VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

IVAN RZHANOI

MUSIC MOOD

MUSIC PLAYER BASED ON USERS FEELINGS

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Ivan Rzhanoi (e1300545)
Title	Music Mood - Music player based on users feelings
Year	2018
Language	English
Pages	38
Name of Supervisor	Gao Chao

The purpose of this thesis is to find ways of connecting the user's feelings with the music they listen to. This should help to both make the listening experience more intriguing in itself and help improve the emotional state of the listener.

The project is intended to be a prototype, and hence it does not go into great detail in regard to measurements and other similar observations. The main focus is the execution of a concept.

The app was done for iOS on Mac using Xcode development environment with MUSE brain-sensing headband as an accessory. This thesis goes over history of the idea, development process, similar research as well as code implementation.

Keywords music, feelings, emotions, brainwaves

CONTENTS

LIST OF FIGURES AND TABLES

ABBREVIATIONS AND ACRONYMS

Abbreviations

API	Application Programming Interface
iOS	iPhone OS
OS	Operating System
Mac	Macintosh (Apple branded personal computer)
SIT	Shibaura Institute of Technology
CRUD	Create Remove Update Delete (Apple Core Data Database system)

1. INTRODUCTION

The goal of this thesis is to provide a new way of listening to music. This is the first iteration of the idea and it is not supposed to be complete. Providing a prototype for further research and implementation is the top priority.

Music is supposed to be picked up based on the user's feelings, which are determined by the use of brainwaves. At the moment the program takes into account only "happy" and "sad" states of mind, but allows for further expansion with more complicated calculations.

MUSE Headband was used for taking the measurements. The application was created for iOS 10 with Swift 3 and Objective-C programming languages.[1] It also utilises API provided by Interaxon Inc. - the creators of the brain sensing headband.[2]

1.1. Structure

The rest of the thesis includes following Chapters: Chapter 2 discusses project history and how the idea was created; Chapter 3 touches upon similar projects that gave an inspiration and helped shape the app; Chapter 4 describes the basics that have led the project such as tools, development environment and how accessory for the app is working; Chapter 5 goes over the visual design of an app and how it is tied with functionality; Chapter 6 presents inner workings of the program, the way it acquires data and determines the mood; Chapter 7 concludes the thesis.

2. PROJECT HISTORY

The idea for this program originally came during the Slush Hackathon in 2015.[3] During the event attendees were tasked with creating something unique and innovative in the span of three days.

The time constraints were too strict for a project of this size, however, the event introduced the technology available for development. It was the first time when I saw the MUSE Headband, which is used for this project.[4]

With this idea, I needed knowledge to develop my future app. I chose to go for exchange, because my home university did not have the capabilities to teach the usage of brainwaves. The only suitable place to go was Shibaura Institute of Technology (SIT), located in Tokyo, Japan. There I was introduced to Doctor Professor Shin'ichiro Kanoh, who is responsible for the laboratory dedicated to measurement of brainwaves. He and his team of students assisted me throughout my stay and helped me shape the application to its current state.

They taught me about the different types of brainwaves and described some of the states. I was also given a crash course in using professional equipment for measuring the brainwaves. Even though my tool is much simpler and cheaper, the ability to use theirs gave me a much clearer understanding of brain operation and gathering of the information.

This application was done in half a year during my exchange period. The majority of this time was delegated towards fixing the bugs within the application.

There is a remarkable lack of comparable research open to public. The analysis done in the laboratory for determining the users' mood was not substantial enough to deduct any strong correlation of selected brainwaves with their emotional state. Instead the idea was guided by various internet sources only as a point of reference.

The main goal of creating the prototype was a limited success. As of writing this thesis, the app utilises only one brainwave to determine the feelings. It however does have the

mechanism and logic for using another four to get more precise results and choose various types of moods.

3. SIMILAR PROJECTS

The main competitor is Neurowear, which was founded in 2011.[5] In April of 2012 they debuted on the market with “necomimi” - headband with cat ears that move depending on the user’s emotion. Their purpose is cosplay. It is based on MindWave headband made by NeuroSky. The original headband was sold for 99 dollars and was the cheapest headband sold by Neurosky.

Necomimi is rather simple in its operation. While Neurowear claims that it uses emotions, in reality it only utilises an alpha brainwave, which corresponds to user concentration. When the user is concentrated, the cat ears move up to show excitement.

3.1. Neurowear Product range

Neurowear offers a variety of other products. In addition to the aforementioned ears, they now have an electronic cat tail that is referred to as “shippo”. Neurocam is a system that takes photos with the smartphone when the user is concentrated. The idea is to capture life moments when the user feels them. The same concept is applied to BrainBookmark, except instead of pictures it saves web pages.

"Neuro tagging map" is trying to get the data of the user’s feelings depending on the place. For example, your friends can know whether you liked a certain cafe or not based on your experience.

“Mononome” is trying to bring robots into our life through fun and interactive experience. Mononome represents two eyes expressing emotions.

“Cotorees” is shaped as a small bird and maintains functions of personal assistant akin to Amazon Alexa or Google Assistant.

“NEURO TURNTABLE” plays music only when the user is concentrated on it. It helps people give higher appreciation to the listened tracks, because if the user is not truly listening to music, it stops.

“Brain Disco” was an experiment where DJs needed to keep the audience's attention. If the attention was too low, they were kicked out of the competition. This goes on the op-

posite end of a spectrum compared to the previous project as this time DJ is the one in need of keeping attention of other people and not the people themselves.

Moving closer in similarities to the project I have done is “ZEN TUNES”. It works in the background during music listening sessions and compiles playlist based on the relaxed or focused state of listener. So later, the user can pick up the song to help them either concentrate or otherwise mediate.

3.2. Competing project

Finally, there is “mico”. The name comes from “music inspiration from your subconsciousness”. Rather bulky headphones have various brainwave sensors inside them. An iPhone app plays music depending on the users feelings just like my project. I have to admit however that this project has an advantage of being built into a headset already. This simplifies a few things. The general principal of operation however is otherwise similar to my project.

Sadly, after the initial presentation in 2013 there was not much info present regarding these headphones. The company is still active and is working on other projects, but mico is seemingly abandoned.

Figure 1. Model Julie Watai wearing a mico headset[6]

4. BASICS OF PROJECT CREATION

The function of our brain is based on electronic impulses between neurones. Neurones are constantly generating ionic current, and with it electromagnetic field. Those electromagnetic fields oscillate a certain amount in the unit of time (second). There are five different brainwaves and each has a corresponding frequency range:

• Gamma	30 - 50 Hz
• Beta	14 - 30 Hz
• Alpha	8 - 14 Hz
• Theta	4 - 8 Hz
• Delta	0.1 - 4 Hz

Figure 2. Brainwaves[7]

In addition to these, there are also Mu (8 - 12 Hz), Sigma (12 - 14 Hz) and Sensorimotor rhythm (12.5 - 15.5 Hz) waveforms. The last one is commonly abbreviated as SMR. They are often delineated in electroencephalographic measurements. They are omitted in this project, because precision is of lesser importance compared to performance. Not to mention the fact that the used equipment itself does not register these waves.

In either case, five main waves shall provide a substantial amount of data to determine the mood of user.

4.1. Preliminary research

Similar research is not very common and due to the tight schedule I was not able to observe the exact correlation between values of brainwaves and moods of the user. Instead I have used a “Brainwaves Analysis of Positive and Negative Emotions” (Fu-Chien Kao, Shinping R. Wang and Yu-Jung Chang) discussing the determination of emotions based on the brainwaves.[8] While not used in its entirety, it gives a good perspective for my application.

Researchers observed 15 subjects, both male and female. They were given three different audio stimuli after the experiment and were asked to decide, which was suited best

to their current emotion. Before that they were listening to 20 seconds of sounds with 10 seconds of breaks in between. Their brainwave was recorded and compared with other subjects.

Observers have defined eight emotional states: four positive and four negative. Interestingly enough they are put into positive-negative pairs: Joyful-Angry, Surprised-Fear, Protected-Sad, Satisfied-Unconcerned. The reason for this is that the brainwaves for those paired emotions are extremely similar, the negative being just slightly more powerful. Even the expensive equipment had trouble detecting the difference. Researchers provided the diagram, which can be seen at the figure below.

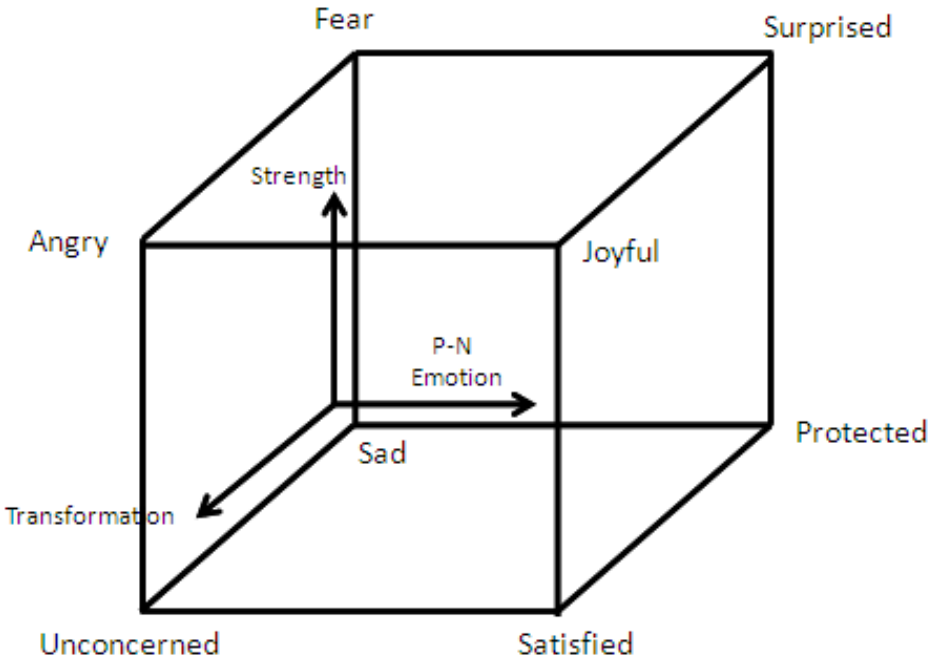


Figure 3. Emotions arranged in relation to brainwaves[8]

Note that Angry and Joyful emotions (for example) are positioned on the same level of Strength and Transformation (figure 3). All emotions are positioned according to Strength on the vertical axis and Transformation on the left side axis with positive emotions on the right side of diagram on the horizontal right axis. While the average power of negative emotion is slightly greater, they decided to omit the difference due to possible inconsistencies in measurements. Meaning, if we were to have a 16-th subject and they would not tell us their current emotion, we would not be able to tell ourselves from

within a pair (Angry - Joyful for example). Different people have different average "power output" in the brainwave measurements.

This provides challenges for creation of my application. Because of this, I decided to temporarily concentrate on Sad and Joyful emotions due to them being completely polar opposite in terms of produced brainwave. According to the diagram Sad emotion should have low values for most brainwaves, while Joyful emotion has higher ones. At the moment, in my application I choose one of two emotions based on alpha brainwave for simplicity, although implementation allows for use of other emotions and brainwaves.

For further details, such as power averages, please see to the referred material. (Please note, I will be quite often referring to Joyful emotional state in my application as "Happy".)

4.2. Equipment

When comparing the MUSE headband with professional grade machinery used in the laboratories we will firstly focus on the operation of equipment utilised at SIT laboratory of Doctor Professor Shin'ichiro Kanoh. It and many other similar systems are used in most of the brainwave researches.

The subject has electrodes attached to their scalpel. These electrodes can register the brainwaves and present it in the form of voltage, which is then carefully amplified. Such measurements always have problems with noise generated by both external and internal factors. A more expensive equipment generally yields more precise results. Signals are registered by a PC using special connectors operated by a Matlab code. The data is typically recorded either in CSV or Excel database.

Thankfully, MUSE Headband simplifies most of the work. It is much simpler in terms of its construction. We will analyse its physical design presented in the picture below and then discuss data acquisition.

Figure 4. Construction of MUSE Headband[9]

Professional equipment usually has many different places to attach electrodes. Meanwhile, the MUSE Headband has only seven sensors in total. Five of them are regular electrodes. Two conductive rubber sensors are used only for auxiliary data, to level the voltage levels from the brain and make the results more precise.

The interesting point of the MUSE Headband is that it presents data to developers not in the voltage spectrum. Instead, it takes the general voltage of your brain (that being the voltage generated by brainwaves) and compares it with voltage levels of other users. After that gives a decimal value between 0 an 1 (for example 0.3456).

It is possible to get the exact voltage levels of brainwaves, but for the purposes of this application they are omitted. We do not require the precise value, we only need to tell whether the alpha or beta brainwaves are “strong”.

Alpha brainwave is commonly used to determine the concentration of a user. The more concentrated the subject, the stronger the emotional response in their brain, which is shown with a higher voltage value. On MUSE Headband it is shown closer to 1 than 0, to represent that.

4.3. Development environment and target platform

Now I would like to discuss the development platform. I will try to explain why I chose it and my equipment. I was creating this application for iPhone first. There is a number of reasons for that.

I have decided to develop for smartphones, because the MUSE Headband as a product is firstly oriented towards ordinary people and not the research scientists. It is assumed that the program will be used with a smartphone before all else. It is possible to do some purely scientific research with the MUSE Headband, but most of the manufacturers support goes towards mobile developers and this is not the goal of this thesis.

An iPhone has been chosen as the main target platform due to my honest lack of proficiency with Windows phone and Android development. Also, neither Android nor Windows devices are in my possession. The iPhone 6S was my daily driver at the time, which simplified certain areas of testing such as UI Design and UX as I was familiar with the equipment. On top of that, development environment has less variations due to similar hardware lineup. It must be also admitted that there is personal a preference involved as I feel more comfortable developing for Apple ecosystem. Lastly, Apple has released a new programming language, which is fast, versatile and offers some great power combined with flexibility. I used Xcode 8.3 with Swift 3.

The MUSE Headband was chosen, because it was the first brain-sensing headband encountered by me. It is also relatively popular, which means a wide spread of my application. The API Reference provided by Interaxon Inc. is quite sufficient; most of the API components are simple and work together effortlessly.

There have been a few problems during the development ranging from incompatibility to code conversion to the headband simply refusing to connect. All this however did not stop the project from release.

4.4. Setup

I would like to go over the implementation of API for MUSE Headband, which is available at their developer portal. Unfortunately, despite the initial ease I still have encoun-

tered certain problems. An official example is made in Objective-C, which has certain constraints. The problem is not only the age of the language, but also the way Xcode handles certain things. For example, Swift applications employ a storyboard file, which can show the layout of an entire application within all available windows. Objective-C however requires a new file for each new application window. This would make the process longer and more complicated. It was chosen to port the code to Swift 3.

The API implementation took longer than expected. While being overall simple, it presents a few caveats.[10][11][12][13][14] Case in point, SDK download and full installation of MuseLab and LibMuse Applications is required no matter whether it will be used or not. LibMuse folder hosts “Muse.framework” file that needs to be imported into an Xcode project. In Xcode, under Build Phases “Muse.framework” is added into “Link Binary With Libraries”. Then we need to click on “+” in the same subsection and add “libc++.tbd” to the project. Bitcode also needs to be disabled in project settings. “Info.-plist” requires “com.interaxon.muse” next to “Supported external accessory protocol” as “Item 0”. Finally, the Muse header is imported as “Muse/Muse.h”.

A large chunk of time was dedicated towards solving all those problems and many others. The MUSE Headband was also refusing to connect and send data for very long.

5. APP DESIGN

App functionality is tightly related to its visual representation in Xcode and the functions are generally triggered by the user's actions except for a few files that calculate the values of brainwaves. The attempt was to make interface as simple as possible.

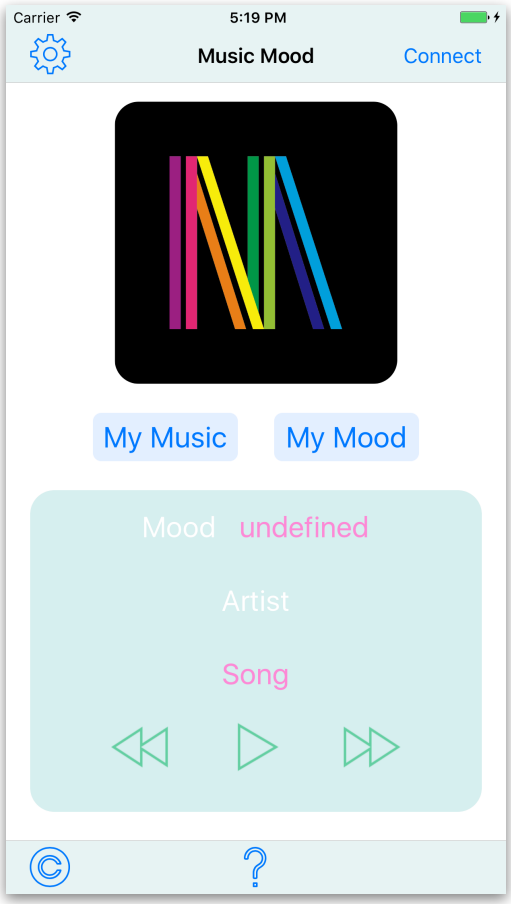
(Please note, the app might be further updated after publishing, which means that the design discussed in this paper might not reflect the one offered in the current version)

Swift 3 and Xcode allow developers to create an app with multiple windows. They are referred to as Scenes. Each scene has a ViewController assigned to it. ViewController is a file written in Swift (sometimes Objective-C), which executes functions written inside it. Scenes are often referred to as ViewControllers, because both of them are united in their functionality and without each other they would be useless.

5.1. Main window

The application utilises the iOS system music player with iTunes library. As such, before any operations, the app asks for permission to access the MusicLibrary. Without access to Media Library the app would not perform its main functionality, that being the playback of music corresponding to the user's feelings.[15] The songs are supposed to have tags in their info section (#happy #sad).[16]

In the beginning the user is greeted with the main screen (figure 5-1). In my project it is connected to ViewController named "ViewController.swift", because it is a centre point of MusicMood application. The user is going to spend most of their time here.



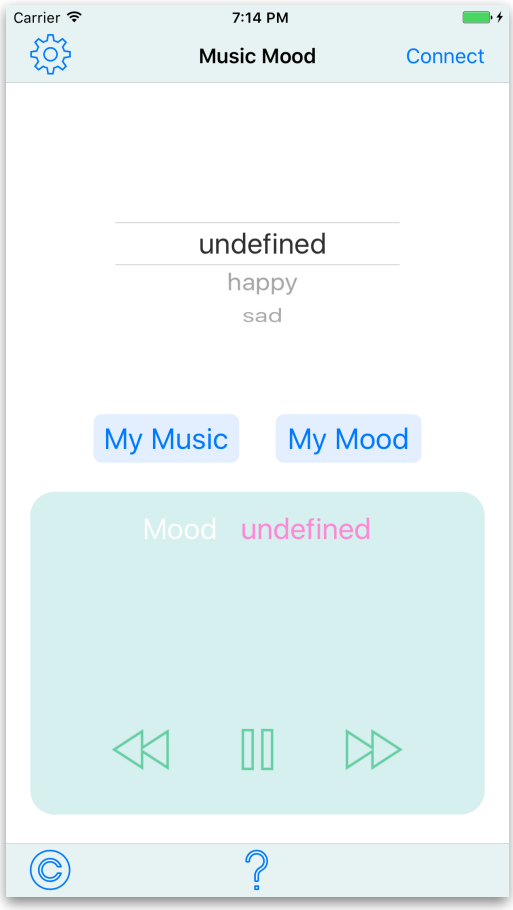


Figure 5-1. With icon, paused

Figure 5-2. With mood selection, playing

Figure 5. Application layout (main window)

As you can see in figure 5-1, there is a logo of the MusicMood application. If the user does not have the MUSE Headband or does not want to utilise it, then they can tap on a button with the text “My Mood”.[17] The view at figure 5-2 is shown.[18] That little section is a PickerView,[19][20] which is referred to in the code as MoodPicker. For now there are three options: Undefined, Happy and Sad. The user can pick the option corresponding to their feelings. When any of the media buttons are pressed, the list of mental states disappears and in its place artwork appears again. By default, it is an icon of this app. However, if there is a song that will be played, then the app displays the album cover.

By pressing My Music button, we will be presented with a list of all the songs we have. This is a Swift 3 builtin function, which not only does that, but also organises all the tracks in a concise manner.

The layout of the app is arranged in a way that visually splits it in half. The lower part represents the Player part. It is all gathered inside a rounded rectangle with a mixture of blue and green colours.[21][22] Inside there is a label showing "Current Mood".[23] Current Mood label is constantly changing after the new mood of user has been determined. The mood is determined after a song is finished. Then we have Artist and Song labels. They are constantly changing as well. Note, that in figure 5-2 Artist and Song labels have disappeared, because the simulator was used for screen capture and it does not have any tracks stored inside of it.

You may also see the basic playback controls, which are Rewind, Play and Fast-Forward buttons. When the track starts playing, the image of Play button changes to Pause as can be seen in figure 5-2. For user this is would be a much clearer portrayal of the player's operation.

Meanwhile, the upper part represents some additional information. If we were out of screen space, the upper part could be essentially cut off. The app was built with a MUSE Headband in mind, so the importance of MoodPicker was lessened during the course of development.

5.2. Legal information and Usage instructions windows

ViewControllers for Legal Information and usage instructions windows do not perform any functions. Their purpose is only to provide user with information and be dismissed upon tap. In figures 6-1 and 6-2 you may observe buttons at the bottom. Upon pressing them speech bubbles are shown.[24] Two design principles behind them are simplicity and playfulness. For this reason the windows were chosen to be smaller than the screen and presented as speech bubbles.

As a side note, if you read the "How to use" text, you may notice that process of operating the app is rather complicated. There is a great challenge in simplifying the instruc-

tions as part of MusicMood is dependant on external factor, iTunes Library as you may already know. Hopefully in the future its usage can be mitigated or simplified.

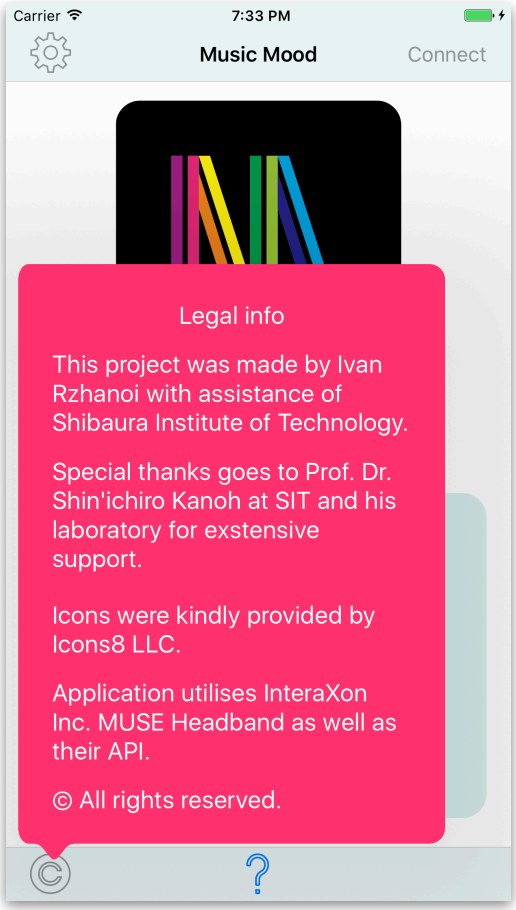


Figure 6-1. Simple legal information

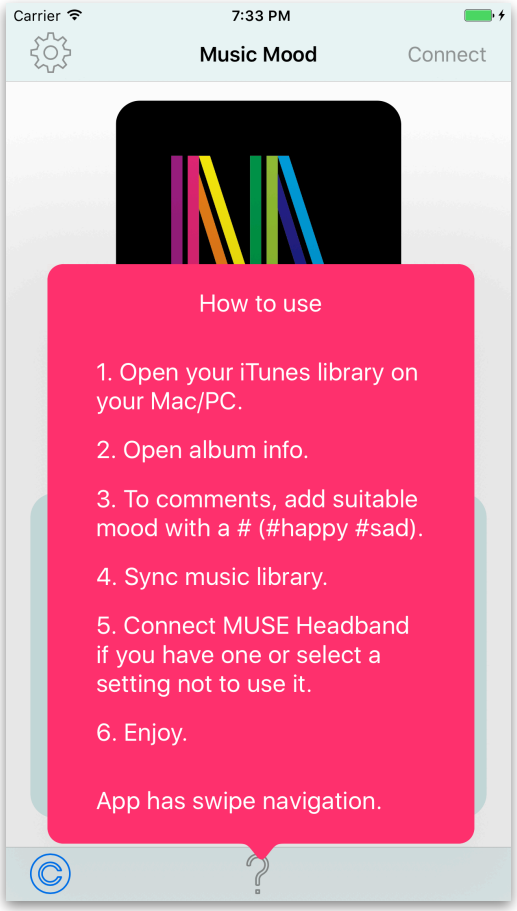


Figure 6-2. Instructions

Figure 6. Pop-ups

5.3. Settings

Moving on, if the user will press the cog-shaped button in the top-left corner, Settings ViewController will be presented (figure 7). Instead of being a popover akin to Legal info it will slide up form the bottom taking the whole screen. For now there are only two settings, but in the future there will be more, which requires an entire iPhone screen estate for clarity. However, right now the options are not put in UITableView, which is a common development practise.[25] As an alternative way of bringing it up users can just swipe up. It may be dismissed by either tapping the Dismiss button or swiping down.

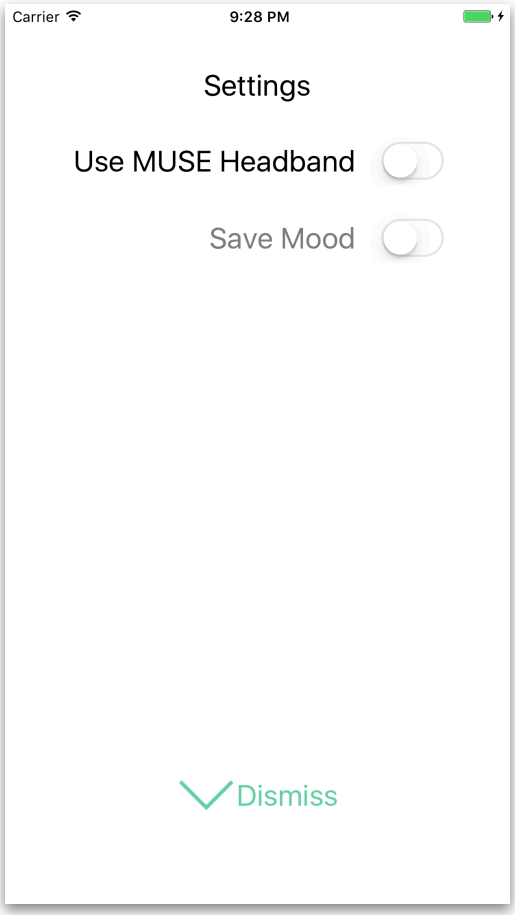


Figure 7. Settings menu

“Use MUSE Headband” is responsible for the song selection method. If turned on, the MoodPicker (figure 5-2) will not appear when My Mood button is pressed. In a way, it is done with intent to prohibit the users with headband from not using it if they have one.[26] Otherwise the user might instinctively rely on usage of MoodPicker.

Save Mood setting is temporarily disabled, because it uses extremely huge amount of resources, which slows down an iPhone to a halt. Apples Core Data is used for saving. [27][28][29] Saving will be implemented in the future when less performance intensive solution is available.

The Apple watch option will be added in the future to filter the songs based on the HeartBeat monitor.

5.4. Connection

The last ViewController in the interface is responsible for the connection of MUSE Headband (figure 8).[30] It can be reached either by pressing the Connect button in the top-right corner of the main menu (figure 5) or by swiping left anywhere on the screen while in the main menu. It can be dismissed by either swiping right from the edge or by pressing the automatically generate button in the top-left corner (figure 7). It is all part of the typical iOS navigation.

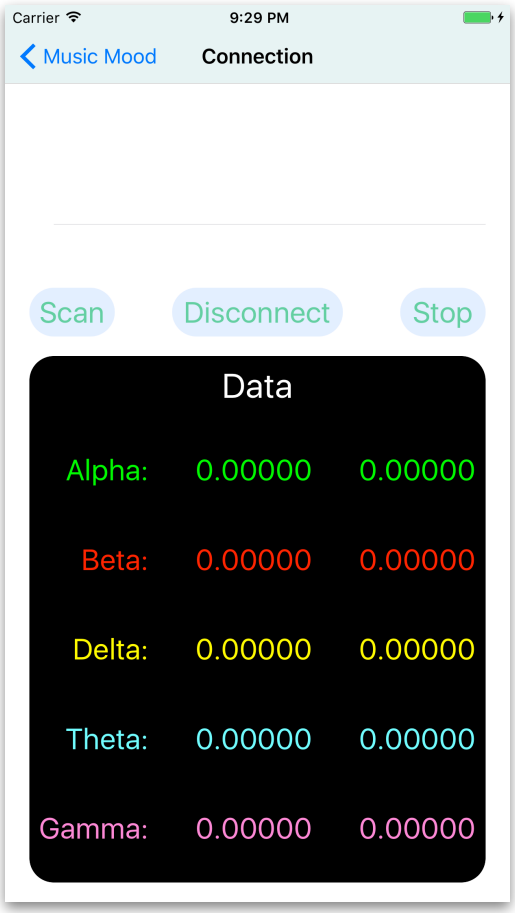


Figure 8. Menu for connecting the headband

At the moment the view is separated into two parts. The bottom shows the list of brain-waves and the data acquired from the headband, which is represented in value between 0 and 1. The first column shows the names, the second presents the current value and the last one gives the average over a certain time.

The top part consists of table view with buttons. The table is not seen very well in a figure 8 except for a single grey stripe above the buttons. Table outline is avoided for a cleaner looking interface.

Once MUSE is powered on it starts the pairing process. On the application's end Scan button displays all Bluetooth devices with Interaxon signature (that is all Muse headbands powered on). User can then select a headband to connect to it. Upon connecting the iPhone stops the search and the data exchange is started.

If the user wants to stop searching for the headband they can press Stop, otherwise MusicMood will stop searching after a set amount of time. If the user wishes to disconnect from the headband, all they need to do is press the Disconnect button.

6. INNER WORKINGS AND CODE

This section goes over inner workings of program hidden from the user. It describes the functionality of code not only linked to the interface, but also the code running in the background, which is responsible for selecting the mood and music.

This section will mainly go over the song filtering. Examples of data acquisition for the MUSE Headband can be found online. Hence there is no need to talk about the code that is explained elsewhere.

6.1. Main file (ViewController.swift)

This ViewController has import of UIKit that communicates with the interface as well as MediaPlayer. Another framework provided by Apple called AVPlayer exists and it is used for general playback of media items. However, we only need music from iTunes library of a user in order to simplify an app. Direct access to the system player allows to play music outside of the application by means of iOS Control Centre.[31]

New objects are created as instances of other classes. The MediaPlayer Controller has a variety of subclasses (figure 9). The variable “player” controls music playback, while “queueCollection” is an array storing songs, which are selected corresponding to user’s mood at the moment.[32][33] The “notificationCenter” is responsible for sending information regarding the playback status such as when song is playing or have reached an end. “defaults” is holding user settings. “data” will load the users current mood and “currentMoodValue” displays it. The “currentMoodValue” is the result of calculations achieved from values available in the “Data.swift” file.

```
import MediaPlayer

class ViewController: UIViewController, UIPickerViewDataSource,
UIPickerViewDelegate, MPMediaPickerControllerDelegate,
UIPopoverPresentationControllerDelegate {

    // Declaration for player
    var player = MPMusicPlayerController()
    //var album = MPMediaItemPropertyAlbumTitle
    var queueCollections = [MPMediaItemCollection]()

    // System declarations
    let notificationCenter = NotificationCenter.default
    let defaults = UserDefaults.standard

    // For determining the mood
    var data = Data()
    var currentMoodValue = Data.Mood.undefined.rawValue
```

Figure 9. Code for declaring the needed objects

Following that, the main file has @IBOutlet objects (figure 10) representing interface elements in the code. With Xcode we can for example connect the label seen in figure 5-1 as “undefined” to “currentMood” UILabel inside the code. Later we can change the mood.

```
    @IBOutlet weak var currentMood: UILabel!
    @IBOutlet var artist: UILabel!
    @IBOutlet var song: UILabel!


    let moodPickerValues = [Data.Mood.undefined.rawValue, Data.-
Mood.happy.rawValue, Data.Mood.sad.rawValue
//                          Data.Mood.melancholic.rawValue,
//                          Data.Mood.angry.rawValue
    ]


    @IBOutlet var artwork: UIImageView!
    @IBOutlet weak var moodPicker: UIPickerView!
    @IBOutlet var playPauseButton: UIButton!
```

Figure 10. Code for connecting interface elements with code and declaring mood values

Function “mood()” (figure 11) allows the user to choose their mood with the picker view as seen in figure 5-2. If the user has selected not to use MUSE Headband, they will instead be presented with a standard iOS notification pop-up that can be dismissed with a tap of a button.

Function “update()” is being called after the song started playing. This is done due to the nature of MediaPlayer and its required behaviour for this app. Each time the track is finished the mood needs to be determined again in order to find a corresponding track. [34] Due to this the list of songs is dynamic and does not exist in a proper form typical for iOS. Upon updating the player does not have a song to get info from. As such, the player is playing the song a bit before pausing it. That way the song is technically added to the queue and the displayed album title can be updated together with artist and album cover.[35] Buttons are also updated in order to follow this logic.

```
@IBAction func mood(_ sender: AnyObject) {
    if defaults.bool(forKey:
Settings.Setting.UseMuse.rawValue)==false{
        if moodPicker.isHidden {
            hideShow(objectA: self.artwork, objectB: self.mood-
Picker)
        } else {
            hideShow(objectA: self.moodPicker, objectB: self-
.artwork)
            player.play()
            update()
            player.pause()
            playPauseButton.setImage(#imageLiteral(resourceName:
"Play"), for: .normal)
        }
    } else {
        let alert = UIAlertController(title: "Picker View dis-
abled", message: "You can't pick the song when using MUSE Head-
band. It will do it for you.", preferredStyle: UIAlertCon-
trollerStyle.alert)

        alert.addAction(UIAlertAction(title: "ok", style:
UIAlertActionStyle.default, handler: nil))

        self.present(alert, animated: true, completion: nil)
    }
}
```

Figure 11. Code for selecting mood

The main in file also has a “playPause()” function that starts and pauses music depending on user selection. “update()” function changes the details of music track visible to user. “previous()” switches to the previous track in the queue. “next()” is atypical for this kind of applications as instead of moving up the queue, it dynamically modifies it by adding another element (song).

The core algorithm for song selection is inside the “runMediaLibraryQuery()” function (figure 12). The current solution is very CPU intensive and because of that it runs only between the songs.[36] Despite that, it is rather fast and does not bog down the system too much. The delay between songs is negligible. Swift allows quick sorting of individual songs by their artists or genre, but not by comments, because for now Apple have decided not to implement this functionality. The official statement says that this type of

filtering is not necessary as comments for songs are changed by user and are not consistent enough.

In the beginning the function checks whether the MUSE Headband is used. Values of brainwaves are updated in the cyclical manner inside “Data.swift”. If the headband is enabled by the user, the current mood is updated based on those values.

The system is based on song comments with hashtags (#happy #sad). The user is required to define it themselves, except for in some rare instances when a song already comes with predefined values. If the songs has the desired comment, for example the user is happy and the song has the corresponding hashtag, it is added to the query.[37][38][39] The song is picked randomly from all suitable songs.

```
func runMediaLibraryQuery() {
    if defaults.bool(forKey: Settings.Setting.UseMuse.rawValue)
{
        data.determineMood()
        currentMood.text = Data.CMV.currentMoodValue
    }

// Get all songs from the library. Needs to be run in sequence,
so asynchronous solution does not work
    let query = MPMediaQuery.songs()
    var queue = [MPMediaItemCollection]()

    // Filter the songs for chosen mood by using the comments
    if let collections = query.collections {
        for collection in collections {
            if let representativeTitle = collection.representa-
tiveItem!.title, let comment =
collection.representativeItem!.comments {
                switch Data.CMV.currentMoodValue {
                case Data.Mood.happy.rawValue:
                    if comment.lowercased().range(of: "#\(Data.-
Mood.happy.rawValue)") != nil {
                        print("Title: \(representativeTitle)
comment: \(comment)")
                        print(Data.Mood.happy.rawValue)
                        queue.append(collection)
                    }

                    case Data.Mood.sad.rawValue:
                        if comment.lowercased().range(of: "#\
(Data.Mood.sad.rawValue)") != nil {
                            print("Title: \(representativeTitle)
comment: \(comment)")
                            print(Data.Mood.sad.rawValue)
                            queue.append(collection)
                        }

                    case Data.Mood.undefined.rawValue:
                        queue.append(collection)

                    default:
                        print("Bizarre case!")
                        queue.append(collection)
                    }
                }
            }

            if queue.count != 0 {
                let randomIndex =
Int(arc4random_uniform(UInt32(queue.count)))
                queueCollections.append(queue[randomIndex])
                self.player.setQueue(with: queue[randomIndex])
            } else {
```

```
                // Need to use var, because sometimes it is be-
ing changed for some reason
                var alert = UIAlertController(title: "No songs
found", message: "There are no songs with \(self.currentMoodVal-
ue) mood", preferredStyle: UIAlertControllerStyle.alert)
                alert.addAction(UIAlertAction(title: "ok",
style: UIAlertActionStyle.default, handler: nil))
                self.present(alert, animated: true, completion:
nil)
            }
        }
    }
```

Figure 12. Code for filtering songs

The player is configured to send notifications regarding its playback state and the song filtering occurs when the track fully stops.[40] Due to this "trackFinished()" function is called only between songs. It would be unreasonable to change the songs during their playback as it may lead to unpleasant listening experience.

The cycle of changing songs can run indefinitely until it is stopped or paused. If the app is closed down, the music will play till the end of current song, but not more as the queue may not be extended. However, the app can still perform its function in the background, while another app is in operation, unless that app utilises sound playback.

6.2. Determining mood (Data.swift)

"Data.swift" is responsible for picking the mood based on values received from MUSE Headband. At the moment value-ranges (figure 13) are arbitrary due to lack of defining research.

```
// This struct determines the mood by the range of brainwave
values
struct WaveRange {
    // Undefined
    static let alphaDefault: Double = 0.0
    static let betaDefault: Double = 0.0
    static let deltaDefault: Double = 0.0
    static let thetaDefault: Double = 0.0
    static let gammaDefault: Double = 0.0

    // Sad mood
    static let alphaRange1000: Range = 0.0..<0.5
    static let betaRange1000: Range = 0.0..<1.0
    static let deltaRange1000: Range = 0.0..<1.0
    static let thetaRange1000: Range = 0.0..<1.0
    static let gammaRange1000: Range = 0.0..<1.0

    // Happy mood
    static let alphaRange5000: Range = 0.5..<1.0
    static let betaRange5000: Range = 0.0..<1.0
    static let deltaRange5000: Range = 0.0..<1.0
    static let thetaRange5000: Range = 0.0..<1.0
    static let gammaRange5000: Range = 0.0..<1.0
}
```

Figure 13. Code for declaring the brainwave values for moods

The “currentMoodValue” (figure 14) variable is put inside a struct and made static in order to allow the main file access the calculated value. The “Waves” variable will hold all the brainwaves received from the headband. The “WavesAverage” variable will hold average value of previous values. Everything is declared with zeros in the beginning.

```
struct CMV {
    static var currentMoodValue = Mood.undefined.rawValue
}

static var Waves = [
    "alpha": Array(repeating: 0.0, count: maxData),
    "beta": Array(repeating: 0.0, count: maxData),
    "delta": Array(repeating: 0.0, count: maxData),
    "theta": Array(repeating: 0.0, count: maxData),
    "gamma": Array(repeating: 0.0, count: maxData)
]

static var WavesAverage: [String: Double] = ["alpha": 0.0,
"beta": 0.0, "delta": 0.0, "theta": 0.0, "gamma": 0.0]
```

Figure 14. Code for average value of brainwaves

The “determineMood()” function (figure 15) compares the data received from the headband with the values for mood using switch.[41][42][43] It is possible to add many other ranges for other feelings.[44][45] For now it is hugely approximate. In case something goes wrong, there is a default clause, which leads to the undefined value.

Before comparing, we calculate the average using the “reduce()” function. “for” loop iterates over all 5 brainwaves.[46]

```
func determineMood() {
        for (wave, value) in Data.Waves {
            Data.WavesAverage[wave] = value.reduce(0.0) {
                return $0 + $1/Double(value.count)
            }
            print("Average of \(wave): \
(Data.WavesAverage[wave]!)")
        }


        // Switching over the values of the headband and deter-
mining the mood
        switch (Data.WavesAverage["alpha"],
Data.WavesAverage["beta"], Data.WavesAverage["delta"], Da-
ta.WavesAverage["theta"], Data.WavesAverage["gamma"]) {

        // Undefined. Default
        case (WaveRange.alphaDefault?, WaveRange.betaDefault?,
WaveRange.deltaDefault?, WaveRange.deltaDefault?, WaveRange.gam-
maDefault?):
            CMV.currentMoodValue = Mood.undefined.rawValue

        // Sad mood
        case (WaveRange.alphaRange1000?, WaveRange.beta-
Range1000?, WaveRange.deltaRange1000?,
WaveRange.deltaRange1000?, WaveRange.gammaRange1000?):
            CMV.currentMoodValue = Mood.sad.rawValue

        // Happy mood
        case (WaveRange.alphaRange5000?, WaveRange.beta-
Range5000?, WaveRange.deltaRange5000?,
WaveRange.deltaRange5000?, WaveRange.gammaRange5000?):
            CMV.currentMoodValue = Mood.happy.rawValue

        default:
            print("Something went really wrong!")
            CMV.currentMoodValue = Mood.undefined.rawValue
        }
    }
```

Figure 15. Code for determining the mood

7. CONCLUSION

Despite many hurdles this project has finally seen the light of day. It can however not be commercially released due to licensing issues with Interaxon Inc, who created the MUSE Headband used in this application. Even the beta version of the app unfortunately will not be available on iOS App Store anytime in near future.

After all, the app works and performs its main functionality of playing music depending on the feelings. As a prototype, it is a success, albeit it can use more refinement both aesthetically and functionally. MusicMood will be updated in the future to reach the standards of other exemplary iOS applications.

All documents and files related to this project will be available on the internet for future revisions and improvements of MusicMood and other programs based on the principle of filtering music depending on users feelings. The software will be released under open license.

8. REFERENCES

1. Apple Developer portal
https://developer.apple.com

2. MUSE Developer portal
http://developer.choosemuse.com

3. Junction Hackathon 2015
https://en.wikipedia.org/wiki/Junction_(hackathon)

4. MUSE Brain-sensing headband
http://www.choosemuse.com

5. Neurowear website
http://neurowear.com

6. Image reference (mico brain-sensing headset)
http://neurowear.com/projects_detail/mico.html

7. Neural Oscillation, from Wikipedia the free encyclopaedia
https://en.wikipedia.org/wiki/Neural_oscillation

8. Fu-Chien Kao, Shinping R. Wang and Yu-Jung Chang. Brainwaves Analysis of Positive and Negative Emotions, from Department of Computer Science & Information Engineering Da-Yeh University (2015)
http://www.wseas.org/multimedia/journals/information/2015/a405709-517.pdf

9. Image reference (MUSE Brain-sensing headband)
http://www.choosemuse.com/what-does-muse-measure/

10. Xcode/Swift 'filename used twice' build error (question asked by user RobertyBob on StackOverflow)
http://stackoverflow.com/questions/34838184/xcode-swift-filename-used-twice-build-error

11. Xcode won't recognise a new Swift class (question asked by user Sam J on StackOverflow)
http://stackoverflow.com/questions/30146269/xcode-wont-recognize-a-new-swift-class

12. Instance member cannot be used on type (question asked by user Aderstedt on StackOverflow)
http://stackoverflow.com/questions/32351343/instance-member-cannot-be-used-on-type

13. How to create the Upload File for Application Loader? (question asked by user Ohad Regev on StackOverflow)

http://stackoverflow.com/questions/5937660/how-to-create-the-upload-file-for-application-loader

14. Certificate has either expired or has been revoked (question asked by user "user6218736" on StackOverflow)

http://stackoverflow.com/questions/36689116/certificate-has-either-expired-or-has-been-revoked

15. Ben Dodson. Media Library privacy flaw fixed in iOS 10 (2016)

https://bendodson.com/weblog/2016/08/02/media-library-privacy-flaw-fixed-in-ios-10/

16. nackpan. [iOS][Swift]MPMediaQueryを使って曲を絞り込む translation jp-en: Use MPMediaQuery to narrow down songs, from nackpan Blog (2015)

http://nackpan.net/blog/2015/09/16/ios-swift-mpmediaquery/

17. Changing image of UIButton via click - Xcode 6 Swift (question asked by user MitchKrendell on StackOverflow)

http://stackoverflow.com/questions/27025759/changing-image-of-uibutton-via-click-xcode-6-swift

18. How to add animation while changing the hidden mode of a uiview? (question asked by user Sanchit Paurush on StackOverflow)

http://stackoverflow.com/questions/6177393/how-to-add-animation-while-changing-the-hidden-mode-of-a-uiview

19. Bharathi. iOS UIPickerView Example using Swift, from Source Freeze (2015)

http://sourcefreeze.com/ios-uipickerview-example-using-swift/

20. Swift: How to set a default value of a UIPickerView with three components in Swift? (question asked by user KML on StackOverflow)

http://stackoverflow.com/questions/25917693/swift-how-to-set-a-default-value-of-a-uipickerview-with-three-components-in-swi

21. Nick Hanan. How to Create a UIColor in Swift (2016)

http://www.codingexplorer.com/create-uicolor-swift/

22. Use storyboard to mask UIView and give rounded corners? (question asked by user Crashalot on StackOverflow)

http://stackoverflow.com/questions/34215320/use-storyboard-to-mask-uiview-and-give-rounded-corners

23. Scale text label by screen size (question asked by user Jeffrey on StackOverflow)

http://stackoverflow.com/questions/29308941/scale-text-label-by-screen-size

24. How to completely colorise UIPopoverPresentationController background color? (question asked by user Mario on StackOverflow)

http://stackoverflow.com/questions/31906070/how-to-completely-colorize-uipopoverpresentationcontroller-background-color

25. How to use UIScrollView in Storyboard (question asked by user Alex Reynolds on StackOverflow)

http://stackoverflow.com/questions/12905568/how-do-i-use-uiscrollview-in-storyboard

26. Detect first launch of iOS app [duplicate] (question asked by user Julian Stellaard on StackOverflow)

http://stackoverflow.com/questions/27208103/swift-detect-first-launch

27. user Takumu Uyama - sasurai_usagi3. Swift3.0のcore dataでCRUD! translation jp-en: CRUD with core data of Swift 3.0! , from Qiita (2016)

http://qiita.com/sasurai_usagi3/items/e47fd82c3cb116c8e953

28. SWIFT: Updating an existing core data object (question asked by user GabrielMSC on StackOverflow)

http://stackoverflow.com/questions/37489634/swift-updating-an-existing-core-data-object

29. Adding Core Data to existing iPhone project (question asked by user swalkner on StackOverflow)

http://stackoverflow.com/questions/2032818/adding-core-data-to-existing-iphone-project

30. Sergey Kargopolov. Customize UINavigationBar appearance in Swift (2015)

http://swiftdeveloperblog.com/customize-uinavigationbar-appearance-in-swift/

31. user sawapi. Swiftで音楽を再生 translation jp-en: Play music with Swift, from Qiita (2014)

http://qiita.com/sawapi/items/e08ab4f56f7e4684defd

32. Pick a random element from an array (question asked by user Fela Winkelmolen on StackOverflow)

http://stackoverflow.com/questions/24003191/pick-a-random-element-from-an-array

33. (Swift) How to create global array? (question asked by user Rashwan L on StackOverflow)

http://stackoverflow.com/questions/34885682/swift-how-to-create-global-array

34. Swift - Detect music playing, whether it's Spotify or iTunes (question asked by user zantuja on StackOverflow)
http://stackoverflow.com/questions/38385464/swift-detect-music-playing-whether-its-spotify-or-itunes

35. In Swift, how do you check if an object (AnyObject) is a String? (question asked by user shim on StackOverflow)
http://stackoverflow.com/questions/26521583/in-swift-how-do-you-check-if-an-object-anyobject-is-a-string

36. Gabriel Thedoropoulos. Grand Central Dispatch (GCD) and Dispatch Queues in Swift 3, from APPCODA (2016)
http://www.appcoda.com/grand-central-dispatch/

37. Paul Solt. How to search for a Character in a String with Swift 2, from super easy apps (2015)
http://supereasyapps.com/blog/2015/8/7/how-to-search-for-a-character-in-a-string-with-swift-2

38. Austin Zheng. Swift 2: Control Flow Pattern Matching Examples (2015)
http://austinzheng.com/2015/09/23/pmatch-control-flow/

39. Austin Zheng. Swift's pattern-matching switch statement (2014)
http://austinzheng.com/2014/12/16/swift-pattern-matching-switch/

40. MPMusicPlayerController not posting notifications? (question asked by user Ben Collins on StackOverflow)
http://stackoverflow.com/questions/3904746/mpmusicplayercontroller-not-posting-notifications

41. user hachinobu. Swiftの列挙型(enum)おさらい translation jp-en: Swift's enumerated type (enum) review, from Qiita (2017)
http://qiita.com/hachinobu/items/392c96820588d1c03b0c

42. Switch case on enum type (question asked by user NikMos on StackOverflow)
http://stackoverflow.com/questions/33910829/switch-case-on-enum-type

43. user akatsuki174. SwiftにおけるSwitchまとめ translation jp-en: Switch in Swift Summary, from Qiita (2015)
http://qiita.com/akatsuki174/items/2720ebc369a6c1d9f629

44. Swift: Multiple intervals in single switch-case using tuple (question asked by user iiFreeman on StackOverflow)

http://stackoverflow.com/questions/25165123/swift-multiple-intervals-in-single-switch-case-using-tuple

45. What are “intervals” in Swift ranges? (question asked by user Chéyo on StackOverflow)

http://stackoverflow.com/questions/25308978/what-are-intervals-in-swift-ranges

46. How to write method to calculate average in Swift-Playground (question asked by user “user1898829” on StackOverflow)

http://stackoverflow.com/questions/24117119/how-to-write-method-to-calculate-average-in-swift-playground