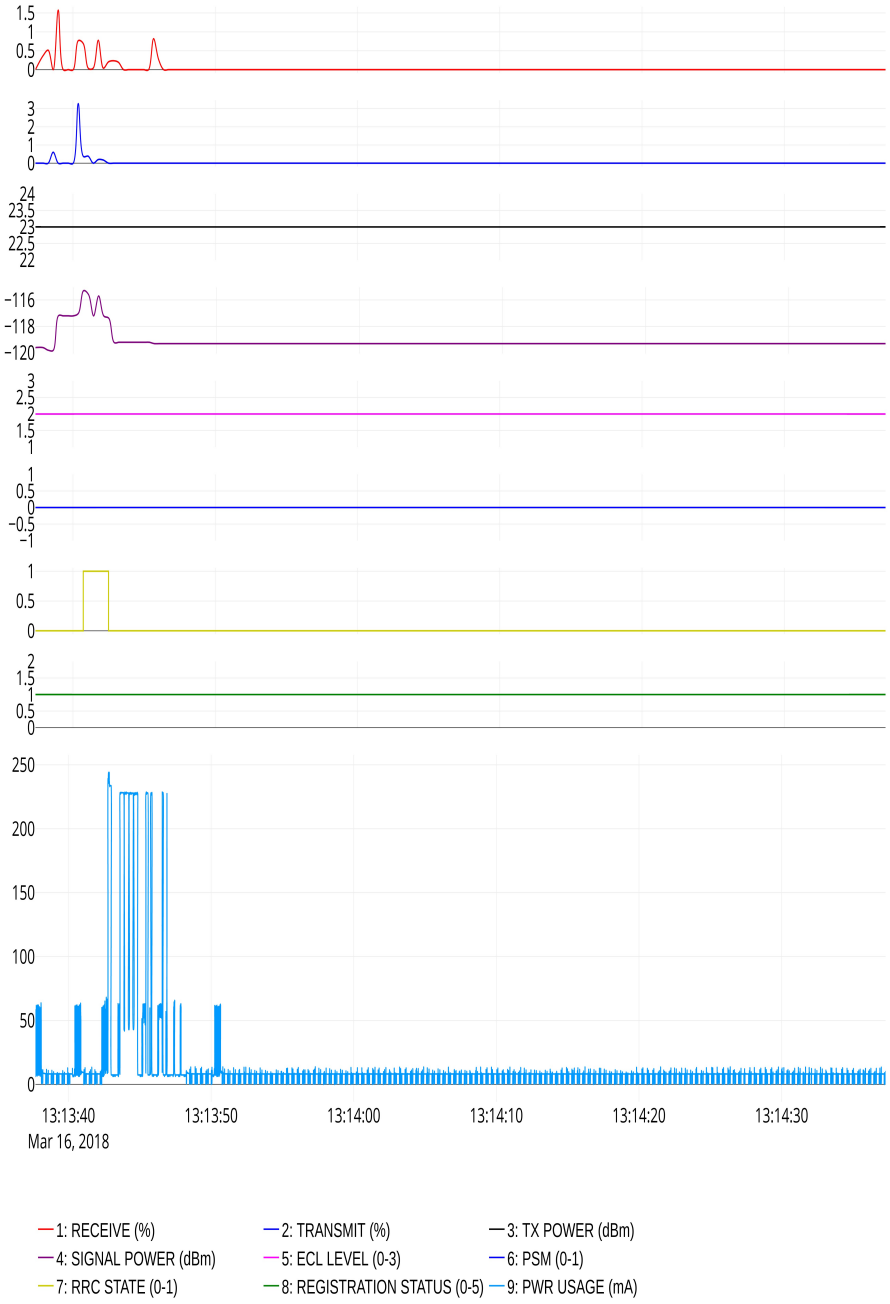


Figure 5.21: Short-term figure - ECL 2

5.3 Long-term tests

A device operating for a long time is prone to many bugs and software quality is essential. This section gives an overview of how the network handles continuous transmits over a longer period. I have tested the device at all locations with both network providers where the device

managed to connect to the network. I define long-term tests as a set of operations over a long period, typically one or several days. When trying to test many scenarios, the test period per network provider and location was limited. However, the results are comparable to the short-term tests and there are clear indications of the networks pros and cons. The reason for the long-term tests is to monitor the behavior of the chip over a longer period, which gives statistics at a more abstract level, and can be combined with the short-term tests. With this relation I will give you an overview of the most common states of a device and how different aspects reflect the behavior. I had some problems logging correct behavior of the device. It seems that at random points in time it looks as if the device looses connection to the network resulting in ECL set to 255, and receive/transmit timers are reset. It is at this stage unclear what evokes this behavior, but one reason might be related to PSM periods as the device shuts down all internal processes only keeping the clock and some logic alive. There were some progress to this problem in late March when I used the program **nbiot_labtest.py**, for the long-term tests. This program constantly pulls the chip for statistical data, hence it is reasonable to assume that the device is more active. Even with this program there were signs of the same behavior and because of this abnormality some of the results are harder to make sense of, especially receive/transmit time. However, the results were very much like the results from the short-term tests and I will use the good results to describe how the network performs. Most of the figures included in this section is taken from the web application so if you are unfamiliar with the format please refer to section 4.3.1 on page 49, for a more detailed description of the graphs. For the best experience I advice you to use the web application to view the data, but the most interesting findings will be included in each apprioriate section.

5.3.1 Signal power and latency

One of the key features of NB IoT is increased link budget. Devices should have good reception in normally bad coverage areas, such as tight city areas and far away mountain sides. The transmit power is $+23dBm$ which is very good, considering that a theoretical $+3dBm$ step is actually doubling in power output. The signal power is not only related to the distance between the UE and the base station, but also the obstacles between. The map 3.3 on page 31, shows that the distance between the UE and the eNB's is low, but the signal power was quite different. With Telenor, the reception was normally at $-60dBm$, while with Telia, the reception was normally at $-80dBm$. The main reason for the big difference is due to what lies between the sender and receiver. There are buildings and obsticles between Q-Free and Telia's eNB, but there is very

few obsticles between Q-Free and Telenor's eNB. It is reasonable to assume that the two network providers use the same technology to offer NB IoT, but it is probably not from the same producer which might explain the big difference in reception as well.

The following two sections gives you insight in two longer transmit sequences. Because of the issues related with the long-term test I have mostly focused on one of the sequences with Telenor since the tests revealed that their network is the most stable of the two.

Telenor

This section will present a transmit session from UiO over 5 days, 23.03.18-28.03.18. The figures 5.22 on page 81 and 5.23 on page 82, is taken from the web application and will be used to point out certain aspects of the sequence. The first section displays two steady lines with signal power around $-95dBm$ and latency between 1.5-3.5 seconds on average. This classifies as quite good signal and resembles normal behavior. A thing I noticed is the flow of the latency line. There is a clear interval between the high latency spikes and the low. I have not been able to find the origin of this behavior, but it has most likely something to do with the process in the core network since both the server and the client in this setup behaves the same way at all times. Another indication that it originates from the core network is that my supervisor, Q-Free's R&D manager, Ola Martin, also has seen this behavior in his tests[38]. Another interesting thing to view is receive/transmit time in the second figure. The transmit graph shows that a normal transmit uses approximately 0.3 seconds which is relatable to the short-term tests. Also looking at the receive time graph the normal receive time between two transmits are around 21 seconds since these transmits were not using the RAI flag. This is also very well in terms of what was presented in the short-term tests. This is useful information since it is now clear that this is not only a one time occurrence, but the normal behavior of a receive/transmit process without RAI set.

The same period shows three clear latency spikes up towards 10 seconds without any indication of bad signal power. However, at each latency spike a similar spike in receive time is present and this indicates that something happened in the network which the device needed to listen in on, hence the transmit towards the server is delayed. The specification states a maximum latency of 10 seconds, so these transmits are on the edge of what is normal behavior. After a while, around 16:00 23.03, the trend shifts. The signal power is the same, but the latency suddenly fluctuates up towards 5-6 seconds and this happens with the same interval flow as previously. To try to see if the behavior is related to the test program I restarted the program at, 20:36 25.03, now with RAI flag set. The same behavior continued until I restarted the development kit at 14:00 26.03, where the signal power and latency normalizes to what classifies

as normal behavior. A thing you might have noticed is the drop in receive/transmit time. There is a noticeable behavior change in receive time when RAI is set and this is very logical since the device goes to sleep directly after the transmit process has finished. Even though the transmit process only covers ~0.3 seconds there are still some overhead related to the transmit process so the typical receive time for Telenor is 2-3 seconds, which is approximately ten times lower than with RAI not set. It is hard to see from the figure, but the transmit time is also reduced to some extent while using the RAI flag. This is related to lowering the network communication overhead of a transmit. The average transmit time with RAI set is approximately 0.15-0.2 seconds, which is around 50% less than without RAI set. Since the transmit process involves higher power usage a decrease in time usage is valuable.

From 10:00 to 12:00 26.03.18 three spikes in the receive time appear, which might indicate trouble with the connection towards the eNB. If the device detects loss of connection it will remain in RRC connected mode for a longer period, hence pulling data on a regular basis which consumes very much power.

At 12:14 27.03.18 I added a $20dBm$ attenuator to the development kit hence the signal power was reduced to around $-120dBm$. The reason why I add attenuators is because it is interesting to see how the network performs in these conditions, even though it will hopefully not be the case of most applications. First of all, the base latency is around the same, but the interval pointed out earlier is not as clear as before and the overall latency is higher. More interesting is to watch the receive/transmit time graphs, as they show drastic behavior changes. The transmit time graph shows that the device retransmits packets very often. Many transmits uses around 2 seconds, while others are between 4-8 seconds. This is close to what was described in section 2.4.8 on page 25, meaning that in addition to time spent in transmit mode the transmit it self also uses maximum power for each transmit. This behavior is extremely battery deficient and will cause poor lifetime.

Moving over to the graph with receive time indicates that the minimum receive time is still 2-3 seconds, but now the line is very unstable. There are more tendencies to spikes as the device has to use more time communicating with the network because of the poor reception. However, the receive time is not affected to the same degree as the transmit time and is kept under 10 seconds for most of the transmits. Looking at the two graphs the transmit time increased with approximately $2.5s/0.2s = 12.5$, while receive time only increased with approximately $5s/2.5s = 2$. This is not surprising given that the device retransmits packets frequently at ECL 1 and 2 to achieve higher receive rate at the eNB.

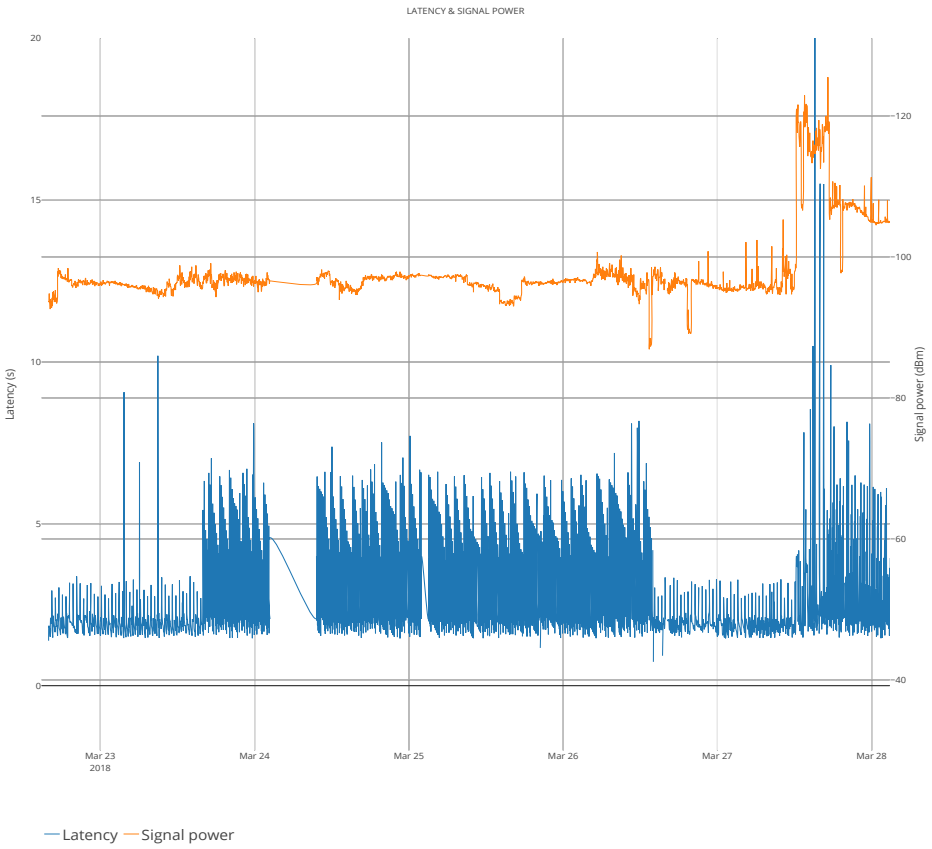


Figure 5.22: The figure displays a transmit session at UiO with Telenor over 5 days, 23.03.18-28.03.18, with the signal power and latency graph. Visit webapp, for more details.

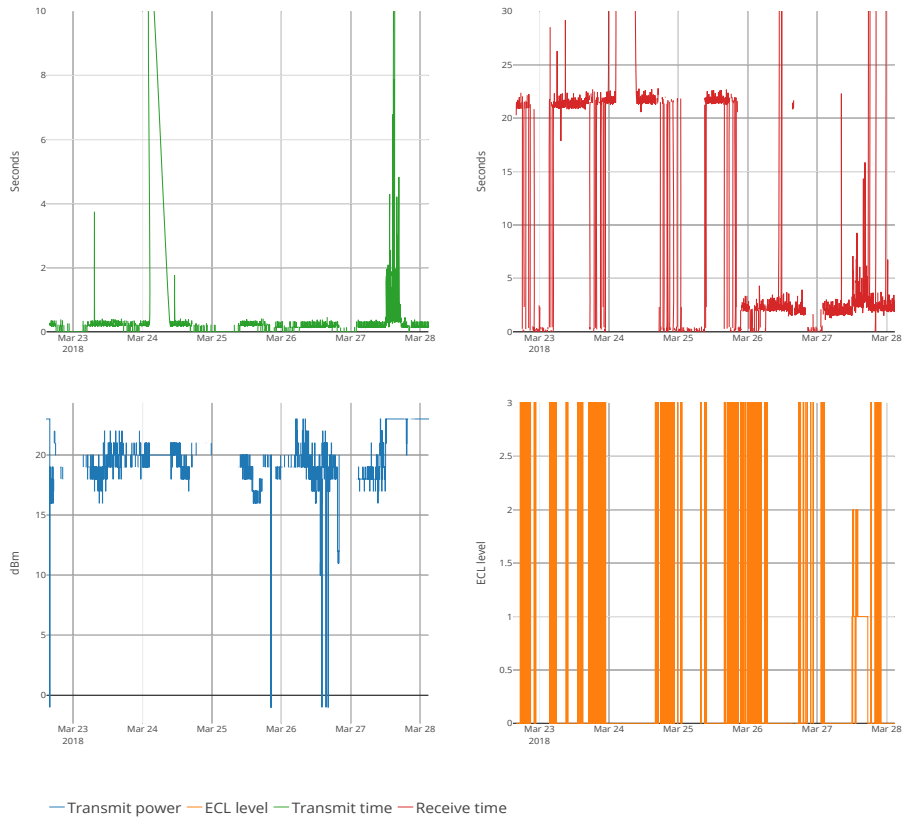


Figure 5.23: The figure displays a transmit session at UiO with Telenor over 5 days, 23.03.18-28.03.18, with the statistics graphs. Visit webapp, for more details.

Telia

Transitioning over to Telia I will present one of the sequences recorded with Telia's network at UiO, from 20.03.18 to 23.03.18. The setup is similar and there are obvious differences. The two figures 5.24 on the next page and 5.25 on page 84, illustrates the results and the beginning of the first figure shows that the graphs are comparable to Telenor's result. However, as you might have noticed most of the transmits are less stable, which affects the results. Since the network does not give any feedback, it is hard to state the reason for this instability, but referring to section 2.5 on page 26, we know that Telia deploys NB IoT in-band within LTE. As previously stated in section 2.2 on page 18, if there are many users in a certain area, this can cause poor performance, hence the results reflect what might be this kind of behavior.

In addition the problem with reconnection is worse with Telia's network. This is probably related to the instabilities we have seen. Since the results are degraded I want you to focus on the results from

the previous section as a kind of benchmark to what is the closest to stabile behavior at this moment. However, the results from Telia shows an improvement in time spent in receive mode with RAI not set. Looking closer at the last part of the graph there is a steady sequence without reconnections. This show that when the transmit process behaves properly the device spends around 6 seconds in receive mode between each transmit. As stated in section 5.1.1 on page 54, Telia uses approximately 60% less time in RRC connected mode resulting in lowered power consumption. At first I thought that this was the reason for the instabilities, but the behavior is the same with RAI set. Only considering transmit and receive time, the two networks perform more or less identical when RAI is set.

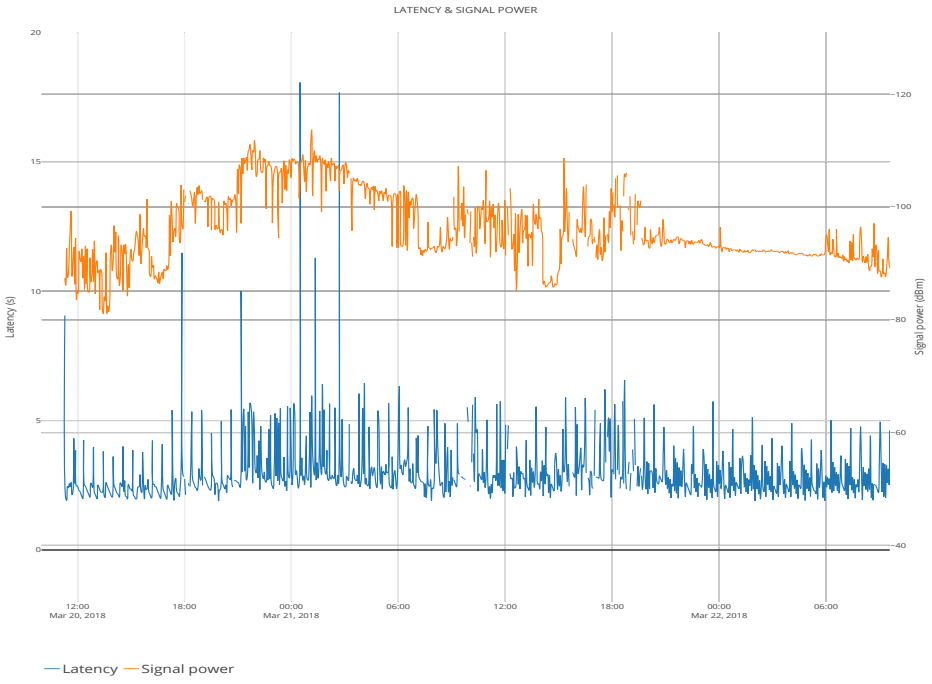


Figure 5.24: The figure displays a transmit session at UiO with Telia over 3 days, 20.03-23.03, with the signal power and latency graph. Visit webapp, for more details.

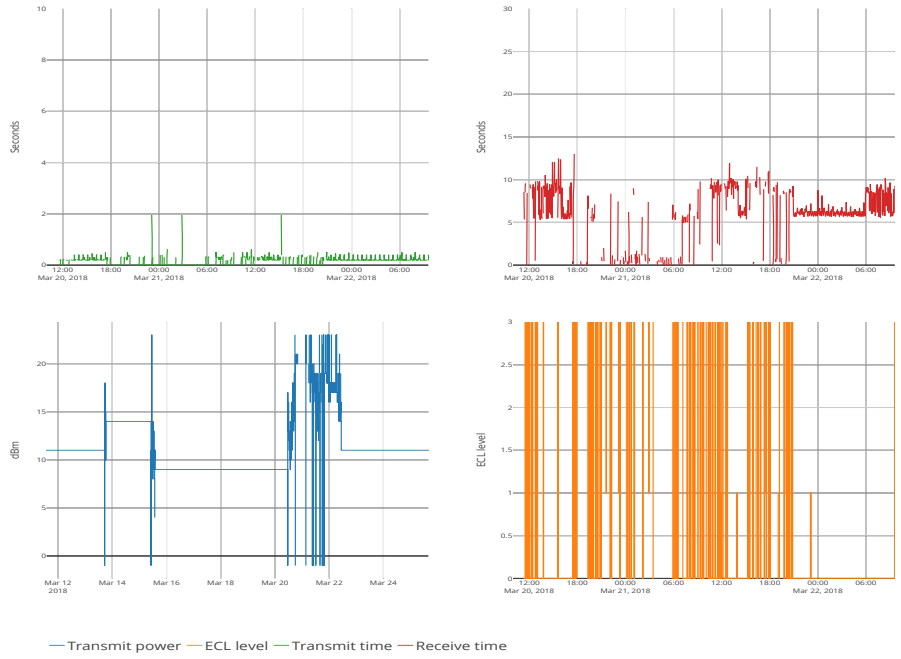


Figure 5.25: The figure displays a transmit session at UiO with Telia over 3 days, 20.03-23.03, with the statistics graphs. Visit webapp, for more details.

5.3.2 Cell selection

When the device has bad reception it will try to reselect which cell it is connected to. I wanted to test how this affects the battery lifetime since this might happen frequently if the device is located equally far from two cells with approximately the same signal power. It might then jump back and forth between these cells which could drain the battery. However, because of the limited NB IoT enabled eNB's I was not able to test this hypothesis. An attempt to connect to a network leads to higher power usage[5.1.7], hence reconnecting to a new eNB will probably use a similar amount of power.

5.3.3 Transmit Success Rate (TSR)

For an IoT application receive rate is not a priority, but it is important with stability and high uptime. All transmits towards the server includes a message id and this is used this to generate statistics about the TSR. The program **uptime.py** takes in three parameters, node id(-c), start date(-start) and end date(-end). If you specify the start and end dates the program will only include data within the given period. This option came handy due to issues related to the downtime on the server. The program calculates the TSR for sequences with a total number of transmits

higher than 5 for the given period and prints a summary of the results. I investigated all continuous sequences and used that to base the results upon. In listing 5.1, you can see the output of one of the results, which displays data for id3 between **2018-03-13T19:30:00** and **2018-03-15T09:00:00**. Here you see that there is one period with a total transmits of 438 and actual received transmits of 426, which results in a TSR of 97.26%.

Listing 5.1 Example of uptime.py

```
1 Statistics about collection: id3
2 Transmits: 438
3 Received transmits: 426
4 Uptime: 97.26%
```

I have looked at the sequences from UiO and can see that the TSR for Telenor and Telia is pretty similar to the listing example. There are some packets which are lost, which surprised me. Considering the number of devices connected to NB IoT at the point of the tests, the TSR should have been higher. A packet loss of ~3% is not very high, but if you imagine devices located in cities with device density of the technical specified limit of 50 thousand, the packet loss will probably increase resulting in degraded quality of service. One does not want the developers to have to implement retransmit logic in their applications so it will be interesting to see how the TSR changes as more devices connect to the network.

Chapter 6

Deviations

6.1 Imprecise clock

The AT command **AT+CCLK?** retrieves the time from the network. The timestamp is precise, but there were signs that after a while the time skewed and the latency on the server became negative. If you were to rely on the internal clock for your application, it would require time synchronization on regular intervals. Since this problem is also time dependent, meaning that after longer uptime the timestamp will be more skewed, the different UE's in your setup will have different offset to the correct time. The UE used was not able to sync the clock at specific intervals, hence I do not have any recommendations to how often one would perform the synchronization.

If your application requires very precise timestamps and you need to use the internal clock, a synchronization process is required. However, if the application tollerates some slack you can use the server timestamp instead of the timestamp retrieved from the UE. The server timestamp is affected by the latency in the network, but given that the UE has good coverage the latency has stayed under ten seconds, which may be good enough for some applications.

6.2 Imprecise NUESTATS

I believe that **NUESTATS** has given great feedback about the behavior of NB IoT. As stated, the command is not 100% accurate and there were certain issues related to ECL and receive/transmit time, but in the end, when I combined the results from the test applications it is clear what the device actually does. As stated in sections 2.5.1 on page 26 and 5.3 on page 77, the short-term tests gives the best picture of the state of the network at the time of the testing phase.

6.3 Network load

Since the NB IoT network was not in production at the time of the tests, the network was not heavily loaded. This means that the latency could be a bit higher with more general activity on the network. When the network is available the latency should be low and that is my impression on NB IoT as well.

6.4 Network density

Since the network was not in production, the density of the cell towers with NB IoT were limited. In the future all cells will implement NB IoT, meaning that the network will perform better. This means better coverage and lower latency given that the connected devices are spread over several eNB's.

6.5 Theoretical vs. practical power usage

The theory behind conversion between dBm, mA and mWh does not apply to the expected extent. I believe that calculating power usage based on the theory gives an indication of the power consumption, but it is only when using the appropriate tools that you will achieve the best understanding of power usage.

Chapter 7

Conclusions and future work

This chapter sums up the most important features of NB IoT and gives a status report on the current situation. I will give a short introduction to best practice guidelines, as well as combining the results.

7.1 Real world application guidelines

The testing phase has given a great overview of what one can expect of NB IoT. There are many possiblilites and with the results in mind I will give you a short introduction to what I believe is best practice guidelines for NB IoT in regard to a real world application. I will use Q-Free's parking sensors specifications to estimate power usage. The sensor is equipped with two 3600mAh batteries at 3.6 volts, which adds up to a maximum 25 920mWh.

Many developers will have a hard time deciding the transmit interval because it is difficult to predict the battery usage without doing any tests. Before calculating the power usage it is necessary to specify what packet size is recommended. Section 5.1.6 on page 64, showed that there was a steady power usage increasement when increasing the packet size, which is logical. By lowering the packet size you will be able to increase the transmit interval if it is necessary. I recommend that your application normally transmits small packets, around 50 bytes, and if necessary have an alternate packet, between 100-200 bytes, which the device can transmit from time to time. The alternate packet may contain a more detailed overview of the state of the device so that the server knows the state of each sensor. This is useful for maintenance and gives an indicator if something is wrong. When transmitting this larger packet you may also unset the RAI flag to allow for downlink communication. If your application only uses RAI, and eDRX is not enabled, it will never know if there is downlink data from the network. By utilizing the opertunity with the alternate packets your application will be able to receive downlink data which might be useful in some cases. A good example of a downlink

message is for configuration purposes. Maybe you want to add data properties to your applications, or change the transmit interval. If so, you can do this if your application supports it and you disable RAI regularly. Keep in mind that the application can't be changed after it is installed, so you are better of with many safety mechanisms in case of errors.

Another dilemma related to an application running for several years is network downtime. There will be periods where the device will loose connection to the network, either because of a power outage or because of unexpected behavior. Your application will need to handle these situations and I belive there are a couple of approaches to this problem. If the application detects network failure it should go into a configured reconnection period. This period can for example cover one day and do a number of things. Firstly the device will try to reconnect to the network before a set of transmits. It is wise to reconnect at an exponential rate, for example reconnecting at a rate equal to the power of 2. Imagine you loose network connection at 10:05, I recommend trying to reconnect to the network a set period prior to the following transmit. If the connection is not up at the time of the transmit the application should reconnect at the second following transmit, and then the fourth following transmit. In addition your application can reboot the device after a number of reconnection attempts. Rebooting the device will be a last resort option and should not be reattempted many times since it drains the battery. If the device does not manage to reinitiate connection towards the network after a day your application can restart the reconnection period, but you might want to reduce the number of reconnection attempts to prehibit more energy waste.

Only considering transmits of 100-200 bytes and with RAI set, the device consumes around $0.2mWh$ per transmits[14]. There has been transmits which used $< 0.05mWh$[51], but at the time of the testing this was not consistent and I will not base the following calculation on these results. It is good to see that the device actually can perform better, especially considering that the UE used was not specifically developed for NB IoT. I will continue using $0.2mWh$ as the base transmit consumption and based on the tests I have concluded that one transmit per hour is a good tradeoff. Given that there are similar circumstances for ten years, this results in a total power consumption of, $0.2mWh * 24 * 365 * 10 = 17\,520mWh$.

Objectivly, it is not likely that the tested setup would perform without flaws for ten years. I have create an excel spreadsheet(**longterm_calulations.xlxs**), which gives an overview of the general results, categorized by the performance of one transmit and accumulated up to twenty years. See figure 7.1 on the facing page, for an overview of the accumulated values for the different performance levels.

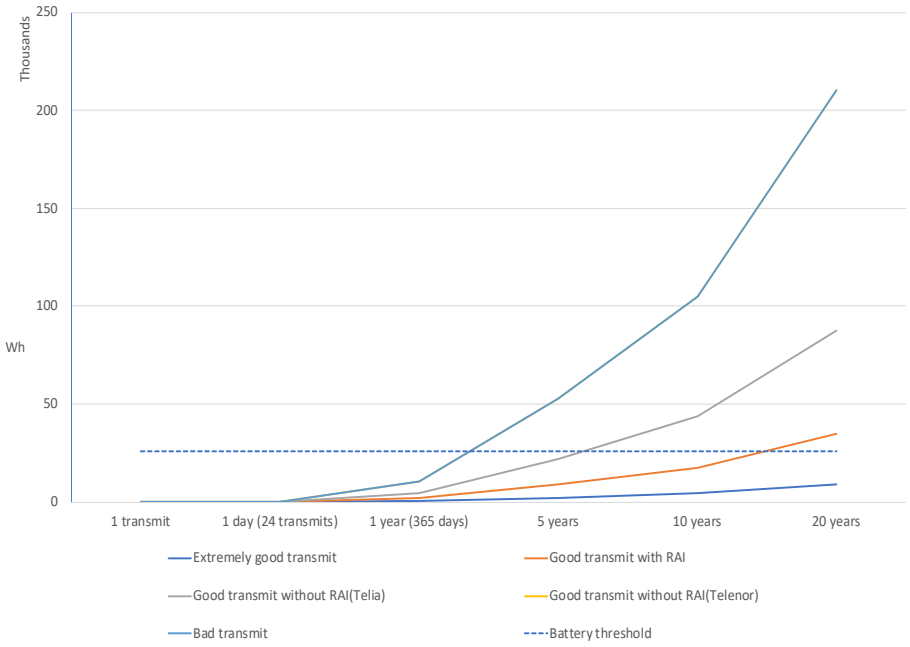


Figure 7.1: The figure gives an overview of the accumulated values for the different performance levels.

In addition I have added a simple form for splitting the amount of transmits over these categories. The spreadsheet is located in the 'general' folder of the thesis github page and lets you select different percentages of good and bad transmits. One out of twentyfour good transmits does not use RAI, and for simplicity the calculation uses the amount of bad transmits as basis for the reboot attribute. The rate of reboots is 10% of the percentage with bad transmits, which means one reboot per twentyfourth transmit, 10% of the time. In table 7.1, you can see what I believe is the closest to what the network would perform today. Reffering to Q-Free's sensor, it's battery size is 25 920mWh, which means that with the calculated performance the sensor would not opperate for ten years.

Transmits over 10 years	87600
% of excellent transmits	10 %
% of transmits without faults	80 %
% of transmits with RAI	67160 transmits
% of transmits without RAI	2920 transmits
% of bad transmits	10 %
% of reboots	1 %
Sum	

Telenor (mWh)	Telia (mWh)
438	438
13432	13432
1460	3504
10512	10512
20.75	182.5
25862.075	28068.5

Table 7.1: Long-term accumulated calculation

7.2 Combining the results

I have presented many aspects of NB IoT and tried to test them with Telenor and Telia, which produced results indicating the state of NB IoT. This section gives a highlight of the pros and cons of NB IoT in general and gives examples from the results. As stated in section 2.6 on page 27, the long-term results are affected of early stage hardware and software, hence the main focus will still be at the short-term tests. The section is split into subsections, each focusing on a specific NB IoT feature.

7.2.1 Transmit process

We have seen many transmit examples and in general they are all very similar to the specifications state. The transmit starts with a period of negotiation process, followed by the actual transmit and a optional RRC period. With RAI set, Telenor and Telia performed very similar and according to the specifications. However, with RAI unset, Telia increases battery lifetime by using what they say is DRX within the RRC period. This resolves in approximately 60% reduced power usage and can affect your application if the transmits do not use RAI. For most application this will not be a problem since it is recommended to use RAI for transmits unless really necessary.

In general the results related to the transmit process are promising. There were some power spikes up to ~250mA in many transmits when having good or excellent signal. The spike was not present at every transmit, which leads me to believe that this is a network related issue. Moreover, the spike is very short and only happens once per transmit and will probably not be a major problem even though the chip uses more power than necessary.

7.2.2 Coverage and latency

Coverage and latency is closely related and the results show that NB IoT meets the requirements of excellent coverage and latency below ten seconds. The specification states a coverage up to 35 kilometers, but this is without any obsticles, hence the coverage rate falls in city areas.

7.2.3 Connection time

Section 5.1.7 on page 66, displayed the results from the reboot tests. These tests showed what happened during the connection period and it was clear that there were big differences between Telenor and Telia. Telenor used a lot longer time to connect to the network, hence increasing the power usage. I did not manage to find the reason for the long connection

time, but it is worth noting the difference, since this might affect the battery lifetime in certain areas where reconnection or reboots will occour often. This is neither a big concern, since the network should be stable and there is potentially no need to reboot the device.

7.2.4 Uptime

Through the long-term tests the stability of the networks was tested and at that time the results were poor. The uptime was not as good as expected and the device needed to be rebooted to reinitiate normal behavior. Telenor's network was more stable than Telia, but not by much.

7.3 Future research

NB IoT is an interesting technology with great potenntial. For future research it would be interesting to explore certain aspects of the technology, for instance the overhead related to the transmit process. The article 'Overview of 3GPP Release 14 Enhanced NB-IoT', gives an overview of the new features and enhancements in 3GPP Release 14, which includes increased data rate, better support for mulit-carriers and lower power usage. In fact, 'in Release 14, a new UE power class with the maximum allowed output power reduced to 14 dBm was introduced to enable using smaller battery form factors for NB-IoT devices'[39]. Furthermore, it will be interesting to follow the development as the density increases.

Probably the most improvement at this time is UE related. It will be very interesting to follow the development of Nordic Semiconductor's NB IoT chip which should improve the flaws revealed by the tests in this thesis.

7.4 Final remarks

It has been a rewarding period working with NB IoT. I believe that this is a promising technology which will improve many applications. IoT is growing and there are companies waiting for good LPWAN's. As stated in section 2.5 on page 26, there is a higher demand for LTE-M1 in the U.S. which will postpone the production of NB IoT hardware to around 2019. The good thing is that Telenor and Telia will have their NB IoT implementations in production by 2018, meaning that the network is ahead of hardware production - which is unusual. There is one major drawback in the network today, which is the ability to investigate what happens when the device fails. The network is very much like a black box without any indication of the status, which can be frustrating for developers. The network providers must have

documentation related to NB IoT, thus the developers will have an easier time configuring their application to meet their requirements. Currently, there are too many unknown states and bugs in the network and the UE, and this needs to be handled before the network providers production set their implementations. Most companies will have to wait for the next generation hardware which increases the stability, and at that point people can start creating solid applications which are battery efficient and easy to maintain without the high level of logic currently needed.

Bibliography

[1] *1 x 40 Q-FREE, Telenor - device logging*. URL: http://158.39.77.97:9000/#/results/Q-FREE_TELENOR_5.02_2018-03-07_1_0x0_40_1_200.

[2] *1 x 40 Q-FREE, Telenor - no device logging*. URL: http://158.39.77.97:9000/#/results/Q-FREE_TELENOR_5.02_2018-03-07_0_0x0_40_1_200.

[3] *1 x 60 UiO, Telia - ECL 0*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_5.02_precision_2018-03-16_1_0x2_60_1_50.

[4] *1 x 60 UiO, Telia - ECL 2*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_5.02_precision_+30dBm_att_2018-03-16_1_0x2_60_1_50.

[5] *1MA266_0e_NB_IoT.pdf*. (Accessed on 03/26/2018). URL: https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_application/application_notes/1ma266/1MA266_0e_NB_IoT.pdf.

[6] *2 x 150 Q-FREE, Telenor*. URL: http://158.39.77.97:9000/#/results/Q-FREE_TELENOR_2018-02-28_1_0x0_150_2_100.

[7] *2 x 150, Q-Free Telia, weird behavior*. URL: http://158.39.77.97:9000/#/results/Q-FREE_TELIA_5.02_precision_2018-03-13_1_0x0_150_2_100.

[8] *2 x 150 UiO, Telia*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_5.02_precision_2018-03-13_1_0x0_150_2_100.

[9] *A Primer on 3GPP Narrowband Internet of Things - IEEE Journals & Magazine*. (Accessed on 02/19/2018).

[10] *A Survey of Traffic Issues in Machine-to-Machine Communications Over LTE - IEEE Journals & Magazine*. https://ieeexplore.ieee.org/document/7416135/. (Accessed on 02/19/2018).

[11] LoRa Allience. *A technical overview of LoRa® and LoRaWAN™*. (Accessed on 23/02/2017). URL: http://www.semtech.com/wireless-rf/iot/LoRaWAN101_final.pdf.

[12] Cisco. *IoT Growth*. (Accessed on 01/02/2017). URL: http://www.cisco.com/c/en/us/about/security-center/secure-iot-proposed-framework.htm.

[13] UN DESA. "World Population to 2300." In: *United Nations Department of Economics and Social Affairs, New York, NY [available at www.un. org/esa/population/publications/longrange2/WorldPop2300final. pdf]* (2004). (Accessed on 01/02/2017). URL: http://www.un.org/esa/population/publications/longrange2/WorldPop2300final.pdf.

[14] *Details - RAI set*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_long_term_2018-03-22_0x2_30_20.

[15] *Details - RAI unset*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_long_term_2018-03-22_0x0_30_20.

[16] *EFC - CEN/TC 278 Intelligent Transport Systems (ITS)*. (Accessed on 04/25/2018). URL: http://www.itsstandards.eu/efc.

[17] "Email correspondance with Telia." URL: https://github.com/henninghaakonsen/thesis/tree/master/references/Emails.

[18] Ericsson. *LTE: AN INTRODUCTION*. (Accessed on 14/01/2017). URL: https://www.ericsson.com/res/docs/2011/lte_an_introduction.pdf.

[19] *expressjs/express: Fast, unopinionated, minimalist web framework for node.* (Accessed on 04/11/2018). URL: https://github.com/expressjs/express.

[20] *Figures constituting the cost comparison figure*.

[21] *Finnsenderen*. (Accessed on 04/01/2018). URL: https://finnsenderen.no.

[22] *fluke8846a-bench-multimeter-500x500.png (500×335)*. (Accessed on 02/19/2018). URL: http://3.imimg.com/data3/AR/PU/MY-754870/fluke8846a-bench-multimeter-500x500.png.

[23] GSMA. *Mobile internet of things low Power wide Area Connectivity*. (Accessed on 10/02/2017). URL: http://www.gsma.com/connectedliving/wp-content/uploads/2016/03/Mobile-IoT-Low-Power-Wide-Area-Connectivity-GSMA-Industry-Paper.pdf.

[24] Thor Hansen. *History of GSM*. (Accessed on 19/01/2017). URL: https://snl.no/GSM.

[25] *IEEE Xplore Full-Text PDF:* (Accessed on 04/24/2018). URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8301831.

[26] *inga_gruen.png (2400×1349)*. (Accessed on 02/19/2018). URL: https://openclipart.org/image/2400px/svg_to_png/219943/inga_gruen.png.

[27] Oslo Kommune. *Statistikk over husholdninger*. (Accessed on 09/03/2017). URL: https://www.oslo.kommune.no/politikk-og-administrasjon/statistikk/befolkning/husholdninger/.

[28] *LearnBoost/cluster: Node.JS multi-core server manager with plugins support*. (Accessed on 04/11/2018). URL: https://github.com/LearnBoost/cluster.

[29] LinkLabs. *LTE eDRX and PSM Explained for LTE-M1*. (Accessed on 17/03/2017). URL: http://www.link-labs.com/blog/lte-e-drx-psm-explained-for-lte-m1.

[30] Lyse. *Automatiske strømmålere*. (Accessed on 10/02/2017). URL: http://www.lysekonsern.no/prosjekter/automatiske-strommalere-article565-321.html.

[31] *mcollina/node-coap: CoAP - Node.js style*. (Accessed on 04/11/2018). URL: https://github.com/mcollina/node-coap.

[32] *moment/moment: Parse, validate, manipulate, and display dates in javascript*. (Accessed on 04/11/2018). URL: https://github.com/moment/moment.

[33] *mongodb/node-mongodb-native: Mongo DB Native NodeJS Driver*. (Accessed on 04/11/2018). URL: https://github.com/mongodb/node-mongodb-native.

[34] "NACK mail correspondance between Ola and Ublox." URL: https://github.com/henninghaakonsen/thesis/tree/master/references/Emails.

[35] Nokia. *EPC and components*. (Accessed on 25/01/2017). URL: http://www.rcrwireless.com/20140509/diameter-signaling-controller-dsc/lte-mme-epc#prettyPhoto/1/.

[36] Nokia. *LTE evolution for IoT connectivity*. (Accessed on 24/01/2017). URL: http://resources.alcatel-lucent.com/asset/200178.

[37] OECD. *Smart Sensor Networks: Technologies and Applications for Green Growth*. (Accessed on 06/04/2017). Dec. 2009. URL: https://www.oecd.org/sti/ieconomy/44379113.pdf.

[38] "Ola Martin Lykkja."

[39] *Overview of 3GPP Release 14 Enhanced NB-IoT - IEEE Journals & Magazine*. (Accessed on 02/19/2018).

[40] *pc.png (350×280)*. (Accessed on 02/19/2018). URL: https://www.reactos.org/images/pc.png.

[41] *Power comparison*. URL: http://158.39.77.97:9000/#/results/Q-FREE_TELENOR_SHORT_TEST_2018-02-28_0_0x2_5_1_100.

[42] *Power example*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_5.02_precision_2018-03-16_1_0x2_60_1_512.

[43] Yevgeniy Sverdlik. *Custom Google Data Center Network Pushes 1 Petabit Per Second*. (Accessed on 19/01/2017). June 18, 2015. URL: http://www.datacenterknowledge.com/archives/2015/06/18/custom-google-data-center-network-pushes-1-petabit-per-second/.

[44] *Telenor reboot*. URL: http://158.39.77.97:9000/#/results/UiO_TELENOR_5.02_precision_reboot_2018-03-16_0_0x0_120_1_0.

[45] *Telia reboot*. URL: http://158.39.77.97:9000/#/results/UiO_TELIA_5.02_precision_reboot_2018-03-16_0_0x0_60_1_0.

[46] *The Evolved Packet Core*. (Accessed on 04/02/2017). URL: http://www.3gpp.org/technologies/keywords-acronyms/100-the-evolved-packet-core.

[47] *thesis/at_command_test.py at master · henninghaakonsen/thesis*. (Accessed on 04/11/2018). URL: https://github.com/henninghaakonsen/thesis/blob/master/code/at_command_test.py.

[48] *thesis/nbiot_labtest_details.py at master · henninghaakonsen/thesis*. (Accessed on 04/11/2018). URL: https://github.com/henninghaakonsen/thesis/blob/master/code/nbiot_labtest_details.py.

[49] *thesis/nbiot_labtest.py at master · henninghaakonsen/thesis*. (Accessed on 04/11/2018). URL: https://github.com/henninghaakonsen/thesis/blob/master/code/nbiot_labtest.py.

[50] *thesis/power_calculator.py at master · henninghaakonsen/thesis*. (Accessed on 04/11/2018). URL: https://github.com/henninghaakonsen/thesis/blob/master/code/power_calculator.py.

[51] *Transmit with very good result*. URL: http://158.39.77.97:9000/#/results/Q-FREE_TELENOR_5.02_precision_2018-03-07_0_0x2_60_1_200.

[52] "Ublox NB-IoT chip." In: (). URL: https://github.com/henninghaakonsen/thesis/tree/master/references.

[53] "Ublox NB-IoT chip - AT commands manual." In: (). URL: https://github.com/henninghaakonsen/thesis/tree/master/references.

[54] Wikipedia. *2016 Dyn cyberattack*. (Accessed on 09/03/2017). URL: https://en.wikipedia.org/wiki/2016_Dyn_cyberattack.

Acronyms

3GPP 3rd Generation Partnership Project. 1, 5, 11, 15, 93

AuC Authentication Center. 6

CAT-M Cat-M. 10, 11

CoAP Constrained Application Protocol. ii, v, 35, 36, 38, 45, 46

dBm Decibel / referenced to milliwatts. 12, 21, 24, 25, 33, 51, 52, 54, 56, 57, 60, 78–80, 88

DDoS Distributed Denial-of-Service. 17

DL Down Link. 4, 5, 18, 21, 57

DRX Discontinuous Reception. 20, 22–24, 54, 55, 92

DSRC Dedicated short range communication. 6

EARFCN Evolved Absolute Radio Frequency Channel Number. 34

ECL Coverage Enhancement Level. i, ii, vii, viii, 24–26, 34, 39, 40, 51–54, 58, 60–62, 76–78, 80, 87

eDRX Extended Discontinuous Reception. i, vii, 22–24, 35, 54, 55, 89

eMTC enhanced Machine Type Communication. 11

eNB Evolved Node B. 2, 5, 6, 11, 18, 21, 24, 31, 34, 60, 68, 78–80, 84, 88

EPC Evolved Packet Core. i, vii, 4–6

ETSI European Telecommunications Standards Institute. 15

GSM Global System for Mobile communications. 11, 16

GSMA GSM Association. 7, 10

HLR Home Location Register. 6

HSS Home Subscriber Server. 5, 6, 18

HTTP Hypertext Transfer Protocol. 46

IoT Internet of Things. i, vii, 1–4, 6, 8, 10, 12, 17, 18, 20, 64, 84, 93

IP Internet Protocol. 38

IPv4 Internet Protocol version 4. 15

ISM Industrial, Scientific and Medical. 10

ITS Intelligent Transportation Systems. 7

Kbit/s Kilobit per second. 12

LoRa . vii, 12

LPWAN Low Power Wide Area Network. vii, 1, 3, 4, 6–13, 15, 17, 18, 20, 24, 26, 27, 93

LTE Long-term Evolution. i, 2–5, 7, 8, 10–12, 15–20, 22, 27, 82

LTE-M1 . 11, 12, 23, 26, 27, 93

M2M Machine-To-Machine. 2, 4, 11, 18, 20, 27

mA Milliampere. 21, 22, 40, 41, 54, 56–58, 60, 61, 88, 92

mAh Milliampere hour. 89

Mbit/s Megabit per second. 10, 12

MME Mobile Management Entity. 5, 18, 24

MPS Maximum Packet Size. 64

MT Mobile Terminal. 34

mW Milliwatts. 21, 22, 25

mWh Milliwatt hour. 25, 40, 58, 60, 61, 63, 66, 88–91

mWs Milliwatt second. 40

NACK negative-acknowledgement. 52, 57, 60

NB IoT NarrowBand IoT. i, ii, v, vii, ix, xiii, 1–5, 8, 10–12, 15–27, 29, 30, 32, 33, 35–39, 41, 51, 53, 78, 79, 82, 84, 85, 87–90, 92–94

NPM Node Package Manager. 43, 44

OECD Organisation for Economic Co-operation and Development. 6, 7

PCI Physical Cell ID. 34

P-GW Packet Data Network Gateway. 5, 6

PRB Physical Resouce Block. 19, 20, 24

PSM Power Saving Mode. i, 17, 20, 22–24, 35, 36, 40, 51, 53–55, 66, 78

QoS Quality of Service. 6, 18

RAI Release Assistance Indicator. vii, 22, 35, 39, 41, 51, 53–55, 58, 60, 62–66, 79, 80, 83, 89–92

REPL Read-Eval-Print-Loop. 36, 38

RRC Radio Resource Connection. i, 20–24, 36, 40, 54–58, 62–64, 66, 67, 80, 83, 92

RSRQ Reference Signal Received Quality. 34

S-GW Serving Gateway. 5, 6

SNR Signal to noise ratio. 34, 54

T3324 Active Time. 22, 35, 36

T3412 Extended TAU Timer. 24, 35

TAU Tracking Area Update. 24, 35, 101

TDMA Time division multiple access. 23

TSR Transmit Success Rate. ii, 18, 24, 84, 85

UDP User Datagram Protocol. 35, 38

UE User Equipment. 2, 4, 6, 8, 11, 16, 18, 19, 21–25, 29, 34–36, 40, 53, 60, 61, 68, 78, 87, 90, 93, 94

UL Up Link. 4, 5, 18, 21, 57

WAN Wide Area Network. 17