IoT-Framework
Course Report
Project CS
(1DT054) - Autumn 2013
Uppsala University

Arias Fernández, José
Bahers, Quentin
Blázquez Rodríguez, Alberto
Blomberg, Mårten
Carenvall, Carl
Ionescu, Kristian
Kalra, Sukhpreet Singh
Koutsoumpakis, Georgios
Koutsoumpakis, Iakovos
Li, Hao
Mattsson, Tommy
Moregård Haubenwaller, Andreas
Steinrud, Anders
Sävström, Tomas
Tholsgård, Gabriel

March 10, 2014
Abstract

Project CS is a 20 week, full time course where students work on a project ordered by a customer. This year, 2013, the customer was Ericsson and the goal was to create a system that can manage, aggregate and visualize data streams from the Internet of Things. This report focuses on the course experiences and describes the methods and tools used while working on the project. There is also an adjoining product report which describes the technical aspects of our product.
# Contents

1 Introduction ........................................ 5

2 Resources ........................................... 7
   2.1 Project Group ...................................... 7
   2.2 Equipment ......................................... 7
       2.2.1 Furniture ........................................... 7
       2.2.2 Workstation ........................................ 7
       2.2.3 Servers ............................................ 8
       2.2.4 Miscellaneous ..................................... 8
   2.3 Expertise ........................................... 8
       2.3.1 Scrum ............................................... 8
       2.3.2 Technical ......................................... 8
   2.4 Tools .............................................. 9
       2.4.1 Linux Ubuntu ....................................... 9
       2.4.2 Git & GitHub ....................................... 9
       2.4.3 Rebar ............................................... 9
       2.4.4 Makefile ........................................... 9
       2.4.5 Text-editor ........................................ 9
       2.4.6 Travis ............................................. 9
       2.4.7 Ruby on Rails ...................................... 10
       2.4.8 Bundle ............................................. 10
       2.4.9 Rake ............................................... 10

3 Project Methodology & Organization ................. 11
   3.1 Scrum ............................................... 11
3.1.1 Keywords ........................................ 12
3.1.2 Roles ........................................ 12
3.1.3 Process ........................................ 13
3.2 Project Management .............................. 15
   3.2.1 Contact Person ............................. 15
   3.2.2 System Administrator ...................... 15

4 Development Process ................................ 17
   4.1 Overview ....................................... 17
   4.2 Quality Assurance ............................. 18
      4.2.1 Code Review .............................. 18
      4.2.2 Definition Of Done ..................... 18
      4.2.3 Pair Programming ......................... 18
      4.2.4 Automatic Testing ....................... 19
   4.3 Github Issues ................................ 19

5 Team ............................................... 20
   5.1 Fika ........................................... 20
   5.2 Rules .......................................... 20
   5.3 Activities ..................................... 21
      5.3.1 BBQ ..................................... 21
      5.3.2 Gaming Nights ......................... 22
      5.3.3 Thursday Pub ......................... 22
      5.3.4 Gokart in Kista ....................... 22

6 Sprint Breakdowns ................................ 23
   6.1 Sprint 1 ...................................... 23
   6.2 Sprint 2 ...................................... 23
   6.3 Sprint 3 ...................................... 23
   6.4 Sprint 4 ...................................... 24
   6.5 Sprint 5 ...................................... 24
   6.6 Sprint 6 ...................................... 24
   6.7 Sprint 7 ...................................... 24
   6.8 Sprint 8 ...................................... 25
6.9 Sprint 9 ................................................... 25

7 Conclusion ............................................. 26

7.1 What went well ....................................... 26
  7.1.1 Communication .............................. 26
  7.1.2 Tasks ........................................ 26
  7.1.3 Retrospective ................................. 27
  7.1.4 No team specialization ..................... 27

7.2 What did not go well ............................. 27
  7.2.1 Sprint planning ............................ 27
  7.2.2 Lack of requirement specification ...... 27
  7.2.3 Task breakup ............................... 28
  7.2.4 Lack of leadership ......................... 28

7.3 What can be improved .......................... 28
  7.3.1 More customer input ..................... 28
  7.3.2 Bigger work area ......................... 28

Bibliography .......................................... 29

Appendices ............................................. 30

A Individual Input .................................. 31
  A.1 Arias Fernández, José ......................... 31
  A.2 Bahers, Quentin .............................. 32
  A.3 Blázquez Rodríguez, Alberto ............ 33
  A.4 Blomberg, Marten ........................... 34
  A.5 Carenvall, Carl ............................... 35
  A.6 Ionescu, Kristian ............................. 36
  A.7 Kalra, Sukhpreet Singh ..................... 37
  A.8 Koutsoumpakis, Georgios ................. 38
  A.9 Koutsoumpakis, Iakovos .................. 39
  A.10 Li, Hao ...................................... 40
  A.11 Mattsson, Tommy ............................ 40
  A.12 Moregård Haubenwaller, Andreas ...... 41
A.13 Steinrud, Anders .................................................. 42
A.14 Sävström, Tomas .................................................. 44
A.15 Tholsgård, Gabriel .................................................. 45

B  Git Workflow ...................................................... 47

C  Coding Standards .................................................. 49
   C.1  Erlang .......................................................... 49
   C.2  Rails ........................................................... 50
Project CS is an advanced 30 credits course that is run each year at Uppsala University. The goal of the course is for groups of 10-15 students to develop software systems while using modern agile methodologies.

The actual system or product that is developed each year is decided by an industry partner. The industry partner for 2013 was Konstantinos Vandikas from Ericsson Research[1].

The product to be implemented was an IoT-framework (Internet of Things framework)[21] where the users can register sensors posting and polling streams of data as well as visualize, rate, aggregate and subscribe to streams. In addition, a website for this system was created, with an emphasis on user experience, providing the end user with a friendly and easy way to handle all these operations.

Internet of Things[21] is the concept that more and more devices are being connected to the internet thereby making them available for interaction and can be used for information gathering. All these devices normally provide functionality or data that can be useful to gather and analyze.

Teaching and assisting the course were Olle Gällmo and Muneeb Khan.
The student group also had technical help from Liam McNamara from SICS\[2\] as well as project methodology help from Mikael Lundgren from Levla AB\[3\].

This report describes in detail the various tools, equipment and overall methodology used during the development progress. For more technical details about the product developed please refer to the product report\[4\].
Chapter 2

Resources

2.1 Project Group

The group consisted of 15 students from six different nationalities. The Project CS team was given a room at Uppsala University for the duration of the project. The project room was located in the IT department buildings at Polacksbacken, more specifically building four, floor four, room number 4408.

2.2 Equipment

2.2.1 Furniture

The project room was supplied with a number of chairs and tables, three wall-mounted whiteboards and one movable. We also had access to a meeting/coffee room.

2.2.2 Workstation

Each student in the group was provided with a desktop computer, monitor, mouse, keyboard and corresponding cables for the hardware.
2.2.3 Servers

Two extra desktop computers were requested and kept in the project room. These were used for testing and demonstration purposes.

2.2.4 Miscellaneous

To be able to hold presentations, both for ourselves and to present the product to Ericsson, the room was equipped with a projector, pointed to one of the white walls in the project room.

2.3 Expertise

All students had different level of expertise in different areas, which helped develop a good atmosphere of learning for everyone. To aid, the project had expert help from the industry. Project CS is a course and the point is to learn new skills, thus it was encouraged to explore new areas.

2.3.1 Scrum

Very few people in the project group had worked with the Scrum methodology before (see 3.1). To help with learning Scrum, how the methodology works and how to use it fairly efficiently, the group had help from Mikael Lundgren from Levla AB. Mikael Lundgren has successfully helped many companies implement the Scrum methodology into their development process and did so for this project as well.

2.3.2 Technical

As technical expert, the group had Liam McNamara from SICS. He is one of the developers of SicsthSense, a system similar to IoT-Framework, that heavily inspired us. He provided a lot of help concerning how to design the project application, which tools to choose and various other smaller things.
2.4 Tools

2.4.1 Linux Ubuntu

During development all computers were installed with Ubuntu 12.04 in order to have a stable long-term support release.

2.4.2 Git & GitHub

For version control software, Git was used. The software is open source so GitHub was used to store the repositories, because GitHub offers free hosting of public repositories, online editing, issue posting and wiki pages.

2.4.3 Rebar

Rebar was used in order to make compiling, testing and documentation generation easier. Rebar is written in Erlang and is intended to be used for Erlang OTP applications. It supports Eunit testing, Edoc documentation generation and other Erlang specific functionality. Overall a very handy tool which simplified the work a lot.

2.4.4 Makefile

In order to make installation of needed software and running the application, a Makefile was constructed to automate such tasks.

2.4.5 Text-editor

Some people were using Eclipse along with Eclipse plugin for Erlang and Git in order to edit source files. Others were using Emacs, Vim or some other text editor.

2.4.6 Travis

Testing is very important part in software development. In order to make sure that testing was performed on all branches, Travis was used. Travis
is a webservice which performs automatic testing on GitHub repositories.

### 2.4.7 Ruby on Rails

Ruby on Rails is an open source web application framework, created by David Heinemeier Hansson in June 2004, which runs on top of the Ruby programming language. It is a full-stack framework which allows developers the creation of applications that retrieve and manipulate data from the own server database or from another source using an intermediary API (which was our main use case). Also, developers might render view templates if they are needed.

### 2.4.8 Bundle

Bundle is one of the Rails built-in command-line tools. The purpose of this tool is the management of the `gem` libraries and dependencies of the project. The flow it follows consists of checking if the installed gems are up to date. If they are, Bundle will stop the execution, otherwise it will download the latest version of the gem.

### 2.4.9 Rake

Rake is another important command line tool in Rails. It is focused on management of the local database, allowing the user to create and remove specific tables or the whole database and create timestamped migrations.
Chapter 3

Project Methodology & Organization

The group was expected to organize themselves and follow an agile methodology based on suggestions from Olle, previous years reports and Mikael Lundgren, project methodology specialist from Levla AB\textsuperscript{3}.

3.1 Scrum

The agile development methodology used in the course was Scrum\textsuperscript{5}. By recommendation of Mikael Lundgren, the whole group was divided into three smaller Scrum groups with five people in each group. It was decided quite early on that the groups would not have any special direction or area of expertise, rather every group should be able to take on any task. To facilitate this structure the groups were carefully divided based on interest and knowledge. The benefits of this was that any group could handle any task within the project. A drawback, however, was that cooperation across the groups was inevitable and that sometimes made it hard to keep track of who the tasks on the boards really belonged to.
3.1.1 Keywords

- Retrospective - A meeting at the end of each sprint discussing good and bad things about the last sprints.

- Customer - Owner of the project. Responsible for prioritising the user stories in the product backlog.

- Task - A small, but more or less self contained, set of work.

- User Story - A feature that the end system should have.

- Backlog - A set of user stories.

- Product Backlog - All user stories that the end product should have, from the customers perspective.

- Sprint Backlog - User (and/or developer) stories that should be done during a sprint.

- Standup meeting - A short meeting held each day to make sure development is coming along as it should.

- Planning meeting - A meeting before each sprint at which all the tasks for the upcoming sprint are set.

- Scrum master - Makes sure that the team follows the Scrum methodology. Leads the stand up meetings and protects the team from distractions.

- Master meeting - Meeting between the masters of multiple Scrum groups, in our case held daily.

3.1.2 Roles

Each group had a Scrum master, which could vary from sprint to sprint. The masters responsibility was to lead the stand up meetings held each morning at 9:15-9:30 and make sure that their team was working on relevant tasks.
Their role also included having an overview of what the other teams were doing and function as connection points between the teams. Each morning after the stand up meeting, the masters would have a meeting with only the three of them to check up on each others teams and make sure the sprint was coming along as planned.

In terms of development, apart from the rotating role of Scrum master, no one had any specific roles. Since none of the groups were specialised, there was flexibility on an individual level for people to work on what they wanted, even if that shifted during the project. Over time a few people ended up working with specific areas, some out of necessity because their skill set was relatively rare in the group, others out of interest.

### 3.1.3 Process

Scrum is an iterative process with short sprints (in our case typically two weeks each) after which the product should be presentable in some sense, without any tasks in progress. This makes it easier to change things during development, both for the development team and for the customer who gets a continuous and active role in quality assurance and the direction of the development. The method also includes daily checkups where each team member presents how they are doing, making sure that everything is running smoothly.

The tasks are generated from a backlog of features (also known as user stories). These were often of such a big scope that they were subdivided into large tasks before being divided further into actual development tasks. While the features and bigger tasks can be very large in terms of development time, the actual tasks being claimed were kept as small as possible. Preferably the small tasks would be less than six hours so that everyone should get to move at least one task from ”claimed” to ”review” per day. This makes it easy to keep track of daily progress.

Each sprint ended in a demo of the features that had been worked on. The customer, technical expert and other relevant outside people would be invited to these demo sessions, but they were as much for the teams benefit,
so the demo would go ahead even if no one showed up.
The other important part of ending a sprint was the retrospect meeting.
These meeting should focus on what had worked well the past sprint within
the scrum context, but our meetings typically included everything that had
either helped or hindered development. Everyone got to say one good and one
bad thing about the past sprint, and the we would discuss concrete solutions
to the issues. Each issue should have a solution connected to it before we
ended the retrospective.

After the end of each sprint a planning meeting was held for the upcom-
ing sprint. At first everyone was involved in the meetings, but it was very
inefficient. After a while the planning meetings were only held by the current
Scrum masters and the upcoming Scrum masters, meaning no more than six
people attended these meetings.

Figure 3.1 shows the Scrum process. While digital Scrum tools were

![Figure 3.1: The Scrum process.](http://commons.wikimedia.org/wiki/File:Scrum_process.svg)

considered, the decision to use sticky notes on white boards were prefered.
Each note represented a task that was claimed by a team member by writing
his name on it, by moving through sections on the board the status changes
from claimed, to review, to done. The reason to do it like this was in part
because it gave everyone an intuitive overview of how and what all the teams were doing, and because it feels satisfying to physically move a note when progress is made.

Figure 3.2: This is what our Scrum boards looked like.

### 3.2 Project Management

#### 3.2.1 Contact Person

The contact person (Steinrud, Anders) was responsible for distributing all information coming from the teachers, the customer and the technical expert to the entire group and also responsible for maintaining mail conversations with said parts.

#### 3.2.2 System Administrator

The system administrator (Carenvall, Carl) was responsible for our technical needs, for example computers, IP-adresses, cables, keyboards etc . . .

Since the university provided the necessary hardware and software, this role’s primary task was as a contact person between the university systems
administration and the other team members. Since the university needs to be sure that they are handing out equipment to the right people, it helped them a lot to only have one person that they needed to recognize and to contact in case information needs to be passed either way. For the team it helped to have a single person in the team to go to about hardware and software issues.
Chapter 4

Development Process

4.1 Overview

Based upon a agile approach, the high-level development process was chosen by the students. It could be described as follows:

1. Person P claims task X, P writes their name on it and moves it to the "Claimed" column of the Scrum board.

2. When task X is finished, the person who finished it moved the task to "Waiting for Review".

3. Someone claims the review process of task X by moving the task to the "Review" part of the board.

4. If there are any problems, the reviewer reports it back to the person who claimed the task who, in turn, fixes it and resubmits it for review. If there were no problems, the process continue to the next step.

5. When task X is properly reviewed the person who finished the task is notified and move the task to the "Finished" part of the board.
4.2 Quality Assurance

4.2.1 Code Review

In order to make sure that each task was done properly, someone else had to review it before it was added to the main branch and the task was considered as done. Another set of eyes reviewing the work of a task had a lot of benefits. Many bugs, logic errors and such problems were discovered quite early thanks to this process. There were drawbacks also of course, such as it being hard for someone not familiar with the task to be able to judge if the logic for something was proper or not.

4.2.2 Definition Of Done

We set up the following definition of when a new feature is completed:

- Coded - The actual code of the new feature is implemented
- Documented - The new code is properly documented
- Tested - New test cases are written, and all previous test cases still pass
- Accepted - The code has been reviewed by a third person, who makes sure that all of the above criteria are met.
- Delivered - The code has been merged with the develop branch

4.2.3 Pair Programming

Coding was done both individually and in pairs during the project. Both methods have merits and the combination of the two strategies had good results. For instance pairing up those with a lot of Erlang experience with those that had not worked with Erlang before was very good. Pair programming during the project meant that one person was coding and another was looking on what the other was coding. Often the pairs were taking turns whenever one of the persons in the pair had an idea or just didn’t want to
just sit by and watch for a while. The benefit with pair programming is that a lot of mistakes are caught very early because there is someone proofreading the code while it is being coded.

4.2.4 Automatic Testing

We had automated testing which was run using Travis [1]. If Travis reported failing tests, the issue had to at least be investigated before the code was merged with the main branch.

4.3 Github Issues

Towards the end of the semester the GitHub issues were used extensively to report bugs in the system, both from the team and from the customer. This worked really well to get feedback from the customer as he set up the system, tried it for himself and then found bugs that he could easily report to us.
Chapter 5

Team

5.1 Fika

Every Monday someone would bring fika. We had a schedule detailing when each person should bring fika. Also, if someone came in late they had to bring fika the next day that had no scheduled fika. Home baked fika is preferred.

5.2 Rules

- Base hours: 08-17
- Core hours: 09-16 (Mandatory attendance in project room)
- Daily standup: 9.15
- Other meetings: Can only be planned and occur during core hours.
- Lunch: 12-13 ± 15 min in 4411 or individual preference.
- Other breaks: Individual preference, at most 40 minutes per day.
- Flex: Allowed, flex has to be communicated and balanced every week.
- Tracking: Individual tracking of 7 hours worth of work per day.
- Weekends: Weekend work is not banned but not encouraged.
• Sick: Tell the team and stay at home.
• Be on time for meetings and presentations or bring fika the next day
• Ask for leave from the group beforehand
• Language: Use English only, even during breaks.
• Keep 4408 and 4411 clean at all times.
• Cellphones should be quiet.
• Calls and visitors should be handled outside 4408, unless during lunch breaks,
• Project visitors has to be planned in advance. Gaming, newsreading, social networking, music listening is selfregulated, bring it up if it is a problem.
• If someone sees a problem with the agreement or the behaviour of a group member they should bring it up as early as possible.
• Food is not allowed in 4408, except during lunch hours
• Water, coffee, tea, soda, fruit, snacks (those that doesn’t make strange noises) are allowed in 4408
• Weekly Fika: Monday at 3 pm
• One person has to provide Fika on Monday

5.3 Activities

5.3.1 BBQ
At the start of the semester when the weather was still warm he had a BBQ as our first teambuilding activity, this was appreciated by all the members of the team.
5.3.2 Gaming Nights

A few times during the course of the semester we played some boardgames with a portion of the team. This was something that was liked very much. We also had one night we spent playing some computer games, this was the evening after we had the BBQ.

5.3.3 Thursday Pub

Each Thursday some members of the team attended the Thursday pub at Uthgård, this was a nice way to relax after a days work and have some teambuillding. The night normally consisted of a few beers, some dinner and a board game. This was appreciated by the members that attended and is something that we highly encourage next years group to try.

5.3.4 Gokart in Kista

After we had our midterm evaluation in Kista we went go-karting in ”Kista gallerian” which Ericsson paid for and then had a dinner at the O’Learys bar also in ”Kista gallerian”. This was a night that was really appreciated by all the team members.
Chapter 6

Sprint Breakdowns

6.1 Sprint 1

The first sprint started with researching the parts of the system that we would need to implement and starting to design it. We also went through the code of the Sicsthsense system to get a good understanding of how these systems can operate.

6.2 Sprint 2

We felt that we still needed to do some more research so this sprint was also spent heavily on research, but parallell to that we started to set up things like the database and some basic front end web pages.

6.3 Sprint 3

This sprint was used to get a system up and running that had the same basic functionality that Sicsthsense had, as we decided to build our own system from scratch using Erlang instead of trying to extend Sicsthsense. We also decided to change the back end to Elasticsearch instead of using the database that is part of Erlang which was decided on the previous sprint.
6.4 Sprint 4

This sprint was spent with a focus on getting started with the Publisher/Subscriber system and working on getting a visualization of the data in the front end. How to add predictions was also researched and started on in this sprint.

6.5 Sprint 5

This sprint was spent on getting everything that Ericsson Research[1] wanted to show at their internal presentation working so it could be demoed, this meant that all work was focused on getting old features completely done and tested.

6.6 Sprint 6

This sprint was mostly spent on creating documentation for the wiki part of our Github[14] repositories, this so the system could be set up and run at Ericsson Research. Some work was also spent on creating aggregation of streams, adding more advanced features for searching, adding request checks in the back end and working on the polling system.

6.7 Sprint 7

The priority in this sprint was mainly on polishing features which had been started in previous sprints but were not completely finished, such as the subscription to streams, the user profile, the ranking system for streams and also some minor fixes regarding the navigation and exposure. However, two new features were developed: the smart creation of streams and a basic trigger mechanism.
6.8 Sprint 8

This sprint was spent on finishing most of the features that still needed work, these included triggers, alerts and virtual streams. A lot of work was also spent on getting the front end to be more uniform, finding bugs and fixing them.

6.9 Sprint 9

Finally, the last sprint consisted mostly of writing the product and course reports. However, some members of the project spent time on fixing minor bugs found and other ones prepared the presentation with the demo.
Chapter 7

Conclusion

To start with we think that the project was a success with regard to the system, we got most of the features requested by the customer into the system and it appears to work well for the small amount of streams, resources, users and data that we have tried in the system.

7.1 What went well

7.1.1 Communication

Despite the team consisting of 15 people divided up into three groups there was good communication between people working on related tasks. The reason for this could be that all the members were located in the same room. The use of Scrum also helped a lot with the administrative things that need to be handled when working in a project with such a large group.

7.1.2 Tasks

The breakdown of features into independent tasks really helped a lot with having people working effectively, as they could focus on the task at hand and did not have to consult with anyone else when doing the task.
7.1.3 Retrospective

The retrospective were also good as they help improve the work process for the next sprint. This meant that we could bring up things that we thought were hindering the development process and find a way to solve these issues.

7.1.4 No team specialization

We decided to have mixed teams, meaning that no team specialized in one area, this lead to that people had a chance to work with what they wanted and could switch area if they could handle the tasks in the new area. This meant that we could switch focus from the back end to the front end when we were getting closer to a finished back end and everyone could help out on tasks that needed to get done.

7.2 What did not go well

7.2.1 Sprint planning

This was something that was improved as the project went along but it was still the part of Scrum we had the most problems with. The main problem was the time estimation, we also had the problem of being too many people in the planning and not getting anywhere. The problem was solved towards the end by only having the current Scrum masters and the next Scrum masters of each team in the planning meeting.

7.2.2 Lack of requirement specification

The customer did not have a consistent view of the wanted functionality through out the development process, causing additions and priority changes in the backlog. There was also a lack of presence from the customer at the planning meetings forcing us to guess what he wanted next.
7.2.3 Task breakup

We were unable to plan ahead properly and break down big tasks into smaller ones, and thus, creating quite vague tasks. As a result, people had a tendency of doing more than each task required.

7.2.4 Lack of leadership

As the customer also had the role of project owner, we lost one step in the chain. Usually the customer wants a product and the owner sees to it that it is feasible, by creating a requirement specification that reflects the customers needs. This gives the product owner sort of a leadership role as he/she have good knowledge of the project and can steer it towards the goal in situations that might arise. This course is deliberately made so that no one of the students would be a project leader, only taking the role of Scrum master during sprints, which leaves sort of a hole in the chain of command.

7.3 What can be improved

7.3.1 More customer input

We would have liked to have more input from the customer in the planning meetings, for most meetings the team needed to guess what the customer wanted next. This is a problem that naturally arises when the customer have more duties than just be customer to the project but trying to plan out so the customer could attend a planning meeting every few sprints would be good.

7.3.2 Bigger work area

The workplace was small for such a large group, the setup we used this year worked for us but it was quite crowded and we did not have a good place to have meetings when we all needed to attend. A bigger workspace would have been better.
Bibliography


Appendix A

Individual Input

A.1 Arias Fernández, José

During the two first sprints I was doing researching work, working with the conceptual model, analyzing the SicsthSense java files and reviewing the analysis done by other team members. After that I helped with the mockups with the aim of performing a paper prototype. In addition, I worked with the databases, discussing which one should be the best option and I did a RESTful workshop with Alberto.

In the third sprint I supported Andreas during the installation of ElasticSearch and the definition of the mappings and I researched erlang APIs in order to use ElasticSearch (ElasticSearch, Tirexs). After these tasks, I started carrying out back end tasks.

In the next sprints (from 4 to 7) I was focused on back end tasks. At the beginning (sprint 3 and 4 mainly) I did mostly pair programming with Andreas since it was my first time programming with Erlang and functional programming but in the following sprints I mixed pair programming, during tasks such as ranking and phrase suggestion, with solo-programming during tasks such as subscription and multistream creation.
I was the Scrum master in the sprint 8. My tasks during this sprint were some bug fixing and small improvements, because in the back end, all the functionalities were implemented and we only needed to do some fixes. I also searched tools for performance monitoring such as SPM (Scalable Performance Monitoring) from sematext.

Finally, in the last sprint I worked on the course and project reports.

A.2 Bahers, Quentin

During the first two research sprints, my task was to look for web frameworks we could use for both the back end and front end of our application. I especially focused on Chicago Boss and Webmachine, two frameworks written in Erlang. We ended up choosing Webmachine for building our API but we were not really convinced by any Erlang web frameworks to develop the front end. Having some knowledge about Ruby on Rails, I gathered some information about it, to demonstrate that it could be a viable solution. During this research phase, I also spent some time helping Mårten setting up Travis, our continuous integration tool.

Once the research sprints were over, I began to work on the front end. I first developed a full authentication system, so that users can register to the website, sign in and sign out. I then did some pair programming with Alberto. Our task was to connect the preliminary version of the front end with the back end that had been developed.

In the middle of the project, Ericsson wanted us to present what we had implemented so far. I then devoted time to fix various bugs that had been introduced as well as improving the overall design.

During the second part of the project, I contributed to the implementation of the search, sort and subscription functionalities, among other things. I also pair programmed the trigger feature with Gabriel. Lastly, I helped
A.3 Blázquez Rodríguez, Alberto

The first couple of sprints were mostly dedicated to the project’s kick off. Although there was a high degree of uncertainty, we tried to understand the customer goals and needs, as well as the meaning of the Internet of Things concept, in order to figure out what kind of system we had to implement. Also, I tried to help the group giving a workshop about Git concepts and commands and another small lecture about the RESTful APIs.

I started to work eventually in the front end with Ruby on Rails in the third sprint. Since we received a preliminary version of the system’s wireframes, the UX started to be a key aspect in our project. Hence, I participated in the group responsible of the evaluation and improvement of the wireframes.

Regarding the system features, I was able among Quentin and Anders, to implement the Resources and Streams with establishing the parent-child relationship between both elements and integrating the front end with the back end, which was a really hard task due to the high unstability of the API. In spite of doing a good demo, due to the new feedback provided by Ericsson Research, in the sprint 6 we carried out an urgent replanning of the tasks in order to satisfy his needs (this replanning is called silver bullet in the SCRUM community). This way, we could readapt our system removing the Resources and making the Streams the core element of the project. Moreover, I worked in the UX enhancement of the stream forms as well as in the smart creation of multiple streams, using the same resource as reference and supporting autocompletion.

In addition, I was one of the 3 Scrum masters of the class in the sprint 7. As the last sprint demo was not quite successful, we encouraged the group to
keep polishing and ending the remaining features rather than starting new functionalities, obtaining a better result than in the previous demo.

Finally, during the last couple of sprints I kept improving and fixing small bugs and writing documentation regarding technical details about the system’s description and the project’s background.

A.4 Blomberg, Mårten

In the beginning, when we weren’t really sure of where to begin, I started working on establishing a workflow and researching our tools. I wrote guides of how to work with Github, how to integrate Git with Eclipse and setup our continuous integration tool Travis.

I was Scrum master for the second and third sprint which meant a lot of managing and adapting our development process so that everyone had a clear understanding of what we where doing and that everyone felt an evenly distributed workload. Apart from that, I spent some time setting up a server with SICSth Sense which we could work with and creating some basic web-pages for handling our systems data.

For the upcoming sprint, which was our first real implementation sprint, I spent time in meetings regarding system architecture as well as researching RabbitMQ and pub/sub systems we could use in our system. I also started researching D3.js and created a first draft for a graph to visualize our stream data.

In the sprint preparing for the presentation at Ericsson, I was working on making everything work for the demo. I worked on GUI improvements as well as implementing the live update feature and integrating it to the front end.

After the presentation we reworked our internal representation of streams which meant a lot of time was spent rewriting and refactoring code. We cleaned up our code and made sure our back end API was consistent. Li Hao started working on a polling system so I set up a couple of fake resources we could use for polling. I also implemented graphs to display as backgrounds.
of the streams in the search result page. In the next sprint, these background graphs were scrapped. Instead I reworked the search result page and implemented a multiline graph which could display the same information.

The last sprint before winter break was spent mostly bug-fixing and improving the graphs. The last sprint of the project I spent preparing the end presentation and demo. I also did some bug fixes as well as improving the GUI to fit our demo. Andreas, Anders and I held the end-presentation.

A.5 Carenvall, Carl

My role as a systems administrator meant that it was my responsibility to make sure that we had the necessary hardware and software available to the team members. The central part of this was to be a contact person between the group and the university’s systems administration.

I was the Scrum master of Scrum group 1 during the first research sprints and the first development sprint. Since a lot of that time went into understanding the project at hand, this role mostly boiled down to making sure everyone had things to do and keeping the stand up meetings concise.

My primary development role in the beginning was as part of the UI design team. This turned out to give some new revelations about the overall project, especially since this was in the beginning of the project. After that I did some development in the back end system before starting on predictions. This took my focus for a significant part of the project. While some of the others helped out with both research and implementation, I was the team member working on it all the way from start to finish. While predictions took a lot of time to get working properly and stable in the back end, I also did some other minor back end work here and there.

During the last couple of sprints I was mainly working on the front end. Since I was not well acquainted with the technology for the front end, I was not able to handle major features of the home page, and some of the time was spent just learning new things.
A.6 Ionescu, Kristian

During the first sprint, I was researching about IoT and development tools, I started working on the initial user cases and user stories together with Anders Steinrud. We had discussions with the rest of the team about how we were going to create a fluid, modern UI that would be tailored for the target audience without sacrificing graceful degradation. We also performed some user interviews, trying to ask people what they would like to see in a service such as the one that we are developing.

I then went on to work with some back end related tasks. Since we were set on using Erlang as our main language for the back end we needed to find a restful API that was developed in Erlang for use with our services. There were not many options to choose from and I found that the restful API for Erlang called “Webmachine” was optimal for our project because of its portability and its understandable documentation. My task was to create a module using the Webmachine API to be able to use methods such as POST/PUT/DELETE on JSON objects with a URL hierarchy that we had previously defined during team meetings.

I also decided to contribute to the pub/sub system. My task was to create a websocket module that would interact with the pub/sub system in order to update data points for a stream in real time on the front end. This was done using Node.js. Gabriel Tholsgård was implementing the back end features for the pub/sub system, when this was done my work was used together with the stream graph visualization that Mårten Blomberg was working on.

My contribution to the project was focused on front end related tasks during the late stages of the course, I took the chance to familiarize myself with Ruby on rails and Coffescript. I was working mostly with the implementation of the map API and developed different modules for filtering, interaction and visualization. I was also working with filters and advanced search in the search window.
This was done together with Mårten Blomberg as he was redesigning the search window. During the last sprint, I was refactoring work done in the searches window and the map API. During this sprint, we were also focusing on fixing bugs on different parts of the project.

A.7 Kalra, Sukhpreet Singh

In the first few sprints, I learned basic things about our project and researched about the database, we were gonna use. We had a lot of meetings regarding the architecture of the system. I learned basic functionality of ElasticSearch. I researched about Erlang Web interfaces as well and JInterface.

In the following sprints, I started with learning basic Ruby language and Rails environment. I researched on the possibility of having a second database for access control mechanisms. I developed a layout for a basic Search page and implemented search functionality. As I had some UI experience and wanted to learn more about that, I helped creating wireframes and also went for the first UX meeting on 24th October from which we got very little response because of time constraint. Then we had our presentation and lot of inconsistency was handled. The Front end and Back end was made stable for the presentation day.

In further sprints, we dropped the idea of having resources on top and having streams as the main thing with which user will interact. WI was mostly working in Front end. We had our second UX meeting which turned out to be very useful. I also worked on sorting of search page and handled some inconsistency. I was also Scrum master for one of the sprints, worked on creating User Profile Page and handling other small issues related to user account pages. I shifted the functionality of making username as primary from default auto-generated user_id.

In the last two sprints, I worked with Georgios on implementing Virtual Stream in Front end. I did a lot of testing and debugging in Front end. The Front end was made stable and all the features were tested, since it was the last development sprint. I made an Android app for the live demonstra-
tion of some features, which turned out to be useful during our sprint-end presentation. Then after Christmas break, we started writing reports and documentation, so no development was done in this sprint. I started with writing my personal report and then moved on to writing the Course and Product Report.

A.8 Koutsoumpakis, Georgios

In the beginning of the project we started researching on how SicsthSense works. I did some research on which database we could use for our system, having a look at MySQL and PostgreSQL. Originally we decided to use MNESIA database so I created a basic API for creating/deleting users.

Later, we decided on using Elastic Search as our main database management system, so I had to read up on its features and understand how to use it. I rewrote the restful API for the user to use elastic search instead of the MNESIA database and I also implemented in the API the ability to search for users.

On the following sprints, I made the back end delete the datapoints of a stream when that gets deleted. Furthermore, I worked with Tommy on a feature for completing data automatically when the user is filling in information for a resource (i.e when specifying model to fill in automatically the manufacturer, as well as to create automatically its streams), based on previously created resources. Additionally, I implemented phrase suggestion on fields (like manufacturer), a mechanism that we were planning on using for autocompleting text when the user started writing on a field, but we later decided to not use this feature.

Later on, I was moved to the front end since they were lacking people. I had to read a bit about Ruby on Rails but I got used to it rather fast since I had a lot of help by the other guys. Firstly, I created the contact us page on the front end and then I implemented search autocompletion in the front end. Then I created the security and terms pages. I added support for the polling system in the front end, and, in the final development sprint, on
which I was the Scrum master for my group, I implemented together with Sukhpreet the front end for virtual streams. After that I worked on finding bugs.

A.9 Koutsoumpakis, Iakovos

In the beginning of the project I spent time understanding how SicsthSense works and researching on tools we could use to organize the team, like iceScrum. We decided that there would be no benefit of using any. Also, contributing to the team’s study on suitable databases, I researched whether MongoDB would serve our needs.

Later on, I created a user management module for registering users that would work as an intermediate level, however, as during the next sprint we decided to switch to Elastic Search, it became obsolete.

After studying Elastic Search, I started implementing a restful api for registering a resource, however due to an unfortunate duplication during the preplanning, some others worked on it too. So I just added _search support for them and then started working on an API for creating data points with support for different ways of retrieving data, like fully functional GET queries with timestamp range, POST and _search.

Then I spent time trying to reach a conclusion on how virtual streams are meant to work and experimenting on forming the basis of the mechanism that would create the virtual stream as an aggregation of the history of several streams over time. Soon I started implementing the API module for creating, returning, searching, updating and deleting virtual streams. I updated the users, streams and data points modules, so that all virtual streams and virtual stream data points will be deleted when their user and/or virtual stream is deleted. I also updated predictions to support virtual streams. Then, I collaborated in merging with the code that dealt with aggregating the new data points posted to the streams involved in a virtual stream. This proved to be tricky and time consuming and required a lot of debugging.

Next was additions, improvements and thorough testing of virtual streams, as well as introducing field restrictions for updating and creating them. Fi-
nally, I updated all the names in the field_restrictions module and the corre-
sponding ones in data points, resources, streams to improve readability.

A.10 Li, Hao

In the first two research sprints, I first researched on the SicsthSense frame-
work, wrote the document analyzing the code, and drew the architecture
overview of the SicsthSense engine. This analysis was supposed to help us
design our own product. Then I also researched on YAWS, to see how it
could be used to build our own RESTful API. I wrote a spike to test its
functionality. However, since the documentation of the YAWS was bad, we
decided to use WebMachine later.

In the coming development sprints, I was the Scrum master in the 7th
sprint. I implemented the polling system to push or fetch the data from
external resources. I also added new interfaces to the RESTful engine to
support the polling system, even though some strange bugs occurred in the
unit test cases, I still made it work. Meanwhile, I also made a spike to use
RabbitMQ, researching how to build the Pub/Sub system. Then I solved the
problem how to connect pub/sub system to the webpages. I solved this issue
via STOMP and websockets, so we knew how to implement the live update.
On the other hand, I also implemented the scrolling pagination in the search
result page, and solved some bugs in the front end.

A.11 Mattsson, Tommy

In the beginning of the project we weren’t sure of what product we were
supposed to produce, so we researched many different things. The things I
was researching was primarily concerning which programming language that
could be good to use for the project. Since I have a lot of previous experience
with Erlang I knew that it was a good candidate, but maybe there was a
better choice for various reasons. I held a couple of short talks about different
programming languages’ benefits and shortcomings when performing various tasks.

Eventually we chose Erlang so I set up the project structure with Makefile, Rebar, install script. Some were setup in the beginning of the project and some was set up later when we found there was a need for it, such as the install script.

During the middle part of development of the project I developed various tasks, the biggest more isolated ones was a library module for manipulating JSON objects, functionality for setting configuration options for our system using a configuration file. Together with Georgios Koutsoumpakis I developed a feature for autocompleting data based on previous submitted data from other users.

Towards the end of the project I worked on implementing triggers together with Tomas. He developed the major part of that and I helped out with parts of it while also further working on setting configuration options for our system and improving the startup script initially written by Kristian Ionescu

A.12 Moregård Haubenwaller, Andreas

I spent the first two sprints getting more information about IoT and researching possible technologies to use. While doing the research I found Elasticsearch, and after some group discussions we decided to use it as our search engine and our data store. With some design and overall architecture choices made the development was started. In the third sprint I worked with Elasticsearch and how to save our data and how to represent resources in the back end.

After the first three sprints I mainly focused on Elasticsearch features, the representation of resources and predictions for streams. I also worked with API consistency and naming convention. I was the Scrum master for sprint four and five. I spent most of my time during sprint five to work with the presentation since I was one of the presenters.

During sprint six I worked with fixing the error handling, API consistency, implemented advanced search with filters in the back end, implemented
phrase-suggestion for the search and I also fixed a recurring problem with predictions. The sprint was long, but I got a lot of work done.

Sprint seven was the last sprint for developing new features in the back end. I implemented user ranking, triggers, multimget for streams and I changed the user representation. I was also tasked with creating streams with data that we could use. I found some online APIs that could be polled and created a script to add streams that would provide real data.

Sprint eight was the last development sprint. I spent a lot of time bugfixing and improving things, I also researched collaborative \LaTeX tools and started working on the report.

In the last sprint I spent some time on the report, but most of the time I focused on the presentation and the demo.

In summary, I would say that I mainly worked with Elasticsearch, searching, ranking, predictions, phrase-suggestion and triggers, all of which are back end related. Beyond this, I was also involved in almost all of the presentations and demos.

A.13 Steinrud, Anders

The first two sprints were spent on research and architectural sketches on my part. Along side this I spent my time working as the contact person, there were a lot of unclarities that needed to be dealt with and design choices being evaluated with the technical expert.

Around the third sprint the architecture was kind of set and I started to develop and maintain the wireframes for the front end, which resulted in a lot of meeting time.

Shortly after this it was time for the Ericsson presentation which I mostly worked on the presentation design and topic layout for. During the sixth sprint I was the Scrum master and it was also the sprint where we introduced the silver bullet (a critical change throughout the whole project or its heading), a lot of work went into understanding exactly what the erroneous assumption that we had designed from was, and from that form a new heading of the project, finding out what needed to be changed and what we could
After the silver bullet I started to work more and more in the front end, especially with the feature to create several streams from a common resource, I also helped Andreas with collaborative Latex tools.

I worked in the layout of the reports, deciding on which topics would go where, writing my individual part and the contact person text.

In summary I would say that in the beginning I put a lot of time in architecture and wireframing, in the middle of the project I had started to help other with their tasks, mostly front end and updating wireframes, and in the end I worked more individually. The role of contact person took maybe a forth of my time daily with was the main factor to why I tried to help on tasks instead of claiming new during the beginning and middle of the project.

As the contact person I was the contact point towards the course teachers, customer and the technical expert from the group. The job did take quite some time each sprint, here are some of the tasks:

- Keeping external conversations up to speed, delegating tasks that the customer introduced or just finding answers to various questions (later on we introduced a system where bugs and feature requests were added by the customer directly to GitHub via their Issues interface).

- In the sprint start: The sprint user stories should be sent to the customer to be accepted as sprint goals, this usually went smoothly but sometimes it needed some discussions with the Scrum masters and the customer.

- Decide on demo time in collaboration with the Scrum masters and make sure all interested parts were invited.

- At each sprint end the result of the retrospect should be sent to the teachers, including solutions on all the negative comments that arose.
A.14 Sävström, Tomas

For the first two sprints I spent most of my time with understanding the code from Sics\textsuperscript{th} Sense, researching what database to use, setting up the database and being part of meetings to discuss the architecture of our system. I was also part of creating and holding the presentation for the first year students in information technology and computer science. During the first sprint I was also the Scrum master.

During the third sprint I was part of more meetings to decide the architecture, I also worked with Elasticsearch to set up ‘rivers’ to get the data we got from SICS into Elasticsearch and deciding the index structure we used. I was also part of the meeting to decide the URI structure to use when communicating with the API. The second week i spent creating the first version of the streams.erl file where I implemented basic CRUD and also a basic search.

During the fourth sprint I worked more with the streams.erl file, first to fix comments that i had gotten and then to add checks for a parent-child relationship between resources and streams and enforce this on deletion. I also refactored the code in the API files and put reused code in a new module. I was also part of discussing virtual streams and starting on the presentation we were going to hold on the 1st of November. During the fifth sprint I created a module to add dummy data to our system and the rest of the time i spent on working with the presentation that we held on the 1st of November.

During the sixth sprint I started by writing documentation about the API in our GitHub-wiki. Then I worked with adding timestamps to data points and creation date for streams and resources. I also added field restrictions for updates and creations and also checks for only accepted fields in JSON objects sent to the API. The last part of the sprint I spent with changing the back end to change the relationship between streams and resources that we misunderstood from our customer. Also fixed an error in the back end that was found by our customer and reported in the GitHub issues.

During the seventh sprint I started with merging the features that had
been added during the last sprint to the changed back end. I also changed the back end to have the username of a user be the ID that was used in the back end and allow data points to be owned by a virtual stream. The rest of the sprint I spent implementing triggers with Andreas, we got this done during the sprint. The last implementation sprint I spent making trigger processes start when the system starts if triggers are in the system and adding so the triggers can trigger on virtual streams and also fixing small bugs in the front end.

During the last sprint I worked mostly on the product report writing some parts of the system description, the background and the back end part in testing, I also added some parts under the functionality of future work. In the course report I wrote most of the team chapter and wrote part of the conclusion chapter. I also helped review and change parts of the product report and the course report.

A.15 Tholsgård, Gabriel

Throughout the project I was Scrum master for group three four times more specifically in sprint one, five, six and eight. I was involved in several decision meetings and helped planning a few sprints. My main development was the pub/sub system and virtual streams, but also helped with various other things in both front end and back end.

I started my work by researching Sicsthsense and making a running local Sicsthsense server as well as defined the database structure that Sicsthsense was using. I also started on creating db_api.erl using Mnesia for creating, reading, updating and deleting streams. I also had presentation of the project to the first year students.

I continued by taking part in a meeting were we decided the URI structure for the Webmachine RESTful API. I read up on and how Elasticsearch could be used and explored creating indicies and attaching a mapping structure to it. Later I defined the mappings in elasticsearch for groups, streams, resources, users and virtual streams. I also worked together with Li Hao to create a skeleton for the polling system. I also started researching about
possible pub/sub systems to use comparing architectures using sockets as the base for communication instead of using a broker. We first decided to use ZeroMQ, brokerless, and started to design the architecture from that, but changed later to RabbitMQ which have a broker. I also attended a meeting on the architecture of the pub/sub system and decided how it should work, implemented it and connected it to a browser such that they are able to subscribe.

I talked in the presentation on 1st Nov 2013 in Ericsson. Later I helped Kristian Ionescu solving issues with web-sockets and we came up with the solution of namespaces.

I wrote documentation about the pub/sub system and the life cycle of data points on the GitHub wiki. In addition, I had a meeting about virtual streams with Jacob, George and Liam, from SICS.

I helped Li Hao write some helper functions for the polling system and also worked on making virtual streams available in the pub/sub system and made it follow the OTP conventions.

After that I looked into how the front end works and fixed various bugs on both front end and back end. I also located new bugs and issues, corrected some of the mentioned bugs/issues and also added new features in the front end.

In the end we all started to write on the course and product report.
Appendix B

Git Workflow

Figure B.1: Flow used by the team developing every feature in a different, isolated branch

Our repository has two main branches: master and develop. No commits should be pushed to these branches directly. Instead whenever a feature is to be added, a new branch is supposed to be created for this feature. When the feature is done, a pull-request is done for the branch with the feature
into the develop branch on GitHub.

When starting working on a new feature, a new branch is created to work on. The naming-convention of this branch is sprintX.Y where X is replaced by the current sprint and Y is replaced by the new feature.

The basic workflow is comprised of the following steps:

- clone from GitHub
- create a new branch locally to work on
- push to GitHub after each commit
- create a pull request from branch to develop
- repeat

After making a new pull request, the person making the request should check that the code passes all Travis tests. If not, the same person should amend the problem in the code, push the new code to Github and check that the Travis test cases passes after the fix was uploaded. Once this is done, someone else should review the code. If the reviewer is satisfied with the quality of the code of the produced feature, the reviewer will merge the feature into develop.

After each sprint, the develop branch is merged with the master branch, so that a stable version of the IOT Framework keeps being available at any time.
Appendix C

Coding Standards

C.1 Erlang

Erlang has a coding convention which were followed to some degree. The convention can be found at [http://www.erlang.se/doc/programming_rules.shtml](http://www.erlang.se/doc/programming_rules.shtml). We probably didn’t follow all parts of it since it is quite extensive. We did however follow these:

- **Variable names**: starts with capital letters for each now word in them. Example: *ThisIsAVariable*.

- **Function names**: contains only lower case letters and is separated with an underscore. Example: *this_is_a_function(Arg)*.

- **Module exports**: Exporting functions from modules is only done when necessary in order to avoid functionality leakage. This among other things means no usage of the *export_all* flag which exports all function in a module.

- **Common code**: is put into library modules instead of being duplicated.
C.2 Rails

The group followed the standard conventions agreed on the Ruby on Rails community. Using the scaffolding features supported by Rails, entities were created of our domain with a common nomenclature. For instance, if creating a User entity, the files will be named as it follows:

- **Model**: Written in singular, e.g. *User*.
- **Controller**: Written in plural, e.g. *UsersController*.
- **View**: Each resource will have multiple views, such as *index*, *show*, *edit* and *new*. Since ERB is used as the language for the views, every file will have *html.erb* as extension.
- **Partials**: Auxiliary files that could be shared and required by several views. Written with an underscore before, e.g. *_form.html.erb*.

In addition, the generated files will be saved in the appropriate folder, which could be *models*, *views* or *controllers*. These folders will be written always in plural, within the */app* folder.

Finally, the name of the methods is always is lowercase, using underscores to separate the words if the name is composed. Routes behave exactly the same way due to they should be compatible with the HTTP and URI standards too.