SUMMARY

The chapter provides a summarized overview of what has been accomplished during this project, both in terms of the framework and of the creation of a GUI.

The framework created allows to connect to the U-CAT ROS messaging service via a web server communicating with WebSocket. The connection is established automatically during startup of the GUI and reports an error if the server is not reachable. In case of a change in base parameters (IP address or port), corresponding values can be easily updated in the code. The GUI has been tested on the real vehicle, and data reception and display from five topics as well as data transmission of one topic have been carried out successfully. The display modes have been designed to match the data to be displayed and to meet ergonomic requirements of readability and comprehensibility. The framework offers various possibilities for changing display options, many of which have not yet been implemented on the graphical level, which means, these cannot be modified during runtime.

The ease of usability of the framework as per its current state of development has been tested by defining a new display category, and the code has been optimized to minimize further programming effort when such basic extensions to the GUI become necessary.

All in all, it has been proven that the Processing code dialect and the functions derived from the `PApplet` class offer convenient shortcuts for the creation of a graphically elaborate and specifically designed GUI. The framework developed in the scope of this project demands a higher effort to implement updates and changes in the GUI than a typical toolkit does, however, the graphical result promises to be of higher quality and ergonomic integrity than by using generic toolkits.

KOKKUVÕTE

Käesolevas bakalaureusetöös loodud tarkvararaamistik võimaldab ühenduda allveeroboti U-CAT operatsioonisüsteemi ROS sõnumvahetuse teenusega. Seda tehakse robotis oleva veebiserveri WebSocketi pordi kaudu. Ühendus robotiga luuakse graafilise kasutajaliidese käivitamisel automaatselt, aga kui server ei ole kättesaadav, siis kuvatakse veateade.

Kasutajaliidest on testitud päris robotil. Samuti on näidatud andmete kättesaamine ja kuvamine viielt erinevalt teemalt (anduritelt) roboti operatsioonisüsteemi kaudu, ning andmete saatmine ühele teemale (täituritele). Põhiparameetreid, nagu IP aadress ja port, saab koodis lihtsasti uuendada.

Kuvarežiimid on disainitud nii, et nad vastaksid andmete iseloomule ja oleksid ergonoomilised, loetavad, ja arusaadavad. Raamistik annab erinevaid võimalusi, et muuta kuva parameetreid, neist mõned ei ole veel graafiliselt implementeeritud. Kasutajaliidese laiendatavust on testitud uue kuvakategooria defineerimisega. Koodi on optimeeritud vähendamaks pingutusi uuenduste ja paranduste programmeerimiseks.

Kokkuvõttes, selles töös on tõestatud, et Java koodidialekt Processing ja funktsioonid, mis on tuletatud PApplet klassist, lubavad luua väiksema vaevaga graafiliselt läbimõeldud ja konkreetse rakenduse jaoks disainitud graafilist kasutajaliidest. Töö käigus loodud raamistik nõuab parandusteks ja uuendusteks suuremaid jõupingutusi kui mõni teine tüüpiline komplekt tarkvaralisi tööriistu kasutajaliideste loomiseks. Samas on tulemus (ehk graafiline kasutajaliides) kõrgema kvaliteediga ja ergonoomiliselt terviklikum kui mõni üldine komplekt tarkvaralisi tööriistu.