Johannes Haukland

Modelling the Energy Consumption of NB-IoT Transmissions

Master's thesis in Communication Technology
Supervisor: Palma, David
June 2019

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Information Security and Communication Technology

Johannes Haukland

Modelling the Energy Consumption of NB-IoT Transmissions

Masteroppgave i Communication Technology
Veileder: Palma, David
Juni 2019

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for informasjonssikkerhet og kommunikajsonsteknologi

Abstract

Narrow-Band Internet of Things (NB-IoT) is an emerging Low Power Wide Area Network (LPWAN) communication protocol designed for IoT applications, which has been deployed in many countries including Norway [Tel]. The thesis gives a model for the energy consumption of NB-IoT with parameters that anyone designing a system would know. The useful properties of the work are many. By modelling the energy consumption it is possible to predict the energy budget of an NB-IoT application. An existing implementation can be optimized, and the energy savings quantified before execution of the application changes. The model can also be used to calculate NB-IoT energy cost in a larger simulation. As a standalone tool, the model gives an interesting insight into the technology.

The thesis compares NB-IoT to LoRaWAN and SigFox and explains the physical resources and power related technologies utilized in the protocol. Throughout the experiments conducted in the work, the energy capabilities of the protocol are studied – resulting in a model for the energy consumption. The model is able to predict the radio communication energy consumption of a NB-IoT device with a uBlox SARA-N211 modem.

Sammendrag

Narrow-Band Internet of Things (NB-IoT) er en voksende Low Power Wide Area Network (LPWAN) kommunikasjonsprotokoll designet for IoT applikasjoner, som har blitt implementert i mange land – inkludert Norge [Tel]. Avhandlingen gir en modell for energiforbruket til NB-IoT med parametere som enhver systemutvikler/designer vil kjenne til. Nyttige bruksområder for modellen er mange. Gjennom å modellere energiforbruket er det mulig å forutsi energibudsjettet til en NB-IoT applikasjon. En eksisterende implementasjon kan bli optimalisert, og den sparte energien kvantifisert før noen endringer blir utført. Modellen kan brukes som en NB-IoT energikost metode i en større simulasjon. Modellen i seg selv gir et interessant innblikk inn i teknologiens egenskaper.

Arbeidet sammenligner NB-IoT med LoRaWan og Sigfox og forklarer de fysiske ressursene og andre strømrelaterte teknologier og teknikker som benyttes i protokollen. Gjennom eksperimentene som utføres i avhandlingen blir energikapabilitetene til protokollen studert og en modell for energiforbruket er resultatet. Modellen kan forutsi energiforbruket til radiokommunikasjonen til en NB-IoT innretning med et uBlox SARA-N211-modem.

Preface

This Master's thesis dissertation is taken in the final semester (spring 2019) of my 5-year integrated MSc in Communication Technologies at the department of Information Security and Communication Technologies at Norwegian University of Science and Technology (NTNU). It awards 30 ECTS credits. The research was split into two parts, with a pre-project in Autumn 2018 and this thesis in Spring 2019. Johannes Haukland conducted the research with David Palma (Assoc. Prof.) as supervisor and Frank Alexander Kramer (Assoc. Prof.) as responsible professor.

The work focuses on the progression of a prediction model for Narrow Band Internet of Things devices through experiments on a network which supports the technology. The model can be used as a standalone tool when designing IoT applications or it can be combined with other tools in a larger simulation. I would like to thank my supervisor – David Palma – and my professor – Frank Alexander Kraemer – for the guidance they have provided throughout the project.

I would like to thank the team at Exploratory Engineering (EE) in Trondheim, especially Per K. Kummermo, Alf E. Helseth and Arne Munch-Ellingsen, for their expertise and unrivalled response time. They contributed to my understanding of how to optimally configure the EE-board for Telenors network.

I would also like to thank Ida M. V. Bosch for taking time out of her own thesis to explain the setup she uses for noise measurement, so that I was able to include a very relevant example of how my work can be used to analyse and improve an Internet of Things applications transmission energy efficiency.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

3GPP The 3rd Generation Partnership Project.

AT-commands Attention Commands or Hayes' Commands.

BER Bit Error Rate.

BPSK Binary Phase Shift Keying.

C-DRX Connected DRX.

CP Cyclic Prefix.

DL Downlink.

DMSR Demodulation Reference Signal.

DRX Discontinuous Reception.

DuCy1 Duty Cycle One.

DuCy2 Duty Cycle Two.

eDRX extended Discontinuous Reception.

eNodeB eUTRAN Node B.

FDD Frequency Division Duplex.

FFT Fast Fourier Transform.

GSM Global System for Mobile communications.

HARQ Hybrid Automatic Repeat reQuest.

H-SFN Hyper System Frame Number.

IFFT Inverse Fast Fourier Transform.

IoT Internet of Things.

ISI Intersymbol Interference.

ISM Industrial, Scientific, and Medical radio band.

ISP Internet Service Provider.

LoRa Long Range.

LPWAN Low Power Wide Area Network.

LTE Long Term Evolution.

LTE-M Long Term Evolution Mobile.

M2M Machine to Machine.

MAC Medium Access Control.

MIB Master Information Block.

MIMO Multiple Input Multiple Output.

MME Mobile Management Entity.

MTU Maximum Transmission Unit.

NB-1500 Arduino MKR NB 1500.

NB-IoT Narrow-Band Internet of Things.

NCID Narrowband physical Cell ID.

NPBCH Narrow-band Physical Broadcast Channel.

NPDCCH Narrow-band Physical Downlink Control Channel.

NPDSCH Narrow-band Physical Downlink Shared Channel.

NPRACH Narrowband Physical Random Access Channel.

NPSS Narrow-band Primary Synchronization Signal.

NPUSCH Narrowband Physical Uplink Shared Channel.

NRS Narrow-band Reference Signal.

NSSS Narrow-band Secondary Synchronization Signal.

NTNU Norwegian University of Science and Technology.

OFDM Orthogonal Frequency-Division Multiplexing.

OFDMA Orthogonal Frequency-Division Multiple Access.

PDCP Packet Data Convergence Protocol.

PRB Physical Resource Block.

PSD Power Spectral Density.

PSK Phase Shift Keying.

PSM Power Saving Mode.

QoS Quality of Service.

QPSK Quadrature Phase Shift Keying.

RB Resource Block.

RND 320 RND lab DC power supply RND 320-KD3005D.

RRC Radio Resource Control.

RU Resource Unit.

SC-FDMA Single Carrier Frequency Division Multiple Access.

S-GW Serving Gateway.

SIB System Information Block.

TAC Tracking Area Code.

TAI Tracking Area Identity.

TAU Tracking Area Update.

TTL Transistor-Transistor Logic.

UART Universally Asynchronous Receiver/Transmitter.

UE User Equipment.

UL Uplink.

UMTS Universal Mobile Telecommunications System.

Chapter 1 Introduction

The number of sensors and simple devices which require an internet connection is rapidly increasing, with predictions of more than 20 billion by 2020 [GBMP13]. These devices are highly self-sufficient – requiring little to no human interaction – and the network they make up is referred to as the Internet of Things (IoT). The communication between *things* is often referred to as Machine to Machine (M2M) communication. Tasks that would be tedious for humans and unreasonably uneconomical to perform with traditional computers can be executed with these devices. The devices should be possible to deploy *anywhere*, without connection to the power grid or internet via Ethernet or Wifi (e.g. on-field monitoring systems). To make this possible they need to be powered by battery and connected to a wide area network.

One of the main challenges with the emergence of Internet of Things (IoT) is having an energy-efficient solution, such that devices in remote locations without access to a power supply are able to function for years without requiring a battery change. This point is reinforced as the number of existing predicted devices is in the order of magnitude of tens of billions, making manual maintenance infeasible. A battery life of years in deployment is possible because of the nature of sensing applications – being without need of large data transmission and suitable for prediction models [LL17] which may decrease the energy usage even further. NB-IoT is a communication technology with the purpose of allowing devices suitable for IoT and LPWANs to be deployed effectively worldwide.

1.1 Motivation

LPWANs are emerging to meet new demands in the field of IoT. Sigfox, LoRa, and NB-IoT are the three LPWAN technologies that compete for large-scale IoT deployment. LPWAN technologies are similar in the sense that they all rely on low energy consumption and wide coverage – as the term implies. The details on how they accomplish this feat and the goals with regards to energy efficiency and range differ enough to allow the protocols to coexist with specialization for different use cases. NB-IoT has been deployed in many countries including Norway [Tel]. The thesis gives a model for the energy consumption of NB-IoT with parameters that are known to a developer. The parameters are (1) measurement interval – how often the application obtains data, (2) measurement size – the size of the data obtained in bytes and (3) analysis interval – how often the data needs to be gathered for analysis. The model can be used for the following:

- **Predicting the energy cost of an NB-IoT application**

  By predicting the energy consumption, the system designer can choose a fitting battery capacity based on the application requirements. The model will tell the developer how much energy the modem uses, so that the cheapest battery that stores enough energy to support it can be used.

- **Optimizing the energy budget of an existing IoT application**

  Comparing the energy consumption of an existing IoT application with the prediction of the model will tell you whether the system could be improved. This could mean changing communication protocol to NB-IoT if other is applied or configuring the NB-IoT system better. By using the mentioned parameters from an existing application – the user can compare the models predicting with the current consumption and choose to change the applications implementation based on the results.

- **Calculating NB-IoT energy cost in a larger simulation**

  The model will be used in a larger project detailed in [MKBT19] as the NB-IoT part of the sensor simulation which the paper refers to as the Sensor Gym. The project uses reinforcement learning to manage wireless, energy-harvesting IoT nodes. The model used in the project uses energy measurements from different communication technologies – including NB-IoT. The model presented in this thesis will be used for this task.

- **Giving a deeper insight into the technology**

  The thesis will allow any interested party, whether it be an engineer, developer or network operator to learn more about the technology and its energy properties.

1.2 Research Goals

There are two goals for this thesis: understanding the different power modes for NB-IoT duty cycling and implementing a model that can predict the energy consumption of a NB-IoT modem. We want to be able to configure the power modes on a modem to measure the energy consumption of the different modes.

RQ1: What are the power modes for Narrow-Band IoT devices?

RQ2: How can the power modes for NB-IoT be configured?

RQ3: How can the energy consumption of NB-IoT be modelled?

The proposed research looks at NB-IoT as it is implemented in Norway in Spring 2019 with the different modems available and the boards implementing these modems. The model will be suitable to predict energy consumption on an end device connected to a network configured to support the power modes.

1.3 Thesis Structure

The structure of the thesis is: Chapter 2 presents the Background and Related Work. The work introduces LPWAN and compare the three most prominent solutions, Long Range (LoRa), Sigfox and NB-IoT. We then present NB-IoT as an IoT evolution from Long Term Evolution (LTE). An overview of the standard is given, with Operation modes, Radio Resource Control and power modes. The model for energy consumption provided by The 3rd Generation Partnership Project (3GPP) is examined and its faults pointed out.

In Chapter 3, we present the research methodology used in this thesis. The chapter also includes a presentation of the equipment used in the experiments as well as an introduction to the experiment design. In Chapter 4, the experiments on an Arduino NB-IoT device is presented with their preliminary results. Chapter 5 and 6 study the energy consumption of a break-out board with the modem for more general results. In Chapter 7, examples illustrate the usefulness of the model, and finally – Chapter 8 discusses the approach to the thesis and concludes the work.

Chapter 2

Background and Related Work

This chapter will the introduce core concepts and technologies which build the foundation of NB-IoT as well as some key terminology. Different LPWANs will be compared to investigate their successful coexistence and related studies will be highlighted if applicable. The technology is newly deployed, meaning that most protocol studies are conducted by 3GPP itself. The last section in the chapter will explain why the energy calculation technique provided by 3GPP is not developer friendly.

2.1 Low Power Wide Area Networks

LPWANs are a set of novel communication paradigms made to address the diverse requirements of IoT applications, complementing the traditional short range wireless and cellular technologies. The term LPWA encompasses technologies with characteristics such as large coverage areas, low bandwidth, possibly very small packet and application-layer data sizes, and long battery life operation [RKS17]. The technologies aim to suit communications between *things*, such as sensors operated on a battery. All constrained networks must balance power consumption, battery life, cost, and bandwidth. LPWANs prioritize power and cost benefits by accepting severe bandwidth and duty cycle[1] constraints. LPWANs are attracting attention for their characteristics, more specifically because they fill an untapped marked in the IoT world [MBCM17]. Short range wireless networks such as Bluetooth and Zig-Bee make different trade-offs, one of them being a lower coverage limited to a couple hundred meters in an ideal scenario or using multi-hop communications [RKS17]. Wide coverage is provided by the cellular networks Global System for Mobile communications (GSM) and LTE. They do however not achieve an energy efficiency suitable for devices in remote locations without power supply requiring over ten years of battery lifetime. This is because of their features including handover between base stations and regular signaling, but mostly because of the high data

[1]The duty cycle is the manner in which the device transitions from one power mode to another.

rates. The leading LPWAN technologies are Sigfox, LoRa and NB-IoT, which the following sections will introduce and compare.

2.2 Comparing Low Power Wide Area Networks

With the rise of Internet of Things, there is a need for devices that do not send much data, need to be cheap, require long range coverage and very small power budgets. Use cases often include a device with a sensor and antenna, measuring a single value and transmitting when needed. Sigfox is a company that realized the potential and made a solution fitting the requirements for extremely low data and long range use cases, with payloads of 12 bytes, a maximum of 140 transmissions per day and promises a coverage range of 10km in urban and 40km in rural areas[MBCM17]. The LoRa-Alliance introduced LoRaWAN (long range wide area network) with a higher payload length of 243 bytes and unlimited messages per day for some implementations [MBCM17]. However, both Sigfox and LoRa are utilizing unlicensed Industrial, Scientific, and Medical radio bands (ISMs) which may be subject to interference from third-party communication systems, among other limitations. NB-IoT is a protocol designed to use the licensed bands of the LTE network and can be supported with only a software upgrade in addition to the existing LTE infrastructure. The LPWAN is specified by 3GPP in release 13 [3GP18a]. It has a much higher payload length than the previously mentioned technologies with a maximum of 1600 bytes. Because Sigfox and LoRa occupy unlicensed bands, they cannot offer the same level of Quality of Service (QoS) as NB-IoT, which means that any IoT application that requires a guaranteed QoS should use NB-IoT. Using the licensed band comes at a great cost, however we will look at how NB-IoT can circumvent using precious frequencies in Section 2.3.3.

Sigfox, LoRa and NB-IoT lay the foundation for User Equipments (UEs) which can be in sleep mode for most of the lifetime, while not transmitting or receiving. This greatly reduces the average energy consumption of a device for an application that requires transmissions seldom enough to be dormant long enough to enter sleep mode. The different access modes employed affect the energy consumption. The OFDM/OFDMA employed by NB-IoT requires more peak current than the access modes of Sigfox and LoRa [MBCM17].

When it comes to latency, NB-IoT and LoRa class C handle low bidirectional latency, making them more energy consuming than Sigfox and other LoRa classes. Therefore, for application that require low latency, NB-IoT and LoRa class C are the better choices [MBCM17]. Sigfox and LoRa support up to 50.000 end devices per cell, while NB-IoT supports up to 100.000, making it more scalable. Other relevant IoT factors to consider are network coverage, range and cost. Sigfox has the highest base station coverage with range of more than 40km, LoRa up to 20km and NB-IoT

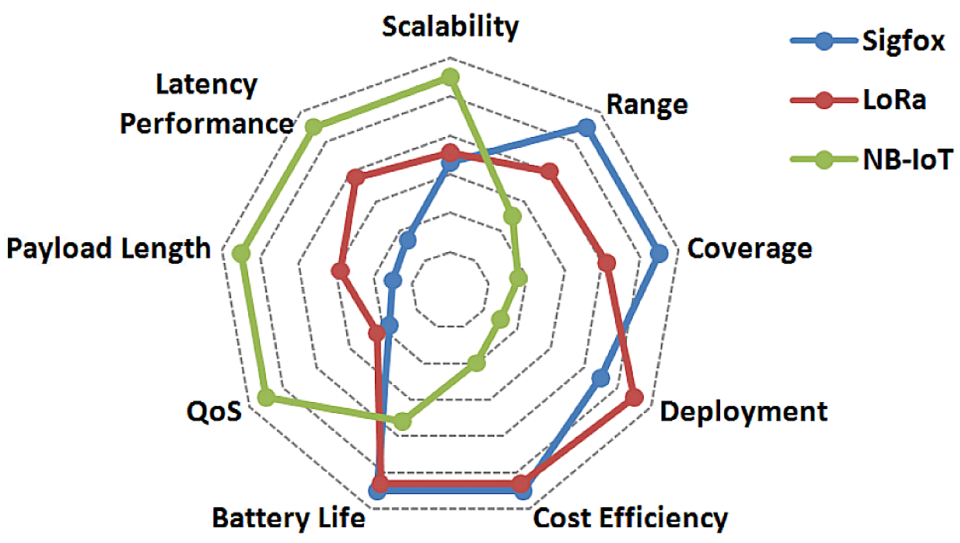


Figure 2.1: LPWANs Sigfox, LoRa and NB-IoT with their respective advantages for IoT factors. Adapted from [MBCM17]. Range is the maximum distance between a device and a base station, while coverage is the ability to reach devices within the range.

up to 10km. Because NB-IoT is an evolution of LTE, the deployment is limited to LTE base station range. In summary, we have seen that Sigfox, LoRa and NB-IoT fit different application needs. Their respective advantages in terms of the presented IoT factors can be seen in Figure 2.1. The figure shows scalability, latency performance, payload length, QoS and battery life as the main features of NB-IoT. It is also clear that Sigfox and LoRa cover different IoT factors – allowing a co-existence [SWH17].

2.3 Narrow-Band Internet of Things

Section 2.2 introduced the most prominent emerging and established LPWANs and their different approaches to solving the requirement of wide area coverage and low power consumption. It was discovered that the factors where NB-IoT outperforms the other LPWAN choices are scalability, latency performance, payload length and QoS. Figure 2.1 shows different use cases where NB-IoT is considered the better choice. The figure shows the study cases where NB-IoT was the best suitable LPWAN [SWH17]. This section will discuss the relationship between LTE and NB-IoT, key technologies in NB-IoT and the network architecture.

Better choice	Study cases	Major IoT categories	Parameters
NB-IoT	Wearables Smart Bicycle Kids monitoring Pet tracking PoS terminals Smart metering	IoT Personal	Range, diversity, latency, Qos
	Smart parking Alarms and event Detectors Smart garbage bins	IoT Public	Range diversity, latency, QoS

Table 2.1: The most relevant use cases for NB-IoT, adapted from [SWH17] where NB-IoT, LoRa and Sigfox are compared for IoT use cases.

2.3.1 LTE (4G) architecture and enhancements for IoT

NB-IoT reuses LTE functionality with simplifications and optimizations. When LTE was introduced it was designed with the main requirements for the new access network consisting of high spectral efficiency, high peak data rates and short round trip time as well as flexibility in frequency and bandwidth [fG08]. High peak data throughput would here mean throughput that is a few orders of magnitude higher than that of Universal Mobile Telecommunications System (UMTS) for 3G.

The requirements for LTE suit mobile device communication perfectly as tailored, while not fitting the requirements of IoT. LTE does however include several Radio Resource Control (RRC) functions that are responsible for the UE management and control. This includes processing of broadcast system information, to help decide which network to connect to. It includes paging to indicate an incoming call to a device in Idle Mode, which is a power mode that will be discussed below. RRC also has several other functions such as RRC connection management between UE and eNodeB, UE measurement reporting and control of signal quality, QoS management and Integrity protection. NB-IoT inherits these properties through the RRC.

Wanting to utilize the already established technology of LTE, 3GPP has specified several enhancements for LTE. The goals for IoT device connectivity were as follows [Sau17]:

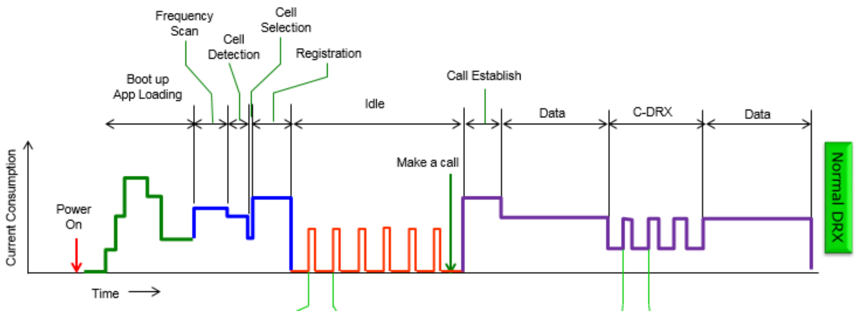


Figure 2.2: DRX for LTE, with two main parts. The red part indicates idle DRX and in the purple part Connected-DRX (C-DRX) in shown. The time between paging is the same. Figure from [Sha18a].

– Low-cost radios for devices ($5 or less)

– Thousands of devices per cell, transmitting a few bytes per day

– Ultra-low power consumption (battery life of up to 10 years when transmitting a few bytes per day)

– Efficient support for devices with low data rates (in the order of tens and hundreds of kilobits per second maximum throughput)

IoT use cases are many and have different requirements which makes it so that no technology fits all. The most important differing factors are transmitting frequency, bitrate and coverage. Pure LTE channels of 10MHz or 20MHz do not provide deep indoor-coverage. With this in mind, 3GPP has standardized four enhancements to LTE, LTE Category 1 (LTE Cat-1), LTE Category 0 (LTE Cat-0), LTE Category M1 (LTE-M) and LTE Category NB1 (NB-IoT). The three first aforementioned enhancements are mainly adding new functionalities to the existing LTE interface. LTE Cat-1 is rarely used as few devices implement it.

2.3.2 Discontinuous Reception and Power Save Mode

Discontinuous Reception (DRX) was introduced with LTE as a way for UEs to save energy when not transmitting data. The device negotiates with the network and agrees on a time interval at which the device will listen for incoming paging. The rest of the time, the device is unreachable by not actively listening for incoming data and thus saving energy. DRX is split into two parts, idle DRX and Connected DRX (C-DRX) as seen in Figure 2.2.

Table 2.2: Comparing key features of LTE enhancements.

	LTE Cat-1	LTE-M	NB-IoT
LTE release	Release 8	Release 13	Release 13
Spectrum	Licenced LTE in-band	Licenced LTE in-band	Licenced LTE in-band, guard-band, stand alone
Bandwidth	1.4 - 20 MHz	1.4 MHz	200 kHz
Max Coupling Loss	144 dB	156 dB	164 dB
Max Data Rate	$<$ 10 Mbps(DL) & $<$ 5 Mbps (UL)	$<$ 1 Mbps	$<$ 170 kbps (DL) & $<$ 250 kbps (UL)
Duplex Mode	Full	Half/Full	Half

Power Save Mode (PSM) is introduced by 3GPP in [3GP15d] as a new device status to minimize energy consumption. The energy consumption is expected to be lower with Power Saving Mode (PSM) than with the existing LTE implementation that is Idle Mode. It is similarly defined as "powered off" mode, but the device stays registered with the network, and thus avoids having to re-establish its network connection.

2.3.3 Narrow-Band Internet of Things Overview

Now that we have discussed the enhancements of LTE towards IoT applications and introduced NB-IoT and some of its key features, let us take a closer look at what makes the enhancement promising for use cases where low energy consumption and extended coverage is required. This includes baseband characteristics, data transmission scheme, PSM and extended Discontinuous Reception (eDRX).

NB-IoT solved the congestion problem that exists in other LPWANs by inheriting the licensed bands of LTE, which also makes it possible to use the existing network hardware and reduce the deployment cost – given that NB-IoT can coexist with GSM and LTE [BK15]. Compared to other 3GPP technologies, the maximum NB-IoT data rate is lower, see table 2.2. The cell coverage however is enhanced, and the hardware complexity reduced compared to LTE, which itself makes it possible for NB-IoT to reduce cost and energy consumption. The most noticeable difference, as the name implies, is the width of the band which is decreased by an order of magnitude compared to Long Term Evolution Mobile (LTE-M).

At the physical layer, NB-IoT occupies 180kHz of spectrum, which is much smaller than the previously mentioned LTE spectrum of 1.4-20MHz. For downlink, NB-IoT has a subcarrier spacing of 15kHz and thus the 12 subcarriers make up the

180kHz of the channel. Slot, subframe, and frame durations are 0.5 ms, 1 ms, and 10 ms, respectively, identical to those in LTE. For uplink, the UE might be assigned either 1, 3, 6 or 12 tones[2]. Multi-tone transmission uses the same 15 kHz subcarrier spacing, 0.5 ms slot, and 1 ms subframe as LTE. Single-tone transmission supports two numerologies, 15 kHz and 3.75 kHz with 12 and 48 subcarriers respectively. For both uplink and downlink 10kHz guard-band is occupied on either side of the used spectrum, resulting in a 200kHz total occupied spectrum.

The antenna is half duplex instead of full duplex, which means that it is not able to transmit and receive in the same time slot. This is a trade-off where transmission speed is decreased to decrease complexity and thus cost of the UE. NB-IoT also deploys a single antenna system instead of Multiple Input Multiple Output (MIMO). Where MIMO has many positive qualities ultimately resulting in an increased data rate, this is another compromise to reduce cost via reduced complexity. The half duplex operation mode is using Frequency Division Duplex (FDD), which is referred to as using HD-FDD mode. HD-FDD for NB-IoT supports 60 kbit/s peak rate in uplink and 30 kbit/s peak rate in downlink, and a Maximum Transmission Unit (MTU) size of 1600 bytes, limited by Packet Data Convergence Protocol (PDCP) layer [Far18]. Any packet size up to the set MTU can be passed to the stack from higher layers. The RLC is responsible for segmentation of the packet and can segment into parts as small as 16 bits. Convolutional coding is used instead of Turbo coding for the downlink, which means that only one decoder is needed, sparing complexity further. The coding also only allows for a single Hybrid Automatic Repeat reQuest (HARQ)[3] process. To simplify modulation, only Binary Phase Shift Keying (BPSK) and Quadrature Phase Shift Keying (QPSK) are allowed[4].

NB-IoT allows for transmission repetitions, Power Spectral Density (PSD) boosting and using single-tone transmission with Phase Shift Keying (PSK) modulations to reduce peak-to-average power ratio in the uplink, which improves the indoor coverage by 20 dB [RMZ+16].

2.3.4 NB-IoT Operation Modes

Three different operation modes are defined for NB-IoT, In-band Mode, Guard-band Mode and Standalone Mode. See figure 2.3 for an illustration. In-band Mode is when NB-IoT is deployed in a legacy LTE band. This is done by allocating one or more Resource Blocks (RBs) from the legacy LTE to NB-IoT. The total eUTRAN Node B (eNodeB) power is shared between LTE and NB-IoT with the possibility

[2] *Tones* and *subcarriers* are used interchangeably in the literature meaning OFDMA/OFDM carrier partitions. OFDM will be introduced in Section 2.4.1

[3] HARQ is a combination of forward error correction and repetitions to ensure correct decoding.

[4] Phase Shift Keying is a modulation process which conveys data by changing (modulating) the phase of a constant frequency reference signal.

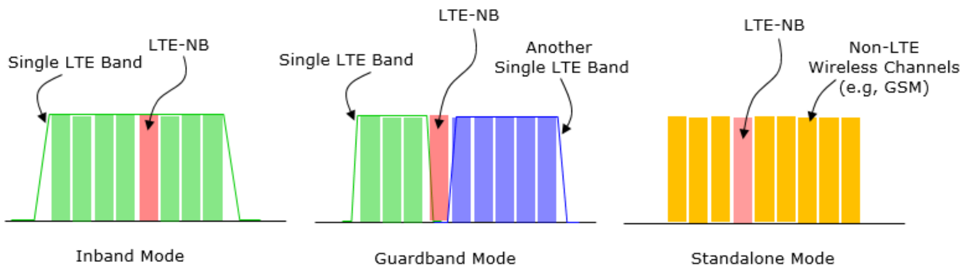


Figure 2.3: The different deployment modes for NB-IoT (here referred to as LTE-NB, figure from [Sha18b].

to use power spectral density (PSD) boosting for NB-IoT. Sharing of Physical Resource Blocks (PRBs) between NB-IoT and LTE allows for more efficient use of the spectrum [RMZ+16]. Guard-band mode is when NB-IoT is deployed within the guard band between two legacy LTE carriers. The advantage being that deploying in this mode does not occupy bandwidth that could be utilized for LTE. Studies done by 3GPP found that NB-IoT can coexist with LTE in-band and guard band with certain observations [3GP18c]. NB-IoT will perform some interference on the first adjacent LTE PRB, while the interference on other PRBs is insignificant or acceptable. The simulations found that guard band operations have slightly better coexistence compared to in-band operation. In previous two modes, LTE-NB coexists with an existing legacy LTE band. But we can think of another case where it is deployed completely independent from any legacy LTE, Standalone Mode. The study also found that NB-IoT can coexist with LTE, GSM and UMTS in Standalone Mode [3GP18b].

2.3.5 Radio Resource Control in NB-IoT

In section 2.3.1 we introduced RRC for LTE. Here we will explore the properties that are inherited by NB-IoT. 3GPP specifies three RRC states, RRC-Idle, RRC-Inactive and RRC-Connected [3GP18d] section 4.2.1. In connected state unicast data is transferred to and from the UE. The UE monitors control channels associated with the shared data channel to determine if data is scheduled for it. If wanted, the UE might be configured with a UE specific DRX at lower layers. When the device is in connected state it consumes more energy than in the other states. In idle state, the UE enters an eDRX cycle. Figure 2.2 shows the legacy LTE DRX. The eDRX extends the time periods between reception availability for both idle DRX and C-DRX. While C-DRX is increased slightly, the idle DRX is increased by several orders of magnitude for a minimum of seconds to a maximum of hours. The UE performs neighbouring cell measurements and cell (re-)selection, acquires system information

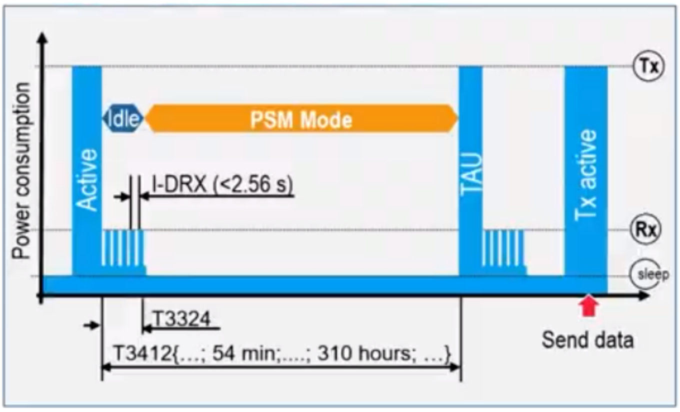
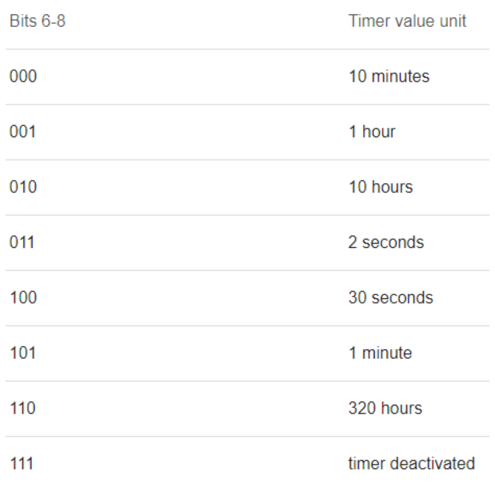


Figure 2.4: Power Save mode. T3324 determines how long the device will stay in idle mode before entering PSM. Figure from [RS17].

and performs logging of available measurements together with location and time for logged measurement configured UEs [3GP18d] section 4.2.1. For NB-IoT, the boundaries of the eDRX acquisition period are determined by Hyper System Frame Number (H-SFN) values for which H-SFN $mod 1024 = 0$ [3GP18e]. The maximum number of hyper frames that a device can request adds up to almost three hours of extra sleep for NB-IoT between paging occasions. This is, in addition to PSM, the most important technological factor with regards to saving energy for the UE.

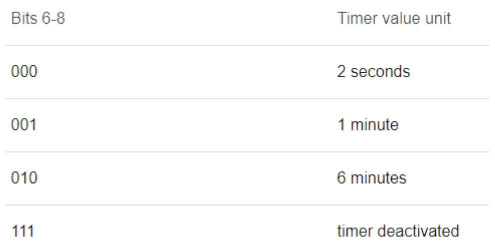
2.3.6 Power Save Mode (PSM) for NB-IoT

PSM is a UE mechanism available for all LTE device categories including NB-IoT to reduce the energy used by the UE. The mode was introduced in 3GPP release 12 [AAP+17]. The UE reports how often and for how long it needs to be active in order to transmit and receive data. However, the final values are determined by the network [GSM17a]. When a device initiates PSM, it provides two timers (T3324 and T3412), and the PSM time is the difference between the timers. See figure 2.4. The device then enters a state similar to power-off with the exception of remaining registered with the network. The network retains the state information. The device is then able to send data before the expiration of the time interval agreed with the network without reattaching, while being unable to be contacted by the network. While the device is asleep, an operator might store incoming packets and forward when the device awakens. PSM mode can be cancelled anytime by the device by sending a Tracking Area Update (TAU) to the network that does not include the PSM timers. The TAU contains the LTE Tracking Area which is analogous to the Location Area in UMTS. It is a geographical combination of several eNodeBs. Each tracking area has a Tracking Area Code (TAC) and Tracking Area Identity (TAI),



Bits 6-8	Timer value unit
000	10 minutes
001	1 hour
010	10 hours
011	2 seconds
100	30 seconds
101	1 minute
110	320 hours
111	timer deactivated

Figure 2.5: T3412 encoding. The 3 most significant bits define the timer unit, while the 5 least significant bits define the timer multiplier value.



Bits 6-8	Timer value unit
000	2 seconds
001	1 minute
010	6 minutes
111	timer deactivated

Figure 2.6: T3324 encoding. The 3 most significant bits define the timer unit, while the 5 least significant bits define the timer multiplier value.

where the TAC identifies the tracking area within the network, and the TAI is a globally unique identifier. The TAU is sent when the T3412 timer expires, or before, to re-establish the connection to NB-IoT and enter connected state (RRC-Connected). The timers are encoded as binary numbers. Figure 2.5 shows the encoding scheme for the three most significant bits of the 8-bit encoding. The following five bits are used as a multiplier [uBl18]. An example is “0010011” where Figure 2.5 tells us the unit is one hour, and the multiplier value is “0011” = 6. The result is then $6 \times 1 = 6$ hours.

The encoding of the T3324 timer is shown in Figure 2.6. The three most significant bits define the timer unit, while the five least significant bits define the timer multiplier value [uBl18].

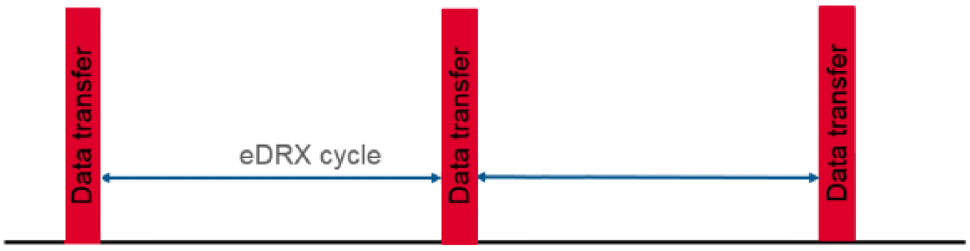
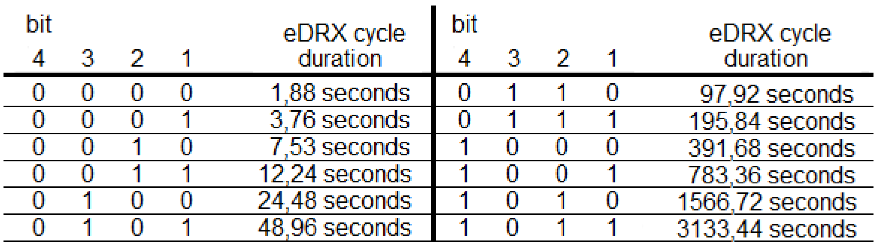


Figure 2.7: The eDRX cycle illustrated, adapted from [GSM17b].



bit 4	3	2	1	eDRX cycle duration	bit 4	3	2	1	eDRX cycle duration
0	0	0	0	1,88 seconds	0	1	1	0	97,92 seconds
0	0	0	1	3,76 seconds	0	1	1	1	195,84 seconds
0	0	1	0	7,53 seconds	1	0	0	0	391,68 seconds
0	0	1	1	12,24 seconds	1	0	0	1	783,36 seconds
0	1	0	0	24,48 seconds	1	0	1	0	1566,72 seconds
0	1	0	1	48,96 seconds	1	0	1	1	3133,44 seconds

Figure 2.8: The eDRX encoding scheme illustrated. Data from [3GP16a].

2.3.7 Extended Discontinuous Reception (eDRX) for NB-IoT

eDRX is an extension to DRX, where the time interval during which a device is not listening to the network is allowed to be greatly lengthened. It can be used without PSM or in conjuction with PSM to obtain additional power savings [3GP16a]. Figure 2.7 shows what the eDRX cycle looks like on its own. When used in addition to PSM, the recommended procedure is to transmit and then enter eDRX for a time period to receive downlink information from an eNodeB that will be transmitted at an unknown time. This way the device is consuming as little energy as possible while being periodically available for a downlink reply. The device should then enter PSM as the energy consumption is lower in this mode than eDRX [3GP16a].

The encoding scheme for the eDRX is a simple 4-bit value. The bits are translated to a certain number of seconds as shown in Figure 2.8. Included in the header of the encoded transmission is a two-bit value which allows eDRX to be configured as disabled when given the value "11". When only transmitting data upstream (from the device), eDRX should be disabled to increase battery life [engb].

2.4 NB-IoT Physical Layer

Here the physical layer refers to the lowest layer on the OSI reference model from [DZ83]. The layer is concerned with the physical connectivity of two different stations and defines among other things the pulses used to represent binary signals. By looking at the physical layer we get an overview over the resources for transmissions with NB-IoT.

First, we need to talk about LTE channels. There are three categories into which the data channels may be grouped - Physical channels, Logical channels and Transport channels [TJ10]. Physical channels are transmission channels that carry user data and control messages. Logical channels service the Medium Access Control (MAC) layer within the LTE protocol structure. Transport channels offer information transfer to MAC and higher layers. The LTE transport channels vary between the Uplink (UL) and the Downlink (DL) as each has different requirements and operates in a different manner. Because of this, the focus will rather be on the physical channels.

2.4.1 Downlink

The three physical channels

– Narrow-band Physical Broadcast Channel (NPBCH)
– Narrow-band Physical Downlink Control Channel (NPDCCH)
– Narrow-band Physical Downlink Shared Channel (NPDSCH)

exist for DL, with two physical signals

– Narrow-band Reference Signal (NRS)
– Narrow-band Primary Synchronization Signal (NPSS) and Narrow-band Secondary Synchronization Signal (NSSS)

The NPBCH transmits the Master Information Block (MIB), which is the first among the System Information Blocks (SIBs) broadcasted by the eNodeB. The MIB includes information such as the system bandwidth, which is needed by a UE to decode any other physical layer channels. The system frame number is also included in the MIB for synchronization purposes. The remaining SIBs and data transfer is done over the NPDSCH. The NPDCCH is used to control the data transfer between the UE and the eNodeB.

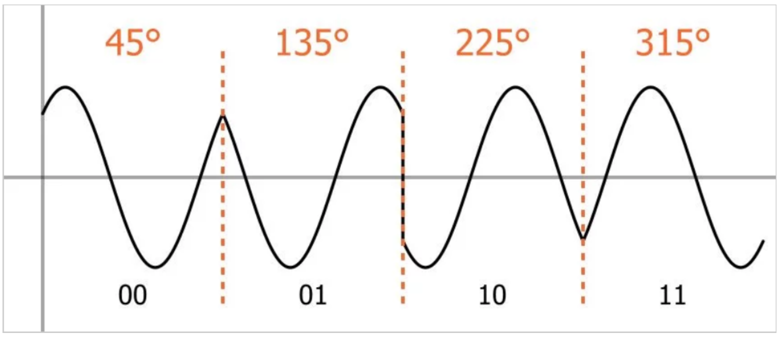


Figure 2.9: The four possible phase shifts of QPSK and the digital data they represent (correspondence scheme is 45 + 90k degrees, k= 0,1,2,3,4. As long as the transmitter and receiver agree to interpret phase shifts in the same way, different correspondence schemes can be used.). Adapted from [Kei16]

In Section2.3.3 we discussed the modulation schemes for NB-IoT. QPSK is always used on the physical DL channels. A modulation scheme is a rule set for representation of digital data in an analog signal. QPSK is an interesting modlution scheme because it transmits two bits per symbol [Ung82]. In other words, a QPSK symbol does not represent 0 or 1 – it represents 00, 01, 10, or 11, see Figure 2.9. This leads to twice the bandwidth of Binary Phase Shift Keying – which uses two symbols, 0 and 1. Each Narrowband cell is assigned a physical cell ID, called Narrowband physical Cell ID (NCID). There are 504 values defined, and the value is provided by the NSSS.

Frame and Slot Structure

For DL, Orthogonal Frequency-Division Multiplexing (OFDM) is applied. OFDM is a specialized frequency-division multiplexing (FDM) method, with the additional constraint that all subcarrier signals within a communication channel are orthogonal to one another [NP00]. OFDM splits data into small sub-carriers, on neighbouring frequencies, over a single channel. It allows sending more data than through single carrier modulation techniques, and at a higher rate. OFDM also handles phenomena such as interference, noise or multipath significantly more efficiently than other modulation methods [NP00]. The OFDM signal is generated by taking the Inverse Fast Fourier Transform (IFFT) of the QPSK subsymbols. The signal consists of N sinusoids[5] with spacing $1/T$ that are modulated by data symbols that have a duration T, equal to the inverse channel spacing [PBM95]. This makes it so that although the modulated sinusoids overlap spectrally, they are orthogonal. Figure 2.10 illustrates the frequency spectra of the signal.

[5] A sinusoid is a mathematical curve that describes a smooth periodic oscillation [Spr14].

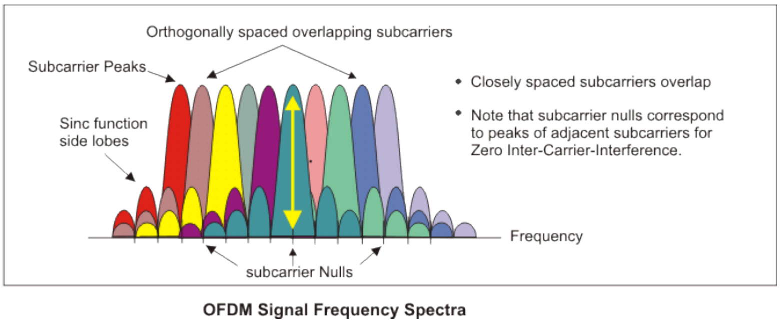

Orthogonally spaced overlapping subcarriers
Subcarrier Peaks
Sinc function side lobes
• Closely spaced subcarriers overlap
• Note that subcarrier nulls correspond to peaks of adjacent subcarriers for Zero Inter-Carrier-Interference.
Frequency
subcarrier Nulls
OFDM Signal Frequency Spectra

Figure 2.10: OFDM signal frequency spectra, adapted from [Tec00]

OFDM needs to be deployed with a Cyclic Prefix (CP) to operate reliably [Not]. The CP has two main functions. It provides a guard interval to eliminate Intersymbol Interference (ISI). ISI is the distortion of a signal where one symbol interferes with subsequent symbols, having a similarly unwanted effect as noise – making the radio communication less reliable and increasing the Bit Error Rate (BER), the expected percentage of erroneous bits [Hum91]. CP also repeats the end of the symbol, making the linear convolution of subcarriers possible to model as circular convolution. This is what separates CP from a prefix and where it gets its name from. A convolution is an integral that expresses the amount of overlap of one function f as it is shifted over another function g [Wei]. The cyclic convolution that is achieved with CP and the use of IFFT for OFDM allows the signal to be decoded with a Fast Fourier Transform (FFT) which is an efficient way to compute the Fourier transform of a signal [SJHB87]. For multiple access, Orthogonal Frequency-Division Multiple Access (OFDMA) is used, which is a multi-user version of the OFDM digital-modulation technology. OFDMA subdivides groups of OFDM symbols into smaller frequency allocations, called Resource Units (RUs). In Figure 2.11, the time-frequency view of a typical OFDMA signal is shown for a case where three users are present. The figure shows that the users' signals are either separated in the time-domain by using different OFDM symbols and/or in the subcarrier domain.

The DL of NB-IoT uses OFDM with a 15 kHz subcarrier spacing with a CP. The OFDM symbols consists of 12 subcarriers occupying the 180 kHz bandwidth, as 20 kHz of the 200kHz bandwidth is the guard-band of 10 kHz on both sides. Seven OFDMA symbols are bundled into one slot, giving the resource grid illustrated in Figure 2.12 [3GP16b]. The resource grid, which is the same for NB-IoT DL and DL is the same as for LTE which makes NB-IoT possible to deploy in in-band mode.

The slots as shown in Figure 2.12 are summed up into subframes and radio frames in the same way as for LTE [3GP16b], the structure is shown in Figure 2.13.

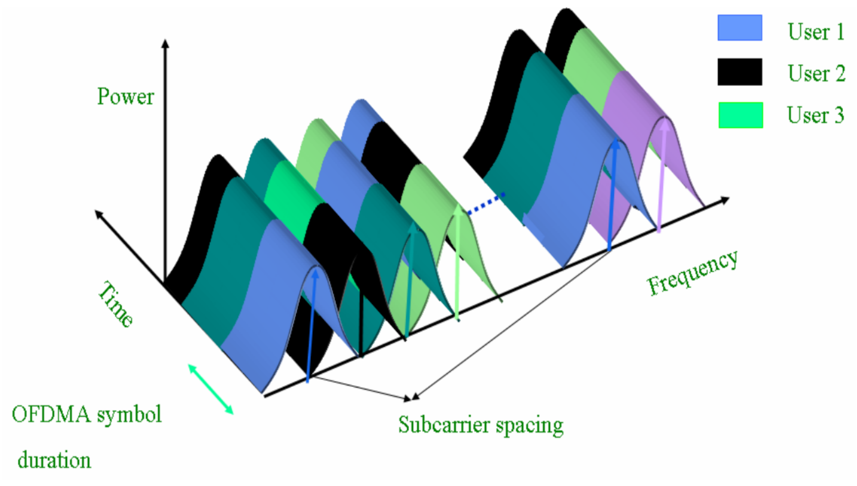


Figure 2.11: Time - Frequency view of an OFDMA signal. Both time and frequency resources are used to support multi-user transmissions. Adapted from [SV06]

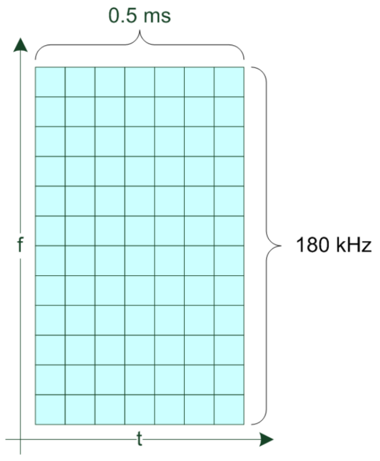


Figure 2.12: The resource grid for one slot. 12 subcarriers (vertically), seven OFDMA symbols (horizontally).

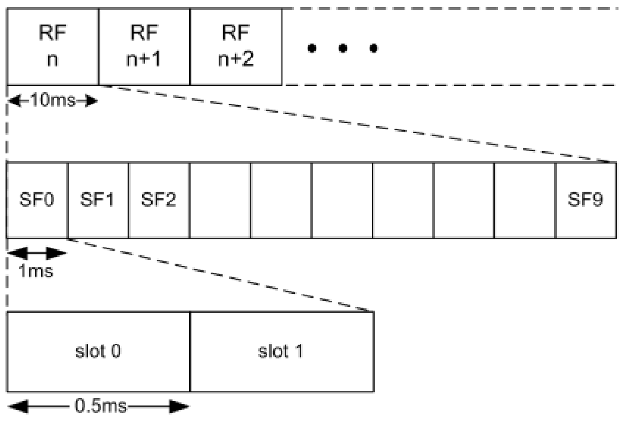


Figure 2.13: NB-IoT UL and DL system frame structure with 15 kHz subcarrier spacing. RF = Radio Frame, SF = Subframe. Adapted from [3GP16b].

Within a system frame there are 1024 radio frames of 10ms time duration, each partitioned into 10 subframes consisting of two slots. The concept of *hyper frames* is introduced, being 1024 system frames which corresponds to around 3 hours [3GP16b].

2.4.2 Uplink

There are two physical channels defined for uplink transmissions

- Narrowband Physical Uplink Shared Channel (NPUSCH)
- Narrowband Physical Random Access Channel (NPRACH)

and one signal

- Demodulation Reference Signal (DMSR)

All data is sent over the NPUSCH, with the exception of random-access transmissions. This means that control information for uplink does not have a separate channel as in LTE.

Slot Structure

Uplink does not use standard OFDMA for multiple access as in downlink, but rather Single Carrier Frequency Division Multiple Access (SC-FDMA) which is a special form of OFDMA with precoding [YAFB10]. The full details of multiple access

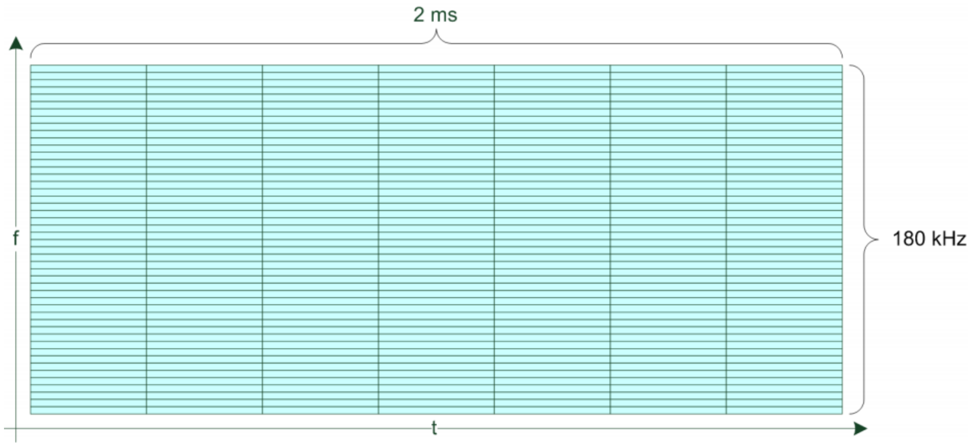


Figure 2.14: Resource grid for 3.75 kHz subcarrier spacing. There are 48 subcarrier for the 180 kHz bandwidth

technique are laid out in [Sli07]. The main benefit of applying SC-FDMA is that the issue of having a high peak-to-average power ratio which leads to a sensitivity to non-linear effects[6] is solved. This is done by introducing a selection of a precoding scheme that distributes the power of each symbol over the OFDM block. [Sli07] shows through simulations the performance gain on the bit-error-rate [7]. SC-FDMA is often chosen for uplink communication because the devices greatly benefit in terms of transmit power efficiency [MLG06], which is especially important for IoT devices. When applied the eNodeB decides between a 3.75kHz or 15kHz subcarrier spacing.

The resource grid is the same as previously illustrated for downlink when 15kHz subcarrier spacing is applied. With 3.75kHz, the resource grid is as shown in Figure 2.14. Note that there are still 7 OFDM symbols within a slot.

2.5 3GPP Energy Calculations for an NB-IoT Device

3GPP has in their report TR 45.820 for cellular IoT [3GP15a] provided the calculation for the battery life for an NB-IoT device in chapter 7.3.6.4 with “Energy consumption evolution” and methodology to calculate in chapter 5.4 with “Energy consumption evolution methodology”. The goal of the work is to be able to calculate the achievable battery life for an IoT device using a specific solution with NB-IoT – which is the same goal as that of this thesis. However, the calculations they lay out require the following Key Input Parameters [3GP15a]:

[6]Non-linear effects include power amplification at the transmitter
[7]The Bit-error-rate is the likelihood of a bit misinterpretation due to electrical noise

1. Battery capacity (Wh): **5**
2. Battery power during Tx(mW)
3. Battery power for Rx (mW)
4. Battery power when Idle but not in PSM (mW)
5. Battery power in PSM (mW) = **0.015**
6. Time between end of IP packet carrying "report" and start of IP packet carrying "ack" on radio (ms)= **1000**
7. Number of reports per day
8. Rx time from PSM exit to re-entry into PSM (ms)
9. Idle time from PSM exit to re-entry into PSM (ms)
10. Tx time from PSM exit to re-entry into PSM (ms)
11. Time from last Rx or Tx activity to entry into PSM (ms) = **20000**

The list includes *eleven* parameters needed to calculate the battery lifetime of an NB-IoT device. It contains four value assumptions, which are highlighted with bold letters. Among the assumptions, the one that stands out is the time from Rx/Tx to PSM = $20.000ms$. This would make the high energy consumption from staying in Connected Mode last 20 seconds longer than the actual transmission/reception. They note that this is the default value in [3GP15a], but in another several hundred pages long document, 3GPP specifies that this timer can be set to "not used", saving the 20 seconds of high energy consumption [3GP15c]. This would make it so that anyone using the formulas would underestimate the device lifetime. That is not the main issue with the presented list of knowledge needed for their calculations – the prediction would also only be possible if the party responsible for the calculations has obtained the remaining 7 variables. Battery power in the different power modes and times spent in them is not something a developer designing an NB-IoT system would know without spending time measuring the different values.

The conclusion is that the calculations are very complex, require specific measurements on the system, and give an unreasonable value for the energy consumption because of a strange assumption. In fairness, the calculations were buried in a several hundred pages long document, and probably not constructed to be used by a software developer/architect. The thesis aims to make a model for energy consumption that is simple to use for a developer making an application, with input parameters being the following:

1. Data collection rate
2. Data sample size
3. Device memory
4. Analysis interval

Data collection rate is how often the device produces data that it is programmed to transmit. *Data sample size* is the size of the individual data samples produced. *Device memory* is how much data the NB-IoT device is able to store, and *analysis interval* is how often the data needs to be collected for analysis by the system. The thesis assumes that these values are something a developer is familiar with when designing an IoT application.

Chapter 3 Methodology and Design

This chapter focuses on a systematic, theoretical analysis of the methods applied in the thesis. The methods and resources utilized for the experiments conducted throughout the thesis will be introduced and the choice motivated where the resource is a deliberate choice among several alternatives. The chapter explains why experiments were conducted in a certain manner, the order they are organized in and how to relate an experiment to the larger goal of the thesis.

3.1 Scientific Method

For this thesis the scientific method used is a combination of methodologies. Some experiments use quantitative methods; they seek to support or dismiss a hypothesis. Other experiments are conducted with qualitative methods, where the goal of the experiment is to learn from observing a certain phenomenon. The information gathering process, including acquiring knowledge about the field and the study of the relevant background material was done using fundamental research techniques. The experiments were affected by design science and were conducted using a black box testing method. The terms and what they comprise of will be highlighted in the following sub-sections.

3.1.1 Design Science

The experiments are influenced by design science as theorized in [Wie14c], with the design cycle as depicted in Figure 3.1 – inspired by the engineering cycle. Each cycle starts in the "Problem Investigation" phase. In this phase, the focus point is on the stakeholders[1], which in the case of the thesis is the responsible professor and the supervisor. The point is that every cycle begins with a reminder of their goals, and thus how the goals of the experiment align with the overall objectives can be

[1]A stakeholder is a party with an interest in an enterprise or project.

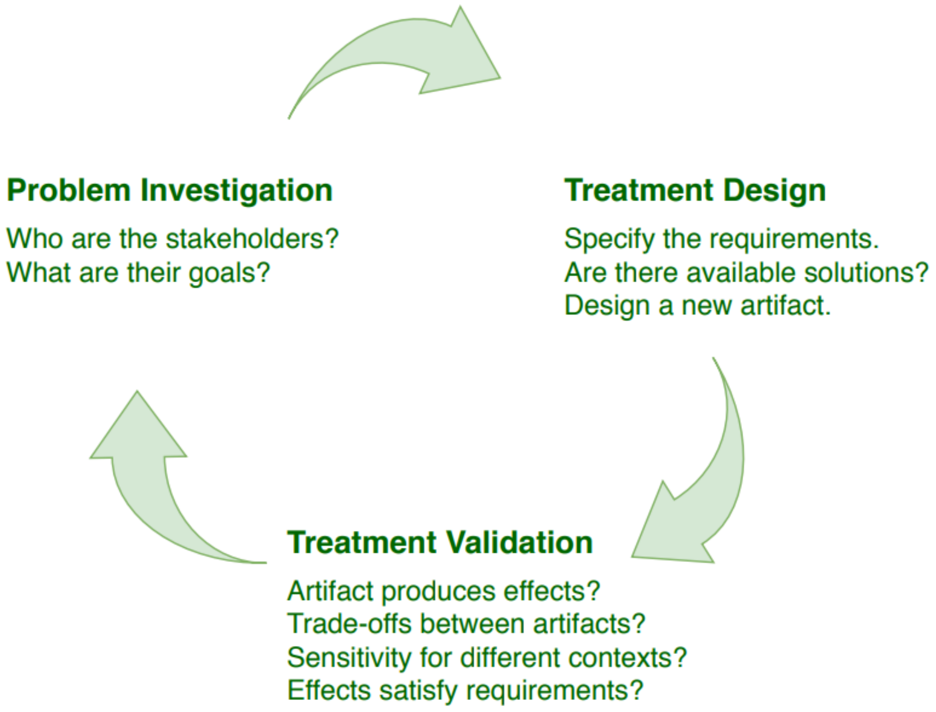

Problem Investigation
Who are the stakeholders?
What are their goals?
Treatment Design
Specify the requirements.
Are there available solutions?
Design a new artifact.
Treatment Validation
Artifact produces effects?
Trade-offs between artifacts?
Sensitivity for different contexts?
Effects satisfy requirements?

Figure 3.1: The General flow of the Design Cycle used for the Thesis Experiments.

understood when entering the next phase. Problem investigation boils down to the question [Wie14a]: Which phenomena must be improved and why?

The second phase is “Treatment Design”, in which the researcher designs an artifact. The design science definition of an artifact is “something created by people for some practical purpose” [Wie14a]. This includes – but is not limited to – algorithms, methods, notations, techniques, and conceptual frameworks. The purpose of the phase is not simply designing an artifact, but to design a desired interaction between the artifact and the problem context, with intent to treat the problem.

The third phase entitled “Treatment Validation”, and as the name implies, this is where phase two is validated. The objective is to investigate whether the design(s) treat the problem or not, and to which degree. The phase is also concerned with comparing designs. The problem is that there is no real-life implementation available on which to test whether the treatment design contributes to the stakeholders goal [Wie14b]. The researcher instead builds validation models of the artifact in context to check if the effects satisfy the requirements.

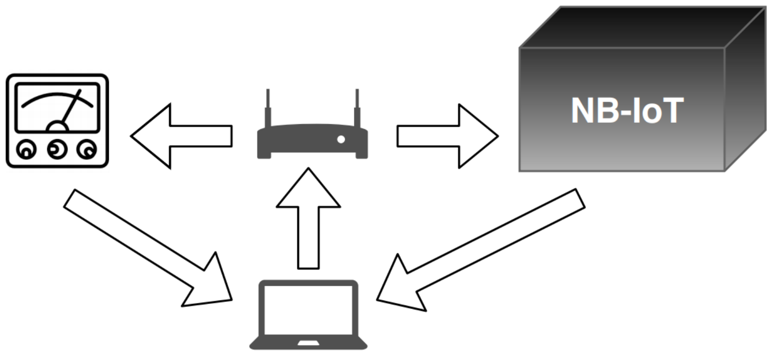


Figure 3.2: NB-IoT as a black box. The laptop instructs the modem to communicate with the box, and an application is updated based on the events inside the box such that the laptop can see that the transmission was received. A power measuring capable device is recording the energy consumption of the modem and sending the results to the laptop.

3.1.2 Black Box Testing

Black box testing is a popular testing method traditionally used for software in which the tester is aware of *what* the system does but not *how* it does it. The definition is as follows:

Definition 3.1. Black box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system [ND12].

The method is powerful as it allows unnecessary details to be excluded and makes the tester focus on the desired testing objective. The testing method can be extended by allowing the black box to be any artifact as defined by design science. After applying the method, the result is a general setup as depicted in Figure 3.2. This is a fitting illustration as the devices allow the programmer to give instructions to the modem, while any communication between modem and the eNodeB is unknown even when radio-sniffed[2] as the data transmitted should be encrypted with the SIM-card. Even if the transmission is not encrypted, the internal structure and communication in the LTE network would be hidden. This is including eNodeB to Mobile Management Entity (MME) and MME to Serving Gateway (S-GW) communication, which has not been detailed as it is irrelevant for the thesis. Instead, NB-IoT is treated as a

[2]Using an antenna to receive signals intended for another device.

Features:

- A sample rate of up to 4k samples per second.
- Measurement range of 1µA-5A.
- Data+Power via USB. 2.1mm DC jack for external power.
- Can record current, voltage and UART logs, displaying them as charts and interactive measurements.
- The ADC has a differential measurement range of -81.9175 mV to 81.92 mV at 1 ohm.
- Otii can measure subsystems while also monitoring the main supply.
- Two General Purpose Inputs
- Two General Purpose Output
- The Sense pins.
- Synchronize debug logs

Figure 3.3: The OTII Arc with features listed, adapted from [ce].

black box to which the modem can communicate and through which an application visible to the laptop is updated.

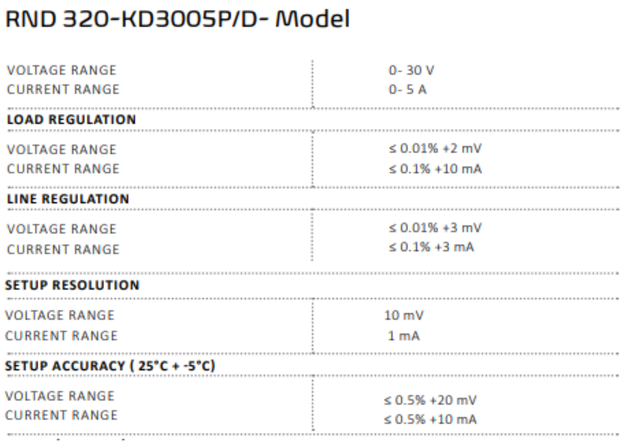
3.2 Equipment and Software

This section describes the different hardware and software used in experiments in this thesis, describing what each piece equipment is for, how we use it and why. As a general note, the usage of equipment needed for experiments have mostly been based on what is already available in the lab. The OTII Arc and the Exploratory Engineering board were ordered specifically to be used in the thesis. Where it is appropriate, we discuss the potential limitations of the chosen equipment.

3.2.1 The OTII Arc by Qoitech

The Otii software runs on multiple platforms, Windows, Ubuntu and macOS. The Otii Arc acts both as a power supply to the IoT device being tested, and as a current and voltage measurement unit. It provides up to 5 V output voltage and runs high resolution current measurements with a sample rate up to 4 kilo-samples per second for the range of 1 μ A - 5 A. See Figure 3.3 for a list of features.

The setup for the full OTII package (OTII Arc and OTII software) is simple. The software is downloaded and installed ready to go. The device is then plug-and-play. Qoitech has a forum for discussions where possible issues can be discussed, making it less likely to get stuck on a fixable issue. The device was ordered for the lab as the thesis was started. The most promising feature of the OTII is being able to use serial communication through the OTII Arc, so that the Universally Asynchronous Receiver/Transmitter (UART) logs can be synchronized to the power measurements. This would make the results more easily verifiable.



RND 320-KD3005P/D- Model

VOLTAGE RANGE	0- 30 V
CURRENT RANGE	0- 5 A
LOAD REGULATION	
VOLTAGE RANGE	≤ 0.01% +2 mV
CURRENT RANGE	≤ 0.1% +10 mA
LINE REGULATION	
VOLTAGE RANGE	≤ 0.01% +3 mV
CURRENT RANGE	≤ 0.1% +3 mA
SETUP RESOLUTION	
VOLTAGE RANGE	10 mV
CURRENT RANGE	1 mA
SETUP ACCURACY (25°C + -5°C)	
VOLTAGE RANGE	≤ 0.5% +20 mV
CURRENT RANGE	≤ 0.5% +10 mA

Figure 3.4: The RND 320 with specs, Adapted from [elf] and [RND].

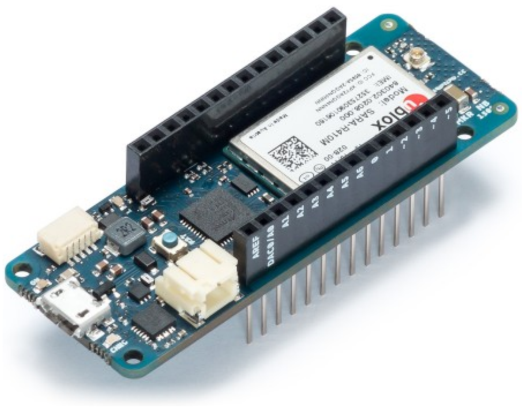
We power the OTII Arc with a USB cable to a Laptop for most of the experiments. However, the OTII needs an external power source to be able to output more than 4.2V. The Arduino MKR NB 1500 has a circuit operating voltage of 3.3V, but voltage input pins for 5V. The setup in Section 4.2 describes the process. The external power supply used is the RND lab power supply introduced in the following subsection.

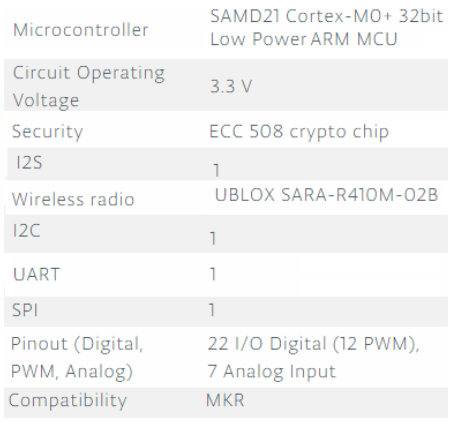
3.2.2 The RND lab DC Power Supply RND 320-KD3005D

As mentioned, the RND lab DC power supply RND 320-KD3005D (RND 320) was chosen as an external power supply for the OTII Arc when using a board that requires more than 4,2V input. The power supplier is among several available in the lab and was chosen for its voltage range and setup accuracy, which can be seen in Figure 3.4. For our purposes we would need a steady supply of 8V, which the RND 320 supplies without issues, as it allows to specify voltage up to 30V with a precision of 10 mV. For supplying the OTII we would need a barrel jack connector. The ones available in the lab would not have a banana plug for input – but splicing the cord and twisting the wire onto the supply of the RND 320 is made simple by having adjustable plastic knobs covering the metal output terminals. The RND 320 connects to a wall outlet.

3.2.3 The Arduino MKR NB-1500

The Arduino MKR NB-1500 (NB-1500) is Arduinos integrated NB-IoT solution. This board is designed for global use, providing NB-IoT deployed bands 2, 3, 4, 5, 8, 12, 13, 20, 28. We only need 20 as that is what Telenor is using in Norway. It has a SAMD21 Cortex-M0+ 32bit Low Power ARM MCU micro controller with a 3.3V





Microcontroller	SAMD21 Cortex-M0+ 32bit Low Power ARM MCU
Circuit Operating Voltage	3.3 V
Security	ECC 508 crypto chip
I2S	1
Wireless radio	UBLOX SARA-R410M-02B
I2C	1
UART	1
SPI	1
Pinout (Digital, PWM, Analog)	22 I/O Digital (12 PWM), 7 Analog Input
Compatibility	MKR

Figure 3.5: The Arduino MKR NB 1500. Adapted from [MKR19].

circuit running voltage and is able to run Arduino scripts. The uBlox SARA-R410M-02B modem is chosen for this board and is integrated, which gives access to the aforementioned NB-IoT bands. The modem supports NB-IoT, PSM, eDRX and more [uBl]. The board is connected to a laptop or special equipment like the OTII with a USB to micro-USB cable and a chip on the board regulates the voltage. This may lead to increased power consumption. The Board is purchased with a dipole GSM antenna for Arduino MKR boards, which is a GSM antenna (850/900/1800/1900 MHz) that can be attached to the board with the Micro UFL connector. The antenna being produced for the board by the same company that produced the board guarantees compatibility. The Arduino MKR NB 1500 was chosen for experiments because it was available in the labs after being used in the introduction course TTM4175 in Spring 2019 [Kra]. The author attended the lab and was unable to configure the device to transmit. The initial setup of the device during the lab was unsuccessful, and the antenna and USB cable needed to be replaced before resulting in a success. Once the device and the Arduino software are configured correctly and every hardware component is working, it is plug-and-play. Being Arduino Software compatible, it is easy to write scripts for the device.

3.2.4 EE-NBIOT-02

The EE-NBIOT-02 board (depicted in Figure 3.6) is a breakout module for the modem uBlox SARA-N211 with a circuit operating voltage of 3.3v. The board comes supplied with an LTE antenna and is equipped with a SIM-card holder and a reset button. The board is designed to add as little complexity as possible to the modem, making it excellent as a tool for measuring the radio energy consumption of NB-IoT.

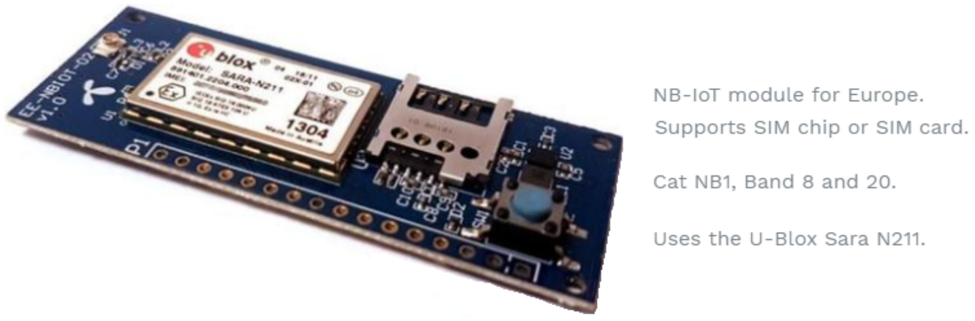
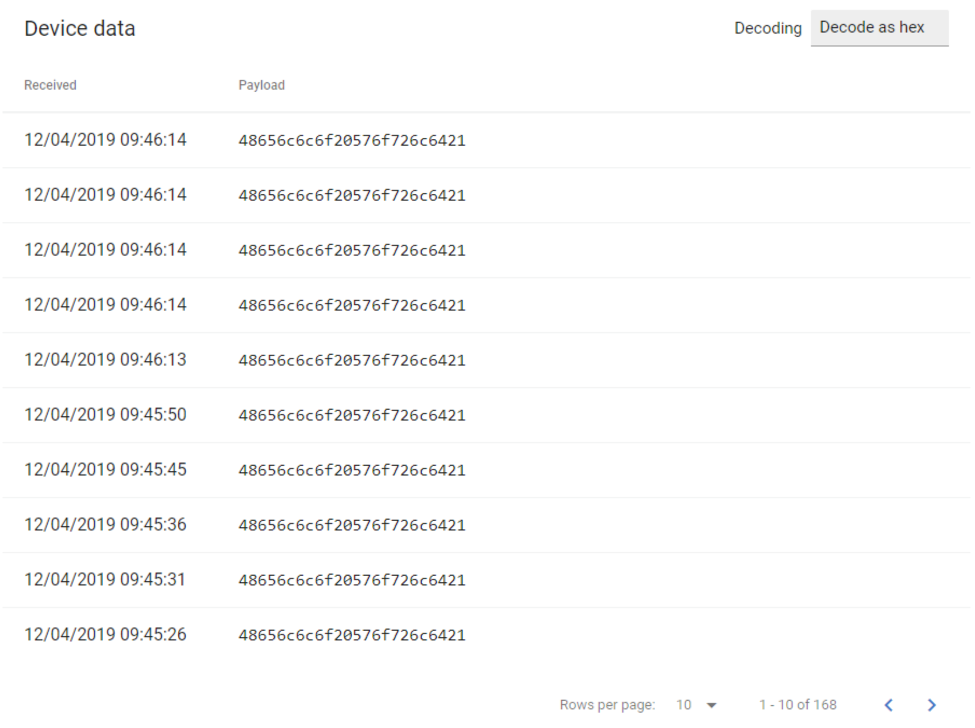

NB-IoT module for Europe.
Supports SIM chip or SIM card.
Cat NB1, Band 8 and 20.
Uses the U-Blox Sara N211.

Figure 3.6: The EE-NBIOT-02 breakout board. Adapted from [Enga].

To connect the board to a computer we use a TTL-232R-3V3 cable which is a USB to Transistor-Transistor Logic (TTL) cable that supplies 3.3V. The term TTL is a little confusing. UARTs transmit one bit at a time at a specified data rate and this method of serial communication is sometimes referred to as TTL serial. This way we can communicate with the modem using any programming language or specific serial communication software. We chose the latter, using CoolTerm as serial communication interface. Coolterm does not support scripting, so we are manually sending Attention Commands or Hayes' Commands (AT-commands) to the modem. This is time consuming and engaging, but we are in great control of the modem. The modem is powered by the OTII at the same time, so that energy measurement can be taken. This way we can see the modem react to our AT-commands through the energy measurements instantly.

3.2.5 Verifying Transmissions with Telenors NB-IoT Web Solution

Telenor has developed an interface for NB-IoT where it is possible to add devices based on IMSI and IMEI. Once added, the application will list any uplink transmission data from the device along with a timestamp. Figure 6.6 shows the ten latest data transmissions from the device "EE-NBIOT02 v1.0". This is very useful for the black box testing as it allows verification of reception of the transmissions, which in turn means they were transmitted successfully. The packet batch transmission experiments in Chapter 6 would not be possible without be able to verify that the correct number of packets were transmitted in the batch in a correct manner.


Device data

Decoding: Decode as hex

Received	Payload
12/04/2019 09:46:14	48656c6c6f20576f726c6421
12/04/2019 09:46:14	48656c6c6f20576f726c6421
12/04/2019 09:46:14	48656c6c6f20576f726c6421
12/04/2019 09:46:14	48656c6c6f20576f726c6421
12/04/2019 09:46:13	48656c6c6f20576f726c6421
12/04/2019 09:45:50	48656c6c6f20576f726c6421
12/04/2019 09:45:45	48656c6c6f20576f726c6421
12/04/2019 09:45:36	48656c6c6f20576f726c6421
12/04/2019 09:45:31	48656c6c6f20576f726c6421
12/04/2019 09:45:26	48656c6c6f20576f726c6421

Rows per page: 10 1 - 10 of 168

Figure 3.7: The ten lastest uplink transmissions from the EE-NBIOT-02 with timestamps.

3.3 Experiment Structure and Prerequisites

This section will introduce the reader to the structure which all the experiments will follow. A single structure has been chosen as it makes it clear what the motivation is and what the results are for each experiment. The section then presents an overview of commands used to communicate with a modem, which will be used throughout the next chapters. Additionally, the section helps the reader contextualize power measurements by illuminating the energy units, how they relate, and which orders of magnitude to expect.

3.3.1 Experiment Structure

The thesis applies a fixed structure to make it easier for the reader to follow the research. Every experiment will start we a short description of the experiment goal, the hypothesis going in and the method used throughout, as shown:

Experiment goal: The motivation for the experiment

Hypothesis: The hypothesis the experiment evaluates

Method used: Short description of method and equipment/software

Throughout the section the findings will be compared to the expected results, and the hypothesis changed, extended or falsified and discarded accordingly. The analysis will motivate the next experiment, which will be introduced the same way.

3.3.2 Modem Communications with AT-Commands

To measure the energy a modem consumed when performing a task, we need to be able to get the modem to perform tasks at our will. For most modems, including the SARA N2xx and SARA N4xx which we are using for the thesis, this can be done using AT-commands. The AT-commands configure and enable the cellular module functionalities according to 3GPP normative and modem manufacturers specifications. The AT commands are issued to the module via a hyper terminal through a command line [DAS]. There are three main categories of AT-commands, set commands, read commands and test commands [ubl19].

Set commands set values or perform actions. Example: AT+CMD=1 where

- AT is the command line prefix
- + is the prefix for extended commands
- CMD is the body of a basic command
- 1 is a subparameter (multiple subparameters are separated by commas)

Read commands check the current values of subparameters. Example: AT+CMD? where

- AT is the command line prefix
- + is the prefix for extended commands
- CMD is the body of a basic command
- ? represents a read command

Test commands test the existence of the command and provide information about the type of its subparameters. Example: AT+CMD=? where

- AT is the command line prefix
- + is the prefix for extended commands
- CMD is the body of a basic command

- =? represents a test command for checking possible sub-parameter values

The Arduino MKR NB 1500 and the EE-NBIOT-0x boards all use ublox modems, which mean they support using AT-Commands for modem communication. For the EE-NBIOT-0x the AT-Commands will be issued directly over a serial port, while with the NB-1500 the AT-Commands will be programmed into the scripts. The ublox documentation is very detailed - making it quite a task to go through - but every AT-command is described thoroughly.

3.3.3 Understanding the Energy Measurements

Throughout the following experiments there will be a constant discussion of energy consumption with statements such as “the energy capacity is 40mAh” and “the average power is $60\mu W$”. This section will explain the terms and how they relate as well as giving a context for the order of magnitude that is expected. The following definitions are the basis for understanding energy measurements.

Definition 3.2. Ampere: Ampere or amp (symbol: A) is the unit of electrical current. One Ampere is defined as the current that flows with electric charge of one Coulomb per second: $1A = 1C/s$ [MW81].

Definition 3.3. Watt: Watt (symbol: W) is the unit of power. One watt is defined as the energy consumption rate of one joule per second: $1W = 1J/1s$. One watt is also defined as the current flow of one ampere with voltage of one volt: $1W = 1V \times 1A$ [MW82b].

Ampere and Watt are useful when discussing how much energy is consumed at a single point in time, with Definition 3.3 showing how to convert from ampere to watt and vice versa using the voltage. When discussing battery capacity or energy consumed during some period an additional dimension is needed, *time*. The following definitions are useful.

Definition 3.4. Ampere hour: An ampere hour or amp hour (symbol Ah) is a unit of electric charge, having dimensions of electric current multiplied by time, equal to the charge transferred by a steady current of one ampere flowing for one hour, or 3600 coulombs [MW82a].

Definition 3.5. Watt hour: The watt-hour (symbolized Wh) is a unit of energy equivalent to one watt (1 W) of power expended for one hour (1 h) of time [MW88].

A device consuming $60W$ power over a period of 75 seconds has a total energy consumption of $60 \times \frac{75}{3600} = 1,25Wh$. Running on a voltage of $5V$, this translates to

$1,25Wh \div 5V = 0,25Ah$. The energy consumption and capacity units are defined, but how does one know which order of magnitude of energy consumption to expect from an IoT application, specifically NB-IoT? This is answered by looking at battery life-time promises by 3GPP and battery capacities of batteries typically used for IoT devices.

3GPP promises a battery life-time of ten years with a capacity of $5Wh$ [3GP15b]. This results in an average power consumption of

$$\frac{5[Wh]}{365,25[days] \times 24[hours]} \approx 570\mu W \tag{3.1}$$

. The devices used for the thesis have circuit voltages of $3,3V$ and $5V$, meaning that the average current should be $570\mu W \div 3.3[V] \approx 173[\mu A]$ and $570[\mu W] \div 5[V] \approx 114[\mu A]$ respectively. This being the consumption averages, it is trivial that the deep sleep values should be lower. We know from Section 2.5 that 3GPP used $15\mu W$ as a set value for PSM power consumption, so a value in the same order of magnitude is expected.

Chapter 4 Initial Experiments

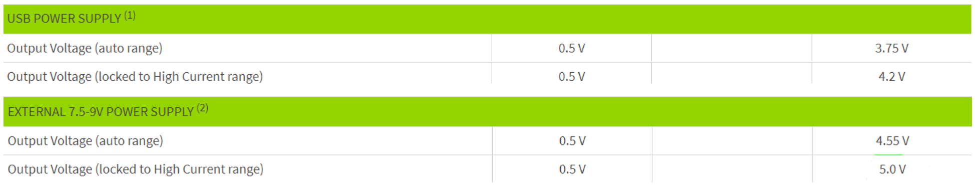
The thesis consists of many experiments all conducted to understand the energy consumption of NB-IoT and answer the research questions. As described, the experiments will start simple and grow in their complexity after knowledge from previous experiments is obtained. Before any of the following experiments were conducted, the Arduino MKR NB 1500 (NB-1500) was successfully connected to an existing NB-IoT network implementation, called “Telenor Connexion - Start IoT” [Tel19]. Through the application supplied by the Internet Service Provider (ISP) Telenor, it was verified that the NB-1500 was able to send data using NB-IoT.

4.1 Relevant AT-commands for the ublox SARA-R410M-02B Modem

The NB-1500 has the ublox SARA-R410M-02B modem integrated, which can be reached by using AT-commands – as described in Section 3.3.2. We will use several of the commands for the following experiments. Table 4.1 shows all AT-commands used with the NB-1500. The table will be referred to throughout the experiments.

Table 4.1: Table containing all used AT-commands. RAT here means Radio Access Technology.

Type	**Command**	**Use**
set	AT+URAT=8	Set RAT to NB-IoT.
read	AT+URAT?	read RAT set correctly.
set	AT+COPS=1,2,<val>	Set operator to <val>.
set	AT+CEDRXS=3,5	Disable eDRX.
set	AT+CPSMS=1,,,<val1>,<val2>	set T3324 and T3412



USB POWER SUPPLY (1)			
Output Voltage (auto range)	0.5 V		3.75 V
Output Voltage (locked to High Current range)	0.5 V		4.2 V
EXTERNAL 7.5-9V POWER SUPPLY (2)			
Output Voltage (auto range)	0.5 V		4.55 V
Output Voltage (locked to High Current range)	0.5 V		5.0 V

Figure 4.1: Output Voltage Overview for OTII

4.2 Experiment one: Setup with the OTII and Arduino MKR

Experiment goal: Measuring energy of the NB-1500 with the OTII.

Hypothesis: The OTII is able to measure energy on the NB-1500.

Method used: Black box method, Arduino Software on the NB-1500.

Before starting to measure selected transmissions in order to understand the consumption of the different power modes, the basics are needed. In this experiment the goal is to be able to measure the energy of a simple NB-IoT transmission.

The NB-1500 is very small which makes measuring the power flow using probes difficult and inaccurate. With this in mind, the OTII was chosen for the task. However, the OTII measures while powering the device. Thus the Arduino sketch needs to be uploaded to the NB-1500 before use. The device is then disconnected from the laptop and only connected to the OTII for power. This can be tedious for testing purposes when changing sketches often would be necessary, but it is a workable solution.

Another issue came with the OTII – it only supplies up to 4.2V without an external source as see in Figure 4.1. The NB-1500 has an operating voltage of 3.3V, but an input voltage of 5V. Although the "ON"-LED would light up when supplied with 3.3V or 4.2V the device would not charge, making it unable to use the modem or antenna.

The OTII is able to supply 5V with a 7-9V external power supply. This was done using RND 320, which is a power supply unit for operation with constant voltage. The OTII requires a DC barrel jack connector for external supply, while the RND 320 has no barrel jack connector. This was solved as mentioned in Section 3.2.2. Banana connectors were used for connecting the OTII to male-to-male jumper cables via HM6410 clamps. The jumper cables connected to the Arduinos' VIN and GND

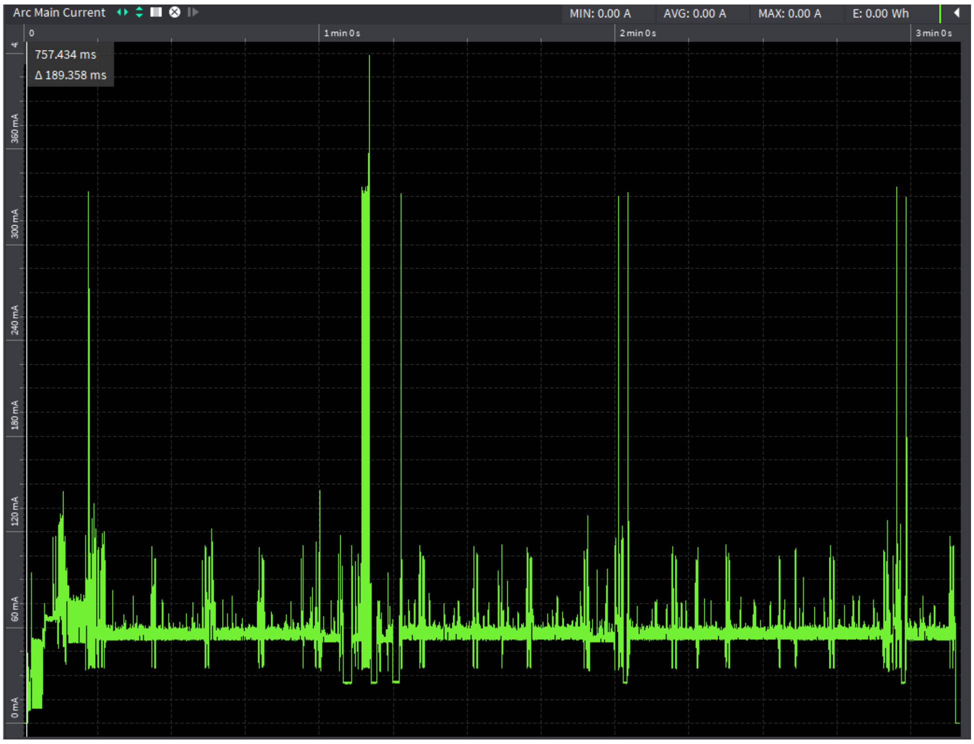


Figure 4.2: The first energy measurement on an NB-IoT transmission scheme

inputs to power it. We used a SIM-card provided by Telenor to connect to their NB-IoT network.

Using the presented setup, energy measurements were done. The Arduino code and algorithms can be found in the Appendix. An algorithm was used is the network setup and connection, and another algorithm used as a loop for sending randomly generated data every 20 seconds. The results can be seen in figure 4.2. From the figure we can see some energy spikes every twenty seconds, which is what we would expect. We also see some initial energy spikes that we do not really understand as of now, together with several larger energy spikes that are spread out at seemingly random intervals and for which there is no explanation as of now. The main goal of the experiment was achieved; we were able to measure the energy used by the NB-1500 while transmitting using NB-IoT.

4.3 Experiment two: Measuring the Pre-Transmission Power Consumption the Device.

Experiment goal: Measuring the energy consumed when powering the device – before any data is transmitted

Hypothesis: It is possible to isolate the pre-transmission power consumption

Method used: Black box method, Arduino software on the NB-1500

In the previous experiment we noticed that the initial power consumption when powering on the device was different from the rest of the measurements. This is expected as the devices will power up, charge and do initial modem operations to connect to the network. Before we start exploring the energy consumption of the different RRC-states we need to know how long it takes for the device to complete the initial network connection and setup to avoid power interference. For this experiment, Algorithm 9.1 and 9.2 in the appendix are used, however the loop() simply consists of a delay. This makes it so the NB-1500 initializes and connects but transmits and receives nothing. The result was that the average initial setup time over ten samples was 64.83 seconds with a standard deviation of 0.74 for an average of 0.78mAh total energy consumption. In practice, the device would only go through the initial power consumption once, so even though the consumption seems high it really depends on the variable energy consumption later.

4.4 Experiment Three: Measuring the Energy Consumption of the Payload – Part I

Experiment goal: Modelling single packet transmissions with an energy per bit function

Hypothesis: The energy consumed during the transmission can be isolated

Method used: Black box method, Arduino software on the NB-1500

Now that we have been able to verify the network connection and the ability to measure the energy consumption of the NB-1500 it is time to start quantifying the energy consumption. We start by observing what happens when the number of bytes in the payload is increased. The loop() section of the Arduino code is changed as seen in Algorithm 9.4. The transmission scheme consists of sending a data chunk of 18 characters, waiting 10 seconds, sending a data chunk of 296 characters and then repeating after 60 seconds. The choice of 18 characters and 296 characters is made because of the size difference. The actual numbers themselves could have been 20 characters and 300 characters respectively. As we want to see if there is a

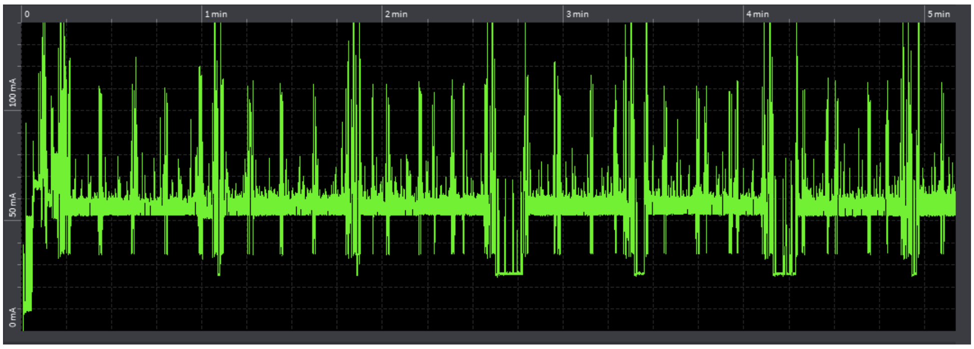


Figure 4.3: The first five minutes after the NB-1500 is powered on. Notice the spikes of different heights occurring every few seconds and the average current of between 50-60mA. This would last for about ten minutes before stabilizing to around 25mA. The reason is unknown.

noticeable effect on the energy consumption when increasing the payload size, the actual numbers are not important. We will call an iteration of the loop a transmission set. As in Experiment One we are using the NB-1500 to connect to Telenors' NB-IoT network and measuring the energy consumption with the OTII.

After running the experiment three times it became apparent that the device stabilizes in terms of the duty cycle after around 10 minutes, with seemingly random power spikes with few seconds between. This can be seen in Figure 4.3. Why this is happening is unclear. The best guess is that the device performs some initial configurations when powered on. Ten minutes might seem like a long time, but remember that the device is made to be powered once and stay on for several years with as few reboots as possible. The technical report for the device does not mention anything about the phenomena or what may be causing it [MKR19]. However, it makes measuring the energy consumption with different configurations a more extensive task. Before the stabilization it is hard to read what is happening from the measurements. Because of this, we let the device run for twenty minutes to capture ten transmissions after the ten minutes initial start-up.

We use these ten transmissions to compute the average power consumption during connected mode and idle mode. The results of the transmissions will look like figure 4.4. This can be simplified to a square wave function with a minimum of 25mA and a maximum of 60mA, spending 30 seconds in minimum and 40 seconds in maximum alternating. Zooming in on a transmission set we can observe the details of how energy is consumed (Figure 4.5). The two transmissions can be seen.

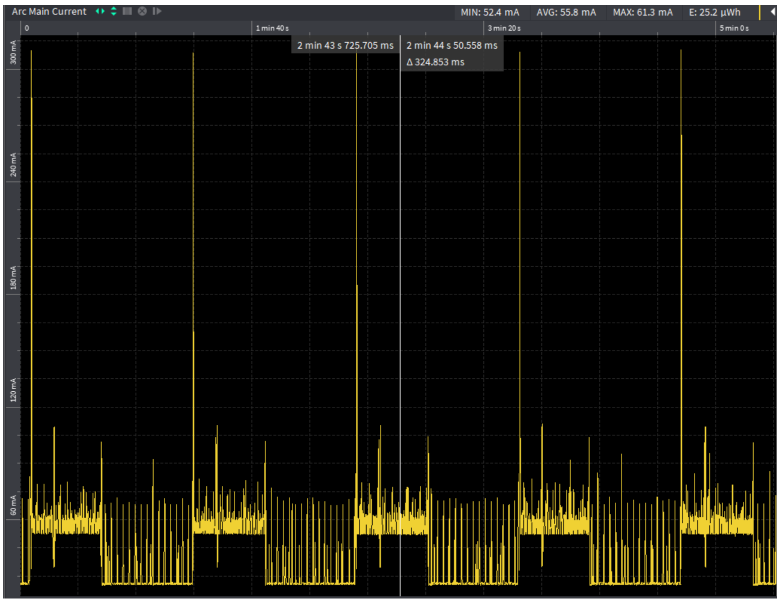


Figure 4.4: Stabilized duty cycle measurements on the NB-1500. We observe that the image resembles a square wave function.

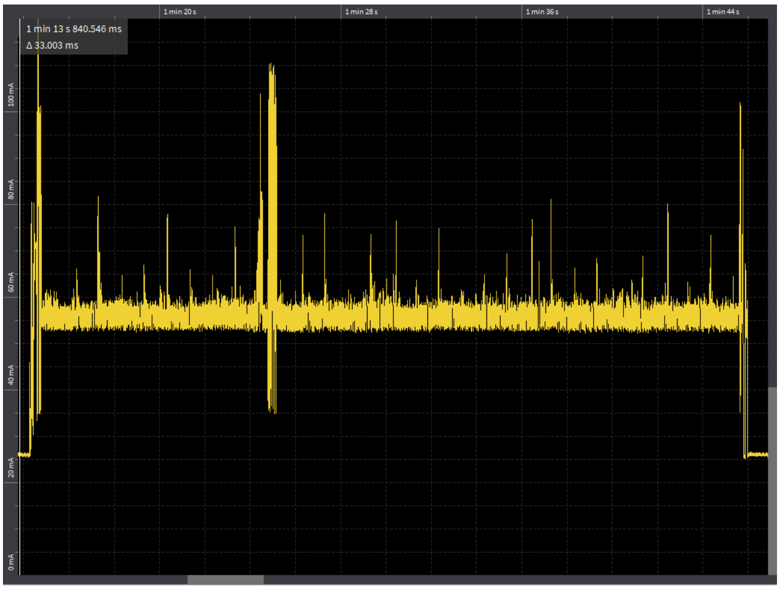


Figure 4.5: Stabilized duty cycle measurements on the NB-1500: zoomed in on a transmission set. The device first enters Connected Mode and transmits a small data chunk, then waits ten seconds before transmitting a larger data chunk. It remains in Connected Mode for approximately 20 seconds before disconnecting.

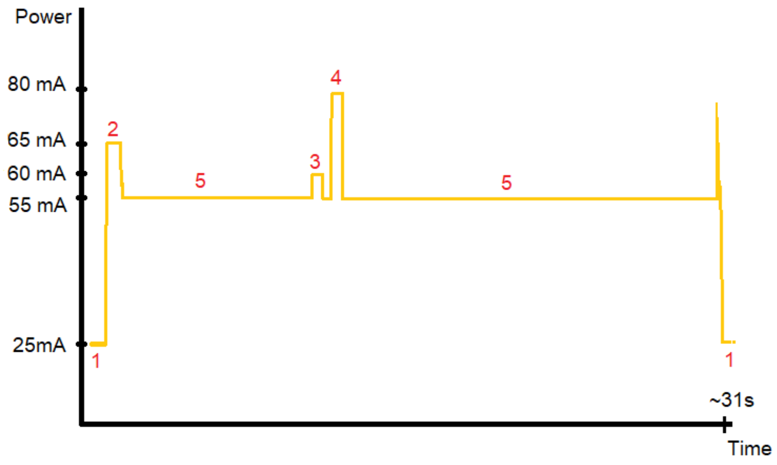


Figure 4.6: A simplification of the image shown in figure 4.5. The energy measurements have been split into five parts: idle(1), transmission one(2), transmission two part one(3), transmission two part two(4), and connected(5).

After running the transmission scheme on the NB-1500 long enough to obtain seven samples of the transmission set in stable condition we computed the average energy consumption and the standard deviation for the different modes and during the transmissions. The energy measurements were split into five parts as can be seen in figure 4.6 to isolate the interesting events. See table 4.2 for the results. The results show that the lowest power consumption achieved was in idle mode – with an average of $\approx 27mA$. With the battery size assumption that 3GPP uses in their calculations, we calculated that the average power consumption should be in the order of $100\mu A$ (Chapter 3). We also note that the total energy consumption over transmission one is

$$66.03mA \times \frac{0.62seconds}{3600\frac{seconds}{hour}} \approx 0.0114mAh = 11.4\mu Ah \tag{4.1}$$

and the total energy consumption of transmission two is

$$63.61mA \times \frac{0.17seconds}{3600\frac{seconds}{hour}} + 78.40mA \times \frac{0.39seconds}{3600\frac{seconds}{hour}} \approx 11.5\mu Ah \tag{4.2}$$

The average energy consumption is almost the same. This is a coincidence, and the results are not reliable. The time spent in transmission one ranged from 0.33 seconds to 0.92 seconds, giving a large standard deviation.

While the experiment yielded some interesting results, especially the square wave approximation, we were not successful in measuring the effect of having a larger payload. This was due to the nature of the transmission scheme. The device would

Table 4.2: The results from Experiment Two divided as seen in figure 4.6. The average current consumption [mA] and time spent [s] for Idle mode, Connected mode, and the different transmission parts as shown in Figure 4.6. "Tx 1" is transmission one, "Tx 2.1" and "Tx 2.2" are the two parts of the second transmission.

Sample number	1st	2nd	3rd	4th	5th	6th	7th	**AVG**	**ST. DIV**
Idle	27.30	27.30	27.30	27.30	27.20	27.20	27.30	**27.27**	**0.05**
Time	38.90	38.69	38.63	38.49	38.48	38.12	38.50	**38.54**	**0.24**
Con.	57.10	57.10	56.90	57.20	57.40	57.60	57.20	**57.21**	**0.23**
Time	31.69	31.59	31.69	31.66	31.80	30.86	31.53	**31.55**	**0.31**
Tx 1	67.20	66.70	64.10	67.60	66.50	64.00	66.10	**66.03**	**1.43**
Time	0.67	0.33	0.52	0.75	0.92	0.46	0.66	**0.62**	**0.19**
Tx 2.1	78.90	82.60	83.50	79.60	82.90	65.40	75.90	**63.61**	**1.05**
Time	0.17	0.17	0.17	0.17	0.17	0.18	0.18	**0.17**	**0.00**
Tx 2.2	78.90	82.60	83.50	79.60	82.90	65.40	75.90	**78.40**	**6.34**
Time	0.44	0.38	0.38	0.44	0.38	0.38	0.37	**0.39**	**0.03**

enter connected mode just before the first transmission, making it impossible to distinguish between the energy consumption of connecting to the network and the energy consumption of sending the first data chunk. Different algorithms and changes to the Arduino loop were employed and can be seen in the appendix. This made it easier to isolate the energy consumption of a transmission. We note the facts that the energy consumption is $27mA$ in sleep mode (two orders of magnitude higher than expected) and the device needs ten minutes to stabilize the energy consumption after booting to make sure the energy chart is coherent. Based on the results, we choose to not use the device for further experiments.

Chapter 5 Modem Experiments

The NB-1500 board by Arduino is very complex and has many hardware elements powered by electricity in addition to the modem – one of them being the micro-controller. As the goal of the research is to model the energy consumption of a NB-IoT application regardless of device, we want to look at the energy consumption of the modem separately. By doing this, we get an overview over the energy characteristics of the radio transmissions themselves. Exploratory Engineering [engc] has created a break-out board for the uBlox modems SARA-N210 and SARA-N211, of which the latter will be used in the following experiments.

5.1 Experiment 1: Setup with the EE-NBIOT-02

Experiment goal: Transmitting with the EE-NBIOT-02

Hypothesis: The modem can be instructed with AT-commands

Method used: Black box method, Coolterm and serial cable

The EE-NBIOT-02 is equipped with a modem and an antenna, but not equipped with a micro-USB power supply or anything else of use to us. The only way to power it is through the pins. To connect the board to a computer we used a TTL-232R-3V3 which is a USB to TTL cable that supplies 3.3V. As the TTL outputs are female, we used male-to-female jumper cables (sc13051 cables) to connect the TTL-232R-3V3 to the board. The baud-rate was set to 9600 as specified by Exploratory Engineering for the board [engc]. With this setup we were able to communicate with the SARA-N21x modem through AT commands using the application CoolTerm to send UART/serial data through a COM-port on the laptop. While the USB to TTL cable allowed for communication and power supply, we were also dependant on measuring the power consumption. This is where the OTII comes in.

5.1.1 Experiment 1.1: Using the OTII for Serial Communication

Experiment goal: Measuring transmissions with the EE-NBIOT-02

Hypothesis: The OTTI Arc can send AT-Commands to the modem

Method used: Black box method, OTII Arc

The computer is simply connected to the OTII through a USB cable. The OTTI then powers the EE-NBIOT-01 via banana cables with HM6410 clamps to male-to-female jumper cables (sc13051 cables) to GND and VCC. For serial communication the OTII is connected to TDX, RDX and DGND via female-to-female jumper cables. We again set the baud-rate to 9600. The reason for using the OTII for measurement was that it tracks the UART data sent through the accompanying software, such that energy measurements can be automatically linked to the modem serial communications. Although it has been confirmed by users representing both Exploratory Engineering and Qoitech (the producers of the EE-NBIOT-xx and the OTII, respectively) that the setup is configured correctly [nbi19], we were unable to send AT-Commands to the modem through OTIIs interface. We conducted an experiment to see whether the OTII was sending any UART data at all. First, we connected the TX on the OTII to the RX on the OTII and verified that the OTII was able to communicate with itself. This was successful. Then the experiment consisted of connecting the OTII to a COM-port on a laptop using a TTL-to-USB cable. CoolTerm was used to monitor the serial communications as it was used successfully in experiment 5.1 and the author is familiar with the software. This experiment resulted with the conclusion that the OTII was unable to communicate with the COM-port. Because the only parameters that can be adjusted are the baud-rate for communication and the jumper cables TX/RX connections, and the OTII is unable to communicate with the COM-port, we assume that the OTII is unable to issue serial communications. Qoitech has been notified and they are looking into it. In the meantime another setup is required.

5.1.2 Experiment 1.2: Using OTII and CoolTerm

Experiment goal: Measuring transmissions with the EE-NBIOT-02

Hypothesis: The OTII Arc and CoolTerm combination works

Method used: Black box method, OTII Arc and Coolterm

Because of the results from Experiment 5.1.1 another setup was needed. Although the OTII is unable to send UART data, it is very accurate for energy measurements, being able to detect power in the order of picoampere (10^{-12}A). It is also able to power devices, with a precision of ± 1mV. We will, with this in mind, use the OTII

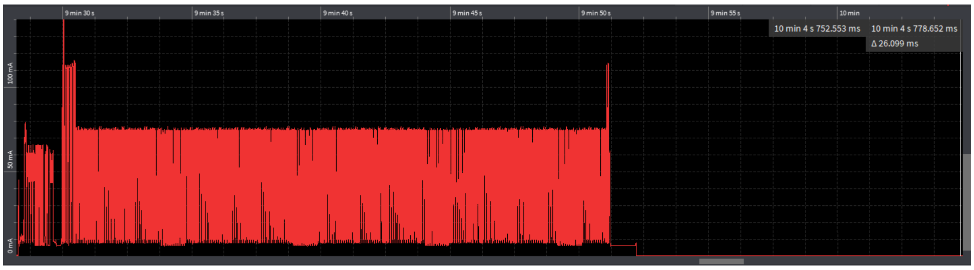
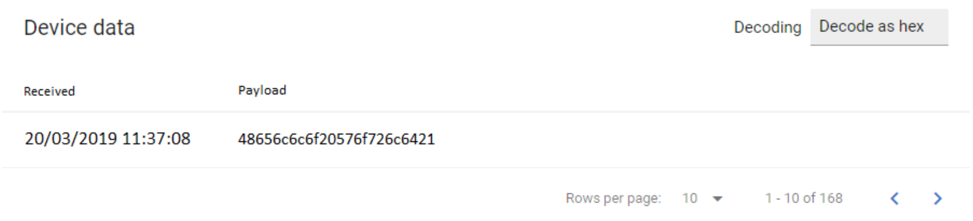


Figure 5.1: Energy measurements from Experiment 5.1.2. Each black cell is 1.25 seconds. Notice the 20 second period with high energy consumption (70-80mA) after the transmission.


Device data

Decoding Decode as hex

Received	Payload
20/03/2019 11:37:08	48656c6c6f20576f726c6421

Rows per page: 10 1 - 10 of 168

Figure 5.2: Verifying that the packet transmitted in Experiment 1.2 was received. The hex code translates to “Hello, World!”.

to power and measure the EE-NBIOT-02, similar to the experiments in Chapter 4. For serial communication, we will use the TTL-232R-3V3 cable with jumper cables and CoolTerm as the serial communication terminal. Here we configure the modem with the following steps:

- Configure the modem: AT+CFUN=1, AT+COPS=0
- Open a socket: AT+NSOCR="DGRAM",17,30000,1
- Send data: AT+NSOST=0,"172.16.15.14" ,1234,11,"4e7973676a65727269673f"

See Figure 5.1 for results. The Figure shows an initial energy consumption block lasting around 1.25 seconds with an average current of approximately 55mA, followed by a short period of about 0.5 seconds with a lower power consumption (approximately 5mA) and then the larger transmission block. This block has an initial current peak of more than 110mA, lasting for around 0.5 seconds, followed by a 20 second period with an average of 70-80mA. We used the web solution presented in Chapter 3 to verify that the packet was received, this can be seen in Figure 5.2. The setup is successful and will be used for the following experiments.

5.1.3 More Realistic Duty Cycling with PSM

The power measurements when transmitting a data frame as seen in Figure 5.1 show that the power consumption is 70-80mA lasting 20 seconds after the transmission was sent. The same can be seen in Figure 4.5 in Chapter 4. There are many possible explanations as to why this is, the most likely being that 20 seconds is the standard period the modem is configured to wait for a reply after a transmission and since no reply is received it causes the device to wait for the time to expire before entering PSM. This is too long for an IoT application. We will illustrate why by calculating the transmission costs. The total energy consumption of the 20 seconds wait per transmission will be

$$75mA \times \frac{20seconds}{3600\frac{seconds}{hour}} \approx 420\mu Ah \tag{5.1}$$

while the energy consumption of the actual transmission was

$$110mA \times \frac{0.5seconds}{3600\frac{seconds}{hour}} \approx 15,3\mu Ah \tag{5.2}$$

so experimenting further on the configuration would not yield very useful results. To decrease the energy consumption after the transmission, we introduce PSM.

PSM can be configured on the SARA-N2xx modems with the AT+CPSMS command. It looks like this: AT+CPSMS=<mode>,<TAU_ONE>,<TAU_TWO>, where TAU_ONE is the requested periodic TAU (T3412) and TAU_TWO is the requested active timer (T3324) which are both discussed in Chapter 2, when PSM is introduced. Configuring PSM with these timers does not make the device enter PSM after a transmission, this has to be explicitly stated in the transmission AT-command. Previously we have been using the AT+NSOST command, which does not utilize PSM settings, it ignores them [uBl18]. The AT-commands we will be using instead is AT+NSOSTF which includes a flag to specify when the modem should release connection. The three options are (1) after the transmission, (2) after next receive packet or (3) default – 20 seconds. First, we want to make sure the configured PSM works as expected. Note: we do not configure eDRX as we are only focusing on transmission.

Configuring PSM

From our discussions on the T3412 and T3324 timers in Section 2.3.6 we know that T3412 should be set to a value as high as application will allow as this is how often the device will wake up from PSM and listen for updates from the network. Additionally, because we know that the T3324 timer decides how long to stay in

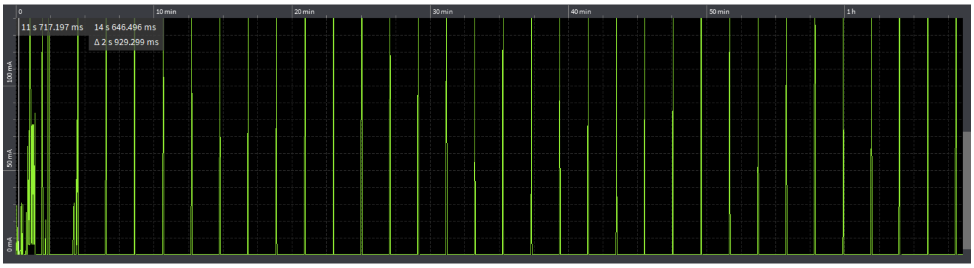
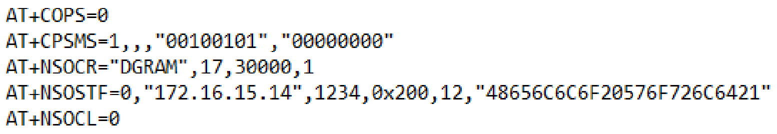


Figure 5.3: PSM configured with T3412 = 2 minutes and T3324 = 0 seconds. notice the power spikes every two minutes.



```
AT+COPS=0
AT+CPSMS=1,,,"00100101","00000000"
AT+NSOCR="DGRAM",17,30000,1
AT+NSOSTF=0,"172.16.15.14",1234,0x200,12,"48656C6C6F20576F726C6421"
AT+NSOCL=0
```

Figure 5.4: The AT-commands used for Experiment

active mode after transmitting and before going into PSM, we want to set the value as low as possible. With this in mind, we stray from the logical implementation to one that can easily be verified as working correctly. We configure PSM with the value "10100010","00000000", which is T3412 as 2 minutes and T3324 as zero seconds. With this implementation we expect to see power spikes in the energy measurements every two minutes, which we do as can be seen in Figure 5.3.

As we were able to configure PSM successfully, we now move on to reducing the time spent in connected mode after a transmission by using the AT+NSOSTF command. We use the flag that indicates disconnection as soon as possible after transmission (0x200). The full list of AT-commands we use for this experiment can be seen in Figure 5.4. The energy measurements can be seen in Figure 5.5. The duty cycling is now much more efficient, with 2, 3-2, 4 seconds in connected mode compared to 20-21 seconds. We do however note that the return to PSM after transmitting is not instantaneous even with the strictest rules for PSM and transmissions.

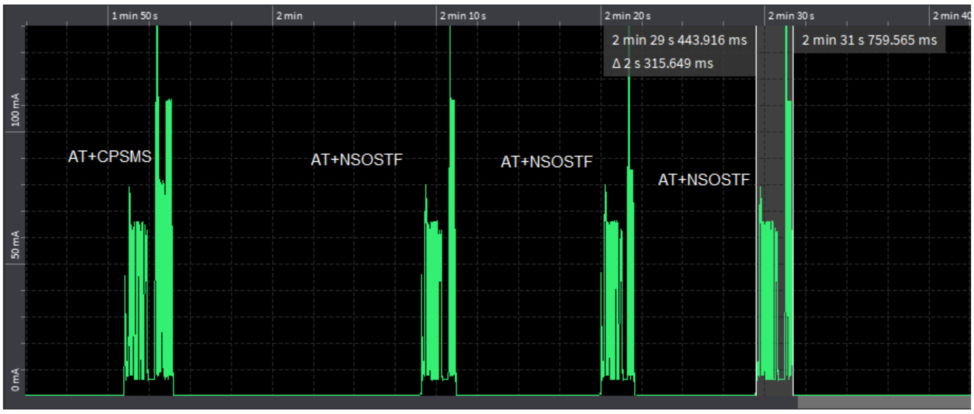


Figure 5.5: Transmissions with PSM configured and used with AT+NSOSTF. The gray area is selected, the transmission takes 2 seconds and around 320 milliseconds total.

5.1.4 Experiment 1.3: SARA N211 Energy Consumption in PSM

Experiment goal: Obtaining an average value for PSM energy consumption

Hypothesis: The power consumption should be $\approx 15\mu W$ in PSM

Method used: Average calculated on ten samples of one minute.

In Section 2.5 when describing the 3GPP calculations for NB-IoT device battery life-time, we saw that 3GPP uses $15\mu W$ as an assumption for PSM power. Using ten samples of measurements of the energy consumption in PSM with each measurements over a period of one minute, we reach an average current in PSM of 3,11 μA with a standard deviation of 0,12. The circuit operating voltage is 3.3V, giving us an average power consumption of $3.11\mu A \times 3.3V \approx 10.2\mu W$ which is in the same order of magnitude as the assumption.

Using the same approach with ten samples over the Connected Mode energy consumption we get a current consumption of 49.4mA using the AT-commands listed in Figure 5.4. As this is more than five orders of magnitude more than the consumption in PSM we see the value of spending approximately 90% less time in connected mode. This was with a 12-byte payload – we want a model that describes the energy consumption of any payload size. With this in mind, let us now recreate the payload experiments conducted in Section 4.4.

5.2 Experiment 2: Payload Experiments

Before we continue the experiment by measuring the energy consumed when transmitting different payload sizes, we want to know what the maximum payload size of a packet is. We do this test even though Telenor have informally stated the maximum is 512 bytes on their network.

5.2.1 Experiment 2.1: Testing the Payload Capacity

Experiment goal: Finding the payload capacity of the system

Hypothesis: The payload capacity is 512 bytes or 2048 bytes

Method used: At-commands to instruct the modem, CoolTerm and OTII Arc

This experiment was done to verify the maximum capacity of the payload. First, we sent 512 bytes of data with the AT+NSOSTF-command. This worked perfectly. We then sent an AT+NSOSTF-command with 513 bytes data to the modem, which resulted in an ERROR. Interestingly, this hints that the modem is the bottleneck, which is confirmed by chapter 12.5.3 in the modem manual [uBl18]. This is less than the NB-IoT protocol supports (2048 bytes) [3GP16b].

5.2.2 Experiment 2.2: Measuring the Energy Consumption of the Payload - Part II

Experiment goal: Modelling single-packet transmission energy

Hypothesis: Sending payloads of 512 bytes is the cheapest per byte

Method used: Black box method, OTII Arc and Coolterm

Now that we can configure PSM and more efficiently duty cycle, we want to see how the energy consumption changes based on the number of bytes transmitted. In this experiment we will vary the payload length and compare the energy measurements. This is a continuation of the experiments conducted in Chapter 4. The goal of the experiment is to model the energy consumption when the transmission data is lower than the network maximum, which in our case is 512 bytes. We gradually increase the payload size and record ten energy measurement samples for each payload size. We then draw the Measurements on a two-dimensional graph and interpolate the points to get a fitting line. The mathematical expression that describes this line will then be a model of the energy consumption when the data transmitted fits in one packet. We do ten samples of 12, 50, 100, 200, 300, 400 and 500 bytes. The results can be seen in Figure 5.6. The mathematical model we get for modelling the energy consumption within a packet is then the following.

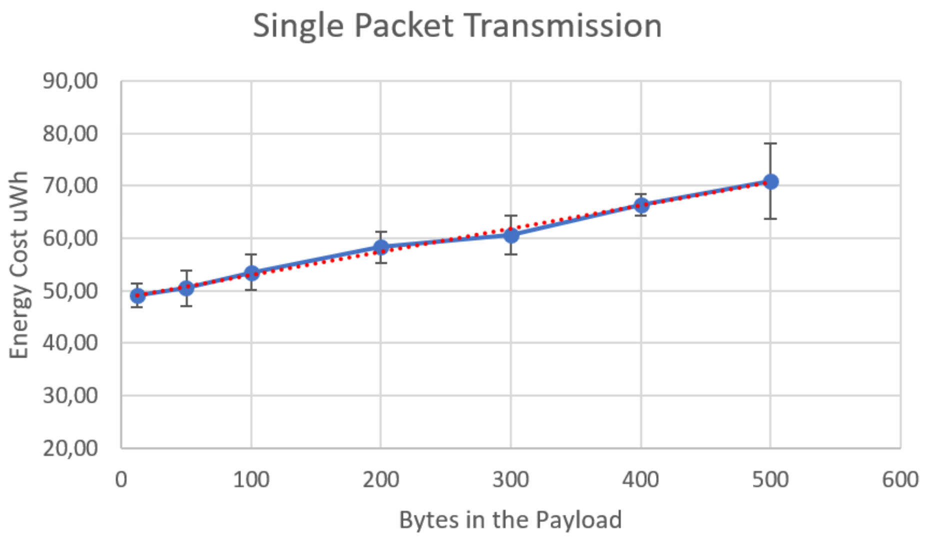


Figure 5.6: The linear interpolation of the energy measurements shown as a dotted red line. Notice that most of the energy cost is fixed – independent of payload size. The 12-byte transmission has a total power consumption of approximately *50mA*, which is more than double the difference between the 500 byte energy consumption and the 12 byte energy consumption of approximately *20mA*.

Energy consumption of a single packet transmission:

$$0,044p_b + 48,70[\mu Wh] \tag{5.3}$$

or translated into ampere-hours

$$0,013p_b + 14,75[\mu Ah] \tag{5.4}$$

where p_b is the number of bytes in the payload. Figure 5.7 uses calculations done with this model to illustrate the energy cost per byte. The graph of the total energy consumption for different payload sizes show that a linear interpolation is a good fit.

The general outcome is that the energy consumption per byte will decrease when the payload increases. This might be a little simplified. Observing the figure, the error bar on the 500-byte payload stands out as being far greater than the others. This makes it reasonable to argue that the most energy efficient payload size has between 400-500 bytes in the payload – the energy results are certainly more predictable for 400 bytes.

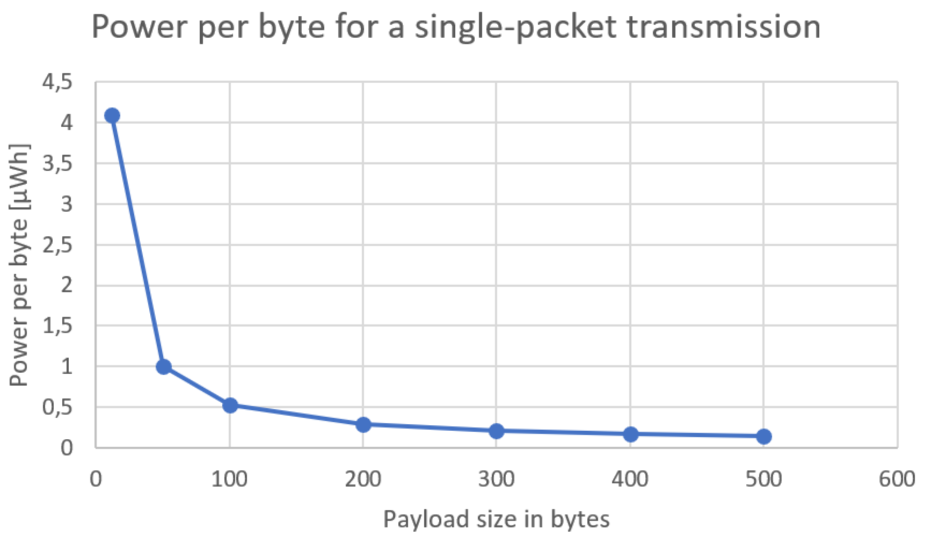


Figure 5.7: Power per byte of a single packet transmission. Shown for 12, 50, 100, 200, 300, 400 and 500 bytes using the model in this chapter to calculate. Notice how sending less than 50 bytes is very expensive. This is because of the fixed cost associated with connecting to the network.

The average transmission energy consumption with 400 bytes observed was 66.28μWh, and the error 2.0μWh. With 500 bytes the average was 70.86μWh and the error 9,6μWh. We look at the cost of transmitting 2000 bytes. With a payload of 400 bytes the energy cost would be a maximum of

$$66.28 + 2.0\mu Wh \times \frac{2000bytes}{400\frac{bytes}{packet}} = 68.28\mu Wh \times 5packets = 341.4\mu Wh \tag{5.5}$$

and an average of

$$66.28\mu Wh \times 5 = 331.4\mu Wh, \tag{5.6}$$

while the maximum energy consumption with a 500-byte payload is

$$70.86 + 9.6\mu Wh \times \frac{2000bytes}{500\frac{bytes}{packet}} = 81.46\mu Wh \times 4packets = 325.8\mu Wh \tag{5.7}$$

and the average is

$$70.86\mu Wh \times 4 = 283.44\mu Wh \tag{5.8}$$

In conclusion, if the application requires a highly *precise* energy consumption then 400 bytes payload is a better choice. If the application stores more than 2000 bytes before transmission and wants the *highest energy efficiency*, then 500-byte payload is the way to go, as even the highest measured energy consumption is more efficient than the average consumption with 400-byte payloads.

Chapter 6

Multiple Packets Experiments

In Chapter 5 the experiments resulted in a mathematical model for the energy consumption of a single packet transmission. Building upon this, we want to investigate the difference between two duty cycles. The first one, which we will call DuCy1, starts in PSM. From PSM the modem then enters Connected mode and transmits one packet, then disconnects and re-enters PSM as soon as possible. This process is repeated for every packet, as seen in Figure 6.1. Note that we can use the formula found in Section 5.2.2 to estimate the energy consumption when using DuCy1 if we know the amount of packets and how many bytes will be in the payload of each packet.

The other way of doing duty cycling, a duty cycle which we will refer to as DuCy2, is a multi-packet transmission solution as shown in Figure 6.2. We start in PSM and accumulate data until (1) the application requires we transmit because of real time requirements or other factors or (2) it is energy efficient to transmit at this moment. When the application decides it is time, the modem enters connected mode. While in connected mode, all the accumulated data is sent, even though it is more than one packet.

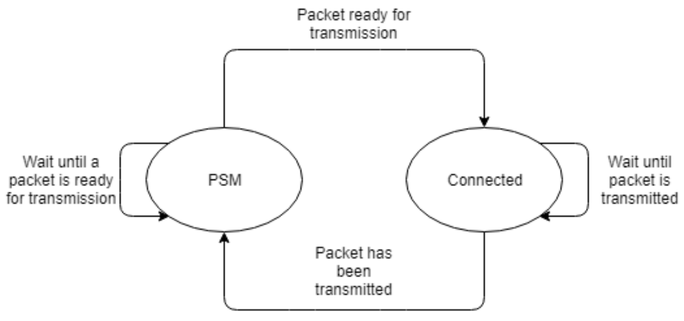

Packet ready for transmission
Wait until a packet is ready for transmission
PSM
Connected
Wait until packet is transmitted
Packet has been transmitted

Figure 6.1: DuCy1 illustrated.

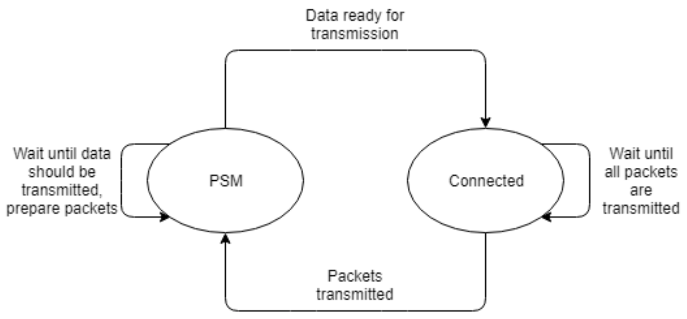

Data ready for transmission
Wait until data should be transmitted, prepare packets
PSM
Connected
Wait until all packets are transmitted
Packets transmitted

Figure 6.2: DuCy2 illustrated.

We conduct this experiment with the hypothesis that DuCy2 will be more energy efficient than DuCy1. This is based on two points from the theory. First, connecting to the network results in a power consumption that is now only done once per batch of packets instead of once per packet as it was done in DuCy1. Secondly, connecting to the network is not instant. This means that after the initial power consumption from the modem entering Connected mode, a non-negligible time is spent in the mode before transmission occurs. We believe that through these experiments we will be able to isolate the radio transmission part of the energy measurements. We would then be able to compute how much energy is consumed by other tasks than transmitting the packet(s), which we hypothesize is consumed before the transmission, as the disconnection and entering PSM should be a small fraction of the total time spent in Connected mode.

Because we already have the measurements from DuCy1, the following experiments will be using DuCy2. The experiments are dependent on the modem being able to send multiple packets while in Connected mode and then disconnecting as quickly as possible. We have learned from previous experiments that this can be done two different ways assuming PSM is configured as efficiently as possible. The two AT-commands which have been used in the experiments in Chapter 4 and 5 differ in one way. AT+NSOSTF attempts to enter PSM as soon as the transmission is completed, while AT+NSOST waits 20 seconds in Connected mode. This means that we can use AT+NSOSTF for all the packets and be careful to not disconnect between the transmissions as this would mean we would reconnect using excess energy. This is possible because even though the modem will try to enter PSM as quickly as possible, the transmission itself takes more than 2 seconds. If another command is issued within this time, the PSM entering is cancelled and a new transmission executed. This would repeat until the last packet which after which the entering of PSM would not be interrupted.

Another approach would be to use AT+NSOST for all packets except the last one in the batch. This would guarantee that the packets would be sent before disconnecting because the 20 second timer should be more than enough. We know from Experiment Three in Section 4.4 that if an attempt is made to send another packet before the 20 seconds are done it will be successful and reset the timer. Another possible benefit from using the AT-Command combination is that AT+NSOST is a shorter command, which will make it ever so slightly less energy consuming to read/execute. The energy saved would multiply with the number of packets sent while connected with AT+NSOST instead of AT+NSOSTF. However, we hypothesize that the energy saved will be very small, less than 1% of the transmission energy cost and it might even be negligible. This is because the AT+NSOSTF only has one more parameter than AT+NSOST which is a 5 bytes flag. This gives us the intuition that if the energy saved by using the combination of AT-commands as opposed to only AT+NSOSTF is worth mentioning, then it would only be so for very small payload sizes – which would be very inefficient and thus not relevant. With all this in mind, we introduce the first experiment.

6.1 Experiment I: Understanding the Energy Consumed In Connected Mode

This collection of experiments in done to understand the energy chart of a transmission. By varying the number of packets in a transmission batch we should be able to see which parts of the energy measurements are unaffected by this change and thus a fixed energy cost. First, we need to verify that we are in fact able to send multiple packets without disconnecting and reconnecting in-between the transmissions. We are unable to get the Arduino MKR NB-1500 working. This is unfortunate as sending several commands quickly one after another is easier and more convenient when implemented with a script instead of manually entering AT-commands. This is even more clear when the number of packets in the batch increases.

We will refer to the person entering the AT-commands manually “the user”. The user in in charge of following DuCy2 with only the AT+NSOSTF commands. This approach is chosen even though using the combination of AT+NSOST and AT+NSOSTF has a clear advantage when it comes to eliminating the possibility of human error. The advantage stems from the nature of the AT+NSOST commands, being a 20 second window for the user to enter the next command while in Connected Mode. Waiting a couple of seconds between each AT-Command, and thus each transmission, would also lead to a more easily readable energy consumption chart. This would make it easier to verify that the transmission was in fact executed. However, this is not a good way to verify a transmission. Instead we will use the web solution by Telenor which was introduced in Subsection 3.2.5 as well as the

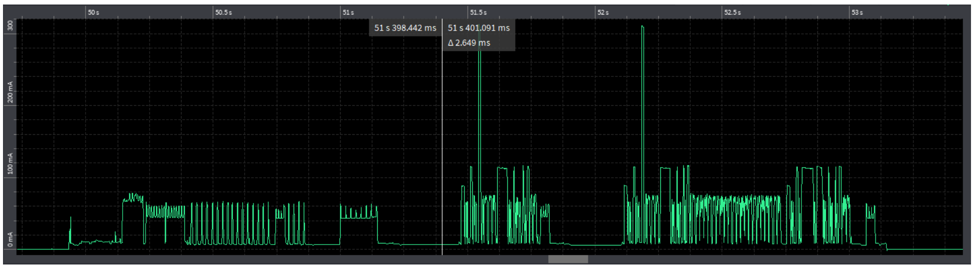


Figure 6.3: The energy chart as seen when five transmissions were executed while the modem was in Connected Mode for the entire duration.

OTII software. The user will need to manually enter AT-commands fast enough that PSM is not entered before the batch is sent. We now conduct the first part of this experiment, which is answering the following question: Can we verify that we are sending multiple packets while in Connected Mode?

6.1.1 Can we Verify Multi-packet Transmissions?

Experiment goal: Verifying a multi-packet transmission scheme

Hypothesis: It is possible to verify that multiple packets are sent without disconnecting in-between transmissions

Method used: Black box method. OTII software and web application by Telenor for seeing the received data. Coolterm and EE-NBIOT-02 for transmissions

The experiment is using the exact same setup as introduced in Section 5.1 with the PSM configurations added. We are using the AT-commands as seen in Figure 5.4, but where the figure only shows one instance of the AT+NSOSTF, we use five in a row. The user enters the commands as quickly as possible while watching the energy chart. The process was repeated ten times, and the chart showed the same energy behaviour, which is shown in Figure 6.3. We can see from the energy measurements that the modem was in Connected modem for the entire transmission batch. However, we can say nothing about how many packets were sent during this period from looking at the chart. We know that the energy chart shows the process of connecting to the network and then transmitting five packets. We can hypothesize that the first part of the energy chart up until the white line in Figure 6.3 is the energy consumption chart resulting from entering Connected Mode from PSM. The energy chart will be analysed in more detail in the following experiment (6.1.2).

For now, we want to verify that the user was able to execute five transmissions during the period. Telenors NB-IoT web solution shows the device data which includes timestamps on the different packets that have been received from the device. This can be seen in Figure 6.6. We see that there were five packets received within a short time frame as the most recent packets, which are the packets sent as described in DuCy2. Note that the user is watching both the energy chart and the device data while transmitting, so it is easy to see the connection. Readers might notice that the figure shows that the previous five transmissions were sent right before the multi packet transmission. This was done by the user to verify that the energy chart acted as expected from the results from the experiments conducted in Chapter 5, which it did. We now know that the modem is able to send multiple packets while in Connected Mode before disconnecting to PSM with DuCy2.

6.1.2 Energy Chart Comparisons

Experiment goal: Isolating the energy cost of transmissions

Hypothesis: The transmissions can be identified by looking at the energy chart

Method used: Black box method. OTII Arc, Coolterm and EE-NBIOT-02

Now that we know we can use DuCy2 we move on to the more interesting part of the experiment. We want to understand the energy consumed by the modem when not transmitting packets with our data. The proposed method to do this is to use NB-IoT as a black box and transmit as described in DuCy2, using the AT-commands that specify quick return to PSM. We then vary the packet batch size, starting with one packet and ending with five. The payload size will remain the same throughout this experiment, the 12 bytes string "Hello World!". For each packet batch size, we will collect ten samples so that we can see whether the energy chart differs much or not and avoid choosing an energy chart anomaly for one of the batch sizes for the comparison. This has already been done for packet batch of size one in Chapter 5 and packet batch of size five in Subsection 6.1.1, but we repeat the process for completion sake. The comparison consists of examining the similarities and differences between the energy charts from the different packet batch sizes.

We show the resulting energy charts from the different batch sizes in Figure 6.4. The similarities can be seen clearly when viewing the energy charts in this manner. The energy consumption of the packet batch sizes illustrated is 52.1 μWh, 79,8 μWh, 96,8μWh, 100,3 μWh and 103,0 μWh respectively. Figure 6.5 illustrates the main parts of the energy chart. We divide the energy chart into three parts, (1) connecting – shown in blue, (2) transmitting the first package – shown in red and (3) transmitting the rest of the packets – shown in green.

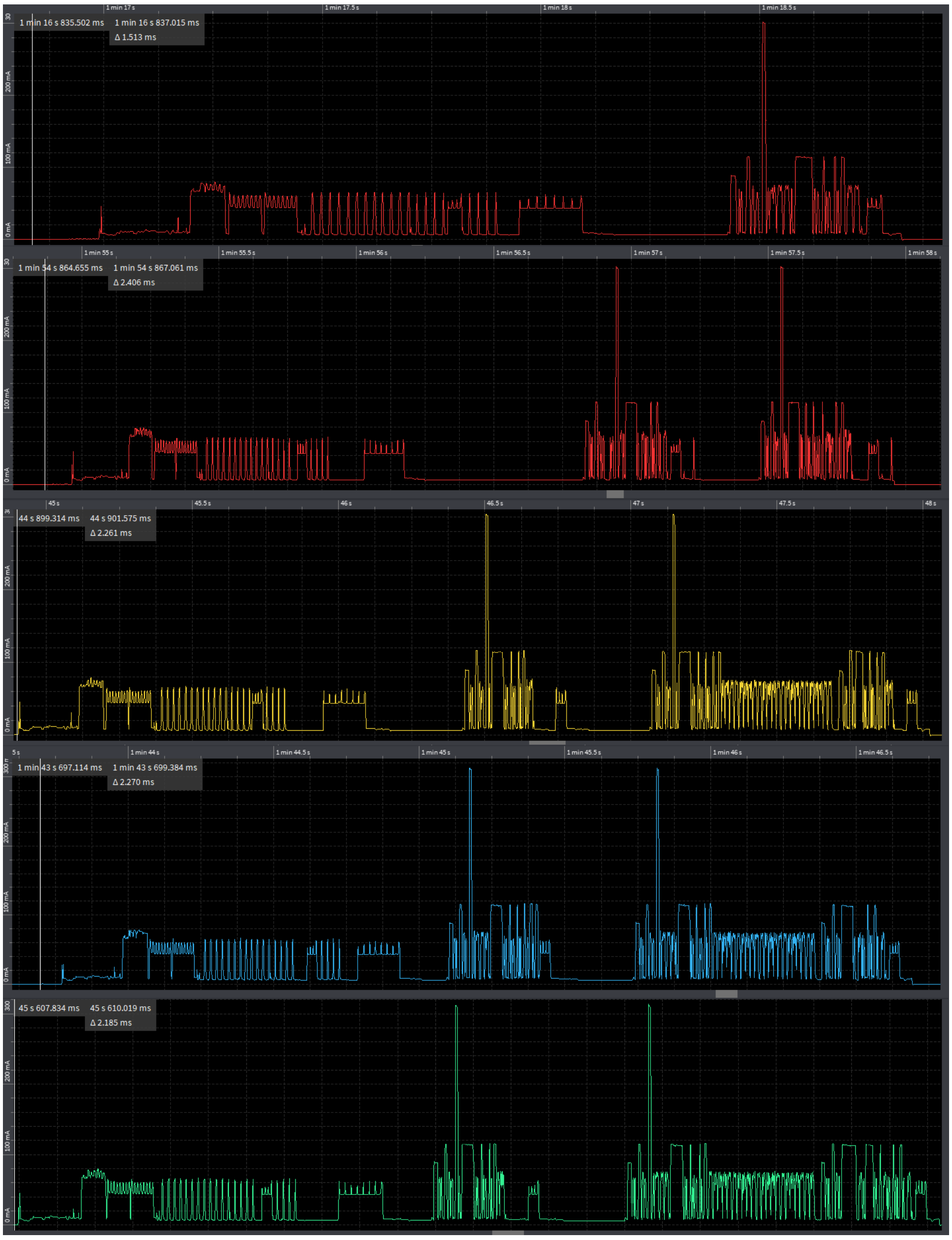


Figure 6.4: Energy charts from the experiment in Subsection 6.1.2. Packet batch size of one (top red) to packet batch size of five (bottom cyan). Note that the charts are scaled differently to fit. The energy consumptions of the charts shown are 52.1 μWh, 79.8 μWh, 96.8μWh, 100.3 μWh and 103.0 μWh respectively.

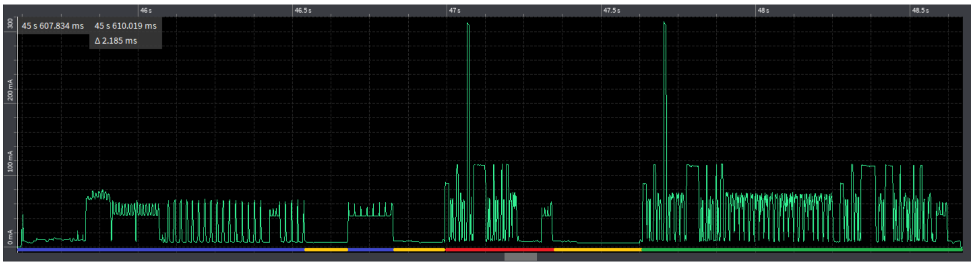
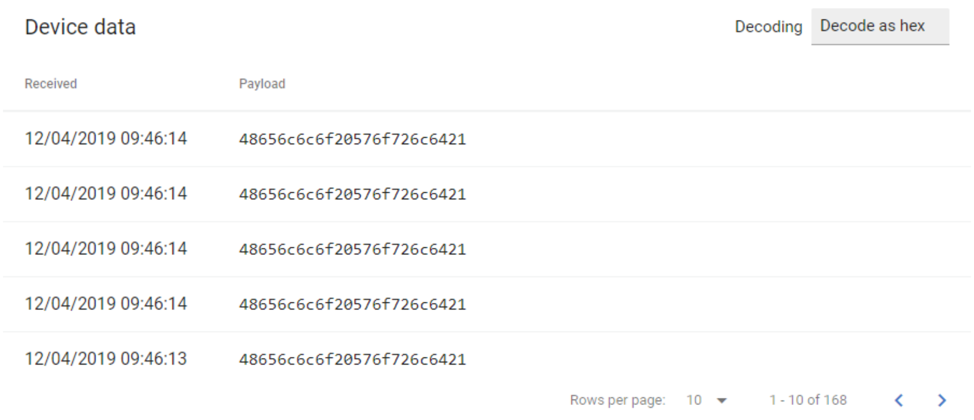


Figure 6.5: Figure 6.3 revisited. The lower part of the chart has been marked with colors to indicate the different main components. Yellow indicates the modem waiting in Connected Mode. In Section 6.2 the energy consumption of the different parts is measured. Three main parts; (1) connecting - shown in blue, (2) transmitting the first package - shown in red and (3) transmitting the rest of the packets - shown in green.



Device data

Decoding Decode as hex

Received	Payload
12/04/2019 09:46:14	48656c6c6f20576f726c6421
12/04/2019 09:46:14	48656c6c6f20576f726c6421
12/04/2019 09:46:14	48656c6c6f20576f726c6421
12/04/2019 09:46:14	48656c6c6f20576f726c6421
12/04/2019 09:46:13	48656c6c6f20576f726c6421

Rows per page: 10 1 - 10 of 168

Figure 6.6: The device data shown on Telenors' NB-IoT web solution after sending five AT+NSOSTF commands to the modem while in Connected mode. We see that there were five packets received within a short time frame, which are the packets sent as described in DuCy2.

Figure 6.5 also shows a fourth part in yellow, which indicates that the modem is waiting in Connected Mode. These parts were consistent for all packet batch sizes. The most notable discovery is that there is a clear pattern in the energy charts showing the first packet in a packet batch being transmitted separately. This repeats for each packet batch size. We see this clearly in Figure 6.4, where the energy charts are lined up. When only one packet is transmitted we see the connection energy consumption which has a small waiting period within, followed by a small waiting part and then the transmission energy consumption (blue, yellow, blue, red in Figure 6.5). For the packet batch with two packets, we see the same pattern, but extended to include two more parts. After the first packet transmission energy consumption there is another short (≈0.5 seconds) waiting period before another transmission energy consumption (blue, yellow, blue, red, yellow, green in Figure 6.5). When the packet batch is increased to three, we see the same exact pattern as with two packets – except the tail of the transmission energy consumption (green) is longer. As the number of packets in the packet batch increases to four and then five, this tail is extended, and the pattern otherwise remains unchanged.

The results of this experiment show a pattern that indicates that the mathematical function describing the energy consumption per packet in the transmission batch might be non-linear. This is based on the observation that less and less energy is added per packet in Figure 6.4. However, the function may be linear from a packet batch of three to higher packet batches with a packet batch of one and two as special cases. The following experiment is conducted to derive this function.

6.2 Experiment II: Modelling the Energy Consumption of a Transmission Batch

This collection of experiments is conducted to understand how the energy consumption is affected by the number of packets in the packet batch of the transmission. In Section 6.1.2 it was discovered that indeed a big part of the energy cost of a transmission is a fixed cost not affected by the number of packets transmitted while in connected mode. Furthermore, we saw that the first packet in a multi-packet batch has an easily isolated energy consumption and the rest of the packets transmissions energy consumption are indistinguishable from one another. This might mean that the first packet is the most expensive packet – and the packet batch with two packets might be a special case energy-wise as the new transmission energy consumption part is included from two packets and up. This would mean that the graph of *the total energy consumption of a packet batch* is linear from three packets and up. Because of the fixed energy costs, it is hypothesized that the energy cost per packet decreases when the number of packets is increased. We start by looking at multi-packet batches of one, two, three, four and five packets.

6.2.1 Transmission Batches up to Five Packets

Experiment goal: Modelling energy per packet in a multi-packet transmission

Hypothesis: The energy per packet relation is non-linear or linear from three packets and up

Method used: Black box testing. EE-NBIOT-02, OTII Arc and Coolterm

As we want to see how the energy consumption is increased when increasing the number of packets in the packet batch, the payload size should not matter. For this experiment we will use a packet payload size of 12 bytes. If applicable based on the findings, the experiment will be extended to include different sizes to see if and how the energy per packet in a multi-packet scheme is affected by payload size. We start by using a packet batch size of one packet. This is the same exact scenario as in the single packet experiment. Then we increase the number by one packet until the batch consists of five. The user is careful to verify with both the energy charts and the web solution from Telenor that the modem stays in connected mode for all the transmissions and that all packets are received. The results are shown in the graphs in Figure 6.7. The results are as expected from the examinations of the energy charts in Section 6.1.2.

We will keep the hypothesis for now, as it has been strengthened by this experiment. To investigate further, we start by testing the possibility of a linear relation beginning when increasing packets from three and up.

6.2.2 Extending the Scope of the Experiment with More Packet Batch Sizes

Experiment goal: Creating a model for multi-packet transmissions

Hypothesis: The total energy consumption of a packet batch increases linearly from three packets and up

Method used: Black box testing. EE-NBIOT-02, OTII Arc and Coolterm

In the previous experiment, we looked at packet batch sizes of up to five. we will extend the scope to include batches with up to eight transmissions to have more data to analyse for the packet batch energy relation function.

Even though the user had the possibility to verify that the AT-commands commands instructed the modem to transmit data correctly, it was a tedious task. Now, with even bigger packet batch sizes, the user needs to be even more careful when using CoolTerm so that the transmission scheme is correctly executed. A Visual Basic

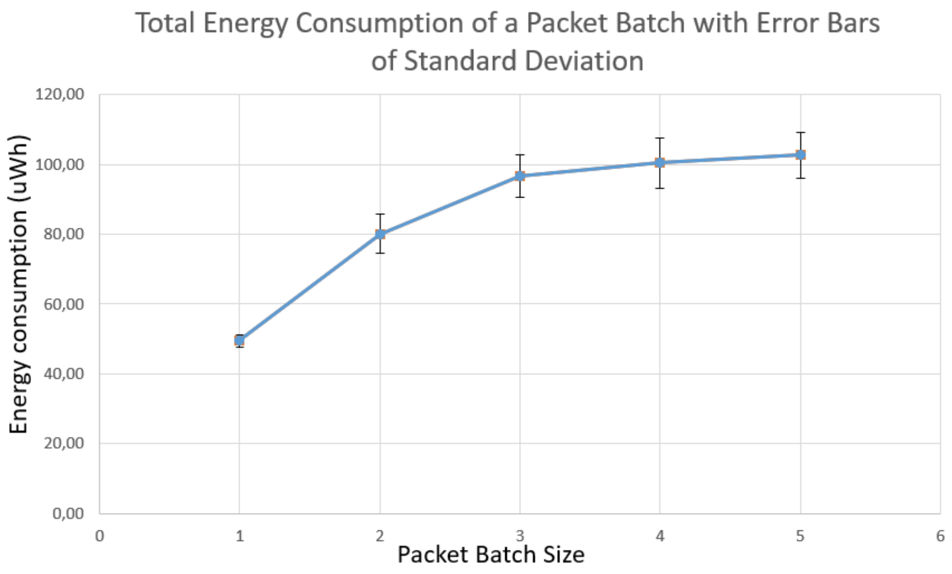


Figure 6.7: Plotting Packet Batch Size to Total Energy Consumption, from the measurements done in Section 6.2.1. Full data shown in appendix.

Packets in the Batch	Connecting[AVG]	First pkt[AVG]	Batch[AVG]	Total Energy[AVG]	Energy per Packet
1	29,6	18,6		51,8	51,8
2	30	18,1	26,7	80,1	40,0
3	29,6	18,4	43,7	96,6	32,2
4	29,5	18	48,3	100,4	25,1
5	29,9	17,7	52,2	102,7	20,5

Figure 6.8: The resulting measurement averages used to graph Figure 6.7. All packets have a payload of 12 bytes, and all measurements are in μWh. Note how the energy per packet decreases with increasing batch size.

Scripting Edition script (VBScript) [KS18] was created to automatically send AT-commands through CoolTerm. The script is an auto-typer where the AT-command, number of packets and time between packets can be specified:

```
set shell = createobject("wscript.shell")
stratcommand = inputbox("AT-command:")
strnumpkts = inputbox ("PacketNumber:")
strtime = inputbox ("Delay(miliseconds):")

if not isnumeric(strnumpkts) then
error=msgbox("PacketNumber!=number")
wscript.quit
end if

msgbox "Enter"
wscript.sleep(1000)

for i=1 to strnumpkts
shell.sendkeys(stratcommand & "")
Shell.SendKeys "{Enter}"
wscript.sleep(strtime)
next
```

Automating the typing makes the task of transmitting a packet batch much smoother, quicker and consistent. The verification process is the same. Figure 6.9 plots the power consumption of the energy batches with three to five packets from the previous example, with the six to eight from this experiment. We see that a linear approximation fits well.

6.2.3 Repeating the Experiment for Different Payload Sizes

Experiment goal: A general model for multi-packet energy consumption regardless of payload size choice

Hypothesis: The total energy consumption to packet batch size relation is linear from three packets and up

Method used: Black box method. OTII Arc, Coolterm and EE-NBIOT-02

We repeat the process so far in Experiment II for different payload sizes to generate a general model that encapsulates both payload size and packet amount. The measurements show a constant relation. This relation is such that when the energy consumption of a single packet transmission of a certain payload size is known

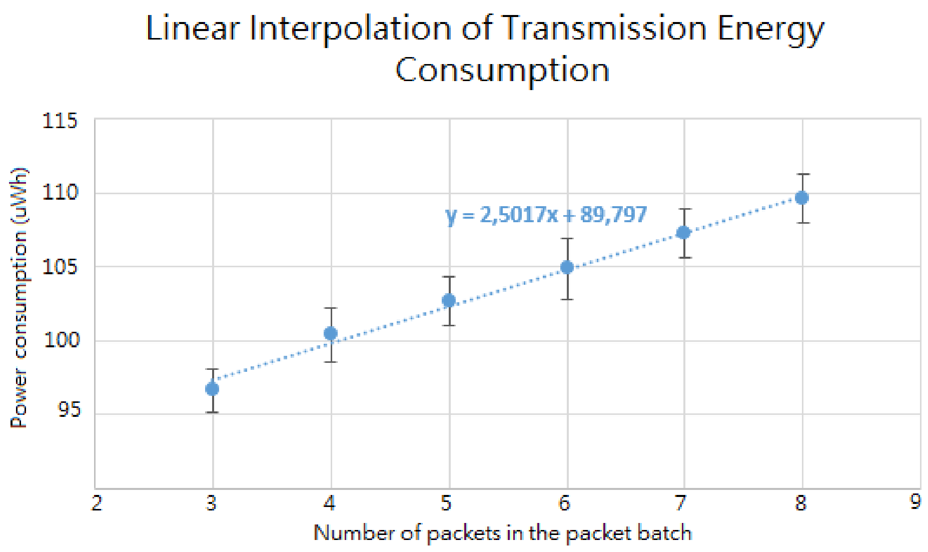


Figure 6.9: Total power consumption plotted against packet batch sizes from three to eight packets. For each plot, the error bar of standard deviation is added.

Table 6.1: Average energy consumption [μWh] of a multi-packet transmissions for different payload sizes

Bytes	12	50	100	200	300	400	500	512
1 pkt	49,44	50,22	53,17	57,91	61,68	66,28	70,72	71,34
2 pkts	80,08	80,99	86,65	98,56	99,47	107,36	114,55	115,55
3 pkts	96,62	98,14	103,91	113,17	120,54	129,53	138,21	139,42
4 pkts	100,40	101,98	107,97	117,02	125,26	135,26	143,61	144,87
5 pkts	102,65	104,27	109,86	120,24	128,06	137,61	146,83	149,55

then the energy consumption of a packet batch can be calculated. The averages energy consumption of the different payload sizes for different packet batch sizes is shown in Table 6.1. The energy cost of a packet batch transmission divided by the cost of a single packet transmission for the same payload size is then computed in Table 6.2. We see that the relation between the energy consumption of the different packet batch sizes stay the same independently of payload sizes. This means that the formula for packet batch energy consumption can use the formula for a single packet transmission.

Table 6.2: The energy cost $[\mu Wh]$ of an packet batch transmission divided by the cost of a single packet transmission for different payload sizes

Bytes	12	50	100	200	300	400	500	512
1 pkt	1	1	1	1	1	1	1	1
2 pkts	1,62	1,61	1,63	1,72	1,61	1,59	1,62	1,66
3 pkts	1,93	1,95	1,95	1,98	1,95	1,95	1,94	1,95
4 pkts	2,03	2,03	2,03	2,02	2,03	2,04	2,03	2,03
5 pkts	2,08	2,09	2,07	2,02	2,08	2,08	2,08	2,10

Based on the findings in this experiment we present the following formulas for a multi-packet transmission scheme. We call the formula for a single packet transmission that was achieved through Chapter 5 for SPT(P_b)[`Unit`], where P_b is the number of bytes in the payload, and `Unit` is the choice between Ah and Wh:

Two packets: $1.65 \times SPT(P_b)[Wh]$

Multiple packets: $1.95 \times SPT(P_b)[Wh] + 2.5[\mu Wh] \times P_n$, where P_n is the number of packets

Chapter 7 Application Examples

In this chapter, we will introduce two relevant LPWAN applications. The applications' data collection strategy will be looked at in detail, including the size of the measurements, how often the measurements are taken and how often the measurements need to be analysed or otherwise used to fulfill the goal of the application. Based on these factors we can tailor a NB-IoT solution for the radio communications.

7.1 LPWAN Application Example: Noise Measurements

A colleague of the author at NTNU, Ida M. V. Bosch, is writing her master's thesis entitled *Efficient Noise Measurement with Energy Constrained IoT Nodes: A Case Study on Working Environment Quality* in which she introduces an IoT setup with the following constraints [Bos19]:

Measurement size: 4 bytes.

Measurement interval: 2 seconds.

Analysis interval: 1 week.

We note first that the analysis interval is very long compared to the measurement interval. This tells us that the application seeks to produce a detailed overview by measuring every two seconds and that the application is not real-time. Because the analysis is happening only once a week, the device could in theory only transmit data every week. There are $60 \times 60 \times 24 \times 7 = 604.800$ seconds in a week, meaning the device would do $604.800 \div 2 = 302.400$ measurements. This would produce $302.400 \times 4 = 1.209.600$ bytes of data. We know from Chapter 6 that the energy consumed per packet in a packet batch transmission duty cycle decreases with the number of packets in the batch. We would send all data in one transmission batch, splitting the data into packets with 512 bytes payload. There is however another constraint on the application; the devices used have a dynamic memory of 16 bytes

[lib]. Because of the memory constraints on the device, transmissions must be done 16 bytes at a time. We can use the formula obtained in Chapter 5 to calculate an energy cost estimation. This gives us an energy consumption of

$0,013 \times 16 + 14,75 \approx 14,96[\mu Ah]$ per transmission.

The application has $1.209.600 \div 16 = 75.600$ weekly transmissions with the constraints. The battery of the device has a capacity of 6,600mAh [lib].

We get a total of $75.600 \times 14,96 = 1.130.976[\mu Ah] \approx 1.131[mAh]$ for a weeks' worth of transmissions when transmitting every 16 bytes (8 seconds). **This means the battery would last less than six weeks.**

The optimal solution would be to store and send all the data at the end of every week. Splitting the data into packets with 512 bytes each, we get 2363 packets accumulated over a week. Using the formula for multipacket transmission we get from Chapter 6, we get

$1,95 \times (0,013[\mu Ah] \times 512[bytes] + 14,75[\mu Ah]) = 41,74[\mu Ah]$

energy consumption for single-packet, giving

$41,74[\mu Ah] + \frac{25}{33}[\mu Ah] \times 2363[pkts] = 1831,89[\mu Ah] \approx 1,83[mAh]$

total energy consumption for the packet batch for the full weeks accumulated data. This is three orders of magnitude cheaper. The energy consumed while in PSM for a week will be

$3,11 \times 10^{-6}[\mu A] \times 24 \times 7[h] \approx 522[\mu Ah]$.

The total energy consumption per week is then $\approx$ 2,35 mAh with the optimal NB-IoT solution. **The battery can now last for more than 54 years.**

As a general note, we know that transmitting less often and in bigger packet batches will be more energy efficient. The example shows that by adding a device memory with capacity of more than 1.209Kb would allow an NB-IoT configuration that is three orders of magnitude more efficient and allows 54 years longer battery lifetime. This is much longer than the 10 years promised by 3GPP – and in practice one would buy a cheaper and lower capacity battery.

7.2 Predicting the Energy Consumption: Smart Metering

For this example, we will use a hypothetical application. We imagine a company called "CleverMeter" which wants to provide an IoT based smart meter solution that is battery powered so there is no need to connect the devices to a power grid. The underlying thought process is that the meters will be easy to deploy and assemble, and possible to install virtually anywhere. The company has done many studies on the usage of meters to validate the business choice. Through careful statistics and mathematical analysis of the systems properties, the following application constraints are found/decided by the stakeholders (all the following application specific numbers and prices are made up to illustrate the usefulness of the model):

Measurement size: 40 bytes.

Measurement interval: 24 seconds.

Analysis interval: 1 day.

Battery lifetime: 5 years.

Given these constraints, they want to choose the most cost-efficient combination of memory and battery to keep the overall cost of the product as low as possible. They know that the electrical components other than the modem draw an average current of $0.02mA$. For the modem they have chosen a uBlox-N211 for NB-IoT. The following hardware is considered:

Battery size	Price
1500mAh	5
2200mAh	9
Memory size	Price
64kB	4
512kB	12

If the other components average 0.02mA it means they consume 0,48mAh/day. The modem will be in PSM for the 24-hour period except for the several seconds spent transmitting, the cost of PSM is the approximated by $0,00311 \times 24 \approx 0,077mA/day$. Together, $0,48 + 0,077 \approx 0,55mA/day$. This is a fixed cost, present regardless of the NB-IoT configuration.

Let us first calculate the total data accumulated throughout a day. We calculate that

$$\frac{40bytes}{24seconds} = 100bytes/min = 144.000bytes/day. \tag{7.1}$$

This means that the 512kB memory is needed to support a days' worth of measurements. The total of 144kB will fit in 282 packets with 512 bytes each. The energy consumption of the NB-IoT transmission will be

$$41,74\mu Ah + \frac{25}{33}\mu Ah \times 282 packets = 255,37\mu Ah. \tag{7.2}$$

With the addition of the fixed energy cost, we are looking at $805\mu Ah/day$. This is a total of

$$0,805\mu Ah/day \times 365,25 days \times 5 years \approx 1470 mAh \tag{7.3}$$

energy consumption over the 5-year period, meaning the cheapest battery can be used. Total cost of this approach is 12 + 5 = **17 per device**.

Another approach is to divide the total data gathered throughout a day into parts that fit into the 64kB memory and transmitting several times during a day. This can be done by splitting the data into three separate packet batches and transmitting three times every day – every time the batch is filled with data. 144kB/3 = 48kB < 64kB – meaning we can split the gathered data into three daily transmissions. The 48kB fits in 94 packets with payload size of 512 bytes. The energy consumption of the NB-IoT transmissions will be

$$(41,74\mu Ah + \frac{25}{33}\mu Ah \times 94 packets) \times 3 transmissions = 338,8\mu Ah. \tag{7.4}$$

With the addition of the fixed energy cost, we are looking at $864\mu Ah/day$. This is a total of

$$0,864\mu Ah/day \times 365,25 days \times 5 years \approx 1577 mAh \tag{7.5}$$

energy consumption over the 5-year period. The more expensive battery is needed. The total cost of this approach is 4 + 9 = **13 per device**.

The example highlights how the model can be used to create the most cost-optimal combination of storage unit and battery. We see that the most energy efficient route is not always the most cost efficient. In fact, the example shows how a 30% cheaper solution was possible within the applications constraints by using a less energy efficient approach.

Chapter 8
Discussion and Conclusion

In this chapter, we will describe the significance of the findings with regards to what is already known about the energy consumption of NB-IoT. This will be done through a discussion of the findings for each research question. The discussion will include important knowledge and parameters for anyone looking to design an application using NB-IoT with the best possible configurations for energy efficiency.

8.1 Knowledge Questions

The first research goal formulated as a research question – "What are the power modes for Narrow-Band IoT devices?" – is to answer a theoretical knowledge question. It was clear that the answer would be found in the 3GPP specifications as the standardization would need to answer the question to be a working standard. Even though NB-IoT is a newly emerged technology it seems that the different power modes are general knowledge already – with most articles looking at NB-IoT mentioning them. Some literature confuses the power states with the power modes and use the terms indistinguishably.

The second research question – "What are the power modes for Narrow-Band IoT devices?" – is also a knowledge question. It became clear that the power mode configurations were done by the modem through negotiating with the LTE base station by setting different timers. We never learned what exactly the modem sends to the base station to set the timers, but we learned that we could specify AT-commands to make the modem configure the power modes correctly. The commands differ slightly between modems, but they should be thoroughly documented. When designing an Internet of Things application using NB-IoT, the developers and software architects should be aware of the relevant AT-commands supported by the modem on the device they are using and how to configure them to utilize Power Save Mode (PSM) and extended Discontinuous Reception (eDRX) efficiently. Without explicitly verifying that the setup is as efficient as possible by understanding the properties of the modem commands, the energy consumption might be atrocious. We saw that transmitting data with AT+NSOSTF instead of AT+NSOST – a difference of a single letter – saved approximately 20 seconds of high energy consumption after a transmission.

8.2 The Model

The knowledge questions were asked because they are part of the foundation needed to look into research question RQ3. In the process of making the model to predict the energy consumption of an NB-IoT device, there were many parameters and additional complexity – making the task more extensive and complicated. The whole process is entirely hardware dependant, even when the inner workings of the system are known. The measuring and analysis methods were chosen after the OTII software did not support serial communication – the OTII was chosen for its ability to link energy consumption with modem commands, such that a link between the two would be apparent. Without this possibility, the energy charts were harder to understand initially.

In Chapter 4, we conducted experiments on the Arduino MKR NB-1500. The device turned out to be too complex after the measurements showed an energy consumption three orders of magnitude too high. Further optimizing the device in terms of energy consumption on other components than the modem might have been possible but was avoided as there were other more specific options to work on. This might have been a blessing in disguise, as analyzing energy charts generated from the Arduino would require extensive analysis and careful planning and execution to isolate energy consumption of different processes. This is due to the fact that when using the Arduino, one uploads a sketch which includes all AT-commands and Arduino-specific commands to the device and then run it. This is compared to the EE-board where AT-commands are issued one by one and the user can see in real-time how the energy chart reacts to the specific command.

The break-out board use in Chapter 5 and Chapter 6 made it possible to study the energy consumption of modulation and transmission – and by manually instructing the modem with AT-commands through a desktop application it was easy to see the correlation between the commands the energy chart. Through the use of several applications by the Internet Service Provider providing the NB-IoT network it was possible to verify that the packets were received as expected. These factors laid the foundation for the black box experiments which resulted in the model. The model is here presented in its full form [Wh]:

PSM power consumption: $10,2\mu W\times$ [time in hours]

Single packet (SPT): $0,044p_b + 48,7[\mu Wh]$, where p_b is the number of bytes in the payload.

Two packets: $1.65 \times SPT(P_b)[Wh]$

Multiple packets: $1.95 \times SPT(P_b) + 2.5[\mu Wh] \times P_n$, where P_n is the number of packets

The PSM power consumption observed is very close to the assumption made by 3GPP ($15\mu W$). Through the black box experiments it became apparent that a packet batch of two packets was a special case, requiring its own model. The multi-packet model assumes that the packets are of the same size. The energy consumption in eDRX is not included. This is because there was no feasible way to transmit data *to* the device and then study downlink and reception energy consumption. The model is made for the ublox SARA-N210 modem. Whether the model can be used as an estimator for other modems is reserved for future work.

Further work

- Expanding the modem to include downlink prediction

- Evaluating the correctness of the model when other modems are used

Further work to improve the model includes the examination of downlink data. While the downlink is mostly reserved for system upgrades such as changing the data collection algorithm and synchronization between UE and eNodeB, excluding it will result in inaccuracies which will be more significant the more the downlink channel is used. If an ISP provides a solution for custom downlink messages or a system is designed for the purpose then the analysis of the downlink energy consumption should be considered.

The model is designed specifically for the SARA-N210 modem. However, the modem was mounted on a break-out board. The resulting energy measurements on the modem are because of this very precise. Because the modem is made for NB-IoT it includes the modulation techniques discussed in Chapter 2 (OFDMA and SC-FDMA). The hardware composition of other modems will differ, but the modulation and demodulation tasks will be the same – which would produce similar energy results. Whether the energy measurements differ much in practise would be a great addition to the model.

A few examples of usage of the model were introduced in Chapter 7 to illustrate the vast coverage of use cases where the model is relevant. The first example is an implementation being used by a colleague of the author – it illustrates how the model can be used to examine and evaluate the energy efficiency of an existing implementation. The second application example is illustrating the usefulness of the model when designing an IoT system. We see that the model can be used for a multitude of purposes, and more importantly we see that the model can be used with simple application-level knowledge. Compared to the model by 3GPP where the user would need to know time spent in every power mode and the corresponding energy consumption for each power mode (eleven parameters in total), the model shows a great improvement in usability.

Chapter 9 Appendix

Algorithm 9.1 The initial imports, instances and variables

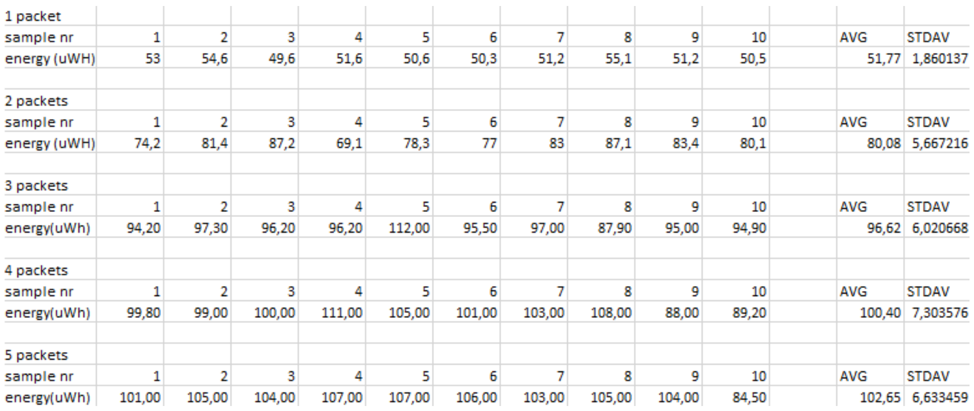
```
#include <MKRNB.h>
#include <Modem.h>
#include <stdio.h>

// PIN Number
const char PINNUMBER[] = "";
// local port to listen for UDP packets
unsigned int MICUdpPort = 1234;

IPAddress MIC_IP(172, 16, 15, 14);

// Initialize the library instance
NBClient client;
GPRS gprs;
NB nbAccess;
NBModem modemTest;
String IMSI = "";
String printOut = "";
byte packetBuffer[512];

// A UDP instance to let us send and receive packets over UDP
NBUDP Udp;
```



1 packet													
sample nr	1	2	3	4	5	6	7	8	9	10		AVG	STDAV
energy (uWH)	53	54,6	49,6	51,6	50,6	50,3	51,2	55,1	51,2	50,5		51,77	1,860137
2 packets													
sample nr	1	2	3	4	5	6	7	8	9	10		AVG	STDAV
energy (uWH)	74,2	81,4	87,2	69,1	78,3	77	83	87,1	83,4	80,1		80,08	5,667216
3 packets													
sample nr	1	2	3	4	5	6	7	8	9	10		AVG	STDAV
energy(uWh)	94,20	97,30	96,20	96,20	112,00	95,50	97,00	87,90	95,00	94,90		96,62	6,020668
4 packets													
sample nr	1	2	3	4	5	6	7	8	9	10		AVG	STDAV
energy(uWh)	99,80	99,00	100,00	111,00	105,00	101,00	103,00	108,00	88,00	89,20		100,40	7,303576
5 packets													
sample nr	1	2	3	4	5	6	7	8	9	10		AVG	STDAV
energy(uWh)	101,00	105,00	104,00	107,00	107,00	106,00	103,00	105,00	104,00	84,50		102,65	6,633459

Figure 9.1: Data gathered in chapter 6.2.1

Algorithm 9.2 The setup()

```
void setup() {
  String response = "";
  response.reserve(100);
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  delay(2000); // Give Serial time to get ready...

  Serial.println("Starting Arduino MKR1500 send to MIC.");
  // Check if modem is ready
  NB_NetworkStatus_t modemStatus;
  for (modemStatus = nbAccess.begin(PINNUMBER); modemStatus!=NB_READY;
  modemStatus = nbAccess.begin(PINNUMBER)) {
    Serial.println("Modem not ready, retrying in 2s...");
    delay(2000);
  }
  // Set radio technology to NB-IoT
  Serial.println("Set RAT to NB-IoT (7 for Cat M1, 8 for NB-IoT): ");
  MODEM.send("AT+URAT=8");
  MODEM.waitForResponse(100, &response);
  Serial.println(response);

  // Set APN to MIC APN
  Serial.println("Set to mda.ee: ");
  MODEM.send("AT+CGDCONT=1,\"IP\",\"mda.ee\"");
  MODEM.waitForResponse(100, &response);
  Serial.println(response);

  // Set operator to Telenor
  Serial.println("Set operator to Telenor: ");
  MODEM.send("AT+COPS=1,2,\"24201\"");
  MODEM.waitForResponse(100, &response);
  Serial.println(response);

  Serial.println("Try to connect...");
  boolean connected = false;
  while (!connected) {
    if (gprs.attachGPRS() == GPRS_READY) {
      Serial.println("Connected!");
      connected = true;
    } else {
      Serial.println("Not connected");
      delay(1000);
    }
  }
  Serial.println("\nSetup socket for connection to MIC...");
  Udp.begin(MICUdpPort);
  // Seed random number generator with noise from pin 0
  randomSeed(analogRead(0));
}
```

Algorithm 9.3 The loop()

```
void loop() {
  int size = 0;

  Serial.print("Send packet to MIC: ");
  sendMICUDPpacket(MIC_IP);
  Serial.println("Check if we have received something..");
  size = receiveMICUDPpacket();
  if (size > 0) {
    Serial.println("Received packet...");
    String bufferString = String((char *) packetBuffer);
    Serial.println("Packet data is: <" + bufferString + ">");
  } else {
    Serial.println("No data received...");
  }
  // Wait 20 seconds before Sending again
  Serial.println("Wait 30s before sending again....");
  delay(20000);
}
int receiveMICUDPpacket() {
  int size = 0;

  size = Udp.parsePacket();
  // Check if size is larger than 0, if yes data is received
  if ( size > 0) {
    Serial.println("packet received");
    // We've received a packet, read the data from it
    Udp.read(packetBuffer, size); // read the packet into the buffer
    return(size);
  }
}
unsigned long sendMICUDPpacket(IPAddress& address) {
  String p1, p2, p3, p4, payload = "";
  String comma = ",";
  float hum, tmp, r = 0.0;

  p1 = "Transmission test";
  hum = 24;
  r = random(0, 9);r = r / 10;
  hum = hum + r;p2 = hum;
  tmp = 20;
  r = random(0, 9);r = r / 10;
  tmp = tmp + r;p3 = tmp;

  payload = p1 + comma + p2 + comma + p3;
  Serial.println("payload is: " + payload);
  Udp.beginPacket(address, MICUdpPort);
  Udp.write(payload.c_str(), payload.length());
  Udp.endPacket();
}
```

Algorithm 9.4 The loop() changed to send two differently sized data chunks. The function sendLongMICUDPpacket added.

```
void loop() {
  int size = 0;
  Serial.print("Send packet to MIC: ");
  sendMICUDPpacket(MIC_IP); \\send the short packet
  delay(10000); // Wait 10 seconds before Sending the Long packet
  Serial.print("Send long packet to MIC: ");
  sendLongMICUDPpacket(MIC_IP);
  delay(20000); \\ wait 20 seconds before restarting the loop
}

unsigned long sendMICUDPpacket(IPAddress& address) {
  String payload = "This is my payload";
  Serial.println("payload is: " + payload);
  Udp.beginPacket(address, MICUdpPort);
  Udp.write(payload.c_str(), payload.length());
  Udp.endPacket();
}
unsigned long sendLongMICUDPpacket(IPAddress& address) {
  \\payloadLong is a string consisting of 296 characters
  String payloadLong = "This is my payload and it is getting bigger,
    in fact it is so long that it might be split into two packets,
    might it not. That is unclear,
    however we know for a fact that this packets payload is much
    bigger in size than the previous one of course,
    and we will see how this affects the trasmission";
  Serial.println("payload is: " + payloadLong);
  Udp.beginPacket(address, MICUdpPort);
  Udp.write(payloadLong.c_str(), payloadLong.length());
  Udp.endPacket();
}
```

Algorithm 9.5 The loop() changed to send small packet then large packet then small packet. Still using the function sendLongMICUDPpacket as introduced in Algorithm 9.4. New function sendMICUDPpacketTwo introduced.

```
void loop() {
  int size = 0;

  Serial.print("Send packet to MIC: ");
  sendMICUDPpacket(MIC_IP);
  Serial.println("Check if we have received something..");
  size = receiveMICUDPpacket();
  if (size > 0) {
    Serial.println("Received packet...");
    String bufferString = String((char *) packetBuffer);
    Serial.println("Packet data is: <" + bufferString + ">");
  } else {
    Serial.println("No data received...");
  }
  // Wait 10 seconds before Sending the Long packet
  Serial.println("Wait 10 second before sending long");
  delay(10000);
  Serial.print("Send long packet to MIC: ");
  sendLongMICUDPpacket(MIC_IP);
  delay(10000);
  Serial.print("Send packet to MIC: ");
  sendMICUDPpacketTwo(MIC_IP);
  delay(60000);
}

unsigned long sendMICUDPpacket(IPAddress& address) {
  String payload = "This is the first payload";
  Serial.println("payload is: " + payload);
  Udp.beginPacket(address, MICUdpPort);
  Udp.write(payload.c_str(), payload.length());
  Udp.endPacket();
}
unsigned long sendMICUDPpacketTwo(IPAddress& address) {
  String payload = "This is the third payload";
  Serial.println("payload is: " + payload);
  Udp.beginPacket(address, MICUdpPort);
  Udp.write(payload.c_str(), payload.length());
  Udp.endPacket();
}
```

Algorithm 9.6 New packets with payloads of 100, 200 and 300 characters defined.

```
unsigned long sendHundredpacket(IPAddress& address) {
  String payload = "FIRST56789b123456789c123456789d123456789e1234
  56789c123456789d123456789e123456789";
  Serial.println("payload is: " + payload);
  Udp.beginPacket(address, MICUdpPort);
  Udp.write(payload.c_str(), payload.length());
  Udp.endPacket();
}
unsigned long sendTwoHundredpacket(IPAddress& address) {
  String payload = "SECOND6789b123456789c123456789d123456789e1234
    56789f123456789g123456789h123456789i123456789j123456789
    a123456789b123456789c123456789d123456789e123456789
    f123456789g123456789h123456789i123456789j123456789";
  Serial.println("payload is: " + payload);
  Udp.beginPacket(address, MICUdpPort);
  Udp.write(payload.c_str(), payload.length());
  Udp.endPacket();
}
unsigned long sendThreeHundredpacket(IPAddress& address) {
  String payloadLong = "THIRD56789b123456789c123456789d123456789e1
  23456789f123456789g123456789h123456789i123456789j1234567
  89a123456789b123456789c123456789d123456789e123456789
    f123456789g123456789h123456789i123456789j123456789
    a123456789b123456789c123456789d123456789e123456789
    f123456789g123456789h123456789i123456789j123456789";
  Serial.println("payload is: " + payloadLong);
  Udp.beginPacket(address, MICUdpPort);
  Udp.write(payloadLong.c_str(), payloadLong.length());
  Udp.endPacket();
}
```

References

[3GP15a] 3GPP. 3gpp tr 45.820 v13.1.0; cellular system support for ultra-low complexity and low throughput internet of things (ciot), chapter 5.4 - “energy consumption evolution methodology”. *3GPP*, 2015.

[3GP15b] 3GPP. 3gpp tr 45.820 v13.1.0; cellular system support for ultra-low complexity and low throughput internet of things (ciot), chapter 7.3.6.4 - “energy consumption evolution”. *3GPP*, 2015.

[3GP15c] 3GPP. 3gpp ts 24.008, timer t3324. *3GPP*, 2015.

[3GP15d] 3GPP. 3gpp ts 24.301;non-access-stratum (nas) protocol for evolved packet system (eps); stage 3 (release 8), chapter 5.3.11. *3GPP*, 2015.

[3GP16a] 3GPP. 3gpp ts 24.008 - mobile radio interface layer 3 specification, values from table 10.5.5.32. *3GPP*, 2016.

[3GP16b] 3GPP. 3gpp ts 36.211 v13.2.0; physical channels and modulation. *3GPP*, 2016.

[3GP18a] 3GPP. 3GPP Release 13. http://www.3gpp.org/DynaReport/GanttChart-Level-2.htm#bm690063, 2018. [Online; accessed 20-October-2018].

[3GP18b] 3GPP. Evolved universal terrestrial radio access (e-utra); nb-iot; technical report for bs and ue radio transmission and reception 6.4.1-6.4.15. *3GPP*, 13(0), 2018.

[3GP18c] 3GPP. Evolved universal terrestrial radio access (e-utra); nb-iot; technical report for bs and ue radio transmission and reception 6.4.18-6.4.20. *3GPP*, 13(0), 2018.

[3GP18d] 3GPP. Evolved universal terrestrial radio access (e-utra); radio resource control (rrc); protocol specification: Ue states and state transitions including inter rat. *3GPP*, 15(4), 2018.

[3GP18e] 3GPP. Evolved universal terrestrial radio access (e-utra); radio resource control (rrc); protocol specification: Ue states and state transitions including inter rat, section 5.2.1.3. *3GPP*, 15(4), 2018.

[AAP$^+$17] P. Andres-Maldonado, P. Ameigeiras, J. Prados-Garzon, J. J. Ramos-Munoz, and J. M. Lopez-Soler. Optimized lte data transmission procedures for iot: Device side energy consumption analysis. In *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 540–545, May 2017.

[BK15] J Brown and J.Y. Khan. A predictive resource allocation algorithm in the lte uplink for event based m2m applications. *IEEE Trans. Mob. Comput.*, 1(1), 2015.

[Bos19] Ida M. V. Bosch. *Efficient Noise Measurement with Energy Constrained IoT NodesA Case Study on Working Environment Quality.* PhD thesis, Norwegian University of Science and Technology (NTNU), 2019.

[ce] core electronics. Qoitech Otii Arc. https://core-electronics.com.au/qoitech-otii-arc.html. [Online; accessed 27-march-2019].

[DAS] ICP DAS. GTM-203-3GWA AT Commands Manual. http://ftp.icpdas.com/pub/cd/usbcd/napdos/3g_modem/gtm-203m-3gwa/manual/gtm-203m-3gwa_atcommands_manual.pdf. [Online; accessed 27-march-2019].

[DZ83] John D. Day and Hubert Zimmermann. The osi reference model. *PROCEEDINGS OF THE IEFJ2, VOL. 71, NO. 12*, pages 1338–1339, 1983.

[elf] elfadistrelec. RND 320-KD3005D - Laboratoriestrømforsyning, 30 V, 5 A Justerbar, RND Lab. https://www.elfadistrelec.no/no/laboratoriestromforsyning-30-justerbar-rnd-lab-rnd-320-kd3005d/p/30061866. [Online; accessed 27-march-2019].

[Enga] Exploratory Engineering. EE-NBIOT-02 v1.0 breakout module. https://shop.exploratory.engineering/collections/nb-iot/products/ee-nbiot-02-v1-0-breakout-module. [Online; accessed 27-march-2019].

[engb] Exploratory engineering. Exploratory Engineering - Conserving battery power. https://docs.nbiot.engineering/tutorials/low-power.html. [Online; accessed 02-jun-2019].

[engc] Exploratory engineering. Exploratory Engineering - We make things that sometimes don't explode. https://exploratory.engineering/. [Online; accessed 30-may-2019].

[Far18] S. Farrell. Low-Power Wide Area Network (LPWAN) Overview. https://datatracker.ietf.org/doc/rfc8376/?include_text=1, 2018. [Online; accessed 23-March-2019].

[fG08] Magdalena Nohrborg for 3GPP. The motivation for LTE. https://www.3gpp.org/technologies/keywords-acronyms/98-lte, 2008. [Online; accessed 19-March-2019].

[GBMP13] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Mrimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *University of Melbourne, Australia*, 1(1):001–016, June 2013.

[GSM17a] GSMA. NB-IoT Deployment Guide to Basic Feature set Requirements. https://www.gsma.com/newsroom/wp-content/uploads/CLP.28v1.0.pdf#page=7&zoom=100,0,233, 2017. [Online; accessed 11-March-2019].

[GSM17b] GSMA. Nb-iot deployment guide to basic feature set requirements version 1.0. *GSM Association*, page 14, 2017.

[Hum91] Pierre A. Humlet. On the bit error rate of lightwave systems with optical amplifiers. *IEE journal of lightwave technology vol 9 no 11*, 9(11):001–002, November 1991.

[Kei16] Robert Keim. Understanding Quadrature Phase Shift Keying (QPSK) Modulation. https://www.allaboutcircuits.com/technical-articles/quadrature-phase-shift-keying-qpsk-modulation/, 2016. [Online; accessed 20-April-2019].

[Kra] Frank Alexander Kramer. TTM4175 - Communication Technology, introduction. https://www.item.ntnu.no/studies/courses/ttm4175/start. [Online; accessed 27-march-2019].

[KS18] John Kennedy and Michael Satran. Microsoft Docs: Using VBScript. https://docs.microsoft.com/en-us/windows/desktop/lwef/using-vbscript, 2018. [Online; accessed 17-mar-2019].

[lib] libelium.com. IoT made easy! http://www.libelium.com/products/plug-sense/technical-overview/. [Online; accessed 15-may-2019].

[LL17] Jinseong Lee and Jaiyong Lee. Prediction-based energy saving mechanism in 3gpp nb-iot networks. *School of Electrical and Electronics Engineering, Yonsei University*, pages 001–004, 2017.

[MBCM17] K. Mekki, Eddy Bjic, Frederic Chaxel, and Fernand Meyer. A comparative study of lpwan technologies for large-scale iot deployment. *Research Centre for Automatic Control of N*, 1(1):001–021, December 2017.

[MKBT19] Abdulmajid Murad, Frank Alexander Kraemer, Kerstin Bach, and Gavin Taylor. Management of energy-harvesting iot nodes using deep reinforcement learning. *13th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2019)*, 2019.

[MKR19] Arduino MKR. ARDUINO MKR NB 1500. https://store.arduino.cc/arduino-mkr-nb-1500, 2019. [Online; accessed 27-march-2019].

[MLG06] Hyung G. Myung, Junsung Lim, and David J. Goodman. Single carrier fdma for uplink wireless transmission. *IEEE Vehicular Technology Magazine, vol. 1, no. 3*, pages 30–38, 2006.

[MW81] Merriam-Webster. ampere, noun. https://www.merriam-webster.com/dictionary/ampere, 1881. [Online; accessed 02-mar-2019].

[MW82a] Merriam-Webster. Ampere-hour, noun. https://www.merriam-webster.com/dictionary/ampere-hour, 1882. [Online; accessed 02-mar-2019].

[MW82b] Merriam-Webster. watt, noun. https://www.merriam-webster.com/dictionary/watt, 1882. [Online; accessed 02-mar-2019].

[MW88] Merriam-Webster. watt, noun. https://www.merriam-webster.com/dictionary/watt-hour, 1888. [Online; accessed 02-mar-2019].

[nbi19] Qoitech Forum Post nbiot. Unable to send AT-commands. https://forum.qoitech.com/t/unable-to-send-at-commands/282/2, 2019. [Online; accessed 08-April-2019].

[ND12] Srinivas Nidhra and Jagruthi Dondeti. Black box and white box testing techniques –a literature review. *International Journal of Embedded Systems and Applications (IJESA) Vol.2, No.2*, pages 33–38, 2012.

[Not] Electronic Notes. OFDM Cyclic Prefix, CP. https://www.electronics-notes.com/articles/radio/multicarrier-modulation/ofdm-cyclic-prefix-cp.php. [Online; accessed 22-April-2019].

[NP00] Richard Van Nee and Ramjee Prasad. *OFDM for Wireless Multimedia Communications.* Artech House Universal Personal Communications, 2000.

[PBM95] Thierry Pollet, Mark Van Badel, and Marc Moeneclaey. Ber sensitivity of ofdm systems to carrier frequency offset and wiener phase noise. *IEEE Transactions on Communications vol 43 no 2/3/4*, 1995.

[RKS17] Usman Raza, Parag Kulkarni, and Mahesh Sooriyabandara. Low power wide area networks: An overview. *IEEE Communications Surveys Tutorials*, 19(2):001–016, June 2017.

[RMZ$^+$16] Rapeepat Ratasuk, Nitin Mangalvedhe, Yanji Zhang, Michel Robert, and JussiPekka Koskinen. Overview of narrowband iot in lte rel-13. *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*, 1(1), 2016.

[RND] RND. USER MANUAL – BEDIENUNGSANLEITUNG RND 320-KD3000 D/P Series. https://www.elfadistrelec.no/Web/Downloads/_m/an/RND%20320_KD3000_mul_man.pdf. [Online; accessed 27-march-2019].

[RS17] Rhode and Schwarz. Let's talk IoT – Insights into NB-IoT in 3GPP Release 14. https://www.rohde-schwarz.com/no/solutions/test-and-measurement/wireless-communication/iot-m2m/iot-m2m-webinars-videos/let-s-talk-iot15-insights-into-nb-iot-in-3gpp-release-14_232250.html, 2017. [Online; accessed 30-March-2019].

[Sau17] Martin Sauter. *From GSM to LTE-Advanced Pro and 5G.* Wiley, 2017.

[Sha18a] ShareTech. eDRX (Extended/Enhanced DRX). http://www.sharetechnote.com/html/Handbook_LTE_eDRX.html, 2018. [Online; accessed 09-March-2019].

[Sha18b] ShareTechNotes. LTE Quick Reference . http://www.sharetechnote.com/html/Handbook_LTE_NB_LTE.html, 2018. [Online; accessed 19-March-2019].

[SJHB87] Henrik V. Sørensen, Douglas L. Jones, Michael T. Heideman, and Sidney Burrus. Real-valued fast fourier transform algorithms. *IEE transactions on acoustic, speech and signal processing*, ASSP-35(6):001–001, June 1987.

[Sli07] Slimane ben Slimane. Reducing the peak-to-average power ratio of ofdm signals through precoding. *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 56, NO. 2*, 2007.

[Spr14] Springer. Sinusoid. https://www.encyclopediaofmath.org/index.php/Sinusoid, 2014. [Online; accessed 22-April-2019].

[SV06] Srikanth S. and Kumaran V. Orthogonal frequency division multiple access: Is it the multiple access system of the future? *Murugesapandian AU-KBC Research center, Anna University, Chennai, India*, pages 002–007, 2006.

[SWH17] Rashmi Sharan Sinha, Yiqiao Wei, and Seung-Hoon Hwang. A survey on lpwa technology: Lora and nb-iot. *ICT express*, 3(1):014–021, March 2017.

[Tec00] Keysight Technologies. Concepts of Orthogonal Frequency Division Multiplexing (OFDM) and 802.11 WLAN. http://rfmw.em.keysight.com/wireless/helpfiles/89600B/WebHelp/Subsystems/wlan-ofdm/content/ofdm_basicprinciplesoverview.htm, 2000. [Online; accessed 22-April-2019].

[Tel] Telenor. IoT på 4G. https://www.telenor.no/bedrift/iot/4g/. [Online; accessed 02-mar-2019].

[Tel19] Telenor. Start IoT login. https://startiot.mic.telenorconnexion.com/login, 2019. [Online; accessed 02-mar-2019].

[TJ10] S. J. Thiruvengadam and Louay M. A. Jalloul. Performance analysis of the 3gpp-lte physical control channels. *EURASIP Journal on Wireless Communications and Networking*, 2010.

[uBl] uBlox. SARA-R4/N4 series. https://www.u-blox.com/en/product/sara-r4n4-series. [Online; accessed 27-march-2019].

[uBl18] uBlox. Sara-n2 modules power-optimized nb-iot modules at commands manual. *uBlox*, 2018.

[ubl19] ublox. Sara-r4/n4 series size-optimized lte cat m1 / nb1 / gprs modules at commands manual. *UBX-17003787 - R12*, 15(4), 2019.

[Ung82] G. Ungerboeck. Channel coding with multilevel/phase signals. *IEEE Transactions on Information Theory*, 1982.

[Wei] Eric W. Weisstein. Convolution. mathworld.wolfram.com/convolution.html. [Online; accessed 22-April-2019].

[Wie14a] Roel J. Wieringa. *The Design Cycle*, pages 27–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[Wie14b] Roel J. Wieringa. *Treatment Validation*, pages 59–69. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[Wie14c] Roel J. Wieringa. *What Is Design Science?*, pages 3–11. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[YAFB10] Chung Him (George) Yuen, Pooyan Amini, and Behrouz Farhang-Boroujeny. Single carrier frequency division multiple acces (sc-fdma) for filter bank multicarrier communication systems. *ECE department, University of Utah, USA*, pages 1–4, 2010.

Johannes Haukland

Modelling the Energy Consumption of NB-IoT Transmissions

NTNU
Norwegian University of
Science and Technology