Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projekt Name: LMU Light me up

Studiengang: Media Systems

Semester: 5

Professor: Prof. Dr. Torsten Edeler

Professor: Prof. Dr. Andreas Plaß

Bryan Cichowski MatNr. 2057122

John Adebiyi MatNr. 2102775

Volkan Demiroglu MatNr. 2096856

Inhalt

1 Erste Idee Ziel des Projekts

Eine App, die Abstimmungen, Umfragen und Quiz-Spiele ermöglicht. Diese dann farblich über eine Lampe digital auf dem Handybildschirm und über Monitore oder LED Boards wiedergeben kann. Dabei soll es dem User möglich gemacht werden eigene Abstimmungen oder Umfragen zu starten (Pro - Kontra Prinzip) oder mit anderen Usern einem Quiz oder Multiple-Choice Fragen zu beantworten.

2 Erste Idee Anforderungsanalyse

Es soll eine App konzipiert werden, welche dem User verschiedene Umfragen, Abstimmungen oder Multiple-Choice Möglichkeiten bietet. Dabei soll die App über den Server fürs erste mit einer Lampe verbunden werden, um so das Ergebnis der Abstimmung usw. farblich wiederzugeben. Danach soll die grafische Wiedergabe erweitert werden auf z.B. Monitore oder LED Boards um Ergebnisse und oder Fragen anzuzeigen.

Die App selbst soll über einen Startbildschirm, vielleicht einen Login, verfügen. Danach soll eine Übersicht über die Features kommen. In diesem Bereich kann der User sich für Umfragen/Abstimmungen und Quiz entscheiden (evtl. kommen weitere Elemente/Kategorien hinzu).

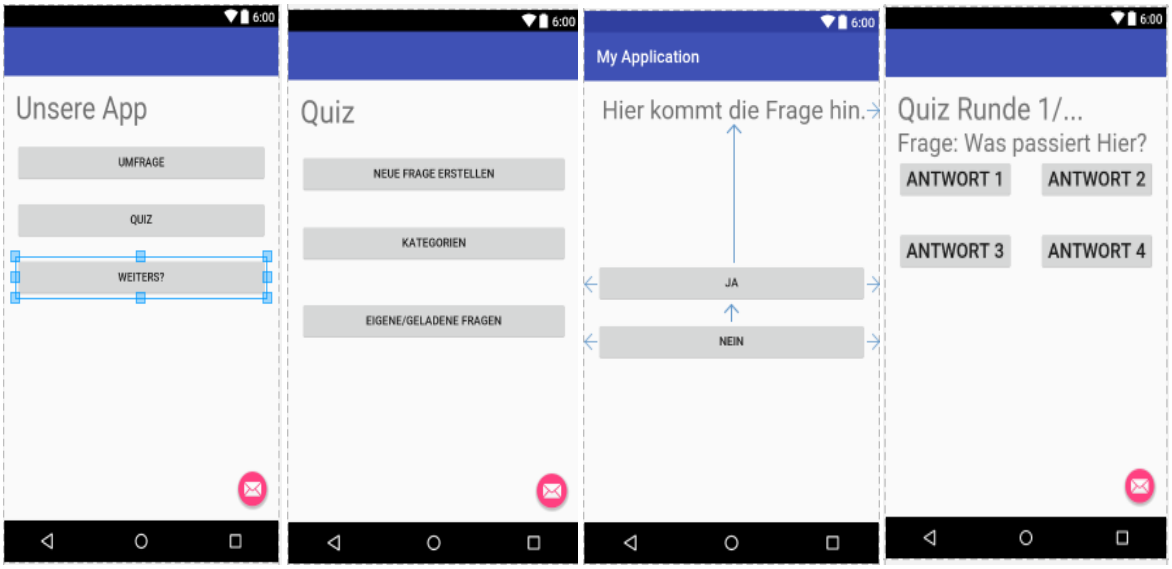
Unter Umfragen/Abstimmungen soll dem User die Möglichkeit geboten werden, vorbereitete Fragen aus diversen Kategorien auszuwählen oder aber selbst Fragen zu erstellen. Dabei soll die Frage für einen bestimmten Zeitraum zur Abstimmung bereit stehen und danach geschlossen werden. Das Ergebnis soll dann auf dem Handy unter der Frage angezeigt werden und farblich auf der Lampe oder auf weiteren Geräten grafisch wiedergegeben werden. Die Wiedergabe auf dem Handy soll grafisch und in Prozent erfolgen.

Unter Quiz soll der User Quiz Fragen aus verschiedenen Bereichen beantworten können, welche vllt. nach Stunden oder nach einer gewissen Anzahl von User-Beantwortungen ausgewertet werden und dann wiedergegeben werden. Des Weiteren soll die App, ähnlich der Abstimmungen, auch Fragen mit Antworten nach einem zeitlichen Intervall einstellen können, z.B. für Tests oder Spiele. Die Antworten und Fragen sollen nach dem Multiple-Choice Prinzip mit vier Möglichkeiten aufgebaut werden. Dem User soll dann ausgegeben werden, wie viele User (Anzahl/Prozentual) die jeweilige Antwort gewählt haben.

Die App soll ihre Anwendung für Umfragen/Abstimmungen und Spiele finden, ob nun im Unterricht oder im Cafe für Unterhaltung sorgen. Z.B. wenn ein Dozent in Verbindung mit der App eine Umfrage startet, kann diese eine Auswertung der gegebenen Antworten von Studenten mit Hilfe von LED Boards liefern.

- **Bedienkonzept:**
    1. Auswahl der Features.
    2. Auswahl in den Features.
    3. Umfragen
    4. Quiz
- **Bedienelemente:**
    - Buttons
    - Pop Up Windows
    - Textview
    - Text Fields

1. 2. 3. 4.



3 Projekt-Entwicklungs-Bericht

3.1 09.05.17 Start des Projekts

Zuerst wurde damit begonnen die App im Android Studio zu programmieren. Die notwendigen Activitys und Displays wurden eingerichtet. Weiterhin hat man sich Gedanken zum Raspberry Pi gemacht. Es wurde versucht dem User die Möglichkeit zu geben, eigene Eingaben mit Hilfe von Datenbanken zu erstellen und auszuwählen.

3.2 16.05.17 Neustart des Projekts

Zum Start wurde die App nun nicht auf dem Emulator, sondern auf dem Smartphone getestet, dieses führte nur zum Abstürzen. Bei der anschließenden Fehlersuche, wurden keinerlei Fehler entdeckt. Daher wurde die App neu programmiert, aber diesmal wurde nach jedem Programmierschritt diese auf dem Handy getestet. Dadurch wurde der Fehler gefunden; welcher durch die Design Theme Auswahl ausgelöst wurde und somit das API Level erhöht hatte. Nach einigem Bugfixing wurde dann das Design im "Hardcode" geändert.

- **Probleme:** Design ändert API
- **Lösung:** Hardcode Design ohne die Theme Wahl.

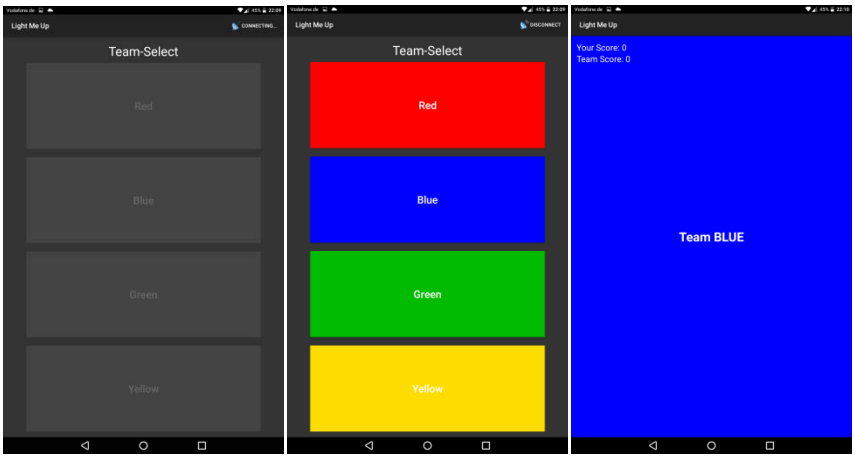
3.3 23.05.17 Zweiter Neustart

Versuch der Implementierung von MQTT bzw. den generellen Verbindungsaufbau zu Raspberry Pi war geplant, wurde dann aber Verworfen, da das Projekt so für Präsentationszwecke zu "langweilig" geworden wäre. Neues Konzept entwickeln und Planen.

Light Me Up entsteht als Idee: Eine Applikation welche den User ermöglicht eines von Vier Teams zu wählen und gegeneinander um die Wette zu Klicken. Dabei soll die App über einen Medienbroker mit dem Raspberry Pi kommunizieren und eine Lampe (16x32 LED Matrix) ansteuern. Die Lampe soll in der Farbe des aktuell führenden Teams leuchten und am Ende das Gewinnerteam benennen.

- **Neues Bedienkonzept:**
    1. Auswahl des Teams
    2. Klicken
- **Neue Bedienelemente:**
    - Buttons
    - Textview (Teamscore)

3.4 30.05.17 Die App wird fertig



Nach der neuen Auslegung der Idee wurde dann die App neu programmiert und alles getestet, damit es auf dem Handy läuft. Auch erste Überarbeitungen im Design und Aufbau. Danach wurde begonnen MQTT zu integrieren. Dieses ging nur über externe Bibliotheken, welche in Android Studio eingefügt werden mussten.

3.5 06.06.17 MQTT

Projektarbeit in der Uni. Nach stundenlangen Verbindungsproblemen, Fehlern und anderen Vorkommnissen sind an diesem Tag nur neue Fehler als Erkenntnis hinzugekommen.

- **Probleme:** VPN und WLAN

3.6 13.06.17 Projekt nimmt Gestalt an

Weitere Bearbeitung der App: Einbau eines Connect Buttons, Einrichtung der Verbindung vom Handy zum Broker und von da zum Raspberry Pi. Erster Verbindungsaufbau.

3.7 20.06.17 Generalprobe

Projekt App fertig, Raspberry Pi die Berechnungen vom Gewinner und das Anbinden einer Lampe fehlt.

3.8 27.06.17 Die LED Matrix ist da

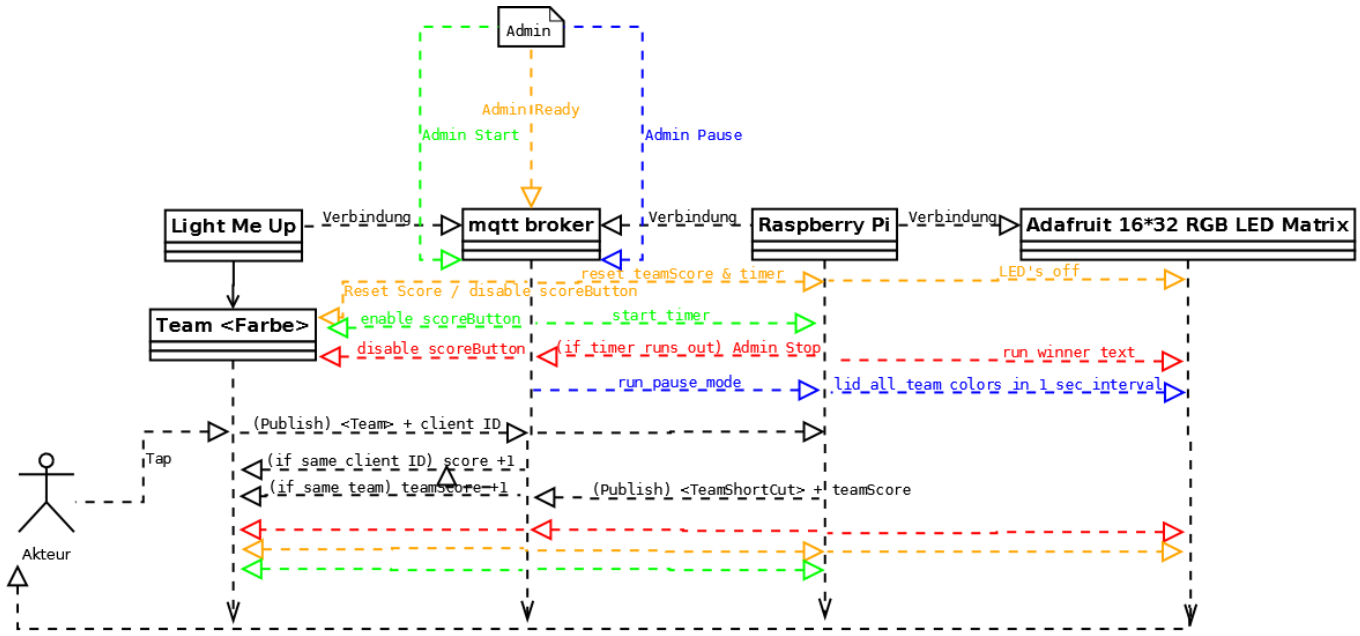
Anbinden einer 16x32 LED Matrix an das Raspberry Pi, welche als visuelle Ausgabe genutzt wird. Programmierung des Raspberry Pi um den Gewinner zu berechnen und Befehle zu empfangen und auszuführen (Start, Stop und Ready).

3.9 04.07.17 Projekt komplett

Projekt Funktionsfähig und nach mehreren Tests Fertig. Kleinere Feinschliffe unter anderem App Icon Design und Name auf den Geräten, Gehäuse für die 16x32 LED Matrix und ein paar kleine Funktionen in der App, wie Standby Licht.

4 Technisches Konzept

Sequenzdiagramm



5 Technische Dokumentation

Die App im Android Studio wurde mit Java programmiert. Die Steuerung der Lampe, sowie später die Berechnung des Teamscores auf dem Raspberry PI wurden mit Python programmiert. Das erste Konzept lehnte sich auf die App und die Funktion mit Datenbanken Inhalte für Quiz und Fragen zu tauschen und selbst zu “Publishing”. Raspberry Pi unterstützt das Konzept nur als visuelle Anzeige für Ja/Nein oder prozentuale Ergebnisse. Später war die

App immer noch wichtig, jedoch bestand das zweite Konzept auf der Intensität des Raspberry Pi und der Berechnung der Gewinner, sowie der Steuerung der Lampe.

Dabei soll der User in der App ein Team wählen und dann durch “Klicken” eines Buttons Punkte für sein Team sammeln, um so zu gewinnen. Die App soll die Punkte (“Klicks”) über das Internet an einen MQTT Medienbroker als Datenpaket senden. Auf dem MedienBroker ist ein Topic auf dem sich die User anmelden und welches die Datenpakete empfängt. Über den MQTT Medienbroker findet durch das Empfangen der Daten, von den jeweiligen Sender (App und Raspberry Pi), die Kommunikation statt. Diese erfolgt dadurch, dass die App (User) und das Raspberry Pi sowohl Datenpakete senden als auch lesen können. Wenn ein Spieler punktet, dann Sender der Raspberry Pi den Gesamtpunktestand der Teams und Spieler an den Broker. Gleichzeitig bedient das Raspberry Pi eine über eine Schaltung verbundene 16x32 LED Matrix. Die LED Matrix leuchtet in den Farben des führenden Teams und dient am Ende als Anzeige des Gewinnerteams. Die App nutzt den berechneten Punktestand um den Teamscore jeden User anzuzeigen. Des Weiteren empfangen beide Geräte Befehle vom MedienBroker, welche das Spiel starten, stoppen und pausieren. Dabei werden bei der App die Bedienelemente gesperrt oder entsperrt und die Punkte zurückgesetzt. Auf dem Raspberry Pi wird die LED Matrix An/Aus bzw. in einen Ruhemodus gesetzt und der Timer sowie die Variablen zurückgesetzt.

5.1 Java

Mit Java haben wir in Android Studio unsere App programmiert, einige der wichtigsten Funktionen sind:

Im Build Gradle.

```
repositories {
  maven {
     url "https://repo.eclipse.org/content/repositories/paho-releases/"
  }
}
```

```
dependencies {
  compile('org.eclipse.paho:org.eclipse.paho.android.service:1.0.2') {
     exclude module: 'support-v4'
  }
}
```

5.2 Im Manifest

```
<service android:name="org.eclipse.paho.android.service.MqttService" >
</service>
...
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

5.3 In der Methode selbst

```
String clientId = MqttClient.generateClientId();
MqttAndroidClient client =
     new MqttAndroidClient(this.getApplicationContext(), "tcp://broker.hivemq.com:1883",
                    clientId);
try {
  IMqttToken token = client.connect();
  token.setActionCallback(new IMqttActionListener() {
     @Override
```

```
        public void onSuccess(IMqttToken asyncActionToken) {
            // We are connected
            Log.d(TAG, "onSuccess");
        }
        @Override
        public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
            // Something went wrong e.g. connection timeout or firewall problems
            Log.d(TAG, "onFailure");
        }
    });
} catch (MqttException e) {
    e.printStackTrace();
}
```

Durch die Einfügung dieser Programmzeilen und zwei externen Bibliotheken ist die Verbindung zu dem MQTT Medienbroker erst möglich. Der Code kann auch erweitert werden um eine automatische SSL Verbindung aufzubauen und das Integrieren von Support Dateien für neuere API Versionen und Funktionen bei Apps zu gewährleisten.

```
            String topic = getString(R.string.topic);
            String payload = getString(R.string.red) + " " + clientId;
            byte[] encodedPayload = new byte[0];
            try {
            encodedPayload = payload.getBytes("UTF-8");
            MqttMessage message = new MqttMessage(encodedPayload);
```

```
client.publish(topic, message);
} catch (UnsupportedEncodingException | MqttException e) {
e.printStackTrace();
}
```

Dieser Programmausschnitt zeigt die Nachricht, die die App an den Medienbroker schickt, um so mit ihm zu kommunizieren.

```
public void subscribe() {
    int qos = 1;
    try {
    IMqttToken subToken = client.subscribe(TOPIC, qos);
    subToken.setActionCallback(new IMqttActionListener() {
    @Override
    public void onSuccess(IMqttToken asyncActionToken) {
        // The message was published
    }
    @Override
    public void onFailure(IMqttToken asyncActionToken,
                Throwable exception) {
        // The subscription could not be performed, maybe the user was not
        // authorized to subscribe on the specified topic e.g. using wildcards
    }
```

```
});

} catch (MqttException e) {

e.printStackTrace();

}
```

Durch diese Codezeilen kann die App sich mit dem Channel, der für die Kommunikation mit dem Raspberry Pi verantwortlich ist, verbinden und somit Nachrichten senden und empfangen.

5.4 Python

Mit Python wurde auf dem Raspberry Pi die Berechnung der Gewinner, sowie deren visuelle Ausgabe programmiert. Einige der Wichtigsten Funktionen sind:

```
import paho.mqtt.client as mqtt

...

url = "broker.mqttdashboard.com"

topic = "Light Me Up"

def on_connect(client, userdata, flags, rc):

    print("connected with result code"+str(rc))

    client.subscribe(topic)

...
```

Damit wird die Verbindung des Raspberry Pi mit dem MQTT Medienbroker aufgebaut, um so Nachrichten zu empfangen und zu senden.

```
...

def on_message(client, userdata, msg):

    task, content = msg.payload.split(" ")
```

...

Diese Funktion beschreibt wie das Raspberry Pi mit eingehenden Nachrichten auf den Medienbroker umgehen soll. In unserem Fall wird die Nachricht in verschiedene Bestandteile unterteilt, auf die das Raspberry Pi dann reagiert.

...

```
elif task == red:
        redScore += 1
        publish(r, redScore)
```

...

```
def publish(team, score):
        scr = str(score)
        message = team + " " + scr
        client.publish(topic, message)
```

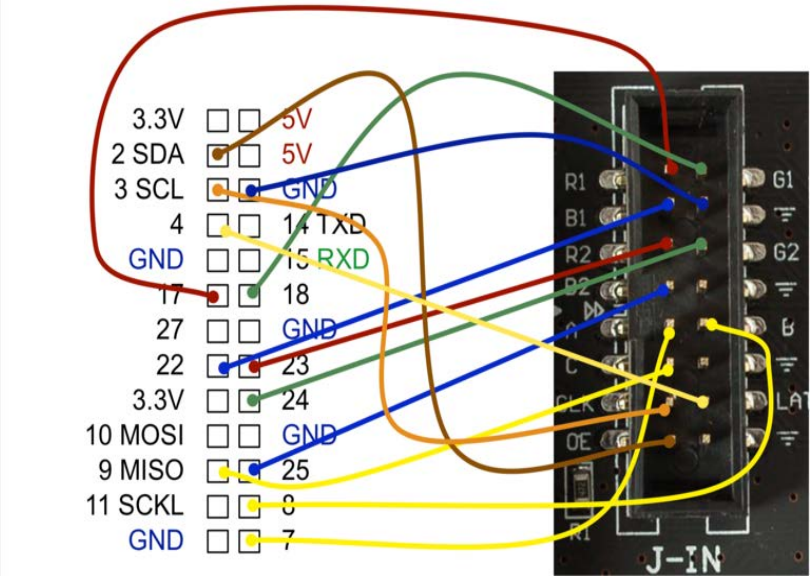
…

Der jetzige Code Ausschnitt zeigt zum Beispiel, wenn ein Roter-Spieler punktet und Red + die ID auf den Medienbroker ankommt, führt das Raspberry Pi den Code aus und erhöht die globale Variable des Teamscores und sendet dann an alle Teamspieler den Teamscore.

6 Probleme

6.1 Verkabelung

Die Verkabelung des Rasperry Pi‘s mit dem 16x32 LED Matrix wurde nach der Anleitung der Adafruit Seite verkabelt. Nach stundenlangen Versuchen die LED Matrix ordentlich zum Laufen zu bringen, wurde nach einiger Nachforschung im Internet festgestellt, dass das Verkabelungs-Diagramm der Seite falsch und veraltet ist



Lösung: Im Internet recherchiert und das richtige Verkabelungs-Diagramm gefunden, mit der das Problem behoben werden konnte.

```
### Wiring diagram
For each of the up to three chains, you have to connect `GND`, `strobe`,
`clock`, `OE-`, `A`, `B`, `C`, `D` to all of these (the `D` line is needed
for 32x32 displays; 32x16 displays don't need it); you find the positions
below (there are more GND pins on the Raspberry Pi, but they are left out
for simplicity).

Then for each panel, there is a set of (R1, G1, B1, R2, G2, B2) that you have
to connect to the corresponding pins that are marked `[1]`, `[2]` and `[3]` for
chain 1, 2, and 3 below.
If you only connect one panel or have one chain, connect it to `[1]` (:smile:); if you
use parallel chains, add the other `[2]` and `[3]`.

To make things quicker to navigate visually, each chain is marked with a separate
icon:

`[1]`=:smile:, `[2]`=:boom: and `[3]`=:droplet: ; signals that go to all
chains have all icons.

|Connection                        | Pin | Pin |  Connection
|---------------------------------:|:---:|:---:|:----------------------------
|                                -  |   1 |   2 | -
|             :droplet: **[3] G1** |   3 |   4 | -
|             :droplet: **[3] B1** |   5 |   6 | **GND** :smile::boom::droplet:
|:smile::boom::droplet: **strobe** |   7 |   8 | **[3] R1** :droplet:
|                                -  |   9 |  10 | **E**    :smile::boom::droplet: (for 64 row matrix, 1:32)
|:smile::boom::droplet: **clock**  |  11 |  12 | **OE-**  :smile::boom::droplet:
|               :smile:  **[1] G1** |  13 |  14 | -
|:smile::boom::droplet:      **A** |  15 |  16 | **B**    :smile::boom::droplet:
|                                -  |  17 |  18 | **C**    :smile::boom::droplet:
|               :smile:  **[1] B2** |  19 |  20 | -
|               :smile:  **[1] G2** |  21 |  22 | **D**    :smile::boom::droplet: (for 32 row matrix, 1:16)
|               :smile:  **[1] R1** |  23 |  24 | **[1] R2** :smile:
|                                -  |  25 |  26 | **[1] B1** :smile:
|                                -  |  27 |  28 | -
|               :boom:   **[2] G1** |  29 |  30 | -
|               :boom:   **[2] B1** |  31 |  32 | **[2] R1** :boom:
|               :boom:   **[2] G2** |  33 |  34 | -
|               :boom:   **[2] R2** |  35 |  36 | **[3] G2** :droplet:
|               :droplet:**[3] R2** |  37 |  38 | **[2] B2** :boom:
|                                -  |  39 |  40 | **[3] B2** :droplet:
```

6.2 Android Studio

6.2.1 API und Design

Bei den ersten Tests, die nicht am Emulator stattgefunden haben, stürzte die App unerwartet ab. Um genau zu sein, startet die App nicht und wirft direkt einen Fehler aus.

Bei erneutem Programmieren des Quelltexts und ständiger Tests auf dem Handy, funktioniert das Programm nun einwandfrei. Als das Design geändert wurde, warf die App wieder Fehler beim Starten aus. Beim Überprüfen der App wurde zudem festgestellt, dass durch die Designänderung die API der App geändert wurde. Um das Design wieder herzustellen, wurden im Manifest Änderungen an einigen Hexadezimalwerten vorgenommen.

6.2.2 Variablen Protected

Bei der Fertigstellung der App, als der Code “bereinigt” wurde, wurden einige Variablen auf Protected gesetzt und der Code der App geordnet. Bei dem darauffolgenden Test startete die App erneut nicht. Da neben dem neuen Anordnen nur einige Variablen auf Protected gesetzt wurden, wurden zuerst diese überprüft. Der Fehler wurde durch bestimmte Variablen ausgelöst, die zwar nur in der Klasse ausgeführt, aber in einem Thread benutzt werden. Die Lösung für diesen Fehler, war das Weglassen von Protected.

7 Auswertung

Viele Probleme sind entstanden aus Mangel an Erfahrung mit dem Android Studio. Auch die Android Studio Java Sprache bat neue Einblicke, die vorher nicht erwartet wurden. Eine Weitere Schwierigkeit war das Verständnis der Einbindung des Medienbroker sowie das richtige Ansprechen und die Benutzung. Trotz allem wurde das Konzept umgesetzt. Neue Erkenntnisse wurden in Python und Android Studio, sowie mit Schaltplänen, einer LED Matrix und dem Raspberry Pi, gesammelt.

8 Quellen

MQTT Verbindung Android Studio

http://www.hivemq.com/blog/mqtt-client-library-enyclopedia-paho-android-service

Python Scanner für Empfangene Nachrichten

https://stackoverflow.com/questions/1751949/python-equivalent-of-rubys-stringscanner

Adafruit 16*32 RGB LED Matrix library

https://github.com/hzeller/rpi-rgb-led-matrix

https://learn.adafruit.com/connecting-a-16x32-rgb-led-matrix-panel-to-a-raspberry-pi/wiring-the-display

Generelle Fragen

https://stackoverflow.com

http://www.python-kurs.eu/