DHBW Karlsruhe

TINF12B5

Studienarbeit

Entwicklung eines Komplettsystems zur Überwachung und Beleuchtung von Innen- und Außenbereichen mit Raspberry Pi und iOS App

---

Autor:
Timo Höting
2185611

Betreuer:
Stefan Lehmann

Erklärung

Gemäß §5(3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 22. September 2011.

Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

---------- ----------

Ort, Datum Unterschrift

Inhaltsverzeichnis

1 Einleitung

Es soll ein Komplettsystem entwickelt werden, dass sowohl die Überwachung als auch die Steuerung der Beleuchtung von Innen- und Außenbereichen ermöglicht. Das System soll nach Entwicklung universell einsetzbar und leicht konfigurierbar sein.

1.1 Vorwort

1.2 Projektbeschreibung

Für die Beleuchtung sollen adressierbare LED-Pixel eingesetzt werden, welche möglichst leicht in ihrer Anzahl variiert werden können. Es müssen passende Bauteile und Produkte evaluiert und getestet werden. Diese müssen vom Raspberry Pi ansteuerbar sein. Die Überwachung findet über eine Kamera statt. Ob diese direkt am Raspberry Pi angeschlossen wird oder sich nur im selben Netzwerk befindet wird im Laufe dieses Projekts erarbeitet. Für die Erkennung von Aktivitäten werden Bewegungsmelder eingesetzt. Gesteuert wird das System über einen Raspberry Pi. Von diesem aus werden die LEDs angesteuert, die Sensorsignale ausgewertet und die Befehle der App empfangen. Um dem User eine einfach Ansteuerung zu ermöglichen wird eine iOS App implementiert. Die hierfür genutzten Sprachen sind Swift und Objective-C. Die Serverfunktionalitäten werden in Python implementiert. Es gibt drei verschiedene Modi in denen sich das System befinden kann:

- Beleuchtung wird durch Bewegungsmelder ausgelöst (Reaktion darauf kann vom User definiert werden)
- Beleuchtung wird manuell vom Benutzer über App gesteuert (Color Chooser, Bereichsauswahl, Leuchteffekte)
- Bewegungsmelder als Alarmanlage, beim Auslösen wird der Benutzer benachrichtigt und Bild der Kamera als Notifcation auf dem Smartphone angezeigt

1.3 Teilprojekte

- LED-Pixel evaluieren / ansteuern
- Implementierung der Ansteuerung / des Protokolls (mit den drei verschiedenen Modi)
- Implementierung der iOS App
- Vollständige praktische Umsetzung an einem Beispielobjekt

2 Hauptteil

2.1 LED-Pixel

2.1.1 Bewertungskriterien

Die Beleuchtung soll durch einzelne LED-Pixel stattfinden. Ein Pixel bedeutet ein Chip auf dem sowohl die LED und der nötige Treiber sitzt. Für die Evaluierung werden folgende Kriterien gewählt:

- RGB-Farbraum
  Die LED muss den gesamten RGB-Farbraum darstellen können.
  Gewichtung: 5, KO-Kriterium

- Ansteuerung
  Da der Raspberry Pi an einigen seiner Pins Pulsweitenmodulation (PWM) bietet, sollten die LED-Pixel ohne extra Hardware ansteuerbar sein. Eine extra Stromversorgung ist aber bei größerer Anzahl an LEDs unabdingbar.
  Gewichtung: 10

- Framework
  Hier wird bewertet ob der jeweilige Hersteller ein fertiges Framework zu seinen Produkten anbietet.
  Gewichtung: 10

- Kosten
  Es werden nur die reinen Produktkosten, also ohne Versand und Zoll, bewertet.
  Gewichtung: 5

- Extras
  An dieser Stelle können mögliche Extras eines Herstellers einfließen.
  Gewichtung: 5

2.1.2 Evaluierung

- Adafruit, Neopixel
  https://www.adafruit.com/neopixel
  LED-Pixel in unzähligen Ausführungen.
  Sitz der Firma in Tampa, Florida, USA
  RGB: Chip ist der WS2801, http://www.adafruit.com/datasheets/WS2801.pdf ->
  Hat volle Abdeckung des RGB-Farbraums
  Ansteuerung: Findet über PWM-Pin des Raspberry Pi statt.
  Framework: Framework von Adafruit, welches eine sehr leichte Ansteuerung ermöglichen soll.
  Kosten: 4 LEDs 7$, 25 LEDs zusammen 39$, durch Lieferung aus USA sehr hohe

Versandkosten (50$)
Extras: Händler bietet verschiedene Formen und fertige Ketten an.

- LED-Emotion GMBH, LED Streifen
  http://www.led-emotion.de/de/LED-Streifen-Set.html
  LED-Streifen, keine Einzelpixel, nur mit Controller, keine API
  RGB: Voller RGB-Farbraum
  Ansteuerung: Nur mit Controller
  Framework: Keine öffentliche Api, möglicherweise mit Raspberry Pi ansteuerbar
  Kosten: 30 LEDs mit Netzteil 79€
  Extras: keine

- DMX4ALL GmbH, MagiarLED Solutions
  http://www.dmx4all.de/magiar.html
  Spezialisiert auf DMX-Ansteuerung, keine öffentliche API
  RGB: VOller RGB-Farbraum
  Ansteuerung: Wird über DMX-Controller angesteuert, dieser setzt die Signale um. Vermutlich wird auch der WS2801 Chip verwendet.
  Framework: DMX-Ansteurung über DMX-Controller
  Kosten: Streifen mit 72 LEDs = 99€
  Extras: viele verschiedene Varianten

- TinkerForge, RGB LED-Pixel
  https://www.tinkerforge.com/de/shop/accessories/leds.html
  Scheinen die gleichen wie von Adafruit zu sein, allerdings werden hauptsächlich Controller im Shop angeboten
  RGB: Chip ist der WS2801, http://www.adafruit.com/datasheets/WS2801.pdf -> Hat volle Abdeckung des RGB-Farbraums
  Ansteuerung: Nach Anfrage an den Anbieter sollen die LEDs baugleich zu denen von Adafruit sein.
  Framework: keins
  Kosten: 50 LEDs = 59€
  Extras: Lieferung aus Deutschland

Fazit: In der Evaluierung schneiden die Produkte von Adafruit und TinkerForge am besten ab. Für eine erste Teststellung werden die einzelnen LED-Pixel von Adafruit aus den USA bestellt (Neopixel). An diesen soll vor allem die Ansteuerung getestet werden. Falls diese sich bewähren wird für den endgültigen Aufbau auf die LED-Ketten von Tinkerforge zurück gegriffen.

2.1.3 Teststellung

Für einen ersten Test wurde das in XXX ausgewählte Produkt als einzelne Pixel bestellt. Der Hersteller Adafruit bietet hier 4er-Packungen an. Diese können leicht in eigene Schaltungen eingelötet oder auf Experimentier-Boards gesteckt werden. Bei geringer Anzahl LEDs reicht die 5V-Stromversorgung des Raspberry Pi aus.

Technische Daten Neopixel:

- Maße: 10.2mm x 12.7mm x 2.5mm
- Protokollgeschwindigkeit: 800 KHz
- Spannung: 5-9VDC (bei 3,5V gedimmte Helligkeit)
- Strom: 18,5mA / LED, 55mA / Pixel

Framework:

- RPI_WS281X (https://github.com/jgarff/rpi_ws281x)
- Sprache: Python
- Entwickelt für Raspberry Pi
- Vorraussetzung: Python 2.7

Ablauf des Tests:

- **Aufbau der Schaltung**
  An die einzelnen LED-Pixel wurden Stecker angelötet, damit sie auf das Experimentierboard aufgesteckt werden können. Dann wird die Schaltung nach folgendem Schaltbild verbunden. Wichtig ist, dass beim Raspberry Pi nur Pins verwendet werden können, welche PWM bieten.

- **Installation des Frameworks**

```
1 wget https://github.com/tdicola/rpi_ws281x/raw/master/python/dist/
      rpi_ws281x-1.0.0-py2.7-linux-armv6l.egg
2 sudo easy_install rpi_ws281x-1.0.0-py2.7-linux-armv6l.egg
```

Listing 2.1: Installation Framework ws281x

- **Testcode**

```
1  from neopixel import *
2
3  LED_COUNT = 4 # Number of LED pixels.
4  LED_PIN = 18 # GPIO pin connected to the pixels (must support PWM!).
5  LED_FREQ_HZ = 800000 # LED signal frequency in hertz (usually 800khz)
6  LED_DMA = 5 # DMA channel to use for generating signal (try 5)
7  LED_INVERT = False # True to invert the signal (when using NPN)
8
9  strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ,
       LED_DMA, LED_INVERT)
10
11 strip.begin()
12 strip.setPixelColor(0, Color(255, 255, 255))
13 strip.setPixelColor(1, Color(255, 255, 255))
14 strip.setPixelColor(2, Color(255, 255, 255))
```

```
15 strip.setPixelColor(3, Color(255, 255, 255))
16 strip.show()
```

Listing 2.2: Testcode zur Ansteuerung der LEDs

Fazit Die einzelnen Pixel sind sehr leicht anzusteuern, unterstützen auch das automatische Abschalten nach einer bestimmten Zeit und haben eine sehr hohe Leuchtkraft. Die Evaluierung hat zu einer guten Produktwahl geführt. In der Endgültigen Projektstellung werden keine einzelnen LEDs eingesetzt, sondern eine fertige Kette des Herstellers.

2.2 Bewegungssensor

In einem der Modi soll die Beleuchtung durch den Bewegunsmelder ausgelöst werden. Hierfür sind zuverlässige und weitreichende Beweungssensoren notwendig.

2.2.1 Bewertungskriterien

- **Ansteuerung**
  Die Anbindung an den Raspberry Pi soll möglichst leicht realisierbar sein. Wünschenswert ist, dass der Sensor einfach ein High-Signal bei Bewegungserkennung ausgibt.
  Gewichtung: 5, KO-Kriterium

- **Reichweite**
  Die Reichweite oder Sensivität des Sensors soll ausreichend und regelbar sein.
  Gewichtung: 3

- **Kosten**
  Es werden nur die reinen Produktkosten, also ohne Versand und Zoll, bewertet.
  Gewichtung: 1

- **Extras**
  An dieser Stelle können mögliche Extras eines Herstellers einfließen.
  Gewichtung: 3

2.2.2 Evaluierung

- **PIR (MOTION) Sensor, Adafruit**
  Link: http://www.adafruit.com/product/189
  Ansteuerung: Gibt High-Signal an einem Pin aus.
  Reichweite: 7m, 120 Grad
  Kosten: 9,95$ + Versand aus USA
  Extras: Kabel inklusive

- **PIR Infrared Motion Sensor (HC-SR501)**
  Link: https://www.modmypi.com/pir-motion-sensor
  Ansteuerung: Gibt High-Signal an einem Pin aus.
  Reichweite: 5-7m, 100 Grad

Kosten: 2,99$ + Versand aus UK
Extras: keine

- **Infrarot PIR Bewegung Sensor Detektor Modul**
  Link: http://www.amazon.de/Pyroelectrische-Infrarot-Bewegung-Sensor-Detektor/dp/B008AESD
  Ansteuerung: Gibt High-Signal an einem Pin aus.
  Reichweite: 7m, 100 Grad
  Kosten: 5 Stück = 7,66€
  Extras: keine

2.2.3 Fazit

Die meisten Infarot-Bewegungssensoren sind von der Bauweise nahezu identisch. Die Unterschiede liegen meist nur in der Empfindlichkeit. Da die Reichweite in diesem Fall nicht von großer Bedeutsamkeit ist, kann eigentlich jedes der Produkte bestellt werden. Auf Ebay und Amazon ist die Anzahl angebotener Sensoren nahezu unbegrenzt, es wurde für die Teststellung also die oben evaluierte Variante von Amazon bestellt.

2.2.4 Teststellung

Der in Punkt X.X.X gewählte Bewegungssensor wurde beim Hersteller bestellt. In der Teststellung reicht die Stromversorgung des Raspberry Pi.

Technische Daten Sensor:

- Die Empfindlichkeit und Haltezeit kann eingestellt werden
- Reichweite: ca. 7m
- Winkel: 100 Grad
- Spannung: DC 4,5V- 20V
- Strom: $<$ 50uA
- Ausgansspannung: High 3V / Low 0V
- Größe: ca. 32mm x 24mm

Ablauf des Tests:

- **Aufbau der Schaltung**
  Der Sensor wird in der Teststellung direkt vom Raspberry Pi mit Strom versorgt. Für die Datenleitung kann jeder beliebige Pin gewählt werden.
- **Testcode**
  Um eine Änderung am Datenpin festzustellen werden zwei Variable angelegt: current_status und previous_status. Das Programm wird in einer Dauerschleife geschickt, in der bei jedem Durchlauf die beiden Status überprüft. Wenn der neue Status (current_status) High ist und das vorherige Signal (previous_state) Low, dann wird eine Bewegung erkannt. Der Code wird mittels Kommentare erklärt.

```
1  import RPi.GPIO as GPIO
2  import time
3
4  GPIO.setmode(GPIO.BCM)
5
6  # Pin definieren
7  MOTION_PIN1 = 7
8
9  # Diese als Input definieren
10 GPIO.setup(MOTION_PIN1,GPIO.IN)
11
12 # Status definieren um verschiedene nderungen zu erkennen
13 Current_State = 0
14 Previous_State = 0
15
16 try:
17         # Loop zur Erkennung einer Bewegung
18         # Sensor erkennt Bewegung -> Signal = High
19         # Wartet 3 Sekunden und setzt Signal = Low
20         while True :
21                 Current_State = GPIO.input(MOTION_PIN1)
22                 if Current_State == 1 and Previous_State == 0:
23                         print "Motion detected!"
24                         Previous_State=1
25                 elif Current_State == 0 and Previous_State == 1:
26                         print "Ready"
27                         Previous_State=0
28                 time.sleep(0.01)
29
30 except KeyboardInterrupt:
31         print "Quit"
32         GPIO.cleanup()
```

Listing 2.3: Testcode zur Bewegungserkennung mit Sensor

Bei der Endversion des Systems sollen mehrere Beweungssensoren integriert werden. Bei Auslösen des ersten Sensors sollen die LEDs angeschaltet werden und nach auslösen eines weiteren Sensors wieder ausgeschaltet werden.

- **Auswertung**
  Das High-Signal des Sensors lässt sich mit dem Raspberry Pi sehr leicht auswerten. Auch die Auswertung von mehreren Sensoren stellt kein Problem da. Das Ergebnis der Evaluierung konnte in dieser Tststellung bestätigt werden.

2.3 Python-Server und Protokoll

2.3.1 Protokoll

Um die LEDs später von einer App aus ansprechen zu können, soll ein auf Strings basierendes Protokoll implementiert werden. Dieses wird in TCP-Paketen übertragen. Hierfür muss als erstes festgelegt werden, welche Informationen übertragen werden sollen:

- Authentifizierung
  Übertragung eines Passworts. Dieses ist als Hashwert im System gespeichert und kann so überprüft werden. Es wird der SHA-224-Algorithmus eingesetzt.
- Control
  Unterscheidung zwischen:
  - X00: Alle LEDs ausschalten
  - X01: Eine LED anschalten
  - X02: LED-Bereich anschalten
  - X03: Effekt für eine LED
  - X04: Effekt für LED-Bereich
  - X05: Effektcode

  Abhängig von diesem Feld werden die nachfolgenden Werte behandelt.
- LED-Nummer
  Falls nur eine LED angesprochen werden soll (Control = X00), so wird hier die Nummer angegeben. Ob sie im gültigen Range liegt wird intern überprüft.
- Bereich Start
  Wenn mehrere LEDs gesteuert werden sollen (Control = X01), so wird hier der Beginn des Bereichs angegeben.
- Bereich Ende
  Und hier das Ende des Bereichs.
- Rot
  Farbwert Rot 0-255
- Grün
  Farbwert Grün 0-255
- Blau
  Farbwert Blau 0-255
- Effekt
  Es können verschiedene LEDs mit Effekten belegt werden, wie zum Beispiel Aufblitzen oder zeitgesteuertes Ausschalten.
- Effektcode
  Hinterlegte, fest programmierte Effekte, zum Beispiel alle LEDs anschalten in weis mit höchster Leuchstärke.

- Hash
  Überprüfung ob die Übertragung erfolgreich war, mittels eines Hashwertes. Es wird der SHA-224-Algorithmus eingesetzt.

Übertragungsbeispiel:
Protokoll:

```
auth:control:ledNo:rangeStart:rangeEnd:red:green:blue:effect:effectcode:hash
pass:X01:0:0:49:255:255:255:0:0:xx
```

Listing 2.4: Beispielübertragung des Protokolls

Dies wrde die LEDs 0 bis 49 einschalten (Farbe weis 255,255,255). Anstelle der XX würde der Hashwert der gesamten Übertragung gesendet.

2.3.2 Framework

Twisted: https://twistedmatrix.com
Es wird das Twisted Matrix Framework eingesetzt. Twisted ist eine in Python geschriebene event-getriebene Netzwerkengine. Die meisten gängigen Protokolle wie TCP, IMAP, SSHv3 und viele mehr werden unterstützt. Somit bietet Twisted die ideale Möglichkeit einen eigenen simplen Server zu implementieren.

Event-Getrieben (event-based): Die Serveranwendung befindet sich in einer Schleife und wartet auf ein Event. Dieses Event ist in diesem Fall der Connect eines Clients zum Server. Für jeden Connect wird eine neue Instanz angelegt, in welcher empfangene Daten bearbeitet werden können. Die Daten werden als String ausgewertet, somit wird ein string-basiertes Protokoll implementiert. [I.]

2.3.3 Testcode

```
1  #!/usr/bin/env python
2  # Copyright (c) Twisted Matrix Laboratories.
3  # See LICENSE for details.
4
5  from twisted.internet.protocol import Protocol, Factory
6  from twisted.internet import reactor
7
8  ### Protocol Implementation
9
10 # This is just about the simplest possible protocol
11 class Echo(Protocol):
12         def dataReceived(self, data):
13                 self.transport.write(data)
14
15
16         def main():
17                 f = Factory()
18                 f.protocol = Echo
19                 reactor.listenTCP(8000, f)
```

```
20                  reactor.run()
21
22 if __name__ == '__main__':
23 main()
```

Listing 2.5: Testcode Echoserver mit Twisted Framework

//TODO ERKLÄRUNG

2.3.4 Implementierung

Im Folgenden wird die Implementierung des Servers näher erläutert.

// TODO CODE des Servers

2.3.5 Hashfunktion

Es wird zu zweierlei Zwecken eine Hashfunktion eingesetzt. Zum einen um die Korrektheit der Übertragung zu überprüfen und zum Anderen um ein Passwort zur Authentifizierung verwenden zu können. Dieses wird als Wort übertragen, auf dem Server aber nur als Hash-Wert abgespeichert. Falls es also jemand schafft die Konfirgurationsdatei abzugreifen, so ist der Passworthash nichts wert.

2.4 Verschlüsselung

2.4.1 SSL vs. TLS

SSL (Secure Sockets Layer) und TLS (Transport Layer Security) sind Protokolle, die Verschlüsselung und Authentifizierung zwischen zwei Kommunikationspartnern bieten. Die beiden Begriffe SSL und TLS werden umgangssprachlich oft als zwei verschiedene Techniken dargestellt, obwohl TLS nur eine Weiterentwicklung von SSL ist. SSL v3 ist die Basis von TLS 1.0.
Aufgrund des Alters und einiger Sicherheitslücken wird SSL als unsicher angesehen und soll nicht mehr verwendet werden. Die aktuellste gefundene Lücke ist POODLE, welche das Auslesen von Informationen aus einer verschlüsselten Übertragung erlaubt. Die Weiterentwicklungen TLS 1.1 und 1.2 sind deutlich sicherer und beheben einige Sicherheitslücken. So schützt die richtige Implementierung von TLS 1.2 auch vor den BEAST Angriffsmethoden.
Eine Variante von TLS ist das sogenannte STARTTLS, bei dem zuerst ein unsicheres 'hello' an den Server gesendet wird. Falls im Anschluss eine Verbindung erfolgreich Zustande kommt, wird zur sicheren Übertragung gewechselt.
Wenn ein Server implementiert wird, so muss er alle Techniken unterstützen, beim Client kann der Entwickler selbst entscheiden. Ein Entwickler sollte immer die höchst mögliche Verschlüsselungstechnik einsetzen.

2.4.2 Vor- und Nachteile TLS

Da TLS auf der Transportschicht aufsetzt kann jedes höhere Protokoll darüber übertragen werden, somit ist die Verschlüsselung unabhängig von der genutzten Anwendung.
Der größte Nachteil besteht darin, dass der Verbindungsaufbau serverseitig sehr rechenintensiv ist. Die Verschlüsselung selbst nimmt, abhängig vom Algorithmus, nur noch wenige Rechenleistung in Anspruch.

2.4.3 TLS Handshake

1. Client Hello
Übertragung von Verschlüsselungsinformationen vom Client an den Server, wie TLS Version oder Verschlüsselungsmöglichkeiten
2. Server Hello
Server sendet seine Informationen und legt Verschlüsselung fest.
3. Server Key Exchange
Server sendet seine Identität in Form seines Zertifikats.
4. Client Key Exchange
Client legt seinen Pre-Shared-Key fest und überträgt ihn verschlüsselt mit dem public Key des Servers.
5. Change Cipher Spec
Aus dem PSK wird ein Master-Secret generiert, mit welchem die folgenden Übertragung abgesichert wird.
6. Application Data
Übertragung der Daten.

2.4.4 Zertifikat und Key

Auf dem Raspberry Pi ist OpenSSL in der neuesten Version installiert. Es wird ein selbstsigniertes Zertifikat im 2048 Bit Key erzeugt.
1. Private Key erzeugen
openssl genrsa -des3 -out server.key 2048
2. Certificate Signing Request
openssl req -new -key server.key -out server.csr
3. Self Signed Certificate
Bei einem öffentlichen Server sollte das Zertifikat bei einer CA (Certificate Authority) signiert werden.
openssl x509 -req -days 1865 -in server.csr -signkey server.key -out server.crt

2.4.5 Beispielcode STARTTLS Server

An dieser Stelle ist der Beispielcode von Twisted am besten verständlich
(Quelle: https://twistedmatrix.com/documents/12.3.0/core/howto/ssl.html)

```
1 from OpenSSL import SSL
2 from twisted.internet import reactor, ssl
3 from twisted.internet.protocol import ServerFactory
```

```
4  from twisted.protocols.basic import LineReceiver
5
6  class TLSServer(LineReceiver):
7      def lineReceived(self, line):
8          print "received: " + line
9
10         if line == "STARTTLS":
11             print "-- Switching to TLS"
12             self.sendLine('READY')
13             ctx = ServerTLSContext(
14                 privateKeyFileName='keys/server.key',
15                 certificateFileName='keys/server.crt',
16                 )
17             self.transport.startTLS(ctx, self.factory)
18
19
20 class ServerTLSContext(ssl.DefaultOpenSSLContextFactory):
21     def __init__(self, *args, **kw):
22         kw['sslmethod'] = SSL.TLSv1_METHOD
23         ssl.DefaultOpenSSLContextFactory.__init__(self, *args, **kw)
24
25 if __name__ == '__main__':
26     factory = ServerFactory()
27     factory.protocol = TLSServer
28     reactor.listenTCP(8000, factory)
29     reactor.run()
```

Listing 2.6: Testcode Echoserver mit Twisted Framework

Es ist gut zu erkennen, dass die Übertragung nur ausgewertet wird, wenn das Stichwort SSTARTTLSäm Anfang der Übertragung enthalten ist. Daraufhin wird mit RREADY“geantwortet um dem Client zu signalisieren, dass jetzt der TLS Handshake begonnen werden kann. Im nächsten Schritt läd der Server sein Zertifikat und seinen Key.
In der Initmethode der Klasse ServerTLSContext können die Verschlüsselungsdetails festgelegt werden. Im obigen Beispiel wird hier zum Beospiel die TLS Version definiert.

2.4.6 Wireshark Trace

Im folgenden ist ein Trace eines TLS Handshakes zwischen einem Client und dem implementierten Server auf dem Raspberry Pi zu sehen. // TODO beschreibung

2.5 Kamera

2.5.1 PI-Kamera vs. Netzwerkkamera

Bei Auslösen des System im Überwachungsmodus soll ein aktuelles Bild der Überwachungskamera an das jeweilige Sartphone gepusht werden. Es gibt zwei mögliche Kame-

ratechniken, entweder eine direkt an den Raspberry Pi Angeschlossene oder eine, die im Netzwerk erreichbar ist.

Raspberry Pi Cam Die Kameras für den Raspberry Pi können direkt an das Gerät angeschlossen werden. Meistens werden sie direkt über die GPIO Pins verbunden. Der Vorteil dieser Kameras ist, dass sie keine externe Stromversorgung benötigen und durch viele verschiedene Frameworks leicht anpassbar und verwaltbar sind. Der große Nachteil ist allerdings, dass die Kamera an dem Raspberry Pi angeschlossen werden muss, auf welchem auch der Server läuft. Da dieser aber möglich wettergeschützt (im Außenbereich) oder unauffällig (im Innenbereich) angebracht ist, lässt sich von diesen Positionen kaum eine effektive Überwachung realisieren.
Als Beispiel wäre ein von der Raspberry Pi Foundation empfohlene Kamera zu nennen: //TODO Beispielkamera

Netzwerkkamera Eine Netzwerkkamera oder auch IP-Kamera genannt befindet sich im Netzwerk und kann über eine Website oder App eingesehen und gesteuert werden. Der Vorteil ist, dass sie sich irgendwo befinden kann, solange sie im selben Netzwerk ist. Somit kann zum Beispiel eine wetterfeste Kamera im Außenbereich angebracht werden und der Server kann sich im geschützten Innenbereich befinden.
Der Nachteil besteht bei IP-Kameras darin, dass es keine einheitliche API zum Abgreifen des Videomaterials gibt. Eine mögliche Lösung wäre das Laden der HTML Seite über einen HTTP-Request und darauffolgend das Ausfiltern des Bildmaterials. Über diese Variante kann aber kein Video sondern nur temporäre Bilder geladen werden. Dies würde aber für eine Notification auf dem Smartphone ausreichen.

Für dieses Projekt wird eine IP-Kamera aufgrund von oben genannten Vorteilen verwendet.

2.5.2 Ansteuerung

Testcode HTTP-Request

Implementierung

2.6 Anwendungsstruktur

2.6.1 Klassen und ihre Funktionen

Die Klasse 'Center' stellt die zentrale Stelle in der Anwendung dar, an der alle Informationen zusammen laufen und verwaltet werden. In der Klasse 'Sensor' werden die einzelnen Bewegungssensoren überwacht. Falls eine Bewegung detektiert wird, so werden in 'Center' die notwendigen Methoden aufgerufen um die LEDs an- oder auszuschalten.
Die Klasse 'LED_Control' verwaltet die eingerichteten LEDs und steuert diese. Hier werden auch die möglichen Effekte gesteuert. Die Methoden in dieser Klasse werden aus der Klasse 'Center' aufgerufen. Ein Zugriff in die andere Richtung ist nicht möglich.

Alle Serverfunktionaltitäten werden in der Klasse 'Server' bereitgestellt. Hier werden die Daten von Übertragungen empfangen und ausgewertet. Die Prüfung der Korrektheit der

einzelnen Protokollbestandteile findet ebenfalls hier statt. Wenn alle Überprüfungen erfolgreich sind, werden die Befehle an 'Center' weiter gegeben und dort ausgeführt. Wenn eine Antwort notwendig ist, zum Beispiel eine Statusabfrage der App an den Server, so werden die Informationen von 'Center' gesammelt und dann von 'Server' verpackt und in korrektem Format an das Mobilgerät gesendet.

Informationen die für den Betrieb des Systems notwendig sind, werden in der 'Config.ini' gespeichert und können nur über die Klasse 'ConfigReader' abgerufen werden. Es werden Informationen wie Anzahl der LEDs, Passworthash oder Adresse der Netzwerkkamera abgespeichert. Die Konfigurationsdatei wird beim Installationsvorgang erstellt.

Abrufen von Bildmaterial von der Netzwerkkamera findet ausschließlich über die Klasse 'Cam' statt. Die Klasse ruft die Informationen ab und filtert das Bildmaterial. Somit geben die Methoden der Klasse nur ein Bild zurück.

In der Klasse 'Status' kann ein Status des Gesamtsystems in der Kommandozeile ausgegeben werden und über 'Log' können Informationen im Logfile gespeichert werden.

Einzelne Rückgabetypen von Methoden, sowie die Initialisierung von allen Klassen kann mit 'UNIT_Test' getestet werden.

```
1  root@raspberrypi:/home/timo/Studienarbeit# python UNIT_Test.py
2  test_Center (__main__.TestSequenceFunctions) ... ok
3  test_Effects (__main__.TestSequenceFunctions) ... ok
4  test_LEDControl (__main__.TestSequenceFunctions) ... ok
5  test_Sensor (__main__.TestSequenceFunctions) ... ok
6  test_Server (__main__.TestSequenceFunctions) ... ok
7  test_Status (__main__.TestSequenceFunctions) ... ok
8  test_camAdress_MUST_FAIL (__main__.TestSequenceFunctions) ... FAIL
9  test_camAvaible (__main__.TestSequenceFunctions) ... ok
10 test_getHashPass (__main__.TestSequenceFunctions) ... ok
11 test_getMotionPin1 (__main__.TestSequenceFunctions) ... ok
12 test_getNumberOfLED (__main__.TestSequenceFunctions) ... ok
13
14 ======================================================================
15 FAIL: test_camAdress_MUST_FAIL (__main__.TestSequenceFunctions)
16 ----------------------------------------------------------------------
17 Traceback (most recent call last):
18   File "UNIT_Test.py", line 67, in test_camAdress_MUST_FAIL
19     self.assertEqual(resultTest, resultCorrect)
20 AssertionError: '192.168.2.205' != '123'
21
22 ----------------------------------------------------------------------
23 Ran 11 tests in 1.873s
```

Listing 2.7: Ausgabe der Klasse UNIT_Test

2.7 Konfiguration und Installation

2.7.1 Konfiguration

2.7.2 Installation

2.8 iOS App

2.8.1 Konzept

2.8.2 ...

2.8.3 ...

3 Praktische Umsetzung

4 Kostenaufstellung

5 Fazit

Abbildungsverzeichnis

Literaturverzeichnis

[I.] "SWR Info - Zahlen, Daten, Fakten über den SWR", http://www.swr.de/unternehmen/unternehmen/kennzahlen/kennzahlen-organisation/-/id=12213420/did=12302978/nid=12213420/eqq46v/index.html, 13.12.2013