Team MYO

Projektabschluss

Tabea Kiupel;Alexander Küppers;Marco Meyer
27.04.2015

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1. Motivation

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

Das Ergebnis der leichten Einstellungen eines solchen Systems, sollte eine mit Hilfe von Gesten gesteuerte Anwendung sein. Diese kann beliebige Soft- und Hardwaresysteme ansteuern. Damit geht auch die Funktion einher, andere Geräte bedienen zu können. Bei der Bedienung liegt der Schwerpunkt darauf, dass das Endgerät, wie z.B. Wearables, Smartphones oder externe Geräte, ohne jegliche Berührung und 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 dienen.

Die ganze Idee kommt von dem Grundgedanken einer Smart Home Automation, welches 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 eines RC Auto.

2. Anforderungen

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

2.1. 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.

2.2. 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.

3. MYO-Script-Control

3.1. System

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

3.1.1. Hauptmenü

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

Zum einen dient sie zur Navigation zu den verschiedenen Unterseiten. Weiterhin gelangt man zur Übersicht der gespeicherten Gestenelemente oder zur Übersicht der gespeicherten Skriptelemente.

Ebenso dient die Startseite zur Anzeige des Verbindungs- bzw. Synchronisationsstatus mit dem MYO.

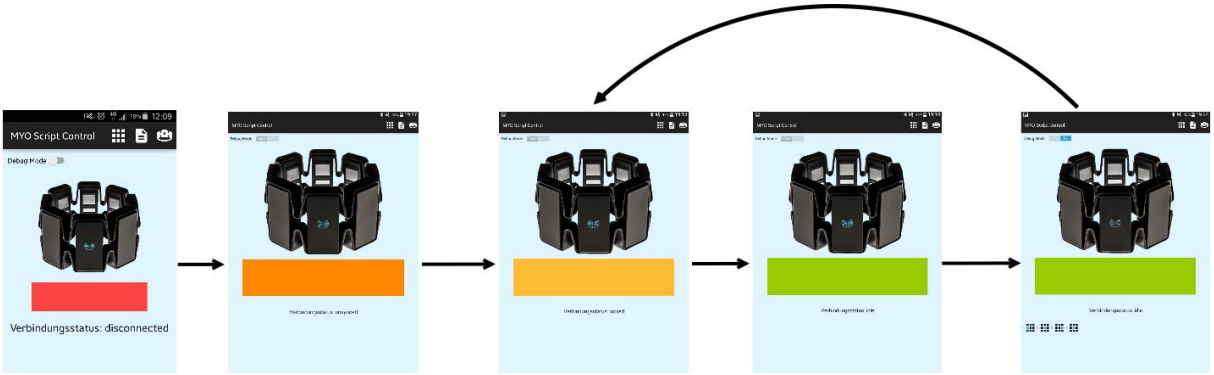

Abbildung 1: Ablauf der Statusanzeige

Entsprechend des Status ist ein Ausführen von hinterlegten Gesten/Skripten möglich.

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“.

3.1.2. 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.

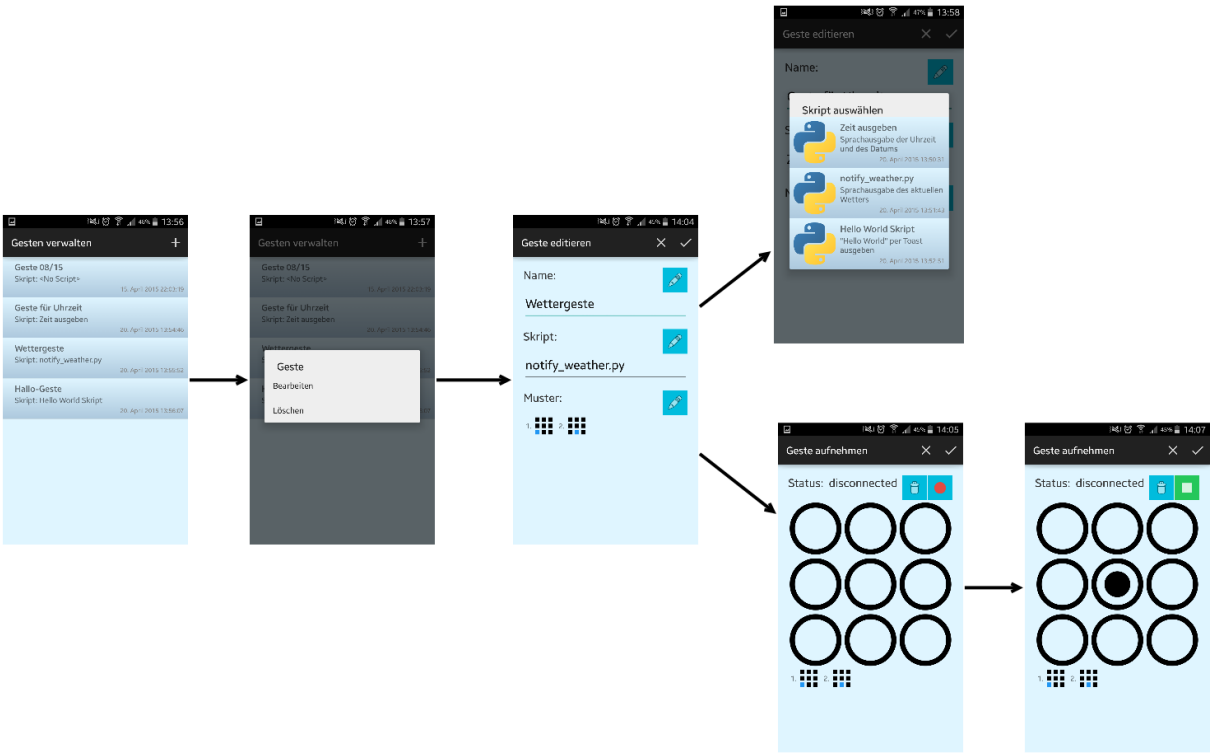

Abbildung 2: Workflow bei den Gesten

3.1.3. 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.

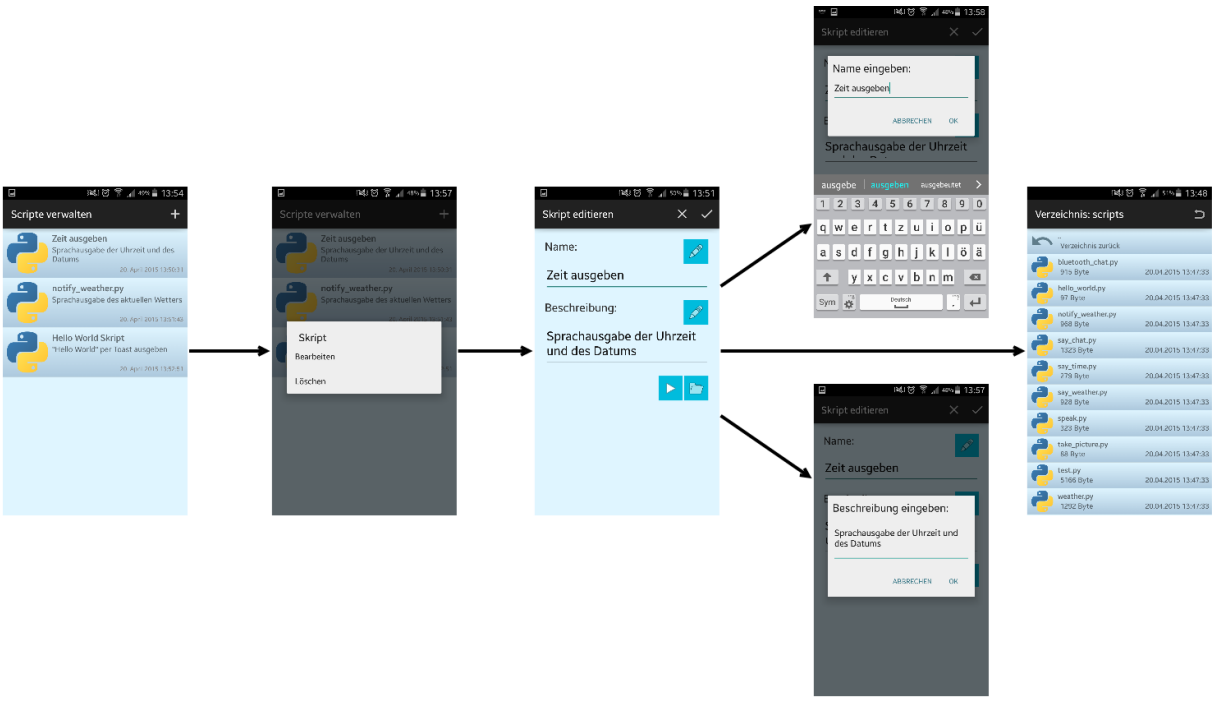

Abbildung 3: Workflow bei den Skripten

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

3.2. Architektur

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

3.2.1. Activities

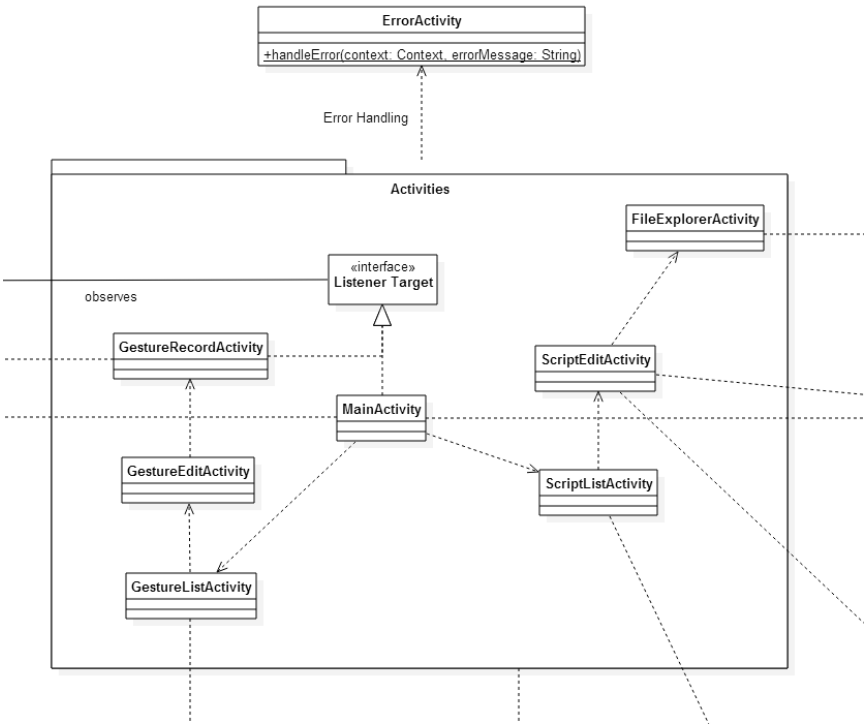

Abbildung 4: Activities Modul

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.

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.

3.2.2. GestureRecording

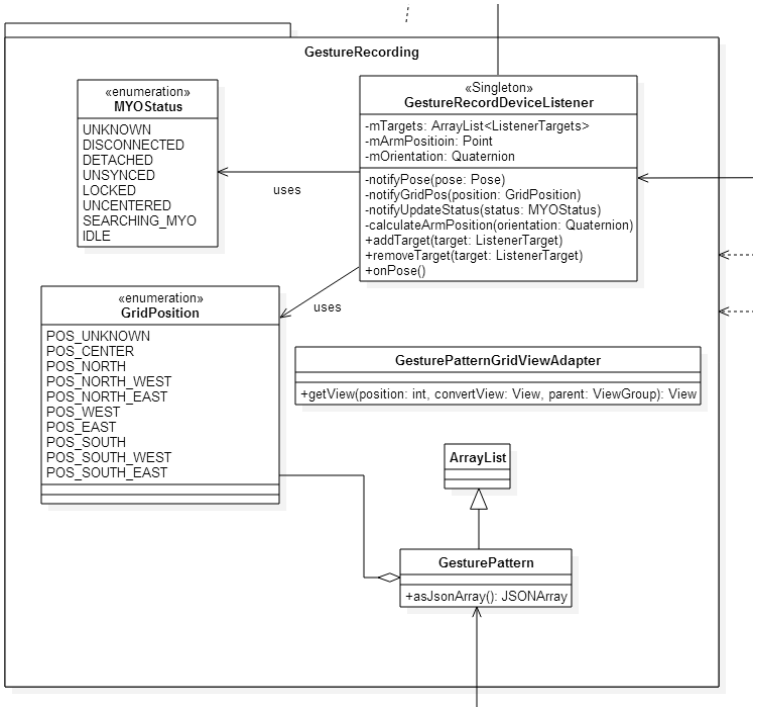

Abbildung 5: GestureRecording Modul

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.

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.

3.2.3. 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.

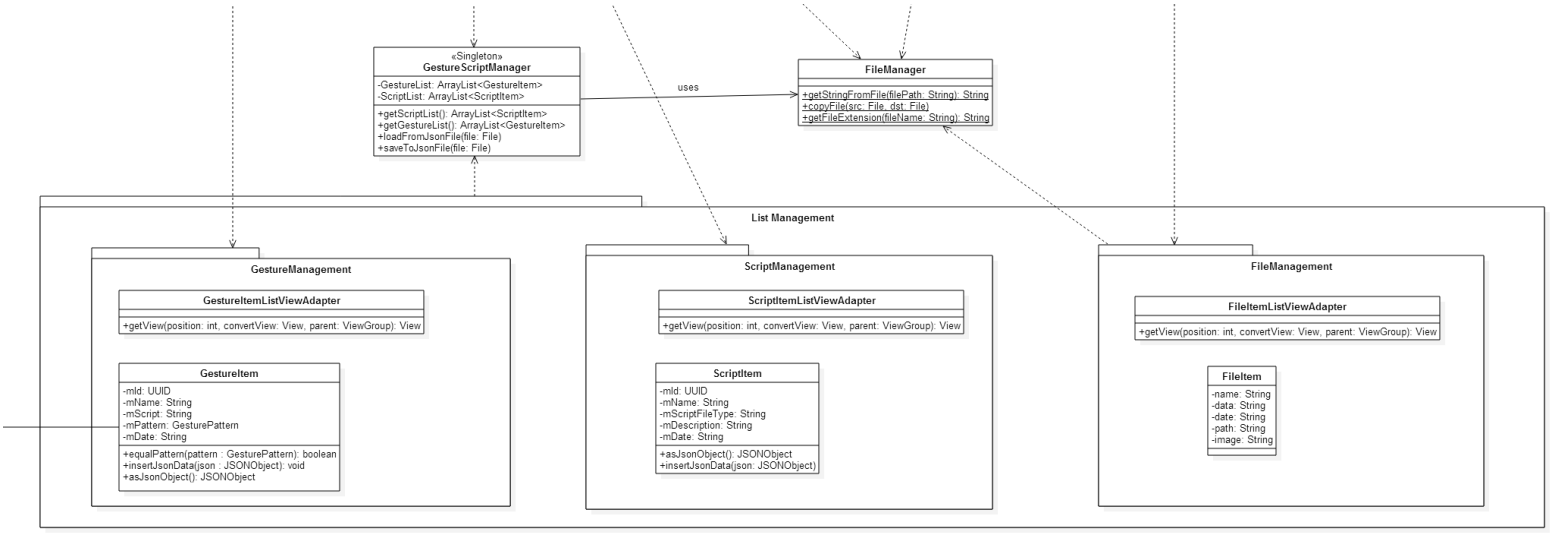

Abbildung 6: ListManagement Modul

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.

3.2.4. 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.

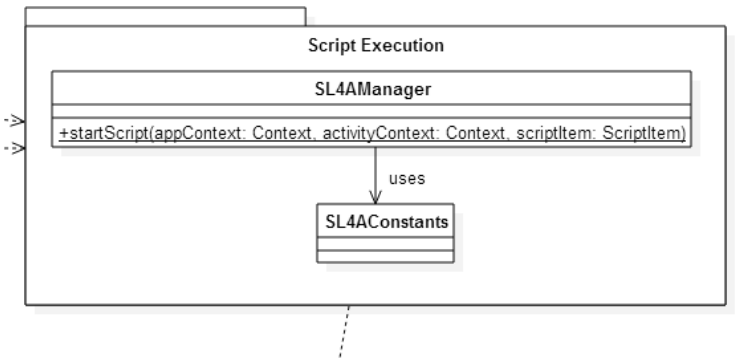

Abbildung 7: ScriptExecution Modul

3.3. Schnittstellen

3.3.1. ListenerTarget

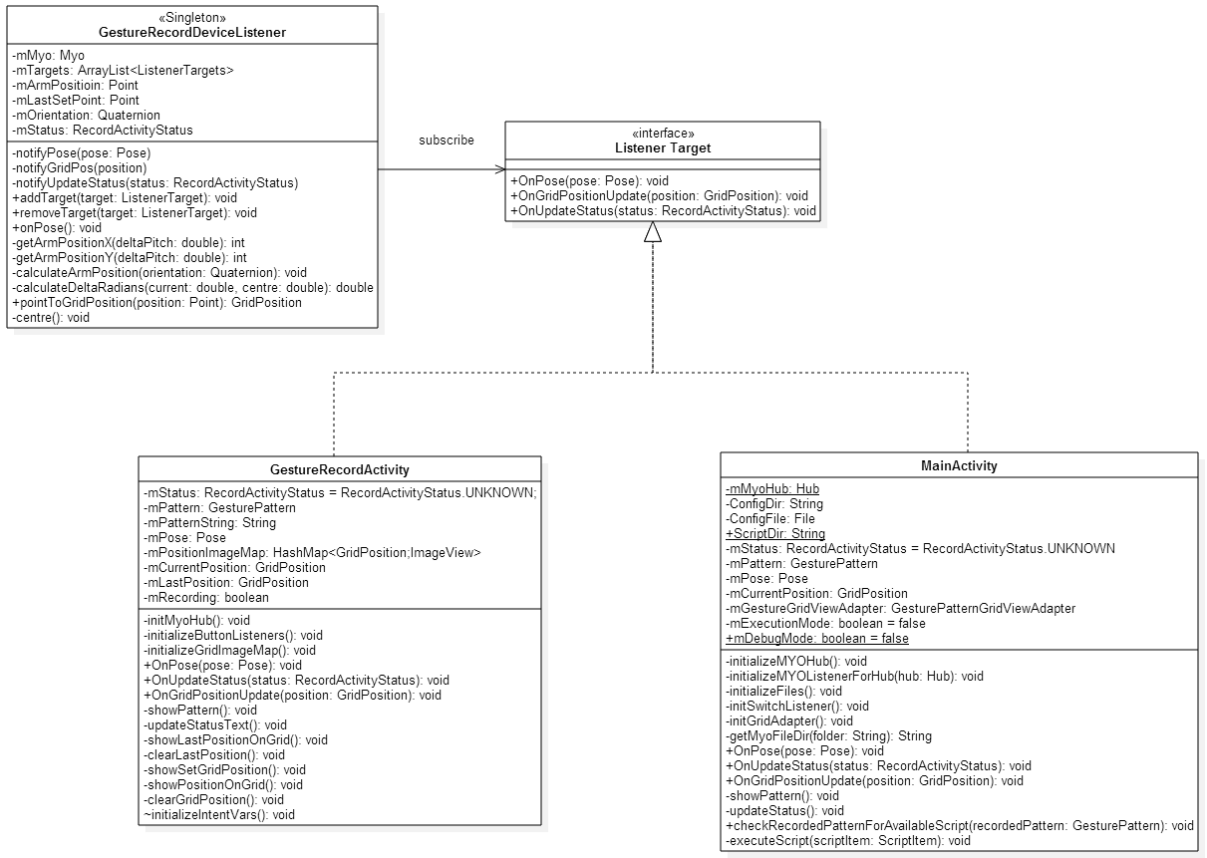

Abbildung 8: Architektur mit ListenerTarget

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

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.

4. Projektablauf und –team

4.1. 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 geleistet wurden.

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

Als erster Punkt sei hier das Estimation-Meeting zu nennen. Dieses Meeting wurde je nach Bedarf angesetzt. Dies geschah, da der Product Owner nicht über die gesamte Projektdauer neue User Stories definiert hat und ins Backlog aufgenommen hat. Falls eine neue Story eingetragen wurde und ein Estimation-Meeting angesetzt war, so wurde hier eine grobe Abschätzung des gesamten Teams gegeben. Es wurde abgeschätzt, welchen Workload diese Story hat. Die Abschätzung erfolgte 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. Dabei wurden jeder Story, welche zuvor in den Sprint gelegt wurde, Tasks zugewiesen. Die erzeugten Tasks wurden abschließend mit einem Workload versehen. Die Abschätzung erfolgte durch das komplette Team und in Stundenangaben.

Anschließend an das Planungsmeeting fand ein Kick-Off statt. In circa 10 Minuten wurde die initiale Aufgabenverteilung für den Sprint festgelegt.

Um immer teamgerecht den aktuellen Stand des Sprints zu wissen, fanden dreimal wöchentlich, also am Mittwoch, Freitag und am Montag, Daily-Scrums statt. Hierbei wurde in circa 15 Minuten beschrieben, was jedes einzelne Team-Mitglied seit dem letzten Meeting getan hat, auf welche Probleme es 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 besprochen, was im letzten Sprint gut lief und was noch verbessert werden kann. Das Ziel des Meetings lag darin, eine bis zwei konkrete Verbesserungen zu beschließen, welche mit in den nächsten Sprint genommen werden sollten.

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, der so genannten Research. Hierbei wurden nötige Forschungen betrieben, um ein Überblick über das jeweilige Thema zu bekommen und das Team darüber zu informieren. Die Research sollte dazu dienen, daran angelehnte Tasks später besser einschätzen zu können. Es sollte danach ein besseres Abschätzen hinsichtlich der dafür benötigten Zeit und der Umsetzungsmöglichkeiten im Rahmen des Projekts, möglich sein.

4.2. 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

4.3. Team

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

Ein weiterer Bestandteil unseres Teams war der Product Owner (Simon Diggelmann) und seine Stellvertreterin (Tabea Kiupel). Hier lautete die Aufgabe das Backlog stets aktuell zu halten und zu pflegen. Außerdem war es die Aufgabe des Produkt Owners, die Sprintplanung zu betreiben und das Planungsmeeting in diesem Zusammenhang vorzubereiten. Schlussendlich kam noch die wichtige Aufgabe hinzu die einzelnen Storys abzunehmen, insofern diese den Anforderungen genügten und den Kunden zufrieden stellen werden.

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

5. Bewertung

Im Folgenden werden einerseits die entstandene Anwendung und andererseits das genutzte Vorgehensmodell evaluiert.

5.1. Bewertung des entwickelten Systems

Die entstandene Anwendung MYO-Scrip-Control kann anhand der an sie gestellten Anforderungen bewertet werden. Ebenso wurde ein Integrationstest durchgeführt, bei welchem die Ergebnisse als Bewertungsgrundlage dienen.

5.1.1. Anforderungen

Das entwickelte System erfüllt die spezifizierten Anforderungen in folgendem Umfang:

F-10	Smartphone-Anwendung	
F-10.1	MYO	In der Anwendung ist es dem Nutzer möglich sich mit einem MYO über Bluetooth zu verbinden
F-10.2	GUI-Design	Eine GUI zur Konfiguration steht zur Verfügung. Allerdings ist es noch nicht möglich Skripte auszuführen, wenn sich die Anwendung im Hintergrund befindet
F-10.3	Verwendete Gesten	Von den fünf bekannten Gesten werden vier genutzt. Nicht verwendet wird die Geste „Wave Left“
F-10.4	Gestenfolgen	Das MYO erkennt Gesten. Alle spezifizierten Aktionen (Aufnehmen, Speichern, Löschen , Skript zuordnen) können von der Anwendung ausgeführt werden.
F-10.5	Aufnahme/Ausführung von Gestenkombinationen	Zur Aufnahme der Gesten werden die spezifizierten Gesten wie z.B. die Unlock-Geste ausgeführt. Weiterhin können nur die spezifizierten Positionen und deren Kombinationsmöglichkeiten für die Aufnahmen genutzt werden. Eine Liste der Gesten wird dem Nutzer angezeigt.
F-10.6	Skripte	Die Anwendung unterstützt neben der Sprache „Python“ noch andere Skriptsprachen. Eine Liste der konfigurierten Skripte wird dem Nutzer angezeigt. Ein Skript kann zusätzlich zu den Anforderungen auch- ohne Ausführen einer Geste - im Editieren-Modus gestartet werden.

F-20	Vuzix- Anwendung	
F-20.1	Vuzix	Auf der Vuzix befindet sich eine Anwendung, welche von der Smartphone-Anwendung ausgeführten Skripte verarbeitet.
F-20.2	Smartphone	Die Skripte werden nicht auf der Vuzix ausgeführt, sondern stellen eine Verbindung mit der Vuzix auf und senden Befehle über Bluetooth an diese.

N-10	Performance	
N-10.1	Reaktionszeit	Das Skript wird in weniger als einer Sekunde ausgeführt, nachdem die Gestenfolge durch den Nutzer bestätigt wurde.

N-20	Installation	
N-20.1	Installation Smartphone-Anwendung	Die Anwendung ist installierbar durch das Kopieren der .apk-Datei auf das Endgerät
N-20.2	Installation Vuzix- Anwendung	Die Anwendung ist installierbar durch das Kopieren der .apk-Datei auf das Endgerät

5.1.2. Ergebnisse Integrationstest

Der Integrationstest bestätigte die Korrektheit des Produktes im Sinne des vorher genannten Anforderungen. Auffälligkeiten ergaben sich vor allem bei der Usability des Produktes:

1. In der Gesten aufnehmen-Oberfläche wird die aktuellen Position des MYOs im 3x3-Feld bereits angezeigt, bevor sich das MYO im Aufnahmestatus befindet.

2. Das Umbenennen einer Geste ist nicht erwartungskonform. Das Textfeld, welches den Namen der Geste enthält, sieht editierbar aus. Jedoch reagiert dieses nicht auf einen Klick. Es ist notwendig, einen unscheinbaren Button am Rand des Textfeldes zu klicken.

5.2. Bewertung des Vorgehensmodell

5.2.1. Allgemein

In jedem (Team-)Projekt ist es wichtig Regeln aufzustellen um den korrekten Ablauf des Projekts zu gewährleisten. In Scrum gilt es Fortschritt und Dokumente aktuell zu halten, um die Transparenz zu gewährleisten.
Dafür muss das Sprint-Backlog so aktuell wie möglich gehalten werden. Die Aktualisierung sollte spätestens zu jedem Daily Scrum erfolgt sein, um sich über den aktuellen Stand austauschen zu können.
Weiterhin sollen Tasks, die getestet werden müssen, sich also im Zustand „to be tested“ befinden, schnellstmöglich überprüft werden. Der ursprüngliche Bearbeiter des Tasks trägt die Verantwortung, dass sein Task getestet wird und muss diese Aufgabe entsprechend ins Team deligieren.
Um Meetings und Besprechungen besser organisieren zu können, wurde ein Terminkalender eingerichtet.

5.2.2. Aufgabenverteilung

Eine eindeutige und gleichmäßige Aufgabenverteilung ist keineswegs gegeben und muss mit klaren Regeln herbeigeführt werden.
Dafür haben wir ein zehn minütiges Kickoff-Meeting in jedem Sprint eingeführt, um die initiale Aufgabenverteilung im Sprint zu besprechen. Diese Besprechung wurde im Anschluss an das Planungsmeeting getätigt.
Weiterhin wurde die maximale Task-Größe auf fünf Stunden eingeschränkt und die Regel eingeführt, dass eine Person maximal einen Task zur selben Zeit bearbeiten darf.

5.2.3. Overhead

Scrum-Meetings stellen vor allem in studentischen Projekten einen nicht zu vernachlässigenden Overhead dar. Auf diese Problematik reagierten wir indem wir unser Timeboxing verschärften und den Overhead durch Meetings auf ein Minimum reduziert haben. Dafür haben wir die Standard-Dauer der Meetings verringert und größeren Fokus darauf gelegt, die Besprechungsdauer einzuhalten.

5.2.4. Übernahme bekannter Regeln in Projekten

Nicht alle Angewohnheiten, die aus der praktischen Arbeit oder vergangenen Projekten bekannt und gepflegt werden, sind bei einem neuen Projekt anzuwenden. Ein gutes Beispiel für diese Thematik ist das Thema „Research“.
Unter Research ist die Ausarbeitung von Storys/Tasks zu verstehen, bevor diese offiziell in den Sprint kommen. Dies geschieht in der Regel im vorangehenden Sprint. Im Rahmen einer Research soll geklärt werden, wie eine Aufgabenstellung zu bewältigen ist, um deren Abschätzung und Einplanung in den Sprint zu vereinfachen. Dabei kann zum Beispiel ein Prototyp für eine Implementations-Task entstehen.
In unserem Projekt zeigte sich, dass eine derart feste Research-Regelung oft gar nicht notwendig war oder sogar Mehraufwand bedeutete. Als Fazit diesbezüglich lässt sich zusammenfassen, dass eine Research durchaus sinnvoll sein kann, in den Anfangsphasen des Projekts sogar notwendig ist. Aber gerade bei kleineren Projekten im Laufe des Projekts nicht immer unbedingt notwendig bleibt.
Im Laufe unseres Scrum-Fortschritts, haben wir die wiederkehrenden Research-Tasks abgeschafft.

5.2.5. Wiederkehrende Evaluation der Regeln

Ein interessantes Ergebnis im Laufe des Projekts war eine Retrospektive, bei der es nicht darum ging aktuelle Themen anzusprechen, weil der Sprint sehr gut lief, sondern die bestehenden Regeln zu evaluieren. Innerhalb dieser Retrospektive wurden so viele Regeln aufgestellt und verändert wie sonst nie. Das zeigt, dass auch die bestehenden Scrum-Regeln in ständigen Iterationen geprüft werden sollten.

5.2.6. Eignung von Scrum für studentische Projekte

Oft kam im Laufe der Projekttätigkeit die Diskussion auf, inwiefern Scrum als Projektmodell die geeignete Wahl bei studentischen Projekten ist. Der Overhead wurde bereits genannt, stellt allerdings nicht das größte Problem von Scrum in einem derartigen Umfeld dar. Das größte Problem von Scrum innerhalb eines studentischen Projekts ist, dass es streng genommen keine tatsächliche Sprint-Kapazität der Projektteilnehmer gibt. Es lässt sich zwar eine ungefähre Arbeitszeit pro Sprint festlegen, aber da es keine tatsächlichen festen Arbeitszeiten gibt variiert sowohl die tatsächlich investierte Arbeitszeit, als auch der Schwerpunkt dieser Zeit innerhalb der Woche stark.

6. Fazit und Ausblick

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

6.1. Ausblick

In der Anwendung gibt es Potential für Erweiterungen. Im Folgenden werden mögliche Erweiterungen kurz dargestellt:

Die Skriptausführung wird über eine externe Anwendung auf dem Android-Smartphone übernommen. Hier kann eine eigene Skript-Anwendung geschrieben werden, welche unter anderem auch das Bearbeiten bzw. Editieren von den Skripten ermöglichen kann. Durch eine mögliche Erweiterung der Skripte-Nutzung, kann dem Nutzer auch ein Ausführen von Gesten und den dazugehörigen Skripten ohne eine GUI ermöglicht werden. Dieses Konzept wurde bereits am Anfang des Projekts vorgesehen, wurde aus Zeitgründen aber noch nicht eingebaut.

Bei der Gestenaufnahme kann dem Nutzer eine zusätzliche Möglichkeit der Aufnahme ermöglicht werden. Anstatt eine Geste nur mit dem MYO aufnehmen zu können, kann man den Aufnahme-Modus so erweitern, dass der Nutzer lediglich die Punkte auf dem 3x3-Feld anzutippen braucht, um eine Position zu vermerken.

Ein weiterer Aspekt ist, dass man die bestehende Android-Applikation für Wearables umschreiben bzw. so erweitern kann, dass diese auf dem Wearable selbst ausgeführt werden kann. Dafür müsste beispielsweise die GUI für kleinere Bildschirme angepasst werden. Die Ausführung auf dem Wearable wurde im Rahmen dieses Projekts 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 Anwendung könnte weiterhin an die Android Developer Guideline für Android 5 angepasst werden, um auch grafisch auf dem aktuellsten Stand zu sein.

7. Anhang

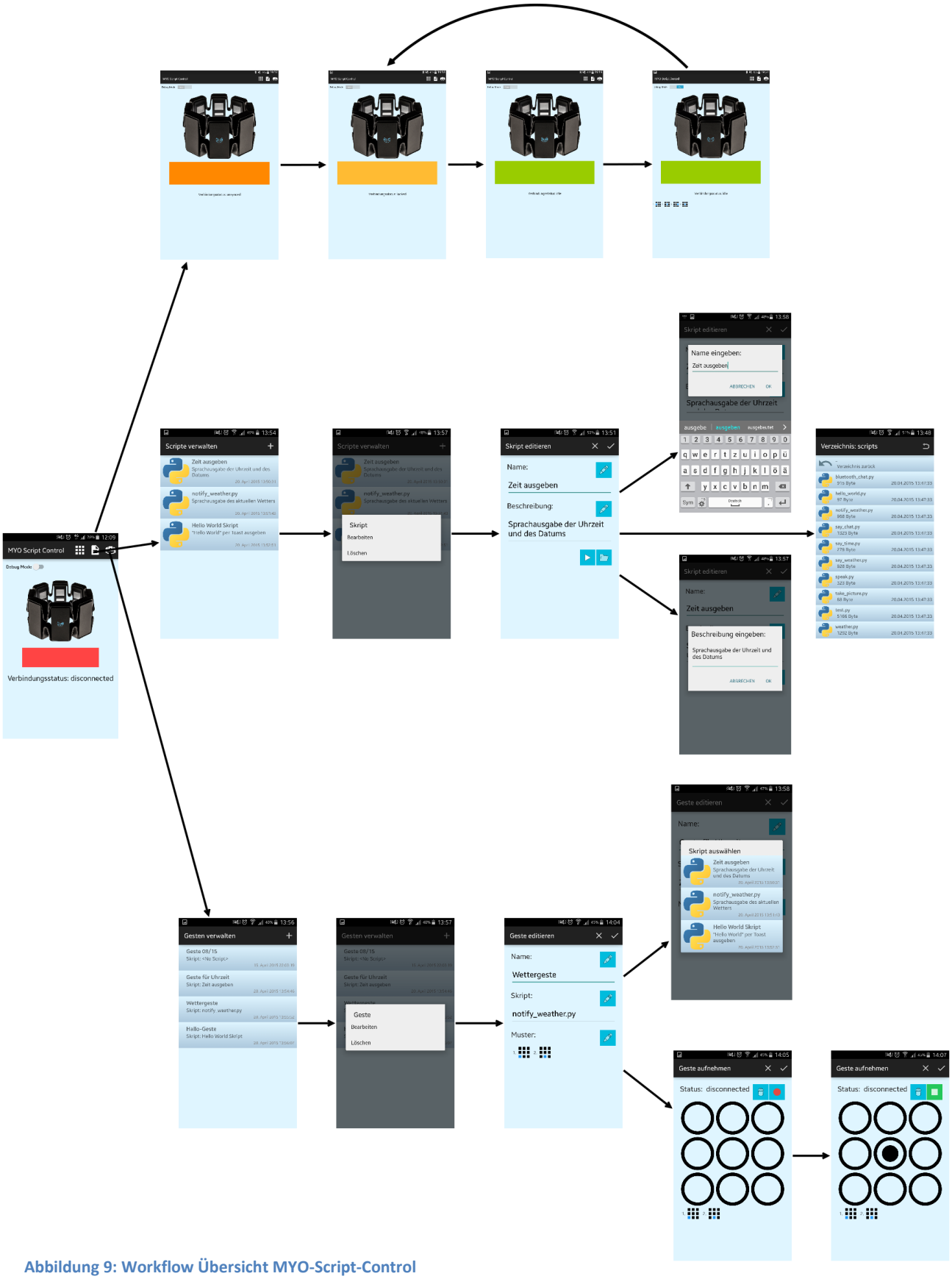

Abbildung 9: Workflow Übersicht MYO-Script-Control

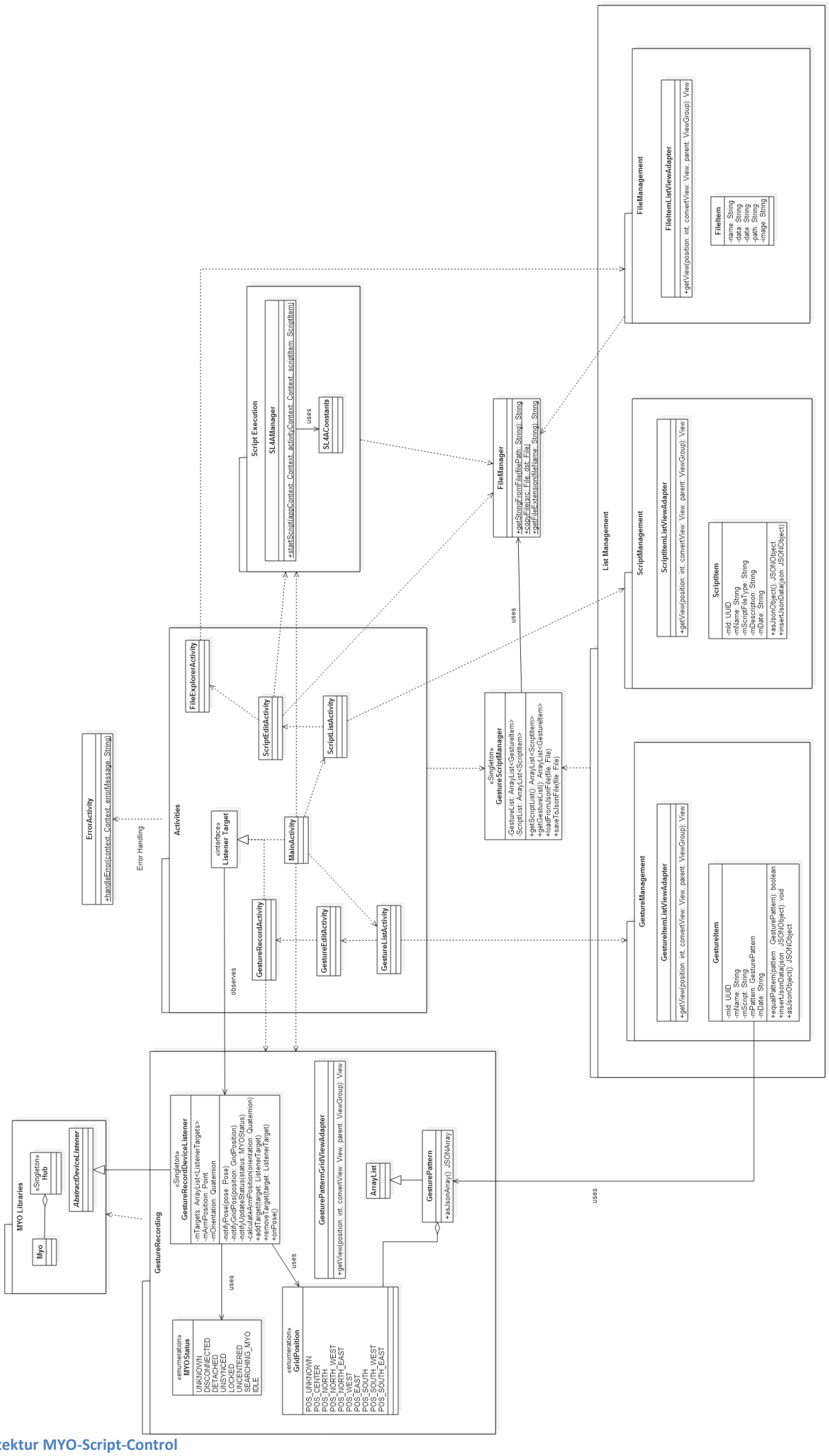

Abbildung 10: Architektur MYO-Script-Control