Erweiterung von Rufhilfegeräten um Funktionalität von Sprachhilfe als Fallback zu herkömmlichen Rettungsrufsystemen

Kamil Glowinski, BSc
Master Cloud Computing Engineering
Fachhochschule Burgenland
Eisenstadt, Austria,
kamil.glowinski@me.com

Mag. Christian Gossmann, BSc
https://www.gossmann.guru
Headquarter of EC-Computer Knowledge
7000 Eisenstadt, Austria
christian.gossmann@edv-computer.eu

Andreas Krawinkler, BA
Master Cloud Computing Engineering
Fachhochschule Burgenland
Eisenstadt, Austria
krawinkler@gmail.com

Dominik Strümpf, BSc
Master Cloud Computing Engineering
Fachhochschule Burgenland
Eisenstadt, Austria
d.struempf@gmx.at

***Abstract* — Für das Monitoring des menschlichen Gesundheitszustandes haben sich mit aufkommen von IoT Technologien neue Möglichkeiten ergeben. Der Einsatz eben solcher Technologien findet in zivilen und auch militärischen [1] Bereich bereits Anwendung. So bieten Rufhilfegeräte gebrechlichen Personen die Möglichkeit, auf Knopfdruck telefonische Hilfe bei Rettungsdiensten anzufordern. Die Herausforderung dabei ist es, dass diese Geräte permanent am Arm oder als Kette um den Hals mit dem entsprechenden Rufhilfeknopf versehen, getragen werden müssen. Vielfach wird genau dies aus Bequemlichkeit oder Vergesslichkeit verabsäumt und dadurch steht im Ernstfall trotz Rufhilfegerät keine Hilfe zur Verfügung [1]. Die vorliegende Arbeit beschäftigt sich mit der Erweiterung von Rufhilfegeräten durch Cloud-Services und Sprachfunktionalitäten, um im Bedarfsfall automatisch eine Alarmierung der Rettung durchzuführen. Um das Problem zu lösen wird folgende Methodik gewählt: Es wurde eine umfangreiche Literaturrecherche durchgeführt. Hier war die Erkenntnis dass in der einschlägigen Literatur viele technische Spezifikationen bereits existieren, die sich mit Skill-Erweiterungen von Sprachassistenten auseinandersetzen. Diese wurden bis dato aber noch nicht auf Rufhilfegeräte angewandt. Im folgenden Proof of Concept werden genau solche Schnittstellen behandelt, um die Verwendung von Fallsensoren ergänzt und in einem Use Case exemplarisch der utomatische Notruf implementiert.**

I. Einleitung

Die Aktivierung der Rettungskette beginnt mit der Verständigung der Rettung. Dabei kommen Rufhilfegeräte wie in Abbildung 1 dargestellt zum Einsatz. Diese bieten vorwiegend älteren oder gebrechlichen Personen die Möglichkeit in Notsituationen einen Hilferuf abzusetzen [2].

Abbildung 1: Rufhilfgereät- Quelle: https://www.roteskreuz.at [11.12.2019]

Oftmals wird es aber verabsäumt den Alarmierungsknopf am Körper zu tragen, wodurch im Ernstfall Lebensgefahr besteht. Das Voranschreiten der technischen Möglichkeiten unter anderem Cloud-Services und Spracheingabe bieten die Chance eine Alarmierung der Rettung ohne Rufhilfeknopf, einfach durch Zurufes auszulösen. Im folgenden Proof of Concept soll ein Rufhilfegerät um genau jene Funktionalität erweitert und damit ein sinnvoller Beitrag für die Gesellschaft geleistet werden [3]. Natürlich sollte man dieser Stelle festhalten, dass die Spracheingabe als hilfreiches Addon zu verstehen ist, bis dato aber keinesfalls den bewährten Rettungsruf ersetzen kann. Das in weiterer Folge vorgestellte Konzept stellt einen Backup-Kanal dar, der nur im Ernstfall zur Anwendung kommen soll, wenn keine Möglichkeit mehr besteht, den Alarmierungsknopf als primäres Gerät zur Alarmierung zu verwenden, dieser defekt ist oder der klassische Telefonkanal nicht zur Verfügung steht. In diesem Fall wird über das Internet mittels TCP / IP-Protokoll automatisch zu Telefonanbieter einen Notruf abgesetzt.

Forschungsfrage:

Dieses Dokuments beleuchtet die Fragestellung wie ein Notruf getätigt werden kann, wenn das vorhandenen Auslösegerät (Rufhilfeknopf) nicht zur Verfügung steht oder sich nicht in unmittelbarer Reichweite befindet. In weiterer Folge wird die Frage der Implementierung und der Kosten um diese Erweiterung umzusetzen nachgegangen.

II. RELATED WORK

Zu Beginn dieser Arbeit wurde eine umfangreiche Literaturrecherche durchgeführt, bei der das Augenmerk nicht ausschließlich auf wissenschaftlichen Arbeiten, sondern insbesondere auf wissenschaftliche Projekte gelegt wurde. Bei der Literaturrecherche war markant, dass der Begriff Rufhilfesystem ausgehend von der Fragestellung und der Thematik des Forschungszuganges unterschiedliche Ansätze

und davon abgeleitete Ergebnisse ausweist. Einerseits erfolgen Betrachtungen unter dem Aspekt von Prozess- und Qualitätsmanagement, der Akzeptanz, der Systemerweiterung, der umfassenderen Gesamtarchitektur oder der Telemedizin. Auch das Thema Datenschutz und EU DSGVO wird bei Notrufsystemen als ein nicht zu unterschätzender Faktor gesehen. Ausgangspunkt des Papers ist die Auseinandersetzung mit derzeit bestehenden Lösungen von Notrufsystemen. An die in einem Notrufsystem ablaufenden Prozesse wird der Anspruch gestellt, dass einerseits „Sensorfunktionalität" und Reichweite gegeben ist [4] und andererseits Ausfallsicherheit und Zuverlässigkeit [5] sichergestellt werden. Aus den Arbeiten von Barkhold, Frerichs ist zu entnehmen, dass Notrufsysteme nur dann effektiv zum Einsatz gebracht werden können, wenn sie dem Hilfesuchenden das Gefühl der Sicherheit vermitteln [6] und von ihm akzeptiert werden. Rufhilfe-Prozesse und -systeme sollten so entwickelt werden, dass diese auch für Menschen mit „sensorischen, physischen, kognitiven oder sonstigen Beeinträchtigungen" von Nutzen sind [7]. Dies ist in unserem Use-Case gegeben, da geplant ist, ein Abfrageintervall zu implementieren, das je nach gewünschter Zeitspanne bei der betreffenden Person fragt, ob diese noch da ist und zur Bestätigung ein Lebenszeichen erbittet. In diesem Fall würde kein Notruf ausgelöst, andernfalls vorsorglich einer erfolgen. Gleiches gilt für einen Sturzsensor bei dem davon ausgegangen wird, dass sich die Person derart schwer verletzt, dass sie das Bewusstsein verliert und ebenfalls den Notruf nicht mehr selbstständig tätigen kann.

Um diesen Aspekten und auch Ansprüchen Rechnung zu tragen wurde versucht, die bewährte Rufhilfemethode um ein Cloud basierendes Sprachfeature zu erweitern.

III. SYSTEMARCHITEKTUR

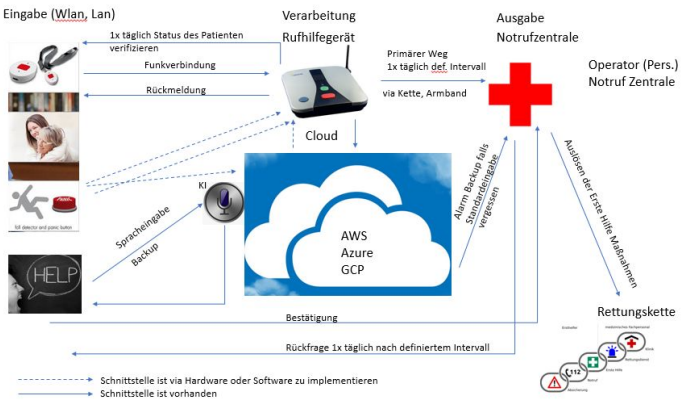

Abbildung 2: Erweiterung bestehender Rufhilfe - Quelle: eigene Skizze

Abbildung 2 visualisiert die Erweiterung des bestehenden Rufhilfe-Konzepts, welches derzeit. beim Österreichischen Roten Kreuz zur Anwendung kommt. Deutlich zu erkennen ist der primäre Notrufkanal mittels klassischem Telefonrufhilfegerät. Der Rufhilfeknopf oder das Rufhilfehalsband kommt immer dann zum Einsatz, wenn eine Notsituation besteht und die Rettung alarmiert werden soll. Als smarte Erweiterung ist in der Darstellung der Fallback-Kanal, welcher mittels Sprachsteuerung z.B. via Smart Speaker und Cloud-Services zum Einsatz kommen könnte, ersichtlich. Dieser Kanal soll daher autark von der primären Telefonleitung funktionieren und über das TCP / IP-Protokoll kommunizieren. Dies kann z.B. eine zusätzliche Telefon- oder Mobilfunkleitung sein, oder könnte auch über Kabel-TV oder Satellitenkommunikation erfolgen. Alternativ könnten auch in unmittelbarer Reichweite befindliche Datenkanäle über WLAN, etc. zur Anwendung kommen. Sollte eine Unterbrechung der Konnektivität zum Cloud Service entstehen [8], eine Internetverbindung jedoch sichergestellt sein, könnte eine Information über ein anderes Service (z.B. E-Mail, SMS) an die Rettungsleitstelle versendet werden. Die Art und Weise wie die TCP / IP-Verbindung sichergestellt wird ist dabei zweitrangig, im behandelten Use-Case wird in regelmäßigen Abständen die Funktionsfähigkeit überprüft, denn entscheidend ist diese erst im Falle eines Sprachrufes. Hier wird der Notruf nicht über den vorgesehenen Knopf ausgelöst, sondern automatisiert durch ein dezidiertes Intervall oder mittels Sprachassistenten, bzw. smarten Eingabegerät. Im Anschluss erfolgen durch die Rettungsleitstelle standardisierte Rückfragen, hier wird abgeklärt ob tatsächlich ein Notfall besteht und eine Hilfestellung der Rettung vor Ort notwendig ist. Die Entsendung der Rettung obliegt letztlich der Entscheidung der Rettungsleitstelle. In der behandelten Fallstudie wird diese Entscheidung durch Sensoren getroffen, sobald durch diese ein Sturz registriert wird oder im definierten Intervall keine Eingabe zum Abbruch des Notrufes durchgeführt wird. Sollte sich nach Rückfrage der Rettungsleitstelle beim Patiente, also nach auslösen des automatischen Notrufs wider Erwarten niemand melden, wird die Rettungskette präventiv in Gang gesetzt. Ist die betroffene Person bewusstlos ist die Alarmierung durch Spracheingabe oder Rufhilfeknopf für das Auslösen der Rettungskette nicht mehr möglich, sondern muss automatisch durch den ISP erfolgen. Hier gilt es nochmals in Erinnerung zu rufen, dass TCP / IP basierende Cloud-Services nicht als primäres Rufhilfesystem zu verstehen sind. Dies wurde vielfach in durchgeführten Tests zu Notrufzwecken diverser Sprachassistenten wie Alexa, Cortana und Siri, untermauert[9]. In Kombination mit einem primären Rufhilfesystem verstehen sich derartige Systeme als Rettungsanker und Backup. Im Appendix findet sich ein auf der Programmiersprache Python basierendes System, das die vorhin aufgezeigten Herausforderungen in einem Proof of Concept exemplarisch verdeutlicht und dessen implementierte Algorithmen bereits über sämtliche Funktionen verfügen. Die konkrete Art der Anbindung in einem endgültigen System, also mittels Sturzsensor in einem Raspberry-Pi, oder auf

einem PC / Laptop mit zusätzlichen Mess-, Regel- und Steuerkarten spielt in diesem Beispiel nur eine untergeordnete Rolle. Gleiches gilt für das Spracheingabesystem mit dem letztlich die Implementierung der Steuerkommandos für die Kommandodatei verwendet wird. Prinzipiell könnte man das System um zusätzliche Sensoren erweitern, z.B. einem Blutdruck- oder Pulssensor, die ebenfalls beim Überschreiten von definierten Grenzwerten automatisch einen Notruf tätigen. Diese müssten lediglich um die IF-Bedingungen, Kommandos und Variablen erweitert werden. Um die Funktionalität in Zusammenarbeit mit dem klassischen Rufhilfesystem gewährleisten zu können, müsste in jedem Fall erst eine entsprechende Schnittstelle hardware- und softwareseitig implementiert werden. Diese sind in der Architekturskizze in Abbildung 2 mittels unterbrochener Linien schematisch dargestellt. Exakt jene Funktionalität schließt die derzeitige Lücke zu dem bereits bestehenden System, da dies unseren Recherchen zufolge auf gleicher Art noch nicht implementiert worden ist. Sollte ein solches System realisiert werden, ist mit hoher Wahrscheinlichkeit davon auszugehen, dass eine Alarmierung in vielen Notsituationen erfolgreich sein wird und daraus eine erhebliche Steigerung der Notrufqualität resultiert.

IV. Implementierung

Bis dato sind Cloud-Systeme kombiniert mit Rufhilfe Sprachassistenzfunktionen noch nicht vorhanden. Für den Proof of Concept wird daher von einem physischen System die Funktion des Fallback-Kanals simuliert. Die Wahl des Cloud Providers und des smarten Sprachassistenten sind dabei nur von nachrangiger Wichtigkeit. Grundsätzlich lassen sich mehrere autarke TCP / IP-Kanäle heranziehen, um nicht nur ein Backupsystem, sondern gleich mehrere Fallback-Systeme zu implementieren, was wiederum in Abstimmung mit dem primären Notrufsystem zu geschehen hat (siehe Abbildung 2). Im Anwendungsfall kommen Python-Skripts auf der Windows Plattform als zentrale Kontrolllogik des Backup-Systems zum Einsatz. Wo die endgültige Implementierung erfolgt, bleibt dem Hardware-Hersteller des betreffenden Gerätes vorbehalten.

Der Fallsensor erfordert Implementierungen mittels Drucksensoren am Fußboden mit entsprechender Verkabelung zum Steuergerät. Im Use-Case wird zur Simulation exemplarisch eine Kommandodatei verwendet, welche das Kommando “Fall” liefert. Die eigentliche Hardware ist hierbei stark von der Wohnsituation abhängig und müsste mittels bautechnischer Maßnahmen umgesetzt werden. Im Use-Case wird eine gestürzte Person simuliert. In einem definierbaren Zeitintervall wird auf eine Eingabe, nämlich den Abbruch des Notrufes durch die Person gewartet. Entdeckt der Steuerungsalgorithmus nach dem Kommando “Fall” keinen Abbruch in einer definierbaren Zeitspanne, geht dieser von einer bewusstlosen Person aus und ein Notruf wird abgesetzt.

Die betreffende Steuerungssoftware selbst kann in einer beliebigen Programmiersprache implementiert werden, wobei die eingesetzte Hardware und deren Spezifikation dafür entscheidend ist. Für einen Raspberry-Pi empfiehlt sich beispielsweise Python, da mit dieser Sprache sehr effizient dessen GPIO-Pins abgefragt werden können und sich auch Eingangssensoren koppeln lassen. Hier wäre es sinnvoll, Sensoren zu normen, da nur so die Steuerung mannigfaltiger Hardware-Modelle und unterschiedlicher Cloud-Services funktionieren kann. Der Aufbau vor Ort beim Anwender erfolgt sowohl bei klassischer Rufhilfe wie auch beim Backup-System durch geschultes Fachpersonal und stellt so keine Schwierigkeit dar. Es ist nämlich davon auszugehen, dass sich gebrechliche, ältere Personen, welche eine große Zielgruppe darstellen, keine Implementierung moderner Cloud-Systeme und Netzwerkinfrastruktur durchführen werden.

In Tabelle 1 sind die einmaligen Kosten für die Anschaffung des Systems sowie die laufenden Kosten im Betrieb dargestellt. Es ist zu beachten, dass je nach Größe der Wohneinheit, welche bestückt werden soll, mehrere Mikrofone sowie Drucksensormatten notwendig sind. Jene Drucksensormatten können beliebig erweitert werden und dabei so konfiguriert werden, dass sobald eine dezidierte zusammenhängende Fläche betätigt wird ein Notruf abgesetzt wird. Ziel ist es zu verhindern, dass durch bloße Berührung ein Fehlalarm ausgelöst wird. Für die Spracherkennung ist in diesem Beispiel ein monatlicher Mittelwert des Rufvolumens von je 500 min. sowohl ausgehend wie auch eingehend angenommen worden. Die Kosten stehen in diesem Beispiel in unmittelbarer Abhängigkeit zur Häufigkeit des in Anspruch genommenen Dienstes.

Einmalkosten		
Raspberry Pi 3 Modell B+, Kit	geizhals.at	€ 60,00/Stk.
Trust MICRO USB Microphone	geizhals.at	€ 20,99/Stk.
Drucksensormatte loadskin Eval Kit 65 x 31 cm	plastic-electronic.com	€ 205,00/Stk.
	Summe Einmalkosten	**€ 285,99/Stk.**
Laufende Kosten		
Programmable Voice	twilio.com	0,0074€/min zum Empfangen
	500 Anrufe/Minuten pro Monat	3,6975€ zum Empfangen
		0,0113€/min zum Anrufen
	500 Anrufe/Minuten pro Monat	5,655€ zum Anrufen
Drei Surf 20GB - 365	drei.at	€ 2,91/M

Tage | 1,7 GB/M.		
	Summe Monatskosten	**12,26 €**

Tabelle 1: Kostenaufstellung - eigene Darstellung

V. Cloud-basierte Applikation

Bezüglich Schnittstellen sei darauf hingewiesen bei virtualisierten Systemen entsprechende Sensoren-Inputs an dieses weiterreichen muss. So fehlen auf einem virtuellen Raspberry-Pi eben die hardwaremäßigen GPIO-Pins. Derzeit gibt es für die verschieden APIs zur Ansteuerung gängiger Sprachassistenten wie Alexa, Cortana und Siri noch keine Norm. Dies wäre jedoch eine Grundvoraussetzung zur Herstellung einheitlicher Notrufsysteme. Da die Hersteller ihre Sprachassistenten bis dato nicht für Notrufe zertifizieren, ist mit einem Standard mittelfristig wohl nicht zu rechnen. Für ein funktionsfähiges Backup-Notrufsystem gilt es daher, das System um eigene Funktionalitäten den sogenannten Skills zu erweitern. Es besteht an dieser Stelle die Notwendigkeit aus allen Anbietern von Sprachassistenten die passenden APIs auszuwählen, diese zu programmieren und auf die betreffende Person zu trainieren. Eine weitere Herausforderung stellen die weltweiten unterschiedlichen Sprachen der Anwender dar und es besteht großer relativ großer Aufwand die Applikation entsprechend anzupassen. Dieses Problem lässt sich für Notrufe auf den Punkt bringen, hier im entscheidenden Fall keine Fehler zu produzieren, wie es derzeit bei Sprachassistenten noch der Fall ist. Eine mögliche Lösung wäre ein Akustiksensor, der lediglich bei überschreiten eines gewissen Lärmpegels von einer Notsituation ausgeht und daraufhin einen Notruf veranlasst. Diese Idee wird im eigentlichen Proof of Concept nicht betrachtet und wäre Teil einer zusätzlichen Implementierung. In Abbildung 3 wird exemplarisch ein Überblick der Fehleranfälligkeit von Sprachsensoren am Beispiel von Google gegeben. Selbstverständlich ist auch bei Rufhilfesystemen darauf hinzuweisen, dass die Qualität der verwendeten Komponenten also Mikrofon und Hardware durchaus relevant sein kann. Konkret kommt es also auch hier letztlich darauf an, ob man teurere oder günstiger Komponenten verbaut, was im Endeffekt eine Budgetfrage darstellt. Entscheidend für den Use-Case und das dortige Python-Skript sind jedoch nicht die verbauten Komponenten, sondern die Darstellung der Funktionalität. Diese ist sichergestellt, da der Algorithmus im Skript von entscheidender Wichtigkeit ist

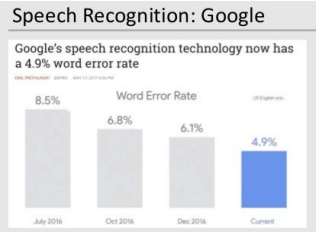

Abbildung 3: Google I/O Keynote, May 2017

VI. Kommerzielle Nutzung

Wird der kommerzielle Nutzen betrachtet, so stellt der Einsatz der verwendeten Technologien wie Internet und Spracheingabe erstmal keine Neuerung dar. Es ist aber deren Kombination welche im betrachteten Use-Case Innovation und einen Kundennutzen generiert. Der monetäre Benefit, sofern überhaupt gegeben, liegt primär in der Betreuung und Wartung der Services, sowie Implementierung der Systeme beim Kunden. Sollten herkömmliche Rufhilfegeräte durch redundante Sprach-KI-Systeme abgelöst werden, amortisieren sich die monatlichen Gebühren für den Betrieb der Geräte. Es bleiben zum jetzigen Zeitpunkt lediglich die monatlichen Fixkosten für die Internetanbindung, welche wie in Abbildung 4 ersichtlich deutlich unter den Kosten für ein Rufhilfegerät liegen.

Abbildung 4: links Kostendarstellung Rotes Kreuz Burgenland
rechts z.B. A1 Festnetz + Mobilfunk Backup - https://www.a1.net/dualpower [09.01.2019]

Zum gegenwärtigen Zeitpunkt und den bereits erörterten Umständen, wird vorerst noch ein clientseitig bestehendes Rufhilfegerät notwendig sein. Es ist anbieterseitig davon auszugehen dass, mit den beschriebenen Funktionserweiterungen von Rufhilfegeräten keine großen Gewinnmargen realisierbar sein werden. Das Gros an Umsätzen kann nur durch Individualisierung und dem Eingehen auf Kundenwünsche generiert werden. Die potentielle Zielgruppe sind primär ältere und gebrechliche Menschen, welchen das nötige Know-how für eine Inbetriebnahme der Devices fehlt. Die angeführte Tabelle illustriert exemplarisch die Marktdurchdringung der Rufhilfegeräte des Roten Kreuzes in Österreich und der dezidierten Regionen. Diese Tabelle kann herangezogen werden, um die potentielle Marktgröße in Österreich abzuleiten.

Rufhilfegeräte des Roten Kreuzes	
Eisenstadt	60
Bezirk Eisenstadt	165

Burgenland	975
Österreich gesamt	41.902

Tabelle 2: Rufhilfegeräte: Stand 2017 - Quelle: Öst. Rotes Kreuz

Auf die Gesamtheit der österreichischen Bevölkerung hochgerechnet, entspricht dies bei 8 Millionen Einwohnern etwa 0,52% der Gesamtheit und stellt einen verschwindend geringen Wert dar. Um einen kommerziellen Vorteil aus dieser Rufhilfe generieren zu können, wäre es notwendig ein Patent für das Backup-Rufhilfesystem anzumelden. Das Rufhilfesystem und das gewählte Projekt hat daher weniger einen kommerziellen Charakter und dient ausschließlich dazu, einen Mehrwert für die Betroffenen zu generieren und den Beweis anzutreten, dass Rufhilfesysteme mit gegebenen Cloud-Services und Technologien mit überschaubarem technischen und monetären Aufwand einfach und effizient adaptiert werden können. Die Höhe einer Technikerstunde zur Implementierung des Services werden in diesem Beispiel mit € 69,-- angenommen [10], d.h. auch hier könnte theoretisch bei einer Anzahl von rund 42.000 Personen einmalig max. € 3.3 Mio. Umsatz generiert werden. Dieser Wert unterliegt naturgemäß einer gewissen Schwankungsbreite zumal über die Dauer der Implementierung und Inbetriebnahme an dieser Stelle nur gemutmaßt werden kann.

VII. Conclusio

Eine Erweiterung bestehender Rufhilfe-Systeme stellt einen wichtigen Aspekt in der Prävention dar und generiert einen sinnvollen Mehrwert durch die Verbindung von bereits vorhandenen Technologien und intelligenten Cloud-Gadgets. Eine Implementierung zum derzeitigen Stand der Technik birgt jedoch noch grundlegende Probleme, zumal die Hardware des primären Rufhilfegeräte noch gar nicht, bzw. über genormte Schnittstellen verfügt. Diesen Gap mit Sprachassistenten zu schließen wäre daher für herkömmliche Rufhilfegeräte von Vorteil, allerdings ist die derzeitige Fehleranfälligkeit von Sprachassistenten nach wie vor eine Einschränkung in Bezug auf die zuverlässige Funktion von Notrufsystemen. Die Nutzungsbedingungen der Hersteller lassen keinen Zweifel offen, dass dieser Umstand bekannt ist. Da diese somit noch keine zuverlässigen Alarmierungsmöglichkeiten anbieten, entsteht hier auch eine gewisse Unsicherheit hinsichtlich der Implementierung und Haftungsfragen, denn genau diese wollen Hersteller ausschließen. Weiter ist die Verfügbarkeit von Cloud Services, welche laut SLA niemals 100% erreichen eine große Einschränkung für ein derartiges Service [11]. Es zeigt sich also, dass, in der Alarmierungskette einige Unsicherheitsfaktoren bestehen. So kann nicht ausgeschlossen werden, dass beim eigenen ISP oder beim präferierten Cloud-Service (Internet zu Telefon oder Sprachassistent) im entscheidenden Bedarfsfall nur eingeschränkte Funktionalitäten der Services auftreten können. Grundsätzlich gibt es diese Einschränkungen auch beim klassischen Telefon und Rufhilfegeräten, allerdings verfügen letztere über ein integriertes Monitoring-System, welches sofort akustisch Alarm gibt, wenn die “Telefonverbindung unterbrochen” ist. Im betrachteten Proof of Concept wird zwar mittels Skripts ebenfalls die Internet-Verbindung geprüft, diese ist aber nicht für die Verfügbarkeit und Funktion der dahintersteckenden Services repräsentativ. Während beim Telefonnetz noch lokale Servicetechniker konsultiert werden können, ist dies für Cloud Services welches unter Umständen auf einem anderen Kontinent betrieben wird schwierig, da dieses nicht im Einflussbereich eines lokalen Technikers vor Ort liegt. Daher ist zusätzlich das Inzident-Management des jeweiligen Cloud Providers zu benachrichtigen.

Um eine annehmbare Reduktion des Risikos erreichen zu können, ist es sinnvoll ein redundantes System unter Zuhilfenahme verschiedener Anbieter zu implementieren. Um eine notfallkritische Redundanz zu erreichen, welche als ernsthafter Ersatz für das primäre Rufhilfegerät angesehen werden kann, besteht die Notwendigkeit auch hier Funktionalitäten der Rufhilfe-Alarmierungsknöpfe zu implementieren. Ein entscheidendes Kriterium ist dabei in jedem Fall eine stabile TCP / IP-Verbindung bei unterschiedlichen Providern und einer zusätzlich redundanten inhouse-Lösung mit z.B. einem geclusterten Switch Paar, sowie USV-Anlagen. Damit begibt man sich allerdings schon auf das Terrain wie man es z.B. von Rechenzentren gewohnt ist. Natürlich stellt sich die Frage der Rentabilität, ob es lohnenswert ist, im Privatbereich Rechenzentrums ähnliche Hardwarestrukturen zu schaffen. Darüber müssen noch weitere Forschungen angestellt werden, v.a. den finanziellen Aspekt betreffend [12]. Da die Wertigkeit eines Menschenlebens [13] jedoch nicht zur Disposition steht, ist die Frage nach der Finanzierbarkeit auf jeden Fall zweitrangig und an dieser Stelle mit einem klaren “im Rahmen des technisch und finanziellen Machbaren” zu beantworten.

VII. LITERATUR

[1] N. Patil, B. Iyer, (2017), Health Monitoring and Tracking System For Soldiers Using Internet of Things (IoT), 2017 International Conference on Computing, Communication and Automation (ICCCA)

[2] U. Albalawi, S. Joshi, (2018), Secure and Trusted Telemedicine in Internet of Things IoT, 2018 IEEE 4th World Forum on Internet of Things (WF-IoT)

[3] Sylke Schemenau, (2000), Intelligente Haussysteme für Alte und Behinderte - eine neue Technologie wird in ihrer gesellschaftlichen Bedeutung "konstruiert", Zugriff am 10.01.2019 unter https://www.ssoar.info/ssoar/bitstream/handle/document/32419/ssoar-zff-2000-3-schemenau-Intelligente_Haussysteme_fur_Alte_und.pdf?sequence=1

[4] Ausfallsicherheit und Zuverlässigkeit“ (Liolios C., et al., 2010)
[5] Barkholdt C., Frerichs F., Hilbert J., Naegele G., & K., 1999
[6] Bruno von Niman und Alejandro Rodriguez-Ascaso User Experience Design Guidelines for TeleCare Services
[7] H.Kersten, G.Klett (2017), Business Continuity und IT-Notfallmanagement: Grundlagen, Methoden und Konzepte, Springer Vieweg, S. 45
[8] Amazon.com, (2018), Amazon Alexa, Zugriff am 7.11.2018 unter https://www.amazon.de/gp/help/customer/display.html?nodeId=201809740
[9] S.Meineke (2017), Sprachassistenten im Ernstfall, Zugriff am 9.10.2018 unter http://www.spiegel.de/netzwelt/gadgets/siri-alexa-cortana-und-google-assistant-im-notfall-test-a-1176863.html
[10] Liwest GmbH (2019), Techniker vor Ort, Zugriff am 7.1.2019 unter https://www.liwest.at/service-hilfe/technische-serviceleistungen/techniker-vor-ort/
[11] Amazon.com, (2018), Amazon Compute Service Level Agreement, Zugriff am 10.1.2019 unter https://aws.amazon.com/de/compute/sla/
[12] G.Purkner, T.Schober, (2015), Kostengenerierung bei Gesundheitssystemen, Zeitschrift für Gesundheitspolitik – Ausgabe 2/2015, S.24
[13] Van Hoof, J., Kort H.S.M., Markopoulos, P., & Soede, M. (2007). Ambient intelligence, ethics and privacy. Gerontech Journal, 6 (3), 155–163.

VIII. ABKÜRZUNGEN

API	Application Programming Interface
BSc	Bachelor of Science
d.h.	das heißt
EC	European Community
etc...	et cetera
GPIO	General Purpose Input Output
IP	Internet Protocol
ISP	Internet Service Provider
KI	Künstliche Intelligenz
Öst.	Österreichisches
PC	Personal Computer
Pi	(Raspberry) Python
TCP	Transmission Control Protocol
TV	Television
v.a.	vor allem

IX. ÜBER DIE VERFASSER

Kamil Glowinski, BSc wurde 2017 sein Bachelor-Abschluss in IT-Infrastruktur-Management an der FH Burgenland verliehen. Derzeit studiert er dort weiter im Masterstudiengang Cloud Computing Engineering. Seit 2011 arbeitet er für die T-Mobile Austria GmbH in vielen technischen Bereichen und sammelte sehr viel Erfahrung im Telekommunikations-Sektor und moderner IT-Infrastruktur. Aktuell ist er im operativen Bereich tätig.

Mag. Gossmann Christian, BSc studierte Kommunikations- und Politikwissenschaften, arbeitete am Zentralen Informatikdienst der Uni Wien und schrieb seine Diplomarbeit zu Digital Rights Management. An der FH Burgenland studierte er IT-Infrastruktur-Management mit Abschlussarbeiten zu Langzeitarchivierung und Künstlicher Intelligenz. Derzeit macht auch er weiter mit dem Master in Cloud Computing.

Andreas Krawinkler, BA wurde sein Bachelor-Abschluss in Business Informatik an der FH Hamburg verliehen. Seine Bachelor-Arbeit beschäftigte sich mit eHealth und Big Data. Er leitet das Datenmanagement eines großen Service-Providers im österreichischen Gesundheitssektor und betreibt derzeit den Masterstudiengang Cloud Computing Engineering.

Dominik Strümpf, BSc ist in Wr. Neustadt, Österreich im Jahr 1985 zur Welt gekommen. 2005 machte er seinen Abschluss an der HTBLuVA Wr. Neustadt in Elektrotechnik und Informationstechnologie. Seinen Bachelor Abschluss in IT Infrastruktur Management an der FH Burgenland erhielt er im Jahr 2017 wo er aktuell den Masterstudiengang Cloud Computing Engineering betreibt. Seit 2010 ist er selbstständiger IT Dienstleister in seiner Heimatortschaft Neudörfl/Leitha.

Appendix

A1. Proof of Concept in Form von Python-Skripts

Um die Funktion zu demonstrieren, bestand das Test-Equipment aus einem Windows 10-64-Bit-Rechner mit installiertem Python in der Version 3.7.2. Python kommt deswegen zum Einsatz, um das Skript einfach auf einen Raspberry Pi portieren zu können, um dessen GPIO-Pins abzufragen. Diese GPIO-Pins dienen ja auf Raspberry-Pi-Systemen als Schnittstelle zu Eingangssensoren für z.B. einen Fall- oder sonstige allfällig zu erweiternde Sensoren wie Puls- und Blutdruckmesser oder Akustiksensoren, die auch bei viel Lärm (Hilfegeschrei) Alarm auslösen. Technisch gesehen wird im Proof of Concept die gesamte Kontrolllogik - unabhängig von der eigentlichen Zielplattform, ob Windows, Raspberry Pi, Android oder iPad / iPhone in Python realisiert. Da das Skript offen für Erweiterungen und auch jegliche Art von Sensoren ist, kann es Blaupause für die eigentliche Implementierung von Notrufsystemen herangezogen werden. Alles was dafür noch getan werden muss, ist, die eigentliche Hardware anzubinden (Sensoren) und eine Wahl für einen gewünschten Sprachassistenten zu treffen, der dann dahingehend trainiert und programmiert werden muss, die Kommando-Datei mit entsprechenden Instruktionen zu befüllen. Den Rest erledigt Python, indem in einer Dauerschleife die verschiedenen Intervalle abgefragt werden, ob denn ein Ereignis vorliegt. Diese Ereignisse sind die Funktion der Internet-Verbindung zu prüfen, ein frei definierbares Intervall abzutesten, nach dessen Verstreichen die jeweilige Person aufgefordert wird, ein Lebenszeichen von sich zu geben sowie die Abfrage des simulierten Fallsensors. Natürlich muss darauf hingewiesen werden, dass von Plattform zu Plattform unterschiedliche Bibliotheken einzubinden sind. Was den Raspberry-Pi anbelangt so findet sich die Vorgehensweise in den Skript-Kommentaren.

Technisch realisiert wurde die eigentliche Abfrage der Kommandodatei ebenfalls in einer frei definierbaren Zeitspanne, ob denn dort tatsächlich ein Kommando vorhanden ist, wie eben z.B. “Fall”, das signalisiert, dass die jeweilige Person nun einmal gestürzt ist. Auch “Rettung”, “Abbruch” oder “alles ok” ist derzeit vorgesehen, wobei die Kommandos selbsterklärend sind. Es wird entweder die Rettung sofort gerufen, der Notruf abgebrochen oder eben keine Aktion durchgeführt, falls alles in Ordnung ist und was auch den Normalfall der Dauerschleife darstellt..

Die Kommandodatei selbst wird exemplarisch noch manuell angelegt. Genau an dieser Stelle können aber die angesprochenen Skills von Sprachassistenten zum Einsatz kommen, um genau diese Datei entsprechend zu generieren und damit für die Auslösung von Notrufen sorgen. Damit ist nicht einmal eine eigene Implementierung der Notruf-Funktionalität im jeweiligen Sprachassistenten erforderlich, sondern nur ein Skill, der im jeweiligen Dateisystem der Zielplattform das entsprechende Kommando in der Kommandodatei ablegt. Auf diese Weise ist höhere Flexibilität gegeben und auch einfachere Wartbarkeit des Skript-Codes.

Für die eigentliche Tätigung des Notrufes wurde der Cloud-Service-Anbieter Twilio.com verwendet, da dieser über umfassende Programmierfunktionen für Internet zu Telekommunikationsanschlusssystemen verfügt. Das Python-Skript macht sich dies zunutze, indem eine vorher hinterlegte und bei Twilio.com registrierte Rufnummer im Notfall angerufen wird. Im Normalfall wird dies die Rettungsleitstelle sein, wobei sich auch diese für Echteinsatz erst einmal bei Twilio registrieren müsste, was hier in Absprache mit betroffenen Personen geschehen könnte und kein technisches, sondern ein Problem des schlichten Wollens ist. Es kann aber grundsätzlich jede beliebige andere Rufnummer registriert werden von Familienmitglieder, Bekannten oder Freunden. Im Proof of Concept wird daher eine eigene Mobilnummer verwendet, die im Notfall angerufen wird und als Beispiel für automatisierte Anwahl einer Zielrufnummer im Notfall dient.

Das System besteht technisch aus zwei Python-Skripts. Aufgabe des ersten Skripts ist die Implementierung der eigentlichen Notrufkontrolllogik, während das zweite Skript die Kommandodatei als Eingabe für das erste zur Verfügung stellt. Der Grund dafür ist, dass Python technisch nicht über eine Eingabefunktion verfügt, ohne das laufende Skript anzuhalten. Für Überwachung von Sensoren, des Internets, eines Kommandos oder zur Abfrage eines Lebenszeichens ist es aber unzulässig deswegen das gesamte Notrufkontrolllogik-Skript anzuhalten. Ist die betreffende Person nämlich bewusstlos und kann keine Eingabe tätigen, würde auch kein Notruf abgesetzt werden, da das Skript “steht” und auf Eingabe wartet..

Genau aus diesem Grund ist auch das Abfrageintervall für die Kommandodatei standardmäßig sehr kurz gehalten (alle 5 Sekunden), was sich aber auch anpassen lässt. Dennoch ist es zu empfehlen, bei den 5 Sekunden zu bleiben, während die anderen Intervalle an die individuellen Bedürfnisse der Person angepasst werden sollten. Sobald das Skript einen Alarmstatus empfängt, wartet es zuerst einmal noch auf einen Abbruch innerhalb der definierten Zeitspanne, widrigenfalls die Rettung alarmiert wird, d.h. also, es wird eine bewusstlose Person angenommen, wenn keinerlei Reaktion vorliegt.

Ersichtlich ist in jedem Fall, mit vergleichsweise wenigen Zeilen Code ein echtes Notrufsystem zu erhalten, das auf Internet- und Cloud-Funktionalität basiert. Wählt man hier

verschiedene Internet-Leitungen bei unterschiedlichen Providern und implementiert andere Cloud-Anbieter zur Notruf-Funktion, so kann durchaus auch von einer Redundanz gesprochen werden, die es mit einem primären Rufhilfegerät aufnehmen kann. Am Ende dieses Appendix ist eine Aufstellung der vergleichsweisen geringen Kosten für mobile Internetanbindung in heutiger Zeit ersichtlich. Damit lässt sich Rufhilfe und Notruf auch fernab von Festnetzleitungen implementieren. Es ließen sich auch je nach Wunsch mehrere Systeme gleichzeitig aufstellen, also im Endeffekt damit Redundanz durch mehrfaches Vorhandensein erzielen.

A2. Setup des Testsystems

Die eigentliche Installation erfordert unter Windows das dort übliche Setup der jeweiligen EXE Datei - für das 64-Bit-System die Python-Datei: python-3.7.2-amd64, downloadbar von https://www.python.org/downloads/release/python-372/ Zur menügeführten Registrierung bei Twilio.com ist eine Mobiltelefonnummer erforderlich zwecks Verifikation, ob man "die jeweilige Person ist". Ebenso war die Wahl der Programmiersprache wie in Abbildung A1 ersichtlich, erforderlich, was eben Python war. Nach der Freischaltung des Ziellandes aus Sicherheitsgründen, um hier nicht durch Hacker und sog. Wardialer Kosten für Anrufe zu verursachen, steht einer skriptgesteuerten Anwahl einer Telefonnummer (Notruf) nichts mehr im Wege, mit Ausnahme, dass man seine individuellen Tokens in das Python-Skript eintragen muss.

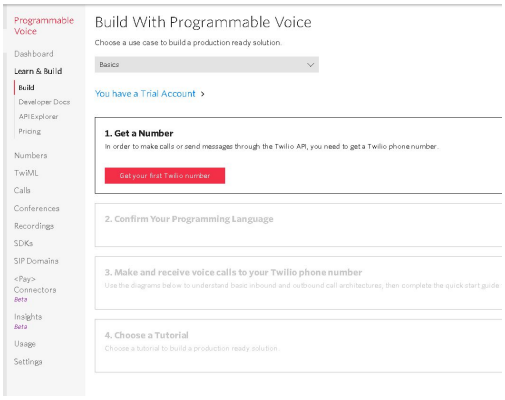

Abbildung A1: Registrierung bei Twilio.com [02.01.2019]

Nach Abschluss des Registrierungsprozesses erhielt man genau diese Authorisierungstokens per E-Mail und ein entsprechendes Code-Snipplet. Dieses wurde bereits in das Beispielskript eingefügt und erfordert nur an der entsprechend kommentierten Stelle im Python-Skript einen individuellen Eintrag. Unsere zu Demozwecken verwendeten Tokens sind nämlich aus naheliegenden datenschutzrechtlichen und account-sicherheitsrelevanten Gründen nicht darin implementiert. Um den Notruf skriptgesteuert durchführen zu können ist für das Zielsystem noch die Twilio-Bibliothek für Python erforderlich, die sich auf der Kommandozeile mittels:

pip install twilio

installieren lässt - Admin-Rechte natürlich vorausgesetzt. Die jeweils aktuellste Version der Notruf-Skripts kann unter https://www.gossmann.at/downloads/Python_Notrufskript_zur_Rufhilfeerweiterung.zip heruntergeladen werden. Für eine mobile Internet-Verbindung fallen exemplarisch folgende in Tabelle A1 genannten Preise an.

Name, Tarif	Netz	GB/M.	Effektivpreis/M.
Drei Surf 20GB - 365 Tage	drei	1,7 GB	€ 2,91 inkl. USt.
Vectone Mobile Datenpaket 1 GB	T-Mobile	1,0 GB	€ 2,94 inkl. USt.
yesss! Classic	A1	1,0 GB	€ 3,58 inkl. USt.
Drei Surf 30GB - 365 Tage	drei	2,5 GB	€ 3,74 inkl. USt.
Vectone Mobile Datenpaket 2 GB	T-Mobile	2,0 GB	€ 3,95 inkl. USt.
Lyca Mobile Daten L	A1	3,0 GB	€ 4,97 inkl. USt.
Vectone Mobile Datenpaket 5 GB	T-Mobile	5,0 GB	€ 4,97 inkl. USt.
Spusu 10GB	drei	10,0 GB	€ 5,90 inkl. USt.
GeOrg surf 5	A1	5,0 GB	€ 6,07 inkl. USt.

Tabelle 2: Datentarif Vergleich [A1]

Zusammenfassend kann gesagt werden, dass jetzt die Hersteller von Hardware gefordert sind, entsprechende Implementierungen von Sensoren vorzunehmen, sowie Softwarehersteller, damit deren Systeme und Sprachassistenten über einheitliche APIs angesprochen werden könnten. Gerade in Hinblick auf ein Notrufsystem wäre es wünschenswert, hier mit einheitlichen Standards arbeiten zu können, denn schon alleine an den erforderlichen Adaptierungen für Raspberry-Pi-Systeme sowie andere Cloud-Anbieter sieht man, dass man von einheitlichen Standards noch sehr weit entfernt ist.

A4. Appendix Literatur

[A1] -Datentarifvergleich, Zugriff am 10.12.2018 unter https://www.tarife.at/telefon-internet/handytarife