**HTL Saalfelden**

**Systemplanung und Projektentwicklung**

****

**Projektdokumentation**

**2023 / 2024**

|  |  |
| --- | --- |
| **Projektbezeichnung** | MBot-Controller |
| **Projektteam** | Tobias Haas, Patrick Thor, Stefan Rautner |
| **Erstellt am** | 27.05.2024 |
| **Letzte Änderung am** | 28.05.2024 |
| **Status** | Fertig |
| **Aktuelle Version** | 1.1 |

**Änderungsverlauf**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr.** | **Datum** | **Version** | **Geänderte Kapitel** | **Art der Änderung** | **Autor** |
| 1 | 27.05.2024 | 1.0 | Alle | Erstellung | Stefan Rautner |
| 2 | 28.05.2024 | 1.1 | 5.1, 5.2 | Ergänzung | Stefan Rautner |

**Inhalt**

[1. Allgemeines / Projektübersicht 4](#_Toc167877553)

[1.1 Projektbeschreibung 4](#_Toc167877554)

[1.2 Projektteam und Schnittstellen 4](#_Toc167877555)

[2. Funktionale Anforderungen 5](#_Toc167877556)

[2.1 Use Cases 5](#_Toc167877557)

[2.1.1 Download WebApp 5](#_Toc167877558)

[2.1.2 Download MBot-Script 6](#_Toc167877559)

[2.1.3 Zwischenserver 6](#_Toc167877560)

[2.1.4 Steuerung 7](#_Toc167877561)

[2.1.5 LineFollower 7](#_Toc167877562)

[2.1.6 SuicidePrevention 8](#_Toc167877563)

[2.1.7 Ambientebeleuchtung 8](#_Toc167877564)

[2.1.8 DinoGame 9](#_Toc167877565)

[3. Nichtfunktionale Anforderungen 9](#_Toc167877566)

[4. Projektplanung 9](#_Toc167877567)

[4.1 Variantenbildung 10](#_Toc167877568)

[4.2 Machbarkeitsstudie 10](#_Toc167877569)

[4.3 Allgemeine Planungsinformationen 10](#_Toc167877570)

[4.4 Projektumfeldanalyse 10](#_Toc167877571)

[5. Softwarearchitektur 10](#_Toc167877572)

[5.1 Aktivitätsdiagramme 11](#_Toc167877573)

[5.1.1 Aktivitätsdiagramm MBot Verbinden 11](#_Toc167877574)

[5.1.2 Aktivitätsdiagramm Daten Senden 12](#_Toc167877575)

[5.1.3 Aktivitätsdiagramm Disconnect 12](#_Toc167877576)

[5.1.4 Aktivitätsdiagramm Close 13](#_Toc167877577)

[5.2 Sequenzdiagramme 13](#_Toc167877578)

[5.2.1 Sequenzdiagramm MBot Verbinden 13](#_Toc167877579)

[5.2.2 Sequenzdiagramm Daten Senden 14](#_Toc167877580)

[5.2.3 Sequenzdiagramm Disconnect 14](#_Toc167877581)

[5.2.4 Sequenzdiagramm Close 14](#_Toc167877582)

[5.3 Komponentendiagramme 15](#_Toc167877583)

[5.4 Verteilungsdiagramme 16](#_Toc167877584)

[5.5 Softwarekomponenten / Programme 16](#_Toc167877585)

[5.5.1 SW Programme 16](#_Toc167877586)

[5.5.2 SW Komponenten 16](#_Toc167877587)

[6. Projektdurchführung 17](#_Toc167877588)

[6.1 Sprint 1 17](#_Toc167877589)

[6.1.1 Sprintplanung 17](#_Toc167877590)

[6.1.2 Sprint Demo 18](#_Toc167877591)

[6.1.3 Sprint Retrospektive 18](#_Toc167877592)

[6.1.4 Sprint Zusammenfassung 18](#_Toc167877593)

[6.2 Sprint 2 19](#_Toc167877594)

[6.2.1 Sprintplanung 19](#_Toc167877595)

[6.2.2 Sprint Demo 20](#_Toc167877596)

[6.2.3 Sprint Retrospektive 20](#_Toc167877597)

[6.2.4 Sprint Zusammenfassung 20](#_Toc167877598)

[6.3 Sprint 3 21](#_Toc167877599)

[6.3.1 Sprintplanung 21](#_Toc167877600)

[6.3.2 Sprint Demo 22](#_Toc167877601)

[6.3.3 Sprint Retrospektive 22](#_Toc167877602)

[6.3.4 Sprint Zusammenfassung 22](#_Toc167877603)

[6.4 Sprint 4 24](#_Toc167877604)

[6.4.1 Sprintplanung 24](#_Toc167877605)

[6.4.2 Sprint Demo 24](#_Toc167877606)

[6.4.3 Sprint Retrospektive 24](#_Toc167877607)

[6.4.4 Sprint Zusammenfassung 25](#_Toc167877608)

[6.5 Sprint 4 26](#_Toc167877609)

[6.5.1 Sprintplanung 26](#_Toc167877610)

[6.5.2 Sprint Demo 26](#_Toc167877611)

[6.5.3 Sprint Retrospektive 26](#_Toc167877612)

[6.5.4 Sprint Zusammenfassung 27](#_Toc167877613)

[7. Installation / Software deployment 29](#_Toc167877614)

[8. Projektabschluß 29](#_Toc167877615)

[8.1 Projektzusammenfassung 29](#_Toc167877616)

[8.2 Attachments 29](#_Toc167877617)

# Allgemeines / Projektübersicht

## Projektbeschreibung

Das Projekt zielt darauf ab, eine Anwendung zur Fernsteuerung eines MBot2s zu entwickeln, die sowohl auf Computern als auch auf mobilen Geräten (Handys, Tablets, …) funktioniert. Der MBot2 ist mit diversen Sensoren und Aktoren ausgestattet und wird über MicroPython programmiert. Die Anwendung ermöglicht die automatische Netzwerkverbindung, zeigt Sensordaten an und erlaubt das Verarbeiten von Steuerbefehlen wie Bewegungsrichtung und Geschwindigkeit. Ein Sicherheitsmodus verhindert Kollisionen und einen Linienfolgemodus zum automatischen Folgen einer Linie. Das Projekt wird dem agilen Projektmanagement SCRUM durchgeführt. Die Projektlaufzeit ist vom 10. Januar 2024 bis zum 12. Juni 2024.

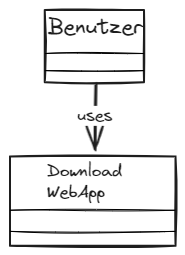
## Projektteam und Schnittstellen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rolle(n)** | **Name** | **Telefon** | **E-Mail** | **Team** |
| Developer | Tobias Haas | Privat | [Tobias.haas@htl-saalfelden.at](mailto:Tobias.haas@htl-saalfelden.at) | Group05 |
| 3D-Designer | Patrick Thor | Privat | [Patrick.thor@htl-saalfeldelden.at](mailto:Patrick.thor@htl-saalfeldelden.at) | Group05 |
| Developer | Stefan Rautner | Privat | [Stefan.rautner@htl-saalfelden.at](mailto:Stefan.rautner@htl-saalfelden.at) | Group05 |

# Funktionale Anforderungen

## Use Cases

### Download WebApp



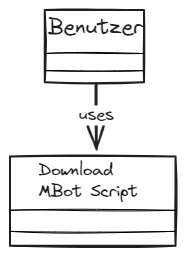
**Akteur:** Benutzer

**Vorbedingung:** Benutzer muss funktionierenden Internetzugang haben

**Beschreibung:** Der Benutzer drückt auf den „Download“-Button auf der WebApp (Home-Screen, relativ weit unten) und der Download startet automatisch, nachdem dieser Abgeschlossen ist, muss der User in dem erscheinenden Dialog auf „Installieren“ drücken um die WebApp zu Installieren.

**Ergebnis:** Die WebApp ist wie eine übliche App auf dem lokalen Gerät des Users installiert und kann ohne Internetzugriff benötigt werden (für die Steuerung des MBot muss aber trotzdem eine Internetverbindung hergestellt werden).

### Download MBot-Script



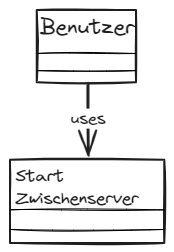
**Akteur:** Benutzer

**Vorbedingung:** Benutzer muss funktionierenden Internetzugang haben.

**Beschreibung:** Der Benutzer drückt auf den „Download“-Text auf der WebApp (Home-Screen, ganz unten) und der Download startet automatisch.

**Ergebnis:** Das MBot Script ist heruntergeladen und kann jetzt vom User weiter entwickelt/verwendet werden.

### Zwischenserver



**Akteur:** Benutzer

**Vorbedingung:** Benutzer muss funktionierenden Internetzugang haben.

**Beschreibung:** Der Benutzer öffnet den Tab „Controller“ in der Navigationsleiste der WebApp, dadurch wird der Zwischenserver (falls das aufrufende Gerät ein PC ist, wenn nicht, dann erscheint ein Dialog) automatisch heruntergeladen. Der Zwischenserver muss danach vom Benutzer manuell ausgeführt werden. Der Zwischenserver lädt sich automatisch die benötigten Module herunter und installiert diese, sodass der Zwischenserver funktionsfähig ist.

**Ergebnis:** Der Zwischenserver ist gestartet und wartet auf Befehle von WebApp und MBot

### Steuerung

Ein Bild, das Text, Entwurf, weiß, Schrift enthält.

Automatisch generierte Beschreibung

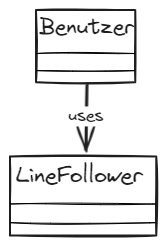
**Akteur:** Benutzer

**Vorbedingung:** Benutzer muss funktionierenden Internetzugang haben.

**Beschreibung:** Der Benutzer öffnet den Tab „Controller“ in der Navigationsleiste, danach muss der heruntergeladene Zwischenserver gestartet werden und die Checkbox „Zwischenserver gestartet“ angekreuzt werden. Jetzt muss der gewünschte MBot ausgewählt werden, danach kann man diesen Steuern.

**Ergebnis:** Der Benutzer kann den MBot über die WebApp steuern

### 2.1.5 LineFollower



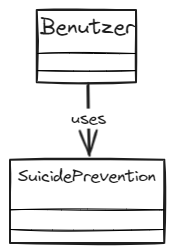
**Akteur:** Benutzer

**Vorbedingung:** Benutzer muss funktionierenden Internetzugang haben.

**Beschreibung:** Der Benutzer öffnet den Tab „Controller“ in der Navigationsleiste, danach muss der heruntergeladene Zwischenserver gestartet werden und die Checkbox „Zwischenserver gestartet“ angekreuzt werden. Jetzt muss der gewünschte MBot ausgewählt werden, danach muss der Benutzer die Checkbox „LineFollower“ ankreuzen.

**Ergebnis:** Der Benutzer hat die Linienverfolgung des MBots aktiviert

### 2.1.6 SuicidePrevention



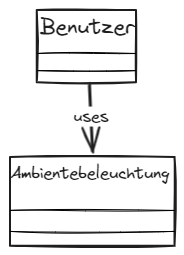
**Akteur:** Benutzer

**Vorbedingung:** Benutzer muss funktionierenden Internetzugang haben.

**Beschreibung:** Der Benutzer öffnet den Tab „Controller“ in der Navigationsleiste, danach muss der heruntergeladene Zwischenserver gestartet werden und die Checkbox „Zwischenserver gestartet“ angekreuzt werden. Jetzt muss der gewünschte MBot ausgewählt werden, danach muss der Benutzer die Checkbox „SuicidePrevention“ ankreuzen.

**Ergebnis:** Der Benutzer hat die SuicidePrevention auf dem MBot gestartet

### 2.1.7 Ambientebeleuchtung



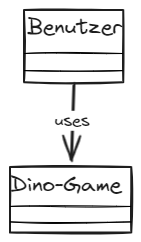
**Akteur:** Benutzer

**Vorbedingung:** Benutzer muss funktionierenden Internetzugang haben.

**Beschreibung:** Der Benutzer öffnet den Tab „Controller“ in der Navigationsleiste, danach muss der heruntergeladene Zwischenserver gestartet werden und die Checkbox „Zwischenserver gestartet“ angekreuzt werden. Jetzt muss der gewünschte MBot ausgewählt werden, danach kann der Benutzer die Farben der jeweiligen LEDs einstellen. (Die Ambientebeleuchtung leuchtet nur, wenn der MBot stillsteht).

**Ergebnis:** Der Benutzer kann die Ambientebeleuchtung auf dem MBot starten und umstellen

### 2.1.8 DinoGame



**Akteur:** Benutzer

**Vorbedingung:** Benutzer muss funktionierenden Internetzugang haben.

**Beschreibung:** Der Benutzer öffnet den Tab „Dino-Game“ und kann dort eine leichte Version des originalen Dino-Games von Google spielen.

**Ergebnis:** Der Benutzer kann das Dino-Game spielen

# Nichtfunktionale Anforderungen

Benutzer benötigt einen Browser

benutzer benötigt eine funktionierende Internetverbindung

Benutzer benötigt einen PC

Verwendetes Betriebssystem: Windows 11 64-bit

Verwendete IDE: PyCharm Version 2024.1.1 & Mblock Version 5.4.3

# Projektplanung

WebApp in den Sprachen HTML, CSS & JS

Zwischenserver in der Sprache Python

MBot in der Sprache Micropython

Unterstützung für alle Geräte & Betriebssysteme (benötigt aber PC für den Zwischenserver)

## Variantenbildung

Mit einem Spring-Boot Zwischenserver (verworfen, weil größeres Framework)

Mit einem Docker-Container (verworfen, Docket nicht so leicht zu bedienen)

Mit einem Zwischenserver in Python (ausgeführt, benötigt nur Python, installiert Module automatisch, sehr benutzerfreundlich)

Ohne Zwischenserver durch WebSocket (verworfen, MBot keine WebSocket Unterstützung)

Ohne Zwischenserver durch UDP (verworfen, UDP im Web nicht möglich)

Ohne Zwischenserver durch TCP (verworfen, TCP im Web nicht möglich)

## Machbarkeitsstudie

Zwischenserver in Python (Unterstützung für WebSockets und UDP & TCP im lokalen Netzwerk)

MBot mit UDP & TCP Sockets verbinden (UDP wegen Broadcast, TCP weil sicherer)

## Allgemeine Planungsinformationen

Aufteilung:

Tobias Haas: grafische Oberfläche & Dino-Game

Patrick Thor: 3D-Model erstellen & anzeigen

Stefan Rautner: Programmierung

## Projektumfeldanalyse

Es gibt noch keine vergleichbaren Produkte am Markt. Es gibt schon Fernsteuerungen für Roboter aber noch keine für den MBot2. Es wird aber in nächster Zukunft mehrere geben, da die anderen Gruppen in unserer Klasse den gleichen Projektauftrag erhalten haben. Die Stakeholder des Projekts sind unsere Professoren Herr Falkensteiner & Herr Eigner.

# Softwarearchitektur

Komponenten: WebApp, Zwischenserver & MBot

Verteiltes System, das auf einem Gerät die von GitHub-Pages gehostete WebApp besucht, auf dem gleichen oder anderen PC muss der Zwischenserver gehostet werden (wird von der WebApp auto gedownloaded), wenn die Steuerung über ein Mobilgerät erfolgt, dann Zwischenserver auf PC und WebApp-Aufruf auf Mobilgerät und MBot Script local auf dem MBot.

Die WebApp und der MBot kommunizieren miteinander über den Zwischenserver miteinander

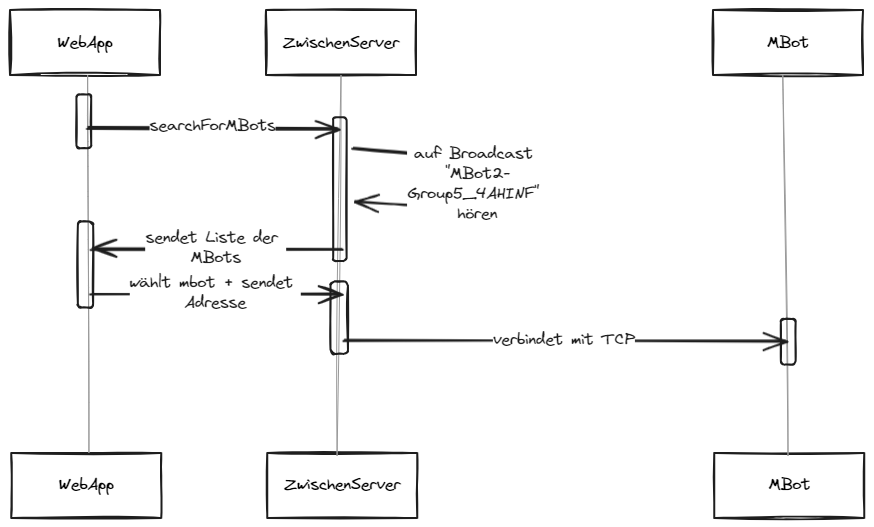
Der Zwischenserver erstellt einen UDP-Socket der nach der Broadcastnachricht des MBot sucht

Der Zwischenserver verbindet sich dann durch einen TCP-Socket mit dem MBot für durchgehende Kommunikation

Die WebApp gibt dem Zwischenserver den Befehl nach MBot-Broadcastnachrichten zu suchen

## Aktivitätsdiagramme

### Aktivitätsdiagramm MBot Verbinden



WebApp sendet Befehl zum MBots suchen, Zwischenserver sucht über den UDP-Socket den Broadcast nach „MBot2-Group05\_4AHINF“ ab. Alle gefundenen MBots werden an die WebApp zurückgegeben, dort wählt der Benutzer einen MBot aus. Die Addresse des MBot wird automatisch an den Zwischenserver gesendet und dieser verbindet sich über die Addresse an den TCP-Socket.

### Aktivitätsdiagramm Daten Senden

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

Automatisch generierte Beschreibung

WebApp sendet Steuerbefehl an Zwischenserver, dieser leitet den Befehl an den MBot weiter. Dort werden die Daten verarbeitet (SuicidePrevention, Ambientebeleuchtung, …), die Sensordaten ausgelesen und an den Zwischenserver gesendet. Dieser Sendet die daten dann an die WebApp weiter. Dort werden die Sensordaten verarbeitet und dem User angezeigt.

### Aktivitätsdiagramm Disconnect

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

Automatisch generierte Beschreibung

Die WebApp sendet den Disconnect Befehl an den Zwischenserver, dieser sendet darauf den „Disconnect“-Befehl an den MBot und schließt die Verbindung zum TCP-Socket. Der MBot „restartet“ dadurch sein Script und bereitet sich für die nächste Verbindung vor.

### Aktivitätsdiagramm Close

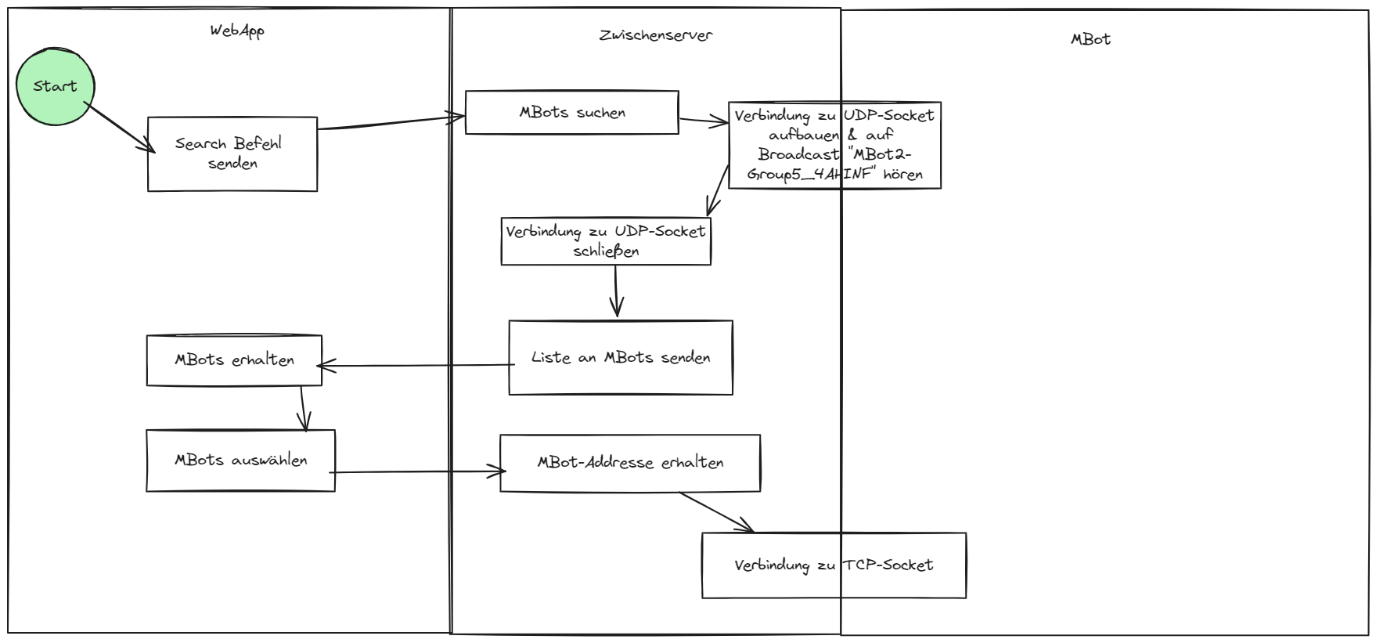
Ein Bild, das Diagramm, Text, Reihe, Plan enthält.

Automatisch generierte Beschreibung

Der Benutzer schließt die WebApp, dadurch sendet wird „close“ an den Zwischenserver gesendet. Der Zwischenserver sendet darauf „disconnect“ an den MBot und schließt den TCP-Socket & den WebSocket. Der MBot „restartet“ dadurch sein Script und bereitet sich für die nächste Verbindung vor.

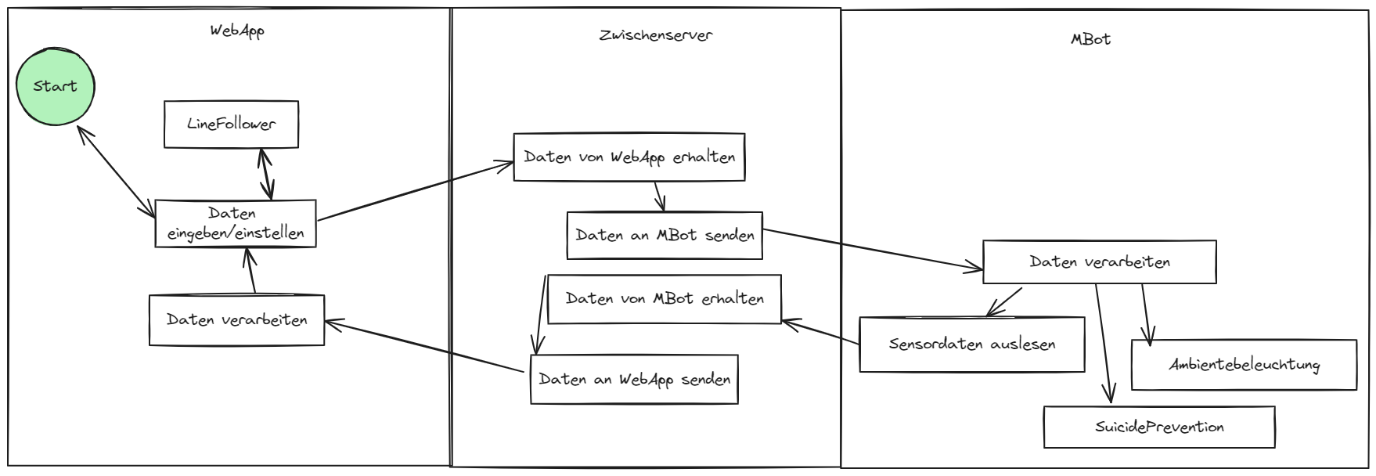
## Sequenzdiagramme

### Sequenzdiagramm MBot Verbinden



Die WebApp sendet einen Search Befehl an den MBot, dieser sucht dann über einen UDP-Socket nach dem Broadcast „MBot2-Group05\_4AHINF“ und gibt dann die gefundenen MBots zurück. Dort wird dann ein MBot ausgewählt und dessen Addresse an den Zwischenserver gesendet. Der Zwischenserver verbindet sich jetzt über diese Addresse mit dem TCP-Socket des MBot.

### Sequenzdiagramm Daten Senden



Der MBot verarbeitet Userdaten und sendet diese dann an den Zwischenserver weiter, dieser sendet die Daten sofort über den TCP-Socket an den MBot weiter. Der MBot verarbeitet die erhaltenen Daten (stellt dadurch die Ambientebeleuchtung, SuicidePrevention, … ein), liest die Sensordaten aus und sendet diese dann zurück an den Zwischenserver. Der Zwischenserver sendet diese Daten dann über den WebSocket an die WebApp weiter. Dieser Kreislauf wiederholt sich automatisch unendlich, bis der Benutzer den MBot trennt oder die WebApp schließt.

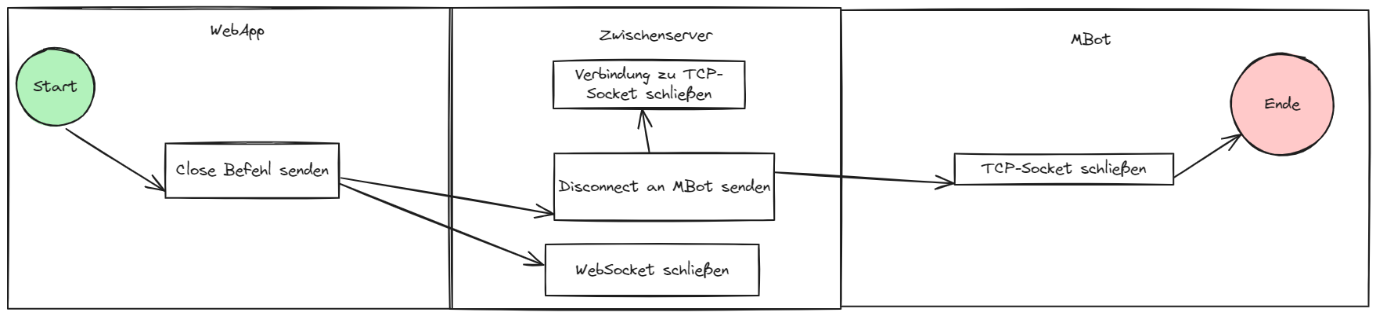
### Sequenzdiagramm Disconnect

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

Automatisch generierte Beschreibung

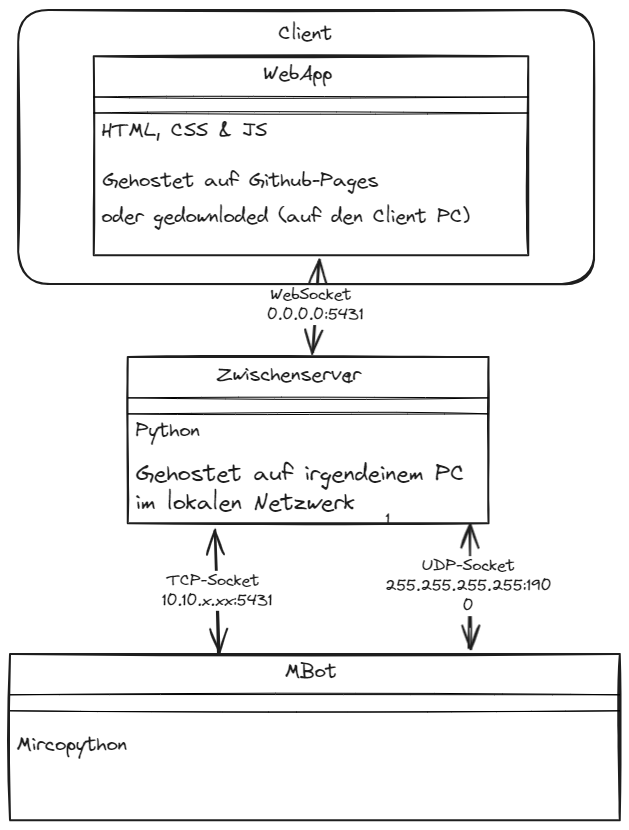
Der Benutzer sendet den „Disconnect“-Befehl an den Zwischenserver, dieser sendet dann eine „disconnect“-Nachricht an den MBot und schließt die TCP-Socket Verbindung zum MBot.

### Sequenzdiagramm Close

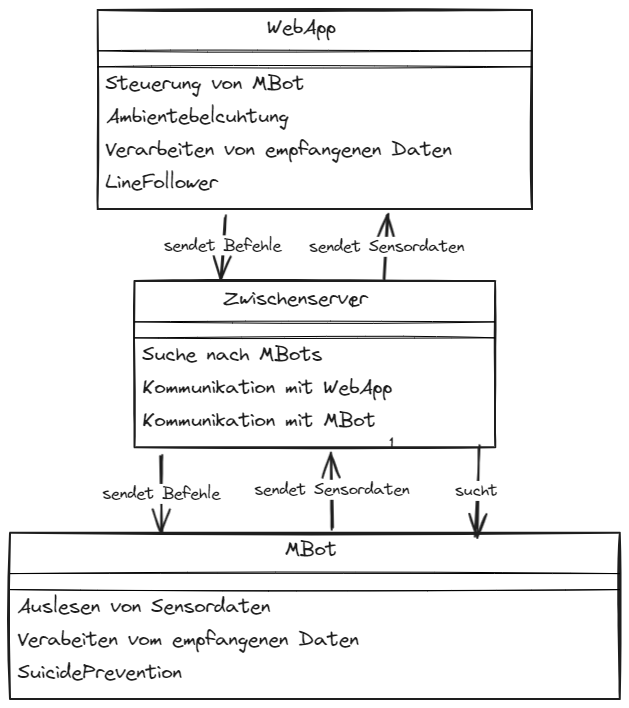


Der Benutzer sendet den „Close“-Befehl an den Zwischenserver, dieser sendet dann eine „disconnect“-Nachricht an den MBot und schließt die TCP-Socket Verbindung zum MBot und den WebSocket.

## Komponentendiagramme



## Verteilungsdiagramme



## Softwarekomponenten / Programme

### SW Programme

PyCharm, Version: 2024.1.1

MBlock, Version 5.4.3

### 5.5.2 SW Komponenten

Python 3.12

Version: 3.12.3

Hersteller: Python Software Foundation

Downloadlink: <https://www.microsoft.com/store/productId/9NCVDN91XZQP?ocid=pdpshare>

Lizenz: <https://docs.python.org/3/license.html>

# Projektdurchführung

## Sprint 1

### Sprintplanung

**Dauer:** 07.02.2024 – 28.02.2024

**Ausgewählte User Stories:**

**Bearbeiter:** Stefan Rautner

**ID:** 6

**Name:** Automatische Verbindung zum WLAN nachdem Mbot eingeschalten wurde

**Bearbeiter:** Stefan Rautner

**ID:** 7

**Name:** Steuerung des Mbot vom Client über den Zwischenserver eingeschalten wurde

**Bearbeiter:** Stefan Rautner

**ID:** 8

**Name:** Verbindung der WebApp mit dem Mbot über den Zwischenserver

eingeschalten wurde

**Bearbeiter:** Stefan Rautner

**ID:** 9

**Name:** WebApp automatisch mit Zwischenserver verbinden

eingeschalten wurde

**Bearbeiter:** Stefan Rautner

**ID:** 10

**Name:** Implementierung eines LineFollower für den Mbot eingeschalten wurde

**Bearbeiter:** Stefan Rautner

**ID:** 2

**Name:** Implementierung einer SuicidePrevention für den Mbot

wurde

Bearbeiter: Tobias Haas

**ID:** 3

**Name:** WebApp mit mobilen Geräten kompatible machen

eingeschalten wurde

**Anzahl geplante Story points:** 120

**Geschaffte Story points:** 45

### Sprint Demo

Die Eingabemethoden: WASD, Pfeil-Tasten, Touch & Controller sind implementiert worden und funktionieren

Die Verbindung vom MBot zum WLAN funktioniert

Die Steuerung des MBot ist begonnen, aber nicht abgeschlossen worden

Die Verbindung des MBot ist begonnen, aber nicht abgeschlossen worden

Die WebApp verbindet sich automatisch mit dem Zwischenserver

Der LineFollower wurde nicht implementiert

Die SuicidePrevention wurde begonnen

Es wurde begonnen, eine Unterstützung der WebApp für mobile Geräte zu erstellen

### Sprint Retrospektive

Da er der Erste Sprint war, hatten wir noch keine Erfahrung, wie viel wir in einem Sprint machen können.

Ich (Stefan Rautner) habe mich in diesem Sprint überschätzt und bin dadurch nur mit ca. 50% meiner Aufgaben fertig geworden.

Die User Story von Tobias Haas ist rein technisch nicht möglich, dass Sie in diesem Sprint fertig wird, weil die Grafische Oberfläche im Verlauf der zukünftigen Sprints immer wieder aktualisiert werden wird.

### Sprint Zusammenfassung

Grundsätzlich ist der Sprint gut gelaufen, aber ich (Stefan Rautner) habe mich sehr hoch verschätzt.

Es wurden keine neuen User Stories hinzugefügt, da es sich um den Ersten Sprint handelt.

Es wurden ebenfalls keine User Stories entfernt.

Auf Basis dieser Sprint Velocity ist das Projekt früher als erwartet fertig.

Die Durchschnittliche Sprint Velocity beträgt 45.

## Sprint 2

### Sprintplanung

**Dauer:** 29.02.2024 – 20.03.2024

**Ausgewählt User Stories:**

**Bearbeiter:** Stefan Rautner

**ID:** 11

**Name:** Mbot mit der Zeit immer schneller fahren lassen

**Bearbeiter:** Stefan Rautner

**ID:** 12

**Name:** Client zeigen, dass Mbot mit WLAN verbunden ist

**Bearbeiter:** Patrick Thor & Stefan Rautner

**ID:** 13

**Name:** Ausrichtung des Mbot durch 3D-Model mit dessen Gyrodaten auf dem Client anzeigen

**Bearbeiter:** Tobias Haas

**ID:** 14

**Name:** Grafische Oberfläche für die WebApp

**Anzahl geplanter User Stories:** 31

**Geschaffte Story Points:** 21

### Sprint Demo

Der MBot beschleunigt jetzt je länger eine Taste gedrückt wird oder je weiter der Stick des Controllers in eine Richtung gedrückt wird

Dem Client wird jetzt angezeigt, dass der MBot mit dem WLAN verbunden ist

Die WebApp beinhält jetzt ein 3D-Model welches abhängig von den Gyrodaten des MBot geupdatet wird

Die Grafische Oberfläche der WebApp wurde verbessert

### Sprint Retrospektive

In diesem Sprint war die Planung der User Stories deutlich besser.

Die User Story von Tobias Haas ist wie im letzten Sprint ebenfalls nicht so schnell zu erledigen, weil die grafisch Oberfläche in folgenden Sprints sicher noch geändert wird.

### Sprint Zusammenfassung

Diesmal ist der Sprint besser ausgefallen und wir haben fast alle Arbeitspakete erfolgreich erledigen können.

Es wurde 2 neue Anforderungen hinzugefügt:

Anzeigen der Farben unter den RGB-Sensoren des MBot während aktivem LineFollower

Die 3D-Anzeige des MBots aufgrund seiner Gyrodaten

Es wurden keine User Stories entfernt

Auf Basis dieser Sprint Velocity ist das Projekt später als erwartet fertig.

Die Durchschnittliche Sprint Velocity beträgt jetzt 33.

## Sprint 3

### Sprintplanung

**Dauer:** 21.03.2024 – 17.04.2024

**Ausgewählt User Stories:**

**Bearbeiter:** Stefan Rautner

**ID:** 23

**Name:** Ausführung des Zwischenservers

**Bearbeiter:** Stefan Rautner

**ID:** 4

**Name:** Daten der 4 RGB-Sensoren bei aktiviertem LineFollower an den Client senden

**Bearbeiter:** Stefan Rautner

**ID:** 5

**Name:** Geschwindigkeit des Mbot durch dessen Beschleunigungssensors an den Client senden

**Bearbeiter:** Tobias Haas

**ID:** 14

**Name:** Grafische Oberfläche für die WebApp

**Anzahl geplanter User Stories:** 41

**Geschaffte Story Points:** 31

### Sprint Demo

Der Zwischenserver ist jetzt ausführbereit und hat alle benötigten Funktionen für unseren Anwendungsfall

Die 4 RGB-Sensoren werden jetzt bei aktivem LineFollower ausgelesen und auf der WebApp angezeigt

Die Geschwindigkeit des MBot wird jetzt ausgelesen und auf der WebApp dargestellt

Die Grafische Oberfläche der WebApp wurde weiter verbessert

### Sprint Retrospektive

In diesem Sprint war die Planung der User Stories relativ gleich wie im Letzten.

Die User Story von Tobias Haas ist wie in den letzten zwei Sprints ebenfalls nicht so schnell zu erledigen, weil die grafisch Oberfläche in folgenden Sprints sicher noch geändert wird.

### Sprint Zusammenfassung

Diesmal ist der Sprint besser ausgefallen und wir haben fast alle Arbeitspakete erfolgreich erledigen können.

Es wurden keine weiteren User Stories hinzugefügt.

Es wurden die User Stories um die Firebase entfernt.

Auf Basis dieser Sprint Velocity wird das Projekt ca. zum Abgabezeitpunkt fertig.

Die Durchschnittliche Sprint Velocity beträgt jetzt 32,33.

## Sprint 4

### Sprintplanung

**Dauer:** 18.04.2024 – 15.05.2024

**Ausgewählt User Stories:**

**Bearbeiter:** Stefan Rautner

**ID:** 20

**Name:** Ambientebeleuchtung auf Mbot durch Client einstellen & einschalten

**Bearbeiter:** Stefan Rautner

**ID:** 21

**Name:** Liniendiagramm mit den Daten der Beschleunigungssensors erstellen

**Bearbeiter:** Stefan Rautner

**ID:** 22

**Name:** Liniendiagramm mit den Daten des Ultraschallsensors erstellen

**Bearbeiter:** Tobias Haas

**ID:** 24

**Name:** Dino-Game implementieren, dass vom Client gespielt werden kann

**Anzahl geplanter User Stories:** 27

**Geschaffte Story Points:** 27

### Sprint Demo

Die Ambientebeleuchtung kann jetzt vom User ausgewählt und geändert werden.

Ebenfalls gibt es jetzt Liniendiagramme für die Daten des Beschleunigungssensors & des Ultraschallsensors (Zeitraum: 5 Sekunden, Abstände zwischen daten: 500ms)

Der User kann jetzt das Google Chrome Dino-Game auf der WebApp spielen

### Sprint Retrospektive

In diesem Sprint war die Planung der User Stories relativ gleich wie in den Letzten zwei.

Die User Story von Tobias Haas ist wie in den letzten zwei Sprints ebenfalls nicht so schnell zu erledigen, weil die grafisch Oberfläche in folgenden Sprints sicher noch geändert wird.

### Sprint Zusammenfassung

Diesmal ist der Sprint sehr gut ausgefallen. Wir haben alle Arbeitspakete erfolgreich abgeschlossen.

Es wurden keine weiteren User Stories hinzugefügt.

Es wurden die User Stories um die Firebase entfernt.

Auf Basis dieser Sprint Velocity wird das Projekt ca. zum Abgabezeitpunkt fertig.

Die Durchschnittliche Sprint Velocity beträgt jetzt 31.

## Sprint 4

### Sprintplanung

**Dauer:** 16.05.2024 – 12.06.2024

**Ausgewählt User Stories:**

**Bearbeiter:** Tobias Haas

**ID:** 15

**Name:** WebApp anschaulich & benutzerfreundlich gestalten

**Bearbeiter:** Stefan Rautner

**ID:** 16

**Name:** Mbot Script ändern, sodass Reset möglich ist

**Bearbeiter:** Stefan Rautner

**ID:** 17

**Name:** Verbindung zu Zwischenserver ändern

**Bearbeiter:** Stefan Rautner

**ID:** 18

**Name:** Dokumentation (NEW)

**Bearbeiter:** Patrick Thor

**ID:** 19

**Name:** Dokumentation (OLD)

**Anzahl geplanter User Stories:** 44

**Geschaffte Story Points:** 44

### Sprint Demo

Die grafische Oberfläche der WebApp wurde in diesem Sprint endgültig fertig gestellt. Ebenfalls wurde das Script des MBot geöndert, sodass dieser (wenn getrennt wird) das Programm neu startet. Ebenfalls wurde die Verbindung zum Zwischenserver geändert und die Unterstützung für Mobilegeräte hinzugefügt.

### Sprint Retrospektive

In diesem Sprint war die Planung der User Stories relativ gleich wie in den Letzten drei.

Die User Story von Tobias Haas ist wie in den Letzten Sprints angegeben jetzt fertig geworden, weil jetzt keine Erneuerungen mehr dazu kommen.

### Sprint Zusammenfassung

Diesmal ist der Sprint sehr gut ausgefallen. Wir haben alle verbleibenden Arbeitspakete erfolgreich abgeschlossen.

Es wurden keine weiteren User Stories hinzugefügt.

Es wurden die User Stories um die Firebase entfernt.

Auf Basis dieser Sprint Velocity wird das Projekt genau zum Abgabepunkt fertig.

Die Durchschnittliche Sprint Velocity beträgt jetzt 33,6.

# Installation / Software deployment

Windows Store öffnen > Python 3.12 suchen > installieren

JetBrains Website öffnen > PyCharm suchen > herunterladen > .exe ausführen

Visual Studio Website öffnen > VS-Code suchen > herunterladen > .exe ausführen

# Projektabschluß

## Projektzusammenfassung

Prinzipiell lief das Projekt gut. Manchmal war die Motivation sehr gering, aber es wurde trotz allem das Projekt erfolgreich fertig gestellt. Die Arbeitsaufteilung im Team war ehrlich gesagt eher mangelhaft, hauptsächlich aufgrund des Engagements einzelner Teammitglieder.

## Attachments

Sprint-Review Präsentationen sind auf GitHub in dem Repository **Dokumente** zu finden

Die Projektbeschreibung ist in dem Repository **Dokumente** zu finden

Quellcode ist in dem Repository **Project** zu finden