UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

Konzeption und prototypische Implementierung einer „DogCam“

Bachelorarbeit

zur Erlangung des Grades Bachelor of Science (B.Sc.)
im Studiengang Informatik

vorgelegt von

Bernhard Szudra

Erstgutachter:	Prof. Dr. Stefan Müller (Institut für Computervisualistik, AG Computergrafik)
Zweitgutachter:	M.Sc. Nils Höhner (Institut für Computervisualistik, AG Computergrafik)

Linkenbach, im Februar 2019

Erklärung

Ich versichere,

dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	☐	☒

Linkenbach, 01. Februar 2019

Bernhard Szudra

Abstract

Ziel dieser Arbeit ist es, ein einfaches Konzept zur Überwachung von Hunden, die mehrere Stunden alleine zu Hause sind, zu entwickeln. Die prototypische Implementierung einer solchen „DogCam“ kann als Proof of Concept angesehen werden.

Die Grundlage für die Implementierung des Prototypen sind die im Rahmen einer Anforderungsanalyse herausgearbeiteten Anforderungen.

Weiterhin zeigt die vorliegende Arbeit auf, welche Verbesserungen und Erweiterungen der prototypischen „DogCam“ möglich sind und welche ähnlichen Projekte bereits existieren.

The aim of this work is to develop a simple concept for monitoring dogs that are alone at home for several hours. The prototypical implementation of such a "DogCam" can be considered as proof of concept.

The basis for the prototype‘s implementation are the requirements identified within a conducted requirement analysis.

Furthermore, the present work shows which improvements and extensions of the prototypical "DogCam" are possible and which similar projects already exist.

Inhaltsverzeichnis

1 Einleitung

(Berufstätige) Hundebesitzer, wie u.a. der Autor der vorliegenden Arbeit, können ihren Hund nicht überall hin mitnehmen. Dennoch ist es von Interesse zu wissen, wie der Hund sich verhält, wenn er alleine zu Hause ist. Ziel dieser Arbeit ist daher die Konzeption und prototypische Entwicklung eines Systems zur Überwachung von Hunden im heimischen Umfeld (Wohnung/ Haus).

Dabei soll ein Raspberry Pi mit einer Kamera ausgestattet werden und regelmäßig Einzelbilder aufnehmen. Diese Bilder sollen über eine Weboberfläche, optional auch über eine Smartphone-App (Android), abgerufen werden können. Der Schwerpunkt dieser Arbeit liegt dabei nicht auf der Entwicklung eines schönen (Web-/ App-) Frontends, sondern auf der Konzeption und prototypischen Umsetzung eines solchen Systems.

Im zweiten Kapitel dieser Arbeit werden zunächst die Grundlagen für die Konzeption und prototypische Implementierung der „DogCam“ dargestellt. Anschließend werden in Kapitel 3 die Anforderungen an die „DogCam“ und ihr Konzept beschrieben. Weiterhin wird ihre Implementierung dargelegt. Kapitel 4 beschreibt und bewertet die erzielten Ergebnisse der vorliegenden Arbeit, bevor in Kapitel 5 das Fazit gezogen wird.

2 Grundlagen

Das vorliegende Kapitel beschreibt den Raspberry Pi, der als Hardware für die „DogCam“ genutzt wird. Weiterhin werden Anwendungen, die der „DogCam“ ähnlich sind, vorgestellt.

2.1 Raspberry Pi

Der Raspberry Pi gehört mit mehr als 17 Millionen verkauften Stück seit 2012 [1] zu den wohl beliebtesten Einplatinencomputern weltweit. Das Ziel der Raspberry Pi Foundation ist es, mit dem Raspberry Pi einen günstigen Computer zu entwickeln mit dem jeder das Programmieren lernen kann. [2]

Inzwischen gibt es eine Vielzahl von Projekten aus unterschiedlichen Kategorien (z.B. Smart Home, Internet of Things), die mit dem Raspberry Pi realisiert wurden. [3] Die Herausforderung bei allen Projekten besteht darin, die begrenzten Ressourcen des Einplatinencomputers optimal zu nutzen.

MODELL	CPU	RAM
RASPBERRY PI 1 MODEL A	700 MHz, Single Core	256 MB
RASPBERRY PI 1 MODEL B	700 MHz, Single Core	512 MB
RASPBERRY PI 2 MODEL B	900MHz, Quad Core	1 GB
RASPBERRY PI 3 MODEL B	1.2GHz, Quad Core, 64 Bit	1 GB

Tabelle 1: Vergleich ausgewählter Raspberry Pi Modelle [4] [5] [6] [7]

Für die vorliegende Arbeit wird der Raspberry Pi 3 Model B verwendet. Er kam im Februar 2016 auf den Markt und bietet deutlich mehr Leistung als seine Vorgängermodelle (siehe Tabelle 1).

BEZEICHNUNG	BESCHREIBUNG
CHIPSATZ	Broadcom BCM2837
ARCHITEKTUR	ARM Cortex-A
CPU	1,2 GHz Quad Core, 64 Bit, ARM Cortex-A53
GPU	Broadcom Dual Core VideoCore IV
ARBEITSSPEICHER	1 GB
NETZWERK	10/100 Mbit/s (Ethernet), Broadcom BCM43143 2,4 GHz, b/g/n (WLAN)
BLUETOOTH	4.1 LE
USB	4x USB-2.0
VIDEO	HDMI (Typ A)

AUDIO	HDMI (digital); 3,5-mm-Klinkenstecker (analog)
BETRIEBSSYSTEM	Raspbian (Debian-basierte Linuxdistribution)
MAßE	93,0mm x 63,5mm x 20,0mm (gesamt) 85,6mm x 56mm (Platine)
STROMVERSORGUNG	5V via Micro-USB-B

Tabelle 2: Spezifikation des Raspberry Pi 3 Model B (gekürzt) [7]

Die Spezifikation des Raspberry Pi 3 Model B (siehe Tabelle 2) erlaubt es, den Raspberry Pi auch für komplexere Anwendungen zu nutzen. Aktuell ist der Raspberry Pi in Version 3 Model B+ erhältlich, in der eine 1,4 GHz Quad Core CPU (Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit) verbaut ist.

2.2 Ähnliche Anwendungen

Es existieren einige ähnliche Anwendungen wie die im Rahmen der vorliegenden Arbeit konzipierten und implementierten „DogCam“. Dabei wird oft die Bibliothek motion[1] oder auch motionEyeOS[2] verwendet:

- Artikel „Webserver für Überwachungskamera einrichten“ [8]
- Artikel „Raspberry Pi: Überwachungskamera Livestream einrichten“ [9]
- Artikel „Mit dem Raspberry Pi die Wohnung überwachen“ [10]
- Artikel „Raspberry Pi Security System With Motion Detection / Camera“ [11]

Andere Anwendungen werden auch um Funktionalitäten wie z.B. Gesichter-/ Bild- und Bewegungserkennung erweitert, wobei dabei die Überwachung eine untergeordnete oder keine Rolle spielt:

- Artikel „Raspberry Pi Image Recognition with Alexa Voice“ [12]
- Artikel „Raspberry Pi Facial Recognition“ [13]

Die vorliegende Arbeit baut allerdings nicht auf einem bestehenden Projekt auf. Sie soll aufzeigen, dass auch ohne die Nutzung fertiger Bibliotheken, wie z.B. motion, die Implementierung der „DogCam“ auf einfache Art möglich ist und zudem eigene Weiterentwicklungen offenlässt (siehe auch Abschnitt 4.3).

[1] https://motion-project.github.io/; Zugriff am 28. Januar 2019

[2] https://github.com/ccrisan/motioneyeos; Zugriff am 28. Januar 2019

3 Konzept und Implementierung

Ziel des aktuellen Kapitels ist die Konzeption und die Beschreibung der prototypischen Implementierung der „DogCam“. Dabei wird zunächst die Anforderungsanalyse erläutert. Die daraus resultierenden Anforderungen bilden die Basis der Implementierung.

3.1 Anforderungsanalyse

Für Erstellung der Spezifikation der „DogCam“ wurde eine Anforderungsanalyse durchgeführt. Diese beinhaltet zum einen die Ermittlung der Stakeholder und die Ziele der „DogCam“, aus denen anschließend die Anforderungen formuliert werden.

3.1.1 Ermittlung der Stakeholder

Für das in der vorliegenden Arbeit durchgeführte Projekt konnten zwei Stakeholder identifiziert werden. Zum einen ist das der Autor dieser Arbeit als Entwickler der „DogCam“, zum anderen ist das eine weitere Person, die im Haushalt des Autors wohnt, als Nutzer des Systems. Weitere mögliche Stakeholder, wie z.B. andere Entwickler, sonstige Nutzer, wurden in dieser Anforderungsanalyse nicht berücksichtigt.

3.1.2 Ziele

Als Ziele der prototypischen Implementierung der „DogCam“ wurden identifiziert:

- Möglichst einfache und schnelle Implementierung
- Nutzer sollen mittels Browser auf die „DogCam“ zugreifen können
- Die „DogCam“ muss über das Internet verfügbar sein
- Die Hardware muss transportabel sein
- Hardwareschäden müssen einfach und kostengünstig zu beheben sein
- Die Software muss (einfach) erweiterbar sein

3.1.3 Anforderungen

Die formulierten Anforderungen an die prototypische Implementierung der „DogCam“ sind nummeriert und nach den Teilsystemen Hardware und Software geordnet. Sie ergeben sich aus den im vorangehenden Abschnitt formulierten Zielen.

#	TEILSYSTEM	ANFORDERUNG
H1	**Hardware**	Basis ist ein Raspberry Pi
H2		Eingesetzt wird das Standard-Kameramodul für den Raspberry Pi
H3		Eine microSD-Speicherkarte wird für das Betriebssystem eingesetzt
S1	**Software**	Raspbian wird in aktueller Version als Betriebssystem eingesetzt
S2		Die Implementierung der Kamerasteuerung erfolgt in Python[3]
S3		Apache 2[4] wird als Webserver genutzt
S4		Ein auf HTML und JavaScript basierendes Webfrontend wird implementiert
S5		1 Bild/ Sekunde muss aufgezeichnet werden
S6		1 Bild/ Sekunde soll im Webfronend dargestellt werden
S7		Router: Freigabe für lokale IP des Raspberry Pi über das Internet
S8		Raspberry Pi muss mind. 1 Tag unterbrechungsfrei laufen
S9		Mehrere Clients müssen gleichzeitig Zugriff haben

Tabelle 3: Definierte Anforderungen

Die in Tabelle 3 aufgelisteten Anforderungen werden in Abschnitt 4.1 auf ihre Erfüllung geprüft.

3.2 Konzept

Im aktuellen Abschnitt wird die Systemarchitektur der „DogCam“ erläutert.

Der Router wird so konfiguriert, dass der Raspberry Pi, der im lokalen Netzwerk eingebunden ist, über die IP des Internetanschlusses und einen bestimmten Port erreichbar ist. So können verschiedene Clients über das Internet auf die „DogCam“ zugreifen (Abbildung 1). Erleichtert

[3] https://www.python.org/; Zugriff am 28. Januar 2019

[4] https://httpd.apache.org/; Zugriff am 28. Januar 2019

wird dies über den Service MyFRITZ![5] der AVM GmbH[6], da als Router die FRITZ!Box 6360 Cable verwendet wird.

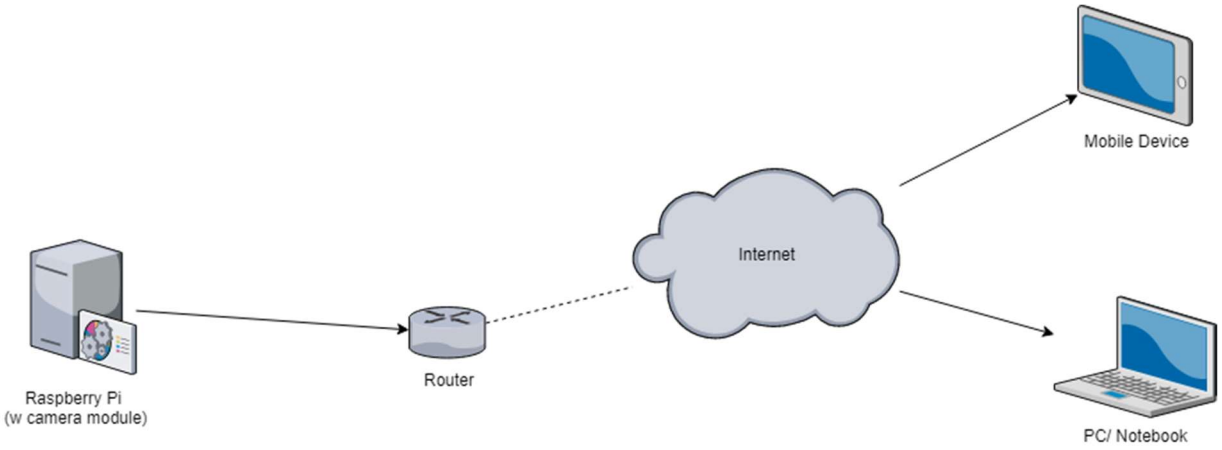

Abbildung 1: Systemarchitektur Gesamtprojekt (eigene Darstellung)

Der Raspberry Pi nutzt den Apache 2-Webserver. Ein Python-Skript erstellt gemäß den Anforderungen (S5, siehe Tabelle 3) regelmäßig Einzelbilder, die im Arbeitsverzeichnis des Apache 2

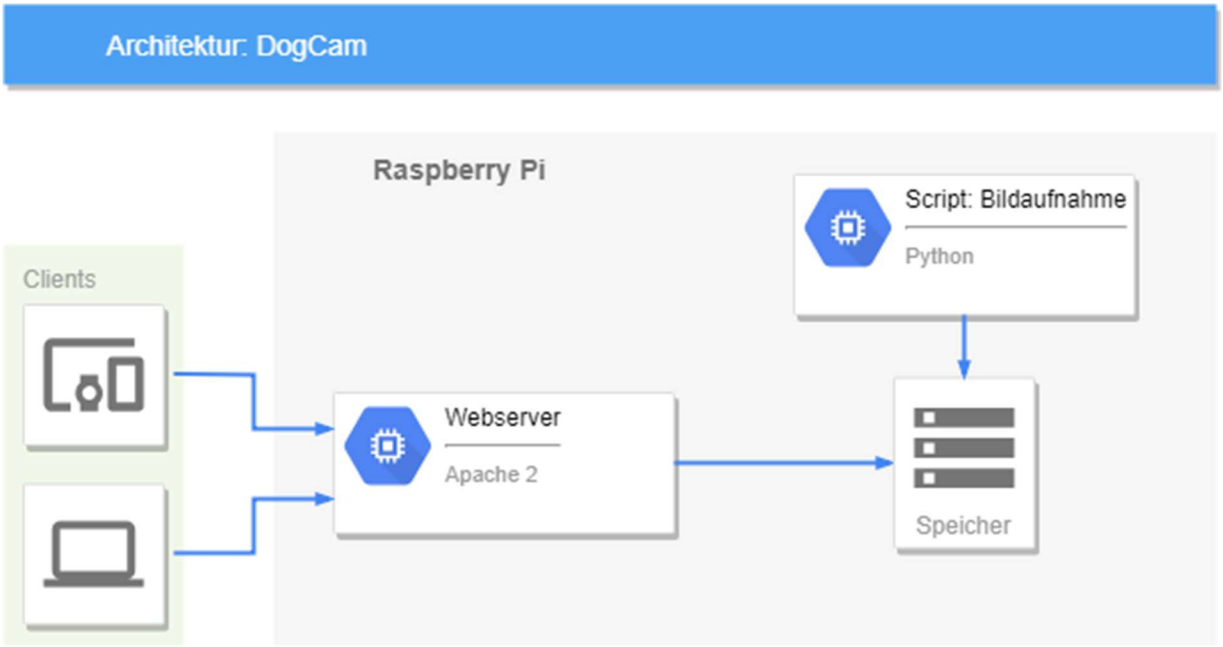

Abbildung 2: Systemarchitektur Raspberry Pi (eigene Darstellung)

immer unter dem Namen „image.jpg“ (siehe Code 1 Z. 19 & 22) gespeichert werden, d.h. es

[5] https://sso.myfritz.net/; Zugriff am 28. Januar 2019

[6] https://avm.de/; Zugriff am 28. Januar 2019

existiert genau eine Bilddatei, die der Webserver über das Webfrontend an die Clients ausliefert.

3.3 Implementierung

Um die „DogCam“ prototypisch zu implementieren, sind ein Pythonskript und eine HTML-Datei notwendig.

```
1.  import picamera
2.  import time
3.  import datetime as dt
4.
5.  camera = picamera.PiCamera()
6.  camera.resolution = (320, 240) # also possible (640, 480) or (1024, 768)
7.
8.  while True: # endless loop
9.      # record image every second
10.     time.sleep(0.46)
11.
12.     camera.start_preview()
13.
14.     # add current time to image
15.     camera.annotate_background = picamera.Color('black')
16.     camera.annotate_text = dt.datetime.now().strftime('%d.%m.%Y %H:%M:%S')
17.
18.     # path is the working directory of the apache 2 webserver
19.     path = '/var/www/html/image.jpg'
20.
21.     # capture image and write it to path
22.     camera.capture(path)
23.
24.     # uncomment following lines to append timestamp to image name – debug/ test only
25.     # ts = time.time()
26.     # camera.capture('/home/pi/ba/recording/image_%s.jpg' % ts)
27.
28.     camera.stop_preview()
```

Code 1: record-image.py

Um die Bilddateien klein zu halten und somit auch den Traffic bei der Nutzung, wird die Auflösung der Bilder auf 320px*240px festgesetzt (siehe Code 1, Z. 6).

Abbildung 3: Beispiel einer Aufnahme der DogCam

Damit der Nutzer weiß, wie alt das dargestellte Bild ist, wird die bei der Aufnahme des Bildes aktuelle Zeit in das Bild eingefügt (siehe Code 1, ZZ. 15-16, sowie Abbildung 3). Die Bilddatei wird direkt in das Arbeitsverzeichnis des Apache 2-Webservers geschrieben, so dass die HTML-Datei „index.html“ (siehe Code 2) direkten Zugriff auf sie hat (siehe Code 1, Z. 19 und Z. 22).

Bevor ein Bild aufgenommen und in das Arbeitsverzeichnis des Apache 2-Webservers geschrieben wird, pausiert das Pythonskript für 460ms (siehe Code 1, Z. 10). Laut Anforderung S5 (siehe Tabelle 3) muss 1 Bild/s aufgezeichnet werden. Die Latenz der Kamera, die nicht genau zu ermitteln ist, sowie die Pause von 460ms im Skript ergeben eine Gesamtpause von ca. 1s (siehe auch Abbildung 6).

```
1.  <!doctype html>
2.  <html>
3.      <head>
4.          <title>DogCam</title>
5.          <script src="https://ajax.google-
    apis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
6.      </head>
7.      <body id="body">
8.          <img id="image" src="image.jpg" alt="current image" />
9.      </body>
10.     <script type="text/javascript">
11.         function toDataUrl(url, callback) {
12.             var xhr = new XMLHttpRequest();
13.             xhr.onload = function() {
14.                 var reader = new FileReader();
15.                 reader.onloadend = function() {
16.                     callback(reader.result);
17.                 }
18.                 reader.readAsDataURL(xhr.response);
19.             };
20.             xhr.open('GET', url);
21.             xhr.responseType = 'blob';
22.             xhr.send();
23.         }
24.         setInterval(function() {
25.             toDataUrl("/image.jpg", function(base64) {
26.                 if(base64 !== "data:") { // error handling
27.                     $("#image").attr("src", base64);
28.                 }
29.             });
30.         }, 1000);
31.     </script>
32. </html>
```

Code 2: index.html

Das aufgenommene Bild liegt zwar im Arbeitsverzeichnis des Webservers und ist somit direkt abrufbar, muss aber so manuell aktualisiert werden. Um die Aktualisierung zu automatisieren, wurde die Datei index.html (siehe Code 2) implementiert. Das Bild wird asynchron geladen und Base64-kodiert in Form einer Data-URL im src-Attribut des Bild-Tags aktualisiert (siehe Code 2, ZZ. 11 - 30).

4 Ergebnisse

Das vorliegende Kapitel zeigt zunächst die Ergebnisse der prototypischen Implementierung der „DogCam“ auf und enthält weiterhin mögliche Verbesserungen oder Erweiterungen der „DogCam“.

4.1 Bewertung der prototypischen Implementierung

Zur Bewertung der Implementierung werden die in Abschnitt 3.1.3 beschriebenen Anforderungen auf ihre Erfüllung überprüft.

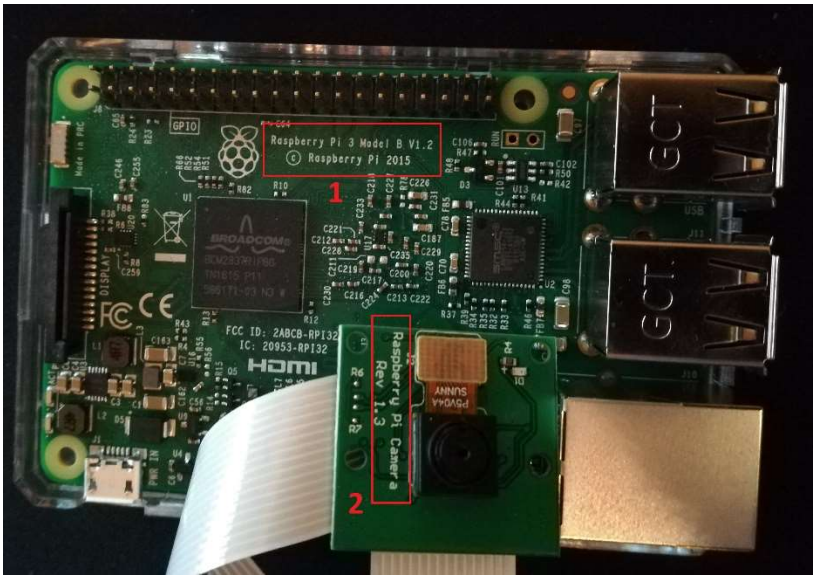

Abbildung 4: Raspberry Pi 3 Model B mit Kameramodul

Zum Einsatz kommt der Raspberry Pi 3 Model B V1.2 (siehe Abbildung 4, Nr. 1) mit dem Standard-Kameramodul Raspberry Pi Camera Rev 1.3 (siehe Abbildung 4, Nr. 2). Das Betriebssystem Raspbian in Version 4.4.21 ist auf eine MicroSD-Karte aufgespielt, was einen Austausch des Raspberry Pi bei Defekt oder Upgrade auf ein neueres Modell einfach macht. So sind die Anforderungen H1 - H3, sowie S1 (vgl. Tabelle 3) erfüllt.

Raspbian in der eingesetzten Version 4.4.21 beinhaltet Python in Version 2.7.9, sowie den Apache 2-Webserver, der das Webfrontend der „DogCam“ (siehe Code 2) bereitstellt. Dadurch werden die Anforderungen S2 - S4 (vgl. Tabelle 3) erfüllt.

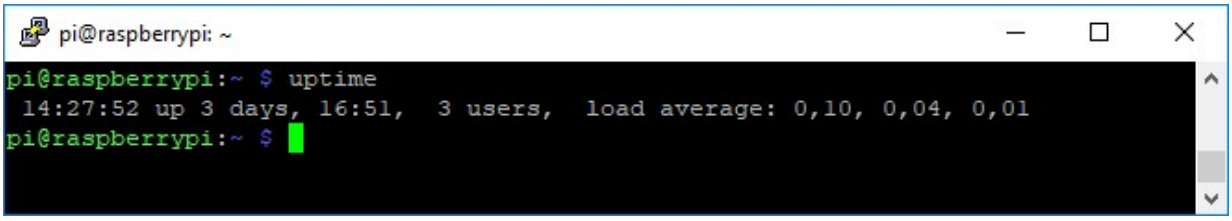

Abbildung 5: Unterbrechungsfreie Uptime des Raspberry Pi: 3 Tage und 17 Stunden

In mehreren Tests lief der Prototyp der „DogCam“ mehrere Tage unterbrechungsfrei (siehe Abbildung 5; vgl. Tabelle 3, S8). Ebenso wurde erfolgreich der parallele Zugriff mehrerer Clients (mind. 5, max. 10) auf das Webfrontend getestet (vgl. Tabelle 3, S9). Vorgenannter Test paralleler Zugriffe wurde nicht nur über das lokalen Netzwerk durchgeführt, sondern ebenso über das Internet (vgl. Tabelle 3, S7) mittels der durch den MyFRITZ!-Service zur Verfügung gestellten URL für den externen Zugriff auf die eingesetzte FRITZ!Box (siehe auch Abschnitt 3.2).

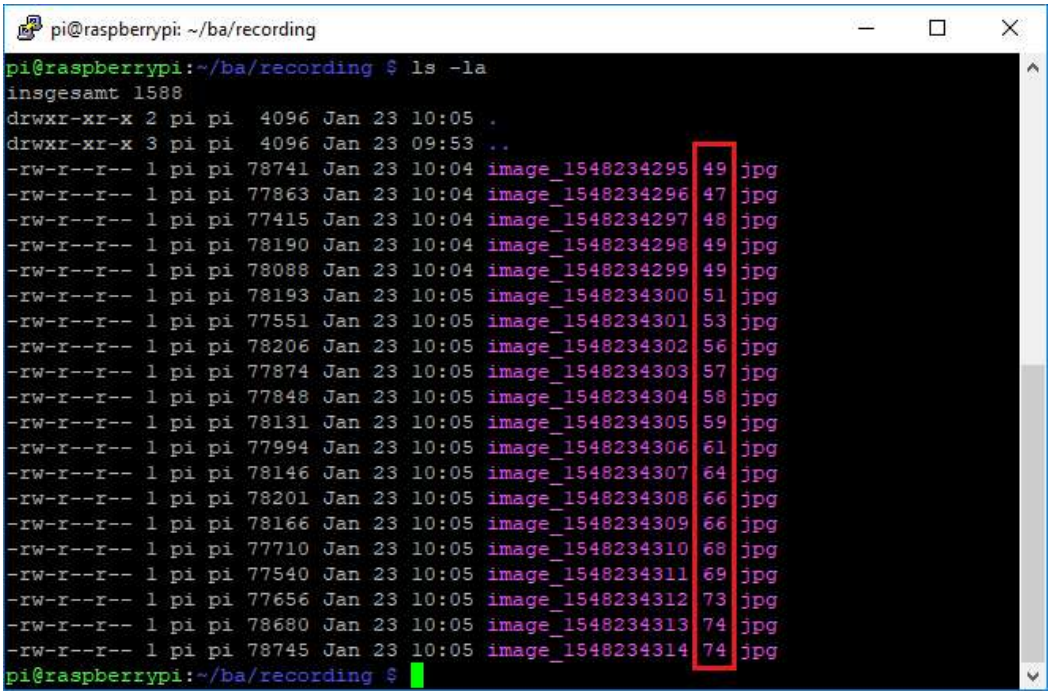

Abbildung 6: Aufzeichnung von Bildern mit der Kamera des Raspberry Pi

Um sicherzustellen, dass die „DogCam“ 1 Bild/ Sekunde aufzeichnet (Tabelle 3, S5), wurden im Python-Skript (Code 1) die Zeilen 25 und 26 einkommentiert sowie die Zeile 22 auskommentiert. So wurde an den Namen des aufgezeichneten Bildes der jeweils aktuelle Zeitstempel angehängt. Dadurch konnte verifiziert werden, dass bis auf eine Abweichung von ca. 6ms jede Sekunde ein Bild aufgezeichnet wird (siehe Abbildung 6). Im Frontend diente die in das Bild eingefügte Zeitangabe (Code 1, ZZ. 15 - 16) zur Verifikation der Anforderung S6 (siehe Tabelle 3).

4.2 Verbesserungen

Ein Punkt, der im Rahmen der prototypischen Implementierung der „DogCam“ (bewusst) nicht berücksichtigt wurde, ist die Sicherheit des Systems. In der durchgeführten Implementierung erfolgt die Kommunikation der Clients mit dem Apache 2-Webserver unverschlüsselt und ohne

Authentifizierung. Somit kann jeder, der die öffentliche IP des Internetanschlusses, hinter dem die „DogCam“ betrieben wird, kennt, jederzeit auf die aufgenommenen Bilder zugreifen.

Dieser erhebliche Mangel muss bei der Weiterentwicklung der „DogCam“ berücksichtigt werden. So kann zunächst eine statische Authentifizierung mittels htaccess[7] erfolgen. Für ein flexibleres Usermanagement wird ein umfangreicheres Konzept benötigt, welche eine eingeschränkte Nutzerregistrierung und -authentifizierung erlaubt. Alternativ ist auch ein Zugriff über VPN denkbar, falls der eingesetzte Router dies unterstützt, was bei den aktuellen FRITZ!Box-Modellen der Fall ist[8].

Weiterhin sollte die Kommunikation der Clients mit der „DogCam“ über das https-Protokoll, also verschlüsselt, laufen.

4.3 Erweiterungen

Die im Rahmen der vorliegenden Arbeit erfolgten prototypischen Implementierung der „DogCam“ kann in verschiedene Richtungen erweitert werden.

4.3.1 Sicherheit

Wie im vorangehenden Abschnitt 4.2 erläutert, kann und sollte ein Sicherheitskonzept implementiert werden.

4.3.2 Hardware

Anstatt der verwendeten Kameramoduls kann das NoIR Kamera-Modul eingesetzt werden, welches aufgrund des fehlenden Infrarotfilters bessere Bilder bei schwacher Beleuchtung liefert.

Weiterhin ist der Einsatz mehrerer Raspberry Pis mit Kameramodul denkbar. Über das Webfrontend muss dann die Auswahl einer Kamera oder die gleichzeitige Darstellung der Bilder mehrerer oder aller Kameras möglich sein. Hierbei stellt allerdings die verfügbare Bandbreite der Internetanbindung einen limitierenden Faktor dar.

Das Konzept der „DogCam“ kann beim Einsatz eines Motion Sensor Moduls so angepasst werden, dass ein Videostream bei erkannter Bewegung den Clients zur Verfügung gestellt wird und andernfalls ein statisches Bild.

---

[7] https://wiki.selfhtml.org/wiki/Webserver/htaccess; Zugriff am 28. Januar 2019

[8] https://avm.de/service/vpn/uebersicht/; Zugriff am 28. Januar 2019

Wird die „DogCam“ funktional erweitert (siehe Abschnitt 4.3.3), so kann die Performance des Systems durch den Einsatz des zu dem Zeitpunkt jüngsten Raspberry Pi Modells gesteigert werden.

4.3.3 Software

Anstatt den Clients Einzelbilder zur Verfügung zu stellen, kann auch ein Video gestreamt werden. Hierfür kann die Bibliothek motion (siehe auch Abschnitt 2.2) eingesetzt werden.

Zur Integration der „DogCam“ z.B. in ein System zur Home Automation kann anstatt der in der vorliegenden Arbeit genutzten prototypischen Implementierung Node-RED[9] zum Einsatz kommen. Ein dafür notwendiger Node-RED-Flow node-red-contrib-camerapi[10] ist aktuell in Version 0.0.38 verfügbar. Eine Anpassung des vorgenannten Flows kann allerdings notwendig sein.

Für den Zugriff auf die „DogCam“ kann auch eine App für Smartphones und Tablets, beispielsweise unter Verwendung des Ionic-Frameworks[11], entwickelt werden.

[9] https://nodered.org/; Zugriff am 28. Januar 2019

[10] https://flows.nodered.org/node/node-red-contrib-camerapi; Zugriff am 28. Januar 2019

[11] https://ionicframework.com/; Zugriff am 28. Januar 2019

5 Fazit und Ausblick

Die vorliegende Arbeit hat gezeigt, dass die Implementierung einer „DogCam“ relativ einfach umgesetzt werden kann. Das erarbeitete Konzept konnte verifiziert werden. Weiterhin wurden alle Anforderungen erfüllt. Da es sich bei der Implementierung um einen Prototyp handelte, konnte der Aspekt der Sicherheit zunächst vernachlässigt werden. Für die Weiterentwicklung der „DogCam“ muss aber eben vorgenannter Aspekt bevorzugt beachtet werden (siehe auch Abschnitt 4.3.1).

Nach der Konzeption und Implementierung eines Sicherheitskonzepts, welche nicht Bestandteil der vorliegenden Arbeit ist, sollen die Endlosschleifen im Python-Skript (siehe Code 1, ZZ. 8 - 28) und jQuery-Skript (Code 2, ZZ. 24 - 30) entfernt werden. Ersetzt werden könnten diese z.B. durch den Einsatz eines MQTT-Servers (Message Queuing Telemetry Transport), der das neu aufgenommene Bild Base64-kodiert an den Client sendet.

Literaturverzeichnis

[1] yeebase media GmbH, „t3n.de: Ranking der meistverkauften Computer: Raspberry Pi überholt C64," 20. März 2017. [Online]. Available: https://t3n.de/news/computer-raspberry-pi-c64-806761/. [Zugriff am 28. Januar 2019].

[2] Raspberry Pi Foundation, „Raspberry Pi Foundation: About Us," [Online]. Available: https://www.raspberrypi.org/about/. [Zugriff am 28. Januar 2019].

[3] Avnet, „hackster.io: Raspberry Pi's community hub," [Online]. Available: https://www.hackster.io/raspberry-pi. [Zugriff am 28. Januar 2019].

[4] Adafruit, „adafruit.com: Raspberry Pi Model A 256MB RAM ID: 1344 - $29.95 : Adafruit Industries, Unique & fun DIY electronics and kits," [Online]. Available: https://www.adafruit.com/product/1344. [Zugriff am 28. Januar 2019].

[5] Adafruit, „adafruit.com: Raspberry Pi Model B 512MB RAM ID: 998 - $39.95 : Adafruit Industries, Unique & fun DIY electronics and kits," [Online]. Available: https://www.adafruit.com/product/998. [Zugriff am 28. Januar 2019].

[6] Raspberry Pi Foundation, „Raspberry Pi 2 Model B," [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/. [Zugriff am 28. Januar 2019].

[7] Raspberry Pi Foundation, „Raspberry Pi 3 Model B," [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Zugriff am 28. Januar 2019].

[8] M. Stahl, „Raspberry Pi: Überwachungskamera und Webserver bauen - so geht's - CHIP," CHIP Digital GmbH, 07. August 2018. [Online]. Available: https://praxistipps.chip.de/raspberry-pi-ueberwachungskamera-und-webserver-bauen-so-gehts_104442. [Zugriff am 28. Januar 2019].

[9] Raspberry Pi Tutorials (Felix Stern), „Raspberry Pi: Überwachungskamera Livestream einrichten," [Online]. Available: https://tutorials-raspberrypi.de/raspberry-pi-ueberwachungskamera-livestream-einrichten/. [Zugriff am 28. Januar 2019].

[10] J. Donauer, „Mit dem Raspberry Pi die Wohnung überwachen - PC WELT," IDG Tech Media GmbH / PC-WELT, 27. Oktober 2018. [Online]. Available:

https://www.pcwelt.de/ratgeber/Mit_dem_Raspberry_Pi_die_Wohnung_ueberwachen-Sicherheit-8638548.html. [Zugriff am 28. Januar 2019].

[11] M. Williams, „Raspberry Pi Security System With Motion Detection / Camera / hackster.io," Avnet, 19. April 2016. [Online]. Available: https://www.hackster.io/FutureSharks/raspberry-pi-security-system-with-motion-detection-camera-bed172. [Zugriff am 28. Januar 2019].

[12] K. Walker, „Raspberry Pi Image Recognition with Alexa Voice / hackster.io," Avnet, 30. Mai 2018. [Online]. Available: https://www.hackster.io/ken-walker/raspberry-pi-image-recognition-with-alexa-voice-6b7a01. [Zugriff am 28. Januar 2019].

[13] Anton, „Raspberry Pi Facial Recognition / hackster.io," Avnet, 15. März 2017. [Online]. Available: https://www.hackster.io/gr1m/raspberry-pi-facial-recognition-16e34e. [Zugriff am 28. Januar 2019].