Bachelor log 01. February 2023

Had a group meeting discussing the tasks for the next client meeting with Solwr, as per the following notes:

For the next meeting: Explore different tech for point one, and possibly 2. Get a better idea about what could work and not.

As well as the action points sent over email from Sondre from Solwr:

\*   Solwr sends Navitec API spec  
  \*   Solwr sends UX design sketches  
  \*   The Group explore possible tech for point 1 and possible point 2  
     \*   Discuss pros and cons  
  \*   The group brings NTNU standard NDA agreement  
  \*   Physical meeting at Solwr HQ

We still have not received the API nor any UX sketches, which makes it difficult to create any concrete plans. We did, however, decide that we will be using React native to program the app, and that the app will scan for both IP-addresses and names (assuming there is a predefined naming convention for the robots).

We also decided to use QR-codes to allow the user to choose which robot to connect to directly, without having to scan the entire network and find the correct robot from the resulting list.

We also translated all previous meeting notes to English.

**Following are the notes from the session in their entirety:**

**Notes on point 1:**

**We need a way to differentiate the IP-addresses of the robots from the other units on the network. Pre-defined list? Can the local server on each robot contain a list of robot IPs? Could also just use host name, assuming these follow a predefined standard.**

**Notes on point 2:**

**All robots could also have a QR-code that could be scanned via the app, so the user can choose the robot directly, instead of needing to scan the entire network and find the correct robot from the list.**

**Conclusion:**

**Might as well use QR-codes for directly choosing a robot in physical proximity. This QR code could contain the relevant info of the robot, like name and IP-address**

**Framework:**

**We have decided to develop our app using the React Native technology. The reasoning behind this is that by default, React Native is a cross platform framework. This allows the app to not just be run on android, but on IOS as well, with the same code base. Furthermore this allows for faster development and better maintenance, since we are only working on one cross platform app and not two separate. (Android native, and IOS native).**

**Idea:**

**We create an app (using the React Native framework) which its functionality is to create a connection with the desired robot. 2 buttons will be displayed when the app is opened, ‘Scan network for robots’ and ‘Connect with QR code’. Network scan is a feature we will be implementing in our code, as well as the QR code connect function. We figured UDP sockets would fit well for this feature. (Still need to figure out how to send commands like moving, using the Navitech API combined with the sockets)**