Jarmo Hirvonen

Design and implementation of Andon system for Lean manufacturing

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 08.11.2018
Thesis supervisor: Prof. Pekka Eskelinen
Thesis advisor: DI Risto Malmstedt
Lean manufacturing provides tools and concepts to manufacture more with less by reducing waste. The waste in Lean may be found from almost anything that does not increase the value of product being manufactured. Visual systems are one of the tools used in Lean manufacturing. These visual tools include, Andon, system that is developed in this thesis. Andon functions as colored signal on production line to quickly give status information.

New system was developed capable of controlling multi-colored Z-wave LED light bulbs. Also, simple latch buttons with Z-wave enabled were created to allow alert generation. The system is able to communicate with other application by using websocket and HTTP request methods. The complete Andon system is able to control multi-colored LED lights in production floor with colors depending on the status of the alerts, which are generated by press of a button or by using the other application.

The application has been taken into use successfully in the company. To this date 30 lights have been installed. These are being controlled by three different Raspberry Pis with RaZberry daughter card installed and approximately 6500 alarms have been generated to this date. More lights are being taken into use in production and at some point the Andon system will cover all the production stations on multiple floors.

**Keywords** andon, raspberry pi, z-wave
Lean-tuotannon ideana on tuottaa enemmän vähemmällä. Tämä tapahtuu vähentämällä hukkaa erilaisten työkalujen ja konseptien avulla. Leanissa hukkaa voidaan löytää lähes kaikkialta, mikä ei tuota itsessään arvoa tai nosta jo tuotannossa olevan arvoa. Yksi työkalu jonka Lean tuo, on näkyvyyttä lisäävät järjestelmät. Andon, joka on tämän diplomityön kehityksen kohteena, kuuluu näihin järjestelmiin. Andon toimii värillisenä signaalina tuotantolinjalla antaen nopean tilannekuvan linjan tilasta.


Avainsanat andon, raspberry pi, z-wave
Preface

This thesis was conducted at General Electric Healthcare Finland Oy. The thesis didn’t start as a thesis work but was pushed into being one. The company had decided on development of Andon system and at first it was going to be developed by 3rd party. After further discussion it was decided that the system would be developed in-house, me as the developer.

I would like to thank my manager of that time Pirjo Kuittinen for actively looking for thesis project for me. I would also like to thank Jyrki Vornanen and Erik Tigerstedt on the ideas for the application developed. I would also like to thank Professor Pekka Eskelinen for his comments on written parts of the thesis.

Helsinki 08.11.2018

Jarmo Hirvonen
Table of contents

Abstract
Abstract (in Finnish)
Preface
Table of contents........................................................................................................... i
Abbreviations.................................................................................................................. iii
List of Figures.................................................................................................................... iv
List of Tables .................................................................................................................. v
1 Introduction.................................................................................................................. 1
2 Background................................................................................................................... 3
   2.1 Lean....................................................................................................................... 3
      2.1.1 Lean principles............................................................................................... 3
      2.1.2 Lean tools....................................................................................................... 5
      2.1.3 Visual systems............................................................................................... 5
   2.2 Z-wave.................................................................................................................... 7
      2.2.1 Z-way.............................................................................................................. 9
   2.3 Raspberry Pi ......................................................................................................... 10
   2.4 RaZberry .............................................................................................................. 11
   2.5 HTTP and HTTP request methods ..................................................................... 12
   2.6 Websocket and STOMP ..................................................................................... 13
   2.7 JavaScript Object Notation .............................................................................. 13
   2.8 Unified Modeling Language ............................................................................. 14
   2.9 Node.js and promises ......................................................................................... 15
3 Requirements ............................................................................................................. 16
   3.1 General requirements ......................................................................................... 16
   3.2 Requirements to Application 2 from Application 1a........................................... 18
   3.3 Requirements to Application 2 from Application 1b........................................... 20
4 Andon-raspberry hardware ....................................................................................... 22
   4.1 RaZberry ............................................................................................................ 22
   4.2 Raspberry Pi with RaZberry support .................................................................. 22
   4.3 Multicolored Z-wave LED light bulbs................................................................. 23
   4.4 Light sockets for the bulbs ................................................................................ 23
   4.5 Buttons to create alerts ..................................................................................... 23
   4.6 Range extenders for Z-wave network ................................................................. 25
   4.7 Component overview ....................................................................................... 25
5 Andon-raspberry software ....................................................................................... 26
   5.1 Overview ............................................................................................................. 26
   5.2 Libraries ............................................................................................................. 27
   5.3 Database and data schematics ......................................................................... 28
      5.3.1 ‘token’ database entry ................................................................................... 28
      5.3.2 ‘user’ database entry .................................................................................... 29
      5.3.3 ‘alert’ database entry................................................................................... 29
      5.3.4 ‘andon’ database entry .............................................................................. 30
      5.3.5 ‘button’ database entry .............................................................................. 30
      5.3.6 ‘zwave’ database entry .............................................................................. 31
   5.4 Communication with Application 1 ..................................................................... 32
5.5 Communication with Z-way server ....................................................... 32
5.6 Website and API .................................................................................. 34
5.7 Utilities.................................................................................................. 36
5.8 Combined functionality ........................................................................ 38
6 Use case .................................................................................................. 42
7 Conclusions ............................................................................................ 45
References .................................................................................................. 47
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>ID</td>
<td>Identification</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>LED</td>
<td>Light-emitting diode</td>
</tr>
<tr>
<td>OS</td>
<td>Operating system</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RPi</td>
<td>Raspberry Pi</td>
</tr>
<tr>
<td>STOMP</td>
<td>Simple Text Oriented Messaging Protocol</td>
</tr>
<tr>
<td>TPS</td>
<td>Toyota Production System</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>USR</td>
<td>User Requirements</td>
</tr>
</tbody>
</table>
List of Figures

Figure 2-1 Five principles of Lean thinking .......................................................... 4
Figure 2-2 Traditional Andon light stack [1, pp. 79] ........................................ 6
Figure 2-3 Distance between controller and receiver with maximum of possible jumps [8] ................................................................. 7
Figure 2-4 Network Route map of simple mesh network from Z-way Expert view ...... 8
Figure 2-5 Network Route map of more complicated mesh network from Z-way Expert view ........................................................................................................ 8
Figure 2-6 Raspberry Pi 3 Model B in plastic case without cover. RaZberry2 daughter card connected to GPIO ......................................................... 10
Figure 2-7 RaZberry2 RPi daughter card .................................................................. 11
Figure 3-1 High-level block diagram of Andon system ............................................. 16
Figure 3-2 Block diagram for communication from Application 1a to Application 2 ... 18
Figure 3-3 Block diagram for communication from Application 2 to Application 1a ... 19
Figure 3-4 Connection requirements for generating alert to Application 1b from Application 2 ..................................................................................... 20
Figure 3-5 Connection requirements for getting updates on alerts to Application 2 from Application 1b ............................................................... 21
Figure 4-1 Fibaro Universal Sensor connected to function as Z-wave supported push button. Powered by AC/DC Power supply ........................................ 24
Figure 4-2 Closed button with Z-wave support .......................................................... 24
Figure 5-1 High-level block diagram of andon-raspberry communications .......... 26
Figure 5-2 Database structure and schematics ......................................................... 28
Figure 5-3 Home page with alerts with different statuses ....................................... 34
Figure 5-4 Edit alert modal ...................................................................................... 35
Figure 5-5 Change light status modal .................................................................... 35
Figure 5-6 Block diagram of alert syncing ............................................................... 37
Figure 5-7 Block diagram of token retrieval ........................................................... 38
Figure 5-8 UML interaction diagram of the application launch ............................ 39
Figure 5-9 Button pressed UML ............................................................................ 40
Figure 5-10 UML of alert update to light change .................................................... 41
Figure 6-1 Light bulb configured to function as Lean Andon system .................. 42
Figure 6-2 Light bulb configured to function as Kanban system ............................ 43
Figure 6-3 Combination of andon-raspberry configurations ................................... 44
List of Tables
Table 2-1 RaZberry models and usable areas [11, pp. 12] .................................................. 12
Table 2-2 Common HTTP request methods ........................................................................ 12
Table 2-3 UML symbols and descriptions used in this thesis .............................................. 14
Table 3-1 Original user requirements for complete Andon system [30] .............................. 17
Table 3-2 Updated user requirements for Application 2 ......................................................... 17
Table 3-3 Explanation of the JSON keys and values [32] ...................................................... 19
Table 3-4 Explanation of the JSON keys and values [33] ...................................................... 21
Table 4-1 Component overview with prices ........................................................................... 25
Table 5-1 The important libraries used with short descriptions ............................................. 27
Table 5-2 Overview of the andon-raspberry website routes excluding API ......................... 36
Table 5-3 logger.js component log level descriptions ......................................................... 37
1 Introduction

Lean manufacturing is a manufacturing method concentrating on minimizing waste. In Lean, the waste can be found basically from anything not producing value or increasing the value of product being created. By reducing waste Lean allows organization to produce more and more with less and less. This allows the organization to concentrate on value generation and specifically generating more value as perceived by the customer.

Lean brings systematic basic principles and tools to assist in the waste reduction. One of the tools introduced by Lean is the usage of visual systems, and one such system is called Andon. The Andon is usually a traffic light colored light stack that provides visual signal of the situation on the production line; green – no issues, yellow – issue requiring attention and red – production stopped due to issue. Andon might also include Andon cord which enables production workers to inform supervisors of issues on production line by pulling the cord and changing the light color.

The Andon assists in the waste reduction of lean by lowering the response time to issues. The system is supposed to give quick overview of production visually and attract the attention of supervisors. When red lights are seen it is known that there are issues on the production line that needs supervisor assistance immediately or possible waste might be produced (loss of time or possibly scrapped products). Yellow meaning that attention is required but the priority is lower than red light and no immediate production stop is happening. Green gives information to supervisors that process is working as intended.

The purpose of this thesis was to develop a part of Andon system. The Andon system in this case consists of two applications: one application with webpage of Andon alerts and notification system to supervisors and the second application managing the Andon lights and Andon cord (button) to generate alerts. The development of latter is the main task of this thesis. This application will also communicate with the other application to keep Andon alerts and Andon light colors synchronized.

The lights and buttons shall be controlled wirelessly to allow easier production layout changes. The wireless technology chosen for this thesis was Z-wave technology. The Z-wave is commonly used in home-automation applications and has wide selection of different devices that can be used also for Andon system, including multi-colored Light-emitting diode (LED light bulbs and different sensors. The Application will be running on Raspberry Pi with Z-wave capable daughter card installed on Raspberry Pi’s GPIO pins.

The application developed in this thesis handles the communication with notification application. Also, it will handle the Z-wave device control. It shall provide User Interface for users to add new Z-wave devices to the application with configurable alerts. The system developed shall be able to cover all of the production which is located in multiple floors with minimum delay when changing the colors of the lights.

This thesis consists of seven chapters, first being Introduction. Chapter 2 provides background information on the Lean manufacturing and Z-wave. It also presents technologies and concepts used during this thesis. In Chapter 3 the
requirements for the system in development are discussed. Chapters 4 and 5 go into the actual implementation of the system, including hardware and software overview. Chapter 6 presents a use case on how the system is being used in the company. Finally, Chapter 7 summarizes the thesis, presents the most important observations and lists some further development tasks.
2 Background

This chapter provides background information required to understand this thesis. First the Lean and Andon are introduced and next the wireless Z-wave technology is explained. After these the Raspberry Pi (RPI) is detailed and following RPi, short explanations on multiple technologies and concepts used during this thesis are outlined.

2.1 Lean

Lean manufacturing originates from Toyota Production System (TPS). It is a production approach developed by Toyota after World War II. Lean manufacturing was developed for the need to manufacture more and more with less and less: less inventory, factory space, equipment and labor. [1, pp. 3-6] [2, pp. 3, 9] The initial goal of TPS was to maximize the production of cars with minimum resources but it also allowed improved quality of products. Nowadays Lean manufacturing is simply referenced as Lean. [1, pp. 3-6].

Lean may also be seen as a tool to minimize muda, waste. There are eight primary wastes [2, pp. 15, 355]:

- Defects - in products
- Overproduction - of goods not needed
- Inventories - of goods awaiting further processing or consumption
- Processing - unnecessary
- Movement - unnecessary movement of people
- Transport - unnecessary transport of goods
- Waiting - by employees for process equipment to finish its work or on an upstream activity
- Design of goods and services - which do not meet users' needs

By reducing these sources of the waste, more value can be generated to the product and customer.

This chapter explains the principles that the Lean is based around. After the principles, Lean tools, which are used to assist in the waste reduction are quickly, detailed and finally one Lean tool, visual systems, is explained in more detail due to Andon being one of the visual systems.

2.1.1 Lean principles

Lean thinking can be summarized to five basic principles, which can be visualized as seen in Figure 2-1. The first principle is the value as perceived by customer. Second principle is to determine what creates value (value stream), third is to re-organize production to create value in flow, fourth principle is to implement pull – only produce when customer has a need and the final principle is to continue improving and pursue perfection. [2, pp. 15-28]
The five basic Lean principles are explained in more detail below. When customer is mentioned it can be either internal customer, inside a company or even inside the process or external customer.

**Value**

Defining the value is important in Lean thinking because it is based on idea to produce more with less and to achieve this only the actions which generate more value are needed. In Lean thinking the value should always be defined from the view of customer. The production creates the value to the customer. [2, pp. 15-28]

**Value stream**

Value stream is all the specific actions required to bring product to the customer. Value stream can be divided in three main tasks:

1. Problem solving – Steps from the concept to the production launch
2. Information management – Steps from the order by customer to delivery of the finished product to customer
3. Physical transformation – Includes all steps from raw materials to how the final product ends to the customer.

After value stream has been identified it may be analyzed with Value Stream Analysis. Value stream analysis helps identifying the steps that: create value, do not create value but are required for the current production methods and the steps that do not create value and are not needed. By eliminating the unnecessary steps waste can be removed and production made more efficient. [2, pp. 15-28]

**Flow**
In Lean thinking the flow means that the product moves through whole production process without stopping. The value of the product is increased in each of the steps of process and no steps without added value should exist in the process. [2, pp. 15-28]

**Pull**
The pull means that the manufacturing of the product should not start before the customer creates an order. This lets customer to create pull of the product from the manufacturer and not push of the product from manufacturer to customer. [2, pp. 15-28]

**Perfection**
After all the four previous principles comes pursuing of the perfection. This means that the production can always be improved upon and the final product can be produced to be closer to the value customer expects. [2, pp. 15-28] Pursue of perfection can be seen in Figure 2-1 when five Lean basic principles form an ongoing circle.

### 2.1.2 Lean tools
Besides Lean thinking, Lean provides Lean tools to assist in identifying and eliminating the different types of wastes identified earlier. These tools are [1, pp. 70]:
1. Simplify
2. Streamline
3. Standardize
4. Use visual systems
5. Mistake-proof processes and product design
6. Synchronize
7. Collocate
8. Reduce changeover time

Two of these tools are important for this thesis: visual systems and synchronize. Synchronize brings one concept that is important tool in Lean and mentioned multiple times in this thesis, Kanban. Visual systems include Andon, visualization that is being developed in this thesis. Andon is introduced in more detail in chapter 2.1.3. Kanban will be briefly explained in the next paragraph.

Kanban means signal card or sign. It is commonly used to synchronize processing rates and functions as pull mechanism (Lean principle). Kanban gives visualization to production if specific step should be done: production started, material moved to workstation, material ordered from supplier and so on. When the visualization is not visible, the step should not be done. [1, pp. 111] Also, the Kanban does not have to be a sign or card, any object or signal that is capable of showing the signal is enough [3, pp. 177].

### 2.1.3 Visual systems
Visual systems reduce the waste by saving the supervisor’s time by directing the attention to where it is required, by reducing the downtime created by problems and reducing the amount of underutilized personnel and machinery by making the problems
more apparent. This allows for quicker problem solving which leads to less idle time for people and machines. [1, pp. 101] Visual systems also add additional transparency to production as they can be seen easily in production. One such visual system is called Andon and developing Andon system is the purpose of this thesis.

Andon is one of the visual control tools in Lean, usually displayed as different colored alarm lights. These are used to indicate problems or shortages of material in production. [3, pp. 173] Traditionally Andon colors are [1, pp. 79]:

- green – no problems
- yellow – problem requires attention
- red – production has stopped, and attention immediately needed

The lights are commonly displayed as Andon light stack which can be seen in Figure 2-2 below.

![Andon light stack](image)

**Figure 2-2 Traditional Andon light stack [1, pp. 79]**

Andon system might also include Andon cord. Andon cord allows the production line operators to alert management of possible issues by pulling the cord. The cord might be pulled when there is shortage of required parts or some production devices are broken. Music might also be included to Andon system. The music might be production location specific which allows the manager to immediately know where the issue has risen without the need to look for lights. [1, pp. 99]
2.2 Z-wave

Z-wave is a wireless communication technology developed for smart home applications. [4] In Europe the Z-wave operates in frequencies 868.4 MHz and 869.85 MHz. [5] It is cheap and has low power consumption. [6] The range is approximately 30m indoors (100m outdoors) between two devices but this can be extended due to mesh network topology. The mesh network allows Z-wave messages to be repeated over four repeater devices. This allows extension of the maximum range indoors to over 100m. [7]

Figure 2-3 presents direct route from sender to receiver with maximum number of jumps. In case of maximum or near maximum number of jumps, they might start adding additional delay to relaying the messages and cause stability issues. This is even more true if the route can only be routed through one possible way as seen in Figure 2-3. The message between the controller and receiver would not be delivered in the Figure 2-3 case if one of the repeaters would malfunction. By having more repeaters in the network, the network will get more flexible and robust [8] even if it gets more complicated. This is due to mesh network topology.

Figure 2-3 Distance between controller and receiver with maximum of possible jumps [8]

Figure 2-4 shows network map for a simple Z-wave mesh network. Device 1 is the controller and other devices are for devices capable of repeating the Z-wave messages. Even if node 1 cannot directly reach node 7 it can send the message by using properties of mesh network: the message from 1 gets repeated by 2 and finally reaches node 7. Also, from the same figure it can be seen that the nodes can reach each other by multiple ways, so even if one connection is disturbed the message can reach the destination, as example message from device 1 to device 2: 1 -> 2 or 1 -> 4 -> 9 -> 2.
The mesh networks can get multiple times more complicated as can be seen in Figure 2-5.

The devices in Z-wave network can be divided in three categories: controller, slave and slave with routing capability. The category where the device belongs depends on the device’s capabilities on knowledge of routing table and ability to send messages. Controller has access to complete routing table of the network, knows all devices in the network and if route exists it can communicate with messages with every device in the network. Slave can only reply to messages and has no information on the routing table where as slaves with repeater capabilities can also relay the messages between devices they have in their routing table. [8]
To have a network at least two nodes must exist. These nodes must have something in common with each other that allows them to communicate with each other. In Z-wave network this is called Home Identification (ID). Home ID is a common identification for all the nodes in the same logical Z-wave network. To distinguish the different nodes from each other in the same network unique identification is needed; in Z-wave network this is called Node ID. The devices in same Z-wave network cannot have same Node ID but they will have same Home ID. This allows the control of each of the nodes individually. Also, in different networks the Node IDs might be same, but Home IDs will differ as the networks are isolated. [8]

Adding and removing devices to Z-wave network have two specific concepts: inclusion and exclusion. Inclusion means the adding or installing the Z-wave device to the controller. To include device to Z-wave network a separate command to start inclusion procedure on the controller is required and after this a physical confirmation from the device itself. When a device is included to Z-wave network it is given a unique Node ID. And exclusion means the removal of Z-wave devices from the controller. To remove device separate command from controller and physical confirmation from the device itself are needed. [9]

2.2.1 Z-way

Z-way is possible smart home controller software with Z-wave support. Besides Z-wave support it supports EnOcean, WIFI, 433MHz. Z-way can be run on multiple different hardware setups such as Raspberry Pi, Windows and Debian. To give Z-way Z-wave support Z-WaveMe hardware such as UZB or RaZberry is required. [10] Z-way consists of seven blocks:

1. Z-way lib – Implements the Z-wave controller functions (closed source).
2. EnOcean lib – Implements EnOcean drivers (closed source).
3. Automation engine – Allows running automation logic and integrates wired and wireless third-party protocols (mostly open source).
5. Smart Home User Interface – Another interface, uses Automation Engine.
6. Cloud platform – Allows access to home network from outside the network.
7. Native Apps for iOS and Android – Allows usage from outside the network. Apps available from app stores for mobile devices. [10]

For the application developed in this thesis, the Z-way lib Application Program Interface (API) and Z-wave Expert User interface are used.

To communicate with Z-wave compatible transceiver hardware, Z-way uses Z-wave core. The Z-wave core used standard Sigma Design Serial API which is not public but is available to owners of Sigma Designs Development Kit. The Z-wave core can be accessed by using Z-wave Device API (zDev API). There are two device APIs available: Z-Wave API as JSON API and Z-wave API as C Library API. In this thesis the JSON API is used. This allows usage of websockets and Representational State Transfer (REST). [11, pp. 119]

The Z-wave Device API allows direct access to Z-wave network. The devices in the network are referenced by their node ids. The devices may also have
multiple instances of the same function (multiple LEDs in same light bulb). These are referenced as daughter objects in Z-wave Device API with instance ID. In case if only one instance exists, it will be referenced as instance ID 0. [11, pp. 120]

The device variables and commands in Z-wave devices are grouped as command classes. These variables and commands can be used and changed by directly accessing the Z-wave API. The API can be accessed by using URL (these do require authentication of the user)

http://ZwayIP:ZwayPORT/ZWaveAPI/Run/devices[x].instances[y].commandClasses[z].* [11, pp. 120]

To get updates on changes on Z-wave network, websocket connection may be opened to Z-way server. [11, pp. 144-145]

2.3 Raspberry Pi

Raspberry Pi is a small – credit card-sized, single board computer. RPi uses Linux as an OS (operating system). Four different basic models are currently available: 3, 2, 1 and Zero. All of the models also have different revisions on them. The models and revisions differ by their features such as having bluetooth and wlan support and Zero has been designed to be the lowest cost RPi available for smaller projects. [12]

The model used in this thesis is Raspberry Pi 3 Model B+ which has Quad Core 1.2GHz Broadcom BCM2837 64bit CPU, 1 gigabytes of RAM, 40-pin extended GPIO, full size HDMI and 100 base Ethernet [13]. In Figure 2-6 one of used Raspberry Pis is shown. Additional RaZberry daughter card has been connected to the 40-pin extended GPIO (using GPIO pins 1-10) of the Raspberry Pi and it is also fitted inside of third party plastic case.

![Figure 2-6 Raspberry Pi 3 Model B in plastic case without cover. RaZberry2 daughter card connected to GPIO.](image)

The Raspberry Pi Model 3 B was chosen most importantly due to the support for RaZberry card. As all RPi models, the RPi Model 3 B is also cheap when compared to
laptops or personal computers even when it is a high-end model of RPi. Also, the potential users for the system might not be familiar with Linux and good number of connectors make debugging easier for such users; external monitor, USB keyboard and USB mouse may be connected to Model 3 B. Linux based Raspbian is used as OS which is based on to provide graphical user interface.

### 2.4 RaZberry

RaZberry is a Raspberry Pi daughter card that allows RPi to be used as Z-wave based Smart Home Gateway [14] without blocking any USB ports. The RaZberry hosts Sigma Designs ZM5202 Z-wave transceiver module for Z-wave communication. [15] The RaZberry is connected to RPi’s GPIO connector pins GND, VCC (3.3V), Serial TX and Serial RX [11, pp. 10-11].

RaZberry2 daughter card can be seen in Figure 2-7. The board has connector for RPi (1, 2), reset button (3), two positions of additional antennas (4, 5) and status LEDs (6).

![Figure 2-7 RaZberry2 RPi daughter card](image)

The status LEDs are turned off during normal operation. The green LED will be turned on when data is being transmitted and red LED is used as an indicator for Z-wave inclusion and exclusion. The LEDs are also on during the self-test of the
RaZberry board. The self-test occurs when the RaZberry is powered on. The LEDs are supposed to turn off in a few seconds but if they remain lit it indicates a hardware issue and the RaZberry must be changed. [11, pp. 11-12]

RaZberry shield could be tuned to every frequency supported by Z-wave. However, an external antenna filter is used to protect the transceiver and the Z-wave from high energy emissions of nearby frequencies. This brings limitation to RaZberries and makes it limited to three areas. The frequency can only be changed within the areas that share the same antenna filter according to Table 2-1 below: [11, pp. 12]

<table>
<thead>
<tr>
<th>RaZberry model and limited frequencies</th>
<th>Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZMEEUZB2 (865 ... 869 MHz)</td>
<td>Europe [default]</td>
</tr>
<tr>
<td></td>
<td>India</td>
</tr>
<tr>
<td></td>
<td>Russia</td>
</tr>
<tr>
<td></td>
<td>PR China</td>
</tr>
<tr>
<td></td>
<td>RSA</td>
</tr>
<tr>
<td></td>
<td>Middle East</td>
</tr>
<tr>
<td>ZMEUUZB2 (908 ... 917 MHz)</td>
<td>All the Americas except Brazil and Peru</td>
</tr>
<tr>
<td></td>
<td>[default]</td>
</tr>
<tr>
<td></td>
<td>Israel</td>
</tr>
<tr>
<td>ZMEAUZB2 (919 ... 921 MHz)</td>
<td>Australia/New Zealand/Brazil/Peru/Malaysia</td>
</tr>
<tr>
<td></td>
<td>[default]</td>
</tr>
<tr>
<td></td>
<td>Hongkong</td>
</tr>
<tr>
<td></td>
<td>Japan/Taiwan</td>
</tr>
<tr>
<td></td>
<td>Korea</td>
</tr>
</tbody>
</table>

When selecting a daughter card the frequencies must be considered.

2.5 HTTP and HTTP request methods

Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia documents. HTTP was designed for communication between web browsers and servers but may also be used for other purposes. It follows a client-server model with the client opening the connection, making a request and then waiting until response is received.[16].

The request methods used each implement different semantic. The request methods indicate the desired action to be performed for a given resource. In Table 2-2 the most common HTTP request methods are explained [17].

<table>
<thead>
<tr>
<th>Request method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Get requests data from specified resource. Should only be used to retrieve data.</td>
</tr>
<tr>
<td>POST</td>
<td>Submit entity to specified resource and often cause change with the submitted entity.</td>
</tr>
</tbody>
</table>
PUT | Replaces all current representations of the target resource with the request payload. Calling this multiple times has always the same effect. When calling POST multiple times, it might have additional effects. [18]

DELETE | Deletes specified resource.

GET and POST request methods are used in the application developed in this thesis.

### 2.6 Websocket and STOMP

Websockets make it possible to open two-way communication between server and client [19, 20]. The client must open websocket connection to server and after the connection has been established the client can receive messages from server (client may also be able to send messages if allowed). Ping-pong frame may be used to verify that the end point is till responsive or to serve as a keep-alive method [19], pp 37. After the websocket connection is no longer needed it may be closed.

Simple Text Oriented Messaging Protocol (STOMP) is a frame-based protocol. The frame consists of command, set of headers and optional body. [21] It defines an interoperable wire format which allows any of the available STOMP clients to communicate with any STOMP message brokers. This allows easy messaging interoperability between different programming languages and platforms. [22] When used with websockets the STOMP websocket connection provides few specific frames such as [22]:

- connect
- disconnect
- subscribe
- unsubscribe
- send

STOMP may be used without websocket but in this thesis it is needed for forming websocket connection.

### 2.7 JavaScript Object Notation

JavaScript Object Notation (JSON) is a text-syntax for storing and exchanging data. [23] The JSON objects are key/value pairs [24] and the values in JSON must be object, array, number, string or one of these literal names false, null or true [25] Storing the data in JavaScript object allows easy access to the values behind keys. Below is a simple example of JSON object. The value of data.message in example JSON object can be accessed by data.message and will return “text”.

```json
  {
    "data": {
      "message": "text",
      "number": 1
    },
    "key": "value",
  }
```
When sending data to server from client the JavaScript object can be converted to JSON by using JavaScript built in function JSON.stringify(JavaScriptObject) and back to JavaScript with function JSON.parse(JSONObject). [23]

### 2.8 Unified Modeling Language

Unified Model Language (UML) is a language for visualizing, specifying, constructing and documenting software system. [26] UML provides a way to model the software or functionalities of the software in simplified way. It also provides visualization for the software. In this thesis the UML is used to provide diagrams to explain the functionality and flow of the application. In the Table 2-3 the used UML symbols used in this thesis are described:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Symbol description</th>
<th>Functionality description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Dotted vertical line" /></td>
<td>Dotted vertical line</td>
<td>Lifeline of the process.</td>
</tr>
<tr>
<td><img src="image" alt="Vertical rectangle on top of dotted vertical line" /></td>
<td>Vertical rectangle on top of dotted vertical line</td>
<td>Operation</td>
</tr>
<tr>
<td><img src="image" alt="Rectangle with text" /></td>
<td>Rectangle with text</td>
<td>Process</td>
</tr>
<tr>
<td><img src="image" alt="Vertical continuous arrow" /></td>
<td>Vertical continuous arrow</td>
<td>Message</td>
</tr>
<tr>
<td><img src="image" alt="Vertical dotted arrow" /></td>
<td>Vertical dotted arrow</td>
<td>Response to message</td>
</tr>
<tr>
<td><img src="image" alt="Stick figurine" /></td>
<td>Stick figurine</td>
<td>Actor</td>
</tr>
</tbody>
</table>

The symbols used are based on the UML but are not the same. Also the UML diagrams in this thesis make no difference between the different arrowheads even if they might have different meanings in normal UML diagrams.
2.9 Node.js and promises

Node.js is a JavaScript run-time environment. It has been designed to build scalable network applications due to its asynchronous and event driven nature. [27] Node.js was chosen as the programming language for the application developed in this thesis due to JavaScript’s increasing popularity as programming language. Node.js can also be ran on multiple platforms and allows running JavaScript on the server [28].

To provide synchronous functionality to Node.js JavaScript promises may be used. The idea of promise is to represent the value of asynchronous operation. This lets asynchronous actions to return values like synchronous actions but instead of returning the final value the promise returns promise that the value will be supplied at some point in future. [29] A promise is always in one of three states [29]:

- pending: initial state, operation not yet fulfilled/rejected
- fulfilled operation completed successfully
- rejected: operation failed

The application developed in this thesis is mostly written using promised functions. This is due to the application requiring synchronous functionality on most of the operations it performs.


3 Requirements

Few production lines were already using Andons. These Andons have colored light bulbs and the colors can be changed by using remote control. They only give the visual information on the production line and no additional information is delivered. Purpose of this thesis is to update this system to completely new Andon system. This system will include two applications; other will be sending out notifications to users and other will be controlling the Andon devices, lights and buttons. The one controlling Andon devices, is developed in this thesis and the application will be referenced as andon-raspberry.

This chapter explains the requirements for the Andon system. The system is divided in two different applications and the purpose of this thesis is to develop the application 2. This application handles the wireless Z-wave communication and communication with application 1. The chapter has been divided in three different parts. First part has the general requirements for Application 2. Second and third part give more specific requirements that are required by Application 1. The application 1 was changed during the development of Application 2. The requirements from both application 1s are explained here but after this chapter only the Application 1b will be discussed as that is the final version of the Application 1 that will be used in production. All later mentions of Application 1 after Chapter 3 will reference to Application 1b.

3.1 General requirements

The company had decided to start implementing new Andon system to be used by production. The system was split between two separate applications:

- Application 1: Alert application which would handle sending notifications to users
- Application 2: Communicating with Application 1 and handling the lights and buttons.

Figure 3-1 displays high level block diagram between the two systems. Both systems will have their own front and back end layers. The purpose of this thesis is to develop the Application 2.

![Figure 3-1 High-level block diagram of Andon system](image)

The user requirements for the Application 2 came from the company. The original USR can be seen in Table 3-1. The requirements outline some technical
specifications such as light colors, wireless technology, delay and for of communication between the two applications. [30]

Table 3-1 Original user requirements for complete Andon system [30]

<table>
<thead>
<tr>
<th>User requirement (USR)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USR 1</td>
<td>Robust, wireless buttons (e.g. Z-wave – range ca. 30m)</td>
</tr>
<tr>
<td>USR 2</td>
<td>Wireless signal light poles at least with green, yellow and red colors</td>
</tr>
<tr>
<td>USR 3</td>
<td>Pressing a button shall change the color of Andon light to red. The Andon shall change color under 2s from pressing the button.</td>
</tr>
<tr>
<td>USR 4*</td>
<td>Pressing a button shall write agreed interface message (.csv or JSON) to agreed address</td>
</tr>
<tr>
<td>USR 5*</td>
<td>HTTP protocol for integration.</td>
</tr>
<tr>
<td>USR 6</td>
<td>Read the interface messages e.g. in 5s interval from agreed folder. Recognize from filename (Loc+line) that the message is for it (that hub).</td>
</tr>
<tr>
<td>USR 7</td>
<td>Change light color according to status in interface message. If many open alerts for same LightID at same time, Andon Light color is controlled according to most severe status (severity order: red, yellow, green).</td>
</tr>
<tr>
<td>USR 8</td>
<td>The buttons shall be configurable as which Andon(s) (light(s)) each button controls (Browser based setup form).</td>
</tr>
<tr>
<td>USR 9</td>
<td>The system shall be able to handle a minimum of 500 buttons and 100 Andon lights.</td>
</tr>
</tbody>
</table>

Based on the discussion with the company it was decided that the Application 2 would be implemented on Raspberry Pi and by using Z-wave as the wireless technology. Application 1 would be developed by different developer. These decisions were made to allow easier layout changes and ensure easy availability of the products needed as Z-wave devices and Raspberry Pis are widely available. [31]

The requirements were updated during the development of Application 2 due to complete rework of Application 1. The updated requirements can be seen in Table 3-2. The most major changes were changes to JSON format and to the communication protocol between the applications. This rework of Application 1 also caused complete rewrite to Application 2’s code.

Table 3-2 Updated user requirements for Application 2

<table>
<thead>
<tr>
<th>User requirement (USR)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USR 1</td>
<td>Robust, wireless buttons (e.g. Z-wave – range ca. 30m)</td>
</tr>
<tr>
<td>USR 2</td>
<td>Wireless LED lights with at least green, yellow, red and blue colors.</td>
</tr>
<tr>
<td>USR 3</td>
<td>Pressing a button shall change the color of Andon light to red. The Andon shall change color under 2s from pressing the button.</td>
</tr>
</tbody>
</table>
Pressing button shall generate POST request to Application 1b to initiate alert.

Websocket protocol for integration.

Change light color according to status in interface message. If many open alerts for same LightID at same time, Andon Light color is controlled according to most severe status (severity order: red, yellow, green). Blue lights are used for information.

The buttons shall be configurable as which Andon light is controlled by which button.

One Z-way server is capable of handling 255 Z-wave devices. These can be either lights or buttons.

### 3.2 Requirements to Application 2 from Application 1a

This is the first iteration of Application 1.

To communicate with Application 1a, the Application 2 shall host web server that is able to respond to JSON formatted HTTP POST requests. When Application 1a sends POST request to Application 2 the Application 2 must respond with response status code 200 and string used as body see Figure 3-2 for block diagram. [32]

![Figure 3-2 Block diagram for communication from Application 1a to Application 2](image)

The Application 2 shall also be able to send JSON formatted HTTP POST requests to generate new alerts. The Application 1a will respond with JSON formatted response that has information on the alert if the generation was successful. See Figure 3-3 for block diagram.
Both JSON POST request have specific keys and value types they must follow. Below is an example of the POST request from Application 2 to generate alert and the response generated by Application 1a for successful request.

Example POST request generated by Application 2 to Application 1a:

```json
{
    "LocationName": "location",
    "LineName": "line",
    "EventType": "DEFAULT",
    "SubEventType": "DEFAULT",
    "Status": "New",
    "ApiKey": "secretkey",
    "AppId": "secretid"
}
```

Response to successful request above:

```json
{
    "Timestamp": 1473230699,
    "EventID": 1,
    "LocationName": "location",
    "LineName": "line",
    "EventType": "DEFAULT",
    "SubEventType": "DEFAULT",
    "Status": "EventCreated"
}
```

When the Application 1a sends status updates to Application 2, the same format of HTTP request is used as above. Explanation for the key-value pairs can be seen in Table 3-3.

**Table 3-3 Explanation of the JSON keys and values [32]**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>Timestamp as long format</td>
</tr>
<tr>
<td>EventID</td>
<td>Unique long value used to identify alerts</td>
</tr>
<tr>
<td>LocationName</td>
<td>String value used to identify alert</td>
</tr>
<tr>
<td>LineName</td>
<td>String value used to identify alert</td>
</tr>
<tr>
<td>EventType</td>
<td>String value used to identify alert</td>
</tr>
<tr>
<td>SubEventType</td>
<td>String value used to identify alert</td>
</tr>
</tbody>
</table>
Status

Value which has the status of the alert. Possible response values are EventCreated, EventAcknowledged or EventClosed. When creating new alerts New shall be used as value.

ApiKey

Unique value used to authenticate to Application 1

AppId

Unique value used to authenticate to Application 1

The Application 1a was replaced by Application 1b during this thesis. The Application 1b was chosen to be used due to additional support of development team and it being more user-friendly.

3.3 Requirements to Application 2 from Application 1b

This is the second and final iteration of Application 1.

The requirements from Application 1b to Application 2 are more complex; to get updates to alerts websocket connection must be opened and to create alerts additional authorization token is needed before accessing Application 1b. [33] Figure 3-4 displays high level block diagram of the communication required to create new alert to Application 1b.

Application 2 can also generate HTTP POST requests to Application 1b to generate different requests besides create new alerts, such as update existing alerts and retrieve information on all open alerts. All these POST requests require similar communication with tokens between Application 2 and Application 1. [33]

To receive updates from Application 1b Application 2 must form websocket listener connection to specified address. The address depends on what updates are listened for. Figure 3-5 displays high level block diagram of the connections required to receive updates as websocket messages.
Below is an example JSON object sent by Application 1b as websocket message when alert is updated. The object has been simplified for this thesis to only display purposeful information. Object includes timestamp of the alert update, string type of the update being performed in the websocket message and updated alert object. [33]

```json
{
    "type": "Resolve",
    "timestamp": 1473230699,
    "alert": {
        "id": 1,
        "alertDefinition": {
            "id": 1,
            "alertType": {
                "id": 1,
                "name": "Issue",
                "description": "Issue"
            },
            "locationId": "123-abc-456-def",
            "name": "Issue",
            "description": "Issue found",
        },
        "status": "Resolved"
    }
}
```

The Table 3-4 explains the key-value pairs in the JSON above. The keys in the table are in same order as they appear in the JSON object.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String value of the type of websocket message. Possible values are Initiate, Acknowledge, Unacknowledge, Resolve, Pause, Unpause, Delete, Sla, Update and Comment.</td>
</tr>
<tr>
<td>timestamp</td>
<td>Long formatted timestamp</td>
</tr>
<tr>
<td>alert</td>
<td>Object that holds all information on the alert</td>
</tr>
<tr>
<td>id</td>
<td>Unique identification number for the alert</td>
</tr>
</tbody>
</table>
Also, for Application 2 to be able to retrieve authorization tokens needed for POST requests it must be registered to token service. In addition to this to generate new alerts the Application 2 must have unique single sign on account registered for it. Both must only be done once during the development. [33]

4 Andon-raspberry hardware

The hardware requirements for this system were to be easily available, mostly wireless and if possible cheap. All the components used during this thesis are explained below. All the prices are rounded to closest euro.

4.1 RaZberry

RaZberry is a daughter card for RPi which allows RPi to communicate wirelessly with Z-wave devices. There are two models of RaZberry and the RaZberry2 model boards were chosen due to better availability and having better antenna for improved communication. The price for RaZberry2 board is approximately 61€.

4.2 Raspberry Pi with RaZberry support

The Raspberry Pi Model 3 B+ was chosen because it is the newest model of the Raspberry Pi and has best performance and the availability should be best. The price is also low when compared to laptops and personal computers. RaZberry will be connected to RPi GPIO and together they will form a small computer capable of Z-wave communication.

In addition to Raspberry Pi 3 Model B+ additional components are needed: case, microSD card for operating system and power supply for Raspberry Pi. These were bought in one package that is commonly called Raspberry Pi Starter Package. The package did include: Raspberry Pi 3 Model B+, HDMI cable, RJ45 ethernet cable, power supply, microSD card with NOOBS installed and plastic case. The package did cost 95€.
4.3 Multicolored Z-wave LED light bulbs

Due to the color requirements of Andon system the Z-wave controlled light bulbs must support at least red, yellow and green. The light bulbs shall also be connected to E27 socket to allow easy connect ability and work in correct Z-wave frequency (in this case 868MHz). Two models of light bulbs were used: Z-Wave Plus Aeotec LED Bulb and Z-Wave Hank RGB Bulb. Aeotec bulb costs around 67€ and Hank bulb 59€.

Both LED bulbs have RGBWW (Red, Green, Blue, Warm White and cold White) LEDs inside and can be controlled with same Z-wave commands. Both LED bulbs support 16 million colors. It should be possible to use other Z-wave controlled LED bulbs if they function with same commands and have all required colors. ZipaBox RGBW Bulb 2 was tried out during this thesis, but it had slower response time and slowed down the Z-wave network due to additional communication requests and as such is not recommended to be used in this system.

4.4 Light sockets for the bulbs

Any light socket supporting the chosen light bulbs may be used (E27 socket in this thesis). Hanging light sockets were used for simple implementation to the roof of production floors. The light sockets used were ordered online and did cost 7€ each.

4.5 Buttons to create alerts

The buttons to create alerts were first supposed to be wireless buttons but these caused issues in the Z-wave network and had additional delays. This was caused by the wireless buttons going into standby mode after not being used for some time. To avoid this buttons were changed to ones with corded power supply. This allowed the usage of wireless Z-wave technology for communication without the issue with standby.

The buttons were built by using Z-Wave Fibaro Universal Sensor. The sensor was connected to push-button switch and to connector for supply voltage (9-30V DC voltage is needed to power the sensor [34]. These were installed inside plastic case to protect the assembly, see Figure 4-1. The sensor was connected to button and dc socket according to the Fibaro Universal Sensor datasheet [34].
The button in center of Fibaro sensor is a maintenance button used to include and exclude the device to Z-wave network. Figure 4-2 shows the finished button after the case has been closed.

The costs of the components used to build these buttons are detailed below:

- The Fibaro Universal sensor – 34€
- Plastic case – 11€
- Push-button – 2€
- DC panel socket – 1€
- AC/DC 9V power supply – 7€
4.6 Range extenders for Z-wave network

In some locations additional range extenders were used to either allow longer jumps between Z-wave devices or to improve the Z-wave network robustness. The range extenders that were used were Z-Wave Plus Aeotec Range Extender 6 – EU with cost of 44€. Other extenders may also be used but as for other Z-wave devices, the support for correct frequency must be observed.

4.7 Component overview

In Table 4-1 all the components used are collected for easier overview. All the prices are rounded to closest euro.

<table>
<thead>
<tr>
<th>Component</th>
<th>Price [€]</th>
</tr>
</thead>
<tbody>
<tr>
<td>RaZberry2 daughter card for Z-wave control</td>
<td>61</td>
</tr>
<tr>
<td>Raspberry Pi Model 3B+</td>
<td></td>
</tr>
<tr>
<td>Micro SD card for Raspberry Pi</td>
<td></td>
</tr>
<tr>
<td>Plastic case for Raspberry Pi</td>
<td></td>
</tr>
<tr>
<td>Raspberry Pi power supply</td>
<td>95</td>
</tr>
<tr>
<td>RJ45 Ethernet cable for Raspberry Pi</td>
<td></td>
</tr>
<tr>
<td>Z-Wave Plus Aeotec LED Bulb</td>
<td>69</td>
</tr>
<tr>
<td>Z-Wave Hank RGB Bulb</td>
<td>59</td>
</tr>
<tr>
<td>Light socket (E27)</td>
<td>7</td>
</tr>
<tr>
<td>Fibaro universal sensor</td>
<td>34</td>
</tr>
<tr>
<td>Case</td>
<td>11</td>
</tr>
<tr>
<td>Push-button</td>
<td>2</td>
</tr>
<tr>
<td>DC panel socket</td>
<td>1</td>
</tr>
<tr>
<td>AC/DC 9V power supply</td>
<td>7</td>
</tr>
<tr>
<td>Additional Range extender</td>
<td>44</td>
</tr>
</tbody>
</table>

All the components and hardware were ordered online. When choosing the components and where the parts were ordered it was observed that the selected parts were readily available also in future. The parts demonstrated in this chapter were used during the system development in this thesis, but other similar parts should also be applicable.
5 Andon-raspberry software

In this chapter the software for andon-raspberry is discussed. First short overview of the complete application is given. Next the important libraries are explained. Third the database structure and schematics are defined. After these three the communication between Application 1 and Z-way server are looked into. After that the utilities are explained and finally the andon-raspberry is investigated as a complete application with all the previous parts.

5.1 Overview

Andon-raspberry application has been divided in three building blocks: communication with application 1, communication with Z-way server and the user interface, which includes website and API. In addition to these there are multiple important utilities that are used in most of the building blocks. Figure 5-1 presents high level block diagram of andon-raspberry communications.

![High-level block diagram of andon-raspberry communications](image)

Andon-raspberry forms websocket connections to both Application 1 and Z-way server to receive updates of alert and Z-wave device changes. POST requests are used to initiate alerts and manage the colors of the lights.
5.2 Libraries
The most important libraries used for andon-raspberry are presented in Table 5-1 below with short descriptions.

<table>
<thead>
<tr>
<th>Library</th>
<th>Description</th>
<th>Available from</th>
</tr>
</thead>
<tbody>
<tr>
<td>body-parser</td>
<td>Used to assist with JSON response parsing from HTTP requests</td>
<td><a href="https://github.com/expressjs/body-parser">https://github.com/expressjs/body-parser</a></td>
</tr>
<tr>
<td>express</td>
<td>Used to host the HTTP server for the andon-raspberry website and API.</td>
<td><a href="https://github.com/expressjs/express">https://github.com/expressjs/express</a></td>
</tr>
<tr>
<td>nedb</td>
<td>Used as the database. Lightweight and fast.</td>
<td><a href="https://github.com/louischatriot/nedb">https://github.com/louischatriot/nedb</a></td>
</tr>
<tr>
<td>camo</td>
<td>Modelling for the nedb database to help with managing the database entries. Provides additional functions and validation.</td>
<td><a href="https://github.com/scottwrobinson/camo">https://github.com/scottwrobinson/camo</a></td>
</tr>
<tr>
<td>schedule</td>
<td>Used to create cron formatted tasks. In this application used to run syncing function in time-based schedule.</td>
<td><a href="https://github.com/node-schedule/node-schedule">https://github.com/node-schedule/node-schedule</a></td>
</tr>
<tr>
<td>faye-websocket</td>
<td>The websocket library for this application. Chosen due to easy support of proxy servers.</td>
<td><a href="https://github.com/faye/faye-websocket-node">https://github.com/faye/faye-websocket-node</a></td>
</tr>
<tr>
<td>webstomp</td>
<td>stomp connection is required to create websocket connection to Application 1.</td>
<td><a href="https://github.com/JSteunou/webstomp-client">https://github.com/JSteunou/webstomp-client</a></td>
</tr>
<tr>
<td>Winston</td>
<td>Logging library</td>
<td><a href="https://github.com/winstonjs/winston">https://github.com/winstonjs/winston</a></td>
</tr>
<tr>
<td>Library</td>
<td>Description</td>
<td>Website</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>passport</td>
<td>Handling hashing the andon-raspberry website user account passwords.</td>
<td><a href="https://github.com/jaredhanson/passport">https://github.com/jaredhanson/passport</a></td>
</tr>
<tr>
<td>underscore</td>
<td>Provides multiple useful functions. In this application mostly used assist in array and object handling.</td>
<td><a href="https://underscorejs.org/">https://underscorejs.org/</a></td>
</tr>
<tr>
<td>request-promise</td>
<td>Used to create promisified HTTP requests.</td>
<td><a href="https://github.com/request/request-promise">https://github.com/request/request-promise</a></td>
</tr>
</tbody>
</table>

### 5.3 Database and data schematics

The database is built by using nedb as database with camo as schematic language. Schematics are used to ensure that the data structures are required to have specific key-value pairs filled. In Figure 5-2 all the database schematics are presented with their dependencies. *user* and *token* objects are not linked to any other database and are used individually.

![Figure 5-2 Database structure and schematics](image)

Below the database objects are explained in more detail.

#### 5.3.1 ‘token’ database entry

![Object:token](image)
Only one entry in ‘token’ database exists at the time. The entry consists of access_token and timestamp entries. The access_token is only valid for set amount of time by comparing the timestamps the decision will be made if new token must be retrieved from token server. access_token is used when making any request to Application 1.

5.3.2 ‘user’ database entry

<table>
<thead>
<tr>
<th>Object: user</th>
</tr>
</thead>
<tbody>
<tr>
<td>username= string, required, unique</td>
</tr>
<tr>
<td>password=string, required</td>
</tr>
</tbody>
</table>

The ‘user’ is used to save the user accounts used to login to the andon-raspberry website. ‘username’ must be unique. ‘password’ will be saved as string that has been salted from clear text format to add small layer of security.

5.3.3 ‘alert’ database entry

<table>
<thead>
<tr>
<th>Object: alert</th>
</tr>
</thead>
<tbody>
<tr>
<td>alertId= number, required</td>
</tr>
<tr>
<td>alertDescription= array</td>
</tr>
<tr>
<td>type=number, required, possible values 0,1,2</td>
</tr>
</tbody>
</table>

This object consists of alertId, alertDescription and type. ‘alertId’ is an id received from Application 1 when alert gets initiated. The alertId should be unique, but it is not required in the schematic due to handling done in Application 1’s end. This is done to identify alerts between each other. ‘alertDescription’ is stored as an array due to splitting done to before saving the data to database. This is explained in chapter 5.1. This helps between the comparison of ‘alert’ and ‘andon’ database objects. ‘type’ is stored as numeric values presenting the statutes of alerts 0 – Resolved, 1 – Acknowledged and 2 – Initialized. The alert type is received in string format from Application 1 and changed to numeric value before saving to database.
5.3.4 ‘andon’ database entry

<table>
<thead>
<tr>
<th>Object: andon</th>
</tr>
</thead>
<tbody>
<tr>
<td>lightId=number, required</td>
</tr>
<tr>
<td>deviceType=string, required, default: 'AeotecRGBv1'</td>
</tr>
<tr>
<td>alertDescriptions=array</td>
</tr>
<tr>
<td>used=boolean, required, default: true</td>
</tr>
<tr>
<td>red=string, required, default: '[0,0,99,0,0]'</td>
</tr>
<tr>
<td>yellow=string, required, default: '[0,0,50,30,0]'</td>
</tr>
<tr>
<td>green=string, required, default: '[0,0,0,5,0]'</td>
</tr>
<tr>
<td>blue=string, required, default: '[0,0,0,0,1]'</td>
</tr>
</tbody>
</table>

‘andon’ database entry is the most used entry in the database with ‘alert’. This includes all the necessary information on the Z-wave controlled multicolor LED light bulbs. ‘lightId’ is the nodeId of the light being controlled. One light can be controlled with multiple alertDescriptions and that is why ‘alertDescriptions’ is saved as an array. See chapter 5.1 for more information on this. ‘used’ may be used to enable/disable the light. ‘deviceType’ left for future use to allow other lights be used.

The lamps have four different colors that can be configured. These are saved as strings and the format is very specific. These strings are directly used in the link generation which is used to control the light. The format is ‘[W,W,R,G,B]’ and Ws are white LEDs which must be 0 if any other light will be used, R is red, G is green and B is blue. The maximum allowed value is 255 but the same result will be achieved by using 100. By changing the value between 0-100 the brightness of the light can be lowered or increased. The colors are bound to alert statuses: Red value is for status 2, yellow for status 1 and green for status 0. The setting blue may be used as information color if needed.

5.3.5 ‘button’ database entry

<table>
<thead>
<tr>
<th>Object: button</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodeId= number, required</td>
</tr>
<tr>
<td>alertDefinitionId= number, required</td>
</tr>
<tr>
<td>timestamp= number, required, default: 0</td>
</tr>
<tr>
<td>comment= string, required, default: &quot;Andon btn pressed!&quot;</td>
</tr>
<tr>
<td>deviceType= string, required, default: 'fibaroBinarySensor'</td>
</tr>
<tr>
<td>used= boolean, required, default: true</td>
</tr>
</tbody>
</table>

‘button’ entries are used to save information of the buttons used to generate alerts. ‘nodeId’ is used to retrieve additional data based on the button pressed from the
database. ‘alertDefinitionId’ is an identifier used to generate alerts in Application 1 with ‘comment’. One button can only generate one kind of alerts based on the alertDefinitionId and comment. ‘deviceType’ is for future use to support different kind of buttons that might require different handlers in code. ‘used’ can be used to disable the button from use.

5.3.6 ‘zwave’ database entry

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodeId</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>cmdClass</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>used</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>deviceType</td>
<td>string</td>
<td>'default'</td>
</tr>
</tbody>
</table>

‘zwave’ schematic is used to save information of the Z-wave devices added to the andon-raspberry. This object holds the unique ‘nodeId’ of the devices, ‘cmdClass’ which can be used to determine the type of the device (38 command class for light and 48 for button), ‘used’ information if the device is in use and ‘deviceType’ that is not currently being used but can be used to develop different routines for different devices.
5.4 Communication with Application 1

When andon-raspberry communicates with Application 1 two protocols are used: HTTP POST requests and websockets. Websocket listener is used to receive all information about the alerts happening in specified location and HTTP POST requests are used to retrieve access tokens and initialize alerts with andon-raspberry to Application 1.

Making requests to Application 1 requires the usage of access_tokens. These tokens are retrieved from token server by using application specific authentication token. The access_token is valid for 7200 seconds and may be used multiple times. After valid access_token is available, requests can be sent to Application 1. When initializing alerts POST request generated to specific url, has the authorization token included and JSON type body with alert comment and user generating the alert. Below is an example Uniform Resource Locator (URL) used to initialize alert:

https://example.com/api/xx/alertDefins/123/alerts

Alert definition ID is retrieved from the ‘button’ database entry depending on which button has been pressed. Same for the ‘comment’ that is in the JSON body of the POST request. The alert definition ID defines the alert which is generated.

Before forming websocket connection, andon-raspberry will attempt GET request to the Application 1 to verify that the application is running. If GET request fails, it will be attempted again once a minute. After the GET requests is successful, opening the websocket connection will continue. Ping-pong of 30 seconds is used to keep the websocket listener connection open.

After the websocket listener connection is established to chosen location, andon-raspberry start waiting for websocket messages. The websocket messages are in JSON format described in chapter 3.3. The message will be parsed and passed to next handlers to update light colors if needed. If websocket connection gets closed due to an error, connection will be tried to be reopened every 10 seconds.

5.5 Communication with Z-way server

The communication between Z-way server is divided in two different blocks: GET requests to control Z-wave devices and websocket listener to get Z-wave device updates. GET requests are used to change the LED bulb colors and websocket listener to get information on button presses, inclusion and exclusion of Z-wave devices.

The Z-wave light colors are changed by creating GET request with request-promise library. The GET requests have timeout set to 2000ms to prevent the application from getting stuck for too long. The GET requests require username, password and URL. The username and password are same as used to access Z-way server.

The URL is generated based on the requirements in chapter 2.2.1. The URL is generated based on the Z-way server IP, port, light’s node ID and the color
string from database. Below is an example URL to change light (node ID 3) color to red:

```
http://example:8083/ZWaveAPI/Run/devices[3].instances[0].
```

CommandClasses[51].SetMultiple([0,1,2,3,4],[0,0,99,0,0],0)

CommandClasses[51] is command class for SwitchColor. It allows control of multicolored LED bulbs and LED strips. [11], pp 202] SetMultiple(array, array, 0) function for this command class used to control multiple LEDs in the light at the same time. This command will send five requests to Z-wave device, one for each of the LEDs. The first array includes the index information of the LEDs and second array has the brightness values for each of the LEDs. If 0 or 1 index are turned on, the lights will only show white light. [35] Change time is used to control the change speed of the color and when configured to zero the change speed is instant.

When andon-raspberry is started it will create websocket listener connection to Z-wave server with faye-websocket. The connection address and port are same as for Z-way website with ‘ws://’ as protocol instead of ‘http://’. This websocket listener is used to listen for updates in Z-wave network. 60 second ping-pong verifies that the websocket connection is alive. If close message is received from the websocket, the connection will be tried to reopen every 10 seconds. This 10 second reconnection has no timeout and will be retried until andon-raspberry is closed as this is fundamental connection for the application to work as intended.

The Z-way will send websocket messages about the Z-wave traffic in JSON format (changed to JSON object with JSON.parse()) which is parsed for useful information. The information listened for are inclusions, exclusions and changes to button statuses. When listening to any of these nodeId and cmdClass are parsed for. Inclusion and exclusion data is used to add and remove devices from the ‘zwave’ database. When adding the device to andon-raspberry ‘cmdClass’ and ‘nodeId’ are saved. When removing the device from database, ‘nodeId’ is used to find the correct device which will be removed.

When button is pressed the data.type must be ‘device-OnOff’ and data.message.l ‘off’. Off is triggered when button latch is released. This trigger is used together with timestamp comparison to prevent creation of duplicate alerts. Without these checks pressing the button could sometimes create to alerts if button is pressed twice in quick succession.
5.6 Website and API

The website is created with express.js library using pug as template language. It has been divided to controllers and routes. Each of the six sections (/alerts, /home, /andon, /button, /zwave and /api) have their own controllers and routes. Routes handle the routing and calling of the needed controllers. The controllers handle all the function calls and updating the andon-raspberry databases. The usage of most of functionalities of the website has been protected with password. The ones that do not require password have been rate-limited to prevent repeated requests. The purpose of the website is to give graphical user interface for users to configure Z-wave devices function with Application 1 and update alerts on andon-raspberry’s end.

Home section can be used to make quick modifications to the andon-raspberry database and to get overview of the alerts open in andon-raspberry. Figure 5-3 displays how the website displays alerts on the homepage. The user can edit alerts from this view by pressing the ‘Edit’ button which will open modal displayed in Figure 5-4. By pressing the numbered button on ‘Light’ column the user may control the light color and on/off status, see Figure 5-5.

<table>
<thead>
<tr>
<th>Light</th>
<th>Alerts in Andon database</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Light 9" /></td>
<td>AlertId</td>
</tr>
<tr>
<td><img src="image" alt="Light 6" /></td>
<td>AlertId</td>
</tr>
<tr>
<td><img src="image" alt="Light 4" /></td>
<td>AlertId</td>
</tr>
</tbody>
</table>

![Figure 5-3 Home page with alerts with different statuses](image)

The alert rows in Figure 5-3 are colored according to the alert status. The light button is colored according to the highest status of alerts for that light. This page can be used to get quick overview of the light colors and made quick updates if needed.
By updating the alerts with Edit alert in Figure 5-5 the light color will also be changed and this is the preferred way to update the light colors as it keeps the database up-to-date.

The light modal seen in Figure 5-5 should be mostly used to test the color changing of the lights. It can be used as shortcut to access site to edit light settings. In rare cases
when the databases between andon-raspberry and Application 1 do match and the light color is still incorrect this modal can be used to correct the light color.

The Table 5-2 gives quick overview of the available routes in the andon-raspberry website. These routes may be used to manage the andon-raspberry application. The API is excluded from this table.

<table>
<thead>
<tr>
<th>Section</th>
<th>Routes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/home</td>
<td>N/A</td>
<td>Gives quick overview of the alert statues and provides quick access with modals to make small changes to light colors and edit alerts</td>
</tr>
<tr>
<td>/alerts /create</td>
<td>Allows full control of all functionalities of the alerts in andon-raspberry. The alerts are edited with similar modal as seen in Figure 5-4. /alertsopen may be accessed without password. Sync may be used to trigger alert syncing manually.</td>
<td></td>
</tr>
<tr>
<td>/andons /create</td>
<td>Allows user to view all the available Andons (lights), delete and edit them. /create may be used to create new Andons</td>
<td></td>
</tr>
<tr>
<td>/buttons /create</td>
<td>Allows user to view all the available buttons, delete and edit them. /create may be used to create new buttons.</td>
<td></td>
</tr>
<tr>
<td>/zwave /create</td>
<td>Allows user to view all the available zwave devices, delete and edit them. /create may be used to create new zwave devices to andon-raspberry.</td>
<td></td>
</tr>
</tbody>
</table>

Usage of the API urls (/api/syncaalerts and /api/alertsopen) do not require password. Syncaalerts may be used to trigger syncing operation sync.js manually. Functionality of sync.js is explained in next chapter. Alertsopen may be used to retrieve the data of all open alerts in andon-raspberry database. The alertsopen returns the alert data as JSON string. /alerts/open webpage may be accessed to get more visual presentation of the open alerts. As /alerts/open does not require password to access, no changes can be made to alerts directly from that page. API functionalities were created in case other application was developed to work with andon-raspberry.

### 5.7 Utilities

Andon-raspberry has three major utility functionalities: logger.js, sync.js and token.js.

**logger.js**

Logger.js handles all the different levels of logging. Winston is used as logging library. Winston has 6 different levels for logging: error, warn, info, verbose, debug and silly.
These are levels are ranked by most important to least important. When least important level is used it will also log the higher importance messages. [36] logger.js uses three of these levels as described in the Table 5-3 below:

<table>
<thead>
<tr>
<th>Logging level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>Only for error logging</td>
</tr>
<tr>
<td>info</td>
<td>Logging high level operation of components, such as initializations and websocket messages for configured alerts.</td>
</tr>
<tr>
<td>debug</td>
<td>Logs websocket messages from Application 1 on more accurate level than info. These can be used for debugging purposes if more information on the system is needed.</td>
</tr>
</tbody>
</table>

All the log levels will write their own .log files to \logs folder. Log files are limited to 5 megabytes per file and five files per logging level. The info level log messages are also shown as console log messages for quick debugging.

**sync.js**

sync.js is used to retrieve alert information from Application 1. The alert database on andon-raspberry won’t be properly updated if messages are dropped during websocket communication. sync.js is used to correct these desyncing issues. Sync.js uses POST requests to eAndon API to generate list of desynced alerts. After the list is generated the andon-raspberry alert database is updated and the light colors get updated after that. The syncing can be triggered by using andon-raspberry API and from the website. It is also executed on main application on cron format configurable interval. Figure 5-6 displays the syncing process in block diagram.

**Figure 5-6 Block diagram of alert syncing**

**token.js**

Figure 5-7 has a block diagram of functionality of token.js. token.js keeps the authorization tokens required by Application 1 up to date. When the token.js is called it check if the token has expired and if it has retrieves new one. The tokens are retrieved from token server with secret authorization token. The tokens are valid for 7200 seconds. The token.js saves the valid token as database entries with the authorization token and timestamp to /data folder.
5.8 Combined functionality

When andon-raspberry and Application 1 are communicating the alert description field is used to update the alerts in andon-raspberry’s side. This was the most convenient way to make the alert updating simple as possible as this allows one alert to control multiple lights or one light to be controlled by multiple alerts. The configuration that needs to be done in Application 1 is to use underscore (‘_’) in alert description to split between lights. andon-raspberry parses the alert description field for ‘_’ and uses the divided parts to control lights if needed. This is the only specific configuration that should be noted between the communication of these two applications.

This chapter explains the functionality of andon-raspberry as a complete application by three UML interaction diagrams:

1. Starting the application
2. Button pressing to generate alert
3. Alert update to update in light color

The Figure 5-8 presents the start of the application. When the app is started first all the required libraries are initialized. After the initialization the andon-raspberry attempts to open websocket listener connection to Application 1. Next the Z-way server websocket listener connection is opened. HTTP server will be started after the websocket connections have been established. And finally, the alerts are synced between Application 1 and andon-raspberry and scheduled event to run syncing is started. After all these steps the application is ready receive updates and manage the Z-wave devices on the production floor.
Button press to new alert in Application 1 has been explained as UML interaction diagram in Figure 5-9. When operator presses the button to generate alert, Z-way will send websocket message to andon-raspberry. First andon-raspberry will check button database for timestamp when the button was last time pressed and compare it to current timestamp. If the delay is larger than the configured delay (default: five seconds), the timestamp will be updated to new value, alert definition ID and comment will be retrieved based on the node ID of the pressed button.

After the data has been retrieved, the token.js is used to check for valid access_token, if token is not valid new token is retrieved, else POST request is send to Application 1 with the alert definition ID, comment and with access_token to initiate an alert. If the alert generation is successfully the Application 1 will respond with success message with alert information. This response message is not used to update andon-raspberry database as all the alert updates are done based on the websocket messages from Application 1.
Figure 5-10 presents the light bulb color change process when alert has been updated in Application 1; this also includes the generation of new alerts. When the alert status changes in Application 1 it will send a websocket message to websocket listeners that have subscribed to listen for that location. First the andon-raspberry checks if the alert has been configured for that specific andon-raspberry. If the alert has not been configured it will be ignored. In case of configured alert, the andon-raspberry will next check which lights are handling the configured alert and get node ID for the lights. Next the andon-raspberry will update the alert database. After the alert database has been updated, andon-raspberry will make comparison between all alerts for light (node ID) and check the highest priority alert that is currently open (from highest to lowest: initiated, acknowledged and resolved). After this information has been retrieved the andon-raspberry will form the URL to change the color and send GET request to Z-way server. This is done even in the cases when no light color change is needed, as the andon-raspberry is not able to get real-time data from the light colors. The color knowledge is based on the statuses of the alerts from alerts database.
Figure 5-10 UML of alert update to light change
6 Use case

As of 12th of September 2018 there are three Raspberry Pis with Z-wave support in use. 26 Z-wave lights are configured to function as Andon systems and three as Kanban systems. [37] These two applications of andon-raspberry are explained later in this chapter. Almost 6500 different kinds of alerts have been created and resolved between January 2017 and October 2018. [38]

Multiple RPis are used due to Z-wave communication limitations caused by obstacles. There are three main obstacles in the production floor: distance between two nodes, elevator shaft and stairway metallic structures. Using multiple RPis adds an extra layer of configuration and handling but it has been done to ensure reliable Z-wave communication between the nodes.

In the first explained application the system is used as Lean Andon system. This means that the lights operate on three different colors depending on the situation on the production line: Green – no issues, Yellow – issue that has been acknowledged and Red – issue initialized on the production line. The Figure 6-1 depicts the flow of configuration of andon-raspberry. This is the most common application of andon-raspberry system.

![Diagram](Image)

**Figure 6-1 Light bulb configured to function as Lean Andon system**
The Alert Description in Application 1 has been configured as PRODLINE. The same configuration has been given in andon-raspberry to the light in specific location. When the operator on PRODLINE location creates an alert the Andon light will turn red and supervisor will get notification from Application 1. After the supervisor notices the issue and acknowledges in Application 1 the light will turn yellow. And finally, when the problem has been completely solved the supervisor will resolve the alert in Application 1 and the production line’s light color returns to green to indicate that there are no open alerts for PRODLINE.

In the second application the initialized state of alert is configured to function as blue light. This color works as information color to inform operators that new alert has been initialized. This system is used in storage to allow faster delivery of new parts and in repair department to improve response time to fix issues. No issues and acknowledged states have been configured to display no color. Flow for this kind of system has been presented in Figure 6-2.

In this example the alert description has been configured as SURPLUS. When issue is initialized on the production line the light in storage will turn blue. This tells operators to go and check Application 1 for open alerts. After the operator acknowledges the alert will turn off.

![Figure 6-2 Light bulb configured to function as Kanban system](image)

These two applications may be combined. If using the situations in Figure 6-1 and Figure 6-2 the alert description would be configured as PRODLINE_SURPLUS. Figure 6-3 has this combination situation presented. In this example there are two Z-wave controlled light bulbs installed: one configured as Andon system and one as Kanban system. Andon system light has been configured to respond to alert descriptions that include PRODLINE and Kanban light respond to SURPLUS alert descriptions.

When there is upcoming shortage on the production line, operator will create new alert. This alert will turn the light on production line red and on storage to blue. The storage personnel will notice that there is new alert in Application 1 that
requires attention. After checking the alert, the storage personnel will acknowledge the alert and the light will turn yellow on production line and turn off in storage room. From the change of color to yellow in production line the operator knows that the parts are being collected.

After the required parts have been collected the storage personnel will resolve the alert and start delivering the parts. This turns the light on production line back to green to inform operator that the part is being delivered soon.

Figure 6-3 Combination of andon-raspberry configurations
7 Conclusions

Before this thesis production stations which had Andon were using light bulbs controlled by using remote control. These light bulbs didn’t have any additional communication and their only function was to generate visualization on the production. Most of the production lines didn’t have any visualization available. The company wanted to improve these and a new more useful Andon system was developed. This thesis was to develop part of the Andon system which handled the control of lights and buttons and communication with Application 1.

In this thesis Lean manufacturing concept was first introduced, also Andon was explained in more detail. Next the Z-wave communication was explained and the Z-way server and RaZberry and RPi were presented.

Next requirements for the application were discussed. The requirements did outline the technologies and some hardware to be used. System was to use Z-wave technology and be ran on Raspberry Pi, lights were to have at least three colors (red, yellow and green) and both lights and buttons were to be wireless. Also, the requirements did introduce how the communication was done between the Application 1 and andon-raspberry developed in this thesis.

After requirements, the hardware and software were introduced in more detail. The hardware of the system consists of Raspberry Pi with RaZberry daughter card to manage Z-wave communication, Z-wave multi-color LED light bulbs and buttons with Z-wave support. In the software chapter the important libraries and database schematics were introduced, communication of andon-raspberry with Application 1 and Z-way server were explained. Also, the combined functionality of the complete system was explained.

Finally, use case was presented on how the andon-raspberry is being currently used in three different ways. The setups are:

- Andon system with red, yellow and green colored lights
- Kanban system, which informs operators on new alerts by showing blue light when alert is generated
- And combination of both. For example, used during part surplus alerts.

All the remote controlled light bulbs have been replaced with andon-raspberry; there are currently 30 Z-wave light bulbs in the production use, controlled by three different RPis due to range limitations. Approximately 6500 alarms have been generated in Application 1 and andon-raspberry has provided visualization for them in the production floor. The buttons provided by andon-raspberry have been currently replaced by barcode reader which allows generating more complex alerts directly from Application 1.

As a conclusion, Andon system has been developed and has been taken into use in the manufacturing site. More light bulbs will be most likely added to cover all the production stations.

Future work
The application developed in this thesis can still be further developed. Some ideas to extend the application are presented below:

- Adding more Z-wave supported devices to andon-raspberry. For example, by using Z-wave controlled sensors, alert generation could be automated. Also, in addition to lights, sounds could be added to add extra layer of notifications for critical alerts.
- The debugging of issues within the Application 1 communication or Z-wave communication could be improved upon. Currently the andon-raspberry only saves log files and displays some information as console print.
- Usage of the andon-raspberry could be made more robust if wanted. Currently there is a small learning curve before using the application.
References


