|  |
| --- |
| Team MYO |
| Projektabschluss |
|  |

|  |
| --- |
| Tabea Kiupel;Alexander Küppers  24.04.2015 |

Inhaltsverzeichnis

[Abbildungsverzeichnis 3](#_Toc417740039)

[1. Einleitung 4](#_Toc417740040)

[1.1. Motivation 4](#_Toc417740041)

[2. Anforderungen 5](#_Toc417740042)

[2.1. Funktionale Anforderungen 5](#_Toc417740043)

[2.2. Nicht-Funktionale Anforderungen 6](#_Toc417740044)

[3. MYO-Script-Control 7](#_Toc417740045)

[3.1. System 7](#_Toc417740046)

[3.1.1. Hauptmenü 7](#_Toc417740047)

[3.1.2. Gesten 7](#_Toc417740048)

[3.1.3. Skripte 8](#_Toc417740049)

[3.2. Architektur 9](#_Toc417740050)

[3.2.1. Activities 9](#_Toc417740051)

[3.2.2. GestureRecording 10](#_Toc417740052)

[3.2.3. ListManagement 10](#_Toc417740053)

[3.2.4. ScriptExecution 11](#_Toc417740054)

[3.3. Schnittstellen 12](#_Toc417740055)

[3.3.1. ListenerTarget 12](#_Toc417740056)

[4. Bewertung des Systems 13](#_Toc417740057)

[5. Projektablauf und –team 14](#_Toc417740058)

[5.1. Vorgehensmodell 14](#_Toc417740059)

[5.2. Projektablauf 14](#_Toc417740060)

[5.3. Team 14](#_Toc417740061)

[6. Fazit und Ausblick 15](#_Toc417740062)

[7. Anhang 16](#_Toc417740063)

# Abbildungsverzeichnis

[Abbildung 1: Ablauf der Statusanzeige 8](file:///C:\Users\Tabea\Documents\Repositories\TINF13AIBI-MYO\Documents\TINF13AIBI%20Team%20MYO%20Projektabschluss%20v0.0.docx#_Toc417739804)

[Abbildung 2: Workflow bei den Gesten 9](#_Toc417739805)

[Abbildung 3: Workflow bei den Skripten 9](#_Toc417739806)

[Abbildung 4: Activities Modul 10](file:///C:\Users\Tabea\Documents\Repositories\TINF13AIBI-MYO\Documents\TINF13AIBI%20Team%20MYO%20Projektabschluss%20v0.0.docx#_Toc417739807)

[Abbildung 5: GestureRecording Modul 11](file:///C:\Users\Tabea\Documents\Repositories\TINF13AIBI-MYO\Documents\TINF13AIBI%20Team%20MYO%20Projektabschluss%20v0.0.docx#_Toc417739808)

[Abbildung 6: ListManagement Modul 12](file:///C:\Users\Tabea\Documents\Repositories\TINF13AIBI-MYO\Documents\TINF13AIBI%20Team%20MYO%20Projektabschluss%20v0.0.docx#_Toc417739809)

[Abbildung 7: ScriptExecution Modul 12](file:///C:\Users\Tabea\Documents\Repositories\TINF13AIBI-MYO\Documents\TINF13AIBI%20Team%20MYO%20Projektabschluss%20v0.0.docx#_Toc417739810)

[Abbildung 8: Architektur mit ListenerTarget 13](file:///C:\Users\Tabea\Documents\Repositories\TINF13AIBI-MYO\Documents\TINF13AIBI%20Team%20MYO%20Projektabschluss%20v0.0.docx#_Toc417739811)

[Abbildung 9: Workflow Übersicht MYO-Script-Control 17](file:///C:\Users\Tabea\Documents\Repositories\TINF13AIBI-MYO\Documents\TINF13AIBI%20Team%20MYO%20Projektabschluss%20v0.0.docx#_Toc417739812)

[Abbildung 10: Architektur MYO-Script-Control 18](file:///C:\Users\Tabea\Documents\Repositories\TINF13AIBI-MYO\Documents\TINF13AIBI%20Team%20MYO%20Projektabschluss%20v0.0.docx#_Toc417739813)

# Einleitung

## Motivation

Die Idee des Projektes war es eine MYO-Entwicklungsumgebung zu schaffen, welche sich ohne tiefgreifende Kenntnisse bedienen lässt und die individuelle Nutzung des MYOs erleichtert. Ein weiterer Aspekt ist die freie Konfiguration des Systems, welche ohne Hindernisse konfiguriert werden kann.

Als Ergebnis der leichten Einstellungen eines solchen Systems sollte eine mit Hilfe von Gesten gesteuerte Anwendung sein, welche beliebige Soft- und Hardwaresysteme ansteuern kann. Damit geht auch die Funktion einher andere Geräte zu bedienen. Bei der Bedienung liegt der Kern in der Idee, dass das Endgerät, wie zum Beispiel Wearables, Smartphones oder externe Geräte, ohne jegliche Berührung und ohne Sprachbefehle gesteuert werden kann.

Den Nutzen, den wir mit diesem Projekt erreichen wollen ist, dass MYO-Entwickler einen deutlich erleichterten Zugriff auf MYO-Gesten haben. Das MYO soll zudem als Steuerungseinheit in bestehenden skriptgesteuerten Systemen eingebunden sein.

Die ganze Idee kommt von dem Grundgedanken einer Smart Home Automation, dass über ein System verfügt, welches als zentrale Anlaufstelle für die Steuerung vieler Geräte dient.

Da die Kombination von Gesten und Skripten einen vielfältigen Einsatzbereich versprochen hat, haben wir uns für den Weg entschieden dies als Grundlage unseres Systems zu nutzen.

Als exklusives Highlight wäre hier die Möglichkeit einer Fernsteuerung von Geräten, wie zum Beispiel ein RC Auto.

# Anforderungen

Es folgt eine Auflistung der an die Anwendung gestellten Anforderungen. Dabei wird zwischen den funktionalen und nicht-funktionalen Anforderungen unterschieden.

## Funktionale Anforderungen

* F – 10 – Smartphone- Anwendung
  + F – 10.1 – MYO
    - In der Anwendung kann eine Verbindung mit einem MYO aufgebaut werden.
  + F – 10.2 – GUI-Design
    - Zum Verwalten der Skripte und Gestenfolgen stellt die Anwendung ein GUI zur Verfügung. Sobald Skripte und Gestenfolgen konfiguriert sind können diese ausgeführt werden während die Anwendung ohne GUI im Hintergrund aktiv ist.
  + F – 10.3 – Verwendete Gesten
    - Das MYO erkennt standardmäßig 5 verschiedene Gesten. Diese sind:
      * Fist
      * Wave Right
      * Wave Left
      * Fingers Spread
      * Double Tap
    - Die Anwendung kann höchstens mit diesen Gesten umgehen.
  + F – 10.4 – Gestenfolgen
    - Die Anwendung kann mit dem MYO ausgeführte Gesten erkennen, aufnehmen und abspeichern.
    - Weiterhin können gespeicherte Gesten ausgeführt werden. Eine erfolgreiche Ausführung einer Geste hat eine Ausführung des ihr zugeordneten Skriptes zur Folge.
    - Verwaltung von Gesten: Aufnehmen, Speichern, Löschen, Skript/Aktion zuordnen
  + F – 10.5 – Aufnahme/Ausführung von Gestenkombinationen
    - Zur Aufnahme neuer Gesten werden die vorhanden Gesten mit den Positionsdaten des MYOs zu neuen Gesten kombiniert.
    - Eine Gestenkombination beginnt mit einer Unlock-Geste, welche die aktuelle Position des MYOs als Ausgangsstellung setzt. Darauf folgt eine Anzahl von Gesten, wobei jede Geste in einem bestimmten Winkel ausgeführt wird. Der Winkel drückt die aktuelle Drehung des MYOs relativ zur Ausgangsstellung aus.
    - Die Anzahl der möglichen Positionen ist dabei auf neun beschränkt, diese sind:
      * + links
        + rechts
        + oben
        + unten
        + linksoben
        + rechtsoben
        + linksunten
        + rechtsunten
        + Ausgangsstellung
    - Die einzelnen Punkte können dabei innerhalb einer Kombination mehrfach genutzt werden.
    - Der Nutzer kann eine Liste aller gespeicherten Gestenkombination aufrufen, aus der er sich zu jeder jeweils die Abfolge der Gesten anzeigen lassen kann.
  + F – 10.6 – Skripte
    - Die Anwendung verwaltet verschiedene Skripte der Sprache “Python”. Der Nutzer hat die Möglichkeit sich diese in einer Liste anzeigen zu lassen und dann ein einzelnes Skript zu löschen oder ihm eine eingespeicherte Gestenfolge zuzuordnen.
    - Führt der Nutzer nun die zugeordnete Gestenfolge erneut aus, so wird das Skript gestartet.
* F – 20 – Vuzix- Anwendung
  + F – 20.1 – Vuzix
    - Die Anwendung ermöglicht auch das Ausführen von Skripten auf einer Vuzix.
    - Hierzu wird eine App auf der Vuzix installiert, die über Bluetooth mit der laufenden Smartphone-Anwendung kommuniziert.
  + F – 20.1 – Smartphone
    - Die Gestenerkennung und Konfiguration erfolgt auf der Smartphone-App, jedoch werden die Skripte auf der Vuzix ausgeführt.

## Nicht-Funktionale Anforderungen

* N – 10 – Performance
  + N – 10.1 – Reaktionszeit
    - Die Anwendung startet das einer Gestenfolge zugeordnete Skript in weniger als einer Sekunde, nachdem die Geste vollständig ausgeführt wurde.
* N – 20 – Installation
  + N – 20.1 – Installation Smartphone- Anwendung
    - Die Anwendung lässt sich wie eine gewöhnliche App manuell installieren und ist nicht über einen App-Store erreichbar.
  + N – 20.1 – Installation Vuzix- Anwendung
    - Die Anwendung lässt sich wie eine gewöhnliche App manuell installieren und ist nicht über einen App-Store erreichbar.

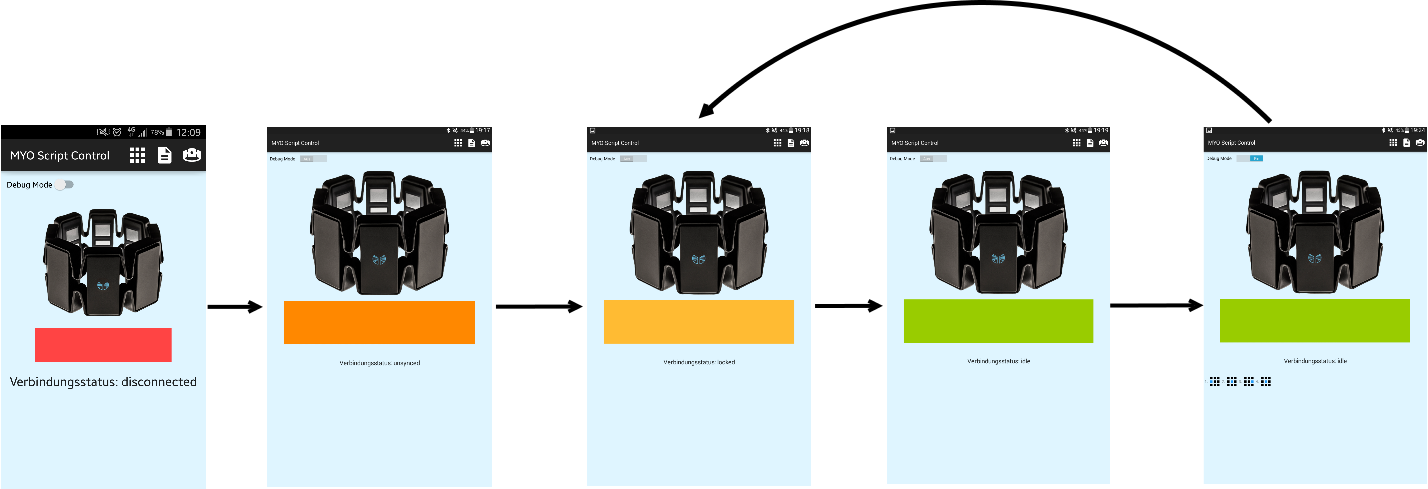
# MYO-Script-Control

## System

In diesem Abschnitt wird eine Übersicht über die Funktionsweise der Anwendung gegeben.

### Hauptmenü

Die Startseite der Anwendung MYO-Script-Control hat mehrere Funktionen.

Zum einen dient sie zur Navigation zu den verschiedenen Unterseiten. Auf der einen Seite gelangt man zur Übersicht der gespeicherten Gestenelemente. Auf der anderen Seite gelangt man zu Übersicht der gespeicherten Skriptelemente.

Ebenso dient die Startseite zur Anzeige des Verbindungs- bzw. Synchronisationsstatus mit dem MYO. Entsprechend des Status ist ein Ausführen von hinterlegten Gesten/Skripten möglich.

Abbildung 1: Ablauf der Statusanzeige

Anfangs befindet sich man in dem Verbindungsstatus „disconnected“. Von diesem gelangt man in den Status „unsynced“, sobald die Anwendung mit einem MYO verbunden wurde. In den Status „locked“ kommt man durch Ausführen der Synced-Geste. Nach Ausführen der Unlock-Geste folgt der Status „idle“. Die Anwendung ist nun bereit eine Geste durch das MYO zu Erkennen. Nach der Bestätigungs-Geste wird die aufgenommene Geste überprüft und man gelangt wieder in den Status „locked“.

### Gesten

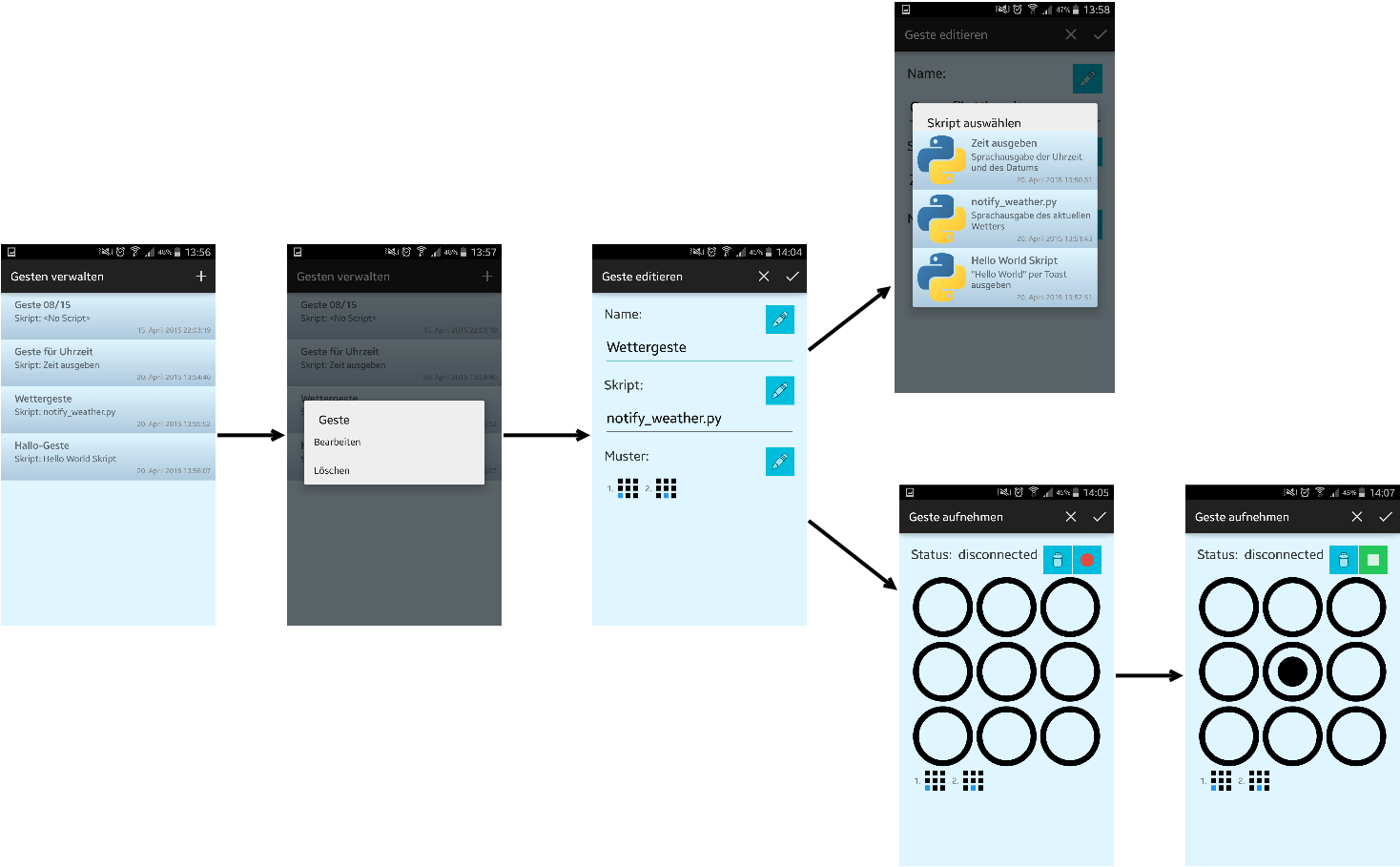
Die abgespeicherten Gesten werden aufgelistet. Von dort ist es möglich ein neues Element hinzuzufügen oder ein bestehendes Element zu bearbeiten oder zu löschen. Bei der Bearbeitung eines Gestenelements ist es möglich dieses zu benennen. Ebenso kann dem Element einerseits ein abgespeichertes Skriptelement zugeordnet werden und andererseits eine Geste aufgenommen werden.

Abbildung 2: Workflow bei den Gesten

### Skripte

Die abgespeicherten Skriptelemente werden aufgelistet. Von dort ist es möglich ein neues Element hinzuzufügen oder ein bestehendes Element zu bearbeiten oder zu löschen. Bei der Bearbeitung eines Skriptelements ist es möglich dieses zu benennen und eine Beschreibung hinzuzufügen. Ebenso kann dem Element ein Skript zugeordnet werden.

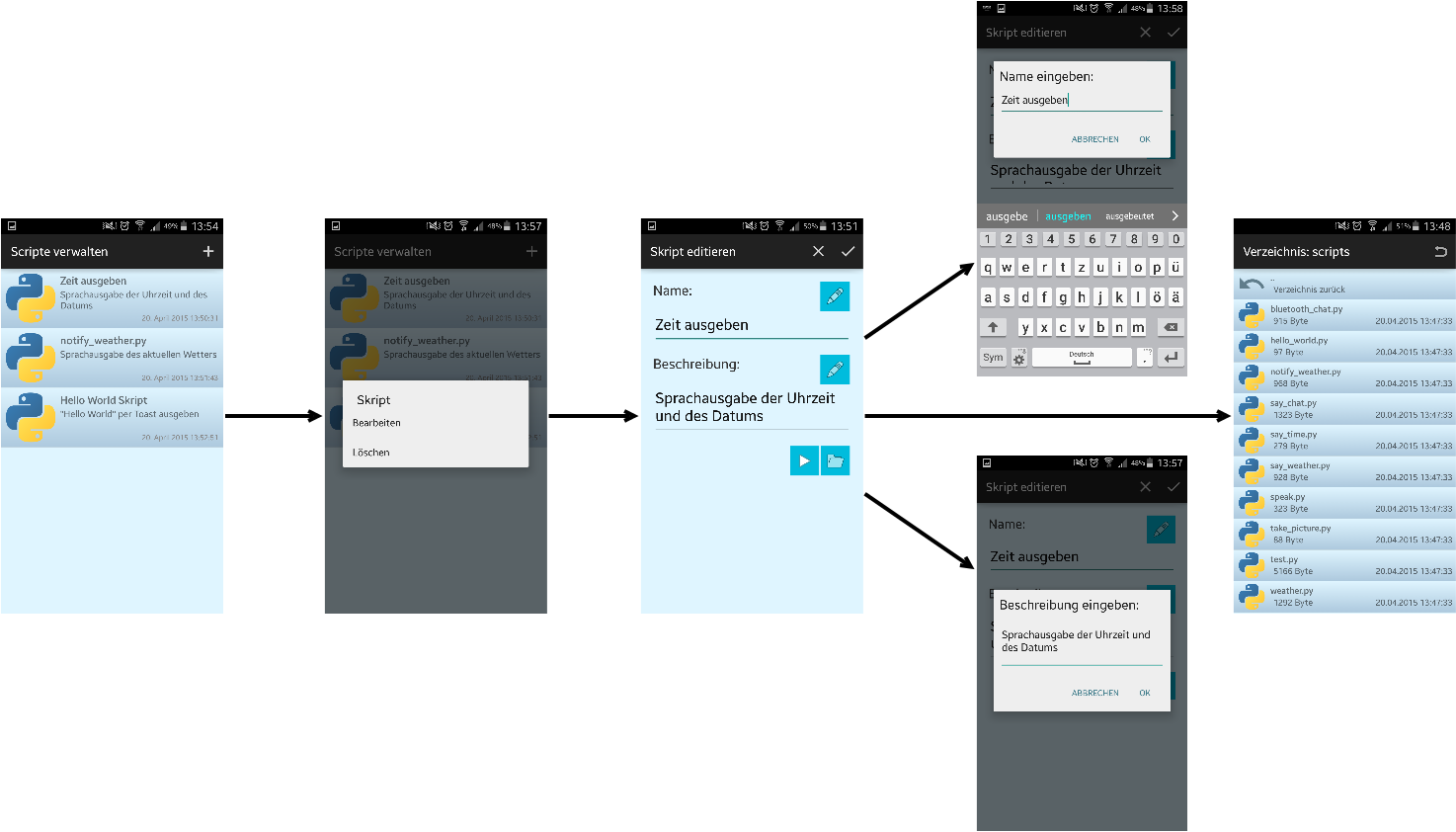


Abbildung 3: Workflow bei den Skripten

Die komplette Übersicht der Workflows befindet sich im Anhang (siehe Abbildung 9).

## Architektur

Im Folgenden wird eine Übersicht über die Architektur der Anwendung gegeben. Die einzelnen Module werden näher beschrieben.

### Activities

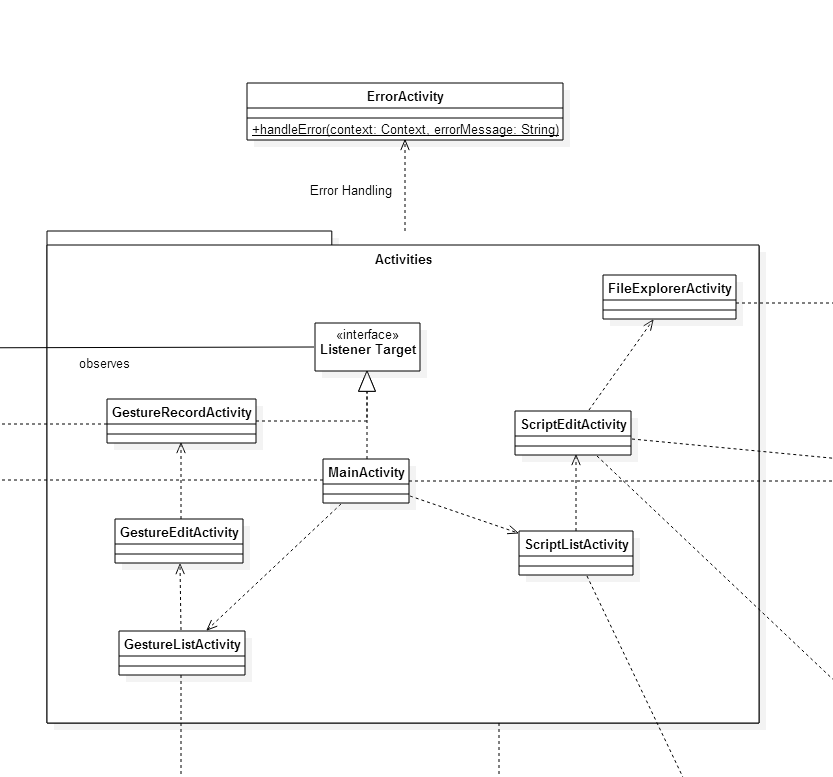
Das Activities Modul ist für die Repräsentation der GUI durch die verschiedenen Activities zuständig. Die zentrale Activity ist die MainActivity. Von ihr aus wird zu den anderen Activities navigiert. Dabei wird dem bereits dargestellten Workflow gefolgt. Die GestureListActivity und ScriptListActivity verwenden die Module GestureManagement und ScriptManagement aus dem Modul ListManagement.

Abbildung 4: Activities Modul

Die GestureEditActivity ermöglicht das Bearbeiten eines Gestenelements. Von dieser Activity gelangt man zur GestureRecordActivity. Dort kann der Nutzer eine Geste aufnehmen. Dafür greift die Activity auf das Modul GestureRecording zu.

Die ScriptEditActivity ermöglicht das Bearbeiten eines Skriptelements.

Von ihr gelangt man zur FileExploreActivity. Diese Activity ermöglicht den Import eines Skripts aus dem Dateisystem des Endgeräts, auf welchem die Anwendung installiert ist. Dafür greift sie auf das Modul FileManagement zu.

Weiterhin kann man in der ScriptEditActivity ein importiertes Skript ausführen. Dafür wird auf das Modul ScriptExecution zugegriffen.

Da in der MainActivity sowohl Gesten und Skripte ausgeführt werden, wird auf die Module GestureRecording und ScriptExecution zugegriffen.

### GestureRecording

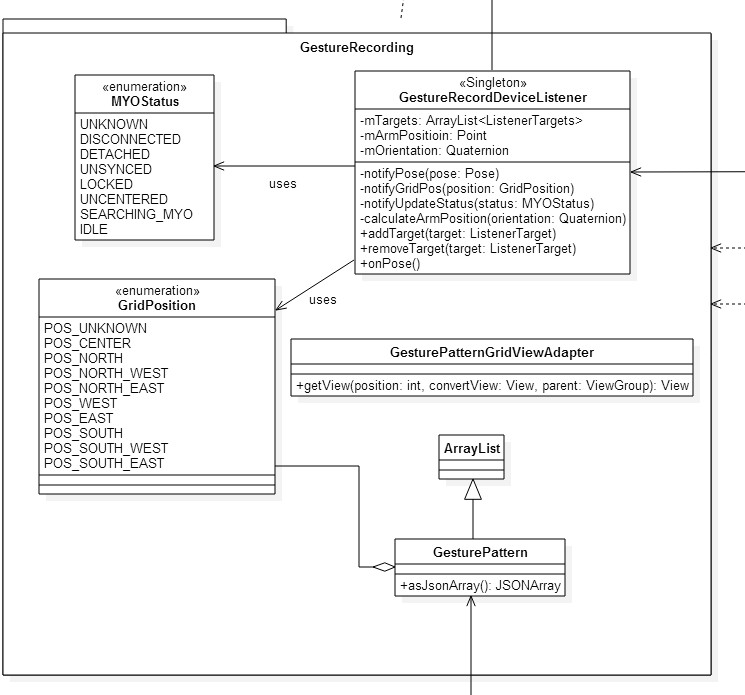
Das GestureRecording Modul ist für die Erkennung der MYO-Gesten zuständig. Die zentrale Klasse des Moduls ist der GestureRecordDeviceListener. Dieser erbt von dem AbstractDeviceListener des MYO-SDKs und ist als Singleton implementiert. Der GestureRecordDeviceListener erhält Positions- und Gestendaten des MYOs. Über das Interface ListenerTarget kann sich eine Klasse auf dem Listener registrieren und diese Daten erhalten. Zur Speicherung des MYO-Status verwendet der Listener das Enum RecordActivityStatus. Die aktuelle Position des MYOs wird über das Enum GridPosition bekanntgegeben. Für die aktuelle Geste wird das Enum Pose verwendet, welches Teil des MYO SDKs ist.

Abbildung 5: GestureRecording Modul

Das GestureRecording Modul enthält weiterhin noch die Klasse GesturePattern, welche im Wesentlichen eine ArrayListe von GridPositions ist. GesturePattern dient somit zur Speicherung der Gestenabfolge und bietet außerdem die Möglichkeit diese in das JSON-Format zu konvertieren.

Der GesturePatternGridViewAdapter stellt eine Geste in der MainActivity und der GestureRecordActivity dar.

### ListManagement

Das ListManagement Modul ist für die Speicherung von Gesten und deren zugehöriger Skripte, sowohl auf Dateiebene, als auch in der Benutzeroberfläche, zuständig. Es ist unterteilt in die Module GestureManagement, ScriptManagement und FileManagement. Die Klassen GestureScriptManager und FileManager regeln die Kommunikation mit den Activities.

Der GestureScriptManager ist eine Singleton-Klasse, die Zugriff auf die Module GestureManagement und ScriptManagement, über die Member GestureList und ScriptList ermöglicht.

GestureList ist eine Arrayliste von GestureItems des Moduls GestureManagement. Ein GestureItem repräsentiert eine MYO-Geste.

ScriptList ist eine Arrayliste von ScriptItems des Moduls ScriptManagement. Ein SkriptItem repräsentiert ein Skript.

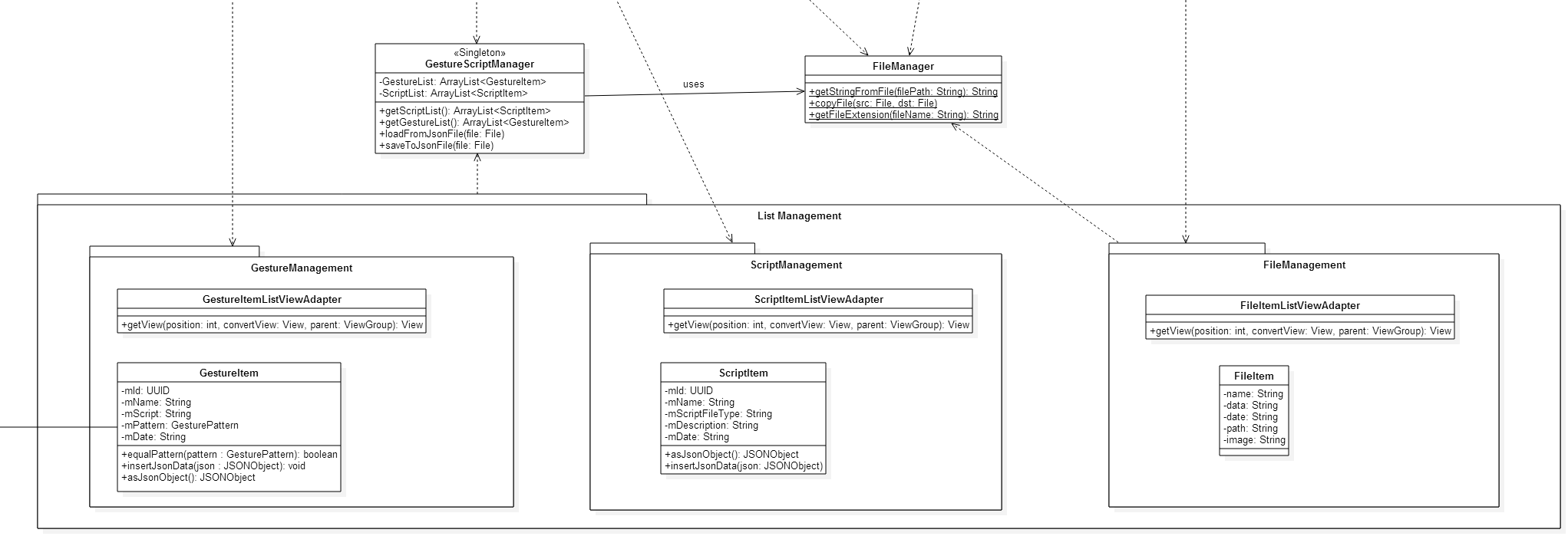


Abbildung 6: ListManagement Modul

Der FileManager ist für die Speicherung von Gesten und Skripten auf Dateiebene zuständig und sorgt für Speichern und Laden der Konfigurationsdatei.

Die Klasse FileItem wird beim Skriptimport in der FileExplorerActivity verwendet und repräsentiert eine Datei.

Weiterhin ist in Gesture-, Script- und Filemanagement eine ListView-Adapterklasse enthalten, die für die Listendarstellung der Items in den entsprechenden Activities verantwortlich ist.

### ScriptExecution

Das Modul ScriptExecution ist für die Ausführung der Skripte zuständig. Dies geschieht über die externe Anwendung SL4A, die von der Klasse SL4AManager gestartet wird.

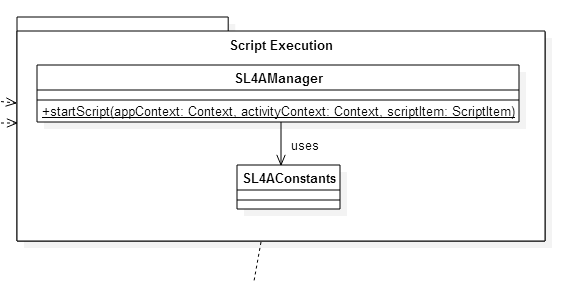
Dabei erstellt die Methode startScript() einen Intent, dem als Parameter das auszuführende Skript und eine spezifische SL4A-Konstante, die angibt wie das Skript ausgeführt werden soll, hinzugefügt wird. Über den Intent wird anschließend SL4A gestartet.

Abbildung 7: ScriptExecution Modul

## Schnittstellen

### ListenerTarget

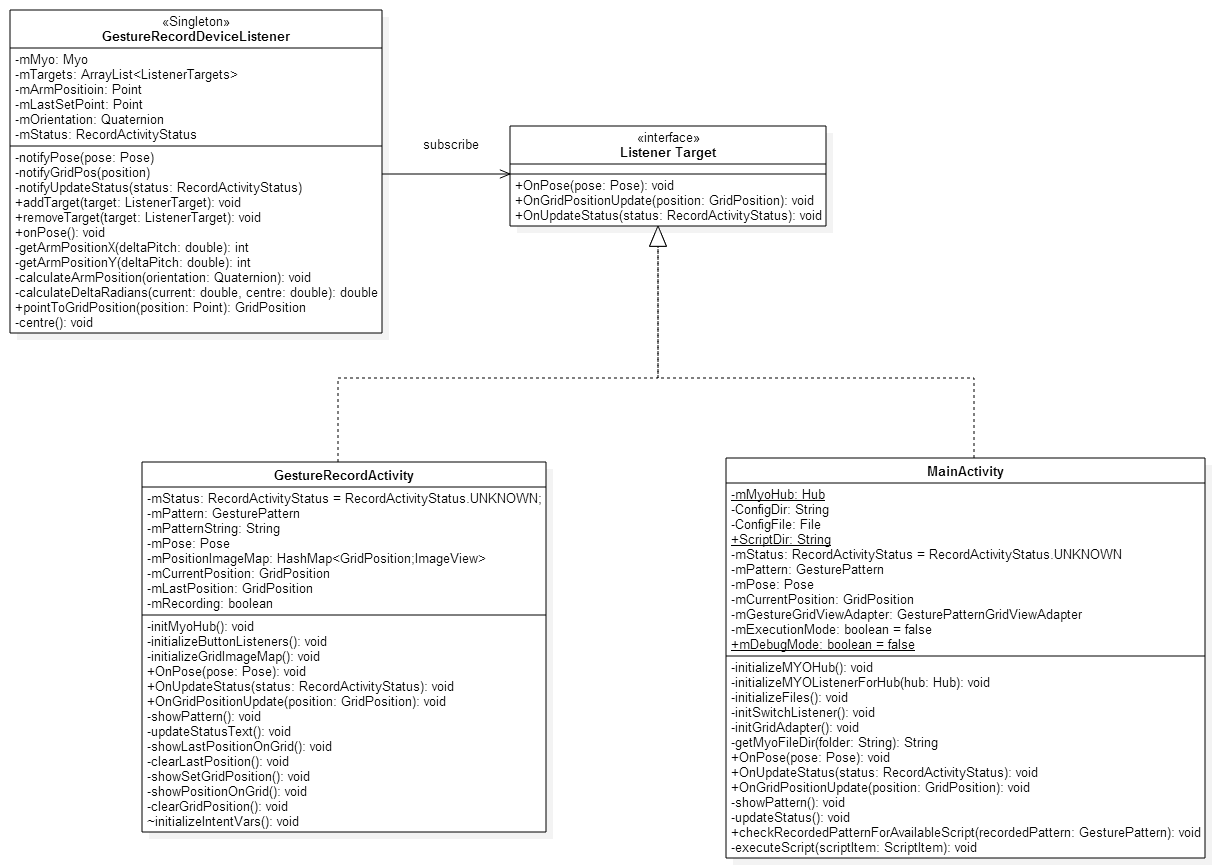
Das ListenerTarget beschreibt die Schnittstelle zwischen der GestureRecordActivity bzw. der MainActivity und dem GestureRecordDeviceListener. Sie definiert die Methoden OnPose(), OnGridPositionUpdate() und OnUpdateStatus().

Abbildung 8: Architektur mit ListenerTarget

In der Methode OnPose() wird in den erbenden Klassen die Funktionalität implementiert, welche für die Erkennung der Gesten zuständig ist.

In der Methode OnGridPositionUpdate() wird beschrieben, was passiert, wenn sich die Position des MYOs in dem definierten 3x3-Feld ändert.

In der Methode OnUpdateStatus() wird beschrieben, was eine Änderung des Status des MYOs in der entsprechenden Activity bewirkt.

# Bewertung des Systems

# Projektablauf und –team

## Vorgehensmodell

Für das Projekt wurde das agile Vorgehensmodell Scrum gewählt. Hierbei wurden einwöchige Sprints angesetzt, welche von Dienstags bis Dienstags gingen. Das Projekt hat insgesamt 10 Sprints beinhaltet, in denen in etwa 575 Arbeitsstunden untergebracht waren beziehungsweise geleistet wurden.

In den einwöchigen Sprints wurden folgende verschiedene Meetings abgehalten, welche auch in den für alle Teammitglieder zugänglichen Google-Kalender eingetragen wurden:

Als erster Punkt wäre hier das Estimation-Meeting zu nennen. Dieses Meeting war variabel angesetzt aufgrund dessen, dass nicht immer neue Arbeitspakete definiert worden, die mit in das Backlog gekommen waren. Falls ein neues Backlog Item eingetragen wurde und ein Estimation-Meeting angesetzt war, so wurde hier eine grobe Abschätzung des gesamten Teams gegeben, welchen Workload dieses Backlog Item hat. Die Abschätzung erfolgte hier in Personen-Tagen mit jeweils 8 Stunden.

Als nächstes Meeting ist hier das Planungsmeeting zu nennen, welches immer am ersten Dienstag, also beim Beginn eines jeden Sprints, stattgefunden hat. Als grober Rahmen wurde hier eine Zeit von 60 Minuten angesetzt, um den Backlog Items Tasks zuzuweisen und diese abschließend mit einem Workload zu versehen, welche wieder in der Gruppe abgeschätzt wurde.

Anschließend an das Planungsmeeting hat ein Kick-Off stattgefunden, bei dem in circa 10 Minuten die initiale Aufgabenverteilung für den Sprint festgelegt wurde.

Um immer teamgerecht den aktuellen Stand des Sprints zu wissen, fanden dreimal wöchentlich, also am Mittwoch, Freitag und am Montag, Daily-Scrums statt. Hierbei wurden in circa 15 Minuten beschrieben, was jedes einzelne Team-Mitglied bis jetzt, seit dem letzten Meeting, getan hat, auf welche Probleme er gestoßen ist und was er bis zum nächsten Meeting machen wird.

Am Ende eines jeden Sprints, also am 2. Dienstag, findet die so genannte Retrospektive statt, welche mit einer Dauer von etwa 30 Minuten angesetzt ist. In diesem Meeting wurde immer besprochen, was im letzten Sprint gut lief und was noch verbessert werden kann. Das Ziel des Meetings lag darin eins bis zwei konkrete Verbesserungen, die mit in den nächsten Sprint genommen werden sollen.

Im Anschluss der Retrospektive folgt das Review, welches rund 20 Minuten dauert. Die Ergebnisse, für den Sprint relevanter, Storys werden hier vorgestellt.

Das Scrum Modell wurde des Weiteren ein wenig angepasst bezüglich Nachforschung neuer Themen, dem so genannten Research. Hierbei wurden nötige Forschungen betrieben, um ein Überblick über das jeweilige Thema zu bekommen und das Team darüber zu informieren, um daran angelehnte Tasks besser einschätzen zu können, zum einen hinsichtlich der dafür benötigten Zeit und zum Anderen bezüglich was möglich ist für das Projekt um zu setzen.

## Projektablauf

Der Projektablauf wurde gegliedert durch die vom Kunden vorgegebenen Meilensteine.

Die Meilensteine haben sich wie folgt zusammengesetzt:

1. 24.02.2015 – Erstellung eines Terminplans mit Arbeitspaketen und Aufwandsabschätzungen für das Projekt
2. 12.03.2015 – Das Pflichtenheft muss fertiggestellt sein und wird dem Kunden präsentiert
3. 23.03.2015 – Das Inspektions- und Testkonzept wird vorgestellt und im Projekt umgesetzt
4. 13.04.2015 – Die Software-Architektur steht fest, ist digitalisiert und wird vorgestellt
5. 27.04.2015 – Die Abschlusspräsentation des fertigen Produktes

## Team

Unser Team hat sich zusammengesetzt aus einem Scrum Master (Marco Meyer) und einem Stellvertreter (Daniel Thomalla). Dieser war für die Organisation und Durchführung der Retrospektive zuständig. Des Weiteren war es seine Aufgabe stets die Einhaltung der Teamregeln zu kontrollieren und gegebenenfalls auf die Einhaltung hin zu weisen.

Ein weiterer Bestandteil unseres Teams war der Product Owner (Simon Diggelmann) und seine Stellvertreterin (Tabea Kiupel). Hier lautete die Aufgabe, dass stets eine Backlog-Wartung durchgeführt wird. Außerdem sollte das Planungsmeeting vorbereitet werden zusammen mit der Sprintplanung. Schlussendlich kam noch die wichtige Aufgabe hinzu die einzelnen Storys ab zu nehmen, insofern diese den Anforderungen genügten und den Kunden zufrieden stellen werden.

Als weitere Rolle im Team waren die Developer (Simon Diggelmann, Felix Helfrich, Tabea Kiupel, Alexander Küppers, Marco Meyer, Daniel Thomalla). Ihre Aufgabe war zu aller erst die Entwicklung der Anwendung in Bezug auf die Vorgaben und Anforderungen. Ein weiterer, nicht zu vernachlässigender Teil, war das Testen, hier wurden Unit-Tests geschrieben und jeweils durchgeführt um nach möglichen Fehlern zu suchen.

# Fazit und Ausblick

Als Fazit über das Vorgehensmodell Scrum, so, wie es in diesem Projekt angewendet wurde, kann gesagt werden, dass es effektiv war, jedoch manchmal eine zeitliche Fehleinschätzung vorlag.

Anzusprechen war hier, dass aufgrund von Meilensteine in anderen Projekten, die wöchentliche Arbeitszeit sehr variiert hat und das aufgrund eines Krankheitsfalles eines Teammitglieds, welche nicht voraussehbar war und teilweise punktuell wieder kam, eine zeitliche Fehlplanung vorlag. Besonders positiv ist jedoch an zu merken, dass eine sehr gute Stimmung im Team geherrscht hat, vom ersten bis zum letzten Tag. Des Weiteren konnte aufgrund von geregelten Kontaktlisten mit Kommunikationsangaben immer gut Kontakt zu anderen Team-Mitgliedern aufgenommen werden.

Das Projekt kann also aus Teamsicht als gelungen angesehen werden und in Hinsicht auf die eingehaltenen Anforderungen vom Kunden auch als akzeptabel angenommen werden.

Als Ausblick ist hier zu erwähnen, dass hier noch Verbesserungspotential liegt, bei hohem Zeitaufwand, da die Script-Execution bis jetzt über eine externe Anwendung auf dem Android-Smartphone übernommen wird. Hier könnte eine ganz eigene Skript-Application geschrieben werden, welche aber wieder ein eigenes Projekt beanspruchen würde. Als weiteren Aspekt könnte man noch die Android-App für Wearables schreiben, welche dann auf kleinere Bildschirme angepasst werden sollte. Dies wurde hier im Rahmen nicht umgesetzt, da die zur Verfügung stehende Vuzix M100 eine zu niedrige Android-Version hatte um mit dem MYO per Bluetooth zu kommunizieren. Das Design der App könnte noch an die Android Developer Guideline für Android 5 angepasst werden um auch grafisch auf dem aktuellsten Stand zu sein.

# C:\Users\Tabea\Documents\Repositories\TINF13AIBI-MYO\Pictures&Flows\Übersicht_komplett.pngAnhang

Abbildung 9: Workflow Übersicht MYO-Script-Control

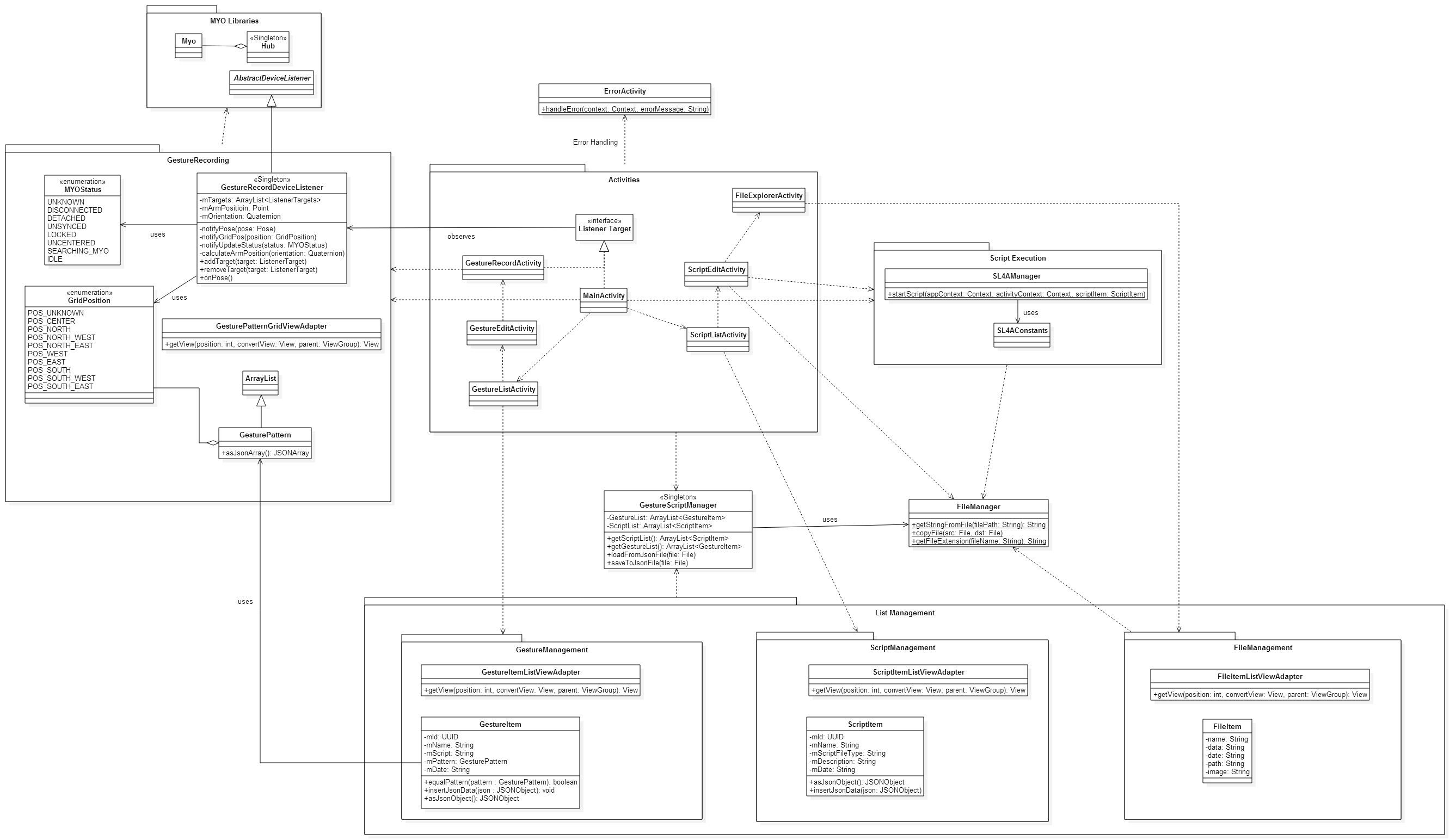


Abbildung 10: Architektur MYO-Script-Control