



**MALMÖ UNIVERSITY**  
SCHOOL OF TECHNOLOGY

Faculty of Technology and Society  
Computer Engineering

**Bachelor thesis**  
**180 credits**

Hosting a building management system on a  
smart network camera:  
On the development of an IoT system

Användning av en nätverkskamera som värd för ett system för  
fastighetsautomation:  
Om utvecklingen av ett IoT-system

Alf Stenbrunn  
Theodor Lindquist

Exam: Bachelor of Science in Engineering  
Subject area: Computer Engineering  
Date of final seminar: 03-06-2015

Examiner: Mia Persson  
Supervisor: Ulrik Eklund



## Abstract

The Internet of Things (IoT) is an umbrella term for smart things connected to the Internet. Connected sensors may be used to the benefit of smart building management systems.

This thesis describes the development of a sensor based building management system prototype, lightweight enough to run on a single network camera. The focus of the research was investigating if the system prototype was scalable, and capable of storing and analyzing data gathered from a large amount of sensors relevant to the field of building management. The prototype was developed through a five-stage systems development process, and evaluated using simulations and case studies.

The finished prototype was able to gather and store data from a few hundred real-time sensors using limited hardware. Tests showed that the network camera should be capable of managing at least 100 sensors. The system itself is scalable with the use of more powerful hardware. However, using a distributed architecture would be preferable if more sensors are required. This could be achieved by creating a distributed network of cameras, where each camera manages its own set of sensors. This could both increase scalability and make the system more robust and reliable.

Keywords: Internet of Things, IoT, Sensors, Multiple sensor device, Network camera, Building Management, Distributed systems



## Acknowledgements

First of all, we would like to thank everyone involved with the CoSIS project for making this thesis possible, and providing invaluable feedback and input. In particular, we want to express our gratitude to our supervisor, Ulrik Eklund, and CoSIS coordinator Jan Persson. Finally, we would like to thank CoSIS partners Axis Communications AB and Sigma Technology for providing us with equipment.



# Contents

<b>Definitions and abbreviations</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Research aim and research questions . . . . .	2
1.2.1 Research aim . . . . .	2
1.2.2 Research question . . . . .	2
1.3 Scope and limiting factors . . . . .	3
<b>2 Theoretical background</b>	<b>4</b>
2.1 Internet of Things . . . . .	4
2.1.1 What is the Internet of Things? . . . . .	4
2.1.2 Performance and scalability . . . . .	4
2.1.3 IoT architecture and design . . . . .	5
2.1.4 Security and privacy concerns . . . . .	6
2.1.5 Software development and design patterns . . . . .	6
2.2 Benefits of smart building management . . . . .	7
2.3 Existing methods for counting people . . . . .	7
2.3.1 IR beam counters . . . . .	7
2.3.2 Cross line people detection . . . . .	7
2.3.3 Group estimation with multiple cameras . . . . .	7
2.3.4 Radio irregularities in IoT . . . . .	8
2.3.5 Wi-Fi counting . . . . .	8
<b>3 Related work</b>	<b>9</b>
3.1 Kuutti et al. - Real time building zone occupancy detection and activity visualization utilizing a visitor counting sensor network . . . . .	9
3.2 Mysen et al. - Occupancy density and benefits of demand-controlled venti- lation in Norwegian primary schools . . . . .	9
3.3 Musa, Eriksson - Tracking Unmodified Smartphones Using Wi-Fi Monitors .	10
3.4 Liu, Mu - An efficient algorithm for fault tolerance in multisensor networks	11
3.5 Bin et al. - Research on data mining models for the internet of things . . .	11
3.6 Wahl, Milenkovic, Amft - A Distributed PIR-based Approach for Estimating People Count in Office Environments . . . . .	12
3.7 Kamilaris, A. and Pitsillides, A. - Towards interoperable and sustainable smart homes . . . . .	13
<b>4 Method</b>	<b>14</b>
4.1 Step 1: Design and prototype a building management system . . . . .	14
4.1.1 Construct a conceptual framework . . . . .	14
4.1.2 Develop a systems architecture . . . . .	14
4.1.3 Analyze and design the system . . . . .	15
4.1.4 Build the system . . . . .	15
4.1.5 Observe and evaluate the system . . . . .	15

4.2	Step 2: Investigate the possibilities of using the network camera as the host of the system . . . . .	16
<b>5</b>	<b>Results</b>	<b>17</b>
5.1	System prototype . . . . .	17
5.1.1	Sensor selection . . . . .	17
5.1.2	Requirement specification . . . . .	17
5.1.3	System architecture . . . . .	18
5.1.4	Subsystem: Integration platform . . . . .	20
5.1.5	Subsystem: Network camera . . . . .	21
5.1.6	Subsystem: Multiple Sensor Device . . . . .	21
5.1.7	Subsystem: Wi-Fi monitor . . . . .	22
5.1.8	Subsystem: Database . . . . .	23
5.1.9	Stability tests . . . . .	25
5.1.10	Simulations and stress test . . . . .	25
5.2	Case studies . . . . .	26
5.2.1	Case study in a controlled environment . . . . .	26
5.2.2	Case study in a real-life environment . . . . .	26
5.3	Using a network camera as the system host . . . . .	27
<b>6</b>	<b>Discussion</b>	<b>29</b>
6.1	Discussion of test results . . . . .	29
6.1.1	Stability test . . . . .	29
6.1.2	System scalability . . . . .	29
6.1.3	Case studies . . . . .	29
6.2	Does the prototype meet the system requirements? . . . . .	31
6.3	Can the system run from a network camera? . . . . .	31
6.4	Vertical and horizontal scaling . . . . .	32
<b>7</b>	<b>Conclusions</b>	<b>33</b>
7.1	Process . . . . .	33
7.2	Findings . . . . .	33
7.3	Further work . . . . .	33
7.4	Contributions of this thesis . . . . .	34
<b>A</b>	<b>Wi-Fi monitor graphs</b>	<b>39</b>
<b>B</b>	<b>Simulation graphs</b>	<b>41</b>
<b>C</b>	<b>Case study graphs</b>	<b>45</b>

## Abbreviations and definitions

**Axis:** The company Axis Communications.

**BLE:** Bluetooth Low Energy, also called Bluetooth Smart. BLE is a low energy variant of the Bluetooth Standard used for wireless radio communication [1].

**CoSIS:** Cooperative, Self-Aware and Intelligent Surveillance Systems. A project at IOTAP focusing on the design of Intelligent Surveillance systems using connected sensors and cameras [2].

**DBMS:** Database Management System. A collection of interrelated data and programs used to access and manage the data collection. A DBMS perform various functions needed to ensure data integrity, data consistency and data security [3].

**MSD:** Multi-sensor device: A device with several different built-in sensors, such as temperature, humidity, or light intensity sensors.

**IoT:** Internet of Things. A term for how machines, devices, vehicles, household appliances, clothes and other objects are outfitted with sensors and processors, with the ability to connect to networks and share collected data. The data can be used for analysis and control of systems [4].

**IOTAP:** Internet of Things and People. A research center at Malmö University researching ways to utilize the Internet of Things [5].

**PIR:** Passive infrared (sensor). Used in motion detectors [6].

**Sigma:** The companies Sigma Connectivity and Sigma Technology.

# 1 Introduction

The "Internet of Things" (IoT) has been described as the next technological revolution [7, 8]. Numerous products and solutions are emerging that could be described as following the concept of IoT, such as autonomous cars, automated surveillance systems, smart houses, and even smart cities [9, 10].

This chapter explains the background of this thesis, its research aim and research questions as well as also its scope and limiting factors.

## 1.1 Background

Internet access has become significantly cheaper in the last couple of years, and the number of Internet users are steadily rising. Broadband connections are common, and smart phones can connect to the Internet as long as the user has access to a Wi-Fi connection or a 3G/4G data plan. However, not only computers, smart phones and tablets are connected nowadays. Companies have started to outfit all kinds of machines and devices with a connection to the Internet or local networks. These connected things include but are not limited to: printers, vehicles, household appliances, lights, cameras and clothes. The company Ericsson estimates that by the year 2020 there will be over 50 billion Internet-connected devices across the globe [11].

Of course, an Internet connection does not necessarily provide any value on its own. A connected device is also equipped with at least one sensor used to collect necessary data, and a processor. By collecting and processing data, the device can make intelligent decisions based on the data, or simply share the collected data through the network [4]. For example, by installing a connected thermostat and heater in a building it is possible to continuously monitor and automatically adjust the temperature of the house, likely lowering energy costs [12].

One name for this concept of connected devices is the "Internet of Things" (IoT) [7, 8]. Collected and shared data can be used to automate processes or combined to create new, potentially useful information [13]. One example is placing pairs of infrared motion detectors in every doorway of an office and analyzing their data to keep track of how many people are in every room at any time [14]. Similarly, it is possible to track and estimate the number of people in an area by analyzing data from Wi-Fi access points communicating with their smart phones [15]. Also, video feeds from multiple cameras can be analyzed together with various data such as the position of the cameras in order to better estimate the number of people being observed and which direction they are heading [16].

In these examples similar or identical sensors are used, but gathering data from completely different sensors has even greater potential for raising efficiency and enabling automation. A ventilation controlling system could be even more efficient if it could keep track of not only the number of people in a building, but also the temperature, humidity and CO<sub>2</sub> levels. By utilizing sensors, their data and various actuators, there is great potential for automating heating, ventilation, lighting, power management, and security systems. The use of smart building management systems could potentially result in significant energy savings and more efficient use of resources [12, 17].

However, there are various challenges to overcome when developing a system making use of different sensors. For example, the sensors might use completely different protocols,

making it impossible for the devices to communicate directly, or for the system to store data from the sensors without modifying the data first. For a system to be able to handle different sensors they need to be properly integrated by having their protocols translated into a standardized protocol the system can handle. This makes integration of sensors and devices from different companies difficult and time-consuming [17, 18].

IOTAP (Internet of Things and People) is a research center at Malmö University that focuses on projects on how to get the most out of IoT [5]. One of their current projects, CoSIS (Cooperative, Self-Aware and Intelligent Surveillance Systems) focuses on the design of intelligent surveillance systems that make use of connected sensors and cameras. Besides Malmö University, the companies Axis Communications (hereafter, Axis), Sigma Connectivity and Sigma Technology (hereafter collectively referred to as Sigma) are involved in the project [2]. This thesis is a contribution to the CoSIS project.

## 1.2 Research aim and research questions

This section describes the aim of this thesis and also details its research questions, prerequisites and limiting factors.

### 1.2.1 Research aim

As stated in section 1.1 the integration of sensor data is an obstacle to the development of IoT systems. For a smart building management system to work as intended, the process of gathering and analyzing data should be reliable and scalable [17]. If a system successfully integrates and analyzes sensor data, and is scalable enough to support the large amount of sensors and data processing needed, it could be the foundation of a larger building management system. The research aim of this thesis is:

To investigate reliable, scalable and modular methods of gathering, storing and analyzing sensor data useful for improving and automating building management.

### 1.2.2 Research question

A research question was formulated in order to concretize the research aim. It was decided that the research should result in a prototype system using a network camera and at least one multi-sensor device (hereafter MSD), since they were deemed integral to the thesis. The network camera has the ability to detect movement, which can be used to count people moving in and out of rooms, a method used in several projects [6, 14, 19, 20]. The MSD can monitor temperature, light intensity levels, humidity and other objective measures, which can be used by a building management system [12, 17]. See section 5.1.5 and 5.1.6 for more information about the camera and MSD. The research question of this thesis is:

How feasible is it to use a smart network camera as the central hub of a system gathering, storing and analyzing sensor data usable for building management?

The research question was solved in two steps:

- 1) Design and prototype a system that in practice demonstrates how a network camera and MSD can be used to gather data beneficial for building management. Other devices or sensors can be implemented as well if their inclusion is justified.
- 2) Investigate the possibilities of using the network camera as the host of the system described in 1).

If a working, scalable prototype could be installed on a network camera, the camera would function not only as a sensor, but also control the system. This would eliminate the need to install the system on a separate computer.

### **1.3 Scope and limiting factors**

Prototype testing was limited to the monitoring of a single room, with a single entryway. This was done due to several reasons. First of all, the project's budget was limited, and by constricting the testing to a single room, less equipment was required. It also made it easier to perform tests in a real-life environment, since only the people entering the room had to be informed that a monitoring system was running. So the test was simplified while still giving an indication of the system's capabilities and limitations. Because of time constraints only one real-life environment test was performed. Also, it was not possible to test the scalability of the system using physical sensors. Acquiring a large amount of MSDs was not feasible due to budgetary constraints. Instead, the system's scalability was measured using simulated sensors. CO<sub>2</sub> sensors were deemed too expensive for this project and not included, even though CO<sub>2</sub> sensors could have been used to measure the air quality of a room (see section 3.1 and 3.2).

## 2 Theoretical background

In this chapter, topics relevant to the thesis are presented, such as IoT systems and how to design them, the benefits of smart building management, and people counting methods.

### 2.1 Internet of Things

In section 2.1.1 the concept of IoT is explained, and examples of IoT projects are given. Section 2.1.2 describes the issues of performance and scalability, while section 2.1.3 explains the architecture of IoT systems, and how to design the systems. Section 2.1.4 discusses issues regarding security, privacy, and personal integrity in the field of IoT. Finally, the concept of design patterns is explained, as well as how the use of design patterns might benefit developers of IoT systems.

#### 2.1.1 What is the Internet of Things?

The phrase "Internet of Things" was first introduced in 1999 [13, 21]. IoT describes a network of physical objects connected to the Internet in order to communicate and share data with each other [11]. These objects can be all kinds of devices and appliances, vehicles, clothes, and even the human body itself. The objects are outfitted with - or wirelessly connected to - sensors which are used to gather relevant data. The goal of IoT is making use of these large amounts of data to make systems more efficient and automated [11, 13, 22]. The concept is quite similar to ubiquitous computing, in that it describes a world in which many different objects can contain processors and connect to the Internet, not just powerful desktop computers or smartphones [4]. But since sensors play such an important part, IoT can also be described as a network of sensors [13]. In short, the point of IoT is to use physical objects to gather data with no input from humans. Instead of wasting time collecting data manually, humans can concentrate on finding ways of using the data to reduce waste of resources [21].

There are several examples of IoT related solutions to various problems. Such as using light beam and IR counting sensors to analyze occupancy levels of a house and adjust the airflow of the ventilation system accordingly [19], using video streams and location data to better count the number of pedestrians in an area [16] or decrease traffic congestion and energy consumption in a whole city [23]. IoT projects relevant to this study are reviewed in chapter 3.

#### 2.1.2 Performance and scalability

IoT systems often have to deal with large amounts of sensors and data [24]. It is important that a system is capable of scaling up its performance in order to handle an increased workload, such as more users, more sensors and actuators, or added functionality [25]. One method of increasing scalability is through vertical scaling, which involves adding more RAM or CPU cores to the computer performing the tasks. Another is through horizontal scaling, which involves adding more computers to the system, and having them share the workload. This might not speed up processing time of the individual tasks, but

allows the system to handle additional tasks. Of course, the system needs to be designed with this in mind for this to work effectively [26].

### 2.1.3 IoT architecture and design

When developing an IoT system there are several aspects regarding its architecture and design to consider. The architecture of IoT systems can be described as being divided into three separate layers. The layers are - from bottom to top - the sensor, network, and application layer. The sensor layer includes the sensors as well as the network connecting them to each other, and to the gateway used to bridge the gap between the sensor and network layers. The network layer processes data from the sensor layers and handles all communication between the sensor layer, the application layer, and the Internet. Finally, the application layer represents the applications, such as intelligent homes, making use of the collected data [13]. There are several variations of the system architecture, some simply using different names, and some going further in-depth by using more layers [13, 23, 27].

A device, or object, used in an IoT network should have four important attributes [24]. They are:

1. Automation: the gathering, processing and sharing of data should be automated [24].
2. Intelligence: the object should be capable of acting intelligently according to the situation [24].
3. Dynamicity: objects placed in a new area within a different network should be recognized dynamically [24].
4. Zero configurations: users should be able to configure the object without technical expertise [24].

Lee and Kim also point out other aspects that need to be taken into consideration when designing IoT systems. It's important that communication between devices works properly regardless of the scale of the application area. A scalable IoT system works whether it's used in a home environment or a large building. However, a large number of sensors will result in big data volumes. If every sensor tries to communicate with all the others, there is a risk of them running out of memory or processing power. The system needs to have ways of handling this, and also be robust enough to function even if some devices stop working or if there are problems with the communication [24].

One major obstacle to IoT systems development is the issue of sensors from different manufacturers using incompatible protocols, since it complicates the integration of data. One solution to this problem is letting the sensor and network layer communicate through a shared platform which only accepts data formatted after a standardized protocol. In this case, the data needs to be properly formatted before it is sent to the platform [17]. Software called middleware can be used to make sure different components of a system can communicate properly [25].

There are many different kinds of platforms, but they can be categorized into three kinds of systems: centralized, semi-distributed and fully distributed systems [28].

1. Centralized systems transfer sensor and actuator information through a centralized server, or servers (cloud based or otherwise). This means it is difficult to establish

connections between end devices, and if the central server fails, the rest of the system cannot function [28].

2. Semi-distributed systems instead use initiate session protocols to first establish contact between the connected devices, which then communicate directly. A centralized node is used for coordinating the communication between the devices. This solution is often faster and easier to scale than centralized systems, however, it still leaves the system vulnerable due to the centralized node [28].
3. Fully distributed systems let each individual device store and administer its own information, effectively creating a peer-to-peer network. This makes the systems scalable and resilient, since the failure of a single device does not significantly effect the others. This solution does however require some overhead to function, and the devices need to perform more operations. This makes the systems less lightweight, so they might require more powerful devices [28].

Another design issue that will become more important the more IoT systems emerge is how to handle security [29].

#### **2.1.4 Security and privacy concerns**

IoT increases the need for well implemented security measures, since IoT systems increase the number of connected devices available. There is also a trend towards developing more autonomous systems, furthering the need for adequate security. Some important security issues that need further research are data encryption, secure communication protocols, and the protection of sensor data [29].

However, security is not the only concern. The automated gathering of data raises privacy and integrity concerns which should not be ignored [30]. The Swedish Personal Data Act regulates how personal information may be stored and used. The law prevents violation of personal integrity when personal data is processed. When the number of personal devices are counted using a Wi-Fi monitor, the system will collect a MAC address for each device. The MAC address could be classified as personal information and therefore be under the guard of the personal data act. However, according to Adolf Slama and Martin Brennan at the Swedish Data Protection Authority, it has not yet been decided if the MAC address should be regarded as personal information in Sweden. According to Mr. Brennan, a report on a similar situation will arrive sometime in the year of 2015. Mr. Brennan suggested treating MAC addresses as personal information in the meantime, which means that privacy and integrity should be a priority when developing a system utilizing MAC addresses to count devices [31].

#### **2.1.5 Software development and design patterns**

When designing and developing systems, making use of design patterns can save both time and resources. Patterns are a way of describing software designs. Each pattern documents a well-known problem and well-known solutions to this problem. It also describes the context in which a solution might be preferable, and when it should be avoided. Systematic use of patterns ensures that a software developer does not need to re-invent solutions for every problem, but instead can utilize solutions that are empirically proven to work [25]. Design

patterns can be applied when developing IoT systems, in order to ensure their functionality as well as simplifying the development process [32].

## **2.2 Benefits of smart building management**

The cost for building maintenance is about 70% of the total building lifetime costs. Of these 70%, as much as 60% is spent on heating, ventilation and air conditioning (HVAC). Needless to say, there is great potential to reducing building lifetime costs by improving building management. Using sensors for building management is a way to reduce energy and maintenance spending. An implemented and intelligent building management system can reduce heating costs by 25% and lighting costs by 15% [33].

Sensors used to improve building management include temperature, humidity, electricity meters, light intensity sensors, motion sensors, CO<sub>2</sub> and sensors for counting people and estimating building occupancy [17, 19].

## **2.3 Existing methods for counting people**

A lot of research has been and is being put into how to best count people in various locations. Today, there are a number of ways to accomplish this task. Examples include the use of image analysis in a single video surveillance camera, multiple image sensors, and radio irregularity in the IoT.

Researchers have previously shown that when multiple sensors are combined, greater accuracy can be achieved both with cameras [16] and with other optical people counting sensors [34].

### **2.3.1 IR beam counters**

A beam of infrared light can be used as a people counter. Whenever the beam is broken, a person or other object has passed through the counting point. With multiple beams and clever placement, occlusion can be avoided and accuracy improved [35].

### **2.3.2 Cross line people detection**

This method involves mounting a camera above a passage looking down on the passage. The camera should be mounted vertically with respect to the floor in order to minimize occlusions. When a person or group of people is detected by the camera, they are tracked until they reach a specified "counting line". Then, the number of people in the group is calculated and they are counted as having crossed the line [36].

### **2.3.3 Group estimation with multiple cameras**

The number of people in a group can be difficult to count. Individuals might be hard to distinguish from each other if standing close, since the individuals might obscure each other from a camera's view. Even with multiple cameras an individual might be obscured from all views. A way to solve this is by extracting all foreground objects from the background

in each camera picture and then extending the camera's visual cone behind any individuals blocking the view. This area is of a known size and can be decreased by cameras with different views of the scene. An estimation is made of both the least and highest number of people that can fit in the calculated area. When these two numbers converge, an exact count for the number of people in the area can be determined. If the two numbers do not converge, it is most likely that the lowest number is the one closest to reality [37].

#### **2.3.4 Radio irregularities in IoT**

Wireless communications is an important part of IoT. This communication is subject to radio irregularities, which means that radio waves are scattered, reflected or absorbed by objects in the radio wave path. When people pass between two nodes in a radio network, the signal strength will fluctuate. This can be utilized to count people moving through a network of wireless IoT devices [22].

#### **2.3.5 Wi-Fi counting**

Smartphones with Wi-Fi enabled continuously transmit probe request messages to detect nearby Wi-Fi networks. This can be used to detect and track smartphones even when they are not associated with a Wi-Fi access point. Each Wi-Fi message sent out by a device contains a unique address which enables the detection and counting of unique devices in the vicinity [15].

### 3 Related work

In this chapter articles relevant to the thesis are reviewed. In the end of each section, comments on how the paper relates to this thesis are presented.

#### 3.1 Kuutti et al. - Real time building zone occupancy detection and activity visualization utilizing a visitor counting sensor network

The objective of this study by Kuutti et al. was to implement a network of counting sensors inside a building in order to determine room occupancy levels and use the data to control a demand-controlled ventilation (DCV) system, by raising the airflow if occupancy levels increased [19]. To count people visiting the building, they used twelve light beam sensors and three IR camera sensors spread throughout ten zones on three different floors. In order to determine if visitors were entering or exiting rooms, the sensors were direction sensitive and sent a different signal depending on the direction of the visitor.

The reason they wanted to use light sensitive sensors was that the commonly used CO<sub>2</sub> sensors make the DCV slow to react to high occupancy levels, since it takes time for the CO<sub>2</sub> levels to rise. Counting sensors, on the other hand, react without delay. Furthermore, counting the number of visitors can be used for commercial and security purposes.

Every sensor was connected to a logger, which in turn communicated wirelessly with one of two gateways, using ZigBee radio modules. The gateways were in turn communicating with a server computer through an Ethernet connection. For every visitor the logger also added a timestamp. Every five minutes the collected data was used to calculate the occupancy levels.

The study showed that it is possible to control DCV by counting the visitors. However, due to issues such as two cameras counting a visitor simultaneously, and multiple adjacent visitors being counted as one, the system was prone to cumulative errors. The system had to compensate by not allowing less than zero visitors, and by resetting the count every midnight. The authors suggest using more sophisticated counting algorithms, and that the counting sensors might be used to complement CO<sub>2</sub> sensors. They also note that video analysis might be a more reliable way of counting visitors, but that it would significantly increase cost.

##### Comments

The paper raises several issues that we had to take into consideration when developing and implementing our solution. However, our experiment is lower in scale, with only one room being surveyed. This should make it easier to avoid cumulative errors when counting. Also, we are not using ZigBee, but instead use Bluetooth and Wi-Fi.

#### 3.2 Mysen et al. - Occupancy density and benefits of demand-controlled ventilation in Norwegian primary schools

In this paper Mysen et al. present data gathered from the inspection of 150 classrooms at 81 randomly selected Norwegian schools [20]. The goal was to compare CO<sub>2</sub> levels depending on whether a school used: 1) a constant air volume (CAV) ventilation system, in this case using the ventilation at full capacity throughout the whole school day, 2) a demand-controlled ventilation (DCV) system based on data from CO<sub>2</sub> sensors, or 3) a DCV system

based on data from IR occupancy sensors. The IR sensors were not used to determine the number of students in classrooms. Instead, they were only used to ascertain whether the classrooms were occupied or not. In short, the DCV-IR system turned the ventilation on when a room was occupied and turned the ventilation off when it was unoccupied. The resulting data shows that using CAV is inefficient, especially since the classrooms seldom are fully occupied. In fact, using DCV-CO<sub>2</sub> reduced the energy usage during a day by 38%, compared with using CAV. Using DCV-IR reduced it by 51%. The only problem with using the IR sensor was that it made the ventilation system use full force even if only a few persons were in the room, making it drafty while also wasting energy. The authors suggested using IR sensors capable of measuring the number of people in the room.

### **Comments**

The paper does not provide many details on how to set up sensors or integrate an IoT system. However, it does prove the usefulness of sensor based control of actuators such as ventilation systems. Implementing even a simple binary solution like DCV-IR resulted in a more energy efficient ventilation of the classrooms, compared with having the ventilation on at all times. If they had actually counted the number of people in the classrooms and adapted the air flow accordingly, the system might have saved even more energy.

### **3.3 Musa, Eriksson - Tracking Unmodified Smartphones Using Wi-Fi Monitors**

In this paper, Musa and Eriksson explore ways to track smartphones using Wi-Fi monitors [15]. Smartphones periodically send out probe messages to nearby access points when searching for available Wi-Fi networks. A Wi-Fi monitor can be used to pick up these probe messages, which contain the phone MAC-address. This can be used to detect the number of unique devices with Wi-Fi enabled that are in the vicinity of the monitor. Musa and Eriksson use multiple Wi-Fi monitors placed along a highway in order to track smartphone movement along the highway.

They also describe methods for making the smartphones send out messages more frequently. They use three methods for this. First, they emulate access points with two popular access point names (SSID's), this increases the number of phones detected. Second, they emulate access points that correspond to the SSID's looked for by each smartphone, this increases the number of probe packets sent out by each phone. Third, they send RTS (Request to Send) packets to each detected phone, which makes them respond with a CTS (Clear to Send) packet.

The paper concludes that the authors "have demonstrated that tracking unmodified smartphones using Wi-Fi monitors is both practical, economical and accurate".

### **Comments**

Musa and Eriksson described in fairly good detail how this can be accomplished in an efficient and accurate way [15]. The work by Musa and Eriksson inspired the authors of this thesis to include a Wi-Fi monitor in the prototype, as a means of counting people.

### **3.4 Liu, Mu - An efficient algorithm for fault tolerance in multisensor networks**

This paper, by Liu and Mu, addresses the problem with sensors in a multisensor network either failing or reporting contradictory measurements [38]. They propose an algorithm based on statistical methods for sorting out the correct sensor values from hundreds of measurements.

The algorithm is robust and works even when the number of working sensors is low in comparison to the number of faulty sensors. The authors test their algorithm by simulating sensors. Even with only 50 usable sensors of a total 2050 sensors, the algorithm can reliably catch the correct value in a narrow interval.

#### **Comments**

A problem with this approach might be that faulty sensors may not give measurements that are evenly distributed along a wide range, but instead give measurements offset from the real value by a certain amount. In this case, the algorithm presented would probably give the wrong value when the number of faulty sensors is equal to or greater than the number of working sensors. However, the algorithm might work well for errors caused mostly by noise.

The methods described in the paper might be of use to us during the design of our systems. For example, we could use multiple Wi-Fi monitors and apply this method to the output data. Also, the method might be useful during the analysis of our data, especially if we end up getting inaccurate measurements.

### **3.5 Bin et al. - Research on data mining models for the internet of things**

This paper begins by stating that large amounts of data will be generated in an IoT network. The example given is a supermarket, with over 700,000 RFID tags being read every second. This would amount to 544TB of data every day. Therefore the efficient management, analysis and mining of IoT data is an important subject [39].

Bin et al. look into four models for IoT data mining; a multi-layer model, a distributed model, a grid based model and a model derived from a multi-technology integration perspective. Two of these models, the multi-layer approach and the distributed model, are especially interesting to us.

The multi-layer approach defines four layers. The data collection layer, which performs collection of the actual data from sensors, RFID streams, GPS, etc. The next layer is the data management layer, which uses a centralized or distributed database to manage and store collected data. The event processing layer integrates data, it is used to analyze events in IoT. The final layer is the data mining service layer. This is based on the data management and event processing done by the event and data layers. In this layer objects and events are classified, forecast, clustered and association analysis and pattern analysis of the data is provided.

The distributed model aims to deal with a few problems that the centralized model has. These problems occur due to the decentralized nature of IoT as well as the large data amount. Large volumes of data is stored at different locations, which makes centralized processing of the data harder. The large volume of data that require real time processing

means that centralized nodes have a high hardware requirement. Another problem is the security issues, privacy issues and legal constraints of putting all data together in a single database. Finally, the transmission of data is energy-costly and does not use the limited resources of nodes in an optimal way. The distributed model solves these problems by preprocessing data in each node and only transmit the necessary data.

The grid based model involves a grid computing solution to analyze data. The grid allows for quick and convenient access to processing power when it is needed.

The model from multi-technology integration uses different context aware sources that send their data to a central "hub". The data is then analyzed and the gained knowledge is forwarded to service oriented applications, such as intelligent transportation, property or logistics.

The paper also looks at a few key issues in IoT data mining and what factors must be considered when designing an IoT data mining system. For example, one issue is that IoT will produce massive amounts of data, it is therefore necessary to consider how to manage the data in an effective manner.

### **Comments**

The paper covers a lot of subjects relevant to us. Especially when we look at the scalability of our system design, as the paper provides a few topics to consider regarding scalability and large amounts of data. Also when designing our system in the first place, we need to think about some of the topics in this paper.

### **3.6 Wahl, Milenkovic, Amft - A Distributed PIR-based Approach for Estimating People Count in Office Environments**

In 2012 Wahl, Milenkovic and Amft published two articles about their system for counting people in office environments using passive infrared (PIR) sensors. The main difference between the two articles is that in the first (the article in the heading) they perform a field test in an office [14] while using a simulation in the second [6]. They also expand on the systems functionality by including algorithms for estimating distance and movement paths [6]. Their goal was to use off-the-shelf products to develop a working, energy efficient counting system which can be used for controlling the lighting or ventilation in office spaces [6, 14].

In this system the PIR sensors are placed in doorways to count people as they pass by. The sensors cannot determine which direction a person is moving so every doorway has one sensor outside and one inside the room. By analyzing time stamps it is then possible to determine which direction a person is moving and how many are inside any given room. For the field test every sensor was connected to a micro controller, which in turn used a wireless communication unit. Every such "node" was powered by solar cells. When a person enter the PIR sensor's field of view, it sends a trigger to the controller, which forwards the event to the wireless unit. The unit then transmits the event to a remote access point. When waiting for a trigger, the node is in sleep mode to conserve energy.

According to the field test the sensors work well with few errors [14]. For the simulation tests the authors plotted an office space with several sensors, and tested two different algorithms to simulate people detection complete with errors; one based on distance, one based on movement. The scenarios of the simulation, that is the movement of employees,

were based on the real life field test. The results of the simulation test are also encouraging [6].

#### **Comments**

These articles show that it would be possible to determine movement direction using simple IR sensors, as long as at least two are used for every doorway. This enables the system to keep accurate count of people inside an office complex, even when rooms are connected. A difference between this system and our own is that we're planning on gathering more data than just the number of people going in or out of rooms.

### **3.7 Kamilaris, A. and Pitsillides, A. - Towards interoperable and sustainable smart homes**

In 2013, Kamilaris and Pitsillides published an article about smart homes and the integration of different devices [17]. They describe the difficulties of integrating multiple heterogeneous sensors into a single system as different device manufactures use different standards and protocols for their devices. They state that the integration process requires advanced programming skills and a considerable amount of time.

Kamilaris and Pitsillides describe a system architecture suitable as a framework for smart homes and how the system can be split into layers and components. They also performed case studies, mostly focused on the benefits of energy awareness and the effects of energy awareness on energy consumption.

#### **Comments**

This article is relevant for our thesis because it describes the difficulties and challenges of designing and implementing a system that integrates heterogeneous sensors. The parts of this article that do not relate to this thesis are mainly about the presentation layer. Kamilaris and Pitsillides have done a lot of work to make their system user friendly and to allow non-programmers or inexperienced programmers to use their system.

## 4 Method

This section explains the research process and the methods used in performing the two steps necessary to solve the research question of this thesis (see section 1.2.2).

### 4.1 Step 1: Design and prototype a building management system

The prototype was developed using a five-stage systems development process, proposed by Nunamaker and Chef. The stages should be done in order but if new insights are gained it's encouraged to revise design decisions made in earlier stages in order to improve the system, see figure 1. This can be done at any time [40]. The following sections will describe the process for this particular project. The results are shown in chapter 5).

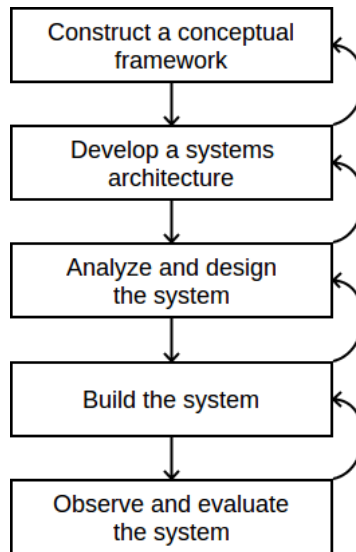


Figure 1: The different method stages and their relations.

#### 4.1.1 Construct a conceptual framework

During this stage it was important to clearly define the research question so it could provide a focus for the rest of the development. It was already decided that the thesis would involve a system making use of a network camera and at least one MSD (see section 1.2). However, this was not enough to decide on a research aim and research question, or if other sensors were needed. So a literature survey about IoT systems, and building management, was performed, in addition to brainstorming sessions and meetings with representatives from Malmö University, Axis Communications, Sigma Connectivity and Sigma Technology. The study provided the information used in chapter 2 (theoretical background) and chapter 3 (related work).

#### 4.1.2 Develop a systems architecture

With the conceptual framework as a basis, this stage was dedicated to specifying the individual components of the system, their purpose and how they should interact. Now, when

a research aim and research question had been formulated (see section 1.2), a requirements specification could be written. Most of the requirements were decided on during another brainstorming session, using the knowledge gained during the first stage as a basis. A rough system was proposed and split into subsystems, then further requirements were specified for each subsystem. See section 5.1.2 and 5.1.3 for the final list of requirements and final system architecture. Ways to validate the system against the requirements were also discussed.

Various design patterns were considered, mostly patterns detailing distributed systems. The solution to how the sensors were to communicate with the platform was inspired by the "Broker" pattern, which details a method for communication between system components. Instead of sending messages, a component invoke methods in the receiving component in order to deliver data or requests. This can optimize communication by reducing overhead [25]. However, the focus was on developing a working system, not on optimization, and no pattern was fully utilized.

#### **4.1.3 Analyze and design the system**

During this stage the system and subsystems were designed and solutions to each requirement were proposed. Each solution proposal was analyzed in order to deduce whether it would satisfy the relevant requirements or not. The process resulted in diagrams describing the system's architectural and physical layout, flow charts, sequence diagrams and a database model. Also, necessary hardware was bought at this stage.

#### **4.1.4 Build the system**

At this point the system designed during the previous phase was implemented. The first subsystems to be implemented were the database and the integration platform. When these system were nearing completion, the MSDs were implemented as the first data sources. The MSDs were then used to test and verify the database and integration platform subsystems. The continued development focused on the camera and the Wi-Fi monitor. The system was developed using a test-driven development process in order to ensure basic functionality.

#### **4.1.5 Observe and evaluate the system**

Finally, the system and subsystems were evaluated against the requirements (see section 5.1.2). To evaluate the stability of the system, it was allowed to run for days at a time, while all errors where logged. Evaluations of scalability were done using simulations where artificial data was sent to the system. To evaluate system reliability, and gather data from the sensors, two case studies were performed. One took place in a controlled environment, the other in a real-life environment - a computer lab at Malmö University. The system was deployed for a day, during business hours, and all students were notified about the test. See section 5.1.9, 5.1.10, and 5.2 for the results of these tests. Performing case studies is a commonly used method for evaluating prototypes [41].

## **4.2 Step 2: Investigate the possibilities of using the network camera as the host of the system**

This step used the result of the previous step as a foundation. To investigate whether the camera could be used as a host, the following activities needed to be performed:

1. Investigate which programming languages the camera supports.
2. Port the most necessary subsystems to a supported programming language, if the camera does not support the language used for the prototype.
3. Perform tests to make sure the system works as intended.
4. Perform tests to evaluate how the camera's performance compares to that of the prototype.

## 5 Results

In this chapter the results of the work to solve the research question are presented and discussed.

### 5.1 System prototype

The work done during step 1 (see 4.1 for details) resulted in a finished proof of concept prototype of a smart building management system. This chapter describes the various requirements of the system and its subsystems. Then the system's logical and physical architectures are detailed, followed by a section describing the subsystems in greater detail. Throughout this section the design choices made during the project are explained.

#### 5.1.1 Sensor selection

After information about smart buildings (section 2.2) and methods for people counting (section 2.3) had been gathered and various related works (section 3) had been reviewed, it was decided which sensors the system would use.

- A network camera, used to count the number of people entering and leaving a room.
- Two MSDs, used to gather objective measures, such as the temperature, humidity and light level of the immediate area. One of the MSDs also used infrared light to detect movement.
- A Wi-Fi monitor, used to detect and count devices with Wi-Fi enabled in the area.

#### 5.1.2 Requirement specification

In this section, the requirements of the prototype, decided on during the system architecture development (see section 4.1.2), are listed.

##### Functional requirements

1. The system should be able to add new sensors during run-time.
2. The system should keep unique identifiers for each data source.
3. Analysis should be possible at the time a sensor sends data, and during regular intervals.
4. The Wi-Fi monitor should detect devices that are associated with a Wi-Fi network as well as those that are not associated with a Wi-Fi network.
5. The Wi-Fi monitor should not count the same device multiple times when estimating the number of devices in an area.
6. The Wi-Fi monitor should be implemented in a manner that does not violate individual humans' right to integrity. Detected devices should not be possible to link to a person and should not be uniquely identifiable if they are removed from the monitored area and returned at a later time.

7. The hardware used must be compatible with Linux and be able to run in monitor mode.
8. The data sources should be allowed to send data at any time.

### Non-functional requirements

1. The system should be scalable and collect, process and store data gathered from many (1000+) heterogeneous data sources.
2. The system should be reliable, and able to run for at least 48 hours straight without crashing or losing data.

### 5.1.3 System architecture

Figure 2 describes the architectural layers of IoT applied to the system (see section 2.1.3 for information about IoT system architecture). The bottom layer is the sensor layer. This layer consists of various sensors and their associated gateways. The gateway bridges the gap between the sensor layer and the network layer. In the network layer, the integration platform collects and stores the sensor data in a database. The data is then analyzed and used in the application layer. Applications for the prototype system is mainly building management.

The prototype uses a limited number of sensors. The sensors consist of two wireless MSDs, a network camera, and a Wi-Fi monitor. Since the MSDs use Bluetooth Low Energy (BLE) for communicating, a Bluetooth enabled smartphone is used as a gateway to collect data from the MSDs.

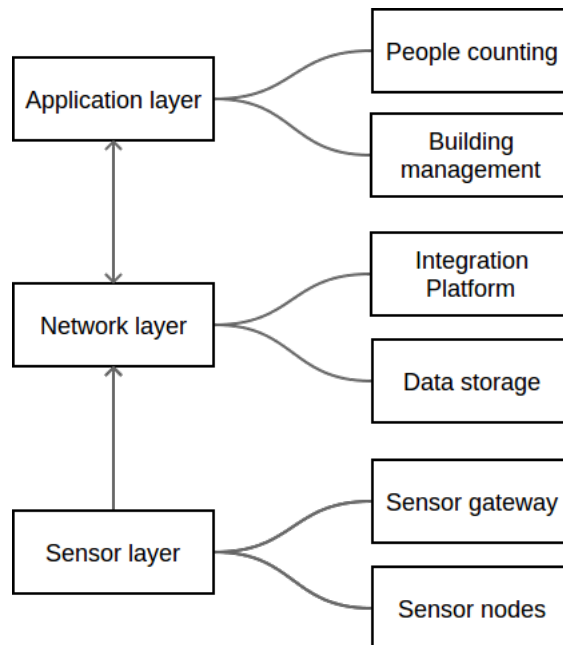


Figure 2: The layers of IoT applied to the system.

Figure 3 describes the logical system architecture of the system. The system uses a number of sensors that gather different kinds of data. Because of this, the sensors are integrated

in the so called "integration platform". Each sensor type has its own module, containing the middleware software necessary for communicating with the platform. The integration platform stores data from the sensors in a data storage. The stored data can then be analyzed at any time.

## Sensors

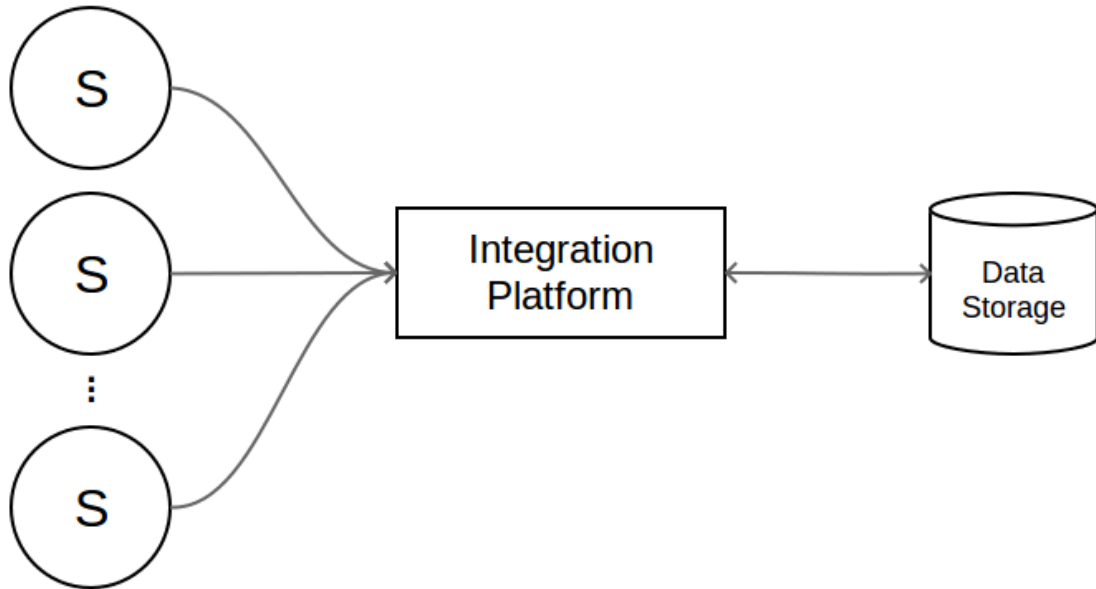


Figure 3: A logical system diagram describing the components of the system. "S" represents a sensor, and the integration platform communicates with an unknown number of sensors.

Figure 4 describes the physical system architecture of the system. The sensors used are a network camera, two MSDs and a Wi-Fi monitor. A smartphone is used as a gateway to transmit data from the MSDs to a propriety cloud service. The control application, which is written in the Python programming language, receives data from the cloud service, the camera and the Wi-Fi monitor, then stores the data in a local database for analysis.

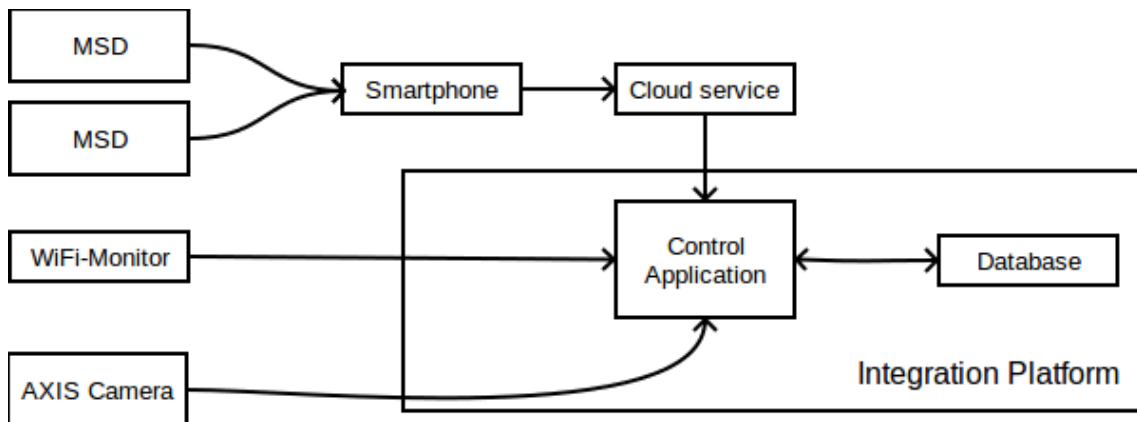


Figure 4: A physical system diagram describing the actual components used in the system.

### 5.1.4 Subsystem: Integration platform

The primary subsystem is the integration platform. It contains a control application that can gather data from multiple heterogeneous data sources and then inserts it into a SQLite database. The use of a database makes it possible to draw conclusions based on the history of the system.

The first candidate for the programming language to be used for implementing the integration platform was Python. Python is a language suitable for building quick prototypes. The rapid development was more important than for example performance in this project. The second candidate was C, since this is the preferred language when writing code to be executed on the network camera used in this project.

The decision of going with Python was based primarily on the benefits of higher abstraction level and a typeless language, allowing more functionality implemented in less time and less time spent converting between data types [42].

The integration platform provides two services. The addition of new data sources and the addition of new data. The addition of new data sources is primarily done during the start-up of the integration platform, but can also be done when the platform is running. A data source has to be added before any data can be added for that data source. The sequence diagram in figure 5 describes the process of adding a new data source.

Since every sensor device used its own protocol, middleware software had to be written for every device, to make sure they could communicate properly with the integration platform (see section 2.1.3). The modules containing the middleware for each sensor type are automatically loaded at startup.

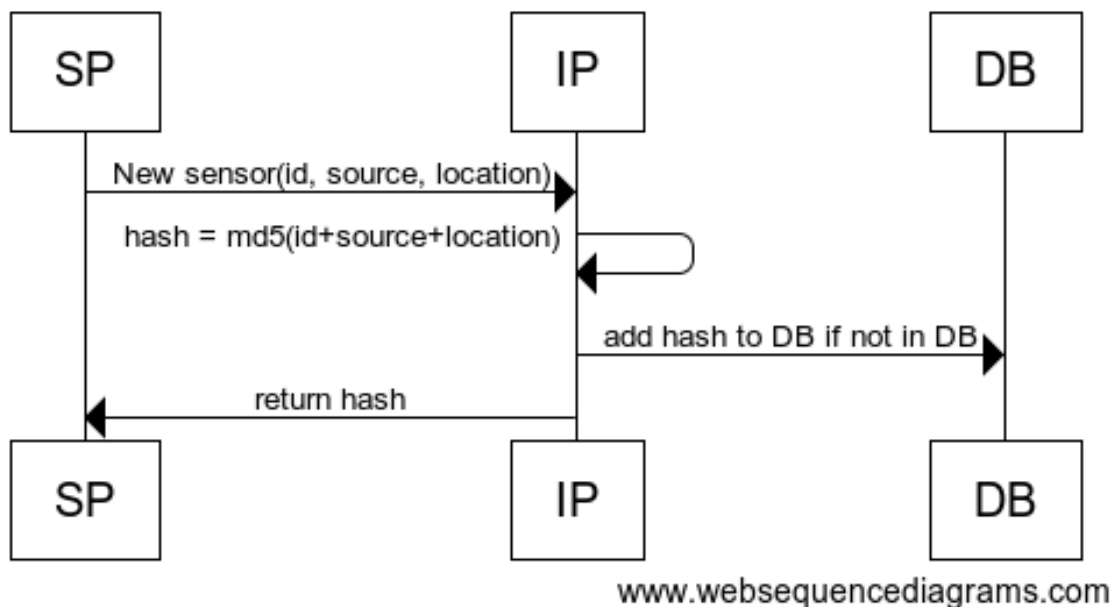


Figure 5: Sequence diagram for adding a new data source to the system. SP is a sensor platform, IP is the integration platform and DB is the database.

The second service provided by the integration platform is the addition of new data. When a data source is added, as described above, a unique identifier is provided. This identifier

is then used for storing and associating data with an individual sensor. The sequence is depicted in figure 6 and the encrypted unique identifier is labeled "hash" in this figure.

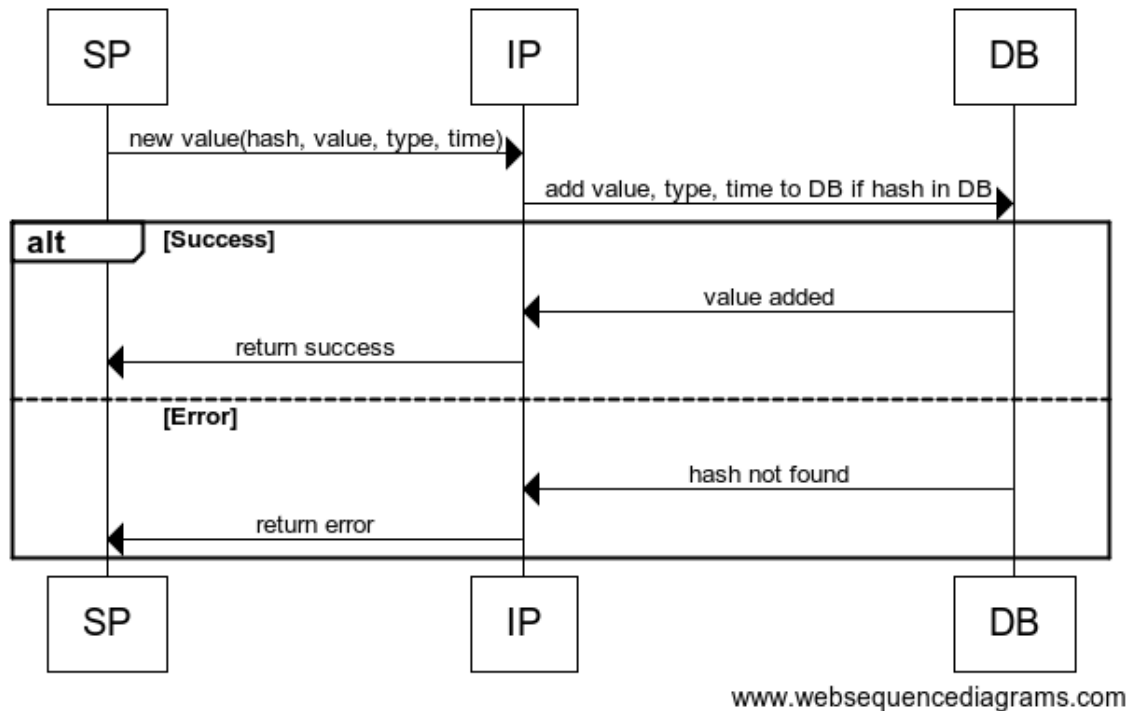


Figure 6: Sequence diagram for adding new data to the system. SP is a sensor platform, IP is the integration platform and DB is the database.

### 5.1.5 Subsystem: Network camera

The camera is an Axis P3367 fixed dome network camera which is capable of not only recording video but also detecting movement [43]. For this thesis, the camera is used for detecting movement at the right and left edges of the image. When such movement is detected, an event is generated and sent to the integration platform. The code for the Axis Camera implements a basic TCP server that listens for messages and then performs simple analysis of the messages. When movement is detected in the right part of the image and a few seconds later, movement is detected in the left part of the image, this is interpreted as a person moving from the right to the left. No video data is stored for to preserve the integrity of anyone passing by the camera.

### 5.1.6 Subsystem: Multiple Sensor Device

Two different MSDs were used, both are so called "Sensmitters" provided to us by Sigma. Both Sensmitters gather the temperature, humidity and light levels of the immediate area. In addition to these measurements, one of the Sensmitters also collect barometric pressure while the other detects movement using infrared light. The MSD subsystem gathers data from the MSDs, which use BLE to communicate with a smartphone. The smartphone runs a specific application provided by Sigma that periodically stores data in a proprietary

cloud service. The integration platform then requests the data from the cloud service every five seconds and adds it to the database.

### 5.1.1.7 Subsystem: Wi-Fi monitor

The Wi-Fi monitor is used to count all devices, excluding access points, with Wi-Fi enabled and within range. A Wi-Fi monitor is made up of wireless card capable of running in monitor mode, software to listen to all Wi-Fi radio traffic in the vicinity and a computer running the software. To enable monitor mode on the wireless card, a program from the Aircrack-ng [44] suite called Airmon-ng [45] is used. The program Tcpcap [46] is then run as a subprocess by the integration platform and is used for collecting Wi-Fi data packets. A Python script filters the captured packets and extracts the necessary information, which is the MAC addresses of devices in the vicinity.

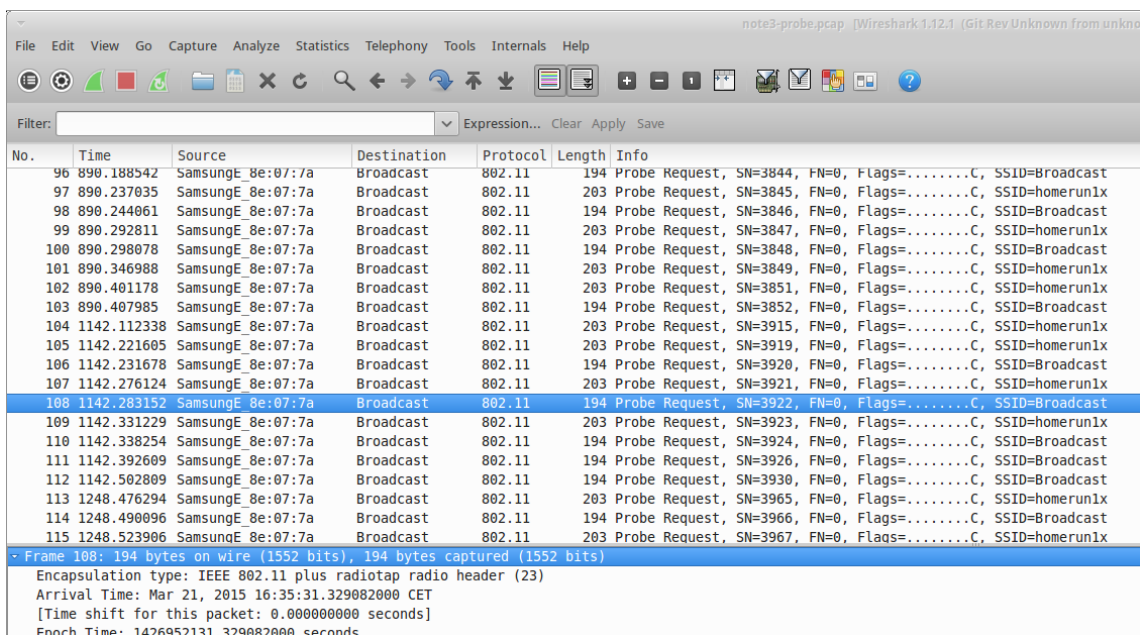


Figure 7: A screenshot of the program Wireshark [47] displaying captured probe packets from a Samsung Note 3 smartphone

Monitor mode means that the card is only used for receiving radio traffic and not for transmitting any messages. In short, everything that the card can receive, it will receive. When operating normally, only packets with the card as destination is received and forwarded to the operating system. The captured packets are then searched for probe request packets that devices periodically sends out when looking for nearby access points. The MAC-address, included with each packet, is used as a unique identifier for every device so that it is not counted twice.

If a device has not been detected for a few minutes, it is removed from the system, assuming it has left the vicinity. Figure A.1 and figure A.2 in appendix A shows the difference between a 5 minute timeout and a 30 minute timeout.

Device detection and removal was tested during development. The tests consisted of running the Wi-Fi monitor while a known smart-phone was associated with an access point,

and running the monitor while the same known smart-phone was not associated with an access point. Both cases resulted in detection and in both cases was the smart-phone removed after turning off Wi-Fi completely.

The database only stores the number of devices reported from the Wi-Fi monitor. Also, each MAC is encrypted using 128 bit MD5 hashing as soon as it enters the system, in order to protect the security and integrity of the device's owner.

### 5.1.8 Subsystem: Database

A database management system (DBMS) was used to store relevant information about each sensor and the data gathered from them. There were primarily two alternatives; SQLite and MySQL, both popular and free DBMSs. SQLite was chosen since it is lightweight and fast which makes it a good choice for embedded systems such as the network camera. For example, SQLite requires no server configuration and stores the database locally in a file, making it highly portable. A downside is that SQLite has limited support for threaded computing, which means the transaction of data from multiple sources needs to be serialized programmatically. DBMSs such as MySQL are able to write data from several sources simultaneously, which would greatly simplify the process of implementing the subsystems [48, 49].

Some data sources, such as the Wi-Fi monitor, only sent one kind of data at a time. Others, such as the MSDs, sent several different kinds of data. The database was designed with this in mind, allowing a single device to have multiple sources of data, while also allowing the user to perform detailed queries involving specific devices, types of data, time periods and locations. The database was normalized in order to maximize performance, and avoid unnecessary duplicate data. See figure 8 for the chosen data model.

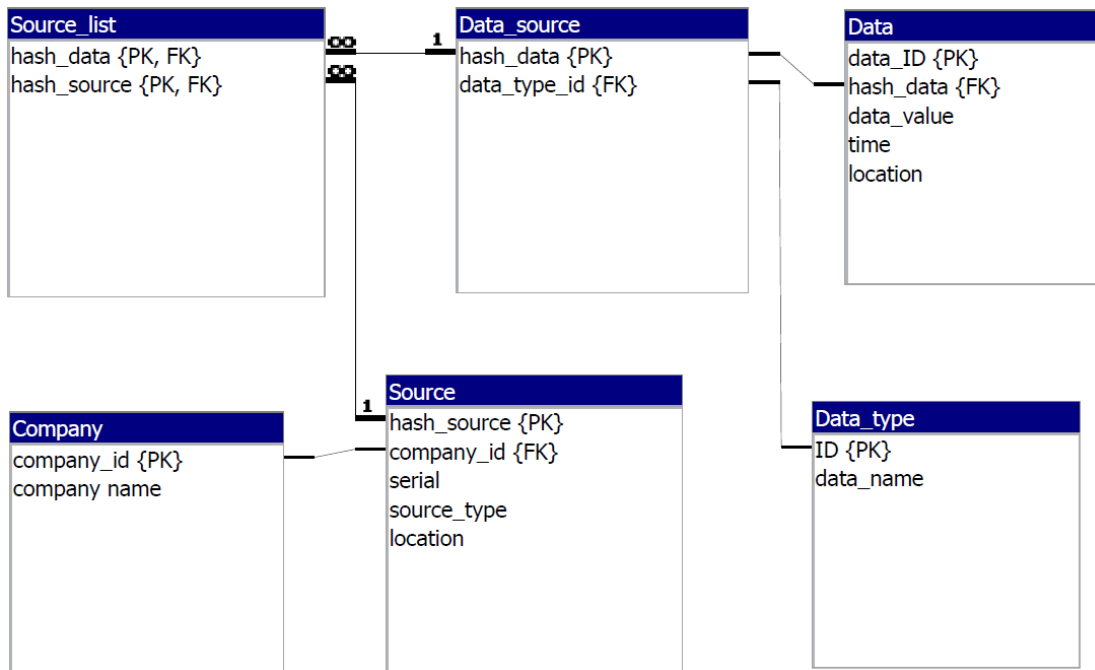


Figure 8: The data model of the SQLite-based database. "PK" and "FK" stand for primary key and foreign key, respectively.

As seen in figure 8, there are six tables:

1. Source: Contains information about the individual devices, such as the company that produces it, the serial number, the type of device (such as MSD, camera, or Wi-Fi monitor), and the device's current location. The primary key (PK) that identifies the individual device consists of the company name and the value used by the company to uniquely identify the device, such as its MAC address. The key is encrypted using 128 bit MD5 hashing. By using both the company name and the device ID, the risk of duplicates is eliminated, even if the device ID for some reason were not unique.
2. Company: Contains the names and unique identifiers of all companies registered in the database.
3. Data\_source. Contains information about all registered data sources. A data source can only send data about one type of measurement (temperature, humidity or other). For example; if device A would send data about temperature and humidity, information about A would be stored in Source, while Data\_source would store the measurements as two separate items. The table contains the type of measurement gathered, and the unique identifier of the data source made from the company name, device ID, and the type of measurement data stored. As with the Source primary key, this key is encrypted. It is presumed a device does not have any duplicate sensors, such as two or more temperature sensors. However, as long as they all have unique measurement data types any number of data sources can be attached to a device.
4. Data\_type: Contains the name and unique identifier of all registered data types. This table is manually maintained, while the other tables are populated with data sent from the devices and their associated software.
5. Data: Contains the data gathered about the data types by the sensor or sensors in the devices. Every time device A sends data about temperature, the record automatically receives an identifying integer value, and is stored as an individual data item containing the ID, the hash value of the data source, the received value, the timestamp detailing when the data was recorded, and the current location of the device. By including the data source hash value it is possible to identify which data source and device the data stems from. The current location is copied from the Source table at the time of writing. This ensures that if the device is moved the change will not affect older data from the same data source.
6. Source\_list: The table contains pairs of unique Source and Data\_source identifiers. It is used to identify which data sources belong to which devices.

When the platform starts up, it receives all relevant information about the compatible and approved devices and data sources in the set location. After this initial setup, the devices will send data either when the platform ask for new data, as with the MSDs, or whenever new data becomes available, such as with the security camera used to monitor doorways. Since the platform needs to be able to handle irregular, sometimes concurrent, transactions of data a queue system was implemented to make sure there are no conflicts when storing or reading data. SQL statements are simply added to a list which is iterated trough and emptied periodically.

A number of queries used to analyze gathered data were written. The system is able to run the queries at regular intervals and log the results, but since there are no actuators connected the system does not make use the information. For this reason the queries were

not executed during the stability and stress tests. However, they were used during the case studies. The queries are:

**Check room occupancy:** The user specifies a location and two timestamps (date / hour / minute / second). The system then retrieves all camera events, such as people leaving or entering the room, between the two timestamps. It then calculates the sum of the events and returns the information. For example, if two people left the room and three entered it during this period, the query returns "1". Of course, if it is the other way around, the result is "-1". If assumed that the room is empty at 00:00 am, this query can be used to calculate the occupancy level of the location at any given time of that same day.

**Check temperature:** The user specifies a location and the highest allowed room temperature. The system then retrieves the latest stored temperature value from sensors positioned at the given location. If the current temperature is higher than allowed, another SQL query retrieves the current room occupancy level.

Since the database is stored locally, it is possible to search for data and perform analysis when the system is not running.

### 5.1.9 Stability tests

During development of the prototype, after the database, integration platform and MSD subsystems were implemented, regular stability tests were performed. These tests were done by letting the system run for several days at a time in a controlled environment.

The tests resulted in several system crashes and at the beginning it was not possible to run the system for more than 12-14 hours. Fortunately error logging had been implemented and the cause of the problem was easily identified and fixed. The problem was insufficient error handling in the code for the MSDs. During the last stability test, the system ran for five days without any problems.

### 5.1.10 Simulations and stress test

A simple simulation data source was implemented. This data source starts a number of threads and each thread adds new data to the system in intervals that are random within a given range.

Experiments were then made where the number of threads was increased from zero in intervals of 5 up to 50 threads, and then in intervals of 10 up to 120 threads. Each thread added new data in randomized intervals of between 1 and 5 seconds. Each experiment was left running for 120 seconds to allow the simulated sensors to add some data to the database. The processor use of the integration platform process was monitored.

All measurements were performed with the integration platform running on a Raspberry Pi Model B+ at 900 MHz and using Python 3.4.

The first measurements resulted in a loss of data when using many threads, data was simply not added to the database as it should. The degree of data loss can be viewed in appendix B, figure B.4. When using 120 threads, nearly 20% of measurements sent to the integration platform were not added to the database.

After the addition of a queue functionality to the system, the simulations were redone. This time the number of threads started at 50 and was increased in intervals of 50 up to 700 threads. The results of these measurements can be viewed in appendix B, figure B.1 and B.3.

Figure B.1: The highest and lowest CPU loads in percentage, depending on the number of threads used. The graph shows that the highest CPU load steadily increased with the number of threads, reaching 95% when 700 threads were in use. In comparison, the lowest CPU load increases at a slower rate, never reaching more than 18%.

Figure B.2: The SQL statement queue size after each test. This graph shows a steady increase in average queue size, when cleared, with each added thread.

Figure B.3: The percentage of RAM used depending on the amount of threads. The graph shows a very small growth in RAM usage with added threads. After the addition of the queue functionality, not a single measurement sent to the integration platform was lost. However, the new limit for what the system can manage is around 700 threads. With more than 700 threads, the queue builds up faster than it can be processed.

## 5.2 Case studies

### 5.2.1 Case study in a controlled environment

The system was set up in a controlled environment, where only the researchers could walk past the camera and PIR sensor. The camera and PIR sensors were placed so that only a single person could pass them at any time. One person went past the motion detectors ten times each. Afterwards, the database showed that both the camera and PIR sensor had caught every event, and sent the data to the database successfully. The camera would accurately detect a person, whether they moved from the left to the right, or the other way around. Tests showed that neither the camera nor the PIR sensor sent false positives when no one passed by the sensors. The two MSDs are configured so that they gather equal values for any sensors they both have, such as temperature and humidity sensors, if they are placed in the same spot.

### 5.2.2 Case study in a real-life environment

The system was set up and left running for 6 hours in a real-life environment. The room, a computer lab at Malmö University, was actively used by students during the experiment and the measurements gathered from the MSDs are listed here. One MSD was placed on a table where a student would normally sit. The other, with a PIR sensor, was placed in a doorway connecting the two rooms of the lab, close to the floor. Data was collected from the MSDs with a five second interval. The camera was placed by the entrance, where it was supposed to count all students entering and exiting the lab. Unfortunately no useful data was gathered from network camera during this experiment due to network issues. The approximate number of people during time periods was however noted manually. Only one Wi-Fi monitor was used during the case study since one was enough to monitor the area. The monitor was placed by the entrance, next to the camera. With the exception of the camera the system ran for the entire case study without incident.

During the first two hours there were on average 4 people in the room, during the second two hours there were on average 10 people, and during the final two hours there were on average 17 people. Graphs showing data gathered during the day can be viewed in appendix C. They are:

Figure C.1: The number of people passing by the doorway with the PIR sensor. The sensor did not differentiate between people entering or exiting the room. During the first hour and a half (5400 seconds) there was not much activity, but afterwards students were constantly moving between the two rooms. The integration platform downloaded data from the cloud service storing the PIR values without checking if the stored data had been updated. To avoid any duplicate values during analysis, all rows with duplicate time values were removed. According to the sensor data, 247 people passed the door during the six hours. That is on average 41 people an hour.

Figure C.2: The temperature measured by the two MSDs. The temperatures (in degrees Celsius) only changed insignificant amounts during the study. However, there was a difference of almost one degree between the two MSDs, with the one close to the floor having the lower temperature.

Figure C.3: The humidity level of the room in percentage. During the first hour and a half, the humidity decreased slightly. Then it increased during the second third of the period, until staying relatively stable during the last two hours.

Figure C.4: The atmospheric pressure in BAR. The pressure stayed almost perfectly still during the entire test.

Figure C.5: The number of Wi-Fi enabled devices detected by the Wi-Fi monitor. The data was updated once every five minutes. The device count fluctuated quite a bit, with constant and noticeable peaks and dips. At the beginning of the test, the number of devices went from 600 to over 900 in a small amount of time, then back to 600 directly afterwards. During the middle of the period the values were relatively stable, with around 400 devices detected. At the end of the test, the amount had decreased to around 300.

### 5.3 Using a network camera as the system host

Some parts of the system were ported to C, which enabled them to be compiled for and then run on the network camera. The ported subsystems were the integration platform, the database and the MSD module. For full details of the system, see section 5.1.

The network camera is based on the MIPS architecture, which is different from most personal computers. In order to use a PC with an x86-64 architecture to compile code for a machine with MIPS architecture, a process known as cross compiling is needed. A cross compiler is a compiler capable of producing code intended to run on a different platform than the one that the compiler is running on.

All libraries used must also be compiled for the target platform and then linked when cross compiling. The database subsystem requires SQLite and the MSD module requires "Curl" to connect to the cloud service. This means that both SQLite and Curl had to be cross compiled before the project code could be cross compiled. Curl is described below:

Curl is a command line tool and library for transferring data with URL syntax, supporting DICT, FILE, FTP, FTPS, Gopher, HTTP, HTTPS, IMAP,

IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMTP, SMTPS, Telnet and TFTP [50].

The system was finally tested on the network camera and was able to run, read MSD data from the cloud service, and add the data to the database.

Figure 9 displays the results of a benchmark run on both the Raspberry Pi and on the Axis Camera. The benchmark allows the comparison of performance on a Raspberry Pi and the network camera. From the graph we see that the Raspberry Pi is almost 17 times faster when performing floating point operations and 4 times faster when performing fixed point operations.

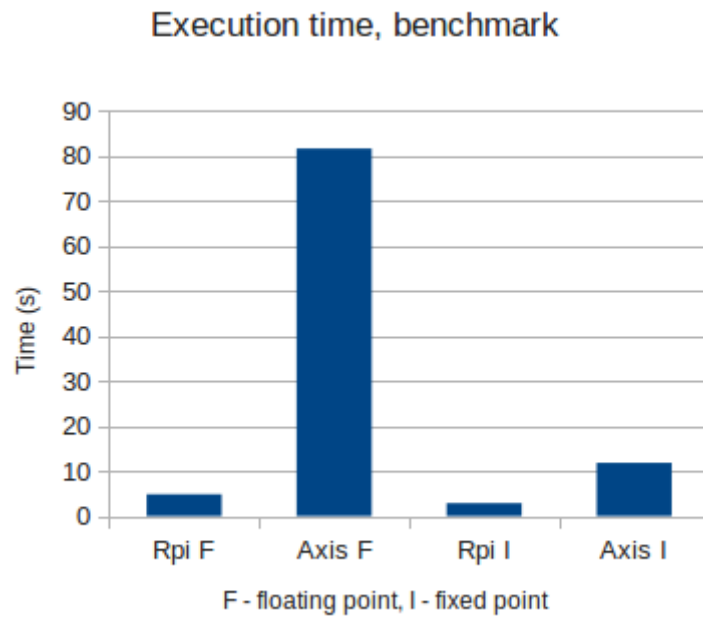


Figure 9: Benchmark of Raspberry Pi Model B+ (Rpi) and Axis Camera (Axis). The figure shows execution time of the benchmark using floating point operations (Rpi F, Axis F) and using fixed-point operations (Rpi I, Axis I). Lower is better.

## 6 Discussion

In this chapter the test results are discussed, and if the system prototype satisfies the system requirements. The feasibility of using a network camera to host the prototype is also discussed.

### 6.1 Discussion of test results

In chapter 5, sections 5.1.9 - 5.3 results from stability tests, scalability tests, and case studies were presented. In this section, these results are discussed.

#### 6.1.1 Stability test

During development the system was regularly tested for stability. Stability was one of the system requirements and important if the system was going to be used in an extended experiment setup. The system was set up and left running during extended periods of time in an apartment.

Stability issues could be noted and the bugs that caused them could easily be fixed before the system was once again tested for stability. Testing the stability during development was in retrospect a very good idea. This ensured that the system would be able to run during the final experiment setup. Had the system stability not been tested beforehand the stability issues would be noticed first during experimentation runs which could have delayed the project considerably.

#### 6.1.2 System scalability

With our current setup, running on a Raspberry Pi the system can manage about 700 simulated sensors adding data very frequently (1-5 second intervals). This is visible from the results presented in section 5.1.10. The system we have implemented is well suited for gathering real time data from a room, a floor or even a smaller building. For some types of sensors, less frequent measurements can be suitable. The temperature in a room does not fluctuate frequently and one measurement per minute or even five minutes would certainly suffice.

Our centralized system is capable of 700 sensors adding data frequently. The work by Bin et al., described in section 3.5, discuss solutions involving up to 700,000 sensors adding data every second. The approach we use has similarities with the multi-layer approach that Bin et al. describe. Our results are consistent with some of the problems Bin et al. describe with a centralized model; the limited hardware capabilities of a Raspberry Pi or a smart network camera is only sufficient to manage a smaller set of real time sensors.

#### 6.1.3 Case studies

The controlled environment case study (section 5.2.1) showed that the system could gather and store data from every sensor, and that both the PIR motion sensor and network camera worked as intended during ideal conditions.

The results of the real-life case study (section 5.2.2) show that the PIR sensor from one of the MSDs efficiently picked up on people passing by it. The data gathered from the sensor seems probable. However, we should have validated the number of people detected, using another PIR sensor, but we did not have one at hand. Unfortunately connection issues during the real-life test made it impossible to use the network camera. If two PIR sensors had been used, they could have been placed on opposite sides of the doorway. Not only could the extra sensor have been used to validate the data, but also to calculate whether a student moved in or out of the room, a technique inspired by the articles by Wahl et al. (see section 3.6). If both the camera and two PIR sensors had been available, we could have compared their data. Both the camera and PIR sensors seem to have a lot of potential when it comes to keeping track of people's movement in a building. However, more case studies and data are needed to validate this.

The temperature sensors in the two MSDs had different values throughout the entire study. Since they were placed in different positions it is apparent that the position of the sensors is important. According to the MSDs, the humidity of the room increased somewhat during the day, which seems probable, since the number of people present in the room increased during the day. The case studies show that the MSDs can be used to gather data that were deemed relevant to building management systems. But for the data to be useful, real-time analysis of the data is needed.

As we can see from our measurements in appendix A and appendix C, figure C.5 the Wi-Fi monitor does not give a satisfactory estimation of the number of devices in the vicinity. The values fluctuate frequently because some far away devices have a very weak signal which does not reach the monitor most of the time. This is true both when the timeout is set to 5 minutes (figure A.1) as well as when the timeout is set to 30 minutes (figure A.2).

However the two timeout settings give very different device counts. This is again because of far away and faint signals being allowed more time between "check-ins" before being dropped with a 30 minute timeout than with a 5 minute timeout. The use of the Wi-Fi monitor, as we have implemented it, for precisely estimating the number of devices is not a very good idea.

The measurements from Wi-Fi monitors can be useful when compared to each other. Measurements from multiple Wi-Fi monitors at different locations within an area can be used to compare how crowded an area is relative to the others. Measurements taken from the same Wi-Fi monitor at different times can be used to determine when an area is more crowded and when it is less crowded.

There is room for improvement regarding the Wi-Fi monitors. As described in section 3.3, Musa and Ericsson have a few suggestions which might improve the performance of Wi-Fi monitors. Musa and Ericsson describe a process of sending messages to devices in order to in turn promote wireless traffic from the device [15]. A possible improvement on the Wi-Fi monitors is to send messages periodically to devices and, if no response is detected, more effectively remove them from the system. Further work and experimentation in this area is needed.

## 6.2 Does the prototype meet the system requirements?

In this section it is discussed whether or not the prototype system developed for this thesis satisfies the requirements listed in section 5.1.2.

### Functional requirements

1. The system will never store duplicates of data sources, since the database uses primary and foreign keys to avoid this. The database will reject any attempt to register a duplicate. See section 5.1.8.
2. The system never counts the same data source twice. The process for adding new sensors ensures this. See figure 5.
3. It is possible to perform analysis at any time. However, at the moment the system only uses two simple queries. They are more a proof of concept than a full-fledged analysis subsystem 5.1.8.
4. The Wi-Fi monitors can detect devices as intended, see 5.1.7 for a description of tests made to verify this.
5. The Wi-Fi monitors does not count the same device twice since it uses the MAC as a unique identifier for each device. See 5.1.7
6. The implementation of the Wi-Fi monitors and the system as a whole safeguards personal integrity. See 5.1.7 for details of the implementation.
7. The Wi-Fi monitor hardware was compatible with Linux and able to run in monitor mode. If it had not been, none of the tests described in 5.1.7 would have been possible.
8. The data sources can send data to the integration platform at any time.

### Non-functional requirements

1. The system was scalable, but not up to 1000+ sensors. With more powerful hardware, this would have been a possibility. Results of scalability tests are found in section 5.1.10.
2. The system is sufficiently reliable. Section 5.1.9 describes the stability tests performed to verify this.

All system requirements were met, except requirement number 1, which was partly met. Requirement number 1 is dependent on hardware. If the integration platform had been hosted on more powerful hardware than the Raspberry Pi the system could theoretically manage more than 1000 real time sensors. However, as the hardware of the network camera is even more limited than a Raspberry Pi, perhaps it is simply not feasible with today's hardware to expect thousands of real time sensors managed by such a small device as a Raspberry Pi or smart network camera.

## 6.3 Can the system run from a network camera?

According to our results, the use of a network camera as the host of the system is possible. During scalability simulations (section 5.1.10) it was evident that the system we designed is efficient enough to run on the camera. The system was partly ported to C, cross compiled,

and successfully run on the network camera. It could gather and store data from the MSDs (section 5.3). The port included the database, integration platform, and MSD module.

As seen in section 5.3, figure 9, the Raspberry Pi which we used as a host for our full system, is up to 17 times faster than the network camera. However, when using fixed-point operations the Raspberry Pi is only 4 times faster. Care must be taken in minimizing the number of floating point operations if the system is ported to the network camera. If this can be done, the camera should be able to manage approximately 25% of the sensors that the Raspberry Pi could manage. Given that figure B.1 indicates that the Raspberry Pi can manage about 700 simulated sensors. We estimate that the network camera can realistically manage 175 sensors adding data in intervals of between 1 and 5 seconds. To be entirely sure, we would need to port the whole system prototype to the camera, and perform the same stress tests and case studies.

The tests that were performed showed that the network camera is a possible host and controller of a smaller smart building management system.

## 6.4 Vertical and horizontal scaling

As briefly mentioned in the previous section, and in section 6.1.2, the system is suitable for running on a smart network camera up to a system size of about 175 real time sensors. To raise the capabilities of the system further, vertical scaling (using more powerful hardware) could be used. Since even the very limited hardware of a Raspberry Pi could manage 700 sensors, a modern computer should be able to manage thousands of sensors.

Another solution would be to distribute the system, to scale horizontally. As described in section 2.1.3 and section 3.5, distribution is preferable for IoT systems. Distribution makes the systems both very scalable as well as very robust. If a node ceases to work correctly, the system can still continue with its intended purpose. Distribution gives an IoT system the ability to handle the massive amounts of data gathered and processed.

The system we have designed could be refitted and instances of our integration platform (see section 5.1) could be used as nodes in a larger distributed network. Multiple network cameras, each running an instance of our integration platform, could work together. This would allow each camera to gather and analyze a small amount of data and then communicate with other instances of the integration platform.

It is possible that the system would have been better optimized if we had made use of design patterns throughout the development process. However, since there were so many unknown factors when development began, such as which kinds of sensors would be used, we had difficulties deciding which pattern or patterns would best suit our needs. The focus of this project was not to produce a fully optimized system based on well-known and tested design patterns, but to investigate the feasibility of running a building management system on the limited hardware of a network camera. However, if we would develop a similar system in the future, our new found experience and knowledge would make the selection of patterns simpler, which could lead to a better optimized and more scalable system.

## 7 Conclusions

By using different kinds of sensors, it is possible to gain useful data and information, which can be used to automate processes and decrease the waste of resources. This thesis focused on investigating the feasibility of using a smart network camera as the host of a system gathering, storing and analyzing sensor data usable for building management.

### 7.1 Process

First a literature study was performed. With the study as a foundation, a conceptual framework was written, together with the system requirements. Also, at that stage it was decided which sensors to use. Afterwards a prototype system was designed, and built. Simulations and case studies were used to evaluate the prototype against the requirements. After the prototype had satisfied the requirements, select parts of it were ported to the network camera. Tests were performed to ensure basic functionality.

### 7.2 Findings

The system architecture we designed is suitable for building management systems requiring a few hundred real time sensors. It can be scalable with the use of more powerful hardware. With less powerful hardware, such as a network camera, using around 100-175 real time sensors is more realistic. If more sensors are required, or the system is deployed on a larger scale, distribution of both data gathering and data analysis is recommended.

The results show that a smart network camera can be used as a host for a building management system. However, since the number of sensors is limited due to hardware reasons, the system should be distributed. One possible solution would be creating a network of cameras, each running an instance of the system, and covering a certain area. This would increase scalability and also make the system more robust, since the system would still function as a whole even if an individual camera were to suffer hardware failure.

Wi-Fi monitors can be useful for roughly estimating the crowd density of an area at different times. This could potentially be used for comparing crowd densities in different areas at different times, and estimating how people move between areas throughout a day. However, Wi-Fi monitors are not ideal for actually counting the number of people in an area, since they are too inaccurate.

### 7.3 Further work

This thesis focused more on the gathering and storing of data than analysis of data. Further work will be necessary before the prototype can be used as a full-fledged building management system.

The prototype can gather and store necessary data, and it is possible to perform analysis. However, at the moment the analysis is limited. More research about different ways of combining and analyzing the data, and its possible uses, is needed.

Right now the most basic functionality has been ported to the network camera. Further work could entail porting the rest of the subsystems, such as the camera people counter

and the Wi-Fi monitor. After that, similar stress tests and case studies used to evaluate the system prototype can be used to evaluate the camera prototype.

Since the number of sensors used is limited due to the processing power needed to gather data from them, it is advisable to look for more effective ways to perform database transactions. It is possible another DBMS than SQLite would work better, as well.

As discussed in section 6.4, horizontal scaling is preferable for IoT systems. It would be interesting to research and add functionality for distributing the system to the integration platform and run the system on multiple network cameras which would work together and coordinate work between themselves. There is also a need to research and test which kind of distributed system (semi- or fully distributed) would be preferable for a building management system. No matter which kind, a distributed solution could be the key to extending our initial work into a system that satisfies the strong scalability requirements of IoT systems such as this.

In future projects, design patterns should be used in order to develop a better optimized, and more stable, system with increased scalability.

#### **7.4 Contributions of this thesis**

The main contribution of this thesis is the acquired knowledge of when a smart building management system can use a centralized architecture and when it should use a distributed architecture. The use of limited hardware, such as a smart network camera, as the possible host for a building management system has been investigated. The test results show that using a network camera as a host is a possibility, but only up to a certain system size. It is possible to increase scalability by using more powerful hardware, but a preferable method would be to distribute the workload across several host cameras. A building management system, consisting of a distributed network of smart network cameras and sensors, could be both scalable and reliable.

## References

- [1] Inc Bluetooth SIG. *Bluetooth Smart / Bluetooth Low Energy / BLE / Bluetooth / Bluetooth Technology Website*. URL: <http://www.bluetooth.com/Pages/Bluetooth-Smart.aspx> (visited on 03/18/2015).
- [2] Malmö University. *Cooperative, Self-aware and Intelligent Surveillance Systems – CoSIS - IOTAP*. URL: <http://iotap.mah.se/cosis/> (visited on 05/15/2015).
- [3] Vijay Krishna Pallaw. *Database Management Systems*. Delhi: Asian Books Pvt Ltd, 2010.
- [4] Adrian McEwen and Hakim Cassimally. *Designing the Internet of Things*. IoT. Chichester: John Wiley & Sons, Ltd., 2014.
- [5] Malmö University. *This is IOTAP - IOTAP*. URL: <http://iotap.mah.se/about/> (visited on 05/15/2015).
- [6] F. Wahl, M. Milenkovic, and O. Amft. “A Distributed PIR-based Approach for Estimating People Count in Office Environments”. In: *2012 IEEE 15th International Conference on Computational Science and Engineering (CSE)*. Dec. 2012, pp. 640–647. DOI: 10.1109/ICCSE.2012.92.
- [7] Miao Wu, Ting-Jie Lu, Fei-Yang Ling, Jing Sun, and Hui-Ying Du. “Research on the architecture of Internet of Things”. In: *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*. Vol. 5. IoT. Aug. 2010, pp. 484–487. DOI: 10.1109/ICACTE.2010.5579493.
- [8] M. Gomes, R. da Rosa Righi, and C.A. da Costa. “Internet of things scalability: Analyzing the bottlenecks and proposing alternatives”. In: *2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IoT. Oct. 2014, pp. 269–276. DOI: 10.1109/ICUMT.2014.7002114.
- [9] Jiong Jin, J. Gubbi, Tie Luo, and M. Palaniswami. “Network architecture and QoS issues in the internet of things for a smart city”. In: *2012 International Symposium on Communications and Information Technologies (ISCIT)*. IoT. Oct. 2012, pp. 956–961. DOI: 10.1109/ISCIT.2012.6381043.
- [10] J.A. Stankovic. “Research Directions for the Internet of Things”. In: *IEEE Internet of Things Journal* 1.1 (Feb. 2014). IoT, pp. 3–9. ISSN: 2327-4662. DOI: 10.1109/JIOT.2014.2312291.
- [11] Y. Qin, M. Sheng, and E. Curry. “Matching Over Linked Data Streams in the Internet of Things”. In: *IEEE Internet Computing* PP.99 (2015). IoT, pp. 1–1. ISSN: 1089-7801. DOI: 10.1109/MIC.2015.29.
- [12] J. Pan, R. Jain, S. Paul, T. Vu, A. Saifullah, and M. Sha. “A Internet of Things Framework for Smart Energy in Buildings: Designs, Prototype, and Experiments”. In: *IEEE Internet of Things Journal* PP.99 (2015). IoT, pp. 1–1. ISSN: 2327-4662. DOI: 10.1109/JIOT.2015.2413397.
- [13] Jihong Liu and Li Yang. “Application of Internet of Things in the Community Security Management”. In: *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN)*. IoT. July 2011, pp. 314–318. DOI: 10.1109/CICSyN.2011.72.
- [14] F. Wahl, M. Milenkovic, and O. Amft. “A green autonomous self-sustaining sensor node for counting people in office environments”. In: *Education and Research Conference (EDERC), 2012 5th European DSP*. Sept. 2012, pp. 203–207. DOI: 10.1109/EDERC.2012.6532255.
- [15] A. B. M. Musa and Jakob Eriksson. “Tracking Unmodified Smartphones Using Wi-fi Monitors”. In: *Proceedings of the 10th ACM Conference on Embedded Network Sensor*

- Systems*. SenSys '12. New York, NY, USA: ACM, 2012, pp. 281–294. ISBN: 978-1-4503-1169-4. DOI: 10.1145/2426656.2426685. URL: <http://doi.acm.org/10.1145/2426656.2426685> (visited on 03/06/2015).
- [16] N.C. Tang, Yen-Yu Lin, Ming-Fang Weng, and H.-Y.M. Liao. “Cross-Camera Knowledge Transfer for Multiview People Counting”. In: *IEEE Transactions on Image Processing* 24.1 (Jan. 2015), pp. 80–93. ISSN: 1057-7149. DOI: 10.1109/TIP.2014.2363445.
- [17] A. Kamilaris and A. Pitsillides. “Towards interoperable and sustainable smart homes”. In: *IST-Africa Conference and Exhibition (IST-Africa), 2013*. IoT. May 2013, pp. 1–11.
- [18] Yeong-Sheng Chen and Yu-Ren Chen. “Context-Oriented Data Acquisition and Integration Platform for Internet of Things”. In: *2012 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IoT. Nov. 2012, pp. 103–108. DOI: 10.1109/TAAI.2012.64.
- [19] J. Kuutti, P. Saarikko, and R.E. Sepponen. “Real time building zone occupancy detection and activity visualization utilizing a visitor counting sensor network”. In: *2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV)*. Feb. 2014, pp. 219–224. DOI: 10.1109/REV.2014.6784260.
- [20] Mads Mysen, Sveinung Berntsen, Per Nafstad, and Peter G. Schild. “Occupancy density and benefits of demand-controlled ventilation in Norwegian primary schools”. In: *Energy and Buildings* 37.12 (Dec. 2005), pp. 1234–1240.
- [21] Kevin Ashton. *That 'Internet of Things' Thing - RFID Journal*. IoT. URL: <http://www.rfidjournal.com/articles/view?4986> (visited on 03/18/2015).
- [22] Wei-Chuan Lin, W.K.G. Seah, and Wei Li. “Exploiting radio irregularity in the Internet of Things for automated people counting”. In: *2011 IEEE 22nd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*. IoT. Sept. 2011, pp. 1015–1019. DOI: 10.1109/PIMRC.2011.6139649.
- [23] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. “Internet of Things for Smart Cities”. In: *IEEE Internet of Things Journal* 1.1 (Feb. 2014). IoT, pp. 22–32. ISSN: 2327-4662. DOI: 10.1109/JIOT.2014.2306328.
- [24] Gyu Myoung Lee and Jeong Yun Kim. “The Internet of Things #x2014; A problem statement”. In: *2010 International Conference on Information and Communication Technology Convergence (ICTC)*. IoT. Nov. 2010, pp. 517–518. DOI: 10.1109/ICTC.2010.5674788.
- [25] Frank Buschmann, Kevin Henney, and Douglas C. Schmidt. *Pattern-Oriented Software Architecture, A Pattern Language for Distributed Computing*. 1st ed. Vol. 4. Chichester: John Wiley & Sons Ltd, 2007.
- [26] C. Janiesch, I. Weber, J. Kuhlenkamp, and M. Menzel. “Optimizing the Performance of Automated Business Processes Executed on Virtualized Infrastructure”. In: *2014 47th Hawaii International Conference on System Sciences (HICSS)*. Jan. 2014, pp. 3818–3826. DOI: 10.1109/HICSS.2014.474.
- [27] M. Leo, F. Battisti, M. Carli, and A. Neri. “A federated architecture approach for Internet of Things security”. In: *Euro Med Telco Conference (EMTC), 2014*. Nov. 2014, pp. 1–5. DOI: 10.1109/EMTC.2014.6996632.
- [28] Stefan Forsström, Victor Kardeby, Patrik Österberg, and Ulf Jennehag. “Challenges when Realizing a Fully Distributed Internet-of-Things - How we Created the SensibleThings Platform”. In: *The Ninth International Conference on Digital Telecommunications ICDT*. 2014, pp. 13–18.

- [29] Hui Suo, Jiafu Wan, Caifeng Zou, and Jianqi Liu. “Security in the Internet of Things: A Review”. In: *2012 International Conference on Computer Science and Electronics Engineering (ICCSEE)*. Vol. 3. IoT. Mar. 2012, pp. 648–651. DOI: 10.1109/ICCSEE.2012.373.
- [30] M.G. Michael, K. Michael, and C. Perakslis. “Uberveillance and the Internet of Things and people”. In: *2014 International Conference on Contemporary Computing and Informatics (IC3I)*. IoT. Nov. 2014, pp. 1381–1386. DOI: 10.1109/IC3I.2014.7019829.
- [31] Adolf Slama and Martin Brennan. *Interview with Swedish Data Protection Authority*. Swedish. Mar. 2015.
- [32] G. Ozvural and G.K. Kurt. “Advanced approaches for wireless sensor network applications and cloud analytics”. In: *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on*. Apr. 2015, pp. 1–5. DOI: 10.1109/ISSNIP.2015.7106979.
- [33] P. Borkowski, M. Pawlowski, and T. Makowiecki. “Economical aspects of building management systems implementation”. In: *PowerTech, 2011 IEEE Trondheim*. June 2011, pp. 1–6. DOI: 10.1109/PTC.2011.6019439.
- [34] J. Hutchins, A. Ihler, and P. Smyth. “Modeling Count Data from Multiple Sensors: A Building Occupancy Model”. In: *2nd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, 2007. CAMPSAP 2007*. Dec. 2007, pp. 241–244. DOI: 10.1109/CAMPSAP.2007.4498010.
- [35] E. Mathews and A. Poigne. “An Echo State Network based pedestrian counting system using wireless sensor networks”. In: *2008 International Workshop on Intelligent Solutions in Embedded Systems*. July 2008, pp. 1–14. DOI: 10.1109/WISES.2008.4623302.
- [36] M. Rossi and A. Bozzoli. “Tracking and counting moving people”. In: *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*. Vol. 3. Nov. 1994, 212–216 vol.3. DOI: 10.1109/ICIP.1994.413857.
- [37] D.B. Yang, H.H. Gonzalez-Banos, and L.J. Guibas. “Counting people in crowds with a real-time network of simple image sensors”. In: *Ninth IEEE International Conference on Computer Vision, 2003. Proceedings*. Oct. 2003, 122–129 vol.1. DOI: 10.1109/ICCV.2003.1238325.
- [38] Hong-Wei Liu and Chun-Di Mu. “An efficient algorithm for fault tolerance in multisensor networks”. In: *Proceedings of 2004 International Conference on Machine Learning and Cybernetics, 2004*. Vol. 2. Aug. 2004, 1258–1262 vol.2. DOI: 10.1109/ICMLC.2004.1382385.
- [39] Shen Bin, Liu Yuan, and Wang Xiaoyi. “Research on data mining models for the internet of things”. In: *2010 International Conference on Image Analysis and Signal Processing (IASP)*. Apr. 2010, pp. 127–132. DOI: 10.1109/IASP.2010.5476146.
- [40] Jr. Nunamaker J.F. and M. Chen. “Systems development in information systems research”. In: *Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences, 1990*. Vol. iii. Ulrik. Jan. 1990, 631–640 vol.3. DOI: 10.1109/HICSS.1990.205401.
- [41] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. “Design Science in Information Systems Research”. In: *MIS Quarterly* 28.1 (Mar. 2004). Ulrik, pp. 75–105. ISSN: 0276-7783. URL: <http://www.jstor.org/stable/25148625> (visited on 03/20/2015).
- [42] W. Wright and D. Moore. “Agile language development: the next generation”. In: *2006 IEEE Aerospace Conference*. 2006, 6 pp. DOI: 10.1109/AERO.2006.1656063.

- [43] Axis Communications AB. *Axis p3367-v*. URL: <http://www.axis.com/lk/en/products/axis-p3367-v> (visited on 06/07/2015).
- [44] Aircrack-ng. *Aircrack-ng*. URL: <http://www.aircrack-ng.org/> (visited on 06/07/2015).
- [45] Aircrack-ng. *Airmon-ng*. URL: <http://www.aircrack-ng.org/doku.php?id=airmon-ng> (visited on 06/07/2015).
- [46] Tcpdump/Libpcap. *Tcpdump*. URL: <http://www.tcpdump.org/> (visited on 06/07/2015).
- [47] Wireshark Foundation. *Wireshark*. URL: <https://www.wireshark.org/about.html> (visited on 06/07/2015).
- [48] SQLite. *SQLite Frequently Asked Questions*. URL: <https://www.sqlite.org/faq.html> (visited on 05/27/2015).
- [49] *MySQL :: Why MySQL?* URL: <http://www.mysql.com/why-mysql/> (visited on 05/27/2015).
- [50] Daniel Stenberg, Dan Fandrich, and Yang Tse. *Curl and libcurl*. URL: <http://curl.haxx.se/> (visited on 05/25/2015).

## A Wi-Fi monitor graphs

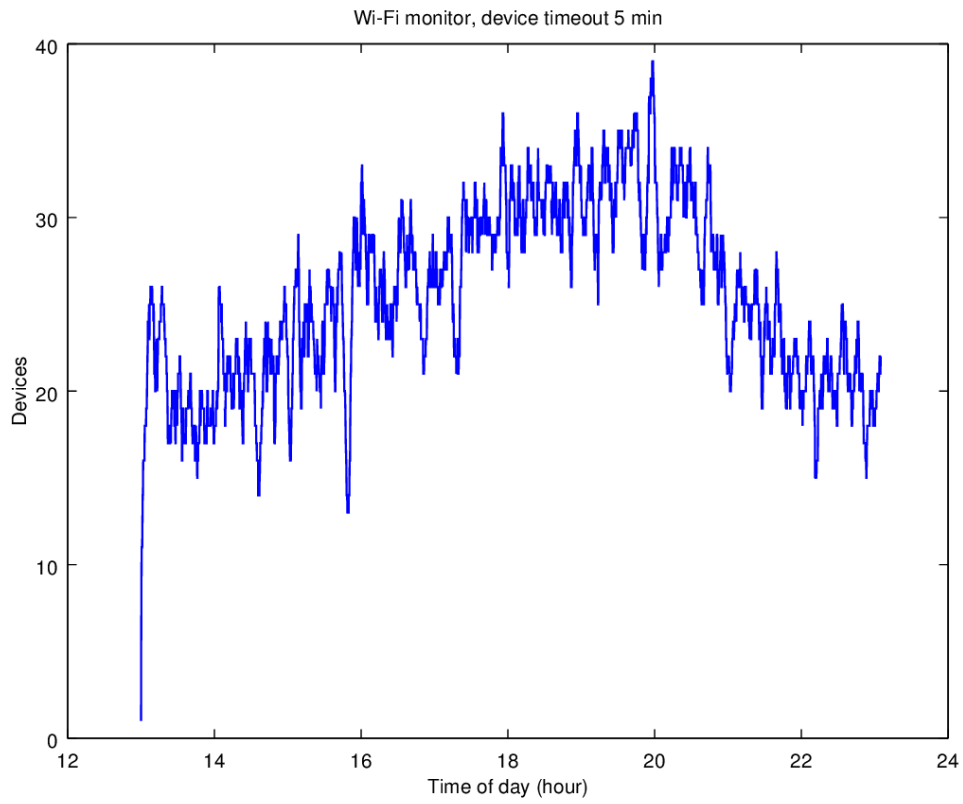


Figure A.1: The number of devices counted during a period of approximately 10 hours, with a 5 minute timeout.

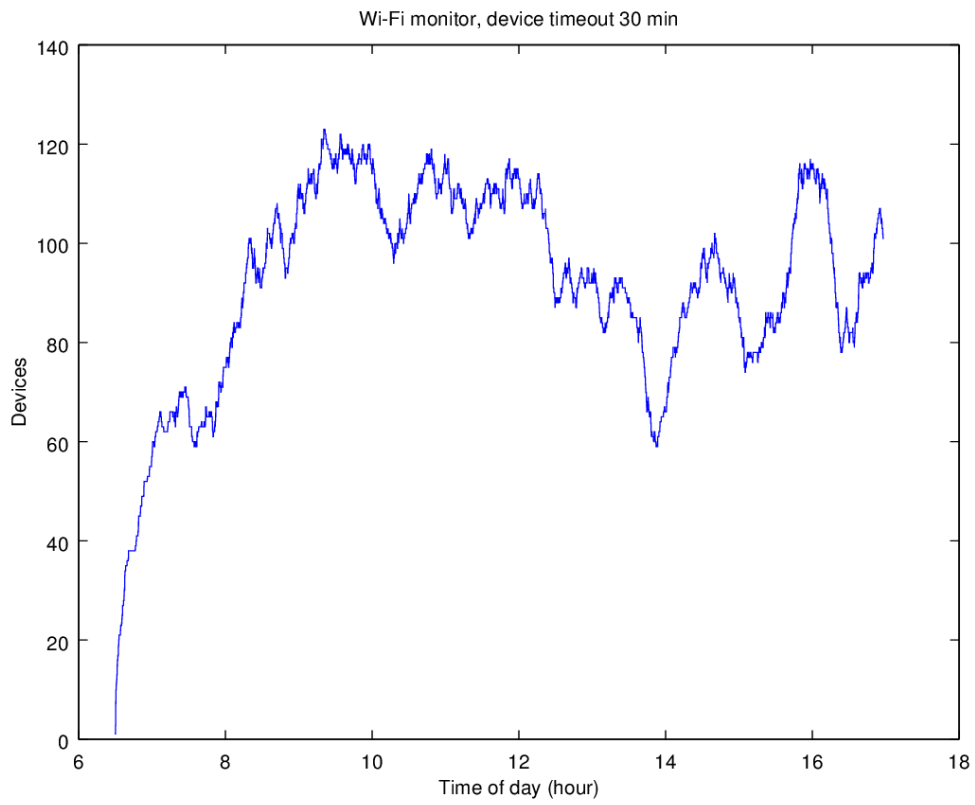


Figure A.2: The number of devices counted during a period of approximately 12 hours, with a 30 minute timeout.

## B Simulation graphs

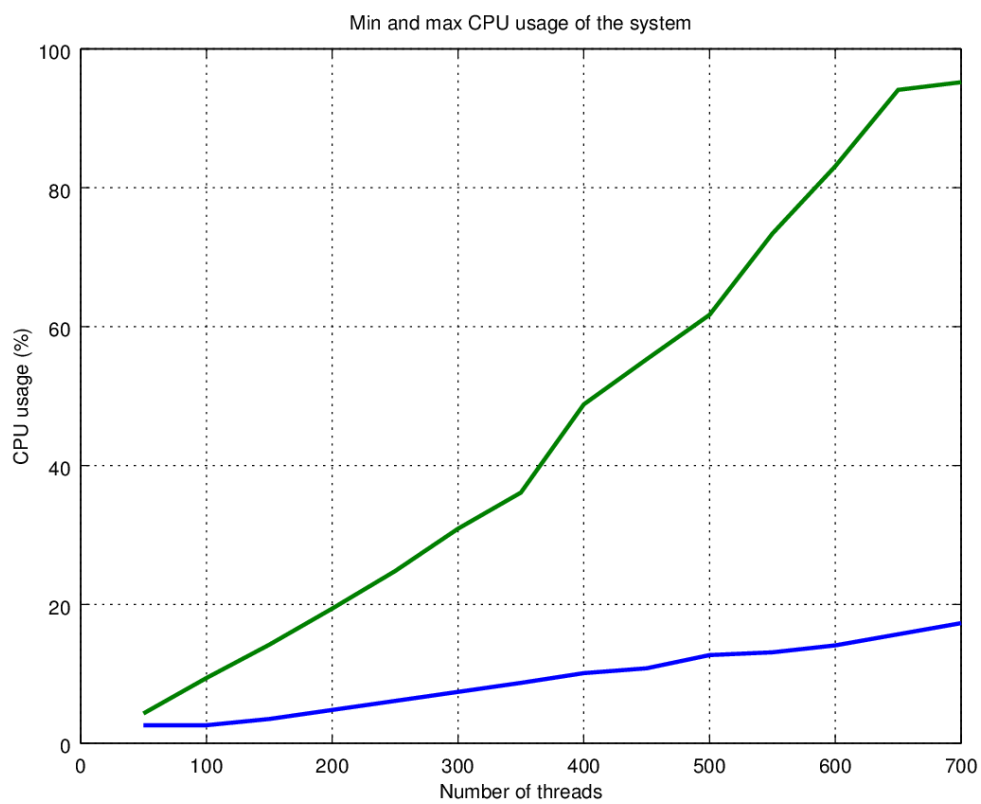


Figure B.1: The minimum and maximum CPU load of the system.

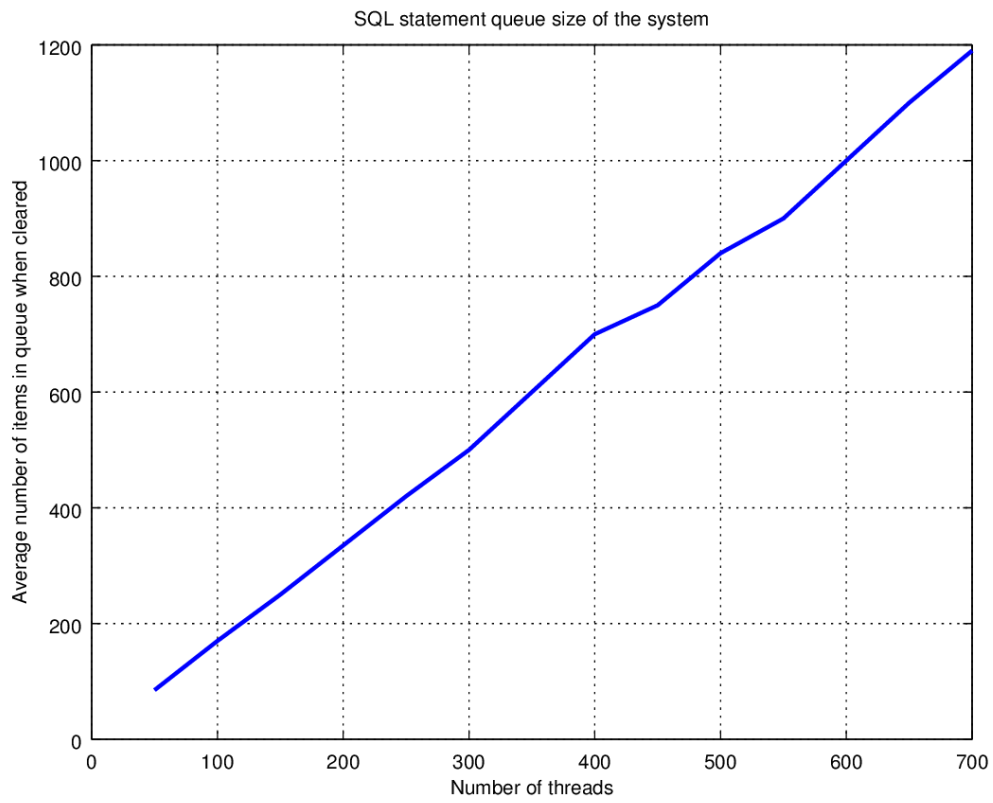


Figure B.2: The average number of items in the SQL statement queue when the queue was cleared by the system.

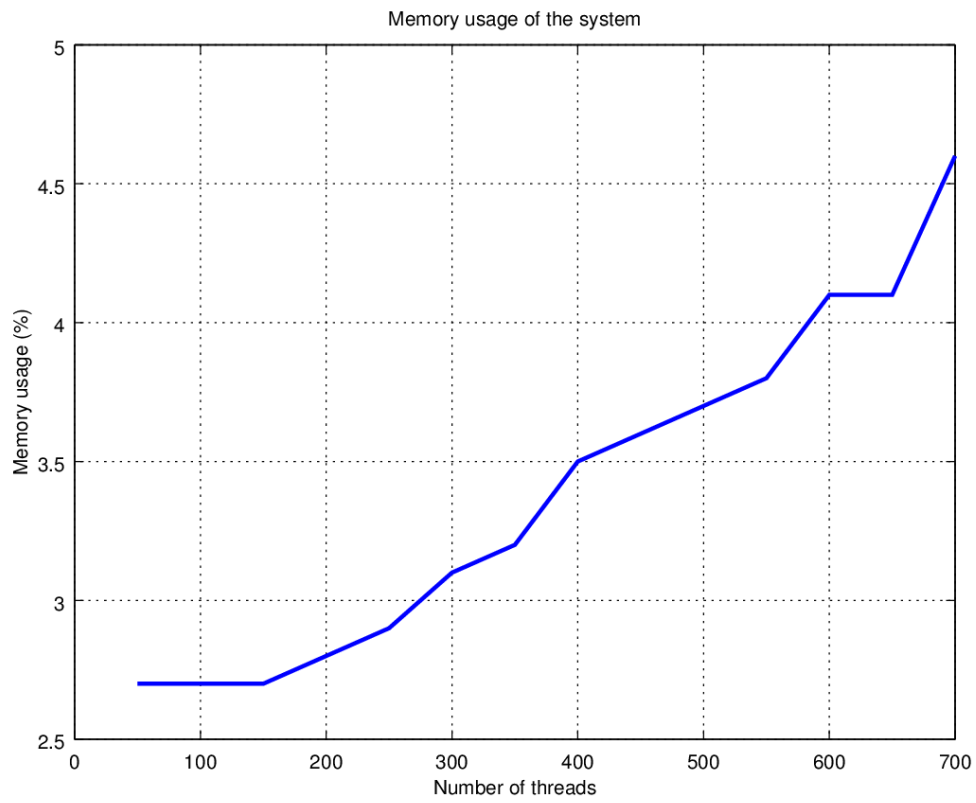


Figure B.3: The RAM usage of the system.

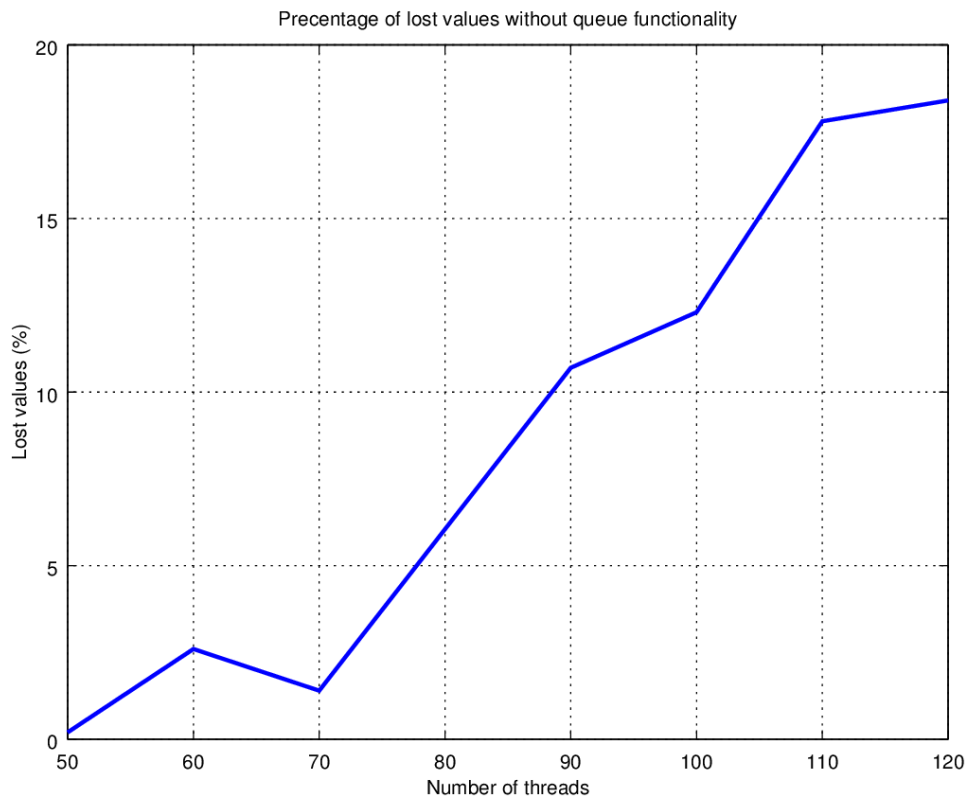


Figure B.4: Percentage of lost values without a SQL statement queue.

### C Case study graphs

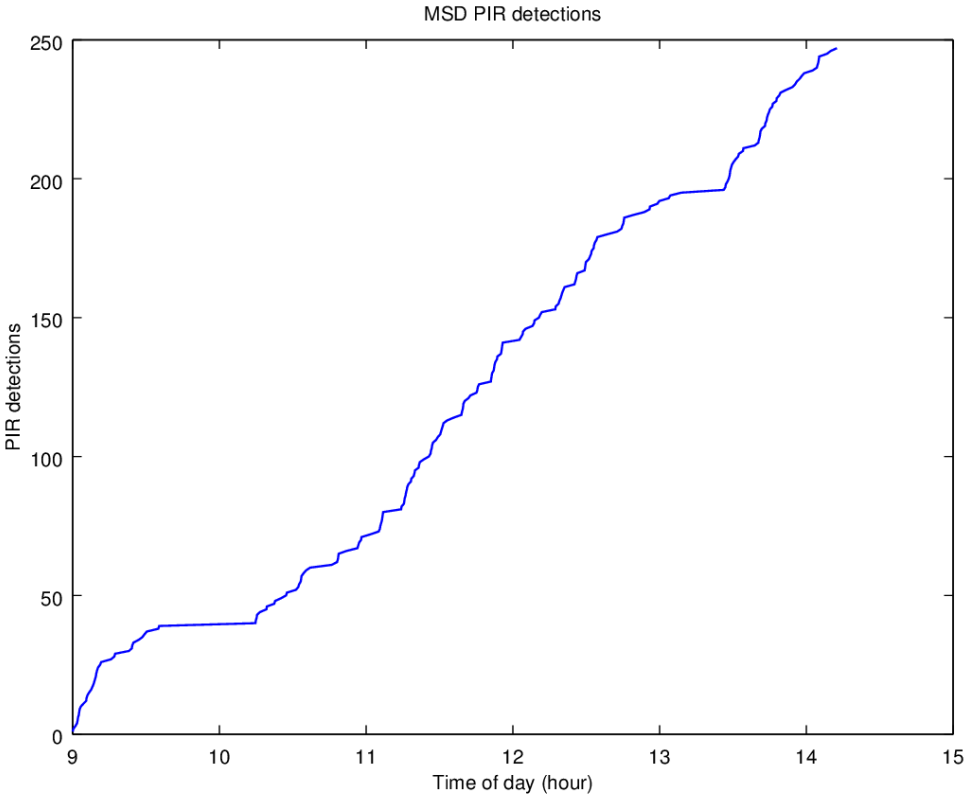


Figure C.1: Number of motion detections at doorway.

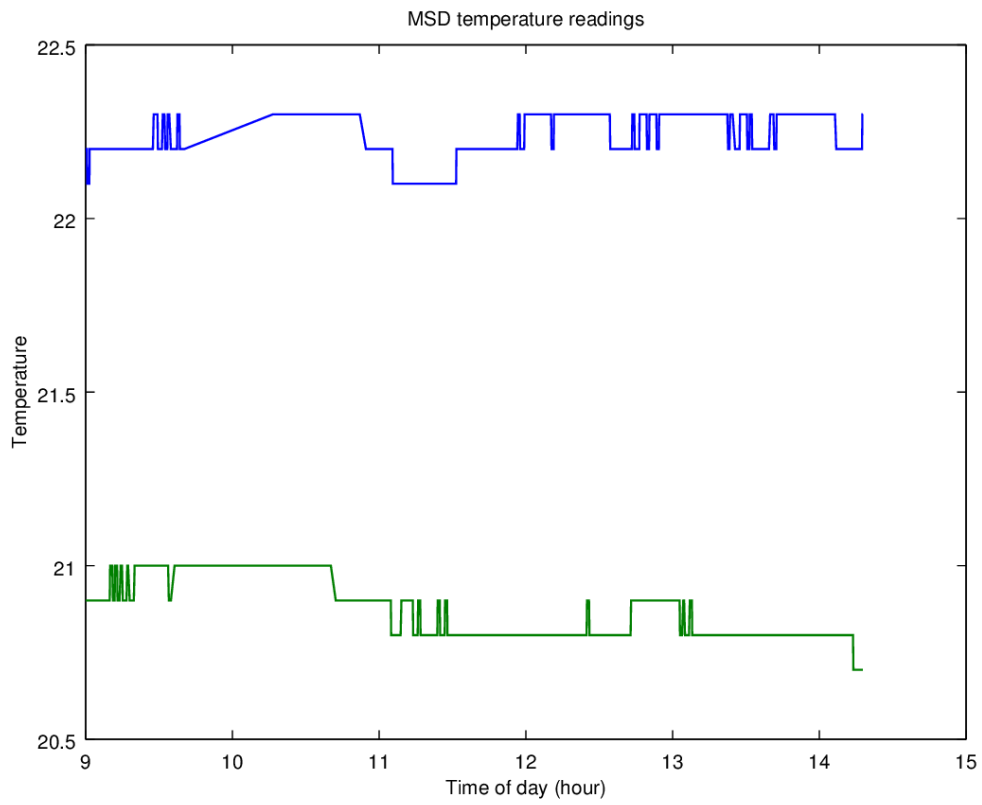


Figure C.2: Temperature (degrees Celsius) at floor level (lower plot), and at table level (upper plot).

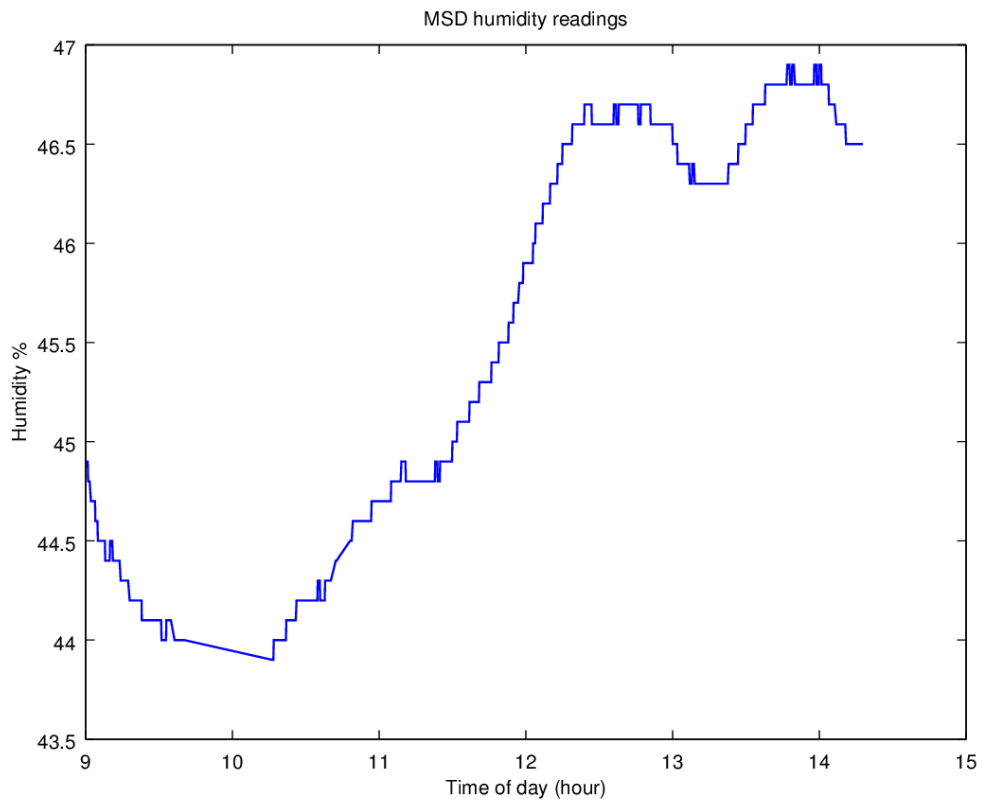


Figure C.3: Humidity level in the room.

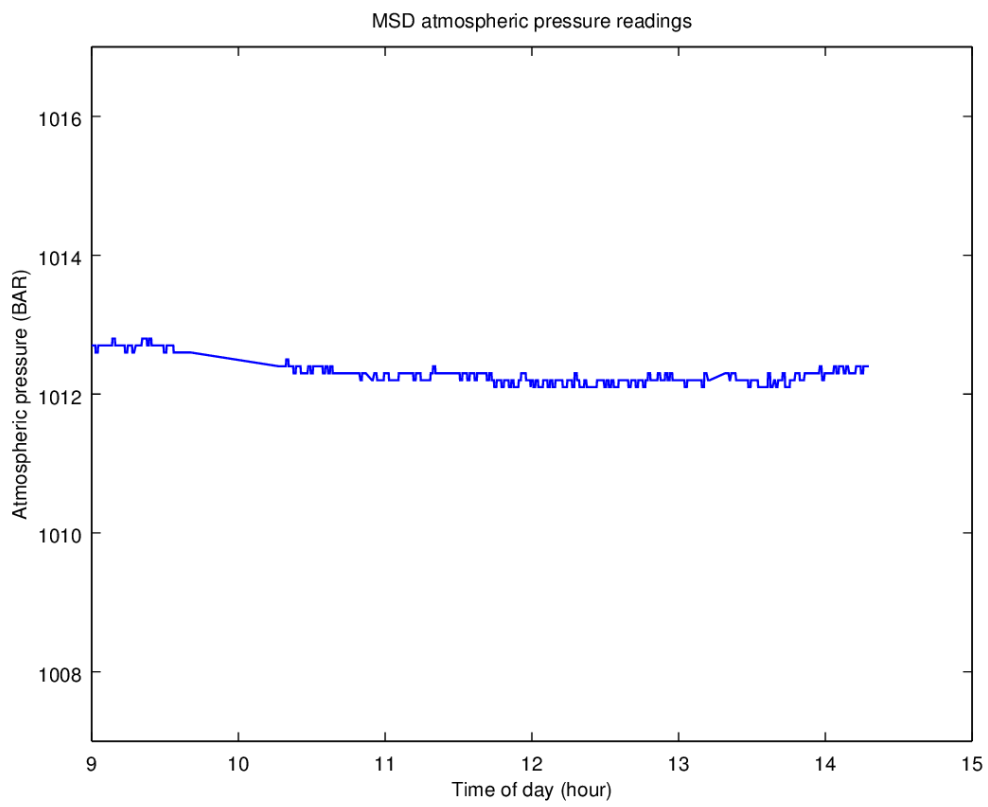


Figure C.4: Pressure in room.

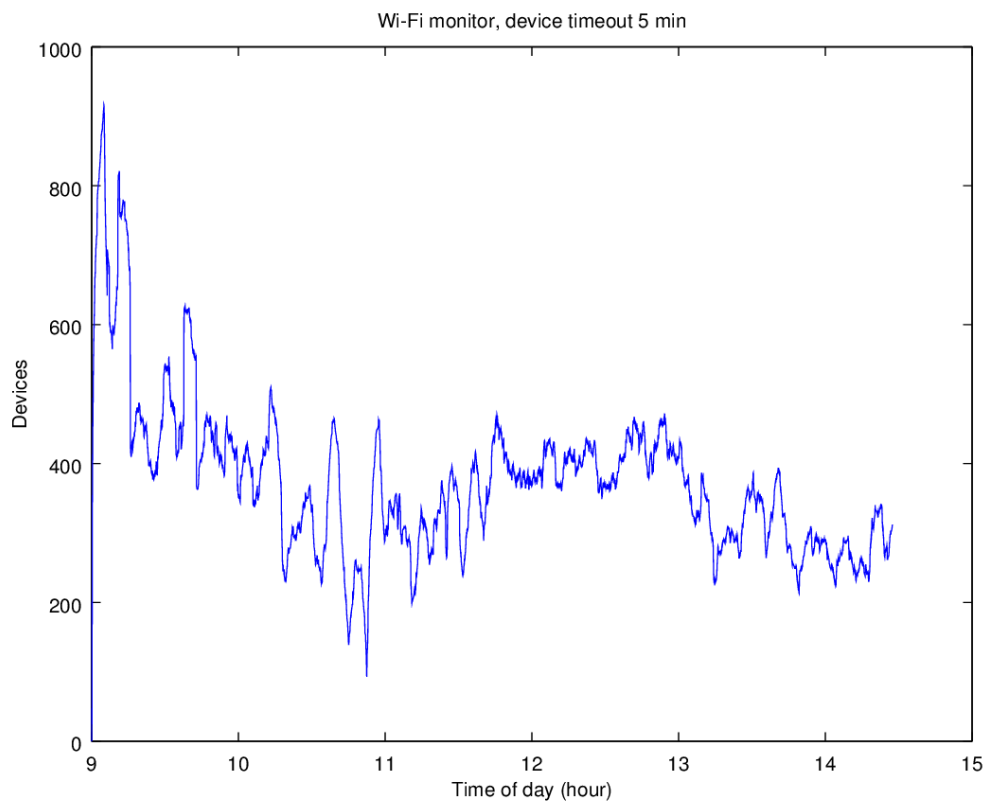


Figure C.5: Wi-Fi enabled devices detected by the Wi-Fi monitor.