Programmable calculator

An open system design using Raspberry Pi: A case study

Degree project report in Electrical Engineering

Vincent Sinclair

Department of Electrical Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Degree project 2025

Programmable calculator

An open system design using Raspberry Pi: A case study

Vincent Sinclair

CHALMERS

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg 2025

Programmable calculator
An open system design using Raspberry Pi: A case study
Vincent Sinclair

Advisor: Alessio Cicero, Department of Computer Science and Engineering, Chalmers University of Technology
Examinor: Nicholas Smallbone, Department of Computer Science and Engineering, Chalmers University of Technology

Degree project report 2025
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg 2025

Programmmable calculator
An open system design using Raspberry Pi: A case study
Vincent Sinclair
Department of Electrical Engineering
Chalmers University Technology

Summary

This report describes the development of a physical programmable calculator using a Raspberry Pi as an open alternative to traditional closed calculators. The project addresses the limitations of traditional calculators by creating a modular and extensible alternative. This system allows for both software customization and hardware expansion. The implementation consists of a single board computer (Raspberry Pi 3B), a 4x4 matrix keypad, an I2C LCD, and a custom designed 3D printed case. The calculator is written in the Java programming language using the Pi4J library. This results in a modular design centered around a CalculatorState class that enables easy addition of new operations.

The resulting calculator performs basic arithmetic operations comparable to a simple traditional calculator while additionally offering significant extensibility. Adding new functionality requires minimal code, as demonstrated through the implementation of a square operation which required four simple steps. However, the tradeoffs include higher power consumption and a larger form factor in the use of a Raspberry Pi.

The project validates the feasibility of creating an open source calculator alternative using readily available components. This provides complete user control over both hardware and software while maintaining functionality comparable to traditional calculators.

Key words: Raspberry Pi, calculator, Java, Pi4J, modular, extensible, I2C, 3D printing, hardware, software.

Foreword

This project was inspired by my desire to gain experience with the Java programming language and to explore multiple technical fields.

I would like to express my sincerest gratitude to my advisor, Alessio Cicero, for his invaluable support throughout the course of this project. His feedback on writing, assistance with design challenges and help with project planning have been instrumental in successfully completing this project. I also wish to thank my examiner, Nicholas Smallbone, for providing the opportunity to undertake this degree project.

Vincent Sinclair, Gothenburg, May 2025

Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

API	Application Programming Interface
GPIO	General Purpose Input/Output
LCD	Liquid Crystal Display

Nomenclature

Below are the key hardware and software components referenced throughout this thesis.

0.1 Hardware Components

I2C	Inter-Integrated Circuit protocol, used to communicate with the LCD display.
LCD	Liquid Crystal Display, used to show the calculator's output.

0.2 Software Components

AddOperation	Implements the addition operation for the calculator.
Calculator	The main class that initializes and runs the calculator, coordinating inputs, operations, and outputs.
CalculatorState	Tracks the calculator's current state, such as the current sum, the input field value, and the last used operation.
DigitOperation	Updates the display field with the digit defined during initialization.
DivideOperation	Implements the division operation for the calculator.
EvalOperation	Executes the last set mathematical operation.
Keypad4x4	Implements mapping operations to the keypad executing them on key press
MultiplyOperation	Implements the multiplication operation for the calculator.
Operation	Abstract class defining the structure for calculator operations, stretching the definition to include both mathematical and functional operations.
Pi4J	Java library for controlling Raspberry Pi hardware.
SubtractOperation	Implements the subtraction operation for the calculator.
QapassLcd	Implements the updating of the display through the I2C protocol.

Contents

List of Figures

1

Introduction

1.1 Background

Calculators are traditionally closed systems, offering no access to hardware or source code for their users. This choice of design limits them to predefined functions, such as basic arithmetic, and prevents customization. Consequently, innovation, accessibility, and adaptability to user needs are hindered. Furthermore, the lifespan of a traditional calculator's software is deeply connected to its hardware. This means that when the hardware becomes obsolete or fails, the software becomes unusable even if it remains functional. By separating the software from the hardware, the software's lifespan can be extended, allowing it to evolve with its users and the times.

1.2 Purpose

This thesis project aims to address the aforementioned issues through the development of a programmable calculator using a Raspberry Pi. The calculator will offer modifiable source code and expandable hardware via GPIO pins, allowing users to modify it to suit their needs. Additionally, it will serve as a case study into developing such an open system. This project spanning ten weeks will be documented and result in a functional programmable calculator.

1.3 Goals

- Develop a functional programmable calculator using a Raspberry Pi, capable of performing basic arithmetic operations (addition, subtraction, multiplication, division).
- Provide modifiable source code, written in Java, to enable user customization of the calculator's functionality.
- Incorporate expandable hardware through the Raspberry Pi's GPIO pins, allowing users to add new components.
- Document the development process as a case study to evaluate the feasibility of creating an open-source calculator.

1.4 Delimitations

- The project will focus on basic arithmetic operations (addition, subtraction, multiplication, division) and exclude more advanced functions like graphing due to time constraints.
- The calculator will use a character based I2C LCD rather than a graphical display to simplify implementation. The reason being that it is part of the pool of components on hand.
- The case serving as the housing for the components is intended to be simplistic, meaning that it is not composed of many parts, due to time constraints and lack of expertise.

2
Method

The workflow for developing the calculator will be an iterative process relying on trial and error. By dividing the project into distinct phases, each phase will act as a building block, progressively constructing the final product.

2.1 Setup phase

The project will begin with a setup phase in which core parts of the project will be confirmed. The idea being that fundamental obstacles can be discovered early and circumvented. The plan is to begin with a setup and trial run of the chosen Raspberry Pi model. The following are readily available and on hand:

1. Raspberry Pi Zero W
2. Raspberry Pi 1
3. Raspberry Pi 2B
4. Raspberry Pi 3B
5. Raspberry Pi 4

Through this it would be immediately determined if the Raspberry Pi model and the operating system (Raspberry Pi OS) is sufficient for the project's aims. This will be assessed by the installation of necessary software and the testing of simple applications written in the chosen programming language (Java). If obstacles surface here they can be circumvented through the swapping of Raspberry Pi models, operating systems, and even programming languages if need be.
If things proceed smoothly, focus will change to familiarization of the Pi4J API. The Pi4J API will handle GPIO related tasks, enabling communication with components on hand, the core of which are listed here:

1. I2C LCD.
2. 4x4 keypad module.

2.2 Implementation phase

The implementation phase begins after confirming that the hardware, software and toolchain are capable of meeting the project's aims. During this phase the program structure is defined through the use of a high level diagram, and the implementation of core functionality listed here:

1. Allow for writing characters to the screen through the I2C protocol using the Pi4J API.
2. Get input from the 4x4 keypad module through connections to the GPIO pins.
3. Allow for the clearing of the display.
4. Allow for basic mathematical operations to be performed (Addition, subtraction, multiplication, division).
5. Allow for a button press to evaluate the operation and display the answer.

2.3 Shell and study phase

This phase serves to prepare for the project's finalization through the design and creation of a 3D printed case to house the Raspberry Pi, I2C LCD, 4x4 keypad module, and additional components. The design will be made through the use of Blender, the reason for this is because it is a free and open source tool meaning that it is easily accessible to a large user base, resulting in more readily available learning resources. However, due to limited experience it is yet to be determined if this tool is appropriate, as such if problems surface this tool can be swapped out. After the design has been constructed an order will be placed for it to be printed and delivered.
To investigate the Raspberry Pi powered calculator's portability, this phase includes a basic assessment of its power consumption based on published specifications rather than testing. This approach is chosen for the reasons listed:

1. Isolating power drain causes between devices that operate in fundamentally different ways (Raspberry Pi vs microcontrollers) is very difficult.
2. Long term battery drain tests are unreliable and too time consuming.
3. Tools for precise measurements are not readily available.

This will result in quicker and more accurate assessments, and actually prove fruitful in answering the question on portability for the purposes of the project.

2.4 Finalization phase

This phase will integrate hardware and software, test the system with simple calculations, and finalize documentation. However, if for some reason the earlier phases proceeded more smoothly than anticipated and time allows for it, additional features listed below are implemented:

1. Decimal support.
2. Error handling.

3
Technical Background

This chapter serves to present explanations and background information on the technologies used in the project.

3.1 Raspberry Pi

Raspberry Pi [1] is a series of single board computers, originally designed for teaching purposes. However, it later deviated from that narrow purpose due to its low cost, size, and flexibility. It is now used in various fields and for various types of projects spanning from enterprise to hobbyist.

In this project it serves to run Raspberry Pi OS and through it utilize its included functionality to allow for communication with external peripherals. The Raspberry Pi model chosen for this project ended up being the Raspberry Pi 3B due to its 64 bit architecture and its compatibility with Pi4J dependencies and minimal application examples.

3.2 Raspberry Pi OS

Raspberry Pi OS [2] is a version of the operating system Debian [3], optimized for Raspberry Pi hardware. At its core it consists of the Linux kernel and the GNU software toolkit. In this project it serves to run the calculator software using the Java Virtual Machine (JVM), allowing for remote development through the SSH protocol.

3.3 Java

Java [4] is a general-purpose, high-level, memory-safe, object-oriented programming language. Java is designed to enable programmers to write software once and then run it anywhere [5]. The reason for its use in this project is due to this cross compatibility, its widespread use [6], and personal interest in the language.

3.4 Blender

Blender [7] is free and open source software, specifically tailored to 3D creation. It supports everything from pipeline modeling to animation. The main reasons for its use in this project are due to its availability and popularity. These factors contribute to the abundance of learning resources and documentation which was deemed to be critical because of limited experience in 3D modeling. Using Blender, the 3D models for the case parts were created, and those files were then sent for 3D printing.

3.5 SSH

The Secure Shell Protocol (SSH) [8] is a cryptographic network protocol used for secure remote operation of devices on a network. In this project, it is utilized by SSH clients like OpenSSH, Termius, and for deployment by the Ant build system. SSH allows for secure and convenient remote development, and access to the Raspberry Pi. Whereas, without it a physical connection would be required.

3.6 Maven

Maven [9] is a tool for building and managing a Java project. It is used to handle dependencies and compilation. Additionally, by using the maven-antrun-plugin, compilation, transfer, and execution on the external Raspberry Pi can be handled remotely.

3.7 IntelliJ Idea Community Edition

IntelliJ Idea Community Edition [10] is an integrated development environment (IDE) tailored for Java development. It was used in this project to smoothly develop the calculator application.

4

Implementation

This chapter aims to give an overview of the implementation process and the design decisions made throughout the project.

4.1 Overview

The project involved the development of a functional calculator with a Raspberry Pi serving as the processing unit. The completed Raspberry Pi calculator, shown in Fig. 4.1 consists of:

- A custom-designed case
- A 4x4 keypad for user input
- An I2C LCD for output
- A calculator application written in the Java programming language

This implementation offers functionality comparable to traditional calculators while providing the additional benefit of extensibility through modular software and hardware.

4.2 Hardware implementation

4.2.1 Raspberry Pi setup

During the setup phase, a Raspberry Pi Zero W, chosen for its size and minimal power consumption, was deemed incompatible. The reason being its inability to run a minimal Pi4J application provided by Pi4J documentation, for the purposes of verifying compatibility and introducing the API. The inability stems from a 32-bit compatibility issue with the "libgpiod" dependency in charge of handling GPIO pins. Therefore, in order to rectify this, it was replaced by a Raspberry Pi 3B for its 64-bit architecture. However, the consequence was compromising the benefits of the form factor and power consumption.

4.2.2 Peripherals

4.2.2.1 LCD

The LCD is physically connected to the Raspberry Pi via the GPIO pins, specifically those labeled SDA (Serial Data) and SCL (Serial Clock) dedicated to communication

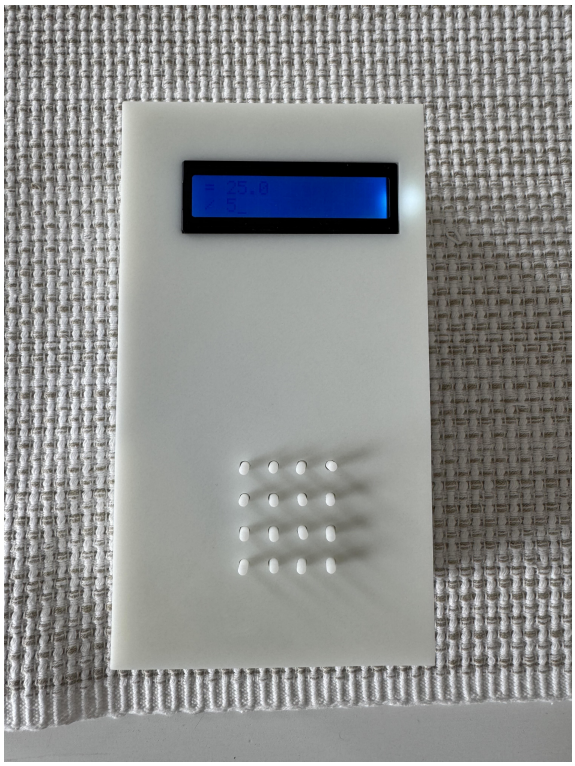

Figure 4.1: The finished and assembled Raspberry Pi calculator

via the I2C protocol. The I2C address was found using the i2cdetect command line tool. This address was then incorporated into the software implementation to enable communication.

4.2.2.2 Keypad

The keypad consists of a 4x4 matrix of switches arranged in rows and columns, labeled R1-R4 and C1-C4. The matrix consists of the circuit shown in Fig. 4.2. Button press detection can be implemented through software in which a scanning algorithm individually sets each row to high, and iterates through the columns checking for high signals. Therefore, a specific button press can be detected by using the intersection of the currently set high row and the currently observed high column.

4.2.3 Case design

The case was designed in Blender without prior 3D modeling experience, which limited the design to simple geometric shapes. Through the use of online resources and direct measurements, the design process began with the bottom portion Fig. 4.3. This part houses the battery, cables and Raspberry Pi 3B in the larger compartment, while the upper cutout houses the keypad. The blocks in the bottom serve as padding to keep the battery and Raspberry Pi from moving and to provide structural support.

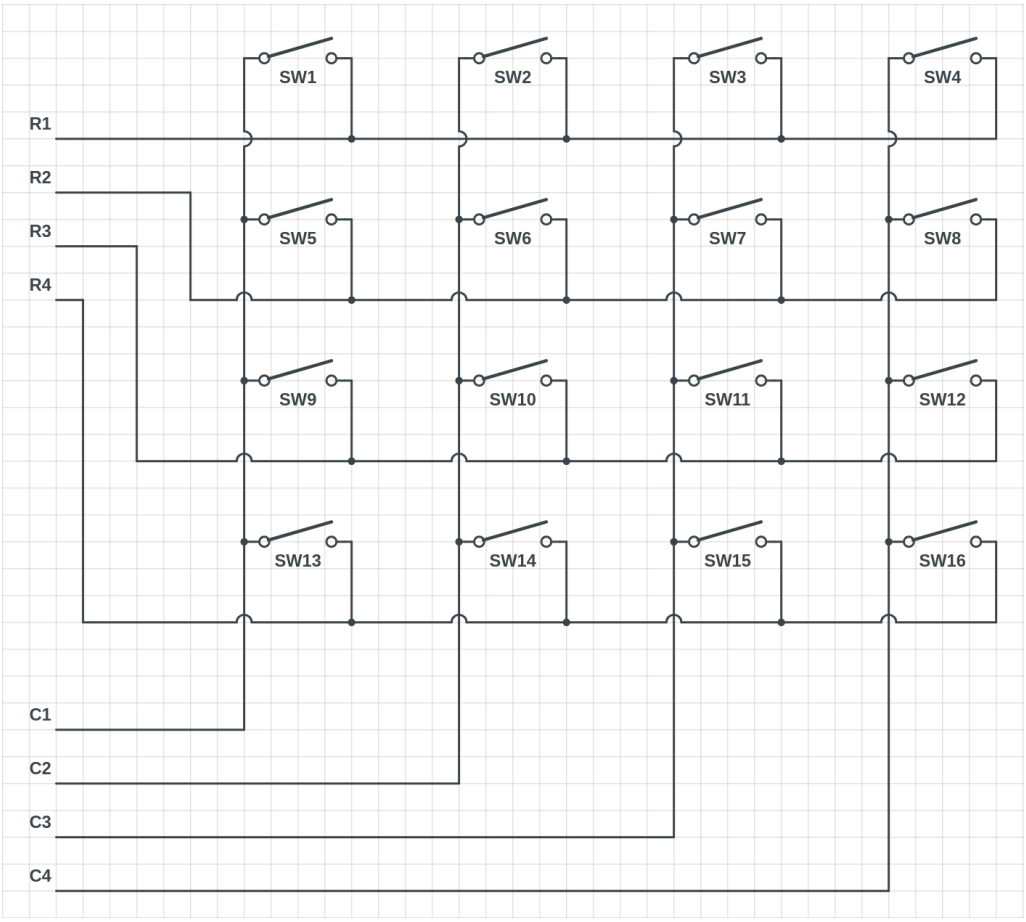

Figure 4.2: The keypad circuit

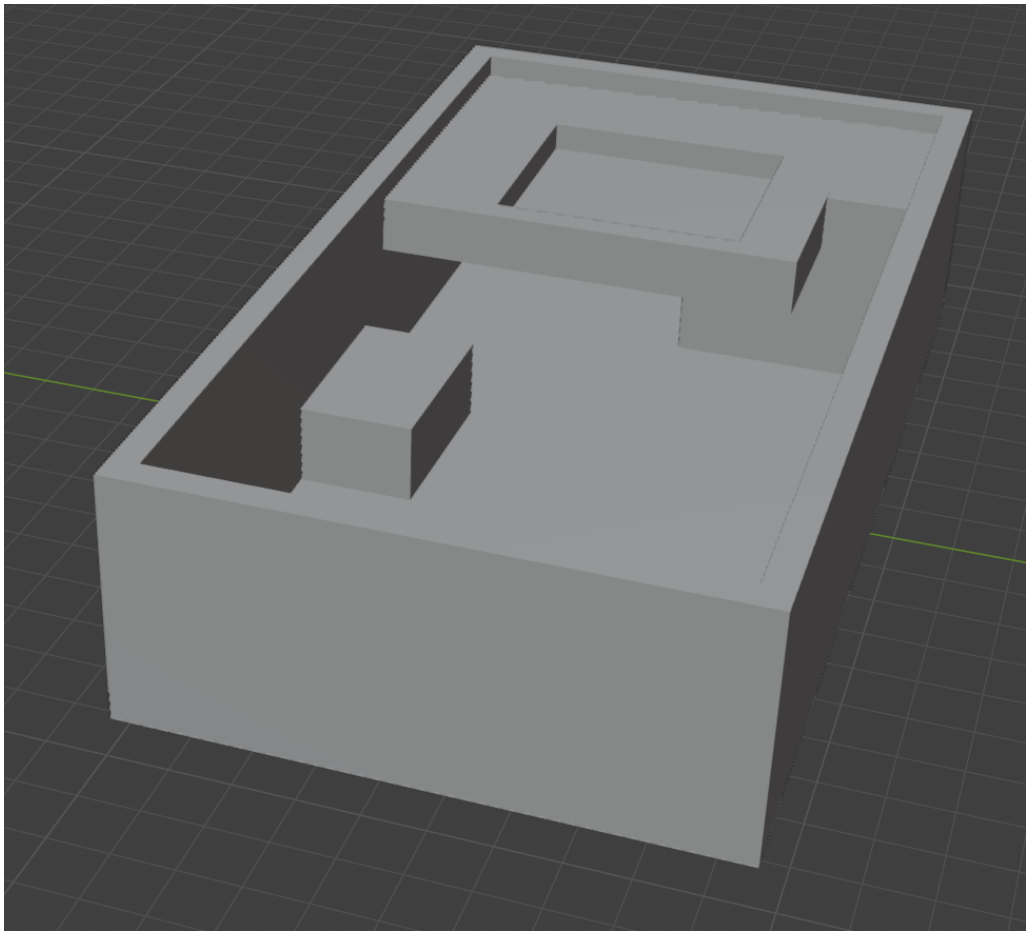

Figure 4.3: The bottom half of the case in Blender

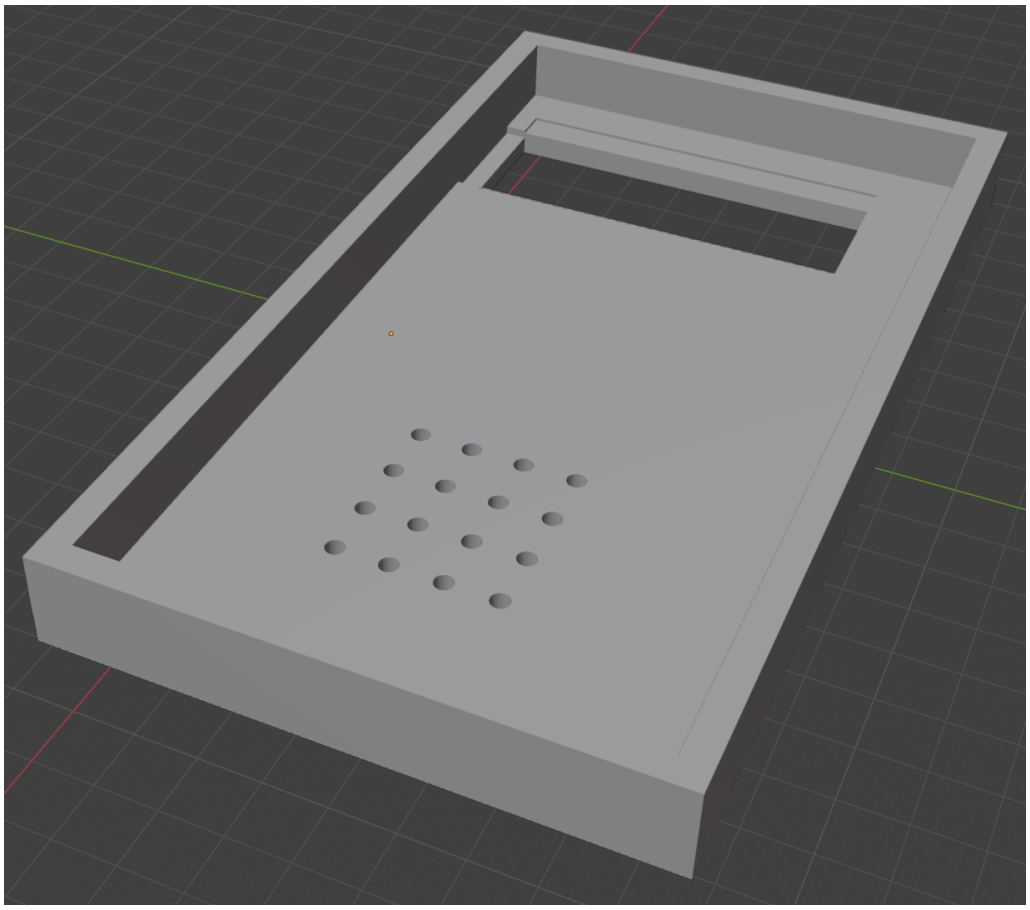

Figure 4.4: The top half of the case in Blender

The top portion was designed as a lid with cutouts for the LCD display and the button extenders Fig. 4.4. To provide access to the enclosed keypad, 16 button extenders were created Fig. 4.5.

All components were then exported to the STL file format and manufactured by PCBway, a 3D printing service provider. The final case design prioritizes functionality over looks due to design constraints and limited experience, resulting in a functional but larger than ideal case.

4.3 Software implementation

4.3.1 Development environment setup

The development environment was set up through a Pi4J-maven-archetype, which is a Maven template for quickly setting up a project. This template allows for remote development and deployment through the ant-build system. The setup involved the filling of project generation parameters and configuration ranging from IP addresses, ports, file paths and login information. Additionally, by mapping Maven commands to Intellij Idea Community Edition, the IDE of choice, the build system could be

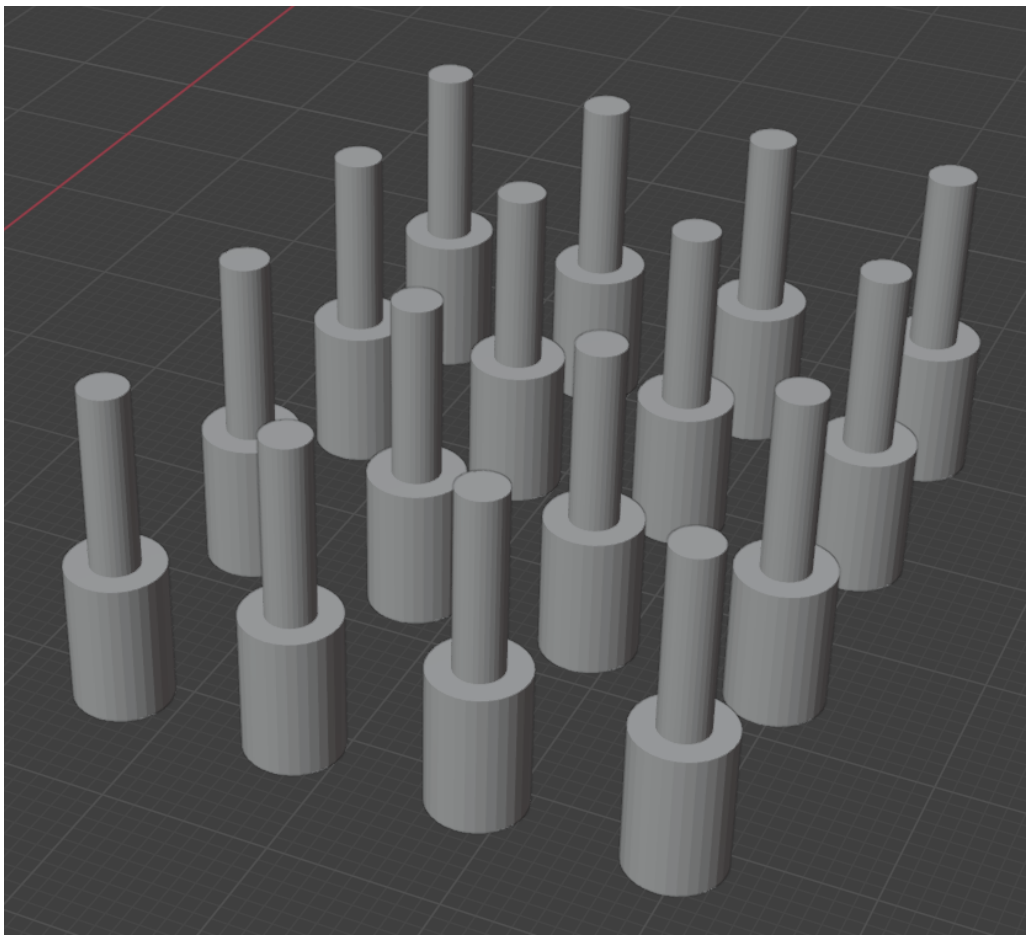

Figure 4.5: Button extenders in Blender

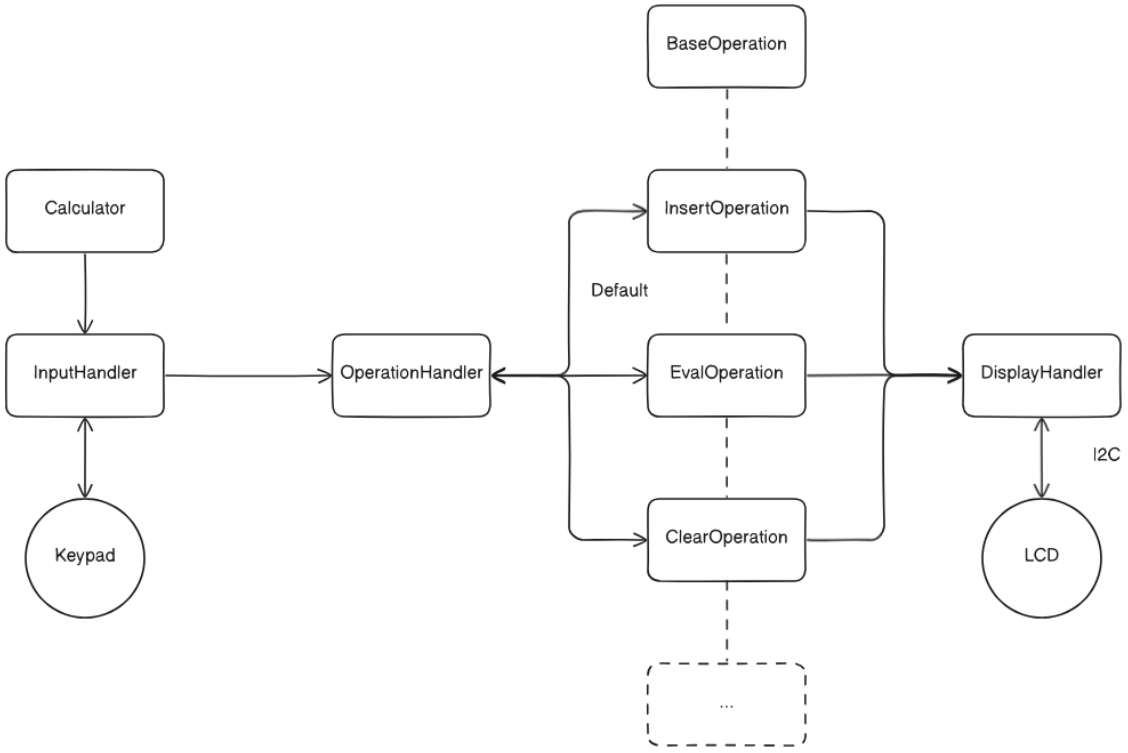

Figure 4.6: High-level initial design of calculator application

triggered at the press of a button.

4.3.2 Calculator application

Prior to the implementation phase's beginning, a high-level system design was created, see Fig. 4.6. During the development process, this initial design was evaluated and deemed suboptimal. The reason being attributed to tight coupling between the operations and the DisplayHandler class. In this setup, it would compromise modularity given that a change in the display implementation would require modification of all operations.

Therefore, the new design Fig. 4.7 was created centralized around a CalculatorState class, meaning that modularity is kept and the implementation is simplified. In this setup, all operations would instead interact with the display through said class. Additionally, the new design is more logical in its structure, with the calculator class consisting of a keypad, LCD, and state. The keypad consists of operations which modify the state, while the LCD updates based on it.

The final application defines all calculator functions as operations with the capability to manipulate the CalculatorState class. These operations are mapped to the keypad and are executed by it, while the LCD updates based on the values in the CalculatorState class. The application implements seven distinct operations as described in more detail in Section 0.2.

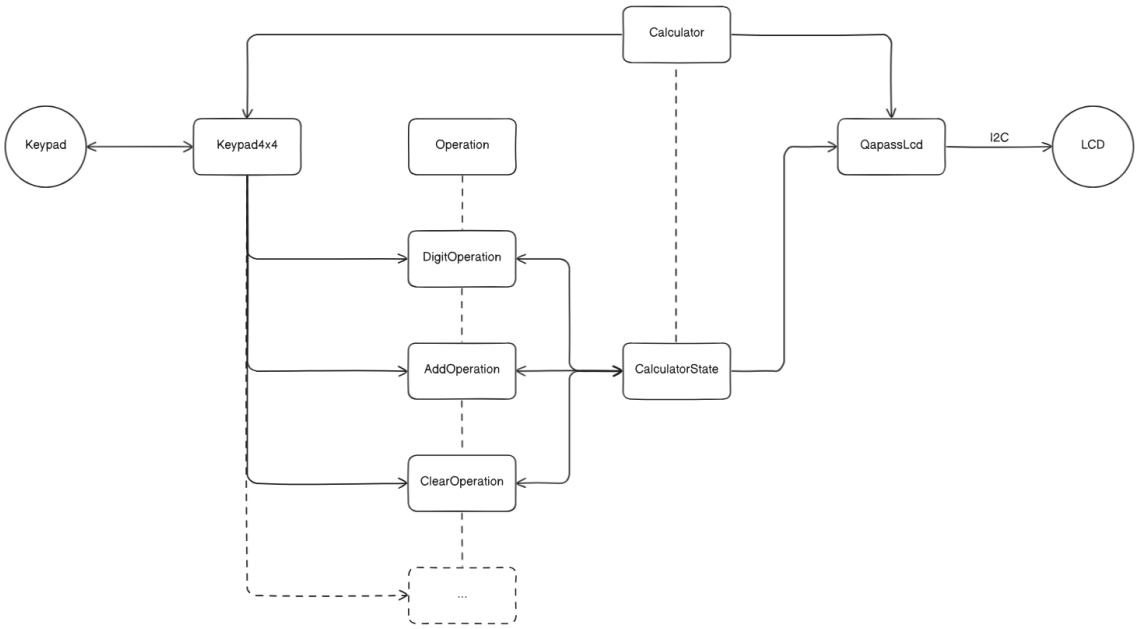

Figure 4.7: High-level final design of calculator application

The mathematical operations usually fetch the current sum and the input field value to perform the calculation, while sometimes doing additional adjustments to match traditional calculator behavior. In addition, the PrevOperation pointer is updated in order to allow for the EvalOperation to function as one would expect. The functional operations on the other hand simply operate on the CalculatorState according to their defined behavior.

5
Results

The results in this chapter are organized into sections that address the questions outlined in the planning report, followed by additional findings.

5.1 Validity as an open alternative to traditional calculators

The calculator performs basic operations without issues. Be it addition, subtraction, multiplication, and division even during extended periods of use. The software, written in Java using the Pi4J library, is fully accessible and modifiable, enabling users to adapt it to their needs. The hardware configuration, comprising a Raspberry Pi, a 4x4 keypad, and an I2C LCD display is also open for customization. Therefore we can determine that it is a valid alternative to a traditional calculator in terms of features and potential extensibility. However, it is worth noting that the current implementation has several limitations listed here:

- The case that encompasses the calculator is very big.
- There is no power button.
- There is no way to charge the device when enclosed in the case.
- The case has no sealing mechanism.
- Calculations are limited to Java's double precision ranges.
- The current implementation uses floating-point arithmetic which can cause rounding errors.
- The software is tied to the Raspberry Pi ecosystem due to the use of the Pi4j library.

Therefore, we can conclude that this hardware and case configuration is not a suitable alternative in terms of compactness. Apart from this we can conclude that the reliance of floating-point arithmetic in the current implementation, while not hindering basic functionality, could be improved upon by utilizing Java's BigDecimal class for exact decimal calculations.

5.2 Ease of adding additional functionality

The underlying system was designed with smooth extensibility and modularity in mind, and therefore, thrives in those endeavors. To demonstrate, these are the steps

required in order to implement a square operation:

1. Create the SquareOperation class: A new class was defined in a new file by extending the existing operation class, inheriting its structure and ensuring consistency with other operations like addition or subtraction.
2. Setup the constructor: The constructor was configured to pass the shared CalculatorState object to the superclass, linking the new operation to the calculator's current value.
3. Implement the execute Method: The operational logic was added to compute the square of the value in the input field.
4. Assign the Operation: The new operation was mapped to a keypad button by updating the Calculator class's initialization method in its respective file, requiring only a single line of code to register the button binding.

This implementation is almost completely restricted to a single newly created file dedicated to the new class consisting of the added feature. Thus, restricting the risk of conflict significantly. The only outlier being a single line added to the initialization method of the Calculator class, for mapping the operation to the desired button.

As previously mentioned, the calculator is modular and extensible. Therefore it has the capacity to be tailored to one's needs and wishes. It has the capacity for modifications ranging from:

- The creation, alteration or removal of mathematical operations.
- The creation, alteration or removal of calculator functions. For example, the clearing of the screen, resetting the calculator state, and repeating an operation.
- The mapping of mathematical operations or calculator functions to the desired button.
- The swapping of peripherals by implementing their communication protocols, or through the reuse of existing implementations. For example, for a keypad or an LCD.
- The modification of core design choices. For example, the use of a state class, or the lack of an event driven design.
- The swapping of Raspberry Pi models. The calculator application as a consequence of being written in Java has the capacity of running on any Raspberry Pi model capable of running a java virtual machine, "write once run anywhere" (WORA).

The system's modular design theoretically allows for limitless expansion, in which constraining components would be replaced or practical solutions would be employed to resolve forthcoming issues. Examples include:

- Too many operations for use with the keypad: Swap keypad, add an additional keypad or add software layers.
- Computational capabilities are insufficient: Upgrade Raspberry Pi, connect components with the required capabilities or outsource to an online solution as the Raspberry Pi has internet access.

- Screen is incapable of displaying results with enough accuracy: Upgrade the display, implement scrolling text or add an additional display.

5.3 Suitability for portability given power consumption

The power consumption of the Raspberry Pi 3B and even other Raspberry Pi models, is significantly higher compared to that of traditional calculators [11, 12]. In fact using these values we can determine that the Raspberry Pi 3B calculator consumes approximately 24000× more power than a traditional calculator.

As given by the relative change formula:

$$\text{Relative change} = \frac{B - A}{A} \times 100\% \tag{5.1}$$

In which:
1. B is 2.4 W
2. A is 0.1 mW

Furthermore, a measurement conducted using a Python script running in conjunction with the calculator application, logging the uptime each minute, resulted in a battery life of 21 hours and 6 minutes. This in comparison to the lowest approximate battery life of several Casio calculators as indicated by [12], would mean that the battery life of a traditional calculator would approximately be 830× longer than the Raspberry Pi calculator.

As given by the relative change formula:

$$\text{Relative change} = \frac{B - A}{A} \times 100\% \tag{5.2}$$

In which:
1. B is 2 years in seconds
2. A is 21 hours and 6 minutes in seconds

We can therefore conclude that the Raspberry Pi based calculator has higher power consumption and is significantly worse in terms of battery life. However, this does not mean that the Raspberry Pi calculator is unsuitable for portability. In fact it can be argued that is more than suitable given that it exceeds the average exam length of several universities [13, 14, 15]. Especially, when you consider that power consumption can be reduced further through optimization as indicated by [16] and the fact that other Raspberry Pi models consume less power [11].

6
Discussion and conclusion

6.1 Method Evaluation

The division of the project into distinct phases proved fruitful in allocating work. The current division allowed critical issues to surface and be circumvented early, like the compatibility issue with the Raspberry Pi Zero W. The time allocated for each phase was generally sufficient, with various problems including everything from dependency issues to case design never falling outside the allotted time. However, the Shell and Study phase significantly underestimated the time and effort involved in the learning of Blender and designing a case with it. In hindsight, more time should have been allocated to the case design portion of the project.

6.2 Future work

The project can be developed further in several areas. Firstly, investigating alternative dependencies for the Raspberry Pi Zero W to address the compatibility issue, which would result in a smaller form factor and lower power consumption. Secondly, the case design can be optimized through reduced material usage by means of rounded edges, cutouts for charging and power button access. Finally, the accuracy of calculations can be significantly increased by adjusting the implementation to instead utilize the BigDecimal class provided by the Java Standard Library.

6.3 Critical discussion

The project demonstrated several strengths, particularly the modular calculator application that allows for easy extensibility, as demonstrated by the implementation of the square operation. Additionally the structure of the project proved effective in revealing and circumventing issues early.

However, significant challenges emerged during development. The case design suffered from inexperience in Blender, limiting the design to simple geometric shapes, resulting in excessive material usage. The project also clearly displays the significantly higher power consumption present in the calculator in comparison to traditional calculators, which compromises one of the core utility purposes of calculators, that being portability.

These findings highlight fundamental tradeoffs in the design. The Raspberry Pi offers extensive computational capabilities that allow for more complicated functionality. However, it comes at the cost of a much higher power consumption.

6.4 Source evaluation

The sources referenced in this report are all from reputable institutions, organizations and persons. They include established universities, official company documentation, academic publications from IEEE conferences and government standards organizations (RFC specifications). Additionally, the power consumption data was provided by Jeff Geerling, who is a published author, content creator and developer with extensive Raspberry Pi experience.

6.5 Conclusion

The project demonstrated the feasibility of creating an open source calculator alternative using readily available components. This directly addresses the closed system limitations described in the introduction. The implementation achieved the primary objectives:

- Functionality: The calculator performs all basic arithmetic operations, matching the feature set of a traditional basic calculator, with the addition of the potential for further enhancements.
- Extensibility: The modular design enables the addition of new operations requiring minimal changes in code and reducing integration risks. The demonstration of adding the square operation in four simple steps solidifies this.
- Accessibility: The hardware and software components are fully accessible and modifiable, giving users complete control of their device.
- Documentation: The described development process serves as a comprehensive case study of the creation of open systems.

Bibliography

[1] University of Cambridge, "The life of Pi: Ten years of Raspberry Pi", 2022. [Online]. Available: https://www.cam.ac.uk/stories/raspberrypi (accessed on: 2025-05-24)

[2] Raspberry Pi Holdings., "Raspberry Pi OS", 2024. [Online]. Available: https://www.raspberrypi.com/documentation/computers/os.html#content (accessed on: 2025-05-19).

[3] The Debian Project, "About Debian", 2025. [Online]. Available: https://www.debian.org/intro/about(accessedon:2025-05-19) (accessed on: 2025-05-27)

[4] Oracle Corporation, "Java™ Programming Language", n.d. [Online] Available: https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html (accessed on: 2025-05-25)

[5] Oracle Corporation, "How Will Java Technology Change My Life?", n.d. [Online] Available: https://docs.oracle.com/javase/tutorial/getStarted/intro/changemylife.html (accessed on: 2025-05-25)

[6] Github, "Octoverse: AI leads Python to top language as the number of global developers surges", 2024. [Online] Available: https://github.blog/news-insights/octoverse/octoverse-2024/

[7] Blender Foundation, "Free Software Never Looked This Awesome", n.d. [Online] Available: https://www.blender.org/features/ (accessed on: 2025-05-21)

[8] The Secure Shell (SSH) Transport Layer Protocol, RFC 4253, Network Working Group, 2006. Available: https://datatracker.ietf.org/doc/html/rfc4253 (accessed on: 2025-05-21)

[9] The Apache Software Foundation, "What is Maven?", 2025. [Online] Available: https://maven.apache.org/what-is-maven.html (accessed on: 2025-05-21)

[10] JetBrains, "IntelliJ IDEA The IDE for Professional Development in Java and Kotlin", n.d. [Online] Available: https://www.jetbrains.com/idea/ (accessed on: 2025-05-21)

[11] J. Geerling, "Power Consumption Benchmarks" Raspberry Pi Dramble. [Online] Available: https://pidramble.com/wiki/benchmarks/power-consumption (accessed on: 2025-05-20)

[12] Casio, "Specifications", n.d. [Online] Available: https://support.casio.com/global/en/calc/manual/fx-82MS_85MS_220PLUS_300MS_350MS_en/technical_informatoin/specifications.html (accessed on: 2025-05-20)

[13] Chalmers University of Technology, "Find examination dates", 2025. [Online] Available: https://www.chalmers.se/en/education/your-studies/examinations-and-other-summative-assessments/

find-examination-dates/?sort.exDate=asc&sort.starts=asc (accessed on: 2025-05-21)

[14] University of Oxford, "Advanced Computer Science", 2025 [Online] Available: https://www.ox.ac.uk/sites/files/oxford/JACS2425T.pdf (accessed on: 2025-05-21)

[15] MIT, "MIT Final Exam Schedule: Spring 2025", 2025. [Online] Available: https://finalexams.mit.edu/#/Finals

[16] F. Astudillo-Salinas et al., "Minimizing the power consumption in Raspberry Pi to use as a remote WSN gateway," in 2016 8th IEEE Latin-American Conference on Communications (LATINCOM), 2016, pp. 1-5, doi:10.1109/LATINCOM.2016.7811590.

A

Appendix 1

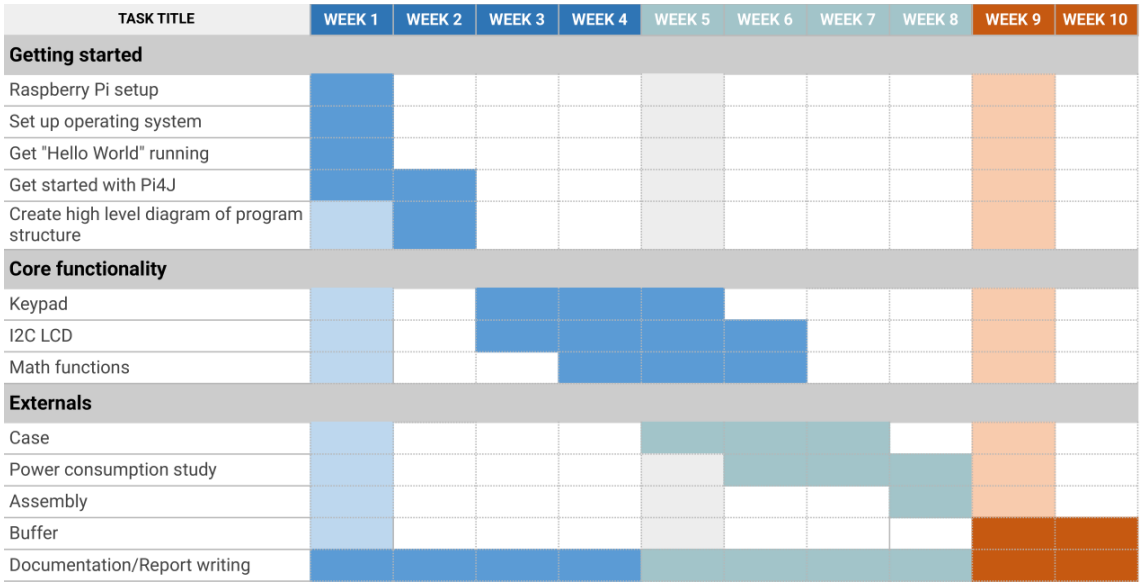

Figure A.1: Gantt chart of project plan

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS