

Verteilte Systeme

VS Praktikum SoSe 2025

Manh-An David Dao, Philipp Patt, Jannik Schön, Marc Siekmann

9. Juli 2025

Inhaltsverzeichnis

1	Einführung und Ziele	4
1.1	Aufgabenstellung	4
1.2	Qualitätsziele	5
1.3	Stakeholder	7
2	Randbedingungen	8
2.1	Technische Randbedingungen	8
2.2	Organisatorische Randbedingungen	8
3	Kontextabgrenzung	9
3.1	Fachlicher Kontext	9
3.1.1	Use Cases	9
3.1.2	Fachliche Randbedingungen	9
3.2	Technischer Kontext	10
3.2.1	Technische Anforderungen an die Middleware	10
3.3	Externe Schnittstellen	11
4	Lösungsstrategie	12
4.1	Controller	12
4.2	Model	13
4.2.1	StateService	13
4.2.2	MoveAdapter	14
4.2.3	ActuatorController	15
4.3	View	16
4.3.1	IO Funktionen	16
4.3.2	UI Funktionen	16
5	Bausteinsicht	17
5.1	Bausteinsicht Level 1	17
5.2	Bausteinsicht Level 2	18
5.3	Bausteinsicht Level 3 Model	19
5.4	Bausteinsicht Level 3 View	19
6	Laufzeitsicht	21
6.1	Sequenzdiagramme	21
6.2	FSM-Diagramme	23
6.3	Aktivitätsdiagramme	24
7	Verteilungssicht	25
7.1	Begründung	25
7.2	Qualitäts- und Leistungsmerkmale	26

7.3	Zuordnung von Bausteinen zu Infrastruktur	26
8	Konzepte	27
8.1	Offenheit	27
8.2	Verteilungstranzparenzen	27
8.3	Sicherheit (Safety)	27
8.4	Bedienoberfläche	27
8.5	Ablaufsteuerung	27
8.6	Ausnahme- und Fehlerbehandlung	27
9	Entwurfsentscheidungen	28
10	Qualitätsszenarien	29
10.1	Quality Requirements Overview	29
10.1.1	Ziele für Software Engineering	29
10.1.2	Ziele der Verteilte Systeme	31
10.2	Bewertungsszenarien	33
11	Risiken	34
11.1	Ziel des chapters	34
11.2	Technische Risiken	34
11.3	Technische Schulden	34
12	Glossar	35

1 Einführung und Ziele

1.1 Aufgabenstellung

Es soll eine Applikation entwickelt werden, die beliebig hinzugefügte Roboterarme erkennt und ansteuern kann. Jeder Roboter besteht aus vier unabhängigen Motoren. Über ein ITS-Board (STM32F4) sollen die Roboterarme innerhalb eines kontrollierten Areals (z.B. BT7 R7.65 – als realer Testbereich) sicher gesteuert werden. Mögliches Fehlverhalten der Software oder Architektur soll keine Gefahr für Anwesende darstellen. Der Benutzer soll einen zu steuernden Roboterarm auswählen und diesen anschließend mit den folgenden Bewegungen steuern:

- Roboterarm hoch
- Roboterarm runter
- Roboterarm links
- Roboterarm rechts
- Greifer auf
- Greifer zu

1.2 Qualitätsziele

Tabelle 1.1: Qualitätsziele der Software Engineering

Ziel	Beschreibung	Metrik
Funktionalität	Der ausgewählte Roboterarm muss Steuerbefehle korrekt umsetzen	Alle Abnahmetests werden erfolgreich bestanden
Zuverlässigkeit	Fehler dürfen den Betrieb nicht gefährden. Fehlererkennung und -toleranz müssen integriert sein.	Das System ist über dem gesamten Abnahmezeitraum stabil (ca. 1,5 h)
Skalierbarkeit	Zusätzliche Roboter oder Komponenten sollen ohne Änderungen an der bestehenden Architektur integrierbar sein.	Es können beliebig viele Roboter hinzugefügt und entfernt werden (0 - 254)
Leistung	Reaktionszeiten auf Steuerbefehle und Ereignisse müssen innerhalb definierter Zeitgrenzen liegen.	max. 200 ms bis Befehlsausführung
Sicherheit (Safety)	Es bewegt sich immer genau ein Roboterarm. Sollte das System nicht wie gewünscht reagieren, wird ein sicherer Zustand erreicht	Reaktionszeit max. 250 ms, bis Roboterarm stoppt
Wartbarkeit	Der Code muss übersichtlich sein, gut dokumentiert sein und wenig Komplexität enthalten.	Zyklomatische Komplexität ≤ 10 und LOC ≤ 30 pro Methode/Funktion exklusive Kommentar
Benutzerfreundlichkeit	Bedienung ist intuitiv, sodass die Nutzer möglichst wenig Zeit mit der Einarbeitung in die Bedienung benötigen.	Keine Einweisung erforderlich
Anpassbarkeit	Neue Funktionen, Sensoren oder Roboterarme sollen ohne tiefgreifende Änderungen am System integriert werden können.	Erweiterungen können durch modulare Struktur und erweiterbare Schnittstellen einfach hinzugefügt werden.

Ziel	Beschreibung	Metrik
Kompatibilität	Das System soll mit verschiedenen Hard- und Softwareplattformen kompatibel sein.	Es unterstützt die Kommunikation mit Embedded-Systemen.

1.3 Stakeholder

Tabelle 1.2: Interessen der Stakeholder

Stakeholder	Interesse
Nutzer	<ul style="list-style-type: none">• vollständige Funktionalität• Benutzerfreundlichkeit• Zuverlässigkeit: Das System kann über den gesamten benötigten Zeitraum ohne Ausfälle genutzt werden• Reaktionszeit: Die Robotersteuerung reagiert innerhalb eines definierten Zeitfensters• Sicherheit: Während des Betriebs kommen keine Personen durch Fehler des Systems zu Schaden
Betreiber	<ul style="list-style-type: none">• Portabilität: Das System kann auf verschiedenen Plattformen betrieben werden.• Zuverlässigkeit: Das System kann über den gesamten benötigten Zeitraum ohne Ausfälle genutzt werden• Sicherheit: Während des Betriebs kommen keine Personen durch Fehler des Systems zu Schaden
Entwicklerteam	<ul style="list-style-type: none">• Professor bzw. der „Kunde“ ist Mittwoch Nachmittag verfügbar. Bis dahin sind alle offenen Fragen zusammenzustellen.• Wartbarkeit• Portabilität: Das System kann auf verschiedenen Plattformen betrieben werden (z.B. Testen)• Austauschbarkeit: Softwaremodule können ohne großen Aufwand ersetzt werden
Professor	<ul style="list-style-type: none">• Zugang zu allen Arbeitsmitteln zwecks Bewertung und Kontrolle• Das Endprodukt besitzt alle geforderten Funktionalitäten

2 Randbedingungen

2.1 Technische Randbedingungen

Das verteilte Steuerungssystem unterliegt mehreren festgelegten technischen Rahmenbedingungen, die den Entwicklungs- und Implementierungsspielraum einschränken. Diese Bedingungen sind im Folgenden aufgeführt:

- **Hardwareplattform:** Das System basiert auf ein Raspberry Pi der den jeweiligen Roboterarm steuert. Eine Software auf dem Raspberry Pi stellt eine API zur Verfügung, sodass die Roboter sich ohne Einschränkungen bewegen können. Ein ITS-BRD dient als Steuerung/Administration der Roboterarme.
- **Programmiersprachen:**
 - **C:** Das ITS-Board wird mit C programmiert.
 - **Java:** Die Roboterarme werden mit einem Javaprogramm angesteuert.

2.2 Organisatorische Randbedingungen

- **Umgebung:** Der Abnahme bereich befindet sich im Raum BT7 R7.65. Die Steuerung und Navigation des Roboters müssen innerhalb der räumlich definierten Grenzen erfolgen.
- **Zeit:** Entwicklungszeitraum beträgt 12 Wochen.
- **Vorwissen:** Einige Konzepte und Herangehensweisen werden erst im Laufe der 12 Wochen gelernt.
- **Budget:** Es steht kein Budget zur Verfügung.

3 Kontextabgrenzung

Ziel dieses Kapitels ist es, das zu entwickelnde System innerhalb seines fachlichen und technischen Umfelds klar einzugrenzen. Dazu wird das System in Bezug auf seine Aufgaben (fachlicher Kontext), seine Einbettung in die bestehende technische Infrastruktur (technischer Kontext) sowie die definierten externen Schnittstellen beschrieben.

3.1 Fachlicher Kontext

Die Applikation ermöglicht es, beliebig viele Roboterarme (zwischen 1 und 254) in einem Raum zu steuern. Die Kommunikation mit dem Nutzer sowie die Steuerung der Roboterarme erfolgen über ein ITS-Board. Zur Unterstützung des Nutzers wird auf einer Benutzeroberfläche (UI) bereitgestellt, die eine intuitive Orientierung und Bedienung der Roboterarme ermöglicht.

3.1.1 Use Cases

ID	Name	Beschreibung
U1	Roboterarm auswählen	Der Nutzer ist in der Lage einen vollständigen Roboterarm auswählen zu können. Der Nutzer hat jederzeit eine Übersicht, welche Roboterarm ausgewählt werden können und ob keiner auswählbar ist
U2	Bewegung auslösen	Der Nutzer kann die Bewegung des Roboterarms auslösen und bekommt Feedback.
U3	Bewegungsrichtung unterscheiden	Die vom Nutzer ausgelöste Bewegungsrichtung wird dem richtigen Motor zugewiesen
U4	Neuen Motor erkennen	Die einzelnen Motoren melden sich an. Sobald ein neuer vollständiger Roboterarm (4 Motoren) angesteuert werden kann, ist dieser auswählbar und steuerbar.

3.1.2 Fachliche Randbedingungen

- Es können bis zu 254 Roboterarme ausgewählt werden
- Jeder Roboterarme hat 4 Motoren, die unabhängig voneinander sind
- Teile der Applikation sind auf dem ITS-Board zu implementieren
- Die Applikation ist sicher. Personen dürfen nicht zu Schaden kommen.

3.2 Technischer Kontext

Die Applikation besteht aus einem ITS-Board und jeder Roboterarm wird von einem vorgeschalteten Raspberry Pi angesteuert. Es existiert eine Benutzeroberfläche (UI). Optional kann die Applikation um externe Dienste, die auf anderen Plattformen, wie der ICC-Cloud erweitert werden.

3.2.1 Technische Anforderungen an die Middleware

Um die fachlichen Use Cases umzusetzen, muss die Middleware folgende technische Funktionen bereitstellen:

- **Roboterarm auswählen U1:**

- Der Nutzer sucht durch eine Eingabe den gewünschten vollständigen Roboterarm aus
- Die Applikation wertet die Auswahl aus, ob die Gruppe auswählbar ist
- Die Gruppe aus Motoren wird als "ausgewählt" für die Ansteuerung markiert
- Die Auswahl wird als "erfolgreich" markiert
- dem Nutzer werden die Auswahländerungen angezeigt und bekommt die "erfolgreich"-Mitteilung

- **Bewegung auslösen U2:**

- Der Nutzer betätigt eine Auswahl für die Bewegungsrichtung
- Der Steuerungsbefehl wird inklusive Richtung der Applikation übergeben
- Der Steuerungsbefehl wird übersetzt
- Die Bewegung wird anhand einer Prozentualen Bewegung durchgeführt
- Der erfolgreiche Befehl wird dem Nutzer mitgeteilt

- **Bewegungsrichtung unterscheiden U3:**

- Der Steuerungsbefehl wird inklusive Richtung der Applikation übergeben
- Anhand der Steuerungsrichtung wird entschieden welcher Motor angesprochen wird
- Es wird überprüft, ob der Motor angesprochen werden kann/auswählbar ist
- Der Steuerungsbefehl wird an den ausgewählten Roboterarm weitergegeben

- **Neuen Motor erkennen U4:**

- Ein neuer Motor wird angeschlossen
- Die Software des Motors sendet einen Anmelde-Anfrage mit dessen Zugehörigkeit und Bewegungsfunktion
- Der neue Motor wird eingetragen

- Es wird geprüft, ob durch den Motor eine kompletter Roboterarm nun auswählbar ist
- dem Nutzer wird ggf. mitgeteilt, dass ein neuer Roboterarm auswählbar ist

3.3 Externe Schnittstellen

- System: Der Benutzer kann über ein Bildschirm einzelne erreichbare Roboterarme erkennen. Die Auswahl und Steuern der einzelnen Roboterarme wird durch IO-Buttons realisiert.
- ITS-Board: Das ITS-Board kommuniziert über einen leichtgewichtigen UDP-Server mit einer eigenen IP-Adresse mit dem System.
- Roboterarm: Der Roboterarm wird über eine IP-Adresse angesprochen. Die Kommunikation erfolgt über einen vorgeschalteten Raspberry Pi, der einen eigenen UDP-Server und eine API zur Steuerung bereitstellt.

4 Lösungsstrategie

4.1 Controller

Funktion	Voraussetzung	Semantik
void update(AvailableRobots: String[], SelectedRobotIdx: int, Error: bool, Confirmation: bool)	Modell hat Änderungen	Benachrichtigt den Controller, der die View aktualisiert.

Tabelle 4.1: Funktionen des Controllers

4.2 Model

4.2.1 StateService

Funktion	Voraussetzung	Semantik
void select(SelectDirection selectDirection)	Liste der verfügbaren Roboter ist nicht leer	Wählt einen Roboter aus der Liste der verfügbaren Roboter aus und aktualisiert den Status.
void select(int sd)	sd ist ein gültiger Index für die SelectDirection-Enum	Ruft die Auswahl eines Roboters basierend auf dem Richtungsindex auf.
void setError(boolean error, boolean confirm)	Änderung des Fehler- oder Bestätigungsstatus notwendig	Setzt den Fehler- und Bestätigungsstatus und benachrichtigt den Controller.
void register(String motorName)	Motorname im richtigen Format (z.B. R1A1)	Registriert den Motor und fügt den zugehörigen Roboter der Verfügbarkeitsliste hinzu, falls dieser vollständig ist.
void subscribe()	keine	Placeholder-Methode, aktuell ohne Implementierung.
String getSelected()	Ein Roboter ist ausgewählt (selectedRobot > 0)	Gibt den Namen des aktuell ausgewählten Roboters zurück.
private void sendUpdate()	Änderungen im Status des StateService	Benachrichtigt den Controller über die aktuellen Verfügbarkeiten und Zustände.
boolean isAvailable()	keine	Prüft, ob alle vier Aktoren (A1, A2, A3, A4) des Roboters aktiviert sind. Liefert true , wenn der Roboter vollständig verfügbar ist.

Tabelle 4.2: Funktionen der Komponente StateService inklusive Robot-Klasse

4.2.2 MoveAdapter

Funktion	Voraussetzung	Semantik
void move(int md)	md ist ein gültiger Index der RobotDirection-Enum	Ruft die Bewegungsfunktion basierend auf dem Richtungsindex auf.
void move(RobotDirection rd)	Ein Roboter muss im State-Service ausgewählt sein	Führt die Bewegung des Roboters in die angegebene Richtung aus. Bei Erfolg wird der Fehlerstatus im State-Service zurückgesetzt. Bei Fehler wird der Fehlerstatus im StateService gesetzt.

Tabelle 4.3: Funktionen des MoveAdapter

4.2.3 ActuatorController

Funktion	Voraussetzung	Semantik
void move(int md)	md ist ein gültiger Index der Direction-Enum	Wandelt den Index in eine Richtung um und ruft die entsprechende Bewegungsfunktion auf.
void move(Direction direction)	Der Aktuator ist initialisiert	Bewegt den Aktuator in die angegebene Richtung (INCREASE oder DECREASE) und setzt den neuen Wert.
private void applyValue()	Ein gültiger Aktuatorname muss gesetzt sein	Setzt den aktuellen Wert des Aktuators physisch am Roboterarm und gibt den Zustand in der Konsole aus.
int getValue()	keine	Gibt den aktuellen Positionswert des Aktuators zurück.

Tabelle 4.4: Funktionen der Komponente ActuatorController

4.3 View

Die View besteht aus den unabhängigen Blöcken IO und UI. Die UI bietet eine Softwareschnittstelle an, die IO keine.

4.3.1 IO Funktionen

Funktion	Voraussetzung	Semantik
void readInputs()	Eingabe durch den Benutzer erfolgt	Überprüft die Benutzereingaben und löst entsprechende Aktionen aus.
int initIO()	IO-Hardware verfügbar	Initialisiert und überprüft die IO-Hardwareschnittstellen und gibt einen Fehlercode bei Problemen zurück.

Tabelle 4.5: IO Funktionen

4.3.2 UI Funktionen

Funktion	Voraussetzung	Semantik
void update-View(AvailableRobots: String[], SelectedRobotIdx: int, Error: bool, Confirmation: bool)	Gültige Modell-Daten vorhanden	View-Schnittstelle. Aktualisiert die UI mit den neuesten Roboter-Daten und Statusinformationen (Fehler, Bestätigung).

Tabelle 4.6: UI Funktionen

5 Bausteinsicht

Um ein besseres Verständnis über die Struktur des Systems zu bekommen, nutzen wir die Bausteinsicht. Sie hilft dabei ein gemeinsames Verständnis des Systems innerhalb des Teams zu bekommen. Die zur Zerlegung benutzte Dekompitionsstrategie ist funktional.

5.1 Bausteinsicht Level 1

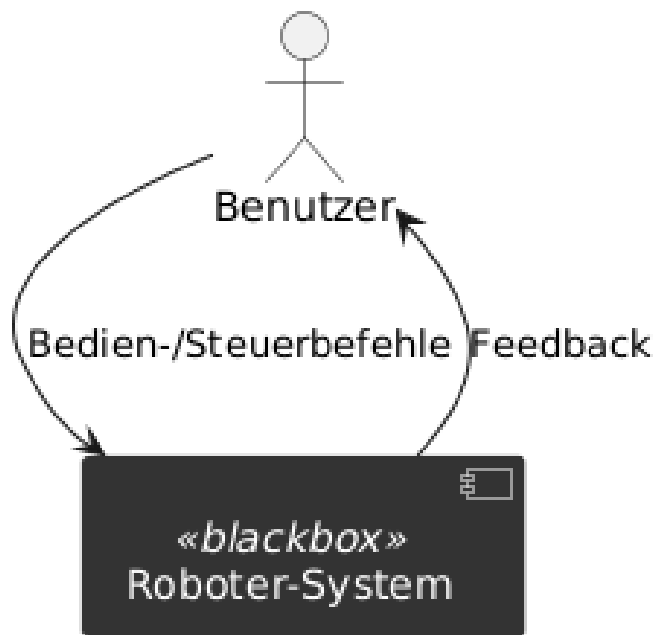


Abbildung 5.1: Bausteinsicht Level 1

Komponente	Beschreibung
Roboter-System	Gesamtsystem, das alle internen Steuer-, Safety- und Kommunikationsfunktionen kapselt. Empfängt Bedien-/Steuerbefehle vom Benutzer, verarbeitet sie und liefert Feedback.

Tabelle 5.1: Bausteinsicht Level 1

5.2 Bausteinsicht Level 2

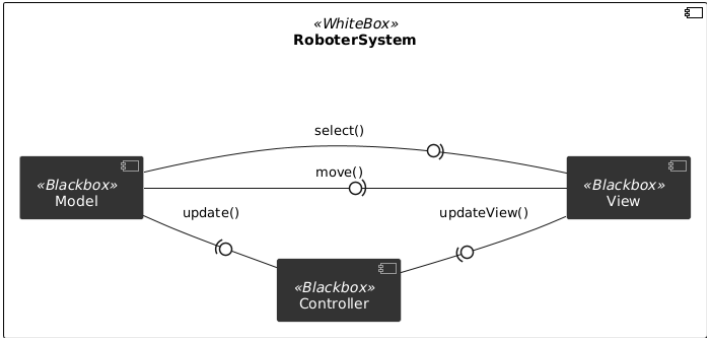


Abbildung 5.2: Bausteinsicht Level 2

Komponente	Beschreibung
Model	Beinhaltet die Geschäftslogik. Sendet Zustandsänderungen an den Controller.
Controller	Der Controller fungiert als Observer vom Model und gibt Zustandsänderungen des Models an die View weiter.
View	Bietet Interaktionsmöglichkeiten für den Nutzer und stellt dem Benutzer Informationen dar.

Tabelle 5.2: Bausteinsicht Level 2

5.3 Bausteinsicht Level 3 Model

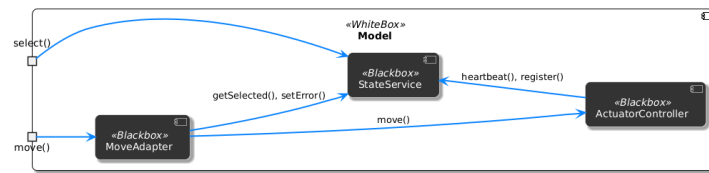


Abbildung 5.3: Bausteinsicht Level 3 Model

Komponente	Beschreibung
StateService	Speichert den ausgewählten Roboter und die verfügbaren Roboter. Bei Zustandsänderungen informiert er den Controller.
MoveAdapter	Empfängt einen Steuerungsbefehl vom View, übersetzt diesen mithilfe des StateServices, um den passenden Aktuator anzusprechen.
ActuatorController	Empfängt einen Steuerwert, überprüft die Gültigkeit und setzt diesen mithilfe der ICadsRoboticArm API.

Tabelle 5.3: Bausteinsicht Level 3

5.4 Bausteinsicht Level 3 View

Komponente	Beschreibung
IO	Leitet die Eingaben des Benutzers an den MoveAdapter weiter.
UI	Stellt dem Nutzer dar welche Roboter verfügbar sind, welcher ausgewählt ist und informiert über Fehler.

Tabelle 5.4: Bausteinsicht Level 3

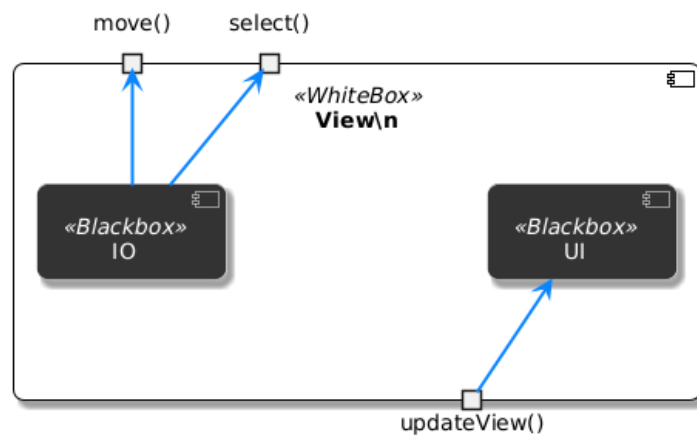


Abbildung 5.4: Bausteinsicht Level 3 View

6 Laufzeitsicht

6.1 Sequenzdiagramme

Szenario I: Auswählen eines Roboters

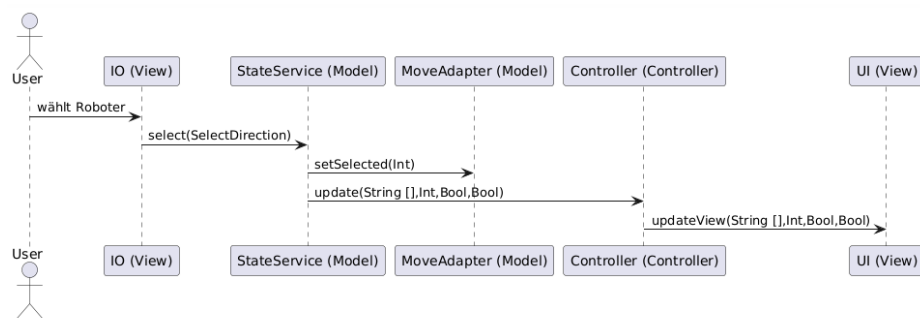


Abbildung 6.1: Auswahl des Roboters

Szenario II: Bewegungsbefehl über GUI

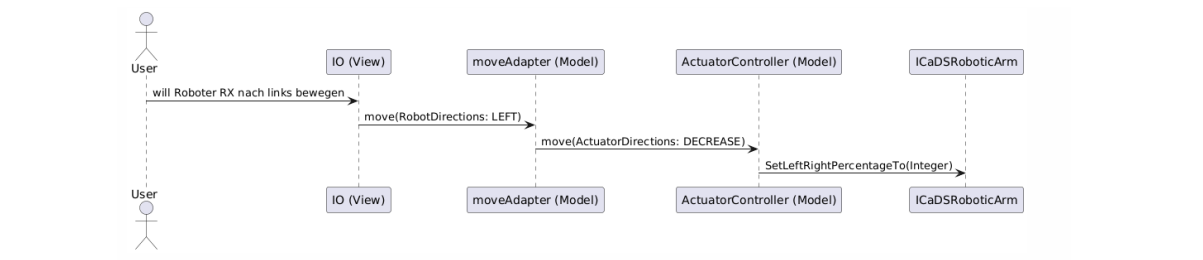


Abbildung 6.2: Bewegungsbefehl über GUI

6.2 FSM-Diagramme

FSM I: IO

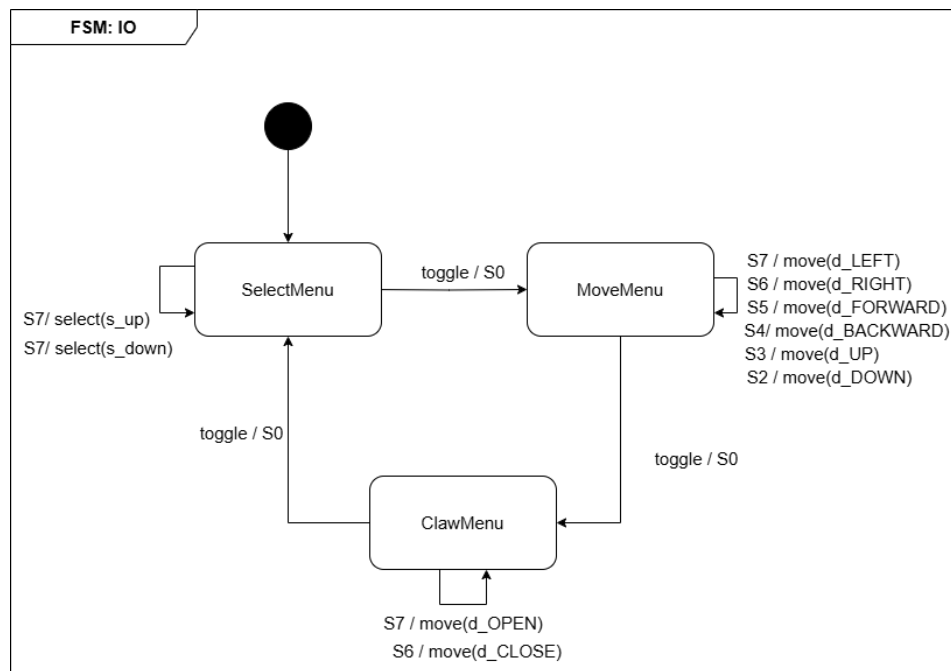


Abbildung 6.3: IO States Diagramm

6.3 Aktivitätsdiagramme

7 Verteilungssicht

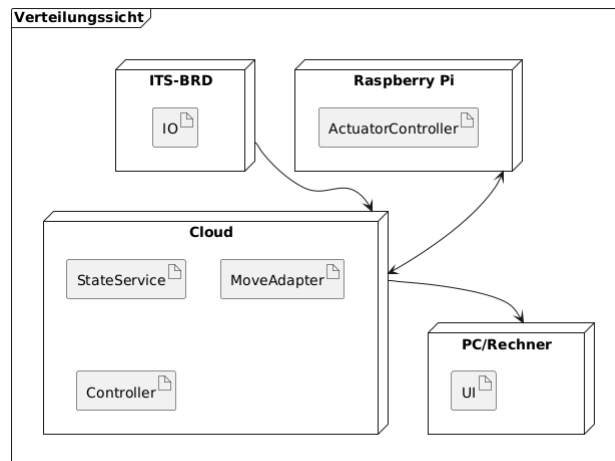


Abbildung 7.1: Deployment Diagramm

7.1 Begründung

Die Verteilung der Software-Bausteine auf verschiedene Hardware-Komponenten (PC, ITS-BRD, Raspberry Pi, ICC-Cloud) wurde gewählt, um folgende Systemanforderungen zu erfüllen:

7.2 Qualitäts- und Leistungsmerkmale

Merkmal	Beschreibung
---------	--------------

Tabelle 7.1: Qualitäts- und Leistungsmerkmale der Verteilungssicht

7.3 Zuordnung von Bausteinen zu Infrastruktur

Software-Baustein	Hardware-Komponente
UI	PC/Rechner
IO	ITS-BRD
ActuatorController	Raspberry Pi
StateService	Cloud
MoveAdapter	Cloud
Controller	Cloud

Tabelle 7.2: Zuordnung der Software-Bausteine zu den Infrastrukturkomponenten

8 Konzepte

8.1 Sicherheit (Safety)

Für jede Betätigung des Nutzer wird nun ein Bewegungsbefehl an den Roboter weitergegeben. Sobald der Nutzer die Taste loslässt, hört der Roboter auf sich zu bewegen.

8.2 Bedienoberfläche

Die Bedienung wird textuell auf dem Display des ITS-Boards dargestellt. Es wird auf Einfachheit und Eindeutigkeit geachtet.

8.3 Ablaufsteuerung

Ablaufsteuerung von IT-Systemen bezieht sich sowohl auf die an der (grafischen) Oberfläche sichtbaren Abläufe als auch auf die Steuerung der Hintergrundaktivitäten. Zur Ablaufsteuerung gehört daher unter anderem die Steuerung der Benutzungsoberfläche, die Workflow- oder Geschäftsprozesssteuerung sowie Steuerung von Batchabläufen.

8.4 Ausnahme- und Fehlerbehandlung

Das ITS-Board ruft per RPC die API der Middleware auf, diese wird dann in eine Bewegung des Roboterarms übersetzt. Die Nachrichten werden asynchron ausgetauscht. Die Anmeldung des Roboters erfolgt ebenfalls durch einen RPC aufruf. am ITS-Board.

9 Entwurfsentscheidungen

Entscheidung	Alternativen	Begründung	Woche
Logik liegt auf Raspberry Pi des zu steuernden R-Arms	Auf ITS-Board, Externer Server	Leistungsstark, kein Single Point Of Failure, Selbstverwaltung, erweiterte Skalierbarkeit	2

Tabelle 9.1: Zentrale Entwurfsentscheidungen

10 Qualitätsszenarien

- Bezug auf section 1.2 - weniger wichtige Requirements müssen genannt werden

10.1 Quality Requirements Overview

-

10.1.1 Ziele für Software Engineering

Ziel	Beschreibung	Metrik
Funktionalität		Alle Abnahmetests werden erfolgreich bestanden
Zuverlässigkeit	Teilausfälle dürfen den Gesamtsystembetrieb nicht gefährden. Fehlererkennung und -toleranz müssen integriert sein.	Das System ist über dem gesamten Abnahmezeitraum stabil (ca. 1,5 h)
Skalierbarkeit	Zusätzliche Roboter oder Komponenten sollen ohne Änderungen an der bestehenden Architektur integrierbar sein.	Es können beliebig viele Roboter hinzugefügt und entfernt werden (0 - N)
Leistung	Reaktionszeiten auf Steuerbefehle und Ereignisse müssen innerhalb definierter Zeitgrenzen liegen.	Reaktionszeit max. 250 ms
Sicherheit (Safety)	Es bewegt sich immer genau ein Roboterarm. Sollte das System nicht wie gewünscht reagieren, wird ein sicherer Zustand erreicht. Das System darf unter keinen Umständen eine Gefährdung für Personen/Gegenstände darstellen. Bei Fehlern muss sofort ein sicherer Zustand erreicht werden (z.B. Notstopp).	Reaktionszeit max. 250 ms, bis Roboterarm stoppt
Wartbarkeit	Der Code muss modular, gut dokumentiert und testbar sein. Fehlerdiagnose und Protokollierung sollen integriert sein.	
Portabilität	Die Software soll ohne großen Aufwand auf vergleichbaren Embedded-Systemen lauffähig sein.	
Benutzerfreundlichkeit	Konfiguration und Überwachung müssen intuitiv bedienbar und gut visualisiert sein.	Keine Einweisung erforderlich
Anpassbarkeit	Neue Funktionen, Sensoren oder Regeln sollen ohne tiefgreifende Änderungen am System integrierbar sein.	
Kompatibilität	Das System soll mit bestehenden Standards und Protokollen kommunizieren können.	

Tabelle 10.1: Qualitätsziele der Software Engineering

10.1.2 Ziele der Verteilte Systeme

Ziel	Beschreibung	Metrik
Ressourcenteilung	...	siehe Tabelle 10.3 siehe Tabelle 10.4
Offenheit	...	
Skalierbarkeit	...	
Verteilung Transparenz	...	

Tabelle 10.2: Qualitätsziele der Verteilten Systeme

Skalierbarkeit

Ziel	Metrik	Metrik
Vertikale Skalierung	...	
Horizontale Skalierung	...	
Räumliche Skalierbarkeit		1
Funktionale Skalierbarkeit	...	
Administrative-Skalierbarkeit		1

Tabelle 10.3: Skalierbarkeit von verteilten Systemen

Verteilungs-Transparenzen

Ziel	Beschreibung	Metrik
Zugriffstransparenz	...	
Lokalitäts-Transparenz	...	
Migrationstransparenz	...	
Replikationstransparenz	...	
Fehlertransparenz	...	
Ortstransparenz	..	
Skalierbarkeits- Transparenz	...	

Tabelle 10.4: Verteilungs-Transparenzen

10.2 Bewertungsszenarien

ID	Context / Background	Source / Stimulus	Metric / Acceptance Criteria
QS-1	Gesamtsystem betriebsbereit. Der Roboter befindet sich im Ruhezustand (Stop).	Bediener sendet Bewegungsbefehl und das Stromkabel des ITS Board wird gezogen	Roboter geht innerhalb von 250 ms in den Ruhezustand.
QS-2	Gesamtsystem betriebsbereit. Der Roboter befindet sich im Ruhezustand.	Bediener sendet Bewegungsbefehl und das Stromkabel des Raspberry Pi wird herausgezogen.	Roboter innerhalb von 250 ms in den Ruhezustand.

Tabelle 10.5: Bewertungsszenarien nach q42-Modell

11 Risiken

Beispielintrag für ein Glossarverweis: Eine Liste identifizierter technischer Risiken oder technischer Schulden, nach Priorität geordnet!

11.1 Ziel des chapters

Frühzeitige Identifikation und Dokumentation technischer Risiken
Aufzeigen von bewusst eingegangenen technischen Schulden
Unterstützung bei Risikomanagement und fundierter Entscheidungsfindung

11.2 Technische Risiken

Fehlende Erfahrungen, Komplexität und Externe Abhängigkeiten

11.3 Technische Schulden

Hardcodierung und Fehlende Test

Form:

Eine Liste der jeweiligen Risiken und Schulden. Zusätzlich sollten vorrangsgehensweise und lösungswege hinzugefügt werden. Wie ist man mit den Risiken umgegangen? Warum nimmt man die Schuld in kauf?

12 Glossar

Begriff	Definition Erklärung
GUI-Software	Überbegriff für Steuerungssoftware, die mit Hilfe des LCD Displays dem Nutzer die Steuerung der Roboter ermöglicht. Es handelt sich hierbei um ein Programm/Executable
Roboter-Software	Überbegriff für Software, die für die direkte Steuerung eines Roboters zuständig ist. Es handelt sich hierbei um ein Programm/Executable
Roboter/Roboterarm	Hardware, umfasst den Arm selbst sowie das entsprechende RaspberryPi

Tabelle 12.1: Glossar