



Matteo Urthaler

Development of an autonomous underwater camera system for observing octopuses.

BACHELOR THESIS

to obtain the academic degree of
Bachelor of Science

Bachelor's programme
Software Engineering and Management

submitted to the
Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr. techn. Wolfgang Slany
Institute of Software Engineering and Artificial Intelligence

Graz, November 2025

STATEMENT ON FAITH

I declare on my honour that I have written this paper independently, have not used any sources or aids other than those indicated, and have identified any passages taken verbatim or in substance from the sources used as such.

Date, signature

Preface

This bachelor thesis was written at the Institute of Software Engineering and Artificial Intelligence at Graz University of Technology.

Special thanks go to Prof. Wolfgang Slany for making this thesis possible.

I would also like to express my sincere gratitude to my friends in Graz and South Tyrol.

Finally, my greatest thanks go to my entire family, especially my father Kurt, for his unconditional and motivating support throughout my academic career.

Abstract

This thesis outlines the development of an autonomous underwater camera system designed to observe octopuses. The aim was to create a system that would be able to record and store video footage independently once installed, while also providing power and data transmission via a cable.

To this end, a pressure-resistant housing was developed that incorporates a mini-PC, a battery, a switch, a monitor and various IP cameras, all of which are powered via Power over Ethernet. The software was programmed in C++. Performance monitoring and Gigabit Ethernet connections using RustDesk also ensure reliable transmission and secure remote access.

Tests confirm stable performance with low CPU and RAM usage, efficient storage, and a near-maximum Gigabit transfer rate. The energy and storage requirements enable realistic calculations for 24-hour operation to be made.

Table of contents

1	Introduction.....	- 1 -
1.1	Problem definition	- 1 -
1.2	Scientific question.....	- 1 -
1.3	Structure of the thesis	- 2 -
2	Fundamentals	- 3 -
2.1	Cameras and video transmission	- 3 -
2.2	Power supply via PoE	- 3 -
2.3	Mini PC as central control unit	- 3 -
2.4	Software and data formats.....	- 3 -
2.5	Performance and data transmission.....	- 4 -
3	Hardware implementation of the system	- 5 -
3.1	Cameras	- 5 -
3.2	Housing	- 7 -
3.2.1	Controller – Mini-PC.....	- 8 -
3.2.2	Battery	- 8 -
3.2.3	Switch	- 9 -
3.2.4	Monitor	- 10 -
3.2.5	PoE-Injector	- 11 -
4	Software implementation of the system.....	- 13 -
4.1	Camera Software.....	- 13 -
4.1.1	Network settings.....	- 13 -
4.1.2	Image settings	- 13 -
4.1.3	Motor settings	- 14 -
4.2	Mini-PC Software.....	- 15 -
4.2.1	Image recording.....	- 15 -
4.2.2	System load recording	- 20 -
4.3	Connection software	- 21 -
4.3.1	Data transmission	- 21 -
4.3.2	Data transfer speed	- 22 -
5	Results	- 23 -
6	Conclusion and outlook	- 25 -
7	References	- 26 -
8	List of figures	- 27 -

1 Introduction

„This is probably the closest we will come to meeting an intelligent alien.“
- Godfrey-Smith Peter [1]

This quote draws attention to the special role played by octopuses. These animals have long fascinated researchers, not least because of their surprisingly 'human' behaviour. They solve problems, respond to visual stimuli and demonstrate remarkable adaptability. It is precisely these characteristics that make them such an exciting subject for study.

The aim of this work is to develop a technical system that enables octopuses to be observed in their natural environment with as little influence as possible. Traditional methods such as diving or the use of simple underwater cameras quickly reach their limits. Underwater in particular, factors such as light, energy supply and data transmission pose special challenges that need to be overcome in a targeted manner.

1.1 Problem definition

Observing animals in their natural environment is often difficult. On land, autonomous systems such as wildlife cameras are already being used to reliably document animal movements over long periods of time. Underwater, however, conditions are far more challenging: poor lighting conditions, high pressure and limited energy and transmission options make it difficult to use conventional camera technology.

Conventional methods such as diver observations or simple underwater cameras quickly reach their limits, especially with octopuses, which are known for their intelligence and complex behaviour. There is a lack of a specially adapted, autonomous system that enables continuous and uninfluenced recording of their natural behaviour.

1.2 Scientific question

The aim of this work is to develop an automated system that, once installed, independently records and stores video footage of its surroundings. The system should be able to operate reliably underwater over a long period of time, making independent decisions about the recording process. Another important aspect is resource-saving operation in order to use the available energy as efficiently as possible.

The stored data should be accessible via a cable running from the seabed to the surface. The same cable should also be used to charge the battery to ensure continuous operation of the system.

To achieve this goal, the following scientific question was formulated:

'How can an autonomous, resource-saving camera system be developed that reliably records video underwater over a longer period of time, independently controls processes and ensures data and energy supply via a connecting cable to the surface?'

1.3 Structure of the thesis

Following the introduction, Chapter 2 presents the technical fundamentals essential for the development of the system. These include transmission protocols, libraries used, and the role of the mini-PC as the central control unit.

Chapter 3 describes the hardware implementation, from the cameras used to the housing, power supply, monitor, and network components. Chapter 4 deals with software development. The focus here is on the recording and control logic, performance monitoring, and solutions for data transmission and remote access.

The results of this work are presented in Chapter 5 and evaluated based on measurements of system performance, energy efficiency, and data transmission. Chapter 6 concludes with a summary and an outlook on possible further developments, particularly regarding practical tests and future enhancements.

2 Fundamentals

Various technical fundamentals are relevant for the development of an autonomous underwater camera system. These include the transmission and processing of video data as well as the power supply and operating system of the computer used.

2.1 Cameras and video transmission

The IP cameras used deliver their data via the following standardised protocols:

- RTSP (Real Time Streaming Protocol): Used for both industrial cameras. It is suitable for real-time transmission of video data and is widely used in the field of network cameras. [2]
- HTTP (Hypertext Transfer Protocol): Used in the project for data transmission from the homemade camera. It is universally applicable, but less optimised for real-time applications.

2.2 Power supply via PoE

Power over Ethernet (PoE) is used to simplify cabling. This involves transmitting data and power via the same Ethernet cable. This reduces the number of cable connections required and enables stable operation of the cameras underwater. [3]

2.3 Mini PC as central control unit

The mini-PC is the heart of the system. It handles the recording, storage and management of video data. Sufficient CPU power, RAM and SSD storage space are crucial for processing multiple video streams in parallel.

2.4 Software and data formats

The following tools are primarily used for software implementation:

- - C++ as the programming language for controlling the recording processes,
- - OpenCV for recording and storing video streams,
- - JSON (JavaScript Object Notation) as a lightweight configuration format in which camera information and recording parameters are stored.

2.5 Performance and data transmission

The Performance Monitor is used in Windows to monitor system performance. It allows key figures such as CPU load, RAM consumption and hard disk usage to be recorded at specified intervals.

Data is transferred to the interface using RustDesk software, which enables both remote access and file transfer. The network connection can be tested using iperf to ensure that a stable gigabit connection is available.

3 Hardware implementation of the system

The following chapters deal with the hardware implementation of the system. We begin with the general structure. This is divided into the part that is located directly in the water (cameras and housing) and the interior of the housing with the built-in technology.

3.1 Cameras

The cameras are the eyes of the system and are essential for gathering information in the form of video footage. They monitor the environment and deliver a constant video stream to the mini-PC.

The following three cameras were installed:

Barlus IP68 Underwater IP Camera

- Resolution 8Mp
- Lens 8mm
- 1920x1080
- Anti-Fouling wipers
- LED-lighting



Figure 1: Barlus IP68 Underwater IP Camera 1

Barlus IP68 Underwater IP Camera

- Resolution 8Mp
- Lens 3,6mm
- 1920x1080
- Anti-Fouling Wipers
- LED-lighting
- Infrared mode

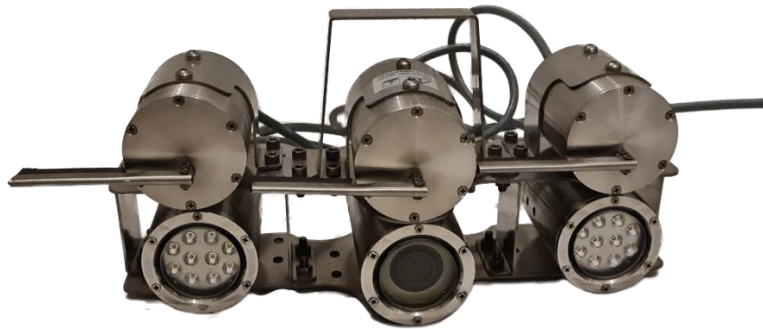


Figure 2: Barlus IP68 Underwater IP Camera 2

OctoWatch (Self-made camera by Thomas Ederer) [4-8]

- Resolution 2Mp
- 1920x1080
- Infrared mode



Figure 3: OctoWatch (self-made camera by Thomas Ederer)

For the two Barlus models, data is transferred to the mini-PC via an RTSP stream, whereas for the homemade camera, it is transferred via an HTTP stream.

All cameras are powered via Power over Ethernet (PoE), which is provided by a PoE injector in each housing. An Ethernet cable connected to the switch and a power cable for the power supply lead out of the injector. This means that only one cable is needed to connect the housing to the camera, supplying both power and data.

3.2 Housing

The housing forms the protective shell that surrounds the components necessary for the device to function. It must withstand the pressure underwater and houses the following component.

- Mini-PC
- Battery
- Switch
- Monitor
- PoE-Adapter

The wiring can be seen in the following illustration.

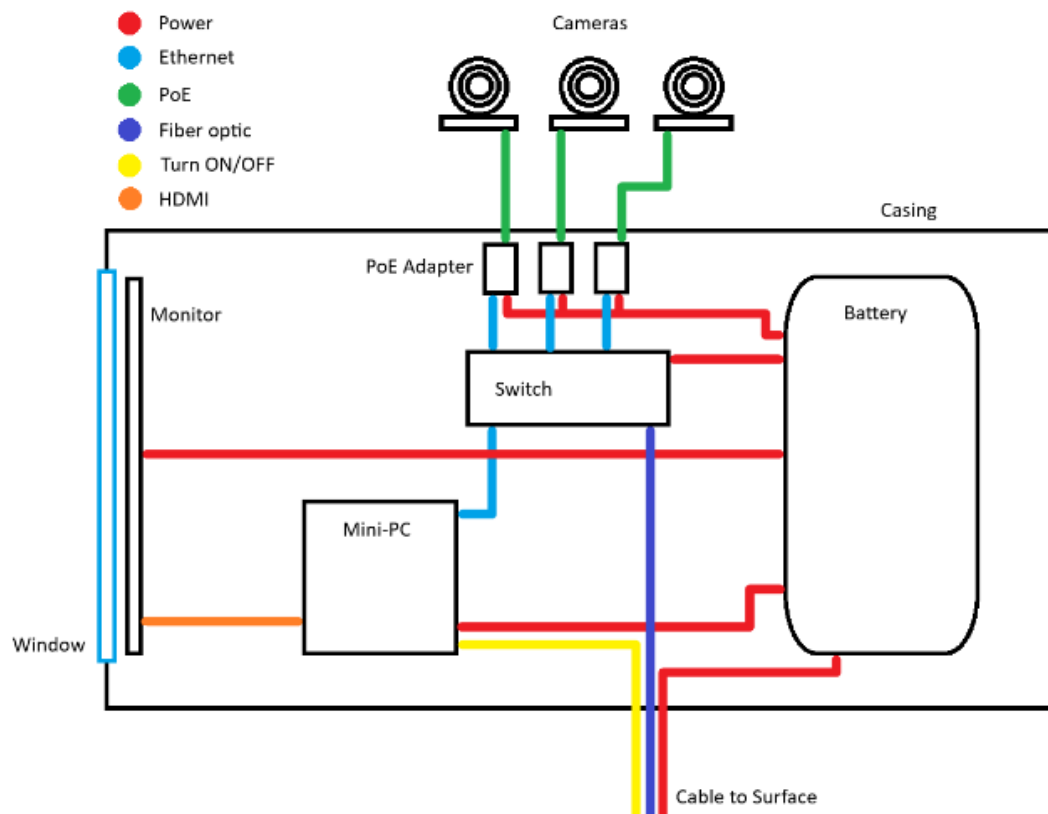


Figure 4: Schematic wiring of the components within the housing

The PoE cables for the cameras are routed out of the housing. In addition, a bundled cable is required that contains power, fibre optics and an on/off switch and must reach the surface. This can then be retrieved from the seabed to recharge the battery and transfer data.

3.2.1 Controller – Mini-PC

The mini-PC, or more precisely the 'ACEMAGICIAN AM06 Pro AMD Ryzen 7 Mini PC', has the following specifications:

- 32 GB RAM
- AMD Ryzen 7 5825U mit Radeon Graphics
- 512 GB SSD
- Windows 11 Pro



Figure 5: ACEMAGICIAN AM06 Pro AMD Ryzen 7 Mini PC (Source: ACEMAGIC)

This forms the heart of the system and is connected to the switch via an Ethernet cable at an RJ45 port. The image output is also connected to the monitor via an HDMI cable.

3.2.2 Battery

The battery can be freely selected, but it must have the necessary capacity to supply the system with power for a sufficiently long period of time.

3.2.3 Switch

The switch is a 'Davuaz 2.5G Ethernet network switch' with eight Ethernet ports and two SFP ports for fibre optic connection. The Ethernet ports have a data transfer rate of 2.5 Gbit/s, while the SFP ports have a data transfer rate of 10 Gbit/s.



Figure 6: Davuaz 2.5G Ethernet Network Switch (Source: Davuaz)

The switch is the central interface for data transmission. It connects the cameras to the mini-PC via Ethernet and the interface via SFP with fibre optics. An 'Optical Gigabit Dual Fibre Module' from the manufacturer 'Dengliquiong' is used to connect the fibre optic cable to the switch.



Figure 7: Optical gigabit dual fiber module (Source: Router-switch Ltd.)

3.2.4 Monitor

The monitor serves as a visual interface between the system and the octopuses. The model used is a 'UColor O2' from UPERFECT with the following specifications:

- 16-Inch-OLED-Monitor
- Resolution: 2880 x 1800 (3K)
- Aspect ratio: 16:10
- Contrast ratio: 100.000:1
- Colour space: 100% DCI-P3
- Colour depth: 10 Bit
- Brightness: 500 cd/m²
- Refresh rate: 120 Hz



Figure 8: UColor O2 (Source: UPERFECT)

These characteristics make the monitor particularly well suited for displaying detailed, high-contrast images. The high resolution ensures a realistic picture, while OLED technology and the extended colour space ensure clear images even in low light conditions. The high refresh rate also ensures smooth motion reproduction.

The decision to integrate a monitor into the system is based not only on technical considerations, but also on observations of octopus behaviour. As Montgomery describes, many aquarists have observed that octopuses show a strong interest in moving and colourful images and even enjoy watching television.

„Many home aquarists report that their octopuses appear to enjoy watching television with them. They particularly like sports and cartoons, with lots of movement and color ... King and her coauthor, Colin Dunlop, even suggest placing the tank in the same room as the TV, so owner and octopus can enjoy programs together.”[9]

When placed in front of the viewing window of the enclosure, the monitor can serve as a 'cinema' for the animals. It generates visual stimuli, enabling additional behavioural observations to be made.

3.2.5 PoE-Injector

A PoE (Power over Ethernet) injector is a device that supplies power to a terminal device via a network cable in addition to data. This allows IP cameras to be powered directly via the Ethernet cable without the need for a separate power cable.

Two different types of PoE injectors are used to supply power to the cameras.

Both Barlus models are equipped with a built-in passive PoE injector, which is powered by 24 V DC, 6.5 A via an external LRS-150 power supply unit from Mean Well.



Figure 9: Passive PoE injector of the Barlus cameras (Source: Barlus)



Figure 10: LRS-150 power supply (Source: Mean Well)

An ALL0488V6 PoE injector from Allnet is used for the DIY camera. This does not require an external power supply and is connected directly to the 230-volt mains supply.



Figure 11: ALL0488V6 PoE injector (Source: Allnet)

4 Software implementation of the system

The following chapters deal with the software implementation of the system. This is basically divided into three parts. There is the camera software, the mini-PC software and the receiver software.

4.1 Camera Software

The pre-installed software from the manufacturer is used for the Barlus cameras. The Raspberry Pi Foundation's 'rpikam-apps' software is used for the homemade camera. These are used to configure the following three points.

4.1.1 Network settings

To ensure that the network can access both cameras without any problems, a static IPv4 address must be set up in the settings.

For the Barlus devices, the IP address of the first camera was set to 192.168.1.88 and that of the second to 192.168.1.89.

The IP address for the DIY camera is 192.168.1.1 and therefore does not need to be changed.

4.1.2 Image settings

The image settings focus on producing a sharp image with low memory usage. This is achieved by the following settings:

A bit rate of 3,000 to 6,000 kbps is recommended for a resolution of 1080p and 30 FPS. [10]

The cameras produce 25 FPS at a resolution of 1080p. Since lighting conditions are poorer underwater, which increases image noise, the bit rate is fixed at the upper end at 6,000 kbps.

4.1.3 Motor settings

Biofouling refers to the undesirable accumulation and growth of biofilms, in this case specifically algae and other marine deposits. This phenomenon occurs in many situations and can be prevented by installing anti-fouling devices. [11]

The two Barlus cameras are each equipped with three motors that serve to remove algae growth (biofouling). They can be set to wipe twice across the camera's viewing window and lighting at specific intervals. The interval has been set to 30 minutes to save energy and guarantee a clear field of vision.

The homemade camera does not have an integrated anti-fouling device and is illuminated by an external UV-C LED to prevent algae growth. (Homemade by Thomas Ederer [4-8])



Figure 12: Anti-fouling UV-C LED

4.2 Mini-PC Software

The mini-PC forms the central processing unit of the system. It is responsible for image recording and utilisation recording.

4.2.1 Image recording

A programme was written in C++ to implement the image recording. It consists of:

- a JSON file containing information about the cameras.
- an absorption process that is responsible for effective absorption.
- a control process that starts and stops the recording process for each camera as soon as the specified recording time has been reached.

The basic procedure is as follows:

As soon as the 'Controller.exe' process is started, it reads all the necessary information from the JSON file and starts the recording process for each individual camera one after the other with a short time interval.

Once the maximum recording time has been reached, the recording is saved and the recording process is terminated. The controller registers that the previous recording has been completed and starts a new one.

This process repeats itself until the battery is empty and no further recordings can be made.

The following paragraphs describe the individual components in detail.

4.2.1.1 JSON-File

JSON (JavaScript Object Notation) is a text-based data format that is mainly used for data exchange between systems. It is both human-readable and easy for machines to process. Due to its structure of key-value pairs and lists, it is particularly well suited for representing configuration information. This makes it perfect for providing basic information for a programme.



```
1 {
2   "cameras": [
3     {
4       "name": "Cam1",
5       "active": 1,
6       "stream": "rtsp://192.168.1.88"
7     },
8     {
9       "name": "Cam2",
10      "active": 1,
11      "stream": "rtsp://192.168.1.89"
12     },
13     {
14       "name": "Cam3",
15       "active": 1,
16       "stream": "http://192.168.1.1/video"
17     }
18   ],
19   "info": {
20     "video_length_min": 60
21   }
22 }
```

Figure 13: JSON file with the camera information

The basic structure of the JSON file consists of an object containing two main elements:

- **'cameras'**: an array of objects. Each of these objects describes a camera with the properties **'name'** (name for saving the recordings), **'active'** (indicates whether this camera should make recordings) and **'stream'** (address of the stream).
- **'info'**: another object containing additional information, including **'video_length_min'**, which specifies the maximum length of the video recording in minutes.

In short: The outermost object consists of a list of cameras and a block containing general information.

4.2.1.2 Controller

The controller is a programme written in C++ that is responsible for starting and organising all recordings. It only uses the JSON file and therefore does not require any additional parameters.

At the start, the recording time and the list of all camera information are read from the JSON file. Then, a recording process is started for each active camera at three-second intervals. The camera information for the running cameras is also stored in a vector so that the individual processes can be checked later.

Once all camera processes are running, periodic checks are performed to determine whether a recording process has been terminated due to the maximum recording duration being reached or due to an error. In such cases, a new recording process is started for the respective camera, and the cycle begins again.


If a cancellation command (Ctrl + C event) has been registered by the user, no new recording process is started; instead, the system simply waits for the controlled cancellation of the current process. Once all recording processes have been completed, the controller process terminates itself.

4.2.1.3 Recorder

The recorder is also a programme written in C++ that is responsible for processing and storing the camera streams. It does not use the JSON file but obtains the necessary information from the parameters passed when the process is started. The OpenCV library is used for recording.

If the programme starts correctly, the folders for storing the video files are first created, if they do not already exist. To ensure that the video files can be easily organised later, a timestamp is created using the ctime-library, which is used to name the video files.

Next, the availability of the video stream, the resolution and the frames per second are checked. From this point onwards, the actual recording process can begin. This was created using the official documentation from the OpenCV organisation. [12]



```
1  int frame_count = 0;
2  int max_frames = static_cast<int>(fps * RECORD_SECONDS);
3  Mat frame;
4  while (frame_count < max_frames && !stop_requested)
5  {
6      cap >> frame;
7
8      if (frame.empty())
9      {
10         cout << NAME << " error: Failed to grab frame." << endl;
11         break;
12     }
13
14     writer.write(frame);
15     frame_count++;
16 }
```

Figure 14: Code piece of the recording process

At the start, the VideoCapture function captures a frame from the stream. This is written to the storage medium using the VideoWriter function and the number of captured frames is increased by one. This process is repeated for a number of 'max_frames' times, where 'max_frames' is calculated by multiplying the FPS and the specified recording time in seconds.

If no frame can be captured or a cancellation command is received, the recording made so far is saved and the process is terminated. In the first case, the controller would start a new recording process. In the event of cancellation, the current video file is saved and the process is terminated.

4.2.2 System load recording

To obtain an overview of how the PC performs during continuous recording, a performance log is created when the device is started up.

The Windows Performance Monitor is used for this purpose. This is a tool for monitoring and analysing system performance. A 'Data Collector Set' is created for this purpose, to which performance indicators for CPU utilisation, memory (available MBytes) and hard disk usage (% disk time, % free space) are added. The collector interval for recording has been set to 15 seconds.

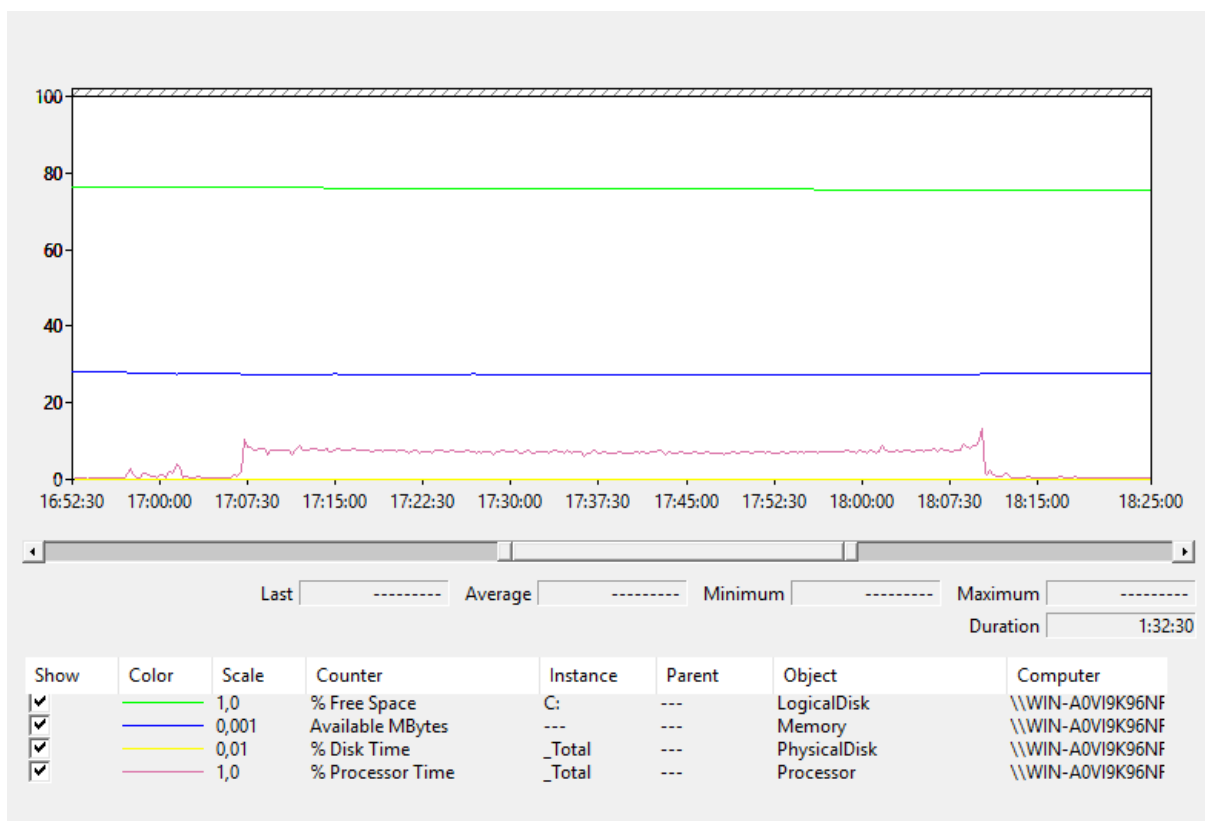


Figure 15: System Load Recording (Performance Monitor)

Here, the utilisation of a one-hour recording was recorded. The recording was started at 5:07 p.m. You can see how the processor time rises to 10% and stabilises at this level until the end of the recording. The working memory remains relatively constant at 30%, which shows that there is sufficient working memory available and that the recording does not lead to bottlenecks in the RAM. In terms of free memory, you can see that it decreases by a few percent between the start and end of the recording. This is due to the video being saved.

Overall, it can be said that the PC offers more than enough performance for the given setup. This also leaves open the possibility of adding further cameras in the future.

4.3 Connection software

To establish a connection to the network underwater, a few settings had to be configured. The mini-PC has two Ethernet ports. One of these is used to connect it to the switch. To ensure that it can be easily identified, both Ethernet ports were assigned a static IPv4 address. These are 192.168.1.101 for the 'Realtek PCIe GbE Family Controller' and 192.168.1.102 for the 'Intel(R) Ethernet Controller I226-V'.

RustDesk is used to connect to the interface. RustDesk is an open-source solution for remote access and remote control of computers. It enables access to user interfaces via the local network without the need for a physical connection or direct presence. The advantage over commercial alternatives such as TeamViewer or AnyDesk lies in particular in the openness of the source code, the possibility of self-hosting the server components, and independence from centralised cloud services.

This allows settings to be adjusted, the status of the system to be checked and maintenance work to be carried out without having to retrieve the device from its underwater environment. Underwater, RustDesk always starts automatically and is set up so that anyone who knows the password can connect to it from the surface.

4.3.1 Data transmission

Data transfer also takes place via RustDesk. Once the connection is established, data can be copied using drag-and-drop or RustDesk's data transfer function.

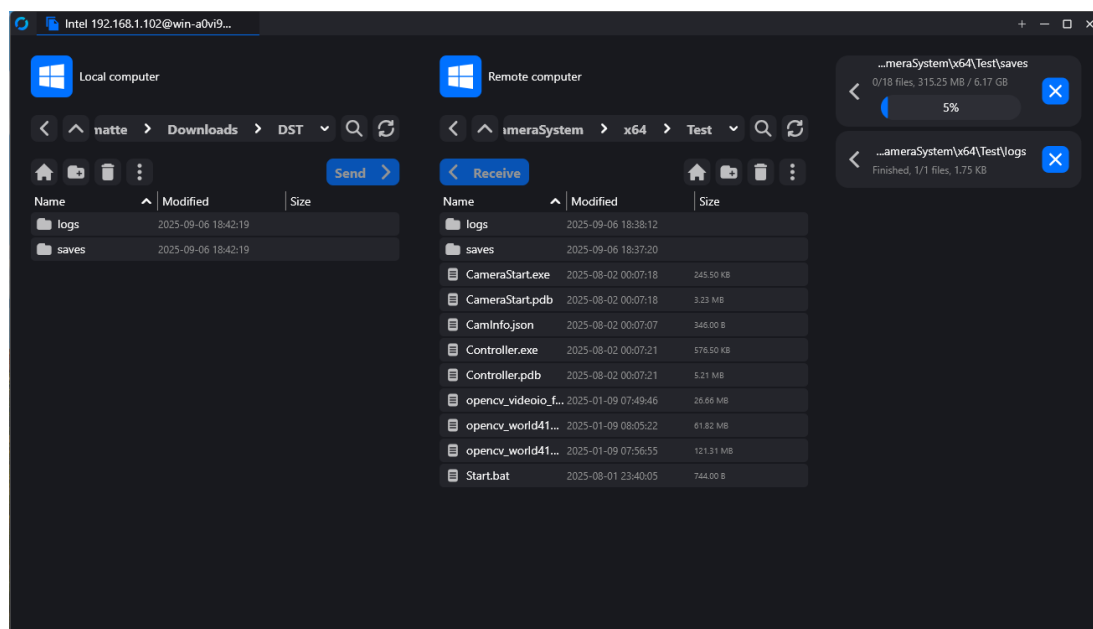


Figure 16: Data transfer in RustDesk

4.3.2 Data transfer speed

Iperf is used to measure data transfer speeds. It is a reliable tool for checking the available bandwidth and throughput of a network connection. In this case, the connection between the mini-PC and the PC on the surface was tested, via which the stored video files are transferred.

```
C:\iperf3.19_64>iperf3 -c 192.168.1.57
Connecting to host 192.168.1.57, port 5201
[ 5] local 192.168.1.102 port 49674 connected to 192.168.1.57 port 5201
[ ID] Interval           Transfer     Bitrate
[ 5]  0.00-1.01   sec    115 MBytes   950 Mbits/sec
[ 5]  1.01-2.00   sec    112 MBytes   948 Mbits/sec
[ 5]  2.00-3.02   sec    114 MBytes   948 Mbits/sec
[ 5]  3.02-4.01   sec    112 MBytes   949 Mbits/sec
[ 5]  4.01-5.01   sec    113 MBytes   949 Mbits/sec
[ 5]  5.01-6.01   sec    113 MBytes   949 Mbits/sec
[ 5]  6.01-7.00   sec    112 MBytes   948 Mbits/sec
[ 5]  7.00-8.01   sec    114 MBytes   949 Mbits/sec
[ 5]  8.01-9.01   sec    112 MBytes   946 Mbits/sec
[ 5]  9.01-10.01  sec    113 MBytes   948 Mbits/sec
-- -- -- -- --
[ ID] Interval           Transfer     Bitrate
[ 5]  0.00-10.01  sec    1.11 GBytes   948 Mbits/sec
[ 5]  0.00-10.01  sec    1.11 GBytes   948 Mbits/sec
sender
receiver
iperf Done.
```

Figure 17: Transfer speed measurement with iperf

The test showed a stable throughput of around 948 Mbit/s over a period of ten seconds, which is almost the maximum performance of a gigabit connection. This ensures that the recorded video data can be transmitted to the surface efficiently and without bottlenecks.

5 Results

Various tests were carried out as part of the evaluation of the developed system. A key aspect of the investigation was to check the stability during longer video recordings. Analysis using Windows Performance Monitor showed that system utilisation remained at a consistently low level during the one-hour test period. The measurements showed a constant processor utilisation of around 10%, while the RAM was utilised at around 30%. Hard disk usage was also moderate, always ensuring reliable storage of video data. The results show that the system has sufficient performance reserves to process multiple video streams in parallel and that upgrades are possible.

Furthermore, data transmission to the surface was tested. The measurement showed a stable throughput of approximately 948 Mbit/s. This corresponds almost exactly to the maximum performance of a gigabit connection. This ensures efficient and bottleneck-free transmission of the recorded video data.

In addition, the system's power and memory consumption was measured over a period of one hour. With the light deactivated, consumption was measured at 0.065 kWh, while with the light activated, a value of 0.140 kWh was achieved. Memory consumption amounts to approximately 3.6 GB per hour.

The measurements show that lighting has a significant impact on energy consumption.

Based on these values, the requirement for 24-hour operation with realistic lighting conditions can also be calculated. It was assumed that lighting would be required for around eight hours a day, while it would remain switched off for the remaining 16 hours.

Daily energy consumption

$$(8 * 0.140 \text{ kWh}) + (16 * 0.065 \text{ kWh}) = 2.16 \text{ kWh}$$

Memory usage per day

$$3.6 \text{ GB} * 24 = 86.4 \text{ GB}$$

Based on the determined data rate of approx. 948 Mbit/s, the copying time for the stored files can also be estimated.

Copying time

$$86,4 \text{ GB} = 691.200 \text{ Mbit}$$

$$691,200 \text{ Mbit} \div 948 \text{ Mbit/s} = 729 \text{ s} \approx 12.15 \text{ min}$$

The analysis shows that the daily storage requirement of 86.4 GB results in a transfer time of approximately 12 minutes. This ensures that even large amounts of data can be transferred to the surface within a reasonable time.

Overall, the results confirm that the system operates reliably in terms of both performance and data transfer. The measured power and memory consumption as well as the copying

time also provide valuable information for planning energy supply, storage capacities and data management.

6 Conclusion and outlook

As part of this work, a concept for an underwater camera system was developed that will enable the observation of octopuses in their natural environment. The results show that the technology used has sufficient performance for the parallel recording of multiple video streams and that fast and stable data transmission via Gigabit Ethernet is feasible. In addition, power and memory consumption were recorded, enabling an initial assessment of the requirements for power supply and storage capacity.

The system presented is currently only available as a theoretical model and has not yet been tested in a real underwater environment. The next step is therefore to conduct practical trials at sea in order to test its resilience and reliability under real conditions.

In future work, the system could be expanded to include automatic image analysis, additional sensors or optimisation of energy consumption. This would increase both the scientific benefit and technical efficiency and enable long-term, uninfluenced observation of octopus behaviour.

7 References

- [1] Godfrey-Smith, Peter, „*Other Minds: The Octopus, the Sea, and the Deep Origins of Consciousness.*“, New York 2016, S. 13.
- [2] cf. Schulzrinne, H., Rao, A., and R. Lanphier, „*Real Time Streaming Protocol (RTSP)*“, RFC 2326, DOI 10.17487/RFC2326, April 1998, <https://www.rfc-editor.org/info/rfc2326>
- [3] cf. Mendelson, Galit. „*All you need to know about Power over Ethernet (PoE) and the IEEE 802.3 af Standard.*“, PowerDsine, 2004
- [4] Ederer, Thomas, „*OctoWatch: Umsetzung und Evaluierung einer kostengünstigen Unterwasserkamera zur Beobachtung von Oktopussen.*“, 2024
- [5] Ederer, Thomas, Underwater Camera Project (Anleitung zum Nachbau von OctoWatch), <https://underwater-camera-project.github.io/> [access: November 2025]
- [6] Ederer Thomas, „*OctoWatch-Video Service*“, GitHub Repository, 2025, <https://github.com/tederer/octowatch-videoservice/tree/v0.4.0> [access: November 2025]
- [7] Ederer, Thomas; Ivkić Igor, „*Implementing video monitoring capabilities by using hardware-based encoders of the Raspberry Pi Zero 2 W.*“, SoftwareX, Volume 31, 2025, 102274. ISSN 2352-7110, <https://doi.org/10.1016/j.softx.2025.102274> [access: November 2025]
- [8] Ederer, Thomas; Slany, Wolfgang; Ivkić, Igor. „*Reducing Underwater Observation Costs by Leveraging Cloud Technology.*“, In: Arabnia, H.R.; Deligiannidis, L.; Shenavarmasouleh, F.; Amirian, S.; Ghareh Mohammadi, F. (Hrsg.): Computational Science and Computational Intelligence (CSCI 2024). Communications in Computer and Information Science, Band 2508. Springer, Cham, 2025, https://doi.org/10.1007/978-3-031-95133-6_14 [access: November 2025]
- [9] Montgomery, Sy, „*The Soul of an Octopus: A Surprising Exploration into the Wonder of Consciousness.*“, New York u. a. 2015, S. 68.
- [10] Sydney Roy (Whalen), „*What Is Video Bitrate (And What Bitrate Should You Use)? (Update)*“, Wowza Blog, 2025, <https://www.wowza.com/blog/what-is-video-bitrate-and-what-bitrate-should-you-use> [access: September, 2025]
- [11] Flemming, HC, „*Biofouling in water systems – cases, causes and countermeasures.*“, Appl Microbiol Biotechnol 59, 2002, S. 629–640. <https://doi.org/10.1007/s00253-002-1066-9>
- [12] Moukthika, „*Reading and Writing Videos using OpenCV*“, OpenCV Blog, 2025, <https://opencv.org/blog/reading-and-writing-videos-using-opencv/> [access: September, 2025]

8 List of figures

Figure 1: Barlus IP68 Underwater IP Camera 1	- 5 -
Figure 2: Barlus IP68 Underwater IP Camera 2	- 6 -
Figure 3: OctoWatch (self-made camera by Thomas Ederer)	- 6 -
Figure 4: Schematic wiring of the components within the housing	- 7 -
Figure 5: ACEMAGICIAN AM06 Pro AMD Ryzen 7 Mini PC (Source: ACEMAGIC)	- 8 -
Figure 6: Davuaz 2.5G Ethernet Network Switch (Source: Davuaz)	- 9 -
Figure 7: Optical gigabit dual fiber module (Source: Router-switch Ltd.)	- 9 -
Figure 8: UColor O2 (Source: UPERFECT)	- 10 -
Figure 9: Passive PoE injector of the Barlus cameras (Source: Barlus)	- 11 -
Figure 10: LRS-150 power supply (Source: Mean Well)	- 11 -
Figure 11: ALL0488V6 PoE injector (Source: Allnet)	- 12 -
Figure 12: Anti-fouling UV-C LED	- 14 -
Figure 13: JSON file with the camera information	- 16 -
Figure 14: Code piece of the recording process	- 18 -
Figure 15: System Load Recording (Performance Monitor)	- 20 -
Figure 16: Data transfer in RustDesk	- 21 -
Figure 17: Transfer speed measurement with iperf	- 22 -