A Real-time, Remotely Operated Vehicle, using a Raspberry Pi and a Virtual Reality Headset

Yernar Kenzhetayev, István Nagy

Óbuda University, Bánki Donát Faculty of Mechanical and Security Engineering, Institute of Mechatronics and Vehicle Engineering, Dept. of Mechatronics Népszínház u. 8, H-1081 Budapest, Hungary; E-mail: yernar.kenzhetayev@stud.uni-obuda.hu; nagy.istvan@bgk.uni-obuda.hu (ORCID: 0000-0002-8250-7969)

Abstract: The usage of modern information and communication technologies, such as virtual and mixed reality, offers new options for controlling and monitoring IoT devices. For example, Head Mounted Displays (HMDs) are gaining popularity as a tool to enhance user productivity and enjoyment. This development is also related to the recent advancements in computer technology and the decline in the price of that technology: HMDs are now more functional while also being more widely available on the market. This paper presents a two-wheel robot car that can be controlled remotely in real-time using HMD. The remote control is done in Virtual Reality with the help of Unity 3D. The open-source game engine decreases cost and development time. There are separate objects for the steering wheel, transmission, screen, and stop button. Both controllers and the user's hands can be used as input manipulators. The Oculus headset's external cameras use hand recognition to implement this feature. The Raspberry Pi 4 has three main functions: first is to control DC motors with GPIO pins, second is to send video stream from the camera to HMD and third is to accept control signals from HMD and perform them. The data transfer of the Virtual Reality headset and Remotely Operated Vehicle (ROV) is done through server-client communication. Raspberry plays the role of a server, which is written on the Flask framework of the python programming language. This server works using asynchronous principles and the OpenCV library for working with images. GPIO pins are controlled by the server, and it receives requests as well. VR headset is a client, which is written in C# on the Unity game engine. The device interacts with the server when the user does any action and transfers the video stream to the screen in real-time. The configuration of input systems is done with the help of the official Oculus Software Development Kit.

The convenience of new input systems and their inherent advantages and disadvantages are discussed. Full-scale tests and findings on whether the suggested approach is practical for actual offshore activities are described as well. The results reveal that it is easier and less expensive to modify the input layout while maintaining the same message-sending technology without the limitations of a physical control panel for the ROV operator.

Keywords: Virtual Reality (VR); Remotely Operated Vehicle (ROV); Head Mounted Display (HMD); Raspberry Pi; Unity 3D; OpenCV

1 Introduction

The modern world is surprising us with new technologies, breakthroughs, and creative solutions every day. The tempo of growth is so fast, that it can affect even our daily routine and work. The improvement in both hardware and software opened a new opportunity for Virtual Reality (VR) technology. The working environment will get harsher, and the control requirements will get more stringent in the future. There are different control methods for vehicle control, for example in [28] a Fuzzy based velocity planner is used for the speed regulation of a wheelchair type mobile platform. The usage of ultra-remote control and virtual reality technology can liberate employees from the constraints of their working location. By recreating the real world with the aid of a virtual reality system and combining human perception with the exact control of a manipulator robot, virtual reality-based operator robot control can give users an immersive experience.

The need in Unmanned Aerial Vehicle control is growing and new alternative approach could extend existing solid methods. VR provides opportunity to create new method. In comparison one example of method with feedback signal from external environment was studied from [29]

The goal of this project is to develop a system that requires a minimum of changes and can operate with any hardware setup in real-time. The creation of such a complex system requires a detail-oriented approach to work with every part of it.

The significant factor in the quality of performance is choosing appropriate technologies and hardware. Even the newest libraries of programming languages can be limited because VR is a new field of software development. There are a few engines or applications that support creating a virtual reality environment. They are using high-level programming languages like C# while working with hardware is done by using low-level languages. That is why it is important to think about the risks related to compatibility issues. Usually, mixed work of different platforms faces large performance issues, such as problems with latency, lack of documentation, and going to the undiscovered area.

Creating exemplary architecture or system design is the crucial step in this project. During the creation of the architecture priorities of the project must be kept. It can be done only by selecting the most important features and making that list clear.

The vehicle should be examined by several criteria. First of all, free movement in any direction with common limitations of a landed device. A simple car has three degrees of freedom: moving to any point in two dimensions and controlling its rotation around the z-axis. The final robot car is needed to have at least the same characteristics. Secondly, the connection between the devices should be established via a local or global network. There is a two-way connection to be set up. Sending signals from the user's side (VR headset). After, ROV performs instructions that were received. At the same time, ROV records images by the camera and sends them to HMD. The communication between ROV and VR headset accepts only low latency of signal transmission because of live video transmission from the camera of the robot. The application has to be unified since the input and output devices for a virtual helmet may differ. The VR environment requires many items to be created. For instance, the steering wheel, the screen, and the start and stop buttons of the movement. The steering wheel should closely resemble the real-world model since it is easier for any user to use. The screen plays the role of the display, which demonstrates frames caught by the ROV's camera.

The results are mandatory to be analyzed and compared to other solutions. The future work can be defined from the analysis.

2 Architecture of the Developed System

For creating a better solution, similar attempts should be taken into consideration. This approach helps to avoid meeting the same limitations and mistakes. The flowchart of the system is given in Figure 1.

Both hardware and software structures have to satisfy that operational requirement. After detailed learning of the functions the final system should have, it was decided to make the server-client type of connection between systems. The server is carried by the mobile robot itself and HMD connects to it via Wi-Fi. The programming code of the system can be found in Annexes section. The following hardware elements were chosen for the project:

- Raspberry Pi 4
- L298N Motor Driver Module
- 2 DC motors and vehicle body with wheels
- Oculus Quest 2

The computational power of chosen parts is enough to produce desired functions. Raspberry Pi 4 can solve as a "mini-PC" for good calculations and establishing the WiFi connection. The motor driver module (L298N) is commonly used for driving two DC motors at the same time, moreover can be directly connected to the Raspberry Pi 4. The DC motors have been chosen based on the L298N, criteria, i.e., the supply voltage of the motors had to be between 5-35 [V] DC.

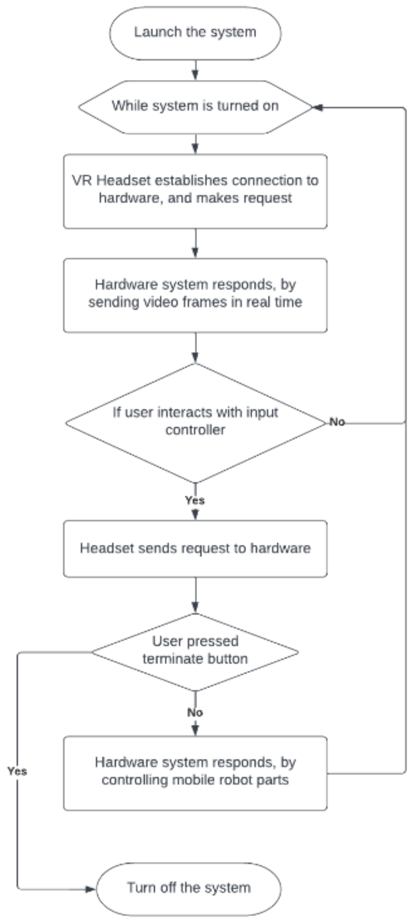

Launch the system
While system is turned on
VR Headset establishes connection to hardware, and makes request
Hardware system responds, by sending video frames in real time
If user interacts with input controller
No
Yes
Headset sends request to hardware
User pressed terminate button
No
Yes
Hardware system responds, by controlling mobile robot parts
Turn off the system

Figure 1
The flowchart of the operation of system

2.1 Hardware Structure of the System

This project's hardware loop is made up of three main components that communicate with one another through a software stack. The three components appearing in Figure 2, are Oculus Quest 2 VR headset, Raspberry Pi 4, and Mobile robot parts. It was possible to decrease the number of the required elements because both the VR helmet and Raspberry are performing multiple functions, that were not available before.

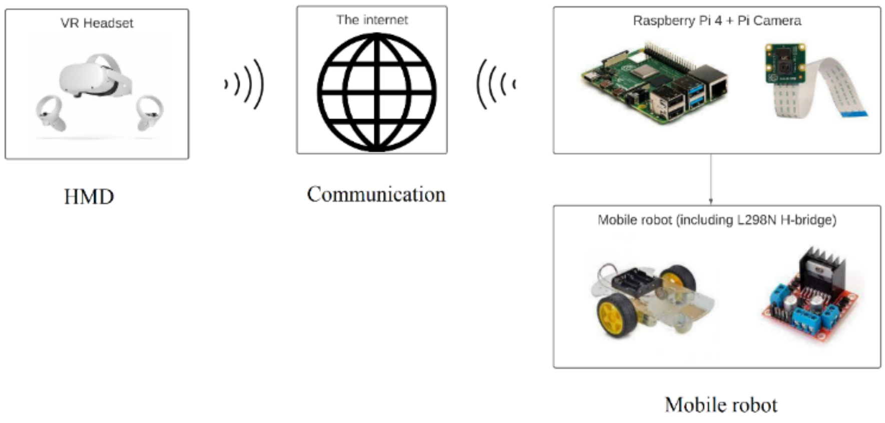

VR Headset
HMD
The internet
Communication
Raspberry Pi 4 + Pi Camera
Mobile robot (including L298N H-bridge)
Mobile robot

Figure 2
Hardware structure

Oculus Quest 2 VR headset is an innovative fully autonomous device, As the Quest 2 is running a VR simulation environment where the operator sits in a chair and can operate the ROV, it handles all of the heavy computations in terms of video processing power. Real-time orientation and movement information is provided by the VR headset on its own for rendering and displaying a VR scene. It does not require a ground station as it was with its previous model Oculus Rift, meaning the need for separate computers in this project is eliminated. HMD is connected to the internet and used as the main input controller of the user. The networking characteristics of Quest 2 support web client function as well. It reaches Raspberry Pi through an internet connection.

Raspberry Pi 4 is interfacing between physical devices. It receives the signal from the user and performs the right instructions as streaming video packages back to the input device or sending movement signals to mobile robot components. The video capturing is done via the Pi camera. The specific pins are used to connect Raspberry to H-bridge.

Mobile robot components are the main body, 2 driving wheels, which are moved by DC motors, and one metal swivel wheel. The L298N H-bridge is used for controlling them. The mobile robot is capable of moving forward, and backward, rotating to the right and left around the z-axis. It has all the required functionality and was chosen over a four-wheel mobile robot after deep analysis.

A more detailed description of hardware components is provided below.

2.1.1 Mobile Robot Selection

Selecting a mobile robot for a project's needs can be a challenging process that requires comparison of multiple criteria. In order to determine the most suitable robot for a given project, it is necessary to evaluate factors such as the robot's size, speed, and agility, as well as its sensors, actuators, and other hardware.

Additionally, the software capabilities of the robot must be considered, including its programming language and available libraries. Furthermore, the price, availability, and support options for the robot must be taken into account to ensure that it can be obtained and maintained over time. Ultimately, the selection of a mobile robot for a project's needs is a complex process that requires careful consideration of many different factors.

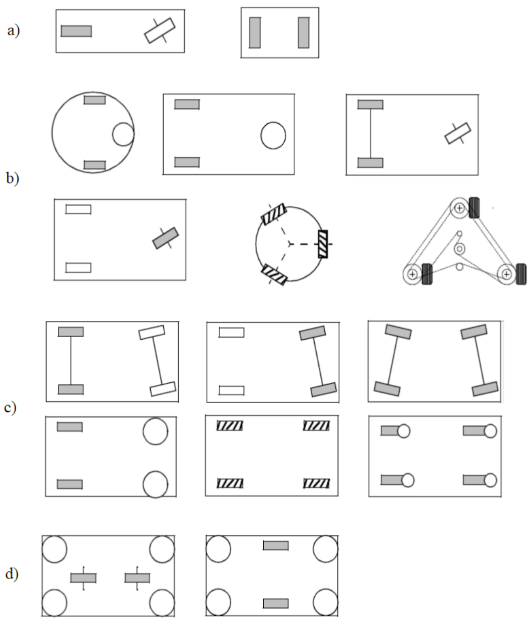


Figure 3
Examples of different wheel arrangements: a) Two-wheel b) Three-wheel c) Four-wheel d) Six-wheel robots [23]

Numerous robot layouts are shown in Figure 3. It is preferable to specify particular criteria that fulfill the present scope of the study due to the broad variety of mobile robot kinds. The choice of the mobile robot type was based on several factors like degree of mobility, degree of steerability, degree of maneuverability and complexity of robot's implementation.

It was chosen to perform experiments on a mobile robot with two separate driving wheels that can be orientated and controlled by acting on each wheel's speed, as illustrated on the schematic model, after studying several types of mobile robots with driving wheels and encoder systems.

Notations used in Figure 4. Coordinates XR, YR, are representing the center position (R) of the mobile robot, while αR is the orientation, in Cartesian system. Parameter „L" is representing the axial distance between the rear wheels, VR and VL are the robot's linear velocities of right and left wheels, and ωL, ωR are the angular velocities, respectively. Let r denote the wheel radius of the driving wheels.

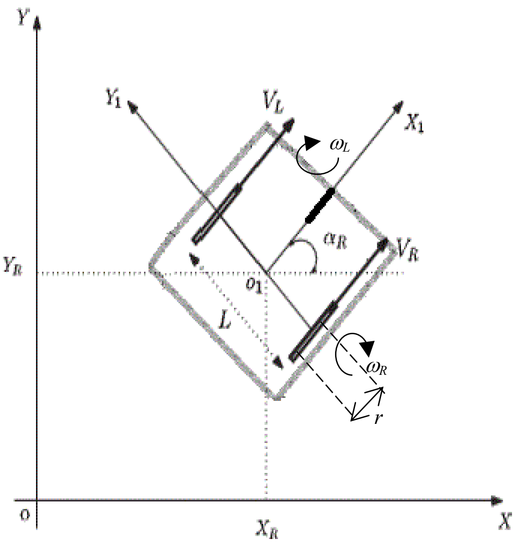


Figure 4
The schematic model of a mobile, nonholonomic robot

The kinematic models [direct: eq.(1), inverse: eq.(2)], for differential drives is represented by equations:

$$X_R(t) = \frac{1}{2}\int_0^t [V_R(t) + V_L(t)]\cos[\alpha_R(t)]dt; \quad \dot{X}_R = V.cos \propto_R$$
$$Y_R(t) = \frac{1}{2}\int_0^t [V_R(t) + V_L(t)]\sin[\alpha_R(t)]dt; \quad \dot{Y}_R = V.sin \propto_R \tag{1}$$
$$\alpha_R(t) = \frac{1}{l}\int_0^t [V_R(t) - V_L(t)]dt$$

$$\omega_R = \frac{V_R}{r}; \ \omega_L = \frac{V_L}{r} \tag{2}$$

One of the differential-driven robot's benefits is the ability to turn, it helps to park and avoid obstacles in a better way. The four-wheel robots have a few troubles with those. The four-wheel robots should use more motors, materials, and energy which is considered a drawback in the current test purposes. The robots using Omni-wheels have great maneuverability. However, they were not used due to complicated structure and position calculations. They are also proven to have low efficiency as not all wheels can move in the same direction.

In addition to the above analysis, usage of the three-wheel differential-driven robot with the caster wheel seems as the most appropriate variant for the purposes of the project. The prototype can be seen in Figure 5.

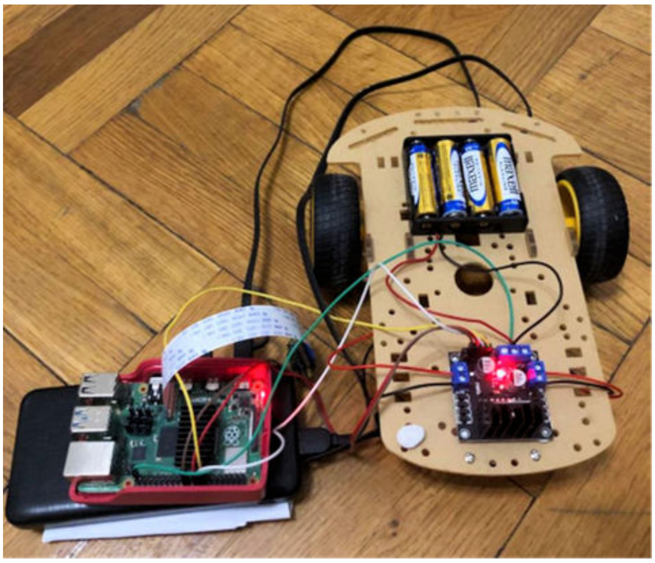


Figure 5
The mobile robot prototype, built by the author

2.1.2 Mobile Robot Electronic Components

H-Bridge

An integrated monolithic circuit, the L298N is built in a 15-lead Multiwatt and PowerSO20 packaging [7]. It is a dual full-bridge driver with high voltage and current that can drive inductive loads including relays, solenoids, DC motors, and stepping motors and it is designed to accept normal TTL logic levels [7]. Without relying on the input signals, the device may be enabled or disabled using one of two enable inputs. Lower transistors of each bridge are coupled together at their emitters, and an external sensing resistor can be attached to the corresponding external terminal. An extra supply input is provided to allow the logic to function at a lower voltage.

The good example for electric circuits used in vehicles can be followed in [31].

The body of the car is a robust acrylic with many holes for the flexible mounting of components. The size of it is 100 x 200 x 60 mm.

Raspberry Pi 4

The Raspberry Pi is a brand of single-board computer, which is a device with a processor, memory, USB port, and other components on a single PCB board. Four models have been released so far, with updated upgrades for each model that have improved performance and included new capabilities.

Raspberry Pi is the perfect match for the needs of this project, as it is capable of controlling hardware by running high-level software applications. There are some limitations as the basic number of pins, or specific Operation System (OS) installed on it, but it is not causing any problem in this case.

Specifications:

- Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 4GB LPDDR4-3200 SDRAM
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless LAN
- Bluetooth 5.0
- Raspberry Pi standard 40-pin General Purpose Input Output (GPIO) header

Raspberry Pi camera is able to capture full HD 1080p video as well as high-resolution photos, and they can all be completely programmed. The still resolution of the used camera is 8 MegaPixels. Its horizontal Field of View (FoV) is 62.2 degrees and its vertical FoV is 48.8 degrees. The supported image and video formats are JPEG, JPEG + DNG (raw), BMP, PNG, YUV420, RGB888, and raw h.264, MJPEG [21]. The libcamera built-in library is used to control all kinds of actions related to the camera with Raspberry.

GPIO pins can be set up as general-purpose input, general-purpose output, or one of up to six unique alternate configurations, each of which has a different function depending on the pin. There are 28 GPIO pins accessible, which match the GPIO pins on the 40-pin header of the Raspberry Pi 4 Model B.

The main advantage of RaspbianOS is that it can be used in the same way as Linux distributions, meaning that high-level software applications are compatible to be performed on it. It comes with Python2.7 libraries already preinstalled, but using any other language is also not an issue. Python 3.6 with Flask and OpenCV libraries are used by the system for maintaining the server, which works with images taken by the Raspberry camera. The server uses built-in GPIO control functions for setting high or low output on specific pins.

2.1.3 Mobile Robot Driving (locomotion) System

Direct current (DC) stepper motors are a type of electric motor that produce precise, controlled movements, frequently in small increments or steps. It is made up of a DC motor, a set of gears, and a number of electromagnets that are used to regulate the motion of the motor's shaft or rotor.

This H-Bridge is used for controlling two-stepping DC motors, which are responsible for the forward and backward movement of the wheels. Four in-and-out pins are available. These DC motors' input range of voltage is between 3V and 12V.

The motor reduction ratio of 48:1 [8]. This means that for every 48 turns of the motor's shaft, the output shaft will make one full revolution.

An odometer is a device that is used to measure the distance traveled by a vehicle or other object. It consists of a wheel attached to a DC motor, which rotates to measure the distance traveled. The number of gaps in the wheel determines the resolution of the device, with a higher number of gaps resulting in greater accuracy. The gear ratio of the motor is an important factor, as it determines the smallest step or increment that the wheel can move. The odometer type is relative in this case. The resolution of the gadget will be 360/21, or 17 degrees, in this situation when the wheel on the odometer has 21 gaps. The mentioned H-bridge and DC motors are chosen as standard components that are widely used for similar purposes and tested by time.

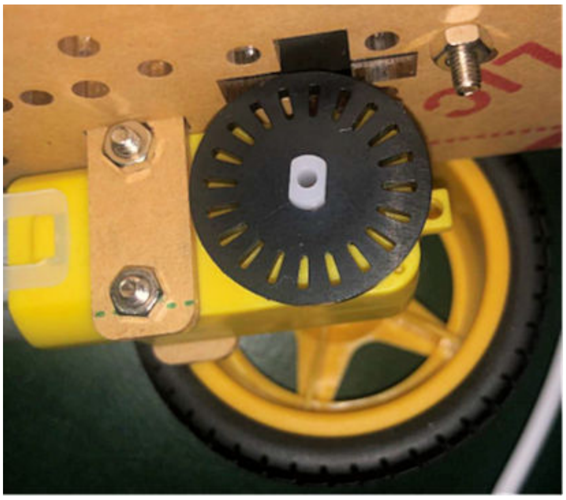


Figure 6
The odometer on the robot

In case of low computational power, the existing video processing method can be replaced by Radio-frequency triangulation method in indoor environment, as it requires special trackers [30]. The operation of chosen video processing is described further.

2.1.4 Oculus Quest 2

Oculus Quest 2 is used as HMD in this work. It is one of the most known headsets on the market, which is available both for developers and consumers. Since the Oculus Quest 2 is primarily intended for VR applications, where the ability to move around in a virtual world is crucial, it contains a magnetometer, accelerometer, and gyroscope for detecting head angle at a high refresh rate. An external infrared tracker may assess the device's relative position in space in addition to its attitude, which further improves the precision of attitude readings. It provides both position and orientation prediction, meaning work in 6 Degrees of Freedom (DoF). It is much better compared to Oculus Go, which has only 3 DoF.

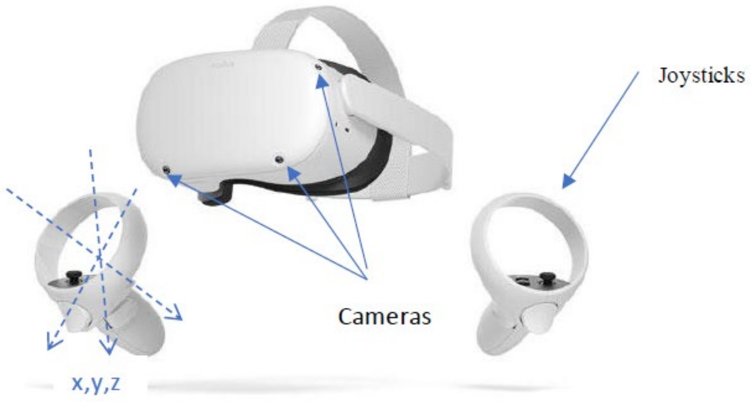


Figure 7
Oculus Quest 2. Cameras and coordinates of joysticks [19]

Also, compared to his ancestor, which is Oculus Rift, Quest works in an autonomous way. A connection to a personal computer is not required for processing. The only dependence now is access to the internet. If it has the internet, HMD can work anywhere, removing that limitation.

The VR helmet contains gyroscopes and accelerometers that can determine the position of the head. The quaternion operation formula is used to determine the rotational angle. The quaternion then determines the Euler angle, and this angle of rotation is applied to regulate the biaxial suspension. As can be seen in Figure 8, (θ, ψ, φ) are usually used to represent the rotation of a coordinate system. θ, ψ, φ are the pitch angle around the x-axis, the yaw angle around the y-axis and the roll angle around the z-axis correspondingly.

The inside-out cameras on the Meta Quest headset are used for hand tracking. As can be seen from Figure 7, there are four infrared cameras, which are the tracking room, hands, and controllers. The user's hands, their direction, and the arrangement of their fingers are detected by the headset. After being identified, computer vision algorithms are used to follow the hands' position and orientation. Moreover, pretrained algorithms allow using gestures, like Point and Pinch, Pinch and Scroll, and Palm Pinch, for executing different kinds of functions.

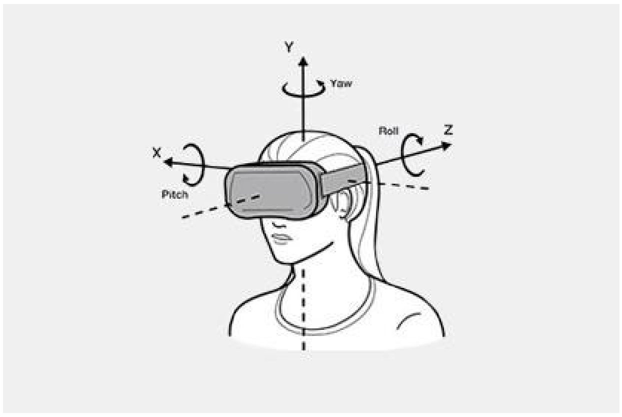


Figure 8
Oculus Quest 2 coordinate system [11]

2.2 Software Structure of the System

To achieve real-time performance, the software stack aims to reduce latency in the information transmission from the VR headset to the ROV. With this goal in mind, the major responsibility of the embedded system is to transmit data.

A two-way server-client connection allows the software architecture's two central nodes to communicate with one another. Figure 14 depicts how the primary elements of the two systems interact with one another. The first system is a program that is designed with Unity 3D, which includes the main functions of the Virtual Reality side. Turning left and right is done by manipulating the steering wheel and moving forward and backward is realized via using a similar wheel of transmission. For terminating the connection and turning off GPIO pins, a separate red button was created. The web client script written in C# sends GET requests of different types, like getting video packages asynchronously, or asking to stop the robot, to the server on Raspberry.

The second system is an application written on Python, which is run on Raspbian OS - the Raspberry's specific Unix system based on Debian. The application is responsible for sending answers to Unity and reacting according to their instructions. Commands can be divided into two groups, sending a video that is captured by the camera, and setting GPIO pins to low and high values. By changing the GPIO output, the server controls the car.

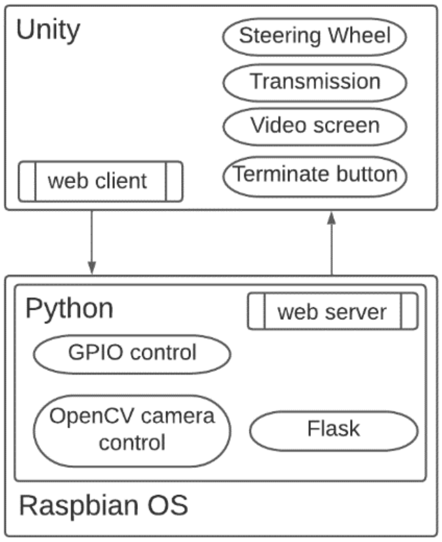


Figure 9
Software structure

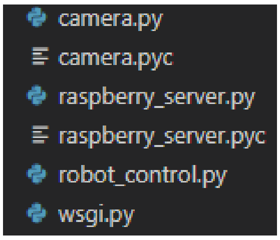
2.2.1 Python

Python is a general-purpose programming language, which has a huge number of libraries and frameworks. It works as an interpreter, which means that compiling time is more significant compared to other languages such as C++. Nevertheless, convenience, compatibility, and a wide list of opportunities make it the most appropriate choice for the project goals. There are two main libraries that are used in this project: Flask and OpenCV.

Flask is the web application framework of Python. The Jinja2 template engine and the Werkzeug WSGI toolkit are the foundations of Flask. Python web application development has traditionally used the Web Server Gateway Interface (WSGI) as a standard.

Open-source Camera Vision (OpenCV) is a large open-source framework for image processing, machine learning, and computer vision [20]. It can analyze pictures and videos to find faces, objects, and even real handwriting [20]. The number of opportunities rises when it is integrated with different libraries, such as Numpy, which is a highly optimized library for numerical operations. All operations that can be performed with Numpy can be combined with OpenCV. During the real-time video stream, it is much more efficient to send messages in byte form.

Three separate python files were created for a more understandable view of the solution. They can be seen in Figure 11, camera.py, raspberry_server.py and robot_control.py. The ***.pyc*** files are binaries that are generated during their usage.



```
camera.py
camera.pyc
raspberry_server.py
raspberry_server.pyc
robot_control.py
wsgi.py
```

Figure 10
List of files running on Raspberry Pi

In the robot control file special Robot class is created. It uses a built-in library of Raspberry Pi - RPi.GPIO for manipulating the pins. Four pins are attached to H-bridge and two of them are responsible for moving forward, the other two are for moving back. Based on these pins six functions are created, which are __init__, move_forward, move_reverse, turn_left, turn_right, stop, and terminate. The init function sets up the pins and prepares them to work. Straight movement is done by turning on forward and backward pins. While rotating is made by turning on one forward, and one backward pin. Terminate and stop functions are similar by turning off all pins but terminate also cleans their setup done by the init function, so the application has to restart the robot.

In the camera.py file, the PiVideoStran library is imported and the VideoCamera class is created. The initialization is done by setting up the camera. The important function here is get_frame, it gets a jpg image from the camera and transfers it to byte format via the imencode() function as it was mentioned above. Additionally, a flip function has been written for case when the image should be flipped, since in some cases the position of the camera cannot be changed easily physically, so here is a software solution is provided.

The main file that builds the server and merges by using the classes above is raspberry_server.py. It imports the Flask library and uses web server features for responding to requests and running Python commands. Here the IP address of the server is set, which can be LAN or WAN address. It means this server can be reached from any device that has access to the internet, not only members of a private Wi-Fi network.

The wsgi.py file is used for running the server with specific parameters. Activation of this server is included in daemons of RaspbianOS, meaning the server is active at the boot of Raspberry Pi 4 automatically.

2.2.3 Unity and Oculus SDK

A VR headset can interact with the ROV in the real world by using virtual environments created with the help of the Unity framework. A complete virtual command station with instructions for how it should move and fit together is created by creating several 3D objects in Unity. The control elements can be seen in Fig 16.

They were made manually in Unity by adding simple shapes together. The steering wheel and gear box are made of cylinder and cubes, and everything else is made of cubes.

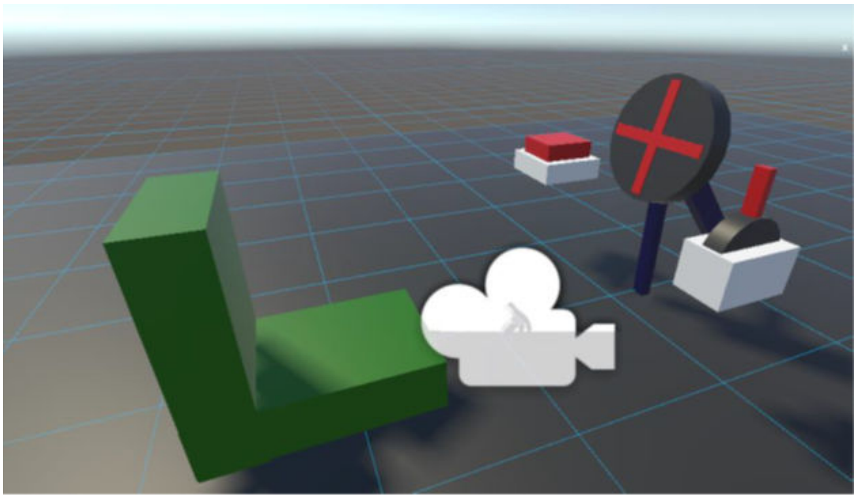


Figure 11
The virtual input controllers in Unity: steering wheel, transmission and terminate button

The next thing is a screen for showing video streams in real time. After investigation and testing, it was decided to stay with the simple 2D screen instead of 360 screens as the camera image looks better on this kind of screen.

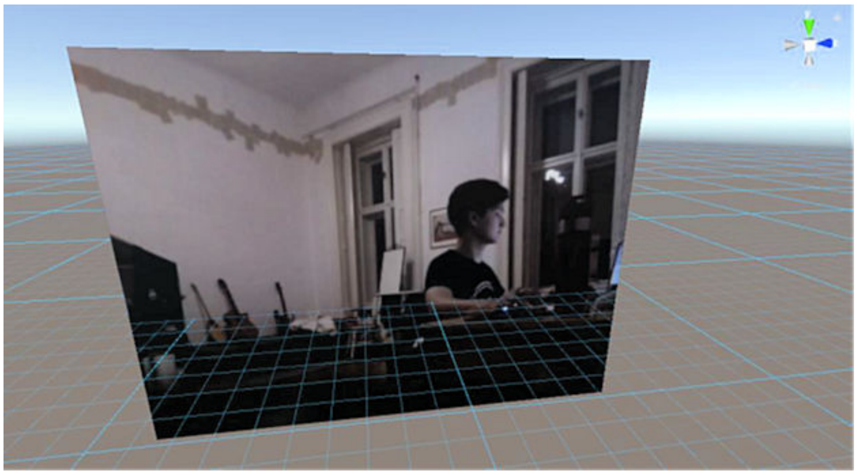


Figure 12
The live video screen in Unity

The Oculus SDK package is imported into the project along with the main Unity assets, delivering a variety of helpful scripts and objects to facilitate rapid and easy VR creation as well as a Player object that instantly links the VR headset to Unity. By adjusting a few parameters in the Unity interface, Oculus SDK scripts enable lever pulls and switch flips in addition to the headset connection. The configuration of the user's camera and input hardware input controllers is done with the SDK too.

Unity 3D provides tools for creating objects with methods and parameters. The CAD models of the objects can be imported or constructed manually in the app itself. Several scripts were created for configuration. Part of them is responsible for working with the web server and sending, and receiving packages, while another part is for the logic behind virtual objects such as steering wheel control. All of them are written in C# as it is the only language used in Unity.

3 Results

Several operators were asked for feedback in an effort to verify correct software operation and evaluate how user-friendly the VR interface is. The VR controls appeared to be simple to learn but challenging to master, according to early reviews. The tests were performed for two types of input controllers, joysticks, and bare hands. The stable Wi-Fi was used for testing networking and long-time connection between the operator and vehicle.

The rendering is done in 120 Frames per second (FPS) mode since there are two lenses and two screens. Double size of the load did not cause any problems, as designed models and functions are relatively lightweight.

The performance of the joystick controllers is assessed in Figures 13-14. To test the system's response, the steering wheel is grabbed with both hands and one hand simultaneously. Fortunately, joystick controller setup is robust and effective for these kinds of tasks. By pushing the controllers' side buttons, the grasping is executed. Nothing will happen if the wheel is outside the interaction range. The user only needs to push the joystick on top of the push button to activate the terminate function. Both the steering wheel and the gearbox wheel need to be grabbed and rotated.

The signal from the controllers has position data, that way headset can read the location of them even if they are not visible by the camera sensors.

In Figures 15-16 the hand controllers' performance is evaluated. The user's experience is unique, compelling, and practical. A suitable gesture is used to do the grabbing. The hand tracking effectively recognizes boundary lines since, for instance, pressing the terminate button may be done with the hands in any posture. The shape of the hands repeats real shape in exact same way, so size and features are same.

The hand input tests yielded lower results compared to the joystick input tests. Zero failures indicate that the system is operating efficiently, and that the application's logic is sound. Only the second table in this instance has errors since the test hand vanished during the test. The cameras fail identifying the hands because of unsuitable angles. For instance, it was unable to fully push the ending button. No answers during continuous connection depict missing packages.

Figure 13

The joystick inputs are tested by the operator (External view)

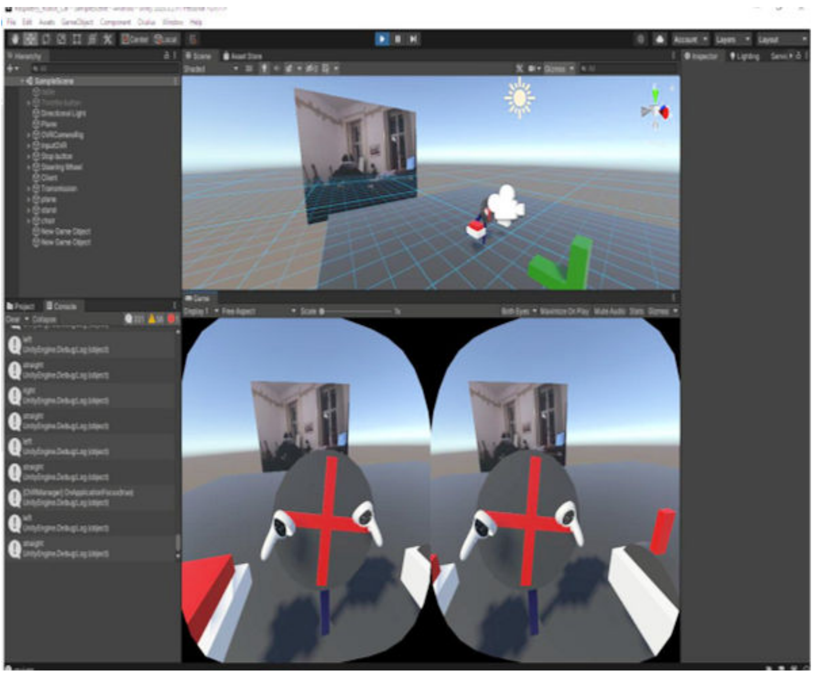


Figure 14
The joystick inputs are tested by the operator (Internal view)

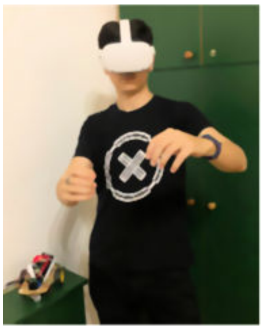


Figure 15
The hand inputs are tested by the operator (External view)

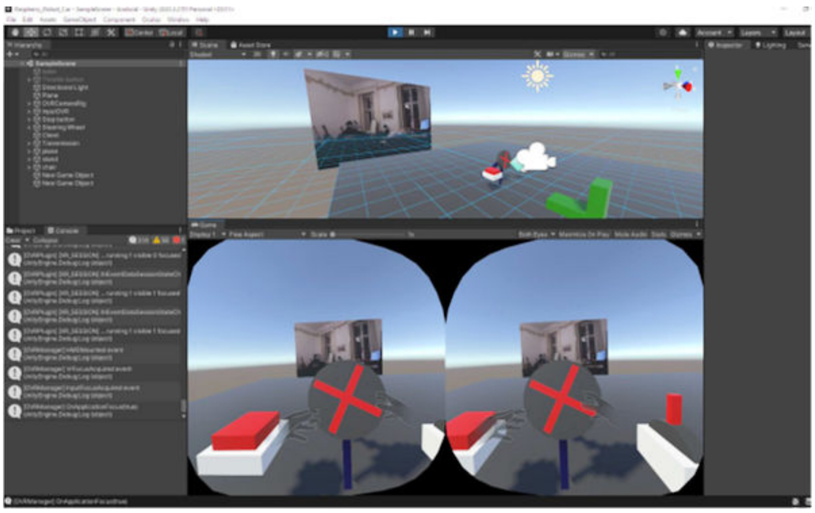


Figure 16
The hand inputs are tested by the operator (Internal view)

Conclusion and Future Work

The main goal of the project was accomplished. After studying the subject, a great deal of relevant literature was found. Considering various approaches, the an optimal solution was generated. The solution eliminates known limitations and continued development will increase the quality of the system.

The future work will be grouped into two methods. The first is handling existing limitations and uncomfortable features. During tests, it was difficult to turn the mobile robot to a specific degree. It can be solved by replacing the two DC motors with more advanced motors or use a slower rotation. Advanced motors can have several inputs from the GPIO, that will indicate the turning speed of the robot. This method solves the sensitivity issue. The second methodology is by adding new features that better fit this project. The computing power of the new Raspberry Pi 4 is not used to the full extent. It supports more complicated applications, such as, running recognition algorithms. For a mobile robot that operates on land, the path planning algorithm can be run by Raspberry at the same time as the current program. It will be able to avoid obstacles, do the video processing and draw on existing video transmissions. In addition, LED lights and Infra-Red sensors would improve remote operation, by allowing orientation at night and preventing bumps, by detecting objects near to robot.

References

[1] Amouri, L., Novales, C., Jallouli, M., Poisson, G., & Derbel, N., „An effective fuzzy-DVZ controller for an omnidirectional mobile robot”, International Multi-Conference on Systems, Sygnals & Devices, Chemnitz, Germany, 2012, doi:10.1109/ssd.2012.6197996

[2] Baklouti, E., Jallouli, M., Amouri, L., & Ben Amor, N., „Remote control of mobile robot through 3D virtual reality environment", ICBR conference, Sousse, Tunisia, 2013, doi:10.1109/icbr.2013.6729276

[3] Bucsai, S., Kucera, E., Haffner, O., & Drahos, P. „Control and monitoring of IoT devices using mixed reality developed by unity engine", K&I conference, Velke Karlovice, Czech Republic, 2020, doi:10.1109/ki48306.2020.9039901

[4] Candeloro, M., Valle, E., Miyazaki, M. R., Skjetne, R., Ludvigsen, M., & Sorensen, A. J, „HMD as a new tool for telepresence in underwater operations and closed-loop control of ROVs", OCEANS 2015 - MTS/IEEE Washington, Washington DC, 2015, doi:10.23919/oceans.2015.7404466

[5] Chin, C. S., Kamsani, N. B., Zhong, X., Cui, R., & Yang, C., „Unity3D serious game engine for high fidelity virtual reality training of remotely operated vehicle pilot" ICMIC conference, 2018, Guiyang. doi:10.1109/icmic.2018.8529900

[6] Cipresso, P., Giglioli, I. A. C., Raya, M. A., & Riva, G. „The past, present, and future of virtual and augmented reality research: A network and cluster analysis of the literature", Frontiers in Psychology, Vol. 9, 2018, doi:10.3389/fpsyg.2018.02086

[7] Datasheet of L298N https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf, (achieved: 2023,02,10)

[8] Datasheet of DC motors: https://www.auselectronicsdirect.com.au/assets/files/TA0132%20Specifications.pdf, (achieved: 2023,02,10)

[9] Dorneich, M., Christian, S., Whitlow, W., Rogers, K., & Feigh, E. „Adaptive user interface for semi-automatic operation", U.S. Patent, 8, 2015

[10] Guangsheng, Z., „Development of software and hardware complex for remote control of manipulation robot with virtual reality technology", StudNet, 5(6), 6073-6095, 2022

[11] Hoell, Markus & Heran, Nikolaus & Lepetit, Vincent, „Augmented Reality Oculus Rift", 2016

[12] Honig, W., Milanes, C., Scaria, L., Phan, T., Bolas, M., & Ayanian, N., „Mixed reality for robotics", IEEE/RSJ, IROS conference, Hamburg, Germany. 2015, doi:10.1109/iros.2015.7354138

[13] Jen, Y., Taha, Z., & Vui, L., „VR-Based Robot Programming and Simulation System for an Industrial Robot", International Journal of Industrial Engineering: Theory", IJIE, pp. 314-322, 2008

[14] Jones, Joseph L. & Flynn, Anita M., „Mobile robots: inspiration to implementation", Wellesley, Mass: A. K. Peters, 1993

[15] Kharoub, Hind & Lataifeh, Mohammed & Ahmed, Naveed, „3D User Interface Design and Usability for Immersive VR", Applied Sciences. Vol. 9, 4861, 10.3390/app9224861, 2019

[16] Kim, H., & Jung, S., „Control of a two-wheel robotic vehicle for personal transportation" Robotica, Vol. 34, pp. 1186-1208, 2016, doi:10.1017/S0263574714002173

[17] Martins, A., Almeida, J., Almeida, C., Matias, B., Kapusniak, S., & Silva, E., „EVA a hybrid ROV/AUV for underwater mining operations support", 2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO), Kobe, 2018, doi:10.1109/oceanskobe.2018.8558880

[18] Nguyen, L. A., Bualat, M., Edwards, L. J., Flueckiger, L., Neveu, C., Schwehr, K., Zbinden, E., „Virtual Reality Interfaces for Visualization and Control of Remote Vehicles", Autonomous Robots, Vol. 11, No. 1, pp. 59-68, 2001, doi:10.1023/a:1011208212722

[19] Oculus documentation for VR development. https://developer.oculus.com/documentation/ (achieved: 2023,02,10)

[20] OpenCV. OpenCV library reference. http://docs.opencv.org (achieved: 2023,02,10)

[21] Raspberry Pi 4 official documentation and specifications. https://www.raspberrypi.com (achieved: 2023,02,10)

[22] Rojo, A., Cortina, J., Sánchez, C. (2022) „Accuracy study of the Oculus Touch v2 versus inertial sensor for a single-axis rotation simulating the elbow's range of motion", Virtual Reality, https://doi.org/10.1007/s10055-022-00660-4 (achieved: 2023,02,10)

[23] Siegwart, R. Y., Nourbakhsh, I. R., & Scaramuzza, D. „Locomotion", in Introduction to Autonomous Mobile Robots, Cambridge, Massachusetts, MIT Press, 2004, sec 2, pp. 13-56

[24] Sutherland I. E. „The Ultimate Display, Multimedia: From Wagner to Virtual Reality", in Exploring human-computer interactions in virtual performance and learning in the context of rehabilitation, Frontiers in Virtual Reality, BMC Neurol, pp. 17-86, 1965, doi: 10.1186/s12883-017-0871-9

[25] Taheri, S., Matsushita, K., and Sasaki, M., „Virtual Reality Driving Simulation for Measuring Driver Behavior and Characteristics", Journal of Transportation Technologies. Vol. 7, pp. 123-132, 2017, doi: 10.4236/jtts.2017.72009

[26] Wang, S., Mao, Z., Zeng, C., Gong, H., Li, S., and Chen, B., "A new method of virtual reality based on Unity3D", 18th ICG conference, 2010, doi: 10.1109/GEOINFORMATICS.2010.5567608

[27] Xinyi, Z., „Building a battlefield simulation test environment based on Unity3D and VR technologies", Sidian University, 2020

[28] Simge U., Murat C. Y., Yasin Yagin, Hatice K., Volkan S. "A New Velocity Planning Method for Autonomous Robots with Varying Mass." Acta Polytechnica Hungarica, Vol. 19, No. 9, 2022, doi: 10.12700/APH.19.9.2022.9.9

[29] Thi T. M., Chyi-Yeu L., Nguyen G. H., Luong D. N., Pham C. H., Hoang H., "Hybrid SLAM-based Exploration of a Mobile Robot for 3D Scenario Reconstruction and Autonomous Navigation", Acta Polytechnica Hungarica, 2021, Vol. 18, No. 6, doi: 10.12700/APH.18.6.2021.6.11

[30] Sándor T. B., István Z. S., "Radio Frequency (RF)-based, Real-Time, Indoor Localization System for Unmanned Aerial Vehicles and Mobile Robot Applications." Acta Polytechnica Hungarica, Vol. 17, No. 9, 2020, doi: 10.12700/APH.17.9.2020.9.2

[31] Nagy I., "Hierarchical Mapping of an Electric Vehicle Sensor and Control Network", Acta Polytechnica Hungarica, Vol. 18, No. 9, pp. 161-180, 2021, doi: 10.12700/APH.18.9.2021.9.10