PROJECT REPORT

INDUSTRIAL ROBOT MONITORING FOR OPIFLEX AB

GROUP 5

2017-12-18

VIKTING GUSTAV LINDBERG, WILLIAM ACHRENIUS, NOÉ BAYLAC JACQUÉ, NAIDA DEMIROVIC, ANDREAS HEMLÉN, ERIK MARTIN CAMPAÑA, ELAHEH AILINEJAD, PONTUS WEDÉN

Table of Contents

1 Background

Our client for this project has been OpiFlex Automation AB, located in Västerås. They sell industrial robots mounted on mobile platforms, and what they call a “fenceless security solution”. The fenceless security solution is based on laser scanners detecting any obstructions near the robots, and the mobile platforms allow for manual or automatic movement of robots at a customer’s location.

The robots that OpiFlex sell also gathers important production data, such as how many items it has produced and the input and output levels. A robot fitted with the fenceless security solution will also produce messages regarding potential obstructions near the robots. This information is displayed on a tablet that currently must be physically connected to a robot. This, in turn, means that to check on a robot’s status, an operator must be present on the robot site.

The product OpiFlex asked us to develop is divided into two parts, a server that gathers information from the robots, and a mobile application that presents the data stored on the server. With this product, the need for operators to be physically present at the location of one or more robots is eliminated. Our delivered product supports several customers, differentiated via a login functionality, which means only a specific companies’ robots will be displayed for a specific user (operator) in the app.

2 Project Results

2.1 Deliverables

The project began with assigning initial responsibilities in the group (project manager, client communication, document templates, configuration management). After this, we gathered some initial requirements from OpiFlex that were enough to start the project with.

We delivered the first version of project plan document on 15th November 2017. When we got feedback, we improved it and sent the final version on 7th December 2017. We were all included in project plan which was presented on 22nd November 2017. The design document was first submitted on 30th November 2017, and after feedback the work has been continuous over the weeks until final submission on 9th January 2018. The design document presentation was complete on the 29th November 2017, and presented on our second presentation on 6th December 2017. In the final stages of the project, we were working with documentation, presentation integration and testing. We held the final presentation of our project on 10th January 2018, and the final project report was delivered on 11th January 2018. A summarization of these dates can be found in Table 1.

Deliverable	**Deadline**	**Date finished**
1st version project plan document	2017-11-16	2017-11-15
2nd version project plan document	2018-01-11	2017-12-07
Project plan presentation	2017-11-22	2017-11-22
1st version design description document	2017-11-30	2017-11-30
2nd version design description document	2018-01-11	2018-01-09
Design description presentation	2017-12-06	2017-11-29
Project report document	2018-01-11	2018-01-11
Final project presentation	2018-01-10	2018-01-10

Table 1. Deliverables and their completion date.

2.2 Product Features

The final product is divided in two parts: the server side part and the android application part. For the server side, we implemented a way to gather data from the robots. The data is structured in a database in a way that we can access the robot status, the current errors, the past errors and user notes on errors. This server can be accessed using the android application but can also be accessed by an administrator using the firebase web interface and potentially from a desktop application or a web application due to its open API and the multiple libraries available for Firebase on different platforms.

The most important features of our finished product are:

- To make sure we deliver the right data to the right company, we implemented an account functionality. Each account is linked to one company, so when you log in in the application, you have access to your company's data and not to another company's data.
- For the client side, we implemented a proof of concept using our database. The application displays in real time the status of the robots, the current errors and the errors history.
- For each error type, any user can make notes about that kind of error. Each time a user from the same company will check that error type, he will see all the notes related to the same error. For example, if a robot has a weakness in the arm, a user can put a note on the "arm malfunction" error type so that every company technicians are aware of that weakness if the error occurs again.
- There are settings to be able to hide some information that is not relevant for the user.
- Push notifications have also been implemented to alert the user about any errors that a robot has raised.
- For testing purposes, there is a python adapter that connects with the robot program sent by OpiFlex. This adapter sends the data from multiple robots to the database so the data is accessible by the clients. The python adapter is needed because we couldn't edit the robot simulator. When our product is used by OpiFlex, the robot should send by itself the data to the server.

2.3 Acceptance Test Results

As the acceptance testing coincided with a busy period for OpiFlex, we agreed with them to perform the acceptance tests in-house, and present our results to them later. The test cases were gathered from section 3.3 of the Project plan, and focuses what a user can do with the system, since those interactions were the most important to OpiFlex. Table 2 shows the result of our performed acceptance tests.

Test	**Outcome**
See list of robots and general information	Pass
Change settings	Pass
See information on specific robot	Pass
Display error log	Pass
Display error notes page	
From error logs page	Pass
From detailed robot information page	Pass
Document an error	Pass
Edit an error note	Pass*
Solve an error	
From error logs page	Fail
From detailed robot information page	Pass

Table 2. Summarized acceptance test results.

As can be seen in Table 1, 90% of our test cases passed, although it should be noted that for editing an error note, the current functionality is deleting it and adding a new note, not modifying the existing one. We fail on solving errors from the errors logs page, simply because that functionality has not been implemented there. The functionality missing from the error logs page is already implemented on the detailed robot page, which means that solving this test case should be easy for OpiFlex when they receive the code.

2.4 Missing features

OpiFlex wanted an application that would look appealing and create a more enjoyable experience. What this meant was never specified directly. Not much time was spent figuring out exactly how the application would look like, mostly because it was not a requirement and we chose to spend the time on the main features that OpiFlex requested. We had some short discussions regarding creating a more enjoyable experience, and came up with the idea of some sort of reward system. The reward system would keep track of what the users do and rewards the users for doing a good job, but this feature was never implemented.

The design of the application remains a standard android studio format. Both since the team consists solely of programmers, and both OpiFlex and the team felt like the functionality of the application was more important. Although, we created some samples of how we imagine the product to could like for consideration in the future.

An administrator account was a suggested as an addition to the login feature. The main idea was for every company to have an administrator. This account would have had additional permissions, such as extra settings for the information displayed to the users, and the ability to create new users. Currently this is not supported, although users and companies can be managed via the server web interface.

2.5 Future Improvements

The product can be improved in some areas:

- Statistics of the robot can be added to be able to check the overall productivity of the robots and of the factory in general.
- The android app only displays data from the database and send data to the database for now. A good improvement for the app can be the ability to interact with the robot. For example, when a robot stopped because of a problem, after the problem has been fixed, the operator can press a button in the app to restart the robot.
- The company expressed the wish to have a gamification aspect in the application. For example, a system could be implemented that rewards a company which is efficiently using the robots by making scores and assessments about the robots’ uptimes or amount of material produced.

3 Project Work

3.1 Changes to Organization and Routines

The project has followed the general organizational outline described in the project plan, although with some changes. The meetings have been carried out as planned, we have had one meeting per week, before the steering group meeting, and remote meetings were held during the holidays.

Throughout the project, the work has been divided into sub teams of the project group, something that we did not specify in the project plan that we would do. Sub teams were created for the server development, the application development, each deliverable and each presentation. This allowed us to work in parallel with different tasks during the weeks, rather than the whole group being locked to a single task for a whole week. We feel that this significantly increased the amount of work that the group could complete during each week.

As the project progressed the responsibilities of the project manager become more shared with the client communication responsible, such as scheduling meetings and maintaining the trello board. This is something that happened naturally and has not felt like a problem in the project. The configuration manager responsibility has remained unchanged, and the document template responsibilities became redundant as that responsibility fell on the sub team assigned to that specific document deliverable instead.

We specified a goal of 20 hours worked per week per group member, something that we did not achieve for any of the reported weeks. Although, this has not been a problem as the planned tasks for each week were, for the most part, always completed. When we failed to complete a task for a specific week it was more often caused by circumstances unrelated to the number of hours worked.

As specified in the project plan, we wanted to start with the design and implementation phase as soon as possible, which is something which we managed to do. Adding to that, we also started working on the deliverables as soon as possible, which meant we could complete them with time to spare before the deadlines. Starting early with development and deliverables decreased the overall workload per week, since all the planned tasks could be distributed over several weeks, this could explain why we never met the 160-hour total worked hours per week for the group.

3.2 Total Project Effort

For the whole team, both server and application, the plan was to work 20 hours a week for each group member. In the weekly group meetings, we delegated tasks to different group members, and estimated the time needed to complete these tasks together. As can be seen in section 3.2.1, no one met the goal of 20 hours worked per week, the reason for this could be the previously mentioned in section 3.1. Another possible reason, is that we did not become good enough at making time estimations of the tasks during the limited time of the project. Nevertheless, out of the requirements set, we feel that we have satisfied most them.

3.2.1 Worked Hours per Group Member

Table 3 and Figure 1 illustrates worked hours of everyone during the project, the last column of Table 3 shows the total hours worked per person.

	Week 1	**Week 2**	**Week 3**	**Week 4**	**Week 5**	**Week 6**	**Week 7**	**Week 8**	**Week 9**	**Total**
Pontus	11	17	16	17	13.5	11	12	6.5	9	113
William	9	5	16.5	11	12.5	10.5	6.5	3	11	85
Noé	10	8	13	11	15	7.5	4	6	3	77.5
Naida	10	7	8.5	14	10	9	2	2.5	3	66
Elaheh	10	7	12.5	13	10	6	5.5	7.5	4	75.5
Erik	9	9	13	12	11.5	11	5.5	6.5	3	80.5
Andreas	8	12	14	15	13	11	4	10	3	90
Viking	9	5	13	12.5	11	6	6.5	3	7	73
									Group total:	660.5

Table 3. Worked hours for each group member and group total worked hours.

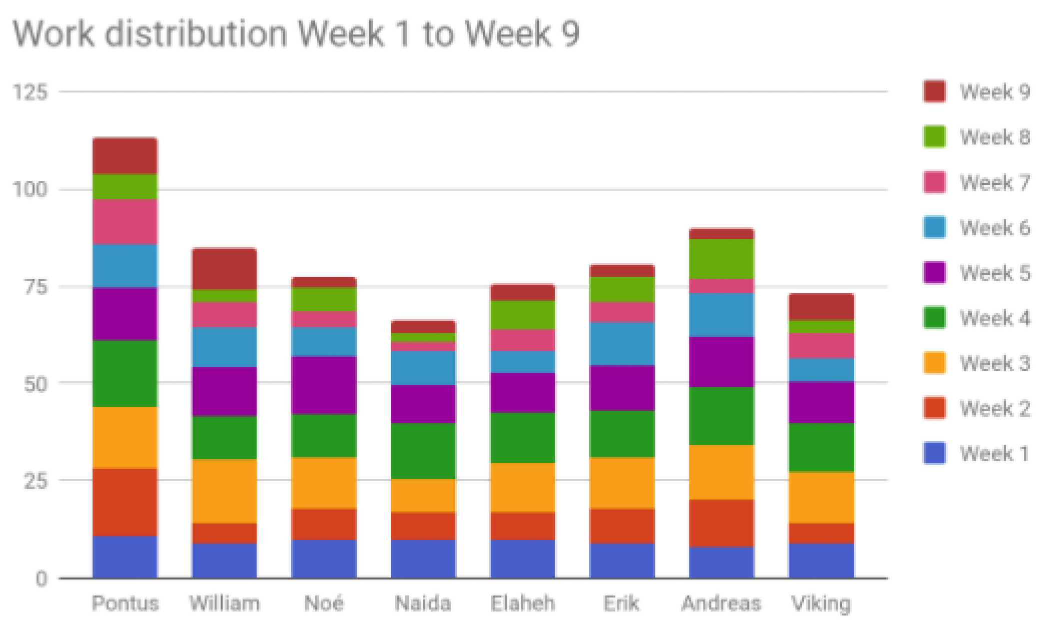

Figure 1. Worked hours visualized for each group member.

3.3 Responsibilities

Below follows a section for each group member, summarizing their responsibilities and what they've worked on.

3.3.1 Naida

I was present on steering meetings, group meetings, meetings with OpiFlex and project presentations. I worked on the first presentation which was related to our project plan. I was a part of the application team. My responsibilities in implementation phase were push notifications functionality, error detail page design and its functionality. I was also responsible to connect error detail page to the database which I finished with Eric since we organized that task as one person from the application team and one from the server team. At the end, I was working on writing test cases for acceptance testing and writing project report.

3.3.2 Noé

I was present on every meeting (steering, group meetings and meeting with OpiFlex and presentations). I did some parts in the three documents: quality assurance in the first document, graphical user interface in the second document and description of the key features and possible improvement in this third document. I presented the GUI design and the high-level architecture in the second presentation.

For the product itself, I was on the server team. I did a small python server that takes data sent by the robot program provided by the company and puts that data into the firebase database so the android application can access the data. I participated in the design of the database structure. I did the basic communications between the application and the database (getting robots and errors information from the database to the android application). I have also debugged the android application and written some minor functionalities in the internal of the mobile application.

3.3.3 Pontus

As client contact responsible I have arranged the meetings with OpiFlex and been the main point of contact for the company. I have been involved in each of the group, steering and company meetings.

In the group meetings, I have been responsible for updating the Trello board and creating the presentations for the steering group meetings.

Development wise I have been a part of the application team, where I have been responsible for the detailed robot information and settings pages. I have also been helping with general android development advice and bugfixes for some of the other pages.

For the deliverables, I have written the introduction/background for this document and the project plan, the high-level description in the project plan, the detailed design of the application in the design document, the result of the acceptance tests part of this document and the organizational changes part of this document. I presented my part of the design document during the second presentation.

3.3.4 Elaheh

I was present in project meetings (steering meetings, group meetings, meetings with OpiFlex and presentations). I did the presentation for project plan with two other group members, where I presented the introduction and project organization parts.

At first, we divided into two teams, application and server, and I was in application team. I did login page design, error log page design and their functionalities. I also was responsible for populating error log page, which I did together with Noé from the server team. For the last part of the project, I have some tasks for acceptance testing and writing project report.

3.3.5 William

I was present in every meeting, steering, group and OpiFlex meetings and presentations. I was assigned to take notes during the meetings. I have done some parts in each document, I did the initial product backlog in the Project plan, the system overview and system architecture in the design document and wrote this small section about the work I've done in the Project report. I was part of the final project presentation.

After the initial planning within the group I was assigned to the server team. There I have been researching firebase and in what ways we can use it for our app. I have been involved in the server design, how to structure our data, and have also written a cloud function for the server so that it can properly structure the received data the way we want it. I also created the login functionality in the application and connected it to firebase.

3.3.6 Erik

I was present in every meeting (steering, company and group meetings as well as in the presentations). I was part of the teams in the first presentation and the last one. Of the two teams that were formed, I was part of the one in charge of the server. I was part in the server and data structure design. Among others, my tasks were related to the error notes. This included how to structure notes in the server and implementing functions to get and send data between app and server. I also helped in the design and implementation of the part of the app in charge of the error details together with Naida. I also prepared the demo of the final app following what was said in the test cases.

3.3.7 Andreas

I was present in almost every meeting, steering, group and OpiFlex meetings and presentations. I missed one steering and one group meeting do to illness. In the design document, I wrote detailed software design for the server. In this project report, I wrote about missing features. I also took part in the presentation of the design document. I was in the server team, I did the setup for the firebase server and created firebase functions which lead to the finalization of the push notification. I also took part in designing of the server structure. For me, a lot of time where put in to researching firebase to learn about the different features that we could use in our project.

3.3.8 Viking

I was present on every meeting except for one company meeting, two steering and two group meetings. I was the project manager which meant that I was responsible to keep in contact with the steering group and send in the documents. I was part of the final presentation where I presented the high-level design, requirements and unfinished requirements. I've written the and planned effort part of the project plan, and the background and high-level description for the design document. I did not write anything on this report (except for this section) as I was part of the final presentation. I was a member of the android team, and my role was to make the RobotList page which would list the robots and show them in a list, as well as making the connection between the robot list and the detailed robot information page.

3.4 Positive experiences

We have all gained new experiences in this course. We needed to become familiar with some new tools and techniques. In that way, we could compare those tools with some that we knew before. Our knowledge is higher now. We started thinking about the organization in companies and were imagining working there. The most valuable thing we have learnt is organizing as a team and performing detailed planning of all the tasks that are needed in a moment. We have learnt how to deal with someone's weaknesses and strengths and how to transmute idea to reality.

Overall, it was great course with lots of practice and problems from reality. What we would like to be improved for next year is that the university arranges with companies enabling students to have weekly meetings with them, because it would be more official instead of having students do it on their own.

3.4.1 Advice for Students Taking the Course Next Year

Using the right tool can be very helpful and helps finishing the product on time. The goal of this course is not to develop something revolutionary from scratch, the goal is to deliver a usable product and to provide good documentation of said product.

4 Appendix – Test Cases

4.1 See list of robots and general information

- Execution sequence:
    1. Login to the application with user credentials.
        - The application shall redirect the user to the robot list page.
- Expected outcome:
    1. The list of robots is displayed.
    2. Each list item has retrieved and displays information from the server.

4.2 Change settings

- Execution sequence:
    1. Select a robot from the robot list.
    2. Press the “Menu” button.
    3. Select settings.
    4. Change settings by selecting and deselecting the checkboxes.
    5. Press save settings.
    6. Press the back button
- Expected outcome:
    1. The information on the robot page shall now reflect the selected information from the settings page.

4.3 See information on specific robot

- Execution sequence:
    1. Select a robot from the robot list.
- Expected outcome:
    1. Application changes to the detailed information view for the robot id that was pressed in the list.

4.4 Display error log

- Execution sequence:
    1. Press the “Error logs” button.
- Expected outcome:
    1. The page displaying all the errors for each robot should be displayed.

4.5 Display error notes page

This should be possible from either the detailed robot information page, or the complete error log page.

4.5.1 Error logs page

- Execution sequence:
    1. Press the “Error logs” button.
    2. Press an error in the list of errors.
- Expected outcome:
    1. The page with notes for that kind of error should be displayed.

4.5.2 Detailed robot information page

- Execution sequence:
    1. Select a robot from the list of robots on the robot list page.
    2. Select an error from the displayed list of errors on the robot detail page.

- Expected outcome:
    1. The page with notes for that kind of error should be displayed.

4.6 Document an error

- Execution sequence:
    1. Select a robot from the list of robots on the robot list page.
    2. Select an error from the displayed list of errors on the robot detail page.
    3. Type in a message in the text field at the bottom of the notes page.
    4. Press "Send new note"
- Expected outcome:
    1. The new note shall be added to the list of notes on the page.
    2. The new note shall be stored on the server.

4.7 Edit an error note

- Execution sequence:
    1. Select a robot from the list of robots on the robot list page.
    2. Select an error from the displayed list of errors on the robot detail page.
    3. Select the error to be edited.
    4. Change or add to the message via the "New note" text field.
    5. Press "Send new note"
- Expected outcome:
    1. The updated note shall be added to the list of notes on the page.
    2. The old note entry shall be deleted.
    3. The edited note message shall be stored on the server.

4.8 Solve an error

This should be possible from either the detailed robot information page, or the complete error log page.

4.8.1 Error logs page

- Execution sequence:
    1. Press the "Error logs" button.
    2. Long press an error in the list of errors.
- Expected outcome:
    1. The error should disappear from the list.
    2. The entry for that error on the server shall be marked as solved.

4.8.2 Detailed robot information page

- Execution sequence:
    1. Select a robot from the list of robots on the robot list page.
    2. Long press on an error in the list of errors on the robot detail page.
- Expected outcome:
    1. The error should disappear from the list.
    2. The entry for that error on the server shall be marked as solved.