Keeping water under surveil-lance

Telecommunications and Power Supply

L.B. Kunkels & J.H. Belier

Technical University of Delft

TU Delft

Keeping water under surveillance

Telecommunications and Power Supply

by

L.B. Kunkels & J.H. Belier

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Monday June 27, 2016 at 3:30 PM.

Student numbers:	4301536	
	4324072	
Project duration:	April 18, 2016 – July 1, 2016	
Supervision:	Dr. ir. A. Bossche,	TU Delft, supervisor
	Ing. J. Bastemeijer,	TU Delft, supervisor
	Ir. E. van der Putte,	SkyDowser, proposer
Thesis committee:	Dr. ir. B.J. Kooij,	TU Delft, chair
	Dr. ir. A. Bossche,	TU Delft, supervisor
	Dr. ir. A. van Genderen,	TU Delft, jury
	Ir. E. van der Putte,	SkyDowser, jury

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

This study is done as part of creating a groundwater-analyzing sensor system. This thesis elaborates on finding a cost-efficient way of implementing a telecommunications channel between a measuring node and a data-server, together with forming an energy-efficient power supply system that powers a system over a cable of 60 metres.

The cellular network is used as the main method of transmitting data between a node and a server. To be able to connect to the cellular network, the SIM5320E chip has been implemented into the system, featuring access to both 2G and 3G technologies, which is controlled by sending AT commands. These commands are sent automatically by a C++ program, which is executed each time the module is turned on. The system will return to sleep after successful data transmission. Data communication is done through the use of the HTTPS protocol, but in the future CoAP might be a better alternative to reduce the energy needed per transmission by reducing the amount of bytes added to the data packet.

A power circuit has been designed that supplies each component of the groundwater-analyzing sensor system with the right voltage. For energy efficiency, a rechargeable Li-ion battery rated at 3.6V is combined with a solar cell to make sure the complete system is supplied with a steady power stream and can work autonomously for at least two consecutive years. A power-over-bus system is implemented to provide the subsystem at the end of the cable of 60 metres.

Theoretical research is done on the matter of information communication and the use of the network technology LoRa. The information communication research points out that different protocols such as MQTT and CoAP reduce the amount of bytes necessary to successfully transmit data. Due to the research on LoRa it has been discovered that the usage of LoRa could greatly reduce energy costs and nullify yearly costs and would be a possible improvement in the future. Other possible improvements in the future besides using a different network technology such as LoRa are improving the error detection of the control program and reducing energy consumption during transmissions by e.g. using a different protocol.

Preface

This thesis is written as part of the Electrical Engineering Bachelor programme, provided by Delft University of Technology. To successfully obtain the degree Bachelor of Science and to that end completing the Bachelor programme, a group project should be executed and judged to be satisfactory. One of the requirements of the project is to divide the group, consisting of six people, into subgroups and each subgroup writing their own thesis about their part in completing the project. This thesis is the product of that project, a project about the telecommunications and power supply components of a product for keeping water under surveillance.

However, this thesis and research could not have been completed without the help provided by several people. As a token of gratitude, we would like to thank the following persons specifically:

- Dr. ir. A. Bossche and ing. J. Bastemeijer, for their continuous support and insight throughout the project.
- Ir. E. van der Putte, for granting us the opportunity to work on a unique project as this and providing us insight in working with a client.

L.B. Kunkels & J.H. Belier
Delft, June 2016

Contents

1 Introduction

Clean water is one of the most essential things in life, but unfortunately water sources do not provide consistent clean water. The start-up company SkyDowser is planning to provide products to inform people about the state of groundwater. Currently SkyDowser has developed a drone to map the contents of the subsoil in water stressed regions. While these drones give excellent results, the cost of maintaining real time water surveillance at specific places is too high compared to the utility it provides. Therefore, SkyDowser searched for students to design a diver to make real time water surveillance possible at a low price. These divers could be used to monitor the water levels and quality in water wells.

SkyDowser wants to start deploying their diver in Africa, because real-time water surveillance is not yet available in that region of the world, due to the fact that other products like this are too expensive. If farmers could get insight of their water resources they could set up their irrigation in a more sustainable way and could possibly saves great parts of their yield. The first prototype will be built for Kenya, because groundwater control is a major issue there. This also gives the students a specific area to develop the prototype for.

The diver to be developed should be able to measure the ground water level, measure the conductivity of the water and measure the temperature. These quantities will determine if the water is suitable for irrigation. Furthermore, it needs to be possible for customers to add new sensors in case better sensors are developed or the customer wants to receive other information about their water supply. In order for the customer to use the data of the measurements, the data will be made available to them through the SkyDowser server, so the diver must be able send the data to this server. Furthermore the system needs to work autonomous and be modular.

This thesis will describe the telecommunications and power supply system of the diver. To provide a good understanding of the inner workings of this system, the thesis is subdivided into several parts of the project. Firstly, the overall concept of the project will be presented. After this the requirements will be defined, followed by telecommunications concepts and an explanation about the chosen module and its implementation. The next chapters will discuss the codes used to control the telecommunications system, the power supply and after this test results will be provided and elaborated on. Finally, the thesis will close with a conclusion and discussion, which include possible improvements on the system. At the end of the thesis the appendices and bibliography can be found. Connections between the different components are shown in fig. 1.1.

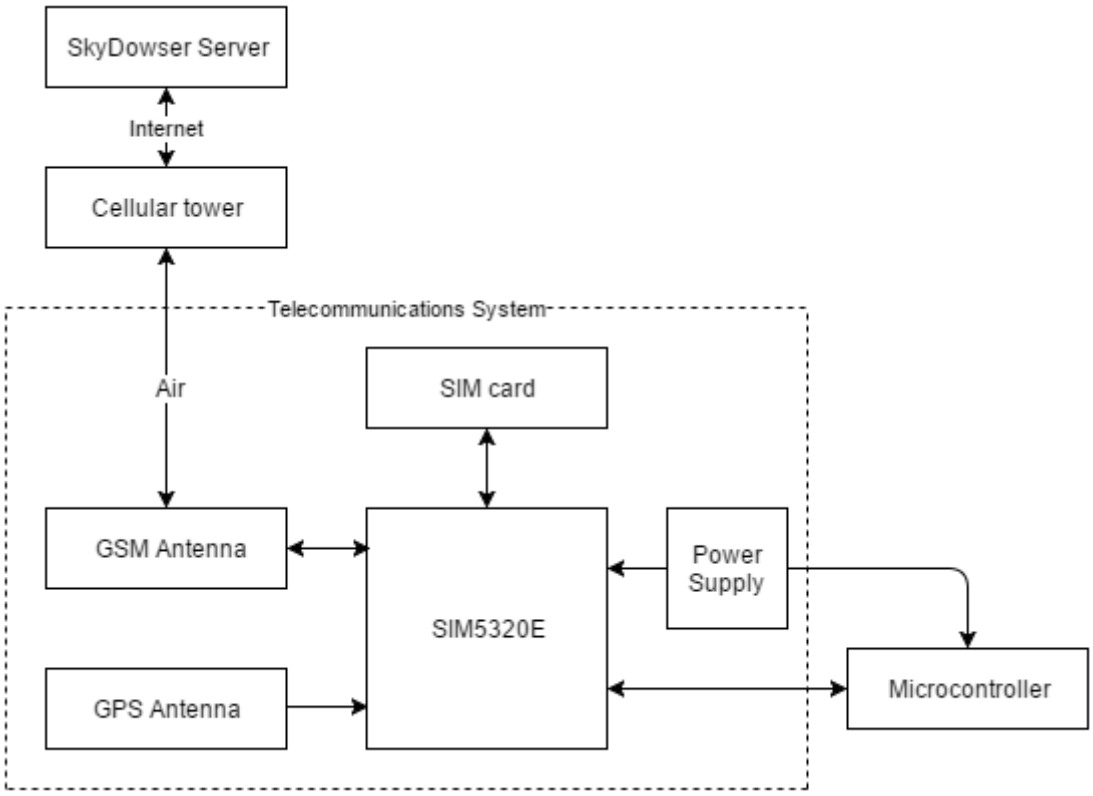

Figure 1.1: Schematic overview of the telecommunications system

2

Overall Concept

2.1. Function analyses

In this chapter the overall concept of SubDowser will be discussed. The overall concept is designed by the entire BAP group in collaboration with the external proposer, to the end of meeting external demands. The functional analysis gave a better understanding of what the main functions of SubDowser are. The functional analysis is shown in figure 2.1.

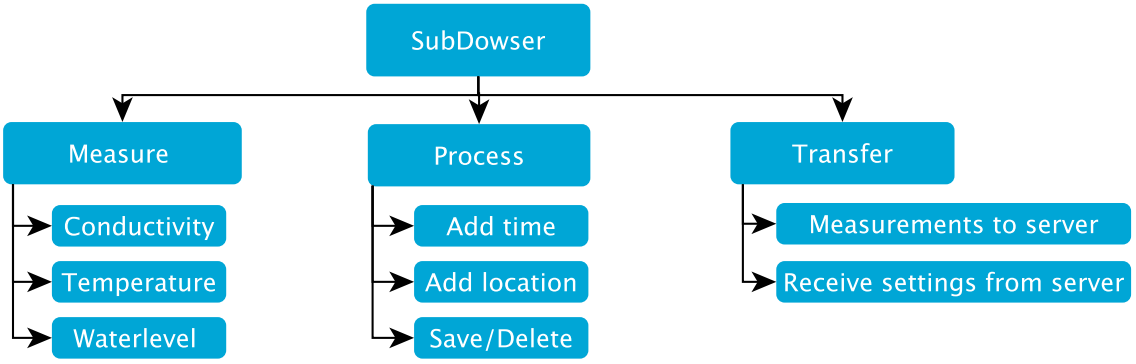

Figure 2.1: Function analyses of the complete system

Three functions comprise the primary functions of the system. Measuring several states of groundwater is the first function. The SubDowser needs to measure the ground water level compared to the ground surface level. The SubDowser must measure the conductivity of the water and the SubDowser must measure the temperature of the water. The second function deals with the processing of the data. The measured data needs to be processed, other data (time and location) needs to be added and saved. The third and last function deals with the telecommunications part. The measured data needs to be sent to the SkyDowser server according to the imposed settings. To divide the whole assignment into three parts, one group of student will focus on designing the sensors, another other group will work on the control of the system and the last group will focus on data communication between the servers of SkyDowser and the product. The design of the power supply circuit of the system is also done by the data communications group.

2.2. Requirements

For the entire system to function properly and to meet the demands of the external proposer for the system, the BAP group constructed a list of requirements. The most important requirements of the overall concept are shown below. The full list of requirements is defined in appendix C.

1. The product needs to measure the level, conductivity and temperature of the groundwater.

2. The product needs to send the measurement data to the server of SkyDowser.
3. The product needs to be able to adjust settings, such as measuring and sending frequency, according to changes given by the SkyDowser servers.
4. The product needs to be Plug & Play for the customer.
5. The product needs to be modular.
6. The costs of the product materials need to be lower than €50, assemblage not included.
7. The product needs to function autonomously for two successive years.
8. The product needs to fit in a borehole with a minimal diagonal of 6 centimetres.
9. The product needs to be able to work in a borehole with a maximum depth of 60 metres.

2.3. Interfaces

The three groups separately must keep the requirements in mind when designing their subsystems. To separate the overall concept into three parts, clear interfaces need to be agreed upon to mark the boundaries of each subsystem.

2.3.1. Sensor subsystem

The subsystem that needs to measure the groundwater level, conductivity and temperature of the water will do this by using sensors. Both analog and digital sensors will be used. The task of the control subsystem will be to digitise the voltages of the analog sensors using ADC's with a resolution of 12 bits and to communicate with the digital sensors using I2C.
Further details about the sensor subsystem can be read in the thesis *"Sensorsysteem voor SubDowser"* [37], written by R.C. van Beelen and S.C.A. de Groot.

2.3.2. Control subsystem

The subsystem that controls both the sensors and the telecommunications uses microcontrollers to this end. It awakens the subsystems, commands them to perform the needed tasks and if everything is satisfactory sends them back to sleep. It also controls when the power supply DC signal should be sent to the sensors or not, so that energy efficiency will be kept as high as possible. The complete description of this subsystem and its inner workings are available in the thesis *"Keeping water under surveillance - Control"* [38], by M.A. van Remundt and T.J. Witte.

2.4. Design of the overall concept

To get a better understanding of what the total concept will look like in practice, the design is described and illustrated below. The general concept of the SubDowser system will look as displayed in fig. 2.2.

The part of the system that is within the borehole will be made waterproof. Besides it being waterproof it should be possible to add new sensors to this part of the system, so the system should be modular. The encasing of the part above ground will be resistant against high temperatures, rain and dust.

The main power supply will be placed in the part above ground. From there the telecommunications system, both microcontrollers and the sensor system will be powered. To power the bottom part of the system, the power will be transferred over the communication link between the two microcontrollers, making it a power over bus system.

The design choices that are named here concerning the telecommunications subsystem can be read in chapter 3. Design choices concerning the other subsystems are noted in the previously stated theses.

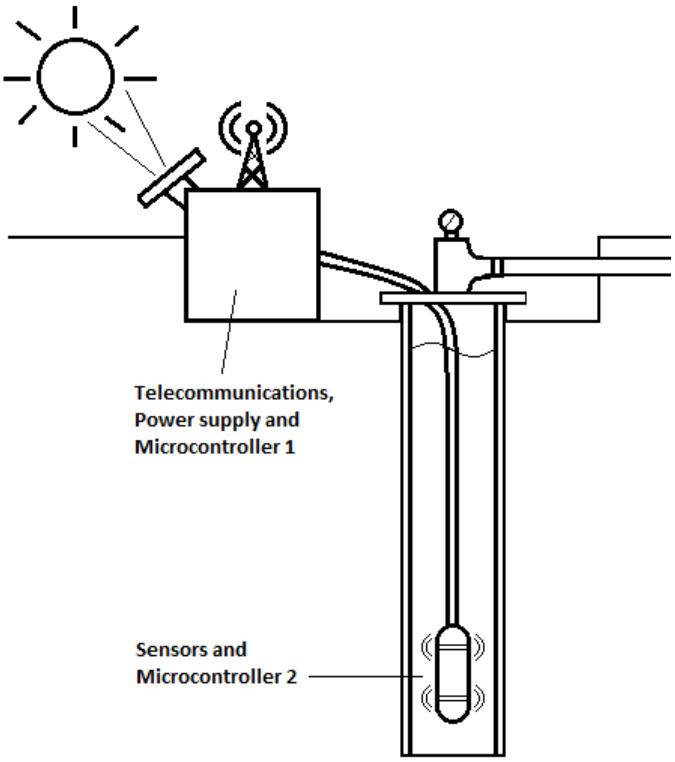

Figure 2.2: Schematic of the total system

3

Requirements

As noted in the introduction, one part of the SubDowser product is the telecommunications system. This subsystem establishes the connection between the microcontroller and the servers of SkyDowser. Therefore, the telecommunications system should be able to send the data provided by the microcontroller to the servers and receive data back. To successfully accomplish this goal, the telecommunications system needs to meet a number of requirements. As this thesis also covers the power supply, additional requirements due to power supply limitations are listed as well. Note that the realization of the cost requirements is not performed in any chapter, but is instead documented in appendix D.

1. **Functional requirements:**
   - The obtained data should be transmitted to the SkyDowser data servers.
   - The module should be able to receive data from the SkyDowser servers and send it to the microcontroller.
2. **Non-functional requirements:**
   - 2.1. **Product requirements:**
     - 2.1.1. **Usability:**
       - The yearly costs are not allowed to exceed 10 euros.
       - It should be possible to change the measurement frequency from the servers.
     - 2.1.2. **Reliability:**
       - It should be possible to send the measured data a maximum of 1 time per half hour to the SkyDowser servers.
       - The system should be operational in a mountainous area, at least to a height of 1000 metres.
       - The system should be able to perform 2 years autonomously before maintenance is needed. Production errors are not taken into account.
       - The measured data should be labeled with the time with 1 second accuracy. The time should be displayed in UTC.
       - The location of SubDowser should be transmitted at least once to the servers, accurate within a radius of 100 metres.
       - In the end all transmitted data should be available without errors on the SkyDowser servers.
       - The system should be able to communicate using a communications standard that will still be in use 5 years after the design of this system.
     - 2.1.3. **Safety:**
       - The subsystem needs to meet the European General Product Safety Directive 2001/95/EC.

- The data should not be obtainable by other parties during transmission.
- The product should be resistant to a maximum temperature of 70 degrees Celsius.
- The product must satisfy the IP-64 [5] coding.
- The product must satisfy normal mechanical impact, with a minimum of IK-07 [5].

2.2. **Organizational requirements:**

2.2.1. **Operational:**

- The product must be easy to use by the end user.
- It should be possible to operate the product by the end user, without help of SkyDowser.

3. **Design choices**

- The amount of data received from the SkyDowser servers each time will consist of a package of 6 bytes and will be input as a character array, excluding protocol headers.
- The received data from the SkyDowser servers should be output as a string.
- The amount of data sent to the SkyDowser servers each time will consists of a package of strings, each string consisting of 70 bytes, excluding headers. Multiple strings can be sent per transmission, depending on settings regarding transmission and measurement frequencies [38].
- The GPS data (location and time) should be output as a string.
- The module should be able to operate using the 3G network.
- Data communications should be performed using the HTTPS protocol.

4

Concepts

At the beginning of the project, several alternative ways to implement the telecommunications system were sought and analyzed. There are many ways to implement a wireless communications system and several different technologies to use. At the start of the project some of the requirements from the previous chapter were already known, thus it was possible to reduce the amount of options available. The remaining solutions to implement the network needed for wireless communication that fulfilled the early requirements include the already available cellular network, constructing a new base station network or building an ad hoc network.

4.1. Choosing a communication method

4.1.1. Cellular network

In Kenya an open wireless communication system is already in operation namely the cellular network. Using this network removes the need to build a new network. In Kenya 95%[15] of the population has cellular coverage, meaning a high likelihood of receiving signal on the spot of a borehole. In comparison with other technologies, this network requires a contract with a mobile operator while most others do not. This increases the configuration costs and adds a yearly cost to be paid to the operators. The advantages include not having to construct a new network. This increases the ease of adding a low amount of nodes in an area, as building a private network starts to become beneficial only after a certain amount of users are within the range of a gateway. Another drawback of the cellular network is the relatively high energy consumption in comparison to systems stated below.

4.1.2. Alternative already installed "open" networks

Next to the cellular network, two other promising networks are being setup by third parties, for example SigFox and LoRa. SigFox is a single company installing network facilities around the globe to provide low bit-rate network connections for sensor nodes. LoRa is an open standard for communication between sensor nodes and base stations. At the moment LoRa is being installed by companies around the globe. In the Netherlands KPN took the lead by planning to provide coverage nationwide. Both of these networks show growth potential, but no signs have yet been found of networks being installed in Kenya. SigFox tries to accomplish the same goal as LoRa, however the gateway technologies are not supplied to developers, as SigFox does not allow the construction of a developer's own network.

4.1.3. Private networks using base stations

Another solution would be to install a base station. Each sensor node would transmit their data to the installed base station, which in turn relays the data towards the SkyDowser servers. To construct such a network, a choice has to be made about the kind of technology to use. Different technologies exist to this end and the most attractive options seemed to be LoRa. The problem with installing a base station is the additional cost of maintenance and the relatively high start up costs. Modules have been represented in the range of 500 Euros. This makes this solution viable only if multiple nodes were to be installed in a region. Benefits of using networks like LoRa are reduced power consumption and non dependent on third party operators. Other technologies include the IEEE 802.3 standards, which is limited by the range.

4.1.4. Ad hoc networks

In an ad hoc network, data is relayed from node to node. Some sort of end-node is required to serve as the connection between the SkyDowser servers and the sensor nodes. Building an ad hoc network would require nodes to be in close proximity to each other, which might not be true in the case of Kenya's borehole distribution. A lot of boreholes would have to participate in this project to work. This can only be completed by a joint effort of borehole owners to install these nodes. This is mainly the reason why this option was not chosen. Another solution would be a hybrid between ad hoc and cellular networks for nodes distant to other nodes. A few of the technologies to accomplish this goal are Zigbee, WiFi and Bluetooth. SkyDowser informed an average distance larger than a kilometer between nodes was usually the case, however these technologies could only reach these lengths when using special antennae configurations and high energy usage, which made this option less attractive.

4.2. Choice

Eventually the main preferred choices were presented to the client, those being using the cellular network and building a network using LoRa technologies. For more certainty, the client chose to use the cellular network, though using LoRa would be a promising option in the future. To this end, LoRa technologies will only be theoretically explained in this thesis, but the cellular network will actually be implemented.

5 Module

The cellular network has been chosen as the method to enable communication between nodes and the SkyDowser server. Elements required to communicate with this network include an antenna, a SIM card and a telecommunications module. Reports have shown that GSM towers will be disappearing next year from the biggest carriers in the USA. No reports have been found regarding the disappearance of GSM towers in Kenya, but this is something to look out for. For this reason a module was chosen that enables both technologies from the 2nd generation and technologies from 3rd generation to increase the life expectancy of the product.

Many companies exist in the telecommunications chip market that offer telecommunications modules to connect M2M or IoT solutions to the cellular networks. A few of the biggest companies include SIMCom/TeLiT among others. For prototyping purposes it is chosen to use a SIMCom module. This company offers a broad range of modules with different specifications. The SIM5320E module supports all communication technologies all the way up to HSPA. This chip also supports many carrier frequencies to be used around the world. For prototyping purposes a development board created by AdaFruit [18] has been used.

5.1. SIM5320

The SIM3520 is a quad-band GSM/GPRS/EDGE and dual-band UMTS/HSPA telecommunication module, as specified in the datasheet [30], which means that the module supports both 2G and 3G telecommunications. Configuring and controlling the telecommunications module is done through the use of AT commands. These commands are in string format. These commands are documented online [29] and are created to provide an easier means of communicating with a modem. Several instructions have to be sent before being able to transmit and/or receive data. The telecommunication module has to be connected to a SIM card to be able to connect to the cellular network. The SIM card will unlock its features to connect to the cellular network if the PIN-code is entered correct.

5.1.1. AT Commands

Even though the module is able to send texts, make phone calls and much more, these functions are left untouched. As stated in the requirements chapter of this thesis, the system will send and receive solely with encrypted data. CoAP is chosen as the application layer protocol for transmission of sensor data and HTTPS for registration purposes, as will be discussed in chapter 10. For prototyping purposes HTTP is used and if enough time is left the CoAP protocol will be build. The HTTPS protocol is specifically designed for secure communication over HTTP. This protocol is supported by the SIM5320 module.

Preparing the module for communication over HTTPS/TCP/IP is documented in chapter 16.5.5 of the AT command manual[29]. First, *AT+CHTTPSSTART* will acquire a HTTPS protocol stack. Next, using *AT+CHTTPSOPSE=<host>,<port>,<server type>* will set up a session with the indicated host through either port 80 (used for HTTP servers) or port 443 (used for HTTPS servers). Server type defines if the type of server is HTTP or HTTPS. Now the module is ready to send requests to the connected server, so data can

be either received (through a GET request) or transmitted (through a POST request). Sending this request is done via the instruction *AT+CHTTPSSEND=<length>*, where length indicates how many characters are contained in the request line. Upon receiving this command, the SIM5320 responds by asking what request will be done and thus either a GET or POST request can be done. If this is done correctly according to syntax data, the request is completed. In the case of a GET request, the received data can be read by using *AT+CHTTPSRECV=<max receive length>*. Finally, the HTTPS session will need to be closed with *AT+CHTTPSCLSE* followed by *AT+CHTTPSSTOP* to stop the HTTPS protocol stack. The flow of this process is depicted in fig. 5.1.

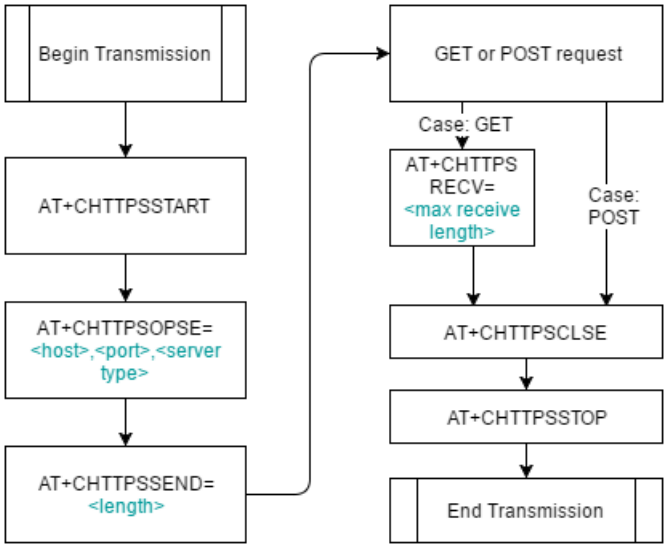

Figure 5.1: Data transmission through HTTPS

5.1.2. Tasks

In this project, the module will be required to perform a few tasks before being sent back to sleep by the control system (see [38]). Once awakened the module will first send the data provided by the control system to the server. To ensure the data has been correctly arrived at the server an acknowledgment message will be sent by the server, so the module will need to be ready to receive this message. If the server told the module transmission was not satisfactory, the module will resend the data and this will go on until data transmission is done correctly. The server could count the number of consecutive faulty transmissions and indicate a possibly broken sensor to the user, but as the operations of the server are out of the scope of this project this will not be elaborated upon. After the acknowledgment has been received, the module is instructed by the control system to either go back to sleep or to request the server to transmit data. This data, unlike the one sent by the module, contains settings to be sent to the microcontroller and will dictate how often the sensors will need to perform measurements. Finally, if this transmission has been completed as well, the module will be sent to sleep as it has completed the tasks to be performed. Though this is the standard telecommunications cycle, there are some cycles that include one extra step at the start. The control system determines whether GPS data will be requested or not, as it is not necessary to determine this every single time a transmission cycle occurs. The cycles in which the GPS data is to be requested, this is done before sending the data provided by the control system to the server. Because the GPS data is obtained before transmission it can be added to the to be sent data package, so that the package contains the current location and time instead of those from the previous transmission.

6 Implementation

For the development of the product two development boards were used: the *Adafruit's FONA 3G Cellular + GPS Breakout* [18] and a microcontroller development board created by NXP based upon the LPC1768 microcontroller [23]. The FONA 3G Cellular is controlled by the LPC1768 microcontroller over a UART connection. Though these development boards were used, in the final product the SIM5320 chip (which is the base of the FONA 3G Cellular) is implemented in a PCB. This is done because the development board enables the use of many features the SIM5320 contains that are not necessary for this implemented system. This chapter will therefore elaborate on the design of the PCB, but also shortly addresses the use of the GSM and GPS antennae, as these antennae are necessary to enable transmission of data and obtain the current location and time.

6.1. PCB design

A PCB is designed to responsibly connect the components of the SubDowser above ground. This PCB is built using KiCAD [6]. KiCAD is an open source program developed by research organization CERN. The designed circuit is relatively simple, therefore the wiring/tracing was done manually. The first step in the design is to create a schematic of all the components and its connections. The second step is to link the footprints to each component. As no directly implementable footprint was available of the SIM5320, the SIM5320 footprint had to be manually created. The third step is to place the components on the PCB and add traces to create the connection between the pins. Building this PCB, design recommendations created by the manufactures were considered. The SIM5320 Hardware Design Guide was consulted [30], the TI SN65HVD1780 datasheet [33], the LPC1768 data sheet [23], the microchip MCP73834 [20] and the LM3670 [34]. Currently only the schematic is built. This schematic can be found in figures 6.1, 6.2 and 6.3. Labels are added to provide a clear picture of the designed PCB.

6.2. Antennae

This section is dedicated to choosing the antenna for the transmission of data. The same antenna has been chosen as was recommended by AdaFruit to be used with the 3G FONA Cellular breakout board [18]. The antenna used in combination with this board is developed by Dongguan Boju Electronics[14]. Also an active antenna recommended by AdaFruit will be used for the GPS communication. To connect these antennae to the SIM5320 module attention should be paid to the connection characteristics. The connection between the antenna and the SIM5320 module should have a characteristic impedance close to 50Ω. This is a consequence of the load in the SIM5320 being close 50Ω. According to Ulaby [36], if the characteristic impedance of a transmission line is equal to the load, then no reflection will occur. The reflected signal causes interference with the incident signal and could therefore inflict errors. Company Polar provides an application [17] to calculate the characteristics impedance of a transmission line trace over a PCB. This will be used to approximate this 50Ω transmission line.

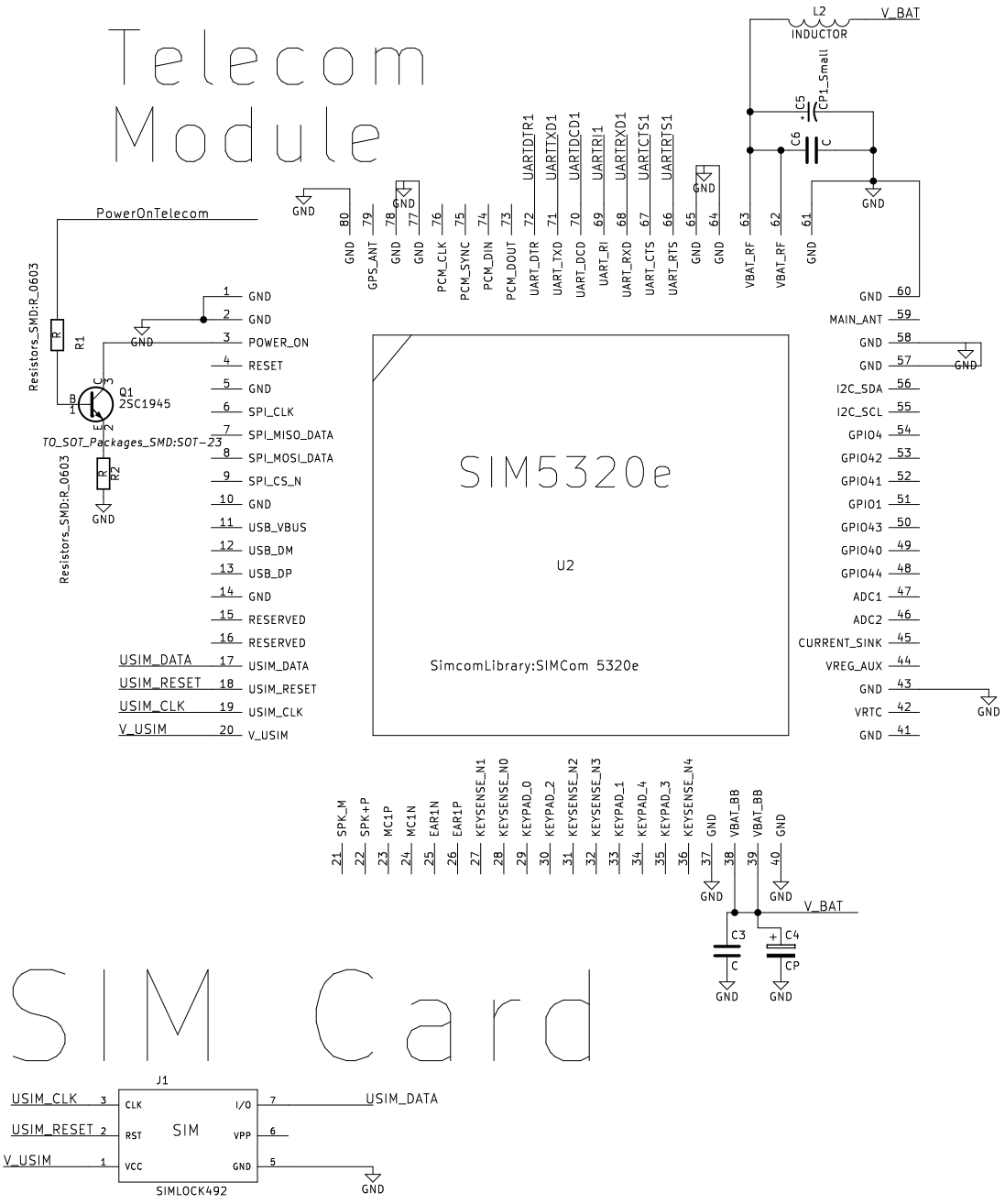

Figure 6.1: Schematic containing telecom module and SIM Card holder

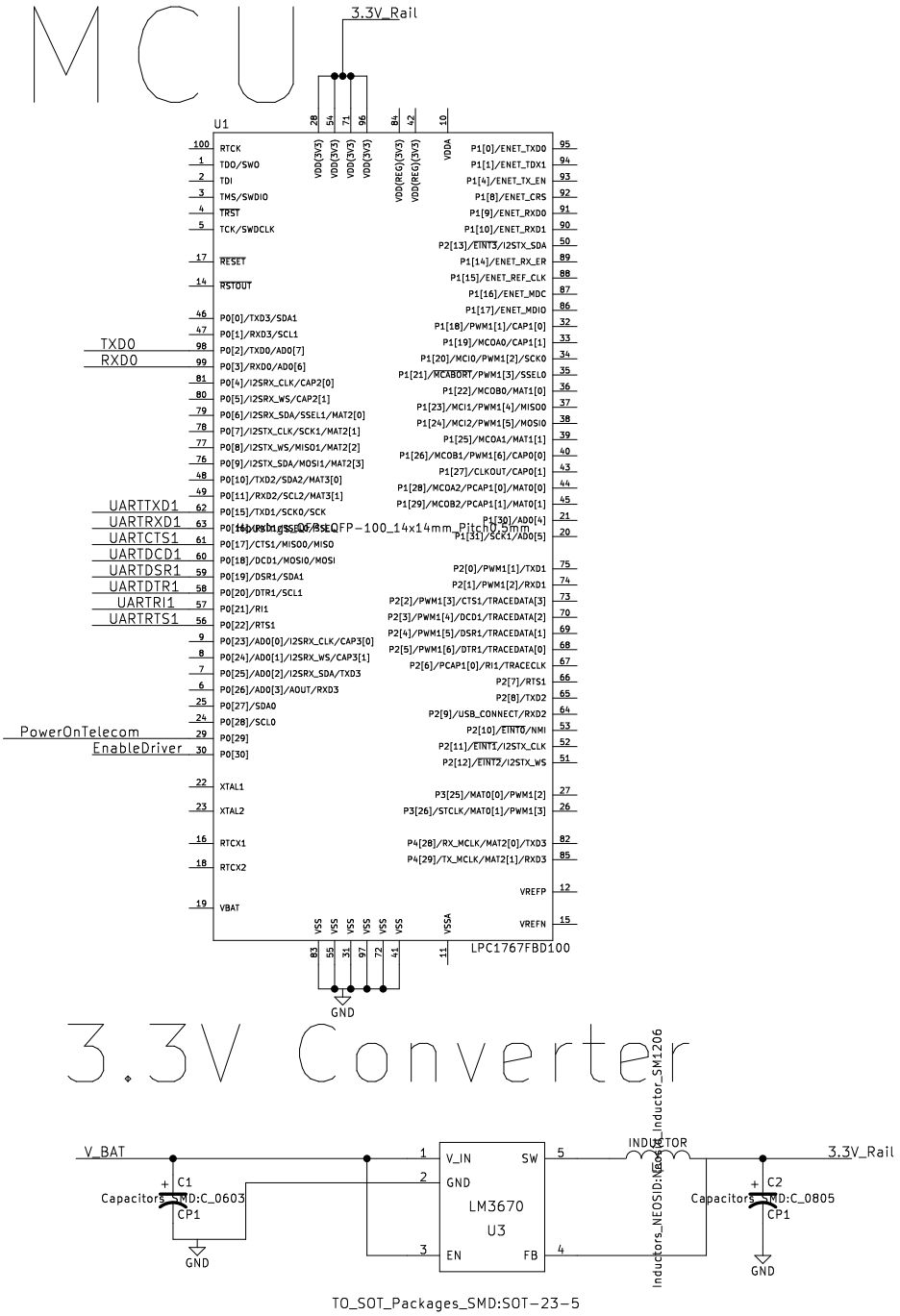

Figure 6.2: Schematic containing MCU and 3.3V voltage converter

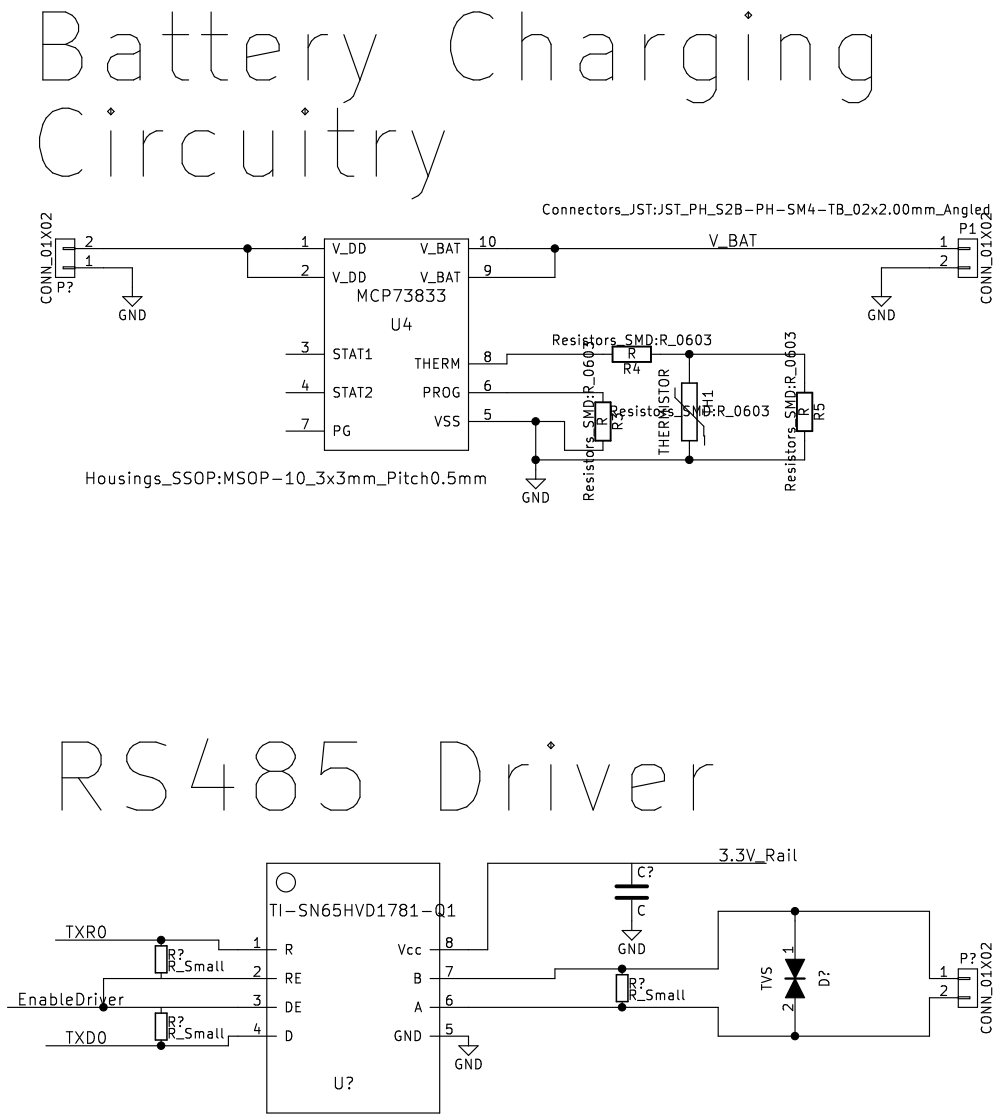

Figure 6.3: Schematic containing the battery charging circuitry and the RS485 driver

7

Coding

The cellular module is controlled using a microcontroller. The microcontroller is programmed to perform these controls. The program (see appendix A) will determine what commands to send to the SIMCom module and will also parse and make sense of the information it receives back from the SIMCom module. The program is written in a structured way to enable third parties to edit to code with ease. The structure is as follows: at the top the constants are defined. These are defined upfront and will be used consistently throughout the code. Pins are defined below that, to make sure the microcontroller can communicate with the module through the use of UART and a reset signal is available. Next, variables and functions are declared to be used by the program. After all these declarations the main code follows, which is fully executed during each cycle (when the telecommunications module is awake). Finally, all functions called by the main function are written at the end of the code. Elaboration on the inner workings of functions are available as comments in the appendix. It should be noted that at the time of writing this thesis, the main functions it should perform are implemented, however details on error correction and other comparable functions are not yet implemented in this version of the program.

7.1. Communication

The SIMCom module is controlled by sending natural language commands. These are the so-called AT commands, partially standardized by the organization ETSI. The code will send these commands and starts with initialization steps. After the configuration the main function will start sending and receiving data as described in chapter 5. The microcontroller will send the module to sleep and the program will run again after a specified time. How the microcontroller functions is explained in the thesis of the control subgroup (see [38]).

7.2. Error correction

As many errors can occur, it has been attempted to take as many of them in account as is necessary to provide a reliable system. The biggest problem that can occur is when AT commands just don't get received correctly anymore. Testing has provided no insight in what the actual cause is, but turning the module off and on again seemed to solve this problem most times. Therefore, the codes keep track of the amount of times the message "ERROR" has been returned consecutively by the module. If this amount has reached a certain value, a hard reset will be performed by pulling the reset pin of the module low for 100 ms. Now the program will start again from the beginning.

However, this hard reset will only be performed in the worst case. Other errors, like network faults, are detected by the cellular module and are returned as an error code. Reading which error code has been returned will result in the program calling for different functions that should send the right AT commands to correct the problem. This way, it can be ensured that probable errors can be fixed quickly, with as little delay as possible. At the time this thesis was written, the code did not yet include these specific detection lines, but could easily be added.

8

Power Supply & Energy Saving

This chapter covers the part of the solution that provides power to each individual component. The components in our solution use constant low-voltage rails. The power supply is required to power devices with a total of three different voltages and to this end the circuit is to be designed in such a way that each component of the system receives the correct voltage.

8.1. General overview rail distribution

To illustrate the total design of the power circuit, a schematic overview is displayed in fig. 8.1.

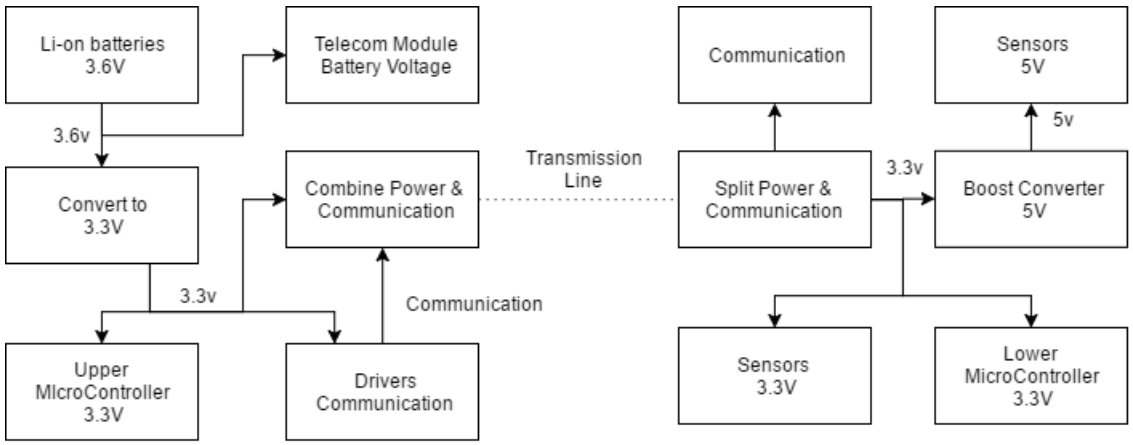

Figure 8.1: Power distribution

8.2. Power Requirements

As different voltages are required for each component of the system, different rails and power supplies are needed. Necessary parts are:

- Voltage rails rated at 3.3V, 5V and a rail rated at 3.4V-4.2V. The latter being the voltage supply for the SIM5320 module.
- Li-ion battery rated at 3.6V. This battery is the main power supply of the power circuit, as all other voltages are obtained through up- or down-stepping of this voltage source. Also, the battery is rechargeable.
- Small solar cell to generate durable power for the power circuit. As the Li-ion battery can be recharged, a solar cell is an ideal way of accomplishing this.
- A boost converter, to step up the voltage of 3.3V to 5V, to be supplied to one part of the sensor system.

- A buck converter, to step down the voltage of 3.6V to 3.3V, to be supplied to the microcontrollers and to one part of the sensor system.

The following sections will elaborate on the choices made concerning using specific products as components in the power circuit.

8.3. Battery

To provide energy to the components some energy source has to be tapped. In the case of powering the device solely by solar cells, a varying energy supply would be the result. To provide a constant stable stream of power a battery is added. As the battery is to function together with a solar cell to create a stable supply, the battery is required to be rechargeable. Alkaline batteries can therefore not be used in a rechargeable system. The three main battery composition that do satisfy this requirement are lead-acid, NiCd/NiMH and Li-ion batteries. Note that the maximum temperature is usually below 70°C, which is the temperature the system should be able to function on, so additionally the battery should be encased by a material that can provide the right temperature for the battery.

8.3.1. Alkaline

Alkaline batteries are, as stated above, not rechargeable. However, if the life-expectancy is high enough to satisfy the requirement of the system being autonomous for 2 years, they could be used instead of a rechargeable battery system if it also satisfies other specifications. Alkaline batteries usually operate on a voltage of 1.2V, so at least 3 would be needed to deliver 3.6V to the system. They are operable withing a temperature range of -20 to 55°C, but the main advantage of alkaline batteries is the fact that they have a very low self-discharge rate, namely 0.3% per month [19]. The life-time of such a battery is often around 10 years, but only if the load draws very little current. As the SubDowser system should be able to transmit often and transmission draw a considerable current, this can not be guaranteed and as of such alkaline batteries are not very preferable in constructing a robust system.

8.3.2. Lead-acid

The lead-acid battery is the oldest type of rechargeable battery. This battery is able to supply high surge currents and has a low cost for its typical applications. Also, it features a low (relative to the other battery types) self-discharge rate of 5% per month at room temperature. However, a lead-acid battery is a large product and thus is not a favorable component in forming a compact system. Asides from this, this battery has a self-discharge rate of 3-20% per month[43], meaning a lot of power is lost without any efficient use. As the preferred system is required to be durable, this characteristic does not fit the image.

8.3.3. NiCd/NiMH

NiMH and NiCd batteries are smaller than the lead-acid battery, however the material costs are higher. They offer a high cycle durability [41], meaning they can be charged and discharged a lot of times without losing quality. Also, they are fast to charge, as NiMH needs 2 to 4 hours and NiCd only needs 1 hour to charge. The main drawback of these types of batteries however, is that they have a very high self-discharge rate, making them impractical for use in systems that only draw small currents. Temperature influences this rate and especially in the case of the NiMH battery the self-discharge rate becomes enormous, having a self-discharge rate of 36.4-97.8% per month [42] at 45°C. Since this system is supposed to be used in Kenya, where temperatures on the ground can easily reach 60°C, the system would only be able to last a few months at maximum. Also, the cell voltage of these batteries is usually around 1.2V, which means multiple batteries would be needed to deliver the correct voltage of 3.6V to the circuit. Another drawback of using these batteries is the fact that they need maintenance every 60 to 90 days, even every 30 to 60 days for NiCd batteries [4]. As it is important that the system is able to function autonomously for at least 2 years without needing any maintenance, this battery-type does not fit the system.

8.3.4. Li-ion

Li-ion batteries feature a high energy density and a much lower self-discharge rate than for example NiCd/NiMH batteries, as at a temperature of 60°C the battery discharges at a rate of 31% per month [40], which would mean the battery would last about 3 months without being used. This makes the Li-ion battery favorable for low-current systems, especially since it is low maintenance. The cycle durability is a bit lower than the NiCd/NiMH batteries though and manufacturing costs are higher. Also, this type of battery requires a protection circuit which limits voltage and current, as it has very low overcharge tolerance. An advantage of the Li-ion type is the high cycle life (to 80% of initial capacity) it features, namely 500-1000 cycles [4]. The recharge time is between 2 to 4 hours, which is about average compared to other battery types. Considering the limitations and strong points of the Li-ion battery, it is an attractive option for this system and as of such it is chosen to be used.

8.4. Voltage converters

In total three voltage levels are requested, as can be seen in fig. 8.1. The telecommunications module requests the Li-ion battery voltage of 3.6V. This leaves two voltage levels to convert to. For these voltage rails rated at 3.3V and 5V fixed buck/boost converters are used.

The converters were selected based on efficiency, output current and cost. Comparing different available converters with these characteristics in mind, the LM3670MF-3.3 buck converter [34] and the SP6641AEK-L-5-0 boost converter[31] were chosen. Other transformers of the same types often offered more features or a small increase in efficiency, but at a greater cost that outweighed the benefits of using those converters.

8.5. Solar Power

Solar cells have greatly reduced in price over the last couple of years. As the entire module is made to be energy efficient, a relatively small solar panel will suffice in supplying the battery with energy. The additional cost of adding a solar panel is relatively small. In order to connect the solar cell to the battery some circuitry has to be added to prevent possible problems. Some of the occurring problems include overcharging and leakage. The overcharging problem is solved by adding a voltage regulator after the solar panel. A Li-ion battery is about 80% charged if the voltage across its terminals is 4 volts. This charge level is in a relatively safe area. The solar panel will provide voltage rated at 6V. This voltage is higher than the specified 4V, therefore a voltage regulator will be implemented to reduce this voltage to a constant 4V. Solar cells do not provide a constant voltage during the day. The voltage level of the solar cell could even go below 4 volts. This could cause some of the energy stored in the battery to leak back into the solar cells, which is counter productive. Adding a diode after the voltage regulator will solve this issue as diodes only allow unidirectional current movement. The 4 volts supplied after the regulator is further reduced by the forward bias voltage of the diode. This issue is partially solved choosing a diode with a low forward bias voltage, for example a Schottky Diode. Schottky Diodes offer better performance voltage forward bias wise than PN-junction diodes, therefore such a diode was chosen to be used in this circuit.

Charging of the Li-ion battery should preferably be done at time periods during which temperatures are relatively low. This is because the Li-ion battery can withstand temperatures of 60°C only if it is not being charged [7]. During charging periods, this temperature tolerance drops to 45°C. Therefore charging should only be done if the temperature of the battery is below 45°C, or in the case of a temperature that exceeds this limit only slightly the charging current should be reduced.

8.5.1. Charge regulation

To both regulate the voltage used for charging the battery and making sure the battery doesn't get charged at high temperatures, an IC can be used. The company Microchip has multiple different battery management IC's available. The list gets reduced to four IC's when sorting out chips that do not fulfill the requirements named above, namely the MCP73833, MCP73834, MCP73838 and MCP73837 (datasheets are available on [9]). The first two are cheaper, but the second two feature a lower leakage current. Making the choice based on these facts, the MCP73834 is used to make the system as cheap as possible. If more weight is placed on the energy efficiency of the system, one of the other two microchips can be selected.

These microchips feature a temperature monitor, which means a range can be set within which the battery will be charged. As stated in the previous subsection, the battery should not be charged above 45°C, or below 0°C. To set this range, following section 6.1.1.6 of the datasheet [20] two resistors are placed in series-parallel with the thermistor. Using the formulas stated in that section, with the named 10kΩ thermistor with a sensitivity index of B=3892, the necessary values of RT1 and RT2 can be obtained. The equations are solved below:

$$R_{COLD} = 10 * 10^3 * e^{-3892(\frac{1}{298.15} - \frac{1}{298.15-25})} = 33027.3\Omega \tag{8.1}$$

$$R_{HOT} = 10 * 10^3 * e^{-3892(\frac{1}{298.15} - \frac{1}{298.15+20})} = 4401.64\Omega \tag{8.2}$$

Inserting the values obtained by equations 8.1 and 8.2 into the formulas stated in the section named above, it becomes clear that for the set range resistors RT1=834.688Ω and RT2=77579.5Ω are needed to set the preferred temperature range for this charge regulator.

8.6. Power-over-Bus

One problem with using only a battery above ground is that the power, which should also reach the sensors in the borehole, needs to travel over a maximum 60 metres of cable. Instead of adding extra wires to this end, which is expensive, it was decided to use a power-over-bus system. A power-over-bus system sends, as the name implies, power over the same bus used for data communication, which eliminates the need for additional wires. As the data communication stream is AC and the power stream is DC, a simple filtering circuit can be used to combine and split the AC and DC streams at the end-points of the cable. Figure 8.2 depicts the typical configuration used for such a system, which is also the same way it is implemented in this project. The capacitors and inductors are used as filters: DC can not pass through capacitors but can pass through inductors, whilst for AC it is the other way around. As data can be sent in both directions, a diode bridge is used to make sure the power is delivered to the load in the correct direction. Drivers are used to ensure the information passed through the 60 metres long cable is correctly received at the other end, which is documented in [38].

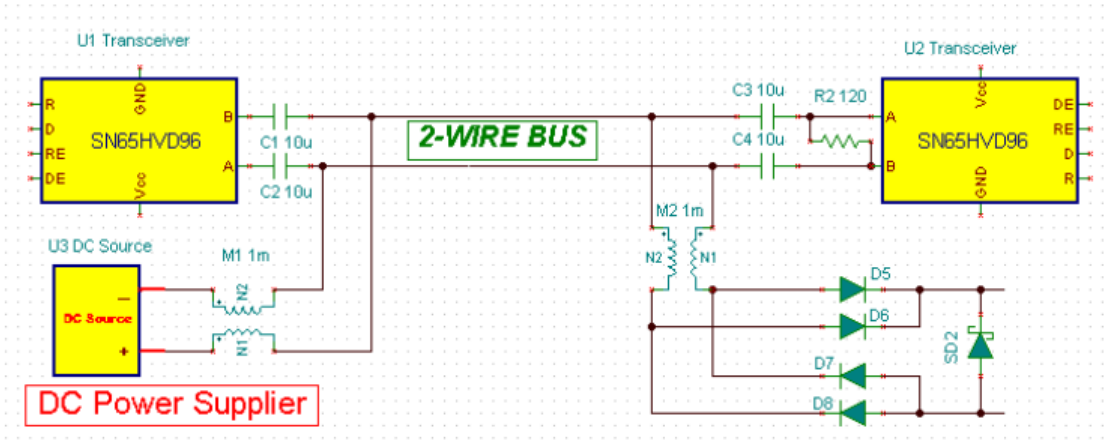

Figure 8.2: Power-over-Data lines [33]

8.7. Telecommunication congestion measurements

Usage of the cellular network results in different levels of energy usage. As at night the network is less accessed by people and therefore less occupied, energy usage is lower than when a lot of people are accessing the network at the same time. This is caused by increased transmission times in crowded networks.

As energy efficiency is an important issue, the SubDowser system should keep this problem in mind. To reduce energy costs, it might be a better idea to only transfer data during periods in which cellular activity is at a minimum. If the SubDowser system is ordered by the servers to obtain measurement data every half hour, an efficient way of transmitting this to the server would be buffering these measurements and combining them all into one data packet. This packet, which would contain all measurements performed during a

certain period, would then be sent to the servers at a convenient time. This method would increase energy efficiency, but exact figures can not be named as little research is done on the matter. A drawback of this method would form if consumers would actually like to receive measurement data every time it is measured, to be able to take actions immediately if the water is not suitable for irrigation due to raised salinity levels or other problems indicated by the system. If data is only transmitted once a day during a period of minimal cellular activity, this would not be possible. A compromise could be formed by choosing multiple times per day to transmit the data, each time in a period of lowered activity.

9 Testing

This chapter will describe the tests that were performed on the software and hardware during the project. Each test was performed to ensure that every part of the system would work as expected and to determine the limitations of certain subsystems.

9.1. Cellular Module

9.1.1. SMS

Firstly, the main part (the SIM5320E) of the telecommunications system was tested. To test this chip the Adafruit FONA 3G Cellular breakout was used, as stated in chapter 6.

The breakout was connected to a personal computer and tested using AT Commands. As a first test, it was important to know if the module functioned as it should: was it possible to connect to a cellular network and send data using this network? To be able to check this the 'AT Command Tester v23' (see [3]) was used, which provides an easy way to communicate with the module through AT Commands.

As the SIM card was inserted into the breakout it was necessary to enter the PIN-code connected to this card. This was done using command *AT+CPIN=xxxx*, where xxxx is the PIN-code paired with the SIM card. The next step was to select a network using *AT+COPS=0*, where 0 sets the network selection mode to "automatic" and thus an available network will be selected automatically. After this a message service was chosen using *AT+CSMS=1* and the preferred message storage was set to be the SIM card, using *AT+CPMS="SM","SM","SM"*. To be able to display the text messages in text format, *AT+CMGF=1* is used.

Since the configuration was now done, a test SMS was sent to a mobile phone and back. As these texts were correctly received on both ends it could be concluded a basic function of the module functioned as expected.

9.1.2. HTTPS

Next the network capabilities of the module were tested. As the module should be able to send and receive data through a secure data connection, the HTTPS protocol should be able to be used.

A few AT commands should be sent before being able to use this protocol and to that end, GPRS. First a PDP context should be defined, using *AT+CGDCONT=1,"IP","portalmmm.nl","0.0.0.0",0,0*. "portalmmm.nl" is the Access Point Name of Simyo, the provider this project makes use of. Defining the socket PDP context parameters is done with *AT+CGSOCKCONT=1,"IP","portalmmm.nl"*, after which connecting to this context is accomplished by sending *AT+CGATT=1* and *AT+CGACT=1,1*.

Now it should be possible to send and receive data using HTTPS. To accomplish a successful transmission a certain cycle should be completed, which was described in chapter 5.1.1. Using these AT commands a GET

or POST request can be done. To test the GET requests, the following data was sent using the AT Command Tester:

```
AT+CHTTPSOPSE="www.m2msupport.net",80,1
AT+CHTTPSSEND=72
GET /m2msupport/http_get_test.php HTTP/1.1
Host: www.m2msupport.net
```

The first line connects the module to the HTTP server "www.m2msupport.net" through port 80 (the standard port used for HTTP connections). The next line tells the module that a string containing 72 characters will follow. The GET request follows and handles a strict syntax: to correctly end the request, this string should be followed by an empty command in the AT Command Tester. This sends the control characters "carriage return" and "newline", which notifies the server of the end of the GET request.

Though this works in the AT Command Tester, it should also be done using C++, as this is the programming language used to program the used microcontrollers. The GET string stated above is translated in C++ as the following string:

```
GET /m2msupport/http_get_test.php HTTP/1.1\r\nHost: www.m2msupport.net\r\n\r\n
```

Not using the correct control character sequence at the end of the string results in the GET request not being performed correctly and thus the module not receiving any data from the server.

9.1.3. Communicating with the microcontroller

As in the final product the module should communicate with the microcontroller without any user input, tests should be performed to see if the communication between the microcontroller and cellular module is correctly implemented. To this end the microcontroller was connected to the module using breadboards. The correct connections of the pins are available in appendix B. Programming microcontroller with the C++ code described in chapter 7 and making use of "Tera Term"[11], an open-source terminal emulator to display printed strings on a personal computer through USB connection, this communication was tested. Several test lines were added to the code to print intermediate results, so that it could easily be seen if the cellular module sent responses back to the microcontroller. This was confirmed and thus the module could correctly communicate with the microcontroller.

9.1.4. Complete code run-through

The program currently is able to correctly exchange data with a server according to the tests. As of such, the program is implemented satisfactory and the minimal requirements set are fulfilled.

10

Information Communication

Communication between the server and the SubDowser system has a presumable big toll on energy usage. Compressing the total amount of data to be transmitted to decrease energy consumption could be done through the use of several techniques. In this chapter analyses are made of the amount of data to be transceived in several configurations. A configuration is defined as the rate at which connections are opened with SubDowser networks and the wireless technology in use. Also, recommended security measures and the format of data packets are defined.

10.1. State Diagram - Connection with SkyDowser servers

The SubDowser is capable of being in three different telecommunication states. The device is either powered off, in registration mode or is in active mode. Before being able to transmit data, the SubDowser will firstly register its device. During this state several properties are assigned values, which is elaborated later on in this chapter and in figure 10.1, which depicts the state diagram. This is done to accomplish better performance in the later transmission of actual sensor data.

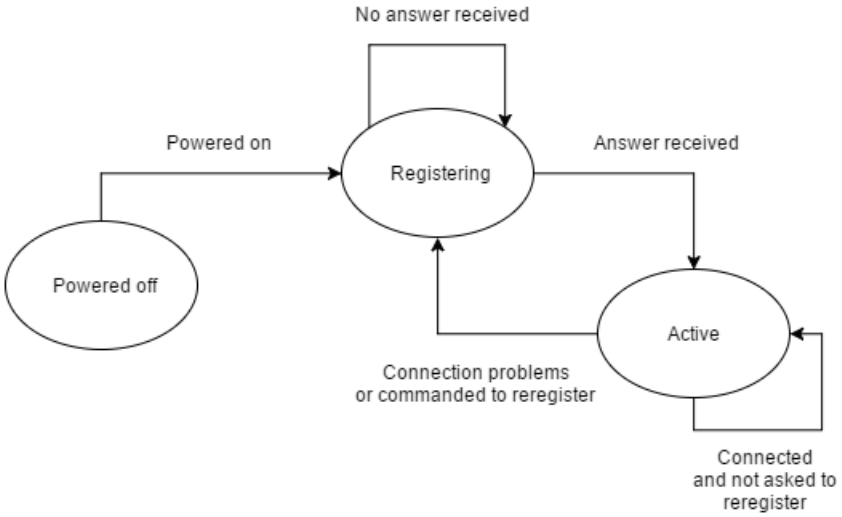

Figure 10.1: The state diagram of the connection between SkyDowser servers and the sensor node

10.1.1. Registration SubDowser servers

Before being capable to transmit sensor data, the sensor node first needs to register. During this stage, several properties are assigned values. The sensor node will define its connected sensor modules as the device was made to be modular. The SubDowser will do this by delivering all product id's and the corresponding invisible (as it is embedded in the module and thus not shown to the user) serial numbers of both the SubDowser and

its sensor modules. It will transmit the used application layer protocol and version, together with transmitting the used encryption method combined with a public key. For the last property the GPS location will be transmitted. After the SkyDowser server has received this packet, the server will answer by providing several IP addresses and domain names. These are the addresses the SubDowser is capable of reaching to establish a connection with the SkyDowser servers. The public key is sent to be used for encryption. In figure 10.2 an example of the structure of both the documents is given. The structure uses the JSON format. Registration occurs over HTTPS. This secure connection is important as the invisible serial number may not be stolen. If the serial number is obtained by a third party in the middle of transmission, then this party would be capable of registering to the SkyDowser servers and influencing measurement data.

```
{
  "device": {
    "id": "[productid]-[serialnumber]",
    "alp": "coap-[versionNumber]",
    "encrypt": "dtls-[versionNumber]-[publicKey]",
    "gps": "[gpsLocation]",
    "sensors": [
      {
        "id": "[productId]-[serialNumber]",
        "name": "[productName]"
      },
      {
        "id": "[productId]-[serialNumber]",
        "name": "[productName]"
      }
    ]
  }
}
```

```
{
  "server": {
    "publickey": "[publicKey]",
    "ip": [
      "[addr1]-[port1]",
      "[addr2]-[port2]",
      "[addrX]-[portX]"
    ],
    "domainnames": [
      "www.skydowser.nl",
      "server.skydowser.nl"
    ]
  }
}
```

Figure 10.2: On the left the registration message send by the sensor node. On the right the message send by the server to to sensors

10.2. Sensor data transmission (Active)

During this stage, sensor data is transmitted to the server. For maximum energy efficiency, several methods are considered to reduce the amount of data to be transmitted and received. In the next sections, analyses are made of the several protocols available and the final data packet.

10.2.1. MAC headers

Wireless communication technologies add MAC headers during transmission on the physical level for reliable transmission. Each added byte in the MAC header is additional energy used for communication, which is an unwanted occurrence, as the system should preferably be as energy efficient and durable as possible. Disappointingly, the amount of bytes included in such a MAC header is standard and can not be changed. Energy savings should therefore be done in the other parts of the transmission process.

10.2.2. Internet layer Protocol

In our analysis of data consumption the headers added on the internet layer form a significant part of the data packet transmission. The internet layer protocol is level 3 of the OSI model. Three kinds of headers exist to communicate over the internet, namely IPv4 [26], IPv6 [22] and 6LoWPAN. IPv4 and IPv6 are the actual headers added to packets for the transmission of data over internet. 6LoWPAN is a compressed header that will be converted to a IPv6 header after it passes a gateway.

IPv4 and IPv6

IPv4 and IPv6 form the primary formats to communicate over the internet. IPv4, the older of the two, is cur-

rently facing problems as almost no unused addresses are left. IPv6 was brought to life to solve this problem among several other problems. The minimal IPv4 header size is 20 bytes and the minimal IPv6 header size is 40 bytes. For our data analysis, IPv6 was assumed to be the solely used header as this might be the case in the future. The length of the added internet layer protocol is therefore 40 bytes.

6LoWPAN

IPv6 headers are a minimum of 40 bytes. This is the most basic format. Currently a format is in development to compress this header. This format is build to be used together with the IEEE 802.15.4 [1] standard and is called the 6LoWPAN standard. 6LoWPAN is created to decrease the header size of both IPv6 and UDP to a minimum of a mere 6 bytes[2] instead of the 48 bytes. This standard has been introduced to cope with the limiting size of sending of packets of IEEE 802.15.4, which is a mere 127 bytes. This format can not be used on the cellular network, however support is added to LoRa networks. A node will send the data with a 6LowPAN header to a gateway, which in turn will transform this header to a IPv6 header. For future versions of this product, versions that support LoRa, 6LoWPAN can be used to significantly reduce energy usage. As with using 6LoWPAN less bytes will have to be sent per transmission and thus less energy would be required per sending and receiving cycle.

10.2.3. Transport layer protocol

The transport layer is the lowest end-to-end node layer. This is layer 4 of the OSI model. Several protocols exist that differ in size and functionality. The two primary protocols are TCP and UDP. TCP [24] in comparison to UDP is more reliable. The TCP protocol splits data into several TCP packets. These TCP packets are numbered and include a check sum in the header. The TCP protocol includes several steps to establish a new connection. If a checksum is wrong or no acknowledgment is received over TCP, the packet is resent. The size of a TCP header is 20-60 bytes. UDP [25] in contrast is classified as unreliable as no acknowledgment is sent and therefore no re-transmission occurs, it does however contain checksums. TCP is generally used in the world wide web. UDP stands out as it has low latency. It doesn't require re-transmission of packets, therefore it is beneficial in VoIP and gaming solutions. UDP is only 8 bytes in size. The SIM5320e module has built-in support for both TCP and UDP.

10.2.4. Application layer protocols

An application layer (layer 7 of the OSI model) protocol is used to enable communication between applications. The most extensively used protocol for surfing the web is HTTP [39]. This protocol is very resource intensive, significantly bloating the size of the header and requiring more complex parsing methods. Several application protocols exist beside HTTP that could serve the same function for M2M/IoT applications that needs less resources. A few of the promising ones include MQTT (MQ Telemetry Transport); MQTT-SN (MQ Telemetry Transport-sensor) and CoAP (Constrained Application Protocol). Out of these three protocols, MQTT-SN cannot be used in combination with a cellular version. MQTT-SN needs a gateway between the sensor device and the SkyDowser servers. An MQTT-SN connection is established between the sensor node and the gateway. The gateway will then establish an MQTT connection with the SkyDowser servers. MQTT-SN is built to be more efficient than MQTT. MQTT-SN is only ruled out for the cellular version, but for later implementations that will support gateways, this might still be a possible solution. The two remaining protocols considered are CoAP and MQTT.

CoAP

CoAP [35], an abbreviation of constrained application protocol, is a more recently introduced protocol than MQTT. This protocol is linked to the unreliable transport layer protocol UDP. This time the reliability feature will be added on the application layer instead of on the transport layer. This protocol is based upon a request/response model. To send data to the SkyDowser servers, CoAP allows to send data in one packet with confirmation or without confirmation, depending on if the SkyDowser servers are requested to do so. This is not required, but adds reliability. Then the sensor will receive a second message with the confirmation. CoAP has a fixed standard header size of 4 bytes. To this header more options may be added. This sums up to one packet to be sent and one packet to be received. CoAP however has no checksums included.

MQTT

MQTT [8] is an application layer protocol used in combination with TCP. The header of MQTT is 2 bytes in size. MQTT has been around since the 1999, and is therefore a more mature standard. As the data packets

to be sent to the SkyDowser servers are a relative small amount, the connection setup of a TCP connection are of great influence to the overall amount of data transmission. To send a datapacket with sensor data to the SkyDowser server with a confirmation, first the connection should be established. This is a three step handshake. The second step is to send the data packet with information followed by an acknowledgment of the server. After this the TCP connection is shutdown by a 4 step handshake. Two packets are sent and two packets are received. This totals up to 5 packets to be transmitted and 4 packets to be received. Some remarks have to be made with this explanation. If more information has to be sent or received the same connection can be used. The second remark is that the TCP header is larger in size compared to UDP. The TCP header 20 bytes in size and the UDP header is 8 bytes in size.

Final choice - CoAP/UDP

By carefully considering both CoAP and MQTT, CoAP is chosen to be the selected protocol. With the same amount of reliability, CoAP does the same job as MQTT while reducing the amount of transmission data at least four times. If assumed that each datapacket is about the same size, MQTT needs to transmit five times as many packets compared to CoAP. CoAP is also a more future proof standard as CoAP is linked to UDP and can therefore be used in combination with the IEEE 802.15.4 standard. The IEEE 802.15.4 standard, created for low power wireless personal networks, allows a maximum transmission size of 127 bytes, rendering TCP less favorable and with that MQTT as well.

10.2.5. Data encryption layer - security

In order to enforce secure transmission of data, the messages are encrypted. A generally used and considered safe format is the TLS [32] standard. However TLS is made for TCP connections and does not work together with UDP. This led to the introduction of DTLS[16]. This encryption standard is made for UDP, which is based upon the TLS standard. It's also named TLS over UDP. CoAP has several methods to encrypt its messages using DTLS. The key could be pre-shared, meaning each SubDowser is given the key at installation. The SubDowser also might use asymmetric keys, meaning that with different requests the device is given the public key. These public keys are then reused for some time. Both methods include a variety of advantages. The pre-installed security key decreases the amount of data to be transmitted and therefore increases energy usage. However if a key is compromised, third parties might continue using the same key. Another disadvantage might be improvements in cracking in the future. On the other hand, requesting public keys keeps security up to date, but uses more energy. As for the amount of bits for the security key two considered two size keys are generally used. These are 128 bits and 256 bits. According to ClickSSL [10] to crack a 128 bit encryption, a computer at the moment must operate 10^{18} years. For 256 bits this is 10^{56} years. The time to crack a 128 bit encrypted message is secure in comparison to the sensitivity of the data. As computer performance won't increase that significant over the next ten years and this product is meant to operate for less than 10 years, the 128 bit encryption will suffice.

10.2.6. Node identification - security

To identify whether data received by the server is coming from a registered node, a randomly generated identification code assigned to the SubDowser node of 5 bytes is used. Node identification is implemented to ensure that no third party is capable of transmitting false information. Possible motivations for third parties to do so include, but is not limited to, insurance fraud or sabotage. A connecting node is only able to log in using a certain identification code a maximum of five times, before the node is given a timeout. Five bytes allow 10^{12} possibilities. The node identification code is a kind of password.

10.2.7. Actual Data Format

The data to send to the SkyDowser servers and the data to be received from the SkyDowsers should be formatted as to maximize information delivery. In this section, several formats are discussed with different configurations. If the sensing is done in a planned fashion, the time at which a sensing occurs can be derived without explicitly stating it, reducing the amount of data to send. Figures 10.3 and 10.4 show 4 possible formats to send sensor data. Figure 10.5 shows 2 formats the server could use to send data back.

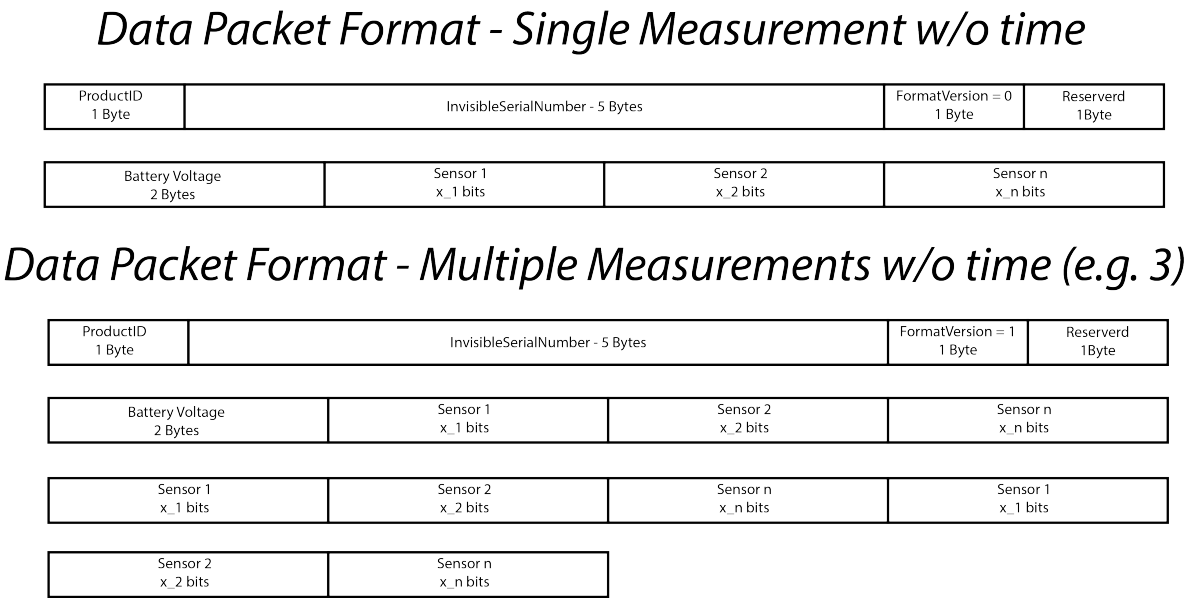

Figure 10.3: A datapacket with sensor data. Time is derived by capturing the time of transmission.

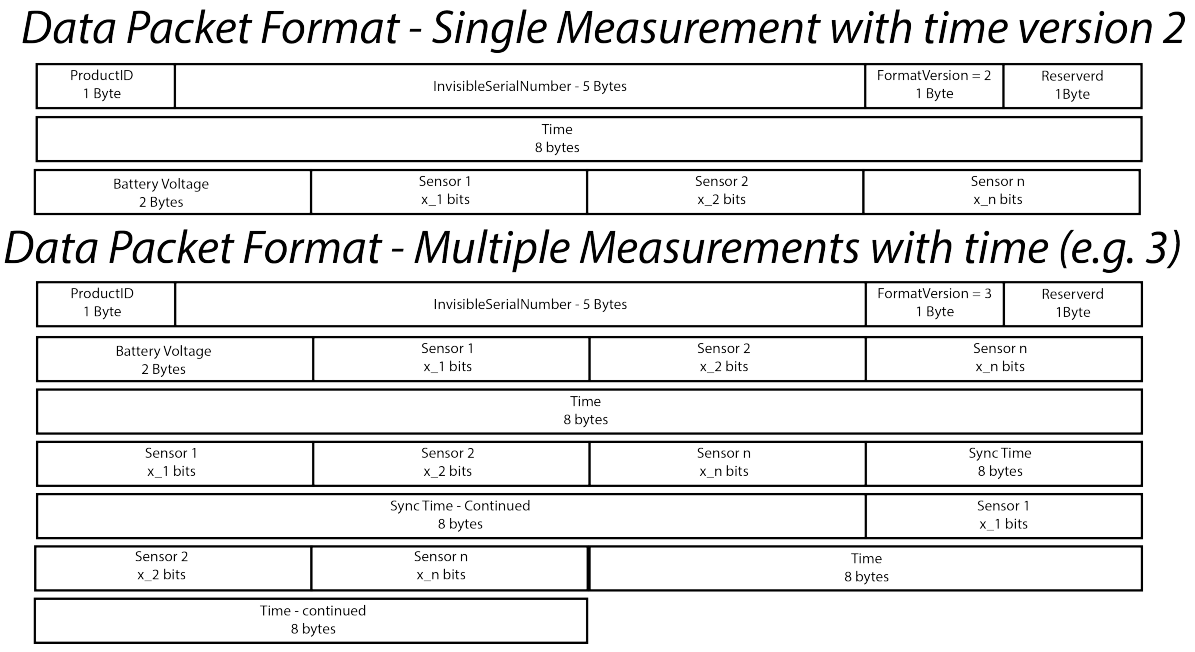

Figure 10.4: A datapacket with sensor data. Time is added for each sensing.

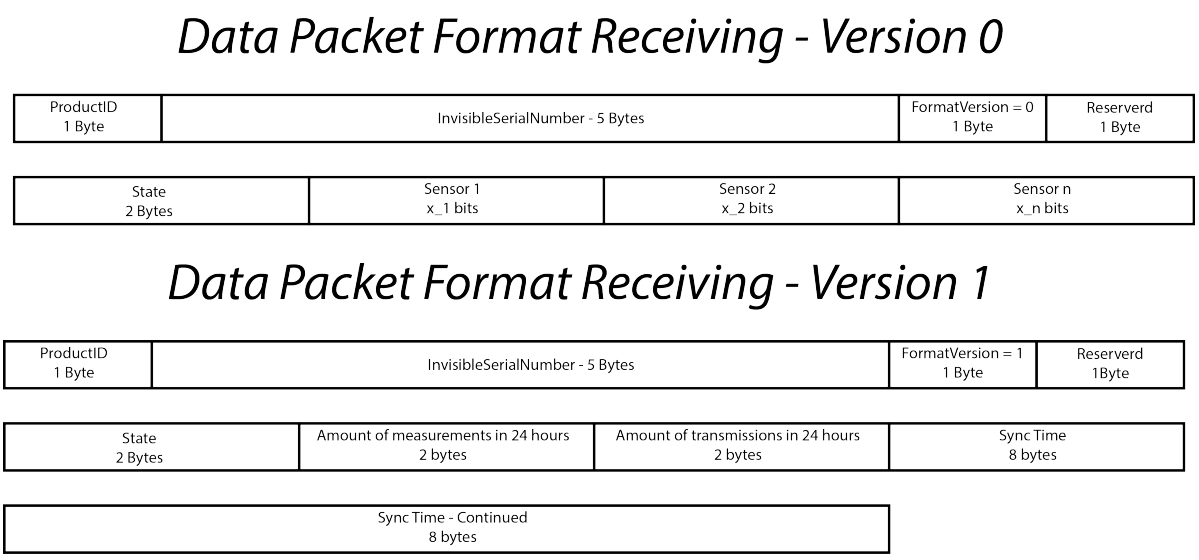

Figure 10.5: A datapacket the SubDowser sensor node receives after sending a packet.

11

LoRa/LoRaWAN

LoRa[28] is a recently introduced modulation technology. Semtech, the proprietary owner of the modulation technology claims to transceive [27] data 19dB below the noise threshold. Semtech claims this is at least 20dB below any other modulation technology. The 20dB difference extends the range in comparison to other techniques. Semtech claims their reach for communication is greater than 15km for suburban areas. Little information has been found regarding the energy consumption per bit, but claims have been made about LoRa IoT solutions functioning 10 years plus on batteries. The definition of LoRa and LoRaWAN is different. LoRa is the name given to the modulation technology and its proprietary owner is Semtech. LoRa is the PHY layer. LoRaWAN[13] is owned by the LoRa alliance (handed over by Semtech) and is the MAC layer. LoRaWAN is built to be used alongside LoRa.

11.1. LoRa affiliation to SubDowser

An issue LoRa is currently facing is its limited coverage. Trials have been setup in Amsterdam and other progressive cities. No signs however are available that show Kenya receiving LoRa deployment any time soon. The objective of SubDowser however is to deliver modules world wide. A SubDowser version with a LoRa communication system might therefore be viable in other parts of the world. Take for example the Netherlands. KPN is currently taking lead deploying LoRa coverage nationwide. For our business research, third party interest was shown for deployment of Subdowsers in the Netherlands. LoRa has a few advantages over the currently proposed cellular version. A LoRa node is not required to be subscribed to a mobile operator, therefore the configuration costs and yearly costs will be reduced. Semtech claims significantly less energy usage for LoRa communication in comparison to cellular. According to organization The Things Network[12], the LoRa network might not need to be deployed by mobile operators. The range of communication between a node and a gateway is relatively high. Less gateways therefore are required to achieve nationwide coverage, making nationwide coverage a relatively cheap product. Since LoRa gateways can be deployed without any bandwidth licensing issues, as they operate in open frequency bands, governments or individuals might be providing free coverage in the future, reducing costs even further.

11.2. Theoretical Implementation LoRa

Currently Microchip is the only provider of LoRa node chips, rumors have been spreading about other companies providing LoRa chips in the future though. These companies pay Semtech royalties for the LoRa modulation technique. Currently Microchip provides two versions of their LoRa node chips. Different frequency bands are used for LoRa modulation in the US compared to Europe. The RN2483[21] version is made for the European market and the RN2903 is made for the American market.

12

Conclusion

This project has been done to find ways of forming a cost-efficient telecommunications method between a measurement node and a server in Kenya. Throughout this thesis multiple choices have been made regarding the implementation of the telecommunications system. The research has made clear that there are several different ways in which the system could also have been constructed, each with different advantages and disadvantages. To the end of fulfilling the requirements set at the start of the project, the actually implemented system is expected (as not the full code has been tested yet) to be satisfactory but open to improvements.

The main part of the system is the SIM5320 chip, to which all other components of the telecommunications and power supply system are connected. Data from the microcontroller is sent to this chip through UART. The power supply for the other subsystems is redirected from this chip and a SIM card is used to enable network functionality of the chip. Aside from these components, a GSM and a GPS antenna are connected to the SIM5320. The GSM antenna connects the module to the cellular network, through which the data will be transmitted to the server. Improvements can be implemented, such as using another, more energy efficient network technology for transmissions, supplementing the coding with more error detection functions and using a different power source. A theoretical overview of the LoRa technology has been described to the end of improving the system in the future, together with different protocols to be used for more efficient transmissions. However, in the current implementation the existing cellular network is used for easy of implementation.

The goal of this project was to construct a way to enable the transmission of measurement data to and from the SkyDowser servers, together with supplying the total system with power. Data transmission has been successfully implemented through the use of the HTTPS protocol, whilst the power supply system has been designed to use a rechargeable Li-ion battery rated at 3.6V in combination with a 6V-solar cell as the main power supply. This solar cell is added to extend the lifetime of the battery, which on its own would be 3 months. The cycle lifetime of the battery is 500-1000 cycles, which combined with the solar cell ensures that the system would last at least 2 years autonomously. As multiple voltage levels are required for the different components of the total system of SubDowser and only one battery is used, buck- and boost converters are integrated into the power circuit. To reduce the amount of wires needed a power-over-bus system is used, to which end filters are built at each end of the data-transmission cable.

Not every aspect of the design of such a system has been fully explored. As not many research results have been obtained on the matter of energy consumption by different components of the system, no real conclusions have been formed on the matter. As the measurement frequency can be changed by the server, the active time of the SubDowser system is not accurately defined. This is not the only improvable aspect, as is made clear in the discussion chapter. However, the current implementation of the telecommunications system is sufficient in fulfilling the proposed goals with the available knowledge at the time.

13 Discussion

During the project some problems occurred that hindered progression. This chapter will be devoted to elaborating on these problems and their solutions, as well as noting what could be improved or should be done differently in a next project.

13.1. FONA 3G Module

The telecommunications module was the main delaying factor during this project. Some of the problems that arose using this module were fixed eventually, but not all of those fixes can be explained by the performed research, as they seemed to be solved "suddenly". An example of this kind of problem is that, after testing HTTPS capabilities of the module, the HTTPS capabilities did not work anymore. Extensive testing and researching solutions did not seem to be helpful in any way. A decision was made to try and update the module's firmware manually, which birthed more problems than it solved. The update went wrong, resulting in the firmware being deleted of the module and therefore the module not responding to any commands anymore. Several days were lost trying to solve these problems, but eventually switching laptops to try and install the firmware resulted in the problem being tackled. Even the HTTPS capabilities were returned, of which can only be hypothesized that the older firmware was indeed blocking access to those commands. However, this does not explain why the HTTPS commands were responsive in the beginning and thus it is not sure how this problem occurred.

Downloading data from internet also took up a lot of time. Using a GET request was tested around the start of the project, but this was not done correctly until several weeks later. This was not just because of the problems stated above, but because of not using the correct syntax. Even though research was done online on the syntax it seems small mistakes kept being made, which resulted in the GET or POST request completely failing to get the server to respond. The information on the internet also seems to contradict itself a lot, which is destructive in the case of such a sensitive syntax.

13.2. Timetable

At the start of the project all subgroups together agreed upon a timetable. The first few weeks of this schedule were dedicated to conducting research in advance on both concepts and existing alternatives. A literature study was done during the first week as well. The performed activities in those first few weeks could, in hindsight, have been done in at least three weeks instead of four. However, the start of the project was also the first time really working out a whole project without a manual (though, with assistance of supervisors), which could have been the cause of the project starting out a bit slower than if the group members had more experience in the field.

13.3. Possible improvements

Naturally, the implemented system is not perfect. There are some noteworthy changes that can be applied to this system to improve its operating efficiency, besides ones already mentioned in earlier chapters of this thesis, such as in chapter 10. For example: the measuring frequency is set by the server. These settings get updated with the same frequency as the measurements are sent to the server. Therefore, if the measuring period is set to 48 hours, the settings would not be updated until 48 hours later. An improvement would be setting a maximum period for receiving data from the server, so that it is not dependent on the period of sending data to the server.

The coding can be improved by inserting more error correction cases, as well as adding codes that actually check some of the responses to determine the exact state the module is in (instead of just checking for response "OK", as it does now in a lot of cases). However, adding more of these codes might result in a longer transmission time and overall performing time going up, which is less energy efficient and as such might not be preferable. Besides this, functions that wait for a response can be edited to require an additional input variable: a timeout value. The function should then use this variable to check if a response is taking too long and take appropriate measures. This could improve the transmission time drastically in cases where a connection is unstable or the module gets stuck.

As stated in chapter 8, a solar cell can be added to the power supply circuit. Especially in areas that are continuously exposed to sunlight this is a very durable solution to charging the battery and thus improving the lifetime of the product.

Instead of using GSM modules, a different technique like LoRa can be used, as named in chapter 11. This not only removes the need of using a SIM card and therefore reducing yearly costs, but it also improves the energy efficiency of the overall system and building such a network provides the occupied area access to new techniques that LoRa enables.

As not much research has been done on energy consumption regarding usage of the FONA 3G module, it might be better not to send the FONA module to sleep at all but to keep it turned on permanently. At the start of the project the transmission frequency was set to be once per day, in which case turning the module off and on would be more energy efficient. However, as the requested transmission frequency has changed over the course of the project, this might not be the best option anymore. As the transmission frequency is now dictated by the server and can be set to different values, such as once per half hour or once per day, it would be the most efficient to make whether the module gets send to sleep dependent on the transmission frequency.

A

C++ Code

```
#include "mbed.h"
#include <string>
#include <iostream>
#define MODSERIAL_DEFAULT_RX_BUFFER_SIZE 12288
#define MODSERIAL_DEFAULT_TX_BUFFER_SIZE 12288 //was 4096
#include "MODSERIAL.h"
using namespace std;

/************************\
*         COMMENTS       *
\************************/

/*
-Currently, the code is only checking for "OK" or other positive responses. Do not forget to add error detection or other negative responses
-Add Timeout
-There are some lines marked as \\DEBUG, remove these
-When a hard reset is called, an interrupt should start this code again from the beginning
-Settings_request_length changes if IP has less or more numbers than it had previously
*/

/************************\
*        INITIALIZE      *
\************************/

//TEMPORARY, WILL BE PROVIDED BY CONTROL GROUP
string data_to_send = "id=TestDmsgeDabc"; //This string is received from the control system and contains the sensor values
string sensor_data_length = "16"; //Length of data_to_send string

//Global variables, configure before running
string PIN_code = "6350";
string APN = "portalmmm.nl";
string host_address = "145.94.177.225"; //IP address of the server
string host_page = "subdowser/subdowser.php"; //Page to access on the server. If only the main page of host_address is needed, leave empty
string port = "80"; //443 if HTTPS, or 80 if it's a HTTP server
string server_type = "1"; //2 if HTTPS, or 1 if it's a HTTP server
string acknowledgment = "TestDmsgeDabc"; //Or another message the server sends back
string data_to_send_length = "151"; //Length of POST request string
string ack_length = "36"; //Length of acknowledgment to be received (only the body)
string settings_request_length = "63"; //Length of settings request
string settings_length = "335";
string settings_received=""; //The string in which the settings are stored after receiving

//Define pins
MODSERIAL gsm(p28,p27); //TX, RX pins
PwmOut gsm_reset(p26); //Reset pin

//Initialize variables
```

```
int wait_response=1;
int consecutive_errors = 0;
int i = 0; //Used for loops
int count = 0; //Used for waits
int baudrate = 115200;
string settings = ""; //The settings for the measurement frequency etc.
int max_consecutive_errors = 5;

//Define functions
void configure_network();
void configure_GPRS();
void send_cmd(string);
int wait_for_response(string, int);
string wait_for_response_string(string, int);
void send_data(string);
string receive_data();

/************************\
*       FUNCTIONS        *
\************************/

/*Main function, the starting point of the program*/
int main(){
   while(1){
        //Define baudrate
        gsm.baud(baudrate);
        //Initialize PWM to not-reset (stay HIGH)
        gsm_reset.period(1.0f);  // 1 second period
        gsm_reset.pulsewidth(1.0); // Always HIGH

        while(wait_response!=0){
            send_cmd("ATE0"); //Disable Echo
            wait_response = wait_for_response("OK",7);
        } wait_response = 1;

        configure_network();
        configure_GPRS();

        send_data(data_to_send);
        settings = receive_data();

        while(1){cout << "Settings: " << settings << "\n";} //DEBUG settings display
        break;
   }
   return 0;
}

/*Configure network, make module ready for SMS*/
void configure_network(){
   while(1){
        while(wait_response!=0){
            send_cmd("AT+CPIN="+PIN_code); //Enter PIN
            wait_response = wait_for_response("PB DONE",12);
        } wait_response = 1;
        while(wait_response!=0){
            //send_cmd("AT+COPS=0"); //Automatically assign a network
            send_cmd("AT+COPS=4,2,20408"); //Force NL KPN, replace by line above for automatic assigning
            wait_response = wait_for_response("OK",15);
        } wait_response = 1;
        while(wait_response!=0){
            send_cmd("AT+CSMS=1"); //SMS compatible with GSM phase 2+
            wait_response = wait_for_response("OK",7);
        } wait_response = 1;
        while(wait_response!=0){
            send_cmd("AT+CPMS=\"SM\",\"SM\",\"SM\""); //Choose SIM as storage
            wait_response = wait_for_response("OK",6);
        } wait_response = 1;
        while(wait_response!=0){
            send_cmd("AT+CMGF=1"); //Set message format to text
            wait_response = wait_for_response("OK",7);
        } wait_response = 1;
```

```
119
            break;
121     }
    }
123
    /*Configure PDP context, make ready for using GPRS*/
125 void configure_GPRS(){
        while(1){
127         while(wait_response!=0){
                send_cmd("AT+CGDCONT= 1,\"IP\",\"" + APN + "\",\"0.0.0.0\",0,0"); //Edit first PDP context
          entry to contain the correct APN
129             wait_response = wait_for_response("OK",7);
            } wait_response = 1;
131         while(wait_response!=0){
                send_cmd("AT+CGSOCKCONT= 1,\"IP\",\"" + APN + "\""); //Configure socket
133             wait_response = wait_for_response("OK",7);
            } wait_response = 1;
135         while(wait_response!=0){
                send_cmd("AT+CGATT=1"); //Attach module to Packet Domain service
137             wait_response = wait_for_response("OK",7);
            } wait_response = 1;
139         while(wait_response!=0){
                send_cmd("AT+CGACT=1,1"); //Connect to the first PDP context entry
141             wait_response = wait_for_response("OK",2);
            } wait_response = 1;
143         break;
        }
145 }

147 void send_data(string sensor_data){
            //Connect to server
149         while(wait_response!=0){
                send_cmd("AT+CHTTPSSTART"); //Acquire HTTPS protocol stack
151             wait_response = wait_for_response("OK",100);
            } wait_response = 1;
153         while(wait_response!=0){
                send_cmd("AT+CHTTPSOPSE=\"" + host_address + "\"," + port + "," + server_type); //Open HTTPS
          session
155             wait_response = wait_for_response("OK",100);
            } wait_response = 1;
157
            //Send data and wait for acknowledgment
159         while(wait_response!=0){
                send_cmd("AT+CHTTPSSEND=" + data_to_send_length);
161             wait_response = wait_for_response(">",5);
            } wait_response = 1;
163         while(wait_response!=0){
                send_cmd("POST /" + host_page + " HTTP/1.1\r\nHost: " + host_address + "\r\nContent-Length: "
          + sensor_data_length + "\r\nContent-Type: application/x-www-form-urlencoded\r\n\r\n" + sensor_data +
          "/r/n"); //Send data to server
165             wait_response = wait_for_response("EVENT",7);
            } wait_response = 1;
167         while(wait_response!=0){
                send_cmd("AT+CHTTPSRECV=195");//Receive header of the message, so that only the body is left
          in the buffer
169             wait_response = wait_for_response("RECV: 0",14);
            } wait_response = 1;
171         while(wait_response!=0){
                send_cmd("AT+CHTTPSRECV=" + ack_length);
173             wait_response = wait_for_response(acknowledgment,13);
            } wait_response = 1;
175
            //End HTTPS session
177         while(wait_response!=0){
                send_cmd("AT+CHTTPSSTOP");
179             wait_response = wait_for_response("OK",7);
            } wait_response = 1;
181 }

183 string receive_data(){
            //string settings_received="";
```

```
185
        //Connect to server
187     while(wait_response!=0){
            send_cmd("AT+CHTTPSSTART"); //Acquire HTTPS protocol stack
189         wait_response = wait_for_response("OK",100);
        } wait_response = 1;
191     while(wait_response!=0){
            send_cmd("AT+CHTTPSOPSE=\"" + host_address + "\"," + port + "," + server_type); //Open HTTPS
    session
193         wait_response = wait_for_response("OK",100);
        } wait_response = 1;
195
        //Receive settings
197     while(wait_response!=0){
            send_cmd("AT+CHTTPSSEND=" + settings_request_length);
199         wait_response = wait_for_response(">",5);
        } wait_response = 1;
201     while(wait_response!=0){
            send_cmd("GET /" + host_page + " HTTP/1.1\r\nHost: " + host_address + "\r\n"); //Receive
    settings from page
203         wait_response = wait_for_response("EVENT",7);
        } wait_response = 1;
205     while(wait_response!=0){
            send_cmd("AT+CHTTPSRECV=" + settings_length);
207         settings_received = wait_for_response_string("Message",16);//The settings string should have
    one word in it that is constant, for detection purposes (else the program does not know which line of
     the package to return as settings)
            if (settings_received != "-1"){
209             wait_response = 0;
            }
211     } wait_response = 1;
        //End HTTPS session
213     while(wait_response!=0){
            send_cmd("AT+CHTTPSSTOP");
215         wait_response = wait_for_response("OK",7);
        } wait_response = 1;
217
    return settings_received;
219 }

221 /*Function that sends AT commands to the module*/
    void send_cmd(string AT_cmd){
223     cout << "Command:" << AT_cmd << "\n"; //DEBUG
        string AT_cmd_endline = AT_cmd + "\r\n";
225     gsm.puts(AT_cmd_endline.c_str());//Add "%s" and .c_str() to use a string as input variable
    }
227
    /*Waiting for responses*/
229 int wait_for_response(string response_expected, int max_string_length){
        char response_received_arr[100]={};
231     while(1){
            for(i=0; i < max_string_length; i++){ //Read each character and store it in an array, to form a
    string
233             if(gsm.readable()){
                response_received_arr[i] = gsm.getc();
235             response_received_arr[i+1] = '\0'; //Mark the end of the string
                    if(response_received_arr[i] == '\n'){
237                     response_received_arr[i] = '\0';
                        break;
239                 }
                }
241         }
            string response_received(response_received_arr); //Convert character array to string
243
            cout << "Response: " << response_received << "\n"; //DEBUG
245
            if(response_received.find(response_expected) != std::string::npos){ //Stop waiting if part of the
    response matches the expected response ("abcdOKef" matches response_expected "OK")
247             consecutive_errors = 0;
                gsm.rxBufferFlush(); //Empty RX buffer
249             gsm.txBufferFlush(); //Empty TX buffer
```

```
            break;
        }
        if(response_received.find("ERROR") != std::string::npos){
            consecutive_errors += 1;
            /*if(consecutive_errors >= max_consecutive_errors){//Call for a hard reset, something seems to be going wrong
                gsm_reset.pulsewidth(0.9); // 0.1 second pulse LOW, required for hard reset
            };*/
            return -1; //Add "if(wait_response==-1){};" to code that called this function to determine how to solve the ERROR
        }
        //cout << "Wrong response, expected was: " << response_expected << "\n"; //DEBUG
    }
    return 0;
}

/*****************************************************\
*       FUNCTIONS SHARED WITH CONTROL SYSTEM GROUP      *
\*****************************************************/

string getlocation(){
    string GPS_data=""; //String to be returned to control system group

    while(wait_response!=0){
        send_cmd("AT+CGPS=1"); //Turn on GPS
        wait_response = wait_for_response("OK",100);
    } wait_response = 1;
    while(wait_response!=0){
        send_cmd("AT+CGPSINFO"); //Request GPS data
        GPS_data = wait_for_response_string("OK",95);
        if(GPS_data != "-1"){
            wait_response=0;
        }
    } wait_response = 1;

    return GPS_data;
}

/*Waiting for responses AND return received string*/
string wait_for_response_string(string response_expected2, int max_string_length2){
    char response_received_arr2[100]={};

    while(1){
        for(i=0; i < max_string_length2; i++){ //Read each character and store it in an array, to form a string
            if(gsm.readable()){
            response_received_arr2[i] = gsm.getc();
            response_received_arr2[i+1] = '\0'; //Mark the end of the string
                if(response_received_arr2[i] == '\n'){
                    response_received_arr2[i] = '\0';
                    break;
                }
            }
        }
        string response_received2(response_received_arr2); //Convert character array to string
        //string response_settings(response_received_arr2);

        cout << "Response: " << response_received2 << "\n"; //DEBUG

        if(response_received2.find(response_expected2) != std::string::npos){ //Stop waiting if part of the response matches the expected response ("abcdOKef" matches response_expected "OK")
            consecutive_errors = 0;
            gsm.rxBufferFlush(); //Empty RX buffer
            gsm.txBufferFlush(); //Empty TX buffer
            return response_received2; //Return the received string
        }
        if(response_received2.find("ERROR") != std::string::npos){
            consecutive_errors += 1;
            /*if(consecutive_errors >= max_consecutive_errors){//Call for a hard reset, something seems to be going wrong
                gsm_reset.pulsewidth(0.9); // 0.1 second pulse LOW, required for hard reset
```

```
            };*/
317         return "-1"; //Add "if(wait_response==-1){};" to code that called this function to determine
        how to solve the ERROR
        }
319     //cout << "Wrong response, expected was: " << response_expected2 << "\n"; //DEBUG
    }
321 }
```

B

FONA 3G Pins

This appendix describes the functions and connection points of each pin of the Adafruit FONA 3G Cellular + GPS Breakout. The connector field is marked as either optional or unneeded for pins that are not used in this project. The connector field value is based on the LPC1768 microcontroller [23].

Table B.1: FONA 3G pin connections

Pin Name	**Connector**	**Function**
5V	Optional	Checks if microUSB is plugged in
Bat	+	Battery input
Gnd	-	Ground
Vio	+	Determines logic level
Rst	PIN26	Reset, LOW for 100ms to perform hard reset
RX	PIN28	Input signal UART
TX	PIN27	Output signal UART
RTS in	Optional	Hardware flow control. If used, the pin can start and stop data transfer from FONA to microcontroller
DTR	Optional	Hardware flow control. Control data/command mode TCP/IP
CTS out	Optional	Hardware flow control. If used, the pin can determine when the buffer is full to stop/start transfer to FONA from microcontroller
Key	Optional	Power on/off indicator. Tie to ground permanently to disable turning FONA off, for power saving. Send LOW signal for 3-5 seconds to turn the module off or on
RI	Optional	'Interrupt' out, used for calls and SMS receiving
PS	Optional	Power status indicator, tied to power led
NS	Optional	Network Status, also connected to net led. Indicates current status of module
Mic+	Unneeded	Connect external microphone
Mic-	Unneeded	Connect external microphone
SPKR+	Unneeded	Connect to external 8 ohm speaker
SPKR-	Unneeded	Connect to external 8 ohm speaker

C

Program of Requirements

C.1. Functional requirements

1. The product needs to measure the groundwater level, conductivity and temperature of the water
2. The product needs to send the measurement data to the server of SkyDowser

C.2. Non-functional requirements

C.2.1. Product requirements

Usability

3. The product needs to be able to adjust settings, such as measuring and sending frequency, according to changes given on the SkyDowser servers
4. The product needs to be Plug & Play for the customer
5. The product needs to be modular
6. The cost price, without assembly cost of the product, needs to be lower than 50 Euros at an order of 10000 pieces
7. The annual operating cost may not exceed 10 Euros
8. The product shall first be used in Kenya and Tanzania

Performance

9. The measurement frequency must be adjustable up to once every 10 minutes. From that point it should be possible to adjust this setting in steps of 10 minutes, with a maximum measure frequency of 48 hours
10. The send frequency must be adjustable up to once every 10 minutes. From that point it should be possible to adjust this setting in steps of 10 minutes, with a maximum send frequency of 48 hours
11. The measured data must be labeled with time at an accuracy of **1 second** and must use the UNIX timestamp format
12. The location coordinates of SubDowser must be sent to the SkyDowser servers at least once, with an accuracy of **100 m**
13. All the measured data must be sent correctly to the SkyDowser servers
14. The product must also be capable of operating in hilly area (0-1000m)
15. The product must work autonomously for at least **2** years before any maintenance is required. Manufacturing failures are excluded

16. The conductivity should be able to be measured within a reach of **0 to 15000 uS/cm** at a resolution of **1uS/cm** and a maximum deviation of ±1.5**%**
17. The temperature of the water in the well should be able to be measured within a reach of **5 to 20 °C** at a resolution of **0.5 °C** and a maximum deviation of ±1**°C**
18. The water level should be able to be measured within a reach of **0 to 60 m** with a maximum water column of **20 m** and a accuracy of ±5 **cm** and a resolution of **1 cm**

Dimensions

19. The SubDowser must fit in a **6 cm** diagonal wide tube
20. The SubDowsers have an average density of **2-3** SubDowser per square kilometre
21. The average distance between two SubDowsers in a work area is a maximum of 1 kilometre

Safety

22. The subsystem needs to meet the European General Product Safety Directive 2001/95/EC
23. The measurement data may not be intercepted while being transferred to the SkyDowser servers
24. If the product is above ground it needs to be resistant to temperatures up to **70** °C
25. If the product is below ground it needs to be resistant to temperatures up to **30** °C when operational. During transport it needs to be resistant to temperatures up to **70** °C
26. If a part of the SubDowser is underground and underwater it should meet the IP-68 encryption
27. If a part of the SubDowser is above ground it should meet the IP-64 encryption
28. The part above ground must be resistant to normal mechanical impact, minimum of IK-07.
29. The part underground must be resistant to mediate mechanical impact, minimum of IK-05.

Operational

30. The product must be easy to use by the customer
31. The product needs to be able to be activated without the assistance of SkyDowser, so by the customer himself

Environmental

32. The product must not contaminate the groundwater as specified in the law for groundwater of the European Union

D

Cost Overview

For the telecommunications and power supply system of the SubDowser project a preferred budget of about 25 euros was available. This appendix documents the total cost of the implemented system and possible alternatives, delivery and assemblage costs excluded. The minimum order quantity will be around 10000, as stated by the client. Some prices are rounded to whole units and if multiple prices are available an average is taken, as it is taken into account that not every source might be as reliable as the other. The column "Type of component" notes whether the component is used in the prototype or is an available alternative for cost reduction. As the prototype (at the moment of writing this thesis) makes use of a breakout board instead of just the chip itself, costs are much higher and as such only the chips without breakout/development board are included in the list, marked as alternatives. Other components that are only necessary for the prototype, such as breadboards and jumper wires, are also excluded from the list. The overview is shown below in table D.1.

Aside from these components some resistors, capacitors, inductors and wires will be needed. The total expected cost of these materials is around 1 euro. As can be seen from the list, using solely the materials that are also used in the prototype will result in the budget of 25 euros being exceeded. Therefore, using alternatives and with that placing less weight on certain performance qualities is recommended if more importance is assigned to costs. Note that not all alternatives are listed, there are cheaper but qualitatively less recommended alternatives that could also be selected. However, using the alternatives that are listed above will sum up to slightly less than 25, which means the budget requirement would be fulfilled.

Table D.1: Cost per component

Component name	Price [Euro]	Quantity for this price	Type of component	Note
SIM5320	35	5	Prototype	Supports both 3G and 2G
SIM808	8	10	Alternative	Does not support 3G, only 2G
GPS Antenna - External Active Antenna - 3-5V 28dB 5 metres SMA	10	100	Prototype	Active antenna for speed
Passive GPS Antenna uFL - 15mm x 15mm 1 dBi gain	3	100	Alternative	Passive antenna, is slower
Slim Sticker-type GSM/Cellular Quad-Band Antenna - 3dBi uFL	3	1	Prototype	-
SMA to uFL/u.FL/IPX/IPEX RF Adapter Cable	3	100	Prototype	-
Lithium Ion Polymer Battery - 3.7v 1200mAh	8	100	Prototype	-
Lithium Ion Polymer Battery - 3.7v 380mAh	2,60	5	Alternative	Alternative battery
LM3670MF	0,31	5000	Prototype	Buck converter
SP6641AEK-L-5-0	0,45	6000	Prototype	Boost converter
Solar Cell	2	1000	Prototype	Used for charging the battery
Simyo SIM card	15	1	Prototype	Prepaid card with a large data packet
SIM card Kenya	2	1	Alternative	SIM card in Kenya

Bibliography

[1] Internet of things: 802.15.4, 6lowpan, rpl, coap. https://www.utwente.nl/ewi/dacs/colloquium/archive/2010/slides/2010-utwente-6lowpan-rpl-coap.pdf,.

[2] 6lowpan: An open iot networking protocol. http://events.linuxfoundation.org/sites/events/files/slides/6lowpan-openiot-2016.pdf,.

[3] At command tester v23. www.m2msupport.net/m2msupport/module-tester/.

[4] What's the best battery? http://batteryuniversity.com/learn/archive/whats_the_best_battery.

[5] Ingress protection rating. http://www.lumascape.com/html/IP_IK_rating.htm.

[6] Kicad eda. http://kicad-pcb.org/.

[7] Bu-410: Charging at high and low temperatures. http://batteryuniversity.com/learn/article/charging_at_high_and_low_temperatures.

[8] Mqtt. http://mqtt.org/.

[9] Microchip's battery management solutions. http://www.microchip.com/design-centers/analog/power-management/battery-management.

[10] 128-bit ssl encryption vs 256-bit ssl encryption. https://www.clickssl.net/blog/128-bit-ssl-encryption-vs-256-bit-ssl-encryption.

[11] Tera term. http://download.cnet.com/Tera-Term/3000-20432_4-75766675.html.

[12] The things network - homepage. https://thethingsnetwork.org/.

[13] N. Sornin (Semtech) M. Luis (Semtech) T. Eirich (IBM) T. Kramp (IBM) Hersent (Actility). LoRaWAN Specification. RFC 1, LoRaWAN Alliance, January 2015. URL https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf.

[14] *RF Antenna L=35mm MHF*. Bongguan Boju Electronics, 1991.

[15] International Finance Corporation. Safaricom feasability study. Technical report, Internat. Finance Corp., 2012.

[16] Inc. E. Rescorla RTFM, Inc. N. Modadugu Google. Datagram Transport Layer Security Version 1.2. RFC 6347, ietf, January 2012. URL https://tools.ietf.org/html/rfc6347.

[17] Polar Instruments. Cits25. http://www.polarinstruments.com/products/cits/cits25support.html.

[18] lady ada. *Adafruit FONA 3G Cellular + GPS Breakout*. Adafruit Industries.

[19] *AN1088 - Selecting the Right Battery System For Cost-Sensitive Portable Applications While Maintaining Excellent Quality*. Microchip Technology Inc., 2007.

[20] *MCP73834 - Stand-Alone Linear Li-Ion/Li-Polymer Charge Management Controller*. Microchip Technology Inc., 2009.

[21] *LoRa RN2483*. Microchip Technology Inc., 2015.

[22] S. Deering Cisco R. Hinden Nokia. INTERNET PROTOCOL. RFC 2460, ietf, December 1998. URL https://tools.ietf.org/html/rfc2460.

[23] *LPC1768/66/65/64*. NXP, 2009.

[24] Information Sciences Institute University of Southern California. Transmission Control Protocol. RFC 793, ietf, September 1981. URL https://www.ietf.org/rfc/rfc793.txt.

[25] J. Postel. User Datagram Protocol. RFC 768, ietf, August 1980. URL https://tools.ietf.org/html/rfc768.

[26] DARPA INTERNET PROGRAM. INTERNET PROTOCOL. RFC 791, ietf, September 1981. URL https://tools.ietf.org/html/rfc791.

[27] *LoRa FAQ*. Semtech, 2015.

[28] *LoRa Modulation Basics*. Semtech, 2015.

[29] *AT Command Set SIMCOM_SIM5320_ATC_EN_V2.02*. SIMCom, .

[30] *SIM5320_Hardware Design_V1.07*. SIMCom, .

[31] *SP6641A/6641B - 500mA Alkaline DC/DC Boost Regulator in SOT-23*. Sipex, 2005.

[32] Inc. T. Dierks Independent E. Rescorla RTFM. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, ietf, August 2008. URL https://tools.ietf.org/html/rfc5246.

[33] *Using SN65HVD96 to Create a Power-Over-Data and Polarity Immunity Solution*. Texas Instruments Inc., 2013.

[34] *LM3670 Miniature Step-Down DC-DC Converter for Ultralow Voltage Circuits*. Texas Instruments Inc., 2016.

[35] Z. Shelby ARM K. Hartke C. Bormann Universitaet Bremen TZI. The Constrained Application Protocol (CoAP). RFC 7252, ietf, June 2014. URL https://tools.ietf.org/html/rfc7252.

[36] Fawwaz T. Ulaby. *Fundamentals of Applied Electromagnetics*. Pearson International, Reading, Massachusetts, 2014.

[37] R.C. van Beelen and S.C.A. de Groot. Sensorsysteem voor subdowser.

[38] M.A. van Remundt and T.J. Witte. Keeping water under surveillance - control.

[39] R. Fielding UC Irvine J. Gettys Compaq/W3C J. Mogul Compaq H. Frystyk W3C/MIT L. Masinter Xerox P. Leach Microsoft T. Berners-Lee W3C/MIT. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, ietf, June 1999. URL https://www.ietf.org/rfc/rfc2616.txt.

[40] Wikipedia. Lithium-ion battery. https://en.wikipedia.org/wiki/Lithium-ion_battery, . Online; accessed June-2016.

[41] Wikipedia. Nickel–cadmium battery. https://en.wikipedia.org/wiki/Nickel-cadmium_battery, . Online; accessed June-2016.

[42] Wikipedia. Nickel-metal hydride battery battery. https://en.wikipedia.org/wiki/Nickel-metal_hydride_battery, . Online; accessed June-2016.

[43] Wikipedia. Lead-acid battery. https://en.wikipedia.org/wiki/Lead-acid_battery, . Online; accessed June-2016.