Technische Dokumentation „Illuminated Mind“

Modul Mobile Systeme / IT-Systeme SS 2019

Prof. Dr. Andreas Plaß
Prof. Dr. Torsten Edeler

Verfasser:

Andreas Scholten (2323152)
Meral Sahin (2285355)
Nadine Krietenbrink (2322554)
Rebecca Bruhn (2338815)

Kurzbeschreibung

Bei „Illuminated Mind“ handelt es sich um ein kurzes Denkspiel auf Grundlage des bekannten Deduktionsspiels Mastermind. Die Spieler müssen durch das Kombinieren von 6 verschiedenen Farben (Elementarrunen) die vom System vordefinierte 4er-Kombination spielerisch herausfinden. Die Steuerung und der Spielablauf werden über eine Smartphone-App geregelt. Die App kommuniziert mit einem Arduino, um Papierkugeln durch LEDs auszuleuchten, die die vom Spieler ausgewählten Farben in der realen Welt darstellen. Wird das Spiel erfolgreich beendet, öffnet sich automatisch eine Schatztruhe, mit Hilfe eines Servomotors.

Aktion & Reaktion im Spielverlauf

Zur Veranschaulichung relevanter Spielphasen hier eine Übersicht zu den Reaktionen der technischen Komponenten auf die in der App ausgeführten Aktionen.

Aktion in der App	Reaktion der technischen Komponenten
Bluetooth-Verbindung aufbauen	-keine-
Starten der App mit Einführung	Kugeln leuchten weiß
Start des Spiels	Erste Kugel bleibt weiß, Rest geht aus
Eingabe der Farben	Kugeln leuchten in den ausgewählten Farben
Feedback	1. Kugel erlischt (jeweilige Farbe falsch) 2. Kugel blinkt (jeweilige Farbe richtig aber an falscher Stelle) 3. Kugel wird heller (richtige Farbe an richtiger Stelle) und bleibt im Spielverlauf an
Beginn Runde x (x >1)	Erste Kugel, die noch nicht richtig ist, leuchtet weiß
Eingabe der Farben & Feedback	s.o.
Spielgewinn	1. Kugeln, die zuvor noch nicht richtig waren, werden heller, anschließend werden alle stetig auf- und abgedimmt 2. Kiste wird angeleuchtet und öffnet sich
Spielabbruch	Kugeln werden weiß

Hardware und Aufbau

Für den praktischen Aufbau des Projekts wurden vier Expolite TourLEDs mit einem Handdimmer in Reihe geschaltet, der mit einer 1kW Leonardo DeSisti Stufenlinse verbunden war. Aufgrund der geringen Anzahl von Geräten, war kein Abschlusswiderstand vonnöten. Anschließend wurden die Geräte für DMX konfiguriert und mit den Kanälen 1-13 belegt. Jede TourLED belegte bei der verwendeten Einstellung ARC.1 drei Kanäle für die RGB-Helligkeitswerte (Kanäle 1-12). Der Dimmer belegte nur einen Kanal (13).

Die TourLEDs wurden anschließend in umfunktionierte Umzugskartons verpackt. In diesen beleuchteten sie durch ein Loch 4 Papierkugeln, die aus Servietten und Biokleister gebastelt wurden. Die Stufenlinse wurde auf einem Stativ befestigt. Sie wurde auf eine kleine Schatztruhe gerichtet, die auf einem weiteren Karton stand. In der Kiste befand sich ein Aufbau bestehend aus einem Servomotor, der verbunden war mit einem Lego-Gelenkarm. Dieser Arm wurde am Kistendeckel angebracht und sorgte dafür, dass der Deckel sich bei richtiger Drehung des Motors öffnete.

Die Verschaltung des Bluetooth-Moduls HM10 und des DMX-Schnittstellenbausteins MAX485 mit dem Arduino Mega, erfolgte wie im Schaltplan ersichtlich (Abb. 1). Der zweite Arduino Uno wurde der Einfachheit halber zur Stromversorgung des Servomotors verwendet. Zu beachten dabei ist, dass die Grounds beider Arduinos gleich sind.

Vor Gebrauch des Bluetooth-Moduls HM10 musste dieses konfiguriert werden, um AT-Befehle empfangen und verarbeiten zu können. Die Kommunikation mit der App fand bidirektional statt.

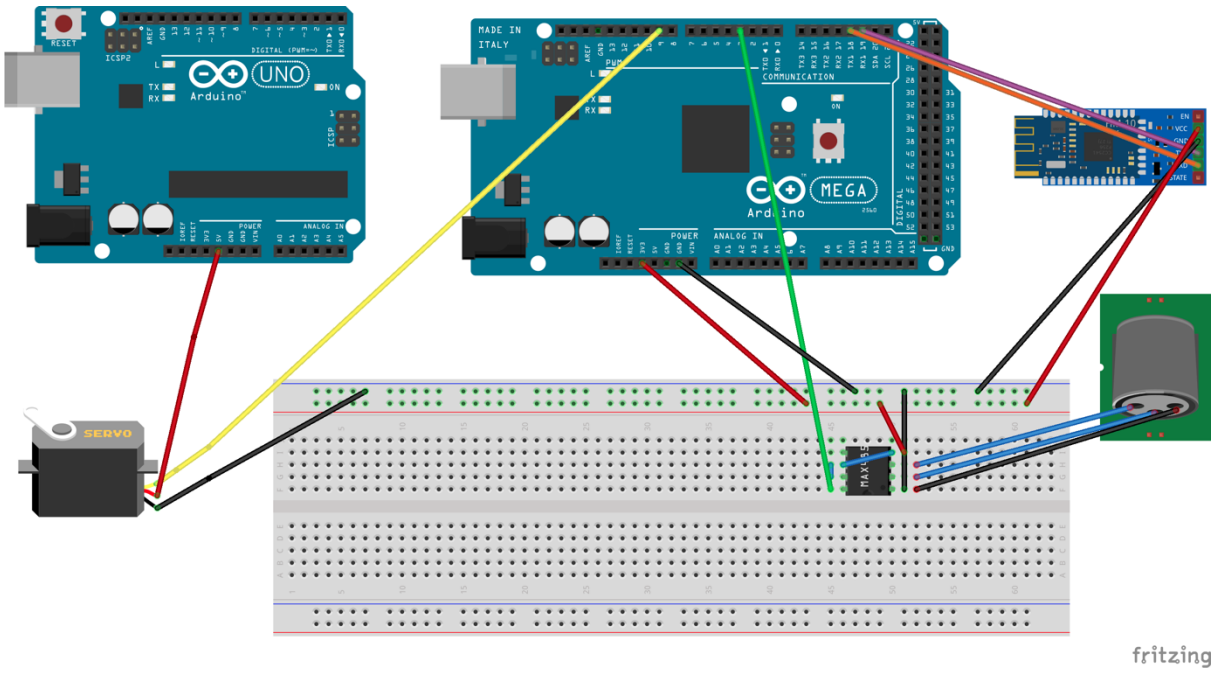

Abbildung 1: Arduino-Schaltplan

Technische Umsetzung des Arduinos

Das Programm für den Arduino läuft flüssig, fehlerfrei und stabil. Nichtsdestotrotz hat es Optimierungspotential hinsichtlich Übersichtlichkeit bzw. Kompaktheit des Quellcodes und u.U. der Performance. Einige der Optimierungspunkte sind Relikte älterer Strukturen, die in Folge der komponentenweisen Erweiterung des Codes entstanden sind. Da leider für jegliche Anpassungen zum Testen und zur Fehlerkontrolle die TourLEDs vonnöten sind und diese uns nicht zur Verfügung standen, entschieden wir uns dagegen, am bestehenden Code relevante Änderungen vorzunehmen.

Programmcode des Arduinos

Der Arduino verfügt über keine eigene Spiellogik. Es wird nur auf Anweisungen reagiert, die als Strings von der App gesendet und über das Bluetooth-Modul vom Arduino empfangen werden. Grundsätzlich wird in vier verschiedenen Anweisungstypen kategorisiert. Es gibt die Start-Anweisungen, die mit “s” beginnen, die Farbänderungs-Anweisungen mit “e”, die Feedback-Anweisungen mit “f” und die Anweisungen für ein Spielende mit “x”.

Start

Zum Spielbeginn wird eine Anweisung beginnend mit dem Buchstaben “s” übermittelt. Darauf folgt die Zahl 0. Alle TourLEDs leuchten weiß auf. Auf die Eingabe “s0” folgt “s1”, wodurch alle Scheinwerfer, außer der erste, erlöschen.

Farbänderung

Der zu empfangende String besteht aus 4 Zeichen. Beispielsweise “e152”. Die auf das “e” folgende Ziffer gibt die jeweils zu verändernde Leuchte (1-4) an und die darauffolgende Zahl einen der möglichen Farbtöne (1-6). Das vierte Zeichen gibt an, welche Leuchte als nächstes aktiviert werden soll, um diese weiß leuchten zu lassen, sofern die Spielrunde nicht vorbei ist.

Um eine Farbänderung auszulösen, wird die Funktion *setFarbe()* angerufen. Diese erwartet als Parameter die Nummer der TourLED, welche geändert werden soll, sowie die Zahl, die den Farbton repräsentiert. In der Hilfsfunktion *Umwandlung()* sind die verschiedenen Farbtöne in Form von 8-Bit-RGB-Werten enthalten. Sie gibt diese bezogen auf den übergebenen Parameter (Zahl des Farbtons) zurück. Die Hilfsfunktion *Position()* ordnet der Nummer des Scheinwerfers (Parameter) die entsprechenden RGB-Kanäle zu und gibt diese zurück. Der Scheinwerfer 3 belegt z.B. die Kanäle 7 bis 10.

In der Funktion *setFarbe()* werden diese Rückgabewerte im Array(String) *pos[]* zwischengespeichert. Anschließend werden die Farbsignale per DMX an die TourLEDs gesendet.

Zum Start einer Spielrunde, die nicht die erste ist, wird ein String der Form “e00x” an den Arduino übermittelt. Das “x” ist erneut die Nummer des nächsten aktiven Scheinwerfers. Da nicht jede Runde zwangsläufig mit der Eingabe des ersten Scheinwerfers beginnt, ist diese Fallunterscheidung notwendig.

Feedback
Als Feedback gibt die App Rückmeldung über die Farbeingabe des Spielers, bezogen auf den Lösungscode. Feedback-Anweisungen beginnen mit dem Buchstaben “f”. Die Art der Eingabe unterscheidet sich von denjenigen der Farbeingabe insofern, als dass nicht für jeden Scheinwerfer ein eigenes Feedback übermittelt wird, sondern in einem String die Informationen für alle Scheinwerfer stecken. Ein String sieht beispielsweise wie folgt aus: “f7899”. Die jeweilige Position einer Ziffer im String korreliert dabei mit der Nummer des Scheinwerfers und der entsprechende Wert an der Position mit der Art des Feedbacks.

Es gibt drei verschiedene Arten von Feedback, auf die die einzelnen Scheinwerfer entsprechend reagieren:

- 7) Die Farbe an dieser Stelle entspricht genau derjenigen im gesuchten Farbcode. Die Helligkeit wird ausgehend von der aktuellen Helligkeit hochgedimmt, proportional zur Variablen “m”. Diese wird mit jedem Programmdurchlauf um 4 erhöht, bis sie größer als 250 ist.

- 8) Die Farbe ist im gesuchten Farbcode enthalten, jedoch nicht an der Position des aktuellen Scheinwerfers. Diese Situation hat zur Folge, dass der entsprechende Scheinwerfer blinkt. Dabei wird im Hintergrund eine Zählvariable “k” hochgezählt und bei jedem ungeraden Wert leuchtet der Scheinwerfer für eine zufällige Zeit zwischen 0,1 und 0,15 Sekunden auf. Bei jedem geraden “k” vergeht eine Zeit zwischen 0,03-0,07 Sekunden.

- 9) Die Farbe an dieser Stelle ist nicht im Farbcode enthalten und somit falsch. Ausgehend von der aktuellen Farbhelligkeit, wird der entsprechende Scheinwerfer proportional zur Variablen “n” heruntergedimmt, bis er aus ist. Die Variable “n” nimmt mit jedem Programmdurchlauf um 10 ab. Um den Fall zu vermeiden, dass die TourLED bei einer schnell folgenden Eingabe noch nicht ganz aus ist, werden bei jeder Eingabe alle betroffenen Scheinwerfer auf null gesetzt.

Spielende
Die letzte Art von Anweisungen, die an den Arduino übermittelt werden, betreffen das Ende des Spiels. Sie beginnen mit dem Buchstaben “x” und lassen sich in zwei Fälle unterscheiden: das Spiel wurde abgebrochen (“xa”) oder gewonnen (“xg”).

Im Falle eines Spielabbruchs werden alle Lichter auf weiß gesetzt. Ist das Spiel gewonnen, so werden zunächst alle Scheinwerfer, die noch nicht “richtig” waren, stufenweise heller, bis alle ein einheitliches maximales Level erreicht haben. Anschließend folgt ein stetiges auf- und abnehmen der Helligkeit für alle Scheinwerfe. Eine Art Pulsier-Effekt in Abhängigkeit zur Variablen “q”. Dafür wird “q” abwechselnd hoch- und heruntergezählt.

Zusätzlich dazu dreht sich der Servomotor, der die Kiste öffnet, um 95° und der Dimmer an der Stufenlinse wird hochgesetzt.

Empfangsbestätigung
Jeder vom Arduino empfangene String wird wieder an die App zurückgesendet, um den in der Testphase auftretenden Fall, dass mehrere Eingaben auf einmal eintrafen und dann nicht einzeln verarbeitet werden konnten, zu vermeiden (siehe: Umsetzung der App - Funktionalitäten der Klassen - Bluetooth Model).

Technische Umsetzung der App

Umsetzung mit dem Framework Flutter

Die App wurde mit der Programmiersprache Dart und dem Open-Source-Framework *Flutter* umgesetzt. Durch dieses ist es möglich, mobile Anwendungen für Android und iOS zu entwickeln, die auf nur einer Codebasis beruhen. Eines der Hauptmerkmale von Flutter sind die sogenannten „Widgets“, mit denen das komplette UI aufgebaut wird. Jedes angezeigte Element kann ein eigenes Widget sein. Widgets werden ineinander verschachtelt, wodurch ein Widget-Tree entsteht. Die sogenannten „Pages“ sind wiederum die einzelnen Ansichtsseiten einer App.

Packages von Drittanbietern

Für die technische Umsetzung der App wurden folgende vier Flutter-Packages von Drittanbietern implementiert:

1. **flutter_blue** (Version 0.5.0)
   Realisierung der bidirektionalen Bluetooth-Verbindung zum Arduino
2. **scoped_model** (Version 1.0.1)
   Weiterreichung von Daten eines Elternelements an die unterliegenden Kinderelemente, dadurch kann von allen Kinderelementen auf die Daten zugegriffen werden
3. **audioplayers (**Version 0.12.0)
   Gleichzeitiges Abspielen mehrere Audiodateien
4. **vibration** (Version 1.1.0)
   Umgang mit der Vibrations-Api auf Android und iOS-Geräten

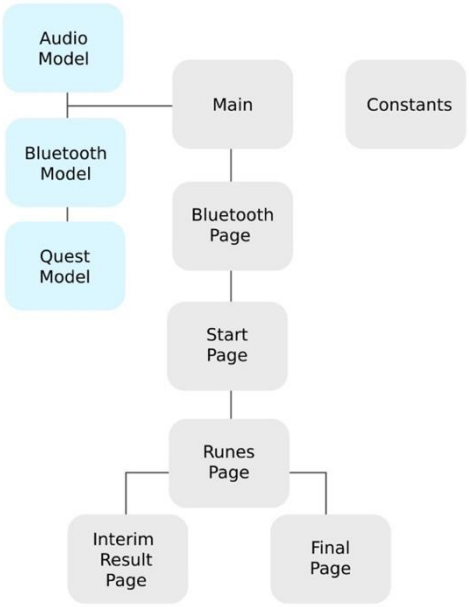

Abbildung 2: Struktur der App

Funktionalitäten der Klassen

Main: Die oberste Ebene in der App ist die Main-Ebene. Hier werden alle *Scoped-Models*, die für diese App benötigt werden, eingebunden, so dass man von überall darauf zugreifen kann. Zusätzlich wird mit Hilfe von *SystemChrome* und *ThemeData* ein allgemeines Design und Verhalten festgelegt. Außerdem werden die Routen definiert, wodurch von überall mit dem entsprechenden String auf eine andere Seite gewechselt werden kann.

Bluetooth Model: Dieses *Scoped-Model* beinhaltet die Bluetooth-Verbindungsdaten, welche auf der Bluetooth-Page initialisiert werden. Außerdem werden über dieses Model Daten an den Arduino gesendet sowie Daten vom Arduino empfangen. Um das zu schnelle Absenden von Daten an den Arduino zu verhindern, werden alle Daten in einem Array gespeichert und nacheinander versendet. Wird von dem Arduino mit demselben Datensatz geantwortet, ist dieser bestätigt und der nächste wird zugesendet.

Quest Model: Das *Quest-Model*, auch ein *Scoped-Model*, ist für die Logik des Spiels zuständig. Es generiert das Endergebnis per Zufall und speichert es ab. Außerdem wird hier das Zwischenergebnis festgehalten und entsprechend ausgewertet. Zusätzlich findet an dieser Stelle auch die Layer-Verwaltung statt, wodurch bei der Runenauswahl immer der korrekte Zustand übermitteln werden kann.

Audio Model: Dieses *Scoped-Model* ist zuständig für das Laden und Abspielen der im Spiel verwendeten Audiodateien. Alle Audiodateien werden zu Beginn in einen Zwischenspeicher geladen, um einen flüssigen App-Ablauf zu gewährleisten.

Bluetooth Page: Beim Start der App wird diese durch die Main-Klasse auf die Bluetooth-Seite geleitet. Durch das Package *flutter_blue* beginnt an dieser Stelle die Suche nach Bluetooth-Geräten in der Nähe. Bei Auswahl eines Geräts, wird anschließend eine Verbindung aufgebaut. Je nach Status der Bluetooth-Verbindung werden unterschiedliche Widgets dargestellt. Ist eine Verbindung aufgebaut, kann mittels eines Buttons das Spiel gestartet werden. An dieser Stelle wird das Endergebnis generiert und das Signal „s0“ als Startsignal an den Arduino übermittelt.

Start Page: Wurde eine Verbindung gefunden, wechselt die App auf die Startseite. Dem Spieler wird an dieser Stelle eine kurze Geschichte zum Spiel erzählt, wodurch ein kleines narratives Erlebnis geschaffen werden soll. Die Darstellung der entsprechenden Texte wird mit Hilfe eines *Statefulwidgets* gelöst. Bei Tab auf den Button wird der Counter hochgezählt und das entsprechende Bild dargestellt. Bei dem letzten Tab auf den Button wechselt die App auf die Runenauswahl und das Signal „s1“ wird an den Arduino gesendet. Das ist das Signal dafür, dass die Runenauswahl begonnen hat.

Runes Page: Auf dieser Seite findet die Auswahl der Runen durch den Spieler statt. Die Aufgabe ist es, insgesamt vier Runen auszuwählen. Bei Tab auf eine Rune, wird die Funktion *_handleRuneTapped()* ausgeführt. Dadurch wird im *Bluetooth-Model* das Zwischenergebnis gesetzt, eine Animation der Rune abgespielt und durch das Package *vibration* eine kurze Vibration des Smartphones durchgeführt. Anschließend wird an den Arduino ein entsprechendes Signal geschickt, so dass die Kugel in der entsprechenden Farbe leuchtet. Wenn alle vier Runen ausgewählt wurden, wird das vollständige Zwischenergebnis ausgewertet, an den Arduino geschickt und je nach Ergebnis wechselt die App auf die Zwischenergebnis-Seite oder auf die Finalseite. Weitere Funktionalitäten auf dieser Seite sind das Abbrechen des Spiels und die Darstellung der Spielregeln mittels eines Icon-Buttons.

Interim Result Page: Wechselt die App von der Runen-Seite auf diese Zwischenergebnis-Seite, wurde vom Spieler die falsche Kombination von Runen gewählt. Beim ersten Rendern der Ansicht, wird das ausgewertete Ergebnis mit der Kennung „f“ am Anfang an den Arduino geschickt. Die Kugeln werden darauf entsprechend beleuchtet. Auf der Seite selbst werden dem Spieler die ausgewählten Runen angezeigt. Mit Hilfe eines Buttons gelangt er zurück zur Runenauswahl, um die nächste Runde zu starten. Dabei wird die Funktion *generateInterimResult()* aus dem *Quest-Model* aufgerufen, welche das Zwischenergebnis für die nächste Runde vorbereitet. Dabei werden alle Stellen aus dem Zwischenergebnis mit dem ausgewerteten Ergebnis verglichen. Wenn eine Stelle falsch ist, wird diese im Zwischenergebnis auf 0 gesetzt. Dadurch werden die richtigen Runen im Zwischenergebnis beibehalten und bei der Runenauswahl nicht mehr dargestellt sowie übersprungen.

Final Page: Wechselt die App von der Runenauswahl auf diese Finalseite, wurde vom Spieler die richtige Auswahl getroffen und das Spiel gewonnen. Die Truhe öffnet sich. Um das Spiel neustarten zu können, wird zusätzlich ein Button dargestellt. Bei Tab auf den Button wird die Funktion *_playAgain()* ausgeführt. Das Spiel wird durch die im *Quest-Model* zur Verfügung stehenden Funktion *resetGame()* zurückgesetzt. Das Signal „xa“ wird an den Arduino geschickt und die App wechselt zur Startseite.