INTERN REPORT HRO - TI

November 11, 2019

Paul Wondel
Stud.nr.: 0947421
Internship period: 1 September 2019 - 7 Februari 2020
First Attempt

Hogeschool Rotterdam
Technische Informatica
Rotterdam University of Applied Sciences
Applied Computer Science

Teachers: G.W.M. van Kruining, R. van Doorn
Intern Supervisor: A.C. van Rossum

Glossary

- **BLE** - Bluetooth Low Energie
- **Crownstone** - The smart device created by Crownstone B.V.
- **JSON** - JavaScript Object Notation: A extension file type
- **OS** - Operating System
- **CLI** - CommandLine Interface
- **pip** -
- **git** -
- **wget** -
- **npm** - npm is a package manager for the JavaScript programming language
- **CPU** - Computer Processing Unit

Contents

1 Introduction

In my third year of education I am required to do an internship at a IT company. In this report I will write about my experiences during the internship and the process of my assignment. The company where I did the internship is named Crownstone B.V. for 2 semesters. From 1st September 2019 until 7th Februari 2020 I will be working on the assignment, which will be explained in the next chapters.

Crownstone B.V. is a company that combines indoor localization with automation for home and office spaces. They are the creators of the Crownstones. Crownstones are smart devices that make your home or office smart. A crownstone can function as a switch, a dimmer a power monitor and a standby killer. The devices are connected to a smartphone using Bluetooth Low Energie (BLE). The crownstones use indoor positioning. When connected to an owner's smartphone they can calculate his posistion and execute their functions in a room based on the posistion of the owner's smartphone. The crownstones have machine learning capabilities. They have the ability to learn which type of rooms there are in the huis and where they are.

The assignment that I have been given is to make the crownstone compatible with the arduino environment.

2 Assignment Information

This assignment was given to me by the CEO of Crownstone B.V., Anne van Rossum. The assignment is to make the Crownstone compatible to run Arduino code and to be run by Arduino code. The reason for this assignment is to attract the open-source community. At the moment the Crownstone runs on its own code. The Crownstone has many functions such as dimming lights, turning lights on and off, locating your position in the house. All these functions need to be accessible to the programmer when writing a arduino script.

2.1 Requrement List

This assignment has a list of requirements that need to be met before it can be considered as finished.

- DFU (update over the air) flashing/uploading of code intead of OpenOCD/JTAG
- Should be able to run basic arduino code
- Needs to be compatible with Platformio
- Needs to be compatible with Arduino IDE
- Automatic download of the toolchain
- Crownstone needs to be added to the Arduino IDE board manager

3 Crownstone B.V. Background

Crownstone, founded in 2016, is a company that combines indoor localization with building automation for homes and offices. Crownstone went through the Rockstart accelerator (demo day July 2016). We acquired a large network of business contacts, have a scrutinized business model and it ranks our company as a potentially disruptive and scalable player. Crownstone has support from Almende as research company and from Almende Investments as investor. This guarantees that Crownstone can produce beyond state-of-the-art technology and has enough budget to grow organically. Crownstone's unique selling point is indoor localization. This has not been capitalized upon by any competitor in the home automation market, and only a few in the building automation market.

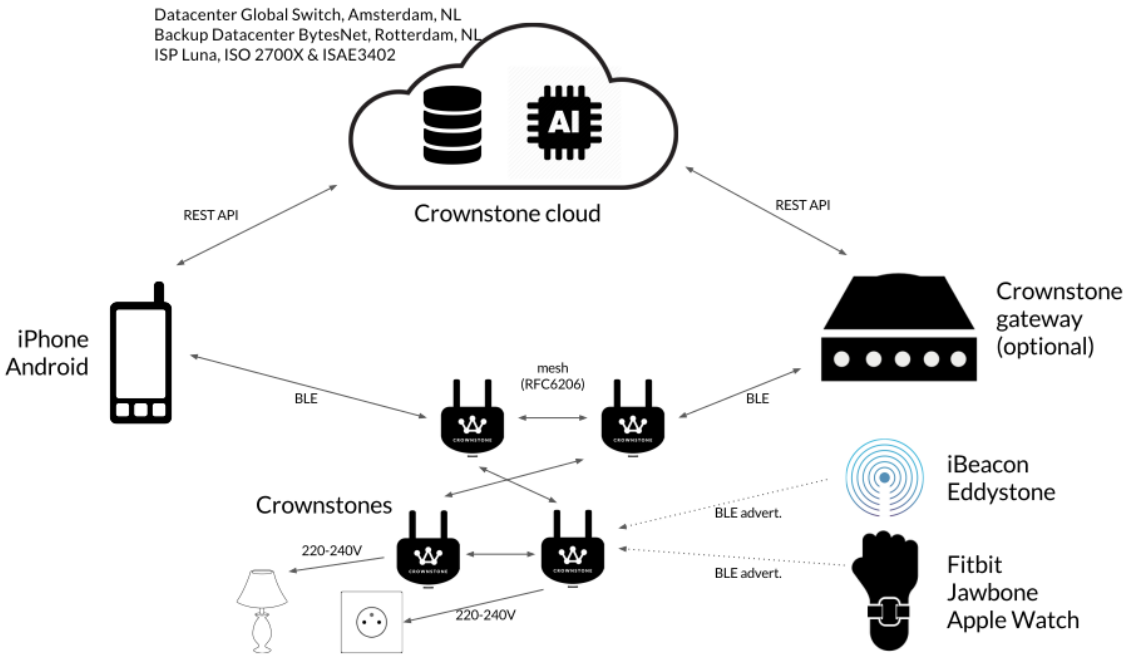


Figure 1: Crownstone Architecture

4 Internship goals

4.1 Activities

Beheren De afgestudeerde is in staat om in een gegeven beroepssituatie het proces van ontwikkeling, ingebruikname en gebruik van ict-systemen beheersbaar te laten verlopen, rekening houdend met de context en relevante stakeholders.

Analyseren De afgestudeerde kan een probleem ontleden door gegevens over bestaande of nieuwe technologieën, gebruikers, processen, producten of informatiestromen te verzamelen, te beschrijven, te verwerken tot bruikbare informatie, daarover een oordeel te vormen en op basis daarvan een oplossingsrichting te selecteren of te formuleren.

Adviseren De afgestudeerde kan een onderbouwd en richtinggevend advies uitbrengen over processen, software en/of nieuwe technologieën en kan dit overtuigend en begrijpelijk presenteren.

Ontwerpen De afgestudeerde kan binnen vooraf gestelde kaders een systeem vormgeven in termen van functionaliteit, interactie, structuur en architectuur.

Realiseren De afgestudeerde kan een ontwerp omzetten in een bruikbare ict-oplossing, die aansluit bij bestaande systemen, door het schrijven, testen, debuggen, optimaliseren en documenteren. Deze ict-oplossing omvat een combinatie van hardware en software, in de zin dat er software wordt geschreven voor hardware die nieuw wordt samengesteld uit bestaande componenten (sensoren, actuatoren, microcontrollers, communicatie-apparatuur enz.), of dat er software wordt geschreven voor een bestaand, special-purpose hardware-systeem.

Behaald op stage:

Beheren

Analyseren Vooronderzoek en literatuur onderzoek

Adviseren Maken van Testrapporten en rapport

Ontwerpen

Realiseren

4.2 Projectgoals:

1. De student kan de scope van de eigen stageopdracht helder formuleren en duidelijk afbakenen.
2. De student kan de eisen en wensen van de klant vertalen naar een mogelijke oplossing.
3. De student kan een stakeholdersanalyse uitvoeren.
4. De student kan alternatieven en bestaande oplossingen beargumenteerd afwegen, rekening houdend met de stakeholders en/of de techniek.
5. De student kan de Software Configuratie Management in kaart brengen en toepassen.
6. De student kan een risicolog opstellen, actief bijhouden en daarop acteren.
7. De student past versiebeheer op documentatie toe.
8. De student kan issue tracking toepassen.
9. De student kan de opdrachtgever op passende wijze adviseren over de resultaten en conclusies van de verrichte analyse.
10. De student kan de systeemarchitectuur in kaart brengen.
11. De student kan bepalen welke ontwerpen relevant zijn voor de eigen opdracht en deze ontwerpen passend opstellen.
12. De student kan de correcte werking van het gerealiseerde prototype helder demonstreren.
13. De student heeft de gerealiseerde oplossing aantoonbaar getest met passende testvormen en dit gedocumenteerd.

Proof during internship

Project goal	Reference page
1	6
2	
3	12
4	
5	
6	10
7	
8	10
9	
10	7
11	
12	
13	30

4.3 Risk Register

Every project comes with its own risk factors. In this table I have written down some risk that may play of might have played a role during the internship.

Table 1: Risk Register Table

Risk Register				
ID	Description	Probability (1 - 5)	Impact (1 - 5)	Risk Score (1 - 25)
1	Calling in sick	5	5	25
2	Missing a deadline	4	3	12
3	Hardware breaks	3	2	6
4	Supervisor is unavailable	3	2	6
5	Laptop breaks	1	5	5
6	Software gets corrupted	3	4	12
7	Documentation files are lost	5	5	25
8	Research takes longer than expected	5	3	15
9	Missing expertise in work area	5	5	25
10	Wrong approach on assignment (loss of time)	4	3	12

4.4 Issue Tracking

Issue tracking is done using the issue system from github. There I write down the issues I encounter during the assignment.

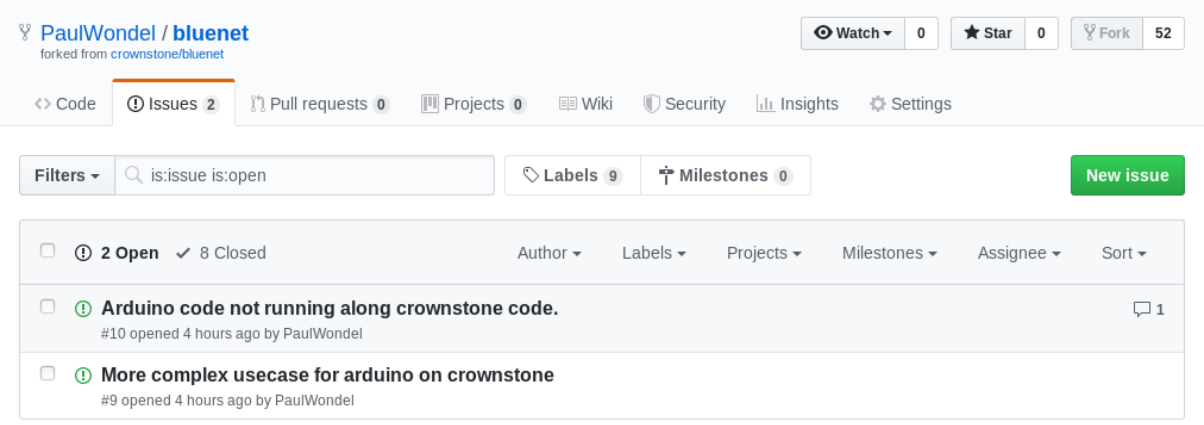


Figure 2: Example: Open issues on Github
(dated: 5 November 2019)

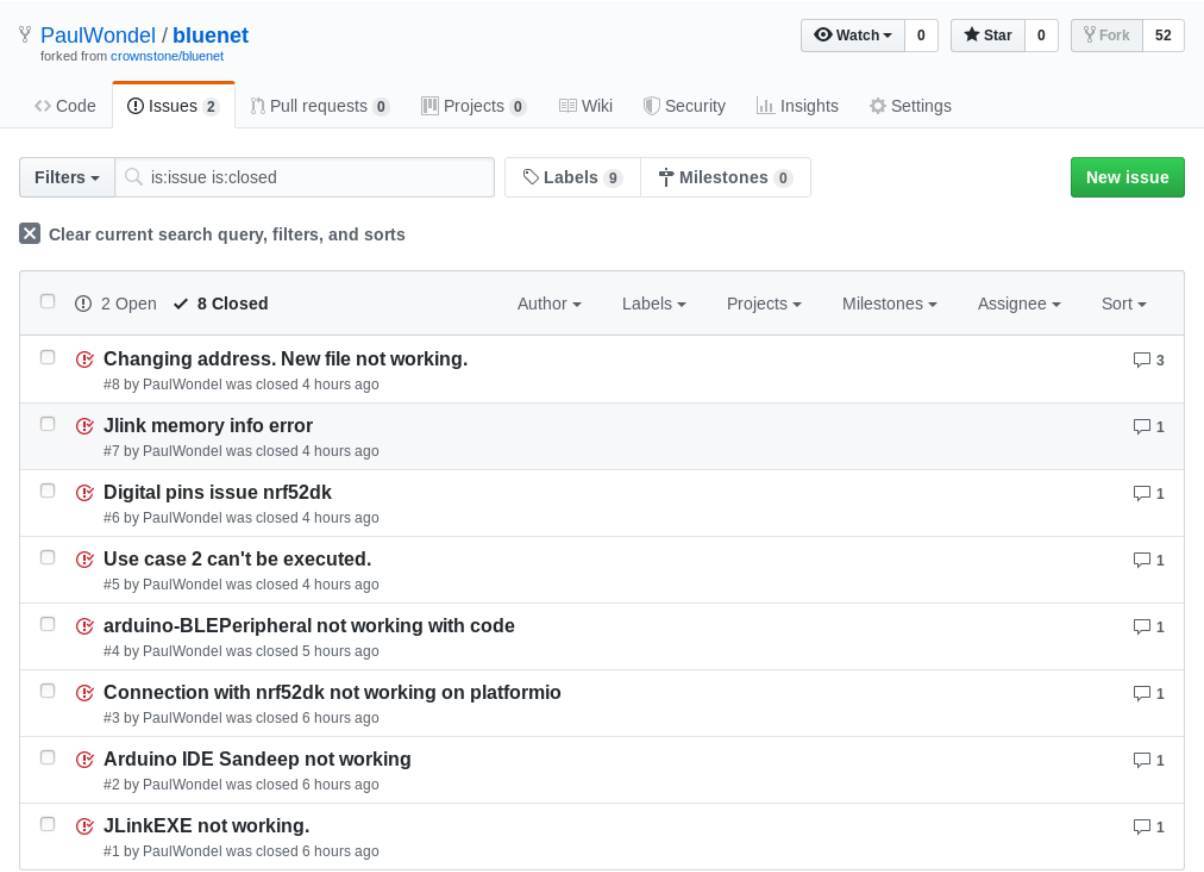


Figure 3: Example: Closed issues on Github
(dated: 5 November 2019)

4.5 Version Control

4.6 Stakeholder Analysis

During this assignment there are stakeholders that have certain interests and influences on the assignment. In the table below the stakeholders are described along with their level of influence and relation.

Table 2: Stakeholder Analysis Table

Stakeholder	Info	Interest	Influence On Assignment	Relative priority
Anne Van Rossum	CEO & Product Owner	Has set the requirements of the end product	High	High
Alex de Mulder	Designer & Software developer	Works on the software which the assignment is depending on	Medium	Low
Bart van Vliet	Software developer	Is the main developer on the firmware for the crownstone	High	Medium
Arend de Jonge	Algorithm Designer & Firmware Developer	Works on firmware, which is a heavy influence on the assignment	Low	Low
Crownstone Customers	-	Is the target group of the assignment	Low	High
Arduino Community	Community that is filled with arduino code developers for various boards	After the assignment the product owner will reach out to arduino developers	Medium	High

5 Research

In this chapter I have collected all the information I have come across during my research of the project. Literature and theoretical cases that have been researched during this internship are also included in this chapter.

5.1 Crownstone Devices

Crownstones are smart devices that can change your home or office into a smart environment. A crownstone can function as a switch, a dimmer a power monitor and a standby killer. The devices are connected to a smartphone using Bluetooth Low Energie (BLE). The crownstones use indoor positioning. When connected to an owner's smartphone they can calculate his posistion and execute their functions in a room based on the posistion of the owner's smartphone. The crownstones have machine learning capabilities. They have the ability to learn which type of rooms there are in the huis and where they are.

5.1.1 Crownstone Plug

The Crownstone plug is a crownstone device that can be easily inserted into an outlet. The plug has the same properties as the in-built crownstone.

5.1.2 In-built Crownstone

The in-built crownstone is a crownstone device tat needs to be installed into the electrical circuit of the house, preferably near the outlets. The in-built has the same properties as the plug crownstone.

5.1.3 Guidestone

This is an iBeacon that is used for the positioning.

5.2 Crownstone Code

De Crownstone code is geschreven door het team van Crownstone B.V. zelf. Crownstone heeft eigen board waarop hij de code draait. De Crownstone draait op een nRF52 Chip van Nordic. De code van de crownstone is open source en heet bluenet. De code voor de crownstone is te vinden op: `https://github.com/crownstone/bluenet`.

5.3 Arduino IDE

Een van de onderdelen van de opdracht is het draaien van arduino code draaien op de crownstone via de arduino. Om borden te kunnen toevoegen aan de arduino IDE moet er een board

5.4 PlatformIO

PlatformIO[2] is a cross platform code builder and library manager for other platforms like Arduino or MBED support. PlatformIO has toolchains, debuggers and frameworks that work on popular plantforms

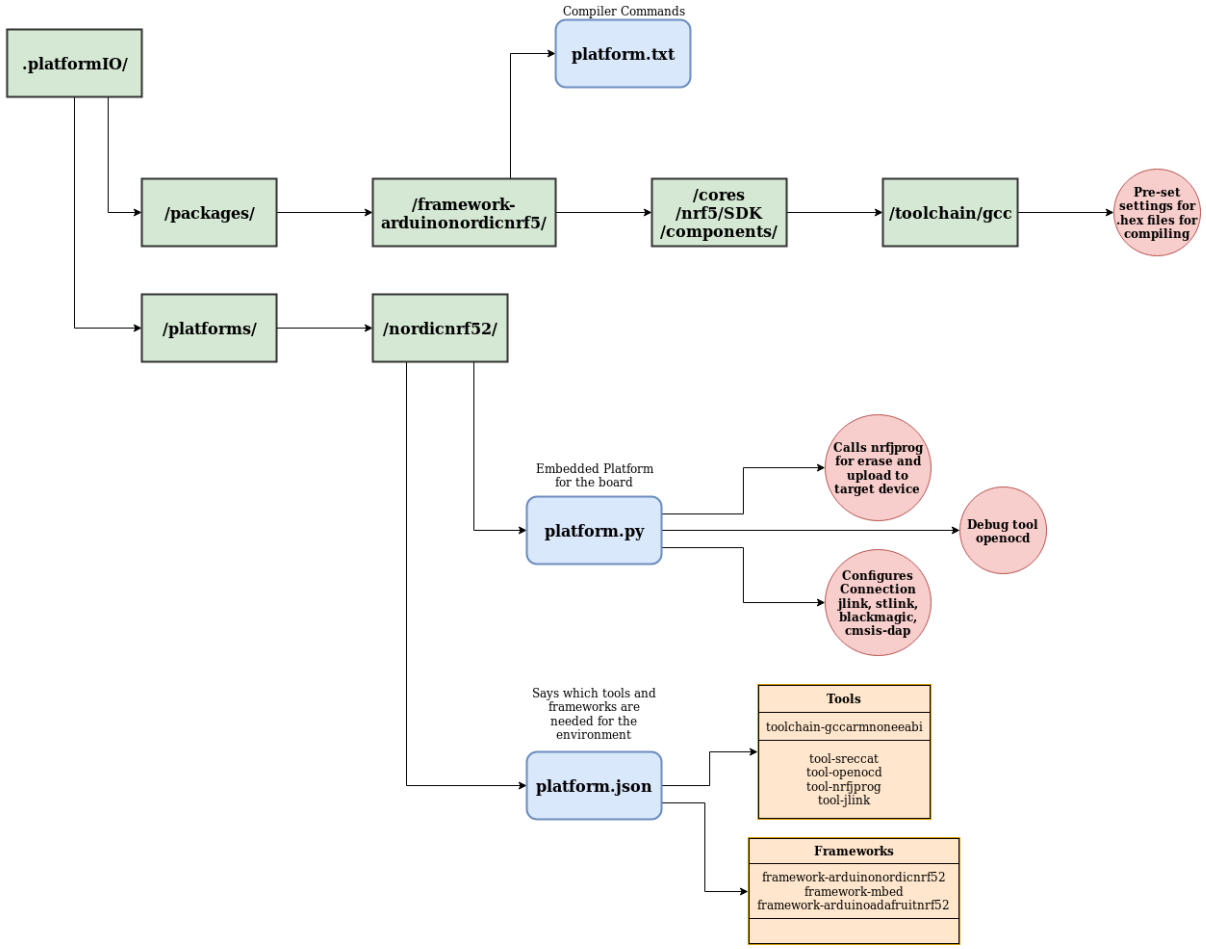


Figure 4: File Structure PlatformIO

like Windows, Mac en linux laten werken. There is support for more than 200 developmnent boards along with 15 development platforms and 10 frameworks. Most of the popular boards are supported. PlatformIO was actually meant to be used through the command line, but with success it is able to be with other IDE's like Eclipse of Visual Studio.

PlatformIO has the options and possibility to add new boards. For a new board the be added on PlatformIO, there needs to be a **JSON structure (.json)** file made for it. PlatformIO published a installation manual for adding new boards.

- Source: PlatformIO documentation

5.4.1 Development platform

Bij platformIO hoort er ook een development platform te zijn dat de toolchains, build scripts en instellingen voor het bord. Een custom Development Platform heeft een paar onderdelen nodig om gebruikt te kunnen werken n.l. packages, een manifest file (platform.json), buildscript (main.py).

- Source: PlatformIO documentation

5.4.2 Advanced Scripting

PlatformIO Build System allows a programmer to extend the build process with their own custom scripts. Main and framework scripts can be found in the /builder/ folder of the platform board folder.

- Source: PlatformIO documentation

5.5 Pinlayout Nordic nRF52-DK

The Pinlayout of the Nordic nRF52-DK does not have a default layout for arduino use. Depending on the framework you pins are assigned variables and names in the header files. To find out how the pins are setup on the Nordic developer board, I had to read the datasheet and introduction guide of the Nordic nRF52-DK.

5.6 Bluetooth Low Energie (BLE)

5.7 DFU

5.8 Tools during the internship

My supervisor has expressed the expectations he has of me. The basic knowledge of certain compiler and debugging tools. He listed a few new terms for me that I should look into.

- Ldd-tool
- nm-tool
- Hexdump -link manual
- objcopy link manual
- objdump
- readelf: Executable and Linkable Format and it defines the structure for binaries, libraries, and core files. -link 1 -ELF link 2
- Binary file -link
- Checksum (md5sum)
- nrfconnect - This is a tool with build gui for developing the nRF52832 developer board
- nrfjprog - This tool is used to flash hexfiles to the nRF52832
- srec_cat -link manual -link examples

5.9 HEX File

A HEX file is program file with binary information commonly used for programming microcontrollers. A compiler converts the program's code into machine code or assembly and outputs it into a HEX file. The HEX file is then read by a programmer to write the machine code into a PROM or is transferred to the target system for loading and execution. A hex file is unreadable for a person. Only machines can read it. To read the data in a hex file, the file needs to be converted into another file type.

- Source: Wikipedia

5.10 .ARM.exidx

LLVM and the ARM ELF .ARM.exidx* section [1, Article on ELF for ARM]

5.11 Random-Access Memory (RAM)

RAM (Random-Access Memory) is a form of computer data storage that is used to store working data and machine code that can be read or changed in any order. And it allows data items to be read or written almost the same amount of time irrespective of the physical location of data inside the memory. Random access memory is volatile, i.e., it requires a steady flow of electricity to maintain its contents, and as soon as the power goes off, whatever data that was in the RAM is lost. It is commonly known as read/write memory, and is an integral part in computers and other devices like printers.

- Source: Wikipedia

5.12 Flash Memory

Flash memory is a derivative of the EEPROM memory. It is designed to make storing large amounts of data in a small space possible, allowing reading and writing in multiple memory locations with the same operation. This type of memory, is based on the use of semiconductors. In addition to being non – volatile and rewritable, it possesses almost all the features of RAM, along with the added advantage that it is non-volatile, meaning, what is stored in this type of flash memory, does not get deleted when you disconnect the device from the PC or the apparatus, unlike RAM. Flash memories are extremely important, especially in today's computer world, owing to its low power consumption, portability and size, as well as safety and efficiency; makes them ideal for supporting data and information created with digital cameras, smartphones, audio devices, among other gadgets. Even, they are quite resistant to any blow or fall, which represents a huge improvement over portable mass storage devices of previous generation.[3]

- Source: Wikipedia

5.13 Linkerscripts

Linkerscripts are are text files with the commands to make object files compiled by a compiler into executable programs. The linker puts the input files into a single output file. The object files have a list of sections. Sections in object files have names and sizes. Most sections also have an associated block of

data, known as the section contents A section may be marked as loadable, which means that the contents should be loaded into memory when the output file is run. Sections with no contents may be allocatable, which means that an area in the memory should be set aside. Nothing in particular should be loaded there. In some cases this memory area should be zeroed out. A section which is neither loadable nor allocatable typically contains some sort of debugging information.

Every loadable or allocatable output section has two addresses. The first is the VMA (virtual memory address) and the second is the LMA (load memory address). VMA is the address the section will have when the output file is run. LMA is the address at which the section will be loaded.

Every object file also has a list of symbols. A symbol may be defined or undefined. Each symbol has a name, and each defined symbol has an address, among other information.

- Source: Linker Manual

5.14 Interrupt Handler

Interrupt handlers are blocks of code with a special condition. They are used for transitioning between protocols. They can be both on hardware level and software level. Interrupt handlers vary based on what triggered the interrupt and the speed at which the interrupt handler completes its task. Zero-to-Main Part 1 Zero-to-Main Part 2

- Source: Wikipedia

6 Phase Progression

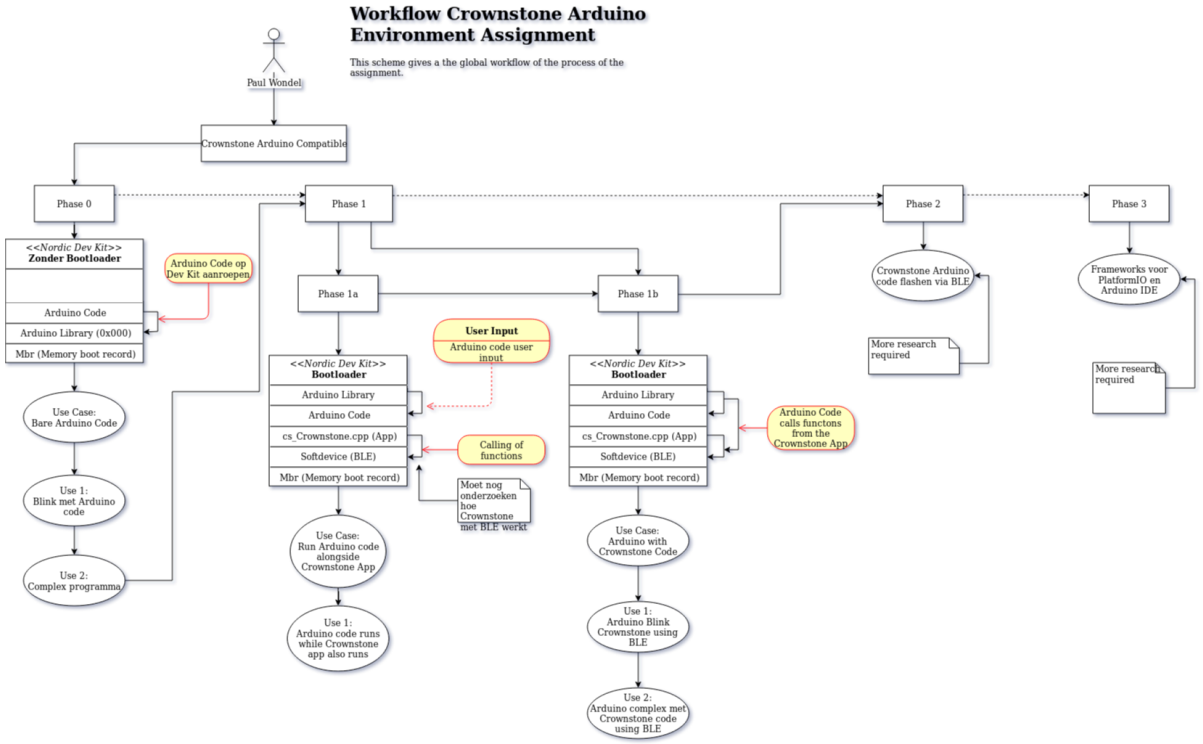


Figure 5: Phase Plan Workflow Scheme

6.1 Phase 0

The goal of phase 0 is to get arduino code to run on the Nordic nRF52-DK Developer kit without a bootloader and without the crownstone app. During the first attempt of connecting the I discovered that the dev kit has no way of flashing code on my windows 10 laptop. Windows 10 does not use the required drivers automatically. If Windows 10 is wished to be used then the drivers for the Nordic nRF52-DK Developer kit have to be installed manually. Therefore I switched from a Windows 10 to Ubuntu 18.4.1.

On the Linux system it is essential to have the right package for the J-Link tool installed. Not all Linux distributions can work with the package. The recommended Linux OS for the package is a Debian based OS such as Ubuntu & Linux Mint. I had Arch Linux installed which had difficulties with the package. It was not the right architecture for the package and the package was not available in the aur repository of Arch Linux. De official website of SEGGER has the right packages available for download. I downloaded the packages from the site and successfully installed the J-Link tool.

PlatformIO has to be installed on the operating system. This is required for programming the Nordic nRF52-DK with arduino code. It can be **PlatformIO Core (CLI)** or **PlatformIO IDE**. During this research I have made use of PlatformIO IDE on **VSCode**. In PlatformIO the right package is needed for the Nordic nRF52-DK board. PlatformIO already has the support for nRF52 Nordic boards. The platform

files still need to be installed which can be done at the "platforms" tab in the PlatformIO IDE. After the required packages are installed, a new project needs to be created with the nRF52 as board and Arduino as framework.

Programming arduino code is the same as programming code for an Arduino Uno. The difference is that the pinlayout is not the same as with Arduino. The pins also have different names and are called upon in a different way.

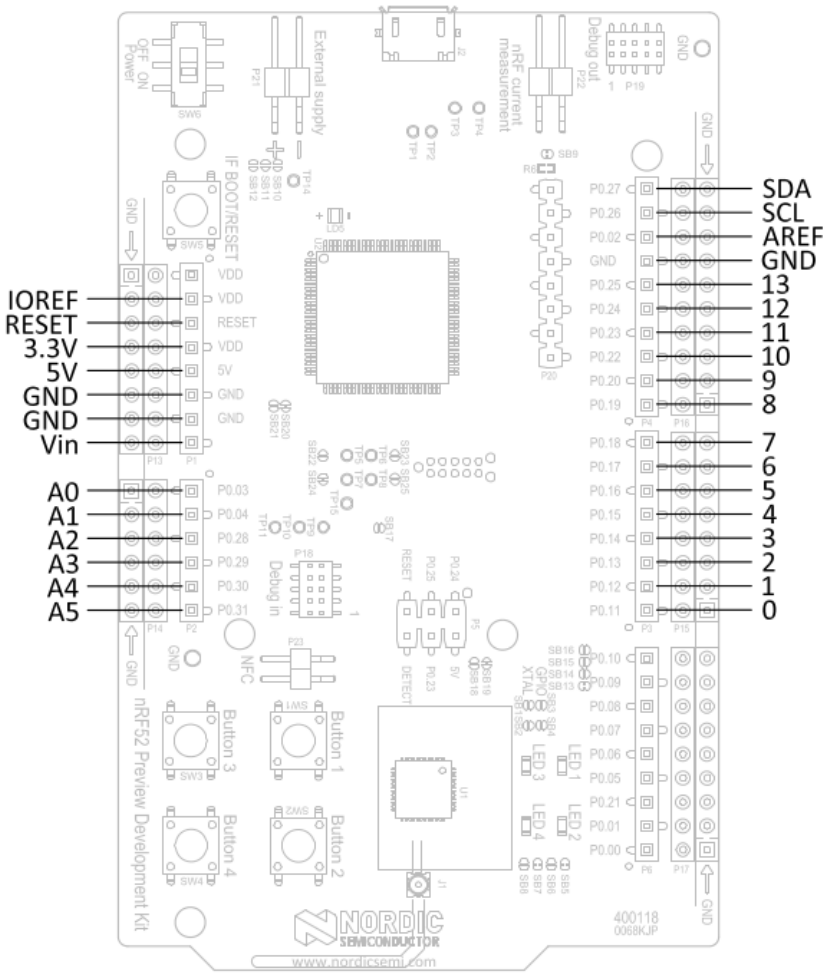


Figure 6: nRF52-DK Pinlayout

Source: nRF52 Preview Development Kit User Guide v1.2, Page 16

The analog pins are called upon by using them as {A0, A1, A2, A3, A4, A5} and the digital pins as {(0),(1),(2),(3),(4),(5),(6),(7),(8),(9),(10),(11),(12),(13)}. Some of the digital pins are already assigned to some buttons and LEDs that are on the developer board.

Here is a example of how the pin definition goes in arduino code:

GPIO	Part	Arduino signal
P0.13	Button 1	2
P0.14	Button 2	3
P0.15	Button 3	4
P0.16	Button 4	5
GPIO	**Part**	**Arduino signal**
P0.17	LED 1	6
P0.18	LED 2	7
P0.19	LED 3	8
P0.20	LED 4	9

Figure 7: Assigned Pins onboard LEDs & Buttons nRF52-DK

Source: RF52 Preview Development Kit User Guide v1.2, Page 17,18

```
#define ledPin (7) //dgital pin 7 on the board (P0.18)
#define irPin PIN_A1 //Analog pin A0 on the board (P0.05)
#define temPin PIN_AREF //AREF pin on the board (P0.02)
```

Use Case: Bare Arduino Code

In this use case the goal is to have bare arduino code run on the Nordic nRF52-DK. This is use case is devided into 2 use cases:

Use Case 1: Arduino Blink A simple arduino script to light up the onboard LEDs by using the onboard buttons.

Use Case 2: Auto Intensity Control of Power LED To test the control of the analog pins as well as the digital pins with pwm using sensors for input data.

The testplan for this phase can be found in the appendix (7).

6.2 Phase 1

In phase 1 we want the arduino code with arduino library to run simultaneously with the crownstone code. To achieve the goal more securely, this phase has been devided into phase 1a and phase 1b. In phase 1a the goal is to run the arduino code along side the crownstone code both independently. The crownstone should be running and should be able to call functions from the softdevice (BLE) regardless of the arduino code. And the arduino code should be able to call funtions of the crownstone.hex to use them in the arduino code.

6.2.1 Phase 1a

6.2.1.1 Installing Crownstone App (bluenet) Installing the crownstone app unto the Nordic nRF52-DK requires a few programs te be pre-installed. These programs are assumed to be installed:

- `git` - sudo apt install git
- `pip3` - sudo apt install python3-pip
- `wget` - sudo apt install wget

When the required programs are installed, the following install instructions can be followed on the repository of crownstone called bluenet: `https://github.com/crownstone/bluenet/blob/master/docs/INSTALL.md`.

During the installation I came across a complication that the tool nrfutil could not be found. The program pip is installing the tool into the python directory. This makes it so that the tool is only usable in a python environement. If this happens, make sure that you install the tool using `sudo pip3 install nrfutil`. If this doesn't help, search on the internet for a way to put the tool in the dir /usr/bin/local so that the operating system can use the `nrfutil`-command outside of the python environement. After successfully installing `nrfutil`, run the `make build_bootloader_settings` command again and follow the rest of the installation instructions.

```
→ default git:(master) make build_bootloader_settings
Scanning dependencies of target build_bootloader_settings
Create bootloader settings
** Firmware version: 3.0.1
** Bootloader version: 2.0.0
** Use files in directory: /home/gmprincep/gitfiles/bluenet/build/default
make[3]: nrfutil: Command not found
CMakeFiles/build_bootloader_settings.dir/build.make:57: recipe for target 'CMakeFiles/build_bootloader_se
ttings' failed
make[3]: *** [CMakeFiles/build_bootloader_settings] Error 127
CMakeFiles/Makefile2:328: recipe for target 'CMakeFiles/build_bootloader_settings.dir/all' failed
make[2]: *** [CMakeFiles/build_bootloader_settings.dir/all] Error 2
CMakeFiles/Makefile2:335: recipe for target 'CMakeFiles/build_bootloader_settings.dir/rule' failed
make[1]: *** [CMakeFiles/build_bootloader_settings.dir/rule] Error 2
Makefile:268: recipe for target 'build_bootloader_settings' failed
make: *** [build_bootloader_settings] Error 2
→ default git:(master)
```

Figure 8: The error that occurred.

The bootloader requires a bootloader settings page that contains information about the current DFU process. It also contains information about the installed application and its version. The bootloader verifies if the application is correct by checking it against the bootloader settings, so each time the application changes, you the bootloader settings also change.

To build and write the bootloader settings:

```
cd build/default
make build_bootloader_settings
make write_bootloader_settings
```

Figure 9: The step of the installation where the error occured.

6.2.1.2 Flashing The crownstone code is put into a hexfile which is that flashed to the nRF52 using the textttnrfjprog tool. In the bluenet repository there is a build directory in which you will find a file called CMakeList.txt. In this list you will find the commands that are used to flash the bootloader settings, the crownstone code itself and how to reset the nRF52. These are the commands:

```
$ nrfjprog -f nrf52 --eraseall
$ nrfjprog -f nrf52 --program softdevice_mainpart.hex --sectorerase
$ nrfjprog -f nrf52 --program crownstone.hex --sectorerase
$ nrfjprog -f nrf52 --program bootloader_settings.hex --sectorerase
$ nrfjprog --reset
```

6.2.1.3 Memory Map

Nordic Semiconductor (The company that makes the Nordic nRF52-DK) has tools available for different uses. In this phase I wanted to know the current used memory registers. To get a visual of the realtime used memory registers I used the tool: `NRFCONNECT_PROGRAMMER`. I have installed the `nrfconnect_core` to get the tools. Within the menu of this tool I chose the programmer tool to be installed. I could not install the programmer seperately from the `nrfconnect_core` due to a npm package installation error.

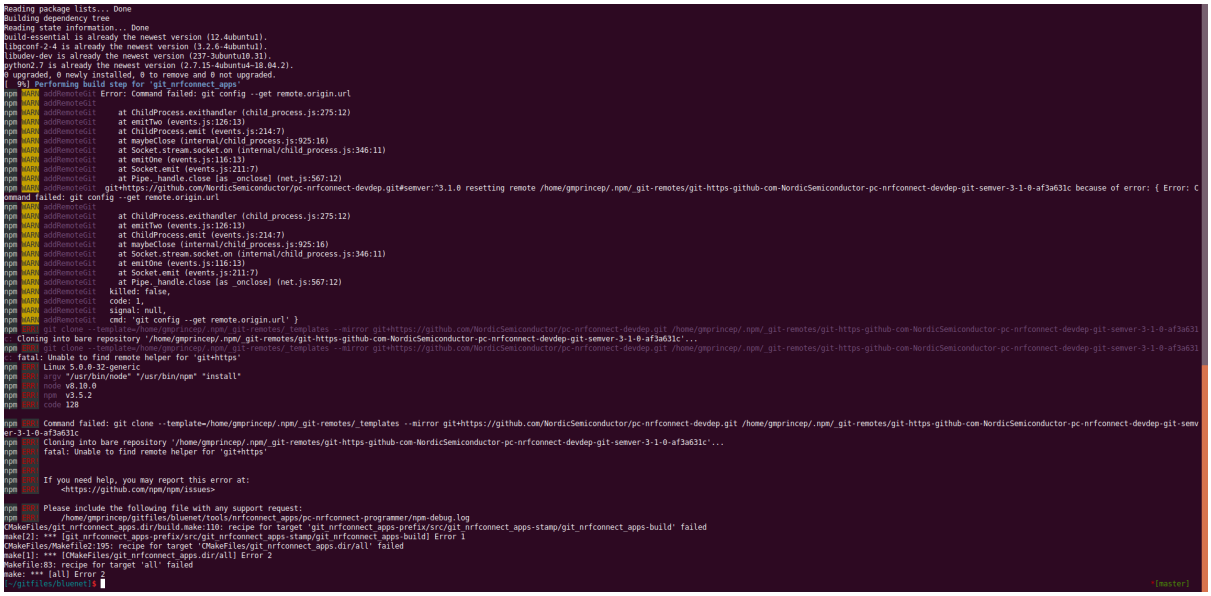


Figure 10: NRFCONNECT_PROGRAMMER Download Error

The installation guide can be found on the git repository of Crownstone: git@github.com:crownstone/bluenet. Make sure that npm installed before following the instructions.

You can enable the download of `nrfconnect` by:

```
cmake .. -DDOWNLOAD_NRFCONNECT=ON
make
```

This particular tool requires `npm`. Install it through something like `sudo apt install npm`. Subsequently, it downloads a lot of stuff, amongst which also `nrfjprog` it it cannot find it. Make sure it does not lead to version conflicts. You can run these by:

```
make nrfconnect_core_setup
make nrfconnect_core
```

These run in separate shells. The `_setup` you at least have to run once. After that it can do continuous rebuilds. You can select the tool to use from the list of apps. By default there are now quite a few apps there. The programmer can also be downloaded separately by setting the `-DDOWNLOAD_NRFCONNECT_PROGRAMMER` flag at `CMake`.

Figure 11: NRFCONNECT Install Guide

Source: https://github.com/crownstone/bluenet/blob/master/docs/BUILD_SYSTEM.md

After installing the nrfconnect_core you will need to run the `nrfconnect_programmer`. To start the tool, run the command `make nrfconnect_core` in the bluenet directory. The `nrfconnect_core` gui tool looks like this.

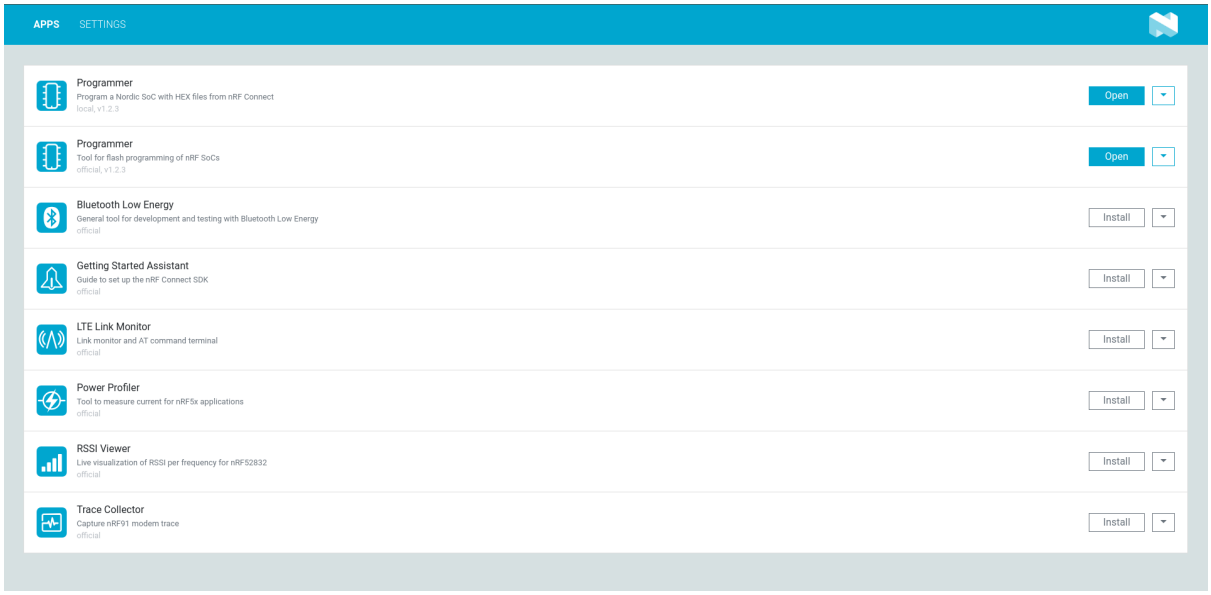


Figure 12: NRFCONNECT_CORE GUI

With the `nrfconnect_programmer` tool the memory map of the current device can be displayed in realtime. It also displays the addresses the files are loaded unto. The programmer tool gui looks like this.

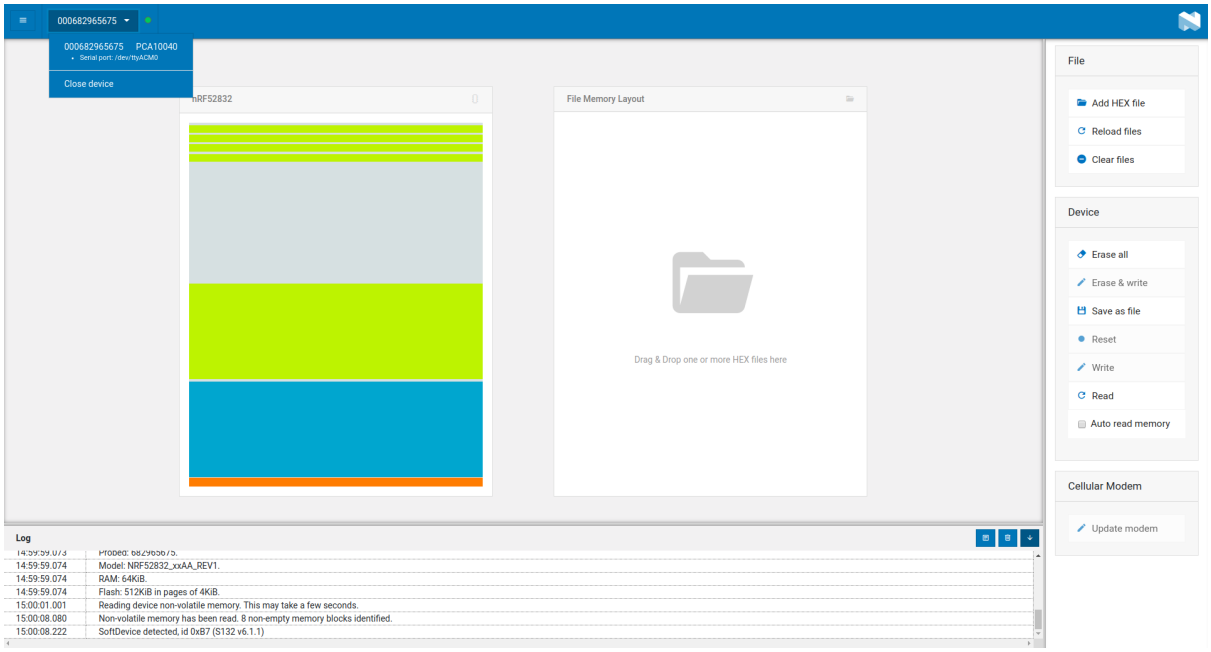


Figure 13: NRFCONNECT_PROGRAMMER GUI

After displaying the memory map the addresses in the memory had to be found. The programmer tool has the option to show the start address of the hex files. I placed the hex files of the crownstone firmware along with the bootloader settings in the upload section of the tool to display the sizes of the files with their start addresses. This way I could see their addresses in the memory. These files are available in a different git repository that can be downloaded from: git@github.com:crownstone/bluenet-release. The files that are revelevant here are the: crownstone.hex, bootloadersettings.hex, softdevice_mainpart.hex. For the certainty of having the right start addresses of the files, I inspected the files seperately in commandline. In the .elf files the start address of the program is located. To read a .elf file, you use the command `readelf [option] [inputfile.elf]`. To get the addresses I used the command `readelf -Sh crownstone.elf`. I have applied this to each of the files and came up with an estimation of their sizes.

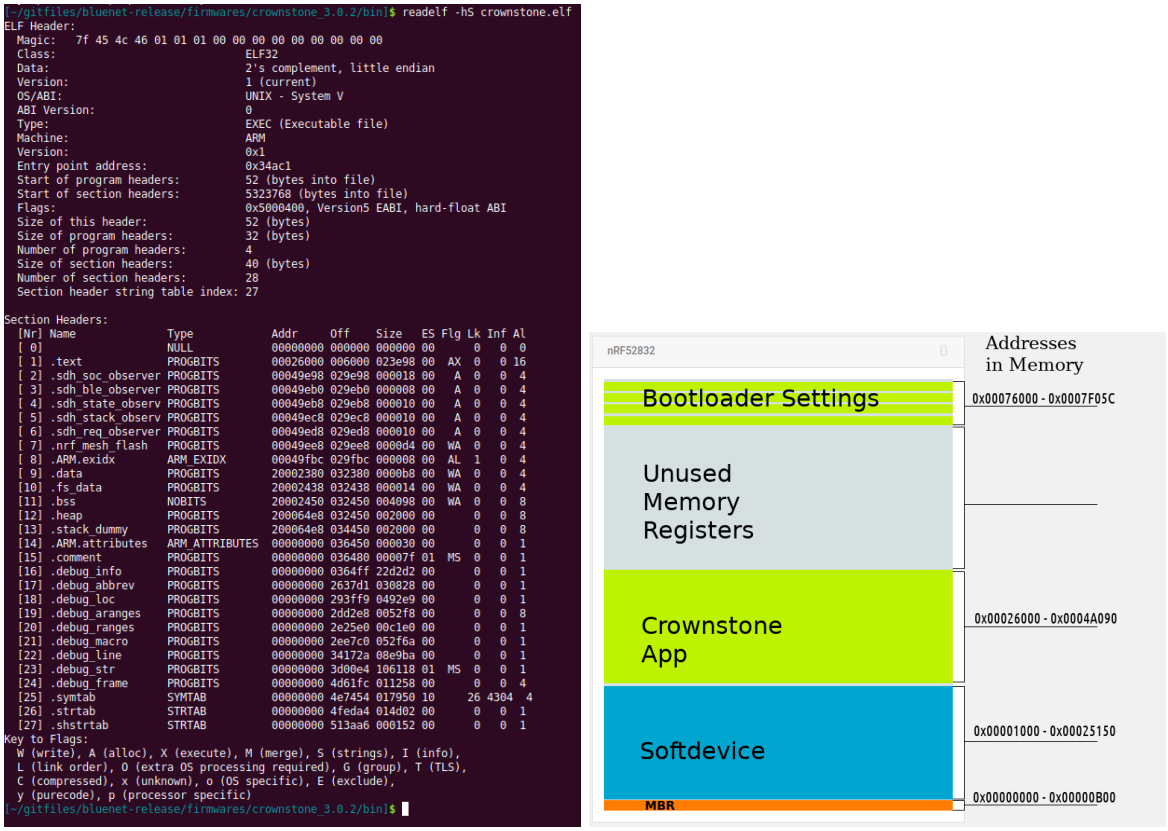


Figure 14: Addresses of the .hex files in the memory registers on the nRF52

6.2.1.4 Arduino HEX-file

When compiling arduino code in platformio, the .hex files are stored in a directory. This directory is hidden and only used for the .hex file to be stored in and to be flashed from. Because I needed the .hex and .elf files of the arduino code I had to get them from /home/$USER/Documents/PlatformIO/Projects/PROJECT_NAME/.pio/build/nrf52dk/. In there the files firmware.hex and firmware.elf are located, which contains the arduino.cpp converted into assembly code for the chip to be read.

6.2.1.5 Flashing Arduino HEX-file

PlatformIO uses the same tools that bluenet from crownstone uses to compile, build & flash to the nRF52-DK. The tools are:

- `nrfjprog` - For writing files to nrf52
- `jlink` - For connecting to nrf52
- `nrfutil` - A Python package that includes the nrfutil utility and the nordicsemi library
- `gcc-arm-none-eabi` - For compiling and building the .elf and .hex files

The start address of the firmware.hex is 0x00000000.

When trying to flash the arduino file, I need to know if the file will any complications with the addresses. By running the command `readelf -e firmware.elf` I could read the start address of the

program which indeed happened to be 0x00000000. This comes in conflict with the softdevice of the crownstone that already has that address reserved for itself.

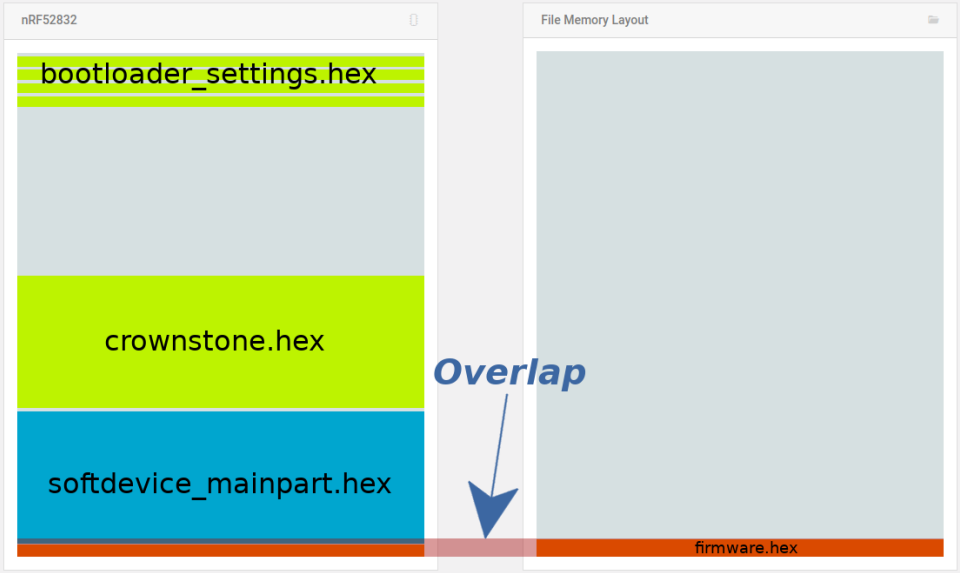


Figure 15: Crownstone Code VS Arduino Code

The solution is to change the address of the `firmware.hex` to a different address. To edit a hex file its .elf file needs to be edited and then converted to .hex. To write to an .elf file I used the arm-none-eabi-objcopy tool. By running the command:

```
arm-none-eabi-objcopy --change-addresses 0x00026000 -I elf32-little firmware.elf -O
elf32-little test.elf
```

I was able to change all the addresses of the sections within the firmware.elf and called the output file test.elf. After changing the address in test.elf I had to convert the file into a .hex file. To do that I ran the command: `arm-none-eabi-objcopy -O ihex test.elf test.hex` and this created the output file test.hex. I flashed the test.hex along with the softdevice_mainpart.hex and bootloader_settings.hex using the following commands:

```
$ nrfjprog -f nrf52 --eraseall
$ nrfjprog -f nrf52 --program softdevice_mainpart.hex --sectorerase
$ nrfjprog -f nrf52 --program test.hex --sectorerase
$ nrfjprog -f nrf52 --program bootloader-settings.hex --sectorerase
$ nrfjprog --reset
```

After the successfull flashing of files, the code did not execute.

6.2.1.6 Re-compiling the Arduino Hex-file in PlatformIO

After discussing with my supervisor, I realized that I had taken the wrong approach to changing the start address of the `firmware.hex`. Instead of changing the address of the already compiled hexile, I need to delve into the compiling process

of platformio where it explicitly uses the commando or script and change the start address there. To find out where in the process PlatformIO does the compiling I had to refer to the documentation of PlatformIO. The command for compiling code in platformio is `platformio run`. I needed to know what platformio does during this command, so running the command with verbose would show me at least some of the processes that are run in the background.

Figure 16: Verbose output platformio run command

In the output that `platformio run` gave, the command `arm-none-eabi-gc++` is shown to be called upon. In the command .platformio/packages/framework-arduinonordicnrf5/cores/nRF5/SDK/components/toolchain is set as input in this command for compiling. When I went into the folder I found the linkerscripts that are used to build the firmware.hex file. The linkerscripts are located in the gcc folder and the script with the start address is named nrf52_xxaa.ld. The contents of the file should display as:

```
/* Linker script to configure memory regions. */

SEARCH_DIR(.)
GROUP(-lgcc -lc -lnosys)

MEMORY
{
  FLASH (rx) : ORIGIN = 0x00000000, LENGTH = 0x80000
  RAM (rwx) :  ORIGIN = 0x20000000, LENGTH = 0x10000
}


INCLUDE "nrf52_common.ld"
```

The FLASH (rx) : ORIGIN = 0x00000000, LENGTH = 0x80000 is the start address for where the firmware.hex is going to be placed in the memory of the nRF52-DK. The reason why platformio puts the address at 0x000000000 is because that is where the CPU starts. I changed the address to 0x00026000 and then uploaded the hex file along with the softdevice_mainpart.hex and bootloader_settings.hex. After uploading the files and resetting the device, the arduino code was being executed by the cpu. The arduino code is the same code that is made in 6.1.

6.2.1.7 Running Arduino code alongside Crownstone (both independently)

The crownstone application is the only one that is being executed by the cpu. There is no function in the crownstone

to refer cpu to the arduino code to execute it. There needs to be a reference in the crownstone code to the address of the arduino code.

The arduino code and crowstone code function as two different applications. There is no way for one application to just call a function or an object in another application.

6.2.2 Phase 1b

Use Case: Arduino code with Crownstone code

In this use case the goal is run arduino code on a nrf52-dk that also runs crownstone code. The crownstone code is to be used in arduino code for certain features.

Use Case 1: Arduino Blink A simple arduino script to light up LEDs by using the onboard buttons. The arduino script is only to be executed. It has no function calling yet from the crownstone.

Use Case 2: Arduino Bluetooth Sensor The arduino code is going to read values from the sensor using bluetooth connected sensor. The arduino must not directly connect the radio of the nrf52-dk board but must call upon the bluetooth functions of the crownstone code.

References

[1] R. Earnshaw. *ELF for the ARM Architecture.* ARM: Development systems Division - Compiler Tools Group, v0.3 edition, December 2003. document nr: GENC-003538.

[2] PlatformIO-Revision. What is platformio? `https://docs.platformio.org/en/latest/what-is-platformio.html`.

[3] K. Reddy. What is flash memory? `https://www.quora.com/What-is-flash-memory`, October 2016.

7 Appendix

List of Figures

List of Tables

Testplan
Fase 0

Paul Wondel
0947421
Technische Informatica

HRo - Crownstone B.V. — October 3, 2019

Project Information

In phase 0 the goal is to run raw arduino code on the nordic develepor kit (Nordic nRF52-DK). Phase 0 has been broken down to 2 use cases.

- Use case 1: Run Arduino Blink
- Use case 2: Auto Intensity Control of Power LED

This test is for use case 1. The goal is to test the basic functions of the arduino library on the Nordic nRF52 DK. The functions digitalWrite(), digitalRead(), analogRead() & analogWrite() are subjected to the test in this plan. Also the posibility to use the analog and digital pins on the Nordic nRF52-DK is to be tested. In order to test this there are a few requirements that need to be met.

1 Use Case 1

1.1 Requirements

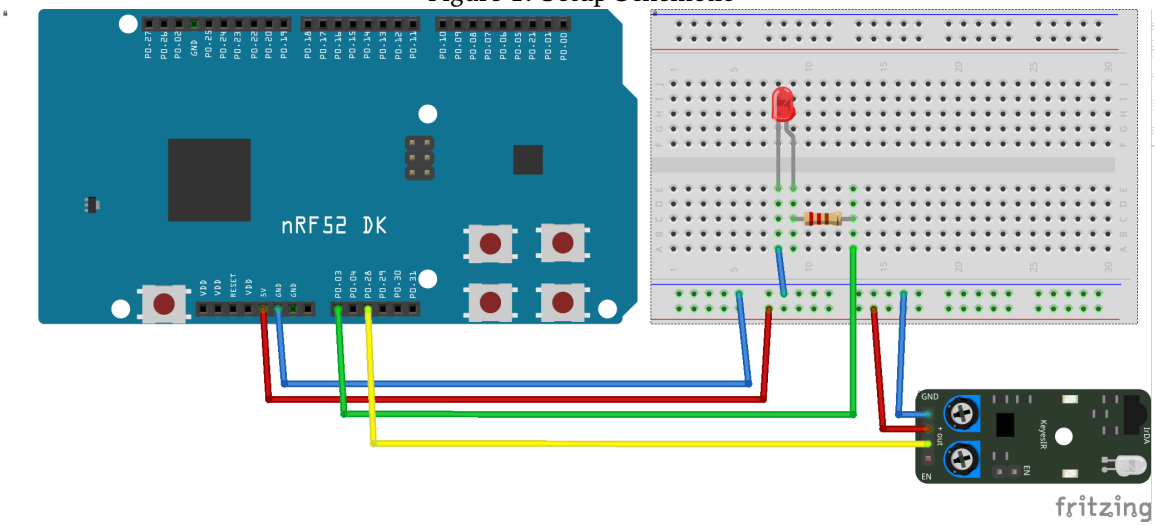
In preparations of this test these are the requirements that should be met:

- A Editor (Visual Code, Atom, eclipse, etc)
- Platformio IDE installed.
- Segger J-Link & tool-jlink (in Platformio) installed
- 1x Nordic nRF52 Developor Kit
- 1x Small LED
- 2x Circuit building wires
- 1x 220K Resistor
- Connecting Wires
- 1x Breadboard
- 1x Infrared Sensor
- Arduino Script: *'Blink'*

1.2 Constructing

These are the schematics and code for the test setup. Setup the parts in the same positions as in the schematics. Then upload the code to the board.

Figure 1: Setup Schemetic



Listing 1: Test Code

```
#include <Arduino.h>

#define ledPin PIN_A0
#define irPin PIN_A1

void setup() {
  Serial.begin(9600);
  pinMode(ledPin,OUTPUT);
  pinMode(irPin,INPUT);
}

void loop() {
  Serial.println(analogRead(irPin)); // Shows the values from the sensor
  if(analogRead(irPin) < 600){
    digitalWrite(ledPin, HIGH); // set the LED on
    Serial.println("Led On");
  }
  else {
    digitalWrite(ledPin, LOW); // set the LED off
    Serial.println("Led Off");
  }
}
```

1.3 Testing

Instructions: After constructing the circuit and uploading the script, the Infrared Sensor has to be configured. The Infrared Sensor has it's own resistor on the circuitboard which can be adjusted with a screwdriver. When the sensor has been configured, test the object detection. The values should appear in the serial monitor. The low values (when there is no object detected) should be between 0-200 and the high

values should be higher than 600. To test the setup place an object in front of the sensor. The led should light up when the object is detected by the sensor.

Questions during the performance For performing the test itself there are a few questions that need to be answered during the test.

Question 1

Does the LED turn on when an object is placed in front the sensor?

a) Yes b) No

Question 2

Does the Infrared Sensor detect the object when placed in front of the sensor? (The light on the sensor itself wil light up when detecting an object)

a) Yes b) No

Question 3

Do the printed lines 'Led on' or 'Led off' appear in the serial monitor?

a) Yes b) No

2 Use Case 2

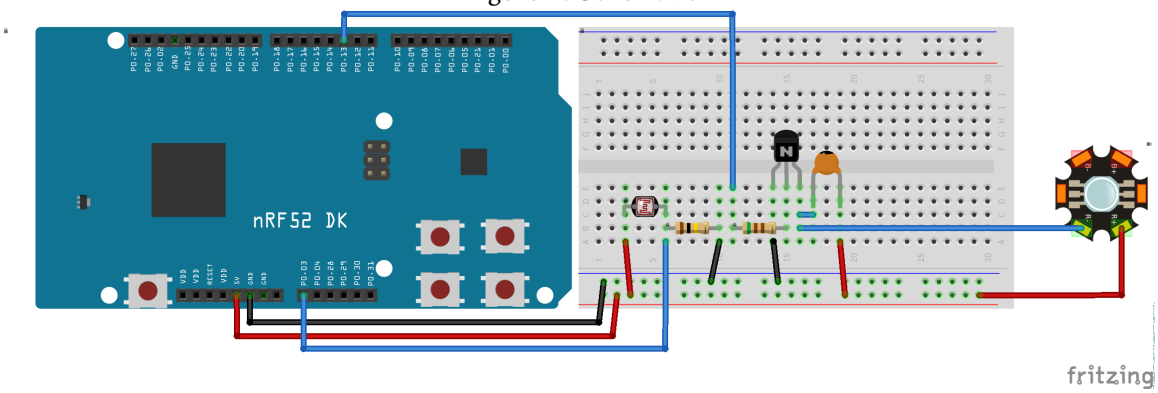
2.1 Requirements

- A Editor (Visual Code, Atom, eclipse, etc)
- Platformio IDE installed.
- Segger J-Link & tool-jlink (in Platformio) installed
- 1x Nordic nRF52 Developor Kit
- 1x LDR (Light Density Resistor) Sensor
- 1x Resistor (510, 100k ohm)
- 1x Capacitor (0.1uF)
- 1x Transistor 2N2222
- 1x 1 watt Power LED
- Connecting Wires
- 1x Breadboard
- Flashlight or mobile light source

2.2 Constructing

These are the schematics and code for the test setup. Setup the parts in the same positions as in the schematics. Then upload the code to the board.

Figure 2: Schematic



Listing 2: Test Code

```
#include <Arduino.h>

int pwmPin = (2); // assigns pin 12 to variable pwm
int pot = A0; // assigns analog input A0 to variable pot
int c1 = 0;   // declares variable c1
int c2 = 0;   // declares variable c2

void setup()  // setup loop
{
  pinMode(pwmPin, OUTPUT);
  pinMode(pot, INPUT);
  Serial.begin(9600);
```

```
}

void loop()
{
  int value = analogRead(pot);
  Serial.println(value);
  c1= value;
  c2= 500-c1;          // subtracts c2 from 1000 ans saves the result in c1

  if (value < 500)
  {
  digitalWrite(pwmPin, HIGH);
  delayMicroseconds(c2);
  digitalWrite(pwmPin, LOW);
  delayMicroseconds(c1);
  }
  if (value > 500)
  {
    digitalWrite(pwmPin,LOW);
  }
}
```

2.3 Testing

Instructions: After constructing the circuit and uploading the script, the setup is ready to be tested. To test the setup you must aim the light source in straight unto the LDR Sensor. This should cause the LED (that is turned on by default) to turn off.

Questions during the performance For performing the test itself there are a few questions that need to be answered during the test.

Question 4

Does the LED turn off when you shine light unto the LDR?

a) Yes b) No

Question 5

Does the serial monitor show the values given by the LDR?

a) Yes b) No

Daily Log

```
02-9-2019
  - Eerste meeting en kennismaking met het bedrijf en collegas
  - Personeels document ingevuld
  - Begonnen met doornemen van bluenet Documentatie
  - Logboek opgesteld
  - opgezocht hoe arduino libraries worden gemaakt

03-9-2019
  - Onderzoeken in Sandeep's projecten hoe zij de nRF52 board hebben laten samenwerken
    met een arduino API
  - test Arduino library aan het schrijven (succes)

04-9-2019
  - C++ tutorial
  - Todo list en planning voor opdracht
  - platformio IDE bekijken
    - platformio is een betere keuze om de crownstone compatible mee te maken
      http://blog.marxy.org/2016/03/platformio-arduino-ide-for-programmers.html

05-9-2019
  - Platformio onderzocht voor nieuwe embedded boards
  - onderzocht wat de crownstone nodig heeft voor toepassing bij platformio

06-9-2019
  - Stageplek verkennen

09-9-2019
  - Stage Poster gemaakt
  - Begin verslag gemaakt
  - Met stagebegeleider gezeten en gesproken over de opdracht. (bekijk vragen PVA.docx)
    (Feedback)
  - begin van lijst gemaakt met arduino functies voor de crownstone

10-9-2019
  - Verder aan lijst met arduino functies gewerkt
  - Virtualbox ubuntu opgesteld voor platformio
  - platformio installeren (begin)

11-9-2019
  - Wat is Meshing opgezocht
  - Lijst met functies van de crownstone gemaakt (niet af)

12-9-2019
  - Gewerkt aan verslag

13-9-2019
  - Opdracht verdeeld in kleinere stukken met stagebegeleider (Feedback)
  - Planning gemaakt voor Stage periode
  - Gewerkt aan verslag
  - SLC Opdracht gedaan
```

```
16-9-2019
  - Gewerkt aan verslag
  - Feedback van stagebegeleider gehad over planning
  - Planning aangepast
  - Onderzoeksverslag begonnen te schrijven
  - Begonnen aan fase 0

17-9-2019
  - Doorwerken aan fase 0, Arduino proberen te draaien op nordic dev Kit
  - Arduino IDE met Nordic Dev Kit van sandeep niet gewerkt op Windows 10, er is een
    probleem met de jlink drivers die niet geaccepteerd worden
  - Arduino IDE met nordic dev kit via platformio werkt niet op windows 10, jlink.exe
    can niet verbinden met de dev kit vanwege de drivers.

18-9-2019
  - J-link tool installeren op linux, duurt veels te lang (programma blijft compilen,
    maar ik weet niet wat)
  - J-link tool succes op linux.
  - Tijdens installatie van J-Link tool is mijn cpp lib op archlinux in de war geraakt
  - Nordic met succes op linux, uploaden en flashen lukt. (Via platformio)
  - Verder aan verslag gewerkt (thuis)

19-9-2019
  - Arduino Blink werkend gekregen op Nordic Dev Kit nRF52832
  - Arduino Button met LED werkt op nordic dev kit nRF52832
  - Moet een complexer programma in arduino code laten draaien op de dev kit (moet nog
    bedenken)
  - Framework arduino bekijken van platformio voor nRF52832 (Nog niet af)
  - Probeer BLE te gebruiken met arduino code

20-9-2019
  - BLE lib niet te gebruiken op arch linux. moet vcpkg installeren.
  - ARduino IDE kan niet flashen naar nRF52, libraries zijn niet beschikbaar voor
    archlinux (moet ubuntu/macOS gebruiken)
  - vcpkg geinstalleerd, nog niet te gebruiken

23-9-2019
  - Ubuntu installeren op Lenovo laptop

24-9-2019
  - Tools for Nordic, vscode, latex installeren
  - Blink weer succes op nodric board
  - Ik probeer nu pinnen aan te sturen met arduino code op nordic
  - LED en Infrarood aansturen met nordic pins, success

25-9-2019
  - Docent bezoek voorbereiden (presentatie maken etc)
  - Tips vanuit Anne en Gerard na de presentatie:
    - Testplan maken voor opdrachten
    - Activiteiten verwerken in mijn planning voor de opdracht
    - architectuur plaatje (waarin mijn project te maken heeft) maken voor mijn (Model
    van de werkelijkheid van crownstone)
    - voordat je gaat programmeren knelpunten identificeren (moeilijkste dingen eerst)
```

```
    - Plan van aanpak weer oppakken en blijven mee werkend
    - Requirement list opstellen voor de opdracht
    - Heldere afspraken met stagebegeleider
    - Als je van de planning afwijkt, ga professioneel er mee om, meldt optijd en
    bespreek met stage begeleider
Todo:
  - Stappenplan maken
  - Plan van aanpak maken
  - Activiteiten verwerken in planning
  - Architectuur schets maken van crownstone
  - Requirementlist voor de opdracht
  - Risico Analyse maken

26-9-2019
  - Complexe ardiuno code test opstellen.
  - Testplan opstellen voor fase 0

30-9-2019
  - Finishing testplan voor fase 0
  - Complex programma gevonden:
    https://circuitdigest.com/microcontroller-projects/auto-intensity-control-of-power-
    led-using-arduino/
  - Testplan use case 1 af.
  - test use case 2 niet mogelijk, ik mis componenten
1-10-2019
  - Test plan use case 2 aanmaken
  - Verslag bijwerken

2-10-2019
  - Digitale pins aansturing werkt niet, oorzaak zoeken

3-10-2019
  - Oplossing zoeken aansturing digitale pins
  - digitale Pinnen ipv D0 - D13, (0)-(13).

7-10-2019 --> 11-10-2019
  - Ziek, afwezig op kantoor

14-10-2019
  - Begonnen aan Fase 2
  - J-Link connectie voor memory inzage (niet gelukt)

15-10-2019
  -   sudo JLinkExe -device nRF52832_xxAA -if swd -speed 4000
  -   J-Link Connectie gelukt voor memory inzage na invoer van commando hierboven
  -   Gesprek met Anne:
    - Onderzoeken wat platformio op de achtergrond doet
    - Crownstone app eerst plaatsten op de nordic dev kit
    - via crownstone bootloader kijken naar memory registers
    - use case complexer maken van connectie en niet zeer op het programma/script zelf
  - Installing crownstone app op nordic dev kit

16-10-2019
```

```
  - Installing crownstone app nordic device
  - Complicaties met commando 'make build_bootloader_settings' "nrfutil: Command not
    found" (hulp gevraagd bij Bart en Alex)
  - nrfutil niet geinstalleerd door standaard crownstone installer (anne op de hoogte
    gesteld, gefixed in commit)
  - CMakeLists.txt voor commando's te gebruiken
    - nrfjprog -f nrf52 --program crownstone.hex --sectorerase
    - nrfjprog --reset

17-10-2019
  - onderzocht: wat is hexfile
  - onderzocht: upload hexfile naar crownstone
  - onderzocht: hoe maakt platformio een hex file
  - onderzocht: welke tools gebruikt platformio (srec_cat & Mergehex)
  - onderzocht: wat is checksum

21-10-2019
  - NRF_CONNECT: realtime memory map programma
  - NRF_CONNECT_CORE: installatie gelukt
  - NRF_CONNECT_PROGRAMMER: installatie niet gelukt
  - research verslag uitgebreid

22-10-2019
  - Mermory map bekeken met NRF_CONNECT_CORE,
  - firmware.hex(arduino code via platformio) overlapt met softdevice_mainpart.hex, moet
     adres van firmware.hex veranderen
  - change intel hex file address: nios2-elf-objcopy --change-addresses <mem address ex
    .: 0x300000> <inputfile>.hex <outputfile>.hex
  - verslag gewerkt

23-10-2019
  - readelf tool - onderzocht
  - srec_cat - onderzocht
  - geprobeert start address te veranderen van firmware.hex (niet gelukt)
  - Verslag verwerkt

24-10-2019
  - werken aan verslag
  - adress hexfile proberen te veranderen met
  - hexdump tool - onderzocht
  - objdump tool - onderzocht
  - objcopy tool - onderzocht
  - van elf file naar hex file converten (onderzocht en uitgevoerd: gelukt met objcopy)
  - checksum uitgevoerd op test.hex en vergeleken met firmware.hex

25-10-2019 (Inhaal dag)
  - gcc-arm-none-eabi tool (onderzocht)
    - arm-none-eabi-objcopy (onderzocht)
  - start address veranderen in elf file (gelukt met arm-none-eabi-objcopy)
  - elf file succesvol converted naar hex file
  - test.hex wordt niet uitgevoerd (firmware.hex is voor ARM gespecificeerd, test.hex
    niet)
  - zoek manier om hex te specificeren voor ARM
```

```
  - .ARM.exidx section schuift niet mee,

28-10-2019
  - Sections verschuiven werkt niet zomaar. Een deel van de file gaat naar de flash, het
    ander deel in de ram (Twee weken tijd verloren)
  - Moet Anne meer wijzen waar ik mee bezig bedenken
  - opzoek naar het commando waar platformio die elf file maakt en de addressen aangeeft
    .

29-10-2019
  - opzoek naar het commando waar platformio die elf file maakt en de addressen aangeeft
    .
  - flowchart gemaakt van de platformio files
  - linker files gevonden waar de addressen worden bepaald voor the hexfile die
    gecompiled wordt
  - TODO: extended build script schrijven voor eigen arduino compiling.

30-10-2019
  - File .platformio/packages/framework-arduinonordicnrf5/cores/nRF5/SDK/components/
    toolchain/gcc/nrf52_xxaa.led heeft het start address
  - address succesvol veranderd van 0x00000000 naar 0x00026000
  - firmware.hex gecompileerd in platformio
  - firmware.hex samen met softdevice_mainpart.hex en bootloader_settings.hex geupload
    op nrf52-dk
  - moet crownstone samen runnen met arduino:
    - op dit moment is er geen functie om de cpu te verwijzen naar arduino code
      Hoe wordt de cpu nu naar het address gewezen om de crownstone app lezen?
      Waar in de crownstone plaats ik de functie om naar de arduino code te gaan?
    - de arduino code en crownstone code zijn twee verschillende applicaties en kunnen
    elkaar niet zomaar oproepen
      er moet een handler komen die naar dat adres wijst zodat de code uitgevoerd kan
    worden
    - kijken in de crownstone code naar de handlers hoe die gedefineerd zijn
    - kijken in de linkerscripts van crownstone

31-10-2019
  - Werken aan verslag
    - paragraven vertalen, nieuwe derbij schrijven

1-11-2019
  - What is ram
  - What is flash memory
  - what is 6.2.1.7 verder schrijven
  - what is interupt: Onderzoeken
  - (kantoor werk: geen betrekking tot stage opdracht; Batterijen opladen)
  - vragen schrijven voor verder literatuur onderzoek
  - Aniket's presentatie gevolgd, pointers opgepakt:
    - Niet te veel tekst op de slides
    - Duidelijke advies of definition dat je wilt leren aan het publiek
    - Grafieken gebruiken die meer bekend zijn in het algemeen (maakt het makelijker
    voor het publiek)
    - Niet te veel specifieke namen gebruiken tijdens de presentatie (niemand onthoud ze
     daarna)
```

```
    - verwijs materiaal komt in het rapport/verslag en niet op de presentatie slides

4-11-2019
  - Risico log gemaakt
  - onderdelen in verslag vertaald

5-11-2019
  - Issue Tracking program installeren\
  - Issues opschrijven
  - Gesproken met Anne:
    - Complex use case: Arduino roept crownstone code op voor bluetooth functie (niet
    gelijk naar de radio op het bord)
    - make fork van bluenet met arduino compatible version van crownstone code. (write
    own code because of copyright)
    - fork sandeep's arduinonrf5 repo (voor de toolchain etc)
    - maak example repo voor example code voor arduino code op crownstone
    - Issue tracking kan je op git doen bij de forked bluenet git repo

7-11-2019
  - Stakeholderanalyse verwerken in verslag
  - Presentatie maken voor 50%-presentatie op school

8-11-2019
  - Presentatie verbeterd
  - Stakeholder tabel aangepast
  - Risico log aangepast
  - interrupts in linkerscripts: onderzoek

11-11-2019
  - verslag bijgewerkt
  - onderzoek: wat zijn linkerscripts
```