NTNU
Kunnskap for en bedre verden

Department of ICT and Natural Sciences

IDATA2304 - computer communication and network programming

Greenhouse Project Documentation

Author:
Signe Ekern, Matthew Hunt, August Oksavik

December 2024

1 Introduction

Case: We work for a company called "SMG" (Student-Made Gadgets) that makes smart systems for farmers. A farmer with a greenhouse usually has many different devices for controlling their environment, such as devices for analysing the temperature and humidity and devices for controlling it such as fans, opening windows and heaters. All these devices can be connected to electricity and the internet, but as of now the farmers does not have an overall view of them. SMG has therefore gone out to make an application how connects all these devices to one server and display them collectively in one place.

This document is a comprehensive summary of how the work was performed, distributed, and executed during the final project on computer communication and network programming IDATA2304 for group 5, according to the scrum / Kanban methodologies.

2 Work planning

We started the project by reading the project requirements and becoming familiar with the task and what is to be expected as a result. We continued to clone the template code and create a new git repository and include all members. Before anything else was done, we systematically went through the project requirements and made every requirement we found into its own issue on a GitHub issue board. This was to ensure that we had an easily legible and structured approach to the project, so that no requirement was forgotten or overlooked. All tasks were then put as separate issues on the Kanban board, where every task had a larger description, priority, and size.

In accordance with the project requirement, our work was structured after scrum, with some elements of Kanban mixed in. We decided to use this hybrid between scrum and Kanban because we have found that it works better in smaller groups where the work is done collectively in one place. We use the typical Kanban board to delegate tasks and hinder the number of tasks one person can work on at a time. We ended up using the sprints from the scrum ideology, but did not include the daily Standup since our communication was already continuous.

Before starting the project, all members of the group watched the videos on network programming to get an overview, as well as participated in all lectures. This gave all the same starting point, and made all have a better overall view of the project, but to complete the project we have had to read from more resources online as well.

3 Work assignment and distribution

We did not set strict roles or parameters for the distribution of work throughout the project. Due to the nature of how we work, which is usually in a group and right next to each other, we had a very casual way of assigning work. The different tasks from the backlog were usually discussed at the start of the day and then assigned to a person based on availability and suitability. Each member generally chose to start on a specific issue that was pertinent to what they had already done or that interested them. Generally, we would work on that issue for the forthcoming sprint.

After each sprint, we discussed how the work was progressing and what was holding us back or making things difficult, and ending the retrospective with a sprint summary. Because experience begets expertise, some work naturally fell on those that had already worked on some issue previously, and because we each chose to go different routes at the start, we each ended up specializing in different elements of the project. However, defining what one person or another worked on specifically is essentially not possible, as all remaining members worked on nearly all parts of the project. With that said, the members could be roughly divided into one working on the graphical user interface and the control panel logic, one working on the server and its handlers, and one working on the greenhouse nodes. To conclude, all members worked on nearly all parts of the project but with a focus more on some specific parts.

4 Work breakdown

As stated above, to identify and create issues, the project description was systematically combed through to find all the parts that we understood to be critical to the successful completion of the project. While we were searching for issues to create, we made sure that the issues were as small as reasonable, concise and descriptive, and had a scope limited to be achievable in one or two sprints. The issues were considered closed or finished when the functionality it was supposed to implement, worked as expected.

5 Sprints and sprint retrospectives

We generally managed to implement or complete the issues we chose during the sprint meetings. Some, of course, took more time or resources than expected or what was thought to be a good implementation turned out to be good enough for later functionality. However, such issues are expected when developing software. It is unreasonable to assume that every time or scope estimate will be confined enough to finish in a singular week. These issues that were found to be too large, were subsequently broken down into smaller issues that were easier to finish for one person in a sprint.

5.1 Sprint retrospectives

Sprint nr.1 10/10/24 - 17/10/24
Status: on track
Created various issues from the Project description and assigned T-Shirt sizes to each issue to better help us assign different parts to the different team members depending on workload. Created an issue for an initial draft for the protocol so that we can agree on certain points before anyone starts programming. The actual protocol document will be written in full later in the project, but our initial draft will act as the basis for the finished protocol. Assigned a issues to people on the team that should be worked on until the next sprint report.

Sprint nr.2 24/10/24 - 30/10/24
Status: on track
For next time we wish to finish the client handler and a rudimentary solution for communicating between clients and the server. We also wish to implement multithreading to allow multiple clients to communicate independently from eachother. The goal is also to start implementing observable/observer interfaces to notify all clients when needed etc.

Sprint nr.3 30/10/24 - 07/11/24
status: on track
Managed to get multi-threading and basic command sending to work. Until next week we will work on GUI as well as TCP communication between greenhouse and nodes. Will also work on improving commands sent and received.

Sprint nr.4 07/11/24 - 14/11/24
Status: on track
Greenhouse has been refactored into standalone nodes. Multi-threading for reading sensor data on different threads has been implemented. Control panel logic has been refactored and readied for communication. For next week we wish to implement control panel communication with our greenhouse communication

Sprint nr.5 14/11/24 - 21/11/24
Status: at risk
Communication from greenhouse to server and communication from control panel to server now works as intended. For next week we will try to finish sending sensor data from the server to control panel once it receives data from the greenhouse. TCP document has been improved. New

data structure for the commands created. Difficulties with retaining information between threads, needs to be sorted.

Sprint nr.5 21/11/24 - 28/11/24
Status: on track
We merged control panel communication and Gui branches into each other. We have also gotten the actuators to update and actuate from the control panel. Continued work on the protocol. For next time we need to integrate broadcast and extras. We also need to continue work on video and report :)

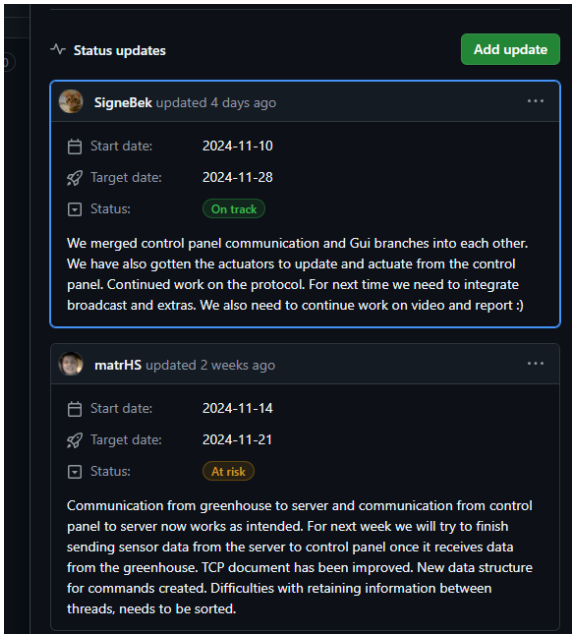

Figure 1: Sprint retrospective log on GitHub