System Architektur Spezifikation

3D-Bionics Hand

Version: 1.0

Datum: Stand 20.07.2021

Inhaltsverzeichnis

1 Einleitung

1.1 Zweckbestimmung

Das Dokument dient als umfassende Zusammenfassung und Überblick über die Architektur der **3D-Bionics Bionic Hand**. Die Architektur wird aus mehreren Blickwinkeln (Views) betrachtet um die verschiedenen Funktionen zu beschreiben. Es soll die Grundlage für die Entwicklung darstellen und wichtige, Architektur-spezifische Entscheidungen verständlich erklären.

1.2 Geltungsbereich / Scope

Von diesem Dokument aus wird die Architektur und Implementierung für die Software des **3D-Bionics Bionic Hand** gesteuert. Darunter fallen:

- Hardware-Software Verbund
  - Kommunikationsstandart
  - Kontrolle der Software über die Hardware
- Modul-Aufbau
- Klassendeklarationen
- Benutze Dependencies
- Oberflächen Design

1.3 Definitionen und Abkürzungen

- **Hand, Handmodell:** Das Produkt für welches die Architektur geschrieben ist. Eine mechanische Hand, gesteuert über einen Arduino + Servomotoren.

  - **Im Kontext der Kontrollsoftware**, bezeichnet es die interne Software-Repräsentation der Hand

- **Arduino:** Ein in dem Handmodell verbauter Mikrocontroller. Steuert die Servomotoren.

- **Servomotor, Servo:** In dem Handmodell verbaute Servomotoren, welche für die Bewegungen in der Hand zuständig sind.

- **Finger**

  Als Finger wird das mechanische Bauteil an dem Hand-Modell bezeichnet, welches über den Arduino bzw. einen Servomotor bewegt wird. Die Finger sind gleichnamig den Bezeichnungen und Positionen an einer echten Hand nachempfunden. Sie werden können im folgenden Text wie folgt abgekürzt werden.

  - Kleiner Finger: `kF`
  - Ring Finger: `rF`
  - Mittel Finger: `mF`
  - Zeige Finger: `zF`
  - Daumen: `dF`

- **Animation:** Eine Animation beschreibt eine Abfolge von mehreren Positionen der Finger um eine bestimmten Bewegungsablauf der Hand darzustellen.

- **Kontroll-App, Kontroll-Software:** Applikation über welche die Hand gesteuert werden kann

- **UI, Userinterface:** User-Interface der Kontroll-App

- **Communication-Framework, Comframe, COM:** Framework, welches die Kommunikation der Positionen der einzelnen Finger zwischen der Kontroll-Software und dem Arduino regelt. Es legt die Grundstruktur und unterliegende Protokolle fest.

1.4 Überblick

Das SAS ist in 5 Teile untergliedert:

Innerhalb von **Ziele und Beschränkungen** werden die einzelnen Funktionsanforderungen beschrieben und unter welchen Beschränkungen die Implementierung dieser geschehen muss.

Der **Use-Case View** gibt einen Überblick über die tatsächliche Funktion des Programms anhand von verschiedenen Use-Cases.

Im **System Architektur Überblick** wird ein grober Überblick über die Grundfunktionen und -Ideen sowie Aufbau des Projektes gegeben. Es wir sowohl die Hardware-Software-Beziehung beschrieben sowie der konzeptionelle Aufbau der Kontroll-Software.

Der **Logical View** teilt dass Programm in Module auf und beschreibt dessen Funktionen und die Verbindungen zwischen den einzelnen Modulen welche auch als Grundlage für die Implementierung dienen. Des Weiteren werden auch die wichtigsten Klassen genauer beschrieben sowie die Verwendung der verschiedenen Libraries erläutert.

Der Punkt **Implementierung** beschreibt die Grundstruktur des Programmcodes und die zugrundeliegenden Konventionen. Außerdem wird gezeigt, wie sich die Programmfunktionen in einzelne Programmcode-Module aufteilen.

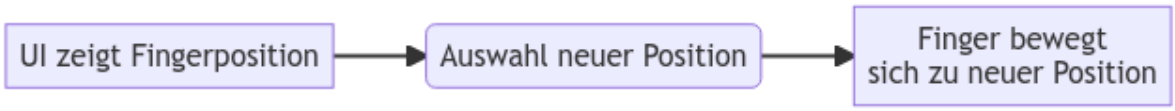
2 Ziele und Beschränkungen

Die Ziele und Beschränkungen sind innerhalb der System Requirement Spezification beschrieben.

3 Use-Case View

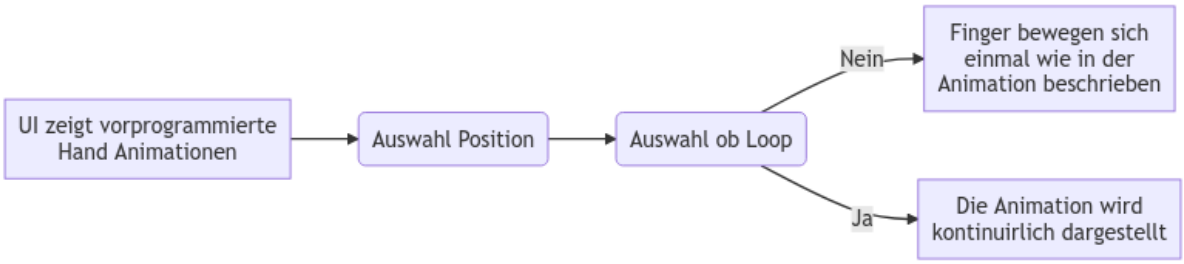
Der **Use-Case** View gibt einen Überblick über die tatsächliche Funktion des Programms anhand von verschiedenen Use-Cases. Diese sind grundgebend für den restlichen Ablauf des SAS. Ebenfalls werden die Module benannt welche für die jeweilige Funktion verantwortlich sind.

3.0.1 Einzelne Ansteuerung eines Fingers über die Kontroll Software

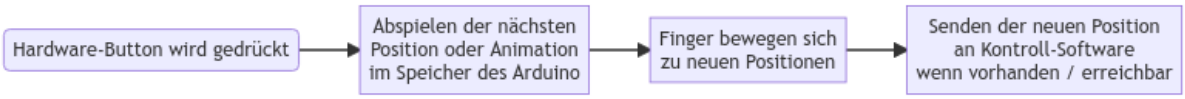


3.0.2 Ansteuerung der ganzen Hand über die Kontroll Software

3.0.3 Ablauf einer Animation (mit Loop) über die Kontroll Software



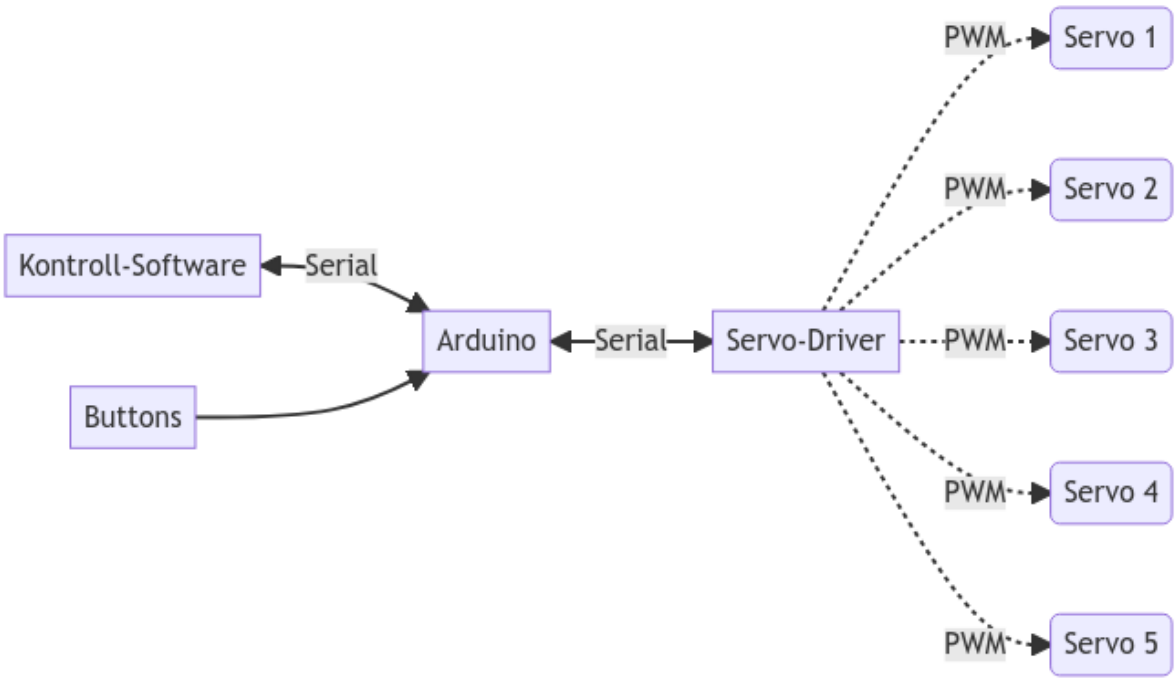
3.0.4 Festlegen einer vorprogrammierten Position oder Animation über Hardware-Button



4 System Architektur Überblick

4.1 Hardware-Software Verbund

4.1.1 Zusammenhang des Systems

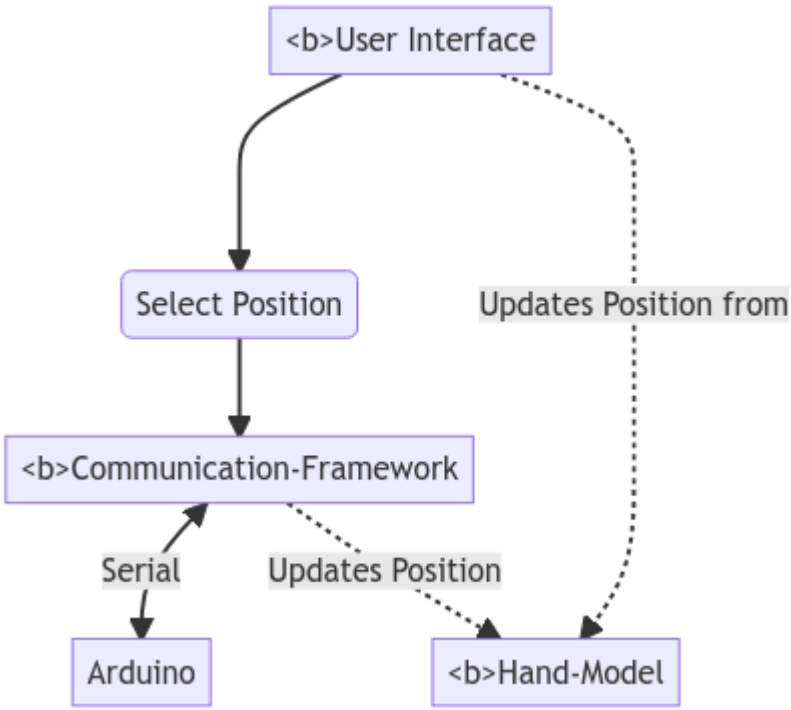


Die Kontrollsoftware sendet über eine serielle-Schnittstelle Datenpakete an den Arduino, welcher die einzelnen Servo-Motoren über einen externen Servo-Driver ansteuert.

Der Arduino (und damit die Servos) kann theoretisch auch über Buttons welche direkt an den Arduino angeschlossen sind angesteuert werden. In diesem Fall gibt der Arduino lediglich die neuen Positions-Werte der Finger an die Kontroll-App zurück.

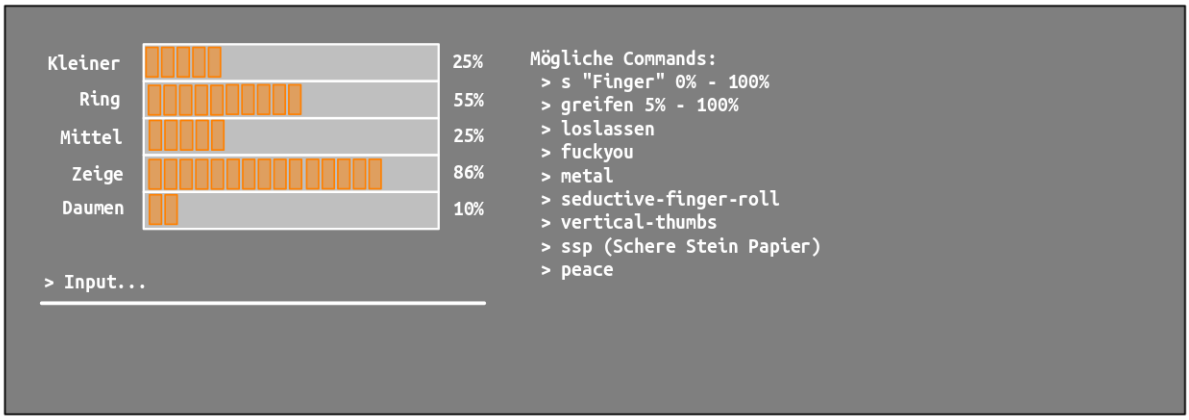
4.2 Kontroll-Software

4.2.1 Aufbau



Der Aufbau des Programms ähnelt des eines MVC-Paradigmas. Der Controller ist in diesem Fall das Communication-Framework. Es regelt die Kommunikation zwischen dem User-Interface und dem Arduino für die Hand Steuerung. Das Hand-Modell fungiert als Modell und repräsentiert die interne Position der Hand. Sie wird von dem Communication-Framework mit den Positionen des Arduino aktualisiert.

4.2.2 Mögliches UI Design



5 Logical View

5.1 Dependencies

5.1.1 PowerBroker2 / SerialTransfer

Library, welche für das tatsächliche transferieren von Informationen zwischen Arduino und Kontroll-Software verantwortlich ist. Sie dient als Backend für das Communication-Framework.

Es gibt eine separate Library für den Arduino und in Python.

5.1.1.1 Links

- PowerBroker2/SerialTransfer
- PowerBroker2/pySerialTransfer

5.1.2 NPYScreen

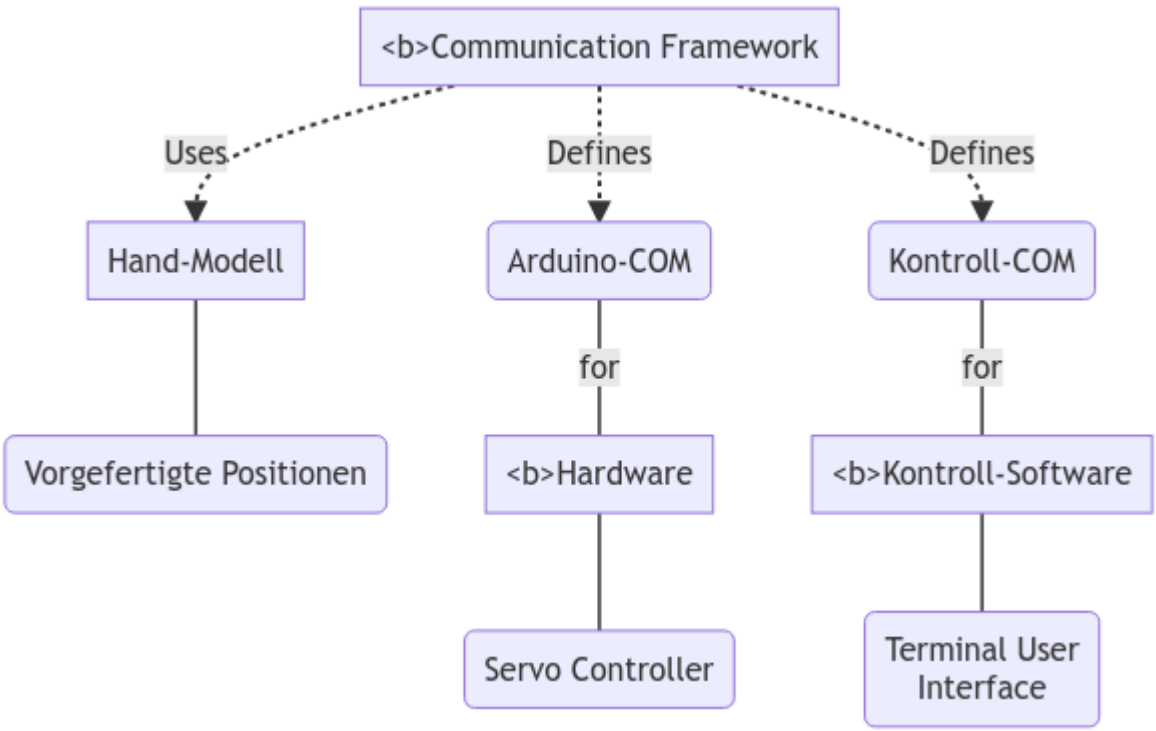
> *Npyscreen is a python widget library and application framework for programming terminal or console applications. It is built on top of ncurses, which is part of the standard library.*
>
> *Quelle*

Zuständig für:

- Anzeigen und Erstellen des Terminal User Interfaces (==Füge Funktionale Anfoderung aus SRS ein==)
- Managen des Main-Update-Loops, welchem auch das Communication-Framework untergestellt ist

5.2 Module

5.2.0.1 Übersicht



5.2.1 Communication Framework

Das Communication-Framework oder auch COM ist dafür zuständig die Positionen der einzelnen Finger zwischen der Kontroll-Software und dem Arduino zu managen. Es regelt die Kommunikation über die serielle Schnittstelle und legt das Protokoll der Übertragung fest.

Das Modul Communication-Framework beschreibt das Framework konzeptionell und legt die Anforderungen fest. Die eigentliche Implementierung des Frameworks ist in den Modulen Arduino-Com und Kontroll-COM individuell realisiert.

5.2.2 Hand-Modell

Das Modul **Hand-Modell** beschreibt die interne Repräsentation der Hand für die Kontroll-Software. Es dient außerdem als Basis für das gesendete Format des Hand-Modells für das Communication-Framework.

5.2.2.1 Vorgefertigte Positionen

Nach `SRS FA 3.1.2: Vorgefertigte Positionen`

Das Submodul legt das Format für Repräsentation und Speicherung von vorgefertigten Positionen sowie für Animationen fest.

5.2.2.1.1 Schere Stein Papier

Nach `SRS FA 3.1.2.1: Schere Stein Papier`

Für "Schere Stein Papier" wird eine Position zufällig ausgewählt, um mit der Hand "Schere Stein Papier" spielen zu können.

5.2.3 Hardware

Das Modul **Hardware** beschreibt alle Funktionalitäten welche direkt mit der Hardware zu tun haben. Dazu zählt zum einen die Programmierung des Arduinos sowohl die dafür benötigten Klassen zur Kontrolle der Servos (Submodul Servo-Controller) sowie die Kommunikation zur Kontroll-Software (Submodul Arduino-Com)

5.2.3.1 Servo-Controller

Das Submodul **Servo-Controller** beschreibt die gleichnamige Klasse *ServoControll* welche als Interface für die fünf Servos der Finger dient. Das Ziel ist es die Servos über ein Array wie es in dem Modul Hand-Modell beschrieben ist direkt anzusteuern.

5.2.3.2 Arduino-COM

Das Submodul **Arduino-COM** ist die Implementierung des Communication-Frameworks auf dem Arduino.

5.2.4 Kontroll-Software

Nach `SRS FA 3.1.3: Graphische Benutzeroberfläche`

5.2.4.1 Terminal User Interface

Das Modul **Terminal User Interface** beschreibt jegliche Funktionalität die für das Userinterface für die Kontroll-Software nötig ist. Dazu gehören:

- Forms und Widgetklassen für NPYScreen
- Callback-Definitionen für Interaktion mit der Software
- Einbindung des Kontroll-COM in die Kontroll-Sofware

5.2.4.2 Kontroll-COM

Das Submodul **Kontroll-COM** ist die Implementierung des Communication-Frameworks für die Kontroll-Software.

5.3 Modul: Hand-Modell

5.3.1 Interne Repräsentation der Hand

Die Hand wird software-intern als die Position der fünf einzelnen Finger dargestellt.

Die Streckung der Finger wird als eine Prozent-Angabe repräsentiert wobei 0% für "Finger komplett entspannt" und 100% für "Finger komplett geschlossen" steht.

5.3.1.1 Eigenschaften

- **Format: ** `[Position kF, Position rF, ...]`
- Die Reihenfolge innerhalb des Arrays ist **kF, rF, mF, zF, dF**

5.3.2 Animationen

Animationen werden intern als Array von Positionen (wie oben beschrieben) abgespeichert. Die Positionen werden nach einander gesendet um eine kontinuierliche Bewegung darzustellen. Das Timing zwischen den einzelnen Positionen ist abhängig von der Sende-Rate der einzelnen Positions-Updates.

5.3.2.1 Beispiel:

```
animation = [

    #[position kF,rF,mF,zF,dF] Format von Positionen

    [60, 50, 40, 30, 20],

    [70, 60, 50, 40, 30],

    [80, 70, 60, 50, 40],

    [90, 80, 70, 60, 50],

    ....

]
```

5.3.3 Speicherung von vorgefertigten Positionen

Positionen oder Animationen werden zusammen mit einem Namen, innerhalb einer Map gespeichert. Die Kontroll-Software ließt diese Map aus um die Menü-Optionen zu generieren.

5.4 Modul: Communication-Framework

Das Communication-Framework oder auch COM ist dafür zuständig die Positionen der einzelnen Finger zwischen der Kontroll-Software und dem Arduino zu managen. Es regelt die Kommunikation über die serielle Schnittstelle und legt das Protokoll der Übertragung fest.

Das Communication-Framework besteht aus zwei Teilen. Der Erste Teil befindet sich in der Kontroll-Software und der zweite im Arduino.

5.4.1 Aufgaben nach SRS

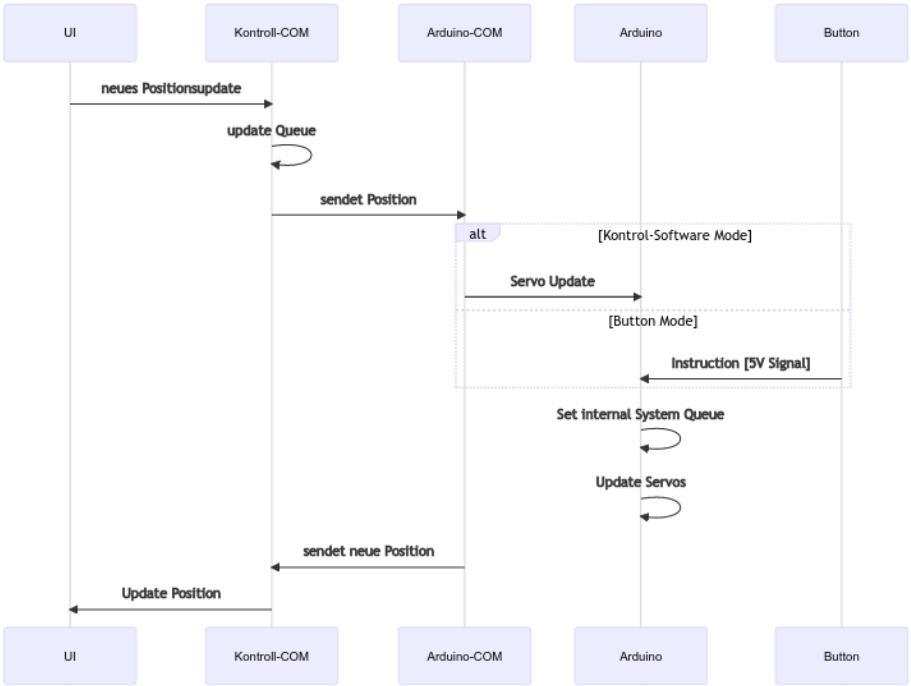
- Positionen der einzelnen Finger über die Kontroll-Software an den Arduino weitergeben `SRS 3.1.1`
- Komplexere Positionsabläufe über die Kontroll-Software an den Arduino weitergeben `SRS 3.1.2`
- Das interne Hand-Modell mit Positions-Updates des Arduinos zu aktualisieren `SRS 3.1.3`
- Automatisches Aufbauen einer Verbindung zwischen Kontroll-Software und Arduino über eine serielle Verbindung `SRS 3.2.1`

5.4.2 Kommunikations-Standard

5.4.2.1 Anforderungen

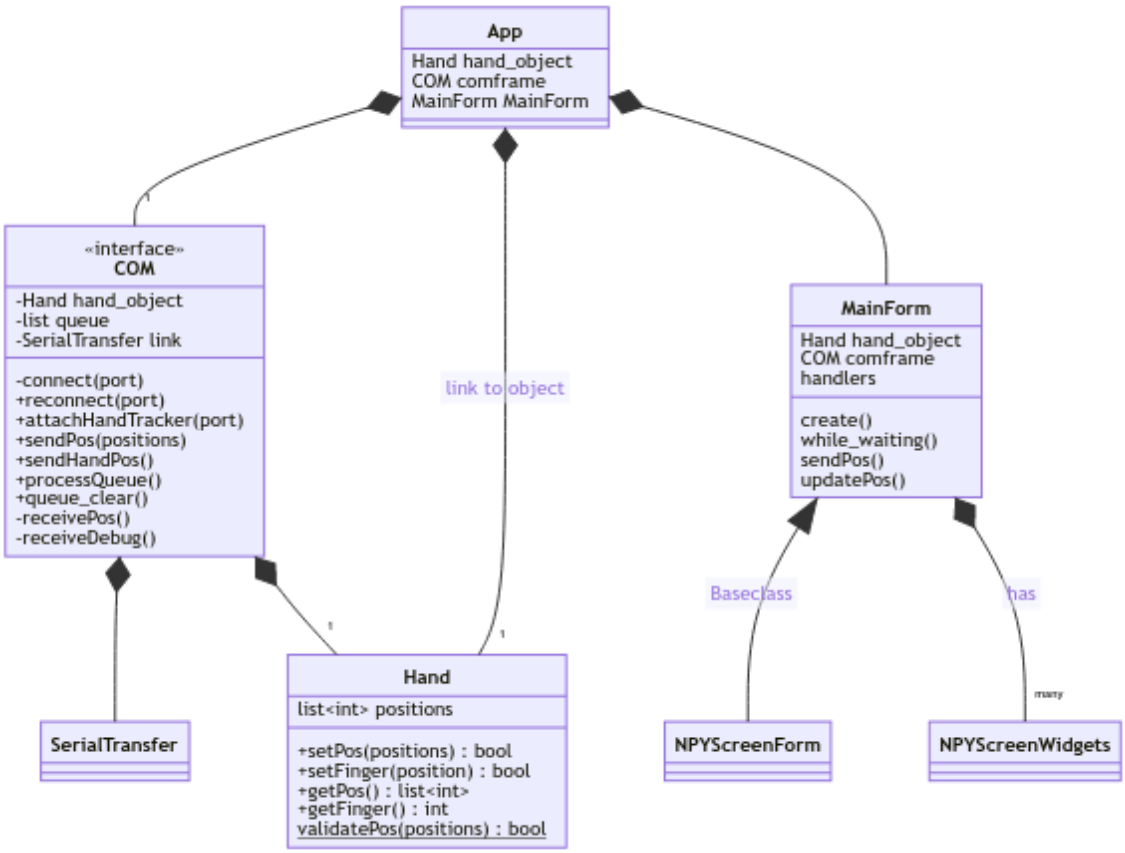
- Die Kommunikation läuft über eine serielle Schnittstelle zwischen Kontroll-Software und Arduino.
- Es werden nur dann Positionen gesendet wenn sich auch etwas an der Position der Finger verändert hat, oder sich etwas ändern soll
- Gesendete Positionen sind stets im Format wie es in Hand-Modell / Format beschrieben wird
- Eine Animation ist eine Liste von hintereinander gesendeten Positionen
- Das Communication-Framework hat eine interne Message-Queue, welche asynchron abgearbeitet werden kann.

5.4.2.2 Sequenz eines Positionsupdates

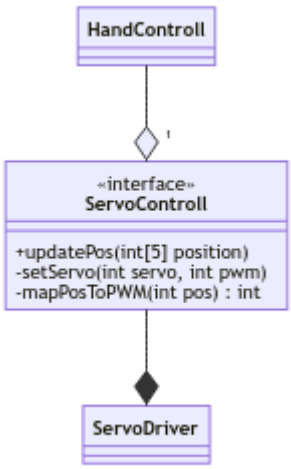


5.5 Klassen

5.5.1 Kontroll-Software



5.5.2 Arduino



6 Implementierung

6.1 Python Project Tree

```
handcontrol # complete Package
  ├── __init__.py
  ├── __main__.py # Entrypoint
  └── src
      ├── communication_framework # Implementation für Back-End
      │   ├── CallableDict.py
      │   ├── Comframe.py # Implementation für Comframe
      │   ├── hand_object.py # Implementation für Hand-Modell
      │   ├── __init__.py
      │   └── positions.py # Spezifikation der vorgefertigten
      │                    # Positionen und Animationen
      ├── demo.py # Demo script für Demonstrationen
      ├── __init__.py
      ├── ui_App.py # Definition des TUI
      └── ui_widgets.py # Definition der Widgets für TUI
```

6.2 Implementation asynchroner Kommunikation im Communication-Framework

Das COM hat einen internen Buffer welcher als Speicher für die abzuspielende Animation fungiert. Mit den Methoden `queue_position` und `queue_clear` kann auf diesen Buffer geschrieben werden.

Eine worker-Methode wird definiert, welche den internen Buffer abarbeitet und die Positionen, mit einem entsprechenden Delay zwischen den Nachrichten, versendet. Zugleich wird innerhalb der Worker-Methode eine Funktion aufgerufen, welche die einkommenden Pakete verarbeitet. Diese Worker-Methode arbeitet auf einem separaten Daemon-Thread, welcher bei Objekt-Erstellung gestartet wird.

Für die eigentliche Kommunikation ist die Library SerialTransfer zuständig. Der Worker ruft lediglich Funktionen aus dieser Library auf, um Positionen zu versenden.

6.3 SerialTransfer Callback Funktionen

Alle Pakete haben eine PaketID. Mit Hilfe von SerialTransfer ist es möglich für eine PaketID eine Callback-Funktion oder Methode aufzurufen.

In der Python Implementation wird bei dem Empfang eines Positions-Updates das interne Hand-Modell geupdated.

In der Arduino Implementation wird bei dem Empfang eines Positions-Updates eine Positions-Bestätigung zurück gesendet, sowie die Servos auf die richtigen Positionen ausgerichtet.