Individual goals

1. Creating our own modified version of the Rust nRF24l01+ library. The existing Rust library does not seem to be functional, most likely because it is too outdated. But we would still prefer to use Rust in this project for a few reasons:

   - Rust would also run much faster than Python, while being easier to make memory- and runtimesafe than C
   - We both currently take the course in functional programming, where Haskell is used. So it would be nice to solidify own knowledge by using Rust in this course
   - We think that we could make the existing library more effective, by decreasing the address length used in the nRF24l01+ transciever from 5 bytes to 3 bytes

   This could would be evaluated by running the code. If it managed to transmit data between the transcievers, our implementation of the library works. To determine if the decrease in address length is successful, we could show the code determining the address length in the library and that the program still works with it.

2. Implementing CI/CD to make the development process as easy as possible, by rebuilding and redeploying the code with a single command. Depending on how well this is done, the initial setup could also be minimized through deploying scripts to the Rasberry Pis

   This could be evaluated by running the CI/CD pipeline and checking that the code is deployed to the Raspberry Pis. If the code is deployed to both Raspberry Pis, where one Pi becomes the base station and the other one a mobile unit, the CI/CD pipeline works.

3. Using tailscale to use the same IP address for the Raspberry Pi in the lab and at home. This would simplify the setup when moving the Pis between networks.

   This could be evaluated by connecting to the Raspberry Pis using the tailscale IP address instead of the home- or lab addresses.

4. Set the longge interface as the default network interface, which allows us to connect to the mobile unit without an Ethernet or WLAN connection. This would not bring us any performance benefit, but would however provide an additional opportunity to access the Pi even without the ethernet cable. We would also learn even more about network interfaces through implementing this feature.

   This could be evaluated by checking the result from the `ip route list`. If our longge is the default interface, the implementation works. We could

also check it by connecting to the Raspberry Pi without an Ethernet or WLAN connection.

5. Host a Minecraft server on the mobile unit. We both thought this could be a fun way of demonstrating the performance of the network interface. Through some initial googling, it seems that we would require at least 100 Mb throughput per hour, corresponding to about 0.03 Mbps.

   This could be evaluated by connecting to the Minecraft server and playing the game. If the game doesn't crash, and allows at least one player to play on the server, the implementation works.