Präsenzerkennung im Rahmen einer modularen Smarthome Umgebung

Projektarbeit

Sebastian Dohle, Lennart Heimbs, Kevin Holzschuh, Johannes Krug

Fakultät Elektrotechnik Feinwerktechnik Informationstechnik,
Technische Hochschule Nürnberg Georg Simon Ohm,
Wassertorstraße 10,
90489 Nürnberg, Germany
Betreuer: Prof.-Dr. Oliver Hofmann, B.-Eng. Martin Wimmer

In der Projektarbeit wird zunächst eine individuelle Smarthome Umgebung aufgesetzt. Innerhalb dieser Umgebung sollen diverse kommerzielle, sowie eigens Entwickelte Sensoren nutzbar sein. Die Kontrolle der Sensoren und Aktoren soll über eine Web-Oberfläche hochschulintern möglich sein.

Anhand der Sensoren soll anschließend ein System aus Sensoren zur Erkennung und Zählung von anwesenden Personen im Labor entwickelt werden, um einen Überblick über die Auslastung des Raumes zu erhalten. Dabei wird ein kommerzieller Sensor für die Präsenzerkennung mit dem im Rahmen der Projektarbeit entwickelten Erkennungssystem und einer Kamera verglichen.

Keywords: Smarthome, Raspberry Pi, Homematic, Homegear, Node-Red, Präsenzerkennung, ESP8266, MQTT

Inhaltsverzeichnis

1 Einleitung

Intelligente Heim Überwachung und Automatisierung gewinnt stetig an Bedeutung in einer zunehmend vernetzten Gesellschaft. Jüngste Fortschritte in der Mikrocomputertechnik, deren Kosten und in der Netzwerktechnik machen es immer erschwinglicher für Privatpersonen ihr eigenes Heim mit Sensoren zu Überwachen. Neben Bewegungssensoren für die Außenbeleuchtung oder Überwachungskameras gibt es heute eine Vielzahl an Möglichkeiten der Heim Überwachung und Automatisierung.
So kann man zum Beispiel steuerbare Rollläden zusammen mit Temperatur-, Lichtintensitäts- und weiteren Sensoren nutzen um diese automatisch hoch, beziehungsweise herunter zu fahren oder Lichter einschalten, sobald sich eine Person in einem dunklen Raum befindet.

Aus diesen Möglichkeiten der Steuerung und Datenerfassung durch Sensoren ist ein breiter Markt an Systemen entstanden, die Technik für das Smarthome bereitstellen. Diese Technik gliedert sich allgemein in drei Geräte Kategorien:

- Sensoren, die zur Erfassung von physikalischen Umweltdaten dienen,
- Aktoren, die mit der Umwelt interagieren und
- Steuerzentralen, auch Gateways genannt, welche gesammelte Daten darstellen und anhand dieser Daten Aktoren steuern.

Das Zusammenspiel dieser drei Geräte stellt die Intelligenz des Smarthome Systems dar.
Auf dem Markt wird die Anzahl solcher Systeme immer größer, was sich in einer besseren Auswahl für den Konsumenten auswirkt. Allerdings sind diese Systeme in der Regel in ihrem Aufbau geschlossen und benutzen ein eigenes Ökosystem. Sie funktionieren nur mit Geräten eines Herstellers. Beispielhaft für ein solches geschlossenes System ist Homematic von der Firma eQ-3. Der Hersteller ist einer der Marktführer im Bereich Smarthome in Europa und bietet mit Homematic und dem neuen System Homematic IP zwei umfangreiche Smarthome Systeme mit einer sehr großen Auswahl an Geräten an. Homematic Sensoren und Aktoren können jedoch nur mit der hauseigenem Smarthome Zentrale (Central Control Unit, CCU) betrieben werden.
Aufgrund von zum Teil deutlichen Preisunterschieden zwischen verschiedenen Herstellern ist es wünschenswert diese miteinander kombinieren zu können. Zusätzlich ist es möglich eigene Sensoren sehr preiswert zu entwickeln, die wiederum nicht mit den kommerziellen Sensoren kommunizieren können. Aus diesem Grund ist ein Ziel dieser Projektarbeit ein Smarthome System aufzusetzen, welches es möglich macht, eine Vielzahl verschiedener Produkte verschiedener Hersteller, sowie selbstgebaute Sensoren zu verwenden. Um die Funktionalität dieses Systems zu testen wird anschließend ein Homematic IP Sensor zur Präsenzerkennung mit eigens entwickelten Sensoren zur Anwesenheitserkennung von Personen im Labor und einer Kamera, die genutzt wird, um Personen zu zählen, verglichen.

2 Hardware der Smarthome Umgebung

Den Startpunkt des Projekts stellt der Blogeintrag “Homematic mit Node-Red über homegear” von Patrik Mayer [2] dar, in dem eine Smarthome Struktur basierend auf einem Raspberry Pi beschrieben wird. Im Folgendem wird das Aufsetzen der Gateway unter Berücksichtigung des Blogartikels beschrieben. Begonnen wird mit der Hardware, die für die Gateway benötigt wird und deren Konfiguration.

Als Basis für die Gateway wird der Einplatinencomputer *Raspberry Pi* aufgrund seiner guten Verfügbarkeit und seines günstigen Preises gewählt. Mit LAN, WLAN und diversen weiteren Schnittstellen wie UART und SPI und dank seiner geringen Größe ist dieser Computer sehr gut geeignet für den Einsatz als Smarthome Zentrale.

Für das Betriebssystem des Raspberry Pi wird Raspbian Stretch Lite gewählt. Dies ist ein für den Dauerbetrieb geeignetes Betriebssystem basierend auf der Linux-Distribution Debian Stretch. Es ist speziell für den Raspberry Pi konfiguriert und verzichtet auf eine Desktop Umgebung, wodurch der Speicher- und Rechenleistungsverbrauch minimiert wird. Die Installation des Betriebssystems erfolgt über die bereits auf dem Raspberry Pi vorinstallierte Installationsanwendung NOOBS, welche das Betriebssystem automatisch auf der SD-Karte des Raspberry Pi herunterlädt und installiert.

Um mit Sensoren von kommerziellen Herstellern kommunizieren zu können wird zusätzlich ein Funkmodul benötigt. Üblicherweise werden dabei die ISM-Bänder 433MHz und 868MHz verwendet. Da Homematic das 868MHz-Band zur Kommunikation nutzt, wird der Bausatz *HM-MOD-RPI-PCB* von dem Hersteller ELV verwendet. Das Funkmodul wird auf die GPIO-Pins des Raspberry Pis gesteckt und benutzt das UART-Protokoll, um mit dem Raspberry Pi zu kommunizieren. Mit dem Befehl `sudo raspi-config` kann UART unter dem Menüpunkt `5 Interfacing Options` und anschließend `P6 Serial` aktiviert werden. Ist UART einmal aktiviert, ist das Modul Einsatzfähig.

3 Netzwerkkonfiguration des Raspberry Pi

Die Netzwerkkonfiguration des Projektes birgt zwei Herausforderungen:

1. Erreichbarkeit der gewonnenen Daten, beziehungsweise Steuerung der Aktoren über das Hochschulnetzwerk
2. Kommunikation der selbstgebauten Sensoren mit der Gateway über WLAN

Die Erreichbarkeit der Daten wird sichergestellt, indem der Raspberry Pi dauerhaft über LAN an das Hochschulnetz angeschlossen ist. Zudem benötigt er eine bekannte statische IP-Adresse um auf die Daten zugreifen zu können. Die statische IP-Adresse wird eingestellt, indem die Datei `/etc/dhcpcd.conf` folgender Abschnitt für den LAN-Adapter eingefügt wird:

```
interface br0
#eth0
static ip_address=141.75.33.126/24
static routers=141.75.33.1
static domain_name_servers=141.75.40.250

# It is possible to fall back to a static IP if DHCP fails:
define static profile
profile static_br0
static ip_address=192.168.1.23/24
static routers=192.168.1.1
```

```
static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
interface br0
fallback static_br0
```

Um die später beschriebenen selbstgebauten Sensoren über WLAN mit dem Gateway kommunizieren zu lassen, wurde zunächst versucht, das Hochschulnetzwerk *Eduroam* zu werenden. Da dieses jedoch die Sicherheitskonfiguration WPA2 Enterprise nutzt und dieses auf dem benutzten Mikrocontroller *ESP8266* nicht unterstützt wird, muss ein alternatives WLAN-Netzwerk genutzt werden. Als Lösung wurde die Funktionalität des Raspberry Pis genutzt, als WLAN-Accesspoint zu dienen. Der Raspberry Pi, der per LAN mit dem Hochschulnetzwerk verbunden ist, spannt somit ein unabhängiges, lokales WLAN-Netzwerk auf, welches die WPA2-PSK Verschlüsselung benutzt, die auch von dem ESP8266 unterstützt wird.

3.1 Raspberry Pi als Accesspoint

Realisiert wird der Accesspiont durch das Softwarepaket *hostapd*, welches durch folgendes Kommando installiert wird:

```
sudo apt install hostapd
```

Danach wird hostapd für die Konfigurationsphase deaktiviert:

```
sudo systemctl stop hostapd
```

3.2 Raspberry Pi als Server

Da das Netzwerk als Server agieren soll, wird dem Raspberry Pi eine statische IP-Adresse zugewiesen. Dazu wird in der Datei `/etc/dhcpcd.conf` der WLAN-Adapter `wlan0` folgendermaßen konfiguriert:

```
interface wlan0
static ip_address=192.168.252.1/24
nohook wap_supplicant

define static profile
profile static_wlan0
static ip_address=192.168.1.23/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1

interface wlan0
fallback static_wlan0
```

Um die Änderungen zu übernehmen wird der dhcp daemon neu gestartet:

```
sudo service dhcpcd restart
```

3.3 DHCP Server Konfigurieren

Damit die Verbindung des ESP8266 mit dem Raspberry Pi unkompliziert abläuft wird ein DHCP Server eingerichtet. Dazu wird folgende Konfiguration in die Datei `/etc/dnsmasq.conf` geschrieben:

```
interface=wlan0
dhcp-range=192.168.252.2,192.168.252.20,255.255.255.0,24h
```

Somit werden IP-Adressen für die Microkontroller automatisch vergeben.
Anschließend startet man den DHCP Server erneut: `sudo systemctl reload dnsmasq` .

3.4 Accesspoint Konfigurieren

Zur Konfiguraiton des Accesspoints wird die Datei `/etc/hostapd/hostapd.conf` beschrieben. Wichtig dabei ist eine geeignete SSID und ein geeignetes Passwort für das Netzwerk.

```
interface=wlan0
driver=nl80211
ssid=smartroom
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=smarthome
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
ctrl_interface=/var/run/hostapd
ctrl_interface_group=0
```

Erkennbar ist die SSID `smartroom` und das Passwort `smarthome`. Anschließend muss der Accesspoint Software noch die Konfigurationsdatei in der Datei `/etc/default/hostapd` bekannt gemacht werden:

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

3.5 Starten des Accesspoints

Nun muss nur der Accesspoint gestartet werden und die Netzwerkkonfiguration ist abgeschlossen.

```
sudo systemctl unmask hostapd
sudo systemctl enable hostapd
sudo systemctl start hostapd
```

4 Software der Samrthome Umgebung

Der Aufbau der Smarthome Umgebung auf dem Raspberry Pi gliedert sich in drei Ebenen:

- Logik Ebene
- Transport Ebene
- Interface Ebene

Diese Ebenen werden jeweils über bestimmte Softwarepakete realisiert, die auf dem Raspberry Pi installiert und konfiguriert werden müssen. Die Logik Ebene dient dabei zur Programmierung der Logik und Darstellung der erfassten Daten beziehungsweise zur Steuerung der Aktoren. Die Transport Ebene ist die Schnittstelle zwischen der Logik Ebene und der Interface Ebene. Sie stellt sicher, dass die beiden Ebenen jeweils verlässlich strukturierte Daten erhalten. Zusätzlich dient sie als Schnittstelle zu den Mikrocontrollern über den WLAN-Accesspoint. Die Interface Ebene dient dem Anbinden der kommerziellen Sensoren und Aktoren.

Im Folgenden wird die Funktion, Installation und Konfiguration der einzelnen Ebenen genauer erläutert und beschrieben.

4.1 Logik Ebene

Die Logik der Smarthome Umgebung wird durch die Software *Node-Red* realisiert, die Präsentation der Daten und die Steuerung der Aktoren durch die Node-Red Erweiterung *Node-Red-dashboard*.

4.1.1 Node-Red

Node-Red ist ein auf der Plattform node.js basierendes und in JavaScript geschriebenes grafisches Entwicklungswerkzeug. Es ist speziell entwickelt um Hardware, APIs und Dienste mittels eines Baukasten Prinzips miteinander zu verbinden. Somit ist es ideal für dieses Projekt um die Informationen der Sensoren zu sammeln, auszuwerten und darauf basierend Entscheidungen zu treffen.

Die Installation von Node-Red ist dank eines von Seiten des Herstellers bereitgestellten Installationsskriptes sehr einfach. Das Skript wird mittels folgendem Befehl heruntergeladen, ausgeführt und führt anschließend den Nutzer Schritt für Schritt durch die Installation:

```
1 bash <(curl -sL https://raw.githubusercontent.com/Node-Red/raspbian-deb-package/
    master/resources/update-nodejs-and-nodered)
```

Nach erfolgter Installation sorgt der Befehl `sudo systemctl enable nodered.service` dafür, dass Node-Red bei Systemstart automatisch gestartet wird. `sudo service nodered start` startet Node-Red anschließend. Die Entwicklungsumgebung ist ab diesem Zeitpunkt unter der in Kapitel 3 auf Seite 4 genannten IP-Adresse und dem Port 1880 erreichbar: `https://141.75.33.126:1880` .

4.1.2 Node-Red-dashboard

Das Node-Red-dashboard ist eine Erweiterung für Node-Red und ermöglicht die Konfiguration einer Steuerzentrale mittels Node-Red. Es wird nach erfolgter Node-Red Installation über den Befehl `npm i Node-Red-dashboard` installiert. Das Dashboard ist ab dem Zeitpunkt unter dieser Adresse erreichbar: `https://141.75.33.126:1880/ui` . Ebefalls sind nun die Node-Red-dashboard spezifischen Nodes in Node-Red verfügbar. Node-Red und das Node-Red-dashboard werden näher in Kapitel **??** auf Seite **??** erklärt.

4.2 Transport Ebene

Auf der Transport Ebene wurde das Internet of Things (IOT) Protokoll *MQTT* mit der Software *Mosquitto* gewählt.

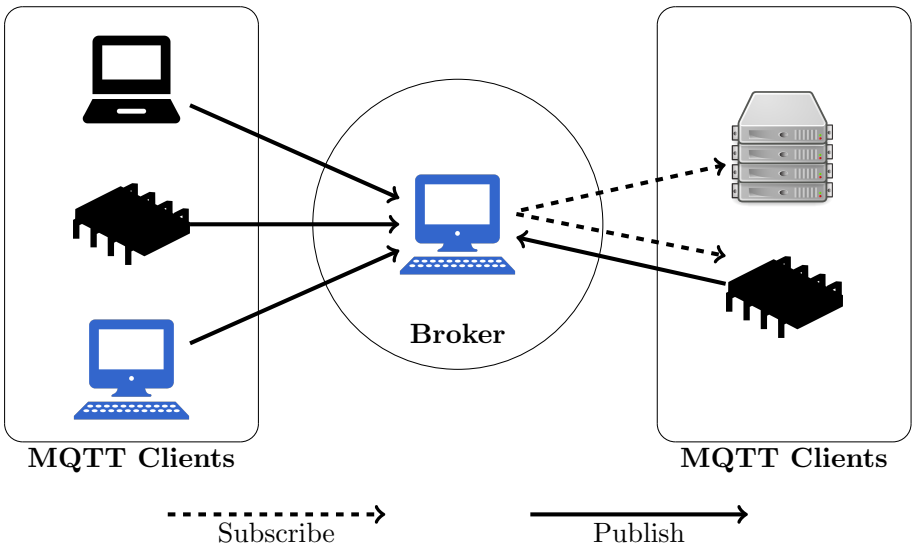


Abbildung 1: Die Architektur von MQTT

Das Message Queue Telemetry Transport (MQTT) ist ein Maschine-zu-Maschine (M2M) Nachrichtenprotokoll, entworfen nach dem Publish/Subscribe-Modell (pub/sub). Es baut auf einem zugrundeliegendem TCP/IP Netzwerk auf und wurde speziell für M2M und mobile Anwendungen entwickelt. Das Publish/Subscribe-Modell basiert auf einem *Broker*, der ähnlich wie ein Server die zentrale Anlaufstelle der Netzwerks und für die Verteilung der Nachrichten zuständig ist. Neben dem Broker gibt es beliebig viele *Subscriber* und *Publisher*. Ein Subscriber empfängt bestimmte Nachrichten vom Broker, ein Publisher sendet Nachrichten an den Broker, wobei ein einzelnes Gerät beide Rollen einnehmen kann. In Abbildung 1 gibt es drei Geräte die ausschließlich Daten an den Broker senden, einen Server, der nur Daten vom Broker empfängt und einen Mikrocontroller, der sowohl Daten sendet, als auch empfängt.

Abbildung 2: Aufbau von MQTT Topics

Das Senden der Daten an den Broker wird veröffentlichen *(publish)* genannt. Möchte ein Client Daten empfangen, muss er diese abonnieren *(subscribe)*. Dieses Abonnement wird bei erstmaliger Verbindung zum Broker in Form einer Publish-Nachricht angemeldet. Identifikation von Nachrichten geschieht über *Topics*, die von dem sendenden Gerät festgelegt werden.

Der Broker benutzt diese Topics, die die Form eines UTF-8 Strings haben, um Nachrichten der verbundenen Geräte zu filtern. Topics bestehen aus mehereren Level, die durch einen Slash voneinander getrennt werden. In Abbildung 2 ist beispielhaft eine in dieser Arbeit benutzte Topic gezeigt, unter der die von einem Ultraschallsensor gesammelten Daten abonniert werden. Dabei ist zu Erkennen, wie die Topic hierachisch mit verschiednen Level aufgebaut ist: Es wird begonnen mit einem allgemein gültigem Level. In diesem Fall kennzeichnet dies die Hochschule (gso). Weiterführend ist das Gebäude, der Raum und der entsprechende Sensor festgelegt. Dadurch wird mit jedem weiteren Level eine immer genauere Herkunft der Daten festgelegt. Die letzte Topic `/#` ist eine *Wildcard*. Wildcards werden bei MQTT genutzt um

mehrere Topics gleichzeitig zu abonnieren. `#` ist dabei eine *Multilevel Wildcard*, die mehrere Topic Level abonniert. So werden beispielhaft die Topics `gso/bb/104/ultraschall/status` und `gso/bb/104/ultraschall/status/subsensor-1` beide mit der Topic aus Abbildung 2 abonniert, wohingegen die Topic `gso/bb/104/pir/status` nicht berücksichtigt wird. Da die Multiline Wildcard bedeutet, dass alle folgenden Level uneingeschränkt abonniert werden, kann sie nur am Ende einer Topic stehen. Neben der Multilevel Wildcard steht die Singlelevel Wildcard, repräsentiert durch das `+`. Sie kann an einem beliebigen Topic Level stehen und ersetzt nur ein einziges Level der Topic. Sollen beispielsweise die Statusmeldungen aller Ultraschallsensoren im Gebäude BB abonniert werden, wird für den Raum die Singlelevel Wildcard benutzt: `gso/bb/+/ultraschall/status`.

Die Organisation des Datenfluss übernimmt der Broker, wie in Abbildung 3 gezeigt: Hier sind zwei Geräte mit dem Broker verbunden, die die Rollen eines Subscribers und eines Pulishers zeigen.

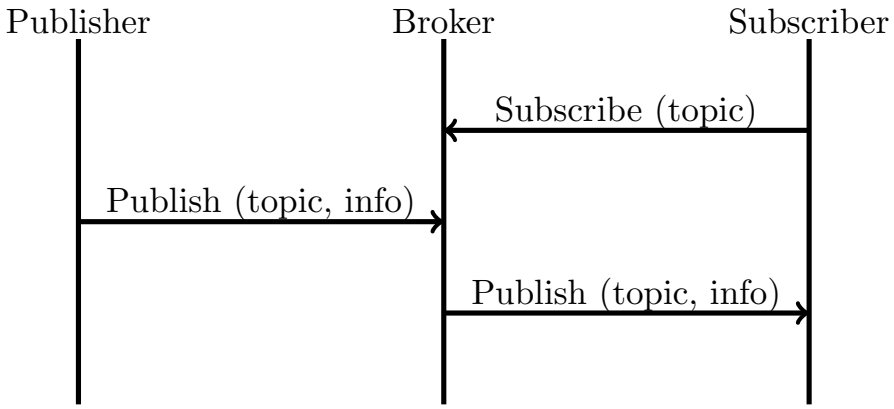


Abbildung 3: Der Publish/Subscribe Prozess von MQTT

Der Subscriber sendet seine Abonnement-Nachricht an den Broker und legt die abonnierte Topic fest. Der Broker registriert diese und leitet von nun an jede Nachricht mit der entsprechenden Topic an den Client mit dem Abonnement weiter. Abonniert zum Beispiel der Server aus Abbildung 1 die Topic `gso/bb/104/ultraschall`, registriert dies der Broker. Veröffentlicht nun ein Sensor Daten unter der selben Topic, gehen diese zunächst beim Broker ein, der die Daten dann sofort an alle Abonnenten dieser Topic, in deisem Fall den Subscriber, weiterleitet.

Neben den bereits gezeigten Eigenschaften von MQTT gibt es noch eine Vielzahl weiterer, wie zum Beispiel die Last-Will Nachricht, die andere Geräte informiert, dass das Sendende Gerät von dem Netzwerk getrennt wurde. Um die Komplexität der Arbeit im Rahmen zu halten wurde aber auf die Implementierung dieser Funktionen verzichtet.

Mosquitto ist eine Open-Source Implementation des MQTT-Protokolls und wird in diesem Projekt aufgrund der einfachen Verfügbarkeit als Debian-Paket für den Raspberry Pi benutzt. Es ist ein Projekt der Eclipse Foundation und da die Software in C geschrieben ist, sehr schnell, flexibel und weitreichend erhältlich. Installiert wird das Paket auf dem Raspberry Pi mit dem Befehl:

```
1 sudo apt install mosquitto
```

Dadurch werden die drei Teile des Mosquitto-Projektes installiert: Der `mosquitto` Server, die `mosquitto_sub` und `mosquitto_pub` Anwendungen und ein MQTT-C/C++-Bibliothek-Wrapper,

der hier aber nicht benutzt wird. Um den Mosquitto Server zu starten werden die folgenden Kommandos benötigt:

```
sudo systemctl enable mosquitto
sudo systemctl start mosquitto
```

Von nun an ist der Mosquitto Broker über die IP-Adresse des Raspberry Pis und mit dem Port 1883 erreichbar. Um die Funktionalität zu testen können die Anwendungen `mosquitto_sub` und `mosquitto_pub` genutzt werden: Mit Hilfe des Befehls

```
mosquitto_sub -t 'test/topic' -v
```

kann man testweise eine Topic abonnieren und mit dem Befehl

```
mosquitto_pub -t 'test/topic' -m 'hello world'
```

eine Testnachricht abschicken. Kommt die Nachricht in dem `mosquitto_sub` -Fenster an, ist der MQTT-Broker einsatzbereit. Um zu testen ob Geräte richtig an das Netzwerk angeschlossen sind, ist es ebenfalls hilfreich den mosquitto_sub Client mit der Wildcard # zu Nutzen um alle eingehenden Nachrichten einzusehen. Im laufenden Betrieb läuft Mosquitto dann weitestgehend im Hintergrund und es muss sich nur um angemessene Definitionen der Topics gekümmert werden.

4.3 Interface Ebene

Die letzte Ebene dient der Kommunikation mit den kommerziellen Sensoren. Da jeder Hersteller von Smarthome Geräten sein eigenes Protokoll benutzt um mit seinen Geräten zu kommunizieren, wird hier Software, die in der Lage ist verschiedene Protokolle zu sprechen, mit Hardware - den Funk-Sendern und -Empfängern - gekoppelt.

4.3.1 Homegear

Die in dem Blogartikel von Patrik Mayer in Kapitel 2 auf Seite 4 genutzte Software um mit verschiedenen Aktoren und Sensoren zu kommunizieren ist *homegear*. Homegear ist ein Open-Source Programm um IoT Geräte zentral zu kontrollieren und zu verwalten. Dazu spricht es eine Vielzahl an Protokollen diverser Hersteller wie Homematic, Intertechno, Philips Hue und Sonos, und unterstützt auch das MQTT-Protokoll. Homegear ist dabei in der Lage sowohl als komplette Smarthome Umgebung inklusive konfigurierbaren Dashboard zu dienen als auch als bloße Schnittstelle die die Verbindung mit den kommerziellen Geräten übernimmt und die Daten über HTTP, MQTT oder anderwertig weiterleitet. Um die Flexibilität der Umgebung zu erhöhen wird homegear hier nur als Schnittstelle genutzt.

Homegear selbst besteht aus dem Homegear-Grundmodul, welches die grundlegende Funktionalität bereitstellt und weiteren spezifischen Modulen, die die verschiedenen Hersteller/Protokolle bedienen oder das Dashboard zur Verfügung stellen. Installiert auf dem Raspberry Pi wird das Grundmodul über das Offizielle Repository:

```
sudo apt install apt-transport-https
wget https://apt.homegear.eu/Release.key && apt-key add Release.key && rm
    Release.key
sudo echo 'deb https://apt.homegear.eu/Raspbian/ stretch/' >> /etc/apt/sources.
    list.d/homegear.list
sudo apt update
sudo apt install homegear homegear-nodes-core homegear-management
```

Die verfügbaren Hersteller-spezifischen Module können in der Homegear Dokumentaiton [1] gefunden werden. Homematic BidCos, welches das Standard Funkprotokoll von Homematic bis 2015 war, kann beispielsweise über das vorher eingerichtete Repository ganz einfach mittels `apt install` installiert werden:

```
sudo apt install homegear-homematicbidcos
```

Nach der Installation eines Moduls muss der homegear-Service mittels

```
sudo service homegear restart
```

neu gestartet werden. Die ordnungsgemäße Funktion des Moduls kann anschließend in der Log-Datei von homegear überprüft werden:

```
tail -n 1000 -f /var/log/homegear/homegear.log
```

Funktioniert das Modul einwandfrei muss als nächstes das Funkmodul Homegear bekannt gemacht werden. Für das Modul aus Abschnitt 2 müssen folgende Zeilen zu der Homematic BidCos Konfigurationsdatei `/etc/homegear/homematicbiscos.conf` hinzugefügt werden:

```
[HomeMatic Wireless Module for Raspberry Pi]
id = My-HM-MOD-RPI-PCB
# Uncomment this if you want the HM-MOD-RPI-PCB to be your default communication
    module.
default = true
deviceType = hm-mod-rpi-pcb
device = /dev/ttyAMA0
responseDelay = 95
gpio1 = 18
```

Homematic BidCos Geräte werden bei korrekter Funktion des Moduls über das Komandozeilen-Interface hinzugefügt. Über `homegear -r` wird das Komandozeilen-Interface gestartet. Mittels `families select 0` wird die Homematic BidCos Gerätefamilie ausgewählt. `pairing on` versetzt homegear schließlich in den Pairing-Modus. Nun muss nur noch der Pairing-Modus an dem entsprechenden Gerät eingeschaltet werden. Nach wenigen Sekunden können mit dem Befehl `ls` die Verbundenen Geräte angezeigt werden.

Um über Node-Red mit den angeschlossenen Geräten zu kommunizieren muss das MQTT-Interface von homegear eingerichtet werden. Dazu wird dient die Konfigurationsdatei `mqtt.conf` in dem homegar Konfigurationsverzeichnis `/etc/homegear`.

4.3.2 Homematics Protokolle

Wie in Abschnitt 4.3.1 bereits erwähnt wurde das Standardprotokoll von Homematic *Homematic BidCos* im Jahre 2015 von *Homematic IP* abgelöst. Homegear unterstützt das neue Protokoll Homematic IP jedoch nicht. Um beispilsweise den bereits angesprochenen Präsenzsensor von Homematic nutzen zu können, der das Homematic IP Protokoll benutzt, ist es notwendig eine Alternative zu homegear zu finden. Dazu wurden nach einer Webrecherche drei Optionen gefunden:

1. RaspberryMatic
2. YAHM
3. piVCCU

RaspberryMatic ist ein von Jens Maus entwickeltes Projekt, welches ein einfaches, Linux/ buildroot-basierendes Homematic kompatibles Betriebssystem für Einplatinencomputer wie den Raspberry Pi zur Verfügung stellt. Es basiert auf der **O**pen-**C**entral-**C**ontrol-**U**nit-SDK (OCCU) die von eq-3 zur freien Verfügung gestellt wird. Damit ist RaspberryMatic in der Lage die gesamte Protokollpalette von Homematic zu bedienen.

Der Vorteil von RaspberryMatic ist die einfachheit der Bedienung und die enorme Anzahl an extra Features. Verworfen wurde RaspberryMatic jedoch, da es nur als komplettes Betriebssystemimage installierbar ist. Dies bedeutet, dass bei Benutzung beziehungsweise bei der Installation von RaspberryMatic die gesamte bestehende Konfiguration des Raspberry Pi zurückgesetzt wird und widerholt werden muss nachdem das neue Betriebssystem installiert ist, was sehr Aufwändig und nicht Anwenderfreundlich ist. Zusätzlich ist der Nutzen eines Alternativ-Programmes zu homegear nur als Zwischenlösung gedacht. Implementiert homegear das Homematic IP Protokoll in der Zukunft oder wird kein Homematic IP Gerät in der Smarthome Umgebung benutzt, ist es wünschenswert RaspberryMatic vom Raspberry Pi entfernen zu können. Dies ist mit RaspberryMatic aber nicht ohne größeren Aufwand möglich. Aus diesen Gründen wurde von der Benutzung von RaspberryMatic abgesehen.

YAHM ist eine von Leonid Kogan entwickelte Lösung, die die Central-Control-Unit 2 (CCU2) von eq-3 in einem Virtuellen Linux Container (LXC) auf dem Raspberry Pi emuliert. Die CCU2 wird normalerweise von eq-3 separat mit eigener Hardware verkauft, durch YAHM läuft sie als eigenständiges System auf dem Raspbery Pi. Das Hinzufügen von Geräten geschieht bei YAHM über das Homematic eigene Webinterface und es werden wie bei RaspberryMatic alle Homematic Protokolle unterstützt.

Die Entscheidung gegen YAHM ist gefallen, da YAHM seit Juni 2018 nicht mehr aktiv Entwickelt wird und damit die Akutalität der Installation nicht gewährleistet werden kann.

piVCCU ist die dritte Alternative um Homematic IP zu benutzen. piVCCU ist ein Project von Alexander Reinert, welches wie YAHM einen Virtuellen Linux Container benutzt um die CCU2 oder auch die CCU3 (die neueste Generation der Homematic CCUs) auf dem Raspberry Pi zu emulieren. Durch den Virtuellen Linux Container wird ebenfalls wie bei YAHM das Homematic Webinterface erreichbar gemacht, womit das Anlernen, Konfigurieren und Bedienen aller mit der CCU2 beziehungsweise CCU3 kompatiblen Homematic BidCos und Homematic IP Geräte ermöglicht wird. piVCCU ist im Rahmen eines Debian-Repositorys für den Raspberry Pi installierbar und lässt sich damit bei Bedarf sehr einfach wieder von der Smarthome Umgebung entfernen. Im Gegensatz zu YAHM wird an piVCCU immer noch aktiv gearbeitet, weshalb die Entscheidung auf piVCCU gefallen ist.

4.3.3 Homematic CentralControlUnit mittels piVCCU

Die Installation von piVCCU erfolgt anhand der auf Github bereitgestellten Installationsanleitung [4]. Dabei wird direkt die aktuelle Version 3 der CCU (CCU3) gewählt.

Nach erfolgreicher Installation wird mittels des Befehls `sudo pivccu-info` in Schritt 10 die IP-Adresse der CCU ermittelt. In diesem Fall ist sie `141.75.33.250`. Unter dieser IP-Adresse ist das Webinterface der CCU von nun an im Hochschulnetzwerk erreichbar und die Homematic Geräte können angelernt werden. Das Anlernen und die Funktionen des Webinterfaces werden in Abschnitt **??** auf Seite **??** genauer erläutert.

Um piVCCU vollständig in die Smarthome Umgebung einzubinden muss noch die Kommunikation mit dem MQTT Broker und Node-Red eingerichtet werden. piVCCU stellt dafür keine eigene Schnittstelle zur Verfügung, weshalb eine Behelfslösung für die Kommunikation gewählt wurde. Diese stellt sich in Form von RedMatic dar.

RedMatic ist ein Addon für die CCU3 und RaspberryMatic, was eine Node-Red Installation auf diesen Geräten verfügbar macht. Da durch piVCCU die originale CCU-Software auf dem Raspberry Pi emuliert wird, ist RedMatic auch unter piVCCU installierbar. Die Nutzung von RedMatic bedeutet, dass eine zweite, eigenständige Node-Red Installation in der Smarthome Umgebung läuft, welche sich aber auf die in dem LXC-Container betriebene CCU beschränkt

und nichts mit der eigentlichen Node-Red Installation auf dem Raspberry Pi zu tun hat. Sie dient lediglich der Kommunikation der CCU über MQTT mit Node-Red auf dem Raspberry Pi. Der Nachteil dieser Lösung ist der größere Fest- und Arbeitsspeicher Bedarf des Raspberry Pis. Dies fällt jedoch nicht ins Gewicht, solange die Smarthome Umgebung sich auf einzelne Räume der Hochschule beschränkt und damit eine überschaubare Anzahl an Geräten angeschlossen sind. Bei einer größeren Expansion des Systems ist es zu Empfehlen, die Kommunikation der CCU mit dem Raspberry Pi auf Ebene des Linux Containers in Form einer TCP/IP-Kommunikation zu realisieren.

Installiert wird RedMatic nach der Installtionsanleitung für RedMatic [3] über die Weboberfläche der CCU. Nach einen Neustart des Raspberry Pis ist RedMatic unter der Adresse `http://141.75.33.250/addons/red` erreichbar. Um den unberechtigten Zugriff auf RedMatic zu verhindern ist der Zugriff Passwort geschützt. Die Zugangsdaten wurden folgendermaßen festgelegt:

```
User: pi
Password: smarthome
```

Die Anbindung von piVCCU an den MQTT Broker geschieht mittels der RedMatic spezifischen Homematic-Nodes. Wie diese Nodes Konfiguriert werden wird in Kapitel **??** auf Seite **??** beschrieben.

5 Fazit der Smarthome Umgebung

Die in dem Blogbeitrag "Homematic mit node-red über homegear" von Patrik Mayer [2] beschriebene Smarthome Umgebung konnte weitestgehend Umgesetzt werden. Das grobe Gerüst mit Node-Red, dem Node-Red-dashboard, MQTT als Transport-Protokoll und Homegear um mit kommerziellen Systemen zu kommunizieren konnte übernommen werden. Wo die beschriebene Gateway von der aus dem Blogbeitrag abweicht, ist die spezifische Netzwerk Konfiguration die durch die Hochschulumgebung begründet ist. Eigens Entwickelte Sensoren können nicht das Hochschulnetzwerk benutzen, da sie die verwendete Verschlüsselung nicht unterstützen. Stattdessen dient der Raspberry Pi als Accesspoint für die Sensoren mit einem eigenen WLAN-Netzwerk.

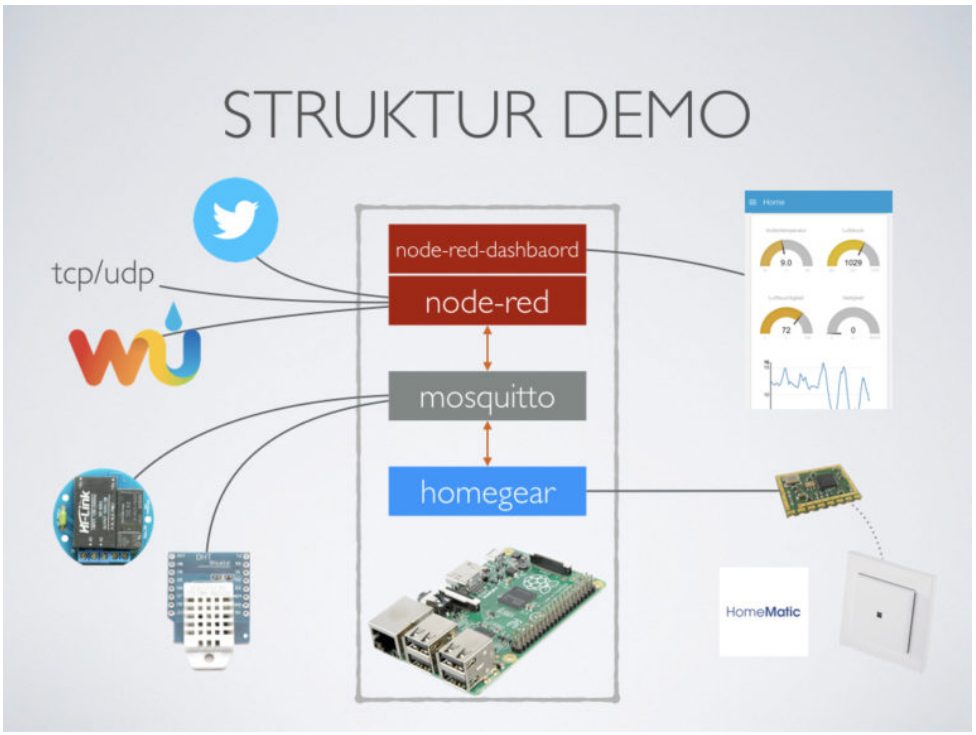
Zusätzlich waren zur Zeit der Veröffentlichung des Blogbeitrags im Jahr 2017 Geräte die das Protokoll Homematic IP nutzen nicht sehr weit Verbreitet. Dies hat sich mit der Zeit geändert und Homematic IP ist mittlerweile laut Herstellerseite eines der meistgenutzten Protokolle in Europa. Da Homegear dieses Protokoll aufgrund der fehlenden Quelloffenheit nicht unterstützt, wurden verschiedene Alternativen getestet und schließlich die originale Gateway Software von Homematic in Form von piVCCU auf dem Raspberry Pi emuliert. Durch Nutzen von piVCCU wird die Kommunikation mit kommerziellen Sensoren zweigeteilt: Homematic Sensoren werden über die Homematic Weboberfläche die unter piVCCU läuft angesprochen. Verwaltung aller anderen kommerziellen Sensoren übernimmt Homegear.

Die Abbildung 4 zeigt die Beschriebenen Unterschiede der beiden Smarthome Umgebungen noch einmal vergleichend auf.

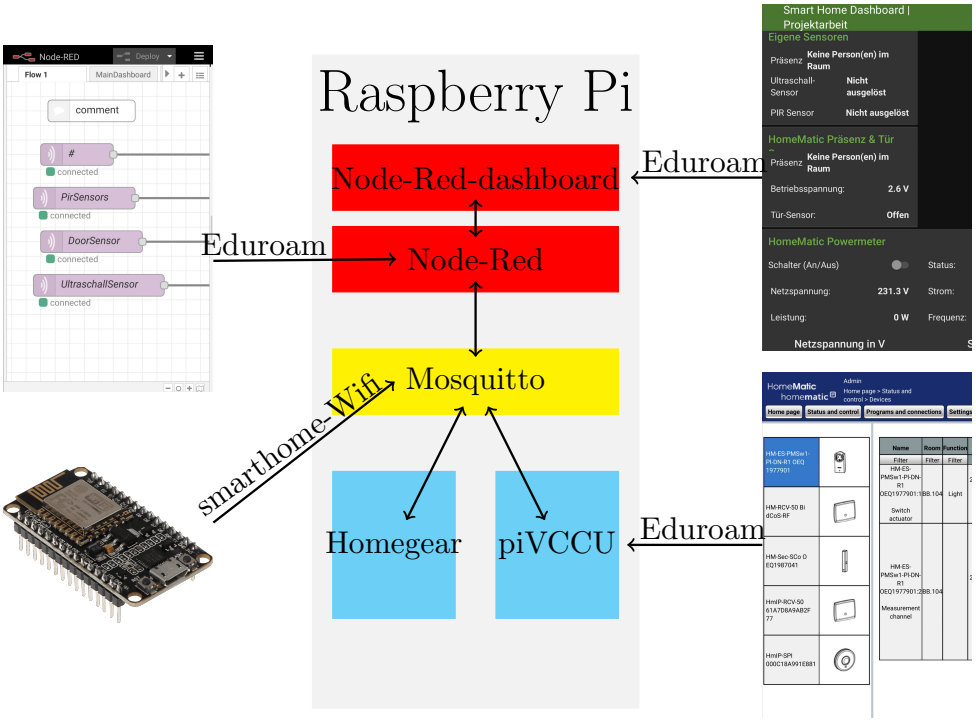
6 Ausblick

Um die entwickelte Smarthome Umgebung robuster zu gestalten, ist es wünschenswert, den bisher eingeschlagenen Weg über RedMatic abzuschaffen. Wie in Abschnitt 4.3.3 auf Seite 13 erwähnt, sollte der Nachrichtentransport der CCU statt über eine zweite Node-Red Installation direkt mit Hilfe eines Skriptes mittels TCP/IP an Mosquitto erfolgen. Alternativ ist es auch denkbar die piVCCU an Homegear über die Homegear-Schnittstelle XML-RPC anzubinden. Dazu ist aber weitere Recherche notwendig. Idealerweise implementiert Homegear in der Zukunft das Homematic IP Protokoll und mach das Benutzen von piVCCU überflüssig.

Um die Sicherheit des Systems zu Verbessern, unterstützt MQTT die Nutzung von SSL-Verschlüsselung, wodurch Nachrichten verschlüsselt übertragen werden können. Dies erfordert die Nutzung von SSL-Zertifikaten, die auch auf den Sensoren bekannt gemacht werden. Außerdem ist es wünschenswert Eduroam für die WLAN-Sensoren zu nutzen. Mit der Verbindung der WLAN-Sensoren zu dem vom Raspberry Pi aufgesetztem WLAN-Netzwerk, ist es Nötig, in jedem Raum in dem Sensoren genutzt werden sollen, einen Raspberry Pi an das LAN-Netzwerk der Hochschule anzuschließen. Sind die Sensoren jedoch im WLAN-Netz der Hochschule reicht ein Zentraler Computer als Zentrale der Samrthome Umgebung.



(a)



(b)

Abbildung 4: Vergleich der Smarthome Umgebungen: 4a zeigt die von Patrik Mayer [2] vorgeschlagene Umgebung, 4b zeigt die in dieser Arbeit umgesetzte Umgebung

7 Personenerkennung mit Hilfe einer Kamera

Als ein weiteres Vergleichsmittel zur Personenerkennung im Labor wurde eine 5 Megapixel Kamera für den Raspberry Pi zusammen mit einem Bilderkennungs Skript genutzt. Aus Datenschutzgründen ist die Benutzung einer Kamera zwar nicht dauerhaft möglich in der Hochschule, zum Vergleichen der Effektivität mit der in dieser Arbeit entwickelten Präsenzerkennung und dem Präsenzerkennungssensor von Homematic jedoch sehr hilfreich.

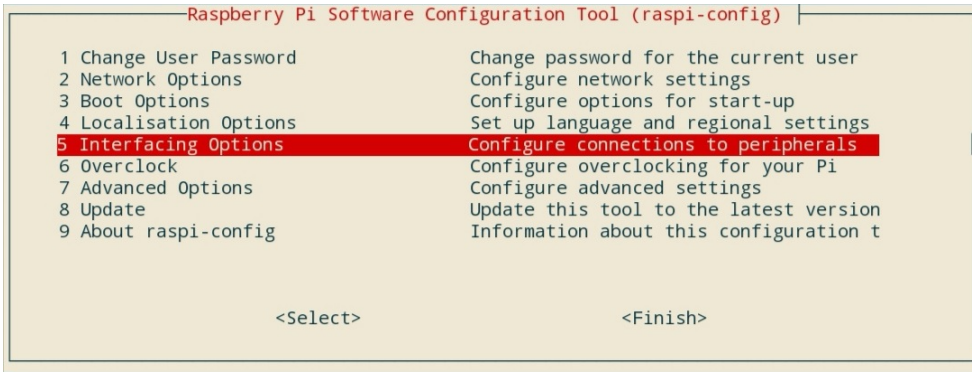
Angeschlossen wird die Kamera über ein spezielles Flachbandkabel an den Raspberry Pi. Die Aktivierung der Kamera erfolgt über das Raspberry Pi Software Configuration Tool, das mit dem Befehl `sudo raspi-config` aufgerufen wird. Dort wird anschließend wie in Abbildung 5 gezeigt die Kamera aktiviert.

Um die von der Kamera aufgenommenen Bilder zu analysieren wird Python zusammen mit den Modulen OpenCV, Numpy, Imutils und imagezmq benutzt. Der Sourcecode ist auf dem EFI Git Service unter dem Projekt `heimbsle69869 / smarthome-presence-detect` in dem Ordner `Camera` zu finden. Zur Personenerkennung wird das Skript `person_detection.py` genutzt. Es nutzt imutils um einen Videostream zu lesen, von dem jeder Frame in ein OpenCV-Array eingelesen wird. Anhand dieses Arrays werden durch ein bereits für die Personenerkennung trainierten Tensorflow-Erkennungsmodell mit einer definierbaren Konfidenz Personen erkannt. Ändert sich die Anzahl der erkannten Personen von einem Frame auf den anderen, wird die neue Anzahl per MQTT an Node-Red veröffentlicht.

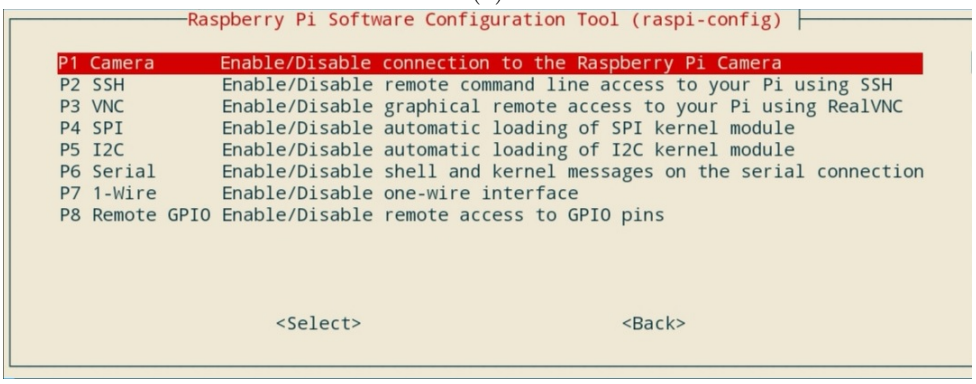
Die Präsenzerkennung mit dieser Methode wird begrenzt duch die Rechenleistung des Raspberry Pi, dem Aufnahmewinkel der Kamera und der Qualität des Trainings des verwendeten Erkennungsmodell. Durch die begrenzte Rechenleistung des Raspberry Pi werden die aufgenommenen Frames nur sehr langsam, mit etwa 3-10 Frames pro Sekunde, verarbeitet. Die Kamera nimmt mit 30 Frames pro Sekunde auf. Dies kann verbessert werden, indem die Frames vom Raspberry Pi über WLAN an einen leistungsfähigeren Computer übertragen werden, auf dem die Analyse stattfindet. So konnte die Verarbeitungsgeschwindigkeit auf fast 30 Frames pro Sekunde erhöht werden.

Das verwendete Erkennungsmodell erkennt Personen während des Tests zuverlässig. Allerdings werden sehr häufig fälschlicherweise mehr Personen erkannt als sich im Bild befinden. Dies könnte mit weiterem Training des Modells verbessert werden.

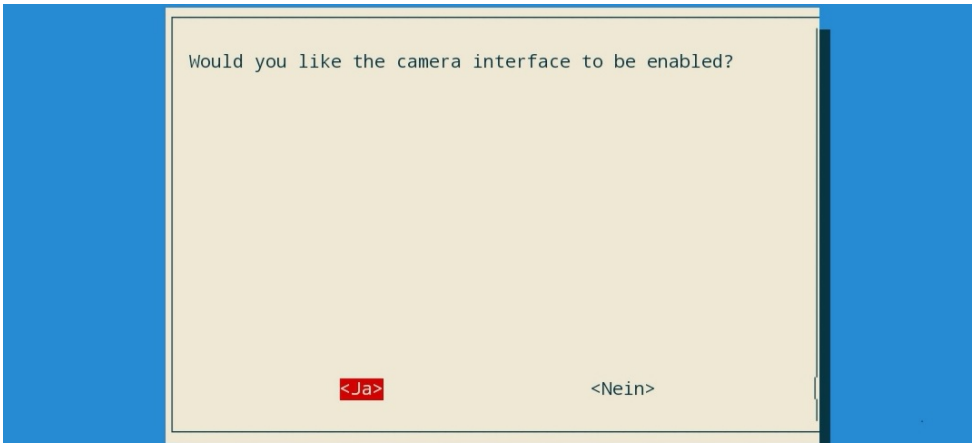
Fazit Kamera: Die Präsenzerkennung mit Hilfe einer Kamera ist nicht realistisch Anwendbar in einer Hochschulumgebung, da heutige Datenschutzrichlinien die Aufnahme Personenbezogener Daten ohne deren Einwilligung nicht erlauben. Sie ist jedoch in Anbetracht der Erkennungsqualität die vielversprechendste Lösung, da mit einem nur grundlegend trainierten Erkennungsmodell bereits sehr gute Ergebnisse erzielt wurden.



(a)



(b)



(c)

Abbildung 5: Drei Aktivierungsschritte der Praspberry Pi Kamera: 5a: Interfacing Options auswählen, 5b: Camera Options auswählen, 5c: Aktivierung bestätigen

Literatur

[1] Homegear. Homegear documentation. https://doc.homegear.eu/overview/, 2016-2018. [Online; aufgerufen 28-08-2019].

[2] Patrik Mayer. Homematic mit node-red über homegear. https://allgeek.de/2017/07/09/homematic-mit-node-red-ueber-homegear/, Juli 2017. [Online; aufgerufen 27-08-2019].

[3] Sebastian Raff. Installation. https://github.com/rdmtc/RedMatic/wiki/Installation, März 2019. [Online; aufgerufen 10-09-2019].

[4] Alexander Reinert. pivccu raspberry pi setup. https://github.com/alexreinert/piVCCU/blob/master/docs/setup/raspberrypi.md, Juni 2019. [Online; aufgerufen 09-09-2019].