Master Thesis

Prototype Development of a Handheld Speed Camera

for the attainment of the academic degree
Master of Mechanical Engineering
submitted by

Muhammad Haziq Bin Mohd Sabtu

Technische Hochschule Brandenburg
University of Applied Sciences
Fachbereich Technik

Technische Hochschule Brandenburg
Department of Technology
Studiengang Maschinenbau
Reviewer 1: Prof. Dr.-Ing. Christian Oertel
Reviewer 2: Prof. Dr.-Ing. Peter M. Flassig

Brandenburg, 1. October 2023

I, MUHAMMAD HAZIQ BIN MOHD SABTU, student in the Mechanical Engineering program of the Brandenburg University of Applied Sciences, declare in oath that this thesis has been written by myself and has not been written with other than the other than the indicated aids. The research presented in this thesis, including all references and sources, is entirely my own work, unless otherwise stated.

I further declare that the work presented in this thesis has not been previously included in any other work, academic or otherwise, for which I have received a degree or diploma. Any assistance received during the course of this research has been acknowledged, and all contributions by individuals or sources have been appropriately cited.

MUHAMMAD HAZIQ BIN MOHD SABTU

Abstract

This research presents the development of a handheld speed measurement device offering a cost-effective alternative to conventional speed pistols. A prototype has been engineered by integrating affordable computational components and high-quality cameras and employing 3D printing technology. The design process was guided by VDI Guideline 2221. Multiple variants were generated and evaluated against their technical and economic rating to find the optimal design. The selected variant was then produced by additive manufacturing and tested for functionality.

Keywords: Speed Mesurement, 3D Printing, VDI Guideline 2221

Kurzfassung

Die vorliegende Arbeit befasst sich mit der Entwicklung eines mobilen Geschwindigkeitsmessgeräts, das eine kostengünstige Alternative zu herkömmlichen Geschwindigkeitspistolen darstellt. Ein Prototyp wurde durch die Integration von kostengünstigen Rechenkomponenten und hochwertigen Kameras sowie durch den Einsatz von 3D-Drucktechnologie entwickelt. Der Entwurfsprozess orientierte sich an der VDI-Richtlinie 2221. Es wurden mehrere Varianten erstellt und anhand ihrer technischen und wirtschaftlichen Bewertung bewertet, um das optimale Design zu finden. Die ausgewählte Variante wurde dann mittels additiver Fertigung hergestellt und auf ihre Funktionalität getestet.

Schlüsselwörter: Geschwindigkeitsmessung, 3D-Druck, VDI-Richtlinie 2221

Contents

List of Abbreviations **1**

1 Introduction **2**
- 1.1 Motivation 2
- 1.2 Problem Statement 2
- 1.3 Previous Work 3
- 1.4 Research Objectives 3

2 State of the Art - Speed Pistol **5**

I Prototype Development **7**

3 Literature Review **8**
- 3.1 Methodic Product Development 8
- 3.2 Fused Deposition Modeling 9

4 Planning and Task Clarification **14**
- 4.1 Establishing the Prototype's Requirements 15
- 4.2 Identifying the Prototype's Requirements 16
- 4.3 Requirement List 21

5 Conceptual Design **24**
- 5.1 Abstraction 24
- 5.2 Function Structures 26
- 5.3 Developing Working Principles 29
- 5.4 Combination of Working Principles 31
- 5.5 Firming Up into Principle Solution Variants 33
- 5.6 Filtering of Solution Variants 41

6 Embodiment Design 44
- 6.1 Basic Rules of Embodiment Design . . . 44
- 6.2 Guideline of Embodiment Design . . . 47
- 6.3 Preliminary Design . . . 49
- 6.4 Evaluation with VDI 2225 . . . 61

7 Detail Design 69

8 Printing and Assembly 76
- 8.1 Printing . . . 76
- 8.2 Assembly . . . 77
- 8.3 Final Product . . . 83

9 Conclusion 86

II GUI Development 88

10 Literature Review 89
- 10.1 Waterfall Model . . . 89
- 10.2 Model-View-Controller . . . 90

11 Requirement Analysis 92
- 11.1 Project Requirements . . . 94

12 Design 96
- 12.1 Software Architechture . . . 96
- 12.2 User Interface Design . . . 100

13 Coding 107
- 13.1 View Implementation . . . 107
- 13.2 Model Implementation . . . 114
- 13.3 Controller Implementation . . . 116
- 13.4 Algorithm Improvement . . . 119

14 Testing and Maintenance 122
- 14.1 Unit Testing . . . 122
- 14.2 Maintenance . . . 124

15 Conclusion 126

III Indexes and Appendix 127

List of Figures 128

List of Tables 130

Bibliography 132

A Appendix 141
A.1 Sketches of Working Principles 141
A.2 CAD Drawings 147
A.3 Technical Specifications 154
A.4 Cost Calculation 165
A.5 Evaluation Data 184
A.6 Documentation 192
A.7 Code snippets 192
A.8 C++ Unit Tests for Bank Account Management 192

List of Abbreviations

3D	Three-Dimensional
ABS	Acrylonitrile Butadiene Styrene
FDM	Fused Deposition Modeling
G-Code	Geometric Code
GPIO	General Purpose Input Output
HDMI	High Definition Multimedia Interface
LAN	Local Area Network
LCD	Liquid Crystal Display
LIDAR	Light Detection and Ranging
PC	Personal Computer
PET	Polyethylenterephthalat
PETG	Polyethylenterephthalat Glycol
PLA	Polylactic Acid
POS	Point of Service
RADAR	Radio Detection and Ranging
USB	Universal Serial Bus
VDI	Verein Deutscher Ingenieure

1 Introduction

1.1 Motivation

Excessive speeding poses a significant threat to the safety of people on our roads. It impacts those directly involved in accidents, their families, and their communities. Over four years, from 2019 to 2022, Germany experienced an average of nearly 38,000 accidents annually due to speeding [Sta23]. These accidents can have long-lasting consequences, inflicting physical and emotional harm. Countless individuals are injured or tragically losing their lives, profoundly impacting their loved ones. It is crucial to address this issue, and the goal of this thesis is to develop an innovative speed gun system to aid in the reduction of excessive speeding.

1.2 Problem Statement

Law enforcement agencies and road safety organizations often find the current speed gun technologies available in the market too expensive for widespread adoption. These technologies can cost around 2000 €, which creates a financial barrier preventing effective speed monitoring in many regions. This thesis aims to explore alternative speed gun systems that use computer vision to address this challenge. By using this technology, we aim to create a more affordable and robust solution for speed monitoring, which will address the financial constraints of traditional speed guns and open up new possibilities for innovation and customization in speed monitoring technology. Ultimately, this will enhance road safety and reduce accidents caused by excessive speed.

1.3 Previous Work

Previous research [BMS23] has delved into various computer vision methods to tackle the complex challenge of speed monitoring. One remarkable innovation is the image alignment algorithm, which employs feature detection techniques to rectify any inadvertent movement during video recording. This algorithm enhances the stability of recorded footage and ensures accurate speed assessment.

Another technique exlored is the optical flow analysis, which is a method that tracks the movement of objects in a video sequence. In this approach, the algorithm analyzes the movement of pixels between consecutive frames to identify moving objects. This technique is computationally efficient, making it a practical and cost-effective solution for real-time speed monitoring.

Lastly, a method to measure object distance using the pinhole camera model was also explored. This model leverages the pinhole camera's geometry to calculate the distance of objects from the camera's vantage point. Integrating this model into speed monitoring provides a streamlined and economically viable method for assessing speeds with high precision.

After conducting thorough testing, it has been established that these techniques exhibit a high level of effectiveness in monitoring speed, achieving an impressive accuracy rate of 94 %. This outcome signifies their viability for advancing to the next stage of development and potential incorporation into a prototype speed gun system.

1.4 Research Objectives

Our research aims to develop an alternative to current implementation of speed gun by utilizing computer vision technology. The work will consist of two parts. In the first part, we will design the physical structure of the prototype using the VDI 2221 methodology. The second part will focus on software development, creating an easy-to-use interface that leverages computer vision for real-time speed assessment and data analysis. The goal is to create a state-of-the-art speed gun that addresses speeding concerns and sets a new road safety technology standard.

2 State of the Art - Speed Pistol

Law enforcement agencies worldwide utilize speed guns, also known as radar guns or speed pistols, as crucial tools to combat speeding. These devices play a pivotal role in maintaining road safety by accurately measuring the velocity of vehicles on the road [Hul20].

The current implementation of speed pistol usually utilize either the LIDAR (Light Detection and Ranging) or RADAR (Radio Detection and Ranging) technology [Rad23] [Fly23] [Sig23] [Tec23]. These technologies serve as the cornerstone of modern speed measurement devices, allowing law enforcement officers to gauge the speed of vehicles in real-time accurately.

LIDAR technology operates on emitting laser pulses towards a target vehicle and measuring the time it takes for the light to bounce back [Fly23]. The LIDAR device can precisely calculate the vehicle's speed by analyzing the returned signals.

On the other hand, RADAR technology has been a reliable tool in speed enforcement for decades. Radar guns emit radio waves as a focused beam, which bounce off the target vehicle and return to the device. By analyzing the frequency shift of the returned signal, the device can determine the vehicle's speed [rad].

However, both of these technologies have their limitations. For instance, LIDAR can be affected by adverse weather conditions like rain or fog, potentially reducing its effectiveness [DSPB23]. Conversely, RADAR signals may be susceptible to interference from nearby objects, which can complicate speed measurements in densely populated areas [Hos].

Both LIDAR and RADAR technologies, while highly effective in speed measurement, come with a notable price tag. A LIDAR device can range from several thousand to tens of thousands of euros [Pro]. In comparison, RADAR guns, though generally less expensive than LIDAR, can still cost around 2000 € for a high-quality unit [Sup].

These costs present a significant consideration for law enforcement agencies, especially those operating with limited budgets or in smaller jurisdictions. This limitation highlights the need for exploring more cost-effective alternatives without compromising accuracy and efficiency in speed enforcement.

Part I

Prototype Development

3 Literature Review

3.1 Methodic Product Development

Methodic Product Development, by Pahl and Beitz, stress the importance of systematic design in product development [Pah07, 9]. They differentiate between design science and methodology, with the latter being a practical approach based on scientific analysis and practical experience.

Pahl and Beitz describe the product development process as a series of stages, each with defined objectives and activities [Pah07, 128-133]. The four main stages are:

Planning and Task Clarification: The process starts with planning and defining tasks, often in collaboration with the marketing or dedicated planning team. It is essential to thoroughly understand the task, whether from a product proposal or a customer request. This step involves gaining detailed insights into requirements, constraints, and their importance, forming a solid foundation for the next steps.

Conceptual Design: This phase involves defining the necessary functions, establishing working principles, and integrating them into a working structure. Ultimately, this leads to creating a fundamental solution that embodies the core of the design vision.

Embodiment Design: Guided by technical and economic considerations, the physical structure is defined. Various initial layouts are assessed to identify design strengths and weaknesses, ultimately leading to selecting the most promising version.

Detail Design: In this phase, precise arrangements, dimensions, materials, and other aspects are defined. Product documentation is created, including drawings, parts lists, and assembly instructions.

Figure 3.1 shows the stages involved in Pahl and Beitz's design process.

3.2 Fused Deposition Modeling

Fused deposition modeling (FDM) is a widely used technique in additive manufacturing, particularly in 3D printing. It offers several advantages that contribute to its popularity in various industries. One of the main advantages of FDM is its ability to reproduce complex geometries with high precision and accuracy [GGA18].

This method makes it suitable for creating intricate and customized designs that may not be achievable with traditional manufacturing methods. Additionally, FDM is a cost-effective process as it reduces material waste by only depositing the necessary amount of material layer by layer [GGA18], which not only saves costs but also promotes sustainability in manufacturing.

Common plastics used in FDM include Acrylonitrile Butadiene Styrene (ABS), Polylactic Acid (PLA), and Polyethyleneterephthalate (PET) [Bat]. These materials have different mechanical properties, advantages, and disadvantages, making them suitable for different applications.

3.2.1 Original Prusa i3 MK3S+

The Original Prusa i3 MK3S+ is an FDM 3D printer designed for desktop use, ideal for tasks such as rapid prototyping and small-scale production. With a build volume of 250 mm x 210 mm x 200 mm, it can achieve layer heights ranging from 0.05 mm to 0.35 mm [Prua]. This printer has a heated bed and is compatible with various materials such as PLA, ABS, PETG, and nylon [Prua]. The default nozzle size is 0.4 mm, although alternate sizes can be utilized based

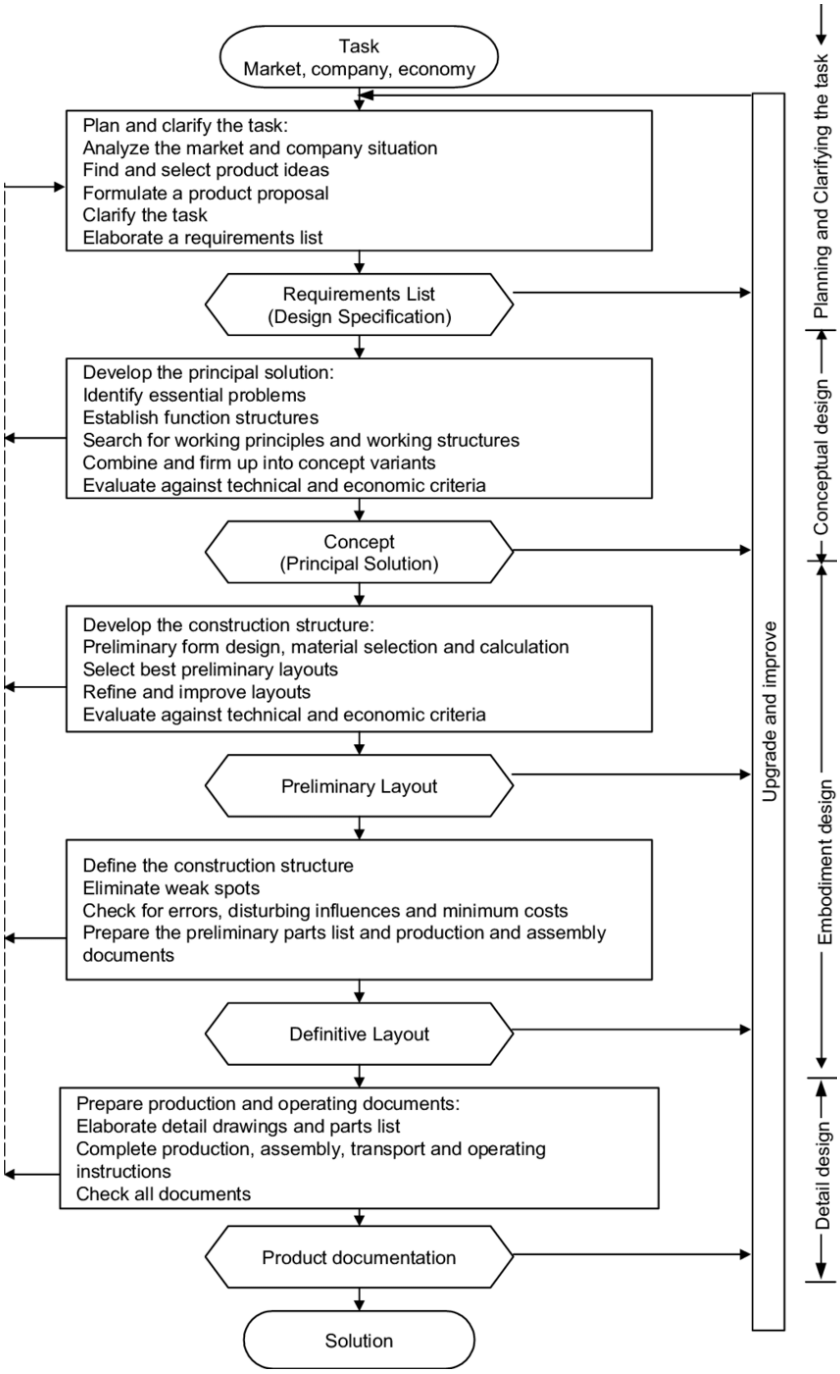


Figure 3.1: Pahl and Beitz's Design Process [Pah07, 130]

on specific printing needs.

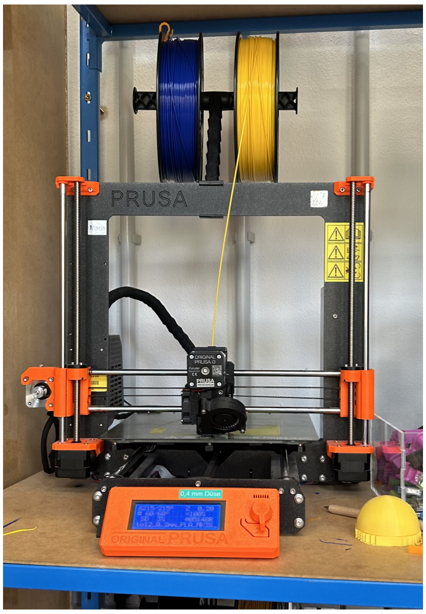


Figure 3.2: Original Prusa i3 MK3S+

This 3D printer is accessible to students and faculty at the University of Applied Sciences Brandenburg and will play a pivotal role in producing the prototype for this work. Figure 3.2 visually represents the Original Prusa i3 MK3S+, located within the University of Applied Sciences Brandenburg Workshop.

3.2.2 PrusaSlicer

PrusaSlicer is a free and open-source slicing software that converts 3D models into G-code, a language instructing the 3D printer on printing the object. It is compatible with a wide range of 3D printers and supports a variety of filament materials. PrusaSlicer offers many features that allow users to customize the printing process to suit their needs.

PrusaSlicer's ability to adjust printing parameters is crucial. These parameters include layer height, infill density, and print speed. Layer height determines the thickness of each layer of the printed object. Infill density refers to the material used to fill the object's inside. Print speed is the rate at which the printer moves while printing the object. Adjusting these parameters can help achieve

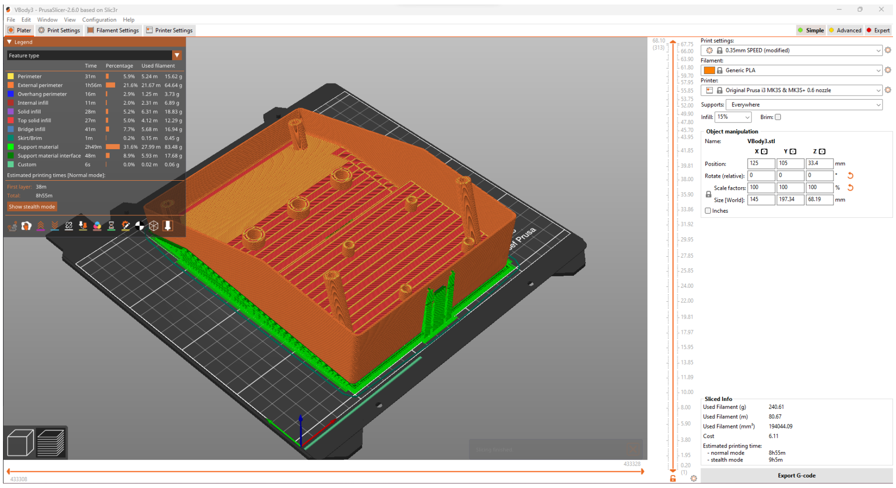


Figure 3.3: Example View of PrusaSlicer

the desired quality of the final product.

PrusaSlicer offers an essential function of adding support to the 3D prints. Supports are structures that print alongside the object to provide extra stability during printing. They prevent any potential collapse or deformation of the object while printing. Supports can be added manually or automatically depending on the complexity of the printed object.

This software can also generate a preview of the printed object, which lets users visualize the result. Additionally, PrusaSlicer provides an estimate of the amount of filament required for the printing process and the duration of the printing process. Figure 3.3 shows a screenshot of PrusaSlicer. The left side of the window shows the 3D model of the object, while the right side shows the various parameters that can be adjusted.

3.2.3 Printing Cost

To perform a cost analysis of the 3D printing process, we will consider the following parameters:

- Material Cost (C_m)

- Energy Cost (C_e)

Equation 3.1 shows the formula used to calculate the material cost. This formula involves multiplying the weight of filament used (m_{fil}) by the cost of filament per kilogram(C_{fil}). The cost of the filament is dependent on the type of material used. We will use PLA for this project, which costs 29.99 €/kg [Prub].

Energy cost refers to the electricity cost of the printing process and is calculated using Equation 3.2. The printing duration (t_p) is estimated directly from the PrusaSlicer software, while the power consumption (P) of the printer is estimated to be about 0.08 kW [Prua]. By observing the average price of electricity in Germany for the year 2022 [Poo], the electricity price (C_{el}) is estimated at 0.235 €/kWh.

Equation 3.3 shows the formula for calculating the cost of 3D printing.

$$C_m = m_{fil} \cdot C_{fil} \tag{3.1}$$

$$C_e = t_p \cdot C_{el} \cdot P \tag{3.2}$$

$$C_{print} = C_m + C_e \tag{3.3}$$

4 Planning and Task Clarification

This chapter delves into planning and clarifying tasks for the prototype, as can be seen in Figure 4.1. As mentioned previously in Chapter 3.1, this step plays a critical role in the product development process. They involve precisely defining and understanding the requirements and expectations related to a specific task or project. The goal is to eliminate confusion and ensure comprehension among all involved parties.

During this step, the task's specific goals, limitations, and requirements are identified [Pah07, 145]. Additionally, Pahl and Beitz formulated a series of questions that must be answered to help better understand the project requirements [Pah07, 145]. These questions are:

- What is the objective of the solution?
- What characteristics should the solution have?
- What characteristics should the solution avoid?

The requirements for the solution can be identified and spelled out by answering these questions. These requirements will serve as the basis for the subsequent phases of the product development process. The outcome of this step is a list of requirements that outline the needs, expectations, and restrictions tied to the task [Pah07, 145].

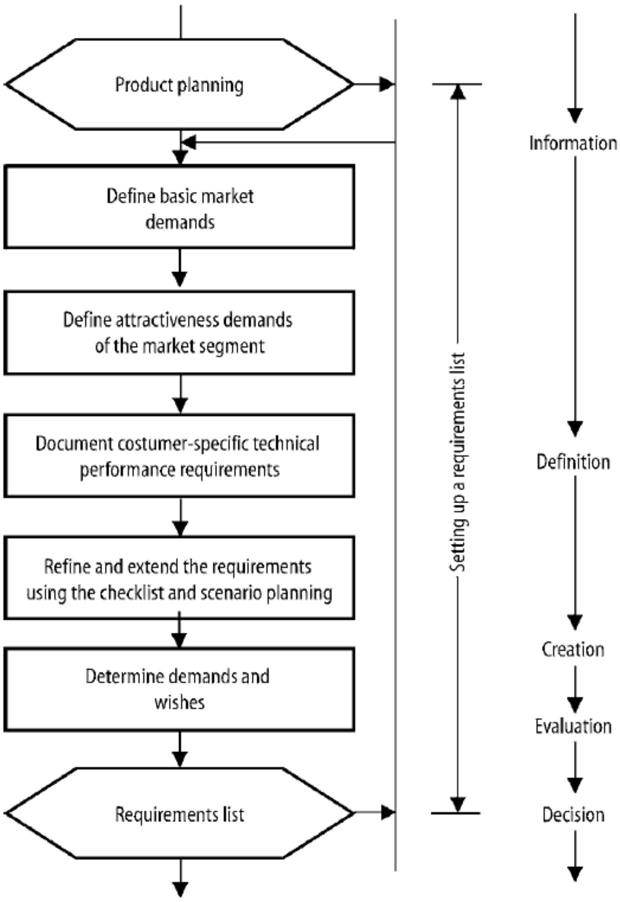


Figure 4.1: Planning and Task Clarification [Pah07, 146]

4.1 Establishing the Prototype's Requirements

To properly establish the requirements for the prototype, it is suggested to properly define the objectives of the prototype and classify them into demands and wishes [Pah07, 146].

Demands, as described by Pahl and Beitz [Pah07, 147], are the essential and non-negotiable requirements that must be fulfilled for the product to be considered successful. They represent the core functionality and characteristics that the product must possess to meet its intended purpose and provide value to the users. Demands are typically based on objective criteria and are crucial for ensuring the product's functionality and compliance.

On the other hand, wishes, as defined by the authors [Pah07, 147], are desirable but non-essential requirements or features that clients or stakeholders would like to see in the product. Wishes often involve additional functionalities, aesthetics, or user experience enhancements that would provide added value or differentiate the product in the market. While wishes may not be mandatory, they can contribute to customer satisfaction, competitive advantage, and overall product excellence.

In addition, if possible, all of the requirements defined must be quantifiable, which means that the requirements must be measurable and testable [Pah07, 147]. This specification is essential for ensuring that the requirements are met and that the product can fulfill its intended purpose.

4.2 Identifying the Prototype's Requirements

In this section, the requirements of the prototype will be established. The checklist (see Figure 4.2) will be used as a guideline to ensure that all the requirements are correctly identified and defined.

Geometry

When creating a prototype, adhering to specific size parameters is essential to ensure its functionality and usability for end-users or customers. The prototype must conform to a general size limit of 300 mm x 300 mm x 300 mm (length x width x height).

Additionally, we must consider the limitations of the 3D printer's size capacity. We have opted to employ the printer mentioned previously in Section 3.2.1. This particular printer has a maximum printing area of 210 mm x 210 mm x 250 mm [Prua].

Ideally, each printed component should fall within the specified printing size range. However, should a component exceed these dimensions, it will be nec-

Main headings	Examples
Geometry	Size, height, breadth, length, diameter, space requirement, number, arrangement, connection, extension
Kinematics	Type of motion, direction of motion, velocity, acceleration
Forces	Direction of force, magnitude of force, frequency, weight, load, deformation, stiffness, elasticity, inertia forces, resonance
Energy	Output, efficiency, loss, friction, ventilation, state, pressure, temperature, heating, cooling, supply, storage, capacity, conversion.
Material	Flow and transport of materials. Physical and chemical properties of the initial and final product, auxiliary materials, prescribed materials (food regulations etc)
Signals	Inputs and outputs, form, display, control equipment.
Safety	Direct safety systems, operational and environmental safety.
Ergonomics	Man-machine relationship, type of operation, operating height, clarity of layout, sitting comfort, lighting, shape compatibility.
Production	Factory limitations, maximum possible dimensions, preferred production methods, means of production, achievable quality and tolerances, wastage.
Quality control	Possibilities of testing and measuring, application of special regulations and standards.
Assembly	Special regulations, installation, siting, foundations.
Transport	Limitations due to lifting gear, clearance, means of transport (height and weight), nature and conditions of despatch.
Operation	Quietness, wear, special uses, marketing area, destination (for example, sulphurous atmosphere, tropical conditions).
Maintenance	Servicing intervals (if any), inspection, exchange and repair, painting, cleaning.
Recycling	Reuse, reprocessing, waste disposal, storage
Costs	Maximum permissible manufacturing costs, cost of tooling, investment and depreciation.
Schedules	End date of development, project planning and control, delivery date

Figure 4.2: Checklist for Establishing the Prototype's Requirements [Pah07, 149]

essary to divide it into two or more parts to facilitate printing. This approach guarantees that each part can be accommodated within the printer's size constraints.

Energy

The energy required for the prototype is crucial because it determines its usefulness and convenience. We have set a requirement that the prototype should function independently for at least an hour using the provided power supply. This guideline is in place to ensure that the prototype can function autonomously and provide users with a seamless experience.

By meeting this requirement, the prototype demonstrates that it can operate reasonably without frequent charging or relying on external power sources. This feature enables users to use the device without any concerns for an extended period, giving them more opportunities to explore its functionality and capabilities.

Forces

The force requirement for the prototype has two main aspects: ensuring it can handle the weight of its components while also adhering to a maximum weight limit.

Firstly, it is crucial to verify that the prototype can effectively support the weight of its components without compromising its overall structure or functionality, which ensures the prototype's durability and ability to withstand the forces exerted by its components. Additionally, it guarantees that the prototype can be manipulated and operated without the risk of damage or malfunction.

Furthermore, there is a specific constraint that the total weight of the prototype must not surpass 2 kg. This requirement encompasses the collective weight of all internal components, including predefined components and any additional materials integrated during the design process. Adhering to this weight limit ensures the prototype remains lightweight and manageable while meeting the intended performance criteria.

Materials

When developing the prototype, it is of utmost importance to thoughtfully consider the specific materials and elements that will be utilized. The client has already preselected specific components for this project, which are mandatory to meet the requirements. These components include the Raspberry Pi 4B, a 7-inch touch screen, and the Raspberry Pi Camera V2.

These elements are fundamental building blocks for the prototype's operation. The Raspberry Pi 4B, functioning as a versatile single-board computer, furnishes computational power and acts as the central control unit for the prototype. The 7-inch touchscreen enhances user interaction by providing a responsive and user-friendly interface for both input and output. The Raspberry Pi Camera V2 facilitates the capturing of images and videos.

Ergonomics

The prototype has specific ergonomics requirements regarding dimensions, weight, and handling. The main goal is to produce a compact and lightweight prototype, making it easy to carry and maneuver, which makes it more comfortable and convenient for users to handle.

Another crucial aspect of the ergonomics requirement is ensuring that users can comfortably hold the prototype. This requirement involves considering the prototype's shape, grip, and balance to ensure it is easy and secure. The design should fit naturally into the user's hand, providing a stable and ergonomic grip.

Production

The client has specified the fabrication of the prototype components to utilize the 3D Printing technology. Furthermore, the design of the prototype must accommodate the use of PLA filament. This material, known for its widespread availability and well-rounded combination of strength and flexibility, aligns with the requirements of the prototype.

Operation

The prototype must fulfill two essential requirements: easy to use freehand without additional support and compatible with a tripod to ensure stability.

The prototype design should allow freehand operation to achieve the first requirement. The design should be ergonomic, with a comfortable grip and easy-to-use controls, making it intuitive for users to use the prototype.

The second requirement is that the prototype should be able to integrate with a tripod to offer better stability when needed. This feature lets users securely attach the prototype to a tripod, ensuring a stationary and stable setup. With tripod compatibility, the prototype can cater to scenarios where steady and controlled operation or positioning is necessary.

Assembly

The assembly requirement for the prototype emphasizes the importance of considering the ease of assembly and disassembly of its components. This design consideration enables users to access the inner components easily, facilitating maintenance and repair tasks.

By designing the prototype with ease of assembly in mind, it becomes simpler for users to put the components together without requiring complex tools or specialized knowledge. This requirement promotes user-friendliness and reduces the time and effort required for initial assembly or subsequent modifications. Similarly, easy disassembly allows users to access the internal components when needed, simplifying troubleshooting, repairs, or component replacements.

The prototype's design should have swappable properties so individual components or modules can be easily removed and replaced without disassembling the entire prototype. Swappable parts enhance modularity, flexibility, and cost-effectiveness, as users can upgrade or replace specific components as needed rather than replace the entire prototype.

Costs

The cost requirement for the prototype focuses on the total cost of production. The manufacturing of the prototype must be within a budget of 300 €, excluding the cost of the predefined components. This budget encompasses the cost of all materials and components used in the production process.

Schedules

The schedule requirement for the prototype focuses on the time required for the design and production phase. The prototype's design must allow for manufacturing within a 2-month window, covering the entire production process, from design to assembly.

Durability

The durability standard for the prototype encompasses considerations for its ability to withstand dust and water, if possible. While achieving complete resistance may only sometimes be attainable, efforts should be directed toward enhancing the prototype's durability in these aspects.

Concerning dust resistance, the prototype's design should minimize the entry of dust particles into its internal components and sensitive areas. This requirement involves using appropriate seals, filters, or protective enclosures to prevent dust from negatively impacting the prototype's performance or functionality.

In terms of water resistance, if relevant to the intended use, the prototype should demonstrate a level of protection against water penetration. This specification may incorporate waterproof or water-resistant materials, seals, or coatings to shield the internal components from moisture.

4.3 Requirement List

Table 4.1 and Table 4.2 shows the list of requirements for the prototype, including the demands and wishes. The demands are marked with a D, while the wishes are marked with a W.

TH Brandenburg		Requirement List Speed Camera	Issued on 1/7/2023 Page: 1
Changes	**D/W**	**Requirements**	
		Geometry	
	D	1. Length < 300 mm	
	D	2. Width < 300 mm	
	D	3. Height < 300 mm	
	W	Parts size: 210 mm x 210 mm x 250 mm or less	
		Energy	
	D	Minimal operation time: 1 hour	
		Forces	
5/7/2023		Total prototype weight < 2 kg	
		Materials	
	D	Use all predefined components	
		Ergonomics	
	W	Leightweight	
	W	Comfortable handling	
	W	Compact	
		Production	
	D	3D Printed	
	D	Use PLA filament	
		Operation	
	D	Able to be use in freehand	
	D	Able to integrate with tripod stand	
		Assembly	
	W	Simple assembly or component used	
21/8/2023	D	Swappable Parts	

Table 4.1: Requirement List (1/2)

TH Brandenburg		Requirement List Speed Camera	Issued on Page:	1/7/2023 2
Changes	D/W	Requirements		
		Costs		
13/7/2023	D	Manufacturing costs < 300 €		
		Schedules		
	D	Finished by: October 2023		
		Durability		
	W	Resistant against water		
	W	Resistant against dust		

Table 4.2: Requirement List (2/2)

5 Conceptual Design

Following the clarification of the task is the conceptual design, where in this section of the product development process, designers engage in creative exploration and evaluation of various design ideas and concepts.

According to Pahl and Beitz, conceptual design is the stage of the design process where important issues are pinpointed through abstraction [Pah07, 159]. The process involves establishing function structures, searching for suitable working principles, and ultimately combining these elements to create a working structure.

Figure 5.1 shows the steps involved in this phase.

5.1 Abstraction

Due to new technologies, procedures, materials, and scientific discoveries, traditional solution principles or designs may not be able to provide optimal answers, and to overcome fixation on conventional ideas, designers utilize abstraction, focusing on the general and essential aspects rather than particular details [Pah07, 161].

To help in identification of the essential problems, following abstraction techniques are used [Pah07, 165]:

- **Step 1:** Eliminate personal preferences.
- **Step 2:** Omit requirements that have no direct bearing on the function and the essential constraints.

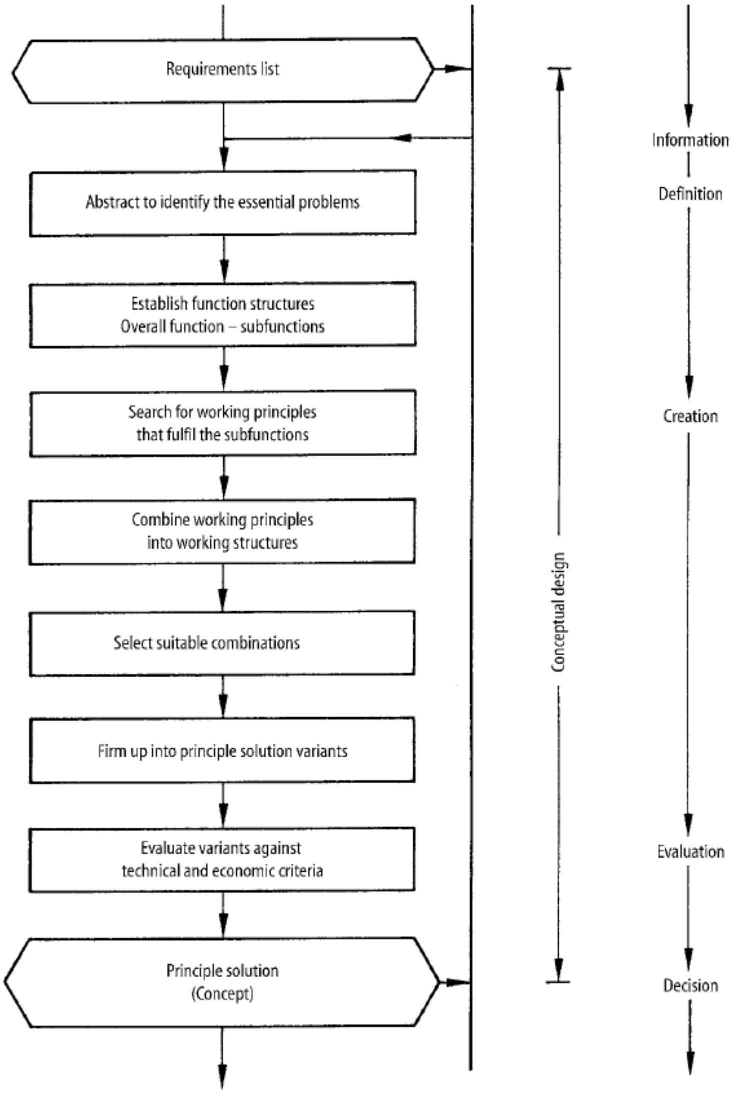


Figure 5.1: Steps in Conceptual Design [Pah07, 160]

- **Step 3:** Transform quantitative into qualitative data and reduce them to essential statements.
- **Step 4:** Generalise the previous step's results.
- **Step 5:** Formulate the problem in solution-neutral terms.

Figure 5.2 shows the result of the abstraction process.

Result of Step 1 and Step 2

- Ergonomic: Comfortable to hold, Easy to use, Weight distributed evenly
- Portable: Lightweight, Small
- Size (MAX):
- Length: 300 mm
- Width: 300 mm
- Height: 300 mm
- Weight (MAX): 2 kg
- Design: Components are packed in a chassis
- Camera: Camera must be presented in the prototype
- Power: Battery powered, Rechargeable battery, Duration min. 1 hour
- Control: Control via touch screen
- Optional Requirements:
- Durability: Water resistance, Dust resistance
- Modular: Easy to assemble and disassemble, Swappable parts
- Features: Mountable on a tripod
- Production: 3D printed parts

Result of Step 3 and Step 4

- Comfortable to hold, easy to use, and have evenly distributed weight.
- Lightweight and small.
- Not exceed 300 mm in dimension.
- Weigh less than 2 kg.
- Power that lasts a minimum of 1 hour.
- Produced with 3D Printer.
- Optional Requirements:
- Durable against water and dust.
- Modular

Result of Step 5 (Problem Formulation)

- Design a portable device that prioritizes user comfort, ease of use, and ergonomic design while utilizing 3D printing.

Figure 5.2: Result of Abstraction Process

5.2 Function Structures

Pahl and Beitz [Pah07, 31] define function structures as a graphical representation of the functions of a system and their interrelationships. It is a hierarchical representation of the functions of a system, starting with the overall function and breaking it down into sub-functions. The function structure is a helpful tool for identifying a system's essential functions and the relationships between the functions.

Figure 5.3 shows the representation of the function structure and the process of breaking down the overall function into sub-functions.

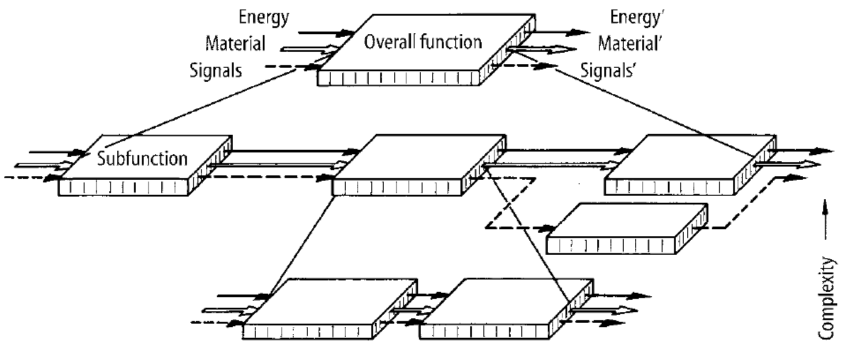


Figure 5.3: Breaking down the overall function into sub-functions [Pah07, 32]

5.2.1 Overall Function

Based on the result of abstraction, the system's overall function can be represented and visualized using a function structure diagram, as shown in Figure 5.4.

In this overall function, the prototype's components are defined as an input, while the prototype itself is defined as the output. The overall function is decomposed into sub-functions in the next section.

Figure 5.4: Overall Function of the System

5.2.2 Sub-Functions

Decomposing the overall function into sub-functions is crucial in the conceptual design process, and as described by Pahl and Beitz [Pah07, 170], the purpose of this decomposition is to reduce the complexity of the overall system and facili-

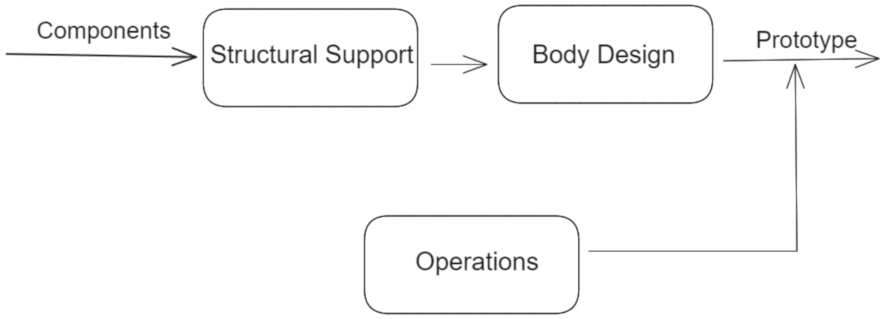


Figure 5.5: Sub-Functions of the System

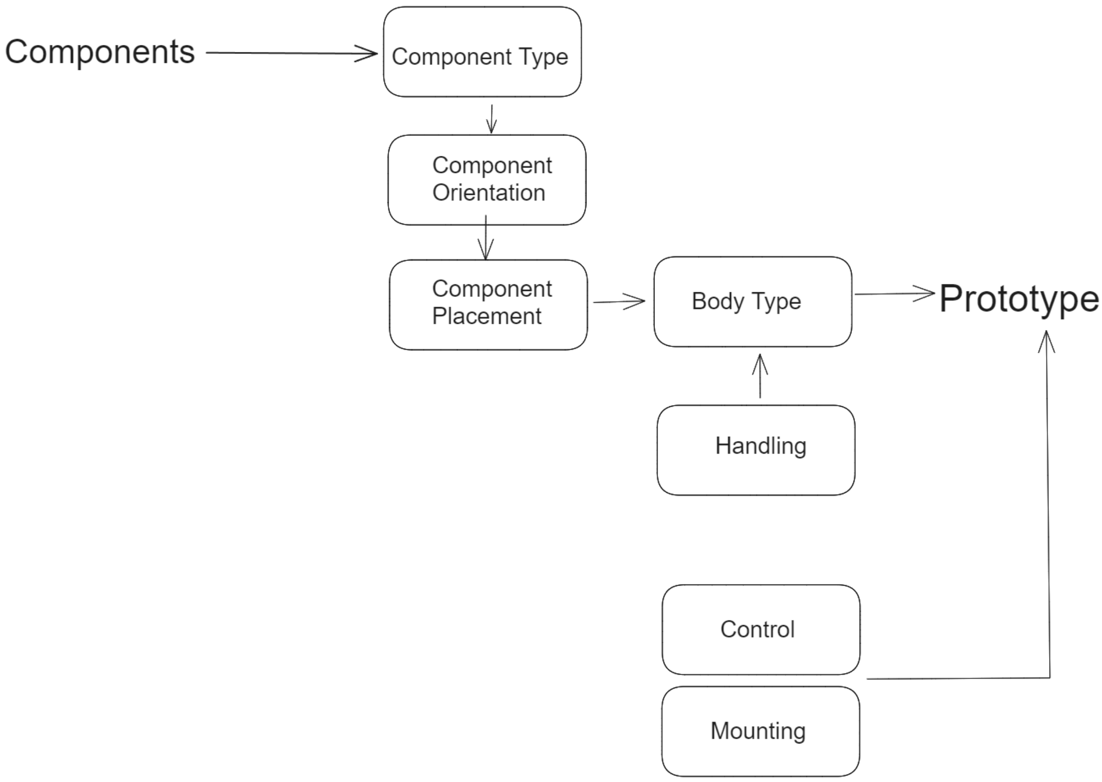


Figure 5.6: Sub-Functions of the System (Final)

tate the identification of suitable solution principles that can fulfill the required functions.

Figure 5.5 illustrates the sub-functions of the prototype. Deriving from the overall function, labeled as *Prototype Design*, it breaks down into three subfunctions, specifically *Structural Support*, *Body Design*, and *Operation*. The function is then further decomposed into more detailed sub-functions in Figure 5.6.

The term *Structural Support* refers to the measures taken to ensure the structural integrity of the prototype. This sub-function encompasses activities such as securing and stabilizing the internal components within the prototype. To simplify the function, it decomposed into three sub-functions: *Component Placement*, which specifies the positioning of internal components; *Component Orientation*, which details the alignment of internal components; and *Component Type*, which is the type of component itself.

Body Design describe the sub-functions involving the prototype's physical structure. To further simplify the task, this sub-function is decomposed into *Body Type*, which defines the outline of the structure, and *Handling*, which describes the handling of the prototype.

Operation deals with how the prototype works. It describes the device's usage and the components involved during operation. This function is then divided into *Control*, which describes the component involved into controlling input from user during operation, and *Mounting*, which refers to the integration of the prototype with the tripod stand.

5.3 Developing Working Principles

As defined by Pahl and Beitz, working principles refer to the physical effects and characteristics that fulfill specific functions of the designed structure [Pah07, 181]. These principles are combined to create the working structure, encompassing physical processes and the form design features. Several potential working structures can be generated by varying the physical effects and form design features, known as the solution field.

In developing working principles, there are multiple available methods in idea generation. These methods are categorized into three groups:

- Conventional methods

- Intuitive methods
- Discursive methods

Pahl and Beitz [Pah07, 18-82] describe the *Conventional Methods* as a systematic and data-driven approach. Designers gather information from various sources, such as literature, trade publications, and competitor catalogs, to stay informed about advancements and best practices. They analyze natural systems and existing technical systems to draw inspiration and identify opportunities for improvement.

On the other hand, the *Intuitive Methods*, as described by them [Pah07, 82-89], tap into creativity and associative thinking. *Brainstorming* fosters a collaborative environment where diverse perspectives generate a wide range of ideas without judgment. *Method 635* adds structure to Brainstorming, allowing for systematic idea development within a group. The *Gallery Method* combines individual work with group discussions, using sketches or drawings to explore solution proposals visually.

Additionally, the *Discursive methods* [Pah07, 89-103] combines systematic, step-by-step procedures with elements of intuition and creativity. They involve deliberate analysis of physical processes, leading to multiple solution variants derived from the relationships between variables. This approach fosters a deeper understanding of the problem space.

5.3.1 Searching for Working Principles

Brainstorming and *Analysis of Existing Technical Systems* are utilized in this work to establish working principles. Table 5.1 shows the result of idea generation. For more detailed sketches of the working principles, please refer to Appendix A.1.

		Working Principles			
		1	2	3	4
Function	Components Arrangement	Tablet-like	Point-of-Service-like	Handheld-PC-like	Camcorder-like
	Screen Orientation	Landscape	Portrait		
	Battery Type	Battery Pack	Power Bank	AAA Batteries with Battery Holder	
	Body Type	Bowl	Skeleton	Sandwich	
	Handling	Body Grip	Bump Grip	Pistol Grip	
	External Mounting	Detachable Plate	Built-in Mounting Plate		
	Control Mechanism	Button	Touch Screen	Trigger	Touch and Button

Table 5.1: Classification Scheme for Working Principles

5.4 Combination of Working Principles

In this step, we will combine the working principles assigned to the sub-functions to create potential working structures. To achieve this, the identified working principles must be linked following the functional structure to fulfill the overall function.

The method we will employ for systematic combination is Zwicky's morphological box [Kus22]. In this approach, the potential principles are represented in a table for better clarity and connected to form functional structures using connecting lines.

Figure 5.7 shows the morphological chart with the generated solution variants. The solution variants are labeled S1 to S8, with each color representing a different solution variant.

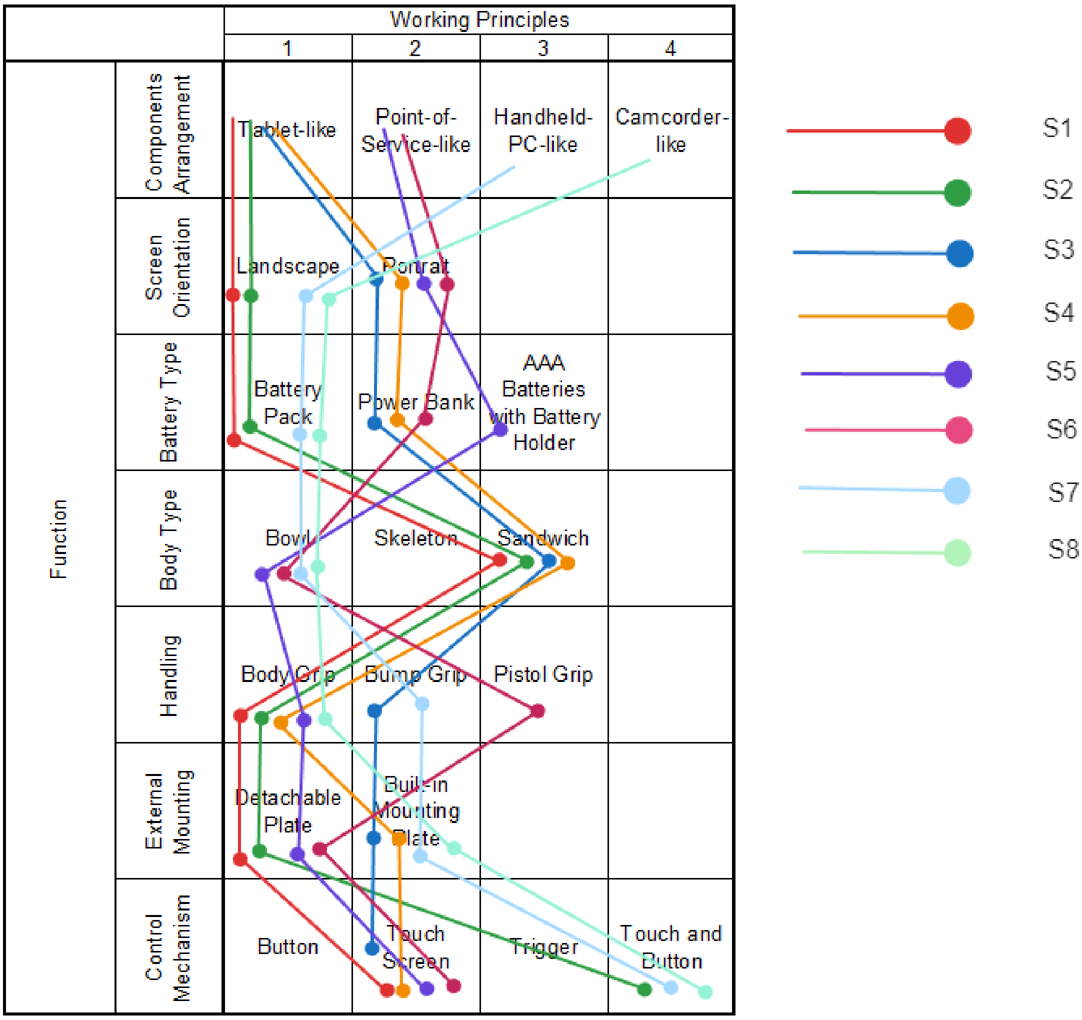


Figure 5.7: Morphological Chart with Solution Variants

5.5 Firming Up into Principle Solution Variants

In this section, we showcase hand-drawn sketches of identified functional structures that have been transformed into practical solution alternatives. Each sketch is accompanied by a brief description of its operations, highlighting its potential strengths and weaknesses.

5.5.1 Solution Variant 1, S1

In Solution Variant 1, we encounter a tablet-like design that closely resembles a typical tablet device. The key components, including the Raspberry Pi, Battery, Camera, and Screen, are arranged in a manner reminiscent of a tablet. The screen orientation is in landscape mode, offering a broader display view for enhanced visual clarity. This orientation is particularly beneficial when the device is used for tasks that require a wider viewing area.

The design is thoughtfully optimized for handheld use, featuring a body grip that ensures comfortable handling. The internal battery integration contributes to a seamless and integrated appearance. A sandwich-type body provides robust protection for the internal components, comprising a top cover, main body, and bottom cover.

For mounting purposes, Solution Variant 1 utilizes a detachable plate tripod system, offering the convenience of easy attachment and removal from a tripod stand. The primary control mechanism for this variant is a touch screen, allowing for intuitive and user-friendly interactions with the device's functionalities.

Figure 5.8 shows the sketch of Solution Variant 1.

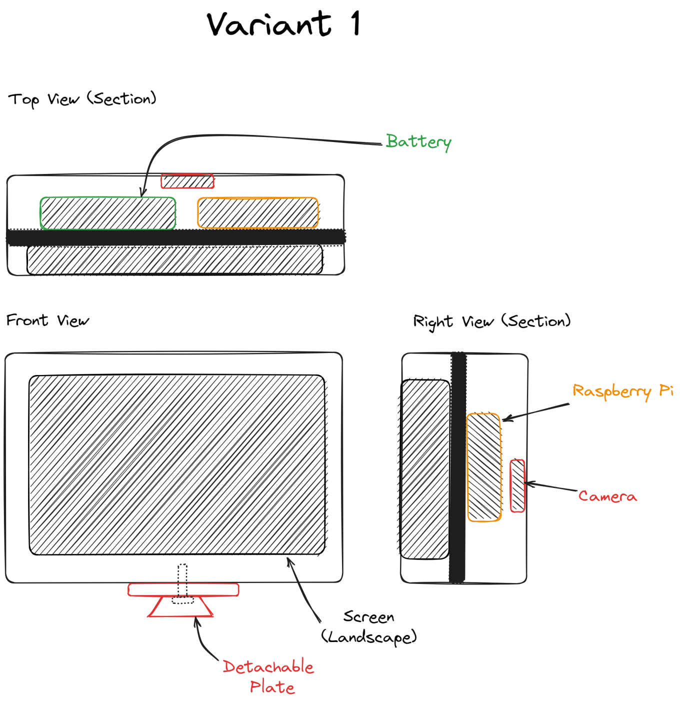


Figure 5.8: Sketch of Solution Variant 1

5.5.2 Solution Variant 2, S2

Similar to previous variant, Solution Variant 2 maintains a tablet-like design, with components arranged like a tablet device (see Figure 5.9). One significant difference lies in the control mechanism. While Solution Variant 1 relies on a touch screen, Solution Variant 2 incorporates physical buttons as the primary control mechanism. This addition enhances versatility and usability in various scenarios, as users can choose between touch-based and tactile input.

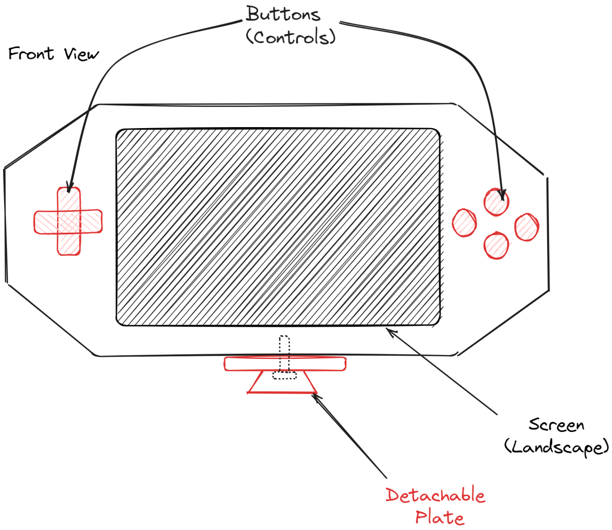


Figure 5.9: Sketch of Solution Variant 2

5.5.3 Solution Variant 3, S3

While Solution Variant 3 maintains the tablet-like component placement found in the previous variants, it introduces a significant change by adopting a portrait screen orientation. This shift opens up new possibilities for the device's usage, particularly in scenarios where vertical screen space is more advantageous than horizontal layout.

The design includes a bump grip for secure and comfortable handling in a vertical position. Notably, the battery is positioned externally in this variant, offering the potential advantage of easier access and replacement. The body structure remains a sandwich-type, providing robust protection for the internal components.

One notable advantage of the portrait screen orientation is the improved stability of the device, as the center of gravity is aligned with the device's center. This alignment enhances balance and control when using the device in various orientations, thus enhancing overall usability and versatility.

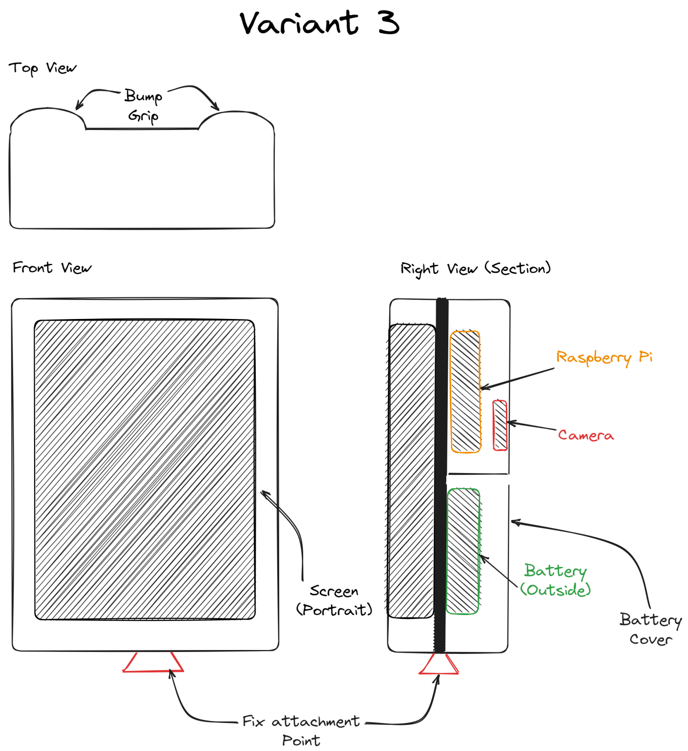


Figure 5.10: Sketch of Solution Variant 3

5.5.4 Solution Variant 4, S4

Solution Variant 4 copies many features from Solution Variant 3, but with one significant change in the body type. Solution Variant 4 opts for a more minimalistic skeleton design, which results in a lightweight yet adequately supportive body for the internal components. This design choice is particularly beneficial for users who prefer a lightweight device for extended usage periods. Figure 5.11 shows the sketch of Solution Variant 4.

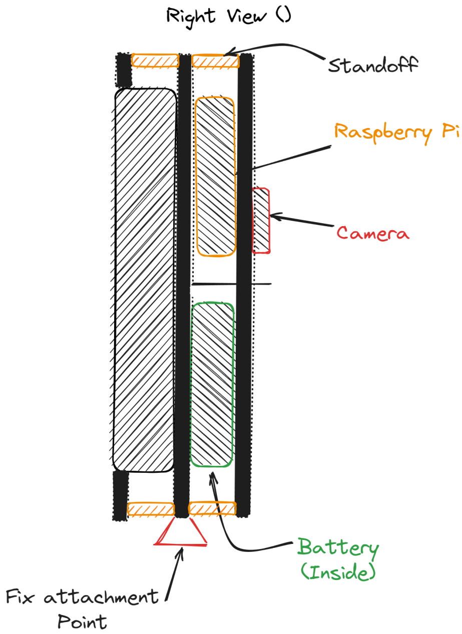


Figure 5.11: Sketch of Solution Variant 4

5.5.5 Solution Variant 5, S5

Solution Variant 5 introduces a unique design approach, deviating from the tablet-like structure seen in previous solutions. Instead, it adopts a Point of Service-like component placement, where the Raspberry Pi, Battery, Camera, and Screen are configured in a distinctive layout. The screen is positioned at an angle, differentiating it from the previous variants (see Figure 5.12).

Regarding screen orientation, Solution Variant 5 retains a portrait mode, which can be advantageous in scenarios requiring vertical displays. The device is designed for body grip handling, offering a secure way to hold and interact with the device.

A notable difference is the external AAA battery setup, which enhances convenience by offering easy battery replacement and compatibility with standard batteries. The body structure follows the familiar sandwich-type design, providing robust protection for the internal components.

For mounting purposes, Solution Variant 5 utilizes the detachable tripod system, enabling seamless attachment and detachment from a tripod stand. Like its predecessors, it relies on a touch screen as the primary control mechanism, ensuring intuitive user interactions.

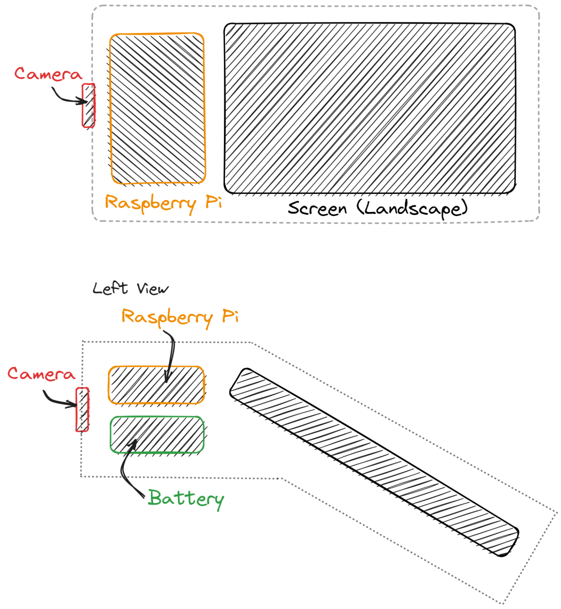


Figure 5.12: Sketch of Solution Variant 5

5.5.6 Solution Variant 6, S6

Solution Variant 6 closely mirrors Solution Variant 5 regarding component placement and screen orientation. This variant, too, adopts the Point of Service-like layout with a portrait screen orientation. However, it introduces a pistol handle for handling, providing a firm and ergonomic grip for users, as can be seen in Figure 5.13.

The battery is positioned externally, offering the same benefits of easy battery replacement and extended usage periods. Regarding body design, Solution Variant 6 employs a bowl-like structure with all components attached to the main body. This design choice provides protection and enclosure while reducing overall weight.

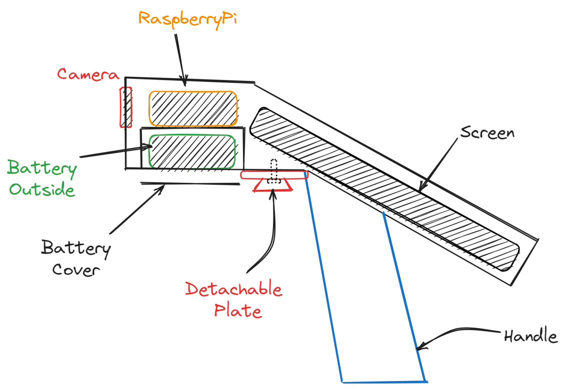


Figure 5.13: Sketch of Solution Variant 6

5.5.7 Solution Variant 7, S7

In Solution Variant 7, a distinct design approach with a Handheld PC-like component placement is produced. This configuration aligns the screen and battery, positioning the Raspberry Pi behind the screen (see Figure 5.14).

This variant is designed with a bump grip for secure and comfortable handling. Notably, the battery is placed internally and utilizes a battery pack for easier charging and replacement.

The chassis structure adopts a bowl-like design, ensuring secure enclosure and protection for all components. The device incorporates a built-in tripod system for mounting, providing a stable attachment to a tripod stand.

Solution Variant 7 combines a touch screen and physical buttons as the control mechanism. This dual-input approach provides users multiple options for interacting with the device's functionalities, enhancing versatility and usability in various scenarios.

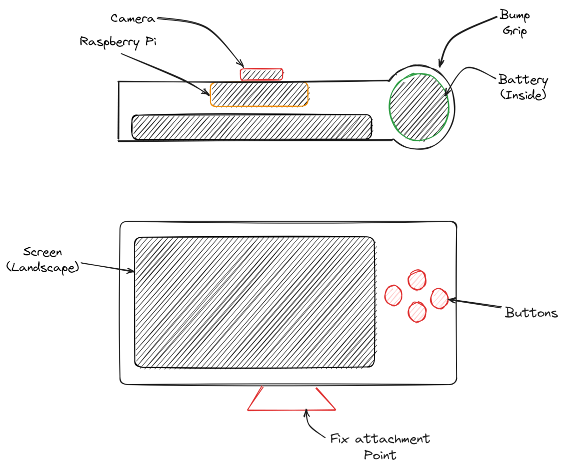


Figure 5.14: Sketch of Solution Variant 7

5.5.8 Solution Variant 8, S8

Solution Variant 8 features a Camcorder-like component placement. The Raspberry Pi, Battery, Camera, and Screen are arranged similarly to a camcorder, with the screen positioned at a hinge, allowing it to change angles for flexible viewing.

The screen orientation remains landscape, providing a broad horizontal display view. The device is designed with a body grip for secure and comfortable handling. The battery is placed internally, and a power bank is used to provide a reliable power source.

The chassis structure follows a bowl-like design, offering protection and sturdiness for the internal components. A fixed-mount tripod system is employed for mounting purposes, providing stability and ease of use when attaching the device to a tripod stand.

As with some of the previous variants, Solution Variant 8 combines both a touch screen and physical buttons as the control mechanism, offering users the flexibility to interact.

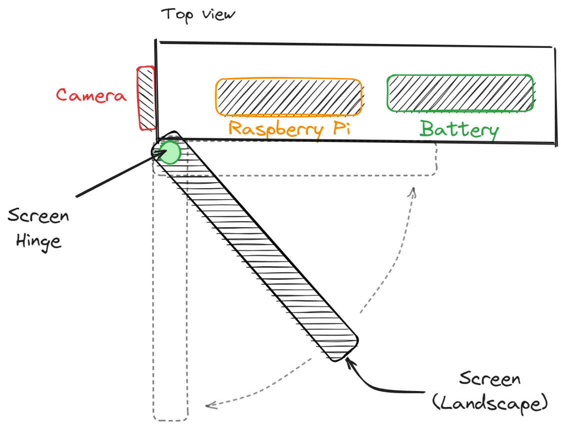


Figure 5.15: Sketch of Solution Variant 8

5.6 Filtering of Solution Variants

As shown in Figure 5.7, multiple solution variants were generated. However, not all of these solutions are feasible and practical. As advised by Pahl and Beitz [Pah07, 106-107], it is necessary to reduce the vast number of theoretically possible but practically unachievable solutions as early as possible. However, caution should be exercised not to discard valuable working principles, as they often play a crucial role in forming a favorable and effective working structure when combined with others.

Additionally, Pahl and Beitz [Pah07, 107] suggest a method that can be used to filter the solution variants. This method is known as the selection chart, which consists of two steps: elimination and preference. Initially, all clearly unsuitable proposals are removed. If a substantial number of solutions remain, preference is given to those who stand out as markedly superior. Only these preferred solutions are evaluated during the final stages of the conceptual design phase.

Pahl and Beitz suggest the following criteria for eliminating unsuitable solutions:

- **Criteria A:** Compatible with the overall task
- **Criteria B:** Fulfill demands of requirement list

- **Criteria C:** Realisable in principle
- **Criteria D:** Within permissible cost

These criteria are applied step by step to examine each solution. If any of the exclusion criteria are not met, the solution is rejected, and further criteria are not assessed. Additionally to the exclusion criteria, the following preference criteria are used to prioritize the remaining solutions:

- **Criteria E:** Incorporates direct safety measures
- **Criteria F:** Preferred by the designer

Criteria E and F are then used to prioritize solutions if there are still too many options after the initial screening. The remarks column provides explanations for excluding or favoring each solution. The final assessment of the functional principles is recorded in the rightmost column of the selection list.

Page 1	Selection Chart								
		Evaluate solution variants according to selection criteria						Decision	
								Remarks: (+) Pursue Solution; (-) Eliminate Solution; (?) More Information Required; (!) Check Specification	
Solutions Variant	No.	Compatible with the overall task	ulfill demands of requirement list	Realisible in principle	Within permissible costs	Incorporates direct safety measures	Preferred by the designer		
		A	B	C	D	E	F		
S1	1	+	+	+	+	?	+	Might have problem with ergonomic	-
S2	2	+	+	+	+	?	?		+
S3	3	+	+	+	+	?	+		+
S4	4	-	+	+	+	+	+	Have almost no protection of inner components	-
S5	5	+	+	+	+	?	+	Less ergonomics due to wide body	-
S6	6	+	+	+	+	?	?		+
S7	7	+	+	+	+	+	+		+
S8	8	+	+	-	?	?	-	Too complex	-

Figure 5.16: Selection Chart for Solution Variants

The result of the selection chart, as shown in Figure 5.16, indicates that solutions S1, S4, S5, and S8 have been eliminated and will not be considered for the next stage of the design process.

6 Embodiment Design

The next phase in the design methodology is embodiment design. This phase, as defined by Pahl and Beitz [Pah07, 227], involves starting with the fundamental solution or concept for a technical product and then advancing the design in alignment with technical and economic criteria, taking into account further information. The ultimate objective is to reach a stage where the subsequent detailed design can smoothly progress into the production phase. Figure 6.1 shows the steps involved in this phase.

6.1 Basic Rules of Embodiment Design

Regarding product design, some basic rules must be followed. As defined by Pahl and Beitz [Pah07, 234-235], they include clarity, simplicity, and safety. Neglecting these rules can potentially result in issues and accidents. Subsequent sections will provide a comprehensive exploration of these guidelines.

6.1.1 Clarity

Clarity, as described by Pahl and Beitz [Pah07, 235], includes establishing clear and unambiguous connections within a design, ensuring straightforward relationships between subfunctions, inputs, and outputs to prevent confusion or misinterpretation.

They also mention that clarity applies to the broader design structure, whether it involves multiple working principles or component combinations. It mandates that the design facilitates the orderly flow of energy, materials, and sig-

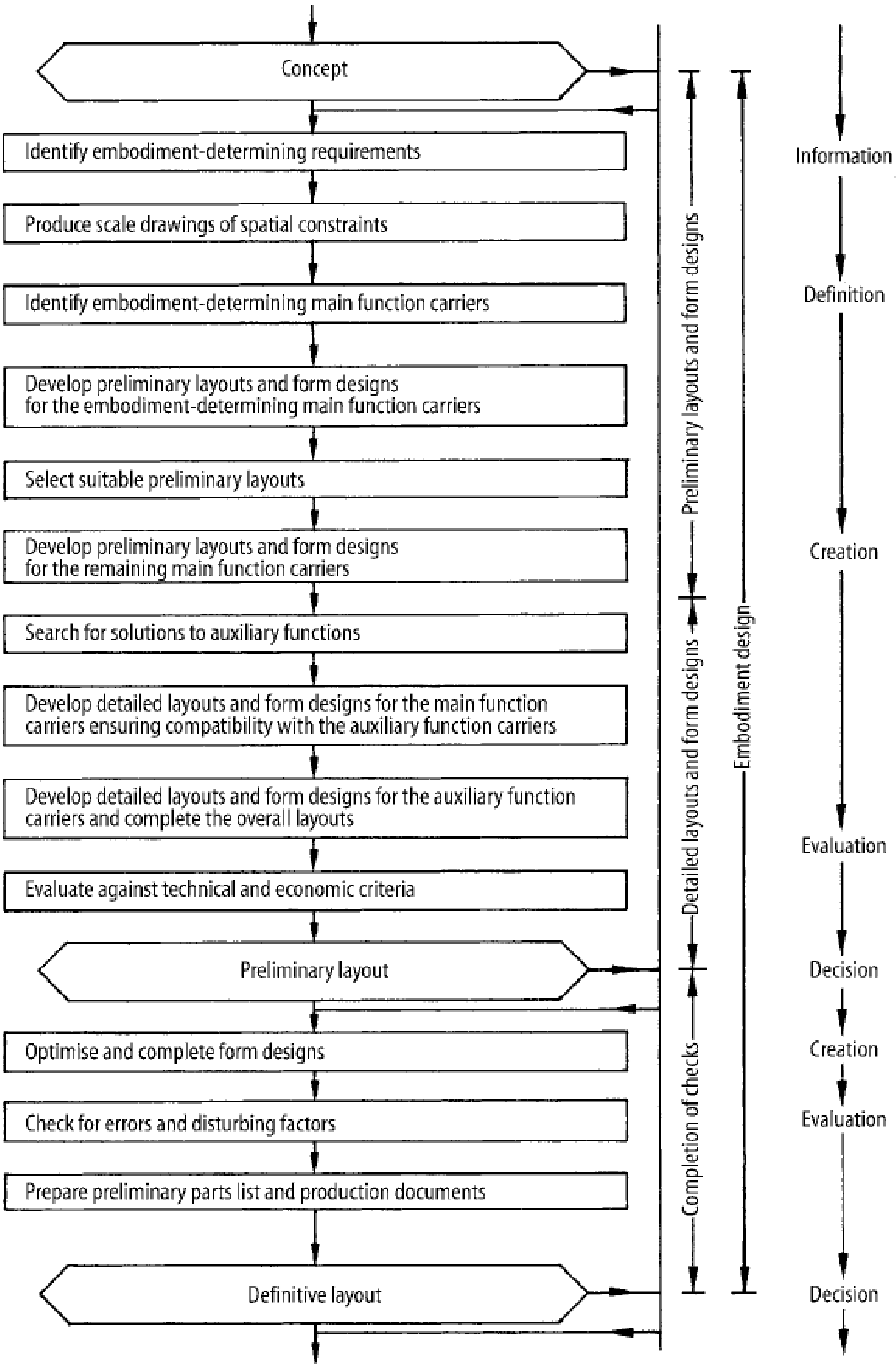


Figure 6.1: Steps in Embodiment Design [Pah07, 229]

nals, preventing adverse effects like excessive forces or wear.

6.1.2 Simplicity

As defined by Pahl and Beitz [Pah07, 242], simplicity in design is characterized by an uncomplicated and easily understandable approach, often achievable by using fewer components. Such simplicity can save costs, reduce wear and tear, and minimize maintenance requirements. Nonetheless, striking a balance is crucial, as certain functions inherently demand a minimum number of components.

6.1.3 Safety

According to Pahl and Beitz [Pah07, 247], safety considerations are crucial in ensuring both the adequate performance of technical functions and the protection of people and the environment. Designers rely on a safety methodology outlined in the German industry standard DIN 31 000, encompassing three levels: direct safety, indirect safety, and warnings. Designers should prioritize direct safety measures, seeking solutions that inherently eliminate potential dangers. Only when this is not feasible should they resort to indirect safety measures involving the construction of specialized protective systems.

Warnings highlighting dangers and hazard zones are best utilized in conjunction with direct and indirect safety measures, clarifying specific risks. As designers address technical challenges, they encounter various constraints, not all of which can be simultaneously overcome. However, their objective remains to develop solutions that come as close as possible to meeting all requirements. It is important to note that exceptionally high safety demands can complicate design, potentially diminishing clarity and economic viability and possibly leading to project abandonment.

6.2 Guideline of Embodiment Design

In addition to the basic rules of embodiment design, Pahl and Beitz [Pah07, 308] also stress the importance of following a set of design guidelines to help designers meet specific requirements and constraints. For this project, the *Design for Production* guideline are applied.

6.2.1 Design for Production

The concept of *Design for Production* outlined by Pahl and Beitz [Pah07, 355-356] underscores the significance of factoring in the production process during the design stage. This methodology empowers designers to fine-tune production costs and timelines while maintaining the product's functionality and quality. They highlight that adhering to fundamental principles of clarity and simplicity sets designers on the correct path towards realizing this objective.

Appropriate Overall Layout Design

Pahl and Beitz [Pah07, 355-362] mentioned that the overall layout design, derived from the function structure, influences product division into assemblies and components, including sourcing decisions (in-house, bought-out, standard parts), production procedures, dimensions, batch sizes, joining methods, and quality control. The layout can lead to differential, integral, composite, or building-block construction methods.

Differential Construction involves breaking down components into quickly produced parts, facilitating adaptability, increased component batch sizes, and more accessible quality assurance. However, it demands more outstanding machining and assembly costs and may have functional limitations due to joints.

Integral Construction combines multiple parts into a single component, usually utilized for product optimization. It can greatly reduce cost for assembly and quality control while enhancing functionality and performance. However it usually requires more complex production procedures and may be less adaptable.

Composite Construction refers to the integration of various components, each constructed differently, into a single unit that may require additional processing. This includes scenarios like combining cast and forged parts. Moreover, it involves employing multiple methods of joining to bring together different elements simultaneously.

Building Block Construction results from splitting components so that the parts or assemblies can be used in other products or variants, offering flexibility and cost savings. This approach is often used in modular design, where components are designed to be easily interchangeable.

Appropriate Form Design of Components

As mentioned by Pahl and Beitz [Pah07, 362], numerous factors influence the cost, time, and production quality. These include parameters like shapes, dimensions, surface finishes, tolerances, and fits. These choices are essential in determining production procedures, machine types, materials, in-house vs. bought-out components, and quality control measures.

Furthermore, production facilities can impact the design of features, such as dimension limitations that necessitate component division or the procurement of bought-out components. Many guidelines are available for designing appropriate component forms. The design guideline for 3D printed components is shown in Figure 6.2.

Appropriate Use of Standard and Bought-Out Components

The authors advised that [Pah07, 374-375], designers should use readily available standard or bought-out components rather than specially produced ones to ensure favorable supply and storage conditions. Bought-out parts are often more cost-effective than in-house production. The decision between in-house or bought-out components depends on factors like production volume, market demand, costs, and available facilities. These factors may change over time, requiring periodic re-evaluation, especially for unique or batch products in heavy engineering.

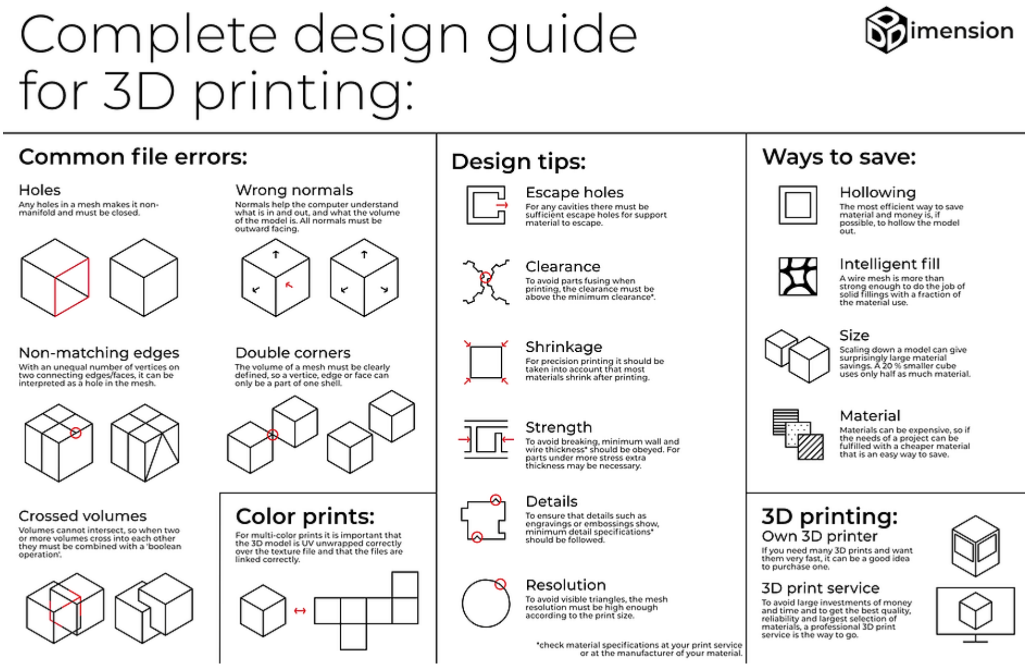


Figure 6.2: Design guidelines for 3D printing [And22]

6.3 Preliminary Design

In this section, we will explore multiple designs for the device. These designs are detailed 3D models of the device that we will use to evaluate their respective designs and assess their feasibility. Each of these preliminary designs will be based on the selected solution from the previous phase. Alongside the models, we will also present the production costs for each of these designs. For a more detailed breakdown of the production costs, please refer to Appendix A.4.

6.3.1 Preliminary Design Variant 2

Figure 6.3 showcases the 3D model of Variant 2, while Figure 6.4 provides various perspectives and body measurements of the device. The key emphasis of this design is its ergonomic shape and user-friendly attributes. With a thickness of 52.2 mm (Figure 6.4b), it successfully balances being slim and accommodating essential components for optimal performance.

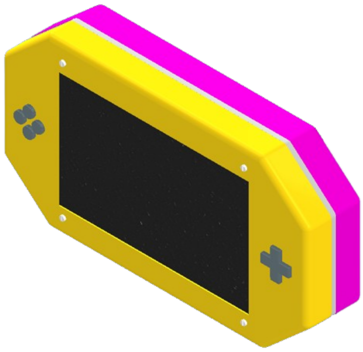
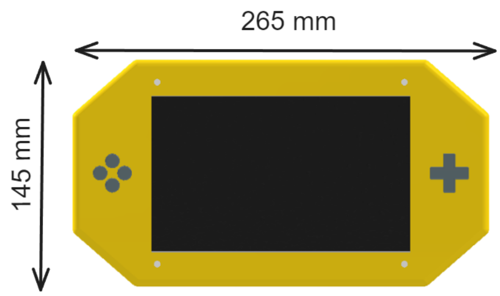


Figure 6.3: Preliminary Design Variant 2



(a) Front View

(b) Right View

Figure 6.4: Views of Preliminary Design Variant 2

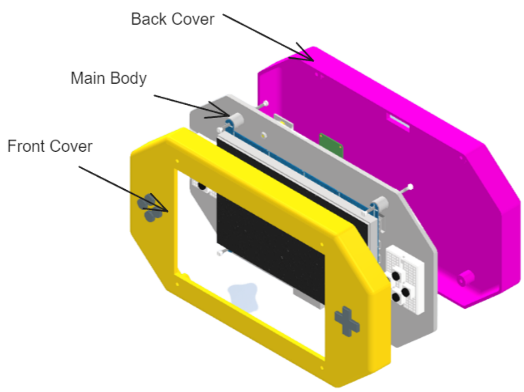
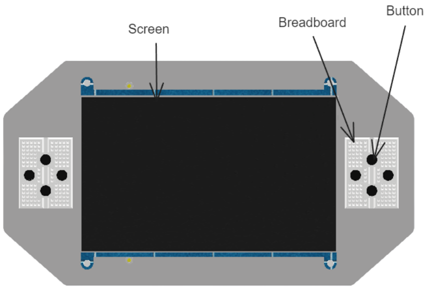
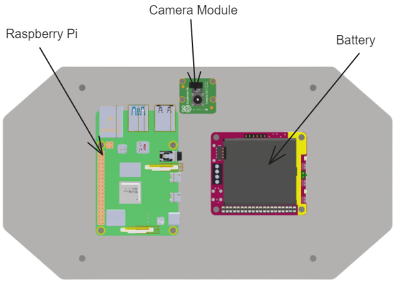


Figure 6.5: Body Components of Preliminary Design Variant 2

The physical design of Solution Variant 2 adheres to a sandwich-like structure comprising a main body, top cover, and back cover (see Figure 6.5). This design



(a) Front View



(b) Back View

Figure 6.6: Placement of inner components for Variant 2

choice ensures the protection of internal components and simplifies assembly and maintenance. The main body of the device functions as the central hub, accommodating the internal components and features. In contrast, the top and back covers act as protective shields, safeguarding the internal parts from any damage from external factors.

A crucial aspect of the design involves arranging internal components within the device. Following a tablet-like configuration, the main LCD is positioned on the front side of the main body, providing users with a straightforward and interactive interface (see Figure 6.6a). Simultaneously, the camera, Raspberry Pi, and battery were placed on the back side of the body (Figure 6.6b) to optimize the weight distribution and ensure a well-balanced user experience. This arrangement enhances the overall usability and convenience of the device, making it suitable for a wide range of applications.

Ensuring the secure attachment of internal components to the main body is vital in the design process. Various methods of component fastening have been con-

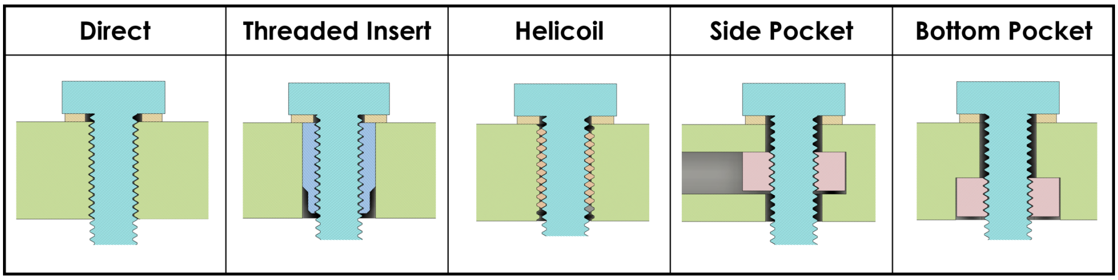


Figure 6.7: Methods to secure 3D-printed components [Her20]

sidered, including direct attachment, threaded inserts, helicoils, side pockets, and bottom pockets, as shown in Figure 6.7.

The most straightforward approach is direct attachment, in which threads are designed into a 3D-printed part to allow components to be screwed in. For more robust connections, threaded inserts can be used by designing holes in the 3D printed part and installing inserts appropriately.

Helicoils offer durable threaded holes by inserting coil-shaped inserts into the holes. Side and bottom pockets create cavities or slots in the 3D-printed part to securely hold components. Each method has its advantages and challenges. After careful evaluation, the variant opts for threaded inserts due to its simplicity and robustness.

Solution Variant 2 employs a hybrid input method combining a touch screen and physical buttons. The touch screen is oriented in landscape mode, while the buttons are positioned on either side of the screen (Figure 6.4a). HDMI and USB connections were established between the touch screen and Raspberry Pi to enable the integration of the touch screen. Additionally, to facilitate the functionality of the physical buttons, they are connected to Raspberry Pi using general-purpose input/output (GPIO) pins.

In Figure 6.8, we observe a quick-release plate designed to be affixed to the tripod stand. To enhance stability during usage, Solution Variant 2 can utilize a quick-release plate, which can be conveniently mounted on a tripod stand.

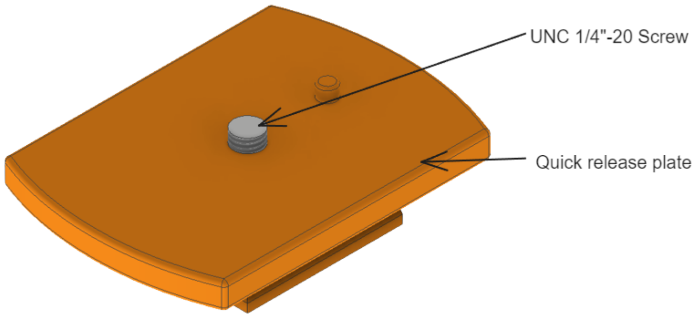


Figure 6.8: Quick release plate

Cost Calculation

Table 6.1 shows the printing cost for each component of this variant, while the total manufacturing cost is shown in Table 6.2. In this calculation, the cost for screen, camera, and Raspberry Pi are not included, as they are used by all variants. The exclusion of these components allows for a more accurate comparison of the manufacturing costs between the variants.

Part Name	Weight of PLA used (g)	Printing Time (h)	Material Cost	Energy Cost	Total Cost
Base	175.07	5.13	5.25 €	0.10 €	5.35 €
Top Cover	105.85	3.32	3.17 €	0.06 €	3.24 €
Back Cover 1	86.73	2.57	2.60 €	0.05 €	2.65 €
Back Cover 2	86.60	2.53	2.60 €	0.05 €	2.64 €
Total	**454.25**	**13.55**	**13.62 €**	**0.26 €**	**13.88 €**

Table 6.1: Printing cost for Variant 2

Parts Name	Amount	Price	Remarks	Total Price
Printing Cost	1	13.90 €	Calculated	13.90 €
Screw M2x10mm (DIN 84)	4	0.05 €	Wuerth	0.20 €
Screw M2.5x10mm (DIN 84)	8	0.05 €	Wuerth	0.40 €
Screw M2.5x18mm (DIN 84)	8	0.11 €	Wuerth	0.88 €
Screw M3x10mm (DIN 84)	2	0.05 €	Wuerth	0.10 €
Nut M2 (DIN 934)	4	0.04 €	Wuerth	0.15 €
Standoff M2	4	0.17 €	Amazon	0.67 €
Threaded Inserts M2.5	16	0.50 €	Ruthex	7.99 €
Threaded Inserts 1/4"	1	0.00 €	Ruthex	0.00 €
Breadboard	2	0.04 €	Amazon	0.08 €
Tactile Button	8	0.00 €	Amazon	0.00 €
Female MicroUsb	1	4.95 €	Adafruit	4.95 €
PiJuice	1	45.49 €	Rasppishop	45.49 €
Total				**74.79 €**

Table 6.2: Manufacturing cost for Variant 2

6.3.2 Preliminary Design Variant 3

Variant 3 maintains a similar component arrangement to variant 2, with the screen at the front and the camera, Raspberry Pi, and battery at the rear, as shown in Figure 6.10. However, Variant 3 introduced significant changes with different screen orientations and battery types.

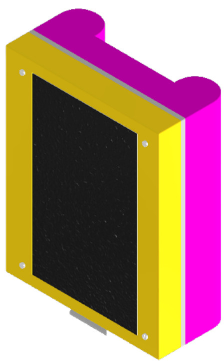
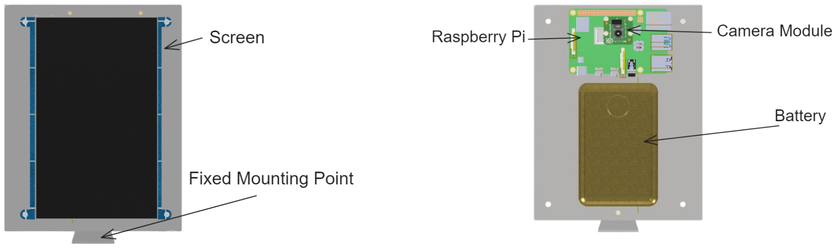


Figure 6.9: Preliminary Design Variant 3

A noteworthy alteration is the inclusion of bumps on the back cover to enhance the ergonomics (Figure 6.11). This adjustment aims to provide a more comfortable grip, improve user engagement, and extend usability. In addition, the tactile bump is added to increase handling stability of the device, ensuring that



(a) Front View

(b) Back View

Figure 6.10: Placement of inner components for Variant 3

the device fits snugly in the user's hand, further enhancing the overall user experience.

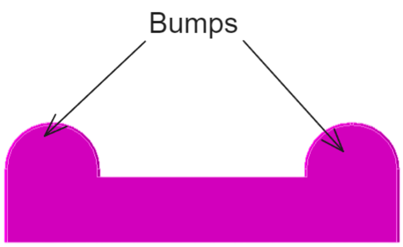


Figure 6.11: Bumps on the back cover

Variant 3 shifted from the standard battery position in variant 2, with a more noticeable difference in battery placement. Figure 6.12 illustrates a designated slot within the back cover, strategically designed to house a power bank outside the body. This configuration enhances the operational stability and simplifies the battery replacement process.

The input methodology was streamlined using a touch screen as the sole interface. This approach simplifies the user experience by eliminating the need for physical buttons and seamlessly integrating screen interactions. Additional information regarding integrating the touchscreen with the Raspberry Pi is provided in Section 6.3.1.

Figure 6.10a shows the position of the mounting point of on the main body in Variant 3. This strategic design allows the mounting point to serve as a sturdy

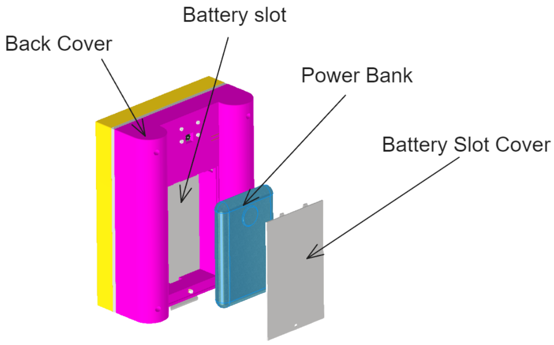


Figure 6.12: Battery Placement

anchor for the device when used in a tripod stand.

Cost Calculation

Table 6.3 shows the printing cost for each component of this variant, while the total manufacturing cost is shown in Table 6.4.

Part Name	Weight of PLA used (g)	Printing Time (h)	Material Cost	Energy Cost	Total Cost
Base	222.96	6.67	6.69 €	0.13 €	6.81 €
Top Cover	71.02	2.08	2.13 €	0.04 €	2.17 €
Back Cover	303.02	9.37	9.09 €	0.18 €	9.26 €
Battery Cover	17.43	0.53	0.52 €	0.01 €	0.53 €
Total	**614.43**	**18.65**	**18.43 €**	**0.35 €**	**18.78 €**

Table 6.3: Printing cost for Variant 3

Parts Name	Amount	Price	Remarks	Total Price
Printing Cost	1	18.78 €	Calculated	18.78 €
Screw M2x10mm (DIN 84)	4	0.05 €	Wuerth	0.20 €
Screw M2.5x10mm (DIN 84)	4	0.05 €	Wuerth	0.20 €
Screw M2.5x18mm (DIN 84)	8	0.11 €	Wuerth	0.88 €
Nut M2 (DIN 934)	4	0.04 €	Wuerth	0.15 €
Threaded Inserts M2.5	12	0.13 €	Ruthex	1.54 €
Veektomx	1	25.99 €	Amazon	25.99 €
Total				**48.08 €**

Table 6.4: Manufacturing cost for Variant 3

6.3.3 Preliminary Design Variant 6

Figure 6.13 provides a detailed and comprehensive overview of the initial design concept of Variant 6. This version stands out by organizing its internal components into a configuration resembling a point-of-service (POS) system. This change from previous iterations purposefully positions the screen at an inclined angle, enhancing user interaction and optimizing the screen visibility (Figure 6.14).

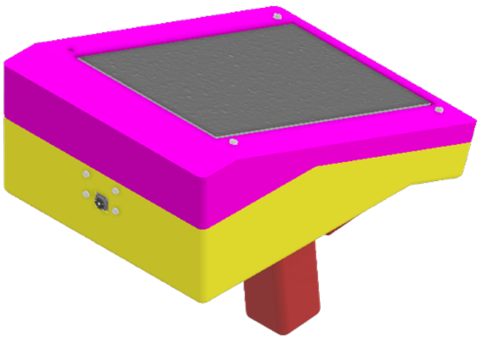


Figure 6.13: Preliminary Design Variant 6

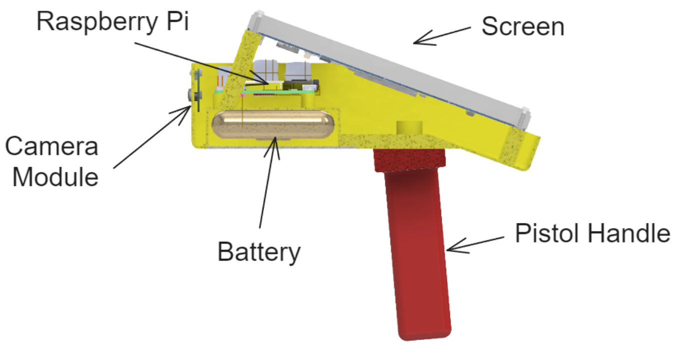


Figure 6.14: Placement of inner components for Variant 3

Figure 6.15 demonstrates the handle grip design, while Figure 6.16a illustrates its placement on the main body. This ergonomic addition ensures a secure and comfortable hold during operation. Additionally, the handling of the device can be easily switched between the quick-change plate and the handle grip, providing users with flexible options for different scenarios (see Figure 6.16b).

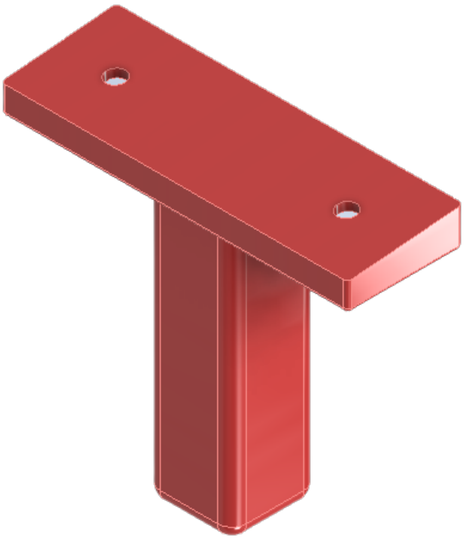
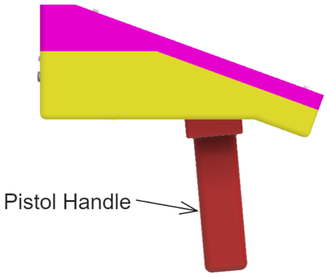
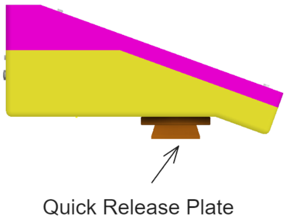


Figure 6.15: Handle Grip



(a) Handle Grip with Main Body



(b) Quick Release Plate with Main Body

Figure 6.16: Placement of handle grip and quick release plate

This variant boasts the same input method and battery placement as Variant 3. Please refer to Section 6.3.2 for a comprehensive explanation.

Cost Calculation

Table 6.5 shows the printing cost for each component of this variant, while the total manufacturing cost is shown in Table 6.6.

Part Name	Weight of PLA used (g)	Printing Time (h)	Material Cost	Energy Cost	Total Cost
Top Cover	55.35	1.87	1.66 €	0.04 €	1.70 €
Main Body	240.25	8.88	7.21 €	0.17 €	7.37 €
Battery Cover	22.21	0.67	0.67 €	0.01 €	0.68 €
Handle Pistol	57.30	1.73	1.72 €	0.03 €	1.75 €
Total	**375.11**	**13.15**	**11.25 €**	**0.25 €**	**11.50 €**

Table 6.5: Printing cost for Variant 6

Parts Name	Amount	Price	Remarks	Total Price
Printing Cost	1	11.50 €	Calculated	11.50 €
Screw M2x10mm (DIN 84)	4	0.05 €	Wuerth	0.20 €
Screw M2.5x10mm (DIN 84)	5	0.05 €	Wuerth	0.25 €
Screw M2.5x18mm (DIN 84)	4	0.11 €	Wuerth	0.44 €
Nut M2 (DIN 934)	4	0.04 €	Wuerth	0.15 €
Threaded Inserts M2.5	8	0.13 €	Ruthex	1.03 €
Threaded Inserts 1/4"	3	0.50 €	Ruthex	1.50 €
1/4" Schraube (Camera)	2	1.40 €	Amazon	2.80 €
Veektomx	1	25.99 €	Amazon	25.99 €
Total				**43.84 €**

Table 6.6: Manufacturing cost for Variant 6

6.3.4 Preliminary Design Variant 7

In Figure 6.17, we can observe a 3D model for variant 7, which draws inspiration from handheld PCs. The design showcases the integration of Raspberry Pi at the back of the screen, creating a compact and unified structure, while the battery is positioned next to the screen.

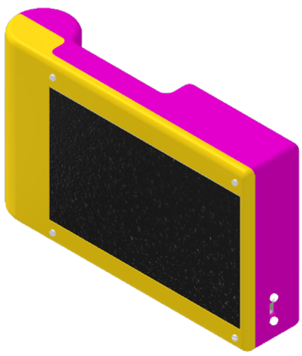


Figure 6.17: Preliminary Design Variant 7

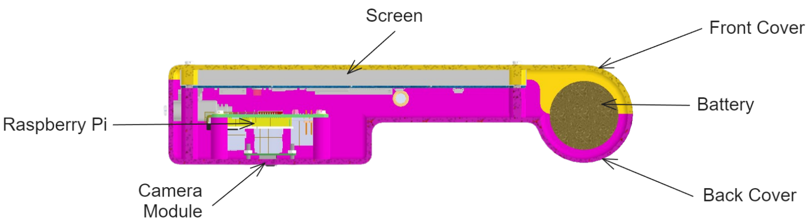


Figure 6.18: Placement of inner components for Variant 7

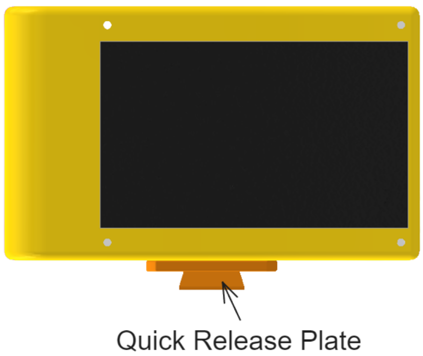


Figure 6.19: Placement of quick release plate

This design includes a bump on the side, enhancing ergonomics as shown in Figure 6.18. As with variants 2 and 6, this variant employs a quick-release plate, facilitating integration with a tripod stand. The placement of the plate can be observed in Figure 6.19.

Cost Calculation

Table 6.7 shows the printing cost for each component of this variant, while the total manufacturing cost is shown in Table 6.8.

Part Name	Weight of PLA used (g)	Printing Time (h)	Material Cost	Energy Cost	Total Cost
Top Cover	61.50	2.08	1.84 €	0.04 €	1.88 €
Back Cover	380.70	11.82	11.42 €	0.22 €	11.64 €
Total	**442.20**	**13.90**	**13.26 €**	**0.26 €**	**13.52 €**

Table 6.7: Printing cost for Variant 7

Parts Name	Amount	Price	Remarks	Total Price
Printing Cost	1	13.52 €	Calculated	13.52 €
Screw M2x10mm (DIN 84)	4	0.05 €	Wuerth	0.20 €
Screw M2.5x10mm (DIN 84)	4	0.05 €	Wuerth	0.20 €
Screw M2.5x18mm (DIN 84)	4	0.11 €	Wuerth	0.44 €
Screw M3x10mm (DIN 84)	2	0.05 €	Wuerth	0.10 €
Nut M2 (DIN 934)	4	0.04 €	Wuerth	0.15 €
Threaded Inserts M2.5	8	0.13 €	Ruthex	1.03 €
Threaded Inserts 1/4"	1	0.50 €	Ruthex	0.50 €
Cylinder Power Bank	1	19.99 €	Amazon	19.99 €
Female MicroUsb	1	4.95 €	Adafruit	4.95 €
Total				**41.07 €**

Table 6.8: Manufacturing cost for Variant 7

6.4 Evaluation with VDI 2225

This section will evaluate the preliminary design variants using the guideline VDI 2225 [Pah07, 109-110]. This guideline is a comprehensive framework for evaluating technical solutions based on a balanced consideration of various aspects.

The guideline advocates for comprehensive evaluation methods covering task-specific requirements and general constraints. These methods aim to quantify and qualitatively assess the properties of different variants, even in the early conceptual phase where information is limited.

The evaluation process, as discussed by Pahl and Beitz [Pah07, 110-124], involves several key steps:

Identifying Evaluation Criteria

This initial step involves defining a set of objectives from which specific evaluation criteria can be derived. These objectives should comprehensively cover decision-relevant requirements and general constraints, ensuring no crucial criteria are overlooked. The objectives should also be as independent as possible and expressed in quantitative or qualitative terms.

The following are the evaluation criteria for the preliminary design variants:

Weight Distribution: The weight distribution is evaluated by measuring the distance between the center of gravity and the point of handling. The point of handling is defined as the point where the user holds the device. A lower distance signifies a more balanced weight distribution, enhancing the user experience. The position of the center of gravity is retrieved from Computer-Aided Design (CAD) models through detailed analysis of the device's structural layout and component placement.

Device Weight: Device weight evaluates the overall heaviness of the equipment. A lighter device is generally easier to handle and transport, reducing user fatigue and enabling greater mobility while maintaining performance and durability. The value for device weight is calculated from CAD models by summing the individual weights of all components, materials, and structural elements that constitute the device.

Device Size: The size criterion considers the device's physical dimensions, assessing its compactness and portability. An optimal device size allows for convenient storage, transportation, and operation in various environments without compromising functionality. The evaluation of device size involves measuring key dimensions such as length, width, height, and any protrusions or extensions.

Ease of Assembly: This criterion evaluates the ease of assembling and disassembling the device. Quick and easy assembly and disassembly saves time and increases user convenience, reducing the risk of errors. Evaluation is done by counting the components used in assembly and disassembly. Fewer components often mean a more straightforward and more user-friendly design. The type and number of fasteners, such as screws or connectors, needed for assembly are also considered.

Swappable Parts: Swappable components refer to the ease with which parts can be interchanged or substituted. This design enhances flexibility, maintenance, and adaptability. The presence of swappable parts encourages component modularity, enabling streamlined repairs and upgrades. Swappable parts are assessed based on the quantity of interchangeable components and their compatibility. A higher number of swappable parts signifies a design that supports versatility and minimizes downtime for maintenance or repairs.

Weighting Evaluation Criteria

After establishing the evaluation criteria, their relative importance is assessed through weighting factors (w). This step is crucial in eliminating less significant criteria before the actual evaluation. Weightings should reflect the relative importance of each evaluation criterion.

Guideline VDI 2225 aims to avoid weightings and instead relies on criteria of roughly equal importance. However, weightings (like 2x or 3x) are used when there are significant differences between criteria. Table 6.9 shows the weighting factors for the evaluation criteria.

Criteria	**Weighting Factor, w**
Weight Distribution	3x
Device Weight	2x
Device Size	1x
Ease of Assembly	1x
Swappable Parts	2x

Table 6.9: Weighting Factors for Evaluation Criteria

Assessing Values

This step involves assigning values (v_{ij}) to the variants based on the relative scale of the determined parameters. Guideline VDI 2225 suggests using a range from 0 to 4. Table 6.10 shows the scale used to evaluate the preliminary design variants. Tables 6.11 to 6.15 show the value scales for the individual evaluation criteria. Equation 6.1 shows the formula used to calculate the weighted value (wv_{ij}) for each variant.

Points, v_{ij}	Meaning
0	unsatisfactory
1	just tolerable
2	adequate
3	good
4	very good (ideal)

Table 6.10: Value Scale for Evaluation [Pah07, 115]

Weight Distribution	
Range, mm	**Point, v_{ij}**
0-10	4
10-50	3
50-100	2
100-150	1
$\geq$ 150	0

Table 6.11: Value Scale for Weight Distribution

Device Weight	
Range, g	**Point, v_{ij}**
0-500	4
500-1000	3
1000-1500	2
1500-2000	1
$\geq$ 2000	0

Table 6.12: Value Scale for Device Weight

Device Size	
Range, mm	**Point, v_{ij}**
0-100	4
100-200	3
200-300	2
300-400	1
$\geq$ 400	0

Table 6.13: Value Scale for Device Size

Ease of Assembly	
Range	**Point, v_{ij}**
0-25	4
25-50	3
50-75	2
75-100	1
≥ 100	0

Table 6.14: Value Scale for Ease of Assembly

Swappable Parts	
Range	**Point, v_{ij}**
≥ 4	4
3	3
2	2
1	1
0	0

Table 6.15: Value Scale for Swappable Parts

$$wv_{ij} = w_i \cdot v_{ij} \tag{6.1}$$

Determining the Overall Value

The overall value of each variant (OWV_j) is calculated by summing the weighted values (wv_{ij}) of all evaluation criteria (see Equation 6.2).

$$OWV_j = \sum_{i=1}^{n} wv_{ij} \tag{6.2}$$

Comparing Concept Variants

With the overall values (OWV_j) of the concept variants, the variants can be compared and evaluated based on their rating (R) which is calculated using Equation 6.3. The technical rating (R_t) is calculated using Equation 6.4, where v_{max}

is the maximum value of the value scale, and n is the number of evaluation criteria.

The economic rating (R_e) is calculated using Equation 6.5, where C_o is the comparative cost, and $C_{variant}$ is the cost of the variant. For this project, C_o is set to be 60% of the cost of the least expensive variant (see Equation 6.6).

The best variant is determined by comparing each variant's total rating (R). The variant with the highest total rating is considered the best variant.

$$R = \frac{R_t + R_e}{2} \tag{6.3}$$

$$R_t = \frac{OWV_j}{v_{max} \cdot \sum_{i=1}^{n} w_i} \tag{6.4}$$

$$R_e = \frac{C_o}{C_{variant}} \tag{6.5}$$

$$C_o = 0.6 \cdot C_{minimum} \tag{6.6}$$

6.4.1 Evaluation of Preliminary Design Variant

The result of the evaluation of the preliminary design variants will be presented in this section. Table 6.16 and Table 6.17 shows the technical evaluation of the preliminary design variants, while Table 6.18 shows the economic evaluation of the variants. The total rating of the variants is shown in Table 6.19. Appendix A.5 shows the detailed calculation of the evaluation.

Based on the total rating, Variant 6 is the best variant, followed by Variant 7, Variant 3, and Variant 2.

Evaluation criteria					Variant 2			Variant 3		
No.	Criteria	Weight	Description	Units	Value	Point	Weighted Weight	Value	Point	Weighted Weight
1	Weight Distribution	3	Distance of center of gravity	mm	2.49	4	12	54.34	2	6
2	Device Weight	2	Weight of device	g	1190.60	2	4	1103.30	2	4
3	Device Size	1	Length of device	mm	265.00	2	2	195.00	3	3
		1	Width of device	mm	145.00	3	3	145.00	3	3
		1	Thickness of device	mm	52.20	4	4	69.20	4	4
4	Ease of Assembly	1	Number of parts required to be assemble	-	58	2	2	42	3	3
5	Swappable parts	2	Number of swappable parts available	-	1	1	2	1	1	2
	Total	11					29			25
	Technical Rating, Rt						0.659			0.568

Table 6.16: Technical Evaluation of Preliminary Design Variants (1/2)

Evaluation criteria					Variant 6			Variant 7		
No.	Criteria	Weight	Description	Units	Value	Point	Weighted Weight	Value	Point	Weighted Weight
1	Weight Distribution	3	Distance of center of gravity	mm	28.09	3	9	92.48	2	6
2	Device Weight	2	Weight of device	g	1112.60	2	4	889.20	3	6
3	Device Size	1	Length of device	mm	198.77	3	3	222.50	2	2
		1	Width of device	mm	145.00	3	3	135.50	3	3
		1	Thickness of device	mm	92.63	4	4	47.70	4	4
4	Ease of Assembly	1	Number of parts required to be assemble	-	49.00	3	3	35	3	3
5	Swappable parts	2	Number of swappable parts available	-	3.00	3	6	1	1	2
	Total	11					32			26
	Technical Rating, Rt						0.727			0.591

Table 6.17: Technical Evaluation of Preliminary Design Variants (2/2)

Production Cost		
Variant	**Cost, $C_{variant}$(€)**	**Economic Rating, R_e**
Variant 2	74.79	0.33
Variant 3	48.08	0.52
Variant 6	43.84	0.56
Variant 7	41.07	0.60

Table 6.18: Economic Evaluation of Preliminary Design Variants

Variant	**Technical Rating,** R_t	**Economic Rating,** R_e	**Total Rating,** R
Variant 2	0.66	0.33	0.494
Variant 3	0.57	0.51	0.540
Variant 6	0.73	0.64	0.645
Variant 7	0.59	0.60	0.596

Table 6.19: Total Rating of Preliminary Design Variants

7 Detail Design

Detail design involves finalizing the specifications of individual components, including their shapes, dimensions, materials, and surface properties [Pah07, 436]. Figure 7.1 shows the steps involved in the detail design process.

Additionally, it encompasses the creation of production documents such as detailed component drawings and assembly instructions [Pah07, 436], which are usually accomplished using Computer-Aided Design (CAD) software.

The evaluation of the preliminary design variants shows that Variant 6 is the best. Any improvements will be added to the design, while any weaknesses will be addressed. The result of this process is the final design of the device. The detailed drawing of the device is shown in Appendix A.2.

Power Switch

This component is critical in controlling the Raspberry Pi's power supply. It is imperative to have a reliable method for powering up and shutting down the Raspberry Pi to ensure smooth operation and prevent potential data corruption.

One available method utilizes the GPIO pins to initiate a shutdown sequence for the Raspberry Pi [Lab21]. While effective in bringing the device to a hibernation state, it is essential to note that this method does not completely cut off power. As a result, the Raspberry Pi still draws a minimal amount of power even in this low-power state [jdb].

A more straightforward approach is recommended to achieve more efficient power management, which involves the implementation of a simple physical push button as switch (refer to Figure 7.2). This implementation directly connects and disconnects the power supply to the Raspberry Pi. As a result, when

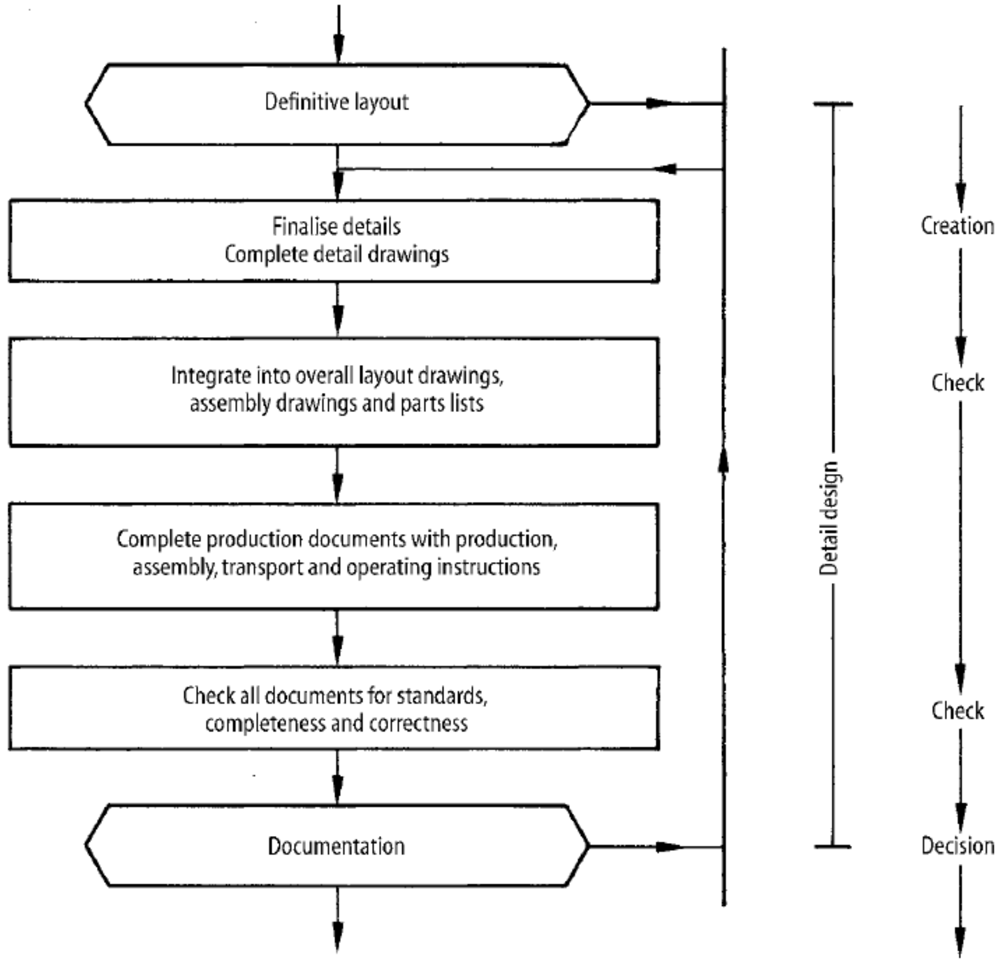


Figure 7.1: Steps in the detail design process

the switch is in the *off* position, it completely severs the power supply, ensuring that the Raspberry Pi consumes no power whatsoever.

Camera Protection

Figure 7.3 shows the original position of the camera component. As seen in the figure, the camera is seamlessly integrated into the device's body, with the lens protruding slightly.

However, this design does pose a potential risk. If the device is inadvertently placed with the lens side facing a surface, it could get damaged, severely affect-

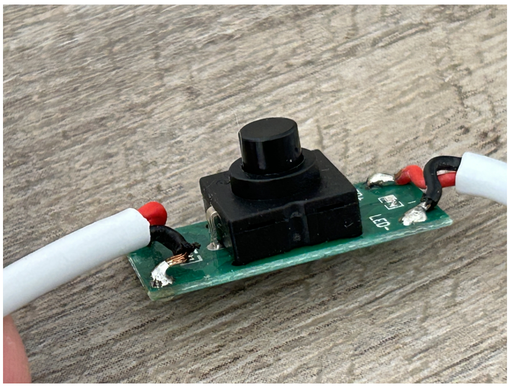


Figure 7.2: Power Switch

ing its performance.

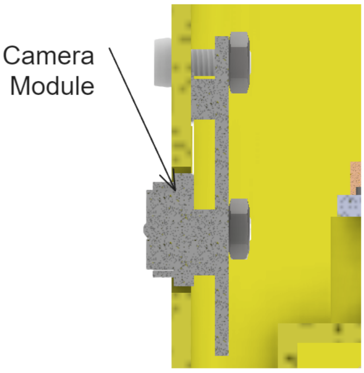
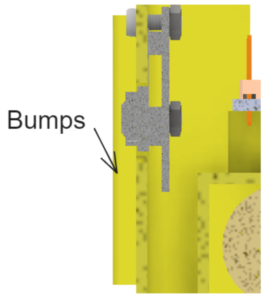


Figure 7.3: Position of the camera component

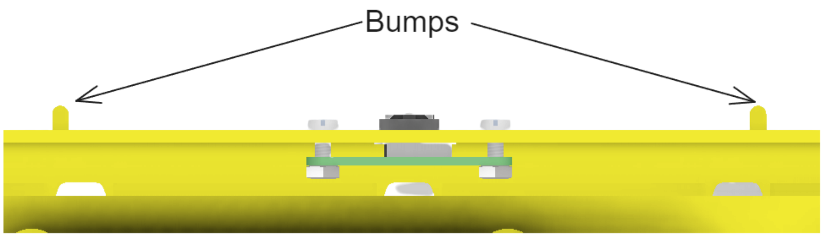
To address this concern, we have incorporated a 3 mm high bump (as seen in Figure 7.4) as a preventative measure. The bump is strategically positioned to lift the camera lens above surfaces, preventing direct contact.

Screen Protection

Similar to the camera, the screen of the device also requires protection. Similar to the camera protection, a protective bump has been incorporated around the screen perimeter to address this issue. Refer to Figure 7.5 for a visual representation.

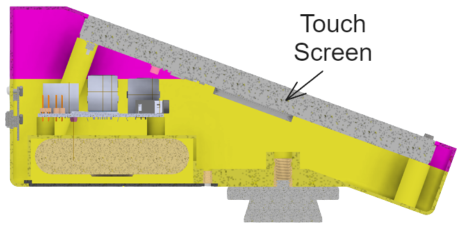


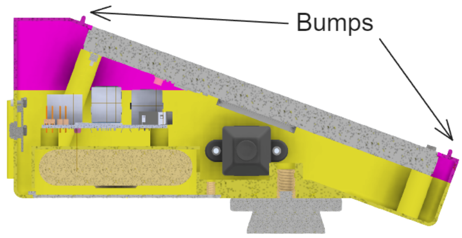
(a) Left View



(b) Top View

Figure 7.4: Protective bump for camera





(a) Before

(b) After

Figure 7.5: Protective bump for screen

Column for Threaded Inserts

Previously, in Section 6.3.1, we delved into utilizing threaded inserts alongside screws to firmly attach components to the body. It is crucial to note that distinct sizes of threaded inserts necessitate particular minimum wall thicknesses for the columns and hole depths to guarantee adequate engagement and steadiness. A sizing guide is provided by Ruthex threaded inserts, as shown in Table 7.1.

Thread Size	Hole size (mm)	Min. thickness (mm)	Min. height (mm)
M2.5	4	1.6	6.7
1/4"	8	3.3	13.7

Table 7.1: Sizing guide for threaded inserts [ruta][rutb]

LAN Port

To enable easier maintenance of the Raspberry Pi, a Local Area Network (LAN) port is added to the device, which enables the Raspberry Pi to be accessed directly without disassembly. This strategic integration allows seamless connectivity, enabling direct access to the Raspberry Pi's functionalities and resources over a local network. Figure 7.6 shows the LAN port.

The LAN port, illustrated in Figure 7.6, is positioned on the side of the device (as shown in Figure 7.7), making it easy for users to perform maintenance tasks.

Figure 7.6: The LAN port

Color Scheme

The selection of colors is crucial in making the product visually appealing, especially since the target market is the police force. To achieve this, we utilized the German police logo for inspiration, as shown in Figure 7.8. The predominant color of blue in the logo is used as the primary color for the device. Additionally, we used yellow from the logo as the color for the handle grip and white for the device's top cover. Figure 7.9 shows the device preview with the recolor.

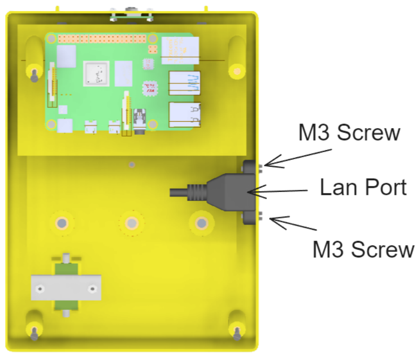


Figure 7.7: Position of the LAN port

Figure 7.8: Germany Police Logo [bun]

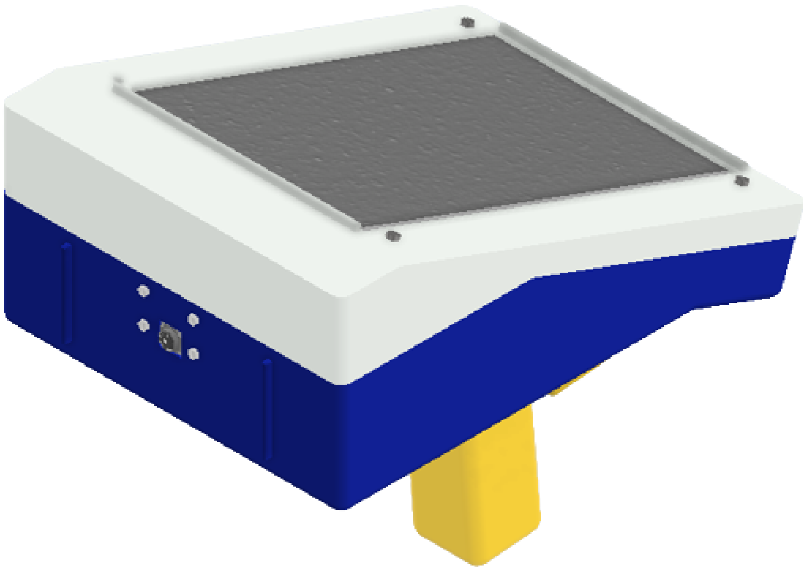


Figure 7.9: Result of recolor

8 Printing and Assembly

8.1 Printing

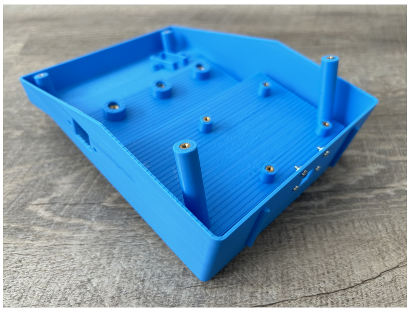
In this section, we will describe the printing process for the prototype. This process was carried out at the Workshop of the University of Applied Sciences Brandenburg using the printer described in Section 3.2.1. The following parameters are utilized for the process.

- Layer Height: 0.2 mm
- Infill Density: 15 %
- Print Speed: 60 mm/s
- Supports: Everywhere
- Filament: PLA

Table 8.1 provides information on the printing time and the weight of filament used. Additionally, Figure 8.1 showcases the final printed parts after removing the support materials.

Part Name	Weight of PLA used (g)	Printing Time (h)
Top Cover	57.71	2.00
Main Body	245.14	9.25
Battery Cover	22.21	0.67
Switch Cover	1.31	0.05
Handle Pistol	64.63	1.98

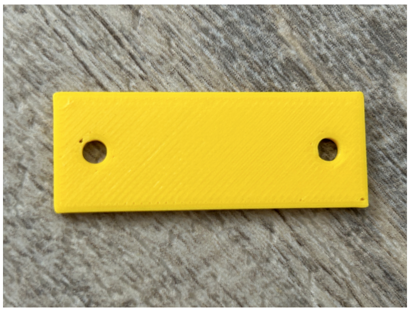
Table 8.1: Printing Time and Filament Used



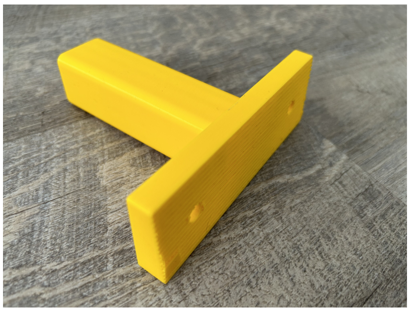
(a) Main Body

(b) Top Cover

(c) Battery Cover



(d) Switch Cover



(e) Handle Pistol

Figure 8.1: Printed Parts

8.2 Assembly

The assembly process is done by following the steps below:

Step 1: Installation of Threaded Inserts

In order to securely install the brass threaded inserts into the main body, it is recommended to use a soldering iron [Her23]. Begin by placing the chosen threaded insert onto the targeted position, aligning it with the desired hole. Heat the soldering iron to a suitable temperature, ranging from 225 °C to 245 °C for PLA material.

Once the soldering iron has reached the appropriate temperature, apply it gently to the top of the threaded insert. This process will transmit controlled heat into the material, causing the wall to soften and allow the threaded insert to be inserted. For a visual representation of how a threaded insert should look once correctly installed into the main body, please refer to Figure 8.2.

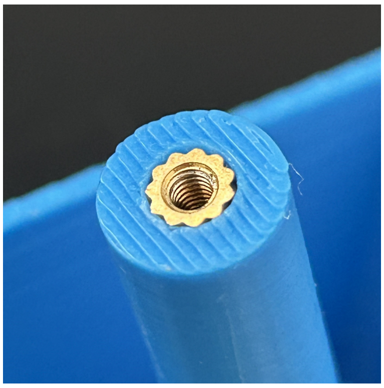


Figure 8.2: The installed threaded insert

Step 2: Installation of Switch

Installing a switch to the main body is a straightforward process that requires a few basic materials: the switch itself, a switch cover, two M2.5 nuts, and M2.5 screws. Position the switch inside the designated switch holder (see Figure 8.3), ensuring that the button faces outward for easy access.

Next, the switch cover is placed on top, aligning it with the switch and the corresponding holes in the main body. Once aligned, the M2.5 screws and nuts secure the switch and the cover to the main body. Figure 8.4 shows the completed installation of the switch.

Step 3: Installation of LAN Port

This step begins by locating the slot of the LAN port on the main body, which is located on the right side of the main body (see Figure 8.5). The LAN port is

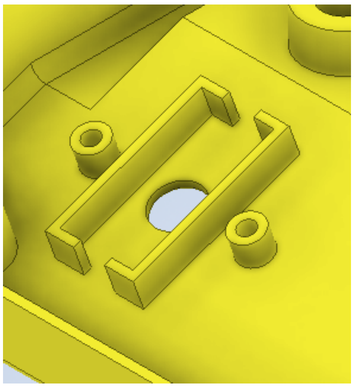


Figure 8.3: The Switch Holder

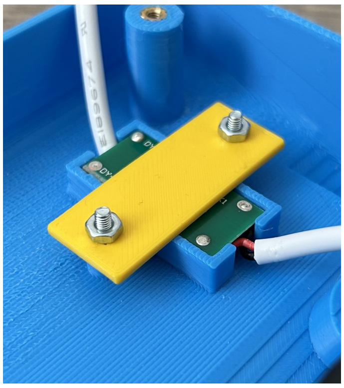


Figure 8.4: The installed switch

inserted into the slot and secured using the M3 screws. Figure 8.6 shows the completed installation of the LAN port.

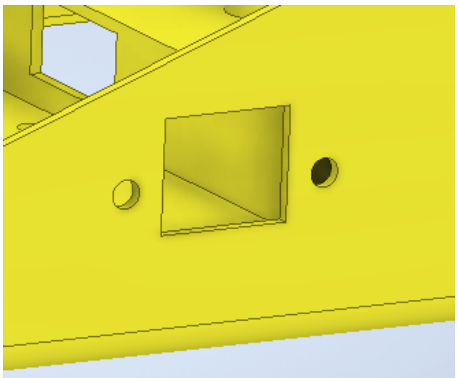


Figure 8.5: The LAN Port Slot

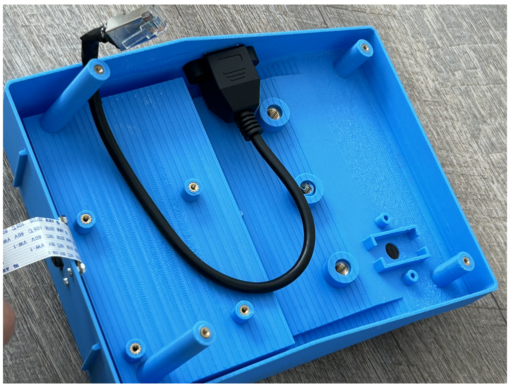


Figure 8.6: The installed LAN port

Step 4: Installation of Camera Module

The camera module is installed to the main body using M2 screws. The camera module is placed in the designated slot on the main body (see Figure 8.7). The M2 screws are then used to secure the camera module to the main body. Figure 8.8 shows the completed installation of the camera module.

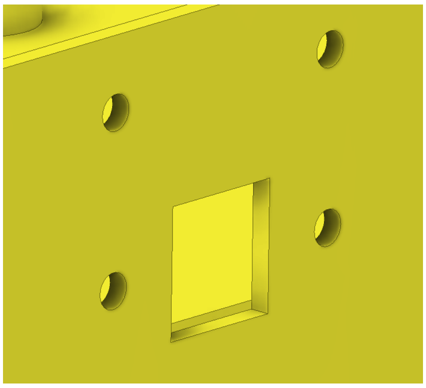


Figure 8.7: The Camera Module Slot

Step 5: Installation of Battery

To start the installation process, insert the battery into the holder as shown in Figure 8.9. Then, connect the battery to the switch with a 90-degree USB-A to USB-C connector.

To fasten the battery to the main body, place the battery cover over the battery

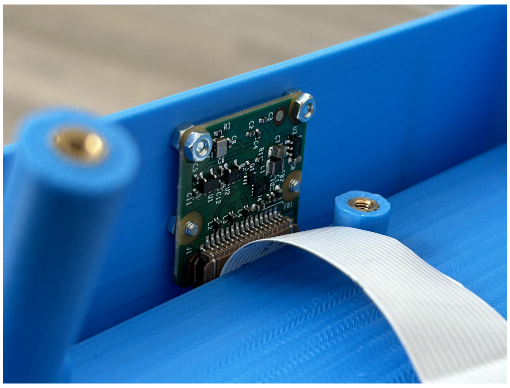


Figure 8.8: The installed camera module

and main body. Use the M2.5 screws to secure the battery cover to the main body. The finished battery installation can be seen in Figure 8.10.

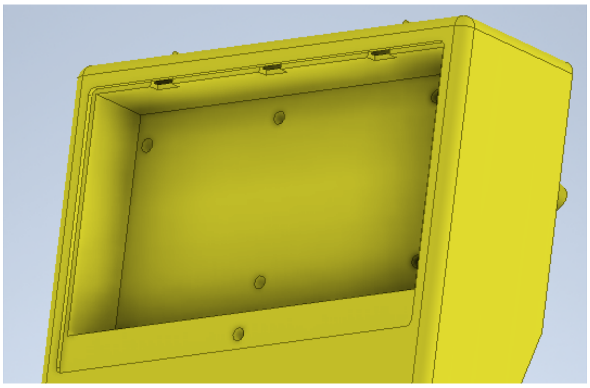


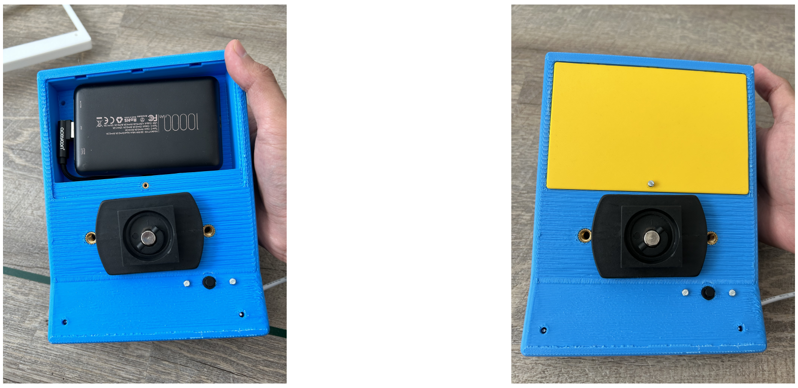
Figure 8.9: The Battery Holder

Step 6: Installation of Raspberry Pi

The Raspberry Pi is installed on the main body by using the M2.5 screws. The Raspberry Pi is placed in the designated slot on the main body (see Figure 8.11). The M2.5 screws are then used to secure the Raspberry Pi to the main body.

Next, the following connections are made to the Raspberry Pi:

- The LAN port is connected to the Raspberry Pi via a LAN cable.
- The camera module is connected to the Raspberry Pi via a ribbon cable.
- The switch is connected to the Raspberry Pi via a USB-C cable.



(a) Without Battery Cover **(b)** With Battery Cover

Figure 8.10: The installed battery

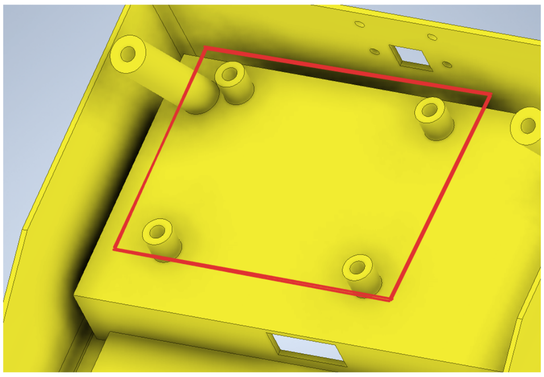


Figure 8.11: The Raspberry Pi Slot

Step 7: Installation of Screen and Top Cover

The final step is to install the screen and the top cover. Begin by placing the screen into the designated slot on the main body (see Figure 8.12) and align the hole on the screen with the hole on the main body. Next, the top cover is placed on top of the main body. The M2.5 screws are then used to secure the top cover to the main body. Figure 8.13 shows the completed installation of the screen and the top cover.

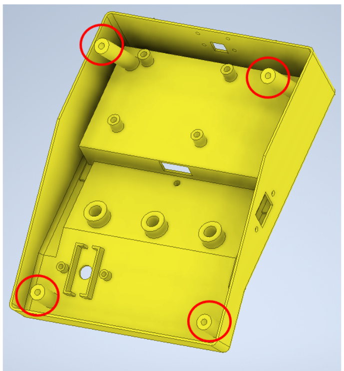


Figure 8.12: The Screen Slot

Figure 8.13: The installed screen and top cover

8.3 Final Product

Figure 8.14 shows the final product. The total cost of building the product including the cost of printing and all of the materials is shown in Table 8.2 and Table 8.3 respectively.

Figure 8.14: The Final Product

Part Name	Material Cost	Energy Cost	Total Cost
Top Cover	1.73 €	0.04 €	1.77 €
Main Body	7.35 €	0.17 €	7.53 €
Battery Cover	0.67 €	0.01 €	0.68 €
Switch Cover	0.04 €	0.00 €	0.04 €
Handle Pistol	1.94 €	0.04 €	1.98 €
Total	**11.73 €**	**0.26 €**	**11.99 €**

Table 8.2: Total Printing Cost

Parts Name	Amount	Price	Remarks	Total Price
Printing Cost	1	11.99 €	Calculated	11.99 €
RaspberyPi 4B 2GB	1	35.00 €	RaspberryPi	35.00 €
Camera Module v2	1	25.00 €	RaspberryPi	25.00 €
Waveshare 7inch screen	1	53.99 €	Waveshare	53.99 €
Veektomx Battery	1	25.99 €	Amazon	25.99 €
Screw M2x10mm (DIN 84)	4	0.05 €	Wuerth	0.20 €
Screw M2.5x10mm (DIN 84)	7	0.05 €	Wuerth	0.35 €
Screw M2.5x18mm (DIN 84)	4	0.11 €	Wuerth	0.44 €
Screw M3x10mm (DIN 84)	2	0.05 €	Wuerth	0.10 €
Nut M2 (DIN 934)	4	0.04 €	Wuerth	0.15 €
Nut M2.5 (DIN 934)	2	0.04 €	Wuerth	0.07 €
Threaded Inserts M2.5	9	0.13 €	Ruthex	1.16 €
Threaded Inserts 1/4"	3	0.50 €	Ruthex	1.50 €
1/4" Camera Screw	2	1.40 €	Amazon	2.80 €
LAN Port	1	5.67 €	Aliexpress	5.67 €
Switch Pi	1	3.90 €	Amazon	3.90 €
90-Degree USB-A to USB-C	1	2.75 €	Amazon	2.75 €
90-Degree USB-A to micro USB	1	5.99 €	Amazon	5.99 €
90-Degree HDMI	1	2.12 €	Aliexpress	2.12 €
Total				**168.29 €**

Table 8.3: Total Material Cost

9 Conclusion

In summary, this research has developed and demonstrated a highly promising handheld device engineered to measure vehicle speed precisely. This prototype is a cost-effective alternative to the existing speed measurement tools, particularly the conventional speed pistol. By integrating cheap computational components, high-quality cameras, and the power of 3D printing, we propose an affordable, highly accurate, and reliable device.

The design process of the prototype was carried out based on VDI guideline 2221 and divided into four parts: task clarification, conceptual design, embodiment design, and detail design. The task clarification phase was conducted by identifying the problem, defining the requirements, and setting the specifications.

The conceptual design phase was carried out by defining function and function structure. The function structure was then used to generate working principles with the help of brainstorming and analysis of existing technical systems. The result of idea generation is presented inside Zwicky's morphological box. The working principles are then combined to form eight different working structures. After careful evaluation, only four were selected for further development.

In the embodiment design phase, the four working structures were developed further by defining their 3D model with Autodesk Inventor. With the help of the 3D model, the working structures were evaluated based on their physical properties, manufacturability, and ergonomics. The estimated cost of manufacturing each variant is also calculated and compared. The evaluation results show that variant 6 is the most suitable to be chosen as the final design.

The detailed design phase was carried out by developing the final design of the prototype based on the chosen variant. In this phase, the 3D model of the prototype was developed further by adding more details and components. The 3D model was then used to generate the technical drawing of the prototype.

The final design was then manufactured using 3D printing technology. The prototype was assembled and tested to ensure it worked as intended. The test result shows that the prototype can securely hold the inner components properly and protect them. In terms of ergonomics, the handling of the device seems stable and lightweight.

The prototype of the speed pistol has been developed at a remarkably low cost, totaling only 168.29 €. This cost is significantly lower, approximately 90%, compared to the conventional speed pistol discussed in Chapter 2. However, it is essential to note that this prototype cost does not serve as a direct indicator for setting the product's final selling price. Further development work is required to enhance the prototype's durability and reliability.

In other words, while the current prototype shows promise in terms of cost-efficiency, additional investment and refinement are necessary to ensure that the final product meets the necessary quality and performance standards, which may involve additional testing, material improvements, and refining the production process to create a market-ready product.

A proper Graphical User Interface (GUI) will be developed in the next step to allow the user to interact with the device.

Part II

GUI Development

10 Literature Review

10.1 Waterfall Model

The waterfall model is one of the oldest and most well-known software development methodologies and was initially proposed by Winston W. Royce in 1970 [Roy87]. The model is characterized by its linear and sequential approach to project management [NST23]. Each phase must be completed before moving on to the next, and it is particularly well-suited for projects with well-defined and stable requirements.

The original waterfall model, as outlined by Royce, consisted of seven phases: System Requirements, Software Requirements, Analysis, Program Design, Coding, Testing, and Operations [Roy87]. However, over time, variations of the model have emerged. Hausen [Hau] mentioned that, the most commonly recognized version includes five phases: Requirements Analysis, Design, Coding, Testing, and Maintenance .

One advantage of the waterfall model is its simplicity and ease of understanding and implementation [SYM20]. It provides a clear structure and allows for a step-by-step progression through the development process [SYM20]. Rachma and Muhlas [RM22] argue that the waterfall model is particularly well-suited for projects that involve generic software or systems that provide services to buyers.

However, the waterfall model has also been criticized for its limitations. Zahia et. al [BZJ14] argues that, this model is best suited for hardware production and may neglect the unique characteristics of software development . Yahya and Maidin [YM23] stated that the waterfall model does not allow for flexibility or adaptability during the development process, which means that, once a stage is completed, it is difficult to make changes or go back to a previous stage. This lack of flexibility can be problematic when uncertainties arise or when there is a need for iterative development.

Figure 10.1 shows the waterfall model that will be utilized for this project.

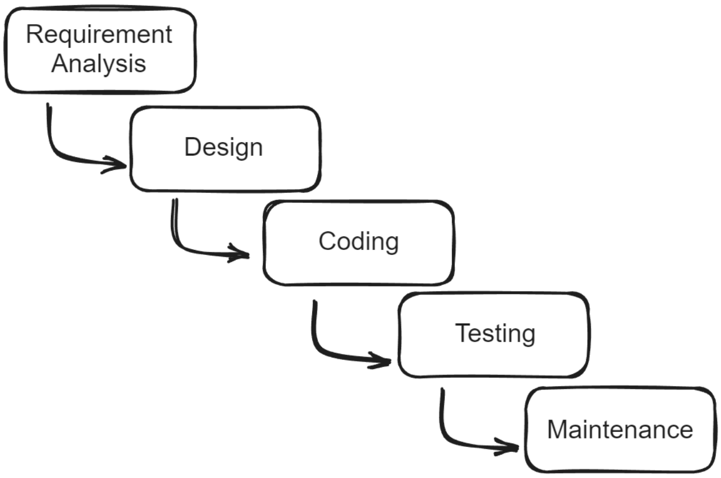


Figure 10.1: Waterfall Model

10.2 Model-View-Controller

The Model-View-Controller (MVC) architecture is a design pattern commonly used in software development to separate the concerns of an application into three distinct components: the model, the view, and the controller [Gar23, 47]. The model represents the data and business logic of the application, the view is responsible for presenting the data to the user, and the controller handles user input and updates the model and view accordingly [SZA14].

The MVC architecture offers several benefits for software development. Firstly, it promotes code reusability by separating the different aspects of the application [SZA14]. Additionally, MVC allows for multiple representations of the same information, enabling the creation of different views for different user interfaces [SZA14]. This flexibility is particularly useful in interactive applications where users may have different preferences or requirements.

Another major advantage of MVC is that it allows for seperation of concerns [Ste]. By isolating the data, business logic, and presentation layers, developers can more easily understand and modify each component without affecting the others. This modularity improves the maintainability and scalability of the application, as changes can be made to one component without impacting the entire system [Koz23].

Figure 10.2 shows the MVC architecture that will be utilized for this project.

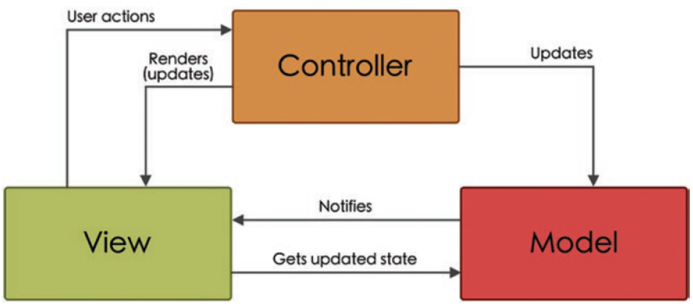


Figure 10.2: Model-View-Controller Architecture [Gar23, 46]

11 Requirement Analysis

In the waterfall model, as proposed by Royce, requirements analysis is typically the first phase of the development process [Roy87]. This phase involves gathering, documenting, and analyzing the needs and expectations of stakeholders to define the scope and objectives of the software project [Hau]. The objective of requirements analysis is to pinpoint the user needs and translate them into explicit, quantifiable, and attainable requirements that the software development team can employ to design and build the system [Wal23].

According to Walker [Wal23], within a large organization, this phase is usually carried out by diverse team. Business Analysts identify and document system needs, while Project Managers oversee the process. Developers use these requirements to design the system, and Testers validate them. Stakeholders provide essential input, and Subject Matter Experts offer specialized knowledge to help ensure the requirements are accurate and complete.

According to Geogy and Dharani [GD16], one of the major challange in this phase is determining what needs to be constructed, as unlike other phases in development, errors in this phase can have far-reaching consequences if identified later on. They also stated that, one of the primary hurdles in the requirements specification process is striking a balance between providing sufficient detail for clear understanding without overly constraining the system.

Hence, to properly define the project requirements, the following steps are taken [Sim23]:

Identify Key Stakeholders and End-Users

In the initial phase, it is imperative to identify key stakeholders who hold significant influence over the project, playing a crucial role in determining the project's scope. Additionally, pinpointing the end-users, as the primary beneficiaries of the product, is equally imperative. For this project, the end-users will be the police officers and the administrators of the police department.

Capture Requirements

The subsequent step involves gathering requirements from stakeholders and end-users. Various techniques such as one-on-one interviews, focus groups, use cases, and prototypes are employed to capture their specific needs. Each technique serves a unique purpose, from gathering detailed individual requirements to visualizing the product's functionality.

Categories Requirements

Once requirements are gathered, they are categorized into functional and non-function requirements. This categorization ensures clarity and avoids potential confusion.

Interpret and Record Requirements

Following categorization, a meticulous process of interpretation and recording ensues. This involves assessing the feasibility, prioritizing requirements, conducting impact analyses, resolving conflicts, and ensuring precise and detailed descriptions aligned with business objectives.

After a thorough analysis, a comprehensive document detailing the requirements is created and distributed among key stakeholders, end-users, and development teams.

Sign off

Finally, to maintain project scope integrity, obtaining a signed agreement from key stakeholders is essential. This signifies a final decision on the requirements and prevents any uncontrolled expansion of the project's scope.

11.1 Project Requirements

Figure 11.1 and 11.2 shows the result of the requirement analysis process.

Project Name

Handheld Speed Measurement Device

Objective

Create a portable, lightweight handheld device that accurately measures a moving object or vehicle's speed.

Stakeholders

1. Police officers
2. Police departments
3. Government officials

Figure 11.1: Project Requirements (1/2)

Functional Requirements

1. Capture multiple images or videos.
    a. Capture the image and the timestamp accurately.
    b. The image quality is high.
2. Track object
    a. The algorithm should be able to perform object tracking.
    b. The method does not consume high computational power.
3. Image processing
    a. The image should be aligned to remove any movement during capturing.
4. Calibration
    a. Lane calibration should be able to be performed automatically.
    b. Solve problems with inconsistency in road width.
    c. When automatic calibration cannot be performed, a redundant method should come into play (manual calibration or distance measurement)
5. Data Handling
    a. Users should be able to save and load the captured data whenever possible.

Non-Functional Requirement

1. Performance
    a. Ensure faster processing by taking leverage of parallel processing with a thread pool.
    b. Optimize algorithms by analyzing bottlenecks.
2. Usability
    a. The user interface should be user-friendly and straightforward.
3. Scalability
    a. The program should be designed with scalability in mind.
    b. Developers should be able to add features in the future.

Figure 11.2: Project Requirements (2/2)

12 Design

The design focuses on the development of both hardware and software architecture, which addresses aspects like performance, security, and data structure [Hau]. Furthermore, alongside technical considerations, attention is given to the user interface, ensuring it is user-friendly and navigable [Hau].

This section covers the project's software architecture, emphasizing the MVC model and the integration of a Thread Pool for efficiency. It then delves into user interface design, highlighting principles like familiarity and simplicity. The wireframe and color scheme choices are introduced, along with the font selection. The concept of user flow is explained through task flows and wireflows.

12.1 Software Architechture

The design of the MVC architecture fot this project is shown in Figure 12.1. Within the figure, the seperation between each MVC components are clearly shown. The view and the model component are completely decoupled from each other, and the controller component is responsible for mediating the interaction between the view and the model.

Another feature that is shown in Figure 12.1 is the implementation of multiple views and controllers. This implementation ensures the scalability of the system, which is an important project requirement, as defined previously in Section 11.1. By doing so, the functionality within the project can easily be expanded in the future.

Additionally, the decision to implement multiple controllers is also to prevent

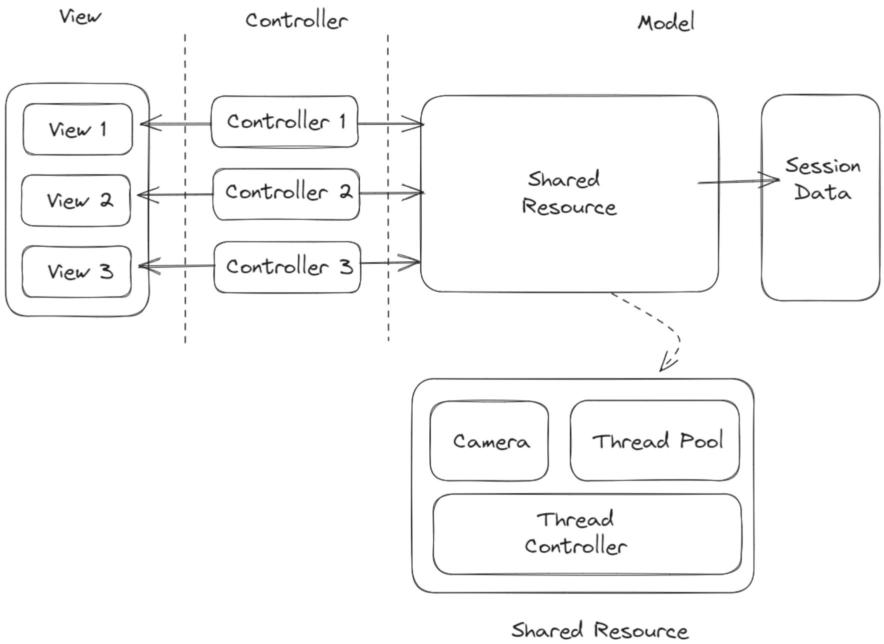


Figure 12.1: Software Architecture

the "Giant Controller" problem. This problem is defined as a controller that handles too many responsibilities, which can cause the controller to become bloated and difficult to maintain. By seperating the controllers, it ensures that each controller only handles a specific set of responsibilities, which makes the system easier to maintain.

Notably, the view components are meticulously separated from one another. This deliberate segregation prevents any unintended mixing of view elements. For instance, View 1 might necessitate specific UI components, while View 2 might require an entirely different set. This simplifies the implementation process and renders it more comprehensible.

Within the model components, there are two major subcomponents, which are the Shared Resource and Session Data. Both of these components are responsible for managing the data within the application. The child components within the Shared Resource component will be discussed in detail in the later section.

12.1.1 Thread Pool

A thread pool is a collection of worker threads that efficiently execute asynchronous callbacks on behalf of the application [KBM]. It is primarily used to reduce the number of application threads and provide management of the worker threads [KBM].

The working principles of thread pool is relatively simple. It begins by initializing a pool of threads, or sometimes referred to as worker or executor. The number of threads in the pool is usually dependent on the number of cores in the CPU [Gee20]. Too large thread pool size can results in resource trashing, where time is wasted in context switching between threads [Gee20].

When workers creation is completed, they are now ready for execution. Task can now be created and placed inside a queue [Gee20]. Idle workers will then be assigned to a task in the queue and once the task is completed, the worker will be assigned a new task until the queue is empty [Gee20]. If all workers are busy, the task will be placed inside the queue and wait for an available worker to be assigned to it [Gee20].

Figure 12.2 shows an overview on thread pool architecture.

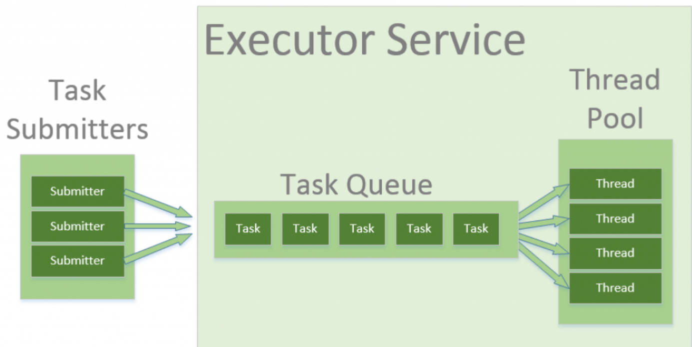


Figure 12.2: Thread Pool Architecture [Eug23]

For this project, the thread pool object will play a pivotal role in improving the efficiency of the system. It is one of the required features for this project, as defined in Section 11.1. It solve one of the problem mentioned in [BMS23], where some computer vision algorithms are computationally expensive and can take a long time to execute.

If these tasks are to be run in single threaded execution, the user will have to wait for a long period of time before the task is completed. This is not ideal for a user interface application, as it will cause the application to be unresponsive. However, if multiple of this instances are to be run in parallel, it will cause the system to be overloaded and can cause the system to crash.

This is where the thread pool comes in to solve this problem. It will create a fixed number of available threads, which allows for parallelism but also prevents the system from being overloaded. The thread pool will also manage the threads, which means that the user does not have to worry about creating and managing the threads. The user only needs to create the task and submit it to the thread pool, which will then assign the task to an available thread.

12.1.2 Thread Controller

The Thread Controller is a critical component introduced to alleviate the responsibilities of the main controller in a system. By incorporating a Thread Controller, the main controller can focus on efficiently handling requests between the view and the model, without becoming bogged down with intricate threading operations.

Within the Thread Controller, there exists multiple sets of threads. It's important to note that these threads are distinct from those defined within a thread pool. In this context, the threads within the Thread Controller are specialized units responsible for executing specific business logic. Each thread operates in parallel, enabling the system to handle multiple tasks simultaneously.

Moreover, the Thread Controller is designed with access to a Thread Pool. This integration empowers the system to execute tasks that necessitate parallel processing in an organized manner. Tasks can be placed in a queue within the Thread Pool, ensuring they are executed efficiently and without resource conflicts.

This implementation stands as a powerful facilitator of scalability. Developers have the flexibility to define their own sets of threads, tailoring the system to meet specific performance requirements. Additionally, Developers gain the ability to exercise fine-grained control over thread execution directly from the controller, promoting a high degree of customization and adaptability to varying workloads. This dynamic combination of a Thread Controller and Thread Pool lays a robust foundation for responsive, efficient, and scalable applications.

12.2 User Interface Design

User Interface (UI) design is a crucial aspect of product development that focuses on the look, style, and interactivity of a product [Cou23]. It is the first thing users encounter when they use an application or visit a website [Cou23]. A user-friendly UI design holds a pivotal role in enriching the user experience, boosting user engagement, and ultimately shaping the success of an application [Alg23].

There are various design guideline that can be utilized in order to create a user-friendly UI design. Paun [Pau20] stated that, familiarity in UI design benefits both users and designers. It streamlines workflows, ensures a seamless experience, and allows for efficient use of established conventions. Adhering to standards and maintaining consistency are key for a unified and positive user experience.

Fleck [Fle21] argues that, simplicity in design plays a crucial role in UI design. A simple design minimizes cognitive load and allows users to accomplish tasks efficiently. The author also emphasize that it is important to remember that simplicity does not mean sacrificing functionality; rather, it means prioritizing essential elements and removing any extraneous details that may cause confusion or overwhelm the user.

In GUI design, feedback is as important as consistency and simplicity. It comes in forms like visual cues and error messages, helping users understand the product's state and how to interact with it. This feedback lessens confusion, builds trust, and aids in learning how to use the product, even when errors occur. [Flo22]

12.2.1 Wireframe

A wireframe is a basic, simplified layout of a digital interface that outlines content placement and page components [Whi23]. It serves as a quick way to present a concept or idea without extensive time or resources [Whi23].

Gupta [Gup23] stated that, a good wireframe abstains from including elements that could divert attention from decisions pertaining to the site's structure, but it instead prioritizes the functional purpose over visual aesthetics, presenting a straightforward, unadorned portrayal of the website's features in monochrome. The author also stated that, designers should provide annotations to explain and elaborate on specific features, providing clarity and context.

Figure 12.3 shows the rough sketch of the user interface that will be used for this project. The user interface will be divided into 4 main sections: the title panel, image panel, status panel, and button panel.

The title panel will be used to display the title of the application. It provide information to the user on which panel they are currently in. Additionally this component also provide other functionality for user such as closing the application, accessing the setting, and provide some basic information on the application.

The image panel is purposed to display images to user. It will be used to display the image that is captured by the camera, and also the image that is being processed. The status panel will be used to display the status of the application to the user. It is important to provide feedback to the user on the status of the application, so that the user will know what is happening in the application.

The button panel will be used to display the buttons that the user can interact with. In this panel, buttons will be grouped into its own categories, depending on its functionality. There will be two types of buttons in this panel, which are the bitmap button and the text button.

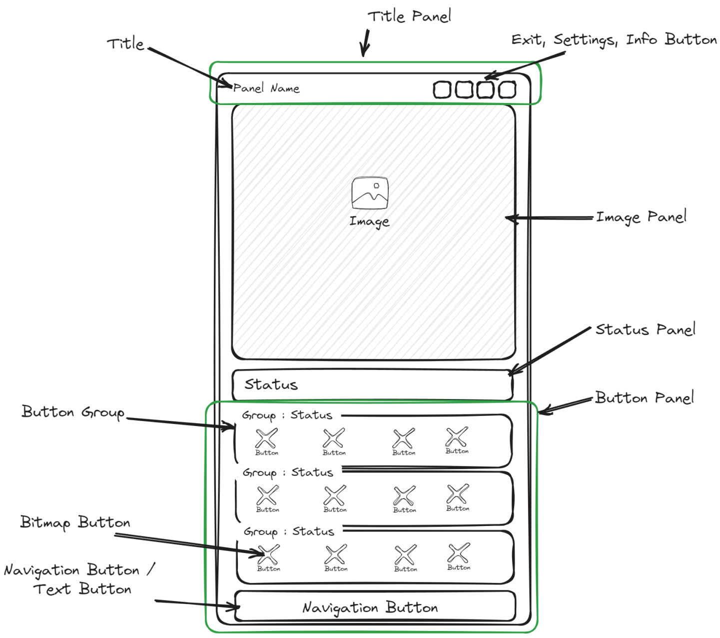


Figure 12.3: Wireframe

12.2.2 Color Scheme

Color is a pivotal element in UI design, playing a vital role in evoking emotions, establishing hierarchy, and distinguishing design components [M.23]. With correct implementation, color can greatly improve the usability and overall user experience of your website or mobile application [M.23].

Ou et al. [OLWW04] emphasized the strong correlation between color and emotion, highlighting how different colors evoke specific emotional responses. Their research underscores the significant impact that color choices can have on the emotional experience of users.

Lewandowska and Olejnik-Krugly [LOK21] stated that, color is a significant aspect of visual communication. It plays a crucial role in conveying information. It is integral in identifying products, indicating quality, and influencing user interfaces.

They also stated that, colors can evoke diverse visual sensations, and their impact is often experienced in combination rather than in isolation, at which it underscores the intricate role of color in human perception and communication.

Zhang and Ferris [FZ16] discusses factors for choosing colors in design: number of colors, color harmony, and text overlay. For the number of colors, options range from a single color with various shades to a 60-30-10 rule for triadic schemes, emphasizing primary, secondary, and accent colors. Figure 12.4 shows a few examples of color combination with 60-30-10 rule.

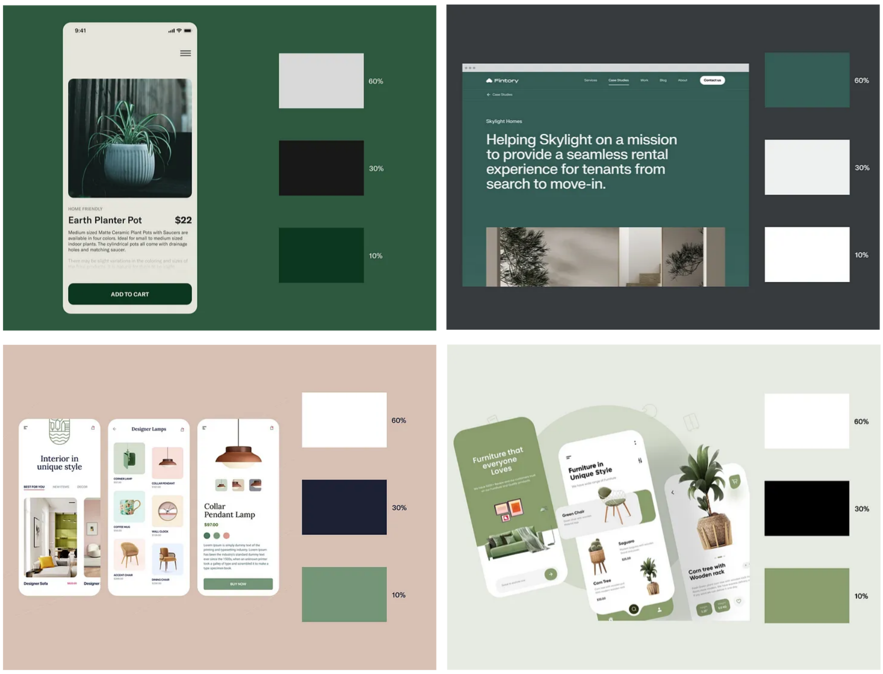


Figure 12.4: Example of Color Combination with 60-30-10 Rule [M.23]

In choosing the primary color, it is important to consider the brand's message and the emotions that the brand wants to evoke [M.23]. The secondary color should complement the primary color and provide contrast to support the dominant color, while the accent color is used in features like buttons and pop-ups [M.23].

Figure 12.5 shows the color combination that will be used for this project. A simple white and black color combination is used for the primary and secondary color respectively. Since the target user will be the policemen, the blue shade is used for the accent color to represent the police force.

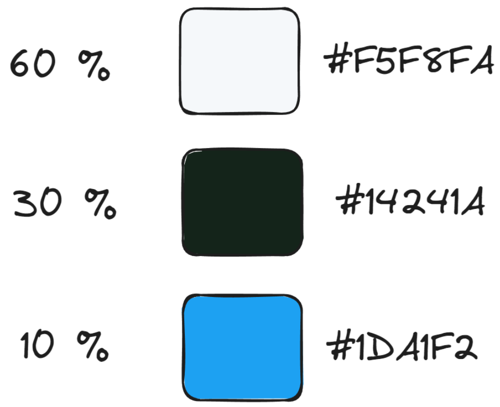


Figure 12.5: Color Combination

12.2.3 Typography

Typography is a critical element in UI design, as over 90% of online information is presented in text form [FP22]. It involves more than just selecting attractive fonts and other factors such as typeface, font, character, baseline, and height are important [FP22].

Sawyer et al. [SDCR20] stated that, picking the right font is really important. It affects how things look and how easy they are to read. This is especially true for quick glances on screens. The authors also stated that, recent studies found that different fonts can make a big difference in how easy things are to read in real-life situations.

For this project, the font that will be used is Roboto. Roboto is a sans-serif typeface family developed by Google as the system font for its mobile operating system Android [Mot22]. According to Google, this font is optimized to provide optimal reading experience across various devices [Mot22]. Figure 12.6 shows the Roboto font that will be used for this project.

Figure 12.6: Roboto Font

12.2.4 User Flow

User flow in UI design refers to the path a user takes to complete a specific task within an application or website. It includes each step, from the starting point to the endpoint [Dai22]. User flows are typically represented as diagrams that display the complete path a user takes when using a product [Bro23]. They help designers understand and anticipate the cognitive patterns of users in order to create products that enable a state of flow [Bro23].

For this project, we will utilize two types of user flow to help describe the user interaction with the system. The first type is the Task Flow, which is used to illustrates a linear process for completing a particular task [Tri23]. Task flows use natural language and do not contain any design elements [Tri23]. They are typically linear and represent the high-level actions that a user would take to arrive at a particular objective or endpoint [Rah22].

The other type of is wireflow, which is a visual representation of the screens and interactions a user follows to complete specific tasks [Tri23]. It combines aspects of a basic wireframe, task flow, and flowchart to advanced screen flows that depict multiple navigation paths in one diagram [Tri23]. Arrows and annotations between wireframes are added on a single canvas to indicate the paths a user may take while using the product [Ang].

Figure 12.7 shows the task flow of the application, showing the task required, from start to finish, to succesfully measure the speed of moving object. The wireflow of the application is shown in Figure 12.8, showing available panels that user can navigate to.

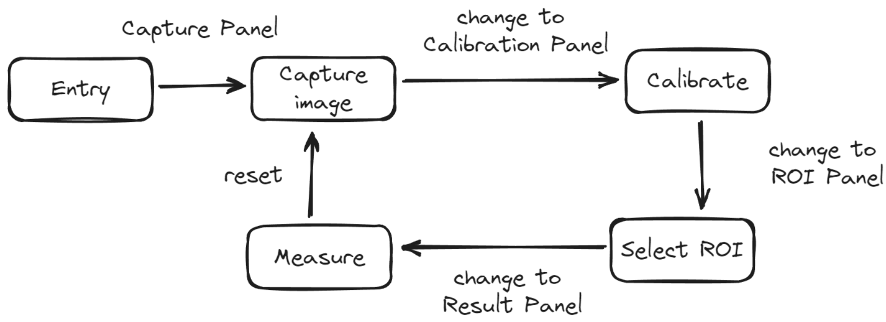


Figure 12.7: Task Flow

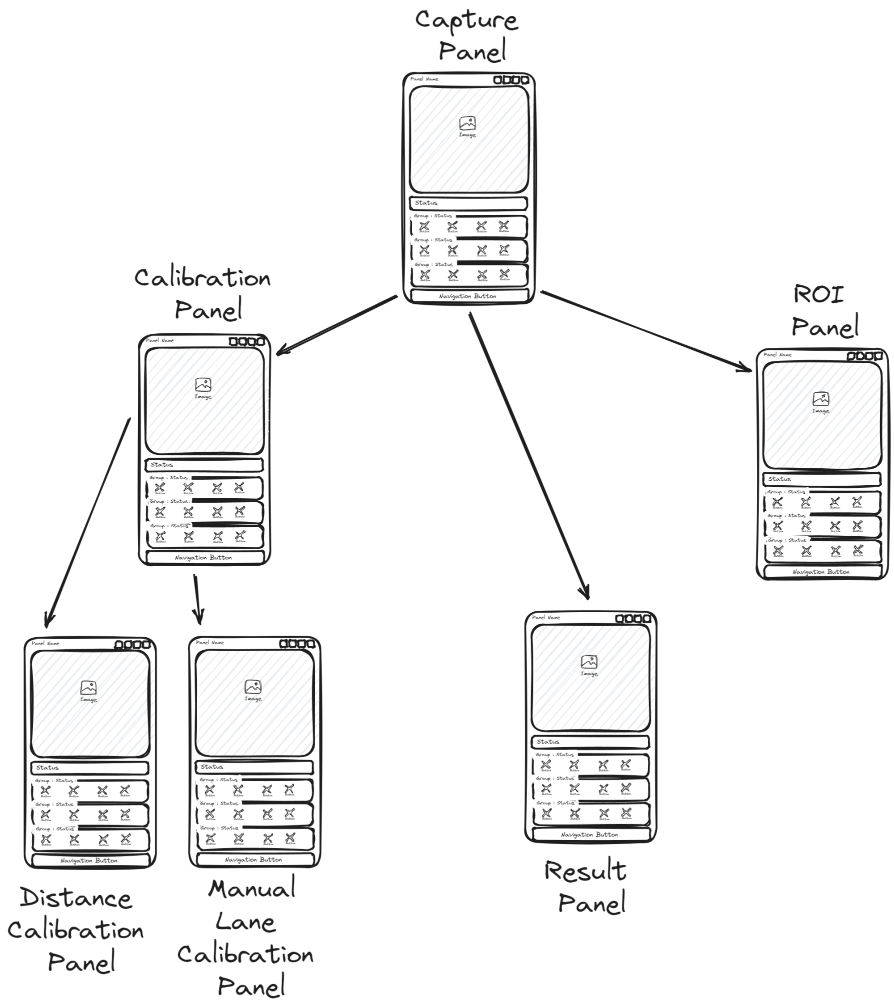


Figure 12.8: Wireflow

13 Coding

The third phase of the project involves the actual implementation of the application with the end goal is to produce a functional program that aligns with the established specifications [Hau]. This chapter will provide detail explanation on the implementation process of the application.

Each of the component from MVC architechture will be explained. Examples will be given where necessary. Additionally, improvement to the algorithm will also be discussed. The application will be developed with the C++17 programming language. Following libraries are used throughout the development process:

- OpenCV v4.5.5
- wxWidgets v3.2.2.1
- libcamera v0.0.4

13.1 View Implementation

As discussed previously in Chapter 10.2, the view is responsible for displaying the data to the user, in other words, it defines the user interface and displays the information from the model [KV20]. In this section, the implementation of the view is discussed in detail.

To further streamline the development process of view, we have set up a set of guidelines that we have followed throughout the development process. These guidelines are as follows:

Handling User Input

This guideline emphasizes that the View should be adept at managing and processing user interactions. It is responsible for capturing user inputs, such as mouse clicks, keyboard entries, or touch gestures, and conveying them to the Controller for further processing. This can involve events like form submissions, button clicks, or interactions with interactive elements on the user interface.

Avoid any business logic

This directive underscores the importance of maintaining a clear separation of concerns within the MVC architecture. The View should refrain from incorporating any form of business logic, which involves tasks like data validation, computation, or decision-making processes. Instead, these responsibilities are delegated to the Model and Controller components. By adhering to this guideline, the View remains focused on its core function of presenting data and interacting with the user interface.

Avoiding Direct Model Manipulation

This guideline reinforces the principle of ensuring that the View does not directly modify the state or data of the Model. Instead, any alterations to the underlying data should be orchestrated through the Controller. This establishes a controlled flow of data within the application, maintaining data integrity and adherence to the MVC architectural pattern.

Utilizing Templates or Layouts

This recommendation advocates for the adoption of templating engines or layout systems in view development. These tools facilitate the creation of modular, reusable components that can be dynamically combined to construct the user interface. By employing templates or layouts, developers streamline the process of designing and rendering the View, resulting in a more maintainable and scalable codebase.

Error Handling and Feedback Mechanisms

This guideline emphasizes the necessity of implementing robust error handling mechanisms within the View. It is imperative to provide clear and informative feedback to the user in the event of errors or invalid inputs. This ensures that users are guided through the application's workflow, even in scenarios where unexpected events occur. Effective error handling contributes to a more user-friendly and reliable user experience.

13.1.1 Main Layout

The implementation of the main layout is the first step in the development of the view. The main layout is the universal layout that is used throughout the application. The implementation is done based on the result of the wireframing process which is done in Section 12.2.1.

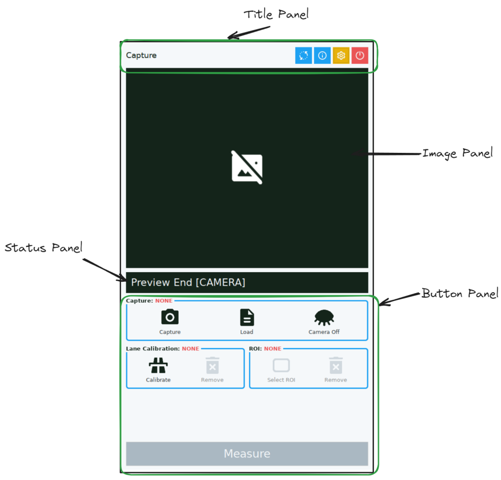


Figure 13.1: Main Layout

The main layout is shown in Figure 13.1. The main layout consists of a title panel, image panel, status panel and button panel. All of the function of the components are explained previously in Section 12.2.1.

13.1.2 Bitmap Button

Bitmap Button is a type of button that is mainly used within the application. It contain a bitmap image and a text lable positioned on below the image. Figure 13.2 shows a couple of bitmap buttons that are used within the application.

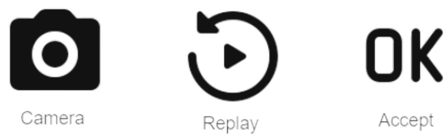


Figure 13.2: Bitmap Button

Throughout the implementation process, there are 3 different types of bitmap button implemented. They are Type 1, Type 2 and Type 3. Two of these types, Type 1 and Type 2, share a similar design concept, but it differ in its state handling.

Figure 13.3 shows the button appearance on different states. This type of buttons are useful when the button is used to trigger a process that require a long time to complete. For example, in the context of capturing an image, the button assumes a 'normal' state when no image has been captured. When pressed, the process commences, and the button transitions to an 'active' state, signifying that the process is underway. During this active state, the button is also 'disabled' to prevent any potential interruptions or inadvertent re-triggering of the process. Once the process is successfully completed, the button shifts to a 'disabled' state, indicating that the process cannot be re-initiated.

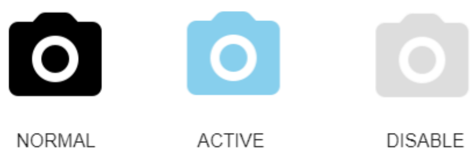


Figure 13.3: Type 1 Button State

The states for Type 2 button are shown in Figure 13.4. This type of button is used when the button is used to trigger a process that require toggeling. For

Figure 13.4: Type 2 Button State

instance, in the context of turning on a camera, the button assumes an 'off' state when the process has not been initiated and is available for activation. Upon pressing the button, the process initiates, and the button transitions to an 'on' state, indicating that the process is actively running. If necessary, pressing the button again will revert the state back to 'off', effectively terminating the ongoing process.

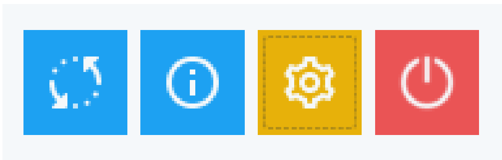


Figure 13.5: Example of Type 3 Button

The last type of bitmap buttons is Type 3. In terms of appearance, it differs from the previous two types. This button contains only bitmap image, without the text label, and coloured backgrounds. It does not have states, and it is used to trigger a very simple straightforward process, such as exiting the application or opening the settings panel. This type of buttons can be seen in the title panel (see Figure 13.5).

Handling Button Input

The process of handling button input is done by using event handling mechanism provided by wxWidgets. For a more detailed explanation on how the event handling mechanism works, please refer to wxWidgets Documentation [wxW].

In general, the process of handling button input is done by binding the button to a function. By pressing the button, an event is triggered, and the function is executed. Depending on the implementation, different button can trigger different function. For example, the button to turn on the camera will trigger a function to turn on the camera, while the button to capture an image will trigger a function to capture an image.

This process can be dony by assigning a unique ID to each button. By doing so the button can be identified and binded to a specific function. Following is a short example of how the button is binded to a function.

Updating Button State

A proper feedback mechanism is essential to ensure that the user is aware of the current state of the application. In the context of the application, the button state is used to provide feedback to the user on the current state of the application. For example, when the user interact with a button to turn on camera, the feedback to the user is the button state changes from 'off' to 'on'.

The process of updating button state is done by using event handling similar to the process of handling button input. Whenever an update is required, for example, after a button pressed, or after a process is completed, an event is triggered, and the button state is updated accordingly. This process is usually done by the controller component, as it is responsible for handling the application logic.

However it is also important to note that while updating component state is crucial, updating it too frequently can be detrimental to the application performance. Therefore, it is important to find a balance between the two.

13.1.3 Error Handling and Feedback Mechanisms

In real world scenarios, errors are inevitable. Therefore, it is important to implement a proper error handling mechanism to ensure that the user is guided through the application's workflow, even in scenarios where unexpected events occur. Effective error handling contributes to a more user-friendly and reliable user experience.

In the context of the application, the error handling mechanism is implemented by using a message dialog. A message dialog is a dialog that displays a message to the user, along with buttons for the user to click.Figure **??** shows an example of a message dialog.

The *ErrorEvent* can be utilized to display an error message to the user. When an exception is thrown during a process, an *ErrorEvent* will be created by the controller component and passed to the view component. The view component will then display the error message to the user by using a message dialog.

13.1.4 Image Panel

Displaying Image to the User

The image panel is the component that is responsible for displaying the image to the user. To display image to the user, the *UpdatePreviewEvent* is used. For example, when turning on the camera, a thread controller will initialized a thread which responsible for the process of capturing image from camera. For each iteration, the thread will capture an image and create the *UpdatePreviewEvent* and pass it to the view component. The image panel will catch the event and display the image to the user.

Handling Touch Input

touch input can be used ad another method of user input. The process of handling touch input is similar to the process of handling button input. It can be done by binding the touch event to a function. By doing so, whenever a touch event is triggered, the function will be executed.

The implementation via event by wxwidgets such as wxMouseEvent. Alternatively, the implementation can also be done by utilizing the *BasePanelWithTouch* class when implementing the views. This class already contains the implementation of touch event handling, and methods to handle the events are also included. For more information, please refer to project Documentation on A.6.

13.1.5 Navigating between different panel

For this application, multiple panels or views are implemented based on its functionality. For example, the capture panel is used to capture an image, while the calibration panel is used to perform camera calibration. Since multiple views are implemented, a proper method in displaying the views to the user is required.

Therefore to simplify the process, we have set that within the frame, only one view can be displayed, while others are kept hidden. This is done by using the *Show()* and *Hide()* method provided by wxWidgets. When navigating to a different view, the current view will be hidden, and the new view will be shown.

Figure 13.6 shows an example of navigating between different panel by using *Show()* and *Hide()* method.

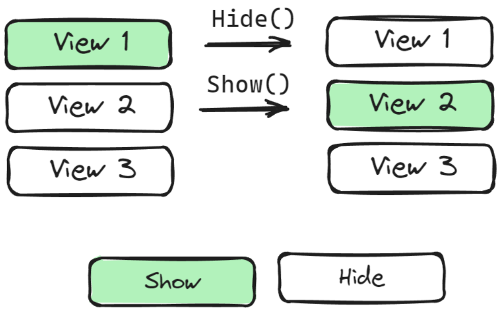


Figure 13.6: Example of navigating between different panel

13.2 Model Implementation

13.2.1 Model Overview

In this section, the overview of the model is provided. The model is responsible for storing the data and the application logic. Figure 13.7 shows the overview of the model.

The model consists of 5 main components, which are *SessionData*, *captureData*,

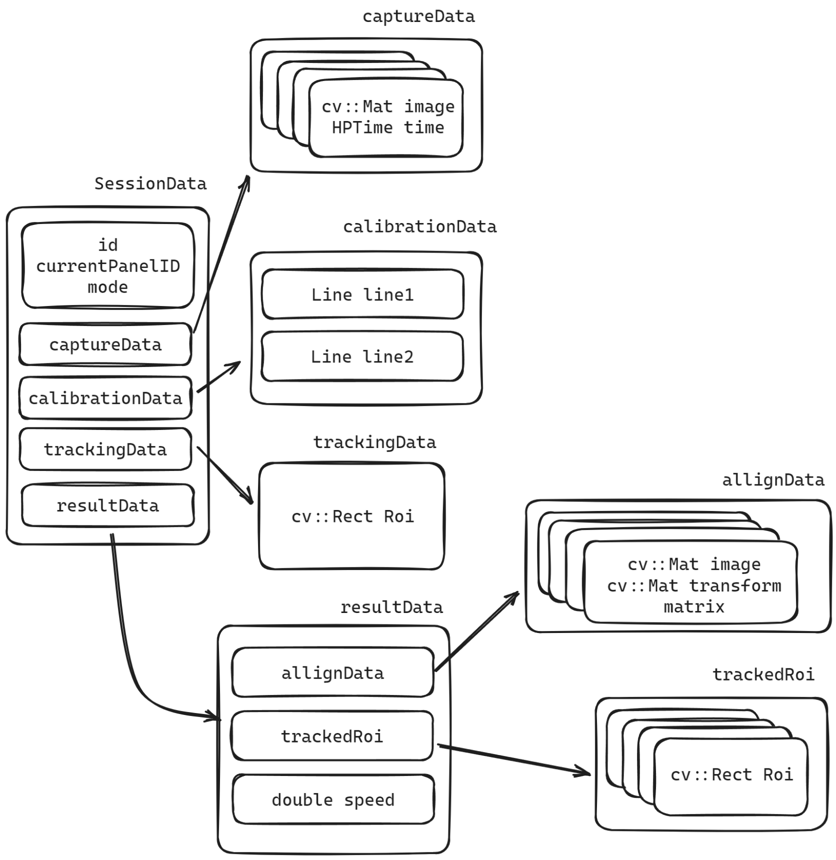


Figure 13.7: Overview of Model

calibrationData, *trackingData* and *resultData*. Each of these components is explained in detail in the following sections.

SessionData: Responsible for storing the current overall state of the application. The *id* variable describe the current id of the session, while *currentPanelID* provide information on current active panel to be rendered by the application. *Mode* describe on which calculation mode is choosen by the user. For more information on *mode*, please refer Section **??**.

CaptureData: This data stores a vector of original images captured by the camera. Alongside the image, this data type also responsible to store the time on which the image is captured, to be used for measurement purpose.

CalibrationData: Contains the result of calibration process, which are the variables *line1* and *line2*. For more information on these variables, please refer to Section **??**.

TrackingData: This data stores the initial region of interest (ROI) on which the tracked object is located.

ResultData: This data stores the result of the calculation process. Within this data, there are 3 variables, which are *allignData*, *trackedRoi* and *speed*. *allignData* stores the result of the allignment process, while *trackedRoi* stores the result of the tracking process. *Speed* variable is the value of the speed of the tracked object, which is calculated by using the result of the allignment process and the result of the tracking process.

13.3 Controller Implementation

13.3.1 Individual Controller

As mentioned previously in Section **??**, each view will have its own controller associated to it. By doing so, each controller will each have its own responsibility, which is to handle the application logic for the view. Furthermore, this design also helps in terms of application scaling. By having multiple controllers, the application logic can be divided into smaller parts, which makes it easier to maintain and scale.

However, a proper controller design guidelines are required to ensure that these controllers are implemented properly. Therefore, we have set up a set of guidelines that we have followed throughout the development process. These guidelines are as follows:

Only handle request and response

This guideline emphasizes that the controller should only handle request and response. It should not contain any business logic. Instead, the business logic should be implemented in the threads, which will be discussed in Section 13.3.2.

Provide similar endpoints

To ensure proper communication between the controller and the view, it is important to provide similar endpoints for each controller. By doing so, developers can easily identify the methods that are responsible to communcate between the views and the controller. To properly implement this guideline, all methods which are responsible for communication between the view and the controller should be named with the prefix *e_*, which stands for *endpoint*. For example *e_startCamera()*, *e_captureImage()* and *e_calibrate()*.

Only handle one view

This guideline emphasizes that the controller should only handle one view. This is to ensure that the controller is not overloaded with too many responsibilities. By doing so, the controller can be easily maintained and scaled.

Singular responsibility

Each endpoint should only handle one responsibility. For example *e_startCamera()* should only handle the process of starting the camera, and not other process such as capturing an image or calibrating the camera.

Handle error

All endpoints should be wrapped with a try-catch block to handle any error that might occur during the process. If an error occurs, an *ErrorEvent* will be created and passed to the view component. The view component will then display the error message to the user by using a message dialog. For more information on how the error handling mechanism works, please refer to Section 13.1.3.

Prevent handling outside active panel

This guideline emphasizes that the controller should only handle request and response when the view is active. This strict rule is implemented to prevent any unwanted behaviour that might occur when the view is not active. The *checkPrecondition()* provide method to check whether the view is active or not. If the view is not active, error message will be displayed to the user.

13.3.2 Threads and Thread Controller

To reduce the responsibility of the controller, the process of handling the business logic is delegated to the *threads*. By seperating the execution of business logic on different threads, it prevents the main thread or the GUI thread from being blocked. This is important to ensure that the application is responsive and does not freeze.

Additionally the threads also has the access to the thread pool, to take leverage of parallel execution whenever required. The *thread controller* is responsible to manage the threads by starting, stopping and joining the threads. Furthermore the thread controller also responsible to provide status on the current state of the threads, either it is running or not.

Table **??** shows the list of threads that are implemented within the application.

13.3.3 Request Handling Example

In this section, an example of how the request is handled by the controller is provided. A sequence diagram is used to illustrate the process, as shown in Figure 13.8.

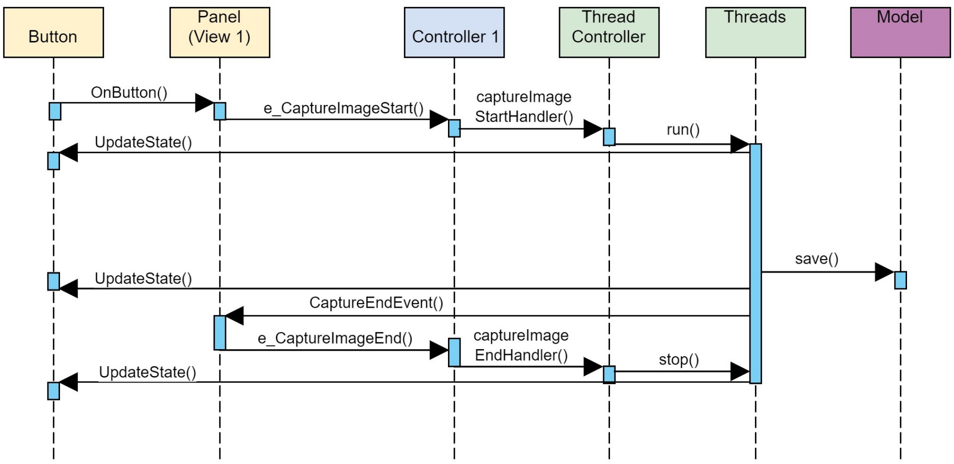


Figure 13.8: Example of Request Handling

With this example, the request starts by pressing a button. This action will trigger an event, which will be handled by the *View1*. The view will then request for an initialization of process or thread via the endpoint *e_CaptureImageStart()*. The request will be forwarded to the *ThreadController* and a threads to capture image is spawned. The state is updated to signal the succession of the request. The thread will run on its own, and when it finished capturing image, the data is saved to the model via the *save()* method. The state is once again updated and an event signaling the completion of the process is sent to the view. With this event, the view signal the controller to stop the thread via the endpoint *e_CaptureImageStop()*. The thread is then stopped and joined. The state is updated to signal the completion of the process.

13.4 Algorithm Improvement

13.4.1 Image Alignment

Based on previous work [BMS23], the image alignment process is done by using a combination of feature detection algorithm and feature matching algorithm. The feature detection algorithm is utilized to detect features or keypoints between two images, the result are matched with each other and the matching result is used to calculate the transformation matrix, which are then used to align the images. Figure **??** shows the overall process of image alignment process.

However as stated by the author, the overall process can be time consuming, especially when large image size is involved. To further understands which part is causing the process to be time consuming, each steps of the process is timed. The result is shown in Table **??**. From the result, it can be seen that the feature detection process is the most time consuming process, which takes up to 90% of the overall process.

Since the process of aligning an array of images, require the first image to be used for each iteration, we can somehow compute feature detection on the first image once, and cache the result, to be used for subsequent iterations. This will reduce the time taken for the feature detection process, and hence reduce the overall time taken for the process.

Figure **??** shows the overall process of image alignment process with caching. Table **??** shows the result of the process with caching. From the result, it can be seen that the time taken for the feature detection process is reduced by 90%, which in turn reduce the overall time taken for the process by 90%.

Furthermore, with the implementatio of Threadpool as stated in **??**, the process can be further improved by utilizing parallel execution. Table **??** shows the result of the process with caching and threadpool. From the result, it can be seen that the time taken for the feature detection process is reduced by 90%, which in turn reduce the overall time taken for the process by 90%.

13.4.2 Custom lane

Previous work [BMS23], stated that in measurement of object distance from camera, the distance between the two lane is required, and in Germany the distance is usually 3.5 m. However in real world scenario, the distance between the two lane can vary, and thus reducing the accuracy of the measurement. The author also mentioned the importance of selecting the right lane that represent the actual distance between the two lane. A small error in selecting the lane can result in a large error in the measurement.

Therefore, to further simplify the process of selecting the correct line, a much simpler approch is implemented. A custom object of known dimension is placed within the camera view and will act as a reference object (see Figure **??**). With this alternative, it is now easier for user to compute the distance between the two lane, as the user only need to select the lane that is closest to the reference object.

13.4.3 Distance Measurement

Distance measurement, is based on the work of Javadi et al. [JDP19], will act as an alternative speed calculation method to the lane measurement which is discussed previously on Section **??**. This method of measurement will act as a redundant method, which is one of the patterns in fault tolerance implementation.

The process of measurement is almost similar, to lane measurement, as shown in Figure **??**. The process involved first the selection of two lines which represents the start and end point of the measurement. Hoewver, the real world distance between the lines must known beforhand, and to simplify this step, an object of known length, is placed within the camera view, and will act as a reference object (see Figure **??**).

With the result of tracking, we can analyze at what frame, the object cross the start line and the end line. With this information, the time it crossed the start line (t_{start}) and the time it crossed the end line (t_{end}) can be calculated. With known object length (l_{object}), the speed estimation can be done by using the following formula:

$$Speed = \frac{l_{object}}{t_{end} - t_{start}} \tag{13.1}$$

This method is a much straightforward approach compared to the lane measurement method, however as stated by Javadi et al. [JDP19], this method can cause inaccuracy within calculation, as the time taken the object cross the start line and the end line cannot be exactly determined.

This measurement method is implemented in *DistSpeedMeasurement* class. For more information please refer the project documentation on A.6.

14 Testing and Maintenance

Software testing and maintenance are essential activities in the software development cycle. Software testing is the process of verifying and validating that a software is funtioning correctly and meeting its requirements and specifications [ibm] [Ham23].

This phase enables the early detection and resolution of bugs and errors, ensuring a smoother and more efficient delivery of the final product [Ham23]. Additionally, thorough testing guarantees the reliability and security of the software, particularly in applications handling sensitive data or with critical functionalities [Ham23].

In software testing, there are numerous available methods and techniques to ensure the quality of the software. Some of the most common ones are unit testing, integration testing, system testing, and acceptance testing [ibm]. For this project, we will utilize unit testing, which will be discussed in detail in Section 14.1.

14.1 Unit Testing

Unit testing is a type of software testing that focuses on individual units or components of a software system and the purpose of performing this test is to validate that each unit of the software works as intended and meets the requirements [gee23b].

They are designed to validate the smallest possible unit of code, such as a function or a method, and test it in isolation from the rest of the system,which will allows developers to quickly identify and fix any issues early in the development process, improving the overall quality of the software and reducing the time required for later testing [gee23b].

In writing unit test, it is advised to include multiple number of scenarios to ensure that the code is working as expected. This includes testing the code with valid and invalid inputs, as well as edge cases [Var22]. Additionally, to further simplify the process of testing, it is recommended to use a unit testing framework, which provides a set of tools and utilities to automate the process of writing and executing unit tests [Var22]. For this project, we will be using the GTest framework, which is a C++ unit testing framework developed by Google.

14.1.1 Example of Good Unit Test

An example of a good unit test can be found in the Appendix A.8. This unit test is written in C++ using the GTest framework. The code demonstrates a simple bank account management system, encapsulated within the *BankAccount* class. This class offers functionalities like depositing (*deposit()*), withdrawing (*withdraw()*), and checking the account balance (*getBalance()*).

To ensure robustness, the code has been augmented with exception handling mechanisms. Specifically, when an attempt is made to deposit a negative amount or withdraw more than the available balance, the methods will throw appropriate exceptions, either std::invalid_argument or std::runtime_error.

The unit tests will thoroughly examine the behavior of these functionalities. The *BankAccountTest.Deposit* case validates that funds are successfully deposited, asserting that the balance matches the expected value. Similarly, *BankAccountTest.Withdraw* verifies correct withdrawal behavior by comparing the resultant balance after a withdrawal operation.

In *BankAccountTest.WithdrawTooMuch*, an attempt to withdraw an excessive amount triggers an exception. The test checks whether this exception is properly thrown, ensuring the balance remains unchanged. Lastly, *BankAccountTest.DepositNegative* evaluates the system's response when attempting to deposit a negative amount. This test expects a std::invalid_argument exception to be raised. The subsequent assertion checks that the account balance remains unaffected.

14.2 Maintenance

Software maintenance can be described as the modification of a software product after it has been delivered, which includes correcting faults, enhancing performance, and adapting the product to a modified environment [BSM04]. This phase includes tasks like bug fixing, adding new features, improving performance, and ensuring compatibility with new hardware or software [gee23a].

Software maintenance can be categorized into four types [BSM04][gee23a]:

Corrective Maintenance: This addresses errors or bugs in the software, ensuring it functions properly. Swift resolution of issues enhances the software's reliability and user satisfaction.

Preventive Maintenance: This involves making changes, upgrades, or adaptations to prevent potential problems in the future. It helps identify and correct latent errors before they lead to disruptions.

Perfective Maintenance: After a software is introduced, user needs evolve, prompting the addition of new features or the improvement of existing ones. This ensures the software remains relevant and efficient.

Adaptive Maintenance: This focuses on adapting the software to changes in technology, policies, and regulations. It includes adjustments for new hardware, operating systems, and compliance requirements.

For our project, following maintenance activities will be performed:

Bug Fixing: This involves correcting any errors or bugs in the software, ensuring it functions properly. Swift resolution of issues enhances the software's reliability and user satisfaction.

Adding New Features: This involves adding new features to the software to improve its functionality and user experience. We are interested to explore other algorithms which can improve either the accuracy of the calculation, or the ease of use of the software.

User Interface Improvements: This involves improving the user interface of the software to improve its usability and user experience. To developed a more user-friendly interface, numerous iteration based on user feedback will be performed.

15 Conclusion

Part III

Indexes and Appendix

List of Figures

3.1 Pahl and Beitz's Design Process [Pah07, 130] 10
3.2 Original Prusa i3 MK3S+ 11
3.3 Example View of PrusaSlicer 12

4.1 Planning and Task Clarification [Pah07, 146] 15
4.2 Checklist for Establishing the Prototype's Requirements [Pah07, 149] 17

5.1 Steps in Conceptual Design [Pah07, 160] 25
5.2 Result of Abstraction Process 26
5.3 Breaking down the overall function into sub-functions [Pah07, 32] 27
5.4 Overall Function of the System 27
5.5 Sub-Functions of the System 28
5.6 Sub-Functions of the System (Final) 28
5.7 Morphological Chart with Solution Variants 32
5.8 Sketch of Solution Variant 1 34
5.9 Sketch of Solution Variant 2 35
5.10 Sketch of Solution Variant 3 36
5.11 Sketch of Solution Variant 4 37
5.12 Sketch of Solution Variant 5 38
5.13 Sketch of Solution Variant 6 39
5.14 Sketch of Solution Variant 7 40
5.15 Sketch of Solution Variant 8 41
5.16 Selection Chart for Solution Variants 42

6.1 Steps in Embodiment Design [Pah07, 229] 45
6.2 Design guidelines for 3D printing [And22] 49

6.3 Preliminary Design Variant 2 50
6.4 Views of Preliminary Design Variant 2 50
6.5 Body Components of Preliminary Design Variant 2 50
6.6 Placement of inner components for Variant 2 51
6.7 Methods to secure 3D-printed components [Her20] 52
6.8 Quick release plate 53
6.9 Preliminary Design Variant 3 54
6.10 Placement of inner components for Variant 3 55
6.11 Bumps on the back cover 55
6.12 Battery Placement 56
6.13 Preliminary Design Variant 6 57
6.14 Placement of inner components for Variant 3 57
6.15 Handle Grip 58
6.16 Placement of handle grip and quick release plate 58
6.17 Preliminary Design Variant 7 59
6.18 Placement of inner components for Variant 7 60
6.19 Placement of quick release plate 60

7.1 Steps in the detail design process 70
7.2 Power Switch 71
7.3 Position of the camera component 71
7.4 Protective bump for camera 72
7.5 Protective bump for screen 72
7.6 The LAN port 73
7.7 Position of the LAN port 74
7.8 Germany Police Logo [bun] 74
7.9 Result of recolor 75

8.1 Printed Parts 77
8.2 The installed threaded insert 78
8.3 The Switch Holder 79
8.4 The installed switch 79
8.5 The LAN Port Slot 79
8.6 The installed LAN port 80
8.7 The Camera Module Slot 80

8.8 The installed camera module 81
8.9 The Battery Holder 81
8.10 The installed battery 82
8.11 The Raspberry Pi Slot 82
8.12 The Screen Slot 83
8.13 The installed screen and top cover 83
8.14 The Final Product 84

10.1 Waterfall Model 90
10.2 Model-View-Controller Architecture [Gar23, 46] 91

11.1 Project Requirements (1/2) 94
11.2 Project Requirements (2/2) 95

12.1 Software Architecture 97
12.2 Thread Pool Architecture [Eug23] 98
12.3 Wireframe 102
12.4 Example of Color Combination with 60-30-10 Rule [M.23] 103
12.5 Color Combination 104
12.6 Roboto Font 104
12.7 Task Flow 106
12.8 Wireflow 106

13.1 Main Layout 109
13.2 Bitmap Button 110
13.3 Type 1 Button State 110
13.4 Type 2 Button State 111
13.5 Example of Type 3 Button 111
13.6 Example of navigating between different panel 114
13.7 Overview of Model 115
13.8 Example of Request Handling 118

List of Tables

4.1 Requirement List (1/2) 22
4.2 Requirement List (2/2) 23

5.1 Classification Scheme for Working Principles 31

6.1 Printing cost for Variant 2 53
6.2 Manufacturing cost for Variant 2 54
6.3 Printing cost for Variant 3 56
6.4 Manufacturing cost for Variant 3 56
6.5 Printing cost for Variant 6 59
6.6 Manufacturing cost for Variant 6 59
6.7 Printing cost for Variant 7 60
6.8 Manufacturing cost for Variant 7 61
6.9 Weighting Factors for Evaluation Criteria 63
6.10 Value Scale for Evaluation [Pah07, 115] 64
6.11 Value Scale for Weight Distribution 64
6.12 Value Scale for Device Weight 64
6.13 Value Scale for Device Size 64
6.14 Value Scale for Ease of Assembly 65
6.15 Value Scale for Swappable Parts 65
6.16 Technical Evaluation of Preliminary Design Variants (1/2) 67
6.17 Technical Evaluation of Preliminary Design Variants (2/2) 67
6.18 Economic Evaluation of Preliminary Design Variants 67
6.19 Total Rating of Preliminary Design Variants 68

7.1 Sizing guide for threaded inserts [ruta][rutb] 73

8.1 Printing Time and Filament Used 76

8.2 Total Printing Cost 84
8.3 Total Material Cost 85

Bibliography

[Alg23] ALGOREPUBLIC: *The importance of User Interface Design in app development*. https://www.linkedin.com/pulse/importance-user-interface-design-app-development-algorepublic/, May 2023.

[And22] ANDERSTHOESTESEN, DDDIMENSION: *Complete Design Guide for 3D printing*. https://www.ddd-imension.com/en/post/complete-design-guide-for-3d-printing, Mar 2022.

[Ang] ANGELES, MIKE: *Wireframing user flow with wireflows: Wireframing Academy: Balsamiq*. https://balsamiq.com/learn/articles/wireflows/.

[Bat] BATES, HOWIE: *Materials for 3D printing by fused deposition*. https://www.materialseducation.org/educators/matedu-modules/docs/Materials_in_FDM.pdf.

[BMS23] BIN MOHD SABTU, MUHAMMAD HAZIQ: *Developing Speed Measurement Algorithm with OpenCV and Raspberry Pi*. 2023.

[Bro23] BROWNE, CAMREN: *What are user flows in UX design? [full beginner's guide]*. https://careerfoundry.com/en/blog/ux-design/what-are-user-flows/, Apr 2023.

[BSM04] BHATT, PANKAJ, GAUTAM SHROFF and ARUN K. MISRA: *Dynamics of Software Maintenance*. SIGSOFT Softw. Eng. Notes, 29(5):1–5, sep 2004.

[bun] BUNDESPOLIZEI, DIE: *Die bundespolizei*. https://www.bundespolizei.de/Web/DE/05Die-Bundespolizei/03Organisation/Organisation_node.html.

[BZJ14] BEN-ZAHIA, MOHAMED A. and IBRAHIM JALUTA: *Criteria for selecting software development models*. In *2014 Global Summit on Computer Information Technology (GSCIT)*, pages 1–6, 2014.

[Cou23] COURSERA: *What is Ui Design? definition, tips, best practices*. https://www.coursera.org/articles/ui-design, Jun 2023.

[Dai22] DAI, ANNIE: *The beginners guide to user flow in UX design*. https://uxplanet.org/the-beginners-guide-to-user-flow-in-ux-design-2022-7a0ab8c7d0bd, May 2022.

[DSPB23] DREISSIG, MARIELLA, DOMINIK SCHEUBLE, FLORIAN PIEWAK and JOSCHKA BOEDECKER: *Survey on LiDAR Perception in Adverse Weather Conditions*, 2023.

[Eug23] EUGEN, PARASCHIV: *Introduction to thread pools in Java*. https://www.baeldung.com/thread-pool-java-and-guava, Aug 2023.

[Fle21] FLECK, RENEE: *10 fundamental UI design principles you need to know*. https://dribbble.com/resources/ui-design-principles, Dec 2021.

[Flo22] FLORIDO, DANIEL: *Key characteristics of good UI design – according to 8 experts*. https://www.uxpin.com/studio/blog/good-ui-design-characteristics/, Jun 2022.

[Fly23] FLYGUYS: *Lidar vs radar*. https://flyguys.com/lidar-vs-radar/, Jul 2023.

[FP22] FITZ-PATRICK, MOLLY: *The UX designer's guide to typography*. https://www.interaction-design.org/literature/article/the-ux-designer-s-guide-to-typography, Oct 2022.

[FZ16] FERRIS, KEVIN and SONYA ZHANG: *A Framework for Selecting and Optimizing Color Scheme in Web Design*. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 532–541, 2016.

[Gar23] GARCÍA, RAÚL FERRER: *iOS Architecture Patterns*. Apress, 2023.

[GD16] GEOGY, MANJU and ANDHE DHARANI: *A Scrutiny of the Software Requirement Engineering Process*. Procedia Technology, 25:405–410, 2016.

[Gee20] GEEKFORGEEK: *Thread pools in Java*, Jul 2020.

[gee23a] GEEKSFORGEEKS: *Software engineering: Software maintenance*. https://www.geeksforgeeks.org/software-engineering-software-maintenance/, Jul 2023.

[gee23b] GEEKSFORGEEKS: *Unit testing: Software testing*. https://www.geeksforgeeks.org/unit-testing-software-testing/, Feb 2023.

[GGA18] GORDEEV, EVGENIY G., ALEXEY S. GALUSHKO and VALENTINE P. ANANIKOV: *Improvement of quality of 3D printed objects by elimination of microscopic structural defects in fused deposition modeling*. PLOS ONE, 13(6):e0198370, June 2018.

[Gup23] GUPTA, SAKSHI: *How to create a wireframe: Step-by-step guide*, Jul 2023.

[Ham23] HAMILTON, THOMAS: *What is software testing? definition*. https://www.guru99.com/software-testing-introduction-importance.html, Sep 2023.

[Hau] HAUSEN, DORIS: *Classic Waterfall Model in Software Engineering*. https://www.medien.ifi.lmu.de/lehre/ws0607/mmi1/essays/Doris-Hausen.xhtml.

[Her20] HERMANN, STEFAN: *Helicoils, threaded insets and embedded nuts in 3D prints - Strength*. https://www.cnckitchen.com/blog/helicoils-threaded-insets-and-embedded-nuts-in-

3d-prints-strength-amp-strength-assessment, May 2020.

[Her23] HERMANN, STEFAN: *Tips Tricks for heat-set inserts used in 3D printing*. https://www.cnckitchen.com/blog/tipps-amp-tricks-fr-gewindeeinstze-im-3d-druck-3awey, May 2023.

[Hos] HOSSAIN, AL-EMRAN: *Advantages and disadvantages of radar systems*. https://www.linkedin.com/pulse/advantages-disadvantages-radar-systems-al-emran-hossain/.

[Hul20] HULL, ROB: *Police forces begin using "Next generation" speed guns on UK roads*. https://www.thisismoney.co.uk/money/cars/article-9013677/Police-forces-begin-using-generation-speed-guns-UK-roads.html, Dec 2020.

[ibm] IBM: *What is software testing and how does it work?* https://www.ibm.com/topics/software-testing.

[jdb] JDB, JDB: *Board index*. https://forums.raspberrypi.com/viewtopic.php?t=243421.

[JDP19] JAVADI, SALEH, MATTIAS DAHL and MATS I. PETTERSSON: *Vehicle speed measurement model for video-based systems*. Computers Electrical Engineering, 76:238–248, June 2019.

[KBM] KARL-BRIDGE-MICROSOFT: *Thread pools - win32 apps*. https://learn.microsoft.com/en-us/windows/win32/procthread/thread-pools.

[Koz23] KOZON, TOMASZ: *Understanding the concept of clean architecture*. https://boringowl.io/en/blog/unveiling-the-core-principles-of-clean-architecture, Jun 2023.

[Kus22] KUSHAHRIN, HAIKAL: *The zwicky box: A powerful method for problem solving and creativity*. https://nesslabs.com/zwicky-box, Dec 2022.

[KV20] Krastev, Georgi and Valentina Voinohovska: *Smart Mobile Application for Public Transport Schedules - Logical Model*. TEM Journal, 9:541–545, 05 2020.

[Lab21] Labidi, Younes: *Raspberry pi shutdown reboot button*. https://blog.berrybase.de/blog/2021/05/10/raspberry-pi-shutdown-reboot-button/, Oct 2021.

[LOK21] Lewandowska, Anna and Agnieszka Olejnik-Krugly: *Do Background Colors Have an Impact on Preferences and Catch the Attention of Users?* Applied Sciences, 12(1):225, December 2021.

[M.23] M., Bryson: *Principles of color in Ui Design*. https://uxplanet.org/principles-of-color-in-ui-design-43708d8512d8, Aug 2023.

[Mot22] Mott, Nathaniel: *Google introduces reading-optimized Roboto Serif typeface*. https://www.pcmag.com/news/google-introduces-reading-optimized-roboto-serif-typeface, Feb 2022.

[NST23] Novianti, Idha, Joko Soebagyo and Wildan Toyib: *Diagnosis of Maths Teaching Efficacy Beliefs Using Expert System*, 2023.

[OLWW04] Ou, Li-Chen, Ming Luo, Andree Woodcock and Angela Wright: *A study of colour emotion and colour preference. Part I Colour emotions for single colours*. Color Research Application, 29:232 – 240, 06 2004.

[Pah07] Pahl, Gerhard: *Engineering design*. Springer-Verlag London Limited, 2007.

[Pau20] Paun, Goran: *Designing with efficiency: How familiarity can enhance experiences*. https://www.forbes.com/sites/forbesagencycouncil/2020/10/02/designing-with-efficiency-how-familiarity-can-enhance-experiences/?sh=355c1d873428, Oct 2020.

[Poo] POOL, NORD: *See yearly day-ahead prices.* https://www.nordpoolgroup.com/en/Market-data1/Dayahead/Area-Prices/ALL1/Yearly/?view=table.

[Pro] PROLASER. https://www.prolaser4.com/en/12-prolaser-4.

[Prua] PRUSA, JOSEF: *Original prusa i3 mk3s+.*

[Prub] PRUSA, JOSEF: *Prusament Pla pristine white 1kg: Original Prusa 3D-Drucker Direkt von Josef Prusa.* https://www.prusa3d.com/de/produkt/prusament-pla-pristine-white-1kg/.

[rad] RADAR, POLICE: *Police RADAR.* http://hyperphysics.phy-astr.gsu.edu/hbase/Sound/radar.html.

[Rad23] RADAR, STALKER: *Stalker II.* https://www.stalkerradar.com/police-radar/police-radar-gun/, Jun 2023.

[Rah22] RAHUL: *User flow and task flow explained.* https://bootcamp.uxdesign.cc/user-flow-and-task-flow-explained-bf229332a16d, Nov 2022.

[RM22] RACHMA, NUR and ITAWIRANDA MUHLAS: *Comparison Of Waterfall And Prototyping Models In Research And Development (RnD) Methods For Android-Based Learning Application Design.* Jurnal Inovatif : Inovasi Teknologi Informasi dan Informatika, 5(1):36, August 2022.

[Roy87] ROYCE, W. W.: *Managing the Development of Large Software Systems: Concepts and Techniques.* In *Proceedings of the 9th International Conference on Software Engineering*, ICSE '87, page 328–338, Washington, DC, USA, 1987. IEEE Computer Society Press.

[ruta] RUTHEX, RUTHEX: *Ruthex 1/4" Gewindeeinsatz.* https://www.ruthex.de/collections/gewindeeinsatze/products/ruthex-gewindeeinsatz-1-4-

kameragewinde-20-stuck-rx-1-4-20x12-7-messing-gewindebuchsen-fur-3d-druck-1.

[rutb] RUTHEX, RUTHEX: *Ruthex m2,5 Gewindeeinsatz*. https://www.ruthex.de/collections/gewindeeinsatze/products/ruthex-gewindeeinsatz-m2-5-70-stuck-rx-m2-5x5-7-messing-gewindebuchsen.

[SDCR20] SAWYER, BEN D., JONATHAN DOBRES, NADINE CHAHINE and BRYAN REIMER: *The great typography bake-off: comparing legibility at-a-glance*. Ergonomics, 63(4):391–398, March 2020.

[Sig23] SIGNALS, KUSTOM: *Directional Talon*. https://kustomsignals.com/handheld-radar/directional-talon, May 2023.

[Sim23] SIMPLILEARN: *What is requirement analysis [with requirement analysis example/sample inside]*, Sep 2023.

[Sta23] STATIS, DE: *Driver-related causes of accidents involving personal injury*. https://www.destatis.de/EN/Themes/Society-Environment/Traffic-Accidents/Tables/driver-mistakes.html, Jul 2023.

[Ste] STEPIEN, MARTY: *How does MVC follow separation of concerns?* https://goformarty.github.io/MVC-SoC/.

[Sup] SUPPLY, DANA SAFETY: *Kustom Signals Talon*. https://danasafetysupply.com/kustom-signals-talon-ii-law-enforcement-radar-gun-directional-motorcycle-mount-option-hand-held-or-dash-mount-corded/.

[SYM20] SUNARDI, SUNARDI, ANTON YUDHANA and GHUFRON ZAIDA MUFLIH: *Sistem Prediksi Curah Hujan Bulanan Menggunakan Jaringan Saraf Tiruan Backpropagation*. JURNAL SISTEM INFORMASI BISNIS, 10(2):155–162, November 2020.

[SZA14] SARKER, IQBAL and KHALID ZINNAH APU: *MVC Architecture Driven Design and Implementation of Java Framework for Developing Desktop Application*. International Journal of Information Technology, 7:317–322, 09 2014.

[Tec23] TECH, LASER: *LTI 20/20 TruVISION*. https://lasertech.com/product/truvision-photo-video-lidar-speed-measurement-device/, Jul 2023.

[Tri23] TRIGO, ALVARO: *What is a Wireflow in UX design*. https://alvarotrigo.com/blog/wireflows/, Oct 2023.

[Var22] VARTANIAN, ERICA: *All about unit testing: 11 best practices and overview*. https://www.educative.io/blog/unit-testing-best-practices-overview#unit-testing-benefits, Mar 2022.

[Wal23] WALKER, RICKIE: *Software requirements analysis*. https://appmaster.io/blog/software-requirements-analysis, Jan 2023.

[Whi23] WHITE, LUCAS: *UI and UX Design: Wireframe*. https://www.codecademy.com/resources/docs/uiux/wireframe, Sep 2023.

[wxW] WXWIDGETS: *Event handling*. https://docs.wxwidgets.org/3.0/overview_events.html.

[YM23] YAHYA, NORZARIYAH and SITI SARAH MAIDIN: *Hybrid agile development phases: the practice in software projects as performed by software engineering team*. Indonesian Journal of Electrical Engineering and Computer Science, 29(3):1738, March 2023.

A Appendix

A.1 Sketches of Working Principles

A.1.1 Screen Orientation

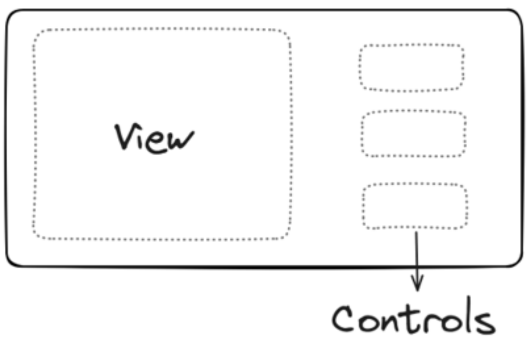
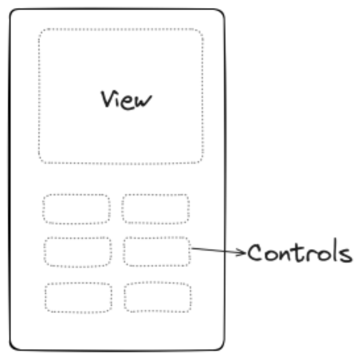
2.1 Landscape	2.2 Portrait
	

Table A.1: Screen Orientation

A.1.2 Battery Type

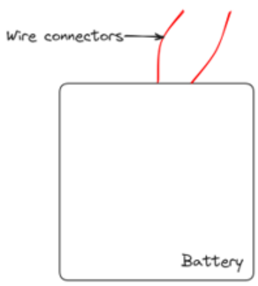
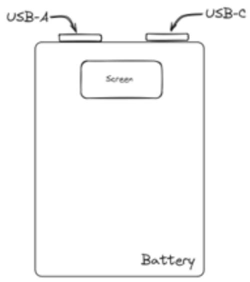
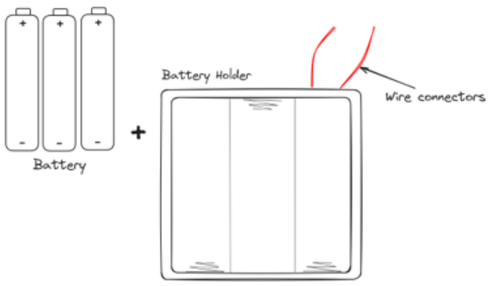
3.1 Battery Pack	3.2 Power Bank	3.3 AAA Batteries with Battery Holder
		

Table A.2: Battery Type

A.1.3 Components Placement

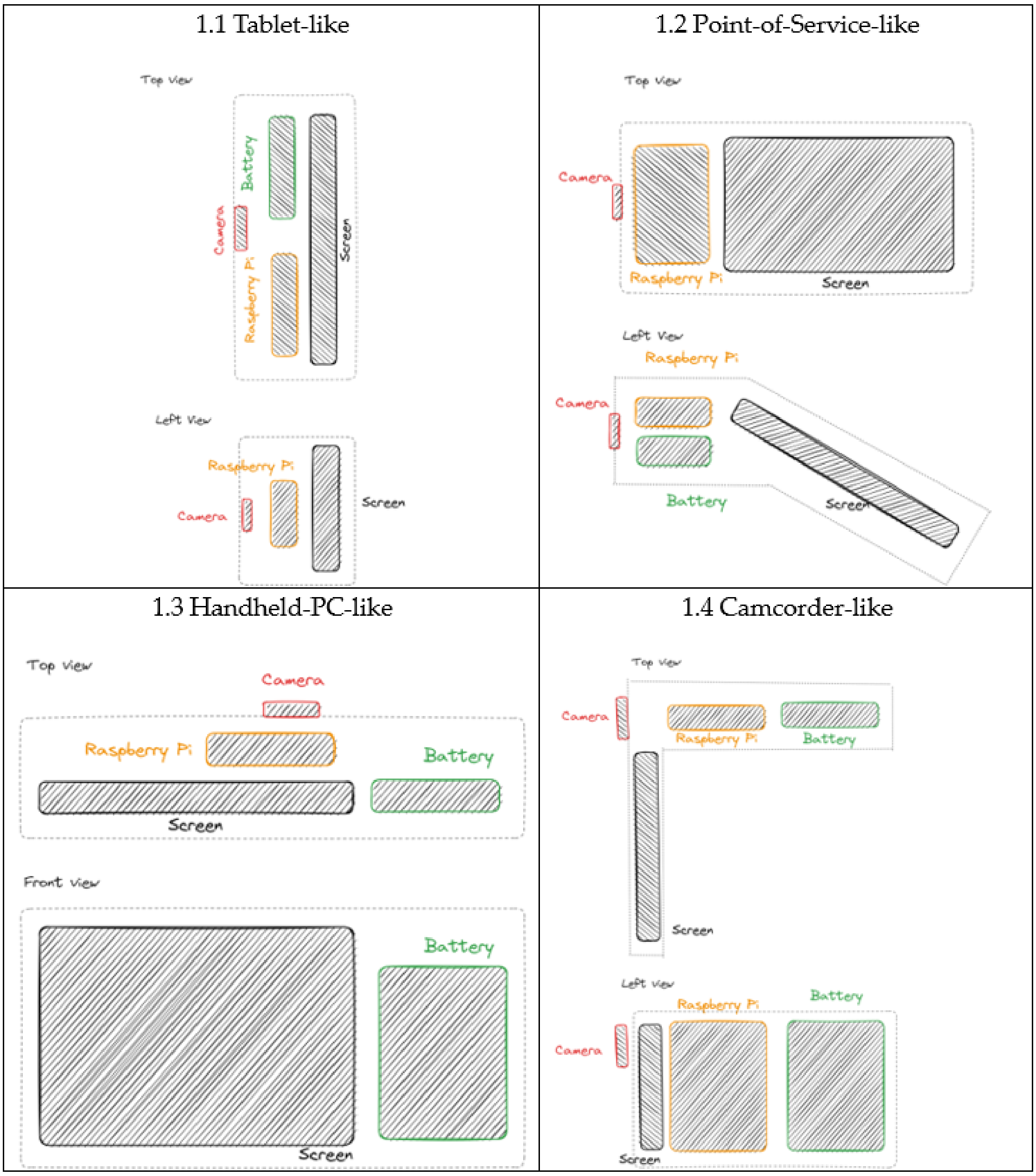


Table A.3: Components Placement

A.1.4 Body Type

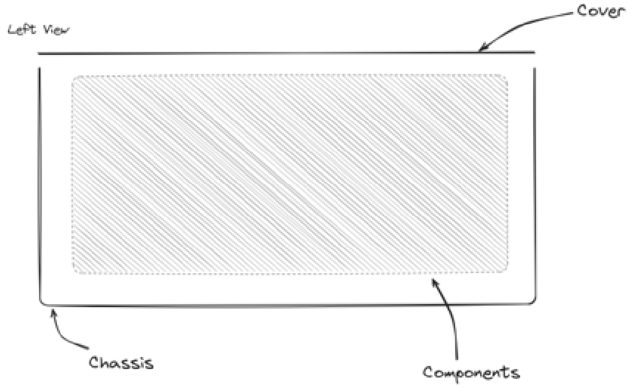
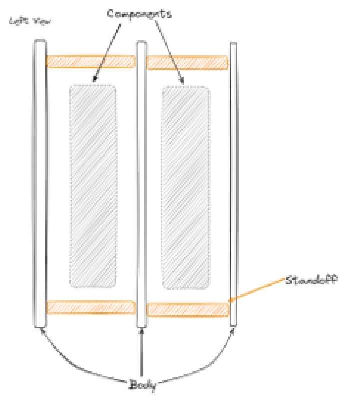
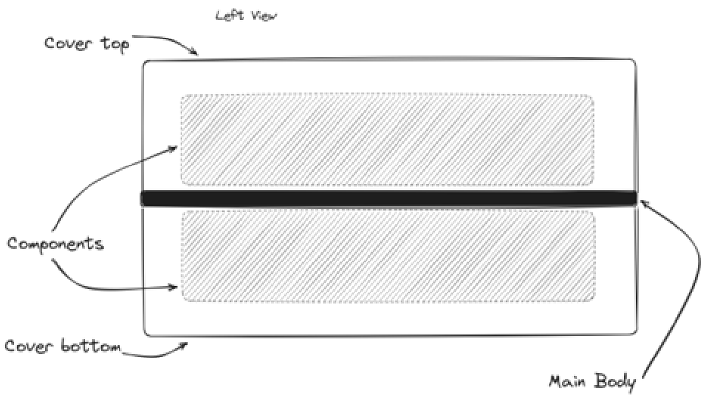
<table>
<tr>
<td>4.1 Bowl

</td>
<td>4.2 Skeleton

</td>
</tr>
<tr>
<td>4.3 Sandwich

</td>
<td></td>
</tr>
</table>

Table A.4: Body Type

A.1.5 Handling

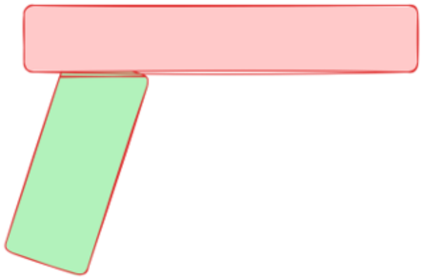
5.1 Body Grip	5.2 Bump Grip	5.3 Pistol Grip
		

Table A.5: Handling

A.1.6 External Mounting

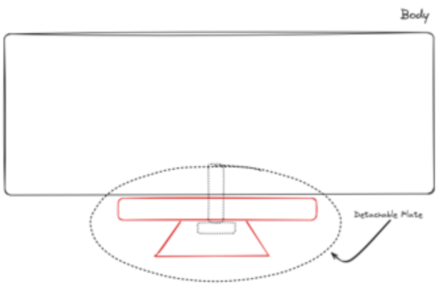
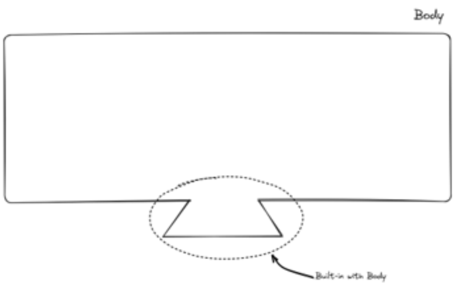
6.1 Detachable Plate	6.2 Built-in Mounting Plate
	

Table A.6: External Mounting

A.1.7 Control Mechanism

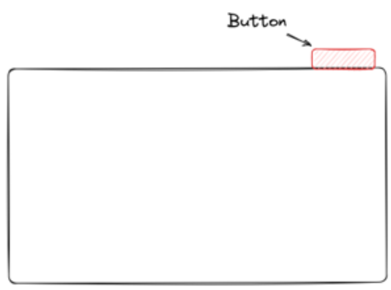
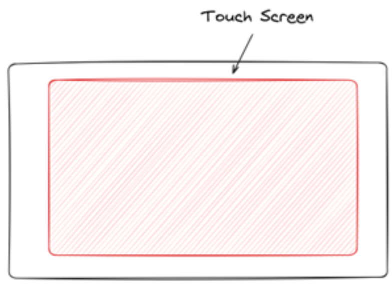
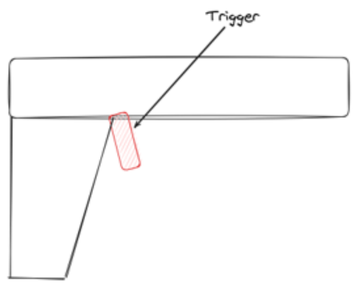
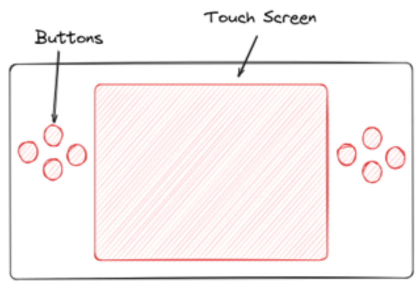
7.1 Button	7.2 Touch Screen
 	
7.3 Trigger	7.4 Touch and Button
 	

Table A.7: Control Mechanism

A.2 CAD Drawings

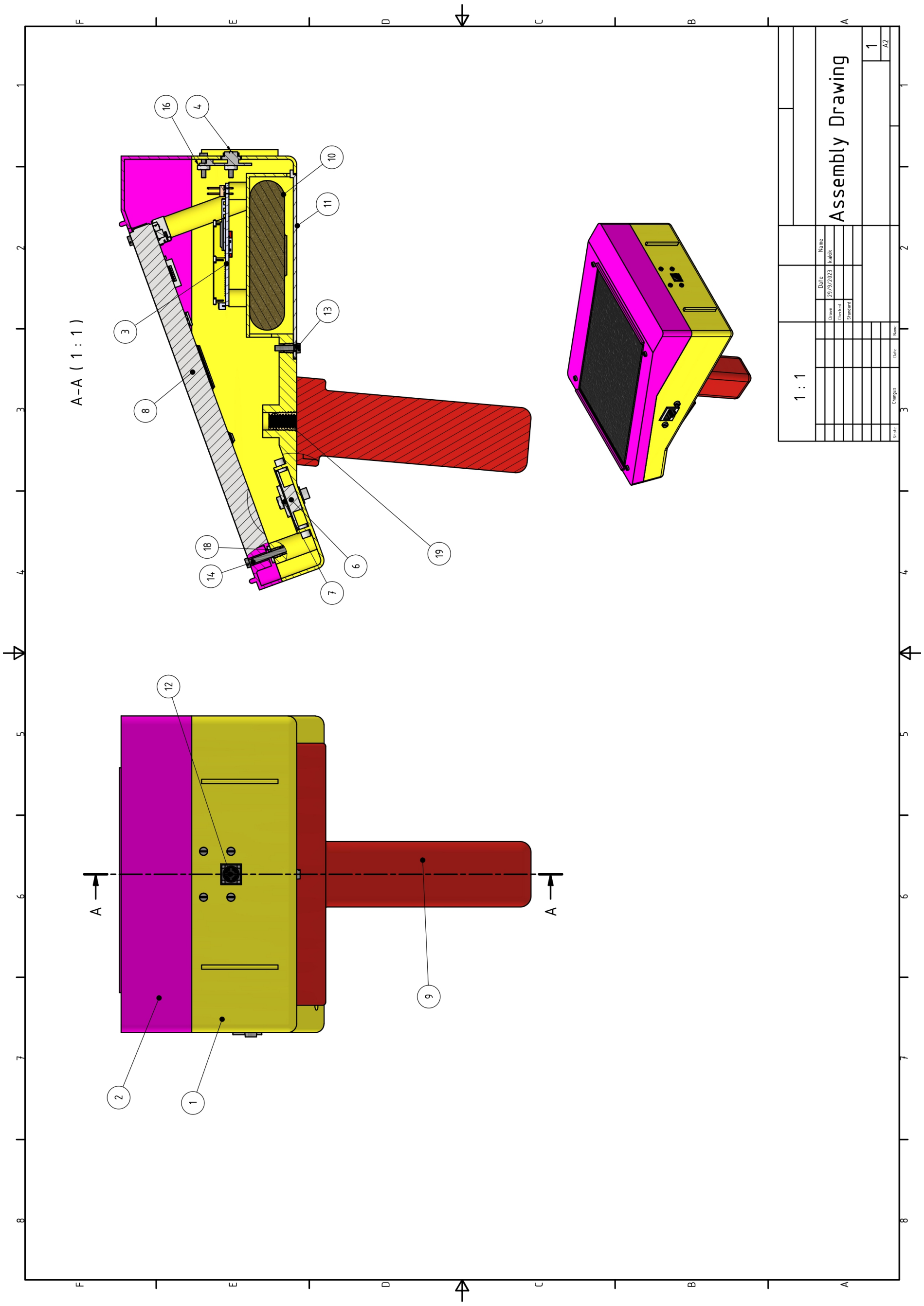


Figure A.1: Assembly Drawing

PARTS LIST			
ITEM	QTY	PART NUMBER	DESCRIPTION
1	1	Main Body	
2	1	Top Cover	
3	1	Raspberry Pi 4	
4	1	Camera Module	
5	1	LAN Port	
6	1	Switch	
7	1	Switch Cover	
8	1	Screen	
9	1	Pistol Grip	
10	1	Battery	Veektomx
11	1	Battery Cover	
12	4	Screw DIN 84 M2x10	DIN 84 (ISO 1207) - A2-Cylinder head screws, ISO 1207
13	7	Screw DIN 84 M2.5x10	DIN 84 (ISO 1207) - A2-Cylinder head screws, ISO 1207
14	4	Screw DIN 84 M2.5x20	DIN 84 (ISO 1207) - A2-Cylinder head screws, ISO 1207
15	2	Screw DIN 84 M3x10	DIN 84 (ISO 1207) - A2-Cylinder head screws, ISO 1207
16	4	Nut M2	DIN 934
17	2	Nut M2.5	DIN 934
18	9	Threaded Insert M2.5	Ruthex
19	3	Threaded Insert 1/4"	Ruthex
20	2	Camera Screw 1/4"	

1 : 1

	Date	Name
Drawn	29/9/2023	kakik
Checked		
Standard		

Bill of Material

2

A4

State | Changes | Date | Name

Figure A.2: Bill of Materials

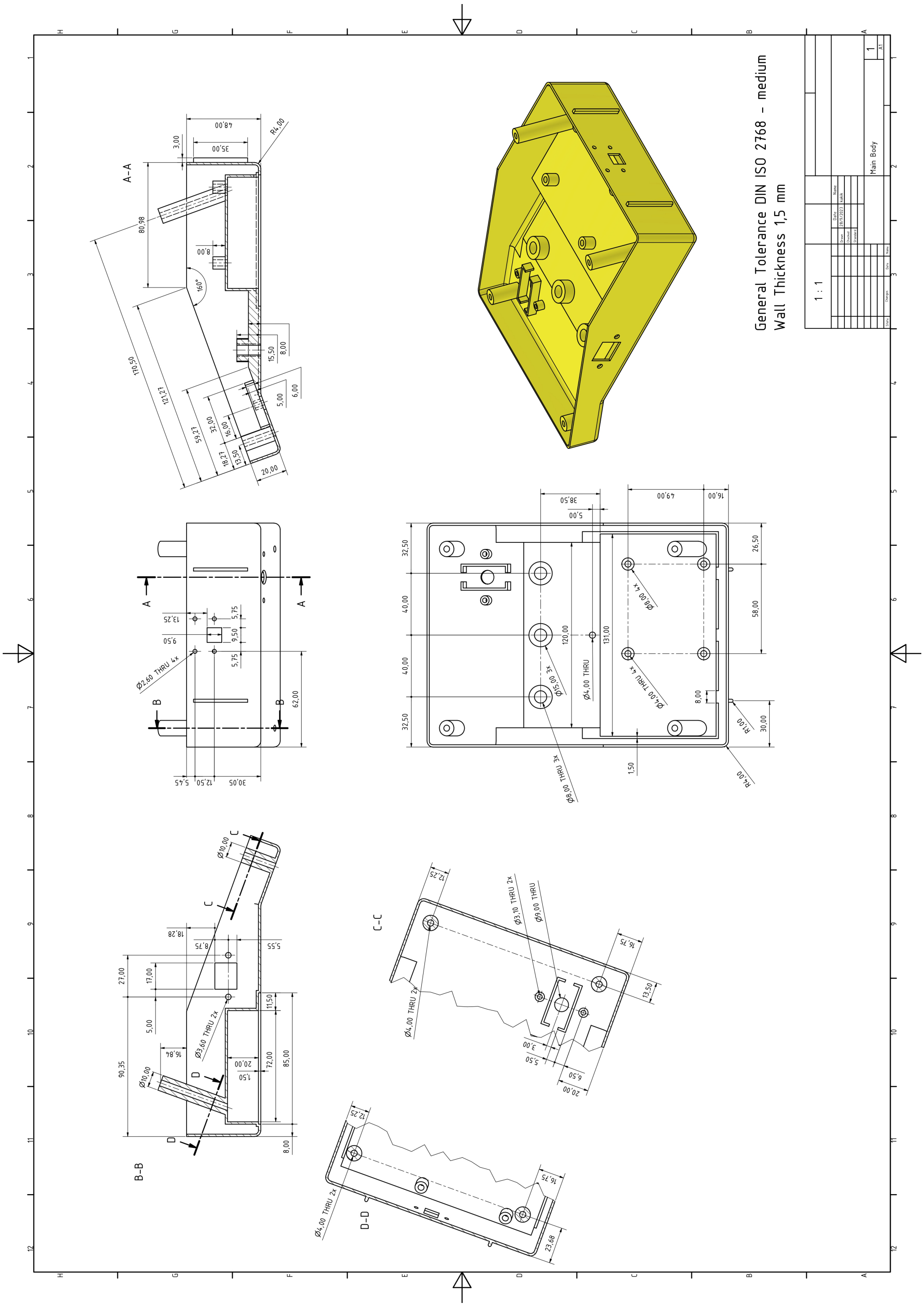


Figure A.3: Main Body Drawing

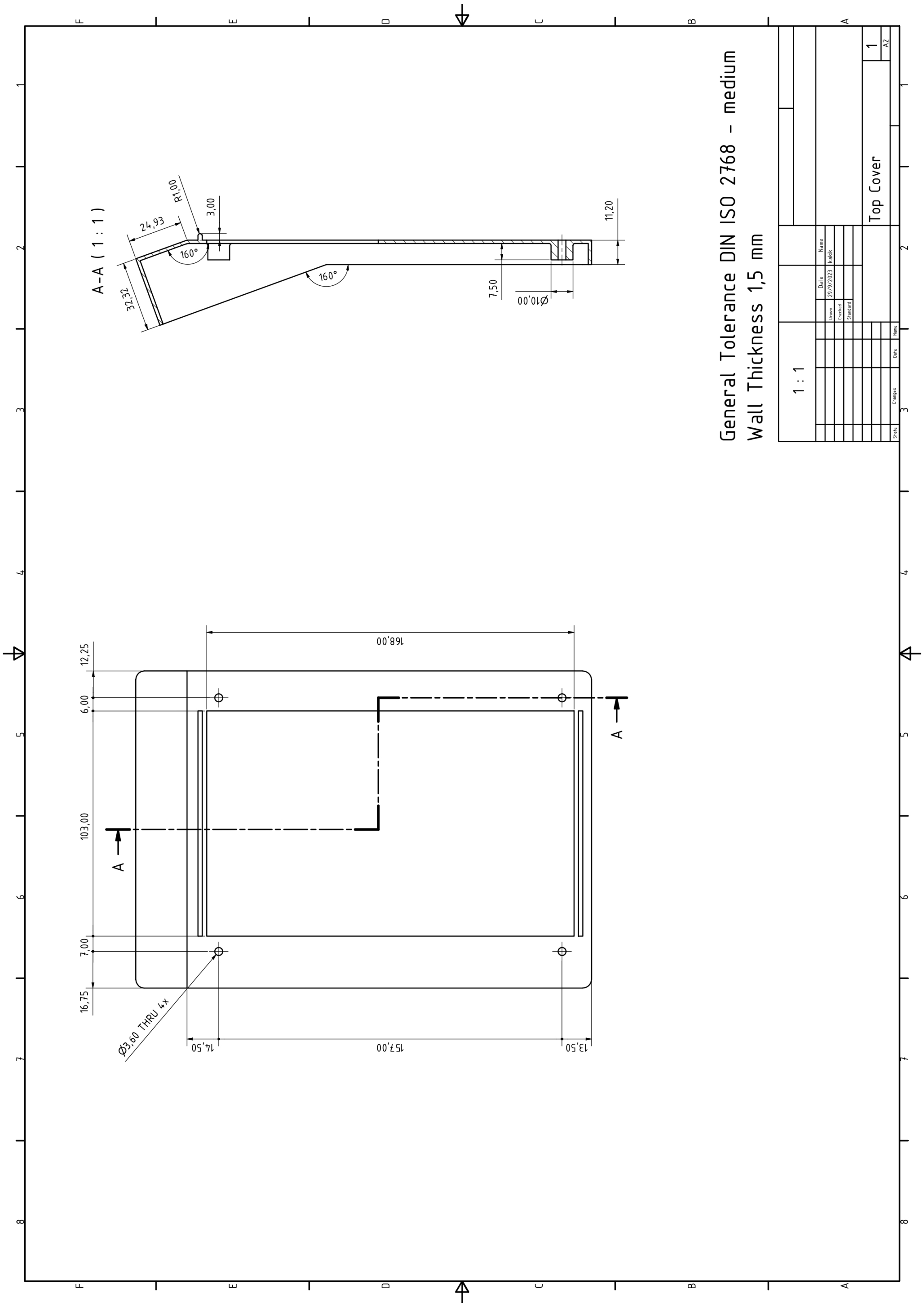


Figure A.4: Top Cover Drawing

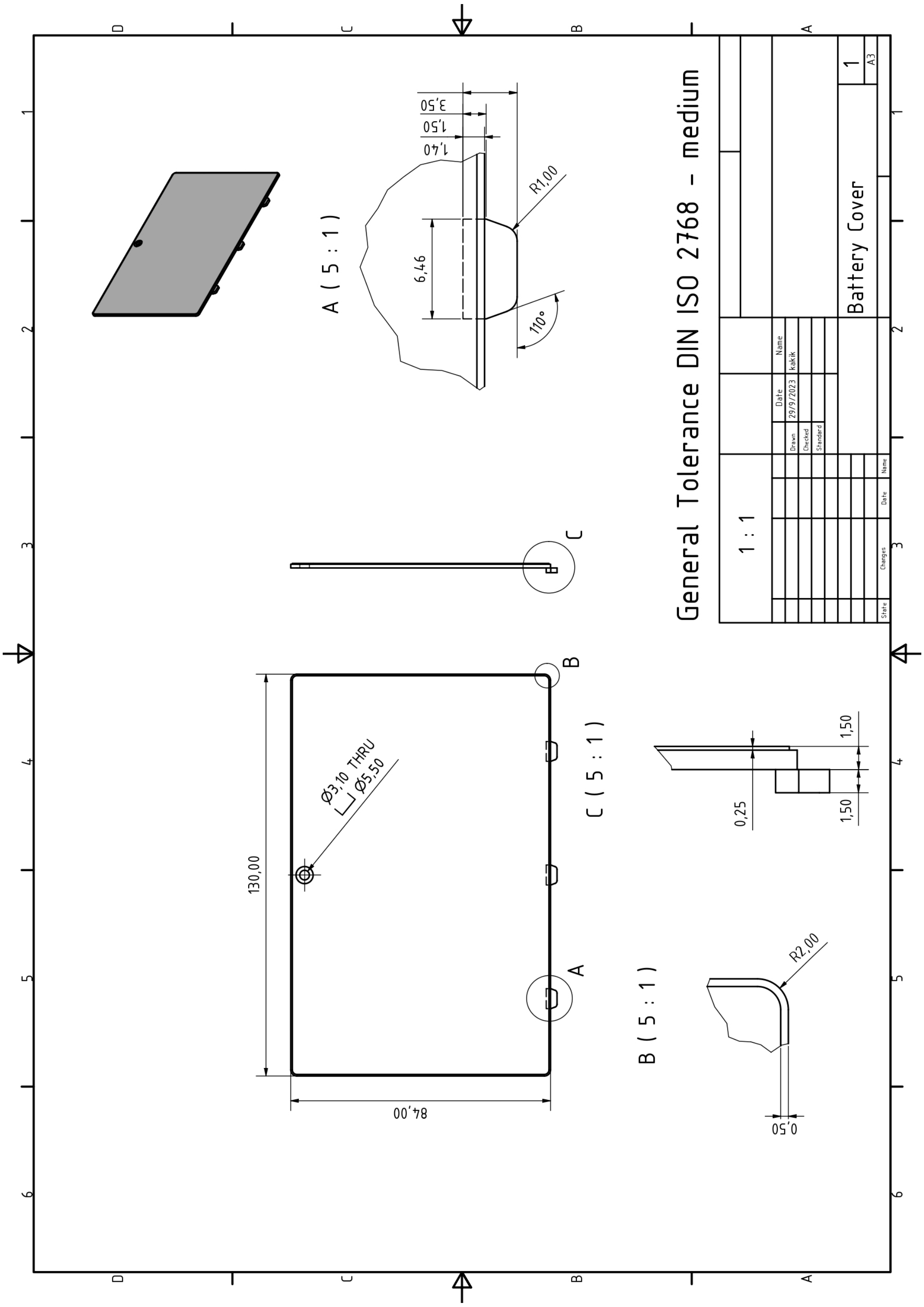


Figure A.5: Battery Cover Drawing

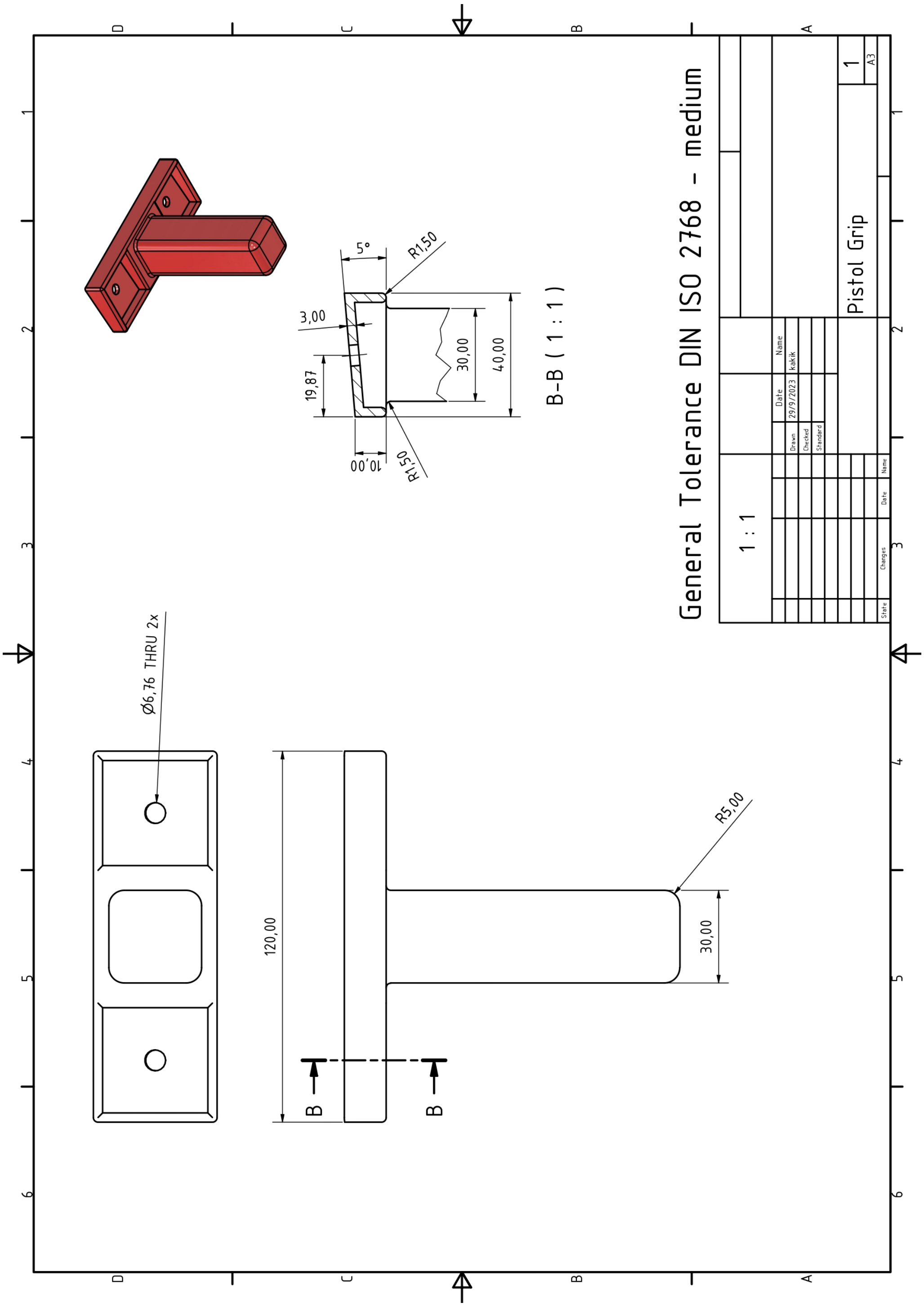


Figure A.6: Pistol Grip Drawing

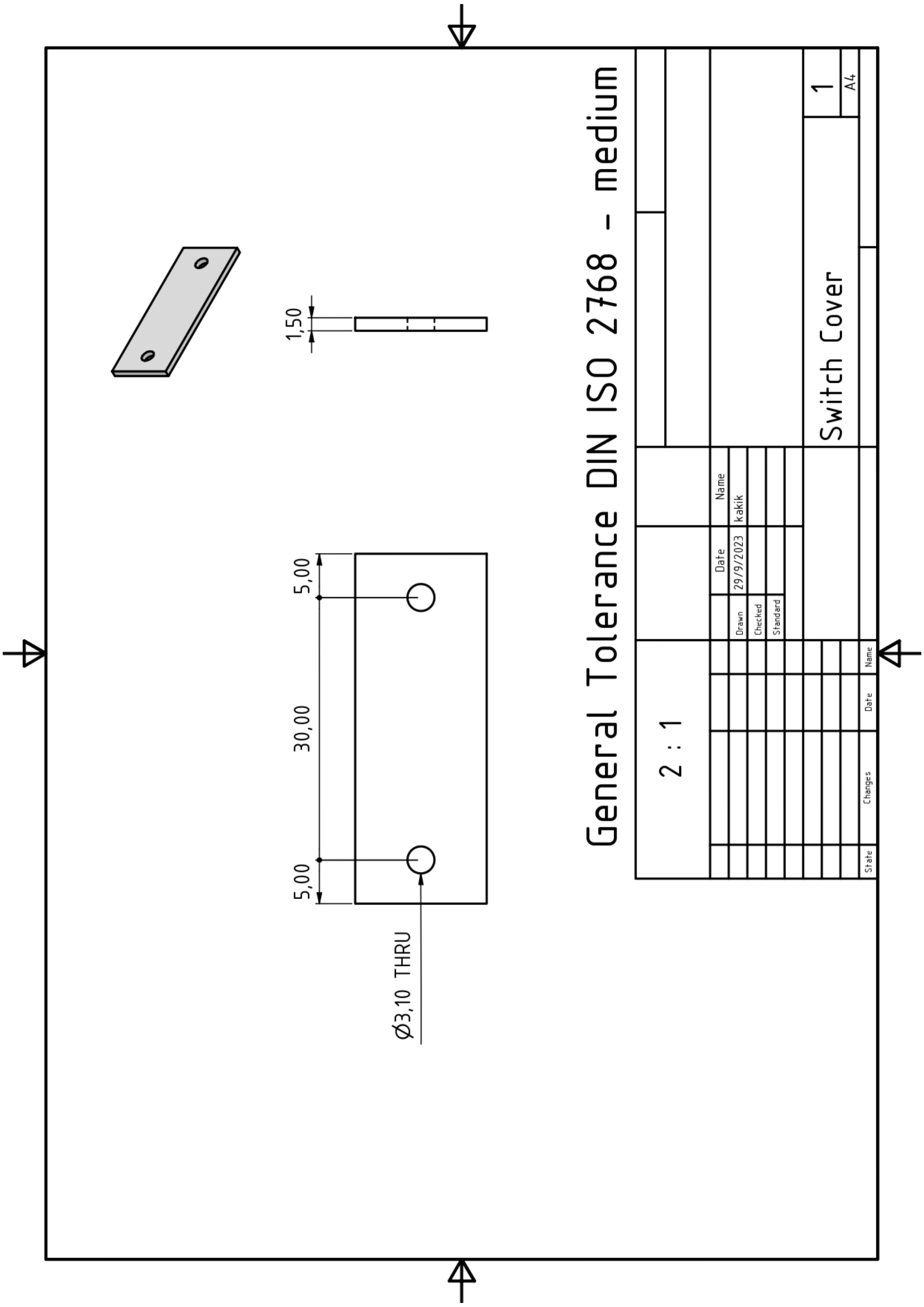


Figure A.7: Switch Cover Drawing

A.3 Technical Specifications

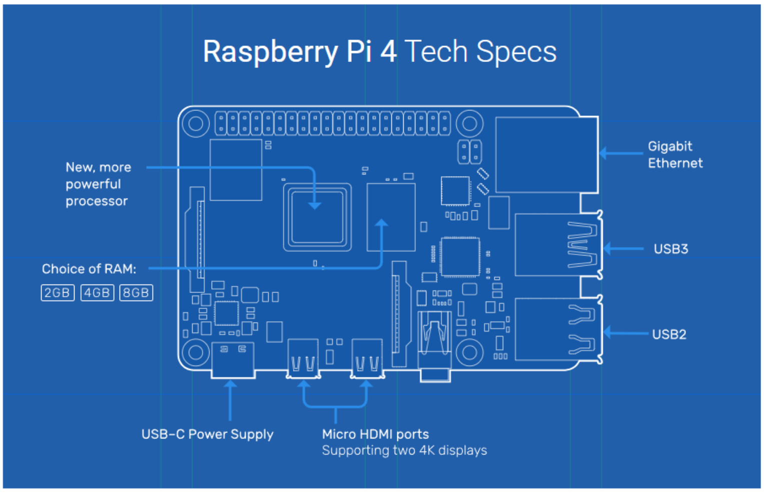
A.3.1 Original Prusa i3 MK3S+ 3D printer

Technical Parameters

Build Volume	25×21×21 cm (9.84"×8.3"×8.3")
Layer height	0.05 - 0.35 mm
Nozzle	0.4mm default, wide range of other diameters/nozzles supported
Filament diameter	1.75 mm
Supported materials	Wide range of thermoplastics, including PLA, PETG, ASA, ABS, PC (Polycarbonate), CPE, PVA/BVOH, PVB, HIPS, PP (Polypropylene), Flex, nGen, Nylon, Carbon filled, Woodfill and other filled materials.
Max travel speed	200+ mm/s
Max nozzle temperature	300 °C / 572 °F
Max heatbed temperature	120 °C / 248 °F
Extruder	Direct Drive, Bondtech gears, E3D V6 hotend
Print surface	Removable magnetic steel sheets(*) with different surface finishes, heatbed with cold corners compensation
Printer dimensions (without spool)	7 kg, 500×550×400 mm; 19.6×21.6×15.7 in (X×Y×Z)
Power consumption	PLA settings: 80W / ABS settings: 120W

Figure A.8: Original Prusa i3 MK3S+ 3D printer

A.3.2 Raspberry Pi 4 Model B



Specifications

Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
1GB, 2GB, 4GB or 8GB LPDDR4-3200 SDRAM (depending on model)
2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
Gigabit Ethernet
2 USB 3.0 ports; 2 USB 2.0 ports.
Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
2 × micro-HDMI® ports (up to 4kp60 supported)
2-lane MIPI DSI display port
2-lane MIPI CSI camera port
4-pole stereo audio and composite video port
H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
OpenGL ES 3.1, Vulkan 1.0
Micro-SD card slot for loading operating system and data storage
5V DC via USB-C connector (minimum 3A*)
5V DC via GPIO header (minimum 3A*)
Power over Ethernet (PoE) enabled (requires separate PoE HAT)
Operating temperature: 0 – 50 degrees C ambient

* A good quality 2.5A power supply can be used if downstream USB peripherals consume less than 500mA in total.

Figure A.9: Raspberry Pi 4 Model B Technical Specifications

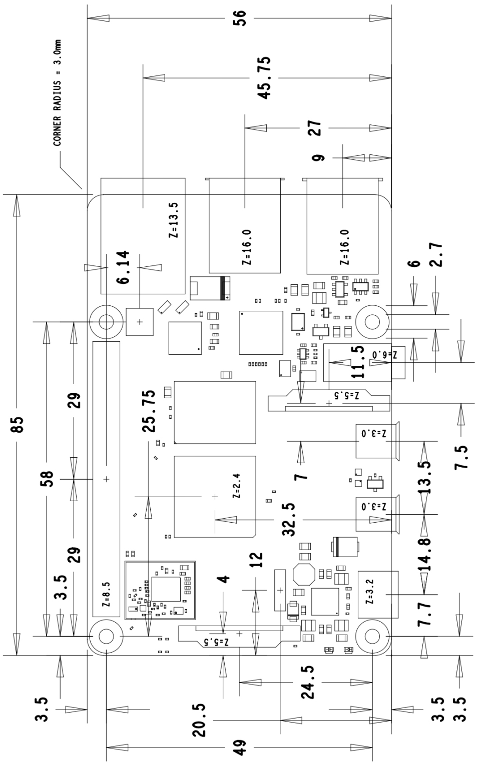


Figure A.10: Raspberry Pi 4 Model B Mechanical Drawing

A.3.3 Raspberry Pi Camera Module V2

	Camera Module v1	Camera Module v2	Camera Module 3	Camera Module 3 Wide	HQ Camera	GS Camera
Net price	$25	$25	$25	$35	$50	$50
Size	Around 25 × 24 × 9 mm	Around 25 × 24 × 9 mm	Around 25 × 24 × 11.5 mm	Around 25 × 24 × 12.4 mm	38 x 38 x 18.4mm (excluding lens)	38 x 38 x 19.8mm (29.5mm with adaptor and dust cap)
Weight	3g	3g	4g	4g	30.4g	34g (41g with adaptor and dust cap)
Still resolution	5 Megapixels	8 Megapixels	11.9 Megapixels	11.9 Megapixels	12.3 Megapixels	1.58 Megapixels
Video modes	1080p30, 720p60 and 640 × 480p60/90	1080p47, 1640 × 1232p41 and 640 × 480p206	2304 × 1296p56, 2304 × 1296p30 HDR, 1536 × 864p120	2304 × 1296p56, 2304 × 1296p30 HDR, 1536 × 864p120	2028 × 1080p50, 2028 × 1520p40 and 1332 × 990p120	1456 x 1088p60
Sensor	OmniVision OV5647	Sony IMX219	Sony IMX708	Sony IMX708	Sony IMX477	Sony IMX296
Sensor resolution	2592 × 1944 pixels	3280 × 2464 pixels	4608 x 2592 pixels	4608 x 2592 pixels	4056 x 3040 pixels	1456 x 1088 pixels
Sensor image area	3.76 × 2.74 mm	3.68 x 2.76 mm (4.6 mm diagonal)	6.45 x 3.63mm (7.4mm diagonal)	6.45 x 3.63mm (7.4mm diagonal)	6.287mm x 4.712 mm (7.9mm diagonal)	6.3mm diagonal
Pixel size	1.4 µm × 1.4 µm	1.12 µm x 1.12 µm	1.4 µm x 1.4 µm	1.4 µm x 1.4 µm	1.55 µm x 1.55 µm	3.45 µm x 3.45 µm
Optical size	1/4"	1/4"	1/2.43"	1/2.43"	1/2.3"	1/2.9"
Focus	Fixed	Adjustable	Motorized	Motorized	Adjustable	Adjustable
Depth of field	Approx 1 m to ∞	Approx 10 cm to ∞	Approx 10 cm to ∞	Approx 5 cm to ∞	N/A	N/A
Focal length	3.60 mm +/- 0.01	3.04 mm	4.74 mm	2.75 mmm	Depends on lens	Depends on lens
Horizontal Field of View (FoV)	53.50 +/- 0.13 degrees	62.2 degrees	66 degrees	102 degrees	Depends on lens	Depends on lens
Vertical Field of View (FoV)	41.41 +/- 0.11 degrees	48.8 degrees	41 degrees	67 degrees	Depends on lens	Depends on lens
Focal ratio (F-Stop)	F2.9	F2.0	F1.8	F2.2	Depends on lens	Depends on lens
Maximum exposure times (seconds)	6 (legacy) / 0.97 (libcamera)	11.76	112	112	670.74	15.5
Lens Mount	N/A	N/A	N/A	N/A	C/CS- or M12-mount	C/CS
NoIR version available?	Yes	Yes	Yes	Yes	No	No

Figure A.11: Raspberry Pi Camera Module V2 Technical Specifications

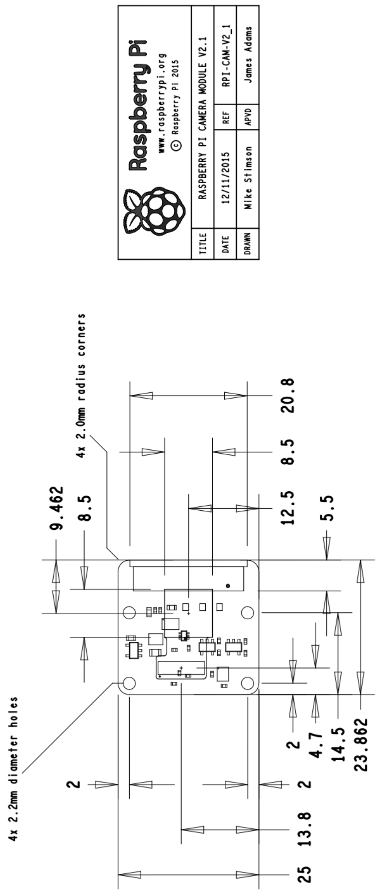


Figure A.12: Raspberry Pi Camera Module V2 Mechanical Drawing

A.3.4 Waveshare 7inch HDMI LCD (H)

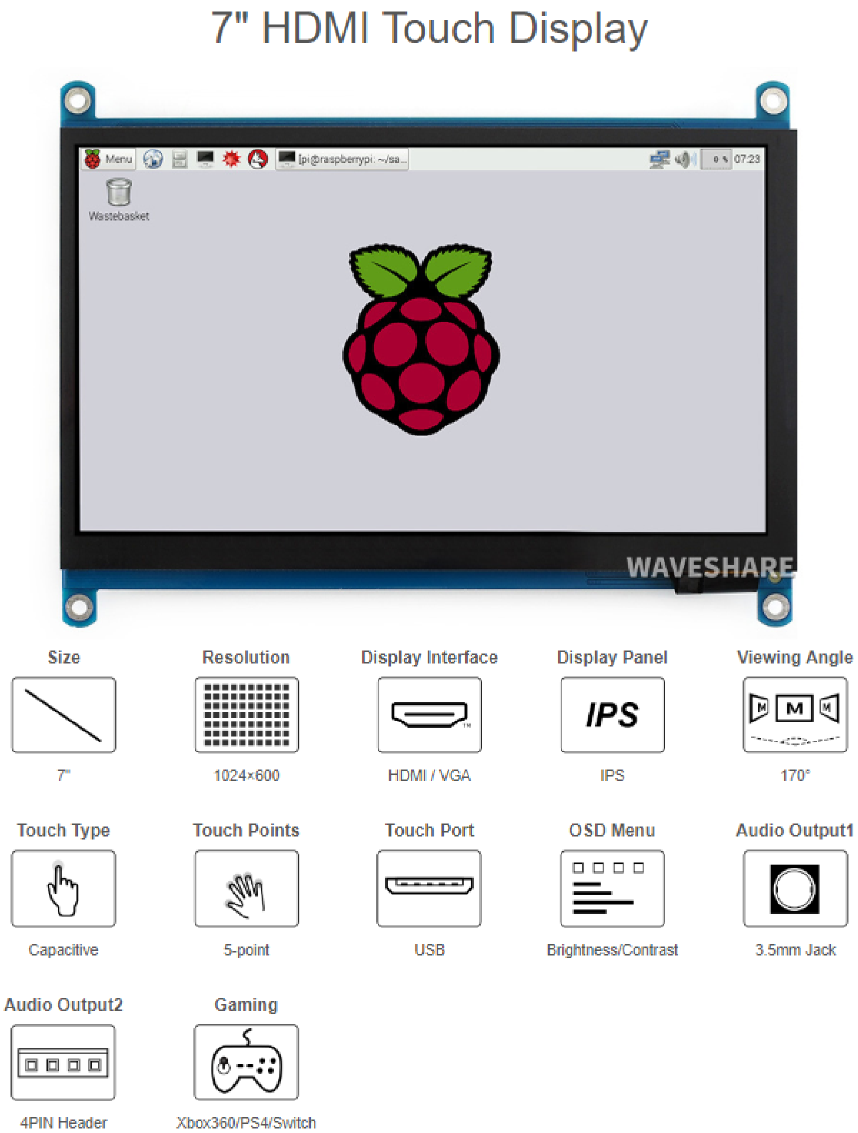


Figure A.13: Waveshare 7inch HDMI LCD (H) Technical Specifications -1

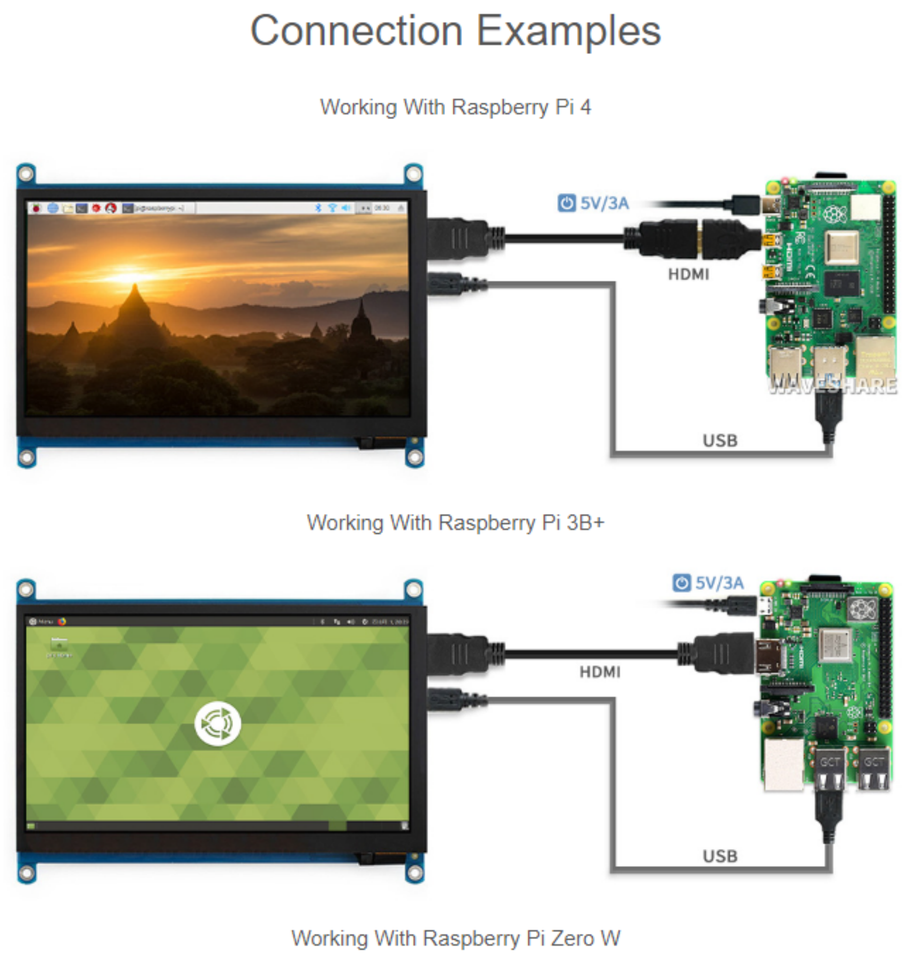


Figure A.14: Waveshare 7inch HDMI LCD (H) Technical Specifications -2

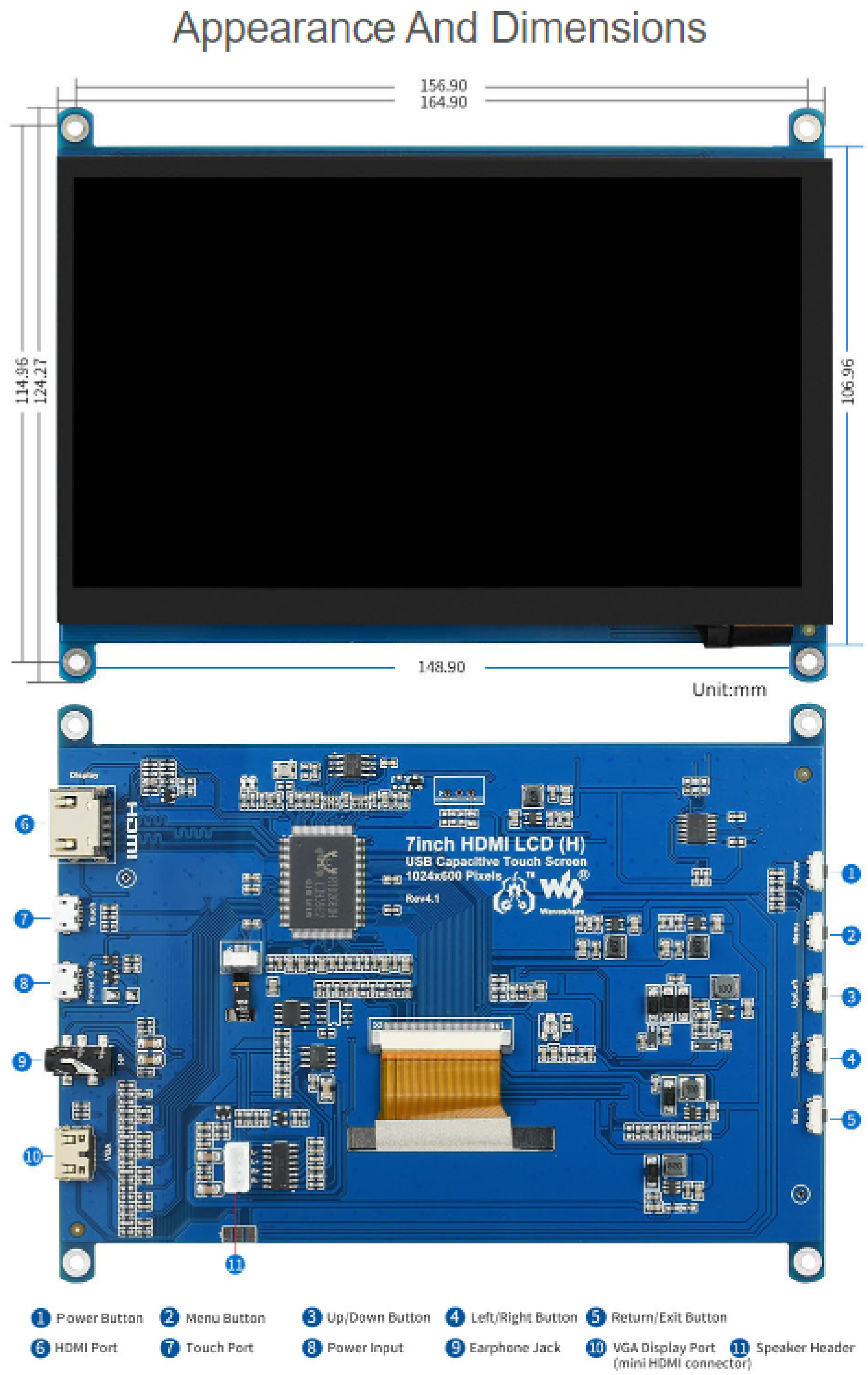


Figure A.15: Waveshare 7inch HDMI LCD (H) Technical Specifications -3

A.3.5 Veektomx VT103

Figure A.16: Veektomx VT103 Technical Specifications

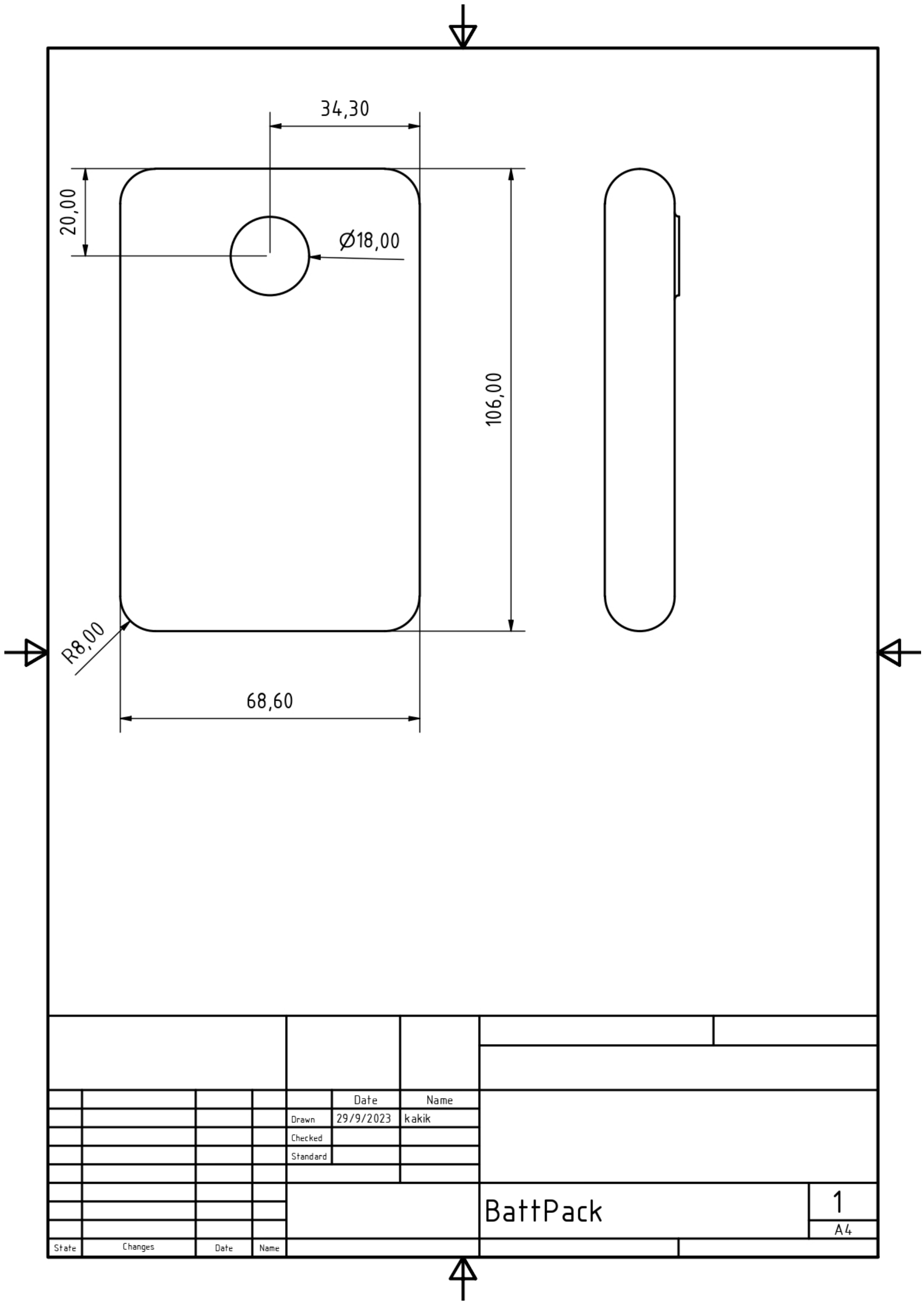


Figure A.17: Veektomx VT103 Mechanical Drawing

A.3.6 Ruthex Brass Inserts

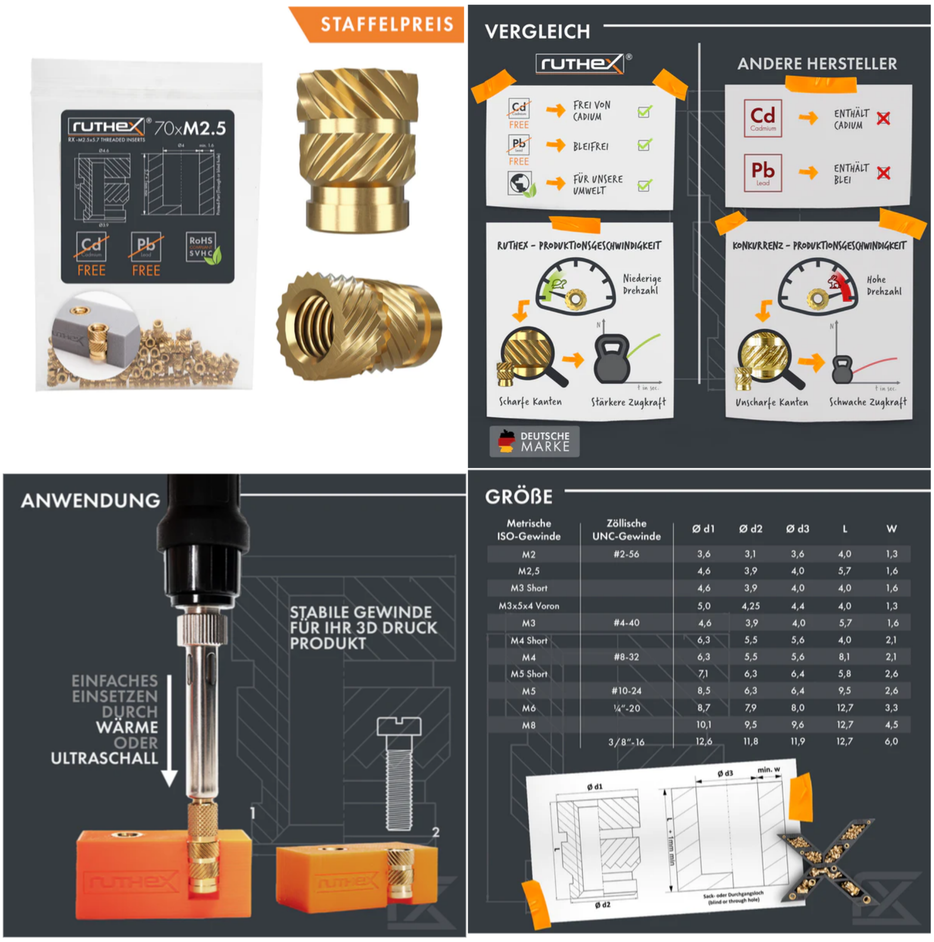
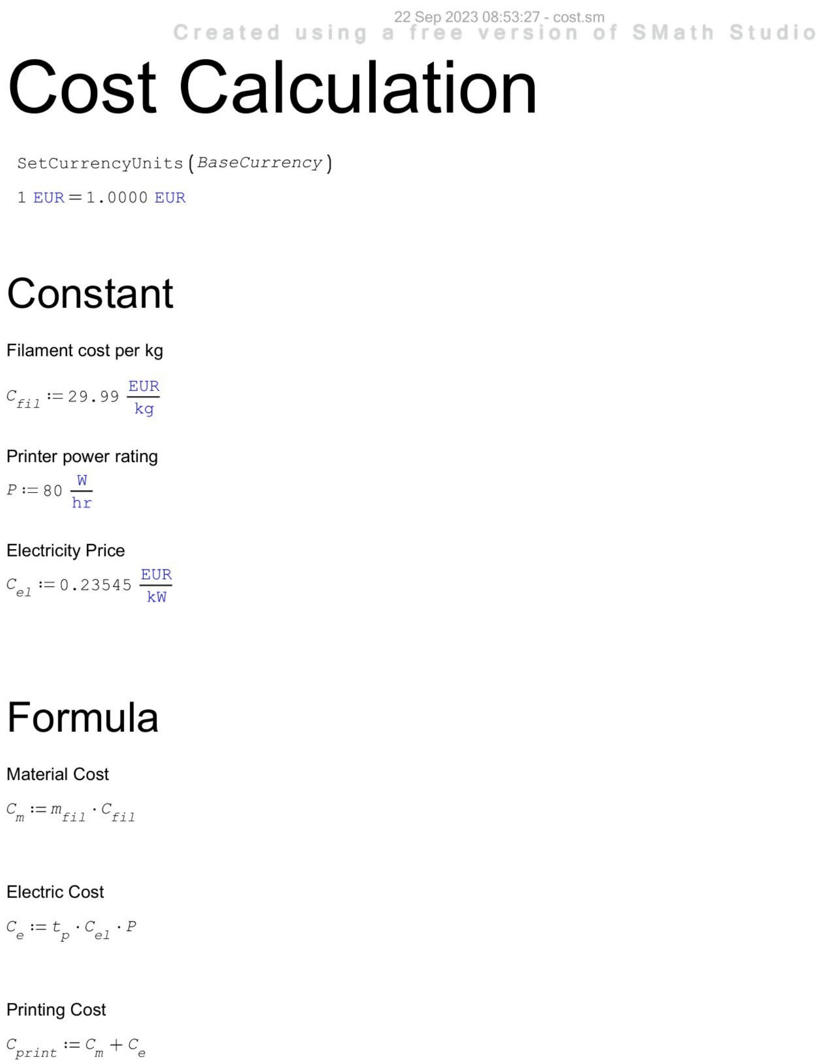


Figure A.18: Ruthex Brass Inserts

A.4 Cost Calculation


22 Sep 2023 08:53:27 - cost.sm

Cost Calculation

SetCurrencyUnits(*BaseCurrency*)

$1\ \mathrm{EUR} = 1.0000\ \mathrm{EUR}$

Constant

Filament cost per kg

$C_{fil} := 29.99\ \frac{\mathrm{EUR}}{\mathrm{kg}}$

Printer power rating

$P := 80\ \frac{\mathrm{W}}{\mathrm{hr}}$

Electricity Price

$C_{el} := 0.23545\ \frac{\mathrm{EUR}}{\mathrm{kW}}$

Formula

Material Cost

$C_m := m_{fil} \cdot C_{fil}$

Electric Cost

$C_e := t_p \cdot C_{el} \cdot P$

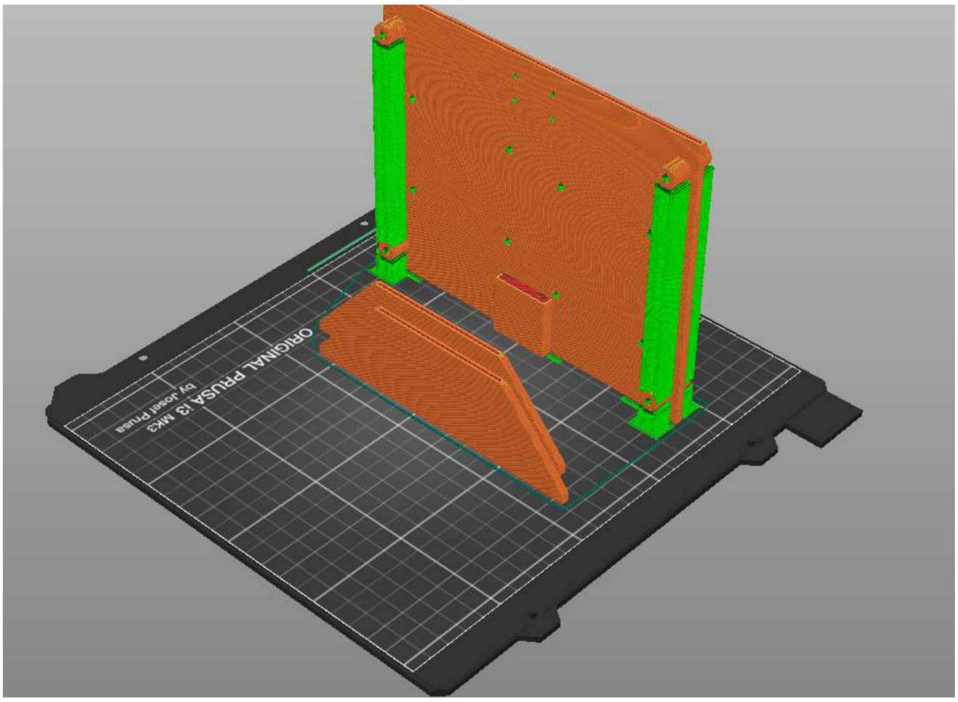
Printing Cost

$C_{print} := C_m + C_e$

Figure A.19: Cost Calculation 1

Variant 2

Base



$m_{fil} := 175.07\ \mathrm{g}$

$t_p := 6\ \mathrm{hr} + 8\ \mathrm{min}$

$C_m = 5.2503\ \mathrm{EUR}$

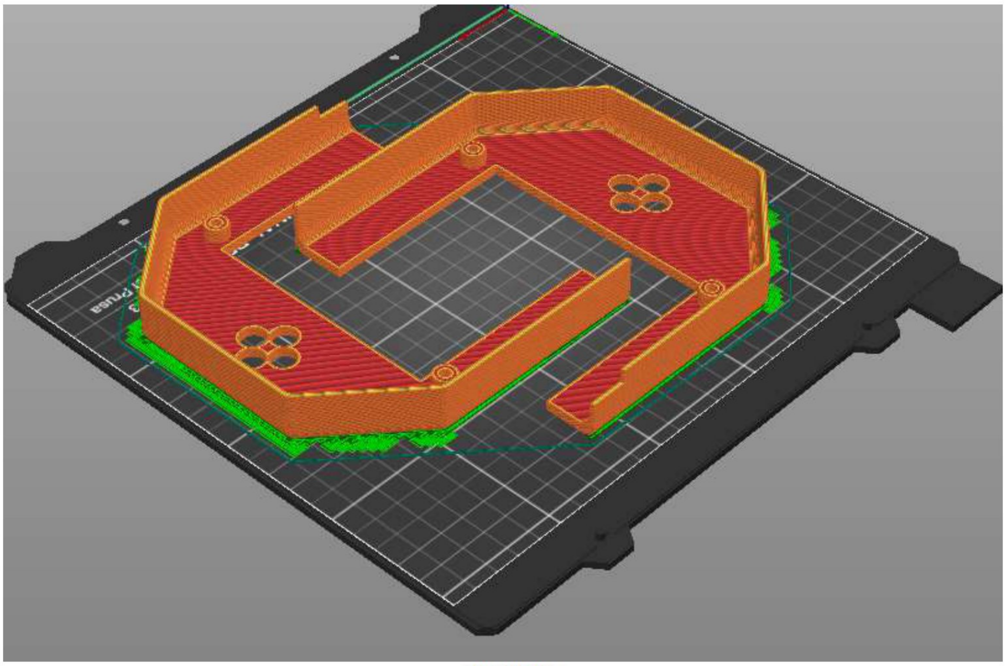
$C_e = 0.1155\ \mathrm{EUR}$

$C_{print} = 5.3659\ \mathrm{EUR}$

Sliced Info	
Used Filament (g)	175.07
Used Filament (m)	58.70
Used Filament (mm³)	141185.08
Cost	4.45
Estimated printing time:	
- normal mode	6h8m
- stealth mode	6h17m

Figure A.20: Cost Calculation 2

Top Cover



Sliced Info

Used Filament (g)	105.85
Used Filament (m)	35.49
Used Filament (mm³)	85360.45
Cost	2.69
Estimated printing time:	
- normal mode	3h19m
- stealth mode	3h22m

$m_{fil} := 105.85\ \mathrm{g}$

$t_p := 3\ \mathrm{hr} + 19\ \mathrm{min}$

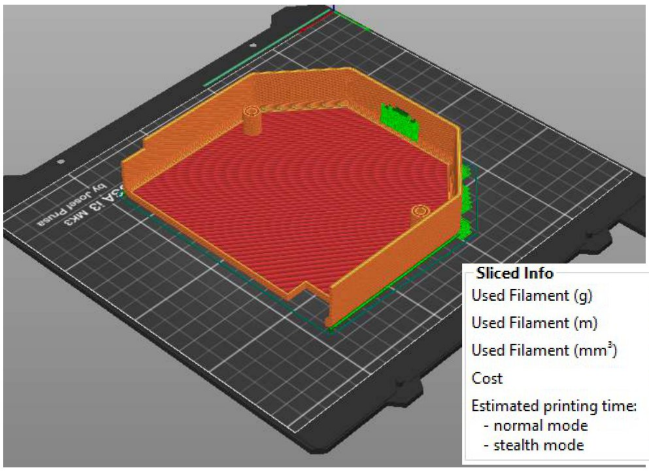
$C_m = 3.1744\ \mathrm{EUR}$

$C_e = 0.0625\ \mathrm{EUR}$

$C_{print} = 3.2369\ \mathrm{EUR}$

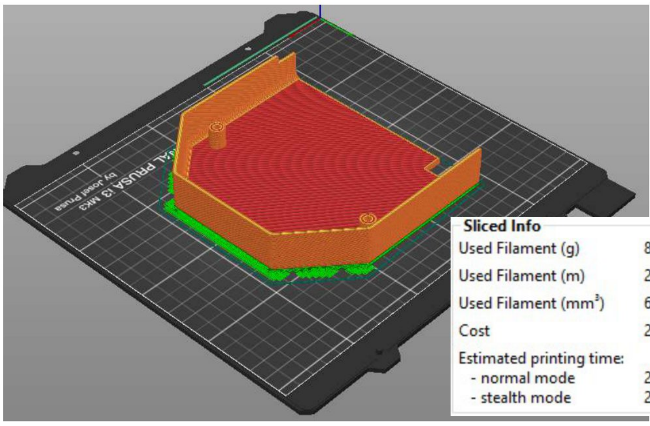
Figure A.21: Cost Calculation 3

Back Cover



Sliced Info

Used Filament (g)	86.73
Used Filament (m)	29.08
Used Filament (mm³)	69943.39
Cost	2.20
Estimated printing time:	
- normal mode	2h34m
- stealth mode	2h36m



Sliced Info

Used Filament (g)	86.60
Used Filament (m)	29.04
Used Filament (mm³)	69841.82
Cost	2.20
Estimated printing time:	
- normal mode	2h32m
- stealth mode	2h33m

$m_{fil} := (86.73 + 86.6)\ \text{g}$

$t_p := 2\ \text{hr} + 34\ \text{min} + 2\ \text{hr} + 32\ \text{min}$

$C_m = 5.1982\ \text{EUR}$

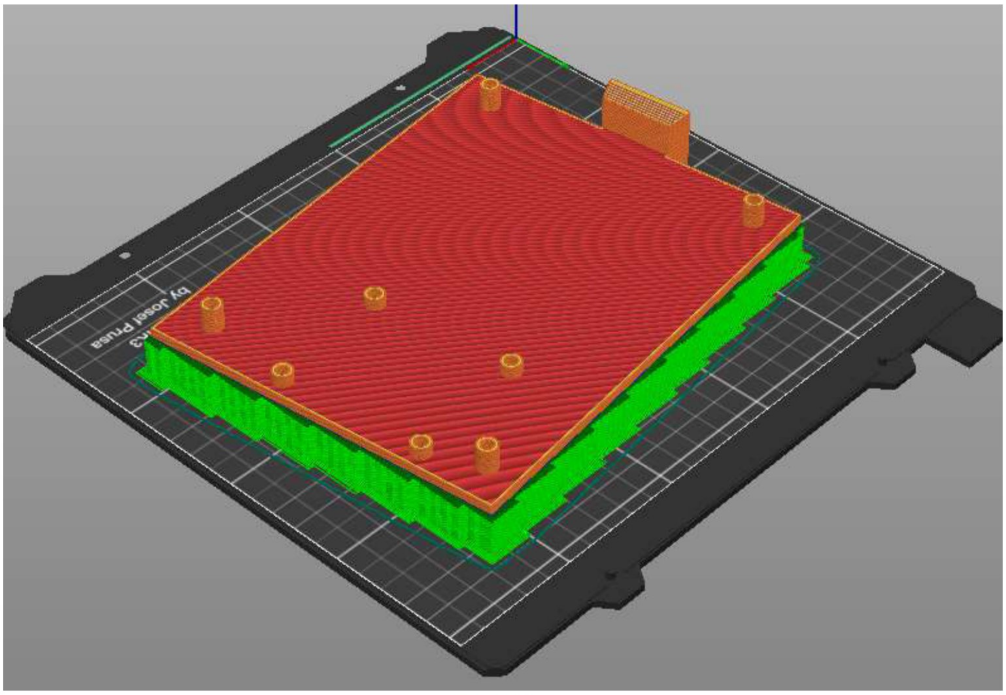
$C_e = 0.0961\ \text{EUR}$

$C_{print} = 5.2942\ \text{EUR}$

Figure A.22: Cost Calculation 4

Variant 3

Base



$m_{fil} := 222.96\ \mathrm{g}$

$t_p := 6\ \mathrm{hr} + 40\ \mathrm{min}$

$C_m = 6.6866\ \mathrm{EUR}$

$C_e = 0.1256\ \mathrm{EUR}$

$C_{print} = 6.8121\ \mathrm{EUR}$

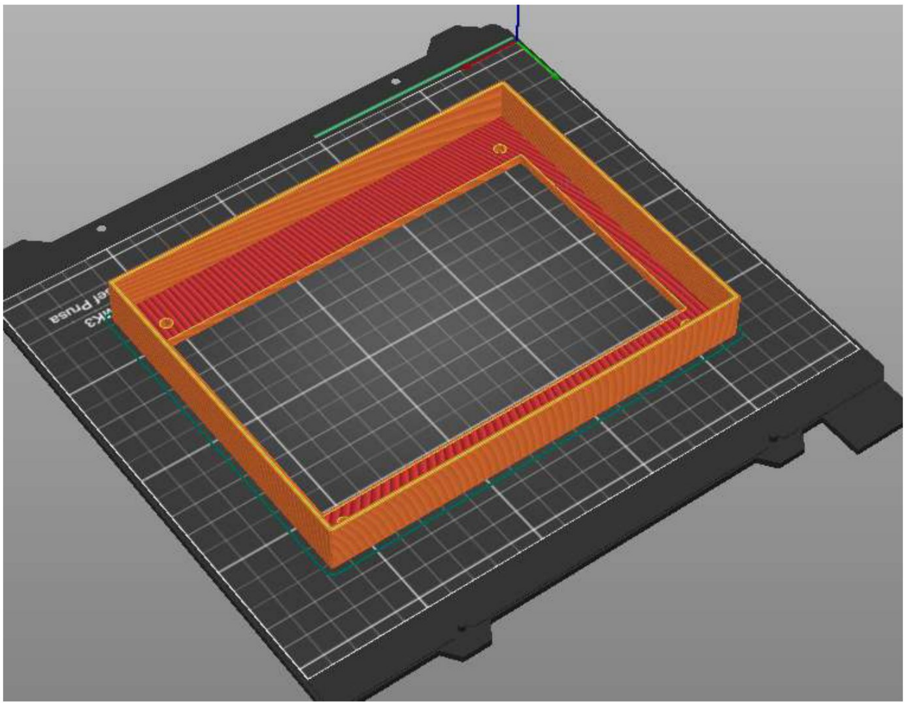
Sliced Info

Used Filament (g)	222.96
Used Filament (m)	74.76
Used Filament (mm³)	179809.70
Cost	5.66
Estimated printing time:	
- normal mode	6h40m
- stealth mode	6h44m

5 / 19

Figure A.23: Cost Calculation 5

Top Cover



$m_{fil} := 71.02\ \mathrm{g}$

$t_p := 2\ \mathrm{hr} + 5\ \mathrm{min}$

$C_m = 2.1299\ \mathrm{EUR}$

$C_e = 0.0392\ \mathrm{EUR}$

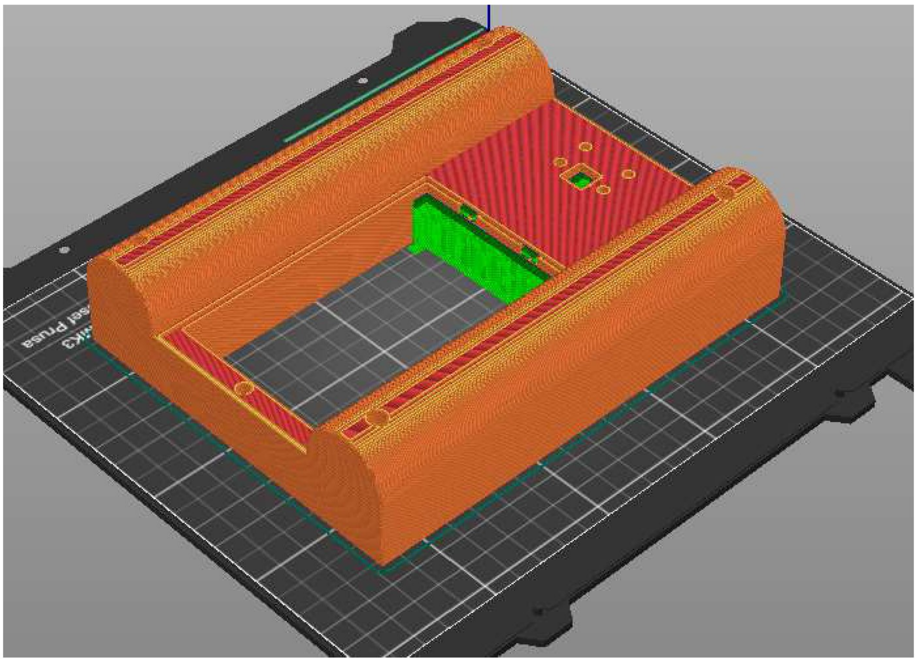
$C_{print} = 2.1691\ \mathrm{EUR}$

Sliced Info

Used Filament (g)	71.02
Used Filament (m)	23.81
Used Filament (mm³)	57270.69
Cost	1.80
Estimated printing time:	
- normal mode	2h5m
- stealth mode	2h6m

Figure A.24: Cost Calculation 6

Back Cover



$m_{fil} := 303.02\ \mathrm{g}$

$t_p := 9\ \mathrm{hr} + 22\ \mathrm{min}$

$C_m = 9.0876\ \mathrm{EUR}$

$C_e = 0.1764\ \mathrm{EUR}$

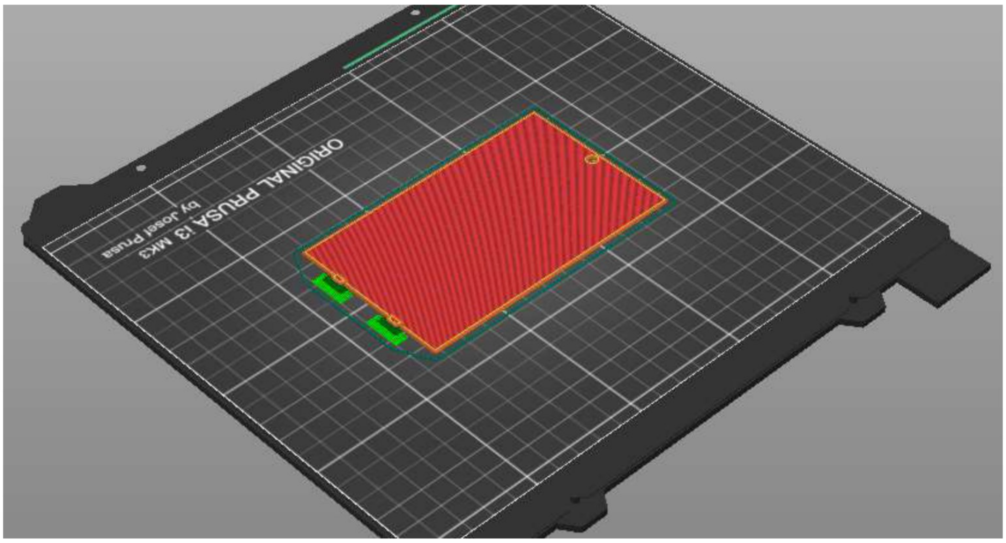
$C_{print} = 9.264\ \mathrm{EUR}$

Sliced Info

Used Filament (g)	303.02
Used Filament (m)	101.60
Used Filament (mm³)	244371.20
Cost	7.70
Estimated printing time:	
- normal mode	9h22m
- stealth mode	9h31m

Figure A.25: Cost Calculation 7

Battery Cover



$m_{fil} := 17.43\ g$

$t_p := 0\ hr + 32\ min$

$C_m = 0.5227\ EUR$

$C_e = 0.01\ EUR$

$C_{print} = 0.5328\ EUR$

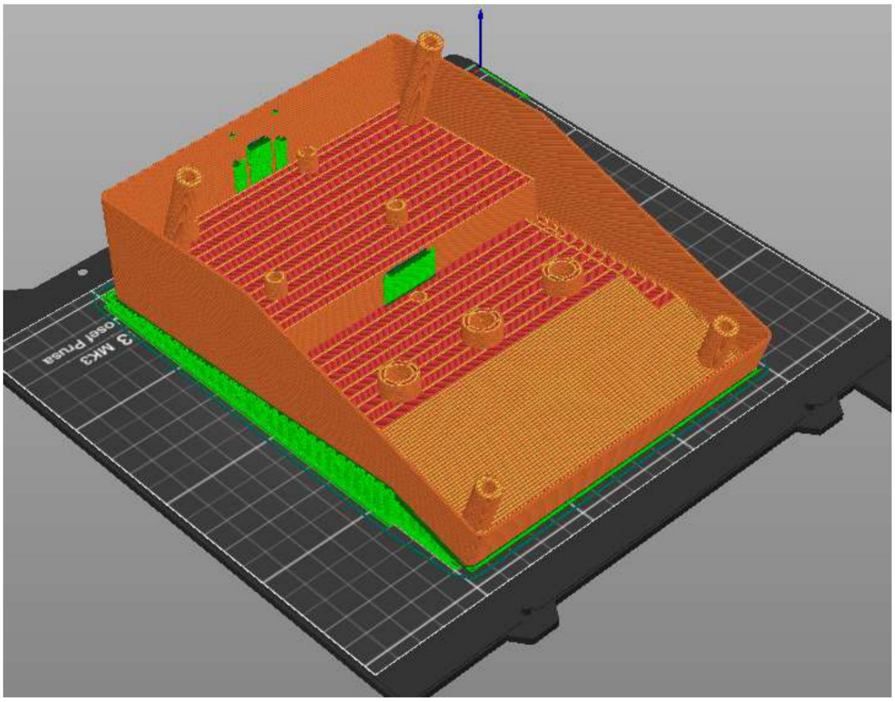
Sliced Info	
Used Filament (g)	17.43
Used Filament (m)	5.84
Used Filament (mm³)	14056.38
Cost	0.44
Estimated printing time:	
- normal mode	32m
- stealth mode	32m

8 / 19

Figure A.26: Cost Calculation 8

Variant 6

Main Body



$m_{fil} := 240.3\ g$

$t_p := 8\ hr + 53\ min$

$C_m = 7.2066\ EUR$

$C_e = 0.1673\ EUR$

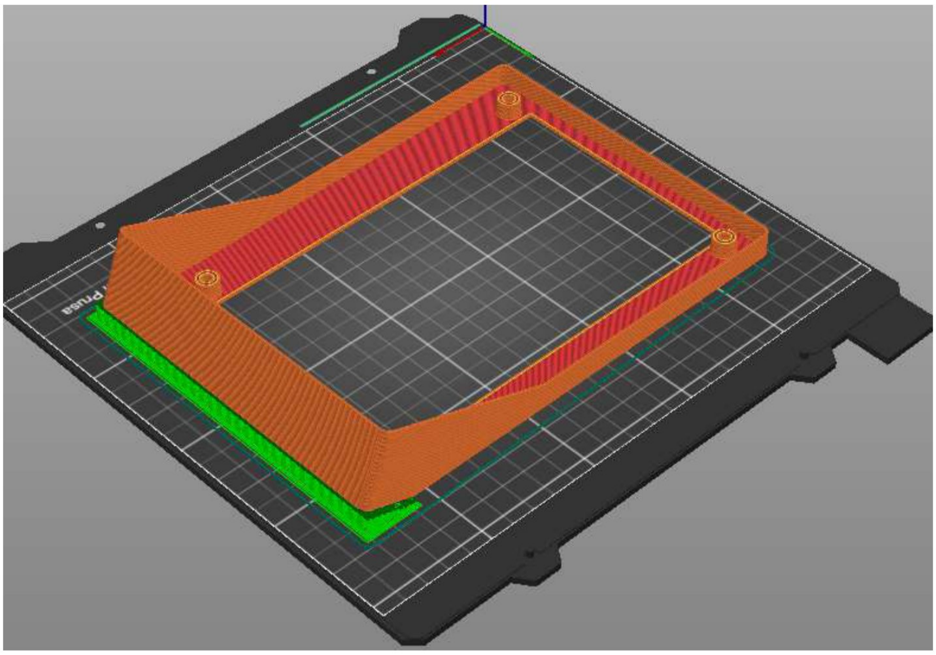
$C_{print} = 7.3739\ EUR$

Sliced Info	
Used Filament (g)	240.30
Used Filament (m)	80.57
Used Filament (mm³)	193788.10
Cost	6.10
Estimated printing time:	
- normal mode	8h53m
- stealth mode	9h3m

9 / 19

Figure A.27: Cost Calculation 9

Top Cover



$m_{fil} := 55.32$ g

$t_p := 1$ hr $+ 52$ min

$C_m = 1.659$ EUR

$C_e = 0.0352$ EUR

$C_{print} = 1.6942$ EUR

Sliced Info

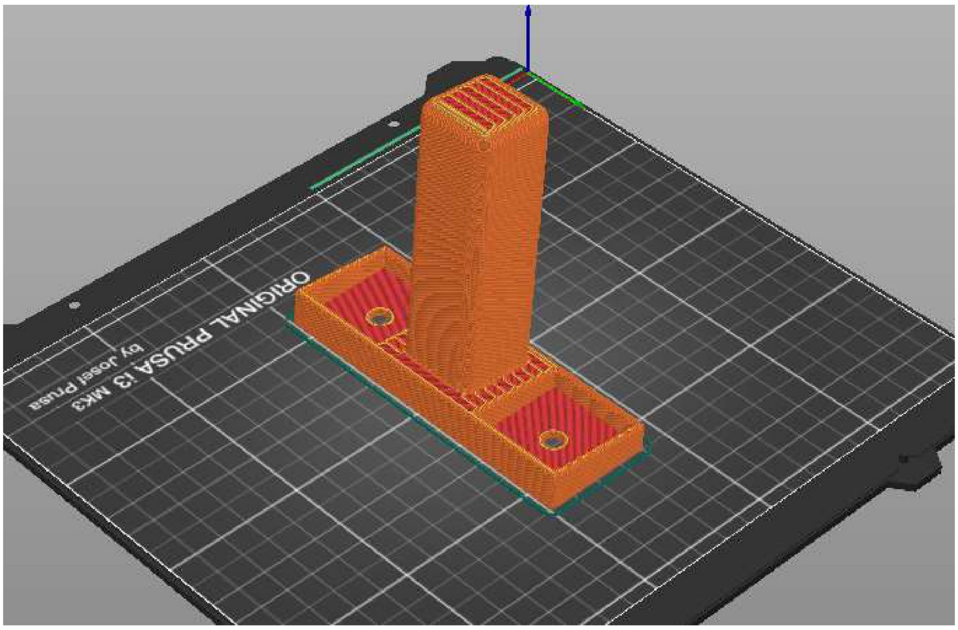
Used Filament (g)	55.35
Used Filament (m)	18.56
Used Filament (mm³)	44638.88
Cost	1.41
Estimated printing time:	
- normal mode	1h52m
- stealth mode	1h54m

10 / 19

Figure A.28: Cost Calculation 10

22 Sep 2023 08:53:27 - cost.sm

Pistol Handle



$m_{fil} := 57.30\ \mathrm{g}$

$t_p := 1\ \mathrm{hr} + 44\ \mathrm{min}$

$C_m = 1.7184\ \mathrm{EUR}$

$C_e = 0.0326\ \mathrm{EUR}$

$C_{print} = 1.7511\ \mathrm{EUR}$

Sliced Info

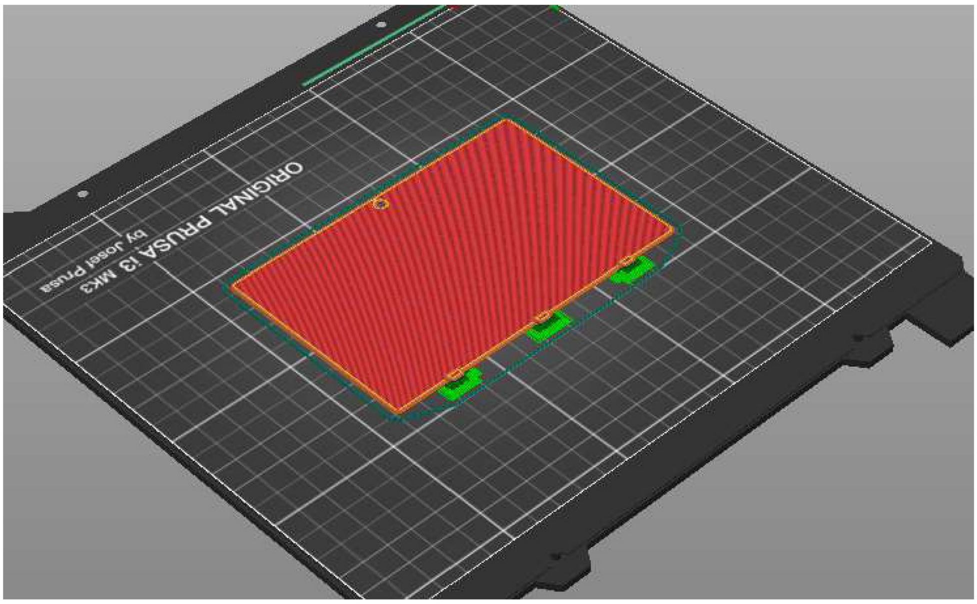
Used Filament (g)	57.30
Used Filament (m)	19.21
Used Filament (mm³)	46212.83
Cost	1.46
Estimated printing time:	
- normal mode	1h44m
- stealth mode	1h45m

11 / 19

Figure A.29: Cost Calculation 11

22 Sep 2023 08:53:27 - cost.sm

Battery Cover



Sliced Info	
Used Filament (g)	22.21
Used Filament (m)	7.45
Used Filament (mm³)	17910.60
Cost	0.56
Estimated printing time:	
- normal mode	40m
- stealth mode	40m

$m_{fil} := 22.21\ \mathrm{g}$

$t_p := 0\ \mathrm{hr} + 40\ \mathrm{min}$

$C_m = 0.6661\ \mathrm{EUR}$

$C_e = 0.0126\ \mathrm{EUR}$

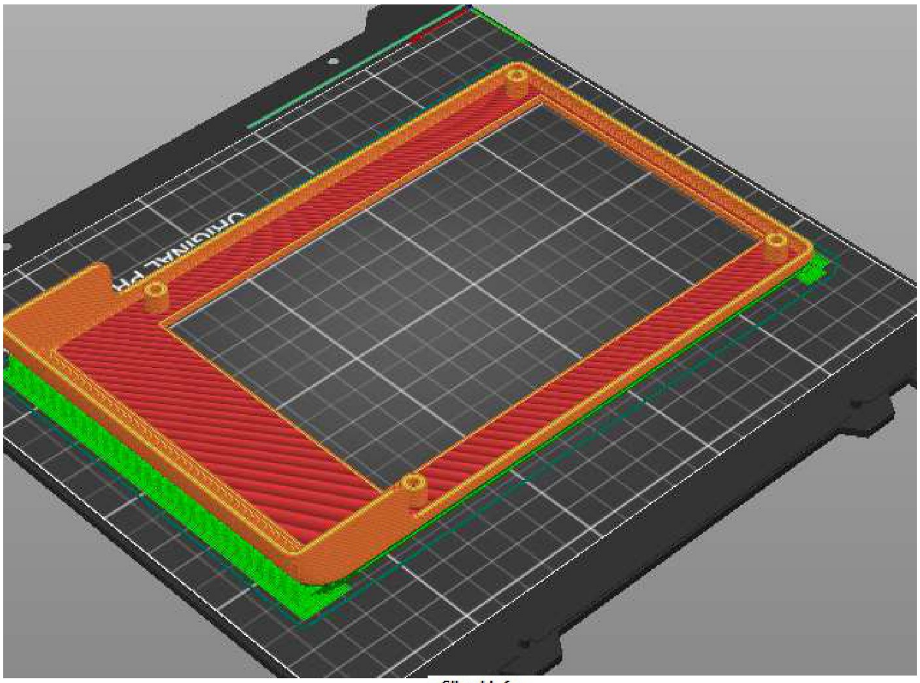
$C_{print} = 0.6786\ \mathrm{EUR}$

12 / 19

Figure A.30: Cost Calculation 12

Variant 7

Top Cover



$m_{fil} := 61.5\ \mathrm{g}$

$t_p := 2\ \mathrm{hr} + 5\ \mathrm{min}$

Sliced Info

Used Filament (g)	61.50
Used Filament (m)	20.62
Used Filament (mm³)	49593.24
Cost	1.56
Estimated printing time:	
- normal mode	2h5m
- stealth mode	2h7m

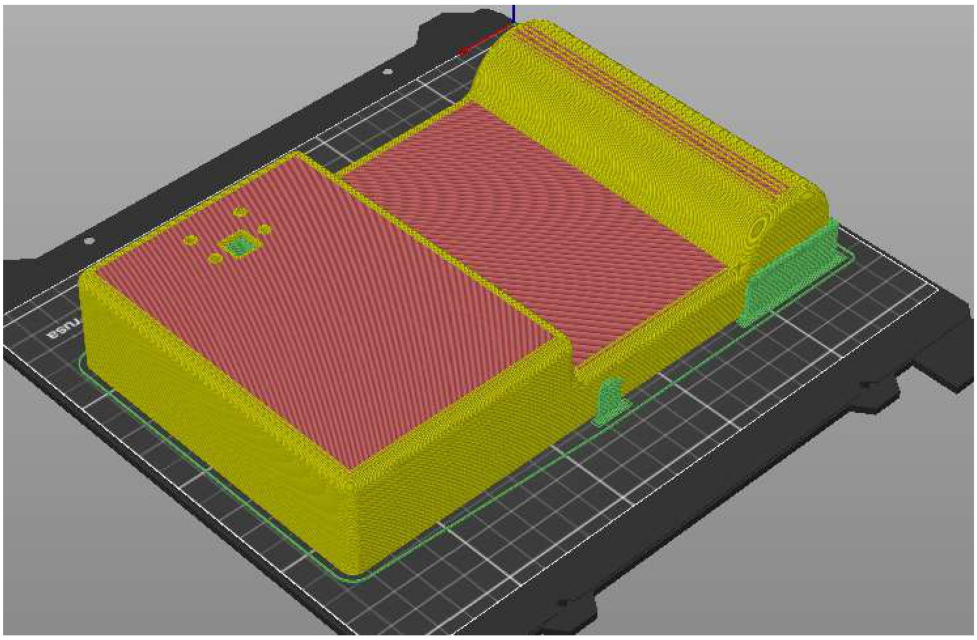
$C_m = 1.8444\ \mathrm{EUR}$

$C_e = 0.0392\ \mathrm{EUR}$

$C_{print} = 1.8836\ \mathrm{EUR}$

Figure A.31: Cost Calculation 13

Back Cover



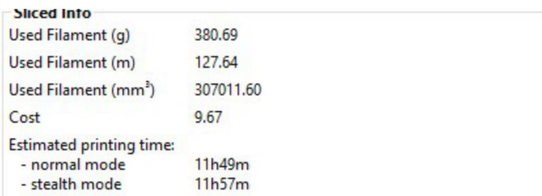
$m_{fil} := 380.69$ g

$t_p := 11$ hr $+ 49$ min

$C_m = 11.4169$ EUR

$C_e = 0.2226$ EUR

$C_{print} = 11.6395$ EUR



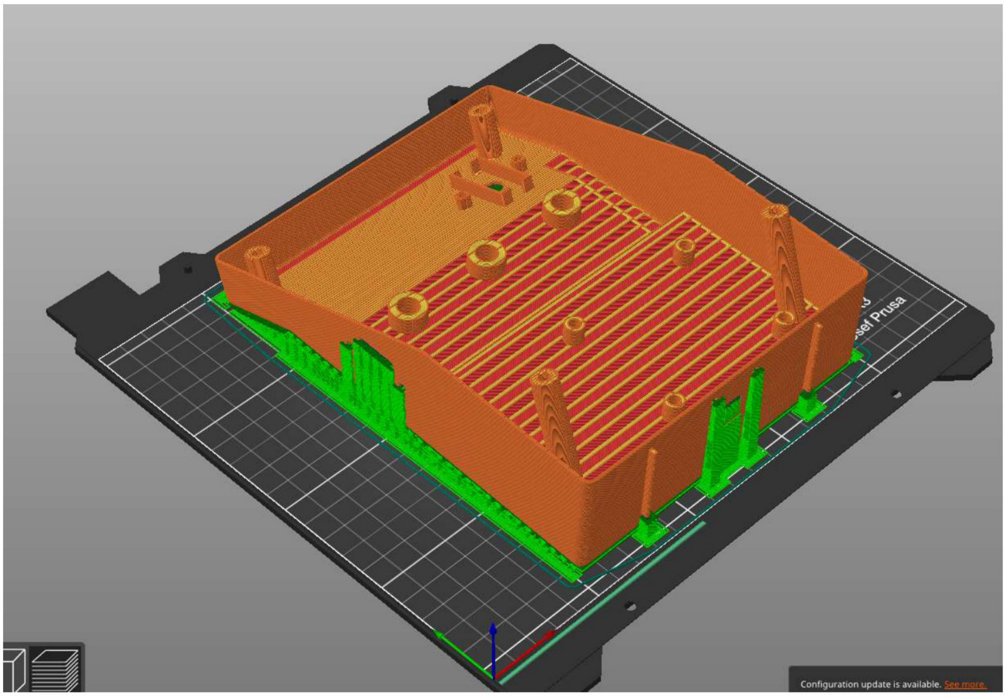
Sliced Info

Used Filament (g)	380.69
Used Filament (m)	127.64
Used Filament (mm³)	307011.60
Cost	9.67
Estimated printing time:	
- normal mode	11h49m
- stealth mode	11h57m

Figure A.32: Cost Calculation 14

Variant 6 Final

Main Body



$m_{fil} := 245.14\ g$

$t_p := 9\ hr + 15\ min$

$C_m = 7.3517\ EUR$

$C_e = 0.1742\ EUR$

$C_{print} = 7.526\ EUR$

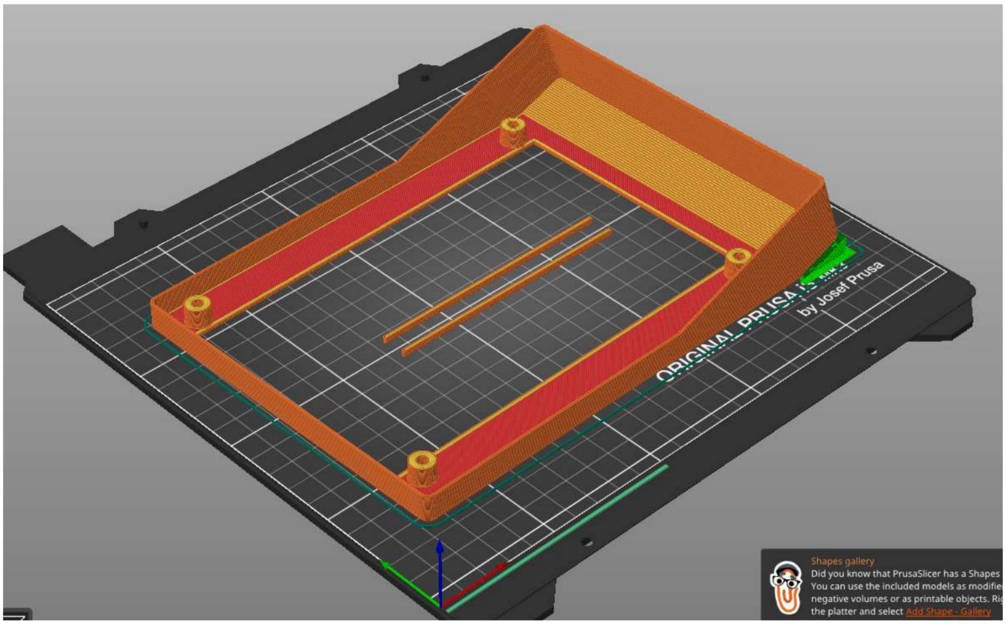
Sliced Info

Used Filament (g)	245.14
Used Filament (m)	82.19
Used Filament (mm³)	197695.71
Cost	6.23
Estimated printing time:	
- normal mode	9h15m
- stealth mode	9h25m

15 / 19

Figure A.33: Cost Calculation 15

Top Cover



Sliced Info	
Used Filament (g)	57.71
Used Filament (m)	19.35
Used Filament (mm³)	46541.58
Cost	1.47
Estimated printing time:	
- normal mode	2h0m
- stealth mode	2h2m

$m_{fil} := 57.71\ \mathrm{g}$

$t_p := 2\ \mathrm{hr} + 0\ \mathrm{min}$

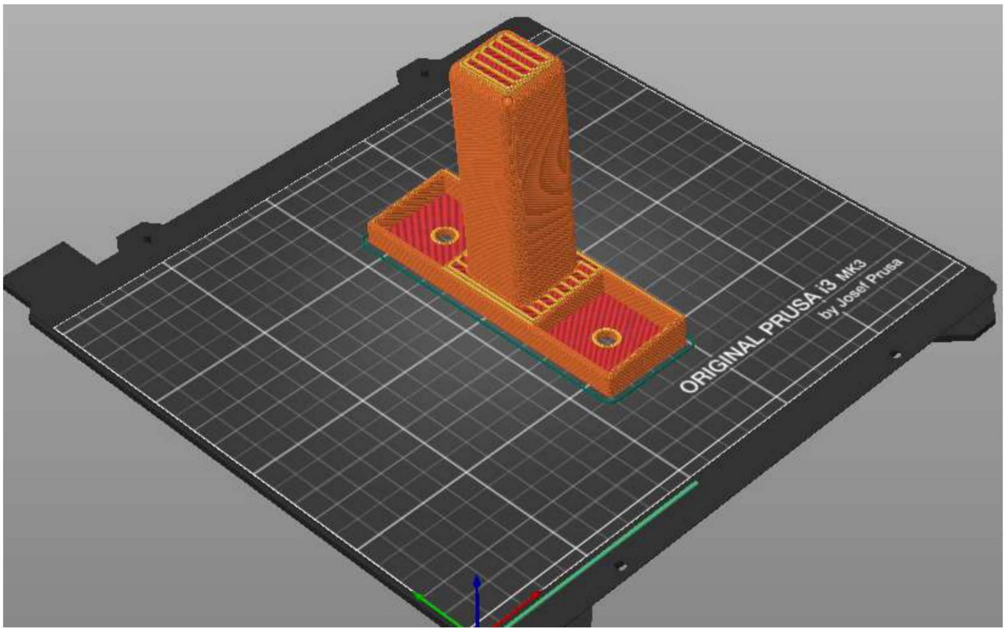
$C_m = 1.7307\ \mathrm{EUR}$

$C_e = 0.0377\ \mathrm{EUR}$

$C_{print} = 1.7684\ \mathrm{EUR}$

Figure A.34: Cost Calculation 16

Pistol Handle



$m_{fil} := 64.63$ g

$t_p := 1$ hr $+ 59$ min

$C_m = 1.9383$ EUR

$C_e = 0.0374$ EUR

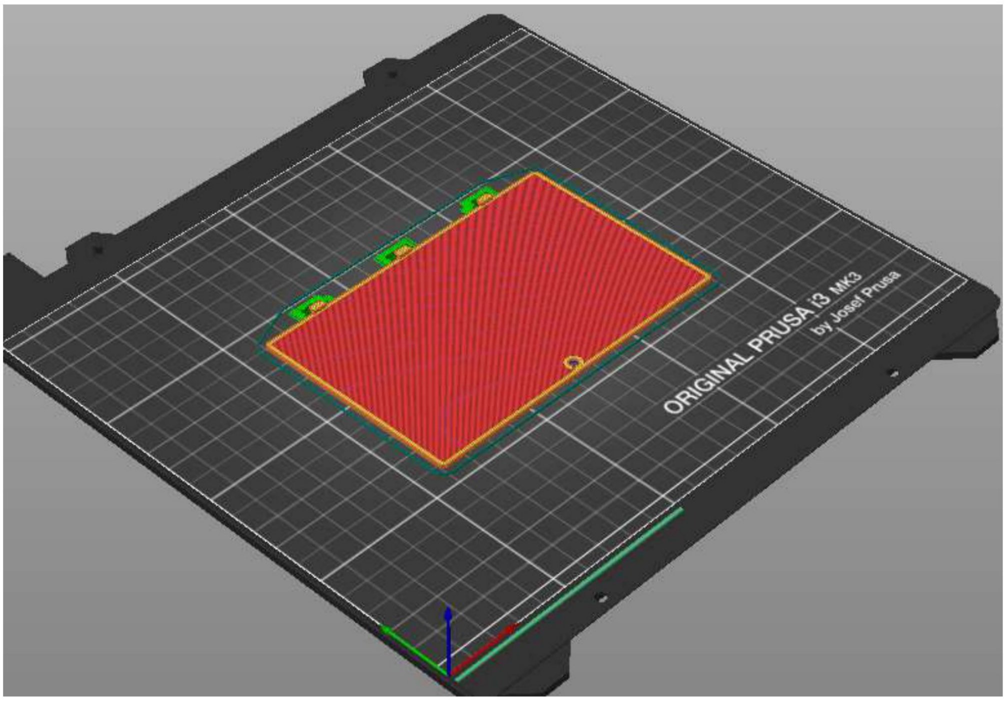
$C_{print} = 1.9756$ EUR

Sliced Info	
Used Filament (g)	64.63
Used Filament (m)	21.67
Used Filament (mm³)	52121.02
Cost	1.64
Estimated printing time:	
- normal mode	1h59m
- stealth mode	2h1m

17 / 19

Figure A.35: Cost Calculation 17

Battery Cover



$m_{fil} := 22.21$ g

$t_p := 0$ hr $+ 40$ min

$C_m = 0.6661$ EUR

$C_e = 0.0126$ EUR

$C_{print} = 0.6786$ EUR

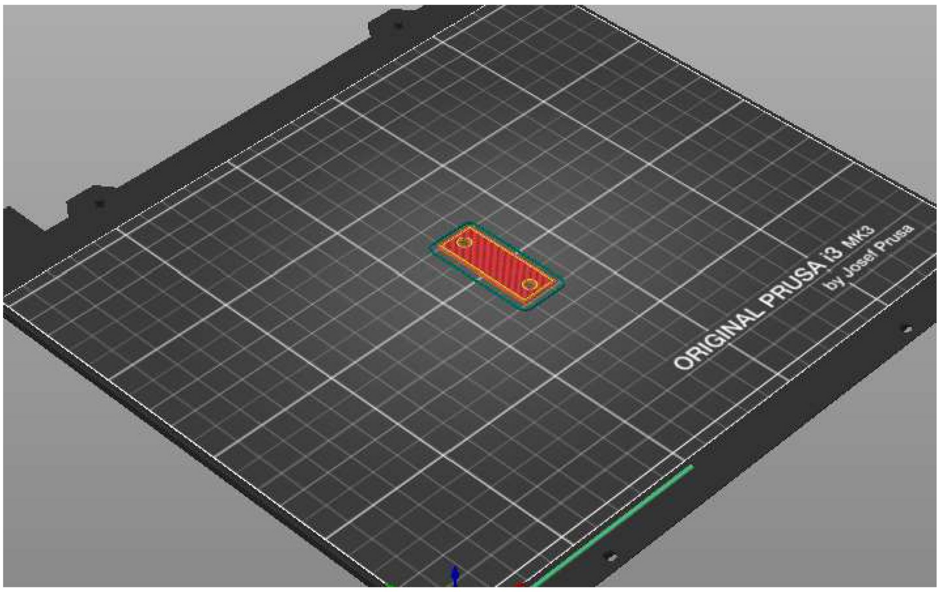
Sliced Info

Used Filament (g)	22.21
Used Filament (m)	7.45
Used Filament (mm³)	17910.60
Cost	0.56
Estimated printing time:	
- normal mode	40m
- stealth mode	40m

18 / 19

Figure A.36: Cost Calculation 18

Switch Cover



$m_{fil} := 1.31\ g$

$t_p := 0\ hr + 3\ min$

$C_m = 0.0393\ EUR$

$C_e = 0.0009\ EUR$

$C_{print} = 0.0402\ EUR$

Sliced Info

Used Filament (g)	1.31
Used Filament (m)	0.44
Used Filament (mm³)	1057.58
Cost	0.03
Estimated printing time:	
- normal mode	3m
- stealth mode	4m

Figure A.37: Cost Calculation 19

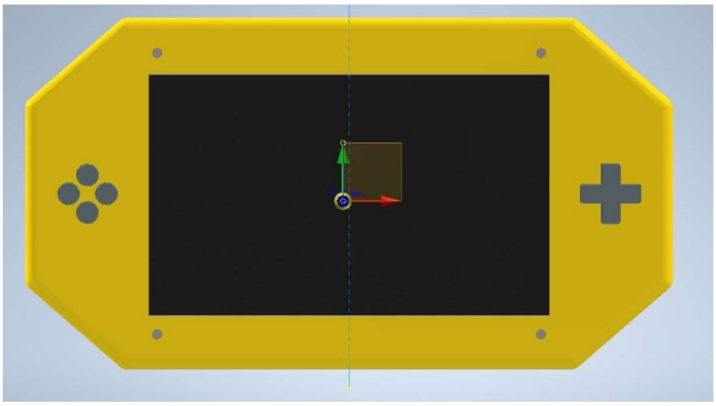
A.5 Evaluation Data

1 Oct 2023 17:07:41 - eval.sm

Evaluation Data

Variant 2

Center of gravity



▼ Measure Results

Minimum Distance	2.492 mm
Angle	0.00 deg

measured from middle plane, due to the nature of the body requiring two hand

Device Weight

☐ Include Cosmetic Welds

Mass 1190.623 g (Relati\

Ease of Assembly / Number of components

58

Swappable Parts

1. Quick change plate

Figure A.38: Evaluation 1

Device Size

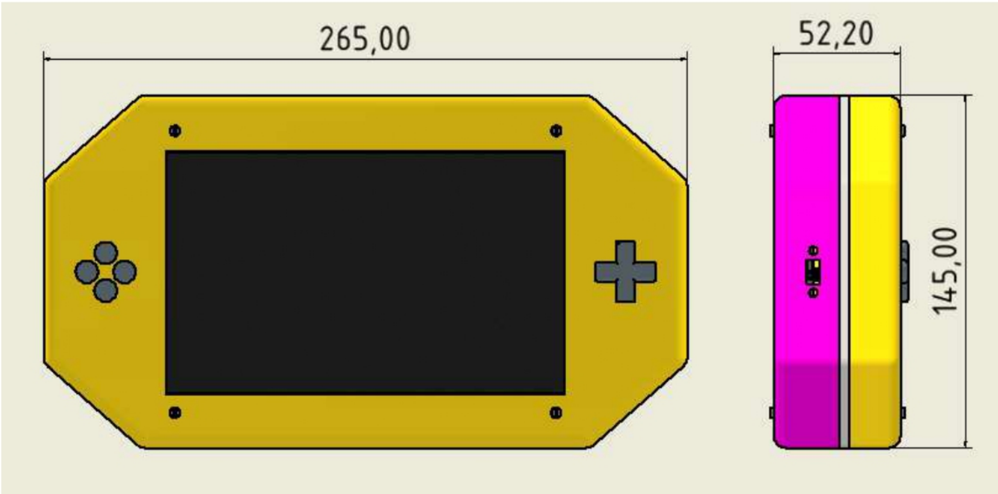
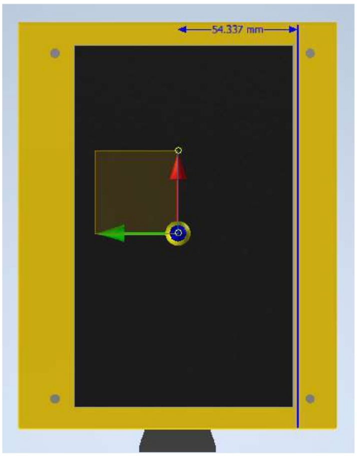
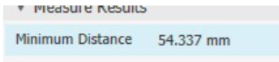


Figure A.39: Evaluation 2

Variant 3

Center of gravity

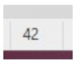




measured from either left or right bump.

Device Weight

Ease of Assembly / Number of components



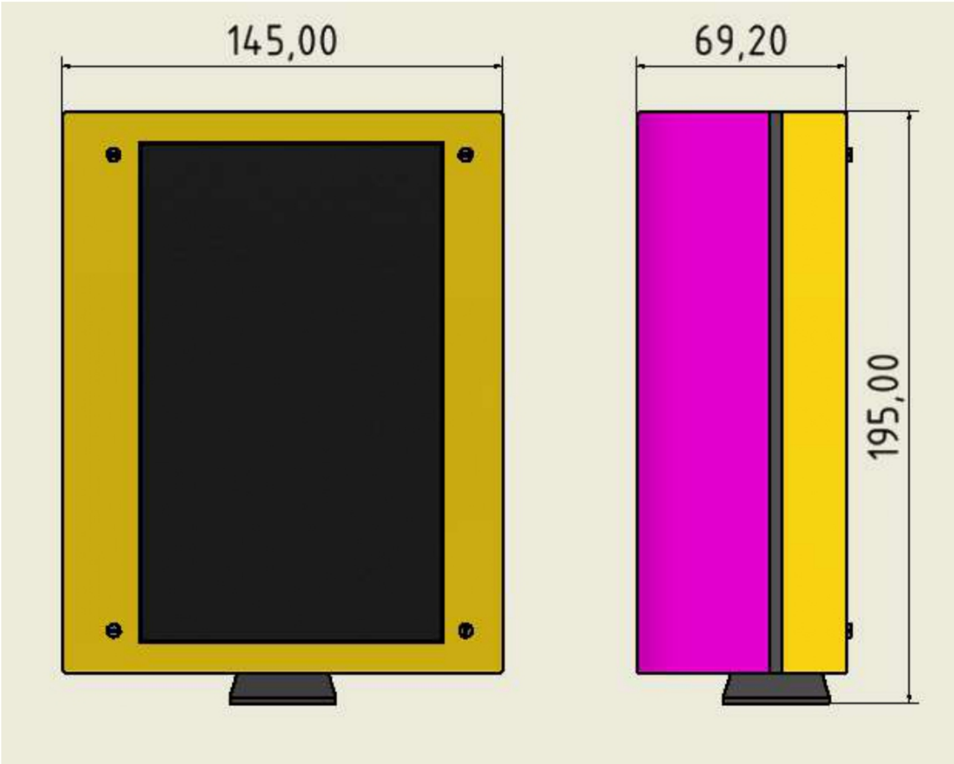
Swappable Parts

1. Battery

Figure A.40: Evaluation 3

1 Oct 2023 17:07:41 - eval.sm

Device Size

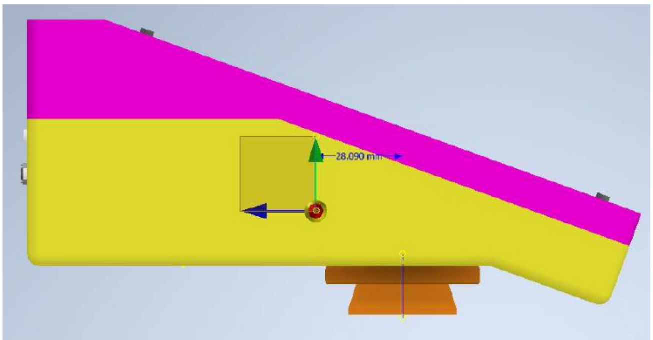


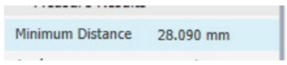
4 / 8

Figure A.41: Evaluation 4

Variant 6

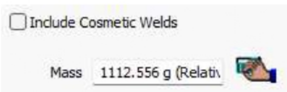
Center of gravity





measured from center of mounting point due to the handle and switch plate is placed there

Device Weight



Ease of Assembly / Number of components

Swappable Parts

1. Battery
2. Switch plate
3. Handle

Figure A.42: Evaluation 5

Device Size

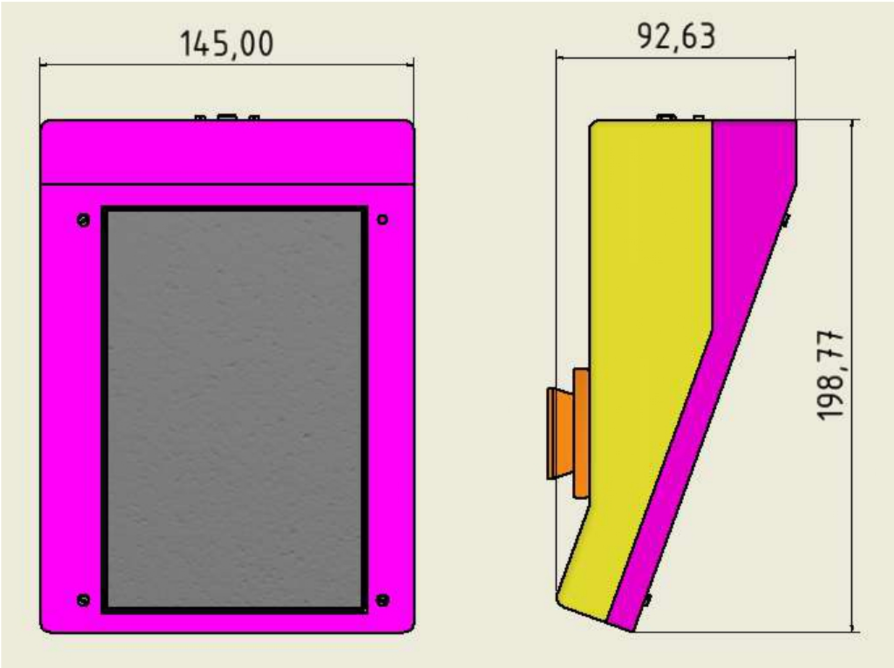
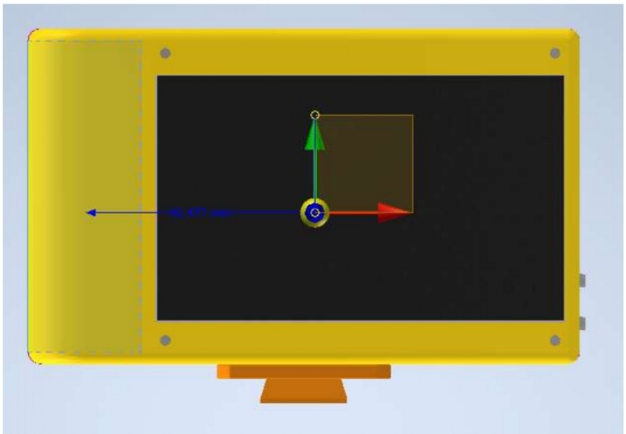
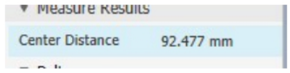


Figure A.43: Evaluation 6

Variant 7

Center of gravity





measured from center of left bump

Device Weight

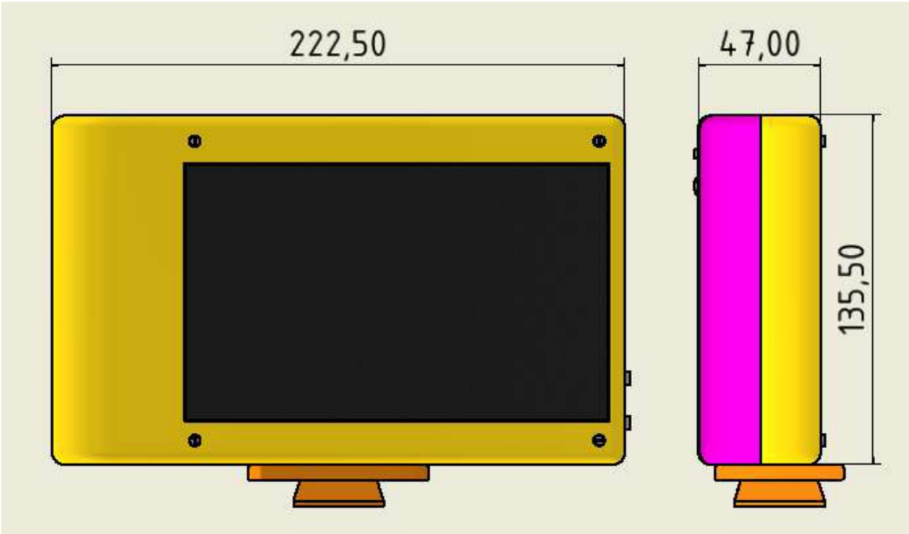
Ease of Assembly / Number of components

Swappable Parts

1. Switch plate

Figure A.44: Evaluation 7

Device Size



8 / 8

Figure A.45: Evaluation 8

A.6 Documentation

- Documentation
- Repository

A.7 Code snippets

A.8 C++ Unit Tests for Bank Account Management

Bank Account Class

```cpp
class BankAccount {
private:
    double balance;

public:
    BankAccount() : balance(0.0) {}

    void deposit(double amount) {
        if (amount < 0) {
            throw std::invalid_argument("Deposit amount
               must be positive");
        }
        balance += amount;
    }

    void withdraw(double amount) {
        if (amount < 0) {
            throw std::invalid_argument("Withdrawal amount
                must be positive");
```

```cpp
        }
        if (amount > balance) {
            throw std::runtime_error("Insufficient funds
               for withdrawal");
        }
        balance -= amount;
    }

    double getBalance() const {
        return balance;
    }
};
```

Unit Tests

Test Case: Deposit

```cpp
TEST(BankAccountTest, Deposit) {
    BankAccount account;
    account.deposit(100.0);
    EXPECT_EQ(account.getBalance(), 100.0);
}
```

Explanation: This test verifies that funds can be successfully deposited into the account. It creates an instance of `BankAccount`, deposits 100.0 units, and then checks if the balance matches the expected value of 100.0.

Test Case: Withdraw

```cpp
TEST(BankAccountTest, Withdraw) {
    BankAccount account;
    account.deposit(100.0);
    account.withdraw(50.0);
    EXPECT_EQ(account.getBalance(), 50.0);
```

```
}
```

Explanation: This test ensures that withdrawals are processed correctly. It first deposits 100.0 units into the account, then attempts to withdraw 50.0 units. The test verifies if the balance is now 50.0 units.

Test Case: Withdraw Too Much

```
TEST(BankAccountTest, WithdrawTooMuch) {
    BankAccount account;
    account.deposit(100.0);

    EXPECT_THROW(account.withdraw(150.0), std::
       runtime_error);
    EXPECT_EQ(account.getBalance(), 100.0);
}
```

Explanation: This test examines the scenario where an attempt is made to withdraw an amount greater than the available balance. It first deposits 100.0 units and then tries to withdraw 150.0 units. The test expects a `std::runtime_error` to be thrown, and ensures that the balance remains unchanged.

Test Case: Deposit Negative

```
TEST(BankAccountTest, DepositNegative) {
    BankAccount account;

    EXPECT_THROW(account.deposit(-50.0), std::
       invalid_argument);
    EXPECT_EQ(account.getBalance(), 0.0);
}
```

Explanation: This test handles the scenario where an attempt is made to deposit a negative amount. It expects a `std::invalid_argument` exception to be thrown. The test also verifies that the account balance remains unaffected.