Mittuniversitetet
MID SWEDEN UNIVERSITY

Multifunctional Control Grip

A Joystick especially developed for forest regeneration.

Fredrik Elfström
Karl Sjölin
Niklas Zetterberg
Willow Noh
Ali Gharieb

Project based product development
Högskolepoäng: 15 hp
Termin/år: HT24
Handledare: Ruben van Westendorp, Martin Haglund, Jimmy Westberg (Bracke Forest AB)
Examinator: Rickard Hamrin

Sammanfattning

Bracke Forest AB utvecklar ett nytt styrsystem till sina skogsmaskiner. De befintliga manöverdonen som används idag passar ej direkt med det nya styrsystemet varpå gruppen fick i uppgift att utveckla ett nytt manöverdon som kan kommunicera med det nya styrsystemet via moderna protokoll. Utveckling av detta manöverdon skedde på tre fronter, ett nytt fysiskt handgrepp, ny tillhörande elektronik och ny mjukvara. Projektet utvecklades i iterationer varpå den nästa byggde på den föregående iterationen. Projektet resulterade i två nya handgrepp med anpassad elektronik och mjukvara som säkerställer rätt kommunikation samt medger omprogrammering av funktioner av användaren.

Abstract

Bracke Forest AB is developing a new control system for their forestry equipment. The control grips currently used are not directly compatible with this new control system. The group was tasked to develop a new multifunctional control grip that can communicate with the control system via modern protocols. Development was done in three different areas, new grip design, electronic design and software development. The development was conducted through iterative phases where the new iteration built upon the last. This project resulted in two new handgrip designs, a custom circuit and software that ensures the right communication as well as programmability of functions by the user.

Preface

The students who worked on this project would like to thank our project providers at Bracke Forest AB who made this project possible. With a well-defined mission with a real-world application, we students have enjoyed ourselves and learned a lot during the project. We hope Bracke Forest AB continues to provide these types of opportunities for future students.

Table of Contests

Terminology

Acronyms

SVN	Apache Subversion
API	Application Programming Interface
IDE	Integrated Development Environment
HID	Human Interface Device
USB	Universal Serial Bus
PCB	Printed Circuit Board
C++	An object-oriented programming language created by Danish computer scientist Bjarne Stroustrup
C#	An object-oriented programming language created by Microsoft that runs on the .NET Framework
SQL	Structured Query Language. A domain-specific language used to manage data.
PostgreSQL	A powerful open-source object-relational database system.
WPF	Windows Presentation Foundation. A free and open-source user interface framework for Windows-based desktop applications.
XAML	Extensible Application Markup Language. A declarative language based on XML(Extensible Markup Language).
GUI	A graphical user interface.
CAD	Computer aided design
pgAdmin4	A PostgreSQL database tool management application.
I2C	Inter-Integrated Circuit.

ICSP	In-Circuit Serial Programming

1 Introduction

1.1 Background and motivation

Bracke Forest AB has initiated a development process of its control systems, which were previously based on the CANopen protocol, which now is considered outdated and limiting in Bracke Forest application. Instead, the new control system will be based on a Windows platform, enabling fewer cables, easier installation and maintenance, new actuators, and new configuration options. To communicate with the new control system, the XInput protocol must be used.

1.2 Overall aim and problem statement

By developing a new and improved control system Bracke Forest AB unlocks a plethora of new possible functions and improvements. One of these is new control grips with enhanced functions like analog inputs, user feedback and remapping of functions. All these new functions will enhance the usage of forestry equipment, possibly resulting in fewer accidents, more precise actions and yielding higher productivity.

The overall aim of this project is to aid Bracke Forest AB in this development journey by developing a new multifunctional control grip that the company can use as a foundation for further development in the future.

1.3 Overarching goal

The goal was to develop a functioning high-level concept of the controller and the communication to a windows-based computer. The following goals need to be fulfilled for the project to be considered successful.

- Develop a new control grip that fulfils the technical demands stated by Bracke Forest AB (see Appendix B: Requirements).
- Creation of a custom I/O box that connects the handgrip to the new control system via a custom PCB.
- Develop custom software for the handgrip that interprets the signal from the controller and allows programmability of output functions.

see Figure 1 for a scheme displaying the different components of the device.

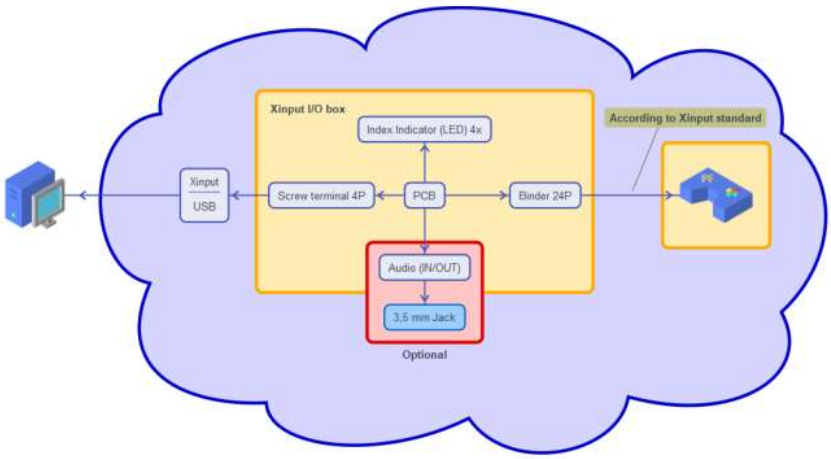


Figure 1, Scheme displaying a diagram for the control device

1.4 Division of work

Fredrik Elfström: Mechanical design, Coordination

Written chapters: Abstract/Sammanfattning, Preface, 1.2, 3.1.9, 3.1.10-3.1.14, 3.2.2, 3.2.3, 3.3.4-3.3.6, 3.4, 4.9, 4.10, 4.11, 6.1, 7.1, 7.2, 7.7, 8.1.1-8.1.3.

Karl Sjölin: Mechanical design, Test leader

Written chapters: 1.3, 2.3, 2.4, 3.2.1, 4.12, 4.13, 4.14, 6.2, 6.3, 11, 12, 13, 14, 15

Niklas Zetterberg Wallin: Programming

Written chapters: 3.1.1-3.1.7, 3.3.1-3.3.2, 4.1-4.8, 5.1, 7.3, 7.4, 7.5, 8.1.4-8.1.5

Willow Noh: Programming

Written chapters: 2.1, 2.2, 3.1.1-3.1.7, 3.3.1-3.3.2, 4.1-4.7, 4.8(diagram), 5.1, 7.3, 7.4, 7.5

Ali Gharieb: Electronic design

Written chapters: 3.3.3, 4.4, 5.2, 5.2.1, 5.2.2, 5.2.3, 5.2,4, 5.2.5, 5.2.6, 7.6, 8.1.6

2 Theory

2.1 Bracke Forest AB

Bracke Forest AB is a manufacturer which specializes in forestry equipment such as disc trenchers for scarifier, mounders, planters and felling heads. Established in 1922 by Hjalmar Ljungberg, the company has focused on producing machines to conserve the environment while maintaining the standards for technical and economic efficiency in forestry. Commenced in Bräcke in Jämtland, Sweden, the company has a presence in Northern Europe, North America as well as South America and it also attracts interest in Asia where its proprietary company, Komatsu Ltd. is located [1].

2.2 Mechanized forestry

The mechanization in forestry is a process that adapts various machines and technology to substitute human labor. Starting from chainsaw and debarker in the mid-1900s, diverse types of heavy instrument are presently operated in the industry including harvesting, planting, transporting and maintaining [2]. Mechanization contributes to ensure the effectiveness of productivity and the sustainability of resources by increasing the precision of work [3].

2.3 Forestry platforms

Bracke Forest AB develops and manufactures scarifiers, equipment for mechanized seeding, planting machines and felling heads for biomass [4].

Bracke makes a few versions of Disc trenchers that gives plants and seeds the best possible start for growth and survival (Figure 2). It works by spinning large discs to turn over the soil.

Figure 2, two-row disc trencher behind forest machine

The mounders from Bracke are used to create humus mounds with mineral soil cover, which makes for very good planting spots (Figure 3).

Figure 3, Mounder creating humus mounds behind forest machine

Planters and seeders conduct regeneration work from scarification to planting. The machine makes a mound of inverted humus, compacts it and sets a plant in the middle of the mound (Figure 4).

Figure 4, Excavator using a planter

Felling heads are used on forestry and bioenergy. It is used to cut down and lift grown trees which are ready for harvest (Figure 5).

Figure 5, The felling head being used to harvest.

2.4 Current control grip

Bracke Forest AB has two current solutions of control grips that can be seen in Appendix D: Operating control.

The first type of control grip has four buttons that can be reached using the index finger, middle finger, ring finger and pinkie. two two-way switches, whereas one is operated using the thumb and the other using the index finger. The handle features one four-way switch which is operated by the index finger.

The other type of control has five buttons. Two buttons can be reached using the thumb and the other three using the index finger, ring finger and pinkie. The handle features two four-way switches and one analogue joystick which are all operated using the thumb.

3 Methodology

The project was split into three different phases, pre-study, creative- and development. Each following phase uses the last as a new starting point to further develop the project.

3.1 Pre-study

In the beginning of the project a pre study was conducted to provide sufficient information for further development. A big part of the pre-study was conducted via literature search.

A visit and interviews with employees at Bracke Forest AB were also conducted to give more clarity and detail to the given problem.

3.1.1 AI-Assistance tools.

To speed up the development of the software development and technical report writing. Different Ai-Assistance tools were researched and studied. The research was conducted by online research. The AI studied was made by the university on lectured held classes regarding AI assistance.

3.1.2 Version Control software

Among the main type of version control systems, centralized system e.g., SVN and distributed system e.g., Git, Git is examined since the system is primarily utilized in the programming field and the Mid Sweden University encourages its use with an introduction in the individual lecture. GitHub and GitHub Desktop are studied as version control software platform and tool.

3.1.3 XInput

In the pre study the XInput protocol was examined by online research. Keyword “XInput”,” XInput Arduino”,” XInput controller”, “XInput hardware requirements”, etc.

3.1.4 Hardware requirements

Suitable hardware was researched by examining what type of microcontroller is compatible with the XInput library by online-research.

3.1.5 Programming language

To determine what Programming language is most suitable for the project’s development, online research was conducted. This research aimed to make an informed and educated choice based on various factors such as performance, community support, ease of use and compatibility with the projects demands.

3.1.6 IDE

Different IDEs which support the XInput library and IDE for the remapping program were examined by online research.

3.1.7 Database management

To fulfil the request of Bracke Forest to use PostgreSQL. PostgreSQL database was studied, different database management tools was examined to determine the most effective solution for our needs.

3.1.8 Proof of concept/ Pretotype

To ensure that the chosen hardware, software and protocols were working as intended with each other a "proof of concept"/" pretotype" was created. The pretotypes purpose is to ensure that a human input could be interpreted by the hardware and later send a XInput signal via USB-HID to the receiving windows computer.

3.1.9 Initial handgrip assessment

To create an initial starting point for the design of the handgrip, multiple simple handgrips were created using clay and evaluated with user test. The different grips were sculpted in modelling clay, 3D-scanned and 3D-printed. The grips were of varying size, orientation and had different supports for the hand. These grips were thereafter user tested to extract the positive and negative attributes of the handles.

3.1.10 Button placement areas

Based on the "best" handgrip assessed in the previous chapter, button placement areas were examined. This by identifying suitable button placement areas via internal group testing.

3.1.11 Problem analysis

Based on the pre-study and the information given by Bracke Forest AB, a more detailed problem analysis was created. The aim of the problem analysis was to extract the most important aspects of the problem and to concretize them into different partial problems.

3.1.12 Function analysis

The handgrips different required and desired functions was examined to further understand the task at hand. This analysis was carried out by dividing the functions into main-, sub- and support functions.

3.1.13 External analysis

An external analysis was conducted to examine different existing solutions that fall into the same scope as our problem. The external analysis was carried out as a literature search where different websites were examined.

3.1.14 Requirements specification

The requirements were set by Bracke Forest AB as well as by the group, based on insights during the pre-study.

3.2 Creative phase

3.2.1 Concept generation

During the concept generation, different areas were targeted one at a time. After sketching by hand, the first iteration of the grip was made using modelling clay. 3D-scanning was then used to create a digital model of the clay-models. All scanned models were 3D-printed to conduct user tests on. The test consisted of the test person placing all grips in order from worst to best according to the test user's opinion. This gave each grip a point from 1 to 4. At the end of the user testing the points were summarized to a total score. The test users were also asked to make positive and negative comments about each grip. They also had the possibility to make the own comments regarding testing and the models (See Appendix E: User testing sheet).

3.2.2 Concept evaluation

After the concept generation was each concept evaluated with the assessment criteria to rank the different concepts. The strengths and weaknesses of each concept were also extracted.

3.2.1 Final concept

A final concept was chosen via the concept evaluation chapter, this final concept is based upon the highest ranking of the assessment criteria. This concept also merges some of the best qualities of all the different concepts.

3.3 Development phase

3.3.1 GitHub

GitHub is utilized during the development phase to manage code changes while working in the team.

3.3.2 Remapping program

The client requested a background application that relays information from the controller to a database. Also integrate a simple GUI that has remapping functionality.

3.3.3 PCB Design

This project's electronic design focuses on practical application, aiming to design a fully integrated PCB capable of performing all prototype functions. The objective was to create a single circuit board, eliminating the need for multiple PCBs to divide or supplement functionalities.

In designing the PCB, multiple critical requirements were addressed to ensure reliable operation based on the technical requirements. Stability, robustness, and durability were prioritized to withstand operational stresses and meet potential future requirements. Altium Designer software was used for both schematic capture and PCB layout.

3.3.4 Prototyping work

With a final concept chosen a more detailed prototype was created with the concept as a starting point. Based on the sketches, clay models and 3D-scans, a higher resolution 3D-model of the handgrip was modelled in iterations using CAD.

3.3.5 PCB enclosure

Based on the technical drawings for the PCB an enclosure was designed to house it. The enclosure provides a safer environment for the electronics and makes mounting possible.

3.3.6 Mounting solution

When the last iteration of handgrips was complete a mounting solution was designed. The mounting solution were made to be compatible with the handgrip and provide a flexible mounting solution.

3.4 AI assistance in text

Generative AI has been usen in chapter 3.3.3 and 5.2.1 to refine the text.

4 Pre-study

4.1 AI-Assistance tools

Various AI-Assistance tools will be used because there are too many to specify, however, the main AI's will be Windows Co-pilot, ChatGPT and GitHub Co-pilot.

4.2 Version Control Software

For this project, Git will be utilized as the version control software. As a git platform GitHub will be used for managing code version, bug tracking, collaborating with team members, and tracking changes [5]. GitHub Desktop, an application that helps Windows users to interact with GitHub will be applied for the project additionally.

4.3 XInput Protocol

XInput is a Microsoft API which collects input from a user via Xbox controller on Windows systems. XInput supports controller query, vibration effects and voice input/ sound output via headset. It also provides suitable functions for a controller, i.e. checking controller state, dead zone implements for thumb sticks and triggers [6].

For Xinput to function in our system, the requirements for the device have been checked, such as HID-function, USB physical layer, USB endpoint buffers, control transfer unit, USB descriptors, circuits for haptic feedback and firmware.

4.4 Hardware requirement

The important factors for the hardware in the project are divided in two parts: the first part is a prototype for the test, Leonardo Arduino Board, combined with push buttons from digital inputs and a built-in joystick from analogue inputs. The second part is a PCB that contains ATmega32u microprocessor. This processor provides HID functionality. The electrical circuits will be built which are needed for haptic feedback (vibrations), joystick, LED-indicator, power supply and data transfer. To be able to communicate via the XInput protocol the hardware must be acknowledged by the XInput library [7]. Since Arduino Leonardo that contains ATmega32u meets the requirement for XInput library, it is chosen as a prototype test [8].

4.5 Programming language

As by the result in 4.4 Hardware requirement the chosen programming language for PCB will be a modified version of C++. It is modified in the sense that it works with the ATmega32u4 microcontroller.

Since the controller needs to be tuned to the user's demand prior to operation, a remapping process must be required. The remapping application will be written in C# which is the most frequently used programming language at Bracke Forest as well as a user-friendly language for Windows application with GUI development. XAML will be partially implemented as part of the GUI development process. SQL will be used to handle the data from the database of the company.

4.6 IDE

The developers of the group, after research, will be using the Visual studio code with PlatformIO extension and Arduino IDE which supports the XInput library while programming a PCB with ATmega32u chip.

To build a remapping program, Visual studio 2022 is used as IDE. This program is developed by Microsoft which supports tools and features for Windows application development [9].

For the circuit diagram and design for PCB prototype, a software Altium designer that provides electronic design automation will be utilized. LTspice is additionally used for circuit simulation.

4.7 Database management

The database is written in PostgreSQL and is requested by the customer to look a specific way, see Figure 6. The database table structure requested from Bracke Forest AB.

```SQL
CREATE TABLE xinput_list (
    index_ INTEGER PRIMARY KEY CHECK (index_ BETWEEN 0 AND 4),
    state_ JSONB,
    command_ JSONB,
    timestamp_ TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
);

CREATE OR REPLACE FUNCTION update_timestamp()
RETURNS TRIGGER AS $$
BEGIN
    NEW.timestamp_ = CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_timestamp_trigger
BEFORE UPDATE OF state_, command_ ON xinput_list
FOR EACH ROW EXECUTE FUNCTION update_timestamp();
```

Figure 6. The database table structure requested from Bracke Forest AB

Index_ is for what index the controller holds, as XInput can handle up to 4 controllers the index gets tracked here.

state_ hold the current state of the controller, such as x and y values for the joysticks, trigger values (0-255) and the bitmap for the button pressed.

Command_ defines what the controller itself is supposed to do at a certain moment, such as haptic feedback or displaying led indicators.

Timestamp_ will display the timestamp of the last changed state.

To manage the database the developers will be using pgAdmin 4.

4.8 Electronic/programming Electronic PoF

The proof of concept was created on an Arduino Micro Pro board, which can be seen in Figure 7. The concept was completed and successfully where button and a joystick were functional by running XInput. See Figure 8 for wiring circuit.

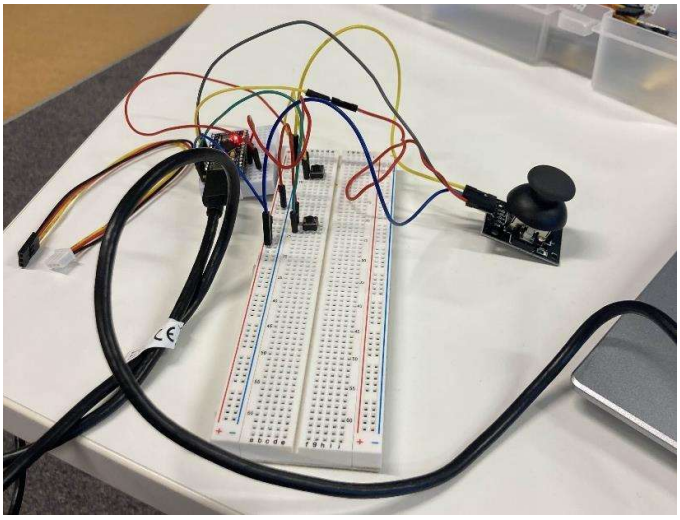


Figure 7: Proof of concept with Arduino Micro Pro

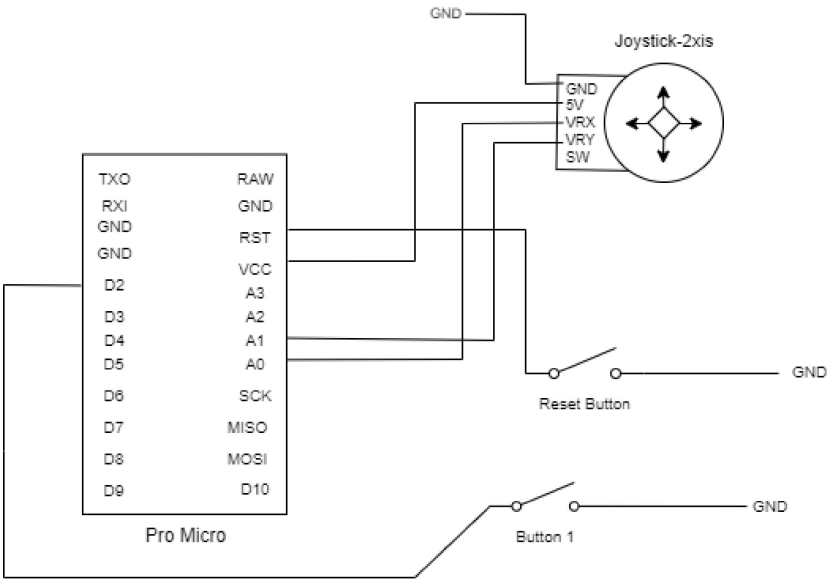


Figure 8: Circuit diagram

4.9 Initial handgrip assessment

Four different handgrips were created. Every handgrip placed the palm in a different angle. These varied from fully horizontal to fully vertical. The handgrips are seen in Figure 9.

Figure 9: Initial handgrip design, model 1-4 from left to right.

The test form in Appendix E: User testing sheet was created to evaluate the four different grips. The users sorted the grips from worst to best and noted positive and negative comments for each grip.

In Table 1 the results from the user form can be seen. In conclusion the users preferred a more upright tilted grip, but with a slightly increased tilt forward than the tested grips. Keep in mind that hand sizes vary. Don't keep the grooves for the fingers.

Table 1: Combined results from user testing of initial handgrips

User test 1				
Grip	Total points	Positive comments	Negative comments	Our reflection based on responses
1.	22	Good support and a good angle for the wrist, the small size give a firm grip	uncomfortable grooves for the fingers	A smaller grip appears to be desired. Aswell as a grip with the palm facing downwards appears to give the wrist sufficent support and angle.
2.	32	Stable and neutral grip with the wrist in a pleasant angle	Not enough space for the fingers	A thin grip appears to provide a firm grip. Positive response was given for the angle of the wrist.
3.	15	Plenty of space	Too big and hard to grip	A grip with a large diameter appears difficult to grasp.
4.	21	Comfortable and thumb in good position. Neutral grip	Not enough stability "outwards", too large and the fingers don't fit the grooves.	Large diameter grip appears hard to grasp, it is however comfortable. The angle of this grip provides a neutral position for the wrist.

4.10 Button placement areas

4.10.1 Thumb input

The viable area that can be used for thumb inputs was researched with user testing on one of the handgrips. Thumb length and reach was documented for 11 individuals. Measurements taken can be seen in Figure 10. And results can be seen in Table 2: Results from thumb measurements.

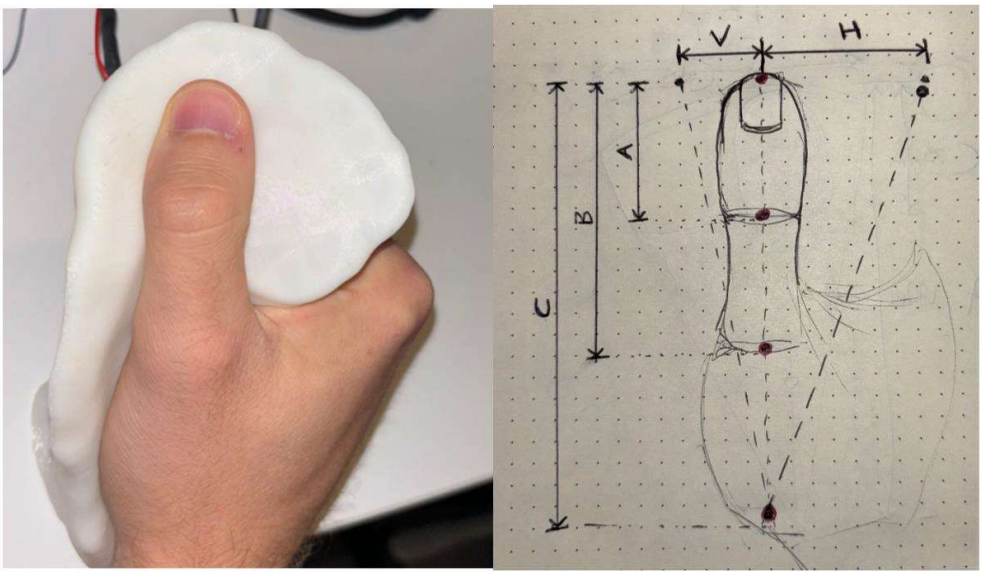


Figure 10: Viable thumb input area measurements.

Table 2: Results from thumb measurements.

												Mean:
A	30	35	32	30	38	33	30	28	32	35	32	32,3
B	62	65	70	60	68	66	60	62	64	72	63	64,7
C	110	110	115	90	105	109	93	100	102	112	99	104,1
V	20	20	10	20	11	20	30	10	18	18	15	17,5
H	40	45	40	40	40	32	50	26	30	38	35	37,8

From this data an approximate button placement area was designed that can be seen in Figure 11. This area uses the mean values from Table 2.

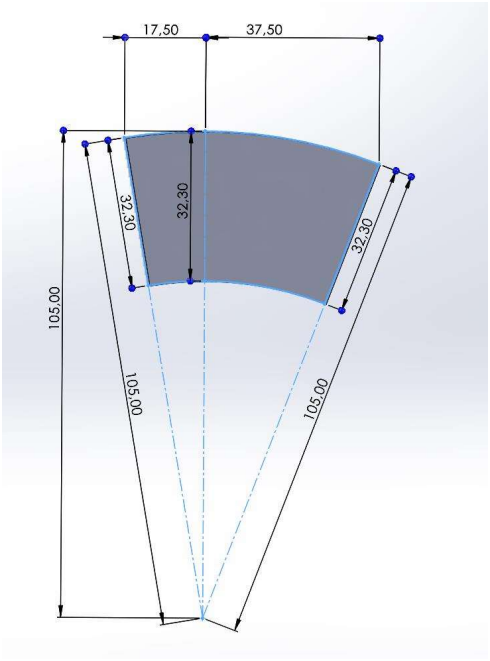


Figure 11: Thumb input viable area

4.10.2 Index-/ middle finger inputs

In Figure 12 three different input placements areas can be seen, one with buttons in the neutral grip position, one with buttons on the outside of the grip position and one with buttons on top over the grip.

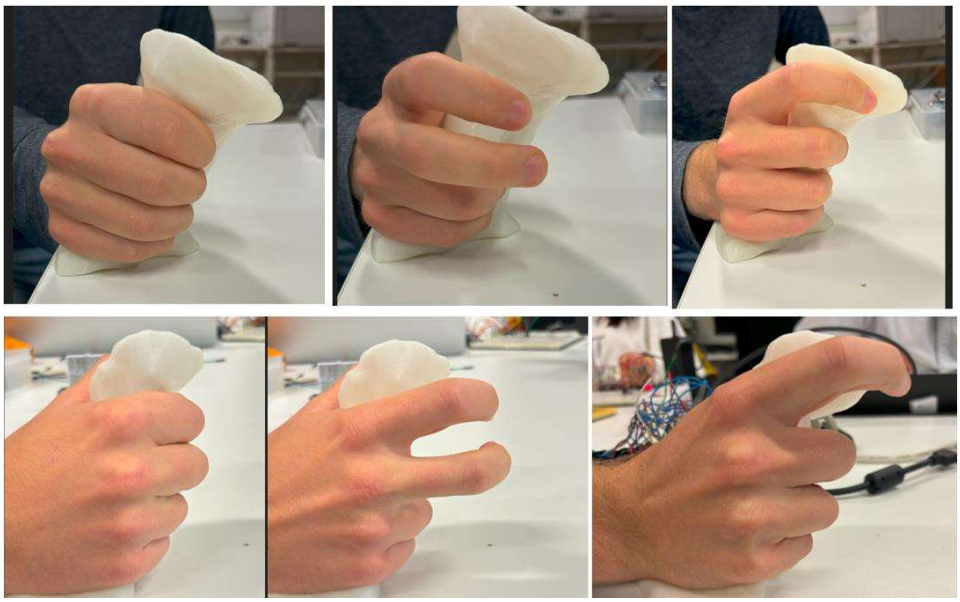


Figure 12: Different index-/middle finger input placements, left: Buttons in neutral grip position, center: Buttons outside of grip, right: Trigger/buttons over grip.

4.10.3 Other input areas

A few other input areas were found but deemed not as relevant for the project. This includes areas below the handgrip where the operator must lift their hand off the handgrip.

4.11 Problem analysis

In Table 3 the results of the problem analysis can be seen, with main and partial problems of the project. All groups are responsible for creating the multifunctional control grip and placement of the input devices. The mechanical design engineers are responsible for designing the handgrip. The programmers are responsible of Programmability of input devices, remapping application and database. As well as communication between the PCB and PC. The electrical engineer is responsible for hardware specification and design of electronic hardware. Interpretation of input devices to PCB is also the electrical engineer's responsibility.

Table 3: Problem analysis results.

No.	Problem	Type	Responsibility
0	Creation of multifunctional control grip	**Main**	All
1	Design of handgrip	Partial	Mechanical

2	Placement of input devices	Partial	All
3	Specification/design of hardware	Partial	Electrical
4	Interpretation of input devices to PCB	Partial	Electrical
5	Communication between PCB and PC	Partial	Programming
6	Programmability of input devices, data-base	Partial	Programming
7	Manufacturing	Partial	Mechanical

4.12 Function analysis

Main function: Without this the product is useless

Sub function: Function that is needed to fulfil the main function

Support function: Function that is good, but not necessary for the product.

The results of the functional analysis were made as a matrix that can be seen in Table 4.

Table 4: Tabel with functional analysis.

Functional area	Function		Class	Comments:
Main	Allow	Communication	Required	Send specific commands
Technical functions				
Sub	Interpret	Human interaction	Required	buttons, joysticks, triggers
Sub	Send	Chosen signal	Required	Xinput
support	provide	feedback	desired	Haptic
Ergonomics/use				
sub	Provide	grip	Required	Form, material, surface
sub	Provide	Support	Desired	rest hand
support	provide	adjustability	Desired	
support	counteract	injuries	Desired	
support	minimize	strain	Desired	lower arm
Design				
sub	enable	mounting	required	
sub	fit	components	required	Buttons, joysticks, pcb, etc
sub	Support	operator	required	Stress, during regular use
Manufacturing				
sub	allow	manufacturing	required	batches of 10-20

sub	enable	assembly	required	
sub	enable	disassembly	required	

4.13 External analysis

The external analysis resulted in several products that served as inspiration for the first iteration of grips. Among their hands on throttle and stick (HOTAS) systems as seen in Figure 13, computer mice & keyboards as in Figure 14, pistol and rifle grips as well as the current solutions.

Figure 13: Saitek HOTAS system

Figure 14: Controller with small keyboard

4.14 Requirement specification

The requirements set by Bracke Forest AB were a grip suitable for the right-hand. It should also be able to connect to a Windows PC using Microsoft XInput (Preferably a separate I/O box). The grip should have at least the func-

tionality as the current grip used, preferably more functions. The final requirement is integrated haptic feedback, controlled via XInput (Appendix B: Requirements).

5 Implementation

5.1 Software development

In this chapter, the group aims to provide the reader with an understanding of the software development taken place during the project.

5.1.1 Overall vision

The project requires a controller connected to a PCB that is plugged into a Windows PC, using the XInput protocol for communication. Where the controller is to be identified as an Xbox controller by the protocol. Communicating with the Windows PC to the microcontroller.

While the controller is connected there is a background application program running on the Windows PC that is relaying the commands from the controller to a PostgreSQL database and containing a simple GUI for easy remapping of buttons.

5.1.2 AI-Assistance in implementation

The use of AI as our 6th member has been utilized while developing software, AI has been used in generating code for certain parts of the program.

The remapping application and Arduino code have been a symbiosis between the developers and AI-Assistance in both generating code and creating structure. Assistance AI has been heavily utilized during error handling for quicker bug fixing.

5.1.3 Libraries

Arduino compatible PCB:

- XInput 1.2.6 : A library allows emulating Xbox 360 controller using a USB-capable Arduino microcontroller. [10]
- DFRobot_MCP23017 : A 16-bit digital IO expansion IC that communicates with main-controller via IIC. It controls LEDs and a few digital pins for the project. [11]
- X360ControllersLED : An Arduino library that is designed to replicate the LED patterns of Xbox 360 controller. [12]

Remapping application:

- SharpDX.XInput : SharpDX is a cross platform library for rendering vector graphics on .NET. [13] SharpDX.XInput provides access to Xbox controller input.
- Hardcodet.Wpf.TaskbarNotification : An implementation of a NotifyIcon (aka system tray icon or taskbar icon) for the WPF platform. [14]

- Npgsql : An open source ADO.NET Data Provider for PostgreSQL, it allows programs written in C#, Visual Basic, F# to access the PostgreSQL database server. [15]
- System.Threading : It Contains types to support the Windows Presentation Foundation (WPF) threading system [16].

5.1.4 Main program for PCB

The main PCB program is written in C++ and is responsible for communicating between the AT32megau4 microcontroller and the Windows system.

The code is split into three parts:

- Initialization
- Setup
- Loop

In the **initialization** the code defines what pins on the microcontroller are used, and then placed upon a variable.

The **Setup** phase runs once when the microcontroller is powered up or rest. Any peripherals such as I/O pins are initialized. The range of joysticks and triggers from the XInput library is also set.

Loop continuously runs after the setup where the microcontroller is acting as a controller. Here it is defined as to what will happen when buttons on the multifunctional control grip are pressed and how we communicate it to the windows PC. Signals from the PC to the controller are also handled here such as feedback and led displaying.

See Appendix A: Source code for a full view of the code.

5.1.5 Program for remapping application

The remapping application is a Windows Presentation Foundation (WPF) written in C# and XAML built with .NET 6.0 also known as .NET Core. The application is responsible for sending information from the controllers' inputs to the PostgreSQL database as a background application. Once clicked on in the system tray it will display a simple GUI that enables the user to do simple remapping of the buttons on the controller and save and load profiles see Figure 15.

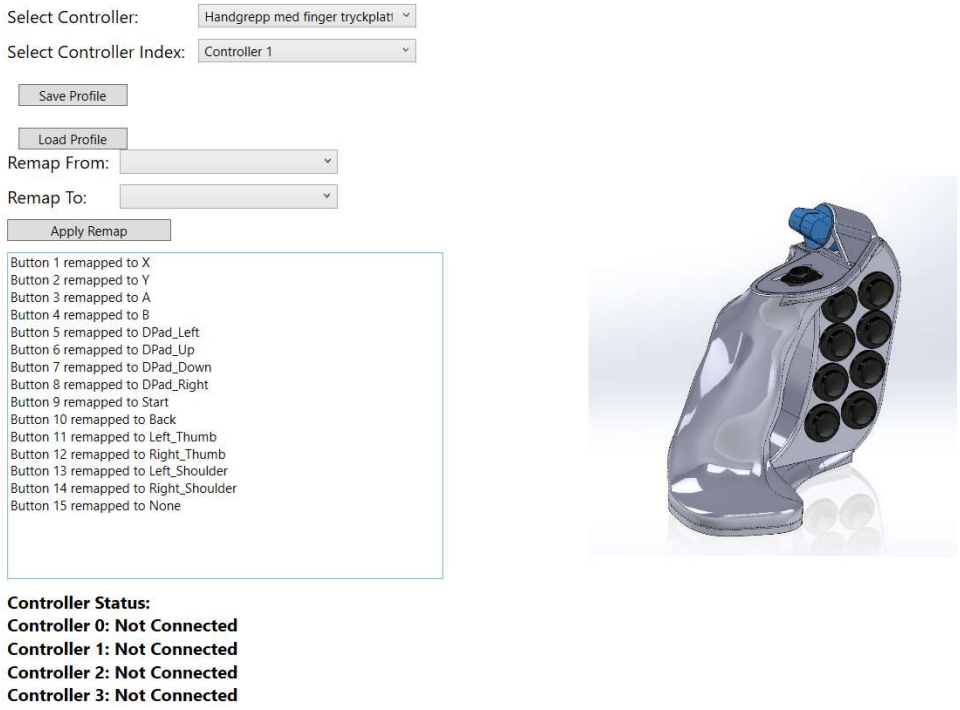


Figure 15: Remapping application GUI

The code is structured in different classes:

- **DatabaseHandler**
- **ControllerHandler**
- **RelayCommand**
- **ButtonMapper**

DatabaseHandler: The code in DatabaseHandler is built to handle the database connection and inserting / updating values to the database. The string variable "connectionString" contains the details require to connect to the PostgreSQL database. Method "AddToDatabase" is designed to insert new data or update existing data into a PostgreSQL table with three parameters, index, stateJson and commandJson with "ON_CONFLICT" handling. By the 'try and catch' statement, any exception will be caught with the error message.

ControllerHandler: This class manages the inputs from the controller, utilizing the sharpDX.XInput library. The event, defined by "StateChanged" is triggered when the controller's state is changed. The constructor "ControllerHandler" that takes an "Action<Gamepad>" as a parameter initializes the controller to the first user index as well as the previous state of the gamepad to an empty state. The method "StartReadingInput()" continuously reads the controller's state in a separate thread. It loops indefinitely in the background

(Task.Run), checking and updating the current state. "PauseUpdates()" and "ResumeUpdates()" handle pausing and resuming the controller's input reading. "GetCurrentState()" returns the current state of the gamepad if it is connected. Otherwise, it returns an empty Gamepad object.

RelayCommand: The code in this class simplifies the use of ICommand interface in the WPF, allowing functions to be assigned for executing actions and determining if the action can execute. "CanExecute" method allows UI elements to know if the command is enabled. "Execute" method executes the command action when invoked by "_execute" delegate. "CanExecuteChaged" event which is linked to "CommandManager.RequerySuggeste" ensures that the UI elements update if the command's "CanExecute" state changes.

ButtonMapper: This class handles the remapping of the buttons. It creates a new Gamepad object and applies remapping rules by copying values from the original Gamepad object to the changed gamepad object, and then changes the button states according to the remapping list. The constructor "ButtonMapper" initializes a new Gamepad object for "changedController" that stores a modified Gamepad state, as well as an empty list for "remaps" which is a list of tuples that contains a "from" and "to" button mapping. "AddRemap()" method adds a new button mapping to the "remaps" list and "MapButtons()" method modifies a Gamepad object's button states by iterating over the "remaps" list based on the remapping rules. "GetRemaps()" returns the list of remapping instructions in "remaps". The first static dictionary, "ButtonNumberMap" maps each button to an integer and the second, "ButtonMap" maps each button's name to its corresponding "GamepadButtonFlags" value. In association with these dictionaries, "GetButtonFlag()" method looks up "GamepadButtonFlags" value by its name from the "ButtonMap" and "GetButtonName()" finds the name corresponding to a "GamePadButtonFlags" value in the "ButtonMap".

The complete program code is available via the GitHub link in Appendix A.

5.2 Electronics development

5.2.1 Power and Communication

The PCB is powered via a USB connection, delivering +5V and up to 500mA. Two communication protocols were utilized: I2C and XInput. The primary component, the ATMEGA32U4-AU microcontroller [17], supports USB 2.0, enabling HID functionality for XInput protocol support. For stable USB communication, two 22-ohm impedance-matching resistors were added to the D+ and D- lines, along with a 100 nF capacitor to stabilize the internal regulator supplying USB power to the microcontroller.

5.2.2 Additional Components

The design includes an FA-238 16.0000MA30X-AG0 16 MHz crystal oscillator with a 1 MΩ bias resistor and two 22 pF load capacitors, with a critical capacitance value of 7 pF. To protect against ESD on the USB port, a TVS diode was added. Various resistors were included as pull-up, pull-down, and voltage-drop resistors for I2C protocol connections, hardware boots, and indicator LEDs. The circuits for the vibration feedback motors employ N-channel MOSFET transistors and diodes rated for up to 10 A.

To extend the digital I/O, an MCP23017-E/SS [18] component was implemented. The PCB design also features a reset function and supports further I2C connections for potential expansion.

5.2.3 Physical Specifications

The PCB is a two-layer board fabricated with FR-4 dielectric material at a standard thickness of 1.6 mm, with dimensions of 54 mm x 69 mm.

5.2.4 Assembly and Programming

A mix of SMD and through-hole soldering was used for component assembly. The PCB includes ICSP capability to facilitate initial programming, which uses SPI protocol.

5.2.5 External Connections

The PCB interfaces with a 24-conductor cable, with allocations for +5V, GND, 14 digital inputs, 6 analog inputs, and two outputs for vibration motors. The motor connections are routed to the drain terminals of two distinct N-channel MOSFET transistors and a + 5V power supply.

5.2.6 Final solution

The final PCB unconnected can be seen in Figure 16, and connected to USB-a and a 24p Binder connection can be seen in Figure 17.

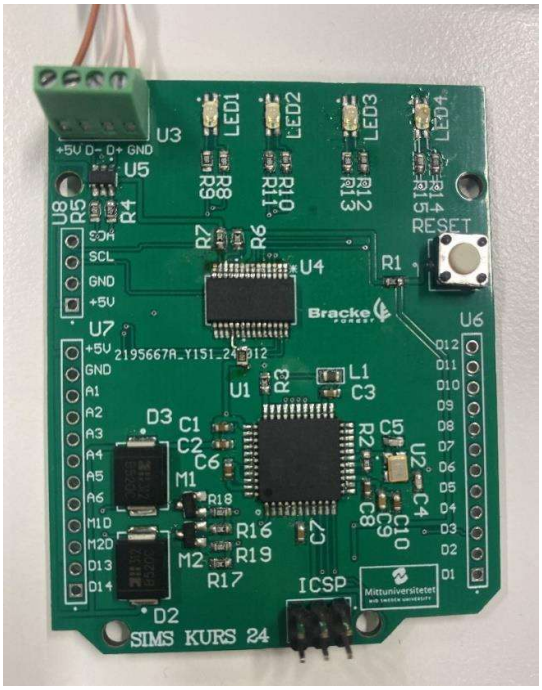


Figure 16:Final PCB design with soldered components.

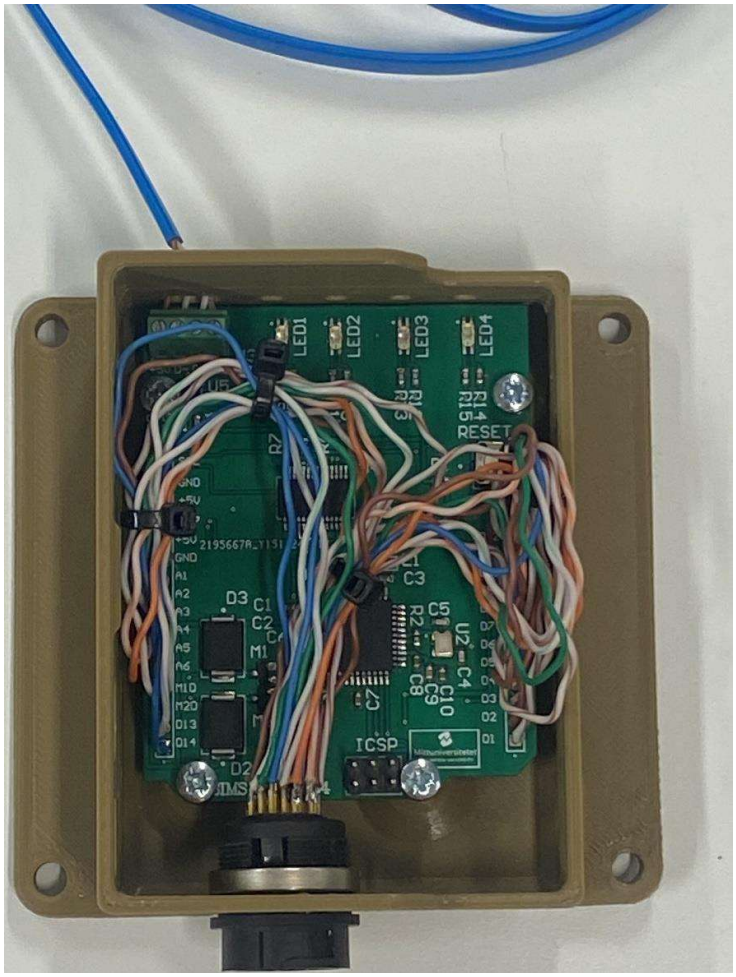


Figure 17: Final PCB wired with 24p binder connector and USB-A and installed in an enclosure.

6 Results, handgrip design

6.1 Creative phase

6.1.1 Concept generation

Nine different concept sketches were developed using pen and paper which can be seen in Figure 18. Based on the sketches, four clay mock-up models were created and later 3D-scanned and 3D-printed. Three of the clay mock-ups can be seen in Figure 19, and the 3D-printed mock-ups can be seen in Figure 20.

Figure 18: Concept sketches of different handgrips

Figure 19: Three concept mockups using clay

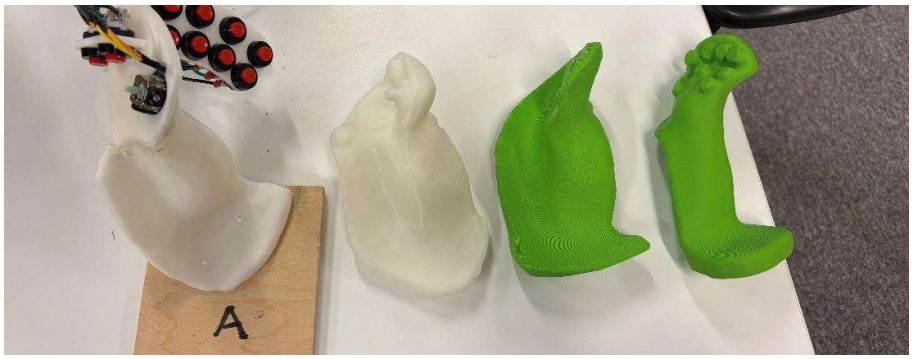


Figure 20: Four concept mockups 3d-printed from scanned clay.

6.1.2 Concept evaluation

The first user test focused on the grip and ergonomics of the controller. The group could draw the conclusion that an upright grip was better for stability, preferably with a slight angle forward and inward. Varying hand sizes was also to be considered for future models.

The focus of the second user test was button placement. Buttons that were easy to reach using the thumb was preferred and button placed on the back-side of the grip was not desired.

The results from the user testing can be seen in Appendix F: User tests.

6.1.3 Final concepts handgrip

For the final concepts of the handgrips two different approaches were chosen, one with fewer buttons, and one with more.

6.2 Development phase

6.2.1 Prototyping work/ iterations.

The first prototype had no buttons placed on the handle. The ergonomic properties remained from the clay models while having a 3D-model that was manageable.

The next iteration of handles featured two separate versions. One had the same grip with two buttons placed on the grip. They were accessible using the index- and middle finger. At the thumb a joystick was placed next to a four-way trim button. The other version placed eight buttons in front of the handle on a plane instead of placing buttons on the grip. As well as a 4-way trim button and a joystick placed at the thumb on the handle.

The third iteration had widened the area where the thumb is able to rest. The joystick was placed at an angle closer to the thumb next to the four-way trim button.

The fourth iteration focused on mount the actual buttons which require the handle to be hollow for cable routing and proper mounting conditions for the buttons and joystick (Figure 21, Figure 22).

Figure 21, Four iterations of the 3D-printed grips.

Figure 22, Four iterations of the 3D-printed grips using more buttons.

6.3 Final design

The final design of the mechanical part of the project was split into two approaches, one minimalistic, with few buttons called MD4, and one maximalistic, with more buttons called MD10.

6.3.1 MD4

The minimalistic grip is constructed from three pieces. The top that houses the joystick and four-way switch is removable to aid installation. The body is hollow to allow wires to the routed through. Inside the top piece there is a small part that fixes the analogue joystick in place.

The handle has two buttons, one four-way switch and one analogue joystick. By combining the buttons with the four-way switch the handle allows for at least digital 16 inputs.

A 3d-model of handgrip MD4 can be seen in Figure 23.

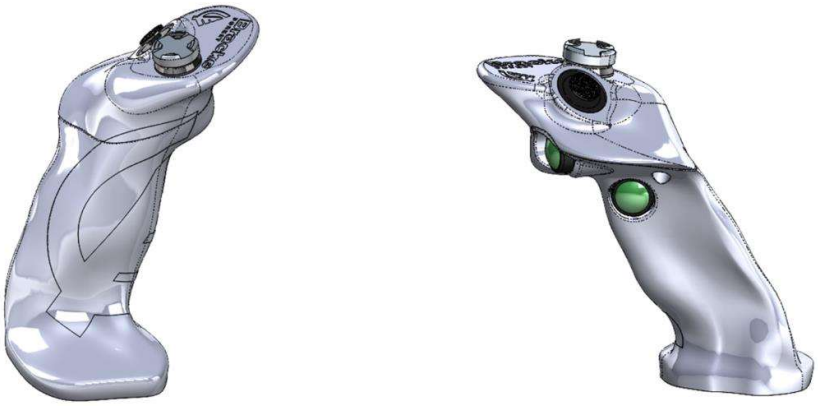


Figure 23: Model of handgrip MD4.

6.3.2 MD10

Maximalistic approach, MD10, provides 2 analog inputs via one joystick, 13 digital inputs via different buttons. The model of this handgrip can be seen in Figure 24.

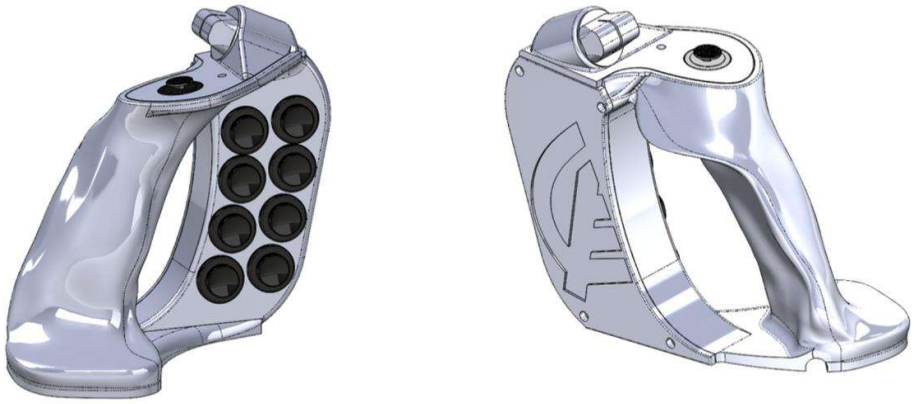


Figure 24: Model of handgrip MD10.

The handgrip is made of three parts, one large handgrip, one thumb panel and a cover plate. The cover plates is separated from the rest of the grip for ease of assembly and making revisions easier if future modifications are needed for the thumb panel.

6.3.3 PCB enclosure

The enclosure provides a safer environment for the electronics and is equipped with mounting holes for cables, holes for the indication LED on the PCB and access to the reset button via a hole in the cover of the enclosure. To mount the PCB enclosure "mounting feet" to screw it to a suitable panel inside the machine. The enclosure can be seen in Figure 25.

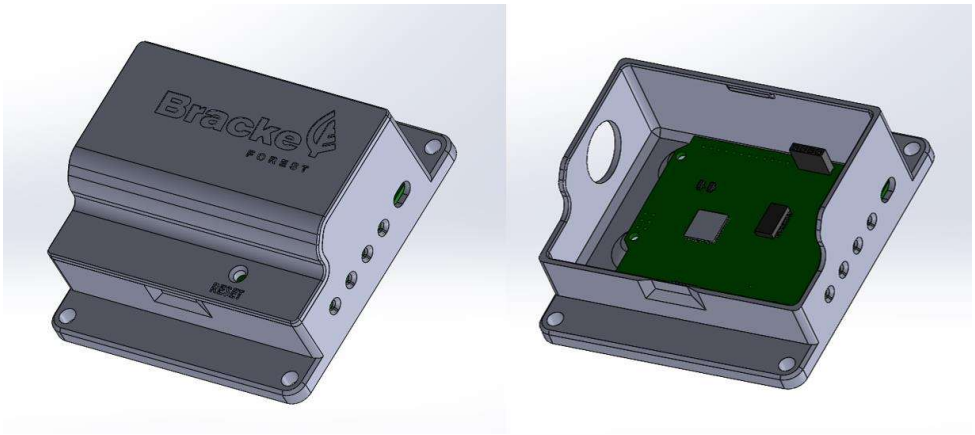


Figure 25: Model of PCB enclosure.

6.3.4 Mounting solution

The mounting solution is made up of a round metal plate with holes and a welded nut. The mounting solution can then be screwed into the handgrips and then mounted on a threaded rod. The mounting solution can be seen in Figure 26.

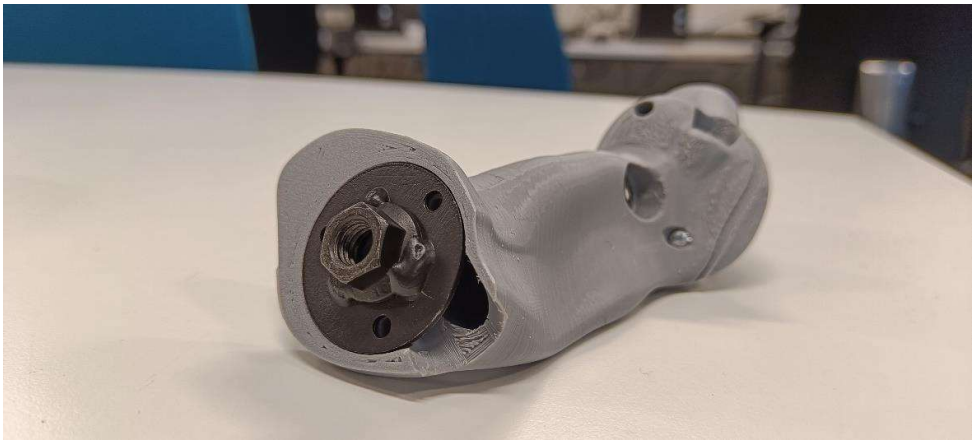


Figure 26: Mounting solution in HG4 handgrip.

7 Discussion

7.1 Analysis and discussion of results and goals

The overall goal of the project is deemed fulfilled. The group was able to create a new control grip with belonging PCB and software, based on the technical requirements.

All the functional requirements are fulfilled. The non-functional requirement is partially fulfilled, no real cable support was developed for the handgrips.

One of the optional desired functions was audio in/out via a 3.5mm jack, which was not fulfilled in the project. Limited by memory, and the microcontrollers who had the required memory did not support XInput protocol. In agreement with the assignment provider further research of the audio possibilities was stopped.

7.2 Project method

The iterative design process that was used was suitable for the project. Since Bracke Forest AB already had a well-defined project and our group were able to easily start the development process. The iterative design process was a great method of working on this type of project. The group made quick progress from the start, and dividing the project into analytical, creative and development phases was a good way to distribute the time of the project.

The group have continuously worked together in an assigned workspace during the project, where we have helped each other across the disciplines. Standardized weekly meetings have been conducted to address problems, make sure that every group member was on the same page and update the work and our timetable for the project. Working closely together was great for the productivity of the group.

7.3 XInput vs other protocols

The customer was very keen to use XInput as the protocol as it is, according to them, an easy protocol with high safety and efficiency.

The XInput API is easy to use, yet the library that is used to run XInput on Arduino or Teensy was created by Dave Madison and it is a re-implementation of Microsoft's XInput. Therefore, it lacks extensive documentation and troubleshooting resources as Windows provides, which make it harder to find solutions for unexpected bugs or performance issues.

There are other approaches to configure our input setting. One other protocol that would make certain things easier is to use DirectInput which supports a wider range of input devices with fewer limitations on the number of buttons

and axes. Additionally, DirectInput supports more advanced force feedback options which are ideal for nuanced haptic feedback for our controller [19].

Besides, one of the main issues with the XInput is that there is no way to force indexes of controllers. This could be easily set in DirectInput since DirectInput supplies a globally unique identifier (GUID) for each DirectInput device while XInput lacks built-in support for GUID [20]. To use a unique identifier or differentiate multiple XInput-devices, we can refer to some suggestions from ChatGPT 4o:

1. To combine XInput with DirectInput for identification: DirectInput can be utilized to identify all the XInput-compatible devices and then use XInput to manage inputs. Although we can manage GUIDs alongside XInput, it is complicated to maintain to use both APIs.

2. Track devices index or port number: By using indexes (0 to 3) that XInput provides, such as XInputGetState(0) or XInputGetState(1) and so on for each controller, these indexes can be stored in a configuration file or database with user specific settings. If controllers are unplugged or plugged into different ports, it may change the physical connection order.

3. Using HID API for detailed identification: Detailed information such as device path, serial numbers, and manufacturer IDs that HID API provides can be used to detect controllers and assign identifiers. This method is also quite complicated without familiarity with the HID API but it is very robust and useful especially for cross-platform compatibility.

As of now we are still searching for a viable fix for the index issue and our best way is to ensure in what order the controllers are connected.

We had a solution that could have worked but as time ran out were unable to make progress on it. The XInput enables checking the battery status and sends a command to the connected computer, indicating one of four states: high, medium, low or none. We were considering tracking the type of controller based on the battery state hardcoded in the Arduino software. This could give us a chance to identify the controller. As in theory the index is not a major concern for us, and we could hardcode an index value for the database according to the controller's battery state.

This approach would mean that there is no specific order for plugging in controllers, nor any restriction on the number or type of controllers connected, offering the customer a more seamless integration on multiple controller usage at the same time.

Another approach that was considered was to use different startup sequences for different controllers, this would also give us a possibility to identify the controllers. We would be able to track the amount of time when a controller is connected to the windows and then sends a signal to the PC when the controller is ready. The time between ready and plugged in could in theory be tracked. If this would work then we could also skip the forced index by the XInput protocol, as again we are not really that interested in the forced index, we mostly just wish to be able to identify them differently so we can have them separated in the database and for constant tracking for the remapping features.

7.4 Database

As the database choice was already made by the customer to run PostgreSQL with a preconfigured structure, see

SQL

```sql
CREATE TABLE xinput_list (
    index_ INTEGER PRIMARY KEY CHECK (index_ BETWEEN 0 AND 4),
    state_ JSONB,
    command_ JSONB,
    timestamp_ TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
);

CREATE OR REPLACE FUNCTION update_timestamp()
RETURNS TRIGGER AS $$
BEGIN
    NEW.timestamp_ = CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_timestamp_trigger
BEFORE UPDATE OF state_, command_ ON xinput_list
FOR EACH ROW EXECUTE FUNCTION update_timestamp();
```

Figure 27: Database structure

The developer's assignment for the database was more to get the current structure to work with the remapping program created in chapter 5.1.5.

As to what the customer wants, it feels as if they are set on certain approaches as it is what they used to work with, but considering a NoSQL real-time database could be a valid approach. A real-time database is designed to handle numerous changes in a short period of time, which might significantly improve response time [21].

The advantages of using a SQL database such as PostgreSQL, however, is that it minimizes undetectable errors. This is because SQL databases require a strict table structure that must be followed to function. Furthermore, PostgreSQL provides detailed error logs and tools for diagnosing issues at the database level which also helps developers to catch problems [22].

The conclusion is that if the database is operating too slowly to read all the information, perhaps changing to a real-time database will be appliable to solve this issue.

7.5 Programming languages

The choice of language for Arduino is always going to be C++ as explained in chapter 4.5. Since C++ is the predominant language in use in the computer engineering program at Mid Sweden university, we have prior knowledge of its use and faced only few difficulties. C++ for Arduino IDE is designed for easy usability and is less advanced than what we typically cover in our program, but it is a fitting challenge for beginners in C++.

There were different languages considered when discussing how to create the remapping program, taken into consideration the program was supposed to run as a background application and have a very simple GUI. The customer also had a request to preferably use C# or Python.

Python was a contender for language on the application as there are a lot of libraries for background functionally as well as GUI libraries. Similarly, C++ offers a range of libraries that support backend processes and GUI creation, making it another strong candidate for application development. C++ is the language most used by the developers in their courses.

Another language we were considering was C++, as mentioned above the developers' most used language, but not very suitable for a simple GUI application that is meant to run in the background. As there are a limited number of GUI libraries for simple applications.

As a result, we decided to use C#. Even though the developers of the group have had little to none experience prior with C# it is particularly good for Windows-focused applications. There are plenty of well-defined libraries for

the GUI as well as reading inputs from our XInput controller using the SharpDX library.

7.6 PCB design

Since no high-frequency components are used, the PCB design doesn't need advanced techniques. This also reduces the risk of parts of the PCB acting like a capacitor or antenna. Using a 24-wire cable to connect the PCB to the joystick could add extra installation time. One solution is to divide the PCB design into multiple parts, with a main PCB and a smaller daughter PCB inside the joystick, and by using a protocol like I2C, reduce the number of required wires.

7.7 Handgrip design

As seen in chapter 6.3.1 two differing approaches were developed for the handgrip. This choice was made to capture different sides of possible handgrip designs. The choice of what handgrip was most suitable for forestry machines could not be evaluated within the scope of the project. However, these two approaches give Bracke Forest AB two very different solutions to evaluate for themselves.

8 Conclusions

8.1 Future work for Bracke Forest AB

Bracke Forest AB should use this work as a foundation for further development. The following chapters states certain areas that the group thinks are most suitable for further development.

8.1.1 Refining of CAD

The current CAD models of the handgrips are roughly modelled and need further work before production. The models should be optimized based on the chosen production method, like 3D-pringting or injection moulding. Right now, the handgrips are 3D-printable, but the models could be refined further.

8.1.2 Adjustability

If the group had more modelling time, the next step would be to include adjustability into the grip. If the user can adjust the handrest with ease it would fit a way bigger population of hand sizes.

8.1.3 Further user testing

More user testing should be conducted to ensure the products' features. These user tests should be done in a more life-like environment or in an actual forestry machine with real operators. This is deemed needed since it will give a better understanding of how our product will be used in its application.

8.1.4 Remapping application

Further work that should be done with the remapping / background application is further GUI work.

Testing of software in a real environment while running the application has not been tested or researched.

8.1.5 Database

The database should be tested on real hardware that will be used in the future to ensure compatibility. As of now it has only been tested on the developer's computer.

This project has not researched the connection between the database and the PLC which controls the machinery. This connection is vital for the useability of the newly developed handgrip and should be a part of the future work.

8.1.6 Bus extender location

A bus extender in the handgrip can open up more functions, like adding indicator LEDs, motors, or additional inputs. By using an extra bus extender, we gain more available digital input and output pins, allowing for the development of additional features. Placing the bus extender inside the handgrip also reduces the number of wires and the size of cables, which leads to more functionality and lower costs.

9 References

[1] "Bracke Forest company," Bracke Forest AB, [Online]. Available: https://www.brackeforest.com/about-us/bracke-forest-company. [Accessed 30 September 2024].

[2] "Swedish forestry through the ages," Swedish Forest Industries, 23 August 2022. [Online]. Available: https://www.forestindustries.se/forest-industry/forest-industry-significance/swedish-forestry-through-the-ages. [Accessed 30 September 2024].

[3] A. C. G. O. a. S. P. Joseph Allott, "Data: The next wave in forestry productivity," McKinsey & Company, 27 October 2020. [Online]. Available: https://www.mckinsey.com/industries/packaging-and-paper/our-insights/data-the-next-wave-in-forestry-productivity. [Accessed 30 September 2024].

[4] Bracke Forest AB, "Brackeforest.com," [Online]. Available: https://www.brackeforest.com/sv/produkter. [Accessed 22 10 2024].

[5] "About GitHub and Git," GitHub, [Online]. Available: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git. [Accessed 25 September 2024].

[6] "Getting Started With XInput in Windows applications," Microsoft, 23 October 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/xinput/getting-started-with-xinput. [Accessed 25 September 2024].

[7] "XInput," Arduino, [Online]. Available: https://www.arduino.cc/reference/en/libraries/xinput/. [Accessed 25 September 2024].

[8] D. Madison, "Supported Boards," GitHub, [Online]. Available: https://github.com/dmadison/ArduinoXInput/blob/master/extras/SupportedBoards.md.

[9] "What is Visual Studio?," Microsoft, 19 June 2024. [Online]. Available: https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022. [Accessed 14 October 2024].

[10] D. Madison, "Arduino XInput Library," github, [Online]. Available: https://github.com/dmadison/ArduinoXInput. [Accessed 22 October 2024].

[11] DFRobot, "DFRobot_MCP23017," GitHub, [Online]. Available: https://github.com/DFRobot/DFRobot_MCP23017. [Accessed 30 October 2024].

[12] D. Madison, "Xbox 360 Controller LEDs Library," GitHub, [Online]. Available: https://github.com/dmadison/Xbox360ControllerLEDs. [Accessed 29 October 2024].

[13] "SharpDX," nuget, 24 August 2018. [Online]. Available: https://www.nuget.org/packages/SharpDX. [Accessed 22 10 2024].

[14] "Hardcodet.Wpf.TaskbarNotification," nuget, 23 November 2013. [Online]. Available: https://www.nuget.org/packages/hardcodet.wpf.taskbarnotification. [Accessed 22 October 2024].

[15] “Npgsql - .NET Access to PostgreSQL,” The Npgsql Development Team, [Online]. Available: https://www.npgsql.org/. [Accessed 22 October 2024].

[16] “System.Windows.Threading Namespace,” Microsoft, [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.windows.threading?view=windowsdesktop-8.0. [Accessed 30 October 2024].

[17] Atmel Corporation, “ATmega16U4/ATmega32U4,” 2014. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Summary.pdf. [Accessed 29 October 2024].

[18] Microchip Technology Inc., “MCP23017/MCP23S17,” [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/MCP23017-Data-Sheet-DS20001952.pdf. [Accessed 29 October 2024].

[19] “Introduction to DirectInput,” Microsoft, 9 October 2011. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ee418273(v=vs.85). [Accessed 28 October 2024].

[20] “Enumerating Microsoft DirectInput Devices,” Microsoft, 11 June 2009. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/bb153253(v=vs.85). [Accessed 28 October 2024].

[21] K. Babitz, “Datacamp,” Data Science writer | Senior Technical Marketing Analyst at Wayfair | MSE in Data Science at University of Pennsylvania, [Online]. Available: https://www.datacamp.com/blog/sql-vs-nosql-databases. [Accessed 28 10 2024].

[22] “About,” PostgreSQL, [Online]. Available: https://www.postgresql.org/about/. [Accessed 28 October 2024].

[23] [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.windows.threading?view=windowsdesktop-8.0.

10 Appendix A: Source code

https://github.com/Nize2200/Br-cke-sims

11 Appendix B: Requirements

Functional requirements are:

- Use Xinput API
- Cable using 24-pole Binder-connector (male) at the control device
- I/O box using 24-pole Binder-connector (female)
- Support the following I/O (according to Xinput) through Binder connector:
    - 14 Digital Inputs (buttons)
    - 2 Analog inputs (0 to 255) (triggers)
    - 4 Analog inputs (-32768 to 32767) (Joysticks)
    - 2 Analog outputs (0 to 65535) (haptic feedback motors)
    - Feed voltage (USB-standard)
    - Ground
- At least 1 analog function on the control device
- At least the same number of functions as the current control devices
- Remapping buttons using a basic interface

Non-functional requirements are:

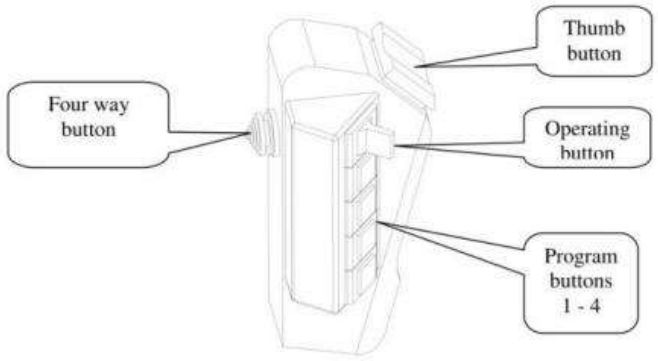
- Cable strain relief
- Haptic feedback
- Able to be manufactured in batches of 10-20 pieces
- The control device should be easily attachable

Wishes:

Audio in/out to the I/O box (3,5 mm)

12 Appendix C: Operating control

Operating control



Operating controls are either left or right handed. Button functions are as follows:

Operating button Three positions, neutral in mid position, scarification and lift to either side.

Program button 1 Switching between *Operating Profiles 1 and 2* in scarification mode and *Separate lift* is achieved by pressing the program button 1 while at the same time moving the four way button briefly in the desired direction.

Program button 2 Switching between *Operating Profiles 3 and 4* in scarification mode and *Separate lift* is disachieved by pressing the program button 2 while at the same time moving four way button briefly in the desired direction.

Program button 3 Switching between *Operating Profiles 5 and 6* in scarification mode.

Program button 4 Switching between *Operating Profiles 7 and 8* in scarification mode.

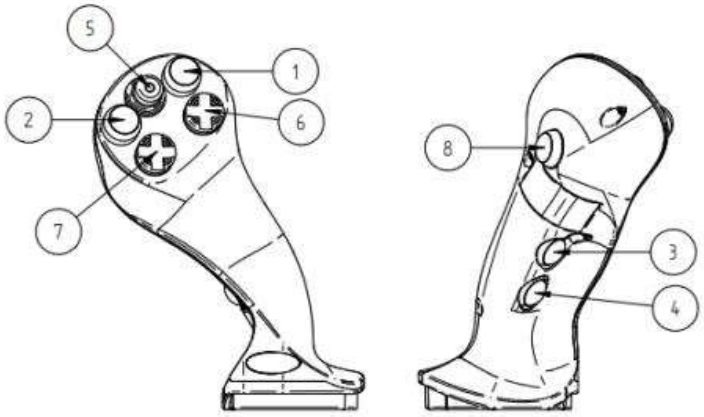
Thumb button Arms can be retracted by moving the thumb button toward the side to be retracted. Both arms can be retracted simultaneously by pressing the middle of the button.

Four way button Temporarily operating program (TOP), for one or more arms.
In scarification mode TOP is activated by the four way button, as described below:

Left	=	Left arm TOP
Upward	=	Middle arm TOP
Right	=	Right arm TOP
Downward	=	All arms TOP

13 Appendix D: Operating control

Manöverreglage



1. Växling mellan körprogram 1 och 2 i markberedningsläge och Temporär Tallriksrotation om knappen hålls in mer än en halv sekund.
2. Växling mellan körprogram 3 och 4 i markberedningsläge och Temporär Lättningstryck + Tallriksrotation om knappen hålls in mer än en halv sekund.
3. Växling mellan körprogram 5 och 6 i markberedningsläge och Temporär Tallriksvinkel + Lättningstryck + Tallriksrotation om knappen hålls in mer än en halv sekund.
4. Växling mellan körprogram 7 och 8 i markberednings läge och Temporär Lyft om knappen hålls in mer än en halv sekund. I Lyftläge aktiveras Risrensningsfunktionen så länge knappen hålls inne.
5. Körknapp, återfjädrande. Ett tryck nedåt växlar till markberedning, ett tryck uppåt växlar till lyft. Dubbeltryck inom två sekunder avaktiverar säkerhetsstopp. När maskinen är i markberedningsläge kan man med ett tryck uppåt växla till lyft och med ett tryck nedåt växla till neutral. När maskinen är i lyftläge kan man med ett tryck uppåt växla till neutral och med ett tryck nedåt växla till markberedningsläge.
6. Fyrlägesknapp justermeny. Används för att navigera i justermenyn och för att ändra värden. Uppåt ökar inställningsvärde, nedåt minskar. Används även för att välja om båda tallrikarna skall vändas åt höger eller vänster.
7. Fyrlägesknapp bredd. Höger eller Vänster arm breddas in under markberedning med höger eller vänster tryckning. Kort tryckning för när armarna skall breddas in med maskinlängds fördröjning, nästa korta tryckning för när armarna skall breddas ut med maskinlängds fördröjning. Tryckning längre än 1 sekund drar in armarna direkt, armarna går ut när knappen släpps. I lyftläge med tallrikarna utåt: Tallriksvinkling körs in/ut med upp/ned, Armar in/ut körs med vänster/höger.
8. OK-knapp. Används för att komma in i justermenyn och för att bekräfta ändringar i den.

14 Appendix E: User testing sheet

Testformulär prototyp 2 | Knappar Testperson _____

Rangordna greppen nedan

Lägst betyg			Högst betyg

Positivt?

1	
2	
3	
4	

Negativt?

1	
2	
3	
4	

Övriga kommentarer:

15 Appendix F: User tests

User test 1

Grip	Total points	Positive comments	Negative comments	Our reflection based on responses
1.	22	Good support and a good angle for the wrist, the small size give a firm grip	uncomfortable grooves for the fingers	A smaller grip appears to be desired. Aswell as a grip with the palm facing downwards appears to give the wrist sufficent support and angle.
2.	32	Stable and neutral grip with the wrist in a pleasant angle	Not enough space for the fingers	A thin grip appears to provide a firm grip. Positive response was given for the angle of the wrist.
3.	15	Plenty of space	Too big and hard to grip	A grip with a large diameter appears difficult to grasp.
4.	21	Comfortable and thumb in good position. Neutral grip	Not enough stability "outwards", too large and the fingers don't fit the grooves.	Large diameter grip appears hard to grasp, it is however comfortable. The angle of this grip provides a neutral position for the wrist.

Conclusion:

An upright grip is better for stability, preferably with a bit more angle than the tests in this iteration. Keep in mind that hand sizes vary and don't keep the grooves for the fingers.

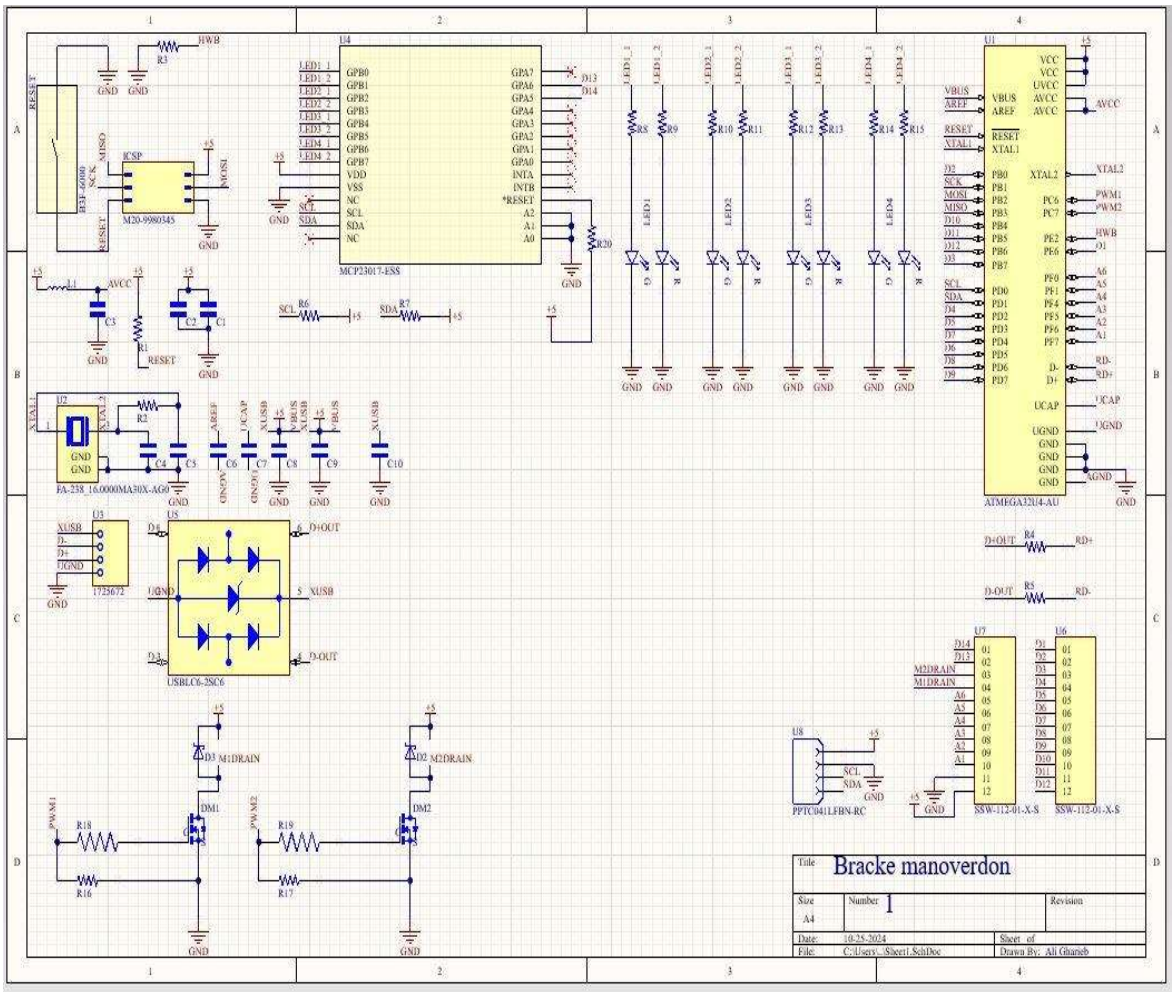
User test 2

Grip	Total points	Positive comments	Negative comments	Our reflection based on responses
1.	17	Steady and neutral grip, most buttons are placed well.	Grip too thin. Many users could not reach most buttons.	The angle and size of this grip appears to fit most hands. The buttons placed on the "blade" however appears hard to reach for most users.
2.	7	The large grip is comfortable.	Hard to press the buttons of the back sida of the grip.	The large size seems comfortable for most users, Most users did not like the buttons on the backside of the grip.
3.	11	Relaxed and natural grip.	The button placement at the thumb is not good.	This grip had poor button placement and the thumb had bad placement
4.	15	Good grip and button placement.	Buttons furthest from the thumb is hard to reach, Trigger button is not good for the grip.	The placement of buttons were good and the grip was well formed. Some buttons were hard to reach and the trigger did not provide a firm grip.

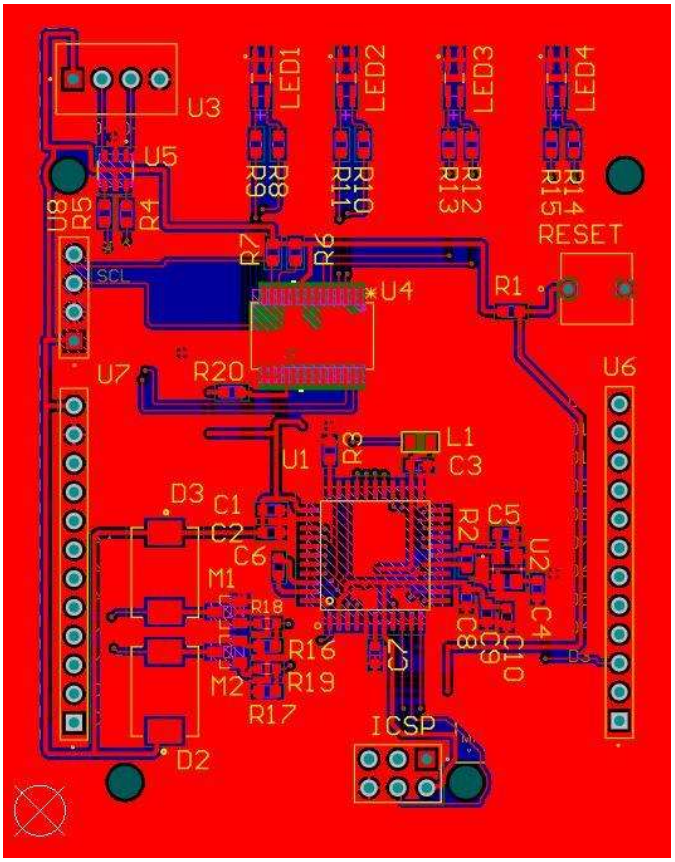
Conclusion:

The grips while being similar to eachother gave mixed impressions. The grip appears comfortable on all models. Button placement is not desired on the back side of the grip. Model no. 4 had very positive response on button placement and model no. 1 had very positive response on grip.

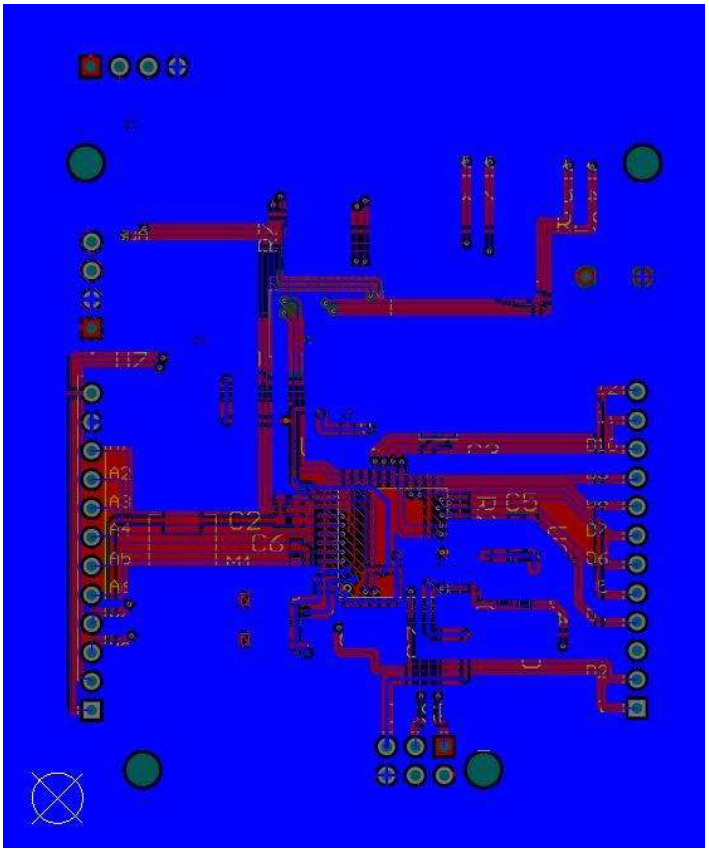
16 Appendix G: Schematic



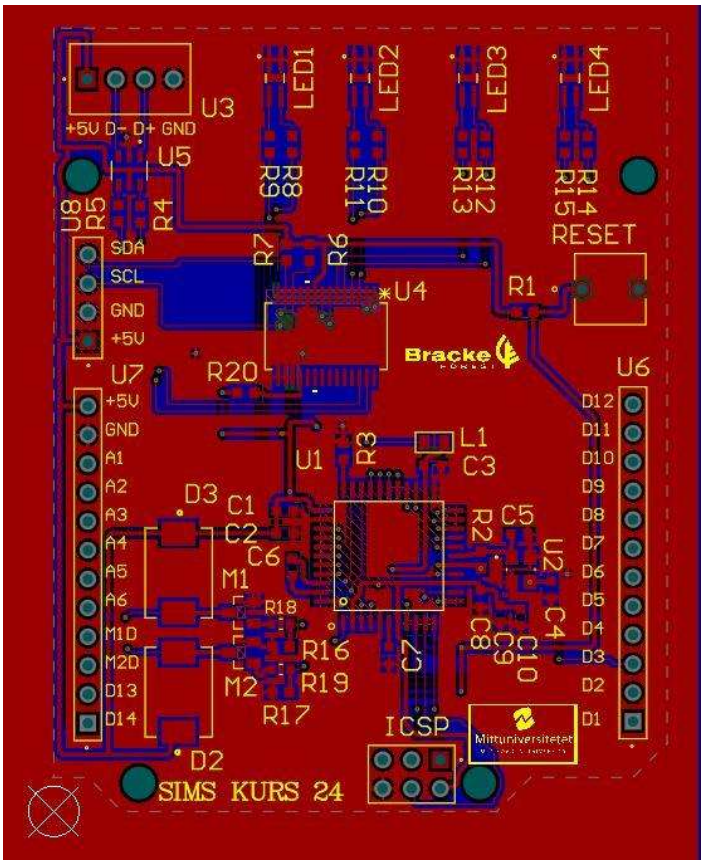
17 Appendix H: Top layer



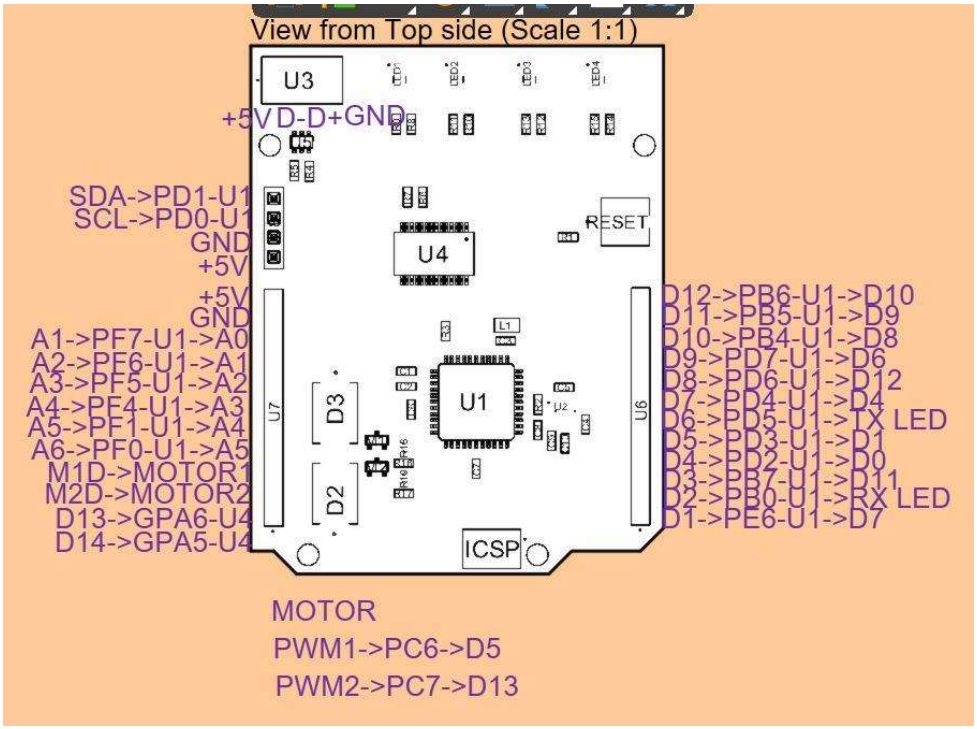
18 Appendix I: Bottom layer



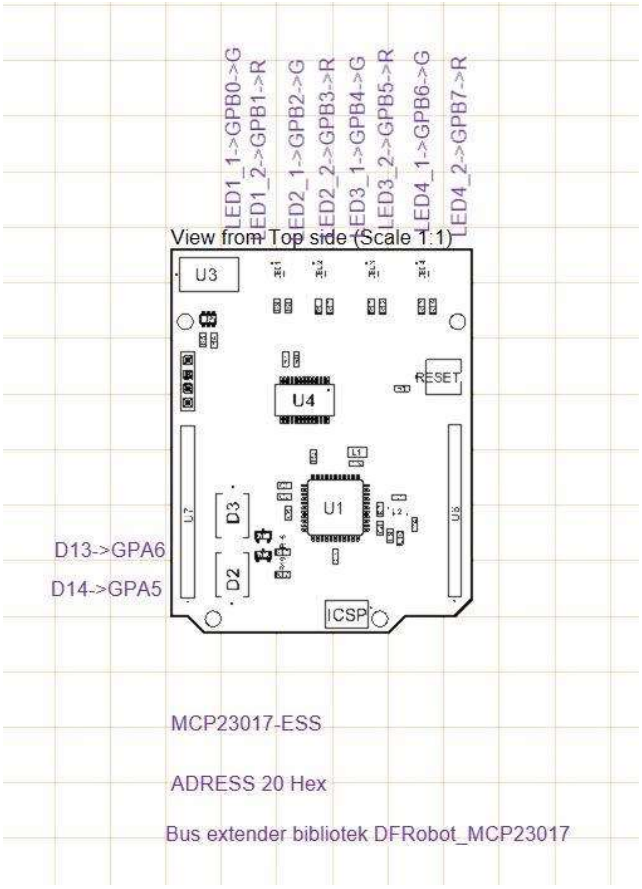
19 Appendix J: Top overlay



20 Appendix K: Pinout



21 Appendix L: Pinout MCP23017-ESS



22Appendix M: Pinout 24-Binder

1->+5V

2->GND

3->A1

4->A2

5->A3

6->A4

7->A5

8->A6

9->M1D

10->M2D

11->D13

12->D14

13->D12

14->D11

15->D10

16->D9

17->D8

18->D7

19->D6

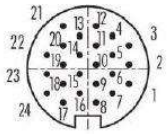
20->D5

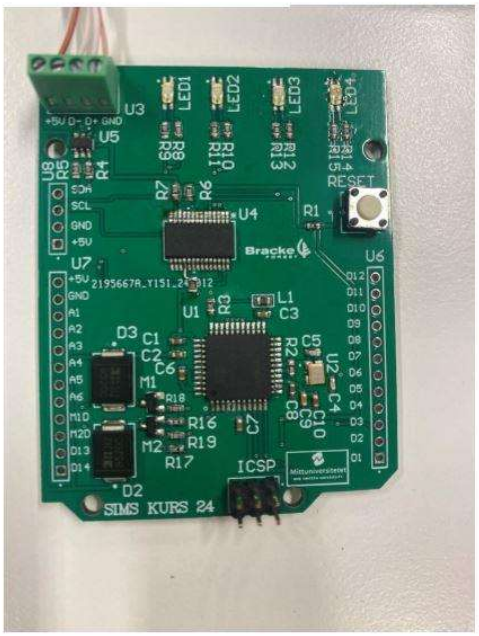
21->D4

22->D3

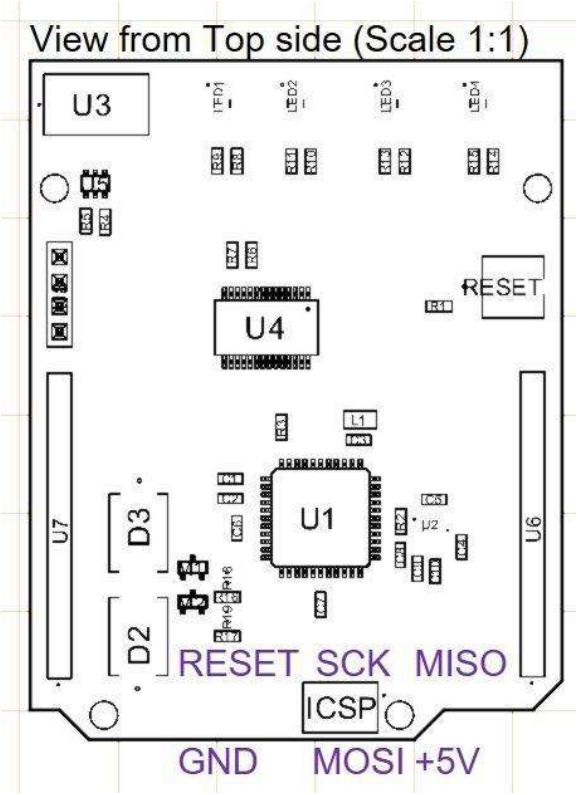
23->D2

24->D1





23 Appendix N: First-time programming



24Appendix M: PCB-Costs

PCB-Kostnader:

Komponentlista:

1x_ATMEGA32U4-AU-ND 54,12 SEK.
IC MCU 8BIT 32KB FLASH 44TQFP
1x_277–1275-ND 27,31 SEK.
TERM BLK 4P SIDE ENT 2.54MM PCB
2x_311-22.0HRCT-ND 2,04 SEK.
RES 22 OHM 1% 1/10W 0603
6x_311-10.0KHRCT-ND 6,12 SEK.
RES 10K OHM 1% 1/10W 0603
2x_311-20GRCT-ND 2,04 SEK.
RES 20 OHM 5% 1/10W 0603
2x_353-SI2312A-TPCT-ND 9,20 SEK.
N-CHANNEL MOSFET, SOT-23
2x_31-B520CQ-13-FCT-ND 15,54 SEK.
DIODE SCHOTTKY 20V 5A SMC
1x_SER4205CT-ND 5,73 SEK.
CRYSTAL 16.0000MHZ 7PF SMD
1x_311-1.00MHRCT-ND 1,02 SEK.
RES 1M OHM 1% 1/10W 0603
2x_720-VJ0603A220KXQPW1BCCT-ND 2,04 SEK.
CAP CER 22PF 10V C0G/NP0 0603
5x_720-VJ0603V104ZXQPW1BCCT-ND 8,20 SEK.
CAP CER 0.1UF 10V Y5V 0603
2x_1276-1946-1-ND 2,04 SEK.
CAP CER 1UF 10V X7R 0603
1x_490-10475-1-ND 1,33 SEK.
CAP CER 10UF 10V X5R 0603
1x_497-5235-1-ND 3,68 SEK.

TVS DIODE 5.25VWM 17VC SOT23-6

1x_SW263CT-ND 4,81 SEK.

SWITCH TACTILE SPST-NO 0.05A 24V

1x_952-2120-ND 3,68 SEK.

CONN HEADER VERT 6POS 2.54MM

1x_MH2029-300YCT-ND 1,02 SEK.

FERRITE BEAD 30 OHM 0805 1LN

1x_MCP23017-E/SS-ND 17,29 SEK.

IC XPNDR 1.7MHZ I2C 28SSOP

4x_1497-1303-1-ND 39,68 SEK.

LED GREEN/RED CLEAR 4SMD

8x_311-150HRCT-ND 8,16 SEK.

RES 150 OHM 1% 1/10W 0603

PCB-Tillverkningskostnader: $2/5pcs.

PCB-Total Kostnader: 229.68 SEK.