

# **FH Aachen**

**Fachbereich Elektrotechnik und Informationstechnik**

Studiengang Informatik

Bachelorarbeit

## **Integration eines eingebetteten Systems in eine Cloud-Infrastruktur am Beispiel eines autonomen Spielfelds**

vorgelegt von

**Marcel Werner Heinrich Friedrich Ochsendorf**

Matrikel-Nr. **3120232**

Referent:

Prof. Dr.-Ing. Thomas Dey

Korreferent:

Prof. Dr. Andreas Claßen

Datum:

25. Mai 2021

# **1 Erklärung**

Hiermit erkläre ich, dass ich die vorliegende Arbeit eigenständig und ohne fremde Hilfe angefertigt habe. Textpassagen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Aachen, den 25.05.2021 \_\_\_\_\_

## **2 Abstract**

Die Kurzfassung gibt auf ein bis zwei Seiten die wesentlichen Inhalte und Ergebnisse der Abschlussarbeit wieder.

Sie gliedert sich inhaltlich in

- das behandelte Gebiet,
- das Zieler Arbeit,
- die Untersuchungsmethode,
- die Ergebnisse und
- die Schlussfolgerungen.

Die Kurzfassung enthält keine Schlussfolgerungen oder Bewertungen, die über die Inhalte der Kapitel der Arbeit hinausgehen.

Alle Aussagen der Kurzfassung finden sich in ausführlicher Form in der Arbeit wieder.  
Die Kurzfassung erhält keine Kapitelnummer.

# Inhalt

1	Erklärung	i
2	Abstract	ii
0.3	Einleitung	1
0.3.1	Motivation	1
0.3.2	Zielsetzung	1
0.3.3	Methodik	2
0.4	Analyse bestehender Systeme und Machbarkeitsanalyse	2
0.4.1	Existierende Systeme im Vergleich	2
0.4.2	User Experience	6
0.4.3	Anforderungsanalyse	7
0.4.4	Machbarkeitsanalyse	7
0.5	Entwicklung der Cloud Infrastruktur	7
0.5.1	API Design	8
0.5.2	Service Architektur	9
0.5.3	Entwicklung Webclient	12
0.5.4	Sicherheit	14
0.6	Grundlegende Verifikation der ausgewählten Technologien	14
0.6.1	Erprobung Buildroot-Framework	14
0.6.2	Verifikation NFC Technologie	14
0.6.3	Schrittmotor / Schrittmotorsteuerung	14
0.6.4	3D Druck für den mechanischen Aufbau	16
0.7	Erstellung erster Prototyp	16
0.7.1	Technologieauswahl für ersten Prototypen	16
0.7.2	CAD Design	16
0.7.3	Schaltungsentwurf Motorsteuerung	16
0.7.4	Anpassung Controller Software	17
0.8	Erstellung zweiter Prototyp	17
0.9	Fazit	17
0.9.1	Ausblick	17
	Literaturverzeichnis	18

<b>Abbildungsverzeichnis</b>	<b>21</b>
<b>Tabellenverzeichnis</b>	<b>22</b>
.1 Anhang . . . . .	23

## 0.3 Einleitung

### 0.3.1 Motivation

- Beginn: Er zieht die Aufmerksamkeit des Lesers durch die Schilderung des Ereignisses auf sich, das zu dem Problem geführt hat.
- Hintergrundinformationen (Herstellung des Kontexts): Gehe tiefer auf das Ereignis ein, indem du mehr Informationen über es vermittelst und dabei auch den Rahmen deiner Forschung skizzierst.
- Brücke zur Problemstellung: Erläutere, inwiefern es sich hierbei um ein Problem handelt, und schlage somit die Brücke zur Problemstellung, die deiner Untersuchung zu Grunde liegt.

### 0.3.2 Zielsetzung

Das Ziel dieser Arbeit ist es, einen autonomen Schachtisch, welcher in der Lage ist Schachfiguren autonom zu bewegen und auf Benutzerinteraktion zu reagieren.

Der Schwerpunkt liegt dabei insbesondere auf der Programmierung des eingebetteten Systems und dem Zusammenspiel dieses mit einem aus dem Internet erreichbaren Servers, welcher als Vermittlungsstelle zwischen verschiedenen Schachtischen dient.

Dieses besteht zum einem aus der Positionserkennung und Steuerung der Hardwarekomponenten (Schachfiguren) und zum anderen aus der Kommunikation zwischen dem Tisch selbst und einem in einer Cloud befindlichen Server.

Mittels der Programmierung werden diverse Technologien von verschiedenen Einzelsystemen zu einem Gesamtprodukt zusammengesetzt. Zu diesen Einzelsystemen gehören:

- Programmierung der Motorsteuerung, HMI (zB. Qt oder simple Buttons), NFC-Tag Erkennung
- Programmierung eines Wrappers für die Kommunikation mit einer Cloud (AWS)
- State-Maschine und Implementierung der Spielflusssteuerung
- Backend mit Datenbankanbindung zwischen Server und Embedded-System

- Verwendung eines CI/CD Systems zum automatisierten Bauen der Linux-Images für das Embedded-System

### **0.3.3 Methodik**

Im ersten Abschnitt werden die zum Zeitpunkt existierenden Ansätze und deren Umsetzung beleuchtet. Anschliessend werden die zuvor verwendeten Technologien betrachtet, welche bei den beiden darauffolgenden Prototypen verwendet wurden. Im darauffolgenden Abschnitt wird die Cloud-Infrastruktur thematisiert, welche für eine Kommunikation zwischen den Prototypen entscheidend ist.

Abschliessend wir ein Ausblick auf

- ausblick weitere features

## **0.4 Analyse bestehender Systeme und Machbarkeitsanalyse**

### **0.4.1 Existierende Systeme im Vergleich**

- nieschenprodukt jedoch einige Projekte im OpenSource bereich verfügbar
- ein kommerzieller hersteller

#### **Kommerzielle Produkte**

Tabelle 0.1: Auflistung kommerzieller autonomer Schachtische

	Square Off - Kingdom [5]	Square Off - Grand Kingdom [4]	DGT Smart Board [2]	DGT Wenge [1]
Erkennung	nein (Manuell per	nein (Manuell per Ausgangsposition)	ja	ja
Figurstellung	Ausgangsposition)			
Abmessungen (LxBxH)	486mm x 486mm x 75mm	671mm x 486mm x 75mm	540mm x 540mm x 20mm	540mm x 540mm x 20mm
Konnektivität	Bluetooth	Bluetooth	Seriell	Bluetooth
Automatisches Bewegen der Figuren	ja	ja	nein	nein
Spiel	ja	ja	ja	ja
Livestream				
Cloud anbindung (online Spiele)	ja (Mobiltelefon + App)	ja (Mobiltelefon + App)	ja (PC + App)	ja (PC + App)
Parkposition für ausgeschiede- ne	nein	ja	nein	nein
Figuren				
Stand-Alone	nein	nein (Mobiltelefon erforderlich)	nein (PC erforderlich)	nein (PC erforderlich)
Funktionalität	(Mobiltelefon erforderlich)			
Besonderheiten	Akku für 30 Spiele	Akku für 15 Spiele	-	-

Bei den DGT-Schachbrettern ist zu beachten, dass diese die Schachfiguren nicht autonom bewegen können. Sie wurden jedoch in die Liste aufgenommen, da diese einen Teil der Funktionalitäten der Square Off Schachbrettern abdecken und lediglich die automatische Bewegung der Schachfiguren fehlt. Die DGT-Bretter können die Position der Figuren erkennen und ermöglichen so auch Spiele über das Internet; diese können sie auch als Livestream anbieten. Bei Schachturnieren werden diese für die Übertragung der Partien sowie die Aufzeichnung der Spielzüge verwendet und bieten Support für

den Anschluss von weiterer Peripherien wie z.B. Schachuhren.

Somit gibt es zum Zeitpunkt der Recherche nur einen Hersteller von autonomen Schachbrettern, welcher auch die Figuren bewegen kann.

### Open-Source Projekte

Bei allen Open-Source Projekten wurden die Features anhand der Beschreibung und der aktuellen Software extrahiert. Besonders bei work-in-progress Projekten können sich die Features noch verändern und so weitere Funktionalitäten hinzugefügt werden.

Zusätzlich zu den genannten Projekten sind weitere derartige Projekte verfügbar; in der Tabelle wurde nur jene aufgelistet, welche sich von anderen Projekten in mindestens einem Feature unterscheiden.

Auch existieren weitere Abwandlungen von autonomen Schachbrettern, bei welchem die Figuren von oberhalb des Spielbretts gegriffen bzw. bewegt werden. In einigen Projekten wird dies mittels eines Industrie-Roboters [10] oder eines modifizierten 3D-Druckers[7] realisiert. Diese wurden hier aufgrund der Mechanik, welche über dem Spielbrett montiert werden muss, nicht berücksichtigt.

Tabelle 0.2: Auflistung von Open-Source Schachtisch Projekten

	Automated Chess Board (Michael Guerero) [3]	Automated Chess Board (Akash Ravichandran) [8]	DIY Super Smart Chessboard [6]
Erkennung	nein (Manuell per Ausgangsposition)	ja (Kamera / OpenCV)	nein
Figurstellung			
Abmessungen (LxBxH)	keine Angabe	keine Angabe	450mm x 300mm x 50mm
Konnektivität	Universal Serial Bus (USB)	Wireless Local Area Network (wlan)	wlan
Automatisches Bewegen der Figuren	ja	ja	nein
Spiel Livestream	nein	nein	nein
Cloud anbindung (online Spiele)	nein	nein	ja

## Inhalt

---

	Automated Chess Board (Michael Guerero) [3]	Automated Chess Board (Akash Ravichandran) [8]	DIY Super Smart Chessboard [6]
Parkposition für ausgeschiedene Figuren	nein	nein	nein
Stand-Alone Funktionalität	nein (PC erforderlich)	ja	ja
Besonderheiten	-	Sprachsteuerung (Amazon Alexa)	Zuganzeige über LED Matrix
Lizenz	General Public License (GPL) 3+	GPL	-

In den bestehenden Projekten ist zu erkennen, dass ein autonomer Schachtisch sehr einfach und mit einfachsten Mittel konstruiert werden kann. Hierbei fehlen in der Regel einige Features, wie das automatische Erkennen von Figuren oder das Spielen über das Internet.

Einige Projekte setzen dabei auf eingebettete Systeme, welche direkt im Schachtisch montiert sind, andere hingegen nutzen einen externen PC, welcher die Steuerbefehle an die Elektronik sendet.

Bei der Konstruktion der Mechanik und der Methode mit welcher die Figuren über das Feld bewegt werden ähneln sich jedoch die meisten dieser Projekte. Hier wird in der Regel eine einfache X und Y-Achse verwendet, welche von zwei Schrittmotoren bewegt werden. Die Schachfiguren werden dabei mittels eines Elektromagneten über die Oberseite gezogen. Hierbei ist ein Magnet oder eine kleine Metallplatte in den Fuß der Figuren eingelassen worden.

Die Erkennung der Schachfiguren ist augenscheinlich die schwierigste Aufgabe. Hier wurde in der Mehrzahl der Projekte eine Kamera im Zusammenspiel mit einer auf OpenCV basierenden Figur-Erkennung verwendet. Diese Variante ist je nach Implementierung des Vision-Algorithmus fehleranfälliger bei sich ändernden Lichtverhältnissen, auch muss die Kamera oberhalb der Schachfiguren platziert werden, wenn kein transparentes Schachfeld verwendet werden soll.

Eine weitere Alternative ist die Verwendung einer Matrix aus Reed-Schaltern oder Halleffekt-Sensoren. Diese werden in einer 8x8 Matrix Konfiguration unterhalb der Platte

montiert und reagieren auf die Magnete in den Figuren. So ist es möglich zu erkennen, welches der Schachfelder belegt ist, jedoch nicht konkret von welchem Figurtypen. Dieses Problem wird durch eine definierte Ausgangsstellung beim Spielstart gelöst. Nach jedem Zug durch den Spieler und der dadurch resultierenden Änderungen in der Figurpositionen in der Matrix können die neuen Figurstellungen berechnet werden.

### 0.4.2 User Experience

Ein wichtiger Aspekt bei diesem Projekt stellt die User-Experience dar. Diese beschreibt die Ergonomie der Mensch-Maschine-Interaktion und wird durch die DIN 9241[9] beschrieben. Hierbei geht es primär um das Erlebnis, welches der Benutzer bei dem Verwenden eines Produktes erlebt und welche Erwartungen der Benutzer an die Verwendung des Produktes hat.

Bei dem autonomen Schachtisch, soll der Benutzer eine ähnlich einfache Erfahrung erleben, wie bei einer Schachpartie mit einem menschlichen Gegenspieler. Der Benutzer soll direkt nach dem Einschalten des Tisches und dem Aufstellen der Figuren in der Lage sein, mit dem Spiel beginnen zu können. Dies soll wie ein reguläres Schachspiel ablaufen; der Spieler vor dem Tisch soll die Figuren mit der Hand bewegen können und der Tisch soll den Gegenspieler darstellen. Dieser bewegt die Figuren der Gegenseite.

Nach Beendigung einer Partie, soll das Spielbrett wieder in die Ausgangssituation gebracht werden; dies kann zum einem vom Tisch selbst oder vom Benutzer manuell geschehen. Danach ist der Tisch für die nächste Partie bereit, welche einfach per Knopfdruck gestartet werden können sollte.

Dies soll auf für abgebrochene Spiele gelten, welche von Benutzer oder durch das System abgebrochen werden. Hierbei soll das Schachbrett sich ebenfalls selbstständig zurücksetzen können.

Ein weiter Punkt welcher bei der User-Experience beachtet werden soll, ist die zeitliche Konstante. Ein Spiel auf einem normalen Schachspiel hat je nach Spielart kein Zeitlimit, dies kann für das gesamte Spiel gelten oder auch für die Zeit zwischen einzelnen Zügen. Der autonome Schachtisch soll es dem Spieler z.B. ermöglichen ein Spiel am Morgen zu beginnen und dieses erst am nächsten Tag fortzusetzen.

Auch muss sich hier die Frage gestellt werden, was mit den ausgeschiedenen Figuren geschieht. Bei den autonomen Schachbrettern von Square Off[4], werden die Figuren an

die Seite auf vordefinierte Felder bewegt und können so wieder bei der nächsten Partie vom System aufgestellt werden. Viele andere Projekte schieben die Figuren auf dem Feld heraus, können diese aber im Anschluss nicht mehr gezielt in das Feld zurückholen. So muss diese Aufgabe vom Benutzer geschehen. Auch wir diese Funktionalität von einigen Projekten nicht abgedeckt und der Benutzer muss die Figuren selbstständig vom Feld entfernen.

### 0.4.3 Anforderungsanalyse

- komplettes vollwertiges Produkt
- alle key requirements welcher der tisch haben soll
- als tabellen

### 0.4.4 Machbarkeitsanalyse

welche technologien werden benötigt \* software architektur anforderungen \* hardware anforderungen \* grosse \* wiederholgenauigkeit \* lautstärke \*

#### Technologien im Makerspace

stehen diese im makerspace zur verfüfung

## 0.5 Entwicklung der Cloud Infrastruktur

Die erste Phase der Entwicklung des Systems bestand in der Entwicklung der Cloud-Infrastruktur und der darauf laufenden Services. Hierbei stellt die “Cloud”, einen Server dar, welcher aus dem Internet über eine feste IPv4 und eine IPv6 verfügt und frei konfiguriert werden kann.

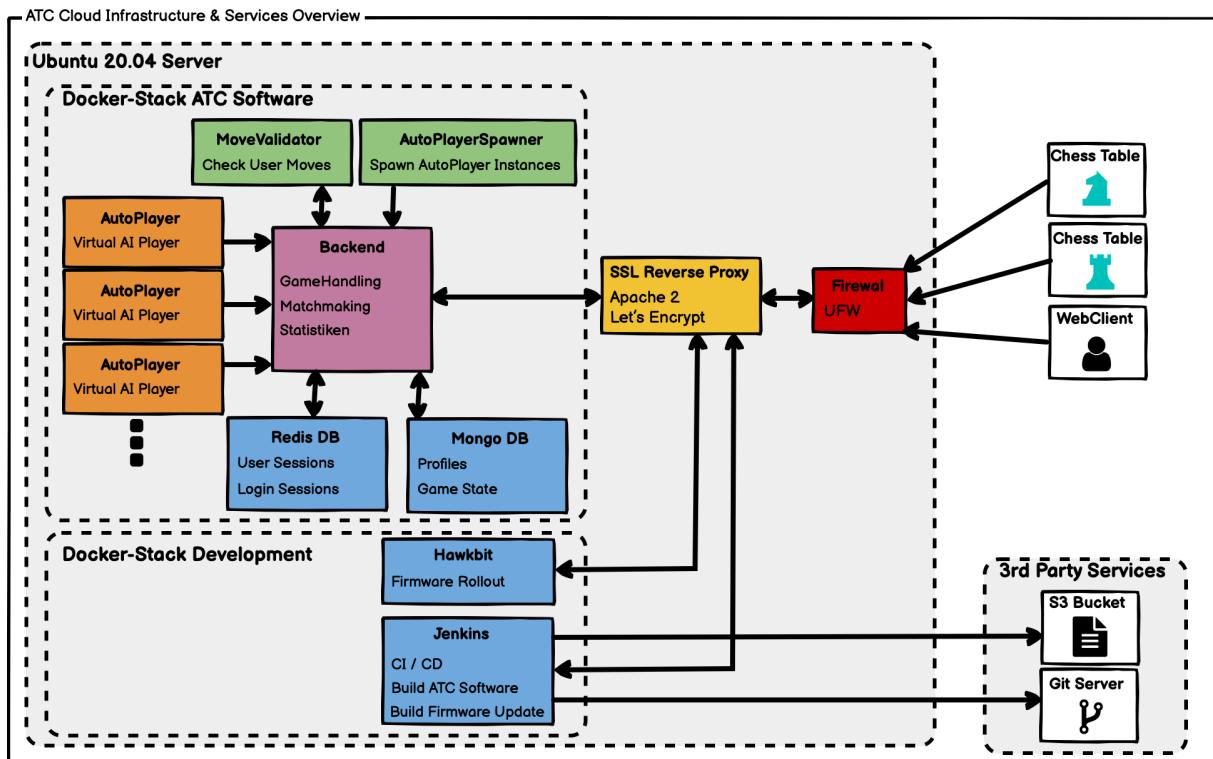


Bild 0-1: Gesamtübersicht der verwendeten Cloud-Infrastruktur

## 0.5.1 API Design

Das System soll so ausgelegt werden, dass es im späteren Zeitpunkt mit verschiedenen Client-Devices mit diesem kommunizieren können. Dazu zählen zum einen der autonome Schachtisch, aber z.B. auch einen Web-Client, welcher die Funktionalität eines Schachtisch im Browser abbilden kann. Hierzu muss das System eine einheitliche REST bereitstellen.

Eine RESTful API bezeichnet eine API welche HTTP-Requests verwendet um auf Daten zugreifen zu können.

- grafik
- 5 requirements

Die RESTful API stellt verschiedene Ressourcen bereit, welche durch eine URI eindeutig identifizierbar sind. Auf diese können mittels verschiedenster HTTP Anfragemethoden (GET, POST, PUT, DELETE) zugegriffen werden. Jeder dieser Methoden stellt einen anderen Zugriff auf die Ressource dar und beeinflusst somit das Verhalten und die Rückantwort dieser.



Bild 0-2: Aufbau einer URI

Eine URI besteht dabei aus mehreren Teilen. Das Schema gibt an wie die nachfolgenden Teile interpretiert werden sollen. Dabei wird bei einer RESTful Schnittstelle typischerweise das Hypertext Transfer Protocol (HTTP) Protokoll, sowie Hypertext Transfer Protocol Secure (HTTPS) verwendet. Dabei steht HTTPS für eine verschlüsselte Verbindung. Des Weiteren gibt es viele andere Schemata, wie z.B. File Transfer Protocol (FTP) welches

Somit stellt die RESTful API eine Interoperabilität zwischen verschiedenen Anwendungen und Systemen bereit, welche durch ein Netzwerk miteinander verbunden sind. Dieser Ansatz ist somit geeignet um die verschiedenen Client Systeme (Schachtisch, Webclient) eine Kommunikation mit dem Server zu erlauben.

### 0.5.2 Service Architektur

- was ist ein Service
- microservice ansatz
- Kapselung der Schachspiel spezifischen funktionalitäten
- verwendung von NoSQL Datenbanken somit müssen tabellen nicht speziell auf Schach spezifische felder ausgelegt sein
- stateless Diese stellen alle wichtigen Funktionen zum Betrieb des autonomen Schachtisches zur Verfügung.

### Vorüberlegungen

- welche funktionalitäten müssen abgedeckt werden
- client aktivitendiagramm

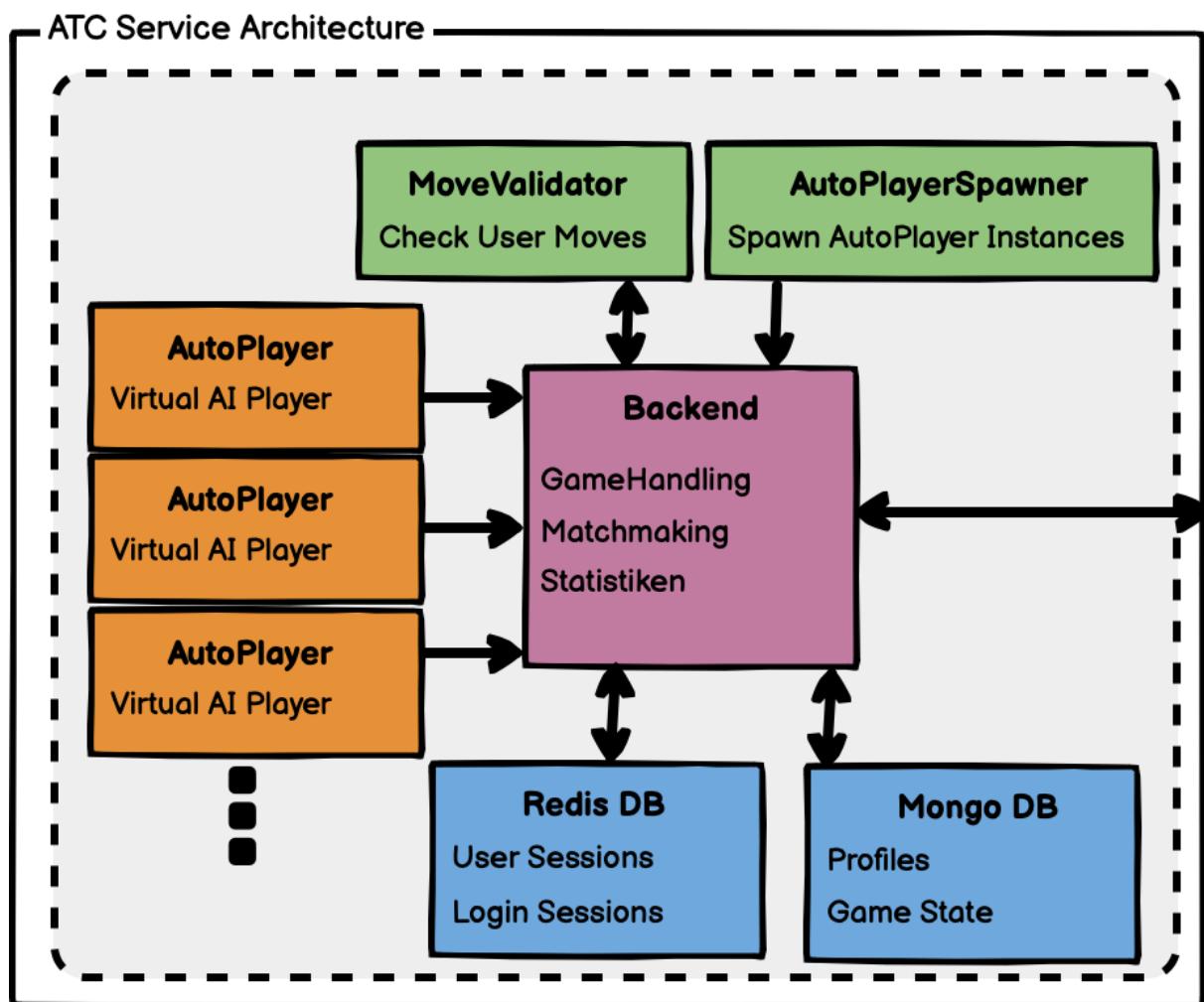


Bild 0-3: Aufbau der Service Architecture

### Backend

- matchmaking schachlogik
- zentraler zugriffspunkt auf das System und stellt diese abi bereit
- stellt spielerprofile aus datenbanken bereit bereit
- authentifizierung der clients und deren sessions
- weiterleitung der von spielerinteraktionen an move validator

### MoveValidator

Der MoveValidator-Service bildet im System die eigentliche Schachlogik ab. Die Aufgabe ist es, die vom Benutzer eingegebenen Züge auf Richtigkeit zu überprüfen und auf daraufhin neuen Spiel-Status zurückzugeben. Dazu zählen unter anderem das neue Schachbrett und ob ein Spieler gewonnen oder verloren hat.

Bevor ein Spiel begonnen wird, generiert der MoveValidator das initiale Spielfeld und bestimmt den Spieler, welcher als erstes am Zug ist.

Der Backend-Service fragt einen neuen Spiel an, oder übergibt einen Schachzug inkl. des Spielbretts an den Service. Der Response wird dann vom Backend in der Datenbank gespeichert und weiter an die Client-Devices verteilt.

Mit diesem Design ist es möglich auch andere Spielarten im System zu implementieren, nur hier die initialen Spielfelder generiert werden und Züge der Spieler validiert werden.

Für ein anderes Spiel müssen drei

- generiter neues brett
- python chess packge welche pesudo legal moves generiert
- beispiel Requests

### AutoPlayer

- stellt schachai dar
- agiert als selbstäniger spieler



Bild 0-4: Webclient: Spielansicht

- wenn nicht genügend menschlich spieler vorhanden sind

### 0.5.3 Entwicklung Webclient

Der Webclient wurde primär dazu entwickelt um das System während der Entwicklung zu testen. Dieser simuliert einen autonomen Schachtisch und verwendet dabei die gleichen HTTP Requests. Dieser wurde dabei komplett in JavaScript (JS) umgesetzt im Zusammenspiel mit Hypertext Markup Language (HTML) und Cascading Style Sheets (CSS) und ist somit komplett im Browser lauffähig.

Ausgeliefert werden die statischen Dateien zur Einfachheit durch den Backend-Service.  
\* express router \* public folder

Während der Implementierung wurde der Webclient weiter ausgebaut und es wurde weitere Features ergänzt. Dazu zählt zum einen eine Übersicht über vergangene und aktuell laufende Spiele. In dieser können Spiele Zug um Zug nachvollzogen werden und weitere Information über den Spielstatus angezeigt werden. Auch ist es möglich

## Inhalt

The screenshot shows the 'LAST GAMES - HISTORY' section with one entry:

#	CREATED	PLAYER WHITE	PLAYER BLACK	MOVE COUNT	GAME_STATE	VIEW
1	2021-04-20_19:18:40	AI_1e2ab	VIRT_legawa_efreron_193	39	RUNNING	<button>SHOW DETAILS</button>

**GAME INFORMATION** 25a34660-a1fc-11eb-9cfc-ff3c5889598c

**CURRENT BOARD**

Bild 0-5: Webclient: Statistiken

aktuell laufende Spiele in Echtzeit anzeigen zu lassen, somit wurde eine Livestream-Funktionalität implementiert.

- techstack js
- backend zu testen
- menschliche spieler zu simulieren
- während der entwicklungsphase des tisches gezielt spiele simulieren zu können
- liefert auch statistiken
- wird zur einfachheit direkt aus dem abckend heraus ausgeliefert da nur statisches html/js/class

## 0.5.4 Sicherheit

- authentifizierung
- https only
- zertifikate auf clientseite generiert jedoch nicht abgefragt

# 0.6 Grundlegende Verifikation der ausgewählten Technologien

## 0.6.1 Erprobung Buildroot-Framework

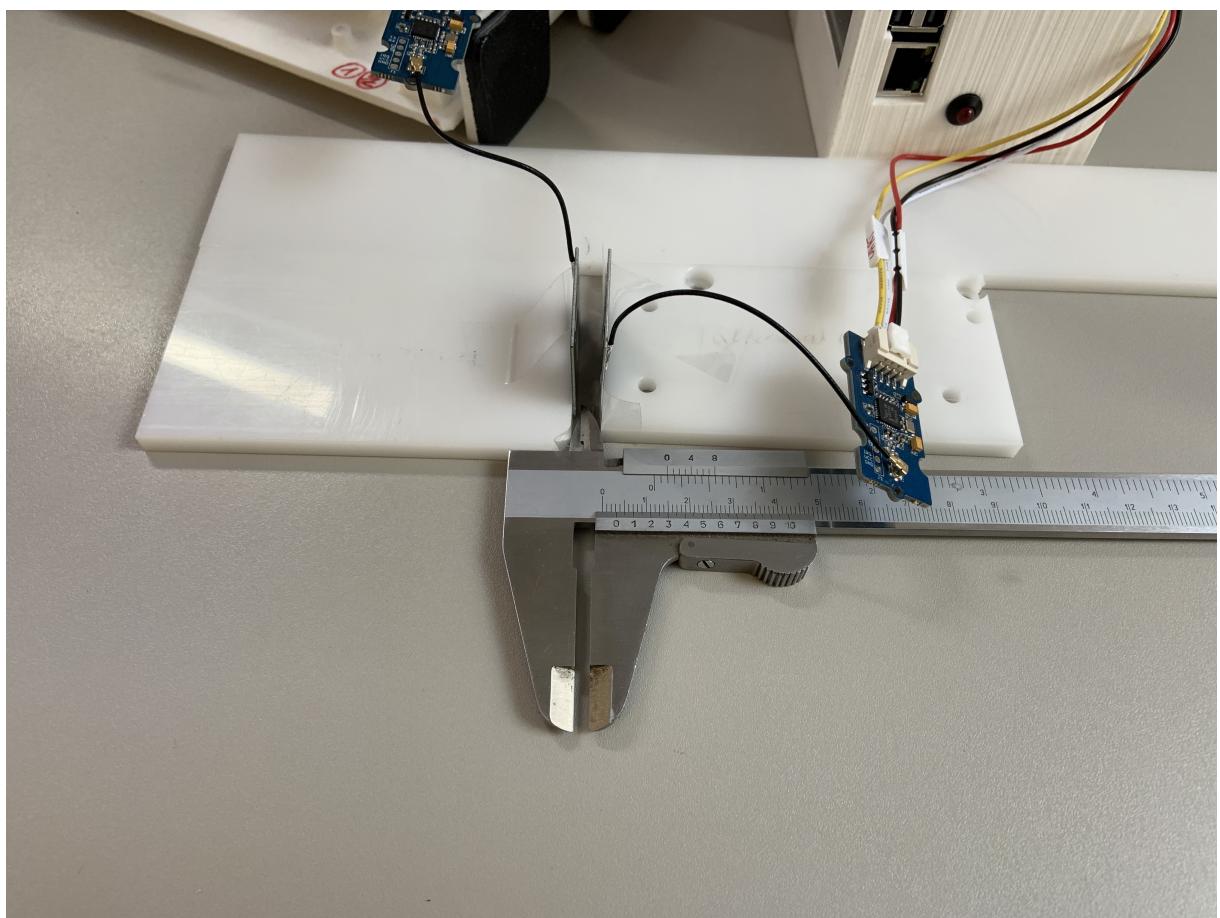
- erstellen eines einfachen images für das embedded System
- inkl ssh Server und SFTP
- qt 5 libraries
- eigenes package atctp
- test der toolchain

## 0.6.2 Verifikation NFC Technologie

- warum gewählter nfc reader => ndef lesen
- reichweiten test mit 22mm
- test mit benachbarten figuren
- warum kein RFID => keine speicherung von id auf der controller seite
- selbsterstellung von eigenen figuren ohne modifikation der controllerseite

## 0.6.3 Schrittmotor / Schrittmotorsteuerung

- warum => einfache ansteuerung
- keine STEP DIR somit muss embedded nicht echtzeitfähig sein und kann ggf auch andere task abarbeiten
- TMC schrittmotortreiber spi configuration



**Bild 0-6:** Grove PN532 NFC Reader mit Kabelgebundener Antenne

- und goto move => wait for move finished irw testen
- dafür einfacher python testreiber geschribene
- schrittverlust nicht zu erwarten

#### 0.6.4 3D Druck für den mechanischen Aufbau

### 0.7 Erstellung erster Prototyp

- vorgaben IKEA tisch al

#### 0.7.1 Technologieauswahl für ersten Protoypen

- durch verifikation bestätigt

#### 0.7.2 CAD Design

- Einabrietung in Fusion360
- Cad design aller bauteile
- standartxy
- erweiterng des spielraums durch zwei Magnete

#### 0.7.3 Schaltungsentwurf Motorsteuerung

- auswahl der Motortreiber (leise, bus ansteuerung)
- ansteuerung pn532 und umsetzung auf uart
- platinendesign
-

## **0.7.4 Anpassung Controller Software**

### **Implementierung HAL**

- ansteuerung des TMC5160
- ansterung des Microncontrollers (PN532, LED)
- integration in controller software

## **0.8 Erstellung zweiter Prototyp**

## **0.9 Fazit**

### **0.9.1 Ausblick**

# Literaturverzeichnis

- [1] DGT: *DGT Bluetooth Wenge.* <https://www.topschach.de/bluetooth-wenge-eboard-figuren-p-3842.html>. Version: 28.03.2021
- [2] DGT: *DTG Smart Board.* <https://www.topschach.de/smart-board-p-3835.html>. Version: 28.03.2021
- [3] GUERERO, Michael: *Automated Chess Board.* <https://create.arduino.cc/projecthub/maguerero/automated-chess-board-50ca0f>. Version: 28.03.2021
- [4] INC., SquareOff: *Grand Kingdom Set.* <https://squareoffnow.com/product/gks>. Version: 28.03.2021
- [5] INC., SquareOff: *Kingdom Set.* <https://squareoffnow.com/product/kds>. Version: 28.03.2021
- [6] MACHINES, DIY: *DIY Super Smart Chessboard.* <https://www.instructables.com/DIY-Super-Smart-Chessboard-Play-Online-or-Against-/>. Version: 28.03.2021
- [7] McEVOY, Brian: *PRINT CHESS PIECES, THEN DEFEAT THE CHESS-PLAYING PRINTER.* <https://hackaday.com/2021/03/06/print-chess-pieces-then-defeat-the-chess-playing-printer/>. Version: 28.03.2021
- [8] RAVICHANDRAN, Akash: *Automated Chess Board.* <https://create.arduino.cc/projecthub/automaters/automated-chess-5dbd7a>. Version: 28.03.2021
- [9] V., Deutsche I. e.: *DIN EN ISO 9241-220 Ergonomie der Mensch-System-Interaktion - Teil 220: Prozesse zur Ermöglichung, Durchführung und Bewertung menschzentrierter Gestaltung für interaktive Systeme in Hersteller- und Betreiber-*

## Literaturverzeichnis

---

*organisationen.* <https://www.din.de/de/mitwirken/normenausschuesse/naerg/veroeffentlichungen/wdc-beuth:din21:289443385>. Version: 04.04.2021

[10] YOUSIFNIMAT: *Chess Robot.* <https://www.instructables.com/Chess-Robot/>. Version: 28.03.2021

# Akronyme

**CSS** Cascading Style Sheets. 12

**FTP** File Transfer Protocol. 9

**GPL** General Public License. 5

**HTML** Hypertext Markup Language. 12

**HTTP** Hypertext Transfer Protocol. 9, 12

**HTTPS** Hypertext Transfer Protocol Secure. 9

**JS** JavaScript. 12

**USB** Universal Serial Bus. 4

**wlan** Wireless Local Area Network. 4

# Abbildungsverzeichnis

<b>0-1</b>	Gesamtübersicht der verwendeten Cloud-Infrastruktur . . . . .	8
<b>0-2</b>	Aufbau einer URI . . . . .	9
<b>0-3</b>	Aufbau der Service Architecture . . . . .	10
<b>0-4</b>	Webclient: Spielansicht . . . . .	12
<b>0-5</b>	Webclient: Statistiken . . . . .	13
<b>0-6</b>	Grove PN532 NFC Reader mit Kabelgebundener Antenne . . . . .	15

# **Tabellenverzeichnis**

0.1 Auflistung kommerzieller autonomer Schachtische . . . . .	3
0.2 Auflistung von Open-Source Schachtisch Projekten . . . . .	4

## **.1 Anhang**