Fachhochschule
Dortmund

Bachelorarbeit

Oleksii Baida
Matrikelnummer 7210384

Sicherheits- & Steuerungssytem für das Haus

Bericht

14. Januar 2025

Inhaltsverzeichnis

1 Einleitung

In einer Welt, die zunehmend von vernetzten Geräten und dem Internet der Dinge geprägt ist, wird die Entwicklung effizienter und benutzerfreundlicher Systeme zur Steuerung und Überwachung von Gebäuden immer relevanter. Im Jahr 2024 wurde in Deutschland die Anzahl von über 19 Millionen Haushalten, die ein oder mehrere smarte Geräte besaßen, verzeichnet. Es wird prognostiziert, dass sich diese Zahl innerhalb der nächsten drei Jahre verdoppeln wird [4].

Moderne Steuerungs- und Sicherheitssysteme tragen zur Effizienzsteigerung und Ressourcenschonung bei. Laut Günther Ohland, Vorstandsmitglied des Branchenverbands SSmarthome Initiative Deutschland", ermöglichen diese Systeme eine Reduktion des Heizenergieverbrauchs um 20 bis 30 Prozent [5]. Die Kosten für die smarte Technik rechnen sich in der Regel nach zwei Jahren. Die Systeme übernehmen ein Teil der täglichen Aufgaben, wie das Ein- und Ausschalten des Lichts, die Reglung der Raumtemperatur oder das Aufräumen des Hauses etc. Der Aufgabenbereich der Systeme ist dabei nur nach den Bedürfnissen der Benutzerinnen und Benutzer abgegrenzt.

Im Rahmen meiner Bachelorarbeit wird ein System zur Steuerung und Überwachung des Hauses entwickelt. Das Ziel dieser Arbeit ist die Erstellung einer Schnittstelle, die die Interaktion des Benutzers mit den Geräten in seinem Haushalt ermöglicht und den Benutzer über gefährliche Vorgänge in seinem Haus informiert.

Im Rahmen der Entwicklung dieses Systems wurden die aktuellen Technologien zur Erstellung eines Webinterfaces und zur Kommunikation zwischen den Geräten eingesetzt. **TODO Kurz erklären was in Kapitel 2,3,4,5 ... erklärt wird**

2 Grundlagen & Theorie

In diesem Abschnitt erfolgt die detaillierte Darstellung der technischen Informationen zu den verwendeten Komponenten und Technologien.

2.1 Hardware

2.1.1 Arduino

2.1.2 ESP8266

2.1.3 Raspberry Pi

2.1.4 ESP32

2.1.5 M5Stick

2.2 Kommunikationsprotokolle

2.2.1 HTTP

2.2.2 MQTT

2.3 Software

2.3.1 Asyncio

2.3.2 FastAPI

2.3.3 SQLAlchemy

2.3.4 Uvicorn

2.3.5 HTML & TailwindCSS

2.3.6 Javascript

2.3.7 WebSocket

2.3.8 Linux-Pakete für Raspberry Pi

3 Konzeption das Systems

Im Rahmen dieses Projektes wurde ein System entwickelt, welches die Funktionalitäten eines Kontroll- und Verwaltungssystems mit denen eines IoT-Systems vereint. Die Integration von Sensordaten und Benutzerinteraktionen stellt einen wesentlichen Aspekt des Systems dar. Die Realisierung erfolgt durch die Kombination verschiedener Technologien und Teilsysteme, darunter eine Webanwendung auf FastAPI, eine MQTT-Kommunikationsschicht und verschiedene Aktoren und Sensoren, die auf Arduino- oder ESP-Module basieren.

3.1 Komponenten des Systems

Das entwickelte System basiert auf einer modularen Architektur, die mehrere Komponenten integriert. Jede dieser Komponenten erfüllt eine spezifische Rolle im Gesamtsystem:

- **Raspberry Pi:** Der zentrale Server, der das lokale WLAN-Netzwerk bereitstellt, den MQTT-Broker hostet und die Webanwendung ausführt.
- **Arduino:** Ausgestattet mit mehreren Sensoren, die Gefahren wie Feuer und Gas erkennen und den Zugang zum Haus sichern. Entsprechende Meldungen werden an den Server gesendet.
- **M5Stick:** Angeschlossen an die Temperatur- und Lichtsensoren und sendet die aufgezeichneten Daten auf den Server.
- **ESP32:** TTTOOOOODDDDOOOO
- **Webserver:** Eine auf FastAPI basierende RESTful API, die Benutzern den Zugriff auf das System und die Steuerung von Geräten ermöglicht.
- **Datenbank:** Eine lokale Datenbank zur Speicherung von Benutzerinformationen und Gerätekonfigurationen.
- **Web-Anwendung:** Eine browserbasierte Benutzeroberfläche, die den Benutzern eine intuitive Steuerung und Visualisierung der Daten ermöglicht.

3.2 Architektur und Datenfluss

Der Raspberry Pi dient als zentraler Server des Systems. Der Minicomputer stellt ein WLAN-Netzwerk zur Verfügung und hostet den Webserver mit der Datenbank sowie den MQTT-Broker. Alle Benutzerinteraktionen, Sensordaten, Datenflüsse und Datenverarbeitungen finden auf dem Server statt. Somit ist das System stark zentralisiert. Das System ist somit stark zentralisiert und arbeitet nur lokal. Das bedeutet, dass sich alle Benutzer in einem lokalen Netzwerk mit dem Raspberry Pi befinden müssen.

Die Geräte verbinden sich mit dem WLAN, das vom Raspberry Pi zur Verfügung gestellt wird. Die Sensoren senden ihre Daten an den MQTT-Broker, der auf dem Raspberry Pi läuft. Die Aktoren abonnieren die Command-Topics mit der entsprechenden "Geräte-ID".

Im Kern des Systems befindet sich ein Webserver, der auf dem Raspberry Pi ausgeführt wird. Dieser Webserver stellt eine RESTful-API zur Verfügung. Für den Zugriff zu der API wurde eine Webseite aufgebaut. Die API bietet folgende Funktionen an:

- Authentifizierung und Autorisierung der Benutzer
- Verwaltung der Gerätekonfigurationen und Benutzerprofile

- Bereitstellung von Endpunkten zur Abfrage und Steuerung von IoT-Geräten
- Bidirektionale Kommunikation mit den IoT-Geräten

Die Benutzerdaten und Konfigurationen werden in der lokalen Datenbank auf dem Server gespeichert.

4 Implementierung und praktische Umsetzung

4.1 Einrichtung der Hardwarekomponenten

4.1.1 Arduino

4.1.2 ESP8266

4.1.3 Raspberry Pi

4.1.4 M5Stick

4.1.5 ESP32

4.2 Softwareentwicklung

4.2.1 Webserver

4.3 Datenbank

4.4 Frontend

4.5 Integration der Komponenten

MQtt client, erhalten Daten von Broker etc

5 Ergebnisse und Diskussion

6 Quellen

Literatur

[1] O. Baida, *Anbindung der Sensoren und Aktoren an den Arduino zur Realisierung eines Sicherheitssystems*, Projektarbeit 1, 2024.

[2] O. Baida, Projektordner für PA2: https://github.com/oleksiibaida/PA2.git

[3] O. Baida, Demo-Video: https://youtu.be/-Ul_ye1KLy0

[4] Statista, *„Smart Home - Anzahl der Haushalte in Deutschland 2028“*, Zugegriffen: 13. Januar 2025. [Online]. Verfügbar unter https://de.statista.com/prognosen/885611/anzahl-der-smart-home-haushalte-in-deutschland

[5] J. Breithut, *„Strom und Heizung: Wann ein Smart Home wirklich beim Energiesparen hilft“*, Der Spiegel, 17. Juli 2022. Zugegriffen: 13. Januar 2025. [Online]. Verfügbar unter: https://www.spiegel.de/netzwelt/gadgets/strom-und-heizung-wann-ein-smart-home-wirklich-beim-energiesparen-hilft-a-ffb4b710-

Links zur verwendeten Hardware:

[6] Arduino.cc, *Arduino UNO*, https://docs.arduino.cc/hardware/uno-rev3/

[7] Raspberry Pi Foundation, *Raspberry Pi 1 B+*, https://www.raspberrypi.com/products/raspberry-pi-1-model-b-plus/

[8] Espressif, *ESP8266*, https://www.espressif.com/, https://www.electronicwings.com/sensors-modules/esp8266-wifi-module

Links zur verwendeten Software:

[9] Dr Andy Stanford-Clark, Arlen Nipper, *Message Queuing Telemetry Transport*, https://mqtt.org/

[10] Guido van Rossum, Python Software Foundation, *Python*, https://www.python.org/

[11] Telegram FZ-LLC, *Telegram Messenger*, https://github.com//telegramdesktop/tdesktop

Linux-Packete:

[12] Jouni Malinen, *hostapd*, https://w1.fi/hostapd/, Zugriff am: 19. September 2024.

[13] Simon Kelley, *dnsmasq*, https://dnsmasq.org/doc.html, Zugriff am: 20. September 2024.

[14] Eclipse Foundation, *Eclipse Mosquitto*, https://mosquitto.org/

ESP- und Arduino-Bibliotheken

[15] Knolleary, *PubSubClient*, https://pubsubclient.knolleary.net/, Zugriff am: 21. Oktober 2024.

[16] ESPWIFI.h, https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html

[17] EEPROM.h, https://docs.arduino.cc/learn/built-in-libraries/eeprom/

[18] Keypad.h https://docs.arduino.cc/libraries/keypad/

[19] R. Scholz, *Syncloop*, Persönliche Mitteilungen

Python-Bibliotheken

[20] Pierre Fersing, Roger Light *paho-mqtt*, https://pypi.org/project/paho-mqtt/, Zugriff am: 21. Oktober 2024.

[21] Open Source, *python-telegram-bot*, https://docs.python-telegram-bot.org/en/v21.6/

[22] Python Software Foundation, *json*, https://docs.python.org/3/library/json.html

[23] Python Software Foundation, *threading*, https://docs.python.org/3/library/threading.html

[24] Python Software Foundation, *queue*, https://docs.python.org/3/library/queue.html

[25] Gerhard Häring, *sqlite3*, https://docs.python.org/3/library/sqlite3.html

[26] Lawrence Hudson, *pyzbar*, https://github.com/NaturalHistoryMuseum/pyzbar/

[27] Intel, *OpenCV*, https://github.com/opencv/opencv-python

[28] Aio-Libs, *aiohttp*, https://github.com/aio-libs/aiohttp

Abbildungsverzeichnis

Tabellenverzeichnis

Programmcode