HTL Saalfelden

Systemplanung und Projektentwicklung

Projektdokumentation
2023 / 2024

Projektbezeichnung	Mbot-Steuerungssystem
Projektteam	David Legenjovic, Jonas Maier, Daniel Jessner
Erstellt am	10.01.2024
Letzte Änderung am	19.06.2024
Status	Abgeschlossen
Aktuelle Version	2.4

Änderungsverlauf

Nr.	Datum	Version	Geänderte Kapitel	Art der Änderung	Autor
1	10.01.2024	1.0	Alle	Erstellung	David Legenjovic
2	21.03.2024	1.1	Sprint 1, Sprint 2, Sprint3, Projektteam und Schnittstellen	Sprints beschrieben	David Legenjovic
3	03.04.2024	1.2	Nicht-Funktionale Anforderungen, Projektplanung	Nicht-Funktionale Änderungen hinzugefügt, Projektplanung Einleitung + Allgemeine Informationen hinzugefügt	Daniel Jessner
4	10.03.2024	1.3	Projektplanung, Funktionale Anforderungen	+: Variantenbildung, Machbarkeitsstudie, Projektziele, benötigte Ressourcen, Use Case 1	Daniel Jessner
5	17.03.2024	1.4	Projektplanung, Entwicklungsmethodik, Kommunikation, Projektrisiko	+: Entwicklungsmethodik, Kommunikation, Projektrisiko	Daniel Jessner
6	24.04.2024	1.5	Projektplanung, Entwicklungsmethodik, Kommunikation, Projektrisiko	+: Projektumfeldanalyse	Daniel Jessner
7	25.04.2024	1.6	Funktionale Anforderungen, Softwarearchitektur	+: Use Cases, Komponentendiagramm, Verteilungsdiagramm	Daniel Jessner
8	08.05.2024	1.7	Funktionale Anforderungen	Use-Case-Diagramm aktualisiert	Daniel Jessner
9	15.05.2024	1.8	Softwarearchitektur	Sequenzdiagramm 1+2 hinzugefügt	Daniel Jessner
10	29.05.2024	1.9	Softwarearchitektur, Funktionale Anforderungen	Sequenzdiagramm 3 -7 hinzugefügt, Use Case 6 überarbeitet	Daniel Jessner
11	05.06.2024	2.0	Softwarearchitektur	Aktivitätsdiagramme 1-6 hinzugefügt	Daniel Jessner
12	09.06.2024	2.1	Installation, Projektabschluss	Projektzussamenfassung erstellt, Attachments aufgelistet	Daniel Jessner
13	12.06.2024	2.2	Inhalt	Inhaltsverzeichnis erstellt	Daniel Jessner
14	19.06.2024	2.3	Sprint 1-5, Attachements	Verbesserung, Ergänzung	David Legenjovic
15	19.06.2024	2.4	Dokument	PDF, Formatierung	Daniel Jessner

Inhalt

1. Allgemeines / Projektübersicht

1.1 Projektbeschreibung

Dieses Projekt ist ein Semesterprojekt der 4. AHINF der HTL Saalfelden und hat als Ziel, ein System zu entwickeln, welches einen sogenannten „mBot2“ des Unternehmens „makeblock" fernsteuert. Dazu eignet sich eine Dreiteilung in der Entwicklung in die Bereiche Server, Steuerungssoftware und den Mbot an sich. Basisanforderungen des Projektes geben vor, dass die GUI, die dem Benutzer zur Verfügung gestellt wird, sowohl am Windows-System als auch auf mobilen Endgeräten laufen soll. Weiters wird der gesamte Projektverlauf unten dokumentiert und jegliche Änderungen mit Version Control via „git“ festgehalten.

1.2 Projektteam und Schnittstellen

Rolle(n)	Name	Telefon	E-Mail
Client-Entwickler	David Legenjovic	+43 677 61331252	david.legenjovic@htl-saalfelden.at
Server-Entwickler	Jonas Maier	+43 664 75087002	jonas.maier@htl-saalfelden.at
mBot-Entwickler	Daniel Jessner	+43 650 6349636	daniel.jessner@htl-saalfelden.at

2. Funktionale Anforderungen

2.1 Use Cases

2.1.1 MBot mit dem Internet und Server verbinden

- **Akteure:**
  Testperson (als Benutzer), MBot (als Client), (Schul-)Netzwerk (als Zugang zum Internet), Server (als Vermittler zwischen MBot und Steuerungssoftware

- **Auslöser / Trigger-Event:**
  Einschalten des MBots

- **Kurzbeschreibung:**
  Testperson testet den Boot-Vorgang des MBots, welcher sich automatisch zum Internet und zum Server verbindet.

- **Beschreibung der einzelnen Schritte:**

Testperson schaltet, nachdem sie die korrekten Konfigurationen eingetragen hat, den MBot ein, welcher sich daraufhin automatisch mit dem angegeben W-LAN verbindet und dann eine Kommunikation zum Server aufbaut:

- Ein Benutzer startet den Roboter.
- Benutzer trägt Konfigurationen bezüglich Netzwerk, usw. ein.
- mBot verbindet sich zum WLAN und sendet dann einen Broadcast für die Ser ver-Verbindung
- Die Anwendung bekommt automatisch vom Server verfügbare Roboter im Netzwerk.
- Sobald ein Roboter gefunden wird, wird eine Verbindung hergestellt und die IP-Adresse sowie der Name des Roboters in der Anwendung angezeigt.

- **Beschreibung alternativer Schritte:**
  Die Testperson trägt versehentlich inkorrekte Konfigurationen ein (Server oder W-LAN) und ermöglicht es dem MBot so nicht erfolgreiche Verbindungen herzustellen. Dieser erkennt, dass keine Verbindung besteht und versucht in einer Endlosschleife, sich erneut zu verbinden.

- **Vor- und Nachbedingungen**:
  Es soll möglich sein, einfach und intuitiv den MBot erfolgreich zu starten.

- **Systemgrenzen:**
  SW-Produkt, mBot, Server, Internet

2.1.2 Steuerungssoftware starten und mit Server verbinden

- **Akteure:**
  Testperson (als Benutzer), Server (als Vermittler zwischen MBot und Steuerungssoftware), (Schul-)Netzwerk (als Zugang zum Internet)

- **Auslöser / Trigger-Event:**
  Starten der Steuerungssoftware auf einem Computer oder einem mobilen Endgerät.

  Kurzbeschreibung:
  Die Testperson startet die Steuerungssoftware, welche sich automatisch mit dem Server verbindet, um eine Liste der verfügbaren MBots im Netzwerk abzurufen.

- **Beschreibung der einzelnen Schritte:**
  - Die Testperson startet die Steuerungssoftware auf ihrem Computer oder mobilen Endgerät.
  - Die Steuerungssoftware versucht automatisch eine Verbindung zum Server herzustellen (Broadcast).
  - Der Server empfängt die Anfrage der Steuerungssoftware und antwortet mit einer Liste der verfügbaren MBots im Netzwerk.

- Die Steuerungssoftware zeigt die verfügbaren MBots mit ihren Namen und IP-Adressen an.

- **Beschreibung alternativer Schritte:**
  Falls der Server nicht erreichbar ist oder keine MBots im Netzwerk gefunden werden, zeigt die Steuerungssoftware eine entsprechende Fehlermeldung an und bietet gegebenenfalls die Möglichkeit, manuell eine Verbindung herzustellen oder erneut zu versuchen.

- **Vor- und Nachbedingungen:**
  Die Steuerungssoftware ist erfolgreich gestartet und hat eine Verbindung zum Server hergestellt.

- **Systemgrenzen:**
  SW-Produkt, Steuerungssoftware, Internet, Server

2.1.3 Anzeigen von Sensordaten in der Anwendung

- **Akteure:**
  Benutzer, mBot (als Datenquelle), Steuerungssoftware (als Anwendung), Server (als Vermittler)

- **Auslöser / Trigger-Event:**
  Aktualisierung der Sensordaten auf dem MBot.

- **Kurzbeschreibung:**
  Die Steuerungssoftware zeigt in Echtzeit die aktuellen Sensordaten des MBots an.

- **Beschreibung der einzelnen Schritte:**
  - Die Testperson hat die Steuerungssoftware gestartet und eine Verbindung zum MBot hergestellt.
  - Der MBot erfasst kontinuierlich Daten von seinen Sensoren, wie beispielsweise Ultraschallabstand, Lichtstärke usw.
  - Die Steuerungssoftware fordert periodisch die aktuellen Sensordaten vom MBot an.
  - Der MBot sendet die aktuellen Sensordaten an die Steuerungssoftware.
  - Die Steuerungssoftware zeigt die empfangenen Sensordaten in einer benutzerfreundlichen Darstellung an, z.B. in Form von Diagrammen, Zahlenwerten oder grafischen Elementen.

- **Beschreibung alternativer Schritte:**
  Falls die Verbindung zwischen MBot und Steuerungssoftware unterbrochen wird, zeigt die Anwendung eine entsprechende Fehlermeldung an und bietet gegebenenfalls die Möglichkeit, die Verbindung wiederherzustellen.

- **Vor- und Nachbedingungen:**

Die Steuerungssoftware ist erfolgreich gestartet und hat eine Verbindung zum MBot hergestellt. Die Sensordaten werden erfolgreich und in Echtzeit in der Anwendung angezeigt.

- **Systemgrenzen:**
  SW-Produkt, mBot, Steuerungssoftware, Internet, Server

2.1.4 Fernsteuerung des mBot

- **Akteure:**
  Testperson (als Benutzer), MBot (als zu steuerndes Objekt), Steuerungssoftware (als Anwendung), Server (Vermittler)

- **Auslöser / Trigger-Event:**
  Benutzer möchte den MBot über die Steuerungssoftware fernsteuern.

- **Kurzbeschreibung:**
  Die Testperson steuert den MBot über die Steuerungssoftware, indem sie Befehle zur Bewegung und Geschwindigkeit des MBots sendet.

- **Beschreibung der einzelnen Schritte:**
  - Die Testperson hat die Steuerungssoftware gestartet und eine Verbindung zum MBot hergestellt.
  - Die Testperson wählt die gewünschten Bewegungsrichtungen (vorwärts, rückwärts, links, rechts) und die Geschwindigkeit aus.
  - Die Steuerungssoftware sendet die ausgewählten Befehle an den MBot.
  - Der MBot empfängt die Befehle und führt die entsprechenden Bewegungen gemäß den Anweisungen aus.
  - Die Testperson kann die Bewegungen des MBots über die Anwendung verfolgen.

- **Beschreibung alternativer Schritte:**
  Falls die Verbindung zwischen Steuerungssoftware und MBot unterbrochen wird, zeigt die Anwendung eine entsprechende Fehlermeldung an und bietet gegebenenfalls die Möglichkeit, die Verbindung wiederherzustellen.

- **Vor- und Nachbedingungen:**
  Die Steuerungssoftware ist erfolgreich gestartet und hat eine Verbindung zum MBot hergestellt. Die Testperson kann den MBot erfolgreich über die Anwendung fernsteuern.

- **Systemgrenzen:**
- SW-Produkt, mBot, Steuerungssoftware, Internet, Server

2.1.5 Aktivierung des Sicherheitsmodus

- **Akteure:**
  Testperson (als Benutzer), MBot (als zu schützendes Objekt), Steuerungssoftware (als Anwendung)

- **Auslöser / Trigger-Event:**
  Benutzer möchte den Sicherheitsmodus des MBots aktivieren, um versehentliche Kollisionen zu verhindern.

- **Kurzbeschreibung:**
  Die Testperson aktiviert den Sicherheitsmodus des MBots über die Steuerungssoftware, um sicherzustellen, dass der MBot nicht mutwillig gegen Hindernisse fährt.

- **Beschreibung der einzelnen Schritte:**
  - Die Testperson hat die Steuerungssoftware gestartet und eine Verbindung zum MBot hergestellt.
  - Die Testperson sucht in der Anwendung die Option zum Aktivieren des Sicherheitsmodus.
  - Die Testperson aktiviert den Sicherheitsmodus durch Auswahl der entsprechenden Option.
  - Die Steuerungssoftware sendet den Befehl zur Aktivierung des Sicherheitsmodus an den mBot:
  - Der mBot empfängt den Befehl und und schaltet den Sicherheitsmodus ein, wobei er vor dem Ausführen eines Befehls prüft, ob Objekte ihn frontal behindern und gegebenfalls ein auditives Signal abgibt und ein Stück zurückfährt.

- **Beschreibung alternativer Schritte:**
  Falls die Verbindung zwischen Steuerungssoftware und MBot unterbrochen wird, kann der Sicherheitsmodus möglicherweise nicht aktiviert werden. Die Anwendung zeigt eine entsprechende Fehlermeldung an und bietet gegebenenfalls die Möglichkeit, die Verbindung wiederherzustellen.

- **Vor- und Nachbedingungen:**
  Die Steuerungssoftware ist erfolgreich gestartet und hat eine Verbindung zum MBot hergestellt. Der Sicherheitsmodus des MBots ist erfolgreich aktiviert.

- **Systemgrenzen:**
- SW-Produkt, mBot, Steuerungssoftware, Internet, Server

2.1.6 Aktivierung des Linien-Folgemodus

- **Akteure:**
  Testperson (als Benutzer), MBot (als zu steuerndes Objekt), Steuerungssoftware (als Anwendung)

- **Auslöser / Trigger-Event:**
  Benutzer möchte den Linien-Folgemodus des MBots aktivieren, um ihm automatisch einer Linie folgen zu lassen.

- **Kurzbeschreibung:**
  Die Testperson aktiviert den Linien-Folgemodus des MBots über die Steuerungssoftware, damit der MBot automatisch einer Linie folgt.
- **Beschreibung der einzelnen Schritte:**
  - Die Testperson hat die Steuerungssoftware gestartet und eine Verbindung zum Server hergestellt.
  - Die Testperson sucht in der Anwendung die Option zum Aktivieren des Linien-Folgemodus.
  - Die Testperson aktiviert den Linien-Folgemodus durch Auswahl der entsprechenden Option.
  - Die Steuerungssoftware sendet den Befehl zur Aktivierung des Linien-Folgemodus an den Server.
  - Der Server empfängt den Befehl und schaltet in den Sicherheitsmodus um, wodurch versehentliche Kollisionen verhindert werden. Dabei übernimmt er die Steuerung des mBots und überprüft vor jedem Senden eines Befehls die frotnalen Lichtsensoren, um zu sehen, ob der mBot auf dunklem Untergrund ist und dementsprechend die Befehle anzupassen.

- **Beschreibung alternativer Schritte:**
  Falls die Verbindung zwischen Steuerungssoftware und MBot unterbrochen wird, kann der Linien-Folgemodus möglicherweise nicht aktiviert werden. Die Anwendung zeigt eine entsprechende Fehlermeldung an und bietet gegebenenfalls die Möglichkeit, die Verbindung wiederherzustellen.

- **Vor- und Nachbedingungen:**
  Die Steuerungssoftware ist erfolgreich gestartet und hat eine Verbindung zum MBot hergestellt. Der Linien-Folgemodus des MBots ist erfolgreich aktiviert.

- **Systemgrenzen:**
- SW-Produkt, mBot, Steuerungssoftware, Internet, Server

2.1.7 Steuern der LEDs am mBot

- **Akteure:**
  Testperson (als Benutzer), MBot (als zu steuerndes Objekt), Steuerungssoftware (als Anwendung)

- **Auslöser / Trigger-Event:**
  Benutzer möchte die LEDs am MBot über die Steuerungssoftware steuern.

- **Kurzbeschreibung:**
  Die Testperson steuert die LEDs am MBot über die Steuerungssoftware, um verschiedene Farben einzustellen.

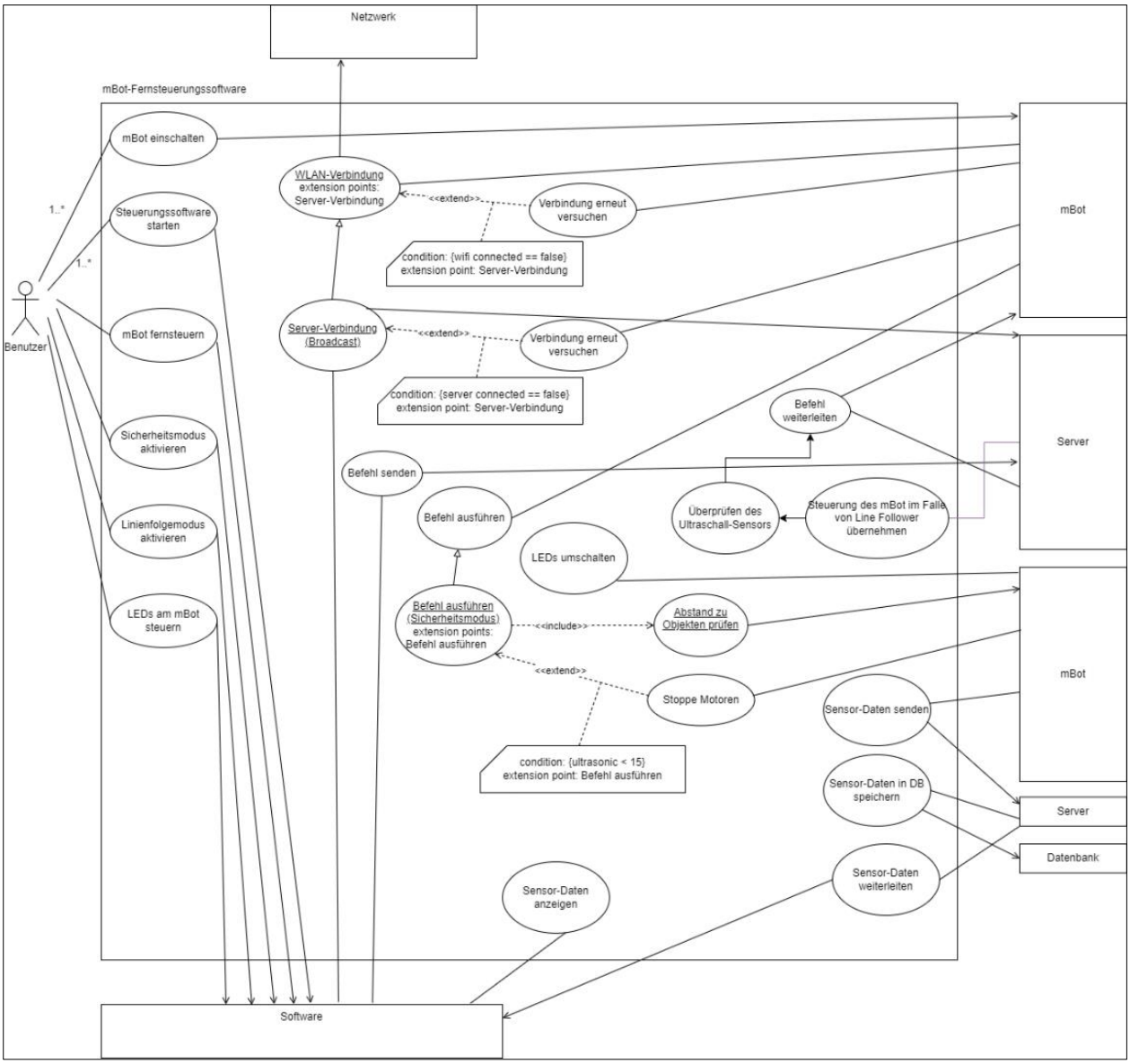
- **Beschreibung der einzelnen Schritte:**
  - Die Testperson hat die Steuerungssoftware gestartet und eine Verbindung zum MBot hergestellt.
  - Die Testperson sucht in der Anwendung die Option zum Steuern der LEDs.
  - Die Testperson wählt die gewünschten Farben für die LEDs aus und bestätigt die Auswahl.
  - Die Steuerungssoftware sendet die ausgewählten Farben an den MBot.
  - Der MBot empfängt die Befehle und ändert die Farben seiner LEDs entsprechend den Anweisungen.
- **Beschreibung alternativer Schritte:**
  Falls die Verbindung zwischen Steuerungssoftware und MBot unterbrochen wird, können die LEDs möglicherweise nicht gesteuert werden. Die Anwendung zeigt eine entsprechende Fehlermeldung an und bietet gegebenenfalls die Möglichkeit, die Verbindung wiederherzustellen.

- **Vor- und Nachbedingungen:**
  Die Steuerungssoftware ist erfolgreich gestartet und hat eine Verbindung zum MBot hergestellt. Die LEDs am MBot sind erfolgreich gemäß den Anweisungen der Testperson eingestellt.

- **Systemgrenzen:**
  SW-Produkt

2.1.8 Use-Case-Diagramm

Folgendes Diagramm umfasst die oben genannten Use Cases im Systemkontext:



3. Nichtfunktionale Anforderungen

3.1 Benutzerfreundlichkeit

Die GUI der Steuerungssoftware sollte intuitiv gestaltet sein, um eine einfache Bedienung für Benutzer aller Erfahrungsstufen zu gewährleisten. Außerdem sollte die Reaktionszeit der Steuerungssoftware sowie die Latenz zwischen Software und Mbot so gering wie möglich sein, um ein reibungsloses und ansprechendes Benutzererlebnis zu ermöglichen.

3.2 Zuverlässigkeit

Es ist ein wichtiger Punkt, dass die Verbindung zwischen Mbot und der Software, also auch über den Server, stabil ist, damit eine konstante, ununterbrochene Steuerung ohne Probleme zur Verfügung gestellt werden kann. Weiters soll das System so weit robust sein, dass es angemessen auf Ausnahmesituationen reagiert, um einen kontinuierlichen Betrieb auch bei unvorhergesehenen Ereignissen sicherzustellen.

3.3 Skalierbarkeit

Es ist gewünscht, das System so zu gestalten, dass es auch in der Zukunft liegende Erweiterungen und zusätzliche Änderung ohne großen Aufwand unterstützt. Die Software sollte die Funktionalität haben, dem Benutzer mehrere gefundene Mbots zur Auswahl zu stellen, ohne dabei die Perfomance oder Stabilität des Systems zu beeinträchtigen.

3.4 Sicherheit

Die Kommunikation zwischen Mbot, Server und Software sollte sicher und privat sein, um äußeren Einfluss beziehungsweise Manipulation zu verhindern.

3.5 Plattformkompatibilität

Wie auch die Grundanforderungen des Projektes vorgeben, sollte die Software derartig unabhängig sein, dass die Mbot-Steuerung sowohl in einer Windows-Umgebung als auch auf mobilen Endgeräten (Android) funktioniert. Dabei ist die Responsivität der Software von großer Wichtigkeit, um eine konsistente Darstellung zu jeder Zeit zu gewährleisten.

3.6 Dokumentation und Wartbarkeit

Das Projekt sollte ausführlich dokumentiert sein, einschließlich Anleitungen zur Installation, Konfiguration und Verwendung der Steuerungssoftware. Dazu gehört auch eine gute

Strukturierung sowie Kommentierung des Quellcodes, um eine einfache Wartung und Weiterentwicklung des Systems zu ermöglichen.

4. Projektplanung

Die Projektplanung ist ein entscheidender Schritt bei der Durchführung eines jeden Projekts, da sie den Rahmen setzt, innerhalb dessen das Projekt durchgeführt wird, und die Grundlage für einen erfolgreichen Abschluss bildet. Dieser Prozess ermöglicht es, die Ziele, Anforderungen, Ressourcen und Zeitpläne zu definieren und zu organisieren, um sicherzustellen, dass das Projekt effizient und effektiv umgesetzt wird.

Für das vorliegende Projekt ist eine gründliche Projektplanung von entscheidender Bedeutung. Das Ziel dieser Planung ist es, die Schritte und Maßnahmen festzulegen, die erforderlich sind, um das Projekt erfolgreich abzuschließen, sowie potenzielle Risiken zu identifizieren und Strategien zu entwickeln, um ihnen zu begegnen.
Die Projektplanung wird sich auf verschiedene Aspekte konzentrieren, darunter die Definition der Projektziele, die Bestimmung der benötigten Ressourcen, die Festlegung eines Zeitplans und die Identifizierung von Verantwortlichkeiten und Zuständigkeiten innerhalb des Projektteams. Darüber hinaus wird die Planung auch die Auswahl der geeigneten Entwicklungsmethodik, die Kommunikations- und Berichterstattungsstrategien sowie die Risikobewertung und -management umfassen.

4.1 Variantenbildung

- Mögliche Varianten:
  - Dreiteilung (mBot, Software, Server)
  - Zweiteilung (mBot, Software+Server)
  - Mit Datenbank, ohne Datenbank
  - Nosql, Mysql

Schlussendlich wurde, wie schon kurz erwähnt, die Dreiteilung mit Datenspeicherung in einer Mongo-Datenbank gewählt, da dieser Weg es ermöglicht, sich jeweils voll und ganz auf einen Bereich zu konzentrieren. Der mBot kümmert sich um die Verarbeitung der Befehle, der Server um die Weiterleitung (Schnittstelle zwischen Software und mBot) und die Steuerungssoftware um die Bedienung durch den Benutzer.

Weiters hat sich das Projektteam für eine einfache, unkomplizierte und schnelle Datenspeicherung für die Nosql-Variante mittels MongoDB entschieden, um das gespeicherte Datenmodell möglichst flexible zu halten und nicht erforderlichen Aufwand einer SQL-Datenbank bezüglich Beziehungen, etc. zu vermeiden.

4.2 Machbarkeitsstudie

Mittels Übungsstunden vor dem Projektstart wurde dem Projektteam der mBot und die mBlock-Umgebung nähergebracht und erste Versuche damit angestellt. Es kristallisierten sich zwei Teilbereiche der Machbarkeitsanalyse also schon im Vorhinein heraus:

- **Technische Machbarkeit:**
  - Die erforderliche Hardware (mBot-Roboter, Sensoren) ist kommerziell erhältlich und relativ kostengünstig.
  - Programmierumgebungen wie Scratch und Arduino IDE oder die konzerneigenen „mBlock“-Umgebung bieten eine benutzerfreundliche Plattform für die Entwicklung und Programmierung des Roboters.
  - Die grundlegenden Funktionen (Hindernisvermeidung, Linienverfolgung) sind technisch machbar und wurden in ähnlichen Projekten bereits erfolgreich umgesetzt.
  - Der Microcontroller des mBot ist mit Micropython kompatibel und lässt einen gewissen Funktionsbereich von Python zu.

- **Zeitliche Machbarkeit:**
  - Die Montage des mBot-Roboters erfordert in der Regel weniger als eine Stunde (in diesem Projekt nicht von Entwickelnden durchgeführt).
  - Die Programmierung der grundlegenden Funktionen kann je nach Kenntnisstand und Erfahrung des Entwicklers einige Stunden bis Tage in Anspruch nehmen.
  - Testen und Feinabstimmung des Roboters erfordern zusätzliche Zeit, um sicherzustellen, dass er zuverlässig funktioniert.
  - Trotz oben angeführten Punkten ist das Projektteam davon überzeugt, dass genügend Zeit zur Verfügung steht

Grundsätzlich ist das Projekt in der Machbarkeit als sehr gut zu bewerten, da im Vorhinein keine besorgniserrengenden oder möglich problematische Aspekte erkannt werden können.

Allgemeine Planungsinformationen

Hier werden andere Planungsinformationen angeführt, welche nicht durch andere Kapitel abgedeckt sind, jedoch durchaus für das vorliegende Projekt von Bedeutung sind.

4.2.1 Projektziele

Grundlegende Steuerungsfunktionen:
Implementierung von Basissteuerungsfunktionen wie Vorwärtsfahren, Rückwärtsfahren, Links- und Rechtsdrehungen, um eine grundlegende Navigation des mBot-Roboters zu ermöglichen (hier durch einen Joy Stick geregelt).

Hindernisvermeidung:
Entwicklung eines Systems zur Hindernisvermeidung mithilfe von Ultraschallsensoren, um den mBot autonom navigieren zu lassen und Zusammenstöße mit Hindernissen zu vermeiden.

Linienverfolgung:
Integration eines Linienverfolgungssystems, das es dem mBot ermöglicht, einer Linie auf dem Boden zu folgen und sich entlang vorgegebener Pfade zu bewegen. Dieser Modus ist partikulär durch Basisanforderungen an die Steuerungssoftware gebunden.

Benutzerinteraktion:
Implementierung einer Benutzerschnittstelle zur Steuerung des mBot über die Steuerungssoftware. Diese Schnittstelle wird in diesem Projekt als Joy Stick realisiert.

Modularität und Erweiterbarkeit:
Design des Steuerungssystems mit einer modularen Architektur, die es ermöglicht, neue Sensoren, Aktuatoren oder Steuerungsfunktionen einfach hinzuzufügen oder zu entfernen, um die Funktionalität des mBot zu erweitern.

Robuste Leistung:
Gewährleistung einer robusten Leistung des Steuerungssystems unter verschiedenen Umgebungsbedingungen, einschließlich unterschiedlicher Bodenbeschaffenheiten, Lichtverhältnisse und Hinderniskonfigurationen, so weit wie die Hardware es zulässt.

Einfachheit und Benutzerfreundlichkeit:
Entwicklung eines benutzerfreundlichen Steuerungssystems mit einer intuitiven Benutzeroberfläche und klaren Anweisungen, um die Konfiguration und Interaktion mit dem mBot für Benutzer aller Erfahrungsstufen zugänglich zu machen.

Dokumentation:
Bereitstellung umfassender Dokumentation, einschließlich Anleitungen, um Benutzern bei der Installation, Konfiguration und Nutzung des Steuerungssystems zu unterstützen.

Sicherheit und Zuverlässigkeit:
Priorisierung von Sicherheit und Zuverlässigkeit bei der Entwicklung des Steuerungssystems, um sicherzustellen, dass der mBot ordnungsgemäß funktioniert und potenzielle Risiken oder Gefahren minimiert werden.

4.2.2 Benötigte Ressourcen

mBot-Roboter:
Der mBot-Roboter ist die zentrale Hardwarekomponente des Projekts. Er besteht aus Motoren, Rädern, einem Mikrocontroller, Sensoren und anderen elektronischen Komponenten.

Sensoren:
Die Sensoren des mBot sind Bauteile die am Gerät selbst, welche dem System interessante Daten bereitstellen, welche in der Datenbank abgespeichert werden und dem Benutzer an GUI anschaulich präsentiert werden. Diese können modular ergänzt werden. Sensoren, welche in diesem Projekt zum Einsatz kommen, sind:

- Ultraschallsensor:
  Zur Hindernisvermeidung und zur Messung von Entfernungen.

- Lichtsensor:
  Für Lichtmessungen oder zur Erkennung von Umgebungslichtbedingungen.

- Front-Light-Sensors:
  Insgesamt vier Stück zur Messung der Untergrundfarbe. Wird für Linienverfolgungsmodus verwendet.

- Lage-Sensoren (pitch, yaw, roll):
  Insgesamt drei Sensoren, welche Daten über Neigung, Drehung um x-Achse und Drehung um y-Achse liefern. Diese helfen dabei, die Lage des mBot visuell in der Steuerungssoftware darzustellen.

- Geräuschsensor:
  Für die Messung von Umgebungsgeräuschen.

- Erschütterungssensor:
  Misst das Maß an Erschütterung, das den mBot erreicht. Je größer der gemessene Wert, desto stärker wird der mBot durch Erschütterung beeinflusst („gewackelt“).

Programmierbare Hardwareplattform:
Für dieses Projekt fiel die Wahl auf den mBot. Dieser bietet mit dem CyberPi eine programmierbare Hardwareplattform, die eine benutzerfreundliche Programmierung des mBots ermöglicht und mit verschiedenen Sensoren und Aktuatoren kompatibel ist.

Anderweitig wäre auch noch „mCore“ als Alternative dazu existent, ist für dieses Projekt aber nicht von Relevanz.

Programmiersoftware:
Neben der berücksichtigten Arduino IDE, welche eine populäre Entwicklungsumgebung für die Programmierung des mBot in der Arduino-Sprache bereitstellt, hat sich das Projektteam für die vom mBot-Hersteller veröffentliche Umgebung „mBlock“ entschieden, da diese mit dem integrierten Python-Editor eine einfache und schnelle Möglichkeit bietet, Programme für den mBot in Micro Python zu entwickeln.

Zusätzliche Bauteile:
Obwohl das Projektteam an sich nicht am Zusammenbau des mBot beteiligt war, sind durchaus die einzelnen Komponenten am Gerät zu erkennen:

- **Batterien**:
  Zur Stromversorgung des mBot-Roboters und anderer Komponenten.

- **Verbindungskabel**: Zum Anschließen von Sensoren, Aktuatoren und anderen elektronischen Bauteilen, aber auch für die Verbindung zur mBlock-DIE.

- **Gehäuse oder Rahmen**: Optional für die Montage und den Schutz des mBot-Roboters und seiner Komponenten.

Dokumentation und Anleitungen:
Während des Projektzeitraumes hat sich das Entwicklerteam regelmäßig externen Input geholt, um Probleme zu lösen oder einfach Informationen über gewisse Funktionalitäten zu bekommen:

- Handbücher:
  Zu den mBot-Roboterbausätzen und den einzelnen Sensoren (auch direkt von mBlock zur Verfügung gestellt)

- Online-Tutorials:
  Anleitungen und Informationen nicht nur über die Programmierung des mBot, sondern auch zu Avalonia (Software) und Springboot (Server)

- Community-Foren:
  Online-Plattformen, auf denen Benutzer Fragen stellen, Erfahrungen austauschen und Unterstützung erhalten können (bsp.: stackoverflow).

Zeit und Engagement:
Das Projektteam berechnet Zeit sowohl für die Programmierung des Steuerungssystems und die Durchführung von Tests und Feinabstimmungen als auch für das Engagement für kontinuierliches Lernen und Experimentieren mit neuen Funktionen und Technologien.

Berücksichtigt man die 3 Übungsstunden pro Woche, die für das Projekt vorgesehen sind, ergeben sich also grundsätzlich 6h/Sprint.

4.3.3 Entwicklungsmethodik

Die Entwicklungsmethodik bezieht sich auf die Wahl des Ansatzes, der nicht nur für die Durchführung, sondern auch für die Planung des Projektes verwendet wird. Dieses Projekt wird anhand der agilen Projektmanagement-Methode „**Scrum**“ geplant, geleitet und bewertet.

Das bedeutet: Anforderungen werden auf User Stories heruntergebrochen, geschätzt (in Value Points & Story Points) und bei den jeweiligen Sprintplanungen (alle 2 Wochen) vom

Product Backlog in das Sprint Backlog übernommen. Am Ende eines Sprints findet ein Sprint Review und (optional) eine Sprint Retrospektive.

Ausschlaggebend bei dieser Methode ist der kontinuierliche Kontakt zum Kunden. In diesem Fall wird am Ende eines Sprints der Projektfortschritt den leitenden Lehrpersonen präsentiert und sozusagen direktes Feedback von den „Kunden“ eingeholt.

4.3.4 Kommunikations- und Berichterstattungsstrategie

Wie schon im obigen Punkt „Entwicklungsmethodik“ erwähnt wurde, werden alle 2 Wochen alle Projektfortschritte den Stakeholdern präsentiert. In diesen Präsentationen wird darauf eingegangen, welche Sprintziele erreicht wurden und welche nicht, welche User Stories im Sprint Backlog verbleiben und an den nächsten Sprint weitergegeben werden und welche erledigt sind und auch wie viele Story Points insgesammt noch im Product Backlog verbleiben. Zusätzlich werden der Projektverlauf und die Sprint Velocity („Schnelligkeit des Entwicklungsteams“) in einem Burndown-Chart und einem Säulendiagramm dargestellt.

Live-Demo:
Um den Stakeholdern auch effektiv vor Augen zu führen, was sich im Projekt im Laufe des Sprints getan hat, wird eine Live-Demo zur Verfügung gestellt. Auf diese Weise können die Stakeholder den Fortschritt des Projekts auch auf persönlicher Ebene bewerten und Feedback abgeben.

Dokumentation:
Das ganze Projekt wird laufend in einer Dokumentation festgehalten, welche mit jedem Sprint aktualisiert wird. Diese könnte während, oder auch am Ende des Projektes bei möglich auftretenden Fragen helfen und den Stakeholdern eventuell einen besseren Einblick gewähren.

4.3.5 Projektrisiko(-bewertung)

Da das hier angeführte Projekt ein Schul-Projekt darstellt und in keinster Art und Weise kommerzielle oder finanzielle Auswirkungen mitsichzieht, ist ein solches Projektrisiko eher milder zu bewerten. Trotzdessen gibt es auch hier einen Punkt, der durchaus zu berücksichtigen ist:

Mangelnde Verfügbarkeit von Ressourcen:
Beschreibung:
Das Risiko besteht darin, dass möglicherweise nicht genügend Ressourcen wie Zeit, Materialien oder „Personal“ für das Schulprojekt zur Verfügung stehen.

Ursachen:
Dieses Risiko kann durch verschiedene Faktoren verursacht werden, darunter unvorhergesehene Verzögerungen oder unerwartete Änderungen in den Projektanforderungen.
(*Projektanforderungen wurden 1x durch Stakeholder ergänzt)

Auswirkungen:
Wenn nicht genügend Ressourcen vorhanden sind, könnte dies zu Verzögerungen bei der Durchführung des Projekts führen, die Qualität des Endprodukts beeinträchtigen oder sogar dazu führen, dass das Projekt nicht rechtzeitig abgeschlossen wird.

Risikobewertung:
Die Wahrscheinlichkeit dieses Risikos hängt von verschiedenen Faktoren ab, darunter die Komplexität des Projekts, die Verfügbarkeit von Ressourcen und die Erfahrung des Projektteams.

Risikominderungsstrategien:
Um das Risiko mangelnder Ressourcen zu mindern, könnten Maßnahmen ergriffen werden wie die frühzeitige Identifizierung und Planung von benötigten Ressourcen, die Aufstellung eines realistischen Zeitplans, die Suche nach zusätzlichen Finanzierungsquellen oder die Einbindung zusätzlicher Unterstützungspersonen, Sprichwort: **Scrum**.

Verantwortlichkeiten:
Die Projektverantwortlichen sind dafür verantwortlich, das Risiko zu überwachen, frühzeitig auf Anzeichen von Ressourcenmangel zu reagieren und geeignete Maßnahmen zur Risikominderung zu ergreifen.

4.3 Projektumfeldanalyse

Zielsetzung des Projekts:
Das Hauptziel des Projekts ist die Entwicklung eines Systems zur Fernsteuerung des mBot2 von Makeblock. Die Aufgabe ist in drei Hauptbereiche unterteilt: Server, Steuerungssoftware und mBot2.

Anforderungen an das Projekt:
- Das System soll eine grafische Benutzeroberfläche (GUI) haben, die sowohl auf Windows-Systemen als auch auf mobilen Endgeräten funktioniert.
- Es müssen Basisfunktionalitäten implementiert werden, um die Kommunikation zwischen der Steuerungssoftware und dem mBot2 zu ermöglichen.
- Der gesamte Projektverlauf muss dokumentiert werden, und Änderungen sollen mit Hilfe von Version Control via Git festgehalten werden.

Technologische Anforderungen:
- Entwicklung von GUI-Software für Windows und mobile Endgeräte.
- Implementierung einer Serverkomponente zur Verbindung zwischen Steuerungssoftware und mBot2.

- Programmierung des mBot2, um Befehle von der Steuerungssoftware entgegenzunehmen und auszuführen.

Rahmenbedingungen:

- Das Projekt wird im Rahmen eines Semesterprojekts der 4. AHINF der HTL Saalfelden durchgeführt.
- Es gibt Zeitbeschränkungen, die mit dem Ende des Semesters einhergehen.
- Ressourcen wie Unterrichtsraum, Laborausrüstung (mBot2, Sensoren, Netzwerk, …) und finanzielle Unterstützung (hauptsächlich mBot2 und Sensoren) stehen durch die Schule zur Verfügung.
- Expertise der der beteiligten Schüler

Externe Faktoren:

- Verfügbarkeit von Hardware (mBot2, Computer, mobile Geräte) und Software (Entwicklungsumgebungen, Bibliotheken).
- Eventuelle externe Hilfe muss berücksichtigt werden. Dazu gehören Frage-Foren wie z.B. die bereits erwähnte Seite „stackoverflow“.
- Risiken wie unvorhergesehene technische Schwierigkeiten, begrenzte Ressourcen oder Zeitmangel sollten berücksichtigt werden.

Wettbewerbsfaktoren:

- Erfolgreiche Projekte aus früheren Semestern oder ähnliche Projekte an anderen Bildungseinrichtungen könnten als Benchmark dienen.
- Innovationspotenzial: Gibt es Möglichkeiten, das Projekt von anderen ähnlichen Projekten abzuheben? -> mBot2 als Mittelpunkt

Projektmanagement:

- Eine klare Aufgabenverteilung und Zeitplanung sind entscheidend, um sicherzustellen, dass das Projekt rechtzeitig abgeschlossen wird.
- Kommunikation und Zusammenarbeit zwischen den Teammitgliedern sind unerlässlich, insbesondere bei einem Projekt mit verschiedenen technischen Aspekten wie diesem.

Rechtliche Aspekte:

- Es ist wichtig sicherzustellen, dass alle verwendeten Technologien und Softwarelizenzen den rechtlichen Anforderungen entsprechen.
- Datenschutz- und Sicherheitsaspekte müssen ebenfalls berücksichtigt werden, insbesondere wenn das System drahtlos kommuniziert und persönliche Daten verarbeiten könnte (Datenspeicherung in Datenbank -> eventuelle Login-Erweiterung)
- Diese Analyse bietet einen umfassenden Überblick über das Projektumfeld und dient als Leitfaden für die Planung, Durchführung und Bewertung des Projekts zur Entwicklung des mBot2-Fernsteuerungssystems.

5. Softwarearchitektur

Ein grundlegendes Verständnis des Aufbaus einer Software oder Hardware ist von entscheidender Bedeutung, um ihre Funktionsweise zu erfassen und effektiv damit arbeiten zu können. Dieses Kapitel widmet sich genau diesem Zweck, indem es einen detaillierten Einblick in die Struktur und Zusammensetzung des SW-Produkts bietet. Beginnend mit einer Aufschlüsselung der einzelnen Komponenten, wird erläutert, aus welchen Elementen das System besteht und wie sie miteinander interagieren. Dabei wird insbesondere darauf eingegangen, wie die Software- und Hardwarekomponenten in einem möglicherweise verteilten System zusammenarbeiten. Darüber hinaus wird die Art und Weise der Kommunikation zwischen den Komponenten beleuchtet, um ein umfassendes Verständnis für die Gesamtfunktionalität des Systems zu vermitteln.

Das Folgende Kapitel besteht hauptsächlich aus visueller Erläuterung durch **Diagramme**.

Grundlegende Software-Komponenten:

- **MBot-Client (Micropython)**
- **Server (Spring Boot)**
- **Datenbank (MongoDB, MQL)**
- **Steuerungssoftware (C#)**

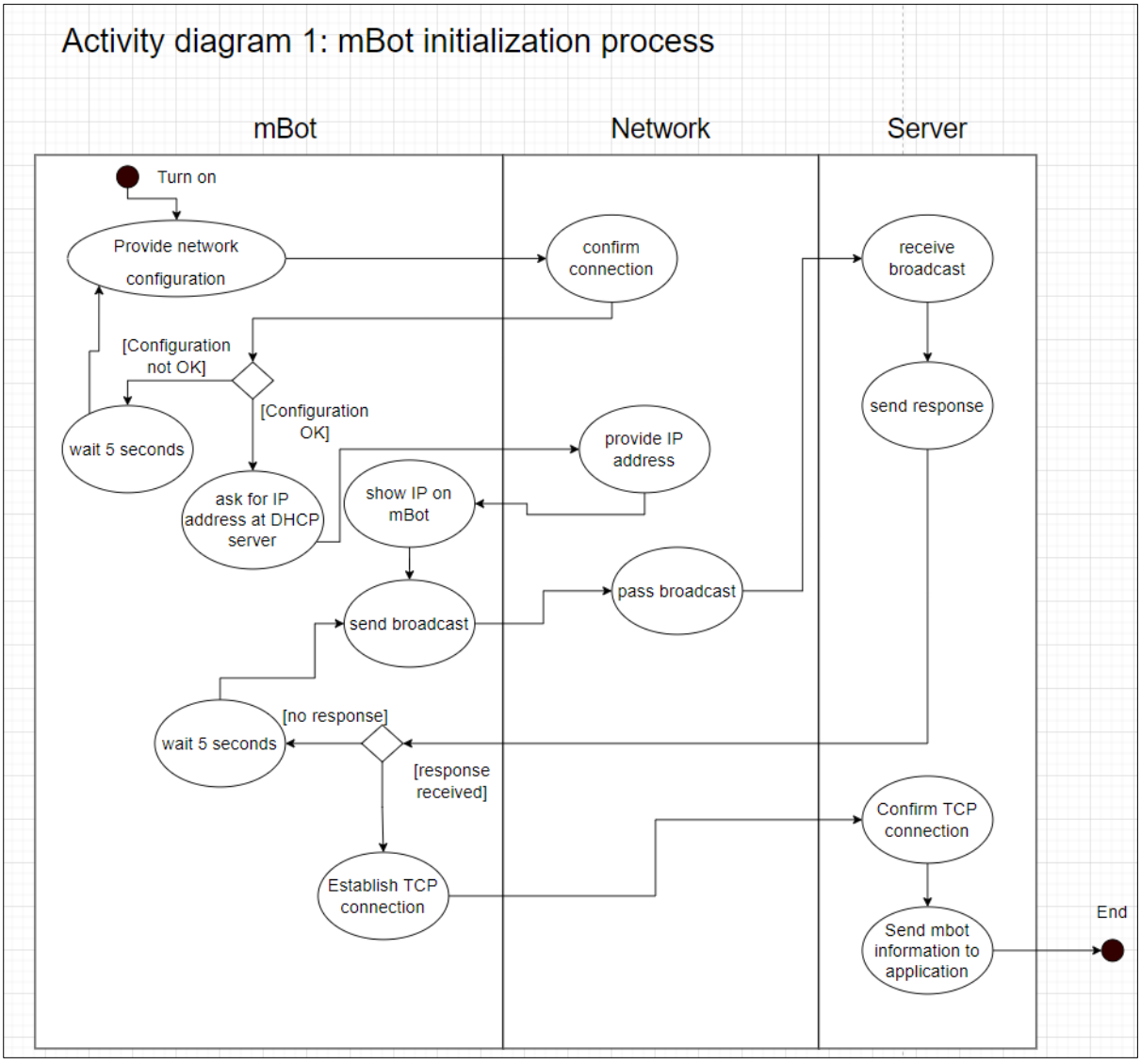
5.1 Aktivitätsdiagramme

Aktivitätsdiagramme sind ein wesentliches Werkzeug in der Softwaredokumentation, um Arbeitsabläufe und Prozesse innerhalb eines Systems grafisch darzustellen. Sie helfen dabei, die Sequenz von Aktivitäten und die logischen Abläufe in einer Anwendung zu visualisieren. Dieses Kapitel enthält die relevanten Aktivitätsdiagramme für unser Projekt, die detailliert die verschiedenen Prozesse und deren Abläufe illustrieren.

5.1.1 Aktivitätsdiagramm 1 – mBot Initialisierungsprozess

Der Initialisierungsprozess des mBot ist ein entscheidender Schritt, um den Roboter für den Betrieb vorzubereiten. Diese Aktivität beschreibt die notwendigen Schritte und Abläufe, um sicherzustellen, dass der mBot korrekt gestartet und einsatzbereit ist.

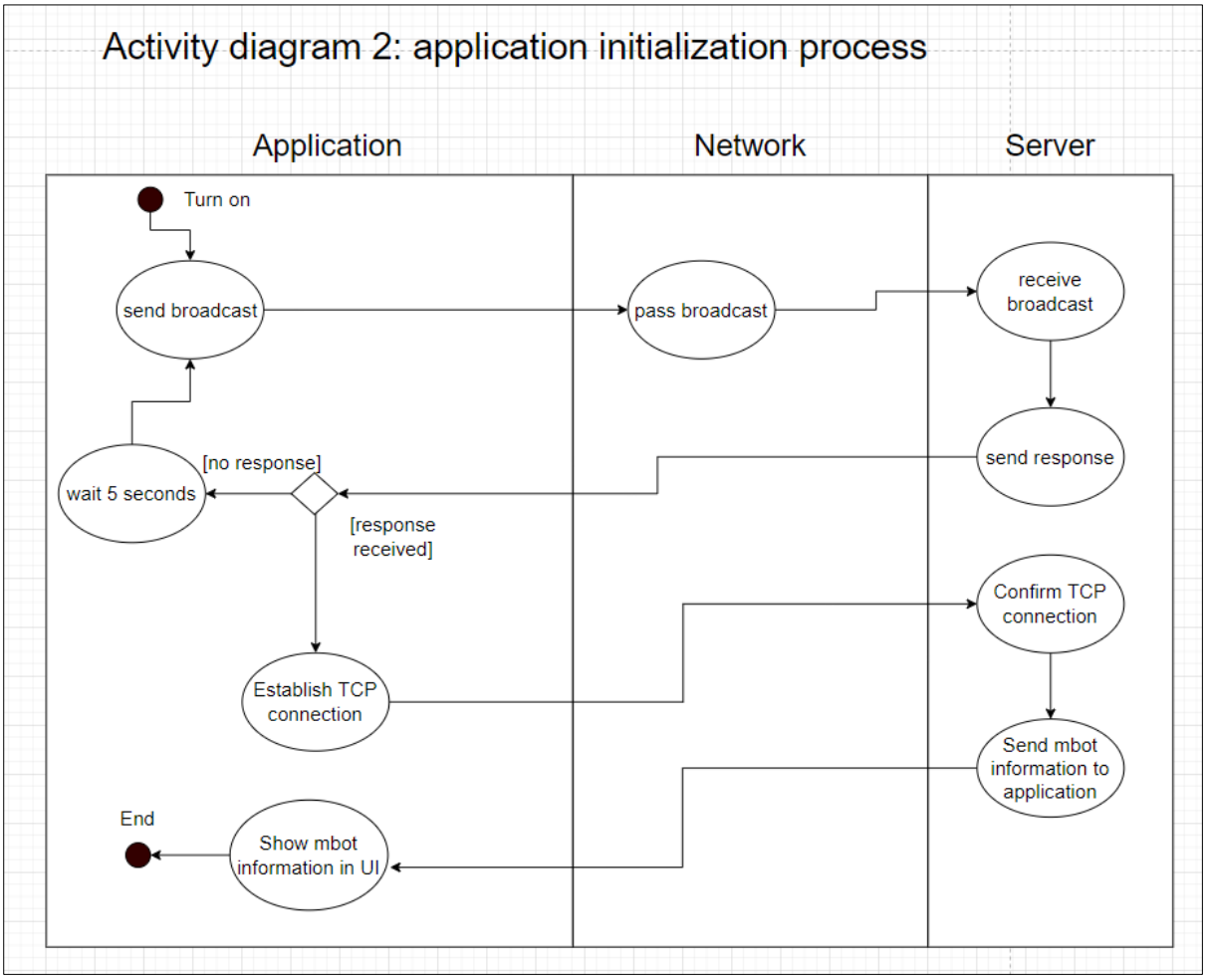
Aktivitätsdiagramm:



5.1.2 Aktivitätsdiagramm 2 – Software Initialisierungsprozess

Der Initialisierungsprozess der Steuerungssoftware ist ein entscheidender Schritt, um den das System für den Betrieb vorzubereiten. Diese Aktivität beschreibt die notwendigen Schritte und Abläufe, um sicherzustellen, dass die Steuerungssoftware korrekt gestartet und einsatzbereit ist.

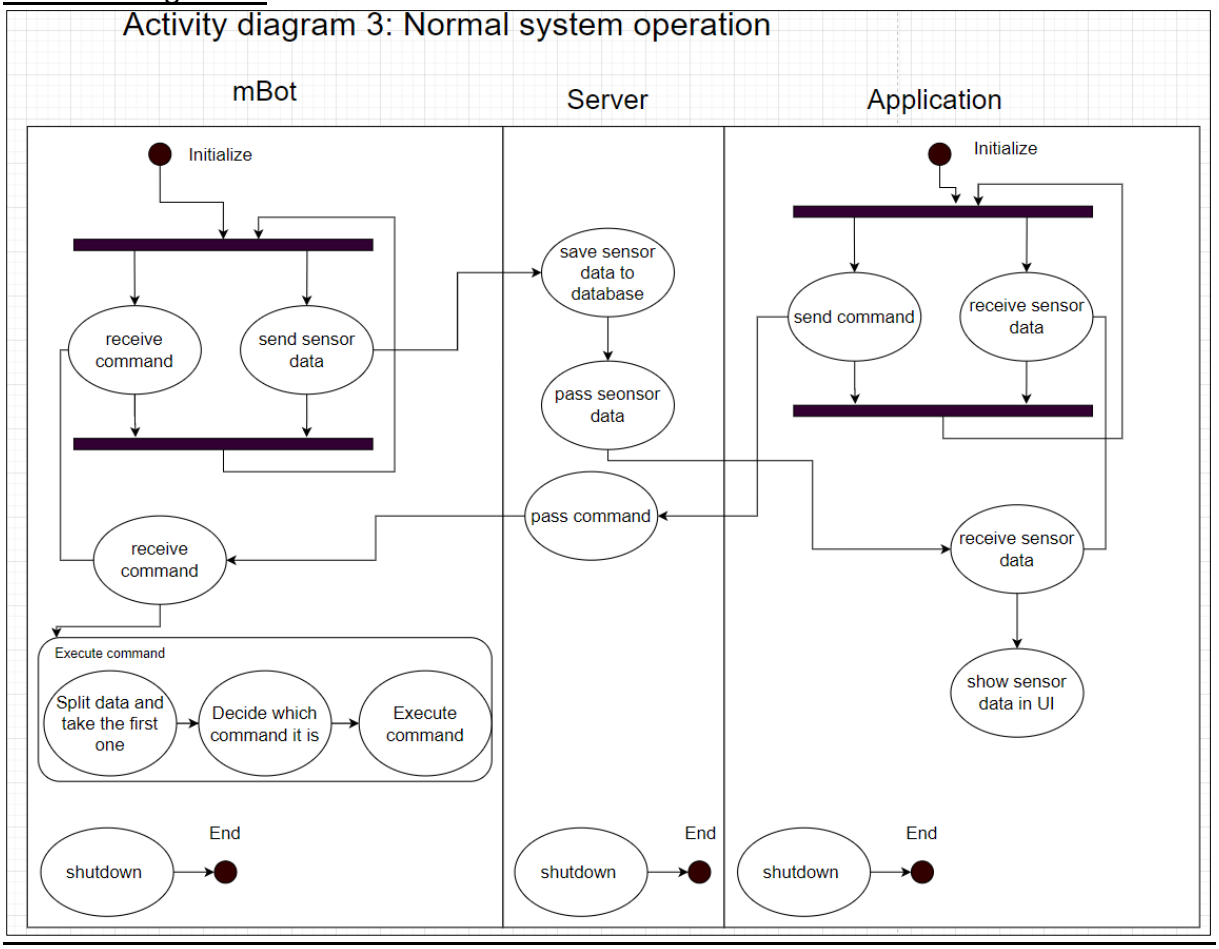
Aktivitätsdiagramm:



5.1.3 Aktivitätsdiagramm 3 – Normaler Ablauf des Systems

Das folgende Aktivitätsdiagramm zeigt alle Schritte, die während des Normalbetriebs des Systems getätigt werden. Dabei wird davon ausgegangen, dass mBot und Steuerungssoftware sowie der Server einsatzbereit sind.

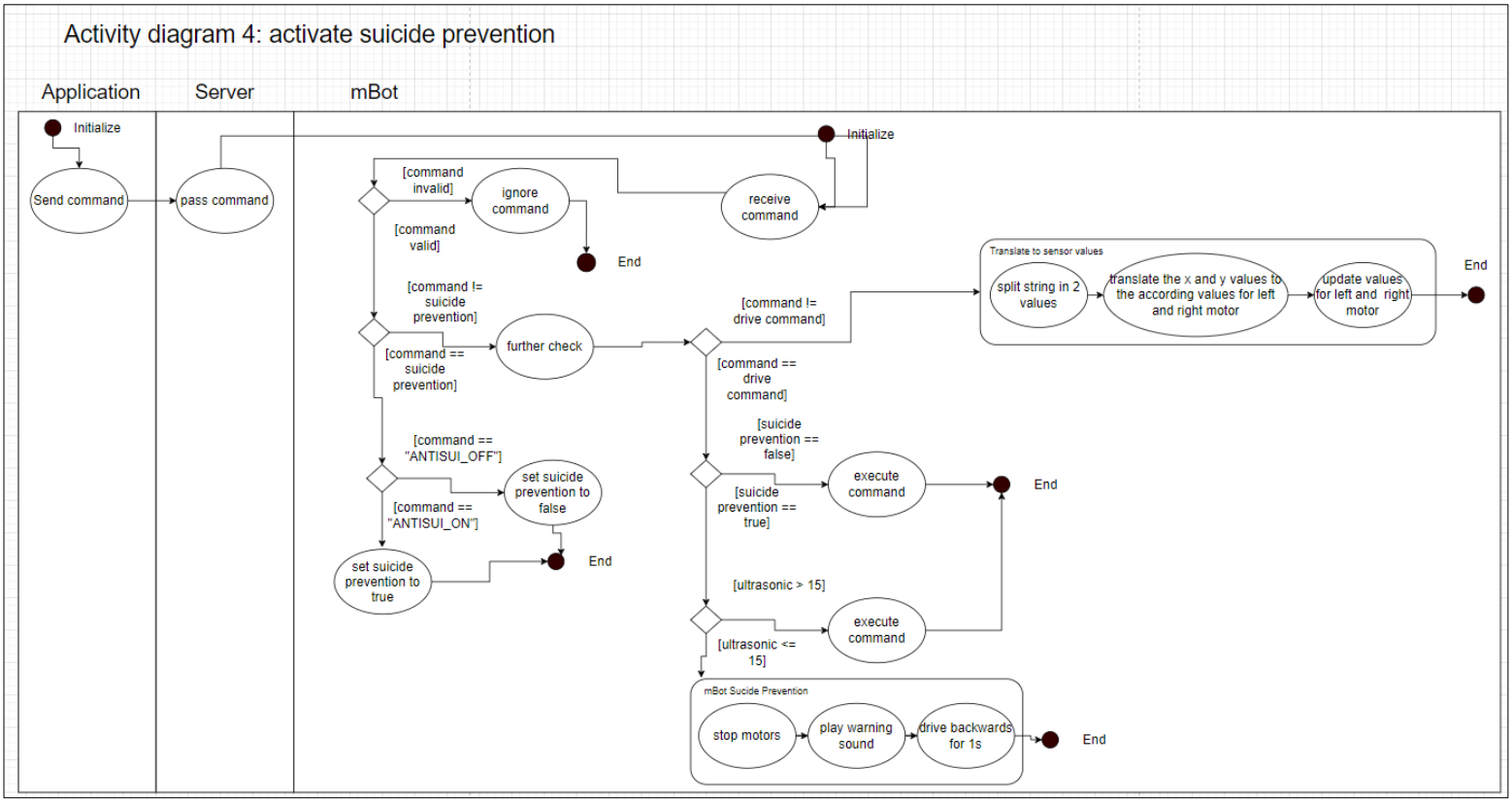
Aktivitätsdiagramm:



5.1.4 Aktivitätsdiagramm 4 – Aktivierung des Sicherheitsmodus

Das folgende Aktivitätsdiagramm zeigt alle notwendigen Schritte für die Aktivierung des Sicherheitsmodus und den Ablauf, sollte dieser eingeschaltet sein.

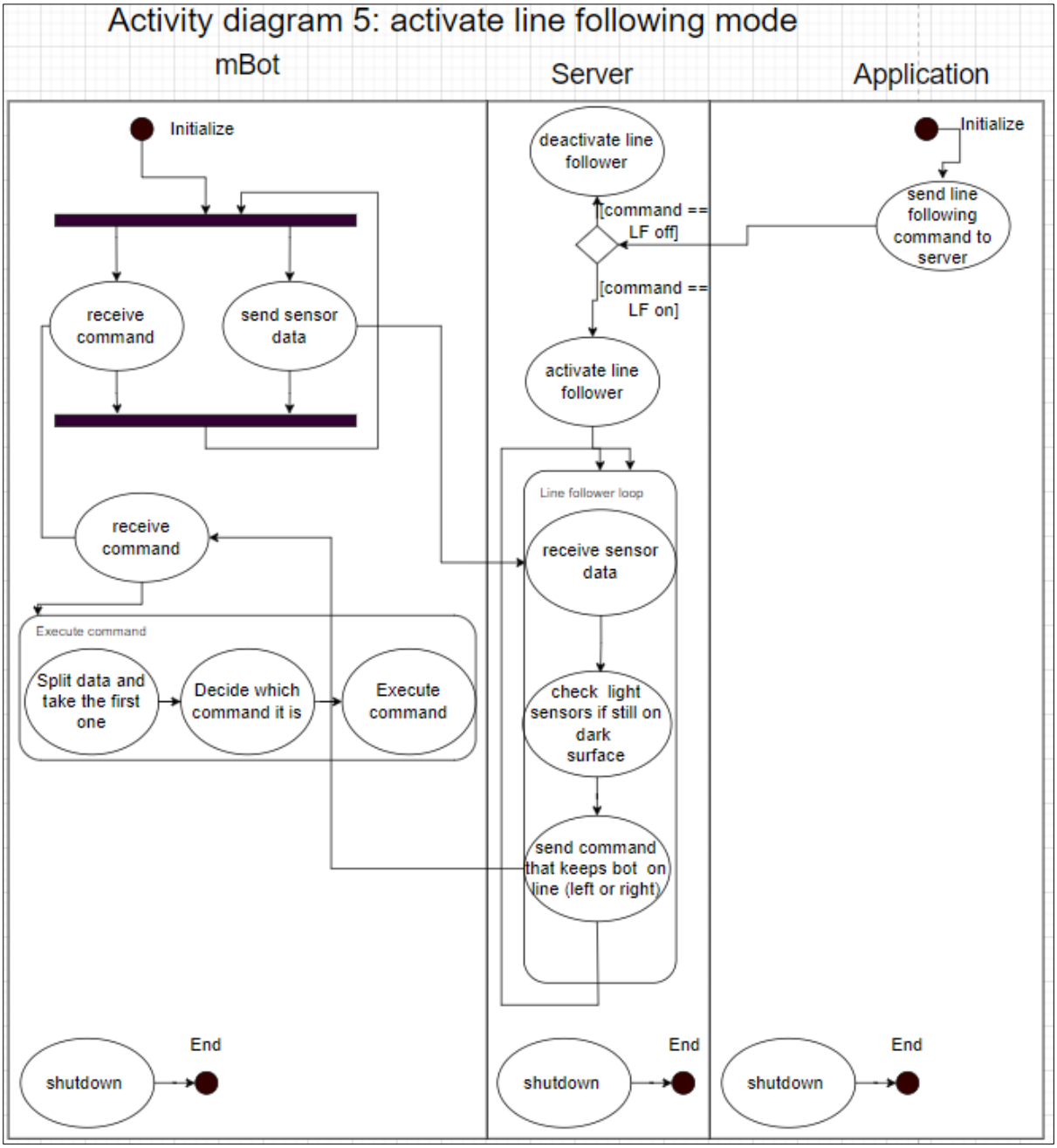
Aktivitätsdiagramm:



5.1.5 Aktivitätsdiagramm 5 – Aktivierung des Linienfolgemodus

Das folgende Aktivitätsdiagramm zeigt alle notwendigen Schritte, die getätigt werden müssen, um den Linienfolgemodus einzuschalten. Hierbei übernimmt der Server die Steuerung des mBot.

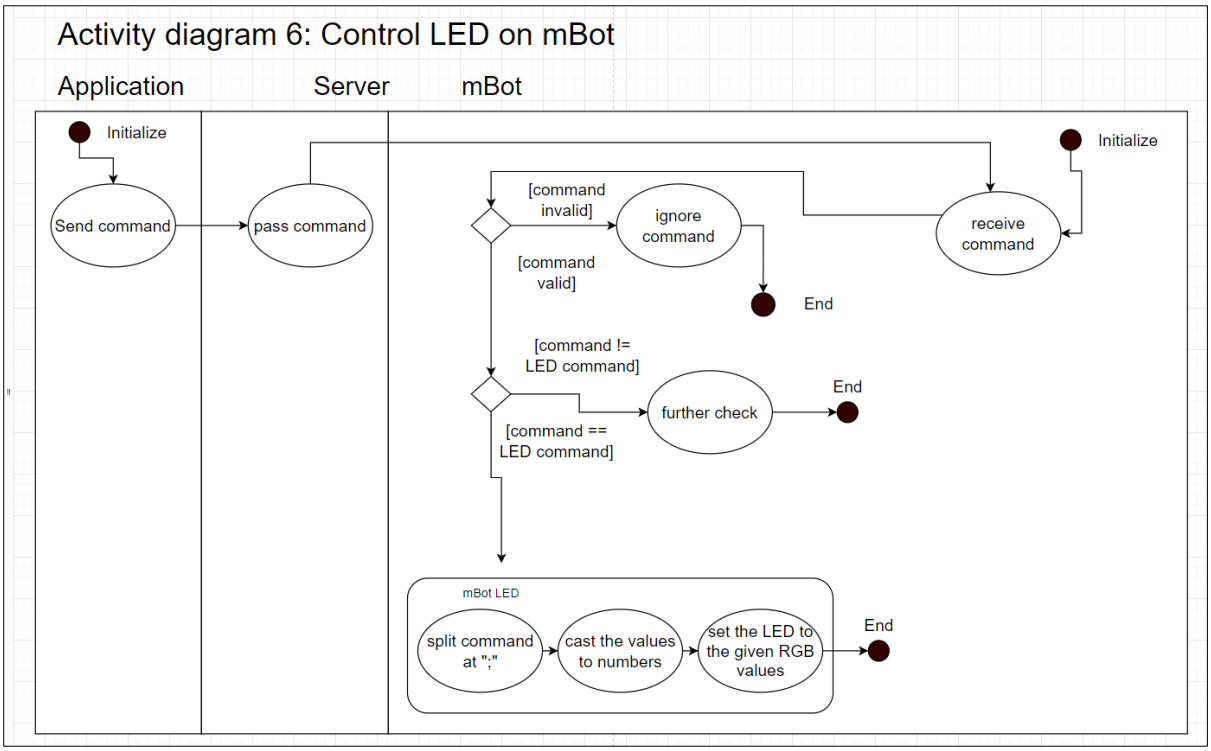
Aktivitätsdiagramm:



5.1.6 Aktivitätsdiagramm 6 – Steuerung der LEDs am mBot

Folgendes Aktivitätsdiagramm zeigt alle notwendigen Schritte, die getätigt werden müssen, um die LEDs am mBot zu steuern.

Aktivitätsdiagramm:



5.2 Sequenzdiagramme

Sequenzdiagramme sind genauso wie die Aktivitätsdiagramme ein wesentliches Werkzeug in der Softwaredokumentation zur Darstellung von Interaktionen zwischen Systemkomponenten im zeitlichen Verlauf. Dieses Kapitel enthält die relevanten Sequenzdiagramme für unser Projekt, die detailliert die Kommunikation und Abläufe zwischen den verschiedenen Systemelementen veranschaulichen.

Folgende Sequenzdiagramme wurden mit dem Tool von **sequencediagram.org** erstellt.

5.2.1 Sequenzdiagramm 1 – MBot Internet und Server Verbindung

Das folgende Sequenzdiagramm zeigt die Schritte, die durchgeführt werden, um den mBot2 mit dem WLAN und dem Server zu verbinden. Es veranschaulicht die zeitliche Abfolge und die Interaktionen zwischen den beteiligten Komponenten, einschließlich des mBot2, dem WLAN-Router und dem Server.

Die wichtigsten Schritte in dieser Sequenz umfassen:

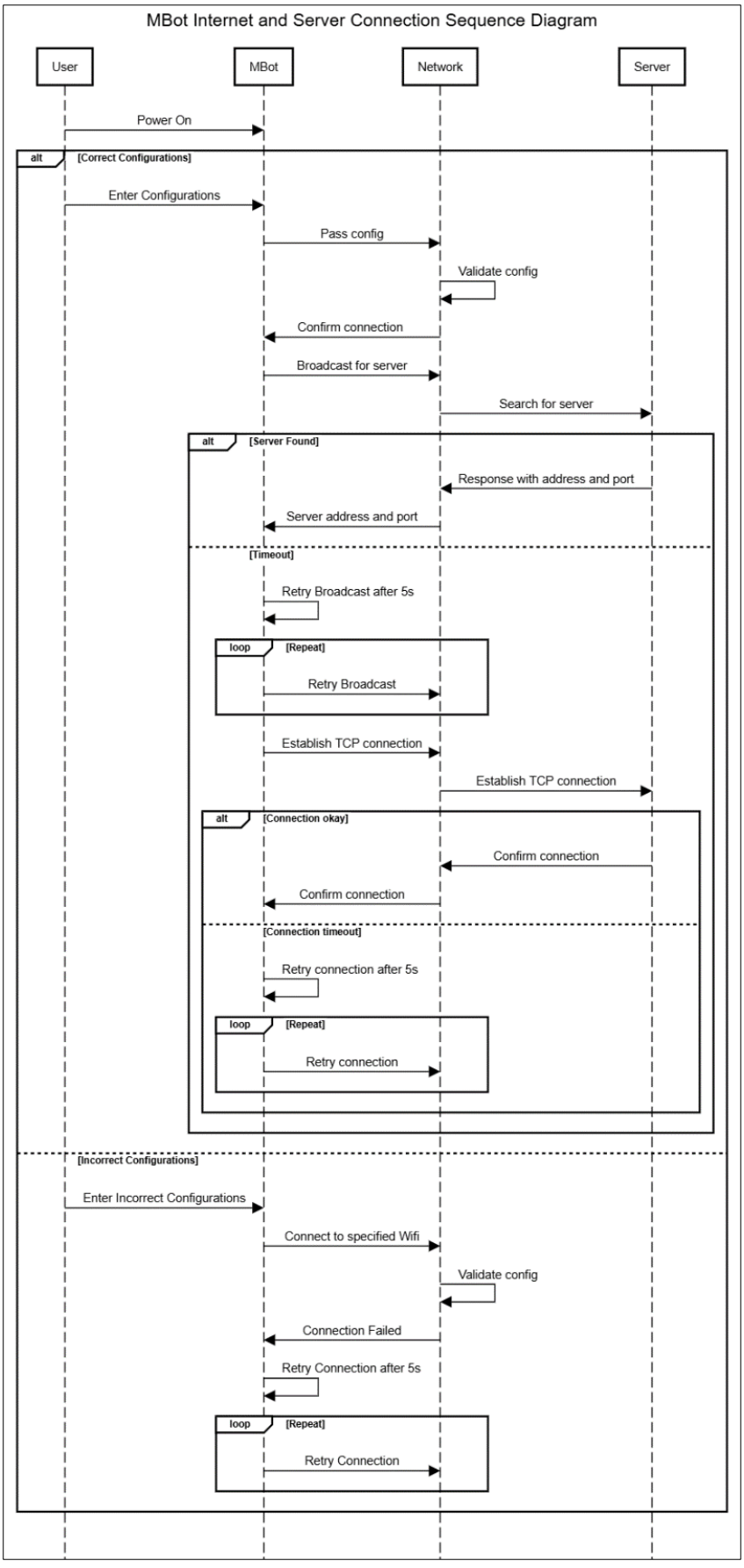
1. **WLAN-Suche und -Verbindung:**
   Der mBot2 sucht nach verfügbaren WLAN-Netzwerken und wählt das konfigurierte Netzwerk aus. Er sendet eine Verbindungsanforderung an den WLAN-Router und erhält eine Bestätigung der erfolgreichen Verbindung. Sollte die Verbindung zum WLAN fehlschlagen, wird kurz gewartet und ein erneuter Versuch gestartet.

2. **IP-Adresse beziehen:**
   Nach der Verbindung mit dem WLAN fordert der mBot2 eine IP-Adresse vom DHCP-Server des Routers an und erhält diese.

3. **Serveranfrage:**
   Der mBot2 sendet eine Anfrage zur Verbindung zum Server in Form eines Broadcasts und wartet auf eine Antwort. Sollte kein Server reagieren, wird kurz gewartet und ein erneuter Versuch gestartet.

4. **Verbindungsbestätigung:**
   Der mBot2 erhält die Bestätigung dass ein Server exisitiert und verbindet sich über TCP, um bereitzusein für den Datenaustausch und weitere Operationen.

Sequenzdiagramm:



5.2.2 Sequenzdiagramm 2 – Anwendung-Server-Verbindungs

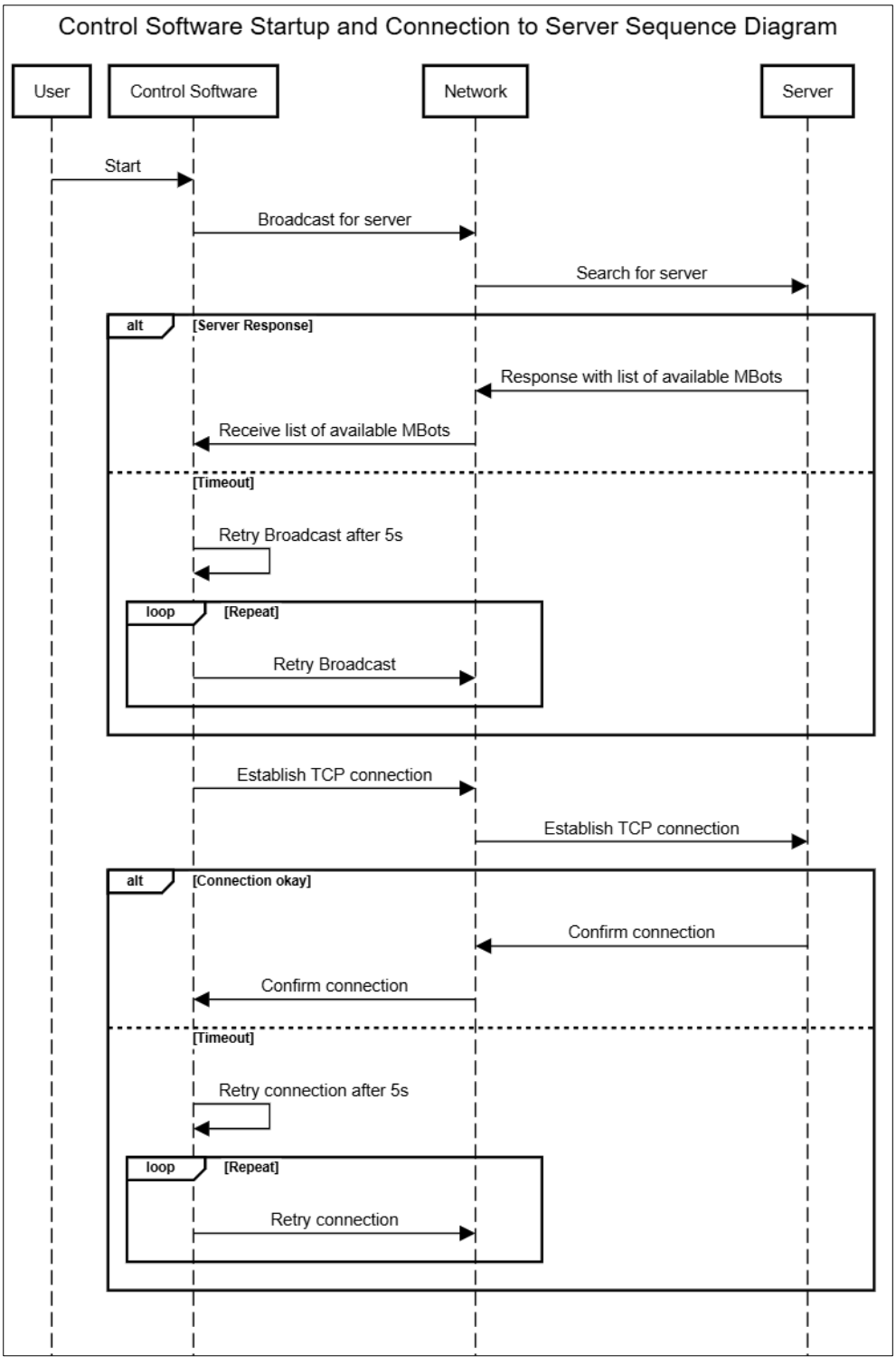
Das folgende Sequenzdiagramm zeigt die Schritte, die durchgeführt werden, um die Steuerungssoftware mit dem Server zu verbinden. Es veranschaulicht die zeitliche Abfolge und die Interaktionen zwischen den beteiligten Komponenten, einschließlich der Software und dem Server.

Die wichtigsten Schritte in dieser Sequenz umfassen:

1. **Serversuche**
   Genau wie der mBot schickt auch die Steuerungssoftware einen Broadcast zum Finden des Servers, welcher mit einer Liste der verfügbaren mBots antwortet, sollte er existieren. Wenn kein mBot auf den Broadcast reagiert, wird kurz gewartet und erneut angefragt.

2. **Verbindungsbestätigung**
   Nachdem der Server gefunden wurde, verbindet sich die Steuerungssoftware via TCP, um bereitzusein für den Datenaustausch und weitere Operationen. Sollte die Verbindung fehlschlagen, wird kurz gewartet und ein erneuter Versuch gestartet.

Sequenzdiagramm:



5.2.3 Sequenzdiagramm 3 – Anzeigen von Sensordaten

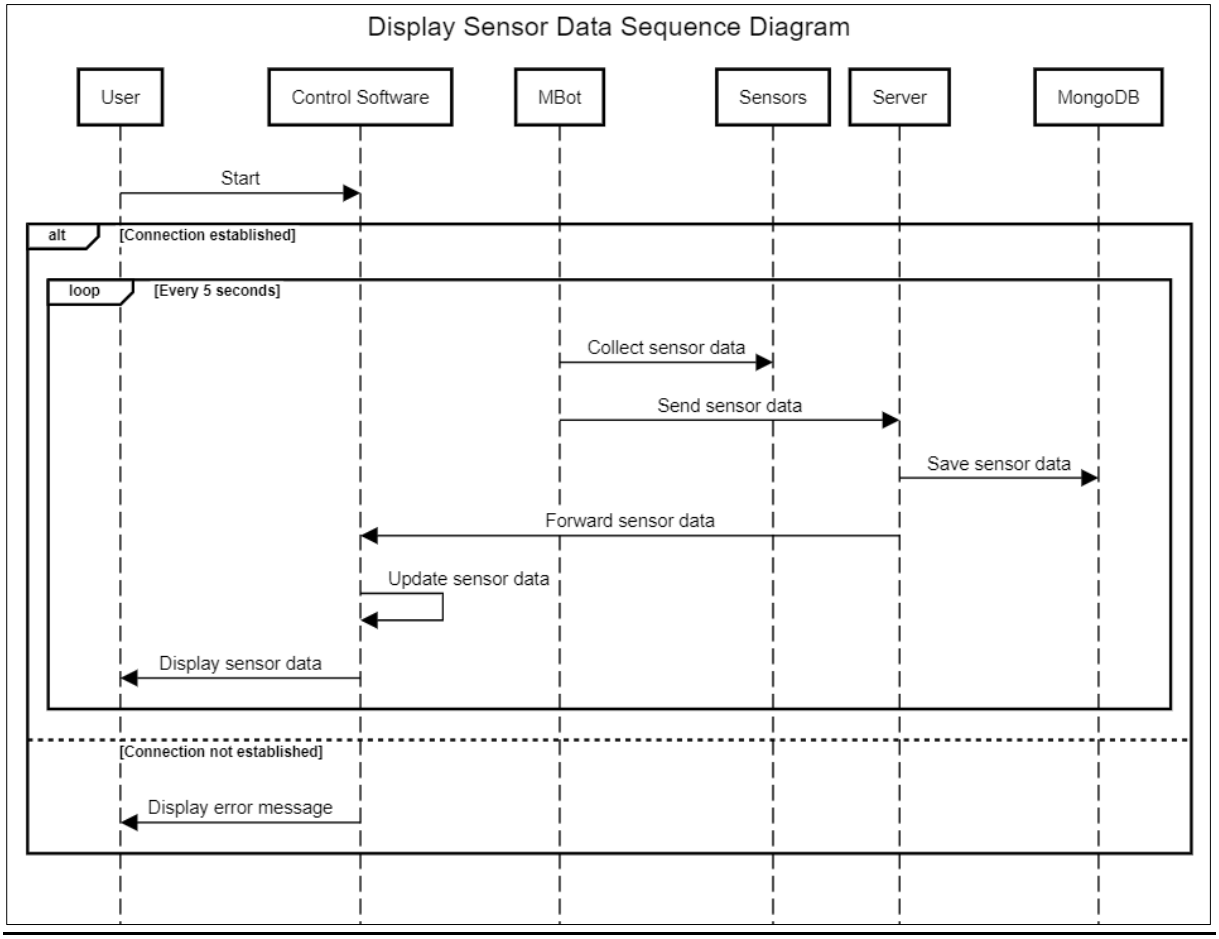
Das folgende Sequenzdiagramm zeigt die Schritte, die durchgeführt werden, um die Sensordaten des mBot in Echtzeit in der Steuerungssoftware anzuzeigen.

Die wichtigsten Schritte in dieser Sequenz umfassen:

1. **Senden von Daten**
   Der mBot holt intervallartig (alle fünf Sekunden) Daten von seinen Sensoren und sendet diese im JSON-Foramt an den Server, welcher diese in der Datenbank speichert und an die Software weiterleitet.

2. **Anzeigen von Daten**
   Die Software empfängt die übermittelten Daten und stellt diese Visuell in der Anwendung dem Benutzer zur Verfügung.

Sequenzdiagramm:



5.2.4 Sequenzdiagramm 4 – Steuerung des mBot

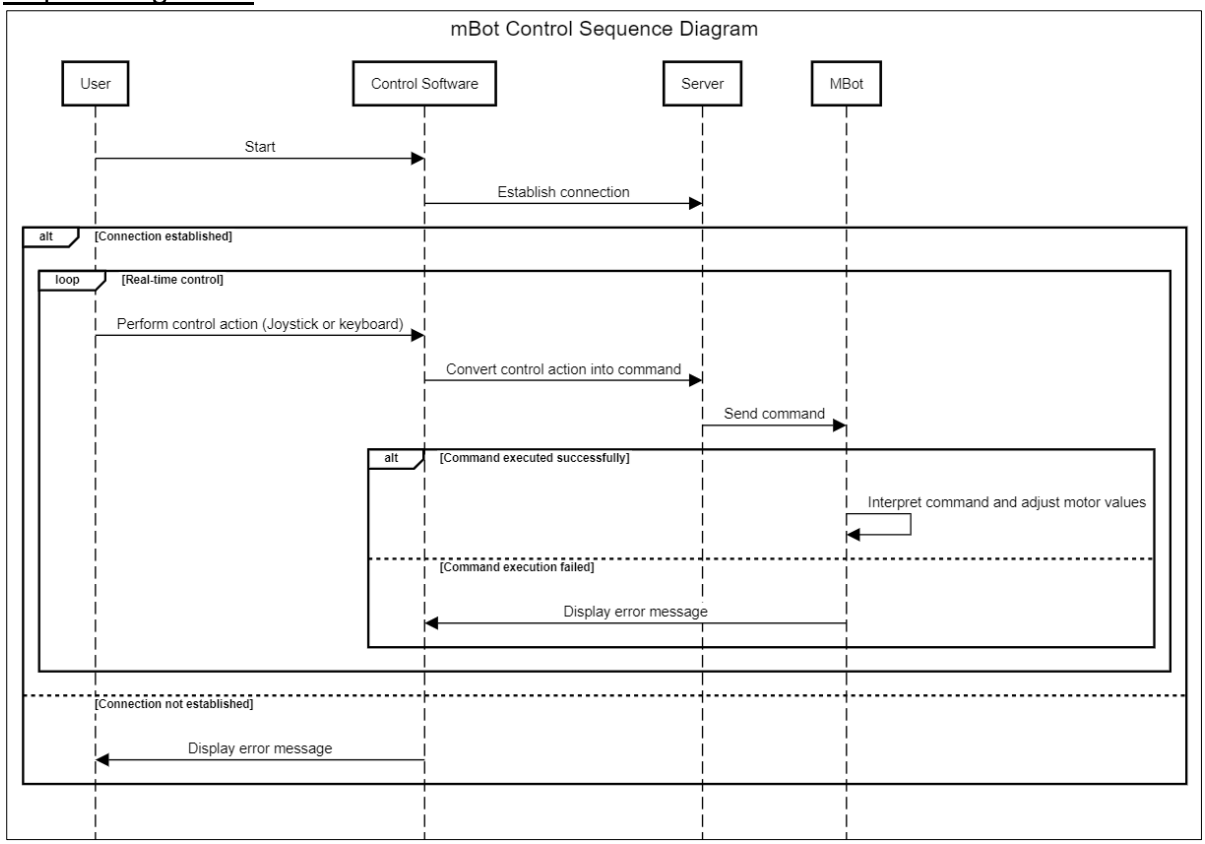
Das folgende Sequenzdiagramm zeigt die Schritte, die durchgeführt werden, um den mBot in Echtzeit fernzusteuern.

Die wichtigsten Schritte in dieser Sequenz umfassen:

1. **Senden eines Befehls**
   Die Steuerungssoftware implementiert zwei Wege den mBot zu steuern: JoyStick und tastaturbasierte Steuerung. Führt der Benutzer eine Steuerungsaktion aus wird diese in einen Steuerungsbefehl umgewandelt und an den Server gesendet, welcher diesen an den mBot weiterleitet.

2. **Ausführen des Befehls**
   Der mBot empfängt einen Befehl und wandelt diesen in Werte für den linken und rechten Motor um, besonders beim JoyStick müssen die Werte umgerechnet werden, stoppt den letzten Befehl und führt den neuen aus. Kann der Befehl nicht interpretiert werden, wird das als Fehlermeldung angezeigt und der Befehl ignoriert.

Sequenzdiagramm:



5.2.5 Sequenzdiagramm 5 – Aktivieren des Sicherheitsmodus

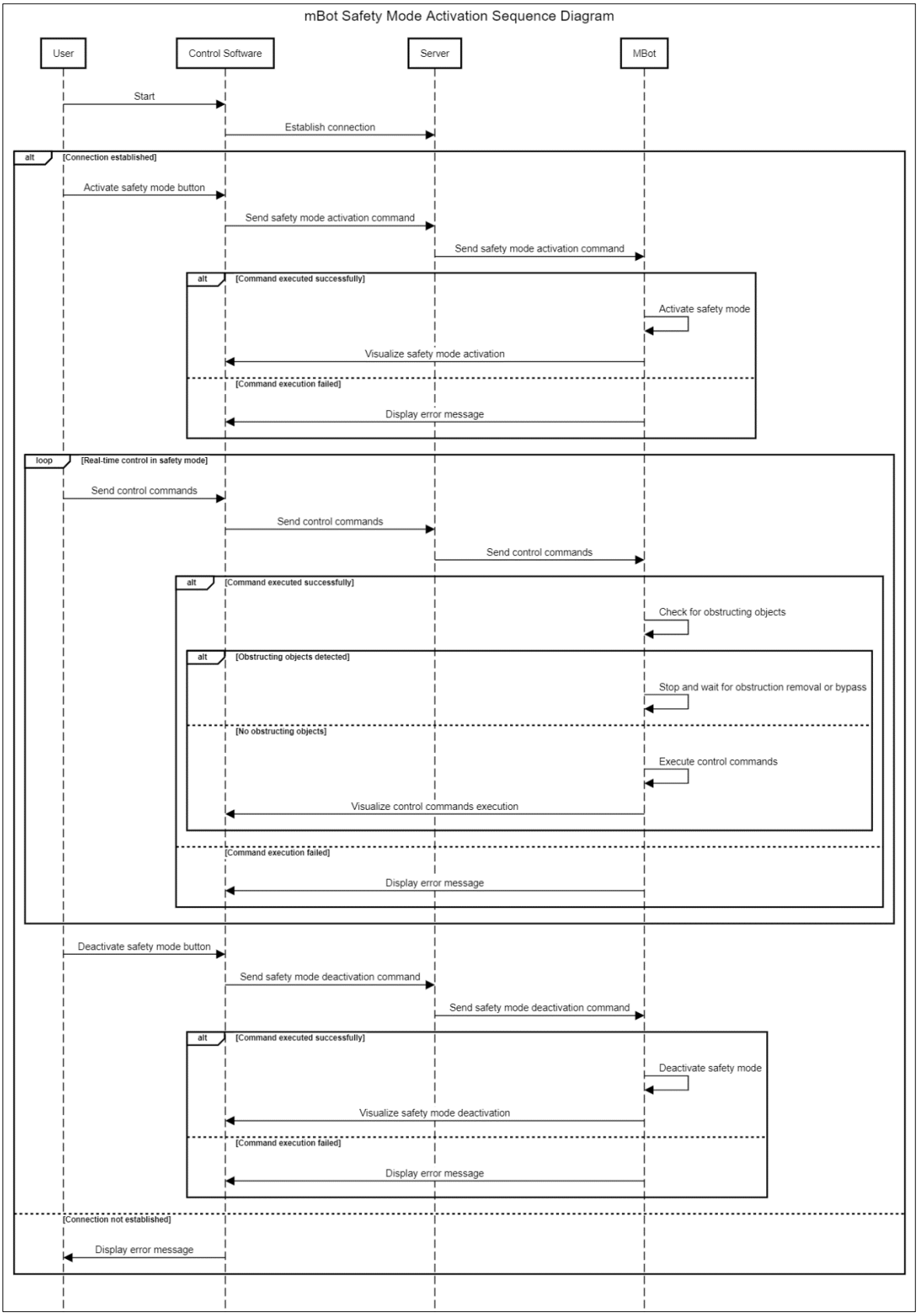
Das folgende Sequenzdiagramm zeigt die Schritte, die durchgeführt werden, um auf dem mBot den Sicherheitsmodus „Suicide Prevention“ zu aktivieren.

Die wichtigsten Schritte in dieser Sequenz umfassen:

1. **Senden des Befehls**
   Die Steuerungssoftware implementiert einen Knopf, welcher den Befehl zur Aktivierung des Sicherheitsmodus an den Server sendet, welcher diesen an den mBot weiterleitet.

2. **Ausführen des Befehls**
   Der mBot empfängt den Befehl und aktiviert den Sicherheitsmodus. Wird nun ein Steuerungsbefehl gesendet, prüft der mBot vor der Ausführung des Befehls, ob ihn Objekte frontal blockieren, bleibt gegenfalls stehen und wartet darauf, dass das Objekt entfernt oder umfahren wird. Sollte der Befehl nicht interpretiert werden können, wird das als Fehldermeldung angezeigt und der Befehl wird ignoriert.

Sequenzdiagramm:



5.2.6 Sequenzdiagramm 6 – Aktivierung des Linienfolgemodus

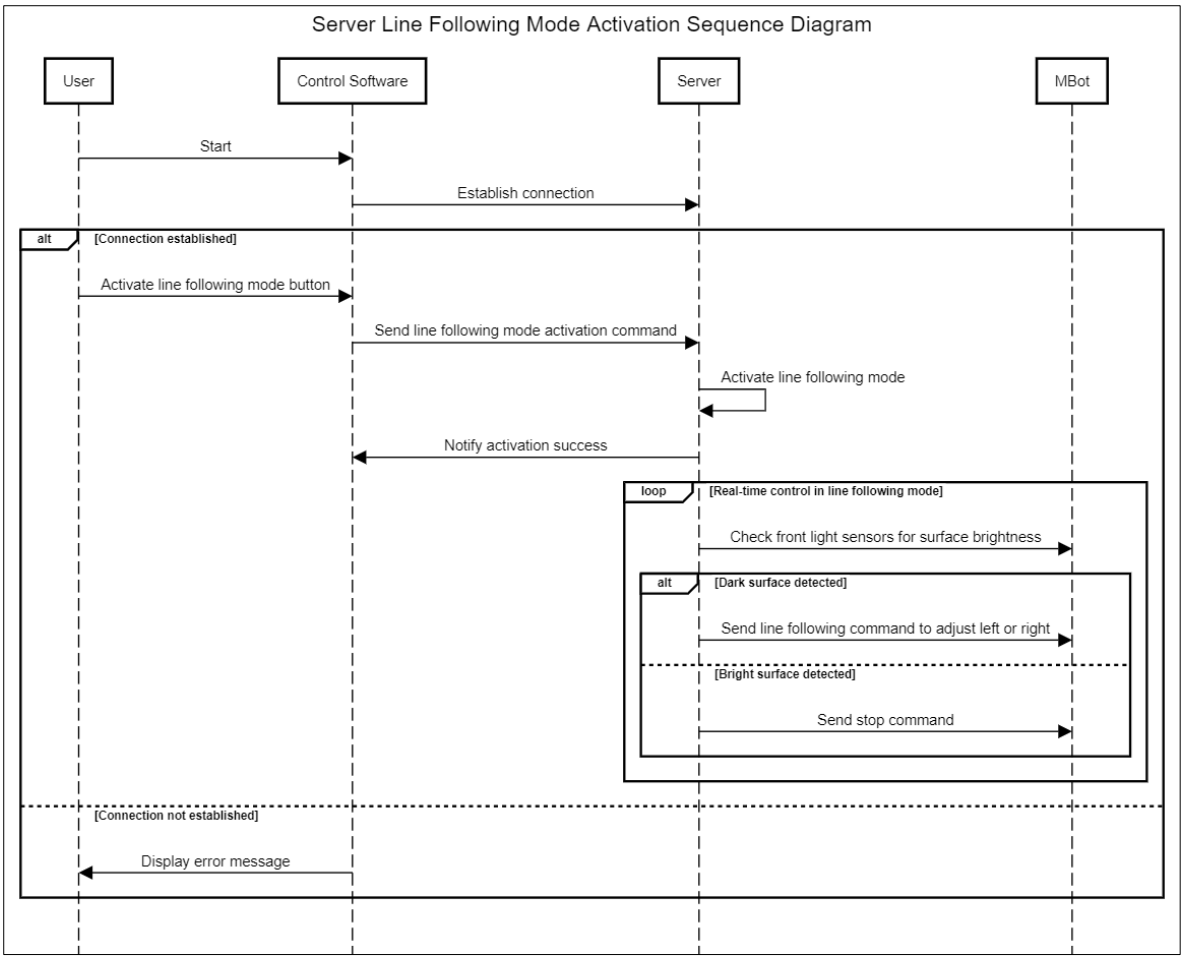
Das folgende Sequenzdiagramm zeigt die Schritte, die durchgeführt werden, um auf dem Server den Linien-Folgemodus zu aktivieren.

Die wichtigsten Schritte in dieser Sequenz umfassen:

1. **Senden des Befehls**
   Die Steuerungssoftware implementiert einen Knopf, welcher den Befehl zur Aktivierung des Sicherheitsmodus an den Server sendet.

2. **Ausführen des Befehls**
   Der Server empfängt den Befehl und aktiviert den Linienfolgemodus. In diesem Modus übernimmt der Server die Steuerung des mBot und prüft mittels Beobachtung der vier frontalen Lichtsensoren vor jedem Senden eines Befehls, ob der mBot noch auf dunklem Untergrund ist oder nicht und sendet Befehle entsprechend dem Folgen der Linie.

Sequenzdiagramm:



5.2.7 Sequenzdiagramm 7 – Steuerung der LEDs am mBot

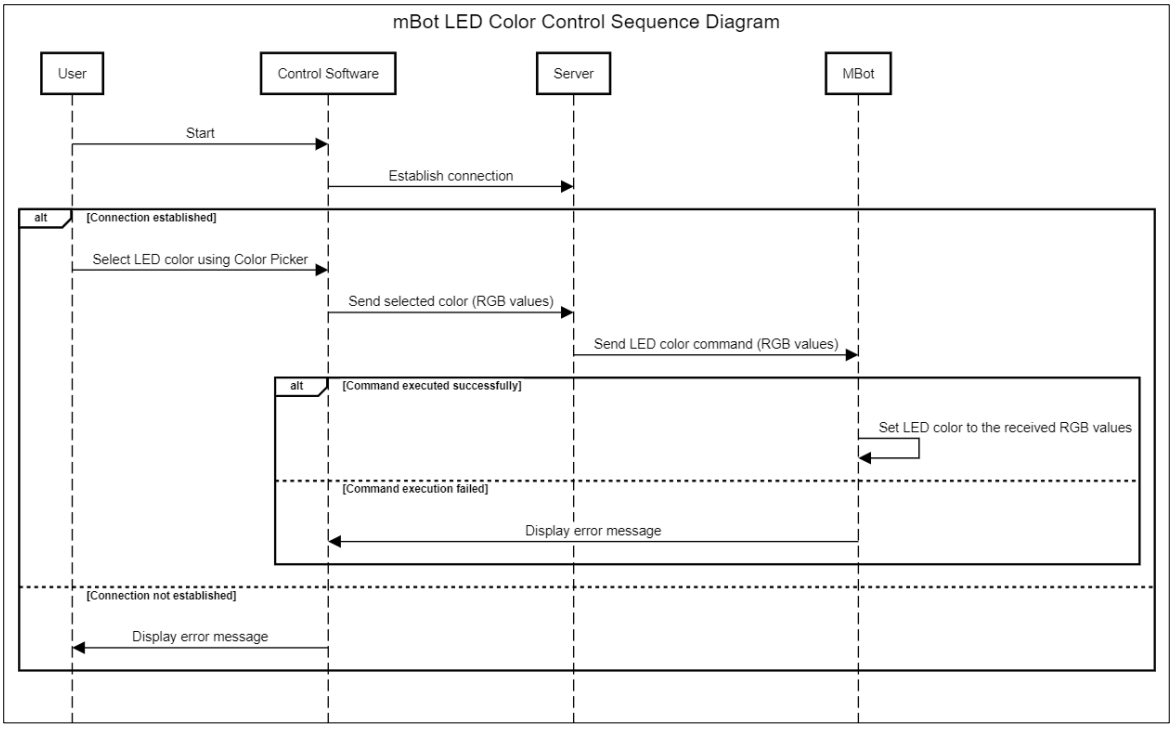
Das folgende Sequenzdiagramm zeigt die Schritte, die durchgeführt werden, um die Farben der LEDs am mBot zu steuern.

Die wichtigsten Schritte in dieser Sequenz umfassen:

3. **Senden des Befehls**
   Die Steuerungssoftware implementiert einen Color-Picker, welcher es dem Benutzer ermöglicht, eine Farbe auszuwählen, welche dann in Form von RGB-Werten an den Server gesendet wird, welcher den Befehl an den mBot weiterleitet.

4. **Ausführen des Befehls**
   Der mBot empfängt den Befehl und setzt seine LEDs auf die übermittelten werte. Dabei wird der Steuerungsprozess nicht unterbrochen. Sollte der Befehl nicht interpretiert werden können, wird das als Fehlermeldungangezeigt und der Befehl ingoriert.

Sequenzdiagramm:



5.3 Komponentendiagramm

Ein Komponentendiagramm dient dazu, die strukturelle Organisation eines Systems zu visualisieren und die Beziehungen zwischen seinen Komponenten aufzuzeigen. Es bietet einen ganzheitlichen Überblick über die einzelnen Bausteine des Systems und ihre Interaktionen.

- **Titel:**
  Komponentendiagramm Steuerungssoftware mBot2

- **Beschreibung:**
  Das Komponentendiagramm zeigt die verschiedenen Komponenten des Systems sowie deren Beziehungen zueinander. Jede Komponente repräsentiert eine logische Einheit innerhalb des Systems, die bestimmte Funktionalitäten bereitstellt oder bestimmte Aufgaben ausführt. Die Beziehungen zwischen den Komponenten stellen Abhängigkeiten, Schnittstellen oder Kommunikationskanäle dar.

- **Komponenten:**
  - Client-Anwendung (Steuerungssoftware):
    - Beschreibung: Die mit Avalonia (C#) entwickelte Software dient dem Benutzer als Schnittstelle zum mBot. Dazu kommuziert diese zuerst über den Broadcast ins Netzwerk und anschließend über den TCP-Port 5000 mit dem Server. Über diese Verbindung kann sie den mBot Steuern und bekommt stetig aktuelle Sensor-Daten. Über den HTTP-Port 8080 werden anfangs im Netzwerk gefundene mBots abgefragt und dem Benutzer zur Verfügung gestellt.

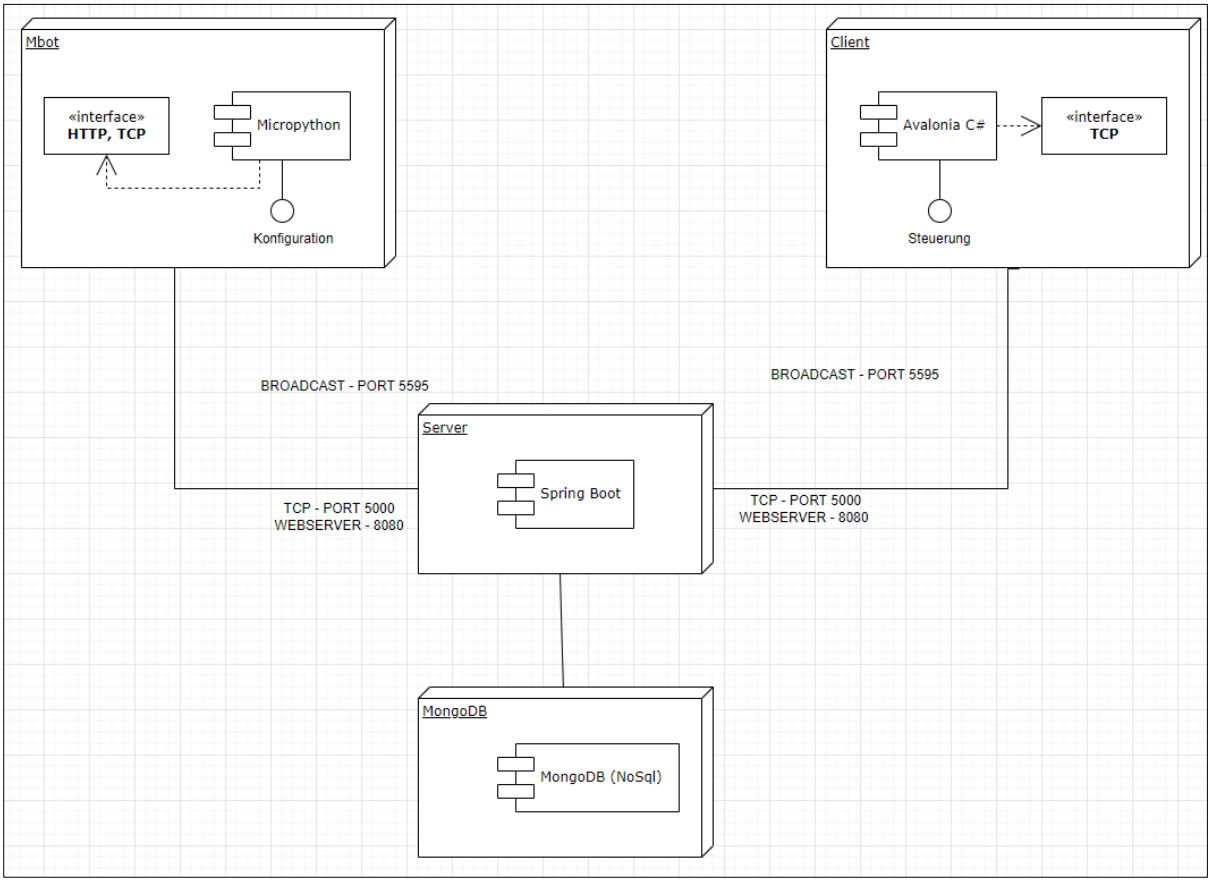
  - Server
    - Beschreibung: Der mit Spring Boot (Java) entwickelte Server als Schnittstelle zwischen Software und mBot. Dazu erwartet er vont beiden Seiten einen Broadcast übers Netzwerk und stellt anschließend über den TCP-Port 5000 eine Verbindung zu beiden her. Über diese Verbindungen kann er die Steuerbefehler von der Software entegennehmen und an den mBot weiterleiten. Außerdem ist er der Verwalter einer MongoDB-Datenbank und speichert Sensor-Daten im Hintergrund ab.

  - mBot-Anwendung
    - Beschreibung: Die mit Micropython entwickelte Software auf dem Microcontroller des mBot dient der Ausführung der Benutzer-Befehler und dem kontinuierlichen Übertragen der Sensor-Daten. Dazu kommuziert diese zuerst über den Broadcast ins Netzwerk und anschließend über den TCP-Port 5000 mit dem Server. Über diese Verbindung kann sie Befehle entgegennehmen und ausführen. Über den HTTP-Port 8080 werden stetig aktuelle Sensor-Daten an den Web-Server gesendet.

- MongoDB-Datenbank
  - Beschreibung: Die NoSQL-Datenbank „MongoDB“ wird vom Server verwaltet und speichert Sensordaten als Dokumente.

- **Systemgrenzen:**
  internen Komponenten und deren Beziehungen innerhalb des definierten Systems, SW-Produkt

Komponentendiagramm:



5.4 Verteilungsdiagramm

Ein Verteilungsdiagramm zeigt an, wie die einzelnen Teile der Software auf die Hardwarekomponenten verteilt sind und wie die Hardwarekomponenten miteinander verbunden sind. Sprich: Auf welchem Rechner läuft welche Software und wie sind diese übers Netzwerk miteinander verbunden.

- **Titel:**
  Verteilungsdiagramm Steuerungssoftware mBot2

- **Beschreibung:**
  Das Verteilungsdiagramm zeigt die Verteilung der Systemkomponenten auf die zugrunde liegenden Hardware-Ressourcen sowie deren Interaktionen. Es illustriert,

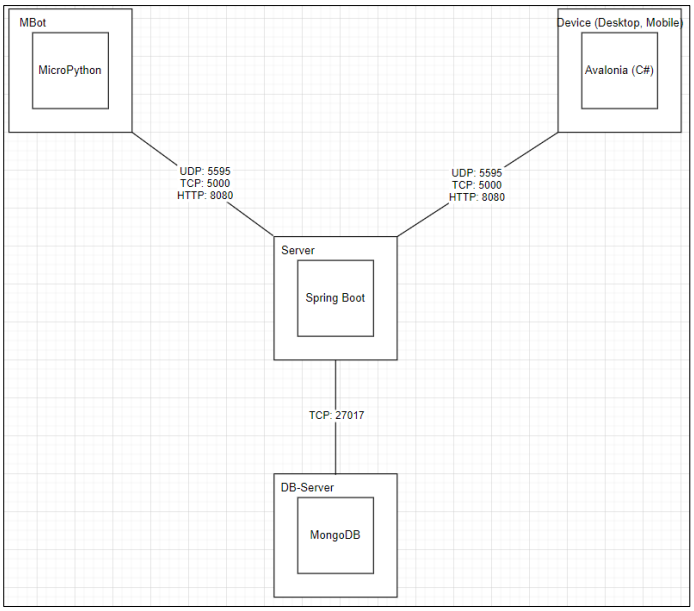
wie die verschiedenen Teile des Systems miteinander verbunden sind und wie die Kommunikation zwischen ihnen stattfindet.

- **Komponenten:**
  - Client-Anwendung (Steuerungssoftware):
    - Beschreibung: Die Client-Anwendung läuft auf einem Computer oder einem mobilen Endgerät und dient als Benutzerschnittstelle zur Steuerung des MBots.
  - Server
    - Beschreibung: Der Server läuft auf einem einem Computer und dient als Schnittstelle zwischen Andwendung und mBot.
  - mBot-Anwendung
    - Beschreibung: Die mBot-Anwendung läuft auf dem Microcontroller des mBot2 und dient sowohl der Verarbeitung der Benutzer-Befehle als auch dem Senden der Sensor-Daten.

- **Interaktionen:**
  - Die Client-Anwendung kommuniziert über das Netzwerkprotokoll mit dem Server, um Befehle an den MBot zu senden und Daten von ihm zu empfangen.
  - Der Server leitet die Befehle der Client-Anwendung an den MBot weiter und sendet die Sensordaten des MBots zurück an die Client-Anwendung.
  - Der MBot empfängt die Befehle des Servers und führt entsprechende Aktionen aus, wie z.B. Bewegungen ausführen oder Sensordaten erfassen.

- **Systemgrenzen:**
  beschreibt die Verteilung der Systemkomponenten innerhalb des definierten Softwareprodukts und zeigt die Interaktionen zwischen ihnen auf.

<u>Verteilungsdiagramm:</u>



5.5 Softwarekomponenten / Programme

5.5.1 SW Programme

Dieses Projekt wurde in unterschiedlichen Programmen bearbeitet und bediente sich selbst auch welcher, z.B. für Dokumentation oder Absicherung des Fortschrittes.

Folgende **Programme** wurden verwendet:

Programm	Version	Hersteller	Beschreibung
Visual Studio	2022 (17.9.1)	Microsoft	Entwicklungsumgebung für C#
IntelliJ IDEA	2023 (2.6)	JetBrains	Entwicklungsumgebung für Java
mBlock	5.4.3	makeBlock	Entwicklungsumgebung für
Github Desktop + git	3.3.13 (x64)	Github, Inc.	Versionskontroll-Software zur Code-Sicherung
Microsoft Word	/	Microsoft	Textbearbeitungsprogramm

5.5.2 SW Komponenten

SW-Komponent	Version	Hersteller	Bezugsquelle	SW-Lizenz	Beschreibung
Avalonia	11.0.10	AvaloniaUI OÜ	https://www.nuget.org/packages/Avalonia	https://licenses.nuget.org/MIT	Framework für Crossplattform-Entwicklung in C# (WPF)
Micropython	1.23.0	Damien P. George	Vorinstalliert auf mBot-Controller	/	
Spring boot	3.2.2	VMware	https://spring.io/projects/spring-boot	/	Webserver mit Anbindung an Datenbank & weiteren Features.

6. Projektdurchführung

6.1 Sprint 1

6.1.1 Sprintplanung

Dauer: 07.02.2024-06.03.2024

Name	Nummer	Kurzbeschreibung	Story Points
Create Software	3	Create Software to Monitor and Controll MBots	1
Connect Mbot to network	8	Make MBot connect to	3
Setup Spring Server	15	Setup the Server which connects MBots and client	21
Get all MBots from Server	6	Client should receive all MBot Data	5
MBot receive input	9	Mbot should receive input from server to control its movement	3
Send user input to Mbot	13	Send the user input to the server which forwards it to the MBot	5
Make DetailedView	7	Make a detailed view of the mbot in the client	5

Anzahl Story points: 43

Ausgewählte Punkte aus der Impediment Liste: noch keine Impediment-Liste vorhanden

6.1.2 Sprint Demo

Kurze Beschreibung welche User Stories umgesetzt worden sind und welche Funktionen erfolgreich präsentiert worden sind.
Auflistung welche Punkte nicht umgesetzt werden konnten und warum.

Name	Nummer	Nicht, weil…
Create Software	3	
Connect Mbot to network	8	
Get all MBots from Server	6	
Setup Spring Server	15	Zeitlich überschätzt
MBot receive input	9	Zeitlich überschätzt
Send user input to Mbot	13	Zeitlich überschätzt
Make DetailedView	7	Zeitlich überschätzt

6.1.3 Sprint Retrospektive

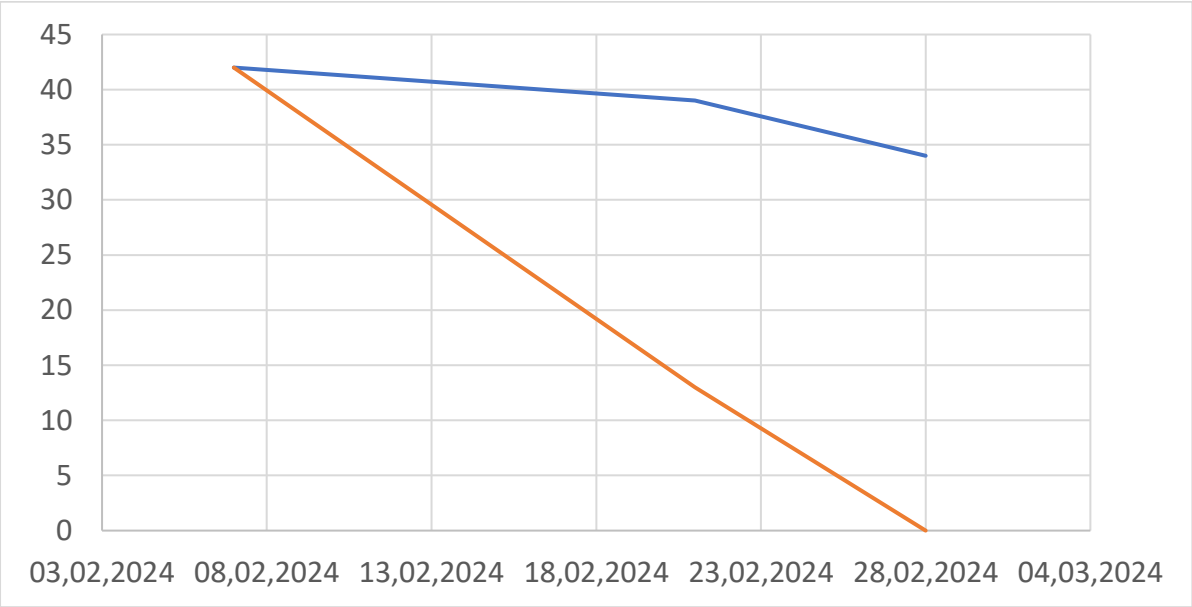
Gut	Schlecht
Dokumentation	Keine Branches verwendet
Projektstart	Kommunikationsschwierigkeiten

Impediment Liste
Kommunikationsschwierigkeiten
Designfehler

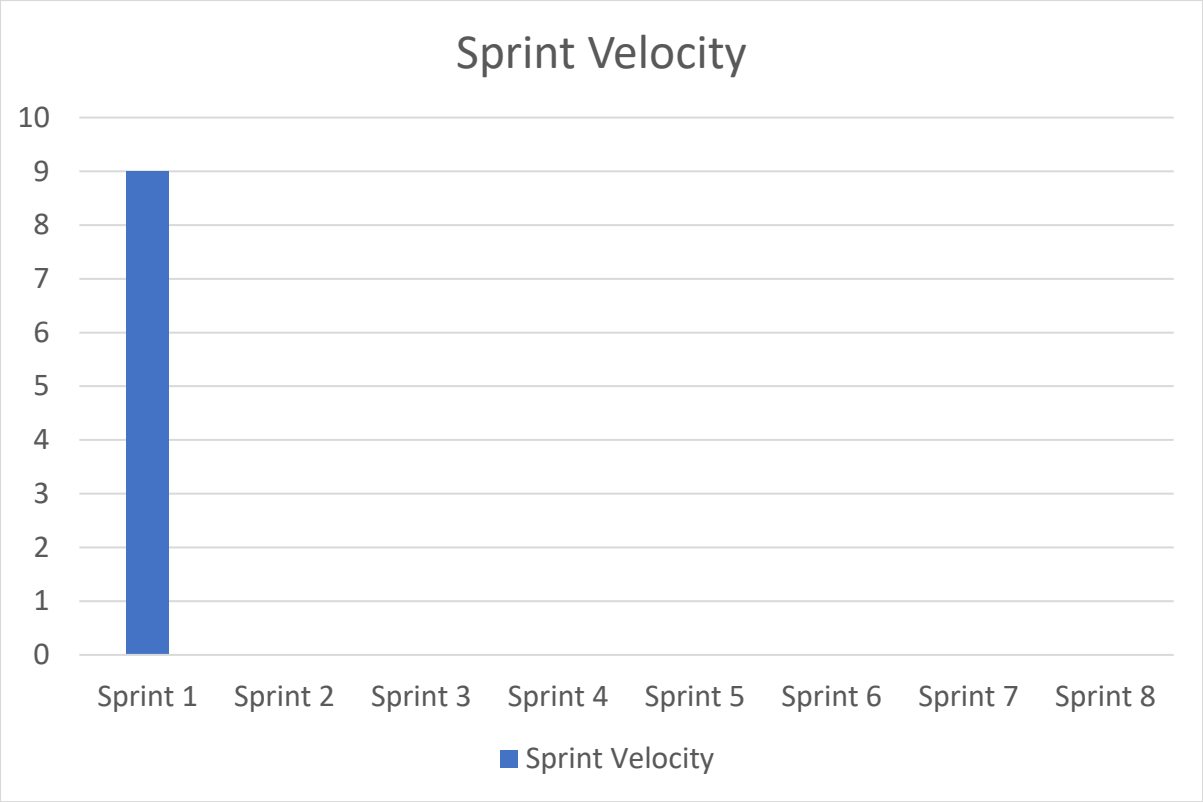
6.1.4 Sprint Zusammenfassung

Wurden in diesem Sprint neue User Stories in das Product Backlog eingefügt und wenn ja, welche?
Wurden in diesem Sprint User Stories aus dem Product Backlog entfernt und wenn ja, welche und warum?

Burndownchart



Sprint Velocity



Durchschnittliche Velocity: 9

6.2 Sprint 2

6.2.1 Sprintplanung

Dauer: 06.03.2024-20.03.2024

Name	Nummer	Kurzbeschreibung	Story Points
Server Broadcast	19	Add code which listens to broadcasts	5
MBot send sensor data to server	17	Sends the data of the MBot to the server	5
Make DetailedView	7	Make a detailed view of the mbot in the client	5
Setup Spring Server	15	Setup the Server which connects MBots and client	21
MBot receive input	9	Mbot should receive input from server to control its movement	3
Send client commands to Mbot	21	Forwards the clients commands to the mbot	5
Send data to client	23	Send the MBots data continuously to the client	8
Send user input to the server	13	Send the commands to the server	5

Anzahl Story points: 57

Ausgewählte Punkte aus der Impediment Liste: ---

6.2.2 Sprint Demo

Kurze Beschreibung welche User Stories umgesetzt worden sind und welche Funktionen erfolgreich präsentiert worden sind.
Auflistung welche Punkte nicht umgesetzt werden konnten und warum.

Name	Nummer	Nicht, weil…
Server Broadcast	19	
MBot send sensor data to server	17	
Make DetailedView	7	
Setup Spring Server	15	
MBot receive input	9	
Send client commands to Mbot	21	Zeitlich überschätzt
Send data to client	23	Zeitlich überschätzt
Send user input to the server	13	Zeitlich überschätzt

6.2.3 Sprint Retrospektive

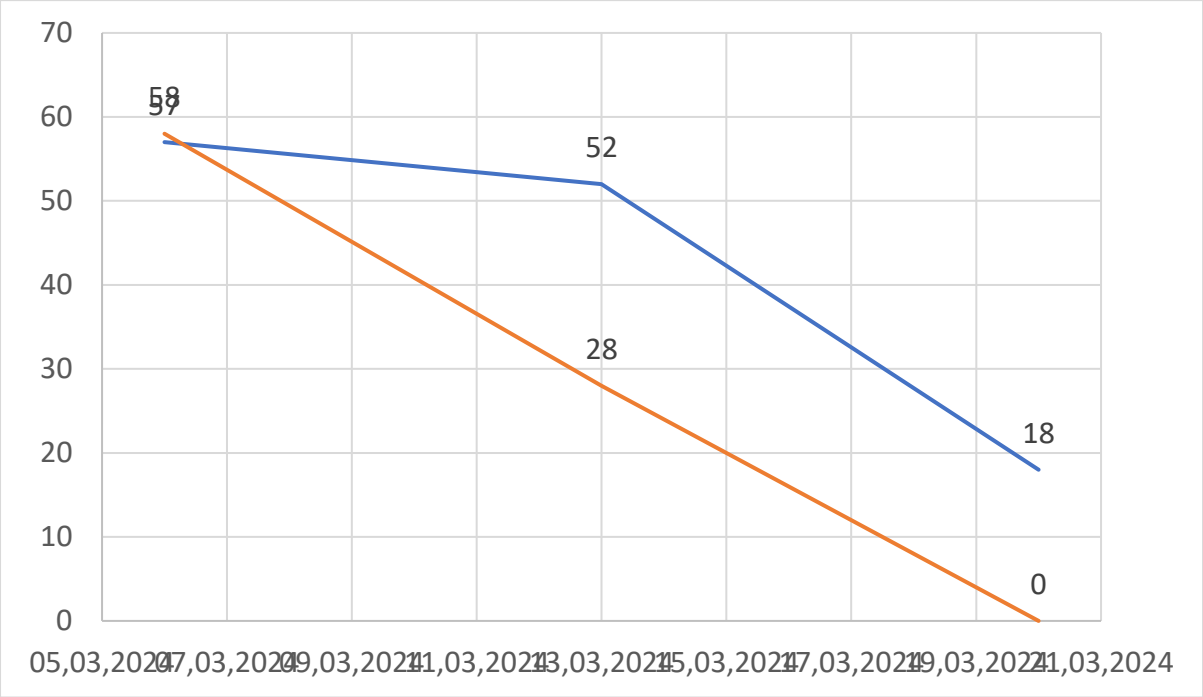
Gut	Schlecht
Bessere zeitliche Einschätzung	Kommunikationsschwierigkeiten

Impediment Liste
Kommunikationsschwierigkeiten
Verständnisprobleme zur SW-Architektur

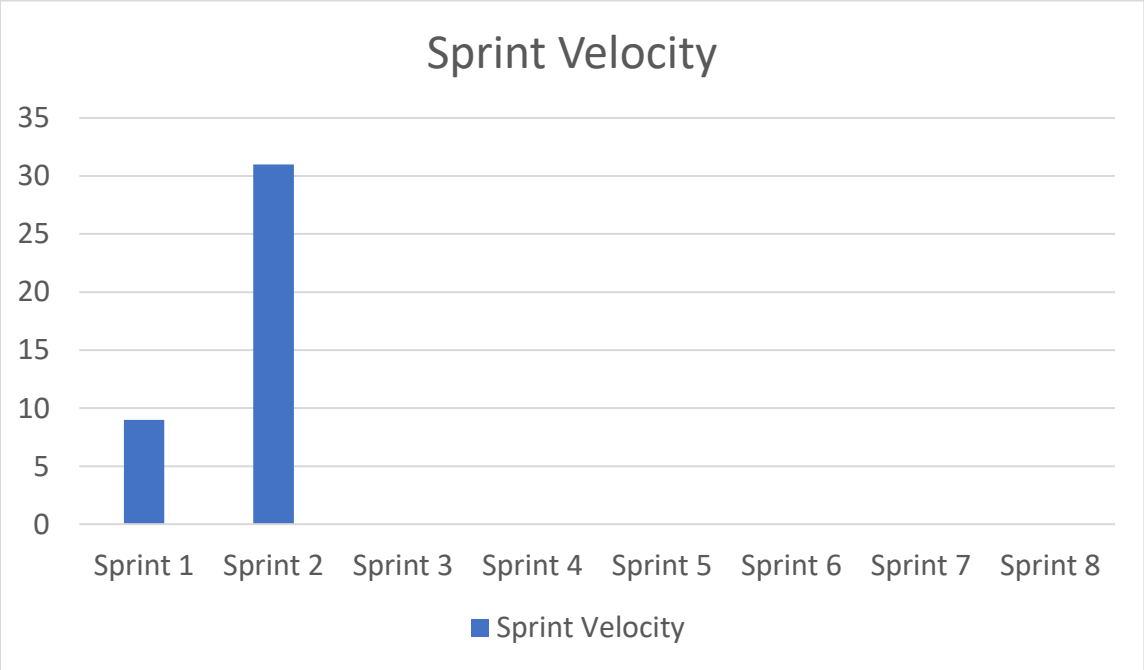
6.2.4 Sprint Zusammenfassung

Wurden in diesem Sprint neue User Stories in das Product Backlog eingefügt und wenn ja, welche.
Wurden in diesem Sprint User Stories aus dem Product Backlog entfernt und wenn ja, welche und warum.

Burndownchart



Sprint Velocity



Durchschnittliche Velocity: 20

6.3 Sprint 3

6.3.1 Sprintplanung

Dauer: 21.03.2024-10.04.2024

Name	Nummer	Kurzbeschreibung	Story Points
Server Broadcast	19	Add code which listens to broadcasts	5
MBot send sensor data to server	17	Sends the data of the MBot to the server	5
Make DetailedView	7	Make a detailed view of the mbot in the client	5
Setup Spring Server	15	Setup the Server which connects MBots and client	21
MBot receive input	9	Mbot should receive input from server to control its movement	3
Send client commands to Mbot	21	Forwards the clients commands to the mbot	5
Send data to client	23	Send the MBots data continuously to the client	8
Send user input to the server	13	Send the commands to the server	5

Anzahl Story points: 57

Ausgewählte Punkte aus der Impediment Liste: ---

6.3.2 Sprint Demo

Kurze Beschreibung welche User Stories umgesetzt worden sind und welche Funktionen erfolgreich präsentiert worden sind.
Auflistung welche Punkte nicht umgesetzt werden konnten und warum.

Name	Nummer	Nicht, weil…
Server Broadcast	19	
MBot send sensor data to server	17	
Make DetailedView	7	
Setup Spring Server	15	
MBot receive input	9	
Send client commands to Mbot	21	Zeitlich überschätzt
Send data to client	23	Zeitlich überschätzt
Send user input to the server	13	Zeitlich überschätzt

6.3.3 Sprint Retrospektive

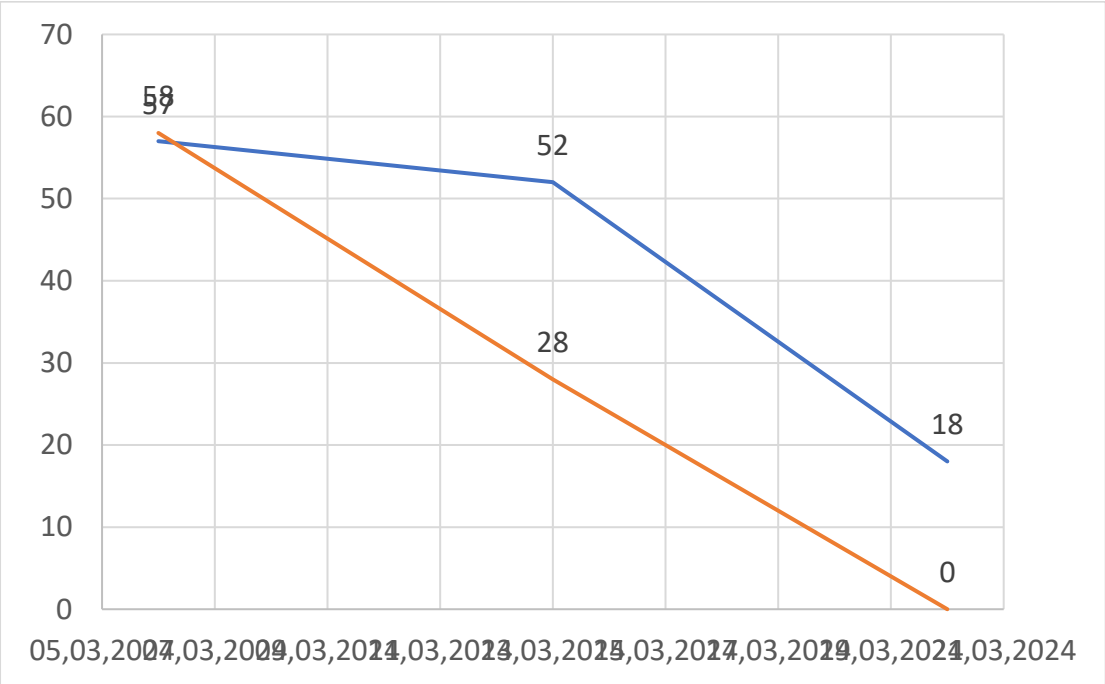
Gut	Schlecht
Beseitigung von Abhängigkeiten	Kommunikationsschwierigkeiten

Impediment Liste
Kommunikationsschwierigkeiten
Verständnisprobleme zur SW-Architektur

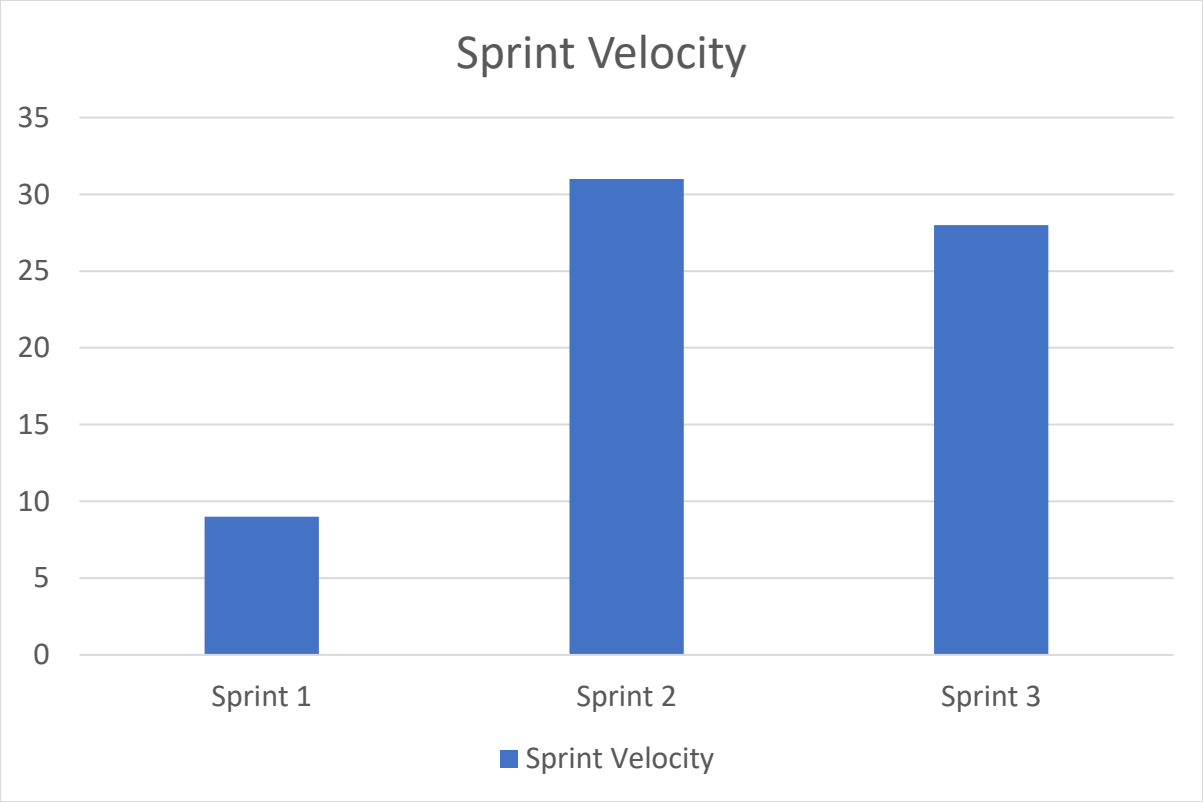
6.3.4 Sprint Zusammenfassung

Wurden in diesem Sprint neue User Stories in das Product Backlog eingefügt und wenn ja, welche.
Wurden in diesem Sprint User Stories aus dem Product Backlog entfernt und wenn ja, welche und warum.

Burndownchart



Sprint Velocity



Durchschnittliche Velocity: 20

6.4 Sprint 4

6.4.1 Sprintplanung

Dauer: 18.04.2024 – 16.05.2024

Anzahl Story points: 56

Ausgewählte Punkte aus der Impediment Liste: ---

User Story Nummer	Bearbeitet von	User Story Text	StoryPoints
33	David Legenjovic	Socket Connection with Server	6
30	Jonas Maier	Fix server bug relating client disconnections	8
34	Jonas Maier	Add UDP server	8
37	David Legenjovic	Improve MBotDetailPage	5
40	Jonas Maier	Additional information for data transfer	3
38	Jonas Maier	Add error handling when server stops existing	5
35	Jonas Maier	Fix database bug	13
25	Daniel Jessner	Solve threading problem on micropython controller, optimize code	8

6.4.2 Sprint Demo

Kurze Beschreibung welche User Stories umgesetzt worden sind und welche Funktionen erfolgreich präsentiert worden sind.
Auflistung welche Punkte nicht umgesetzt werden konnten und warum.

Name	Nummer	Nicht, weil…
Socket connection with server	33	
Server bug, client disconnections	30	
Add UDP server	37	
Add info to data transfer	40	
Server error handling	38	Nicht fertig geworden
Fix database bug	35	Nicht fertig geworden
Threading problem on mbot controller	23	Nicht fertig geworden

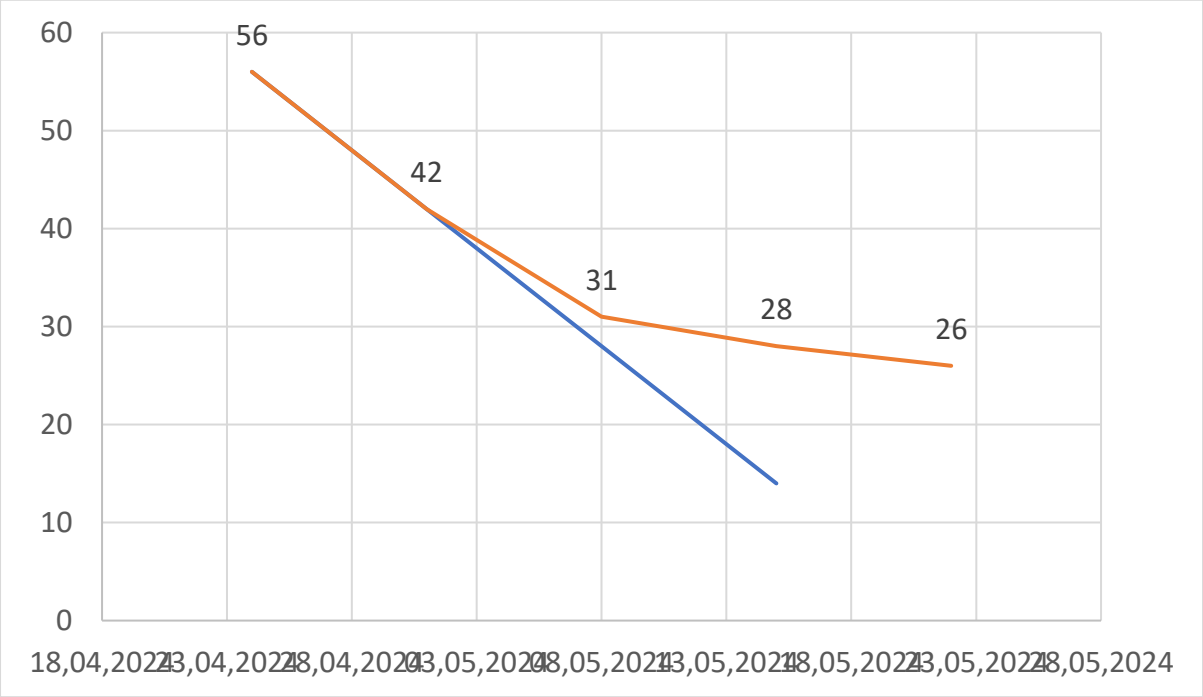
6.4.3 Sprint Retrospektive

Gut	Schlecht
Beseitigung von Abhängigkeiten	Kommunikationsschwierigkeiten

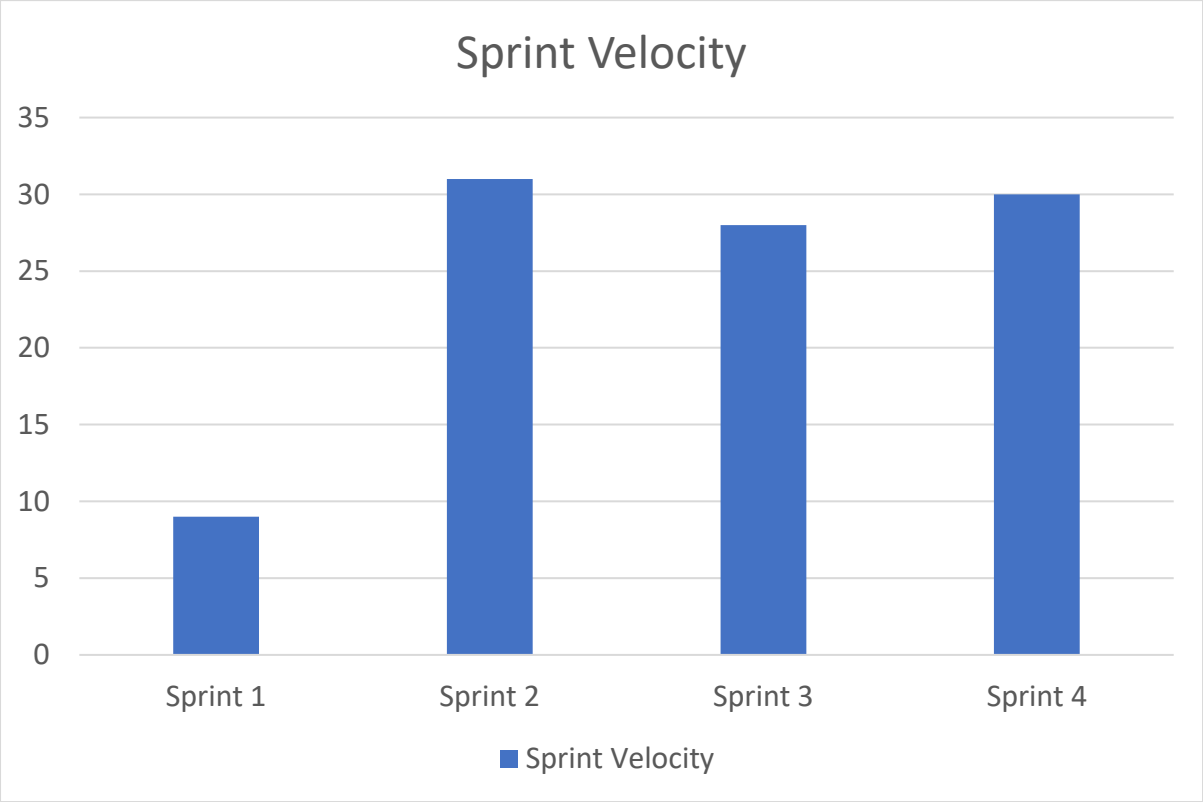
Impediment Liste
Kommunikationsschwierigkeiten
Verständnisprobleme zur Arbeitsaufteilung

6.4.4 Sprint Zusammenfassung

Burndownchart



Sprint Velocity



Durchschnittliche Velocity: 20

6.5 Sprint 5 / Abschluss

6.5.1 Sprintplanung

User Story Nummer	Bearbeitet von	User Story Text	StoryPoints	Status
35	Jonas Maier	Fix DB Saving Bug	13	erledigt
38	David Legenjovic	Add error handling when server stops existing	5	Nicht fertig
48	Jonas Maier	activate line followermode	13	Nicht fertig
49	Jonas Maier	activate suicide prevention	8	Nicht fertig
12	David Legenjovic	Add 3D map to MBot Detail View	21	Nicht begonnen

6.5.2 Sprint Demo

Name	Nummer	Nicht, weil…
Fix DB Saving Bug	35	
Server error handling	38	Nicht fertig geworden
activate line followermode	48	Nicht fertig geworden
activate suicide prevention	49	Nicht fertig geworden
3D map	12	Nicht begonnen

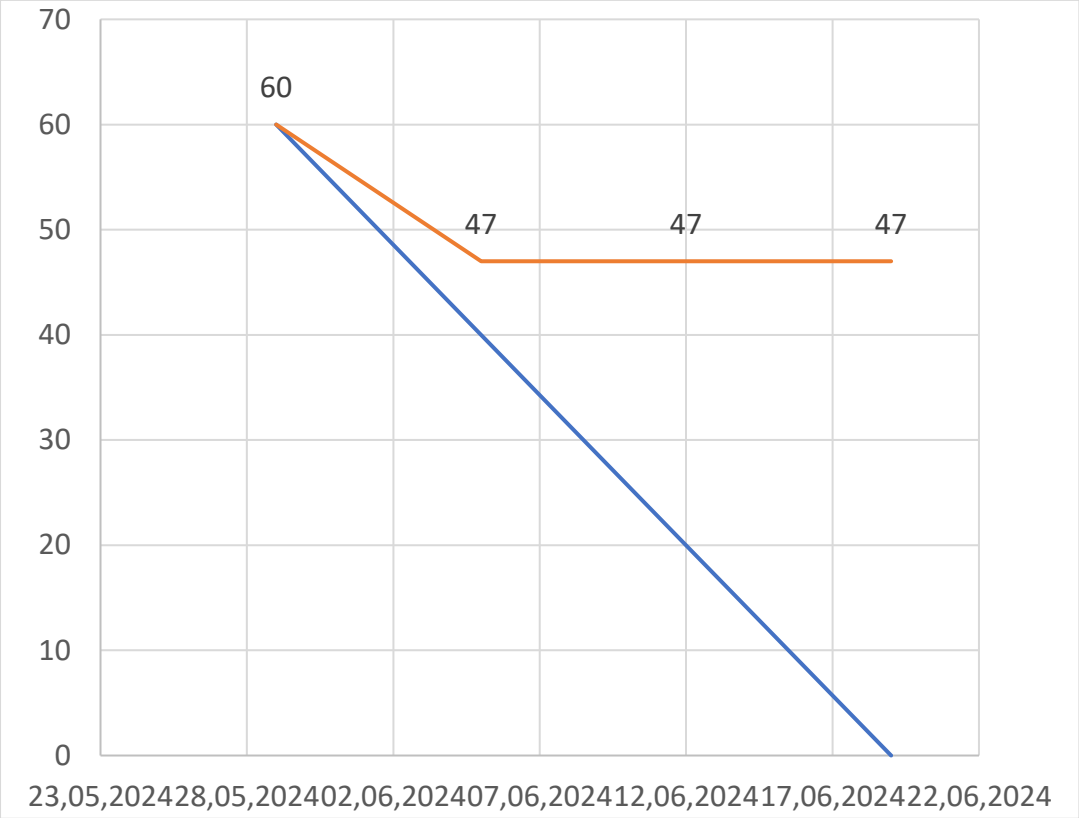
6.5.3 Sprint Retrospektive

7. Gut	Schlecht
Beseitigung von Kommunikationsproblemen	Kurze Sprint Dauer

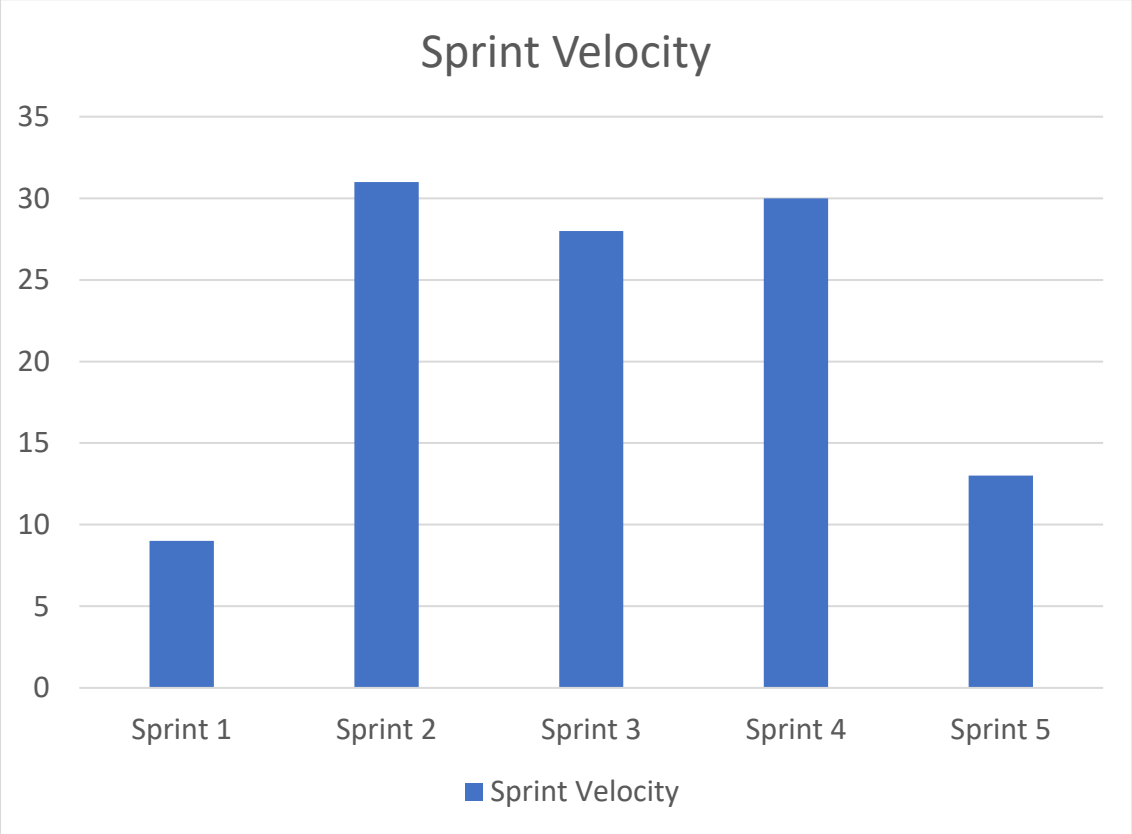
Impediment Liste
Kurze Sprint Dauer

7.1.1 Sprint Zusammenfassung

Burndown Chart



Sprint Velocity



8. Installation / Software deployment

Folgende Schritte sind für die Installation der Steuerungs-Software notwendig:

MBot:
- mBot mit jeweiligen Sensoren bereistellen
- Micropyhton-Skript mit passender Konfiguration befüllen und auf mBot hochladen
- mBot starten

Server:
- ZIP-Datei entpacken und abspeichern
- Passende Java Version installieren/konfigurieren
- Projekt in IDE öffnen und konfigurieren/indexen
- Server über IDE oder Konsole starten

Steuerungssoftware:
Android:
- ZIP-Datei entpacken
- Software über DIE oder EXE-Datei starten

Mobile:
- APK-Datei herunterladen
- Software installieren

9. Projektabschluss

9.1 Projektzusammenfassung

Was lief gut?

- **Teamarbeit und Engagement:** Das Team zeigte ein hohes Maß an Engagement und Zusammenhalt. Die Zusammenarbeit unter den Teammitgliedern war konstruktiv und unterstützend.
- **Zielerreichung:** Trotz der Herausforderungen wurden die Hauptziele des Projekts grundlegend erreicht.

Was lief schlecht?

- **Kommunikation:** Die Kommunikation war oftmals nicht ausreichend. Es gab Missverständnisse und Verzögerungen, weil Informationen nicht rechtzeitig oder nicht klar weitergegeben wurden.
- **Zeitliche Unterschätzung:** Der Zeitaufwand für bestimmte Projektphasen wurde unterschätzt. Dies führte zu Stress in den Endphasen des Projekts.
- **Ressourcenplanung:** Die Ressourcenplanung war nicht immer optimal. Es gab Phasen, in denen wichtige Ressourcen nicht verfügbar waren, was zu Verzögerungen führte.

Welche Erkenntnisse wurden während der Durchführung des Projektes gewonnen?

- **Bedeutung klarer Kommunikation:** Eine klare und regelmäßige Kommunikation ist entscheidend für den Projekterfolg. Alle Teammitglieder müssen über den gleichen Informationsstand verfügen, um effizient arbeiten zu können.
- **Realistische Zeitplanung:** Es ist wichtig, den Zeitaufwand realistisch einzuschätzen und Pufferzeiten einzuplanen, um unerwartete Verzögerungen auffangen zu können.
- **Flexibilität und Anpassungsfähigkeit:** Flexibilität im Umgang mit Änderungen und Anpassungsfähigkeit an unvorhergesehene Herausforderungen sind essenziell für den Projekterfolg.

Was würde man nun anders machen bzw. wieder gleich machen?

- **Verbesserung der Kommunikation:** Künftige Projekte würden von einer klareren Kommunikationsstrategie profitieren. Regelmäßige „Meetings“ und klare Dokumentationsrichtlinien sollen eingeführt werden, nicht nur für die Stakeholder.
- **Realistischere Zeitplanung:** Eine gründlichere Analyse und Planung des Zeitaufwands für jede Projektphase würde sicherstellen, dass alle Aufgaben in der vorgegebenen Zeit erledigt werden können.
- **Erfolgreiche Aspekte beibehalten:** Der hohe Teamgeist und das Engagement sollen weiterhin gefördert werden. Team-building-Aktivitäten und Anerkennung für gute Leistungen würden beibehalten und weiter ausgebaut werden.
- **Optimierung der Ressourcenplanung:** Eine sorgfältigere und flexiblere Ressourcenplanung würde sicherstellen, dass alle notwendigen Ressourcen zur richtigen Zeit verfügbar sind.

9.2 Attachments

mBot:

Datei	Beschreibung
mBot_script.py	mBot-Skript für den Betrieb des Systems

Server:

Datei	Beschreibung
server.zip	Spring Boot Server

Steuerungssoftware:

Datei	Beschreibung
Steuerungssoftware.zip	Visualstudio-Projekt in C#

Sonstige Projektdateien:

Datei	Beschreibung
Definition_Of_Done.docx	Endprüfung, ob das Projekt fertig ist.
Sprint_1_Review.pptx	Projektstand-Präsentation nach 1. Sprint
Sprint_2_Review.pptx	Projektstand-Präsentation nach 2. Sprint
Sprint_3_Review.pptx	Projektstand-Präsentation nach 3. Sprint
Sprint_4_Review.pptx	Projektstand-Präsentation nach 4. Sprint
Sprint_5_Review.pptx	Projektabschlusspräsentation
SYP_4_Projektbeschreibung_mbot2.docx	Projektbeschreibung von den Stakeholdern
SYP_4_Projektdokumentation_GRUPPE1.docx	Dokumentation des Projekts der Gruppe 1
aktivitätsdiagramme.drawio	Enthält alle Aktivitätsdiagramme
komponentendiagramm.drawio	Enthält Komponentendiagramm
UseCaseDiagramm.drawio	EnthältUseCase-Diagramm
Verteilungsdiagramm.drawio	Enthält Verteilungsdiagramm
Sequenzdiagramm1.png bis …7png	Sequenzdiagramme 1 bis 7
Github-Repo	https://github.com/huthuthiddeli/SYP_PROJEKT_23-24_Maier