

# **Intelligente Tür am TECO-Gebäude**

## **PSE-Projekt im Wintersemester 2021/2022**

### **Installationsanleitung**

Lukas Wittenzellner (Mtr.-Nr. 2295063), Daniel Luckey (Mtr.-Nr. 2295405),  
Ahmad Eynawi (Mtr.-Nr. 2307304), Fabian Schiekel (Mtr.-Nr. 2289152)  
und Jens Bosch (Mtr.-Nr. 2301884)

13. April 2022

# Inhaltsverzeichnis

<b>1</b>	<b>Abkürzungsverzeichnis</b>	<b>3</b>
<b>2</b>	<b>Voraussetzungen</b>	<b>4</b>
<b>3</b>	<b>Kamera</b>	<b>5</b>
<b>4</b>	<b>Code</b>	<b>11</b>
<b>5</b>	<b>Slack-Bots</b>	<b>15</b>
<b>6</b>	<b>App</b>	<b>18</b>
<b>7</b>	<b>Lautsprecher und Audiomodul</b>	<b>21</b>
<b>8</b>	<b>WLAN-Button</b>	<b>23</b>

# **1 Abkürzungsverzeichnis**

App      Applikation  
bzw.     beziehungsweise  
z.B.     Zum Beispiel

## 2 Voraussetzungen

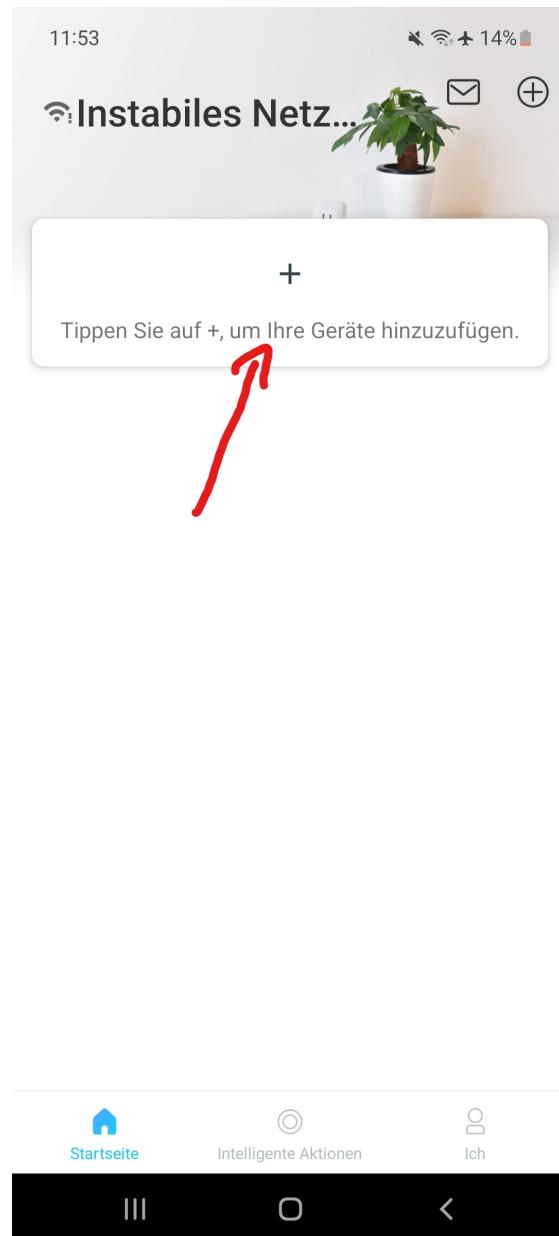
Für die Installation unseres Projektes wird benötigt:

- ein RaspberryPi (wir hatten einen RaspberryPi 3 Model B)
- eine WLAN-Kamera (Wir hatten eine Tapo C200)
- ein Server, auf dem die Konvertierung des Kamera Livestreams stattfinden kann.
- NFC-Tokens
- LED zum simulieren der Ausgaben, oder direkt das Türschloss.
- eine Möglichkeit um auf die Konsole des Pi's zugreifen zu können.
- ein NFC fähiges Handy Mobilgerät für die App.
- ein internetfähiges Endgerät für die Konfigurierung der Slack-Bots.
- einen Lautsprecher (CQRobot-Lautsprecher 8O3W-JST-PH2.0)
- ein Audiomodul für die Kommunikation zwischen dem Raspberry Pi und dem Lautsprecher (Adafruit Voice Bonnet)
- einen WLAN-Knopfschalter (Shelly Button 1)
- einen sekundären Raspberry Pi für die Installation des Audiomoduls (z.B. Raspberry Pi Zero W)

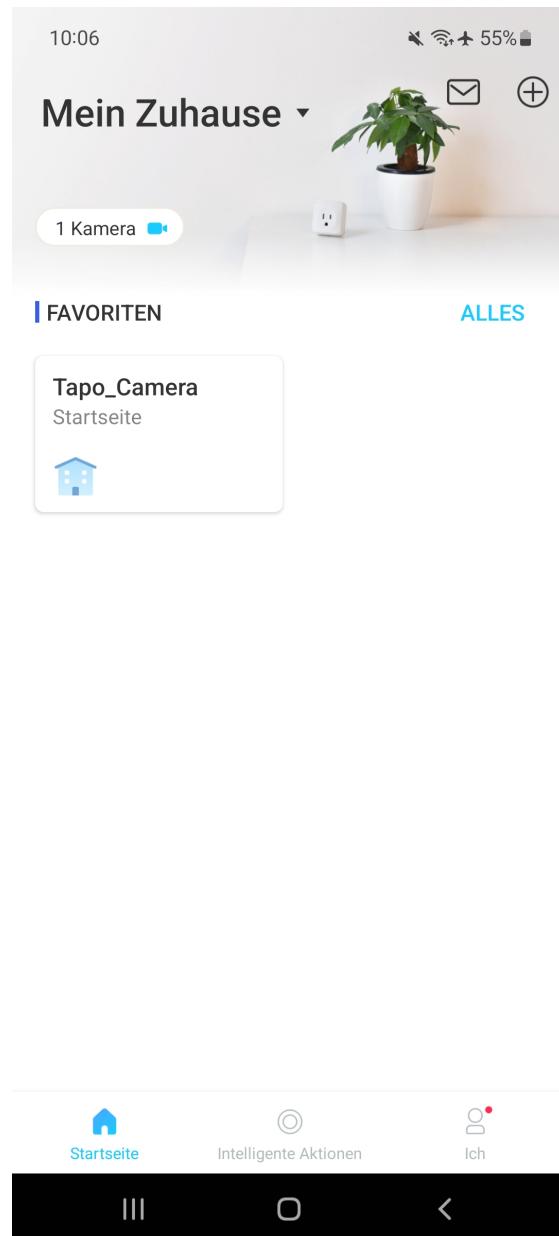
## 3 Kamera

Disclaimer: Die Anleitung ist für eine Tapo Kamera geschrieben und mit einer Tapo C200 getestet worden. Die Installation der Kamera erfolgt über die App von Tapo.

1. Download der App *TP-Link Tapo* aus dem Handy eigenen App-Store.
2. Kamera mit einer Stromquelle verbinden.
3. Die Kamera initialisiert sich und sollte sich dabei drehen. Dies dauert ein paar Sekunden.
4. Nun muss in der App ein Benutzerkonto angelegt werden.
5. Im Startbildschirm der App kann nun eine neue Kamera durch Drücken der Schaltfläche + hinzugefügt werden.



6. Nach der Auswahl des Modells (bei uns eine Tapo C200) leitet einen die App durch die Installation der Kamera.
7. Nach der Installation müssen noch ein paar Einstellungen in der App angepasst werden. Auf dem Startbildschirm sollte nun die Kamera angezeigt werden. In die Einstellungen kommt man durch Drücken der Kamera und anschließend des Einstellungssymbols.



8. Ein Benutzername und Passwort für den Zugriff auf die Kamera kann über Erweiterte Einstellungen->Kamerakonto angelegt werden. Diese Daten benötigt später auch der Pi um mit der Kamera kommunizieren zu können.

10:08 54%

← Erweiterte Einstellungen

- Kamerakonto 
- Verzerrungen korrigieren >  
Aus
- Audio aufnehmen >  
Ein
- Einstellungen der Bildschirmanzeige >
- Privatsphärenmodus >  
Ein
- Stromnetzfrequenz >  
Automatisch
- Diagnostik >  
Aus

Schwenken und Neigen-Korrektur

Tippen Sie hier, um Ihre Kamera auf die Standardposition zurückzusetzen. Sie können die Kamera während der Korrektur nicht mit Schwenk- und Neigefunktion steuern.



10:08 54%



Konto erstellen

Erstellen Sie ein Konto für die Anmeldung der Kamera über Portale von Drittanbietern. Hierdurch können Sie die Kamera verwalten und Videos auf einem NVR- oder NAS-Gerät eines Drittanbieters ansehen.

Benutzername

Passwort

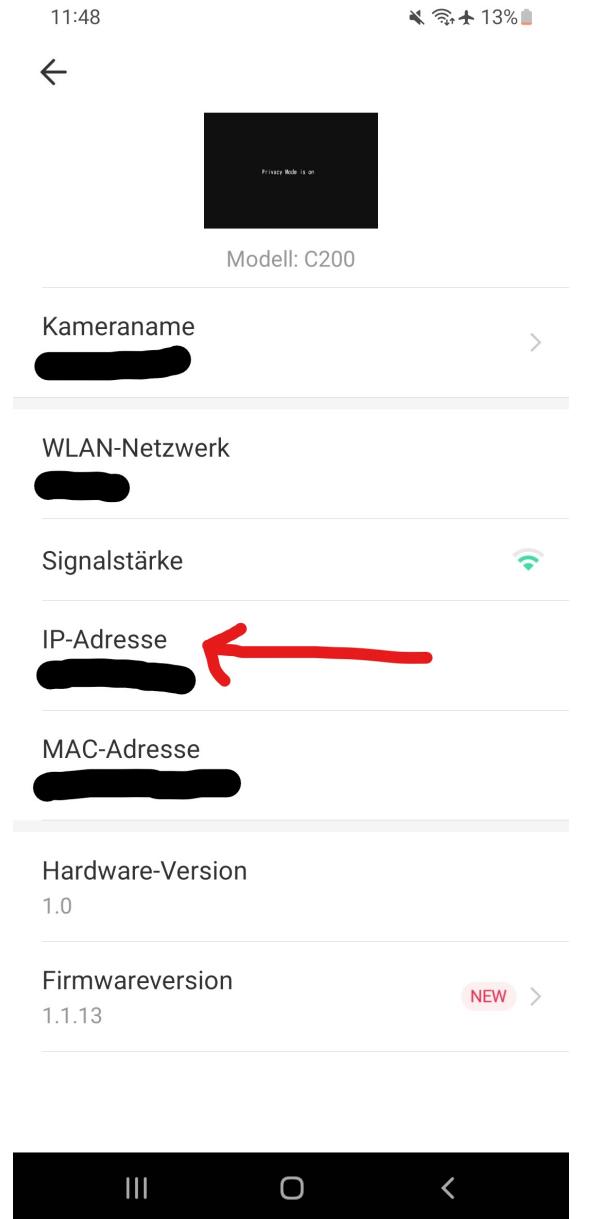
Hinweis: Tapo Care funktioniert am besten, wenn entweder der NVR oder die microSD-Karte aufzeichnet, nicht beide. Zu diesem Zeitpunkt wurde die NVR-Aufzeichnung deaktiviert. Um die Aufzeichnung auf dem NVR wieder zu starten, entfernen Sie die microSD-Karte aus der Kamera.

SPEICHERN



9. Die IP-Adresse kann über Tapo\_Camera->IP-Adresse herausgefunden werden.





- 10.** Empfohlen wird z.B. noch die Einstellungen Erkennung & Wahrnung->Bewegungserkennung und Erweiterte Einstellungen->Audio Aufnehmen aus zu schalten. Ebenfalls sollte immer bei Inaktivität der Kamera der Privatsphäremodus aktiviert sein, dieser wird nach dem Starten unseres Systems bei Inaktivität immer aktiviert sein.

## 4 Code

1. Github Rep clonen. (git clone URL `git clone https://github.com/Gostdragon/IoT-Door-Control-System.git`)
2. Den Ordner mit dem Namen *RaspberryPi* muss auf den RaspberryPi und der Ordner mit dem Namen *Website* muss auf einen Server.
3. Auf dem Pi:  
Disclaimer: Aus Performance Gründen ist unser Programm auf dem Pi in zwei Programme aufgeteilt. Mit dieser Anleitung werden die beiden Programme im Hintergrund laufen, sodass noch Eingaben auf der Kommandozeile für andere Dinge gemacht werden kann
  1. Falls pip nicht installiert ist, erst pip installieren. (`sudo apt-get install python3-pip`)
  2. In die Kommandozeile eingeben und installieren: `sudo apt-get install libsdl2-mixer-2.0-0 libsdl2-image-2.0-0 libsdl2-2.0-0`
  3. In den Ordner .../RaspberryPi/Init wechseln.
  4. In die Kommandozeile eingeben und installieren: `sudo pip install -r rootrequirementsPi.txt`
  5. In die Kommandozeile eingeben: `pip install -r requirementsPi.txt`
  6. Shell Script (ShellSkriptPi.sh) ausführen um die Importe zu testen, .service Dateien anzulegen und um aliase festzulegen. (`sh ShellSkriptPi.sh`)
  7. Warten auf erfolgreiches installieren der Bibliotheken. Bei einem Erfolg wird auf der Kommandozeile: *Installieren der Bibliotheken abgeschlossen.* ausgegeben, bei einem Fehler wird eine entsprechende Fehlermeldung ausgegeben.
  8. Das Shell-Skript ShellSkriptSSL.sh ausführen. Eingaben machen wie gefordert. Notwendig ist nur die Eingabe des Passwortes für die myCA.key, die

restlichen Angaben sind optional. Die Zertifikate werden später noch bei der App benötigt. (*sh ShellSkriptSSL.sh*

9. In die Kommandozeile eingeben: *sudo raspi-config* und dann Interface Options->SPI aktivieren.
10. Es müssen diverse Attribute des Systems angepasst werden. Diese können in der RaspberryPi/src/data\_model/configuration.py als default values gesetzt oder beim Initialisieren in der main.py mit übergeben werden. Es gibt Attribute, die auf jeden Fall gesetzt werden müssen, damit das System gestartet werden kann und es gibt Attribute, die gesetzt werden sollten, aber ohne die das System auch gestartet werden kann. Nachfolgend werden diese Aufgelistet:

Klasse	Variable
BotConfiguration	BotRingChannelID, BotRingToken und BotRingMessage
BotConfiguration	BotLogToken und die entsprechenden Token für die drei Log Stufen. (Siehe SlackBots)
CameraConfiguration	IP-Adresse der Kamera
CameraConfiguration	Name und Passwort der Kamera
PiConfiguration	Die Pinnummern müssen angepasst werden. (PinNumber-Button ist der Pin für das Eingangssignal der Klingel. Die Ausgabe bei dem erfolgreichen Öffnen der Tür liegt momentan an LEDGreen, falls ein nicht valider Token gelesen wird, wird der Ausgang LEDRed geschaltet. LEDYellow wird von uns nicht benutzt.)
PiConfiguration	IP-Adresse des Pi.
PiConfiguration	Pfad zu dem Verzeichnis, indem der Ordner RaspberryPi liegt.

Tabelle 4.1: Notwendig

Konstruktor	Variable
BotConfiguration	TimeBotMessageVisible und BotRingMessage.

Tabelle 4.2: Nicht Notwendig aber empfohlen

11. In der main.py und website.py muss noch der absolute Pfad zum Ordner angegeben werden, in dem der RaspberryPi-Ordner liegt. (Jeweils bei sys.path.append() direkt am Anfang des Moduls)

12. Damit die Aliase übernommen werden, muss bei einer ssh-Verbindung diese neu aufgebaut werden und bei einer Graphischen Oberfläche das Terminal neu gestartet werden.
13. Starten der .service Dateien mit `restart`<sup>1</sup>, `stop`<sup>2</sup> mit `stop`<sup>3</sup>. Initial können die beiden Programme mit `systemctl enable PSE_Main.service PSE_Website.service` gestartet werden.

Hinweis: Bei uns kam es einige male dazu, dass das Modul Webclient im Modul Slack nicht gefunden werden konnte. Sollte dies der Fall sein, sollte eine neuinstallation der Module slack und slackclient den Fehler beheben. Befehle in dieser Reihenfolge eingeben: `pip uninstall slack`, `pip uninstall slackclient`, `pip install slack`, `pip install slackclient`

#### 4. Auf dem Server:

1. In den Ordner .../Website/Init wechseln.
2. In die Kommandozeile eingeben: `sudo pip install -r rootrequirementsWebsite.txt`
3. In die Kommandozeile eingeben: `pip install -r requirementsWebsite.txt`
4. Shell Script ausführen um .service Dateien anzulegen und um aliase festzulegen. (`sh ShellSkriptWebsite.sh`)
5. Warten auf erfolgreiches installieren der Bibliotheken. *Installieren der Bibliotheken abgeschlossen.* auf der Kommandozeile, bei einem Fehler wird eine entsprechende Meldung ausgegeben.
6. Wie eben auf dem Pi müssen hier ebenfalls ein paar Änderungen vorgenommen werden. In dem Modul /Website/src/data\_model/configuration.py müssen die IP-Adresse des Pi und der Kamera in der Klasse PiConfiguration bzw. CameraConfiguration und der Username und das Passwort der Kamera in CameraConfiguration angepasst werden.
7. Wie eben muss hier ebenfalls der Pfad zu den Projekt Verzeichnis eingefügt werden. Hier in der /Website/src/website/website\_controler.py.( Bei `sys.path.append()` direkt am Anfang des Moduls)

---

<sup>1</sup>`systemctl start PSE_Main.service PSE_Website.service`

<sup>2</sup>`systemctl stop PSE_Main.service PSE_Website.service`

<sup>3</sup>`systemctl status PSE_Main.service PSE_Website.service`

8. Damit die Aliase übernommen werden, muss bei einer ssh-Verbindung diese neu aufgebaut werden und bei einer Graphischen Oberfläche das Terminal neu gestartet werden.
9. Starten der .service Dateien mit restart<sup>4</sup>, stop<sup>5</sup> mit stop<sup>6</sup>. Initial können die beiden Programme mit *systemctl enable PSE\_Website\_Server.service* gestartet werden.

---

<sup>4</sup>`systemctl start PSE_Website.service PSE_Website_Server.service`

<sup>5</sup>`systemctl stop PSE_Website_Server.service`

<sup>6</sup>`systemctl status PSE_Website_Server.service`

## 5 Slack-Bots

Falls das System mit Slack-Bots umgesetzt werden soll, ist im folgenden beschrieben, wie man die Slack-Bots konfigurieren muss.

1. Aufrufen der Seite <https://slack.com>
2. Anlegen eines neuen Kontos, oder anmelden mit einem bestehenden Account.
3. Anlegen eines neuen Workplaces. Dies geschieht entweder durch Betätigen der entsprechenden Schaltfläche oder der Prozess wird von Slack automatisch bei der ersten Anmeldung ausgeführt.
4. Für die Konfiguration ist es wichtig, dass Slack im Browser ausgeführt wird.
5. Um neue Apps(Bots) anzulegen muss auf folgende Seite gegangen werden, <https://api.slack.com/apps>.
6. Hier kann dann eine neue App angelegt werden. In unserem Projekt haben wir zwei verschiedene Apps benutzt. Eine für ein Klingelereignis und eine für die Simulation des Logs.
7. Die App für ein Klingelereignis:
  1. Basic Information (links am Rand) -> Add features and functionality -> Permissions. Hier muss bei *Scopes* -> *Bot Token Scopes* -> *add an OAuth Scope das chat::write Scope hinzugefügt werden.*
  2. Anschließend muss die App in den Workplace installiert werden. Dies geschieht über *OAuth Tokens for Your Workspace* -> *install to workplace*. Der Token der dabei generiert wird, ist der Token, mit dem der Bot angesprochen werden kann. Der Token muss dann entsprechend in der configuration.py Datei als BotRingToken gespeichert werden.
  3. Die App muss dann noch in dem Workplace in den Channel installiert werden. Die App kann über Rechtsklick auf den entsprechenden *Channel* -> *Channel-*

*Details öffnen -> Integration -> Eine App Hinzufügen* und dann durch Auswahl der entsprechenden App, installiert werden.

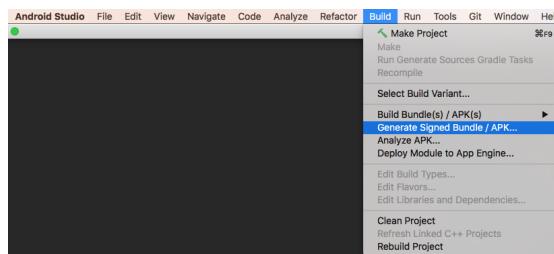
4. Sobald man in dem entsprechenden Channel ist kann der Channel-Token über die URL herausgefunden werden. (<https://app.slack.com/client/XXXX//hier> steht dann der Channel Token) Es müssen dann noch die entsprechenden Attribute in der configuration.py mit den richtigen Channel-Tokens gesetzt werden.
8. Die App für das Log:
  1. *Basic Information -> Add features and functionality -> Permissions. Hier muss bei Scopes -> Bot Token Scopes -> add an OAuth Scope ein chat::write Scope und das commands Scope hinzugefügt werden.*
  2. Anschließend muss die App in den Workplace installiert werden. Dies geschieht über *OAuth Tokens for Your Workspace -> install to workplace*. Der Token der dabei generiert wird, ist der Token, mit dem der Bot angesprochen werden kann. Der Token muss dann entsprechend in der configuration.py Datei als BotLogToken gespeichert werden.
  3. Die App muss dann noch in dem Workplace in den Channel installiert werden. Die App kann über Rechtsklick auf den entsprechenden *Channel -> Channel-Details öffnen -> Integration -> Eine App Hinzufügen* und dann durch Auswahl der entsprechenden App, installiert werden.
  4. Sobald man in dem entsprechenden Channel ist kann der Channel-Token über die URL herausgefunden werden. (<https://app.slack.com/client/XXXX//hier> steht dann der Channel Token) Es müssen dann noch die entsprechenden Attribute in der configuration.py mit den richtigen Channel-Tokens gesetzt werden.
  5. Sofern auch das Löschen des Logs unterstützt werden soll, muss noch eine Weiterleitung des Signals von den Slack-Servern zu dem Pi eingerichtet werden. Dies geschieht in Slack dadurch, dass die Slack-Server bei der Eingabe des Commands eine vorher definierte URL aufrufen, dieser Aufruf erfordert ein gültiges HTTPS Zertifikat. Ein Slash-Command kann über *Basic Information -> Add features and functionality -> Slash-Commands -> Create New Command* hinzugefügt werden. Unser System unterstützt momentan drei verschiedene Logstufen und damit können auch drei verschiedene Slash-Commands zum Löschen der Lognachrichten erstellt werden.  
Da der Pi keine öffentliche IP-Adresse hat, muss eine Weiterleitung an die private IP-Adresse des Pi's passieren. Die URL an den die Anfrage weitergeleitet werden soll ist: [http://\(IP-Adresse des Pi\)/clear\\_history\\_info](http://(IP-Adresse des Pi)/clear_history_info). Statt info

kann auch error bzw. fatal geschrieben werden um dann die entsprechenden Log-Stufen zu löschen.

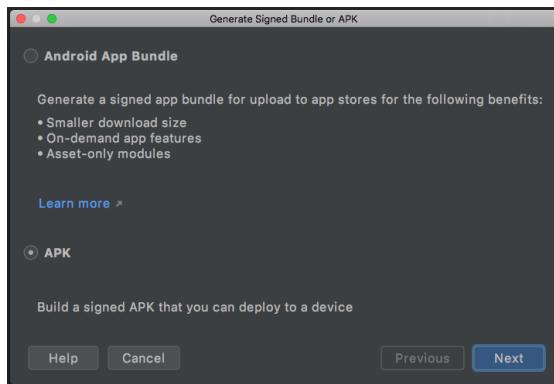
# 6 App

Hinweis: Punkte 1 und 3 müssen nur ausgeführt werden, wenn das SSL-Zertifikat des Servers verändert wurde. Punkte 2 und 3 können ausgeführt werden, um die Standard-IP die von der App als Adresse für den Server verwendet wird zu ändern. Die IP-Adresse kann aber auch nach der Installation in den Einstellungen der App geändert werden.

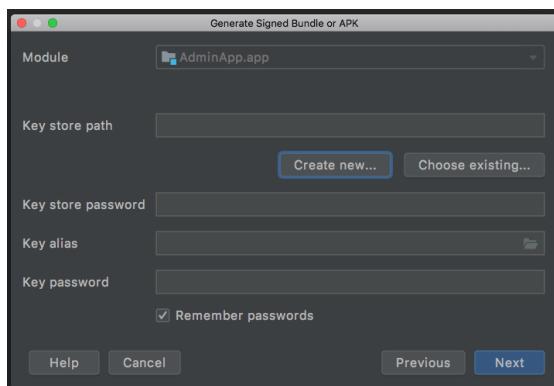
1. Das SSL-Zertifikat welches im Abschnitt Code erstellt wurde in die Datei **AdminApp/app/src/main/res/raw/extracas.cer** kopieren.
2. IP-Adresse des Servers in der Datei **AdminApp/app/src/main/res/values/default\_values.xml** an folgender Stelle einfügen:  
`<string name=fragment_preferences_hostname_default_value translatable=false>IP-Adresse</string>`
3. Signierte APK erstellen. Dies wird im Folgenden für AndroidStudio erklärt:
  - 3.1. Den Ordner **AdminApp** in AndroidStudio öffnen.
  - 3.2. Im Menü **Build > Generate Signed Bundle/APK...** auswählen



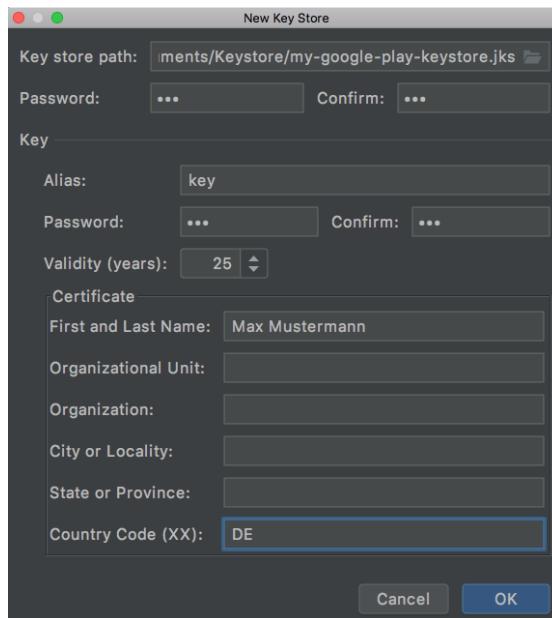
- 3.3. **APK** auswählen und **NEXT** klicken



- 3.4.** Falls noch kein Keystore zum Signieren der APK vorhanden ist, muss dieser erstellt werden. Hierfür **CREATE NEW** klicken.

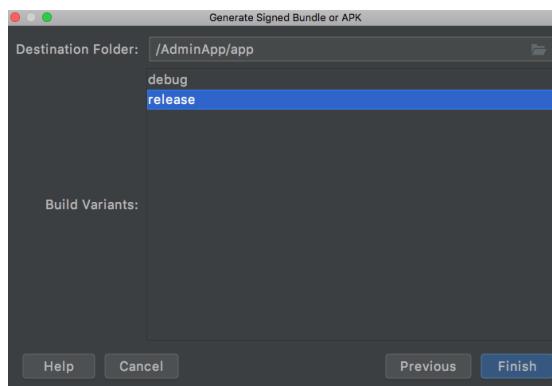


- 3.5.** Dialog ausfüllen und mit **OK** bestätigen.  
Dabei ist **Key store path** ein beliebiger Pfad in dem der Keystore gespeichert wird. **Organizational Unit** bis **State or Province** könne freigelassen werden.



**3.6.** Dialog mit **NEXT** bestätigen

**3.7.** Destination Folder angeben, als Build-Variante **release** auswählen und mit **Finish** bestätigen.



Die signierte APK wird nun erstellt und unter Destination Folder/**release/app-release.apk** gespeichert.

**4.** Installation: APK auf dem Smartphone öffnen. Es erscheint eine Warnung in der darauf hingewiesen wird, dass der Entwickler der App nicht erkannt wurde. Diese muss mit **TROTZDEM INSTALLIEREN** akzeptiert werden.

## 7 Lautsprecher und Audiomodul

**Hinweis:** Da die Kommunikation zwischen dem primären Raspberry Pi und dem sekundären Raspberry Pi, an den das Audiomodul angeschlossen ist, über das MQTT-Protokoll abgewickelt wird, muss ein MQTT-Server (auch Broker genannt) zur Steuerung der Kommunikation auf einem der Raspberry Pis installiert werden. Eine gute Installationsanleitung ist auf den folgenden Webseiten zu finden:

<https://pastebin.com/Etn59pp>

<https://www.youtube.com/watch?v=AsDHEDbyLfg>

1. Auf der Webseite

<https://learn.adafruit.com/adafruit-voice-bonnet/raspberry-pi-setup> wird vom Hersteller des im Projekt verwendeten Audiomoduls ausführlich beschrieben, wie das Audiomodul an den Raspberry Pi angeschlossen und in Betrieb genommen wird.

Im Anhang befindet sich auch die gesamte Installationsanleitung in Form einer PDF-Datei.

**Hinweis:** Der Teil ab *Python Usage* wird für die Inbetriebnahme der Software nicht benötigt, da im Steuerungssystem andere Audio-Bibliotheken verwendet werden.

2. Für die akustische Klingel ist das Modul „doorbell“ im Paket „entities“ zuständig. Damit das doorbell-Modul richtig ausgeführt wird, muss das Modul „paho-mqtt“ mit dem Befehl *sudo pip install paho-mqtt* auf dem Raspberry Pi installiert werden.
3. Das Modul „doorbell“ kann auf einem sekundären Raspberry Pi (wie das im Projekt gemacht wurde) eigenständig ablaufen. Dafür müssen lediglich die im Modulkopf angegebenen Module richtig importiert bzw. installiert werden.
4. Damit die Klingel bei Betätigung des Klingeltasters benachrichtigt wird und einen Signalton abgibt, muss das doorbell-Modul auf dem Raspberry Pi, an den die Lautsprecher angeschlossen sind, über die Konsole ausgeführt werden und im Hintergrund durchgehend ablaufen, sodass die Signale zum Klingeln jederzeit empfangen bzw. verarbeitet werden können.
5. Die globale Variable „RINGTONE\_PATH“ in der Klasse „Doorbell“ muss entspre-

chend so angepasst werden, dass sie den Pfad zur Tondatei speichert, die beim Klingeln von den Lautsprechern abgespielt werden muss. Die anderen Attribute in dieser Klasse müssen nicht angepasst werden.

6. Die Werte der globalen Variablen „BROKER“, „USERNAME“ und „PASSWORD“ in der Klasse „MQTTProtocolConfiguration“ müssen entsprechend so angepasst werden, dass sie mit der IP-Adresse und den Zugangsdaten zum oben genannten MQTT-Server übereinstimmen. Die anderen Konfigurationswerte müssen nicht angepasst werden.

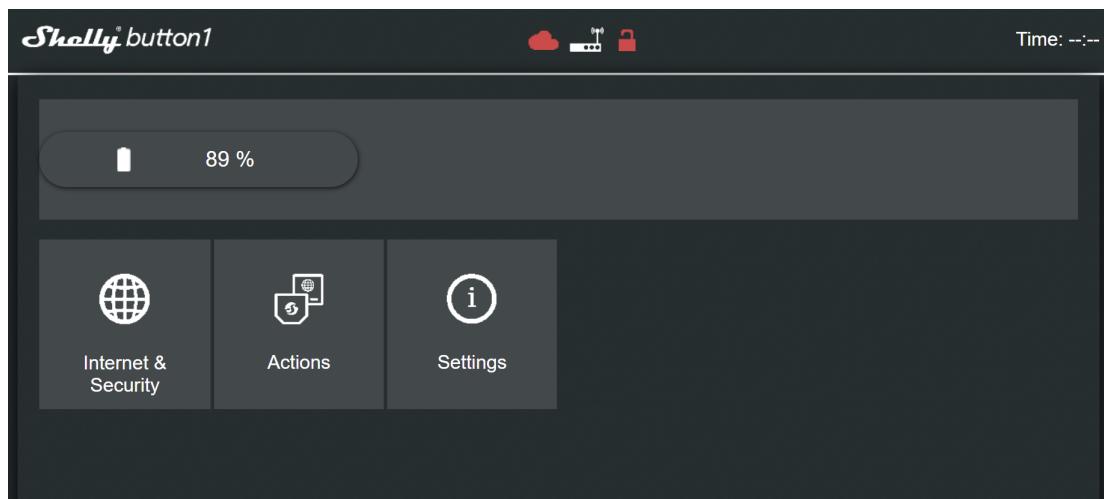
## 8 WLAN-Button

**Hinweis:** Da die Kommunikation zwischen dem primären Raspberry Pi und dem sekundären Raspberry Pi, an den das Audiomodul angeschlossen ist, über das MQTT-Protokoll abgewickelt wird, muss ein MQTT-Server (auch Broker genannt) zur Steuerung der Kommunikation auf einem der Raspberry Pis installiert werden. Eine gute Installationsanleitung ist auf den folgenden Webseiten zu finden:

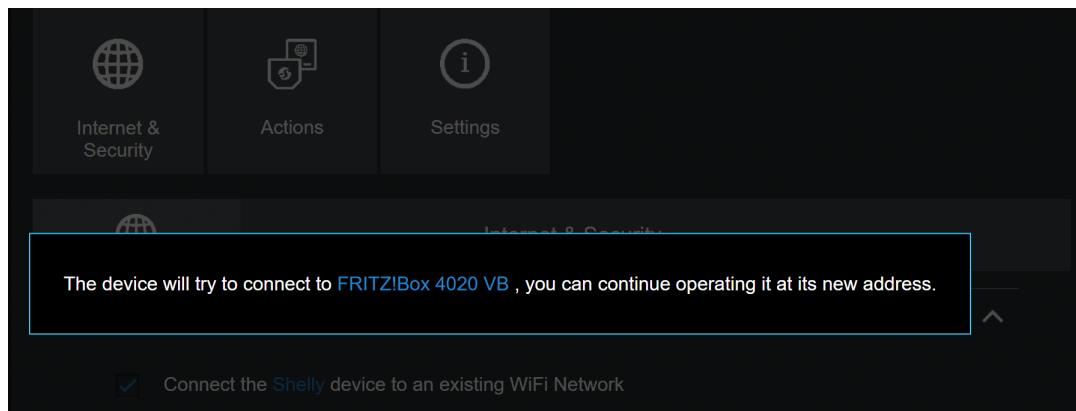
<https://pastebin.com/Etn59pp>

<https://www.youtube.com/watch?v=AsDHEDbyLfg>

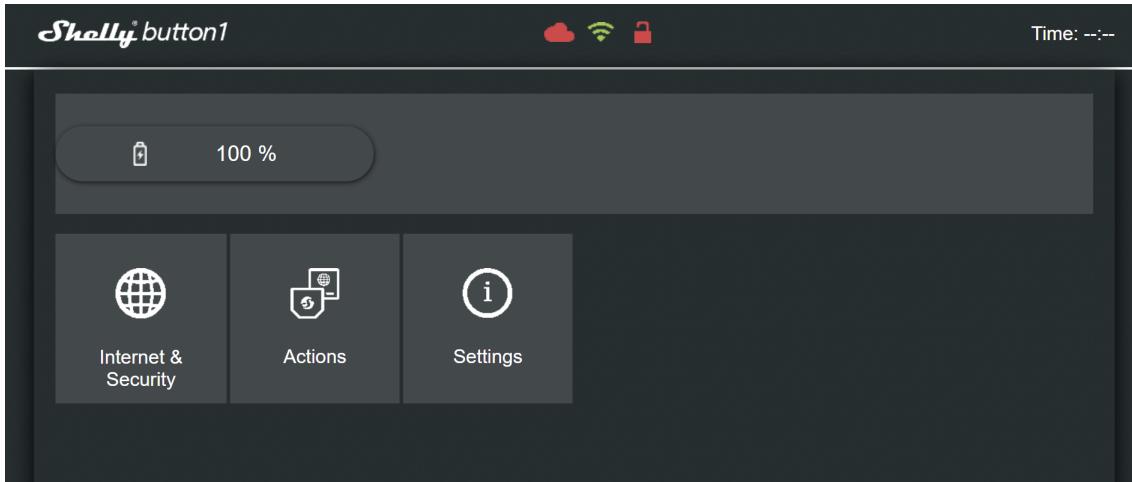
1. Der WLAN-Button muss zunächst mit dem WLAN verbunden werden, mit dem auch der Raspberry Pi zur Türsteuerung verbunden ist. Dafür muss der Deckel entfernt und die Reset-Taste hinter der Batterie für ca. 10 Sekunden gedrückt werden.
2. Nach dem Zurücksetzen erstellt der WLAN-Button automatisch einen öffentlichen WiFi-Access-Point mit einem Namen der Form *shellybutton1-35FA58*. Für die Einrichtung des Buttons muss über ein internetfähiges Endgerät eine Verbindung zu diesem Access-Point hergestellt werden.  
*Hinweis:* Es empfiehlt sich, den WLAN-Button während des Einrichtungsprozesses ans Stromnetz anzuschließen, da es manchmal zu Unterbrechungen im Access-Point gekommen ist, wenn der Button im Akkubetrieb war.
3. Die IP-Adresse 192.168.33.1 in das Adressfeld eines Browsers eingeben, um die Weboberfläche des Shelly-Buttons zu öffnen.
4. Es öffnet sich folgende Ansicht:



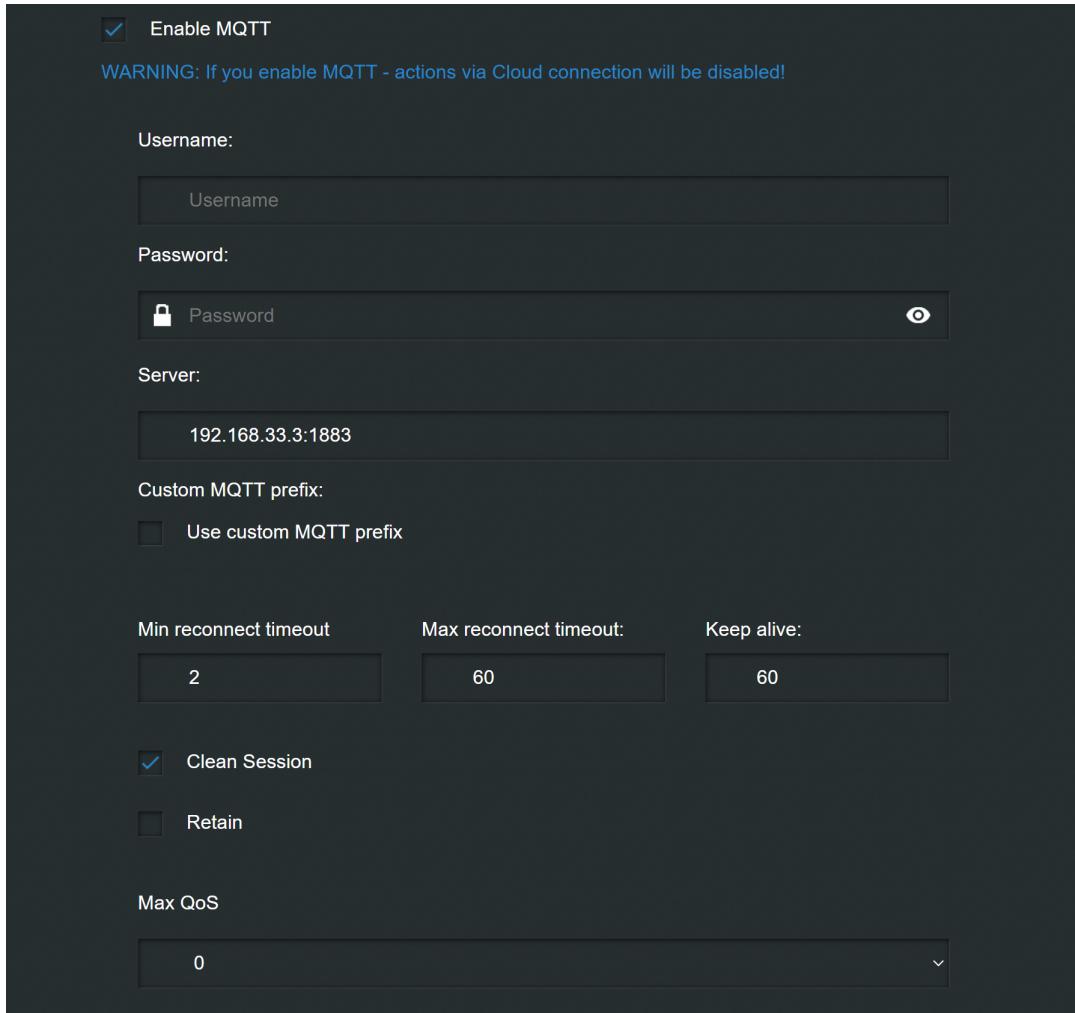
In der Rubrik *Internet & Security* -> *WIFI MODE - CLIENT* muss der WLAN-Button mit demselben WLAN verbunden werden, mit dem der Raspberry Pi verbunden ist.



5. Nun muss die IP-Adresse des WLAN-Buttons im neuen Netzwerk über einen frei wählbaren IP-Scanner (z.B. Angry IP Scanner oder Advanced IP Scanner) ermittelt werden. Nach der Eingabe der neuen IP-Adresse in den Browser erscheint folgende Seite:



6. Da die Kommunikation zwischen dem Raspberry Pi und dem WLAN-Button über das MQTT-Protokoll abgewickelt wird, muss in der Rubrik *Internet & Security* -> *ADVANCED - DEVELOPER SETTINGS* das MQTT-Protokoll aktiviert werden. Hierbei müssen die Zugangsdaten und die IP-Adresse des bereits installierten MQTT-Brokers in die entsprechenden Felder eingegeben werden.



7. Für die Verarbeitung der Signale, die vom WLAN-Button gesendet werden, ist die Klasse „DoorOpenButtonObserver“ im Paket „entities“ zuständig. Damit diese Klasse fehlerfrei funktioniert, muss das Modul *paho-mqtt* mit dem Befehl *sudo pip install paho-mqtt* auf dem Raspberry Pi installiert werden. Die Konfigurationsdaten in der Klasse „DoorOpenButtonObserver“ sind netzwerkunabhängig und müssen daher nicht angepasst werden.
8. Die Werte der globalen Variablen „BROKER“, „USERNAME“ und „PASSWORD“ in der Klasse „MQTTProtocolConfiguration“ müssen entsprechend so angepasst werden, dass sie mit der IP-Adresse und den Zugangsdaten zum oben genannten MQTT-Server übereinstimmen. Die anderen Konfigurationswerte müssen nicht angepasst werden.
9. Weitere Informationen zum Shelly-Button:

- Der Shelly-Button ist batteriebetrieben und verfügt über einen „Wach“- und einen „Schlaf“-Modus.
- Shelly Button befindet sich die meiste Zeit im „Schlaf“-Modus, wenn es mit Batteriestrom betrieben wird, um eine längere Batterielebensdauer zu gewährleisten. Bei Betätigung der Taste wird der Button kurzzeitig aktiv, sendet den gewünschten Befehl und geht in den „Schlafmodus“, um Energie zu sparen.
- Wenn das Gerät ständig mit einem Ladegerät verbunden ist, sendet es den Befehl sofort.
- Im Akkubetrieb beträgt die durchschnittliche Latenz etwa 2 Sekunden.
- Wenn das Gerät über einen USB-Anschluss mit dem Strom versorgt wird, ist es immer verbunden und es gibt keine spürbare Latenz.
- Die Reaktionszeiten des Geräts hängen von der Internetverbindung und der Signalstärke ab.
- Für weitere Informationen zur Einrichtung des Shelly Buttons 1 sei auf die entsprechende Bedienungsanleitung im Anhang sowie auf folgende Webseite vom Hersteller verwiesen:  
<https://shelly-api-docs.shelly.cloud/gen1/#shelly-button1-mqtt>.