Matteo Urthaler

Entwicklung eines autonomen Unterwasserkamerasystems zur Beobachtung von Oktopussen.

BACHELORARBEIT

zur Erlangung des akademischen Grades
Bachelor of Science

Bachelorstudium
Software Engineering and Management

eingereicht an der
Technischen Universität Graz

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

Graz, November 2025

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Datum, Unterschrift

Vorwort

Die vorliegende Bachelorarbeit wurde am Institute of Software Engineering and Artificial Intelligence der Technischen Universität Graz verfasst.

Besonderer Dank gebührt Herrn Univ.-Prof. Dipl.-Ing. Dr. techn. Wolfgang Slany für die Ermöglichung dieser Arbeit.

Ein herzliches Dankeschön gilt darüber hinaus meinen Freunden aus Graz und Südtirol.

Mein größter Dank gilt abschließend meiner ganzen Familie, besonders meinem Vater Kurt für die bedingungslose und motivierende Unterstützung auf meinem akademischen Weg.

Kurzfassung

In dieser Arbeit wird die Entwicklung eines autonomen Unterwasserkamerasystems zur Beobachtung von Oktopussen beschrieben. Das Ziel war ein System, das nach der Installation selbstständig Videoaufnahmen erstellt, speichert und über ein Kabel sowohl die Energieversorgung als auch die Datenübertragung gewährleistet.

Zu diesem Zweck wurde ein druckfestes Gehäuse mit Mini-PC, Akku, Switch, Monitor sowie verschiedenen IP-Kameras, die über Power over Ethernet betrieben werden, entwickelt. Die Software wurde in C++ programmiert. Ergänzend sorgen Performance-Monitoring und Gigabit-Ethernet-Verbindungen mittels RustDesk für eine zuverlässige Übertragung und einen sicheren Fernzugriff.

Tests bestätigen eine stabile Leistung mit geringer CPU- und RAM-Auslastung, eine effiziente Speicherung und eine nahezu maximale Gigabit-Übertragungsrate. Der Energie- und Speicherbedarf erlaubt eine realistische Kalkulation für den 24-Stunden-Betrieb.

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.

Inhaltsverzeichnis

1 Einleitung

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

Mit diesem Zitat wird auf die besondere Rolle der Oktopusse aufmerksam gemacht. Diese Tiere faszinieren die Forschung seit Langem – nicht zuletzt aufgrund ihres überraschend „menschlichen" Verhaltens. Sie lösen Aufgaben, reagieren auf visuelle Reize und zeigen eine bemerkenswerte Anpassungsfähigkeit. Genau diese Eigenschaften machen sie zu einem spannenden Untersuchungsobjekt.

Ziel dieser Arbeit ist die Entwicklung eines technischen Systems, das eine möglichst unbeeinflusste Beobachtung von Oktopussen in ihrer natürlichen Umgebung ermöglicht. Klassische Methoden wie das Tauchen oder der Einsatz einfacher Unterwasserkameras stoßen hierbei schnell an ihre Grenzen. Gerade unter Wasser stellen Faktoren wie Licht, Energieversorgung und Datenübertragung besondere Herausforderungen dar, die es gezielt zu bewältigen gilt.

1.1 Problemstellung

Die Beobachtung von Tieren in ihrer natürlichen Umgebung ist oft schwierig. An Land kommen dafür bereits autonome Systeme wie Wildkameras zum Einsatz, die Tierbewegungen zuverlässig und über längere Zeiträume dokumentieren. Unter Wasser hingegen sind die Bedingungen weitaus anspruchsvoller: Schlechte Lichtverhältnisse, hoher Druck sowie begrenzte Energie- und Übertragungsmöglichkeiten erschweren den Einsatz klassischer Kameratechnik.

Gerade bei Oktopussen, die für ihre Intelligenz und ihr komplexes Verhalten bekannt sind, stoßen herkömmliche Methoden wie Taucherbeobachtungen oder einfache Unterwasserkameras schnell an ihre Grenzen. Es fehlt ein speziell angepasstes, autonomes System, das eine kontinuierliche und möglichst unbeeinflusste Aufzeichnung ihres natürlichen Verhaltens ermöglicht.

1.2 Wissenschaftliche Fragestellung

Das Ziel dieser Arbeit ist die Entwicklung eines automatisierten Systems, das nach einer einmaligen Installation selbstständig Videoaufnahmen der Umgebung anfertigt und speichert. Das System soll über einen längeren Zeitraum hinweg zuverlässig unter Wasser betrieben werden können und dabei eigenständig Entscheidungen über den Aufnahmeprozess treffen. Ein weiterer wichtiger Aspekt ist ein ressourcensparender Betrieb, um die verfügbare Energie möglichst effizient zu nutzen.

Die gespeicherten Daten sollen über ein Kabel, das vom Meeresboden zur Oberfläche führt, ausgelesen werden können. Über dasselbe Kabel soll zudem eine Aufladung des Akkus erfolgen, um den kontinuierlichen Einsatz des Systems sicherzustellen.

Um dieses Ziel zu erreichen, wurde folgende wissenschaftliche Fragestellung formuliert:

„Wie kann ein autonomes, ressourcensparendes Kamerasystem entwickelt werden, das unter Wasser über einen längeren Zeitraum zuverlässig Videoaufnahmen erstellt, eigenständig Prozesse steuert und die Daten sowie die Energieversorgung über ein Verbindungskabel zur Oberfläche sicherstellt?"

1.3 Aufbau der Arbeit

Nach der Einleitung folgt in Kapitel 2 eine Darstellung der für die Entwicklung des Systems wesentlichen technischen Grundlagen. Dazu gehören Übertragungsprotokolle, eingesetzte Bibliotheken und die Rolle des Mini-PCs als zentrale Steuereinheit.

In Kapitel 3 wird die hardwaretechnische Umsetzung beschrieben, von den eingesetzten Kameras über das Gehäuse bis hin zur Energieversorgung, zum Monitor und zu den Netzwerkkomponenten. Kapitel 4 behandelt die Softwareentwicklung. Der Fokus liegt hier auf der Aufnahme- und Steuerungslogik, der Leistungsüberwachung sowie den Lösungen für Datenübertragung und Fernzugriff.

Die Ergebnisse dieser Arbeit werden in Kapitel 5 präsentiert und anhand von Messungen zur Systemleistung, Energieeffizienz und Datenübertragung bewertet. Kapitel 6 schließt mit einer Zusammenfassung und einem Ausblick auf mögliche Weiterentwicklungen, insbesondere hinsichtlich praktischer Tests und zukünftiger Erweiterungen.

2 Grundlagen

Für die Entwicklung eines autonomen Unterwasserkamerasystems sind verschiedene technische Grundlagen relevant. Dazu zählen die Übertragung und Verarbeitung von Videodaten ebenso wie die Stromversorgung und das Betriebssystem des eingesetzten Rechners.

2.1 Kameras und Videoübertragung

Die eingesetzten IP-Kameras liefern ihre Daten über folgende standardisierte Protokolle:

- RTSP (Real Time Streaming Protocol): Wird für die beiden industriellen Kameras verwendet. Es eignet sich für die Echtzeitübertragung von Videodaten und ist im Bereich der Netzwerkkameras weit verbreitet. [2]
- HTTP (Hypertext Transfer Protocol): Wird im Projekt zur Datenübertragung der Eigenbaukamera verwendet. Es ist universell einsetzbar, jedoch weniger für Echtzeitanwendungen optimiert.

2.2 Energieversorgung mittels PoE

Zur Vereinfachung der Verkabelung kommt „Power over Ethernet" (PoE) zum Einsatz. Dabei werden Daten und Strom über dasselbe Ethernet-Kabel übertragen. Dadurch reduziert sich die Anzahl der benötigten Kabelverbindungen und ein stabiler Betrieb der Kameras unter Wasser wird ermöglicht. [3]

2.3 Mini-PC als zentrale Steuereinheit

Der Mini-PC ist das Herzstück des Systems. Er übernimmt die Aufnahme, Speicherung und Verwaltung der Videodaten. Eine ausreichende Ausstattung mit CPU-Leistung, Arbeitsspeicher und SSD-Speicherplatz ist dabei entscheidend, um mehrere Videostreams parallel verarbeiten zu können.

2.4 Software und Datenformate

Für die Softwareumsetzung werden vor allem die folgenden Werkzeuge genutzt:

- – C++ als Programmiersprache zur Steuerung der Aufnahmeprozesse,
- - OpenCV zur Aufnahme und Speicherung von Videostreams,
- - JSON (JavaScript Object Notation) als leichtgewichtiges Konfigurationsformat, in dem Kamerainformationen und Aufnahmeparameter abgelegt werden.

2.5 Performance- und Datenübertragung

Zur Überwachung der Systemleistung wird unter Windows der Performance Monitor verwendet. Damit lassen sich Kennzahlen wie CPU-Last, RAM-Verbrauch und Festplattennutzung in festgelegten Intervallen aufzeichnen.

Die Datenübertragung zur Oberfläche erfolgt über die Software RustDesk, die sowohl Fernzugriff als auch Dateitransfer ermöglicht. Mithilfe von iperf kann die Netzwerkverbindung getestet werden, um sicherzustellen, dass eine stabile Gigabit-Verbindung vorliegt.

3 Hardwaretechnische Umsetzung des Systems

Die folgenden Kapitel befassen sich mit der hardwaretechnischen Umsetzung des Systems. Am Anfang steht der generelle Aufbau. Dieser teilt sich in den Teil, der sich direkt im Wasser befindet (Kameras und Gehäuse), und das Innenleben des Gehäuses mit der verbauten Technik.

3.1 Kameras

Die Kameras sind die Augen des Systems und für die Informationsgewinnung in Form von Videomaterial unerlässlich. Sie überwachen die Umgebung und liefern einen konstanten Videostream an den Mini-PC.

Es wurden die folgenden drei Kameras verbaut:

Barlus IP68 Underwater IP Kamera

- Auflösung 8Mp
- Linse 8mm
- 1920x1080
- Anti-Fouling Scheibenwischer
- LED-Beleuchtung

Abbildung 1: Barlus IP68 Underwater IP Kamera 1

Barlus IP68 Underwater IP Kamera

- Auflösung 8Mp
- Linse 3,6mm
- 1920x1080
- Anti-Fouling Scheibenwischer
- LED-Beleuchtung
- Infrarot Modus

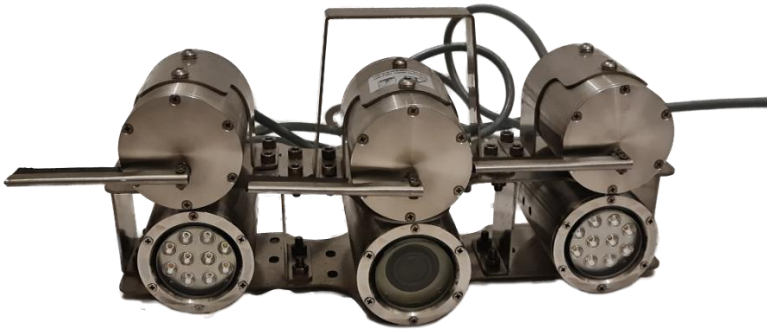

Abbildung 2: Barlus IP68 Underwater IP Kamera 2

OctoWatch (Eigenbau Kamera von Thomas Ederer) [4-8]

- Auflösung 2Mp
- 1920x1080
- Infrarot Modus

Abbildung 3: OctoWatch (Eigenbau Kamera von Thomas Ederer)

Für die beiden Barlus-Modelle erfolgt die Datenübertragung an den Mini-PC mittels eines RTSP-Streams, für die Eigenbau-Kamera hingegen mittels HTTP-Stream.

Alle Kameras werden über Power over Ethernet (PoE) versorgt, was durch jeweils einen PoE-Injektor im Gehäuse geschieht. Aus diesem führen ein Ethernet-Kabel, welches mit dem Switch verbunden ist, sowie ein Stromkabel, welches zur Energieversorgung dient. Dadurch wird nur ein Kabel benötigt, das aus dem Gehäuse zur Kamera führt und sowohl Strom als auch Daten liefert.

3.2 Gehäuse

Das Gehäuse bildet die schützende Hülle, welche die für die Umsetzung der Funktion des Geräts notwendigen Komponenten umgibt. Es muss dem Druck unter Wasser standhalten und beherbergt die folgenden Komponenten.

- Mini PC
- Akku
- Switch
- Monitor
- PoE-Adapter

Die Verkabelung ist in der folgenden Abbildung zu sehen.

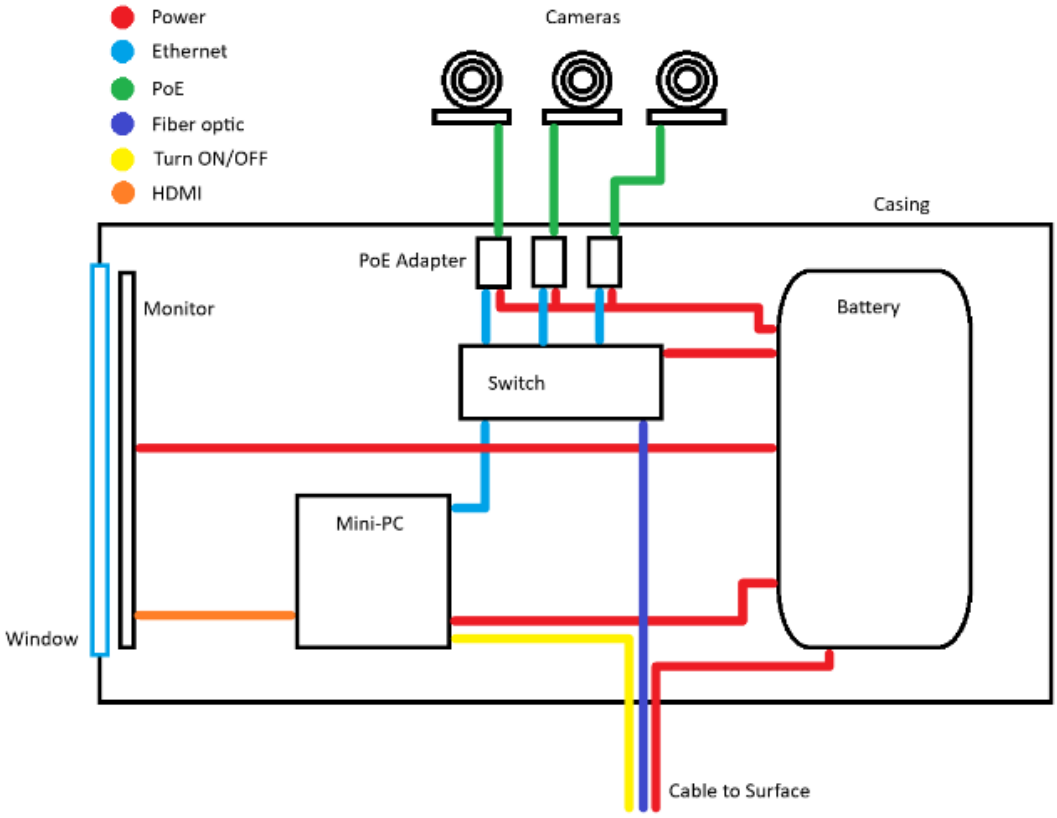

Abbildung 4: Schematische Verkabelung der Bauteile innerhalb des Gehäuses

Aus dem Gehäuse werden die PoE-Kabel für die Kameras herausgeführt. Zudem ist ein gebündeltes Kabel erforderlich, das Strom, Glasfaser und einen An-/Aus-Schalter enthält und bis zur Oberfläche reichen muss. Dieses kann dann zum Aufladen des Akkus und zur Datenübertragung vom Meeresboden heraufgeholt werden.

3.2.1 Controller - Mini PC

Der Mini PC, genauer gesagt der "ACEMAGICIAN AM06 Pro AMD Ryzen 7 Mini PC", weist folgende Spezifikationen auf:

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

Abbildung 5: ACEMAGICIAN AM06 Pro AMD Ryzen 7 Mini PC (Quelle: ACEMAGIC)

Dieser bildet das Herzstück des Systems und ist per Ethernet-Kabel mit dem Switch an einem RJ45-Anschluss verbunden. Der Bildausgang wird außerdem via HDMI-Kabel mit dem Monitor verbunden.

3.2.2 Akku

Der Akku kann frei gewählt werden, er muss jedoch über die nötige Kapazität verfügen, um das System ausreichend lange mit Strom zu versorgen.

3.2.3 Switch

Beim Switch handelt es sich um einen „Davuaz 2,5G Ethernet Netzwerk-Switch“ mit acht Ethernet-Ports und zwei SFP-Ports für die Glasfaserverbindung. Die Ethernet-Ports verfügen über eine Datenübertragungsrate von 2,5 Gbit/s, die SFP-Ports über eine Datenübertragungsrate von 10 Gbit/s.

Abbildung 6: Davuaz 2,5G Ethernet Netzwerk Switch (Quelle: Davuaz)

Der Switch ist die zentrale Schnittstelle für die Datenübertragung. Er verbindet die Kameras über Ethernet mit dem Mini-PC und die Oberfläche über SFP mit Glasfaser. Um das Glasfaserkabel mit dem Switch zu verbinden, wird ein „Optical Gigabit Dual Fiber Module“ des Herstellers „Dengliquiong“ verwendet.

Abbildung 7: Optical gigabit dual fiber module (Quelle: Router-switch Ltd.)

3.2.4 Monitor

Der Monitor dient als visuelle Schnittstelle zwischen dem System und den Oktopussen. Beim eingesetzten Modell handelt es sich um einen „UColor O2“ der Firma UPERFECT mit den folgenden Spezifikationen:

- 16-Zoll-OLED-Bildschirm
- Auflösung 2880 x 1800 (3K)
- Seitenverhältnis: 16:10
- Kontrastverhältnis 100.000:1
- Farbraum: 100% DCI-P3
- Farbtiefe: 10 Bit
- Helligkeit: 500 cd/m²
- Aktualisierungsrate: 120 Hz

Abbildung 8: UColor O2 (Quelle: UPERFECT)

Aufgrund dieser Eigenschaften eignet sich der Monitor besonders gut für die Darstellung detail- und kontrastreicher Bilder. Die hohe Auflösung sorgt für ein realistisches Bild, während die OLED-Technologie und der erweiterte Farbraum auch bei schwachen Lichtbedingungen für klare Darstellungen sorgen. Die hohe Bildwiederholrate sorgt zudem für eine flüssige Wiedergabe von Bewegungen.

Die Entscheidung, einen Monitor in das System zu integrieren, basiert nicht nur auf technischen Überlegungen, sondern auch auf Verhaltensbeobachtungen von Oktopussen. Wie Montgomery beschreibt, haben viele Aquarianer beobachtet, dass Oktopusse ein starkes Interesse an bewegten und farbintensiven Bildern zeigen und sogar gerne fernsehen.

„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]

Wenn er vor dem Sichtfenster des Gehäuses platziert wird, kann der Monitor als „Kino" für die Tiere dienen. Er erzeugt visuelle Reize und ermöglicht so zusätzliche Verhaltensbeobachtungen.

3.2.5 PoE-Injektor

Ein PoE-Injektor (Power over Ethernet) ist ein Gerät, das über ein Netzwerkkabel neben Daten auch Strom an ein Endgerät liefert. Dadurch können IP-Kameras direkt über das Ethernet-Kabel mit Energie versorgt werden, ohne dass ein separates Netzkabel erforderlich ist.

Um die Kameras mit Strom zu versorgen, werden zwei verschiedene Arten von PoE-Injektoren verwendet.

Die beiden Barlus-Modelle sind jeweils mit einem bereits verbauten passiven PoE-Injektor ausgestattet, der mit 24 V DC, 6,5 A durch ein externes LRS-150-Netzteil der Firma Mean Well betrieben wird.

Abbildung 9: Passiv PoE Injektor der Barlus Kameras (Quelle: Barlus)

Abbildung 10: LRS-150-Netzteil (Quelle: Mean Well)

Für die Eigenbaukamera wird ein ALL0488V6 PoE-Injektor der Firma Allnet verwendet. Dieser benötigt kein externes Netzteil und wird direkt ans 230-Volt-Netz gehängt.

Abbildung 11: ALL0488V6 PoE Injektor (Quelle: Allnet)

4 Softwaretechnische Umsetzung des Systems

Die folgenden Kapitel befassen sich mit der softwareseitigen Umsetzung des Systems. Diese teilt sich grundsätzlich in drei Teile. Es gibt die Kamera-Software, die Mini-PC-Software und die Empfänger-Software.

4.1 Kamera Software

Für die Barlus-Kameras wird die vorinstallierte Software des Herstellers verwendet. Bei der Eigenbau-Kamera wird die installierte Software „rpicam-apps" der Raspberry Pi Foundation genutzt. Mit diesen werden die folgenden drei Punkte eingestellt.

4.1.1 Netzwerk Einstellungen

Damit das Netzwerk beide Kameras problemlos erreichen kann, muss in den Einstellungen eine statische IPv4-Adresse eingerichtet werden.

Für die Barlus-Geräte wurde die IP-Adresse der ersten Kamera auf 192.168.1.88 und die der zweiten auf 192.168.1.89 festgelegt.

Bei der Eigenbau-Kamera liegt die IP bei 192.168.1.1 und muss somit nicht geändert werden.

4.1.2 Bildeinstellungen

Bei den Bildeinstellungen liegt der Fokus auf einem scharfen Bild bei geringer Speicherbelastung. Dies wird durch die folgenden Einstellungen erreicht:

Bei einer Auflösung von 1080p und 30 FPS wird eine Bitrate von 3.000 bis 6.000 kbps empfohlen. [10]

Die Kameras produzieren 25 FPS bei einer Auflösung von 1080p. Da unter Wasser die Lichtverhältnisse schlechter sind, was das Bildrauschen verstärkt, wird die Bitrate am oberen Ende bei 6.000 kbps fixiert.

4.1.3 Motor Einstellungen

Als Biofouling bezeichnet man die unerwünschte Ablagerung und das Wachstum von Biofilmen, in diesem Fall speziell von Algen und anderen marinen Ablagerungen. Dieses Phänomen tritt in vielen Situationen auf und kann durch die Installation von Anti-Fouling-Einrichtungen verhindert werden. [11]

Die beiden Barlus-Kameras sind mit jeweils drei Motoren ausgestattet, die zur Entfernung von Algenbesiedelung (Biofouling) dienen. Sie können so eingestellt werden, dass sie in bestimmten Zeitintervallen zweimal über die Sichtfenster der Kamera sowie der Beleuchtung wischen. Das Intervall wurde auf 30 Minuten eingestellt, um Energie zu sparen und ein sauberes Sichtfeld zu garantieren.

Die Eigenbau-Kamera verfügt über keine integrierte Anti-Fouling-Einrichtung und wird von einer externen UV-C-LED beleuchtet, um Algenwachstum zu verhindern. (Eigenbau von Thomas Ederer [4-8])

Abbildung 12: Anti-Fouling UV-C-LED

4.2 Mini PC Software

Der Mini-PC bildet die zentrale Denkeinheit des Systems. Er ist für die Bildaufzeichnung und die Auslastungsaufzeichnung zuständig.

4.2.1 Bildaufzeichnung

Zur Realisierung der Bildaufzeichnung wurde ein Programm in der Sprache C++ geschrieben. Es besteht aus:

- einem JSON-File, in dem Informationen zu den Kameras eingetragen werden.
- einem aufnehmenden Prozess, der für die effektive Aufnahme zuständig ist.
- einem kontrollierenden Prozess, der den Aufnahmeprozess für jede Kamera startet und beendet, sobald die vorgegebene Aufnahmedauer erreicht wurde.

Der grundsätzliche Ablauf ist wie folgt:

Sobald der Prozess „Controller.exe" gestartet wird, liest dieser alle benötigten Informationen aus der JSON-Datei aus und startet nacheinander den Aufnahmeprozess jeder einzelnen Kamera mit einem kurzen Zeitabstand.

Sobald die maximale Aufzeichnungsdauer erreicht ist, wird die Aufnahme gespeichert und der Aufnahmeprozess beendet. Der Controller registriert, dass die vorherige Aufnahme beendet wurde, und startet eine neue.

Dieser Vorgang wiederholt sich, bis der Akku leer ist und keine weiteren Aufnahmen mehr gemacht werden können.

In den folgenden Absätzen wird detailliert auf die einzelnen Komponenten eingegangen.

4.2.1.1 JSON-File

JSON (JavaScript Object Notation) ist ein textbasiertes Datenformat, das hauptsächlich für den Datenaustausch zwischen Systemen verwendet wird. Es ist sowohl für Menschen lesbar als auch für Maschinen einfach zu verarbeiten. Aufgrund seiner Struktur aus Schlüssel-Wert-Paaren und Listen eignet es sich besonders gut zur Darstellung von Konfigurationsinformationen. Daher eignet es sich perfekt, um Basisinformationen für ein Programm bereitzustellen.

Abbildung 13: JSON- Datei mit den Kamerainformationen

Die Grundstruktur der JSON-Datei besteht aus einem Objekt das zwei Hauptelemente enthält:

- **„cameras"**: ein Array aus Objekten. Jedes dieser Objekte beschreibt eine Kamera mit den Eigenschaften **„name"** (Name zum Abspeichern der Aufnahmen), **„active"** (Angabe, ob diese Kamera Aufnahmen machen soll) und **„stream"** (Adresse des Streams).
- **„info"**: ein weiteres Objekt, das zusätzliche Informationen enthält, darunter **„video_length_min"**, das die maximale Länge der Videoaufzeichnung in Minuten angibt.

Kurz gesagt: Das äußerste Objekt besteht aus einer Liste von Kameras sowie einem Block mit allgemeinen Informationen.

4.2.1.2 Controller

Der Controller ist ein in C++ geschriebenes Programm, das für den Start und die Organisation aller Aufnahmen zuständig ist. Er verwendet lediglich die JSON-Datei und benötigt somit keine weiteren Parameter.

Zu Beginn werden die Aufnahmezeit sowie die Liste mit allen Kamerainformationen aus dem JSON-File ausgelesen. Anschließend wird für jede aktive Kamera ein Recorder-Prozess mit einem Abstand von drei Sekunden gestartet. Die Kamerainformationen der laufenden Kameras werden auch in einem Vektor gespeichert, um die einzelnen Prozesse später überprüfen zu können.

Sobald alle Kameraprozesse laufen, wird periodisch überprüft, ob ein Aufnahmeprozess durch Erreichen der maximalen Aufnahmedauer oder durch einen Fehler beendet wurde. In diesen Fällen wird ein neuer Aufnahmeprozess für die jeweilige Kamera gestartet und der Kreislauf beginnt von vorn.

Wurde ein Abbruchbefehl (Ctrl + C-Event) durch den Benutzer registriert, wird kein neuer Aufnahmeprozess gestartet, sondern lediglich auf den kontrollierten Abbruch des aktuellen Prozesses gewartet. Sobald alle Aufnahmeprozesse beendet wurden, beendet sich der Controllerprozess selbst.

4.2.1.3 Recorder

Der Recorder ist ein ebenfalls in C++ geschriebenes Programm, das für die Verarbeitung und das Abspeichern der Kamerastreams zuständig ist. Er nutzt das JSON-File nicht, sondern bezieht die benötigten Informationen über die beim Start des Prozesses übergebenen Parameter. Zur Aufnahme wird die OpenCV-Bibliothek verwendet.

Bei korrektem Start des Programms werden zunächst die Ordner zum Abspeichern der Videodateien erstellt, sofern diese noch nicht vorhanden sind. Damit die Videodateien später mühelos eingeordnet werden können, wird mithilfe der ctime-Library ein Zeitstempel erstellt, der zur Namensgebung der Videodateien dient.

Anschließend werden die Verfügbarkeit des Videostreams, die Auflösung und die Bilder pro Sekunde überprüft. Ab diesem Punkt kann der tatsächliche Aufnahmeprozess beginnen. Dieser wurde mithilfe der offiziellen Dokumentation der OpenCV-Organisation erstellt. [12]

```cpp
int frame_count = 0;
int max_frames = static_cast<int>(fps * RECORD_SECONDS);
Mat frame;
while (frame_count < max_frames && !stop_requested)
{
    cap >> frame;

    if (frame.empty())
    {
        cout << NAME << " error: Failed to grab frame." << endl;
        break;
    }

    writer.write(frame);
    frame_count++;
}
```

Abbildung 14: Codestück des Aufnahmeprozesses

Zu Beginn wird durch die VideoCapture-Funktion ein Frame des Streams erfasst. Dieses wird mithilfe der VideoWriter-Funktion auf das Speichermedium geschrieben und die Anzahl der erfassten Frames wird um eins erhöht. Dieser Vorgang wiederholt sich für eine Anzahl von „max_frames“ mal, wobei sich „max_frames“ durch die Multiplikation der FPS und der vorgeschriebenen Aufnahmezeit in Sekunden ergibt.

Wenn kein Frame erfasst werden kann oder ein Abbruchbefehl empfangen wird, wird die bisherige Aufnahme gespeichert und der Prozess beendet. Im ersten Fall würde der Controller einen neuen Aufnahmeprozess starten. Im Abbruchsfall wird die aktuelle Videodatei gespeichert und der Prozess wird beendet.

4.2.2 Auslastungsaufzeichnung

Um einen Überblick darüber zu erhalten, wie der PC während der Daueraufnahme performt, wird beim Hochfahren des Geräts ein Performance-Log erstellt.

Dafür wird der Performance Monitor von Windows verwendet. Dies ist ein Werkzeug zur Überwachung und Analyse der Systemleistung. Dafür wird ein „Data Collector Set" erstellt, in das Leistungsindikatoren für CPU-Auslastung, Arbeitsspeicher (Available MBytes) und Festplattennutzung (% Disk Time, % Free Space) hinzugefügt werden. Das Collector-Intervall zur Aufzeichnung wurde auf 15 Sekunden fixiert.

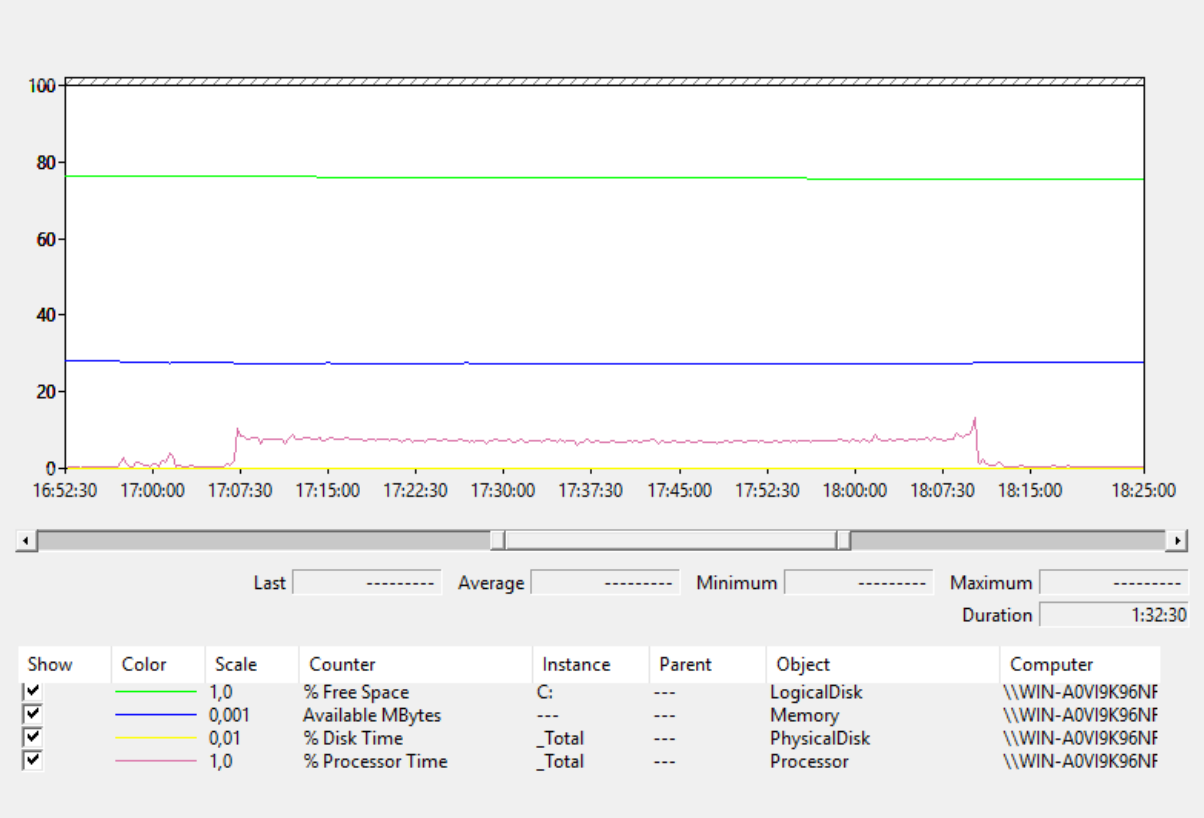

Abbildung 15: Auslastungsaufzeichnung des Systems (Performance Monitor)

Hier wurde die Auslastung einer einstündigen Aufnahme aufgezeichnet. Die Aufnahme wurde um 17:07 Uhr gestartet. Man erkennt, wie die Prozessorzeit auf 10 % ansteigt und sich bis zum Ende der Aufzeichnung auf diesem Niveau stabilisiert. Der Arbeitsspeicher befindet sich relativ konstant bei 30 %, was zeigt, dass genügend Arbeitsspeicher vorhanden ist und die Aufnahme nicht zu Engpässen im RAM führt. Beim freien Speicher kann man beobachten, dass dieser sich zwischen Beginn und Ende der Aufnahme um wenige Prozent verringert. Dies ist auf das Speichern des Videos zurückzuführen.

Insgesamt kann man also sagen, dass der PC mehr als genug Leistung für das gegebene Setup bietet. Dies lässt auch die Möglichkeit, in Zukunft weitere Kameras hinzuzufügen.

4.3 Verbindungs Software

Damit eine Verbindung mit dem Netzwerk unter Wasser hergestellt werden kann, mussten einige Einstellungen vorgenommen werden. Der Mini-PC verfügt über zwei Ethernet-Anschlüsse. Über einen davon ist er mit dem Switch verbunden. Damit er problemlos identifiziert werden kann, wurde beiden Ethernet-Ports eine statische IPv4-Adresse zugewiesen. Dies sind 192.168.1.101 beim „Realtek PCIe GbE Family Controller“ und 192.168.1.102 beim „Intel(R) Ethernet Controller I226-V“.

Zur Verbindung mit der Oberfläche wird die Software RustDesk eingesetzt. RustDesk ist eine quelloffene Lösung für den Fernzugriff und die Fernsteuerung von Rechnern. Sie ermöglicht den Zugriff auf Benutzeroberflächen über das lokale Netzwerk, ohne dass eine physische Verbindung oder direkte Anwesenheit erforderlich ist. Der Vorteil gegenüber kommerziellen Alternativen wie TeamViewer oder AnyDesk liegt insbesondere in der Offenheit des Quellcodes, der Möglichkeit zum Selbsthosting der Serverkomponenten sowie der Unabhängigkeit von zentralisierten Cloud-Diensten.

Dadurch können Einstellungen vorgenommen, der Zustand des Systems überprüft und Wartungsarbeiten durchgeführt werden, ohne das Gerät aus seiner Unterwasserumgebung bergen zu müssen. Unter Wasser wird RustDesk immer automatisch gestartet und ist so eingestellt, dass sich jeder, der das Passwort kennt, damit von der Oberfläche aus verbinden kann.

4.3.1 Datenübertragung

Die Datenübertragung erfolgt ebenfalls über RustDesk. Sobald die Verbindung hergestellt ist, können Daten per Drag-and-drop oder über die Datenübertragungsfunktion von RustDesk kopiert werden.

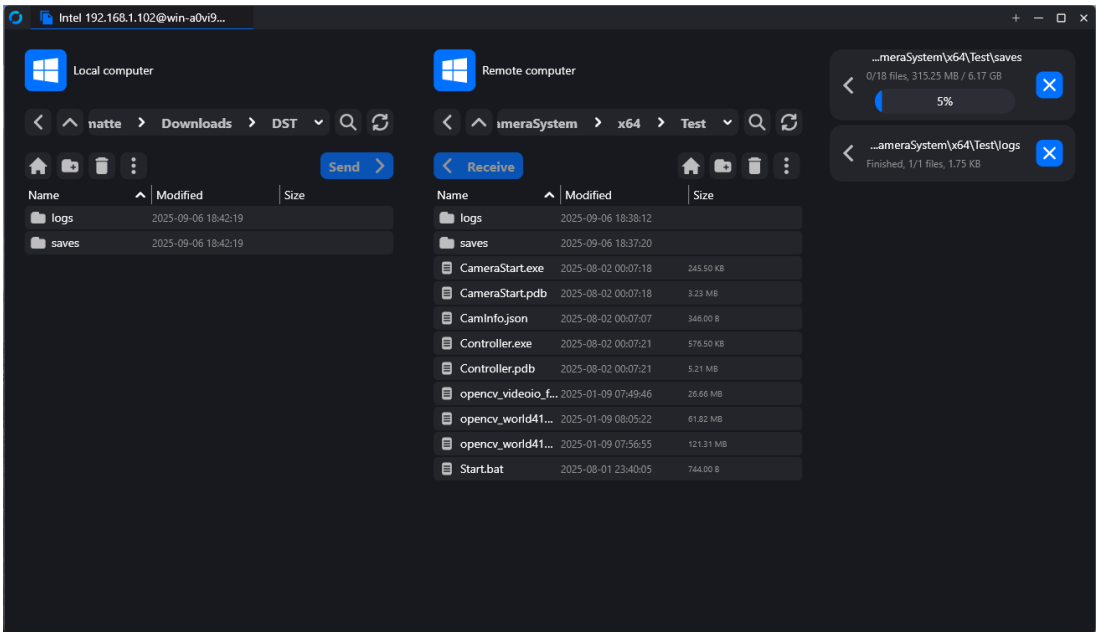

Abbildung 16: Datenübertragung in RustDesk

4.3.2 Geschwindigkeit der Datenübertragung

Zur Messung der Geschwindigkeit der Datenübertragung wird iperf genutzt. Es ist ein zuverlässiges Werkzeug, um die verfügbare Bandbreite und den Durchsatz einer Netzwerkverbindung zu überprüfen. In diesem Fall wurde die Verbindung zwischen dem Mini-PC und dem PC an der Oberfläche getestet, über die die gespeicherten Videodateien übertragen werden.

```
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                  sender
[  5]   0.00-10.01  sec  1.11 GBytes   948 Mbits/sec                  receiver

iperf Done.
```

Abbildung 17: Geschwindigkeitsmessung mit iperf

Der Test ergab über einen Zeitraum von zehn Sekunden einen stabilen Durchsatz von rund 948 Mbit/s, was nahezu der maximalen Leistung einer Gigabit-Verbindung entspricht. Somit ist gewährleistet, dass die aufgezeichneten Videodaten effizient und ohne Engpässe an die Oberfläche übertragen werden können.

5 Ergebnisse

Im Rahmen der Evaluierung des entwickelten Systems wurden verschiedene Tests durchgeführt. Ein zentraler Aspekt der Untersuchung war die Überprüfung der Stabilität bei längeren Videoaufnahmen. Die Analyse mit dem Windows Performance Monitor ergab, dass die Auslastung des Systems während des einstündigen Testzeitraums auf einem konstant niedrigen Niveau verblieb. Die Messungen ergaben eine konstante Prozessorauslastung von etwa 10 %, während der Arbeitsspeicher zu rund 30 % ausgelastet war. Auch die Nutzung der Festplatten war moderat, sodass zu jeder Zeit eine zuverlässige Speicherung der Videodaten gewährleistet war. Die Ergebnisse zeigen, dass das System über ausreichende Leistungsreserven verfügt, um mehrere Videoströme parallel zu verarbeiten, und dass Erweiterungen möglich sind.

Des Weiteren wurde die Datenübertragung zur Oberfläche geprüft. Die Messung ergab einen stabilen Durchsatz von ca. 948 Mbit/s. Dies entspricht nahezu der maximalen Leistung einer Gigabit-Verbindung. Somit ist eine effiziente und engpassfreie Übertragung der aufgezeichneten Videodaten sichergestellt.

Darüber hinaus wurde der Strom- und Speicherverbrauch des Systems über einen Zeitraum von einer Stunde gemessen. Bei deaktiviertem Licht wurde ein Verbrauch von 0,065 kWh gemessen, während bei aktiviertem Licht ein Wert von 0,140 kWh erreicht wurde. Der Speicherverbrauch beläuft sich auf ca. 3,6 GB pro Stunde.

Die Messungen zeigen, dass die Beleuchtung einen wesentlichen Einfluss auf den Energiebedarf hat.

Auf Basis dieser Werte lässt sich auch der Bedarf für einen 24-Stunden-Betrieb mit realistischen Lichtbedingungen berechnen. Dabei wurde angenommen, dass die Beleuchtung täglich für rund acht Stunden benötigt wird, während sie in den übrigen 16 Stunden ausgeschaltet bleibt.

Energieverbrauch pro Tag

$$(8 * \ 0{,}140\ kWh) + (16 * 0{,}065\ kWh) = 2{,}16\ kWh$$

Speicherverbrauch pro Tag

$$3{,}6\ GB * 24 = 86{,}4\ GB$$

Auf Basis der ermittelten Datenrate von ca. 948 Mbit/s lässt sich auch die Kopierzeit der gespeicherten Dateien abschätzen.

Kopierzeit

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

$$691.200\ Mbit \div 948 Mbit/s \ = \ 729s \ \approx 12{,}15\ min$$

Die Analyse ergibt, dass sich für den täglichen Speicherbedarf von 86,4 GB eine Übertragungszeit von ungefähr 12 Minuten ergibt. Dadurch wird sichergestellt, dass selbst

umfangreiche Datenmengen in einer angemessenen Zeit an die Oberfläche übertragen werden können.

Die Ergebnisse bestätigen insgesamt, dass das System sowohl in Bezug auf die Leistung als auch auf die Datenübertragung zuverlässig arbeitet. Der gemessene Strom- und Speicherverbrauch sowie die Kopierzeit liefern zudem wertvolle Informationen für die Planung der Energieversorgung, der Speicherkapazitäten und des Datenmanagements

6 Schlussfolgerung und Ausblick

Im Rahmen dieser Arbeit wurde ein Konzept für ein Unterwasserkamerasystem entwickelt, das die Beobachtung von Oktopussen in ihrer natürlichen Umgebung ermöglichen soll. Die Ergebnisse zeigen, dass die eingesetzte Technik über eine ausreichende Leistung für die parallele Aufzeichnung mehrerer Videoströme verfügt und eine schnelle sowie stabile Datenübertragung über Gigabit-Ethernet realisierbar ist. Zudem konnten der Strom- und Speicherverbrauch erfasst werden, sodass eine erste Einschätzung der Anforderungen an die Energieversorgung und die Speicherkapazität möglich ist.

Das vorgestellte System liegt bislang nur als theoretisches Modell vor und wurde noch nicht in einer realen Unterwasserumgebung getestet. Ein nächster Schritt besteht daher in der praktischen Erprobung im Meer, um die Belastbarkeit und Zuverlässigkeit unter realen Bedingungen zu überprüfen.

In zukünftigen Arbeiten könnte das System um automatische Bildanalyse, zusätzliche Sensoren oder eine Optimierung des Energieverbrauchs erweitert werden. Damit würde sich sowohl der wissenschaftliche Nutzen als auch die technische Effizienz steigern lassen und eine langfristige, möglichst unbeeinflusste Beobachtung des Verhaltens von Oktopussen ermöglichen.

7 Literaturverzeichnis

[1] Godfrey-Smith, Peter, „*Other Minds: The Octopus, the Sea, and the Deep Origins of Consciousness.*", New York 2016, S. 13.

[2] Vgl. 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] Vgl. 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/ [Zugriff: November 2025]

[6] Ederer Thomas, *„OctoWatch-Video Service"*, GitHub Repository, 2025, https://github.com/tederer/octowatch-videoservice/tree/v0.4.0 [Zugriff: 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 [Zugriff: 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 [Zugriff: 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 [Zugriff: 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/ [Zugriff: September, 2025]

8 Abbildungsverzeichnis