**HTL Saalfelden**

**Systemplanung und Projektentwicklung**

****

**Projektdokumentation**

**2024 / 2025**

|  |  |
| --- | --- |
| **Projektbezeichnung** | MBot - Backend |
| **Projektteam** | Aberger Jonas, Haslinger Fabian, Hechenberger Tim |
| **Erstellt am** | 06.02.2025 |
| **Letzte Änderung am** | 22.05.2025 |
| **Status** | Fertig |
| **Aktuelle Version** | 5.0 |

**Änderungsverlauf**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr.** | **Datum** | **Version** | **Geänderte Kapitel** | **Art der Änderung** | **Autor** |
| 1 | 06.02.2025 | 1.0 | Initial / Sprint 1 | Erstellung | Tim Hechenberger |
| 2 | 26.02.2025 | 2.0 | Sprint 2 | Hinzufügen | Fabian Haslinger |
| 3 | 27.03.2025 | 3.0 | Alle | Komplette Überarbeitung + Sprint 3 | Jonas Aberger  Fabian Haslinger |
| 4 | 22.05.2025 | 4.0 | Sprint 4 | Überarbeitung - Sprint 4 | Tim Hechenberger |
| 5 | 22.05.2025 | 5.0 | Alle | Overview + Sprint 5 | Jonas Aberger |

**Inhalt**

[1. Allgemeines / Projektübersicht 3](#_Toc200521937)

[1.1 Projektbeschreibung 3](#_Toc200521938)

[2. Funktionale Anforderungen 4](#_Toc200521939)

[2.1 Use-Case Diagramm 4](#_Toc200521940)

[2.1.1 Steuerung des MBots 4](#_Toc200521941)

[2.1.2 Fahrstrecke manuell erstellen 4](#_Toc200521942)

[2.1.3 Fahrstrecke speichern 5](#_Toc200521943)

[2.1.4 Fahrstrecke abrufen 5](#_Toc200521944)

[2.1.5 LED & Geschwindigkeit anpassen 5](#_Toc200521945)

[2.1.6 Statusinformationen senden 5](#_Toc200521946)

[2.1.7 Kartografie des Raumes 5](#_Toc200521947)

[3. Nichtfunktionale Anforderungen 6](#_Toc200521948)

[4. Projektplanung 6](#_Toc200521949)

[4.1 Variantenbildung 6](#_Toc200521950)

[4.2 Allgemeine Planungsinformationen 7](#_Toc200521951)

[4.3 Projektumfeldanalyse 7](#_Toc200521952)

[5. Softwarearchitektur 8](#_Toc200521953)

[5.1 Aktivitätsdiagramm – Backend Architektur 8](#_Toc200521954)

[5.2 Sequenzdiagramme – Primärfunktionen Backend 9](#_Toc200521955)

[5.3 Komponentendiagramme 10](#_Toc200521956)

[5.4 C4-Diagramm 11](#_Toc200521957)

[5.5 Verteilungsdiagramme 14](#_Toc200521958)

[5.6 Softwarekomponenten / Programme 15](#_Toc200521959)

[5.6.1 SW-Programme 15](#_Toc200521960)

[5.6.2 SW-Komponenten 15](#_Toc200521961)

[6. Projektdurchführung 15](#_Toc200521962)

[6.1 Sprint 1 15](#_Toc200521963)

[6.1.1 Sprintplanung 15](#_Toc200521964)

[6.1.2 Sprint Demo 16](#_Toc200521965)

[6.1.3 Sprint Retrospektive 16](#_Toc200521966)

[6.1.4 Sprint Zusammenfassung 16](#_Toc200521967)

[6.2 Sprint 2 17](#_Toc200521968)

[6.2.1 Sprintplanung 17](#_Toc200521969)

[6.2.2 Sprint Demo 18](#_Toc200521970)

[6.2.3 Sprint Retrospektive 18](#_Toc200521971)

[6.2.4 Sprint Zusammenfassung 19](#_Toc200521972)

[6.3 Sprint 3 19](#_Toc200521973)

[6.3.1 Sprintplanung 19](#_Toc200521974)

[6.3.2 Sprint Demo 20](#_Toc200521975)

[6.3.3 Sprint Retrospektive 20](#_Toc200521976)

[6.3.4 Sprint Zusammenfassung 21](#_Toc200521977)

[6.4 Sprint 4 21](#_Toc200521978)

[6.4.1 Sprintplanung 21](#_Toc200521979)

[6.4.2 Sprint Demo 21](#_Toc200521980)

[6.4.3 Sprint Retrospektive 22](#_Toc200521981)

[6.4.4 Sprint Zusammenfassung 22](#_Toc200521982)

[6.5 Sprint 5 22](#_Toc200521983)

[6.5.1 Sprintplanung 22](#_Toc200521984)

[6.5.2 Sprint Retrospektive 23](#_Toc200521985)

[6.5.3 Sprint Zusammenfassung 23](#_Toc200521986)

[7. Installation / Software Deployment 24](#_Toc200521987)

[8. Projektabschluss 24](#_Toc200521988)

[8.1 Projektzusammenfassung 24](#_Toc200521989)

[8.2 Attachments 24](#_Toc200521990)

# Allgemeines / Projektübersicht

## Projektbeschreibung

Das Projekt umfasst die Entwicklung einer Applikation zur ferngesteuerten Bedienung des MBot2-Roboters. Die Konfiguration des Roboters unterstützt eine zweiseitige Kommunikation zwischen dem Klienten und dem Endgerät, dabei wird neben dem primären Steuerungsmodul eine Vielzahl weiterer Schnittstellen implementiert.

Die Anwendung soll sowohl auf PCs als auch auf mobilen Geräten nutzbar sein.

**Projektteam**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rolle(n)** | **Name** | **E-Mail** | **Team** |
| Entwickler | Jonas Aberger | jonas.aberger@htl-saalfelden.at | Backend |
| Entwickler | Fabian Haslinger | fabian.haslinger@htl-saalfelden.at | Backend |
| Entwickler | Tim Hechenberger | tim.hechenberger@htl-saalfelden.at | Backend |

# Funktionale Anforderungen

## Use-Case Diagramm

### Steuerung des MBots

Der Benutzer steuert den MBot über das Frontend. Während der Steuerung gibt es die Möglichkeit das Programm mit einem Exit-Command zu beenden. Falls das Frontend die Fahrstrecke speichern will, wird der Use Case „Fahrstrecke speichern“ automatisch ausgeführt.

### Fahrstrecke manuell erstellen

Der Benutzer kann über das Frontend eine Fahrstrecke manuell erstellen. Dies bedeutet, dass er eine Route ohne direkte Steuerung des MBot vorgibt. Dies erfolgt durch eine grafische Benutzeroberfläche, bei der vordefinierte Wegpunkte abgefahren werden. Diese Wegpunkte bestehen aus den Attributen Richtung, Geschwindigkeit und Zeit.

### Fahrstrecke speichern

Dieser Use Case speichert die Fahrstrecke des MBot in einer **MongoDB-Datenbank**. Er kann durch zwei verschiedene Aktionen ausgeführt werden.

* Automatisch nach der Steuerung des MBot (wenn {fahrstreckenSpeichern == true}.
* Manuell durch den Use Case **„Fahrstrecke manuell erstellen“,** der das Speichern direkt einbindet´.

### Fahrstrecke abrufen

Der Benutzer kann eine bereits gespeicherte Fahrstrecke aus der Datenbank auswählen und abfahren. Damit eine Fahrstrecke jedoch abgerufen werden kann, muss zuvor mindestens eine Strecke in der Datenbank gespeichert worden sein.

### LED & Geschwindigkeit anpassen

Der Benutzer hat die Möglichkeit, die LEDs des MBot ein- oder auszuschalten, indem er entweder einen physischen Knopf betätigt oder eine digitale Schaltfläche in der Benutzeroberfläche auswählt. Wenn der Benutzer den Knopf drückt, werden die LEDs des MBot entweder aktiviert oder deaktiviert, abhängig von der gewählten Einstellung.

### Statusinformationen senden

Das System tauscht kontinuierlich Daten aus, um dem Benutzer aktuelle Statusinformationen zur Hardware anzuzeigen. Diese Informationen geben einen klaren Überblick über den Zustand der Hardware, wie beispielsweise Batteriestatus oder weitere Sensordaten. Dadurch behält der Benutzer stets den Überblick und kann bei Bedarf reagieren.

### Kartografie des Raumes

Der Benutzer möchte den umliegenden Raum erkunden und automatisch eine übersichtliche Karte erstellen lassen. Dieser Kartografie-Modus läuft eigenständig und unabhängig von anderen Funktionen. Der Benutzer kann ihn aktivieren, und der MBot beginnt, den Raum systematisch zu erfassen und auszumappen. Die gesammelten Daten werden in Echtzeit verarbeitet und als detaillierte Karte dargestellt, die dem Benutzer zur Verfügung steht.

# Nichtfunktionale Anforderungen

Hier werden alle nichtfunktionalen Anforderungen der Backend-Gruppe-1 beschrieben:

* **Systemanforderungen:**
  + Das Backend wird als Webservice entwickelt und läuft auf einem lokalen Rechner
  + Die Kommunikation erfolgt über UDP
* **Leistung und Skalierbarkeit:**
  + Das System soll in der Lage sein, mehrere Roboter gleichzeitig zu verwalten
  + Keine zu hohen Latenzzeiten bei Steuerbefehlen und Synchronisation
* **Speicher und Ressourcennutzung:**
  + Kein zu hohem Speicherbedarf bei speichern der Daten, …

# Projektplanung

## Variantenbildung

* Basierend auf den Projektanforderungen wurden **verschiedene Varianten** analysiert:
* Programmiersprache: Python (mit MicroPython), Java, C#
  + Gewählte Variante: Python mit MicroPython
* Datenbank: FireBase, MongoDB, SQL
  + Gewählte Variante: MongoDB

## Allgemeine Planungsinformationen

**Allgemeine Planungsinformationen:**

* + Die Entwicklung erfolgt nach dem SCRUM-Framework, wobei das Grundkonzept, die Architektur sowie erste Tests bereits umgesetzt wurden
  + Der Code und die Dokumentation werden in einem GitHub-Repository verwaltet
  + Jedes Teammitglied erstellt am Anfang eines Arbeitstages einen eigenen Branch, auf welchem die Veränderungen gepusht werden.
  + Am Ende der Arbeitswoche werden alle Branches zurück auf den Main – mit der Aufsicht eines zweiten Entwicklers – gemerged.

## Projektumfeldanalyse

**Vergleichbare Produkte**

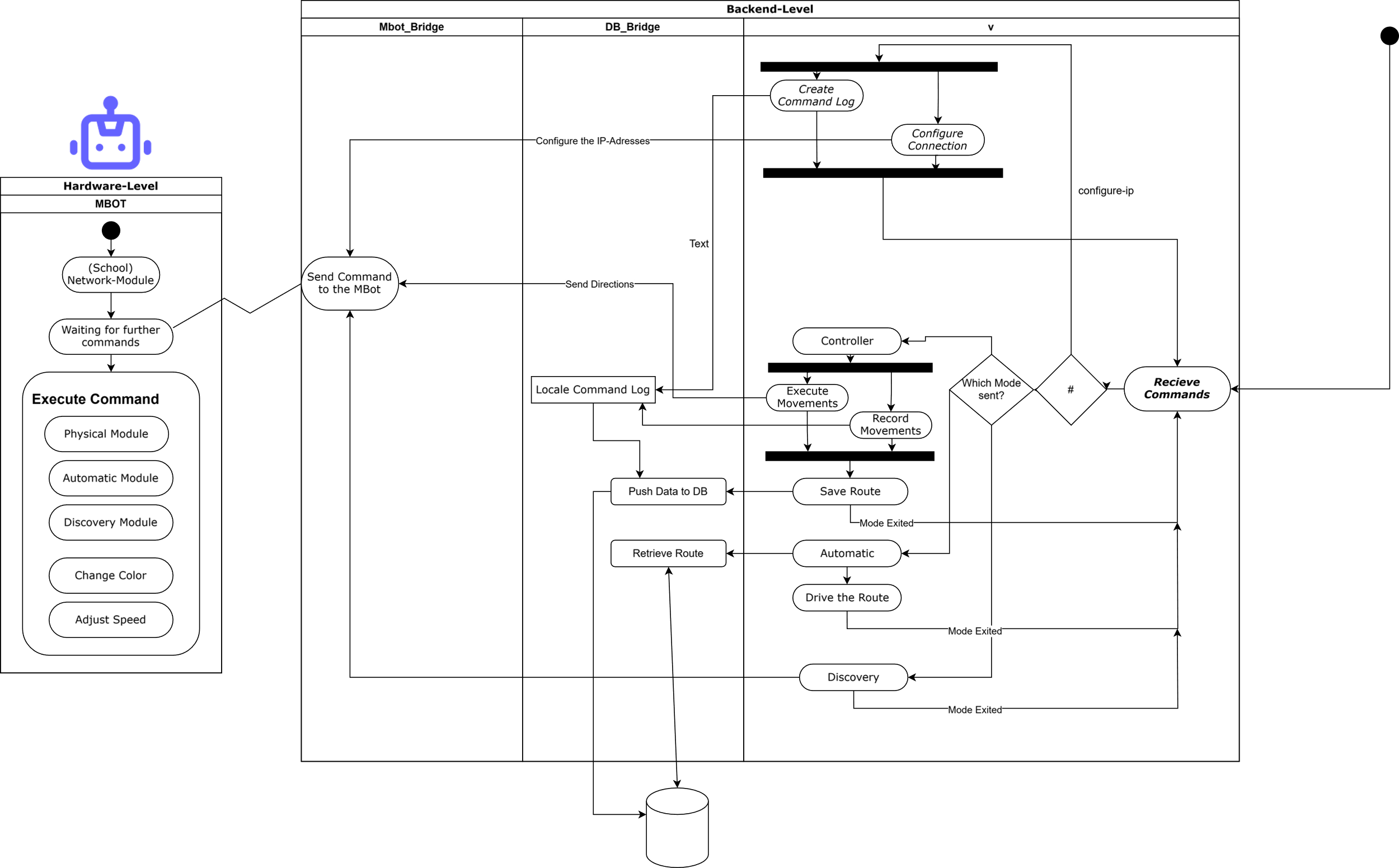
* Es gibt verschiedene MBot2-Steuerungs-Apps von Mblock, die aber keine individuelle Steuerung und Speicherung von Fahrstrecken bieten
* Bestehende Lösungen basieren meist auf Scratch, während unser System eine direkte Steuerung über Python und eine API ermöglicht

**Abgrenzung**

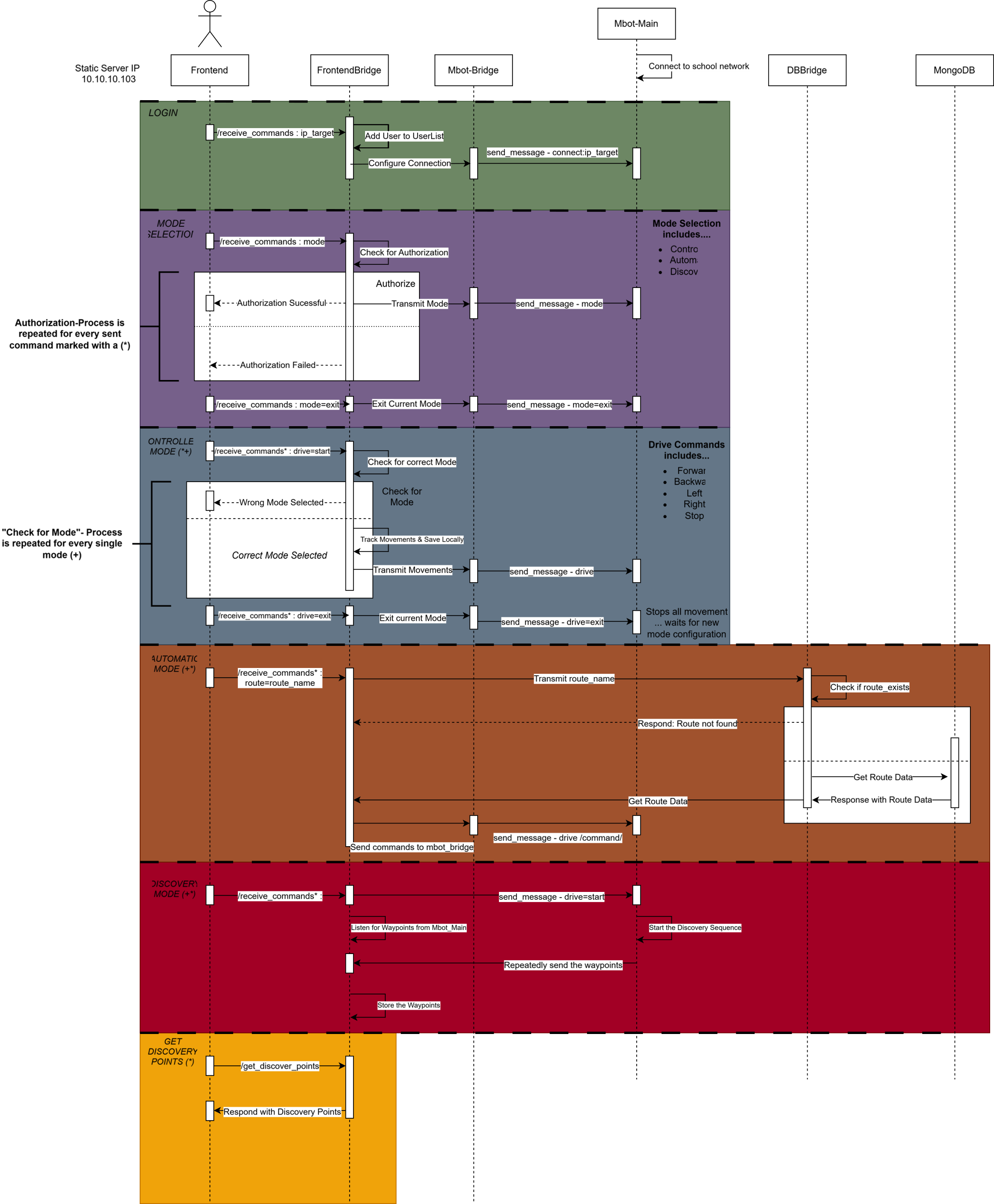
* Unser Projekt ermöglicht eine plattformunabhängige Steuerung (PC & Mobil)
* Der MBot2 kann nicht nur manuell gesteuert werden, sondern auch automatisierte Fahrstrecken abfahren
* Erweiterbare Backend-Architektur, die zukünftige Anpassungen erlaubt

# Softwarearchitektur

## Aktivitätsdiagramm – Backend Architektur



## Sequenzdiagramme – Primärfunktionen Backend



## Komponentendiagramme

Ein Bild, das Diagramm, Text, Plan, parallel enthält.

KI-generierte Inhalte können fehlerhaft sein.

## C4-Diagramm

Ein Bild, das Text, Screenshot, Diagramm, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Diagramm, Schrift enthält.

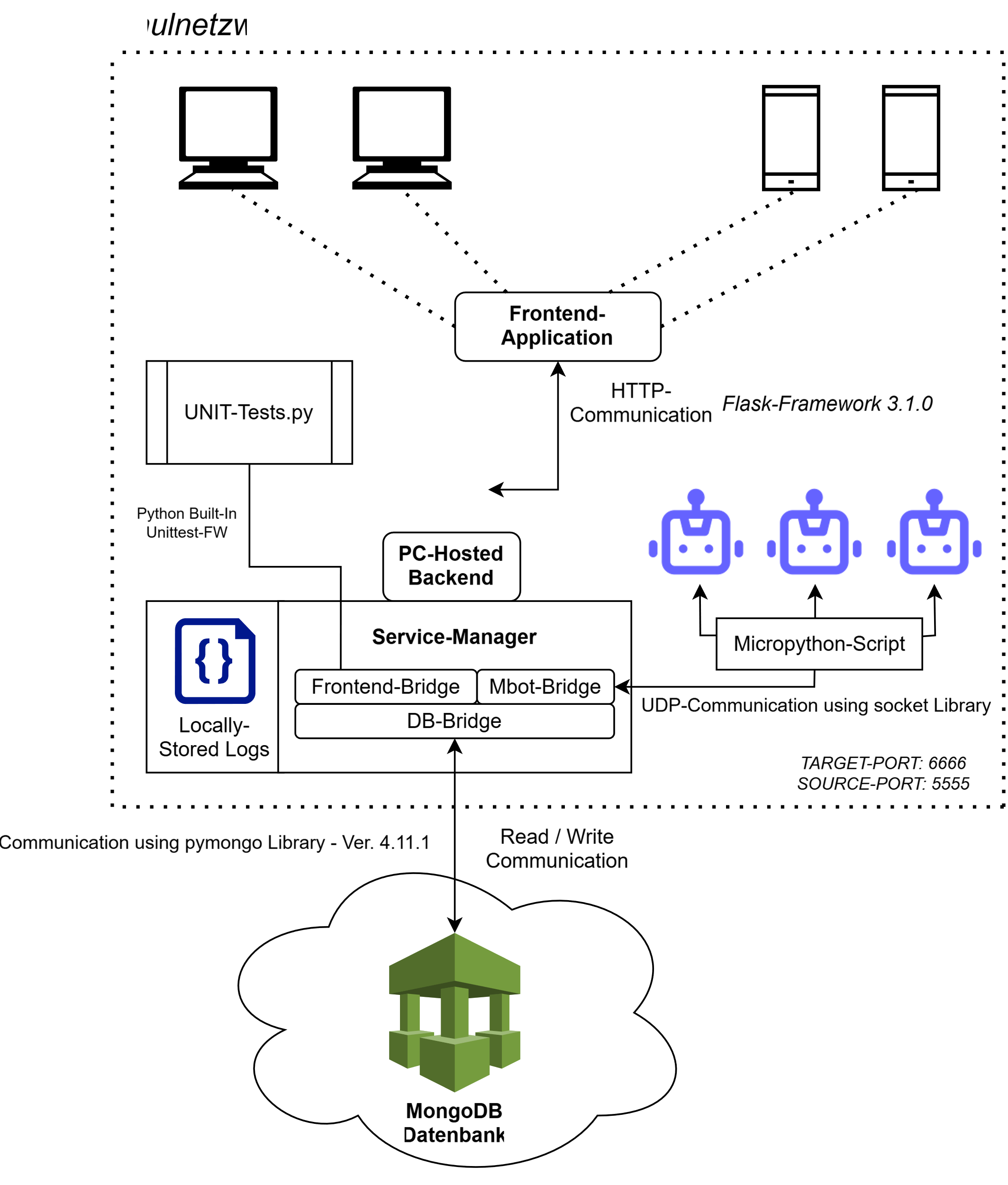
KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Diagramm, Screenshot, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.Ein Bild, das Text, Diagramm, Plan, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Verteilungsdiagramme



## Softwarekomponenten / Programme

### SW-Programme

|  |  |  |
| --- | --- | --- |
| **Name** | **Version** | **Beschreibung** |
| **Visual Studio Code** | 1.97 | IDE |

### SW-Komponenten

|  |  |  |
| --- | --- | --- |
| **Name** | **Version** | **Beschreibung** |
| **Python & Python-Unittest** | 3.0 | Hauptprogrammiersprache für das gesamte Backend |
| **MicroPython** | - | Programmiersprache für den MBot |
| **Flask** | 2.0 | API-Framework für das Backend |
| **MongoDB** | 8.0 | NoSQL-Datenbank zur Speicherung von Anwendungsdaten |
| **ChatGPT** | o4 (GPT-4.5) | KI-Unterstützung bei Code-Generierung und -Analyse |
| **Git / Github** | 3.15.2 | Versionskontrolle und kollaborative Codeverwaltung |

# Projektdurchführung

## Sprint 1

### Sprintplanung

Dauer: 29.01.2025 – 18.02.2925

**Ausgewählte User Stories:**

|  |  |
| --- | --- |
| **User-Story** | **Beschreibung** |
| Netzwerkanbindung | Automatische Verbindung Schul-WLAN und IP anzeigen |
| Verbindung zum MBot herstellen | UDP-Verbindung einrichten und testen |

**Anzahl Story-Points:** 105 Story-Points

**Ausgewählte Punkte aus der Impediment Liste:**

* Netzwerkanbindung:
  + Statusinformation mittels LED ausgeben
  + IP-Adresse anzeigen
  + Eindeutige Identifikation
* Verbindung zum MBot herstellen:
  + Kommunikation mit dem MBot über das Netzwerk
  + Unterstützung von Steuerbefehlen an den MBot
  + Antwort des MBot auf Befehle verarbeiten und ans Frontend zurücksenden

### Sprint Demo

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

KI-generierte Inhalte können fehlerhaft sein.

### Sprint Retrospektive

Während des Sprints lief die Zusammenarbeit im Team sehr gut, und wir konnten die geplanten Aufgaben erfolgreich umsetzen. Anfangs brauchten wir jedoch etwas Zeit, um uns in das Projekt einzuarbeiten und die technischen Anforderungen zu verstehen.

### Sprint Zusammenfassung

Der Sprint verlief sehr gut. Basierend auf unserer Sprint-Velocity können wir davon ausgehen, dass wir den Endtermin voraussichtlich etwas schneller als geplant erreichen werden.

Ein Bild, das Text, Reihe, Screenshot, Diagramm enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Sprint 2

### Sprintplanung

Dauer: 19.02.2025 – 12.03.2025

|  |  |
| --- | --- |
| **User-Story** | **Beschreibung** |
| Kommunikation / Anbindung zum Frontend | Informationsaustausch zwischen Frontend & Backend mittels Schnittstellen; Entgegennahme von Befehlen |
| Physisches Steuerungsmodul | Ansteuerung des MBOTS; Verbindung multipler Geräte |

**Anzahl Story-Points**: 185 Story-Points

**Ausgewählte Punkte aus der Impediment Liste:**

**Kommunikation / Anbindung zum Frontend:**

* Bereitstellung von Schnittstellen
* Genereller Informationsaustausch mit dem Frontend
* Erfolgreiche Entgegennahme von Inputs vom Frontend
* Abfertigung der Frontend-Backend Kommunikation

**Physisches Steuerungsmodul:**

* Erfolgreiche Anbindung an das DB-System
* Lokale Speicherung der Fahrstrecken
* Uploaden der Fahrstrecken zur DB
* Abrufen der gespeicherten Fahrstrecken

### Sprint Demo

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

KI-generierte Inhalte können fehlerhaft sein.

**Zusätzlich angefangene / erledigte User Stories:**

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

### Sprint Retrospektive

Während des Sprints lief die Zusammenarbeit im Team sehr gut, und wir konnten die geplanten Aufgaben erfolgreich umsetzen. Da wir sehr schnell fertig wurden, konnten wir auch noch zusätzliche User Stories beginnen und finalisieren. Allerdings wurde der Fortschritt bei der Fahrstreckenlogik durch Komplikationen mit dem Frontend stark verzögert.

### Sprint Zusammenfassung

Der Sprint verlief sehr gut. Basierend auf unserer Sprint-Velocity können wir davon ausgehen, dass wir den Endtermin voraussichtlich etwas schneller als geplant erreichen werden.

Ein Bild, das Text, Quittung, Screenshot, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Sprint 3

### Sprintplanung

**Dauer: 13.03.2025 – 02.04.2025**

|  |  |
| --- | --- |
| **User Story** | **Beschreibung** |
| DB | Kommunikation und Speicherung/Laden von Fahrstrecken |

**Anzahl Story-Points:** 65 Story-Points

**Ausgewählte Punkte aus der Impediment Liste:**

**DB:**

* Form der Datenspeicherung bestimmen
* Konfiguration & Setup der Datenbank
* Hosting der Datenbank
* Kommunikation mit der Datenbank

### Sprint Demo

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

KI-generierte Inhalte können fehlerhaft sein.

**Zusätzlich angefangene / erledigte User Stories:**

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.

**Spike Story:**

* **Ziel des Spikes:** Erforschung und Entwicklung eines Prototyps für den Discovery Mode des MBot, um Hindernisse zu erkennen und zu umgehen.
* **Timeboxing:** Ein fester Zeitraum wurde gesetzt, um den Prototypen zu entwickeln und Lösungen für die Hindernisvermeidung zu testen.
* **Prototypen / Variantenbildung:**
  + Verschiedene Ansätze wurden getestet, darunter:
    - *BFS (Breitensuche)*
    - *A\* (A\*-Algorithmus)*
    - *Neander Förmig (ein weniger strukturierter Ansatz)*
* **Lösungsansatz:** Alte Ansätze wurden verworfen, um auf einen neuen, von Herrn Eigner vorgeschlagenen Ansatz zurückzugreifen.
* **Ergebnisse:** Der Discovery Mode konnte erfolgreich implementiert werden, Hindernisse wurden korrekt erkannt und umfahren.

### Sprint Retrospektive

Die Arbeitsatmosphäre während dieses Sprints war insgesamt positiv und von einem ruhigen, konzentrierten Miteinander geprägt. Herausforderungen traten insbesondere in der Abstimmung mit dem Frontend-Team sowie im Bereich des Discovery-Mode-Moduls auf (vgl. Spike-Story). Die Kommunikation zwischen den beteiligten Teams war teilweise unklar, was zu Missverständnissen und Verzögerungen führte.

Für den nächsten Sprint wurde beschlossen, den sprachlichen Austausch gezielter zu verbessern – etwa durch regelmäßige kurze Sync-Meetings oder klar definierte Kommunikationskanäle. Zudem soll durch die stärkere Abgrenzung und Isolation einzelner Arbeitsprozesse die Eigenverantwortung innerhalb der Teams gestärkt und Schnittstellenprobleme reduziert werden.

### Sprint Zusammenfassung

Der Sprint verlief insgesamt erfolgreich. Es gab einige Herausforderungen im Zusammenhang mit dem Discovery Mode, die jedoch erfolgreich gelöst werden konnten. Basierend auf unserer Sprint-Velocity gehen wir davon aus, dass wir den Endtermin voraussichtlich etwas früher als ursprünglich geplant erreichen werden.

## Sprint 4

### Sprintplanung

**Dauer: 30.04.2025 – 14.05.2025**

|  |  |
| --- | --- |
| **User Story** | **Beschreibung** |
| Discovery-Mode / Kartographie | MBot sammelt, verarbeitet und überträgt Sensordaten an das Frontend |

**Anzahl Story-Points:** 90 Story-Points

**Ausgewählte Punkte aus der Impediment Liste:**

**Kartographie-Modul:**

* Sammeln der Daten (Sensor)
* Aufarbeitung der empfangenen Daten
* Bereitstellung & Speicherung der Daten

### Sprint Demo

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

KI-generierte Inhalte können fehlerhaft sein.

**Zusätzlich angefangene / erledigte User Stories:**



### Sprint Retrospektive

Die Arbeitsatmosphäre während dieses Sprints war durchweg positiv und produktiv. Die Zusammenarbeit im Team verlief reibungslos, und es traten keine größeren Herausforderungen auf. Alle Aufgaben konnten termingerecht abgeschlossen werden.  
Für den kommenden Sprint liegt der Fokus vor allem auf der finalen Dokumentation sowie dem Beheben kleinerer Bugs. Um den guten Workflow beizubehalten, soll die bisherige Arbeitsweise fortgeführt und bewährte Kommunikationsstrukturen beibehalten werden.

### Sprint Zusammenfassung

Der Sprint verlief insgesamt sehr erfolgreich. Es traten keine nennenswerten Herausforderungen auf, alle Story Points konnten planmäßig abgeschlossen werden. Abgesehen von kleineren Bugfixes steht nun hauptsächlich die Dokumentation aus. Auf Basis unserer Sprint-Velocity gehen wir davon aus, den Endtermin sogar etwas früher zu erreichen.

### Sprint 5

### Sprintplanung

**Dauer: 15.05.2025 – 04.06.2025**

|  |  |
| --- | --- |
| **User Story** | **Beschreibung** |
| Dokumentation | Abschluss der parallel geführten Dokumentation |

**Anzahl Story-Points:** -

### Sprint Retrospektive

Insgesamt war die Team-Moral im vergangenen Sprint als neutral bis ausgeglichen zu bewerten. Die finalen Aufgaben im Rahmen der Dokumentationsaufarbeitung gestalteten sich stellenweise als mühsam, konnten jedoch ohne größere Schwierigkeiten abgeschlossen werden. Während des gesamten Arbeitsprozesses traten keine nennenswerten Rückschläge auf.

### Sprint Zusammenfassung

Die Projektdokumentation wurde auf inhaltliche Lücken überprüft und entsprechend ergänzt. Zudem wurden das MBot-Skript sowie das Hauptprogramm des Backends in Zusammenarbeit mit dem Frontend-Team umfassend auf Bugs und Fehler getestet, dabei identifizierte Probleme konnten im Anschluss – ohne weitere größere Aufwande - direkt behoben werden.

# Installation / Software Deployment

Die fertige Softwarelösung war ursprünglich nicht für ein offizielles Deployment vorgesehen. Vorerst wird die Backend-Applikation lokal und statisch auf einem ausgewählten Rechner ausgeführt. Für eine zukünftige Nutzung und mögliche Deployment-Optionen wären zusätzliche Abstimmungen sowie Erweiterungen am Frontend erforderlich, die im Rahmen dieses Projekts jedoch weder geplant noch umgesetzt wurden.

# Projektabschluss

## Projektzusammenfassung

In diesem Projekt haben wir in fünf Sprints ein System zur Steuerung und Kommunikation mit dem MBot entwickelt. Zuerst ging es um die Verbindung mit dem Schul-WLAN und dem MBot selbst. Danach haben wir die Schnittstelle zwischen Frontend und Backend aufgebaut und den MBot so programmiert, dass er auf Steuerbefehle reagieren kann.

Im dritten Sprint wurde die Datenbank eingerichtet und der sogenannte "Discovery Mode" getestet – ein Modus, in dem der MBot Hindernisse erkennt und umfahren kann. Dabei gab es kleine Schwierigkeiten mit der Zusammenarbeit mit dem Frontend-Team, die aber gelöst wurden.

Im vierten Sprint haben wir die Sensordaten des MBots verarbeitet und an das Frontend geschickt. Alles lief reibungslos. Im letzten Sprint stand die Dokumentation im Fokus. Außerdem wurden letzte Fehler im Code behoben.

Das Projekt war insgesamt sehr erfolgreich. Alle Aufgaben konnten wie geplant oder sogar schneller abgeschlossen werden. Die Teamarbeit funktionierte gut und alle Ziele wurden erreicht.

## Attachments

<https://github.com/jonasaberger/SYP4_MBOT_G1> - GitHub

|  |  |
| --- | --- |
| **Datei** | **Nutzen** |
| main.py | Starten des ServiceManagers |
| service\_manager.py | Starten der API & Server |
| frontend\_bridge.py | Stellt Verbindung zum Frontend her |
| db\_bridge.py | Stellt Verbindung zur Datenbank her |
| MBot\_bridge.py | Kommunikation zum MBot |
| UnitTests.py | Unit Tests |
| mblock\_demo.py | File für den MBot zum Probieren |
| mblock\_main.py | Funktionierender Code für den MBot |