Architecture and Design

Computer Engineering Project 2

A light guidance system

GROUP 4

Adrian Anthony, [redacted]
Simon Aggenem Andreasen, [redacted]
Peter Laust Almvig, [redacted]
Rune Schrøder, [redacted]

Ver. 2
23.05.2024

Contents

1 Introduction

This document provides a comprehensive overview of the system's structure, functionalities, and design principles. The document contains a thorough explanation of hardware and software components of the Glow2Go system, including explanations of the communication procedures, design considerations and deployment plan of the system. Included in the documentation of the architecture and design of the system, are several visualisations. This includes, but is not limited to: class-diagram, sequence-diagram, 4+1-diagram, state-diagram and a layer-diagram.

2 System Overview

Provided below is a very brief layout of the Glow2Go system. For more information, please refer to our 'Requirements Specifications' document.

The overall system can be split into two parts. The light guidance system running on the Raspberry Pi and the web application and database running on a separate server. The light guidance system will get its specified active hours from the database.

The light guidance system becomes active when it is within the active hours. The light guidance system will then light up a path to the bathroom for the resident and log time to bathroom, time in bathroom, time back to bedroom and total time to the database.

This data is available on the web application running on a server, where the active hours of the light guidance system can also be changed.

2.1 4+1 diagram

The system architecture as a whole can be described using Kruchten's 4+1 diagram using most of the sections and diagrams in this document to give the view abstractions from the diagram. this method and the model as a whole is described in detail by Ian Sommerville in Software-Engineering 10.ed section 6.2 'Architectural Views"[1].

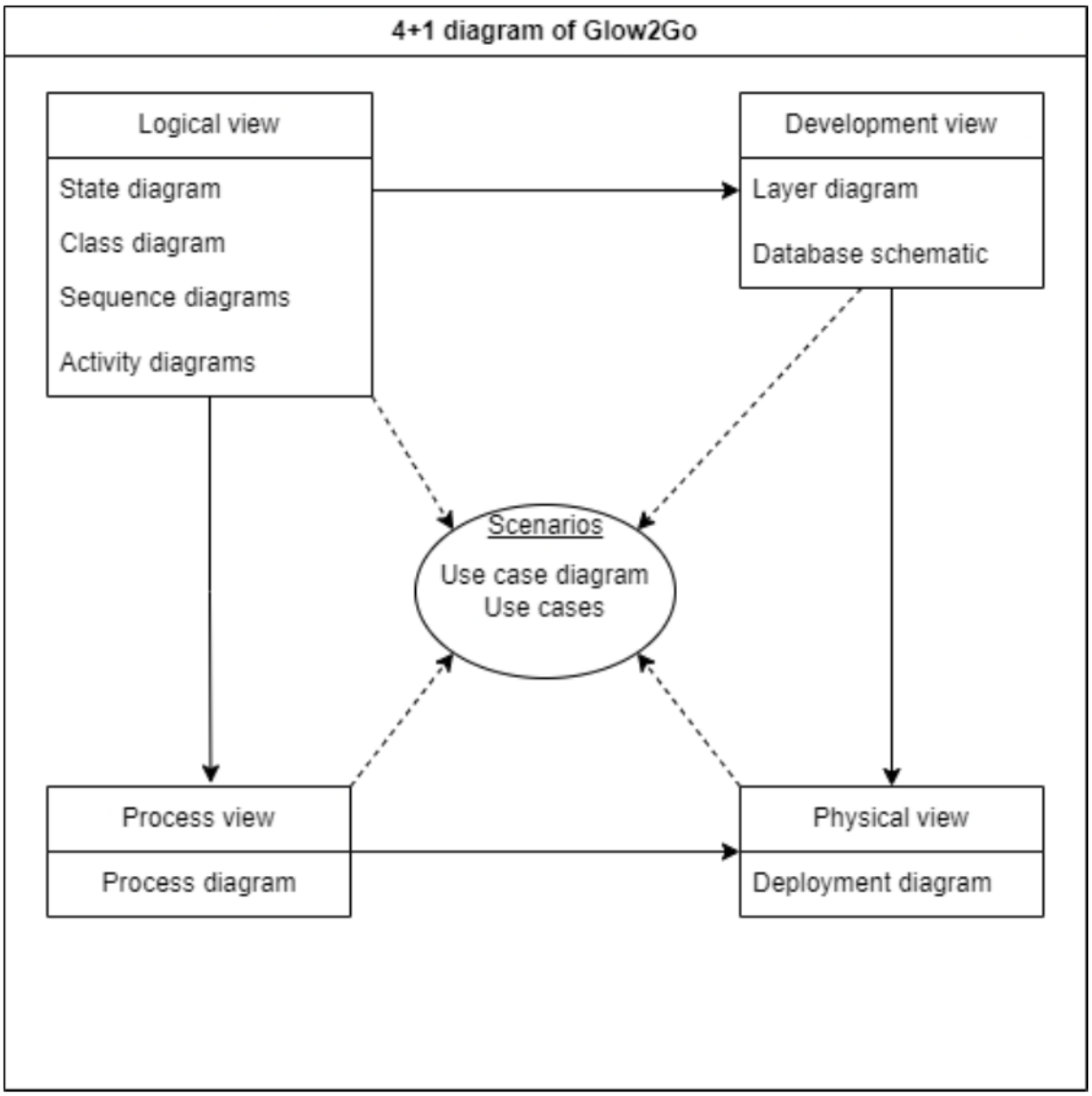

Figure 1: Krutchen's 4+1 diagram referring to different sections of this document.

The last view (+1) is the use case diagram which can be found in our requirements specification document Appendix 1.

3 Logical view

The logical view and the diagrams therein provides an abstraction that outlines the inner workings of a system. What are the inner workings of a system? How does it react to certain events and scenarios? These kinds of questions are emphasised in this view, and is outlined in diagrams based on the requirements specifications of the system.

3.1 State diagrams

The state diagram fall under the logical view in Kruchten's 4+1 view model and shows how the user can interact with the light guidance system.

The system has a few different levels of active that needs to be distinguished between. First, the system can be outside the active hours (by default the system is active $22:00-09:00$). When this is the case we call the system inactive. When the system is active but the user is not going to the bathroom (Or moving in general) we call the system idle. When the user is moving we call the system operating.

The system (when active) can be divided into four states. A full trip from the bedroom to the bathroom will enter all states of the system although the user can cut the trip short by not going to the bathroom.

Generally the system can be viewed as having 3 types of rooms. The bedroom, the bathroom and intermediary/transit rooms. The intermediary rooms are every room inbetween the bedroom and bathroom on the route.

This state diagram begins when the system is active and only handles the light guidance system.

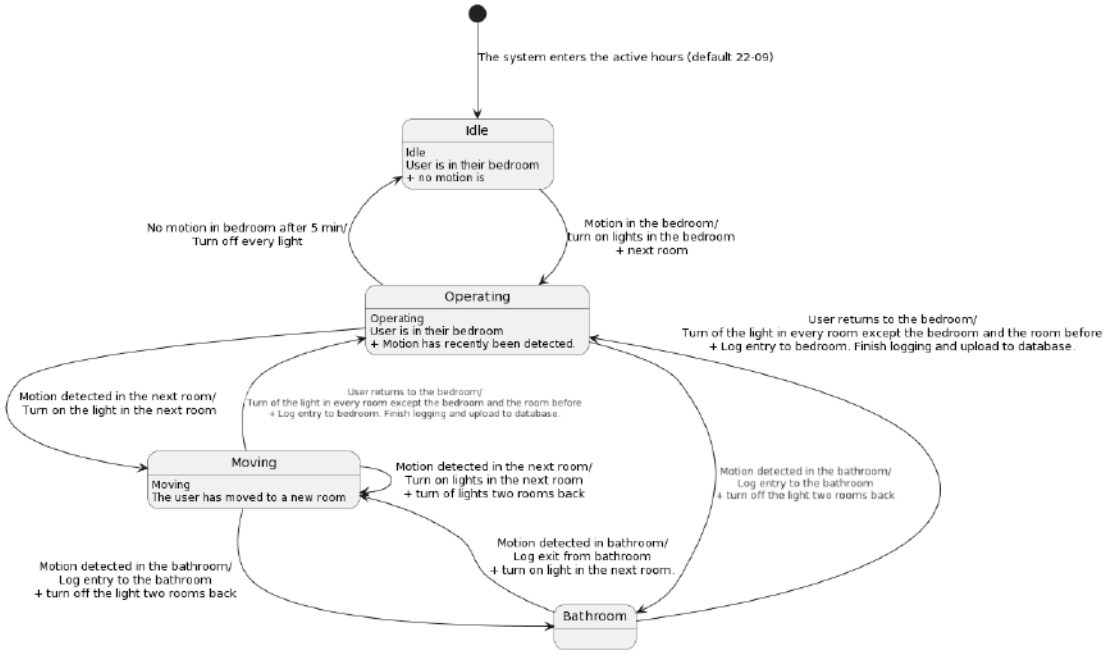

Figure 2: State diagram showing the different ways that a user might use the light guidance system.

The system starts in the `Idle` state, where the user is in the bedroom and no motion is detected. When motion is detected the system goes to the `Operating` state and the light in the next room turns on.
In the `Operating` state the user is still in their bedroom but motion has been detected. This can be seen as a staging state between waking up and going to the bathroom. Either no movement is detected after five minutes and the system goes back to `Idle` or the user leaves the bedroom.
If the bathroom is directly connected to the bedroom (meaning there's no intermediary rooms) the system directly moves to the `Bathroom` state when the user leaves the bedroom, skipping the `Moving` state.

If there's intermediary rooms the system goes to the `Moving` state. The system will stay in this state as long as the user is in an intermediary room or moving between intermediary rooms.
The system will leave this `Moving` state only when the user enters the bedroom or the bathroom. The user does not have to have entered the bathroom before going back to the bedroom and the case where he doesn't enter the bathroom but goes straight to the bedroom from an intermediary room is handled the same as if he entered the bathroom first.

From the intermediary state (`Moving`) if the user enters the bathroom the system will also enter the `Bathroom` state. Here the user can go back to an intermediary room or directly back to their bedroom if there is no intermediary rooms.
Returning to the bedroom from an intermediary room or directly from the bathroom is handled the same way.

3.2 Activity diagrams

Two activity diagrams have been made, one for the behaviour of the lights when moving between rooms. And one for the system behaviour when the resident enters the bedroom.

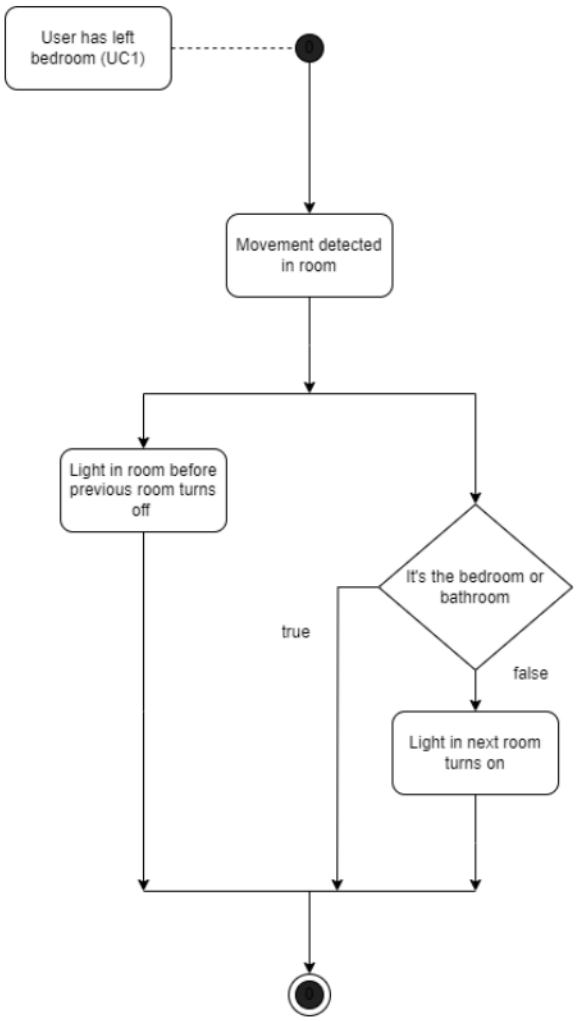

Figure 3: Activity diagram of light, when moving between rooms

This activity diagram is meant to give a high level understanding of how lights actuate when the resident enters a new room. Since UC1 is a precondition the light is on in the current room, previous room and next room (when the current room isn't the bathroom or bedroom).

1. **Resident enters bathroom or bedroom.**

   Lights are on in current room and the previous room.

2. **Resident enters an intermediary room between the bedroom and bathroom.**

   Lights are on in previous room, current room and next room.

When the resident enters a room the behaviour of the Guiding light system can be split into two cases.

The activity diagram below shows how the system should handle UC5, where the trip data is sent to the database and the lights in the bedroom turn off after 5 minutes off no movement.

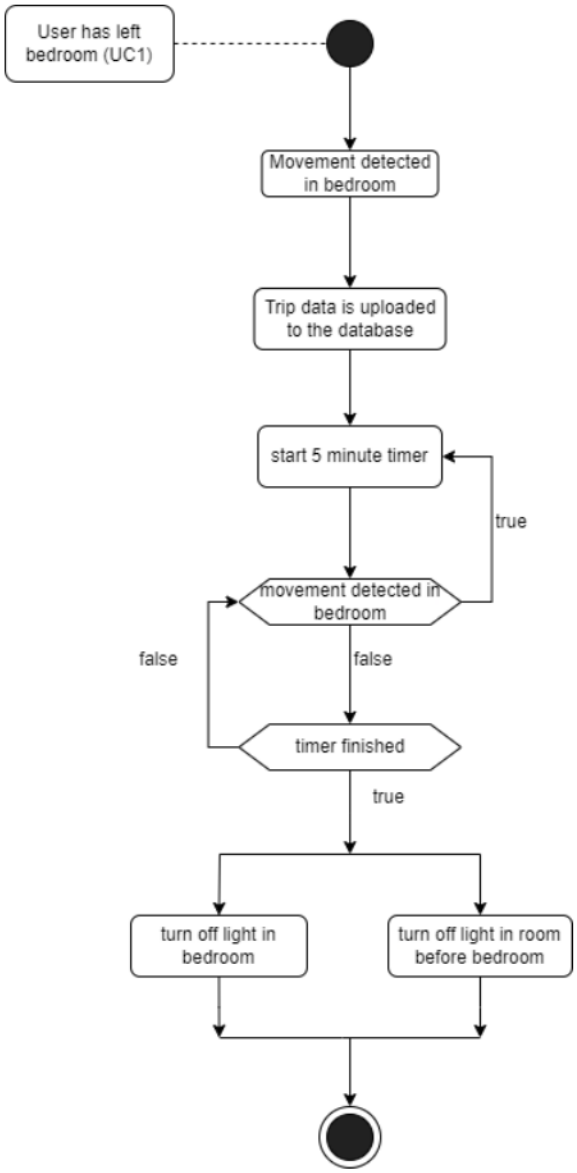

Figure 4: Activity diagram of UC5

The activity diagram for a specific functionality of the system provides an important abstraction for analysis of the general system behaviour, and thereby is good for better system formulation and understanding. It is also considered part of the Logical view in the 4+1 diagram for the system.

3.3 Class diagram

The need to model the current state of the system has become apparent upon the formulation of the state- and activity diagrams, where the need to have some sort of link between the sensors in the rooms has become apparent. To do this, we designed a system that has the basic principles of receiving an event, storing and processing the data, and then pushing the decisions made to the required part of the environment or database. This is visualized with a class diagram for the system, which is also part of the logical view in 4+1 diagram.

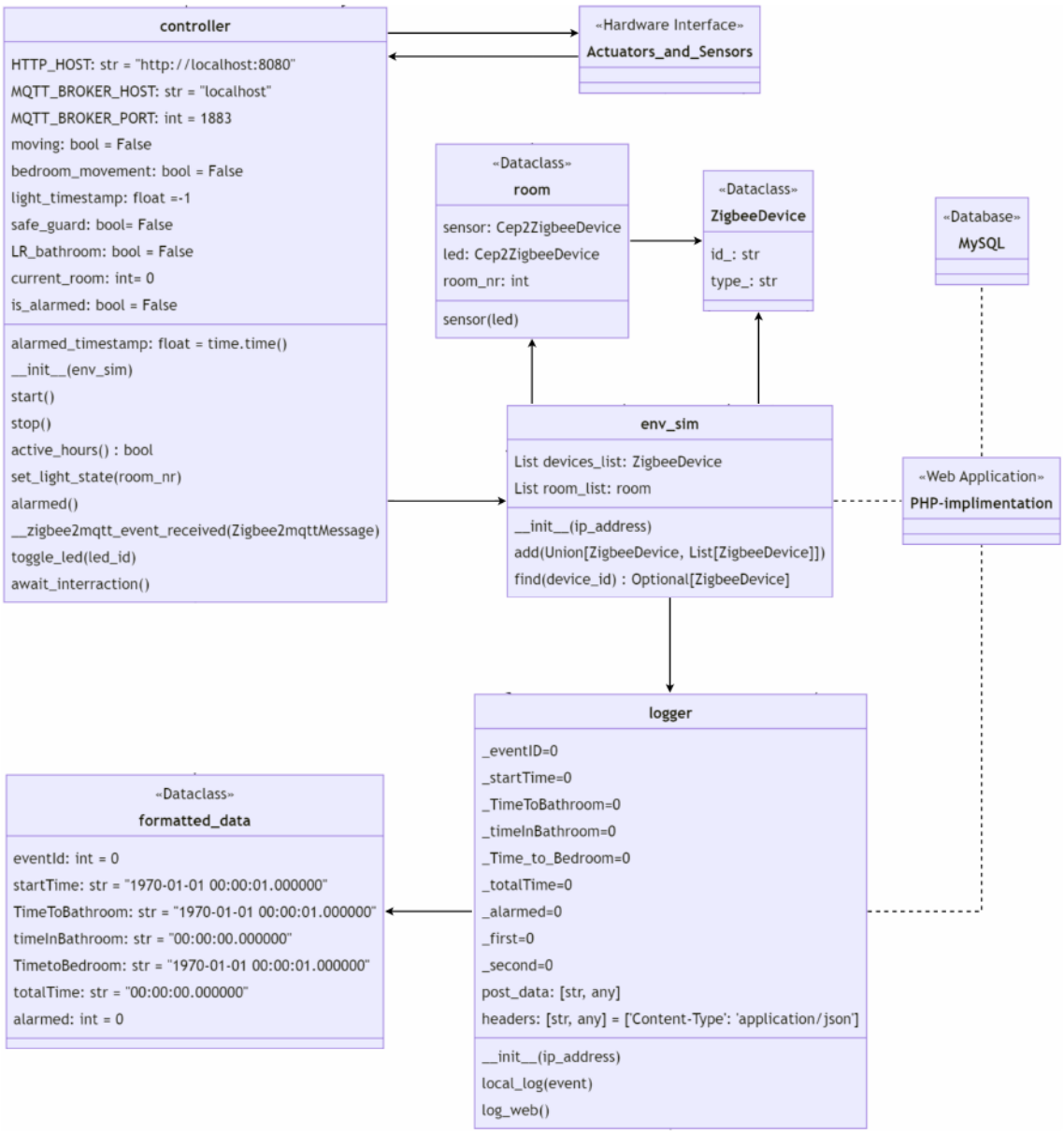

Figure 5: Class diagram, showing the linking of all classes used in software in the system

The system has a controller that is tasked with handling the in-and output to the

sensors and LED-strips through MQTT.
On the other side of this dynamic, we have created a logger that is responsible for reading the data in the environment model and logging it to the database through the PHP API. It should only be capable of writing to the database and not reading from it, since it would over-complicate the logger.
The last, and arguably most vital class of the software system is the environment model class class env_sim (should maybe have been named env_model instead). This contains object simulations of the physical sensors and LED's, and is capable of modelling the environment and is passed to the controller. The decisions made by the controller is thereafter read by the other components such that the states can be forwarded to the required destinations and the required lights can be turned on/off.

3.4 Sequence diagrams

With the classes of the system specified, we can formulate more concretely how the specific use cases should flow between the classes based on the activity diagram and use case scenarios from our 'Requirements and Specifications' document. This is done using the sequence diagram abstraction, which also provides to the logical view of the 4+1 abstraction.

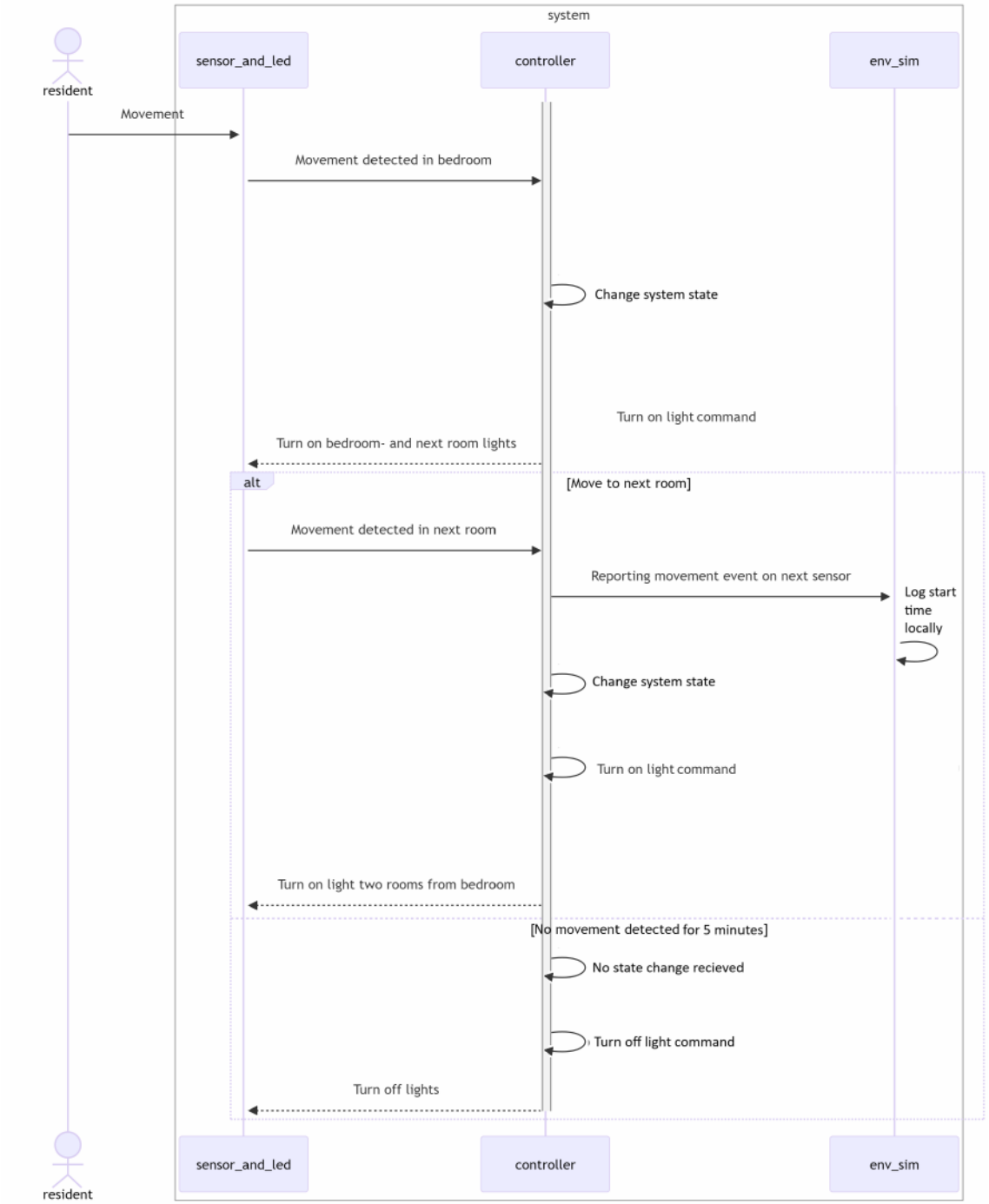

Figure 6: Sequence diagram for Use-case 1, leaving the bedroom, from the "Requirement specifications" document

In Figure 6, the sequence diagram for the flow of UC1 in the software components are displayed. This is the use case that displays the action of initiating the system, ie. changing the system from idle state to operating state. The diagram contains the specific events passed between the components, and how alternate sequences may occur based on the extensions from UC1. This in particular is displayed by the 'alt' path in the diagram, where the sequence may diverge from the main scenario

into the extension.

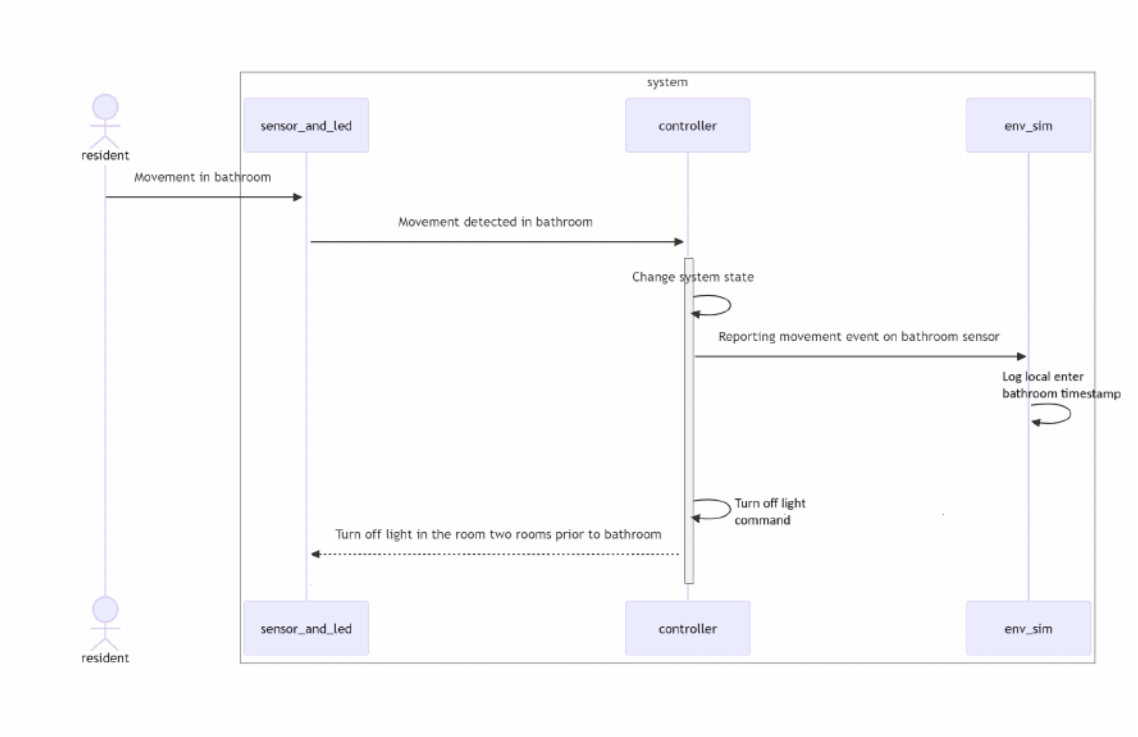

Figure 7: Sequence diagram for Use-case 3, entering the bathroom, from the "Requirement specifications" document

In Figure 7, the sequence of events between the software components are displayed in the same way as in Figure 6, but it covers the main scenario from UC3. UC3 outlines the use case for when the the resident enters the bathroom, and then how the light and data behavior is processed based on the events. This sequence diagram contains no 'alt' sequence, which is because of no presence of an extension in UC3.

4 Development view

The development view of the 4+1 diagram provides abstractions that eases the implementation of a system for the developer. It is especially a useful abstraction for the programmers of a system, since the UML-diagrams mm. of this view can elaborate on how different components should communicate and be separated from each other.
This is also elaborated upon by Ian Sommerville in section 6.2, "Architectural views"[1].

4.1 Layer diagram

As is stated in 6.3.1 of Ian Sommerville's 10th ed.[1], the layered diagram will in this case be implemented to provide the abstraction of independent process separation. The lower layers will be responsible for providing the designated services to the entirety of the upper layers, and the layers in themselves will have a specific task associated with it.

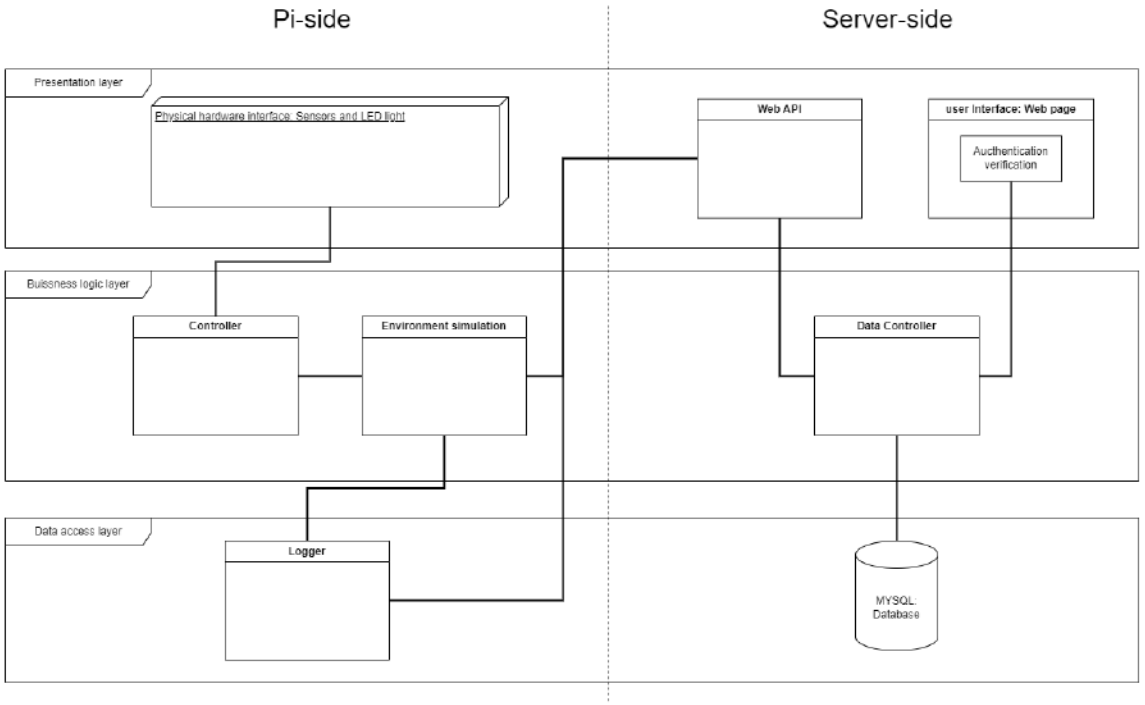

Figure 8: The layer diagram of the system

In our adaption of the diagram, we provide a presentation layer, where its only through theses specified interfaces that the system can receive and display data or forward physical processes, such as turning on an LED-strip. The system change caused in these interactions will always be as a result of the lower laying logic and services from the below layers.
The business layer in this context more or less consist of our software components, and included in this is the main simulation component. This layer therefore contains all the decision-making processes and data forwarding to the physical hard-

ware components, and is arguably the most vital layer in the system. It therefore also reads and writes to and from the database, to log system status and evaluate possible system settings changes made through the user interface.
The database layer in our system more or less consists of only one component, the MYSQL database. Here, all the logged data is stored from the business logic, and is available to be displayed on the web page interface after the user has been authorized. It also contains the information about a given user of the system, and is available such that system parameters such as active hours can be changed.

4.2 Database Schematic

The database doesn't need to hold too much data per resident and consists of two tables. The database is a relational database. In the database schema below the two tables is shown and the primary keys for each table. How each table relates to each other is also shown.

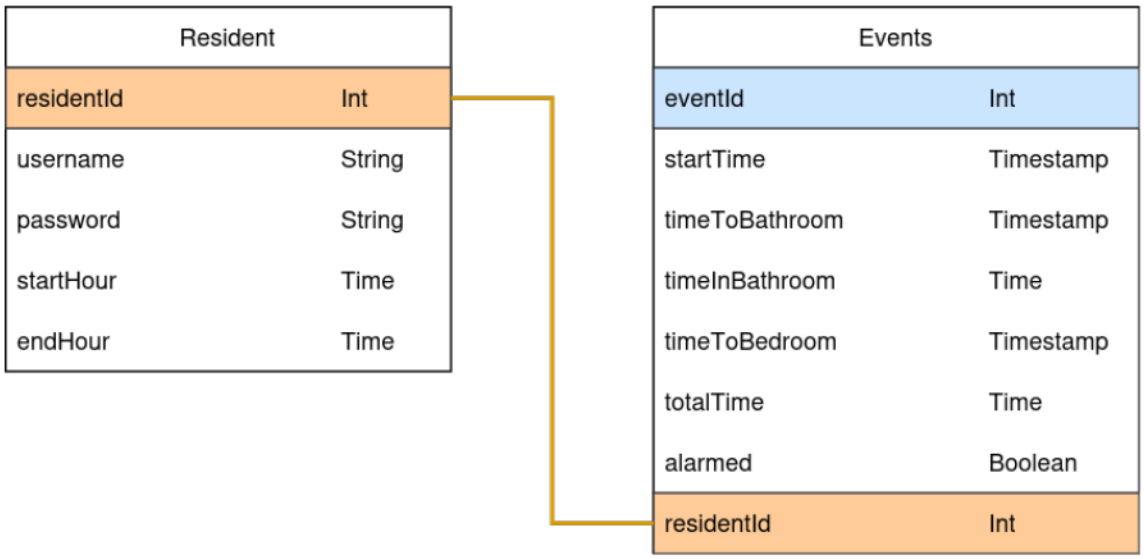

Figure 9: Relational Database Schema for the server showing what is stored and where.

The *Resident* table holds all the information about the user of the system, this being a unique id, a username, a password, and that users active hours. The *Events* table holds all the information we log from a trip and an associated residentId from the *Resident* table. This allows our systems to holds the data from multiple users.

This database schematic could also be considered part of a 'Data view' of an alternate n+1 abstraction model, since it provides a great inside into the organization of the data from the entire system.

5 Physical View

The physical view shows the distribution of the hardware and software components across the processors in the system, providing insight into the thought process of

the Glow2Go system deployment.

5.1 Deployment diagram

The deployment diagram, Figure 10, displays the deployment layout of the Glow2Go system installed in a house with 'n' number of rooms. As specified by the requirements document, the system is deployable on any number of rooms.

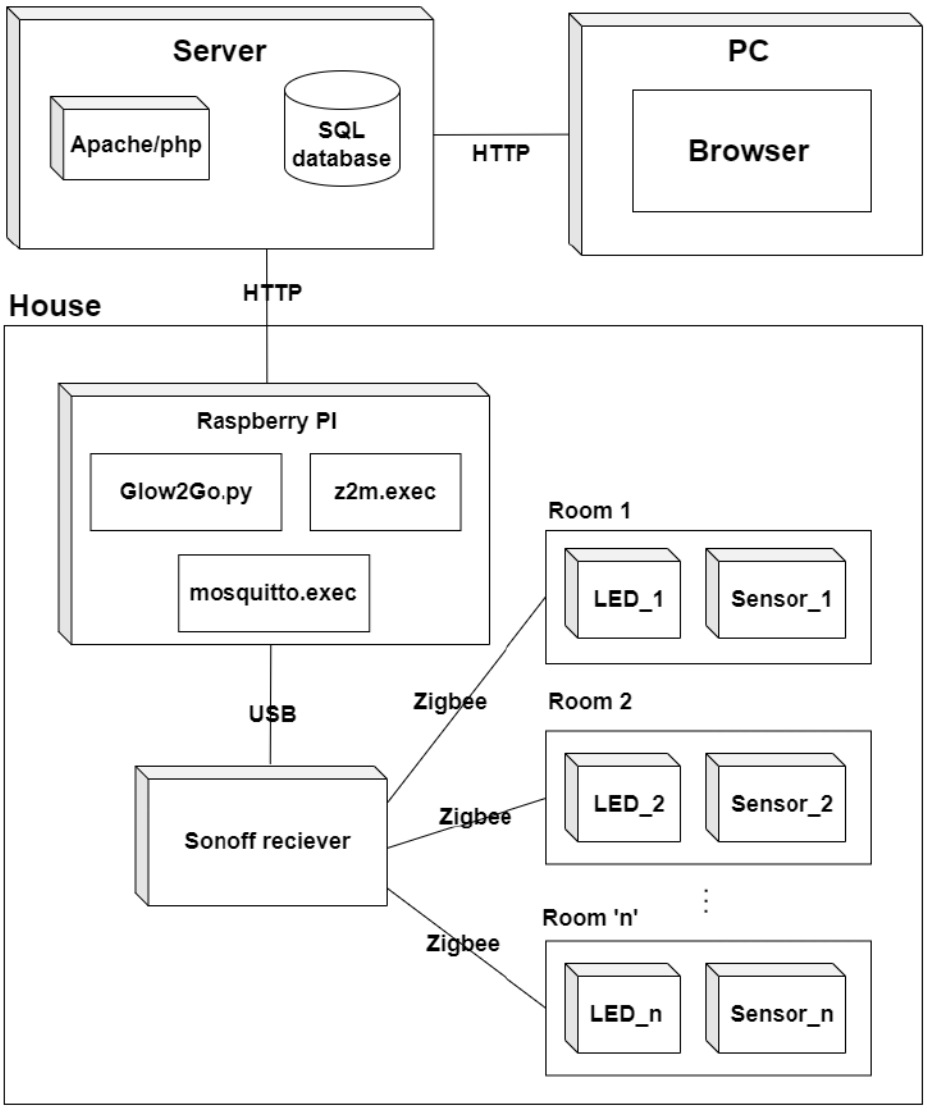

Figure 10: UML Diagram for the Glow2Go system deployment.

Each room has a paired sensor and a LED. The sensor and LED communicate to the Sonoff receiver using the Zigbee connection protocol. The receiver passes this received signal to the Raspberry PI through a USB cable. On the Raspberry PI three services are running; The Zigbee2Mosquitto service, the Mosquitto service and the Glow2Go system service. These services translate the received Zigbee signal to MQTT and then make decisions about which lights to turn on/off. Relevant

data (specified by the requirements document) is then logged to the web-API using HTTP, and is then stored in the SQL database. The logged data can then be accessed by users or admins on a PC through a web-browser. All together this encapsulates the deployment of the Glow2Go system and expresses the physical view of the 4+1 diagram.

6 Process view

The process view provides insight into how the system operates at runtime. This is done by highlighting the different processes which run at once. This view can help to discover redundancies in the individual processes.

6.1 Sequence diagram for system processes

To represent how the individual processes interact, we can create a sequence diagram, where each thread in the diagram represents a process responsible for a specific task in the Glow2Go system.

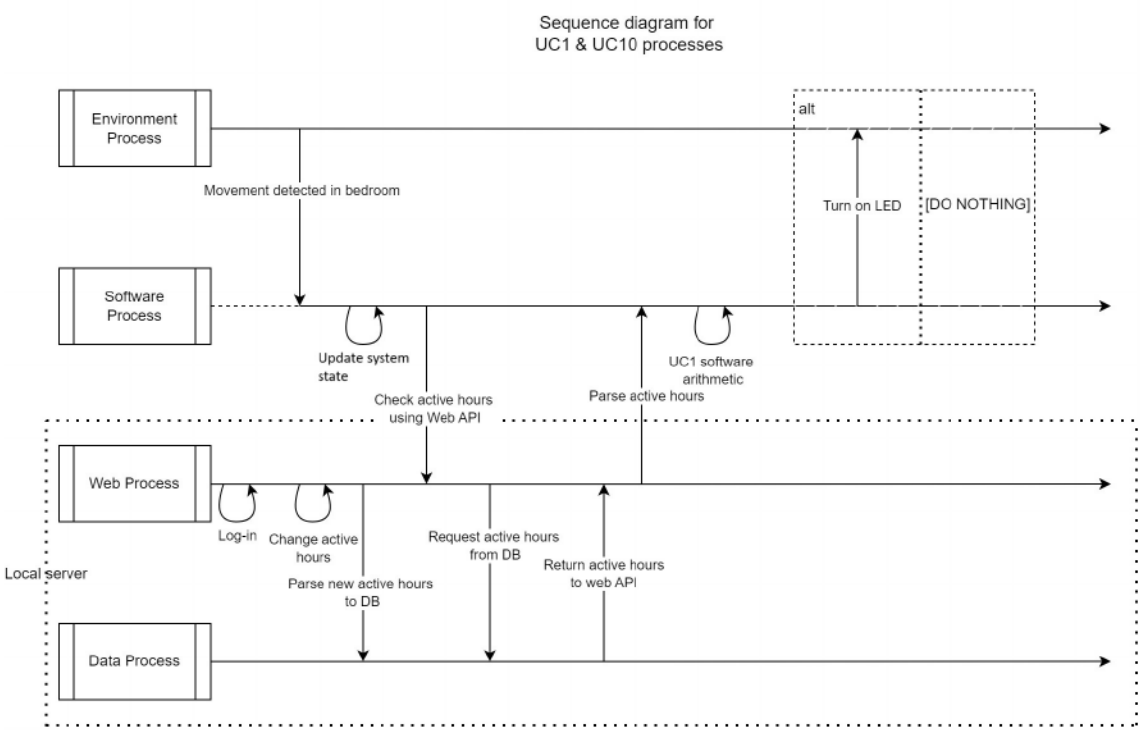

Figure 11: Sequence diagram highlighting the interactions between the concurrent processes of the Glow2Go system at runtime wile running .

The diagram in Figure 11 provides an abstraction of how the system flow of two specific use cases can run simultaneously and have impact on each other. It also shows how a use-case is handled covering the different process threads and how the decisions for the system flow are made on more than one thread.

The diagram specifies the main system processes as the 'Environment','Software',Web', and 'Data' processes.
The diagram then outlines the process-flow of the system while operating to process two use cases concurrently (UC1 & UC10, see requirements specifications for more information). This is a good example, since it shows how decisions made on one process thread impacts the decision made on another.
Note, that the diagram specifies that the Web and Data processes are running on a local server. These processes could also be deployed on external servers as well.

7 References

[1] Software Engineering by Ian Sommerville, *Literature on Software Engineering by Ian Sommerville*, Ian Sommerville, Software engineering ISBN: 9781292096148 10th edition, 2016.

[2] Mermaid, *Mermaid diagrams and mermaid live editor used for automatic diagram generation.*, `https://mermaid.js.org/`

[3] DrawIO, *Draw.io Software used to draw architecture diagrams.*, `https://app.diagrams.net/`

[4] Cep2App, *Cep2App template application used for the system.*, `https://github.com/jmiranda-au/cep2app`

[5] Tutorials, *All the tutorials used in the development of the system.*, The tutorials are available on brightspace under the content tab of the course 'Computerteknologiprojekt II (F24.285201U030.A)'

[6] Wikipedia, *4+1 architectural view model on Wikipedia*, `https://en.wikipedia.org/wiki/4%2B1_architectural_view_model`

8 Appendix

Appendix 1 - Requirements specification rev 1
path: Appendix/Requirements specification rev 1.pdf