FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Entwicklung eines Prozessmanagementsystems auf Raspberry PI 3 mit Activiti für Smart Home

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Software and Information Engineering

eingereicht von

Anja Klipic
Matrikelnummer 01448440

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Ao.Univ.Prof. Dipl.-Inf. Dr.-Ing. Jürgen Dorn

Wien, 1. Oktober 2019

Anja Klipic

Jürgen Dorn

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Development of a Process Management System for a Raspberry PI 3 with Activiti for Smart Home

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Software and Information Engineering

by

Anja Klipic
Registration Number 01448440

to the Faculty of Informatics
at the TU Wien
Advisor: Ao.Univ.Prof. Dipl.-Inf. Dr.-Ing. Jürgen Dorn

Vienna, 1st October, 2019

Anja Klipic	Jürgen Dorn

Erklärung zur Verfassung der Arbeit

Anja Klipic

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Oktober 2019

Anja Klipic

Kurzfassung

Smart Home ist ein Begriff, der eingeführt wurde um moderne Haushalte zu beschreiben, die mittels Installation von *intelligenten Geräten*, wie zum Beispiel intelligente Steckdosen oder Sensoren, zu energieeffizienten Eigenheimen ausgebaut werden, die man unter Verwendung von Endgeräten wie Smartphones, Tablets oder Computer steuern kann. Dies eröffnet neue Möglichkeiten, Teile des alltäglichen Lebens einer Privatperson zu automatisieren. Damit hätte man potentiell eine bedeutende Lösung, um kranke oder ältere Menschen in ihrem Alltag zu unterstützen, ohne sie ihrer vertrauten Umgebung zuhause entreißen zu müssen.
Automatisierung ist seit jeher ein Teil der Art und Weise, wie große Unternehmen ihren täglichen Geschäftsablauf steuern, Aufgaben vereinfachen und organisatorischen Mehraufwand minimieren. Geschäftsprozessmanagement hat sich als passende organisatorische Kategorie etabliert, und BPMN (Business Process Model and Notation) dient als Standard für die Modellierung der betroffenen Geschäftsprozesse.

Um potentiell von einem bereits gängigen und bewährten System wie dem Geschäftsprozessmanagement zu profitieren, beschäftigt sich diese Thesis mit der Frage ob Smart Home Automatisierung mit den gleichen Techniken und Technologien modelliert und implementiert werden kann wie Geschäftsprozesse. Dies geschieht durch die Erschaffung einer Smart Home Prozessmanagementsystem-Prototypenanwendung unter Verwendung des BPMN Frameworks Activiti und dessen API in einer Spring Anwendung geschrieben in Java, mit Smart Home Prozessen modelliert im BPMN2.0 Standard. Die Prozesse werden in der Activiti Anwendung ausgeführt, auf einem Raspberry Pi 3 Mikrocontroller, der als Steuereinheit für das SHPMS dient.

Die erfolgreiche Ausführung aller Schritte, die in der Prozessdefinition der Smart Home Prozesse definiert wurden beweist, dass ein Raspberry Pi die Activiti Process Engine tragen kann und somit Potential besitzt, um in einem ausgefeilteren Smart Home Prozessmanagementsystem als Steuereinheit verwendet zu werden.

Abstract

Smart Home is a term introduced to describe modern homes which, through the installment of smart devices such as smart power outlets or sensors, are being made into energy-aware homes that can be controlled using end devices like smartphones, tablets or computers. It opens up new possibilities for automating parts of the everyday lives of private individuals, which could be a great solution for supporting the sick or elderly in living their lives in the comfort of their own home.
Automation has already been a part of the way big corporations and businesses handle their daily business processes, simplifying tasks and minimizing organisational overhead. Business Process Management has evolved as the corresponding organisational strategy, and BPMN (Business Process Model and Notation) serves as the standard for modeling the business processes in question.

In order to potentially benefit from an already established, proved system such as BPM, this thesis aims to investigate whether Smart Home Automation could be modelled and implemented with the same techniques and technology as Business Processes. It does this by creating a Smart Home Process Management System prototype application using the BPMN framework Activiti and its API in a Spring Application written in Java, with Smart Home Processes modelled in the BPMN2.0 standard. The processes are executed using the Activiti application, on a Raspberry Pi 3 serving as the Control Unit for the SHPMS.

The successful execution of all steps defined in the Process Definitions of the Smart Home Processes proves that a Raspberry Pi can support an Activiti Process Engine and therefore has potential to be used in a more sophisticated Smart Home Process Management System.

Contents

Kurzfassung **vii**

Abstract **ix**

Contents **xi**

1 Introduction **1**
1.1 Overview 1
1.2 Scientific Issue 1
1.3 Method 2
1.4 Outline 2

2 Process Management **3**
2.1 Overview 3
2.2 BPMN 3
2.3 Tools and Frameworks 4
2.4 BPMN 2.0 6

3 Smart Home Automation **11**
3.1 Internet of Things 11
3.2 Common Home Automation Use Cases 12
3.3 Fixed vs. Flexible Setup 16
3.4 Benefits beyond Comfort 17

4 Chosen Automation Processes **19**
4.1 Weekday Morning Routine 19
4.2 Weekday Night Time Routine 21

5 Hardware Setup **23**
5.1 Raspberry Pi 3 Model B+ 23
5.2 LEDs 24
5.3 BME680 24
5.4 HC-SR501 24

6 Smart Home Process Management System **25**
6.1 Problem Statement 25
6.2 Implementation 26

7 Evaluation **35**

8 Summary and Conclusion **37**

List of Figures **39**

List of Tables **41**

Bibliography **43**

A Instructions for Installing SHPMS Prototype on Raspberry Pi 3 **45**
A.1 Preparation 45
A.2 Steps 46

B Instructions for Creating a new Process in SHPMS Prototype **47**
B.1 Process Model 47
B.2 Implementation 48

CHAPTER 1

Introduction

1.1 Overview

Everyday things are made to be *smart* by the minute. Companies all over the world have realised the potentials of the so called Internet of Things and are attempting to keep up with the growing number of connected services intended to simplify our lives.

The possibilities seem endless but a few big players stand out: *Industry 4.0, Smart City, and Smart Home*. Large scale investments are being made in these fields, due to their observed and predicted short and long-term benefits.

However, the fact that a lot of parties want their share of the cake in this new, evolving and promising environment has the effect that many individual solutions are being developed across the market. This means that making the home smart can potentially result in a lot of different setup processes and similarly many systems to manually maintain throughout their life cycle.

1.2 Scientific Issue

This thesis looks into the possibilities of taking the topic of Smart Home and combining it with so-called Business Processes, to find answers to the following two questions:

1. Can Smart Home Automation be viewed as a process in the context of Business Processes or rather, which processes can be identified in the Smart Home?
2. How can such processes be developed flexibly and efficiently?

These questions were attempted to be answered by creating a lightweight system that allows amateur programmers (in this case, students of the *Technical University of Vienna*)

to test how various Use Cases in the Smart Home field could be implemented, automated and configured.

1.3 Method

Using a *Raspberry Pi 3 Model B+* as the *Smart Home Control Unit*, an attempt was made to implement or simulate two *Processes* that should be automated and managed using the *Activiti BPMN Engine* (2.3.1) running on the Control Unit (CU). In order to get an authentic *Smart Home Automation Setup*, a number of sensors were included, communicating with the CU for relevant data collection purposes.

The focus of the project was not to create a fully-fledged-out *Smart Home Application*, but rather to test the compatibility of the *Activiti BPMN Engine* on a small amateur Raspberry Pi Control Unit in a *prototyping* manner. In order to keep it lightweight and low-budget, a simulation of some parts of a given process was chosen over the actual implementation, to save hardware costs and keep the scope of the work within its boundaries.

1.4 Outline

In the following pages, a brief introduction to the topics that make up the purpose of the project is provided.

Chapter 2 gives an overview of the term Process Management and BPMN, touches upon some Tools and Frameworks supporting the development of Process Management systems and introduces the BPMN Engine used in the implementation of the prototype for a Process Management System done in the scope of this project.

In Chapter 3, Smart Home Automation and the Internet of Things are explained briefly, including some practical examples of their use and some points of motivation behind this particular piece of work.

Further, Chapter 4 describes the processes which were chosen to be worked with, while Chapter 5 provides a description of the hardware needed to realise and implement the automation of said processes.

Finally, Chapter 6 covers the project itself; a lightweight amateur Smart Home Process Management System.

The findings of the implementation process - including a discussion about what did and did not work as intended or expected - are covered in Chapter 7, followed by the summary and conclusion in Chapter 8.

CHAPTER 2

Process Management

2.1 Overview

The term *Process Management* is mainly known within the Business and Economics sector (more specifically as *Business Process Management*), used as the standard for automating and managing the various business processes of an organisation or company [19].

Business Process Management encapsulates end-to-end activities needed to complete a transaction, as formulated by Salibindla [19]. The processes contain regularly performed actions or transactions to reach certain goals or goal states. Using BPM, it becomes possible to not only define, but also continuously improve the processes within a business that are valuable to both customers and the business itself, as well as other stakeholders concerned [19].

Due to the fact that organisations, especially big ones, largely depend on well-functioning and efficient processes when running their business, *(Business) Process Management* carries an important weight. However, the more time has to be spent on configuring and maintaining it, the less time is available for the actual business processes themselves.

Therefore, efforts have been made to create ways to automate *Business Processes*. Not only does this noticeably facilitate the management task(s), it also saves valuable time and allows for greater scalability and flexibility, all of which are necessary features for a successful, adaptable business.

2.2 BPMN

An important step towards enabling the automation of *Business Processes* is the introduction of a standardized way of modeling such processes. Standardized models allow for

quick, widely understandable descriptions of very specific processes, and with the right tools they can be created quickly and efficiently.

One standard, created for exactly those purposes described above, is the *Business Process Model and Notation* standard, or short *BPMN*. It is a graphical notation showing the steps of a business process in the sequence in which they are generally executed. It further illustrates the participants involved in the process and how they communicate with each other (e.g. the message flow). The graphical model can be created using a respective framework or simply by defining it in XML. Further, not only does the BPMN describe a process in an easy-to-understand, visual way, it is also technically executable [6].

Using the BPMN standard in the business world can be loosely compared to the usage of the UML notation in Software Engineering. While UML (or *Unified Modeling Language*) is a standard used to model the detailed views of a Software in preparation for or during its implementation, BPMN provides a free standardized way to model complex business processes both at a higher and lower level, depending on the user [17].

Within the scope of this project, the low-level view is going to be the focus when talking about and using *BPMN*, since the idea is to use it to implement the automation of two chosen *Smart Home* processes.

Smart Home processes do not directly fit into the category of what is commonly referred to as a "Business Process", however, the structures resemble one another. A smart home process can be defined by an objective and the number of activities that need to be performed in order to reach said objective. This matches the decription for Business Processes in Section 2. Since the process is meant to be automated, it was decided that using BPMN to model and eventually execute it was a good choice for this research project.

2.3 Tools and Frameworks

In order to support and simplify the usage of *BPMN*, several tools and frameworks have been developed. The choice of the right tool depends on what specifications it offers and what specifications are required. Additionally, a good tool can be a matter of personal opinion.

Hesse [15] collected BPMN tools available and performed a comparison over their core facts and features. The collection lists the tools with details about their Platform/OS-compatibility, supported BPMN Standard, first and latest release as well as the Software License under which they are distributed. As an overview, it serves its purpose, unfortunately it cannot be reliably said whether the page is still actively maintained, due to how some dates in the "Latest Release" column seem older than the release tags of the respective Github repositories show.

Therefore, the focus on the comparison was kept on the static facts that have not changed over time.

Name	**Creator**	**Platform/OS**	**BPMN Version**
Activiti Modeler	Alfresco / Activiti community	Cross-Platform	BPMN 2.0
Bonita BPM	Bonitasoft	Windows, Linux, Mac	BPMN 2.0
bpmn.io	Camunda Services GmbH	Cloud	BPMN 2.0
Camunda Modeler	Camunda	Cloud	BPMN 2.0
Eclipse BPMN2 Modeler	Eclipse.org / Eclipse SOA project	Cross-Platform	BPMN 2.0
Imixs Workflow	Imixs	Cross-Platform	BPMN 2.0
RunaWFE	Runa Consulting Group	Cross-Platform	BPMN 2.0
simpl4	transparent solutions GmbH	Java, Cloud-Platform PaaS/SaaS	BPMN 2.0

Table 21: Free and Open Source BPMN Tools as per [15]

Of the 67 BPMN tools listed, 10 were identified as "Free and Open Source", another 9 offering both a "Free and Proprietary" version, the remaining 48 tools use proprietary Software Licenses.

Open source software generally offers a more flexible approach to its usage. It not only offers transparency of the software's implementation, but also allows a potential extension of its existing features by the user itself, according to his or her needs. This can happen by building upon an already existing system instead of needing to reinvent the wheel.

The choice of a suitable BPMN framework for this project was therefore performed from within the pool of the open source frameworks in Table 21.

2.3.1 Activiti

As the similarity between those 10 frameworks does not single out the one best candidate for the job, Activiti was the experimental choice for this project and related potential extended research. The code base can be found on Github [1].

Activiti consists of the *Activiti Core* and the *Activiti Cloud*, two deployment options that can be chosen from depending on the developer's needs and requirements.

Activiti Cloud represents the framework in its Cloud Native whole, posing as a way to create, manage and run BPM implementations in the cloud. It was made with the goal of keeping it lightweight and single-focused, but with a set of services that will allow most BPM implementations to be performed through the use of the framework. These services are kept independent from one another, so that the developer can choose precisely which ones are needed and which ones are not. They can even be replaced by custom implementations, thereby increasing the flexibility of use [5].

[1] `https://github.com/Activiti/Activiti`, accessed on 07.06.2019

Activiti Core contains two Java Runtime APIs (*TaskRuntime API* and *ProcessRuntime API*) to be used inside a Spring Boot application; in other words, it acts as a library in a Java application [4].

The purpose of the Process Runtime is to parse BPMN 2.x business Process Definitions in order to automate their executions [5].

Lastly, the Activiti Cloud can be split into three main components:

- Activiti Cloud Infrastructure
- Activiti Cloud Applications
- Activiti Cloud Modeler

For this project, the latter two are particularly relevant as they offer the BPMN 2.x Modelling functionality for the processes described in Chapter 4 as well as the actual automation using the ProcessRuntime in a Java Spring Boot Application [2].

2.4 BPMN 2.0

For the sake of completeness and a better understanding of the elements used in Figures 41 and 42 - the models visualizing the processes described in Chapter 4 - this section is aimed at introducing those elements and their role in the BPMN 2.x Standard. The following descriptions were collected and adapted from the **BPMN Guide** written by Gagné and Ringuette [12].

2.4.1 Events

There are three main event types in BPMN 2.0; Start event, Intermediate event and End event. Start- and End event are crucial to modelling a process, because the way a process starts and the way it ends need to be well-defined.

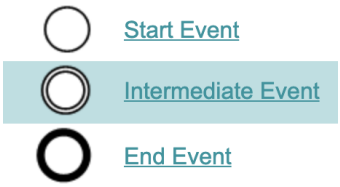


Figure 21: BPMN 2.0 Events

An event itself can be of a specific type. Start events can be *Signal*-, *Message*-, *Timer*- or *Conditional* Start events, End events can be of type *Signal*, *Message*, *Escalation*, *Error*, *Compensation* or *Terminate*. If a type is not explicitly set, Start- and End events can

also just be plain. The process models of this project only use Signal Start Events and plain End events, so the other types will not be further explained.

The Signal Start Event signifies that the modelled process starts upon the receipt of a certain signal; this could be a signal coming from a type of sensor, for example. All events can be labeled to describe their role in the process more clearly.

Figure 22: Wakeup Alarm Start Event

Figure 22 shows the Signal Start Event used in the model of the Weekday Morning Process found in Section 4.1 of Chapter 4.

2.4.2 Gateways

The two Gateways relevant for this project, as they are the ones used in the process models, are the *Parallel Gateway* and the *Exclusive Gateway*.

Parallel Gateways allow for the sequence flow to be split up into two or more parallel paths. The tasks in those paths are executed in parallel, until a second *Parallel Gateway* joins them back together into one. The joining parallel gateway will only fire the continuation of the process flow once all parallel paths have reached it. Figure 23 shows how the *Parallel Gateway* was used in the Weekday Morning Routine Process Model (see Fig. 41).
It is a simple example showing how the sequence flow gets split into two parallel paths each containing one task to be executed before the joining gateway brings the sequence flow back into one path. The joining gateway will wait for both the `CreateBasicClothingSuggestion` and `LightActivationBySunStatus` tasks to finish before it continues in the process flow.

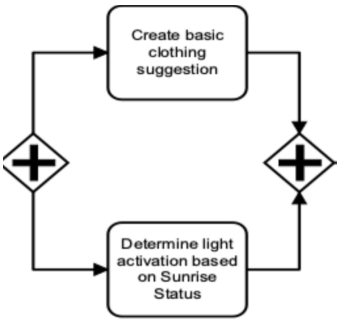
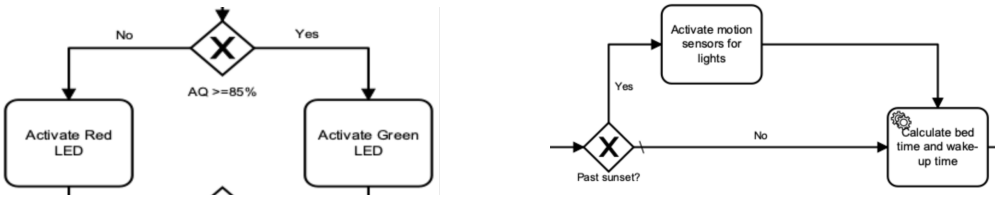


Figure 23: Parallel Gateway Morning Process



(a) Weekday Morning Routine

(b) Weekday Nighttime Routine

Figure 24: Exclusive Gateways from two Process Models

An *Exclusive Gateway* also splits a single incoming sequence flow into two or more paths, however, instead of a parallel run through these paths, each is coupled with a conditional expression that determines which of the paths should be chosen. Normally, the *Exclusive Gateway* is labeled with the condition that is to be checked, and the forking paths are labeled with the results they represent.

It is possible to define one of the paths as the *default* path to take, for example if there are two paths and only one condition leading to a certain path is specified, the default path will be taken in all other cases.

2.4.3 Tasks

Tasks are a subgroup of BPMN Activities. Activities are connected through a flow, and can be *Tasks*, *Transactions*, *Sub-Processes* or *Call Activities*.
There are various kinds of tasks that can be modelled in a BPMN Process. Each task has a clear objective and should be modelled to closely represent the nature of the action it is required to perform. BPMN2.0 has introduced many task types, including:

- Service Tasks
- User Tasks
- Send and Receive Tasks
- Manual Tasks
- Script Tasks
- Business Rule Tasks

The implementation of the SHPMS prototype was done using **Activiti version 7**, as it was the latest version at the time of writing this paper. In that version, not all task types were supported by the API. That is why in the Process Models of this project, only *Service and User Tasks* were used.

According to the Camunda BPMN Reference ([7]), Service tasks are normally done by Software. Usually, BPMN assumes that they are web services, but it could be any

implementation. With regards to the Activiti-Smart-Home project, any tasks that either use an external web service (like the Weather or Google Calendar Events polling) or execute external scripts (for example to control sensors and actuators of the Smart Home System) would rightfully be modelled by a *Service Task*.

(a) Weekday Morning Routine

(b) Weekday Nighttime Routine

Figure 25: Service Tasks from two Process Models

Figure 25 shows two of the Service Tasks used in the Protoype Process Models.

Lastly, the User Task is self-explanatory in that it is a task that is to be done by a User. The difference to a *Manual Task* is that with a User Task the Process Engine requires feedback on the completion of the task, otherwise it won't continue with the process execution. A Manual Task has no influence on the process flow.

Figure 26: User Task Nighttime Process

Figure 26 shows the User Task used in the Weekday Nighttime Process Model; the Process Engine receives the completion update by means of a REST request to the route `/readconfirmation`.

CHAPTER 3

Smart Home Automation

3.1 Internet of Things

In order to fully grasp the concept of the *Smart Home* or *Home Automation*, it makes sense to first look into the definition of the **Internet of Things**.

According to Gubbi et al. [14], the Internet of Things is created through a network of sensors and actuators that are well integrated into our surrounding environment in order to collect information, share it and use it to improve actions and processes that are already happening in the physical world. The devices that are part of the network are uniquely addressable and able to communicate in a uni-, bi- or multidirectional manner with either a form of parent *hub*, other sibling devices in the network, or bigger network nodes like cloud or server nodes.

The fundamentals of IoT research have their origin in so-called *Wireless Sensor Networks* (WSN) and *Radio-frequency identification* (RFID) technology, which have been established before Internet of Things started being used as a term. Through the advancements of technologies such as micro-electro-mechanical systems (MEMS) or wireless communication, it was possible to create devices of small size, like sensors, which would sense their environment, perform minor computational tasks and were able to communicate wirelessly in a network-like structure, posing as the nodes of a WSN [20]. Kevin Ashton, a British technology pioneer, coined the term *Internet of Things* in 1999 in a production context, but over the years it spread into various other areas including healthcare, transport, building technology etc [14].

Naturally, in order to differ from a simple WSN, the IoT has to be more than just a collection of sensors communicating with each other. As cited in Stojkoska and Trivodaliev [20], adding a networking, service and interface layer to the one made up of the sensors gives the IoT its larger-scale reach and raises it from a WSN to a successful platform for a smart world.

Integrating ICT (*Information and Communication Technology*) into pyhsical objects that used to exist completely independently of computer-technological devices is one of the main concepts of IoT, and it brought with it a major research interest into application areas like the *Smart Home*. Stojkoska and Trivodaliev [20] state that there is an incentive for modern homes to be made into energy-aware smart homes through the usage of smart devices such as smart power outlets, sensors, smart appliances and so forth. However, implementations of smart homes are still rarer than those in the industrial or commercial sector.

Nevertheless, as interest is growing and research is continuously improving technology, the number of smart objects integrated into the home in the next few years is likely to be higher than ever before. In 2011, more interconnected devices than people existed in the world, by 2020 the number will rise to approximately three times of today's world population (at the time of writing, [18] counts 7.71 Billion), making it 24 billion smart devices on this earth.

According to Wilson et al. [21], the global market for smart appliances is expected to be at about $26bn as of this year in 2019, with potentially over half a million households in Germany owning a form of smart appliance or device.

3.2 Common Home Automation Use Cases

In the *Smart Home*, many smaller or larger domestic scenarios can be identified for potential automated support or even full automation. Those scenarios follow the general idea of the IoT advancing a more efficient and safer society by means of making 'everything' a service [20].

In order to visualize the growing interest into the topic, three potential use cases were chosen and their increase in popularity analysed by Google Search Trends [1] results of search words fitting into each use case. Two different search phrases per use case were formulated to be analysed and visualised in a graph.

The analysis was made using a set time frame of the year 2004 until today (at the time of writing, "today" marks April 2019), listing measures for each month in the specified time span. As for the location, search results from worldwide were included.

The values returned by the analysis express the search interest relative to the highest point in the diagram for the chosen region in the set time frame. Hereby, the number 100 signifies the highest level of search interest for the given phrase, and a value of 50 in the same diagram would mean the search interest was only half as high as the highest level measured. A result of 0 indicates that there was not enough data available for the given search phrase. [2]

[1] `https://trends.google.com/trends/?geo=AT`, accessed on 23.05.2019
[2] See footnote 1

3.2.1 Smart Energy Management

The first Use Case is Smart Energy Management. In this category, the two chosen phrases to have analysed by Google Search Trends are *Smart Lighting* and *Smart Thermostat*. Their increase in search interest is similar, the latter can be observed to have an earlier rising start and more spikes, however both sit at just under 50 at the current time.

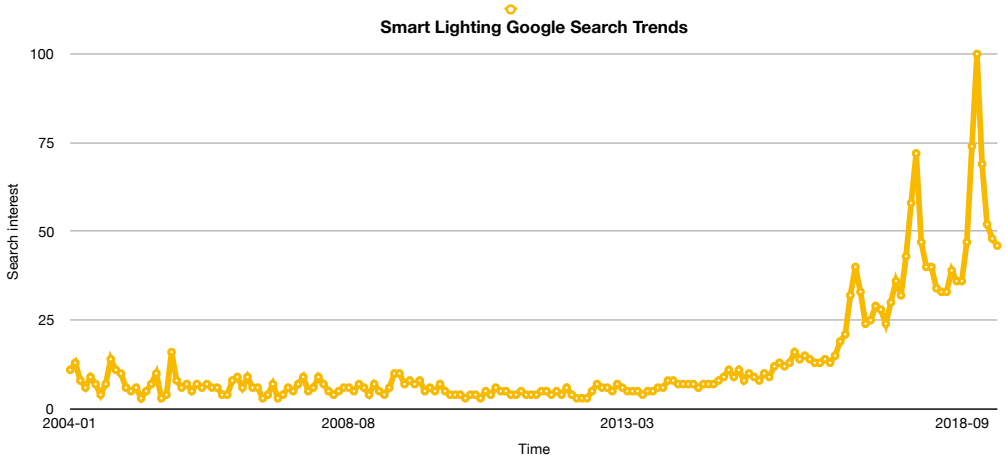


Figure 31: Smart Lighting Google Search Trends

Google Search Trends does not disclose the exact searches that create the search interest value, therefore it cannot be said why the spikes exist the way they do. However, the tables that these charts were created upon list the spike values in the later months of a year, specifically November and December. This could suggest that searches go up right before Christmas when people tend to purchase these kinds of technological gadgets as christmas gifts.

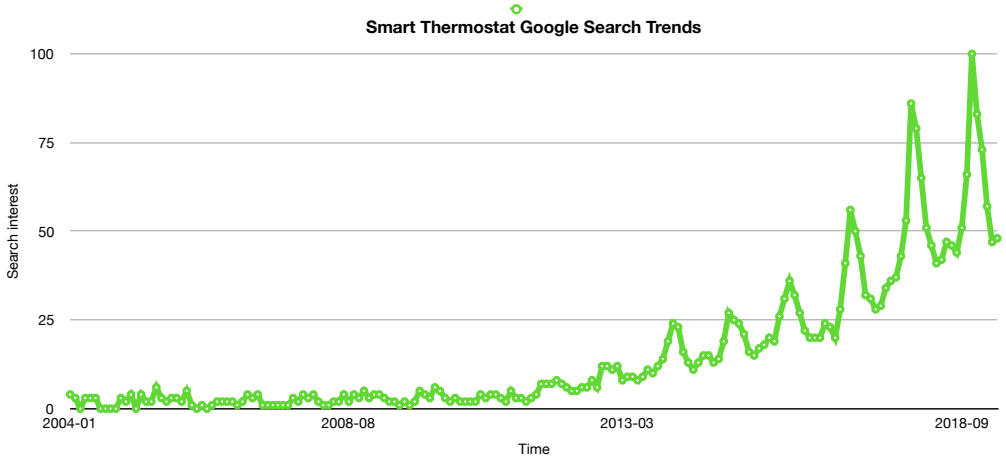


Figure 32: Smart Thermostat Google Search Trends

3.2.2 Smart Appliances

The second Use Case is Smart Appliances. For this category, two common home appliances (in their *smart* version) were chosen as a search phrase, namely *Vacuum Robot* and *Smart Fridge*.

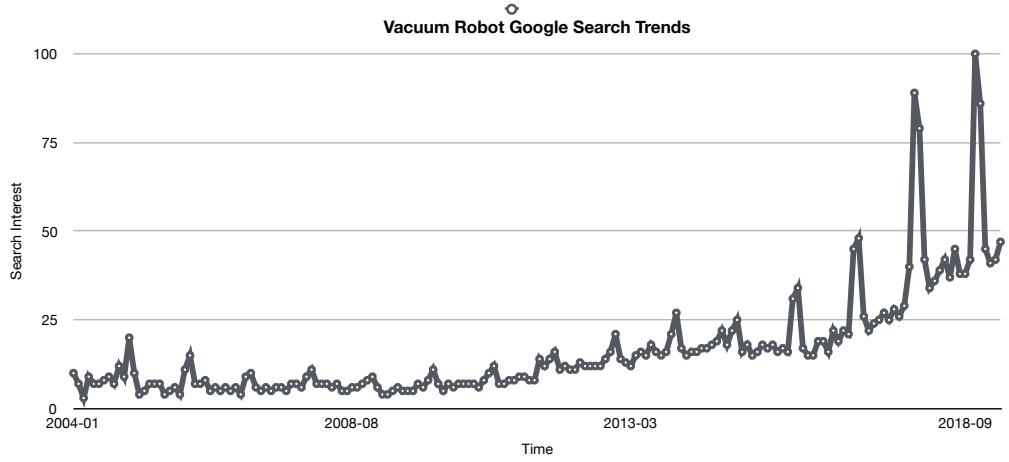


Figure 33: Vacuum Robot Google Search Trends

Their trendline is very similar, although the Smart Fridge search interest (see Fig. 34) can be observed to rise quicker. There are also fewer extreme spikes in 34 than in Figure 33, and currently the Smart Fridge search interest sits at just under 60, while the most recent Vacuum Robot search interest measurement is slightly below 50.

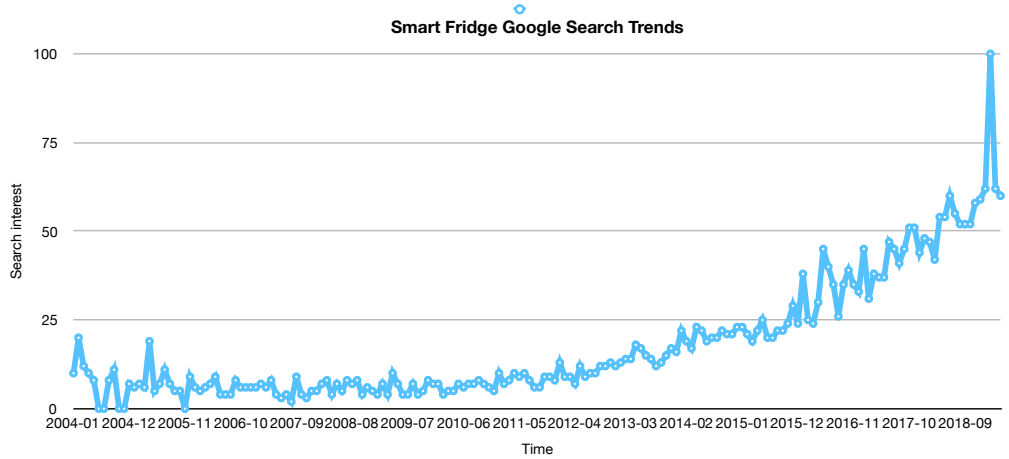


Figure 34: Smart Fridge Google Search Trends

3.2.3 Smart Home Security

The third and last Use Case is Smart Home Security. Here, the first search phrase used is a specific product line belonging to the Smart Home Security Systems market, called *Nest Security System* [3] and the second phrase is the broad topic of *Smart Home Security* itself.

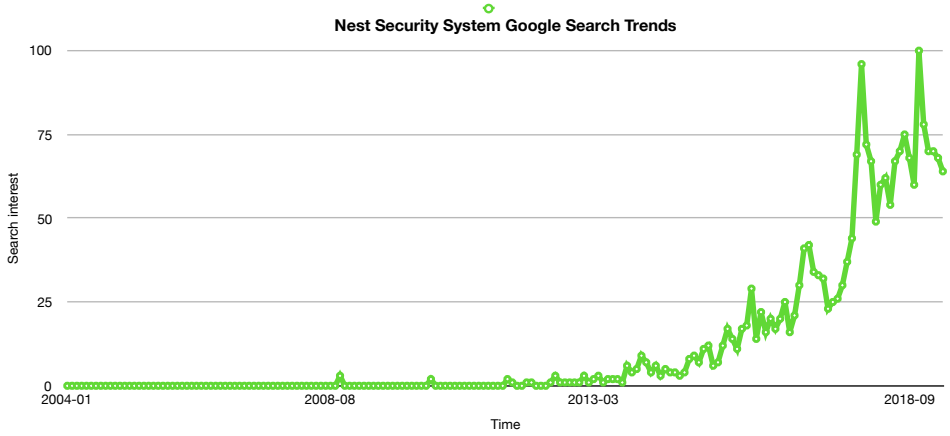


Figure 35: Nest Security System Google Search Trends

In Figure 35, it can be seen that until around 2013, there were no records found for the search interest on the phrase *Nest Security Systems* (disregarding the small spikes in mid-2008 and around 2010, those could be either mismatches of the search phrase, related to announcements for the technology, or a different cause entirely). However, once values above 0 start appearing, the trendline can be drawn steeply up, with major spikes in the past 2 years.

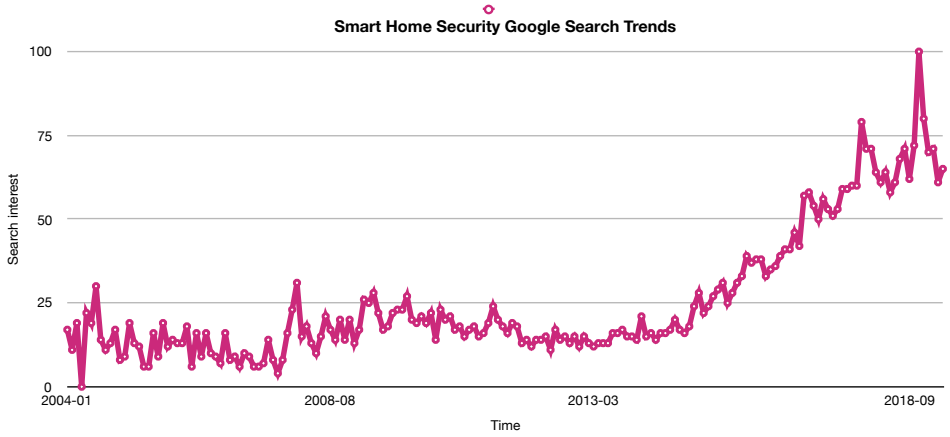


Figure 36: Smart Home Security Google Search Trends

[3] `https://nest.com/alarm-system/overview/`, accessed on 23.05.2019

Figure 36 shows the highest values for the period betweeen 2004 and 2008 of all other Figures 31 - 35, but the overall trendline starting at around 2013 is again very similar to the other search interest results, with the current value sitting at just over 60.

3.3 Fixed vs. Flexible Setup

Sections 3.1 and 3.2 illustrate the increasing interest for Smart Home. Upon its growing demand, companies started to develop products and solutions to compete in the global market. What started out as standalone "smart" products has been taken to wider dimensions by providing whole systems that are integrated well with the existing home and other services.

Customers can choose between various providers of systems such as (*Smart*) Lighting, Assistant speakers, Appliances etc. The rapid evolution of the Smartphone also brings with it a rise in competing connected services which use an App on the phone as the central control unit for the user. The end product for the user therefore presents itself packaged as a complete system including hardware and necessary software (both programmed into the hardware as well as Apps/Web Applications interacting with the hardware).

For the average end user, such a complete system can be sufficient and the most straightforward and simple solution if they want to make their home "smart". It comes with a full defined set of features and is in most cases installable and operable without requiring any deeper technological skills.

However, when one thinks further than one specific use case realised by a specific setup, the question quickly arises about flexibility, scalability and interoperability. Nowadays, Smart Services can often be integrated into a more complex process with the help of a Smart Assistant like the Google Home[4], or Amazon's Echo series[5], or Automation-and-Interoperability-Supporting services like IFTTT (*If-this-then-that*)[6].

Smart Assistants like the ones mentioned above might not come with integrated automation functionality, instead they are voice-controlled and offer a quicker and more natural way of triggering tasks in the smart home (i.e., "Alexa, turn on kitchen lights"). However, they allow the use of services such as IFTTT to extend their voice-controlled functionalities, as described on the Google Home Help page ([13]). Examples for the joint use involve setting up custom commands to trigger IFTTT routines via the Google Home using natural language. A natural spoken sentence like "Hey Google, it's time to wake up" could then trigger the activation of a coffee maker [13].

[4] `https://store.google.com/product/google_home`, accessed on 08.06.2019
[5] `https://www.amazon.de/gp/product/B06ZXQV6P8/ref=fs_aucc`, accessed on 08.06.2019
[6] `https://ifttt.com/`, accessed on 08.06.2019

IFTTT itself is a service supporting the creation of short automatic routines by connecting standalone apps with one another to communicate. This allows apps that normally are not connected to perform joint tasks in an automated and scheduled way; IFTTT serves as the control unit of such "Applets". In addition to the possibility of creating a new tailored Applet, users can also choose from the number of existing, ready-to-use Applets that are stored in the IFTTT's database [16].

The restriction with IFTTT is that it is limited in the length of the automation routine it can provide. So far, it only allows the pairing of up to two apps to automatedly perform certain routines. Therefore, the limit for scaled processes is reached quickly.

A process management system for Smart Home like the one prototyped in this piece of work could provide an alternative to services like IFTTT in the sense that it allows similarly "applet"-like automation routines, but in a larger and more flexible scale. A potential scenario for its use includes providing various "Smart Services" that can be picked and graphically combined to a flexibly designed Smart Home Process in a drag-and-drop like manner via some User Interface (i.e., a web application wrapping the Activiti Modeler). This could result in a valid BPMN graph which is then automatically translated into a deployable code base implementing the services and the automated process.

3.4 Benefits beyond Comfort

While it is valid and not uncommon to integrate Smart Things into one's home for the sole purpose of increasing the comfort and Quality of Life, there are other motivators for the research into the field of Smart Home and Smart Home Automation that go beyond comfort only.

Domestic Care for the Sick, Disabled and Elderly is a field in which Smart Home Automation has been found to have the potential to help in various ways. For example, a smarter home can allow the elderly to live in their own homes for longer without the help of an often expensive caretaker, just by taking over the coordination of certain tasks they might not be able to perform on their own anymore. A similar argument can be made for disabled people. Not only does such an alternative save costs for the residents and their family members, it also gives the affected individuals an increased sense of control over their own lives [11].

CHAPTER 4

Chosen Automation Processes

A requirement for the processes to be modelled was that they are chosen to a large enough extent. This means that they should include a sufficient number of subtasks making up the overall process. Further, they should be set in the context of Smart Home Automation, as this is the chosen field of research.

The following two sections describe the two processes which were designed to fulfill the aforementioned requirements, while offering a low enough complexity overall to allow for a simple prototyping approach to implementing the automation system.

4.1 Weekday Morning Routine

The following process serves as an example for the type of routines that might be described by users requesting the service of a business providing custom integration of Smart Home elements into the user's homes and lives. For the purpose of this thesis, a morning routine was created based on some common scenarios that are likely to occur on weekday mornings of a person with a 9-to-5 job, while also choosing tasks/features that are often used as examples in the Smart Home Automation context.

The Process is started on the trigger of the alarm. On the assumption that some form of Smart Assistant Speaker is available, a brief summary of useful information for the day is played, including calendar events scheduled for the day, weather forecast and a suggestion for an outfit as determined on the basis of the weather forecast.

A measurement of the air quality outside is being pulled and evaluated, to inform the resident via red and green LEDs whether or not it is recommended to do an outdoor morning fitness activity. This can be particularly helpful for people who have problems with their lungs, such as asthmatics for example.

By including the light control in the responsibility of the Control Unit as part of the process, a poll of the sunrise time provided by the weather service could regulate the

activation of lights in the home by only activating the motion sensors, which will give the signal to turn on lights in the room where movement was sensed, when the sun has not risen yet.

Finally, the system will listen for the resident's mobile device disconnecting from either the home WiFi network or the Bluetooth connection established with the Control Unit. Either disconnect will trigger an end to the process.

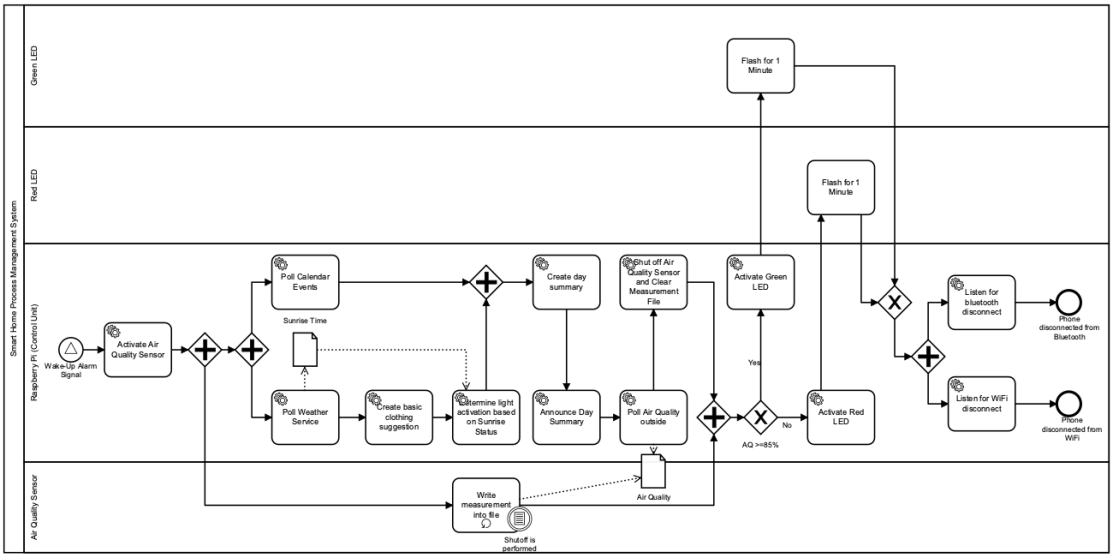


Figure 41: Weekday Morning Routine Process Model

Figure 41 shows the **BPMN 2.0 diagram** that was designed based on the description above. A *BPMN Pool* encloses the tasks that belong to the process to be automated, the *lanes* represent the different device participants in the system. The automated part of the process is found in the Control Unit lane.

4.2 Weekday Night Time Routine

Again, as mentioned in Section 4.1, a user might describe a nighttime routine based on the following, which could be supported by a Smart Home Process Management System. Typical 'getting-ready-for-bed' actions were paired with common Smart Home scenarios to offer a useful basis for the research.

As it is hard to identify a clear moment for when it makes sense to start the Nighttime Process, the start event was defined to be the user's phone connecting to the Home WiFi after some specific time of day (for example 6pm, in order to avoid starting the night time process too early; a resident generally enters their home more often than just after work). The Night Time Routine Process starts off by the Control Unit pulling the first few entries of the next day in the resident's calendar, as well as the weather forecast. The sunset time included in the weather data is checked against the current time, to decide whether the motion sensors in the house should be on to switch on lights as the user moves around, or not. Using the earliest of the polled events, the Control Unit calculates a suggested wake-up- and bedtime for the resident and notifies them by means of an email with the information. In order to ensure that the message has been received and read, the email provides the user with a confirmation link they should visit to confirm receipt and trigger the continuation of the process.

In the fifteen minutes leading up to the calculated bedtime, several preparatory tasks are scheduled. Devices like smartphone, laptop and/or tablet that are needed the next day should be connected to their chargers. The Control Unit takes care of switching on the powerpoints to provide the necessary electricity, and reminding the resident via email to charge the devices. The air quality sensor then performs a measurement in the room to determine if the windows should be opened to let some air in (display information to user in the form of an LED as mentioned in 4.1).

Finally, the collected information for the next day, including scheduled events and weather forecast, are reported by the Smart Assistant. At the set bedtime, the Control Unit switches off the connected powerpoints powering things like kitchen appliances or devices that would otherwise be on standby overnight (e.g., TV), and any lights that are still on. This marks the end of the Weekday Night Time Routine. Figure 42 shows the **BPMN 2.0 diagram** that was designed based on the description above.

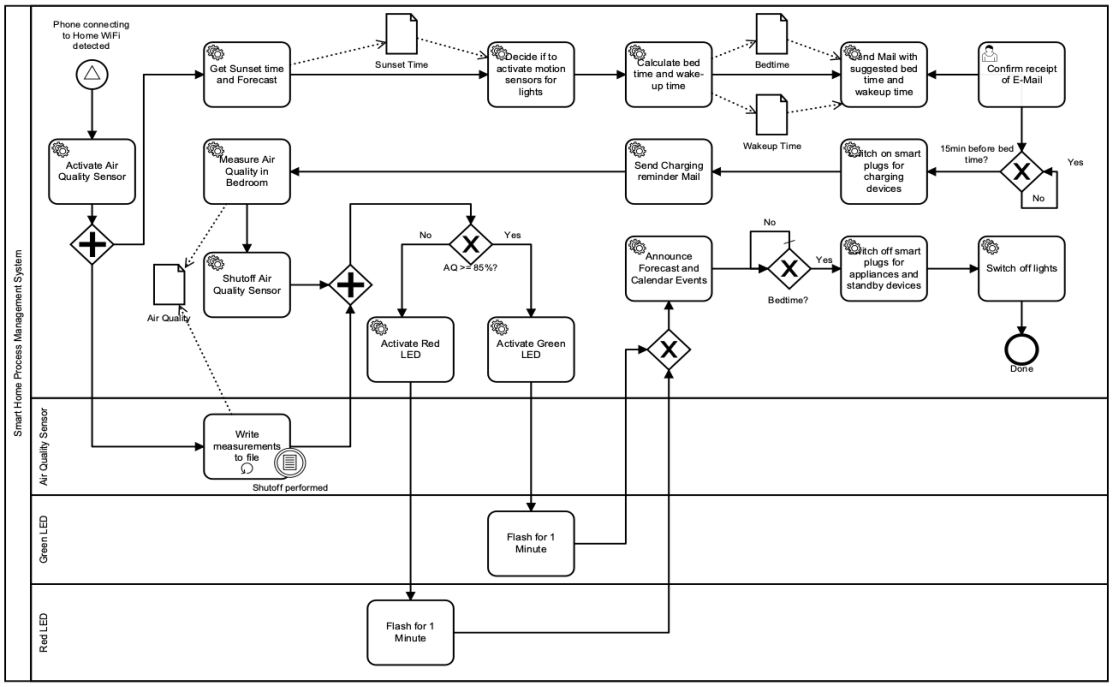


Figure 42: Weekday Night Time Routine Process Model

CHAPTER 5

Hardware Setup

5.1 Raspberry Pi 3 Model B+

The Raspberry Pi 3 Model B+ is a single-board computer provided by the Raspberry Pi Foundation. It is the last revision of the third-generation Raspberry Pi range.

Some of the most relevant specifications as found on the official Raspberry Pi website [10]:

- 1.4GHz 64-bit quad-core processor
- 1GB LPDDR2 SDRAM
- Dual-band wireless LAN
- Bluetooth 4.2/BLE
- Power-over-Ethernet support
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- Extended 40-pin GPIO header
- 5V/2.5A DC power input
- 4 USB 2.0 ports

The Raspberry Pi single-board computers were specifically made for the purpose of facilitating the learning experience of amateur programmers and providing a cheap and handy small computer anyone can use to quickly create prototypes for their own creative ideas, for work and play alike [9]. This also made it a good choice for attempting to using it as the Control Unit of the prototype Smart Home Process Management System.

5.2 LEDs

Used as a flashing signal for the user, two 5mm LEDs in green and red, with a 220 Ω resistor each, were plugged to the 3.3V and GND pin of the Raspberry Pi. They can be powered by explicitly controlling the voltage GPIO programmatically.

5.3 BME680

The BlueDot BME680 Environmental and Gas Sensor allows measurements of temperature, humidity, pressure and altitude as well as volatile organic compounds (VOCs) in the air. This combination of measurements makes it possible to calculate the air quality in the surroundings of the sensor [8]. The connection to the Raspberry Pi was done via I2C. There are six pins available on the sensor:

- **VCC pin** gets connected to either 5V or 3.3V pin of the Raspberry Pi
- **GND pin** gets connected to one of the GND pins on the Raspberry Pi
- **SDI pin** gets connected to BCM 2 (SDA) pin on the Raspberry Pi
- **SCK pin** gets connected to BCM 3 (SCL), the i2c clock pin of the Raspberry Pi i2c pins
- **SDO pin** is optional; it defines the I2C address used for the communication with the Raspberry Pi. By default, the I2C address 0x76 is used, if the SDO pin is connected to a GND pin on the Raspberry Pi, the address is set to 0x77.
- **CS** is to be left unconnected

5.4 HC-SR501

The HC-SR501 is a so-called *Passive Infrared (PIR)* motion detector, and it can be operated on its own or interfaced to a microcontroller. The sensitivity of the sensor can be adjusted to cover an area in a radius from 3 to 7 meters. The output can also be adjusted to stay active from 3 seconds to 5 minutes [22].

There are 3 pins to be connected on the motion detector's board:

- **VCC pin** gets connected to a 5V pin on the Raspberry Pi
- **OUTPUT pin** gets connected to BCM 4, or physical pin 7 on the Raspberry Pi; when the output is LOW, no motion was detected, conversely when it's HIGH, motion was detected.
- **GND pin** gets connected to a GND pin on the Raspberry Pi

CHAPTER 6

Smart Home Process Management System

6.1 Problem Statement

In order to define the Problem Statement for this thesis, it is useful to repeat the two core questions fueling the research and the project itself: *Can Smart Home Automation be viewed as a process in the context of Business Processes, or which processes can be identified in the Smart Home* and *How can such processes be developed flexibly and efficiently?*

The first question can be explored in two parts: Trying to define a few processes in Smart Home contexts using the methods of BPMN as done in Chapter 4, and using such trial processes in an implementation of a system that should be able to handle the Process Definitions and execute them automatically.

The second question motivates the search for the right tools and frameworks, but also the right circumstances needed to achieve a flexible and efficient development environment for a potential use case such as BPMN in the Smart Home.

The ideal outcome of this attempt to combine Business Process Model and Notation (*BPMN*) tools and frameworks with realistic Smart Home Automation Use Cases would be a functioning prototype of a Smart Home Process Management System (*SHPMS*) that runs on a Raspberry Pi, is able to work with valid BPMN Process Definitions and automatically execute those definitions, inlcuding control over different participants of a larger-scale Smart Home setup in the form of sensors and actuators common to the Smart Home. Within the scope of this thesis, successfully creating such a prototypical SHPMS is the goal.

6.2 Implementation

6.2.1 Development Environment

In order to figure out if it is possible to put the chosen BPMN engine Activiti into operation on a Raspberry Pi for the Smart Home processes textually defined in Chapter 4, the first step was setting up the development environment.

The RaspberryPi is limited in its computing power as well as its speed when it comes to running programs with a graphical interface (which often requires a certain amount of RAM available to the machine; the RPi 3 B+ only features 1GB of RAM, but as an example, the IntelliJ IDEA Java development environment by Jetbrains [1] lists a minimum of 2 GB RAM and a recommended 8 GB RAM in its System requirements).

In order to facilitate and speed up the process of writing the code that is to be run on the Control Unit, the choice was made to do the development part on a laptop with sufficient computing power and RAM, and mirror that code base to the RPi for when it has to be run over there once it is considered production state. The laptop of choice was the author's MacBook Pro 2018 [2] with a 2,3GHz Intel Core i5 processor and 8GB RAM. In the following pages, this laptop will be called *Development Laptop* in order to differentiate it from the Control Unit.

For the Java code needed to use the functionality of the Activiti Engine, IntelliJ IDEA by Jetbrains (as mentioned above) was picked as the preferred IDE. It comes with a built-in deployment tool that allows configuring an ssh connection to a server that the code should be deployed on, making it easy to mirror all the files of the SHPMS project onto the Control Unit, the RPi.

6.2.2 Process Models

Creating the BPMN2.0 Models of the processes was done using Activiti's own modeler application running on the Development Laptop. The Activiti Modeler can be executed as a standalone application, the code base for it can be found on Github [3]. The *README* file in the repository lists a few options of running the application, however, a different way was chosen in this case.

Activiti offers a collection of examples of their various application and usage types that can be run via Docker in its *Activiti Cloud Examples* repository [1]. Using the main *docker-compose.yml* file with a specific target, the chosen application can be built and run without having to worry about performing the compile and build steps manually (which is especially useful when unfamiliar with the concrete code and project itself). Additionally, a *Makefile* was created to abstract from any Docker commands.

[1] `https://www.jetbrains.com/idea/download/index.html#section=linux`
[2] `https://support.apple.com/kb/SP775?locale=en_GB`
[3] `https://github.com/activiti/activiti-modeling-app`

Provided the computing device of choice has Docker installed, the following two steps should be enough to start up the modeler application:

1. Clone the Activiti Cloud Examples from the Github Repository [1].
2. Follow the steps described in [3].

The command

```
make modeler
```

should then be enough to start up the Activiti Modeler Application in the browser.

In the Modeler, new projects can be created and processes modelled in the scope of those projects. The elements available for the models are the BPMN2.0 elements that the Activiti version the modeler application runs in supports. In the case of version 7, the one used in the scope of this project, not all standard BPMN2.0 elements were available for the process models. When creating the models for this project, it was therefore necessary to make some alternative choices for cases that would normally be modelled using specific elements that were not available in the used version.

One such element was the *Timer Event*. In the standard BPMN2.0, a *Timer Event* can be used to model scenarios where the process flow is determined by a certain Date or Time, which triggers the sequential continuation of the process. A *Timer Event* can only be catching starter events or intermediate events. In the case of the process model in Figure 42, there were two occasions where such an intermediate Timer Event would have been the correct element to choose.

Once the process models are done, they can be downloaded as xml files. These can then be added to the Java Activiti project as resources.

6.2.3 Activiti Setup

The Java Activiti Project is where the implementation of the SHPMS was realised using the Activiti API with Spring Boot integration. The basic project setup was performed with the help of the *Guide to Activiti with Java*[4] and the *Introduction to Activiti with Spring*[5] tutorial by **Baeldung**[6].

The dependencies needed were handled by Maven. The project's `pom.xml` includes Spring Boot, Activiti and Database dependencies, as well as dependencies needed for the implementation of various Service Tasks, for example *Google API* dependencies for Calendar Event fetching.

[4] `https://www.baeldung.com/java-activiti`
[5] `https://www.baeldung.com/spring-activiti`
[6] `https://www.baeldung.com/about/`

6.2.4 Code

Figure 61 shows the project structure of the SHPMS prototype application.

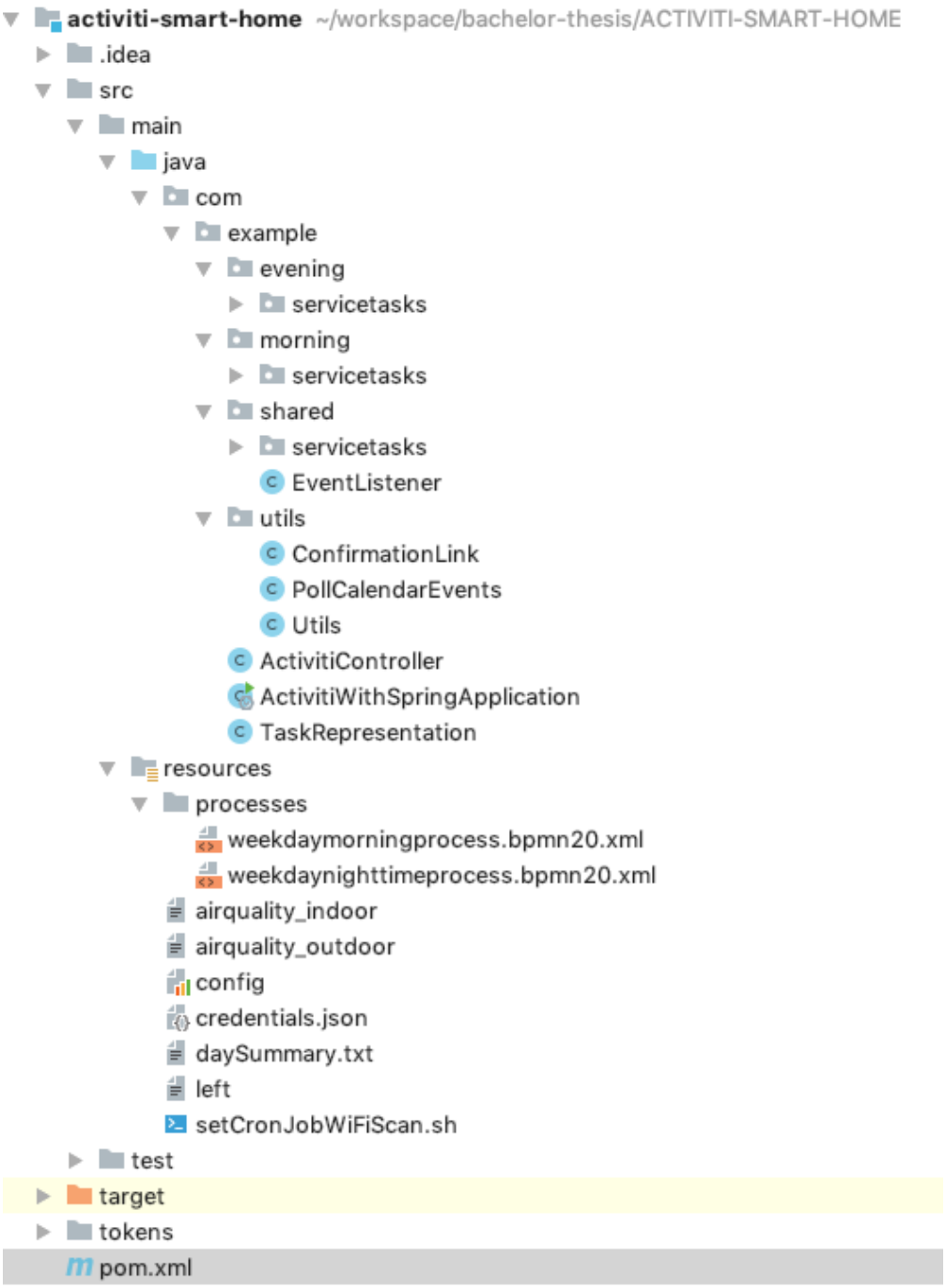


Figure 61: SHPMS Project Structure

Among other files, the `example` directory contains the Java Classes for the Service Tasks of the two sample processes, as well as the two main Spring Application Classes, `ActivitiWithSpringApplication` being the starting point of the application, and `ActivitiController` providing the Process Handling.

The `ActivitiController` is a Spring *@RestController* Class providing routes used to interact with a deployed Activiti Process.

- **`/start-process`** A `GET Request` to the route will create a new Process Instance (from a Process Definition key) using the **RuntimeService** provided by the Activiti API.

- **`/get-user-tasks/:processInstanceID`** A `GET Request` to the route will return a representation of all UserTasks found in a given Process Instance. UserTasks are tasks that need to be executed manually by a user.

- **`/readconfirmation/:processInstanceID`** A `GET REQUEST` to the route will complete the UserTask "Confirm receipt of E-Mail" of the given Process Instance, using the **TaskService** provided by the Activiti API.

Additional routes can be defined as needed, for example if there are more UserTasks which need to be completed.

When a Process Instance was started, the RuntimeService follows the Sequence Flow of the Process Definition and step by step handles the elements from the model. The Process Definition in turn contains information on what should happen when certain elements are reached. Service Tasks, for example, should be bound to a piece of code, be that a full Java Class or just a method, which serves as the implementation of either the service directly or an interface to an external service. An example of such a binding looks as follows:

```
<serviceTask id="ServiceTask_weather" name="Poll Weather Service"
activiti:class="com.example.shared.servicetasks.PollWeatherService"/>
```

The bound class `PollWeatherService` is a Java Class implementing the *JavaDelegate* Interface from the Activiti Engine library, and is used to perform the necessary steps to complete the service for the process. A JavaDelegate class has to implement the method `public void execute(DelegateExecution execution)`, which is executed when the Sequence Flow of the Process reaches that Service Task. It should contain the code necessary to complete the service task. Other methods can be defined additionally in the class if necessary. When the end of the `execute` method is reached, the service task is exited and the process continues to the next step.

Java Class	**Description**
PollWeatherService	Requests current weather data and sunrise time from the openweathermap.org API
PollDayEvents	Uses utils class `PollCalendarEvents` to send request to Google Calendar API to poll the upcoming events in the user's calendar.
LightActivationBySunStatus	Compares Sunrise time to current time to decide if the movement sensor for automatic light activation should be on or off.
CreateBasicClothingSuggestion	Uses polled weather data to create a clothing suggestion for the day based on temperature and weather status.
CreateDaySummary	Polled events and weather data are collected in a text file which is used as a Day Summary.
AnnounceDaySummary	The text file containing the day summary is being "announced" by printing it in the terminal.
PollAirQuality	Polls the airquality measured by the outdoor air quality sensor by parsing the value from the respective text file that the measurements are being written into.
ActivateRedLED	If the air quality measured is below a threshold set for a "good" level, a shell command is executed to start a python script that makes a red LED flash to signify the poor air quality.
ActivateGreenLED	If the air quality measured is above a threshold set for a "good" level, a shell command is executed to start a python script that makes a green LED flash to signify sufficiently good air quality.
ListenForBTDisconnectDetection	If the resident's phone was connected to the Pi using Bluetooth, and the connection breaks off because the resident has left their home, the Pi recognises the loss of connection and gives feedback to the SHPMS.
ListenToARPScanCronJob	Similar to the bluetooth task, if the resident's phone was connected to the same network the Pi is in, via WiFi and the connection is lost, a script will check and inform the SHPMS.

Table 61: ServiceTasks in the Weekday Morning Process

Java Class	**Description**
PollWeatherService	Requests current weather data and sunset time from the openweathermap.org API
LightActivationBySunStatus	Compares Sunrise time to current time to decide if the movement sensor for automatic light activation should be on or off.
CalculateBedWakeupTime	Uses utils class `PollCalendarEvents` to send request to Google Calendar API to poll the first few events of the next day in the user's calendar. Based on the first event, a wakeup- and bedtime is calculated and set in the process as variables.
SendMailBedWakeupTime	Notifies the user of the calculated bed- and wakeup time by sending them an E-Mail containing the information.
PlugSwitch	Executes a shell command that will run a script to either disable or enable smart plugs connected to the system; the smart plugs are either used to charge devices (like laptop, tablet or phone) or powering appliances and devices.
SendMailChargingReminder	When the smart plugs for devices to be charged are enabled, an E-Mail is sent to remind the user to charge their devices.
PollAirQuality	Poll the airquality measured by the indoor air quality sensor by parsing the value from the respective text file that the measurements are being written into.
ActivateRedLED	If the air quality measured is below a threshold set for a "good" level, a shell command is executed to start a python script that makes a red LED flash to signify the poor air quality.
ActivateGreenLED	If the air quality measured is above a threshold set for a "good" level, a shell command is executed to start a python script that makes a green LED flash to signify sufficiently good air quality.
AnnounceTomorrow	Announces the first few events of the following day as well as the weather forecast, by parsing the contents of the summary and printing it to the terminal.
LightSwitch	Executes a shell command that will run a script to turn off smart lights connected to the system.

Table 62: ServiceTasks in the Weekday Night Process

Tables 61 and 62 list the Service Tasks implemented for the Weekday Morning and Weekday Nighttime Routine Processes (see Figures 41 and 42) with a short description of their purpose.

For certain situations, the Process Engine might need to keep track of certain values that are relevant to the execution of the process. This is where *process variables* become of use. Variables can be used directly in the Process Definition pre-deployment, however they can also be set and read during runtime through the Process Instance. An *Exclusive Gateway* can use a variable to check a conditional expression to determine which path to take in a conditional flow. If the variable is not set ahead of the process deployment, it can either be set directly in the following method:

```
ProcessInstance startProcessInstanceByKey(String processDefinitionKey,
Map<String, Object> variables)
```

or during *process execution*. The execution is the point in the process that is currently active. For example, a Service Task implementing JavaDelegate can access its own execution in the `execute` method. Using the currently active execution, one or more variables can be set

```
execution.setVariables(Map<String, Object> variables)
execution.setVariable(String name, Object value)
```

or read.

```
execution.getVariables()
execution.getVariable(String name)
```

The processes chosen for this SHPMS prototype both include elements where process variables are essential for the execution of the proper sequence flow. In both processes, an *Exclusive Gateway* checks the variable `airQuality` against a given threshold and its value, which was set in the previous ServiceTask "PollAirQuality", decides which action happens in the next step of the process - the flashing of a Red or a Green LED. Further, in the Weekday Nighttime Routine process (see Fig. 42), the process waits until certain moments (*15 mins prior to bedtime* and *bedtime*) before continuing with the process flow. This was, again, achieved through the process variable `bedtime`.

Coming back to the project structure of the Activiti-Smart-Home project once more (see Fig. 61), it is essential to note the existence of the `resources` directory. For Activiti to be able to deploy and execute a process, it needs a process definiton. It is the product of the Activiti Modeling step described in Subsection 6.2.2 and it should be stored in a subdirectory of the `resources` folder, called `processes`. This is where Spring Boot will be looking for processes to be deployed. When starting a Process Instance, the process key defined in the Process Definition is needed to identify which of the deployed processes from the `processes` directory should be executed.

6.2.5 Sensors and Actuators

The sensors and actuators described in Chapter 5 needed to be included into the execution of the processes in the Activiti-Smart-Home project. In order to do that, scripts were prepared on the Control Unit, onto which all sensors were attached for the prototype SHPMS. These scripts allowed for the sensors to be taken into operation, and write their measurements out, or acted as activation and deactivation switches of their functionality.

Since the hardware was directly attached to the Control Unit, and with the knowledge of where the scripts were located on the system, all that needed to be done was to execute the scripts via the application using a Java ProcessBuilder instance. The ProcessBuilder can be passed a command which it will execute on the system using the desired shell.

This functionality gives ample opportunity for any type of interaction with the system the application is running on. If the necessary components for a specific technical use-case are already available and ready-to-use on the system, a JavaDelegate class could simply serve as an interface to that use-case by sending commands to be executed to a ProcessBuilder instance. Not only can the commands be executed, but the ProcessBuilder instance can wait on them to finish and listen to any outputs that might occur.

In other words, in the case of a Smart Home setup with various sensors: So long as the Control Unit itself has a way of dealing with the sensors that are part of the setup, either via scripts or simple commands in a shell, the Activiti application can do the same by executing these commands or running the relevant scripts through a ProcessBuilder instance.

6.2.6 Launching the SHPMS Prototype

Once the hardware is set up as needed (see Chapter 5), the SHPMS prototype application needs to be launched on the Raspberry Pi Control Unit. If the application was developed on a *Development Laptop* or other device than the Control Unit, it should be verified that all files that belong to the project are up-to-date and available on the Control Unit before launching the application.

The Spring Application should contain a `pom.xml`, which would allow it to be controlled using **Maven** commands. The startup command for the application has to be executed in a terminal of the Control Unit from within the directory containing the `pom.xml`. The single command `mvn spring-boot:run` downloads all dependencies, compiles and builds the project and starts the application.

Due to the default Spring Security Settings, the access to the main application is only allowed through a login. Spring Security generates a new password every time the application gets launched, and prints it to the terminal in the logs. This password has to be copied and used in the Login page which appears when `localhost:8080` is accessed through a web browser. The default user is *user*.

After a successful login, the process is started by visiting:

```
localhost:8080/start-process
```

If the process has not encountered any problems during startup, the browser page should display the message: *Process started. Number of currently running process instances = 1*

The process engine will lead the program along the sequence flow of the running process, executing tasks as it goes. If it encounters a *User Task*, it will wait for manual feedback on its completion; in the SHPMS prototype, a UserTask comes up in the Weekday Nighttime Process (see Fig. 42) after the Bedtime Reminder Mail gets sent, and would be confirmed by a feedback from the User in the form of a confirmation link sent in the Mail, which triggers a GET request to:

```
localhost:8080/readconfirmation/{processInstanceId}
```

CHAPTER 7

Evaluation

This thesis has led to the development of a very basic form of Smart Home Project Management System with the main purpose of creating a Proof-of-Concept for its general feasibility. The Processes created for this attempt were kept simple with regards to the hardware used to simulate a *Smart Home System*. The sensors used were standalone pieces not coupled to any existing system solution, so they had to be individually connected and programmed to work with the Raspberry Pi. The Pi's limited GPIOs also required the additional use of a *Breadboard* so that all pieces of hardware could be connected on the correct pins.

It is debatable whether the scenarios used for the processes (the Weekday Morning Routine Process in 41 and the Weekday Nighttime Routine Process in 42) are realistic scenarios for a potential production-ready version of this prototype, however the focus of the thesis was kept on testing out if BPMN concepts and technology can be combined with concepts and technology in the Smart Home. Therefore, it was decided that it is sufficient to create prototype processes that simply represent common aspects of the Smart Home.

Activiti was the choice for the BPMN framework to be used for the evaluation of feasibility. Its compatibility with *Spring* and *Spring Boot* made it easily integratable into an application that would be suitable to run on a microcontroller such as the Raspberry Pi 3. The Pi was able to support the size and workload of the finished application; during the execution of the application its CPU load was measured at a high of around 75%. In the scope of the project, it cannot be said whether certain Service implementations might cause issues on the limited computing power and memory of the Raspberry Pi, but in any case, this factor has to be taken into account when making choices about the services that a Smart Home Project Management System such as this could support.

Lastly, the way the processes were modelled was not entirely conform to general BPMN2.0 guidelines and best-practices. The reason for that was the limits that Activiti version 7

used for the implementation of the application imposed on the process definitions. As per the Activitiy Developers Guide [5], the BPMN elements supported in the 7.1.x Release Train was:

- Start / End Events
- SequenceFlows (conditional, default)
- Service Task
- User Task
- Gateways: Parallel, Exclusive, Inclusive
- Call Activity
- Signal Intermediate Catch Event, Signal Intermediate Throw Event, Signal Boundary Event
- Embedded Subprocesses

For this reason, it was not possible to implement some parts of the processes with the appropriate elements available in the official BPMN2.0 standard because they would not have been supported by the Activiti Engine in the application. More precisely, the Nighttime Process (see Fig. 42) contains two *Exclusive Gateways* (15-min-before-bedtime and bedtime) with reflexive paths imitating a loop that is exited once the condition returns true, i.e. the correct time was reached. In the official BPMN2.0 standard, this situation would have been modelled using *Timer Events*, which fire when the defined time has come. Since Activiti 7.1.x did not support this element, the closest possible modelling option was chosen.

Another problem noticed was the usage of pools and lanes in version 7.x.x of the Activiti modeler and API. In order to model a situation or a process in detail with all participants involved, pools and lanes are used to show which participants perform which tasks in which overlying scenario. In the version of Activiti that was chosen for the development of the prototype in this piece of work, the modeler application (see 6.2.2) does not support lane tags, and therefore would not successfully load a model xml containing pools and lanes. An improvement for a potential future enhanced version of the SHPMS could be attempted to be achieved by trying out the SHPMS code base in Eclipse, a different Java IDE for which an Activiti plugin was developed that provides an application called *Activiti Designer*, supporting some more elements and a slightly different way of handling the Activiti framework. This would however potentially require a larger rework of the SHPMS and the work as it has been done, and was chosen not to be pursued in the scope of this thesis to not increase the workload for a limited remaining time.

Finally, the resulting prototype has successfully run through both process definitions on the Raspberry Pi 3, and therefore proved that the defined Smart Home Processes could be handled by the Activiti Process Engine on a microcontroller like the Raspberry Pi 3.

CHAPTER 8

Summary and Conclusion

In the scope of this thesis, a search for answers to the following two questions was performed:

1. Can Smart Home Automation be viewed as a process in the context of Business Processes or rather, which processes can be identified in the Smart Home?
2. How can such processes be developed flexibly and efficiently?

By creating a light-weight prototype of a Smart Home Process Management System with an Activiti-Spring-Application in Java and various sensors attached to a Raspberry Pi 3, it was attempted to find some answers to those questions.

In order to respond to Question 1, the terms *Smart Home Automation* and *Business Processes* were analysed and described, to gain a common understanding of the concepts that influence the situation. A quick look into the *Internet of Things* and its role in today's market and technological ecosystem underlined the potential of the evolution of what is currently being done with Smart Home and Smart Home Automation technology.

Eventually, based on the gathered information to the topics just mentioned, an attempt was made to create Business Processes out of identified Smart Home scenarios. The resulting processes can be seen in Figures 41 and 42.

The second question was the basis for the implementation of the SHPMS using the BPMN Engine and Framework Activiti (see 2.3.1). The models of the identified Smart Home processes were created using the *Activiti Modeler*, and these models, or Process Definitions as they are also called, were used as input for the *Activiti Process Engine* in order to create an automatic process execution in a Spring Application (see 6).
The prototype itself was kept simple due to it serving mainly as a Proof-of-Concept, however, based on the successful results described in Chapter 7, it is possible to start

looking further into a more evolved SHPMS that might have the potential to make the development of such Smart Home Process Management Systems more efficient and flexible.

For the future, a more sophisticated version of such a SHPMS might allow even users largely unfamiliar with the ins and outs of BPMN or without the expertise in configuring extensive, complicated systems of Smart Home Applications to set up or at least define processes that they would like the SHPMS to make available in their homes. It would abstract from the complexity of the workings of such a system and make it accessible and straightforward to automate scenarios using the benefits of Business Process Management and BPMN.

The fully-fledged SHPMS might offer an interactive interface allowing users to graphically define a process that they would like the SHPMS to configure and execute in the context of their Smart Home. Hereby, the BPMN engine Activiti by Alfresco would serve as the base BPMN support, providing both the modeling functionality and the API used by a Java Spring Application to perform the setup and execution of any given Process Definition.

A database containing common sensors and actuators that can be used for various types of Smart Home use cases might also contain sample implementations of scripts that are needed to get the sensors functioning on the Control Unit. Based on the use case a user would define in its process model, the SHPMS could extract the appropriate sensor and the corresponding script for the Service Task.

Another improvement to the prototype would be to exchange the Spring Security login with a newly generated password at every launch for a permanent login by the User of the SHPMS. The user would register themselves as an authorized user of the system at the beginning, and would provide their own credentials on initialisation of the process. This could also be done using an OAuth-based OpenConnect login. In that case, the user could login with their Google Account for example, which also gives instant access to the calendar in case they want to incorporate that into the process (as the prototype does with the Event Forecast).

These are just some of the improvements that could give this SHPMS prototype the potential to be a functional and effective system for bringing the benefits of Business Process Management into private homes in order to simplify the everyday processes in our lifes.

List of Figures

21 BPMN 2.0 Events 6
22 Wakeup Alarm Start Event 7
23 Parallel Gateway Morning Process 7
24 Exclusive Gateways from two Process Models 8
25 Service Tasks from two Process Models 9
26 User Task Nighttime Process 9

31 Smart Lighting Google Search Trends 13
32 Smart Thermostat Google Search Trends 13
33 Vacuum Robot Google Search Trends 14
34 Smart Fridge Google Search Trends 14
35 Nest Security System Google Search Trends 15
36 Smart Home Security Google Search Trends 15

41 Weekday Morning Routine Process Model 20
42 Weekday Night Time Routine Process Model 22

61 SHPMS Project Structure 28

List of Tables

21 Free and Open Source BPMN Tools as per [15] 5

61 ServiceTasks in the Weekday Morning Process 30
62 ServiceTasks in the Weekday Night Process 31

Bibliography

[1] Activiti cloud examples repository, . URL https://github.com/Activiti/activiti-cloud-examples. accessed on 23.5.2019.

[2] Activiti cloud components, . URL https://activiti.gitbook.io/activiti-7-developers-guide/components. accessed on 07.06.2019.

[3] Getting started - docker compose, . URL https://activiti.gitbook.io/activiti-7-developers-guide/getting-started/getting-started-activiti-cloud/getting-started-docker-compose. accessed on 13.04.2019.

[4] Getting started - activiti core, . URL https://activiti.gitbook.io/activiti-7-developers-guide/getting-started/getting-started-activiti-core. accessed on 13.04.2019.

[5] Activiti and activiti cloud overview, . URL https://activiti.gitbook.io/activiti-7-developers-guide/overview. accessed on 07.06.2019.

[6] Bpmn workflow engine, . URL https://camunda.com/products/bpmn-engine/. accessed on 11.5.2019.

[7] Bpmn reference, . URL https://camunda.com/bpmn/reference/. accessed on 11.5.2019.

[8] BlueDot. Bluedot bme680 environmental and gas sensor. URL https://www.bluedot.space/sensor-boards/bme680/. accessed on 30.06.2019.

[9] Raspberry Pi Foundation. Raspberry pi about, . URL https://www.raspberrypi.org/about/. accessed on 01.08.2019.

[10] Raspberry Pi Foundation. Raspberry pi 3 model b+ specs, . URL https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. accessed on 01.08.2019.

[11] Eric Franck, Joram Nauta, and Robin de Haan. Business case for smart homes. pages 413–426, 2017. doi: 10.1007/978-3-319-01583-5_52. URL https://doi.org/10.1007/978-3-319-01583-5_52.

[12] Denis Gagné and Simon Ringuette. Bpmn quick guide. accessed on 18.4.2019.

[13] Google. Control your home. URL https://support.google.com/googlehome/answer/7194656?co=GENIE.Platform%3DDesktop&hl=en. accessed on 08.06.2019.

[14] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2013.01.010. URL http://www.sciencedirect.com/science/article/pii/S0167739X13000241. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services adn Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond.

[15] Moritz Hesse. Bpmn tool matrix. URL https://bpmnmatrix.github.io/. accessed on 14.5.2019.

[16] IFTTT. Ifttt. URL https://ifttt.com/about. accessed on 08.06.2019.

[17] Object Management Group Business Model Management and Notation. Official bpmn website. URL http://www.bpmn.org/. accessed on 10.3.2019.

[18] Max Roser and Esteban Ortiz-Ospina. Our world in data. URL https://ourworldindata.org/world-population-growth. accessed on 24.4.2019.

[19] Jyothi Salibindla. Intelligent business process management. *International Journal of Engineering Research and Technology*, 6:335 – 338, 10 2017. ISSN 2278-0181.

[20] Biljana L. Risteska Stojkoska and Kire V. Trivodaliev. A review of internet of things for smart home: Challenges and solutions. *Journal of Cleaner Production*, 140:1454 – 1464, 2017. ISSN 0959-6526. doi: https://doi.org/10.1016/j.jclepro.2016.10.006. URL http://www.sciencedirect.com/science/article/pii/S095965261631589X. accessed on 20.4.2019.

[21] Charlie Wilson, Tom Hargreaves, and Richard Hauxwell-Baldwin. Benefits and risks of smart home technologies. *Energy Policy*, 103:72 – 83, 2017. ISSN 0301-4215. doi: https://doi.org/10.1016/j.enpol.2016.12.047. URL http://www.sciencedirect.com/science/article/pii/S030142151630711X.

[22] DroneBot Workshop. Dronebot workshop hc-sr501 tutorial. URL https://dronebotworkshop.com/using-pir-sensors-with-arduino-raspberry-pi/. accessed on 30.06.2019.

APPENDIX A

Instructions for Installing SHPMS Prototype on Raspberry Pi 3

The following serves as a guideline for how to install the Activiti-Smart-Home Project Management System Prototype on a Raspberry Pi 3.

A.1 Preparation

Before using the project on the Raspberry Pi, the Pi should be upgraded and updated using the commands:

```
sudo apt-get upgrade
sudo apt-get update
```

The project uses the JDK11, which is usually not by default installed on the Pi. Therefore, it might be necessary to download it from the web and install it to the system. The following steps were found on `hirt.se/blog`[1]:

1. Get the latest JDK 11 build of the Liberica JVM (Debian package for ARM v7 and v8) which can either be downloaded directly from the web browser at `https://www.bell-sw.com/java.html` or using the terminal:

   ```
   wget https://github.com/bell-sw/Liberica/releases/download/11.0.2
   /bellsoft-jdk11.0.2-linux-arm32-vfp-hflt.deb
   ```

2. Install it, for example:

   ```
   sudo apt-get install ./bellsoft-jdk11.0.2-linux-arm32-vfp-hflt.deb
   ```

[1] `hirt.se/blog/?p=1116`

3. Set defaults if required:

```
sudo update-alternatives --config javac
sudo update-alternatives --config java
```

If an ssh connection to the Pi is required or desired, the respective interface has to be enabled on the Pi.
Lastly, the project should be transferred onto the Pi if it was developed on another device.

A.2 Steps

1. Navigate into the project directory to where the `pom.xml` is located.
2. In the terminal, execute the command `mvn spring-boot:run`. It downloads all dependencies, compile and builds the project and starts the application. The Spring Application logs will show up in the terminal.
3. Due to the default Spring Security Settings, a login is required to authenticate for REST requests to the application. Spring Security generates a new unique password every time the application is launched, and prints it to the terminal inbetween the logs.
   In a browser, visit `localhost:8080`, it will redirect to the login page. The default username is `user`, the password can be found in the terminal as explained above.
4. If the login was successful, and the application runs without errors, navigating to `localhost:8080/start-process` will start the process that was defined in the code. In this version of the SHPMS application, two routes that are defined are:

   - `/start-process`
   - `/readconfirmation`

   Other routes (including the default empty route) will show a *Whitelabel* Error page, which can safely be ignored as long as the logs indicate a correctly running application. The process will be executed step by step.

APPENDIX B

Instructions for Creating a new Process in SHPMS Prototype

Creating a new executable process that will run on the SHPMS prototype requires two main tasks: *Creating the process model* and *implementing the code needed to successfully run the process model*.

B.1 Process Model

The SHPMS prototype runs on *Activiti Version 7.x*. Activiti is working on integrating more and more of the official BPMN2.0 elements into the API so as to allow execution of more complex prototypes. However, so far the number of elements supported is limited. Therefore, it is recommended to create the process model that should run on the prototype using a version of Activiti's own modeller application. The modeler can be started in a number of ways, the following steps describe the way chosen for the scope of the SHPMS prototype and require a running installation of Docker:

1. Clone the *Activiti Cloud Examples*[1] repository from Github.
2. Follow the steps described here:

   `https://activiti.gitbook.io/activiti-7-developers-guide/getting-started/getting-started-activiti-cloud/getting-started-docker-compose`
3. The application can be found when navigating to:

   `http://$(DOCKER_IP)/activiti-cloud-modeling`

[1] `https://github.com/Activiti/activiti-cloud-examples`

The login page requires username and password, by default that is: Username `modeler` and Password `password`

4. After modelling the process, it should be exported and saved as an `xml` file, and moved to the `resources/processes` directory in the SHPMS prototype application.

B.2 Implementation

The prototype application is structured in the following way:

- The Java classes containing the framework for handling the processes are the main part of the application.

  `ActivitiWithSpringApplication` does not need to be touched, it serves as the trigger for the Spring Application. The `ActivitiController` is the REST controller providing the routes that serve as trigger points for the process that is to be executed. Any routes similar to `/readconfirmation` used to confirm the completion of a manual user task, should be defined here. `/start-process` starts the process with the *Process Definition Key* set in the `xml` file of the process that should be executed (the attribute `id` in the <process> tag).

  Implementations for any tasks in the process model that require some type of computing to reach their objective can be realised using Java classes implementing the `JavaDelegate` interface. It requires an implementation of the `execute` method, which should contain the logic of the task. The Activiti Process Engine executes this method when the respective task is reached in the sequence flow. In order for the Process Engine to know where the implementation of the Task can be found, it has to be bound to the ServiceTask in the Process Definition, which is the `xml` file of the model. An example is the following:

  ```
  <serviceTask id="ServiceTask_weather" name="Poll Weather Service"
  activiti:class="com.example.shared.servicetasks.PollWeatherService"/>
  ```

  Additionally, `EventListeners` can be used to handle incoming events by the process, i.e. the `EndEvent` that signals the end of the process execution.

- Any additional resource files needed for e.g. configuring the application, collecting output created by service tasks, etc. should be stored in the `resources` directory, as well as the process definition xmls.