STELLENBOSCH UNIVERSITY

Z-wave Open Source Home Automation Network Simulated In SimPy

Author:
Jason la Cock

Supervisor:
Mr Arno Barnard

17595304

Report submitted in partial fulfilment of the requirements of the module Project (E) 448 for the degree Baccalaureus in Engineering in the Department of Electrical and Electronic Engineering at the University of Stellenbosch

November 1, 2016

Declaration of Authorship

I, the undersigned, hereby declare that the work contained in this report is my own original work unless indicated otherwise.

Signed:

_____

Date:

_____

STELLENBOSCH UNIVERSITY

Abstract

Faculty of Electrical and Electronic Engineering

Z-wave Open Source Home Automation Network Simulated In SimPy

by Jason la Cock

Home automation protocols serve as the language through which home devices communicate. A multitude of factors in home environments attenuate the quality of the achieved communication link. The purpose of this study was to investigate the networking performance of a specific home automation protocol. Subsequently a simulation model which accurately simulates observed network performance for the identified protocol was constructed.

First a comparison was drawn between selected home automation protocols. Network performance considerations addressed were throughput, latency and range. Z-wave was selected as the best home automation protocol for deployment in homes. Z-wave creates an interoperable, source-routed network with simple, affordable implementations available. The report additionally investigated the specific details of standards governing z-wave operation. Two z-wave networks,facilitating networking analysis, were proposed.

A simulation model of a z-wave network was constructed utilizing the SimPy simulation library. An accurate model simulating the temporal nature of packet transmission within z-wave networks was created. The report documents the construction of the proposed physical networks, validating the theory concerning the z-wave protocol. Finally essential network parameters were observed and incorporated into the simulation model, optimizing simulation accuracy.

STELLENBOSCH UNIVERSITY

Abstract

Faculty of Electrical and Electronic Engineering

Z-wave Open Source Home Automation Network Simulated In SimPy

by Jason la Cock

Tuis automatisering protokolle dien as die taal waardeur toestelle met mekaar kommunikeer. A hele aantal faktore binne in die tuis omgewing dra by tot die verswakking in die kwaliteit van ń kommunikasie skakel. Die doel van hierdie studie was om ondersoek in te stel rondom die optrede van n geídentifiseerde tuis automatisering protokol. Daarna was n simulasiemodel opgestel wat die netwerk prestasie van die geídentifiseerde protokol namaak.

Eerstens is n vergelyking getrek tussen sekere geídentifiseerde tuis automatisering protokolle. Netwerk prestasie oorwegings wat geádresseer was sluit in deurset, vertraaging en afstand. Z-wave was gekies as die beste tuis automatisering protokol vir toepassing aan tuis omgewings. Z-wave skep n tussenwerkbare, bron aangestuurde netwerk met eenvoudig en bekosbare oplossings beskikbaar van n groot hoeveelheid verskaffers. Die verslag het verder navorsing ingestel rondom die standaarde waarvolgens z-wave funksioneer. Daarna is twee z-wave netwerke, wat die analise van netwerk bewerkstellig, voorgestel.

n Simulasie model van ń z-wave netwerk is opgestel deur gebruik te maak van die SimPy simulasie biblioteek. ń Akkurate model wat die temporale natuur van boodskap transmissie binne z-wave netwerke namaak was geskep. Dié verslag dokumenteer die konstruksie van die voorgestelde fisiese netwerke. Die saamgestelde netwerke dien om die teorie rondom die z-wave protokolle te bevestig. Laasten is essentiele netwerk parameters waargeneem en daarna in die simusiemodel ingevoer. Dus word akkurate simulasies bewerkstellig.

Acknowledgements

I would like to sincerely thank my study leader, Mr Arno Barnard, for his invaluable guidance and thoughtful insight throughout the project. Additionally for the calm leadership he displayed following issues with the implementation of the project.

Further I would like to thank Mr Johan Arendse for his assistance in soldering the ZM5304 modules.

Contents

List of Figures

List of Tables

List of Abbreviations

JVM	**J**ava **V**irtual **M**achine
IP	**I**nternet **P**rotocol
iOS	**A**pple **O**perating **S**ystem
WPAN	**W**ireless **P**ersonal **A**rea **N**etwork
UPB	**U**niversal **P**owerline **B**us
UART	**U**niversal **A**synchronous **R**eceiver **T**ransmitter
USB	**U**niversal **S**erial **B**us
GPIO	**G**eneral **P**urpose **I**nput **O**utput
SSH	**S**ecure **S**hell
SD	**S**ecure **D**igital
HD	**H**igh **D**efinition
DNS	**D**omain **N**ame **S**erver
OSGi	**O**pen **S**ervice **G**ateway **I**nitiative
GPLv3	**G**eneral **P**ublic **L**icense **V**ersion 3
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
MQTT	**MQ** **T**elemetry **T**ransport
IFTTT	**I**f **T**his **T**hen **T**hat
UI	**U**ser **I**nterface
OS	**O**perating **S**ystem
SoC	**S**ystem **o**n **C**hip
SDK	**S**oftware **D**evelopment **K**it
API	**A**pplication **P**rogram **I**nterface
IoT	**I**nternet **o**f **T**hings
SUC	**S**tatic **U**date **C**ontroller
ART	**A**pplication **R**esponse **T**ime
FSK	**F**requency **S**hift **K**eying
RF	**R**adio **F**requency
DSSS	**D**irect **S**equence **S**pread **S**pectrum
MHz	**M**ega **H**ertz
LOS	**L**ine **O**f **S**ight
dB	**D**ecibel
Tx	**T**ransmitter
Rx	**R**eceiver
ISM	**I**ndustrial **S**cientific and **M**edical
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers

Physical Constants

Speed of Light $\quad c_0 = 2.997\,924\,58 \times 10^8\,\mathrm{m\,s^{-1}}$ (exact)

List of Symbols

f	Frequency	Hz

Chapter 1

Introduction

1.1 Background

The Internet of Things (IoT) is a network connecting devices to the internet to facilitate communication. Home Automation is one sphere of the IoT, striving towards the creation of a "connected" home, through integration of smart IoT devices. The installation costs involved with the creation of the "connected" home are, however, extremely expensive.

To meet the growing demand, a large number of home automation protocols are available. Home automation protocols serve as the platform through which home connectivity is achieved. Each home automation protocol adopts its own unique set of rules and standards through which communication is achieved. The performance metrics which these protocols strive to achieve include: the interoperability of devices, the maximum range of communication as well as the maximum number of devices which can be included into a home automation network.

Every home environment, however, has a unique topology. Additionally, the connectivity needs of every individual, addressed by the designed home automation solution, are unique. Finally, there exists a wide range of unknown physical factors within a typical home, such as size and interference, all of which affect the quality of communication between connected devices. A typical home owner, however, will not be able to quantify the severity of the impact these factors prior to installation of the home automation network. Accordingly it would be desirable to create a simulation model allowing the user to model the performance of the selected home automation protocol within a home. This will therefore allow the user to evaluate the viability of the proposed home automation installation.

1.2 Objectives of this study

Considering the costs involved, as well as the uniqueness of the needs addressed by the designed home automation network, it is essential to select the correct home automation protocol. This report identifies and compares leading home automation protocols. The objective of the analysis is to ultimately select a single protocol which fully addresses the needs of home owners. Additionally, however, the selected protocol must also offer open source libraries through which network analysis is achieved.

The report documents the construction of a simulation model which accurately models the performance of the selected home automation protocol. The simulation model is constructed utilizing the SimPy simulation library. The objective is to generate a simulation model which

accurately models the quality of the communication link established by the selected protocol. To achieve this, the simulation must yield accurate approximations of critical network performance metrics such as the link margin, throughput and latency.

A physical network, implementing the selected home automation protocol, must be constructed. The objective of the physical network is to firstly verify the theoretical performance of the selected protocol. Thereafter the physical network is utilized to quantify the affect of signal attenuation within home environments on the quality of the communication link achieved. Finally the performance of the physical network is analysed to allow for optimization of the generated simulation model. The objective of the optimization is to ensure that the generated simulation model presents a practical network analysis tool.

A final objective was introduced following implementation issues experienced with the selected home automation protocol. Subsequently, the report aims to draw a comparison between the performance of standalone system-on-chip (SoC) modules issued by vendors and proprietary home automation products developed by product vendors. The objective is to inspect the viability of the low-cost home automation solution presented by the SoC modules.

Chapter 2

Literature Study

2.1 Home Automation Network Protocols

2.1.1 Candidate Home Automation Network Protocols Considered

There exists a large variety of different network protocols within the field of home automation. These protocols exist to provide a set of rules governing communication between the nodes contained within a network. In this section we will draw a comparison between some of the more popular home automation protocols available.

Zigbee

Zigbee is an Institute of Electrical and Electronics Engineers (IEEE) 802.15.4-based specification for a suite of high-level communication protocols used to create wireless personal area networks (WPAN) with small, low-power digital radios. Zigbee was disregarded as a possible network implementation due to the fact that Zigbee has a reputation for poor interoperability between devices from different manufacturers [1]. This is an essential factor for a standard home automation set-up where one would typically want to control a variety of devices such as lights, blinds, televisions etc. Additionally the frequency range at which Zigbee devices operate would cause significant interference issues due to shared bandwidth with WIFI and Bluetooth devices which typically form an integral part of modern homes. A Zigbee implementation would be considered for a large scale solution where devices from a single manufacturer are needed e.g. factories, hospitals etc.

Universal Powerline Bus

UPB (Universal Powerline Bus) is a reliable, robust wired peer-to-peer networking implementation which yields greatly improved performance in comparison to the original X10 networking solutions [2]. The viability of UPB as a home automation implementation is the fact that UPB utilizes existing connections in the home, hence further reducing the cost of network implementation. Modern technological developments however emphasise the importance of selecting a wireless implementation when considering a home automation installation. Additionally there is a limited number of network devices available for UPB solutions in comparison with some of the other emerging home automation networking standards. UPB was disregarded as a possible network standard solution in order to rather focus on a wireless networking solution.

Insteon

Insteon is a networking standard which creates a dual-band, peer-to-peer wireless mesh network for operation in the sub GHz frequency band **insteon** Additionally Insteon allows for the extension of existing X10 wired network connections. This facilitates network operation even when radio frequency communication is impaired. Being a peer-to-peer network there is no need for a central controller while additionally enabling simulcasting of messages which greatly improves the speed and reliability of message transfer. Further network devices in an Insteon network are automatically added to the network upon receiving power, allowing simplifying network installation and improving network maintainability.

Z-wave

The final network standard considered was z-wave. Z-wave is a wireless networking standard which makes use of message routing from a central controller to create a dual mesh network. Z-wave operates within a sub 1 GHz frequency band, providing immunity from interference with WIFI and Bluetooth devices installed in the home. Z-wave is expanding rapidly worldwide and hence there are a large number of supported devices available for all sorts of functions within the home. Critically however in comparison to the Zigbee networking standard is the fact that interoperability between Z-wave devices is guaranteed, improving feasibility as a home automation networking standard.

2.2 Advantages and Disadvantages of Candidate Network Protocols

The advantages and disadvantages of the candidate network protocols are discussed below.

2.2.1 Insteon

Advantages

Insteon is reliable due to the dual mesh configuration implemented. Insteon devices are peers, making it simple to configure large networks (100+ devices) as peers simulcast messages. Simulcasting additionally ensures communication is instantly responsive and fast. Further Insteon devices are automatically added to the network at power-on and are immune to interference in 2.4 GHz WIFI frequency band. The user has the capability to expand an established X10 wired network, hence offering an affordable wireless. Finally, each device has a unique network ID , hence providing immunity from eavesdropping. (AES-256 networking security for user protection)

Disadvantages

Insteon users are limited to a single vendor with a limited range of products available. Additionally these products are slightly more expensive than z-wave technology. Benefit of simulcasting is negligible in a home. Designed for simple message transfer, hence may struggle with high data load

2.2.2 Zigbee

Advantages

Zigbee is an established product which has been on the market for more than 10 years.

Disadvantages

Zigbee is vulnerable to interference from WIFI communications in the 2.4 GHz frequency band. Additionally this high frequency implies a short operating range (<10m). Finally interoperability issues between products from different suppliers is common.

2.2.3 Z-wave

Advantages

The z-wave protocol over internet protocol (IP) allows seamless communication of mesh, home and internet. Further interoperability of devices from all suppliers is guaranteed. Sub 1 GHz operating frequency implies a larger operating range (>30 m) as well as being highly penetrating, hence ideal for communication through walls in homes. Further the selected operating frequency implies immunity from interference in WIFI frequency band. There are a large number of z-wave compatible devices available. Additionally z-wave products are supported by many open source home automation software platforms. Z-wave implements a source-routed network. This promotes accelerated communication as routing is only performed once at the source. A source-routed network offers a margin of flexibility as the routing can be easily adapted by the source. Finally network operation is not limited by the chosen network topology as the source has access the ability to communicate with all nodes. These factors all contribute the interoperability of the designed network. Each device has a unique network ID, hence providing immunity from eavesdropping. (AES-256 networking security for user protection) Finally, z-wave is an established product which has been on the market for more than 10 years.

Disadvantages

Z-wave is proprietary to a single supplier, Sigma Designs. Z-wave devices need to be manually included in the network. Z-wave is designed for simple message transfer and may struggle with high data loads. Z-wave is a source-routed network. This implies enlarged packet overhead as routing information is stored in the packet, hence decreasing the achieved throughput.

2.2.4 UPB

Advantages

UPB is a peer-to-peer network, hence no network controller needed. UPB transmits over very long distances (>1km). UPB works over existing wires in a home, therefore does not require additional wiring. Finally, UPB exhibits accelerated performance and reliability in comparison with X10 wired connections.

Disadvantages

UPB offers few supported products and provides no encryption.

2.3 Comparison of Candidate Network Protocols

A comparison identifying the technical attributes of the candidate network protocols are illustrated in table 2.1.

	Insteon	Zigbee	Z-wave	UPB
Media	Powerline	RF	RF	RF and Powerline
Topology	Complete mesh	Partial mesh	Partial mesh	
Frequency	915 MHz	2.4 GHz	868 MHz	
Data Rate (bps)	38.8 k	250 k	9.6 k	
Global Clock	Powerline zero-crossings	Optional Beaconing	None	
Message Propagation	Simulcasting repeaters	Routing tables	Routing tables	Superimposed UPB pulse
Data Modulation	Powerline: BPSK RF: FSK	DSSS	FSK	Pulse Position Signalling
Message Length	15-33 bytes	128 bytes	Variable	0-18 bytes
Maximum Devices	400		232	
Interoperability	Yes	No	Yes	Yes
Device Types	Peers	Controllers, routers and slaves	Controllers, routers and slaves	
Costs (lowest cost alternatives)	Controller: $89 Node: $30		Controller: $45 Node: $20 Module: $14	

TABLE 2.1: Home Automation Networking Protocols Comparison

2.4 Z-wave vs Insteon

Ultimately the best two possible networking implementations for home automation were Insteon and Z-wave. If one compares these two solutions there is little difference with regards to the parameters which concern home automation:

2.4.1 Critical Comparison of the Z-wave and Insteon Protocols

Insteon enables significantly accelerated communication speeds, however for the application of home automation one will not be able to notice the difference in communication speeds when

for example switching on a light as Z-wave is certainly fast enough for this purpose without noticeable delay. Both standards share identical security and encryption methods which have proved to be reliable and secure thus far. Both standards operate in the sub 1 GHz frequency band, and are therefore immune to interference from WIFI and Bluetooth which operate within the 2.4 GHz frequency band.

In terms of cost both standards can be considered to be a low cost home automation implementation, however here one may note that Z-wave solutions can be achieved at a slightly lower price. This is due to the large quantity of Z-wave devices being made available on the market, hence low cost implementations are available, while Insteon is sold by a single vendor with no competition to speak of. Additionally Z-wave offers a range of low-cost modules for custom home automation development. Insteon has the capability to connect a higher number of devices into the network. However, since for home automation applications the maximum number of devices that can be included into a z-wave network is more than sufficient, this fact is negligible.

Insteon devices are peer-to-peer, hence each device can serve as a controller or standard node. This facilitates simulcasting of messages to create a robust, reliable network. For home automation applications however this fact is negligible to the limited number of devices which can be installed into a standard home. One may further state that Insteon is a far more applicable networking implementation for large scale industry applications where hundreds of devices are connected to form a large, complex, mesh network. Insteon devices are automatically added to the network upon receiving power while Z-wave end devices need to be manually configured and added to the network. However considering the fact that the majority of devices to be added to the home automation network (e.g. televisions, lights, curtains, heaters) are generally static or fixed, the improved flexibility of Insteon networks will not make a significant difference in the home.

Insteon does allow for one to build onto existing X10 wired connections, however performance achieved with X10 communication is unreliable and slow in comparison. Z-wave provides a margin of flexibility with the design of the solution due to large number of devices available on the market which additionally can be extended with 5th generation Z-wave modules installed into micro-controllers such as Raspberry Pi's. This allows for flexible, always-on design of home automation networks.

The Z-wave networking standard was chosen for implementing the home automation network. The metrics considered and comparisons drawn in the previous section revealed that both Insteon and Z-wave would be applicable home automation networking solutions. However the networking solution to be designed is a small-scale home implementation in order to optimize network performance and topology. Many of the large-scale advantages of Insteon are not applicable to this project. Additionally, Z-wave provides far greater flexibility due to the number of devices available on the market from various suppliers. Further Z-wave has a large community of support while additionally it is supported by most open source software platforms for performing network analysis.

2.5 Network Analysis Indicators

When modelling a z-wave network, there are several critical network attributes to be considered. These include latency, throughput and the maximum range of reliable communication.

2.5.1 Latency

Latency is considered as the time delay in any cause-effect relationship [3]. For application towards networking, latency is the time delay observed in completing a network task such as the routing or queueing of transmitted packets.

Transceiver Delay

The transmit-after-receive delay and receive-after-transmit delays indicate the physical lag introduced by a transceiver between identifying a request and responding to an acknowledged request. For a transmitter this is the time interval from when the energy of the last received bit is removed until the transmitter keys the first bit of the response. The maximum allowed transceiver delay for z-wave networks is 1 ms [4].

Propagation Delay

The propagation delay is the time delay observed as the transmitted packet propagates across the physical distance separating nodes. Z-wave, however, facilitates short-range packet transmissions within a home, where the transmitted radio frequency (RF) signals travel at the speed of light in free space. Accordingly, the propagation delay is considered to be negligible. Therefore once a symbol is transmitted, assume it is immediately available for detection at the receiver.

Transmission Delay

Transmission delay refers to the time taken by the RF transmitter to transmit all the bits contained within the packet. The transmission delay can therefore considered to be a delay introduced by the limited symbol rate of the transmitter. Additionally one will note that the transmission delay introduced is directly proportional to the number of bits contained within the transmitted packet. For the purpose of this report, the implemented z-wave network communicates with a symbol rate of 9600 bits/s.

Queueing Delay

The time interval during which a transmission request is queued at the primary controller before being executed. This time delay is induced by limitation that only a single packet can be transmitted across a communication channel simultaneously

2.5.2 Throughput

For the purpose of this report, the throughput is calculated as the number of information bits delivered within a network environment over a certain period of time. Here information bits only include useful bits contained within the data payload of a packet and excludes packet

overhead and retransmitted packets. Throughput will provide an indication of the robustness of the network when facing challenging transmission environments or large distances between nodes. Throughput also provides an indication of the responsiveness of the network to requests received.

2.5.3 Range of Transmission

The maximum distance between nodes allowing for successful packet transmission provides an indication of the full operational capabilities of the network.

2.6 Signal Attenuation

Signal attenuation is considered as the reduction of the signal strength due to propagation in a medium. To successfully construct a model simulating z-wave network performance, it is essential to quantify the signal attenuation which occurs when transmitting a packet between two nodes. The magnitude of the observed signal attenuation will be utilized to determine whether a transmitted signal will successfully reach its intended destination.

2.6.1 Attenuation Assumptions

For the purpose of this report the following simplifying assumptions are made with regard to the attenuation of signals:

1. Environmental weather conditions do not influence the signal transmission or cause signal attenuation.
2. The 868 MHz z-wave signal has a high penetration ability, hence transmitted signals have the ability to penetrate obstacles, such as walls, within a home. This implies that the transmitted signal will follow the direct pathway between two nodes, given that the signal has sufficient power to penetrate the wall and be detected by the receiver.
3. Considering the high penetration ability of the transmitted z-wave signal, a significant portion of the reflected signal's energy is lost when the signal collides with and penetrates an obstacle. Accordingly, assume that no packets propagating through signal reflection will successfully reach the receiver.
4. The antenna gain of the sending node is independent of the orientation of the antenna.
5. The interference caused by multipath signal propagation is negligible. This assumption is validated by the following observations: RF signals travel at the speed of light, hence propagation delay is negligible. Accordingly, multiple occurrences of a single transmitted packet will give only a small jitter delay per bit.

2.6.2 Link Margin

The link margin indicates the difference between the power of the received signal (also known as the link budget) and the minimum signal power that the receiving node would be able to successfully detect [5]. For indoor communication, the link budget is calculated as the sum of transmitter and antenna gains minus the sum of the path loss due to transmission through free

air as well as attenuation occurring due to collisions of packets with obstacles within a home. Therefore:

$$link\_budget = (TX\_power + antenna\_gain) - (path\_loss + obstacle\_attenuation) \quad (2.1)$$

$$link\_margin = link\_budget - RX\_sensitivity \quad (2.2)$$

The link margin, measured in dBm, therefore indicates the safety margin afforded to the sending node that the transmitted packet will successfully be detected by a receiving node. For example a link margin of 10 dBm indicates a margin of 10 mW while a link margin of 20 dBm indicates a margin of 100 mW.

Path Loss

To determine the signal attenuation which occurs in air as the transmitted RF waves move away from the transmitter, we shall consider the Friis equation for path loss (measured in decibels (dB)):

$$path\_loss = 20log(\frac{4\pi r}{\lambda}) \quad (2.3)$$

$$\lambda = wavelength = \frac{c}{f} \quad (2.4)$$

$$c = 3 \times 10^8 m/s \quad (2.5)$$

Where r is the distance between the transmitter and the receiver in meter and f is the frequency at which communication occurs. Within South Africa allocated communication frequency is 868 MHz.

Note that the observed path loss is inversely proportional to the wavelength of the transmitted signal. Accordingly, from (2.4), (2.5) and the frequencies listed in table 2.1, the wavelength of a z-wave signal is calculated to be 0.346 m. In similar fashion the wavelength of a signal operating in the 2.4 GHz frequency band is calculated to be 0.125 m. Thus sub 1 GHz frequency signals will have significantly less attenuation than 2.4 GHz signal, over the same distance, emphasizing the benefit of operating a lower frequencies.

2.7 Z-Wave Open Source Software Libraries

The Z-Wave networking standard is proprietary to a single supplier, namely Sigma designs. Hence for development purposes in order to incorporate Z-Wave functionality into a simple home network one needs to acquire expensive Z-Wave SDK and annual licensing. An open source SDK, as well as the firmware embedded on z-wave devices is however not available. Hence one needs to acquire proprietary z-wave hardware which have a form of controller firmware embedded on the device. This installed firmware would in turn allow the user to enable communication through the serial API. This, facilitates the development and use of open source software libraries, allowing the user to create an interface through which communication can be achieved.

2.7.1 OpenZWave

OpenZWave is an open source software library developed in C++ which supports interfacing with Z-Wave controllers and communication with z-wave devices [6]. Many open source software development tools are built on top of OpenZWave, serving as the foundation for communication with the z-wave serial API. The following open-source utilities were utilized in the completion of this project:

Open-Zwave-Control-Panel

The open-zwave-control-panel (OZWCP), is an open source development tool providing an application which allows one to manage and monitor nodes in the Z-Wave network [7]. The OZWCP application provides a way in which developers can monitor physical low-level communication commands issued to the z-wave serial API. This allows for the monitoring of essential lower-level network performance parameters such as throughput and collisions. The OZWCP development tool is therefore utilized for physical networking purposes to monitor network performance.

2.7.2 Python-Open-Zwave

Python-open-zwave is a python wrapper for OpenZWave open source software library [8]. Python-open-zwave allows users to implement the full networking capabilities of the OpenZWave library within a python development environment. This is beneficial for this project since the SimPy simulation model to be created necessitates the use of the Python programming language. Additionally the Home Assistant open-source home automation platform, to be discussed later, utilizes the python-open-zwave library.

All control and management of the generated z-wave network is achieved from one central location, namely the OpenZWave manager class. Further the event-based nature of network operation is regulated through use of notifications. Within the python development environment notifications are implemented through use of the PyDispatcher.

PyDispatcher

PyDispatcher allows one to transmit/dispatch a message signal onto a channel which is in return received by nodes registered/connected to the channel [9].

Utilizing PyDispatcher, python-open-zwave creates several z-wave network signals which are dispatched by the manager class in response to the triggering of specific events within the z-wave network. Significant events include for example the removal/failure of an included node or the change of a value stored on an included node. Network management is hence achieved by connecting to dispatched z-wave signals and appropriately processing events in the callback function triggered by the event.

2.8 Comparison of Open Source Home Automation Platforms

2.8.1 OpenHAB

OpenHAB (Open Home Automation Bus) is a home automation platform written in Java and fully based on the open service gateway initiative (OSGi). OSGi implies that specifications regulate the way plug-ins interact and one requires a machine which can support Java virtual machine (JVM) [10]. OpenHAB has an extensive, established network of supported devices, hence facilitating communication regardless of device manufacturer, hardware specifications or network protocol. The platform has a large community of users for support if one wishes to configure ones own devices. Further OpenHAB supports deployment onto both android and Apple operating systems, while the license for the platform's source code is readily available.

2.8.2 Domoticz

Domoticz is a home automation platform written in C/C++ which provides a HTML5 frontend and is therefore scalable and accessible from desktop and mobile devices [11]. The platform is primarily intended for lightweight applications such as sending notifications to simple sensors such as smoke detectors. The source code is made available under the GPLv3.

2.8.3 Home Assistant

Home Assistant is an open source home automation platform utilizing the python 3 programming language [12]. Home Assistant offers an all-in-one installer specifically for deployment onto a Raspberry Pi. The all-in-one installer automatically installs and configures useful packages such as the python-open-zwave wrapper of the OpenZWave library as well as the open-zwave-control-panel, radically simplifying the installation process within a Linux environment. Home Assistant was made available under a MIT license and the source code is available on GitHub. Finally Home Assistant provides seamless interaction with other communication interfaces (such as MQTT and IFTTT) as well as an attractive user interface.

Conclusion

Home Assistant was selected as the host application to govern communication within the Z-Wave network. The seamless network interoperability, python development environment as well as the readily available accessibility of the open-zwave-control-panel greatly facilitates creating and monitoring performance of custom network devices. Domoticz is more suited to provide lightweight support to simple proprietary devices, however this may impede development of nodes and analysing network performance. OpenHAB, despite boasting an extensive user community and support for OpenZWave, provides a less friendly user interface, complicating simple development of nodes and deployment of the Z-Wave network.

Chapter 3

The Z-wave Standard

3.1 ITU-T G.9959 standard

Z-wave adopts the ITU-T G.9959 standard which standardizes the media access control (MAC) layer, physical(PHY) layer, segmentation and re-assembly(SAR) layer and logical link control (LLC) layer specifications of short-range, narrow-band digital radio communication transceivers. In order to construct a simulation model which accurately models packet transmission, the key attributes of the above mentioned layers, as specified by the ITU-T G.9959 standard, are identified. Unless stated otherwise, all information contained within this section was found obtained from the ITU-T G.9959 recommendation [4].

3.2 Z-wave PHY Layer

The PHY layer specifies the modulation format, data rate as well as all methods implemented on the physical layer which ensure synchronization of transmitted frames. These methods include: clear channel assessment (CCA) and link quality assessment. The modulation and coding implemented is illustrated in table D.2.

3.2.1 Minimum Receiver Sensitivity

The minimum receiver sensitivity is the minimum power level at which the receiver will successfully detect the reception of a received packet. The minimum receiver sensitivity of the constructed z-wave network, with a data rate of 9600 bit/s, is -103 dBm [13].

3.2.2 Clear Channel Access

A node first queries the availability of the PHY layer before transmitting. This assessment ensures that all communication channels within wireless range of the sending node are idle. This assessment also implicitly ensures that the intended receiver is not currently transmitting packets and is in receiver mode. The maximum duration of this channel assessment is 1100 ms.

3.3 Z-wave MAC Layer

The MAC layer is responsible for controlling the radio frequency channel through which communication is achieved. Control is implemented by controlling channel access (through querying the PHY layer), collision-avoidance algorithms as well as coordinating retransmissions.

The MAC layer constants according to which the RF communication medium is regulated are listed in Appendix D

3.3.1 Frame Data Format

The frame which is transmitted consists of a preamble, start of frame (SOF), frame data and end of frame (EOF) symbol. The frame data is constructed according to the format illustrated in table D.3.

3.3.2 Collision Avoidance

Collision avoidance is achieved by placing a node in receiver mode when it is not currently transmitting bits. Accordingly if the physical layer to which a node is connected is found to be idle, the sending can assume the intended receiver to be in receiving mode and available for packet reception.

Random Backoff

In the event that a transmitted packet is corrupted, or if an acknowledgement is not received, the packet is retransmitted. To however prevent simultaneous retransmission of failed packets from different nodes, a random time delay between 10 ms and 40 ms is enforced.

3.3.3 Collision Occurrence

Packet collisions occur when two nodes attempt to simultaneously transmit a packet to the same receiving node. The collision is not avoided due to the fact that the CCA merely ensures that the receiving node is in receiving mode and additionally that all communication channels to which the sending node is subscribed are idle. The CCA does not detect that another node is currently busy transmitting a packet to the receiving node. The CCA error hence occurs during the transmission delay of the other packet which is transmitting a packet to the receiving node. The sequence of events leading to a packet collision are illustrated in figure 3.1.

3.4 Routing Layer

A transmitting node will always first attempt to transmit a packet directly to the intended receiver. In the case that receiving node is however out-of-range, or if the direct path has become corrupted, the sender will utilize signal routing to transmit a message to the intended receiver. Message routing involves transmitting the message to several intermediate repeater nodes which subsequently either transmit the message to the intended receiver or route the message through additional repeater nodes. Note that 4 repeating nodes is the maximum number of nodes which may be involved in the routing of a single message in a z-wave network.

3.4.1 Routing Responsibilities of Included Nodes

The z-wave protocol implements a source-routed mesh network. Here source-routed implies that all routing decisions are performed at the source, i.e. the primary controller. The primary controller in a z-wave network can communicate with all nodes included in the z-wave

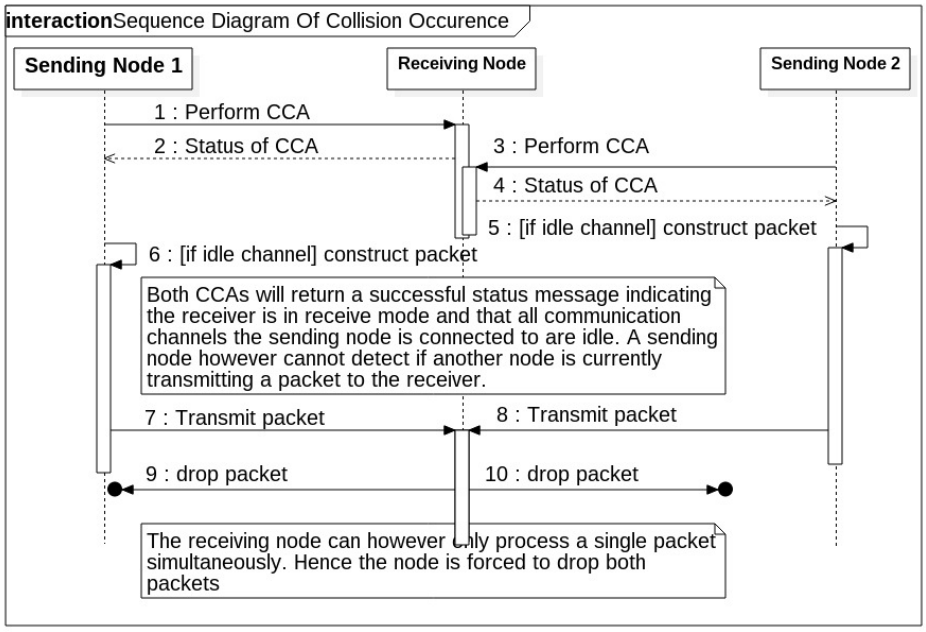


FIGURE 3.1: Sequence Diagram of Packet Collision

network. This through direct packet transmission, or by implementing message routing. This necessitates that the primary controller has access to the full routing table describing the z-wave network.

3.4.2 Determining Routes

All nodes included in a z-wave network have the ability to identify and report on all nodes within direct wireless range. This implies that controllers, slaves and routing slaves all have the ability to identify their direct neighbours with whom they have the capacity to communicate in the network. Hence the controller utilizes each node's capability to identify neighbouring nodes in order to create and correctly maintain the full routing table of a dynamic z-wave network.

The z-wave protocol analyses 3 possible methods to facilitate message routing. The primary controller first attempts packet transmission by utilizing the previous successful communication path. If this fails, the protocol will attempt packet transmission utilizing known possible routing alternatives established from the routing table available to the primary controller. The final possible method utilizes an explorer frame. Note that the z-wave protocol is limited to two retransmission attempts following failed packet transmission. Hereafter the primary controller will register the intended receiving node as a failed node.

3.5 Z-wave Transfer Layer

The z-wave transfer layer is responsible for controlling the transfer of data between 2 nodes.

3.5.1 Frame Types

The frame type of z-wave frame is determined by the number of intended receivers the frame is transmitted to. A singlecast frame is a frame which is transmitted to a specific target node where the receiver transmits a transfer acknowledge frame to notify the sender that the frame has been received. In the case where either the singlecast frame or the transfer acknowledge frame is lost or corrupted during transmission, the singlecast frame is retransmitted. A sequence diagram illustrating packet transmission within a two-node network is illustrated in figure 3.2.

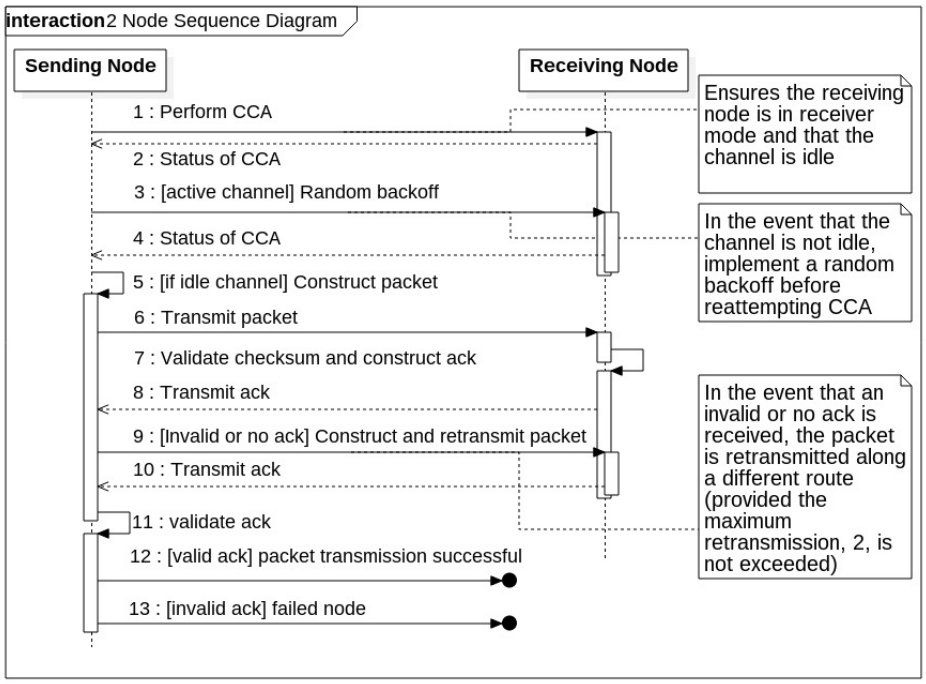


FIGURE 3.2: Sequence Diagram for 2-node Z-wave Network

A multicast frame can be transmitted to multiple nodes in the network while a broadcast frame is transmitted from the controller to all nodes included in the network. Both the multicast frame and broadcast frame however do not support the transmission of a transfer acknowledgement message, implying that unreliable data transfer can occur for these types of frames.

3.6 Z-wave Application Layer

3.6.1 Node Information Frame

All nodes forming part of a z-wave network (controllers, slaves and routing slaves) have the ability to determine which nodes included in the network are within direct communication range. When configuring network routing, all nodes report their neighbouring nodes to the primary controller. The primary controller subsequently uses this information to construct a rooting table describing all possible communication paths within the network.

3.6.2 Node Inclusion

Inclusion of nodes into the z-wave network is performed by the primary controller. When a node not included in a z-wave network is activated it broadcasts it's node information frame (NIF) to all nodes within wireless range. Provided the primary controller is in inclusion mode when the NIF is received, the primary controller will assign the network's unique home ID to the node to be included while also issuing the node with a node ID with which it will be identified within the network. Upon accepting the assigned home ID the node is included in the z-wave network.

3.7 Z-wave Network Operation

A state diagram indicating the integration of the above mentioned layers responsible for successful packet transmission within a z-wave network are illustrated in figure 3.3.

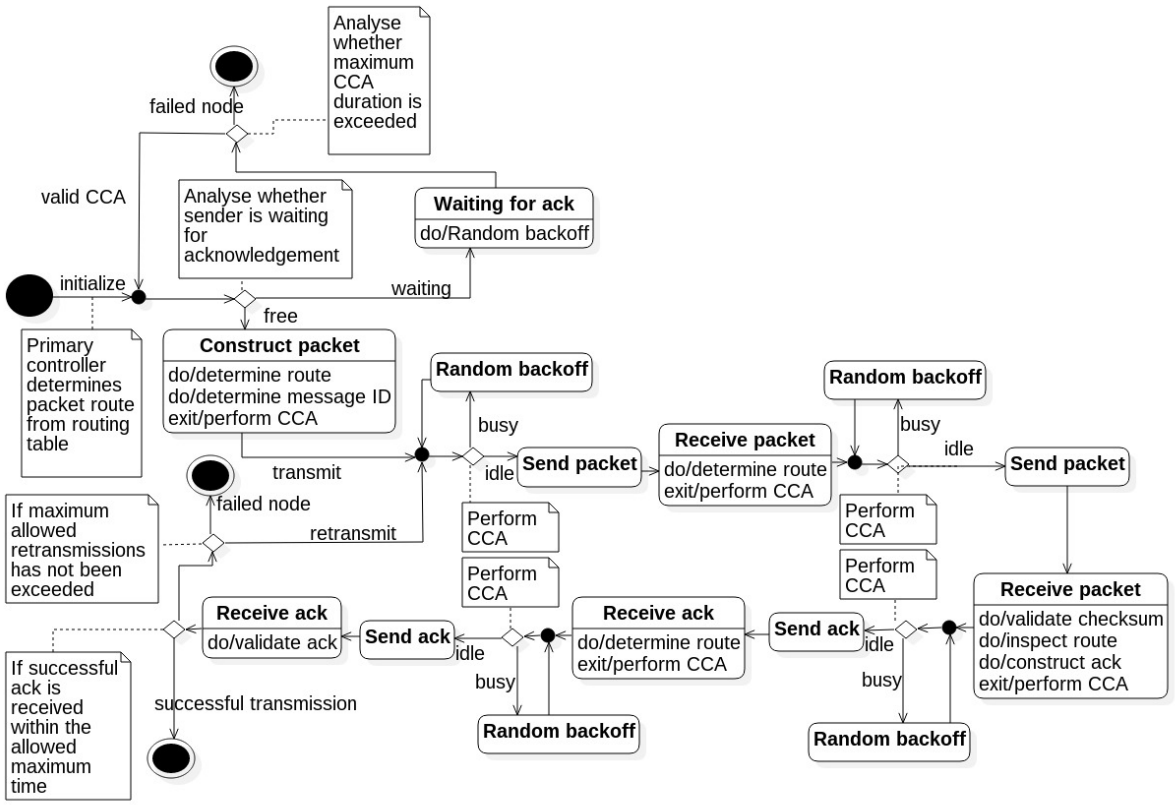


FIGURE 3.3: State Diagram for 3-Node Z-wave Network

Chapter 4

Z-wave Network Setup

4.1 Network Components

The created z-wave networks will consist of the following components:

4.1.1 Sigma Designs ZM5304 Module

The ZM5304 module [14] provides a z-wave wireless modem with a built-in antenna and fully integrated z-wave capabilities. The ZM5304 module is implemented to serve as either a primary or secondary controller within a created z-wave network. This node can be configured and controlled by open source software libraries through serial access of the z-wave serial API by a host application on a remote controller. Please see the datasheet attached in Appendix A for full specifications and capabilities of the module.

4.1.2 Aeotec Z-stick Gen5

A Z-stick USB controller [15] is implemented as it is an affordable hardware solution which allows the user to turn a host computer or micro-controller into a dedicated Z-Wave controller. This allows for the creation of a unique open source Z-Wave network as well as providing a customizable gateway for monitoring network performance. This specific manufacturer and model is selected based on reviews of users within the Z-Wave development community.

4.1.3 Aeotec Z-Wave LED Light Bulb Gen5

This z-wave enabled light bulb [16] is implemented to analyse the networking performance of the proprietary z-wave network. The device serves as a dedicated always-on slave, however it also has routing capabilities. This can facilitate analysis of packet routing between multiple nodes.

4.1.4 Raspberry Pi Model 1B

A Raspberry Pi model 1B micro-controller [17] will serve as the host for hardware and software solutions used to implement the Z-Wave network. This micro-controller is ideal for this application as it is low-cost, readily-available and runs the Linux operating system needed to support the identified Z-Wave open source platforms. The USB 2.0 ports housed on the Raspberry Pi will allow for the connection of the Z-stick USB controller. Further, the Raspberry Pi has an Ethernet port, allowing for the establishment of a SSH connection between the micro-controller and computer. Finally, the General-purpose input/output (GPIO) pins of the Raspberry Pi will

allow for the establishment of a serial connection between the host and z-wave serial interface modules.

4.2 Setting up the Raspberry Pi

The Raspberry Pi is configured in order to serve as the host of the z-wave network.

4.2.1 Preparing the SD card

The operating system of the Raspberry Pi is loaded on an SD card inserted into the micro-controller. The operating system implemented is Raspbian Jessie, the newest release of Raspbian which is based on Debian Linux. The image file of the latest distribution can be downloaded directly from the Raspberry Pi web-page under the Downloads tab [17].

Before the operating system can be installed on the Raspberry Pi the SD card needs to be formatted. The following guide outlines the steps to correctly flash an SD card: `http://qdosmsq.dunbar-it.co.uk/blog/2013/06/noobs-for-raspberry-pi/`

Having successfully formatted the SD card one can proceed to write the downloaded Raspbian image file onto the SD card in order to install the Raspbian operating system. The following guide correctly outlines this procedure: `http://elinux.org/RPi_Easy_SD_Card_Setup`

4.2.2 Establishing a Secure Shell (SSH) Connection

For the regular network configuration and communication applications of this project, the host micro-controller is operated into a 'headless' state. This implies that the micro-controller does not require an high definition monitor, keyboard and mouse; rather communication is established through a SSH connection with a computer. However to establish this SSH connection the micro-controller needs to be assigned a static internet protocol (IP) address, facilitating communication with computer.

4.2.3 Packet forwarding between Raspberry Pi and computer internet connection

In order to install open source software platforms, as well as update driver software as needed on the Raspberry Pi, an internet connection is needed by the Raspberry Pi. To achieve the required internet connection, one needs to allow packet forwarding from to the computer's domain name server (DNS) to the Raspberry Pi. If you want to use the computer as an internet gateway, one must configure the computer to allow packet forwarding through the gateway to which the Raspberry Pi is connected, as well as configuring the Raspberry pi to treat the static IP address assigned to the computer as the name server through which packets will be received/forwarded.

4.2.4 Installing Home Assistant on the Raspberry Pi

Home Assistant provides several methods for simple installation onto a Raspberry Pi micro-controller. These include an all-in-one installer, a docker image or a vagrant file. The Raspberry Pi all-in-one installer was utilized.

The only pre-requisite for this installation option involves upgrading the OS on your Raspberry Pi to the latest version of Raspbian (Jessie) described previously. This pre-requisite is due to the fact only a single all-in-one installer has been developed for installation onto the configuration files of the newest Raspbian installation, hence Jessie. Thereafter simply run the following three commands in the Terminal:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ wget -Nnv https://raw.githubusercontent.com/home-assistant/
  fabric-home-assistant/
  master/hass_rpi_installer.sh && bash hass_rpi_installer.sh
```

4.3 Z-wave Network Implementation Issues

Z-wave implements an interoperable, source-routed mesh network. To analyse the full networking capabilities of the z-wave protocol, the original proposed network was to consist of the following elements: an Aeotec z-stick serving a primary controller of the z-wave network; ZM5304 modules included as secondary controllers and finally an Aeotec light bulb serving as an always-on slave. The ZM5304 modules were considered as they presented a low-cost solution to extending the wireless routing capabilities of the z-wave network. Note that following inspection of the ZM5304 datasheet, attached in Appendix A, the assumption was made that the ZM5304 modules are embedded with z-wave controller firmware through which the user can access the serial API.

An error was however made when placing an order for the modules. Accordingly ZM5304U modules configured for operation at 915 MHz, the frequency allocated to the z-wave protocol in America, were acquired. Note that the acquired ZM5304U modules did well have z-wave static controller firmware installed, allowing one to establish a serial connection with the z-wave serial API. Upon discovery of the error, the correct ZM5304E modules configured for operation at 868 MHz, the frequency allocated to the z-wave protocol in South Africa, were ordered. The acquired ZM5304E modules, however, did not come with z-wave static controller firmware embedded. Accordingly the acquired ZM5304E modules were non-functional without the acquisition of an expensive SDK, which is outside of the scope of this project. Despite discussions with Sigma Designs, a temporary lease of the SDK for the application of this project could not be attained.

4.4 Description of Implemented Z-wave Networks

Subsequently, two separate z-wave networks are implemented. The first network is constructed using ZM5304 modules configured for operation at 915 MHz. The second network is constructed from an Aeotec z-stick and Aeotec z-wave light bulb configured for operation at 868 MHz. Note that the two separate networks still allow for analysis of networking performance achieved by the z-wave protocol. Additionally a comparison can now be drawn between the performance attained by a network constructed from stand-alone system-on-chip (SoC) modules versus a network constructed from proprietary z-wave products.

4.4.1 Z-wave Network Constructed with ZM5304 Modules

The first network is constructed from two identical z-wave serial interface modules. One node serves as the primary controller of the network, while the other serves as the secondary controller of the network. Both modules are hosted on respective Raspberry Pi micro-controllers, programmed with the python-open-zwave wrapper of the OpenZWave open source library. A deployment diagram of the 2-node network constructed with ZM5304 modules is illustrated in figure 4.1.

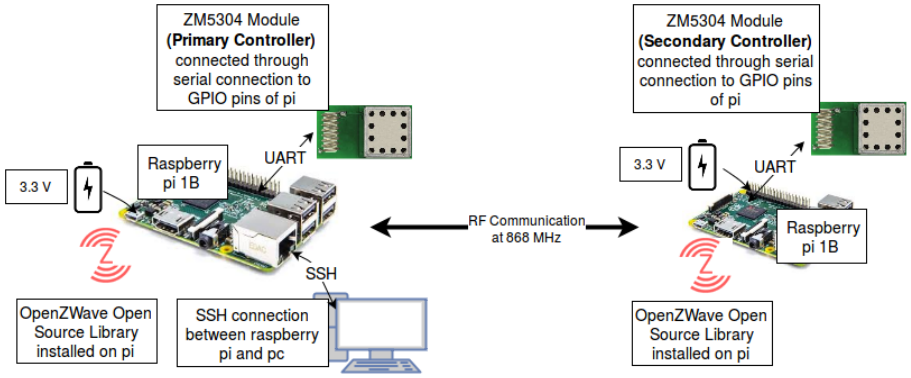


FIGURE 4.1: 2-node network constructed with ZM5304 modules

Connection of Elements Forming part of the Node containing the ZM5304 module

To facilitate communication between the ZM5304 modules and the Raspberry Pi, a universal asynchronous receiver/transmitter (UART) serial connection is established. The serial connection is established through use of the TX and RX pins of the Raspberry Pi's General purpose input/output (GPIO) pins, as well as the corresponding TX and RX pins on the ZM5304 modules. To interface and control the created z-wave network, an SSH connection is established between a computer and the Raspberry Pi hosting the primary controller of the z-wave network. An additional GPIO pin is selected to reset the module. Finally, the module is powered from the 3.3 V pin available on the Raspberry Pi.

Inclusion of Nodes into the Z-wave Network Constructed with ZM5304 Modules

The ZM5304 modules, from which the z-wave network is constructed, are programmed with static controller firmware. This implies that both nodes possess a unique homeID and have the capacity to serve as primary controller of a z-wave network. Therefore, upon receiving power during network bootstrapping, both nodes will independently create their own individual z-wave networks using their assigned homeID. A z-wave network may however only have a single primary controller. To facilitate communication between the nodes, one of the two nodes must be included as a secondary controller into the other existing z-wave network.

The inclusion of a secondary controller is achieved by placing the intended primary controller into inclusion mode and receiving the network configuration of the node to be added as a

secondary controller of network. Upon successful reception of the secondary controller's network configuration, the primary controller forgets it's own network configuration, adopts the received network configuration and becomes the primary controller of the z-wave network identified by the homeID specified by the received network configuration. The described inclusion process is illustrated in figure 4.2.

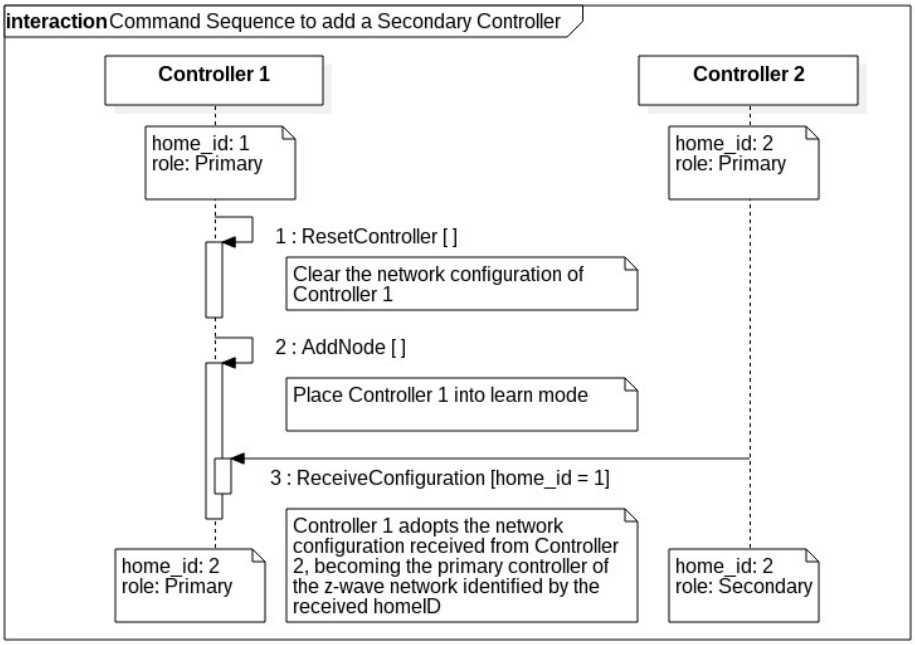


FIGURE 4.2: Command sequence to add a secondary controller

Pin Connections Between the ZM5304 Module and Raspberry Pi

The pin layout specifying the connection between the Raspberry Pi and ZM5304 module is illustrated in Appendix A. Note that, following recommendations specified in the datasheet, a 1.5 kΩ pull-up resistor is connected between the reset line and the 3.3 V rail of the Raspberry Pi. This pull-up resistor will ensure that the reset pin is not in a floating state when accessed by the Raspberry. Additionally, the pull-up resistor will implement a time constant, ensuring the module can detect and respond to the reset command issued. The power supply line is decoupled sufficiently with a 100 nF decoupling capacitor.

4.4.2 Z-wave Network Constructed with an Aeotec Z-stick and an Aeotec Z-wave Light Bulb

The second constructed network consists of an Aeotec Z-stick, hosted on a Raspberry Pi microcontroller programmed with OpenZWave software libraries, as well as an Aeotec z-wave light bulb. The primary controller is controlled by interfacing with the Raspberry Pi through an SSH connection. A deployment diagram of the 2-node network constructed with an Aeotec Z-stick and Aeotec Light Bulb is illustrated in figure 4.3.

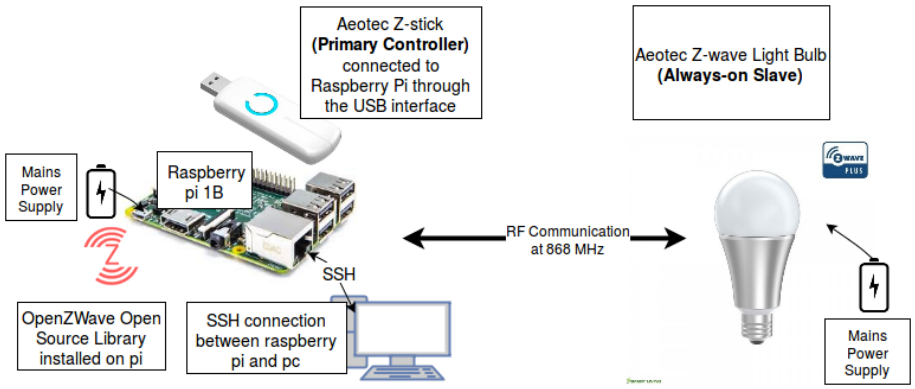


FIGURE 4.3: 2-node network constructed with an Aeotec Z-stick and Aeotec Light Bulb

Inclusion of Nodes into the Z-wave Network Constructed with an Aeotec Z-stick and an Aeotec Z-wave Light Bulb

The Aeotec z-wave light bulb will serve as an always-on slave, therefore no transfer of primary controller roles is necessary. A slave node is included into a z-wave network by placing the primary controller into inclusion mode and thereafter transmitting the NIF frame of the slave to the primary controller while it is in inclusion mode. This inclusion process for the proposed network is performed as follows:

- The Aeotec Z-stick is placed into inclusion mode by issuing the controller command, $AddNode$, through the serial interface. Subsequently the primary controller will await and receive the NIF frame from nodes to be included, provided they are within wireless range.
- The NIF frame of the Aeotec light bulb is transmitted in the event that the light bulb is powered on while not included into a z-wave network. Hence, provided the light bulb is not already included into a different z-wave network, simply power on the light bulb while the primary controller is in inclusion mode.

4.5 Interfacing with the Primary Controller of the Z-wave Network

Given the utilization of the Home Assistant all-in-one installer, there are 3 different methods though which one can interface with the primary controller of the z-wave network: the Home Assistant web application, the OZWCP or finally running a regular python script which includes the OpenZWave library.

Interfacing with the Primary Controller through the Home Assistant Web Application

The all-in-one installer configures auto-start of the Home Assistant web application at port 8123 of the Raspberry Pi. Therefore, following reboot of the Raspberry Pi, the web application is accessed: `http://ip_of_the_Raspberry_Pi:8123`

The configuration of included network components are stored and loaded from the configuration.yaml file located within the Home Assistant configuration folder. The configuration of slave nodes included into the z-wave network by the primary controller are automatically written to the configuration.yaml file. However, to utilize the primary controller which is connected to the Raspberry Pi through a serial connection, one must manually specify the location of the device's configuration file within the configuration.yaml file.

Interfacing with the Primary Controller through the OZWCP

The OZWCP is launched from the terminal and subsequently can be accessed at port 8888 of the Raspberry Pi. Before launching the OZWCP, first ensure that the Home Assistant web application is not running at port 8123, otherwise the OZWCP will not be able to lock the connected controller. The OZWCP is launched from the terminal with the following two commands:

```
$ cd /srv/hass/src/open-zwave-control-panel
$ sudo ./ozwcp -p 8888
```

Having launched the OZWCP, simply enter the path to the connected primary controller and select $initialize$.

Interfacing with the Primary Controller through the Python-Open-Zwave Library

The generated python script can be run from the terminal or any python development environment. This emphasises the flexibility of the designed solution,

Chapter 5

SimPy Network Simulation

5.1 The SimPy Simulation Library

A model of a z-wave network is created through utilization of the SimPy simulation library. SimPy is an event-driven discrete simulation library which allows for simulating physical components such as vehicles or message transmission [18].

Installation of the SimPy Simulation Library

The SimPy simulation library is ran on Python 3, simply as it is the newest Python distribution. Python 2 and PyPy can however also be used. If using an earlier Python distribution, one may need to first install *pip*, which is installed by default on Python 3. SimPy is installed with the following simple terminal command:

```
$ pip install simpy
```

Fundamental Operation of the SimPy Simulation Library

SimPy creates a simulation model through the use of Python generator methods, called processes. Processes found within a SimPy environment can both generate events and yield to generated events in order to wait for the events to be triggered. Note that yielding to an event implies that the current process, which generated and yielded the event, is suspended until the yielded event is triggered.

Jupyter Notebook

The simulation model was developed utilizing a Python 3 Notebook within the Jupyter Notebook. The Jupyter Notebook [19] was chosen as it provides an interactive development environment allowing one to develop and run small individual blocks of code independently within the same Notebook. This was useful for the development of isolated methods responsible for the various processes involved in packet transmission.

5.2 Modelling a Z-wave Network with SimPy

By utilizing SimPy, a simulation model of a z-wave network is created. A SimPy environment is created to represent the network environment to be modelled.

5.2.1 Modelling Temporal Network Performance

All the components of system latency can be modelled by yielding a timeout within the environment created for the network. Each subsequent phase involved in transmitting a packet will only be executed once the yielded timeout event is triggered. One can therefore model the temporal nature of packet transmission within the network, including network performance parameters such as throughput and round-trip-time (RTT).

5.2.2 Modelling Components of Packet Transmission

The event-driven nature of the SimPy simulation environment allows one to model the various processes facilitating the transmission and retransmission of packets within a network environment.

Communication Channels

The communication channels between nodes included in the network are modelled utilizing SimPy Stores. A SimPy Store allows one to model the production and consumption of python objects. Accordingly, a SimPy store is generated for each neighbouring node of the sending node, representing the communication channel between the two nodes. Networking is therefore modelled as the insertion and retrieval of generated packets (python objects) from the SimPy Store representing the communication channel between two nodes.

Clear Channel Assessment

Each node included within the z-wave network is modelled as a python object class. Inherent to this object class is a python list of neighbouring nodes of the node in question. Further, each node possesses a boolean status flag indicating whether the node in question is in receiver mode. Clear channel assessment (CCA) is therefore implemented through inspection of the status flag of each neighbouring node of a sending node. The result of the CCA is successful in the event that all neighbouring nodes of the sending node are in receiver mode. Upon completion of a successful CCA, the status flag of the sending node is set to false (indicating packet transmission) and is only reset to true following successful packet reception at the receiver.

Acknowledgement Recognition

Acknowledgement recognition is modelled by creating 2 SimPy events: one which is triggered by the reception of an acknowledgement frame from the receiver; the other in the event that the maximum allowed waiting time has elapsed. One can yield to the union of two events as follows:

```
response = yield event1|event2
```

The yielded conditional event will return an object containing the name of the triggered event as well as the returned value. Accordingly, inspection of the returned object allows the primary controller to appropriate manage packet transmission:

- In the event that a successful acknowledgement frame is received within the allowed time, packet transmission is successful and the process is terminated.

- If the maximum allowable time is exceeded, the method either initializes packet retransmission (if an alternative routing option exists and the maximum number of retransmissions has not been exceeded) or terminates packet transmission and registers the intended receiver as a failed node.

Waiting for Acknowledgement

A z-wave controller can only process a single packet transmission simultaneously. In the event that an additional transmission request is received concurrently, the new request will be queued while waiting for packet acknowledgement. The queue of packets awaiting transmission is modelled utilizing a SimPy Resource. A maximum capacity of transmission requests is assigned to the Resource, reflecting the memory available at the sending node. In the event that transmission requests are received at a rate faster than that at which the sending node can process requests, the queue will inevitably exceed it's maximum capacity, hence leading to packet loss. Finally each allocated resource needs to be released once packet transmission is complete, allowing the next queued transmission request to be executed. The freeing of resources is achieved by utilizing a $with$ statement, as follows:

```
with queue.request() as req:
        yield req
        event = sender.trigger_next
        response = yield event
```

5.2.3 Modelling Packet Transmission

Following successful queueing of a command to be executed, as well as a valid CCA, the processes of routing, packet construction and packet transmission are initiated. These processes are controlled by the $manage\_transmission()$ method. A communication diagram of the python methods involved in packet transmission are illustrated in figure 5.1. Please refer to the comments contained within the code for more information.

5.2.4 Modelling Node Inclusion

The transmission of an NIF frame is simulated by creating an object class for the node inclusion procedure. The generated object class represents the broadcast message and the communication channels connecting the sender to each of its neighbouring nodes. These communication channels are once again modelled as SimPy stores, however for the application of broadcasting, an individual SimPy store is generated for each neighbouring node within wireless range. Subsequently the frame to be broadcast is transmitted by simultaneously inserting the broadcast message into each identified neighbours' SimPy Store.

Additionally an event is created for each communication channel contained within the broadcast object. The generated event will be triggered in the event that a neighbouring node receives the broadcast NIF frame and responds with by transmitting its own NIF frame to the included node. If this response is received within the maximum time allowed for neighbour detection, the included node's list of neighbours will be updated accordingly. The node inclusion process is initialized by the *request_NIF ()* method.

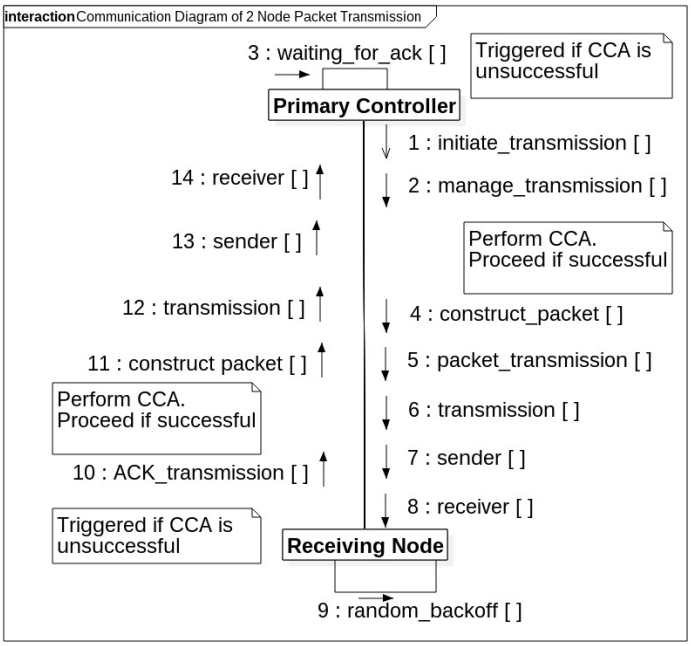


FIGURE 5.1: Python methods involved in modelling 2-node packet transmission

5.3 Simulating the Physical Topology of Nodes and Obstacles Contained Within the Z-wave Network

The SimPy simulation is created in order to model the essential network performance factors of a z-wave network. These metrics include latency, signal attenuation and throughput. To simulate the listed performance metrics, one needs to model the physical distance separating nodes and obstacles included in the z-wave network.

Physical modelling is achieved through the creation and maintenance of two NumPy arrays: one representing the location of included nodes, the other representing the location of walls separating nodes. Therefore a 2-dimensional (2D) representation of the physical network is created from which a model of the z-wave network can be constructed. Utilizing the created 2D network representation, one can determine the distance between included nodes as well as the number of obstacles a packet will penetrate during transmission between nodes.

5.3.1 Construction of the Obstacle Table

To quantify the link margin of a proposed route and accurately model signal attenuation, one needs to model the attenuation caused by obstacle penetrations. For the purpose of this project, a direct node-to-node transmission path is assumed, ignoring reflection and path bending. The number of walls a packet collides with, during transmission between 2 nodes, is determined by the primary controller following inclusion of a node into the z-wave network. Subsequently an obstacle table, similar to the routing table describing nodes within wireless range of one another, is constructed. The obstacle table is constructed by executing the process, described

below, for all obstacles contained within the network. This process is implemented by the $obstacles()$ method:

1. Construct 2 linear equations: one describing the direct transmission path between nodes; the other describing the wall in question.
2. Solve the two linear equations simultaneously in order to determine the x and y coordinates of the point of intersection of the equations describing the transmission path and wall.
3. In the event that the intersection point lies along the physical boundaries of the wall, increment the index of the obstacle table reflecting collisions occurring along the proposed transmission route.

5.3.2 Construction of the Routing Table

To facilitate packet routing, a routing table describing nodes within wireless range of one another is constructed by the primary controller. The routing table is constructed by analysing the link margin between the 2 nodes in question.

5.3.3 Determining Routes

The primary controller contains a list of all routes that can be travelled to reach an intended receiver. The *sense_routes()* method facilitates finding all new routing alternatives following inclusion of a node into the z-wave network. This is achieved by determining, for each included node, all routes which can be travelled from the node to an intended receiver.

During route discovery the primary controller queries the node in question for its neighbouring nodes. The primary controller will thereafter repeat the query procedure for each identified neighbouring node. This provided the following conditions are met:
The neighbouring node in question has not been previously queried in this particular route. The number of repeaters involved in the route does not already meet the maximum allowed amount of repeaters. The neighbouring node in question has routing capabilities i.e. it is either a controller or routing slave. Finally the neighbouring node is not the intended receiver, in which case the route discovery process is complete.

Provided the above listed conditions are met, the *investigate_route()* method is recursively called until the route discovery process is complete. The successfully identified route is thereafter added to the list of possible routing options.

5.4 SimPy Simulation Results

The generated SimPy simulation models the following network performance factors: link margin, throughput, latency and maximum range of communication. An example run illustrating the modelling of the z-wave protocol is illustrated in Appendix G.

Chapter 6

Physical Z-wave Networking

6.1 Method for Z-wave Networking Analysis

The python-open-zwave open source library is utilized to create, monitor and analyse the performance of the physical z-wave network implemented. Network management is achieved by connecting to signals dispatched by the OpenZWave manager class and thereafter processing events within the callback function triggered by the dispatched signal.

6.1.1 Network Creation

The z-wave network is initialized by creating a z-wave network object; triggering the created object to start the z-wave network and thereafter waiting for the state of the z-wave network to become ready. Once the network is ready one can connect to the z-wave signals of interest, dispatched by the OpenZWave manager class, through which network management is achieved.

6.2 Investigation to Quantify Observed Latencies for a 2-Node Network

To generate a simulation model which accurately models the performance of a z-wave network, it is essential that one can accurate model the temporal nature of the network. Simulation accuracy is however achieved, not only through theoretical predictions, but also by constructing a model from physically observed network performance parameters. The observed latency is analysed by synchronously transmitting commands from the primary controller to the included slave and monitoring achieved results. Reliability of achieved results is ensured by repeating packet transmissions 1000 times for each investigation discussed below.Additionally, sufficient time is allowed between successive retransmissions to ensure successful packet transmission (500 ms).

6.2.1 Background for Quantifying Observed Latency

In order to analyse observed latency when transmitting a packet between two nodes, it is essential to first have an thorough understanding of the packet being transmitted. The process to set the brightness of the light bulb consists of 3 command classes. The command class through which this dimming action is achieved is illustrated in Appendix B. The sequence of commands issued to adjust the brightness of the Aeotec z-wave light bulb is illustrated in figure 6.1.

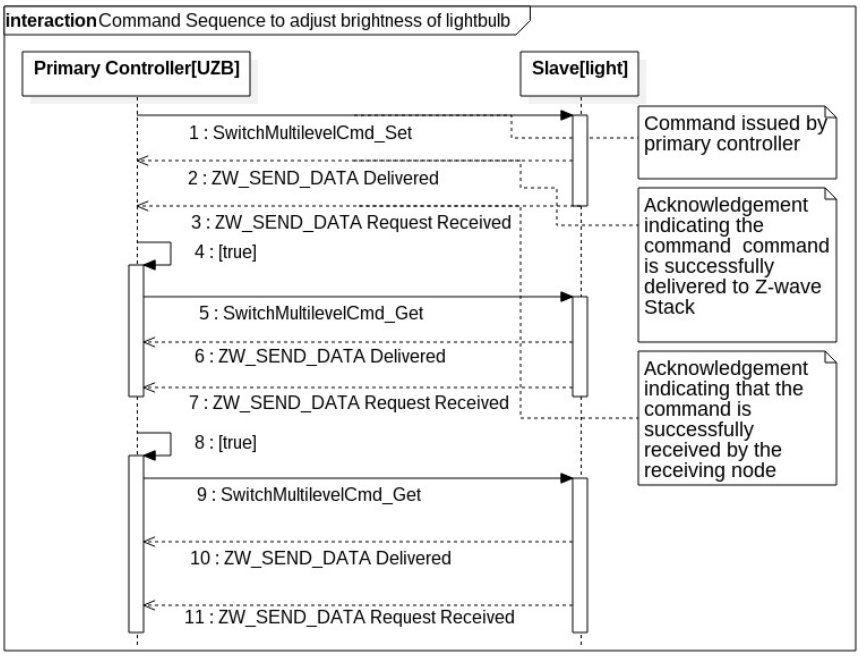


FIGURE 6.1: Command sequence to adjust brightness of the Aeotec z-wave light bulb

Message Length

The minimum, as well as most commonly occurring, length of the packet preamble for the symbol rate specified is 10 bytes [4]. Additionally each packet transmitted has a start of file (SOF) and end of file (EOF) byte included. The Set command to set the light bulb's brightness has a length of 4 bytes while the command to request the light bulb's brightness has a length of 2 bytes. The transmitted message frame therefore has a message length of 25 and 23 bytes for the Set and Get commands respectively. In the case of the Set command, the status of the returned acknowledgement frame is contained within an enumeration constant. This implies that the acknowledgement frame has zero data payload and the total message length of the acknowledgement frame is 21 bytes. In the case of the Get command, the brightness of the light bulb is returned as a single integer byte. The length of the acknowledgement frame for the Get command is therefore 22 bytes.

Method for Analysing Packet Transmission

In order to determine the delays involved with the transmission of message and acknowledgement frames, a comparison is drawn between the theoretical and observed transmission rates. In order to construct a significant observation, it is assumed that theoretically predicted transmission delays at the specified symbol rate are achieved. Any additional observed latency is therefore caused by queueing (channel congestion), routing, environment interference or transceiver delay.

Due to the fact that z-wave implements a source-routed mesh network, all message routing is performed by the primary controller. This therefore implies that all delays associated with queueing and packet routing occur at the primary controller at the start of packet transmission. Receiving nodes merely inspect the received message packet, validate the checksum and transmits an acknowledgement frame along the same route the message frame travelled to reach the node.

Additionally, there is a short theoretical time delay associated with the transition of a node from receiving mode to sending mode and vice versa. This is known as the TX-to-RX and RX-to-TX time delay. This transceiver limitation has a maximum allowed delay of 1 ms [4]. For the purpose of this investigation, assume a TX-to-RX and RX-to-TX time delay of 0.5 ms. For the 3-command sequence, there will be 3 TX-to-RX and 3 RX-to-TX time delays respectively, hence implying a total transition delay of 3 ms.

Finally, it is observed that a repeated command is only queued at the primary controller on the 1^{st} occasion that the command is transmitted. This observation was drawn through analysis of the command log displayed in the OZWCP. Accordingly, when adjusting the brightness of the light bulb, only two of the three commands issued by the primary controller will be queued prior to packet transmission.

6.2.2 Investigation to Quantify the Latency Associated with Queueing

The 1^{st} investigation will attempt to quantify the latency associated with the queuing of commands issued by the primary controller of the z-wave network. The investigation involves issuing a controller command to an intended receiver. The controller command, $requestNodeNeighbourUpdate$ is utilized for testing purposes.

The transmission of a controller command is broken into 3 distinct phases: phase 1 includes the queueing of commands to be transmitted by the primary controller, while the 2^{nd} and 3^{rd} phases include packet transmission and waiting for receiver acknowledgement respectively. Note that the OpenZWave manager class dispatches a signal on every occasion that the status of a controller command is updated. The possible states of an issued controller command include: $Starting$, $InProgress$ and $Completed$. The phases mentioned above are therefore triggered by connecting to the dispatched signal, $signal\_controller\_command$, and monitoring the state of the controller command. The latencies observed for the 3 phases of a controller command are illustrated in table 6.1.

Phase	**Bytes Sent**	**Theoretical Transmission Delay**	**Observed Latency (Average)**	**Observed Latency (Fastest)**	**Observed Latency (Slowest)**
1	0	0	8.919 ms	6.48 ms	21.661 ms
2	25	20.83 ms	61.771 ms	56.655 ms	86.94 ms
3	Unknown	Unknown	266.849 ms	253.968 ms	318.824 ms

TABLE 6.1: Latencies associated with the various phases of issuing a controller command

The phase of interest for this investigation is phase 1: the duration of this phase provides an indication of the typical latency associated with the primary controller with regard to queueing of issued commands. From the results illustrated in table 6.1, this latency is observed to be approximately 8.919 ms. This however involves the processing and queueing of 2 commands to be executed. The latency associated with the queueing of issued commands is therefore 4.45 ms per command issued.

6.2.3 Investigation to Quantify the Latencies Associated with Packet Routing and Packet Transmission

The 2^{nd} investigation will utilize the observed latency associated with queuing of issued commands (derived from investigation 1), as well as theoretically expected transmission delays (derived from the symbol rate), in order to determine the remaining components of system latency. The investigation again involves synchronously adjusting the brightness of a light bulb.

Note that the OpenZWave manager class dispatches a signal on every occasion that the value of a node, as seen by the primary controller, is updated. The value of a node, as seen by the primary controller, will however only change following a Get request from the controller, querying the value of the node. Recall that the command issued to adjust the brightness of the light bulb involves 1 Set command and two Get commands. The transmission of the command is therefore broken into 2 distinct phases: phase 1 includes the transmission of a Set and Get request, while phase 2 only includes a Get request. The phases mentioned above are triggered by connecting to the dispatched signal, $signal\_value\_update$, and monitoring the associated latencies.

Having accounted for the latency associated with the queueing of issued commands, the remaining components of system latency can be determined. These latencies include the time delays associated with packet routing at the primary controller an processing of received commands at the receiving node. The latencies observed for the 2 phases described previously are illustrated in table 6.2.

Phase	**Bytes Sent**	**Theoretical Transmission Delay**	**Observed Latency (Average)**	**Observed Latency (Fastest)**	**Observed Latency (Slowest)**
1	91	75.83 ms	118.817 ms	84.747 ms	233.596 ms
2	45	37.5 ms	51.175 ms	46.194 ms	73.673 ms

TABLE 6.2: Comparison of theoretical and observed latency

There are several other non-ideal latencies involved during packet transmission. Examples include interference or non-idealisms within the home environment, additional CCA's which may need to be performed due to unexpected packet loss as well as environmental conditions. For the purpose of this project, however, we will not attempt to quantify the latency introduced by these factors.

Latency Associated with Processing of Received Commands at the Receiver

An ideal transmission environment can be considered as one where all channels within range of both the sender and receiver node are idle. Additionally interference within the transmission environment will be negligible. Assuming that the fastest observed latency will occur when the transmission environment is ideal, one can derive the latency associated with processing of the received commands at the receiving node from the difference between the mean and fastest observed latencies. From the results for phase 2, illustrated in table 6.2, this latency is observed to be approximately 4.981 ms.

Latency Associated with Packet Routing

The final latency to be quantified is the latency associated with determining packet routing. Routing is performed by the primary controller prior to packet transmission. The total latency associated with packet routing is determined by subtracting all theoretical and observed latencies from the mean total transmission time of the command sequence. The routing associated with packet routing is calculated as follows:

$$
\begin{aligned}
t_{routing} &= t_{total} - (t_{queue} + t_{receiver} + t_{transmission} + t_{transition}) \\
t_{routing} &= 169.992 - ((2 \times 4.45) + (3 \times 4.981) + 113.33 + 2) \\
t_{routing} &= 30.819ms
\end{aligned}
$$

Note that latency calculated above is the routing latency observed for the entire 3-command sequence. The latency associated with determining packet routing at the primary controller is therefore 10.273 ms per command issued.

6.2.4 Extraction of Observed Latencies

The observed latencies can now be incorporated into the generated simulation model, allowing one to accurately model the temporal nature of z-wave network operation. The extracted network parameters are illustrated in figure 6.2.

6.3 Investigation to Quantify Signal Attenuation for a 2-Node Network

Signal attenuation within a z-wave network is analysed by investigating the data throughput and maximum range of communication for several typical home configurations. Physical configuration factors to be considered include the distance travelled by a packet as well as the number of obstacles (e.g. walls) a packet penetrates during transmission.

6.3.1 Determining Throughput of a Z-wave Network

To analyse the affect of signal attenuation on the observed data throughput, it is essential to specify and strictly adhere to the specified definition of throughput.

Note, for this project, throughput is defined as the number of bytes contained within the

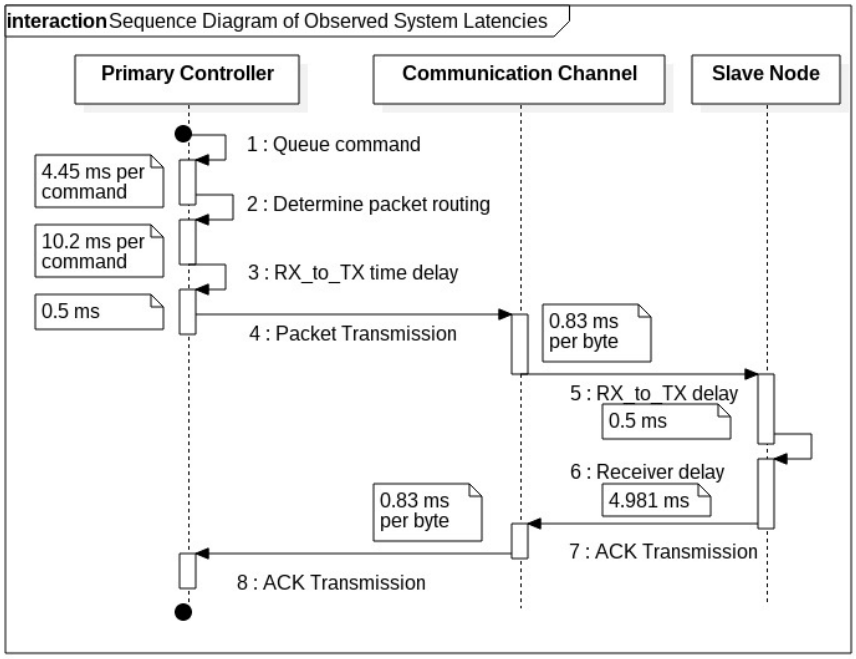


FIGURE 6.2: Sequence Diagram of Observed System Latencies

transmitted data payload (representing issued command classes). This implies that, given one knows the total number of bytes representing the commands to be issued, the only other parameter needed to quantify the achieved throughput is the total time taken to successfully issue the commands.

Data Payload

For a given command sequence, the number of successfully transmitted bits is determined by monitoring the number of commands successfully executed. For the constructed z-wave networks, this is once again achieved by subscribing to signals of interest dispatched by the OpenZWave Manager class.

Consider for example the case of adjusting the brightness level of a multilevel switch: The *Set* command has a data payload of 4 bytes while the returned acknowledgement frame has a data payload of 0 bytes. The two *Get* commands which follow both have a data payload of 2 bytes while the acknowledgement frames returned have a data payload of 1 byte each. The total data payload of the command class (including all three commands issued) is therefore contained within 10 bytes (or 80 bits) of data.

Total Round Trip Time

The total time taken to successfully issue commands is determined by summing the round trip times observed with each executed command. The current time is determined by utilizing the

time() method contained with Python's time library. The duration of packet transmission is determined through use of the callback function of dispatched signals. It is therefore essential to connect and monitor signals dispatched by the OpenZWave Manager class. Here the reader may note that following points of importance associated with determining the round trip time:

- The event where a single signal is triggered multiple times for an individual command issued is handled by only responding to a signal on the first occasion it is detected. This therefore eliminates the possible effects of multipath or signal jitter.
- In the event that multiple commands are queued at the primary controller, the recorded round trip time is allowed to continue until there are no commands left in the queue. Here it is assumed that queued commands are executed immediately upon completion of the previous command.
- In the event that the maximum number of retransmissions is exceeded, the round trip time is calculated as the time duration from issuing the command until the final retransmission attempt fails. Additionally, considering that the issued command has failed, the number of successfully transmitted bits contained within the data payload is reduced.

6.3.2 Method for Analysing Network Performance

To analyse the observed throughput and maximum range at which reliable packet transfer is achieved, a standard method is followed for all networking tests performed.

Networking Configurations Analysed

To analyse signal attenuation, several typical home networking scenarios are investigated. The tested scenario's include communication between 2 nodes which are: (a) in line-of-sight with one another, (b) obstructed by a single wall or (c) obstructed by 2 walls. The environment in which networking tests were performed is illustrated in figure 6.3.

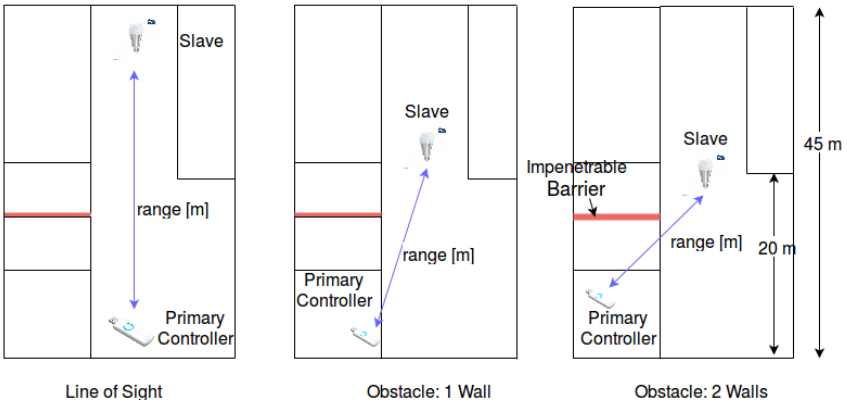


FIGURE 6.3: Networking Configurations Analysed

Note that the impenetrable barrier within the testing environment (indicated by the thick red line in figure 6.3) represents a metal cabinet for the storage of electronic equipment, which occupied the entire wall. The observed throughput when attempting to communicate with a

node placed on the other side of the barrier was extremely poor/negligible. Subsequently one may deduce that z-wave does not have the penetration capacity to penetrate metal objects.

Method for Analysing the Maximum Range of Communication

The primary controller and intended receiving node of the constructed network are first placed (within the networking configuration being analysed) at a distance where successful packet transmission is achieved. The test Python script is ran and achieved network performance is recorded. Thereafter the range of communication between the nodes is systematically increased, 1 meter at a time, until the range is detected where communication completely fails and packet transmission is no longer possible. During this procedure, the observed networking results are recorded for the threshold distance where observed throughput begins to decline; as well as for each test performed thereafter until the point where communication fails.

Method for Ensuring the Reliability of Observed Networking Results

The assumptions made with regard to signal attenuation necessitate the practise of a standard method with regards to network set-up and application. Applying a standard method will ensure consistent, reliable results while also minimizing the impact of factors, assumed to be negligible, which may potentially influence signal attenuation. The following standard practises were enforced:

- The orientation of included nodes were kept in a consistent position, oriented in the x-y plane, at all times. This minimized the effect of variable antenna gain according to orientation.
- When analysing the penetration capabilities of the z-wave protocol, the primary controller was placed within a 1 m of the obstacle to be penetrated. This eliminates the possibility of signal reflection as a reflected signal will be attenuated significantly by obstacles and reflections along the far longer transmission path.
- The primary controller for both constructed networks was hosted on a Raspberry Pi 1B micro-controller running the Raspbian Jessie operating system. Additionally no other processes were active on the micro-controller while tests were being conducted. This minimized the influence of the host's processing capabilities on observed results.
- All forms of possible interference within the testing environment were minimized at all times e.g. individuals walking across the communication path during tests.

6.3.3 Observed Networking Results

The networking tests performed allowed for an analysis of the observed throughput and maximum attainable range of communication for the described networking scenarios. The maximum range of communication observed for the various networking scenarios is illustrated in table 6.3. Note that the maximum range of reliable communication refers to the maximum distance where optimum throughput is achieved with zero errors.
The complete results from the performed networking tests are illustrated in Appendix C

Communication Scenario	Maximum Range of Reliable Communication	Range at which Communication Fails
LOS	39 m	43 m
1 Wall	22 m	25.3 m
2 Walls	15 m	17.5 m

TABLE 6.3: Maximum Ranges of Communication for the Network Constructed with an Aeotec Z-stick and Aeotec light bulb

Quantifying Signal Attenuation due to Environmental Interference

There are many factors within the environment contributing to signal attenuation which are not contained within the simple link budget equation (2.1). It is, however, outside of the scope of this project to account for the influence of these environmental factors on the observed signal attenuation.

The signal attenuation due to environmental interference is approximated as the link margin observed at the maximum possible range of line-of-sight (LOS) communication. LOS communication is chosen for this approximation to minimize the influence of attenuation due to penetrations with obstacles. The maximum range of communication is considered as, having accounted for all quantified causes of signal attenuation and minimized the attenuation caused by obstacles, the signal power remaining in the link margin must therefore be accounted for by environmental interference. From equation (2.3), the path loss at the maximum observed range of 43 m is 63.88 dBm. Applying this value to equations (2.1) and (2.2), the signal attenuation due to environmental factors is calculated to be 22.52 dBm.

Quantifying Signal Attenuation due to Penetration with a Wall

The observed networking results are utilized in order to quantify the signal attenuation caused by penetration of a z-wave packet with a wall. The observed signal attenuation, measured in dBm, is thereafter incorporated into the simulation model for accurately modelling the performance of z-wave networks.

Assume the signal attenuation due to environmental interference to be identical for all three testing scenarios (LOS, 1 wall and 2 walls). The signal attenuation due to obstacle penetration is derived from the difference in the path loss, observed at maximum communication range, for:

1. The network configurations involving LOS communication and communication through a single wall
2. The network configurations involving communication through 1 and 2 walls.

Utilizing equation (2.3), the difference in path loss for the 2 comparisons listed above is calculated to be 4.607 dBm and 3.201 dBm respectively. The signal attenuation caused by penetration with a wall is calculated as the mean of the 2 attenuations listed above. The signal attenuation due to penetration of a z-wave packet with a wall is therefore 3.904 dBm per wall penetrated. A

summary of the source of signal attenuation quantified during this investigation is illustrated in figure 6.4.

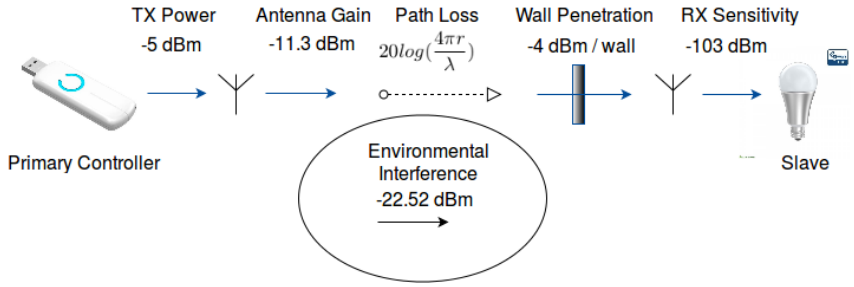


FIGURE 6.4: Quantified Sources of Signal Attenuation within a Z-wave Network

Probability of Successful Packet Transmission

A comparison of the data throughput achieved at different ranges of communication, for the 3 networking configurations tested, is illustrated in figure 6.5. The probability of successful packet transmission for the z-wave protocol is observed to approximate a step function. The probability of successful packet transmission is therefore unity for all ranges of communication up until the distance where the observed throughput begins to decline. Thereafter the probability is represented by a steep negative gradient with regard to throughput. This negative gradient continues until the probability of successful packet transmission is 0, implying packet transmission will fail.

For the 3 networking configurations tested, the width of the region between reliable packet transmission and packet failure is observed to be 0.848 dBm, 1.214 dBm and 1.339 dBm respectively. The mean width of this region is calculated to be 1.134 dBm. The probability of successful packet transmission within this region is calculated as the ratio of the remaining link margin over the mean width of this region. The probability of the successful packet transmission with a link margin of x dBm is therefore:

$$P(success) = \begin{cases} 1 & x \leq 1.134 \\ \frac{link\_margin}{1.134} & 1.134 \leq x \leq 0 \\ 0 & x \leq 0 \end{cases} \tag{6.1}$$

The steep gradient within the region where throughput declines can be accounted for by the error correction capabilities of the z-wave protocol.

6.4 Comparison of Networking Results for the Networks Constructed with ZM5304 modules and Aeotec products

The static z-wave controller firmware embedded on the ZM5304 modules only allow for the issuing of controller commands associated with node inclusion and network maintenance. Accordingly, due to the nature of the available command set, it is not possible to analyse the

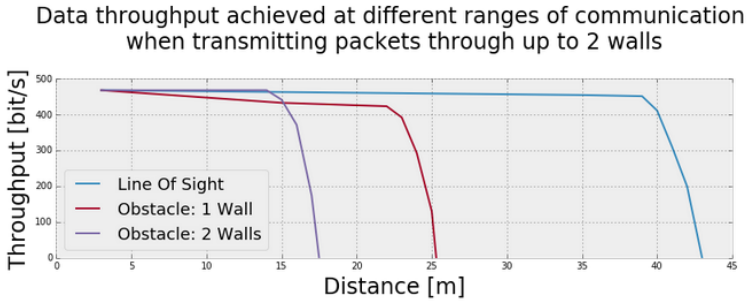


FIGURE 6.5: Data throughput achieved at different ranges of communication when transmitting packets through up to 2 walls for the z-wave network constructed with an Aeotec Z-stick and Aeotec light bulb

networking performance of the ZM5304 modules through synchronous transmission of a sequence of controller commands. A comparison between the networks is therefore drawn by analysing the maximum range and total round trip time (RTT) when issuing a short sequence of network maintenance commands. The Aeotec z-stick and Aeotec z-wave light bulb are both constructed modules supplied by Sigma Designs. One would expect similar networking results from the two constructed z-wave networks.

The results from the networking comparison are illustrated in figure 6.6. As expected the two constructed z-wave network exhibit similar performance. The z-wave network constructed with ZM5304 modules is, however, found to be slightly more robust, with a higher likelihood of successful packet transmission. Accordingly the total RTT is also improved. One may speculate that the slight lag in the performance of the Aeotec products is caused by the different z-wave firmwares embedded on the respective devices. Therefore, despite the fact that device interoperability is satisfactorily achieved, performance cannot complete with that of the identical controller modules.

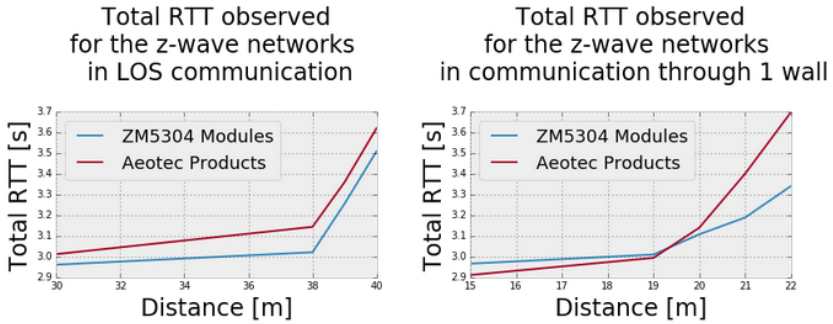


FIGURE 6.6: Comparison of Networking Results for the 2 Z-wave Networks Constructed

The complete results from the performed networking tests are illustrated in Appendix C

Chapter 7

Conclusion and Recommendations

7.1 Conclusion

The purpose of the report was to identify, analyse and select a home automation protocol applicable for deployment in homes. Additionally the selected protocol needed to provide open-source alternatives facilitating network analysis. Accordingly the z-wave protocol was selected. Z-wave is an affordable and accessible home automation protocol operating at IoT frequencies. Essentially, several open-source platforms (all of which leverage the OpenZWave library) exist, allowing for networking analysis. The z-wave protocol identified in the report therefore satisfy the specified objectives. Additionally the report successfully documents the standards though which the z-wave protocol achieves communication.

The report thereafter aimed to document the construction of a SimPy simulation model, modelling signal attenuation of z-wave packets. The simulation model described in the report satisfies this objective by modelling the attenuation caused by distance as well as penetrations with walls. Utilizing the SimPy simulation library, the described model accurately models throughput, latency and link margin achieved by the z-wave protocol.

Finally, the requirements of the report specified the construction of a physical z-wave network. Note that, following implementation issues experienced, two separate z-wave networks were constructed. The constructed networks allowed for successful quantification of signal attenuation through observation of latency, range and throughput. Networking analysis was facilitated by the OpenZWave open-source library, there emphasising the benefit of the z-wave protocol. Finally it is noted that standalone SoC modules are a viable low-cost z-wave solution.

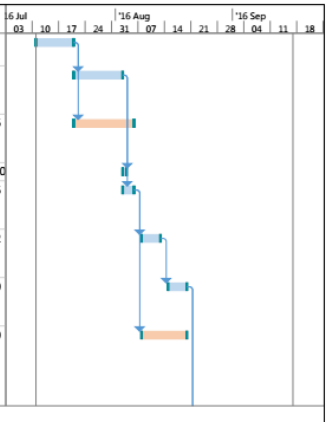
7.2 Recommendations

Given the limited time and budget assigned to the completion of the project, the project can be extended in numerous ways. Following the implementation issues experienced, the routing capabilities of the z-wave protocol were not tested. The report only analyses attenuation due to distance and penetration with walls. It would be desirable to construct a more robust simulation model, trained with data observed in a variety of network configurations. The testing environment utilized was far from ideal. Accordingly it is recommended to locate an appropriate testing environment with minimal sources of interference.

Appendix A

Project Planning Schedule

ID	Task Mode	Task Name	Duration	Start	Finish
1		Research networking protocols & standards	1.6 wks	Mon 16/07/11	Wed 16/07/20
2		Research available z-wave hardware and open source home automation software technologies	1.8 wks	Thu 16/07/21	Tue 16/08/02
3		Document literature study including research conducted, findings, decisions made as well as reasoning	12 days	Thu 16/07/21	Fri 16/08/05
4		Order hardware	1 day	Wed 16/08/0	Wed 16/08/0
5		Install & configure open source home automation software on raspberry pi microcontrollers	3 days	Wed 16/08/03	Fri 16/08/05
6		Construct 2 node z-wave network consisting of primary controller & 1 end device (pi with zwave module installed)	5 days	Mon 16/08/08	Fri 16/08/12
7		Model and simulate 2 node network in SimPy. Analyse simulated network performance.	5 days	Mon 16/08/15	Fri 16/08/19
8		Document implementations, results achieved, final designs and draw comparisons between the physical network and the simulated results for 2 node network	10 days	Mon 16/08/08	Fri 16/08/19



Appendix B

Outcomes Compliance

Appendix C

Physical Networking Results

C.1 Networking Results Attained from Adjusting the Brightness of the Aeotec Z-wave Light Bulb

C.1.1 Line-Of-Sight Packet Transmission

The following results were attained with a 2-node, indoor z-wave network where the sender and receiver nodes were kept in clear line-of-sight of one another at all times

Z-wave Network Constructed with an Aeotec Z-stick and an Aeotec Z-wave Light Bulb

Distance (m)	SOFs	Writes	ACKs	Total RTT (s)	Throughput (bits/s)	Errors
35	904	302	301	17.597	454.627	1
39	903	301	300	17.721	451.442	1
40	926	306	303	19.442	411.471	3
41	1108	333	324	25.802	310.054	9
42	1044	320	318	39.301	199.484	2

TABLE C.1: Results from modelling a 2-node z-wave network with LOS communication for the Z-wave Network Constructed with an Aeotec Z-stick and an Aeotec Z-wave Light Bulb

Distance (m)	Avg RTT (s)	Min RTT (s)	Max RTT (s)	Avg P1 (s)	Min P1 (s)	Max P1 (s)	Avg P2 (s)	Min P2 (s)	Max P2 (s)
35	176.0	133.9	272.9	124.7	85.3	219.9	51.2	47.1	63.5
39	177.21	133.5	281.2	124.9	84.2	219.1	52.31	48.1	68.1
40	194.4	132.1	305.7	140.3	83.2	229.8	54.1	47.9	90.2
41	258.0	132.9	429.0	183.8	85.6	380.2	74.2	21.9	195.0
42	401.0	159.5	1100.7	272.7	88.5	457.7	117.1	29.2	339.5

TABLE C.2: Results from modelling a 2-node z-wave network with LOS communication for the Z-wave Network Constructed with an Aeotec Z-stick and an Aeotec Z-wave Light Bulb

C.1.2 Packet Transmission Through 1 Wall

The following results were attained with a 2-node, indoor z-wave network constructed with ZM5304 modules where packets penetrate 1 wall duration transmission

Z-wave Network Constructed with an Aeotec Z-stick and an Aeotec Z-wave Light Bulb

Distance (m)	SOFs	Writes	ACKs	Total RTT (s)	Throughput (bits/s)	Retries or Dropped
15	927	304	304	18.476	432.9	0
22	906	301	301	18.90	423.280	0
23	987	314	314	20.4	392.178	0
24	1132	341	336	27.32	292.822	5
25	939	287	282	38.786	129.945	6

TABLE C.3: Results from modelling a 2-node z-wave network with 1 wall obstructing communication for the Z-wave Network Constructed with an Aeotec Z-stick and an Aeotec Z-wave Light Bulb

Distance (m)	Avg RTT (s)	Min RTT (s)	Max RTT (s)	Avg P1 (s)	Min P1 (s)	Max P1 (s)	Avg P2 (s)	Min P2 (s)	Max P2 (s)
15	184.76	134.1	319.3	130	85.6	255.6	54.7	30.7	158.6
22	189	133.5	303.12	133.2	85.1	280.11	55.8	33.1	171.1
23	204	133.95	383.6	143.8	85.5	332.3	60.2	30	160
24	273.1	133.1	546.0	200.3	85.1	426.1	72.8	28.9	192.5
25	343.3	137.7	657.9	231.6	86.1	493.4	99.6	29.1	282.3

TABLE C.4: Results from modelling a 2-node z-wave network with 1 wall obstructing communication for the Z-wave Network Constructed with an Aeotec Z-stick and an Aeotec Z-wave Light Bulb

C.1.3 Packet Transmission Through 2 Walls

The following results were attained with a 2-node, indoor z-wave network where packets penetrate 2 walls duration transmission

Z-wave Network Constructed with an Aeotec Z-stick and an Aeotec Z-wave Light Bulb

Distance (m)	SOFs	Writes	ACKs	Total RTT (s)	Throughput (bits/s)	Retries or Dropped
14	900	301	300	17.089	468.142	1
15	900	302	300	18.114	441.644	2
16	980	315	311	21.528	371.605	4
17	926	293	290	44.723	174.43	3

TABLE C.5: Results from modelling a 2-node z-wave network with with 2 walls obstructing communication for the Z-wave Network Constructed with an Aeotec Z-stick and an Aeotec Z-wave Light Bulb

Distance (m)	Avg RTT (s)	Min RTT (s)	Max RTT (s)	Avg P1 (s)	Min P1 (s)	Max P1 (s)	Avg P2 (s)	Min P2 (s)	Max P2 (s)
14	181.1	133.1	305.3	129.8	84.7	255.2	51.2	48.0	61.8
15	181.1	133.1	305.3	129.8	84.7	255.2	51.3	48.0	61.8
16	215.3	114.6	424.7	156.2	85.5	286.3	57.9	31.7	128.6
17	222.3	133.4	374.0	164.3	84.9	286.3	57.9	31.7	128.5

TABLE C.6: Results from modelling a 2-node z-wave network with LOS communication for the Z-wave Network Constructed with an Aeotec Z-stick and an Aeotec Z-wave Light Bulb

C.2 Comparison Between Networking Results Attained from Z-wave Networks Constructed with ZM5304 Modules and Aeotec Products

C.2.1 Line-Of-Sight Packet Transmission

The following results were attained with a 2-node, indoor z-wave network where the sender and receiver nodes were kept in clear line-of-sight of one another at all times

Z-wave network constructed with ZM5304 modules

Distance (m)	SOFs	Writes	ACKs	Total RTT (s)	Retries or Dropped
30	30	20	20	2.961	0
38	30	20	20	3.021	0
39	28	29	29	3.256	0
40	27	17	17	3.3.51	0

TABLE C.7: Results from modelling a 2-node z-wave network with LOS communication for the z-wave network constructed with ZM5304 modules

Z-wave Network Constructed with an Aeotec Z-stick and an Aeotec Z-wave Light Bulb

Distance (m)	SOFs	Writes	ACKs	Total RTT (s)	Retries or Dropped
30	30	20	20	3.012	0
38	30	20	20	3.144	0
39	28	18	17	3.36	1
40	24	16	16	3.662	4

TABLE C.8: Results from modelling a 2-node z-wave network with LOS communication for the z-wave network constructed with an Aeotec Z-stick and an Aeotec z-wave light bulb

C.2.2 Packet Transmission Through 1 Wall

The following results were attained with a 2-node, indoor z-wave network constructed with ZM5304 modules where packets penetrate 1 wall duration transmission

Z-wave network constructed with ZM5304 modules

Distance (m)	SOFs	Writes	ACKs	Total RTT (s)	Retries or Dropped
15	30	20	20	2.966	0
19	30	20	20	3.01	0
20	30	20	20	3.108	0
21	28	18	18	3.189	0
22	24	17	17	3.341	0

TABLE C.9: Results from modelling a 2-node z-wave network with 1 wall obstructing communication for the z-wave network constructed with ZM5304 modules

Z-wave Network Constructed with an Aeotec Z-stick and an Aeotec Z-wave Light Bulb

Distance (m)	SOFs	Writes	ACKs	Total RTT (s)	Retries or Dropped
15	30	20	20	2.911	0
19	30	20	20	2.994	0
20	30	20	20	3.14	0
21	28	19	18	3.401	1
22	25	17	15	3.698	2

TABLE C.10: Results from modelling a 2-node z-wave network with 1 wall obstructing communication for the z-wave network constructed with an Aeotec z-stick and an Aeotec z-wave light bulb

Appendix D

MAC/PHY Layer Constants

D.1 MAC Layer Constants

The MAC layer constants are the parameters according to which communication within the z-wave network is regulated.

MAC Constant	Description	Value
aPhyTurnaroundTimeTXRX	TX-to-RX maximum turnaround time	1 ms
aMacMaxRetransmitDelay	Maximum random back-off	40 ms
aMacMinRetransmitDelay	Minimum random back-off	10 ms
aMacMaxFrameRetries	Maximum number of retries after a transmission failure	2

TABLE D.1: MAC Layer Constants [4]

D.2 Modulation and Coding Format

The modulation and coding implemented is illustrated in table D.2.

Data Rate	Modulation	Coding	Frequency Offset	Separation	Symbols
9.6 kbit/s	FSK	Manchester	20 kHz	40 kHz	Binary

TABLE D.2: Modulation and coding format [4]

D.3 Frame Data Format

The frame data is constructed according to the format illustrated in table D.3.

Home ID	Source_ID Node ID	Frame Header	Length	Destination Node ID	Data Payload	Checksum
4 bytes	1 byte	2 bytes	1 byte	1-4 bytes	0-54 bytes	1 byte

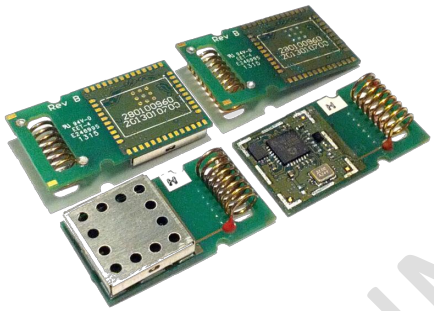
TABLE D.3: Frame Data of Z-wave Packet [4]

Appendix E

ZM5304 Datasheet

DATASHEET: ZM5304

FULLY INTEGRATED Z-WAVE® WIRELESS MODEM WITH ON-BOARD ANTENNA



The Sigma Designs ZM5304 Modem is a fully integrated Z-Wave modem module in a small 27mmx15.2mmx5.5mm form factor. It is an ideal solution for home control applications such as access control, appliance control, AV control, building automation, energy management, lighting, security, and sensor networks in the "Internet of Things".

A baseband controller, sub-1 GHz radio transceiver, crystal, decoupling, SAW filter, matching, and the antenna is included to provide a complete Z-Wave solution to an application executing in an external host microcontroller. The ZM5304 Modem is certified with the FCC modular approval, ready to be used in any product without additional testing and license costs.

The ZM5304 Modem is based on an 8-bit 8051 CPU core, which is optimized to handle the data and link management requirements of a Z-Wave node. The UART or USB interface can be used to access the Z-Wave stack available in the on-chip Flash memory, or to easily upgrade the modem firmware.

FCC ID	TBD
IC ID	TBD

Features

- Complete Z-Wave stack available over UART or USB
- 32kB of byte addressable NVM memory
- Fully Integrated crystal, EEPROM, SAW filter, matching circuit, and antenna
- Supply voltage range from 2.3V-3.6V for optional battery operation
- No external components required
- FCC modular approval
- CE self-certified
- ITU G.9959 compliant

Radio Transceiver

- Receiver sensitivity with SAW filter down to -103dBm
- Transmit power with SAW filter up to +2dBm
- Z-Wave 9.6/40/100kbps data rates
- Supports all Z-Wave sub-1 GHz frequency bands (865.2-926.3 MHz)
- Supports multi-channel frequency agility and listen before talk
- Regulatory Compliance
  ACMA: AS/NZS 4268
  CE: EN 300 220/489
  FCC: CFR 47 Part 15 Modular Approval
  IC: RSS-GEN/210
  MIC: ARIB STD-T108

Modem

- UART speed up to 230.4kbps
- USB 2.0 full speed
- Z-Wave serial API accessed over UART or USB
- Firmware upgradeable via UART or USB
- TX mode current typ. 40mA @ +2dBm
- RX mode current typ. 32mA
- Normal mode current typ. 15mA
- Sleep mode current typ. 2µA
- Less than 1ms cold start-up time
- Power-On-Reset / Brown-out Detector

Appendix F

Command Classes

F.1 Multilevel Switch

It is important for the reader to note that the z-wave light bulb utilized for this project implements version 2 of the Multilevel switch command class.

F.1.1 Multilevel Switch Set Command

Byte	Description
1	Command Class = COMMAND_CLASS_SWITCH_MULTILEVEL
2	Command = SWITCH_MULTILEVEL_SET
3	Value
4	Duration

TABLE F.1: Command class frame for to set the value of a multilevel switch [20]

F.1.2 Multilevel Switch Get Command

Byte	Description
1	Command Class = COMMAND_CLASS_SWITCH_MULTILEVEL
2	Command = SWITCH_MULTILEVEL_GET

TABLE F.2: Command class frame for to get the value of a multilevel switch [20]

F.2 Node Neighbour Update

F.2.1 Node Neighbour Update Request Command

Byte	Description
1	Command Class =COMMAND_CLASS_NETWORK_MANAGEMENT_INCLUSION
2	Command = COMMAND_NODE_NEIGHBOR_UPDATE_REQUEST
3	Sequence Number
4	NodeID

TABLE F.3: Command class frame to instruct a node to perform a Node Neighbour Search [14]

F.2.2 Node Neighbour Update Status Command

Byte	Description
1	Command Class =COMMAND_CLASS_NETWORK_MANAGEMENT_INCLUSION
2	Command = COMMAND_NODE_NEIGHBOR_UPDATE_STATUS
3	Sequence Number
4	Status

TABLE F.4: Command class frame to request the status of a Node Neighbour Update Request [14]

Appendix G

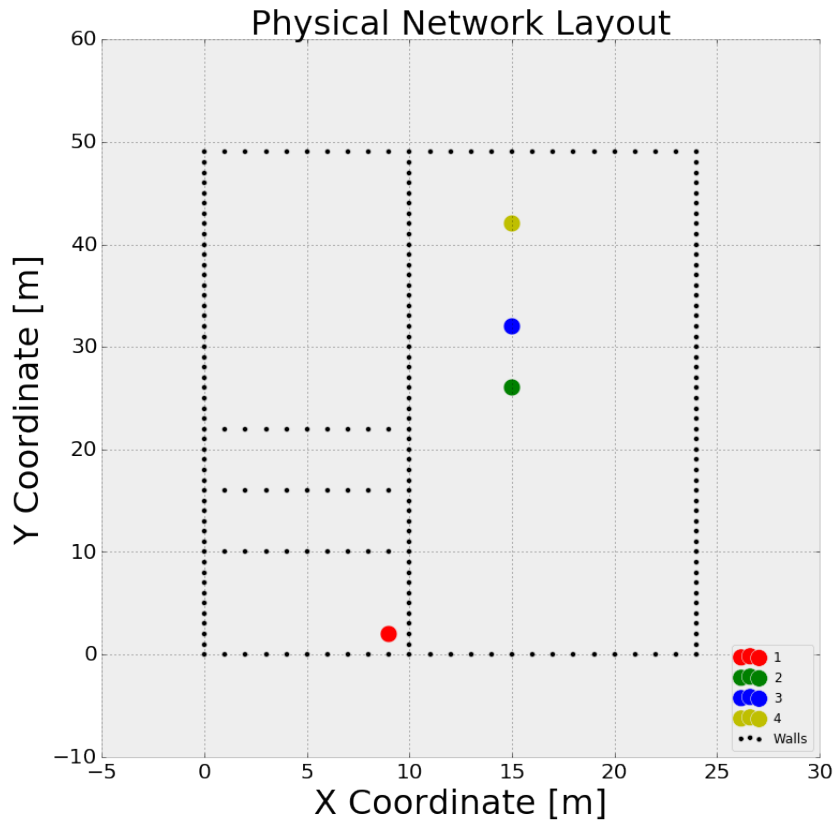
Simulation Results

```
# roots.addNode(env, 7, 8, 0, 0)

# Slaves
roots.addNode(env, 15, 26, 0, 1) # Node 3
roots.addNode(env, 15, 32, 0, 1) # Node 4
roots.addNode(env, 15, 42, 0, 1) # Node 5


#display visual representation of network
roots.show_grid()
```

```
/home/jason/anaconda3/lib/python3.5/site-packages/ipykernel/__main__.py:1018: DeprecationWarning: using a
```




```
Coordinates of nodes
[[ 9  2]
```

```
 [15 26]
 [15 32]
 [15 42]]
Number of nodes: 4
Distance between nodes
[[  0.          24.73863375  30.59411708  40.44749683]
 [ 24.73863375   0.           6.          16.        ]
 [ 30.59411708   6.           0.          10.        ]
 [ 40.44749683  16.          10.           0.        ]]
Link margins
[[  0.           0.80475314  -1.06368057  -3.27764652]
 [  0.80475314   0.          17.1048083    8.58543366]
 [ -1.06368057  17.1048083    0.          12.66783331]
 [ -3.27764652   8.58543366  12.66783331   0.        ]]
Rooting table
[[ 0.  1.  0.  0.]
 [ 1.  0.  1.  1.]
 [ 0.  1.  0.  1.]
 [ 0.  1.  1.  0.]]
Available Routes
Routes from node 1

Node 1 - Node 2:
[1 2]

Node 1 - Node 3:
[1 2 3]
[1 2 4 3]

Node 1 - Node 4:
[1 2 3 4]
[1 2 4]
Routes from node 2

Node 2 - Node 1:
[2 1]

Node 2 - Node 3:
[2 3]
[2 4 3]

Node 2 - Node 4:
[2 3 4]
[2 4]
Routes from node 3

Node 3 - Node 1:
[3 2 1]
[3 4 2 1]

Node 3 - Node 2:
[3 2]
[3 4 2]
```

```
Node 3 - Node 4:
[3 2 4]
[3 4]
Routes from node 4

Node 4 - Node 1:
[4 2 1]
[4 3 2 1]

Node 4 - Node 2:
[4 2]
[4 3 2]

Node 4 - Node 3:
[4 2 3]
[4 3]
Wall Grid:
[ 0.  1.  1.  1.]
[ 1.  0.  0.  0.]
[ 1.  0.  0.  0.]
[ 1.  0.  0.  0.]

In [11]: #Transmit individual packets with a delay between transmissions
         env.process(roots.send_message(100))
         # env.process(roots.send_message(200))
         # env.process(roots.send_message(300))

Out[11]: <Process(send_message) object at 0x7fb680ab1c50>

In [12]: print_network_statistics()

Successes: 0
Collisions: 0

In [13]: try:
             env.run(until=10000)
             print('Simulation ran according to plan')
         except RuntimeError:
             print('Simulation ran slow')

Start Node: 1
Target Node: 2
New transmission attemptMessage ID: 1Sender: Node 1Target: Node 2
Node 1 is waiting for an ACK
Message 2 added to queue
Node 1 is waiting for an ACK
Message 3 added to queue

 Message sent from 1 to 2 at 119

Packet transission time:  21.333  s
Target Destination Reached!Route Packet Travelled:  [1 2]

 ACK sent from 2 to 1 at 140
```

```
queue count:  2
Transmission SuccessfulMessage ID: 1
Round Trip Time:  54.921  ms
Retransmission attemptMessage ID: 2Sender: Node 1Target: Node 2
Retransmission attemptMessage ID: 3Sender: Node 1Target: Node 2
Node 1 is waiting for an ACK
Message 3 added to queue

 Message sent from 1 to 2 at 175

Packet transission time:  19.667  s
Target Destination Reached!Route Packet Travelled:  [1 2]

 ACK sent from 2 to 1 at 194

queue count:  1
Transmission SuccessfulMessage ID: 2
Round Trip Time:  53.254  ms
Retransmission attemptMessage ID: 3Sender: Node 1Target: Node 2

 Message sent from 1 to 2 at 229

Packet transission time:  19.667  s
Target Destination Reached!Route Packet Travelled:  [1 2]
Lost Packet
No Packet Received, try again

 Message sent from 1 to 2 at 729

Packet transission time:  19.667  s
Target Destination Reached!Route Packet Travelled:  [1 2]

 ACK sent from 2 to 1 at 749

queue count:  0
Transmission SuccessfulMessage ID: 3
Round Trip Time:  553.254  ms
Total Round Trip Time:  672.329  ms
Throughput:  142.787  bits/s

Simulation ran according to plan
```

```
In [14]: print(roots.broadcast[3].neighbours)
         print(roots.broadcast[2].neighbours)
         print(roots.broadcast[1].neighbours)
         print(roots.broadcast[0].neighbours)
```

```
[3, 2]
[2, 4]
[1, 3, 4]
[2]
```

```
In [15]: a = []
         len(a)
```

```
Out[15]: 0
```

Bibliography

[1] D. Prindle. (Jan. 2014). Home automation standards explained, [Online]. Available: http://www.vesternet.com/resources/technology-indepth/how-z-wave-controllers-work (visited on 10/01/2016).

[2] S. Automated. (2016). About upb technology, [Online]. Available: http://www.simply-automated.com/UPB_Technology.php (visited on 07/25/2016).

[3] M. Louse. (2016). Latency, [Online]. Available: http://whatis.techtarget.com/definition/latency (visited on 08/01/2016).

[4] I. T. Union, *Short range narrow-band digital radiocommunication transceivers – PHY, MAC, SAR and LLC layer specifications*, *Recommendation ITU-T G.9959*. Jan. 2015. [Online]. Available: http://www.itu.int/rec/T-REC-G.9959-201501-I.

[5] Digi, *Indoor Path Loss*.

[6] J. Hammond. (2016). Openzwave, [Online]. Available: https://github.com/OpenZWave/open-zwave (visited on 08/01/2016).

[7] ——, (). Open-zwave-control-panel, [Online]. Available: https://github.com/OpenZWave/open-zwave-control-panel.

[8] python openzwave. (2016). Python wrapper for openzwave, [Online]. Available: https://github.com/OpenZWave/python-openzwave (visited on 10/01/2016).

[9] M. C. Fletcher. (). Pydispatcher, [Online]. Available: http://pydispatcher.sourceforge.net/.

[10] openHAB. (2016), [Online]. Available: http://www.openhab.org/ (visited on 07/25/2016).

[11] Domoticz. (2016), [Online]. Available: https://domoticz.com/ (visited on 07/25/2016).

[12] H. Assistant. (Aug. 2016). Raspberry pi all-in-one installer, [Online]. Available: https://home-assistant.io/getting-started/installation-raspberry-pi-all-in-one/.

[13] S. Designs, *Fully integrated z-wave wireless modem with on-board antenna*, *Datasheet: ZM5304*. Sigma Designs, 2013. [Online]. Available: https://fccid.io/pdf.php?id=2052478.

[14] N. T. Johansen, *Software Design Specification*, *Z-Wave Command Class Specification, N-F*. Sigma Designs, Aug. 2016, pp. 281–324.

[15] Aeotec, *Z-Wave Aeon Labs Z-Stick USB Controller gen5 Manual*. Nov. 2015. [Online]. Available: http://www.vesternet.com/downloads/dl/file/id/664/product/1541/z_wave_aeon_labs_z_stick_usb_controller_gen5_manual.pdf.

[16] ——, *Z-Wave Aeon Labs LED Bulb gen5 Manual*. Nov. 2015. [Online]. Available: https://zasmarthomes.co.za/wp-content/uploads/2015/11/z_wave_aeon_labs_led_bulb_gen5_manual.pdf.

[17] R. Pi. (2016). Raspbian, [Online]. Available: https://www.raspberrypi.org/downloads/raspbian/.

[18] SimPy. (2016). Documentation for simpy, [Online]. Available: https://simpy.readthedocs.io/en/latest/contents.html (visited on 10/01/2016).

[19] Jupyter. (2016), [Online]. Available: http://jupyter.org/ (visited on 09/01/2016).

[20] N. T. Johansen, *Software Design Specification, Z-Wave Command Class Specification, A-M*. Sigma Designs, Aug. 2016, pp. 281–324.