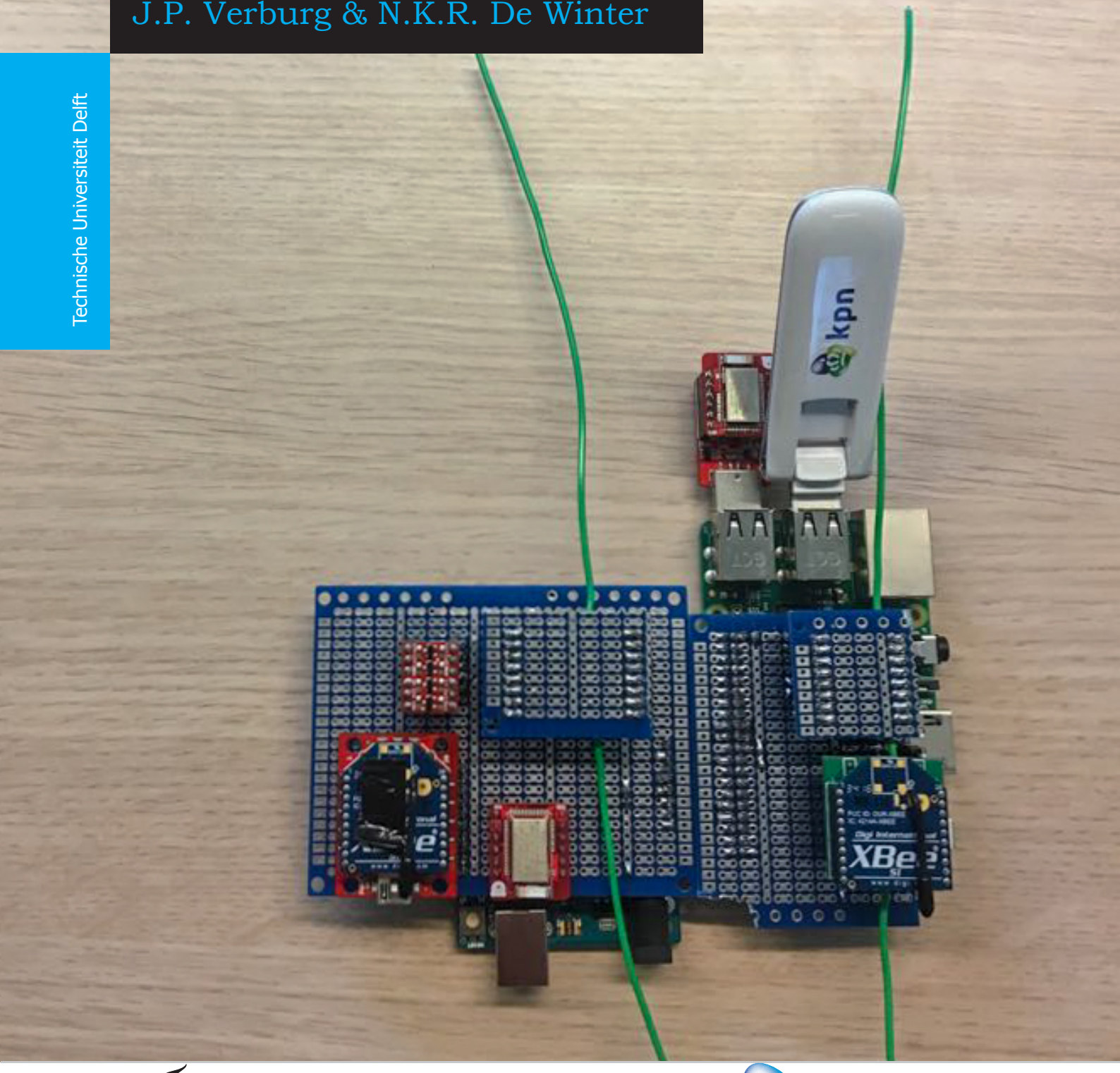
Smart sensors and communication using Internet of Things in supermarkets

Sensor to Server communication

J.P. Verburg & N.K.R. De Winter

Technische Universiteit Delft



TU Delft Delft University of Technology

kpn

Smart sensors and communication using Internet of Things in supermarkets

Sensor to Server communication

by

J.P. Verburg & N.K.R. De Winter

in partial fulfilment of the requirements for the degree of

Bachelor of Sciences
in Electrical Engineering

at the Delft University of Technology,

Students:	J.P. Verburg	4018575
	N.K.R. De Winter	4325052
Project duration:	March 24, 2017 – July 7, 2017	
Supervisor:	Dr. J. Hoekstra (TU Delft)	
	Ir. P. Marcelis (KPN)	
	B. Frens (KPN)	

This thesis is confidential and cannot be made public until July 7, 2022.

1

Abstract

During 9 weeks of investigation multiple solutions for implementing a communication module in a supermarket have been explored. These results led to three options which have been further researched. Bluetooth Low Energy (BLE), 433 MHz and ZigBee have been selected as the best communication protocols for sensor to gateway communication. Chips for these protocols have been bought and for BLE and 433 MHz libraries have been developed. For the gateway and the communication to the server a Raspberry PI and Mobile Internet have been selected and these have been integrated.

During the measurements we came to the conclusion that for the RFM69HCW (433 MHz) and the BLE Nano (BLE) the sleep mode did not work as specified. Resulting in high power consumption and thus worse battery life (sleep mode consumption for RFM69HCW 3.37 mA, BLE Nano 4.38 mA and for XBee 43.2 μA). The final conclusion of this thesis is that, with the current research, XBee (ZigBee) is the best communication protocol for communicating within a supermarket. But in further research BLE Nano and RFM69HCW can perform better if the sleep mode is improved to work according to what is specified.

2

Acknowledgement

This thesis has been produced by two student but without the support of our supervisors we would not have been able to conduct all the research. Thanks for your support and feedback, Paul Marcelis, Jaap Hoekstra and Bart Frens.

We also would like to thank KPN New business for giving us the opportunity to do this project.

Contents

3

Introduction

This report is the result of 9 weeks of research and development for KPN. In this introduction we will explain the background first. The second part will be about the problem definition. After this the thesis setup will be highlighted in the synopsis.

3.1. Background

Internet of Things (IoT) is the next big thing in our ever changing society. IoT is adding Internet to (household) apparatus to make them smart. Slowly companies are investigating the possibility to enhance their performance using IoT. The subject of our research is to investigate the possibility of enhancing a supermarket performance and safety with smart sensors. This is a cooperation between the TU Delft and KPN. And it is an actual business case for KPN. For KPN's client enhancing the customer satisfaction and safety is a key component for this project.

3.2. Problem definition

KPN has given us the opportunity of selecting our own projects in the mainframe of developing smart sensors. These projects are completed in groups of 2 students. The projects chosen were:

- Fruit and shelf observations
- Obstacle detection in hallways and near emergency exits
- Communication of the data from the sensor to the internet.

We have been tasked with investigating the communication of the data from the sensor to the internet. The deliverable of our project is an easy to add system that can be added to an existing

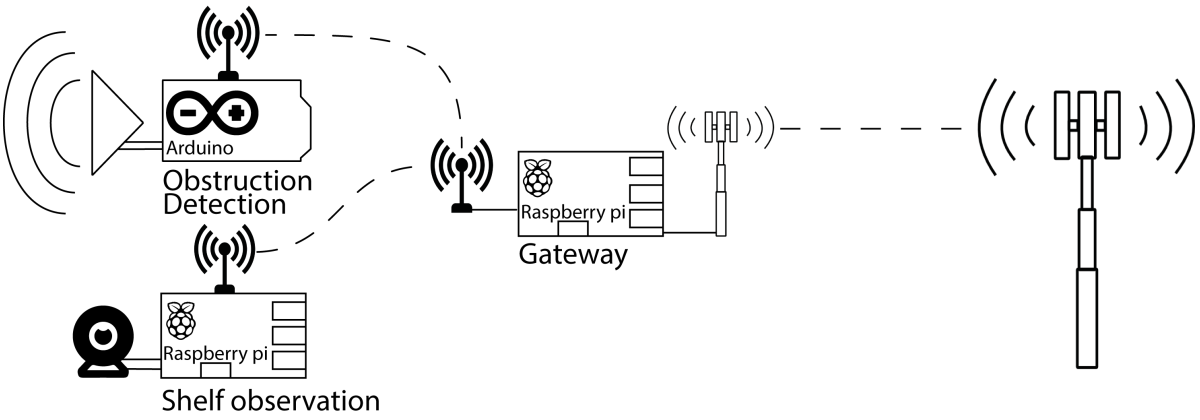


Figure 3.1: Total overview of the system.

platform. This addition will communicate with a gateway which will process the data and send it to the server.

3.3. Thesis synopsis

The investigation of the communication module will take place in 4 steps. First we will split the product in the overview. After that each of the modules will be discussed and the chosen technologies will be explained. After this the technologies will be compared based on our measurements and criteria. And finally our conclusion and recommendations will be further explained.

3.4. KPN restrictions

Because we do our Bachelor End thesis at KPN there are some restrictions. These restrictions prohibit us from talking about any of KPN's clients and any specifications of the KPN owned networks.

4

Total system overview

4.1. Functional requirements

KPN's client asked KPN to investigate the possibility of implementing sensors in their supermarkets. These sensors will have to be connected with some kind of network. This task has been handed to us. KPN and its client have given us the freedom to specify the technical requirements ourself, these are:

- **Easy to implement on the sensor**
  This in order to make the module as versatile as possible. This means that it will most probably be a breakout board or two wire based design to connect to the sensor.
- **Lowpower on the sensor side**
  Some of the sensors in the supermarket will be battery powered. For the rest of the network this is not mandatory.
- **Independent of any existing local network**
  KPN asked us to make the communication network independent of the local network. This is because local network is not always available or unreliable.
- **Is easily extendable**
  In the total project there will be two sensors. But after this proof of concept the number of sensors could grow substantially. This means that the network has to have the possibility to scale.

We decided to split the problem into three parts, this orthogonalizes the problems into more manageable parts. These are:

- Sensor to gateway
- Gateway
- Gateway to internet

4.1.1. Sensor to gateway

Communicating from a sensor to a central device in the supermarket from now on called the gateway. This communication is the most important part of our research because it has the most restrictions and thus is the most complicated to design. The criteria for this component are defined in Section 5.1.

4.1.2. Gateway

Secondly it is needed to have a higher power device that aggregates all the data and communicates this to a central server. It's important that this device operates reliably. All the criteria are further explained in Section 6.1.

4.1.3. Gateway to internet
In order to get the data to the server a communication method must be used. In Chapter 7 the different methods and technologies available will be looked at.

4.1.4. Out of scope
Outside of our scope is the implementation of the central server and the use and analysis of the data on that server. This has been decided because it is not closely related to Electrical Engineering and would increase the size of our scope to much.

4.2. General system requirements
Every part of our design has it's own requirements but there are a few components that apply to all of them we discuss those here.

4.2.1. Production and putting into use
The development is focused on a proof of concept device. After the finalisation of the project and with approval of the client the product will be moved to a different department that will handle the large scale implementation of the product. The requirements to enable the production are:

- Documentation of the software with clear installation instructions.
- Documentation of the products used and how to connect them.
- List of suppliers for the used components.

4.2.2. Discarding
The buildup of the system is very modular to facilitate reuse-ability. After the product is decommissioned it can be broken down into separate components to be reused in new projects. The only thing that needs to be taken into account when producing is that the electric connections used between these components is easily dis-connectable. This would also make repairs cheaper since only parts need to be replaced and not the whole system.

4.2.3. Development of manufacturing methodologies
The final large scale production of the product will be handled by a different department so we need to only adhere to the requirements already specified in Section 4.2.1.

4.2.4. Liquidation Methodologies
When the product is decommissioned, it should be handled like other electronic components. This means that working components be reused and most metals of broken components be reused. The only two exceptions is that the storage of the gateway needs to be properly erased or destroyed and that encryption keys need to be removed from the modems.

4.3. Ecological embedding in the environment
The device will be installed in a supermarket, the only requirement from the client is that it does not disrupt customer experience. This is easily done by installing the device in an out of view location. For example above a system ceiling.

5

Sensor to gateway

The first part of our systems and research focuses on getting the data from the sensor to the gateway. In the following paragraphs every method will be elaborated by looking at our requirements.

5.1. Requirements

The requirements for the sensor to gateway communication are:

- **32+ bits payload support**
  To give the sensors enough room to transmit all essential data a 32+ bits payload has been chosen. This size is a default size for floats and integers.
- **Unique id (UUID [128 bit])**
  In order to identify our sensors an unique id has to be added to each sensor, an UUIDv4[1] scheme is used because this is a widely used standard and easy to implement.
- **Low power (if battery based)**
  Some of the sensors will be battery powered, by using too much power for transmission of the data the battery can be drained very quickly. This means that our method should be low power. As guidelines we used that the transmission current should be lower then 50mA and sleep current should be lower then 1mA
- **Easy to interface with communication chip (I2C[2], SPI[3], UART/Serial[4])**
  To facilitate easy interfacing a common and easy to use communication protocol will be required.
- **Sensor health transmitting**
  Sensors that have temporal components (batteries for example) need to transmit the status of those components for maintenance.
- **Availability**
  Is the technology available for purchase, reasonably priced and usable for commercial use.
- **Secure**
  Transmission of information needs to happen securely. This means that an recognised encryption algorithm needs to be used.

5.2. Communication technologies

In this section the different communication technologies from the sensor to the gateway will be analyzed. We have selected the most common and most available technique based on papers from the IEEE database [5]. These are:

- Zigbee
- Z-Wave
- LoRa
- 433 MHz (RF69HCW)
- Wifi
- Dash 7
- Bluetooth 4.2
- Bluetooth Low Energy

5.2.1. Zigbee

Zigbee is a standard protocol in the world of low power communication. The modules are specially designed to be easily usable and configurable.

Frequencies

The frequencies the Zigbee protocol uses are 2.4 GHz, 915 MHz and 868 MHz. The data rate of the system depends on the frequency used, comparison is found in Table 5.1.

In Table 5.1 it is clear to use the 2.4 GHz frequency due to the high data rate. But taking a closer look at the objective of this research, a lower bandwidth could be good enough. By using a higher frequency the range is significantly lower than using the 868 MHz according to the "Hands-on zigbee implementing 802.15.4 with microcontrollers" [6]. This is another argument for using a lower frequency. Thus it is the best to use the ZigBee on a 868 MHz signal because it full fills all our needs. ZigBee uses 2 modulation techniques OQPSK for 2.4 GHz and BPSK for 868 MHz

Frequentie	Datarate
2.4 GHz	250 kb/s
915 MHz	40 kb/s
868 MHz	20 kb/s

Table 5.1: Frequency datarate table

Security

As mentioned in the criteria, security is an important part of our solution. The process of starting a Zigbee network starts with a beacon. The beacon is needed to detect any nearby Zigbee networks. It starts by broadcasting on all 16 channels until it finds a network to communicate with. In order to receive the necessary information during the initialisation the Personal Area Network (PAN) and the Media Acces Control (MAC) layer are not encrypted for these messages [7]. The security is done based on AES-128. The master key in the network is predefined and must be transferred using a secure medium (pre-programmed or transport). It is not possible to establish a link without the masterkey. An XBee without encryption can never join an encrypted network without a physical way of getting a master key.

Power

According to the XBee datasheet [8] the consumption of the 1 mW transmit variant is 1.65 mW. The 1 mW transmit power allows us up to 100 meters of range.

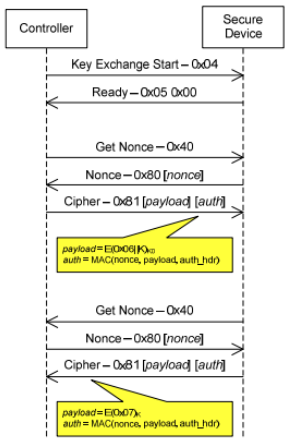


Figure 5.1: Z-wave setup connection by pairing with pseudo random codes. Afterwards synchronisation of the cipher will commence [10].

.

Criteria matched list

32+ Bit payload	Yes, 100 Bytes
UUID	Yes
Low power	Yes
Easy communication with chip	Yes, Serial
Sensor health	No, not without extra chip
Available	Yes, multiple sources
Secure	Yes, AES-128 using preshared key
Pro's	Many affordable implementations with good software support
Cons	None

5.2.2. Z-Wave

Z-wave is a 868 MHz communication protocol (in Europe). The protocol is a source-routed mesh network, this means that there does not have to be a direct connection between each node in the sensor network. Example: is "A" to "C" is not directly possible. If "B" is within reach of both parties the connection will go through "B". If "B" is not available or not on. "A" will try each node in the network before eventually ending its efforts.

Frequency

As mentioned before Z-wave uses the 868 MHz frequency. It has a capacity of 100 kbit/s and this is more than enough for our intended use. The used bandwidth is 1 MHz [9]. Z-wave uses GFSK modulation.

Security

Setting up a network is done by pairing devices to each other. This must be done by manually pressing the connect button. After this the keys are generated pseudo randomly [10]. After this the commands can be sent securely using the new pseudo random key. This makes it possible to exchange a random cipher for further communication. This makes sure the encryption is secure. If a synchronisation goes wrong, it will accept the next 256 pseudo random codes. If these do not match up, the connection is lost and the devices have to be paired again. This process can be seen in Figure 5.1.

Power

Due to this constant looking for new messages to redirect and receive, the power efficiency for our intended use is too high [11]. Although the datasheet of the ZW0201 mentioned it runs on 171,6 mW in normal mode [9].

Criteria matched list

32+ Bit payload	Yes, 64 bytes
UUID	Yes
Low power	No
Easy communication with chip	Yes (SPI)
Sensor health	Yes
Available	No, currently not available in easy to use breakout board, only single chips
Secure	Yes, AES-128 using established key
Pro's	Has all the possibilities for our solution
Cons	Is not available

5.2.3. LoRa

LoRaWAN is short for Long Range Wide Area Network. The technology was developed to be implemented in remote sensors with a long battery life and a long range. The frequency band used specifies a maximum transmit duty cycle of 1% this is to allow many devices to coexist.

Frequency

The system operates at 868 MHz and has a capacity between 59 and 230 bytes of payload per message. This is more than enough for our use. Lora uses the chirp spread spectrum modulation technique.

Security

Unlike the other protocols LoRa is used for very long range (>1 Km) this means that it is not always possible to do the authentication by hand. LoRa has a NetworkSessionKey and ApplicationSessionKey (managed by KPN). NetworkSessionKeys are used for signing network requests so the network server can check authenticity. ApplicationSessionKey is used for encrypting the message payload end-to-end (from device to application server). Payload encryption is done using AES-128. A visual representation of this can be found in Figure 5.2. These session keys can be programmed in the factory or established when the device first joins the network using an factory programmed ApplicationKey [12].

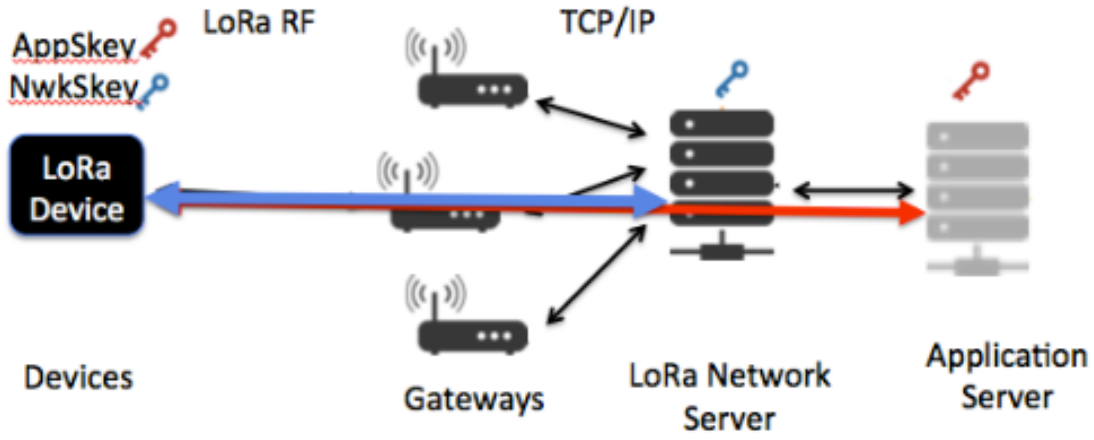


Figure 5.2: LoRaWAN key use [13]. NetworkSessionKey (NwkSkey) used for signing the network message. ApplicationSessionKey (AppSKey) used for encrypting the payload end-to-end (from device to application server).

Power

LoRa is specifically developed to be low power. Unfortunately this means that the indoor coverage is not optimal. Considering our case a supermarket that means the communication is possibly close to zero. A way to resolve the issue is having a a LoRa hotspot in the store. But by doing so it is hard to not go over the 1% duty cycle rule [14].

Criteria matched list

32+ Bit payload	Yes, 59 bytes minimum
UUID	Yes
Low power	Yes
Easy communication with chip	Yes (SPI or Serial)
Sensor health	Yes
Available	Yes, multiple sources
Secure	Yes, AES-128 using ApplicationKey
Pro's	Has a lot of options and possibilities for our solution
Cons	Will violate the 1% rule if a hub is used

5.2.4. Wifi

Wifi is a certification label given out by the WiFi alliance. WiFi is mostly used to connect to the internet but it is also possible to make a network for sensors.

Frequency

Wifi works on 2.4 GHz and has a bit rate of up to 150 Mb/s. This is more than enough for the use it will have in this project. There is a 5 GHz alternative which has the capacity of delivering 780 Mb/s. This means that under the same power output the range of the 5 GHz signal will be lower. Wifi uses BPSK, QPSK, 16-QAM, 64-QAM or 256-QAM modulation techniques. The newest version of the Wifi standard only uses 64-QAM or 256-QAM.

Security

Wifi has 3 different security protocols:

- **WEP**
  Wired Equivalent Protocol is the weakest form possible of encryption. It uses RC4 encryption, this method does an XOR operation with the data and the key. This result is sent wirelessly to the receiver who has the same key and does the same operation in reverse. Because of this easy way of encrypting it can be cracked in seconds. It is now retired, although RC4 128bit is still used in other encryption methods.

- **WPA** Wifi Protected Access (WPA) is an improved version of WEP. It still uses RC4 128bit encryption but also includes an initialisation vector. This vector can be implemented in several ways. The most common is Wifi Protected Service (WPS). By adding this vector the data becomes even more encrypted and it is assured the data is almost safe.

- **WPA-2** WPA-2 is the second version of WPA and the main change was that the 128 bit encryption was changed to AES 128 bit encryption. This method is very secure and is widely used all over the world.

Power

These secure protocols come at a cost. The relative cost (money) of WPA-2 Wifi chips is not that high (around €5 [15]) but the cost in power consumption is very high (850 mW [16]).

5.2.5. Dash 7

Dash-7 is a protocol developed by the DASH7-Alliance. It is very new and only used in a few small projects and in 1 car park [17]. This makes working with this protocol troublesome because available products, documentation en libraries are limited.

Frequency

Dash-7 operates at 433, 868 & 915MHz. These frequencies enable the data to be transmitted through water and concrete. This makes it great for an indoor sensor network. Dash7 has the capacity to transport 167Kb/s. The protocol has several topologies: Node-to-node, Star & Tree (see Figure 5.3). These can be configured beforehand, allowing for indirect connections with the host provider.

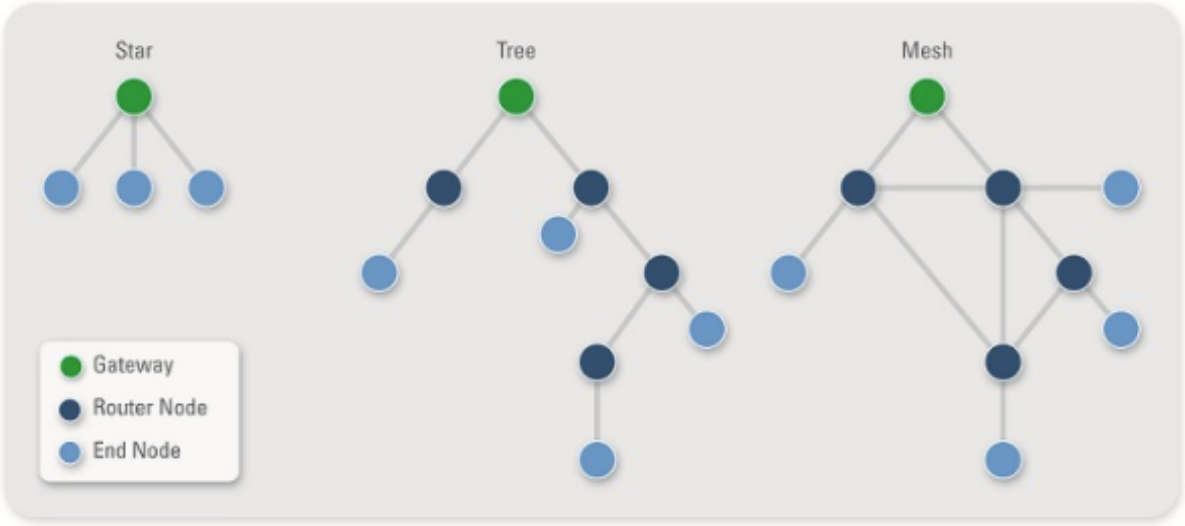


Figure 5.3: Network Topologies of Dash-7 [18], freely configurable and thus usable for many kinds of setups

Security

Dash-7 uses AES-128 bit encryption with a preshared key. There is not much known about the full implementation because it is still in the development stage [19].

Power

Since the Dash-7 protocol is still in development it is not known how much power is needed for transmission.

Criteria matched list

32+ Bit payload	Yes, 256 bytes
UUID	Unknown
Low power	Yes
Easy communication with chip	Unknown, chips not available
Sensor health	No, chips not available
Available	No, currently not available in easy to use breakout board, only complete (very ex
Secure	Yes, AES-128 using preshared key
Pro's	High payload
Cons	Is not commercially available

5.2.6. Bluetooth 4.2

A more common communication protocol is Bluetooth. The node-to-node structure is used a lot for data transfer and for short range communication to PC peripherals like keyboards, mouses and headsets.

Frequency

Bluetooth works on 2.4 GHz. It has 79 channels each with a bandwidth of 1 MHz. The system is designed to use adaptive frequency hopping in order to increase the chance of a successful transmission. The system changes frequency around 800 times per second. The standard uses 8DPSK or $\frac{\Pi}{4}$ QPSK modulation this enables the system to transmit data at 2 and 3 Mbit/s respectively [20].

Security

Bluetooth is very secure. It uses a AES-128 encryption. The connection is setup by using a pin code. This can be a predetermined pin code or randomly generated code. After key exchange is finished the secure transmission of data can start.

Power
Nowadays Bluetooth is mainly used for large packets of data transfer in short ranges. These can include up to 672 bytes of payload and take up to 100 ms of transmission time. This can be done by using 10 mW of power, allowing a range up to 100 m.

Criteria matched list

32+ Bit payload	Yes, 672 bytes
UUID	Yes
Low power	No
Easy communication with chip	Yes, several options
Sensor health	Depends on module
Available	Yes
Secure	Yes, AES-128 using established key
Pro's	High payload and many (extra) options
Cons	Not power efficient

5.2.7. Bluetooth Low Energy
In the year following the release of Bluetooth 4.2. A new market was beginning to develop, IoT. Bluetooth Low Energy (BLE) is a smaller, less power consuming variant of Bluetooth designed to be used in low power devices (battery powered).

Frequency
The frequency of BLE is the same as for Bluetooth, 2.4 GHz. It uses GFSK modulation [21] with a modulation index of 0.5 +- 1%. The modulation technique represents a Binary 1 as a positive frequency shift. The minimum frequency deviation is 185KHz, the maximum is 1MHz.

Security
In security BLE is very different from normal Bluetooth. BLE uses AES-128 encryption combined with a circular encryption key. This means that the there are more keys, each time a message is sent a different key is used. By adding this feature the data is almost guaranteed to be transmitted securely.

Power
BLE is optimised for low power consumption. The way it improved from the normal Bluetooth is by sending shorter messages. The message can only take up to 3 ms (compared to the 100 ms of Bluetooth).This means that the data rate is significantly lower than that of Bluetooth. But this leads to very low power consumption. The longest data package BLE is able to send is 20 bytes while Bluetooth has a max payload of 672 bytes.

Criteria matched list

32+ Bit payload	Yes, 20 bytes
UUID	Yes
Low power	Yes
Easy communication with chip	Yes, UART and SPI
Sensor health	No, not directly
Available	Yes, convenient development kits available
Secure	Yes, AES-128 using established key
Pro's	Possibly very energy efficient
Cons	None

5.2.8. 433 MHz
433 MHz is a frequency on which many protocols operate. It is a very common frequency for in house operations because of its long range and low cost.

Frequency
As mentioned before the modules operates at 433 MHz. It was decided to look at the RF69HCW chip [22]. This chip has a bandwidth of 1 MHz and uses FSK modulation.

Security
The RF69HCW has an built-in security encryption option. During the programming you can provide your own cipher. This is not the most secure method but it will work for our project. Encryption is done using an hardware implementation of AES-128.

Power
The power consumed by the RF69HCW is around 80 mW in receiving mode. This is not the mode we have to use because the sensor side will only have to send. During the sleep mode it uses around 5 μW and during wake up and transmission it uses maximum 250 mW [22].

Criteria matched list

32+ Bit payload	Yes, 64 bytes
UUID	No, but it can be generated
Low power	Yes
Easy communication with chip	Yes, SPI
Sensor health	No, not directly
Available	Yes
Secure	Yes, AES-128 using preshared key
Pro's	Low power and high range
Cons	None

5.3. Preliminairy results
For the rest of our study we have selected a few options. It was decided to buy the following protocols and modules based on the literature study:

- **Zigbee | XBee**
- **433MHz | RF69HCW**
- **Bluetooth Low Energy | BLE Nano**

5.3.1. Zigbee | XBee
For the Zigbee protocol it was chosen to use the XBee module. It is one of the most documented modules available with lots of customisation options. In Section 5.4 the module will be investigated further including a detailed explanation of some of its functions.

5.3.2. 433 MHz | RFM69HCW
The RFM69HCW is one of the more common modules when using the 433 MHz frequency. The module has an extensive library for both Arduino and Raspberry PI. The library enables us to use Acknowledgements, encryption and read out power levels so many of options can be explored in order to get the best result.

5.3.3. Bluetooth Low Energy | BLE Nano
The BLE Nano is a small programmable microcontroller with integrated BLE stack. Because it is programmable it can operate independently of an Arduino. It can be programmed via the Arduino software and is able to work with the Raspberry PI because of compatible logic levels. It looks very promising and easy to use. The BLE Nano is made by RedBear, a small start-up focused on building shields and chips for wireless communication.

5.4. Technology investigation
In this section the 3 chosen technologies will be investigated.

5.4.1. XBee settings
Within XBee there are a lot of options for device setup. In the following paragraph we will discuss the most important and decide if and how they will be used.

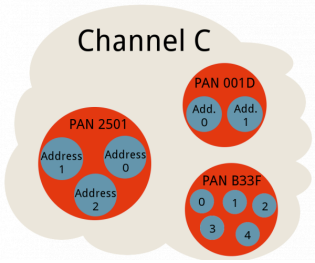


Figure 5.4: Network layout XBee, Using PAN Networks within a channel to assure less congestion.

AT & API mode

The most important setting in the XBee protocol is the use of AT (Transparent) or API (Application Programming Interface) mode.

AT mode is designed to communicate on a one-to-one base. This means that the send address is determined beforehand. The data has not a specified format. The XBee's will now communicate via a serial like communication.

API mode is specially developed for multi user messages. This is done by using a TCP like setup. Where the destination and payload are specified beforehand. The API mode is especially useful if the network is dynamic and data can come and go to each user in the network. It has built in ACK and can also do error detection by having a checksum.

Network settings

There are several network settings that need to be configured in a network of XBee's. These are "Channel" and "PAN ID". The Channel indicates the frequency the XBee will use. The PAN ID is the Personal Area Network ID, this indicates the local network within a frequency. A visual representation of this is seen in Figure 5.4.

RF and sleep Settings

In the RF settings the transmission power can be regulated. This can be done between 0 and 4. Where 4 is the highest and 0 is the lowest. In the default setting the power level is set to 4. The power ratio can be seen in Table 5.2.

0	-10 dBm
1	-6 dBm
2	-4 dBm
3	-2 dBm
4	0 dBm

Table 5.2: Transmission power of XBee.

Next to that there is the CCA value. This value is the threshold for determining if the network is free. This value is standard set to 0x2C (-dBm). In the Sleep settings the sleep mode, time till sleep and the cycle sleep period. For our use the sensor can sleep for most of the time. The best setting for this is pin hibernate. The function of this pin is to get the sensor out of hibernation. There is a second option called pin dose. This option is quicker but also consumes more power. The coordinator (placed on the Raspberry PI) is unable to sleep. Furthermore there are several options for time to sleep and cycle sleep period. These settings should be altered for each sensors. These will be different for each sensor.

5.4.2. 433 MHz

The RFM69HCW is a lot easier to setup. In this paragraph we will discuss the network settings and the sleep mode of the RFM69HCW.

Network settings

A network is defined by two pre-programmed variables. The network number and the node number. The network number identifies the network the sensor is in. There are 256 different network id's

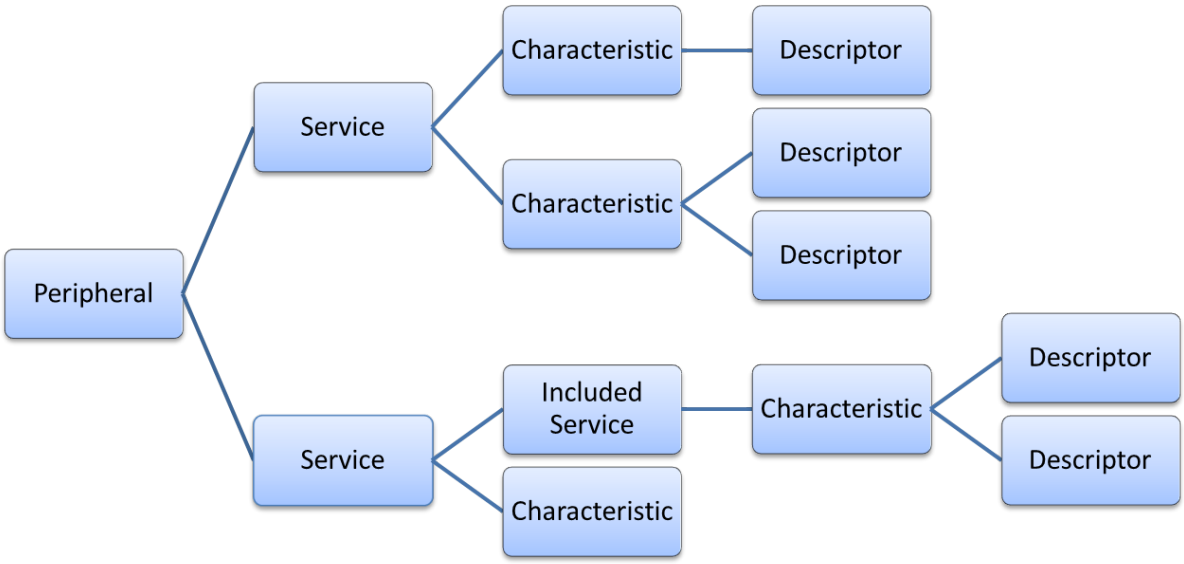


Figure 5.5: Layer structure of BLE, a device has 1 or more Services that contains Characteristics containing parts of a Service. For example a write-able configuration value.

read	the value can be read
write	the value can be written to
writeWithoutResponse	the value can be written to without application level acknowledgement
notify	can be subscribed to, will notify central of value changes
indicate	can be subscribed to with application level acknowledgement that a value change was properly registered

Table 5.3: Different Characteristic types, can be combined to create a both readable and write-able Characteristic.

possible. Within each network there are 255 (0-254) different node possibilities. Node id 255 is reserved for contacting the whole network (broadcast). There is also a setting on the chip called "Promiscuous Mode" in this mode every message sent within a network is received, especially useful for debugging.

Sleep mode

The RFM69HCW is designed with a built in sleep mode. This can be enabled by setting register 0x01 to 000. After doing this the whole system is in rest. To start up the chip will go into standby mode and thus powering up the local oscillator and the top regulator. After this the chip goes into FS (Frequency synthesizer) mode. After this the chip can go into two different modi (Sending and Receiving). The power up is complete. In sleep mode the power consumption is 0.1 μA [22].

5.4.3. Bluetooth Low Energy

BLE is designed with a lot of predefined product services to help encourage the design of inter-operable peripherals like heart rate sensors. All services on BLE peripherals contain so called characteristics they are meant to describe certain parts or sensors of a peripheral. Characteristics can have several properties that describe how can be interfaced with the characteristic. Further described in Table 5.3.

Some Characteristics have CharacteristicDescriptors which is read only information about the Characteristic. For example the rate and precision of a heart rate monitor. An overview can be seen in Figure 5.5.

Custom service

Because BLE does not contain an predefined Service for bi-directional communication we created our own Service using two Characteristics. One with the write property allowing communication from the central to peripheral and another one with the notify property allowing communication from the peripheral to the central.

5.5. Implementation

5.5.1. RF69HCW

The RF69 series chips is a well established 433 MHz implementation with many supported platforms. Furthermore the chips are very configurable making them usable for a wide set of scenarios.

Arduino side

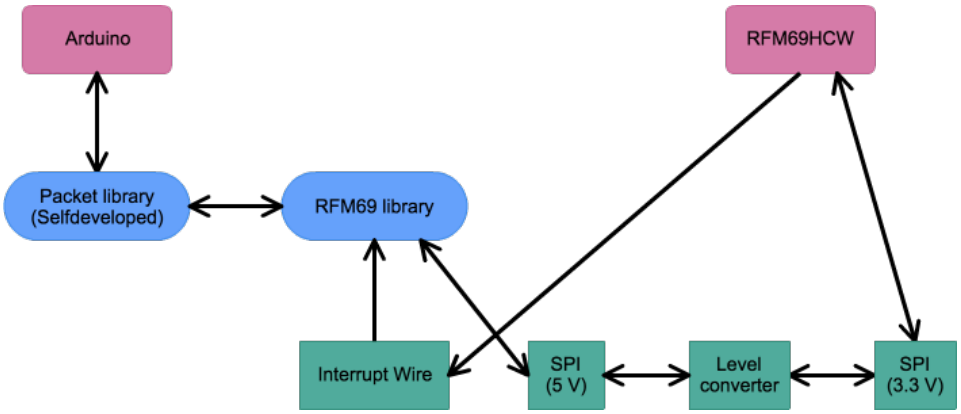


Figure 5.6: Software and hardware interfacing structure for Arduino to RFM69HCW, the Packet library was added to simplify communication. The interrupt wire was used to notify the Arduino of new received packets.

The software implementation of the Arduino side was very simple using the provided library [23]. The hardware side was more complicated since the logic level of the Arduino is different to the logic level the RFM69HCW operates on (5 V versus 3.3 V). This was solved using a logic level converter [24] between the RFM69HCW and Arduino, which correctly translates any 5 V signals to 3.3 V and from 3.3 V to 5 V. When connecting the RFM69HCW, SPI is used for data communication which is level converted as mentioned before. Besides the SPI interface there is also an interrupt wire that is used by the RFM69HCW to notify the Arduino of new received packets, luckily this did not needed to be level converted since it is a one way communication from the RFM69HCW to the Arduino and the Arduino inputs can operate on voltages as low as 3 V (0.6 * Vcc; Vcc = 5 V) [25]. A complete overview of these connections can be seen in Figure 5.6.

Raspberry PI side

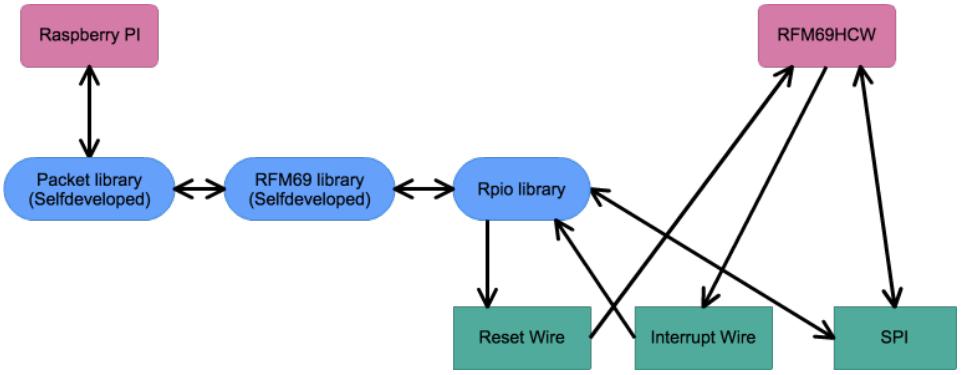


Figure 5.7: Software and hardware interfacing structure for Raspberry PI to RFM69HCW, the rfm69 library was made to replace the broken library and the packet layer was added to simplify communication. The interrupt line was used to notify the Raspberry PI of new packets. The reset line was used to make sure no old configuration settings remained of the broken library.

Sadly the RF69HCW library [26] found during the initial investigation did not function correctly, so library had to be written all over in order to communicate with the RFM69HCW. This was done using the rpio [27] library and our own code for handling the registers and state management. The datasheet [22] was

used to find the necessary values. All the data communication was done over SPI, one GPIO pin was used for handling packet received interrupts and one GPIO pin was for hard resetting the RFM69HCW configuration. The reset pin was not needed on the Arduino side because the library worked correctly, however because this was not the case on the Raspberry PI a method to hard reset the RFM69HCW configuration was needed. This was solved by connecting a GPIO pin from the Raspberry PI to the RFM69HCW. A complete overview of these connections can be seen in Figure 5.7.

5.5.2. BLE Nano

Getting the BLE Nano to work took significant time due to unclear documentation on how the configure and use the device. Extensive research into the BLE Nano source code was needed to found out how the BLE Nano works. It was found that there were a lot of quirks in the BLE implementation that needed to be circumvented, for example writing to and reading from characteristics worked differently on the central en pheripheral side. To avoid further problems an abstraction layer was written to hide and handle these quirks. On top of this abstraction layer an UART communication layer was built to allow communication to the Arduino and Raspberry PI.

Arduino side

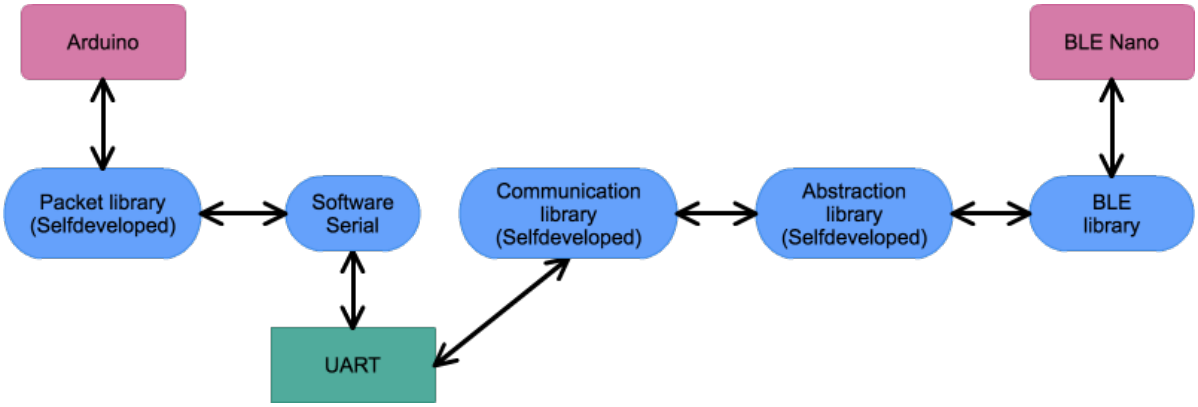


Figure 5.8: Software and hardware interfacing structure for Arduino to BLE, the abstraction layer was added to handle BLE Nano quirks, the communication layer was added to make interacing over UART easier and finally the packet layer was added to simplify communication.

On top of the UART connection a small library was built to simplify communication. The complete overview can be seen in Figure 5.8.

Raspberry PI side

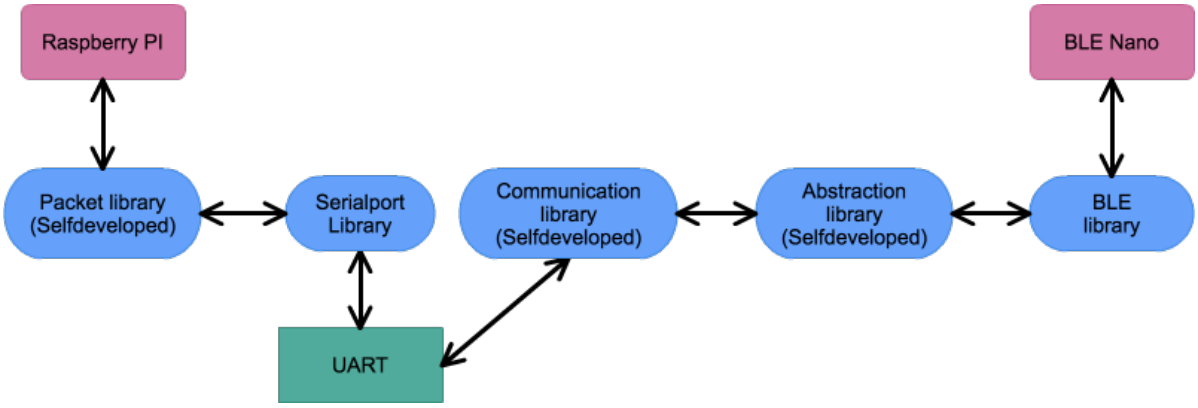


Figure 5.9: Software and hardware interfacing structure for Raspberry PI to BLE, the abstraction layer was added to handle BLE Nano quirks, the communication layer was added to make interacing over UART easier and finally the packet layer was added to simplify communication.

Initially it was tried to use the onboard BLE chip to implement the BLE communication sadly this did not work because of interoperability problems between the Raspberry PI BLE chip and the BLE Nano chip. After several attempts to get this working we chose to implement communication using an UART interface from the Raspberry PI to the BLE Nano. A simple library was built on top of the UART connection to simplify communication. The software connection to the UART interface was handled using the SerialPort library on the Raspberry PI side. A complete overview of these connections can be seen in Figure 5.9.

5.5.3. XBee

Initially it was attempted to implement communication using the API mode because this seemed the obvious choice. However this was not the case because it incurred too much overhead for our setup. Then the switch was made to the AT mode since it is a lot easier and faster to implement. Furthermore developing with these XBee modules is very convenient because they accept logic level from 2.3 V making it easy to directly connect to both Raspberry PI and Arduino.

Arduino side

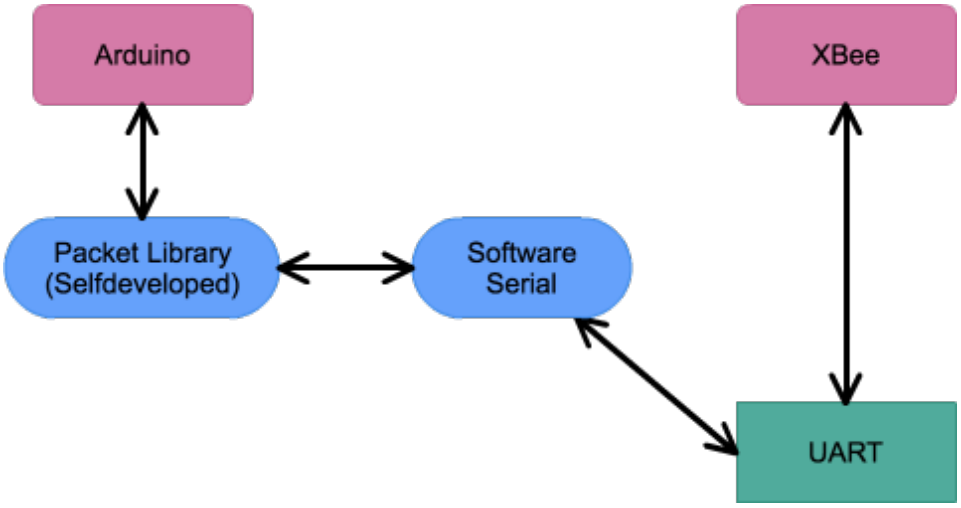


Figure 5.10: Software and hardware interfacing structure for Raspberry PI to XBee, the packet layer was added to simplify communication.

A simple packet based protocol was built on top of the serial (AT mode) interface.

Raspberry PI side

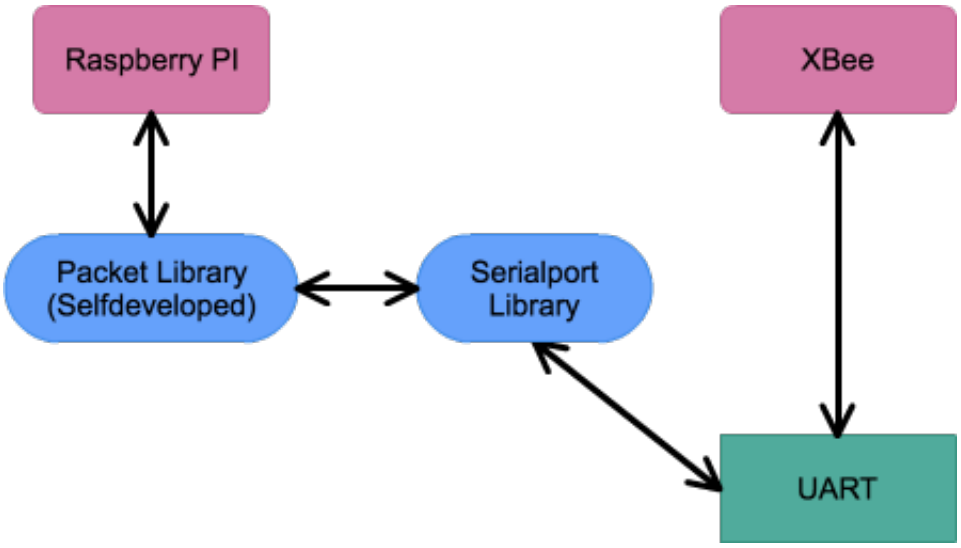


Figure 5.11: Software and hardware interfacing structure for Raspberry PI to XBee, the packet layer was added to simplify communication.

The same packet based protocol as created on the Arduino side was implemented but then in the NodeJs engine using the Serialport library [28] to interface to the UART connection.

6 Gateway

In the following chapter the gateway criteria and options will be discussed. After that, the systems chosen will be discussed and at the end the implementation will be explained.

6.1. Requirements

The Gateway will the main hub for connecting our network and the internet. It will monitor sensor health, clusters sensor updates and store information until it can be properly sent to the server. The criteria are:

- Can interface with UART, SPI and I2C.
- Has the possibility to connect to 2G, Ethernet and LoRa directly or indirectly.
- Has the possibility to store data.

The gateway does not have the same low power requirements since it will be placed in a fixed location in the supermarket where power is available. This location is also chosen for optimal signal reception for the internet signal.

6.2. Research

6.2.1. Computer

For our prototyping not a lot of processing power is required. This means that the most basic computer will full fill our need. But it needs to have the possibility to connect easily to USB, I2C, UART and SPI. The most easily available unit is the Raspberry PI. It has wired and wireless networking capabilities which makes it easy to connect with it. While at the same time having many available libraries to easily interface with peripherals.

There are a lot of Raspberry PI equivalents but these were not guaranteed to be compatible with the libraries previously found.

6.2.2. OS - Linux

From investigation the conclusion arose that a computer with Linux OS is the best solution to our problem. Linux OS is a development friendly OS and has many libraries for our communication modules (both internal and external). This allows for easy prototyping. Furthermore it has the possibility to house a database (depending on the size).

6.2.3. OS - Windows IoT

Windows is easy to install and use but it does not have a great development environment. At this moment SPI and I2C does not work on the Windows IoT OS. Thus meaning an Arduino will have to be used for some preprocessing. Because of the lack of these features Windows IoT is not the preferred choice.

6.2.4. Programming languages

For implementing on Linux a programming language was needed that had support for interfacing with low level interfaces (SPI, UART, I2C). C++, Python, Javascript and Java were found that met this requirement. It was decided to use Javascript since there was extensive previous programming experience in Javascript, also installing the engine (NodeJs) is very easy using the Debian package manager (Aptitude). Added benefit of using NodeJs is that it comes with a package manager for NodeJs modules called npm that allows easy installation of any needed extensions.

6.3. Implementation

The implementation was done using Raspbian [29]. A Debian based Linux operating system. Installing Raspbian was done by using Etcher [30] to directly burn an Raspbian image onto the SD card. Afterwards the Wifi chip was set into AP mode (Wifi access point) and a DHCP server was installed to allow to easily connect to the Raspberry PI for development purposes.

7

Gateway to internet

In this chapter the connection for the gateway to the KPN server will be investigated. Firstly the requirements will be explained. After this the technologies will be explained and lastly the chosen technology will be implemented.

7.1. Requirements

For the last part of our system there are the following requirements:

- **Independent of local infrastructure**
  KPN's client requested to built our network of sensors independent of the local infrastructure in the supermarket. This was asked because of the large variety of networks in supermarket currently available and these are not always available for our use.
- **Allowed to send continues data**
  As mentioned before some frequencies have a maximum amount of data that is allowed to be send in a certain time period. In the future there can be a lot of sensors in supermarkets, knowing this we have to implement this requirement. Minimum required bandwidth is 100 KBit/s.
- **Availability**
  Is the technique commercially available.

7.2. Technologies

In this paragraph the technologies that are currently available will be discussed. We have looked at:

- ADSL/Fiber/Cable
- LoRaWAN
- Mobile internet

7.2.1. ADSL/Fiber/Cable

These techniques would meet our second (has at least 1 MBit/s bandwidth) and third requirement (at least one of the techniques is available anywhere in the Netherlands). However the client required that no existing network would be needed. Also installing is not an option because it would be too invasive because it would require installing additional network equipment or not possible at all due to existing available infrastructure already being used by the clients equipment.

7.2.2. LoRaWAN

As mentioned in Section 5.2.3 the frequency band in which LoRa operates has the restriction that you can only send 1% of the time. Because of this restriction LoRa was not a viable option for such a (potential) large number of sensors.

7.2.3. Mobile internet

A mobile internet connection would meet the requirement of being independent of local infrastructure. Secondly mobile internet is fast enough for sending the data of many sensors. Lastly mobile internet is a readily available technology. Being the only viable option for the setup it was chosen to implement communication using mobile internet, more on this in Section 7.3.

7.3. Implementation

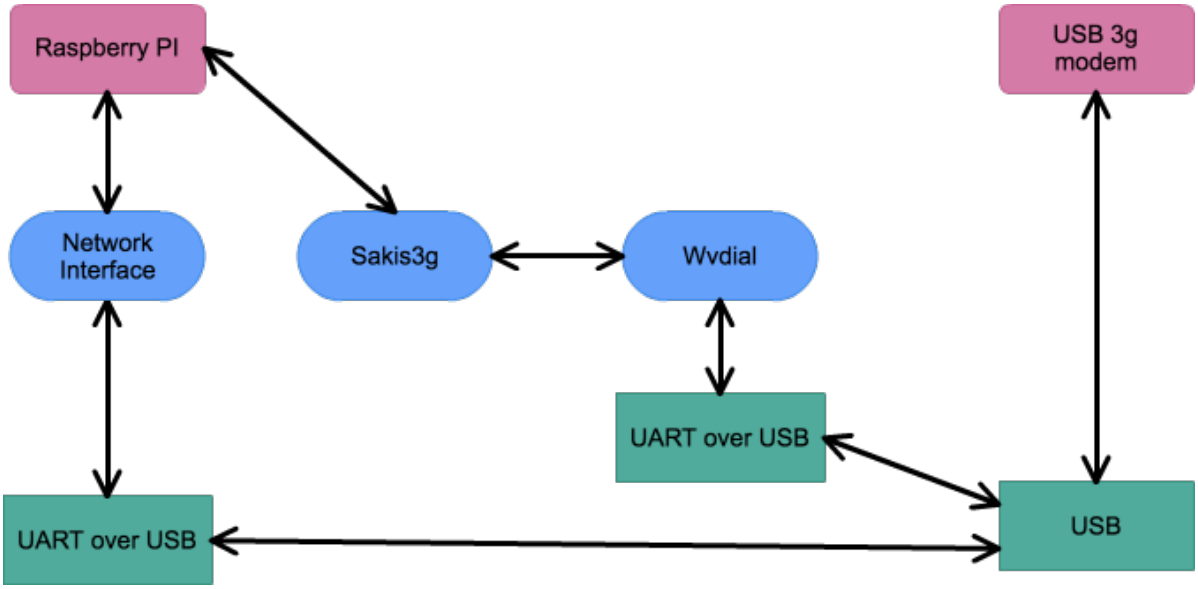


Figure 7.1: Software and hardware interfacing structure for Raspberry PI to 3G modem in connected state, showing 2 UART interfaces. One UART interface is used as data channel for data to the internet. The other UART channel is used for talking to the modem allowing to check connectivity status.

Working with KPN made it very easy to obtain a SIM card with activated data and a USB 3G modem to connect it to the Raspberry PI. An overview of all the interfaces in connected state can be seen in Figure 7.1. Getting a working internet connection was done by using Sakis3g, an application that handles the configuration of the modem and the setting up of the network interface. Sakis3g [31] uses Wvdial [32] to handle setting up the network interface to allow internet connectivity. This works by using the UART interface to send the configuration to the modem, this is done by Wvdial based on the provided configuration by Sakis3g. After the configuration is finished and the modem was able to connect a second UART interface is made available which is used for internet data communication.

When this second UART interface is set up Sakis3g creates a linux network interface to allow linux to communicate to the internet. The initial UART channel is maintained to be able to still communicate with the Modem, for example querying the connection status/quality.

8

Measurements

In Chapter 5 different options for transmitting data from the sensor to our gateway have been investigated. In this chapter the method of measuring its power consumption and the measurements itself will be discussed.

8.1. Method

The three systems that have been bought were XBee, BLE Nano and RFM69HCW. These will all be tested in several scenarios. The test will be conducted by measuring the current through the sensor except for the range measurement. In this test the RSSI value will be used. For each of the sensor there will be different tests.

8.1.1. Types of measurements

For all systems the following test will be conducted:

- **Sleep mode**
  For all modules the sleep mode will be investigated. XBee has 2 different modes. These will be compared. This test is relevant because the modules will be in rest for most of the time.

- **Transmission power**
  Some of the communication modules can transmit at different power levels. These will be investigated. This test is relevant to determine an optimal transmission rest time.

- **Range**
  Next to the transmission power the range module will be analized. This will be done by looking at the Received Signal Strength Indicator (RSSI).

- **Influence of encryption**
  For all system encryption will be mandatory. Thus it is interesting to look at the power use of this encryption.

XBee specific

For the XBee there are different sleep options. In all options the setting will be end device, this is chosen because all our sensor will be end devices. The network coordinator will be placed on the gateway. This is chosen this way because end devices can go into sleep mode.

8.1.2. Current measurement

In order to measure the current drawn by different modules a voltage divider will be used consisting of the module and a resistor. This creates a voltage across the resistance equally proportional to the current. The base voltage across the voltage divider is known thus knowing the current.

8.2. Measurement

In this section the measurements will be explained. This will be done by looking at each of the modules seperatly.

8.2.1. 433 MHz

As mentioned before the investigation will be split into 4 parts. These will be explained in the following paragraphs.

Sleep mode

In Appendix A.2 the current is shown of a module in sleep mode. The average current is 3.37 mA. This means that it can last up to 61.8 days in sleep mode on a 5000 mAh battery. The difference between sleep mode and and standby mode is 1.3 mA this decreases the lifetime to 44.6 days. The difference in use is that the device does not have to wake up. The wake up does not require more power than the transmission and takes only 120 μs. The datasheet [22] says that the sleep mode current should be around 0.1 μA, in our setup this was not the case. This difference is due to the module not properly entering sleep mode. In Section 9.4 some improvements will be discussed.

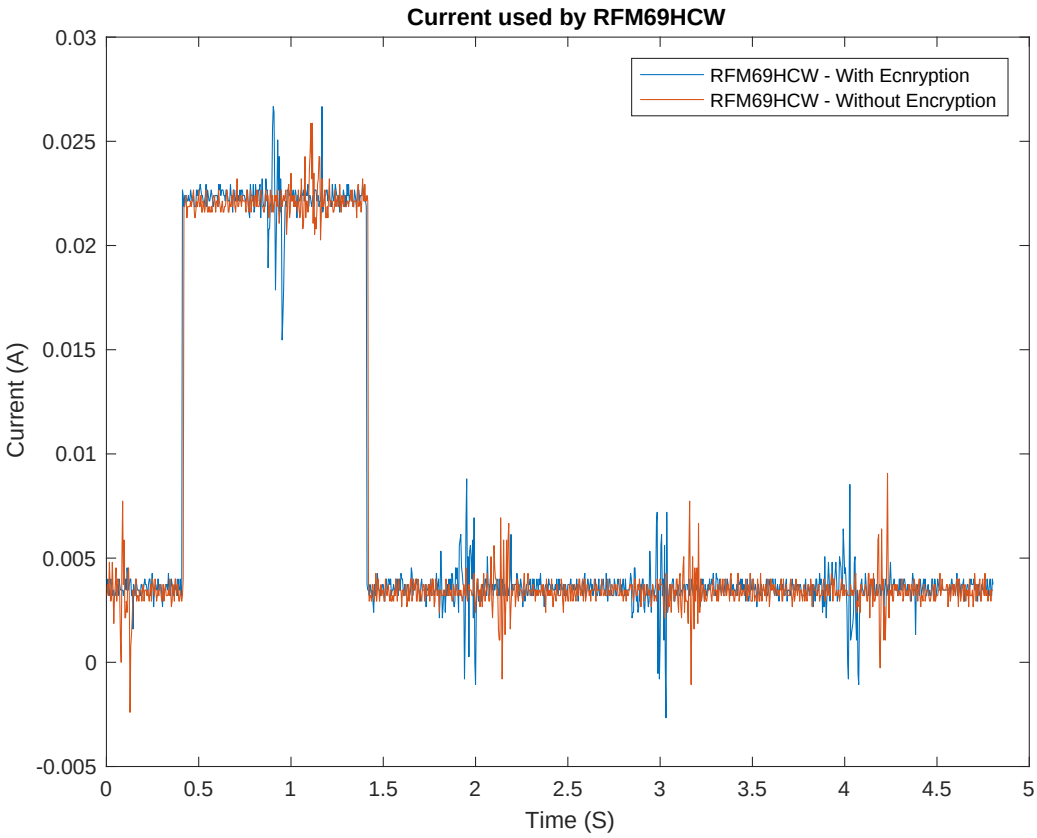


Figure 8.1: Measured current through the RFM69HCW during transmission, with and without encryption. Influence of encryption on power consumption is minor due to hardware support.

Transmission power

In Figure 8.1 one transmission is visible. It becomes clear that the standby current is 4.6 mA. The transmission current is visible as the block. This block is on average 23 mA.

Range

The RSSI plot of the RFM69HCW can be found in Figure 8.2. At a range longer than 10 meters there was no data received. Our recommendations on the antenna to increase the range can be found in

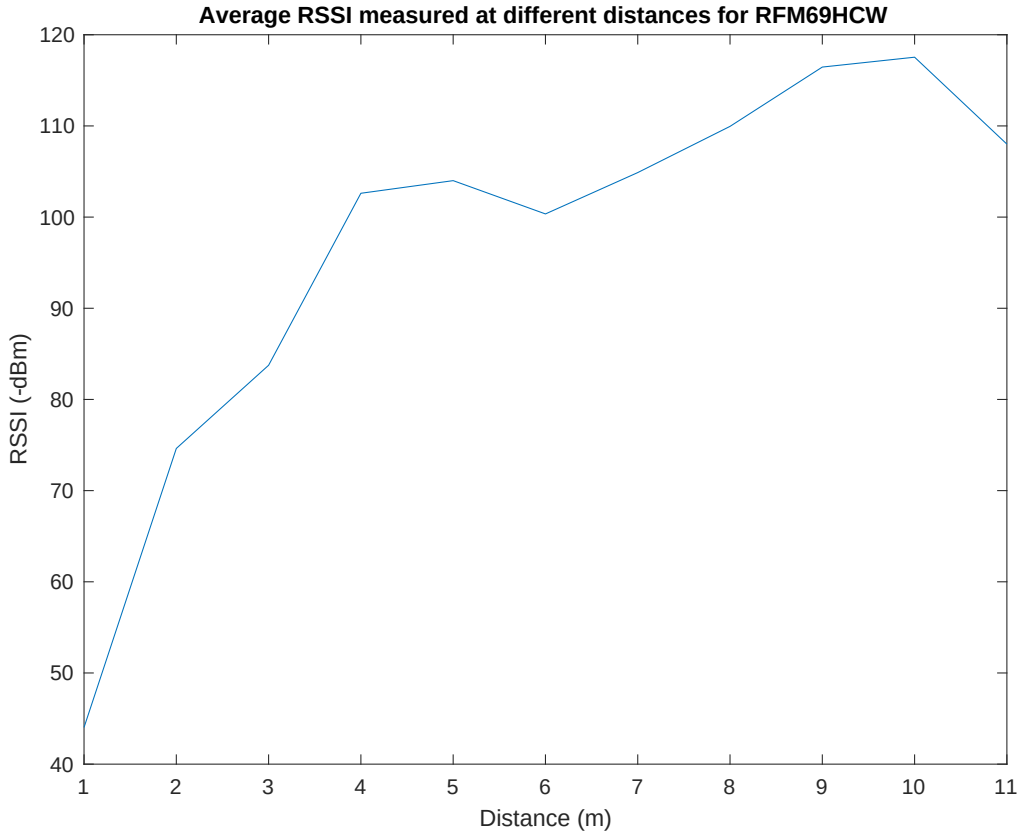


Figure 8.2: RSSI measurement for RFM69HCW. Average RSSI measured every meter. Decreasing RSSI can be clearly seen.

Section 9.4.

Influence of encryption

In Figure 8.1 the influence of encryption can be seen. The influence from the encryption is almost 0. The transmission peak is a bit later because of the encryption. This means that it takes a bit (0.2 s) longer to compute, this is not a problem because the total time is still one second.

8.2.2. XBee

The measurement for the XBee have all been conducted on the lowest power level and with the device in end node mode. API was enabled on the receiver in order to receive the RSSI.

Sleep mode

XBee has 2 different sleep modes, Hibernate and Dose. The difference between these two is the amount of time required for wake up and the power required. Dose uses more power but is also much quicker 2.35 ms compared to 13.2 ms according to [8]. The sleep current is according to the datasheet 50 μA and 10 μA. From this datasheet it is clear that for sensors that do not have to send a lot of data and that have the possibility of waiting 13.2 ms hibernate is the best option.

When looking at Figure 8.3, it can be concluded that the Dose function is indeed much faster than the Hibernate function. Taking more than twice as long. The current used by the chip is not increased severely. The average current used in Hibernate is 43.2 μA and in Dose it is 73.01 μA. The corresponding Figure can be found in in Appendix A.1.

This means that on average current the hibernate function uses is 40.8% less. In practice this would mean that on a normal (5000 mAh battery) the lifetime will in be increase form 7.81 to 13.2 years when sleeping

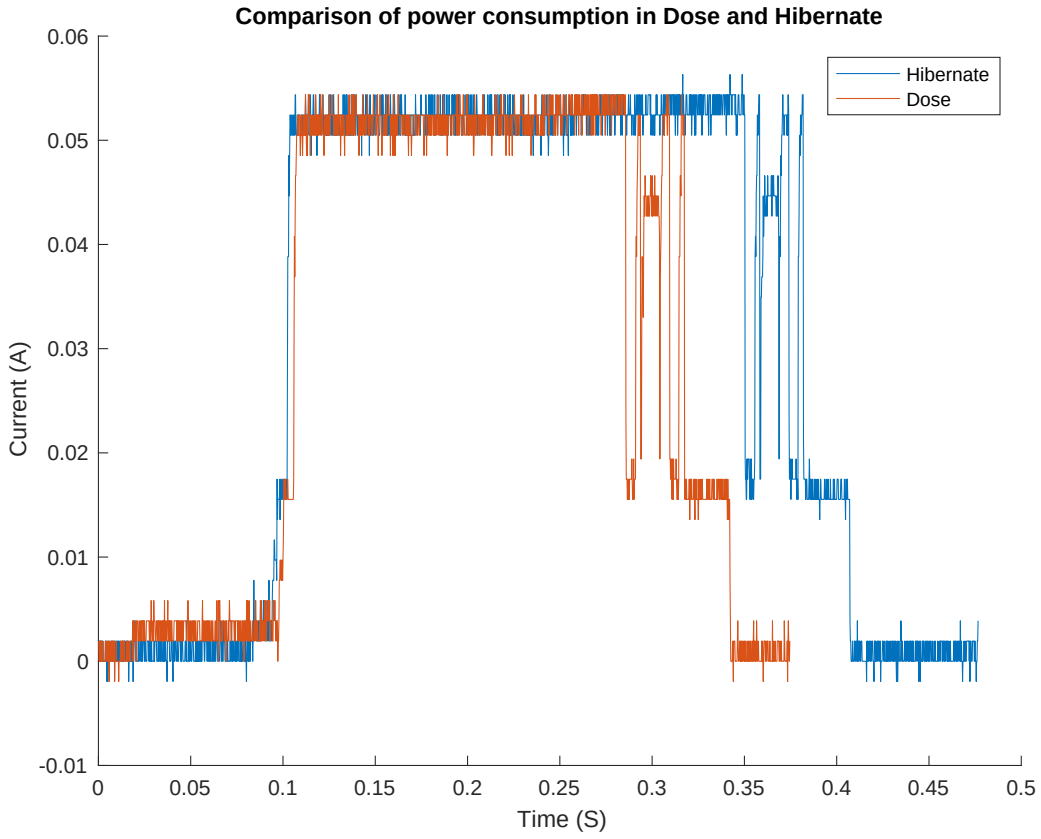


Figure 8.3: Current through XBee when waking up from Hibernate or Dose mode.

Transmission power

In Figure 8.4 3 different power consumption's are shown. The 100 ms sleep setting can immediately be recognised from the transmission less current, which goes down after roughly 100 ms. It can also be seen that the transmission takes considerable extra time then just waking up, 240 ms is spent doing the actual data processing and transmission. Another 120 ms extra is used when encryption is activated, this is spent on encrypting the message before transmission. During the wake up, data processing and transmission the module uses on average 52 mA. This is independent of the encryption or transmission.

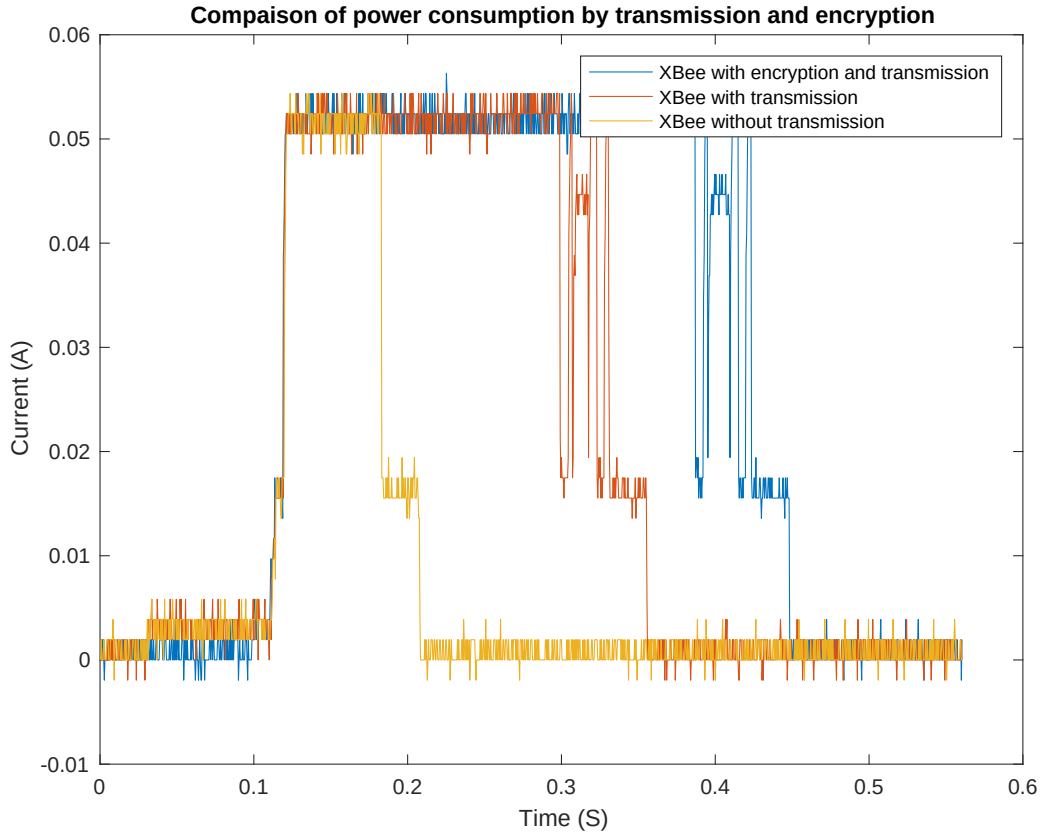


Figure 8.4: Current through the XBee. A clear difference can be seen between only activating, transmission and transmission with encryption

Range

In Appendix A.3 the RSSI for each meter (up to 10 m) can be seen. This data is very confusing so it has been averaged out. The average can be seen in Figure 8.5. It is clear that the RSSI does not change a lot based on the distance. As part of our research some measurements at 19 and 29 meters have been conducted. The data transmission was error free. The results can be found in Appendix A.5. From the previously obtained data we argue that RSSI is not an accurate way to determine the distance to a sensor.

Influence of encryption

The influence of encryption can also be seen in Figure 8.4. The processing time is extended due to the encryption. The total time of processing the data is 320ms. This means that it takes 33% longer to compute.

8.2.3. BLE Nano

In the BLE Nano setup most of the previous test have also been conducted.

Sleep mode

In Figure 8.7 there are some big spikes at 23ms. These spikes are the power consumption during the initiation phase of the connection. After this the module goes into standby modes and uses 4.69mA. The spikes at 4.5 seconds are those of the actual transmission. Although the datasheet mentions a lower power consumption during sleep mode this was not achieved in practice. This can be explained that the BLE Nano was used in central mode and not in peripheral mode, this was ofcourse necessary for our use case to work.

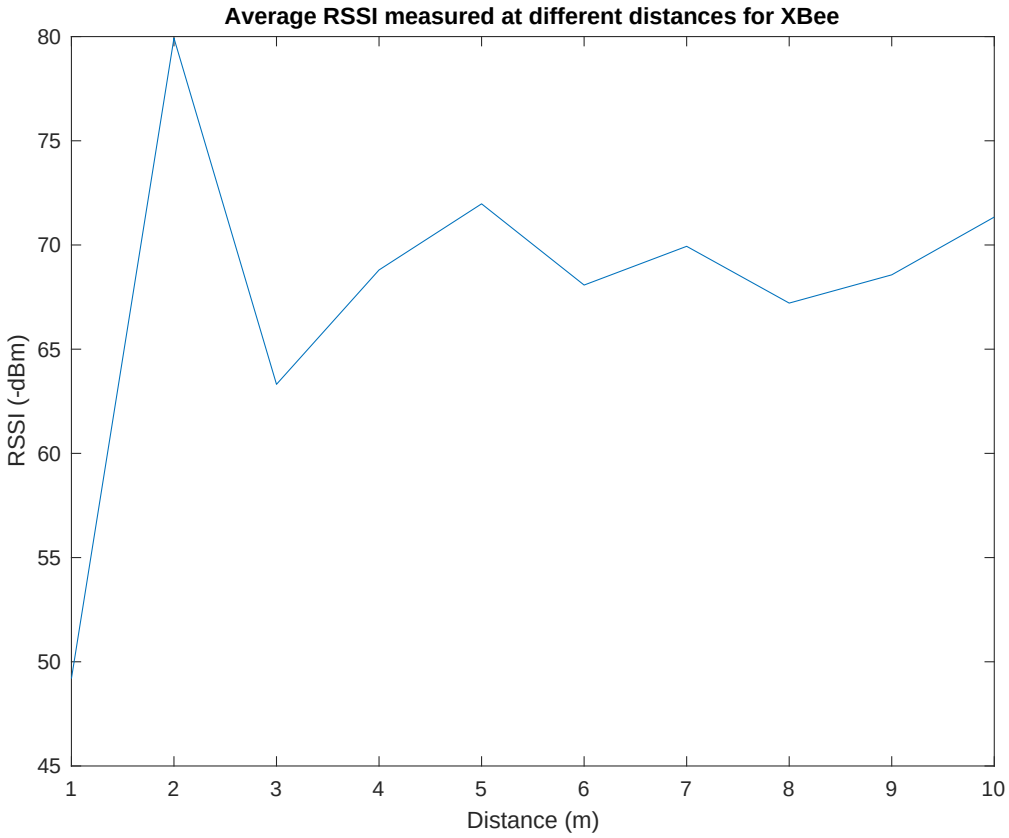


Figure 8.5: RSSI measurement for XBee. Average RSSI measured every meter.

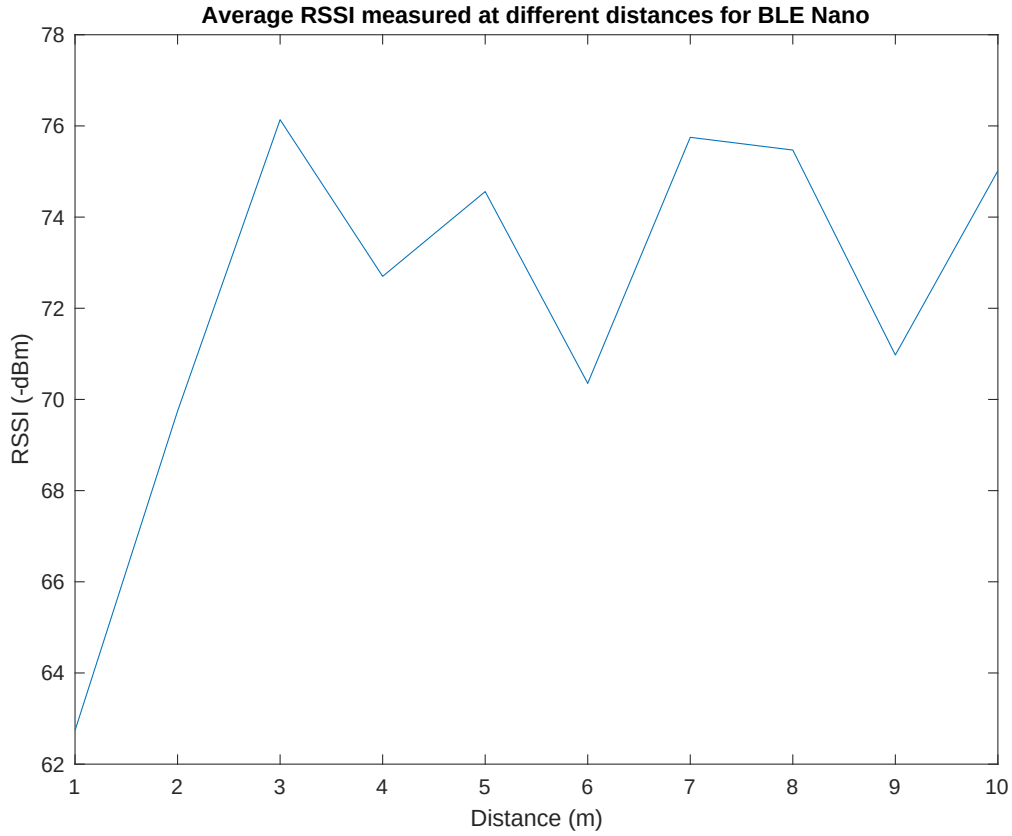


Figure 8.6: RSSI measurement for BLE Nano. Average RSSI measured every meter.

Transmission power

The average power for setting up the connection is 12.1 mA. The datasheet tells us that this has to be in the range of 17 mA peak. The transmission power is a max of 8 mA this is the same as mentioned in the datasheet [33]. From this can be concluded that setting up a connection with BLE Nano is very power consuming.

Range

In Figure 8.6 the RSSI for each meter (up to 10 m) can be seen. It can be concluded that at these distances the received power is mostly stable and does not decrease considerably. Some minor test on longer distances (up to 30 m) were also conducted but only showed minor deterioration. Longer distance test could not be run because of space limitation in our testing area.

Influence of encryption

The BLE Nano system does not allow us to turn off the encryption. This means that that test between encryption consumption and non encryption consumption can not be made.

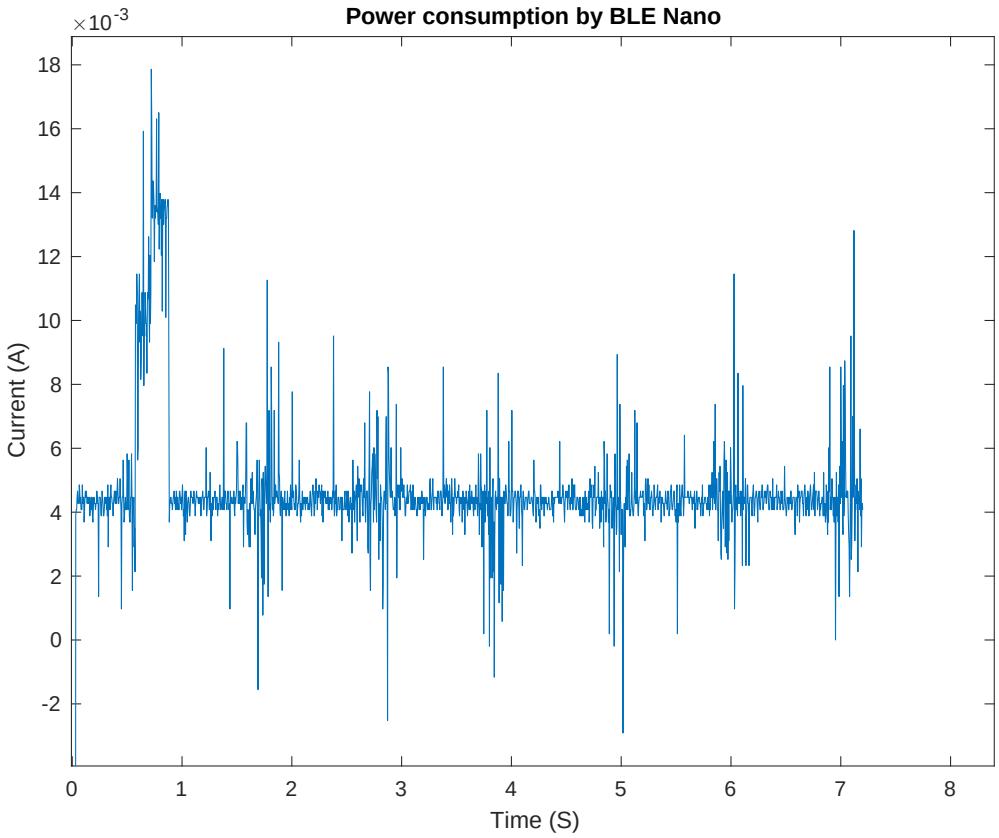


Figure 8.7: Current drawn by BLE Nano. The initial connection peak can be clearly seen in the beginning.

9

Conclusions and recommendations

9.1. Theoretical comparison

In this section an comparison will be made between measured results and theoretical values for every module.

Module	Theoretical sleep mode	Measured sleep mode	Theoretical transmission power	Measured transmission power	Measured range
RFM69HCW	0.1 μA	3.37 mA	45 mA	23 mA	10 Meter
XBee	32 - 100 μA (Vcc = 3.2 V - 3.3 V)	43.2 μA	45 mA	51 mA	30+ Meter
BLE	2.6 μA	4.38 mA	11.8 mA	12.1 mA	30+ Meter

Table 9.1: Comparison of measured consumption versus theoretical (datasheet) consumption in different modes, also shown is the measured range to give an indication of range performance.

When looking at Figure 9.1 the lifetime comparison based on the number of years it can run on a 5000 mAh battery is shown. The results found are not always in line with what the datasheets of the modules say about their power consumption. The lifetime based on datasheet specified consumption can be found in Figure 9.2.

The theoretical model for the RFM69HCW shows a straight line due to the fact that the transmission current is 1.3 million times larger then the sleep consumption. In the real data the difference was only a factor 6.82. This can be explained because the module would not go into sleep mode properly. This can be improved in future research, see Section 9.4.1.

For the XBee the theoretical difference between sleep and active is 1406 - 445 times (Vcc = 3.2 V - 3.3 V), in the real world the difference was 1180 times. This matches the value in the datasheet for the supply voltage in the range 3.2 V - 3.3 V, it is entirely possible that our power supply (a laptop) gave a imperfect 3.3 V. Thus explaining our results.

For the BLE Nano the difference between the theoretical 4538 times and the real 2.76 times is enormous. This difference can be explained because power saving mode is not correctly implemented in central mode.

9.2. Factual comparison

In this section the three systems will be discussed according to the data found in Chapter 8. This is summarised in Table 9.1.

The data visualised in Figure 9.3 gives a clear picture on what module to use. It is clear that the XBee is the best option based on these results. That is because XBee is very power efficient in sleep mode and also because of considerable lower transmission time compared to RFM69HCW as can be seen in 9.2. The fact that the BLE Nano is not that efficient is because of its higher inactive mode current.

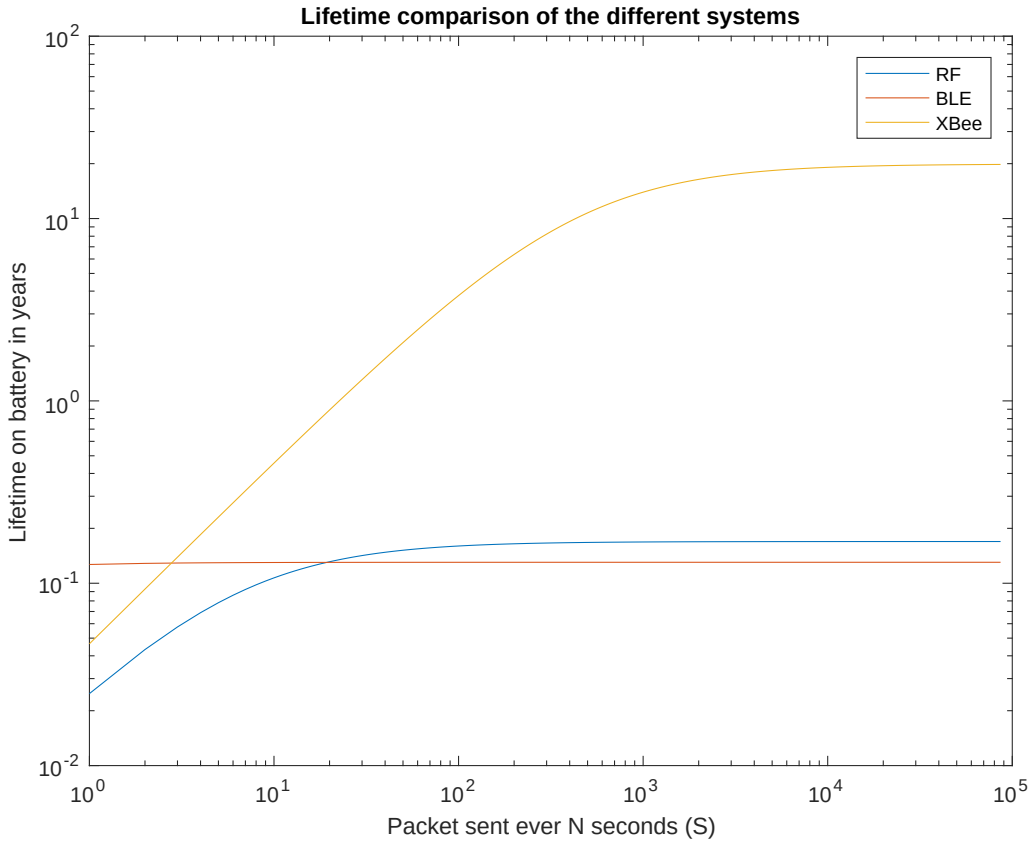


Figure 9.1: Comparison based on measurement data, XBee is the best choice because of its lowest sleep current.

	XBee	BLE Nano	RFM69HCW
Time	240 ms	16 ms	1000 ms

Table 9.2: Measured duration of a single message transmission.

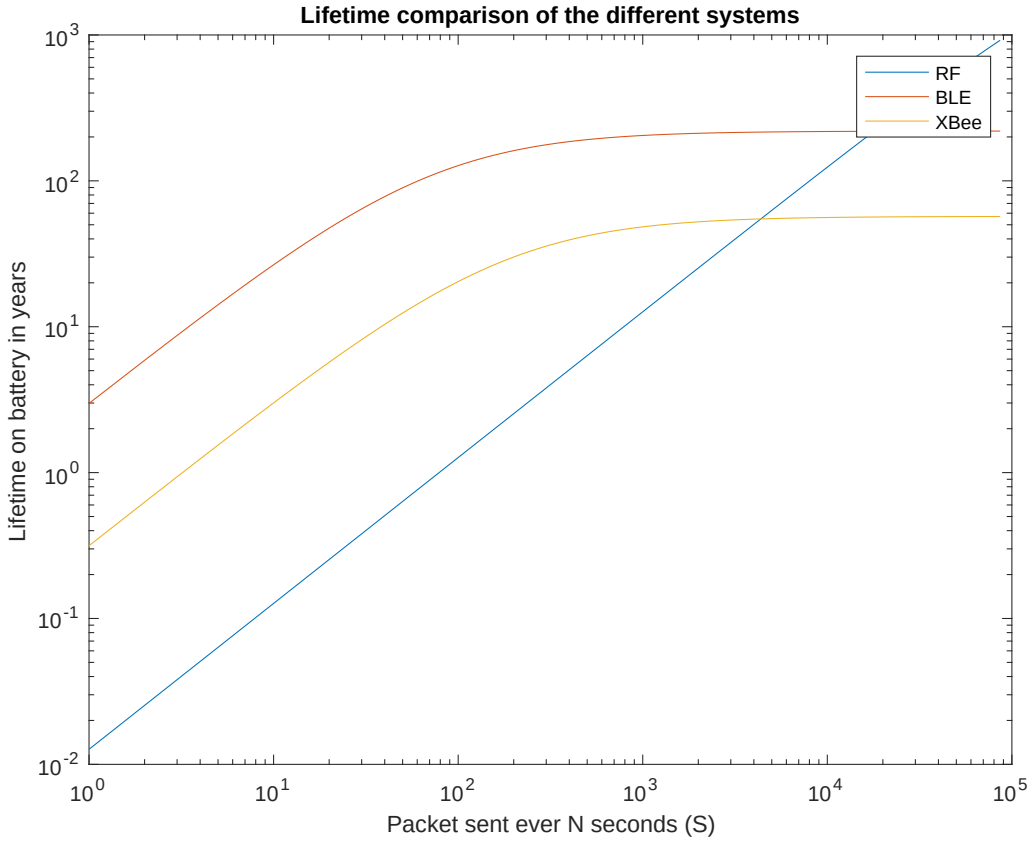


Figure 9.2: Comparison on a theoretical bases, RFM69HCW is almost linear because transmission is 1.3 million times more expensive than sleeping and BLE Nano is the best choice for intervals smaller than once a day.

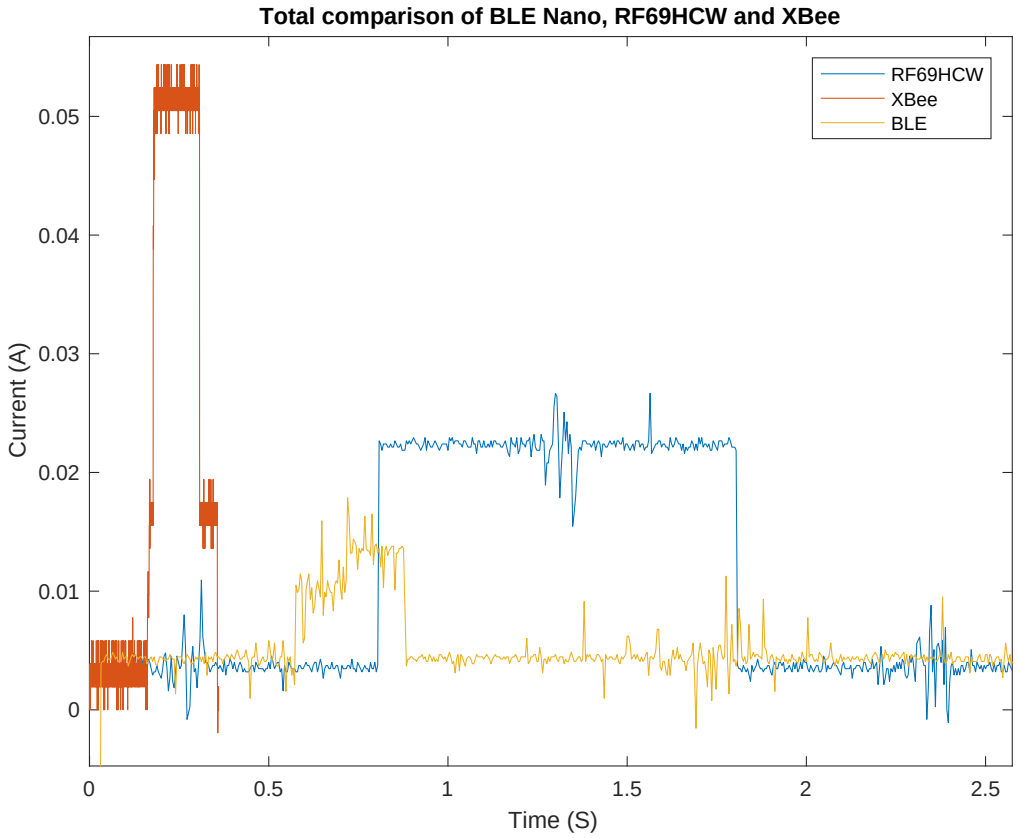


Figure 9.3: Comparison between all three modules. one transmission per module is shown.

9.3. Conclusion

Our conclusion is that if KPN wants to implement a communication system they should choose XBee. Because of its low power consumption and long range it the best choice with the technology currently available.

If further research is conducted and the results match up to the theoretical model. The best choice is BLE Nano in most cases. Except when sending once every five hours and above. For these sensors RFM69HCW should be used.

9.4. Recommendations

During this proof of concept some interesting difficulties were encountered. Some of these were overcome but some of these can be improved in future studies. These improvements will be explained in the following sections.

9.4.1. RFM69HCW

For the 433 MHz there are a lot of improvements to be made. These are discussed below:

- **Sleep mode**
  To get a better picture of the real power consumption the device should be able to go into a deeper sleep mode. This was not accomplished in our study because the library used did not achieve the wanted sleep mode depth. It was attempted to change this behaviour, sadly without success.

- **Antenna**
  In order to increase the range of the device a good antenna should be bought or created. The half dipole we used was not good enough.

- **ATC**
  To decrease the power consumption Automatic Transmit Power control(ATC) could be implemented. This program uses the RSSI and adjusts it transmission current accordingly (Low RSSI, thus the device is close, decrease transmission current).

9.4.2. XBee

For the XBee there are not a lot of improvements because the system preformed to the standards. In further research there could be more focus on reducing the power consumption. This can be done by altering the following settings:

- **Power setting variations**
  Due to time constrains we couldn't experiment with all the different power settings. By altering this the performance could further improve.

- **Sleep setting variations**
  By altering the sleep cycle the device could go into sleep mode even earlier.

- **CCA threshold**
  To improve the capacity of the network a change in the CCA threshold could be implemented. By lowering this value the amount of traffic in the network could be enlarged. This test can only be conducted in a real life scenario. Because this value should be based on the local 2.4GHz signals available, who can disturbe the XBee network.

9.4.3. BLE Nano

For the BLE Nano there are several parts were improvements could be made and different choices could be made to also get better results.

- **Improve central mode power consumption**
  The idle consumption of an BLE Nano in central mode is currently very high this could be improved by putting the device into deeper sleep when it does not need to do anything.

- **Switch central mode and peripheral mode**
  Make the BLE Nano on the sensor side a peripheral mode BLE device, this could potentially decrease idle power usage. However this would incur extra configuration effort because the gateway needs to know before hand how often to query the sensor for data. Also properly distributing effort over all the sensors equally is gonna need a load balancer. An extra disadvantage would be that the sensor cannot initiate communication and thus only strictly periodic transmission is possible.

- **Connection less communication**
  So far no method to communicate without connection/services was found, however this could be researched. Most active power is consumed during establishing the connection and discovering the services if this could be prevented this would reduce the active power consumption considerably.

Bibliography

[1] P. Leach, M. Mealling, and R. Salz, "Rfc 4122: A universally unique identifier (uuid) urn namespace," *RFC*, 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4122.txt

[2] (2017, mar) The i2c arduino documentation. [Online]. Available: https://www.arduino.cc/en/reference/wire

[3] (2017, mar) The spi arduino documentation. [Online]. Available: https://www.arduino.cc/en/reference/SPI

[4] (2017, mar) The serial arduino documentation. [Online]. Available: https://www.arduino.cc/en/reference/serial

[5] (2017, may) Ieee explorer. [Online]. Available: http://ieeexplore.ieee.org/Xplore/home.jsp

[6] F. Eady, "Hands-on zigbee implementing 802.15.4 with microcontrollers," http://www.sciencedirect.com/science/article/pii/B9780123708878500006.

[7] (2017, mar) Zigbee encryption. [Online]. Available: http://knowledge.digi.com/articles/Knowledge_Base_Article/ZigBee-Encryption

[8] (2017, jun) Xbee/xbee-pro oem rf modules. [Online]. Available: https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Manual.pdf

[9] (2017, jun) Low power z-wavetm transceiver with microcontroller. [Online]. Available: http://img.chipfind.ru/pdf/zensys/zw0201.pdf

[10] J. D. Fuller and B. W. Ramsey, "Rogue z-wave controllers: A persistent attack channel," in *2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops)*, oct 2015, pp. 734–741.

[11] S. S. I. Samuel, "A review of connectivity challenges in iot-smart home," in *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, mar 2016, pp. 1–4.

[12] P. Bregman, "Kpn lora, technical service description," *KPN Internal*, 2017.

[13] (2017, jun) Lora layout image. [Online]. Available: http://www.trustpointinnovation.com/blog/2017/01/17/lorawan-security-overview/

[14] (2017, jun) En 300 220 - electromagnetic compatibility and radio spectrum matters (erm); short range devices (srd);. [Online]. Available: http://www.etsi.org/deliver/etsi_en/300200_300299/30022001/02.04.01_40/en_30022001v020401o.pdf

[15] (2017, jun) Mod-wifi-esp8266-dev. [Online]. Available: https://www.olimex.com/Products/IoT/MOD-WIFI-ESP8266-DEV/open-source-hardware

[16] (2017, jun) Mod-wifi-esp8266-dev datasheet. [Online]. Available: https://www.olimex.com/Products/Components/IC/ESP8266EX/resources/0a-esp8266_datasheet_en_v4.4.pdf

[17] J. Norair, "Introduction to dash7 technologies," *rfidjournal*, mar 2009.

[18] (2017, jun) Network typologies image. [Online]. Available: http://www.ni.com/white-paper/10789/en/

[19] H. Seo and H. Kim, "Network and data link layer security for dash7," *researchgate*, sept 2012.

[20] J. Donovan. (2017, mar) Bluetooth goes ultra-low-power. [Online]. Available: https://www.digikey.com/en/articles/techzone/2011/dec/bluetooth-goes-ultra-low-power

[21] (2017, may) Ble core requirements. [Online]. Available: https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=229737

[22] (2017, may) Datasheet rfm69hcw. [Online]. Available: https://cdn.sparkfun.com/datasheets/Wireless/General/RFM69HCW-V1.1.pdf

[23] (2017, jun) Rfm69hcw arduino library. [Online]. Available: https://github.com/LowPowerLab/RFM69

[24] (2017, jun) Sparkfun logic level converter - bi-directional. [Online]. Available: https://www.sparkfun.com/products/12009

[25] (2017, jun) Atmega328/p datasheet. [Online]. Available: http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf

[26] (2017, jun) Rfm69 nodejs library. [Online]. Available: https://www.npmjs.com/package/rfm69

[27] (2017, jun) Rpio - nodejs library for low level interfaces. [Online]. Available: https://www.npmjs.com/package/rpio

[28] (2017, jun) Serialport - nodejs library for serial connections. [Online]. Available: https://www.npmjs.com/package/serialport

[29] (2017, jun) Raspbian - debian based linux distribution for raspberry pi. [Online]. Available: https://www.raspberrypi.org/downloads/raspbian/

[30] (2017, jun) Etcher - image burner for mac os x. [Online]. Available: https://etcher.io/

[31] (2017, jun) Sakis3g - 3g library. [Online]. Available: https://github.com/Trixarian/sakis3g-source

[32] (2017, jun) Wvdial - ppp dialer. [Online]. Available: https://github.com/wlach/wvdial

[33] (2017, jun) nrf51822 datasheet. [Online]. Available: https://cdn.sparkfun.com/datasheets/Wireless/Bluetooth/nRF51822_PS%20v3.1.pdf

A

Appendices

A.1. Appendix 1

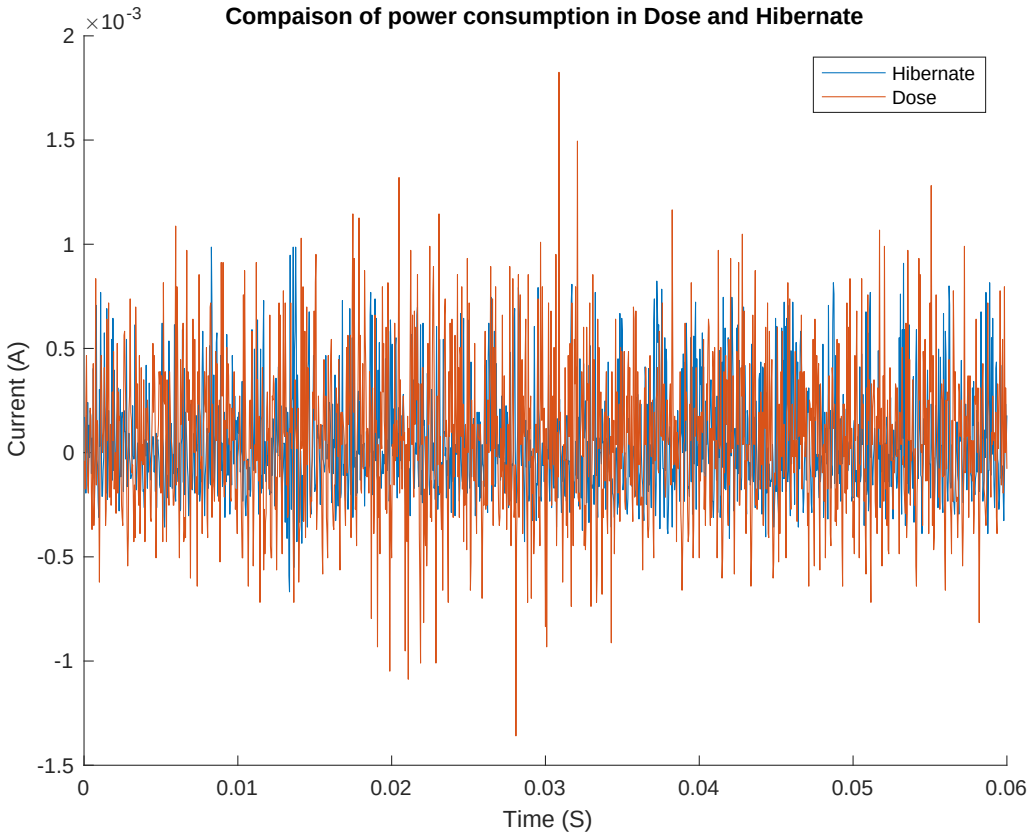


Figure A.1: Dose vs Hibernate comparison in sleep mode. Average for Dose is 73.01 μA. Average for Hibernate is 43.2 μA.

A.2. Appendix 2

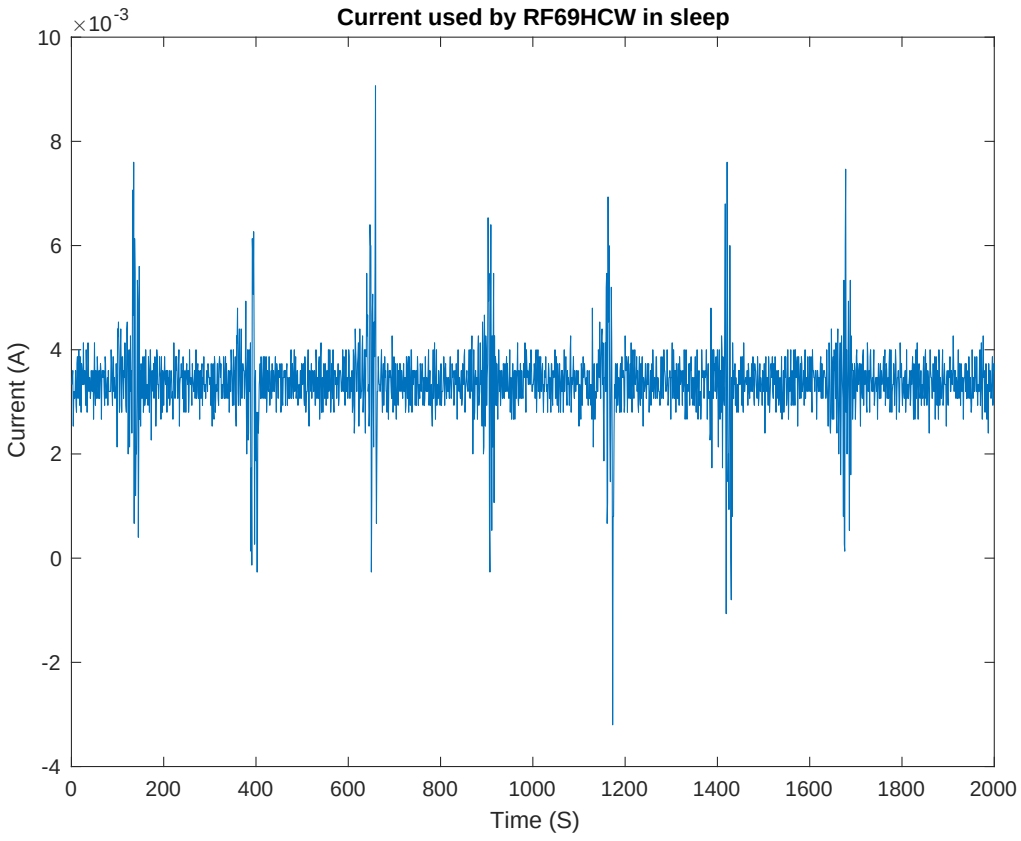


Figure A.2: Current drawn by RF69HCW in sleep mode.

A.3. Appendix 3

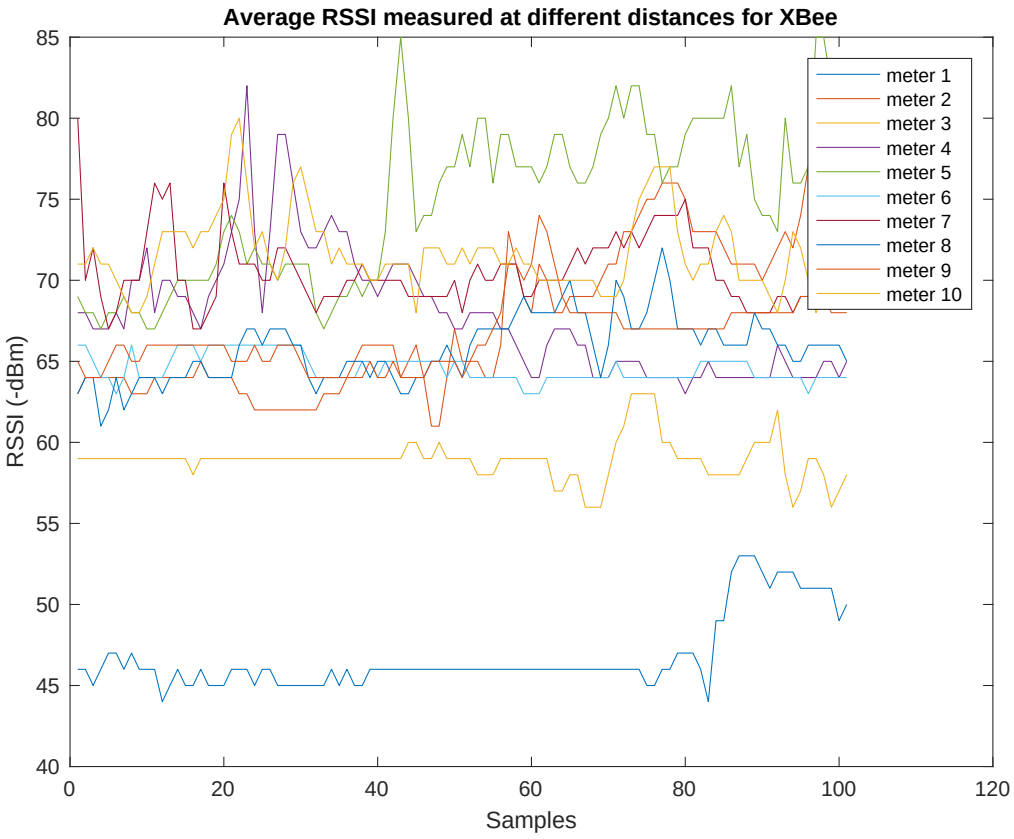


Figure A.3: Different XBee RSSI distance measurements.

A.4. Appendix 4

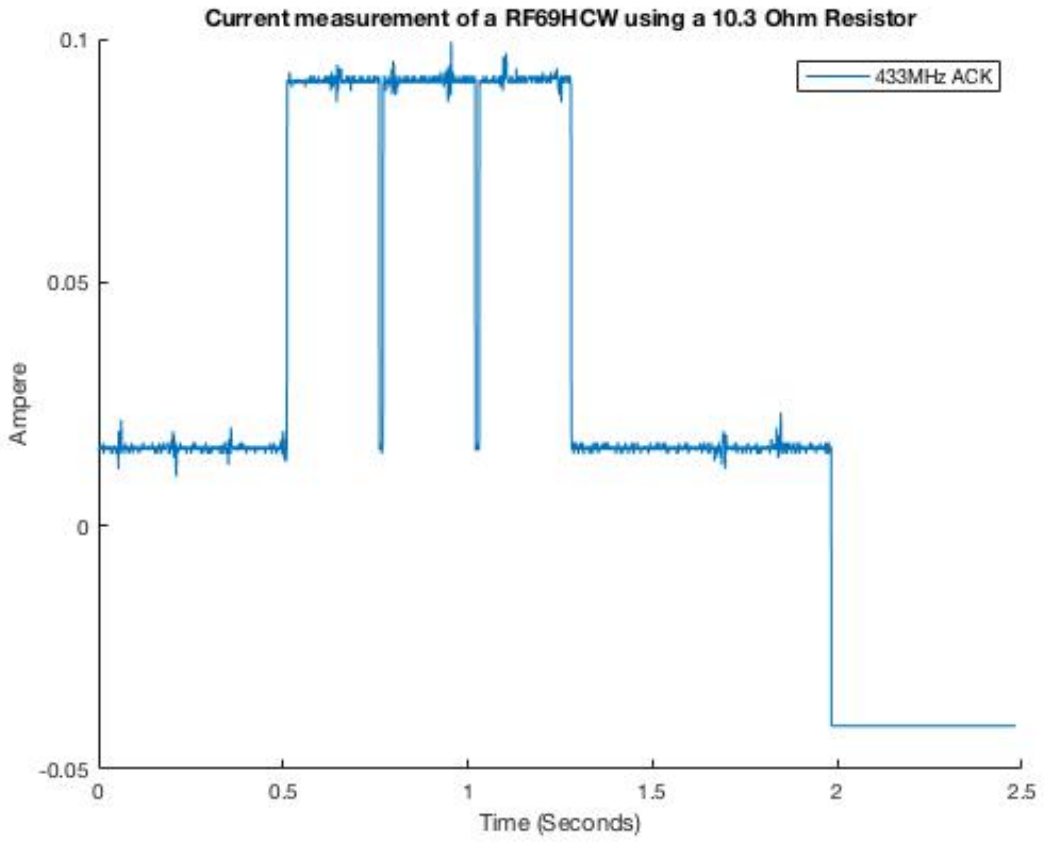


Figure A.4: Current drawn by RF69HCW - With ACK.

A.5. Appendix 5

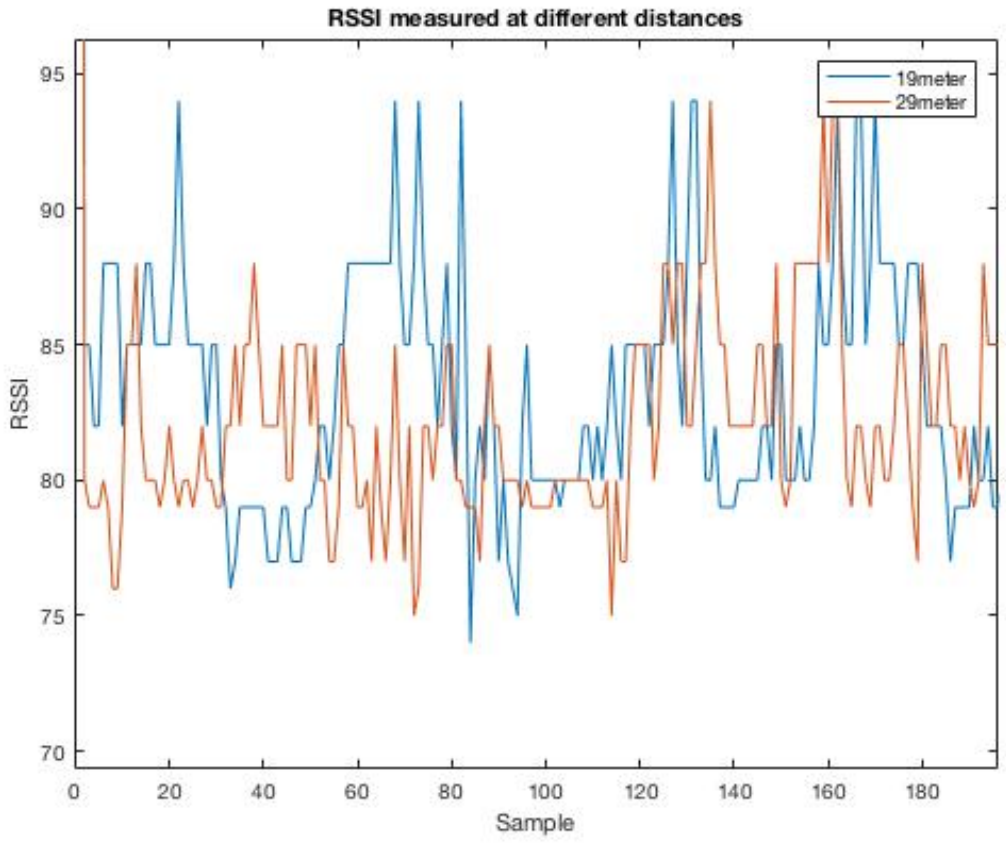


Figure A.5: RSSI at 19 and 29 meters.