Projektkonzept

Morse-Messenger

.. - ... / -..-. / -- --- ... -.--

Sommersemester 2019
für die Kurse IT-Systeme und Mobile Systeme
bei Prof. Dr. Thorsten Edeler und Prof. Dr. Andreas Plaß
von Jan Jessen (2202032), Maryam Safi (2224027), Malte Saupe (2315546) und Caroline Wolf (2320493)

Inhalt

Einleitung

Für die Kurse IT-Systeme und Mobile Systeme für die Bachelor-Studiengänge Medientechnik und Media Systems des Departments Medientechnik der HAW Hamburg ist als Prüfungsleistung ein Projekt zum Thema „Mobile DMX“ abzugeben.

Technische Voraussetzungen für dieses Projekt sind die Erstellung einer Mobile App mit Kotlin, Swift oder Flutter und die Nutzung eines Arduino, Raspberry Pi oder ähnlichem Gerät und die Verwendung von DMX-Signalen.

Wir realisieren dazu eine Messenger-App mit Morse-Protokoll. Eine Android-App sendet den eingegebenen Text per Bluetooth an einen Arduino welcher ihn in Morse-Code umwandelt und per DMX auf einen Scheinwerfer sendet. Ein identisches Set aus Arduino und Smartphone empfängt den Text mittels eines Helligkeitssensors am Arduino.

Projektziel

Ziel des Projektes ist es, mit zwei identischen Sets aus Smartphone und Arduino mit angeschlossenem Helligkeitssensor, Scheinwerfer und Bluetooth-Modul über eine Raumdistanz zuverlässig kommunizieren zu können, also eine Zweiwegekommunikation mit Entwicklung eines zugehörigen Kommunikationsprotokolls.

Außerdem soll das Projekt im Rahmen der Jahresausstellung am 11.07.2019 voll funktionsfähig ausgestellt werden.

Anforderungsanalyse

Funktionale Anforderungen

DMX & Arduino
Der Arduino
- Muss einen per Bluetooth empfangenen String in Morsecode umwandeln
- Muss über den BT.socket Rückmeldung an die App geben können
- Muss per DMX einen Scheinwerfer steuern
- Muss den Input eines Helligkeitssensors interpretieren und auf relative Helligkeitsänderungen zuverlässig reagieren. (Signal von Rauschen unterscheiden).
- Muss bi-direktional verwendbar sein, also einen Sendemodus, Empfängermodus und Wartemodus besitzen.
- Muss per Paritätskontrolle die Integrität einer empfangenen Nachricht verifizieren können.

App
Die Software …
- muss mit dem Bluetooth-Modul HC-06 des Arduino kommunizieren können.
- muss per Bluetooth senden und empfangen können.
- muss einen Bereich haben, indem man das Gerät auswählen kann, welches per Bluetooth gekoppelt werden soll.
- muss Text-Strings inkl. Smileys in ein Bluetooth-Signal umwandeln können.
- muss die Bluetooth-Daten vom Arduino zurück in einen Text-String wandeln können.

- muss bestimmte Zeichenketten in Strings analysieren können, um bestimmte Zeichenkombinationen in Smiley-Bilder umzuwandeln.
- muss nutzerfreundlich designt sein: Buttons mit derselben Funktion sollten gleich aussehen; Buttons sollten sich farblich vom Hintergrund abheben; Farben sollten so gewählt werden, dass sie genug Kontrast zueinander haben, damit auch Nutzer mit eingeschränktem Farbsehen die App benutzen können; der Name der App sollte sichtbar sein, damit der Nutzer immer weiß, in welcher App er sich gerade befindet; Es sollten max. 5 verschiedene interaktive Elemente pro Activity genutzt werden, um den Nutzer nicht zu überfordern.

Nichtfunktionale Anforderungen

- **Zuverlässigkeit**:
  Es kann ein Text in Länge von 2^8 Zeichen versendet werden, ohne dass der Speicher von App oder Arduino die Zeichen nicht mehr aufnehmen kann.
  Es können in der App mindestens 200 Nachrichten gespeichert werden, ohne dass die Schnelligkeit der App sich verringert.
- **Benutzbarkeit**:
  Die Bedienung der App wird vom Benutzer innerhalb von 1 Minute verstanden und erlernt.
- **Schnelligkeit**:
  Die Übertragung der Datensätze per Bluetooth soll innerhalb von 500 Millisekunden stattgefunden haben.
  Die Übertragung der Datensätze per DMX an den Scheinwerfer soll innerhalb von 500 Millisekunden stattgefunden haben.
  Die Wiedergabe der Morse-Zeichen über den Scheinwerfer soll mit mindestens 10bit/sec stattfinden. Weitere Optimierungen sind wünschenswert.

Technische Rahmenbedingungen

Für die Realisierung werden folgende Geräte benötigt:

- 2x Arduino - 4duino Wireless Modul HC-06 (ist bei Reichelt erst ab 23.04 lieferbar)
- 2x Arduino Mega 2560, ATmega 2560, USB
- 2x MAX485 (bei Reichelt gibt es nicht, nur PCD SHD RS485)
- 2x Android-Smartphones
- 2x Scheinwerfer
- 2x Lichtsensoren (eventuell: Arduino - Grove Sonnenlicht-Sensor v1.0)
- Kabel?
- Link zu dem Warenkorb: https://www.reichelt.de/my/1576939

Die App wird in Android Studio mit der Programmiersprache Kotlin umgesetzt, da die meisten Gruppenmitglieder ein Android-Gerät besitzen und sich die Anwendung so einfach testen lässt. Für Kotlin haben wir uns entschieden, um eine neue Programmiersprache kennenzulernen.
Das verwendete Android-SDK wird mindestens 23 sein, um die aus der Vorlesung bekannte Bluetooth-Service-Klasse nutzen zu können.

Die Umsetzung findet auf Arduinos mit Bluetooth-Modulen HC-06 statt. Die Arduino Software wird in C programmiert.

Für die korrekte Nachrichtenübertragung werden Scheinwerfer und Lichtsensoren eingesetzt. Bei beiden muss die finale Hardware noch ermittelt werden. Kriterien sind beim Scheinwerfer Verfügbarkeit, Helligkeit und schnellstmöglicher Takt (Zeit eines dunkel, hell, dunkel-Zyklus).

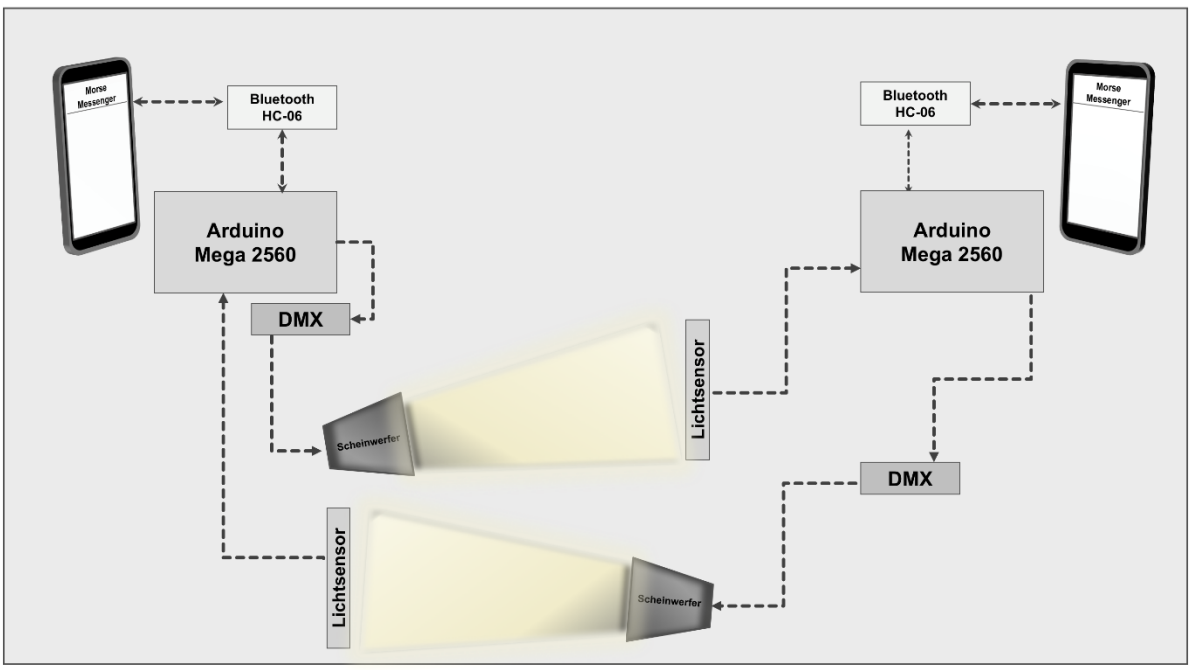
Der Helligkeitssensor sollte zu Kalibrierzwecken analog ausgelesen werden können (kein vorgeschalteter IC) und schnelle Reaktionszeiten bieten.

Die Versionskontrolle erfolgt mittels Git über GitHub.

Technisches Konzept

Der Nutzer gibt eine Nachricht in der App ein. Diese wird per bt.serial an den Arduino gesendet, die einzelnen Buchstaben werden identifiziert, in Morsecode umgewandelt und per DMX an den Scheinwerfer „gesendet“.
Vor jeder Nachricht gibt der Sende-Scheinwerfer als Startsignal das Morse Signal “Beginn” und wartet auf die Bestätigung des Empfängers. Während des Sendevorgangs gibt der Arduino über Bluetooth Rückmeldung an die App. Es werden Informationen darüber verschickt, welcher Buchstabe gerade gesendet wurde. In der App wird dieser Buchstabe grafisch hervorgehoben. Dies dient als Fortschrittsbalken und Sendebestätigung. Über ein System aus Paritätsbits lässt sich eine Empfangsbestätigung realisieren, diese wird ebenfalls in der App angezeigt.

Der Lichtsensor misst die angekommene Lichtmenge und gibt die morsecodierten Informationen an Arduino auf der Seite des Empfängers weiter. Dieser Arduino wandelt sie wiederrum in normalen Text um, was als Ergebnis in der App beim Empfänger angezeigt wird.



App:
Die App besteht aus zwei Activity-Klassen, der Bluetooth-Service-Klasse aus der Vorlesung und einer Nachricht-Klasse.

Aufbau der LogInActivity:

- TextView für „Dein Name“-Headline
- EditText um Namen einzugeben
- TextView für „Bluetooth“-Headline
- Switch, um Bluetooth aus und an zu schalten
- Button „Geräte suchen“

- RecyclerView um die Bluetooth-Geräte anzuzeigen
- Button „Zum Chat“

Aufbau der MessengerActivity:
- RecyclerView um Nachrichten anzuzeigen
- EditText um Nachricht einzugeben
- Button „Senden“

Realisiert wird der ganze Aufbau mit dem Constraint-Layout, da man dort die einzelnen Elemente ohne diverse Verschachtelungen gut positionieren kann. Außerdem ist dieses Layout sehr flexibel und kann sich einfach und gut an die unterschiedlichen Bildschirmgrößen anpassen.

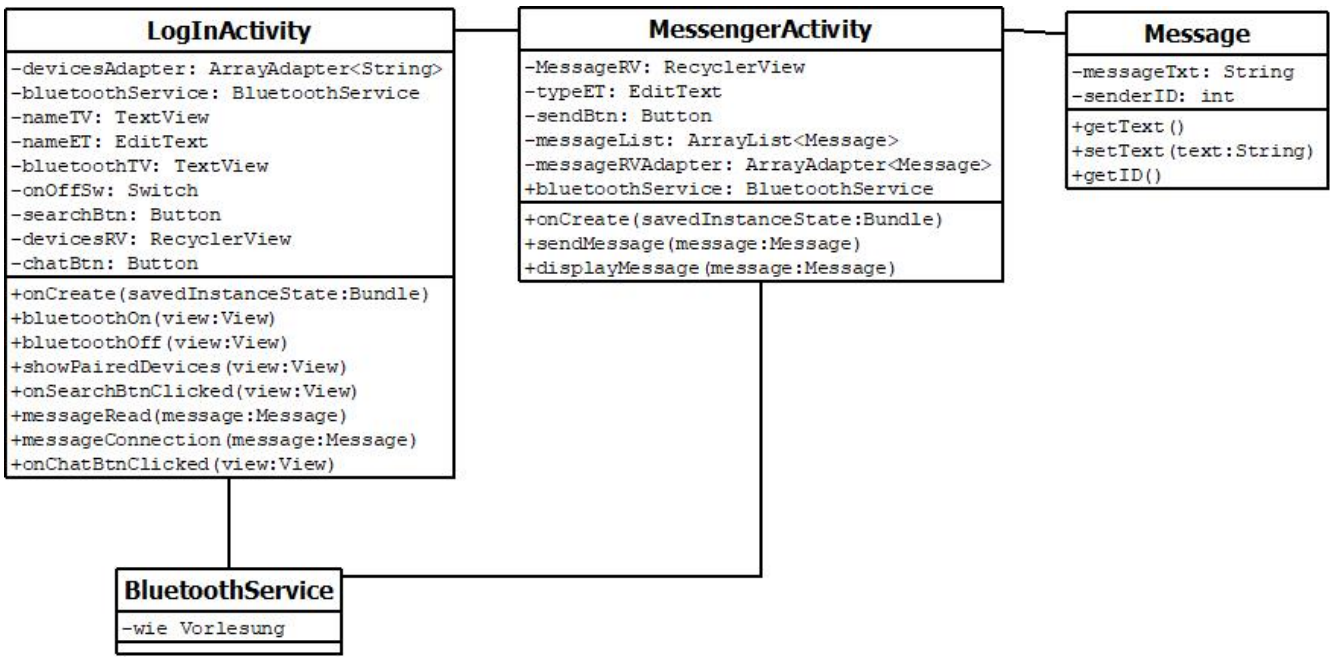
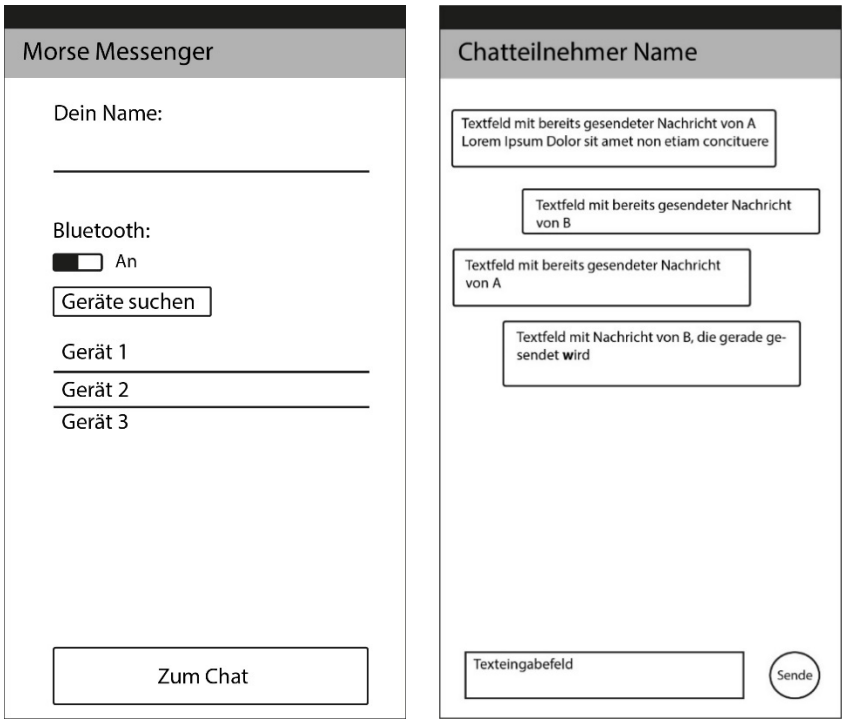


Abbildung 1: UML-Diagramm für die App

Die LogInActivity öffnet sich als erstes. Hier muss ein Name angegeben werden sowie die Bluetooth-Verbindung zum Arduino hergestellt werden. In der OnCreate-Methode wird das Layout aufgebaut. Für die Verbindungsherstellung mit Bluetooth wird die Klasse BluetoothService verwendet. In den Methoden bluetoothOn und bluetoothOff wird die Bluetooth-Funktion des Smartphones an bzw. ausgeschaltet. Die Methode onSearchButtonClicked wird durch den „Suchen“-Button aufgerufen. Es werden die vorhandenen Devices registriert und die Methode showPairedDevices aufgerufen. In dieser Methode wird die RecyclerView mit Einträgen gefüllt.
Per Click auf den „Zum Chat“-Button wird die Methode onChatBtnClicked aufgerufen. Hier wird überprüft, ob eine Verbindung zu einem Device hergestellt und ein Name eingetragen wurde. Wenn beides zutrifft, werden die Daten per Intent an die MessengerActivity weitergegeben und diese geöffnet.

In der MessengerActivity wird wieder in der OnCreate-Methode das Layout aufgebaut. In der sendMessage-Methode wird mit der BluetoothService-Klasse der String aus dem EditText-Feld entnommen und per Bluetooth an den Arduino weitergeleitet. Außerdem wird die Nachricht in derdisplayMessage-Methode in ein Message-Objekt eingefügt und dieses in die messageList eingefügt. Dann wird mit Hilfe des messageRVAdapters die RecyclerView, die die Nachrichten anzeigt, aktualisiert. In der displayMessage-Methode wird außerdem das Bluetooth-Signal analysiert, welches vom Arduino zurückkommt und die einzelnen momentan gesendeten Buchstaben angibt. Der jeweilige Buchstabe wird dann in der zuletzt gesendeten Message gefettet und die RecyclerView wieder geupdatet.

Bedienkonzept



Abbildungen 2+3: Log-In (links) und Messenger-Oberfläche (rechts)

Nach Öffnen der App erscheint ein Log-In Fenster, in dem man seinen Namen eingeben muss. Außerdem muss hier Bluetooth aktiviert und mit einem Gerät (Arduino) verbunden werden. Durch den „Zum Chat“-Button gelangt der Nutzer zur Messenger-Oberfläche. Dies funktioniert nur, wenn die Verbindung mit einem Arduino existiert. Die Messenger-Oberfläche besteht aus wenigen leicht identifizierbaren Elementen:

- In der Leiste oben wird der zuvor eingegebene Name angezeigt.
- Wenn man den Namen des Chatteilnehmers ändern möchte, kann man diesen anklicken. Es öffnet sich ein Dialog-Fenster, in dem man die Änderungen vornehmen kann.
- Ganz unten befindet sich ein Texteingabefeld und ein Sendebutton.
- Wenn man das Texteingabefeld antippt öffnet sich die Smartphone-Tastatur, mit der man sowohl Text, als auch Smileys einfügen kann.
- Durch den „Senden“-Button wird der Text abgeschickt. D.h. der Text wird im mittleren Bereich des Displays als gesendete Nachricht angezeigt.
- Während der Sendevorgang per Morse-Lichtzeichen läuft wird der im aktuellen Moment gesendete Buchstabe der eigenen Nachricht als „Fett“ abgebildet.
- Alle gesendeten und empfangenen Nachrichten können im mittleren Teil der App angesehen werden. Es gibt eine Scroll-Funktion, um ältere Nachrichten lesen zu können.

Aufwandsschätzung

Aufgabe	**Zeit (in Personenstunden)**
Ideenentwicklung	16
Konzept schreiben	16
Arbeitsplatz einrichten mit entsprechender Software	8
Einarbeitung Android und Kotlin	20
Einarbeitung in Arduino/DMX	20
Design für App	8
Programmierung App	50
Programmierung Arduino	50
Qualitätssicherung + Testing	30
Projektmanagement	30
Technische Dokumentation	25
Gesamt	273

Tabelle 1: Aufwandsschätzung

Zeitplan

Der folgende Plan (Tabelle 2) zeigt unsere Zeitplanung für das Projekt an.

Datum	01. – 07.04	08. – 14.04	15. – 21.04	22. – 28.04	29.04 – 05.05	06. – 12.05	13. – 19.05	20. – 26.05
Ideenfindung								
Planung								
Recherche								
Konzeption								
Design App								
Programmierung App								
Programmierung Arduino								
Testen								
Konzeptabgabe		09.04						
Datum	27.05 – 02.06	03. – 09.06	10. – 16.06	17. – 23.06	24. – 30.06	01. – 07.07	08. – 14.07	15. – 21.07
Programmierung App			Puffer	Puffer	Puffer			
Programmierung Arduino			Puffer	Puffer	Puffer			
Testen				Puffer	Puffer			
Plakat								
Technische Doku								
Unsere Deadline	28.05							
Generalprobe		04.06						
Bugfixes/Optimierung					23.06			
Projektabnahme						25.06		
Präsentation							11.07	
Techn. Doku fertig								18.07
Abgabe Deadline								20.07

Tabelle 2: Zeitplan

Meilensteine

Ideenfindung
Brainstorming und Ideenfindung soll bis 05.04.2019 abgeschlossen und in der Gruppe besprochen worden sein.

Planung/Konzept
Das Konzept soll am 09.04.2019 präsentiert werden und muss bis 08.04.2019 fertig erstellt sein.

Tools einrichten
Die nötigen Softwareinstallationen und Implementierung der Libraries sowie Einrichtung der Versions-Kontrolle mittels Git sollen bis 12.04.2019 beendet sein, damit danach direkt mit der Programmierung gestartet werden kann.

Design
Das Design für die App soll bis 28.04.2019 fertig sein. Dies umfasst das Layout-Design im Grafikprogramm sowie die Umsetzung des Designs in Android-Studio.

Programmierung

- App: Design bis 28.04 (siehe oben); In-App-Funktionalitäten bis 12.05; Bluetooth-Implementierung bis 19.05
- Arduino: bis 12.05

Gesamt
Ziel ist es, mit dem kompletten Projekt bis zum 28.05 fertig zu sein, um es bei der Generalprobe am 04.06 präsentieren zu können. Bugs, die nach der Generalprobe auftreten, sollten so schnell wie möglich, spätestens aber bis 23.06 gefixt sein.

Zur Einhaltung der Meilensteine werden von Beginn an wöchentliche Besprechungstermine festgelegt. Hier wird der Fortschritt besprochen, sowie Möglichkeiten Probleme zu lösen, um den Zeitplan einzuhalten und die Meilensteine zu erreichen.

Teamplanung

Teammitglieder sind Jan Jessen, Maryam Safi aus dem Studiengang Medientechnik und Malte Saupe, Caroline Wolf aus dem Studiengang Media Systems

Entscheidungen werden zusammen und nach gemeinsamer Absprache getroffen.

Herr Jessen und Frau Safi übernehmen Simulation und Aufbau der Schaltung sowie die Programmierung des Arduino und das technische Zusammenspiel der Hardware.

Herr Saupe und Frau Wolf übernehmen alle Aufgaben, die zum Thema mobile App gehören (z.B. die Programmierung) und unterstützen gegebenenfalls bei der Programmierung des Arduino und Aufbau der Hardware.