Ausarbeitungen zum Seminar

Messen, Steuern, Vernetzen: Überwachen und Manipulieren mit dem Raspberry Pi

Wintersemester 2014/2015

10. Februar 2015

CARL VON OSSIETZKY universität OLDENBURG

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Abt. Systemsoftware und verteilte Systeme

www.uni-oldenburg.de/svs

Inhaltsverzeichnis

Cleveres Beleuchtungsmanagement

Kjel Barjenbruch und Steffen Diekmann

I. Einleitung

SMARTHOME - Ein neumodischer Begriff, der in der heutigen Zeit immer mehr Aufsehen erregt. Wer träumt nicht davon morgens aufzuwachen und in der Küche einen bereits frisch aufgebrühten Kaffee vorzufinden? Das „Smarthome“ bezeichnet ein Haus, das intelligent auf die Bedürfnisse seiner Bewohner reagieren kann. Des Weiteren soll es den Bewohnern Arbeiten abnehmen, welche sich durch automatische Verfahren bewerkstelligen lassen [4, S. 1]. Im Rahmen dieser Ausarbeitung beschäftigen wir uns mit einer spezifischen Arbeit und zwar dem Ein-/Ausschalten des Lichts in bestimmten Räumen, auch Lichtsteuerung genannt.

Jeder kennt die Problematik: Man verlässt das Haus und hat vergessen das Licht auszumachen. Es wäre praktisch wenn das Haus selber dafür sorgen könnte, dass alle Lichter bzw. alle Stromgeräte die nicht benötigt werden ausgeschaltet werden, wenn sich niemand mehr im Haus aufhält. An dieser Stelle greift unser Projekt welches sich mit der Ortung der Bewohner befasst.

II. Was ist eine Lichtsteuerung?

Man bezeichnet Mechanismen bzw. Systeme, welche es ermöglichen Lichter gezielt ein- und auszuschalten als Lichtsteuerung. Die wohl herkömmlichste und weitverbreitetste Realisierung einer Lichtsteuerung im Haushalt basiert auf der Verwendung von manuell zu bedienenden Schaltern oder Relais. Hierbei werden Kabel vom Verteiler an den Schalter und vom Schalter an die Lampe verlegt.

Durch die Betätigung des Schalters wird der Stromkreis geschlossen und die Lampe somit eingeschaltet (Abb. 1) [1, S.62/63]. Wird der Schalter erneut betätigt wird der Stromkreis wieder unterbrochen und die Lampe erlischt. Eine solche Realisierung hat diverse Nachteile. Der wohl offensichtlichste Nachteil ist, dass die Schalter per Hand bedient werden müssen. Will man das Licht ein- oder ausschalten, muss man sich in räumlicher Nähe zum Schalter befinden. Ein weiterer Nachteil ist, dass die Position der Schalter sowie die Verkabelung in der Regel beim Bau des jeweiligen Hauses festgelegt werden und ein späterer Umbau äußerst aufwändig wäre. Dieses gilt insbesondere für Großgebäude, wie Büroanlagen oder ähnlichen Gebäudekomplexen, denn mit der Größe der Gebäude steigt auch die benötigte Menge an Verkabelung und Schaltern. Ferner kommt hinzu, dass man eventuell komplexere Schaltungen realisieren möchte. Beispielsweise Wechselschaltungen (Abb. 2), bei denen das Licht von mehreren im Raum verteilten Schaltern bedient werden kann [5]. Hierbei ist es zusätzlich notwendig die Schalter mit Kabeln untereinander zu verbinden. Dies führt zu einer weiteren Vergrößerung der benötigten Kabelstrecke.

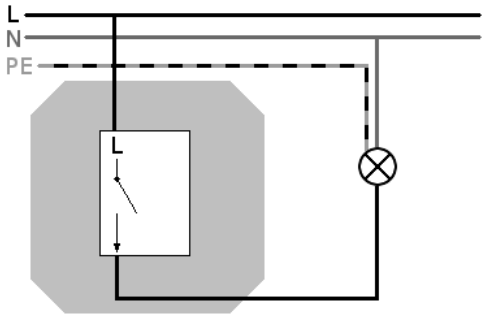


Abbildung 1. Einfache Lichtschaltung: Hier muss ein Kabelstrang vom Verteiler zum Schalter, ein Kabelstrang vom Schalter zur Lampe und ein Kabelstrang von der Lampe zum Verteiler gelegt werden. Ist der Schalter direkt an der Lampe angebracht, ist der Gesamtkabelweg deutlich kürzer.

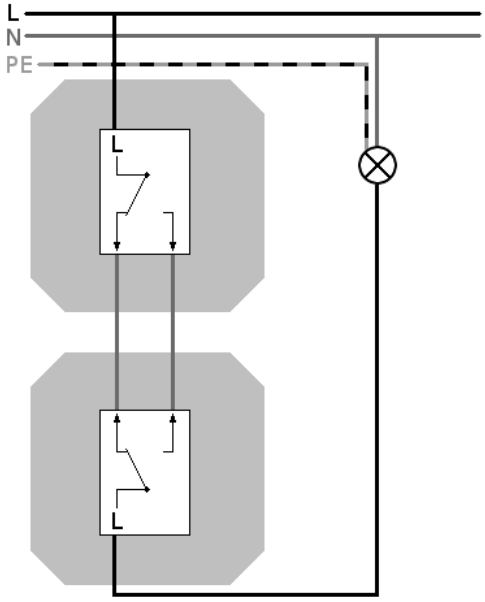


Abbildung 2. Wechselschaltung: Hier muss zusätzlich zu den grundsätzlich notwendigen Kabeln noch ein Kabel vom einen zum anderen Schalter gelegt werden. Bei einer intelligenten Lichtsteuerung wäre solch ein Aufbau nicht notwendig, da der Computer die Logik realisiert. Dies macht die Installation deutlich einfacher.

Es stellt sich die Frage: Gibt es eine elegantere Möglichkeit, die die Nachteile der herkömmlichen Realisierung umgeht? Diese Frage lässt sich mit ja beantworten, es gibt intelligente Lichtsteuerungen die diese Nachteile umgehen. Es gibt allerdings auch einfachere Lösungsansätze, die einige der oben genannten Probleme vermeiden. In einigen Großgebäuden werden die Lampen beispielsweise an Stelle von Schaltern

direkt mit Bewegungs- beziehungsweise Präsenzmeldern verbunden. Diese schalten das Licht automatisch ein oder aus in Abhängigkeit davon, ob sich jemand in ihrem Sensorbereich bewegt beziehungsweise befindet.

III. Intelligente Lichtsteuerung

Intelligente Lichtsteuerung bezeichnet eine computergestützte Lichtsteuerung, die häufig in der Heimautomatisierung zum Einsatz kommt. Der Steuerungscomputer verfügt dabei über Logik, die Geräte mit dem Beleuchtungssystem verbindet oder die die Beleuchtung automatisch steuert. Dadurch sind keine Schalter mehr nötig, da das Licht entweder über entsprechende Geräte oder automatisch ein- und ausgeschaltet wird. Die mit Schaltern verbundenen Probleme entfallen also. Bei einer solchen Lichtsteuerung werden die Lampen z.B. direkt mit Funkschaltern verbunden, welche über den Computer geschaltet werden. Die erste Variante der Verknüpfung aller Lampen über einen Computer mit einem Steuergerät bietet den Vorteil, dass man das Licht von Überall an- und ausschalten kann. Vorausgesetzt man befindet sich mit seinem Gerät in Empfangsreichweite des Computers. Erweitert werden kann ein solches System, in dem man den Computer mit dem Internet verbindet, damit ist es möglich das Licht über das Internet von überall zu bedienen. Stellt man auf der Arbeit fest, dass man vergessen hat, das Licht im Flur auszuschalten, so lässt sich dieses Versäumnis direkt beheben. Stellt man allerdings nicht fest, dass man das Licht angelassen hat, so hilft auch dieses System nicht. Hier bieten sich automatische Systeme an, welche mithilfe von Sensoren ermitteln, ob das Licht aktuell gebraucht wird oder nicht. Allerdings vermisst eine solches System alleinstehend die Möglichkeit, dass man das Licht bewusst ein- und ausschalten kann, unabhängig davon, ob die Logik dieses als sinnvoll erachtet. Die wohl optimale Realisierung wäre eine Verknüpfung eines automatischen Systems mit einer integrierten Steuerung per Gerät. Optimal aus zwei Gründen; zum einen ist es äußerst bequem, zum anderen ist es energieeffizient. Das Licht leuchtet nur bei Bedarf, andernfalls ist es ausgeschaltet und verbraucht keine Energie. Energieeffizienz bietet ökonomische und ökologische Vorteile. Ist der Stromverbrauch geringer, senkt dies die Kosten direkt und auf lange Sicht führt es dazu, dass weniger Energie pro Person verbraucht wird. Der Primärenergieverbrauch in Deutschland ist von 1990 bis 2013 lediglich um 7% gesunken ist, wobei der Anteil an fossilen Energieträgern lediglich um 6,2% gesunken ist (Abb. 3) [2]. Insofern ist jede weitere Senkung des Energieverbrauchs wünschenswert und eine solche Automation ökologisch sinnvoll.

IV. Ausführung

A. Lichtsteuerung per Fernbedienung/ Handy

Bei einer Lichtsteuerung über Fernbedienung oder Handy werden Funk oder WLAN-Schalter in einem Haus angebracht, welche dann mithilfe der Fernbedienung oder des Smartphones gesteuert werden können. Die Installation solcher Systeme ist in der Regel sehr einfach und erfordert keine besonderen Fachkenntnisse. Die Systeme laufen sehr zuverlässig und nehmen sich des Problems der herkömmlichen Realisierung an, dass

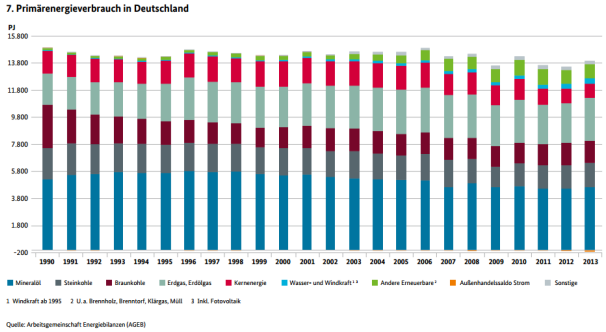


Abbildung 3. Primärenergieverbrauch in Deutschland von 1990 bis 2013

der Akteur sich in der Nähe eines Schalters befinden muss, um das Licht zu bedienen.

B. Lichtsteuerung per Sensorik

Wie bereits erwähnt, gibt es die Möglichkeit die Lichtsteuerung direkt durch Sensoren umzusetzen. Hierbei werden Bewegungs- und/ oder Präsenzmelder mit den entsprechenden Lampen verschaltet, um die Lampe mit einem „intelligenten“ Verhalten auszustatten. Beide Arten von Meldern ziehen ihre Daten aus der Abwärme von Objekten und deren optischen Veränderung bei Bewegungen. Präsenzmelder sind jedoch viel empfindlicher, um jede noch so kleine Bewegung und teilweise auch Änderungen des Lichtverhältnisses zu erfassen. Des Weiteren unterbrechen beide Meldervarianten den Stromkreis, sofern eine bestimmte Zeit lang kein Signal erkannt wird. Wird ein Signal erkannt, wird diese Zeit immer wieder zurückgesetzt.

C. Bestehende Systeme

Natürlich gibt es auf dem Markt bereits Systeme, die mit dem Schlagwort Smarthome werben. Die Telekom bietet ein System an, welches die Möglichkeit zur Kopplung von Sensoren und Fernbedienungen bietet [6]. Die sogenannte Basisstation bildet hierbei die Schnittstelle zwischen den Sensoren, Funkschaltern und der hauseigenen App, die zur Steuerung über das Handy genutzt wird. Die Basisstation kommuniziert mit den Sensoren und Schaltern über Funk und ist zusätzlich selbst am Netzwerk angeschlossen. Das System bietet eine komfortable Möglichkeit Sensoren mit Schaltern zu verbinden, da nahezu keine Fachkenntnisse benötigt werden. Die Erweiterbarkeit des Systems ist ebenfalls gut, da für den Leihen eine Vielzahl von Sensoren zur Auswahl stehen. Die Bedienung über das Smartphone macht das System zusätzlich komfortabel.

Auch RWE bietet ein Smarthome System an, welches dem der Telekom ähnelt [7]. Zusätzlich bietet RWE aber auch Funkschalter und Funksteckdosen an, die in der Wand versenkt werden können und so die Optik des Hauses nicht beeinträchtigen.

Für beide Systeme steht das Energiesparen im Mittelpunkt. Ein Smarthome soll möglichst energieeffizient arbeiten. Hierzu werden diverse Sensoren eingesetzt, die z.B. das Licht nur dann einschalten, wenn eine Person im Raum ist oder die

Heizung einschalten, wenn die Temperatur im Raum unter einem bestimmten Mindestwert fällt. Was die Systeme jedoch nicht bieten, ist eine Möglichkeit das Verhalten der Bewohner zu dokumentieren und zu analysieren.

D. Smarthome mit dem Raspberry Pi

Der Raspberry Pi ist dafür bekannt auf verschiedenen Anwendungsgebieten sehr gute Ergebnisse zu erzielen. Durch die diversen Schnittstellen wie z.B. dem GPIO oder dem USB-Anschluss lassen sich alle gängigen Sensoren ohne Komplikationen mit dem Raspberry Pi verbinden. Auch Funksensoren lassen sich mithilfe eines entsprechenden USB-Empfängers mit dem Raspberry Pi verbinden. Es gibt auch Softwarelösungen, welche einem das Einrichten eines Smarthomes enorm erleichtern. Ein Beispiel wäre die kostenlose Software OpenHab [8]. OpenHab basiert auf Java und ermöglicht eine unkomplizierte Kopplung der gewünschten Sensoren über z.B. das Funkprotokoll FS20. Es bietet die Möglichkeit über eine App bestimmte Funktionen der Software auszulösen, wie beispielsweise die Lichtsteuerung. Weitere Funktionen lassen sich über selbstgeschriebene Java-Methoden realisieren.

Als native Möglichkeit bietet der Raspberry Pi d GPIOs an, welche wir in unserem Projekt genutzt haben. Die GPIO Pins (General Purpose Input/Output) [9, S.31] bietet die Möglichkeit Sensoren, Schalter, Dioden und viele weitere Elektrokomponenten direkt und einfach mit dem Raspberry Pi zu verbinden. Mithilfe entsprechender Breadboards lassen sich damit schnell und einfach kleinere Schaltungen realisieren.

V. Unser Projekt

A. Vorstellung

Die Kernthematik unseres Projektes ist die Ortung von Personen in einem Gebäude und die daran gekoppelte Steuerung spezifischer Schaltanlagen. In unserem Projekt haben wir uns auf die Steuerung des Lichtes beschränkt. Wir nutzen als Logikeinheit einen Raspberry Pi, der sich aufgrund der Größe und der leichten Erweiterbarkeit optimal für unser Projekt eignet. Des Weiteren möchten wir zeigen, was man mit dem Raspberry besser machen kann als mit den bereits etablierten Systemen.

B. Funktionsweise

Für die Entwicklung unserer Lichtsteuerung sind wir von einem Minimalmodell ausgegangen, welches wir zunehmend erweitert haben. In diesem Modell befindet sich lediglich eine einzelne Person im Haus. Das Haus besteht aus zwei Räumen A und B und einer Tür T0. Jeder Raum hat eine Lampe LA und LB. Die Tür ist mit zwei Bewegungssensoren ausgestattet, einem auf jeder Seite der Tür. Es befindet sich also ein Sensor BSA auf der Raum A Seite der Tür und ein Sensor BSB auf der Raum B Seite der Tür. Die Person kann sich nur in einem Raum aufhalten. Wechselt sie in den anderen Raum, dann gilt der vorherige Raum als verlassen. Es war uns wichtig, dass das Licht in einem Raum eingeschaltet wird, bevor man ihn betritt, deshalb ergibt sich dieser Modellaufbau. Der Modellablauf ist wie folgt: Befindet sich die Person in Raum A, leuchtet die

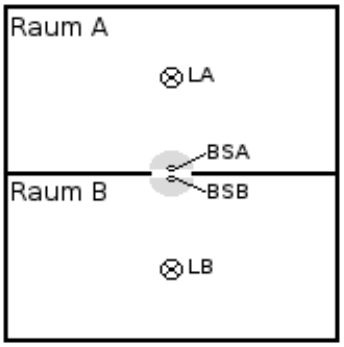


Abbildung 4. Modell 1: Zwei Räume A und B, verbunden durch eine Tür T0. Jeweils ausgestattet mit einer Lampe (LA und LB) und einem Bewegungssensor an der Tür auf der jeweiligen Raumseite (BSA und BSB).

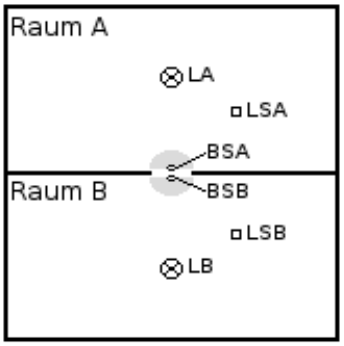


Abbildung 5. Modell 2: Zusätzlich zum Aufbau von Modell 1 verfügt jeder Raum über einen Lichtsensor (LSA und LSB).

Lampe LA. Bewegt sich die Person nun auf die Tür T0 zu und löst den Sensor BSA aus, so wird das Licht LB in Raum B eingeschaltet. Geht die Person innerhalb einer festgelegten Zeit durch die Tür und löst den Sensor BSB aus, so wird das Licht LA in Raum A abgeschaltet. Geht die Person in der entsprechenden Zeit nicht durch die Tür, wird das Licht LB in Raum B wieder ausgeschaltet. Befindet sich die Person in Raum B und nähert sich der Tür T0 gilt der Ablauf analog, die Elemente des Raumes A sind jeweils mit den Gegenstücken des Raumes B getauscht.

Dieses Modell zeigt eine Erweiterung: In Raum A befindet sich ein Lichtsensor LSA und in Raum befindet sich ein Lichtsensor LSB. Das Licht wird in dem Fall nur dann eingeschaltet, wenn der Lichtsensor in dem Raum ausgibt, dass das Licht unzureichend ist. Außerdem haben wir das Modell um eine Tabelle ergänzt, in der gespeichert wird, in welchem Raum sich die Person befindet. Dies ist für die Lichtsteuerung nicht zwingend notwendig, bietet aber eine Vielfalt an Erweiterungsmöglichkeiten.

Das Modell ist atomar und lässt sich somit sehr einfach erweitern. In dem man mehrere Räume auf die gleiche Art und Weise verbindet. Dafür ist es notwendig die durch Türen verbundenen Räume sinnvoll abzuspeichern. Eine Möglichkeit

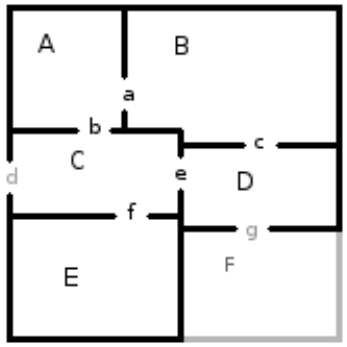


Abbildung 6. Raumplan als Draufsicht: In dieser Darstellung werden die Räume in der Draufsicht dargestellt. In Großbuchstaben sind die Räume abgebildet, in Kleinbuchstaben die Türen. F ist ein Balkon. Tür d und g führen nach draußen (, wo ggf. kein Licht vorhanden ist). Tür d ist die Eingangstür.

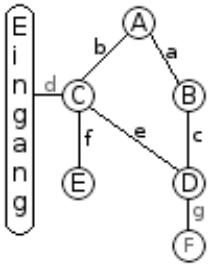


Abbildung 7. Raumplan als Graph: Die Knoten sind die Räume. Die Türen sind die Kanten. Der Eingang kann als zusätzlicher Raum aufgefasst werden. Es fehlt die Information dazu, ob eine Tür nach draußen führt.

	A	B	C	D	E	F	Eingang
A		1	1				
B	1			1			
C	1			1	1		1
D		1	1			1	
E			1				
F				1			

Abbildung 8. Raumplan als Tabelle: Wenn es zwischen zwei Räumen eine Tür gibt ist eine 1 eingetragen, andernfalls ist Nichts eingetragen. Der Eingang ist ein Sonderfall, davon abgesehen ist die Tabelle symmetrisch. Dies kann man sich zu nutzen machen. In dem man die Bewegungssensoren auf eine vergleichbare Tabelle abbildet.

hierzu würden einfache Graphen bieten, deren Knoten die Räume und deren Kanten die Türen wären. Alternativ kann man auch Verknüpfungstabellen oder ähnliche Speichermethoden verwenden.

Der Fakt, dass das Modell lediglich für eine Person ausgelegt ist, macht es für den real Gebrauch nur begrenzt einsetzbar. Allerdings ist das Modell auch in diesem Punkt einfach erweiterbar. Eine Möglichkeit wäre es, mit zusätzlichen Zählvariablen für jeden Raum zu arbeiten und solange das Licht eingeschaltet zu lassen bis alle Personen den Raum verlassen haben, die Zählvariable des Raumes also den Wert Null anzeigt. Die zweite Möglichkeit wäre, dass man zusätzliche Präsenzmelder in den Räumen anbringt, welche

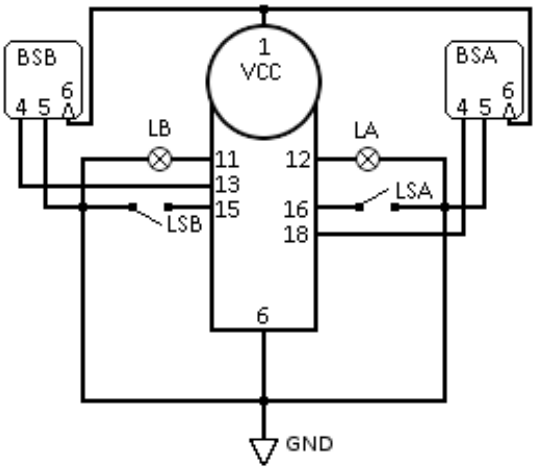


Abbildung 9. Schaltplan unserer Realisierung vom Modell 2: BSA und BSB sind die Bewegungssensoren. LSA und LSB sind die Schalter die als Abbildung der Lichtsensoren dienen. LA und LB sind die Dioden, die als Abbildung der Lampen dienen. Der große Baustein in der Mitte ist der Raspberry Pi. Über dessen Vcc und GND die anderen Elemente verschaltet werden.

die Information geben, ob sich noch jemand im Raum aufhält. Günstiger wäre die erste Methode, wobei die zweite Methode wesentlich fehlertoleranter wäre, sie kann sich beispielsweise nicht „verzählen“. Unser Prototyp entspricht dem Modell 2, die Lichtsensoren werden aus Vorführungsgründen nicht durch Lichtsensoren, sondern durch Schalter realisiert. Wir haben uns für eine Realisierung per GPIO entschieden, weil wir im Modell so besser die Bewegungssensoren ansprechen konnten. Der PI gibt den Versorgungsstrom, welcher für die Sensoren benötigt wird, auf das Breadboard. Und das Breadboard wird mit dem Ground des PI verbunden. Die Sensoren und Schalter sind zusätzlich mit jeweils einem GPIO-Pin des PI verbunden, welcher als Eingang konfiguriert ist. Die LEDs, welche uns in dem Modell als Lampen dienen, sind mit den als Ausgängen konfigurierten GPIO-Pins des PI und dem Ground auf dem Breadboard verbunden.

C. Erweiterbarkeit

In puncto Erweiterbarkeit bietet unser Modell eine entscheidende Funktionalität, die für andere Bereiche der Hausautomatisierung verwendet werden kann. Das Modell ermöglicht die Ortung einer Person im Haus. Im Fall der Lichtsteuerung ermöglicht dieses, das gezielt das Licht des aktuellen Raumes eingeschaltet werden kann. Hinzu kommt, dass der Raspberry Pi weiß, welchen Räumen sich der Nutzer nähert und man so auf den eventuellen Raumwechsel reagieren kann. Eine Heizungssteuerung ließe sich mit den Daten unseres Modells ebenfalls betreiben, würde man festlegen, dass jeder Raum in dem sich mindestens eine Person befindet beheizt werden soll. Man könnte auch einen Log schreiben welcher das Verhalten der Bewohner dokumentiert und dieses später zur Entwicklung eines Algorithmus nutzen, welcher eine vollständige oder zumindest teilweise Automatisierung der Schaltungen im Haus ermöglicht.

Auf technischer Ebene lässt sich unser Modell relativ leicht erweitern, da wir mit dem Raspberry Pi arbeiten haben wir sowohl die Möglichkeit der Verkabelung als auch die Möglichkeit der Nutzung von Funk (z.B. FS20) und WLAN. Es gibt diverse Funkschalter die sich relativ unkompliziert in unser System integrieren lassen können.

Für den Raspberry Pi gibt es ferner eine große Auswahl an Sensoren, welche für eine Erweiterung in Betracht gezogen werden könnten. Lichtschranken würden das Erkennen eines Raumwechsels erleichtern. Lichtsensoren ermöglichen dem System die aktuelle Tageszeit bzw. den Stand der Sonne zu ermitteln, um so eine intelligente Steuerung des Lichts ohne Zeitschaltuhr zu gewährleisten.

D. Kritische Sicht des eigenen Projektes

Im Vergleich zu herkömmlichen Systemen, die auf Fernbedienungen oder eine direkte Verschaltung der Sensoren mit den Lampen bauen, ist unser System mit einer leicht erweiterbaren Logikeinheit ausgestattet. Bei Systemen die auf Fernbedienungen basieren, ist es oft nicht möglich das System automatisch laufen zu lassen, da die entsprechende Sensorik nicht vorhanden ist. Systeme die auf eine direkte Verkabelung von Sensoren und Lampen setzen, sind in der Regel nicht leicht zu erweitern und fordern diverse Fachkenntnisse bei der Installation. Unser System ist relativ leicht zu implementieren und leicht zu erweitern. Leider steht auch unser System vor dem Problem, welches wir bereits am Anfang geschildert haben. Setzt man kabelgebundene Sensoren ein, so muss man die Verkabelung wieder planen und evtl. zusätzliche Stromversorgungen einrichten. Bei einer Realisierung mit Funksensoren muss man beachten, dass die Wände des Gebäudes die Signale nicht zu stark dämmen, da sonst das System nicht zuverlässig arbeiten würde.

Im Bezug auf die Logik hat unser Projekt einen anderen Ansatz als die herkömmlichen Systeme. Unsere Logik ermittelt nicht nur, wo die Person gerade ist, sondern auch wo sie sich hinbewegt. Dies ermöglicht uns theoretisch im Gegensatz zu anderen Systemen bestimmte Verhaltensmuster zu erkennen und diese später mit einer Art Zeitschaltung abzubilden. Für den Anwender sollte eigentlich kein großer Unterschied der Systeme erkennbar sein, denn heutzutage reagieren die Sensoren so schnell, dass auch wenn der Anwender beim Betreten des Raumes den Sensor auslöst, das Licht innerhalb weniger Millisekunden eingeschaltet sein sollte.

E. Fazit

Der Markt bietet für Laien ein durchaus interessantes Angebot für die Umsetzung von Smarthomes. Leider sind diese Systeme immer an den Hersteller gebunden und nicht leicht anzupassen. Man kann zwar aus allen gängigen Sensoren und Schaltern wählen und diese auch bequem mit einer Fernbedienung oder ähnlichem steuern, ist dabei aber immer durch den Hersteller limitiert. Hinzu kommt der Kostenfaktor. Fertige Systeme sind in der Regel teurer als Einzelkomponenten. Nutzt man zur Einrichtung des Smarthomes den Raspberry Pi in Kombination mit einer Software wie OpenHab und unserem Algorithmus, lässt sich bei geringerem Kostenaufwand ein ebenbürtiges, wenn nicht sogar besseres System einrichten. Natürlich wird ein bisschen Spaß am Basteln vorausgesetzt.

LITERATUR

[1] B. Achendorf, *Energiemanagement durch Gebäudeautomation : Grundlagen - Technologien - Anwendungen*, 1. Aufl. Dordrecht, Deutschland: Springer, 2014.

[2] —-, (2015, Jan.), Bundesministerium für Wirtschaft und Energie Redaktion, [Online]. Verfügbar: https://www.bmwi.de/BMWi/Redaktion/PDF/E/energiestatistiken-energiegewinnung-energieverbrauch.pdf

[3] E. F. Engelhardt, *Coole Projekte mit Raspberry Pi*, 3. Aufl. Haar bei München, Deutschland : Franzis, 2013.

[4] J. C. Augusto und C. D. Nugent, *Smart homes can be smarter*, in: J. C. Augustu und C. D. Nugents (eds), *Designing smart home: the role of artifcial intelligence*, 1st ed. Berlin [u.a.] : Springer, 2006.

[5] —-, (2015, Jan.), Elektrische Lichtschaltungen, [Online]. http://www.elektro-klose.at/knowhow/lichtschaltungen.html

[6] —-, (2015, Jan.), Smart Home, [Online]. https://www.smarthome.de/home

[7] —-, (2015, Jan.), Was ist RWE SmartHome?, [Online]. https://www.rwe-smarthome.de/web/cms/de/457156/smarthome/informieren/was-ist-rwe-smarthome/

[8] —-, (2015, Jan.), Was ist RWE SmartHome?, [Online]. http://www.openhab.org/

[9] E. F. Engelhardt, *Hausautomation mit Raspberry Pi*, 1. Aufl. Haar bei München, Deutschland : Franzis, 2014.

Automatische Bewässerung für Zimmerpflanzen

Marius Wybrands
CvO Universität Oldenburg
Deutschland, Oldenburg
marius.wybrands@uni-oldenburg.de

Benjamin Christian Bierbrauer
CvO Universität Oldenburg
Deutschland, Oldenburg
benjamin-christian.bierbrauer@uni-oldenburg.de

***Zusammenfassung*—Das Ziel dieser Arbeit ist es, herauszufinden, ob die Bewässerung einer Zimmerpflanze mithilfe des Mini-Computers, dem Raspberry Pi, automatisch möglich ist. Es ist ein horizontaler Hardware-Prototyp aus Raspberry Pi, verschiedenen Sensoren (Bodenfeuchte, Wasserstand, Temperatur und Lichtintensität), einer Tauchpumpe, einem Wassertank und Hilfskomponenten wie einem Digital-Analog-Wandler und einem Relais entstanden. Der Raspberry Pi dient als Steuereinheit und die Sensoren und Aktoren sind über die GPIO-Schnittstelle angeschlossen. Eine weitere Stromversorgung wird nicht benötigt. Über eine sehr einfache Weboberfläche wird das System gesteuert und überwacht. Der Prototyp zeigt, dass die automatische Pflanzenbewässerung mit dem Raspberry Pi möglich ist. Jedoch ist das eingreifen des Nutzers weiterhin unerlässlich, da jede Pflanze sehr individuelle Bedürfnisse besitzt. Der Prototyp übernimmt eine unterstützende Funktion für den Nutzer, indem er Sensordaten auswertet und visualisiert. Zudem kann die Zimmerpfanze unproblematisch mit Wasser versorgt werden.**

I. Einleitung

In dieser Arbeit geht es um die automatische Bewässerung von Zimmerpflanzen. Mithilfe eines Minicomputers, dem Raspberry Pi, soll eine automatische Pflanzenbewässerung entstehen. Das Ziel ist es einen Prototypen zu entwickeln und mit diesem zu zeigen, dass eine automatische Bewässerung von Zimmerpflanzen möglich ist. Es sind verschiedene Sensoren, Aktoren und Hilfskomponenten in dem Prototyp verbaut.

In dem ersten Teil dieser Arbeit geht es um verwandte Arbeiten und Komponenten, die sich mit der selben Problemstellung befassen. Genauer wird auf das Open Source Projekt OpenSprinkler eingegangen. Im dritten Kapitel geht es um die Bedürfnisse von Zimmerpflanzen. Da der Raspberry Pi als zentrale Steuereinheit eingesetzt wird, ist er im vierten Kapitel beschrieben. Im Detail ist die GPIO-Schnittstelle beschrieben, da sie die Schnittstelle für die Sensoren, Aktoren und die Stromversorgung ist.

Das fünfte Kapitel befasst sich mit dem entwickelten Prototypen und den eingesetzten Komponenten. Die verschiedenen Sensoren (Bodenfeuchte, Wasserstands-Alarm, Temperatur und Licht), die Tauchpumpe als Aktor und die eingesetzten Hilfskomponenten wie der Digitale-Analog-Wandler MCP3008 und das Relais zur Steuerung der Tauchpumpe sind im Detail beschrieben. Diskutiert wird, wieso der Raspberry Pi für die automatische Pflanzenbewässerung eingesetzt wird und ob es Alternativen gibt.

Der letzte Abschnitt dieser Arbeit befasst sich mit den aufgetretenen Problemen und zeigt die Grenzen des Prototypen auf.

II. Verwandte Arbeiten

Schon viele Menschen haben sich Gedanken gemacht und der automatisierten Pflanzenbewässerung einen Namen gegeben. Ob nun „Blumenwächter“ [1], „Windowfarm“ [2], „Parrot Flower Power“ [3], „Mikrocontrollergesteuerter Blumengießautomat“. Diese Projekte überwachen und kontrollieren die Bedürfnisse von Pflanzen. Je nach Gerät wird auf andere Bedürfnisse eingegangen, bzw. anders mit ihnen umgegangen.

Der sogenannte Blumenwächter beobachtet mit Sensoren die Erde und misst, welche Nährstoffe fehlen um dem Benutzer mitzuteilen, dass diese fehlen. Er bietet an über WiFi auf einer „App“ die Möglichkeit auf den Blumenwächter zuzugreifen und verschiedene Daten abzufragen, beispielsweise Temperatur, Feuchtigkeit der Erde und Lichteinstrahlung, auch um welche Pflanze es sich handelt kann dort abgelesen werden. Der Blumenwächter kann auch den Weg zum nächsten Laden weisen, wenn einem der Dünger ausgeht. Aber auch, wenn er alle Daten sammelt, so bleibt dem Benutzer überlassen die Pflanze zu gießen, zu düngen, und an einen anderen Platz zu stellen [1]. Auch der „Parrot Flower Power“ überwacht die Bedürfnisse der Pflanze. Der Unterschied zum Blumenwächter ist die Zugriffsmethode, anstatt mit WiFi wird mit Bluetooth auf das Gerät zugegriffen [3].

Die Firma Windowfarms bietet ein kompletten Aufbau, einmal das Gerüst und die Pflanzen samt Nährstoffe. Auch der Kauf teilt sich in zwei Bereiche auf. Zum einen entscheidet man sich für die Anzahl der Reihen, die man kaufen möchte, zum anderen für ein Pflanzenpaket. Der Aufbau wird vom Käufer getätigt. Monatlich werden neue Pflanzenpakete zugeschickt, welche in der Windowfarm gedeihen können. Bewässert werden die Pflanzen kontinuierlich per Zeitschaltuhr, anstatt nach Bedürfnisse [2].

Diese Projekte befassen sich mit einem Problem und können nicht anders eingesetzt werden, diese Lösungen bezeichnet man als Insellösungen.

A. OpenSprinkler

Ein weiteres Projekt mit dem Namen „OpenSprinkler“ [4] bewässert Pflanzen mit ferngesteuerten Sprinklern. Dieses Projekt ähnelt diesem Projekt am Meisten. Es ist OpenSource und es werden mehrere Sensoren bzw. Aktoren von einer zentralen Steuereinheit überwacht und kontrolliert. Diese zentrale Steuereinheit steuert angeschlossene Sprinkler-Ventile. Das Gerät kann mit dem Internet verbunden werden und bietet verschiedene Programmiermöglichkeiten. Die zentrale Steuereinheit kann mit sogenannten „Extensions“ erweitert

werden und bietet so die Möglichkeit mehr als 8 Sprinkler-Ventile, so viele unterstützt die Grundeinheit, anzuschließen. Wenn das Gerät mit dem Internet verbunden ist, bietet es die Möglichkeit, von zu Hause oder der Arbeit aus zuzugreifen. Außerdem kann es Daten aus dem Internet zum Thema Wetter beziehen um die Sprinkler richtig zu schalten. Es gibt nicht nur eine feste Grundeinheit, es werden mehrere Steuereinheiten angeboten, unter anderem auch der Raspberry Pi als Kontrollmodul.

B. rPi vs OpenSprinkler

Dieses Projekt ist ein Mikrocontrollergesteuerter Blumengießautomat, es wird durch einen Mikrocontroller, dem Raspberry Pi, gesteuert und gießt automatisch eine Pflanze, wenn sie es benötigt. Die Idee ist nicht neu, wie oben beschrieben. Dennoch soll dieses Projekt andere zum Nachbauen anregen. Der Raspberry Pi ist zudem für 35 Dollar ein günstiges Bauteil. Die Software wird gratis zum Download angeboten, und erweiternde Komponenten haben sich an den Preis des Raspberry Pi angepasst und sind vergleichsweise günstig.

In erster Linie unterscheiden sich beide Projekte dadurch, dass der OpenSprinkler sich auf die Bewässerung von Parks, Gärten und Feldern spezialisiert, während dieses Projekt einzig und allein eine Zimmerpflanze gießt. Auch geht der OpenSprinkler nicht auf die Bedürfnisse der Pflanzen ein. Der OpenSprinkler bietet nur die Möglichkeit, bei Regen die Ventile nicht anzusprechen und bei viel Sonne häufiger die Ventile anzusprechen. Wenn es mit dem Internet verbunden wurde und somit die Daten auslesen kann.

III. Bedürfnisse von Zimmerpflanzen

Bei der Pflege von Zimmerpflanzen sind verschiedene Faktoren zu beachten: Jede Pflanzenart hat spezielle Ansprüche an Licht und Schatten, Erde in der die Zimmerpflanze eingepflanzt ist, Temperatur, Luftfeuchtigkeit, Wasserzugabe, Nährstoffen und Temperaturschwankungen. Die Faktoren beeinflussen sich dabei gegenseitig. Beispielsweise ist die Wasserzugabe abhängig von der Lufttemperatur und der Größe des Blumentopfes. Genau definierte „Norm[en] und Regeln“ [5, S. 147] zur Pflege von Pflanzen existieren nicht. Jedoch gibt es Empfehlungen und Richtlinien, welche die Pflege vereinfachen. Beispielsweise vertragen die Pflanzen der Gattung Agave volle Sonneneinstrahlung, brauchen eine Mindesttemperatur von 11 Grad Celsius und sollten mäßig gegossen werden [6, S. 98]. Nach Jacobi [5, S. 147] und Mart [6, S. 61] sind neben den Empfehlungen und Richtlinien die eigene Erfahrung, Beobachtungen und der „grüne Daumen“ ebenfalls ausschlaggebend für die erfolgreiche Pflege von Zimmerpflanzen.

IV. Raspberry Pi

Der Raspberry Pi ist ein Kredikartengroßer Minicomputer mit 512 MB RAM. Er hat einen 700 MHz ARM11-Prozessor, 2 USB Anschlüsse und einen Ethernet-Controller.. Diese Angaben beziehen sich auf den Raspberry Pi Version B. Der Raspberry Pi hat einen SD-Karten-Slot und die GPIOs. Ein Betriebssystem kann über eine SD-Karte oder einen USB-Stick gelesen werden [7, S. 1].

Die Raspberry Pi Foundation bietet auf der offiziellen Website mehrere Linux-Distributionen zum Download an, unter anderem Raspbmc, Risc OS, Openelec und Pidora. Außerdem stellen sie die empfohlene Linux-Distribution Raspian zur Verfügung, welche ein Ableger von Debian ist (Raspberry trifft Debian). Auch ein Paket aus allen wird angeboten. Dieses Paket wurde „New Out Of the Box Software“ (kurz „NOOBS“) getauft und enthält seit kurzem nur noch das Betriebssystem Raspian. Nur mit einem Internetanschluss können die anderen Distributionen installiert werden [8]. „The offline NOOBS package now only contains the Raspbian archive. To install Arch, Pidora, OpenELEC, RaspBMC or RISC OS you will require a network connection” [9].

A. Der Raspberry Pi als Forschungsplattform

Da viele Studenten keine Programmierkenntnisse in das Studium mitbrachten und es nicht genug Computer gab um allen das Programmieren zu ermöglichen, rief die Universität Cambridge das Projekt eines Kostengünstigen Mini-Computers ins Leben. Jetzt ist der Raspberry Pi eine Lernplattform für jedermann [10, S. 1].

Dabei ist nicht nur die von vornherein eingebundene Programmiersprache Python die einzige Programmiersprache, die genutzt werden kann, sondern auch andere Programmiersprachen sind möglich. Wie bei Linux üblich, kann GCC, die GNU Compiler Collection, installiert werden und damit die präferierte Programmiersprache kompiliert werden.

Eine große Community unterstützt die Foundation und zeigt Menschen mit Begeisterung zum selber Bauen, mittlerweile viele, Anleitungen und gibt Hilfestellungen bei Problemen aller Art. Nicht nur Software erweitert den Raspberry Pi, auch Hardware kann an die vorhandenen GPIOs angeschlossen werden, genaueres zu GPIOs später. Diese Hardware erweitert dann den Raspberry Pi zum Beispiel um eine Kamera, einen Sensor oder Aktor, oder mehrere Sensoren und Aktoren. Diese können dann Manuell mit dem Pi oder automatisch durch selbstgeschriebene Programme gesteuert werden [8].

B. GPIOs

Die GPIOs (General Purpose Input/Ouptput) des Raspberry Pi sind zur Erweiterung der Grundfunktionen des Raspberry Pi gedacht. Es können Displays, Touchscreens, Kameras oder andere externe Geräte angeschlossen werden. Dazu stellt die Version B des Raspberry Pi 26 Pins bereit. Von den 26 Pins sind 17 frei programmierbar. Und die anderen 9 übernehmen Systemfunktionen wie Spannung bereitstellen und als Masse dienen. [7, S. 157] Die GPIOs senden ausschließlich Digital. Wie der Name schon sagt können die GPIOs als Eingangs als auch als Ausgangssignal gelten. Entweder ein Sensor sendet Daten über die GPIOs an den Raspberry Pi, oder der Raspberry Pi schickt ein Signal an einen Aktor. So ist es möglich auch komplexe Schaltungen zu realisieren oder beispielsweise Touchscreens anzuschließen.

Zum Steuern der Pins ist keine besondere Software nötig, über das Terminal und den Zugriff auf „/sys/c1ass/gpio“ mit root-Rechten kann man den Pins ein LOW oder High zuweisen. So sind einfache Schaltkreise zu realisieren. Als Beispiel sei gewählt eine LED mit einem Vorwiderstand an die Pins 12 als Datensignal, und Spannungsgeber, und Pin 25 als Masse, um den Stromkreis zu schließen, anzuschließen. Die LED kann nun mit den richtigen Befehlen ein- und ausgeschaltet werden.

MyIO	GPIO	WiringPi		WiringPi	GPIO	MyIO
–	–	–	3V3 Power / 5V Power	–	–	–
p2	–	8	GPIO 0 (SDA) / 5V Power	–	–	–
p3	–	9	GPIO 1 (SCL) / Ground	–	–	–
p4	4	7	GPIO 4 (GPCLK0) / GPIO 14 (TXD)	15	14	p23
–	–	–	Ground / GPIO 15 (RXD)	16	15	p22
p6	17	0	GPIO 17 / GPIO 18 (PCM_CLK)	1	18	p21
p7	–	2	GPIO 21 (PCM_DOUT) / Ground	–	–	–
p8	22	3	GPIO 22 / GPIO 23	4	23	p19
–	–	–	3V3 Power / GPIO 24	5	24	p18
p10	10	12	GPIO 10 (MOSI) / Ground	–	–	p17
p11	9	13	GPIO 9 (MISO) / GPIO 25	6	25	p16
p12	11	14	GPIO 11 (SCKL) / GPIO 8 (CE0)	10	8	p15
–	–	–	Ground / GPIO 7 (CE1)	11	7	p14
MyIO	GPIO	WiringPi		WiringPi	GPIO	MyIO

Abbildung 1: Belegung von WiringPi, BCM(GPIO) und Raspberry Pi.

Erst wird der Pin 18 exportiert um zur Verfügung zu stehen. Danach gibt man dem Pin zu verstehen, dass 0 für 0V Spannung steht. Im Anschluss wird der Pin 18 als Ausgang definiert und kann dann mit einer 1 an und einer 0 ausgeschaltet werden. Dies schaltet die LED an bzw. aus. Der einzugeben Code in der Linuxshell des Raspian-Betriebsystems ist wie folgt [10, S. 175] [10, S. 176]:

```
1 echo 18 > /sys/class/gpio/export
2 echo 0 > /sys/class/gpio/gpio18/active_low
3 echo out > /sys/class/gpio/gpio18/direction
4 echo 1 > /sys/class/gpio/gpio18/value
5 echo 0 > /sys/class/gpio/gpio18/value
```

Um das zu automatisieren muss Programmcode verwendet werden, welcher die Möglichkeit bietet die Pins zu steuern. Dafür wurden Bibliotheken implementiert, welche man zur Steuerung nutzen kann. Eine Programm-Bibliothek wäre WiringPi. WiringPi wurde in C geschrieben und muss, um in beispielsweise Python Verwendung zu finden „gewrappt“ werden. Es arbeitet nach einem eigenen Belegungsplan, kann aber auch mit der BCM-Pinbelegung als auch mit der originalen Pinbelegung arbeiten.

Eine weitere Bibliothek, welche in Python geschrieben wurde, ist RPI.GPIO. Bei der Nutzung von RPI.GPIO muss zu Beginn ein Belegungsplan gewählt werden, BCM oder Pi. Hat man eine der Bibliotheken eingebunden so können die Pins per Programmcode gesteuert werden. [10, S. 173]. Abbildung 1 zeigt die Belegung

V. Prototyp: Automatische Bewässerung für Zimmerpflanzen

Abbildung 2 zeigt den Ablauf einer Bewässerung: Um eine Pflanze, die in Erde eingegraben ist, zu bewässern, muss beobachtet werden, ob die Erde trocken oder feucht ist. Die Konzentration des Wassers in der Erde gibt Auskunft über den Versorgungszustand der Zimmerpflanze. Dieses wird normalerweise durch einen Feuchtigkeitstest mit dem Finger ermittelt. Da ein Raspberry Pi nicht die Möglichkeit hat, wie ein Mensch zu fühlen, ob die Erde trocken oder feucht ist, benötigt er dafür einen Sensor. Dieser gibt an, wie feucht die Erde ist. Ein Schritt der intuitiv durch den Menschen realisiert wird, ist die Überprüfung, ob genügend Wasser in dem Wasserbehälter ist und die Wässerung der Pflanze. Mit einem Raspberry Pi muss dieser Vorgang durch einen Aktor gelöst werden. Mithilfe einer Wasserpumpe, die das Wasser aus dem Behälter auf die Erde pumpt wird das Wasser aus dem Behälter zur Pflanze befördert. Ein zweiter Sensor in dem Wasserbehälter „überwacht“ den Wasserstand. Solange die Erde nicht genügend feucht ist bleibt die Pumpe aktiv. Der Endzustand ist erreicht, wenn die Feuchtigkeit der Erde ausreichend ist. In Abbildung 2 ist das Befüllen des Wasserbehälters durch den Nutzer realisiert. Dieses ist auch in dem Prototyp in dieser Form realisiert.

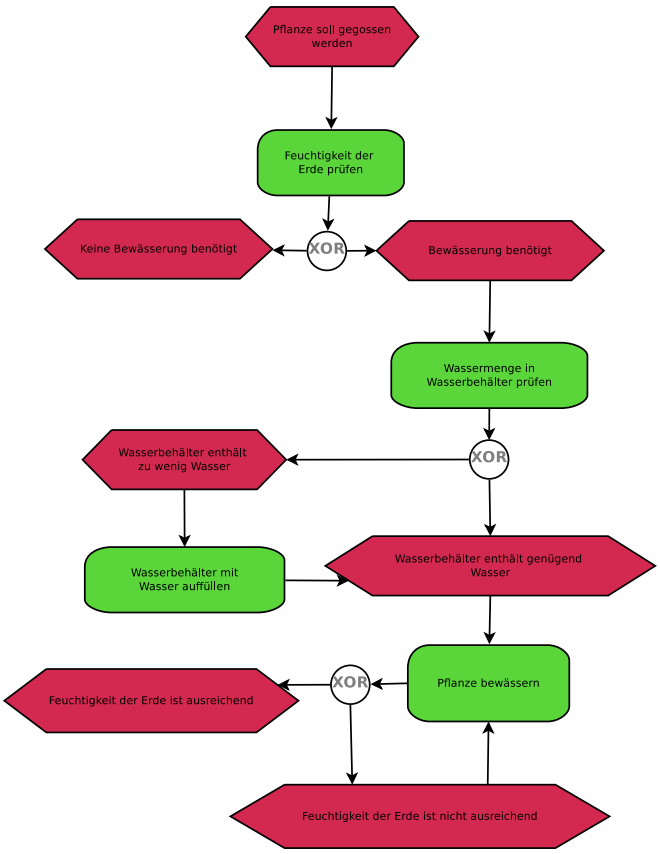


Abbildung 2: Ereignisgesteuerte Prozesskette des Bewässerungsvorgangs von Zimmerpflanzen.

Der Grad der Automation kann an dem Diagramm veranschaulicht werden: Je mehr Ereignisse der Prototyp übernimmt, desto automatisierter ist die Pflanzenbewässerung. Somit ist der Begriff „automatisch“, in dieser Ausarbeitung, als die Ausübung von einzelnen Arbeitsschritten zur Bewässerung von Zimmerpflanzen durch den Prototypen definiert.

Da der Prototyp aus verschiedenen Bauteilen besteht, wur-

den zuerst einzelne Bauteile seperat getestet und auf ihre Funktionsfähigkeit untersucht. Mithilfe des ersten Prototypen konnte der Nutzer durch betätigen eines Tasters die Tauchpumpe ansteuern. Diese hat das Wasser aus dem Wasserbehälter zur Pflanze befördert. Es lag somit eine erste Automation vor. Dieses sollte jedoch nicht das endgültige Ziel des Projektes sein. Durch das Auslesen von Sensordaten ist nicht nur das „Pflanze bewässern“-Ereignis automatisiert worden, sondern auch die Feuchtigkeitsprüfung der Erde sowie die Prüfung der Wassermenge in dem Wasserbehälter. Im folgenden wird der aktuelle Stand des Prototypen beschrieben.

A. *Komponenten des Prototypen*

Die primäre Komponente bei dem Prototyp ist ein Raspberry Pi Model B mit einer 8 GB SD Karte und einem W-Lan-Adapter von LogiLink. Als Stromversorgung dient ein 2000 mA Netzteil. Das Betriebssystem ist die Raspbian Version vom 24.12.2014. Als ergänzende Pakete sind ein Apache Webserver mit PHP5- und Python-Erweiterung für die Web-Anwendung, sowie das Open Source Projekt WiringPi, welches ein einfaches Framework für den Zugriff auf die GPIO-Schnittstelle darstellt, installiert.

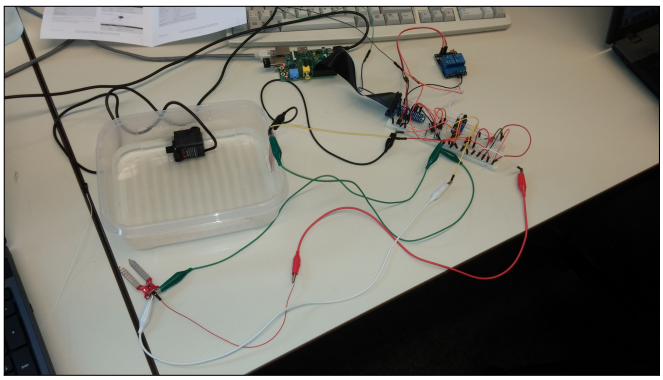


Abbildung 3: Aufbau des Prototypen nach Abbildung 4 ohne Pflanze und Wasserschlauch.

Über die GPIO-Schnittstelle sind verschiedene Sensoren und Aktoren angeschlossen: In Abbildung 4 ist ein Schaltplan abgebildet, der den Aufbau des Prototypen zeigt. Der Bodenfeuchte- und Wasserstandssensor von der Firma Funduino sowie der Lichtsensor liefern, wenn sie mit der ausreichenden Spannung versorgt sind, ein analoges Signal. Da der Raspberry Pi nur digitale Signale auswerten kann, wird der Analog-Digital-Wandler MCP3008 (Abbildung 7b) eingesetzt. Er wandelt das analoge Signal mit einer Auflösung von 10-Bit in ein digitales Signal um. Als vierter Sensor ist der digitale Halbleitertemperatursensor TO92-Housing über den GPIO-Pin 4 angeschlossen. Es ist der Standard Pin für den 1 Wire Bus des Raspberry Pi. Als Aktoren sind eine Tauchpumpe mit einer Versorgungsspannung von 3,5 - 12 V und einer Stromstärke von 100-400 mA und eine rote LED eingesetzt. Da die Tauchpumpe mit einer festen Spannung von 5 V betrieben werden muss, wird ein Auslöserelaismodul eingesetzt. Somit ist es möglich mithilfe eines steuerbaren GPIO-Pins das Relais zu schalten und eine 5 V Spannung an die Tauchpumpe anzulegen. Es wird keine externe Stromquelle genutzt, da alle

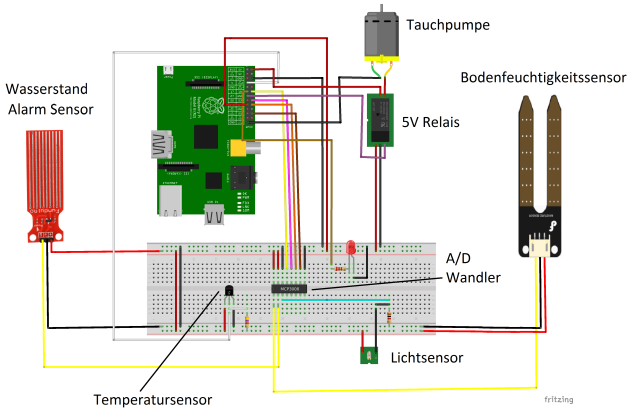


Abbildung 4: Schaltplan des Prototypen für die automatische Bewässerung von Zimmerpflanzen.

Komponenten über die GPIO-Schnittstelle mit einer Spannung von 3,3 V und 5 V betrieben werden können.

In Abbildung 3 ist der aktuelle Stand des Prototypen zu sehen. Es ist ein horizontaler Prototyp, der das Zusammenspiel der verschiedenen Komponenten zeigt. Horizontal bedeutet, dass alle Sensoren, Aktoren und eine Verbindung zur Web-Applikation implementiert sind, jedoch die Funktionalität wie z. B. die Aggregation der Daten zur Berechnung der optimalen Wassermenge für die Zimmerpflanze fehlt. Ebenso ist die Weboberfläche sehr einfach gestaltet.

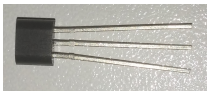
B. *Sensoren und Aktoren*

In diesem Abschnitt wird genauer auf die verschiedenen Sensoren und Aktoren eingegangen. Die Anode und Kathode des Pflanzensensors (Abbildung 6) sind von Pflanzenerde umgeben und der Stromkreis ist über die Pflanzenerde geschlossen. Je feuchter die Pflanzenerde ist, desto geringer ist der Widerstand zwischen Anode und Kathode. Dieses führt zu einer höheren Spannung, die am Signalausgang des Sensors gemessen werden kann. Hat der Sensor kein Medium zwischen Anode und Kathode ist die gemessene Spannung 0 V. Ist zwischen der Kathode und Anode ausschließlich Leitungswasser liefert der Sensor einen Spannungswert von 2,3 V. Dieses sind die Extremwerte zwischen denen die Feuchtigkeit der Erde liegen kann. Der Wasserstands-Alarmsensor (Abbildung 5c) ist in dem Wassertank an einer Seitenwand angebracht und liefert einen Wert von 0 V, wenn er keinen Kontakt zum Wasser hat. Er liefert einen maximalen Wert von 2,3 V wenn die Kontaktfläche des Sensors vollständig im Wasser eingetaucht ist. Der Temperatursensor (Abbildung 5d) lässt sich zu diesem Zeitpunkt noch nicht auslesen, da er nicht vom Raspberry Pi als Knoten auf dem 1 Wire Bus erkannt wird. Ebenso ist der bestellte Lichtsensor (Abbildung 5b) nicht für die Anforderungen an das Projekt geeignet. Er gibt nicht, wie erwartet, den aktuellen Helligkeitswert als analoges Signal aus, sondern reagiert auf Lichtveränderungen mit einem abfallen des Signals auf 0V und einer anschließenden linearen Steigerung bis zu einer maximalen Spannung von 2,3 V. Dieses ist jedoch für die automatische Bewässerung von Zimmerpflanzen geeignet. Die Tauchpumpe (Abbildung 7c) befindet sich,

(a) Bodenfeuchtesensor

(b) Licht-veränderungssensor

(c) Wasserstands-Alarmsensor



(d) Temperatursensor

Abbildung 5: Sensoren des Prototypen.

wie der Wasserwasserstands-Alarmsensor, in dem Wassertank. Durch einen Plastikschlauch mit einem Innendurchmesser von 6 mm, ist er an der Tauchpumpe angeschlossen und dient als Wasserzuleitung für die Versorgung der Zimmerpflanze mit Wasser. Es wurden verschiedene Innendurchmesser getestet. Als Ergebnis ist festzuhalten, dass je dünner der Schlauch ist, desto mehr Zeit benötigt die Pumpe um einen Liter Wasser zu befördern. Bei dem Innendurchmesser von 6 mm dauerte es

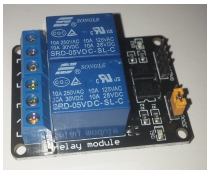
Abbildung 6: Bodenfeuchtigkeitssensor in Pflanzenerde.

83 Sekunden. Bei Versuchen mit größerem Innendurchmessern von 8 und 12 mm konnte die Zeit auf 36 bzw. 32 Sekunden reduziert werden. Die Entscheidung ist auf den kleinsten Innendurchmesser gefallen, damit die Wassermenge genauer dosiert werden kann. In einem weiteren Versuch wurde die Höhe der Wassersäule bei den verschiedenen Innendurchmessern gemessen. Die Ergebnisse variieren zwischen 80 und 90 cm. Versuche die Pumpe mit einer Spannung von 3,3 V zu betreiben waren nicht erfolgreich. Über das Relais (Abbildung 7a), dass über den GPIO-Pin 27 geschaltet ist, kann der Stromkreis der Tauchpumpe gesteuert werden.

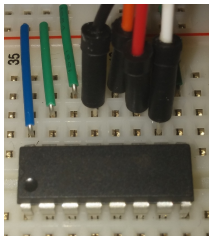
Über die rote LED kann der Nutzer sehen, ob genügend Wasser in dem Wassertank vorhanden ist. Sobald die ausgelesenen Werte des Sensors einen definierten Wert unterschreiten, wird ein Signal auf die LED gegeben.

C. Webinterface

Das Webinterface ist sehr einfach gehalten. Es ist in Abbildung 8 zu sehen. Es sind die Angaben in Prozent des Bodenfeuchtesensors und des Wasserstand-Alarmsensors zu



(a) Relais-Modul



(b) Analog-Digital-Wandler MCP 3008

(c) Tauchpumpe

Abbildung 7: Analog-Digital-Wandler, Tauchpumpe und Relais des Prototypen.

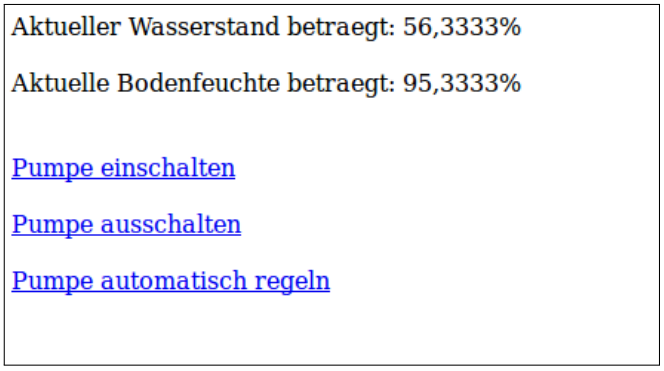


Abbildung 8: Webinterface des Prototypen mit Anzeige der Feuchtigkeit am Bodenfeuchtigkeitssensor und Wasserstand, gemessen über den Wasserstands-Alarmsensor.

sehen. Die Ermittlung des maximalen Wertes zur Berechnung der prozentualen Angabe ist durch das vollständige Eintauchen der Sensoren ermittelt worden. Bei jedem Aufruf der Webseite wird ein Python-Script ausgeführt, dass die aktuellen Daten von den Sensoren ausließt und neu berechnet. Für eine Aktualisierung muss die Webseite erneut aufgerufen werden.

Mithilfe der ersten zwei Links kann die Tauchpumpe manuell geschaltet werden. Der unterste Link startet ein Python-Script, dass die Pflanzenbewässerung automatisch übernimmt. Anhand von Sensordaten des Bodenfeuchtesensor wird entschieden, ob die Tauchpumpe betrieben wird. Es existiert jeweils ein Schwellwert für die maximale bzw. minimale Feuchte. Sollte das Wasser im Wassertank einen Mindestwert unterschreiten wird die Pumpe abgeschaltet. Zum stoppen des Scripts muss der Raspberry-Pi neugestartet werden.

D. Mikrocontroller, eine Alternative zum Raspberry Pi?

Ob der Raspberry Pi die geeignete Steuereinheit für die Bewässerung von Pflanzen ist, wird in diesem Abschnitt diskutiert: Mögliche Alternativen gibt es sehr viele. Gerade die Arduino-Boards mit einem ATmega328-Mikrocontroller sind eine potenzielle Konkurrenz für den Raspberry Pi. Aber auch noch kleinere Mikrocontroller wie z. B. der Attiny85 sind fähig, die analogen Signale auszuwerten und mit Hilfe eines Relais auch die Tauchpumpe zu schalten. Dabei sind sie in den Anschaffung- und Betriebskosten geringer als beim Raspberry

Pi. Mit benötigten Peripheriekomponenten kostet der Raspberry Pi ca. 63 Euro und hat einen Eigenstromverbrauch von ca. 3,5 W [7]. Ein ATTINY 85-20 PU ist beispielsweise bei dem Elektrofachhändler Reichelt für unter einen Euro zu erwerben. Dazu müssten noch weitere Komponenten für die Kommunikation eingesetzt werden. Eine genaue Kostengegenüberstellung existiert jedoch nicht.

Der Nachteil der Mikrocontroller ist, dass dort kein Betriebssystem genutzt wird, sondern das geschriebene Programm direkt über eine Schnittstelle auf den Mikrocontroller gespielt wird. Es ist somit nicht ohne weiteres möglich ein Webserver oder andere Dienste anzubieten. Dieses ist die Stärke des Raspberry Pi gegenüber den zwei Mikrocontrollern. Bei der Spannungsversorgung gibt es keinen Unterschied zwischen Raspberry Pi und Mikrocontroller. Es müsste in jedem Fall eine Spannungsversorgung von mindestens 5V bereitstehen.

Um die vermuteten Kostenvorteile der Mikrocontroller und die Webschnittstelle des Raspberry Pi zu nutzen, wäre eine Server-Client-Architektur denkbar. Der Raspberry Pi als Server und eine mikrocontrollerbasierte Komponente als Client an der Pflanze. Mit dieser Architektur ist es sogar möglich mehr als eine Pflanze zu überwachen und trotzdem eine Webschnittstelle zu nutzen.

E. Einschränkungen und Nachteile

Es ist ein erster Prototyp zur Überprüfung des Konzeptes und gibt Aufschluss über die Realisierbarkeit einer automatischen Pflanzenbewässerung. Im Fokus stand das Zusammenspiel von Hardware-Komponenten wie einem Bodenfeuchtesensor und einer Pumpe zur Beförderung von Wasser. Er wurde gezeigt, dass es technisch Möglich ist eine Zimmerpflanze mit Hilfe eines Raspberry Pi und Sensoren mit Wasser zu versorgen.

Dabei sind jedoch einige Einschränkungen aufgetreten. Als besondere Anmerkung muss gesagt werden, dass es sich hier um einen Prototypen handelt, dieser befindet sich noch in der Entwicklung und kann weiterentwickelt werden für mehr Funktionen und weniger Fehler. Dieser Prototyp kann technisch eine Pflanze bewässern, jedoch nur eine Pflanze zum jetzigen Standpunkt des Projekts. Des weiteren sind Probleme mit den Bauteilen aufgetreten, zum einen konnte durch ganz fest gesteckte Stecker das Signal der Sensoren als ”pendelnd”beobachtet werden. Die Messdaten waren verfälscht, das Ergebnis war ein ”pendelndesänsteuern des Motors. Zum anderen ist der Wasserstandsalarmsensor als Wasserstandsbeobachter für diesen Zweck ungeeignet. Da meist eine Restfeuchte auf dem Alarmsensor vorhanden war, besagte das Messergebnis das weiterhin Wasser im Behälter war, jedoch war das Wasser schon weiter abgepumpt. Die Pumpe lief Gefahr heiß zulaufen. Bei kleineren Pflanzen sollte dieses Problem nicht auftreten, da sie kürzer bewässert werden müssen, sich so der Wasserstand einpendelt und der Wasserstandssensor keine verfälschten Messdaten ausgibt. Ein weiteres Problem, was die Messdaten verfälschen kann, ist wenn der Schlauch zu nah am Bodenfeuchtesensor ist. So wird nicht nur die Schaltung riskiert, wenn Wasser an die Kontakte des Sensors kommt, sondern auch die Richtigkeit der Messdaten, wenn der Bodenfeuchtesensor annimmt es wäre nasser Boden, was nur daran liegt, dass er direkt im Wasser steht, die Pflanze aber nicht genug bewässert wurde. Genau umgekehrt, wenn der Sensor zu weit vom Schlauch entfernt ist, kann es sein, dass mehr Wasser auf die Pflanze gegossen wird, als sie benötigt, dadurch dass sich das Wasser erst in der Erde verteilen muss.

Mit den gegebenen Mitteln, also eine Spannungsversorgung von 5V, Kompaktheit der Schaltung, Größe des Wassertanks, einer kleinen Wasserpumpe und des gewählten Schlauches ist nur die Versorgung einer kleinen Pflanze möglich. Eine große Pflanze würde nicht ausreichend mit einem Wassertank befüllt werden können, dazu kann der Bodenfeuchtesensor keine genaue Aussage über den Wassergehalt der Erde sagen, da die Erde auch nicht gleichmäßig bewässert werden kann und auch die Tiefe des Sensors nicht weit genug in die Erde hineinreicht.

VI. FAZIT

Das Ziel war es, einen Prototypen zu entwickeln, der eine Pflanze automatisch bewässern kann. Dieses wurde in dem Projekt umgesetzt. Der Begriff „automatisch“, ist in dieser Ausarbeitung als eine Ausübung von einzelnen Arbeitsschritten zur Bewässerung von Zimmerpflanzen durch den Prototypen definiert. Der Prototyp ist schrittweise und evolutionär entwickelt worden. Im ersten Schritt wurden die verschiedenen Sensoren und Aktoren einzeln getestet. Es entstand ein erster Prototyp, der mithilfe eines Tasters Wasser aus dem Wasserbehälter zur Pflanze befördern konnte.

Im zweiten Schritt wurde der Grad der Automation erhöht, indem auf Grundlage von Sensordaten eines Bodenfeuchtigkeitssensors die abgegebene Wassermenge angepasst wurde. Je nach Positionierung des Bodenfeuchtigkeitssensors in der Erde ist eine ausreichende Bewässerung erkannt worden.

Abschließend ist zu sagen, dass das Projekt erfolgreich ist. Der Prototyp funktioniert im definierten Rahmen und erfüllt die Anforderungen. Das Ziel einen Prototypen mithilfe eines Raspberry Pi zu entwickeln ist umgesetzt.

VII. AUSSICHT

Die Zukunft dieses Projekts wurde schon bei der Entwicklung geformt. So wurden von vornherein optionale Ziele festgelegt, die das Projekt nun leiten sollen. Im Vordergrund steht dabei, dass das Handling und die Benutzerschnittstelle einfacher und genauer gestaltet werden. Dazu kommt, dass mindestens eine weitere Pflanze bewässert werden kann.

Eine Pflanze benötigt mehr als nur Wasser zum überleben, auch Sonneneinstrahlung, Temperatur und Nährstoffe gehören zu den Faktoren, die eine Pflanze wachsen lassen. Damit lässt sich ein weiteres Ziel formulieren, die Erweiterung der Sensoren, sodass mehr Bedürfnisse gestillt werden können.

Wenn der Raspberry Pi nicht an eine Steckdose angeschlossen werden kann, oder, wenn er an eine Steckdose angeschlossen ist und die Stromversorgung trotzdem unterbrochen wird, muss die Weiterversorgung versichert werden. Dafür soll der Raspberry Pi entweder durch Batterien oder mit einem Solarpanel versorgt werden.

LITERATUR

[1] Koubachi, "Meet koubachi," http://www.koubachi.com/, o. J., besucht am: 01.02.15.

[2] Windowfarms, "Windowfarms," http://www.windowfarms.com/, 2014, besucht am 02.02.15.

[3] Parrot, "Parrot flower power," http://www.parrot.com/de/produkte/flower-power/, 2014, besucht am: 01.02.15.

[4] OpenSprinkler, "Opensprinkler," https://opensprinkler.com/, 2014, besucht am 01.02.15.

[5] K. Jacobi, *Das farbige Hausbuch der Zimmerpflanzen*, 12nd ed. BLV Verlagsgesellschaft, 1989.

[6] G. Mart and J. Mart, *So pflegt man Zimmerpflanzen*, 1st ed. Verlag Paul Parey, 1973.

[7] R. Follmann, *Das Raspberry Pi Kompendium*. Springer Berlin, 2014.

[8] RaspberryPi, "Raspberrypi," http://www.raspberrypi.org/, 2014, besucht am 02.02.15.

[9] ——, "Raspberrypi," http://swag.raspberrypi.org/products/noobs-8gb-sd-card/, 2014, besucht am 02.02.15.

[10] K. Dembowski, *Raspberry Pi - Das Handbuch: Konfiguration, Hardware, Applikationserstellung*. Wiesbaden ;s.l: Springer Vieweg, 2013.

Distributed Heating Control with Raspberry Pi

Thomas Nordlohne and Patrick Schmale
Carl von Ossietzky University of Oldenburg
26129 Oldenburg, Germany
{thomas.nordlohne, patrick.schmale}@uni-oldenburg.de

***Abstract*—The reliance on using Raspberry Pi for both hobby and industry domain is increasing. A Raspberry Pi is a small-sized and a low-budget alternative to conventional computer. In addition, an important benefit is the comfortable (graphical) user interface while programming because of the underlying linux distribution. This paper illustrates an approach of using a Raspberry Pi for a low-budget home automation alternative, serving a distributed heating control. One of the main part of this approach is the design of a low-budget temperature sensor communicating with the Raspberry Pi over the *industrial, scientific and medical* (ISM) band, e.g. 433 or 868 MHz. In addition, a communication between multiple Raspberry Pi's is illustrated to exchange control values.**

***Keywords*—Raspberry Pi, Smart Home, FHEM, Heating Control.**

I. Introduction

Within the last years, home automation became a popular topic, because it increases the energy-efficiency of the home environment. For example, reducing the room temperature by one degree, results in energy savings up to five percent [1]. Especially in autumn and winter, an average household spends 85 % of it's energy needs for heat production [1]. So there is an enormous potential for energy savings by intelligent heating control systems. Such systems has to consider different use cases. For instance turning on the heating, if nobody is at home, isn't really necessary. Also if somebody opens a window, it's self-defeating to turn up the heating.

Many manufacturers already supply such systems, e.g. Siemens Synco Living, ENOCEAN or HomeMatic. But the problem with these systems is that they are often very expensive. Furthermore, the manufacturers use proprietary solutions, which aren't expandable.

This paper provides an initial concept of a low-budget distributed heating control based on the Raspberry Pi (shortly RP). The RP is a low-priced single-board computer. With it's different interfaces and General Purpose Input Output (GPIO) Pins, it enables engineers to build complex control systems. There are also many shields for the Raspberry Pi to extend it's functionality. Due to these facts, it is adequate to build a distributed heating control using RP.

A. Related Work

There exist many principles for heating a flat or a building. Finding an optimal solution for this heating problem is the focus of current research activities. In [2], J. Zacek identifies three significant problems for building a heating control system. The first one deals with the regulation of the valves in each room. Therefore, programmable thermostats are necessary, which opens and closes the valve depending on the room temperature. Another problem is the measurement of the temperature in each room. The simplest method is the integration of temperature sensor within the thermostats. But this solution has an enormous disadvantage, because the influence of the heating on the sensor is significant. It would be much better to measure the temperature in the middle of the room. The third problem is the existence of a central control unit collecting all information (e.g. temperature, valve opening percentage) and perform specific actions.

There is not much literature about using the Raspberry Pi as a heating control system. J. Zacek uses the RP to build a programmable heating control system with components connected via Ethernet [2]. Since many households don't have an Ethernet network to connect the necessary devices of the heating control system, this solution is impractical. Another opportunity is presented by E. F. Engelhardt in [1]. He wired a temperature sensor with the GPIO-Pins of the RP and used the open source server FHEM[1] for controlling and displaying tasks. This solution is limited to the wired connection between the temperature sensor and the RP. This leads to many restrictions. For instance, the temperature sensor must be placed in the direct environment of the RP, which limits the flexibility. Furthermore, it is difficult to connect many temperature sensors from different rooms with the RP. Therefore, long wires are necessary, or each room has to be equiped with an own RP. This increases the costs.

B. Proposed Solution

We propose to design a low-budget distributed heating controller. Fig. 1 illustrates our scenario for an autonomous heating control system using a Raspberry Pi. In our approach, we use one RP provided with a wireless transceiver to control the temperatures of different rooms independently. Therefore, each room requires an own heater equipped with a radio receiver module within its thermostat. To measure the actual room temperature, each room needs a radio temperature sensor module, too. This module sends the actual room temperature via a wireless connection to the Raspberry Pi. In addition the RP requires a reference temperature input (desired indoor temperature) the user can define by itself. After calculating the control values, the Raspberry Pi sends each of these values to the right thermostat via the wireless connection.

[1] Freundliche Hausautomation und Energie-Messung [3]

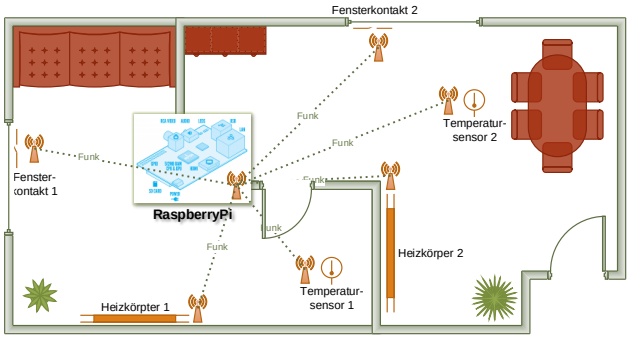


Fig. 1: Scenario of distributed heating control

Another important task of the Raspberry Pi is the detection of disturbances like an opened window. Due to that, the RP has to recognize whether a window is closed or opened. On this account, a window will be provided with a sensor which detects an opened window. To realize an economical heating control system, the RP turns off a thermostat if a window is opened in the room. Each window sensor owns a radio module connected to the RP, too.

The next feature of this approach is a humidity control (see Fig. 5). The autonomous humidity control faces the challenge of having limited efficient electrical opportunities to change the humidity. To avoid dry air, the user can for example open a window or can power on an electrical humidifier. Furthermore, considerable efficient electrical capabilities to reduce a high humidity do not exist. To save this feature, we decided to display the humidity and, if necessary, to warn the user. In addition, a second Raspberry Pi powers on an electrical humidifier if the humidity is too low.

The reason why using another RP is the flexibility of the positioning of these RPs. Thus, the RP for the heating control (RP1) could positioned hidden while the RP in combination with the humidifier (RP2) could be positioned in the middle of the room. Both of them will be connected to the local area network (LAN) in order that RP1 sends RP2 the signal to (de-) activate the humidifier.

II. Raspberry Pi with FHEM

FHEM is a server for house automation written in perl, which is used to automate common tasks in home environments, e.g. switching the lights or control the heating. FHEM comes with different front ends. There are user interfaces for web browsers (see Fig. 2) as well as for smartphones and other mobile devices. Furthermore, FHEM runs on all servers with a perl interpreter, which makes it well-suited for the RP. To access the actors and sensors, some additional hardware is necessary. To establish a wireless connection with the sensors and the thermostats, the RP needs a radio transceiver module. Therefore, we decided to use the CC1101 Transceiver[2], because it could be mounted directly onto the RP and it supports all relevant frequencies (433 MHz / 868 MHz / 915 MHz)

[2]http://shop.busware.de/product_info.php/products_id/97

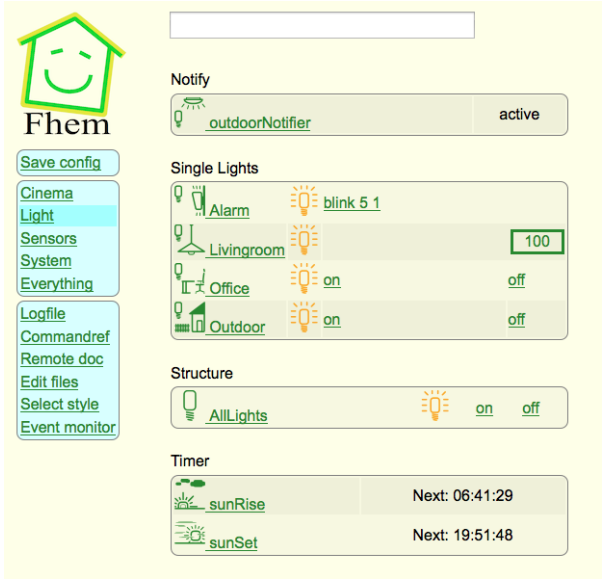


Fig. 2: Screenshot of FHEM's Web-Frontend

used in the home automation context. Another benefit of the CC1101 is that its integration into FHEM is easy. The following paragraphs describe how FHEM could be installed on the RP and the configuration of home automation devices in FHEM.

A. Installing FHEM on Raspberry Pi

Since the RP comes with the linux-based distribution Raspbian, it is not complicated to install FHEM. The initial point is a clean installation of Raspbian on the RP. The scripting language perl is already installed in this distribution and must only be completed by the "serialport" package. After this, the current version of FHEM could be downloaded as a debian package from the website with the following command:

```
1 wget  http://fhem.de/fhem-X.Y.deb
```

To install FHEM, the dpkg-command with the current downloaded FHEM version must be executed. Next, FHEM could be accessed via a webbrowser using the IP address of the RP and Port 8083, e.g.:

```
1 http://10.42.0.55:8083/fhem
```

Now FHEM is completely installed on the RP and always runs in the background as a service.

B. Configuration of Thermostats and Window Contact Sensors

After a successful install of FHEM on the RP, the radio transceiver module CC1101 has to be prepared for the communication with the sensors and thermostats. Therefore, the serial line used by the CC1101 has to be freed-up. This could be achieved by removing any references to *ttyAMA0* from */etc/inittab* and */boot/cmdline.txt*. The Reset-Pin of the radio

Fig. 3: thermostat FHT8V from ELV

transceiver module is connected to GPIO 17 of the RP and is held by default. To unreset the radio module, this pin must be pulled to high explicitly. It is done by the following terminal commands:

```
if test ! -d /sys/class/gpio/gpio17; then
    echo 17 > /sys/class/gpio/export; fi
echo out > /sys/class/gpio/gpio17/
    direction
echo 1 > /sys/class/gpio/gpio17/value
```

Now the CC1101 is active and it's LED starts blinking. Since this actions have to be executed after every restart of the RP, it would be smart to save them within a shell script executed on startup.

The next step is to add the radio transceiver module to FHEM. This could be achived by simply executing `define` in the command line of FHEM with the baudrate of the CC1101 and it's specific house code as parameters:

```
define <name> CUL /dev/ttyAMA0@38400 <
    housecode>
```

Now the radio transceiver module appears as an own device in FHEM and could be used to connect thermostats and other sensors. The `housecode` parameter is necessary to interconnect actors and sensors in the same household. Furthermore, it provides a simple security mechanism and prevents unauthorised communication with other devices.

For the thermostats we are using the devices FHT8V[3] (see Fig. 3) from ELV, because they aren't very expensive and they use the 868 MHz frequency band. To integrate these thermostats in FHEM, the `define` command with it's device code has to be executed:

```
define <name> FHT8V <housecode>
```

Now the thermostat appears in the list of the registered devices in the FHEM web front end. The next step is to pair the device defined in FHEM with the physical thermostat mounted onto the heating, which could easily be achieved by executing:

```
set <name> pair
```

To check if pairing was successful, a setpoint could be sent to the thermostat:

```
set <name> valve 75
```

The value is represented in percent and has to be between 0 and 100. It has to be considered, that the thermostat doesn't receive values immediately. Instead, it wakes up every two minutes and checks if a new value was sent. So there is a delay in the transmission of the setpoint.

The window contact sensors come also from ELV and are called FHT80TF[4]. They operate at the same frequency of the thermostats and have a low energy consumption. The integration of the window contact sensors in FHEM is as easy as the integration of the thermostats. Only the `define` command has to be executed in the FHEM console:

```
define <name> CUL_FHTTK <devicecode>
```

The devicecode of the sensor is a unique id which is assigned during manufacturing. It appears in the logfile of FHEM. Also the FHT80TF doesn't send state changes immediatly. Furthermore, the module sends a message with the current state every 1 to 4 minutes. This value is displayed in the device properties of the window contact sensor in the FHEM web front end.

III. Designing a Temperature & Humidity Sensor

The challenge of the design of the temperature and humidity sensor is the development of a small and low-power sensor board satisfying all the defined requirements. The main task of this board is to send the current temperature and humidity value wirelessly to the RP after grabbing these values from the sensor. Such sensors exists already [4], but they are expensive for private consumer if many rooms will be equipped with these modules. Finally, we decided to develop an own low-budget temperature sensor board.

Fig. 4 illustrates our approach of this design. The circle on the right in Fig. 4 presents the voltage supply. In this case, it is a 3 V button cell with a diameter of 24 mm. The red and black lines in Fig. 4 illustrates the supply and ground wires.

On the top left corner, the SHT21 sensor is placed, which provides the current values for temperature and humidity. On the bottom left corner, an ATtiny84 microcontroller is located. The wireless RFM02-transmitter is placed on the bottom right corner.

The ATtiny84 is a low-power 8-bit microcontroller having a very small chip-size (only 16 pins) and an 8k programmable flash. Its operating voltage is in range of 2.7 -5 V and it has a power consumption of 0.1 μA in power-down mode or of 300 μA in active mode at 1 MHz system clock. Since this

[3] http://www.elv.de/ventilantrieb-fht8v-2-arr-bausatz-inkl-batterien.html

[4] http://www.elv.de/funk-tuer-fensterkontakt-fht80tf-2.html

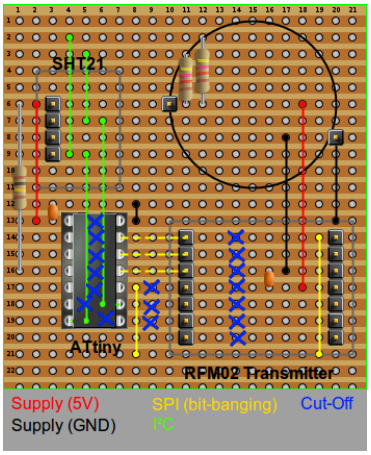


Fig. 4: Design of the temperature sensor board

microcontroller has a serial peripherial interface (SPI) for in-system programming [5], it can be programmed via the SPI general-purpose input/output registers (GPIO) of a RP and the avrdude[5] package [7]. These are all reasons why an ATtiny84 is predestinated for our approach.

The utilization of SHT21 is accounted for by its very tiny foot-print of 3x3 mm and its accuracy of ±0.3 % °C (±2 % RH for humidity) [8].

Because of the usage of the 868 MHz frequency, the RFM02-transmitter is selected [9]. There are already initial C-libraries and application notes for the communication between the ATtiny84 and these integrated cicuits (IC) [10] [11]. The software of these libraries has to be adapted to this design (port(-pin) allocation etc.).

The RFM02 transmitter has to be connected to the microcontroller via an SPI port (see Fig. 4, yellow lines) and the SHT21 sensor could be connected via the inter-integrated circuit (I^2C) interface (see Fig. 4, green lines). The utilization of the ATtiny84 faces the challenge to use the SPI and the I^2C interface simultaneously as the clock pin for SPI and I^2C are placed at the same pin. Finally, one of both interfaces has to be emulated by making other use of free pins (*bit-banging*). In our design, the SPI interface (see Fig. 4, yellow lines) should be emulated.

The first execution sequence of the microcontroller is the initialization of the two ICs. A neccessary configuration for the RFM02-transmitter is the frequency which could be set to 433, 868 or 915 MHz. After configuring the ICs, the microcontroller enters an infinite loop. Inside this loop, the controller requests new 16-bit values x_T and x_{RH} from the SHT21 sensor via the I^2C buThis sensor has usually an I^2C address of 0x40 [8]. Afterwards, the received values has to be normalized with the formula

$$T = -46.85 + 175.72 \cdot \frac{x_T}{2^{16}} \tag{1}$$

and for the current humidity with the formula

$$RH = -6 + 125 \cdot \frac{x_{RH}}{2^{16}}, \tag{2}$$

respectively [8].

After converting the temperature and the humidity, these values have to be sent via the RFM02-transmitter to the remote Raspberry Pi. For this communication, a special protocol is required has to be implement by the software of the ATtiny84, too (see section III-A). Because temperature changes are very sluggish, the microcontroller could send this data every other seconds to satisfy a long-term operating time. Thus, the ATtiny could operate at a very low system clock (e.g. at double-digit kHz level).

Finally, by using this design layout, the acquisition costs of one temperature and humidity sensor board could be reducible to under 12 €[6].

A. *Communication between sensors and Raspberry Pi*

As mentioned in Section III, a communication protocol is required to transmit the sensor values from the RFM02 to the RP. In the field of home automation there doesn't exist a standard protocol for wireless communication. In fact every manufacturer implements his own proprietary solution. The radio transceiver module CC1101 used in this solution, has a property `rfmode` to support these different protocols. The default value is `SlowRF` for communication with FS20[7] or FHT[8] devices at a data rate of 1 kHz. To communicate with HomeMatic[9] or MAX![10] devices the data rate must be set to 10 kHz which could be achieved by changing the `rfmode` to HomeMatic or MAX. In summary the mode of the radio transceiver module depends on the devices used for home automation and can't be changed dynamically. This fact makes it impossible to use devices from different manufacturers at the same time and exacerbates the implementation of a wireless communication protocol.

Since the thermostats and window contact sensors use the FHT protocol, it would be a good idea to implement this protocol for the sensors, too. In table I, the structure of this protocol is shown. Every transmission starts with a synchronisation of 13 bits with the pattern `0000000000001`. This is necessary to inform the receiver about the new transmission. The payload of the protocol is separated into 8-bit packages with a parity bit after each package. It starts with the `housecode` of the device (see Section II). Due to the fact that this is a 16-bit value, it has to be divided into its High-Byte (HC1) and Low-Byte (HC2). The next package contains the address of the device. Since the FHT protocol uses one-hot encoding and the address is an 8-bit value, it is possible to control eight different thermostats. A broadcast can be sent by `0x00`. The next package contains the command, which should be executed by the receiver. A list of some relevant commands can be found

[5]"Avrdude is an utility to download/upload/manipulate the ROM and EEPROM contents of AVR microcontrollers using the in-system programming technique (ISP)" [6].

[6]These costs include the prices of the ICs and the necessary supplies.
[7]http://www.elv.de/fs20-funkschaltsystem.html
[8]http://www.eq-3.de/fht.html
[9]http://www.homematic.com/
[10]http://www.eq-3.de/max-heizungssteuerung.html

TABLE I: FHT protocol

Sync	HC1	Parity	HC2	Parity	Address	Parity	Command	Parity	Extension	Parity	Checksum	Parity	EOT
13 bit	8 bit	1 bit	8 bit	1 bit	8 bit	1 bit	8 bit	1 bit	8 bit	1 bit	8 bit	1 bit	1 bit

in table II. The most commonly used commands are `0x0F` to pair the thermostat with the sender, `0x01` to opens the valve completely, and `0x02` to close the valve completely.

The next package contains the checksum, which is formed by the summation of `0x0C`, the High- and Low-Byte of the housecode, the address, and the command. The last bit (EOT) marks the end of the transmission.

There is the opportunity to expand the data package by an extension byte, which could be used to transmit some additional values (e.g percentage value of the valve opening). This byte is inserted between command byte and the checksum byte. Considering a data rate of 1 kHz, the transmission of one command takes between 47,6 ms and 65,6 ms.

Table II shows that there isn't any command to transmit temperature or humidity values. This complicates our preferred solution, since the FHT protocol has to be extended by commands for the temperature and the humidity. Due to this fact, we will only explain the concept how to achieve this, but won't give any implementation examples. Nevertheless, it has to be considered that the extension of the FHT protocol may lead to incompatibilities, if the manufacturer decides to revise his proprietary implementation.

In table III some possible commands are displayed, which could be used to transmit the temperature and the humidity of the SHT21. The current sensor value should be stored in the extension byte. The ATtiny84 has to implement the transmit routine of this extended FHT protocol. This could be very difficult to test, because there isn't a comfortable debug support. Furthermore, FHEM must be able to receive the temperature and humidity values from the ATtiny84. Therefore, a new module has to be written in Perl, which interprets the received data of the CC1101 transceiver and extracts the current temperature and humidity values. The integration of these two components could also be a problem, because of the wireless transmission of the data.

TABLE II: List of FHT commands

hex	description
`0x00`	execute synchronisation
`0x01`	open valve completely
`0x02`	close valve completely
`0x06`	valve opening in %
`0x08`	offset setting
`0x09`	execute decalcification
`0x0C`	synchronisation is running
`0x0E`	testing the actors
`0x0F`	pairing of the thermostat
`0x10 - 0x1F`	low battery alert signal
`0x20 - 0x3F`	extension bit
`0x40 - 0x7F`	bidirectional command
`0x80 - 0xFF`	repetation of a previous command

TABLE III: Possible extension of FHT commands to transmit temperature and humidity values

hex	description
`0x03`	transmit temperature value
`0x04`	transmit humidity value

IV. Communication between multiple distributed Raspberry Pi's

Our approach proposes a communication between multiple RPs (see Fig. 5). Since many households are using local area networks (LAN), the RPs can communicate via the ethernet interface or an extra WLAN module for the RP. Finally, the Raspberry Pi Model B or higher has to be chosen.

This communication is necessary to control the humidifier connected to the GPIO of the RP2. The RP1 calculates the current humidity by reading the SHT21. Afterwards, it has to match the value against a minimum and a maximum reference value. Assuming the current humidity deceeds the minimum reference value, the RP1 sends a command to the RP2 via LAN. Finally the RP2 has to power on the humidifier. In the other case of exceeding the maximum value, the RP1 has to send a another command to the RP2.

```
from socket import *
# open socket for inet stream
s = socket(AF_INET, SOCK_STREAM)
# connect to server (busy waiting)
s.connect((host,port))
# after connection is established, send data
conn.send(data)
```

Listing 1: Example of Python source code for a client

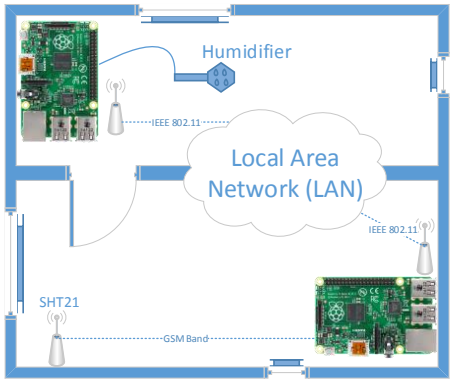


Fig. 5: Raspberry Pi's communication over a local area network to exchange data

To establish a connection between these RPs, each Raspberry Pi instantiates a socket connection using the transmission control protocol (TCP) or the user datagram protocol (UDP). Because of using the RP, the sockets for the client (see Listing 1) and the server (see Listing 2) can be realized by a Python script. The programming language Python facilitates an easy and quick software implementation.

```
from socket import *
# ... define 'host' and 'port' ...
# open socket for inet stream
s = socket(AF_INET, SOCK_STREAM)
# bind socket on local ip address and port
s.bind((host, port))
# wait for clients
s.listen(1);
# accept new client
conn,clientAddress = s.accept()
# wait for data
data = conn.recv(1024)
```

Listing 2: Example of Python source code for a server

To control the humidifier by setting a GPIO-pin of RP2, the RP has to set this GPIO-pin to an output pin. This configuration could be done by using a shell script or shell commands within the Python source code (see Listing 3). The configuration of each GPIO-pin is stored in a seperate file located in */sys/class/gpio/*.

```
import subprocess
# ... define 'gpioNr' ...
# Enbale GPIO pin
subprocess.call("echo \"" + str(gpioNr) + "\" > /sys
    /class/gpio/export",shell=True)
# Set pin to output pin
subprocess.call("echo \"out\" > /sys/class/gpio/gpio
    "+str(gpioNr)+"/direction",shell=True)
# Write value (1 or 0) to pin
subprocess.call("echo \""+str(value)+"\" > /sys/
    class/gpio/gpio"+str(gpioNr)+"/value",shell=True
    )
```

Listing 3: Example for configuring a GPIO-pin in Python

V. Conclusion

Taking everything into account this paper shows one possibility to create a heating control system with the Raspberry Pi. Therefore, it's a good idea to use FHEM as a home automation server on the RP, because its installation is simple and it supports a wide range of home automation tasks. Although the RP doesn't provide wireless communication out of the box, this could easily be achieved by extending the RP with a radio transceiver module (e.g. CC1101). While establishing a connection between the RP and the thermostats isn't complicated, the design of a temperature and humidity sensor turned out to be more difficult.

Due to the limited time of this work, the implementation of the sensor can't be realized. In Section III-A the concept of this implementation is shown. Although we already started implementing and testing the sensor and it's wireless communication, we didn't get any positive results. The reasons therefore are the complexity of the RFM02 module and difficulties during debugging and testing the ATtiny84. There was no opportunity to visualize the IC traffic between the SHT21 and the ATtiny84 and the SPI traffic between the ATtiny84 and the RFM02, which makes error detection quite difficult. Furthermore we were confronted with delivery problems and first got the wrong components.

Proceeding this work, the next steps would be to finish the design of the temperature and humidity sensor. Therefore, the FHT protocol should be extended to transmit the sensor values of the SHT21 via the RFM02 module. This communication has to be implemented in both FHEM and ATtiny84. A first draft of the software obtaining the sensor values from the SHT21 and sending them via the RFM02 module already exits and could be used for further improvements.

Finally, we would like to emphasize that our proposed solution can be realized. But this requires deep knowledge of wireless communication protocols and magnificent experience in programming embedded devices.

References

[1] E. F. Engelhardt, *Hausautomation mit Raspberry Pi*, 3rd ed. Franzis-Verlag GmbH, 2014.

[2] J. Zacek and M. Janosek, "Programmable control of heating for systems with long time delays," in *Control Conference (ICCC), 2014 15th International Carpathian*, May 2014, pp. 705–709.

[3] U. Maaß, *Heimautomatisierung mit fhem*, http://fhem.de/Heimautomatisierung-mit-fhem.pdf, 2014, last checked: 05.01.2015.

[4] E. GmbH, "Stm 331 - batterieloses temperatursensor-funkmodul - inklusive solarzelle und helix-antenne," Website, EnOcean GmbH, https://www.enocean.com/de/enocean_module/stm-331/, last checked: 05.01.2015.

[5] *Datasheet of 8-bit Microcontroller with 2/4/8K Bytes In-System Programmable Flash*, Atmel Corporation, http://www.atmel.com/Images/doc8006.pdf, 2010, last checked: 04.01.2015.

[6] F. S. F. Inc., "Avrdude - avr downloader/uploader," Website, http://www.nongnu.org/avrdude/, last checked: 05.01.2015.

[7] C. Hoile, C. Bowman, S. Meijer, B. Corteil, L. Orsini, and T. Mott, *Make: Raspberry Pi and AVR Projects: Augmenting the Pi's ARM with the Atmel ATmega, ICs, and Sensors*. Maker Media, Incorporated, 2014. [Online]. Available: https://books.google.de/books?id=U5Y8BQAAQBAJ

[8] *Datasheet SHT21 - Humidity and Temperature Sensor IC*, Sensirion, http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT21_Datasheet_V4.pdf, May 2014, last checked: 04.01.2015.

[9] *RFM02 Universal ISM Band FSK Transmitter*, HopeRF Electronic, http://www.hoperf.com/upload/rf/RFM02.pdf, 2014, last checked: 04.01.2015.

[10] *Sample Code for SHT21*, Sensirion, www.sensirion.com/nc/en/products/humidity-temperature/download-center/?cid=880&did=102&sechash=c204b9cc, February 2010, last checked: 04.01.2015.

[11] *RFM01 and RFM02 libraries*, posted by contrechoc on June 22, 2010, http://myfablab.wordpress.com/2010/06/22/rfm01-and-rfm02-libraries-for-arduino-18-version-0-1/, 2014, last checked: 04.01.2015.

Ausarbeitung kooperatives, verteiltes Rauchmeldesystem mit dem Raspberry Pi

Thorben Kloppe und Hendrik Schönland

I. Einleitung

Unser Arbeitsauftrag im Proseminar 'Raspberry Pi' bezieht sich auf die Fragestellung, ob und wie man mit Hilfe des heute, zumindest im schulischen Bereich immer beliebter werdenden kleinen Low-Cost-Computers Raspberry Pi ein kooperatives, verteiltes Rauchmeldesystem entwickeln kann.

Wir beschreiben im Folgenden die Problemstellung und umreißen unsere Zielsetzung.

Dann skizzieren wir die unterschiedlichen existierenden Typen von Brandmeldesystemen und stellen den Raspberry Pi als von uns gewählte Schaltzentrale eines solchen Systems kurz vor.

Anschließend gehen wir relativ kurz und allgemein darauf ein, wie wir unser Vorhaben umgesetzt haben.

Zu unserem Vorgehen im Einzelnen verweisen wir auf das auftragsgemäß ebenfalls von uns vorgelegte, direkt auf den Nachvollzug des Aufbaus, der Implementation notwendiger Software sowie der Anwendung unseres Systems ausgelegte Handbuch.

II. Problemstellung

Brandmelder sind für die Erkennung von Bränden im frühestmöglichen Entstehungsstadium gedacht. Sie sollen durch einen lauten Signalton Personen vor Brandrauch sowie dem eigentlichen Feuer warnen.

Statistisch gesehen sterben in Deutschland wie auch anderswo auf der Welt nur 5 Prozent der von einem Brand betroffenen Menschen am eigentlichen Feuer, d. h. sie verbrennen. 95 Prozent der Toten fallen hingegen dem mit dem Feuer einhergehenden Brandrauch zum Opfer [1]. Im Brandrauch ist vor allem das Kohlenstoffmonoxid verantwortlich für die meisten Vergiftungen, die im schlimmsten Fall zum Tod der Betroffenen führen [2].

Wichtig ist zu erwähnen, dass die meisten Todesopfer in der Nacht zu beklagen sind. Die Menschen werden durch den Brand im Schlaf überrascht. Um diese Todesopfer zu vermeiden haben die Landesregierungen der Bundesrepublik einschlägige Gesetze bzw. Verordnungen erlassen. In der Niedersächsischen Bauordnung (NBauO) vom 3. April 2012 ist in §44(5) geregelt, dass Rauchmelder in Neu- und Umbauten Pflicht sind. Für bestehende Wohnungen ist der Einbau bis zum 31.12.2015 nachzuholen.

Zwingend anzubringen sind Rauchmelder in Schlafräumen, Kinderzimmern sowie Fluren, die über Rettungswege von Aufenthaltsräumen führen. Das soll verhindern, dass Personen von Rettungswegen abgeschnitten werden.

Verantwortlich für den Einbau sind die Eigentümer, für die Betriebsbereitschaft die Mieter, Pächter oder sonstige Nutzer [1]. Durch Brandmelder sind die Brandopfer seitdem um 50 Prozent zurückgegangen [3].

Über diese Notwendigkeit und die zweifellos nachvollziehbare dahinter liegende Sinnhaftigkeit hinaus ist es jedoch anzustreben, dass die einzelnen Meldegeräte vernetzt sind und somit miteinander kommunizieren können. Nur so kann in größeren Gebäuden eine ernsthafte Warnung stattfinden. Sprich: Löst ein Melder aus, muss er den anderen „Bescheid sagen“ und diese auch zur Auslösung bringen.

Des Weiteren sollte das System die Brandmeldung auch nach außen senden können (Eigentümer, Wachdienst, Polizei, Feuerwehr usw.).

Schließlich wäre es - gerade in Bezug auf den zuletzt genannten Aspekt - hilfreich, wenn die Information, welcher der Brandmelder, sprich: an welchem Ort, der Alarm ausgelöst wurde, sprich: die Quelle des Feuers zu suchen ist.

III. Ziele und Einschränkungen des Projektes

Ziel unseres Projektes ist es somit, ein System vernetzter Feuermelder herzustellen und auszuprobieren. Dabei wollen wir so kostengünstig wie möglich vorgehen. Unsere Absicht ist, günstiger als bisher auf dem Markt erhältlich vernetzte Feuermelder herstellen zu können.

Grund für dieses Streben ist, dass möglichst jeder die Möglichkeit erhalten sollte, für seine Sicherheit sorgen zu können.

Dazu nehmen wir den Raspberry Pi als Schaltzentrale zur Hilfe. Seine Funktionalität ist ausreichend; sein Preis angemessen. Hinzu kommt, dass wir die Rauchmelder ja nicht nur vernetzen wollen, sondern eher eine Art Brandmeldeanlage, wie sie in großen Einkaufszentren sowie in Gebäuden mit großer Anzahl an Personen vorkommen, im Blick haben. Diese Anlagen geben bei Alarm auch klare Signale nach außen, sprich: an Dritte. Ein solches Brandmeldesystem sollte u. E. für normale Privathaushalte finanziell erschwinglich sein.

Grundsätzlich kann man Feuermelder auch ohne Raspberry Pi vernetzten. Das Ganze geht vom Prinzip her auch ausschließlich mithilfe von Stromschaltkreisen zu realisieren. Hier ist aber mangels zentralem Knotenpunkt (Schaltzentrale) weder eine Identifizierung des auslösenden Melders noch ein Versendung einer Nachricht nach außen möglich.

Die Brandmeldeanlage in unserem Projekt soll aber gerade in der Lage sein zu registrieren, dass ein ganz bestimmter, genau zu identifizierender Rauchmelder ausgelöst hat. Daraufhin soll diese Information, zunächst einmal als einfaches Signal z.B. an eine LED weitergeleitet werden. Hat man das geschafft, ist es nicht mehr sonderlich schwierig weitere Rauchmelder auszulösen. Gleichzeitig ist es dann auch möglich zu differenzieren, d.h. anzuzeigen, welcher der Rauchmelder ausgelöst hat. Insbesondere kann dann an eine Weiterleitung per SMS an den Hausbesitzer oder an die Leitstelle der Feuerwehr gedacht werden. Ist ein Internetanschluss in der Brandmeldezentrale vorhanden, ist ein Informationsversand über das Internet, beispielsweise per E-Mail, möglich.

In unserem Projekt benutzen wir fertige, d.h. auf dem zugänglichen Markt gekaufte optische Rauchmelder, da es zu aufwendig gewesen wäre, selbst einen eigenen Rauchmelder herzustellen, der die in Deutschland geltende, europäische Norm 14604 (DIN EN 14604) erfüllt [4]. Nur Brandmelder, die diese DIN erfüllen, sind in Deutschland zugelassen. Konkret wird in diesem Projekt also versucht, ein System aus konventionellen, kommerziell gehandelten optischen Rauchmeldern und dem Raspberry Pi zu erschaffen.

Nach ursprünglich weitergehenden Ideen haben wir uns im Laufe der Umsetzung in unseren Zielen begrenzt. Im Kontext des Zeitrahmens eines Proseminars schien es uns im praktischen Teil angemessen, uns exemplarisch auf die kabelgebundene Verbindung eines Rauchmelders mit dem Raspberry Pi zu beschränken. Als wiederum exemplarische Reaktion des Pi's haben wir das Verständigen einer Person mittels E-Mail auf ein Handy gewählt. Hier nutzen wir mangels kabelgebundenem Direktzugang in der Uni einen WLAN-Adapter.

Insgesamt können wir vier Rauchmelder per Kabel an einen Pi anschließen. Als Reaktion des Gerätes kann durch relativ einfache Umarbeitung des Programmcodes erreicht werden, dass sowohl eine Benachrichtigung nach außen erfolgt als auch, dass der auslösende Rauchmelder die anderen drei benachrichtigt und einschaltet.

IV. Verschiedene Typen von Brandmeldern

Im Folgenden wird auf verschiedene Typen von Brandmeldern eingegangen und erläutert, mit welchem in unserem Projekt gearbeitet wird. Hierbei ist zu beachten, dass es eine ganze Palette unterschiedlicher Brandmelder gibt. Wir beschränken uns dabei auf vier Typen.

Brandmelder werden in automatische und nicht automatische eingeteilt. Bei den nicht automatischen Brandmeldern handelt es sich um sogenannte Handmelder. Diese zeichnen sich in der Regel durch einen Druckknopf aus, den Personen im Brandfall manuell auslösen können. In der Regel wird durch Drücken de Knopfes ein immer wiederkehrendes Signal im Gebäude ertönen sowie die zuständige Feuerwehr verständigt. Gleichermaßen existieren in solchen Gebäuden Evakuierungspläne, die dann greifen sollen. Im Allgemeinen sind solche Handmelder in öffentlichen Gebäuden, beispielsweise in Schulen, weniger in privaten Wohnungen, an mehreren Stellen fest installiert. Derartige Geräte sind nicht im Rahmen unseres Themenbereichs.

Im Folgenden werden deswegen lediglich vier Arten von automatischen Brandmeldern kurz vorgestellt und ihre Funktionsweise aufgezeigt: der Ionisationsrauchmelder, der Wärmemelder, der Flammenmelder und der optische Rauchmelder.

A. Der Ionisationsrauchmelder

Der Ionisationsrauchmelder ionisiert mit Hilfe von Alpha- oder sogar Betastrahlung die Umgebungsluft im Sensor. Die Leitfähigkeit der Luft wird durch zwei Elektroden gemessen. Tritt rauchhaltige Luft auf, wird die Leitfähigkeit der Luft heruntergesetzt, der Sensor reagiert und löst aus.

Diese Art von Rauchmelder ist veraltet und wird aufgrund ihrer Strahlungseigenschaft durch neuere Technologie ersetzt [5].

Der Preis eines Ionisationsrauchmelders liegt zurzeit bei 12,90 Euro [6].

B. Der Wärmemelder

Der Wärmemelder enthält einen Temperatursensor zur Bestimmung der anliegenden Temperatur. Bei dieser Art von Meldern gibt es verschiedene Sorten, die sich in der Art und Weise unterscheiden, wie eine Alarmauslösung geschieht. Sie enthalten einen temperaturabhängigen Widerstand, der je nach eingestellter Maximaltemperatur ein Signal durchschaltet.

Nachteil dieser Melder ist, dass sie langsamer auf Entstehungsbrände reagieren als Rauchmelder, da aufsteigender Rauch bereits abgekühlt ist, bevor er bei dem Wärmemelder ankommt. Sinn machen Wärmemelder allerdings genau dort, wo Rauchmelder nicht eingesetzt werden können, z.B. in Küchen, in denen der Wasserdampf einen Fehlalarm auslösen könnte, oder in Garagen und Produktionsstätten, in denen z. B. Schweißarbeiten durchgeführt werden [7].

Die Kosten für einen solchen Melder liegen bei 10,00- 15,00 Euro [8].

C. Der Flammenmelder

Der Flammenmelder reagiert, wie der Name bereits sagt, auf durch Flammen emittierte elektromagnetische Strahlung. Um Störungen durch z.B. UV Licht zu verhindern, arbeiten Flammenmelder in der Regel im ultravioletten oder infraroten Bereich. Im Allgemeinen werden diese Brandmelder in einer Zweimelderabhängigkeit realisiert um Fehlalarme zu vermeiden. Entweder werden dabei ein ultravioletter und ein infraroter Flammenmelder kombiniert, oder es wird von zwei verschiedenen Standorten auf ein Objekt gezielt.

Nachteil bei diesen Brandmeldern ist, dass sie eine direkte Verbindung zum Brandobjekt benötigen und gleichzeitig keine starke Rauchentwicklung vorhanden sein darf [9].

Preislich ist ein solcher Flammenmelder relativ teuer. Ein Exemplar kostet zwischen 200-250 Euro [10].

D. Der optische Rauchmelder

Der optische Rauchmelder hingegen ist relativ preiswert. Er arbeitet mit einer Messkammer, in der in regelmäßigen Abständen Lichtstrahlen ausgesandt werden, die im Normalfall nicht auf die sich an einer bestimmten Stelle der Messkammer befindliche Fotozelle treffen. Gelangt nun Rauch in die Messkammer, bewirkt dieser, dass die Lichtstrahlen gebrochen bzw. gestreut werden. Dadurch gelangen die gestreuten Lichtstrahlen u.a. auch an die Fotozelle. Dadurch entsteht ein Signal und der Melder wird ausgelöst. Der optische Rauchmelder ist der am häufigsten anzutreffende Rauchmelder in deutschen Haushalten.

Ein wesentlicher Grund dafür ist der Preis [11].

Der optische Rauchmelder kostet zwischen 4 und 13 Euro ohne Vernetzung.

Da dieser Rauchmelder der aktuell am häufigsten in deutschen Haushalten vorkommende ist, und dieser zudem am günstigsten auf dem Markt zu erhalten ist, haben wir diesen Rauchmelder in unserem Projekt genutzt, um ihn mit weiteren Rauchmeldern derselben Art zu vernetzen.

E. Vernetzte Brandmelder

Der Unterschied von vernetzten und nicht vernetzten Brandmeldern ist recht schnell und

einfach erklärt. Ein Brandmelder, unabhängig von welchem Typ, kann für sich alleine stehen oder verbunden mit anderen sein. Die Idee die sich hinter der Vernetzung verbirgt ist die, dass Brandmelder, die sich nicht direkt am Brandherd befinden, ebenfalls auslösen, sobald derjenige Alarm schlägt, der „vor Ort“ ist.

Sinnvoll bzw. sogar notwendig ist dies im Kleinen bei Bränden in Häusern, in denen unter Umständen der Brand im Erdgeschoss ausbricht, die Familie aber im ersten oder zweiten Geschoss schläft und den auslösenden Melder nicht hören kann. Im Großen kann es sinnvoll sein, wenn wir von großen Gebäudekomplexen reden, wo der auslösende Rauchmelder möglicherweise nicht weit genug durch mehrere Wände zu hören ist. Wie oben bereits erwähnt, ist das Entscheidende bei Feuermeldern ja, dass sie gehört werden, um Personen vor dem drohenden Feuer zu warnen.

Es gibt zwei Möglichkeiten Feuermelder zu vernetzen. Entweder eine kabelgebundene oder eine kabellose Vernetzung. Die kabelgebundene Vernetzung beinhaltet, wie bereits das Wort verrät, die Vernetzung der Feuermelder durch Kabelverbindungen. Die kabellose Vernetzung ist über verschiedene Schnittstellen wie zum Beispiel Funk und WLAN möglich. Diese Art der Vernetzung ist in der Regel aber mit einem größeren Kostenaufwand verbunden. U.a. deshalb werden wir in unserem Projekt eine Vernetzung mittels Kabel verwenden. Ein optischer, über Funk vernetzter, Rauchmelder kostet bei der Elektronik-Discount-Firma Reichelt 49,95 Euro pro Stück [12]. Geht man davon aus, dass wir ein Schlafzimmer, zwei Kinderzimmer, und einen zentral gelegenen Flur in einem fiktiven Haus haben, benötigt man insgesamt vier der angesprochenen Rauchmelder (4 x 49,95 Euro = 199,80 Euro). Als unmittelbare Defizite bleiben hier festzuhalten, dass dieses System keine externen Meldungen nach außen absetzen kann, dass mangels Schaltzentrale keine Information über das spezifische Gerät auflaufen kann, das den Alarm auslöst, und schließlich auch, dass die räumlichen Gegebenheiten so beschaffen sein müssen, dass die Funksignale ungehindert durchkommen. Kabelbindung und zentraler Knoten, der Signale nach außen senden kann, machen also Sinn.

V. Der Raspberry Pi

Der Anfang 2012 auf dem Markt gekommene Raspberry Pi ist ein Einplatinencomputer. Sein wesentlicher Vorteil gegenüber üblichen Personal-Computern besteht einerseits in seinem signifikant geringeren Preis (um die 30 € der „nackte“ Pi in der am weitesten entwickelten Version B+), andererseits in seiner geringen physikalischen Größe (nahezu Kreditkartenformat). Ansonsten könnten prinzipiell alle nachstehend beschriebenen Prozesse und Funktionen auf jedem anderen, auch älteren PC ausgeführt werden, sofern dieser über Netzwerk- , Tastatur und Mauszugang sowie mindestens 512 MB Arbeitsspeicher und 4 GB Festspeicher verfügt.

Alternativ könnten zudem auch Konkurrenzprodukte zum Raspberry wie z.B. der Arduino oder Homematic in Erwägung gezogen werden. Beim Arduino ergibt sich dabei zudem der Vorteil, dass er analoge Signale direkt verarbeiten kann und man sich so den A/D-Wandler sparen kann. Unter reinen Kostengesichtspunkten wäre also eine genaue Prüfung der Alternativen interessant.

Der von uns genutzte Raspberry Pi organisiert alle notwendigen Abläufe über bis zu vier USB-Zugänge, einen LAN-Anschluss, einen Micro-SD-Slot (bis 64 GB) und 512 MB RAM, Darüber hinaus ist eine CSI-Schnittstelle für Kameranutzung und ein Digital Serial Interface (DSI) für Lichtsteuerung vorhanden. Letztere beiden sind aber für unseren Zusammenhang nicht von Bedeutung.

Bedeutsam für uns ist hingegen die vorhandene GPIO-Schnittstelle mit ihren insgesamt 26 Pins, von denen wir sechs benutzen. Über diese können wir die von uns benötigte Hardware direkt an den Pi anschließen. Da das Gerät als Ganzes nicht im Mittelpunkt unseres Themas steht, wollen wir es bei diesem kurzen Überblick belassen. Für weitere Informationen siehe [13].

VI. Die Umsetzung

Ohne ein laufendes Arbeitsjournal geführt zu haben, können wir unsere Projektarbeit wie folgt zusammenfassen:

Zum einen haben wir an allen Pflichtveranstaltungen aktiv teilgenommen (erstes Treffen, Bibliotheksschulungsteilnahme, Konferenzteilnahme mit Betreuer).

Zum Anderen haben wir umfangreiche Recherchetätigkeiten und Informationssammlungsprozesse im Internet und Bibliothek durchgeführt.

Nachdem diese beiden Phasen abgeschlossen und ausgewertet waren, trafen wir unsere Entscheidungen für das geschilderte System und bestellten die erforderlichen Hardwaregegenstände für die geplante Umsetzung. Dies geschah auf Basis der weiter oben bereits geschilderten Reduzierung der Zielsetzung unseres Versuchsansatzes. Wir wollten eine überschaubare Fragestellung exemplarisch und möglichst erfolgreich testen und beschränkten uns deswegen auf einen Rauchmelder, seine kabelgebundene Verbindung zum Raspberrry Pi und auf die E-Mail-Benachrichtigung als Reaktion auf die Auslösung des Rauchmelders.

Zeitgleich begannen wir mit der Ausarbeitung.

Im letzten Kapitel unser Auswertung reflektieren wir in unserem Fazit das praktisch Erreichte vor dem Hintergrund der ursprünglichen Zielsetzung. Dabei gehen wir insbesondere auf die für uns klar auf der Hand liegenden Ausbaustufen ein, die u.E. aufgrund unserer getätigten praktischen Versuche prinzipiell ohne großen Aufwand zu realisieren sind.

Nachdem die benötigten Hardwarebauteile vorhanden waren, wurde der Prototyp entworfen und die Ausarbeitung, sowie eine Bau- und Benutzerhandbuch verfasst.

Alle weiteren konkreten Schritte unserer praktischen Arbeit sind dem Handbuch zu entnehmen.

VII. Fazit

Unsere praktische Umsetzung der Aufgabenstellung muss gegenüber unserer ursprünglichen, eher euphorisch zu nennenden Absicht als eher bescheiden genannt werden. Wir sind jedoch in enger Absprache mit unserem Betreuer vorgegangen und denken, insgesamt exemplarisch unsere Fragestellung beantwortet zu haben. Mittels Raspberry Pi und preiswerten optischen Rauchmeldern ist es möglich, kabelgebunden ein kooperatives, verteiltes Rauchmeldesystem aufzubauen, das den von uns oben formulierten Anforderungen genügen kann. Realisiert haben wir so ein System allerdings lediglich auf Basis eines einzigen Rauchmelders, dessen Reaktion eine externe Benachrichtigung per Mail auf z. B. ein Handy ist.

Wir haben uns in unserer praktischen Ausführung unseres Prototypen dafür entschieden, nur einen Rauchmelder und nicht vier (was möglich ist!) anzuschließen, da das zu Grunde liegende Prinzip, um weitere Rauchmelder nachzurüsten, das gleiche ist. Sobald wir in der Lage sind, das Signal des einen Rauchmelders mit den Raspberry Pi aufzunehmen und zu verarbeiten, ist jede Art der weiteren Nutzung des Signals möglich. Zum prinzipiellen Nachweis der Aufnahme des analogen Signals des Rauchmelders, seiner Umwandlung in ein digitales und seiner Verarbeitung durch den Pi haben wir uns für den E-Mail-Versand entschieden.

Der Anschluss von vier Rauchmeldern an den Pi würde lediglich eine Selektion im Programmcode erfordern, an welchen Pins das Signal ankommt. Damit steht der Ort des Brandes fest und könnte mit der Mail nach außen geleitet werden. Gleichermaßen wären in diesem Augenblick die anderen drei, nicht betroffenen Melder identifiziert und könnten durch ein entsprechendes Signal des Pi's ausgelöst werden. Der dafür benötigte Code ist überschaubar.

Unsere Lösung erscheint uns allerdings in Altbauten realistischerweise nur eingeschränkt einsetzbar, da sie kabelgebunden ist. Dies bringt für alten bestehenden Wohnraum einen relativ hohen Nachrüstaufwand mit sich (einschließlich notwendig werdender Renovierungen). Bezüglich Neubauten ist das anders. Hier und auch bei Bauten, die nicht allzu alt sind, oder auch für große Hallen bzw. Firmengebäude, in denen die ästhetische Frage keine so große Rolle spielt, erscheint uns unsere Lösung durchaus anwendbar.

In solchen Gebäuden sind in der Regel heutzutage bereits kleine Zwischendecken vorhanden, in denen teilweise schon Kabelkanäle verlaufen. In jedem Fall aber können die Kabel verstaut

werden. Vereinzelnd gibt es diese Zwischendecken auch in älteren Bestandsbauten mit Rigips oder OSB-Platten. Bei Neubauten ist zu erwähnen, dass unser System in die Bauplanung mit aufgenommen werden kann, sodass es direkt mit in den Neubau integriert werden kann.

Grundsätzlich ist festzuhalten, dass die meisten bereits existierenden Brandmeldeanlagen in Gebäuden kabelgebunden sind. Hier ist einfach die größere Zuverlässigkeit ausschlaggebend.

Bei Funk ist der Aspekt entscheidend, dass es zu Störungen in Form von Ausfällen und Reichweiteproblemen kommen kann. Eine Alternative wären WLAN-fähige Brandmelder, die allerdings der Markt bisher nicht handhabbar anbietet. Und jedem Melder einen Pi mit WLAN-Adapter zu 'spendieren' erscheint uns zwar eine Lösung, dürfte aber zu teuer sein.

Wichtig zu erwähnen ist, dass die Kosten für unser Produkt mit der Anzahl der Rauchmelder und in diesem Zusammenhang mit der Größe des Gebäudes stark ansteigen. Grund dafür sind allein die Kabelkosten, die bei großen Gebäuden sehr ins Gewicht fallen, da man zudem stabilere als bei uns und fest verbundene Kabel nutzen sollte.

Fraglich erscheinen uns auch alternative Lösungen z. B. mit Homematic, mit der beispielsweise sowohl Rauchmelder, Licht und Haustüren geschaltet werden können. Diese Lösung ist sehr kostenintensiv und ist mit dem Raspberry Pi prinzipiell ebenso umsetzbar. Dies haben wir bereits damit gezeigt, dass wir in der Lage sind eine E-Mail mit dem Raspberry Pi zu versenden. Damit ist es nicht mehr fern ein Licht oder ein elektrisches Schloss zu steuern.

Zusammenfassend bleibt zu sagen, dass, egal für welche Lösung man sich entscheidet, mehr Geld in die Hand genommen werden muss, als zu Anfang von uns gedacht. Es handelt sich dabei um mehrere Hundert Euro. Je nachdem wie groß das Gebäude ist, können auch mehr Kosten anfallen.

LITERATUR

[1] "Aus den Bundesländern - Rauchmelder retten Leben." http://www.rauchmelder-lebensretter.de/home/gesetzgebung/aus-den-bundeslaendern/.

[2] "Statistiken und Fakten zum Thema Brandschutz - Brandschutz / Ratgeber." http://www.abus.com/Ratgeber/Brandschutz/Rauchtote-Statistik.

[3] "Zusammensetzung von Brandrauchrauchmelderpflicht.eu." http://www.rauchmelderpflicht.eu/gefahrenquelle-brandrauch/brandrauch-zusammensetzung/.

[4] "Anwendungsnorm DIN 14676 - Rauchmelder retten Leben." http://www.rauchmelder-lebensretter.de/fachberater/normen-richtlinien/anwendungsnorm-din-14676/.

[5] "Ionisationsrauchmelder – Wikipedia." http://de.wikipedia.org/wiki/Ionisationsrauchmelder.

[6] "First Alert SA300UK Ionisationsrauchmelder, mit Stille und Prüftaste: Amazon.de: Baumarkt." http://www.amazon.de/First-Alert-SA300UK-Ionisationsrauchmelder-Prüftaste/dp/B00EK04NOK.

[7] "Wärmemelder - Brandschutz - Brandmelder/Geräte - baunetzwissen.de." http://www.baunetzwissen.de/standardartikel/Brandschutz-Waermemelder_3181245.html.

[8] "Bavaria BARM6 Hitzemelder / Brandmelder, reinweiß: Amazon.de: Baumarkt." http://www.amazon.de/Bavaria-BARM6-Hitzemelder-Brandmelder-reinwei%C3%9F/dp/B004YK320W/ref=sr_1_4?s=diy&ie=UTF8.

[9] "Flammenmelder - Brandschutz - Brandmelder/Geräte - baunetzwissen.d." http://www.baunetzwissen.de/standardartikel/Brandschutz-Flammenmelder_3181503.html.

[10] "Flammenmelder bei Mercateo günstig kaufen." http://www.mercateo.com/kw/flammenmelder/flammenmelder.html.

[11] "Optische Rauchmelder - Rauchmelder retten Leben." http://www.rauchmelder-lebensretter.de/home/rauchmelder-anwendung/funktionsweise/optische-rauchmelder.

[12] "HM FRM nur 49,95 Euro : Funk-Rauchmelder bei Reichelt Elektronik." http://www.reichelt.de/Sensoren-Homematic/HM-FRM/3//index.html?ACTION=3&GROUPID=4785&ARTICLE=104631&OFFSET=16&.

[13] "What is a Raspberry Pi?." http://www.raspberrypi.org/.

Überwachung und Steuerung eines Aquariums

Moritz Morawietz

I. Einleitung

In diesem Projekt geht es um die Steuerung und Überwachung eines Aquariums mit Verwendung des Raspberry Pi. Die Aufgaben, welche übernommen werden, vereinfachen den Alltag des Aquaristen und erhöht die Lebensqualität der Fische. So wird das Licht statt wie üblich nicht über eine Zeitschaltuhr ein- und ausgeschaltet, sondern langsam gedimmt, und mit einem zusätzlichen RGB LED Streifen um eine Mondlichtimitation erweitert. Für den Aquaristen ist es durch den angeschlossenen Futterautomaten kein Problem mehr seine Fische zu füttern, sollte er mal nicht daheim sein. Die Steuerungsmöglichkeiten werden auf einem Webinterface bereitgestellt, sodass man weltweit darauf zugreifen kann.
Für die Überwachung werden Temperaturen an verschiedenen Stellen gemessen, der Wasserstand ermittelt und festgestellt, wann die Wasserheizung aktiv ist. Diese Daten werden Aufgezeichnet und auf dem Webinterface grafisch dargestellt. Zusätzlich können bei kritischen Werten Warnungen an den Benutzer gegeben werden.

Dieses Projekt ist nicht das erste seiner Art, es gibt bereits unzählige Hobbyprojekte. Diese basieren entweder auf Industrie Steuer/Regelsystemen [1], Arduinos[2], oder eben dem Raspberry Pi [3] [4]. Sie sind alle stark den jeweiligen Bedürfnissen angepasst, so auch bei meiner Umsetzung. Besonders an der hier vorgestellten Lösung ist die Nutzung eines Mikrocontrollers und Raspberrys in Kombination. Wir können die Vorteile beider Systeme nutzen, und auf gleich zwei Communitys zugreifen, und der Funktionsradius erhöht sich deutlich. So ist es zum Beispiel einfacher, einen Sensor mit dem Mikrocontroller auszulesen, dafür kann ein Webinterface auf dem Rasperry problemlos realisiert werden. Bemerkenswert ist auch die und Erweiterbarkeit des Systems. So ist unter anderem geplant, weitere Sensoren zur Überwachung der Wasserqualität zu installieren. Diese können eigenständig entwickelt und dann an den Mikrocontroller angeschlossen werden, ohne andere Teile der Schaltung zu beeinflussen. Auch eine komplette Automation eines Aquariums ist in den nächsten Jahren vorgesehen. In einem Neubau werden Leitungen zum Becken gelegt, und mit denen über Magnetventile ein Wasserwechsel implementiert wird. Die einzige Aufgabe des Besitzers bleibt dann das gelegentliche nachfüllen des Futtervorrates und, was ja Sinn eines Aquariums ist, genießen.

Im folgenden wird der Aufbau des Systems betrachtet, und dabei auf den Entwicklungsprozess eingegangen. Getrennt wird in Hardware, Software für den Mikrocontroller und das Webinterface inklusive Backend.

II. Aufbau des Systems

Unser System ist in zwei eigenständige Systeme getrennt. Das Hardwaremodul und den Raspberry Pi. Zur Kommunikation wird die Serielle Schnittstelle UART (Universal Asynchronous Receiver Transmitter) verwendet. Beide Teile, also sowohl der Raspberry als auch der verwendete Mikrocontroller haben diese in Hardware implementiert.

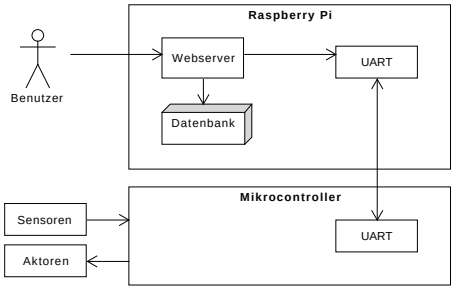


Fig. 1. Trennung der Systeme

III. Hardware

Die Hardware übernimmt die Aufgabe des Auslesens der angeschlossenen Sensoren, Steuern der Neonröhren Beleuchtung und LED-Streifens, sowie die Fütterung, wozu ein umgebauter kommerzieller Futterautomat genutzt wird. Im folgenden sind die Sensoren aufgelistet.

- Temperatursensoren, insgesamt werden 4 verwendet. Einer am Filterauslass, da die Heizung sich im Filter befindet. Zwei auf Unterschiedlichem Niveau im Becken, und der Letzte an der Luft, um die Raumtemperatur zu messen.
- Der Wasserstandssensor, welcher ein Luftdrucksensor mit ins Wasser geführtem Rohr ist. Durch den Wasserdruck steigt der Luftdruck im Rohr.
- Ein Fotowiderstand zur Überwachung der Heizung. Dies ist ein Hack um zu ermitteln, wann die Heizung aktiv ist. Dafür wird der Fotowiderstand an der Kontrollleuchte der Heizung befestigt und lichtdicht abgedeckt.

Das Zentrum der Schaltung ist der Mikrocontroller. An ihm sind die Sensoren und Aktoren wie eigene Module angeschlossen. Der code spricht diese Module bei Bedarf an.

A. Mikrocontroller

Die Wahl des Mikrocontrollers richtete sich nach den Mindestanforderungen. Der Flaschenhals waren die die zu steuernden LED-Streifen und das EVG, dafür werden 4 PWM-Kanäle benötigt. PWM (Pulse Width Modulation) ist ein weg, ein digitales Signal in ein Analoges, in diesem Fall Helligkeit,

zu wandeln. Viele Mikrocontroller besitzen Hardwaretimer mit PWM Kanälen. Endscheidend war hier also die Anzahl dieser Kanäle. Aus den Sensoren und den weiteren Aktoren lassen sich also folgende Mindestanforderungen ableiten.

- Vier PWM Kanäle für die Beleuchtung,
- Zwei Analog zu Digital Wandler für den Wasserstandssensor und die Überwachung der Heizung,
- Ein digitaler Pin für die Kommunikation mit den Temperatursensoren
- Eine Programmierschnittstelle zum Programmieren des Mikrocontroller im System (ISP)
- Ein Hardware UART

Bereits vorhandene Kenntnisse mit der Atmega-Serie von Atmel schränkten die Suche erheblich ein. Der kleinste Mikrocontroller dieser Serie mit 4 PWM Kanälen ist der Atmega16. Er erfüllt alle Anforderungen und lässt dabei noch Raum für zukünftige Erweiterungen. Für den Code hat der Atmega16 16kB Flashspeicher, einen Arbeitsspeicher von 1kB SRAM sowie einen Permanenten EEPROM von 512 Bytes zum speichern über einen Neustart hinaus[5]. Hier zeigt sich ein weiterer Vorteil des Atmega16. Sollte der Speicher für zukünftige Anwendungen nicht mehr reichen, kann man ihn mit einem vollständig Pin und Feature kompatiblen Atmega ersetzen, z. B. den Atmega32, welcher die doppelte Speicherkapazität besitzt.

Für eine vollständige Übersicht der Features des Atmega16 ist Seite 1 aus dem Datenblatt zu betrachten.

B. Stromversorgung

Die Stromversorgung teilt sich in zwei Spannungsbereiche auf; 5V und 12V. 5V sind für den Mikrocontroller inklusive Peripherie, die Sensoren und den Futterautomaten. Die 12V werden für den LED-Streifen und die 1-10V Schnittstelle des EVG genutzt. Realisiert werden die Spannungen über die Spannungsregler Bausteine 7805 und 7812, wobei der 7805 hinter den 7812 geschaltet ist. Der 7812, und somit die Versorgung des Boards erfolgt über ein 13,5V 1A Netzteil. Zur Glättung der Spannung und dem Filtern von Störungen wie kurzen Spannungsabfällen sind die Kondensatoren C1 bis C4 eingebaut. Die LED 2 zeigt, ob das Board Spannung hat. Da sie an 5V angeschlossen und die Spannungsregler in Reihe geschaltet sind, indiziert sie beide Spannungen. Die Diode D3 ist zum Schutz der Schaltung vor Verpolung der Anschlüsse.

C. Peripherie

Am Mikrocontroller sind für den Betrieb ein Quarz und eine Status LED angeschlossen. Der Quarz wird benötigt, da der interne Oszillator nicht genau genug für den UART ist. Bei der Wahl der Frequenz des Quarzes wurde darauf geachtet, das dieser mit dem Mikrocontroller Kompatibel ist. Eine Frequenz von 8MHz ist in diesem Bereich.

Die Status LED ist zur Indikation von verschiedenen Zuständen nutzbar, wie z.B. Kommunikation oder Auslesen einer der Sensoren. Während der Entwicklungsphase ist sie auch hervorragend als Debugausgabe zu nutzen, indem man sie blinken lässt. Einen weiteren Vorteil beim Testen der

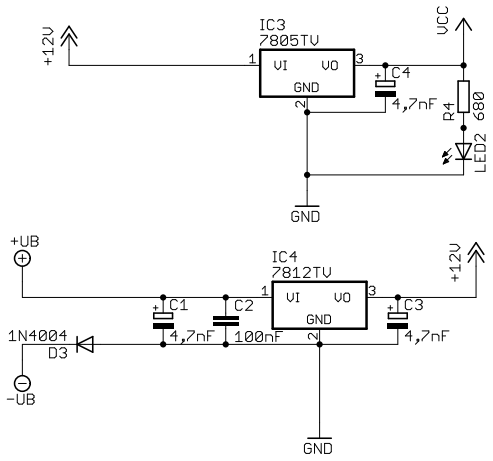


Fig. 2. Stromversorgung

Software bietet der Reset-Taster. Dieser startet das Programm im Mikrocontroller neu und setzt die Register neu.

Wie in Abbildung 3 zu sehen, ist außer dem der Widerstand R1 verbaut, welcher den Reset Pin auf High hält. Dies ist wichtig, da sonst für jeden Programmiervorgang via ISP die Stromversorgung unterbrochen werden müsste.

Für den Analog zu Digital Wandler sind Pin 30 bis 32 zuständig. ARef ist über C5 an GND angeschlossen, da eine interne Referenz verwendet wird. C5 ist zur Glättung der internen Referenz. Statt AVCC wird die interne Versorgung benutzt. [6] [**?**].

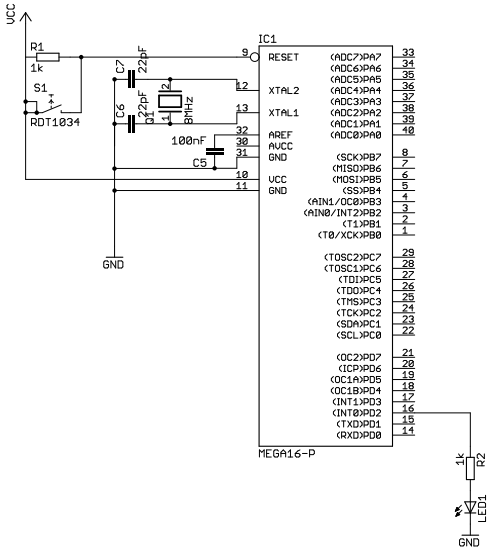


Fig. 3. An den Mikrocontroller angeschlossene Peripherie

D. UART Schnittstelle

Der UART ist eine universelle, asynchrone, serielle Schnittstelle. Seriell bedeutet Daten werden Bit für Bit übertragen. Asynchron bedeutet, das keine Clockleitung existiert, welche zeigt wann ein Bit gesendet wird. Aus diesem Grund benötigt man auch den Quarz, da dieser die erforderliche Genauigkeit liefert. Der UART benötigt 3 Leitungen; GND,

Rx und Tx. Das im Mikrocontroller verbaute UART Modul hat diese Ports, der Pegel dort liegt allerdings bei 5V, der Raspberry nur 3,3V, und ist höheren Spannungen gegenüber empfindlich. Dieses Problem besitzt eine einfache Lösung. Für die Empfangsrichtung reicht eine Diode zum Schutz des Raspberry vor Überspannungen, sollte versehentlich der Ausgang am Microcontroller auf High, also 5V geschaltet werden. Der AVR erkennt einen High-Pegel ab $\frac{Vcc}{2} = 2,5V$[**?**]. Für die Senderichtung wird ein einfacher Spannungsteiler mit einem Verhältnis von $\approx 6 : 4$ eingesetzt, so entstehen ca 3V für den Raspberry bei eingeschaltetem High-Pegel. Das Hardware UART Modul des Raspberry Pi's ist über die GPIO pins zu verbinden. Dabei wird Rx des RPis an Tx des Mikrocontrollers angeschlossen und umgekehrt.[7]

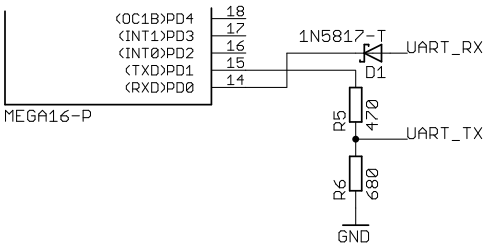


Fig. 4. UART zum Rasperry

E. LED Streifen

Der RGB-LED-Streifen ist eine flexible Platine, auch Flat Flex genannt, mit aufgelöteten dreifarbigen LEDs und zugehörigen Vorwiderstand. Anzusteuern ist er über seine 4 Pins, 12V sowie je ein GND für jede Farbe. Die Ansteuerung ist durch die hohen Ströme nicht direkt durch den Mikrocontroller möglich. Zum Treiben der LEDs werden MosFETs verwendet. Diese können Ströme bis zu einem Ampere ab, somit ausreichend für die LEDs mit je Farbe ca. 0,3A. Die MosFETs enthalten auch eine Freilaufdiode. Diese ist erforderlich, da sich der LED Streifen aufgrund der Länge wie eine Spule verhält, und so beim Ausschalten durch Induktion eine Gegenspannung entsteht, welche dann frei über die Diode fließt, statt die Schaltung zu belasten. Die Helligkeit jeder Farbe wird über ein PWM Signal des Microcontrollers gesteuert. Obwohl LEDs nicht nachleuchten, ist eine Glättung des Signals nicht nötig. Die Frequenz ist so hoch, das die einzelnen Pulse für das menschliche Auge (oder in diesem Fall das fischige Auge) nicht zu unterscheiden sind. Beim Betrieb des LED Streifens entsteht zusammen mit der restlichen Schaltung einen gemessenen Strom von 0,95A. Somit entsteht eine Verlustleistung von $(13,5V - 12V) * 0,86A = 1,29W$, welche in Form von Wärme abgegeben wird. Bei dieser Menge wird ein Kühlkörper am Baustein 7812 benötigt. [8] Beim Einschaltvorgang können Spannungsabfälle entstehen, die hoch genug sind um den Mikrocontroller abstürzen zu lassen. Um diese zu kompensieren sind C11 bis C13 verbaut.

F. OneWire, Temperatursensor

Die Wahl der Temperatursensoren erwies sich als schwieriger als angenommen. Es gibt viele Verschiedene

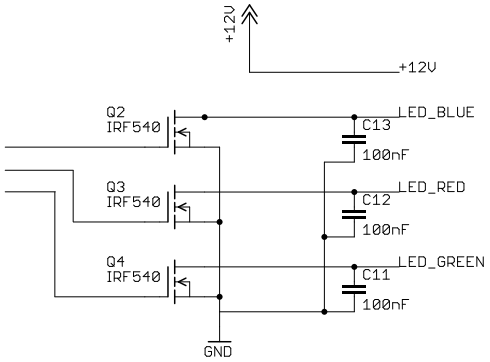


Fig. 5. MOSFETs für den LED-Streifen

mit unterschiedlichsten Eigenschaften. Sie Teilen sich hauptsächlich in Analoge und Digitale auf. Die analogen sind meist variable Widerstände, die simpelste und günstigste Art. Der Widerstand korrespondiert mit der Temperatur. Es sind auch ICs auf dem Markt, welche eine Spannung je nach Temperatur ausgeben. Nachteil dieser Sensoren ist, das sie einen ADC benötigen und meistens nicht geeicht sind, das bedeutet, dass jeder Sensor einzeln eingestellt werden müsste. Digitale Sensoren sind ICs welche die Temperatur über eine digitale Schnittstelle ausgeben. Das ist meistens ein Bussystem, sodass mehrere Sensoren an die gleichen Leitungen angeschlossen werden können. Die Anforderungen an unseren Temperatursensor sind Folgende:

- Eine Auflösung von ungefähr 0,1°C
- Eine Genauigkeit von ca. 0,5°C
- Keine Eichung nötig
- Messbereich muss 15°C bis 30°C umfassen
- Wenn digital, dann ist eine vorhandene Bibliothek für den Atmega gewünscht.
- Günstiger Preis, da wir 4 Messpunkte haben, sind >10€ Sensoren nicht wünschenswert

Nach langem hin und her wurden zuerst die Analogen ausgeschlossen, da diese zu viele Nachteile mit sich bringen. Die Wahl fiel dann auf den DS18B20 von Dallas Semiconductors, welcher bei Reichelt für 1,60€ erhältlich ist[9]. Er ist mit einer Genauigkeit von 0,0625°C sogar noch besser als die Anforderungen. Das bei den Thermometern verwendete OneWire Protokoll unterstützt bis zu 100 Sensoren an der gleichen Leitung, ohne das weitere Verschaltung an den Sensoren nötig wäre (Im Gegensatz zum I²C Bus, dort müssen Adresspins geschaltet werden).
Das OneWire Protokoll benötigt insgesamt 3 Adern. Gnd und Vcc für die Stromversorgung (5V) und eine Datenleitung. Der Ruhezustand der Datenleitung ist High, dafür wird der Pullup R12 eingesetzt. Die Datenleitung wird mit dem Microcontroller verbunden.

G. Vorschaltgerät

Normale Vorschaltgeräte lassen sich nicht dimmen. So auch die zwei ursprünglich in die Beleuchtung eingebauten Geräte. Daher müssen diese ausgetauscht werden. Es existieren dimmbare EVGs und welche mit eingebauter Schnittstelle. Um das

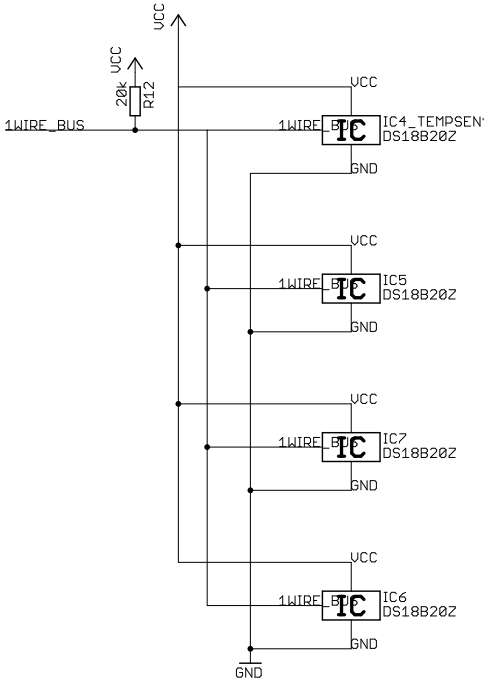


Fig. 6. Ansteuerung der Temperatursensoren

System einfach zu halten, verwenden wir die mit integrierter Schnittstelle.

Das gewählte Vorschaltgerät besitzt eine 1-10V Schnittstelle für die Helligkeit. Bei 10V ist die Helligkeit am höchsten, bei 1V am geringsten, jedoch nicht aus. Um die Beleuchtung vollständig abzuschalten, ist in der Versorgungsleitung zum EVG ein Relais in Reihe geschaltet. Das Relais braucht, wie die LED Streifen, eine Freilaufdiode parallel zur Spule des Relais, um Induktionsspannungen nicht auf die Schaltung wirken zu lassen. Das Relais ist zusammen mit der Diode und Sockelleisten zur Verkabelung auf einer externen Platine. Auch die 1-10V werden über diese Platine geleitet, um die Anschlüsse an einem Ort zu haben.

Im ausgeschalteten Zustand ist das Relais leitend, so kann bei einem Ausfall der Schaltung auf eine Zeitschaltuhr zurückgegriffen werden.

Um die 1-10V zu erzeugen wird auch hier ein PWM Signal von 0-5V erzeugt und anschließend verstärkt. Im Gegensatz zu den LEDs ist hier eine Glättung des Signals erforderlich, Test haben gezeigt, dass das EVG ohne Glättung keine konstante Helligkeit behält. Geglättet wird mit einem dreistufigen Tiefpassfilter, R7-9 und C14-16. [10].

Die Verstärkung wird über einen Operationsverstärker mit einer Verschaltung die zu einem Gain von 2 führt, wird daraus ein Signal von ca 0V bis 10V erreicht. die 0V werden hier nicht erreicht, da der Operationsverstärker dies nicht zulässt. Das EVG hat bei einer Spannung von < 1V die geringste Helligkeitsstufe, deswegen ist es nicht tragisch, dass der Bereich 1-10V unterschreitet.

H. Wasserpegel-,Drucksensor

Der Wasserpegelsensor war eine weitere Herausforderung. Die Anforderungen sind simpel, Änderungen im Millimeterbereich der oberen 5cm feststellen. Jedoch zeigte sich bei den Recherchen, dass zwar eine Vielzahl an Sensoren existiert, diese aber entweder zu ungenau, oder zu anfällig gegenüber Störungen sind. Kapazitive Pegelsensoren können zum Beispiel durch Algenbewuchs beeinflusst, Schwimmer durch Pflanzen.

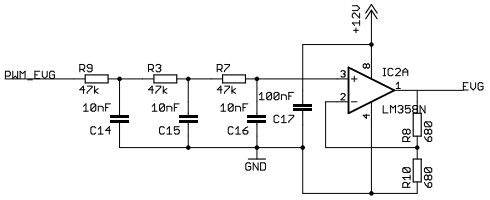


Fig. 7. Tiefpassfilter und OpAmp für die 1-10V schnittstelle des EVG

Die Art von Sensor, welche sich durchgesetzt hat sind Luftdrucksensoren. Ein Rohr wird an einem ende mit dem Sensor verbunden und senkrecht ins Wasser getaucht. Das Wasser drückt ins Rohr, so entsteht ein Druck, welcher dann gemessen werden kann.

Drucksensoren sind zwar noch nicht allzu verbreitet, jedoch fand sich eine Auswahl bei Reichelt. Ein Meter Wasser wirkt einen Druck von ungefähr 10kPa. Mit einem Messbereich von 0 bis 4 kPa kann also der Pegel der gesamten Beckenhöhe erfasst werden. Der Drucksensor MPXV5004 umfasst genau diesen Bereich.

Zur Ansteuerung hat der Sensor drei Pins. Neben der Stromversorgung gibt es einen Ausgangspin, der eine Analoges Signal von 0-5V ausgibt, je nach Druck am Sensor. Er kann somit direkt an einen ADC Pin des Mikrocontrollers gelegt werden. Die umliegenden Kondensatoren dienen der Störungsfilterung. [11]

Das Rohr wird gleichzeitig zur Befestigung der Temperatursensoren in verschiedenen Höhen verwendet.

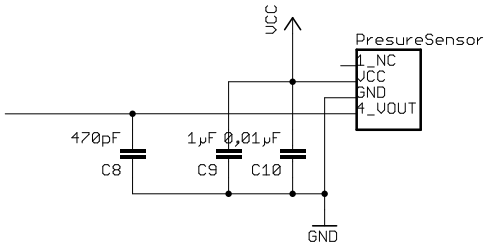


Fig. 8. Drucksensor mit störungsfilternden Kondensatoren

I. Heizungsüberwachung

Die Heizungsüberwachung erfolgt über die Kontrollleuchte des im Filter integrierten Heizers. Dazu wird ein Fotowiderstand vor der Leuchte befestigt und abgedeckt. Über den Spannungsteiler 9 wird eine Spannung erzeugt, welche vom ADC des Mikrocontrollers gemessen wird. Der Fotowiderstand wird über 2 Drähte am Board befestigt

IV. Software Microcontroller

Die Mikrocontroller Software wurde mit C programmiert. Zum Compilieren wird der avr-gcc verwendet. zum entwickeln

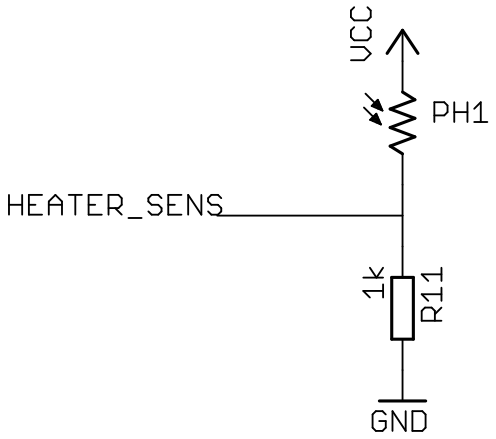


Fig. 9. Spannungsteiler mit Fotowiderstand zum ermitteln des Heizerstatus

der Atmegas existieren einige Bibliotheken für AVRs. Unter der Verwendung dieser war die Entwicklung ein einfacher Prozess.
Die Software auf dem Mikrocontroller sendet keine Daten ohne Aufforderung. Er wartet auf Befehle und führt diese dann aus.

A. Das Protokoll

Zur Kommunikation zwischen Mikrocontroller und Raspberry musste ein Protokoll festgehalten werden. Da Strings variabler Größe auf einem Mikrocontroller zu Problemen führen (es fehlt eine Speicherverwaltung) ist die Befehlslänge fest auf 4 ASCII Zeichen gesetzt. Dies ist auf dem Raspberry kein Problem, es wurde hier nur darauf geachtet, dass die gesendeten Daten leicht ausgewertet werden können.
§§§ Steht im Folgenden für eine dreistellige Nummer von 000 bis 255.

1) Raspberry zum Board: Tabelle I beschreibt die Befehle, die der Mikrocontroller versteht.

TABLE I
PROTOKOLL RASPBERRY ZU MIKROCONTROLLER

Befehl	Bedeutung
r	Das einzelne Zeichen setzt den Zeichenzähler zurück, sodass die nächsten 4 Zeichen einen Befehl bilden
L§§§	Setzt die Helligkeit der Neonröhren. Ist §§§=000, so wird das Relais gesetzt, das Licht ist dann aus.
[R/G/B]§§§	Helligkeit des LED Streifens für jede Farbe. R für Rot, G für Grün und B für Blau.
VREQ	Abfrage (Polling) der Werte. Dies fordert den Mikrocontroller dazu auf, alle Sensoren auszulesen und die Daten zu senden.
FEED	Startet die Fütterung

2) Board zum Raspberry: Tabelle II beschreibt die Daten, die der Mikrocontroller versendet.

B. Der UART

Das UART Modul wird beim Start des Controllers ko rt. Dafür werden in den Kontrollregistern die entsprechenden

TABLE II
PROTOKOLL MIKROCONTROLLER ZUM RASPBERRY

Inhalt	Bedeutung
T:$ID:$val	Eine Temperaturmessung. $ID ist die 64 Bit Identifikation des Sensors, $val der Wert mit 4 Stellen vor, und 4 Stellen nach dem Komma. Der Wert ist in °C
W:$val	Messung des Drucksensors. $val ist eine 10 Bit Nummer
H:$stat	Status des Heizers. $stat ist entweder "ON" oder "OFF"

Bits gesetzt. Die Konfiguration beinhaltet die Baudrate(9600), Anzahl der Datenbits (8), Anzahl der Stopbits (1) und Parity(None).
Zur Berechnung der Werte welche in die Baudratenregister geschrieben werden, stellt die Bibliothek "util/setbaudrate" Makros bereit, die zur Compilierzeit ausgeführt werden.
Bei jedem vollständigen Byte löst der UART Controller ein Interrupt aus. In der Interrupt-Methode wird der 4 Stellige befehl zusammengesetzt, und, wenn er komplett ist, ausgeführt.

C. Temperatursensoren

Die Temperatursensoren werden über das OneWire Protokoll angesteuert. Hierfür wurde ein bestehendes Projekt von Peter Dannegger eingebunden und modifiziert.
Die Sensoren müssen, bevor sie Ausgelesen werden können, erst einmal die Temperatur konvertieren und speichern. Dazu benötigen sie ca. 750ms. Um bei einem Pollbefehl möglichst schnell ein Ergebnis zu erhalten, sendet der Mikrocontroller jede Sekunde eine Konvertieraufforderung an die Sensoren. Nach einer Sekunde prüft er dann, ob in der Zwischenzeit ein poll kam. ist dies der Fall, liest er die Sensoren aus und sendet die Ergebnisse an den Raspberry.

D. Drucksensor

Der Drucksensor liefert permanent eine Spannung zwischen 0V und 5V. Diese muss vom internen ADC konvertiert werden. Um Störungen auszugleichen, werden mehrere Messungen ausgeführt und ein Mittelwert gebildet.
Bei der Initialisierung des Mikrocontrollers werden die Register des ADC auf die benötigten Einstellungen gesetzt. Diese beinhalten die Spannungsreferenz, aktiviert den ADC und führt eine Erste Messung durch. Durch diese Messung wird sichergestellt, das keine fehlerhaften Werte in den Registern stehen.

E. PWM der LED-Streifen und des Vorschaltgerätes

Der Mikrocontroller besitzt 4 Hardware Timer mit integriertem PWM Modul. Nachdem die Module eingestellt wurden, das heißt Taktvorteiler und den PWM Modus, kann man über setzen eines einzigen Registers die Pulsweite einstellen. Die beiden 10Bit Timer werden in einen 8Bit Modus versetzt, damit sind auch sie mit einem Wert von 0 bis 255 vollständig auflösbar.

V. Rasperry Software

Auf dem Raspberry ist ein Webserver installiert, auf dem ein Servlet läuft, und das Webinterface bereit stellt. Das Servlet übernimmt dabei das Sammeln von Daten, es sendet Poll Befehle und speichert die Sensordaten in einer Datenbank.
Die erste Idee war, das Webinterface mit dem auf Python basierendem Framework "Django" zu entwickeln. Dies lag nahe, da auf dem Raspberry Pi Python eine bevorzugte Sprache ist. Diese Idee wurde jedoch verworfen, da die nötigen Kenntnisse mit Django fehlten. Stattdessen nutzen wir GWT, ein von Google entwickeltes Toolkit, mit dem man in Java interaktive Webseiten entwickelt. Das GWT-Servlet läuft in einem Tomcat Server, welcher auf dem Raspberry installiert ist.

A. Kommunikation mit dem UART

Da Java eine Plattform unabhängige Sprache ist, gestaltet sich die Kommunikation mit der Hardware, in diesem Falle unsere Serielle Schnittstelle als umständlich. Es Existiert zwar eine Java Bibliothek javax.comm mit einer entsprechenden Implementierung für Linux, gnu.io, diese erfordert aber eine manuelle Installation im System des Raspberry, und bringt weitere Anfälligkeiten mit sich. Deswegen nutzen wir Javas Fähigkeiten, ein externes Programm zu starten. Um auf die Serielle Schnittstelle zuzugreifen, startet das Servlet separate Python Scripte, eines zum simplen senden von Strings, und ein anderes zum empfangen. Diese Scripte verwenden das Python Modul python-serial. Damit dieses Zugriff auf die Schnitstelle hat, sind einige Anpassungen notwendig[12] und der Server muss als Root laufen. Das ist auf lange Sicht ein Sicherheitsproblem und muss behoben werden.
Das Empfangsscript wird in einem separaten Thread gestartet und ausgelesen. Von dort aus Werden die Daten dann mit einem Zeitstempel versehen und in der Datenbank gespeichert.

B. Datenbank

Um die Installation einfach zu halten, verzichten wir auf eine externe Datenbank, welche erst installiert werden müsste. Stattdessen wird eine SQLite DB verwendet, diese besteht nur aus einer Datei. Das Servlet greift über entsprechende Treiberpakete unter Benutzung von SQL auf die Datenbank zu. Verwendet werden 3 Tabellen, für jeden Sensortyp eine. Es werden Uhrzeit und Wert gespeichert Die Temperatursensoren teilen sich eine Tabelle, sie besitzt zusätzlich eine ID Spalte.

C. Weboberfläche

Die in GWT gestaltete Weboberfläche umfasst 3 Bereiche. Die direkte Steuerung,hier kann der LED Streifen bedient werden, eine Fütterung ausgelöst oder das Licht abgeschaltet werden. Dann die Einstellungsmöglichkeiten für die zeitlich gesteuerten Ereignisse wie Fütterungen oder Lichtschaltungen. Als dritten Bereich die Anzeige der aktuellen und aufgezeichneten Werte wie Temperaturen, Wasserlevel und Heizstatus.
Zur besseren Bedienbarkeit werden für die Einstellungen der Helligkeit Slider verwendet. Diese werden als ein Modul in GWT geladen.[13]
Die Darstellung der Aufgezeichneten Werte erfolgt über Google Charts, eine offizielle GWT API[14][15]. Sie kann Diagramme erstellen, welche man einfach mit Daten in Tabellenform füllen kann.

VI. Ausblick

Das vorhandene System wird nun erstmal getestet. Es ist an einem Wohnzimmeraquarium installiert und liefert munter Daten. Es ist geplant, Erweiterungen sowohl Software, als auch Hardwareseitig zu installieren. Die Überwachung der Wasserwerte beschränkt sich zur Zeit auf die Temperatur, wünschenswert sind noch PH-Wert, Leitwert und Ähnliches. Softwareseitig ergibt dann auch ein Benachrichtigungsservice Sinn, welcher bei Messungen im Kritischen Bereich über verschiedene Kommunikationskanäle Meldung gibt. Auch ist der Schritt zum vollständig automatisiertem Aquarium nicht mehr weit. Die Möglichkeiten stehen hier dank des Modularen Aufbaus weit offen.

References

[1] Forumbenutzer "mirk", “Aquariumsteuerung mit einer S7-1200,” http://www.meerwasserforum.com/index.php?page=Thread&threadID=56938, 2012, [Online; accessed 04.02.2015].

[2] Forumbenutzer "jener8tionx", “The Quantified Fish: How My Aquarium Uses Raspberry Pi,” http://forum.arduino.cc/index.php?topic=7807.0, 2009, [Online; accessed 04.02.2015].

[3] L. Orsini, “The Quantified Fish: How My Aquarium Uses Raspberry Pi,” http://readwrite.com/2014/03/04/raspberry-pi-quantified-fish-acquarium, 2014, [Online; accessed 04.02.2015].

[4] Forumbenutzer "Christian", “Aquariumcomputer Marke Eigenbau mit Raspberry Pi,” http://www.aquaristikfreaks.de/aquariumcomputer-marke-eigenbau-mit-raspberry-pi-t998.html, 2014, [Online; accessed 04.02.2015].

[5] Atmel Corporation, “Datenblatt atmega16,” 2010, revision 2466T–AVR–07/10. [Online]. Available: http://www.atmel.com/images/doc2466.pdf

[6] Mikrocontroller.net, “Avr-tutorial: Adc,” [Online; accessed 16.01.2015]. [Online]. Available: http://www.mikrocontroller.net/articles/AVR-Tutorial:_ADC

[7] elinux.org, “Rpi low-level peripherals,” [Online; accessed 19.01.2015]. [Online]. Available: http://elinux.org/RPi_Low-level_peripherals

[8] P. Bastian, H. Bumiller, M. Burgmaier, W. Eichler, T. Käppel, W. Klee, K. Kober, J. Manderla, J. Schwarz, O. Spielvogel, K. Tkotz, U. Winter, and K. Ziegler, *Fachkunde Elektrotechnik*. Europa Lehrmittel, 2008.

[9] Reichelt, “Temperatursensor ds18b20,” [Online; accessed 04.02.2015]. [Online]. Available: http://www.reichelt.de/DS-18B20/3/index.html?&ACTION=3&LA=446&ARTICLE=58169&artnr=DS+18B20&SEARCH=DS18B20

[10] Forumgast "Dirk", “Tiefpass mit Verstärker,” http://www.mikrocontroller.net/topic/62057#491759, 2007, [Online; accessed 17.01.2015].

[11] Freescale Semiconductor, “Datenblatt mpxv5004 drucksensor,” 2009, revision 12. [Online]. Available: http://cache.freescale.com/files/sensors/doc/data_sheet/MPXV5004G.pdf

[12] Blogger Henry, “Raspberry pi – serielle schnittstelle (rs232) nutzen,” [Online; accessed 04.02.2015]. [Online]. Available: http://www.henrykoch.de/raspberry-pi-serielle-schnittstelle-rs232-nutzen

[13] kiouri@gmail.com, “Rgwt sliderbar,” [Online; accessed 02.02.2015]. [Online]. Available: http://code.google.com/p/gwt-slider-bar/

[14] Google, “Google charts,” [Online; accessed 02.02.2015]. [Online]. Available: https://developers.google.com/chart/interactive/docs/gallery

[15] Google, “gwt-google-apis,” [Online; accessed 02.02.2015]. [Online]. Available: http://code.google.com/p/gwt-google-apis/wiki/VisualizationGettingStarted?tm=6

Kombinierte Säuglingsüberwachung mit dem Raspberry Pi

Jan Steffen Becker

Christoph Küpker

***Zusammenfassung*—In dieser Arbeit wird ein auf dem Raspberry Pi basierendes System zur Säuglingsüberwachung vorgestellt. Es werden Anforderungen an das Produkt definiert und deren Umsetzung in einen Prototyp detailliert beschrieben. Es folgt ein Vergleich des Prototypen mit kommerziell erhältlichen Produkten.**

***Keywords*—Raspberry Pi, Säuglingsüberwachung, SPI**

I. Einleitung

Die Unterstützung des Alltags durch Technologien nimmt in immer größerem Maße zu. Dies betrifft auch die Kindererziehung und -versorgung. Immer mehr Geräte sind zur Unterstützung der Eltern in den ersten Jahren der Kindererziehung erhältlich. Dies betrifft unter anderem Spielzeug zur Lernentwicklung, aber auch Geräte, die den Eltern den Alltag mit Neugeborenen erleichtern oder das Sicherheitsgefühl der Eltern verstärken sollen.

Das Babyphone, ein Gerät zur Überwachung der Geräusche eines Kindes aus der Entfernung, gehört mittlerweile zur Standardausstattung. Darüber hinaus gibt es aber noch weitere Geräte zur Fernüberwachung, z.B. um die Temperatur des Aufenthaltsraumes des Kindes zu ermitteln, das Kind per Kamera zu überwachen oder die Atemfunktion des Kindes überwachen zu können. Offensichtlich handelt es sich bei der Fernüberwachung von Neugeborenen um ein Thema von großem Interesse und mit Produkten aus diesem Bereich lässt sich, wie in Abschnitt III dargestellt, Umsatz generieren.

Ziel dieser Arbeit soll es sein, den Raspberry Pi in der Art einzusetzen, dass die Fernüberwachung eines Kindes mit den zuvor genannten überwachten Werten mit nur einem Gerät erfolgen kann.

In Abschnitt II erfolgt zunächst eine Aufstellung anderer wissenschaftlicher Arbeiten auf dem Gebiet der Kinderüberwachung, speziell der Atemfunktionskontrolle. In Abschnitt III werden kommerzielle Produkte dargestellt, durch deren Kombination sich die in Abschnitt IV aufgestellten Anforderungen erfüllen lassen. Die Architektur und die technische Umsetzung der Anforderungen mit dem Rasperry Pi sind in den Abschnitten V und VI beschrieben. Abschließend erfolgt in Abschnitt VII die Evaluation des entwickelten Prototypen und in Abschnitt VIII ein Ausblick auf weitere Möglichkeiten ausgehend von diesem Prototypen.

II. Verwandte Arbeiten

Ein wesentlicher Schwerpunkt ist die Überwachung der Atemfunktion des Säuglings. Dabei sind quantitative Werte (z.B. Atemfrequenz oder Atemvolumen) nicht von Interesse. Vielmehr gilt es, zuverlässig festzustellen, *ob* der Säugling atmet.

A. Medizinischer Hintergrund

Grund der Atemüberwachung ist es, das Risiko für den plötzlichen Kindstod (englisch *sudden infants death syndrome*, kurz SIDS) zu senken. Der plötzliche Kindstod beschreibt den plötzlichen Tod des Säuglings mit (noch) unbekannter Ursache [1]. Meist finden die Eltern den Säugling morgens tot im Bett auf, ohne dass zuvor Anzeichen für eine Erkrankung bestehen.

Auch wenn die Ursache des plötzlichen Kindstods ungeklärt ist, sehen viele Theorien einen Zusammenhang zwischen Atemstillständen (Apnoen) im Schlaf und dem Eintreten des plötzlichen Todes. Die Überwachung der Atemfunktion ist somit eine mögliche Strategie, um das Risiko eines plötzlichen Kindstods zu senken [2]. Der Nutzen des Heim-Monitorings ist allerdings nach wie vor umstritten. Die in [2] durchgeführte Studie beschreibt jedoch, dass die Atemüberwachung aufgrund von Fehlalarmen anfangs zwar zu einer erhöhten Belastung der Familien führt, nach einer Eingewöhnungsphase allerdings zumindest das Sicherheitsgefühl der Eltern stärkt. Die Autoren betonen jedoch, dass ein Heim-Monitor nur sinnvoll ist, wenn die Eltern zuvor geschult werden, im Alarmfall richtig zu reagieren, z.B. wenn nötig Reannimationsmaßnahmen am Kind durchzuführen.

Neben Apnoen im Schlaf sind weitere vermeidbare SIDS-Risikofaktoren bekannt [1]. Insbesondere

- Rauchen der Eltern während der Schwangerschaft und Stillzeit, insbesondere im Schlafzimmer des Kindes.
- Schlafen in Bauchlage
- Eine Überhitzung des Säuglings. Die ideale Raumtemperatur für das schlafende Kind beträgt 18 bis 19°C.

B. Strategien zur Atemüberwachung

In einer Arbeit von 1976 werden fünf verschiedene Ansätze beschrieben, die Atmung eines Säuglings kontaktlos zu überwachen [3]. Kontaktlos meint, dass die Geräte zur Überwachung nicht direkt auf der Haut des Kindes befestigt werden.

- *Luftmatratze:* Der erste Ansatz ist eine Art Luftmatratze, auf der der Säugling liegt. Durch die Bewegungen beim Atmen fließt Luft innerhalb der Matte. Diese bewirkt eine Kühlung an Sensoren, die sichtbar gemacht wird.

- *Drucksensor:* Der zweite Ansatz ist ein kapazitiver Drucksensor, der unter den Schulterbereich des Kindes gelegt wird. Die Atembewegungen bewirken eine Veränderung des Drucks auf den Sensor, wodurch sich dessen Kapazität ändert. Dies kann gemessen werden.
- *Magnetsensor:* Hier wird ein starker Permanentmagnet auf der Brust des Säuglings befestigt. Dieser bewegt sich bei der Atmung. Diese Bewegungen können mithilfe eines in der Nähe fest montierten Magnetometers gemessen werden.
- *Kapazitätssensor:* Dies ist eine sehr störanfällige Technik: Der Säugling befindet sich in der Nähe von oder zwischen zwei Elektroden. Gemessen wird die Kapazität zwischen diesen, die sich durch die Atembewegung ändert.
- *Radar:* Es wird eine stehende Radarwelle erzeugt, die an der Brust des Säuglings reflektiert wird. Die reflektierte Welle wird mithilfe zweier Detektoren gemessen. Die Wellenlänge liegt im Mikrowellenbereich. Der Output eines Detektors entspricht der Phasenverschiebung der Welle am Detektor. Da die Phasenlänge ca. 30mm beträgt, lassen sich so schon kleine Bewegungen sichtbar machen.

Bei dieser Arbeit wurden mit dem Radarsystem die besten Ergebnisse erzielt. Oftmals haben Eltern allerdings Einwände gegen den einsatz von Mikrowellen, auch wenn die Strahlenbelastung dabei weit unter den empfohlenen Richtwerten liegt. In diesen Fällen empfehlen die Autoren die Drucksensor-Methode.

Ein weiterer Ansatz wird in einer aktuellen Arbeit von 2014 beschrieben [4]. Hier werden vier Piezoelemente unter die Bettpfosten gelegt. Diese sind empfindlich genug, dass sowohl Atmung als auch Herzschlag überwacht werden können.

III. Kommerzielle Produkte

Ziel dieses Abschnittes ist es zu ermitteln, inwieweit das Angebot kommerzieller Produkte die Säuglingsüberwachung ermöglicht. Insbesondere interessiert hier die Frage, zu welchem Preis eine vollständige Überwachung gemäß des angestrebten Umfangs dieser Arbeit möglich ist, um später die Kosten für den entwickelten Prototyp mit den kommerziellen Produkten vergleichen zu können.

Zur Geräuschüberwachung werden überlicherweise sogenannte Babyphones verwendet. Diese bestehen meist aus einer Station die im überwachten Raum platziert wird und dort die Geräusche aufnimmt, und einem Elternteil der mobil im Akkubetrieb verwendet werden kann und die Geräusche aus dem überwachten Raum wiedergibt. Solche Geräte sind ab ca. 30 Euro [5] erhältlich. Teurere Geräte bieten beispielsweise höhere Reichweiten für die kabellose Übetragung, strahlungsärmere Sender, integrierte Temperaturüberwachung oder Gegensprechfunktionen.

Zur Temperaturüberwachung können neben Babyphonen in denen diese direkt integriert ist, auch einfache Wetterstationen verwendet werden. Solche sind bereits für unter 20 Euro [6] erhältlich und enthalten einen Sender, der im zu vermessenden Raum untergebracht wird und einen Empfänger, der die gemessenen Werte via Funk empfängt und anzeigt.

Die Bildüberwachung lässt sich entweder mit Babyphones mit integrierten Kameras oder mit WLAN-fähigen Kameras mit Nachtsichtfunktion umsetzen. Die Nachtsichtfunktion ist notwendig, da anzunehmen ist, dass der zu überwachende Raum während des Nacht- oder Mittagsschlafs verdunkelt ist. Die Babyphones mit integrierter Bildüberwachung haben gegenüber eigenständigen Kameras den Vorteil, dass das aufgenommene Bild direkt über den Elternteil des Geräts betrachtet werden kann.

Zur Atemüberwachung werden bei kommerziellen Produkten meist Sensormatten verwendet, die unter dem schlafenden Kind platziert werden. Diese Matten detektieren Gewichtsveränderungen, die durch die Atmung des Kindes verursacht werden und ermitteln daraus, ob das Kind noch atmet. Das Überwachungsgerät vom Hersteller Angelcare wird mit einer unverbindlichen Preisempfehlung von $89,99$ Euro vertrieben [7].

Der Hersteller Angelcare bietet neben reinen Atemüberwachungsgeräten auch kombinierte Geräte, wie sie im Rahmen dieser Arbeit entwickelt werden sollen. Das Gerät mit dem größten Funktionsumfang bietet Geräusch-, Bild- und Atemfunktionsüberwachung. Die Kamera unterstützt dabei auch Aufnahmen bei Dunkelheit. Die Atemfunktionskontrolle löst einen Alarm aus, sobald die Sensormatte, die unter dem Kind platziert wird, 20 Sekunden lang keine Atembewegungen wahrnimmt. Die aufgenommenen Daten können mit dem zugehörigen Elterngerät abgerufen werden. Die unverbindliche Preisempfehlung für ein solches Gerät liegt bei $299,99$ Euro. [8]

Aus dieser Übersicht kommerziell erhältlicher Produkte und ihrer Funktionaltitäten wird im nächsten Abschnitt der Anforderungskatalog an eine mit dem Raspberry Pi prototypisch zu entwickelnde Lösung erstellt.

IV. Anforderungen

Mit dem in dieser Arbeit entwickelten Prototypen sollen mithilfe des Raspberry Pi vier Hauptfunktionalitäten umgesetzt werden.

1) Geräuschüberwachung
2) Temperaturüberwachung
3) Kameraüberwachung mit Nachtsichtfunktion
4) Atemfunktionskontrolle

Die dazu vom Raspberry Pi zu messenden und auszuwertenden Daten sollen ohne direkte Kabelverbindung an ein geeigenetes Empfangsgerät übertragen werden können. Für die prototypische Umsetzung im Rahmen dieser Arbeit soll ein Webinterface auf dem Raspberry Pi bereitgestellt werden, dass von jedem internetfähigen Endgerät im selben Heimnetzwerk wie der Raspberry Pi abgerufen werden kann. Auf diesem Webinterface sollen die folgenden Daten und Auswertungen aus dem überwachten Raum ausgegeben werden können:

5) Wiedergabe des derzeit aufgenommenen Tones.
6) Wiedergabe des derzeit aufgenommenen Bildes.
7) Anzeige der derzeit gemessenen Temperatur.
8) Anzeige der letzten zwei Minuten der Messkurven der Atemfunktionsüberwachung.
9) Angabe, ob die Atemfunktion nach Einschätzung des Systems in Ordnung ist.

Darüber hinaus muss es eine Alarmfunktionalität geben, die Anwender auch dann über relevante Vorgänge im überwachten Raum informiert, wenn sie nicht das Webinterface vor sich haben. Dazu soll für die Umsetzung in dieser Arbeit eine Mail an den Anwender versendet werden, wenn einer oder mehrere der folgenden Punkte zutreffen:

10) Die Atemfunktion des Kindes wird vom System als kritisch bewertet.
11) Der im überwachten Raum herrschende Geräuschpegel übersteigt einen zuvor sinnvoll definierten Schwellwert.
12) Die Temperatur im überwachten Raum liegt außerhalb eines sinnvoll definierten Sollbereichs.

Nach Eingang einer solchen Mail kann der Anwender dann entweder das Kind direkt aufsuchen oder die direkte Bild- und Tonübertragung auf dem Webinterface benutzen, um den Zustand zu prüfen.

Darüber hinaus sollte es aber möglich sein, die erfassten Daten auch generisch über einen Webbrowser hinaus direkt an weitere Endgeräte zu übermitteln. Dabei sollen insbesondere folgende Möglichkeiten beachtet werden:

13) Übertragung von Bild- und Tonstreams aus dem überwachten Raum an geeignete Mediengeräte wie Software-Mediaplayer oder netzwerkfähige Funk- und Fernsehgeräte.
14) Direkte Alarmierung an ein Smartphone oder ein anderes geeigenetes Gerät mit direkter Übertragung von Bild und Ton aus dem überwachten Raum, beispielsweise durch einen Anruf mittels eines bild- und tonfähigen Instant Messengers.

Die technische Umsetzung dieser Anforderungen sowohl auf Software- als auch auf Hardwareseite ist in den folgenden Abschnitten beschrieben.

V. Architektur

Die grobe Architektur sieht folgendermaßen aus: Auf dem Raspberry-Pi läuft das Betriebssystem Raspbian mit einem Tomcat 8 als Webcontainer, in dem die eigentliche Anwendung – eine Webapplikation – ausgeführt wird. Kern der Anwendung ist zum einen die Datenerfassung via SPI, die in den nachfolgenden Abschnitten im Detail erklärt wird. Die Daten werden dann über ein Webinterface ausgegeben. Dazu dient ein einfacher WebSocket. Zusätzlich werden die Gewichtssensorwerte vom BreathingMonitor, der für die Atemüberwachung zuständig ist, ausgewertet. Wenn ein Atemstillstand erkannt wird, wird eine Nachricht an den Benutzer geschickt. Dazu steht momentan ein SMTP-Client bereit, der die Nachricht per e-Mail an den Anwender schickt. Dies ist in Abbildung 1 dargestellt.

Das Signal zur Bildüberwachung wird als MJPEG-Stream von einem NoIR-Kameramodul für das Raspberry Pi direkt an den Webclient ausgeliefert. Das spezielle NoIR-Kameramodul enthält im Gegensatz zum normalen Kameramodul keinen Infrarotfilter und ermöglicht somit Nachtaufnahmen, wenn eine Infrarotbeleuchtung eingesetzt wird. Dies ist jedoch nur unter Einsatz eines Raspberry Pi nicht möglich da damit

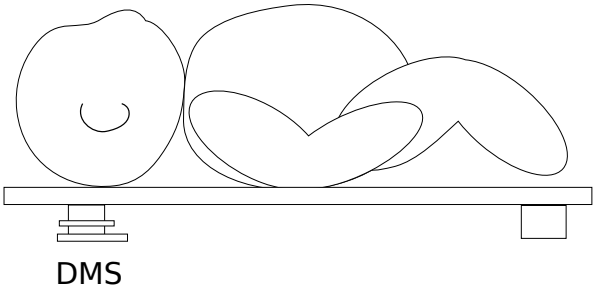


Abbildung 2. Aufbau des Prototyps. Die DMS befinden sich auf Kopfhöhe des Säuglings

keine für Infrarotbeleuchtung notwendige 12 Volt Versorgungsspannung geliefert werden kann. Entsprechend muss für Nachtsichtüberwachung auf ein separates Beleuchtungsmodul zurückgegriffen werden.

Die Tonüberwachung erfolgt separat über einen auf dem Raspberry Pi installierten Audiostream-Server Icecast. Dieser nimmt das von einem angeschlossenen Mikrofon aufgenommene Signal auf, transkodiert es und sendet es an alle Clients, die mit dem Stream verbunden sind. Der Stream kann mit allen gängigen Mediaplayern geöffnet werden. Eine Einbettung in das Webinterface des Prototypen ist aufgrund von Imkompatibilitäten zwischen dem Streamsignal und dem HTML5 `<audio>`-Element im Rahmen dieser Arbeit nicht möglich gewesen.

VI. Technische Umsetzung

Im Rahmen dieser Arbeit wurde ein Prototyp erstellt, der die Gewichtsverlagerung des Säuglings und die Raumtemperatur überwacht. Im Folgenden wird die Technische Umsetzung, also der Aufbau der Messschaltung, die Implementierung der Treibersoftware und die Auswertung der Messwerte beschrieben.

A. Idee zur Atemüberwachung

Die grundlegende Idee zur Atemüberwachung wird aus den zuvor vorgestellten Arbeiten übernommen: Es wird versucht, die Atmung des Säuglings zu erfassen, indem seine Gewichtsverlagerung beim Atmen aufgenommen wird. Im Gegensatz zu den in der Literatur verwendeten kapazitiven Drucksensoren verwenden wir Dehnungsmessstreifen, wie man sie in herkömmlichen Küchenwaagen findet. Von diesen werden zwei wie in Abbildung 2 dargestellt verbaut: Der Säugling liegt auf einer massiven Platte, die am Fußende so auf Sockeln befestigt ist, dass sie sich zumindest in der Querachse geringfügig drehen kann. Auf Kopfhöhe des Säuglings ruht die Platte auf zwei Füßen der Küchenwaage, in denen sich jeweils zwei Dehnungsmessstreifen befinden. Eine Gewichtsverlagerung zum Kopf- oder Fußende hin bewirkt somit eine Änderung der Krafteinwirkung auf die Messstreifen.

B. Aufbau der Peripherie

Die Messwerte für Gewicht und Temperatur werden über geeignete Sensoren erfasst und mittels eines A/D-Wandlers digitalisiert. Der A/D-Wandler ist über SPI mit dem Raspberry Pi verbunden. Die Peripherie wird über den 3,3V-Ausgang

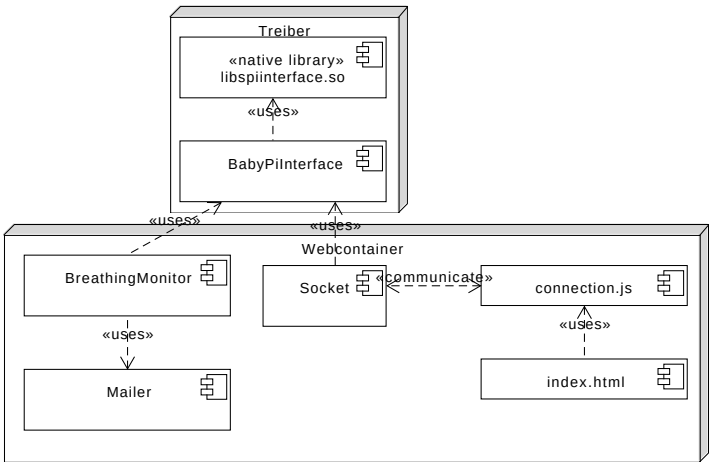


Abbildung 1. Softwarearchitektur des Prototyps

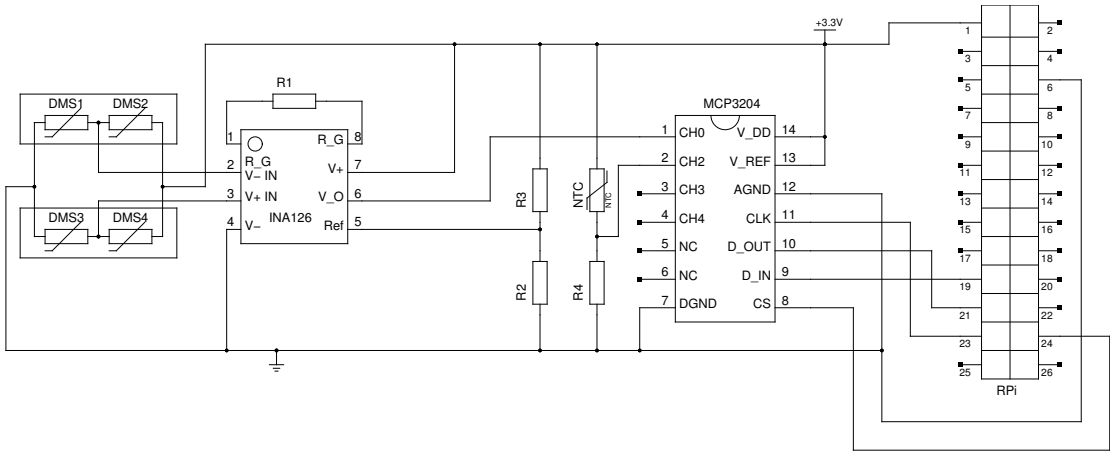


Abbildung 3. Schaltplan für die Peripherie

des Raspberry Pi versorgt. Die Schaltung ist in Abbildung 3 abgebildet.

Das Kernstück der Schaltung bildet der A/D-Wandler. Hier wurde der MCP3204 [9] der Firma Microchip Technology Inc. gewählt. Dieser hat 4 analoge Eingänge mit einer Auflösung von jeweils 12 Bit. Der erste Eingang (CH0) ist mit der Messschaltung für die Gewichtsmessung, der zweite (CH1) mit der Messschaltung für die Temperaturmessung verbunden. Der MCP3204 hat zwei Masse-Eingänge, AGND für die analogen Eingänge und DGND für SPI. Beide sind mit der Masse-Leitung des Raspberry Pi (GPIO-Pin 6) verbunden. Sowohl die Versorgungsspannung (V_DD) als auch die Referenzspannung (V_REF) sind 3,3V vom Raspberry Pi (Pin 1). Die Steuer- und Datenleitungen sind mit den entsprechenden Pins am GPIO-Header des Pi verbunden.

1) Gewichtsmessung: Für die Gewichtsmessung wurden zwei Sensoren aus einer preiswerten handelsüblichen Küchenwaage ausgebaut. Jeder Gewichtssensor besteht aus zwei Dehnungsmessstreifen (DMS), von denen einer bei Belastung gestaucht und einer gestreckt wird (vgl [10]) . Dies wurde vorab durch ein Experiment bestätigt. In Ruhelage beträgt der Widerstand eines DMS genau 1000Ω. Wenn mit der Hand Druck ausgeübt wurde, veränderte sich der Widerstand eines DMS um 2 bis 3Ω.

Im Prototyp werden zwei Gewichtssensoren, also vier DMS (DMS1 – 4) zu einer Wheatstone-Brücke (auch bekannt als Messbrücke) [11] verschaltet. Bei Krafteinwirkung erhöht sich der Widerstand von DMS2 und DMS3, während sich der von DMS1 und DMS4 verringert. Dadurch kann zwischen den Widerständen eine Spannung gemessen werden. Diese wird durch einen Instrumentenverstärker (vgl. [10]) ca. um das 100fache erhöht und in den A/D-Wandler gespeist. Als Instrumentenverstärker dient der INA126 der Burr-Brown Corporation [12]. Der Verstärkungsfaktor wird durch den Widerstand $R1 = 820\Omega$ eingestellt. Da nicht der gesamte Bereich am Ausgang ausgenutzt werden kann, wird die Masse-Referenz (V_O) des INA126 mithilfe eines Spannungsteilers aus $R2$ und $R3$ ($R2 = R3 = 1000\Omega$) auf +1,65V gesetzt.

2) Temperaturmessung: Zur Temperaturmessung wird ein Heißleiterwiderstand (NTC-Widerstand) gewählt. Diese eignen sich wegen ihrer großen Empfindlichkeit im Bereich der Raumtemperatur besonders gut. Die Wahl fiel auf den Typ TS-

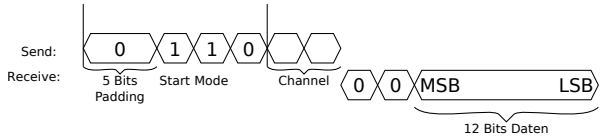


Abbildung 4. Datenübertragung zwischen Raspberry Pi und A/D-Wandler

NTC-202 der B+B Thermo-Technik GmbH [13]. Der NTC-Widerstand wird als Spannungsteiler in Reihe mit $R4 = 2000\Omega$ geschaltet. Die Spannung an $R4$ kann direkt vom A/D-Wandler gemessen werden.

C. Treibersoftware

Der A/D-Wandler kann über SPI angesteuert werden. Für den Raspberry Pi existiert ein Linux-Treiber, um die SPI-Schnittstelle zu steuern. Dieser erlaubt einen relativ einfachen Zugriff über das Dateisystem, bzw. IOCTL-Befehle [14]. Da das System in Java implementiert werden soll, wurden die nötigen IOCTL-Aufrufe durch das Java Native Interface (JNI) [15] gekapselt.

Die System-Schnittstelle ermöglicht das Senden und Empfangen von Daten in full-duplex mit einer Wortbreite von 8 Bits. Sender und Empfänger nutzen dabei das gleiche Clock-Signal.

Die Kommunikation mit dem MCP3204 sieht folgendermaßen aus: Der Master (in diesem Fall der Raspberry Pi) sendet das Start-Bit gefolgt von vier Steuerbits. Das erste gibt an, ob der A/D-Wandler im Single-Ended (1) oder im Differential-Modus (0) arbeiten soll. Das zweite Bit wird ignoriert, die dritten und vierten Bits kodieren den Kanal, der ausgelesen werden soll. Der A/D-Wandler wartet nach dem vierten Steuerbit zwei Takte, in denen 0 als Antwort gesendet wird, danach kommen die 12 Datenbits, die den gemessenen Wert kodieren. Das höchstwertige Bit wird zuerst gesendet.

Die Wortlänge bei der Übertragung beträgt 8 Bits. Damit das letzte Datenbit auf das letzte empfangene Bit fällt, werden vor dem Start-Bit 5 Bits mit 0 gesendet. Die gesamte Übertragung ist in Abbildung 4 dargestellt.

D. Auswertung der Messwerte

Der A/D-Wandler gibt eine 12-Bit-Zahl aus, die proportional zur gemessenen Spannung ist [9]. Es ergibt sich also für den Output X

$$X = \frac{4096 \times V_{IN}}{V_{REF}} \tag{1}$$

Da in der hier vorliegenden Anwendung nur das Verhältnis x von gemessener Spannung zur Referenzspannung interessiert, gilt

$$x = \frac{V_{IN}}{V_{REF}} = \frac{X}{4096} \tag{2}$$

1) Berechnung der Zimmertemperatur: Zur Messung der Raumtemperatur wird der NTC-Widerstand mit einem weiteren Widerstand R in Reihe geschaltet und die Spannung an R gemessen. Im Folgenden seien

- $x \in [0, 1]$ der Output des A/D-Wandlers
- $R = 2000\Omega$ der Widerstand, an dem gemessen wird
- T_{NTC} die momentane Temperatur des NTC-Widerstands
- T die gesuchte Raumtemperatur (in Kelvin)
- R_{NTC} der momentane Widerstand des NTC-Widerstands
- $T_0 = 25°\mathrm{C} = 298,15°\mathrm{K}$
- $R_0 = 2000\Omega$ der Widerstand des NTC-Widerstands bei T_0
- $B = 3976\mathrm{K}$ der Koeffizient des NTC-Widerstands
- $K = 1,2\frac{\mathrm{mW}}{\mathrm{K}}$ die Eigenerwärmung des NTC-Widerstands
- $U = 3,3\mathrm{V}$ die Gesamtspannung an der Messchaltung

Die spezifischen Werte ergeben sich aus der Schaltung und dem Datenblatt des NTC-Widerstands [13]. Für einen NTC-Widerstand gilt [10]

$$R_{NTC} = R_0 e^{B\left(\frac{1}{T_{NTC}} - \frac{1}{T_0}\right)} \tag{3}$$

Eine Umformung ergibt

$$T_{NTC} = \frac{BT_0}{T_0 \ln\left(\frac{R_{NTC}}{R_0}\right) + B} \tag{4}$$

R_NTC lässt sich aus

$$x = \frac{R}{R + R_{NTC}} \Leftrightarrow R_{NTC} = \frac{R}{x} - R \tag{5}$$

berechnen.

Durch seine Eigenerwärmung liegt die Temperatur des NTC-Widerstands ca. 2K über der Raumtemperatur. Die Eigenerwärmung muss also berücksichtigt werden. Für die Eigenerwärmung gilt [16]

$$\Delta T = (T_{NTC} - T) = \frac{P}{K} \tag{6}$$

wobei P die Leistung am NTC-Widerstand ist. Am NTC-Widerstand liegt die Spannung $U(1 - x)$ woraus mit dem Ohmschen Gesetz folgt:

$$P = \frac{(U(1-x))^2}{R_{NTC}} \tag{7}$$

und somit

$$\Delta T = \frac{(U(1-x))^2}{KR_{NTC}} \tag{8}$$

2) Messwerte für die Atmung: Um zu überprüfen, ob der Säugling atmet, wird das Gewicht des Säuglings auf die Dehnungsmessstreifen gemessen. Da diese sich nur unter der oberen Körperhälfte befinden, führen Gewichtsverlagerungen zu Änderungen der Messwerte.

Die Messwerte des A/D-Wandlers sind leider zu stark verrauscht, um sie direkt zu benutzen. Daher werden die Daten zunächst in Software gefiltert. Dazu wird ein *FIR-Filter* [17] benutzt, der mittels der *Frequency-Sampling-Technique* [17]

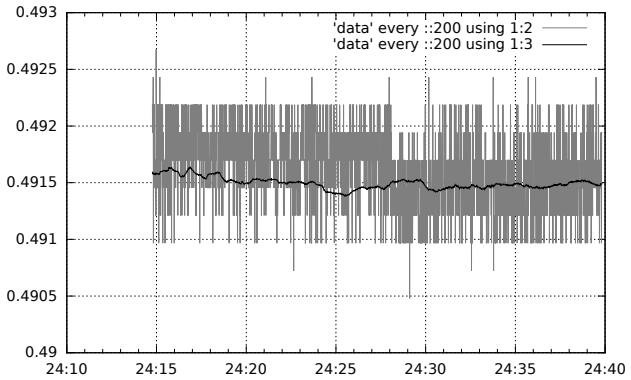


Abbildung 5. Eine Aufnahme von den unbelasteten DMS über 25 Sekunden. Die ungefilterten Daten sind grau, die gefilterten schwarz.

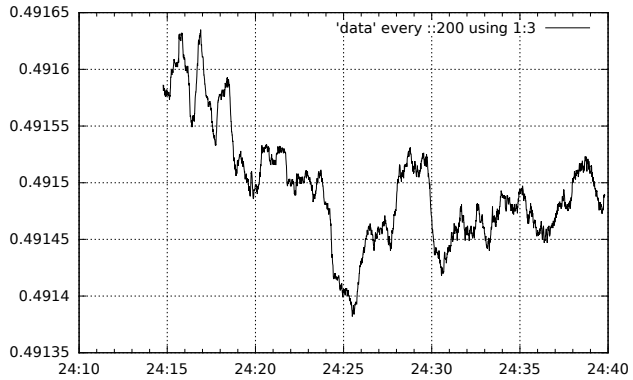


Abbildung 6. Die gefilterten Daten aus Abbildung 5.

aus einem idealen Tiefpass-Filter abgeleitet wird. Für den zeitdiskreten Output eines FIR-Filters mit Impulsantwort h gilt

$$y(n) = (x * h)(n) = \sum_{k=-\infty}^{\infty} x(n-k)h(k) = \sum_{k=1}^{M} x(n-k)h(k) \tag{9}$$

wegen $h(k) = 0$ für $k < 0 \vee k > M$. Für die Systemfunktion (z-transformierte der Impulsantwort) $H(\omega)$ eines idealen Tiefpass mit Grenzfrequenz f_c gilt

$$H(\omega) = \begin{cases} 1 & \text{für } \omega < \frac{2\pi}{f_c} \\ 0 & \text{sonst} \end{cases} \tag{10}$$

Nach [17] kann $h(k)$ aus $H(\omega)$ nach der Vorschrift

$$h(n) = A_0 + \sum_{k=1}^{\lfloor \frac{M}{2} \rfloor} 2A_k \cos\left(\frac{2\pi k\left(n+\frac{1}{2}\right)}{M+1}\right) \tag{11}$$

mit

$$A_k = \frac{1}{M+1} H(\frac{2\pi}{f_s}) \tag{12}$$

berechnet werden, wobei f_s die Samplingfrequenz ist.[1] Mit (10) ergibt sich

$$h(n) = \frac{1}{M+1}\left(1 + \sum_{k=1}^{\lfloor \frac{f_s}{f_c} \rfloor} 2\cos\left(\frac{2\pi k\left(n+\frac{1}{2}\right)}{M+1}\right)\right) \tag{13}$$

Es ist $M > \frac{2f_s}{f_c}$ zu wählen. In der Implementierung wird $M = \left\lfloor \frac{4f_s}{f_c} \right\rfloor$ benutzt.

In Abbildung 5 wurden Daten von den Dehnungsmessstreifen über einen Zeitraum von 25 Sekunden mit dem A/D-Wandler gemessen. Die Daten wurden mit dem oben beschriebenen Filter mit $f_s = 100\text{s}^{-1}$ und $f_c = 2\text{s}^{-1}$ gefiltert. Da die Atemfrequenz eines Säuglings bei ca. 40 Atemzügen pro Minute liegt, ist eine Grenzfrequenz von 2Hz auch für den Prototyp geeignet.

[1] In [17] wird f_s nicht berücksichtigt. Stattdessen findet sich $A_k = \frac{1}{M+1} H(\frac{2\pi k}{M+1})$.

Die Atemfrequenz interessiert beim Prototyp zunächst nicht. Daher wird zur Überprüfung der Atmung auf eine sehr einfache Technik zurückgegriffen: Seien

$$y_{\min} = \min\{y(t) | t > T - 20\text{s}\}, \tag{14}$$
$$y_{\max} = \max\{y(t) | t > T - 20\text{s}\} \tag{15}$$

der kleinste bzw. größte Wert, der innerhalb der letzten 20 Sekunden gemessen wurde. Wenn

$$a = y_{\max} - y_{\min} > \epsilon \tag{16}$$

gilt, ist eine Gewichtsverlagerung festzustellen – der Säugling atmet somit oder bewegt sich anderweitig. Dabei ist ϵ so zu wählen, dass in Ruhe noch $y_{\max} - y_{\min} < \epsilon$ gilt. Nach den oben gemessenen Werten eignet sich dazu $\epsilon = 3 \cdot 10^{-4}$ (siehe Abbildung 6). Das gewählte Intervall von 20s kann dabei variiert werden.

VII. Evaluation und Vergleich

In diesem Abschnitt erfolgt die Auswertung des umgesetzten Prototypen, indem dessen Funktionalität den Anforderungen aus Abschnitt IV und den erwerbbaren kommerziellen Produkten aus Abschnitt III gegenübergestellt wird.

A. Erfüllung der Anforderungen

Mit der im vorhergehenden Abschnitt beschriebenen Umsetzung des Prototypen sind die meisten der in Abschnitt IV geforderten Anforderungen erfüllt worden. Die vier Hauptfunktionalitäten $1-4$ sind alle umgesetzt worden. Die Anforderungen $5-9$ zur Darstellung der gemessenen Daten in einem Webinterface sind bis auf die Wiedergabe des Tons (Anf. 5) ebenfalls umgesetzt worden. Aus den Alarmierungsanforderungen $11-13$ wurden ebenso alle Anforderungen bis auf den Lautstärkeschwellwert umgesetzt.

Die Gründe für die fehlende Umsetzung der Anforderungen zur Tonüberwachung sind, wie bereits in Abschnitt V beschrieben, technischer Natur. Die umgesetzte Lösung mittels Icecast und Ices zeigt aber sehr wohl, dass das Raspberry Pi in der Lage ist, alle gesammelten Daten gleichzeitig und mit angemessener Latenz (wenige Sekunden Verzögerung in einem Gigabit-Ethernet) zur Verfügung zu stellen. Aufgrund der technischen Probleme ließen sich auch die tonbezogenen

Tabelle I. KOSTEN DES ENTWICKELTEN PROTOTYPEN (GERUNDET)

Posten	Kosten (Euro)
Raspberry Pi	28,50
NoIR Kamera-Modul	27,50
USB-Soundkarte	17,50
Mikrofon	12,00
Küchenwaage (Dehnungsmessstreifen)	14,00
Temperaturabhängiger Widerstand, Kabel, Stecker, ...	10,00
Summe	**109,50**

Alarmierungsoptionen nicht umsetzen. Generell sollte all dies aber mit mehr Zeit umsetzbar sein.

Die Anforderungen zur generischen Erweiterbarkeit des Datenempfangs wurden im Rahmen des Prototyps erprobt, jedoch bisher nicht dokumentiert. Im Rahmen dieser Arbeit wurde unter anderem der VLC Media Player [18] als mögliche Lösung für den Videostream an eine Webseite untersucht. Der Videostream in das HTML5 `<video>`-Element war damit leider nicht möglich, jedoch ließ sich feststellen, dass das Videosignal des Kameramoduls mithilfe des VLC Media Players an andere VLC Instanzen übertragbar ist. Da der Player für viele verschiedene Endgeräte, darunter auch Tablets und Smartphones, zur Verfügung steht, ist Anforderung 14 für den Videoanteil prinzipiell erfüllbar. Für die Tonübertragung ist dies durch die Verwendung von Icecast ohnehin erfüllt.

Die Anforderung 15 zur Alarmierung anderer Endgeräte mit direkter Übetragung des Bild- und Tonsignales wurde im Rahmen dieser Arbeit vorwiegend aufgrund proprietärer Kommunikationsprotokolle möglicher Messenger aber auch aus Zeitgründen nicht untersucht.

B. Vergleich zu kommerziellen Produkten

Die Anschaffungskosten für die Umsetzung des Prototypen sind in Tabelle VII-B dargestellt. Anhand dieser Kosten lässt sich ein Vergleich zu den kommerziellen Produkten aus Abschnitt III erstellen. Die gesamte Funktionalität des entwickelten Prototypen ließe sich durch eine Kombination von vier der dort beschriebenen Einzelprodukte erreichen: Babyphone, Thermometer, WLAN-fähige Kamera mit Nachtsicht und Atmungsüberwachungsgerät. Der Gesamtpreis dieser Produkte beliefe sich ohne Kamera bereits auf mindestens $139,99$ Euro. Das Angelcare AC1100 bietet drei der vier Funktionalitäten des Prototypen in einem Gerät und kostet $299,99$ Euro. Der entwickelte Prototyp ist also rein an den Hardwarekosten gemessen weitaus günstiger als kommerziell erhältliche Produkte.

Der entwickelte Prototyp hat aber mit den Anforderungen an zentrale, nicht an ein spezielles Endgerät gebundene Empfangsmöglichkeit und die Erweiterbarkeit ebendieser Endgeräte eine Funktionalität, die keines der kommerziellen Produkte bietet. Somit stellt die Idee auch unter Berücksichtigung der Entwicklungskosten einen Mehrwert dar.

VIII. AUSBLICK

Die in dieser Arbeit beschriebene Umsetzung eines Prototypen zur Säuglingsüberwachung ist, wie in Abschnitt VII beschrieben, erfolgreich durchgeführt worden. Die Erfüllung der Anforderungen aus Abschnitt IV war generell erfolgreich und es ließ sich ein Gesamtprodukt herstellen, dass bei größerem Funktionsumfang günstiger ist als kommerzielle Produkte.

Einige Anforderungen konnten aufgrund der zeitlichen Begrenzung der Arbeit oder Problemen bei der Medienverarbeitung nicht umgesetzt werden, wurden jedoch prinzipiell auf Realisierbarkeit geprüft. Auf genau diesen unerfüllten Anforderungen baut eine weitere, grundlegende Funktionalität auf, mit der sich die prototypische Entwicklung noch erweitern ließe: Nutzung der Kamera und des Tons zur Alarmierung. Insbesondere die Nutzung von Videotelefonie, um Eltern über Aktivitäten oder Alarmsignale aus dem Kinderzimmer direkt auf ihrem Smartphone oder Tablet mit Livevideo und Ton zu informieren, könnte den Nutzen des Produkts stark erweitern.

Als weiteres Einsatzgebiet wäre die medizinische Überwachung von Patienten denkbar. Mithilfe der hier dargestellten Mittel ließe sich, nach einer sicherheitskritischen Entwicklung und unter Berücksichtigung von Hochverfügbarkeitsanforderungen, ein einfaches, generell einsetzbares und kostengünstiges Modul mit Netzwerkanbindung entwickeln. Dies könnte von Krankenhäusern zur Überwachung der Vitalfunktionen von Patienten eingesetzt werden.

LITERATUR

[1] G. Jorch, „Prävention des plötzlichen Kindstods," *Monatsschrift Kinderheilkunde*, Vol. 158, April 2010.

[2] S. Brandt-Niebelschütz *et al.*, „Der plötzliche Säuglingstod: Akzeptanz und Nutzen des Heim-Monitorings," *Deutsches Ärzteblatt*, Vol. 55, Nr. 6, S. 396–401, Februar 1991.

[3] C. I. Franks, B. H. Brown, und D. M. Johnston, „Contactless respiration monitoring of infants," Mai 1976.

[4] Shoko Nukaya *et al.*, „A noninvasive heartbeat, respiration, and body movement monitoring system for neonates," *Artif Life Robotics*, Vol. 19, S. 414–419, 2014.

[5] Motorola. (2015) Motorola MBP8. [Online]. Available: http://www.motorola.de/consumers/babyfon-header-de/Digitales-MBP8-Audio-Babyfon/mbp8-baby-de.html

[6] Hama. (2015) Hama Wetterstation EWS-165. [Online]. Available: https://de.hama.com/00092659/hama-elektronische-wetterstation-ews-165-schwarz

[7] Angelcare. (2015) Angelcare AC300. [Online]. Available: http://www.angelcare.de/produkte/angelcare-ac300.html

[8] ——. (2015) Angelcare AC1100/AC1100-D. [Online]. Available: http://www.angelcare.de/produkte/angelcare-ac1100.html

[9] *MCP3204/3208: 2.7V 4-Channel/8-Channel 12-Bit A/D Converters with SPI Serial Interface*, Microchip Technology Inc., 2008.

[10] E. Schrüfer, *Elektrische Meßtechnik: Messung elektrischer und nichtelektrischer Größen*. München: Hanser, 1983.

[11] W. D. Cooper und A. D. Helfrick, *Elektrische Meßtechnik: Electronic instrumentation and measurement techniques*. Weinheim: VCH, 1989.

[12] *MicroPOWER INSTRUMENTATION AMPLIFIER: Single and Dual Versions*, Burr-Brown Corporation, 1997, INA126/INA2126 Datenblatt.

[13] *Präzisionstemperatursensor CON-TS-NTC*.

[14] W. W. Gay, *Raspberry Pi Hardware Reference*. Berkeley, 2014.

[15] J. Goll, C. Weiß, und F. Müller, *Java als erste Programmiersprache: vom Einsteiger zum Profi*, 3. Aufl. Stuttgart: Teubner, 2001.

[16] M. Nau, *Elektrische Temperaturmessung: mit Thermoelementen und Widerstandsthermometern*, JUMO GmbH & Co. KG, Fulda, 2007.

[17] L. B. Jackson, *Digital Filters and Signal Processing: with MATLAB Exercises*. Kluwer Academic Publishers, 1999.

[18] VideoLAN Organization. (2015) VLC Media Player. [Online]. Available: http://www.videolan.org/vlc/index.html

Kameraüberwachter Nistkasten

Jenny Inge Röbesaat
Carl von Ossietzky Universität Oldenburg
WS 2014/2015
Seminar: Messen, steuern, verteilen:
Überwachen und Manipulieren mit dem Raspberry Pi

***Abstract*—Der Raspberry Pi ist eine Plattform mit vielseitigen Anwendungsmöglichkeiten. Dazu werden in dieser Ausarbeitung die Entwicklung und der Bau eines kameraüberwachten Nistkastens, basierend aus einen Raspberry Pi, beschrieben. Der kameraüberwachte Nistkasten kann genutzt werden, um das Brut- und Aufzuchtverhalten von Gartenvögel genauer zu untersuchen. Dazu wird im Nistkasten ein Kameramodul montiert, welches Bilder aufnimmt. Um dabei Speicherplatz und Rechenaufwand zu sparen, ist der Nistkasten mit einem Bewegungsmelder ausgestattet, damit das Kameramodul nur anhand von registrierten Bewegungen Bilder aufnimmt. Zum Sichern der Bilder werden die aufgenommenen Bilder weiter zu einem zweiten Raspberry Pi gesendet und dort auf einem externen Speicher gesichert. Zudem wird die aktuellste Bildaufnahme über einen Webserver im lokalem Netzwerk zur Verfügung gestellt. Nachdem die Brut- und Aufzuchtzeit der Gartenvögel beendet ist, wird aus allen aufgenommenen Bildern ein Video erstellt.**

I. Einleitung

Heutzutage haben es heimische Vogelarten immer schwerer einen geeigneten Brutplatz zu finden. Geeignete Brutplätze wie alte absterbende Bäume, dichte Hecken oder Sträucher gibt es aufgrund von intensiv gepflegten Gärten und Parkanlagen sowie intensivere Betreibung der Forst- und Agrarwirtschaft immer weniger [1]. Deshalb rufen Naturschutzorganisationen wie der Naturschutzbund (NABU) oder der Bund für Umwelt und Naturschutz Deutschland e. V. (BUND) immer wieder dazu auf Nistkästen für Vögel aufzustellen. Um das Brut- und Aufzuchtverhalten von Vögeln im Nistkasten genauer zu untersuchen, kann ein Nistkasten mit einer Kamera ausgestattet werden.

A. Aufgabenbeschreibung

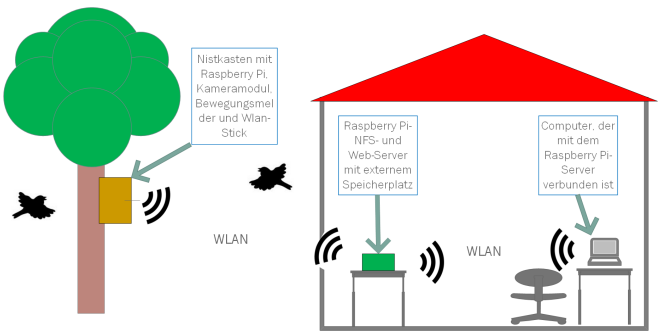


Fig. 1. Szenario

Gegenstand dieser Ausarbeitung soll die Entwicklung und der Bau eines kameraüberwachten Nistkastens sein. Für die Umsetzung wird im Rahmen des Seminars "Messen, steuern, verteilen: Überwachen und Manipulieren mit dem Raspberry Pi" ein Raspberry Pi verwendet. Beim Raspberry Pi handelt es sich um einen kostengünstigen, kleinen Computer, der auf Linux basiert. Dadurch, dass bereits ein vollständiges Betriebssystem auf dem Raspberry Pi installiert ist, ist eine bequeme Entwicklung möglich. Das Kernstück des Raspberry Pis ist ein ARM-Prozessor mit $700MHz$. Durch die eingebaute GPIO-Schnittstelle (General Purpose Input/ Output) lassen sich verschiedene Sensoren und Aktoren steuern. Neben der GPIO-Schnittstelle besitzt der Raspberry Pi weitere Anschlüsse wie zwei USB-Ports und eine Camera Serial Interface-Schnittstelle (CSI) [12]. Das in Abbildung 1 dargestellte Szenario zeigt einen Nistkasten, der von Gartenvögel zur Brut genutzt werden soll. In dem Nistkasten befindet sich eine Kamera, die Bilder aufnimmt. Bei den Aufnahmen muss berücksichtigt werden, dass er im Nistkasten sehr dunkel sein kann. Da der Speicherplatz auf des SD-Karte begrenzt ist und die Bildaufnahmen interessant gehalten werden sollen, werden die Bildaufnahmen nur durchgeführt, wenn sich im Nistkasten etwas bewegt. Weiterhin sollen aus Gründen wie geringer Speicherplatz, aber auch zum Schutz des Bildmaterials, die ausgenommenen Bilder direkt weiter zu einem zweiten Raspberry Pi geschickt werden und dort auf einem externen Speicher gespeichert werden. Es soll mithilfe eines Webserver das aktuell aufgenommene Bild im lokalem Netzwerk zur Verfügung gestellt werden. Zusätzlich soll am Ende der Brut- und Aufzuchtzeit ein Video aus den einzeln aufgenommen Bilder erstellt werden.
Es gibt bereits einige kommerziellen Lösungen, bei denen ein Nistkasten von einer Kamera überwacht wird und das aktuelle Bild an einem Bildschirm dargestellt werden kann. Jedoch gibt es keine kommerziellen Produkte, bei denen die Bildaufnahmen durch Bewegungserkennungen gestartet oder das Bildmaterial sicher gespeichert wird. Neben den wenigen Funktionen, sind die kommerziellen Lösungen oft sehr teuer. Ziel dieser Ausarbeitung soll es sein, das dargestellte Szenario umzusetzen und zu beschreiben.

Bei der Realisierung des Szenarios muss zunächst ein geeigneter Nistkasten gewählt werden. Dabei muss beachtet werden, dass dieser artenspezifisch ist und abhängig von der Vogelart verschiedene Anforderungen erfüllen muss. Weiterhin

muss berücksichtigt werden, dass es im Nistkasten dunkel ist. Das Anbringen von Leuchtmittel würde jedoch die Vögel vertreiben. Eine mögliche Alternative wäre den Raum mit Infrarotlicht zu bestrahlen. Infrarotlicht ist für Menschen und Vögel nicht sichtbar und würde die brütenden Vögel nicht vertreiben. Für geeignete Bildaufnahmen mit Infrarotlicht muss ein Kameramodul ausgewählt werden, dass keinen Infrarotfilter besitzt. Für diesen Einsatz eignet sich das Raspberry Pi-NoIR-Kameramodul. Dieses besitzt keinen Infrarotfilter. Zudem lässt sich das Kameramodul leicht an den Raspberry Pi über die vorhandene CSI-Schnittstelle anschließen. Eine Herausforderung bei der Speicherung der Bilder ist der begrenzte Speicherplatz. Der Speicher darf während der Brut- und Aufzuchtzeit nicht gewechselt werden, da dies die Vögel erheblich stören würde. Auch bei der Verwendung eines ausreichend großen Speichers besteht das Problem, dass die Bilddaten nicht zwischendurch gesichert werden können und der Speicher z.B nicht gegen Wettereinflüsse oder Vandalismus der Vögel gesichert ist. Daher werden die Bilder direkt nach der Aufnahme zu einem zweiten Raspberry Pi gesendet und dort auf einem ausreichend großen externen Speicher gesichert werden. Für die Übertragung des Bildmaterial wurde eine WLAN-Verbindung ausgewählt. Um die Bildübertragung simpel zu halten, wurde auf dem zweiten Raspberry Pi ein Network File System Server eingerichtet. Zusätzlich soll das aktuellste Bild aus dem Nistkasten mittels eines Webservers dargestellt werden. Am Ende der Brut- und Aufzuchtszeit soll ein Video aus allen aufgenommenen Bildern erstellt werden.

Die Ausarbeitung gliedert sich wie folgt: Zunächst wird im Abschnitt II die ausgewählte Vogelart, für die der Nistkasten ausgelegt sein soll, vorgestellt. Im Anschluss wird im Abschnitt III die Aufnahmemöglichkeiten mit dem Kameramodul und die Möglichkeit zur Bewegungserkennung thematisiert. Der Bau des Prototyps wird im Abschnitt IV beschrieben. Im darauf folgendem Abschnitt V wird das Versenden und Speichern des Bildmaterials sowie das Hochladen des aktuellen Bildes auf einen Webserver erklärt. Die Implementierung des erstellten Programms wird im Abschnitt VI vorgestellt. Im abschließendem Abschnitt VII erfolgt ein Fazit mit den Ergebnissen, sowie zukünftigen Verbesserungs- und Erweiterungsmöglichkeiten.

II. Vogelart

Für den Bau eines kameraüberwachten Nistkastens muss zunächst geklärt werden, für welche Vogelart der Nistkasten ausgelegt werden soll. Abhängig von der Vogelart müssen beim Nistkastenbau verschiedene Kriterien erfüllt werden. Im folgenden werden die ausgewählte Vogelart, die Kohlmeise, und die zu berücksichtigen Nistkastenbaukriterien beschrieben.

A. Kohlmeise

Als Vogelart wurde die Kohlmeise ausgewählt, da diese sehr stark in Deutschland vertreten ist und dadurch die Wahrscheinlichkeit deutlich erhöht ist, dass der Nistkasten angenommen wird. Die in Abbildung 2 dargestellte Kohlmeise ist die größte Meisenart in Deutschland. Sie erreicht eine durchschnittliche Körperlänge von ca. 14cm und ein Gewicht von ca. 20g. Ihr Äußeres ist gekennzeichnet durch einen schwarz-weißen Kopf und eine gelbe Unterseite mit einem schwarzen Bauchstreifen. Kohlmeisen gehören zu den Höhlenbrüter und bevorzugen es in Bäumen oder in Nistkästen zu brüten. Für die Brut polstern sie das Nest mit Moos, Wolle, Haaren, Halmen und Federn aus. Sie brüten bis zu zweimal pro Jahr und legen dabei bis zu zwölf Eier. Die Brutdauer beträgt 14 Tage. Anschließend folgen eine sogenannte ”Nestlingszeit” mit bis zu 20 Tagen und eine sogenannte ”Ästlingszeit”, in der die Jungen das Nest langsam verlassen, von 14 Tagen . Um die Brut ausreichend zu ernähren, werden diese mit Blattläusen, Raupen, Spinnen und Sämereien gefüttert. Zu den Feinden der Kohlmeise zählen unter anderem Katzen, Marder und Elstern. [1],[2],[3]

Fig. 2. Die Kohlmeise (Abb. aus [13])

B. Nistkasten

Beim Nistkasten für Kohlmeisen ist darauf zu achten, dass das Einflugloch einen Durchmesser von ca. 32mm hat und dass der Nistkasten im Idealfall nach Osten ausgerichtet ist, da der Wind meisten aus Westen kommt und eine Ausrichtung nach Süden den Nistkasten zu stark erwärmen würde [5]. Weiterhin ist zu beachten, dass der Nistkasten sicher vor Feinden aufgehangen wird. Beim Verbau der Elektronik im Nistkasten muss zudem auf eventuell auftretende Feuchtigkeit geachtet werden [6].

III. Bildaufnahme bei Bewegungserkennung

Das im Nistkasten angebrachte Kameramodul soll bei einer Bewegungserkennung Bilder aufnehmen, die später zu einem Video zusammengefügt werden. Im folgenden werden Möglichkeiten zu Bildaufnahmen, Bewegungserkennung und Videoerstellung vorgestellt.

A. Raspberry Pi-NoIR-Kameramodul

Im Mai 2013 hat die Raspberry Pi Foundation das kostengünstige Raspberry Pi-Kameramodul vorgestellt. Diese Kamera besteht aus einem kleinen PCB (Printed Circuit Board), auf dem ein Omnivision OV5647 Kameramodul angebracht ist. Mittels einem ca. 15cm langem Flachbandkabel kann die Kamera über den CSI-Anschluss des Raspberry Pis mit diesem verbunden werden [7]. Mit der Kamera können

Videos im HD-Format von 1080p oder 720p bzw. im VGA-Format 640x480 aufgezeichnet werden [8].

Das normale Kameramodul besitzt einen Infrarot-Filter mit dem Licht im infraroten Bereich weggefiltert wird. Infrarotes Licht hat eine Wellenlänge zwischen 3mm und 780nm und ist damit für das menschliche Auge nicht sichtbar. Später kam das Raspberry Pi-NoIR-Kameramodul (Abb. 3) auf den Markt. Diese Kamera besitzt keinen Infrarotfilter. Daher ist es mit dieser Kamera möglich infrarotes Licht zu erfassen [8]. Dieser fehlende Filter kann tagsüber zu verfälschten Farbaufnahmen führen. Jedoch können dunkle Umgebungen mit IR-Licht bestrahlt werden, wodurch es mit dem Kameramodul ohne Infrarot-Filter möglich ist, diese Bilder beleuchtet darzustellen [10].

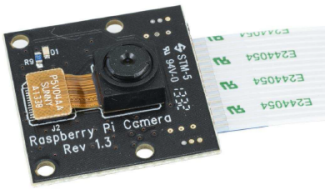


Fig. 3. Das Raspberry Pi-NoIR-Kameramodul (Abb. aus [6])

B. Infrarot-Light-Emitting-Diode

Infrarot-Light-Emitting-Diodes (IR-LEDs) funktionieren wie normale LEDs mit dem Unterschied, dass deren Licht für Menschen und Vögel nicht im sichtbaren Lichtspektrum liegt [8]. Für dieses Projekt werden IR-LED vom Hersteller Sanken Electric der Reihe SID1050M verwendet. Die IR-LEDs strahlen Licht mit einer Wellenlänge von 940nm ab. Sie benötigen eine Spannung von 1,3V und einen maximalen Strom von 50mA [19]. Um den Strom zu begrenzen, benötigen die IR-LEDs einen Vorwiderstand. Für den Prototypen werden zwei IR-LEDs parallel zueinander geschaltet (Abb. 4). Jede IR-LED wird in Reihe mit einem Vorwiderstand geschaltet. Die LEDS werden mit den Pins ”GND” und dem freiprogrammierbaren Pin ”25” der GPIO-Schnittstelle des Raspberry Pis verbunden.

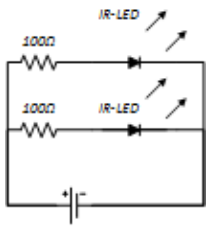


Fig. 4. Schaltung IR-LEDs mit Vorwiderständen

Der Raspberry Pi wird so programmiert, dass die IR-LEDs eingeschaltet werden, sobald eine Bewegung erkannt wird und bleiben solange angeschaltet, bis die aktuellen Aufnahmen beendet sind. Die verwendeten Vorwiderstände lassen sich nach dem Ohmschen Gesetz berechnen. An jeder LED fallen 1.3V ab, so dass von den ursprünglichen 5V noch 3.7V am Widerstand abfallen müssen:

$$U_R = U_{ges} - U_{led} = 5V - 1.3V = 3.7V$$

Weiterhin soll ein Strom von 50mA fließen. Dadurch ergibt sich aus

$$R = \frac{U}{I} = \frac{3.7V}{50mA} = 74\Omega$$

ein Vorwiderstand von 74 Ω. Da der maximale Strom 50mA betragen darf, wird sicherheitshalber ein größerer Widerstand von 100 Ωverwendet. Dadurch ergibt sich folgender Strom:

$$I = \frac{U}{R} = \frac{3.7V}{100\Omega} = 37mA$$

Die Aufnahmen im Dunklen mit den eingeschalteten IR-LEDs zeigen ein beleuchtetes und gut erkennbares Bild. [25]

C. Bewegungserkennungen

Für die Erkennung von Bewegungen gibt es verschiedene Möglichkeiten. Zum einem kann dieses mit einem Sensor realisiert werden zum anderem gibt es auch Softwarelösungen, die Bewegungen erkennen können.

1) Passive Infrared Sensor (PIR): Ein häufig verwendeter Bewegungsmelder ist der Passive Infrared Sensor (PIR). Dieser erkennt anhand von Temperaturveränderungen Bewegungen von Objekten wie Tiere und Menschen [10]. Der für diesen Prototypen verwendeter PIR-Sensor ist der passive infrarot Bewegungsmelder Low Power (PIL-LP) der Firma Hygrosens Instruments GmbH (Abb. 5), der mit $3 - 5V$ betrieben wird [4]. Der Sensor besitzt eine runde, milchige Kuppel, die mit vielen einzelnen Waben versehen ist. Der PIR-Sensor hat 8 Pins. Davon werden zwei Pins für die Stromversorgung und einer für das Auslesen des Sensorwertes verwendet [4]. Diese Pins werden mit der GPIO Schnittstelle des Raspberry Pis verbunden (Vgl. Abb. 7). Die Empfindlichkeit des PIR-Sensors kann durch Schließen von bis zu vier Lötbrücken verringert werden.

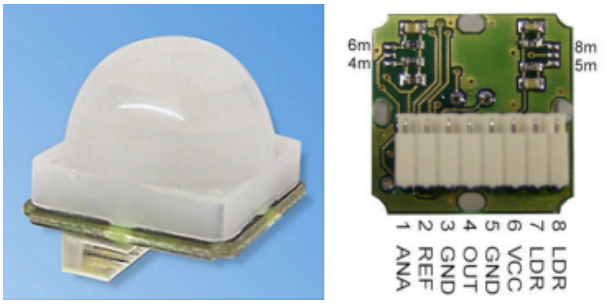


Fig. 5. PIR-Sensor der Firma Hygrosens Instruments GmbH (Abb. aus [4])

2) Das Paket ”motion”: Eine Alternative zum PIR-Sensor ist das Paket *motion*, mit dem es möglich ist, Bewegungen zu erkennen. Dabei überwacht *motion* das Videosignal der Kamera und erkennt, wenn sich ein wesentlicher Teil des Bildes ändert [17].

D. Video-Erstellung

Aus allen Bildaufnahmen soll ein Video erstellt werden. Die aufgenommenen Bilder wurden unter einer fortlaufenden Nummer gespeichert, sodass das Programm, das das Video erstellt, die Aufnahmen in der richtigen Reihenfolge zusammenfügen kann. Für das Zusammenstellen des Videos wird der Audio-Video-Converter *avconv* verwendet. Da jedoch das Zusammenfügen der Bildaufnahmen sehr rechenintensiv ist, kann dieses nicht auf dem gleichen Raspberry Pi durchgeführt werden, auf dem auch die Bildaufnahmen stattfinden. Deshalb werden die Bilder zunächst an einen zweiten Raspberry Pi geschickt, bei dem das Video mithilfe von *avconv* erstellt wird [10].

IV. Nistkastenbau

Für den Bau des kameraüberwachten Nistkastens wird ein Nistkasten der Firma LifeTimeGarden (Abb. 6) als Basis verwendet. Dieser Nistkasten hat ein Einflugloch mit einem Durchmesser von 32mm und ist daher gut für Kohlmeisen geeignet. Weiterhin ist er imprägniert, sodass keine Flüssigkeiten wie Regen in das Innere des Nistkasten gelangen können. Der Deckel des Nistkastens lässt sich aufklappen. Dadurch ist es möglich die Elektronik einfach auf der Deckelunterseite zu verbauen.

Fig. 6. Nistkasten (Abb. aus [16])

Der Schaltplan wird in Abbildung 7 dargestellt. Dieser besteht aus zwei IR-LEDs mit jeweils einem 100 Ω-Vorwiderstand (Vgl. 4), dem PIR-Sensor und dem Raspberry Pi. In dem Schaltplan wird nicht das NoIR-Kameramodul, welches über die CSI-Schnistelle verbunden ist, dargestellt.

Beide LEDs mit den Widerständen werden an den Pins ”25”, ein freiprogrammierbarer Pin und ”GND” der GPIO Schnittstelle des Raspberry Pis angeschlossen. Über den Pin ”25” sollen die LEDs angesteuert. Beim PIR-Sensor werden drei Pins mit der GPIO-Schnittstelle verbunden. Das sind zum einem die Pins ”3.3V” und ”GND” für die Stromversorgung und ”Pin 26”, ein frei programmierbarer Pin für das Auslesen

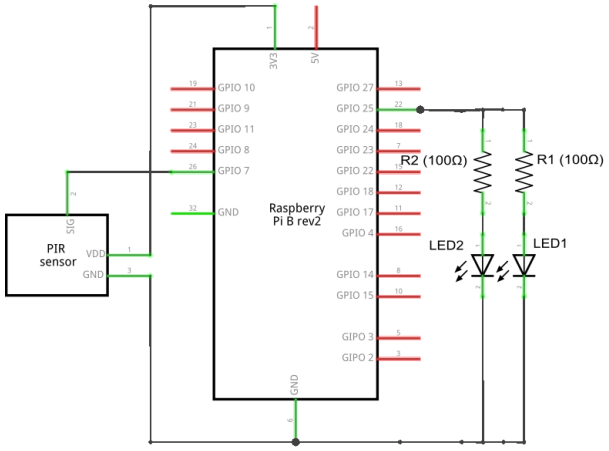


Fig. 7. Schaltplan

der Sensordaten. In Abbildung 8 ist die Umsetzung auf der Unterseite des Nistkastendeckels zu sehen. Im Zentrum des Deckels ist, das Kameramodul angebrecht. Die IR-LEDs wurden mit den Widerstanden auf kleine Streifenrasterplatinen gelötet und in zwei Ecken des Deckel, diagonal zueinander, befestigt. Dadurch soll der Raum des Nistkastens ausreichend und gleichmäßiger bestrahlt werden. Der PIR-Sensor wird mithilfe der inneren Plastikverpackung eines Überraschungseies präpariert und in einer anderen hinteren Ecke des Nistkastendeckels befestigt. Die Streifenrasterplatinen mit den LEDs, der PIR-Sensor und das NoIR Kamera Modul werden mithilfe von Reißzwecken an die Unterseite des Deckels befestigt. Zur besseren Befestigung und zur Isolierung der Bauteile mit den Reißzwecken wird zusätzlich Isolierband verwendet.

V. Versand und Speichern des Bildmaterials auf einem externen Speicher

Nachdem die Bilder aufgenommen wurden, sollen diese über eine WLAN-Verbindung an einen zweiten Raspberry Pi gesendet und dort auf einen externen Speicher gespeichert werden. Dieser Abschnitt beschreibt zunächst das Einrichten einer WLAN-Verbindung und das Mounten eines externen Speichers. Anschließend wird beschrieben, wie ein Network-File-System-Server auf dem zweiten Raspberry Pi, dem Raspberry Pi-Server, eingerichtet wird. Abschließend wird das Darstellen des aktuell aufgenommene Bild mittels einem Webserver im lokalem Netzwerk erklärt.

A. WLAN-Verbindung herstellen

Das Versenden der Bilder soll über eine WLAN-Verbindung geschehen. Dafür werden beide Raspberry Pis mit dem ”Wireless N 150Mbps USB Adapter” der Firma LogiLink ausgestattet. Da nur die beiden Raspberry Pi miteinander verbunden werden sollen, wird ein Ad-hoc-Netz eingerichtet. Nachdem die verwendeten WLAN-Adapter in die Raspberry Pis eingesteckt wurden, wurde der Befehl *ifconfig* ausgeführt. Es erscheint die neue Netzwerkschnittstelle *wlan0*. Da beide Raspberry Pi mit keinem DHCP-Server verbunden sind,

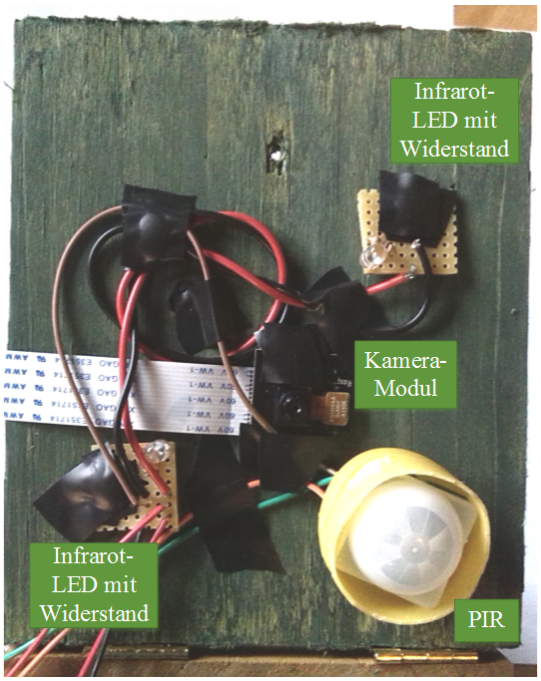


Fig. 8. Aufbau der Elektronik auf dem Nistkastendeckel

müssen beiden WLAN-Schnittstellen eine statische IP Adresse zugeordnet werden. Dies passiert, indem in die Konfigurationsdatei */etc/network/interfaces* die neue Netzwerkverbindung hinzugefügt und entsprechend konfiguriert wird. In Tabelle I sind die hinzugefügten Einträge dargestellt.

Raspberry Pi 1:
auto wlan0
iface wlan0 inet static
address 192.168.1.2
netmask 255.255.255.0
wireless-channel 1
wireless-essid MYNETWORK
wireless-mode ad-hoc
Raspberry Pi 2:
auto wlan0
iface wlan0 inet static
address 192.168.1.1
netmask 255.255.255.0
wireless-channel 1
wireless-essid MYNETWORK
wireless-mode ad-hoc

TABLE I
AUSCHNITT AUS /ETC/NETWORK/INTERFACES

Zunächst wird mit *auto wlan0* festgelegt, dass diese Schnittstelle bei jedem Bootvorgang automatisch gestartet wird. Anschließend wird mit *iface wlan0 inet static* die Schnittstelle konfiguriert. Dabei wird mit *inet* angegeben, dass mit dem Netzwerkprotokoll IPv4 kommuniziert werden soll und mit *static* gibt man an, dass es sich bei der Konfiguration um eine statische IP-Adresse handelt. Jeder Schnittstelle wird mit *address* eine IP-Adresse zugeordnet. Diese IP-Adressen müssen sich unterscheiden, aber im gleichen Subnetzwerk liegen. Anschließend wird den Schnittstellen die Netzmaske zugewiesen. Diese ist bei allen IP-Adressen, die im gleichen Subnetzwerk liegen, identisch. Daraufhin wird mit *wireless-channel 1* der Channel angegeben über den die Daten gesendet werden sollen. Mit *wireless-essid MYNETWORK* benennt man das Netzwerk mit MYNETWORK und mit *wireless-mode ad-hoc* wird der Schnittstelle mitgeteilt, dass es sich um Ad-hoc-Netz handelt. Nachdem die Konfiguration der beiden Schnittstellen abgeschlossen wurde, wird mit

$$ping\ 192.168.1.1$$

bzw. mit

$$ping\ 192.168.1.2$$

die Verbindung zwischen den beiden Raspberry Pis getestet. [14], [15]

B. Externen Speicher einrichten

Die aufgenommen Bilder müssen gespeichert werden. Da SD-Karten nur einen kleinen Speicher aufweisen bzw. bei größeren Speichern sehr teuer werden, wird ein kostengünstiger, externer Speicher am zweiten Raspberry Pi eingerichtet. Dieser externe Speicher könnte z.B. eine externe Festplatte mit mehr als einen Terrabyte Speicher sein und hätte damit genügend Speicherplatz, die während der gesamten Brut-und Aufzuchtzeit aufgenommenen Bilder, zu speichern. Bei dem entwickelten Prototyp wird ein USB-Stick von Intenso mit einem Speicher von 64GB verwendet. Dieser kann aber mit geringem Aufwand gegen eine größere Festplatte ersetzt werden. Als erstes wird der USB-Stick mit dem EXT4 Dateisystem formatiert. Danach wird der USB-Stick ”gemountet”. Dafür wird im Verzeichnis *media* ein neues Verzeichnis mit dem Namen *nistkasten* angelegt. Unter dem Verzeichnis *nistkasten* soll der USB-Stick, nachdem er ”gemountet” wurde, zu finden sein. Damit der USB-Stick nach jedem Booten automatisch erkannt wird, muss dieser in der Konfigurationsdatei $/etc/fstab$ angegeben werden. [9], [20]

C. Network File System Server einrichten

Das Network File System (NFS) ist ein Protokoll mit dem Unix- und Linux-Systeme über ein lokales Netzwerk auf Daten zugreifen können. Der Benutzer kann so auf einen NFS-Server zugreifen wie auf eine lokale Festplatte [22],[23]. Der NFS-Server wird auf dem zweiten Raspberry Pi, dem Raspberry Pi-Server, auf den auch der USB-Stick gemountet ist, eingerichtet. So kann der Raspberry Pi-Client, der die Fotos aufnimmt, diese auf dem NSF-Server speichern. Der Vorteil eines NFS-Server sind der geringe Verbrauch des Arbeitsspeichers und die kurzen Antwortzeiten über ein lokales Netzwerk [24]. Für das Einrichten eines NFS-Server muss zunächst das Paket *nfs-kernel-server* installiert werden. Nach der Installation erscheint eine Warnmeldung. Das liegt daran, dass versucht wurde den NFS-Server zu starten. Dies ist aber nicht möglich, da die Datei *etc/exports* noch keine Verzeichnisse enthält. Damit der NFS-Server ein Verzeichnis freigegeben kann, muss dieses Verzeichnis in die

Datei *etc/exports* eingefügt werden. Dazu wird das bereits erstellte Verzeichnis *media/nistkasten* in die Datei *etc/exports* hinzugefügt. Bevor der NSF-Server gestartet werden kann, muss in der Datei */etc/netconfig* der IPv6-Betrieb deaktiviert werden, da der Raspberry Pi diesen nicht unterstützt [21], [18]. Nachdem der NFS-Server gestartet wurde, kann der Raspberry Pi-Client diesen wie eine lokale Festplatte mounten und anschließend Daten vom freigegebenem Verzeichnis lesen und auf diesen auch Daten speichern.

D. Webserver

Wie im Szenario beschrieben (Vgl. Abschnitt 1) sollen das aktuelle Bild über eine Webserver im lokalem Netzwerk dargestellt wird. Dafür muss der Raspberry Pi-Server mit einem lokalem Netzwerk verbunden sein. Für die Umsetzung wird ein *lighttpd*-Webserver auf dem Raspberry Pi-Server eingerichtet [9]. *lighttpd* ist ein Webserver, der weniger Ressourcen benötigt als z.B der Apache und ist deshalb besonders für schwache Systeme wie den Raspberry Pi geeignet [26]. Es wurde ein HTML-Programm geschrieben, das das aktuelle Bild auf dem Webbrowser darstellt (Tabelle II). Die angebenen Metadaten sorgen dafür, dass das Bild alle 5 Sekunden aktualisiert wird und das kein Cache verwendet wird, in dem ältere Bilder gespeichert werden könnten. Zudem wird mit dem HTML-Programm die Überschrift, die Größe und der Pfad des Bildes angegeben.

```
<html>
<head>
<meta http.equiv="Refresh" content="5">
<meta http.equiv="Pragma" content="no-cache">
<title>Nistkasten</title>
<head>

<body>
<h1 align="center">Nistkasten</h1>
<p align="center">
<img src="1_webserver.jpg" width="640" height="480">
</p>
</body>
</html>
```

TABLE II
HTML-DATEI (AUS [9])

Sowohl die HTML-Datei als auch das aktuelle Bild befinden sich wie auch die anderen aufgenommen Bilder im Verzeichnis */media/nistkasten*. Damit beim Aufruf der Webserver auch die richtige HTML-Datei dargestellt wird, muss der Webserver über die Datei /etc/lighttpd/lighttpd.conf entsprechend konfiguriert. Anschließend ist es möglich von jedem Computer, der sich im selben lokalem Netzwerk wie der Raspberry Pi-Server befindet, über die Ip-Adresse des Raspberry Pis, auf den Webserver zuzugreifen und das aktuelle Bild aus dem Nistkasten darstellen zu lassen.

VI. IMPLEMENTIERUNG

Für die Umsetzung des kameraüberwachten Nistkastens wurde ein Programm in Python3 geschrieben. Der Ablauf der Implementierung ist in Abbildung 9 dargestellt. Als erstes

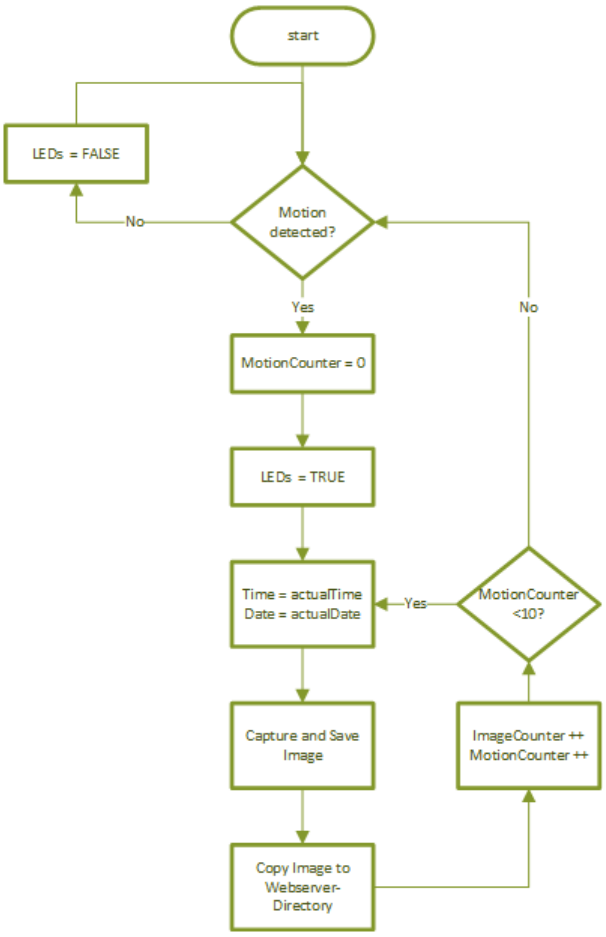


Fig. 9. Ablauf der Implementierung

wird auf eine Bewegungserkennung gewartet. Nachdem eine Bewegung erkannt wurde, sollen zehn Bilder hintereinander aufgenommenen erden. Dazu wird ein *MotionCounter*, der die Anzahl der Bilder nach einer Bewegungserkennung zählt, zunächst auf Null zurückgesetzt. Gleichzeitig werden die IR-LEDs angeschaltet, damit der Nistkasten beleuchtet ist und die Bilder beleuchtet aufgenommen werden können. Im Anschluss startet eine Schleife, in der die zehn Bilder aufgenommen werden. Als erstes werden in der Schleife das Datum und die Uhrzeit, welche in die Bildaufnahme geschrieben werden, aktualisiert. Daraufhin findet die Bildaufnahme und das Speichern des Bildes auf dem Raspberry Pi-Server statt. Das aufgenommene Bild wird zudem in das Verzeichnis des Webservers kopiert, sodass das aufgenommene Bild als aktuelle Bildaufnahme auf dem Webserver dargestellt werden kann. Danach werden der Zähler für die Durchnummerierung der Bildaufnahmen und für das Zählen der Bildaufnahmen um eins hochgezählt. Es wird anschließend kontrolliert, ob bereits zehn Bildaufnahmen seit der letzten Bewegungserkennung durchgeführt wurden oder nicht. Im Falle, dass noch keine zehn Bilder aufgenommen wurden, wiederholt sich die Schleife. Ansonsten wird wieder auf eine Bewegungserken-

nung gewartet und die IR-LEDs werden ausgeschaltet.

VII. Fazit

A. Ergebnisse

Der kameraüberwachte Nistkasten erfüllt seine Aufgaben mit einer Ausnahme wie erwartet. Lediglich der PIR-Sensor signalisiert, sobald das Kameramodul angeschlossen ist, durchgehend eine Bewegungserkennung. Sobald eine Bewegung vom PIR-Sensor erkannt wurde, beginnt die Bildaufnahmen mit dem NoIR Kamera Modul wie gewünscht. Auch das Speichern der Bilder auf dem zweiten Raspberry Pi, auf dem der NSF-Server eingerichtet ist, funktioniert problemlos. Weiterhin ist es möglich auf den Webserver zu zugreifen und das aktuellste Bild anzeigen zu lassen. Zudem ist die Bildqualität sehr gut. Dies liegt unter anderem daran, dass die zwei IR-LEDs den dunklen Nistkasten sehr gut mit IR-Licht bestrahlen. Ein weiter Vorteil war die die kompakte Bauweise des Nistkastens. Die Elektronik war einfach unterzubringen. Insgesamt ist der Raspberry Pi und das NoIR-Kameramodul sehr gut für die Umsetzung eines kameraüberwachten Nistkastens geeignet. Dies liegt unter anderem an den vielen Bibliotheken, die die Programmierung stark unterstützen.

Vor der Inbetriebnahme des Nistkastens sollten noch folgende Aspekte berücksichtigt werden:

- **Befestigung der Elektronik:** Da es sich nur um einen Prototypen handelt, wurde die Elektronik nicht fest montiert, sondern nur mit Reißzwecken befestigt. Für den Einsatz sollte die Elektronik entweder festgeschraubt oder -geklebt werden.
- **Schutz der Elektronik vor den Vögel:** Die Elektronik sollte vor den Vögeln gesichert werden, da diese sehr neugierig sein können und die Elektronik und insbesondere die Verbindungen beschädigen könnten.
- **Geruch des Nistkastens:** Der verwendete Nistkasten riecht sehr stark nach dem Imprägnierspray, das das Holz wasserfest macht. Abhängig von der Empfindlichkeit von Vögeln sollte evtl. überlegt werden, diesen Nistkasten gegen einen weniger stark riechenden Nistkasten auszutauschen.
- **Feuchtigkeit:** Bevor der Nistkasten in Betrieb genommen wird, sollte gewährleistet sein, dass keine Feuchtigkeit in den Nistkasten kommen kann. Andernfalls würden die Vögel wegbleiben und die Elektronik Schaden erleiden.

B. Ausblick

Es sind noch verschiedene Erweiterungs- und Verbesserungsmöglichkeiten vorhanden wie:

- **Weitere Sensoren:** Will man die Vögel und deren Verhalten genauer untersuchen, so könnte man weitere Sensoren an den Nistkasten anbauen. Man könnte einen Sensor zur Berührungserkennung an die Sitzstange des Nistkastens oder eine zusätzliche Kamera, die das Einflugloch filmt, anbringen.
- **Echtzeituhr:** Da der Raspberry Pi keine Echtzeituhr besitzt und auch nicht mit dem Internet verbunden ist, kann die Uhrzeit und das Datum nicht zuverlässig dargestellt und in das Bild geschrieben werden. Daher wäre es von Vorteil eine zusätzliche Echtzeituhr an den Raspberry Pi anzuschließen.
- **Ton:** Das Raspberry Pi-NoIR-Kameramodul hat kein Mikrofon integriert und kann daher keinen Ton aufnehmen. Für die Tonaufnahme müsste ein Mikrophon an den Raspberry Pi angeschlossen werden. Damit wäre es möglich das ”Zwitschern” der Vögel aufzunehmen.

References

[1] Meistersängerin *NABU*. Dezember 2014, available at http://www.nabu.de/aktionenundprojekte/stundedergartenvoegel/die40haeufigstengartenvoegel/03656.html

[2] Kohlmeise *LBV*. Dezember 2014, available at http://wiki.lbv.de/kohlmeise.html#c3509

[3] Die Kohlmeise *Brodowski Fotografie*. Dezember 2014, available at http://www.brodowski-fotografie.de/beobachtungen/kohlmeise.html

[4] Datenblatt - Passive-Infrarot-Bewegungsmelder Low Power-Ausführung *Conrad*. Januar 2015, available at http://www.produktinfo.conrad.com/datenblaetter/150000-174999/172526-da-01-de-PIR_SMD_MODUL_3_5_V_80_UA.pdf

[5] Nistkästen richtig aufhängen *Mein Schöner Garten*. Dezember 2014, available at http://www.mein-schoener-garten.de/de/meinschoenesland/natur_tiere/nistkaesten-richtig-aufhaengen-22891

[6] The MagPi Issue 18 *The MagPi*. Dezember 2014, avaiable at http://www.themagpi.com/issue/issue-18/

[7] The MagPi Issue 14 *The MagPi*. Dezember 2014, avaiable at http://www.themagpi.com/issue/issue-14/

[8] E.F. Engelhardt *Foto und Video mit Raspberry Pi*. 1. Ausgabe, München: Franzis Verlag, 2014

[9] H. Bernauer *Hannahs 25 Raspberry Pi Server*. 1. Ausgabe, München: Franzis Verlag, 2014

[10] Dr. M. Kofler, C. Kühnast und C. Scherbeck *Raspberry Pi - Das umfassende Handbuch*. 1. Ausgabe, Bonn: Galileo Press, 2014

[11] Documentation picamera *Picamera*. Monat Jahr, available at http://picamera.readthedocs.org/en/release-1.9/

[12] Raspberry Pi Guide *Raspberry Pi Guide*. Januar 2015, available at http://raspberrypiguide.de/

[13] Meise mit Pils *Natur-Portrait*. Januar 2015, available at http://www.natur-portrait.de/foto-58171-meise-mit-pilz.html

[14] interfaces *ubuntuusers*. Januar 2015, available at http://wiki.ubuntuusers.de/interfaces

[15] WiFi Ad-hoc Network *Debian*. Januar 2015, available at https://wiki.debian.org/WiFi/AdHoc

[16] Vogelnest Nistkasten *Hit-Meister*. Januar 2015, available at http://www.hitmeister.de/product/301346795/

[17] Motion *Wikipedia*. Januar 2015, available at http://en.wikipedia.org/wiki/Motion_(surveillance_software)

[18] Setting Up an NFS Server *SourceForge*. Januar 2015, available at http://nfs.sourceforge.net/nfs-howto/ar01s03.html

[19] SID1050M Datasheet *AllDataSheet*. Januar 2015, available at http://www.alldatasheet.com/datasheet-pdf/pdf/168380/SANKEN/SID1050M.html

[20] EXT4 Festplatte am Raspberry Pi mounten und formatieren *ZNIL-WIKI*. Januar 2015, available at http://znil.net/?title=EXT4_Festplatte_am_Raspberry_Pi_mounten_und_formatieren

[21] Raspbian: NFSv4 Server einrichten *Good To Know Database*. Januar 2015, available at http://www.gtkdb.de/index_36_1976.html

[22] NFS Server *Wel!s Blog*. Januar 2015, available at http://www.welzels.de/blog/projekte/raspberry-pi/low-budget-nas-mit-einem-raspberry-pi/pi-nas-datei-server-und-zubehor/

[23] NFS *ubuntuusers*. Januar 2015, available at http://wiki.ubuntuusers.de/NFS

[24] Network File System *Wikipedia*. Januar 2015, available at http://de.wikipedia.org/wiki/Network_File_System

[25] Widerstandsrechner *LumiTronix*. Januar 2015, available at http://www.leds.de/Widerstandsrechner

[26] lighttpd *ubuntuusers*. Januar 2015, available at http://wiki.ubuntuusers.de/lighttpd

Erweiterung einer mobilen und autonomen Wasserqualitätsprüfstation

Stephan Balduin
Universität Oldenburg
Department für Informatik
Abt. Systemsoftware und
verteilte Systeme
Email: stephan.balduin@uni-oldenburg.de

Carsten Krüger
Universität Oldenburg
Department für Informatik
Abt. Systemsoftware und
verteilte Systeme
Email: carsten.krueger@uni-oldenburg.de

***Zusammenfassung*—In der gemeinsamen Arbeit von [1] wurde eine mobile Wasserqualitätsprüfstation als Prototyp entwickelt. Diese sammelt autonom Messdaten über einen frei wählbaren Zeitraum. Der Funktionsumfang der Messtation beschränkt sich auf die Ermittlung von Temperatur und pH-Wert. In der vorliegenden Arbeit wurde der Funktionsumfang um zusätzliche Sensoren erweitern und das System optimiert. Teile dieser Optimierungen sind eine leistungsstärkere Stromversorgung, softwareseitiges sequentielles Herunterfahren und eine intelligente Abweichungskorrektur der Messzeitpunkte.**

I. Einleitung

Die Bestimmung der Wasserqualität von größeren Gewässern wird mittels containergroßer Messstationen durchgeführt. Diese liefern täglich und kontinuierlich Daten. Kleinere Gewässer, wie zum Beispiel Moorgebiete in der Umgebung von Oldenburg, werden allerdings nicht unbedingt auch von einer derartigen Messstation betreut. Da die Beschaffenheit des Wassers für die dort beheimateten Pflanzen und Tiere dennoch eine entscheidende Rolle spielt, muss die Wasserqualität auf eine andere Weise bestimmt werden. Dieser Vorgang kann sehr zeitaufwendig sein, da händisch Wasserproben genommen und analysiert werden müssen. In der gemeinsamen Arbeit von [1] wurde prototypisch eine tragbare Messstation mit dem Ziel entwickelt, diesen Vorgang zu automatisieren. Mit dieser Messstation können Messdaten über einen frei wählbaren Zeitraum und mit beliebiger Häufigkeit gesammelt werden. Im Rahmen der vorliegenden Arbeit soll das Funktionsspektrum dieser Messstation erweitert werden.

II. Über die Wasserqualität

Soll die Wasserqualität bestimmt werden, gilt es zunächst, einige Fragen zu beantworten. Die erste Frage ist, welche Art von Wasser untersucht werden soll. Für Trinkwasser müssen zum Beispiel andere Gütekriterien erfüllt werden, als für Flusswasser. Die zweite Frage ist, welche Gütekriterien für das entsprechende Gewässer relevant sind und welche Messwerte nicht über- oder unterschritten werden dürfen. Die hier vorliegende Messstation wurde hauptsächlich mit dem Ziel entwickelt, natürliche Gewässer zu überwachen. Für die in diesen lebenden Pflanzen und Tiere sind besonders die Wassertemperatur, der Sauerstoffgehalt, der pH-Wert und die elektrische Leitfähigkeit von Bedeutung. Diese haben sich aufgrund der vorherrschenden, ökologischen Bedingungen entwickelt, sodass größere Abweichungen der Normalwerte destruktive Auswirkungen auf diese Lebewesen haben kann. Neben dem offensichtlichen Grund, dass die Lebewesen zum Beispiel bei zu niedriger Temperatur erfrieren, beschleunigt eine steigende Wassertemperatur biochemische Reaktionen im Wasser. Die elektrischen Leitfähigkeit bietet eine relativ einfache Abschätzung, ob ein Gewässer sauber oder verunreinigt ist. Eine hohe Leitfähigkeit lässt auf viele im Wasser gelöste Zusatzstoffe schließen, was einer starken Verunreinigung entspricht. Ebenso gibt der pH-Wert Auskunft über die Qualität des Wassers. Während reines Wasser einen pH-Wert von 7 hat, bedeuten höhere oder niedrigere Werte größere Konzentration von Zusatzstoffen im Gewässer. Der Sauerstoffgehalt ist für die Lebewesen im Wasser ebenfalls von sehr großer Bedeutung, da auch diese den Sauerstoff zum Überleben benötigen. Ist die Konzentration zu gering, können die Organismen nicht mehr ausreichend versorgt werden. Die Löslichkeit des Sauerstoffs im Wasser nimmt mit steigender Temperatur ab.

III. Systemarchitektur

Zunächst soll die bestehende Architektur untersucht werden, um die Komponenten deutlich zu machen, die erweitert werden sollen. Dabei werden Hardware- und Software-Architektur jeweils getrennt betrachtet.

A. Hardware-Architektur

In Abbildung 1 ist der schematische Aufbau der Hardware-Architektur zu sehen. Die Kernkomponente des Systems ist ein Raspberry Pi vom Typ B [4]. Dieser steuert alle internen Abläufe und ist gleichzeitig die Schnittstelle zu den Hardware-Komponenten. Der Raspberry Pi wurde mit einem externen WLAN-Modul ausgestattet. Dieses Modul garantiert einen einfachen Zugriff auf das System über ein Endgerät, wie zum Beispiel ein Smartphone oder ein Notebook. Auf diese Weise kann ein Benutzer ohne tiefergehende Kenntnisse Messaufträge anlegen und organisieren, sowie Messdaten abrufen.

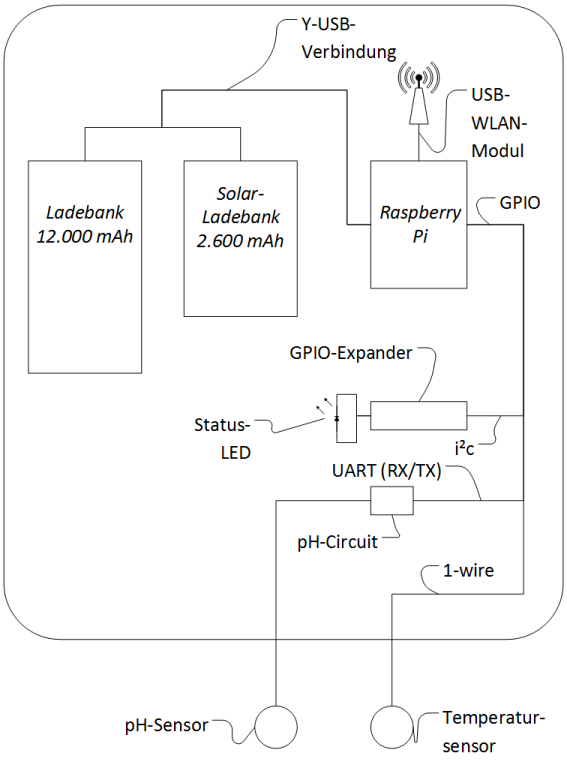


Abbildung 1. Schematischer Aufbau der Hardware-Architektur

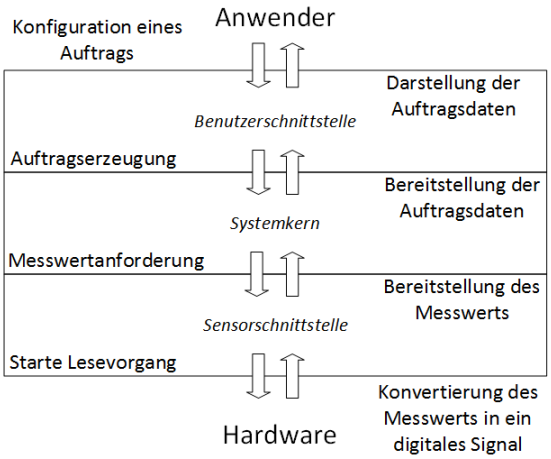


Abbildung 2. Schichtenmodell der Systemsoftware

An die GPIO-Pins des Raspberry Pi ist ein wasserfester Temperatursensor mit der Bezeichnung DS18B20 [5] angeschlossen, der das 1-wire Protokoll [6] verwendet. Außerdem ist ein pH-Wert-Sensor angeschlossen, der über das UART-Protokoll [7] angesteuert wird. Bei diesem handelt es sich um einen Sensor mit BNC-Anschluss, wie er auch bei Multifunktionsmessgeräten zum Einsatz kommt. Durch die Art und Weise, wie die pH-Wert-Messung durchgeführt wird, liefert der Sensor ein ziemlich schwaches Signal, was die Verwendung eines Signalverstärkers notwendig macht. Hierbei wurde auf die Lösung eines professionellen Anbieters [8] gesetzt. Dieser bietet vorgefertigte Schaltkreise an, die zum Einen die Signalverstärkung übernehmen und zum Anderen die Messwerte für das UART-Protokoll aufbereiten. Des Weiteren wurde ein GPIO-Expander mit der Bezeichnung MCP23017 [9] angeschlossen, der das I^2C-Protokoll [10] verwendet. Die Stromversorgung des Systems wird über zwei transportable Ladestationen organisiert, mit denen man auch Smartphones wieder aufladen könnte. Diese Ladebänke sind über ein Y-USB-Kabel mit dem Raspberry Pi verbunden. Dadurch wird es ermöglicht, eine der Ladebänke im laufenden Betrieb zu entfernen und beispielsweise durch eine andere zu ersetzen. Die Gesamtkapazität der verwendeten Ladebänke beträgt 14.600 mAh. Dies reicht aus, um den Raspberry Pi sechzehn bis achtzehn Stunden lang mit Energie zu versorgen.

B. *Software-Architektur*

Auf dem Raspberry Pi ist das Raspbian-Betriebsystem installiert, auf dem eine speziell für die Messstation entwickelte Software läuft. Wie in Abbildung 2 zu sehen ist, setzt diese sich aus drei Schichten zusammen. Über der Hardware ist die Schicht der Sensorschnittstelle, die für die Verwaltung und Ansteuerung der Sensoren zuständig ist. Darüber befindet sich die Systemkernschicht, in der die Logik implementiert ist. Die dritte Schicht ist die Benutzerschnittstelle, die in Abbildung 2 als oberste Schicht dargestellt ist. Die Ansteuerung der Sensoren wurde unter Verwendung des von [2] entwickelten und in [3] erweiterten Rahmenwerks für Sensor- und Aktuatorknoten implementiert. Der Aufruf ist dabei folgendermaßen: das System stellt eine Leseanfrage mit Angabe einer Sensor-ID an das Rahmenwerk. Dieses wählt den entsprechende Sensorknoten aus und fordert eine Neuberechnung der Sensorwerte an. Da ausschließlich Sensoren (und keine Aktuatoren) verwendet werden, findet an dieser Stelle der Zugriff auf den entsprechenden Sensor statt. Anschließend werden die Messdaten über die Infrastruktur des Rahmenwerks zurück geliefert und können vom System verwendet werden.

Die Benutzerschnittstelle bietet die Möglichkeit, bestimmte Eingaben über eine Weboberfläche zu tätigen und Messergebnisse anzeigen zu lassen. Von hier aus können Aufträge erstellt und konfiguriert werden. Der Anwender kann einen Sensor auswählen und Startzeit, Häufigkeit der Messung und Endzeit definieren. Diese Angaben werden in einer Datenstruktur gespeichert und an das System übergeben. Ebenso kann sich der Anwender ermittelte Daten der Aufträge anzeigen lassen oder diese von der Weboberfläche auf sein Endgerät kopieren.

Doch wie werden Aufträge und Messdaten intern verwaltet? Jeder vom Anwender erstelle Auftrag bekommt einen Leichtgewichtsprozess (engl. *thread*) zugewiesen. Dieser wird zunächst bis zum Startzeitpunkt, sofern in der Zukunft liegend, schlafen gelegt. Zu Beginn des Messintervalls wird der Leichtgewichtsprozess aufgeweckt und stößt einen Messvorgang an. Sobald der Messwert vorliegt, wird dieser in eine Datenbank

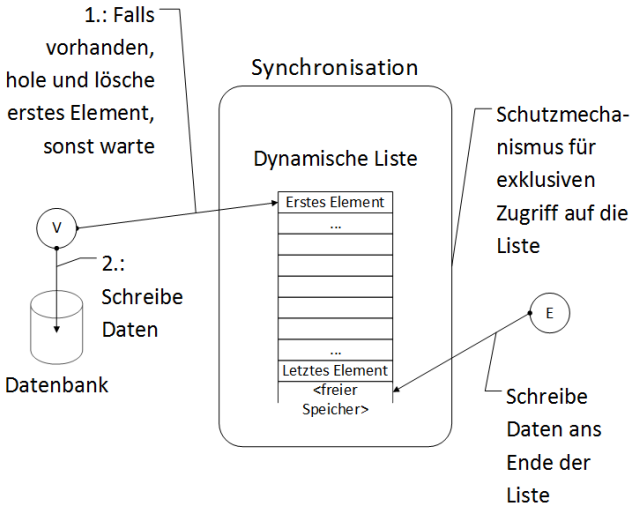


Abbildung 3. Synchronisation für das Schreiben in die Datenbank. Aufträge (E) schreiben in die gemeinsame Liste, der Verbraucherprozess (V) überträgt die Datensätze in die Datenbank. Der Zugriff auf die Liste wird mittels gegenseitigem Ausschluss geschützt

geschrieben und der Prozess bis zum nächsten Messzeitpunkt wieder schlafen gelegt.

Da aber mehrere Aufträge zeitgleich ablaufen können, bedeutet dies, dass auch mehrere Prozesse gleichzeitig aktiv sind, die miteinander synchronisiert werden müssen. Es darf beispielsweise immer nur ein Prozess auf die Datenbank zugreifen oder einen bestimmten Sensor ansteuern. Der Zugriff auf die Sensoren wird mit gegenseitigen Ausschluss realisiert. Bei Datenbankzugriffen mit gegenseitigem Ausschluss entstünden Verzögerungen der Messintervalle, wenn jeder Auftrag den Datenbankzugriff selbstständig durchführen würde. Deshalb wird eine Listenstruktur verwendet, die in Anlehnung an das klassische Erzeuger/Verbraucher-Problem einen geteilten Speicherbereich darstellt. Die Auftragsprozesse sind die Erzeuger, die kontinuierlich Datensätze in die Liste einreihen. Ein einzelner Verbraucherprozess arbeitet diese Liste von vorne ab und schreibt die Datensätze in die Datenbank. Auf diese Weise müssen sich die einzelnen Prozesse nur beim Zugriff auf den Listenspeicher synchronisieren (vgl. Abbildung 3).

IV. Hardware-Erweiterungen

In diesem Abschnitt werden die in diesem Seminar geplanten Hardware-Erweiterungen vorgestellt.

A. Integration einer Realzeituhr

Der Raspberry Pi besitzt keine eigene Realzeituhr, sondern bezieht die aktuelle Zeit mittels *Network Time Protocol* über das Internet. Wenn der Raspberry Pi nicht mit dem Internet verbunden ist, wird die Systemzeit nicht aktualisiert. Die Software der Messstation stellt für diesen Fall eine Funktion bereit, mit der die Systemzeit manuell eingestellt werden kann. Da dies jedoch unkomfortabel ist, wurde die Messstation um eine

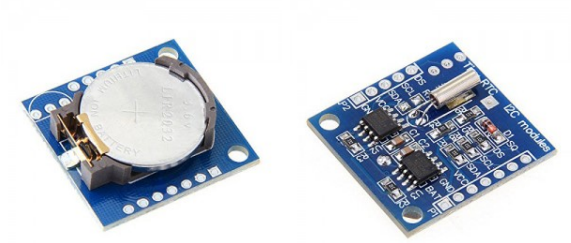


Abbildung 4. Realzeituhr DS1307

Realzeituhr erweitert. Die in Abbildung 4 dargestellte Realzeituhr mit der Bezeichnung DS1307 [11] verwendet das I^2C-Protokoll und lässt sich deshalb wie in Abbildung 5 mit dem I^2C-Bus des Raspberry Pis verbinden. Anschließend muss der Raspberry Pi entsprechend konfiguriert werden, sodass die Systemzeit mit der Realzeituhr synchronisiert wird. Dazu wurden die Systemsoftware dahingehend erweitert, dass die Systemzeit bei jedem Startvorgang die Uhrzeit der Realzeituhr abfragt und übernimmt.

B. Anschluss weiterer Sensoren

In der bereits bestehenden Architektur waren bereits ein Temperatursensor und ein pH-Wert-Sensor integriert. Weitere Sensoren für Sauerstoffgehalt, Leitfähigkeit und Redoxpotenzial erweitern die bestehende Architektur. Ähnlich wie der pH-Wert-Sensor werden auch diese Sensoren über einen BNC-Anschluss angeschlossen und liefern nur ein sehr schwaches Signal. Deshalb werden für jeden Sensortyp ebenfalls eigene Signalverstärker (-Schaltkreise) benötigt. Mit den drei neuen Sensoren besitzt das System bereits vier Sensoren, die das UART-Protokoll verwenden. Es ist daher notwendig, einen Multiplexer für die UART-Datenleitungen zu verwenden, damit jeder Sensor separat angesteuert werden kann. Es wurde ein Carrier-Board vom Hersteller der Schaltkreise verwendet, dass neben Aufsätzen für die Schaltkreise und Anschlüssen für BNC-Steckverbindungen auch eine Multiplexer-Funktion bereitstellt. Die benötigten Steuerleitungen werden durch die Ausgänge des GPIO-Expanders realisiert. Somit ist es möglich, eine große Anzahl Sensoren über das UART-Protokoll anzusteuern, da mehrere Multiplexer über GPIO-Expander, die I^2C verwenden, angesteuert werden können. Die Implementierung der Sensortreiber ist ähnlich zu dem des pH-Wert-Sensors, lediglich die Ansteuerung der richtigen Steuerleitungen musste zusätzlich definiert werden. Durch die Erweiterbarkeit der Software war nur die Implementierung der Treiber in der Sensorschnittstelle und die Registrierung der neuen Sensortypen im System notwendig. In Abbildung 5 ist die erweiterte Architektur dargestellt.

C. Stromversorgung

Das System besitzt eine Stromversorgung, die über zwei autonome Ladestation mit jeweils 5 V USB-Ausgängen verfügt. Eine der bisher verwendeten Ladebänke besitzt 12.000 mAh, während die andere Ladebank mit 2.600 mAh eine deutlich

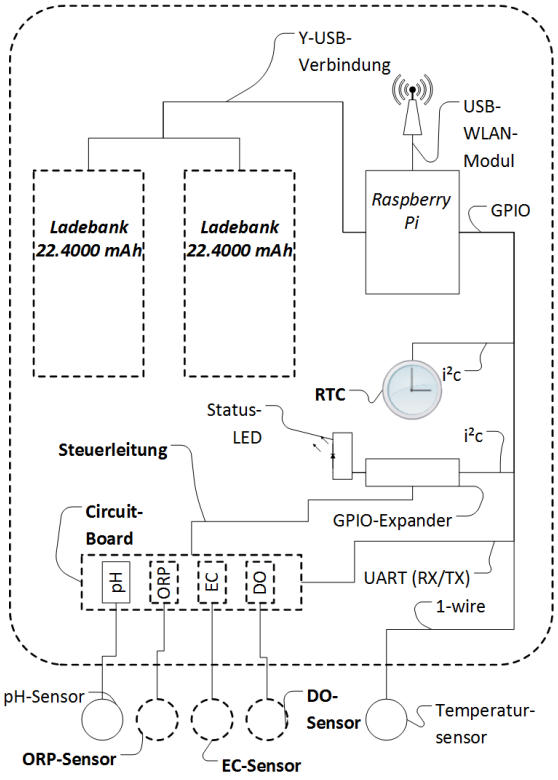


Abbildung 5. Erweiterte Hardware-Architektur. Änderungen gegenüber der vorherigen Architektur aus Abbildung 1 werden durch gestrichelte Linien dargestellt. Die Abkürzungen bei den Sensorbezeichnungen stehen für *oxidation reduction potential* (ORP), *electrical conductivity* (EC) und *dissolved oxygen* (DO).

geringere Kapazität besitzt. Um eine längere Stromversorgung zu gewährleisten, werden diese durch Ladebänke ersetzt, die mit jeweils 22.400 mAh über eine größere Kapazität verfügen. Der Verbrauch durch den Raspberry Pi des Modells B liegt ungefähr bei 700 mA, ein zusätzlicher Energieverbrauch entsteht durch das USB-WLAN-Modul mit bis zu 130 mA und die Sensoren. Der benötigte Stromverbrauch der Sensoren liegt im Bereich von 1 mA bis 15 mA. Insgesamt errechnet sich ein theoretischer Verbrauch von maximal 862 mA. Bei einer Gesamtkapazität von 44.800 mAh ergibt sich eine Laufzeit von etwa 52 Stunden. Dies entspricht einer Steigerung von ungefähr 300 %.

D. Gehäuse

Das bisherigen Gehäuse wurde durch die alte Hardware maximal ausgenutzt und es ist nicht möglich gewesen, weitere Bauteile zu integrieren. Die Einbindung von drei weiteren Sensoren, größere Ladestationen und eine größere Lochrasterplatine sorgen dafür, dass ein neues Gehäuse benötigt wird. Dieses entspricht, ebenso wie das vorherige Gehäuse, den IP44-Schutzanforderungen nach DIN 40 050, Teil 9. Zusätzlich wurde die Anzahl Ausgänge auf fünf angehoben, die jeweils mit Kabelanbaustutzen vor Wassereintritt geschützt sind. Mit einer Länger von 31 cm, einer Breite von 23 cm und einer Höhe von 14 cm ist das Kunststoffgehäuse immer noch kompakt genug, um von einem Benutzer transportiert zu werden, ohne Hilfsmittel zu benötigen.

V. Software-Erweiterungen

In diesem Abschnitt werden die geplanten und durchgeführten Software-Erweiterungen vorgestellt.

A. Dynamische Sensorerkennung

Die bisherige Architektur hat vorausgesetzt, dass alle unterstützten Sensoren mit dem System verbunden sind. Die Erweiterungen im Rahmen dieses Seminars erhöhen die Anzahl der unterstützten Sensoren auf fünf. Um die Anforderung der Flexibilität zu erfüllen, soll es möglich sein, eine individuelle Auswahl dieser Sensoren zu selektieren. Es könnten beispielsweise nicht alle Sensoren vorhanden sein oder bewusst nur eine Teilmenge der unterstützten Sensoren verwendet werden. Daher ist es wünschenswert, wenn nicht angeschlossene Sensoren vom System einheitlich und dynamisch erkannt werden könnten. Auf diese Weise ließen sich Messungen ignorieren, die einen nicht angeschlossenen Sensor verwenden. Diese Situation könnte auftreten, wenn beispielsweise ein Auftrag für den pH-Wert angelegt wurde, das System heruntergefahren, dann der Sensor entfernt und anschließend das System wieder gestartet wird. Das System sollte dahingehend erweitert werden, dass ein nicht angeschlossener Sensor erkannt wird. Dazu wurden die Rückgabewerte der Lesefunktionen entsprechend angepasst, sodass diese eindeutig als Fehlerwert erkannt werden. Konkret wurde der Wert `INT_MIN` gewählt, da dieser weit genug von jedem möglichen Rückgabewert entfernt ist. Beim Systemstart, und immer wenn der Benutzer einen neuen Auftrag anlegen will, wird eine Überprüfungsfunktion aufgerufen, die bei jedem Sensor eine Messung durchführt, ohne dass diese in der Datenbank aufgenommen wird. Wurde beim Lesen der Wert `INT_MIN` ermittelt, bedeutet dies, dass der entsprechende Sensor nicht erreichbar ist. Darüber hinaus werden alle Messungen mit dem Messwert `INT_MIN` ignoriert, ohne dass entsprechende Aufträge angehalten werden. Auf diese Weise können Aufträge ihre Arbeit dynamisch wieder aufnehmen, sobald der Sensor erneut erreichbar ist.

Während die Umsetzung softwareseitig ohne größere Probleme durchgeführt werden konnte, bereitete die Hardware an dieser Stelle Schwierigkeiten. Die Schaltkreise liefern, entgegen der Spezifikation ihres Datenblatts, keinen Fehlercode, wenn kein Sensor angeschlossen ist. Stattdessen ist der Rückgabewert nicht definiert, aber durchaus im gültigen Wertebereich, sodass diese Werte nicht genutzt werden können, um das Fehlen eines Sensors festzustellen. Dies hat zur Folge, dass die dynamische Sensorerkennung nur mit der Einschränkungen funktioniert, dass sowohl der Sensor, als auch der dazugehörige Schaltkreis vom System getrennt sein müssen, damit das Fehlen erkannt werden kann.

B. Abweichungskorrektur und Sequentielles Herunterfahren

Die bestehende Auftragsverwaltung wurde um Aspekte wie sequentielles Herunterfahren und Abweichungskorrekturen

der Messzeitpunkte erweitert. Ein Auftrag wird jeweils in einem eigenen Thread gestartet, der während den Wartezeiten zwischen zwei Messungen schlafen gelegt wird. Bezogen auf die aktuelle Systemzeit, der letzten Messung und der Auftragsperiode lässt sich der jeweils nächste Messzeitpunkt ermitteln. Verzögerungen durch die Datenermittlung oder den Kontextwechsel der Threads machen sich als Abweichungen bemerkbar und müssen nach jeder Datenermittlung korrigiert werden. Eine einfache Addition der Periode auf die Zeit der letzten Messung oder der bestehenden Systemzeit führt zu aufsummierten Abweichungen, wodurch die Ergebnisse in falschen Intervallen gemessen werden. Die bestehende Abweichung zwischen zwei Messungen lässt sich durch Division mit Rest zwischen der Startzeit und der aktuellen Systemzeit ermitteln. Dieser Rest wird bei der Bestimmung für den nächsten Messzeitpunkt mit einbezogen. In Listing 1 ist in vereinfachter Form der Algorithmus der Auftragsverwaltung dargestellt.

```
1  next_measure : INTEGER := 0;
2  db : DATABASE;
3  cur_time : INTEGER;
4  periode : INTEGER;
5  end_time : INTEGER;
    -- Ende der Laufzeit des Auftrags
6
7  start_job ( ) : VOID
8  BEGIN
9   next_measure := db.get_next_measure;
10  sleep_for ( . . . );
     -- Thread legt sich schlafen
     -- bis erste Messung beginnt
11
12  WHILE cur_time < end_time LOOP
13   IF( next_measure + periode )
      < cur_time THEN
14     -- Korrigiere Abweichung und
15     -- setze next_measure neu
16   ELSE
17     -- Addiere periode minus Abweichung
18     -- zu next_measure
19   END IF;
20
21   IF global_next_measure
      > next_measure THEN
22    global_next_measure := next_measure;
23   END IF;
24
25   db.insert( sensor.read ( ) );
26   db.set_next_measure ( next_measure );
27   IF( next_measure - cur_time ) > 0 THEN
28    sleep_for ( . . . );
29   END IF;
30  END LOOP;
31 END start_job;
```

Listing 1. Algorithmus zur periodischen Messdatenerfassung

Die Systemsoftware ist nur aktiv, wenn eine Messung durchgeführt wird, ansonsten wartet sie die Verwendung der Weboberfläche. In Phasen, wo diese nicht genutzt wird und keine unmittelbar bevorstehenden Messungen existieren, wird die Stromversorgung unnötig belastet. Sequentielles Herunterfahren nutzt diese inaktiven Phasen, um das System Herunterzufahren und Strom einzusparen. Dazu muss ermittelt werden wie viel Zeit zwischen der aktuellen Systemzeit und allen kommenden Messungen liegt, sowie eine Überprüfung der Weboberfläche auf Aktivität. Die Aktivität auf der Weboberfläche lässt sich über den Webserver durch die Existenz aktiver Verbindungen (engl. *sessions*) feststellen. Zur Ermittlung der globalen nächsten Messung verwendet das System eine geschützte Variable, auf die jeder Thread exklusiven Zugriff hat. Jeder Auftrag vergleicht den Zeitpunkt der eigenen nächsten Messung mit dem Zeitpunkt globalen nächsten Messung. Ist der Zeitpunkt in der Variablen später, wird dieser durch den Messzeitpunkt des Auftrags ersetzt. Dieser Vorgang sorgt dafür, dass immer der frühste Zeitpunkt für die nächste Messung in der Variablen enthalten ist. In einem weiterem Thread sorgt ein `SHUT_ DOWN_ HANDLER` für die zeitliche Auswertung und kann entscheiden, ab wann sich ein Herunterfahren lohnt. An ein System, welches das sequentielle Herunterfahren realisiert, kann der Zeitpunkt zum Reaktivieren übergeben werden. Ein vereinfachter Algorithmus zum sequentiellen Herunterfahren ist in Listing 2 zu sehen.

```
1  global_next_measure : INTEGER;
2  sessions : INTEGER;
3  sleep_time : INTEGER := 120;
    -- Zeit in Sekunden, die der Thread
    -- nach Neustart wartet
4  minimum_time : INTEGER := 300;
5  cur_time : INTEGER;
6
7  shut_down_handler ( ) : VOID
8  BEGIN
9   sleep_for( sleep_time );
10  WHILE true LOOP
11   IF( global_next_measure - cur_time )
      > minimum_time AND sessions <= 0 THEN
12     -- Starte sequentielles Herunter-
       -- fahren fuer global_next_measure -
       -- cur_time Sekunden
13   ELSE
14    sleep_for( global_next_measure
       -cur_time );
15   END IF;
16  END LOOP;
17 END shut_down_handler;
```

Listing 2. Algorithmus zum sequentiellen Herunterfahren

C. Optimierung der Benutzeroberfläche

An der Benutzeroberfläche des Systems wurden ebenfalls einige Erweiterungen vorgenommen. Diese betreffen die Möglichkeiten bei der Betrachtung der Messdaten. Das System bietet zwei mögliche Darstellungen in Tabellenform und eine grafische Ansicht, in der die Messdaten in einem Koordinatensystem visualisiert werden. Die grafische Darstellung war bisher statisch: es wurde für jeden angeschlossenen Sensor ein Graph angezeigt, vorausgesetzt, dass ein Auftrag bereits

Messdaten für diesen Sensor ermittelt hat. Während dies bei zwei Arten von Messwerten, Temperatur und pH-Wert, überschaubar ist, ist es bei mittlerweile fünf Arten von Messwerten wünschenswert, sich nur bestimmte Daten anzeigen zu lassen. Deshalb wurde die Grafik um die Möglichkeit erweitert, bestimmte Messreihen ein- oder auszublenden. Außerdem wurde die zweite Y-Achse aktiviert, an die sich dynamisch Messreihen binden lassen. So ist es zum Beispiel möglich, einen Graphen, der sich an der linken Y-Achse orientiert, für Temperaturwerte und einen zweiten Graphen, der sich an der rechten Y-Achse orientiert, für die elektrische Leitfähigkeit darzustellen (siehe Beispiel in Abbildung 6).

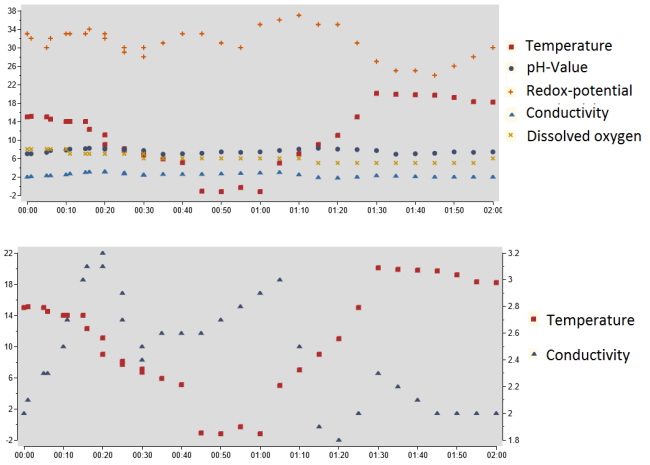


Abbildung 6. Anzeige der Messdaten: im oberen Bild alle Daten, orientiert an der linken Y-Achse; im unteren Bild nur Temperatur und Leitfähigkeit, jeweils an eine unterschiedlichen Y-Achse gebunden

D. Anpassung der Sprachversion

Bisher existierte für die Benutzeroberfläche lediglich eine deutsche Sprachversion. Damit das System zugänglicher für Menschen außerhalb des deutschen Sprachraumes wird, ist es sinnvoll, weitere Sprachversionen zur Verfügung zu stellen. Deshalb wurde die Benutzeroberfläche des Systems um eine Funktion erweitert, die verschiedene Sprachversionen unterstützt. Dabei wurde darauf geachtet, dass neben einer englischen Übersetzung auch weitere Sprachversionen hinzugefügt werden können. Auf der Weboberfläche lässt sich dynamisch eine dieser Sprachversionen auswählen, wodurch alle Textfelder in die entsprechende Sprache übersetzt werden. In der aktuellen Version existiert eine deutsche und eine englische Sprachversion.

VI. Feldtest

Die Funktionstüchtigkeit der Messstation sollte nach Abschluss der Erweiterungen in einem Feldexperiment getestet werden. Aus den Ergebnissen lassen sich Rückschlüsse ziehen, welche Aspekte der Messstation gut funktionieren und welche in zukünftigen Arbeiten weiter optimiert werden könnten. Außerdem soll anhand der erhobenen Messwerte ein Interpretationsversuch durchgeführt werden.

A. Vorbereitung

Zunächst musste ein Gewässer für die Messungen gefunden werden. Die Wahl fiel auf einen ungefähr zwei Meter breiten Graben. Dieser hatte zum Zeitpunkt des Feldtest saisonbedingt einen Wasserstand von weniger als einem halben Meter. Die Sensoren wurden, sofern notwendig, kalibriert und für jeden Sensor ein Messauftrag anlegt. Diese gingen jeweils von siebzehn bis elf Uhr drei Tage später und maßen minimal zeitversetzt alle fünfzehn Minuten. Die Gesamtdauer des Feldtests betrug mit 66 Stunden mehr als die theoretische Maximaldauer, um den tatsächlichen Stromverbrauch zu ermitteln. Nach den Vorbereitungen wurde die Messstation am Rand des Gewässers platziert und die Sensoren ins Wasser eingetaucht.

B. Ergebnisse

Am Ende der Durchführungszeit wurden die im Folgenden beschriebenen Feststellungen gemacht. Mit der Kapazität der Ladebänke konnte das System für die erwartete Zeit mit Energie versorgt werden. Die letzte aufgezeichnete Messung fand ungefähr um viertel nach zehn am Vorabend statt. Daraus ergibt sich eine Gesamtlaufzeit von 53 Stunden. Dies entspricht einem Stromverbrauch von ungefähr 850 mA und ist somit um 1,3 % geringer als der Sollwert. Außerdem wurde festgestellt, dass die Realzeituhr ausgefallen ist. Unklar ist allerdings, ob lediglich die Batterie der Uhr aufgebraucht ist oder es sich um einen technischen Defekt handelt. Des Weiteren waren die Messzeitpunkte in den vorgegebenen Intervallen und wiesen wenige bis keine Abweichungen auf. Geringe Verschiebungen wurden durch die Abweichungskorrektur behoben.

C. Interpretation der Messwerte

Zunächst muss erwähnt werden, dass der Wasserstand im Verlauf des Feldtests weiter abgenommen hat und sich deshalb nicht mehr alle Sensoren vollständig unter Wasser befunden haben. Aus diesem Grund sind die Messwerte weniger aussagekräftig, da unklar ist, ab welchem Zeitpunkt der Wasserstand die notwendige Höhe unterschritten hat. Werden spätere Werte außer Acht gelassen, lässt sich aus Abbildung 7 ablesen, dass sich die Wassertemperatur zwischen 1,5 °C und 2,5 °C bewegt. Dieser Wert ist niedrig, jedoch für die Jahreszeit nicht ungewöhnlich. Der pH-Wert liegt im Bereich pH $>$ 5,5 bis pH $<$ 6,0, was auf eine schwache Belastung mit Säuren schließen lässt. Nach [12] befindet sich dieser Wert an der unteren Grenze des durchschnittlichen Wertebereichs. Die elektrische Leitfähigkeit liegt hingegen mit Werten im Bereich von 1.500 μS/cm bis 2.200 μS/cm über dem durchschnittlichen Wert von 1.000 μS/m. Der Sauerstoffgehalt liegt unter dem für Lebewesen empfohlenen Minimalwert von 4 mg/l.

VII. Fazit

Die gegebene Wasserqualitätsprüfstation konnte größtenteils erfolgreich erweitert werden. Einige Änderungen waren allerdings so klein oder nebensächlich, dass sie bisher aus Platzgründen nicht erwähnt wurden. Deshalb werden alle Änderungen im Folgenden nochmals aufgelistet und kurz erläutert.

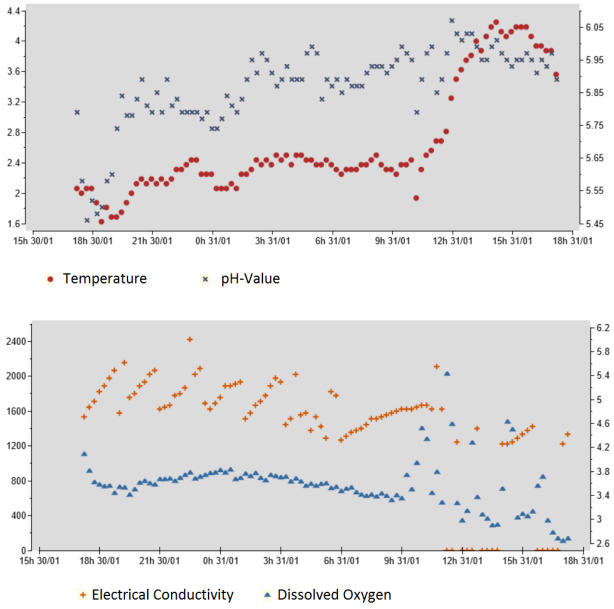


Abbildung 7. Messdaten des Feldtests. Temperatur (oben, in °C) und Leitfähigkeit (unten, in μS/cm) jeweils an die linke Y-Achse gebunden, pH-Wert (oben) und Sauerstoffgehalt (unten, in mg/l) jeweils an die rechte Y-Achse gebunden.

- Eine Realzeituhr sorgt dafür, dass der Raspberry Pi auch ohne aktive Internetverbindung die aktuellen Uhrzeit verwendet.
- Drei neue Sensoren, für Redox-Potenzial, elektrische Leitfähigkeit und ungelösten Sauerstoff, erweitern das System und ermöglichen es, die Wasserqualität genauer zu untersuchen.
- Für diese Sensoren wurde jeweils eine Kalibrierungsfunktion implementiert.
- Die vorherige Stromversorgung wurde durch eine leistungsfähigere ersetzt und verlängert die Einsatzdauer der Messstation auf 53 Stunden.
- Ein größeres Gehäuse bietet auch den neuen Komponenten ausreichend Platz, erfüllt aber weiterhin die Anforderungen der Tragbarkeit und Mobilität.
- Die dynamische Sensorerkennung konnte hingegen nicht vollständig realisiert werden. Hier muss noch eine Lösung ausgearbeitet werden, damit die Schaltkreise das Fehlen eines Sensors erkennen. Dies könnte zum Beispiel über einen BNC-Stecker realisiert werden, der beim Fehlen eines Sensors mit dem entsprechenden Anschluss verbunden wird. Unklar ist allerdings noch, ob dieser Stecker kurzschließen oder mit einem hohen Widerstand ausgestattet sein muss, um den gewünschten Effekt zu erzielen und gleichzeitig die Schaltkreise nicht zu beschädigen.
- Die Auftragsverwaltung wurde um eine Abweichungskorrektur erweitert, um aufsummierte Verzögerungen der Messzeitpunkte zu reduzieren.
- Auf Softwareseite wurden Vorbereitungen für sequentielles Herunterfahren des Systems getroffen.
- Die Benutzeroberfläche lässt sich nun in zwei Sprachen (Deutsch und Englisch) anzeigen. Außerdem können beliebige Sprachen hinzugefügt werden.
- Die Seite der Benutzeroberfläche, die zum Anzeigen einzelner Aufträge vorgesehen ist, wurde überarbeitet und funktioniert zuverlässiger als die vorherige Version.
- Auf der Benutzeroberfläche wurde die graphische Ansicht der Daten um kleine Funktionalitäten erweitert. Es lassen sich beliebige Messwertreihen ein- oder ausblenden und diese dynamisch an die linke oder rechte Y-Achse binden. Außerdem können diese als Linie oder als Punkte (Rechteck, Kreis, Kreuz, ...) angezeigt werden.
- Die Einstellungsseite der Benutzeroberfläche wurde aufgeräumt. Durch die neuen Sensoren sind weitere Kalibrierungsfunktionen hinzugekommen. Um die Übersichtlichkeit zu wahren, wurden diese auf eine neue Seite verschoben.

In zukünftigen Arbeiten könnte das System um Komponenten, wie ein GPS-Modul oder ein GSM-Modul erweitert werden. Das GPS-Modul könnte genutzt werden, um die Messungen mit Standortdaten zu versehen. Das GSM-Modul bietet die Möglichkeit, mit größerer Reichweite auf das System zuzugreifen oder systemkritische Informationen zu senden. Die Integration einer Hardware-Komponente, die sequentielles Herunterfahren realisiert, wäre eine weitere Möglichkeit zur Verbesserung des Systems.

Literatur

[1] S. Balduin und C. Krüger, *Planung und Realisierung einer Wasserqualitätsprüfstation auf Basis eines Raspberry Pi*, Bachelorarbeit. Universität Oldenburg, Department für Informatik, Abt. Systemsoftware und verteilte Systeme, 2014.

[2] Jens Krayenborg, *DSA: Dynamische Verwendung von Sensor- und Aktuatorknoten (HW/SW) zur Verbrauchsoptimierung von WSAN*, Bachelorarbeit. Universität Oldenburg, Department für Informatik, Abt. Systemsoftware und verteilte Systeme, 2013.

[3] Connor Fibich, *Konzeptionierung und Implementierung einer universellen Management-Schnittstelle für HW/SW-Sensoren und Aktuatoren*, Bachelorarbeit. Universität Oldenburg, Department für Informatik, Abt. Systemsoftware und verteilte Systeme, 2013.

[4] Raspberry Pi Foundation, *Raspberry Pi Model B*. http://www.raspberrypi.org/products/model-b/. Zuletzt besucht am: 25.01.15

[5] Adafruit, *Waterproof DS18B20 Digital Temperature Sensor*. http://www.adafruit.com/product/381. Zuletzt besucht am: 25.01.15

[6] Maxim Integrated Products, Inc., *1-Wire Communication Through Software*. http://www.maximintegrated.com/en/app-notes/index.mvp/id/126. Zuletzt besucht am: 25.01.15

[7] Wikipedia, *Universal Asynchronous Receiver Transmitter*. http://de.wikipedia.org/wiki/Universal_Asynchronous_Receiver_Transmitter. Zuletzt besucht am: 25.01.15

[8] Atlas Scientific LLC, *EZO pH Circuit*. http://www.atlas-scientific.com/product_pages/circuits/ezo_ph.html?. Zuletzt besucht am: 25.01.15

[9] Microchip Technology Inc., *GPIO Expander MCP23017 Datasheet*. http://ww1.microchip.com/downloads/en/DeviceDoc/21952b.pdf. Zuletzt besucht am: 25.01.15

[10] telos Systementwicklung GmbH, *Der I2C-Bus*. www.i2c-bus.org/de/i2c-bus/. Zuletzt besucht am: 25.01.15

[11] Maxim Integrated Products, Inc., *Real-time clock DS1307 Datasheet*. http://datasheets.maximintegrated.com/en/ds/DS1307.pdf. Zuletzt besucht am: 25.01.15

[12] Institut für Umweltverfahrenstechnik, Universität Bremen, *pH-Wert*. http://www.wasser-wissen.de/abwasserlexikon/p/phwert.htm. Zuletzt besucht am: 02.02.15

Navigation via iBeacons

Jörn Bellersen, B. Sc.
Carl von Ossietzky Universität Oldenburg
Fakultät II Department für Informatik
Seminar: Raspberry Pi: Messen, steuern, regeln - WS2014/15
E-Mail: joern.bellersen@uni-oldenburg.de

***Zusammenfassung*—In dieser Seminarausarbeitung wird die Entwicklung und Evaluation eines Navigationsmoduls für mobile Roboter beschrieben. Das Navigationsmodul wird für den Außeneinsatz ausgelegt und soll durch die Verwendung von standardisierten Schnittstellen universell einsetzbar sein. Als Grundlage dieser Seminarausarbeitung dient ein mobiler Roboter namens „Pathfinder", welcher an der Carl von Ossietzky Universität Oldenburg im WS2013/14 im Rahmen des Seminars „Raspberry Pi - inspected" von dem Autor entwickelt wurde. Der Roboter dient dazu Algorithmen zu entwickeln und umzusetzen, mit denen Rasenmähroboter "intelligenter werden" und einen optimalen Pfad zum Abmähen einer Rasenfläche finden. Grundlage für einen "intelligenten" mobilen Roboter ist die Bestimmung des aktuellen Ortes, an dem er sich befindet, um seine Abläufe planen zu können. Das in diesem Dokument beschriebene Verfahren zur Positionsbestimmung baut auf dem von Apple eingeführten iBeacon Standard auf. iBeacons sind eine Art „funkende Leuchttürme", die Funknachrichten an Stelle von Lichtzeichen aussenden. Wie die Seefahrer die Lichtzeichen zur Navigation verwenden, können mobile Roboter die Funksignale empfangen und die Distanz zur Quelle berechnen. Werden Signale von mindestens drei iBeacons empfangen, kann die Position des Roboters anhand der Distanzen ermittelt werden. Dieses Verfahren wird durch den Fachbegriff „Trilateration" beschrieben.**

Keywords*—*Raspberry Pi, autonomous mobile system, navigation, iBeacon, Bluetooth Low Energy, trilateration

I. Einführung

In der Einführung wird das Problem der Positionsbestimmung mobiler Roboter aufgezeigt und anschließend die Aufgabenstellung spezifiziert. Abgeschlossen wird das Kapitel mit einer Übersicht über die weitere Gliederung.

A. Motivation

Eine exakte Positionsbestimmung ist für mobile Roboter unerlässlich, wenn diese definierte Aufgaben erledigen und dabei bestimmte Positionen hinreichend genau anfahren sollen. Der Begriff der hinreichenden Genauigkeit wird im Zusammenhang dieser Seminarausarbeitung mit einer Positionierungsgenauigkeit im Sub-Meter-Bereich beschrieben. Dies gilt besonders dann, wenn Aufgaben nach den Gesichtspunkten der Minimierung der Bearbeitungszeit bzw. der zurückgelegten Strecke optimiert abgearbeitet werden sollen. Diese Gesichtspunkte gelten unter anderem für Rasenmähroboter, die nicht „kreuz und quer" über den Rasen fahren und einige Bereiche mehrfach bearbeiten sollen, während andere Bereiche kein einziges Mal bearbeitet werden. Ein „selbstbewusster" Roboter hingegen ist in der Lage, seine Position innerhalb eines Areals mit einer bekannten Genauigkeit zu bestimmen und anhand dieses Wissens seine Folgeschritte zu planen. Dies bezieht sich auf die Wegplanung unter gewissen Gesichtspunkten, wie beispielsweise Energieverbrauch oder Entfernung, um sich von einem Punkt A zu einem weiteren Punkt B zu bewegen. So ist der Roboter in der Lage, diesen optimalen Pfad selbstständig zu berechnen und anschließend abzufahren. Für das Beispiel eines autonomen Rasenmähroboters bedeutet dies, dass der Roboter, abhängig von der jeweiligen Rasenfläche, gezielt parallele Bahnen abfahren kann und die Anzahl mehrfach befahrener Gebiete, sowie die für das komplette Abmähen benötigte Fahrtstrecke, minimieren kann.

B. Aufgabenbeschreibung

Gegenstand dieser Seminarausarbeitung ist die Entwicklung eines Moduls mit dessen Hilfe mobile Roboter die eigene Position innerhalb einer definierten Fläche im Sub-Meter-Bereich bestimmen können. Diese Positionsdaten können vom Roboter anschließend genutzt werden, um eine autonome Wegplanung und Navigation durchführen zu können. Wichtig bei der Entwicklung ist außerdem eine einfache Portierung des Moduls auf ähnliche autonom fahrende Plattformen. Dadurch eignet sich das Modul nicht nur für den Betrieb mit der Entwicklungsplattform, sondern auch für Roboter, die über die entsprechenden Architekturen und Schnittstellen der Entwicklungsplattform verfügen. Die wichtigsten Voraussetzungen dafür sind das Vorhandensein mindestens einer USB-Schnittstelle, sowie die Verwendung eines Linux-Betriebssystems, auf die die weitere Software aufbaut.
Das Navigationsproblem von mobilen Robotern hat einen großen Stellenwert, da die akkurate Positions- und Lagebestimmung mobiler Roboter schwierig und nicht trivial ist. Im Außenbereich können sich Roboter nicht an Fixpunkten, wie beispielsweise an Wänden oder Gegenständen orientieren, wie es für Roboter im Innenbereich möglich ist. Die Positionsbestimmung muss daher unabhängig von Objekten in der Umgebung erfolgen, in der sich der Roboter bewegt. Das bedeutet, dass künstliche ortsfeste Orientierungspunkte geschaffen werden müssen, mit denen eine Berechnung der Position möglich ist. Um dieses Prinzip umzusetzen, werden Funkmodule verwendet, die Datensätze mit einer bekannten Sendeleistung ausstrahlen. Anhand

der empfangenen Signalstärke kann eine Abschätzung der Entfernung getroffen werden. Das System baut auf einem Raspberry Pi und Navigationselementen zur Positionsbestimmung auf, die zusammen mit einer Spannungsversorgung auf einem fahrbaren Untersatz montiert sind. Das System soll dabei so kostengünstig und simpel gehalten werden wie möglich.

C. Weiterer Aufbau

Der weitere Aufbau der Ausarbeitung gliedert sich wie folgt: Es wird im Kapitel II ein Überblick über das Einsatzgebiet des Navigationsmoduls gegeben. Im Grundlagenkapitel III wird der zum Verständnis des vorgestellten Entwurfs dienende theoretische Hintergrund vorgestellt. Des Weiteren wird in Kapitel IV auf die an das Navigationsmodul gestellten Anforderungen eingegangen und in Kapitel V der aktuelle Stand der Technik erläutert. In Kapitel VI werden die Vorgehensweisen beim Entwurf der Architektur und der Algorithmen des Systems beschrieben. In dem Ergebniskapitel VII wird das konstruierte System gegen die Anforderungen geprüft. Abgeschlossen wird diese Arbeit mit einem Ausblick in Kapitel VIII auf mögliche Erweiterungen und Verbesserungen des Systems.

II. Szenario

Die Grundproblemstellung dieser Seminarausarbeitung ist in Abbildung 1 skizziert. Dabei existiert eine zu bearbeitende Rasenfläche und mögliche weitere Flächen, die an die Rasenfläche angrenzen oder diese Fläche unterbrechen. Diese Nicht-Rasenflächen, wie beispielsweise Terrassen oder Beete, dürfen von dem Roboter nicht befahren werden. Das bedeutet für den Roboter, dass zu jedem Zeitpunkt bekannt sein muss, an welcher Position er sich innerhalb der Rasenfläche befindet.

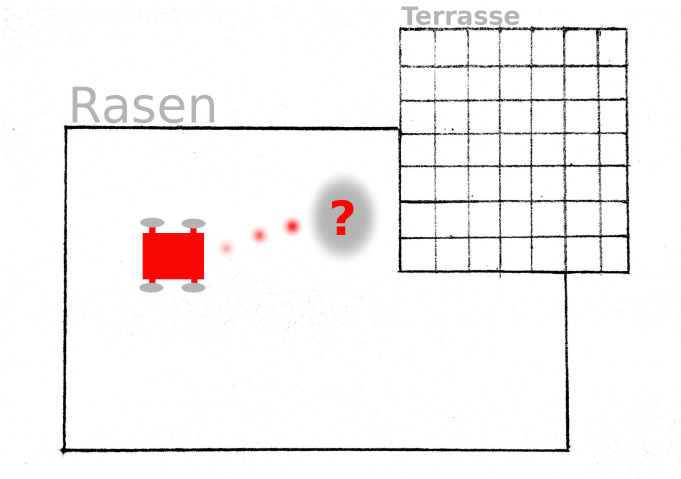


Abbildung 1. Beispiel eines Einsatzszenarios

III. Theoretische Grundlagen

In den folgenden Abschnitten werden die theoretischen Grundlagen des Navigationsmoduls vorgestellt und erläutert.

A. Bluetooth Low Energy

Bluetooth Low Energy wird durch BLE abgekürzt. Der BLE-Standard wurde als Erweiterung des immer noch aktuellen Bluetooth 3.0-Standards entwickelt. BLE wird auch als Bluetooth 4.0-Standard bezeichnet. Im Gegensatz zu Bluetooth 3.0 bietet der BLE-Standard einen extrem reduzierten Stromverbrauch. Dies schlägt sich allerdings in der Übertragungsgeschwindigkeit nieder: während Bluetooth 3.0 Daten mit mehreren $\frac{Mbit}{s}$ überträgt, liegt die Übertragungsgeschwindigkeit bei dem BLE-Standard bei lediglich mehreren hundert $\frac{kbit}{s}$ [1].
Punkt-zu-Punkt-Verbindungen sollen laut [1] nur bis zu 10 m möglich sein, danach ist eine stabile Verbindung nicht mehr gewährleistet.

B. iBeacon

Die Technologie der iBeacons wurde von Apple im Jahre 2013 eingeführt und verwendet den BLE-Standard zum Übertragen der Nachrichten [2]. Eine grobe Funktionsbeschreibung findet sich in dem Begriff „Beacon“ wieder, welches aus dem Englischen übersetzt „Leuchtfeuer“ bedeutet. Neben dem Sendemodul verfügt ein iBeacon auch über eine eingebaute Knopfzellen-Batterie. Dadurch können die Geräte, durch die Verwendung des BLE-Standards, häufig jahrelang aktiv bleiben, ohne dass die integrierte Batterie ausgewechselt werden muss [8].
Die iBeacons senden unter einer individuellen und eindeutigen Kennung Daten. Die eindeutige Kennung besteht aus drei Feldern: der UUID, sowie dem Major- und Minor-Feld. UUID ist die Abkürzung für „Universal Unique Identifier“ und repräsentiert eine Herstellernummer. Mit dem Major-Feld kann eine Art Gruppe von iBeacons definiert werden, auf die mit einem entsprechenden Filter reagiert werden kann. Das Minor-Feld beschreibt die ID eines iBeacon innerhalb einer Gruppe. In Abbildung 2 sind aufgenommene Beispieldatensätze mehrerer iBeacons dargestellt.

```
UUID: 20CAE8A0-A9CF-11E3-A5E2-0800200C9A66 MAJOR: 125 MINOR: 19021 POWER: -78 RSSI: -77
UUID: 20CAE8A0-A9CF-11E3-A5E2-0800200C9A66 MAJOR: 125 MINOR: 16352 POWER: -78 RSSI: -68
UUID: 20CAE8A0-A9CF-11E3-A5E2-0800200C9A66 MAJOR: 7   MINOR: 45106 POWER: -78 RSSI: -59
UUID: 20CAE8A0-A9CF-11E3-A5E2-0800200C9A66 MAJOR: 125 MINOR: 17326 POWER: -78 RSSI: -72
```

Abbildung 2. Darstellung von empfangenen BLE-Datensätzen

Eine Navigation wird durch das Auswerten der Signalstärke, mit der die Daten des jeweiligen iBeacons empfangen wurden, erreicht.

C. Signalstärke

Die Signalstärke, mit der Radiowellen ausgesendet oder empfangen werden, kann als Leistungspegel verstanden werden. Ein Pegel wird durch die logarithmische Einheit Dezibel [dB] angegeben. Dabei liegt der Einheit der dekadische Logarithmus $\log_{10}(x)$ zu Grunde. Dezibel gibt die Intensität einer bestimmten Größe in relativer Beziehung zu einer Referenzgröße an.

$$P_{Power} - P_{RSSI} = 10 \cdot \log_{10}(r^2) \Leftrightarrow r = \sqrt{10^{\frac{P_{Power} - P_{RSSI}}{10}}} \quad (0)$$

Formel 0 beschreibt den Zusammenhang zwischen der Dämpfung eines Signals bei quadratischer Zunahme des Abstandes **r**. Der empfangene Leistungspegel P_{RSSI} wird auf den Leistungspegel P_{Power} als Referenz bezogen, mit dem das Signal am Sender ausgestrahlt wurde. RSSI ist die Abkürzung für Radio Signal Strength Indicator und gibt die Signalstärke an, mit der ein Datenpaket oder ein einzelnes Bit an der Antenne detektiert wird [3].
Kalibriert wurde der P_{Power}-Wert bei einem Abstand von 1 Meter. Dabei sind drei Fälle für die Berechnung des Abstandes zwischen Sender und Empfänger zu unterscheiden:

1) $P_{Power} - P_{RSSI} > 0 \Rightarrow$ Abstand größer 1 m
2) $P_{Power} - P_{RSSI} = 0 \Rightarrow$ Abstand gleich 1 m
3) $P_{Power} - P_{RSSI} < 0 \Rightarrow$ Abstand kleiner 1 m

D. Trilateration

Das Prinzip der Lateration baut auf dem Messen von Distanzen zwischen einem unbekannten Punkt und einem oder mehreren Referenzpunkten auf. Die Voraussetzung für dieses Verfahren ist jedoch, dass die Position der Referenzpunkte bekannt ist. Die gemessene Distanz kann als Radius verstanden werden, welcher einen Kreis um die Position des Referenzpunktes im Koordinatensystem zieht. Der Empfänger kann sich dabei auf einem beliebigen Punkt auf dieser Kreisbahn befinden. Anhand von mehreren gemessenen Distanzen und die daraus ermittelten Kreisbahnen können Schnittpunkte berechnet werden, um die wahre Position einzugrenzen.
Bei der Positionsbestimmung ist es wichtig, mindestens drei Distanzen in die Berechnung einzubeziehen, damit das Ergebnis eineindeutig ist. Durch das Einbeziehen von drei Distanzen in die Berechnung wird dieses Verfahren auch als **Tri**lateration bezeichnet. Bei der Verwendung von nur zwei iBeacons gibt es doppeldeutige Ergebnisse, wovon ein Ergebnis durch den dritten Kreis, der einen der beiden Punkte schneidet, verworfen werden kann. Dadurch ist es möglich, die wahre Position des Empfängers zu bestimmen. Das Einbeziehen von mehreren Referenzpunkten erhöht dabei die Genauigkeit, mit der die Position bestimmt werden kann [4].

IV. Anforderungsspezifikation

Im Folgenden sind die Anforderungen an das Navigationsmodul in einer numerischen Auflistung dargestellt.

1) Die Kosten des Systems sollen sich auf maximal 200 € belaufen.
2) Der Konfigurations- und Installationsaufwand des System soll minimal gehalten werden.
3) Anpassungsfähigkeit und Kompatibilität an viele Linux basierte Robotersysteme.
4) Das Navigationsmodul soll während des Bootvorgangs des Betriebssystems automatisch gestartet werden.
5) Das Navigationsmodul soll aus so wenig wie möglich Komponenten bestehen.
6) Das Navigationsmodul soll so simpel wie möglich gehalten werden.
7) Das Modul besteht aus einem Empfänger, welcher Signale mehrerer Sender verarbeitet.
8) Die minimale Anzahl von Sendern liegt bei drei.
9) Die maximale Anzahl von Sendern liegt bei 20.
10) Die Genauigkeit der Positionsbestimmung soll im Sub-Meter-Bereich erfolgen.

Nachfolgend werden Methoden und Verfahren kommerziell eingesetzter Produkte erläutert und beschrieben.

V. Stand der Technik

Kommerziell vertriebene Rasenmähroboter sind in der Regel nicht besonders intelligent, da diese über keine akkuraten Positionsbestimmungssysteme verfügen und daraus bedingt auch keine Wegplanung durchführen können. Viele Hersteller setzen bei der Abgrenzung der Rasenflächen ihrer Produkte auf Begrenzungsdrähte [5]. Diese Begrenzungsdrähte spannen die Arbeitsbereiche auf, in denen sich der Roboter bewegt. Zusätzlich dienen sie dazu, den Weg zurück zur Ladestation zu weisen, wenn der Roboter daran entlang fährt. Erreicht ein Roboter solch einen Draht, welcher durch induktive Sensoren erfasst wird, „macht der Roboter kehrt“ . Anschließend fährt dieser in einen zufällig gewählten Winkel in eine andere Richtung weiter, bis er erneut auf den Begrenzungsdraht trifft und die Prozedur von neuem beginnt. Dieser Ansatz widerspricht den bereits erwähnten Optimierungskriterien der Minimierung der Bearbeitungszeit bzw. der zurückgelegten Wegstrecke und der mehrmals befahrenen Abschnitte.

A. Navigationssysteme

In diesem Abschnitt wird eine Übersicht über Arten der absoluten und relativen Positionsbestimmung gegeben, die bei anderen mobilen Plattformen eingesetzt werden. Außerdem wird die Roboterplattform vorgestellt, die als Basis für die Entwicklung des Navigationsmoduls dient.

1) GPS: In der maritimen Technik wird häufig ein General Positioning System (GPS) verwendet, um die Position von Schiffen auf Gewässern oder Ozeanen zu bestimmen.
Das Funktionsprinzip eines gewöhnlichen GPS-Systems besteht darin, die Signale mehrerer GPS-Satelliten (mindestens drei für die reine Positionsbestimmung) auszuwerten. GPS-Satelliten übermitteln stetig die aktuelle Position, sowie einen hochakkuraten Zeitstempel und befinden sich in einem Orbit von ungefähr $20 \cdot 10^6$ m über der Erdoberfläche. Elektromagnetische Wellen bewegen sich im Vakuum mit rund 299792458 $\frac{m}{s}$. Daraus folgt, dass Signale ungefähr $\frac{20 \cdot 10^6 m}{299792458 \frac{m}{s}} \approx 66.71 ms$ nach dem Aussenden vom GPS-Empfänger detektiert werden.
Aus den Abweichungen in den Signallaufzeiten, sowie der Phasenverschiebung der einzelnen Satelliten und der jeweils vom Satelliten übertragenen Position auf dessen Orbit, lässt sich die Position des Empfängers auf der Erdoberfläche bestimmen, sowie die Höhe darüber [6]. GPS-Empfänger ohne Mechanismen, die eine Steigerung der Genauigkeit hervorrufen, weisen typischerweise Genauigkeiten im Bereich von ±(10 bis 15) m auf [7]. Eine Möglichkeit

der Genauigkeitssteigerung wird durch die Verwendung eines differentiellen GPS-Empfängers gegeben.
Bei einem differentiellen GPS (DGPS) wird das System um einen ortsfesten GPS-Empfänger erweitert, dessen genaue Position bekannt ist und als Referenzpunkt bezeichnet wird. Ein solcher Referenzpunkt sendet ein Funksignal aus, welches die Differenz aus theoretischer und realer Signallaufzeit beinhaltet. Differentielle GPS-Empfänger können dieses Signal empfangen, dekodieren und auswerten. Durch die Einbeziehung der Daten aus dem Korrektursignal ist das mobile GPS-System in der Lage, seinen Fehler beim Empfang der GPS-Signale der Satelliten zu bestimmen und den relativen Positionsfehler zu verringern [6]. Dadurch wird die Genauigkeit gesteigert und liegt laut Aussage von [7] im Bereich von ±(3 bis 5) m für DGPS-Systeme. Diese Genauigkeit ist allerdings nicht ausreichend, um eine hinreichend genaue Positionsbestimmung im Sub-Meter-Bereich durchzuführen.

2) Odometrie: Unter Odometrie wird die Messung der zurückgelegten Wegstrecke eines mobilen Systems mit Hilfe des Drehwinkels der Antriebsräder beschrieben. Das System kann sich dabei entweder durch Räder oder einen Kettenantrieb fortbewegen. Das Messverfahren beschreibt die Position eines mobilen Systems, allerdings nicht absolut zu einem Referenzpunkt, sondern relativ zu einem Startpunkt der Messung. Daher zählt die Odometrie zu einem relativen Messverfahren, welches durch das Gesetz der Fehlerfortpflanzung bei steigender zurückgelegter Strecke, eine wachsende relative Messunsicherheit mit sich bringt [10].

Aufgrund der maximalen Genauigkeit von ±3 m für die Positionsbestimmung eines DGPS und dem proportional zur zurückgelegten Wegstrecke wachsenden Fehler der Odometrie, eignen sich diese beiden vorgestellten Navigationsverfahren für die Umsetzung des Navigationsmoduls nicht.

B. Roboterplattform „Pathfinder“

Diese Seminarausarbeitung baut auf einem mobilen Roboter namens „Pathfinder“ auf, welcher an der Carl von Ossietzky Universität Oldenburg im WS2013/14 im Rahmen des Seminars „Raspberry Pi - inspected“ von dem Autor entwickelt wurde. Der Roboter mit den Maßen 18 cm x 17 cm x 12.5 cm (L x B x H) ist in Abbildung 3 zu sehen.
Der Pathfinder-Roboter dient als Plattform, um neue intelligente Algorithmen im Bezug auf Positionsbestimmung- und Wegplanungsaufgaben zu entwickeln und umzusetzen. Diese Algorithmen sollen bei zukünftig entwickelten mobilen Robotern eingesetzt werden. Ein Beispiel dafür, um einen optimalen Pfad zum Abmähen einer Rasenfläche oder zum Säen eines Ackers zu finden.
Unter der in Abbildung 3 dargestellten vorderen Abdeckung befindet sich die USB-Schnittstelle um WLAN- oder Bluetooth-Funkadapter mit dem System zu verbinden.

Abbildung 3. Darstellung der Pathfinder-Roboterplattform

VI. Entwurf

Der Entwurf des Navigationsmoduls geht auf die verwendeten Komponenten und Lösungsansätze ein. Es wird die Software- und Hardwarearchitektur beschrieben, sowie die Art und Weise, wie diese implementiert wurden.

A. Material & Methoden

1) iBeacons von Onyx Beacon: Die iBeacons gibt es mittlerweile von mehreren Herstellern aus aller Welt. Die einzelnen iBeacons unterscheiden sich prinzipiell nur in Größe und Aussehen voneinander. Die Funktion ist bei allen identisch: Es werden in vordefinierten Intervallen Datensätze mit der eindeutigen Kennung gesendet, die unter anderem auch Nutzdaten enthalten können. Die Intervalle, in denen Daten gesendet werden, lassen sich konfigurieren. Der Zyklus der Intervalle kann von wenigen Sekunden bis zu mehreren Stunden eingestellt werden [8]. In der folgenden Grafik sind zwei der verwendeten iBeacons der Firma Onyx Beacon dargestellt.

Abbildung 4. Verwendete iBeacons des Herstellers Onyx Beacon

2) BLE-Modul: Als Bluetooth Low Energy Modul wurde ein einfacher Adapter verwendet, welcher sich mit einem der USB-Ports des Raspberry Pi verbinden und durch die Linux-Bibliothek namens BlueZ ansprechen lässt [9]. Bei dem Adapter handelt es sich um ein Gerät der Firma Conrad Elecktronik, welches den Namen „Bluetooth USB-Stick 4.0 + EDR“ trägt.
Dieser BLE-Adapter ist in der Abbildung 5 dargestellt.

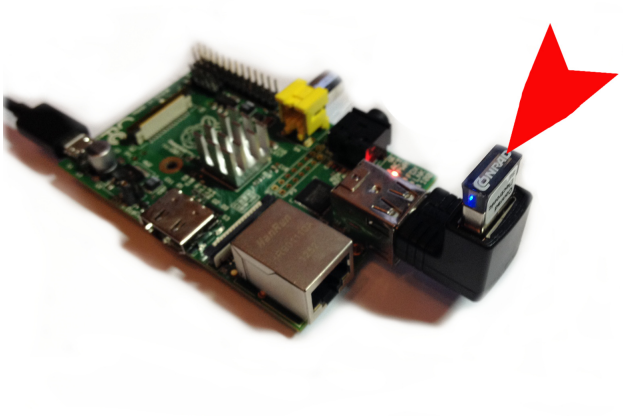


Abbildung 5. Raspberry Pi mit eingestecktem USB-BLE-Modul

In der Abbildung ist der BLE Adapter über einen USB-Winkeladapter mit dem Raspberry Pi verbunden, wie es auch bei dem Pathfinder-Roboter der Fall ist. Durch den Adapter soll ein besserer Empfang gewährleistet werden, da sich die Antenne dadurch außerhalb des Gehäuses befindet.

3) Programmierung: Die Wahl der Programmiersprache ist entscheidend, um die in Kapitel IV spezifizierten Anforderungen bestmöglich umsetzen und erfüllen zu können.
Zur softwareseitigen Umsetzung des Navigationsmoduls wurde ein hybrider Ansatz aus der Programmiersprache *C* und der Skriptsprache BASH (Bourne Again Shell) gewählt. Die Umsetzung des Ansprechens und anschließenden Auslesens der Rohdaten aus dem Bluetooth-Treiber gestaltet sich in *C* relativ aufwendig. In der BASH-Umgebung hingegen kann dieser Vorgang durch einen einzigen Betriebssystemaufruf realisiert werden. Die Performanz von BASH im Hinblick auf mathematische Berechnungen ist im Vergleich zu *C* deutlich schlechter. BASH wurde bei der Entwicklung nicht für komplexe mathematische Berechnungsanweisungen auf Benutzerebene ausgelegt [11]. Daher wurden bei diesem hybriden Ansatz die Stärken der beiden Programmiersprachen genutzt: In BASH wird der gesamte Kontroll- und Datenfluss des Navigationsmoduls realisiert. Die mathematischen Berechnungen werden in ein *C*-Programm ausgelagert. In dem *C*-Programm wird die Bibliothek `math.h` inkludiert, um die benötigten mathematischen Funktionen bereit zu stellen. An den entsprechenden Stellen im BASH-Programmablauf wird dann das *C*-Programm mit den benötigten Übergabeparametern aufgerufen und die Rückgabewerte abgefangen und weiter verarbeitet.

4) Positionsberechnung: Anhand der am Empfänger bestimmten Signalstärke einer Nachricht und der kalibrierten Sendeleistung des jeweiligen iBeacons, kann wie in Kapitel III-C beschrieben, der Abstand zu dem iBeacon berechnet werden. Empfängt der Roboter lediglich das Signal von einem einzigen iBeacon, dann kann lediglich die Position einer Kreisbahn um das iBeacon mit dem berechneten Radius bestimmt werden. Werden die Signale von zwei iBeacons empfangen, so bilden die zwei Schnittpunkte der Kreisbahnen, mit den berechneten Radien, den möglichen Aufenthaltsort des Roboters. In Abbildung 6 sind diese möglichen Aufenthaltsorte durch die Schnittpunkte s1 und s2 gekennzeichnet. Der Anwendungsfall beim Empfang von nur zwei iBeacons ist in Abbildung 6 aufgezeigt.

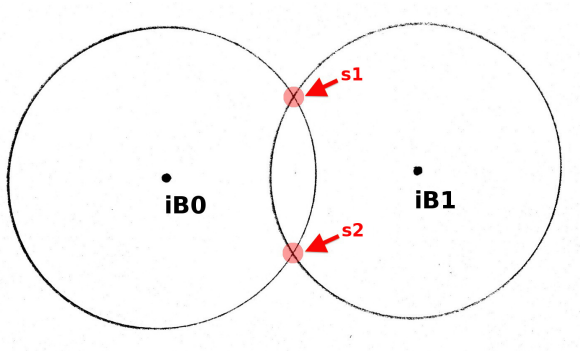


Abbildung 6. Empfang von Signalen zweier iBeacons
Bestimmung der möglichen Aufenthaltsorte s1 und s2 des Empfängers

Werden hingegen Signale von mindestens drei iBeacons simultan empfangen, dann kann die Position des Roboters exakt bestimmt werden. In Abbildung 7 wird dieses Szenario dargestellt.

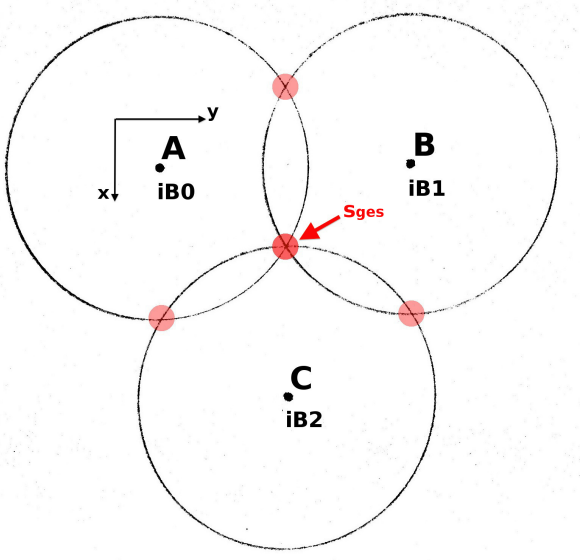


Abbildung 7. Empfang von Datensätzen von drei iBeacons

Schneiden sich die Kreisbahnen der berechneten Radien in einem Punkt s_{ges} (Abbildung 7), so ist dies mit großer Wahrscheinlichkeit der Aufenthaltsort des Roboters.

5) Platzierungsschemata der iBeacons: Es gibt verschiedene Möglichkeiten, die iBeacons auf einer Rasenfläche zu platzieren. Die zwei Varianten, die in dieser Seminarausarbeitung näher betrachtet werden, sind zum einen das Vergraben der iBeacons unter der Rasennabe (Abbildung 8) und zum anderen das Anbringen der iBeacons in einer gewissen Höhe über den Ecken und Kanten der Rasenfläche (Abbildung 9). Die X- und Y-Positionen der iBeacons müssen bekannt sein und dürfen sich nicht verändern, da spätere Berechnungen diese Beziehungen der Abstände der iBeacons untereinander verwenden, um die Position des Roboters zu bestimmen.

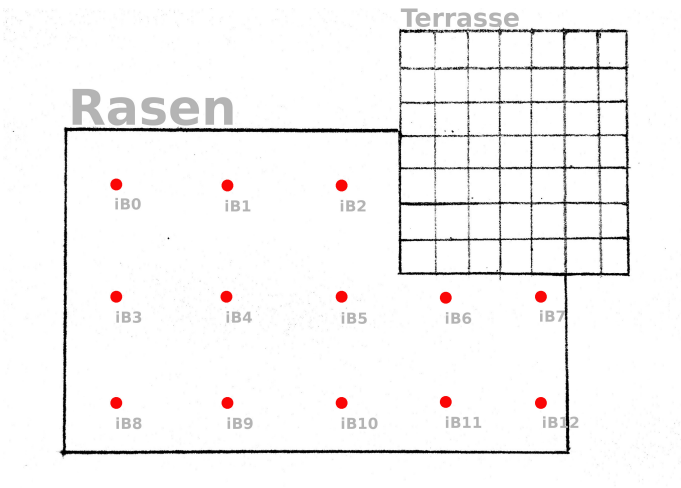


Abbildung 8. Anordnung iBeacons in der Rasenfläche

Abbildung 8 zeigt eine Anordnung der iBeacons in einer Art Gitterstruktur. Dabei sind die iBeacons wasserdicht abzudichten, sodass kein Wasser aus dem Erdreich in das Gehäuse eindringen kann. Gegebenenfalls sind Markierungen im Rasen einzurichten, um einzelnen iBeacons bei Wartungsarbeiten schnell ausfindig zu machen.

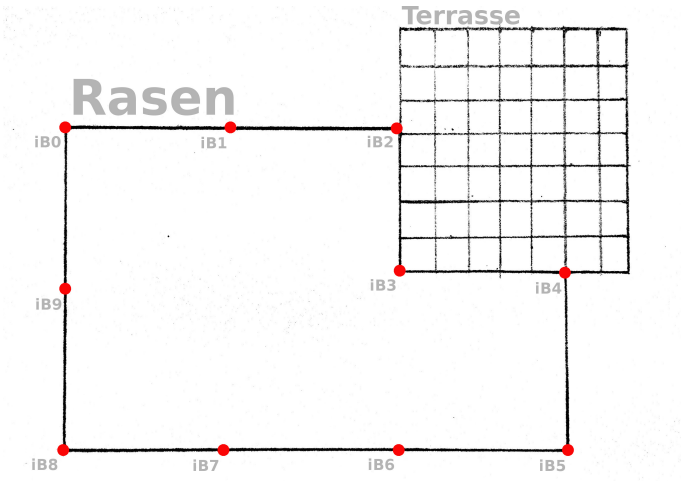


Abbildung 9. Anordnung iBeacons Seitenkanten und Eckpunkten

Abbildung 9 zeigt die Installation der iBeacons überhalb der Rasenoberfläche, beispielsweise an Stäben, die auf den Kanten der Fläche in den Rasen gebohrt werden.

B. Architektur

Im Kapitel der Architektur wird auf die Strukturen und den Aufbau der Hard- und Software näher eingegangen. Zusätzlich wird der Auflauf der Software beschrieben.

1) Hardware: Der verwendete Hardwareaufbau ist der Abbildung 5 zu entnehmen. In folgendem Diagramm sind die verwendeten Komponenten, sowie die Verbindungen untereinander dargestellt.

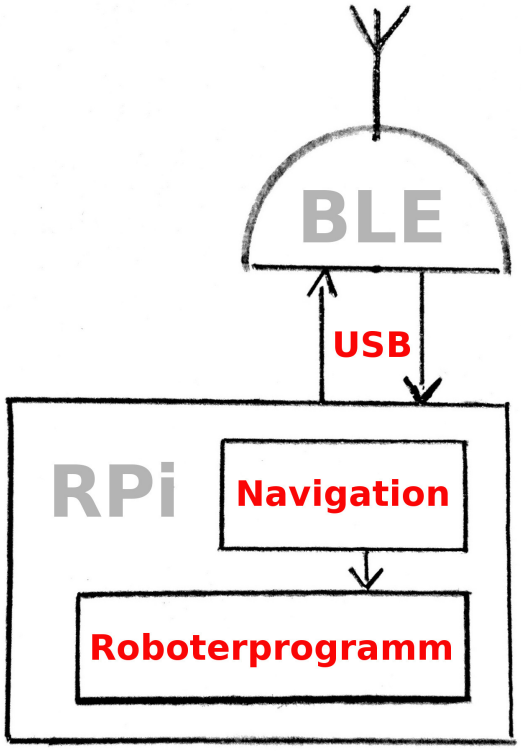


Abbildung 10. Hardwarearchitektur des Robotersystems

Die Erweiterung des bestehenden Systems um das Navigationsmodul umfasst nur wenige Komponenten, wie in Abbildung 10 abgebildet. Als Steuereinheit des Roboters und des Navigationsmoduls wird ein Raspberry Pi Modell B der 2. Revision verwendet.
Über USB ist derzeit noch ein WLAN-Adapter mit dem Raspberry Pi verbunden, um eine drahtlose Kommunikation mit einem Computer herstellen zu können. Dieser WLAN-Adapter wird durch den Bluetooth Adapter ersetzt, da der Raspberry Pi nur eine begrenzte Anzahl von USB-Schnittstellen zur Verfügung hat. Durch den Bluetooth Adapter können die Signale der iBeacons empfangen und zusätzlich noch eine Verbindung zu einem Computer aufgebaut werden, um den Roboter zu steuern oder zu überwachen.

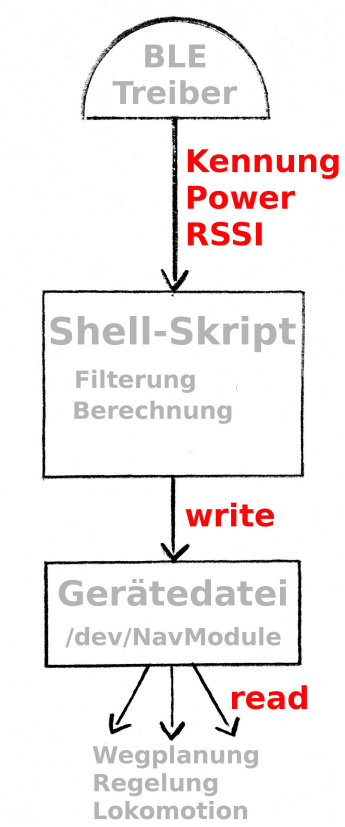


Abbildung 11. Softwarearchitektur des Navigationsmoduls

2) Software: Die grobe Darstellung der Softwarearchitektur des Navigationsmoduls ist in der Abbildung 11 aufgezeigt. Zusätzlich ist in der Grafik der Datenfluss dargestellt, der sich innerhalb des Navigationsmoduls bewegt. Aus der Abbildung geht zusätzlich hervor, dass die Software des Navigationsmoduls nicht ein einziges Programm beinhaltet, sondern aus mehreren Komponenten besteht. Das Shell-Skript steuert den Kontrollfluss im Navigationsmodul. Dazu führt es Systemaufrufe im Linux-Kernel aus, die die vom Bluetooth-Treiber empfangenen Rohdaten abholen und zwischenspeichern. In einem Folgeschritt werden die Rohdaten mit Hilfe von Systemaufrufen des Tools „sed“ gefiltert und die Datenfelder `UUID`, `MAJOR`, `MINOR`, `POWER` und `RSSI` extrahiert.
Die Berechnung der Distanz d zwischen Empfänger und iBeacon, sowie die Berechnung der aktuellen Position $(x\,,y)$ wird in einem C-Programm ausgelagert, welches innerhalb der Shell-Skriptes aufgerufen wird. Dieser Zwischenschritt wurde eingeführt, da die Programmiersprache BASH nicht darauf ausgelegt wurde, mathematische Berechnungen auf Anwendungsebene durchzuführen. Die Auslagerung von Berechnungen in einem C-Programm hat auch Vorteile im Bezug auf Performanz gegenüber der direkten Implementierung in BASH-Code.

C. Implementierung

Im Implementierungskapitel wird die Umsetzung der Hard- und Softwarearchitektur beschrieben, sowie auf das Zusammenspiel von Hard- und Software eingegangen.

1) Hardwareaufbau: Bei der Platzierung der iBeacons gibt es mehrere Muster, diese zu platzieren. Die zwei bereits in Kapitel VI-A5 vorgestellten Muster, werden beide unter realen Bedingungen im Garten getestet, um die bessere Methode zu evaluieren. Dabei werden mehrere Faktoren berücksichtigt. Es wird die Abdeckung der Rasenfläche durch die Sendeleitung der iBeacons bewertet. Außerdem wird auf die notwendige Präparation geachtet, die notwendig ist, um die iBeacons unter der Grasnabe oder entlang den Kanten der Rasenfläche zu platzieren.

2) Algorithmus: Zur Berechnung der aktuellen Position wurde ein Algorithmus entwickelt, welcher sich aus den Überlegungen und Gedankengängen aus dem Kapitel VI-A4 ergeben hat.
Die Herleitung ist im folgenden Dargestellt:

$$A: \; d_a^2 = (x-x_a)^2 + (y-y_a)^2 + \cancel{(z-z_a)^2} \tag{1}$$
$$= \; x^2 - 2\cdot x_a \cdot x + x_a^2 \; + \; y^2 - 2\cdot y_a \cdot y + y_a^2$$
$$B: \; d_b^2 = (x-x_b)^2 + (y-y_b)^2 + \cancel{(z-z_b)^2} \tag{2}$$
$$= \; x^2 - 2\cdot x_b \cdot x + x_b^2 \; + \; y^2 - 2\cdot y_b \cdot y + y_b^2$$
$$C: \; d_c^2 = (x-x_c)^2 + (y-y_c)^2 + \cancel{(z-z_c)^2} \tag{3}$$
$$= \; x^2 - 2\cdot x_c \cdot x + x_c^2 \; + \; y^2 - 2\cdot y_c \cdot y + y_c^2$$

Die drei Punkte A, B und C beschreiben die Position der iBeacons, die in die Berechnung einbezogen werden. Mathematisch werden die iBeacons als Strahler modelliert, die Kugelgleichungen beschreiben, bei denen die Z-Komponente nicht betrachtet wird. Somit wird aus einer Kugel im dreidimensionalen Raum eine Projektion eines Kreises in die X-Y-Ebene, wie in Abbildung 7 dargestellt. Um die Schnittpunkte der Kreisgleichungen zu erhalten, werden aus diesen Gleichungen durch das Subtraktionsverfahren zwei neue Gleichungen erzeugt. Gleichung $I = A$ - B und $J = A$ - C.

$$I: d_a^2 - d_b^2 \; = \; -2x(x_a - x_b) - 2y(y_a - y_b) + x_a^2 - x_b^2 - y_a^2 - y_b^2 \tag{4}$$
$$J: d_a^2 - d_c^2 \; = \; -2x(x_a - x_c) - 2y(y_a - y_c) + x_a^2 - x_c^2 - y_a^2 - y_c^2 \tag{5}$$

Wird anschließend eine Trennung der Variablen bei den neuen Gleichungen durchgeführt, werden die Gleichungen I und J wie folgt umgeformt:

$$I: x(x_a - x_b) + y(y_a - y_b) \; = \; \frac{x_a^2 - x_b^2 + y_a^2 - y_b^2 - d_a^2 + d_b^2}{2} \tag{6}$$

Der rechte Term der Gleichung 6 ist konstant und wird durch die Hilfsvariable i substituiert.

$$\Rightarrow x(x_a - x_b) + y(y_a - y_b) \; = \; i \tag{7}$$

Durch Umstellung der Gleichung 7 erhalten wir y:

$$\Rightarrow y \; = \; \frac{i - x(x_a - x_b)}{y_a - y_b} \tag{8}$$

$$J :\Rightarrow x(x_a - x_c) + y(y_a - y_c) = \frac{x_a^2 - x_c^2 + y_a^2 - y_c^2 - d_a^2 + d_c^2}{2} \quad (9)$$

$$\Rightarrow x(x_a - x_c) + y(y_a - y_c) = j \quad (10)$$

Der Rechte Teil der Gleichung 9 ist ebenfalls konstant und wird durch die Hilfsvariable j substituiert. Anschließend wird y (Gleichung 8) in J (Gleichung 10) eingesetzt.

$$y\,in\,J : x(x_a - x_c) + \frac{i - x(x_a - x_b)}{y_a - y_b} \cdot (y_a - y_c) = j \quad (11)$$

$$\Rightarrow x \cdot \frac{x_a - x_c}{y_a - y_c} + \frac{i + x(x_a - x_b)}{y_a - y_b} = \frac{j}{y_a - y_c} \quad \Rightarrow x = \frac{\frac{j}{y_a - y_c} - \frac{i}{y_a - y_b}}{\frac{x_a - x_c}{y_a - y_c} - \frac{x_a - x_b}{y_a - y_b}}$$

$$\Rightarrow x = \frac{j(y_a - y_b - i(y_a - y_c))}{(x_a - x_c) \cdot (y_a - y_b) - (x_a - x_b) \cdot (y_a - y_c)} \quad (12)$$

$$\wedge\ k = (x_a - x_c) \cdot (y_a - y_b) - (x_a - x_b) \cdot (y_a - y_c) \quad (13)$$

3) C-Programm: Das in Kapitel VI-B2 erwähnte *C*-Programm „iBeacon_calculate“ implementiert die in Kapitel VI-C2 hergeleiteten Gleichungen zur Berechnung der Distanz d, sowie der Abszisse x und der Ordinate y. Zur Berechnung der Distanz wird der rechte Teil aus Gleichung 0 verwendet. Diese Berechnung benötigt als Übergabewert das `Power` und `RSSI` Feld des jeweiligen Datenpaketes. Zur Berechnung der Abszisse wird Gleichung 12 verwendet. Die Variablen i und j werden durch die Gleichungen 7 bzw. 10 substituiert. Zur anschließenden Berechnung der Ordinate y wird Gleichung 8 implementiert, welche ebenfalls die durch Gleichung 7 substituierte Variable i verwendet.

4) Shell-Skript: Das Shell-Skript soll direkt während der Bootsequenz des Systems gestartet werden. Dazu wird das Skript von ein einem Service gestartet, welches unter Debian-Linux in dem Verzeichnis `/etc/init.d` abgelegt wird [12].
Wie bereits in Kapitel VI-B2 näher erläutert wurde, führt das Shell-Skript einige Systemaufrufe aus, welche die Daten von dem USB-Adapter abgreifen, interpretieren und filtern. Ein Programmablaufplan des Shell-Skripts ist in Abbildung 12 dargestellt. Aus den gefilterten Rohdaten werden die Felder `UUID`, `Major`, `Minor`, `Power` und `Rssi` extrahiert. Anschließend wird die Distanz berechnet in dem das *C*-Programm aufgerufen wird und der Rückgabewert abgespeichert wird. Sind einmalig Distanzen zu drei unterschiedlichen iBeacons bekammt, kann die Bestimmung der Position durch Berechnen der X- und Y-Position getriggert werden. Dazu wird das *C*-Programm erneut mit geänderten Übergabeparametern aufgerufen. Anhand der Anzahl der Übergabeparameter entscheidet das Programm, ob die Berechnung für x oder y ausgeführt werden soll. Für die Bestimmung der X-Koordinate werden in der Summe 9 Übergabeparameter benötigt. Dabei handelt es sich um die bekannten X- und Y-Positionen der drei iBeacons, sowie der zuvor berechneten Distanzen zu den einzelnen iBeacons. Mit der berechneten X-Koordinate kann anschließend die Y-Koordinate bestimmt werden. Dazu wird das *C*-Programm erneut mit geänderten Übergabeparametern aufgerufen. Für diese Berechnung werden 7 Übergabeparameter benötigt. Bei den Übergabeparametern handelt es sich um die X- und Y-Koordinate der iBeacons A und B, sowie den entsprechenden Distanzen (siehe Abbildung 7) und der zuvor berechneten Abszisse x. Anschließend werden die berechneten Positionen in einer Datei im Dateisystem des Linux-Betriebssystems abgelegt. Diese Datei liegt im Verzeichnis `/dev` und heißt „Navigation“. Auf diese Datei können die Programme, die den Roboter steuern, zugreifen und eine Bahnplanung und Lokalisierung durchführen.

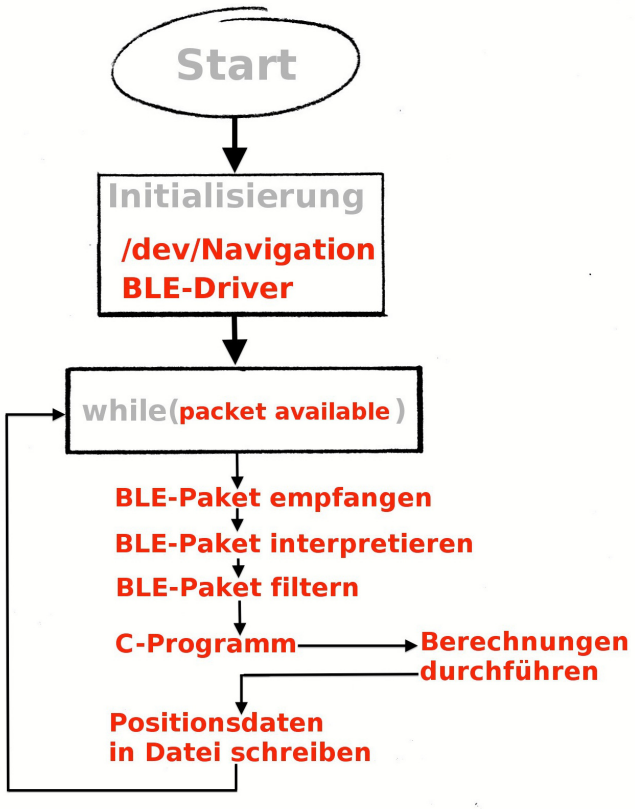


Abbildung 12. Programmablaufplan des Shell-Skripts

VII. Ergebnisse

Das gesetzte Budget von rund 200 € wurde nicht erreicht, wenn der Raspberry Pi bereits vorhanden ist. Es wurden 179 € für 10 iBeacons der Firma OnyxBeacons und ca. 10 € für den BLE-Adapter der Firma Conrad Elektronik ausgegeben. Der Raspberry Pi würde zusätzlich mit rund 30 € verbucht werden.
Durch den simplen Hardwareaufbau und den wenigen benötigten Komponenten ist der Installationsaufwand sehr gering gehalten. Zusätzlich lässt sich das Navigationsmodul durch die universelle Softwarearchitektur auf einer großen Anzahl durch Linux-Betriebsysteme gesteuerten mobilen Plattformen verwenden.
Bei den durchgeführten Tests, bezüglich der Anordnung der iBeacons, hat sich herausgestellt, dass die Positionierung der iBeacons auf den Seitenkanten oder Eckpunkten nicht praktikabel ist (siehe Abbildung 9). Wenn die zu bearbeitende Rasenfläche eine gewisse Größe erreicht, werden die Distanzen zu den iBeacons und die damit verbundenen Ungenauigkeiten der Messungen ebenfalls immer größer.

Überschreitet die Rasenfläche eine maximale Größe, entstehen in der Mitte der Fläche Bereiche, an denen keine Signale mehr von iBeacons empfangen werden können und somit keine Positionsbestimmung mehr erfolgen kann. Werden die iBeacons jedoch in einer Gitterstruktur angeordnet und unter der Grasnabe platziert (siehe Abbildung 8), entsteht dieses Problem nicht. Wenn die Größe der Rasenfläche steigt, dann kann diese Struktur durch weitere iBeacons erweitert werden, sodass die Abstände der iBeacons zueinander konstant bleibt und der Betrag der Ungenauigkeit ebenfalls konstant bleibt. Ein Nachteil dieser Methode ist jedoch, dass bei der Herleitung der Gleichung eine Vereinfachung vorgenommen wurde: Es wurde davon ausgegangen, dass sich Sender und Empfänger in derselben Ebene befinden und dadurch der Anteil der Z-Komponente gleich Null ist. In Wirklichkeit ist der Betrag der Z-Komponente ungleich Null und muss daher berücksichtigt werden. Bei den vereinfachten Gleichungen ergibt sich daher ein systematischer Fehler, der proportional zum Betrag der Z-Komponente und umgekehrtproportional zur gemessenen Distanz ist: $err_{syst} = sin(\frac{Distanz}{|Z-Komponente|})$
Bei der Betrachtung der gemessenen Distanzen ist aufgefallen, dass die Messwerte teilweise sehr große Ausreißer aufwiesen. Durch das Anwenden eines Mittelwertfilters könnten diese Ausreißer stark dezimiert werden.

Verfügt die Roboterplattform neben dem Navigationsmodul zusätzlich über einen Kompass, dann kann sich der Roboter gezielt auf einen Wegpunkt ausrichten und diesen mit Hilfe des Navigationsmoduls anfahren.

Bei der Verwendung des Bluetooth-Adapters und der iBeacons ist aufgefallen, dass sich die sendende Antenne des iBeacons und die empfangene Antenne des Bluetooth-Adapters in derselben Polarisationsebene befinden müssen, damit eine korrekte Ermittlung der empfangenen Signalstärke stattfindet [13]. Tritt ein Tilt der Antenne des empfangenen Bluetooth-Adapters im Vergleich zur Ausrichtung der Antennen der iBeacons auf, so erfährt das empfangene Signal eine Dämpfung und es werden unrealistische Distanzen berechnet.

VIII. Ausblick

Mit Hilfe des Navigationsmoduls ließ sich die Position innerhalb des abgesteckten Areals mit dem in den Anforderungen definierten Sub-Meter Bereich bestimmen. Jedoch kann die dabei erreichbare Genauigkeit durch die in Kapitel VII beschriebenen Erweiterungen noch gesteigert werden. Dadurch könnte es sogar möglich werden, Positionsbestimmungen im Dezimeter-Bereich durchzuführen.
Eine weitere Möglichkeit die Genauigkeit des Navigationsmoduls zu steigern, besteht darin, die internen Sensoren der Roboterplattform wie z.B. Odometrie, Beschleunigungssensor, Gyroskop und Kompass zu nutzen und in die Berechnung mit einzubeziehen. In diesem Fall würde eine Sensordatenfusion angewandt werden, um aus den Messwerten der verschiedenen Sensoren eine Position mit einem reduzierten Fehler zu bestimmen.
Um priorisiert iBeacons zu verwenden, die die geringste Distanz zum Empfänger haben und damit das beste Empfangssignal liefern, könnte eine Tabelle implementiert werden. Diese Tabelle enthält Informationen über die Kennung der iBeacons, sowie über die Positionierung auf der Rasenfläche. Diese Tabelle müsste in einem weiteren *C*-Programm implementiert werden, da BASH nur eindimensionale Vektoren verarbeiten kann und daher keine Matrizen für eine 2-D oder 3-D Abbildung der iBeaconpositionen in der Umgebung implementieren kann [11]. Diesem Programm würde die ID des aktuellen iBeacons übergeben werden. Als Rückgabewert werden dessen Nachbarn in X- und Y-Richtung zurückgegeben.

Literatur

[1] **Elektronikkompendium**
http://www.elektronik-kompendium.de/sites/kom/1805171.htm
zugegriffen am 25. November 2014

[2] **Developer.Apple.com**
https://developer.apple.com/ibeacon/
zugegriffen am 20. November 2014

[3] **B.Sc.-Thesis: Bellersen, Jörn**
„Entwicklung einer autonomen Tracking - Antenne, basierend auf einem Embedded-System zur MAV-Kommunikation“, 2013

[4] **B.Sc.-Thesis: Oblonczek, Zahn**
https://users.informatik.haw-hamburg.de/übicomp/arbeiten/bachelor/oblonczek_zahn.pdf
zugegriffen am 4. Januar 2015

[5] **Chip**
http://www.chip.de/video/Maehroboter-von-Gardena-und-Bosch-Test-Video_63120870.html
zugegriffen am 19. Dezember 2014

[6] **Dr. Jan Wendel**
Integrierte Navigationssysteme - Sensordatenfusion, GPS und Iertiale Navigation, 2te überarbeitete Auflage
Oldenbourg Verlag München, 2011

[7] **Kowoma**
http://www.kowoma.de/gps/Genauigkeit.htm
zugegriffen am 22. Dezember 2014

[8] **OnyxBeacon**
http://www.onyxbeacon.de/beacon-hardware/
zugegriffen am 22. November 2014

[9] **Gay, Warren**
Mastering the Raspberry Pi - A Complete Reference Guide And Projekt Idea Generator For The Raspberry Pi
Apress, 2014

[10] **LXRobotics**
http://www.lxrobotics.com/berechnung-der-position-durch-odometriedaten
zugegriffen am 22. Dezember 2014

[11] **Wolf, Jürgen; Kania, Stefan**
Shell-Programmierung - Das umfassende Handbuch, 4te Auflage
Galileo Press Bonn, 2013

[12] **Follmann, Rüdiger**
Das Raspberry Pi Kompendium
Springer-Verlag Berlin Heidelberg, 2014

[13] **Kark, Klaus W.**
Antennen und Strahlungsfelder
Vieweg+Teubner Wiesbaden, 2010

Kooperative, verteilte Überwachung eines Gebiets auf Eindringlinge

Dennis Lisiecki, Torsten Kühl

I. Einleitung

Eine Überwachungskamera dient im allgemeinem dem Zweck, einen bestimmten Bereich dauerhaft zu überwachen. Das Bild wird kann direkt auf einem Monitor wiedergegeben und parallel aufgezeichnet werden, in Supermärkten erfolgt die Ausstrahlung des Live-Bilds sogar zur Abschreckung oftmals direkt im Verkaufsraum. Zum Schutz vor Vandalismus oder ähnlichem steigt auch im privaten Bereich die Verbreitung von Überwachungskameras. Dabei tendiert der Markt zu immer kleineren Modellen, mit höheren Auflösungen. Geräte solcher Art gibt es bereits zu Genüge, weswegen unser Projekt sich auf einen anderen Ansatz konzentriert.

Unser Ziel war es, ein System für eine kooperative, verteilte Überwachung mit mehreren Raspberry Pis zu realiseren. Die Raspberry Pis verfügen jeweils über eine Kamera und einen Bewegungsmelder den PIR-Sensor (Passive Infrared Sensor). Die Geräte sollen dazu an unterschiedlichen Stellen platziert werden, die den zu überwachenden Raum aus unterschiedlichen Blickwinkeln beobachten. Die Kamera des jeweiligen Geräts soll nun auf jegliche wahrgenommene Bewegung reagieren. Um die Genauigkeit zu erhöhen, soll die Kamera außerdem mit dem PIR-Sensor zusammenarbeiten.

Der PIR-Sensor reagiert lediglich auf Objekte, die eine Wärmesignatur ausstrahlen. Wenn nun durch die Kamera und den PIR-Sensor eine Bewegung erkannt wird, sendet der Raspberry Pi per Broadcast eine Nachricht in das Netzwerk, welche von dem Anwender für jegliche weiteren Schritte verwendet werden kann. Die Wahrscheinlichkeit, dass es sich um einen Mensch oder ein Tier handelt wäre in diesem Fall sehr hoch.

Um eine noch höhere Genauigkeit zu erzielen, sollen die Geräte bei unserem Ansatz das gleiche Gebiet aus verschiedenen Blickwinkeln beobachten:

Stellen mehrere Raspberry Pis eine Bewegung fest, wird der Anwender darauf hingewiesen, dass ein Eindringling das Gebiet betreten hat. Wie das funktioniert und in welcher Art und Weise die Systeme miteinander kommunizieren, wird im Folgendem erklärt.

II. System

A. Raspberry Pi

Der Raspberry Pi, welcher in unserem Projekt die Grundlage bildet, ist ein voll funktionsfähiger PC im Scheckkartenformat. In erster Linie wurde der Raspberry Pi mit dem Ziel entwickelt, interessierten Menschen das erlernen von hardwarenaher Programmierung zu erleichtern. Jedes Gerät besitzt ein frei programmierbares General Purpose Input Output-Board ("GPIO-Board"). Das GPIO-Board stellt je nach Modell 26 oder 40 Pins zur Verfügung, von denen 17 bzw. 26 Pins frei programmierbar sind und die weiteren der Spannungsversorgung oder als Masse dienen. Die einzelnen Pins dieses Boards lassen sich mit selbst programmierten Programmen ansteuern und können vielseitigen Zwecken dienen. So kann diverse externe Peripherie angesteuert werden, wie z.B. ein Temperatur-Sensor, ein Ultraschall-Sensor, ein Motor oder sogar ein kleiner externen Monitor mit Touch-Funktion. Als Betriebssystem können unter anderem an die Architektur angepasste Linux-Distributionen wie das auf Debian basierende *Raspbian* installiert werden. Auch Betriebssysteme, welche den Raspberry Pi zum Mediencenter umfunktionieren, um damit Filme und Musik abzuspielen, sind von der Community mittlerweile zur Verfügung gestellt worden. Wie solche Projekte zeigen, ist der Raspberry Pi nicht nur zum lernen gut geeignet. Die Video- und Audioausgabe erfolgt über eine HDMI-Schnittstelle, für die Audioausgabe steht alternativ auch ein 3,5mm Klinkenanschluss zur Verfügung. Für die Stromversorgung wird ein 5-V-Micro-USB-Anschluss genutzt. Hier stehen dem Anwender viele Türen offen:

Neben z.B. den meisten Handy-Ladegeräten kann die Stromversorgung auch über Batterie und Solarzelle erfolgen. So kann der Raspberry Pi auch mobil verwendet werden. Bis dato konnte sich der Raspberry Pi knapp 4 Millionen mal verkaufen und ist inzwischen in seiner vierten Version erschienen. [7] Die erste Version dieses Rechner kam Anfang 2012 auf den Markt und erfreut sich seither größter Beliebtheit. Je nach Ausführung ist das Gerät zwischen 25 und 35 Euro teuer. Die unterschiedlichen Ausführungen unterscheiden sich in gewissen Punkten:

Modell A und A+ besitzen 256 MB Arbeitsspeicher und nur einen USB-Anschluss, Modell B und B+ besitzen 512 MB Arbeitsspeicher, eine Ethernet-Schnittstelle sowie zwei, respektive vier USB-Anschlüsse. Das aktuellste Modell ist der Raspberry Pi 2 B (stand 05. Februar 2015). Der Raspberry Pi 2 B verfügt über 1.024 MB RAM, vier Cortex-A7-Kerne mit 900 MHz und soll den gleichen Preis kosten, wie der Raspberry Pi B+. Alle Modelle verwenden eine Micro-SD-Karten als Speichermedium. Für unsere Prototypen verwenden wir den Raspberry Pi B+.

B. Raspberry Pi Kameramodul

Für unsere Ausarbeitung verwenden wir das Raspberry Pi Infrarot Kamera Modul ("Pi NoIR Camera Board"), weil es vom Raspberry Pi selbst auf jeden Fall unterstützt wird und der Support der Raspberry Pi Community hervorragend

ist. *Das Camera Board und die PiNoIR sind kleine nackte Digitalkamera-Boards, die speziell für die Verwendung mit dem Raspberry Pi entwickelt wurden [...] . Das Camera Board kam als erstes auf den Markt und ist aufgrund des einfachen Anschlusses, der sehr leichten Bedienung und den vielfältigen Einsatzmöglichkeiten ein echtes Muss für Hobbybastler. Einige Zeit später wurde ein zweites Modell der Kamera entwickelt, das den Namen PiNoIR trägt. In der PiNoIR ist der gleiche Bildsensor wie in ihrem Vorgänger verbaut,und sie unterscheidet sich auch von der äußeren Form her nicht. Der PiNoIR fehlt lediglich der Infrarotfilter in der Kameralinse. Damit ist es möglich, infrarotes Licht zu erfassen. Eine normale Fernbedienung wird im Bild der PiNoIR plötzlich zur Taschenlampe, die einen dunklen Raum ausleuchtet.*[5, S. 511] Die Kamera bietet eine Auflösung von bis zu 5 Megapixel und kann bei statischen Aufnahmen mit einer Auflösung von bis zu 2592 x 1944 Pixel aufwarten. Mit Abmessungen von 25 x 20 x 9 mm ist die Kamera äußerst klein, muss allerdings auch ohne eigenes Gehäuse auskommen. Für die Raspberry Pi Kamera gibt es am Raspberry Pi einen eigenen Slot, in dem das Flachbandkabel der Kamera passt, sodass die GPIO-Anschlüsse am Raspberry Pi vollständig für andere Aufgaben verwendet werden können. Unser Prototypen verfügen über keine Infrarotlichtquellen, können jedoch um diese Funktion erweitert werden.

C. PIR-Sensor

Beim zweiten Sensor handelt sich um ein passiven Infrarot Sensor:
Der PIR-Sensor (Passive Infrared Sensor) ist einer der gängigsten Bewegungsmelder und ist oftmals auch an Außenleuchten oder Alarmanlagen verbaut. Erkennbar ist der PIR-Sensor an seiner meist runden, milchigen Kuppel, die mit vielen einzelnen Waben versehen ist. Der Sensor reagiert auf Temperaturveränderungen in seinem Sichtfeld. Somit können Menschen oder Tiere im Aktionsradius des Sensors erkannt werden. Jedoch kann der Sensor nur Veränderungen wahrnehmen. Bleibt ein Objekt ruhig im Bereich des Sensors stehen, so wird es nicht weiter erkannt. Sobald es sich weiterbewegt, schlägt der Sensor erneut an.[5, S. 493]

Im inneren eines solchen Moduls befinden sich zwei Folien, die an ihrer Oberfläche unterschiedliche elektrische Ladungen aufweisen. Trifft nun die Wärmestrahlung eines bestimmten Frequenzbereichs auf diese Folien, wird deren Polarisation verschoben und eine elektronische Spannung erzeugt, welche den Sensor zum auslösen bringt. Die milchige Kuppel auf dem Sensor erweitert den Erfassungsbereich des Sensors, indem es wie eine Anordnung von Linsen fungiert und lenkt die Wärmestrahlung direkt auf eine der beiden Folien. Bewegt sich nun eine Wärmequelle durch den vom Sensor überwachten Raum, kann man eine Bewegung über einen Bereich von weniger als 120 Grad und einer Entfernung von höchstens sieben Metern feststellen. Um den PIR Sensor am Raspberry Pi anzuschließen, werden insgesamt 3 GPIO Anschlüsse benötigt. Einen 5V Anschluss für die Stromversorgung, einen Ground und ein frei programmierbarer GPIO Pin, um den PIR-Sensor zu steuern. Die Ausgangsspannung des PIR-Sensors beträgt 3,3V und kann somit vom Raspberry Pi ohne elektronischen Widerstand dazwischen verarbeitet werden.

III. Codis: Kooperative, verteilte Überwachung

Um eine kooperative, verteilte Überwachung zu realisieren, haben wir das Programm Codis entwickelt. Codis steht für cooperative, distributed surveillance[1]. Entdeckt Codis einen Eindringling, sendet es eine INTRUDER Nachricht an alle Geräte im Netzwerk. Wie diese Nachricht verarbeitet wird und welche Maßnahmen eingeleitet werden sollen, wenn ein Eindringling entdeckt wurde, ist den Anwendern von Codis überlassen. In diesem Abschnitt befassen wir uns damit, wie Codis als verteiltes System ein Gebiet auf Eindringlinge überwacht. Dazu klären wir, wie Codis einen Eindringling entdeckt und wie aus Codis ein verteiltes System entsteht. Anschließend stellen wir fest, wie Codis als verteiltes System seine Effizienz verbessert und seine Genauigkeit beim Entdecken von Eindringlingen erhöht. Wir verwenden im Nachfolgenden den Begriff Sensoren für die Kamera und den PIR-Sensor eines Raspberry Pis.

A. Bewegungserkennung mit der Kamera

Der Raspberry verwendet einen H264 Kodierer zur Videokompression. Um Videos zu kodieren, verwendet der Kodierer ein Verfahren, das als Motion Estimation[2][8] bezeichnet wird. Dabei wird das Bild in 16x16 Pixel große Quadrate eingeteilt. Diese Quadrate bezeichnet man als Makroblöcke. Beim Kodieren von Videos vergleicht der H264 Kodierer das aktuelle Bild mit einem Referenzbild. Dazu untersucht der Kodierer jeden einzelnen Makroblock im aktuellen Bild und sucht nach dem ähnlichsten Makroblock im Referenzbild. Der jeweilige Abstand zwischen den Makroblock im aktuellen Bild und im Referenzbild wird vom Kodierer gespeichert und gilt auch als Ausmaß einer Bewegung. Anhand dieses Abstands kann gemessen werden, wie sehr sich ein Makroblock im Bild gegenüber dem Referenzbild bewegt hat. Bis zu diesem Schritt wurde der komplette Vorgang im H264 Kodierer implementiert, um Bewegungen zu erkennen.[10] Als nächstes nehmen wir uns für jedes Bild die Bewegungsdaten als Array und berechnen damit das Ausmaß an Bewegung, die im Bild stattfindet. Dazu berechnen wir das Ausmaß der einzelnen Vektoren im Array mit dem Satz des Pythagoras'. Eine Bewegung wurde dann erkannt, wenn im Array mindestens zehn Vektoren vorhanden sind, deren Bewegungsausmaß mindestens 60 beträgt.[4]

B. Codis als verteiltes System

Codis' Hauptfunktion ist es, ein Gebiet aus verschiedenen Blickwinkeln zu überwachen. Dieser Ansatz soll die Fehlertoleranz beim Erkennen von möglichen Eindringlingen verbessern. Dazu verwendet Codis mehrere Raspberry Pis, die sich in einem Netzwerk befinden und untereinander Nachrichten

[1] zu Deutsch: Kooperative, verteilte Überwachung
[2] wortwörtlich: Bewegungsvorhersage

austauschen. Wir verwenden im Folgenden den Begriff Codis-System für die Menge aller Raspberry Pis, auf denen Codis läuft und die miteinander in einem Netzwerk kommunizieren. Als Koordinator bezeichnen wir einen Raspberry Pi im Codis-System, der spezielle Aufgaben übernimmt, die wir im Laufe dieses Abschnittes klären. Der Koordinator ist allen Raspberry Pis im Codis-System bekannt und wird innerhalb des Codis-Systems ausgewählt. Das Codis-System wird dann erzeugt, wenn ein Raspberry Pi Codis ausführt, während kein anderer Raspberry Pi im Netzwerk Codis ausführt und wird dann zerstört, wenn Codis vom letzten Raspberry Pi im Codis-System beendet wird. Um die Implementierung des Prototypen einfach zu halten, verwendet Codis das UDP-Protokoll.

1) Codis-Liste: Codis verwendet eine verteilte Liste der Netzwerkadressen aller Raspberry Pis im Codis-System. Als verteilte Liste bezeichnen wir eine Liste, die auf allen Geräten eines verteilten Systems lokal abgespeichert ist und stets redundant zu den lokal abgespeicherten Listen der jeweils anderen Geräten ist. Die verteilte Liste, die Codis verwendet, bezeichnen wir im Folgenden als Codis-Liste. Die Codis-Liste wird zu Koordinationszwecken zwischen den Raspberry Pis im Codis-System benötigt. Die Codis-Liste wird dann gebildet, wenn das Codis-System erzeugt wird. Möchte ein Raspberry Pi dem Codis-System beitreten, sendet dieser eine JOINREQUEST Nachricht an das Codis-System. Daraufhin horcht der Raspberry Pi fünf Sekunden lang auf eine JOINRESPONSE Nachricht, die von allen Raspberry Pis im Codis-System versendet wird. Die JOINRESPONSE Nachricht enthält die Position des Absenders in der Codis-Liste. Nachdem der Koordinator seine JOINRESPONSE Nachricht versendet hat, wartet er zwei Sekunden und sendet daraufhin eine COORDINATOR Nachricht an den Raspberry Pi, der dem Codis-System beitreten möchte. Die COORDINATOR Nachricht, enthält die Position des Koordinators in der Codis-Liste. Erhält der Raspberry Pi die COORDINATOR Nachricht, dann trägt er ein, welche Position der Koordinator in der Codis-Liste hat. Erhält der Raspberry Pi nach fünf Sekunden keine JOINRESPONSE Nachricht, trägt er sich als Erster in die Codis-Liste ein und ist somit auch der Koordinator im Codis-System. Hat der Raspberry Pi von allen anderen Raspberry Pis im Codis-System eine JOINRESPONSE Nachricht erhalten, trägt er sich ans Ende der Codis-Liste ein und sendet eine JOIN Nachricht an alle Geräte im Codis-System. Erhält ein Raspberry Pi eine JOIN Nachricht, trägt er den Absender der JOIN Nachricht ans Ende seiner Codis-Liste ein.

2) Wahl eines Koordinators: Der Raspberry Pi von dem aus das Codis-System erzeugt wurde, wird als erster Koordinator ausgewählt. Betreten weitere Raspberry Pis das Codis-System, wird ein modifizierter Ringalgorithmus[9][1] ausgeführt, der in einem Intervall von 15 Minuten einen neuen Koordinator auswählt. Der Ringalgorithmus, den wir für Codis modifiziert haben, wird in der Literatur folgendermaßen beschrieben: *Ein weiterer Wahl-Algorithmus basiert auf der Verwendung eines Rings. Anders als einige andere Ring-Algorithmen verwendet dieser kein Token. Wir setzen voraus, dass die Prozesse physisch oder logisch geordnet sind, sodass jeder Prozess weiß, wer sein Nachfolger ist. Wenn ein Prozess erkennt, dass der Koordinator nicht funktioniert, erzeugt er eine ELECTION-Nachricht mit seiner eigenen Prozessnummer und sendet die Nachricht an seinen Nachfolger. Ist der Nachfolger nicht aktiv, überspringt das Senden diese und geht zum nächsten Mitglied entlang des Rings, oder auf den wiederum nachfolgenden, bis ein aktiver Prozess gefunden ist. Bei jedem Schritt trägt der Sender seine eigene Prozessnummer in die Liste der Nachrichten ein, wodurch er sich selbst zu einem Kandidaten macht, der als Koordinator gewählt werden kann. Irgendwann gelangt die Nachricht zurück zu dem Prozess, der das Ganze gestartet hat. Dieser Prozess erkennt dieses Ereignis, wenn er eine einkommende Nachricht empfängt, die seine eigene Prozessnummer enthält. Jetzt wird der Nachrichtentyp in COORDINATOR geändert und noch einmal gesendet, um jeden darüber zu informieren, wer der Koordinator ist (das Listenelement mit der höchsten Nummer) und wer die Mitglieder des neuen Rings sind. Nachdem diese Nachricht einmal den Umlauf gemacht hat, wird sie entfernt, und jeder geht wieder an die Arbeit*[9, S. 300] Um einen logischen Ring darzustellen, verwendet Codis die Codis-Liste. Die Raspberry Pis im Codis-System sind anhand ihrer Position in der Codis-Liste aufsteigend, im Ring angeordnet. Wird ein neuer Koordinator ausgewählt, verschickt der derzeitige Koordinator eine ELECTION Nachricht, an seinen Nachfolger im Ring. Erhält ein Raspberry Pi eine ELECTION Nachricht, dann trägt er sich als neuer Koordinator ein und sendet eine COORDINATOR Nachricht an alle Raspberry Pis im Codis-System. Empfängt der Raspberry Pi, der die ELECTION Nachricht gesendet hat, nach fünf Sekunden keine COORDINATOR Nachricht, sendet er eine HEARTBEATREQUEST an seinen Nachfolger. Erhält er daraufhin eine HEARTBEATRESPONSE Nachricht zurück, sendet er die ELECTION Nachricht erneut an seinen Nachfolger. Empfängt er dagegen keine HEARTBEATRESPONSE von seinem Nachfolger, dann entfernt er diesen aus der Codis-Liste und sendet eine LISTUPDATE Nachricht, an das Codis-System, damit der Nachfolger aus der Codis-Liste entfernt wird. Danach versucht der Raspberry Pi die ELECTION Nachricht, an seinen nächsten Nachfolger zu senden.

3) Abwechselnde Überwachung: Unter abwechselnder Überwachung verstehen wir, dass nur der Koordinator das Gebiet überwacht, bis der nächste Koordinator gewählt wurde. Währenddessen gelten alle anderen Raspberry Pis als inaktiv. Inaktive Raspberry Pis haben ihre Sensoren abgeschaltet und warten auf eine ELECTION Nachricht. Entdeckt der Koordinator einen möglichen Eindringling, dann sendet dieser eine INTRUDER Nachricht an alle anderen Raspberry Pis und geht in einen Alarmzustand über. Empfängt ein inaktiver Raspberry Pi diese Nachricht, geht dieser auch in einen Alarmzustand über. Im Alarmzustand sind die Sensoren des Raspberry Pis aktiviert. Entdeckt ein Raspberry Pi im Alarmzustand für fünf Minuten keinen Eindringling, dann geht er wieder in den inaktiven Zustand, außer der Koordinator der den Alarmzustand lediglich verlässt. Wird ein Raspberry Pi im Alarmzustand zum Koordinator, verbleibt er im Alarmzustand. Wird ein neuer Koordinator gewählt, verbleibt der vorherige Koordinator im Alarmzustand, falls er sich in diesem bereits befindet. Das verhindert, dass der vorherige Koordinator inaktiv wird, wenn sein Nachfolger zum Koordinator wird, während ein Eindringling entdeckt wird.

C. Entdecken eines Eindringlings

Entdeckt Codis in mehreren Bildern für einen kurzen Abstand eine Bewegung, setzt es ein MOTION Flag auf WAHR. Das MOTION Flag wird nach 50 Bildern, in denen keine Bewegung erkannt wurde, auf FALSCH gesetzt. Ein möglicher Eindringling wird dann erkannt, wenn der PIR-Sensor Bewegungen dann erkennt, während das MOTION Flag auf WAHR gesetzt ist. Entdeckt ein Raspberry Pi im Codis-System einen möglichen Eindringling, dann sendet dieser eine INTRUDER Nachricht an den Koordinator. Empfängt der Koordinator eine INTRUDER Nachricht, merkt er sich, von welchem Raspberry Pi diese Nachricht versendet wurde und wann die Nachricht versendet wurde. Entdeckt der Koordinator einen möglichen Eindringling, dann prüft er, ob ein anderer Raspberry Pi in den letzten drei Sekunden einen möglichen Eindringling entdeckt hat. Ist das der Fall, sendet der Koordinator eine INTRUDER Nachricht ans Netzwerk.

IV. Andere Ansätze (bzw. Verwandte Arbeiten)

Eine Realisierung einer kooperativen, verteilten Überwachung mit dem Raspberry Pi konnte bei unserer Recherche nicht gefunden werden. Es existiert jedoch eine Lösung, ein Linux-System als pure Überwachungskamera nutzbar zu machen. Für diese Lösung wird das C-Programm Motion verwendet[6], das wir uns für unsere Ausarbeitung angesehen haben, da es im Kern am ehesten mit unserer Ausarbeitung vergleichbar ist und die am weitesten verbreitete Open Source Lösung dieser Art darstellt.
Motion verfolgt das Ziel, aus dem Computer mit angeschlossener USB-Webcam eine Überwachungskamera zu erstellen. Der Bewegungserkennungsalgorithmus von Motion ist der gleiche, der im H264 Kodierer des Raspberry Pis bereits implementiert wurde. Das heißt, nutzt man Motion auf dem Raspberry Pi um Bewegungen zu erkennen, führt Motion den Bewegungserkennungsalgorithmus aus, obwohl dies bereits vom H264 Kodierer des Raspberry Pis erledigt wurde. Eine erkannte Bewegung führt bei Motion dazu, dass wahlweise ein Foto oder Video aufgezeichnet. Ebenso kann der aktuelle Video-Stream der Kamera abgegriffen werden, um das aktuelle Geschehen selbst zu beobachten. Motion kann von einem Anwender auch so eingestellt werden, dass ein Quadrat um eine Bewegung im Kamerabild gezeichnet wird.
Motion lässt sich im Originalzustand nicht mit dem Raspberry Pi Kameramodul betreiben, es gibt allerdings Projekte, die mit einer modifizierten Version von Motion das Raspberry Pi Kameramodul nutzen. Motion lässt sich durch den Anschluss mehrerer Kameras erweitern, wobei diese sich untereinander in ihrer Funktionsweise nicht beeinflussen. Der Ansatz mit Motion war für uns dahingehend uninteressant, weil durch Motion keine kooperative, verteilte Überwachung realisiert wird. Außerdem bietet der Ansatz mit Motion nicht die Möglichkeit, auf einen Eindringling zu reagieren, wenn dieser entdeckt wurde. In der von uns verwendeten Literatur, findet sich ein Beispielprojekt einer Überwachung, die mit Motion realisiert wurde. Bei diesem Beispielprojekt, wird der Ansatz mit Motion dazu verwendet, das Innere eines Vogelhaus zu observieren.[5]

V. Evaluation

Dieser Abschnitt fasst zusammen, welche Vorteile Codis gegenüber nicht-verteilten Überwachungssystemen hat. Außerdem setzen wir uns damit auseinander, ob Codis bestimmte Eigenschaften eines verteilten Systems erfüllt. Aber zuallererst stellen wir uns die Frage, ist Codis überhaupt ein verteiltes System? Eine Definition von verteilten Systemen lautet: *Bei einem verteilten System arbeiten Komponenten zusammen, die sich auf vernetzten Computern befinden und die ihre Aktionen durch den Austausch von Nachrichten koordinieren. Aus dieser Definition leiten sich die folgenden Eigenschaften verteilter Systeme ab: Nebenläufigkeit der Komponenten, es gibt keine globale Uhr und die Komponenten können unabhängig voneinander ausfallen.*[1, S. 17] Die einzelnen Raspberry Pis im Codis-System arbeiten nebeneinander. Codis verfügt auch über keine globale Uhr, sondern nutzt jeweils die lokalen Uhren der einzelnen Raspberry Pis. Ferner können einzelne Raspberry Pis ausfallen, während Codis ein Gebiet überwacht.

A. CPU-Auslastung als verteiltes System

Codis bietet als verteiltes System eine bessere CPU-Auslastung gegenüber nicht-verteilten Überwachungssysteme. Das ermöglicht die abwechselnden Überwachung von Codis. Während bei einem nicht-verteilten Überwachungssystem ein Raspberry Pi das Gebiet kontinuierlich überwacht, ist bei einer verteilten Überwachung nur der Koordinator des Codis-Systems aktiv. Überwacht ein Raspberry Pi mit Codis das Gebiet, beansprucht er ungefähr 30 Prozent an CPU Leistung, ohne einen Eindringling zu entdecken. Das heißt, ohne der abwechselnden Überwachung wäre die CPU jedes Raspberry Pis im Codis-System dauerhaft für 30 Prozent ausgelastet. Durch die abwechselnde Überwachung, teilen sich die Raspberry Pis die CPU-Last auf, indem nur einer für ein Intervall von 15 Minuten das Gebiet überwacht, bis ein Eindringling entdeckt wurde und daraufhin alle Raspberry Pis im Gebiet aktiv werden. Ein Vorteil ist der geringere Stromverbrauch durch die bessere CPU-Auslast. Die Raspberry Pis, die mit Codis ein Gebiet im Freien überwachen, müssten mit externen Akkus betrieben werden. Durch die abwechselnde Überwachung wird der Akku eines Raspberry Pis weniger beansprucht.

B. Evaluation der Ausfalltoleranz von Codis

Codis hat als verteiltes System den Vorteil, dass das Codis weiterhin funktioniert, wenn darin einzelne Raspberry Pis ausfallen. Codis fällt erst dann aus, wenn alle Raspberry Pis im Codis-System ausgefallen sind. Codis wurde so entwickelt, dass es sich selbst reorganisieren kann, wenn Raspberry Pis ausfallen. Das bedeutet, wenn z. B. ein Raspberry Pi unerwartet abstürzt, wird das vom Codis-System erkannt und der entsprechende Raspberry Pi wird aus der Codis-Liste entfernt. Dazu nutzt Codis ein Heartbeat-System, das vom Web Framework Vaadin eingesetzt wir, um zu erkennen, ob eine Browsersitzung erloschen ist. *UI instances are cleaned up if no communication is received from them after some time. If no other server requests are made, the client-side sends keep-alive heartbeat requests. A UI is kept alive for as long as*

requests or heartbeats are received from it. It expires if three consecutive heartbeats are missed.[3, S. 95] In Codis erhält jeder Raspberry Pi von seinem Nachfolger im Ring in einem Interval von zwei Minuten eine HEARTBEAT Nachricht. Bleiben drei HEARTBEAT Nachrichten nacheinander aus, entfernt der Raspberry Pi seinen Nachfolger vom Codis-System und informiert darüber alle anderen Raspberry Pis im Codis-System, damit diese das Gleiche tun. Die HEARTBEAT Nachricht empfängt nicht speziell der Koordinator, damit Codis auch mit dem des Koordinators leicht umgehen kann. Der schlimmste Fall bei einem Ausfall des Koordinators wäre, wenn der Koordinator dann ausfällt, kurz nachdem dieser eine HEARTBEAT Nachricht gesendet hat. In diesem Fall könnte Codis einen Anwender für sechs Minuten lang nicht darüber informieren, ob ein Eindringling das Gebiet betreten hat. Sollten bis auf ein Raspberry Pi alle anderen im Codis-System ausfallen, informiert der noch vorhandene Raspberry Pi einen Anwender über Eindringlinge, obwohl mindestens zwei Raspberry Pis im Codis-System einen Eindringling entdecken müssten. Das passiert aber nur dann, wenn der letzte Raspberry Pi im Codis-System der Koordinator ist. Das ermöglicht auch, Codis als nicht-verteiltes System zu nutzen. Sämtliche Mechanismen für eine verteilte Überwachung, werden dann nicht mehr von Codis ausgeführt.

C. Evaluation der Fehlertoleranz Codis

Das Problem mit Ansätzen, die Programme wie Motion verwenden, ist, dass diese Überwachungssysteme nicht fehlertolerant ist. Möchte man ein Gebiet überwachen, in dem es windig ist und Dinge wie z. B. Blätter umhergewirbelt werden, dann kann es passieren, dass oft Bewegungen erfasst werden, die aber von keinem Eindringling verursacht werden. Ein Raspberry Pi in Codis hat dagegen den Vorteil, dass Eindringlinge nur entdeckt werden, wenn Kamera und PIR-Sensor ungefähr gleichzeitig Bewegungen erkennen. Der PIR-Sensor erkennt dabei nur Bewegungen, die von einer Wärmequelle verursacht werden.[5] Dadurch dass Codis als verteiltes System Gebiete aus verschiedenen Blickwinkeln überwacht, ist Codis präziser darin, einen Eindringling zu entdecken, gegenüber nicht-verteilten Überwachungssystemen. Das Ergebnis ist, dass die Wahrscheinlichkeit für Fehlalarm bei Codis geringer ist.

VI. Zusammenfassung

Durch seine kompakte Bauform und der frei programmierbaren GPIO-Schnittstelle ist der Raspberry Pi für viele Anwendungsbereiche wie geschaffen. Er kann völlig kabellos betrieben werden und findet Platz in den kleinsten Ecken. So ist es nichts ungewöhnliches, dass das System auch als Überwachungskamera eingesetzt wird. Dabei existiert jedoch kein Ansatz, der in diesem Zusammenhang auch eine kooperative, verteilte Überwachung mit dem Raspberry Pi unterstützt. In unserer Ausarbeitung haben wir uns diesem Umstand angenommen. Unsere Motivation bestand darin, das Vorhandensein mehrerer Raspberry Pis möglichst effektiv auszunutzen. Mit Codis besteht die Möglichkeit, ein leicht zu erweiterndes Netzwerk zur Überwachung zu errichten, bei welchem sich die Geräte gegenseitig in den Alarmzustand versetzen. Wie der Alarmzustand im Endeffekt genutzt wird, kann vielfältiger Natur sein und steht dem Anwender offen. Die leichte Erweiterbarkeit ist einer der größten Vorteile, der dadurch gegeben ist, dass Codis auf Veränderungen im Codis-System völlig autonom reagiert. Zusätzlich wird sehr sparsam mit den Ressourcen umgegangen, da in einem Codis-System nur ein Koordinator existiert, der alle Sensoren aktiv schaltet. Gerade bei Verwendung mit einem Akku ist der Unterschied enorm. Wir möchten auf eine Android-App hinweisen, die wir zusammen mit dem Codis Prototypen entwickelt haben. Diese Android-App kann die Netzwerknachrichten, die Codis sendet, empfangen und weist somit daraufhin, wenn Codis einen Eindringling entdeckt hat. Diese Android-App lässt sich im Google Play Store unter den Namen Surveillance Receiver kostenlos runterladen.

Literatur

[1] George Coulouris, Jean Dollimore und Tim Kindberg. *Verteilte Systeme : Konzepte und Design*. München : Pearson Studium, 2003.

[2] *Elektronikwissen zu: PIR-Sensor - Aufbau und Funktion*. URL: http://www.elv.de/controller.aspx?cid=758&detail=10&detail2=129.

[3] Marko Grönroos. *Book of Vaadin: Vaadin 7 Edition - 2nd Revision*. Vaadin Ltd, 2014.

[4] Hughes, Dave. *Recording motion vector data*. URL: http://picamera.readthedocs.org/en/release-1.8/recipes2.html#recording-motion-vector-data.

[5] Michael Kofler. *Raspberry Pi : das umfassende Handbuch ; [Grundlagen verstehen, spannende Projekte realisieren ; Schnittstellen des Pi, Schaltungsaufbau, Steuerung mit Python ; Erweiterungen für den Pi: Gertboard, PiFace, Quick2Wire u.a. in Hardware-Projekte einsetzen ; aktuell zu allen Versionen, inkl. Modell B+]*. Bonn : Galileo Press, 2014.

[6] Lavrsen, Kenneth. *Motion - Web Home*. URL: http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome.

[7] Mirko Lindner. *Über 3,8 Millionen Raspberry Pi verkauft*. Okt. 2014. URL: http://www.pro-linux.de/news/1/21613/ueber-38-millionen-raspberry-pi-verkauft.html.

[8] Shilpa Metkar und Sanjay Talbar. *Motion Estimation Techniques for Digital Video Coding*. Springer India, 2013.

[9] Andrew S. Tanenbaum und Martinus R. van Steen. *Verteilte Systeme : Grundlagen und Paradigmen. Distributed Systems ¡dt.¿* München : Pearson Studium, 2003.

[10] Upton, Liz. *Vectors from coarse motion estimation*. URL: http://www.raspberrypi.org/vectors-from-coarse-motion-estimation.