Inhaltsverzeichnis

Betreuer: Prof. Dr. Andreas Plaß

Thomas Roloff (2092802)
Karina Ennenga (2092770)

1. Aufgabe

Unsere Aufgabe war es, eine App für ein Smartphone zu programmieren, die einen Sensor anspricht, dessen Werte ausliest und diese in irgendeiner Form verwendet. Es wurde uns freigestellt für welche Plattform wir die App programmieren.
Wir haben uns für Android entschieden.

2. Projektidee

Wir haben uns dazu entschieden, den Lichtsensor zu verwenden. Gemessen werden soll also die Lichtstärke und der Unterschied zwischen Hell und Dunkel. Unsere Idee ist es, einen Musik Player zu programmieren, deren Lautstärke sich je nach Helligkeit anpasst, der also auf Licht reagiert.
Es soll also, sobald die App gestartet wird und eine gewisse Lichtstärke vorhanden ist, Musik losspielen. Diese soll lauter werden, wenn es heller wird und entsprechend leiser werden, wenn es dunkler wird. Ab einer gewissen Dunkelheit soll die Musik dann ganz ausgehen (z.B. wenn man abends die Gardine zuzieht und schlafen will). Die App läuft im Hintergrund aber weiter und "wartet" darauf, dass es wieder heller wird (z.B. wenn man morgens die Gardine wieder aufmach) und fängt wieder an zu spielen und zwar genau an der Stelle, an der sie am Abend aufgehört hat zu spielen.
Daher kommt auch der Name "Sleep Music!". Es ist eine Aufforderung an die Musik, zu schlafen.

Voraussetzung für das Funktionieren der App ist, dass man Musik auf dem Handy gespeichert hat, wahrscheinlich in einem bestimmten Ordner, auf den die App dann Zugriff hat.

Dargestellt werden soll die App durch die Veränderung der Lautstärke der Musik je nach Lichtstärke. Der Bildschirm soll auch sein Aussehen je nach Helligkeit verändern (entweder andere Farben oder er wird dunkler oder heller)

3. Konzept

Zu Beginn unseres Projektes haben wir und in die bereits vorhandenen APIs reingelesen. Wir haben geschaut, welche es bereits gibt und welche für uns relevant sein können und welche wir davon verwenden sollten. Im Abschnitt 5. APIs kann nachgelesen werden, welche APIs wir verwendet haben und was diese machen.
Anschließend haben wir uns daran gesetzt, einen einfachen Code zu programmieren, der ein Lied auf unserem Smartphone wiedergibt (siehe Abschnitt 5. Wichtiger Code, Beispielcode 1.1). Das war dann schon der erste Teil unseres Grundgerüsts für unsere App.

Als nächstes haben wir uns mit dem Lichtsensor auseinandergesetzt. Wir haben auch hier zuerst einen einfachen Code geschrieben, der uns den Maximalwert unseres Lichtsensors ausgibt. So konnten wir sicher sein, dass wir den Lichtsensor richtig ansprechen und hatten außerdem eine Ahnung, mit was für Werten wir es zu tun haben.
Als nächstes haben wir dann den von uns programmierten einfachen Musik Player umgeschrieben, so dass die Lautstärke anhand der Lichtstärke angepasst wird. Dazu haben wir die Volume nicht mehr fix gesetzt (wie im oben genannten Beispielcode rot gekennzeichnet), sondern die Lautstärke in der Methode *onSensorChanged* anhand der Lichtstärke angepasst (siehe Abschnitt 5. Wichtiger Code, Beispielcode 1.2).

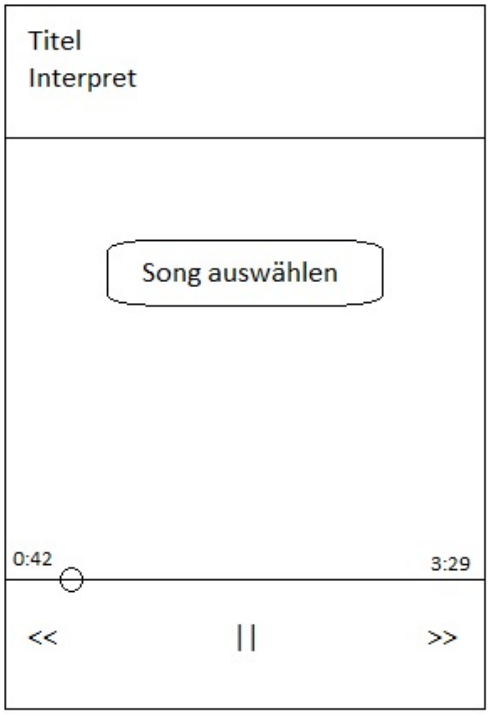
Die war auch unser erster funktionsfähiger Prototyp, den wir bis zm 03.06.2014 fertig gestellt haben wollten/sollten. Die Anforderungen an unseren Prototyp waren, dass er ein Lied abspielen kann, dessen Lautstärke sich je nach Helligkeit/ Lichtstärke anpasst. Auf die grafische Oberfläche und die Usability haben wir noch keinen Wert gelegt.
Wie der Prototyp genau aussah und was für Funktionen er besaß, kann im Abschnitt 4. Prototyp nachgelesen werden.

Ab dem 03.06.2014 galt es für uns, die Fehler, die im Prototyp vorhanden waren, zu beseitigen. Es gab zum Beispiel das Problem, dass das Lied erneut startete, nachdem das Lied gestoppt wurde, das vorige Lied aber auch bei der gestoppten Stelle weiterspielte. Außerdem hatten die Next- und Previous-Button noch keine Funktion und auch die Play- und Pause-Button funktionierten noch nicht wie gewünscht. Wir wollten zum Beispiel, dass der Player, wenn vom

User Pause gedrückt wurde, nicht wieder anfängt zu spielen, wenn sich das Licht verändert, sondern im dem Fall erst, wenn der User wieder auf Play gedrückt hat.

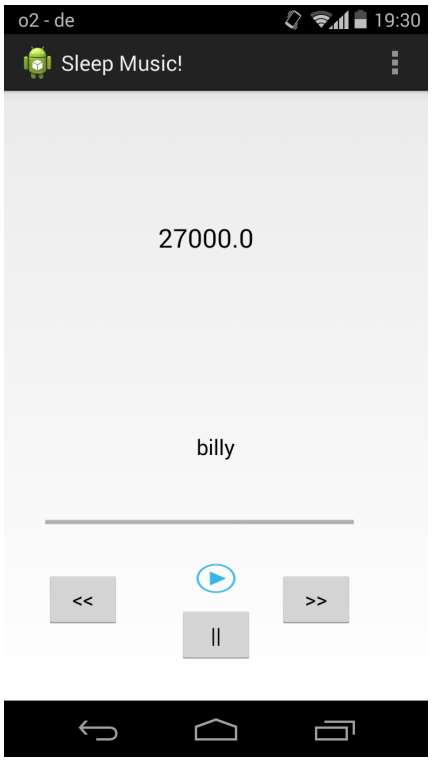
Die zweite große Aufgabe, die wir uns nun stellten, war, die grafische Oberfläche zu erstellen. Wir wollten eine Playlist ausgeben, aus denen ein Lied ausgewählt werden kann. Es sollte außerdem eine Progress Bar angezeigt werden, die anzeigt, wie lange das Lied schon läuft und dies soll auch in Minuten/Sekunden ausgegeben werden . Des Weiteren soll der Songtitel automatisch anhand des gerade gespielten Liedes ausgegeben werden. (Im Prototyp funktionierte dies nur statisch).
Zu den Hauptfunktionen sollte außerdem zählen, dass sich die Hintergrundfarbe der App je nach Lichtstärke ändert.

Unser Konzept sah folgendermaßen aus:



4. Prototyp

Unser Prototyp sah folgendermaßen aus:



Unser Prototyp hatte noch keine großartige grafische Oberfläche. Bei einer App, die die Lautstärke verändert ist es natürlich nicht so einfach zu zeigen, wie bzw. dass der Prototyp funktionierte.
Die Zahl, die zu sehen ist, zeigt die maximale Lichtstärke des jeweiligen Sensors der unterschiedlichen Smartphones. Die haben wir uns einfach ausgeben lassen, damit wir wussten, mit was für Werten wir es zu tun haben.
“billy” ist in diesem Fall der Songtitel, der im Prototypen noch statisch ausgegeben wird, in der fertigen App aber in einer Variable gespeichert werden und ausgegeben werden soll.
Darunter sieht man eine Progress Bar, an der man sehen kann, wie weit das Lied bereits fortgeschritten ist. In der fertigen App soll das dann auch noch in Minuten/Sekunden angegeben werden.
Die Button sind die Standard-Button eines Media Player. Es gibt den Play-Button, der in unserem Media Player aber natürlich nur notwendig ist, wenn die Musik von Hand gestoppt wurde und anschließend wieder weiter spielen soll. Der Pause-Button dient dann wie erwähnt dazu, die Musik per Hand zu stoppen/zu pausieren.
Des Weiteren gibt es noch die Vor- und Zurück-Button, die einfach ein Lied weiterspringen oder entsprechend ein Lied zurück, was im Prototypen aber noch nicht funktionierte und nur der Veranschaulichung diente.

5. Umsetzung

Der Prototyp funktionierte bereits und nun galt es, sich an die Fehlerbehebung / Erweiterung der App zu machen.

Zunächst haben wir uns daran gesetzt, dass die Zeit, die das Lied bereits spielte, ausgegeben wird. Dazu haben wir die Methode *getDuratation()* verwendet. Dieses dann in Minuten und Sekunden umgerechnet und mit einem *TextView* ausgegeben.

Als nächstes haben wir uns überlegt, wie man die Hintergrundfarbe setzt und wie man diese ändert je nach Lichtstärke. Mit mehreren if-Abfragen wechselt der Hintergrund jetzt zwischen ca. 10 Farben je nach Lichtstärke. Wenn es dunkel ist ist der Hintergrund blau und je heller es wird wechselt es zwischen grün, lila, rot, orange bis hin zu gelb.

Um nicht nur immer das gleiche Lied zu hören, sollte noch eine Songliste implementiert werden. Diese sollte aufgerufen werden, nachdem man auf den Button “Song auswählen” klickt. Dazu haben wir eine zweite Activity erstellt, die die Lieder darstellen sollte. Dafür wurde ein SongsManager erstellt, der den Speicher (einen vorher festgelegten Pfad) nach Dateien mit der Endung .mp3 durchsucht und den Pfad und den Titel des gefundenen Liedes in eine HashMap überträgt.
Der Pfad wurde wie folgt festgelegt: *final String MEDIA_PATH = new String("/sdcard/");* .
Um nur .mp3 Dateien (Pfad/Name) mit in die HashMap zu übernehmen wurde eine innere Klasse mit einer Filter Methode erzeugt, siehe Beispielcode 1.4.
Die *PlayListActivity* sollte sich nun um die Darstellung der gefundenen Lieder kümmern. Dazu wurde ein neues Objekt des *SongsManager* erzeugt und die einzelnen Lieder in einer *ListView* ausgegeben. Wird nun auf ein Lied geklickt sorgt die *OnItemClickListener*-Methode dafür, dass die entsprechenden Informationen zurück an die MainActivity(SleepMusic) übergeben werden.

Damit nun der *MediaPlayer* weiß, welches Lied er spielen soll, müssen ihm die Informationen *FilePath* und *SongTitle* übergeben werden. Siehe Beispielcode 1.5. In der *OnItemClickListener*-Methode werden *SongIndex*, *FilePath* und *SongTitle* in Variablen gespeichert. Nun wird ein neuer *Intent* erzeugt, der die *MainActivity* aufruft. Diesem *Intent*

werden die in den zuvor genannten Variablen gespeicherten Informationen mitgegeben. Diese Informationen werden in der *MainActivity* nun ausgelesen und verwendet.

Die *ProgressBar* soll den Fortschritt des Liedes visuell darstellen. Dies wurde mit einem Thread realisiert, da alle 1000ms die aktuelle Position des Liedes mit der Methode *getCurrentPosition*() abgefragt werden musste. Dadurch, dass der Thread jeweils 1000ms “schlief”, veränderte sich der Fortschritt der Progressbar auch nur jede Sekunde. Um die Progressbar zu füllen, musste diese lediglich noch mit der Methode *setProgress*() aktualisiert werden .

Für die Button (Play, Pause, Next, Previous, Forward und Back) haben wir Bilder erstellt und diese dann mittels *ImageButton* dem grafischen Layout hinzugefügt. Für jeden Button haben wir dann eine eigene xml. Datei erstellt, die das Aussehen regelt. Zum Beispiel soll ein anderes Image angezeigt werden, wenn der Button gedrückt wurde.
Für jeden Button haben wir dann einen *onClickListener* erstellt, der jeweils eine *onClick*-Methode enthält. Im Beispielcode 1.3 sieht man den Code für den Play-Button. Dort prüfen wir zuerst, ob der Media-Player gerade spielt. Wenn nein, prüfen wir, ob das ein Lied gerade auf Pause steht. Das prüfen wir anhand der *currentPosition*. Wenn diese größer als 0 ist, befindet sich ein Lied im Player auf Pause. Wenn ein Lied auf Pause ist, soll an der angehaltenen Stelle weitergespielt werden. Andernfalls, wenn noch kein Lied im Player ist, soll die *playSong*-Methode aufgerufen werden. In beiden Fällen wird der Lichtsensor registriert, denn in der *onClick*-Methode des Pause-Buttons rufen wir die *unregisterListener*-Methode auf, die den Lichtsensor “ausschaltet”. Erst mit Drücken der Play-Methode wird (wieder) auf Licht reagiert. Die anderen Button fuktionieren analog, nur mit anderen Funktionen. Beim Next-Button beispielsweise wird zuerst geprüft, ob die Songliste zuende ist. Wenn dem so ist, wird beim ersten Lied wieder angefangen. Ansonsten wird das nächste Lied in der Liste ausgewählt und und der *playSong*-Methode übergeben.

6. Verwendete APIs

Im Folgenden sind die verwendeten API's aufgelistet. Nähere Informationen zu diesen erhält man auf : https://developer.android.com/develop/index.html

- mediaPlayer
- SensorManager
- SensorEventListener
- Sensor
- ProgressBar
- View
- Runnable
- Thread
- FilenameFilter

7. Wichtiger Code

Beispielcode 1.1: *einfacher Music Player* (Ausschnitt)

```
protected void onCreate(Bundle savedInstanceState) {

		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
		final MediaPlayer mediaPlayer=MediaPlayer.create(this, R.raw.billy);

		TextView track = (TextView)findViewById(R.id.track);
		track.setText("billy");

		Button play = (Button)findViewById(R.id.play);
		Button pause = (Button)findViewById(R.id.pause);
		Button reset = (Button)findViewById(R.id.reset);

		mediaPlayer.setVolume(0.9f, 0.9f);

		play.setOnClickListener(new OnClickListener(){
			@Override
			public void onClick(View arg0) {
				mediaPlayer.start();
			}
		});

		pause.setOnClickListener(new OnClickListener(){
			@Override
			public void onClick(View arg0) {
				mediaPlayer.pause();
			}
		});

		reset.setOnClickListener(new OnClickListener(){
			@Override
			public void onClick(View arg0) {
				mediaPlayer.reset();
			}
		});
	}
```

Beispielcode 1.2: Zusatzcode zu 1.1(Lautstärkeregulierung anhand der Lichtstärke)

```
@Override
	public void onSensorChanged(SensorEvent event) {
		//Einstellung der Lautstärke
		float licht = event.values[0];
		float volume = (licht/maxlicht)*100;

		if (volume < 0.1f && mp.isPlaying()== true){
			mp.pause();
		}
		else if (volume>0.1f){
			mp.start();
			mp.setVolume(volume, volume);
		}
	}
```

Beispielcode 1.3: Beispiel OnClickListener für Play-Button

```
//play-Button
	btnPlay.setOnClickListener(new View.OnClickListener() {

	   @Override
	   public void onClick(View arg0) {
	       // wenn Lied auf Pause war
	       if(!mp.isPlaying()){
	             if(mp.getCurrentPosition()>0){
	                  mp.start();
	                  manager.registerListener(listener, sensorLight,
SensorManager.SENSOR_DELAY_NORMAL);}
	                   //wenn neues Lied
	                   else{
	                   playSong(songindex);
	                   }
	            }
	       }
	});
```

Beispielcode 1.4 Filter Methode für HashMap,

```
class FileExtensionFilter implements FilenameFilter {
	 public boolean accept(File dir, String name) {
	     return (name.endsWith(".mp3") || name.endsWith(".MP3"));
	 }
    }
```

Beispielcode 1.5 Übergabe der Songinformationen

```
lv.setOnItemClickListener(new OnItemClickListener() {

	@Override
	public void onItemClick(AdapterView<?> parent, View view,int position, long
id) {
	    // getting listitem index
	    int songIndex = position-1;
	    String go = "go";
	    String songTitle = songsList.get(songIndex).get("songTitle");
	    String songPath = songsList.get(songIndex).get("songPath");

	    // Starting new intent
	    Intent in = new Intent(getApplicationContext(),SleepMusic2.class);
	    // Sending songIndex to PlayerActivity
	    in.putExtra("songPath", songPath);
	    in.putExtra("songTitle", songTitle);
	    in.putExtra("index", songIndex);
	    in.putExtra("go", go);
	    setResult(100, in);
	    // Closing PlayListView
	    // finish();
	    startActivity(in);
	}
 });
```

7. Probleme

Während der Entwicklung der App sind wir auf diverse Probleme und Bugs im Code gestoßen. Zum Anfang hatte wir starke Speicherprobleme, nach dem man einen Song ausgewählt hatte und dann auf Play drückte. Dies lag daran, dass wir nach dem Auswählen des Songs in der Songlist den MediaPlayer erneut initialisiert hatten. Dies konnten wir beheben, indem wir bei wechseln der Activity vorher immer die *release*-Methode des MediaPlayer aufriefen.
Zum Abschluss des Projektes ändert sich die Hintergrundfarbe noch statisch, sprich sie “springt” von einer Farbe zur anderen. Dies hätten wir gerne dynamischer gelöst, so dass ein weicher Übergang stattfindet. Leider fehlte für diese Anpassung am Ende des Projektes die Zeit.

8. Endprodukt

So sieht unsere fertige App aus: