Telehealth and Remote Monitoring of the Elderly

A Generic IoT Approach

Sergio Soto

ISBN 000-00-0000-000-0

TU Delft

Telehealth and Remote Monitoring of the Elderly

A Generic IoT Approach

by

Sergio Soto

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday October 24, 2018 at 11:00 AM.

Student number:	4627288	
Project duration:	February 1, 2018 – October 10, 2018	
Supervisors:	Dr. RangaRao Venkatesha Prasad,	TU Delft, supervisor
	Ir. Nikolaos Kouvelas,	TU Delft
	Warner ten Kate,	Philips Research

This thesis is confidential and cannot be made public until October 24, 2020.

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

PHILIPS Research

TU Delft

CONACYT
Consejo Nacional de Ciencia y Tecnología

Abstract

As the elderly world population increases, caregivers are switching to remote care and monitoring solutions to enable their patients to live autonomously at home for as long as possible. Such services are based on detecting and recognizing Activities of Daily Living (ADL) by using diverse types of sensors at the elder's home that transmit activity data to a remote backend for processing. However, these have a limited reach due to high costs, mostly due to elevated prices of the hardware required, as well as the installation costs. Meanwhile, Smart Home technologies are reaching more homes every year. These are based on identical sets of sensors as those prescribed for remote health monitoring systems. Extensive research has taken place in recognizing ADL given the sensor signals. However, these methodologies require prescribed sensors, the floor plan of the house and the location of the sensors.

In this thesis, we first propose a generic IoT interface that enables the backend to connect to arbitrary sensors, such as those typically found in Smart Homes. Regardless of the brand and communication protocol they use, the sensor data is transmitted to a backend where it can be processed by the ADL recognition algorithms and other telemonitoring services. We evaluate our design in the Philips HomeLab environment where generic sensors are installed. The system correctly transmits the activity detected by the sensors to a remote cloud service. Then, we propose a methodology based on Multidimensional Scaling to estimate the location and distribution of sensors based entirely on monitored sensor activity. Finally, we propose a methodology combining heuristics and Support Vector Machines to classify and label the sensors by the type of room in which they are placed, such as Kitchen, Bedroom or Bathroom. We train and test our model based on 4 different houses where volunteers had sensors monitoring their activity for at least 80 days. Our proposed methodology correctly identifies 5 different key locations and fully characterizes the interior of the house. Our heuristic classifier achieves a sensitivity of 0.84 and specificity of 0.98. The SVM classifier achieves a precision of 0.78 and recall of 0.78. This results in an automated method of generating the configuration data needed by the fore mentioned ADL recognition algorithms. The experiment demonstrates the feasibility of using the IoT paradigm for remote health monitoring systems by reusing existing sensors found in smart homes, reducing burden and costs which can potentially have an impact on a broader adoption of these kind of monitoring services.

To my grandpa, Luis†

Contents

1

Introduction

In the past century, the life expectancy and average age of the global population have increased drastically. According to [1], the 65-and-older population in the US will more than double by 2050, and the 80-and-older population will more than triple by then. Figure 1.1 illustrates the impact of population aging across the world. With this unprecedented growth in the elderly population, health providers need new alternatives to provide adequate care services for the elder, particularly for those that intend to live at home independently. Telemonitoring and remote care services are important elements in preventive health solutions that can have a relevant impact in improving wellness and quality of life of these older adults. With these services, the caregiver can provide attention to the client by constantly having access to information regarding their general well being, detect patterns and anomalies in their behavior and, when needed, take prompt action in risky situations and prevent further decay in health.

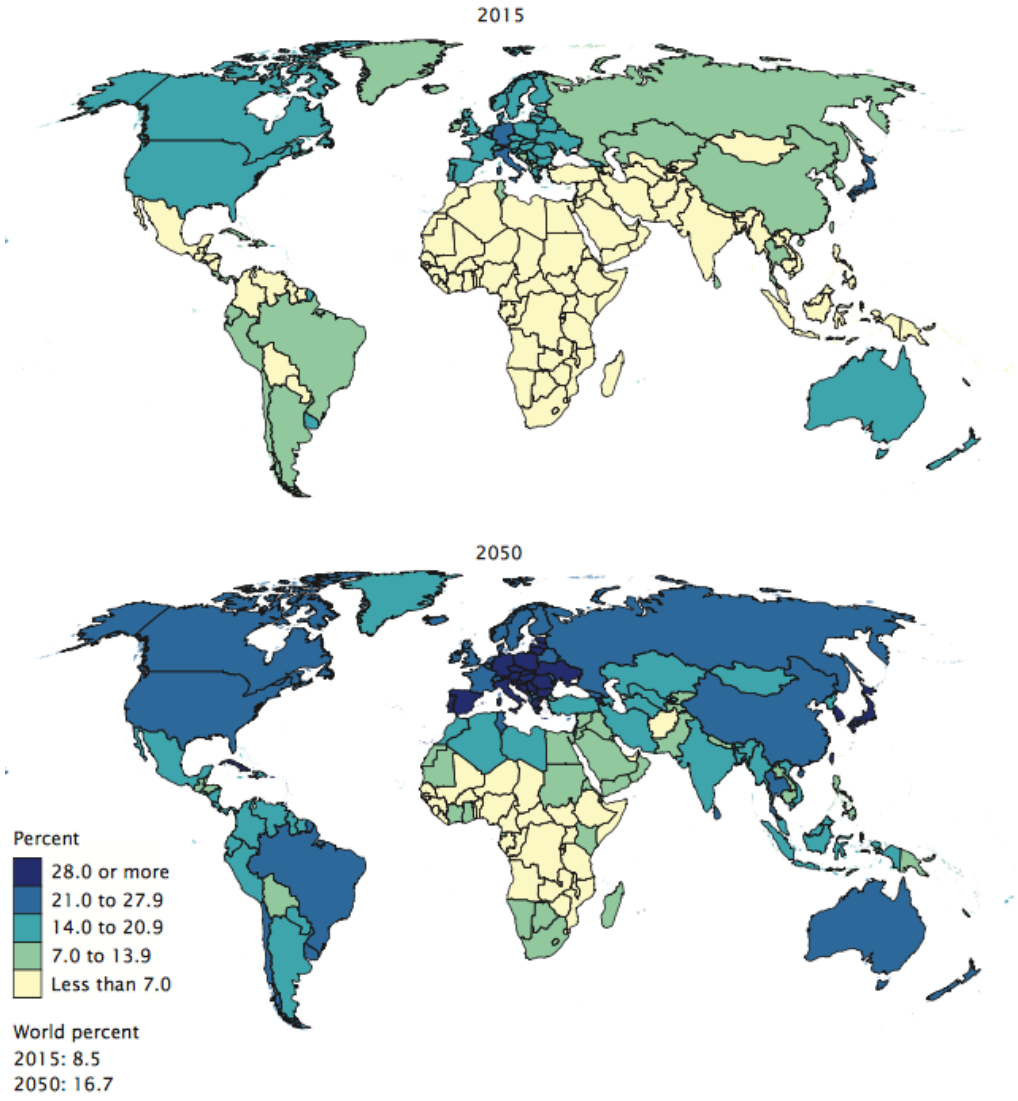


Figure 1.1: Percentage of Population Aged 65 and Over: 2015 and 2050.
Sources: [1], U.S. Census Bureau, 2013, 2014; International Data Base, U.S. population projections.

However, despite their importance, remote care services have a limited reach due to their elevated price. One of the main factors is the cost technological infrastructure needed in order to continuously monitor the client at home from a remote location. Diverse types of devices have to be installed in the client's home in order to track the activities of the inhabitant, like infrared motion sensors, door open/close sensors, pressure mats, cameras, and others. Gateways that transmit the sensor data to a remote server are also necessary. These devices generally require a technician to visit the house to install and configure them, at the expense of the client.

(a) Door sensor

(b) Motion sensor

Figure 1.2: Examples of sensors used for remote monitoring.

At the same time, Smart Home systems rely on similar sets of sensors to provide different kind of services, ranging from commodities like smart lighting systems to smart energy buildings and surveillance suites. Adoption of Smart Home systems is rising across the globe, particularly in developed countries. According to [2], in the UK, household penetration is 19.7% in 2018 and is expected to hit 39.0% by 2022. With the advent of these technologies reaching more homes every year, in the future more houses will be equipped with the necessary infrastructure to acquire the necessary data to assess the health condition of their inhabitants. Evaluating and demonstrating the feasibility of re-purposing these Smart Home devices which are readily available at households, for the purposes of remote care services, is the motivation of this research.

1.1. Problem Statement

The objective of this study is to to address the problems remote elderly care services face in order to become widely adopted, reach a larger portion of the elder population, and be ready for the increase in this segment of the population in the coming years. To achieve this, costs should be reduced while keeping or improving the effectiveness of the current offer. Two main problems are identified and discussed in this section: the use of prescribed sensors, and the requirement for floor plan configuration in telehealth technologies.

Problem 1: Prescribed Sensors

One of the biggest problems to solve in remote care systems is the need for prescribed sensors. These are the sensors that the care provider selects for each home where remote monitoring services will be provided. These sensors are installed and maintained by a technician, on-site. This scheme has several drawbacks:

- *Cost:* Besides the cost of the hardware, users must cover the costs of installation and configuration of the system. Not only does the technician install the sensors in the building, but he must also configure them.

- *Inconvenience:* These systems are not an *out-of-the-box* solution. Configuring the sensors and gateway requires technical proficiency (commissioning sensors to the network, connecting the gateway to the remote server, setting up the sensor and floor plan configuration).

- *Error proneness:* It is difficult to train technicians to install the sensors correctly. For example, a sensor meant to be placed in a hallway could erroneously be placed in the kitchen, or additional, redundant and unnecessary sensors could be placed with the thought that having spare ones could be beneficial.

- *Redundancy:* In the case of Smart Homes, most, if not all, of the necessary sensors are already installed and available throughout the house. Installing another set of sensors on top of these is not only cumbersome, but wasteful.

It follows that if it were possible to re-purpose sensors existing in the Smart Home, costs would be saved, the inconvenience would be minimized, and sensor redundancy would be eliminated. Using smartphones as an analogy: A user has a piece of hardware, the smartphone, that already has a purpose and functions (making phone calls and browsing the internet for example). Whenever he wants to have additional functionality (sharing photos for example), rather than buying a new device with this specific purpose, it is only a matter of downloading and installing the appropriate *App*. Remote care providers offer a scheme similar to this, using the hardware already available at the client's smart home.

However, utilizing Internet of Things (IoT) sensors used in Home Automation applications presents a new challenge. There is a vast array of different brands offering these sensors, operating with different communication protocols not directly interoperable with the remote backend server of the care provider company. Therefore, it is necessary to structure a network interface and architecture capable of interpreting and transmitting data from these generic and unknown sensors.

Problem 2: Floor plan discovery

There is a common ground among all Activities of Daily Living (ADL) inference algorithms and methodologies: they all need access to sensor data and events and, to some extent, to parameters and configuration of these. Concretely, these methodologies require the floor plan of the building is known, as well as the location of the sensors throughout the house. In other words, the algorithms know the distribution of the sensors *a priori*, as well as the different kind of rooms in which they are located (e.g. the kitchen or a bedroom).

Indoor floor plan discovery has been previously studied through different approaches. However, there seems to be a gap in the research. None of the works focus on both identifying the sensor distribution and classifying the rooms in which these are positioned, based solely and entirely on sensor event data. In other words, a complete characterization of the house layout that requires no input from the user, but rather uses the data available from binary state sensors only.

Automatic floor plan discovery is key in procuring remote care services using pre-installed IoT sensors in Smart Homes given that, opposed to the scenario where the care provider prescribes them and has a technician installing them, the ADL inference algorithms will not possess information about their location.

1.1.1. Research Question

How can generic IoT sensors available in Smart Homes be used in order to enable remote elderly care services?

This thesis will address this question by answering the following three sub-questions:

- $Q1$: *What kind of cost-efficient network interface and architecture will enable retrieving data from arbitrary IoT sensors?*
- $Q2$: *How can a sensor distribution map be extracted given only the sensor event data?*
- $Q3$: *How can we classify room types in which generic sensors are located given event data only?*

Questions $Q2$ and $Q3$ are focused specifically on binary state IoT sensors.

1.1.2. Objective, Requirements, and Contribution

There are two main objectives in this research. First, to devise a generic network interface and architecture that can transmit data and events from generic IoT sensors to a remote backend for storage and for activity tracking algorithms to process, while minimizing the burden of installation and configuration at the user's end. Second, to estimate the position of binary motion sensors available inside a house and classify the type of room in which they are.

Our method should require no input or configuration from the user whatsoever. Based on observations of events and activations, it will estimate not only the relative distance between the sensors but also their approximate layout and distribution. This must be done during the normal operation of the sensors and without interfering with their original purpose in the Smart Home environment. Then, based on certain heuristic criteria and supervised learning, it shall estimate the type of room in which each sensor is. In other words, it will generate a graph-like map of the house with corresponding distances and relative distribution, and will label each sensor with a room type (e.g. "bedroom" or "kitchen").

The main contributions can be summarized as follows:

- We propose a generic network interface and architecture that can take as input data and events from IoT sensors for remote care services and transmit it to a remote backend.
- We use pre-existing sensors installed at the home of the user without affecting their original intended functionality and purpose.
- We test our network interface in a custom testbed environment at Philips HomeLab.
- To the best of our knowledge, this is the first work in performing an indoor floor plan discovery based solely on sensor events while requiring no configuration or input from the user.
- We report on results obtained using non-linear dimensionality reduction techniques in order to estimate the position and distribution of sensors around the house.
- We present a combination of heuristics and supervised learning in order to classify the types of rooms in which sensors are placed.
- We report on results from experimental verification of our methodology with data acquired from real homes inhabited by elderly, our intended demographic.

1.1.3. Constraints and Assumptions

The following constraints and assumptions are taken into consideration for this work:

1. The environment assumed is a Smart Home equipped with IoT sensors.
2. We focus on identifying the following *key locations*:
    - Bedroom
    - Bathroom
    - Toilet
    - Kitchen
    - Living Room
    - Hallway
3. All *key locations* have exactly one sensor installed in them.
4. There is at most one sensor in each *key location*, except for hallways.
5. Although other types of sensors operate in a similar manner and produce similar event data, our tests on floor plan discovery will be carried out with binary motion sensors only.
6. It is assumed the house is inhabited by a single person, with no pets present.

1.2. Thesis Outline

The structure of this document is as follows: Chapter 2 presents a literature review of previous works related to this investigation, covering papers in methodologies for and applications of remote care services and implementations of floor plan discovery and room classification. Then, Chapter 3 introduces a general background on Activities of Daily Living, as well as the general structure of methodologies to recognize and classify these. IoT and its role in Smart Home technologies is also discussed in this chapter. Chapter 4 introduces our research methodology. It first describes our proposed architecture and network topology for arbitrary IoT sensor data acquisition. Then, it explains our methodologies for floor plan discovery using multi-dimensional scaling and our proposed room classification algorithm. Later, Chapter 5 describes the test methodologies used for our proposed solution and discusses the results obtained. Finally, Chapter 6 concludes with a brief discussion of the results obtained and the relevance of this study, and proposes research paths to follow up this work.

1.3. Disclaimer

The present work was sponsored and carried out as part of an internship assignment in Philips Research, in Eindhoven, The Netherlands. Part of the data used in this research was provided by Koninklijke Philips N.V., carried out by volunteers in the *Living Labs* in the Eindhoven region as will be described in Section 5.2.1, and other data was generated by the author at the Philips *Experience Lab* as will be described in Section 5.1.1, within the premises of Philips Research in the High Tech Campus in Eindhoven, The Netherlands, under supervision of Warner ten Kate. The datasets, algorithms developed, as well as their implementation in software, are protected property of Philips Research. The data collections have been conducted after approval by the internal ethical committee. The volunteers participated with written consent and could withdraw participation as well as data usage at any moment.

Part of this work was published in the Philips-internal Technical Note PR-TN-2018/00114, *Using IoT Sensors for Monitoring in Telecare*[3].

2

Literature Review on ADL Monitoring Methodologies

Using sensors and IoT devices to detect and recognize human activity has been a widely studied topic in the past decade. A subset of these activities are ADL, and there are countless entries in the literature for research on innovative methodologies for classifying them for remote health monitoring purposes. As the elderly population increases, the dependency on assisted living follows the same tendency [4]. This growth in supervised care is becoming economically unsustainable, and research on methods to enhance independent living at home attempts to respond to this situation.

Our goal is not to propose yet another methodology for ADL recognition, but rather to introduce an innovative tool that enables these techniques to operate in unknown and uncharted smart buildings by re-utilizing sensors already available in these. The biggest challenges that will be faced will be generating an accurate representation of the indoor room distribution of the household and identifying the different spaces within it.

Previous works have been published presenting methods to generate indoor floor plans of buildings via crowdsourcing techniques such as Simultaneous Localization and Mapping (SLAM) [5] [6] or Pedestrian Dead Reckoning (PDR) [7] [8]. However, they involve having the user(s) walking in paths around the building carrying a device or smartphone to generate a drawing of the estimated floor plan. Our objective is to introduce an unobtrusive method where the user does not carry any devices with him, and that does not require a setup or training phase during which the user has to do anything outside of their regular activities.

No literature was found on floor plan recognition using data from *binary sensors*[1] only. Instead, this literature review will be directed towards previous works that focus on identifying ADL. The reason for this is the common set of sensors and environments used in both applications. Although the final objectives are different, there is common ground between both kinds of studies:

- ADL recognition works rely on diverse sensor data in order to characterize an action. For the most part, binary state sensors are used. The intended floor plan recognition proposition will generate a floor plan by using the same kinds of sensors.
- The sensors in both methodologies are installed in smart homes with similar structures (e.g. they have a kitchen, a bathroom, a living room, etc.)
- ADL recognition algorithms extract features from human patterns in order to classify activities. Our proposition follows a similar model, but in order to classify the location from the sensors detecting such patterns.

Diverse methodologies and technologies are employed in order to infer ADL. Camera-based systems use computer vision in order to estimate the client's activity in real time by positioning cameras at the rooms of interest throughout the house. Others use wearable devices for indoor localization, microphones, and even humidity and other types of sensors to estimate room occupancy in order to achieve the same goal. Finally, some solutions present unobtrusive systems based on motion, open/close and other passive sensors throughout the house. A selection of these works will be summarized in this section, and later will be discussed to establish a theoretical framework for this research.

2.1. Related Works

In [9], the use of embedded and worn sensors to detect ADL is studied. A publicly available dataset from the MIT, 'Intense Activity', is used. It consists of activities of a person in a house recorded over a period of 4 hours with over 300 different sensors positioned in the MIT 'PlaceLab'. These consist of switches, audio, video, heart rate, acceleration, humidity, pressure, illumination, temperature, gas, and electrical current sensors. A variant of AdaBoost is learnt, proposed in [10], to classify actions. Out of the entire spectrum of ADL, this work focuses only on classifying activities associated with eating (e.g. cooking, preparing a meal, washing dishes, drinking, etc.). The primary objective is to figure out what is the minimum amount of sensors required for this task. The results show that using close to 100 of these sensors, an accuracy of 90% is achieved, and when reducing this to only 8 sensors, an 82% accuracy. These 8 sensors consist of current consumption on appliances like the dishwasher, humidity sensors, a video camera, and a variety of accelerometers worn around the wrists. Although the achieved results are acceptable, most of the sensors that were found most effective are worn by the user as embedded devices. This goes against our objective of an unobtrusive design where sensors are only placed throughout the house. The sensors are connected to a cluster, grouped by sub-clusters that

[1] Binary sensors are those which transmit one of only two possible values or events. For example, a motion sensor would indicate whether motion is detected in its field of view, or not.

acquire the sampled data for each sensor. These, are located in the PlaceLab, an artificial environment built entirely for the purpose of studying human activity on a variety of sensors. In a Smart Home environment, the challenge will be to categorize these activities using only a subset of the kind of sensors available at this lab.

In [11], an evaluation of data fusion from inertial/magnetic kinematic sensors attached to the user is introduced. In this work, sound, speech recognition, and passive infrared (PIR) motion sensors are placed in the Health Smart Home of Grenoble, a full-size flat inside the faculty of medicine of Grenoble. These are used to classify different user activities. The achieved result was 72% of activities correctly classified using the accelerometer and motion sensor data, and an average of 62% accuracy in recognizing activities based on speech and sound.

Later, in [12], Support Vector Machines (SVM) are ised to classify 7 different ADLs using a similar set of sensors from [11]. The motivation to use SVM was the weak number of available sensor samples, which make the learning process difficult for a larger number of them (in contrast to Naive Bayesian or Neural Networks, for example). The results from using Polynomial and Gaussian (RBF) kernels are compared and finally select features from the different kinds of sensors used are chosen. For the PIR sensors, the percentage of time spent in each room and the number of events detected are chosen as features. A classification accuracy of 86% is achieved, cumulative across the 7 ADL, with a Gaussian kernel.

In [13], a similar approach is followed to that of [12]. Other devices like temperature, light, humidity, open/close, and PIR motion sensors are used. These are installed in an elderly person's house in New Zealand. Although satisfactory results are achieved, the conclusion highlights that it is not straightforward for such a system to integrate sensors from different manufacturers since proprietary sensors and hub were used, which operate with proprietary communication protocols and therefore it would be difficult to extend such implementation.

A combined SVM and Hidden Markov Model (HMM) approach is proposed in [14] to recognize over 12 different activities, achieving an average of 89% precision overall. However, this method uses unconventional and intrusive sensors, such as the Microsoft Kinect, which don't cover our requirement of being unobtrusive.

The study in [15] focuses on the topic of detecting visitors in their residence based on motion sensor data. SVM is also used to classify the ADL. It elaborates on the difficulty of classifying with multiple residents because PIR cannot differentiate who is moving. The study points out how video cameras violate the resident's privacy, and how body-worn tags like RFID (Radio Frequency Identification), or UWB (Ultra-wideband) are impractical and become uncomfortable for the user. The test take place in two houses with different layouts, achieving a 90% accuracy in one, and 67% on the other. As classification features, dwelling time, the number of sensor firings and the number of transitions between rooms are used. It concludes that the different home layout and sensor placement in both houses had a considerable effect in the results.

A hybrid approach for recognizing ADL is proposed in [16]. It combines intensive use of body-worn sensors (a smartphone) and ambient sensors, mainly PIR motion sensors. The novelty of this work is focusing in multi-inhabitant environments. By using the combined sensors, it achieves an improvement of 30% classifying activities compared to pure smartphone-based solutions by exploiting the spatiotemporal behavior of users and having access to their location information.

The high sensor cost used in previous approaches to recognize ADL, and how some of these are intrusive and violate the client's privacy is discussed in [17]. It points out how wearable sensors can only provide recognition on simple activities like walking or running but are not ideal for detecting complex tasks by themselves. The proposal is recognizing activities by using ultrasonic positional sensors in the form of pocket-sized transmitters worn by the user and receivers positioned across the ceiling that divide the house in "position cells", and power meters connected to appliances around the house. The objective is classifying six different ADL, like toileting, cooking, and dressing. In order to extract features from the oversampled data, data is divided into epochs. An SVM learning model is used. The experiment is carried out in the Nara Institute of Science and Technology (NAIST) smart home testbed. It achieves an 85% precision and 82% recall. As future work, it suggests using activity time and deviation in sensor data as features as well.

In [18], the author realizes that many models for ADL recognition are based on labeled data. However, due to the high variability of different contexts, learned parameters in one context are difficult to use in another. That is, differences in house layouts and the way individuals perform activities. The proposed semi-supervised approach maps the sensors from one house into another in order to apply the learned data from one dataset into another, using Transfer Learning. Finally, Hidden Markov Models are used to classify the data. The results show an improvement of the classification results over those of traditional supervised approach. However, the classifier fails in correctly distinguishing toileting and showering since both activities

are carried out in the same room. Such is the case for differentiating eating from cooking.

The work from [19] compares three state of the art classifiers for ADL: Random Forest, SVM, and Naive Bayesian. Data is obtained from motion, temperature, luminescence, humidity and acceleration sensors located in a custom-made smart building installation for his study. In order to process the data, sensor activity is clustered into windows to remove outliers since the measured data was sampled at 0.2Hz and ADL usually take minutes, up to hours. Similar data points are grouped into tokens, representing time spans. Out of each span, the mean and variance of the sensor readings are computed. Short disturbances in most rooms are ignored, given the finding that most ADL are in the order of minutes except for toileting. The study concludes that most of the analysis can be carried out almost entirely on PIR motion sensors. However, it is pointed out that showering and toileting activities are difficult to distinguish without a humidity sensor.

A methodology to localize binary motion sensors in a house is presented in [20]. The approach requires prior knowledge on the floor plan and is provided to their application in the form of an image or drawing. The image is processed and a grid of squares representing each room of the house floor is generated, along with a graph representing the connectivity between them. Then the user labels the rooms in the grid using a smartphone app. The algorithm then uses heuristic methods based on a crowdsourced survey and Bayes theorem to detect the "key locations": Bedroom, Entrance, and Kitchen. Then, the connectivity and relative distance between sensors are calculated based on event timestamps and a graph connecting them is generated. Finally, a mapping between the floor plan graph and the sensor graphs is performed to estimate the location of the sensors in the rooms.

2.2. Discussion

2.2.1. Sensor Technology

Sensor selection is an important element when designing an ADL recognition methodology. These sensors will determine the system architecture, classification methodology, and complexity of the solution. Also, these will affect the user experience and will ultimately be a determining factor in the success of the design. We will discuss briefly the available options, based on the literature previously studied.

Wearable *versus* Ambient Sensors

One of the requirements of our work is avoiding sensors that put a burden on the user such that he has to adapt his routine and behavior in order to benefit from remote care services. This means that wearable or embedded devices such as the accelerometers used in [9], [12], and [19], or the smartphones used in [16]. Having motion data from the client is a powerful tool that allows for finer-grained classification of different human actions. However, our purpose is using ambient sensors installed in a smart home that provide a less invasive form of monitoring.

Binary State Sensors *versus* Others

[9], [12], and [17] demonstrated the effectiveness of sensors with non-binary outputs, such as humidity, CO^2, current and light sensors, in human activity recognition. Having more than two possible values offers a broader spectrum of combinations and patterns that characterize the location and intended activity of a person inside a house. At the same time, however, the design in [9] uses a non-binary sensor in a binary way (i.e. A current sensor in an appliance, indicating only whether this is in use or not) with satisfactory results in classification.

On the other side, except for [17], every other experiment used data from PIR motion sensors. This is in great part due to their versatility. These low-cost sensors usually connect wirelessly and can run on batteries for months. This makes installing them anywhere practical and convenient. Additionally, they can reliably detect the presence of a person in a room, provided they are positioned correctly. This is in contrast to, for example, a water flow sensor that would miss the event of a person visiting the bathroom if he or she would forget to wash their hands. These sensors, however, are not perfect and their output will have to be pre-processed in order to be useful to our algorithms.

Accuracy *versus* Privacy and Convenience

Very accurate results can be achieved when using video camera systems or, as used in [14], a Kinect sensor. However, these sensors present the problem of making the user uncomfortable and having his privacy violated.

Other sensors used, like the water flow sensors used by [13], proved to be significantly challenging to be deployed without resulting in property damage or conflicting with the desire of keeping the sensors as discreet as possible.

2.2.2. Features and Classifier Selection

The authors of the studied literature selected different features to train their supervised learning models. We describe those of particular relevance for our purposes:

- *Epochs:* [15] and [17] realized that ADL duration is between minutes and hours. Therefore, the sensor data was downsampled and separated it into epochs. That is, they divide the 24-hour duration of a day into fixed-length observation windows.
- *Tokens:* The approach in [19] is compressing data into tokens, which represent activity time spans in a room. In other words, sequences of identical sensor readings for specific sensors (the PIR sensors only) are compressed into a single token which has an event duration (time span), and other characteristics derived from the mean and variance of other sensors which then are used as features.
- *Dwell Time:* [17] proposes using the time spent in one room as a feature. This is similar to the model in [19], except that the data does not need to be compressed because samples are acquired on state changes only in the first, opposed to continuous samples in the latter.
- *Sensor Firings:* In [17], the author realizes that the frequency at which different rooms are visited, and therefore the sensors present are fired, proves to be an effective feature for classification.
- *Transitions:* [17] uses transitions between rooms as a feature. This is a very useful tool for classifying activities, but it is unfeasible for the purpose of our research. As an example, motion activity detected in the bedroom, transitioning to the bathroom and finally to the kitchen could be understood as a person waking up, toileting and going to the kitchen to prepare breakfast. However, in our scenario we part from the assumption that all sensors are in unknown locations and therefore we do not possess enough information to use this as a feature.
- *Timestamp:* [17] proposes to use the timestamp of the sensor events as features.
- *Dwell Time Deviation:* Also proposed as future work, [17] mentions using deviations in the time a user spends in a room as an additional feature to improve results.

2.2.3. Gap in Test Environment and Configuration Data

One of the biggest gaps we identify in the literature related to ADL recognition is regarding the use of artificial and custom-made facilities in which these experiments take place. As examples, [9] uses a dataset from the MIT PlaceLab, a facility with over 300 sensors installed. The study in [12] takes place in the Grenoble Health Smart Home inside the faculty of medicine in Grenoble. The experiment in [17] is carried out in a smart home testbed built in the Nara Institute of Science and Technology. Others, like [13], perform the experiment in real houses where they carefully placed a prescribed amount of sensors at key locations. It was not possible to find a related paper that would consider the scenario where the sensors are in unknown locations.

Another gap we identified is regarding the sensor configuration. The methods described above, as well as others like [21], [22], [23], and [24], rely on specific information related to the home and sensor configuration. Specifically, the most essential pieces of information they require, besides the sensor readings, are the floor plan of the house and the location of each sensor. This follows along the gap described in the previous paragraph.

If remote elderly care services are to be adopted in the future, in the context of using pre-installed sensors at Smart Homes, these should be capable of working "out of the box" by requiring minimal user input.

2.2.4. Additional topics

Other topics of interest were studied by these authors, like the detection of visitors in [15], the innovative adaption to multi-inhabitant environments in [16], and the Transfer Learning techniques proposed in [18] to map a set of sensors from one house to another. These would certainly add value to this research but in order to keep our approach simple and concise, these are kept as features for future work.

3

Background and Context

Background and context information will be introduced before diving into our research design and methodology. First, Section 3.1 will explain the definition and purpose of ADL. Then, Section 3.2 will briefly explain the basic structure of remote activity monitoring systems, Section 3.2.1 will describe the most common types of sensors used for these, and Section 3.2.2 will briefly elaborate on sensor data pre-processing. Sections 3.2.3 and 3.2.4 briefly present the front end of ADL monitoring systems that care providers interact with. Finally, Section 3.3 explains the applications of IoT, as well as the different communication protocols and gateways commonly found in smart home applications.

3.1. Activities of Daily Living

Sidney Katz [25] introduced ADL (Activities of Daily Living) in 1976 as a specific set of activities that people carry out in their daily lives that measure their primary sociobiological functions. They are compiled in what today is called Katz ADL scale. Geriatrics use the Katz ADL Scale [26] as a metric to evaluate the autonomy and functional status of a patient. Examples of these ADL are:

- Self-feeding
- Bathing and showering
- Dressing
- Personal hygiene and grooming
- Toileting
- Ambulating (Functional Mobility)

3.2. Remote Activity Monitoring

Human activity monitoring is a widely studied topic and a key element for preventive at-home health assessments [27] and to detect certain health conditions, like Alzheimer's disease [28] or dementia [29], at early stages. In general, such studies attempt to recognize and utilize ADL, as introduced by Katz, to detect behavioral patterns in patients [30] and for remote monitoring and assisted living solutions.

The general structure of a remote activity monitoring system is shown in Figure 3.1, and will be shortly described.

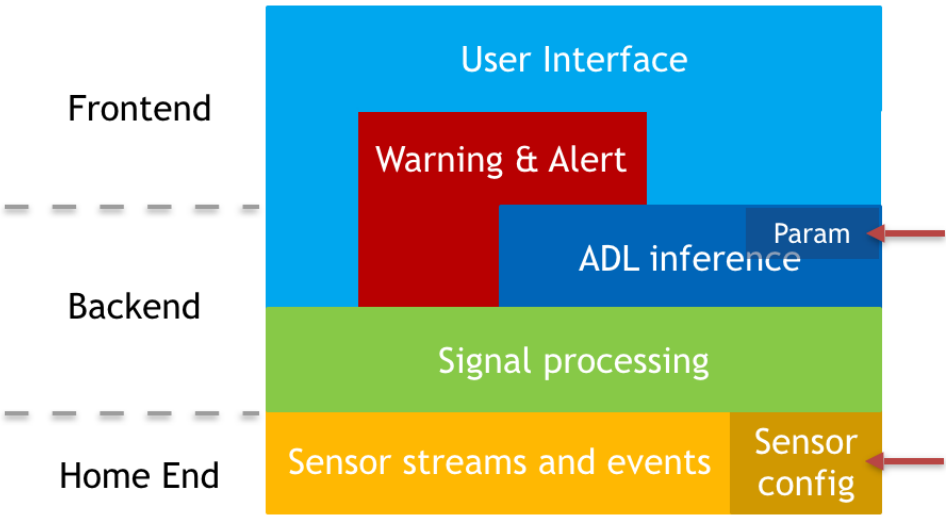


Figure 3.1: Overview of a remote activity monitoring service architecture

3.2.1. Sensors and Gateway

Monitoring systems commonly consist of a set of wireless sensors positioned throughout the house and a gateway to communicate events captured from the user's activities to the remote backend. This, in contrast to

other methods using wearable devices, combinations of body-worn and wall-mounted sensors, smartphone or camera-based sensors, provides with real time monitoring capabilities while being unobtrusive. Typical types of sensors used are:

- *PIR motion sensors:* Passive infrared motion detectors measure infrared radiation emitted by objects within their field of view. They typically have a conical field of view and, when properly positioned, can cover the area of a small to medium room ($30m^2$). These sensors send an activation event when motion is detected, and a deactivation event when the activity has ceased.
- *Open/close (magnetic reed) sensors:* Commonly placed at doors, windows, refrigerators and cupboards, these consist of two main parts. On one side, a reed switch is positioned at the edge of the door or window. On the opposite edge, a magnet that aligns to the switch is attached. When the door or window is closed, both parts sit next to each other, causing the switch to be closed. When it is opened, electric current cannot flow through the switch anymore. An event can be therefore raised on either activity.
- *Pressure mats:* They are commonly used to detect whenever a person walks on a specific surface. The simplest of them consist of mechanism with a normally open contact which closes whenever a person steps on them. If positioned at the front door of a house, for example, these would trigger an event every time the user enters or leaves his or her residence.

Sensor Configuration

Configuration is required in order for the sensors to communicate to the gateway and for the events to be transmitted to the backend where they are processed. This is commonly referred to as "pairing" the sensors to the gateway. The procedure for commissioning these sensors to the network is generally dictated by the brand and model of gateway, as well as the wireless communication protocol used by these. In a typical tele-monitoring service scenario, a technician at the residence installs and commissions pre-configured sensors according to specific guidelines prescribed by the caregiver. An example follows:

> *Client X requests remote monitoring services at his residence. The caregiver provider company requests information about the house, such as its floor plan and the number and type of rooms it has. The company then prescribes the number and kind of sensors that will conform the network and prepares a package of pre-configured ("paired") sensors and gateway. It documents this configuration, as well as the specific parameters of the setup, and prepares an installing procedure manual for the technician. The technician is sent to the residence where he places the sensors and the gateway according to the instructions. An example from such instructions would be, "Install sensor ID_XX at the kitchen, looking away from the living room".*

3.2.2. Signal Processing

Once received on the backend, data from the sensor events must be processed. Given the nature of the sensors used, raw data is often incomplete. For example, a person may be detected entering the room, but missed when leaving it. Likewise, a person may sit on a couch in the living room and remain immobile while watching a TV show and therefore not considered as present. The methods we use for filtering events and pre-processing of the data will be further discussed in Section 4.2.3.

3.2.3. ADL inference

Different methodologies have been studied in order to infer ADL from data acquired from a variety of sensors, including the ones mentioned above. By means of diverse algorithms, processing and mathematical models, these methodologies translate sensor data into activities such as cooking, bathing or sleeping, that are ultimately presented to the care provider. A few of these were discussed in Section2.1.

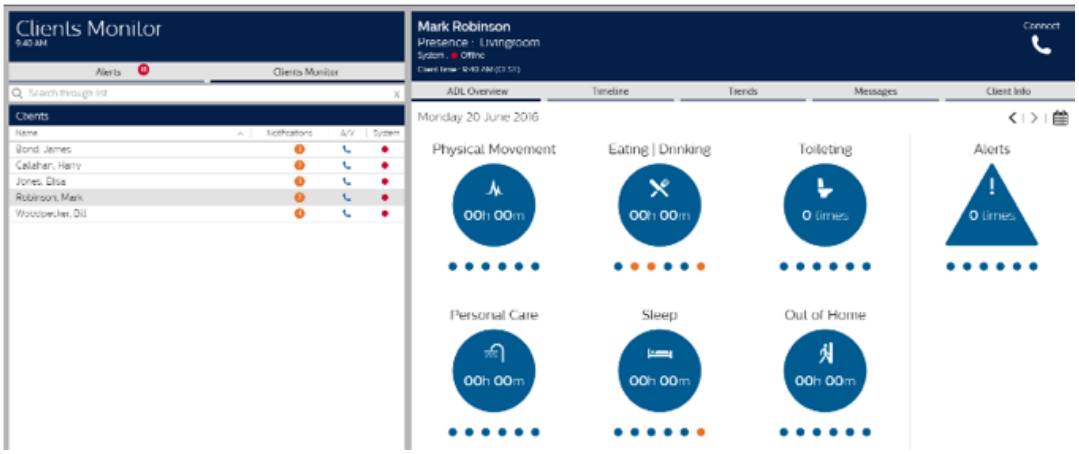
Sensor Parameters

In order to estimate the ADL of the client based on sensor event data acquired by the sensors, ADL inference algorithms require knowledge of certain parameters of the system installed at the house. These describe the layout of the house and its rooms, and the type of rooms in which each sensor is placed.

3.2.4. User Interface, Warnings and Alerts

The ultimate goal is to procure relevant information to the care provider (and sometimes family members) regarding the wellness and health state of the client. Doctors, nurses and other care givers get access to an interface, as shown in Figures 3.2a and 3.2b, which allows them to monitor the patient's state in near real

time. Similarly, whenever a risky situation presents itself (for example, the client falling, or leaving home at unexpected hours), the care giver can receive an alert in order to take prompt action and ensure the well being of the patient.



(a) View of ADL monitored in Dashboard

(b) Detailed view of an ADL

Figure 3.2: ADL Dashboard examples. Used with consent of Philips Research.

3.3. Internet of Things Devices

One of the main objectives of this work is to utilize readily available IoT sensors in the client's home. Therefore, it is important to understand the typical case scenarios where this kind of technology can be found in order to assess the target market.

3.3.1. Applications of IoT

Below we summarize the main forms in which IoT technology exists today in the context of home automation.

Smart Buildings and Home Automation
The most common use for IoT is undeniably the home automation market. Users look for convenient ways to make everyday life easier. From turning on lights in their living room according to different mood settings, to opening garage doors automatically as they approach their home around the corner, the uses of IoT are endless.

In this scenario, several smart devices can be of relevance to this project: PIR Motion sensors, open/close sensors, window shades actuator, smart switches/lighting, pressure mats, curtain controls, garage doors, and smart appliances such as smart TVs, smart washers, smart laundry machines, smart toothbrushes, etc.

Smart home safety
Modern homes are adopting smart security devices more often. Users want the feeling of resting assured their homes are safe while they are away for work or traveling. These devices can be of not only surveillance and anti-theft nature, but also to monitor risks like fires or floods.

In this scenario, several kinds of devices could be of use for the ADL algorithms. Some examples are: Motion sensors, open/close sensors, cameras, smart Locks, presence fobs, and smart valves.

Smart energy
IoT is used to reduce and control energy consumption in smart homes. Consumers are adopting IoT in their homes and industries to reduce energy costs and their environmental footprint. While most of these sensors and actuators in this category are meant to fulfill their purpose while the user is away, some of them are still relevant to this study. Some examples include: Smart plugs, thermostats, occupancy detectors, AC unit control, window control, and smart lights.

3.3.2. IoT Communication Protocols
There are different communication protocols and standards used in IoT, operating on different (or several) layers of the OSI (Open Systems Interconnection) model [31]. For brevity, our focus will be around those acting on the transport layer.

Websites like [32] indicate the leading protocols and standards as the following: Bluetooth, Zigbee, Z-Wave, WiFi, 6LowPAN, Thread, LoraWAN, and NFC (Near-Field Communication). This project will focus on the first four, given they are the most commonly used in consumer applications for Home Automation.

3.3.3. IoT Gateways and Hubs
In order to communicate all these smart sensors and actuators to the Internet, the typical application requires them to connect through a gateway, commonly called a hub or smart hub. Different brands have their own hub. Some communicate to the devices with proprietary protocols, or with limited compatibility (like the Philips Hue Hub using Zigbee Light Link [33]), and some offer multiprotocol communication. Here lies one of the bigger challenges of this project. Since it is impossible to anticipate which brand and make of these devices the user will have in their home, a generic solution must be devised in order to make the system compatible with as most environments as possible.

The following hubs were found to be the most widely adopted by the Home Automation consumers, according to consumer ranking websites like [34]: Samsung SmartThings [35], Wink 2 [36], Vera [37], Logitech Harmony [38], and Home Assistant. [39]

The Samsung SmartThings hub was chosen for this project due to the versatility it offers, as well as its multiprotocol technology.

It is also important to differentiate Home Assistant from the others. This, along other similar options, is an open source software developed by the community that can run on different environments and operating systems. It is meant to be used on a regular computer to control and command smart IoT devices. It requires additional hardware (such as a usb Z-Wave stick) in order to do so. This alternative is also considered for this project, with Home Assistant running on a Raspberry Pi, along with an AEOTEC Z-Wave stick.

4

Research Methodology

As described in Chapter 1, the problem of granting remote care services the capability to autonomously operate on arbitrary smart home environments can be divided into three main parts:

1. Establishing a network interface and architecture that will retrieve data from arbitrary IoT sensors into the backend.

2. Designing an algorithm to automatically generate a floor plan with the distribution of the sensors throughout the house.

3. Developing a classification algorithm to label the rooms in the house by their type.

These three problems will be discussed in this chapter.

4.1. Architecture and Network Topology for Arbitrary IoT Sensor Data Acquisition

Our implementation makes use of two products[1]: Philips HealthSuite Digital Platform (HSDP) [40] and the Samsung SmartThings Hub [35]. HSDP is a suite of core services and cloud-based infrastructure designed by Philips and built on top of Amazon Web Services (AWS) to develop connected healthcare applications. It is out of the scope of this study to describe its operation. However, it will be mentioned that for our purposes we will use its secure authentication module as well as the Telemmetry Data Repository (TDR)[41], a database-like environment, provided by the API.

In order to establish a brand-independent, protocol-agnostic network interface that will allow the intended remote monitoring services to retrieve data from the sensors into the backend, five different approaches and methodologies have been considered and explored. These approaches explore the possibility of acquiring sensor data at different levels of a smart home network. These are shown in Figure 4.1 and described briefly:

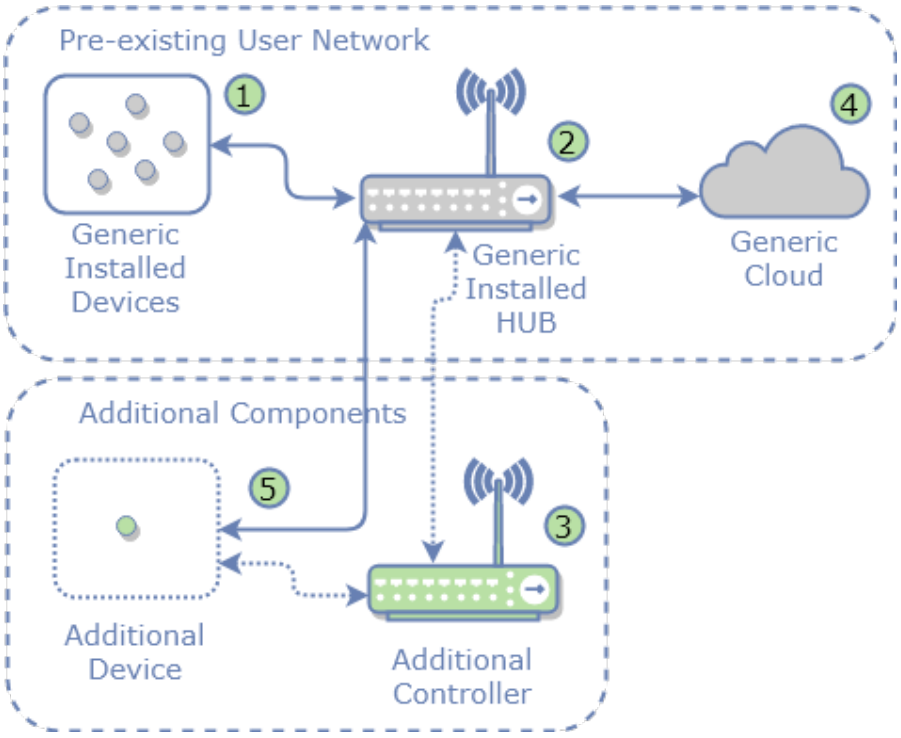


Figure 4.1: Overview of all five data acquisition approaches.

[1] Although our implementation uses these specific products and cloud services, these are not essential to the generic solution and can be extrapolated to equivalent platforms.

1. *(1)Direct sensor*: Acquire the data directly from the sensors through a *sniffer* listening to events in the network.

2. *(2)Gateway bridge*: Re-transmit the sensor data from the gateway/hub.

3. *(3)Gateway override*: Introduce a second controller into the network, overriding the role of primary controller from the original gateway/hub.

4. *(4)Cloud service*: Leverage on the use of cloud services in smart homes to publish sensor data through them.

5. *(5) Custom Hardware*: Design a Zigbee/Z-Wave device that joins the smart home network and is configured to transmit all data advertised in the network.

The detailed description and a discussion on these methodologies can be found in Appendix A. The results are summarized in Table 4.1. From these methodologies, a combination of Gateway Bridge, described in section A.3, and Cloud Service, described in section A.5, were evaluated as the most promising and were chosen as the solution to be implemented.

Approach	Interference w/existing setup	Compatibility	Complexity	Cost	Additional HW
Direct Sensor	✓	✗	✗	✗	✗
Gateway Bridge	✓	✗	✓	✓	✓
Gateway Override	✗	✗	✗	✗	✗
Cloud Service	✓	✓	✓	-	✗
Custom Hardware	-	✓	✗	✗	✓

Table 4.1: Feature comparison of architectural approaches

4.1.1. Cloud Service and Gateway Bridge Data Acquisition

Figure 4.2 depicts the interfaces between the sensors, the remote database, and the remote backend. There are two fundamental layers in the data path:

- *The Access Bridge*: We design a device that provides the gateway/hub with means to securely communicate sensor data to the cloud storage.

- *Authentication*: Provides uniquely-identified Access Bridges (i.e., unique devices installed in different houses) and the backend with access to the cloud storage.

The Authentication layer is out of the scope of this study since it is dependent on the care provider's infrastructure and therefore implementation-specific. For our work we made use of HSDP Authenticate, a module provided by the HSDP API that uses *OAuth* in order to grant temporary access tokens to devices like our Access Bridge.

The Access Bridge and its operation are described below.

Access Bridge

The Access Bridge is a device placed and configured at the residence of the client. It is the only additional piece of hardware required on-site to communicate the pre-existing sensors to the cloud in order to enable activity monitoring services at the premises. It is a bridge between the physical devices and the remote storage/backend (HSDP in our test setup). Most importantly, the Access Bridge ensures that any commercial home automation gateway/hub can communicate with the backend regardless of brand, technology and configuration as long as it supports sending HTTP requests.

The device itself consists of a small, low cost ($35 USD), low power single-board computer. For the purposes of this project, a Raspberry Pi running Raspbian Stretch OS, Linux kernel version 4.14 was selected [42].

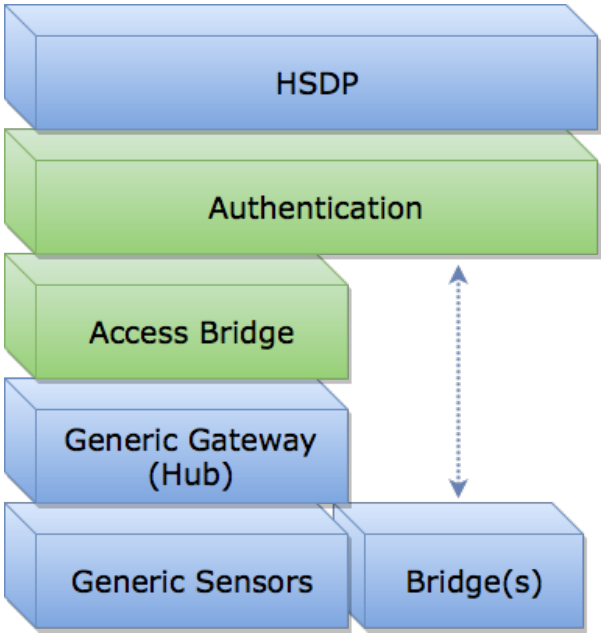


Figure 4.2: Sensor datapath

Static IP *versus* websockets

The Access Bridge must be accessible from the Internet, or third party clouds in case of Cloud Service data acquisition. In the case of using it as a Gateway Bridge, the Raspberry Pi provides Internet connectivity to the gateway itself, and therefore its local address can be translated as the gateway's default gateway.

Having a static IP address on the Access Bridge is the simplest solution from a technical standpoint. However, Internet Service Providers(ISPs) rarely provide their clients with a static IP address. Another alternative is utilizing a Dynamic DNS (DDNS). The downside of this method is that if the client's network is behind a NAT (Network Address Translation) or firewalls, or if doing port forwarding on their router is not possible, the Access Bridge will be inaccessible. Furthermore, both methods require a certain degree of technical proficiency. Therefore, the chosen solution was to use traffic routing through websockets. This can be achieved by having the device as the connection originator to a remote router and once a connection is established, bidirectional communication is possible. Since implementing such a service is outside the scope of this project, a free service from Dataplicity [43] in used. This service allows creating a tunnel through which the device can be accessed remotely from the internet via a provided user URL.

Data Path

The Access Bridge is comprised of four main parts: an Apache HTTP server [44] (not depicted), Node Red [45], a MOSQUITTO [46] Message Queue Telemetry Transport(MQTT) [47] broker, and a Python script. Figure 4.3 depicts the flow of data from the sensors to HSDP, where it is stored and retrieved by the backend. Dotted lines indicate alternative paths the data can take (i.e., directly from the gateway to the Access Bridge, or through the cloud service).

The Apache HTTP server runs with the default configuration, except for a slight modification. HTTP requests on the path "`*/incoming`" are passed through a proxy that routes the traffic through port 1880, designated to Node Red. The purpose of this is to have a URL that the cloud service can send the sensor data to. For example, if the URL provided by Dataplicity is "`https://obstructive-cod-XXX.dataplicity.io`", the cloud or gateway can send HTTP requests to "`https://obstructive-cod-XXX.dataplicity.io/incoming`".

Node Red is utilized in order to speed up development of the network interface to the MQTT broker. Figure 4.4 depicts the configuration chosen for Node Red. It receives HTTP POST requests via "`/incoming`" and responds with an HTTP request with status code 200. Then, a function parses the JSON contained in the HTTP request, and publishes a message to an MQTT broker. This MQTT outlet publishes to a specific channel, corresponding to one of the values parsed from the JSON message. Each channel corresponds to a specific sensor ID. The MQTT server is pointed to at `localhost:1883`, since it runs on the Access Bridge as well.

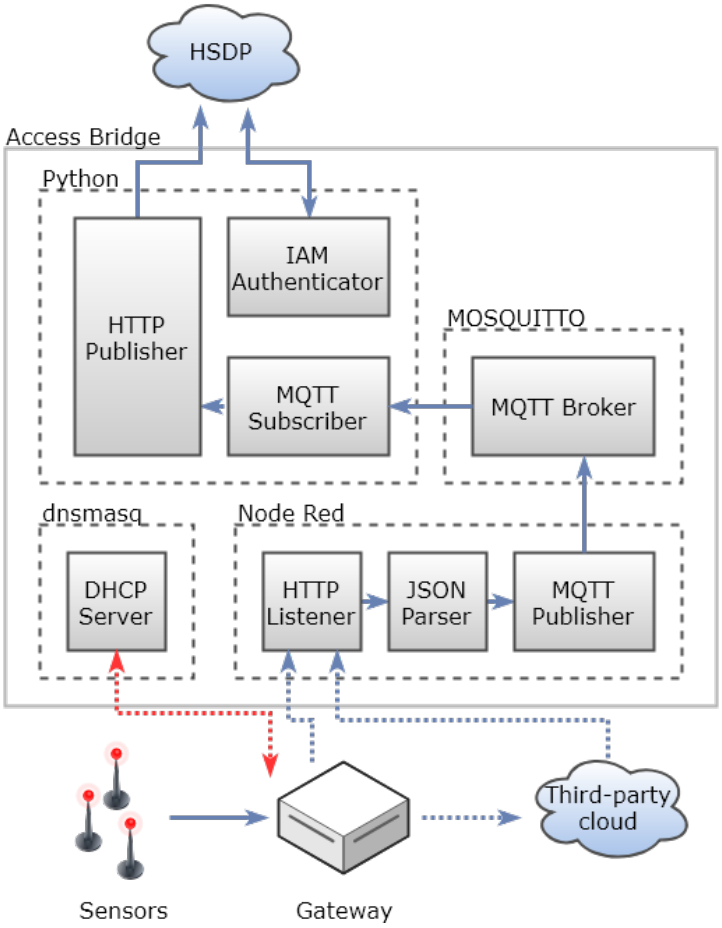


Figure 4.3: Sensor event datapath

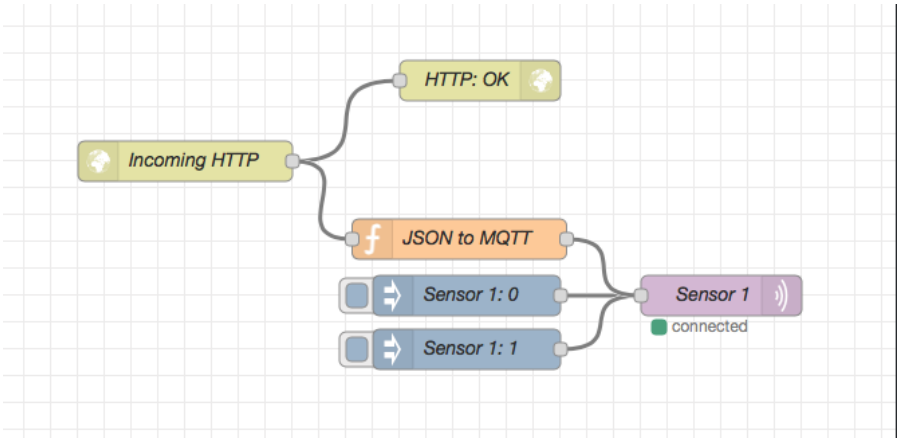


Figure 4.4: Node Red Configuration

A MOSQUITTO MQTT broker service executes on the device. Since it only needs to be accessible from within the Raspberry Pi itself, the default settings are used. Additional security could be configured but for the purposes of the demonstration it can be accessed without credentials. The broker will be receiving values published from Node Red, and a Python script subscribes to the specific channel in the broker where each sensor value is sent in order to receive these values.

Finally, a Python script handles the connection to HSDP and subscribing to the MQTT broker channels. Additionally, it handles various HSDP specific functionalities, like acquiring a unique device ID, Authentica-

tion and token acquisition. Ultimately, it sends sensor event data acquired from the different MQTT channels to the TDR where it is stored to later be fetched by the backend.

4.1.2. Sensor data streaming from origin

For demonstration purposes, we built our solution using the SmartThings Hub (gateway). It allows the user to customize how their devices interact. Furthermore, it has the capability of executing custom applications, called *Applets*, that the user can either code by himself, or download from an application store. One of such applications was developed for this project.

Figure 4.5 describes how the application works. A developer creates a SmartApp (Samsung's brand name for *Applets* that will reside in the cloud. Then, users can install it on their Hub. Finally, SmartApps usually have configuration options which the user can select from a smartphone, which will connect to the cloud and update the Hub accordingly. To the user, it is only a matter of installing an application and choosing simple configuration options. Figure 4.6 shows the interface presented to the user in our SmartApp. The only information required from the user is selecting the sensors that will be used. The code is written in Groovy language.

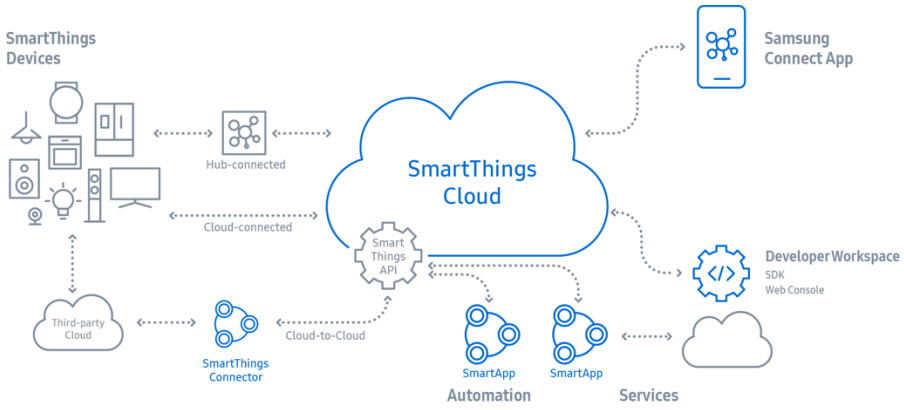


Figure 4.5: SmartThings Cloud

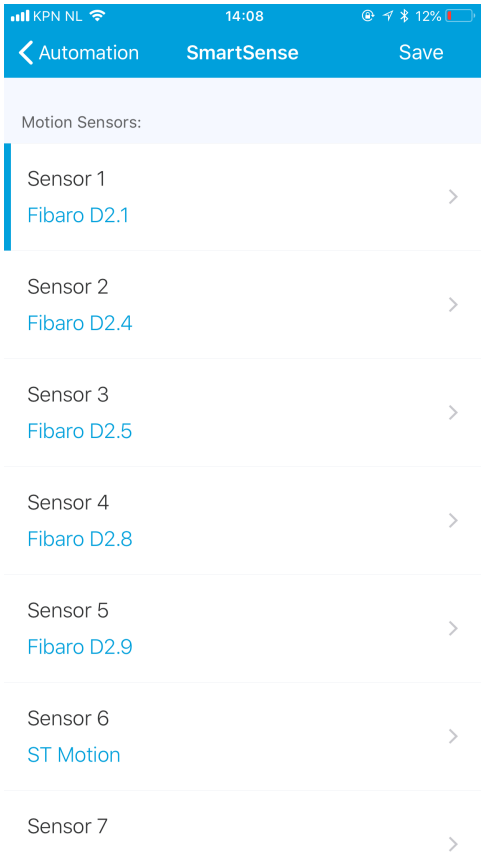


Figure 4.6: SmartApp

4.2. Floor Plan Discovery and Sensor Distribution Estimation

The first step into characterizing the spaces inside the residence and the location of the sensors is to estimate the layout of the rooms in the house and the location of the sensors in it. This step is analogous to the work in [20], but rather than requiring an image or drawing of the floor plan, we generate it automatically based on sensor event data readings.

The generated floor plan is stored and represented as an undirected graph, where nodes indicate sensors and the edges a connection between them. Additionally, the edge distance represents the physical distance between the sensors. The distribution of the nodes will be congruent to that of the sensors, although a mirrored, rotated or otherwise modified but equivalent graph will be acceptable.

4.2.1. System Architecture Overview

In order to establish a telemonitoring system, the client's home must have diverse sensors installed. Commercial binary-based sensors such as PIR motion sensors, door open/close sensors and pressure mats are not uncommon due to their relative cost, low power consumption and the ability to be placed anywhere in the house without the need for a sophisticated infrastructure. These are commonly available as BLE, Zigbee or Z-Wave sensors that connect to a central gateway, from which we would acquire the sensor event data.

Figure 4.7 depicts the process of generating the floor plan. First, event data is acquired from the sensors in the house (Figure 4.7b). This data contains the ID of the originating sensor, the state into which the sensor changed, and a timestamp of the event (Figure 4.7b). We then pre-process the data and generate a time transit matrix based on the median time between immediate observations within pairs of sensors (Figure 4.7c). Finally, we use a technique called Metric Multi Dimensional Scaling on the matrix in order to compute an un-directed graph representing the sensor layout (Figure 4.7d). Each step in this process is described in the following sections.

4.2.2. Data Acquisition

We let $S_{home} = \{s_0, s_1, s_2..., s_{N-1}\}$ denote the set of sensor IDs in the home (Figure 4.7a), where N is the total number of sensors in the house. The data obtained from the sensors is in the form $O_{events} = \{o_0, o_1, o_1, ...| o_i = (s_i, t_i, state_k)\}$, where o_i denotes an observed event, s_i indicates the sensor ID, t_i denotes the timestamp of the event, and $state_k \in \{"on", "off"\}$ denotes the binary state of the event

4.2.3. Data Pre-Processing (1)

The next step in the pipeline is performing a series of transformations on the data. The following steps are taken:

Sorting: Depending on the infrastructure of sensors, gateway and the network connecting to the server, events may be stored in the database in temporal disorder. We sort events according to their timestamp $o_k.t$.

Match events: We ensure that events have matching pair states $o_k.state = "off","on"$ to mitigate for possible data loss or sensor inaccuracies.

- Events with state $o_k.state = "on"$ that have no matching event with $o_k.state = "off"$ following it within a specific time threshold $T_{enforce}$ are enforced one. This new event is inserted in the data, and its timestamp will be:

$$o_{enforced}.t = \begin{cases} o_k.t + T_{enforce} & o_{k+1}.t > o_k.t + T_{enforce} \\ o_{k+1}.t - 1s & o_{k+1}.t \leq o_k.t + T_{enforce} \end{cases} \tag{4.1}$$

- Similarly, events with an $o_k.state = "off"$ with no preceding event with $o_k.state = "on"$ within the threshold time are enforced one. The timestamp of the enforced event will be:

$$o_{enforced}.t = \begin{cases} o_k.t - T_{enforce} & o_{k-1}.t < o_k.t - T_{enforce} \\ o_{k-1}.t + 1s & o_{k-1}.t \geq o_k.t - T_{enforce} \end{cases} \tag{4.2}$$

Enforce on duplicates: When a sensor registers two consecutive "on" events, an "off" event is enforced and *viceversa*. That is, whenever two immediately adjacent events have the same $o_k.s$ and $o_k.state$, an event with the complimentary state is enforced to the first event in the case $o_k.state, o_{k+1}.state = "on"$, and to the second if $o_k.state, o_{k+1}.state = "off"$. The timestamp of the enforced event follows the same rules as in the previous step.

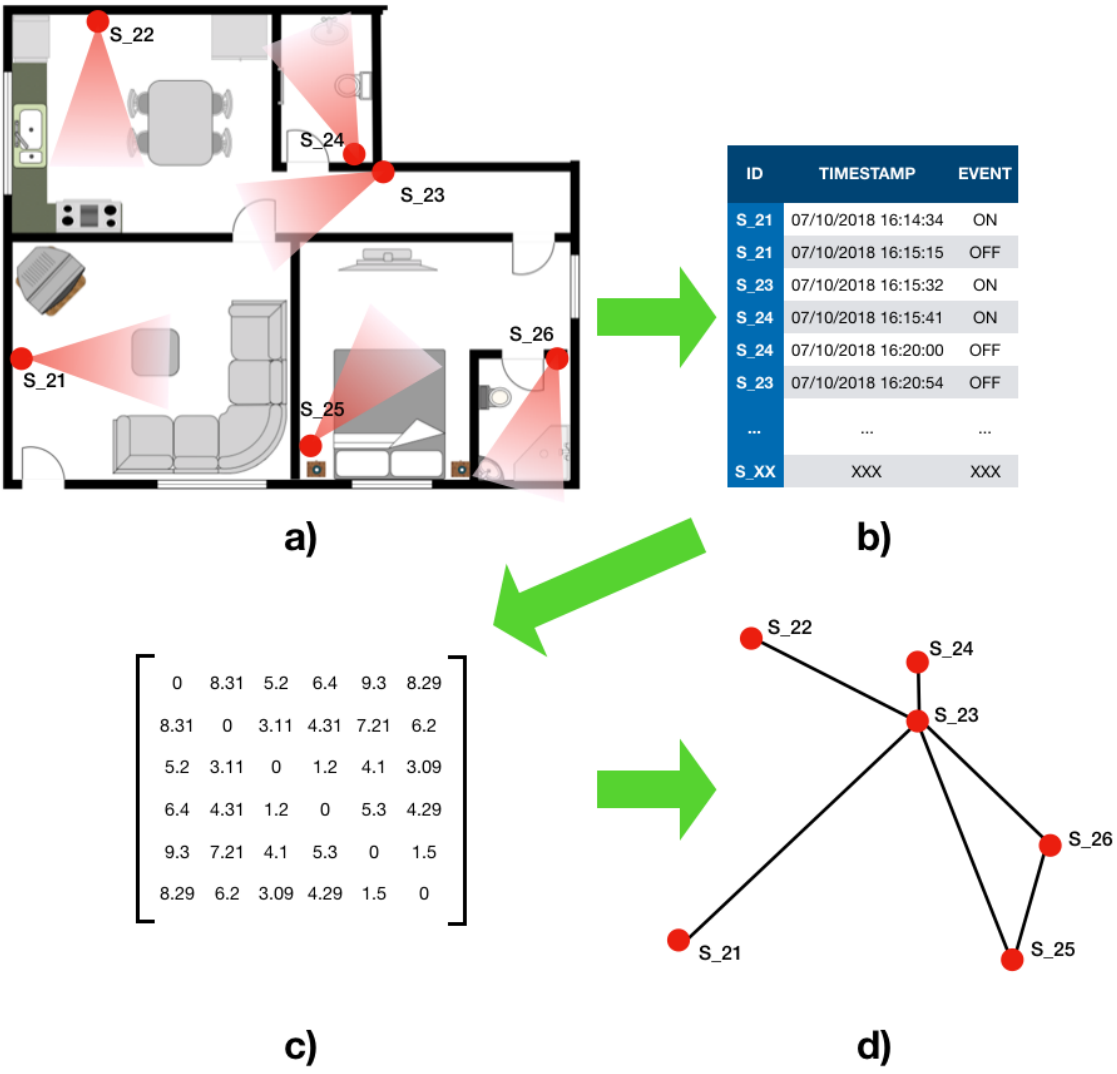


Figure 4.7: Floor plan discovery overview. a) The original floor plan and sensor locations. b) Sensor events captured. c) Time transit matrix. d) Generated floor plan.

4.2.4. Time Transit Matrix

Asymmetric Time Transit Matrix

We compute the median time it takes the user to transition from one room (sensor field of view) into another. In other words, when a person moves between rooms observed by sensors s_i and s_j, the sensors generate activation events o_i, o_j, with their respective sensor ID, timestamp and state. These event pairs are collected for each combination of sensors. We then compute the median of the difference in time between such pairs of event times. These pairs are unique and non-permutable, such that there is a different median value for transitions from $s_i \rightarrow s_j$ and $s_j \rightarrow s_i$. Finally, we arrange these median values into a table and generate a Time Transit Matrix:

$$M_{s_i,s_j} = \{t_{diff} | t_{diff} = Median(T_{ij})\}, \tag{4.3}$$

$$\text{where } T_{ij} = \{\forall_{i,j}(o_i.t - o_j.t) | \exists! k : o_i.t < o_k.t < o_j.t\}$$
$$\text{and where } o_i.s \neq o_j.s, \text{ and } o_i.state = "off", o_j.state = "on".$$

In other words, we calculate the difference between timestamps of immediately adjacent "off" and "on" events of different sensors. We compute the median distance of each pair, and store it in the corresponding

position in the matrix. This results in an asymmetrical matrix where M_{s_i,s_j} indicates the distance, measured as median time difference, from s_i to s_j.

Removing Unreachable Sensors
In specific situations, a sensor is configured in the dataset but no events are captured. Some examples could be a bathroom on a second floor that the client does not use, or an unused guest room. These sensors must be removed from the time transition matrix. Finding them is trivial by exploring the matrix and finding indexes for which the entire row and column have no distance measurements (i.e. 0). That is,

$$S_{unreachable} = \{s_i \forall i | M_{s_k,s_i}, M_{s_i,s_k} = 0\} \tag{4.4}$$

Symmetric Time Transit Matrix
Obtaining a symmetric matrix is done in two steps. First, we obtain the mean value of the distance between sensor pairs in the matrix. That is,

$$M_{s_i,s_j}, M_{s_j,s_i} = Mean(M_{s_i,s_j}, M_{s_j,s_i}) \tag{4.5}$$

Then, we compute the distance between non-adjacent sensors s_p and s_q. For this we implement a recursive shortest hop algorithm, described in Algorithm 1. We store them in the corresponding M_{s_p,s_q} and M_{s_p,s_q}.

Algorithm 1 Shortest Hop Path Finding

```
function FINDSHORTESTPATH(graph, start, end, path = [])
    path ← path + [start]
    if start = end then return path
    if start not in graph then return None
    shortest ← None
    for node in graph[start] do
        if node not in path then
            newpath ← FINDSHORTESTPATH(graph, node, end, path)
            if newpath then
                if not shortest or len(newpath) < len(shortest) then
                    shortest ← newpath
    return shortest
```

4.2.5. Using Metric Multidimensional Scaling to generate a 2D projection in the time domain

Metric Multi Dimensional Scaling [48] is used to transform a matrix containing distance values into a distribution set by coordinates in such a way that the Euclidean distances derived from placing individual elements into an N-dimensional space approximates the original distances between them. We use the MDS library from the Scikit-Learn [49] machine learning package.

As a distance metric, we use time between activation events of sensor pairs in contiguous sequence. We transform the distance matrix into a cross-product matrix and find its eigen-decomposition. The outcome derives into a Principal Component Analysis (PCA), which is then projected into a 2-Dimensional space by generating relative coordinates for each element. Based on these coordinates, we create a graph $G_{transit} = (S_{home}, E_{home})$, where $(s_i, s_j) \in E_{home}$ is a path between two nodes and its length represents a metric of distance in terms of time. In other words, we have mapped a set of points described in a 1-Dimensional metric (time) into a 2D space, such that their distances are isomorphic with the given transition times between them.

4.2.6. Projecting into a 2D distance plane

The floor plan produced distributes the sensors across a 2D plane in time, as opposed to the physical distances that separates them in the physical world. This, however, does not present a problem for our purposes. We assume that the velocity at which an elder adult walks is relatively constant. It then follows

that if $V_{travel} = k$ is constant, the distance between the sensors distributed in the time plane translate to $D_m = k(D_s)$, where D_m is the distance in the physical domain, and D_s is the distance in the time domain. Therefore, our floor plan can be projected into a 2D plane representing the physical distances between the sensors in the real world.

4.2.7. Confidence Ranking

In the generated graph there will likely be edges connecting nodes creating paths between rooms that are not connected. In order to identify such false paths we design a confidence ranking system.

We define $R_{i,j}$ as the number of transitions that were observed between the pair of sensors s_i and s_j. Then, we define R_{max} as the highest R value among all pairs of sensors. Paths between pairs of nodes are ranked according to their $R_{i,j}$ value with respect to R_{max}. In our experimental setup we will make use of this ranking in order to eliminate the edges whose $R_{i,j}$ value is below a certain threshold.

4.3. Room Classification Algorithm

With an estimation of the layout of the house and the distribution of the sensors throughout it, the third major element to fully characterize the sensor layout in the house in order for the ADL algorithms to recognize human activity, is labeling the rooms in which the sensors reside.

For this section, we use an extended version of the available data. We will use observations o_i as described in Section 4.2.1, with the addition of information on the room in which the sensor is. In other words, we have labeled data that will be used for training the model, as well as evaluating the performance of the classification algorithm. We therefore define $O'_{events} = \{o'_0, o'_1, o'_1, ... | o'_i = (s_i, t_i, state_i, room_i)\}$, where o'_i denotes an observed event, t_i denotes the timestamp of the event, $state_i \in \{"on", "off"\}$ denotes the binary state of the sensor (activate or deactivated), and $room_i$ indicates the room in which the sensor is located.

For simplicity, this work will assume only one sensor per key location. We define locations as $L = \{$*"Kitchen", "Bedroom", "Bathroom", "Toilet", "Living Room", "Hallway"*$\}$ and key locations as $L_{key} \subset L = \{$*"Kitchen", "Bedroom", "Bathroom", "Toilet", "Living Room"*$\}$. The objective of this module is to perform a mapping of a subset of the sensors into L_{key}. That is, we find a matching function $\Gamma : S_{home} \rightarrow L_{key}$ given the observation sequence O'_{events}. This will be done in several steps. First, the data will be further pre-processed in order to favor our supervised learning model. Then, by means of heuristic approaches, we will identify two of the *key locations*: the toilet and the bedroom. Next, we proceed to extract diverse features from the sensor event data in order to train an SVM model. Finally, we use the SVM model to classify the remaining *key locations*: the bathroom, kitchen and living room. Figure 4.8 depicts the entire room classification process.

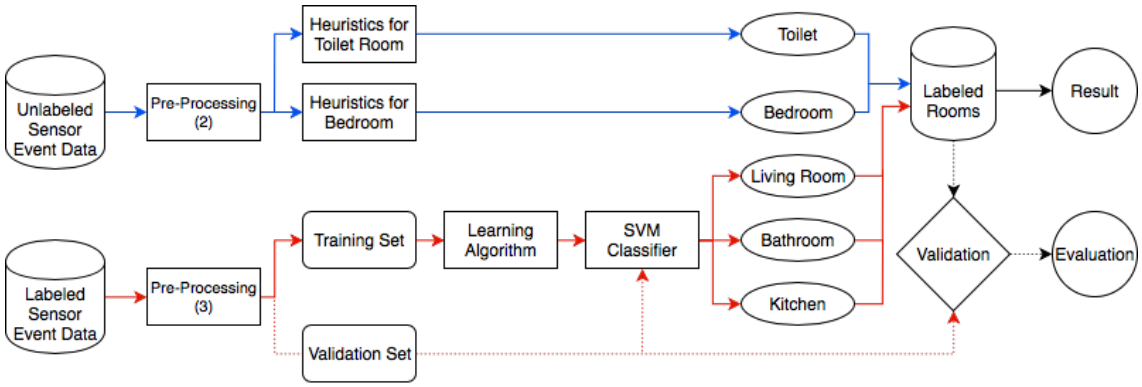


Figure 4.8: Room classification pipeline

4.3.1. Data Pre-Processing (2)

To ease the process of room classification, we perform some transformation of the raw data retrieved from the sensor in a number of steps:

Data Segmentation: Events will processed on a basis of 24-hour frames. Each one of these frames is split evenly into N_{seg} *epochs* per day, such that a set $K_d = \{\kappa_0, \kappa_1, ..., \kappa_{N_seg-1}\}$ holds the N_{seg} epochs of day d. In our implementation, we use $N_{seg} = 48$. That is, 48 epochs of 30 minutes each.

Span computation: We transform the set of observed events O_{events} into a set of spans Δ_s:

$$\Delta_s = \{\forall D_k | D_k = o_i.t - o_j.t\} \tag{4.6}$$

$$\text{Where } o_i.t > o_j.t, o_i.s = o_j.s, \exists! k : o_i.t > o_k.t > o_j.t,$$
$$\text{and } o_i.state = "off", o_j.state = "on"$$

That is, the time between every sensor activation and the immediate corresponding next matching deactivation. In other words, the spans of time that the inhabitant spent in the room corresponding to the sensor activated.

Minimum span length filtering: We define a constant D_{min}, in seconds. Spans with a duration $\Delta_k < D_{min}$ are ignored. PIR motion sensors are prone to noise induced from several sources like changes in temperature or animals crossing their line of sight. Therefore we attempt to remove as many of these unwanted activations

as possible. For our experiment we used $D_{min} = 20$s. The reasoning behind this is that we are classifying the bedroom, kitchen, bathroom and living room with this method. ADL that take place in these rooms, such as cooking, showering or dressing, do not take less than the amount of time chosen. By removing these events, we are mostly removing transitional events (i.e., crossing a hallway or retrieving an item from a room) which are sporadic and do not follow a conventional circardian rhythm (e.g., they do not repeat consistently thoughout different days).

Minimum sensor activation: We define a constant A_{min}. For each sensor, we ignore data from days where a minimum of A_{min} events are not detected. This is to avoid false correlation between unrelated rooms with very low activation rates during the supervised learning step. As an example, a house may have a bathroom that is almost never used by the inhabitant and therefore has very low activation rate. However, if the user leaves home for the weekend, all sensors in the house will have low activation rates for those days, which will result in most of them being classified as bathrooms. Based on observations from the datasets available, we select $A_{min} = 5$.

Late night and Early Morning Data: Analyzing the datasets, it became evident that sensor information after 23:00 and before 3:00 is not very descriptive of the client's activity. Most of the times it would either include false activations, possibly due to pets, or no information at all. Therefore this segment of the data is ignored.

4.3.2. Heuristic Approaches

To simplify the supervised classification of the rooms, we apply two heuristic approaches to identify two of the key rooms: toilet and bedroom. These methods are based on observation and analysis of the available sensor data, as well as intuition based on human behavior.

Bedroom

Common sense dictates that the first room in the house to register activity on a regular day would be in the bedroom. A person normally wakes up in bed and spends at least a few minutes there before heading either to the shower, toilet or kitchen to make breakfast. We corroborate this assumption by looking at visualizations of our data. Figure 4.9 depicts the average amount of active spans per sensor (and its corresponding ID), per κ_i epoch of the day, throughout 83 days of measurements in an apartment inhabited by one elder person. A high value in a specific epoch indicates that the person is consistently active around this time in this room on different days. It can be seen that the room with the highest number of activations early in the morning is the bedroom. This was valid for all the datasets we used for this experiment.

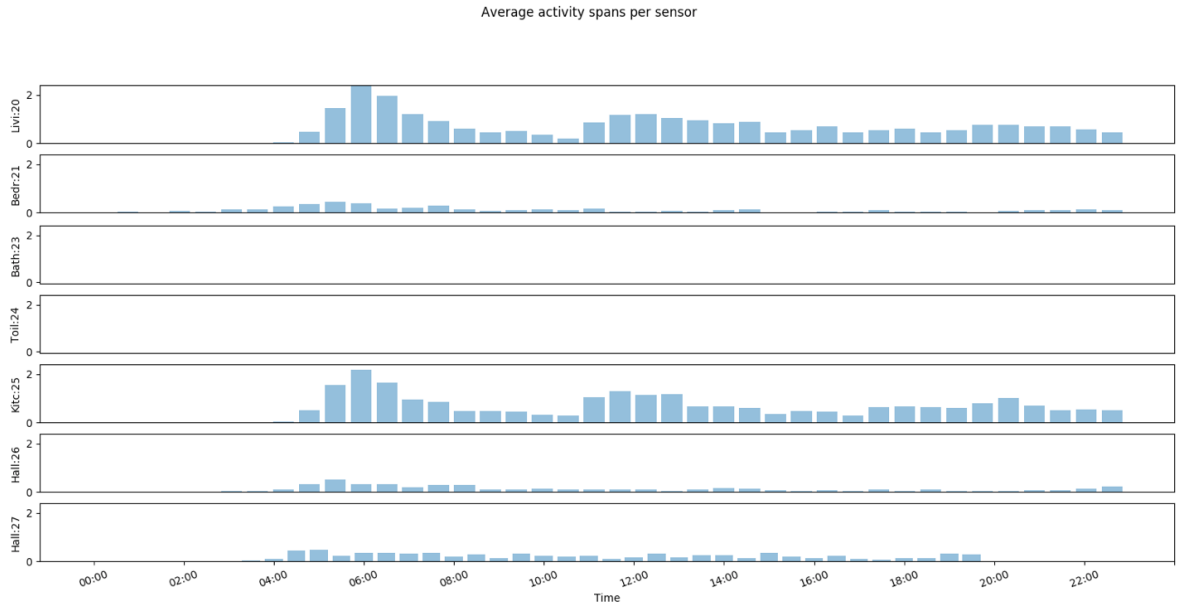


Figure 4.9: Average activity spans throughout 83 days

We compute the average number of activity spans and starting from 3:00 we begin to observe which sensor has the highest activity in each epoch κ_i, up until 6:00. A likelihood factor for each sensor accumulates a value

for every epoch that the corresponding sensor had the highest number of average activations. This value is a weight, scaling down from 1 through 0 as the corresponding time of the epoch increases. In other words, the earlier the epoch is, the higher the weight value assigned will be. The counter with the highest value will indicate the room in which the bedroom is located.

Toilet

Another observation made was that consistently throughout the different datasets we used, the toilet was always the room with the lowest average active spans, as well as the lowest number of activations overall. Once more, using a heuristic approach we validate this information out of our knowledge in human behavior. Visits to the toilet take seconds, up to minutes, and therefore their spans dictate presence in one or at most two κ_i epochs per visit. At the same time, toileting normally happens at inconsistent times throughout different days and therefore the values of the average number of activations per epoch is even lower. This explains why on Figure 4.9 at first glance it would appear that the toilet sensors have no activity registered on them.

Labeling the toilet room is therefore a simple matter of either finding the sensor with the least average number of activations throughout the day, or the one with the lowest cumulative average number of activity spans in all epochs of the day. Both methods were tested and the results were equally satisfactory.

4.3.3. Data Pre-Processing (3)

Further steps, including those described in section 4.2.3 and 4.3.1 are carried out in order to prepare the data for supervised learning classification. Both the training and validation data sets follow the same pipeline:

Daily activation histogram: As described earlier in Section 4.2.3, data for each day is divided in equal length epochs. For each epoch κ_i we then compute the amount times each sensor s_k was activated within the window time frame, per day. This is stored as a 3-dimensional histogram H_{s_i,κ_j,d_k}, where s_i is the sensor ID, κ_j is the epoch, and d_k indicates the day sampled.

Histogram smoothing: The individual epochs κ_j of the histogram H_{s_i,κ_j,d_k} will be used as features for classification. One drawback of this method is that correlated samples could be taken as completely unrelated due to phase shifts. Figure 4.10 illustrates an example of a histogram of a given sensor to be classified compared to a sample in the training set, with an epoch length of 15 minutes. Samples from 4.10a will be seen as uncorrelated by the classifier since the events happened during different epochs and each epoch is a different feature. However, by applying a Hamming window smoothing to the signals the effect of the shift in time of the events is minimized. Figure 4.10b shows the effect of a Hamming window of size 5 and Figure 4.10c of a window with size of 10. A window size of 5 (2.5 hours since an epoch is 30 minutes) gave the best result in our tests.

Ignore hallways: For the training set, we will remove the hallway data from the dataset. The reason for this is that hallways (or otherwise unlabeled rooms) are usually transitional spaces between *key rooms*. Therefore, the occupancy of such spaces by a user is completely dependent of the layout of the house. This is opposed to *key locations* like the bedroom that have regular and identifiable features, such as the average visiting time or the time spent in them. An example can be seen on Figure 4.9, where sensors 26 and 27, both located on hallways on the same house, have completely different behaviors.

Class balancing: For the training set, we balance the number of samples from each of the different classes by randomly removing elements from the over-sampled class.

4.3.4. Feature Extraction

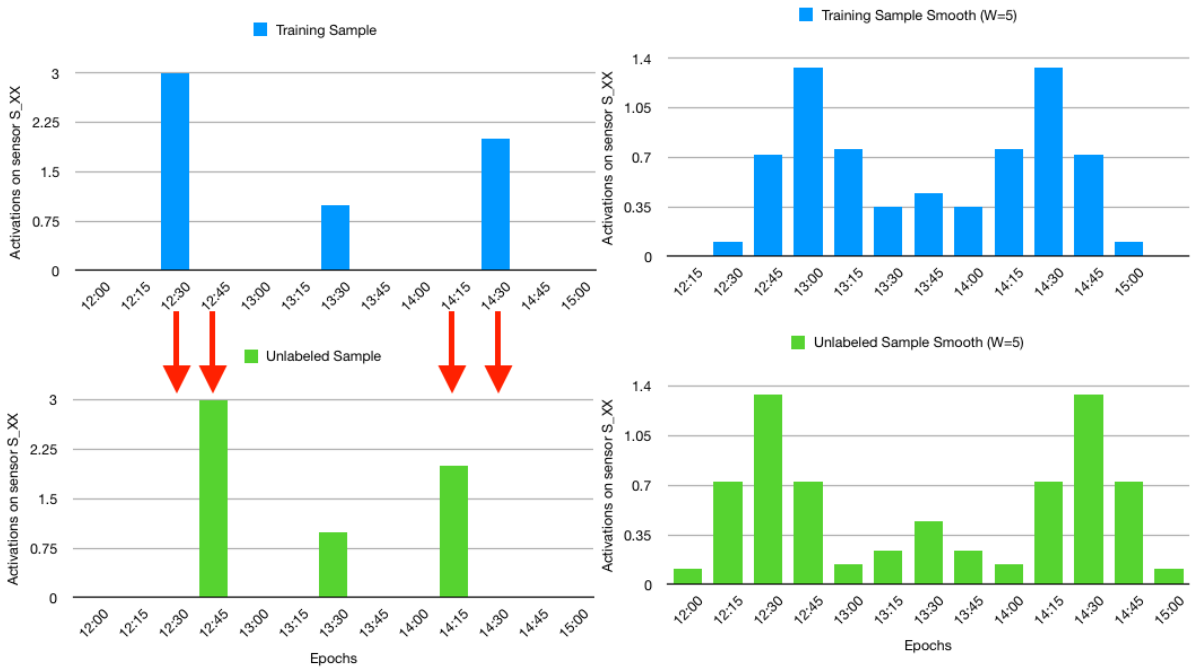
For the supervised classification model, we extract the following features from the data:

Activation Histogram

With a combination of the approaches in [15] and [17] of dividing data into *epochs*, the proposal for future work on using *activity time* in [17], as well as the use of *number of sensor firings* as features, we propose using a histogram of the amount of sensor activations per day for each epoch as a feature.

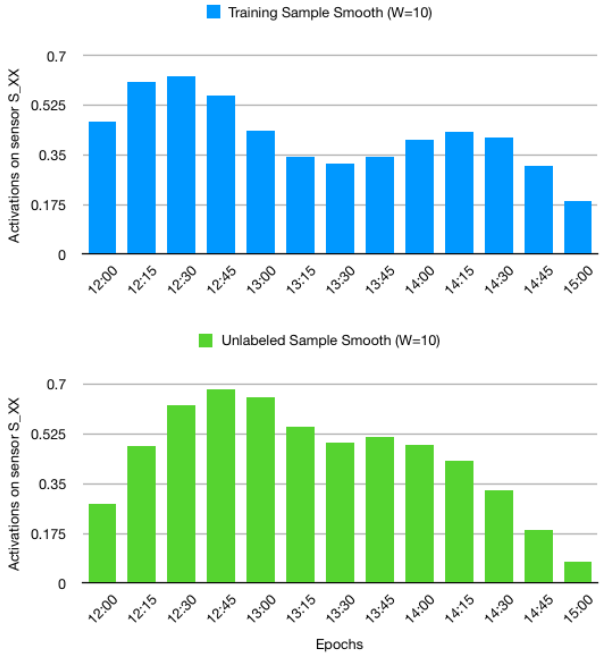
Variance

We base our proposal based on a combination of the proposal in [17] for future work of using deviation in sensor data, and from [19], the use of mean and variance computed from data tokens. For each sensor and epoch in the histogram, we compute the variance of the samples (days), and use it as an additional feature.



(a) Original samples

(b) Samples with a Hamming window of size 5



(c) Samples with a Hamming window of size 10

Figure 4.10: Effects of Hamming smoothing on daily activation histogram

Dwelling time ratio
Lastly, based on the idea in [15], we compute the total amount of time the user spent in each room (sensor) and divide it by the total dwell time in the house. In other words, we use the percentage of time spent in each room as a feature.

4.3.5. Room Classification with SVM Supervised Learning

For the final stage of our methodology, we proceed to identify the three remaining *key locations* in the house: the kitchen, bathroom and living room. This gives a total of 5 unique locations identified, including the previously identified toilet and bedroom.

Derived from the research explored in section 2.1, we train an SVM using the features described in the previous section, and classify unlabeled samples using a linear kernel. We use the SVM library from the Scikit-Learn package [49].

SVM [50] is a well-known method used to classify data in a non-probabilistic manner. It determines non-linear boundaries between classes by mapping data into a high dimensional space where it can be classified by a hyperplane decision boundary. This hyperplane is computed to maximize the distance from eaxh data point, separating them into two classes. It is then mapped into a nonlinear surface using kernel-based transformations.

SVM solves the multiple classification problem by doing binary comparisons between all classes. Therefore, a classification problem with K classes would be separated into $K(K-1)/2$ separate training and classifying instances. Each of these problems will have an output value which will determine the class voted for, based on a numeric discriminating threshold. Finally, the votes from all instances are counted and the sample is given the label with the highest number of votes.

The output of the classification is a list of probabilities, per sensor, for each of the three classes. We assign the corresponding label to the sensor with the highest probability until all three classes have been assigned a sensor. The remaining unclassified rooms are labeled as "hallway".

5

Test Scenario, Results and Discussion

In this chapter we present and discuss the test methodology and results of our study. This is done in two separate sections. In the first section, we discuss the network architecture for data acquisition of arbitrary sensors and results obtained in the demonstration environment at Philips Research. In the second section we present the results obtained by our floor plan discovery and room classification algorithms with data obtained from participants living in the Eindhoven region.

5.1. Arbitrary IoT Sensor Data Acquisition

The objective of this module is to retrieve event data from arbitrary sensors, regardless of their brand, make and the communication protocol they use, to send to the cloud environment selected for our experimental setup, HSDP.

5.1.1. Test Scenario

HomeLab

Philips HomeLab is part of Philips Experience Lab, a test environment located at the High Tech Campus in Eindhoven, The Netherlands. It is a full-size model of an apartment with different rooms, appliances and furniture. The layout (floor plan) of the Experience Lab can be observed in Figure 5.1.

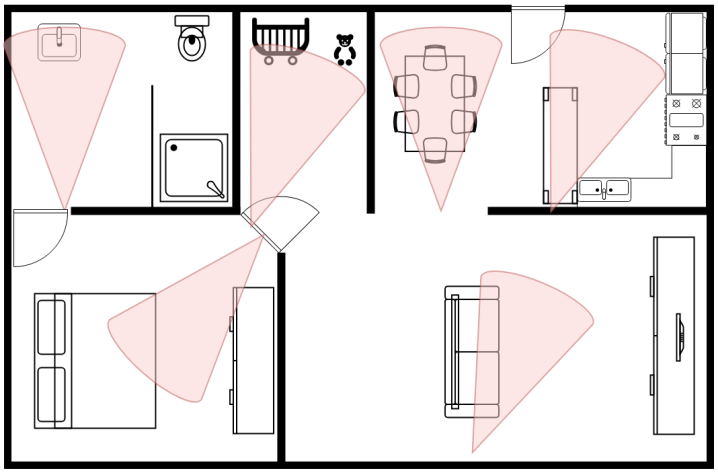


Figure 5.1: Experience Lab floor plan

In order to test the data acquisition of our setup, a set of 6 motion sensors and 1 open/close sensor from different brands (Samsung [35] and Fibaro [51]) were placed in different rooms. Figure 5.2 shows one of these sensors, positioned on the door looking into the bedroom. The monitored rooms were the kitchen, living room, nursery, bedroom and bathroom. The sensors are connected via Z-Wave to a Samsung SmartThings Hub and configured prior to the test in order to simulate a real and functional smart home environment. In Figure 5.1, the orange cones represent the field of view of the motion sensors.

Access Bridge

An Access Bridge was installed in a Raspberry Pi, connected to a WiFi network. It was configured to serve as a DHCP server to the Samsung SmartThings Hub connected to it via Ethernet port in order to provide it with internet connection. Both devices were positioned over a counter in the kitchen. The Access Bridge is connected to internet via Wi-Fi. The Access Bridge automatically runs our Python script which configures the device and connects to HSDP; no further configuration is needed.

Gateway

The SmartThings Hub was configured with the 7 sensors and running our custom test applet, allowing it to transmit sensor data directly to the Access Bridge. All the configuration is done via the SmartThings smartphone app. Once downloaded and installed, the user opens the applet which presents a menu to select which sensors to use for activity monitoring.

Figure 5.2: A Fibaro motion sensor positioned in the bedroom at the Experience Lab

5.1.2. Test Sequence

A person walks around the house for 30 minutes, annotating every room visited with the exact time. The person takes different paths and was occassionaly sitting still. Using a third party SSH (Secure Shell) application on a smartphone, the person connects to the Access Bridge remotely in order to monitor every sensor event stored in HSDP via the script mentioned in section 5.1.1.

The person walks in through the main entrance (kitchen), from where he walks to the bathroom visiting every room in that trajectory, and back. From there, he walked the same path, occasionally skipping the hallway or the living room. After a period of sitting in the kitchen, he walks to the living room and exits the building.

5.1.3. Results

A total of 84 events were recorded by the person. 100% of them were correctly stored in HSDP. We confirm this by retrieving the lists of elements directly from the TDR in HSDP, emulating a generic backend requesting sensor data to process. In other words, there was no data loss in the process of transmitting the sensor events from the gateway to HSDP.

Figure 5.3 presents the sensor event data transmitted from the Access Bridge to HSDP, and retrieved by the emulated backend. Reading this data demonstrates that the designed access method was effective. Also, the data reached HSDP independently of the infrastructure connecting the sensors. The SmartThings Hub acts only as the gateway interconnecting the sensors and publishes the data to the MQTT service running on the Access Bridge. Any generic gateway with capabilities of sending HTTP requests would have achieved the same result. In the figure, the dots represent events (on/off) registered at the sensors, and a line connecting them represents a span in which the user was present in the room. The green and red colors are generated by the ADL algorithms and are irrelevant for this study. The graph clearly shows the pattern followed by the person. It starts at the kitchen, walking through the livingroom, hallway, bedroom and bathroom, then walking back to the kitchen and repeating the same path with slight variations. Finally, we can also see two instances of the person sitting still in the kitchen, right before doing a fast traversal of the house towards the kitchen.

The data depicted in this figure shows the raw data an ADL recognition algorithm would receive from the sensors. For this reason, the person appears to be in two different rooms at the same time. This is due to some areas being within range of two sensors, and the particular *timeout* [1] values these sensors have. However, these algorithms have mechanisms to process this raw data in order to determine where the person truly was. Discussing these is out of the scope of our investigation.

In an second test, an open/close sensor was added to the setup while the transmitting and logging scripts were kept running. The purpose of this test was to demonstrate the flexibility of the Access Bridge without requiring any reconfiguration. Once the sensor was paired to the gateway, it took a few seconds to include it to the configuration via the applet, and the sensor readings immediately started being transmitted to and stored in HSDP without any need for reconfiguration on the Access Bridge side.

[1] The timeout value of a sensor is a set amount of time the it remains in the "on" state after motion has ceased before going into "off" state.

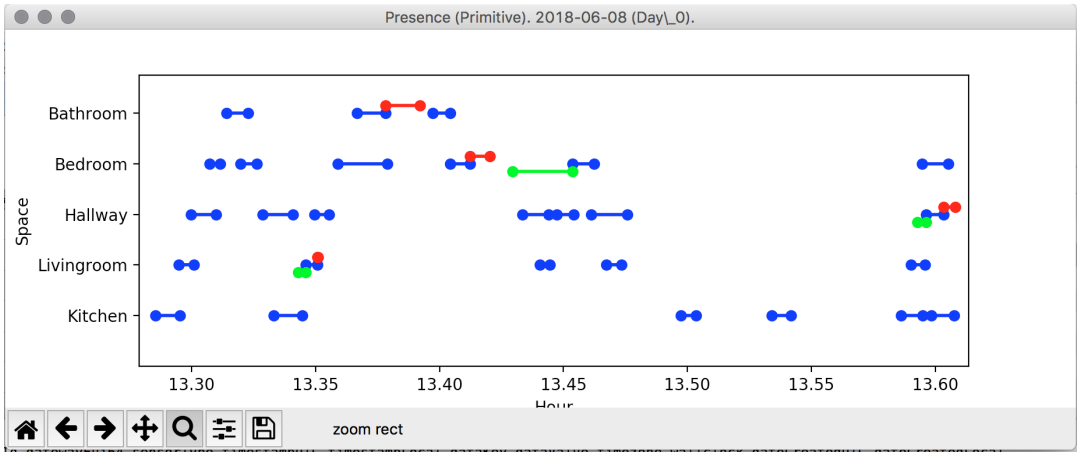


Figure 5.3: Sensor activity events

5.2. Floor Plan Discovery and Room Classification

We are now evaluating the designed solution for calculating the layout of the floor plan, estimating the distribution of sensors located in a house, and labeling the type of room in which these are.

5.2.1. Test Scenario

Previous to this work, Philips Research carried out tests in 12 different houses in the Eindhoven region in The Netherlands, denominated *Living Labs*. These are real houses inhabited by single elderly people who volunteered for the experiment. The data collection has been approved by a Philips internal ethical committee and the volunteers have given their written consent, after having been clearly informed. The datasets derived from these tests were used for the purposes of this thesis. Table 5.1 shows the different datasets available. For the purpose of this investigation, we selected 3 households where no pets were present, and 1 where there is likely to be one (LL04). In addition, we only used the sensor data from the motion sensors.

Dataset	Sensors	Motion Sensors	Days Measured	Events	Pets	Used
LL01	12	9	124	149,304	✗	✓
LL02	11	8	78	173,751	?	✓
LL03	11	9	182	807,893	✓	✗
LL04	12	9	98	78,533	?	✓
LL05	12	9	103	118,366	✓	✗
LL06	10	8	81	59,352	✓	✗
LL08	10	8	83	44,240	✗	✓
LL10	10	8	40	137,831	✗	✗
LL11	9	7	83	86,901	✓	✗
LL12	12	9	82	45,957	✓	✗

Table 5.1: Philips Living Labs Datasets

In each house, sensors were positioned in each key location and in hallways. In some cases, more than one sensor is placed in a single key location. Volunteers were asked to perform their typical daily routine and were given *tablets* to annotate the different activities they were performing. These annotations were meant for ADL recognition research purposes and were not used in our study.

To evaluate the floor plan discovery, sketches of the floor plans are used. These were hand-drawn and provided by the technician installing the sensors in the houses. Without precise dimensions of the house and rooms it is difficult to perform a quantitative evaluation of the results obtained regarding the distances calculated between sensors. Therefore, we present the graphs obtained using time as a distance metric, and discuss the results in a qualitative manner.

To evaluate the room classification, labeled samples from the Living Labs datasets are used. The following steps are followed to classify the rooms:

1. In houses with more than 1 sensor per key location, we remove the events from the one with the least amount of events recorded. This follows our constraint of only having one sensor per key location.

2. We perform our heuristic classification method to identify sensor in the toilet. Samples from this sensor are removed from the dataset.

3. We perform our heuristic bedroom classification algorithm to identify the sensor in the bedroom. Samples from this sensor are also removed from the dataset.

4. We use our SVM classifier to identify the bathroom, kitchen and living room from the remaining samples. A majority vote based on the total amount of predictions for each sensor is used in order to label these three key locations.

5. The remaining unlabeled sensors are labeled as hallway.

5.2.2. Results

The results from the floor plan discovery are shown in the following sections. For each of the houses used in our study, we evaluate the performance of our methodology with the following:

Floor Plan Discovery

- *Floor plan sketch:* A floor plan based on the drawings and annotations made by the technician is demonstrated. The red symbols represent the sensor locations, as well as the direction they point at. The blue crosses represent rooms that were not used for the trials, mostly due to them being unused or irrelevant to the study (for example laundry, guest or storage rooms). Next to each sensor a text label indicates its sensor ID.
- *Discovered Floor Plan Graph:* The computed floor plan is presented as a graph. Nodes represent sensors, edges represent a path between them, and the edge length indicate the distance between them. Distance in both axes is shown in the time domain, in seconds. For example, an edge of length 5 indicates that a person takes 5 seconds walking from one sensor to the other.
- *Regular and Minimal Confidence Graphs:* We present two different graphs for our experiments. For the regular confidence graph we display all computed edges, regardless of their confidence ranking $R_{i,j}$. For the minimal confidence graph, we exclude the edges which are below the minimal confidence rank. These ranks are indicated in the graphs by different colors as a visual aid to observe the relations between nodes:
  - A green path indicates high confidence: $R_{i,j} > 0.50 * R_{max}$.
  - A yellow path indicates minimal confidence: $0.02 * R_{max} < R_{i,j} \leq 0.50 * R_{max}$.
  - A red path indicates low confidence: $R_{i,j} \leq 0.02 * R_{max}$.

Room Recognition

The results for room classification are shown in three steps:

- *Heuristic Toilet Classification:* The sensitivity and specificity achieved by this heuristic classifier are presented and discussed. Supplementary plots of the number of spans recorded for each sensor are provided, along with tables with the sensors with the lowest activity recorded per day. Sensors not appearing in the table reported 0 days with the lowest span count.
- *Heuristic Bedroom Classification:* We evaluate our second heuristic classifier by analyzing sensitivity and specificity achieved. We supplement our results with plots of the likelihood factor of each sensor being in the bedroom. The higher this unit-less scalar is for a sensor, the higher the probability is.
- *SVM Supervised Learning Classification:* The confusion matrix for the SVM classifier are presented and discussed. Precision and Recall are observed to evaluate the accuracy of the results.

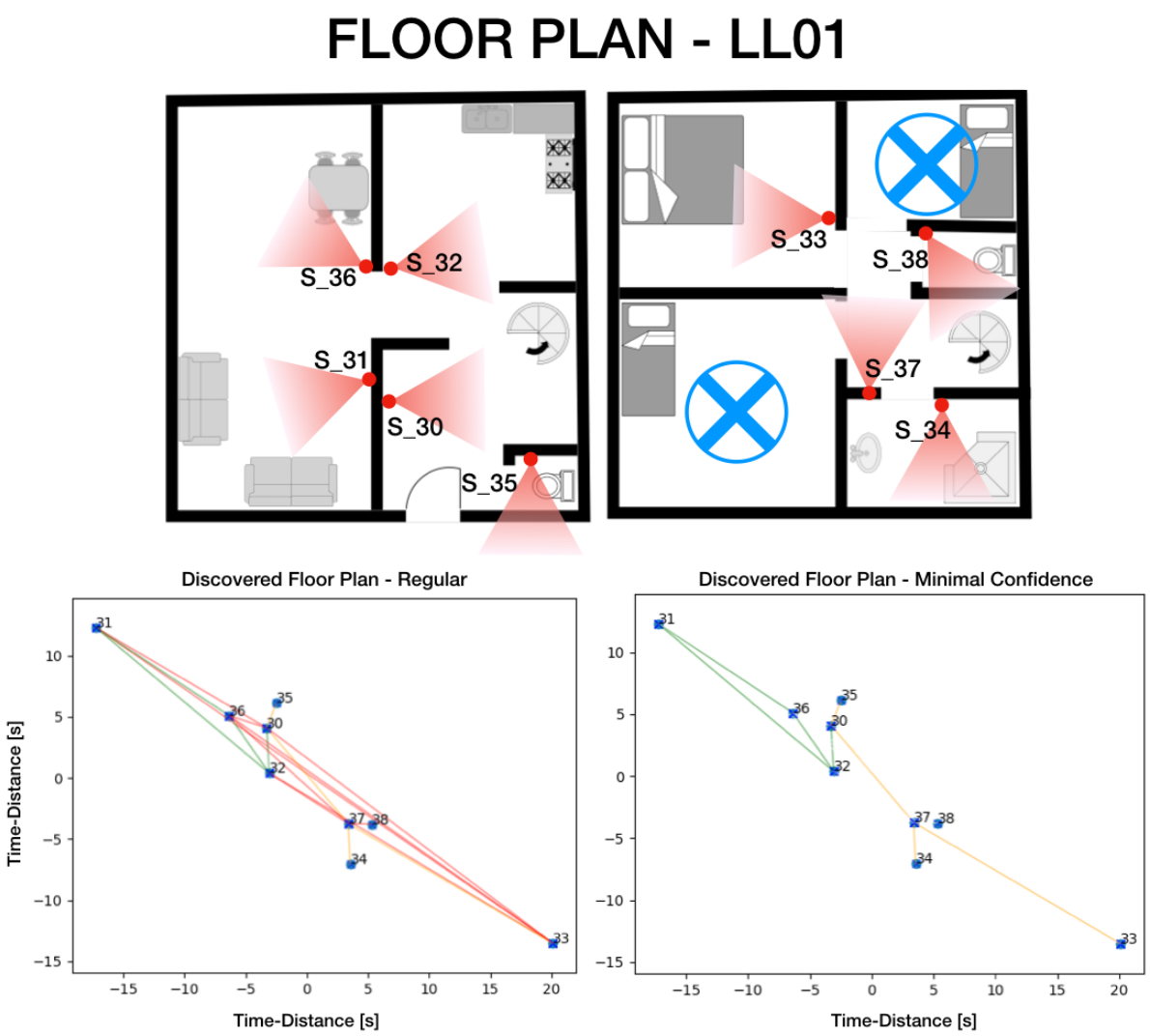


Figure 5.4: Floor plan discovery of house LL01

Test home LL01, shown in Figure 5.4, consists of a two-story house with a single occupied bedroom, two unoccupied rooms, two bathrooms and a shower. It is inhabited by a single elder person with no pets. 124 days of data were recorded and used for this study. After removing the low confidence paths, we can analyze the floor plan obtained.

The graphs are displayed in an (x, y) coordinate system in units of seconds for both axes. Negative numbers are only an effect of the position chosen for the origin of this coordinate system, and have no physical meaning other than distances.

The first observation is how the algorithm separated both stories. A single path is connecting s_{30} and s_{37}, both at the stairs in the top and bottom floors. s_{37} is the only path to reach the shower, bedroom, and top floor toilet. However, the path between s_{37} and s_{38}, the toilet, was of low confidence and therefore eliminated. This is either caused by incorrect sensor positioning, or it indicates it is not used frequently.

In the bottom floor we observe a very accurate representation of the sensor distribution. Sensors s_{36}, s_{31}, and s_{32} are connected. Inspecting the original floor plan, we can observe that it should be possible to reach the stairs, s_{30}, from the diner and living room without passing through s_{32} in the kitchen. A possible reason for s_{32} being the only path to s_{30} is that the field of view from the sensor in the kitchen reaches the corridor leading to the stairs.

The last observation is related to the distances calculated. Most of the sensors are 5 seconds away or less of their adjacent ones. However, the ones in the bedroom, s_{33}, and living room, s_{31}, seem to be considerably far. These are large rooms, and in these a person is normally sitting or laying down. It then takes longer time to leave the field of view of the sensors, depending on where they are positioned, and therefore our algorithm

considers this distance to be long.

In conclusion, the floor plan discovery algorithm correctly estimated the distribution of and paths between sensors. It also computed adequate distances between them.

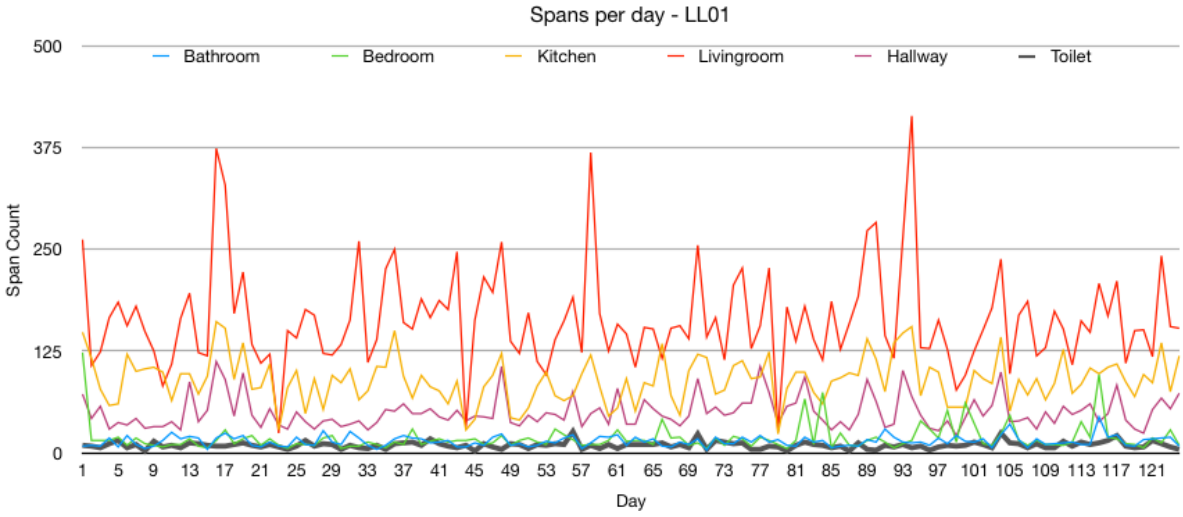


Figure 5.5: Spans recorded per day for each sensor type for house LL01

Figure 5.5 shows the number of spans recorded for each sensor type in house LL01. For our classifier, in the ideal scenario the black line corresponding to the toilet should have the minimum value compared to the other classes. This would indicate consistent low activity in that specific room and therefore, according to our algorithm, would assign the "toilet" label.

Sensor	Location	Count
S_35	Toilet	96
S_34	Bathroom	15
S_33	Bedroom	13

Table 5.2: Days recorded with the lowest span count per sensor in house LL01

Table 5.2 shows that S_{35} was the sensor with the least amount of spans per day and therefore was correctly labeled by our heuristic classifier. The sensitivity and specificity achieved are presented in Table 5.3. A high specificity is expected from this methodology. The reason is we have N sensors to classify and only 1 will be labeled "toilet" and the rest as "non-toilet", therefore there can be at most 1 false positive. For 7 sensors, the lowest specificity possible would be 5/6 = 0.833. The worst scenario in terms of sensitivity would be with only two sensors having the highest low span count. Therefore, the condition for our classifier to guarantee finding the correct sensor in the toilet is having a sensitivity of at least 0.5.

Sensitivity	Specificity
0.774	0.962

Table 5.3: Sensitivity and specificity of Toilet Heuristic Classifier in house LL01

To evaluate the performance of our heuristic approach to find the bedroom, we present the results of the calculated likelihood factor for each sensor. Figure 5.6 shows the scaling of the factor between 3:00 and 6:00 in the morning. For house LL01, the only sensor to register the highest amount of active spans for each of the 6 30-minute epochs between these times was S_{33}. The factor is computed using the average active number of spans per epoch throughout the entire dataset. Therefore, there is only one single classification done and both sensitivity and specificity are 1 since the prediction was correct. However, the downside of this methodology is that several days of data are necessary before this heuristic classifier can provide with reliable results.

Figure 5.7 and Table 5.4 show the confusion matrix and the recognition accuracy result for the SVM classifier, respectively. For this, as well as the rest of the experiments, precision and recall for the hallway class

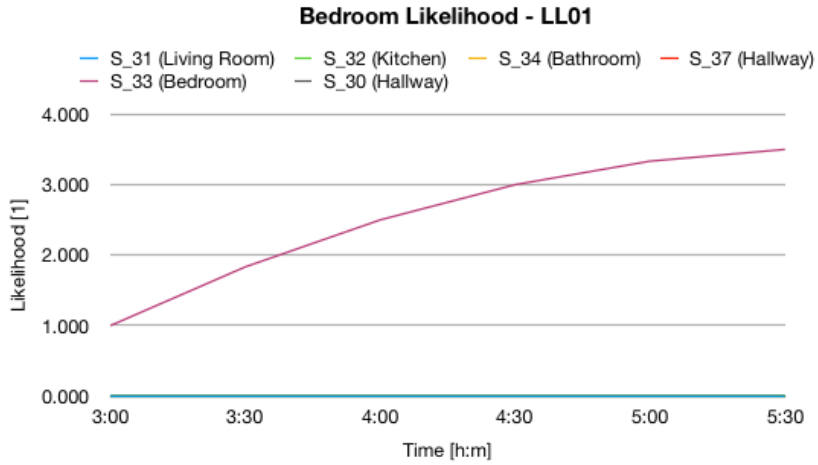


Figure 5.6: Likelihood factor of each sensor for house LL01

will be 1.0, unless one of the other key locations were assigned the wrong sensor. That is because there is no hallway data in our trained model. Instead, once all unlabeled samples have been assigned a (provisional) class, we find the sensor with the highest amount of designations for each label and assign it to that class. The rest of the unlabeled sensors are assigned to the hallway class.

For the house LL01, our SVM classifier had an overall precision of 0.94. The satisfactory results in this home could be attributed to the physical separation between the kitchen, bedroom and bathroom. This avoids confusing the classifier with data that indicates simultaneous presence on different sensors. Additionally, the activities a person performs in the three rooms commonly happen at consistent times throughout the day. For example, a person normally sleeps and eats on a periodic basis dictated by the human circadian rhythms, and the living room is a place where the person relaxes often in the afternoons.

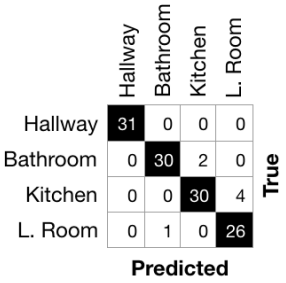


Figure 5.7: Confusion Matrix for the SVM classifier in house LL01

	Precision	Recall	Samples
Hallway	1.00	1.00	31
Bathroom	0.97	0.94	31
Kitchen	0.94	0.88	32
L. Room	0.87	0.96	30
Total	0.94	0.94	124

Table 5.4: Precision and recall for the SVM classifier in room LL01

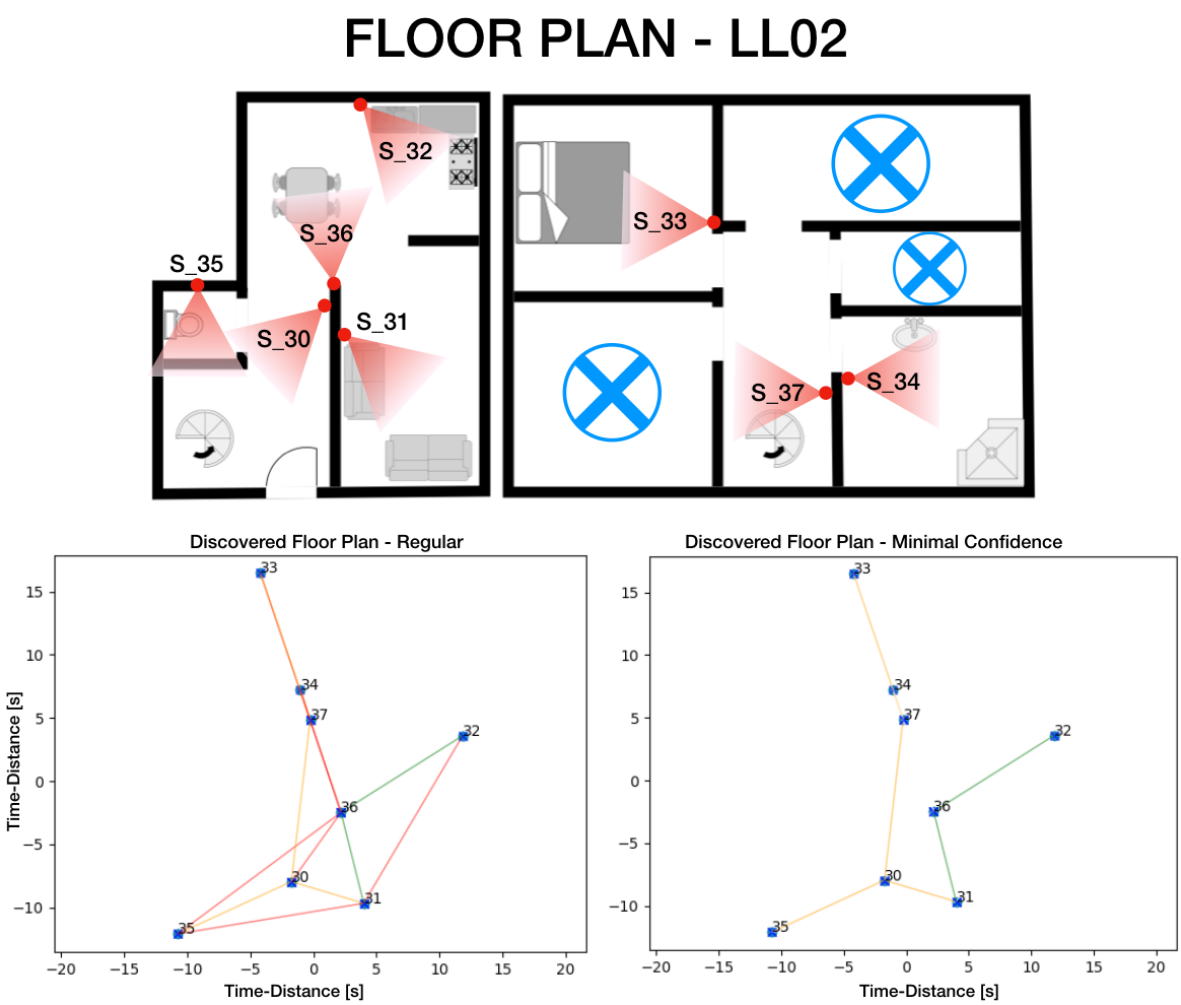


Figure 5.8: Floor plan discovery of house LL02

Test home LL02, shown in Figure 5.8, consists of a two-story house with a single bedroom in the top floor, one toilet in the top floor and a bathroom in the top floor. It is inhabited by a single elder person. There was no information indicating whether a pet lives in this house or not. 78 days of data were recorded. We remove the low confidence paths to analyze the floor plan obtained.

Similar to the results on LL01, there is a clear separation of both floors in the house. There is a single path between them, through sensors s_{30} and s_{37}.

Depending on how accurate the drawing provided by the technician was, most of the low confidence paths were correctly removed. The toilet s_{35} should not be connected to the living room s_{31} or the dining room s_{37}. However, there is a path between the living room and the kitchen s_{32}, unless the field of view from s_{36} blocks. The paths between $s_{33} \leftrightarrow s_{37}$ and $s_{30} \leftrightarrow s_{36}$ should have not been removed. A possible cause for the first path to have been removed is that the resident does not visit the bedroom on the top floor as much as the rest of the house. However, it is expected for the second path, between the front entrance and the rest of the house, to have a large number of transitions $R_{30,36}$. Therefore, possible reasons for these paths could be an inaccuracy in the drawings, improper positioning of the sensors, or the sensitivity of our chosen threshold for the R values.

In regard to the distances, most are within values that make sense according to the original floor plan. The staircase path is the longest one, taking close to 15 seconds, and the rest of the sensors are within 5 seconds of their neighbors. However, $s_{35} \leftrightarrow s_{30}$ and $s_{36} \leftrightarrow s_{32}$ appear to be more distant than what the drawing suggests.

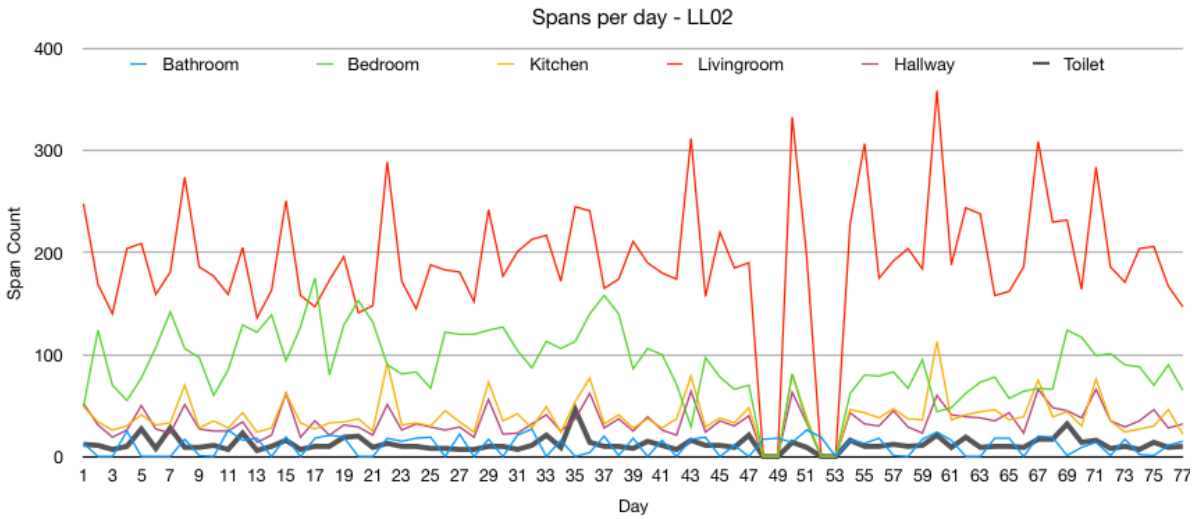


Figure 5.9: Spans recorded per day for each sensor type for house LL02

The span counts recorded for house LL02 can be seen in Figure 5.9 and Table 5.5 shows the sensors with the highest low-span count. In this home, just as in LL01, the bathroom also had the lowest span count on several days. This is expected, given that a person normally takes a shower only once a day. Unless there is a toilet in the same room as the bath, the sensor in the bathroom would not register many active spans and could therefore confuse our classifier. The way we address this problem in our methodology is with the *minimum span length filtering* step of our data pre-processing. Visits to the toilet might be more frequent than the bathroom. However visits to the bathroom are longer since this is where the person showers, grooms, and even dresses. Therefore, many of the shorter toilet visits are filtered by our pre-processing strategy, improving the performance of the classifier.

Sensor	Location	Count
S_35	Toilet	43
S_34	Bathroom	35

Table 5.5: Days recorded with the lowest span count per sensor in house LL02

Table 5.6 shows how our classifier was close to the 0.5 sensitivity guarantee threshold mentioned above. It correctly labeled S_{35} as the toilet. However, this example shows that our methodology has room for improvement in precision.

Sensitivity	Specificity
0.551	0.925

Table 5.6: Sensitivity and specificity of Toilet Heuristic Classifier in house LL01

For house LL02, the result of the calculated likelihood factors is identical to that of LL01 and can be seen in Figure 5.10. They are identical is because, on average, both users have similar patterns on the time they wake up. The chosen timeframe between 3:00 and 6:00 in the morning has a considerable impact on the results of this heuristic methodology. It was selected based on observation of the 12 datasets available, where activity in sensors outside of the bedroom would start to be registered after 6:00 in the morning. A possible weakness of this heuristic approach is that if the user regularly wakes up earlier than this, the results of the classification would select the wrong sensor.

The confusion matrix and the recognition accuracy results are shown in Figure 5.11 and Table 5.7. The average precision was of 0.88. This was mainly due to the classifier confusing the living room and the kitchen. By inspecting the floor plan, we can see that the field of view of the sensor in the kitchen reaches the living room. The classifier performed well classifying the bathroom, having a precision of 0.96. Since no sensors were labeled incorrectly, the hallway class had a precision and recall of 1.0.

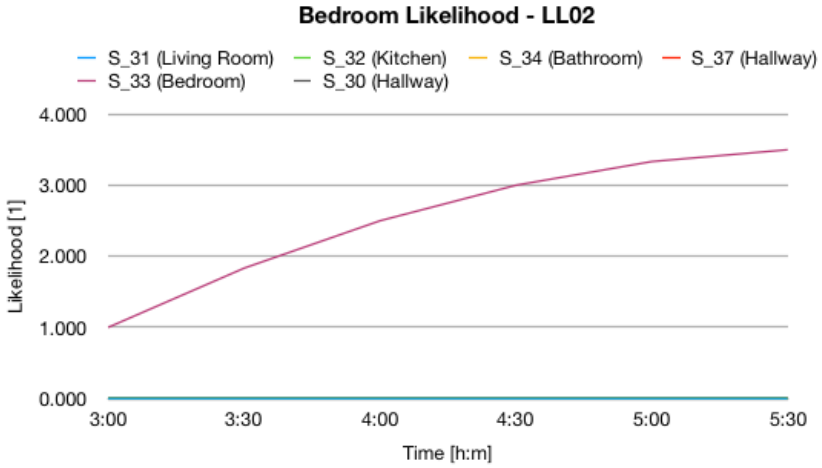


Figure 5.10: Likelihood factor of each sensor for house LL02

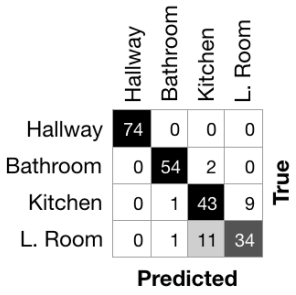


Figure 5.11: Confusion Matrix for the SVM classifier in house LL02

	Precision	Recall	Samples
Hallway	1.00	1.00	74
Bathroom	0.96	0.96	56
Kitchen	0.77	0.81	53
L. Room	0.79	0.74	46
Total	0.88	0.88	229

Table 5.7: Precision and recall for the SVM classifier in room LL02

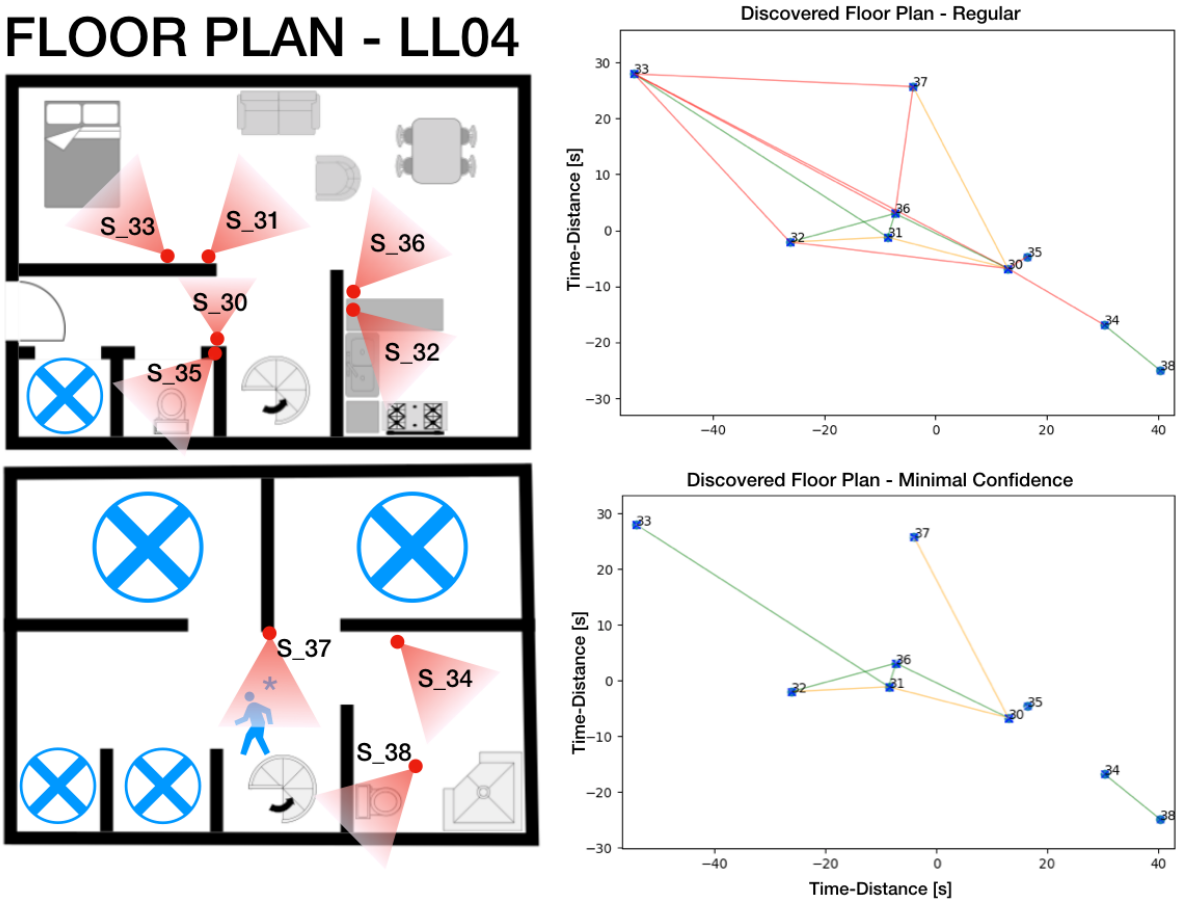


Figure 5.12: Floor plan discovery of house LL04

Test home LL04, shown in Figure 5.12, is a two-story house with an unusual arrangement compared to the others studied. The bedroom, living room, diner, toilet and kitchen on the ground floor are in the same room with no walls separating them. On the top floor the only usable room for our research is the shower with a secondary toilet. The technician provided four important notes for this house. First, the bathroom on the top floor is almost never used by the participant. Second, the stairs have a lift. Third, it is possible that a pet lives in the house. Fourth, the participant only visits the top floor for showering, assisted by a nurse. 98 days of data were recorded.

The first thing to notice is that there is no path between the top of the stairs, s_{30}, and the toilet or bathroom on the top floor, even without removing low confidence paths. By removing them, the bathroom and toilet on the top floor are completely isolated.

There is a clear indication of multiple persons or a pet in the house. A path between the kitchen and the bedroom would have crossed other 2 or 3 sensors in between. Additionally, the distance between these is even shorter than the one between s_{33} and s_{36}.

In general, the floor plan discovery algorithm did not have an acceptable performance representing the layout of the sensors in this scenario. This house was selected because it presented different challenges that are out of the scope and beyond the constraints established for this research. However, it is constructive to observe the limitations of our algorithm in the presence of multiple residents or visitors, pets, mostly-unused rooms like the bathroom on the top floor, and special contraptions like an elevator chair on the stairs.

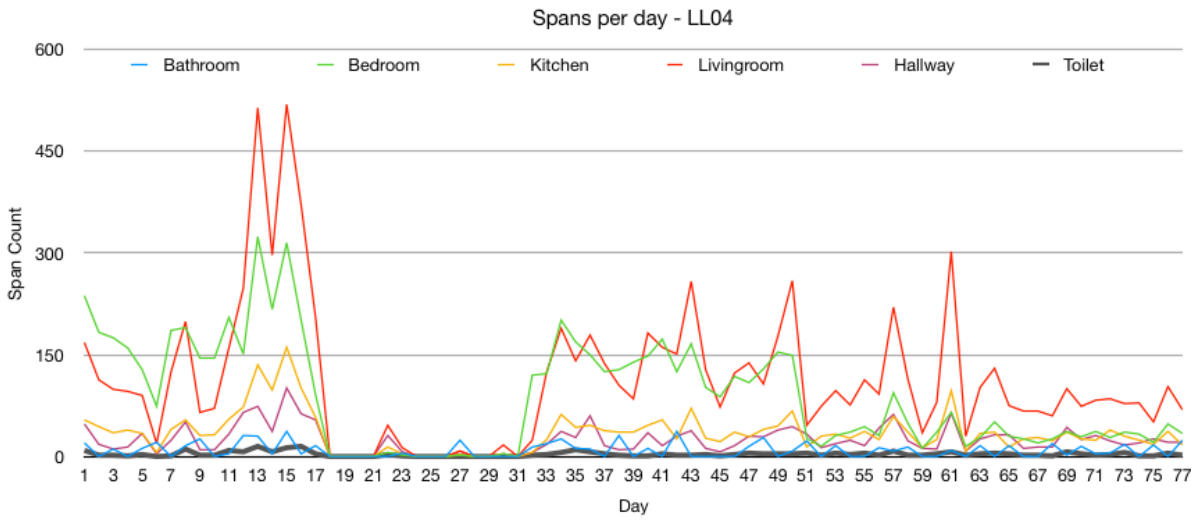


Figure 5.13: Spans recorded per day for each sensor type for house LL02

Figure 5.13 shows the span counts recorded from house LL04, and Table 5.8 shows the sensors with the highest low-span count. We can observe there is a period of days in which no activity was recorded in the house. A small count of activations in these days could be attributed to the possibility of a pet living here.

Sensor	Location	Count
S_35	Toilet	49
S_34	Bathroom	39

Table 5.8: Days recorded with the lowest span count per sensor in house LL02

In this house we were expecting for the classifier to fail in recognizing the bathroom, given the indication that the user almost never visits the second floor, where the bathroom is. Therefore, the expected outcome was for this room to be more inactive than the toilet. However, as shown in Table 5.9 the classifier had a sensitivity of 0.5 and ultimately labeled the correct toilet sensor. Under regular circumstances and according to Table 5.9, the sensitivity value should be higher since the span low-count for the toilet was higher than that of the bathroom. However, on days where there is no activity recorded (or all of it is filtered by our pre-processing procedures), our algorithm will not label any sensor as the toilet, therefore increasing the number of false negatives.

Sensitivity	Specificity
0.5	0.916

Table 5.9: Sensitivity and specificity of Toilet Heuristic Classifier in house LL01

The results of the calculated likelihood factors for this house can be seen in Figure 5.14. In this scenario we find that sensor S_{31} in the living room has, on average, more activity detected than the bedroom in the 4:30 epoch. This is not surprising, given that the bedroom is located next to the living room in this house. It is possible that the field of view of S_{31} reaches part of the bed area. Another possible cause could be the pet in the house activating this sensor. Regardless, our heuristic approach correctly identified the bedroom sensor.

Figure 5.15 and Table 5.10 show the confusion matrix and the accuracy of the results of the SVM classifier for this house. Total precision and recall were 0.59. The peculiar layout of this house could be the cause for the high rate of false positives and negatives. The recall from the living room was 0.36, possibly because this room is not separated from the bedroom. Additionally, the presence of a nurse and a pet have an impact on the accuracy of the classifier. This scenario gives an indication of how vulnerable this methodology is in the presence of more than one inhabitant, as well as the negative effect of unconventional arrangements of rooms in the house. Nevertheless, the total amount of predictions for each class was marginally sufficient and the algorithm correctly labeled the rooms.

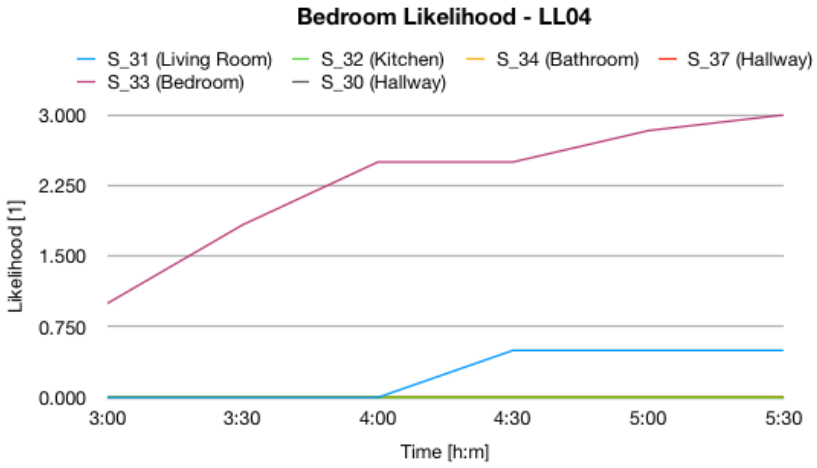


Figure 5.14: Likelihood factor of each sensor for house LL04

Figure 5.15: Confusion Matrix for the SVM classifier in house LL04

	Precision	Recall	Samples
Hallway	1.00	1.00	41
Bathroom	0.38	0.52	29
Kitchen	0.53	0.50	32
L. Room	0.48	0.36	31
Total	0.59	0.59	133

Table 5.10: Precision and recall for the SVM classifier in room LL04

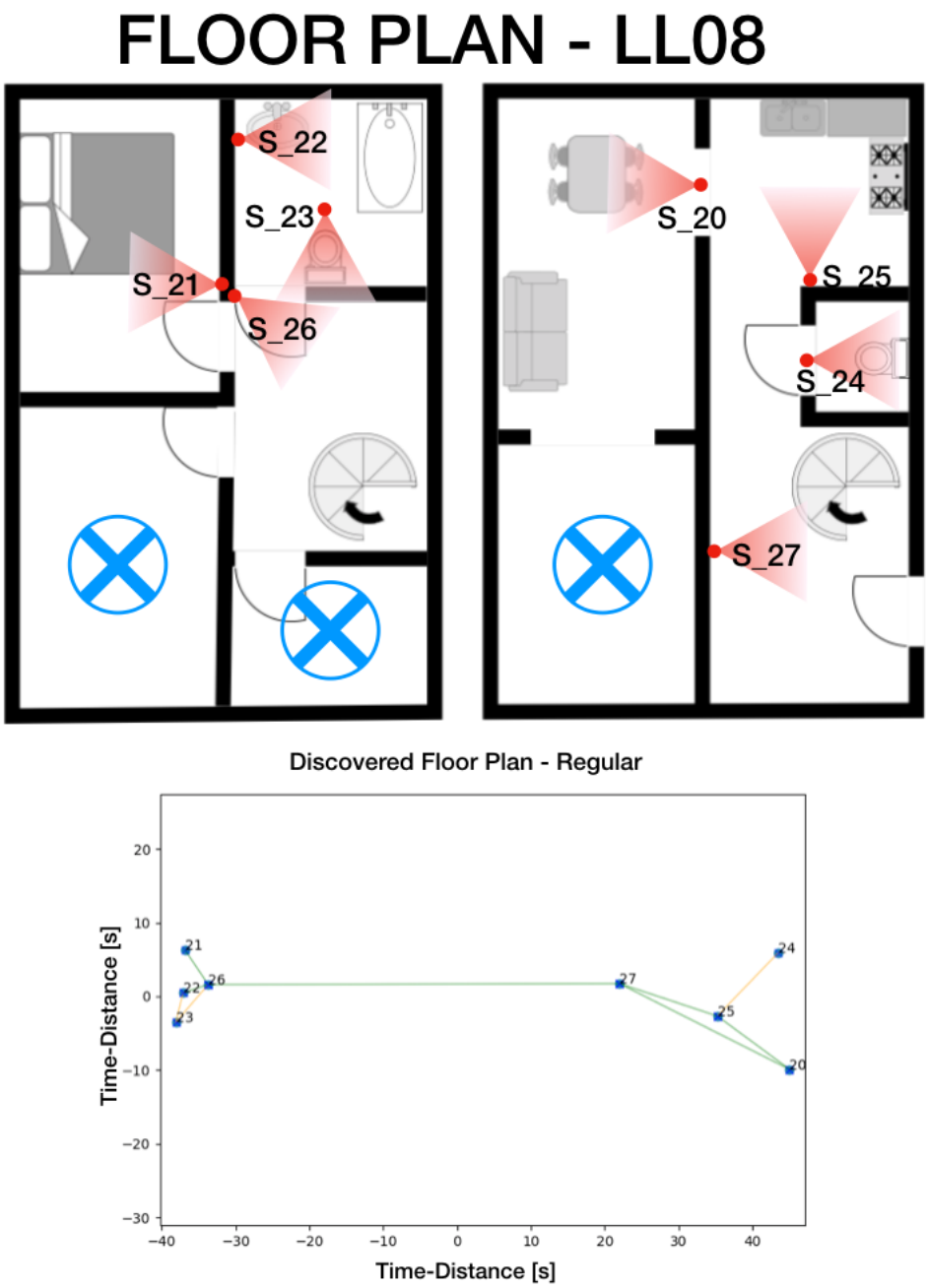


Figure 5.16: Floor plan discovery of house LL08

Test home LL08, shown in Figure 5.16, consists of a two-story house with one bedroom, bathroom, and toilet on the top floor, and one additional toilet on the bottom floor. It is inhabited by a single elder person with no pets. 83 days of data were recorded. For this scenario it was not necessary to present a minimal confidence graph since no paths were ranked as low confidence.

Most of the paths were high confidence, the graph resembles the layout of the sensors, and the distances between the nodes are within reasonable ranges. Aside from missing a path between the front entrance s_{27} and the toilet s_{24}, the floor plan discovery algorithm had excellent results with this dataset.

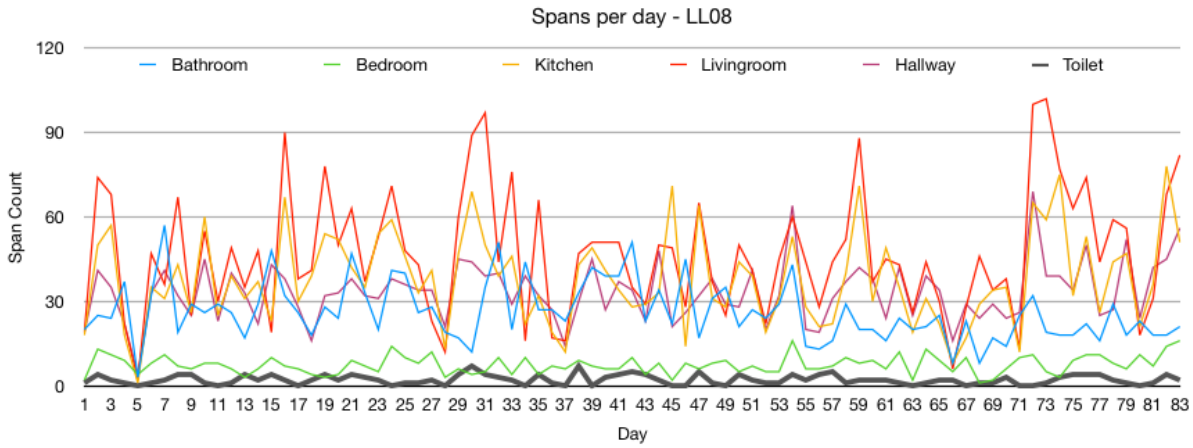


Figure 5.17: Spans recorded per day for each sensor type for house LL08

Figure 5.17 shows the span counts recorded from house LL08. It is easy to observe that the line corresponding to the span activations is clearly below the rest. This is reflected in Table 5.11, where we can appreciate a very low number of false positives. As expected, the classifier had a high sensitivity and specificity, shown in Table 5.12.

Sensor	Location	Count
S_24	Toilet	76
S_21	Bedroom	7

Table 5.11: Days recorded with the lowest span count per sensor in house LL08

Sensitivity	Specificity
0.915	0.985

Table 5.12: Sensitivity and specificity of Toilet Heuristic Classifier in house LL08

The likelihood factors calculated for this house can be seen in Figure 5.18. Here we see that the sensor S_{26} in the hallway immediately outside of the bedroom, as well as S_{22} in the bathroom have moderate activity between 4:00 and 5:00 in the morning. However, the bedroom sensor S_{22} had the highest likelihood factor and was labeled correctly.

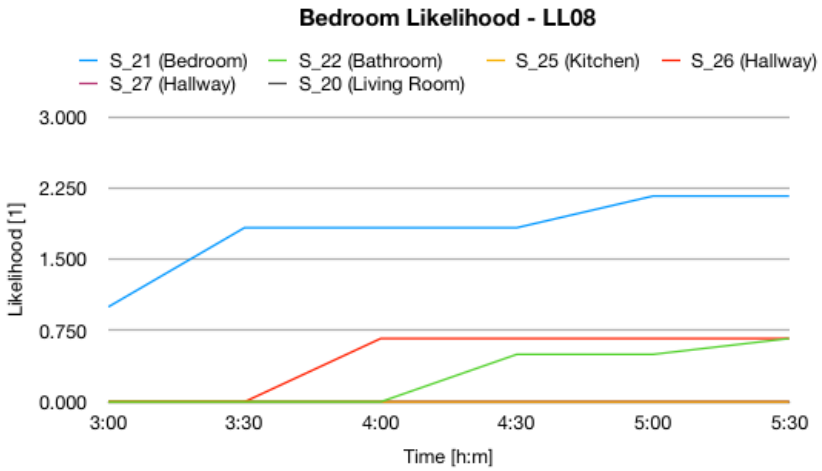


Figure 5.18: Likelihood factor of each sensor for house LL08

Given the good results achieved on this house by the floor plan discovery and heuristic methodologies,

it was expected for the SVM to perform with high accuracy as well. However, the total precision and recall were of 0.73. Similar to the situation observed in house LL02, the kitchen and the living room were difficult to recognize for the classifier. Both rooms are adjacent, and without a precise drawing indicating the exact direction the sensors are looking, it is hard to conclude if the reason for this inaccuracy is caused by improper sensor positioning. The bathroom recall was acceptable, indicating a low amount of false negatives. However, precision was only 0.76.

Further analysis on the average number of events registered per sensor, illustrated in Figure 5.20, provides evidence on the cause of the confusion between the kitchen and living room. Both classes have a very similar behavior, having only a slight difference in magnitude. A possible cause for this could be the length of the Hamming window used for smoothing the signals, which removed particular features found in them. Another possible cause is the length of the epochs. If these were shorter, additional peaks and features in the signal could help better characterize the different rooms.

Overall, the results from the SVM classifier were satisfactory for this test scenario and all rooms were correctly labeled.

Figure 5.19: Confusion Matrix for the SVM classifier in house LL08

	Precision	Recall	Samples
Hallway	1.00	1.00	29
Bathroom	0.76	0.90	21
Kitchen	0.61	0.58	19
L. Room	0.56	0.47	19
Total	0.73	0.73	88

Table 5.13: Precision and recall for the SVM classifier in room LL08

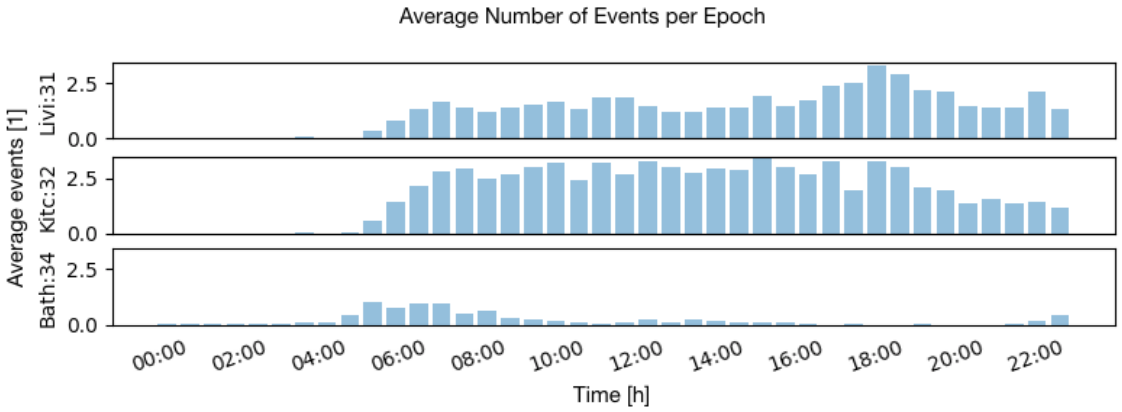


Figure 5.20: Average sensor events per epoch in house LL08

6

Conclusions and Future Work

In this chapter we wrap up our work and present proposals for future work on the remote care using IoT domain.

6.1. Conclusions

The elderly population is growing at unprecedented rates and the demand for remote care services for the elderly will increase in the coming years. Care providers need to offer innovative telehealth technologies that will not only satisfy the needs of older adults living independently at home, but will also be affordable and convenient.

Meanwhile, adoption of commercial IoT sensors and devices is increasing in the home automation industry due to their low cost, reliability and simplicity of use. These devices offer the same capabilities that technologies used in current remote health systems.

There is extensive research in the field of Activities of Daily Living recognition for remote health technologies. Most of these studies, as well as the current systems offered by care providers, rely on a set of sensors and devices prescribed, installed and configured in houses and research facilities. Therefore, the underlying network architecture connecting them to the processing server is taken for granted, and the focus of most research is on the applications that make use of the data retrieved from these sensors. Additionally, algorithms for ADL recognition require information about the configuration and distribution of these sensors throughout the house, which is often provided by means of a floor plan.

In this thesis we proposed a methodology that can make use of IoT sensors which are already installed at the houses for other purposes like home automation applications, rather than requiring clients or users to acquire and install a new set of sensors at their homes. Additionally, we proposed an automatic floor plan discovery that will provide remote care systems with the sensor configuration required and enable them to use the data to detect ADL of the user.

In order to communicate data from arbitrary IoT sensors to the remote backend, we explored different methodologies to extract the data regardless of their brand, model, and communication protocol they use. We designed a low-cost device that acts as a bridge between the gateway connecting the sensors at the home end and the remote backend in a cloud environment. We tested our methodology in the HomeLab, a test environment in Philips Research. We installed arbitrary IoT sensors from different brands, a gateway connecting them running our custom software, and our designed Access Bridge transmitting data to Philips HSDP cloud service. We demonstrated the feasibility of using arbitrary sensors for the purposes of remote health care regardless of the sensor network infrastructure to which they are connected at home.

Our contribution regarding the automatic floor plan discovery is two-fold. First, we designed an innovative methodology to automatically create a floor plan representing the distribution of the sensors throughout the house based solely on event data generated by generic binary sensors. We tested our methodology with datasets from real houses where single elderly participants live. The resulting floor plan results correctly represent the original layout of the sensors, and has acceptable results representing the distances between them. Second, we developed an algorithm to classify the rooms in which sensors are located. We used an innovative combination of heuristics and supervised learning techniques. The heuristic techniques are used to identify the toilet and bedroom of the house based on observation of human patterns. For our supervised learning

technique we chose and implemented an SVM classifier. We used activation times, dwelling time and variance in span lengths as features. Both techniques correctly label the sensors in these locations for all our datasets. However, we find that in scenarios with improper sensor positioning, low activity, visitors, and pets, our methodology can see its performance drastically reduced. Results from our classifying methodologies can be found ini Table 6.1. We demonstrated the feasibility of generating a labeled floor plan of a house using only sensor event data, which can further enable ADL recognition methodologies to acquire and process samples from sensors in unknown environments.

Heuristic Classifier		
	Sensitivity	Specificity
Toilet	0.68	0.96
Bedroom	1.00	1.00
Total	0.84	0.99

SVM Classifier		
	Precision	Recall
Bathroom	0.76	0.83
Kitchen	0.71	0.70
Living Room	0.67	0.63
Hallway	1.00	1.00
Total	0.78	0.78

Table 6.1: Room classification performance summary

The relevance of this thesis and its impact in the field of telehealth and remote monitoring services for the elderly is that we propose an alternative to expensive and care provider-specific sensor systems and infrastructure. We reduce upfront costs for the user and the care provider. We also reduce the burden at the company end training and deploying technicians to install these sensors, and at the user end by dispensing with the need to configure devices and having redundant sensors installed. We hope that this work will serve as a guideline for designing IoT-based remote care solutions of low cost so that this technology will reach those who need it most in the future.

6.2. Future work

The characterization of a house and and its rooms based on binary sensor data only is a novel and relatively unexplored domain. There is room for research and improvements. In this section we discuss a few that were observed during the process of designing this methodology and writing this thesis.

In our work, the floor plan recognition and room classification were two independent modules whose results are combined in order to fully characterize the interior of a house. However, this does not mean that they can not benefit from their individual results in order to improve their mutual performance and accuracy. In the future we consider using the output of these modules as feedback for the other. One example is the floor plan graph generated in the first step. We noticed that nodes with a single edge often represent toilets and bedrooms. Other rooms like a living rooms and kitchens often connect to more than one node. Bedrooms were often found to be connected to a bathroom, and kitchens to living rooms. The room recognition algorithm can use this information to further improve its classification accuracy. Similarly, the floor plan discovery algorithm could leverage from information generated by the room classifier. For example, by knowing the type of rooms the nodes represent, it could estimate connections between them based on probabilities, identify stairs and different floors, or recover missing paths between nodes (as in the case for house LL04 in our experiments).

Our room classifier constantly had difficulties recognizing sensors in adjacent rooms. In the future, a new methodology can be designed that improves our results when the field of view of a sensor partially and covers an unintended contiguous room.

Our classification algorithm relies on several constraints that in many scenarios would be impossible to follow. For example, our work could be extended to be robust in a multi-user situations similar to [16], or mitigate with pets and visitors like in [15].

Another research topic could be allowing for multiple sensors per key location, or multiple key locations (e.g., more than one sensor in a large living room, or more than one bedroom). Our approach is constrained to only one due to the high complexity of classifying hallways. Instead of trying to identify these, we assume any room that is not a key room to be a hallway.

Finally, our methodology cannot be considered as a "real-time" solution. It requires samples from several days before it can perform a classification of the rooms. Our work could be extended to offer a more adaptive solution that could start classifying as soon as the first day and improve its results over time. Using transfer learning techniques like those of [18] could potentially achieve this.

Acknowledgements

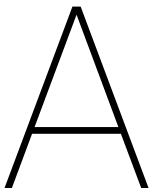
I am sincerely grateful to Warner, my mentor and supervisor at Philips Research, for he did not only give me the opportunity to explore new domains and technologies, meet new people that I now call my friends, and live in the city that I now call my second home. He also guided me with infinite patience throughout the entire project, tolerated my faults and always provided me with sound advice.

I am also grateful to VP and Nikos, my supervisors at TU Delft. I want to thank them for believing in me and this project, for helping with my doubts, answering my emails during weekends and nights on end, and for giving me the opportunity to present this work. I specially want to thank and express my admiration for VP, who continued assisting me and his other students as well as so many different projects despite unfortunate circumstances.

I also want to thank all my friends who were there all along this journey. Victoria, for the good vibes, always being there to listen to me and make me a better person. Simon, for the advice on machine learning and reminding me it is always a good time for speculoos and bouldering. Sylvia, for adopting me into the PIC family and the good times at Monk. Gaby and Gui, for always cheering me up (and cooking for me while I was locked up writing this thesis. Giorgio, Ozan and Philipp, for all the good memories, fun trips and adventures, Kristina for reminding me I still have a home in the US. Finally, Colin and Raul, for always being there (on the other side of the planet) to remind me not to take life too seriously from time to time. Without you all, I would not be who and where I am.

Finally, I'd like to thank my parents, for a lifetime is not enough to repay all the love and sacrifices, and my sisters. We may not see each other for years at times, but I always feel them close to me.

From the bottom of my heart, thank you.



Appendix A: Architectural Approaches

To address the challenge of connecting generic IoT sensors to our backend, different alternatives were explored. This appendix describes the different architectural approaches that were considered and give an indication of the advantages and drawbacks of each one.

A.1. The Generic Scenario

In a typical consumer-grade home automation environment, the typical IoT scenario is as depicted in Figure A.1. It consists of a set of sensors connected to a gateway or hub. The hub may do some local processing and send commands to the devices directly, but the most typical case is that they connect to some sort of cloud service. The remote cloud or servers process inputs from the user and sensors and produce outputs which the hub commands to the diverse actuators in the system.

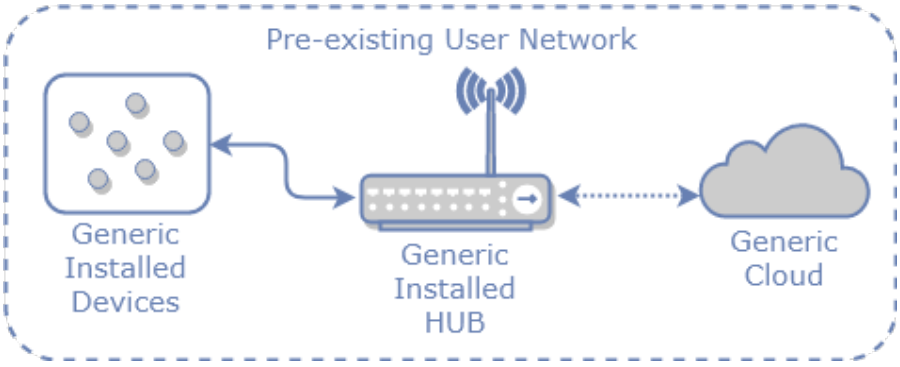


Figure A.1: System overview of a generic scenario.

In general, there are five approaches we can follow in order to acquire the data from the sensors in order to transmit it to the ADL algorithms:

1. Direct sensor data acquisition
2. Gateway bridge data acquisition
3. Gateway override
4. Cloud service data acquisition
5. Custom Hardware

These can be better explained following the diagram in Figure A.2.

In the Direct Sensor approach (1), the data is acquired immediately after it is transmitted by the sensor, and before it reaches the gateway. The options explored following this scenario are explained in section A.2.

The Gateway Bridge alternative (2), on the other hand, lets the data pass from the sensors through the gateway. It is then that the gateway will transmit the information to the backend by diverse means and depending on the brand and make. While this is the preferred solution to the problem given it involves the least interference with the existing system, it relies on the ability of the specific hub to be able to transmit this data in a way the algorithms can make use of. The options exploring this scenario are the ones described in section A.3.

The Gateway Override method (3) involves inserting a new hub to the system which will override the original as primary. Protocols like Z-Wave and Zigbee allow for hubs to take roles as primary and secondary controllers. Although the typical use for this is inserting a secondary hub with additional sensors and actuators to the network, the possibility of overriding the primary without the secondary losing its functionality is explored. This approach is employed by the scenarios in Section A.4.

Cloud Service data acquisition (4) delegates the task of sending sensor data to the cloud service the hub is connected to. Since most cloud services have options to use external services such as IFTTT or send HTTP requests based on events, the potential in this option is difficult to argue. However, this potential comes with drawbacks related to vendor-specific cloud services. These will be explored in section A.5.

Finally, the Custom Hardware alternative (5) is discussed in section A.6. This section explores the possibility of developing a device that can join an existing (Zigbee) network of sensors and devices as just another end device, and can report on any activity the motion sensors detect.

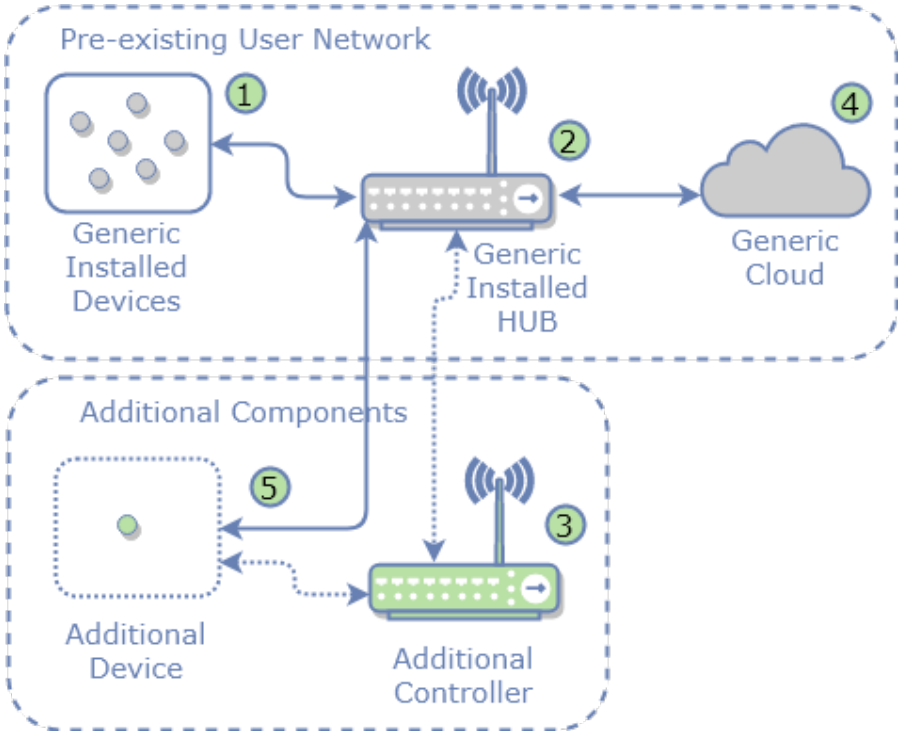


Figure A.2: Overview of all five data acquisition approaches.

A.2. Direct Sensor Data Acquisition

The direct sensor data acquisition approach consists of placing a device within range of the sensor network. This device is in practice a network sniffer with networking capabilities that would be constantly sniffing packets transmitted within the sensor mesh, decoding them and transmitting them over the network into the ADL backend. This approach focuses on Zigbee and Z-Wave sensors only. The device consists of the following elements:

1. Zigbee sniffer
2. Z-Wave USB antenna
3. Processor

Zigbee Sniffer

Zigbee sniffers are commonly used to debug industrial and home automation networks. They are available in the form of a USB stick with an antenna, along with a small processor with custom proprietary firmware and accompanying software tools. A few of these utilize the Texas Instruments CC2531 SoC (System on Chip) or the Atmel AT0259, in a circuit board integrated with an on-PCB (Printed Circuit Board) antenna. The modules investigated were the EZSync CC2531 Evaluation Module, and the Texas Instruments Wireless Connectivity Development Kit.

Setting up the tool is pretty straightforward with the drivers provided. After installing the drivers and Texas Instruments Packet Sniffer it is possible to see all the data being captured from the network in real time. However, the problem is that packets are encrypted, and acquiring the unencrypted payload is not a simple task.

Zigbee Packet Decryption

In order to fully decrypt Zigbee packets, the following parameters have to be known:

1. Source Address
2. Network Key

3. Link Key

The **source address** of the sensors must be known in order to filter the relevant messages. This can only be acquired from the gateway to which the devices connect, and it is not necessarily possible to get this information with all devices. The SmartThings Hub selected for this project is capable of displaying this information via the configuration dashboard. To this point, the capability of providing the device's Source Address on other gateways hasn't been investigated. In the scenario where the gateway would now allow for this, an automated trial and error routine would have to be set up in order to "find" all the sensors of interest in the network.

Next, in order to decrypt the packages it is necessary to acquire keys specific to the network. The Zigbee standard provides complex security measures to protect the network and key exchange. This is done at two different levels: at the Network layer, and Application Support layer.

At the network layer, a network key is used to secure broadcast communications within a mesh. This is a 128-bit key shared between all devices in the network. At any given moment, several of these keys exist but only one can be active. The specifics of how the encryption at this layer works are not relevant to this study and will therefore be omitted. However, since this project focuses only on devices following the Zigbee Home Automation Public Application Profile HAPAP (Home Automation Public Application Profile), an exploit of the security at this layer is possible.

The purpose of the Zigbee HAPAP is to allow manufacturers to develop interoperable devices that can exchange control messages in order to form a home automation network. These messages are well known and provided by the specification. Along with these standard messages, devices must implement Startup Attribute Sets (SAS). From the security perspective, two properties of the SAS are of relevance to us: the Default Trust Center Network Key, and the Default Link Key Join. Any device that intends to be compatible with home automation Zigbee networks comes pre-installed with these default keys. For the **network key**, the value is "0x5A 0x69 0x67 0x42 0x65 0x65 0x41 0x6C 0x6C 0x69 0x61 0x6E 0x63 0x65 0x30 0x39" which translates to “ZigBeeAlliance09”, and the **link key** is "0x01". The HAPAP profile has a critical security vulnerability in this aspect: Any new device joining the network will use these pre-configured keys when joining the network, and in return will be provided with the current active key. That means that if an "attacker" is able to sniff packets on the precise moment a new device joins the network, it will be able to capture the exchange of the active key of the network. Finally, since the Zigbee standard is based on an open trust model, all the layers in the stack trust each other. This implies that encryption only occurs between devices and not across layers.

Z-Wave Sniffer

Unlike Zigbee, Z-Wave is not an open standard and access to the documentation is limited to users who have purchased a (very expensive $1500 USD) development kit from Sigma and have signed an NDA. Part of this kit includes a Z-Wave USB serial interfacing device which can be used to read packets. A few other third party alternatives are available on the market like the Suphacap Z-Wave Sniffer from Suphammer. Alternatively, a Software Defined Radio could be used to decrypt raw analog GFSK samples. Unfortunately, without access to the standard specification, it is impossible to determine the viability of detecting and decrypting messages from the devices of interest in the network for this project.

Processor

Assuming Zigbee and/or Z-Wave packets can be decrypted, the device must process these samples and transmit them to our ADL backend. Since the hardware required to acquire these packets is available as USB connected devices, it makes sense to consider a small single-board computer like a Raspberry Pi or a Beaglebone running Linux. The purpose of this computer would be to acquire the samples, pipe them through an interface into a program or script, and finally transmit these over HTTP.

Advantages and Shortcomings

The biggest advantage of this method is that it would cause no interference with the existing network when installed. It would be a matter of placing the device in the vicinity of the network gateway at the patient's residence.

On the other side, this method has several drawbacks.

- It requires technical knowledge to set up. Specifically, acquiring each of the sensor's source address in the network (Zigbee). Not all gateways might support sharing this information so easily.

- Up to this point, it hasn't been possible for the author to confirm compatibility with Z-Wave
- Requires additional hardware, namely the computer and antenna devices, which imply an additional cost.
- Related to hardware, the solution described here is focused on just a proof of concept. A professional service would probably require the use of a custom built device instead of a combination of hobbyist tools.
- For the Zigbee case, the standard's security vulnerability is employed. This is not only a bad practice that could have legal consequences or give the brand a bad reputation, but also in a future adaption of the standard this "backdoor" could be fixed, rendering the system inoperable.

Given the amount and relevance of the drawbacks, the decision was not to explore this alternative further.

A.3. Gateway Bridge Data Acquisition

The focus of this approach is to relay sensor activity captured by the gateway to the backend. The principle of operation is similar, if not identical, to that from the Cloud Service Data Acquisition approach described in section A.5. The only difference is that the implementation shall not rely on any cloud service and relay the messages either to a local server, or directly to the remote backend.

Our research found that very few gateway brands allow for a direct link to a remote server. Local message relaying is possible in some, but on a very limited basis. As an example, the SmartThings hub allows local processing for a limited list of device types, such as door locks, switches and lights. This capability is only provided as a fallback alternative should there be an internet outage. However, simple tasks as sending an MQTT [47] message to an MQTT (Message Queuing Telemetry Transport) broker running on the local network or sending an HTTP request whenever a there is sensor activity is possible. This is one of the main capabilities that will be employed in our proposed solution.

Unfortunately, most of the gateways rely on cloud services to process messages in a fashion that would be useful to this project. These cloud services are either proprietary, or provided by third parties like IFTTT [52]. The only exceptions to this are the open source home automation software gateways like Home Assistant. Since they are software solutions running on a (unix based) local machine, it is possible to develop scripts that would relay the sensor activity either to a local or remote backend.

Advantages and Shortcomings

The biggest advantage of using this method to retrieve the sensor data is that no additional hardware would be required in case of sending the messages to a remote server, or a very minimal setup would be required to run locally, like a very low cost single board computer. Another advantage is that this setup doesn't rely on proprietary or third party cloud services. This translates to a more reliable, controlled and safe management of the data.

Unfortunately, this approach comes with several drawbacks. Most of the commercial gateways have limited to no local processing capabilities, and they lack the ability to send custom packets over the internet to a remote server. This would limit the solution to merely the open source software gateways like Home Assistant, which are not popular among the average home automation user but rather only among enthusiasts and tech savvy users. However, in combination and as compliment of the Cloud Service Data Acquisition method described in section A.5, this method provides the best results in terms of latency and security whenever the gateway allows local access.

A.4. Gateway Override

The concept behind this approach is partially based on the findings from the Gateway Bridge Data Acquisition described in section A.3. Gateways that can run on a unix environment, such as Home Assistant, Domoticz or Openhab [53], open a whole new set of possibilities in the way that data that reaches the hub can be processed. By means of relatively simple scripts, it would be possible to route the sensor messages to the backend. However, one of the purposes of the project is to reuse an existing network, rather than replacing it even by using pre-existing sensors. It is therefore interesting to explore an alternative that keeps the network untouched while at the same time gain the benefits of having one of these gateways included in the mesh. This approach focuses on introducing a second gateway controller into the network that will take over control

of the devices, while keeping the original gateway and network untouched and functional without the need to reconfigure the devices. This solution assumes a network of devices communicating over Zigbee and Z-Wave only.

Z-Wave certified radio controllers have the ability to act as secondary controllers. This can be achieved by having the original controller to join the primary controller's network. In our case, the primary is a Raspberry Pi running Home Assistant with a USB Aeotec Z-Wave antenna, and the secondary (original) controller a Samsung SmartThings Hub. In theory, once the secondary controller joins the primary's network, the latter would be able to read status changes and control devices on the secondary's network. However, unfortunately this functionality is vendor specific and turned out to not always be the case.

In our experiment, we paired a few motion sensors and window/door Z-Wave sensors to the SmartThings Hub, and then had this one join the Raspberry Pi's network. The network merging was successful and we were able to see the devices in the Home Assistant Dashboard. However, that's as far as it was possible to get by using this setup. The SmartThings Hub, in order to adhere to the Z-Wave Alliance standard, provides the functionality to join another network. However, it does not provide the functionality to relay status update on the devices to the primary controller, and neither would it allow the primary to directly control them. Some research on other devices proved that this is the case on other brands as well, like the Wink 2.

Other issues arise with this method. While the secondary controller keeps it's original functionality (for example, turning on a light when motion is detected) when it joins another network, new devices cannot join the network anymore. In order to do this, the secondary controller has to leave the primary. This automatically dismantles the network so all devices have to be added again. Then, the new device can be added, and finally the controller can rejoin the primary network. This is because hubs like this are designed to be primary controllers only. Another issue is that Zigbee devices present in the secondary network will not be visible to the primary controller and therefore can't be used through conventional methods.

The only viable alternative is to communicate both controllers via methods other than direct Z-Wave communication. For example, this could be done through IFTTT, or through an MQTT broker. This would allow to receive updates from all devices, Zigbee, Z-Wave, and even other technologies like BLE. However, this brings us back to the original issue from the Gateway Bridge Data Acquisition approach described in section A.3, which is relying on cloud services. Also, this would render the secondary controller useless since there's no need for it anymore given that the data can be transmitted directly from the cloud to the backend.

Advantages and Shortcomings

Had this method worked as envisioned, the advantage would have been a simple and non-intrusive way to gather data from the sensors while keeping the network intact and functional. While some gateways properly support acting as secondary controllers, we want to build a generic solution that will work in most scenarios.

The main disadvantage of this method is that one of the standard functionalities in the Z-Wave standard, the so called "primary controller shift", is optional. Not all vendors support it, and this applies for two of the biggest brands in the consumer-level gateways. Another big disadvantage is the inconvenience of having to reconfigure all devices each time a new one is added to the network. This renders this solution completely unacceptable and therefore it is not explored further.

A.5. Cloud Service Data Acquisition

Given the limitations from the previous approaches, we are left with the alternative of abstracting away from vendor specific technologies and devices and instead opt for a mechanism that relies on cloud services. Based on the assumption that most, if not all, of the commercial home automation solutions are oriented towards having the user monitor and control their devices remotely through their smartphones and computers over the internet, it would be safe to presume that an architecture based on web services would be applicable to the largest amount of possible scenarios.

As depicted in Figure A.1 in section A.1, home automation gateways commonly connect to some cloud service in order to provide both services and remote access to the user. Most of them also utilize the same cloud service to configure the behavior of the devices in the network. Furthermore, all these cloud services allow the user to configure interactions with third party clouds or web services via HTTP requests. This can be either via *applets* that the user can either program or install from an application "store", or via a third party IoT bridging service, like IFTTT. All the gateway brands described in section 3.3.3, even the multi-protocol ones like Apple HomeKit, Google Home and Amazon Echo, have the possibility to do this via their own *applets*, IFTTT, or both. Figure A.3 depicts both possibilities:

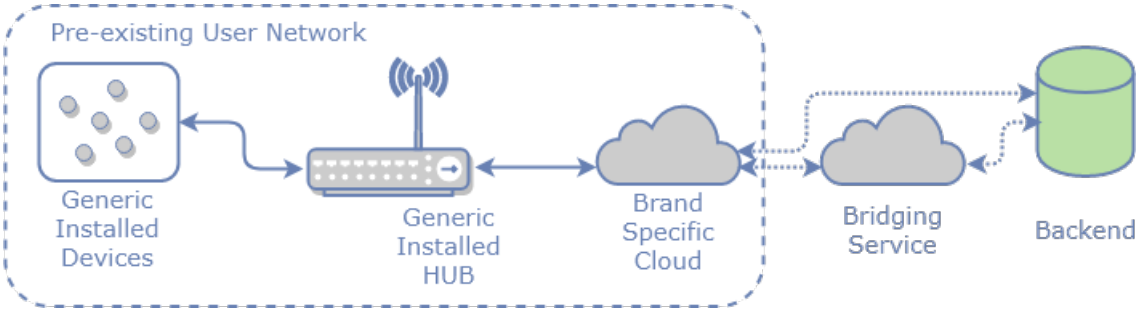


Figure A.3: Cloud Service Approach

Access via Applets

Applets are a mechanism used by different brands to allow the user to configure their home automation environment to their liking and necessities. They can range from very simple script-like routines such as "At X time of the day, turn on Y set of lights", to more complex use cases such as "If my smartphone is within X meters from my home AND there's nobody home, open the garage doors, turn on Y 'mood' setting, adjust the temperature and send a text message to Z number letting them know I'm home". These can be downloaded to the gateway via simple online interfaces, they are configurable and require very little technical knowledge from the user. Moreover, it is possible to write applets and upload them to the "market", so that other users can benefit from them as well. This is a very powerful tool, because we could write an applet that, once the user configures it with populating fields for all the sensors in their home, would send the relevant information and status updates to our backend.

The intended approach would be an applet that gathers sensor data, creates a custom JSON (JavaScript Object Notation) message [54], and transmits it over HTTP to the backend, where it can be processed and then used by the ADL algorithms.

Access via Bridging Services

Bridging services, such as IFTTT, work in a similar fashion to applets. They are services where the user creates an account, connects their devices, selects an applet from the extensive catalog, does minimal configuration and the server takes care of the rest. The only prerequisite is for the company that develops the gateway to have an arrangement with IFTTT, which is the case for all major consumer gateways. Even for highly restricted environments like Apple Homekit there exist bridging applets that allow the user to monitor and control their devices via this service. Like the case with vendor specific applets, it is possible to create custom IFTTT applets to suit the user's needs and publish them to the catalog.

Backend Side

Once the data is able to be published either via applets or bridging services over HTTP, it is possible to retrieve them for our purposes from either a cloud service or our own server. For the purpose of this assignment, the latter is chosen.

Data Retrieval from Cloud Services

The Philips HealthSuite Digital platform (HSDP) [40] was built with the purpose to address modern healthcare technological challenges. HSDP is built on top of Amazon Web Services AWS (Amazon Web Services), therefore it offers a broad range of solutions including those related to IoT. This means that accessing tools such as the AWS IoT Core is possible via HSDP. The most relevant AWS tool for the purposes of this assignment is an MQTT broker service.

MQTT messages can be sent to HSDP via HTTP (Hypertext Transfer Protocol) from an applet. On the backend side, a subscription to particular channels, or "topics" on this MQTT broker can be issued, which would effectively update it virtually in real time of any activity published by the sensors. Using AWS and HSDP guarantees scalability and reliability to this solution.

As an additional note, Philips has a strategic partnership with Samsung in order to integrate their ARTIK cloud services to HSDP [55]. ARTIK, just like AWS IoT, provides IoT solutions and tools relevant to this project including an MQTT broker.

Data Retrieval from a Local Server
An alternative to using HSDP and cloud services is setting up a server running an MQTT broker locally on a Raspberry Pi. As simple as this approach may seem, it is a powerful demonstration of how a full-scale server could be set up in order to subscribe and listen to MQTT messages being published from the applet. It is only a matter of setting up a web server listening to incoming HTTP requests containing sensor data in JSON format that can then be translated and sent as MQTT messages to the broker. An initial implementation of our IoT solution followed this approach, using the open source tool Node Red [45] to handle HTTP requests, before moving on to the HSDP solution described in the previous section.

Advantages and Shortcomings

The obvious advantages of this approach are its compatibility, flexibility, scalability, reliability, security and user friendly mechanics.

- All brands surveyed in this study are **compatible** with this approach, regardless of the wireless communication protocol they use.
- It is **flexible** in the sense that the applets are hosted remotely, which means that if it were necessary to update them it would be as simple as deploying a new version to the catalog. Also, it is flexible in the sense that once the sensor data is available via HTTP, it is possible to do virtually anything with it without limitations. It can be parsed, published and interpreted in any desired manner.
- Since the services are hosted in robust servers provided by the cloud companies, the data transmission is **scalable**, **reliable** and **secure**.
- It requires minimal user interaction and technical knowledge to set up.
- It does not involve any additional hardware to that which is already present at the user's home.

Given the extensive list of advantages, this approach is further explored and developed. The specifics to our implementation can be found in section 4.1.1

A.6. Custom Hardware

The concept in this solution is to develop a Z-Wave or Zigbee Home Automation Profile compliant custom device, including hardware and software design. The device would present itself to the gateway as a regular device type from the HAPAP profile. For example, an RGB light. Since home automation gateways can be configured to act upon events from the sensors, whenever activity from them was detected, the gateway could send a masked message to the custom device. Upon receiving such command, the device could send over WiFi a message to our remote server or to HSDP, for example.

This alternative was promptly dropped given the costs and time involved in developing and manufacturing hardware, as well as the costs involved to acquire Zigbee certification or the SDK for Z-Wave development.

A.7. Approach comparison

Table A.1 summarizes the advantages and disadvantages of each of the different approaches described in this chapter.

Approach	Interference w/existing setup	Compatibility	Complexity	Cost	Additional HW
Direct Sensor	✓	✗	✗	✗	✗
Gateway Bridge	✓	✗	✓	✓	✓
Gateway Override	✗	✗	✗	✗	✗
Cloud Service	✓	✓	✓	-	✗
Custom Hardware	-	✓	✗	✗	✓

Table A.1: Feature comparison of architectural approaches

Bibliography

[1] He Wan and Daniel Goodkind. An aging world: 2015. *U.S. Government Publishing Office*, 2016.

[2] Statista. *Smart Home Market Penetration in UK*, 2018 (accessed October 1, 2018).

[3] Sergio Soto and Warner ten Kate. Using iot sensors for monitoring in telecare. (R-TN-2018/00114), 2018.

[4] Statistics New Zealand. New zealand's 65+ population: A statistical volume. *Wellington: Statistics New Zealand*, 2007.

[5] Hyojeong Shin, Yohan Chon, and Hojung Cha. Unsupervised construction of an indoor floor plan using a smartphone. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 42(6):889–898, 2012.

[6] Michael Hardegger, Daniel Roggen, Sinziana Mazilu, and Gerhard Tröster. Actionslam: Using location-related actions as landmarks in pedestrian slam. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, pages 1–10. IEEE, 2012.

[7] Damian Philipp, Patrick Baier, Christoph Dibak, Frank Durr, Kurt Rothermel, Susanne Becker, Michael Peter, and Dieter Fritsch. Mapgenie: Grammar-enhanced indoor map construction from crowd-sourced data. In *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 139–147. IEEE, 2014.

[8] Moustafa Alzantot and Moustafa Youssef. Crowdinside: automatic construction of indoor floorplans. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 99–108. ACM, 2012.

[9] Beth Logan and Jennifer Healey. Sensors to detect the activities of daily living. In *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pages 5362–5365. IEEE, 2006.

[10] Kinh Tieu and Paul Viola. Boosting image retrieval. *International Journal of Computer Vision*, 56(1-2):17–36, 2004.

[11] Anthony Fleury, Michel Vacher, Hubert Glasson, J-F Serignat, and Norbert Noury. Data fusion in health smart home: Preliminary individual evaluation of two families of sensors. In *ISG'08*, page 135, 2008.

[12] Anthony Fleury, Norbert Noury, and Michel Vacher. Supervised classification of activities of daily living in health smart homes using svm. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 6099–6102. IEEE, 2009.

[13] Debraj Basu, Giovanni Moretti, Gourab Sen Gupta, and Stephen Marsland. Wireless sensor network based smart home: Sensor selection, deployment and monitoring. In *Sensors Applications Symposium (SAS), 2013 IEEE*, pages 49–54. IEEE, 2013.

[14] Haitao Wu, Wei Pan, Xingyu Xiong, and Suxia Xu. Human activity recognition based on the combined svm&hmm. In *Information and Automation (ICIA), 2014 IEEE International Conference on*, pages 219–224. IEEE, 2014.

[15] Johanna Petersen, Nicole Larimer, Jeffrey A Kaye, Misha Pavel, and Tamara L Hayes. Svm to detect the presence of visitors in a smart home environment. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 5850–5853. IEEE, 2012.

[16] Nirmalya Roy, Archan Misra, and Diane Cook. Ambient and smartphone sensor assisted adl recognition in multi-inhabitant smart environments. *Journal of ambient intelligence and humanized computing*, 7(1):1–19, 2016.

[17] Kenki Ueda, Morihiko Tamai, and Keiichi Yasumoto. A method for recognizing living activities in homes using positioning sensor and power meters. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*, pages 354–359. IEEE, 2015.

[18] Tim van Kasteren, Gwenn Englebienne, Ben JA Kröse, et al. Recognizing activities in multiple contexts using transfer learning. In *AAAI Fall Symposium: AI in Eldercare: New Solutions to Old Problems*, pages 142–149, 2008.

[19] Tobias Nef, Prabitha Urwyler, Marcel Büchler, Ioannis Tarnanas, Reto Stucki, Dario Cazzoli, René Müri, and Urs Mosimann. Evaluation of three state-of-the-art classifiers for recognition of activities of daily living from smart home ambient data. *Sensors*, 15(5):11725–11740, 2015.

[20] Nathavuth Kitbutrawat, Hirozumi Yamaguchi, and Teruo Higashino. Localization of binary motion sensors in house. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International*, pages 1132–1137. IEEE, 2017.

[21] Ifat Afrin Emi and John A Stankovic. Sarrima: smart adl recognizer and resident identifier in multi-resident accommodations. In *Proceedings of the conference on Wireless Health*, page 4. ACM, 2015.

[22] Enamul Hoque and John Stankovic. Aalo: Activity recognition in smart homes using active learning in the presence of overlapped activities. In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2012 6th International Conference on*, pages 139–146. IEEE, 2012.

[23] Marjorie Skubic, Rainer Dane Guevara, and Marilyn Rantz. Automated health alerts using in-home sensor data for embedded health assessment. *IEEE journal of translational engineering in health and medicine*, 3:1–11, 2015.

[24] Dan Yang, Weihua Sheng, and Ruili Zeng. Indoor human localization using pir sensors and accessibility map. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE International Conference on*, pages 577–581. IEEE, 2015.

[25] Sidney Katz and C Amechi Akpom. A measure of primary sociobiological functions. *International journal of health services*, 6(3):493–508, 1976.

[26] A. Fuentes-García. Katz activities of daily living scale. *Encyclopedia of Quality of Life and Well-Being Research*, pages 3465–3468, 2014.

[27] Marie Chan, Daniel Estève, Christophe Escriba, and Eric Campo. A review of smart homes—present state and future challenges. *Computer methods and programs in biomedicine*, 91(1):55–81, 2008.

[28] C Nugent and JC Augusta. A logical approach to adl recognition for alzheimer's patients. In *Smart Homes and Beyond: ICOST 2006: 4th International Conference on Smart Homes and Health Telematics*, volume 19, page 122. IOS Press, 2006.

[29] Prafulla Dawadi, Diane Cook, Carolyn Parsey, Maureen Schmitter-Edgecombe, and Miya Schneider. An approach to cognitive assessment in smart home. In *Proceedings of the 2011 workshop on Data mining for medicine and healthcare*, pages 56–59. ACM, 2011.

[30] Gilles Virone, Majd Alwan, Siddharth Dalal, Steven W Kell, Beverely Turner, John A Stankovic, and Robin Felder. Behavioral patterns of older adults in assisted living. *IEEE Transactions on Information Technology in Biomedicine*, 12(3):387–398, 2008.

[31] Wikipedia: The free encyclopedia. *OSI Model*, 2018. `https://en.wikipedia.org/wiki/OSI_model` [Accessed: 2018-07-02].

[32] Postscapes. *IoT Standards and Protocols*, 2018. `https://www.postscapes.com/internet-of-things-protocols/` [Accessed: 2018-07-02].

[33] Koninklijke Philips Electronics N.V. *Zigbee 3.0 support in Hue ecosystem*, 2017. `https://developers.meethue.com/zigbee3` [Accessed: 2018-07-02].

[34] Mike Prospero. *Best Smart Home Hubs of 2018*, 2018. `https://www.tomsguide.com/us/best-smart-home-hubs,review-3200.html` [Accessed: 2018-07-02].

[35] Samsung Smartthings. *Smartthings Hub*, 2018. https://www.smartthings.com/products/smartthings-hubl [Accessed: 2018-07-02].

[36] Samsung Smartthings. *Wink Hub 2*, 2018. https://www.wink.com/products/wink-hub-2/ [Accessed: 2018-07-02].

[37] Vera. *Vera: Smarter Home Control*, 2018. http://getvera.com [Accessed: 2018-07-02].

[38] Logitech. *Harmony Remote Controls*, 2018. https://www.logitech.com/en-us/harmony-universal-remotes [Accessed: 2018-07-02].

[39] Home Assistant. *Hass.io: The Ultimate Home Automation Hub*, 2018. https://www.home-assistant.io [Accessed: 2018-07-02].

[40] Koninklijke Philips N.V. *Philips HealthSuite Digital Platform*, 2018. https://www.usa.philips.com/healthcare/innovation/about-health-suite [Accessed: 2018-07-02].

[41] Koninklijke Philips N.V. *Telemetry Data Repository API Overview*, 2018. https://www.hsdp.io/documentation/telemetry-data-repository [Accessed: 2018-07-02].

[42] Raspberry Pi Foundation. *Raspberry Pi 3 Model B+*, 2018. https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/ [Accessed: 2018-07-02].

[43] Dataplicity. *Dataplicity*, 2018. http://www.dataplicity.com [Accessed: 2018-07-02].

[44] Apache Software Foundation. *The Apache Web Server Project*, 2018. https://httpd.apache.org [Accessed: 2018-07-03].

[45] Node Red. *Flow-based programming for the Internet of Things*, 2018. https://nodered.org [Accessed: 2018-07-02].

[46] Eclipse Mosquitto. *Eclipse Mosquitto*, 2018. http://mosquitto.org [Accessed: 2018-07-02].

[47] mqtt.org. *MQTT: Frequently Asked Questions*, 2018. http://mqtt.org/faq [Accessed: 2018-07-02].

[48] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, Mar 1964.

[49] Scikit-Learn. *Scikit-Learn: Machine Learning in Python*, 2018. http://scikit-learn.org/stable/ [Accessed: 2018-10-02].

[50] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[51] Fibaro. *Fibaro Home Automation*, 2018. https://www.fibaro.com/us/ [Accessed: 2018-07-02].

[52] IFTTT. *Discover IFTTT*, 2018. https://ifttt.com/about [Accessed: 2018-07-02].

[53] OpenHAB. *OpenHAB: Empowering the Smart Home*, 2018. https://www.openhab.org [Accessed: 2018-07-02].

[54] ECMA International. *The JSON Data Interchange Syntax*. Standard ECMA-404, 2017.

[55] Samsung Newsroom Daniel Yoo. *Philips and Samsung Team Up to Expand the Connected Health Ecosystem*, 2018. https://news.samsung.com/us/philips-samsung-connected-health-ecosystem-artik/ [Accessed: 2018-07-02].