

# **SimpleLink Expansion**

Bachelor Thesis

Submitted by Herm Lecluse

In fulfilment of the requirements for the degree  
Bachelor of Science  
To be awarded by the  
Fontys University of Applied Sciences

Venray, June 11, 2018

---

## Information Page

Fontys Hogeschool Techniek en Logistiek  
Postbus 141, 5900 AC Venlo

### Bachelor Thesis Project Mid Term Report

Name of student: Herm Lecluse  
Student number: 2534851  
Course: Informatics - Software Engineering  
Period: February 2018 - July 2018

Company name: Phact B.V.  
Address: Keizersveld 83  
Postcode, City: 5803 AP, Venray  
Country: The Netherlands

Company coach: Johan Kleuskens  
Email: johank@phact.nl  
University coach: Langenhuizen, Marco  
Email: m.langenhuizen@fontys.nl

Examinator: Monsieur, Geert  
External domain expert: Wielenga, Geertjan

Non-disclosure agreement: no

# Preface

This thesis has been written in June 2018 at Phact B.V. under the supervision of Johan Kleuskens.

First, I would like to gratefully acknowledge my supervisor Johan Kleuskens, who continuously helped me during this work with his motivating words, his support and his constantly quick reactions to my numerous questions. Second, I am indebted Patrick Scheepers for being my second supervisor.

Furthermore I would like to gratefully acknowledge my teacher Marco Langenhuizen for his support and time during this project.

At last, I would like to thank the colleagues at Phact B.V. , for their understanding, endless patience, and their support.

## Summary

Phact is a one stop shop for software and hardware solutions and secondment. Phact was founded in 2015 with nearly 25 years of experience in the telecommunication branch.

This report will give the reader an insight into the work that has been done by the intern during the planning, research and development phase of this project. This project is about improving the usability of the Texas Instruments CC 1350 chipset.[TICC17] This chipsets operating system is lacking certain functionalities. These functionalities are the possibility to switch deep sleep and to setup a meshing network. The desired product of this project is a software layer that will be created on top of Texas Instruments SimpleLink™ software platform. This software layer will implement these missing functionalities.

The research that was conducted showed the impact of deep sleep on the lifespan of a battery (increased to up to 525x the battery life) on which a device could be powered with. Another result that came from the research is that knowledge was gained on how the nodes communicate with each other.

An Software Requirements Specification was created, this document describes the software system to be developed. Design documents have been made. These documents help the reader to understand the software layer that has been developed. UML diagrams about the system make clear how the chipset of Texas Instruments is used to fulfil the wishes of Phact B.V.

From these design documents, products were developed. These products are programs written in C for the TI CC 1350 chipset. The product requirements set by Phact B.V. are realised in these applications. A chipset can be configured to repeat an incoming message. Another functionality is that the node can be put into deep sleep.

Last but not least, an advice has been given on how to further improve the application layer. This by making use out of a hashtable for a network in which more devices can operate. Another advice has been given to use an extra chipset, this to improve performance of the network.

# Samenvatting

Phact is een one stop shop voor software- en hardwareoplossingen en detachering. Phact startte in 2015 met al bijna 25 jaar ervaring in de telecommunicatie branch op zak.

Dit rapport geeft de lezer een kijk op het werk dat verricht is door de stagiair tijdens de planning-, onderzoek- en ontwikkelfasen. Bovendien gaat it project over het verbeteren van de inzetbaarheid van de Texas Instruments CC 1350 MCU chipset[TICC17] voor Phact. Het operating systeem van deze chipset mist bepaalde functionaliteiten. Deze functionaliteiten zijn, het mogelijk maken van deep sleep en het maken van een mesh netwerk. Dit project gaat over een softwarelaag die over SimpleLink™ van Texas Instruments gebouwd wordt en die deze ontbrekende functionaliteiten wel implementeert.

Uit het onderzoek blijkt dat deep sleep heel veel impact kan hebben op de levensduur van de batterij waar deze chipset mee gevoed kan worden. Wanneer de chip in deep sleep is, gaat de batterij tot 525x langer mee dan wanneer de chip inactief is. Daarnaast is er onderzocht hoe de chipset communiceert met andere nodes in een netwerk.

Het opzetten van een mesh netwerk kan op meerdere manieren, er is gekozen om een zogeheten flooding mesh te implementeren. Hierdoor kunnen nodes via andere nodes communiceren met de gateway in plaats van alleen rechtstreeks. Verder is er een Software Requirements Specification gemaakt dat laat zien welke eisen er aan deze softwarelaag zitten.

Er zijn ontwerpdocumenten gemaakt die de lezer helpen met het begrijpen van de softwarelaag die ontwikkeld is. Door gebruik te maken van UML wordt er geprobeerd om de lezer visueel te laten zien hoe de chipset van Texas Instruments gebruikt wordt om aan de eisen van Phact B.V te voldoen.

Uit deze ontwerp documenten zijn ook producten gekomen. Deze producten zijn programma's die geschreven zijn in C voor de TI CC 1350 chipsets. De functie eisen van Phact B.V. zijn in deze applicaties gerealiseerd. Zo kan een chipset geconfigureerd worden om een bericht te herhalen, of er kan een chipset in 'deep sleep' gezet worden.

Ten slotte is er advies gegeven hoe deze applicatielaag verbeterd kan worden. Dit door gebruik te maken van een hashtable voor een groter netwerk. Ook om een extra chip te gebruiken voor een betere prestatie van het netwerk.

## Statement of Authenticity

I, the undersigned, hereby certify that I have compiled and written the attached document / piece of work and the underlying work without assistance from anyone except the specifically assigned academic supervisors and examiners. This work is solely my own, and I am solely responsible for the content, organization, and making of this document / piece of work.

I hereby acknowledge that I have read the instructions for preparation and submission of documents / pieces of work provided by my course / my academic institution, and I understand that this document / piece of work will not be accepted for evaluation or for the award of academic credits if it is determined that it has not been prepared in compliance with those instructions and this statement of authenticity.

I further certify that I did not commit plagiarism, did neither take over nor paraphrase (digital or printed, translated or original) material (e.g. ideas, data, pieces of text, figures, diagrams, tables, recordings, videos, code, ...) produced by others without correct and complete citation and correct and complete reference of the source(s). I understand that this document / piece of work and the underlying work will not be accepted for evaluation or for the award of academic credits if it is determined that it embodies plagiarism.

Name: Herm Lecluse  
Student Number: 2534851  
Place/Date: Venray, June 11, 2018

A handwritten signature in black ink, appearing to read 'H. Lecluse', with a stylized, overlapping flourish underneath.

Signature:

# Contents

<b>Information page</b>	<b>ii</b>
<b>Preface</b>	<b>iii</b>
<b>Summary</b>	<b>iv</b>
<b>Samenvatting</b>	<b>v</b>
<b>Statement of Authenticity</b>	<b>vi</b>
<b>Contents</b>	<b>vii</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>ix</b>
<b>List of Acronyms</b>	<b>x</b>
<b>Glossary</b>	<b>xii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Project Plan</b>	<b>2</b>
2.1 Context . . . . .	2
2.1.1 Case . . . . .	3
2.2 Lead-up to the assignment . . . . .	3
2.3 Goal . . . . .	3
2.4 Expected results . . . . .	3
2.5 Scope . . . . .	4
2.6 Program requirements . . . . .	4
2.7 Project phases . . . . .	4
2.7.1 Plan . . . . .	4
2.7.2 Research . . . . .	5
2.7.3 Development . . . . .	5
2.8 Deliverables . . . . .	5
2.9 Activities . . . . .	6
2.10 Project management . . . . .	6
2.10.1 Time . . . . .	6
2.10.2 Quality . . . . .	6
2.11 Risk analysis . . . . .	7
2.12 Project Organization . . . . .	7
2.12.1 Stakeholders . . . . .	7
2.12.2 Communication . . . . .	7
2.13 Milestones . . . . .	8

<b>3</b>	<b>Research and results</b>	<b>9</b>
3.0.1	Overall decisions . . . . .	9
3.0.2	Questions . . . . .	9
3.0.3	Methods . . . . .	10
3.1	SimpleLink . . . . .	10
3.1.1	Communication . . . . .	10
3.2	Deep sleep . . . . .	12
3.2.1	Sensor Controller . . . . .	14
3.3	Mesh networking . . . . .	14
3.3.1	Flood mesh . . . . .	15
3.3.2	Sniffer . . . . .	16
3.4	Conclusion . . . . .	17
<b>4</b>	<b>Software Requirement Specification</b>	<b>18</b>
4.1	Functional Requirements . . . . .	18
4.2	4.2 Non-functional requirements . . . . .	18
<b>5</b>	<b>Design</b>	<b>19</b>
5.1	Usecase . . . . .	19
5.2	Sequence diagram . . . . .	20
5.3	State machine diagrams . . . . .	21
<b>6</b>	<b>Development</b>	<b>22</b>
6.1	Meshing . . . . .	22
6.1.1	Initialization . . . . .	22
6.1.2	Receive (RX) . . . . .	23
6.1.3	Handle Message . . . . .	24
6.1.4	Transmit (TX) . . . . .	26
6.2	Deep sleep . . . . .	26
6.2.1	Entering sleep . . . . .	27
6.2.2	Waking up . . . . .	27
<b>7</b>	<b>Conclusion</b>	<b>29</b>
7.1	Discussion . . . . .	29
7.2	Future . . . . .	29
<b>8</b>	<b>Recommendations</b>	<b>30</b>
8.1	Meshing . . . . .	30
8.1.1	Floodsession . . . . .	30
8.1.2	Performance . . . . .	30
8.2	Deep sleep . . . . .	30
<b>Appendix A : Software Requirements Specification</b>		<b>32</b>
<b>Appendix B : Project Plan</b>		<b>36</b>
<b>Appendix C : Midterm Report</b>		<b>44</b>



# List of Figures

2.1	TI CC 1350 Launchpad . . . . .	2
3.1	TI CC 1350 architecture . . . . .	10
3.2	circuit scheme . . . . .	11
3.3	transmission . . . . .	12
3.4	circuit scheme . . . . .	12
3.5	circuit scheme oscilloscope . . . . .	13
3.6	screenshot oscilloscope . . . . .	14
3.7	flooding mesh Amber Wireless . . . . .	15
3.8	screenshot Wireshark . . . . .	16
5.1	sequence diagram send an order via repeater to the gateway . . . . .	20
5.2	state diagram repeater . . . . .	21
5.3	state diagram deep sleep . . . . .	21

# List of Tables

2.1	Table of project activities . . . . .	6
2.2	Table showing the stakeholders of this project . . . . .	7
3.1	Table displaying measured values . . . . .	13

# List of Acronyms

<b>BIOS</b>	Basic Input/Output System
<b>GPIO</b>	General Purpose Input/Output
<b>ICALL</b>	Indirect Call Framework
<b>IDE</b>	integrated development environment
<b>IoT</b>	Internet of Things
<b>I/O</b>	Input/Output
<b>MAC</b>	Medium Access Control
<b>MCU</b>	Microcontroller Unit
<b>OS</b>	Operating System
<b>PAN</b>	Personal Area Network
<b>RF</b>	Radio frequency
<b>RSSI</b>	Received Signal Strength Indication
<b>RX</b>	Receive
<b>SDK</b>	Software Development Kit
<b>SRS</b>	Software Requirements Specification
<b>TI</b>	Texas Instruments
<b>TI-RTOS</b>	Texas Instruments-Real Time Operating System
<b>TX</b>	Transmit
<b>UART</b>	Universal Asynchronous Receiver-Transmitter

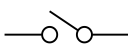
# Glossary

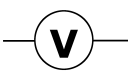
<b>Cooja</b>	Cooja is the Contiki network simulator. Cooja allows large and small networks of Contiki nodes to be simulated.
<b>Contiki</b>	An open source operating system for the Internet of Things.
<b>Deep sleep</b>	State in which a device uses minimal amounts of power (disabling certain components) to save energy.
<b>Gateway</b>	A network node that connects two networks using different protocols together.
<b>Mesh Network</b>	Is a local network topology in which the infrastructure nodes connect directly, dynamically and non-hierarchically to as many other nodes as possible and cooperate with one another to efficiently route data from/to clients.
<b>Node(s)</b>	Devices or data points on a larger network. e.g. personal computer, cell phone, or printer.
<b>Star network</b>	A network with one central host which distributes the messages from the different connected nodes.
<b>SimpleLink</b>	SimpleLink is an micro controller platform developed by Texas Instruments that includes the broadest portfolio of wired and wireless Arm® MCUs (System-on-Chip) in a single software development environment.

## List of symbols

**Power supply** 

**Resistor** 

**Switch** 

**Volt Meter** 

## Version history

Name Author	Date	Reason for changes	Version
Herm Lecluse	15-02-2018	Initial Version	1.0
Herm Lecluse	19-02-2018	Implemented supervisors feedback	1.1
Herm Lecluse	02-03-2018	Implemented teachers feedback	1.2
Herm Lecluse	19-03-2018	Initialization Midterm report	2.0
Herm Lecluse	26-03-2018	Mid Term Report	2.1
Herm Lecluse	26-03-2018	Implemented supervisors feedback	2.2
Herm Lecluse	16-04-2018	Initialization thesis	3.0
Herm Lecluse	26-04-2018	Implemented Feedback midterm presentation	3.1
Herm Lecluse	05-06-2018	Bachelor thesis report first draft	3.2
Herm Lecluse	07-06-2018	Bachelor thesis implemented feedback	3.3

# 1 Introduction

Phact is a 'one stop shop' for software and hardware solutions and secondment. Phact was founded in 2015 with nearly 25 years of experience in the telecommunication branch. [Phac17]

How? Phact originated when a group of people with loads of experience decided to focus on a new horizon. Many of Phact's employees have worked in the telecommunication sector for years and took their expertise to this new endeavour. Phact was born.

People in the team have been developing telecom hardware and software for VoIP and SS7 carriers since 1991. Knowledge is a key part of the business. Therefore, Phact is engaged in close partnerships with our professional partners and suppliers.

Phact might still be very young, but they carry years of experience in software and hardware development.

Currently, Phact B.V. is looking into new possibilities for an existing chipset that has been created by Texas Instruments. This chipset is capable of transmitting data over a sub 1 GHz frequency.[TICC17]

This paper will inform the reader about what problem is at hand and how this project should solve that problem. What deliverables this project has and how these deliverables should be achieved. Stakeholders will be introduced and the current version of the risk analysis will be explained. Furthermore the research done is explained and results are shown. These results have been used to Design a software layer in order to create software to solve the named problems.

## 2 Project Plan

### 2.1 Context

Phact B.V. uses the TI CC 1350 chipset in order to create industrial Internet of Things (IoT) solutions. This chipset is a ultra-low-power, cost-effective wireless Microcontroller Unit (MCU) capable of handling both Sub-1 GHz and 2.4-GHz Radio frequencies[TICC17]. For this project the TI CC 1350 Launchpad is used. The launchpad essentially simplifies communication with the interfaces the chipset provides.



Figure 2.1: TI CC 1350 Launchpad

The pins on the board shown in the picture above are connected to the pins on the MCU. This means that using the launchpad with different kind of sensors is quite easy to do.

Phact B.V. is missing functionalities inside Texas Instruments micro controller platform (SimpleLink™) Operating System (OS) which is called SimpleLink™. The biggest flaw in this OS is the fact that it only supports star networks.

Imagine a node that is too far away from its gateway but in range of a second node. In the current situation this would mean that the first node sends out data, which could not be received by the gateway. What if this first node would be able to transmit its data through the second node to the gateway. This feature is called meshing and essentially means that the first node still can transfer its data to the gateway without being directly in range of the gateway.

Another flaw in SimpleLink™ is the support for deepsleep. The code is very complicated and looks devious. Phact wants this simplified, so that their applications use less excessive power and by doing so, enlarge the time between battery charge cycles.

Contiki[Cont18], an open source OS that provides these missing functionalities, but Contiki is not updated regularly. Contiki is also not supported by Texas Instruments (TI). This is a problem for Phact B.V., as they want to ensure that their application is supported by TI

Amber Wireless is a manufacturer and has made the AMB8826 chipset, which encapsulates the TI CC 1310 module ( which is the same module as the CC 1350 except it has no bluetooth connectivity ).



Amber Wireless extended the SimpleLink™ platform to make it easier to use. The downside of this AMB8826 [AmWi17] is that there is no Input/Output (I/O) compatibility for sensors / actuators which Phact B.V. does desire.

### 2.1.1 Case

This subsection will introduce a case in which the functionalities requested by Phact B.V. are addressed. Furthermore the case will help the reader understand why these functionalities matter.

Imagine a restaurant at the beach. This restaurant has waiters who take orders with a smartphone. These smartphones are connected via WiFi to a modem. An order system in the kitchen is also connected to this network. On a busy day this can cause problems. Due to the fact that (almost) everybody has a smartphone with a WiFi module installed. These modules cause noise in the air. This noise could disturb the connection between the waiters phone and the modem which means no orders can be sent to the kitchen anymore.

The solution Phact B.V. is developing is as follows : Turn off WiFi and mobile data on the smartphone. Connect the device via bluetooth to a small device strapped to the waiters belt. inside this device is a TI CC 1350 which is connected via sub 1 GHz to another CC 1350, which is hooked up to the order system. The waiter can send the information via bluetooth to the device, the device will then transmit the message to the order taking system. This solves the problem at hand because no smartphone has a built in sub 1 GHz module and so cannot create noise on the network anymore.

## 2.2 Lead-up to the assignment

Phact B.V. has applications which run on the CC 1350 chipset and experience that the missing features in the SimpleLink™ described above, are limiting the possibilities of this module. Phact B.V. would like to use this technology for more than the supported OS offers. So that their applications would be more reliable and sustainable in practice.

## 2.3 Goal

The main goal of the commissioning company is to be able to extend usability of the Texas Instruments CC1350 micro controller unit. This opens up opportunities for Phact B.V. to create more applications using this device.

## 2.4 Expected results

- A software layer for the Texas Instruments CC 1350 MCU on top of SimpleLink™ that provides functionality for the MCU so that it is able...:
  - To communicate inside a mesh network.
  - To enable deep sleep for an application.
  - To update the firmware over the air.
- Research report on how to implement the functionalities above.

## 2.5 Scope

Since there are different operating systems, two of which have some of the desired functionalities, a research will be done into Amber Wireless. on how to implement these functionalities for SimpleLink™.

This project will cover a software layer for SimpleLink™, that will be tested and prototyped.

This project will not cover software development for Contiki or Amber Wireless.

The product to be developed for is the TI CC 1350. This was a decision made by Phact B.V.

## 2.6 Program requirements

The software layer has to enable the Texas Instruments CC 1350 MCU nodes to communicate inside a mesh network. This is important because there are cases in which a node could lose its direct connection with a gateway. If this happens the node should first look for a new gateway to connect to, if there is none, try to communicate through another device that is in range of a gateway.

The software layer should also provide functionality for deep sleep. This way it is possible for devices to save energy and so save maintenance costs. (replacing batteries). It is close to impossible to have both these functionalities active at the same time, so the program should be able to toggle between these modes.

Update over the air is important as well. In order to update the applications which are running on devices potentially across the entire world. This is especially interesting for devices which are in the deep sleep modus since the devices are offline once in deep sleep they have to be awake when an update gets pushed to these devices. However, this functionality is the least important of the three. (only if there's time left.)

## 2.7 Project phases

This project is built from different phases. These phases are being explained in this section.

### 2.7.1 Plan

In this phase the first version of this document was created. This phase has also been used to initialize the project, to get used to the new work environment and get in contact with the new colleagues.

### 2.7.2 Research

In this phase, research into the to be developed system will be done. Existing sources will be analysed. The main questions that this research should answer are: “How are the functionalities Phact B.V. desires implemented in the other Operating Systems?” and “How can these functionalities be implemented in SimpleLink™.

The steps of this research are:

- Research in mesh networking and implementations
- Research in deep sleep implementations
- Research in SimpleLink™
- Research in Texas Instruments-Real Time Operating System
- Research in Texas Instruments CC 1350 Microcontroller Unit

### 2.7.3 Development

During this phase the applications will be developed and tested. The end report has been created during this phase as well.

## 2.8 Deliverables

This project has several deliverables, in this section these deliverables are described.

- Project plan
- Research paper
- Mid term report
- Prototype
- Bachelor Thesis

In the next section these deliverables are used to define the activities for this project.

## 2.9 Activities

This section will show the reader what activities this project has.

Activity	Project phase
Creation of Project plan	Planning phase
Delivery of Project plan	Planning phase
Implement feedback on Project plan	Planning phase
Research	Research phase
Report on research	Research phase
Creation of mid-term report	Research phase
Delivery of mid-term report	Research phase
Implement feedback on mid-term report	Research phase
Creation of Software Layer	Development phase
Testing the prototype	Development phase
Delivery of product	Development phase
Creation of report	Development phase
Presenting report and the prototype	Development phase
Handover	Development phase

Table 2.1: Table of project activities

## 2.10 Project management

### 2.10.1 Time

To be sure that the project is feasible in the given time milestones have been defined with a due date bound to it. These milestones are explained later on in this document. The milestones are set up in Taiga[Taig18], which is a project management platform for agile development. In Taiga, the supervisor can track the progress of the project. The development phase shall be done in an agile way because the internship supervisor wants to be able to track the progress that has been made based on functionalities that were implemented. An agile approach is what suits best for this project.

### 2.10.2 Quality

In order to deliver a product of high quality and according to the expectation of the supervisor, weekly meetings will be planned to discuss the current progress. The feedback of these meetings will be used in order to improve the quality of the product. Next to the meetings, the code will be well documented by making sure that all the functions are commented, but also inline commenting will be done. In order to make sure that the application behaves the way it should, integrity testing will be done. These tests show if the code works the way it should.

## 2.11 Risk analysis

This project has several risk factors. If one of those risk events happen, than such a risk could endanger the progress of the project. These risks have been analysed and listed here:

- It turns out that SimpleLink™ prohibits a certain feature.

**Consequence:** This function cannot be built for the desired application.

**Solution:** Plan a meeting with the supervisor to check for a fitting solution.

**Responsible:** Intern.

- Texas Instruments no longer supports the CC 1350 Microcontroller Unit

**Consequence:** The integrated development environment (IDE) of TI can no longer be used

**Solution:** Check if there is an alternative chipset available which is supported by TI or another manufacturer.

**Responsible:** Intern.

If one of these risks occur, a meeting will be planned with (some of) the stakeholders to look if a named solution is possible. if not an alternative will be discussed.

## 2.12 Project Organization

### 2.12.1 Stakeholders

This project has several stakeholders who need to be kept informed on the status of the project. Those stakeholders have a level of interest and a level of power. those are displayed in the table below.

Stakeholder	Level of interest	Level of power
Marco Langenhuizen (teacher)	Medium	High
Johan Kleuskens (supervisor)	High	High
Herm Lecluse (intern)	Very High	Medium

Table 2.2: Table showing the stakeholders of this project

### 2.12.2 Communication

To ensure that the stakeholders get information about the status of this project, there will be frequently contact with them. This contact may differ from stakeholder to stakeholder. For example, the teacher will be informed via mail and reports. This teacher will also visit Phact B.V. minimal once. The supervisors will be informed weekly via meetings and will receive a copy of each report as well.

## 2.13 Milestones

This section will define the milestones for this project. A list will be shown in which is explained when the intern expects to finish these milestones.

<b>Id</b>	<b>Milestone</b>	<b>Due date</b>
001	Project plan	19-02-2018
101	research document	19-03-2018
201	Mid-term report	23-03-2018
202	Mid-term presentation	23-03-2018
301	Meshing functionality	06-04-2018
302	Deep sleep functionality	20-04-2018
303	Over the air updating	04-05-2018
304	Combining the functions	01-06-2018
401	Final report	08-06-2018
402	Presentation	17-06-2018
501	Handover	01-07-2018

## 3 Research and results

In this chapter the research that has been done will be discussed. The methods used will be addressed and results will be shown. The rationale for choices will be explained here as well.

### 3.0.1 Overall decisions

In this subsection a few decisions will be defined that were prerequisite for this project.

- The software layer to be created shall be created in C. This choice was made due to the fact that the SimpleLink™ is also written in C. This makes extending the operating system easier.
- The software layer to be created will be designed for the Texas Instruments CC 1350 Launchpad. This is a demo kit for the Texas Instruments CC 1350 MCU. This enables easy debugging of the micro controller and easy General Purpose Input/Output (GPIO) access to the developer.
- The software will be stored in a Git repository on Phact's GitLab. This way the software is backed up to prevent loss of data.

### 3.0.2 Questions

The (sub)questions regarding this research are listed here.

- How are the requested functionalities implemented in the other operating systems?  
How can these functions be implemented in SimpleLink™  
Are these different implementations useful for this project.?
- What is SimpleLink™?  
How do we communicate between multiple devices on the sub 1 GHz band?
- What is meshing?  
How to setup a mesh network?
- What is deep sleep?  
What is the effect / impact of deep sleep on the power consumption?  
Is deep sleep worth implementing?  
How to bring a device in deep sleep and wake it up again?

### 3.0.3 Methods

The methods that have been used for this research are literature research and an experiment. Reading and learning how to use code has been the main activity during this phase. The second activity was the experiment on deep sleep and a sniff test to see how messages look like.

## 3.1 SimpleLink

SimpleLink is an micro controller platform developed by Texas Instruments that includes the broadest portfolio of wired and wireless Arm® MCUs[SiAO17] (System-on-Chip) in a single software development environment.

- Customer Application Code
- SDK Plug-ins
- The SimpleLink MCU SDK

The first part is the custom application that the MCU should execute. This is the layer in which the behavior of the chipset is programmed The rest is handled by TI in the next two layers.

The SimpleLink Software Development Kit (SDK) is extensible through SDK Plug-ins, modular software that adds support for external components and software. These plug-ins help the developer implement new hardware to the system.

The third part is the actual SDK provided by TI to help developers create new software. It contains drivers for hardware, communication stack's and code examples. Next to these functionalities, the SDK also contains the OS Kernel for TI-RTOS. This is the Operating system that the application will run on.

### 3.1.1 Communication

Communication means that two or more devices are exchanging messages with each other. in this section a closer look will be shown to how this is done in the TI-15.4 [SiSO17] stack of SimpleLink™.

This stack is an implementation of the IEEE 802.15.4 standard. This is a standard for low rate wireless personal area networks (low data rate but very long battery life and very low complexity applications). This stack defines the first two layers of the OSI model. The Data-link (layer 2) will from now on be called: Medium Access Control (MAC). This is how Texas Instruments defined the data link layer. This stack communicates through the Indirect Call Framework (ICALL) module with the logical layer and that layer communicates with the application the developer created. The architecture is shown on the next page.



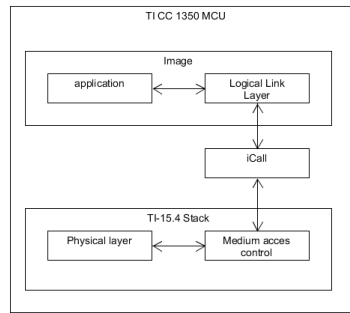


Figure 3.1: TI CC 1350 architecture

## Connection

A connection cannot be made without initializing the Gateway first. In order to do that, SimpleLink™ physical layer resets the MAC. After this the application requests the MAC to scan for the channel with the least amount of radio interference. this returns a list with channels. These channels will then be scanned on active networks. This returns the best channel for the gateway to start its network. After this the gateway will setup its Personal Area Network (PAN) and open its network. From this point on the MAC will send out a beacon every x seconds for nodes that are trying to access the network. This is visualised in the sequence diagram down here:

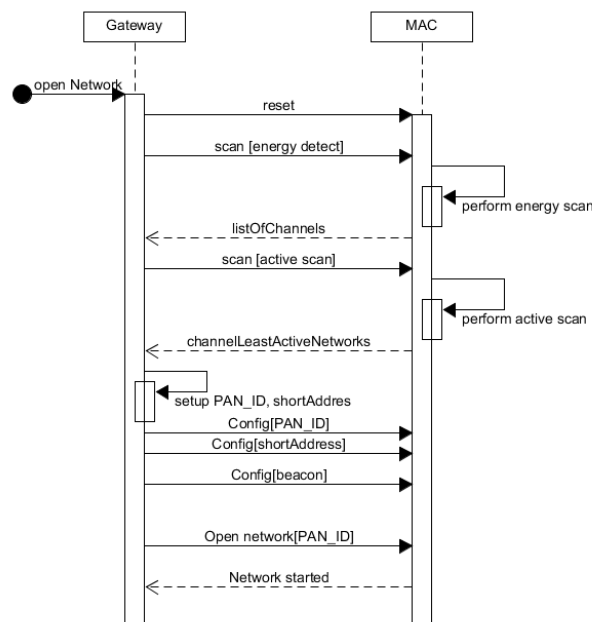


Figure 3.2: circuit scheme

Now that the gateway is up and running a node can join the network. A node would first also reset its MAC layer. The node will then scan for active beacons. Once it finds a beacon the node will configure itself to that PAN. and syncs to that beacon. To acquire beacon synchronization, the device calculates the time difference between the two beacons and uses that time to synchronize itself. From this point on, a connection has been established. The node gets an index number from the gateway to identify itself in.

## Transmission

Once the connection is established, the node can transmit packages over the network. each message has a sequence number which starts counting from the moment that the node connects to the network. sending a message from node to sensor looks like this:

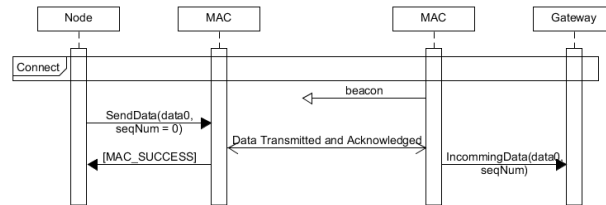


Figure 3.3: transmission

In the other direction the process works the same way except the gateway sends the message and the node receives it. If the node is not in range of the network any more, the data will get lost.

## Disconnect

If a node wants to disconnect from the network, it will send a final message to say 'goodbye'. This will get acknowledged and the session (bound to the ID given upon connection) gets closed. If the gateway closes the network, the node will receive a message that the node is no longer part of that network.

## 3.2 Deep sleep

This section will inform the reader about the research that has been done in deep sleep.[DeSl09] The section will answer the following questions. What is the effect of deep sleep on the power consumption? What is the impact of deep sleep on maintenance costs? And is deep sleep worth implementing? In order to answer these questions a small experiment was prepared and executed. For the last question, 'What is deep sleep?', literature and content research was used to find the answer.

Deep sleep is a state that completely shuts down power consumption of a micro controller. This essentially means that the chipset limits its power consumption to just 185 nA. (0.2 $\mu$ A) [SiDs17] In this state no GPIO, connectivity or any other function of the MCU can be used. In order to use them again the chipset 'wakes up' from his deep sleep state in which the functions regain their usability. The main purpose of the experiment to be executed is to check the effect of deep sleep, and to proof why deep sleep is useful. The following circuit was created:

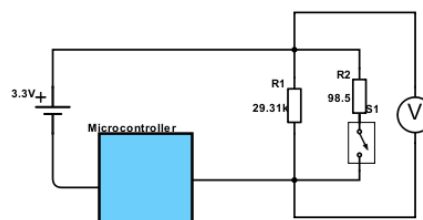


Figure 3.4: circuit scheme

As you can see, there is a switch after R2, this resistor is in place just in order to enable the system to boot. once the switch is closed the total resistance is  $98.17\Omega$ . when the device goes into deep sleep the switch is toggled and the resistance changes to  $29.31k\Omega$ .

This is done because the volt meter is not able to measure the voltage that is running over the circuit with a resistor of  $98.17\Omega$ .

By following the Ohm's law, the amount of volts is equal to the amount of resistance times the current. So by increasing the resistance, the voltage is increased and so measurable by the voltmeter. By using the values measured from the voltmeter and the given resistance values, the current can be calculated. This is done in the table below.

	Idle	Sleep	Transmit
U	0.391V	0.223V	0.018 V
R	$98.5\Omega$	$29.31k\Omega$	$2.2\Omega$
I	3.97mA	$7.61\mu A$	13.4mA

Table 3.1: Table displaying measured values

Table 3.1 shows that there is a big difference in current when the micro controller is idle or in deep sleep. in fact it is almost 525x less when the chipset is in deep sleep.

If we compare that value to what the datasheet showed, a difference can be seen in power consumption.  $7.61\mu A$  was measured and the datasheet shows this should be approximately  $0.2\mu A$ . This difference can be explained by the flash memory that is on the launchpad and is still powered on. This unit uses  $5\mu A$  typically[MXDS17]. This leaves us with  $2.61\mu A$ , a  $2.41\mu A$  difference. This difference can be explained by looking at the accuracy of the measuring equipment, a multimeter is not 100% accurate. Since the value is so small, it's quite likely to measure a wrong value.

For the rest of the experiment, the value  $7\mu A$  is used, due to the fact that this is the actual consumption measured when in deep sleep.

Imagine that the micro controller is powered by a button cell[WiCR17] battery (or watch battery) with a capacity of 225mAh. This means that a TI CC 1350 chipset can run for 56 hours on such a battery when idle. However when in deep sleep, this chipset can run for approximately 3.8 years. This result shows that being able to toggle deep sleep for the TI CC 1350 can cut in maintenance costs quite a lot.

A device that is constantly sleeping is not very useful. If the time it takes to send out a message is measured, the total amount of current required can be calculated. To measure how long it takes to send out a message an oscilloscope is connected to the circuitry shown in figure 3.4 like this :

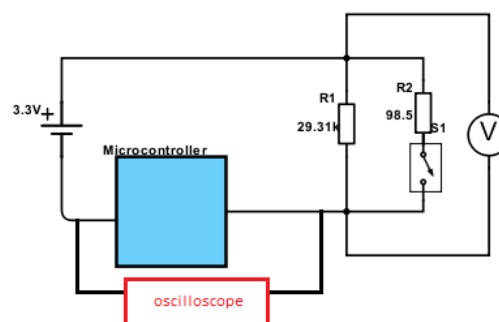


Figure 3.5: circuit scheme oscilloscope

Next up the chipset was configured to be constantly transmitting messages. (with a 60ms message interval to ensure a clear picture on the oscilloscope). the voltage measured is the voltage over the micro controller. Kirchhoffs second law says that "The algebraic sum of the products of the resistances of the conductors and the currents in them in a closed loop is equal to the total emf available in that loop." [WiKH18] which means that when a bigger current is flowing through a circuit, less Voltage will be left for the rest of the circuit. So whenever a message is sent. Less voltage will be measurable by the oscilloscope. This becomes visible in the image below: The white box in the image shows  $\Delta t$  which means the difference in time between the

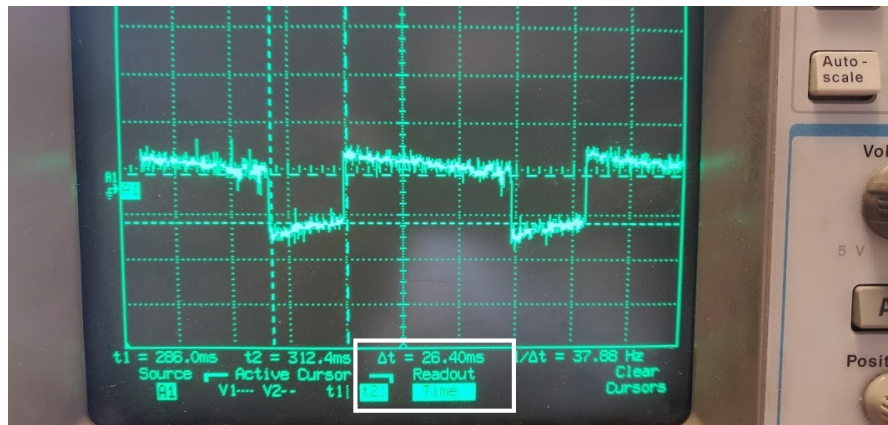


Figure 3.6: screenshot oscilloscope

voltage dipping in, and recovering to normal. This is the time it takes to send out 1 message of 150 bytes. If, for example the waiter from the case (2.1.1), takes 80 orders per hour (on an average of 150 bytes per order). Then the total time actual transmitting orders would be  $26.4ms * 80 = 2112ms$ . this would total to a power consumption of 54.6 mA/h (implying that the application goes to deep sleep immediately after sending). If, the waiter works without breaks, the battery would die out on 1.7 years (80 orders per hour, 150 bytes per order) instead of the 2 days without deep sleep. So in order to answer the question is deep sleep worth implementing? the results above show yes, it most definitely is worth implementing.

### 3.2.1 Sensor Controller

Once the controller is in deep sleep the only part still gets powered is the sensor controller. which essentially checks a value measured by for example an external sensor or a timer. Once the value crosses the threshold set, the sensor controller will create a hardware interrupt (or debounce) which will wake up the sensor. In the experiment above a button was used to create the trigger.

## 3.3 Mesh networking

This section will inform the reader about the research that has been done in creating a mesh network with the CC 1350 chipset.

Meshing is a way of connecting devices dynamically to one gateway which not necessarily means that the device is connected to the gateway directly. [WiMN18] There are many different existing mesh implementations. These implementations can be divided into 3 different categories, flood, semi-flood and routed meshing. Flood meshing means that all nodes in the network repeat an incoming message. This is a quick and easy way for small networks. However the bigger the network, the more messages are repeated. this can cause filling up the network with repeated messages. Routed meshing is a way of meshing that sends messages to a

specific node which then repeats it to the next specific node. This is a highly efficient network but also very complicated to implement. In contrary to the flood meshing, routed meshing does not fill up that quick, since only one node repeats a message. This implementation requires nodes to be physically in the same place all the time (otherwise the route known to a gateway could be incorrect). Semi flood is a combination of the two above, The gateway 'knows' in which particular direction the nodes live, and sends the message into one of the two directions. after which the network repeats the message just like in the flood mesh.

Amber Wireless[AmWi17] has implemented a mesh network, By taking a look at what they have done it became clear that Amber Wireless implemented a flood mesh. a overview of how flood meshing works is given in the image below.

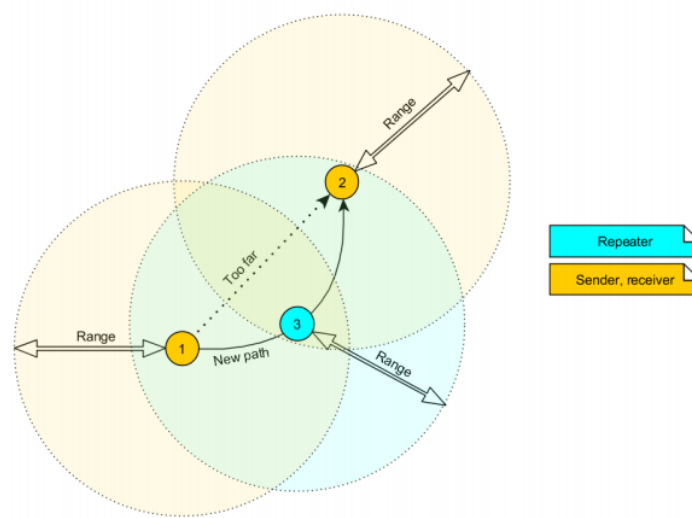


Figure 3.7: flooding mesh Amber Wireless

Since the case described in section 2.1.1 showed that the waiter is not in one place all the time, routed mesh is no option for this case. Semi flooding could be useful but brings a lot of overhead with it. Namely, keeping track of which 'side' of the network an incoming message belongs. The message transmitted would also have to contain an extra bit to indicate what side of the network it came from. and repeaters have to indicate if the message is theirs to repeat. Flooding however, is well documented by Amber Wireless (including pitfalls), has little to no overhead and does what it should do namely, expand the range of the network. For this project flood meshing will be implemented for the reasons listed above.

### 3.3.1 Flood mesh

The chosen mesh strategy is flood meshing. this subsection will explain what has to be done to create a flood mesh network.

A repeater is used to relay the messages inside a network. Such a repeater will need to be able to receive and transmit messages. Such a repeater has to be line-powered. receiving and transmitting messages costs power, a battery will die too fast.

If, a network needs a bigger data throughput (more messages or bigger messages) a parallel network can be setup (different frequency). This does not affect the primary network and increase the amount of data sent that can be transmitted simultaneously.

The repeater has to verify that the message received has not been retransmitted already. This in order to make sure that multiple repeaters are not filling up the network by sending messages between themselves. The timeout for sending out a message has to be bigger. Because repeating a message means that no acknowledgment is instantaneously replied.

### 3.3.2 Sniffer

In order to analyze the packages transmitted in the air, a sniffer was set up. This essentially is a TI CC 1350 which listens to a specific frequency and shows all incoming messages. This helped in analyzing the messages sent by the end node, repeater and gateway. This helped during development of the repeater because all the traffic in the air was immediately visible. The screenshot below shows proof of a message repeated and an acknowledgment repeated in the network.

No.	Time	Source	Destination	Protocol	Len	Info	RSSI
16	1.286829	0x0002	0xaabb	TI 802.15...	148	Data, Dst: 0xaabb, Src: 0x0002	-100 dBm
17	0.020169			TI 802.15...	53	Ack	-38 dBm
18	0.008709	0x0002	0xaabb	TI 802.15...	148	Data, Dst: 0xaabb, Src: 0x0002	-25 dBm
19	0.020169			TI 802.15...	53	Ack	-39 dBm
20	0.013674			TI 802.15...	53	Ack	-25 dBm

▶ Frame 15: 53 bytes on wire (424 bits), 53 bytes captured (424 bits) on interface 0 ▶ Internet Protocol Version 4, Src: 192.168.1.3, Dst: 192.168.1.3 ▶ User Datagram Protocol, Src Port: 17760, Dst Port: 17760 ▲ TI Radio Packet Info Interface: COM 9 Frequency: 863.125 MHz Channel: 0 PHY: 50 kbps GFSK RSSI: -25 dBm Frame Check Status: 0x80 - OK ▶ TI 802.15.4GE SUN PHY without Mode Switch ▶ TI 802.15.4GE Ack, Sequence Number: 101
--

Figure 3.8: screenshot Wireshark

Frame 16 (ID comes from the left column No.) was transmitted by the sensor. This is the data package which has to be transmitted. The last column (Received Signal Strength Indication (RSSI)) shows that the signal received on the repeater is -100 dBm. Two packets later, on frame no. 18 this same package is received by the sniffer but now with a much stronger RSSI namely -25 dBm. This is the repeater retransmitting the package after a small pause. The same can be seen for the Acknowledgement sent out by the collector in frame 17 and 20.

Amber Wireless only repeats the messages but not the acknowledgments sent back to the nodes. In that regard the repeater created by the intern is an improvement of the Amber Wireless module.

### 3.4 Conclusion

The research phase of this internship showed the following results:

- SimpleLink

The way messages are transmitted has become clear.

How the different layers of SimpleLink work.

- Meshing

Amber Wireless has an incomplete implementation of the Mesh network because it misses out on acknowledging the received messages.

Meshing can be achieved in multiple manners.

- Deep sleep

Deep sleep has a great impact on the lifetime of the chipset on a battery.

Deep sleep is triggered by powering the complete chip down except for the sensor controller. This controller is capable of waking up the device again

Deep sleep is most certainly worth implementing.

How the communication between chipsets works in SimpleLink<sup>TM</sup>OS has been learned by reading the documentation from Texas Instruments, the developers manual has also been used in order to understand how the IEEE 802.15.4 packages are build.

Next to literature research, knowledge about deep sleep and meshing has been learned by experimenting with the MCU.

In the end, no further research was done into Contiki. This is because during the research The functionalities requested by Phact B.V. were already satisfactory.

To conclude this research the knowledge needed to continue to development is gathered and has been used for the rest of the project.

## 4 Software Requirement Specification

In this chapter the requirements of the system will be explained. A Software Requirement is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software.

The requirements this project has, are listed below.

### 4.1 Functional Requirements

1. The software layer should be able to toggle the chipset into different states.
2. The software layer shall be able to run on any TI CC 13x0 device.
3. The software layer shall be able to transmit messages over a sub 1GHz network (IEEE\_802.15.4)
4. The user will be able to use this layer to increase the functionalities of the chipset.
5. A prototype for the software layer will be created.
6. The system must be able to handle at least one connection (between node and gateway).

### 4.2 4.2 Non-functional requirements

7. A user manual describing how to use the system shall be made.
8. The code will be tested.
9. The code shall be documented, as well in line as per method.

These requirements are high level requirements for more details the whole software requirement specification can be found in appendix A. These requirements have been discussed with the Project supervisor.



## 5 Design

This chapter shows the reader the design decisions made. Design documents of the designed system will be shown and explained.

The designed system consists of 2 applications. A node that acts as a repeater, and a node is capable of entering deep sleep and only waking up in order to do a (set of) task(s).

### 5.1 Usecase

In order to properly understand what these applications can be used for, a scenario was sketched previously (chapter 2.1.1). This section will show the use case regarding that scenario.

Use case :	Send an order via repeater to the gateway
Actor	Waiter
Description	The waiter has taken an order from the guests and has to send the information to the kitchen over sub 1 GHz network via a repeater.
Precondition	Device is in range of a repeater
Flow of events	<ol style="list-style-type: none"> <li>1. Waiter wakes end node.</li> <li>2. Waiter sends order to the kitchen via end node.</li> <li>3. End node transmits data package on sub 1 GHz frequency.</li> <li>4. Repeater node receives data package.</li> <li>5. Repeater transmits received package.</li> <li>6. Gateway collects data and sends acknowledgement.</li> <li>7. Repeater receives acknowledgement.</li> <li>8. Repeater transmits acknowledgement.</li> <li>9. End node receives acknowledgement.</li> <li>10. End node shows waiter that the transmission was successful</li> <li>11. End node goes back to sleep.</li> </ol>
Exceptions	Message is lost. —> go back to step 3 and repeat.
Result	Order has been transmitted

## 5.2 Sequence diagram

This section will show the sequence diagram that matches the use case described above.

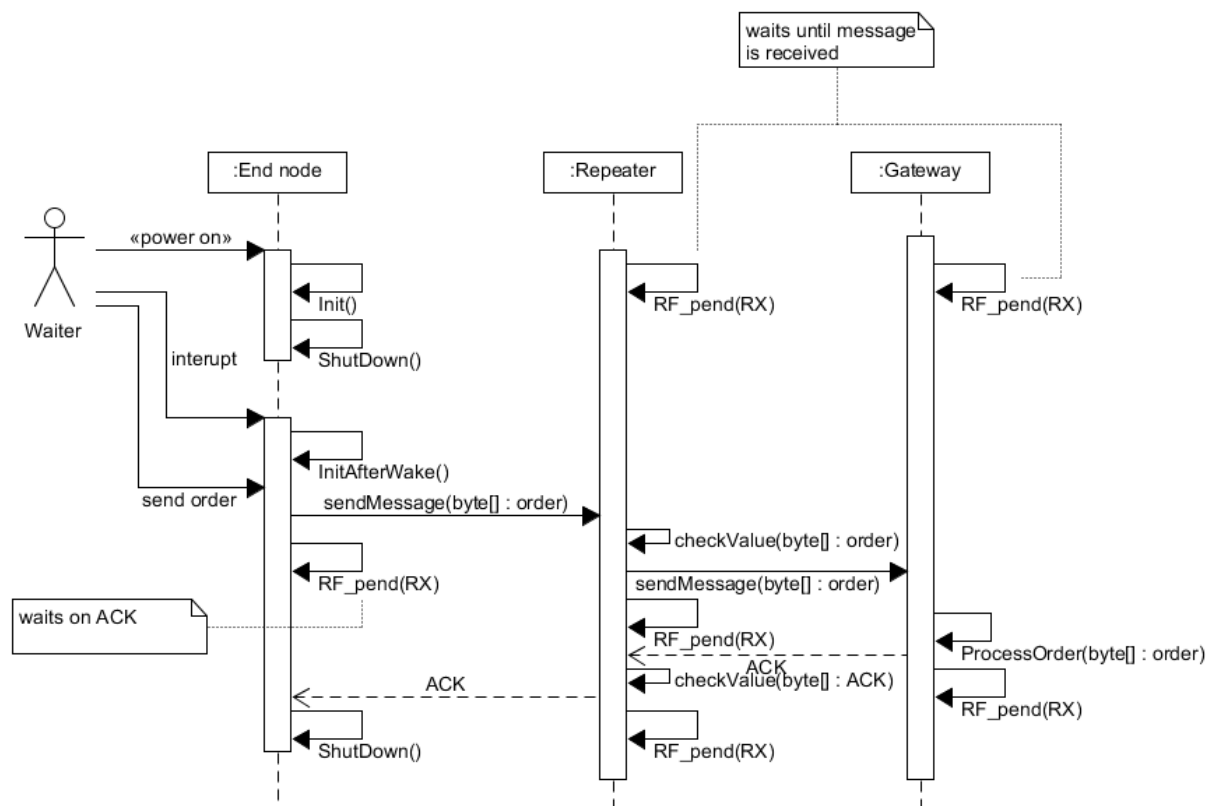


Figure 5.1: sequence diagram send an order via repeater to the gateway

The picture above shows how the communication between the different nodes inside the network is done. The waiter wakes up the end node via an interrupt (this can be a button or something else creating an interrupt). After which the device sends the message to the kitchen. Meanwhile, the repeater is waiting for a message to repeat. Once it receives the message the repeater will check for uniqueness of the message to prevent repeaters from repeating a message infinitely. The repeater sends the unique message to the gateway which will acknowledge the message and takes care of the order. The repeater sees this acknowledgement and checks this for uniqueness as well. The acknowledgement is passed back to the end node, which enters deep sleep afterwards.

### 5.3 State machine diagrams

The two applications are typical state machines. In order to describe how the states are switched and when, state machine diagrams have been made.

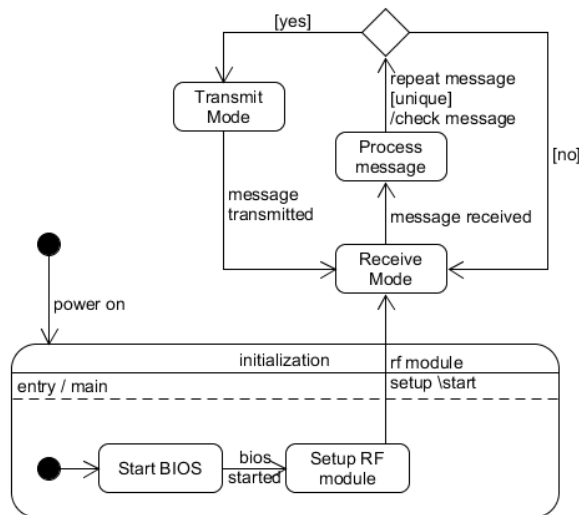


Figure 5.2: state diagram repeater

The repeater is first initialized, during initialization the Basic Input/Output System (BIOS) of the chipset is started. After which the processor can setup the Radio frequency (RF) module. The device is now ready to receive messages (Receive mode state). When a message is received, the machine goes into processing mode, in which no message can be received or transmitted. This state checks if the message is unique, if so the message will be retransmitted, otherwise the receive mode state will be active again. This loop will continue forever.

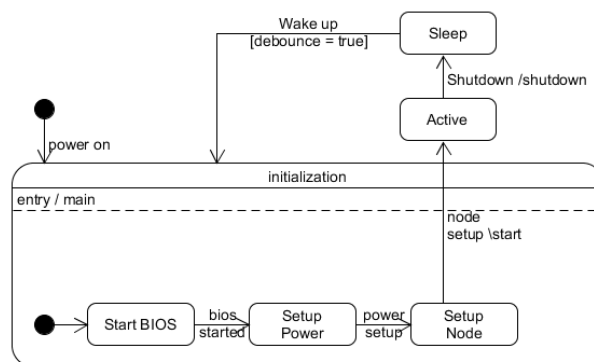


Figure 5.3: state diagram deep sleep

The deep sleep node is (just like the repeater) initialized. First, the kernel is initialized in the BIOS start state. The Power interface setup is the next state. The power interface is used to control the power properties of the chipset. At last the node is setup. In this state the node specific code (functionality) is initialized. Now that the chipset is initialized, the state switches to active, in which the node executes its logic and afterwards enters deep sleep. The state only changes when an interrupt is sent to the sensor. This is done by the sensor controller.

## 6 Development

This chapter will inform the reader about the software that has been implemented based on the design explained in the previous chapter. Code will be shown and algorithms will be explained.

### 6.1 Meshing

As explained earlier flood meshing can be achieved by repeating the message to the next node. This section will show how this application enabled the TI CC 1350 to be used inside a mesh network. In order to repeat a message the following things need to be done.

1. Initialize (only once)
2. Receive message (RX)
3. Handle message (Validation)
4. Transmit message (TX)

#### 6.1.1 Initialization

To initialize the repeater a thread is created and given the following function as its execution task.

```

1  /*
2  * Thread function.
3  * This function is fed to a thread instance. Code runs after BIOS_START is called.
4  */
5  void *mainThread(void *arg0)
6  {
7      RF_Params rfParams;
8      RF_Params_init(&rfParams);
9
10     // Initialize a queue for RF operations
11     if( RFQueue_defineQueue(&dataQueue, //queue object
12         rxDataEntryBuffer, // arraybuffer for incoming messages
13         sizeof(rxDataEntryBuffer),
14         2, //number of operations ; 1 RX and 1 TX command
15         PAYLOAD_LENGTH)) // Packetlength = 150
16     {
17         /* Failed to allocate space for all data entries */
18         return;
19     }
20
21     /* Request access to the radio */
22     rfHandle = RF_open(&rfObject, &RF_prop,
23         (RF_RadioSetup*)&RF_cmdPropRadioDivSetup, &rfParams);
24     /* Set the frequency */
25     RF_postCmd(rfHandle, (RF_Op*)&RF_cmdFs, RF_PriorityNormal, NULL, 0);
26 }
```

As visible in the code listing on the previous page, Parameters for the RF module are defined and initialized. RF\_Params struct's fields are set to its default values. For example, the Inactivity time-out is set to BIOS\_WAIT\_FOREVER by default. After this, a queue is defined. This queue will contain all operations done in one cycle. In this case that's two, one Receive (RX), and one Transmit (TX) operation. In this implementation a maximum packet length of 150 bytes. This because the messages transmitted in the network are never bigger than 150 bytes. The packet size is a global variable that can be configured by the developer to suit their needs. After definition of the queue, the RF module is opened with the configured parameters. This handle will give the application access to the RF module and thus the possibility to receive or transmit messages. Last but not least, The frequency is set on the handle to be able to listen to the nodes that are transmitting on that frequency.

From this point on, the repeater is configured and is able to receive and handle incoming messages.

### 6.1.2 Receive (RX)

In order to pend for a message, a command for the RF module is introduced. This command is parsed to the RF handle which results in the RF handle listening to the frequency in order to receive a message.

```

1 rfc_CMD_PROP_RX_ADV_t RF_cmdPropRx;
2 /* CMD_PROP_RX commands for application needs */
3 /* Set the Data Entity queue for received data */
4 RF_cmdPropRx.pQueue = &dataQueue;
5 /* Discard ignored packets from Rx queue */
6 RF_cmdPropRx.rxConf.bAutoFlushIgnored = 0;
7 /* Discard packets with CRC error from Rx queue */
8 RF_cmdPropRx.rxConf.bAutoFlushCrcErr = 0;
9 /* Implement packet length filtering to avoid PROP_ERROR_RXBUF */
10 RF_cmdPropRx.maxPktLen = PAYLOAD_LENGTH;
11 /* End RX operation when a packet is received correctly and move on to the
12 * next command in the chain */
13 RF_cmdPropRx.pktConf.bRepeatOk = 0;
14 RF_cmdPropRx.pktConf.bRepeatNok = 1;
15 RF_cmdPropRx.startTrigger.triggerType = TRIG_NOW;
16 RF_cmdPropRx.pNextOp = (rfc_radioOp_t *)&RF_cmdPropTx;
17 /* Only run the TX command if RX is successful */
18 RF_cmdPropRx.condition.rule = COND_STOP_ON_FALSE;
19 RF_cmdPropRx.pOutput = (uint8_t *)&rxStatistics;
20
21 //Send command to the handle, returns reason of termination (e.g. Success or
22 RF_EventMask terminationReason =
23 RF_runCmd(rfHandle, (RF_Op*)&RF_cmdPropRx, RF_PriorityNormal, echoCallback,
24 (RF_EventRxEntryDone | RF_EventLastCmdDone));

```

The code above shows the command for the RF handle and how it is configured for this application. As you can see, a data queue is given to this command. This is done in order to store incoming data to that queue so that the handler can use the dataset on a later point in time. The next interesting part is the pNextOp field. This field stores the next RF command inside the struct. This speeds up the relaying part quite heavily. Since all the settings are in place already and don't have to be set again, which saves processing cycles. This command is the Transmission command, which will be explained later this chapter. The RF\_runCmd function sends the command to the RF module. This function will pend on an incoming message. Once it receives something, the echoCallback function is called. This callback handles the incoming message.

### 6.1.3 Handle Message

The message received is bitwise inverted. This means that in order to understand the message all individual bytes have to be reversed. luckily for repeating only 3 bytes have to be inverted namely the Sequence number of the message (one byte) and the source address (two bytes). This information is used to store the message as last seen message. The code snippet below shows how these bytes are reversed.

```

1 /*
2  * Util function to reverse an byte
3  */
4 uint8_t BitReverse(uint8_t byte){
5     int steps=7; //byte has 8 bits , maximum nr of steps is 7
6     uint8_t v = byte; // r will be reversed bits of v;
7     uint8_t r = v;
8
9     for (v >>= 1; v; v >>= 1)
10    {
11        r <<= 1; // shift bits to the left
12        r |= v & 1; // OR the bits with the bits to the left
13        steps--;
14    } // once v = 0, the bits have been switched
15    r <<= steps; // shift when v's highest bits are zero
16    return r; // bitwise inverted byte.
17 }

```

The three bytes are important because they are used to keep track of the message history that ensures that no messages are sent multiple times once they've been acknowledged. In the code listing below is explained how this history is stored.

```

1 #define COMPARE_GT(a,b) ((int8_t)((a)-(b)) >= 0 )
2 floodSession flood_nodeHistory[256]= { 0 };
3
4 bool storeValue(uint16_t sourceId ,uint8_t sequenceNumber ,uint8_t packetLength){
5
6     // 0 is reserved for adress 0xAABB this is the Collector's source and will not fit as
7     // index between 0-255.
8     if(sourceId == 0xAABB){
9         sourceId = 0;
10    }
11    //acknowledgement is the only possible package with less then 10 bytes
12    if(packetLength < 10){
13        return storeAck(sequenceNumber);
14    }
15    if(flood_nodeHistory[sourceId].set == false){
16        flood_nodeHistory[sourceId].lastSequence = sequenceNumber;
17        flood_nodeHistory[sourceId].set = true;
18    }
19    else if(COMPARE_GT(sequenceNumber , flood_nodeHistory [ sourceId ]. lastSequence ))
20    {
21        //update sequence number in table
22        flood_nodeHistory [ sourceId ]. lastSequence=sequenceNumber;
23        flood_nodeHistory [ sourceId ]. acknowledged = false;
24    }
25    else{
26        //Message already seen
27        return false;
28    }
29    return true;
30 }

```

The function `storeValue` takes 3 parameters, the `sourceId` (or source address) is used as an index for the array to store the sequence number in. At last the `packetLength` will be used in order to check if the incoming message is an acknowledgement or not.

The `sequenceNumber` and `sourceId` are used to store a message by assigning a `floodsession` for all addresses. A `floodsession` is a struct, storing a boolean to see if the next message is the first message of that session. An integer which stores the last seen sequence number. The last field is also a boolean, which indicates if the message seen is acknowledged or not. The `storeValue` function checks if the message received is a unique message. This is done to compare the current sequence number with the sequence number stored in the array. If it is a unique message, the function will return true. Otherwise it will return false. Without this functionality, multiple repeaters inside a network would be constantly repeating the messages to each other and so blocking the frequency for any other message to be transported. This would be catastrophic for the communication between a node and a gateway.

`StoreValue` and `BitReverse` are used in the `echoCallback` described below.

```

1 static void echoCallback(RF_Handle h, RF_CmdHandle ch, RF_EventMask e)
2 {
3     uint8_t sequenceNumber;
4     uint16_t source;
5     //Clear the txPacket
6     memset(txPacket, 0, PAYLOAD_LENGTH);
7     if (e & RF_EventRxEntryDone)
8     {
9         /* Successful RX */
10        currentDataEntry = RFQueue_getDataEntry();
11
12        /* Handle the packet data, located at &currentDataEntry->data:
13         * - Length is the first byte with the current configuration
14         * - Data starts from the second byte */
15        packetLength = *(uint8_t *)(&(currentDataEntry->data));
16        packetDataPointer = (uint8_t *)(&(currentDataEntry->data));
17
18        RF_cmdPropTx.pktLen = packetLength;
19        //temporarily store data in object to retrieve source and sequencenumber
20        memcpy(txPacket, packetDataPointer, packetLength);
21        sequenceNumber = BitReverse(tempPacket[4]);
22        source = BitReverse(tempPacket[9]) | BitReverse(tempPacket[10]) << 8;
23
24        //check if sequence was seen already seen or not.
25        if ((storeValue(source, sequenceNumber, packetLength))) {
26            //Unique message. Repeat this message.
27            memcpy(txPacket, packetDataPointer, packetLength);
28        }
29        else {
30            //Message already seen. Don't send anything.
31            memset(txPacket, 0, packetLength);
32        }
33        RFQueue_nextEntry();
34    }

```

This callback, which was introduced earlier this chapter, handles the message received. The first step is to retrieve the data from the queue that was passed into the RX command. From this dataset, the sequence number, source address, `packetLength` and the pointer to the data are extracted. After the collection of the needed information, the application checks if the message already passed the repeater before. If that's not the case, the message will be copied into the `TxPacket` buffer. The callback ends with setting the active operation to the TX command (as explained in section 6.1.2) so that it can be executed afterwards. The message is now handled and ready to be sent again.

### 6.1.4 Transmit (TX)

The message is ready to be sent. All that needs to be done, is to assign the data buffer to the TX command, and set a random delay. This delay was implemented to prevent repeaters from sending messages at the exact same time. This would distort the signal. By waiting an x amount of time before transmitting, collision would become near impossible. At last the TX command containing the message and the random delay is sent to the RF module. Which transmits the message. How this is done is shown in the listing below.

```

1 RF_cmdPropTx.pPkt = txPacket;
2 RF_cmdPropTx.startTime = DelayRandom();
3 if(txPacket.length != 0){
4     //Send command to the handle, returns reason of termination (e.g. Success or Time-out).
5     RF_EventMask terminationReason =
6         RF_runCmd(rfHandle, (RF_Op*)&RF_cmdPropRx, RF_PriorityNormal, NULL,
7             (RF_EventRxEntryDone | RF_EventLastCmdDone));
8
9 }
10 //Reset txPacket for next iteration
11 memset(txPacket, 0,0);

```

After this code block, the process starts over, starting with receiving a message. The message is now properly repeated.

## 6.2 Deep sleep

The second part of the application is the deep sleep functionality which is implemented for a node in the network. The code shown in this section is developed for a sensor node. The sensor sends temperature data to a gateway every x seconds. after that time the node will enter deep sleep mode. This section will explain how this sleep mode is entered and how the application is woken up after a set period, without any user interaction. The code listed below shows how the deep sleep logic is configured during initialization.

```

1 /* Locals */
2 Task_Params taskParams;
3 Semaphore_Params semParams;
4
5 Power_init();
6
7 /* Construct clock for debounce */
8 Clock_Params clockParams;
9 Clock_Params_init(&clockParams);
10 clockParams.arg = (UArg)hButtons;
11 Clock_construct(&buttonClock, buttonClockCb, 0, &clockParams);
12 hButtonClock = Clock_handle(&buttonClock);
13 /* Configure task. */
14 Task_Params_init(&taskParams);
15 taskParams.stack = myTaskStack;
16 taskParams.stackSize = sizeof(myTaskStack);
17 Task_construct(&myTask, taskFxn, &taskParams, NULL);
18 /* Configure deepsleep semaphore. */
19 Semaphore_Params_init(&semParams);
20 semParams.mode = Semaphore_Mode_BINARY;
21 Semaphore_construct(&shutdownSem, 0, &semParams);
22
23 /* Start kernel. */
24 BIOS_start();

```



As the code on the previous page shows, The initialization is done by calling `Power_init`. This function initializes the Power Manager. The power manager makes it possible to use the power functions to access the power of the cc 1350. After this, the application initializes a clock, which will be used to wake the system up. The task created will execute the logic that the node should do when woken up. At last a semaphore is introduced, which is used to enter the deep sleep state. This is done because in order to shutdown all processes should be terminated. This semaphore will act as a gate for the shutdown. Now that the deep sleep is initialized, it is time to go to sleep. The code listing below shows how this is done.

## 6.2.1 Entering sleep

```

1 void shutdown() {
2     /* Turn on LED0 to indicate active */
3     PIN_setOutputValue(hPins, Board_PIN_LED0, 1);
4     /* Pend on semaphore before going to shutdown */
5     Semaphore_pend(Semaphore_handle(&shutdownSem), BIOS_WAIT_FOREVER);
6     /* Turn off LED0 */
7     PIN_setOutputValue(hPins, Board_PIN_LED0, 0);
8     /* Configure DIO for wake up from shutdown */
9     PINCC26XX_setWakeup(TimerTableWakeUp);
10    /* Go to shutdown */
11    Power_shutdown(0, 0);
12 }

```

At first, a led is set to indicate the system is active. This is the place to put the function call to the code a node should execute whenever it wakes up. for example send out a message. Since this is application specific, a led is used here instead. After this, the application waits on the semaphore. Once the semaphore is set (shown later) the application will continue. The led will power down indicating entering deep sleep. In a real application this space can be used to stop other processes. Next in line, a wake up mechanism is initialized. This is the place where the user should provide how the system is woken up. In this example a timer is used. How this code looks like is explained later. At last the application will enter deep sleep by calling `Power_shutdown`. From this point on the device only consumes minimal power +- 7µA. To wake the system up a debounce has to be generated (as explained in 3.3.1). The timer will generate an interrupt to wake up the chipset again. this process is done by the sensor controller as explained in the previous chapter.

## 6.2.2 Waking up

```

1 /*** initialization ***/
2 // Trigger the first iteration in 1 ms from now.
3 evhSetupTimer1Trigger(0, 1, 2);
4
5 /*** event handler ***/
6 // schedule next execution after 10 sec
7 evhSetupTimer1Trigger(0, 80, 9);
8 // generate interrupt();
9 fwGenAlertInterrupt();

```

the code above shows how this sensor controller code looks like. The clock runs at a 4KHz rate, which means that every second has 4096 ticks. The sensor controller `evhSetupTimer1Trigger` takes 3 arguments. The event index, in our case 0, because its the first and only event to be triggered. The other two arguments have to do with the delay. Since storage on the sensor controller is very limited, the function takes a mantissa and an exponent. These are inserted in the following function :  $mantissa * 2^{exponent} = ticks$  which means that with one unsigned 16 bits integer and one unsigned 8 bits integer, an delay of maximum 8355840 ticks can be generated. ( $255 * 2^{15} = 8355840$ ). The maximum delay has to be stored in an 32 bits integer, which is another byte more then the 3 bytes explained earlier.

So in order to generate a ten second delay, 80 is entered for the mantissa and 9 for the exponent which results in 40960 ticks. 40960 divided by 4096 (frequency) results in 10. After this delay an interrupt is generated which will trigger the wakeup sequence shown on the next page.

```
1 void scTaskAlertCallback(void) {  
2     if(!notFromShudown){ //booleans are by default false  
3         init();  
4         /* if the sensor takes longer than the debounce time, do not execute code again. */  
5         notFromShudown = true;  
6         /* Turn on LED0 to indicate active */  
7         PIN_setOutputValue(hPins, Board_PIN_LED0, 1);  
8         /* PUT YOUR ACTIVE CODE HERE */  
9         Semaphore_Post(Semaphore_handle(&shutdownSem),0);  
10    }  
11 }
```

The `scTaskAlertCallback` is a callback that listens to the interrupt generated by the sensor controller. The node now has to be initialized again, since it lost its memory when deep sleep was entered. after which the led is toggled on again indicating that the node has left the deep sleep state. The node specific code must be placed before `Semaphore_Post` since this will trigger the shutdown function. It makes sure that all processes are stopped and will power down the device again. The circle is complete. The node will initialize, do its job, go to sleep , wakes up, executes code and goes back to sleep again.

## 7 Conclusion

After reading this report the reader should now know that the project goals have been achieved. This goal was accomplished by doing the following things.

Research has been done on numerous of topics and results have been collected and processed. Knowledge about how communication between multiple TI CC 1350 MCU chipsets works is collected. The research also showed that deep sleep is achievable on the chipset and what it can do for the lifespan of an battery powered application. Finally, the research also showed that creating a mesh network is possible.

The core questions of this research were : “How are the functionalities Phact B.V. desires implemented in the other Operating Systems?” and “How can these functionalities be implemented in SimpleLink™.

The functionalities that Phact B.V. desired were deep sleep and mesh networking. Amber Wireless showed how meshing was possible and has been used to create a mesh network for the CC 1350. For the other functionality, deep sleep, no extra research was done because Texas Instruments documentation lead to the desired result.

These results have been used to create an application that is capable of repeating messages and so creating a mesh network. The repeater repeats not only messages but also acknowledgements which is an improvement in comparison to the Amber Wireless module since that module only repeats messages. The results of the research on deep sleep have been used to create an application that is capable of entering deep sleep, waking up, send a message and enter deep sleep again.

This means that the goal of this project has been reached. The elements Phact B.V. wanted are in place. The product works and is ready to be used by Phact B.V.

### 7.1 Discussion

This project has been quite busy. This project was started with zero knowledge in the domain. After a lot of reading of documentation and trial and error programming, the first small programs started to come alive. After that a lot of information became understandable and started to make sense. The research became significantly easier to do as the time passed. Due to such a rough start in the research phase not all the research is completed in time. On the other hand, During this research many hours were put into creating software that enabled the device to enter deep sleep. Plus the software that was necessary for the mesh network (a gateway, an end node and a sniffer). This means that some development time has been combined with the research phase.

Creating a mesh network was quite problematic. The first couple of weeks during the development phase were lost due to the fact that the application was living on a too high level. This resulted in a complete reset. This does not mean that these weeks are lost. Knowledge about how to use a sniffer was done, and the structure of such packages has been learned.

### 7.2 Future

For the rest of this internship the deep sleep application will be improved. As visible in the report the code is not completely finished yet, in order to give a proper demo, some extra work has to be spent on that.

## 8 Recommendations

This chapter provides the reader with the advice for future development and improvement of features.

### 8.1 Meshing

#### 8.1.1 Floodsession

At the moment, only 255 nodes can be connected to the mesh network (excluding repeaters). The current structure of the application can store for 255 different nodes whether a message is seen or not. this is enough for now, but if Phact B.V. wants to increase the amount of possible connections, the following is advised. Change the floodsession array to an hash table. That way more devices can be hooked up to the network and this will also decrease the lookup time for the last seen message.

#### 8.1.2 Performance

The way the repeater is implemented now, is with one RF core switching roles between sending and receiving. However when a message is received and being processed, the RF module is deaf. This potentially means that some messages are never repeated. To fix this, I would suggest to use 2 RF cores (for example by using 2 CC 1350 chipsets), one for receiving and one for transmitting. The receiving module would be listening all the time and post the received messages to a queue, which the transmitting core would pend on. This way no message will be missed.

### 8.2 Deep sleep

Calculation of the mantissa and exponent needed for the timer in the sensor controller is done by a program in C# and filled in by hand at the moment. For easier configuration I would advise to create a function that contains the following formula and calculates these two variables. the formulas regarding the variables are listed here.

$$exponent = NextPowOfTwo((input * frequency)/255)$$

$$mantissa = (input * frequency)/2^{exponent}$$

NextPowOfTwo should return the smallest power of two in which the result of the fraction fits. for example 15.6 will fit in  $2^4$  (16) but 16.1 will only fit in  $2^5$  (32).

## Bibliography

- [AmWi17] Amber Wireless (2017) *AMB8826 Manual* [online] Available at: <https://www.amber-wireless.com/en/amb8826.html> [Accessed 9-05-2018]
- [Cont18] Contiki-os (2018) *Contiki* [online] Available at: <http://www.contiki-os.org/start.html#start-cooja> [Accessed 20-02-2018]
- [DeSl09] Youtube (2009) *What is Deep Sleep* Available at: <https://www.youtube.com/watch?v=3wyGUmMnFs> [Accessed 07-03-2018]
- [MXDS17] macronix (2017) *MX25R8035F* [online] Available at: <http://www.macronix.com/Lists/Datasheet/Attachments/6780/MX25R8035F,%20Wide%20Range,%208Mb,%20v1.6.pdf> [Accessed 07-03-2018]
- [SiAO17] Texas Instruments (2017) *Application overview* [online] Available at: [http://dev.ti.com/tirex/content/simplelink\\_cc13x0\\_sdk\\_1\\_30\\_00\\_06/docs/ti154stack/ti154stack-sdg/ti154stack-sdg/Application%20Overview.html](http://dev.ti.com/tirex/content/simplelink_cc13x0_sdk_1_30_00_06/docs/ti154stack/ti154stack-sdg/ti154stack-sdg/Application%20Overview.html) [Accessed 12-03-2018]
- [SiDs17] Texas Instruments (2017) *datasheet CC 1350* [online] Available at: <http://www.ti.com/lit/ds/symlink/cc1350.pdf> [Accessed 06-03-2018]
- [SiSO17] Texas Instruments (2017) *15.4 stack overview* [online] Available at: [http://dev.ti.com/tirex/content/simplelink\\_cc13x0\\_sdk\\_1\\_50\\_00\\_08/docs/ti154stack/ti154stack-users-guide/ti154stack/ti154stack-overview.html](http://dev.ti.com/tirex/content/simplelink_cc13x0_sdk_1_50_00_08/docs/ti154stack/ti154stack-users-guide/ti154stack/ti154stack-overview.html) [Accessed 12-03-2018]
- [Taig18] Taiga (2018) *Love your project* [online] Available at: <https://taiga.io/> [Accessed 06-02-2018]
- [TICC17] Texas Instruments (2017) *CC 1350* [online] Available at: <http://www.ti.com/product/CC1350> [Accessed 06-03-2018]
- [Phac17] Phact B.V. (2017) *Over Phact* [online] Available at: <https://www.phact.nl/en/about-phact.php> [Accessed 07-03-2018]
- [WiKH18] Wikipedia (2018) *Kirchhoff's circuit laws* [online] Available at: [https://en.wikipedia.org/wiki/Kirchhoff%27s\\_circuit\\_laws](https://en.wikipedia.org/wiki/Kirchhoff%27s_circuit_laws) [Accessed 02-04-2018]
- [WiCR17] Wikipedia (2017) *CR 2032* [online] Available at: [https://nl.wikipedia.org/wiki/CR\\_2032](https://nl.wikipedia.org/wiki/CR_2032) [Accessed 05-03-2018]
- [WiMN18] Wikipedia (2018) *Mesh networking* [online] Available at: [https://en.wikipedia.org/wiki/Mesh\\_networking](https://en.wikipedia.org/wiki/Mesh_networking) [Accessed 11-04-2018]

# Appendix A SRS

## Introduction

In this chapter the Software requirements for the SimpleLink™ expansion will be introduced. Throughout this chapter the purpose of this document will be explained, and the scope defined. Furthermore will there be a list with all abbreviations, acronyms will be named and Definitions explained. After that the references will be addressed and at the end there will be an overview of this Software requirement specification.

## Purpose

The purpose for this Software Requirements Specification is to outline the requirements for the SimpleLink Expansion for Phact B.V. This document should present a detailed description of the application mentioned. It will explain the purpose and features of this system, what the system should do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to Johan Kleuskens for its approval.

## Document Conventions

This document will be aligned following certain conventions these are mentioned here.

Text between “<” and “>” defines the level of priority of the requirement defined after the “>” sign with a range of low, medium and high. For example a requirement with <High> as prefix will be a requirement with a high priority.

Every requirement will be numbered for traceability. Traceability is the ability to verify the history or location of a requirement. It enables users to find the origin of each requirement and track every change that was made to this requirement. For this purpose, it may be necessary to document every change made to the requirement.

## Intended Audience and Reading Suggestions

This document is meant for the Research and Development department of Phact B.V., Developer Herm Lecluse, Teacher Marco Langenhuizen, Assessor Geert Monsieur and Domain Expert Geertjan Wielenga.

## Project Scope

The software delivered by Texas Instruments is not complete, The layer to be designed should incorporate these functionalities into the system. This will enlarge the capabilities of the system. These functionalities are : Deep sleep and Mesh (flooding) network.

## Overall Description

This section of the SRS describes the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in detail in Section 4 of the SRS, and makes them easier to understand.

## Product Functions

1. < *High* > The software layer should be able to toggle the chipset into different states.
2. < *Medium* > The software layer shall be able to run on any TI CC 13x0 device.
3. < *High* > The software layer shall be able to transmit messages over a sub 1GHz network (IEEE\_802.15.4)
4. < *High* > The user will be able to use this layer to increase the functionalities of the chipset.

## User Classes and Characteristics

The user of this software layer is going to be either a hardware engineer or a software engineer. Both of these users are going to use this layer to increase the possibilities of Texas Instruments CC 1350 MCU and TI-RTOS.

## Operating Environment

5. < *High* > The system must be able to handle at least one connection (between node and gateway).
6. < *High* > The code will be tested.
7. < *High* > A prototype for the software layer will be created.

## User Documentation

7. < *High* > The code shall be documented, as well inline as per method.
8. < *Low* > A user manual describing how to use the software layer shall be made.

## Assumptions and Dependencies

- Assuming that the system will run in an environment where the sub 1 GHz band is not fully occupied.
- Assuming that the user has enough knowledge about the environment to work with this layer.

## Constraints

- The gateway and node need to get sufficient power.
- The area surrounding the gateway/node should allow the chipset to transmit data.

## External Interface Requirements

### Hardware Interfaces

The software layer will run on the Texas Instruments CC 1350 MCU Launchpad. Which allows easy GPIO access. The software layer will communicate with the chipset.

### Software Interfaces

- The application will make use of SimpleLink™.
- The application will be developed in C.

## System Features

This chapter illustrates the functional requirements for the product by system features, the major services provided by the product.

### Deep sleep

#### Description and priority

The software expansion shall be able to provide deep sleep functionality for the TI CC 1350 Launchpad.

#### Functional Requirements

[Req 1.1 < *High* >] The software layer shall be able to toggle deep sleep.

This is a necessity since this is one of the main purposes of the product.

[Req 1.1.1 < *High* >] The product can enter deep sleep.

[Req 1.1.2 < *High* >] The product can wake itself up.

these two requirements describe the functionality of the product.

### Meshing

#### Description and Priority

The software layer shall be able to setup a mesh (flooding) network.

#### Functional Requirements

[Req 1.2 < *High* >] The software layer will be able to act as a mesh network.(multiple devices)

[Req 1.2.1 < *High* >] The software layer will be able to act as a repeater, gateway or node.

[Req 1.3 < *Medium* >] The application will not be able to enter deep sleep if the chipset is configured to be a gateway or repeater.



## Other Nonfunctional Requirements

### Software Development environment

[Req 2.1 < *High* >] The software will be developed inside Texas Instruments Code Composer Studio.

### Performance Requirements

[Req 6.1 < *High* >] The network shall be able to handle at least one connection at a time.

### Prototype

[Req 5.1 < *Medium* >] The prototype will be an Texas Instruments CC 1350 Launchpad.

[Req 5.2 < *High* >] The prototype will show all functionalities described earlier.

### Software Quality Attributes

[Req 8.1 < *High* >] The code will be tested using integration tests.

[Req 8.2 < *High* >] The system will be tested by deploying the solution to the micro controller.

# Appendix B Project plan

## **SimpleLink Expansion**

Bachelor Thesis - Project Plan

Submitted by Herm Lecluse

In fulfilment of the requirements for the degree  
Bachelor of Science  
To be awarded by the  
Fontys University of Applied Sciences

Venray, March 2, 2018

# 1 Introduction

Phact is a 'one stop shop' for software and hardware solutions and secondment. Phact was founded in 2015 with nearly 25 years of experience in the telecommunication branch.

How? Phact originated when a group of people with loads of experience decided to focus on a new horizon. Many of Phact's employees have worked in the telecommunication sector for years and took their expertise to this new endeavour. Phact was born.

People in the team have been developing telecom hardware and software for VoIP and SS7 carriers since 1991. Knowledge is a key part of the business. Therefore, Phact is engaged in close partnerships with our professional partners and suppliers.

Phact might still be very young, but they carry years of experience in software and hardware development.

Currently, is Phact B.V. looking into new possibilities for an existing chip set that has been created by Texas Instruments. This chip set is capable of transmitting data over a sub 1 GHz frequency.

This report is a representation of the project plan. This paper will inform the reader about what problem is at hand and how this project should solve that problem. What deliverables this project has and how these deliverables should be achieved. Stakeholders will be introduced and the current version of the risk analysis will be explained.

## 2 Project Plan

### 2.1 Context

Phact B.V. is missing functionalities inside Texas Instruments operating system (OS) which is called SimpleLink™. The biggest flaw in this OS is the fact that it only supports star networks. Imagine a node that is too far away from its gateway but in range of a second node. In the current situation this would mean that the first node is impossible to send its data. What if this first node would be able to transmit its data through the second node to the gateway. This feature is called flooding (meshing) and essentially means that the first node still can transfer its data to the gateway without being in range.

Another flaw in SimpleLink™ is the support for deep sleep is very complicated and looks devious. Phact wants this simplified. So that applications use less excessive power and so enlarge the time between battery charge cycles.

Contiki, a generic open source operating system that provides the missing functionalities exists, but contiki is not updated regularly neither is it supported by Texas Instruments (TI). This is a problem for Phact B.V., as they sell applications which should be maintainable from all over the globe. Whenever Texas Instruments pushes an update for SimpleLink™ Phact B.V. must know that their application is still up and running and does not depend on a not supported framework.

Amber Wireless is a manufacturer and has made the AMB8826 chipset, which encapsulates the TI CC 1310 module ( which is the same module as the CC 1350 except it has no bluetooth connectivity ). Amber Wireless extended the SimpleLink™ operating system to make it easier to use. The downside of this AMB8826 is that there is no Input/Output (I/O) compatibility for sensors / actuators which Phact B.V. does desire.

### 2.2 Lead-up to the assignment

Phact B.V. has applications which run on the CC 1350 chip set and experience that the missing features in the SimpleLink™ described above, are limiting the possibilities of this module. Phact B.V. would like to use this technology for more than the supported OS offers. So that their applications would be more reliable and sustainable in practice.

### 2.3 Goal

The main goal of the commissioning company is to be able to extend usability of the Texas Instruments CC1350 micro controller unit. This opens up opportunities for Phact B.V. to create more applications using this device.

## 2.4 Expected results

- A software layer for the Texas Instruments CC 1350 microcontroller unit (MCU) on top of SimpleLink™ that provides functionality for the MCU so that it is able...:
  - To communicate inside a mesh network.
  - To enable deep sleep for an application.
  - To update the firmware over the air.
- Research report on how to implement the functionalities above.

## 2.5 Scope

Since there are 3 different operating systems, 2 of which have some of the desired functionalities, a research will be done into Contiki and Amber Wireless. on how to implement these functionalities for SimpleLink™.

This project will cover a software layer for the SimpleLink™, that will be tested and prototyped. This project will not cover software development for SimpleLink™, Contiki or Amber Wireless.

## 2.6 Program requirements

The software layer has to enable the Texas Instruments CC 1350 MCU nodes to communicate inside a (flood-ing) mesh network. This is important because there are cases in which a node could lose its direct connection with a gateway. If this happens the node should first look for a new gateway to connect to, if there is none, try to communicate through another device that is in range of a gateway.

The software layer should also provide functionality for deep sleep. This way it is possible for devices to save energy and so save maintenance costs. (replacing batteries). It is close to impossible to have both these functionalities active at the same time, so the program should be able to toggle between these modes.

Update over the air is important as well. In order to update the applications which are running on devices potentially across the entire world. This is especially interesting for devices which are in the deep sleep modus since the devices are offline once in deep sleep they have to be awake when an update gets pushed to these devices.

## 2.7 Project phases

This project is built from different phases. These phases are being explained in this section.

### 2.7.1 Plan

In this phase the first version of this document was created. This phase has also been used to initialize the project, to get used to the new work environment and get in contact with the new colleagues.

### **2.7.2 Research**

In this phase, research into the to be developed system will be done. Existing sources will be analysed. The main question that this research should answer is: “How are the functions implemented in the other operating systems?”.

The steps of this research are:

- Research in mesh networking and implementations
- Research in deep sleep implementations
- Research in SimpleLink™
- Research in Texas Instruments CC 1350 microcontroller unit

### **2.7.3 Development**

During this phase, the application will be developed in a test driven manner. The end report will be created during this phase as well.

## **2.8 Deliverables**

This project has several deliverables, in this section these deliverables are described.

- Project plan
- Research paper
- Mid term report
- Prototype
- Bachelor Thesis

In the next section these deliverables are used to define the activities for this project.

## 2.9 Activities

This section will show the reader what activities this project has.

Activity	Project phase
Creation of Project plan	Planning phase
Delivery of Project plan	Planning phase
Implement feedback on Project plan	Planning phase
Research	Research phase
Report on research	Research phase
Creation of mid-term report	Research phase
Delivery of mid-term report	Research phase
Implement feedback on mid-term report	Research phase
Creation of Software Layer	Development phase
Testing the prototype	Development phase
Delivery of product	Development phase
Creation of report	Development phase
Presenting report and the prototype	Development phase
Handover	Development phase

Table 2.1: Table of project activities

## 2.10 Project management

### 2.10.1 Time

To be sure that the project is feasible in the given time milestones have been defined with a due date bound to it. These milestones are explained later on in this document. The milestones are set up in Taiga, which is a project management platform for agile development. In Taiga, the supervisor can track the progress of the project. The development phase shall be done in an agile way because the internship supervisor wants to be able to track the progress that has been made based on functionalities that were implemented. An agile approach is what suits best for this project.

### 2.10.2 Quality

In order to deliver a product of high quality and expectation of the supervisor, weekly meetings will be planned to discuss the current progress. The supervisor will review the developed and tested software in order to make sure that the functions meet his expectations. At the end of each milestone a complete review for that milestone will be done for all functions in that system.

## 2.11 Risk analysis

This project has several risk factors. If those one of those risk events happen, than such a risk could endanger the progress of the project. These risks have been analysed and listed here:

- It turns out that SimpleLink™ prohibits a certain feature.

**Consequence:** This function cannot be built for the desired application.

**Solution:** Plan a meeting with the supervisor to check for a fitting solution.

**Responsible:** Intern.

- Texas Instruments no longer supports the CC 1350 microcontroller unit

**Consequence:** The integrated development environment (IDE) of TI can no longer be used

**Solution:** Check if there is an alternative chipset available which is supported by TI or another manufacturer.

**Responsible:** Texas Instruments.

If one of these risks occur, a meeting will be planned with (some of) the stakeholders to look if a named solution is possible. if not an alternative will be discussed.



## 2.12 Project Organization

### 2.12.1 Stakeholders

This project has several stakeholders who need to be kept informed on the status of the project. Those stakeholders have a level of interest and a level of power. Those are displayed in the table below.

Stakeholder	Level of interest	Level of power
Marco Langenhuizen (teacher)	Medium	High
Johan Kleuskens (supervisor)	High	High
Herm Lecluse (intern)	Very High	Medium

Table 2.2: Table showing the stakeholders of this project

### 2.12.2 Communication

To ensure that the stakeholders get information about the status of this project, there will be frequent contact with them. This contact may differ from stakeholder to stakeholder. For example, the teacher will be informed via mail and reports. This teacher will also visit Phact B.V. minimal once. The supervisors will be informed weekly via meetings and will receive a copy of each report as well.

## 2.13 Milestones

This section will define the milestones for this project. A list will be shown in which is explained when the intern expects to finish these milestones.

Id	Milestone	Due date
001	Project plan	19-02-2018
101	research document	19-03-2018
201	Mid-term report	23-03-2018
202	Mid-term presentation	23-03-2018
301	Meshing functionality	06-04-2018
302	Deep sleep functionality	20-04-2018
303	Over the air updating	04-05-2018
304	Combining the functions	01-06-2018
401	Final report	08-06-2018
402	Presentation	17-06-2018
501	Handover	01-07-2018

# Appendix C Midterm report

## **SimpleLink Expansion**

Bachelor Thesis - Mid Term Report

Submitted by Herm Lecluse

In fulfilment of the requirements for the degree  
Bachelor of Science  
To be awarded by the  
Fontys University of Applied Sciences

Venray, March 27, 2018

# 1 Introduction

Phact is a 'one stop shop' for software and hardware solutions and secondment. Phact was founded in 2015 with nearly 25 years of experience in the telecommunication branch. [Phac17]

How? Phact originated when a group of people with loads of experience decided to focus on a new horizon. Many of Phact's employees have worked in the telecommunication sector for years and took their expertise to this new endeavour. Phact was born.

People in the team have been developing telecom hardware and software for VoIP and SS7 carriers since 1991. Knowledge is a key part of the business. Therefore, Phact is engaged in close partnerships with our professional partners and suppliers.

Phact might still be very young, but they carry years of experience in software and hardware development.

Currently, is Phact B.V. looking into new possibilities for an existing chip set that has been created by Texas Instruments. This chip set is capable of transmitting data over a sub 1 GHz frequency.[TICC17]

This report is a representation of the project plan. This paper will inform the reader about what problem is at hand and how this project should solve that problem. What deliverables this project has and how these deliverables should be achieved. Stakeholders will be introduced and the current version of the risk analysis will be explained.

## 2 Project Plan

### 2.1 Context

Phact B.V. is missing functionalities inside Texas Instruments micro controller platform (SimpleLink™) operating system (OS). The biggest flaw in this OS is the fact that it only supports star networks. Imagine a node that is too far away from its gateway but in range of a second node. In the current situation this would mean that the first node is impossible to send its data. What if this first node would be able to transmit its data through the second node to the gateway. This feature is called meshing and essentially means that the first node still can transfer its data to the gateway without being in range. There are multiple implementations of meshing, in this project a flood mesh will be implemented.

Another flaw in SimpleLink™ is the support for deep sleep is very complicated and looks devious. Phact wants this simplified. So that applications use less excessive power and so enlarge the time between battery charge cycles.

Contiki[Cont18], a generic open source operating system that provides the missing functionalities exists, but contiki is not updated regularly neither is it supported by Texas Instruments (TI). This is a problem for Phact B.V., as they sell applications which should be maintainable from all over the globe. Whenever Texas Instruments pushes an update for SimpleLink™ Phact B.V. must know that their application is still up and running and does not depend on a not supported framework.

Amber Wireless is a manufacturer and has made the AMB8826 chipset, which encapsulates the TI CC 1310 module ( which is the same module as the CC 1350 except it has no bluetooth connectivity ). Amber Wireless extended the SimpleLink™ platform to make it easier to use. The downside of this AMB8826 is that there is no Input/Output (I/O) compatibility for sensors / actuators which Phact B.V. does desire.

### 2.2 Lead-up to the assignment

Phact B.V. has applications which run on the CC 1350 chip set and experience that the missing features in the SimpleLink™ described above, are limiting the possibilities of this module. Phact B.V. would like to use this technology for more than the supported OS offers. So that their applications would be more reliable and sustainable in practice.

### 2.3 Goal

The main goal of the commissioning company is to be able to extend usability of the Texas Instruments CC1350 micro controller unit. This opens up opportunities for Phact B.V. to create more applications using this device.

## 2.4 Expected results

- A software layer for the Texas Instruments CC 1350 microcontroller unit (MCU) on top of SimpleLink™ that provides functionality for the MCU so that it is able...:
  - To communicate inside a mesh network.
  - To enable deep sleep for an application.
  - To update the firmware over the air.
- Research report on how to implement the functionalities above.

## 2.5 Scope

Since there are 3 different operating systems, 2 of which have some of the desired functionalities, a research will be done into Contiki and Amber Wireless. on how to implement these functionalities for SimpleLink™.

This project will cover a software layer for the SimpleLink™, that will be tested and prototyped. This project will not cover software development for SimpleLink™, Contiki or Amber Wireless.

## 2.6 Program requirements

The software layer has to enable the Texas Instruments CC 1350 MCU nodes to communicate inside a (flooding) mesh network. This is important because there are cases in which a node could lose its direct connection with a gateway. If this happens the node should first look for a new gateway to connect to, if there is none, try to communicate through another device that is in range of a gateway.

The software layer should also provide functionality for deep sleep. This way it is possible for devices to save energy and so save maintenance costs. (replacing batteries). It is close to impossible to have both these functionalities active at the same time, so the program should be able to toggle between these modes.

Update over the air is important as well. In order to update the applications which are running on devices potentially across the entire world. This is especially interesting for devices which are in the deep sleep modus since the devices are offline once in deep sleep they have to be awake when an update gets pushed to these devices.

## 2.7 Project phases

This project is built from different phases. These phases are being explained in this section.

### 2.7.1 Plan

In this phase the first version of this document was created. This phase has also been used to initialize the project, to get used to the new work environment and get in contact with the new colleagues.

## **2.7.2 Research**

In this phase, research into the to be developed system will be done. Existing sources will be analysed. The main question that this research should answer is: “How are the functions implemented in the other operating systems?”.

The steps of this research are:

- Research in mesh networking and implementations
- Research in deep sleep implementations
- Research in SimpleLink™
- Research in Texas Instruments-Real Time Operating System
- Research in Texas Instruments CC 1350 microcontroller unit

## **2.7.3 Development**

During this phase, the application will be developed in a test driven manner. The end report will be created during this phase as well.

## **2.8 Deliverables**

This project has several deliverables, in this section these deliverables are described.

- Project plan
- Research paper
- Mid term report
- Prototype
- Bachelor Thesis

In the next section these deliverables are used to define the activities for this project.

## 2.9 Activities

This section will show the reader what activities this project has.

Activity	Project phase
Creation of Project plan	Planning phase
Delivery of Project plan	Planning phase
Implement feedback on Project plan	Planning phase
Research	Research phase
Report on research	Research phase
Creation of mid-term report	Research phase
Delivery of mid-term report	Research phase
Implement feedback on mid-term report	Research phase
Creation of Software Layer	Development phase
Testing the prototype	Development phase
Delivery of product	Development phase
Creation of report	Development phase
Presenting report and the prototype	Development phase
Handover	Development phase

Table 2.1: Table of project activities

## 2.10 Project management

### 2.10.1 Time

To be sure that the project is feasible in the given time milestones have been defined with a due date bound to it. These milestones are explained later on in this document. The milestones are set up in Taiga[Taig18], which is a project management platform for agile development. In Taiga, the supervisor can track the progress of the project. The development phase shall be done in an agile way because the internship supervisor wants to be able to track the progress that has been made based on functionalities that were implemented. An agile approach is what suits best for this project.

### 2.10.2 Quality

In order to deliver a product of high quality and expectation of the supervisor, weekly meetings will be planned to discuss the current progress. The supervisor will review the developed and tested software in order to make sure that the functions meet his expectations. At the end of each milestone a complete review for that milestone will be done for all functions in that system.

## 2.11 Risk analysis

This project has several risk factors. If those one of those risk events happen, than such a risk could endanger the progress of the project. These risks have been analysed and listed here:

- It turns out that SimpleLink™ prohibits a certain feature.

**Consequence:** This function cannot be built for the desired application.

**Solution:** Plan a meeting with the supervisor to check for a fitting solution.

**Responsible:** Intern.

- Texas Instruments no longer supports the CC 1350 microcontroller unit

**Consequence:** The integrated development environment (IDE) of TI can no longer be used

**Solution:** Check if there is an alternative chipset available which is supported by TI or another manufacturer.

**Responsible:** Texas Instruments.

If one of these risks occur, a meeting will be planned with (some of) the stakeholders to look if a named solution is possible. if not an alternative will be discussed.



## 2.12 Project Organization

### 2.12.1 Stakeholders

This project has several stakeholders who need to be kept informed on the status of the project. Those stakeholders have a level of interest and a level of power. Those are displayed in the table below.

Stakeholder	Level of interest	Level of power
Marco Langenhuizen (teacher)	Medium	High
Johan Kleuskens (supervisor)	High	High
Herm Lecluse (intern)	Very High	Medium

Table 2.2: Table showing the stakeholders of this project

### 2.12.2 Communication

To ensure that the stakeholders get information about the status of this project, there will be frequent contact with them. This contact may differ from stakeholder to stakeholder. For example, the teacher will be informed via mail and reports. This teacher will also visit Phact B.V. minimal once. The supervisors will be informed weekly via meetings and will receive a copy of each report as well.

## 2.13 Milestones

This section will define the milestones for this project. A list will be shown in which is explained when the intern expects to finish these milestones.

Id	Milestone	Due date
001	Project plan	19-02-2018
101	research document	19-03-2018
201	Mid-term report	23-03-2018
202	Mid-term presentation	23-03-2018
301	Meshing functionality	06-04-2018
302	Deep sleep functionality	20-04-2018
303	Over the air updating	04-05-2018
304	Combining the functions	01-06-2018
401	Final report	08-06-2018
402	Presentation	17-06-2018
501	Handover	01-07-2018

## 3 Research and results

In this chapter the research that has been done will be discussed. The methods used will be addressed and results will be shown. The rationale for choices will be explained here as well.

### 3.0.1 Overall decisions

In this subsection a few decisions will be defined that were prerequisite for this project.

- The software layer to be created shall be created in C. This choice was made due to the fact that the SimpleLink™ is also written in C. This makes extending the operating system easier.
- The software layer to be created will be designed for the Texas Instruments CC 1350 Launchpad. This is a demo kit for the Texas Instruments CC 1350 MCU. This enables easy debugging of the micro controller and easy General Purpose Input/Output (GPIO) access to the developer.
- The software will be stored in a Git repository on Phact's GitLab. This way the software is backed up to prevent loss of data.

### 3.0.2 Questions

The (sub)questions regarding this research are listed here.

- What is SimpleLink™
- What is TI-RTOS?

How do we communicate between multiple devices on the sub 1 GHz band?

- What is meshing?
- What is deep sleep?

What is the effect of deep sleep on the power consumption?

What is the impact of deep sleep on maintenance costs?

Is deep sleep worth implementing?

### 3.0.3 Methods

The methods that have been used for this research are literature research and an experiment. Reading and learning how to use code has been the main activity during this phase. The second activity was the little experiment on deep sleep.

## 3.1 SimpleLink

SimpleLink is an micro controller platform developed by Texas Instruments that includes the broadest portfolio of wired and wireless Arm® MCUs[SiAO17] (System-on-Chip) in a single software development environment.

- Customer Application Code
- SDK Plug-ins
- The SimpleLink MCU SDK

The first part is the custom application that the MCU should execute. This is your own code. The rest is handled by TI in the next two layers.

The SimpleLink Software Development Kit (SDK) is extensible through SDK Plug-ins, modular software that adds support for external components and software. These plug-ins help the developer implement new hardware to the system.

The third part is the actual SDK provided by TI to help developers create new software. It contains drivers for hardware, communication stack's and code examples. Next to these functionalities, the SDK also contains the OS Kernel for TI-RTOS. This is the Operating system that the application will run on. Texas Instruments-Real Time Operating System is the core of SimpleLink™. This section will explain how communication is done between a gateway and one or multiple nodes.

### 3.1.1 Communication

Communication means that two or more devices are exchanging messages with each other. in this section a closer look will be shown to how this is done in the TI-15.4 [SiAO17] stack of TI-RTOS.

This stack is an implementation of the IEEE 802.15.4 standard. Which is a standard for low rate wireless personal area networks (low data rate but very long battery life and very low complexity applications). this stack defines the first two layers of the OSI model. The Data-link (layer 2) will from now on be called: Medium Access Control (MAC). This is how Texas Instruments defined the data link layer. this stack communicates through the Indirect Call Framework (ICALL) module with the logical layer and that layer communicates with the application the developer created. The architecture looks like this:

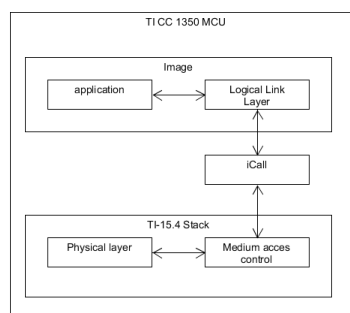


Figure 3.1: TI CC 1350 architecture

## Connection

A connection cannot be made without initializing the Gateway first. In order to do that, TI-RTOS physical layer resets the MAC. After this the application requests the MAC to scan for the channel with the least amount of radio interference. this returns a list with channels. These channels will then be scanned on active networks. This returns the best channel for the gateway to start its network. After this the gateway will setup its Personal Area Network (PAN) and open its network. From this point on the MAC will send out a beacon every x seconds for nodes that are trying to access te network. This is visualised in the sequence diagram on the next page.:

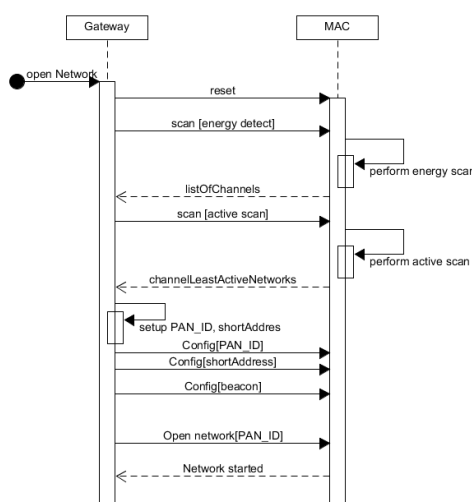


Figure 3.2: circuit scheme

Now that the gateway is up and running a node can join the network. A node would first also reset its MAC layer. The node will then scan for active beacons. Once it finds a beacon the node will configure itself to that PAN. and syncs to that beacon. To acquire beacon synchronization, the device calculates the time difference between the two beacons and uses that time to synchronize itself. From this point on, a connection has been established. The node gets an index number from the gateway to identify itself in.

## Transmission

Once the connection is established. the node can transmit packages over the network. each message has a sequence number which starts counting from the moment that the node connects to the network. sending a message from node to sensor looks like this:

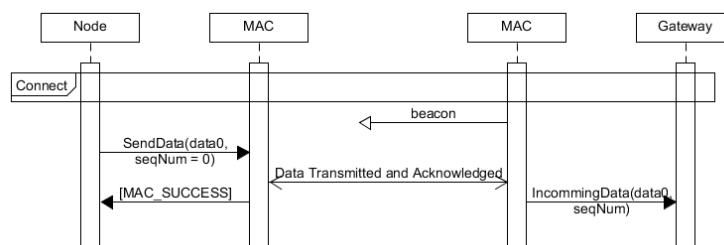


Figure 3.3: transmission

In the other direction the process works the same way except the gateway sends the message and the node receives it. If the node is not in range of the network any more, the data will get lost.

## Disconnect

If a node wants to disconnect from the network, it will send a final message to say 'goodbye'. this will get acknowledged and the session (bound to the ID given upon connection) gets closed. If the gateway closes the network, the node will receive a message that the node is no longer part of that network.

## 3.2 Deep sleep

This section will inform the reader about the research that has been done in deep sleep.[DeSl09] The section will

answer the following questions. What is the effect of deep sleep on the power consumption? What is the impact of deep sleep on maintenance costs? And is deep sleep worth implementing? In order to answer these questions a small experiment was prepared and executed. For the last question, 'What is deep sleep?', literature and content research was used to find the answer.

Deep sleep is a state a MCU that completely shuts down power consumption of a micro controller. This essentially means that the chipset limits its power consumption to just 185 nA. (0.2 $\mu$ A) [SiDs17] In this state no GPIO, connectivity or any other function of the MCU can be used. In order to use them again the chipset 'wakes up' from his deep sleep state in which the functions regain their usability.

The main purpose of this experiment is to check the effect of deep sleep, and to proof why deep sleep is useful. The following circuit was created:

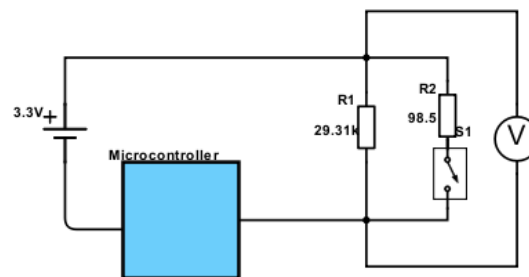


Figure 3.4: circuit scheme

As you can see, there is a switch after R2, this resistor is in place just in order to enable the system to boot. once the switch is closed the total resistance is 98.17 $\Omega$ . when the device goes into deep sleep the switch is toggled and the resistance changes to 29.31k $\Omega$ .

This is done because the volt meter is not able to measure the voltage that is running over the circuit with a resistor of 98.17 $\Omega$ .

By following the Ohm's law, the amount of volts is equal to the amount of resistance times the current. So by increasing the resistance, the voltage is increased and so measurable by the voltmeter. By using the values measured from the voltmeter and the given resistance values, the current can be calculated. This is

done in the table below.

	Idle	Sleep
U	0.391V	0.223V
R	98.5 $\Omega$	29.31k $\Omega$
I	3.97mA	7.61 $\mu$ A

Table 3.1: Table displaying measured values

Table 3.1 shows that there is a big difference in current when the micro controller is idle or in deep sleep. in fact it is almost 525x less when the chipset is in deep sleep.

If we compare that value to what the datasheet told us, a difference can be seen in consumption. As 7.61 $\mu$ A was measured and the datasheet shows this should be approximately 0.2 $\mu$ A. This difference can be explained by the flash memory that is on the launchpad and is still powered on. This unit uses 5 $\mu$ A typically[MXDS17]. This leaves us with 2.61 $\mu$ A, a 2.41 $\mu$ A difference. This difference can be explained by looking at the accuracy of the measuring equipment, a multimeter is not 100% accurate. Since the value is so small, it's quite likely to measure a wrong value.

for the rest of the experiment, the value 7 $\mu$ A is used, due to the fact that this is the actual consumption when in deep sleep.

Imagine that the micro controller is powered by a button cell[WiCR17] battery (or watch battery) with a capacity of 225mAh. This means that a TI CC 1350 chipset can run for 56 hours on such a battery when idle. However when in deep sleep, this chipset can run for approximately 3.4 years. This result shows that being able to toggle deep sleep for the TI CC 1350 can cut in maintenance costs quite a lot.

### 3.3 Mesh networking

This part of the research has not yet been done due to other parts of the research taking longer then expected.

## 4 Software Requirement Specification

In this chapter the requirements of the system will be explained. A Software Requirement is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software.

The requirements this project has, are listed below.

### 4.1 Functional Requirements

1. The software layer should be able to toggle the chipset into different states.
2. The software layer shall be able to run on any TI CC 13x0 device.
3. The software layer shall be able to transmit messages over a sub 1GHz network (IEEE\_802.15.4)
4. The user will be able to use this layer to increase the functionalities of the chipset.
5. A prototype for the software layer will be created.

### 4.2 4.2 Non-functional requirements

6. The system must be able to handle at least one connection (between node and gateway).
7. A user manual describing how to use the system shall be made.
8. The code will be tested.
9. The code shall be documented, as well inline as per method.

These requirements are high level requirements for more details the whole software requirement specification can be found in appendix A. These requirements have been discussed with the Project supervisor.

## **5 Conclusion**

After reading this report the reader should have gotten an impression on what has been going on for the last couple of weeks. a brief collection of activities will be shown here, The project plan has been created, reviewed and feedback has been implemented. Research has been done on numerous of topics and results have been collected and processed. Knowledge about how communication between multiple TI CC 1350 MCU chip sets is done is collected. The research also showed that deep sleep is achievable on the chipset and what it can do for the lifespan of an battery powered application.

### **5.1 Discussion**

This project has been quite busy. This project was started with zero knowledge in the domain. After a lot of reading of documentation and trial and error programming, the first small programs started to come alive. After which a lot of information became understandable and started to make sense. The research became significantly easier to do as the time passed. Due to such a rough start in the research phase not all the research is completed. On the other hand, during this research many hours were put into creating software that was necessary to do the research. This means that some development time can be combined with the research phase. (prototyping)

### **5.2 Future**

For the rest of this internship the research will be finished and the software layer will be developed, once the meshing research has concluded. The research phase continues for another week. After which designing and development for the final product can start. The research showed promising results.