University of Tartu

Faculty of Science and Technology

Institute of Technology

Matevž Borjan Zorec

XR TELEOPERATION DEMO DEVELOPMENT

Master's thesis (30 EAP)

Robotics and Computer Engineering

Supervisor:

Ulrich Norbisrath, PhD

Tartu, 2023

Abstract

This thesis designs an educational real-time visual feedback teleoperation demonstration. The importance of a good user experience is highlighted while showcasing the feasibility of using open-source solutions such as Godot Engine version 4 for teleoperation setups.

Reviewed literature narrowed design requirements, outlining that a representative teleoperation demonstration could provide a positive experience, intuitive movement control, direct real-time visual feedback for teleoperation and be open-sourced, with user and video stream evaluations as research objectives.

Employing design thinking, 'RoverXR' is iteratively developed with M5 RoverC-Pro for movement and serving WebSocket protocol real-time Motion JPEG high-definition video from Raspberry Pi v2.1 Camera Module via a Raspberry Pi Zero. Custom MPV player and Godot scenes were prepared, featuring video stream playback and providing a virtual user interface on the Meta Quest 2 headset.

User evaluation participants report a positive, engaging experience and provide helpful feedback, showcasing the potential of low-latency, high-quality video streaming, and virtual scene representation in teleoperation demonstrations for educational purposes.

Abstract in Estonian

Käesolevas lõputöös kavandatakse hariduslik reaalajas visuaalse tagasisidega teleoperatsiooni demonstratsioon. Rõhutatakse hea kasutajakogemuse tähtsust, demonstreerides samal ajal avatud lahenduste, näiteks Godot Engine (versioon 4), kasutamise võimalikkust teleoperatsioonide ülesseadmisel.

Kirjandusülevaate põhjal kitsendati disaininõudeid, tuues välja, et representatiivne teleoperatsiooni demonstratsioon võiks pakkuda positiivset kogemust, intuitiivset liikumise kontrolli, otsest reaalajas visuaalset tagasisidet teleoperatsioonile ja olla avatud lähtekoodiga, kusjuures uurimiseesmärkideks on kasutajauuring ja videovoo hindamine.

Kasutades disainimõtlemist, töötati iteratiivselt välja "RoverXR" lahendus, mis kasutab liikumiseks M5 RoverC-Pro platvormi ja Raspberry Pi v2.1 kaameramoodulist Raspberry Pi Zero kaudu reaalajas Motion JPEG kõrglahutusega video edastamiseks WebSocketi protokolli. Loodi kohandatud MPV-mängija ja Godot' stseenid, mis sisaldavad videovoo taasesitust ja pakuvad virtuaalset kasutajaliidest Meta Quest 2 peakomplektiga kasutamiseks.

Kasutajauuringu tagasiside põhjal oli kasutajatel positiivne ja kaasahaarav kogemus ning anti kasulik tagasisidet. See tõstab esile madala latentsusega kvaliteetse voogedastuse ja virtuaalse stseeni kujutamise potentsiaali hariduslikel eesmärkidel toimuvates teleoperatsiooni demonstratsioonides.

Märksõna: Teleoperatsioon, Godot Engine, video voogesitus, liitreaalsus, haridus

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine; T125 Automatiseerimine, robootika, juhtimistehnika

Contents

List of abbreviations

RPi - Raspberry Pi
VR - Virtual Reality
AR - Augmented Reality
XR - Extended Reality
HMD - Head Mounted Display
MJPEG - Motion JPEG (Joint Photographic Experts Group)
IP - Internet Protocol, Internet Protocol Address
PSU - Power Supply Unit
PC - Personal Computer
FPS, fps - frames per second

List of Figures

List of Tables

1 Introduction

Teleoperation allows human users to immerse into a distant or inaccessible environment to perform complex tasks [1]. Telerobotics is a rapidly developing area of research that enables remote operation of robots or other devices from a distance. Some solutions use first-person-view (FPV) displays [2]–[5] or VR and game engine frameworks [6]–[11] to facilitate this. These technologies have many potential applications in various areas, including logistics, mining, agriculture, search and rescue, surveillance, military, space exploration, manufacturing and healthcare [12], [13].

However, teleoperation can be a complex and challenging study area, requiring advanced technical skills and knowledge. As such, there is a growing need for simple, hands-on demonstrations that can introduce students and researchers to the basics of teleoperation and inspire them to pursue further study in this field. Moreover, a well-designed teleoperation demo can inspire and motivate students to pursue their research in this area. It may bring new and novel ideas for improving or expanding existing demos, developing new frameworks, and demonstrating the possibilities of teleoperation in a world of constraints and limitations.

In recent years, the University of Tartu has undertaken many works on using VR and game engines for teleoperation or telerobotics. Sherafatian [6] explored the teleoperation of remote-controlled toy cars in VR. Kõvask [7] developed a state-of-the-art VR driving simulation for physical test cars using LiDAR for mapping the surrounding environment. RCSnail [14] developed a toy rally track featuring small remote-controlled cars with onboard WiFi video streaming capabilities. The RCSnail track was a fun public teleoperation demonstration. These works together demonstrate the potential of teleoperation, game engines, and VR, in addition to providing valuable insights into the challenges and opportunities of this field.

The Donkey Car open-source do-it-yourself self-driving platform for small-scale cars [15] successfully provides a framework for developing self-driving solutions by enabling testing on a smaller form factor and overall reducing the cost of deploying autonomous driving solutions to real life. Sherafatian [6] and Kõvask [7] based their work on this platform, where they highlighted areas of improvement, such as streaming quality and network architecture optimisations.

This thesis documents the development of a low-cost, small-form-factor teleoperation demonstration specifically designed for use during open lab events to provide students with hands-on experience. These events are interactive gatherings, typically allowing participants to engage in demonstrations, exhibits, and interactive activities in a collaborative and informal setting. The developed toy teleoperation demo presented in this thesis uses affordable hardware, including the Raspberry Pi Zero 1 W [16], M5Stack Rover C Pro [17] and Meta Quest 2 VR HMD [18, p. 2]. The use of the smaller form factor of the Rover, which incorporates a

holonomic and omnidirectional movement drivetrain, presents additional design challenges compared to larger platforms like the Donkey Car.

This demo can showcase the benefits and limitations of VR, game engines with human-computer interaction (HCI), and telerobotics in a real-world context. Finally, the developed demonstration could help align student expectations when considering studying at the intersection of three University of Tartu labs under the Computer Science or the Science and Technology Institutes: Chair of Distributed Systems [19], Computer Graphics and VR Lab [20], and Intelligent Material Systems Lab [21].

1.1 Problem Statement

Teleoperation is an important area of research in robotics and automation, but it can be challenging for students to understand and appreciate its potential without hands-on experience. To address this need, the main goal of this thesis is to develop an artefact which demonstrates fundamental principles and capabilities of teleoperation clearly and engagingly.

Creating the demonstration required overcoming various design challenges, including Godot Engine [22] streaming support, video streaming limitations, general physical constraints, immersion and comfort. This thesis explores different video streaming solutions and configurations to optimise the final demonstration, concluding with evaluating the presented solution with volunteer participants.

1.2 Research Objective

Based on the problem statement, the following research objective was outlined for this thesis:

Primary objective:

Evaluate how users subjectively perceive the artefact as a teleoperation demo.

Optional objective:

Evaluate the video streaming quality of the artefact for real-time teleoperation.

The primary objective is understanding whether the developed artefact demonstrates teleoperation clearly and engagingly. The additional optional objective provides insight into understanding the success of the developed artefact as a teleoperation demonstration in terms of real-time visual feedback.

2 Background & Related Work

This chapter provides a concise overview of teleoperation research related to mobile robots. It emphasises common trends and provides insights into what makes up a teleoperation demonstration.

This thesis emphasises identifying common teleoperation principles, particularly those related to visual feedback on mobile robots and autonomous vehicles. This understanding helped identify and prioritise critical features during artefact development. Finally, the chapter explores the most common teleoperation applications and how it is relevant to demo development and will provide examples of industrial solutions.

2.1 Teleoperation of Mobile Robots

Moniruzzaman et al. surveyed teleoperation methods and enhancement techniques for mobile robots [13]: enhancement of conventional teleoperation is necessary for effective real-time teleoperation of robotic vehicles, particularly when complex tasks are needed. The term 'teleoperation' derives from the Greek word 'tele,' which translates to *'far off'* or *'at a distance'*. It refers to the human capacity to sense and manipulate objects remotely. Figure 2.1 concisely presents the critical elements of teleoperated mobile robots.
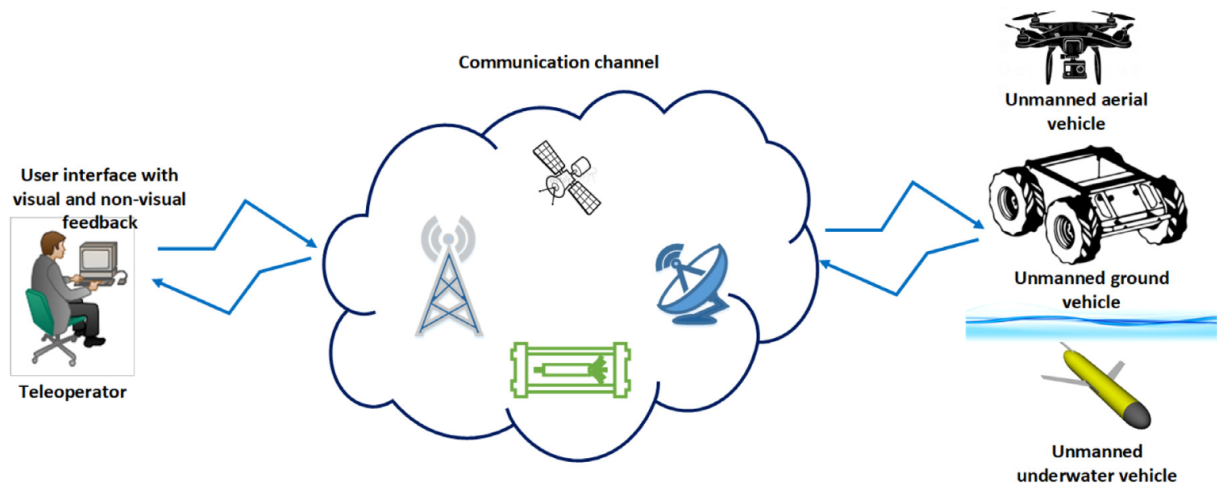
Figure 2.1: Schematic diagram of robotic teleoperation [13]. Teleoperators, or remote operators, communicate with mobile robots using various communication channels, including radio links, satellite connections, cellular networks, and wired or wireless Internet connections. Remote operators send commands to the mobile robot and receive visual and non-visual feedback via a user interface.

Moniruzzaman et al. [13] reinforce that mobile robots are becoming increasingly crucial for performing tasks that are difficult or dangerous for humans. Technology advancement for

improved mobile robotic teleoperation and remote control is vital to enable these robots to operate with increasing autonomy levels and in complex environments. Even platforms capable of high levels of autonomy require some level of human monitoring and external direction.

2.1.1 Teleoperation Methods

Based on the method of teleoperated control, Fong and Thorpe (2001) [23] documented vehicle teleoperation interfaces and classified them into three essential categories: direct teleoperation, supervisory teleoperation, and multimodal teleoperation.

Direct teleoperation involves the teleoperator relying on visual feedback transmitted from the remote vehicle and providing control input using traditional controllers. Supervisory teleoperation is where the teleoperator and the operated robot share duty and control of the whole system - the mobile robot is, in some regard, autonomous, and the operator can occasionally take direct control. Multimodal teleoperation collects and synthesises more than one sensor cue, providing the remote operator with a multimodal view representation of the world. This enhanced interface helps reduce the remote operator's cognitive load [24].

Direct teleoperation is less forgiving to latency, data loss, and other communication issues than multimodal or supervisory teleoperation. Supervisory and multimodal teleoperation methods are more reliant on remote computing capabilities.

At the time of publication, Fong and Thorpe (2001) [23] explained that direct teleoperation was the most common, traditional method for vehicle teleoperation. However, according to Moniruzzaman et al. (2021) [13], supervisory control is presently the most common control mechanism for the teleoperation of mobile robots, followed by direct teleoperation, with multimodal approaches emerging but not yet widely used.

2.1.2 Common Teleoperation Applications

Some of the most common teleoperation applications include the teleoperation of autonomous vehicles, machines, robots, and drones. Remote operators use teleoperation solutions in hazardous environments such as deep sea exploration, space exploration, and nuclear power plant maintenance and for medical purposes such as telesurgery and remote consultations [13].

Teleoperation can provide users with an immersive and interactive experience in the development of demos. There are numerous companies and startups specialising in teleoperation. Some examples of enterprises in this field include Voysys [25], Clevon [26], Starship [27], Ottopia [28], Fernride [29], Seafar [30], Shadow Robot [31] and Roboauto [32]. A great example of teleoperation in the space industry is ESA's ESTEC Teleoperation Workcells with the Interact Centaur robot platform [33].

2.1.3 VR & Teleoperation

Jerald (2016) [34], in his book about VR for human-centred design, explains that VR aims to engage users in the experience through *'immersion'*. Immersion is the technology's ability to make the user feel present in a virtual world, but it is only one part of the VR experience. He explains that the subjective experience of immersion is known as *'presence'*, an internal psychological and physiological state of the user. When present, users are not aware of the technology but instead perceive the virtual objects and characters. Users who feel highly present consider the VR experience a place visited rather than simply perceived. Thus, presence is a crucial aspect of the VR experience that can enhance the user's perception of the virtual world.

In their survey, Moniruzzaman et al. [13] found that VR can immerse a teleoperator into the environment of a robotic platform with high perceptual awareness. They add that most surveyed studies that attempted to enhance teleoperator perception through VR relied on off-the-shelf commercial solutions or gaming tools such as the Meta Quest 2 (Figure 3.6, top left) used in this thesis.

Additionally, Jerald (2016) [34] states in his opening remarks that when people design a VR experience well, the results are brilliant and pleasurable experiences that go beyond what we can do in the real world. When an experience is uncomfortable or negative, users may give up on the product and never try it again. Thus, the quality of an experience is essential for everything, for it determines our quality of life on a moment-by-moment basis.

In summary, teleoperation refers to the human capacity to sense and manipulate objects remotely. While no longer the most common, the traditional method is direct visual teleoperation. This method involves the teleoperator relying on visual feedback from the remote vehicle and providing control input using traditional controllers. Teleoperation has numerous applications across various industries and is relevant to developing demos in various fields. VR can immerse a teleoperator into the environment of a robotic platform with high perceptual awareness.

Based on these findings, the developed teleoperation demonstration should enable users to sense and manipulate objects remotely and provide real-time visual feedback. Additionally, the demo can incorporate VR to better immerse users into the environment of the remote platform.

2.2 Teleoperation Challenges

In their paper, Tener and Lanir [24] explore the challenges and provide guidelines for teleoperation interfaces when remotely operating autonomous vehicles. In particular, they highlight challenges associated with remote driving of autonomous vehicles, including the lack of physical feedback, cognitive and perception factors, video and communication quality, remote

interaction with humans, impaired visibility, and lack of sounds. Figure 2.2 categorically depicts the critical challenges of teleoperating remote vehicles based on the author's findings.
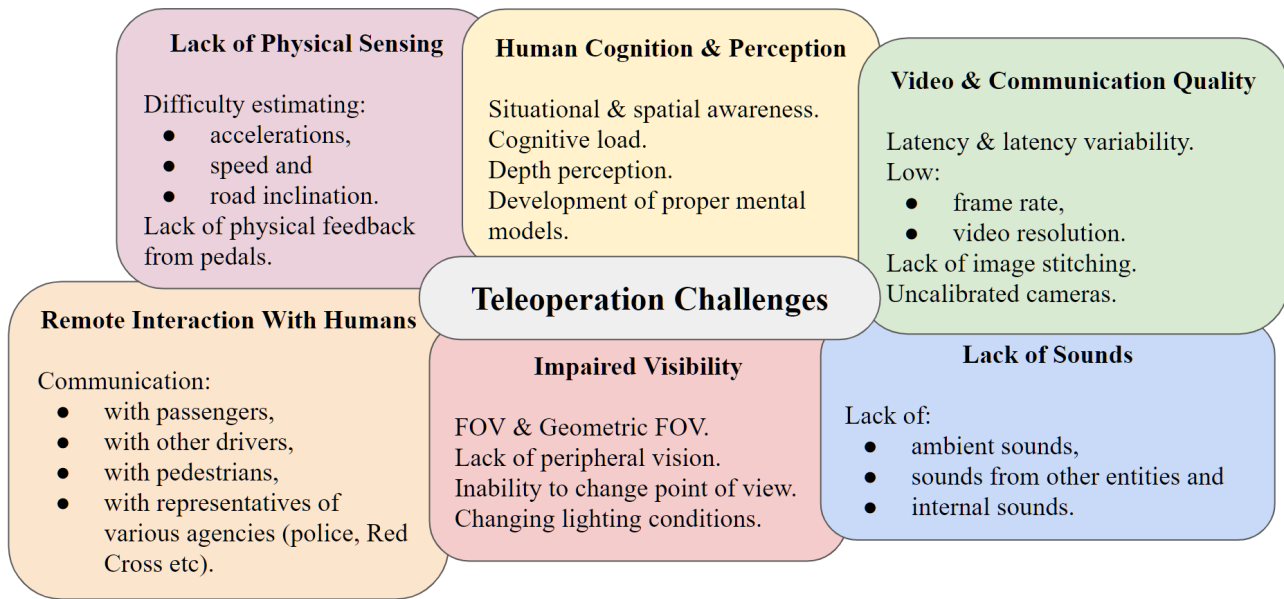
Lack of Physical Sensing

Difficulty estimating:
- accelerations,
- speed and
- road inclination.

Lack of physical feedback from pedals.

Human Cognition & Perception

Situational & spatial awareness.
Cognitive load.
Depth perception.
Development of proper mental models.

Video & Communication Quality

Latency & latency variability.
Low:
- frame rate,
- video resolution.

Lack of image stitching.
Uncalibrated cameras.

Teleoperation Challenges

Remote Interaction With Humans

Communication:
- with passengers,
- with other drivers,
- with pedestrians,
- with representatives of various agencies (police, Red Cross etc).

Impaired Visibility

FOV & Geometric FOV.
Lack of peripheral vision.
Inability to change point of view.
Changing lighting conditions.

Lack of Sounds

Lack of:
- ambient sounds,
- sounds from other entities and
- internal sounds.

Figure 2.2: Based on '*categories of teleoperation challenges*' by Tener et al. [24]. By conducting a survey and noting observations, they extrapolated these challenges for the teleoperation of autonomous vehicles.

Tener and Lanir [24] go on to explain that the lack of physical feedback makes it difficult for remote operators to control a vehicle, leading to nausea and dizziness. Additionally, they found that remote operators face challenges with spatial awareness, depth perception, and the development of mental models. In addition, the authors point out that interaction with humans is crucial in traffic scenarios. Poor visibility and lack of sound also pose significant challenges for the teleoperation of remote vehicles. Finally, video and communication quality affect remote driving, leading to latency-related issues and low-quality video stitching.

Regarding network communication protocols for low-latency video streaming, WebSocket [35], Transmission Control Protocol (TCP) [36], User Datagram Protocol (UDP) [37], and Hypertext Transfer Protocol (HTTP) [38] have all been used for video streaming.

Also, network architecture can directly impact the quality of a teleoperation experience.
Kaknjo et al. [39] explain that latency is an inherent property of a communication channel, regardless of the type, medium, and protocols used for the data transmission.

Video stream latency, typically measured as pixel-to-pixel delay in milliseconds, along with jitter, the inconsistent transmission of video frames, are the two most prevalent challenges of visual feedback teleoperation. Video quality, bitrate, and network bandwidth are other crucial factors that affect the overall streaming quality [13].

A direct visual teleoperation demonstration should be carefully planned to provide the best available network architecture. This includes selecting an appropriate network communication protocol. WebSocket is an efficient and low-latency protocol for bi-directional communication

over a single connection [40]. TCP is a reliable, connection-oriented protocol that guarantees in-order packet delivery [41]. UDP is a connectionless protocol that provides low-latency, best-effort packet delivery [42]. HTTP is a connectionless, widely used protocol for web-based applications that provides high-level abstractions for data exchange [43]. The WebSocket protocol was chosen in the development process, discussed in 3.2.1 Network Architecture.

In summary, the main challenges of teleoperation include the lack of physical feedback, cognitive and perception factors, video and communication quality, remote interaction with humans, impaired visibility, and lack of sounds. Based on these challenges, the developed teleoperation demonstration should showcase these issues and provide some examples of solutions.

2.2.1 Key VR Teleoperation Challenges & Limitations

For robotic teleoperation, a virtual environment helps to create a psychological state in which teleoperators can identify themselves as present in the virtual environment [44, p. 8].

Milgram (1997) [45], in his work exploring AR teleoperation interfaces for unstructured environments, found that for visual feedback-based teleoperation interfaces, the need to trade-off between frame rates, pixels per frame, bits per frame (levels of brightness or greyscale) can be a challenge and a particular limitation of the AR headset technology used.

Jerald and Whitton (2009) [46] explored how scene-motion thresholds relate to latency for VR HMDs. They found that for latencies below ~100ms, users tend not to perceive latency directly but rather its indirect consequences - a static virtual scene could appear unstable in a virtual space when users move their heads.

In his book, Jerald (2016) [34], in chapter 15 goes into great detail, explaining that latency in an HMD-based system causes visual cues to lag behind other perceptual cues, creating sensory conflict, which may lead to nausea and discomfort. If an HMD user observes latency combined with head movement, the visual scene reconstruction can move incorrectly. This effect is called "swimming"; the lagged scene movement latency-induced effect severely impedes usability [34]. Based on these and other negative effects of latency in a virtual setting, Jerald emphasises the need for VR scene designers to carefully consider and understand latency to mitigate and minimise its effects. Figure 2.3 highlights the cumulative effect of various delay sources in a typical VR HMD scene.
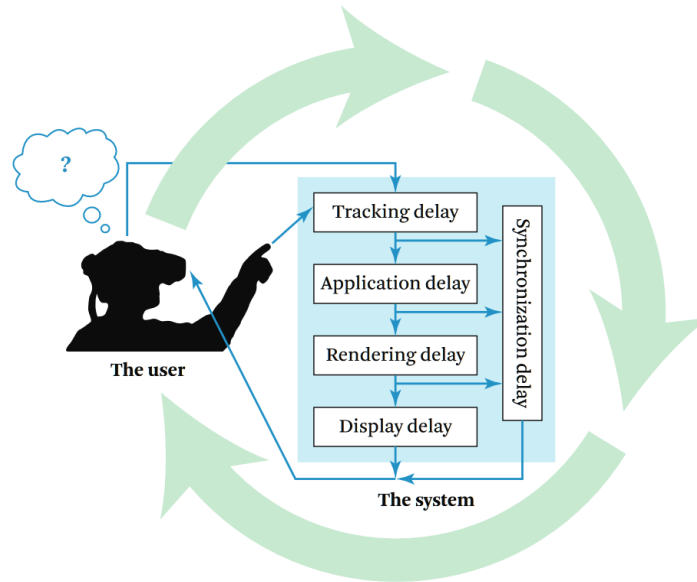
Figure 2.3: Several delays contribute to total system delay. In chapter 15.4, Jerald (2016) [34] presents this figure. End-to-end system delay is the result of individual system components' delays. Total system delay includes tracking user gestures, running applications, rendering, displaying and synchronising these components.

In Chapter 14.1, Jerald (2016) [34] touches on the hardware challenges of VR HMDs, highlighting physical fatigue from the prolonged physical activity of the user, mainly when the mass of the headset is not centred correctly. In Chapter 14.2, Jerald (2016) [34] discusses the relevance of good fit and comfort of the headset, which is important and can be a source of discomfort and headaches if the pressure points between the HMD and the user's head are improper. Pressure points typically occur around the eye sockets, ears, nose, forehead, and back of the head. Loose-fitting HMDs can cause slippage, leading to skin irritation and small scene motions. While lighter HMDs can reduce slippage, some small motions can be compensated for by adjusting the scene to maintain stability relative to the user's eyes.

Based on these findings and observations, careful attention should be paid to user comfort when developing or conducting VR teleoperation demonstrations and the evaluation described in Chapter 4. In fact, during the evaluation, participants did express some HMD-related discomforts, as described above.

2.2.2 User Interface Suggestions

In addition to outlining the challenges in teleoperating remote autonomous vehicles discussed previously, Tener and Lanir (2022) [24] also provide several suggestions for improving the user experience. They suggest enriching the video feed visualisation of the user interface by including informative cues. While their research focuses on the remote operation of autonomous vehicles, many of their suggestions also apply to mobile robots. The following suggestions [24] were particularly interesting for this thesis.

1. Add UI cues to bridge the physical disconnect.

They recommend adding visual cues to the user interface, indicating felt and applied forces with appropriate visualisations, such as brake and throttle pedal applied pressures.

2. Visualise Remote Vehicles' direction based on the current position of the steering wheel.

Providing continuous feedback on vehicle trajectory based on steering wheel rotation angle is essential. They recommend projecting the future trajectory onto the video feed, helping to reduce remote operators' cognitive effort and improve situational awareness. They add that this can also help improve spatial awareness by accurately depicting the vehicle's width.

3. Add depth perception cues.

Adding visual cues using colour gradients to the video feed can help depict the movements of traffic participants. Depicting relative movements of objects can help reduce the cognitive effort of remote operators and further improve situational awareness.

4. Visualise network and video quality.

Video and communication quality are critical challenges of both VR and teleoperation.

To help mitigate and improve the user experience, visualising network quality, speed and video frame rate at any time can help increase situational awareness and help prevent misinterpreting visual information.

According to these suggestions, teleoperation demonstrations could benefit from extended user interfaces, including various intuitive cues and information visualisations.

2.3 Summary & Design Requirements

Based on the reviewed works and listed applications, the key features of a teleoperation demonstration should incorporate a traditional, visual feedback-based solution, which enables a user to control a remote robot base directly. This combination is simultaneously representative of most teleoperation solutions covered and an example of the critical principles and challenges of teleoperation.
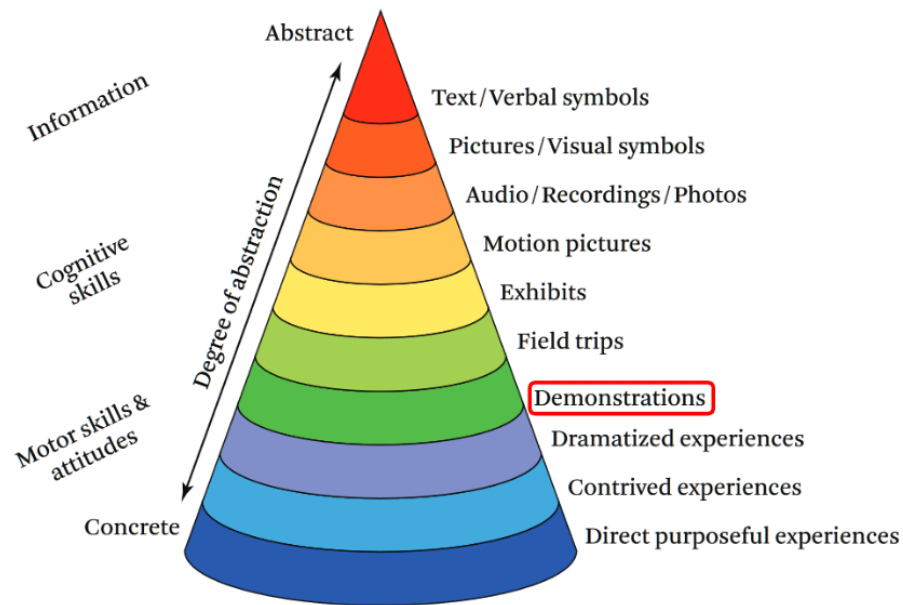
Figure 2.5: The Cone of Experience. Jerald [34] VR uses many levels of abstraction, adapted from Dale, E. (1969). *Audio-Visual Methods in Teaching* (3rd ed.). The Dryden Press.

In his book's introduction, Jerald (2016) [34] explains that whether through immersive storytelling, grasping abstract concepts, or honing practical skills, VR leverages the full potential of human sensory capabilities and motor skills to enhance learning. Direct, purposeful experiences provide the best basis for understanding (Figure 2.5). This helps provide further incentive and reassurance by validating that a functional, meaningful teleoperation demonstration could be a useful pedagogical tool, leveraging the demonstration experience to provide an entry-level understanding of teleoperation as a subject matter.

The use of VR technology may impose additional limitations on the overall teleoperation experience, namely concerns about the overall comfort of the headset and added latency. However, the potential for improved immersion and enhanced situational awareness are important factors to highlight in potential VR teleoperation demonstrations.

Finally, based on the supervisor's inputs, the solution should be accessible and freely adaptable. This way, students, who find inspiration and drive to pursue this area, may also use provided examples of this demonstration solution for their endeavours.

The following design requirements are presented based on the key features of direct visual teleoperation:

1. It must provide a positive experience.
2. It must incorporate key teleoperation solution demonstrations, including:
   a. capable of remote control, with intuitive and responsive movements,
   b. able to transmit visual feedback in real-time, and
   c. provide a VR/AR user interface.
3. It must be open-sourced, documented, easy to use and ready to build on for future work.

An overall positive user experience is vital for demonstrating teleoperation to students.

The solution should also feature some intuitive movement control, accompanied by real-time visual feedback, akin to traditional direct visual teleoperation methods described by Fong and Thorpe [23], mentioned previously.

By providing a VR or AR user interface, students could directly experience the benefits of immersion and enhanced situational awareness.

Lastly, students can access all its resources and examples freely by ensuring the demonstration is open-sourced [47]. This helps streamline their learning process by providing free access to all related resources, documentation and provided examples. The developed prototype is intended to be built upon in later research works and as a teaching tool, thus requiring a permissible, open-use license such as the MIT license [48].

3 Design & Development

This chapter describes the process of creating the teleoperation demonstration artefact, called *'RoverXR'*, composed of hardware and software components. These work together to demonstrate fundamental principles and capabilities of teleoperation.

The goal of the design and development process aimed to prepare a teleoperation demonstration, as discussed in 2.3, which can help inspire students to contribute to this exciting and rapidly growing field. The resulting artefact should be an easily replicable and adaptable solution, for use in various educational settings, primarily for demonstration purposes.

This chapter provides descriptions of design choices and implementations. Furthermore, it discusses the limitations and challenges encountered during the development process.

3.1 Hardware

In any teleoperation demonstration, reliable components and hardware design are crucial for providing physical presence and control in a remote environment. The design of the hardware is an important factor in determining the quality and responsiveness of a teleoperation system.

Throughout the development of RoverXR, I employed a design thinking approach, iterating on multiple versions of the adapter kit for the rover base. The subsequent subsections delineate this method, encompassing component selection and RoverXR design, culminating in a comprehensive bill of materials.

3.1.1 Component Selection

The initial design milestone was determining the essential feature list the final solution should incorporate. Among these, the highest priority was the capability to serve a high-quality video stream to a VR scene. The second highest priority was to provide movement control to the whole setup. The supervisor introduced an additional form factor criteria compared to previous works due to a current incentive towards miniaturising teleoperation demonstrations [6], [7]. After a quick elimination round, based on component availability and form factor, a preliminary list of possible components was ready for a more detailed analysis.

A high-quality video stream requires a camera and accompanying networked computing unit, with excellent documentation and packaged in a compact form factor. The comparison involved two candidates: RPi Zero 1 W (equipped with a Pi camera module) and ESP32-CAM [49]. The

analysis considered the features of each product, with a specific focus on evaluating their advantages and disadvantages. Please, see Table 1 below for a breakdown of the key points between the two options.

Table 1: Pro et Contra Comparison: RPi Zero W vs ESP32-CAM.

Raspberry Pi Zero 1 W		ESP32-CAM	
Pro	*Contra*	*Pro*	*Contra*
+ well documented + hardware encoding + full OS + camera variety + can deploy ROS + low power consumption + more I/O	- limited compute - larger form factor - excess features - 15EUR+	+ well documented + low power consumption + smaller form factor + 5EUR+	- very limited compute - no hardware encoding - no autofocus - manufacturing quality - less I/O

Ultimately, the RPi Zero 1 W was selected due primarily to its hardware encoding support, camera module variety and a possibility of a swap-in replacement with upgraded compute capabilities - the RPi Zero 2 W. The RPi Camera Module v2.1 [50] was selected as the video input device for testing purposes, with a later goal to switch to the more feature-rich Camera Module v3 [51]. In hindsight, the v3 module could have been selected from the start. Both are well-documented and support hardware video encoding.

The M5 Stack RoverC Pro [17], a small, omnidirectional robot base, was selected as the main rover platform for various reasons. Firstly, there was an existing incentive to miniaturise teleoperation demonstrations, as this allows for improved transportability and accessibility compared to previous works, including the possibility of developing autonomous driving solutions in a scaled-down cityscape.

Secondly, the RoverC Pro was available in local stock and underutilised, making it a practical choice. It was also an attractive option: aside from its compact size, it includes a ready-made movement control solution and its own Li-Ion battery power source. However, the included battery's capacity was estimated insufficient to supply both the RoverC and the RPi Zero 1 W long enough for a prolonged demonstration event.

As a result, a second power supply was included, dedicated to powering the RPi Zero. The Wemos 18650 PSU [52] was selected, which provides a suitable, stable power source for the RPi and includes essential battery charging safety features. Unfortunately, in testing, it was found that the side 5V and 3.3V power rails were not controllable with the provided switch. To resolve this, a breakout board with a power switch was installed to supply 5V power directly to the 5V GPIO pins of the RPi Zero. The breakout board also serves to ease future upgrades by providing access to the complete set of GPIO pins the RPi Zero 1 W provides. Figure 3.1 showcases the final components used to develop the teleoperation demonstration solution.
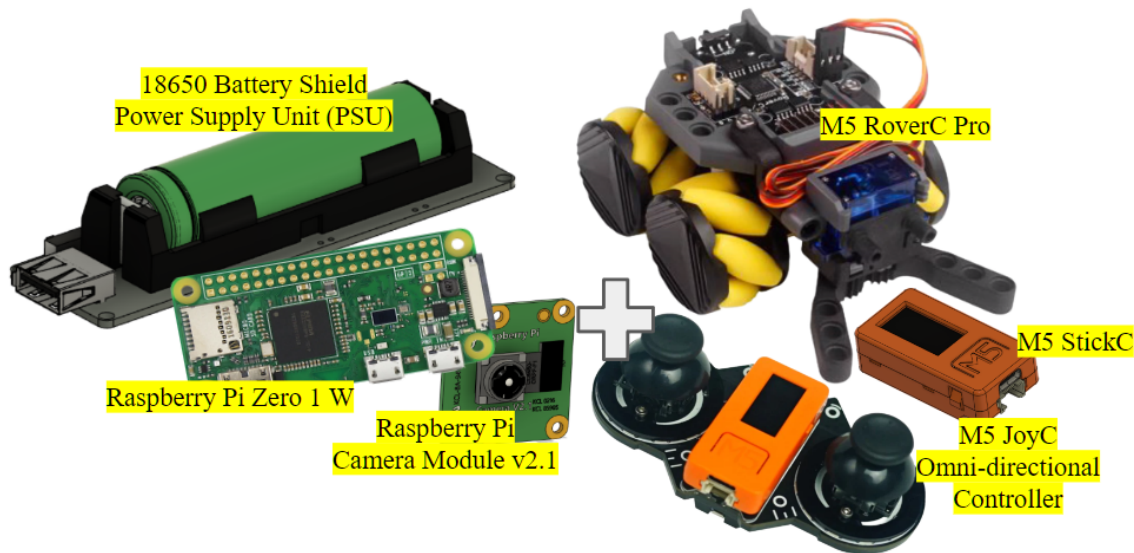
Figure 3.1: Final components to integrate into a teleoperation demonstration solution. Raspberry Pi Zero 1 W and Camera Module v2.1, powered by a 18650 Battery Shield PSU, and the M5 suite: RoverC Pro and JoyC remote controller. Two M5 StickC IoT development kits provide programmable functionalities to the Rover and remote.

Powering the RoverC and the RPi Zero independently successfully extended the overall configuration's battery runtime. However, as the final demonstration solution also includes the Meta Quest 2 VR HMD, with its battery power supply and runtime duration, estimating more precisely how long the demo could last can be difficult. The RoverC's battery would run out first, only after about 1.5 hours of aggressive, continuous use, meaning the overall teleoperation demonstration would be viable for at least that long before having to take a break and recharge (Figure 3.2). However, while running user evaluations, the actual time the demo lasted was not limited by the RoverC's battery. Instead, surprisingly, its M5 StickC microcontroller ran out first, just shy of the 1.5-hour time.

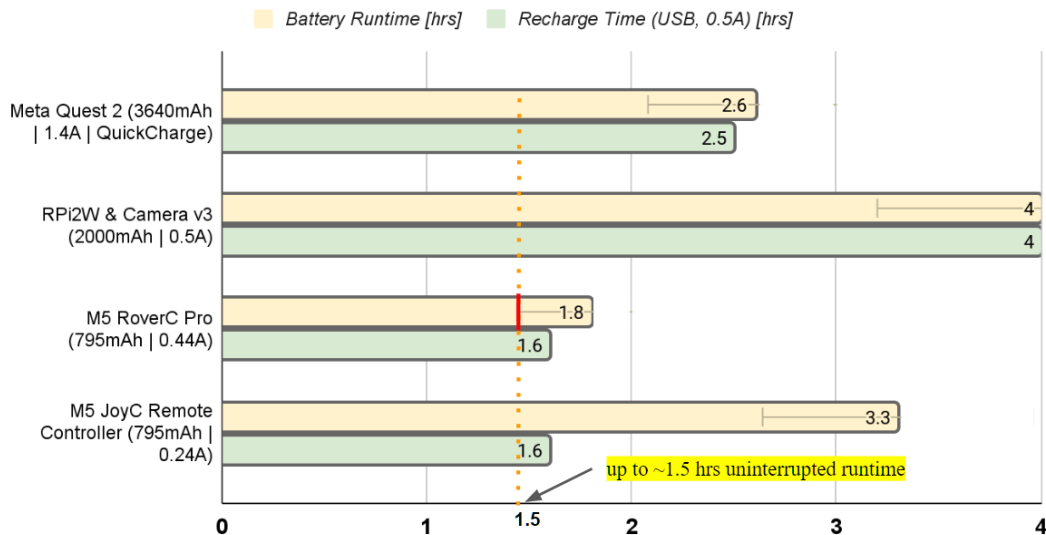Figure 3.2: Battery run and recharge times per component, expressed in hours. These values are calculated based on available manufacturer-provided specifications for battery capacities and peak load current values.

Figure 3.2 shows how long an individual battery-powered component of the teleoperation demo would last, assuming peak load current conditions. It also demonstrates how long it takes to

recharge each component using standard USB 2.0 ports, rated at 0.5A. The Meta Quest 2 supports quick charging, dropping the recharging time to 2.5 hours (shown above) from 7 hours (not shown above) if using a 0.5A USB port. As all components use Lithium-Ion based batteries, the runtime error bars show the last 20% of capacity, helping to identify a safe runtime time to help prevent over-discharging [53], [54].

After completing the initial component selection, the next stage of the development process involved several design iterations of integration attempts, discussed in detail in the next section.

3.1.2 RoverXR Design

The end goal of the design process was to create something like a toy for educational teleoperation demonstration purposes - specifically, for university students. Thus, it should be something a student could interact with, pick up, play with, explore and thoroughly investigate.

Integrating the components required a meticulous process, where rapid prototyping and design thinking was applied to produce a single combined assembly. Simply put, RoverXR comprises a gantry that holds a Raspberry Pi Zero 1 W, a camera and a 18650 PSU. The gantry is then attached to the M5 RoverC Pro robot base. The gantry is a multipart structure, 3D printed from PLA [55] filaments.

Ultimately, two rover versions were prepared, both weighing approximately 340g. Their external dimensions are also similar, at just under 12 cm in height, 15 cm in length and 8 cm in width at the base. Figure 3.3 showcases Rover version 1 (Rover v1) on a scale with various components and printed test pieces, highlighting the iterative design process approach.

Figure 3.3: Printed part tests & breakout board planning. Rover v1 weighed ~342g. Rover v2's design process focused on the lens mount (top middle), lens placement and camera specifications, ensuring good image quality. A breakout board with a switch was added to control the 18650 PSU (top right) power rail.

Rover v1's key design features include a lightweight adapter kit design, adjustable camera mount and good interface access. The essential design drivers included maintaining the overall small form factor and keeping track of the centre of mass to ensure good driving performance and stability. At the same time, the gantry's good stiffness and rigidity were priorities and minimised overall weight. Concept brainstorming occurred in the first design iteration stage, which included understanding the key design drivers. This phase concluded with Rover version 0, which represented a rough sketch of the final concept layout, shown in Figure 3.4 in the bottom left, annotated as 'Phase I.', also depicting the overall RoverXR design process.
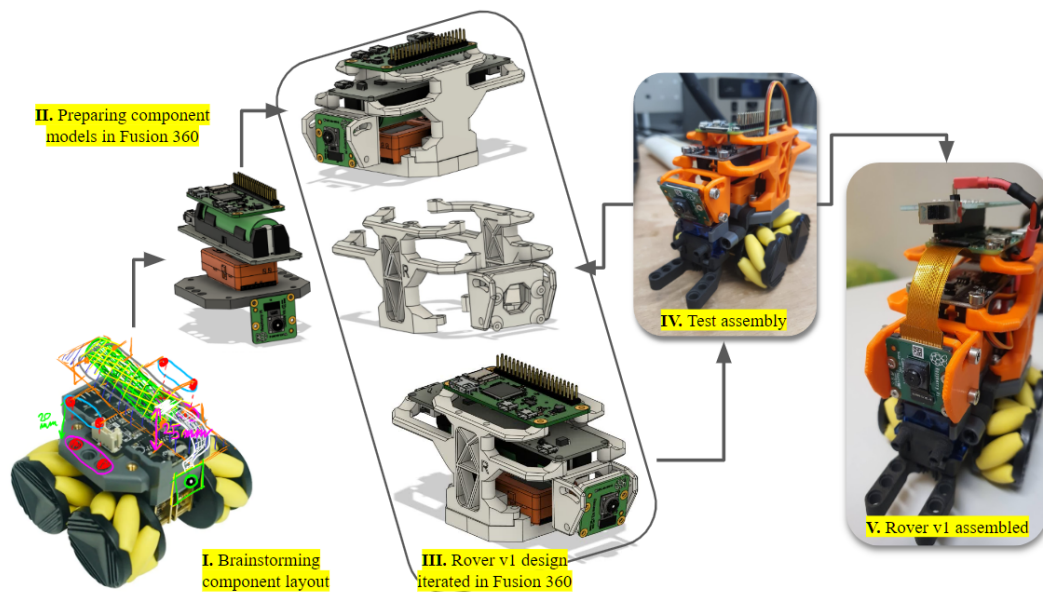

Figure 3.4: Design thinking and iterative rapid prototyping approach. In developing Rover v, the first phase features component layout brainstorming. The second acquires model files, followed by a reassessment of the layout. The third phase adds rough component designs. Various test prints and assemblies gave insight into design issues.

Autodesk Fusion 360 [56] proved immensely powerful and significantly eased the design process of the gantry, also called the *'adapter mod kit'*. Firstly, individual component model files had to be prepared or sourced to begin the design. Public GrabCAD models or manufacturer-provided specifications helped find or prepare appropriate model files whenever possible, in addition to direct physical dimension measurements for confirmation.

The gantry's design was gradual, following the initially sketched component layout. A secure mounting solution was possible using the provided M3 threaded inserts on the RoverC Pro's chassis. The gantry was thus constructed from this point onwards, sequentially fitting and joining components near the v0 prescribed locations. The design of an adjustable camera mounting solution followed this, which conveniently reuses the 18650 PSU mounting holes. The RPi Camera Module v2.1 is connected using a CSI flexible connector cable, and this cable was intentionally routed in between the RPi Zero 1 W and the 18650 PSU circuit to reduce the risk of entanglement or damage to the cable or its fragile connectors.

This design stage's main focus was preventing part interference and ensuring all dimensions were correct. After 3D printing and assembling the prototype gantry, its issues were logged, including mismatches and misalignments to adjust. By applying the design changes, printing and then

reassembling the gantry to the base, Rover v1 was ready. In testing the driving performance, an unexpected handling improvement was observed compared to the stock Rover base.

Rover v1, as a prototype, helped develop and test the video streaming solution. The 5V 4A access points of the 18650 PSU are not switchable by the board's main switch. Adding an inline switch improved the ease of use when powering the RPi on or off. Figure 3.5 below shows this breakout board in its final configuration, top right, and the final design iteration result, named 'RoverXR', on the left, discussed below.
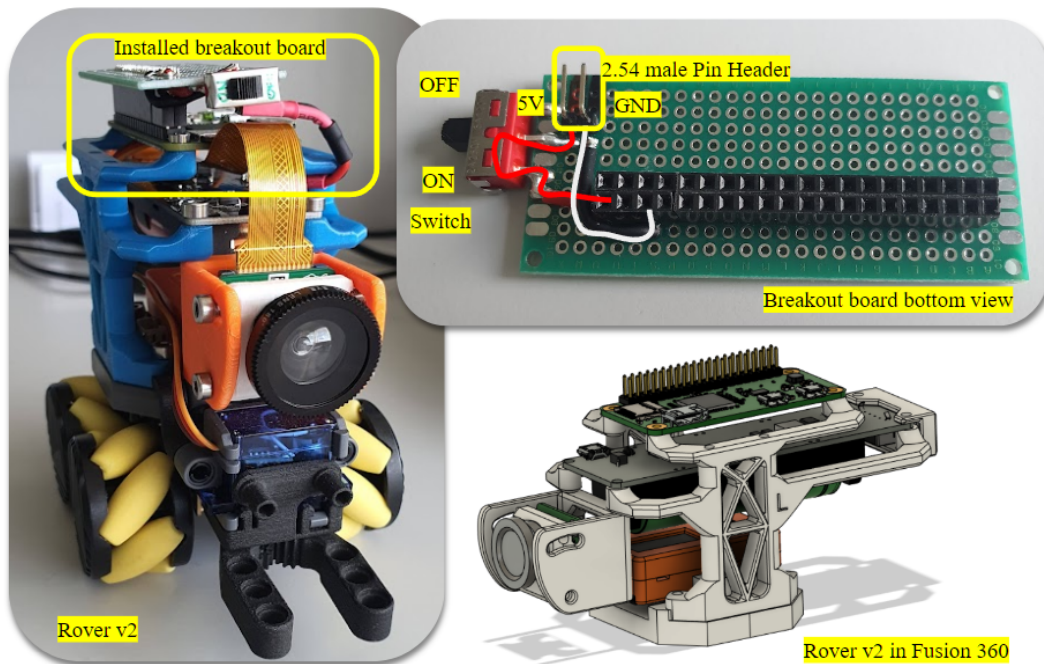

Figure 3.5: Assembled RoverXR, its CAD model and breakout board. The breakout board (top right) features a switch and 2.54 male pin header pins connected to the 18650 PSU. RoverXR's model (bottom left) incorporates many lessons learned from Rover v1, such as elevating the gantry, adding a rear bumper and a new lens mount.

RoverXR represents a relatively small comparable update to v1. It adds a new protective rear bumper and a heavily modified camera mount. The new mount's careful design accommodates a magnetic wide-angle smartphone lens kit. This kit combines a macro and wide-angle lens and a fisheye lens. Ease of access to the M5 Stick C was also significantly improved by increasing the clearance to the 18650 PSU circuit, thus allowing the M5 Stick C to be removed from the assembly more easily.

Aesthetics, particularly the colour palette, were not a priority, apart from ensuring the internal components were discernible without disassembly. Students may visually inspect the internals to understand how RoverXR works. When asked, several persons described Rover v1's appearance as *'construction-like'*, pointing out the overuse of orange-coloured PLA filament. RoverXR's colour palette should have been orange as well. However, some of the printed parts had small deformations. Thus a backup set of test prints was used to finalise the gantry assembly, as seen in blue highlights in Figure 3.5 above.

3.1.3 Bill of Materials

Table 2 below lists RoverXR's components and their descriptions, features and costs. Note, this does not include additional costs of assorted screws, printing filament, Meta Quest 2 VR HMD or appropriate OpenWRT [57] router, or other infrastructure. Please, find an extended version of the bill of materials in Appendix I, with additional component key features presented.

Table 2: RoverXR Bill of Materials.

Component	Description	Cost [EUR]
M5 Rover C Pro	Programmable Mecanum wheel omnidirectional robot base with N20 worm gear wheel motors and servo gripping mechanism.	60
Wemos Battery Shield	18650 Battery Shield (V3) for Raspberry Pi & Arduino Affordable, portable power supply module.	2.5
18650 Li-Ion	Battery: Samsung Li-Ion 18650 cell	11
M5StickC ESP32	M5 Stick C is a mini M5 Stack, powered by ESP32. It is a portable, easy-to-use, open-source IoT development board.	27.4
JoyC Omni-directional Controller	Movement control extension for the M5 Stick C & Rover base.	18.2
Raspberry Pi Zero 1 W	Flexible, compact Raspberry Pi Single Board Computer. Size: 65mm long by 30mm wide, affordable.	13.7
RPi Camera Module v2.1	High-quality camera based on Sony IMX219PQ image sensor. Supports FHD video and still photographs.	23.65
CSI Camera Adapter	Raspberry Pi Zero adapter cable to Camera CSI connector	5.12
0.67x Wide, Macro, Fisheye Lens kit	Magnetic interlocking lens camera add-on kit	7
Totals	**Listed Prices - April 2023**	**168.57**

Overall, RoverXR's design succeeds in demonstrating a straightforward but feature-rich teleoperation solution in a more compact and portable form factor than previously accomplished. The projected aggregate cost for manufacturing the Rover alone is in the sub 200 EUR range, after including the sum costs of assorted screws and filament. Finally, assembly time was measured while assembling RoverXR, taking approximately 20 minutes from start to finish.

Thanks to the iterative design approach, ample testing was conducted to improve the success rate of the 3D printing process. Depending on the Z layer height, printing the gantry took two to four hours. In conclusion, if all components are at hand, it should be possible to fully prepare a teleoperation demonstration using RoverXR within 5 hours, which includes software aspects discussed in the next section.

3.2 Software

RoverXR's software component is responsible for movement, encoding and transmitting real-time video from the RPi, decoding and rendering the received video stream on the remote display. The software solution has two branches:

1. Embedded software running on the M5 Stick C microcontrollers for movement control.
2. An RPi-based streaming solution with an accompanying Godot Engine 4 scene featuring custom scripts for displaying the received, streamed images.

Both are critical in ensuring that RoverXR functions as intended, and each presents unique challenges and opportunities. Figure 3.6 illustrates these branches and their properties, such as communication protocols.
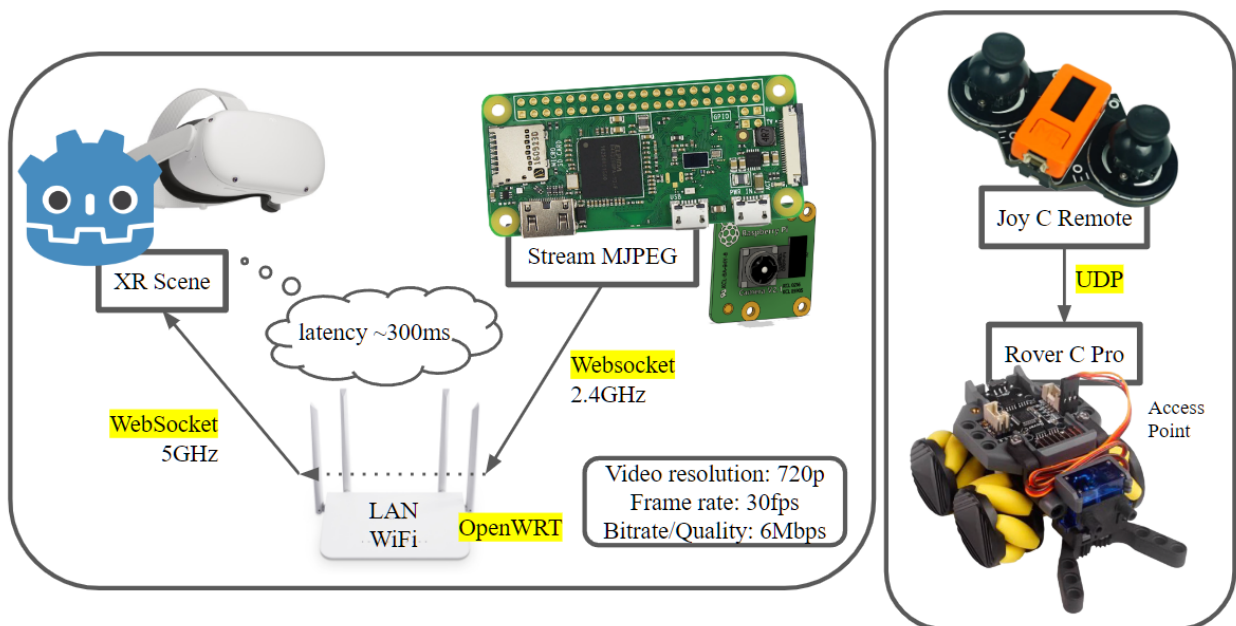

Figure 3.6: RoverXR's two software branches. Rover C Pro [17] and Joy C Remote [58] facilitate movement control. The RPi [16] and its Camera [50] stream 720p 30fps MJPEG video over WebSocket at the 2.4GHz band via an OpenWRT LAN router [59] with ~300ms latency to a Godot scene on Meta Quest 2 VR HMD [18].

This section provides an overview and descriptions of these branches' roles and functions. By the end of this section, the reader will have a good understanding of the software side of the Rover.

3.2.1 Network Architecture

Figure 3.6 shows RoverXR's network architecture, a vital component of any teleoperation solution. The M5 RoverC Pro and JoyC Remote communicate using a separate UDP channel on their independent access point network. As a result, movement control is fast, crisp and responsive. Meanwhile, the RPi streams video over a WebSocket connection, chosen for three main reasons:

1. Godot Engine provides good WebSockets documentation [60]
2. Previous works successfully demonstrate video streaming using WebSockets [6]
3. WebSockets can reduce code complexity and improve performance compared to HTTP or TCP in low-latency applications [40], [61]–[67].

For managing RoverXR's network traffic, most commercial network routers are acceptable. A smartphone hotspot provided convenience, portability and good performance during RoverXR's testing and development. However, the openWRT router [59] used during the evaluation facilitated the best performance by providing both 2.4GHz and 5GHz bands for communication (Figure 3.6). The RPi streamed video to the Godot scene on the VR HMD. A PC laptop and smartphone frequently provided troubleshooting access without grossly impeding network bandwidth.

3.2.2 Raspberry Pi Video Streaming

To develop a high-quality, low-latency video streaming solution, several solutions were investigated, listed in Table 3. They provided examples of good practices and contributed towards a good base subject area understanding for developing RoverXR's real-time visual feedback system.

Table 3: Video streaming solutions investigated.

Solution	Description	Comments
Dietpi [68] with MotionEye [69]	Open source camera surveillance solution.	Video stream quality was poor due to data storage efficiency priorities.
DroidCam OBS [70]	A plugin and Android app which helps turn a smartphone into a web-connected, low-latency video web PC camera.	Outstanding performance; chosen as a reference benchmark for comparison. However, it only supports smartphones.
picamera [71]	Legacy Raspberry proprietary camera library with excellent documentation.	Great performance but requires legacy camera support enabled with raspi-config. Supports only Raspberry Pi camera modules. Provides ready-made video streaming examples.
picamera2 [72]	Libcamera-based library for the Raspberry Pi, with excellent documentation.	Same performance as picamera, but it supports more camera modules. Provides more advanced examples, including several video stream servers.
raspivid [73]	A command line tool for capturing video with a Raspberry Pi camera module.	Legacy camera based with integrated TCP and UDP streaming solutions. The least compute intense on the RPi Zero 1 W for video streaming.
openHD [74]	Suite for long-range video transmission, telemetry and control.	Both require additional components and very specialised solutions for FPV drones.
EZ-WiFiBroadcaster [75]	FPV-like low latency digital data transmission solution.	

DroidCam OBS performed exceptionally well and was a benchmark for comparing latency performance with other solutions, despite only supporting smartphones. On the RPi, raspivid was

the most compute-efficient video streaming solution, at most placing a ~15% load on the CPU. However, raspivid relies on legacy camera drivers. Despite natively providing TCP and UDP streaming utilities, raspivid was less flexible or feature-rich than the slightly CPU-load-unfriendly picamera2 library, which also ships with excellent documentation. Ultimately, picamera2 was chosen for its ease of use and flexibility.

Aside from video streaming solutions, stream playback was also in focus. Figure 3.7 summarises the testing process results, providing glass-to-glass latency measurements in milliseconds (lower is better) across various video stream sources & sinks, protocols, video formats and configuration combinations.
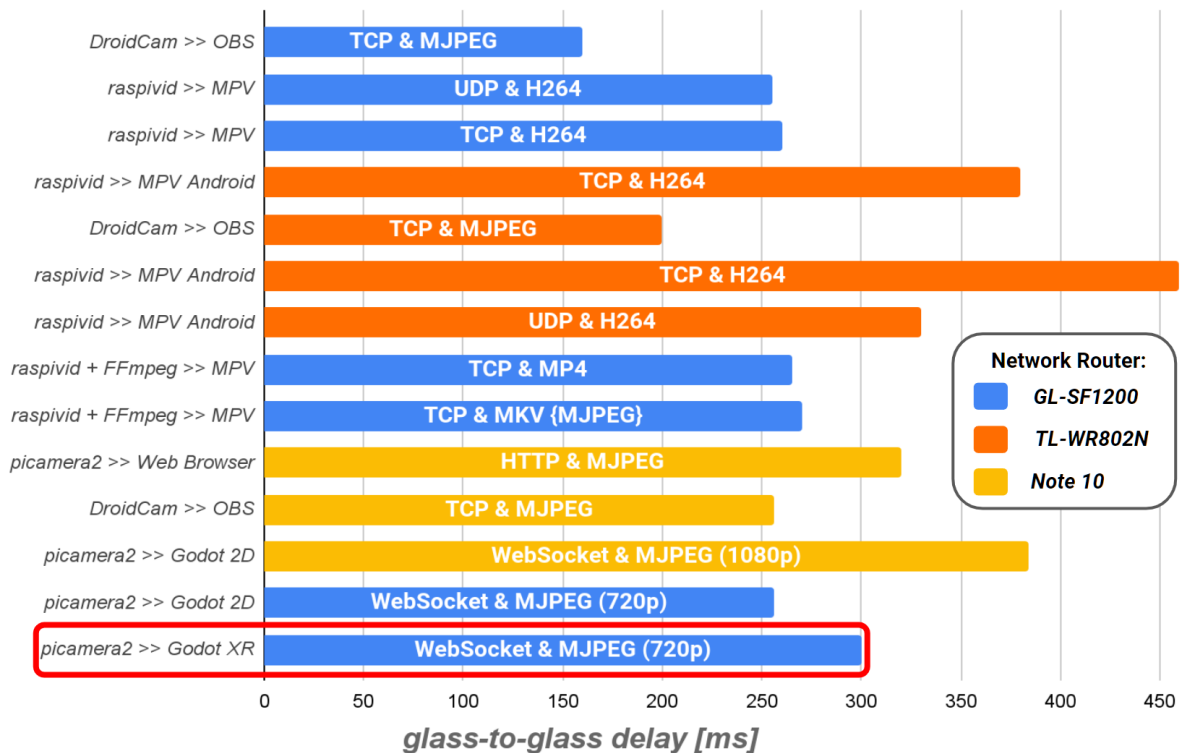

Figure 3.7: Video stream testing latency comparison. Vertical axis lists video stream sources & sinks. Labels show communication protocol and video format info per test. Glass-to-glass delay (lower is better) depends on video stream configuration combination. Red-circled configuration from the Godot AR scene measured ~300ms.

The latency data were collected leveraging the Droste effect [76], a recursive picture in which an image appears within itself in a way that creates an infinite loop.

By looking at the recursive stopwatch images and comparing the displayed time (in milliseconds) of several stopwatch image recursions, it is possible to estimate the glass-to-glass latency (Figure 3.8). However, this method is unreliable due to a high chance of misreading the time values.

Furthermore, single-shot measurements were used exclusively in collecting the latency data of various video streaming solutions. The results should not be regarded as conclusive because of the lack of average values and standard deviations. Regardless, this approach provided a fast and straightforward solution for gauging the latency of a given video streaming configuration.
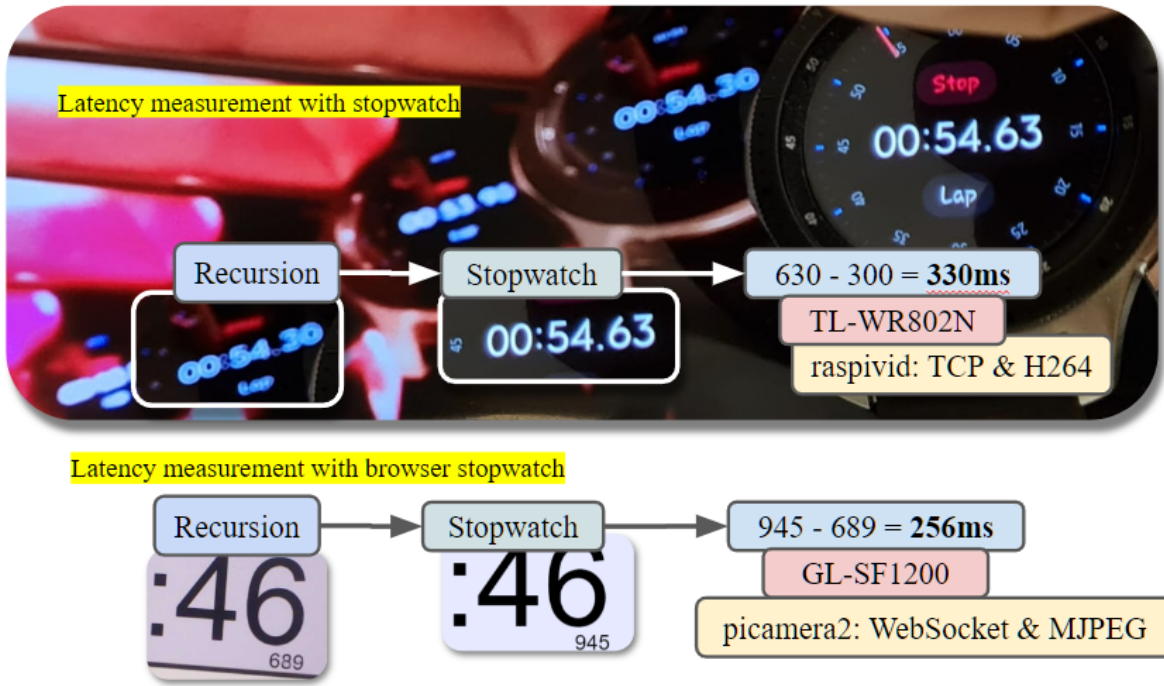
Figure 3.8: Latency measurement examples. Handheld stopwatch (top) and browser stopwatch (bottom). Leveraging the Droste effect and reading the millisecond dial values in recursive frames, glass-to-glass delay can be estimated.

In testing, FFmpeg [77] and MPV media player [78] were explored for video streaming playback. FFmpeg is a free, open-source software project comprising libraries and programs for handling video, audio, and other multimedia files and streams. At its core, the command-line FFmpeg tool is designed to process video and audio files.

MPV player a free, open source, and cross-platform performance command-line media player, supporting a wide range of formats, codecs and platforms, including Linux, Windows, Mac and Android. MPVs versatility allowed for multiplatform stream playback testing and latency measurements. In particular, various raspivid arguments were tested with MPV for remote video playback using TCP and UDP video streaming. Please see Appendix I. (Thesis GitHub Repository: Blueprints/Testing) for a list of tested MPV, raspivid and FFmpeg command line arguments.

When deployed to the Quest 2, MPV (Android) provides a simple solution for video stream playback from VR. This alone provided well-performing and high-definition direct visual feedback when testing teleoperation with RoverXR. However, with a user interface intended for use with smartphones, operating MPV from VR proved time-consuming and exceedingly user-unfriendly, requiring several minutes to configure. While it is possible to devise an autostart solution, which MPV supports, ultimately, this solution would not be scalable - it would be challenging to add new features.

A Godot standalone environment was thus requisitioned, described in 3.2.3, to provide a more positive user experience and facilitate a more robust user interface development framework, in line with design requirements described in section 2.3. However, this came with new limitations as Godot supported Theora video compression format [79], [80] exclusively at the time of

writing. Circumventing this issue required either reencoding the video stream to Theora format on the RPi or, somehow, in Godot or adopting an alternative strategy altogether.

Every computational step would lead to added latency. However, in testing, reencoding to Theora from H264 using FFmpeg would also fully saturate the RPi Zero 1 W CPU. Thus, another approach was necessary. One possible solution was to manually decode the received video stream in Godot or, alternatively, send individual video frames and display them directly.

The latter approach worked out better, and thus Motion-JPEG [81] was chosen as the video format because it was the easiest method to work with and understand, in line with the design requirements outlined in section 2.3. An MJPEG extension video file is created by compressing individual video frames as a JPEG image in a sequence [82]. JPEG files (compressed images) start with an image marker containing the marker code hex values: *'FF D8 FF'*. JPEG files do not include embedded file size information in the file header. However, JPEG files also feature an ending hex value [83]: *'FF D9'*. Thus it is possible to identify individual video frames in an MJPEG video stream by looking for these two hex values and outputting the information between both.

In order to access the video feed, picamera and picamera2 libraries were compared. Picamera [71] is RPi's legacy pure Python interface to the RPi camera modules, with a BSD license. Like raspivid, picamera requires a user to enable legacy camera support using the raspi-config tool manually. Additionally, like raspivid, picamera exclusively supports RPi camera modules and, as legacy software, will not receive updates in the future.

Meanwhile, picamera2 [84], the successor of picamera, does not require legacy camera support enabled in raspi-config. The picamera2 library is based on libcamera [85], an open-source camera and framework for Linux, Android, and ChromeOS, featuring advanced functions and supporting many off-the-shelf USB cameras. As a result, picamera2 packs more features than picamera, such as built-in face detection and object recognition, and offers better hardware encoding support. Picamera2 also delivers a better image aesthetic and extends libcamera's list of supported cameras with RPi camera modules.

Based on the design requirements outlined in 2.3, priority was given to a good user experience. As a result, picamera2 was identified as the more suitable camera library as it does not require additional RPi configuration steps. Additionally, the more extensive list of supported devices, libcamera framework foundation and promise of future support provided added appeal.

The final MJPEG WebSocket server, *'stream_mjpeg_ws.py'*, written in Python3, relies on several libraries and their classes, including io, threading, websockets, asyncio, picamera2 and libcamera. It comprises four components: the main server function, a video stream handler function, camera configuration and a video stream buffer. The developed MJPEG WebSocket servers' camera and frame buffer solution is based on the example solution provided by the picamera2 library *'picamera2/examples/mjpeg_server_2.py'* [72]. Please, see Appendix I. (Thesis GitHub Repository: *Blueprints/stream_mjpeg_ws.py*) to learn more about the WebSocket video server.

The camera is configured once the script is run, followed by the video buffer, using io's BufferedIOBase class. Finally, a server is configured and started. Once a client connects on the appropriate port, individual video frames are sent over the connection when the memory buffer collects a new video frame and notifies the stream server thread using threading's Condition class. While the described example uses an HTTP server, the approach was directly transferable to WebSockets.

The developed 2D video stream testing Godot scene, described in 3.2.3 (Figure 3.10), can consume a full 1920x1080 px (1080p) video stream at 30 fps. This scene was also used to measure latency (Figure 3.7: *Godot 2D*). However, in testing, the AR Godot scene, described in 3.2.3, could only handle up to 1280x720 px (720p) resolution video streams before suffering noticeable performance issues, such as severely reduced frame rates and nausea-inducing motion tracking, rendering the user experience unacceptable.

Due to the observed computational limitations of the Meta Quest 2 and Godot AR scene, the WebSocket MJPEG video stream was limited to 720p. This translates to 44% fewer pixels rendered per frame than a 1080p video resolution. As a result, the latency measured at ~256 ms was also ~62.5 % slower than the Droidcam reference benchmark at ~150 ms.

$$bitrate\ [Mbps]\ =\ bitrate_{ref}\ \times\ \sqrt{\frac{frame_{width} \times frame_{height} \times frame_{rate}}{frame_{width,ref} \times frame_{height,ref} \times frame_{rate,ref}}} \qquad (3.1)$$

The picamera2 library supports manually setting various video quality preset settings, which directly impact compute performance and uses more network bandwidth. Its MJPEG encoder source code [86] calculates the video stream bitrate according to equation (3.1). A set of reference bitrates ($bitrate_{ref}$) per quality preset is also provided, shown in Table 4, for a reference frame width ($frame_{width,ref}$) of 1920 pixels, reference frame height ($frame_{height,ref}$) of 1080 pixels and reference frame rate ($frame_{rate,ref}$) of 30 fps.

Table 4: Picamera2 reference bitrate at 1080p 30 fps in Mbps per JPEG quality preset.

JPEG Quality Preset	Mbps
Very Low	6
Low	12
Medium	18
High	27
Very High	45

At a very low-quality MJPEG encoding and 720p resolution at 30 fps, the configured video stream bitrate consumes 4Mbps or 4 million bits per second of compressed video frames. During the evaluation discussed in Chapter 4, participants, despite this, noted that the video quality was clear, and they could see where they were going.

Finally, to further improve the user experience, I prepared an installation routine with the supervisor's support, configuring an RPi to commence video streaming after booting up. With this autostart feature, if correctly configured, RoverXR goes from power-on to streaming within 5 minutes.

In conclusion, several video streaming and playback solutions were compared and deployed to gauge various setups latencies (Figure 3.7). The MPV-based concept solution left much to be desired and requisitioned a Godot environment, which only supports the Theora video format. The solution was to configure the RPi WebSocket MJPEG server based on the picamera2 library, streaming 720p 30fps video at a very low JPEG quality, with a 4Mbps bitrate, due to Meta Quest 2 and Godot compute limitations.

3.2.3 Godot Engine 4 Integration

Godot Engine [22] is a free and open-source 2D and 3D game engine with native OpenXR [87] support since the release of version 4, making it easy to develop games for various supported VR headsets. This made it the ideal candidate, in line with the outlined design requirements discussed in 2.3.

In preparing RoverXR's user interface, 2D and 3D scenes were developed and deployed to the Meta Quest 2 VR HMD. To export project scenes to the headset, Android Tools and Gradle are used to compile the app apk file, and Android Debug Bridge (ADB) is employed to install the compiled app scenes. All of these utilities are neatly packaged together with Godot.

GDScript [88], [89] is a dynamically typed scripting language explicitly made for Godot. Its syntax is similar to Python's, and its main advantages are ease of use and tight integration with the engine. A good understanding of GDScript proved to be essential throughout the development process. It was used to develop the MJPEG WebSocket client and video stream playback in AR and 2D scenes, described below.

Godot presents all the essential features necessary to build a VR or AR project out of the box. However, specific game mechanics must be implemented on this foundation. While Godot makes this relatively easy, this can be a daunting task. To streamline and extend Godot's VR development capabilities, the team behind Godot OpenXR has developed a toolkit called Godot XR Tools [90] that implements many of the basic mechanics in typical AR/VR games. This includes basic locomotion, object and user interface interactions. Godot XR Tools were designed to support OpenXR and WebXR [91] API standards.
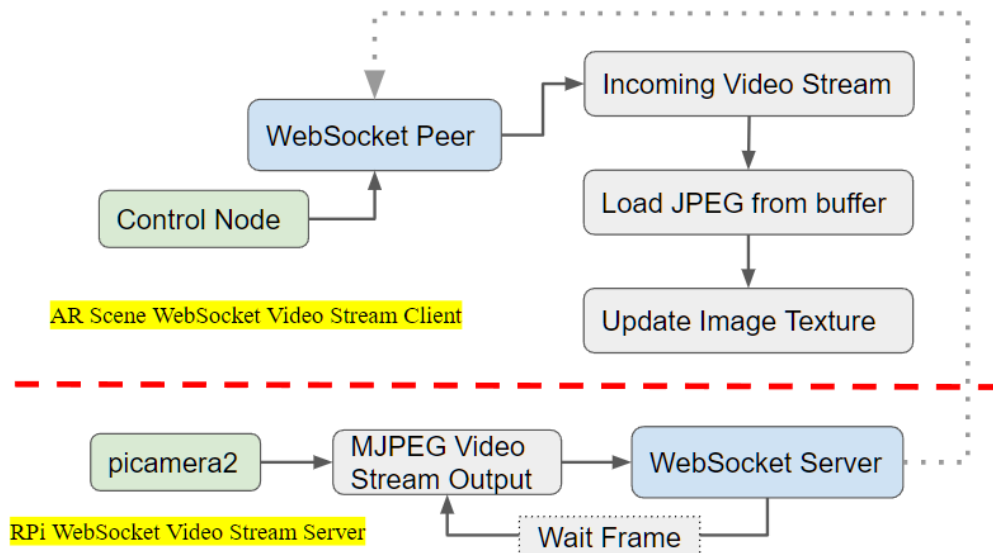
Figure 3.9: WebSocket server-client video stream process diagram. Godot scene (top) connects to the RPi server (bottom), reads incoming packets to JPEG and updates a texture. RPi serves individual picamera2 video frames.

The developed GDScript WebSocket client works as follows.

On load, the Pi2AR project scene tries to connect to the RPi video stream server over a WebSocket connection. On the RPi, the streaming server collects video frames using the picamera2 library [84], based on libcamera [85], to a memory buffer until the hosted WebSocket server can send them to the connected client. Figure 3.9 illustrates the operations responsible for transmitting individual frames from the server and receiving and displaying them in Godot.

In Godot, the WebSocket connection is polled during each game tick, and inbound frames are stored in a local scene buffer until retrieved using the *'load_jpg_from_buffer'* command. An image texture is created using the loaded video frame to update either a MeshInstance3D or a TextureRect2D node's texture.
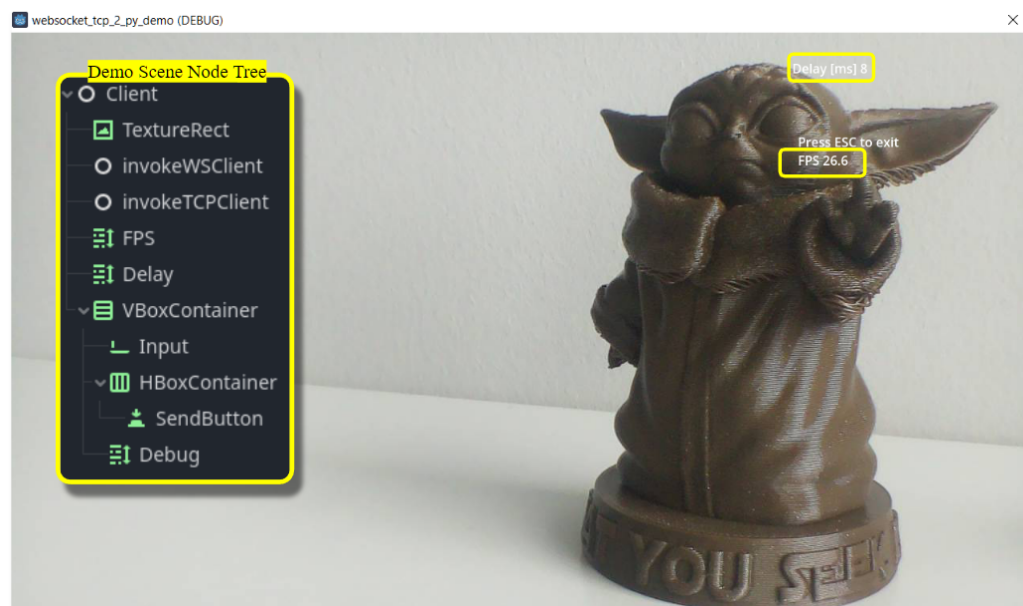

Figure 3.10: 2D 720p video stream scene with a Grogu figurine [92]. To the right is the local render delay and video frame rate, and to the left is the associated node tree.

To learn more hands-on, this 2D 'demo' scene was prepared to be run on a PC in addition to the main AR scene. This scene was used to study, develop and test various video playback features more seamlessly. Figure 3.10 shows the demo scene's node tree to the left consists of a TextureRect node, used to display the received video frames and several Text Label nodes, which display connection debug information, such as if the client is connected or reported connection errors. Once the WebSocket handling script was ready, most of these TextLabel nodes were hidden. Key features of video stream characteristics, such as framerate, connection stability and pixel-to-pixel delay, were tested using this 2D scene. It also served to validate possible TCP, UDP and alternative WebSocket GDScript implementations.

To develop the final scene, titled 'Pi2AR', Godot version 4.0.2 was used. This final project scene was more complex and included both AR and VR environments. However, the AR scene was ultimately selected, as the pass-through functionality gives the user a better awareness of the surrounding environment. This may improve the overall user comfort in possibly crowded open lab days.

Figure 3.11: AR scene with a view from the player. The background stays black as pass-through is not available on PC. Highlighted cubes were added to the scene to exemplify how the scene can be made more immersive and interactive. Featured in the centre is a 16 by 9 m video playback screen.

Figure 3.11 showcases the 'Pi2AR' scene from the user's view, run on a PC. Two solutions for displaying the received video frames were prepared. Their node trees are shown on the right. The first incorporates a Sprite3D node, where a transparent texture of a MeshInstance3D node is redrawn with each received video frame. Similarly, a TextureRect node's texture is redrawn in the second node tree with each received video frame.

The difference between the two approaches is that the Sprite3D scene can be directly incorporated within the AR scene. Meanwhile, the TextureRect approach requires using a Viewport2Din3D sub-scene from the godot-xr-tools addon pack. Both approaches produce the same result, and no performance differences were observed in testing. However, the latter is used in the final Pi2AR project scene. The Viewport2Din3D sub-scene provides extended functionality, which includes advanced user interaction features. While these were not utilised, it was included to provide an example use-case.
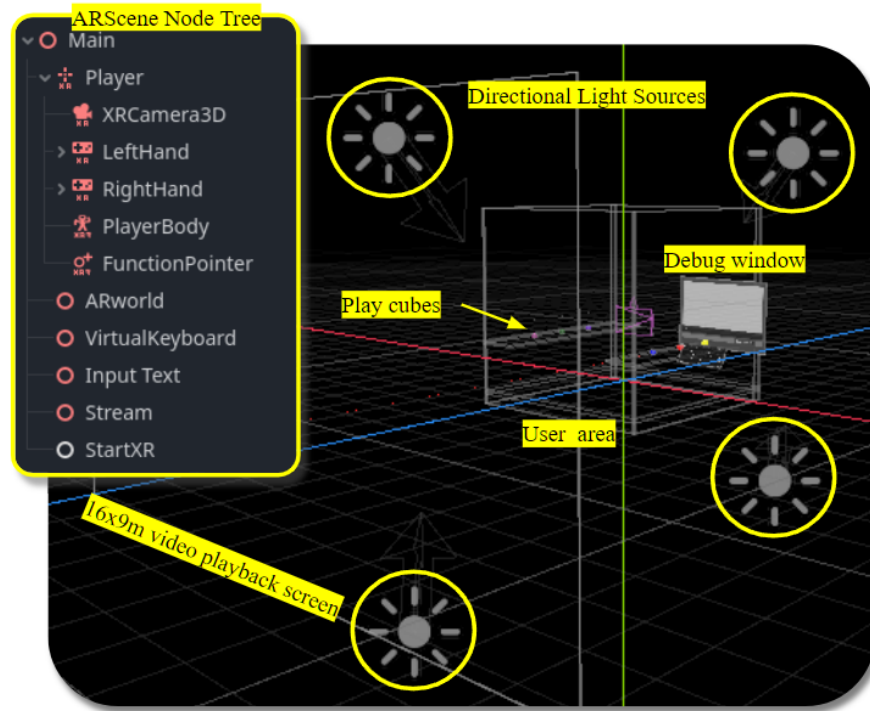
Figure 3.12: Annotated AR scene. The video stream playback screen is placed front and centre, with four directional light sources providing lighting to the user area box with play cubes and a debug terminal opposite the primary video stream screen. Transparent collision boxes confine the user to the moveable area.

The black background environment in the AR scene (Figure 3.12) is configured to be transparent in order to be able to display pass-through video, which in the case of the Quest 2, is black & white. The moveable area allows users to adjust their view angle to the primary floating screen. By moving around, they can also zoom and pan the video stream.
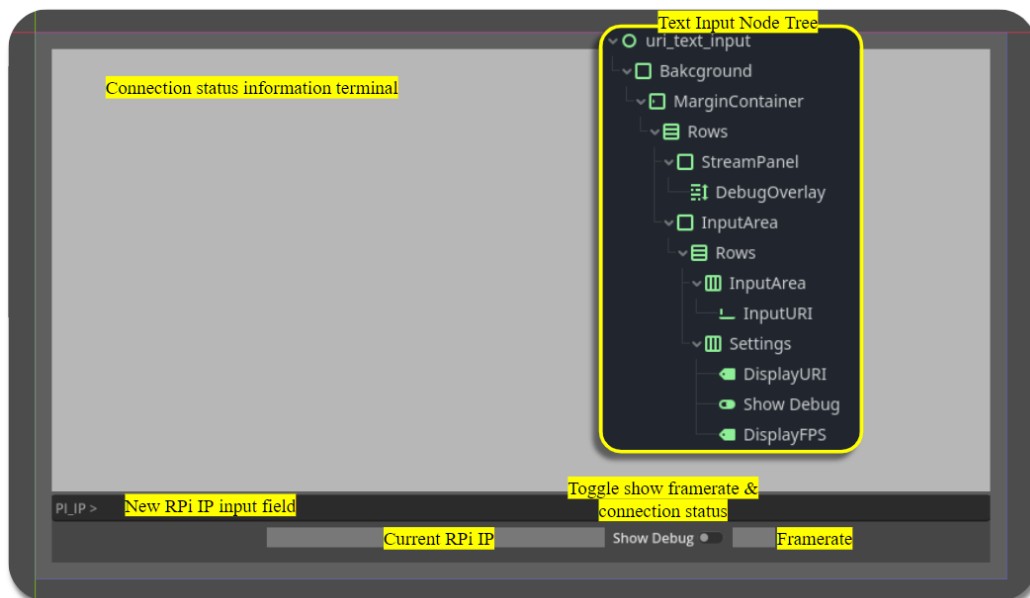

Figure 3.13: Closeup of AR scene's debug panel. Positioned behind the user, it provides information on the connection status and framerate. It includes an input field if a user wishes to change or set a new RPi IP address and two text labels showing the current IP address and framerate. The 'Show Debug' button toggles the display of information.

The added debug panel (Figure 3.13) was included to provide useful information when troubleshooting connection issues from within the AR scene during runtime. It allows a user to manually adjust the RPi's IP address and see relevant connection status information and the framerate of the displayed video stream. Below the debug panel, a virtual keyboard (part of the godot-xr-tools addon) is provided, enabling a user to input the RPi's IP address once the relevant text input box is set to active.

Please, see Appendix I. (Thesis GitHub Repository: Blueprints/Godot, Blueprints/Testing/websocket_2_py_demo) for more information about the Test 2D scene and Pi2AR (Blueprints/Godot/Pi2AR).

3.2.4 M5 RoverC Pro & JoyC Remote Controller Movement Control

The onboard M5 Stick C provides movement control. When paired to a second M5 Stick C in the JoyC remote controller, it conveys speed commands to RoverC's onboard microcontroller (STM32F030C6T6). This way, the JoyC remote controls four independently motorised omnidirectional Mecanum wheels with N20 worm gear motors. Fine, omnidirectional movement control is achieved by carefully adjusting each wheel's rotation speed and direction. Figure 3.14 showcases RoverC Pro manufacturer's provided individual wheel-to-overall-movement control diagram.
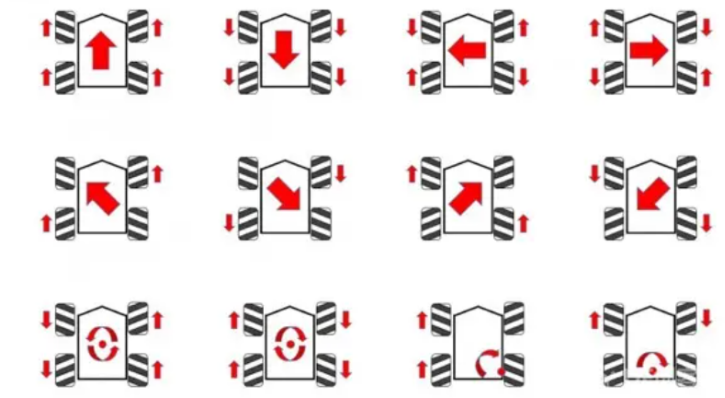

Figure 3.14: Wheel-to-overall-movement diagram [17]. Shown are 12 RoverC outlines with four Mecanum wheels. Carefully adjusting the individual wheel's rotation speed and direction is necessary for omnidirectional movement control. N20 worm gear motors, one for each wheel, are carefully regulated by the RoverC's onboard microcontroller, which receives speed commands over I2C from the M5 StickC.

The RoverC and JoyC remote M5 StickC Arduino libraries incorporate a fully featured movement control example. The only functional alteration to the manufacturer's provided example firmware was to incorporate servo angle control. The RoverC's firmware script first initialises library drivers and creates a Wi-Fi hotspot. It then displays the hotspot name on the device's LCD screen. It also sets the charging current, configures the device to act as a Wi-Fi access point, and starts a server for communication. The script then listens for UDP packets, reads and processes the received data, and controls the device's speed and servo angles accordingly. Finally, the LCD screen is updated periodically with battery voltage, current, and the status of the I2C communication.

JoyC's remote controller firmware first initialises hardware components. After a user selects the RoverC's access point hotspot, it establishes a WiFi UDP connection. It then reads thumbstick sensor data and forwards it to the Rover. Pressing on the thumbsticks actuates the servos. The left thumbstick controls the translation, while the right controls the rotation. Please, see Appendix I. (Thesis GitHub Repository: Blueprints/Firmware) for more information about movement control firmware and a video movement demo taken during the participant evaluation (Thesisaurus/Survey/evaluation_event).

3.3 Limitations & Challenges

While developing RoverXR, many limiting and challenging problems had to be solved. Lacking documentation was one challenge for servo control in the M5 RoverC Pro, requiring extensive experimentation and testing.

Additionally, the filament supply condition limited the quality of 3D prints produced in-house, which led to some mechanical issues during assembly. Finding suitable 3D models for some components was also a challenge. Assembly of the rover also presented some challenges initially, as the components are small and easy to misplace or damage.

The rover's layout proved challenging due to the need to accommodate many components in a very small space. There were also concerns about driving performance, especially over rough terrain, and some components' durability and wear resistance.

Working with the RPi Zero 1 W and developing the video streaming into the VR environment proved exceedingly challenging, primarily because of its single-core processor, limited memory and the fact that it was impossible to develop remotely from within the comfort of a powerful IDE like Visual Studio Code. Above all, video streaming development was challenging due to Godot's limitation, supporting the Theora video format exclusively.

Throughout the development process, Godot was transitioning from version 3 to version 4, which led to difficulties finding up-to-date documentation, with several features only partially available. Godot's official Discord community was consistently supportive. They provided guidance on key issues, such as troubleshooting Gradle deployment to the Quest 2, and key insights into features such as WebSocketPeer class and the OpenXR Tools add-on.

The cost of upgrading to the more powerful RPi Zero 2 W was also prohibitive, preventing a more straightforward RoverXR design, which seamlessly controls the movement from a VR scene.

These limitations and challenges combined contributed to the resulting final product of this thesis, a complex multi-part teleoperation demonstration solution. Many tradeoffs had to be made between framerate, bandwidth, image quality, and various design considerations.

3.3.1 Design & Development Summary

A teleoperation educational demonstration is prepared, incorporating:
1. M5 Rover C Pro and JoyC remote controller
2. custom-designed, 3D printed adapter kit (mod kit)
3. RPi Zero 1 W and Pi camera module
4. Godot AR & 2D scenes

The RPi is configured to serve an MJPEG video stream using the WebSocket protocol to which a counterpart Godot Engine 4 scene connects to and collects video stream bytes. These are decoded into JPEG images individually and displayed in the scene.

Key milestones in the design of the artefact include:
1. selecting suitable components,
2. identifying appropriate streaming solutions,
3. designing the adapter kit,
4. developing video streaming solutions and
5. developing AR & 2D Godot scenes.

During the second milestone, three hackathon-like sprints were conducted under the guidance of my supervisor. Jai Muruganantham, an MSc student at Kaunas University of Technology in Lithuania, joined these sprint sessions remotely. We exchanged solutions for handling, troubleshooting and evaluating video streaming, including using the Droste effect for testing pixel-to-pixel delay using a stopwatch. Together we learned about the intricacies of FFmpeg, an incredibly vast and powerful open-source project for handling a myriad of video, audio, and other multimedia files and streams [77].

4 Evaluation

The success of any solution ultimately depends on how well it meets its intended goals and how its users perceive it. This chapter presents RoverXR's subjective evaluation, assessing how participants perceived RoverXR and its AR interface as a teleoperation demonstration.

Based on eight questions with various response options, including multiple-choice, rating scales, and open-ended text answers, the primary goal of this evaluation was to gain insight into the overall experience of persons using this demo.

4.1 Survey Evaluation

The questions covered aspects such as the user's preference about the overall setups, their level of discomfort, their thoughts on the user interface, and their experience with the Rover. The survey also included questions on the difficulty of driving the Rover and how motivating the experience was for studying teleoperation. Participants were also encouraged to provide any additional comments or feedback. The survey responses were collected and analysed to gain insights into the user's perception of the system.

Please, see Appendix I. (Thesis GitHub Repository: Thesisaurus/Survey/Readme) for an expanded list of questions, including question descriptions and elaborated rationale. Table 5 below showcases the list of questions and their accepted response formats.

Table 5: Survey Questions & Response Formats

Question	Accepted response	Description
Have you experienced nausea or discomfort in the setup?	Yes	Provide insight to general comfort in the combined setup, check whether nausea occurs.
	No	
	A little	
How do you feel about the interface in the VR setup?	Short-answer text	Get keywords describing: feelings about the interface, user descriptions of the Rover, user experience using the Rover, thoughts on improvements for the overall setup
Please, describe the Rover	Short-answer text	
Please, describe your experience with the Rover	Short-answer text	
Please, share your thoughts on how to improve the overall setup (Rover & VR)	Short-answer text	
Please, rate the following [Godot setup rating]	1 to 5 rating	Gather simple ratings for overall setup, driving difficulty and how well this experience motivates

Question	Accepted response	Description
Please, rate the following [Rover driving difficulty]	1 to 5 rating	them to study teleoperation.
Please, rate the following [How well does this experience motivate you to study teleoperation?]	1 to 5 rating	
Additional comments or feedback	Short-answer text	This is an optional final question

The evaluation took place on May 9th, 2023, from 3 pm until 7 pm, outside University of Tartu Delta Center, room 2018. A Toy Cityscape track, maintained by Autonomous Driving Lab [93] staff, is set up to test and develop autonomous driving solutions. The track comprises plywood walls in a configuration resembling a small-scale city track (Figure 4.1).
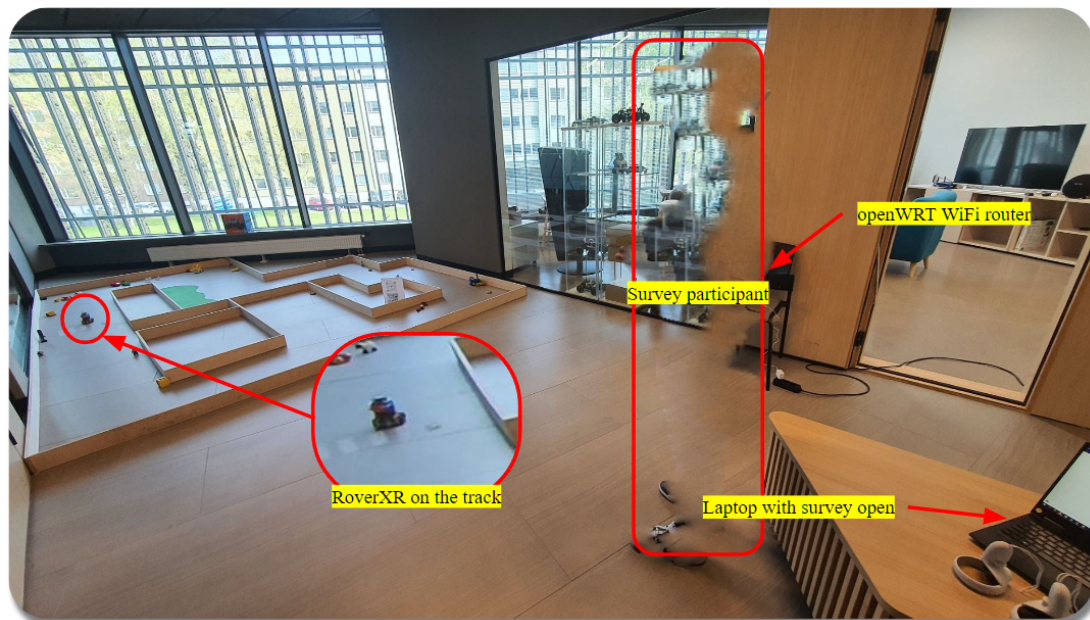
Figure 4.1: User evaluation set-up. Shown top right to bottom left: openWRT WiFi router, blurred survey participant, laptop with survey questions open and to the left, a magnified view of RoverXR on the track.

The floor tiles have a stone-like texture, which provided poor grip for the rover's wheels, resulting in poor handling performance, as reported by the participants. A fully charged RoverXR was prepared and placed on the track, which served the dual purpose of user evaluation and validating standby times. Figure 4.2 shows that the setup included a dedicated openWRT network router. Provided Wifi Analyzer screenshot illustrates network activity at the start of the evaluation.
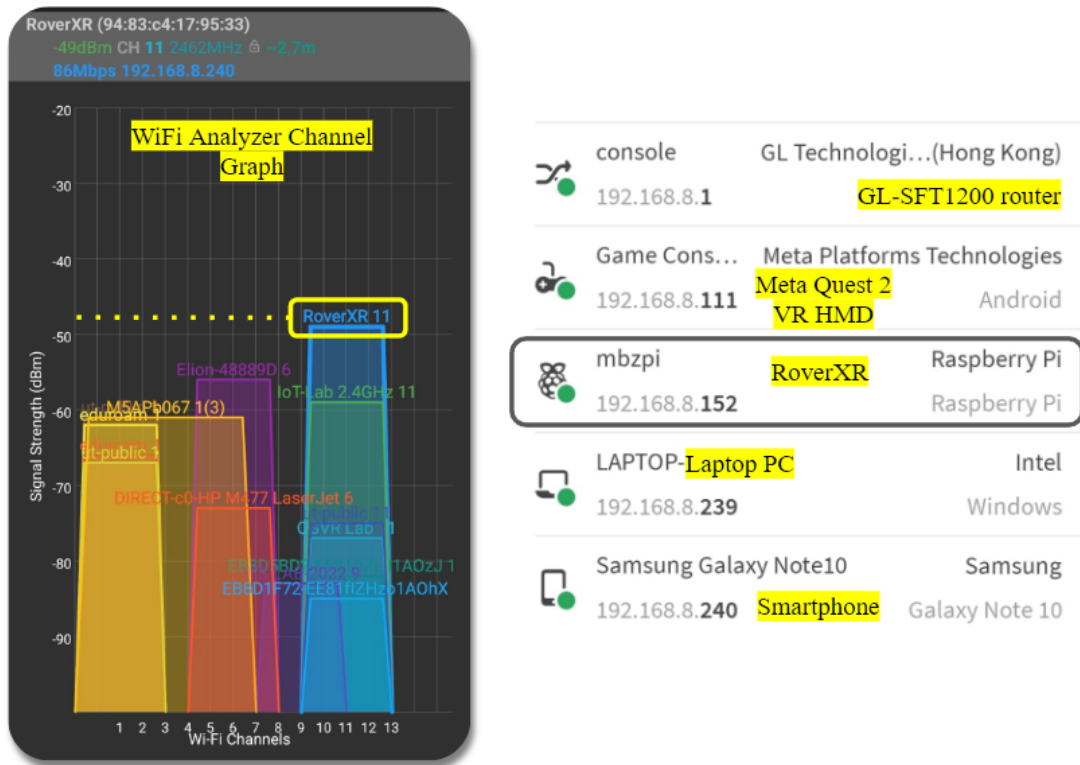
Figure 4.2: WiFi Analyzer [94] and Fing [95] network statistics. During RoverXR's evaluation (left) is a WiFi Channel Graph showing the signal strength of the *'RoverXR 2.4GHz'* network, hosted by the openWRT router. List of devices (right) present on this network at both 2.4GHz and 5GHz bands, including RoverXR's RPi, network gateway router (GL-SFT1200 [59]), Meta Quest 2 VR HMD headset, laptop PC and a smartphone. The video stream is played by inputting the RPi's IP address into the Godot Scene, running on the headset.

The evaluation involved 9 participants, who were first allowed to drive the rover and get accustomed to the controls before being introduced to the AR user interface with a quick showcase of its features. Processing took place individually - one participant at a time.

The participants were given a Meta Quest 2 VR HMD with the 'Pi2AR' Godot scene preloaded and took a moment to adjust the headset for their comfort. They further adjusted the scene to their liking using movement controls within the scene, simulating pan and zoom control of the video feed screen.

Afterwards, participants drove the Rover for one lap. On average, this took about 2 minutes and 25 seconds. Finally, the participants sat down for an informal interview interviewed individually, and their responses were manually input into a Google form for analysis by the interviewer.

4.2.1 Methods & Results

The survey evaluation results indicate that the VR setup was generally comfortable for most participants, with 5/9 reporting that they did not feel nauseous or uncomfortable. In contrast, 4/9 felt uneasy but not uncomfortable, all of which mentioned headset comfort-related issues, likely

associated with poorly adjusted straps, bad headset positioning and incorrectly setting the VR lenses for their particular inter-pupillary distance.

On average, the user interface received a rating of 4.33 out of 5, with 3/9 participants rating it a 5 and 6/9 rating it a 4. This result suggests they overall liked the setup.
However, when it came to driving the rover, participants reported some difficulty, with an average rating of 3.22 out of 5, with 3/9 rating a 4, 5/9 rating a 3 and 1/9 rating a 2.
Despite this, the overall experience was motivating or engaging for most participants, on average rating it a 4 out of 5, with 2/9 rating 5, 5/9 rating a 4 and 2/9 rating the experience a 3.

Two of the answers used to describe the Rover stood out:

'[The Rover] looks really rugged and kind of robust.'

'[The Rover] seems quite well built as far as the material part, it looks a little back top heavy, and going over some obstacles, I can see that it can flip.'

These answers highlight the perceived appearance and build quality of the Rover. The first answer appears superficial. However, it could imply that the look and feel of the Rover made this participant feel more confident about its capabilities.

The second answer touches on the same point and suggests that the top-heavy layout may cause inadvertent flips when traversing obstacles.

These answers stood out when describing the user interface:

'Its okay, I liked the cubes I could throw when I'm bored. I liked that I could see my surroundings. Video [stream] was clear, and I could see where I was going.'

'In the first try, I got lost. In the second try it was good once I understood where to look.'

'Nice, but the problem which I got: the [video] stream had some kind of blurriness, and maybe that made it problematic in [making] driving decisions.'

According to the first participant, the play cubes provide them with an amusing retreat. They appreciate seeing their surroundings, despite the black-and-white pass-through of the Quest 2. Their last point suggests that the video stream is clear.

Based on the second participant's answer, the user interface was not immediately intuitive or easy to understand. They add that they better understood the interface in their second attempt and thought it was good, suggesting that the interface could be improved with additional visual guides to better inform the user of the locations of key features.

The last participant's answer may suggest that their headset's inter-pupillary adjustment slider was not configured correctly, meaning more care and instructions should be provided to users when setting up their headset, especially for first-time users, improving their experience.

Finally, when describing their experience with the overall setup, including the Rover and user interface, these answers were exciting:

'A new experience for me, I haven't tried VR before. I feel that this is very essential when put into real-time applications, maybe some warehouses, or unmanned areas to use vehicles in - when there are no physical people there. I think this has quite a number of applications.'

This participant explains that this was their first VR experience and goes on to identify potential applications for this solution immediately. This answer stands out because they were inspired to rationalise their experience and consider where this application might be useful or applicable.

'It was nice, you are in one place and then controlling in another place. I could quickly imagine myself in the second place and it was an interesting feeling.'

Here the participant describes how well they perceived the remote operation experience. Their answer suggests that the overall experience can successfully create a sense of user immersion and presence.

'Quite very well, it's been a very good experience, it's been long since I've driven anything like a car; this experience is nice.'

This participant expressed that the experience was nice, possibly even satisfying. They seem to connect it to driving a car and appear to appreciate and enjoy it. Based on this answer, the overall experience is positive, and they connected it to the feeling of driving a car.

4.2.2 Analysis

The artefact's evaluation analysis involved using Google Sheets [96] and the Rosette sentiment analysis tool [97]. Participant sentiments were retrieved by providing Rosette with participant answers one at a time, unaltered. Rosette associated the participant's key phrases into the 'Automotive' and 'Technology and Computing' categories. Sentiment analysis of participants' answers to the question: '*How do you feel about the interface in the setup?*' shows that 6/9 participants described the user interface positively. To questions: '*Please, describe the Rover*' and '*Please, describe your experience with the Rover*', Rosette suggests 5/9 participants replied positively to both. Further analysis of participant answers is presented below, grouped by survey question.

Analysis of participant replies when asked how they felt about the interface in the setup.
6/9 participants expressed positive feelings about the teleoperation setup, with comments such as "good" and "fine." 3/9 participants noted that they enjoyed the immersive experience of the VR

headset and appreciated the precise video feed. 4/9 participants suggested improvements, such as wanting a more immersive experience or experiencing blurriness in the video feed. 1/9 of the participants found the cubes gimmicky and wondered about their utility. The feedback suggests that the setup was generally effective, with room for improvement.

Analysis of participant replies when asked to describe the Rover.
7/9 participants' responses suggest that the remote toy rover is generally stable and well-constructed, but some improvements could be made to the controllers and the sensitivity of the wheels. 4/9 participants noted that the rover's size and tank-like appearance influenced their driving style and that getting used to driving in VR took some time. 4/9 participants' responses provide helpful feedback for improving the design and usability of the teleoperated toy rover.

Analysis of participant replies when asked to describe their experience with the Rover.
The participants generally had positive experiences with the teleoperation of the toy rover using a VR headset. They found the experience fun and immersive and could see the potential for the technology in various applications, such as remote control of autonomous vehicles in uncrewed areas and warehouses. Some participants noted minor issues, such as a heavy headset or latency in controls, but overall were positive about the experience. Most participants were new to VR technology and found the experience novel and interesting.

Analysis of participant replies when asked to share thoughts on how to improve the overall setup.
The participants generally liked the VR setup and suggested improvements such as decreasing the sensitivity of the rotation, improving the video quality, using a curved panorama screen, adding more objects in the VR environment, and providing better control of the rover. They also suggested incorporating a 360 camera and a 3-axis robotic arm and making the rover bigger to perform more tasks. Some participants suggested using different ways of controlling the rover, such as VR or physical controllers, while others suggested improving the calibration of the existing controller. A few participants suggested improving the camera angle to better see the environment and obstacles.

4.3 Discussion

Overall, participants expressed positive experiences with the teleoperation setup, finding it immersive and enjoyable. They provided valuable feedback on improving the design and usability of the toy rover, including suggestions for improving the controllers, the sensitivity of the wheels, and incorporating additional features such as a 360 camera and a 3-axis robotic arm.

Participants also suggested improvements to the user interface setup, such as using a curved panorama screen and improving the video quality. In summary, their feedback provided useful and encouraging insights for the further development of RoverXR.

5 Conclusion & Future Work

This thesis presents a design study of a teleoperation demonstration for educational purposes, focusing on good user experience and low latency, high-quality video streaming with a virtual scene representation.

RoverXR's development incorporated integrating the M5 RoverC and a Raspberry Pi Zero into one unit. A gantry was iteratively designed and supported with extensive testing and validation. Various video streaming solutions from the RPi were tested in different settings and compared. Several working solutions are presented. The custom-developed AR scene on the Meta Quest 2, built using Godot Engine 4.0, provides users with real-time visual feedback using a WebSocket high-definition video stream. Good user experience is emphasised to inspire future work and highlights the feasibility of using a game engine like Godot for teleoperation setups.

A subjective evaluation showed that participants had overall positive experiences with the teleoperation setup and found it immersive and enjoyable. Participants' feedback provided valuable and encouraging insights for the further development of RoverXR.

Overall, this thesis demonstrates the potential of low-latency, high-quality video streaming and virtual scene representation, using open-stack solutions, in enhancing the user experience of a teleoperation demonstration for educational purposes.
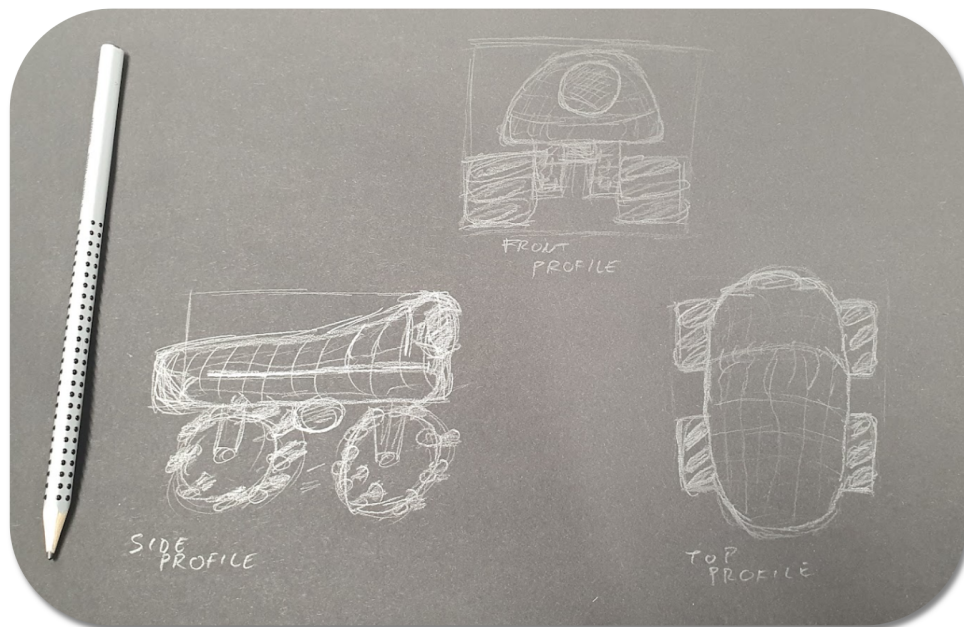
Figure 5.1: Future design concept sketch. Possible future RoverXR design implementation. Bug-like exterior shell is a single flexible part that snaps to the base, reducing the screw count and cost.

Despite accomplishing its objective of seamlessly and reliably integrating all RoverXR components, one aspect of future work includes further gantry design optimisations. For example, RoverXR relies heavily on screws and plastic threads, addressable with a compliant, snap-fit

construction design approach. This would also serve to simplify the assembly and maintenance process. Figure 5.1 reinforces this point by depicting a concept design sketch of a possible next design iteration of RoverXR's appearance. In this proposed design iteration, the RPi Zero 2 W is utilised for movement control and providing a video stream, resulting in a further reduced form factor addition to adding the opportunity to control movement from VR using default controllers.

Table 6 provides key point ideas in areas identified as possible future work, in no order of priority. Please, see Appendix I. (Thesis GitHub Repository: Thesisaurus/Future/Readme) for a more detailed breakdown of this list featuring additional descriptions.

Table 6: Future work recommendations as key points.

Title	Key points
RoverXR Hardware Modifications	*Integrate Pi Zero 2 W*
	New enclosed snap-fit shell (Figure 5.1)
	New camera with autofocus
	Adjustable camera angle
	Improve interface
	Develop a charging base
	Add sensors
RoverXR Software Modifications	*Improve streaming solution*
	Godot scene improvements
	Autonomous driving mode
	Deploy ROS
	Full control from VR
	Godot H264 decoding
RoverXR Evaluation Ideas	*Compare with Unity solution*
	Compare with FPV
	Further evaluate solution

There are many opportunities available for further research and development. Overall, RoverXR, developed in this thesis, provides a solid foundation for future work in teleoperation. Additionally, it demonstrates the key proponents of this research area to future inspired students. RoverXR may additionally be incorporated into a classroom as learning material, providing hands-on teleoperation demos in Godot v4, ready for experimentation.

Finally, with this work, a software artefact can serve as a base for the Autonomous Driving Lab to develop teleoperation solutions in VR - both for RoverXR and Donkey Car. With this open-source solution, it will be easier to maintain and extend support to other platforms.

6 References

[1] "Teleoperation - an overview | ScienceDirect Topics."
https://www.sciencedirect.com/topics/engineering/teleoperation (accessed May 13, 2023).

[2] "DJI FPV - Specs - DJI," *DJI Official*. https://www.dji.com/ee/dji-fpv/specs (accessed May 19, 2023).

[3] By, "2022 FPV Contest: ESP32-Powered FPV Car Uses Javascript For VR Magic," *Hackaday*, Dec. 30, 2022.
https://hackaday.com/2022/12/30/2022-fpv-contest-esp32-powered-fpv-car-uses-javascript-for-vr-magic/ (accessed May 02, 2023).

[4] Consti10, "FPV_VR_OS." Apr. 19, 2023. Accessed: Apr. 28, 2023. [Online]. Available:
https://github.com/Consti10/FPV_VR_OS

[5] "FPV_VR Ultra low latency for FPV Virtual Reality Android App - RC Groups."
https://www.rcgroups.com/forums/showthread.php?2873827-FPV_VR-Ultra-low-latency-for-FPV-Virtual-Reality-Android-App (accessed Apr. 28, 2023).

[6] A. Sherafatian, "Teleoperation of Remote Controlled Toy Cars in VR".

[7] R. Kõvask, "State of the art VR Driving Simulation for Physical Test Car Using LiDAR for Mapping the Surrounding Environment".

[8] E. Lopez, "VR-enabled teleoperation of robotic arm between University of Manchester and University of Edinburgh. An initial approach," *Erwin Lopez Site*, Jan. 04, 2021.
https://www.erwinlopez.com/talk/crosshub-2020/ (accessed Jan. 22, 2023).

[9] G. Silvera, A. Biswas, and H. Admoni, "DReyeVR: Democratizing Virtual Reality Driving Simulation for Behavioural & Interaction Research." arXiv, Jan. 07, 2022. Accessed: Mar. 03, 2023. [Online]. Available: http://arxiv.org/abs/2201.01931

[10] L. S. Yim *et al.*, "WFH-VR: Teleoperating a Robot Arm to set a Dining Table across the Globe via Virtual Reality," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyoto, Japan: IEEE, Oct. 2022, pp. 4927–4934. doi: 10.1109/IROS47612.2022.9981729.

[11] P. Stotko *et al.*, "A VR System for Immersive Teleoperation and Live Exploration with a Mobile Robot," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 3630–3637. doi: 10.1109/IROS40897.2019.8968598.

[12] E. Cano-Marin, *Recent advances in research teleoperation, telepresence and virtual reality*. 2015.

[13] M. Moniruzzaman, A. Rassau, D. Chai, and S. M. S. Islam, "Teleoperation methods and enhancement techniques for mobile robots: A comprehensive survey," *Robot. Auton. Syst.*, vol. 150, p. 103973, Apr. 2022, doi: 10.1016/j.robot.2021.103973.

[14] "RCSnail." https://rcsnail.ee/ (accessed Apr. 27, 2023).

[15] "Donkey® Car," *Donkey® Car*. https://www.donkeycar.com/ (accessed May 13, 2023).

[16] R. P. Ltd, "Buy a Raspberry Pi Zero," *Raspberry Pi*.
https://www.raspberrypi.com/products/raspberry-pi-zero/ (accessed May 06, 2023).

[17] "m5-docs." http://docs.m5stack.com/en/hat/hat_roverc_pro (accessed May 06, 2023).

[18] "Meta Quest 2: Immersive All-In-One VR Headset | Meta Store."
https://www.meta.com/quest/products/quest-2/ (accessed May 06, 2023).

[19] "Chair of Distributed Systems," *University of Tartu*, Jul. 27, 2021.
https://ut.ee/en/chair-distributed-systems (accessed May 15, 2023).

[20] "The CGVR Lab," *The CGVR Lab*. https://cgvr.cs.ut.ee/ (accessed Apr. 11, 2023).

[21] "Intelligent Materials and Systems Lab."
https://ims.ut.ee/Intelligent_Materials%2BSystems (accessed May 15, 2023).

[22]   G. Engine, "Godot Engine - Free and open source 2D and 3D game engine," *Godot Engine*. https://godotengine.org/ (accessed May 05, 2023).

[23]   T. Fong and C. Thorpe, "Vehicle teleoperation interfaces," *Auton. Robots*, vol. 11, no. 1, pp. 9–18, 2001, doi: 10.1023/A:1011295826834.

[24]   F. Tener and J. Lanir, "Driving from a Distance: Challenges and Guidelines for Autonomous Vehicle Teleoperation Interfaces," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, in CHI '22. New York, NY, USA: Association for Computing Machinery, Apr. 2022, pp. 1–13. doi: 10.1145/3491102.3501827.

[25]   "Teleoperation software for remote control of vehicles, machines and robots," *Voysys*. https://www.voysys.se (accessed May 13, 2023).

[26]   "Home page," *Clevon*, Apr. 28, 2023. https://clevon.com/et/ (accessed May 13, 2023).

[27]   "Starship Technologies: Autonomous robot delivery - The future of delivery - today!" https://www.starship.xyz/ (accessed May 15, 2023).

[28]   "Ottopia - Safe and Cyber-Secure Teleoperation Software." https://ottopia.tech/ (accessed May 13, 2023).

[29]   "Fernride - Driving logistics automation." https://www.fernride.com/ (accessed May 13, 2023).

[30]   "Home - Seafar." https://seafar.eu/ (accessed May 13, 2023).

[31]   "Shadow Robot | Dexterous Robotic Hands & Teleoperated Robots," *Shadow Robot*, Sep. 13, 2022. https://www.shadowrobot.com/ (accessed May 13, 2023).

[32]   "Roboauto." https://roboauto.tech/ (accessed May 13, 2023).

[33]   "Teleoperation Workcells." https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Automation_and_Ro botics/Teleoperation_Workcells (accessed May 13, 2023).

[34]   J. Jerald, *The VR book: human-centered design for virtual reality*. in ACM books, no. 8. New York] [San Rafael, California: Association for computing machinery Morgan & Claypool publishers, 2016.

[35]   A. Augustin, "websockets: An implementation of the WebSocket Protocol (RFC 6455 & 7692)."

[36]   "TRANSMISSION CONTROL PROTOCOL." https://www.ietf.org/rfc/rfc793.txt (accessed May 16, 2023).

[37]   "User Datagram Protocol." https://www.ietf.org/rfc/rfc768.txt (accessed May 16, 2023).

[38]   H. Nielsen *et al.*, "Hypertext Transfer Protocol – HTTP/1.1," Internet Engineering Task Force, Request for Comments RFC 2616, Jun. 1999. doi: 10.17487/RFC2616.

[39]   A. Kaknjo, M. Rao, E. Omerdic, L. Robinson, D. Toal, and T. Newe, "Real-Time Video Latency Measurement between a Robot and Its Remote Control Station: Causes and Mitigation," *Wirel. Commun. Mob. Comput.*, vol. 2018, p. e8638019, Dec. 2018, doi: 10.1155/2018/8638019.

[40]   "WireShark: WebSocket." https://wiki.wireshark.org/WebSocket.md (accessed May 11, 2023).

[41]   "Transmission Control Protocol (TCP) (article)," *Khan Academy*. https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-int ernet/xcae6f4a7ff015e7d:transporting-packets/a/transmission-control-protocol--tcp (accessed May 11, 2023).

[42]   "User Datagram Protocol (UDP) (article) | Khan Academy." https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-int ernet/xcae6f4a7ff015e7d:transporting-packets/a/user-datagram-protocol-udp (accessed May 11, 2023).

[43]   "HTTP | MDN," May 10, 2023. https://developer.mozilla.org/en-US/docs/Web/HTTP (accessed May 11, 2023).

[44]   D. M. Maesita, "Chapter 8 Social Influence within Immersive Virtual Environments",

Accessed: May 15, 2023. [Online]. Available:
https://www.academia.edu/38670626/Chapter_8_Social_Influence_within_Immersive_Virtual_Environments

[45]   P. Milgram, "An Augmented Reality Based Teleoperation Interface For Unstructured Environments," Jan. 1997.

[46]   J. Jerald and M. Whitton, "Relating Scene-Motion Thresholds to Latency Thresholds for Head-Mounted Displays," *Proc. IEEE Virtual Real. Conf.*, pp. 211–218, 2009, doi: 10.1109/VR.2009.4811025.

[47]   "The Open Source Definition," *Open Source Initiative*, Jul. 07, 2006. https://opensource.org/osd/ (accessed May 19, 2023).

[48]   "The MIT License," *Open Source Initiative*, Oct. 31, 2006. https://opensource.org/license/mit/ (accessed May 19, 2023).

[49]   "ESP32-CAM Video Streaming and Face Recognition with Arduino IDE | Random Nerd Tutorials." https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/ (accessed May 07, 2023).

[50]   R. P. Ltd, "Buy a Raspberry Pi Camera Module 2," *Raspberry Pi*. https://www.raspberrypi.com/products/camera-module-v2/ (accessed May 02, 2023).

[51]   R. P. Ltd, "Buy a Raspberry Pi Camera Module 3," *Raspberry Pi*. https://www.raspberrypi.com/products/camera-module-3/ (accessed May 07, 2023).

[52]   T.K. HAREENDRAN, "Portable Power- 18650 Battery Shield for Raspberry Pi & Arduino," *ElectroSchematics*, Sep. 12, 2019. https://www.electroschematics.com/battery-shield/ (accessed May 07, 2023).

[53]   "BU-501a: Discharge Characteristics of Li-ion," *Battery University*, Sep. 03, 2010. https://batteryuniversity.com/article/bu-501a-discharge-characteristics-of-li-ion (accessed May 08, 2023).

[54]   "Li-ion Voltage Analysis." https://siliconlightworks.com/li-ion-voltage (accessed May 07, 2023).

[55]   "Polylactic Acid or Polylactide (PLA)," *Bioplastics News*, Apr. 18, 2015. https://bioplasticsnews.com/polylactic-acid-or-polylactide-pla/ (accessed May 08, 2023).

[56]   "Fusion 360 | 3D CAD, CAM, CAE, & PCB Cloud-Based Software | Autodesk." https://www.autodesk.com/products/fusion-360/overview (accessed May 16, 2023).

[57]   R. Brown, "Welcome to the OpenWrt Project," *OpenWrt Wiki*, Sep. 27, 2016. https://openwrt.org/start (accessed May 16, 2023).

[58]   "JoyC (W/O M5StickC) Omni-directional Controller," *m5stack-store*. https://shop.m5stack.com/products/joyc-w-o-m5stickc (accessed May 06, 2023).

[59]   "GL-SF1200 | Dual-band Ultra-fast 5GHz VPN Wi-Fi Router," Jul. 08, 2021. https://www.gl-inet.com/products/gl-sf1200/ (accessed May 06, 2023).

[60]   "WebSocket," *Godot Engine documentation*. https://docs.godotengine.org/en/stable/tutorials/networking/websocket.html (accessed Apr. 24, 2023).

[61]   L. Srinivasan, J. Scharnagl, and K. Schilling, "Analysis of WebSockets as the New Age Protocol for Remote Robot Tele-operation," *IFAC Proc. Vol.*, vol. 46, no. 29, pp. 83–88, Jan. 2013, doi: 10.3182/20131111-3-KR-2043.00032.

[62]   H. Czedik-Eysenberg, "PyImageStream - Python WebSocket Image Stream." Apr. 27, 2023. Accessed: Apr. 28, 2023. [Online]. Available: https://github.com/Bronkoknorb/PyImageStream/blob/2af00b42373035f4a1e300db3af7458d083e8d03/main.py

[63]   N. Kulkarni and T. Eltaieb, "Video Streaming Over Full Duplex Network Using Websocket and Its Performance Evaluation," vol. 2, no. 4, 2015.

[64]   T. Dittmann, "Lessons learned from building a WebSocket server," *Medium*, Sep. 23,

2021. https://itnext.io/lessons-learned-from-building-a-websocket-server-e9bc0bd3ef80 (accessed May 11, 2023).

[65]   J. Pérez and J. K. Nurminen, "Electric vehicles communicating with WebSockets - measurements and estimations," in *IEEE PES ISGT Europe 2013*, Oct. 2013, pp. 1–5. doi: 10.1109/ISGTEurope.2013.6695313.

[66]   D. Skvorc, M. Horvat, and S. Srbljic, "Performance evaluation of Websocket protocol for implementation of full-duplex web streams," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2014, pp. 1003–1008. doi: 10.1109/MIPRO.2014.6859715.

[67]   "WebSocket," *Godot Engine documentation*. https://docs.godotengine.org/en/stable/tutorials/networking/websocket.html (accessed May 16, 2023).

[68]   "Lightweight justice for your SBC!," *DietPi*. https://dietpi.com/ (accessed May 11, 2023).

[69]   "MotionEye OS - Raspberry Valley." https://raspberry-valley.azurewebsites.net/MotionEye-OS/ (accessed May 11, 2023).

[70]   "DroidCam OBS Camera by Dev47Apps." https://www.dev47apps.com/obs/ (accessed May 11, 2023).

[71]   D. Jones, "Picamera 1.13 Documentation".

[72]   "Picamera2." Raspberry Pi, May 11, 2023. Accessed: May 11, 2023. [Online]. Available: https://github.com/raspberrypi/picamera2

[73]   S. Weil, "Raspberry Pi Documentation." Apr. 27, 2023. Accessed: May 11, 2023. [Online]. Available: https://github.com/stweil/raspberrypi-documentation

[74]   "OpenHD/OpenHD." OpenHD, Apr. 28, 2023. Accessed: Apr. 28, 2023. [Online]. Available: https://github.com/OpenHD/OpenHD

[75]   "rodizio1/EZ-WifiBroadcast: Affordable Digital HD Video Transmission made easy!" https://github.com/rodizio1/EZ-WifiBroadcast (accessed Apr. 28, 2023).

[76]   "Droste effect," *Wikipedia*. Apr. 06, 2023. Accessed: Apr. 11, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Droste_effect&oldid=1148515076

[77]   "FFmpeg." https://ffmpeg.org/ (accessed Apr. 24, 2023).

[78]   "mpv.io." https://mpv.io/ (accessed May 11, 2023).

[79]   "Theora.org :: main - Theora, video for everyone." https://www.theora.org/ (accessed May 20, 2023).

[80]   "Playing videos," *Godot Engine documentation*. https://docs.godotengine.org/en/stable/tutorials/animation/playing_videos.html (accessed Apr. 11, 2023).

[81]   P. Stewart, S. McCanne, B. Fenner (ˢˣʸ), L. Berc, and R. Frederick, "RTP Payload Format for JPEG-compressed Video," Internet Engineering Task Force, Request for Comments RFC 2435, Oct. 1998. doi: 10.17487/RFC2435.

[82]   K. Iqbal, "MJPEG - Motion JPEG File Format," Feb. 15, 2021. https://docs.fileformat.com/video/mjpeg/ (accessed May 19, 2023).

[83]   "JPG Signature Format: Documentation & Recovery Example." https://www.file-recovery.com/jpg-signature-format.htm (accessed May 19, 2023).

[84]   "The Picamera2 Library".

[85]   "Welcome — libcamera." https://libcamera.org/ (accessed Apr. 29, 2023).

[86]   "Picamera2/encoders/mjpeg_encoder.py." Raspberry Pi, May 11, 2023. Accessed: May 11, 2023. [Online]. Available: https://github.com/raspberrypi/picamera2/blob/main/picamera2/encoders/mjpeg_encoder.py#L11

[87]   "OpenXR - High-performance access to AR and VR —collectively known as XR— platforms and devices," *The Khronos Group*, Dec. 06, 2016. https://www.khronos.org/openxr/ (accessed May 15, 2023).

[88]   "GDScript reference," *Godot Engine documentation*.
       https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript_basics.html
       (accessed May 11, 2023).

[89]   "Learn GDScript From Zero." GDQuest, Feb. 06, 2023. Accessed: Feb. 06, 2023.
       [Online]. Available: https://github.com/GDQuest/learn-gdscript

[90]   "Introducing XR tools," *Godot Engine documentation*.
       https://docs.godotengine.org/en/stable/tutorials/xr/introducing_xr_tools.html (accessed May
       12, 2023).

[91]   "Immersive Web Developer Home." https://immersiveweb.dev/ (accessed May 15, 2023).

[92]   Thingiverse.com, "Baby Yoda - Free Sample by MarVin_Miniatures."
       https://www.thingiverse.com/thing:4038181 (accessed May 11, 2023).

[93]   "Discover - Autonomous Driving Lab." https://adl.cs.ut.ee/ (accessed Jan. 25, 2023).

[94]   "Wifi Analyzer." https://www.wifianalyzer.info/ (accessed May 11, 2023).

[95]   "Home," *Fing*. https://www.fing.com (accessed May 11, 2023).

[96]   "Google Sheets: Online Spreadsheet Editor | Google Workspace."
       https://www.facebook.com/GoogleDocs/ (accessed May 14, 2023).

[97]   "Rosette Text Analytics Platform - AI for Human Language," *Rosette Text Analytics*.
       https://www.rosette.com/ (accessed May 14, 2023).

[98]   C. Inc, "Clockify - FREE Time Tracking Software," *Clockify*. https://clockify.me
       (accessed May 13, 2023).

[99]   "How Grammarly Works | Grammarly."
       https://www.grammarly.com/how-grammarly-works (accessed May 20, 2023).

[100]   "Introducing ChatGPT." https://openai.com/blog/chatgpt (accessed May 14, 2023).

7 Appendices

I.   Thesis GitHub Repository

https://github.com/mbz4/RoverXR

This is the central thesis GitHub repository, hosting all relevant files and documentation related to RoverXR, divided into two directories: 'Blueprints' and 'Thesisaurus'.

II.   Thesis Workload Analysis

This subsection presents a concise summary and analysis of the workload invested during this thesis. In total, this thesis comprises 12 milestones, completed in the preparation of this thesis with approximately 465 hours invested. Figure 7.1 depicts time spent in hours (vertical axis) per milestone, listed in order of completion (horizontal axis).

Clockify [98] time management software helped track time, allowing for more accurate and precise tracking of specific tasks and activities throughout the design and development process. Clockify was complemented with timestamps in personal notes, denoting milestone deadlines.
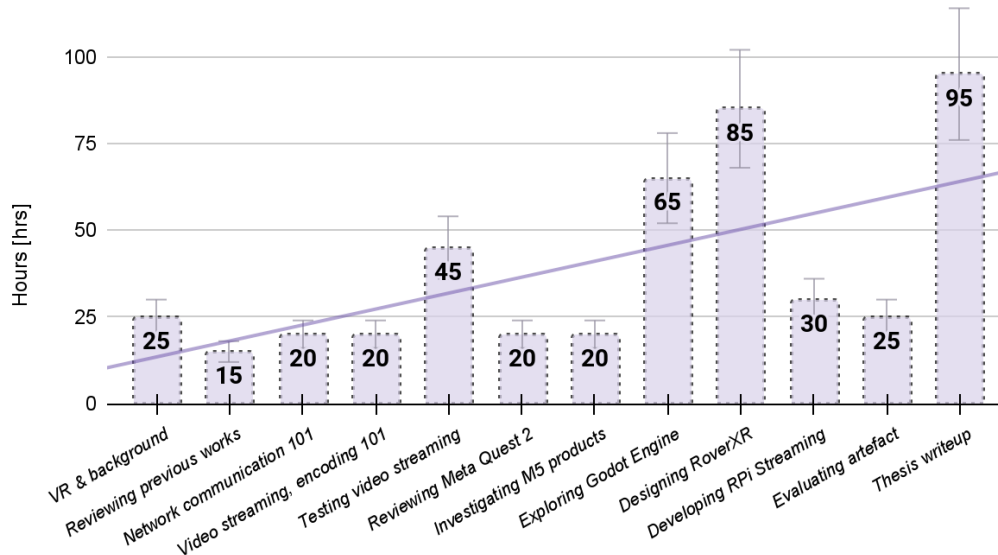
Figure 7.1: Thesis workload analysis across 12 milestones in order of completion. The most effort investment was in the design phase of RoverXR, and about a third less exploring the Godot Engine. Meanwhile, reviewing previous works, learning about video streaming, getting familiar with the M5 product ecosystem and evaluating the artefact received relatively little attention. Based on the trendline, developments were more pronounced towards the end.

The first set of tasks completed included getting familiar with VR with a preliminary background study, which took an estimated 25 hours to complete. The final task was the thesis write-up, which took about 95 hours to complete. Please, see Appendix I. (Thesisaurus/Workload Analysis) for an extended workload analysis, including key point breakdowns of each milestone.

III. Acknowledgements

Matevž B. Zorec

IV. Non-exclusive licence to reproduce thesis and make thesis public

I, Matevž Borjan Zorec

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

"XR TELEOPERATION DEMO DEVELOPMENT"

supervised by Ulrich Norbisrath

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Matevž Borjan Zorec
20. 05. 2023