The Incredible Bulk

Software Engineering Project Methods

OCTOBER 2020

Group Members

Adam Leo
Anton Rödin
Johan Carleklev
William Söder
Albert Aveler
Enis Hasanaj
Johan Kantola

Business Plan

The Project

The project was to produce a prototype of a nuclear power plant safety system. The system should be used by nuclear technicians when they clock in and clock out from a power plant. Furthermore, power plant managers should be able to monitor the activity of nuclear technicians. The safety system consists of three parts: a safety console, a mobile device, and a database.

Approach

The safety terminal, that is the hardware, will communicate with the mobile application over bluetooth. The mobile application will then communicate with the database over the internet. Whenever a nuclear technician clocks in or out the activity will be logged in a database. This information can then be reached by the power plant managers on the admin part of the mobile application. Furthemore, the mobile application will show a timer indicating for how much longer the nuclear technician can stay clocked in. The time left is dependent on the current radiation levels. These radiation levels can be changed on the safety terminal; where numerous parameters simulate real life scenarios:

- Whether a technician has a hazmat suit on
- Radiation levels
- Which room the nuclear technician is in.

This information will also be displayed on the mobile application for the nuclear technician to see.

Limitations and requirements

- An online database is used to store information from the prototype.
- An app for an Android mobile device.
- A microcontroller to interface with.
- The connection between the devices must be wireless.
- Only support one technician at a time as bluetooth only allows for one paired unit at a time.

Implemented Features

UI/UX features

Neumorphism

To get the neumporphistic look the following dependency was used:
`com.github.fornewid:neumorphism`
Which is a project made by github user Fornewid.

Lottie

To use animation the following dependency was used:
`com.airbnb.android:lotti`
Which is a library made by airbnb

Software features

Firstore Database

To store relevant data Firestore database is used. Firstore is a cloud NoSQL document based database.

Android's built-in Bluetooth library.

We used Android's built-in Bluetooth library to be able to communicate over bluetooth. This coupled with handlers that parsed and sent datastreams made it possible to communicate with the hardware.

Hardware features

Arduino Uno

The Arduino Uno is used as the development platform for all of the hardware and embedded code used in this project.

RFID reader MFRC522

The RFID reader is implemented using the SPI(Serial Peripheral Interface) protocol and is used to read physical tags and then relay that information to the microcontroller.

Bluetooth module HC-06

The bluetooth module is connected to the microcontroller using the I2C bus for communication, and is used for serial communication with a mobile phone.

LCD

The 16x2 LCD display is controlled by a parallel interface. We run it in 4-bit mode because it requires four pins less than 8-bit mode and for our use we only need the functionality of the 4-bit mode.

Buttons and potentiometer

The buttons are simple momentary buttons connected to the arduino with an internal pullup and when pressed, pulled down to ground.
The potentiometer is 10kOhm and connected between ground and +5v gives us an input between 0 and 1023 because of the 10 bit AD-converter in the Arduino.

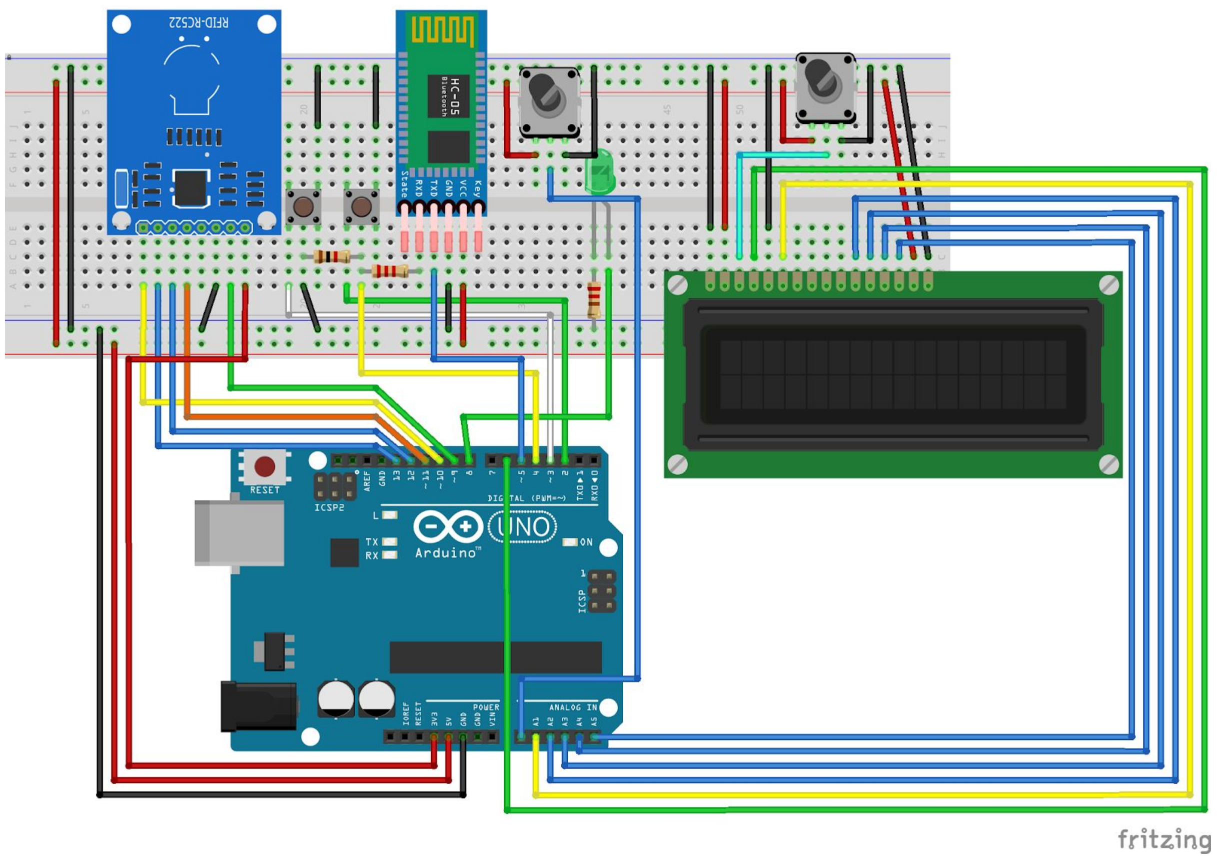


Figure 1, wiring diagram over the hardware on the console.

UX/UI Design

The UI should be intuitive and display vital information in a clear and concise way. We decided to use neumorphism which is a blend of skeuomorphism and realism together with flat design. By using this style, the information we wanted to display had clear contrast and was easy to read. In Figure 2, the UI and UX for the technician is displayed.

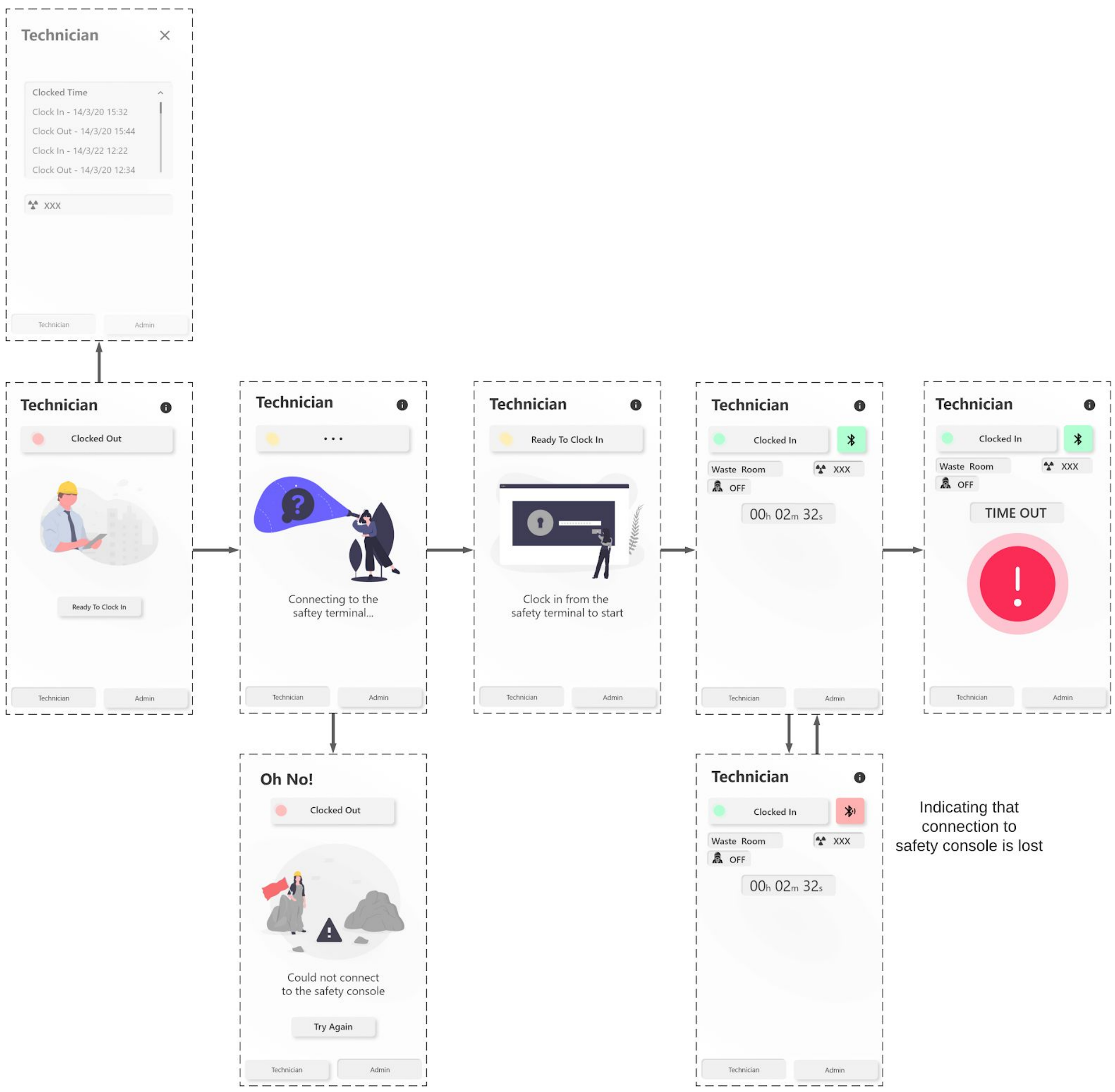
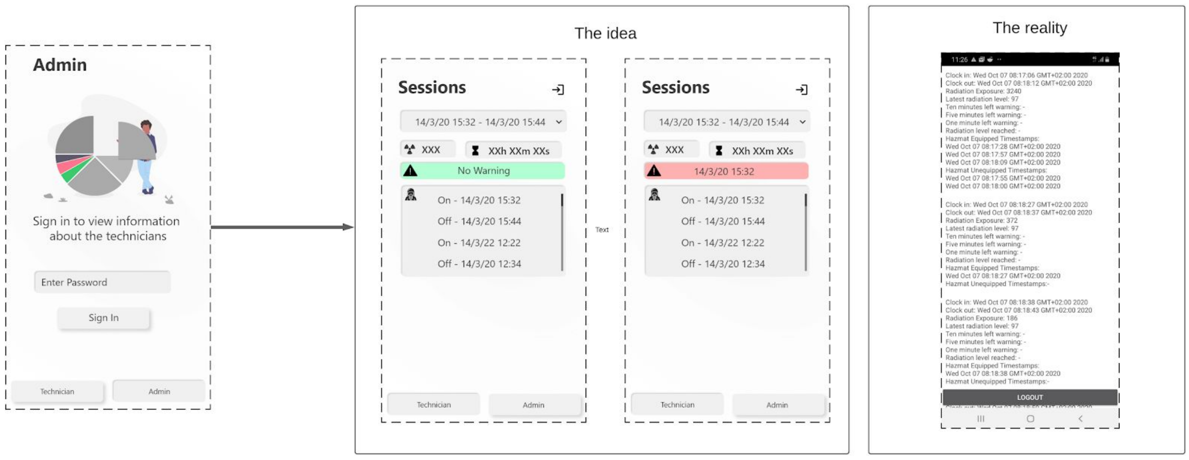


Figure 2, illustrates both the UI/UX as well as the flow of the application.

First the technician makes themselves ready to clock in before clocking in on the safety system. The application then tries to connect to the safety consol. If it fails, a message will be displayed prompting the user to try again. If it succeeds, the timer will start. If the connection between the safety system and the application is lost, an indicator showing that signal is lost will be shown. At the same time, the mobile application will try to reconnect to the safety system. When the time is out, a warning will be shown, prompting the user to leave immediately.

Regarding the admin interface, we had a UX/UI mock-up following the style of the rest of the application. Where information would be displayed in an easy-to-read and clear way. Unfortunately, due to time limitations, we had to opt to dump all the data on the screen instead. Meaning the admin side is not user friendly see Figure 3.



```
Clock in: Wed Oct 07 08:17:06 GMT+02:00 2020
Clock out: Wed Oct 07 08:18:12 GMT+02:00 2020
Radiation Exposure: 3240
Latest radiation level: 97
Ten minutes left warning: -
Five minutes left warning: -
One minute left warning: -
Radiation level reached: -
Hazmat Equipped Timestamps:
Wed Oct 07 08:17:28 GMT+02:00 2020
Wed Oct 07 08:17:57 GMT+02:00 2020
Wed Oct 07 08:18:09 GMT+02:00 2020
Hazmat Unequipped Timestamps:
Wed Oct 07 08:17:55 GMT+02:00 2020
Wed Oct 07 08:18:00 GMT+02:00 2020

Clock in: Wed Oct 07 08:18:27 GMT+02:00 2020
Clock out: Wed Oct 07 08:18:37 GMT+02:00 2020
Radiation Exposure: 372
Latest radiation level: 97
Ten minutes left warning: -
Five minutes left warning: -
One minute left warning: -
Radiation level reached: -
Hazmat Equipped Timestamps:
Wed Oct 07 08:18:27 GMT+02:00 2020
Hazmat Unequipped Timestamps:-

Clock in: Wed Oct 07 08:18:38 GMT+02:00 2020
Clock out: Wed Oct 07 08:18:43 GMT+02:00 2020
Radiation Exposure: 186
Latest radiation level: 97
Ten minutes left warning: -
Five minutes left warning: -
One minute left warning: -
Radiation level reached: -
Hazmat Equipped Timestamps:
Wed Oct 07 08:18:38 GMT+02:00 2020
Hazmat Unequipped Timestamps:-
```

Figure 3, shows how we planned for the admin information to be displayed versus how it actually is displayed.

Software Design

As previously mentioned, the mobile application was built using Kotlin and Android. We worked with both fragments and activities. We used fragments as a way to display temporary information and an activity as a way to store information that we want to continuously check regardless of the current fragment, see the Figure 4 below.

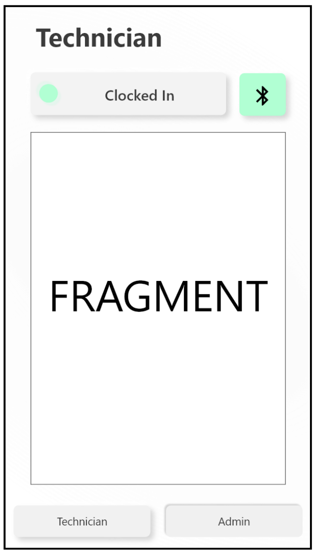


Figure 4, shows how a fragment sits on top of an activity.

In the figure above we can see how a fragment lives on an activity, this enables us to show different information, while still displaying vital information that should be known when the user is on that activity. The activity also worked as a gateway between backend information such as bluetooth and database communication and the UI that displayed said information. In figure 5, we can see how the fragments are created depending on the information we get from the database.

```
fun setStartUpValuesFromDB() {
        bt_status.visibility = View.GONE
        FirestoreDatabase().getClockedInStatus(Globals.technicianId) {
                clockedInStatus, error, errorMessage ->
            if (!error) {
                Globals.isClockedIn = clockedInStatus
                if (clockedInStatus != null && clockedInStatus == true) {
                    setHazmatStatusFromDBValue()
                    setRoomLocationFromDBValue()
                    setRadiationLevelFromDBValue()
                    replaceToClockedInFragment()
                    setClockedInStatusTrue()
                    clockedInFragment.startTimer(Globals.timeLeftMs/1000)
                    bt_status.visibility = View.VISIBLE
                    assertBtOn()
                }
                else {
                    replaceConnectingFragment()
                    setClockedInStatusFalse()
                }
            }
            else {
                Toast.makeText(this, errorMessage, Toast.LENGTH_LONG).show()
                replaceConnectingFragment()
                setClockedInStatusFalse()
            }
        }
}
```

Figure 5, shows how, depending on the information from the database, different actions are taken.

The code above is in the main activity class, that is the class which should persist even when fragments change. The code shows that if an user is clocked in, then we want to use all the stored values within the database to add to the current mobile application values. This enables us to close and then reopen the application while still keeping the current clocked in session. So when a user is clocked in on the database, and then opens the mobile application we want the user to get to the clocking fragment with all relevant information. On the other hand, if the user is not clocked in, we want the user to get to the connecting fragment. From there the user can connect to the safety system if they so please.

On the mobile application, Bluetooth was implemented using the standard built-in Bluetooth library found in Android. For connecting to the safety console, the MAC-address of the safety console's Bluetooth module (HC-06) was hardcoded inside the app itself. The app would then scan the area and find a device with that exact MAC-address. If it is found, an RFCOMM channel is established in order to communicate. The communication itself is done using simple input-/output-streams to which you can read/write simple byte-array data. Note that the bytes are *unsigned*. In order to identify what type of message is being sent, and what data the message consists of, the following protocol was implemented, used by both the mobile app and the safety console:

Safety Console -> App

TYPE	DATA	TYPE-CODE
Potentiometer value	1 - 100	3
Hazmat suit change	1	4
RFID KEY	4 byte	5
Room change	1	9

App -> Safety Console

TYPE	DATA	TYPE-CODE
Clock In/out	0/1	1
Warning	0/1	2
Potentiometer value request	1	3
Hazmat suit status	1/0	4
Time left	1byte / 2byte / 3byte	6
Ping	1	7
Database failure	1	8
Room status	0=break room 1=control room 2=reactor room	9

Byte ->	[Header]	[Length]	[Data]]]]....
	typecode	size of data	data

A sample message sending the current room (reactor room) to the safety console could look like this:
[9, 1, 2]
The above example displays an array with total length 3 which contains the bytes 9, 1, and 2. The first byte represents the type of message, which in this case is 9, meaning it is a “Room status” message. The second byte represents the length of the data. In this case, the length is only 1 since we are only sending a simple byte containing 0/1/2. The third byte is the actual data. In this case, we are sending a 2, meaning the safety console should switch to and display the “Reactor room” as the current room. Note that if the “data” had been more than 1 byte, the byte representing the length would be larger than 1. For example, when the safety console sends an RFID key, the byte-array could look like this:
[5, 4, 46, 191, 30, 51]
Here, the data-length byte is set to 4 since the actual data consists of the 4 bytes 46, 191, 30 and 51.
For the Bluetooth feature, an interface was made that utilizes delegates to raise an event whenever the app connects/disconnects, and whenever a specific message type is received. Users of this interface can then subscribe to these events and have a function called when these events occur. On the mobile application, in the code, this is done as shown in Figure 6.

```kotlin
//Receiving data
fun listen() {
    val BUFFER_SIZE = 1024
    val buffer = ByteArray(BUFFER_SIZE)

    try {
        var bytesReceived = inputStream!!.read(buffer)
        var i = 0
        while(i < bytesReceived){
            val type = buffer[i].toInt()
            if(type == 0){
                break
            }
            else{
                i++
                var dataLength = buffer[i].toInt()
                if(dataLength == 0){
                    val newBuffer = ByteArray(BUFFER_SIZE)
                    val newBytesReceived = inputStream!!.read(newBuffer)
                    dataLength = newBuffer[0].toInt()
                    for(x in 0..(newBytesReceived - 1)){
                        buffer[i + x] = newBuffer[x]
                    }
                }
                val data = ByteArray(dataLength)
                i++
                var o = 0
                while(o < dataLength){
                    data[o] = buffer[i]
                    i++
                    o++
                }
                onMessageReceived(type, data.toUByteArray())
            }
        }
    } catch (e: IOException) {
        doOnMainThread {
            connected = false
        }
        e.printStackTrace()
        shouldListen = false
    }
}
//Sending data
fun send(bytes: UByteArray) {
    if(::socket.isInitialized && outputStream != null && connected){
        outputStream!!.write(bytes.toByteArray())
    }
}
```

Figure 6, the code on the mobile application to listen for incoming messages from the hardware.

What the mobile application does when it receives the message depends on the information within the message. For example, a function that updates the current time estimation is called when the app receives a new potentiometer value (type 3), see Figure 7.

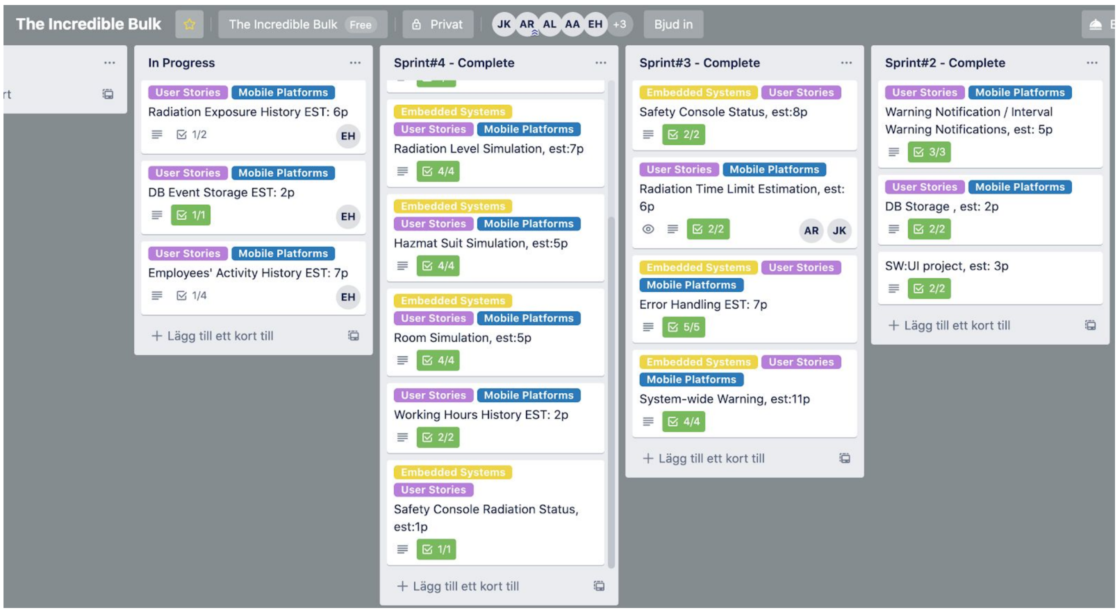
```
BluetoothConnectionHandler.onMessage(3) {
    Globals.potentiometerValue = it[0].toInt()
    updateTimer()
    clockedInFragment.updatePotentiometerUI(Globals.potentiometerValue.toString())
    FirestoreDatabase().setLatestRadiationLevel(Globals.potentiometerValue){
            errorOccurred, errorMessage ->
        if(errorOccurred){
            Toast.makeText(this, errorMessage, Toast.LENGTH_LONG).show()
        }
    }
}
```

Figure 7, shows how the application updates the UI and Timer if it gets a message of type 3.

Scrum with XP

Storyboard

Trello is a digital planning and organization tool used for people and working teams to visualize their work. We used Trello as our main tool for keeping a “scrum board” where the product backlog, current sprint completion, user stories and scrum masters’ notes are clear and visible. This is illustrated in Figure 8 below.



In Progress
- User Stories, Mobile Platforms — Radiation Exposure History EST: 6p — 1/2 — EH
- User Stories, Mobile Platforms — DB Event Storage EST: 2p — 1/1 — EH
- User Stories, Mobile Platforms — Employees' Activity History EST: 7p — 1/4 — EH
- + Lägg till ett kort till

Sprint#4 - Complete
- Embedded Systems, User Stories, Mobile Platforms — Radiation Level Simulation, est:7p — 4/4
- Embedded Systems, User Stories, Mobile Platforms — Hazmat Suit Simulation, est:5p — 4/4
- Embedded Systems, User Stories, Mobile Platforms — Room Simulation, est:5p — 4/4
- User Stories, Mobile Platforms — Working Hours History EST: 2p — 2/2
- Embedded Systems, User Stories — Safety Console Radiation Status, est:1p — 1/1
- + Lägg till ett kort till

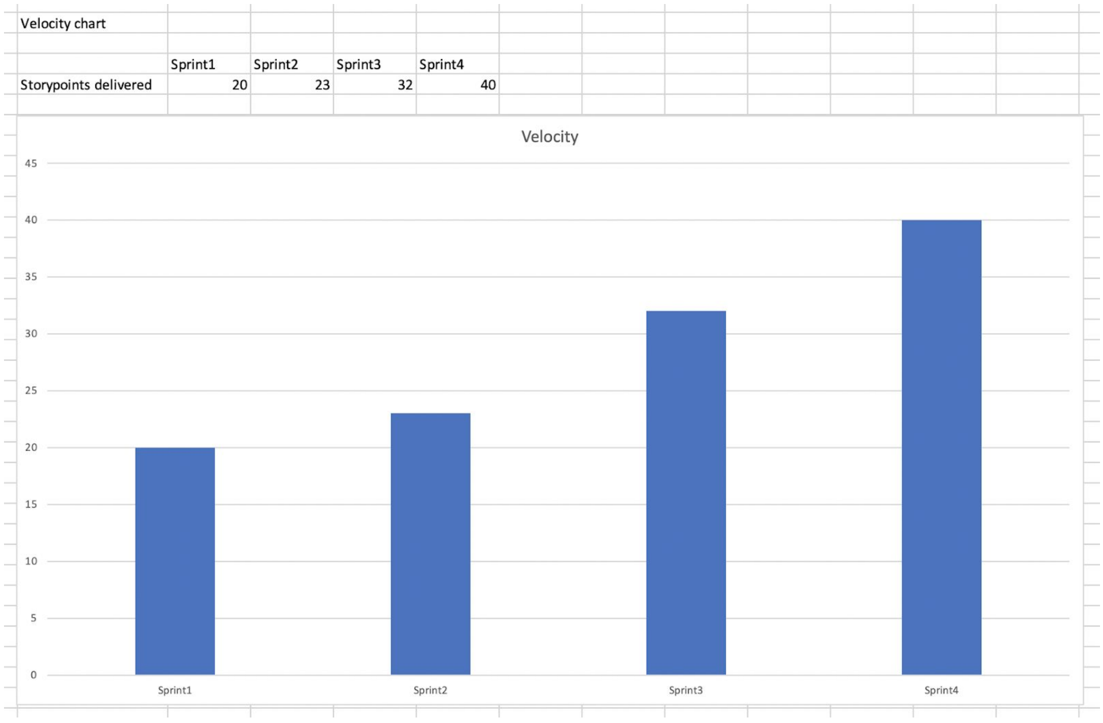
Sprint#3 - Complete
- Embedded Systems, User Stories — Safety Console Status, est:8p — 2/2
- User Stories, Mobile Platforms — Radiation Time Limit Estimation, est: 6p — 2/2 — AR JK
- Embedded Systems, User Stories, Mobile Platforms — Error Handling EST: 7p — 5/5
- Embedded Systems, User Stories, Mobile Platforms — System-wide Warning, est:11p — 4/4
- + Lägg till ett kort till

Sprint#2 - Complete
- User Stories, Mobile Platforms — Warning Notification / Interval Warning Notifications, est: 5p — 3/3
- User Stories, Mobile Platforms — DB Storage , est: 2p — 2/2
- SW:UI project, est: 3p — 2/2
- + Lägg till ett kort till

Figure 8, illustrates how we worked with Trello.

Velocity

When working with scrum, knowing the team's output per sprint is an advantage as it enables the project team to plan more accurate sprints. Therefore a sprint velocity chart was used to keep track of the output of the team, see Figure 9.



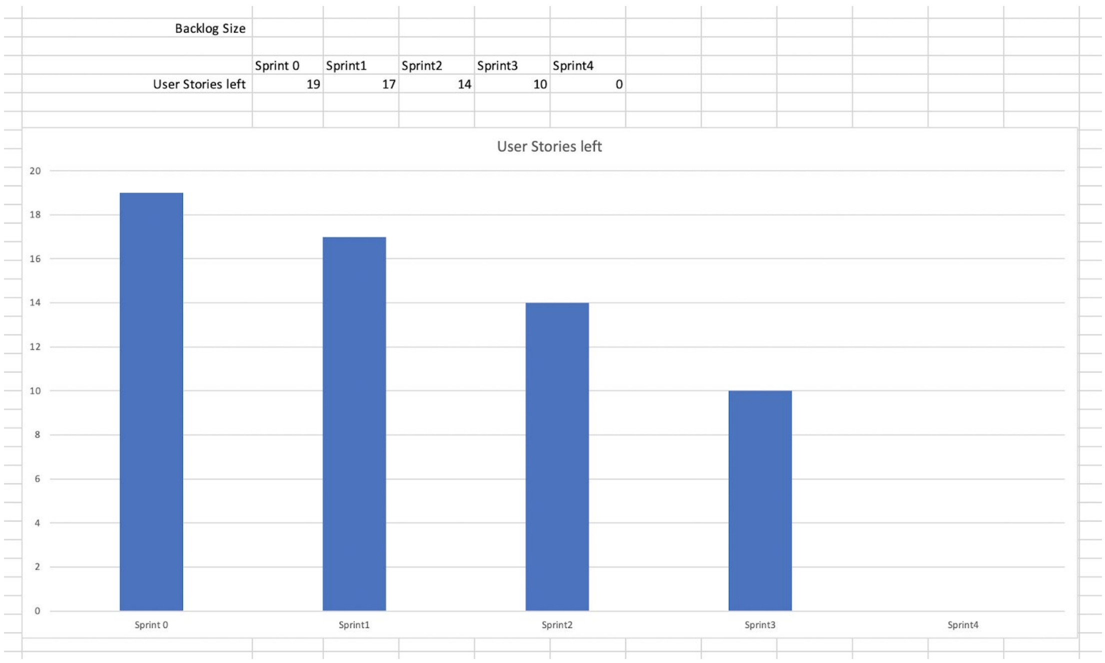
Velocity chart				
	Sprint1	Sprint2	Sprint3	Sprint4
Storypoints delivered	20	23	32	40

Figure 9, a velocity chart showing the number of story points delivered each sprint.

At a first glance it seems that we did twice the work on sprint four in respect to the first. This is not really true. One reason is that we did a lot of refactoring and improvements during sprint two. This lay the foundation for our code base and made the tasks further down the backlog easier to implement because of this solid ground. We probably didn't realise this and therefore over-estimated the tasks in sprint three and four. Another reason for this uneven graph is that we are a newly created group with people who haven't worked in this constellation before and we are new to the idea of estimation. If we were to continue the project in this group the velocity would have settled after probably going up and down for some more sprints.

Burn down chart

Having a good overview of the project itself is important. That is, approximately when will we have finished which stories. Is the product backlog in a good state, and so on. The way we did this was by using a burndown chart, see Figure 10.



Backlog Size

	Sprint 0	Sprint1	Sprint2	Sprint3	Sprint4
User Stories left	19	17	14	10	0

Figure 10, shows the projects burn down chart.

By using the velocity chart together with the burndown chart we got a fairly accurate representation of the current state of the project. Making it possible to give project information to the business owner in a more concrete way. As well as to use the information in scrum planning meetings to plan the next sprint.

Retrospective

After a sprint was completed, we had a retrospective meeting. In the meeting we discussed how the latest sprint went. The scrum master wrote on the whiteboard what the team thought went well and what can be done better. We also discussed the things we wanted to improve from the last retrospective and how well we followed the previous action plan. Each team member also drew a timeline over the sprint to illustrate how they thought the sprint went. By doing this we could point out ups and downs in the sprint, find common patterns and identify things to improve in the following action plan. Figure 11 shows the whiteboard from the second retrospective.

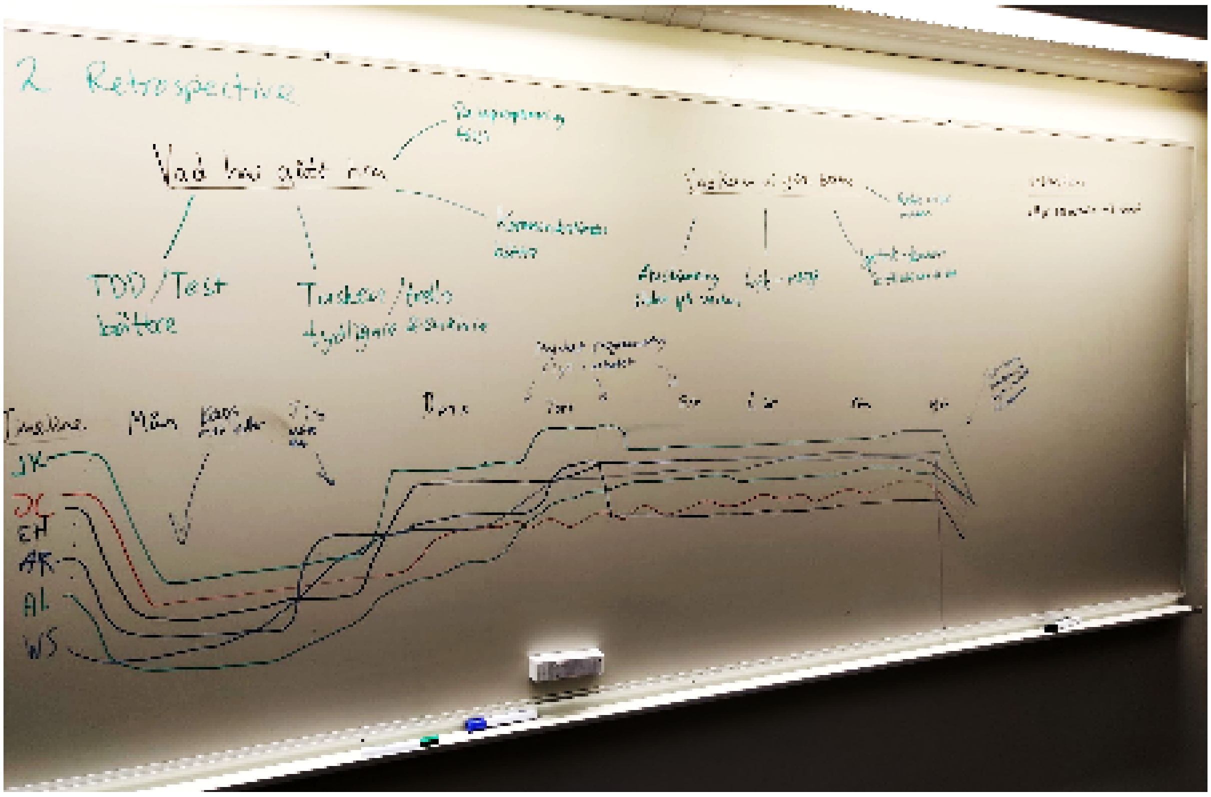
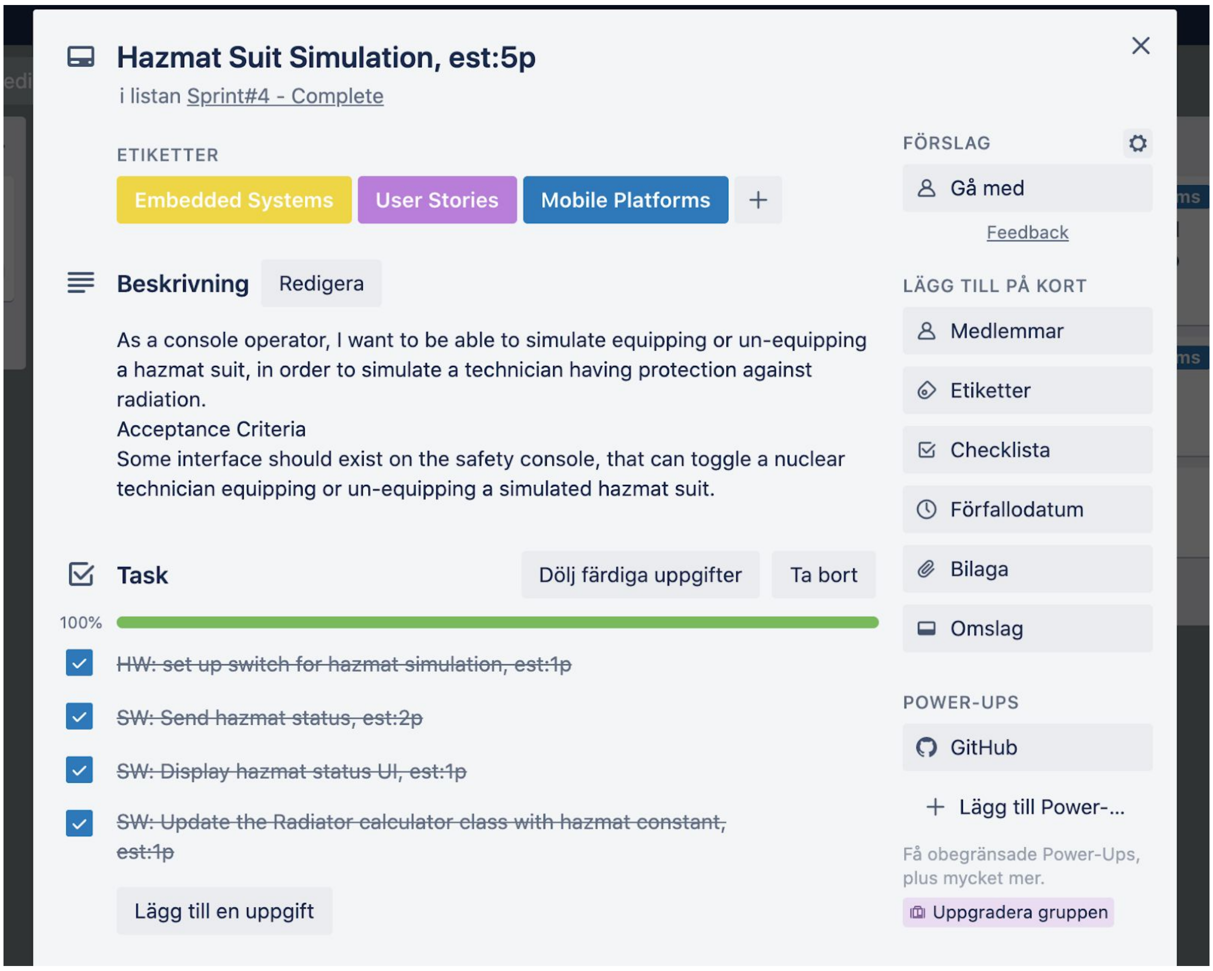


Figure 11, shows how we used timelines and reflections during the retrospective.

Estimation

During the backlog grooming sessions members in groups of two or three took a high priority unestimated story from the product backlog on Trello. Preferably the story would be related to their roles and/or skills as that would yield more accurate estimation. The reason for doing this in a group is that it enabled discussion about the estimation which further helped create an accurate estimation. The group then broke their chosen stories down into tasks written as checkboxes under the story itself. Finally, they wrote the estimation for both each individual task as well as the stories themselves. Where the estimation for the story is the sum of the estimation of all tasks for that particular story, see Figure 12.


Hazmat Suit Simulation, est:5p

i listan Sprint#4 - Complete

ETIKETTER

Embedded Systems | User Stories | Mobile Platforms | +

Beskrivning | Redigera

As a console operator, I want to be able to simulate equipping or un-equipping a hazmat suit, in order to simulate a technician having protection against radiation.
Acceptance Criteria
Some interface should exist on the safety console, that can toggle a nuclear technician equipping or un-equipping a simulated hazmat suit.

Task | Dölj färdiga uppgifter | Ta bort

100%

- ✓ ~~HW: set up switch for hazmat simulation, est:1p~~
- ✓ ~~SW: Send hazmat status, est:2p~~
- ✓ ~~SW: Display hazmat status UI, est:1p~~
- ✓ ~~SW: Update the Radiator calculator class with hazmat constant, est:1p~~

Lägg till en uppgift

FÖRSLAG

Gå med

Feedback

LÄGG TILL PÅ KORT

Medlemmar

Etiketter

Checklista

Förfallodatum

Bilaga

Omslag

POWER-UPS

GitHub

+ Lägg till Power-...

Få obegränsade Power-Ups, plus mycket mer.

Uppgradera gruppen

Figure 12, how we used Trello to estimate stories.

Meetings

Weekly meeting schedule

Monday	Tuesday	Wednesday	Thursday	Friday
	10.00-10.15 Daily standup	10.00-10.15 Daily standup	10.00-10.15 Daily standup	10.00-10.15 Daily standup
12.00-12.30 Sprint demo			10.30-11.30 Backlog grooming	
12.45-13.15 Retrospective				
13.15-14.00 Sprint planning				

This is what a sprint looked like for us. It started on Monday with sprint planning at 13:15. Daily standups were held on the other workdays at 10:00. We did backlog grooming on Tuesdays after the daily. The sprint ended with a sprint demo for the Business owner at 12.00 on Mondays and soon after that a retrospective for the team.

Git

We used Git as our main source control method. We used a private repository on GitHub in order to sync our work between all members of the group. For each new sprint, we created a new branch in the repository representing all the work that was done during that sprint. At the end of each sprint, the current sprint-branch was merged into the master branch of the repository. The master branch acted as the “release” or “working code” branch, while the sprint branches acted as “development” branches. For the embedded side we worked with a new branch when we wanted to implement a new feature, and since we were almost exclusively using pair programming this worked almost the same as on the App side. It really helped us when we wanted to implement a new feature, to be able to create a new branch free from the other implementations. This helped us experiment with just the new feature, and when the feature was implemented we could merge this into our working master branch.

Pair Programming

Pair programming was very well utilized throughout this whole project, and was very liked by the team members. It minimized the introduction of bugs, sped up solving complex problems, and helped group members acclimatize to working with unit testing. On the embedded side, we worked almost exclusively with this pair programming method, since there were only two of us embedded students. On the software side, we either worked in pairs or in threes. Occasionally, some bug fixing or refactoring were done alone.

Refactoring

Refactoring was done all the time in this project, but during sprint 2 we had a better idea on how to structure the code. So much of the work done in sprint 2 was refactoring, and in the long run that really helped us, since it was very easy to implement the coming features. With the help of tests this ensured that our refactoring did not change the functionality of our code base.

TDD / Tests

Our testing in this group is very limited, since most of the group's members never had worked with unit testing at all. This was a significant step to overcome, but every member in the group did their best to learn how to do unit testing and then try to work with a TDD mindset. After having started to work with unit testing and then going into TDD every member in the group started to see the benefits of tests. Some more advanced techniques were used in testing the app, such as a mock database for testing the database interface. Something which is lacking in the unit test department is doing unit tests on the embedded code. Since the scope was limited and a lack of experience in this area there wasn't any time to do unit tests on the embedded code base.

Reflections

Adam Leo

Many reflections will probably mention the infamous product owner misunderstanding as an unfortunate event. It also affected our team negatively. However, one silver lining was that the misunderstanding gave me the opportunity to try out numerous roles. My first roles were both scrum master and team member, as I alternated weekly between the two. When the information came out that a misunderstanding had occurred, I took on the role as a product owner. So for the first week I was a scrum master, the second week a team member, and for the last couples of weeks I was a product owner. The first week as a scrum master was very educational, especially the meeting I had with Andreas regarding the scrum masters role. One thing in particular that was very valuable was his opinion that backlog grooming meetings eases future work. An opinion that I took to heart, extra so when I transitioned to the product owner. I know for a fact that having a product backlog groomed and ready really made a difference in our output velocity for each sprint, as we could see a clear improvement. One thing that helped my work as a product owner was the second week I had contributed as a team member; the reason for this is because I got a deeper understanding of the system, which made it easier to estimate and keep track of the backlog. Another thing that was positive in being a team member was that I got to experience working with XP, something I found valuable. As pair programming, and especially testing, will probably be a large part of my software developer career going forward. All in all, agile, and especially scrum, were a pleasure to work with and I look forward to continuing to do so in the future.

Johan Kantola

Working with scrum and XP was both fun and challenging. I learned many things during the project. As I was the scrum master, I did a lot of research to be able to do the best job I could. Every Wednesday during the project, Adam and I had a meeting with Andreas Rowell. During these meetings, we went through the latest sprint from a scrum master and product owner perspective. Having Andreas as a coach was very educational and he gave us many great tips and ideas to improve the process. I think scrum is a very good and reliable method for software development and I have really understood the purpose and benefit of it. Every meeting is valuable and helps to take the project forward. One thing I understood as a scrum master was that working in a team is so much more than just writing code. Collaboration, communication and having a common goal are very important aspects. Working according to extreme programming was very different from what I am used to. Since I have no previous experience of either pair programming or unit testing, there was a lot to learn in the beginning. I really enjoyed pair programming. It was both fun and educational at the same time. Discussing solutions with the person you are coding with helps a lot in writing simpler and more well-written code. I think TDD was the hardest part of the extreme programming method to follow. We tried to work according to TDD but it was too much to learn and we did not really have that time to spare. We did a lot of unit testing but not in the TDD way. In the end I think we carried out this project in a good way. We did our best to follow the methods we were assigned to and all of us contributed to the end result.

Anton Rödin

I had already worked with scrum and extreme programming, but i hadn't given it much thought before how or why i was doing it. In this course I learned alot about why you do the things that scrum teaches. For example before I researched and thought about the daily scrum meeting I thought it was kind of tedious. After I did some thinking it was clear that it gave me a sense of direction for the project and where it was headed even though I might not have worked on the project that day.

As I said before I had already worked in a similar environment, and there I had learned alot about TDD(Test Driven Development) and unit testing in general. My team members had not, so as the team's Test Engineer i had some work to do. It was very giving to be in the teacher role, and my team members were very eager to learn using unit tests and then going into TDD they could really see the benefits working this way. Implementing the tests was kind of slow in the beginning, but once it got rolling the team could even use some more advanced techniques, such as mocking classes. It felt very good to see that we could get that far since every team member was a beginner at this.

In extreme programming there is an emphasis on pair programming, and I really like this "feature" since it produces very good results. It helps me figure out complex problems, since you can go over them and talk with another person, and paired with TDD this creates a very robust and (mostly) bug free code base.

All in all I think that we as a team worked excellently together. We really adopted the scrum methodology, and all of the team members were competent at implementing the planned work. Our good spirits and helpful mood made our group a delight to work in, and as such everyone was ready to give their all in making this project go as smoothly as possible.

Johan Carleklev

My first encounter with scrum was at my internship last spring. They told me that I should participate in the "daily" each morning. They had a storyboard and told each other what they did yesterday, what they were to do today, and if something stopped them from doing it. Going through this course and this project have teached me that there is so much more to scrum than just the daily, and I like it!

There is a big difference between working alone or in a pair on laborations in previous courses and working on a larger project with a larger group of people. Doing that the XP-Scrum way made the transition not that big.
Regarding the fact that we in The Incredible Bulk had small to no previous experience with scrum+XP I am really satisfied with how fast we adapted to the meetings, workflow and the way of working with scrum and XP.
I had the role of Hardware Engineer. When we finally got our heads around that the importance of this course is how we work with the methods and follow the backlog, not how flashy we can make the product, my focus lay on making sure we implemented the hardware as per the stories each sprint.

Enis Hasanaj

This was the first time I worked on a project with scrum and XP and I can really see the value in working with it. The sprint planning-meetings gave a really good picture of what needed to be done and the daily scrum-meetings made it clear who was working with what, if any problem had occurred and if we were getting things done in time. I also see great value in pair-programming since sometimes one can be stuck on a simple problem for too long and having someone next to you come up with suggestions to solutions can be a great time saver. It also makes the code less filled with bugs since you have another pair of eyes to look for problems in the code that is written. This was also the first time I worked with TDD and unit testing and I can see the importance in both unit testing and TDD. Doing TDD you need to think and extra time on how the code should be structured, giving a good and clean code. I was a team member and had the role of programmer(refactoring) and tried to do as much refactoring as possible but the rest of the team was good at refactoring their own code. We really saw the benefit of continuous refactoring of the code in the last sprints because we were able to implement a lot of features very fast and easy because we had a foundation of clean good-structured code.

Albert Aveler

Working with scrum was an eye opener. I always thought that we should spend a lot of time planning and drawing charts and diagrams so that every step would be calculated in advance. Apparently this does not work with software projects which are subject to change and that is why we should be more agile. In our case, many of the team members have not worked together before, but the daily scrum meetings forced us to talk and discuss things. These meetings were very important and played a huge role in building a more coherent team.
We used to book a room and do some hardware and software testing before starting with the retrospective meeting. Our retrospective meetings were very important to know if we were on the right track or not, and after these meetings we would have a better idea on where the team was lacking, what we should become better in doing and what we were already doing right.

Every member had to draw a chart that reflects how he felt during the week and one common thing was that everyone liked to do pair programming.
Pair programming has many advantages, first of all you avoid spending hours working on a trivial problem that the other programmer could figure out directly and tell you about it. It is like two different perspectives are always better and will always save a lot of time and produce much better optimal code.
I did not know anything about TDD before and I used to jump directly into writing the working code. But after having a testing session with our Test Engineer and trying testing myself I do understand how important testing is and why it is considered essential to produce good code by many experts. By writing tests you know exactly what you want from your code which will lead to better refactored code with minimum bugs. My role was a programmer(bug tracking) and I tried to write tests and try the working code in many different cases to find and fix the bugs.
In general, scrum is probably the best way to lead a team into creating a successful software project and XP with all its benefits will always be a part of my daily routine as a software developer.

William Söder

This was my first time working with scrum, and this project has really shown me how useful and powerful scrum can be as a project method. My role was Build Engineer, and through this role I was able to help the team organize and structure each component of the product. I, of course, helped with other aspects of the project such as refactoring, testing etc as well. I overall really enjoyed the weekly schedule that we set up in the group, and I feel like we were never over-occupied with meetings, and could rather put a lot of the focus on developing the actual product. One of my early concerns was that scrum would force the group to have “unnecessary” meetings, which is clearly not the case. In fact, every meeting, sprint planning, retrospective etc has been valuable in some way or another, and has really helped the group improve the way we work. I also really enjoyed XP. My favorite part of XP is pair programming, which, during the project, has clearly shown to be a very powerful way of working. When programming with someone else, it is much easier to spot bugs, refactor the code and to simply solve problems. This is because pair programming gives the programmer a new perspective on how to work and how to solve problems, and this has been incredibly useful for me during the project. I mainly pair-programmed together with Albert in the group. Other than that, I feel like XP goes hand in hand with scrum, and we were quickly able to adapt to XP’s way of thinking “simple”, and focusing specifically on what needs to be done during the sprint, and putting aside the user stories that are not currently a priority. I wish I had more time to indulge in the testing aspect of XP. Unfortunately, as I had zero previous knowledge of TDD and with the limited time of the project, I was not able to utilize TDD as much as I would have wanted. However, Anton in our group shared some of his knowledge in TDD, so I was able to test the notifications feature of the app together with Albert and Enis. Overall, the project went very well, and I believe we were able to fully realize the potential of scrum and XP.

Appendix

Sprint 0

William
We discussed which technologies we were going to use, which roles in the team each member was going to have, which features each member would focus on the most etc. We decided to make an Android application that would utilize Bluetooth, and Google Firebase as the database. Me and Albert decided to focus mainly on developing Bluetooth inside the app itself.
Adam
Planning and discussion.
Writing group contract, DoD, conflict handling, meeting schedule etc,
What to do? An android application using kotlin.
How to do it? Bluetooth, firestore, hardware stuff
Who should do it? We distributed roles.

Albert
During this sprint we decided that we were going to create an Android application that uses Google Firebase as its database. This application will communicate with Arduino on the hardware side. When it came to selecting roles, we had a wrong idea that the "Product owner" role was already given to someone from outside the team and that is why it was not mentioned in our group-contract. After deciding the roles we agreed on using Github for version control and Trello to keep track of our backlog and sprints.
Anton

Enis
On this sprint we agreed that we were gonna create an Android application using Android studio and Kotlin and that we were gonna use Firebase as our database provider. We chose the roles we were gonna have in the team, wrote a group-contract specifying some rules, roles and schedule for our different types of meetings. We also agreed on some tools to manage the project like GitHub and Trello and that communication should be done mainly through Discord.

Johan C
In sprint 0 we wrote a group contract and decided on which roles each member should take. We defined the definition of done, wrote down a plan for when to have our scrum meetings and what technologie we would use. On the mobile side we quickly agreed on an android app written in Kotlin and Firebase Firestore as a database. Because of a misunderstanding on the

hardware side where we did not get any bluetooth module and never even worked with bluetooth before our (at that time) product owner took for granted that we had a bluetooth module and that we worked with Bluetooth before. Therefore we started thinking about using wifi because that would be simpler than doing everything through the phone. When that mishap was sorted out we went for the Bluetooth way because that was the intended way to go in this project.

Johan K
During sprint zero we decided to develop an Android application that uses Firebase as a database. We wrote a team contract where we determined all the team roles. In the contract we also agreed on the definition of done, communication channel, conflict handling, meeting guidelines and which project method we wanted to use. We also agreed to use github and Trello.

Sprint 1

Adam
For the first sprint I was scrum master, meaning I held in the daily meetings and tried to make the working condition as painless as possible. This included the booking of group rooms, emailing and messaging relevant people for questions. Furthemore, me and Johan K booked a meeting with Andreas to get some more insight into how to be scrum master. I also had the responsibility of UI/UX design, so I spent some time in Adobe XD to get the design and feel of the product right. I then consulted with the team members for feedback. Johan K and I also pair programmed, coding the UI and UX.

Albert
I paired with william and started working together on the Bluetooth functionality. We started by reading some documentation to get an idea on how to implement bluetooth on android. We needed our application to be able to search for available bluetooth devices and connect with the Arduino which has a unique Mac Address. On friday, we tested the bluetooth functionality with the hardware team and we were able to connect to Arduino and send/receive data successfully.

Anton & Johan C
The first sprint we focused on setting up the hardware for the bluetooth module, RFID reader and a simple LED to show a successful clock in/clock out. When we set up the bluetooth module we tested it against a simple bluetooth terminal app on our mobile phone. For the RFID reader we simply printed out the data in a terminal in our development environment. The last

thing we did was to test this with our own app, and if a clock in/clock out we simply lit up a LED indicating whether you are clocked in or out.

Enis

During the first sprint I created a simple database where we could store the clocked-in status of a technician and login credentials of the power plant manager. I implemented functionality on the mobile application for clocking in/out a technician. After doing some research about unit testing I refactored the code and created a test double for the database and wrote some unit tests. I started writing code for admin(power plant manager)-login functionality but realized I was going against XP principles(you should not write code for future features in advance) so I removed that code and learned a lesson.

Johan K

During the first sprint, I focused a lot on researching the scrum master role and how I would relate to it. From the beginning, the thought was that Adam and I would share the scrum master role between the sprints. We worked together to make meeting agendas and what kind of documentation we needed to do throughout the project. On Wednesday we had our first meeting with Andreas Rowell. We talked a lot about the scrum master role and scrum in general. Since I also have the test engineer role in this project, I have done a lot of research on unit tests in Android. In the application, Adam and I had our first pair programming session where we implemented some basic elements in the user interface.

William

Me and Albert did quite a bit of pair programming during the first week. We did this because we had decided during the first Sprint planning on monday that we would focus on getting Bluetooth to work for the mobile app. None of us had worked with Bluetooth before so this was a bit of a challenge for both of us. During the week we read a lot of the documentation on Android's built-in Bluetooth library to get a good understanding on how it works. We thankfully got it working quite early, and on the meet-up on friday with the rest of the team, we tested the Bluetooth connection with the hardware from the DIS-guys, and saw that we could send and receive data successfully.

Sprint 2

Adam

I started this sprint as a team member and ended it as a product owner. Even this week I had a meeting with Andreas, though this time I asked questions regarding the product owner role to get more insight into best supporting my team as the new product owner. This week there was much confusion, and I did my best to ease the confusion by taking care of a lot of administered work, such as asking questions where needed, booking group rooms, looking over the backlog, etc.

Albert
Me and William had a very helpful session on TDD with Anton. We then paired with Enis on writing tests for “notifications” and it worked with no problems. We made the notifications so they would show up on different intervals. In our first implementation we created a simple timer function in order to test if a notification would show up in a specific time. We also worked on clock in/out functionality in the bluetooth section and we tested clocking with RFID tag on the safety console. It worked as it should.

Anton
This week we did not have any embedded systems user stories, so I was pair programming with Johan Kantola to teach him how to work with a TDD mindset. I also sat down with Albert and William to teach them how unit testing works, and also explained how to go forward with a TDD approach.

Enis
During the second sprint I did some pair programming with Albert And William where we implemented Notifications on the mobile application for when a technician should receive warnings about radiation limits and a simple timer that triggers the notifications at certain intervals. We did this using TDD. We also implemented functionality for storing and retrieving previous clock in/outs of the technician and displaying them to the power plant manager.

Johan C
The tasks of this week were not of the type embedded systems so I lay focus on reading course literature to improve my understanding of scrum. And because of my role as Hardware Engineer I took a look at how we should set up the hardware and started to create the wiring diagram which I updated throughout the project.

Johan K
This week I worked on the tasks that are part of the scrum master role. I continued to do a lot of research about scrum and how I could improve as a scrum master. One of the problems that the team came up with from the retrospective meeting was to write better task descriptions on the scrum board in Trello. I worked on simplifying the scrum board, which led to us writing tasks and estimation directly under the user story in the backlog. This made the scrum board look more tidy. On Wednesday, Adam and I had a meeting with Andreas Rowell where we talked about the last sprint. In the application, I mostly worked with writing unit tests. Anton and I had a TDD session where we implemented the Radiator Calculator class. I also had a pair programming session with Adam, where we continued to implement the front-end.

William
Me and Albert got a really good “lecture” from Anton in our group on the basics of TDD. We now had a lot more knowledge on how testing works and how it can be done in Kotlin. We used this knowledge to start testing the Notifications feature in our app, and were able to successfully test this feature. We tested the Notifications together with Enis at Albert’s place. During the week we also decided to implement a “protocol” for sending data between the app and the safety console. This protocol is further explained in the Software Design section of this report. We also tested the app and were successfully able to clock in and clock out using the RFID tag on the safety console.

Sprint 3

Adam
This week the backlog had unexpected reprioritization. I assured my team members that it was nothing to worry about. The reason I could do this is because when researching the backlog I noticed that the tasks that were rearranged were mostly already done or soon to be done. That is, we would not suffer from this reporitization. Furthemore, the team members expressed an uncertainty regarding the UI/UX so I decided to help them with it, even though I was the product owner. This worked out pretty well, because I as a product owner is the one with vision over the product, this made it possible for me to convey my ideas visually.

Albert
This week I was pair programming with Enis and William. We created a Heartbeat function that sends a ping to Arduino to check if the bluetooth connection is still alive. We had some communication bugs with the hardware side, so we spent a lot of time trying to solve these bugs. The Hardware Team showed me how they wanted to receive the data so I can make changes.

Anton & Johan C
This sprint we implemented hardware for the LCD, and tested that it works alright. Then we focused on our communication protocol, we had to agree on a certain set of messages and how these messages would look like. When we had agreed on what it looked like with the rest of the group we implemented this protocol and then tested it with the app.

Enis
During the third sprint I was pair-programming with Albert and William. We did some error-handling regarding database-communication and implemented a function that sends a heartbeat to the safety console to let it know if bluetooth-communication with the app fails. I

did a bit of refactoring in different places of the code and also did some error-handling and improved the unit tests of the database.

Johan K
This week I worked on the tasks that are part of the scrum master role. I held meetings, made sure that the team could focus on the right things and to document the process. On Wednesday I had a last meeting with Andreas Rowell where we talked about how the last sprint went and he gave me some good advice on things that could be improved. In the application, Adam and I worked on restructuring the code so that the different status views in the interface were displayed in fragments instead of in activities. We spent almost two working days on this, because it needed to be done. I also implemented most of the UI unit tests for the main activity.

William
I did a lot of refactoring for the Bluetooth and Notifications modules of the app, and also a little bit for the overall logic of the app. We also saw some issues with the Bluetooth communication, but we managed to fix most of them. I also managed to improve the radiation calculator inside the app so that it continuously tracks the exposed radiation every second, and uses that value for future calculations. I also improved the timer inside the app so the current time left (in minutes) is sent to the safety console every second. As usual me and Albert did these things while pair programming, testing it together with the guys from the embedded systems programme.

Sprint 4

Adam
This sprint was the last, and a short one. Many other project groups decided to not do a sprint 4. But during the sprint demo, the business owner, me as product owner, the project team and the scrum master decided to go for a sprint 4 as we were so close to finish some vital tasks. I started working on the report this week, did the demo video, and worked on the presentation. Once again I decided to work much on things that didn’t tie into the product directly, so the team members could focus on finishing sprint 4.

Albert
I worked with Enis on some tasks fetching sessions from the database and showing activity data for the administrator and some specific data for the technician. We also created some views to show this data and wrote some functions to take care of time conversion and radiation exposure calculation.

Anton & Johan C
This sprint we added additional features such as a potentiometer for radiation level simulation and buttons for the rooms and hazmat suit. Since we had implemented our communication protocol this part was very easy to do. After this we had some refactoring to do, to make our code base easier to understand and readable.

Enis
In this sprint I created a new fragment where a technician can view their previous clock in/outs and implemented functionality for retrieving and displaying that together with Albert. I did some design changes in the database and in the code for how data should be stored in the database. I also implemented functionality for storing the latest radiation level, current room the technician is in, time of each notification with radiation level warning and at what times the technician puts on or takes off the hazmat suit. I also made so that the application on start will get the correct data from the database and display it so in case the application crashes or restarts, we will always display the correct data. I also did some minor refactoring and error-handling. Me and Albert did some pair programming and implemented functionality and some basic UI for displaying all relevant data about different events for the safety console manager. We also implemented a function for calculating historical radiation-exposure per hour of the technician and made so that the technician can view this information.

Johan K
I worked with the usual tasks that are part of the scrum master role. Since this sprint was shorter than the other sprints, I focused on compiling data for both the presentation and the report. I worked a lot with UI unit tests. I implemented tests for the clocked in fragment and connecting fragment. I also moved and refactored all the tests for notifications. I had a lot of trouble with the different tests I wrote. For example, all the tests worked to run separately, but as soon as I ran the tests together, all the tests failed. Sometimes not a single test passed on my mobile phone but passed without any problems on the Android emulator. This was very frustrating and I had to spend a lot of time troubleshooting.

William
I helped fix the remaining bugs in the mobile app, and implemented the “potentiometer” feature in the app. This feature was relatively easy to implement since it basically uses the same communication protocol as for all the other messages.