

Ein Augmented Reality System zur Performance Augmentation in Feuerwehreinsätzen

Studienarbeit (T_3101)

des Studiengangs Informatik
an der Dualen Hochschule Baden-Württemberg Karlsruhe

von

Joel Koch

und

Jonas Weimar

20. September 2021

Abgabedatum	16. Mai 2022
Bearbeitungszeitraum	Oktober 2021 – Mai 2022
Matrikelnummer, Kurs	4875535, TINF19B5
	6628339, TINF19B5
Betreuer der Dualen Hochschule	Prof. Dr. Marcus Strand

Ehrenwörtliche Erklärung

Wir erklären hiermit ehrenwörtlich, dass

1. die vorliegende Studienarbeit ohne fremde Hilfe angefertigt wurde,
2. die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet wurde,
3. die Arbeit in gleicher oder ähnlicher Fassung noch keiner anderen Prüfungsbehörde vorgelegt worden ist.

Wir sind uns dessen bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

Ort, Datum

Koch, Joel (4875535)

Ort, Datum

Weimar, Jonas (6628339)

Abstract

Bei Feuerwehreinsätzen ist Umweltbewusstsein ein Schlüssel für das operative Personal, um die diversen Aufgaben, welche ihnen auf Ertragen werden, erfüllen zu können. Um die Feuerwehreinsatzkräfte in dieser gefährlichen Berufung zu unterstützen und ihre Leistungsfähigkeit sowie die Leistungsfähigkeit der Operationszentrale zu steigern, wird in dieser Arbeit ein System auf Basis eines AR Geräts implementiert und vorgestellt, welches einige Sensoren sowie Thermal Imaging in einer neuen Generation Einsatzhelme zusammenfasst. Ziel hierbei ist es, Augmented Reality zu nutzen, um Feuerwehrleute bei der Suche nach Brandherden, dem Echt-Zeit Kartieren des Einsatzgebiets und dem Lokalisieren von potenziell verschütteten Personen zu unterstützen sowie ihnen Echt-Zeit Sensordaten und Warnungen unter rapiden Veränderungen derselben per Heads-Up-Display darzustellen. Darüber hinaus haben wir die Feuerwehr des Frankfurter Flughafens (FRA) konsultiert, um unser System unter realen Umständen zu testen. Um eine zuverlässige Bewertung unseres Projekts zu geben, wird sich diese Arbeit folgend mit dem Hintergrund der Arbeit, den verwendete Soft- und Hardware sowie der Implementierung und dem Testen des Systems beschäftigen. Folgend wird in einer Evaluierung Relevanz und zukünftige Verwendung des Projektes diskutiert.

Inhaltsverzeichnis

1	Einleitung.....	1
1.1	<i>Motivation.....</i>	2
1.2	<i>Herangehensweise und Methodik.....</i>	3
1.3	<i>Stand der Technik</i>	5
2	Grundlagen HoloLens 2.....	9
2.1	<i>Technische Details.....</i>	10
2.2	<i>Interaktionsfähigkeit und Umgebungsverständnis.....</i>	12
3	Grundlagen des Sensormodul	17
3.1	<i>Bestandteile und Anforderungen des Moduls</i>	17
3.2	<i>Prototyp.....</i>	22
3.3	<i>Verbessertes Sensor-Modul</i>	26
4	Implementierung Sensormodul.....	32
4.1	<i>Installation Raspberry Pi Betriebssystem</i>	32
4.2	<i>Sensoren-Schnittstelle.....</i>	33
4.3	<i>Wärmebildkamera-Schnittstelle.....</i>	40
4.4	<i>Zusätzliche Konfigurationen.....</i>	43
5	Implementierung Holo-Modul	46
5.1	<i>Aufsetzen der Mixed Reality Applikation.....</i>	47
5.2	<i>Implementieren Heads-Up-Display.....</i>	50
5.3	<i>Kartographierung des Einsatzgebiets.....</i>	62
5.4	<i>Steuerung des MARS 1.....</i>	66
5.5	<i>Integration in einen Brandbekämpfungshelm.....</i>	69
6	Untersuchung des Systems	71
7	Fazit und weiterer Ausblick	78
I.	Abkürzungsverzeichnis.....	83
II.	Tabellenverzeichnis	85
III.	Abbildungsverzeichnis.....	86
IV.	Kapitelauftteilung.....	88
V.	Literaturverzeichnis	89
VI.	Anhang.....	94

1 Einleitung

Science-Fiction (Sci-Fi). Viele moderne Technologien gehen auf Inspirationen aus Literatur und Film zurück. Besonders prägend für unseren modernen Aufschwung ist die Film-Industrie der frühen 60er Jahre. Raumschiffe, Zeitreisen, Sprachsteuerung und Robotik öffnen eine Welt der Fantasie, welche in den folgenden Jahrzehnten durch ihren Einfluss zu den größten Erfindungen beitragen soll. Nennenswerte Beispiele sind sprachgesteuerte Computer, das mobile und klappbare Telefon (Communicator) und der computergestützte Übersetzer aus *Star Trek* [1]. Geräte, ohne welche wir heute einem anderen Alltag nachgehen würden. Spätere Sci-Fi Filme, wie *Terminator* (1984) und *Iron Man* (2008), übernahmen frühe Ideen des *Heads Up Displays* (HUD) und führten diese in ihren Filmen komplex aus, wobei die Idee eines Informationsdisplays im Sichtfeld auf den zweiten Weltkrieg zurückgeht. Um Kampfpiloten bei der Zielfindung zu unterstützen, wurden statische Geräte im Blickfeld der Piloten installiert [2]. Heutzutage finden sich Heads-Up-Displays in diversen Einsatzgebieten, wie in der Automobilindustrie und dem Militär. Ziel dieser Arbeit ist es diese in Fiktion und Realität bereits eingesetzte Technologie in einen Helm zu überführen, welcher das Potential zeigt, das Personal der Feuerwehr Branche bei ihren Einsätzen zu unterstützen. Da Feuerwehreinsätze ihr operatives Personal gerade in einer Zeit solch extremen Wandels mit hoch wechselhaften Konditionen und Umgebungsverhältnissen konfrontieren, kann diese Berufung als höchst gefährlich eingestuft werden. Der Beruf umfasst ein sehr großes Spektrum an Aufgaben, außerhalb der Bekämpfung von Bränden. Um den Umfang der Arbeit zu wahren, wird sich das in dieser Arbeit vorgestellte System ausschließlich auf die Brandbekämpfung und das Retten von potenziell verschütteten Personen konzentrieren, denn statistisch sind von 2012 bis 2019 in Deutschland durchschnittlich 378 Personen pro Jahr (ungefähr 5 Personen auf eine Million Einwohner) an Folgen von Bränden, wie Kohlenstoffmonoxid-Vergiftung oder Verbrennungen, verstorben; in den USA liegt diese Zahl bei durchschnittlich 3500 Personen pro Jahr (ungefähr 11 Personen auf eine Million Einwohner). Nicht nur bilden diese Zahlen einen hohen Verlust an Leben ab, sie führen eine Lücke auf, welche unter Verwendung neuester Technologie und dem mit diesem verbundenen Potenzial verkleinert werden kann. So wird in dieser Arbeit ein neues, mobiles und erweiterbares System vorgestellt, welches auf Basis eines *Head Mounted Display* (HMD) mit transparenten Displays verschiedene Sensoren, Algorithmen und Kamerasysteme vereint und für Testzwecke

beispielhaft an die Bedürfnisse und Wünsche der Feuerwehreinsatzkräfte anpasst; die Anpassung bezieht sich in diesem Projekt speziell auf Absprachen mit der Feuerwehr des Frankfurter Flughafens. Um folgend einen mit standardisierten Helmen gleichwertigen Schutz der Einsatzkräfte zu gewährleisten wird das System exemplarisch in einen außer Betrieb genommenen Einsatzhelm eingearbeitet.

1.1 Motivation

Im Sommer, insbesondere im Juni, 2021 kam es in Deutschland und anderen Teilen Europas zu einer schweren Flutkatastrophe, mit schätzungsweise 180 Toten allein in Deutschland. So kam es teilweise zu Regenergüssen von bis zu 150 Litern pro Quadratmeter innerhalb von 24 Stunden. Infrastruktur, wie Bahnstrecken, Straßen und einiger Orts sogar die Wasser-, Strom und Gasversorgung, ist zerstört worden. Häuser wurden in den reißenden Fluten abgetragen. Existenz sind durch den Verlust ihres Hab und Gut an den Rand des Abgrunds getrieben worden. Und Menschen- sowie Tierleben fanden in dieser Katastrophe ihr Ende. Einsatzkräfte diverser Branchen – Feuerwehr, Technisches Hilfswerk, Bundeswehr, Polizei, etc. – sowie freiwillige Helfer haben versucht alles in ihrer Macht Stehende zu unternehmen, um den Betroffenen zu helfen und zu retten was noch zu retten ist [3]. In Kalifornien und der Westküste der USA allgemein kommt es seit Jahren zu immer stärkeren Dürreperioden und Waldbränden, welche Jahr für Jahr Existenz bedrohen und auf fast jährlicher Basis neue Rekorde aufstellen. So führte Kalifornien 2020 den Begriff „Gigafeuer“ ein, für Feuer mit zusammenhängendem Brandherd von 4000 Quadratkilometern. Beispielhaft für die unglaubliche Einschlagskraft dieser Katastrophen [4].

Mit dem unter anderem vom Menschen verschuldeten Klimawandel [5] steht uns ein Zeitalter bevor, welches von potenziellen Ereignissen dieses Ausmaßes geprägt sein könnte. Umso wichtiger ist es, dass neben der Entwicklung zuverlässiger Warn- und Präventionssysteme, auch die Ausrüstung der Einsatzkräfte in ihrer unterstützenden Fähigkeit weiterentwickelt und an die schwerwiegenden Situationen angepasst wird, sodass Einsätze mit maximalem Potential durchgeführt werden können. Um diese Ausrüstung weiterzuentwickeln, bietet die rapide technologische Entwicklung ein enormes Potential und für jene die sich mit dieser Beschäftigen eine hohe Verantwortung.

Mit dem Fokus auf das Auffinden von (teils-)verschüttenden Personen und Brandherden auch unter schwierigen Bedingungen, wird sich diese Arbeit mit der Planung, Umsetzung und dem Testen eines intuitiven Mixed Reality Systems zur weitreichenden Unterstützung der Feuerwehreinsatzkräfte beschäftigen.

1.2 Herangehensweise und Methodik

Um die Anforderungen an ein Mixed Reality Einsatzhelmsystem zu recherchieren ist die Konsultation der selektierten Feuerwehrbranche unabdingbar. In dieser Arbeit ist für die Kommunikation diverser Anforderungen exemplarisch die Feuerwehr des Frankfurter Flughafens herangezogen worden. Im Verlauf einer Reihe von Terminen ist dabei das ursprüngliche Konzept eines allgemein gehaltenen Einsatzhelms auf die Anforderungen der Feuerwehr heruntergebrochen worden, so dass ein Systemkonzept entstanden ist, welches die Einsatzkräfte in ihrem gefährlichen Beruf unterstützt und ihnen durch die Integration verschiedener Mixed Reality Systeme eine Erweiterung ihres Sinnesumfelds durch moderne Sensorik bietet. Die hierbei kristallisierten Anforderungen sind in der folgenden Liste, auf drei Systeme heruntergebrochen, aufgeführt und weiter beschrieben.

ID	Name	Beschreibung
1.	Immersives Heads Up Display	Das immersive Heads Up Display ist die darstellende Schnittstelle des Systems. Es dient der Darstellung aller wichtigen Informationen und besteht aus einzelnen Modulen.
1.1.	WBK Einbindung	Dieses Modul ist für die Wiedergabe eines Wärmebilds innerhalb des HUD zuständig.
1.2.	Einbinden der Sensorwerte	Dieses Modul ist für die Abbildung der Sensorwerte des Systems zuständig.
1.3.	Nachrichten Log	Dieses Modul ist für die Benachrichtigung von Fehlern innerhalb des Systems zuständig.

2.	Immersives Steuer-menü	Um das immersive Heads Up Display zu steuern, wird ein Modul implementiert, welches ein virtuelles Menü darbietet.
3.	Sensor-Modul	Das Sensor-Modul ist für alles rum um das Thema Sensoren zuständig. Es hat die Aufgabe der Anbindung der Sensoren, das Auslesen der Sensoren, der Verarbeitung des Sensorwerte sowie der Weiterleitung an das HoloLens-Modul.
3.1.	Sensoren Basisgerät	Bei diesem Teilmódul handelt es sich um das Basisgerät, worauf alles um das Thema Sensoren stattfindet.
3.2.	Stromversorgung	Dieses Teilmódul ist für die Stromversorgung des Sensoren Basisgerät zuständig.
3.3.	Sensoren Schnittstelle	Einbinden in das System des Basisgerätes, Auslesen der einzelnen Sensoren sowie der Bereitstellung der Daten für die HoloLens
3.4.	Wärmebildkamera-schnittstelle	Einbinden einer Wärmebildkamera in das System sowie bereitstellen der WBK Daten über die Schnittstelle des Basisgerätes für die HoloLens.

Tabelle 1 Liste der Funktionalitäten

Aus der hervorgegangenen Listen sind der Feuerwehr die Punkte Wärmebildkameraintegration und Sensorik (insbesondere Sauerstoff und Kohlenstoffmonoxid) sehr wichtig, da diese in Gefahrensituationen eine besondere Unterstützende Rolle einnehmen können. Aus der Absprache geht außerdem hervor, dass im Bereich Brandherderkennung für die Wärmebildkamera ein Wertebereich von ca. -10 bis 400° Celsius als hinreichend akzeptiert würde, da die meisten konventionellen Brände auf diesem Bereich operieren. Ausnahmen bieten hier bspw. Stahlbrände, welche Temperaturen von über 1000°C erreichen können. Außerdem ist aus den Terminen hervorgegangen, dass Sensoriken eine hinreichende Genauigkeit bieten sollten, ohne maximal akzeptable Abweichungswerte zu spezifizieren.

Um die Einsatzfähigkeit des in dieser Arbeit vorgestellten Mixed Reality Einsatzhelms sowie der verwendeten Einzelsysteme auszuwerten und zu beurteilen, wurde deshalb sowohl eine quantitative als auch eine qualitative Untersuchung am vorliegenden System durchgeführt. Beide Untersuchungen wurden induktiv in Kooperation mit der Feuerwehr des Frankfurter Flughafens durchgeführt. Hierbei sind die Daten durch eine Feldstudie erhoben worden, bei welcher das System auf der Atemschutzsimulationsanlageanlage der Feuerwache 1 des Flughafens und in der Fahrzeughalle der Freiwilligen Feuerwehr Hattersheim-Okriftel zum Einsatz gekommen ist. Um die mit dem System gesammelten Daten (betrifft Subsystem 3, siehe Tabelle 1) folgend quantitativ analysieren zu können, wurden unter Verwendung von aktiv eingesetztem Feuerwehrequipment, dem Honeywell GasAlertQuattro 4-Gas Warnsystem, Referenzdaten erhoben. Hierbei wurden die Referenzwerte der im Dienst der Feuerwehr befindlichen Geräte als wahrhaftig und die Differenz der Mars 1 Messungen als Fehler gesehen. Die qualitative Bewertung der Funktionalität 1.1 bzw. 3.4 und der Benutzerfreundlichkeit des Gesamtsystems erfolgt durch den mündlichen Austausch des Projektteams mit den testenden Einsatzkräften der Feuerwehren.

Die Forschung, Implementierung und Auswertung des Systems hat von Oktober 2022 bis Mitte Mai 2022 stattgefunden. In dieser Zeit sind die notwendigen Module geplant, implementiert und getestet worden. Dabei wurden die unterschiedlichen Sub-Systeme separiert voneinander entwickelt und folgend durch eine WLAN basierte Kommunikation zusammengeführt.

1.3 Stand der Technik

Um die Relevanz des in dieser Arbeit vorgestellten Systems in einen Gesamtkontext einordnen zu können und die durchgeführte Untersuchung zu bewerten, wird in diesem Kapitel auf analoge Forschungsarbeiten und ähnliche Projekte eingegangen. Da sich die Forschung in diesem speziellen Kontext noch auf eine kleine Anzahl von Projekten beschränkt, wurden beispielhaft zwei relevante Systeme herangezogen. Diese werden in den folgenden Abschnitten vorgestellt.

3D Building Reconstruction and Thermal Mapping in Fire Brigade Operations.

Im Rahmen der IEEE Virtual Reality Conference 2013, hat eine Gruppe Forscher der Technischen Universität Wien ein System vorgestellt, welches Gebäude dreidimensional kartographiert und die Tiefen Daten mit den Informationen der LWIR Infrarot Kamera kombiniert; Diese Forscher sind Christian Schönauer, Emmanuel Vonach, Georg Gerstweiler und Hannes Kaufmann gewesen [6]. Bei der Umsetzung des Projekts hat das Team der Technischen Universität Wien auf unterschiedliche Hardwaresysteme zurückgegriffen und diese in einem portablen Setup vereint. Dabei werden die Tiefendaten mit einer Asus Xtion Pro aufgezeichnet. Diese sammelt Tiefendaten unter Verwendung des *Structured Light* Ansatzes mit 30 frames per second bei einer Auflösung von 640x480 Pixeln und bietet eine horizontales Field Of View von 57° und ein vertikales Field Of View von 43° [6].

Exkurs *Structured Light Approach*: Werden Tiefendaten mit dem *Structured Light* Ansatz aufgezeichnet, wird ein strukturiertes Lichtsignal in den Raum projiziert. Aus den Reflektionen dieses Signals kann folgend per Triangulation die ungefähre Tiefe der Umgebung errechnet werden [6].

Um Informationen über die Hitzeverteilung der Umgebung zu sammeln, hat das Team die Xenics Gobi-640GigE LWIR Kamera mit einer Auflösung von 640x480 eingesetzt. Diese bietet in der Spitze eine Sensitivität zwischen 8 und 14µm und kann so ungekühlt ein Temperaturspektrum von -20°C bis 100°C abdecken. Zur Verrechnung der Tiefendaten und der aufgezeichneten Hitzeverteilung setzte das Team auf eine Intel Core i7-3720 Notebook mit 16GB RAM und einer NVIDIA GeForce GTX 680M mit 4096 MB RAM. Dabei werden Tiefensor und WBK über ihre intrinsischen und extrinsischen Parameter aufeinander kalibriert. Zur Visualisierung ihrer Daten setzte das Team auf das Silicon Micro Display ST1080 Head Mounted Display, welches die verrechneten Daten (live) darstellt [6]. Das so entstandene System ist in der Lage parallel die Daten der Wärmebildkamera sowie die des ToF Sensors aufzuzeichnen, miteinander zu verrechnen und über das angebundene Head Mounted Display zu wiedergeben. Hierbei ist das System von 2013 jedoch an einen externen Computer, in diesem Fall einen Laptop mit Intel Chip und NVIDIA Graphics Karte, gebunden um diese Berechnungen durchzuführen. Das System zeigt somit ein hohes Innovationsniveau, ist aufgrund der

zu dieser Zeit verfügbaren Hardware jedoch in seiner Mobilität und Kompaktheit eingeschränkt. Ein Problem, welches in dieser Arbeit aufgegriffen und durch modernere Hardware behandelt wird.

Integrated Visual Augmentation System. Das *Integrated Visual Augmentation System (IVAS)* ist ein Joint-Venture Projekt der United States Army und dem Technologiekonzern Microsoft, welcher maßgeblich auch für die in dieser Arbeit verwendete Hardware verantwortlich ist [7], [8]. Ziel ist es, kampfbereite Augmented Reality HMDs auf Basis der Microsoft Hololens herzustellen, welche das Kampfumfeld der Soldaten um diverse Funktionalitäten erweitern. So dient das Headset laut Hololens-Projektleiter Alex Kipman (Microsoft) der Steigerung der situativen Aufmerksamkeit und Konnektivität der Einsatzkräfte [9]. Genannt wurden hier bereits Funktionalitäten wie Nachtsicht, dem Durchblicken von dichtem Rauch und der Echtzeitgeländedarstellung als Hologramm mit Feind- und Truppmarkierung [9]. Es existieren bereits diverse Prototypen, welche in umfangreichen Testszenarien auf den Prüfstand gestellt werden (Abbildung 1). Um das System zuverlässig steuern und bedienen zu können erhalten die Soldaten eine Steuerungseinheit, welche auf der Brust befestigt wird und per Kabel mit dem HMD kommuniziert (Abbildung 1).



Abbildung 1 Ein Soldat der 1-508PIR, 82nd Airborne Division beim Testen des IVAS [10]

Der aktuelle Forschungsstand auf dem Gebiet der Mixed Reality Einsatzhelme ist somit schwer einzuschätzen, da ältere Systeme auf nun überholter und sperriger Hardware operieren und vergleichbare Systeme, wie das IVAS, bis zu einem bestimmten Punkt der Geheimhaltung unterliegen. Es lässt sich jedoch annehmen, dass das in dieser Arbeit vorgestellte Projekt zu einigen wenigen in dieser Forschungssparte gehört und somit, auch durch die spezielle Anpassung an die Bedürfnisse der Feuerwehr, zu den potenziellen Innovationstreibern derselben zählt.

2 Grundlagen HoloLens 2

Dem Mars 1 liegt die von Microsoft entwickelte HoloLens 2 zu Grunde. Bei den Produkten der Microsoft HoloLens Reihe handelt es sich um völlig eigenständige holografische Computer integriert in HMDs, welche den Nutzern das Eintauchen in die Augmented Reality, einen Bereich des von *Milgram* und *Kishino* im Paper „*A Taxonomy of Mixed Reality Visual Displays*“ definierten Reality-Virtuality Continuum, ermöglichen [11], [12]. Dieses führt eine Skala ein, um Software auf der Skala zwischen Virtualität und Realität einzurichten. Hierbei entstehen zwei Zwischenbereiche, welche Systeme beschreiben können, die einen Mix der Skalen Pole implementieren, bspw. indem das visuelle Bild einer Person durch virtuelle Informationen erweitert wird (siehe).

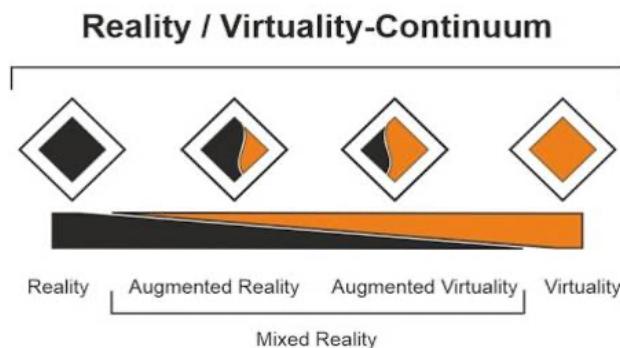


Abbildung 2 Reality / Virtuality Continuum [11], [13]

So können Nutzer durch diese Geräte mit diversen digitalen Inhalten (bspw. dreidimensionalen Hologrammen), unter Einsatz KI gestützter Prozesse wie Hand-Tracking, Eye-Tracking, Kopfbewegungen und Spracherkennung, in einem realen Umfeld interagieren [12]. Dabei realisieren die Produkte der HoloLens Reihe diese Interaktionsprozesse zwischen Nutzer und immersive Software durch komplexe Kamerasysteme und diverse Sensoren, welche in das HMD integriert sind [12]. In diesem Kapitel wird detailliert auf die verwendeten Systeme und die technischen Details der HoloLens 2 eingegangen, um eine Grundlage für den Aufbau des Mars 1 Einsatzhelms zu schaffen.

2.1 Technische Details

Die Microsoft HoloLens 2 ist ein unabhängig einsetzbares, holographisches Head Mounted Display. Sie ist vollends ausgestattet mit eigenem Computer und Akku, so dass ein unabhängiger Betrieb von externen Computern und externer Stromversorgung garantiert ist. So bietet die HoloLens eine solide Basis für mobile, holographische Einsatzzwecke mit visueller Computer-Mensch Schnittstelle. Um dem Endnutzer eben diese Funktionalitäten zur Verfügung zu stellen, umfasst die HoloLens eine große Auswahl an Sensoren und Hardwareelementen, welche die Immersion zwischen realer und virtueller Welt ermöglichen. In den folgenden Abschnitten wird auf eben jene technische Details eingegangen, die diese Immersion ermöglichen.

Display

Die HoloLens 2 zielt darauf hinaus Immersionen digitaler Inhalte innerhalb der Realität zu erzeugen und das Bewusstsein der Nutzer so künstlichen Stimuli auszusetzen, um den Eindruck zu erschaffen, dass eine Interaktion mit realen Objekten stattfinde. Ein Kern-Stimuli, welcher über bekannte Technologien wie Displays angesprochen werden kann, ist der Seh-Sinn. Über ihn können Nutzer Interaktionen mit digitalen Objekten wahrnehmen, indem diese durch komplexe Berechnungen ihren Zustand in Relation zu den Handlungen des Nutzers verändern. Um sich von einfachen Displays weg hin zu realistischen Immersionen der digitalen Inhalte innerhalb der realen Welt zu bewegen, setzt die Microsoft bei der Entwicklung HoloLens 2 auf durchsichtige Displays [14]. Diese Displays basieren auf derselben Technologie wie Glasfaserkabel: den Lichtwellenleitern [15]. Hierbei werden Lichtimpulse durch Glasfaserkabel geleitet, so dass Bilder dargestellt werden können.

Sensorik und Betriebssystem

Die Hololens 2 verfügt über ein großes Array an Sensorik, welche in das Gerät eingesetzte ist und so eine autarke Interaktion zwischen der Umgebung, den Nutzern und den digitalen Inhalten zu ermöglicht. So sind für das Aufzeichnen der Kopfbewegungen und das Erkennen von Oberflächen vier Kameras verantwortlich, welche auf dem sichtbaren Spektrum des Lichts arbeiten; Zwei Infrarot Kameras verfolgen die Augen der Trägerperson, um eine Identifikation digitaler Inhalte in Kombination mit Sprach-

befehlen und so eine flüssigere Interaktion sowie ein Iris-Erkennung basiertes Sicherheitssystem zu ermöglichen; Ein 1-Megapixel Time-of-Flight (ToF) Tiefensensor vermisst die Umgebung und liefert die Daten, aus welchen die Tiefenbilder zusammengesetzt werden und eine reguläre 8-Megapixel Kamera zum Aufzeichnen von Fotos und 1080p30 Videos. Zusätzlich zu den Kameras umfasst das Sensor Paket der Hololens 2 noch eine Inertial Measurement Unit (IMU), um über einen Beschleunigungssensor und ein Gyroskop die Winkelgeschwindigkeit der Hololens 2 in Relation zu der vermessenen Umgebung zu ermitteln [14]. Um alle Daten verarbeiten zu können ohne auf externe Computer und Rechenleistung angewiesen zu sein, hat Microsoft für seine HoloLens Produktreihe ein eigenes hardwareoptimiertes Betriebssystem und Akkukonzept mit Energieeffizienter Recheneinheit entwickelt. Dieses ermöglicht neben dem autarken Nutzen des Geräts noch einen weiteren Aspekt: ein angepasstes Userinterface. Die Hololens 2 kann durch ihr integriertes Hand-Tracking System per Gesten- und Sprachsteuerung bedient werden ohne die Abhängigkeit einer physischen Tastatur oder Maus voraussetzen zu müssen [14].

Rechenleistung und Konnektivität

Um die Sensorwerte verarbeiten zu können und die Rechenleistung für die dreidimensionalen Graphiken bereitstellen zu können, verfügt die HoloLens über diverse Halbleiterprodukte. Dabei dient ein Qualcomm Snapdragon 850 Compute Platform als zentrale Recheneinheit. Für die Berechnung der holographischen Inhalte und die Auswertung und Verarbeitung aller in das Gerät integrierten Sensoren hat Microsoft für seine HoloLens Produktreihe eine spezielle Holographische Recheneinheit (*Holographic Processing Unit*) entwickelt [16]. Die in der HoloLens 2 verbaute zweite Generation dieser Einheit ist in der Lage alle Auswertungen wie Sprach- und Gestenerkennung auch ohne Cloudanbindung durchzuführen und ist so nicht angewiesen auf eine kontinuierliche Internetverbindung [17]. Um alle gesammelten Daten verlustfrei auf dem Gerät auswerten zu können, verfügt die HoloLens 2 über einen 4GB-LPDDR4x-System DRAM. Zusätzlich verfügt die HoloLens 2 über einen 64GB Universal Flash Storage, welcher dem dauerhaften Speichern von Dateien wie Bildern, Videos und diversen anderen Daten dient. Um mit dem Gerät kabellos im Heimnetzwerk zu kommunizieren, verfügt die HoloLens über ein WLAN Modul, welches auf der Basis des 802.11ac WLAN-Standards arbeitet. Zusätzlich zur Kommunikation über WLAN, steht

der HoloLens 2 ein Bluetooth 5.0 fähiger Chip sowie eine USB-C DRP Schnittstelle bereit [14].

Stromversorgung

Die HoloLens 2 hat aufgrund ihrer vielen Sensor- und Hardwareoptionen einen hohen Energieverbrauch. Um den Nutzern ein immersives Erlebnis zu ermöglichen, welches die Dauer von einigen Minuten überschreitet, hat Microsoft einen Lithium Akkumulator in das Gerät eingelassen, welcher laut internen Studien bei aktiver Nutzung zwischen zwei und drei Stunden durchhält. Im Standby-Zustand hält der Akku laut Microsoft zwei Wochen [14]. Ein Vorteil des Systems ist die Möglichkeit der Nutzung im Ladezustand, so muss bei der Entwicklung keine direkte Rücksicht auf den Stand des Akkumulators genommen werden. Eine Voraussetzung zu dieser Funktionalität ist jedoch der Anschluss der HoloLens an ein 15 Watt Ladegerät. Hierbei wird das HoloLens System passiv gekühlt, es sind somit keine Ventilatoren integriert, was ein schmales Design der Recheneinheit des HMD ermöglicht.

2.2 Interaktionsfähigkeit und Umgebungsverständnis

Ziel des Hololens Projektteams ist es gewesen ein Produkt mit umfangreichen Funktionalitäten auf den Markt zu bringen, welches trotzdem instinktiv und einfach zu benutzen ist [18]. Hierbei setzt Microsoft vermehrt auf den Einsatz von Künstlicher Intelligenz um die Handlungen der Träger und die Umwelt zu interpretieren [18]. Die eigens für die Hololens 2 entwickelten KI-Modelle bauen auf den Eigenschaften, Funktionalitäten und Sensoren aus Abschnitt 4.1 auf. Zwei Kernkonzepte dieses Umwelt- und Trägerverständnis, die hohe Relevanz im Kontext dieser Arbeit finden, werden folgend aufgeführt und in den folgenden Absätzen erläutert.

Menschliches Verständnis

Für die intuitive Interaktion zwischen Gerät und Mensch ist das Interpretieren eines Teils des gewaltigen Spektrums der menschlichen Gestik sowie das Verstehen ausgewählter Sprachbefehle fundamental. Da in dieser Arbeit ausschließlich über Gesten mit der Hololens 2 kommuniziert wird, wird der Prozess der Spracherkennung außer Acht gelassen. Um das für die Gestenerkennung zuständige und grundlegende Hand-Tracking-System zu entwickeln, hat das Microsoft Team in einem speziell entwickelten

Aufbau mit diversen Kameras, angeordnet in unterschiedlichen Winkeln, eine vielfältige Anzahl an Händen, unterschiedliche Gesten formend, aufgezeichnet [18]. Aus diesem Datenbestand wurde mittels Cloud-Processing ein 3D Modell der menschlichen Hand gefertigt, welches ein großes Spektrum der möglichen Gesten abdeckt. Mit diesem 3D Modell wurde im Anschluss ein *Compact Deep Neural Network*, ein effizienter Deep Learning Algorithmus, trainiert, welcher auch auf der zentralen Recheneinheit der Hololens 2 (mit deutlich begrenzter Leistung) effizient Berechnungen und Vorhersagen treffen kann [18]. Dieses von Microsoft entwickelte Hand-Tracking-System ermöglicht das Interagieren der Nutzer mit Hologrammen, indem es die Positionen der einzelnen Finger vorhersagt und die Intention der Bewegung folgend interpretiert. In Applikationen kann die vorhergesagte 3D Struktur der Hand beziehungsweise das korrespondierende Mesh auch visualisiert werden. Hierbei erkennt man, dass das Hand-Tracking-System auch die Position von Fingern vorhersagt, welche sich nicht im Sichtfeld der Kameras befinden (Abbildung 3).



Abbildung 3 Beispiel der Handerkennung mit fixiertem Hologramm [19]

Räumliche Koordinatensysteme

Der Hololens 2 liegt als Grundlage für die Lokalisierung ihrer Hologramme ein kartesisches Koordinatensystem zugrunde, welches aus den drei Achsen – X, Y und Z – besteht. Diese Koordinatensysteme finden auch in anderen 3D Anwendungen Verwendung und können in links- und rechtsseitige Systeme unterteilt werden. Die auf Windows basierende Hololens verwendet hier die rechtsausgerichtete Darstellung, das heißt: Die positive X-Achse verläuft nach rechts, die positive Y-Achse zeigt nach oben und die positive Z-Achse auf den Nutzer [20], [21, p. 50]. Windows bezeichnet seine Implementation des Koordinatensystems als *Spatial Coordinate System* (SCS).

Die Abstände in diesen SCS werden in Metern gemessen. Ziel dieser Festlegung ist es, den Entwicklern eine sichere und einfachere Skalierung ihrer Applikationen zu ermöglichen [20].

Bei der Entwicklung einer Hololens 2 Applikation mit Unity definiert das aus der Konfiguration *Stationary frame of reference* bereitgestellte Koordinatensystem den Ausgangspunkt der Applikation als das Gerät (Hololens 2) selbst. So wird bei jedem neuen Start der Applikation das Koordinatensystem neu, ausgehend von der Position der Hololens, initialisiert. Um innerhalb von Szenen mit Abständen von mehr als 5m keine Ungenauigkeiten bei der Positionierung der Hologramme zu verantworten, können Hologramme mit *Spatial Anchors* versehen werden [20], [21, p. 50f]. Diese Anker werden vom System mit besonderer Rücksicht behandelt und tragen Sorge dafür, dass Hologramme sich kontinuierlich anpassen, also bei Änderungen der Ursprungskoordinaten trotzdem ihre ursprüngliche Position beibehalten [20] [21, p. 51].

Spatial Awareness

Neben der Interpretation menschlicher Gesten verfügt die HoloLens 2 auch über ein *Inside-Out* Umgebungsverständnis, welches das System von ähnlichen Produkten der Konkurrenz (z.B. Google Cardboard, Samsung Gear VR) abhebt. *Inside-Out* Umgebungsverständnis beschreibt das Kartieren, Verstehen und die Selbstlokalisierung innerhalb der Umgebung ohne den Einsatz von extern platzierten Sensoren [22, p. 6]. Ziel dieses Prozess ist es, eine detaillierte Repräsentation der Umgebung und ihrer Oberflächen zu erzeugen, welche es Entwicklern ermöglicht, Hologramme und andere Augmented Reality Objekte nahtlos in die Umgebung der Nutzer zu integrieren. So entsteht eine Immersion, welche es Nutzern gestattet mit diesen digitalen Objekten physisch zu interagieren [23].

Bei der Kartierung der Umgebung stellen sich dabei diverse theoretische Fragen, über den Prozess, welcher die Karte aufzeichnet. Das Problem der parallelen Lokalisierung und Kartierung unterliegt grundsätzlich der Frage, ob es für einen Roboter / ein System möglich ist eine unbekannte Umgebung konsistent zu kartieren, während er / es sich in derselben aufhält und selbstständig seine / ihre Position bestimmt [24]. Ein Problem dessen Lösung auf die 1986 IEEE Robotics and Automation Conference in San Francisco zurückgeht. Eine Zeit während welcher probabilistische Methoden Schritt für

Schritt in die Robotik eingeführt worden sind [25]. Grundsätzlich ähneln sich die meisten Lösungen des SLAM Problems dahingehend, dass der Roboter seine Umgebung inkrementell erweitert, indem er die iterativen Aufzeichnungen übereinanderlegt und markante Landmarken miteinander abgleicht, um die an unterschiedlichen Punkten aufgezeichneten Daten miteinander zu vereinen und so eine einheitliche Karte zu erzeugen. Da die Daten relativ zur aktuellen Position des Roboters aufgezeichnet werden, stellt der Roboter die zentrale Referenz des zugrundeliegenden Koordinatensystems in jeder Aufnahme dar [25]. Die Entwicklung autonom agierender Roboter wurde durch diese Lösung um ein Vielfaches vorangetrieben und kann so im Bereich der Robotik als einer der größten Erfolge der letzten Jahrzehnte gesehen werden. So greift die SLAM-Implementierung der HoloLens 2 auf die diversen in Kapitel 2.1 vorgestellten Sensoren zurück, um Landmarken und Umgebungsmerkmale aufzuzeichnen. Von den acht Kameras der HoloLens werden fünf zum Erkennen der Umgebung (vier für das Erkennen der Umgebung und Hände, eine um die Tiefe der Bilder zu kalkulieren) und zwei für das Verfolgen der Augen verwendet. Aus diesen aufgezeichneten Daten erstellt die Hololens 2 in Echtzeit ein detailliertes 3D Modell ihrer Umgebung. Microsoft nennt seine Lösung für das SLAM Problem *Spatial Mapping* und verwendet für die Repräsentation der Umgebung Vektoren zur Abbildung der aufgezeichneten Daten. Diese Daten (folgend auch *Spatial Mesh* genannt) dienen der Hololens als Positionierungsgrundlage für digitale Inhalte [22, p. 6f]. Ein beispielhaftes Abbild des *Spatial Mesh* eines Raumes kann in Abbildung 4 beobachtet werden.

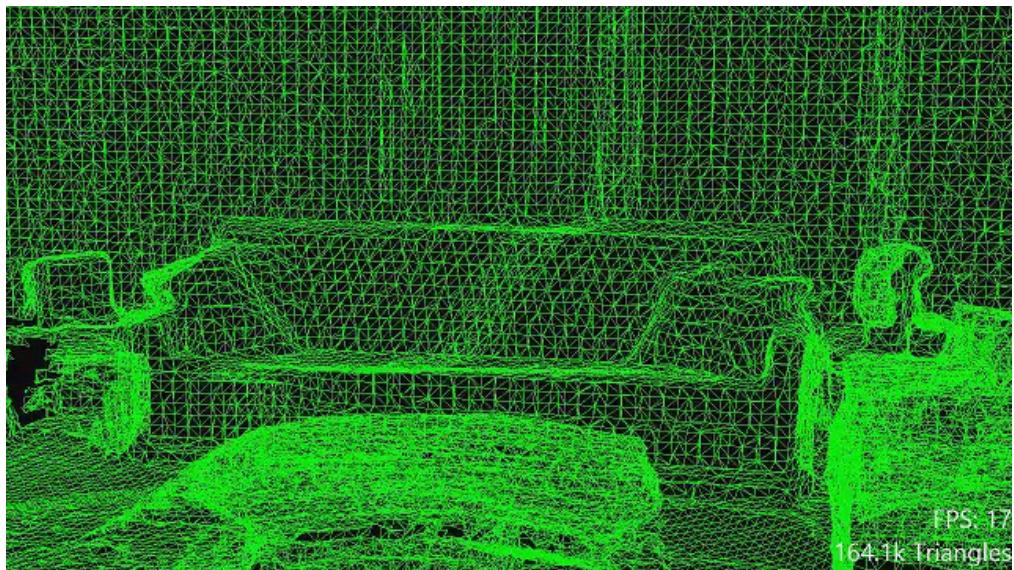


Abbildung 4 Visualisierung des von der Hololens erkannten Umfelds [26]

Dabei wird das Spatial Mesh mit jedem Einsatz der Hololens 2 iterativ weiterentwickelt, da neue und alte Scans miteinander verrechnet werden, um ein möglichst detailliertes Abbild der Umgebung zu schaffen. Aufbauend auf dieser Datengrundlage, setzt Microsoft diverse KI-Systeme ein, um eine Interpretation der Umgebung zu erzeugen, anhand welcher sich weitere Systeme orientieren können. So weiß die HoloLens, ob es sich bei einem Gegenstand um eine Wand einen Tisch oder einen Stuhl handelt und kann diese Information verwenden, um holographische Inhalte präziser platzieren [27].

3 Grundlagen des Sensormodul

3.1 Bestandteile und Anforderungen des Moduls

Im folgenden Abschnitt wird auf die Bestandteile des Sensor-Moduls aus der Tabelle „Tabelle 1 Liste der Funktionalitäten“ eingegangen, Anforderungen an die einzelnen Bestandteile dargestellt sowie auch das Zusammenspiel der einzelnen Bestandteile des Sensor-Modul dargestellt. Wie schon im Abschnitt „Motivation“ beschrieben wird mit der Feuerwehr des Frankfurter Flughafens kooperiert.

So sind die Anforderungen hauptsächlich aus den Gesprächen mit der Frankfurter Flughafen Feuerwehr hervorgegangen.

Zusammenspiel der einzelnen Teile

Die Stromversorgung wird an dem Basisgerät angeschlossen und versorgt dieses mit genug Strom. Die Sensoren der Sensoren-Schnittstelle werden ebenfalls an dem Basisgerät angeschlossen. Sowie auch die Wärmebildkamera der Wärmebildkamera-Schnittstelle. Auf dem Basisgerät laufen die Software der Sensoren-Schnittstelle sowie auch die Software der Wärmebildkamera-Schnittstelle. Über die Software werden die Sensoren sowie die Wärmebildkamera ausgelesen und anschließend kabellos an die HoloLens übertragen. Dieser Vorgang ist noch einmal übersichtlich in der Abbildung „Abbildung 5: Übersicht Sensor-Modul“ einzusehen.

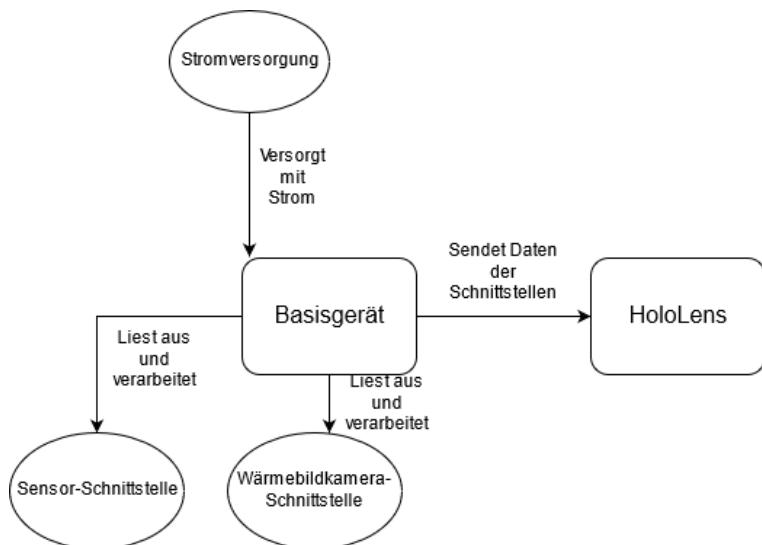


Abbildung 5: Übersicht Sensor-Modul

Basisgerät

Das Basisgerät ist die Grundlage für das Sensor-Modul. Es hat die Aufgabe die verschiedenen Sensoren auslesen, die von den Sensoren ausgelesenen Daten zu verarbeiten und schließlich an die HoloLens weiterzuleiten. Die Anforderungen an das Basisgerät werden in der folgen Tabelle „Tabelle 2: Anforderungen Basisgerät“ aufgelistet und beschrieben.

Anforderung	Beschreibung
Energieeffizienz	Das Basisgerät selbst muss möglichst wenig Strom verbrauchen.
Genug Rechenleistung	Das Basisgerät muss genug Rechenleistung haben, um alle nötigen Aktionen durchführen zu können
Platzsparende Größe	Das Basisgerät selbst muss möglichst klein sein, damit eine Einsatzkraft nicht bei der Arbeit behindert wird und sich unbeschränkt bewegen kann.
Genug Schnittstellen	Das Basisgerät muss genug Schnittstellen für alle Sensoren sowie auch der Übertragung der Daten besitzen, um alle nötigen Sensoren anschließen zu können.
Einfachere Erweiterbarkeit	Das Basisgerät muss möglichst einfach um weitere Bestandteile erweiterbar sein.

Tabelle 2: Anforderungen Basisgerät

Stromversorgung

Natürlich muss das Basisgerät auch mit Strom versorgt werden. Dafür ist die Stromversorgung zuständig. In der folgenden Tabelle „Tabelle 3: Anforderungen Stromversorgung“ werden die Anforderungen an die Stromversorgung aufgelistet und beschrieben.

Anforderung	Beschreibung
Mobilität	Die Stromversorgung soll mobil stattfinden. So soll diese ohne Stromkabel auskommen z.B durch Akkus/Batterien, um eine Bewegungsfreiheit einer Einsatzkraft zu gewährleisten.
Komplexität	Die Stromversorgung muss möglichst die Komplexität des gesamten Sensor-Moduls kleinhalten. So soll die Stromversorgung mit möglichst wenigen Kabeln zu dem Basisgerät auskommen.
Austauschbarkeit	Die Stromversorgung muss möglichst einfach austauschbar sein, falls diese im Falle eines Einsatzes nicht betriebsbereit wäre und somit gegen eine zweite Ersatzstromversorgung austauschbar wäre z.B durch Akkus/Batterien.

Tabelle 3: Anforderungen Stromversorgung

Sensoren-Schnittstelle

Die Sensoren-Schnittstelle baut auf dem Basisgerät auf. Zur Sensoren-Schnittstelle gehören das Auslesen der Sensoren, die Weiterverarbeitung der ausgelesenen Daten sowie wie Weiterleitung der Daten an die HoloLens.

In der folgenden Tabelle „Tabelle 4: Anforderungen Sensor-Schnittstelle“ werden die Anforderungen an die Sensoren-Schnittstelle aufgelistet und beschrieben.

Anforderung	Beschreibung
Temperatur-Sensor	Der Temperatur-Sensor soll Temperaturen im Bereich von bis zu 400 Grad messen können. Für konventionelle Brände reicht dieser der Feuerwehr des Frankfurter Flughafens vom Temperaturbereich in vielen Fällen aus.
Sauerstoff-Sensor	Der Sauerstoff-Sensor muss den Sauerstoffgehalt in der Luft messen können.
Kohlenstoffmonoxid-Sensor	Der Kohlenstoffmonoxid-Sensor muss den Kohlenstoffmonoxidegehalt in der Luft messen können
Sensor-Genauigkeit	Die Sensoren müssen annähernd genau genug sein, um einen Einsatz durch falsche Werte nicht zu gefährden.
Sensor-Warnungen	Bei Überschreitungen von Grenzwerten muss eine Warnung vom Sensor-Modul an die HoloLens übergeben werden.
Datenübertragung	Es muss eine Datenübertragung an die HoloLens stattfinden. Diese muss schnell und zuverlässig stattfinden. Wenn möglich auch kabellos, um die Komplexität möglichst gering zu halten.

Tabelle 4: Anforderungen Sensor-Schnittstelle

Wärmebildkamera-Schnittstelle

Zu der Wärmebildkamera-Schnittstelle gehört die Einbindung der Wärmebildkamera, das Auslesen in Form eines Frames und Weiterleitung eines Frames an HoloLens. Dazu gibt es die in der folgenden Tabelle „Tabelle 5: Anforderungen Wärmebildkamera-Schnittstelle“ gelistete Anforderungen.

Anforderung	Beschreibung
Sehfeld	Die Wärmebildkamera muss ein möglichst großes Sichtfeld abdecken, damit möglichst viel auf einem Blick eingefangen werden kann.
Auflösung	Die Auflösung der Wärmebildkamera muss hoch genug sein, um Objekte und Personen voneinander differenzieren zu können.
Größe	Die Wärmebildkamera muss möglichst klein, damit diese an der HoloLens anbringbar ist und dabei die Einsatzkraft nicht beim Einsatz behindert.
Anschlussmöglichkeit	Die Wärmebildkamera muss mit dem Basisgerät kommunizieren können.

Tabelle 5: Anforderungen Wärmebildkamera-Schnittstelle

3.2 Prototyp

Bei dem ersten Prototyp wird zum Bestimmen des finalen Sensor-Moduls möglich günstige Hardware verwendet.

Ziel ist die Bestimmung Mindestanforderungen für ein funktionierendes Sensor-Moduls zu bestimmen für die Zusammenarbeit mit der HoloLens.

Die Wärmebildkamera-Schnittstelle wird in dem Abschnitt ausgelassen. Während der Aufstellung des Prototyps wurde diese noch nicht eingeplant.

Die einzelnen Bauteile des Prototyp ist werden in den nächsten Abschnitten erläutert und anschließend wird eine Evaluation zu dem Prototyp durchgeführt. Zu dem Prototyp wird die Implementierung selbst nicht beachtet. Diese wird lediglich an mehreren Stellen erwähnt.

Sensoren Basisgerät

Bei dem Prototyp wird der ESP8266Mod12-F-Mikrokontroller als Basisgerät verwendet. Dieser ist ein leistungsfähiger und energieeffizienter Mikrocontroller. Er ist mit einer 802.11 b/g/n WLAN und integrierter 20 dBm-Antenne ausgestattet.

Der Mikrokontroller kann einfach per Lua-Script oder Arduino-Code programmiert werden.

Dabei ist dieser nicht all zu groß (vgl. „Abbildung 6: ESP8266Mod12-F Mikrocontroller“) und mit einem Kostenpunkt von ca. fünf Euro auch nicht all zu teuer.

Der ESP8266Mod12-F-Mikrokontroller bietet mit seinen 16 General-Input-Output-Pins (GPIO) einige Anschlussmöglichkeiten für Geräte und Sensoren.

Die Idee ist es durch den ESP8266Mod12-F-Mikrokontroller ein WLAN-Netzwerk aufzubauen. Mit diesem wird die HoloLens verbunden. Somit können jegliche Daten kabellos übertragen werden. [28]

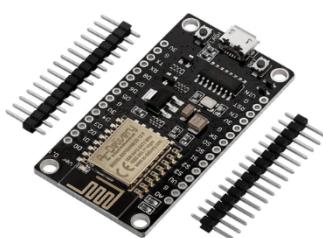


Abbildung 6: ESP8266Mod12-F Mikrocontroller [29]

Stromversorgung

Für die Stromversorgung des Basisgerätes wird in dem Prototyp eine einfache Powerbank verwendet. Diese hat den Vorteil, dass diese in verschiedenen Größen existiert. So kann je nach Länge des Einsatzes eine verschieden große Powerbank eingesetzt werden.

Sensoren-Schnittstelle

Bei dem Prototyp werden für die Sensor-Schnittstelle der MQ9-Sensor, MQ135-Sensor und DHT11-Sensor verwendet.

Der MQ9-Sensor ist ein Gas Sensor. Dieser kann Kohlenstoffmonoxid und entflammbare Gase wahrnehmen. Der Wahrnehmungsbereich des Sensors liegt zwischen 10 bis 1000ppmm. 100ppmm gilt dabei in der Luft als Normalgehalt.

Die Ausgangsspannung steigt mit zunehmender Konzentration der gemessenen Gase. Der Sensor hat eine kurze Reaktionszeit bei Werteänderungen. [30]



Abbildung 7: MQ9-Sensor [31]

Der MQ135-Sensor ist ebenfalls ein Gas Sensor. Dieser kann den Sauerstoffgehalt der Luft messen.

Die Ausgangsspannung steigt mit zunehmender Konzentration der gemessenen Gase. Der Sensor hat eine kurze Reaktionszeit bei Werteänderungen. [32]



Abbildung 8: MQ135-Sensor [33]

Der DHT11 ist ein kostengünstiger, zuverlässiger digitaler Temperatur- und Feuchtigkeitssensor.

Mittels kapazitivem Feuchtigkeitsfühler und Heißleiter misst der Sensor umliegende Luftdaten und gibt diese Werte als ein digitales Signal an den Datenpin weiter. [34]

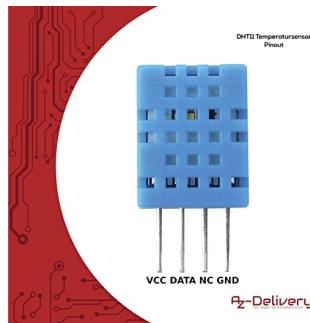


Abbildung 9: DHT11-Sensor [35]

Bei einem Kostenpunkt von ca. fünf Euro je Sensor sind sehr preiswert ersteigbar. Die Datenübertragung geschieht wie schon beschrieben auf dem Basisgerät kabellos über ein Hotspot durch eine schlanke REST-API statt.

Evaluation und Fazit zu dem Prototyp

In diesem Abschnitt werden die ganzen Anforderungen an die einzelnen Bestandteile nochmals in einer Tabelle gelistet (vgl. „Tabelle 6: Überprüfung Erfüllung Anforderungen“) und bewertet ob diese erfüllt sind. Die Bewertung findet auf Basis von eigenen Tests sowie der eigenen Wahrnehmung statt. Die Tests zum Prototyp werden nicht verschriftlicht. Daraus wird ein Fazit gezogen, was für das für das richtige System zu verbessern gilt.

Anforderung	Bauteil	Erfüllt
Energieeffizienz	Basisgerät	Ja, da der Verbrauch sehr niedrig ist.
Genug Rechenleistung	Basisgerät	Nein, da die Rechenleistung für noch mehr Aufgaben nicht reichen wird.
Platzsparende Größe	Basisgerät	Ja, da dieses selbst sehr klein ist.
Genug Schnittstellen	Basisgerät	Nein, die Anschlussmöglichkeiten reichen nicht aus.
Einfachere Erweiterbarkeit	Basisgerät	Ja, viele verschiedene Geräte anschließbar
Mobilität	Stromversorgung	Ja. Durch verschiedene Größen.
Komplexität	Stromversorgung	Ja.
Austauschbarkeit	Stromversorgung	Ja durch andere Powerbank.
Temperatur-Sensor	Sensor-Schnittstelle	Ja.
Sauerstoff-Sensor	Sensor-Schnittstelle	Ja.
Kohlenstoffmonoxid-Sensor	Sensor-Schnittstelle	Ja.
Sensor-Genauigkeit	Sensor-Schnittstelle	Nein, nicht genau genug. Werte springen zu sehr.
Sensor-Warnungen	Sensor-Schnittstelle	Ja, bei Wert Überschreitung bzw. Unterschreitung
Datenübertragung	Sensor-Schnittstelle	Ja, durch Hotspot und schlanke REST-API

Tabelle 6: Überprüfung Erfüllung Anforderungen

Die Tabelle „Tabelle 6: Überprüfung Erfüllung Anforderungen“ zu den Anforderungen zeigt, dass es Verbesserungsmöglichkeiten an dem Prototypen gibt. Somit wird ein anderes Basisgerät mit genug Rechenleistung und möglichst kleiner Größe, eine andere Stromversorgungsmöglichkeit sowie bessere Sensoren benötigt.

3.3 Verbessertes Sensor-Modul

Um die an am Prototyp erkannten Mängel zu beheben wurden die Bestandteile des Sensor-Moduls durch deutlich bessere, teurere und leistungsfähigere Hardware ausgetauscht. Diese neuen Teile und deren Verbesserungen gegenüber dem Prototyp werden in den nächsten Abschnitten betrachtet.

Basisgerät

In dem verbesserten Sensor-Modul wird der Raspberry Pi als Basisgerät verwendet. Das Modell ist der Raspberry Pi Zero W. Der Raspberry Pi ist ein Einplatinencomputer und wurde von der britischen Raspberry Pi Foundation entwickelt und veröffentlicht. Der Raspberry hat ein Ein-Chip-System (SoC) von Broadcom mit einer Arm-CPU. Die Platine hat die Größe und das Format einer Kreditkarte (siehe „Abbildung 10: Raspberry Pi Zero W“. Der Raspberry Pi kam Anfang 2012 auf den Markt. Sein großer Markterfolg wird teils als Revival des bis dahin weitgehend bedeutungslos gewordenen Heimcomputers zum Programmieren und Experimentieren angesehen. Der im Vergleich zu üblichen Personal Computern sehr einfach aufgebaute Rechner wurde von der Stiftung mit dem Ziel entwickelt, jungen Menschen den Erwerb von Programmier- und Hardware-Kenntnissen zu erleichtern. Entsprechend niedrig wurde der Verkaufspreis angesetzt, der je nach Modell etwa 5 bis 100 Euro beträgt. Es ist der meistverkaufte britische Computer: Bis Februar 2022 wurden über 45 Millionen Geräte verkauft. Es existiert ein großes Zubehör- und Softwareangebot für zahlreiche Anwendungsbereiche. [36] Um die Kosten des Pi Zero niedrig zu halten, sind der Prozessor und der Arbeitsspeicher recht einfach gehalten. So wird ein Single-Core ARM mit 1 GHz verwendet und stehen 512 MB RAM zur Verfügung. Über eine Mikro-SD-Karte wird der Raspberry Pi Zero W mit einem Betriebssystem versorgt. Der Einplatinencomputer hat ein Mini-HDMI-Anschluss und zwei Mikro-USB-Anschlüsse und 40 zur Verfügung stehende GPIO-Pins. [37]

Der Raspberry Pi Zero W hat nun die von dem Basisgerät des Prototyp fehlenden Anschlussmöglichkeiten sowie auch die weitere Rechenleistung, welche für die Wärmebildkamera benötigt wird. Dabei bleiben die anderen Anforderungen aus der Tabelle „Tabelle 2: Anforderungen Basisgerät“ weiterhin erfüllt.



Abbildung 10: Raspberry Pi Zero W [38]

Stromversorgung

Zur Stromversorgung des Basisgerätes des verbesserten Sensormodules wird ein The Li-ion Battery HAT von Waveshare verwendet. Der Li-Ion Battery HAT integriert den SW6106 Powerbank-Management-Chip und ermöglicht die Bereitstellung einer geregelten 5V-Stromversorgung. Die nötige Leistung kommt aus einer 14500-Batterie. Dadurch wird der Pi zu einem tragbaren Gerät wird (vgl. „Abbildung 11: Battery Hat“). Es lädt den Akku auch auf und unterstützt bi-direktionale Schnellladung. Dadurch, dass der HAT kein Kabel benötigt wurde die Komplexität der Stromversorgung verringert. Dies erfüllt gleichzeitig Anforderung Komplexität. Ebenso ist die Anforderung der Austauschbarkeit durch die 14500-Batterie gewährleistet. Die Anforderung Mobilität ist ebenfalls gewährleistet. [39]



Abbildung 11: Battery Hat [40]

Sensor-Schnittstelle

Zum Anschließen der Sensoren an das neue Basisgerät wird der Grove Base Hat verwendet. Der Grove Base Hat für Raspberry Pi Zero bietet einen Digital/Analog/I2C/PWM/UART-Port, um alle Ihre Anforderungen zu erfüllen. Mit Hilfe der eingebauten MCU ist auch ein 12-Bit 6-Kanal ADC für Raspberry Pi verfügbar (vgl. „Abbildung 12: Grove Base Hat“).

Derzeit werden mehr als 60 Grove-Sensoren von Base Hat für Raspberry Pi unterstützt.

Im Vergleich zu Grove Pi+ verwendet der Grove Base Hat für Raspberry Pi Zero nicht den ATMEGA-Chip für die Datenkonvertierung. Dadurch ist der Grove Base Hat für Raspberry Pi Zero wesentlich kostengünstiger. [41]

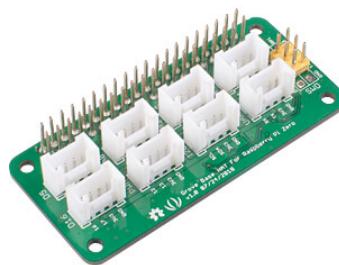


Abbildung 12: Grove Base Hat [42]

Zum Messen der Temperatur wird ein Grove High Temperatur Sensor verwendet (vgl. „Abbildung 13: Grove High Temperature Sensor“).

Der Grove Hochtemperatursensor verwendet ein Thermoelement vom Typ K und einen Thermoelementverstärker, der die Umgebungstemperatur mit Hilfe eines Thermistors zur Kaltstellenkompensation misst. Der Erfassungsbereich dieses Sensors liegt bei -50~600°C und die Genauigkeit beträgt $\pm(2,0\% + 2^\circ\text{C})$. [43]



Abbildung 13: Grove High Temperature Sensor [44]

Die Messung des Sauerstoffwertes erfolgt über ein Grove-Sauerstoffsensor vom Modell MIX8410 (vgl. „Abbildung 14: Grove Oxygen Sensor“). Der Grove-Sauerstoffsensor (MIX8410) ist ein Sensor zum Testen der Sauerstoffkonzentration in der Luft. Dieser basiert auf dem Prinzip der elektrochemischen Zelle. Die elektromagnetischen Zellen können die aktuelle Sauerstoffkonzentration klar erkennen, wenn Sie Spannungswerte proportional zur Sauerstoffkonzentration ausgeben und sich auf die lineare Kennlinie der Sauerstoffkonzentration beziehen. Der Sensor ist geeignet für die Erkennung der Sauerstoffkonzentration in der Umwelt zum Schutz von Personen. Der Grove-Sauerstoffsensor (MIX8410) ist ein organisches Reaktionsmodul, das einen kleinen Strom liefern kann, wenn es in die Luft gebracht wird. Es wird keine externe Stromversorgung benötigt. [45]



Abbildung 14: Grove Oxygen Sensor [46]

Zur Messung von Kohlenstoffmonoxidwerten wird ein Grove Kohlenstoffmonoxid-Sensor vom Modell MQ9B (vgl. „Abbildung 15: Grove CO Sensor“) verwendet. Das Grove Kohlenstoffmonoxid-Sensor(MQ9) ist nützlich für die Erkennung von Gas Lecks. Er eignet sich für die Erkennung von LPG, CO, CH₄. Durch seine hohe Empfindlichkeit und schnelle Reaktionszeit können Messungen in Echtzeit durchgeführt werden. [47]



Abbildung 15: Grove CO Sensor [48]

Das neue Sensor-Modul erfüllt die Anforderungen komplett. So sind diese laut Hersteller genauer als die vorherigen Sensoren. Die ganzen anderen Anforderungen von vorher sind auch erfüllt.

Wärmebildkamera

Dem Einsatzhelm liegt eine Infrarotkamera Zugrunde, welche zur Detektion von Brandherden und Personen eingesetzt werden soll. Infrarotkameras arbeiten unterhalb des roten Endes des sichtbaren Farbspektrums, einer Region zwischen dem sichtbaren Licht und der Mikrowellenregion des elektromagnetischen Spektrums, welche auch Hitzeregion genannt wird, da alle Objekte über dem absoluten Null von -273.15°C bspw. Menschen, flammende Gase oder verwendete Computerchips eine gewisse Menge Infrarotenergie aussenden. Dabei gilt: Desto heißer das Objekt, desto höher die Infrarotenergie [49]. Strahlung innerhalb des Infrarotbereichs bewegt sich ungefähr auf einer Wellenlänge von 0.74 bis 14µm, wobei sich dieser Spektralbereich weiter in die Segmente *Near Infrared* (NIR), *Short-Wave Infrared* (SWIR), *Mid-Wave Infrared* (MWIR) und *Long-Wave Infrared* (LWIR) unterteilen lässt [49]. Es lässt sich jedoch festhalten, dass nicht alle Segmente des Infrarotspektrums für Detektionssysteme wie Wärmebildkameras geeignet sind, da ihre Strahlung selbst durch Materialien wie Wasser oder Kohlenstoffdioxid abgeschirmt wird [49].

Moderne *Forward Looking (Infrared)* (FLIR) *Long-Wave Infrared* Wärmebildkameras arbeiten auf dem Lichtbereich zwischen 8µm – 14µm und können so die Wärmesignaturen diverser Subjekte registrieren, welche sich deutlich von ihrem Hintergrund unterscheiden (e.g. Ein Mensch im Gerstenfeld). Sie lassen sich unterteilen in die Kategorien *gekühlt* und *ungekühlt*. So arbeiten ungekühlte LWIR Kameras mit einem Sensor aus diversen Widerstandspixeln, deren Resistenz sich an die detektierte Strahlung anpasst, da sich die Widerstände über Zeit aufheizen. Aus der Veränderung dieses messbaren Widerstands konstruiert ein ungekühltes LWIR Mikrobolometer ein Bild, welches anhand der Grauwerte surjektiv auf ungefähre Temperaturen in den einzelnen Bereichen (Pixeln) des Bildes schließen lässt [49]; Die Menge der Temperaturen ist größer als die der Abbildbaren Grauwerte. Gekühlte LWIR Detektionssysteme arbeiten mit Photodetektoren, welche mit Hilfe von Halbleitermaterialien Photonen in Elektronen-Loch-Paare umwandeln und so aus Lichtwellen elektrische Signale generieren. Hierbei wird die operative Temperatur der Detektoren unter Verwendung von Kryokühlleinheiten auf ein Niveau gekühlt, bei welchem Hintergrundrauschen der eigenen Sensorik unter dem der aufzunehmenden Szene liegt. So können klarere und genauere Wärmebilder erzeugt werden [49], [50]. Hierbei generieren gekühlte LWIR Detektoren zwar genauere Bilder, sind aber unhandlicher und teurer. So kommen sie für den Bau des Mars 1 Prototypen nicht in Frage und es wird auf ein ungekühltes LWIR System

gesetzt. Nach intensiver Rücksprache mit der Feuerwehr ist die *Teledyne FLIR Lepton* 3.5 für die Implementierung des Sensormoduls ausgewählt worden, da sich diese in ihrer Größe und Auflösung an den kommunizierten Anforderungen der Feuerwehr orientiert.



Abbildung 16 Teledyne FLIR Lepton 3.5 ohne I/O Module in Vergleich zu einem kleinen Geldstück [51]

4 Implementierung Sensormodul

In den nächsten Absätzen wird das Vorgehen zur Implementierung des Sensor-Moduls beschrieben. Dies bezieht sich hauptsächlich auf die Software.

4.1 Installation Raspberry Pi Betriebssystem

Vor der Implementierung der Sensor-Schnittstelle muss ein Betriebssystem für den Raspberry Pi ausgewählt und installiert werden. Zur Auswahl stehen verschiedene Betriebssysteme. Diese sind unter anderem Raspberry Os, Ubuntu Server und DietPi. Die Wahl fällt auf Raspberry OS mit dem Buster Release der Raspberry Pi Foundation. Dieses ist für den Raspberry Pi Zero W am besten optimiert. Ebenfalls läuft das Programm zum Auslesen der Sensoren nur unter diesem Betriebssystem. Für die Installation wird zuerst eine microSD-Karte in den Computer einge-steckt. Anschließend wird der offizielle Raspberry Pi Imager heruntergeladen und in-stalliert. Der Raspberry Pi Imager ist für die Plattformen Windows, macOS oder Linux verfügbare und kann das neueste Raspberry Pi Betriebssystem herunterladen und in-stallieren. Es gibt auch andere Möglichkeiten, dies zu tun, nämlich das Herunterladen einer Raspberry Pi OS-Image-Datei und das anschließende *Brennen* mit einer Drittanbieter-App.

In nächsten Schritt wird auf *Choose OS* (Betriebssystem auswählen) geklickt und Raspberry Pi OS Buster-Release aus dem OS-Menü ausgewählt.

Danach wird auf SD-Karte auswählen geklickt und die verwendete Karte ausgewählt. In dem vorletzten Schritt wird über *Strg + Shift + X* die erweiterten Optionen geöffnet und SSH und WLAN aktiviert sowie auch das WLAN konfiguriert.

Im letzten Schritt wird auf Schreiben geklickt. Die App braucht nun ein paar Minuten, um das Betriebssystem herunterzuladen und auf die Karte zu schreiben. Nach der Meldung, dass das Betriebssystem erfolgreich installiert wurde kann diese aus dem PC entfernt und in dem Raspberry eingesteckt werden.

4.2 Sensoren-Schnittstelle

Vor der Implementierung der Sensor-Schnittstelle müssen die nötigen Pakete installiert werden. Diese sind Python, Python-Pip GrovePi, Flask sowie auch Flask-RESTful.

Zuerst wird das Betriebssystem aktuell gehalten über den Befehl *apt update & apt upgrade*.

Das Paket Python3 wird über den Befehl *apt install python3 -y*, Python-Pip3 über den Befehl *apt install python-pip*, Flask über *pip3 install flask*, Flask-Restful über den Befehl *pip3 install flask-restful* und GrovePi über den Befehl *curl -kL dexterindustries.com/update_grovepi | bash* installiert.

Nach der Installation der Programme ist es nun möglich die Sensoren-Schnittstelle zu implementieren. Diese wird in Python umgesetzt und besteht aus drei Teilen.

Diese sind *groveRest* für die REST-Schnittstelle, *GroveSensorController* zum Kontrollieren der Werte des Sensormoduls und der Klasse *GroveSensorRead* zum Auslesen der Sensoren selbst.

REST steht für REpresentational State Transfer, API für Application Programming Interface. Gemeint ist damit eine Programmierschnittstelle, die sich an den Paradigmen und Verhalten des World Wide Web (WWW) orientiert und einen Ansatz für die Kommunikation zwischen Client und Server in Netzwerken beschreibt.

GroveSensorRead

Der erste Teil der Sensor-Schnittstelle ist zuständig für das Auslesen der Sensoren. Für das Sensor-Modul werden Grove-Sensoren verwendet. So werden diese auch über die Grove-Bibliothek ausgelesen.

```
import time , sys, math
import grove_hightemperature_sensor as grovepi
import base64
import cv2
import serial
from adc import ADC
import gc
class GroveSensorRead():
```

Zunächst erfolgen die benötigten Imports für den Teil der Schnittstelle sowie das Anlegen der Klasse. Diese sind *time*, *sys*, *math*, *grove_hightemperature_sensor*, *base64*, *cv2*, *serial*, und *adc*.

```
def readCarbonmonoxid(adc):
    channel = 2
    value = adc.read(channel)
    return value
```

Die Methode *readCarbonmonoxid* ist zuständig für das Auslesen des Kohlenstoffmonoxid- Sensors. Dazu wird über den übergebenen Analog-Digital-Converter der Sensor im Channel zwei, was dem Port A2 auf dem Grove-Hat gleich kommt ausgelesen und zurückgegeben. Es sind keine weiteren Berechnungen nötig.

```
def readOxygen(adc):
    VRefer = 3.3
    total = 0
    Measuredvout = 0
    channel = 4
    value = adc.read(channel)
    if value != 0:
        voltage = value*3.3/1024.0
        Mix8410Value = voltage*0.21*100/2.0
        value = round(Mix8410Value,2)
    else:
        value = 0
    return value
```

Die Methode *readOxygen* ist zuständig für das Auslesen des Sauerstoff-Sensor. Dazu wird zunächst über den übergebenen Analog-Digital-Converter der Sensor im Channel vier ausgelesen. Solange dieser Wert nicht null beträgt wieder dieser mal 3.3 gerechnet und durch 1024 geteilt um die Spannung des Sensor zu bestimmen. Anschließend wird der Spannungswert mal 0,21 mal 100 genommen und durch 2 geteilt. Damit erhält man schließlich die Sauerstoffkonzentration in Prozent. Der Prozentwert wird noch auf zwei Nachkommastellen gerundet und zurückgegeben.

```

def readTemperature(arduinoSerialData):
    room_temperature_pin = 15
    probe_temperature_pin = 14
    sensor = grovepi
        .HighTemperatureSensor(room_temperature_pin,probe_temperature_pin)
    room_temperature = sensor.getRoomTemperature()
    probe_temperature = sensor.getProbeTemperature()
    return probe_temperature

```

Die Methode *readTemperature* ist zuständig für das Auslesen des High Temperature Sensors. Dies ist der normale Weg den Sensor mit Hilfe der GrovePi-Bibliothek auszulesen. Dazu müssen die Pins definiert werden. Diese sind Pin 14,15 und liegen auf dem Grove-Base-Hat auf dem Port A0. Anschließend wird über die Pins ein Sensor-Objekt angelegt, welches ausgelesen und zurückgegeben wird. Dies hat in der Praxis nicht funktioniert. So wurde für das Testen eine Ersatzlösung ausgedacht.

```

myData = ""
i=0
while(True):
    if (arduinoSerialData.inWaiting()>0):
        myData = arduinoSerialData.readline().decode("ascii")
        myData=myData.strip().replace("a","")
        myData=myData.replace("=", "")
        if(myData[0].isdigit()):
            myData = float(myData)
        break
return myData

```

Um das Problem zu lösen wurde ein Arduino Nano mit dem Sensor an den Raspberry Pi angeschlossen. Die Serielle Schnittstelle des Arduinos wird nun in der Methode ausgelesen. Dies geschieht in einer Dauerschleife. So wird mit *readLine()* die Serielle Schnittstelle ausgelesen, der Wert zu ASCII decodiert und unnötige Zeichen aus dem Ausgelesenen entfernt. Sobald eine Typenprüfung eine Zahl ergibt, wird die Schleife beendet und Wert als Float-Wert zurückgegeben.

```

HighTemp ht(A1, A0);
void setup()
{
    Serial.begin(115200);
    ht.begin();
}
void loop()
{
    Serial.println(ht.getThmc());
    delay(100);
}

```

Der Quellcode darüber ist der Sketch, welcher für den Workaround auf dem Arduino Nano läuft. Dazu wird mit den analogen Pins A1 und A0 ein Hightemp Objekt initialisiert. Bei einem Arduino läuft das Programm in zwei Schritten ab. Diese sind *Setup*, welche ausgeführt wird, sobald der Arduino gestartet wird und *Loop* welche anschließend dauerhaft ausgeführt wird. In dem *Setup* werden die Messungen des Sensors gestartet sowie auch die serielle Konsole gestartet. In der *Loop* wird alle 100 Millisekunden der Wert des Sensors auf die serielle Konsole geschrieben, welche anschließend durch den vorherig beschriebenen Teil ausgelesen wird.

GroveSensorController

Der zweite Teil der Schnittstelle dient der Einordnung von Werten in Kategorien und dem zusammenstellen der Rückgabe für *groveRest*.

```

from GroveSensorRead import GroveSensorRead
from flask import Flask, make_response
class GroveSensorController():

```

Im ersten Schritt erfolgen die Importe dieses Teils der Sensor-Schnittstelle sowie auch das Anlegen der Klasse von *GroveSensorController*. Diese sind *GroveSensorRead*, *Flask* und *make_response*.

```

def readCarbonmonoxid(adc):
    carbonValue = GroveSensorRead.readCarbonmonoxid(adc)
    statusCarbon = str(' ')
    if carbonValue > 0 and carbonValue is not None:
        if carbonValue > 400:
            statusCarbon = 'Danger'
        else:
            statusCarbon = 'Okay'
    else:
        statusCarbon = 'Error with CO-Sensor'

    return {'CarbonValue' : carbonValue, 'StatusCarbon' : statusCarbon}

```

Die Methode *readCarbonmonoxid* ist zuständig für das Aufbauen der Antwort für eine Anfrage über REST bezüglich der Kohlenstoffmonoxid Werte. Dazu wird zunächst über *readCarbonmonoxid* der Sensorwert ausgelesen. Ist dieser größer als null wird geprüft ob der Wert Größer als 400 ist. Ist dies der Fall wird ein Status von *Danger* verwendet und im Gegenfall wird Status *Okay* verwendet. Sollte der Wert kleiner gleich null oder None sein wird der Status auf *Error* gesetzt. Im letzten Schritt erfolgt das Aufbauen der Antwort der REST-Abfrage in JSON-Format mit den Felder *CarbonValue* und *StatusCarbon*.

```

def readOxygen(adc):
    oxygenValue = GroveSensorRead.readOxygen(adc)
    statusOxygen = str(' ')
    if oxygenValue > 0 and oxygenValue is not None:
        if oxygenValue > 8:
            statusOxygen = 'Okay'
        else:
            statusOxygen = 'Danger'
    else:
        statusOxygen = 'Error with O2-Sensor'

    return { 'OxygenValue' : oxygenValue, 'StatusOxygen' : statusOxygen}

```

Die Methode *readOxygen* ist ähnlich der vorher beschriebenen Methode *readCarbonmonoxid* aufgebaut. So wird auch hier zunächst der Sensorwert ausgelesen. Ist dieser größer null wird geprüft ob der Sauerstoffwert größer als 8 ist. Falls dies der Fall ist wird der Status auf *Okay* gesetzt Und falls nicht auf *Danger*. Im Fehlerfall wird der Status auf *Error* gesetzt. Im letzten Schritt erfolgt das Aufbauen der Antwort der REST-Abfrage in JSON-Format mit den Felder *OxygenValue* und *StatusOxygen*.

```

def readTemperature(arduinoSerialData):
    tempValue = GroveSensorRead.readTemperature(arduinoSerialData)
    statusTemp = str(' ')
    if float(tempValue) > 0 and tempValue is not None:
        if tempValue > 200:
            statusTemp = 'Danger'
        else:
            statusTemp = 'Okay'
    else:
        statusTemp = 'Error with Temp-Sensor'

    return {'TempValue' : tempValue, 'StatusTemp' : statusTemp}

```

Die Methode *readTemperature* ist ebenfalls ähnlich der vorherigen zwei beschriebenen Methoden. Zunächst wird der Sensorwert ausgelesen. Anschließend erfolgt die Überprüfung der Werte. Sollte der Wert über 200 sein wird der Status auf *Danger* gesetzt. Ansonsten steht der Status auf *Okay*. Bei einem Fehler wird der Status auch hier auf *Error* gesetzt. Im letzten Schritt erfolgt das Aufbauen der Antwort der REST-Abfrage in JSON-Format mit den Felder *tempValue* und *StatusTemp*.

Die Methode *readAll* ist die Kombination aller drei vorherig beschriebenen Methoden. Der Aufbau dieser ist gleich und wird nacheinander ausgeführt und dann zusammen in einer JSON-Antwort zurückgegeben.

groveRest

Der dritte Teil der Sensor-Schnittstelle ist zuständig für das Übertragen der Daten an die HoloLens. Dies wird mit einer REST-Schnittstelle realisiert. Die REST-Schnittstelle wird in Python durch die vorher installierten Bibliotheken Flask und Flask-RESTful zur Verfügung gestellt.

Im Schritt kommen die für den Teil nötigen Importe und Definitionen.

```

from flask import Flask
from flask_restful import Resource, Api
from adc import ADC
from GroveSensorController import GroveSensorController
import cv2
import serial

app = Flask(__name__)
api = Api(app)
thermalCamera = cv2.VideoCapture(0)
arduinoSerialData = serial.Serial('/dev/ttyUSB0', 115200)
adc = ADC()

```

Die nötigen importierten Bibliotheken sind *flask*, *Resource*, *Api*, *ADC*, *GroveSensorController* sowie *cv2* und *serial*.

Zunächst wird über *app* die Basis-Flask-Applikation definiert, welche im nächsten Schritt benötigt wird. Im nächsten Schritt wird über *api* die Flask-Basis-Applikation in eine REST-Schnittstelle umgewandelt. Über *thermalCamera* wird die Wärmebildkamera, über *arduinoSerialData* wird der Workaround für die nicht funktionierende High Temperature Bibliothek angelegt und über *adc* wird ein AnalogDigitalConversion-Objekt angelegt. Die Variablen *thermalCamera*, *arduinoSerial* und *adc* wird im nächsten Schritt benötigt.

```

class ReadAll(Resource):
    def get(self):
        return GroveSensorController.readAll(arduinoSerialData,adc)

class ReadCarbonmonoxid(Resource):
    def get(self):
        return GroveSensorController.readCarbonmonoxid(adc)

class ReadOxygen(Resource):
    def get(self):
        return GroveSensorController.readOxygen(adc)

class ReadTemperature(Resource):
    def get(self):
        return GroveSensorController.readTemperature(arduinoSerialData)

class ThermalCameraStream(Resource):
    def get(self):
        return GroveSensorController.getStream(thermalCamera)

```

In dem über dem Abschnitt gezeigten Quellcode werden die Daten für die Endpunkte der Restschnittstelle zur Verfügung gestellt. Dazu werden für die Endpunkte Klassen mit ihren benötigten HTTP-Anfragemethoden als Funktionen der Klasse angelegt. In

den Funktionen steht definiert, was bei einem Aufruf des Endpunktes mit Anfragemethode passieren muss. Im Falle des Sensormodules gibt es vier Klassen. Diese sind *ReadAll* für das Auslesen der drei Sensoren, *ReadCarbonMonoxid* für das Auslesen des Kohlenstoffmonoxid-Sensors, *ReadOxygen* für das Auslesen des Sauerstoff-Sensors, *ReadTemperature* für das Auslesen des High Temperature Sensors und *ThermalCameraStream* für das Auslesen der Wärmebildkamera. In den Methoden der einzelnen Klassen erfolgt der Aufruf der jeweils zuständigen Methodenaufrufe aus *GroveSensorController* sowie der Rückgabe dieser an den Aufrufer der REST-Schnittstelle.

```
api.add_resource(ReadAll, '/readAll')
api.add_resource(ReadCarbonmonoxid, '/readCarbonmonoxid')
api.add_resource(ReadOxygen, '/readOxygen')
api.add_resource(ReadTemperature, '/readTemperature')
api.add_resource(ThermalCameraStream,'/thermalCameraStream')
```

In dem Quellcode darüber werden die Endpunkte der REST-Schnittstelle hinzugefügt und die von vorher definierten zuständigen Klassen mit ihren Methoden zugewiesen.

```
if __name__ == '__main__':
    app.run(debug=False, port=5000, host='0.0.0.0')
```

Im letzten Teil steht die Start-Methode der REST-Schnittstelle. Diese läuft über den Port 5000 und kann von jedem Gerät im gleichen Netzwerk aufgerufen werden (host="0.0.0.0").

4.3 Wärmebildkamera-Schnittstelle

Ein zentrales Ziel des Einsatzhelms ist es die Mobilität der Nutzer zu steigern, indem die händisch verwendete Wärmebildkamera durch eine in den Helm integrierte Kamera ersetzt wird. Dieser Funktionalität ermöglicht so indirekt die Erweiterung des natürlichen Sichtfeldes der Einsatzkraft um das Wärmebild-Infrarotspektrum. Da auf dem Gebiet der ungekühlten Long-Wave Infrared Kameras, wie beschrieben in Kapitel 3.3 Abschnitt *Wärmebildkamera*, enorme Fortschritte in Bezug auf Größe, Genauigkeit und allgemeine Verfügbarkeit vermerkt werden konnten, ist das Einbinden einer solchen Kamera nicht mehr mit großem implementativen Aufwand und gemindertem Tragekomfort verbunden. Die für das System MARS 1 ausgewählte Kamera *Teledyne*

FLIR Lepton 3.5 ist eine Radiometrie fähige ungekühlte LWIR-Kamera, welche bei einer Pixelgröße von $12\mu\text{m}$ einen Wärmebild-Spektralbereich von $8\mu\text{m}$ bis $14\mu\text{m}$ abdeckt und zwischen Temperaturen von -10°C bis 400°C differenzieren kann [52]. Da innerhalb dieses Spektrums die meisten konventionellen Brandherde sowie Personen identifiziert werden können (siehe Kapitel 1.2 Herangehensweise und Methodik), werden die Anforderungen der Feuerwehr innerhalb eines angemessenen Preisspektrums erfüllt und die Kamera in das System Mars 1 integriert. Dabei wird die Kamera an per Micro-USB Schnittstelle mit dem Raspberry Pi des Sensormoduls verbunden und ein Skript geschrieben, welches auf die Kamera zugreift, das Bild aufbereitet und per Schnittstellenendpunkt bereitstellt. Hierzu werden den Skripts zwei Methoden und zwei Module hinzugefügt.

Name	Funktion
(numpy-bytesarray) generateFrame()	Ruft ein Bild von der Kamera ab und konvertiert dieses in das PNG-Format
(Response) thermalCameraStream()	Sendet das Bild als Content-Type <i>image/png</i> als Response
opencv-python	Ein Packet für diverse Bildverarbeitungsaufgaben mit Python
make_response	Generiert eine http konformes Response Objekt

Tabelle 7 Zusätzliche Funktionen und Imports der Funktionalität Wärmesicht

Indem mittels *opencv* eine Instanz der Klasse *VideoCapture* mit Referenz auf den Index der Wärmebildkamera im Geräteverzeichnis */dev* erstellt wird, können Bilder derselben abgerufen und weiterverarbeitet werden. Da es sich als sehr kompliziert herausgestellt hat, einen Stream aufzubauen und auf denselben im Holo-Modul zuzugreifen, werden Bilder der WBK einzeln über die Schnittstelle abgerufen und auf Anfrage versendet. Da dies bei Aktivierung der Funktionalität eine hohe Abfragelast zur Folge hat, muss der Schnittstellenendpunkt auf die nötigsten Zeilen Code beschränkt werden, um Ressourcen zu sparen. So wird über die in Tabelle 7 eingeführte Methode *generateFrames()* jedes aufgenommene Bild folgend über die von *opencv* implementierte Methode *imencode(type, raw-image-data)* in das PNG-Format konvertiert, um weitere

Inkompatibilitäten zwischen den originalen Bilddaten und der HoloLens zu vermeiden und als Byte-Array zurückgegeben.

```
import cv2 # opencv-python
thermalCamera = cv2.VideoCapture(0)

def generateFrame():
    success, frame = thermalCamera.read()
    if not success:
        raise Exception('Failed to read Thermal Camera image.')
    ret, buffer = cv2.imencode('.png', frame)
    return buffer.tobytes()
```

Damit das Holo-Modul auf diese Methode zugreifen kann, wird ein Schnittstellenendpunkt implementiert, welcher die generierten Bilder auf Abruf bereitstellt. Für diese Erweiterung der Schnittstelle wird zuerst der Pfad definiert, unter welchem der Endpunkt erreichbar sein wird. Folgend wird das generierte Frame unter Verwendung der importierten Methode *make_response* in ein Flask Response-Objekt eingebettet, welches gekennzeichnet durch den Header *Content-Type: image/png* zurückgeschickt wird. Um einem Absturz der Schnittstelle bei unerwarteten Fehlern vorzubeugen wird der Prozess in ein standardisiertes *try/catch* Statement eingebettet; Fehler können hier beispielsweise das Offline gehen der Kamera durch Zerstörung o. ungesicherte Befestigung umfassen. Im Fall eines Fehlers, wird eine Exception abgefangen und auf die Anfrage an den Schnittstellenendpunkt ein Response Objekt mit dem Http-Status-Code 500 und dem textuellen Hinweis *Could not generate Frame* erwidert.

```
from flask import make_response

@app.route('/thermalCameraStream')
def thermalCameraStream():
    response = Response(response="Could not generate frame.", status_code=500)
    try:
        response = make_response(generateFrame())
        response.headers['Content-Type'] = 'image/png'
    except Exception as e:
        print(e)
    return response
```

Um die hochsensible Wärmebildkamera innerhalb des Sensormoduls zu schützen und sie nicht freiliegend und anfällig für Staub, Splitter oder allgemein Verdreckung zu betreiben, wird mit einem 3D Drucker eine Hülle generiert und ausgedruckt, in welche

die Kamera eingelassen wird. Dieses Casing wird aus einem biologisch abbaubaren Polyactid Filament (PLA) hergestellt, welches bei direktem Kontakt mit der Hitzequelle bis 300°C Schmelzfest ist und so den hohen Umgebungsluft-Temperaturen eines Einsatzes standhalten kann. Ist die Kamera in das Casing eingebettet und mit einem Kabel an das Sensormodul angeschlossen, kann sie am Helm fixiert werden. Hierzu lässt sich Klettband verwenden, so dass die Kamera zusätzlich zum autonomen Betrieb in Blickrichtung, auch per Hand vom System getrennt und verwendet werden kann.

Exkurs Ursprünglicher Implementationsplan und aufgetretene Probleme: Da es sich bei der Infrarot Kamera um eine Kernfunktionalität des Systems handelt, ist das ursprüngliche Ziel gewesen eine direkte und Kabel basierte Verbindung zwischen Kamera und immersive HMD zu schaffen, so dass keine Abhängigkeit vom getrennten Sensormodul und dessen Akku besteht. Hierzu ist die USB-C Schnittstelle der HoloLens in Verbindung mit einem USB-C auf Micro-USB Kabel zum Einsatz gekommen. Unter Verwendung der Unity Klasse *WebCamTexture* sollte folgend auf die Kamera zugegriffen und das Bild direkt in das Sichtfeld der Einsatzkraft projiziert werden. Da Unity das Kamerabild jedoch aufgrund von Formatinkompatibilitäten nicht auslesen konnte und das Implementieren einer Konvertierung nicht im zeitlichen Rahmen lag, wurde innerhalb des Implementierungsprozesses der Anschluss der Kamera auf das Sensormodul umgestellt und das Bild derselben mittels OpenCV und einem Endpunkt der Schnittstelle des Sensormoduls über WLAN bereitgestellt. Durch diese Umstellung sind jedoch unterschiedliche Probleme und Abhängigkeiten, wie

1. die Abhängigkeit der kritischen Funktionalität vom Akku des Sensormoduls, welcher zusätzlich weiter belastet wird,
2. die Abhängigkeit von der WLAN-Übertragung und Störanfälligkeit des Sensor-Moduls und
3. die Abhängigkeit von weiteren Softwarefehlern,

entstanden. Aufgrund dieser Probleme sollten zukünftige Implementierungen nach alternativen Lösungsmöglichkeiten suchen.

4.4 Zusätzliche Konfigurationen

Dieser Abschnitt behandelt die restlichen nötigen Zusatzprogramme des Sensormodules. Diese sind nötig für einen reibungslosen Ablauf. Dazu gehören Log2Ram, UFW und RaspAP. Diese wurden installiert und mit nur leichten Änderungen in der Konfiguration übernommen. Deswegen wird die Installation und Konfiguration ausgelassen und nur die Pakete kurz beschrieben.

Log2Ram

Linux-Systeme wie auch Raspberry OS protokollieren alle Dienste und Serverprozesse. Der Kernel hat eine Logdatei und jeder weiterer Dienst ebenfalls. Einige Programme schreiben von sich aus Logs und um Standardkomponenten und etablierte Dienste kümmert sich das Syslog-Protokoll. Es handelt sich um eine interne Schnittstelle als Unix-Socket, an die laufende Prozesse, zumeist Serverprozesse, ihre Informationen senden können. Die aufgezeichneten Daten landen zur Langzeitspeicherung in Form von Logdateien im Verzeichnis `/var/log`. Diese werden in kurzen Zyklen geschrieben. Somit geht nichts verloren geht.

Diese Schreibzyklen erweisen sich als Manko auf Ein-Platinen-Rechnern wie dem Raspberry Pi, der seine Systempartition üblicherweise auf einer SD-Karte hat: Schreibaktionen sind bei der vergleichsweise schlechten I/O-Leistung von Flashspeichermedien wie SD-Karten generell langsam. Die permanente Art des Loggings ist zudem der Haltbarkeit des eher empfindlichen Flashspeichers von SD-Karten abträglich. Abhilfe schafft die Log2Ram, welches das Verzeichnis `/var/log` beim Systemstart in eine Ramdisk verlagert und während des Betriebs in größeren zeitlichen Abständen auf den langsamen Datenträger synchronisiert. Es handelt sich also um einen Puffer im RAM, der rund 40 MB Arbeitsspeicher kostet. Diese Investition verkraftet selbst ein Raspberry Pi Zero gut. [53]

UFW

Eine korrekt funktionierende Firewall ist der wichtigste Teil der Sicherheit eines kompletten Linux-Systems. Standardmäßig werden Debian- und Ubuntu-Distributionen mit einem Firewall-Konfigurationswerkzeug namens UFW (Uncomplicated Firewall) ausgeliefert. Dabei handelt es sich um ein sehr beliebtes und einfach zu verwendetes Kommandozeilen-Tool zur Konfiguration und Verwaltung einer Firewall auf Ubuntu- und Debian-Distributionen. [54]

RaspAP

Mit RaspAP kann schnell einen drahtloser Zugangspunkt eingerichtet werden, um die Konnektivität vieler beliebter Debian-basierter Geräte, einschließlich des Raspberry Pi, gemeinsam zu nutzen. Eine reaktionsschnelle Schnittstelle gibt Ihnen die Kontrolle über die relevanten Dienste und Netzwerkoptionen. Erweiterte DHCP-Einstellungen, OpenVPN-Client-Unterstützung, SSL, Sicherheitsprüfungen, Themen und mehrsprachige Optionen sind enthalten. So eignet sich RaspAp super zum Aufbau des für die HoloLens benötigten Hotspots zur Übertragung der Sensordaten. [55]

5 Implementierung Holo-Modul

Aufbauend auf der Sensorik des Sensormoduls soll eine Mixed Reality Applikation entworfen und implementiert werden, welche die in Kapitel 1.2 [Herangehensweise und Methodik] beschriebenen digitalen Anforderungen umsetzt und die Daten des Sensormoduls in einem interaktionsfähigen Interface zusammenfasst. Die Funktionalitäten des **Interface** lassen sich dabei in die zwei **Sub-Module** unterteilen und implementieren die in Abbildung 17 dargestellten Komponenten.

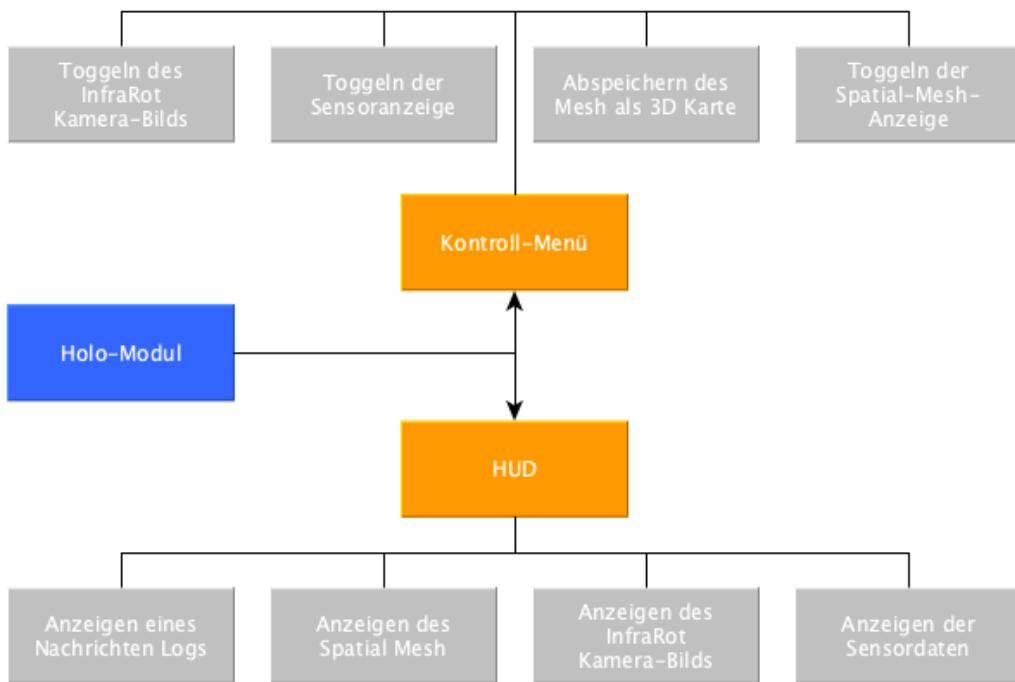


Abbildung 17 Anforderungen an das Holo-Modul

So besteht die Applikation grundsätzlich aus den zwei sichtbaren Bestandteilen **Heads Up Display** und **Kontroll-Menü**. Das **Kontroll-Menü** dient den Einsatzkräften als einfaches und intuitives Kontrollinstrument über die einzelnen Funktionalitäten, welche die Applikation implementiert. Das **Heads Up Display** dient der Darstellung aller Informationen, welche neben dem Kontroll-Menü digital wiedergegeben werden. Hierzu zählen die Sensorwerte, die Log-Benachrichtigungen, das Infrarot-Kamerabild und die Wiedergabe des aktuellen Spatial Mesh Tiefenbilds. Die folgenden Unterkapitel gehen hierzu ausführlich auf die Implementierung der einzelnen Funktionalitäten und Bestandteile ein.

5.1 Aufsetzen der Mixed Reality Applikation

Neben dem Sensormodul bildet die HoloLens 2 in Kombination mit der immersiven Mixed Reality Applikation einen Hauptteil des Prototypen. Sie ist der Kern der Datenwiedergabe und bildet den interaktiven Teil des Prototypen ab, welcher als Schnittstelle zwischen Daten, Realität und Einsatzkraft dient. Alle Informationen laufen im Heads Up Display der Applikation zusammen und unterstützen so die Einsatzkräfte bei der Entscheidungsfindung in den stressigen und gefährlichen Situationen, welchen sie in ihrem Arbeitsalltag ausgesetzt sind. Da es sich bei der Hardware um ein Microsoft Produkt handelt, kann auf ein großes Spektrum an Entwicklungswerkzeugen für Mixed Reality Applikation zurückgegriffen werden. So hat Microsoft in Kooperation mit der Open Source Community ein Produkt entwickelt, um es Entwicklern zu ermöglichen Anwendungen für Microsoft Mixed Reality Produkte zu entwickeln. Dieses dient als grundlegendes Framework für Augmented Reality Implementierungen unter Microsoft Betriebssystem. Dieses *Mixed Reality Toolkit* (MRTK) wurde für den Einsatz in Entwicklungsumgebungen wie Unity und Unreal Engine entwickelt und kann den Entwicklungsprozess einer Augmented Reality Applikation auf Windowsbasis wesentlich beschleunigen [56]. Aufgrund der vereinfachten Integrationsmöglichkeiten für das Mixed Reality Toolkit, der hinreichend ausführlichen Dokumentation und diverser Vorkenntnisse wurde sich im Rahmen der Entwicklung für die Entwicklungsumgebung Unity entschieden. Unity ist mit einem Marktanteil von 45% eine der führenden Plattformen für die Entwicklung und Ausführung von interaktiven Echtzeit-3D-Inhalten und eignet sich als zuverlässiger und sicherer Partner für die Entwicklung der immersiven Applikation [57], [58].

Konfigurieren eines Unity Projekts mit MRTK

Bevor die Oberfläche und die Funktionalitäten mit dem Mixed Reality Toolkit entworfen werden können, muss das zugrundeliegende Unity Projekt initialisiert und konfiguriert werden. Hierzu kann das von Unity veröffentlichte Unity Hub verwendet werden (siehe Abbildung 18).

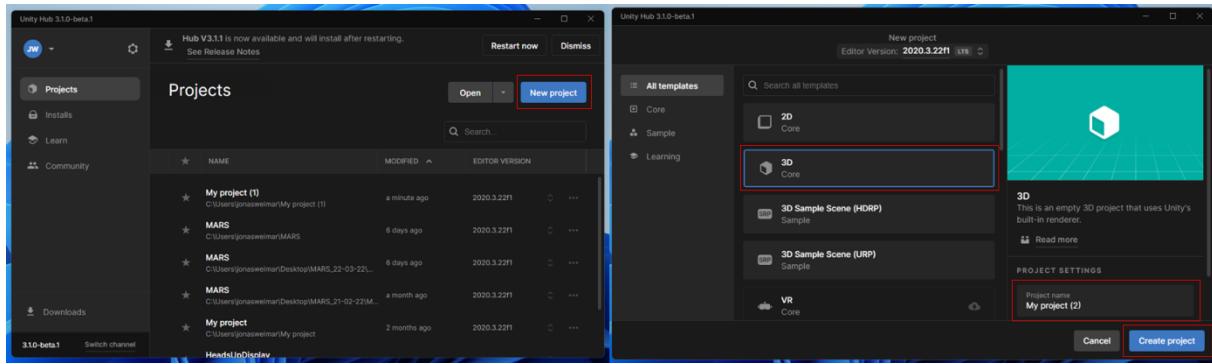


Abbildung 18 Erstellen einer Unity 3D Applikation mit Unity Hub

Hat Unity die Erstellung des Projekts abgeschlossen kann das Mixed Reality Toolkit eingebunden und konfiguriert werden. Hierzu ist der Download des MRTK-Foundation-Pakets als Assetpackage (*.unitypackage*) von GitHub notwendig. Wurde das Paket erfolgreich heruntergeladen muss es über das Unity Menü „Assets à Import Package à Custom Package...“ aus dem Downloadpfad in das Projekt geladen werden. Im folgenden Schritt werden muss das MRTK konfiguriert werden. Hierzu öffnet sich automatisch ein Dialog, welcher einen durch den Konfigurationsprozess leitet (siehe Abbildung 19).

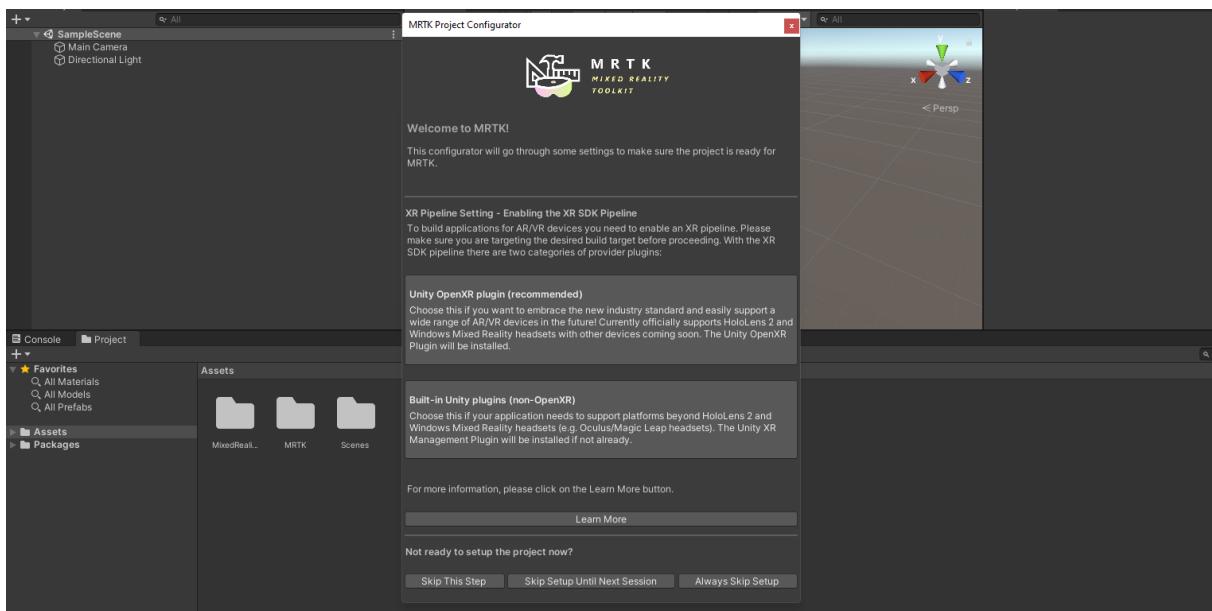


Abbildung 19 Mixed Reality Toolkit Konfigurationsdialog

Um die Konfiguration des Projekts vollständig abzuschließen, wird empfohlen *Text-MeshPro* zu installieren. Dieses Unity-Paket stellt eine Alternative zu den Standard-Textfeldern dar, indem es auf fortschrittlicheren Techniken aufbaut und den Nutzern

flexiblere Anpassungsmöglichkeiten wie benutzerdefinierbare Shader und eine erhöhte Kontrolle bei layouting und styling bietet. So empfiehlt Microsoft TextMeshPro aufgrund der höheren Qualität beim Rendering des Textes in HoloLens Applikationen [59].

Das Mixed Reality Feature Tool

Um auf die benötigten erweiterten Funktionalitäten der HoloLens zugreifen zu können, muss zusätzlich zum Mixed Reality Toolkit noch das Mixed Reality Feature Tool installiert werden. Bei diesem handelt es sich um ein Programm, welches Unity Programmen den Zugriff auf weitere HoloLens-Fähigkeiten, wie das Spatial Understanding Kit gibt. Es lässt sich aus dem Microsoft Download Center herunterladen und kann sofort gestartet werden. Um das Projekt folgend fertig zu konfigurieren, so dass die Applikation Zugriff auf alle für den Mars 1 relevanten Systeme hat, muss das Spatial Understanding Kit über das Mixed Reality Feature Tool installiert werden (siehe Abbildung 20).

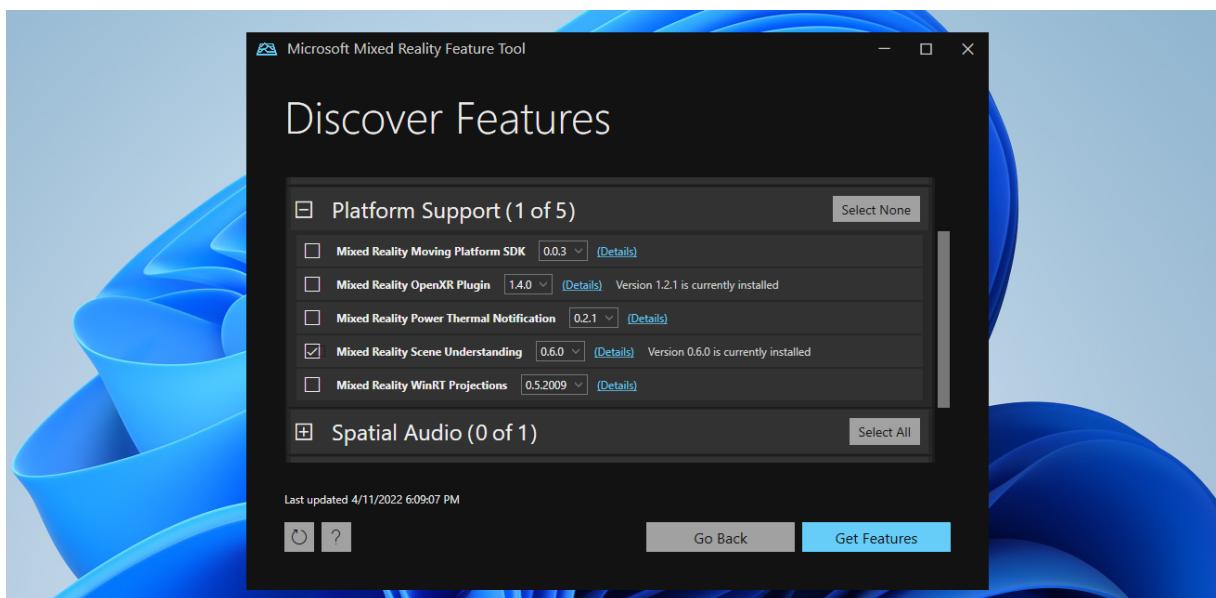


Abbildung 20 Nutzung des Mixed Feature Toolkit

Mit der Fertigen Installation des Spatial Understanding Kits ist das Projekt fertig konfiguriert und eingerichtet. Folgend diesem Prozess kann mit der Entwicklung der eigentlichen Funktionalitäten gestartet werden.

5.2 Implementieren Heads-Up-Display

Den Kern der immersiven Software des Mars 1 bildet das Heads-Up-Display, welches im direkten Sichtfeld der Einsatzkraft liegt und alle von den Subsystemen gesammelten Daten wie Nachrichten und Sensorwerte bereitstellt. Durch den Einsatz eines solchen Systems kann jede Einsatzkraft, ausgestattet mit einem Mars 1, parallel auf wichtige Daten zugreifen, ohne sich von der Gefahrenstelle abwenden zu müssen. Hierbei ist das HUD so konzipiert, dass es das Sichtfeld der Trägerperson nicht vollständig blockiert und durch seine Lichtdurchlässigkeit so weiterhin ein vollumfängliches Sichtfeld garantiert. Um dieses HUD zu implementieren, wird in Unity ein leeres *Canvas* Objekt unter dem Namen *HeadsUpDisplay* hinzugefügt, welches als Grundlage und *Parent* für die restliche Struktur des HUD dient (siehe Abbildung 21).

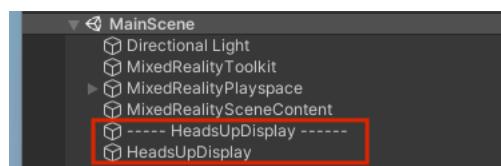


Abbildung 21 HUD Canvas

Diesem *Parent* werden in den folgenden Schritten iterativ die Funktionalitäten *Immersives Wärmebild*, *Sensordatenanzeige*, *Tiefenbildanzeige* und *Nachrichten Log* bestehend aus UI und Skript hinzugefügt, so dass das aus dem Requirements Engineering hervorgegangene User Interface entsteht, welches den Einsatzkräften als Interaktionsebene dienen soll (siehe Abbildung 22).

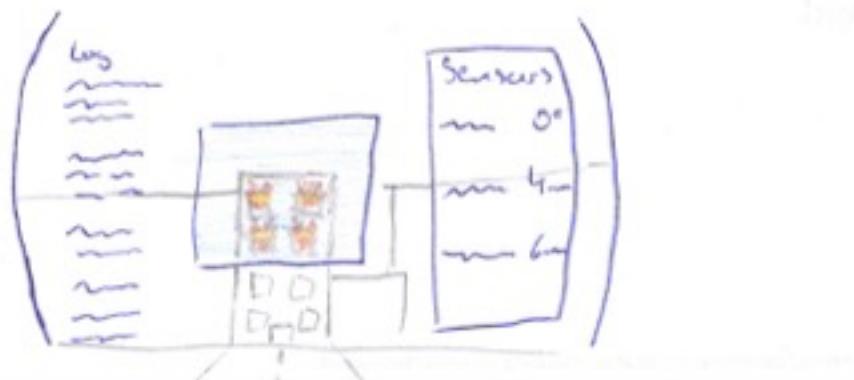


Abbildung 22 Geplantes User Interface

Zentrale Klasse MonoBehaviour

Die meisten Funktionalitäten werden implementiert durch ein Nutzerinterface und ein Skript. Diese Skripts sind zuständig für das Management der Oberfläche und so die Veränderung der Daten, welche den Einsatzkräften zur Verfügung gestellt werden. Aus Gründen der Gewohnheit, werden alle Skripts in dieser Applikation mit C# implementiert. Alle Skript-Klassen einer Unity-Applikation erben dabei von der MonoBehaviour Klasse. Durch diese Vererbung stehen den eingeführten Skripts diverse Standardmethoden bereit, welche für das Managen des Skripts und die Integration desselben in die Applikation unabdingbar sind. Hierzu zählen die Methoden:

```
Start()  
Update()  
FixedUpdate()  
LateUpdate()  
OnGUI()  
OnDisable()  
OnEnable()
```

Relevant für die Entwicklung sind dabei die Methoden *Start* und *Update* welche folgend kurz erläutert werden.

Die Methode *Start* wird mit Beginn des ersten Frames nach Aktivierung des Skripts aufgerufen. Hierbei wird die Methode ausschließlich einmal im Laufe des Lebenszyklus des Skripts aufgerufen. Da sie mittels des Rückgabetyps *IEnumerator* auch als Coroutine definiert werden kann, ist es möglich die Ausführung der Methode zu suspendieren bspw. durch einen *yield return new WaitForSeconds(2.5f)*. Die Methode *Update* wird mit jedem Frame aufgerufen, sollte die Skriptklasse nicht disabled sein. Über sie kann eine Interaktion zwischen Spieler / Nutzer und Applikation auf Frame-Basis geschaffen werden.

Immersives Wärmebild

Die erste Funktionalität, welche dem Canvas hinzugefügt wird, ist die Anzeige der Wärmebildkameradaten. Für die Umsetzung dieser Funktionalität wird die Mixed Reality Applikation im Canvas über eine Instanz der Unity-UI Klasse *RawImage* und im

Code um das Skript *HeatVisionSwitch.cs* für die spätere De- und Aktivierung des Hitzeblicks erweitert. Das *RawImage* dient dabei als Container für die Bilddaten, welche das Skript von der Schnittstelle des Sensormoduls im Millisekunden Takt abfragt und updatet. Einen Überblick über die wichtigsten Parameter des *RawImage Containers* finden sich in Tabelle 8 Überblick *RawImage*.

Name	(Objekt) Funktion
texture	(Texture) Die Textur des <i>RawImage</i> ist die Eigenschaft, welche das Aussehen bzw. die Textur bestimmt.
color	(Color) Die Farbe des <i>RawImage</i> . Über sie kann auch der Alpha-Wert des <i>RawImage</i> und so die Transparenz des Containers manipuliert werden.

Tabelle 8 Überblick *RawImage*

Da die Größe des Bildes der Wärmebildkamera 160x120 Pixel beträgt dies jedoch ein zu klein für eine ordentliche Darstellung ist, wird das Bild um den Faktor 4 hochskaliert, indem die Größe des *RawImage* Containers auf 640x480 gesetzt wird. Hierbei ist die Methode, welche Unity für das Interpolieren der Pixel beim Skalieren verwendet nicht bekannt. Folgend wird das *RawImage* im Zentrum des Canvas platziert, so dass es bei Aktivierung im direkten Sichtfeld der Einsatzkraft liegt. Um die Funktionalität per Skript ein- und ausschalten zu können, wird eine boolesche Variable *isThermalVisionActive* angelegt, mit Hilfe welcher über die Sichtbarkeit des *RawImage* Containers bestimmt wird, indem sie über den Alpha-Wert des *RawImage.color* Objekts und so die Transparenz des Containers entscheidet.

```
public RawImage thermalCameraView;  
private bool isThermalVisionActive = false;
```

----- Ändern des Sichtbarkeitsstatus -----

```
isThermalVisionActive = !isThermalVisionActive;  
  
thermalCameraView.color = isThermalVisionActive  
    ? new Color(thermalCameraView.color.r, thermalCameraView.color.g,  
        thermalCameraView.color.b, 1)  
    : new Color(thermalCameraView.color.r, thermalCameraView.color.g,  
        thermalCameraView.color.b, 0);
```

Um ein Skript in die Applikation aufzunehmen und zu instanziieren, wird in der Hierarchie des Projekts ein leeres Element erstellt, welchem das Skript zugewiesen wird. So wird ein solcher Container mit dem Namen *HeatVisionController* in die Hierarchie integriert und ihm wird das bereits erstellte Skript hinzugefügt. Alle als *public* markierten Eigenschaften der Klasse können über diesen Controller folgend in Unity konfiguriert werden, indem die zugesetzten Elemente in das Feld neben dem Namen der Eigenschaft gezogen werden.

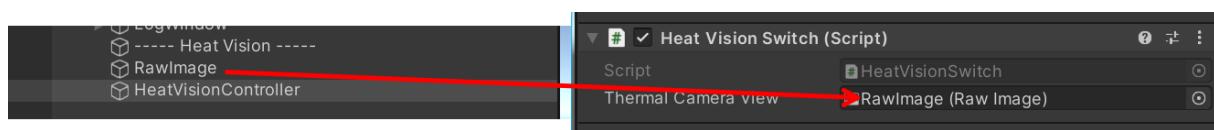


Abbildung 23 Konfigurieren von *public* Parametern mit Referenzen auf UI Objekte

Um auf die Daten der Wärmebildkamera zugreifen zu können, wird folgend eine asynchrone Sub-Routine implementiert, welche mittels der Methode *UnityWebRequestTexture.GetTexture(string uri)* eine Http-Anfrage an das Sensormodul übermittelt und die zurückgesendeten Daten automatisch in die Unity-Klasse *Texture* Typkonvertiert. Die so geladene Textur kann im folgenden Schritt in das bereits angelegte Unity *RawImage* geladen werden. Treten Fehler bei der Datenübertragung auf, werden diese über den internen Log an die Einsatzkraft kommuniziert und so der Status der Kamera übermittelt.

```

IEnumerator GetFrame()
{
    UnityWebRequest request =
        UnityWebRequestTexture.GetTexture(sensorModuleURI);
    yield return request.SendWebRequest();

    if (request.isNetworkError || request.isHttpError)
    {
        Debug.LogWarning(request.error);
        yield break;
    }

    thermalCameraView.texture = ((DownloadHandlerTexture)
        request.downloadHandler).texture
}

```

Da die Schnittstelle aufgrund von Problemen mit Unity und der HoloLens keinen Stream bereitstellt, welchen die immersive Software aufgreifen könnte, müssen die Bilder der Wärmebildkamera einzeln abgefragt und angezeigt werden. Hierzu wird eine Subroutine implementiert, welche über eine while-Schleife, unter der Kondition, dass die Funktionalität aktiviert ist, alle 40ms die Subroutine *GetFrame* startet. Die Methode wird folgend bei Aktivieren der Funktionalität aufgerufen und stellt so eine im Millisekunden Bereich liegende Abfragefrequenz bereit, welche verwendet wird, um Bilder der WBK so zu laden, dass die Illusion eines Videostreams entsteht.

```

IEnumerator UpdateFrame()
{
    while (isThermalVisionActive)
    {
        yield return new WaitForSeconds(0.04f);
        StartCoroutine(GetFrame());
    }
}

```

Wurde diese Routine implementiert, ist das System vollständig einsatzbereit und die Einsatzkraft kann in vollem Umfang auf die Funktionalität Hitzeblick zugreifen. Die so erzeugte Wärmebildreihe wird dann inmitten des HUD angezeigt und bietet einen akkurate Überblick über die Wärmesignaturen der Gegenstände im Sichtfeld der nutzenden Einsatzkraft (siehe Abbildung 24).

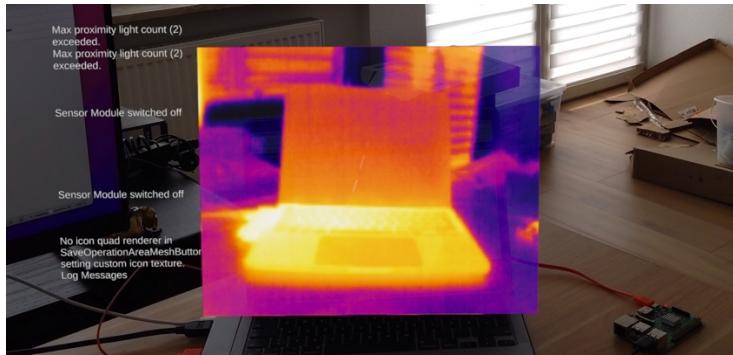


Abbildung 24 Das immersive Wärmebild integriert ins HUD

Sensordatenanzeige

Um den Einsatzkräften die Daten des Sensormoduls bereitzustellen, wird eine Anzeige auf Augenhöhe und rechter Seite des Heads Up Displays implementiert (siehe Abbildung 22). Die Anzeige wird in vier Sektionen unterteilt, welche die mit der Feuerwehr abgeklärten Sensorwerte des Sauerstoffsensors, des Kohlenstoffmonoxidsensors und des Temperatursensors und den Batteriestand der HoloLens 2 abbilden. Um in Umgebungen mit starker Sicht einschränkung oder hektischem Umfeld eine Erweiterung des Sichtfelds zu ermöglichen, wird die Datenanzeige so implementiert, dass die tragende Einsatzkraft dieselbe über das Steuerungsmenü ausblenden kann. Für das Erstellen des User Interface wird auf das Mixed Reality Toolkit zurückgegriffen, da es sich bei der Sensordatenanzeige um ein statisch platziertes Hologramm handelt und das MRTK Bausteine für diese anbietet. So wird im rechten Bereich des Heads Up Displays eine standard MRTK *Backplate* mit blauem Hintergrund in angemessener Größe (ergibt sich durch aktives Testen on-Device) hinterlegt. Diese dient als Container für den weiteren Aufbau des Anzeigesystems. Folgend werden vier *TextMeshPro* Label-Value Paare für die Anzeige der drei Sensorwerte sowie des Batteriestands erstellt und in einem Vertikal angeordneten Stil an die *Backplate* angeheftet. So entsteht das Grundgerüst der Sensordatenanzeige, welches im nächsten Schritt um 8° auf der Y-Achse gedreht wird, um einen angenehmeren Sichtwinkel zu ermöglichen. Folgend werden die Labels der Paare mit dem Namen des Abzubildenden Sensorwerts beschriftet, während die Value Felder der Paare mit Initialwerten beschriftet werden. Diese werden im Betrieb der Applikation, durch die mit Hilfe des angelegten Skripts geladenen Sensorwerte, ersetzt.



Abbildung 25 Beschrifteter Rohling des Sensordatencontainers

Nachdem das rohe Interface der Sensordatenanzeige erstellt worden ist, wird der Controller desselben implementiert. Dieser hat die Aufgabe die Sensordaten vom Sensormodul auszulesen und in die Sensordatenanzeige zu laden. Um den Text der für die Sensordaten bestimmten *TextMeshPro* Felder manipulieren zu können und so die geladenen Daten auf dem User Interface wiedergeben zu können, müssen vier Variablen des Typs *TextMeshProUGUI* im Skript angelegt werden, welche folgend über die Unity Hierarchie mit denen des User Interface nach dem Schema aus Abbildung 23 verknüpft werden.

```
public TextMeshProUGUI temperatureSensorValue;
public TextMeshProUGUI carbonMonoxideSensorValue;
public TextMeshProUGUI oxigenSensorValue;
public TextMeshProUGUI batteryStatus;
```

Um die Daten des Sensormoduls auszulesen, wird die Methode *GetSensorModuleResponse(string uri)* implementiert. Sie sendet unter Verwendung der Klasse *UnityWebRequest* asynchron ein standardisiertes GET-Request an das Sensormodul. Wie bereits innerhalb der Funktionalität *Wärmebild*, wird dieses Request auf Fehler überprüft. Treten solche auf, wird eine Nachricht an die Einsatzkraft weitergegeben. Ist die Antwort des Sensormoduls fehlerfrei, werden die Daten aus dem JSON Format in die Klasse eigens entwickelte Klasse *SensorModuleResponse* serialisiert. Diese dient als primitiver Wrapper der Daten und wird hier nicht weiter behandelt. Ihre Funktionen werden, wenn nicht selbsterklärend, an der relevanten Stelle kurz aufgeführt.

```
UnityWebRequest webRequest = UnityWebRequest.Get(uri);
```

----- Serialisieren der Antwort -----

```
SensorModuleResponse sensorModuleResponse =
    JsonConvert.DeserializeObject<SensorModuleResponse>(
        webRequest.downloadHandler.text
    );
```

Sind die Sensormoduldaten aufbereitet und geladen, können diese über die Schnittstelle *TextMeshProUGUI.text* auf der jeweiligen Label-Variable auf dem User-Interface dargestellt werden. Hierzu wird auf die erzeugte Instanz der *SensorModuleResponse* zurückgegriffen, welche die Werte der JSON-Response speichert. Der aktuelle Ladezustand der HoloLens kann über die System interne Klasse *SystemInfo* abgefragt werden; Der Ladezustand wird hierbei im Bereich 0 bis 1 angegeben. Um diese über String-Konkatenation mit den richtigen Maßeinheiten zu verheiraten, müssen die rohen Werte vorher über die *ToString()*-Methode in Strings konvertiert werden.

```
temperatureSensorValue.text =
    sensorModuleResponse.temperature.value.ToString() + "°C";
carbonMonoxideSensorValue.text =
    sensorModuleResponse.carbon.value.ToString() + "mg/m^3";
oxigenSensorValue.text =
    sensorModuleResponse.oxygen.value.ToString() + "%";
```

----- Laden des aktuellen Ladezustands -----

```
batteryStatus.text = (SystemInfo.batteryLevel * 100).ToString() + "%";
```

Um die, durch das Wärmebild bereits stark belastete, Verbindung nicht weiter zu strapazieren, werden die Daten des Sensormoduls nicht kontinuierlich, sondern in einem Intervall von zwei Sekunden, aktualisiert. Hierzu wird von der Methode *Start()*, vererbt an das Skript durch die Klasse *MonoBehaviour*, Gebrauch gemacht. Diese wird bei Initialisierung eines Skripts aufgerufen und führt Befehle noch vor der ersten Frame Aktualisierung aus; Da das Skript unkonditioniert in die Applikation geladen wird, startet die Methode defacto mit Start der Applikation. Um während der gesamten Laufzeit der Applikation die Daten des Sensormoduls auszulesen und den Nutzern bereitzu-

stellen, wird die Start Methode als Subroutine definiert, welche folgend mit einer unkonditioniert laufenden while-Schleife die Daten aus dem Sensormodul abfragt, indem sie im zwei Sekunden Takt die Subroutine *GetSensorModuleResponse* startet.

```
IEnumerator Start()
{
    while (true)
    {
        yield return new WaitForSeconds(2f);
        StartCoroutine(GetSensorModuleResponse(sensorModuleURI));
    }
}
```

Mit der Implementierung der Routine zum Update der Label, ist die Funktionalität vollständig implementiert. Folgend gilt es, die Sensordatenanzeige über das Steuermenü aus dem Sichtfeld zu entfernen. Auf diese Funktionalität wird konkreter in Kapitel 5.4 Steuerung des MARS 1 eingegangen.

Tiefenbildanzeige

In schlechten Verhältnissen wie beispielsweise dichtem Rauch lassen sich teilweise kaum noch Silhouetten der räumlichen Details wahrnehmen, was eine Desorientierung der Einsatzkräfte zur Folge haben kann. Um diesem Problem entgegenzuwirken, wird, wie beschrieben in Kapitel 1.2 Herangehensweise und Methodik, auf die Fähigkeit der Spatial Mesh Visualization der HoloLens 2 zurückgegriffen und die Kontrolle derselben in die Mixed Reality Applikation integriert. Auch für diese Funktionalität wird ein Skript angelegt, welches aufgrund seiner Parameterfreiheit nicht über die Hierarchie des Unity Projekts konfiguriert werden muss. Das Skript *SpatialAwarenessSwitch.cs* hat hierbei die Aufgabe per Funktionsaufruf die vom Spatial Awareness System der HoloLens aufgezeichneten Daten zu visualisieren und das Spatial Mesh so im Sichtfeld der Einsatzkraft anzuzeigen (siehe Tabelle 9). So wird der Einsatzkraft ein Tiefenbild der Umgebung bereitgestellt, auf welches zu Orientierungszwecken zurückgegriffen werden kann.

Name	Funktion
void ToggleSpatialAwareness()	Schaltet die Sichtbarkeit des Spatial Mesh über alle hinterlegten Mesh Observer Klassen ein / aus.

Tabelle 9 *SpatialAwarenessSwitch* Funktionen

Das Mixed Reality Toolkit stellt über die Klasse *MeshObserver* eine Schnittstelle zur Verfügung, über welches es möglich ist auf die genannten Spatial Mesh Daten zuzugreifen. Auf diese Observer kann über das Kernsystem *SpatialAwarenessSystem* zugegriffen werden, indem dasselbe auf das Interface *IMixedRealityDataProviderAccess* typkonvertiert wird [60]. Dieser Schritt kann mit der Abfrage, ob überhaupt ein Zugriff auf die Data Provider hinterlegt ist, kombiniert werden.

```
if (CoreServices.SpatialAwarenessSystem is IMixedRealityDataProviderAccess provider)
```

Da mehr als ein solches System hinterlegt werden kann, muss folgend über alle im System registrierten *DataProvider* Klassen iteriert werden. Bei jeder Iteration wird folgend geprüft, ob es sich bei dem selektierten Provider um einen Mesh Observer handelt. Handelt es sich bei der Selektion um ein solches System, kann es mittels Typkonvertierung in eine Instanz des *IMixedRealitySpatialAwarenessMeshObserver* überführt werden. Über die in diesem Interface bereitgestellten Funktionen kann die Sichtbarkeit der Spatial Mesh Daten im Blickfeld der Einsatzkraft manipuliert werden, indem auf die Eigenschaft *DisplayOption* des MeshObserver Interface zurückgegriffen wird. Sie kann über den Enum *SpatialAwarenessMeshDisplayOptions* in einen von drei Modi versetzt werden. Die für dieses Projekt relevanten Modi stellen hierbei *None* und *Visible*. Wird die *DisplayOption* des MeshObservers auf *None* gestellt, wird dem Nutzer kein Mesh angezeigt, auf *Visible* wird dasselbe sichtbar.

```

foreach (var observer in provider.GetDataProviders())
{
    if (observer is IMixedRealitySpatialAwarenessMeshObserver meshObserver)
    {
        if (meshObserver.DisplayOption
            .Equals(SpatialAwarenessMeshDisplayOptions.None))
        {
            meshObserver.DisplayOption =
                SpatialAwarenessMeshDisplayOptions.Visible;
        } else
        {
            meshObserver.DisplayOption =
                SpatialAwarenessMeshDisplayOptions.None;
        }
    }
}

```

Um einen Spatial Mesh Observer im System zu registrieren, muss das *Spatial Awareness* System in der Konfiguration des Mixed Reality Toolkits aktiviert werden. Folgend kann ein Betriebssystem abhängiger Observer über die Schaltfläche „+ Add *Spatial Observer*“ hinzugefügt werden (siehe Abbildung 26).

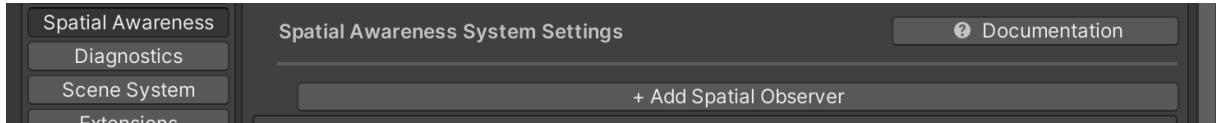


Abbildung 26 Hinzufügen eines Spatial Mesh Observers

Nach dem Implementieren der Funktionalität kann das Spatial Mesh im Sichtfeld beobachtet werden. Wichtig ist, dass sich das Mesh kontinuierlich erweitert und initial so auch „Blind Spots“, Stellen an welchen noch kein Spatial Mesh aufgezeichnet wurde, sichtbar sind. Trotz dieser Tücken, dient das Spatial Awareness System der Einsatzkraft als zuverlässiger Indikator über Tiefe und Form der Umgebung (siehe Abbildung 27).

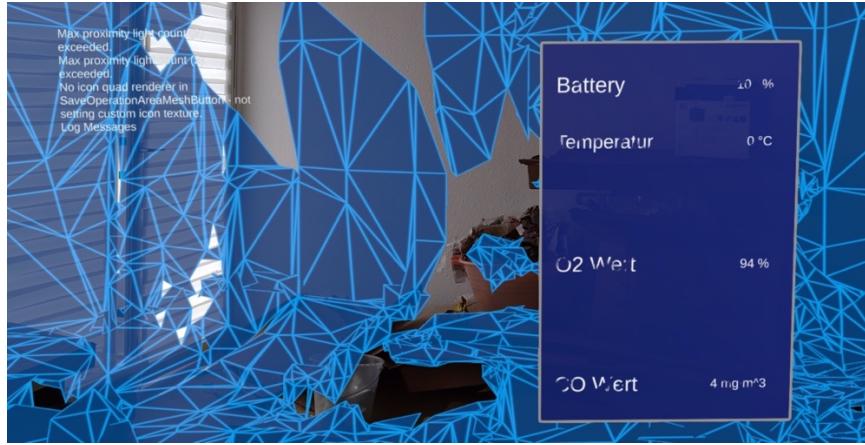


Abbildung 27 Darstellung der aktivierte Tiefenanzeige

Nachrichten Log

In den Systemen der Mixed Reality Applikation kann es vermehrt zu Fehlern kommen beispielsweise, wenn das Bild der Wärmebildkamera ausfällt. Um die Einsatzkraft über diese Fehler und weitere Systeminformationen zu benachrichtigen, wird ein System im Sichtfeld der Einsatzkraft benötigt, welches in der Lage ist Textnachrichten wiederzugeben. Hierzu wird dem Canvas des Heads-Up-Displays in der Hierarchie ein Text-MeshPro Feld hinzugefügt. Der Name dieses Felds wird zu Identifizierungszwecken umbenannt in *LogWindow*. Um auf diesem Textfeld folgend Log-Benachrichtigungen abilden zu können, wird ein Skript mit dem Namen *LogWindow.cs* angelegt und als MonoBehaviour in die Hierarchie des Unity Projekts integriert. Innerhalb des Skripts wird folgend eine öffentliche Variable des Typs *TextMeshProUGUI* angelegt.

```
public TextMeshProUGUI textMesh;
```

Diese wird mit dem *TextMeshPro* Objekt des User Interface auf gleiche Weise verknüpft, wie die öffentliche Variable dargestellt in Abbildung 23. Um auf Nachrichten des internen Log-Systems zugreifen zu können, wird eine öffentliche Methode *LogMessage (void)* implementiert. Diese folgt dem Schema der internen Funktionen des *Application.LogCallback* mit den Parametern *message (string)*, *stackTrace (string)* und *type (LogType)*.

```
public void LogMessage(string message, string stackTrace, LogType type)
```

Der Parameter *message* wird folgend verwendet, um die Skript interne Variable *text-Mesh* zu aktualisieren. Hierzu wird die Nachricht an die anderen Nachrichten vorne angehängt.

```
textMesh.text = message + "\n" + textMesh.text;
```

Damit der Text aus den von Unity gesteuerten LogNachrichten folgend automatisch an die erstellte Callback-Funktion übergeben werden kann, wird das erstellte Callback an den internen Verteiler *Application.logMessageReceived* angehängt und innerhalb der durch *MonoBehaviour* vererbten Funktion *OnEnable (void)* aufgerufen.

```
void OnEnable()
{
    Application.logMessageReceived += LogMessage;
}
```

Unter dieser Konfiguration werden alle geworfenen Log Nachrichten auf dem Nachrichten Display des HUD angezeigt. Um nur bestimmte Nachrichten, spezifisch auf die Applikation zugeschnitten anzeigen zu lassen, muss der interner Enum *LogType* erweitert werden, so dass nach dieser Erweiterung im Callback mittels *if*-Clause gefiltert werden kann.

5.3 Kartographierung des Einsatzgebiets

Für eine sinnvolle Aufbereitung der Einsätze und das Treffen realistischer Einschätzungen über Situation und Lage ist das Sammeln von Daten von großer Relevanz. Ziel ist es hier, nach Abschluss eines Einsatzes das Einsatzgebiet weiter analysieren zu können, um bspw. potenzielle Brandursachen zu lokalisieren. Hierbei wird das Spatial Awareness System der Hololens 2 eingesetzt, um das Einsatzgebiet auf Knopfdruck dreidimensional zu kartographieren. Das bereits vorgestellte Spatial Awareness System der Hololens 2 ist entwickelt worden, um es Hologrammen möglich zu machen mit der Umgebung zu interagieren und so beispielsweise hinter Objekten zu verschwinden, als würden sie durch diese verdeckt. So wird den Nutzern des Headsets realer Eindruck der Hologramme vermittelt. Um diese Eindrücke zu ermöglichen, erstellt das Spatial Awareness System mit Hilfe des ToF Sensors und den diversen Kameras eine

Tiefenkarte der Umgebung sowie ein 3D Koordinatensystem zur Lokalisierung der Hologramme. Es entsteht somit ein auf Mesh basierendes dreidimensionales Modell der Umgebung (Abbildung 28).

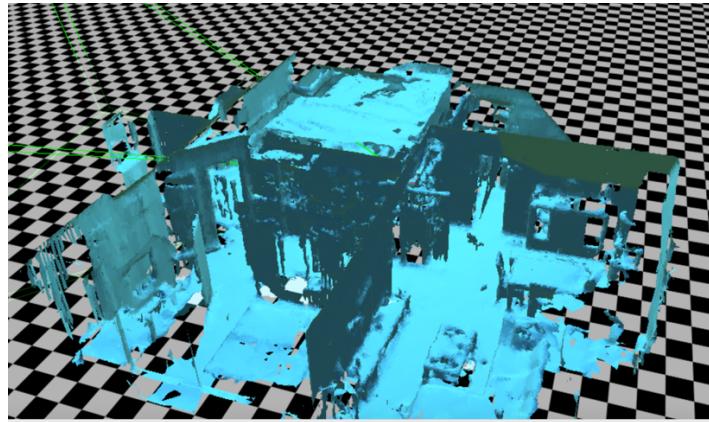


Abbildung 28 Selbsterstellte dreidimensionale Kartographie

Bevor die Karte abgespeichert werden kann, muss mittels des Mixed Reality Feature Tools das Paket *Mixed Reality Scene Understanding* (SUKit) installiert werden. Dieses baut auf dem grundlegenden Spatial Awareness System auf und stellt weitere Funktionalitäten, wie das Berechnen einer vollständigen Scene sowie das Klassifizieren der in ihr enthaltenen Objekte, bereit [61], [27]. Um folgend die von der HoloLens generierte Karte abzuspeichern, wird ein weiteres Skript implementiert. Das neu installierte Tool wird folgend importiert, um die Funktionen des SUKit nutzen zu können. Es werden mehrere Funktionen für das kontinuierliche Berechnen der Karte, das Kopieren der letzten Berechnung und das Speichern der Daten auf dem Gerät implementiert (siehe Tabelle 10 Funktionen des Kartographiesystems)

Name	Funktion
void ComputeSceneUnderstanding-Data()	Liest die von der HoloLens konstruierte Karte als Byte Array aus dem System aus und speichert diese zur weiteren Verarbeitung in einer globalen Variable.
byte[] GetLatestSceneBytes()	Erstellt eine Copy der aktuellen Scene Daten und gibt diese zurück.

<pre>void SaveBytesToDiskAsync()</pre>	<p>Speichert die gesammelten Scene Daten als .byte Datei auf dem Gerät, indem auf die Funktion <i>GetLatestSceneBytes()</i> zurückgegriffen wird.</p>
--	---

Tabelle 10 Funktionen des Kartographiesystems

Um kontinuierlich die generierten Spatial Mesh Daten der HoloLens zu berechnen und in der angelegten Skriptklasse abzuspeichern, wird die Methode *ComputeSceneUnderstandingData* implementiert. Sie lädt die vom Spatial Mesh System generierten Daten mittels der Funktion *SceneObserver.ComputeSerializedAsync(QuerySettings settings, float radius)* des SUKits in die private Eigenschaft *SceneUnderstandingData* des Skripts. Hierbei wird die Abfrage in den Query Settings so konfiguriert, dass das Mesh in unlimitiertem Detail aufgezeichnet werden kann; also in der höchsten Stufe.

```
private byte[] SceneUnderstandingData = null;

----- Laden der Spatial Mesh Daten -----

SceneQuerySettings querySettings;

querySettings.EnableSceneObjectQuads = true;
querySettings.EnableSceneObjectMeshes = true;
querySettings.EnableOnlyObservedSceneObjects = false;
querySettings.EnableWorldMesh = true;
querySettings.RequestedMeshLevelOfDetail = SceneMeshLevelOfDetail.Unlimited;

SceneBuffer newSceneBuffer = SceneObserver.ComputeSerializedAsync(
    querySettings,
    10.0f
).GetAwaiter().GetResult();

SceneUnderstandingData = new byte[newSceneBuffer.Size];
newSceneBuffer.GetData(SceneUnderstandingData);
```

Um die Daten folgend kontinuierlich ab Start der Applikation in der Klasse zu hinterlegen, wird die der Skript Klasse vererbte Methode *Start* als Subroutine implementiert. Diese fragt die Zugriffsberechtigung auf das Scene Understanding Kit und die aktuelle Runtime ab, um potenziellen Fehlern durch nicht vorhandene Zugriffsrechte und der nicht möglichen Verwendung eines UWP exklusive Speicherprogramms vorzubeugen.

Folgend startet die Subroutine eine an die Lebensdauer der Klasseninstanz gebundene while-Schleife, welche im 1.25 Sekunden Takt die bereits beschriebene Kalkulationsmethode *ComputeSceneUnderstandingData* aufruft.

```
if (Application.isEditor)  
if (!SceneObserver.IsSupported())  
  
SceneObserverAccessStatus access = SceneObserver  
    .RequestAccessAsync().GetAwaiter().GetResult();  
if (access != SceneObserverAccessStatus.Allowed)  
  
----- Kopf der Überprüfungen sowie kontinuierliche Aufrufe der Berechnung -----  
  
while (this.isActiveAndEnabled)  
{  
    yield return new WaitForSeconds(1.25f);  
    this.ComputeSceneUnderstandingData();  
}
```

Um die so berechneten Daten im File System der HoloLens 2 abzuspeichern und so über den Lebenszyklus der Applikation hinweg zu persistieren, werden die Methoden *SaveBytesToDiskAsync* und *GetLatestSceneBytes* implementiert. Während die letztere ausschließlich dem Anfertigen einer Kopie der gesammelten dient, speichert *SaveBytesToDiskAsync* diese Daten auf der HoloLens. Hierzu lädt die Funktion die kopierten Daten und speichert diese über den UWP exklusiven Service *WindowsStorage* unter Verwendung eines einzigartigen Dateinamens im byte Format ab; der Dateiname folgt dem Schema *SU_{year}-{month}-{day}_{hour}-{minutes}-{seconds}.bytes*. Bei einer Namenskollision wird dieser durch den WindowsStorage automatisch in ein einzigartiges Konstrukt geändert.

```

if (QuerySceneFromDevice)
{
    string fileName = string.Format("SU_{0}-{1}-{2}_{3}-{4}-{5}.bytes",
        year, month, day, hour, min, sec);

    byte[] OnDeviceBytes = this.GetLatestSceneBytes();

#ifndef WINDOWS_UWP
    var folder = WindowsStorage.ApplicationData.Current.LocalFolder;
    var file = await folder.CreateFileAsync(fileName, WindowsStorage
        .CreationCollisionOption
        .GenerateUniqueName);

    await WindowsStorage.FileIO.WriteBytesAsync(file, OnDeviceBytes);
#else
    Debug.LogWarning("Only available in Universal Windows Applications (UWP).");
#endif
}

```

Folgend wird eine Schnittstelle nach außen benötigt, welche den Speicherablauf Prozess aufruft und ausführt. Hierzu wird die öffentlich zugängliche Methode *SaveOperationAreaMesh* implementiert, welche ausschließlich für das sichere Ausführen der Methode *SaveBytesToDiskAsync* verantwortlich ist. Sie fängt Fehler ab und leitet diese an die Einsatzkraft weiter. Aufgrund unbekannter Probleme ist das System jedoch nicht vollständig funktionsfähig. Wird die Funktion betätigt, stürzt der Prototyp ab, ohne Fehlernachrichten im Debug Log der Visual Studio Entwicklungsumgebung widerzugeben. Selbst nach diversen Änderungen und einem strikten Folgen der Dokumentation Microsofts, konnte keine Verbesserung erzielt werden. Somit bleibt diese Funktionalität implementiert, jedoch nicht einsatzbereit.

5.4 Steuerung des MARS 1

Folgend der Implementation der einzelnen Eigenschaften des Funktionsabschnitts 1 (siehe Tabelle 1 Liste der Funktionalitäten), wird mit Funktionalität 2 ein digitales Menü eingeführt. Dabei dient das Steuermenü als digitale Interaktionsschnittstelle zwischen Mensch und System, welches den Einsatzkräften die interaktive Steuerung der einzelnen Funktionalitäten wie folgt ermöglicht.

1. Anzeigen von Datum und Uhrzeit
2. Ein- / Ausblenden der Sensordatenanzeige
3. Ein- / Aus(blenden | schalten) der Wärmebildanzeige
4. (De-) Aktivieren der Tiefenbildanzeige im Sichtfeld
5. Abspeichern des aufgezeichneten Spatial Mesh

Um diese Interaktionsschnittstelle einzuführen, wird ein digitales Steuerfenster entworfen und implementiert. Das Mixed Reality Toolkit bietet eine große Auswahl an einfach zu konfigurierenden Menüs an, welche fast nahtlos in individuelle Projekte integriert werden können. Hierbei setzt das MRTK auf das Hand-Tracking-System der HoloLens 2 um Interaktionen zwischen den Menüs und Nutzern zu erkennen. Microsoft empfiehlt eine Ulna seitige Positionierung des Menüs, da es so während der Interaktion zu keiner Überlappung der Hände kommt und das Hand-Tracking System optimal und ohne Interferenzen arbeiten kann, was deutlich höhere Erfolgsquoten bei der Interaktionsinterpretation zu Folge hat [62]. Für die Implementierung wird ein Standardmenü über den Projektordner aus dem MRTK SDK in die Hierarchie des Projekts überführt. Folgend werden alle vorkonfigurierten Bausteine aus dem Menü entfernt, so dass ein Rohling übrigbleibt, welcher an die Anforderungen des Systems angepasst werden kann. Um eine Ulna Seitige Positionierung des Menüs zu gewährleisten, wird in den Konfigurationen des Menüs unter *Hand Constraint à Safe Zone* der Unterpunkt *Ulnar Side* ausgewählt (siehe Abbildung 29).



Abbildung 29 Auswahl der Ulna Seite in der Hand Menü Konfiguration

Ist die Anzeige des Menüs konfiguriert, können die Knöpfe und Labels für die zuvor beschriebenen Funktionen installiert werden. Standardversionen der MRTK Buttons finden sich wie auch das Hand Menü in einem Ordner des MRTK SDK. Für die Label

werden die Standard Unity UI Elemente verwendet. Folgend der Imports aller Elemente in die Hierarchie des Projekts, können Labels und Icons der Buttons an den Einsatz innerhalb des Systems angepasst werden, so dass ein fertiger UI-Rohling des Menüs entsteht (siehe Abbildung 30).



Abbildung 30 UI-Rohling des Hand Menüs

Das Menü stellt grundsätzlich ein User Interface für die in den Skripten implementierten Aktivierungs- bzw. Deaktivierungsmethoden der einzelnen Funktionalitäten dar. Ist das Menü UI seitig vollständig implementiert, werden den Buttons und den dynamischen Labels (Uhrzeit und Datum) die korrespondierenden Funktionen und Skripte zugewiesen. Für die Buttons werden die in der Hierarchie untergebrachten Instanzen der Skripte in der Button Konfiguration *Button Pressed ()* gezogen. Folgend kann ausgewählt werden, welche Funktion der Skript-Instanz bei Auslösen des Knopfdruck-Events ausgeführt werden soll. Hier werden die bereits vorgestellten, präparierten *ToggleXXX* Methoden der Skripte ausgewählt (siehe Abbildung 31).



Abbildung 31 Konfigurieren eines Button-Skripts

Eine Ausnahme hier ist geboten für das Ein- und Ausblenden der Hand Menüs über den Knopf Close. Dieser ändert auf Knopfdruck den Active Status des Game Objects Hand Menü auf *inactive* und lässt dasselbe so aus dem Sichtfeld ausblenden.

Um folgend Datum und Uhrzeit anzeigen zu lassen werden zwei Unity UI Labels auf der Oberfläche des Hand Menüs angebracht. Folgend werden, wie bereits bei vorherigen Funktionalitäten, zwei Skripte angelegt, welche über ihre durch die Klasse *MonoBehaviour* vererbten Update Methoden im Takt der Display Refresh Rate die Labels mit den aktuellen Daten versorgen. Auf die aktuellen Uhrzeit und Datums Daten kann über die *System* Klasse zugegriffen werden. Folgend wird dieser Text dem bereits initialisierten Label zugewiesen.

```
infoPanelDate.text = System.DateTime.Now.ToString("dd/MM/yy");
```

----- Schreiben des Datums- / Uhrzeit-Labels -----

```
infoPanelClock.text = System.DateTime.Now.ToString("HH:mm");
```

Nach Konfiguration der Buttons und Labels ist die Implementation des Hand Menüs abgeschlossen. Die fertiggestellte Implementation des Hand-Menüs wird bei erhobener Hand neben der Ulna Seite aktiviert und stellt den Einsatzkräften die geforderte Interaktionsschnittstelle zur Verfügung (siehe Abbildung 32). So kann das operative Personal der Feuerwehr während laufender Einsätze auf die einzelnen Funktionalitäten zugreifen und diese bspw. ins Sichtfeld einblenden lassen.

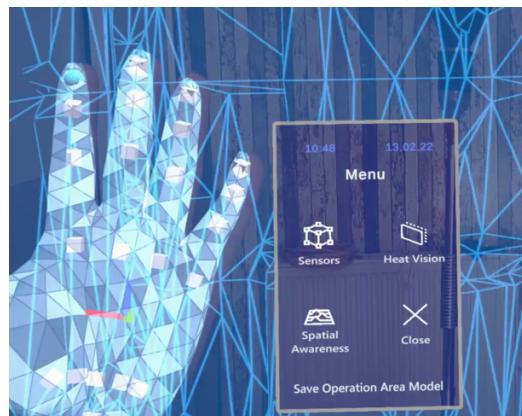


Abbildung 32 Hand-Menu des MARS 1 Prototypen

5.5 Integration in einen Brandbekämpfungshelm

Neben der Performance Augmentation verfolgt der Mars 1 Prototyp weitergehend das Ziel die tragende Einsatzkraft in gefährlichen Situationen auch vor möglichen Trümmer- und Splitterteilen zu schützen. So implementiert der Mars 1 das Head Mounted Display in einen außer Betrieb genommenen Feuerwehreinsatzhelm des Frankfurter Flughafens. Da die HoloLens 2 nicht ohne Anpassung in den Brandbekämpfungshelm passt, wird dieser unter Verwendung eines Winkelschleifers auf die ungefähren Maße derselben zugeschnitten. Folgend kann das Holo Modul in den Helm eingelassen und mittels Gaffer Tape am selben befestigt werden (siehe Abbildung 33). Schlussendlich wird das Sensormodul über Isolierband an der Helmrückseite befestigt, mit der Wärmebildkamera auf der Vorderseite des Helms (Kein Bild vorhanden).



Abbildung 33 Integration des Holo Moduls in einen Brandbekämpfungshelm

6 Untersuchung des Systems

Im Rahmen dieser Arbeit ist ein System entwickelt worden, welches unter Verwendung eines Mixed Reality Head Mounted Displays, diverser Sensoriken, eines Raspberry Pi und immersiver Software Einsatzkräfte in selektierten Eventualitäten ihres Berufsfelds unterstützen soll. Insbesondere bei der Brandaufklärung und Suche nach sowie dem Schutz von Personen. Um dieses System auf seine Genauigkeit und Zuverlässigkeit zu überprüfen, wurden mehrere Tests durchgeführt. Die Ergebnisse dieser werden folgend chronologisch aufgeschlüsselt abgebildet. Hierbei sind die Tests nach der bereits beschriebenen Methodik durchgeführt worden und gehen sowohl auf quantitative wie auch qualitative Merkmale der erhobenen Daten ein.

Am 02 Mai 2022 wurde auf der Atemschutzsimulationsfläche der Feuerwehr des Frankfurter eine erste Testserie mit Ziel der Überprüfung der Sensorik sowie der Wärmebildkameradaten des Systems durchgeführt. Ziel dabei ist es gewesen Differenzen in den aufgezeichneten Daten festzustellen und eine Person in dichtem Rauch mittels der WBK auf der Atemschutzsimulationsfläche zu lokalisieren. Dabei ist die Version 1 des Mars 1 Helms zum Einsatz gekommen. Sie bestand aus dem Holo- und dem Sensor Modul auf Basis des Raspberry Pi Zero W (siehe Kapitel 4 Implementierung Sensormodul). Das System war zu diesem Test noch nicht in einen Brandbekämpfungs-helm eingebunden (siehe Abbildung 34).

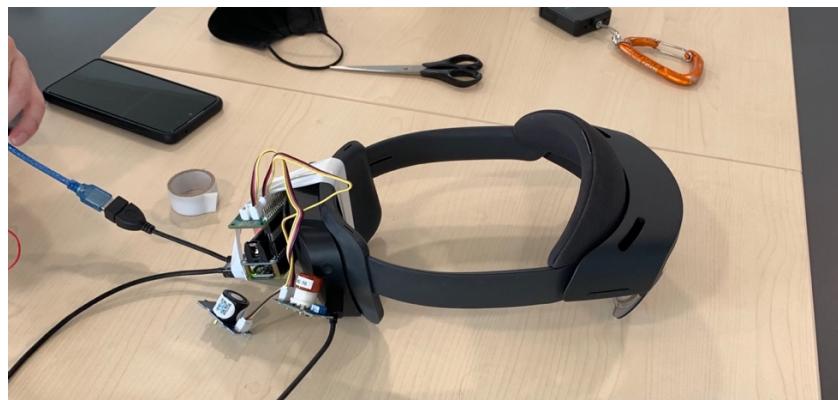


Abbildung 34 Version 1 des MARS 1 in der ersten Testserie mit Raspberry Pi Zero W

Bereits in den ersten Versuchen die Daten auszulesen und die Kameradaten zu empfangen, stellte sich heraus, dass diverse Probleme auftreten. So ist das Sensormodul

auf Basis des Raspberry Pi Zero W immer wieder an seine Limits in Bezug auf die eigene Rechenleistung und Hauptspeicherkapazität gekommen. Von Seiten des Holo Moduls musste notiert werden, dass die Raumerkennung in der Dunkelheit in ihrer Funktion eingeschränkt operierte. Daraus folgt, dass das Hand Menü zum Steuern der Applikation nicht Ulna seitig der Hand platziert werden konnte: die Möglichkeit das System zu steuern unterlag somit einem Totalausfall. Der Test wurde folgend nach 1 Stunde 40 Minuten (14:50 bis 16:30) abgebrochen, da es nicht gelungen ist kontinuierlich Daten zu empfangen und das System sich bei Aktivierung der Wärmebildkamera fortlaufend aufgehängt hat.

Am 04. Mai 2022 wurde das System auf den leistungsstärkeren Raspberry Pi 3B+ umgestellt und in Folge dieser auf den Empfang von Datenwerten sowie eine Verbesserung in der Performanz und Speicherauslastung getestet, da erste Vermutungen den geringen Hauptspeicher des Raspberry Pi Zero W als Kern des Problems identifiziert haben. Für die Umstellung wurde der auf den Raspberry Pi 3B+ ausgelegte Grove Base Hat bestellt und angebracht. Da sich das Raspberry Pi Image durch den Tausch der SD-Karte auf den 3B+ übertragen lässt, ist die Umstellung der Software ohne weitere Probleme ermöglicht worden. Folgend wurden die Sensoren am neuen Sensormodul angebracht, um es vollständig fertigzustellen (siehe Abbildung 35).

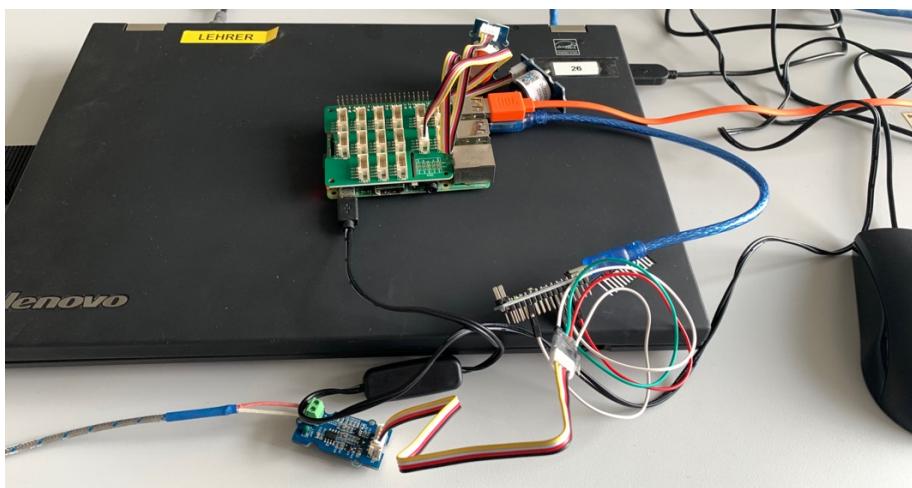


Abbildung 35 Sensormodul in der Konfiguration Raspberry Pi 3B+

Unter der neuen Konfiguration ist es gelungen alle Sensorwerte auszulesen und innerhalb der immersiven Software widerzugeben, ohne eine besonders auffällige Belas-

tung des Arbeitsspeichers festzustellen. Sobald die Wärmebildkamerafunktionalität jedoch aktiviert worden ist, kam es zu einer starken Belastung des Arbeitsspeichers. Dieser wurde in der Spur bis zu ~800 MB / 1024 MB belastet (ausgelesen mit *htop*). Durch weitere Tests wurde daraufhin festgestellt, dass eine ineffiziente Programmierung der Bildabfrage und ein schlechtes Speichermanagement hierfür verantwortlich gewesen sind. Die Folge ist ein harkendes und teilweise stehendes Bild im Holo-Modul, welches in einer aktiven Einsatzsituation nicht zu verwenden wäre. Der Test wurde daraufhin beendet, um Anpassungen an der Software des Sensormoduls durchzuführen. Nach Beendigung der Tests ist klar geworden, dass die Überlastung des Speichers durch die vermeidbar mehrfache Instanziierung der OpenCV *cv2.VideoCapture()* einhergeht. Nach der Änderung ist in einem weiteren kleinen Test am Folgetag eine deutliche Verminderung der Auslastung des Arbeitsspeichers und in Folge eine flüssige Übertragung des Wärmebilds festgestellt worden.

Der 06. Mai 2022 wurde als möglicher Ausweichtermin für die fehlgeschlagenen Tests des 02. May 2022 vereinbart. Aufgrund nicht weiter kommunizierter interner Komplikationen auf Seiten der Feuerwehr, kam an diesem Tag jedoch kein weiteres Treffen auf dem Gelände der Atemschutzsimulationsfläche zustande. So musste aus zeitlichen Gründen von weiteren Tests mit der Feuerwehr des Frankfurter Flughafens abgesehen werden. Glücklicherweise ließ sich jedoch mit einer lokalen freiwilligen Feuerwehr kurzfristig ein Termin für den 10. Mai 2022 vereinbaren, um eine kleinere Version der Tests durchzuführen.

Am 09. Mai 2022 wurde die Speicher optimierte Version der Wärmebildkamerasoftware in der Dualen Hochschule getestet. Das Ziel dieser Tests ist es gewesen Personen in den Gängen zu identifizieren. Dies konnte aufgrund unbehinderter Verhältnisse tadellos bewerkstelligt werden.

Vor den anstehenden Tests des 10. Mai 2022 ist es aufgrund von Terminunstimmigkeiten zwischen der Fraport Feuerwehr und dem Projektteam zu einer Umstellung des Testpartners gekommen. Hierzu wurde eine lokale Feuerwehr kontaktiert. Für das Testen der Genauigkeit der Sensorik wurde so folgend mit der Freiwilligen Feuerwehr Hattersheim-Okriftel (folgend die Feuerwehr genannt) die Hard- und Software optimierte Version des Sensormoduls in Verbindung mit dem Holo Modul getestet. Die

Tests fokussierten sich hierbei hauptsächlich auf die Korrektheit der ausgegebenen Sensorwerte und ihre Abweichungen in Relation zu Referenzmessungen der Feuerwehr, um die mit dem Mars 1 Prototypen erhobenen Daten interpretieren und in einen vergleichbaren Kontext rücken zu können. Für die Referenzmessung hat die Feuerwehr ihre Daten zu den geforderten Messwerten *Kohlenmonoxid* und *Sauerstoff* mit dem *Honeywell GasAlertQuattro*, dem von der Feuerwehr verwendeten 4-Gaswarngerät (für Spezifikationen siehe Anhang), aufgezeichnet. Dabei wurden die neun Messungen der Feuerwehr und des Projektteams parallel in einer Halle der Feuerwehr durchgeführt. Die erhobenen Daten sind manuell auf dem Mobiltelefon notiert worden, um diese in einer späteren Auswertung zu visualisieren und weiterzuverarbeiten. Um eine Änderung der Werte zu simulieren, wurde nach der sechsten Datenmessung der Motor eines Feuerwehreinsatzfahrzeugs (FWEF) gestartet. Dieser emittiert diverse Gase, darunter auch Kohlenstoffmonoxid. Die erste Auswertung wird sich auf die Auswertung der Aufzeichnungen exakt dieses Gases fokussieren.

Kohlenstoffmonoxid ist ein farb-, geruchs- und geschmacksloses Gas, welches bereits in geringer Dosis eine toxische Wirkung auf den menschlichen Körper haben kann. Es entsteht bei der unvollständigen Verbrennung von Kohlenstoffen und ist somit eines der am häufigsten vorkommenden Gase bei Bränden. Es entfaltet seine toxische Wirkung, indem es sich im Blutkreislauf an die roten Blutkörperchen heftet und den Sauerstoff verdrängt, was zu einer Sauerstoffunterversorgung und somit auch zum Tod führen kann [63], [64]. Da es sich bei diesem um ein vom menschlichen Körper nicht deduzierbares Gas handelt, kommen während Feuerwehreinsätzen diverse Detektoren zum Einsatz. Aufgrund der toxischen Wirkung ist eine akkurate Messung des Gases also von hohem Wert. In der durchgeföhrten Untersuchung über neun Datenpunkte sind die in Abbildung 36 graphisch Dargestellten Sensorwerte, bemessen in *parts per million*, aufgezeichnet worden. Um eine Änderung des umgebenden Luftgemisches zu erzeugen, wurde ab der sechsten Aufzeichnung der Motor des FWEF gestartet. Wie in der Abbildung zu sehen ist, hat der Start des Motors eine nicht unerhebliche Auswirkung auf die Aufgezeichneten Daten beider Messungen, sowohl der des Prototypen wie auch der des Referenzgeräts. Jedoch ist in der Untersuchung zu notieren, dass der Prototyp bereits vor dem Start des FWEF Motors einen erheblichen CO Gehalt in der Luft festgestellt hat, welcher jedoch durch die Referenzmessung nicht bestätigt werden konnte.

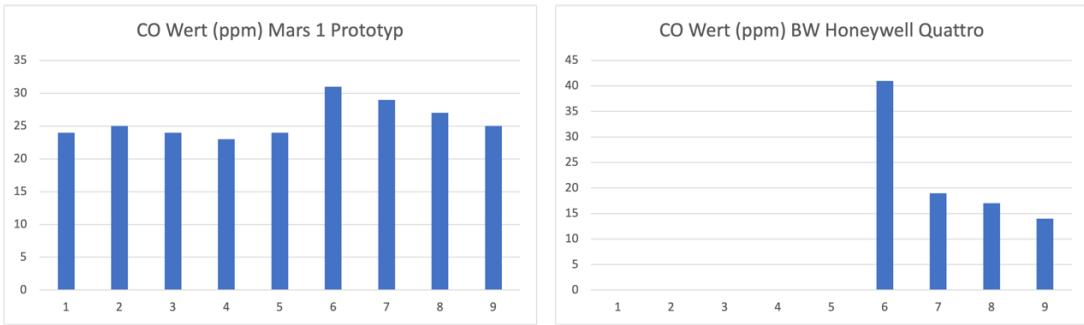


Abbildung 36 graph. Darstellung der CO Messwerte des Mars 1 sowie des von der Freiwilligen Feuerwehr Okrifel bedienten Honeywell GasAlertQuattro Multisensorgeräts. Ab dem sechsten Datenpunkt wurde in den parallel durchgeführten Messungen der Motor eines FWEF gestartet.

Der zweite Vektor der Messung und somit der letzte, welcher durch die Feuerwehr mit Referenzwerten in einen vergleichbaren Kontext gerückt werden konnte, bezog sich auf den Sauerstoffgehalt der Umgebung. Dieser wurde wie üblich in Prozent wieder-gegeben. Während der Messung konnte keine auffällige Schwankung in den Werten beobachtet werden, selbst, nachdem der Motor des FWEF gestartet worden ist. Hierbei ist zu beachten, dass erwartungsgemäß eine Minderung des Sauerstoffgehalts ab der sechsten Messung hätte stattfinden müssen, was im Falle dieser Untersuchung nicht aufgetreten ist. Der Wert ist über die neun Datenpunkte hinweg stabil verblieben, wobei eine konstante Differenz von ~4% zwischen den Mars 1 und den Referenzmess-werten vermerkt werden konnte.

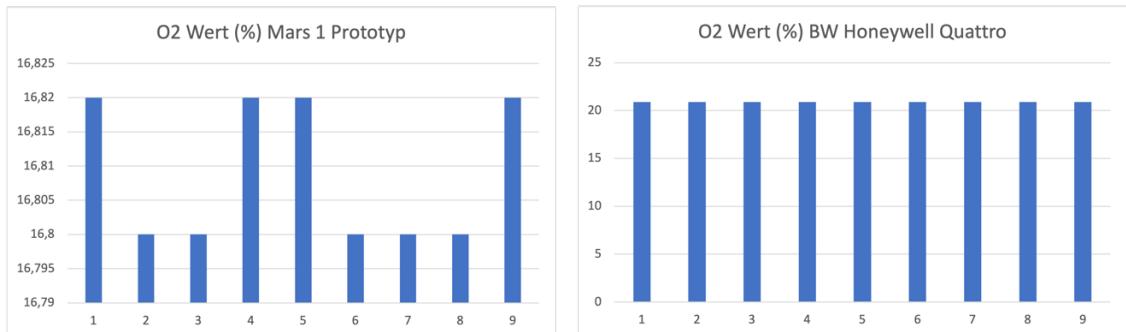


Abbildung 37 graph. Darstellung der O2 Messwerte des Mars 1 sowie des von der Freiwilligen Feuerwehr Okrifel bedienten Honeywell GasAlertQuattro Multisensorgeräts. Ab dem sechsten Datenpunkt wurde in den parallel durchgeführten Messungen der Motor eines FWEF gestartet. Es ist eine hohe Differenz feststellbar.

Der letzte Wert, welchen das System Mars 1 erhebt, ist der Temperatur Wert. Dieser dient der Orientierung der ungefähren Umgebungstemperatur. Außerdem kann dieser durch den Fühler des Sensormoduls auch gerichtet gelesen werden, um beispiels-

weise die Temperatur an einer gewissen Stelle auszulesen. Da es mit der lokalen Feuerwehr nicht möglich gewesen ist, diverse Temperaturen zu simulieren, wie es der Fall auf der Atemschutzstrecke der Flughafen Feuerwehr gewesen wäre, wurde hier ausschließlich die Umgebungstemperatur innerhalb der Fahrzeughalle getestet. Hierbei wurden durch die Freiwillige Feuerwehr auch keine Referenzdaten aufgezeichnet. Wie Abbildung 38 zu entnehmen ist, verblieben die Temperaturwerte konstant bei 29,5°C bis zu dem Zeitpunkt, an dem der FWEF Motor gestartet worden ist. Die von diesem ausgehende Hitze, hat die nahe Umgebung stark erwärmt und es folgte ein Ausschlag der Werte. Da diese Werte ausschließlich als Indikator dienen und des Weiteren nicht im Kontext dargestellt werden können, wird von einer weiteren Untersuchung derselben abgesehen. Nennenswert sei jedoch festgehalten, dass sich diese mit den durch der Wettervorhersage bereitgestellten sowie mit den aus einem FWEF ausgelesenen Temperaturwerten decken.

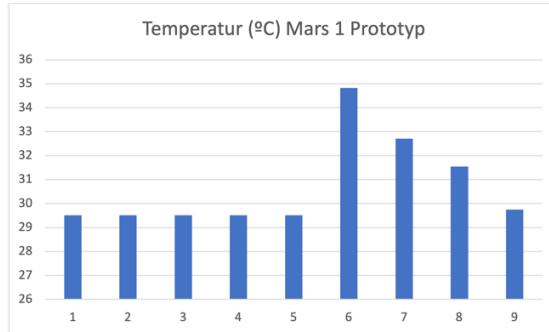


Abbildung 38 graph. Darstellung der mit dem Mars 1 Prototyp aufgezeichneten Temperaturwerte. Als Referenz liegt hier ausschließlich die Wettervorhersage vor. Ab dem sechsten Datenpunkt ist auch hier der Motor des FWEF gestartet und wieder ausgeschaltet worden.

Um die mittlere Abweichung zwischen den Messungen festzustellen, wurde aus den neun Datenpunkten jeweils für die Messungen der Feuerwehr sowie für die Messungen des Projektteams der Durchschnitt nach der Formel des arithmetischen Mittels

$$\bar{x} = \frac{\sum_1^n a_n}{n}$$

erzeugt. Um die so entstandenen vier Mittelwerte nun in die mittleren Abweichungen der Mars Werte zu den Referenzwerten umzuformen, werden diese voneinander abgezogen. Der absolute Wert dieser Differenz stellt die mittlere Abweichung der Messungen zueinander dar und kann Abbildung Abbildung 39 entnommen werden. Dabei fällt auf, dass während der Sauerstoffwert die bereits aus Abbildung 37 entnehmbare

vier Prozent Hürde abbildet, sich die mittlere Abweichung des wichtigen CO Werts um über 15ppm unterscheidet.

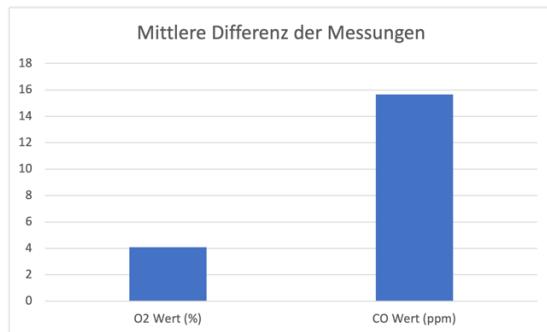


Abbildung 39 graph. Darstellung der mittleren Abweichung zwischen den Aufzeichnungen des Mars 1 Prototypen und des Honeywell GasAlertQuattro Referenzgeräts. Eine hohe Differenz im Bereich des CO Sensors ist zu verzeichnen.

Abschließend der Testserie muss festgehalten werden, dass sich während der Durchführung aller Tests herausgestellt hat, dass die Hologramme auf dem Display der Hololens 2 in hellen Umgebungen kaum zu erkennen sind. So dass es in einer hellen Umgebung oder beim direkten Blick auf helle Umgebungsstellen eine Auswertung der in der Sensordatenanzeige dargestellten Werte erschwert wird. Trotz der Abweichungen in den Messwerten und der Transparenz der Hologramme, ist dem Wehrführer der Freiwilligen Feuerwehr Hattersheim-Okriftel, Carsten Klebe, das System als „sehr beeindruckend“ aufgefallen.

7 Fazit und weiterer Ausblick

Ziel dieser Arbeit ist es gewesen einen funktionsfähigen Prototypen zu entwickeln, welcher die mit der Feuerwehr des Frankfurter Flughafens kommunizierten Anforderungen implementiert und dem operativen Personal so eine potenzielle Effizienzsteigerung in zukünftigen Einsätzen ermöglicht. So wurde hauptsächlich auf die vereinbarten Sensorwerte Temperatur, Kohlenstoffmonoxid und Sauerstoff sowie die ins System integrierte Wärmebildkamera, die Tiefenbilder und die Nutzerfreundlichkeit des Systems geachtet. Das in dieser Arbeit vorgestellte System implementiert dabei erfolgreich die kommunizierten Funktionalitäten und konnte für eine Einordnung in einer Untersuchung getestet werden. Wie die Untersuchung des entwickelten Systems zeigte, setzen die Anforderungen gewisse Voraussetzungen an die Rechenleistung der verwendeten Hard- und das Speichermanagement der implementierten Software. Dabei spielt, wie die ersten Tests darlegten, die Implementierung der Software des Sensormoduls in Bezug auf das Speichermanagement eine große Rolle, da die verwendete Raspberry Pi Produktreihe nicht auf hohe Belastungen ausgelegt ist. Ein effizienter Umgang mit den zur Verfügung stehenden Ressourcen, wie dem Arbeitsspeicher, ist somit von ungeahnter Wichtigkeit. Außerdem ist eine effiziente Implementierung der REST-Schnittstelle nötig. Durch einen Implementierungsfehler dieser Schnittstelle wurden die Objekte für die Sensoren sowie auch für die Wärmebildkamera immer wieder neu instanziert, was schlussendlich das gesamte Sensor-Modul überlastet hat. Durch Aufdeckung und Behebung des Implementierungsfehlers können die Daten der Wärmebildkamera sowie der Sensoren folgend problemlos und ohne Überlastung des Sensor-Modules an die HoloLens übertragen werden, was eine flüssige Wiedergabe derselben im Holo Modul zur Folge hat. Des Weiteren deckte die durchgeföhrte Untersuchung auf, dass das visible Light basierte Spatial Awareness System der HoloLens im Dunkeln Probleme aufzeigt, die Umgebung zu erkennen und in Folge an die Umgebung gebundene Hologramme nicht weiter dargestellt werden. Dies spiegelt sich wider, durch den Verlust der Darstellung des Hand Menüs sowie der Tiefenbilder in Umgebungen, welche durch ihren verminderten Lichteinfall das Erkennen der Umgebung über die *Visible Light Cameras* der HoloLens unmöglich machen. Als Beispiel seien dunkle Korridore oder schlecht belichtete Räumlichkeiten genannt. In diesen kann aufgrund der wenigen sammelbaren Spatial Mesh Daten kein Koordinaten-

system für die Positionierung der Hologramme erstellt werden. Daraus folgt, dass ausschließlich Blickfeld statische Hologramme, wie das HUD, abgebildet werden und das Steuersystem seine Funktionalität verliert. Da das Spatial Awareness System an fest-integrierte Sensoren gebunden ist, besteht unter Verwendung der HoloLens 2 ausschließlich die Möglichkeit ein externe Steuerungsschnittstelle zu implementieren, um eine sichere Steuerung des Moduls zu gewährleisten. Eine mögliche Lösung dieses Problems kann eine physische und möglicherweise Knopf basierte Steuerschnittstelle bspw. montiert auf der Brust der Einsatzkraft bieten, wie sie in den prototypisierten IVAS Systemen der US Army eingesetzt wird (siehe Abbildung 1). Anhand der in den Untersuchungen aufgezeichneten Messwerte sowie der Interpretation aller dokumentierten Probleme und Fehler des Systems, kann eine Einordnung des Mars 1 Mixed Reality Einsatzhelm Prototypen in Bezug auf seine Einsatzbereitschaft durchgeführt werden. So zeigte die Untersuchung, dass einem operativen Einsatz hohe Abweichungen in den wichtigen Sensorwerten Sauerstoff und Kohlenstoffmonoxid im Weg stehen. Insbesondere die hohe Abweichung der toxischen CO Werte in Vergleich mit den Referenzdaten zeigt, dass ein Einsatz unter der in dieser Arbeit vorgestellten Konfiguration des Sensormoduls katastrophal für die tragende Einsatzkraft enden kann. Einzig durch die Verwendung akkuratester Sensorik wäre eine einsatzbereite Implementierung des Einsatzhelmsystems denkbar, wenn auch das Grundkonzept bei operativem Feuerwehr Personal einen hohen Anklang gefunden hat. So zeigten sich diverse Feuerwehrleute interessiert an der Thematik und bezeichneten das System bei Vorführung als sinngemäß „sehr beeindruckend“ und „zukunftsträchtig“ [65].

Darüber hinaus entstehen auf Basis der durchgeföhrten Untersuchung Fragen und Weiterentwicklungsmöglichkeiten, welche in zukünftigen Projekten bearbeitet werden können. So werden diese in den folgenden Absätzen behandelt.

Potenzielle Weiterentwicklung des Systems

Ein Ziel dieser Arbeit ist es gewesen, das System möglichst modular aufzubauen, um in zukünftigen Iteration weitere Module anschließen zu können und das System so zu erweitern oder auf andere Branchen wie das THW umzustellen. Für diese Modularität ist insbesondere das Sensormodul verantwortlich, da hier die einzelnen Datensysteme zusammenlaufen, deren Werte dem Holo Modul über eine REST Schnittstelle bereit-

gestellt werden. Es existieren unterschiedliche Möglichkeiten das in dieser Arbeit vorgestellte System zu erweitern, beispielsweise durch den einfachen Aufbau des modularen Sensormoduls, so dass weitere Sensoren an das Grove Base Hat angeschlossen werden und dieses um diverse Sensorsysteme vergrößert wird. Hierbei besteht die Möglichkeit das vorliegende System von der Fokussierung auf den Feuerwehr Bereich hinweg auf andere Bereiche wie Polizei oder THW umzustellen, indem über neue Anforderungen kommuniziert und Systemanpassungen durchgeführt werden. Weitere Möglichkeiten der Weiterentwicklung bieten

1. die Vervollständigung der noch nicht funktionstüchtigen Funktionalität des Aufzeichnens einer Operation Area Karte. Unter der Bedingung, dass die Fehler im Abspeichern der Karte beiseitegelegt werden, können aufbauend auf dieser diverse Funktionen implementiert werden. Dabei kann ein zukünftiges System mit dem Scene Understanding Kit des MRTK über eine solche Karte hinweg navigieren und Einsatzkräften so die Möglichkeit geben schnell aus gefährlichen Situationen zu entkommen. Darüber hinaus könnten diese Karten unter mehreren Instanzen des Systems verteilt auf mehrere Einsatzkräfte synchronisiert werden, um eine genauere Karte des Einsatzgebiets zu generieren. Unter Verwendung dieser kann zusätzlich die Positionierung der Team Kollegen anhand ihrer Positionsdaten innerhalb der geteilten Karte eingetragen werden kann.
2. der Aufbau auf der Forschung von Christian Schönauer, Emmanuel Vonach, Georg Gerstweiler und Hannes Kaufmann. So kann eine synchronisierte Karte gleichzeitig mit den Wärmedaten der integrierten Kamera überlagert werden. So entsteht eine für alle Teammitglieder zugängliche Operation Area Karte, mit genauer Darstellung der aufgezeichneten Hitzesignaturen.
3. Das Einbinden intelligenter Systeme. Durch die stetige Weiterentwicklung im Bereich der Künstlichen Intelligenz insbesondere in den Bereichen Sprach- und Bildverarbeitung kann in zukünftigen Iterationen außerdem auf die Implementierung diverser intelligenter Funktionalitäten, welche die Performance der Einsatzkräfte weiter steigern, eingegangen werden. Dabei besteht die Möglichkeit eine Sprach basierte Steuerung des Systems über Language Processing Module wie GPT-3 oder Personenerkennung innerhalb des HUDs entwickelt mit

TensorFlow und Image Segmentation Techniken hinzuzufügen. Auch auf dem Gebiet der intelligenten Vorhersagen kann aufbauend auf synchronisierten Operation Area Modellen weiter geforscht werden, indem intelligente Modelle in einem multidimensionalen Vektorraum eine Überlagerung der Hitze-, Karten und Sensorwerte erstellen, um Einsatzkräfte frühzeitig vor potenziellen Gefahrensituationen zu bewahren.

Abschließend zu dieser Arbeit werden Fragen aufgeworfen, welche im Rahmen dieser Arbeit nicht beantwortet werden konnten und möglicherweise in zukünftigen Forschungsiterationen behandelt werden können, um das Spektrum des Mixed Reality Einsatzhelms auf andere Einsatzgebiete auszuweiten und die Grenzen des Systems zu erforschen.

1. Die Frage, weshalb das Kartierungssystem nicht vollständig einsatzfähig funktioniert und weshalb die Applikation ohne genauen Fehlerlog zusammenbricht, konnte im Rahmen dieser Arbeit nicht vollständig geklärt werden.
2. Des Weiteren bleibt offen, inwiefern die Rechenleistung und die maximale Tragedauer des HoloLens Systems ausgereizt werden können, sollte mehr Sensorik angeschlossen und weitere Funktionalitäten implementiert werden.
3. Weiterführend kann erforscht werden wie viele Einsätze der Helm durchhält bis der Akkumulator an Leistung verliert und inwiefern sich die mögliche Einsatzdauer des Systems daraufhin verändert.
4. Alternative Hardware als Grundbaustein des Mars 1 Helms ist eine Forschungsfrage, welcher sich in dieser Arbeit nicht gewidmet werden konnte. Hierbei wäre es interessant alternative HMDs, Laufzeitumgebungen und Sensoren auszutesten und in Vergleich zu diesem System zu evaluieren.

Für die Beantwortung der aufgekommenen Fragen, können zukünftige Forschungsarbeiten auf eine Re-Implementierung mit anderen Bausteinen setzen oder das in dieser

Arbeit vorgestellte System um die forschungsbedingt benötigten Funktionalitäten erweitern. So kann abschließend gesagt werden, dass die in dieser Arbeit durchgeführte Untersuchung zeigt, dass ein modulares Mixed Reality System eine große Reichweite an Bedürfnissen abdecken kann und somit das Potenzial birgt, einer neuen Generation von Einsatzhelmen als Grundlage zu dienen. Stehen einem Unternehmen oder einer Forschenden Gemeinschaft die nötigen Mittel zur Verfügung, bestünde die Möglichkeit ein solches System nach heutigem Stand der Technologie unter bestimmten Veränderungen einsatzfähig auf den Markt zu bringen.

I. Abkürzungsverzeichnis

Advanced RISC Machines	ARM
American Standard Code for Information Interchange	ASCII
Analog Digital Converter	ADC
Application Programming Interface	API
Carbon monoxid	CO
Dynamic Host Configuration Protocol	DHCP
Feuerwehreinsatzfahrzeug	FWEF
Forward Looking Infrared	FLIR
Head(-)Mounted(-)Display	HMD
Heads(-)Up(-)Display	HUD
General-Input-Output	GPIO
Inertial Measurement Unit	IMU
Input-Output	I/O
Integrated Visual Augmentation System	IVAS
Inter-Integrated Circuit	I2C
Liquified Petroleum Gas	LPG
Longwave Infrared	LWIR
Methan	CH4
MicroController Unit	MCU
Midwave Infrared	MWIR
Mixed Reality Toolkit	MRTK
Near Infrared	NIR
parts per million	ppm
Polyactid Filament	PLA
Power Management	PWM
Representational State Transfer	REST
Scene Understanding	SU
Scene Understanding Kit	SUKit
Shortwave Infrared	SWIR
Spatial Coordinate System	SCS
Secure Sockets Layer	SSL
System-on-a-chip	SOC

Time of Flight	ToF
Universal Asynchronous Receiver Transmitter	UART
User Friendly Firewall	UFW
Wärmebildkamera	WBK

II. Tabellenverzeichnis

Tabelle 1 Liste der Funktionalitäten	4
Tabelle 2: Anforderungen Basisgerät.....	18
Tabelle 3: Anforderungen Stromversorgung	19
Tabelle 4: Anforderungen Sensor-Schnittstelle	20
Tabelle 5: Anforderungen Wärmebildkamera-Schnittstelle	21
Tabelle 6: Überprüfung Erfüllung Anforderungen.....	25
Tabelle 7 Zusätzliche Funktionen und Imports der Funktionalität Wärmesicht	41
Tabelle 8 Überblick RawImage	52
Tabelle 9 SpatialAwarenessSwitch Funktionen.....	59
Tabelle 10 Funktionen des Kartographiesystems	64

III. Abbildungsverzeichnis

Abbildung 1 Ein Soldat der 1-508PIR, 82nd Airborne Division beim Testen des IVAS [10]	7
Abbildung 2 Reality / Virtuality Continuum [11], [13].....	9
Abbildung 3 Beispiel der Handerkennung mit fixiertem Hologramm [19]	13
Abbildung 4 Visualisierung des von der Hololens erkannten Umfelds [26]	15
Abbildung 5: Übersicht Sensor-Modul	17
Abbildung 6: ESP8266Mod12-F Mikrocontroller [29].....	22
Abbildung 7: MQ9-Sensor [31].....	23
Abbildung 8: MQ135-Sensor [33]	23
Abbildung 9: DHT11-Sensor [35].....	24
Abbildung 10: Raspberry Pi Zero W [38]	27
Abbildung 11: Battery Hat [40]	27
Abbildung 12: Grove Base Hat [42].....	28
Abbildung 13: Grove High Temperature Sensor [44].....	28
Abbildung 14: Grove Oxygen Sensor [46].....	29
Abbildung 15: Grove CO Sensor [48].....	29
Abbildung 16 Teledyne FLIR Lepton 3.5 ohne I/O Module in Vergleich zu einem kleinen Geldstück [51].....	31
Abbildung 17 Anforderungen an das Holo-Modul	46
Abbildung 18 Erstellen einer Unity 3D Applikation mit Unity Hub.....	48
Abbildung 19 Mixed Reality Toolkit Konfigurationsdialog	48
Abbildung 20 Nutzung des Mixed Feature Toolkit	49
Abbildung 21 HUD Canvas.....	50
Abbildung 22 Geplantes User Interface.....	50
Abbildung 23 Konfigurieren von public Parametern mit Referenzen auf UI Objekte .	53
Abbildung 24 Das immersive Wärmebild integriert ins HUD	55
Abbildung 25 Beschrifteter Rohling des Sensordatencontainers.....	56
Abbildung 26 Hinzufügen eines Spatial Mesh Observers.....	60
Abbildung 27 Darstellung der aktivierten Tiefenanzeige	61
Abbildung 28 Selbsterstellte dreidimensionale Kartographie	63
Abbildung 29 Auswahl der Ulna Seite in der Hand Menü Konfiguration.....	67
Abbildung 30 UI-Rohling des Hand Menüs	68
Abbildung 31 Konfigurieren eines Button-Skripts.....	68
Abbildung 32 Hand-Menu des MARS 1 Prototypen.....	69
Abbildung 33 Integration des Holo Moduls in einen Brandbekämpfungshelm.....	70
Abbildung 34 Version 1 des MARS 1 in der ersten Testserie mit Raspberry Pi Zero W	71
Abbildung 35 Sensormodul in der Konfiguration Raspberry Pi 3B+	72
Abbildung 36 graph. Darstellung der CO Messwerte des Mars 1 sowie des von der Freiwilligen Feuerwehr Okriftel bedienten Honeywell GasAlertQuattro Multisensorgeräts. Ab dem sechsten Datenpunkt wurde in den parallel durchgeführten Messungen der Motor eines FWEF gestartet.	75
Abbildung 37 graph. Darstellung der O2 Messwerte des Mars 1 sowie des von der Freiwilligen Feuerwehr Okriftel bedienten Honeywell GasAlertQuattro Multisensorgeräts. Ab dem sechsten Datenpunkt wurde in den parallel	

durchgef�hrten Messungen der Motor eines FWEF gestartet. Es ist eine hohe Differenz feststellbar.....	75
Abbildung 38 graph. Darstellung der mit dem Mars 1 Prototyp aufgezeichneten Temperaturwerte. Als Referenz liegt hier ausschlie�lich die Wettervorhersage vor. Ab dem sechsten Datenpunkt ist auch hier der Motor des FWEF gestartet und wieder ausgeschaltet worden.	76
Abbildung 39 graph. Darstellung der mittleren Abweichung zwischen den Aufzeichnungen des Mars 1 Prototypen und des Honeywell GasAlertQuattro Referenzger�ts. Eine hohe Differenz im Bereich des CO Sensors ist zu verzeichnen.	77

IV. Kapitelaufteilung

Kapitel	Geschrieben von
<i>1 Einleitung</i>	Jonas & Joel
<i>1.1 Motivation</i>	Jonas
<i>1.2 Herangehensweise und Methodik</i>	Jonas & Joel
<i>1.3 Stand der Technik</i>	Jonas
<i>2 Grundlagen HoloLens 2</i>	Jonas
<i>3 Grundlagen des Sensormodul</i>	Joel
<i>3.3 Verbessertes Sensor-Modul: Abschnitt Wärmebildkamera</i>	Jonas & Joel
<i>4 Implementierung Sensormodul</i>	Joel
<i>4.3 Wärmebildkamera-Schnittstelle</i>	Jonas & Joel
<i>5 Implementierung Holo-Modul</i>	Jonas
<i>6 Untersuchung des Systems</i>	Jonas & Joel
<i>7 Fazit und weiterer Ausblick</i>	Jonas & Joel

V. Literaturverzeichnis

- [1] „Handelszeitung,“ Ringier Axel Springer Schweiz AG, 08 09 2016. [Online]. Available: <https://www.handelszeitung.ch/panorama/star-trek-als-inspiration-nur-beamen-bleibt-ein-traum>. [Zugriff am 31 01 2022].
- [2] „BAE Systems,“ [Online]. Available: <https://www.baesystems.com/en/feature/our-innovations-hud>. [Zugriff am 31 01 2022].
- [3] „BPB.de,“ [Online]. Available: <https://www.bpb.de/kurz-knapp/hintergrund-aktuell/337277/jahrhunderthochwasser-2021-in-deutschland/>. [Zugriff am 05 05 2022].
- [4] „Handelsblatt.com,“ [Online]. Available: <https://www.handelsblatt.com/politik/international/usa-hitze-duerre-grossbraende-der-kalifornische-traum-wird-von-der-realitaet-eingeholt/27416006.html>. [Zugriff am 05 05 2022].
- [5] „Tagesschau.de,“ [Online]. Available: <https://www.tagesschau.de/ausland/europa/weltklimarat-bericht-vorab-101.html>. [Zugriff am 05 05 2022].
- [6] C. Schönauer, E. Vonach, G. Gerstweiler und H. Kaufmann, „3D building reconstruction and thermal mapping in fire brigade operations,“ *2013 IEEE Virtual Reality (VR)*, pp. 1-2, 2013.
- [7] „Army.mil,“ United States Army, [Online]. Available: <https://www.peosoldier.army.mil/Program-Offices/Project-Manager-Integrated-Visual-Augmentation-System/>. [Zugriff am 09 02 2022].
- [8] S. Nellis und P. Dave, „Reuters,“ 31 03 2021. [Online]. Available: <https://www.reuters.com/article/us-microsoft-army-idCAKBN2BN36B>. [Zugriff am 09 02 2022].
- [9] „Youtube.com,“ ABC News, 19 12 2020. [Online]. Available: <https://www.youtube.com/watch?v=XTV13SDWB-g>. [Zugriff am 09 02 2022].
- [10] „Army.mil,“ United States Army, 04 2021. [Online]. Available: <https://www.peosoldier.army.mil/portals/53/Images/ivas-overview.jpg>. [Zugriff am 09 02 2022].
- [11] P. Milgram und F. Kishino, „A Taxonomy of Virtual Reality Visual Displays,“ *IEICE Transactions on Information Systems*, Bde. %1 von %2Vol E77-D, Nr. 12, December 1994.
- [12] J. Ronsdorf, „Microsoft.com,“ Microsoft, 04 12 2020. [Online]. Available: <https://news.microsoft.com/de-de/microsoft-erklaert-was-ist-microsoft-hololens-definition-funktionen/>. [Zugriff am 10 02 2022].
- [13] „conrad.com,“ [Online]. Available: <https://asset.conrad.com/media10/isa/160267/c3-/de/a3b02e5a5544c4b85b78c7cebbe71613e/reality-virtuality-continuum.jpg?x=514>. [Zugriff am 2022 02 06].
- [14] „Microsoft.com,“ Microsoft, [Online]. Available: <https://www.microsoft.com/de-de/hololens/hardware>. [Zugriff am 10 02 2022].
- [15] J. Ashley, „ImaginativeUniversal.com,“ 18 10 2015. [Online]. Available: <https://www.imaginativeuniversal.com/blog/2015/10/18/how-hololens-displays-work/>. [Zugriff am 10 02 2022].

- [16] „Microsoft Holographic Processing Unit,“ Microsoft, [Online]. Available: <https://www.microsoft.com/en-us/research/blog/second-version-hololens-hpu-will-incorporate-ai-coprocessor-implementing-dnns/>. [Zugriff am 25 04 2022].
- [17] „Windows United,“ [Online]. Available: <https://windowsunited.de/microsoft-zeigt-erste-details-von-hololens-und-hpu-2-0/>. [Zugriff am 25 04 2022].
- [18] J. Roach, „Microsoft.com,“ 07 11 2019. [Online]. Available: <https://news.microsoft.com/innovation-stories/hololens-2-shipping-to-customers/>. [Zugriff am 14 02 2022].
- [19] „Docs.Microsoft.com,“ Microsoft, [Online]. Available: <https://docs.microsoft.com/de-de/windows/mixed-reality/design/images/ulnarsidehandmenu.gif>. [Zugriff am 14 02 2022].
- [20] „Docs.Microsoft.com,“ Microsoft, [Online]. Available: <https://docs.microsoft.com/en-us/windows/mixed-reality/design/coordinate-systems>. [Zugriff am 16 02 2022].
- [21] A. Leber, C. Rindchen und S. Bergner, „Ein Konzept zur Registrierung von Koordinatensystemen zur Interaktion von AR-Endgeräten mit Robotersystemen und Implementierung exemplarischer Aufgaben,“ Duale Hochschule Baden-Württemberg, Karlsruhe, 2021.
- [22] S. Ong und V. K. Siddaraju, Beginning Windows Mixed Reality Programming: For HoloLens and Mixed Reality Headsets, Bd. Second Edition, Apress, 2021.
- [23] „Docs.Microsoft.com,“ [Online]. Available: <https://docs.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping>. [Zugriff am 16 02 2022].
- [24] H. Durrant-Whyte und T. Bailey, „Simultaneous localization and mapping (SLAM): part II,“ *IEEE Robotics Automation Magazine*, Bd. 13, Nr. 3, pp. 108-117, 2006.
- [25] H. Durrant-Whyte und T. Bailey, „Simultaneous localization and mapping: part I,“ *IEEE Robotics Automation Magazine*, Bd. 13, Nr. 2, pp. 99-110, 2006.
- [26] „Docs.Microsoft.com,“ [Online]. Available: <https://docs.microsoft.com/en-us/windows/mixed-reality/design/images/surfacereconstruction.jpg>. [Zugriff am 16 02 2022].
- [27] „Docs.Microsoft.com,“ Microsoft, [Online]. Available: <https://docs.microsoft.com/en-us/windows/mixed-reality/design/scene-understanding>. [Zugriff am 12 04 2022].
- [28] „az-delivery,“ az-delivery, [Online]. Available: https://cdn.shopify.com/s/files/1/1509/1638/products/lolinunverlotetmain_500x.jpg?v=1606123869. [Zugriff am April 2022].
- [29] az-delivery, „az-delivery.de,“ az-delivery, [Online]. Available: <https://www.az-delivery.de/products/nodemcu>. [Zugriff am April 2022].
- [30] makershop, „makershop.de,“ makershop, [Online]. Available: <https://www.makershop.de/sensoren/gas/mq-9-gas-sensor/>. [Zugriff am April 2022].
- [31] „makershop.de,“ makershop, [Online]. Available: https://www.makershop.de/wp-content/uploads/2015/11/61spj9p-EGL._SL1010_.jpg. [Zugriff am April 2022].
- [32] „az-delivery.de,“ az-delivery, [Online]. Available: <https://www.az-delivery.de/products/mq-135-gas-sensor-modul>. [Zugriff am April 2022].
- [33] „az-delivery.de,“ az-delivery, [Online]. Available: https://cdn.shopify.com/s/files/1/1509/1638/products/1.Main_1x_MQ-135GasSensorModul_500x.jpg?v=1617085843. [Zugriff am April 2022].

- [34] „az-delivery.de,“ az-delivery, [Online]. Available: <https://www.az-delivery.de/products/5-x-dht11-temperatursensor>. [Zugriff am April 2022].
- [35] AZDelivery, „amazon.de,“ AZDelivery, [Online]. Available: <https://www.amazon.de/AZDelivery-DHT11-Temperatursensor-Luftfeuchtigkeitssensor-Parent/dp/B07TYPT2NJ?th=1>. [Zugriff am April 2022].
- [36] zdnet, „zdnet.com,“ zdnet, [Online]. Available: <https://www.zdnet.com/home-and-office/smart-office/raspberry-pi-at-10-the-tiny-board-with-a-giant-impact/>. [Zugriff am April 2022].
- [37] sparkfun, „cdn.sparkfun.com,“ sparkfun, [Online]. Available: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwi_8MOwyuP3AhVNf0HHYOCq4QFhoECAQQAQ&url=https%3A%2F%2Fcdn.sparkfun.com%2Fassets%2Flearn_tutorials%2F6%2F7%2F6%2FPiZero_1.pdf&usg=AOvVaw3_DKHikQ_3YUY9ty4_iE5. [Zugriff am April 2022].
- [38] berrybase, „berrybase.de,“ berrybase, [Online]. Available: https://www.berrybase.de/media/image/07/80/db/ID_61859_origFZE3SL5UuFgoj.jpg. [Zugriff am April 2022].
- [39] berrybase, „berrybase.de,“ berrybase, [Online]. Available: https://www.berrybase.de/raspberry-pi/raspberry-pi-computer/gpio-hats-phats/power-management/mobile-stromversorgung/li-ion-akku-hat-f-252-r-raspberry-pi?sPartner=g_shopping&gclid=EA1alQobChMlx5Sficvj9wIVCtZ3Ch3IdgAYEAQYASABEgLC1PD_BwE. [Zugriff am April 2022].
- [40] berrybase, „berrybase.de,“ berrybase, [Online]. Available: https://www.berrybase.de/media/image/25/80/57/ID_65812_orig.jpg. [Zugriff am April 2022].
- [41] berrybase, „berrybase.de,“ berrybase, [Online]. Available: <https://www.berrybase.de/raspberry-pi/raspberry-pi-computer/gpio-hats-phats/breakouts/seeed-grove-base-hat-f-252-r-raspberry-pi-zero>. [Zugriff am April 2022].
- [42] berrybase, „berrybase.de,“ berrybase, [Online]. Available: https://www.berrybase.de/media/image/f4/97/80/ID_104957_orig.jpg. [Zugriff am April 2022].
- [43] berrybase, „berrybase.de,“ berrybase, [Online]. Available: <https://www.berrybase.de/sensoren-module/temperatur/seeed-grove-high-temperature-sensor>. [Zugriff am April 2022].
- [44] berrybase, „berrybase.de,“ berrybase, [Online]. Available: https://www.berrybase.de/media/image/40/40/76/ID_105098_orig.jpg. [Zugriff am April 2022].
- [45] seedstudio, „wiki.seedstudio.com,“ seedstudio, [Online]. Available: https://wiki.seeedstudio.com/Grove-Gas_Sensor-O2-MIX8410/. [Zugriff am April 2022].
- [46] seedstudio, „wiki.seedstudio.com,“ seedstudio, [Online]. Available: https://wiki.seeedstudio.com/Grove-Gas_Sensor-O2-MIX8410/. [Zugriff am 2022 Mai].
- [47] seeedstudio, „wiki.seeedstudio.com,“ seeedstudio, [Online]. Available: https://wiki.seeedstudio.com/Grove-Gas_Sensor-MQ9/. [Zugriff am April 2022].
- [48] .berrybase, „.berrybase.de,“ .berrybase, [Online]. Available: https://www.berrybase.de/media/image/bb/1d/21/ID_105109_orig.jpg. [Zugriff am Mai 2022].

- [49] „Xenics Infrared Solutions,“ [Online]. Available: <https://www.xenics.com/infrared-technologies/>. [Zugriff am 08 05 2022].
- [50] „Teledyne FLIR - cooled or uncooled,“ [Online]. Available: <https://www.flir.com/discover/rd-science/cooled-or-uncooled/>. [Zugriff am 11 04 2022].
- [51] „flir.de,“ Teledyne, [Online]. Available: https://www.flir.de/globalassets/imported-assets/image/lepton_pdp.png. [Zugriff am 12 05 2022].
- [52] „Teledyne FLIR,“ [Online]. Available: <https://www.flir.de/products/lepton/?model=3.5+Lepton&vertical=lwir&segment=oem>. [Zugriff am 07 04 2022].
- [53] pcwelt, „pcwelt.de,“ pcwelt, [Online]. Available: <https://www.pcwelt.de/ratgeber/Raspberry-Pi-Schneller-dank-Logdateien-im-RAM-11047025.html>. [Zugriff am April 2022].
- [54] tecmint, „tecmint.com,“ tecmint, [Online]. Available: <https://www.tecmint.com/setup-ufw-firewall-on-ubuntu-and-debian/>. [Zugriff am April 2022].
- [55] billz, „github.com,“ [Online]. Available: <https://github.com/RaspAP/raspap-webgui>. [Zugriff am Mai 2022].
- [56] „Docs.Microsoft.Com,“ [Online]. Available: <https://docs.microsoft.com/de-de/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05>. [Zugriff am 17 02 2022].
- [57] „Unity.com,“ [Online]. Available: <https://unity.com/de/our-company>. [Zugriff am 17 02 2022].
- [58] „iGlobalPartners,“ [Online]. Available: <https://www.iglobepartners.com/portfolio-unity.html>. [Zugriff am 11 04 2022].
- [59] „TextMeshPro,“ [Online]. Available: <https://docs.microsoft.com/de-de/windows/mixed-reality/develop/unity/text-in-unity>. [Zugriff am 11 04 2022].
- [60] „MRTK Spatial Awareness,“ Microsoft, [Online]. Available: <https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/spatial-awareness/usage-guide?view=mrtkunity-2021-05>. [Zugriff am 15 04 2022].
- [61] „Docs.Microsoft.com,“ Microsoft, [Online]. Available: <https://docs.microsoft.com/de-de/windows/mixed-reality/develop/unity/scene-understanding-sdk>. [Zugriff am 12 03 2022].
- [62] „MRTK - Hand Menu,“ [Online]. Available: <https://docs.microsoft.com/de-de/windows/mixed-reality/design/hand-menu>. [Zugriff am 12 04 2022].
- [63] „fwvbw.de,“ Feuerwehrverband BW, [Online]. Available: <https://www.fwvbw.de/kohlenstoffmonoxid-die-unsichtbare-gefahr,35.html>. [Zugriff am 10 05 2022].
- [64] „thieme.de,“ [Online]. Available: <https://www.thieme.de/de/anaesthesiologie/kohlenmonoxid-unterschaetze-gefahr-patienten-retter-119022.htm>. [Zugriff am 10 05 2022].
- [65] C. Klebe und L. Hehlgans, Interviewees, [Interview]. 02 sowie 10 05 2022.
- [66] „fluke.com,“ Fluke, [Online]. Available: <https://www.fluke.com/de-de/mehr-erfahren/blog/thermografie/wie-funktionieren-waermebildkameras>. [Zugriff am 31 01 2022].

- [67] „fluke.com,“ Fluke, [Online]. Available: <https://dam-assets.fluke.com/s3fs-public/styles/800x534/public/flukeig/articles/images-general-web-cards/web-cards/training/6005944-card-ti-how-they-work-715x360.jpg>. [Zugriff am 31 01 2022].
- [68] H. Durrant-Whyte, D. C. Rye und E. M. Nebot, „Localization of Autonomous Guided Vehicles,“ Nr. DOI:10.1007/978-1-4471-0765-1_69, 1996.
- [69] az-delivery, „az-delivery,“ az-delivery, [Online]. Available: <https://www.az-delivery.de/products/nodemcu>. [Zugriff am April 2022].

VI. Anhang

Tabellarische Auflistung der Forschungsergebnisse

Mars 1 Messungen

Temperatur (°C)	O2 Wert (%)	CO Wert (ppm)	Kommentar
29,51	16,82	24	Truck aus
29,51	16,8	25	Truck aus
29,51	16,8	24	Truck aus
29,51	16,82	23	Truck aus
29,51	16,82	24	Truck aus
34,83	16,8	31	Truck an
32,71	16,8	29	Truck aus
31,55	16,8	27	Truck aus
29,75	16,82	25	Truck aus
30,71	16,80888889	25,77777778	

BW Honeywell Quattro Messungen

Temperatur (°C)	O2 Wert (%)	CO Wert (ppm)	Kommentar
null	20,9	0	Truck aus
null	20,9	0	Truck aus
null	20,9	0	Truck aus
null	20,9	0	Truck aus
null	20,9	0	Truck aus
null	20,9	41	Truck an
null	20,9	19	Truck aus
null	20,9	17	Truck aus
null	20,9	14	Truck aus
20,9	10,11111111		

DifferenzwertTabelle

O2 Wert (%)	CO Wert (ppm)
4,091111111	15,66666667

BW Honeywell Quattro Datenblatt



**GasAlert
Quattro**

Multigaswarngerät



H₂S

CO

O₂

UEG

Visuelle Prüfung, einfache Kontrolle

Das robuste und zuverlässige GasAlertQuattro 4-Gaswarngerät verfügt über ein umfangreiches Funktionsangebot mit einfacher Eintastenbedienung. Dank der flexiblen Stromversorgungsoptionen ist der GasAlertQuattro jederzeit einsatzbereit. Auf der grafischen LCD-Anzeige erscheinen leicht zu identifizierende Symbole zur Anzeige von Informationen zum Gerätebetrieb, z. B. Funktionstests und Kalibrierungen, die eine Prüfung vor Ort vereinfachen. IntelliFlash liefert kontinuierlich visuelle Informationen zur einwandfreien Funktion des Gaswarngeräts. Der GasAlertQuattro eignet sich für eine breite Palette industrieller Anwendungen, einschließlich des Einstiegs in CS-Bereiche, und ist mit dem automatischen Test- und Kalibriersystem MicroDock II von BW kompatibel. GasAlertQuattro verfügt über den europäischen Leistungsnaheis und entspricht der europäischen Schiffsaurüstungsrichtlinie (MED).



Einfache Bedienung mit einer Taste



Jederzeit einsatzbereit



Einfache visuelle Funktionskontrolle

- Minimale Kosten und Einarbeitung dank Eintastenbedienung
- Forderprobe Surecell-Sensoren bieten eine noch nie dagewesene Leistung selbst unter extremen Einsatzbedingungen
- Austauschbare Stromversorgungsoptionen mit erweiterter Akkulaufzeit für längere Schichten

WASSERDICHT

BW
Technologies
by Honeywell

Wear yellow. Work safe.
See 'green.'

Standardmerkmale von BW-Produkten:

- Permanentes LCD zeigt Gaskonzentrationen in Echtzeit an
- Kompakte und leichte Bauweise sorgt für hohen Tragekomfort
- Einfaches Verfahren zur automatischen Justierung: kompatibel mit der automatischen Test- und Kalibrierstation MicroDock II von BW
- Selbsttest aller Funktionen von Sensor(en), Batterie/Akku und Elektronik sowie der akustischen/optischen Alarne beim Einschalten und kontinuierlicher Sensortest
- Helle, weitwinkelige Alarmanzeigen
- Integriertes, stoßfestes Schutzgehäuse

GasAlertQuattro - Spezifikationen

Größe	13,0 x 8,1 x 4,7 cm
Gewicht	- 316 g (mit wiederaufladbarem Akkupack) - 338 g (mit Alkaline-Batteriepack)
Temperatur	-20 bis +50°C
Relative Luftfeuchtigkeit	10% – 100% rF (nicht kondensierend)
Alarne	- Optisch (sechs rote LEDs), Vibration und akustisch (95 dB) - Low (A1), High (A2), STEL, AGW, OL, Bereichsüberschreitung
Tests	Sensor, Stromkreis, Batterie/Akku und akustische/optische Alarne bei Aktivierung, Batterie/Akku (kontinuierlich), Sensor (kontinuierlich)
Pumpe	Kompatibel mit der motorisierten Gasprobenahmepumpe „Sampler“
Batterie/Akku-Lebensdauer	AA-Alkaline-Batterien: 14 Stunden (+20 bis 50°C) Akku: 20 Stunden (+20 bis 50°C) 18 Stunden (-20 bis 0°C)
Anwenderoptionen	Funktionskontroll-Tonignal Drehbares Display Funktionskontroll-Blinksignal Messung brennbarer Gase (% UEG oder Vol. % Methan) STEL-Interval setzen Vom Anwender einstellbare Sensor ein/aus Justergaskonzentration Speralarme Funktionstest aktivieren Sicherer Anzeigemodus Sprachwahl (5) Justierung aktivieren Programmierbare Automatische Einschaltung Nulpunktjustage beim Einschalten Einschaltung Einschalten Datenaufzeichnungsintervall
Schutzklassen	EMV/RF: Erfüllt die EMV-Richtlinie 2004/108/EC IP66/IP67
Zertifizierungen und Zulassungen	Class I, Div. 1, Gr. A, B, C, D Ex ia IIC T4 Ex ia IIIB T4 Ex ia IIIC T4 Ex ia IICT4 Ex ia IIAT4 Ex ia IIAT4
Garantie	Volle 2-Jahres-Garantie einschließlich aller Sensoren

Zusätzliche Merkmale des GasAlertQuattro:

- Stromversorgung über ein austauschbares Akkupack oder ein Alkaline-Batteriepack mit 3 AA-Batterien
- Minimaler Einarbeitungsaufwand dank einfacher Bedienung mit einer Taste und intuitiver Benutzeroberfläche
- Umfangreicher Speicher für Daten- und Ereignisaufzeichnung
- IntelliFlash bestätigt den Betrieb und die einwandfreie Funktion für Bediener und Aufsichtspersonen über eine Distanz von bis zu 6,1 m
- Erhöhte Festigkeit gegenüber häufigen Querempfindlichkeiten durch Industriegase, z. B. Methanol und Ethanol (CO- und H₂S-Sensoren)
- Multilingual: Deutsch, Englisch, Französisch, Spanisch und Portugiesisch
- Felderprobte Surecell-Sensoren bieten eine noch nie dagewesene Leistung selbst unter extremen Einsatzbedingungen.

Zubehör und Extras



Für eine vollständige Zubehörliste wenden Sie sich bitte an BW Technologies by Honeywell.

Sensorspezifikationen

Gas	Messbereich	Auflösung
Schwefelwasserstoff (H₂S)	0 - 200 ppm	0,1 ppm
Kohlenmonoxid (CO)	0 - 1000 ppm	1 ppm
Sauerstoff (O₂)	0 - 30,0%	0,1%
Brennbare Gase (%UEG)	0 - 100 % UEG 0 - 5,0 Vol. %	1 % 0,1 %

Die Alarmeinstellungen für sämtliche Sensoren sind anwendbar einstellbar.
Die gesetzten Einstellungen werden beim Einschalten des Geräts automatisch angezeigt.

Vor Ort erhältlich bei



IM ZUFE KONTINUERLICHER FORSCHUNG UND PRODUKTENTWICKLUNG BEHALTEN WIR UNS VOR, SPEZIFIKATIONEN OHNE VORHERIGE ANKÜNDIGUNG ZU ÄNDERN.

Europäischer Hauptsitz

Life Safety Distribution AG
Javastrasse 2
8604 Hegnau
Switzerland
Tel.: +41 (0) 44.943.4300
Fax: +41 (0) 44.943.4398
www.gasmonitors.com

bwesales@gasmonitors.com

Europa +44 (0) 1295.700.300
Frankreich +33 (0) 442.98.17.70
Deutschland +49 (0) 2137.17.6522
Naher Osten +971.4.4505852
USA 1.888.749.8878

lateinamerika +55.11.3475.1873
Südostasien +65.6580.3468
China +86.10.6786.7305
Australien +61.3.9464.2770
Andere Länder +1.403.248.9226

H_GasAlertQuattro_D801103_V9_31-13_DE
© 2013 Honeywell International Ltd. Alle Rechte vorbehalten.

127/5

Im Rahmen der Arbeit entstandene Forschungsplakate

Mars 1 – Entwicklung eines MR Feuerwehreinsatzhelms

**Jonas Weimar und Joel Koch,
Studiengang Informatik**

Projektziel ist die prototypische Entwicklung und Auswertung eines Mixed Reality Einsatzhelms mit Bezug zur Feuerwehr. Dieser soll in der Lage sein den Einsatzkräften auf einem transparenten Heads Up Display diverse Sensorwerte, eine Tiefenkarte der Umgebung sowie ein Hitzebild zur Verfügung zu stellen. Um den Prototypen zu bewerten und in einen Kontext einzuordnen, sind in Kooperation mit diversen Feuerwehren mehrere Tests mit dem System durchgeführt worden.



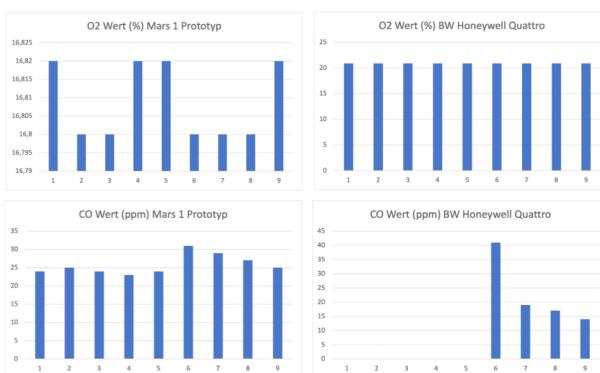
Das Heads Up Display des Mars 1 (l.), der Prototyp (m.) und das Sensormodul auf Raspi 3B+ Basis (r.).

Implementierung des Systems

Das System wurde zweiteilig entwickelt, aufgeteilt in Holo Modul und Sensor Modul. Für das Holo Modul kam die Game Engine Unity zum Einsatz, für das Sensor Modul wurde auf das Grove Sensor Kit, sein Python Framework und einen Raspberry Pi 3B+ zurückgegriffen.

Untersuchung des Systems

Die Untersuchung des Systems ist unterteilt in mehrere Teile: Sensorik, Software und Wärmebildkamera. Die Messwerte über die Sensorik des Systems sind in Tests mit der Flughafen Feuerwehr sowie mit der Freiwilligen Feuerwehr Hattersheim-Okriftel erhoben worden. Dabei konnten für die Werte Kohlenstoffmonoxid und Sauerstoff enorme Abweichungen zwischen Prototyp und Referenzmessung festgestellt werden.



Die mittlere Differenz, ermittelt durch das absolute arithmetische Mittel, liegt bei 4,09% für die Sauerstoff Messungen und bei 15,6ppm für die Kohlenstoffmonoxid Messungen. Ab dem 6. Datenpunkt wurde zur Simulation eines veränderten Luftgemisches der Motor eines Feuerwehreinsatzfahrzeugs gestartet. Die Ergebnisse der CO Messwerte sind besonders ernüchternd, da es sich um ein toxisches Gas und somit einen kritischen Wert handelt.

Die Wärmebildkamera sollte ursprünglich in vollem Umfang auf der Atemschutzsimulationsfläche der Feuerwehr getestet werden. Es kam jedoch zu Komplikationen, so dass als alternative ein Test im Gebäude der DH stattfand. Hierbei funktionierte das Modul ohne Einschränkungen und ist dazu in der Lage gewesen, Personen von der Hintergrundstrahlung zu trennen. Ihre Funktion unter schlechten Verhältnissen konnte nicht überprüft werden.

Bei den Tests hat sich die Erwartung bestätigt, dass ein Software gebundenes an den visible Light Sensor der HoloLens 2 geknüpftes Steuersystem nicht zuverlässig funktioniert, sobald dieser keine Daten mehr aufnehmen kann (bspw. in der Dunkelheit eines Gebäudes ohne Licht).

Ergebnisse

- » Der Untersuchung kann entnommen werden, dass der Mars 1 Prototyp mit der momentanen Sensormodifikation nicht für den aktiven Einsatz bereit ist. Hier wäre vorerst eine Remodifikation mit akkurateren Sensoren von Nöten.
- » Das Display der HoloLens 2 ist zu dunkel, um Hologramme in hellen Umgebungen zu erkennen
- » Das System ist grundsätzlich „sehr beeindruckend“ (Carsten Klebe, Freiwillige Feuerwehr Hattersheim Okriftel, 2022)
- » Physische Steuerung des Systems nötig, um in Gefahrensituationen Steuerung zu ermöglichen.

Ausblick

- » Einbau von KI & Bildverarbeitungssystemen
- » Kartierungssystem fertigstellen & Synchronisation mit Teampositions- sowie Hitzedaten
- » Akkurate Sensoren verbauen
- » Alternative Hardware testen

Kooperative Partner



Quellen

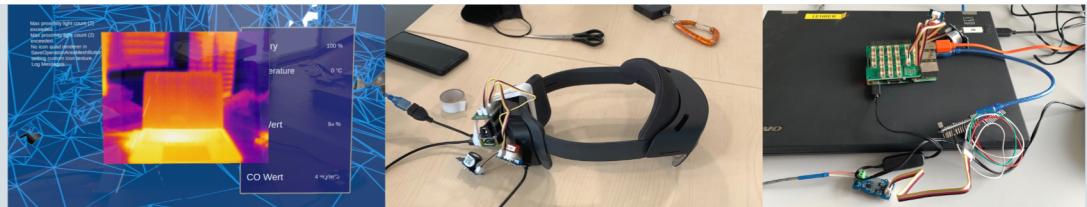
- » Die Durchgeführte Untersuchung des Projektteams
- » Die zu diesem Thema eingereichte Studienarbeit
- » Weitere Quellen können der Studienarbeit entnommen werden

Kontakt

Duale Hochschule Baden-Württemberg

koch.joel@student.dhbw-karlsruhe.de
weimar.jonas@student.dhbw-karlsruhe.de

Modular Augmented Reality System 1



Das Heads Up Display des Mars 1 (l.), der Prototyp (m.) und das Sensormodul auf Raspi 3B+ Basis (r.)

Projektziel ist die prototypische Entwicklung und Auswertung eines Mixed Reality Einsatzhelms mit Bezug zur Feuerwehr gewesen. Dieser soll in der Lage sein den Einsatzkräften auf einem transparenten Heads Up Display diverse Sensorwerte, eine Tiefenkarte der Umgebung sowie ein Hitzebild zur Verfügung zu stellen. Das System wurde zweiteilig entwickelt, aufgeteilt in Holo Modul und Sensor Modul. Für das Holo Modul kam die Game Engine Unity zum Einsatz, unter welcher mit dem Mixed Reality Toolkit eine immersive Software für die Darstellung des HUDs dessen Systeme und dessen Steuerung entwickelt worden ist. Für das Sensor Modul wurde auf das Grove Sensor Kit, sein Python Framework und einen Raspberry Pi 3B+ zurückgegriffen. Unter Einsatz eines Grove High Temperature Sensors, eines MQ9B CO Sensors, eines MIX8410 Sauerstoffsensors und einer Teledyne FLIR Lepton 3.5 Wärmebildkamera werden die Daten des Sensor Moduls aufgezeichnet und dem Holo Modul über eine REST Schnittstelle bereitgestellt. Die Verbindung der Beiden funktioniert über WLAN.

Ein Mixed Reality System zur Performance Augmentation in Feuerwehreinsätzen

Joel Koch, TINF19B5 & Jonas Weimar, TINF19B5

Betreuer: Marcus Strand