Bachelor's thesis

Stian Pedersen
Johan Strand
Niklas Hatteberg Leivestad

Automation Of Digital Scales

May 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Bachelor’s thesis

2021

Stian Pedersen
Johan Strand
Niklas Hatteberg Leivestad

Automation Of Digital Scales

Bachelor's thesis
May 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

English

In a competitive breeding industry there is, besides desired genes, a large amount of data and data management needed to stay ahead. Norsvin is a company that relies on collecting data on pigs and breed the ones that have desired factors, this requires compilations of frequent weighing for accurate analysis. With this assignment Norsvin aims to improve their efficiency and accuracy of their compilations with the use of digital weights. Our solution is made for desktop, developed with Java and includes support for Bluetooth and RS232 connection, storing recorded data in a local database, with the added options of exporting and importing entire compilations of such data. The application also has a system-wide hotkey for fetching weight from the connected scale, and returning it through the computer's copy and paste function - making the application compliant with other existing solutions that lack connection with the scales. In this project we have created a solution which can be used in the product environment by Norsvin's farmers. We have studied transmitting and receiving data from external hardware, and we have been evaluating their respective Java packages in terms of private and public use, simplicity and efficiency. We have learned about the Serial Communication Protocols such as RS232, UART and RFCOMM for Bluetooth. And learned about different database system, and how to integrate them into applications. Last but not least, the whole group has become a lot more familiar with and proficient in the Java programming language.

Norsk

I en konkurransedyktig avlsindustri er det, i tillegg til ønskede gener, en stor mengde data- og datahåndtering som trengs for å holde ledelsen. Norsvin er et selskap som er avhengig av å samle inn data om griser og avle frem svin med de ønskede kvalitetene. Dette krever samlinger av hyppig veiing for nøyaktig analyse. Med denne oppgaven ønsker Norsvin å forbedre effektiviteten og nøyaktigheten av samlingene sine ved bruk av digitale vekter. Løsningen vår er laget for PC, utviklet med Java og inkluderer støtte for Bluetooth- og RS232kommunikasjon, lagring av registrerte data i en lokal database, med tilleggsmuligheter for å eksportere og importere hele samlinger. Applikasjonen har også en hurtigtast for å hente vekt fra den tilkoblede vekten og returnerer vekten via PC-ens kopier og lim inn funksjon. Dette gjør applikasjonen kompatibel med andre eksisterende løsninger som ikke kan kommunisere med vektene. Med dette prosjektet har vi laget en løsning som skal brukes av hos Norsvins bønder. Vi har studert overføring og mottak av data fra ekstern maskinvare, samt evaluert dems respektive Java-pakker etter privat og offentlig bruk, enkelhet, og effektivitet. Vi har lært en del om serielle kommunikasjonsprotokoller som RS232, UART og RFCOMM for Bluetooth. Vi har også lært om forskjellige database systemer, og hvordan integrere dem inn i appligajoner. Sist men ikke minst, har hele gruppen blitt mye mer kjent og erfaren i bruken av Java-programmerings språket.

Contents

Figures

Tables

Glossary

.csv A Comma Separated Values (CSV) file is a plain text file that contains a list of data. These files are often used for exchanging data between different applications. For example, databases and contact managers often support CSV files [1]. 33

Buffer reader Reads text from a character-input stream. It buffers the characters in order to enable efficient reading of text data. [2] . 29

CI/CD pipeline CI/CD bridges the gaps between development and operation activities and teams by enforcing automation in building, testing and deployment of applications.[3]. 12

Code With Me Code With Me is a collaborative development and pair programming service [4]. 52

Controller A instance of this class is created when the FXML file is loaded. The FXML controller class binds the graphical user interface components declared within the FXML file together. Making the controller class the mediator [5]. 40

Hashmap A HashMap store items in "key/value" pairs. [6]. 39

i18n Internationalization and localization, where 18 stands for the number of letters between the first i and the last n in the word internationalization [7]. 35

JavaDoc JavaDoc is a documentation generator for the Java language that generates API documentation in HTML format from Java source code. The HTML format is used for adding the convenience of being able to hyperlink related documents together [8]. 14, 52

JavaFX JavaFX is a open source software platform for creating and delivering mobile, desktop applications and embedded systems build on Java [9]. 26

jSerialComm Java Library which gives platform-independent serial port access [10]. 25, 30

l10n Localization, where 18 stands for the number of letters between the first L and the last N in the word Localization [11]. 35

ResourceBundle Resource bundles contain locale-specific objects. When your program needs a locale-specific resource, a String for example, your program can load it from the resource bundle that is appropriate for the current user's locale. [12]. 35

RFCOMM RFCOMM emulates RS-232 serial ports.. iii, iv

RFID Radio frequency identification, RFID. Is a method of storing and retrieving data using small devices call RFID tags [13]. 2, 17, 31, 37

RS232 RS232, Recommended Standard, is a standard protocol used for serial communication, it is used for connecting computer and its peripheral devices to allow serial data exchange between them [14]. iii, iv, 5, 28

RXTX An open source java class library that provides serial and parallel port communication [15]. 25

SPP Serial port profile is intended to replace RS-232 cables (or other serial communication interfaces). SPP is for sending and receiving bursts of data/information between two devices [16]. 29, 39

Tatonumber Unique identification number of the pig. 4, 16, 19, 31, 45, 51

UART Universal asynchronous receiver-transmitter. A computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable [17]. iii, iv, 25

Chapter 1

Introduction

1.1 Project Description

English translation of the official project description

Client:

Norsvin SA, Storhamargata 44, 2317 Hamar

Task:

Application for reading digital scales.

To increase the degree of digitization as well as secure and simplify the process of weight registration on pigs in pig herds, digital scales have been used in Norsvin that can communicate with external hardware.

Objectives of the assignment:

In this assignment, it is desired to develop a desktop software for PC, and possibly a mobile application, that can read from digital scales and transfer this to the writing cursor, be it a text editor, web application or similar.

Thesis requirements:

In Norsvin, several different weights are used and the reading application must be able to freely configured to support all of these. The configuration application must, after configuration, be able to run as a background application or service. It must be possible to read the weight with a global hotkey (e.g. function key F7).

Which key should act as the hotkey, shall to the extent possible be determined by the user when configuring the application.

Desired options:

Initially, only weight reading is required, but the application should also have an option for reading of eg the animal's ID via an barcode or RFID reader which is connected to the system running the application. The application should also be able to add fixed data fields that can be sent when weight is gained read. Examples of such additional data fields can be: today's date, time stamp for weight reading.

1.2 Background

Why we chose the assignment

The team had previously agreed that we would prioritize tasks that involved application development as its core workload. From the task description, we knew that it revolved around making an application. We quickly deduced what API's and other functionality that was likely to be integral parts of the application, some of them involving communication with external hardware. After the first meeting with Norsvin we presented our vision and what we could offer as a solution. The feedback we received got us moving forward, and we more or less confirmed what technologies were going to be involved. Norsvin told us that their digital scales uses both Bluetooth and RS232 as their external communication interfaces, implying that the application has to support communication through both.

There were also a requirement that the application could be used without access to the internet, as not all farmers has internet connection in their barn. Thus, data had to be stored locally, and uploaded when the user had access to the internet. The application was also intended to be used by employees in other countries, so it had to be internationalized. Based on this, there was a wide range of different aspects involved to create the application, which made it an interesting task and gave us free rein on how to develop the interface itself. This gave us the opportunity to be creative in how we developed the application. This application will be used frequently and will run over and over again, so the design needs to fit the usage and be as simple as possible.

Our related backgrounds

We all study Engineering - Computer Science. Stian Pedersen and Johan Strand have taken the subject Application Development and had some experience in developing desktop applications. While Niklas Leivestad has taken the subject Ergonomics in Digital Media which provides insight into the importance of design. The client also wanted this to be done in Java, which all group members had some previous experience with. None of the groups members had any experience communicating with external hardware. Which should prove to be somewhat more difficult than we initially expected. We also had to communicate with and control/manage the database from our application, which was a new experience for all of us.

1.3 About Norsvin

Norsvin and the assignment

Norsvin is a cooperative (SA) owned by Norwegian pig producers. A breeding company that conducts breeding work on pigs in Norway. They distribute genetics in the form of semen, to pig producers both nationally and internationally. Norsvin's genetics are among the best in the world, and are offered to customers globally through the company Topigs Norsvin; one of the world's largest companies in pig genetics. Norsvin's headquarters is located in Hamar, with approximately 70 employees [18].

The desire to gather and storing data of each pig in every weighing session more effectively, is the reason for this assignment. This is why our task is to increase the degree of digitization, to secure and simplify the process around weight registration on pigs in pig herds. Today's routines for weighting is to manually enter the pig's ID number and weight. Norsvin uses digital scales that can communicate with external hardware and therefore wants to utilize this feature. The scales communicate via Bluetooth and RS232 port.

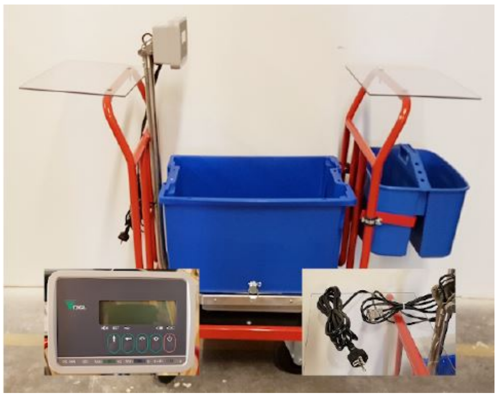


Figure 1.1: Digital weight scale

Purpose

Norsvin's purpose is to help secure the finances of Norwegian pig producers by offering a high quality genetic material. Norsvin's international investment therefore aims to contribute to earnings for research and development work and at the same time reduce the cost associated with inseminating [18].

Until Norsvin was founded in 1958, the breeding of pigs was run by a few farms, and the progress that was created was not available to everyone. Due to this, Norsvin was formed and founded on the following guiding principles:

- The farmer's right to self-determination.
- Progress for the many.
- The achievements of science.

These guiding principles are equally valid today, and in short, this means that Norsvin will base its breeding work on knowledge and modern breeding methods [18].

Norsvin owners

The owners, who are pig producers/farmers - about 1300 today - will be involved in deciding important breeding priorities. The members are organized in county councils, which in turn can be organized in local community groups. Everyone should be allowed to take part in the progress that breeding work provides [18].

1.4 Goals and scope

Current procedure of recording new weights.

Today the farmers will have to manually check and write down; Tatonumber or RFID, date and weight - And record them into the central database individually by hand.

We got to see a demonstration of how the recording process worked, and all of the data points had to be inserted individually i.e: You first type out the Tatonumber and click on a "Next" button. The website will then load a new page, where you will have to fill out the date and click on the "Next" button again. All recorded data will follow this procedure of continuously filling in one piece of data and moving on to the next page to fill in the next piece of data. This process is both inefficient and includes a lot of points where human errors can occur.

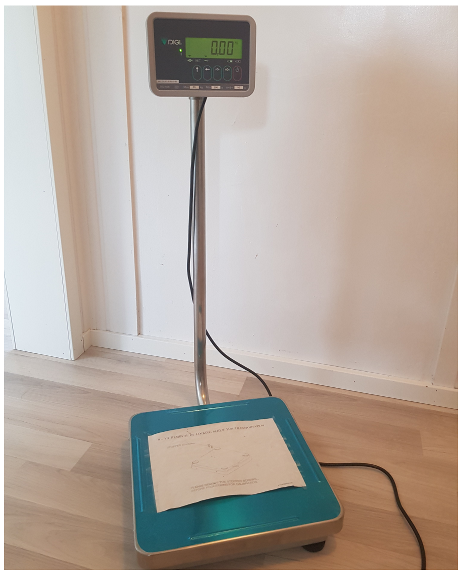
Our application.

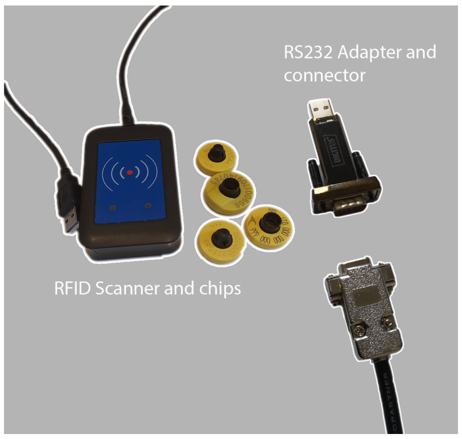
Our solution provides the user with the ability to scan the RFID tag or barcode on the animal, which will automatically fill in the fields Tatonumber, Race, Sex, Location and Date. The farmer will then click on the weigh button, which will fetch the current weight applied on the connected digital scale, and record it to our application. Now the farmer only has to click on the "Add" button, and a new weigh instance will be stored in a local database. The farmer can then continue weighing the rest of the animals. Once the farmer has completed all recordings for the day, they can then export all new recordings to a .csv file, and import them into the central database. For convenience, the farmer also has the option of importing new tatonumbers from a .csv file so that they don't have to manually type in the tatonumber of new animals by hand. This also reduces the possibility of submitting tatonumbers with typos in them to the central database. The application also supports the use of a global hotkey, which will get the current weight from the connected scale, insert it into clipboard and paste it. This functionality is meant for those that has access to internet while weighing the animals, and lets them skip the extra steps of exporting and importing the recorded data.

1.5 Hardware

We were provided with three digital scales, a T-Scale model BWS, two DIGI model DS-166 scales - one equipped with Bluetooth and one equipped with RS232. We also got an RFID scanner and 4 RFID chips. See figure: 1.2 for an overview of hardware. We had the option of getting a barcode reader sent over as well, but we knew from previous experience that such readers just needs to be plugged into a computer, and they work just as intended. Needless to say, they require no additional configuring to make them work. The RFID scanner works in a similar manner, so we didn't have to spend any time implementing support for the device.

The model DS-166 has a weight limit of 30kg, and the BWS model has a limit of 200kg. The BWS model didn't come with any form of connection so we didn't get to use it for testing our software. But the standard transferring protocol should be compliant with RS232


(a) DIGI model DS-166 scale



(b) RFID scanner, RFID chips and RS232 Adapter and connector

Figure 1.2: Hardware used in the development process.

1.6 Framework

Norsvin SA wants a Windows desktop application. The code should be easily converted to android for mobile devices for scalability to mobile devices in the future.

Delimitation

Operating systems: Mainly supported operating systems are Linux and Windows, but iOS should be supported by default as it is Unix compatible.

The application should be able to fetch the weight of an object from the provided digital weights - Either using RS232 cable or Bluetooth.

- The application should be able to export its stored data to an .csv file.
- The application should be able to import data from .csv files.
- The application should have a convenient and efficient way of getting information about specific animals or groups of animals.
- The application should be compatible for internationalization.

1.7 User group

Users of the application

The main target group for this application is farmers, but also Norsvin's employees. They will use the application in everyday work, to make weight reading and registration of pigs more efficient.
As the largest target group is farmers, the users will have different technological experiences, which must be taken into account, and the application will be used in the field where the conditions are not always optimal. Hence the interface must be designed with self-explanatory and clear design, with big visible buttons as well as minimize unnecessary information. Simple alerts has to be given when the user has entered faulty data, or when other unexpected errors occur.
Norsvin also has employees in several other countries with different languages, so we have therefore internationalized all text strings, and added language support for Norwegian, English and Dutch.

Readers of the report

The report will first and foremost give the examiner and supervisor an insight into how we carried out our project from a developer's perspective. But also for the client who has their own developers, who may want to develop the application further in the future.

1.8 Roles

Stian Pedersen as developer. He had main responsibility of the back-end, with the implementation of the database and csv import/export, and was leading the agenda in meetings.
Johan Strand as developer. He had main responsibility of the Bluetooth scale and graph, from implementation to front-end.
Niklas Leivestad as developer. He had main responsibility of the RS232 scale, from

implementation to front-end and was the recorder on the meetings.
Tom Røise was supervisor. With meetings with the project group members every other week on Wednesday or Thursday, this frequency increased to every week one month from the deadline.
Rune Sagevik was product owner and representative of Norsvin. Rune and Lars Terje Bogevik expressed Norsvins vision during the project. They both attended the meetings we had throughout the project.

Chapter 2

Development process.

2.1 Development model

SCRUM Sprint

At the start of the project, we haphazardly and unanimously agreed to follow the SCRUM Development framework, because it is a very versatile and organized model that frankly, is very popular among software developers. As we started the project without actually thinking through what is required for success with this model, we were a bit ill prepared when starting with the development, because none of the team members had any previous experience with SCRUM. Thus we started the project without any preparation for how to set up a functional framework beforehand [19].

This would prove to be a mistake on our end. We did not scale the SCRUM framework to suit our small team and lacked both the commitment to follow through with all the formalities, as well as learning the appropriate processes necessary to uphold an optimal SCRUM framework, such as having a dedicated scrum master responsible for planning meetings and logging progress as well as planning the next sprint.

Kanban

Why kanban?

After realizing that our empty shell of a SCRUM framework was falling apart, we fully committed to switch over to only using Kanban, which was previously only a supplementary tool for our SCRUM model. We did this decision after researching a bit, and found that Kanban was a very fitting model for our workflow.

	Scrum	**Kanban**
Cadence	Regular fixed length sprints (ie, 2 weeks)	Continuous flow
Release methodology	At the end of each sprint	Continuous delivery
Roles	Product owner, scrum master, development team	No required roles
Key metrics	Velocity	Lead time, cycle time, WIP
Change philosophy	Teams should not make changes during the sprint.	Change can happen at any time

Table 2.1: Summarized comparison between scrum and kanban [20]

We had previously established all necessary prerequisites for a fully functional Kanban framework, which was a Kanban board and close communication with team members/developers and Norsvin. We used two separate communication channels, Microsoft Teams and Discord. Microsoft Teams was used to communicate and host meetings with our supervisor and Norsvin, while Discord was used for communication internally within the development team 2.2.

Transition

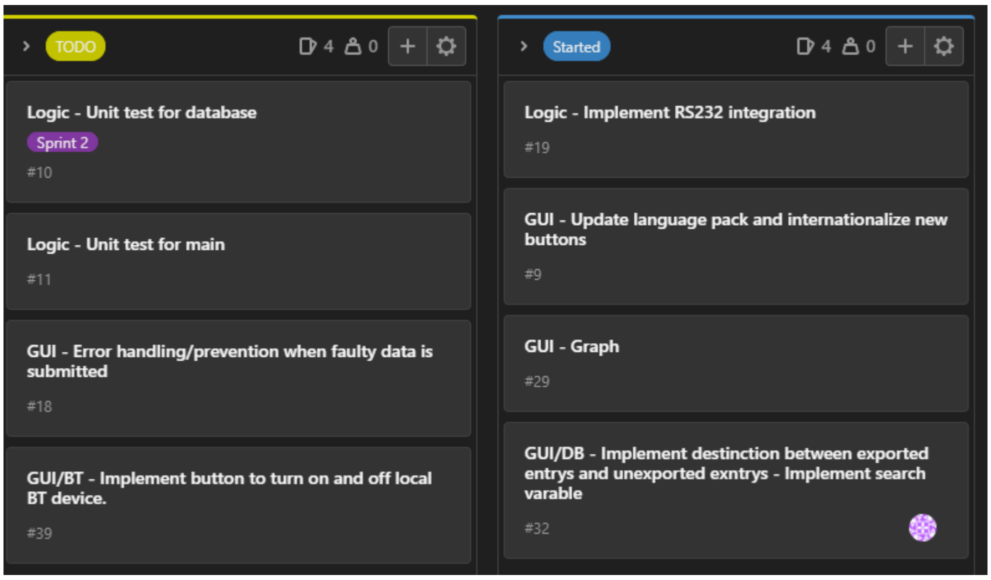
After switching to Kanban, we were more proactive in using our issue-board. Adding issues whenever noticing missing functionality or errors, as well as during meetings with Norsvin - when they had feedback on things to alter, or new desired functionality.

The table 2.2 contains the guidelines for putting cases in correct places, which represents in what state that specific case is in at any given time. The table was produced to give the team a collective perception of where to put the cases they were working with [21].

In addition to the guideline table, we also categorized all Kanban cases for better readability. The categorizations would be stated before the description of the case, with the intent of making it easier for the developers to filter out the cases that fit what they want to work on. Figure 2.1 is a snippet of the Kanban board, containing some examples of the categories. "*GUI/DB - Implement distinction between exported entries and unexported entries - Implement search variable*" This case is assigned the tags GUI/DB because it involves development in both the GUI and DB.

Column	Description
Open	Cases that has been added because it would be a nice future but not a necessity.
TODO	Anything that has to be addressed at some point. Such as implementing requested functionality from Norsvin, bugs and other missing functionality that has priority of getting implemented.
Started	Cases where development has been started.
On halt	This tag is used when hitting a roadblock, where either major changes needs to be done or the developer is simply stuck and does can not complete the issue. The group will then assemble and tackle it together.
Waiting for review	When a case has been been implemented, and a merge request is being made - The request will then be reviewed by at least one other developer.
Reviewed - OK	Case is moved to this column if the reviewing developer did not find any errors in the code, the reviewer will also approve the merge request.
Closed	Cases are moved here after it has been implemented into the master branch.

Table 2.2: Kanban Column descriptions


TODO 4 0

Logic - Unit test for database
Sprint 2
#10

Logic - Unit test for main
#11

GUI - Error handling/prevention when faulty data is submitted
#18

GUI/BT - Implement button to turn on and off local BT device.
#39

Started 4 0

Logic - Implement RS232 integration
#19

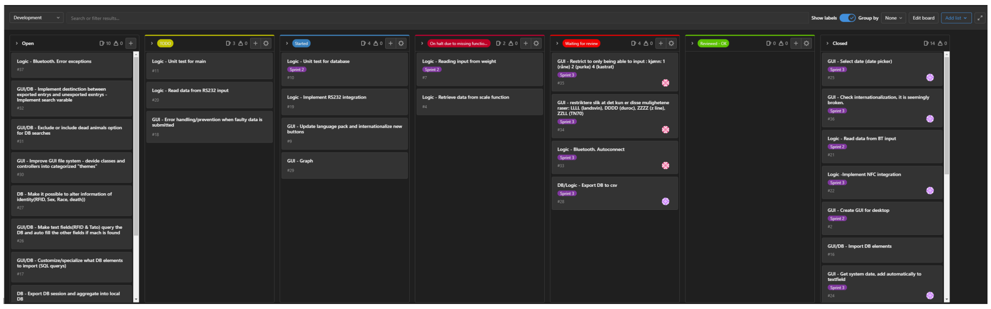
GUI - Update language pack and internationalize new buttons
#9

GUI - Graph
#29

GUI/DB - Implement destinction between exported entrys and unexported exntrys - Implement search varable
#32

Figure 2.1: Highlight of our categorized case names

We did not have any specific priority listings other than "TODO" and "On halt...", because the assignment specification was inherently very loose and did not state any requirements other than being able to receive the weight output by the digital scales. This made us focus more on quality and usability which did not have any specific priority, as there were only three core requirements to fulfill.



Open 10 0
- Logic - Bluetooth. Error exceptions #37
- GUI/DB - Implement destinction between exported entrys and unexported exntrys - Implement search varable #32
- GUI/DB - Exclude or include dead animals option for DB searches #31
- GUI - Improve GUI file system - devide classes and controllers into categorized "themes" #30
- DB - Make it possible to alter information of identity(RFID, Sex, Race, death)) #27
- GUI/DB - Make text fields(RFID & Tato) query the DB and auto fill the other fields if mach is found #26
- GUI/DB - Customize/specialize what DB elements to import (SQL querys) #17
- DB - Export DB session and aggregate into local DB

TODO 3 0
- Logic - Unit test for main #11
- Logic - Read data from RS232 input #20
- GUI - Error handling/prevention when faulty data is submitted #18

Started 4 0
- Logic - Unit test for database Sprint 2 #10
- Logic - Implement RS232 integration #19
- GUI - Update language pack and internationalize new buttons #9
- GUI - Graph #29

On halt due to missing functio... 2 0
- Logic - Reading input from weight Sprint 2 #7
- Logic - Retrieve data from scale function #4

Waiting for review 4 0
- GUI - Restrict to only being able to input : kjønn: 1 (råne) 2 (purke) 4 (kastrat) Sprint 3 #35
- GUI - restriktere slik at det kun er disse mulighetene raser: LLLL (landsvin), DDDD (duroc), ZZZZ (z line), ZZLL (TN70) Sprint 3 #34
- Logic - Bluetooth. Autoconnect Sprint 3 #33
- DB/Logic - Export DB to csv Sprint 3 #28

Reviewed - OK 0 0

Closed 14 0
- GUI - Select date (date picker) Sprint 3 #25
- GUI - Check internationalization, it is seemingly broken. Sprint 3 #36
- Logic - Read data from BT input Sprint 2 #21
- Logic -Implement NFC integration Sprint 3 #22
- GUI - Create GUI for desktop Sprint 2 #2
- GUI/DB - Import DB elements #16
- GUI - Get system date, add automatically to textfield Sprint 3 #24

Figure 2.2: Snippet of our Kanban board in Git Lab.

2.2 Project management

Git

We are using NTNU's GitLab server for our application's source control. [1]

Repository configuration

To prevent the possibility of having conflicts between the group member's code, we decided to:

- Make the master branch protected, and restrict all push requests.
- Merge requests must be approved by at least one repository maintainer, which does not include the developer submitting the merge request.
- Before a branch can be merged with the master branch, it has to pass the CI/CD pipeline first.

With using these three restrictions combined, we have effectively removed all easily preventable conflicts and commit mistakes. Our pipeline is mostly just using the standard configuration for Gradle, which checks imported libraries and tests if the application builds properly. The only functionality we added to the pipeline was an automatic JavaDoc generation process. See: [2.3]

[1]`https://git.gvk.idi.ntnu.no/`

Discord

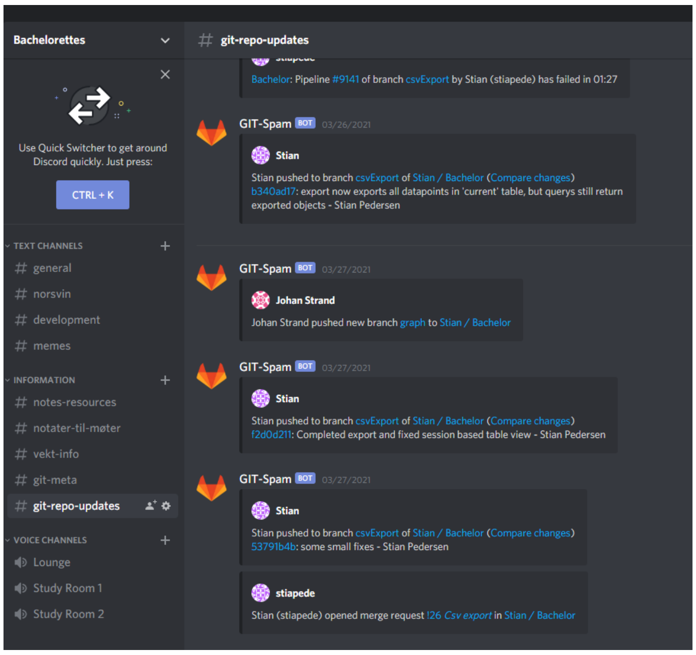
We made our own Discord server soon after establishing the project group, because it is a really convenient tool for communication with a team. Discord allows its users to make multiple channels within a server, where everyone can share useful resources and discuss specific parts of the development in an orderly manner (See 2.3).

Communication

Discord has both a desktop and mobile application, which makes communication with all members available at all times. And has the same communication capabilities as Teams, with the difference of convenience and accessibility [22] . In addition Discord has support for web-hooks, which can be used to push Git Lab notifications.

Notifications from Git-Lab

Implementing the Git Lab web-hook was integral for keeping track of changes on the repository, which is very important when other developers have to read over the code in the merge request before it can be merged. The use of discord has helped us immensely with keeping track of the general development of the application, while simultaneously being our central hub for communication - Both in chatting and in digital work sessions.



Bachelor: Pipeline #9141 of branch csvExport by Stian (stiapede) has failed in 01:27

GIT-Spam BOT 03/26/2021
Stian
Stian pushed to branch csvExport of Stian / Bachelor (Compare changes)
b340ad17: export now exports all datapoints in 'current' table, but querys still return exported objects - Stian Pedersen

GIT-Spam BOT 03/27/2021
Johan Strand
Johan Strand pushed new branch graph to Stian / Bachelor

GIT-Spam BOT 03/27/2021
Stian
Stian pushed to branch csvExport of Stian / Bachelor (Compare changes)
f2d0d211: Completed export and fixed session based table view - Stian Pedersen

GIT-Spam BOT 03/27/2021
Stian
Stian pushed to branch csvExport of Stian / Bachelor (Compare changes)
53791b4b: some small fixes - Stian Pedersen

stiapede
Stian (stiapede) opened merge request !26 *Csv export* in Stian / Bachelor

Figure 2.3: Git notification channel.

2.3 Gradle and CI

Gradle

Gradle is a build automation tool for multi-language software development. It controls the development process in the tasks of compilation and packaging to testing, deployment, and publishing. The goal of Gradle is to add functionality to a project, and is highly customizable [23].

We used Gradle because of its ease of use and support for implementing external libraries. We also had previous experience with using it, so the benefits of choosing Gradle as our build tool over Maven and ANT, ended up far outweighing the other options. There is also possibilities to make android applications with Gradle, which makes the code reusability more available for further development.

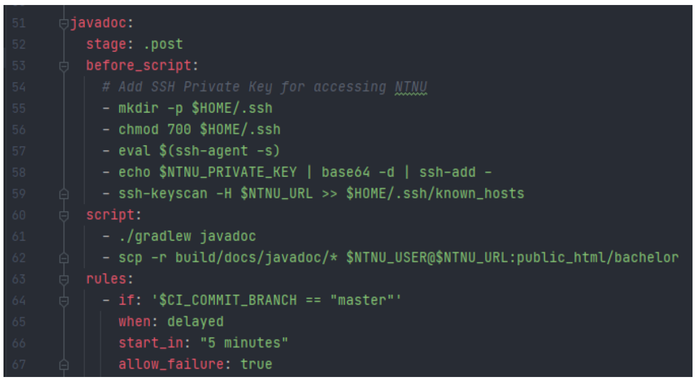
Gradle has integrated support for the JavaDoc API , which makes it quite simple to integrate the generation and publishing of our JavaDoc with the pipeline.

CI/CD pipeline

Our CI will generate new JavaDoc files of our master branch 5 minutes after the other pipelines has succeeded, and will then copy the generated HTML files to the web-server for students, hosted by NTNU. Our JavaDoc API is posted here: [24]

As showcased in the snippet of our YAML file [2.4], most of the setup for the JavaDoc API is for interacting with the web server in a secure way. None of us had any previous experience with setting up or configure the Git pipeline, which made implementing our solution quite challenging. The goal for the CI script was to generate the JavaDoc files, and copy those files to the web-server. By following these guides provided by GitLab: [25], [26] - We eventually managed to figure out how to use the GitLab CI variables and how to make the scripts necessary for secure server interaction.

We have configured the script to only apply to the master branch, and it will execute 5 minutes after the build and test jobs has passed successfully.



```yaml
javadoc:
  stage: .post
  before_script:
    # Add SSH Private Key for accessing NTNU
    - mkdir -p $HOME/.ssh
    - chmod 700 $HOME/.ssh
    - eval $(ssh-agent -s)
    - echo $NTNU_PRIVATE_KEY | base64 -d | ssh-add -
    - ssh-keyscan -H $NTNU_URL >> $HOME/.ssh/known_hosts
  script:
    - ./gradlew javadoc
    - scp -r build/docs/javadoc/* $NTNU_USER@$NTNU_URL:public_html/bachelor
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'
      when: delayed
      start_in: "5 minutes"
      allow_failure: true
```

Figure 2.4: yml code for generating and publishing the JavaDoc

Chapter 3

Requirements specification

3.1 Use-case

With the following use-cases and use-case-diagram [3.1], we want to provide an overview of the application in a simple manner. We have been using use-cases from earlier projects and know its strengths, such as showing complicated systems in a simple way. It will give a deeper insight into the different processes the user can perform, and how they operate. Some of the biggest 'eureka moments' from this project were the first connection with the scale and application. Later we finally got to receive the current weight. Therefore, we have chosen to attach an extended use case of these functions.

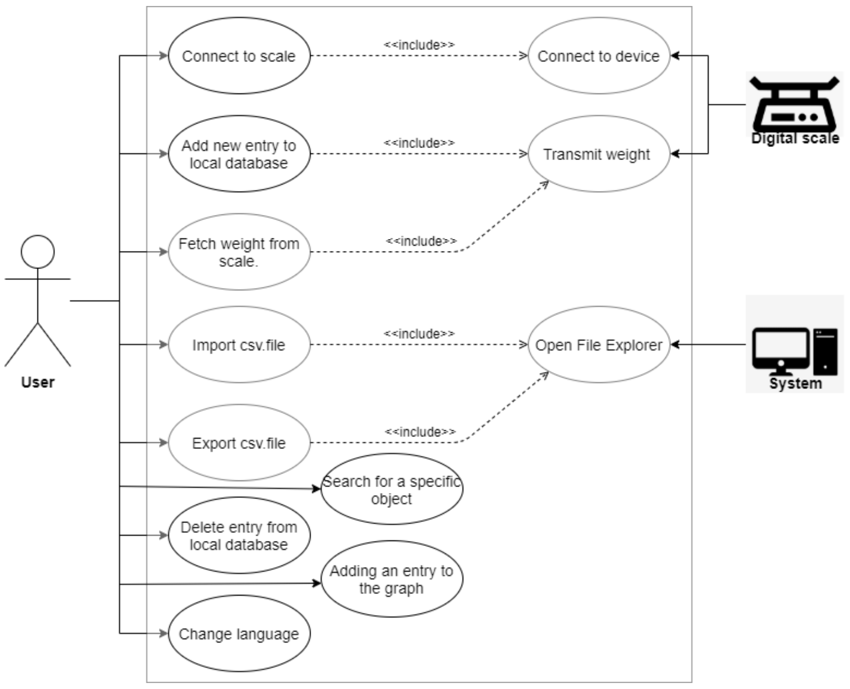


Figure 3.1: Use case diagram made in draw.io

High level use case descriptions

Use case name: Connect to scale
Actor: User
Goal: Connect to a digital scale
Description: The user clicks on settings, and chooses form of connection in the drop down menu. A pop up box with list of available scales will appear [3.2]. Choose one by clicking it and then click the connect button.

Use case name: Add new entry to local database
Actor: User
Goal: Add a pig's new weight information to the database
Description: The user fills in either Tatonumber or RFID related to the pig. Click on weigh. Then click on "Add" to add the new weight into the Currently Registered table [6.1].

Use case name: Fetch weight from scale
Actor: User
Goal: To get the current weight from the scale
Description: In application: Click the Weigh button. The current weight will be retrieved from the connected scale, and inserted into the "Weight" text field.
As a background service: Press F4 to put the weight in any chosen text field.

Use case name: Import CSV file
Actor: User
Goal: Importing pigs from Norsvin's central database.
Description: Click Import. A file explorer will pop up and lets you navigate through your system files. And you will be able to select the .csv file you want to import. Only .csv files and folders are displayed in the file explorer.

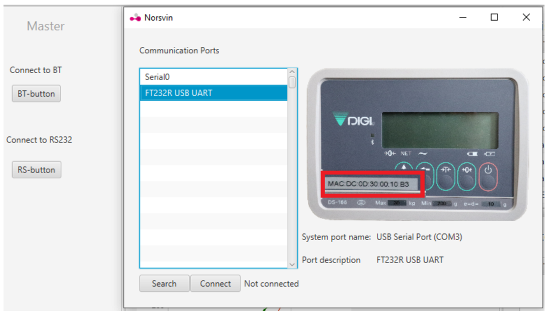
Use case name: Export CSV file
Actor: User
Goal: Export everything in the Currently Registered table [6.1] into a CSV file.
Description: Click Export. A file explorer will pop up and lets you navigate through your system folders. And you will be able to write a name for the .csv file you want to export/create.

Use case name: Delete entry from local database
Actor: User
Goal: Remove an entry from the table
Description: Pick the entry in the Currently Registered table [6.1], or from the search-result table. Click delete. A confirmation dialog will appear. Click OK to delete entry.

Use case name: Search for a specific object
Actor: User
Goal: Find all entries of a specific instances of an object
Description: Below the Searched table [6.1], there are text fields for all descriptive fields of an entry. Write part of the RFID, Location or pick the date you want to list out and click Search. This will fetch all entries matching the constraints given in the text fields.

Use case name: Adding an entry to the graph
Actor: User
Goal: Looking at the development of an object
Description: By clicking on an instance in the Currently Registered table [6.1] it will automatically be added to the graph. Click clear graph to start over.

Use case name: Change language
Actor: User
Goal: Changing the language in the application
Description: Click the Settings button. Hover your cursor above Language in the drop down menu. Click on the language to open the language drop down menu. Pick your desired language. The settings menu will disappear and the language has changed.



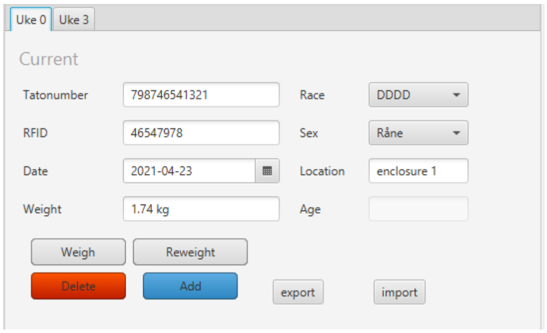
MAC:DC:0D:30:00:10 B3

Not connected

System port name: USB Serial Port (COM3)

Port description FT232R USB UART

(a) List of available communication ports and Bluetooth's MAC address in the red rectangle selected



Tatonumber	798746541321	Race	DDDD
RFID	46547978	Sex	Råne
Date	2021-04-23	Location	enclosure 1
Weight	1.74 kg	Age	

(b) Bluetooth scale not in reach

Figure 3.2: Screenshots of the GUI.

Expanded use-case descriptions

Connecting to scale	
Actors:	User
Purpose	Connect application to scale
Description	To configure the scale, go to settings, pick form of connection, and a list of either available Bluetooth scales or COM(communication ports) will pop up. Pick your scale and click connect.
Pre-conditions:	You will need a computer to run this application, as well as a scale able to connect through Bluetooth or a RS232 with the cable from the scale into the computer. Make sure that the scale is turned on.
Basic flow - RS232:	1: Go to settings 2: Pick RS232 as communications form the list of communication ports will appear [3.2]. Each list entry contains information of the system port name, by clicking on the list entry the addition information will emerge. This is in the format "Communication port(Com1)" and port description on the format "FT232R USB UART" or "Serial3". You are looking for the one belonging the scale, which for this USB to serial UART adapter is FT232R. 3: Pick the port matching your scale. 4: Click connect. You are now connected and ready to weigh.
Basic flow - Bluetooth:	1: Go to settings. 2: Pick Bluetooth as communications form in the settings menu and the list of available Bluetooth scales will appear. 3: Pick the one matching the scales MAC-address [3.2]. 4 : Click connect. You are now ready to weigh.
Alternate flow	You only need to configure the scale once, next time you open the application you can start weighing right away.
Exception flows:	*You connected to the wrong communications port:* If you still have the list up, switch to the correct port and click connect. If else you click the RS232 button to retrieve the list and pick the right one and click connect.

Table 3.1: Connecting a scale to the application

Add new entry to local database	
Actors:	User
Purpose	Add a pig's new weight information to the database.
Description	The user puts in the relating data of the pig [3.2]. Which includes Tatonumber, RFID, Date, Race, Sex, Location. Age will fill in automatically, depending on the date. Add current weight from scale. Then put this information to the Current Registered table.
Pre-conditions:	You will need a computer with this application installed, a pig, a scale connected to the application.
Basic flow:	1: fill in the data relating the pig, as given in the description. sex. 2: Click the Weigh button. The new weight will be added as soon as the scale sends its response. 3: Click the blue Add button. This will update the Currently Registered table [6.1] with this new entry.
Alternate flow	You pull up known data from a pig which have been weighted before, and therefore is already in the database. You can search for a pig in the database using either RFID or the Tatonumber. Then you get all the entries of the pigs containing those numbers. You can also sort your local database on pig enclosures. When you know you are going to weigh the pigs in enclosure 1 you can pull up those pigs and easily click on the pig you are going to weight to fetch its data. The date will be updated to current day and the only data you need is weight, this is done by clicking Weigh button.
Exception flows:	*You wrote the wrong RFID or Tatonumber* 1: Click the newly added entry 2: Click delete 3: Start over

Table 3.2: Use-case of how to add new entry in the local database

3.2 Functional requirements

The functional requirements in the Project description 1.1 can be summarized into the following points:

- Has to be made for desktop.
- Has to be able to read weight off of the scale.
- Has to be able to write the weight to anywhere using a hotkey.
- Requires support for multiple digital scales.

- Must be configurable
- Option for reading bar-code or RFID.
- Should be able to record timestamps as well as the weight.

As the requirements are very loose and not very extensive, we mostly had free reins to implement anything. Which is why we had regular meetings with Norsvin, to air our thoughts and ideas of what to implement next.

The end result of our list of functional requirements:

- Has to be made for desktop.
- Has to Be able to read weight off of both Bluetooth scales and RS232 scales.
- Has to be able to fetch weight from the scales using a global hotkey.
- Has to record timestamp when weighing.
- Global hotkey has to be configurable.
- Has to be able to store recordings locally.
- Has to be internationalized.
- Has to be able to export stored data.
- Has to be able to import external data.

3.3 Design requirements

The application is meant to be utilized out in the barn, where maybe the lighting and other conditions are not optimal for computer use. To account for those conditions, the application is required to have a minimalist design, paired with large buttons and large font sizes.

3.4 Security requirements

Our application will primarily be used in conjunction with logging the weight and date of animals, with no direct interaction with other existing applications/solutions. We had considered implementing a form of login procedure and encrypting data, but after consideration and discussion with Norsvin, we deemed it to be unnecessary, and a waste of the team's time. The reason being that Norsvin did not categorize the data available in our database as crucial or critical. Because in isolation, that data is not valuable for anyone, it has to be linked with other data and analyses, which are stored in Norsvin's central systems. Since our application is a separate and independent system, we refrained from adding login procedures because most users would very likely just use the same username and password that they already use in other services. Considering our team's nonexistent experience in securely managing passwords and other sensitive information, we would rather not expose our selves to the possibility of being a potential major breach in password security. Especially when the data behind the login barrier, can also be found on the computer in the form of exported .csv files 5.3.

3.5 Operational and non functional requirements

Although Norsvin didn't specify any operational explicit requirements, an intuitive, responsive and sturdy environment, is still implied for an application made with the sole purpose of improving existing solutions. With this in mind, we have gone though several points of consideration, and discussed their relevance with Norsvin 3.3.

Internationalization	Norsvin is an international company, with many costumers and partners throughout Europe. The application thus has to be made compatible with internationalization standards for easier future expansions.
Capacity/Throughput	Because the application is client-side, it will only administer a single user at once, who will be able to produce an estimated throughput of maximum 2 animals a minute. By that metric, we concluded that anything related to capacity and throughput was redundant, as it is influenced entirely by the host device - which is responsible for only a single user.
Reliability	Reliability the most important aspect of the application, considering that the only purpose for the existence of the application, is to make logging weights more convenient. The application is thus required to not crash, and prevent the user from inserting data that produces errors.
Storage	All data will be stored until the user chooses to delete it. And since 4.5 is practically a limitless solution in both storage and scale-ability, the only limiting factor would be the storage capacity of the host device. [27]
Windows desktop application	The application is required to at least support Windows as an operative system.

Table 3.3: Non functional requirements

Chapter 4

Technology

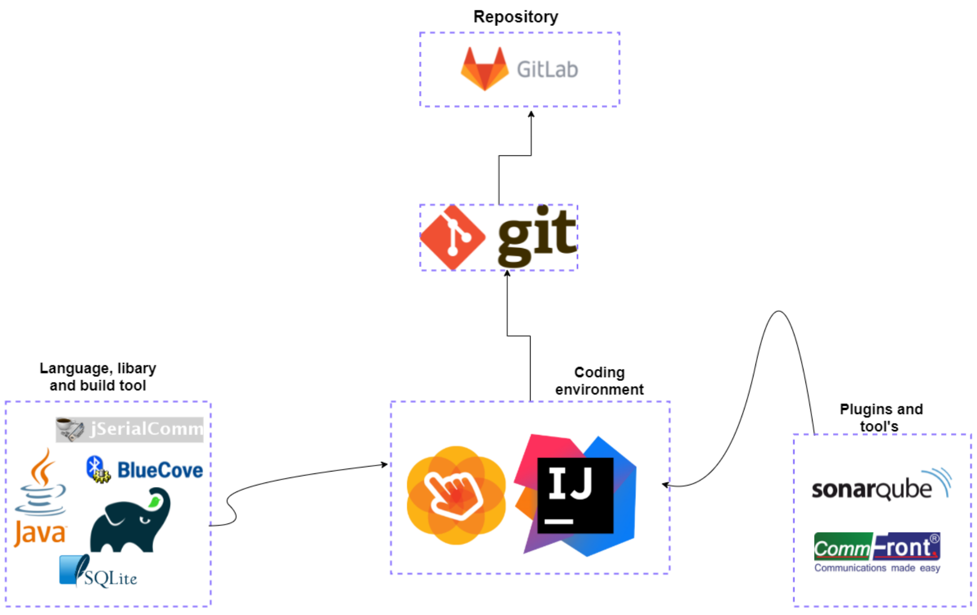


Figure 4.1: Overview of tools,language and software

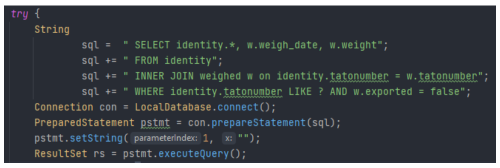
This chapter is dedicated to mentioning the technologies used throughout the project.

4.1 Java

Norsvin expressed their preference for Java as the coding language of choice for the application, because the in-house developers of Norsvin are familiar with it, and will be able to continue to support and further develop the application after we have delivered the proof of concept.

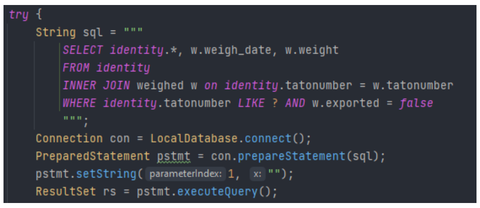
New futures with Java 15 compared to Java 8

With Java 15, they introduced the use of Text Blocks, which are very convenient when making SQL queries for the database. The coding experience is better in general because null-pointer errors are more expressive, and thus easier to find the source of the problem. And the try/catch statements are equipped with a more robust and easy to use local variable cleanup.



```java
try {
    String
            sql =  " SELECT identity.*, w.weigh_date, w.weight";
            sql += " FROM identity";
            sql += " INNER JOIN weighed w on identity.tatonumber = w.tatonumber";
            sql += " WHERE identity.tatonumber LIKE ? AND w.exported = false";
    Connection con = LocalDatabase.connect();
    PreparedStatement pstmt = con.prepareStatement(sql);
    pstmt.setString( parameterIndex: 1,  x: "");
    ResultSet rs = pstmt.executeQuery();
```

(a) SQL statement with pre Java 15 syntax.



```java
try {
    String sql = """
            SELECT identity.*, w.weigh_date, w.weight
            FROM identity
            INNER JOIN weighed w on identity.tatonumber = w.tatonumber
            WHERE identity.tatonumber LIKE ? AND w.exported = false
            """;
    Connection con = LocalDatabase.connect();
    PreparedStatement pstmt = con.prepareStatement(sql);
    pstmt.setString( parameterIndex: 1,  x: "");
    ResultSet rs = pstmt.executeQuery();
```

(b) SQL statement with new text block syntax.

Figure 4.2: Sample comparison between old and new syntax

Cons of using Java 15 instead of Java 8

The latest official standalone JRE is 8, which means all newer versions of java, is mostly for developers as only new JDK are released. And to get an executable file for consumers, the JAR file has to be turned into an executable file with an embedded JRE version 8. This can be done using jLink tool.

Why we chose to use Java 15 instead of LTS.

The downside of using Java 15 entirely on the fact that the latest official, standalone JRE is for Java 8. So anything made with a newer version has to be ported down to a compatible version. While this restraint is quite annoying, developing apps on a newer version of java is generally a much better experience as they make the language both more robust and easier to use in every new version, which is enough of a reason to use it. The figure 4.2 displays one new added difference in syntax. While using the new syntax, you also gain the benefit of having SQL syntax support within the text block, this also works with suggestions and auto complete of tables and columns from the database.

4.2 232Analyzer

Norsvin gave us a tips to start off communication with the scales by using 232Analyzer [28]. This software is a serial protocol analyzer, that displays information received from communication ports on the computer. We used this software for the initial connection and debugging the connection with the digital scales, because Norsvin had tested and confirmed that the software worked with the RS232 scale.

We could then utilize it as a benchmark to determine whether or not we had connected to the scale, and that the data received in our Java application mirrored the results from 232Analyzer.

4.3 BlueCove

Since the application should be able to communicate with a Bluetooth scale, we chose to use BlueCove. Which is a Java library for Bluetooth connection (JSR-82 implementation). BlueCove provides an further developed API of JSR-82, and can be used in Java Standard Edition (J2SE) 1.1 or newer [29]. BlueCove provides modules for discovery of nearby Bluetooth devices, identify services provided by the remote device, and access to information such as Bluetooth address, Bluetooth device name etc.

4.4 jSerialComm

We used the jSerialComm package for configuring UART communication and to access the communication ports through Java. This package is a library based on the old Java Communications API and RXTX, but with added functionality. This made it possible for us to list out the computer's available communication ports. The user is then able to select the port where the RS232 scale is connected.

4.5 Database - SQLite

The goal for the application, was for it to be a "plug and play" type of application. Where you only need to install the application, and be ready to use it. We chose to use SQLite, because SQLite is a server-less database and is self-contained. This is also referred to as an embedded database which means the DB engine runs as a part of the app. On the other hand, MySQL requires a server to run. MySQL will require a client and server architecture to interact over a network. With using SQLite, we minimize the preparation required for the application to be able to communicate with a database in a seamless "plug and play" manner, this also eliminates a lot of potential points of failure as well.

4.6 Coding environment

Intellij

Our team has been using Intellij Ultimate as our IDE. Mainly because it was the IDE we were familiar with, and partly because NTNU provides its students with a free licence.

Scenebuilder

Scenebuilder is a tool for making GUI interfaces with JavaFX. Scenebuilder uses a modulated drag and drop interface. By using Scenebuilder we had a fast and easy way of building our GUI, this let us focus more on the functionality of the application instead of the GUI.

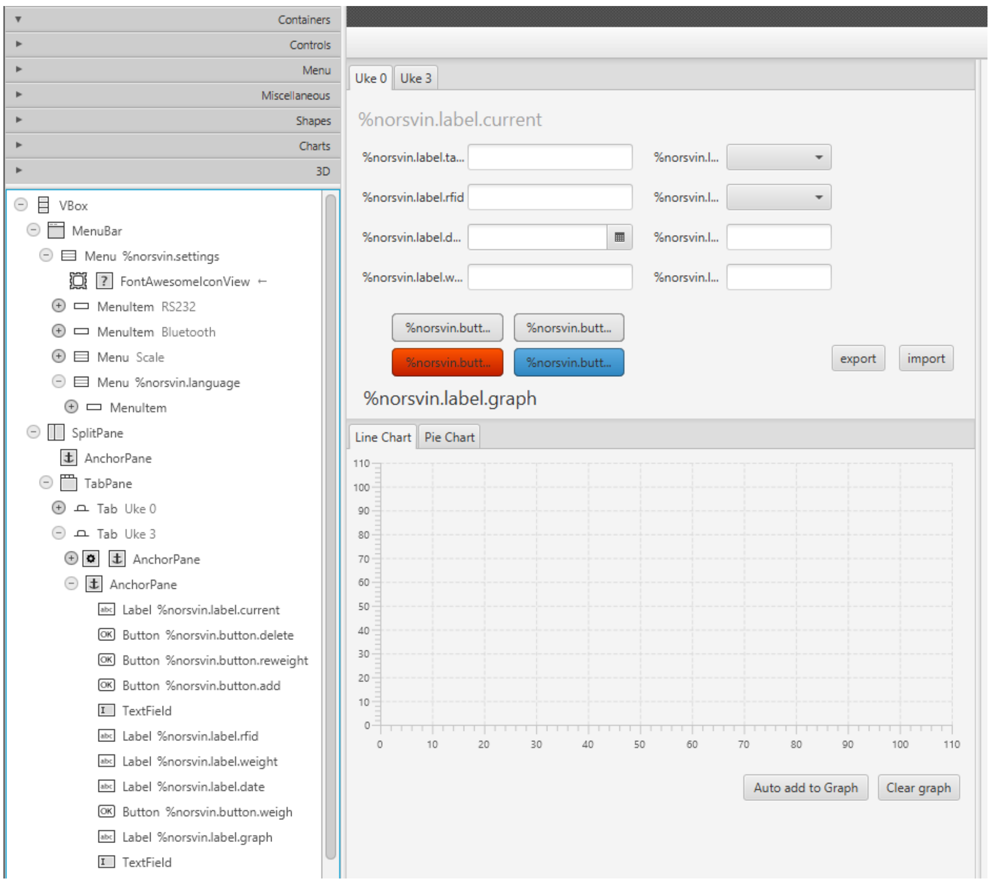
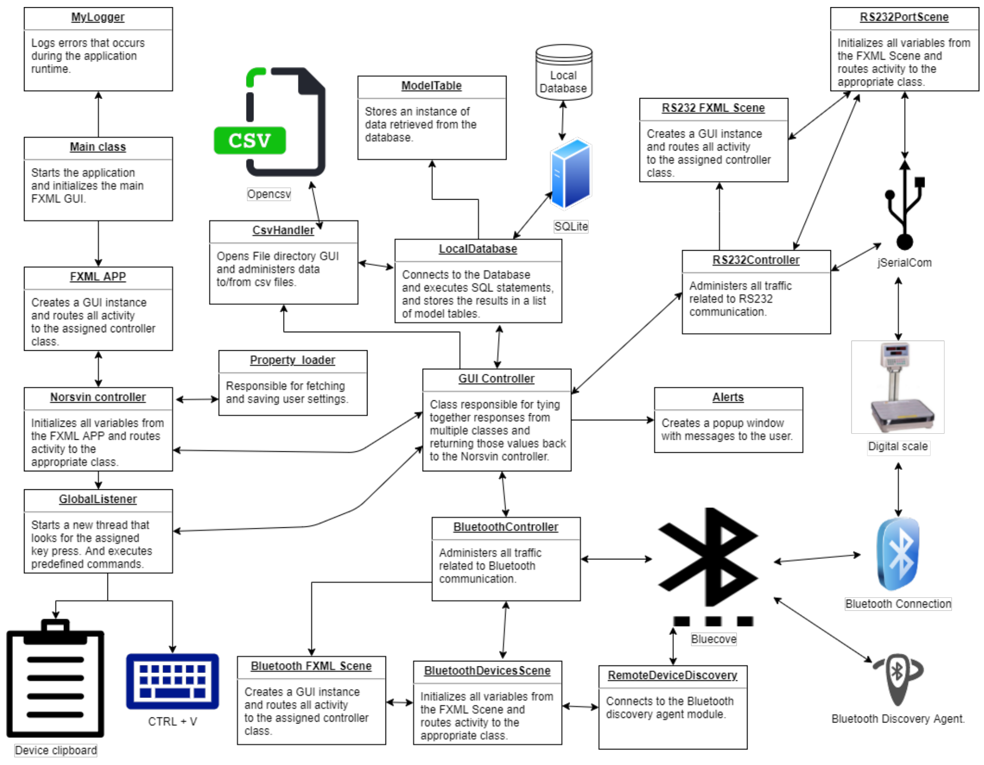


Figure 4.3: GUI under development in Scenebuilder

Chapter 5

Structure and implementation

Now we will take a closer look at how we have implemented the different parts of our system.

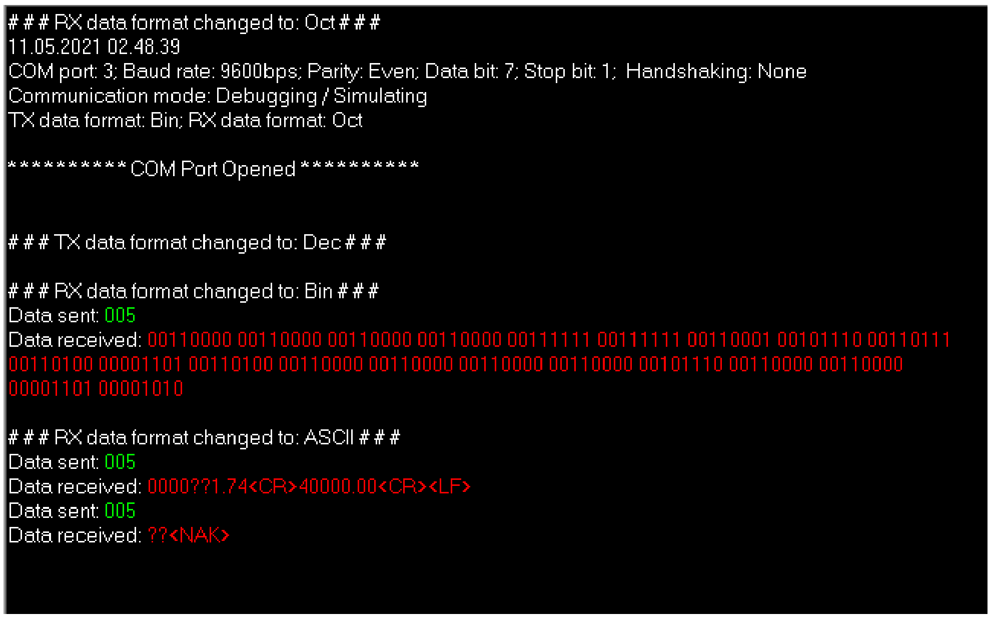


MyLogger
Logs errors that occurs during the application runtime.
Main class
Starts the application and initializes the main FXML GUI.
FXML APP
Creates a GUI instance and routes all activity to the assigned controller class.
Norsvin controller
Initializes all variables from the FXML APP and routes activity to the appropriate class.
GlobalListener
Starts a new thread that looks for the assigned key press. And executes predefined commands.
Device clipboard
CTRL + V
CSV
Opencsv
CsvHandler
Opens File directory GUI and administers data to/from csv files.
Property_loader
Responsible for fetching and saving user settings.
ModelTable
Stores an instance of data retrieved from the database.
Local Database
SQLite
LocalDatabase
Connects to the Database and executes SQL statements, and stores the results in a list of model tables.
GUI Controller
Class responsible for tying together responses from multiple classes and returning those values back to the Norsvin controller.
Alerts
Creates a popup window with messages to the user.
BluetoothController
Administers all traffic related to Bluetooth communication.
Bluetooth FXML Scene
Creates a GUI instance and routes all activity to the assigned controller class.
BluetoothDevicesScene
Initializes all variables from the FXML Scene and routes activity to the appropriate class.
RemoteDeviceDiscovery
Connects to the Bluetooth discovery agent module.
Bluecove
Bluetooth Connection
Bluetooth Discovery Agent.
RS232PortScene
Initializes all variables from the FXML Scene and routes activity to the appropriate class.
RS232 FXML Scene
Creates a GUI instance and routes all activity to the assigned controller class.
RS232Controller
Administers all traffic related to RS232 communication.
jSerialCom
Digital scale

Figure 5.1: Application structure chart.

5.1 Scale Connection

Both the Bluetooth and RS232 scales uses the RS232 protocol when transferring data, effectively transmitting data with the same format on both scales. This let us implement a uniform way of handling the data, regardless of the source port. The fact that both scales used the same standard also let us use the 232Analyzer to check whether or not there actually was a connection between the computer and the scale, by sending and receiving data.



```
# # # RX data format changed to: Oct # # #
11.05.2021 02.48.39
COM port: 3; Baud rate: 9600bps; Parity: Even; Data bit: 7; Stop bit: 1;  Handshaking: None
Communication mode: Debugging / Simulating
TX data format: Bin; RX data format: Oct

* * * * * * * * * * COM Port Opened * * * * * * * * * *


# # # TX data format changed to: Dec # # #

# # # RX data format changed to: Bin # # #
Data sent: 005
Data received: 00110000 00110000 00110000 00110000 00111111 00111111 00110001 00101110 00110111
00110100 00001101 00110100 00110000 00110000 00110000 00110000 00101110 00110000 00110000
00001101 00001010

# # # RX data format changed to: ASCII # # #
Data sent: 005
Data received: 0000??1.74<CR>40000.00<CR><LF>
Data sent: 005
Data received: ??<NAK>
```

Figure 5.2: Retrieving 1.74kg from the RS232-scale through 232Analyzer

We had to start a separate thread when sending a request to the digital scales, because the application would stop functioning until a response from the scale was received, which would last indefinitely if there was no connection to the scale. 7.3

Bluetooth

Getting connected

Before we started implementing the Bluetooth scale for the application. We had to test whether the Bluetooth scale was actually possible to communicate with. We had initially tried connecting with the scale, using three different computers, running either Windows or Linux, as well as four different phones, testing in Android and iOS, but to no avail. We could see the scale in the list of available devices, but connecting to the device was seemingly impossible.

One of our contacts from Norsvin had tried this himself, but was not able to connect to the scale with a stable connection either. We then decided to send an email in correspondence between Norsvin and the supplier of the scale(DIGI).

Where they informed us about parity, stop bit, bandwidth and which command to send to the Bluetooth scale to receive a response. Using RS232Analyzer and changing Bluetooth COM port settings with the correct stop bit and bandwidth etc., we finally got a response from the scale.

Non-fixed connection to the Bluetooth scale

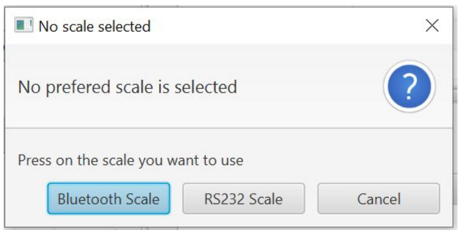
After a meeting with the client, they told us that they will be developing Bluetooth communication on a already existing android application that they use for a different purpose on a later stage. When the scale uses a fixed Bluetooth connection, no one else will be able to use the scale. This will cause problems where the same scale is used with different devices and applications, then the previously used device or application must be quit to make it available for others to use. But with a non-fixed connection, different applications and devices can use the scale at the same time, without having to quit the application to make the Bluetooth scale available.

Exceptions handling

Bluetooth connection with a non-fixed connection made exception handling a lot easier. Instead of having the application constantly pinging the scale to check if the scale is turned off or not in range. The Buffer reader and Bluetooth connection are closed after 5 seconds if no response is received. If the buffer reader is NULL, the user is notified that the weight is out of range and if the buffer reader is initialized but returns an empty String, the user is notified that the weight is turned off.

Implementation

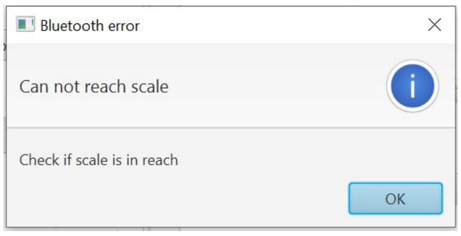
When implementing the Bluetooth scale, we have focused on the application being easy to use regarding the user's technical background. Information about error situations such as the Bluetooth scale being switched off, not being within reach should appear visible to the user. When using the application for the first time, the user will be notified when they press the "weigh button" to configure the scale. The user will be prompt with a message showcased in figure: 5.3(a) telling them to choose between Bluetooth scale or RS232 Scale. If the user selects Bluetooth option it will open up a new window that searches for all paired and non-paired devices that support SPP, Bluetooth communication.


No scale selected

No prefered scale is selected

Press on the scale you want to use

(a) Bluetooth no preferred scale selected


Bluetooth error

Can not reach scale

Check if scale is in reach

(b) Bluetooth scale not in reach

Figure 5.3: Alert message for Bluetooth

RS232

As mentioned in section 5.1, the format received from the scales is equal for both the RS232 scales and Bluetooth ones. The only difference is what port is used for communication. The farmers using the RS232 scales should be able to use either a RS232 port or use an RS232 to USB adapter to connect with the RS232 scale. We got a USB-adapter from Norsvin because RS232 is an older standard that does not often come with modern PC's. With jSerialComm we have retrieved a list of the available serial ports. From here we are able to open, connect and configure each port. This is used when searching for the scale's port and when setting the configuration to be able to communicate through RS232. To establish connection with the RS232 scale, we first have to change the port's baud rate, data-, parity- and stop bit equal to the proprietary settings of the scale before the connection can be used. [30] The code in figure 5.4 displays what parameters needs to be sent to the scale in order to connect and communicate with the RS232 scale.

```
public void connectToDevice( String comport){
    sp = SerialPort.getCommPort(comport);
    sp.setComPortParameters(9600, 7, 1, 2); // settings given from
     ↪  DIGI
    sp.setComPortTimeouts(SerialPort.TIMEOUT_WRITE_BLOCKING, 0, 0);
     ↪  // block until bytes can be written
}
```

Figure 5.4: Code snippet from port communication with the use of jSerialComm

5.2 Database

Database Structure

As showcased in the logical model [5.5], our database consists of two tables, where the Identity table contains all the static data and Weighed table contains data that will change between each individual weighing of an animal. This is done to have an easily accessible unique primary key, while still allowing new data to be recorded on that specific key.

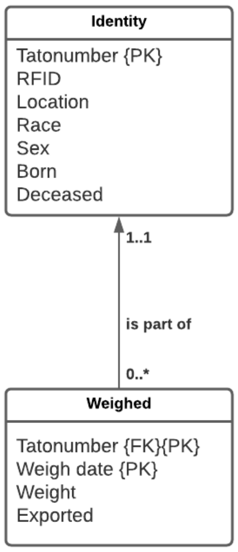


Figure 5.5: Logical model of the database.

Identity table

All data in this table will contain only information about the identity of the animal, which is information that does not change throughout its lifetime. Storing only this type of data in one table is beneficial because all animals added to the list will not need multiple entries, and all animals in the list will stay unique. Keeping all animals unique in the table, lets us use a single point of reference as a Primary key. This lets us check the database for a specific animal with using only the Tatonumber or RFID.

Weighed table

This table contains mostly data that changes with every new entry, with the sole exception of the Tatonumber, which ties the animals added in weighed table with the identity table. Here we have used the combination of Tatonumber and the date as a primary key, which allows for multiple entries of the same Tatonumber and date, but only a single entry with the same date and Tatonumber. This setup will restrict accidental weighing of the same animal multiple times on the same date. We chose to use date as the latter part of the primary key instead of the weight because the weight is way too unstable/variable, and can not guarantee only a single weighing on any given day.

Search Filter

Because the user can use the search table as a convenient way of automatically filling essential information into the text fields, we had to make a robust search filter, capable of precisely restricting the search results to fit what the farmer is looking for. Figure 5.6 is a snippet of our code, displaying all the restricting parameters in the filter.

```
public static boolean search(String[] filter,
 ↪  ObservableList<ModelTable> list) {
...
String sql = """
        SELECT identity.*, w.weigh_date, w.weight
        FROM identity
        LEFT JOIN weighed w on identity.tatonumber = w.tatonumber
        WHERE
        (identity.tatonumber LIKE ? OR identity.rfid LIKE ?)
        AND (identity.location LIKE ?)
        AND (identity.race LIKE ?)
        AND (identity.sex LIKE ?)
        AND (w.weigh_date BETWEEN ? AND ?)
        AND (w.weight BETWEEN ? AND ?)
        AND (w.exported = ?)
        AND (identity.deceased = ?)
        """;

try (Connection con = LocalDatabase.connect()) {
    PreparedStatement pstmt = con.prepareStatement(sql);
    pstmt.setString(1, "%" + filter[0] + "%");
    pstmt.setString(2, "%" + filter[0] + "%");
    pstmt.setString(3, "%" + filter[1] + "%");
    pstmt.setString(4, "%" + filter[2] + "%");
    pstmt.setString(5, "%" + filter[3] + "%");
    pstmt.setDate(6, Date.valueOf(filter[4]));
    pstmt.setDate(7, Date.valueOf(filter[5]));
    pstmt.setFloat(8, Float.parseFloat(filter[6]));
    pstmt.setFloat(9, Float.parseFloat(filter[7]));
    pstmt.setString(10,  filter[8] );
    pstmt.setString(11,  filter[9] );
...
```

Figure 5.6: Code extract from database search filter function.

5.3 CSV export and import

The initial plan was to make the application automatically sync the local database with Norsvin's central database, but we didn't get access to that database in time. In consideration to 3.4, our team and Norsvin instead agreed to operate though exporting and importing the local data into .csv files. Norsvin will implement import and export functionality in their central database, that fits our import and export format. As the export/import functionality of .csv files amended into the project in the latter part of the development time-frame, we had to refactor a bit of our data handling processes, from a predefined parameters to lists, which can contain multiple variations of parameters. The transition to lists was made to simplify the code and remove the necessity for having multiple methods of importing data, which had to be run depending on the parameters fetched from the imported file - we now run a single method that checks the amount of parameters in a list, and adds the imported data to the database in an appropriate manner.

Export

Initially we had no way of distinguishing between newly weighed animals and the previously weighed ones, which made it impossible to select only the newly registered animals, and ignoring previously exported ones. Our solution was to implement a boolean "exported" in the weighed table to signify if it has been exported previously. The boolean is set to be false at default, making all newly registered weights "tagged" as not exported, this boolean will then be updated to true once the export function has been completed successfully.

Import

The import function supports multiple import "formats", in the way that you can choose to import only the Tatonumber, the whole identity or entire weighing records. We ended up using the simplest method possible - using a switch, and run different methods depending on how many parameters are included in each row. Where 6 parameters will signify the import of all identity data points, 8 parameters will add entire records while anything else will only import the Tatonumber. While this method is both simple and effective, it is not particularly robust because it assumes that everything is imported in a specific order, with specific values. Though this should in theory not pose a problem, as the method is made specifically for importing .csv files tailored for the application. But prevention from accidentally trying to import the wrong file is still very important because of how the switch is set up. To ensure that nothing but real Tatonumbers are inserted into the database, we first have to check if the format of column[0](See 5.7) matches what real Tatonumbers should look like. Without utilizing such a filter, literally any .csv file, with any amount of data exempt for those that has 6 and 8 columns would pass as a valid object, and would get imported.

```
public void addToDBFromImportList(List<String[]> list) throws
 ↪  SQLException {
...
    switch (s.length) {
        case 6 -> attemptInsertIdentity(s[0], s[1], s[2], s[3],
         ↪  s[4], LocalDate.parse(s[5]));
        case 8 -> {
            attemptInsertIdentity(s[0], s[1], s[2], s[3], s[4],
             ↪  LocalDate.parse(s[5]));
            attemptUpdateweighed(s[0], LocalDate.parse(s[6]), s[7]);
        }
        default -> attemptInsertIdentity(s[0], null, null, null,
         ↪  null, null);
...
}
```

Figure 5.7: Function that selects how to Import data from selected file.

```
private static boolean regexPass(String tato) {
    Pattern pattern = Pattern.compile("([A-z]{2})(\\d{8})");
    Matcher matcher = pattern.matcher(tato);
    return matcher.find();
}
```

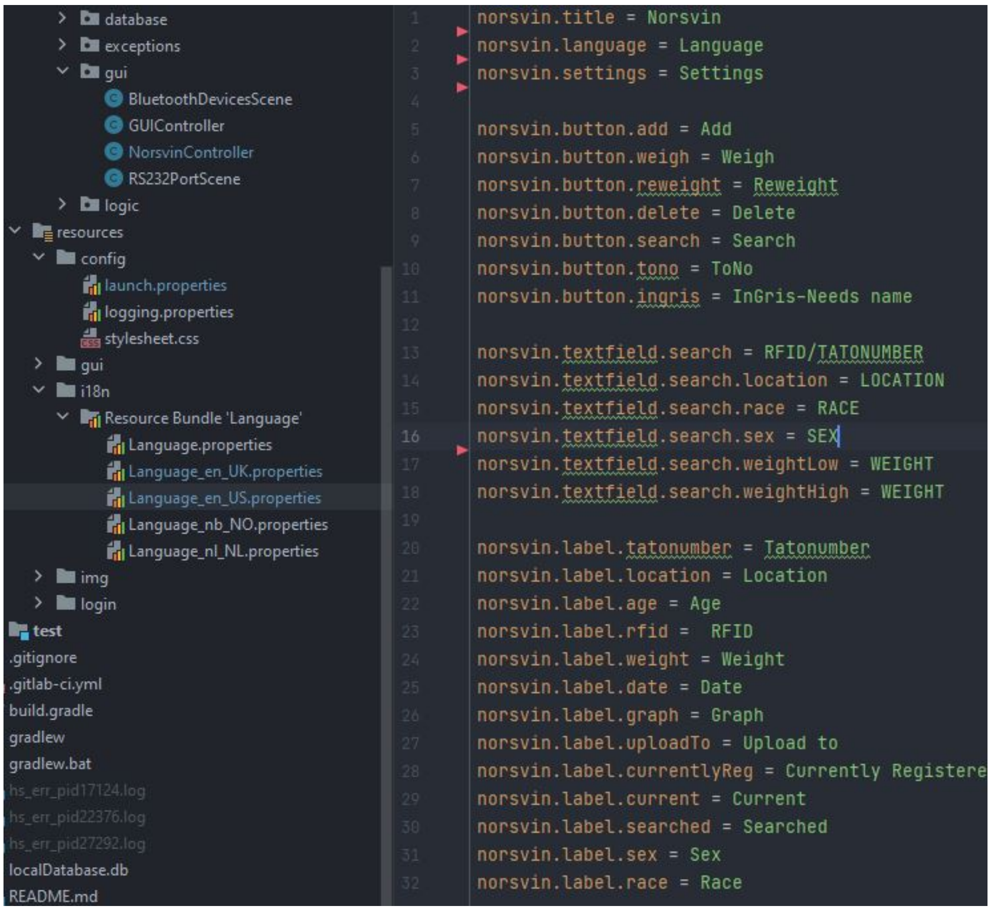
Figure 5.8: Function that returns true or false if the parameter fits the pattern limitations.

5.4 Internationalizing

We decided to implement Norwegian, English and Dutch languages as the default supported languages because Norsvin's demographic is mostly either Norwegian or Dutch, we decided to add English as well because it is very universal. The internationalization system has been implemented using the i18n and l10n standards[31], where language is automatically changed to the user's selected OS language at startup. Language can also be changed manually by the user, this change will be stored in a property file 5.5 that will remember the selected language on the next startup. All language options are stored in our ResourceBundle using the i18n standard for naming the language packs, this makes the language packs compliant with locations and language returned through Java (See 5.9). The Locale.getDefault() function returns the user's OS language for example (en-US) for English, and selects the correct language property file from the language bundle. Every button and text is internationalized with the use of a internationalized string, that is read from the language property file. (See code snippet: Fig: 5.10)

```java
@Override
public void start(Stage primaryStage) throws IOException {

...
/* The primary stage. */
    //Language bundle, i18n. Finds users pref. language
    ResourceBundle bundle =
     ↪  ResourceBundle.getBundle("i18n.Language",
     ↪  Locale.getDefault());
    //Load scene with selected language.
    //Loads the FXML GUI files.
    FXMLLoader loader = new
     ↪  FXMLLoader(getClass().getResource("/gui/Main.fxml"),
     ↪  bundle);
...
```

Figure 5.9: Code snippet for loading language bundle and getting Local language



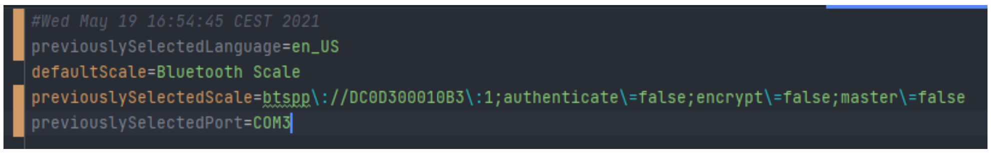
```
    > database
    > exceptions
    v gui
        BluetoothDevicesScene
        GUIController
        NorsvinController
        RS232PortScene
    > logic
v resources
  v config
      launch.properties
      logging.properties
      stylesheet.css
  > gui
  v i18n
    v Resource Bundle 'Language'
        Language.properties
        Language_en_UK.properties
        Language_en_US.properties
        Language_nb_NO.properties
        Language_nl_NL.properties
  > img
  > login
test
.gitignore
.gitlab-ci.yml
build.gradle
gradlew
gradlew.bat
hs_err_pid17124.log
hs_err_pid22376.log
hs_err_pid27292.log
localDatabase.db
README.md
```

```
norsvin.title = Norsvin
norsvin.language = Language
norsvin.settings = Settings

norsvin.button.add = Add
norsvin.button.weigh = Weigh
norsvin.button.reweight = Reweight
norsvin.button.delete = Delete
norsvin.button.search = Search
norsvin.button.tono = ToNo
norsvin.button.ingris = InGris-Needs name

norsvin.textfield.search = RFID/TATONUMBER
norsvin.textfield.search.location = LOCATION
norsvin.textfield.search.race = RACE
norsvin.textfield.search.sex = SEX
norsvin.textfield.search.weightLow = WEIGHT
norsvin.textfield.search.weightHigh = WEIGHT

norsvin.label.tatonumber = Tatonumber
norsvin.label.location = Location
norsvin.label.age = Age
norsvin.label.rfid =  RFID
norsvin.label.weight = Weight
norsvin.label.date = Date
norsvin.label.graph = Graph
norsvin.label.uploadTo = Upload to
norsvin.label.currentlyReg = Currently Registere
norsvin.label.current = Current
norsvin.label.searched = Searched
norsvin.label.sex = Sex
norsvin.label.race = Race
```

Figure 5.10: Internationalized strings for English

5.5 Property file

As the application should simplify and make the process of weighing pigs faster, storing configurations such as selected Bluetooth-scale/RS232-scale or preferred language, will be stored automatically in the application's configuration file. This configuration file will be loaded at startup, and will override the standard configuration of the application 5.11.



```
#Wed May 19 16:54:45 CEST 2021
previouslySelectedLanguage=en_US
defaultScale=Bluetooth Scale
previouslySelectedScale=btspp\://DC0D300010B3\:1;authenticate\=false;encrypt\=false;master\=false
previouslySelectedPort=COM3
```

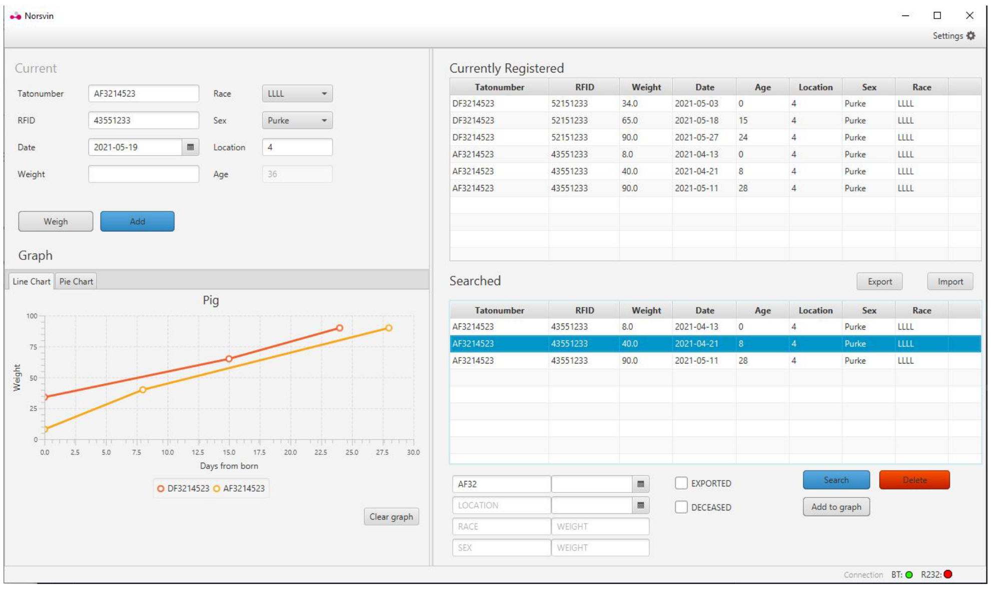
Figure 5.11: Stored settings, which will be loaded on next startup

Chapter 6

User interface

6.1 Layout

The layout of the application has been designed according to the requirements specified in section: 3.3. Under "Current" in the top left corner of the GUI is where the user register the pig (See figure: 6.1). Here RFID is read using a RFID scanner, the date is automatically set to the current system date, and the weight is entered when the user clicks the "Weigh" button. As Norsvin only has four different races in which they breed, we have chosen to have a static combo-box, where the user selects from the drop down list. We have done the same with sex. Both of these remains the same at the next weighing, as the user most likely only has one type of race. To make the process of weighing pigs even faster.



Current

Tatonumber	AF3214523	Race	LLLL
RFID	43551233	Sex	Purke
Date	2021-05-19	Location	4
Weight		Age	36

Currently Registered

Tatonumber	RFID	Weight	Date	Age	Location	Sex	Race
DF3214523	52151233	34.0	2021-05-03	0	4	Purke	LLLL
DF3214523	52151233	65.0	2021-05-18	15	4	Purke	LLLL
DF3214523	52151233	90.0	2021-05-27	24	4	Purke	LLLL
AF3214523	43551233	8.0	2021-04-13	0	4	Purke	LLLL
AF3214523	43551233	40.0	2021-04-21	8	4	Purke	LLLL
AF3214523	43551233	90.0	2021-05-11	28	4	Purke	LLLL

Searched

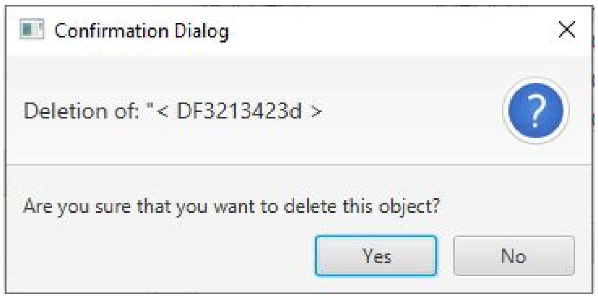
Tatonumber	RFID	Weight	Date	Age	Location	Sex	Race
AF3214523	43551233	8.0	2021-04-13	0	4	Purke	LLLL
AF3214523	43551233	40.0	2021-04-21	8	4	Purke	LLLL
AF3214523	43551233	90.0	2021-05-11	28	4	Purke	LLLL

AF32
LOCATION
RACE
SEX
WEIGHT
WEIGHT
EXPORTED
DECEASED

Figure 6.1: Graphical user interface

Button layout

The buttons for weight, delete and add have been made large. So that it is easy to press, even if the user wears gloves, or uses a pc with a small screen. For delete button we have chosen to use a red color, as red commonly associate with potential risk. Making the user take a pause before pressing it. [32] But we have also made sure that if accident happens the user will be prompt with a alert window, to confirm the deletion. See Fig: 6.2 how this is displayed.


Confirmation Dialog

Deletion of: "< DF3213423d >

Are you sure that you want to delete this object?

Figure 6.2: Alert confirming the deletion

Tables

"Currently registered" shows an overview of pigs that the user has registered and not exported yet. We have also added a table where the user can search for pigs with a search filter 5.2. This will simplify the weighing process for the user, when they want to weigh from only a certain location, or a specific sex etc. When the user clicks on one of the objects in either the "Searched" or "Currently Registered" tables, all static information from that specific object will automatically be transferred over to the text fields used for adding new entries, leaving them with only having to click on the weigh button before adding the newly recorded weight.

Graph

We have also implemented a graph window, where the user will receive a line chart of the selected pig, with age on the X-axis and weight on the Y-axis. Here the user has the opportunity to compare progress with the other pigs. When "Add to graph" is clicked (See Fig: 6.1). Every object in the search table is inserted into the graph window.

6.2 Connecting To A Scale

Usage

We added a progress indicator when searching for devices to let the user be aware that the program is searching for devices (See Fig: 6.3), because BlueCove will usually spend some time searching for devices before returning a list of results. We created two classes that utilizes features from BlueCove. One of the classes uses the RemoteDeviceDiscovery package, which fetches devices already paired with system and all devices in range (Without pair). These devices are then added to a vector list and returned to ServicesSearch class. Which checks what services the different Bluetooth devices offers. All devices that offer SPP services is placed in a Hashmap, together with Bluetooth name and it's connection URL string that might look like:

btspp://DC0D300010B3:1;authenticate=false;encrypt=false;master=false

where MAC/Bluetooth address for the device is "DC0D300010B3", "1" is the SPP services, and the last parameters is related to security and roles. When the search for available Bluetooth scales is complete, all found devices will be displayed, the user must then select the device that has a MAC/Bluetooth address or model name corresponding with what is printed onto the scale. Once the user has selected the correct scale and connected, the specific choice of Bluetooth or RS232 will be stored in a property file.

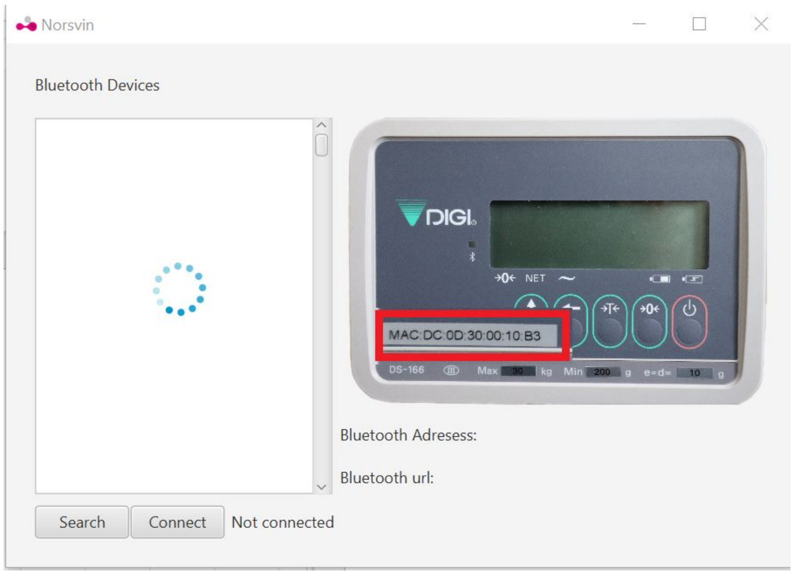


Figure 6.3: Progress indicator spinning, indicating that the application is searching

6.3 FXML

FXML is an XML-based language designed to build the user interface for JavaFX applications [33]. Where you can use FXML to build an entire Scene, instead of doing it directly in the source code. This allows separating front-end logic from back-end logic, and internationalizing 5.4 can be localized as the file is read. [34]

(See code snippet on how an localized file is read: Fig: 5.9 and how a localized file for (en_US) looks like. Fig: 5.10).

Front-end structure

Fig: 6.4 shows an overview of controllers that communicate with the individual GUI scenes.

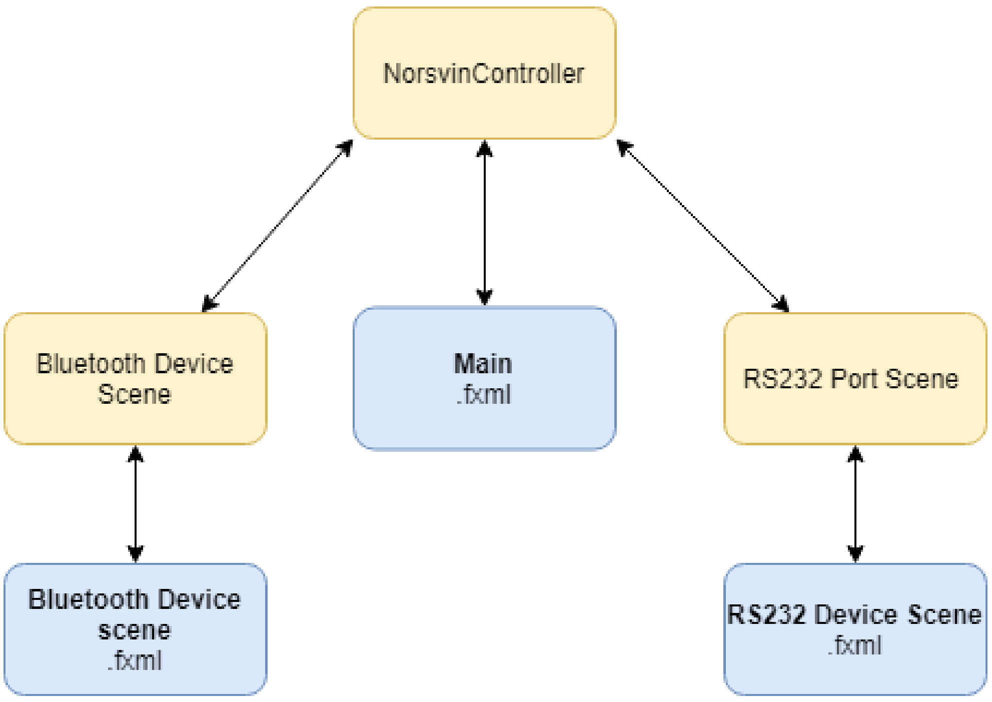


Figure 6.4: FXML Structure

NorsvinController: Is theController for the main GUI, which controls the other classes. Since the main controller should be able to control the entire front-end, and ensure loose coupling and high cohesion. Calculations and other manipulations of the data are done in separated classes.

Main.fxml: Is the main view of the application, and only communicates with the main controller(NorsvinController).

BluetoothDeviceScene: Controls bluetooth configuration and BluetoothDeviceScene.fxml, and controls all communication between Bluetooth weight and NorsvinController.

BluetoothDeviceScene.fxml: Is the actual Bluetooth device search view, and only communicates with the BluetoothController class.

RS232PortScene: Controls RS232 configuration and RS232DeviceScene.fxml, and controls all communication between RS232 weight scale and NorsvinController.

RS232DeviceScene.fxml: Is a separate view where device search and connection is configured, and only communicates with the RS232Controller class.

Controllers

One of the major problems at the start of our project, was the actual layout of the code. In the beginning we had divided the front of the application into three controller classes. The first one was the actual framework of the application, where the second controller class controlled the actual communication to the hardware and weighing, and the last controller class controlled the tables. The problems started when we wanted to share information between the controller classes. This was not as easy as with regular classes. As controller classes had to be passed between classes at load. After trying out different methods, and spending a good amount of time on this. We agreed to create the project with one controller class, which talks to the interfaces and where the actual calculation and communication with hardware was done in separate classes.

Chapter 7

Testing and code quality

7.1 Procedure

Our team has been using four methods for testing our code while making the application.

- Basic functionality testing.
- Code review.
- Single-user performance testing.
- Static code analysis.

Basic functionality testing

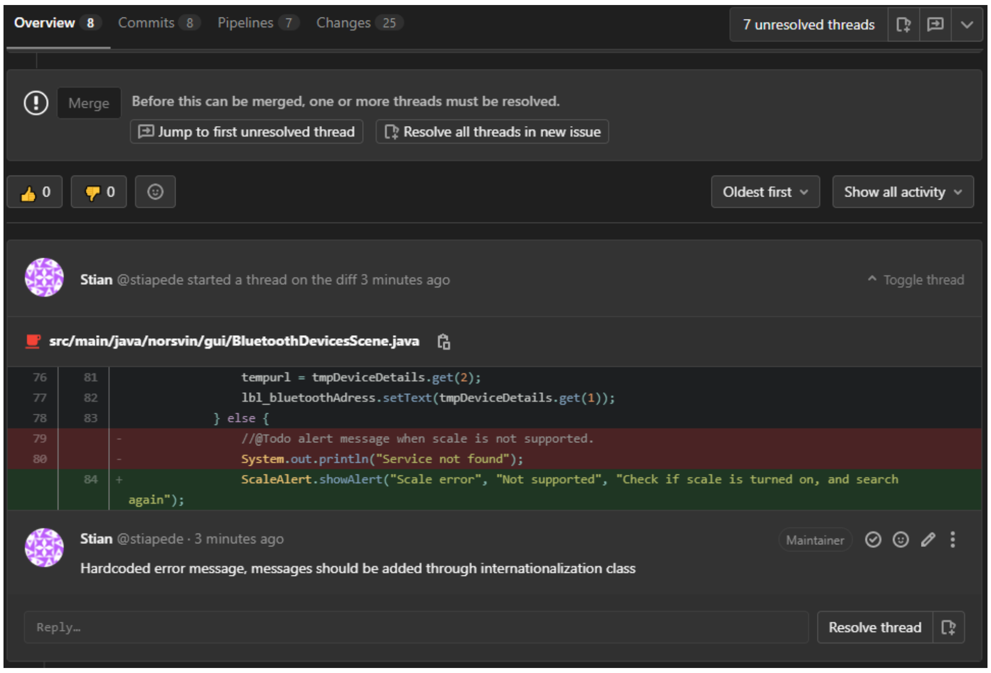
We have used Basic functionality testing for new code as soon as it is written. Here we test if the code does what it is supposed to do when correct data is entered. This testing is done by the developer that made the code, and has been our most extensively used test because we used many packages that we had no previous experience with, and continuously testing their basic functionality is a very fast way of learning how they operate.

Code review

As mentioned in the Project Management 2.2, we implemented rules regarding code reviews before code could be merged master. While this measure has not been very effective at catching errors, it has helped us maintain a relatively uniform code structure and code quality. 7.1

Single-user performance testing.

We used Single-user performance testing after a module/functionality had successfully been implemented ie. the entire module has passed the basic functionality test. By using this approach, we could get an overview of the module as a whole, and have an easier time debugging as well as having a better overview of



Before this can be merged, one or more threads must be resolved.

Stian @stiapede started a thread on the diff 3 minutes ago

src/main/java/norsvin/gui/BluetoothDevicesScene.java

```
                tempurl = tmpDeviceDetails.get(2);
                lbl_bluetoothAdress.setText(tmpDeviceDetails.get(1));
            } else {
-               //@Todo alert message when scale is not supported.
-               System.out.println("Service not found");
+               ScaleAlert.showAlert("Scale error", "Not supported", "Check if scale is turned on, and search
    again");
```

Stian @stiapede · 3 minutes ago

Hardcoded error message, messages should be added through internationalization class

Figure 7.1: Code review in Git

potential refactoring. When preforming the Single-user performance test, we try our best to make the code fail, by putting in wrong data or turning off components etc. Our application is also made to only operate for a single user at a time, which makes the Single-user performance test double as a simulated workload as an end user. Section 7.3 delves into concrete examples of what the Single-user performance testing has helped us identify and fix.

Static code analysis.

We have used SonarQube as our static code analyzer, but mainly for code quality. We didn't use SonarQube regularly, and instead opted to use it when lots of new functions had been implemented. Because of this choice, rather than regularly maintaining the code, we had sessions dedicated to implementing bug fixes and improving code quality 7.4

7.2 User testing

We wanted to arrange a testing session with one of Norsvin's farmers, to observe if what we as developers think of as self explanatory and easy to use, is actually perceived as such for the end user. But sadly we didn't get to have any tests with real workloads due to COVID-19 restrictions. By having an actual end user testing out the application in a typical work environment, we could get concrete and unbiased feedback on every aspect end users come into contact with. As such, this kind of testing would be our most rewarding test, regarding usability and general user experience, and no amount of internal revisions could replace the value of having an actual end user test the application.

7.3 Important discoveries

Thread lockdown upon no response from scales

While doing the Single-user performance test, we found a bug that crashes the program when requesting weight from the connected scale. This happens when you first connect to the scale, and the scale turns off, making it unable to send a response to the application. The problem stems from the way we initialized the request, since everything ran on the same thread, the application would just get stuck waiting for a response. To prevent this kind of error from occurring, we had to start a new thread[35] where we could send the request to the scale, this lets the application do other stuff while it is waiting for a response from the scale. And after utilizing multiple threads, we were also able to set a timer on the tasks, and eliminate the task if no response is received within a specified time.

In addition to adding multiple threads, we avoided having a fixed connection to the scale. Instead, the scale is connected when the user presses the weigh button and closes after response. The application sends a command (005) to the Bluetooth scale that requests weight. Responses from the scale are then read and if the result is null, a message is sent to the user of the application that the scale is not within range Fig:5.3, and if the response message is empty. The user is notified that the scale must be switched on.

Weighing the same animal multiple times

After first implementing the database and finished our basic functionality test, we had forgotten that the current version of the database was only a rudimentary database, designed only for testing and learning how the interaction between SQLite and Java worked. Leaving the database with only a single table to contain all data. This worked perfectly until we tested more extensively and were not able to add multiple weights to a given Tatonumber.

```
public void discoverBluetoothDevices() {
    final int[] intDevicePosition = {0};
    ServicesSearch ss = new ServicesSearch();
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
            //Starts/visible the progress indicator. To indicate
             ↪  that search has started.
            progressIcon.setVisible(true);
            //Search for new bluetooth devices.
            mapReturnResult = ss.getBluetoothDevices();
            for (Map.Entry<String, List<String>> entry :
             ↪  mapReturnResult.entrySet()) {
                //Only add device with SPP service to list. As SPP
                 ↪  service will be the 3rd entry.
                if(entry.getValue().size() > 2){
                list.add(entry.getValue().get(0));
                //Map result to a hashMap.
                mapDevicePosition.put(intDevicePosition[0],
                 ↪  entry.getValue());
                intDevicePosition[0]++;
                }
            }
            //Sets result from bluetooth search to gui list.
            deviceList.setItems(FXCollections.observableList(list));
            //Turns of progress indicator, to indicate that search
             ↪  is over.
            progressIcon.setVisible(false);
        }
    });
    thread.start();
}
```

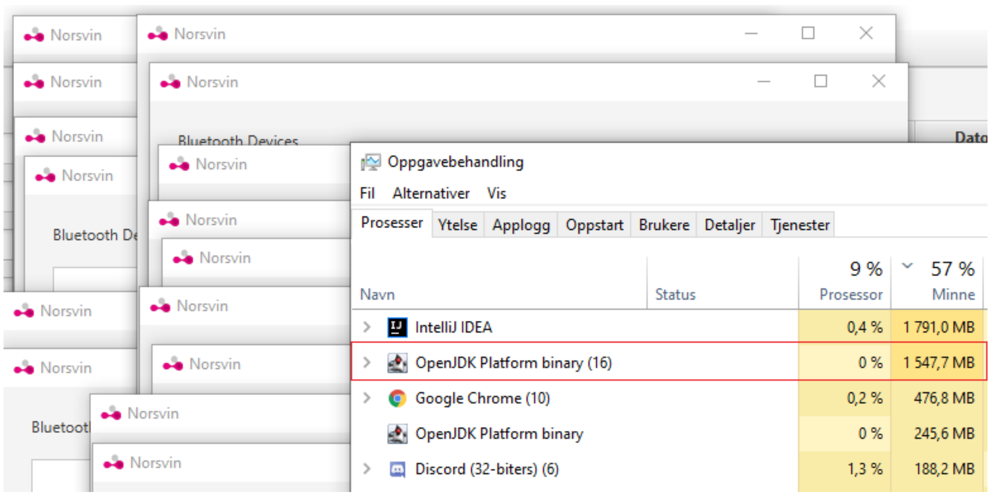
Figure 7.2: New thread created for searching after Bluetooth devices

Excessive memory usage

We noticed this error quite late into the development process, so we didn't have enough time to diagnose what the specific cause of the problem is, but we suspect that the problem stems from initializing new instances of objects and classes multiple times, without removing or reusing the already initialized ones, snippet of suspicious code 7.3. The showcased code will always initialize new instances whenever it is being executed, this lets the application start multiple unnecessary objects, that are also very difficult to manage. Figure 7.4 showcases the result of opening multiple instances of the Bluetooth and RS232 device searching scenes.

```
...
ResourceBundle bundle = ResourceBundle.getBundle(BASE_LOCATION,
 ↪  Locale.getDefault());
//Load scene with selected language.
// The loader.
FXMLLoader loader = new
 ↪  FXMLLoader(getClass().getResource("/gui/BluetoothDevices" +
 ↪  ".fxml"), bundle);
Parent root = loader.load();
Scene scene = new Scene(root);
...
```

Figure 7.3: What we suspect to be the main culprit for excessive memory usage



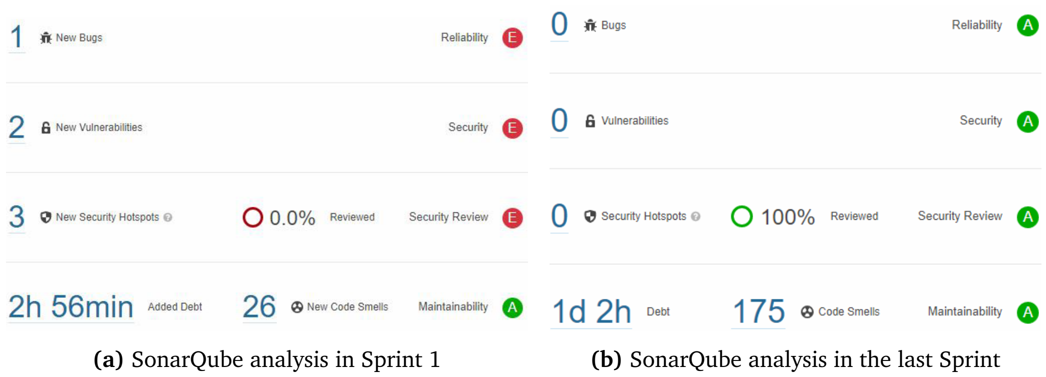
Navn	Status	9 % Prosessor	57 % Minne
IntelliJ IDEA		0,4 %	1 791,0 MB
OpenJDK Platform binary (16)		0 %	1 547,7 MB
Google Chrome (10)		0,2 %	476,8 MB
OpenJDK Platform binary		0 %	245,6 MB
Discord (32-biters) (6)		1,3 %	188,2 MB

Figure 7.4: Screenshot of multiple remote device scenes and memory usage.

7.4 Code quality

SonarQube

To ensure better code quality and code security, we used SonarQube on our code-base. SonarQube is an open-source platform developed by SonarSource. It analyzes for duplication of code , complexity, bugs and vulnerabilities to ensure a clean, maintainable and secure code-base [36]. The first weeks of developing, SonarQube was used to clean up existing code and to detect security vulnerabilities. But further into development, this was not done continuously, which led to us having to make major changes towards the end of the project. This is a screenshot (Fig 7.5) of one of our SonarQube analysis at first Sprint, when most of the GUI, controller class and simple features were implemented.


1 New Bugs Reliability E
2 New Vulnerabilities Security E
3 New Security Hotspots 0.0% Reviewed Security Review E
2h 56min Added Debt 26 New Code Smells Maintainability A

0 Bugs Reliability A
0 Vulnerabilities Security A
0 Security Hotspots 100% Reviewed Security Review A
1d 2h Debt 175 Code Smells Maintainability A

(a) SonarQube analysis in Sprint 1 **(b)** SonarQube analysis in the last Sprint

Figure 7.5: SonarQube analysis, before and after refactoring

Refactoring

SonarQube helped us find code that was creating memory leak. When opening up a new BufferedReader, InputStream or any new resource . We had to close the stream inside a finally clause or with the use of a try-with-resources Statement.[37] Instead of closing the resource inside a try catch, where it may happen that the application gets an exception and that the current resource is not getting closed. (See code snippet, Fig:(7.8, 7.9) and with the use of try-with-resource Statement, (See code snippet, Fig: 7.7 7.6

```
public void exportCsv(String fileNameAndLocation) throws Exception {
    //Instantiating the CSVWriter class
    CSVWriter writer = new CSVWriter(new
     ↪  FileWriter(fileNameAndLocation));
        writer.writeAll(list);
        writer.flush();
    }
```

Figure 7.6: Before refactoring

```java
public void exportCsv(String fileNameAndLocation) throws Exception {
    //Instantiating the CSVWriter class
    try(CSVWriter writer = new CSVWriter(new
     ↪  FileWriter(fileNameAndLocation))) {
        writer.writeAll(list);
        writer.flush();
    }catch(Exception e){
        logg.log(Level.WARNING,"Export Csv: ", e);
    }
}
```

Figure 7.7: After refactoring for memory leak with use of try-with statement

```java
public List<String[]> importCsv(String fileNameAndLocation) throws
 ↪  IOException {
    Reader reader =
     ↪  Files.newBufferedReader(Paths.get(fileNameAndLocation));
    CSVReader csvReader = new CSVReader(reader);
    csvReader.close();
    return csvReader.readAll();
}
```

Figure 7.8: Before refactoring for memory leak

```java
public List<String[]> importCsv(String fileNameAndLocation) throws
 ↪  IOException {
    CSVReader csvReader = null;
    try(Reader reader =
     ↪  Files.newBufferedReader(Paths.get(fileNameAndLocation))){
        csvReader = new CSVReader(reader);
    }catch(Exception e){
        logg.log(Level.WARNING,"Import Csv: ", e);
    }finally{
        assert csvReader != null;
        csvReader.close();
    }
    return csvReader.readAll();
}
```

Figure 7.9: After refactoring for memory leak

Chapter 8

Conclusion

8.1 Result

Norsvin wanted a desktop application with the ability to operate as a service and retrieve the weight in a fast and simple manner, for their existing solutions. After a few meetings with Norsvin we came to the conclusion that the application should serve as both a background service and be a more feature-rich application containing the functionalities and possibilities to be an alternative to existing solutions. We opted out of the mobile application and instead kept expanding Norsvin's wishes for a desktop application in collaboration with them.

With our solution, the user spends less time logging the animals, and will have the option of spending that time on increasing the animal welfare instead, by having more time to care for each individual animal being weighed. It will reduce the manual labour required to write each individual weight, date and correct Tatonumber. And through this method, make it easier to weigh more animals each session.

Most wishes in the project description (see section 1.1) have been achieved, and the represents from Norsvin has shown satisfaction with the final result.

8.2 Alternative options and choices

Alternative solutions may have been to spend more time on the graph. Making our graph more configurable, where you can view various presets to show statistics with the added functionality of making that data exportable.

We would also have liked to spend more time on making the global hotkey configurable. Such as making it possible to edit what key to use, and making the hotkey able to output more types of data, for example: Date and timestamp.

If we could choose, a closer bond with Norsvin, with physical meetings with both our contacts and the farmers, would have been more optimal. This had provided faster clarity in several parts of the thesis.

8.3 Future work

This report in combination with our JavaDoc [24] could serve well as documentation for any further development. Norsvin's development team uses Java and will be able to familiarize themselves with our solution. It is possible to connect it more closely with their internal systems, such as Ingris and Tono, similar to how the web application for weighing works today.

With the technology behind our solution, you can also create a mobile application that we opted out of. Using Java and Gradle. Parts of the code can be reused here.

This is a general application which can be used to weigh and store the sessions for anything, and is not strictly adhered to only support weighing of pigs and animals.

8.4 Evaluation of group work

The teamwork has worked well. Some of the project members did not know each other in advance, but this has had little impact. We initially planned to use scrum, but mainly relied on GitLab's integrated Kanban tool; the Issue Board. We have had many internal meetings, both online and physically. We have also had collective coding sessions, with live coding over Discord, as well as Code With Me in IntelliJ.

For the report, we have used Overleaf, as well as google docs to create checklists where everything takes place in real time. Here, too, we have read over each other's parts as it fills up.

8.5 Ending

We are pleased to have developed what we consider to be a full-fledged application that does nearly everything what we wanted, as well as what Norsvin wanted. Our hope is that they use our solution as a background service or our code to get the weight more efficiently in the future. As an end note, we would like to thank Norsvin for a good collaboration and the opportunities they gave us with this task. And Tom Røyse

Bibliography

[1] C. Hoffman, *What is a csv file, and how do i open it?* [Online]. Available: https://www.howtogeek.com/348960/what-is-a-csv-file-and-how-do-i-open-it/.

[2] Oracle, *Bufferreader*. [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html.

[3] Wikipedia, *Ci/cd*. [Online]. Available: https://en.wikipedia.org/wiki/CI/CD.

[4] JetBrains, *Code with me: The ultimate collaborative development service by jetbrains*. [Online]. Available: https://www.jetbrains.com/code-with-me/.

[5] P. Pedamkar, *Introduction to javafx controller*. [Online]. Available: https://www.educba.com/javafx-controller/.

[6] Oracle, *Hashmap*. [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html.

[7] Wikipedia, *Internationalization and localization*. [Online]. Available: https://en.wikipedia.org/wiki/Internationalization_and_localization.

[8] Oracle, *Javadoc*. [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html.

[9] Wikipedia, *Javafx*. [Online]. Available: https://en.wikipedia.org/wiki/JavaFX.

[10] W. Hedgecock, *What is jserialcomm?* [Online]. Available: https://fazecast.github.io/jSerialComm/.

[11] Wikipedia, *Internationalization and localization*. [Online]. Available: https://en.wikipedia.org/wiki/Internationalization_and_localization.

[12] Oracle, *Class bufferedreader*. [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html.

[13] Wikipedia, *Radiofrekvensidentifikasjon*. [Online]. Available: https://no.wikipedia.org/wiki/Radiofrekvensidentifikasjon.

[14] Wikipedia, *Rs232*. [Online]. Available: https://en.wikipedia.org/wiki/RS-232.

[15] Nightowl, *Java uses rxtx for serial port communication*. [Online]. Available: https://programmer.ink/think/java-uses-rxtx-for-serial-port-communication.html.

[16] Serialio, *What's the difference between bluetooth le and bluetooth spp (ble vs spp)?* [Online]. Available: https://www.serialio.com/faqs/whats-difference-between-bluetooth-le-and-bluetooth-spp-ble-vs-spp.

[17] Wikipedia, *Universal asynchronous receiver-transmitter*. [Online]. Available: https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter.

[18] Norsvin, *Om oss*. [Online]. Available: https://norsvin.no/om-oss/.

[19] C. Drumond, *What is scrum?* [Online]. Available: https://www.atlassian.com/agile/scrum.

[20] M. Rehkopf, *Kanban vs. scrum: Which agile are you?* [Online]. Available: https://www.atlassian.com/agile/kanban/kanban-vs-scrum.

[21] D. Radigan, *Using workflows for fun & profit*. [Online]. Available: https://www.atlassian.com/agile/project-management/workflow.

[22] Discord, *What makes discord different?* [Online]. Available: https://discord.com/why-discord-is-different.

[23] I. Gaba, *What is gradle and why do we use gradle?* [Online]. Available: https://www.simplilearn.com/tutorials/gradle-tutorial/what-is-gradle.

[24] S. Pedersen, J. Strand and N. Leivestad, *Javadoc*. [Online]. Available: https://folk.ntnu.no/stiapede/bachelor/.

[25] M. Amirault, *Using ssh keys with gitlab ci/cd*. [Online]. Available: https://docs.gitlab.com/ee/ci/ssh_keys/.

[26] T. Watson, *Gitlab ci/cd variables*. [Online]. Available: https://docs.gitlab.com/ee/ci/variables/README.html.

[27] SQLite, *Limits in sqlite*. [Online]. Available: https://www.sqlite.org/limits.html.

[28] CommFront, *Advanced serial protocol analyzer*. [Online]. Available: https://www.232analyzer.com/.

[29] *Bluecove documentation*. [Online]. Available: http://www.bluecove.org/.

[30] I. Fazecast, *What is jserialcomm?* [Online]. Available: https://fazecast.github.io/jSerialComm/.

[31] Wikipedia, *Internationalization and localization*. [Online]. Available: https://en.wikipedia.org/wiki/Internationalization_and_localization.

[32] N. Babich, *Using red and green in ui design*. [Online]. Available: https://uxplanet.org/using-red-and-green-in-ui-design-66b39e13de91.

[33] A. Pomarolli, *Javafx fxml controller example*. [Online]. Available: `https://examples.javacodegeeks.com/desktop-java/javafx/fxml/javafx-fxml-controller-example/`.

[34] Oracle, *Why use fxml*. [Online]. Available: `https://docs.oracle.com/javase/8/javafx/fxml-tutorial/why_use_fxml.htm#CHDIEGBB`.

[35] S. Harding, *What is a cpu thread? a basic definition*. [Online]. Available: `https://www.tomshardware.com/reviews/cpu-computing-thread-definition,5765.html`.

[36] SonarQube, *Sonarqube*. [Online]. Available: `https://www.sonarqube.org/`.

[37] Oracle, *The try-with-resources statement*. [Online]. Available: `https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html`.

Appendix A

Project agreement

NTNU skapelige universitet

Vår dato

Vår referanse

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

Norsvin SA
(oppdragsgiver), og

Stian Pedersen, Johan Strand og Niklas Leivestad (studenter).

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studentene skal gjennomføre prosjektet i perioden fra 11.01 2021 til 31.05.2021 .

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon, reiser og nødvendig overnatting på steder langt fra NTNU i Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle beståtte bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv NTNU Open.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studentene leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, studenter og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): Tom Røise

Oppdragsgivers kontaktperson (navn): Rune Sagevik

Student(er) (signatur): [illegible] dato 21.01.2021

Stian Pedersen dato 21.01.2021

Johan Strand dato 21.01.2021

Oppdragsgiver (signatur): [illegible] dato 29.01.2021

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): ______________________ dato ____________

Appendix B

Norsvin's assignment

Oppdragsgiver :

Norsvin SA, Storhamargata 44, 2317 Hamar

Oppgave:

Applikasjon for avlesing av digital vekt.

For å øke graden av digitalisering og sikre og forenkle prosessen rundt vektregistrering på gris i grisebesetninger er det i Norsvin tatt i bruk digitale vekter som kan kommunisere med ekstern maskinvare.

Oppgavens mål:

I denne oppgaven ønskes utviklet en desktop programvare for pc (og ev. app for Android og/eller iOS basert mobil/tablet) som kan lese av vekt og formidle denne til skrivemarkøren i den til en hver tid aktive applikasjonen som kjøres (det være seg en teksteditor, webapplikasjon eller liknende).

Oppgavens krav:

Det blir i Norsvin benyttet flere forskjellige vekter og avlesningsapplikasjonen må fritt kunne konfigureres til å støtte alle disse. Avlesningsapplikasjonen skal, etter konfigurering, kunne kjøres som en bakgrunnsapplikasjon eller service. Avlesing av vekt skal kunne aktiveres med en, for systemet, global hurtigtast (f.eks funksjonstast F7). Hvilken tast som skal fungere som hurtigtast skal, i så stor grad som det lar seg gjøre, kunne bestemmes av bruker ved konfigurasjon av applikasjonen.

Ønskede opsjoner:

I første omgang ønskes bare innlesing av vekt, men applikasjonen bør også ha opsjon for innlesing av f.eks dyrets id via strekkode eller RFID-leser som er tilkoblet systemet som kjører applikasjonen. Applikasjonen bør også ha mulighet for å legge til faste datafelter som kan sendes når vekt blir avlest. Eksempel på slike ekstra datafelter kan være: dagens dato, tidsstempel for vektavlesing.

Appendix C

Meeting logs

Andre møtet med oppdragsgiver Norsvin 15.02

Deltakere

Alle prosjektmedlemmer
Fra Norsvin:
-Rune Sagevik
-Lars Terje Bogevik

Mål

Øke klarheten i kravene.

Agenda

Møteleder oppdaterer oppdragsgiver om status og visjonen fremover.
Avklare om det skal være innlogging.
Avklare hvilket felt som skal være i databasen.
Purre på levering av vekter.

Innlogging

Nei, eventuelt få en pretestfil hvor man skal sjekke opp brukernavn/passord.

Databasen

Tatonummer, RFID før grisen er 21 dager gammel, dato, vekt, rase, kjønn, binge.

Vekter

Kommer til helga.

Veiledning med Tom Røise 26.03

Deltagere

Tom Røise
Alle prosjektmedlemmer

Mål

Komme i gang med rapportskriving.

Agenda

Kan man ta i bruk alt fra forprosjektet
Hvordan unngå gjentakinger.
Requirement specs?

Forprosjektet

Bruk det i første kapittel og utdyp.

Gjentakinger

Vise enkelt hvordan løsninger fungerer før man senere går dypere inn.

Requirement specs

Her går man dypere inn i det man har forklart overfladisk.

Levering av utkast torsdag etter påska, med møte på fredag.

Appendix D

Pre-project plan

Prosjektplan

1 Mål

1.1 Bakgrunn

Norsvin SA er et samvirkeforetak eid av 1500 svineprodusenter. Under avlsarbeidet ønsker de å øke graden av digitalisering og sikre og forenkle prosessen rundt vektregistrering på gris i grisebesetninger. Dagens rutiner på vekting er å manuelt skrive inn svinets ID-nummer og vekt. Norsvin tar i bruk digitale vekter som kan kommunisere med ekstern maskinvare og ønsker derfor å utnytte denne egenskapen. Vektene kommuniserer over Bluetooth og RS232 port. Her ønsker Norvin å få utviklet desktop programvare for pc og mobil applikasjon. Disse skal kunne lese av vekt og formidle denne til clipboard i den til enhver tid aktive applikasjonen som kjøres.

1.2 Prosjektmål

Resultatmål
Målet er å ferdigstille desktop programvare for pc og en mobil applikasjon for Android som kan brukes med ulike vekter tatt i bruk av Norsvin uten store modifikasjoner. Applikasjonene skal kunne benytte seg av Bluetooth og skannere som registrerer data automatisk. Applikasjonen skal også inneholde og bruke en intern database som er mindre og lettere å bruke for individuelle bønder.

Effektmål
Målet er å effektivisere den daglige vektregistreringen på gris i grisebesetninger, samt forenkle prosessen ved å kunne velge mellom pc og mobil.

Læringsmål
Lære å innhente informasjon fra hardware via Bluetooth, RFID/NFC og RS232.
Hvordan utvikle Android applikasjoner som benytter seg av bluetooth og RFID/NFC scanner.
Bruke SCRUM-metodikk i et reelt prosjekt.
Lære oss bruk av testing og kvalitetskontroll
– Bruke unittester på viktige komponenter.
– Bruke kvalitetssikringsverktøy som øker kvaliteten på koden.
– Gjennomføre og se nytten av brukertester.
Lære oss hvordan vi kan lage intuitive grafiske brukergrensesnitt.
Utvikle en rapport som dokumenterer fremgangen og sluttprodukt fra Norsvins oppdrag.

2 Omfang

2.1 Fagområde

Vekting av gris er en daglig aktivitet hos bøndene i Norsvin. Grisen går fra nyfødt til 90 kg de første 100 dagene i livsløpet. Tett oppfølging er viktig for å følge med på utviklingen av hver gris. Å fjerne penn og papir eller manuell utfylling av data fra vektingen vil frigi tid for både gris om bonde.

2.2 Avgrensning

- Desktop app
- Intern database
- Android app
- Bluetooth overføring

2.3 Oppgavebeskrivelse

Vi skal utvikle en desktop- og Androidapplikasjon som Norsvins bønder kan bruke under fødsels- og treukersveiing av gris i svinebestand med digitale vekter fra Digisystem. Applikasjonene skal ha følgende funksjonalitet:

Brukeren må logge inn også kan man velge fødsels- eller treukersveiing og hvilken vekt man tar i bruk.
 - Brukernavn må ligge i databasen for å få tilgang.
Finne en gitt gris ved søk på tatonummer, som er et unikt nummer hver gris i Norsvin får ved fødsel. (Skal det stå her?)
Legge til RFID på grisen under treukersveiing. Begrunnelse? Dette nummeret får grisen når den er mellom 17 og 25 dager. Kilde?
Les av digitalvekt og legg til ny entry i databasen.
Få opp graf av griser på treukersveiing som har hatt fødselsveiing.? det blir topunktsgraf:p

Det som er hovedfokuset i oppdraget fra Norsvin er avlesing av vekt og formidle videre til clipboard som de vil implementere i sine nåværende systemer.

2.4 Ressursbehov:

Hardware eksemplar av digital vekt og eventuelle kabler.
API til eksisterende database, for innsending av loggført data.
Tilgang til Norsvin sine systemer.
Direktekontakt med systemansvarlige/IT-avdeling.

3 Prosjektorganisering

3.1 Ansvarsforhold

Produkteier - Rune Sagevik
 - Vår hovedkontaktperson i Norsvin SA og vil være tilstede på alle sprint review-møter.

Scrum master (Stian Pedersen)
 - Sørge for at produkteier forstår sin rolle.
 - Være bindeledd mellom utviklere og produkteier.
 - Sørge for at utviklingen går i riktig retning og tempo ved å iverksette tiltak hvis nødvendig.
 - Innkalle til nødvendige møter og lede disse.

Utviklere (Niklas Leivestad og Johan Strand)
Følge utviklingsrutiner ref. punkt 5.

NTNU - gi tilgang til veileder gjennom hele semesteret.

3.1 Regler og rutiner i gruppen

Grupperegler
Hvert gruppemedlem skal ha som mål å jobbe minimum 25 timer per uke.
Gruppemedlemmene skal minst ha 2 møter i løpet av en uke.
Timene skal loggføres og legges frem for gruppen på prosjektmøter.
Det skrives referat etter alle prosjekt- og statusmøter. Niklas er satt som referent og Stian skal påse at det er gjort.

3.2 Deltakere

Oppdragsgiver: Norsvin
Studenter: Niklas, Stian, Johan
Veileder: Tom Røise, NTNU Gjøvik

4 Planlegging

4.1 Fremdriftsplan:

Utviklingsmetoden har vi valgt å gå for Scrum Sprint. Da hensikten med sprint planleggingen er å definere hva som kan leveres i sprinten og hvordan det arbeidet skal oppnås. Vi vil ha en samtale på slutten av hver scrum med oppdragsgiver, der vi stiller spørsmål og gjør eventuelle endringer på arbeidet. Dette vil gi oss en bedre oversikt over hva som må gjøres og gir en jevnlig kvalitet kontroll av arbeidet. Hver andre uke vil vi ha et møte med sprintplanlegging som gjøres i samarbeid med hele gruppen.

Sprint 1 :
Her vil vi sette oss inn i hvilket programmeringsspråk, valg av database og verktøy som passer til vårt prosjekt. Opplæring av android utvikling.
Vi vil også ha et møte med Norsvin for å få ordnet tilganger til deres systemer. Her vil også prosjektplanen og prosjektavtale skrives.

Sprint backlog:

- Internasjonalisering av applikasjonen, ønsket språk er Nederlandsk, Engelsk og Norsk.
- Lese om vekt standarder, android kit.
- Tilbakemeldinger fra oppdragsgiver
- Innlesning av data fra vekt, via bluetooth og usb.
- Innlogging til norsvin sine systemer og opplasting av data.
- Opprette lokal database ved bruk av sqlite.
- Unit test for database
- Hente data fra Tonor database

5 Kvalitetssikring:

Gitlab:
Bruk av branches/grener. Slik at all ny kode blir gjennomgått av andre på gruppen før den blir sammenslått med master/hovedkoden. Det blir også brukt issue board i gitlab, hvor vi planlegger for hver sprint de enkelte funksjonene som må gjøres.

SonarQube: Vi vil ta i bruk SonarQube som er en åpen kildekode-plattform utviklet av SonarSource for kontinuerlig inspeksjon av kodekvalitet. Dette verktøyet vil gi oss en bedre kodekvalitet for å unngå for kompleks kode, potensielle bugs, struktur, og duplisering av kode.

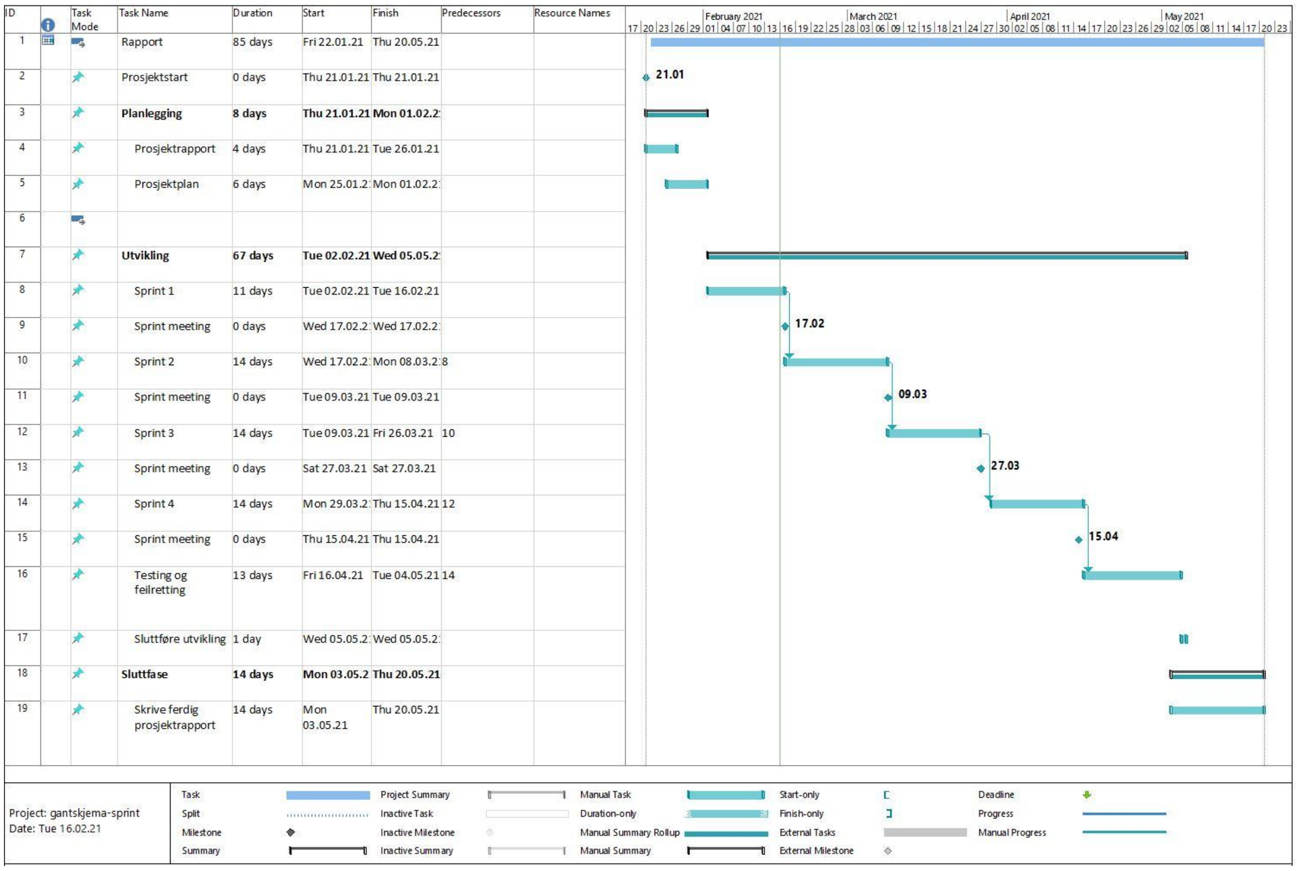
Unit Testing:
Vi vil opprette unit testing på både database, grensesnitt, avlesning av RFID og vekt. Som vil gi oss en god kvalitetssikring.

Utviklingsrutiner *- videreutvikle rutinene til å gjenspeile arbeidet fremover.*

Definer hvordan vi bruker branching.
All ny kode skal kommenteres lett forståelig og informativ før den commits.
Navngivingen til klasser, funksjoner og variabler skal være informative.

6 Plan for gjennomføring

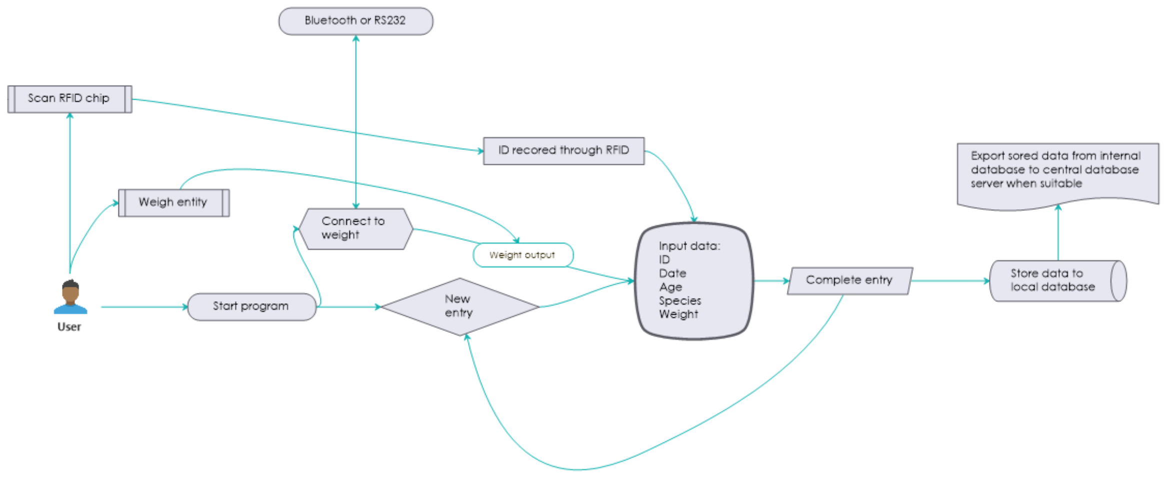
Figur 1: Tekstversjon av gannt-skjema.



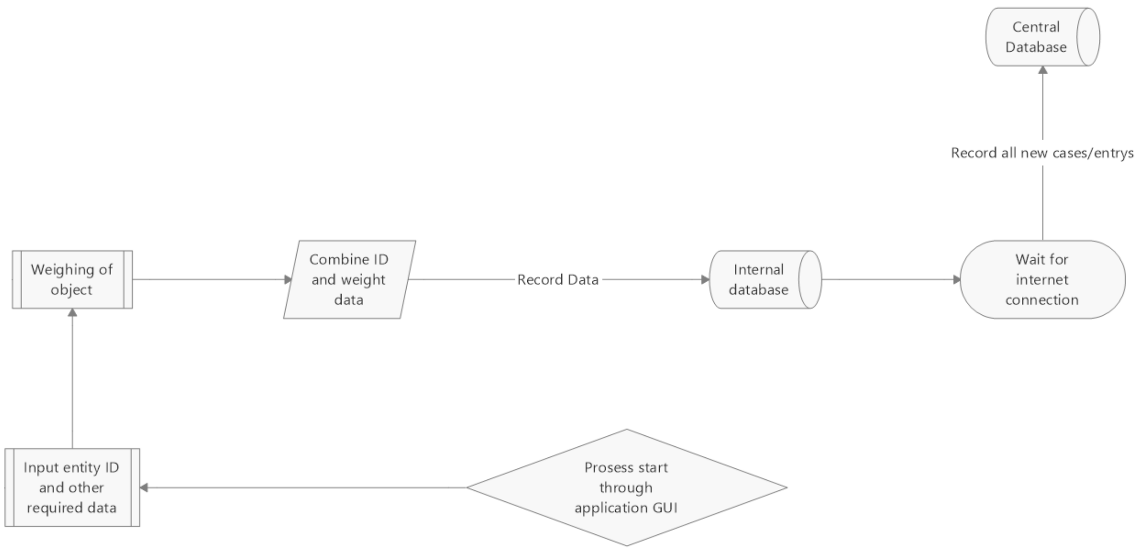
ID	ℹ	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	▦		Rapport	85 days	Fri 22.01.21	Thu 20.05.21		
2			Prosjektstart	0 days	Thu 21.01.21	Thu 21.01.21		
3			**Planlegging**	**8 days**	**Thu 21.01.21**	**Mon 01.02.2**		
4			Prosjektrapport	4 days	Thu 21.01.21	Tue 26.01.21		
5			Prosjektplan	6 days	Mon 25.01.2	Mon 01.02.2		
6								
7			**Utvikling**	**67 days**	**Tue 02.02.21**	**Wed 05.05.2**		
8			Sprint 1	11 days	Tue 02.02.21	Tue 16.02.21		
9			Sprint meeting	0 days	Wed 17.02.2	Wed 17.02.2		
10			Sprint 2	14 days	Wed 17.02.2	Mon 08.03.2	8	
11			Sprint meeting	0 days	Tue 09.03.21	Tue 09.03.21		
12			Sprint 3	14 days	Tue 09.03.21	Fri 26.03.21	10	
13			Sprint meeting	0 days	Sat 27.03.21	Sat 27.03.21		
14			Sprint 4	14 days	Mon 29.03.2	Thu 15.04.21	12	
15			Sprint meeting	0 days	Thu 15.04.21	Thu 15.04.21		
16			Testing og feilretting	13 days	Fri 16.04.21	Tue 04.05.21	14	
17			Sluttføre utvikling	1 day	Wed 05.05.2	Wed 05.05.2		
18			**Sluttfase**	**14 days**	**Mon 03.05.2**	**Thu 20.05.21**		
19			Skrive ferdig prosjektrapport	14 days	Mon 03.05.21	Thu 20.05.21		

Project: gantskjema-sprint
Date: Tue 16.02.21

Figur 2: Flytdiagram av applikasjon - Basert på initiell tolkning av oppgaven.


Bluetooth or RS232
Scan RFID chip
ID recored through RFID
Export sored data from internal database to central database server when suitable
Weigh entity
Connect to weight
Weight output
Input data: ID Date Age Species Weight
Complete entry
Store data to local database
User
Start program
New entry

Figur 3: Prosessdiagram av applikasjon


Central Database
Record all new cases/entrys
Weighing of object
Combine ID and weight data
Record Data
Internal database
Wait for internet connection
Input entity ID and other required data
Prosess start through application GUI

Oppgaven

Oppdragsgiver : Norsvin SA, Storhamargata 44, 2317 Hamar
Oppgave: Applikasjon for avlesing av digital vekt. For å øke graden av digitalisering og sikre og forenkle prosessen rundt vektregistrering på gris i grisebesetninger er det i Norsvin tatt i bruk digitale vekter som kan kommunisere med ekstern maskinvare.

Oppgavens mål: I denne oppgaven ønskes utviklet en desktop programvare for pc (og ev. app for Android og/eller iOS basert mobil/tablet) som kan lese av vekt og formidle denne til skrivemarkøren i den til enhver tid aktive applikasjonen som kjøres (det være seg en teksteditor, webapplikasjon eller liknende).

Oppgavens krav: Det blir i Norsvin benyttet flere forskjellige vekter og avlesning applikasjonen må fritt kunne konfigureres til å støtte alle disse. Avlesning Applikasjonen skal, etter konfigurering, kunne kjøres som en bakgrunns applikasjon eller service. Avlesing av vekt skal kunne aktiveres med en, for systemet, global hurtigtast (f.eks funksjonstast F7). Hvilken tast som skal fungere som hurtigtast skal, i så stor grad som det lar seg gjøre, kunne bestemmes av bruker ved konfigurasjon av applikasjonen.

Ønskede opsjoner: I første omgang ønskes bare innlesing av vekt, men applikasjonen bør også ha opsjon for innlesing av f.eks dyrets id via strekkode eller RFID-leser som er tilkoblet systemet som kjører applikasjonen. Applikasjonen bør også ha mulighet for å legge til faste datafelter som kan sendes når vekt blir avlest. Eksempel på slike ekstra datafelter kan være: dagens dato, tidsstempel for vektavlesning.

Internasjonalisering: Oppdragsgiver ønsker at applikasjonen skal ha mulighet til å støtte både norsk, engelsk og nederlandsk.