|  |  |
|---|---|
| Project Acronym: | **MusicBricks** |
| Project Full Title: | **Musical Building Blocks for Digital Makers and Content Creators** |
| Grant Agreement: | **N°644871** |
| Project Duration: | **18 months (Jan. 2015 - June. 2016)** |

# D3.3 Final release of API

|  |  |
|---|---|
| Deliverable Status: | **Final** |
| File Name: | **MusicBricks_D3.3.pdf** |
| Due Date: | **31 December 2015 (M12)** |
| Submission Date: | **4 January 2016 (M13)** |
| Dissemination Level: | **Public** |
| Task Leader: | **Universitat Pompeu Fabra** |

***Authors:*** *Jordi Janer (UPF), Thomas Lidy (TUW), Jakob Abeßer (IDMT), Sascha Grollmisch (IDMT), Alexander Schindler (TUW), Frederic Font (UPF).*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

The MusicBricks project consortium is composed of:

| | | |
|---|---|---|
| **SO** | Sigma Orionis | France |
| **STROMATOLITE** | Stromatolite Ltd | United Kingdom |
| **IRCAM** | Institut de Recherche et de Coordination Acoustique Musique | France |
| **UPF** | Universitat Pompeu Fabra | Spain |
| **Fraunhofer** | Fraunhofer-Gesellschaft zur Foerderung der Angewandten Forschung E.V | Germany |
| **TU WIEN** | Technische Universitaet Wien | Austria |

## Revision Control

| Version | Author | Date | Status |
|---|---|---|---|
| 0.1 | Jordi Janer (UPF) | Dec 18, 2015 | Initial Draft with contributions from: Thomas Lidy (TUW), Jakob Abeßer (IDMT), Sascha Grollmisch (IDMT), Alexander Schindler (TUW), Frederic Font (UPF) |
| 0.2 | Marta Arniani (SIGMA) | Dec. 21, 2015 | Review |
| 0.3 | Jordi Janer (UPF) | Dec 22, 2015 | Final Draft |
| 1.0 | Marta Arniani (SIGMA) | Jan. 4, 2016 | Final Draft reviewed and submission to the EC |

## Table of Contents

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

## Executive summary

The present document is a deliverable of the MusicBricks project, funded by the European Commission's Directorate-General for Communications Networks, Content & Technology (DG CONNECT), under its Horizon 2020 research and innovation programme.

This document introduces the final release of the #MusicBricks technology portfolio provided by the consortium research partners. In particular, in Work Package 3 we deployed these technologies in form of APIs for an easy integration by creative users. During the reported period since the previous deliverable (D3.2), these technologies have been refined, updated and extended based on the feedback gathered in the Creative Testbeds. All technologies come along with technical documentation. The funcitonality and usability of these technologies have been demonstrated in the Creative testbeds projects. Additionally the technologies are planned to be further integrated in several projects during the Industrial Testbeds towards the end of the project (M18).

As a result, with the Final Release of the APIs, #MusicBricks provides to the creative community a comprehensive portfolio of music algorithms: Melody Extraction, rhythm and Timbre analysis, Gesture Sensors, Search Similar Sound, Freesound, Real-time Pitch Detection, Music Transcriber, Real-time Onset Description, Spectral Synthesis and Melody replacement.

One of the goals of this work package was to facilitate the combination of different technologies from the portfolio. In this report we introduce two examples of new demo applications to foster the creation of novel applications by the creative community using these tools.

# 1. Introduction

This document is elaborated upon the previous deliverable D3.1 and D3.2. It provides a thorough description of the technologies included in the *Final Version of the APIs*. This description includes details about the current technical status of each technology in terms of readiness, documentation, installation and licensing options.

## 1.1 Context

During the reported period (July to December 2015), technology partners have further refined and extended the portfolio of #MusicBricks technologies. Additionally, partners have jointly collaborated in developing demo applications with the goal of demonstrating how novel applications can be easily built with the #MusicBricks portfolio.

The release of the Final Version of the APIs (December 2015) concludes the tasks of the Work-package 3 (API).

## 1.2 Objectives

As specified in the Description of Work document, the goal of this deliverable was to ensure that the provided technologies can be used in the Industry Testbeds (*First release for use in Industry Testbeds*), and are mature to be further used in a potential future exploitation. These objectives are principally framed in the Task 3.3 Wrapping existing Tools and technologies together. Another goal is to demonstrate how the provided #MusicBricks portfolio can be easily used by combining different tools. In this deliverable we report the activities related to Task3.2 Creating Specific tool Combinations for advanced features, which resulted in two demo applications.

The objectives of the document are addressed in the next sections:

1. Updating the MusicBricks portfolio
   a. Technology updates from the creative testbeds
   b. New added technologies from external partners
2. Description of the Final MusicBricks portfolio
   a. API reference, current status and licensing conditions for us in the industry testbeds
3. Documentation and installation
   a. Comparative table and Links
4. Combining and Extending MusicBricks
   a. Example application combining MusicBricks

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

## 2. Updating the #MusicBricks Portfolio

During this 6-month period, the technology partners (TU Wien, Fraunhofer IDMT and MTG-UPF) have refined and extended their technologies and APIs to build the final #MusicBricks technology portfolio.

From the past Creative Testbed, we identified a number of new functionalities or need of more detailed documentation in order to better promote the use of these technologies among creative users. For example, a real-time implementations and audio transformation and synthesis were a clear demand by users, not explicitly provided in the initial version of the API.

In the next table we summarizes the work carried out by the technology partners in this Work Package:

| | |
|---|---|
| **Fraunhofer:** | MusicBricks Transcriber: added new output formats .mxl (zipped MusicXML) & .abc (ABC notation format)<br>MusicBricks Transcriber: added automatic chord detection<br>MusicBricks Transcriber: added python methods for parsing generated xml files<br><br>New MusicBrick - Real-Time Pitch-Shifting & Time-Stretching: Library for real time pitch shifting or time stretching of audio data<br>New MusicBrick - Goatify: Replace the main melody in a song with the a sample, sample will be pitched according to the melody |
| **TUW** | rp_extract: added new input audio file formats (m4a, aiff)<br>rp_classify: added completely new function to train and predict high-level music concepts (genre, mood, etc.)<br><br>New MusicBrick: Search by Sound: Fast audio analysis engine, Music Similarity Search + REST API for online use<br>New MusicBrick: Sonarflow: Visual music discovery app for iOS and Android devices (both are completely open source) |
| **UPF** | MusicBrick Onset Description: adding new functionalities to real-time version<br>MusicBrick Melody Extraction: providing python tutorials as ready to use documentation<br><br>New MusicBrick Spectral Synthesis API: adding new synthesis functionalities for concatenative synthesis. |

## 3. Final Release of #MusicBricks APIs

### 3.1 Introduction

In this section we describe the technologies and APIs provided by each research partner as part of the *Final Version of the API*. This table is an extended and updated version of the table provided in the deliverable document D3.2.

Lists of APIs:

| | |
|---|---|
| | **Melody extraction**<br>This module includes a number of pitch tracking and melody transcription algorithms implemented in the Essentia library. We will demonstrate their differences in performance and give suggestions on which algorithm to use with respect to instrumentation, genre and task on a number of practical examples. Applications include visualization of predominant melody, pitch tracking, tuning rating, source separation. |
| | **Real-time Onset description**<br>This module allows to detect onsets in real-time and provide a number of audio descriptors. It is part of **essentiaRT~,** a real-time subset of Essentia (MTG's open-source C++ library for audio analysis) implemented as an **external for Pd and Max/MSP**. It provides a number of features to slice and provide on-the-fly descriptors for classification of audio in real-time. A number of extractors analyse instantaneous features like the onset strength, the spectral centroid and the MFCC's over a fixed-size window of 2048 points, after an onset is reported. Furthermore, *essentiaRT~* is able to perform estimations on larger time-frames of user-defined lengths, and to report finer descriptions in terms of noisiness, f0, temporal centroid and loudness. |
| | **Rhythm and Timbre Analysis**<br>This is a library that processes audio data as input and analyzes the spectral rhythmic and timbral information in the audio to describe its acoustic content. It captures rhythmic and timbral features which can be stored or directly processed to compute acoustic similarity between two audio segments, **find similar sounding songs (or song segments)**, create **playlists of music of a certain style**, **detect the genre** of a song, make **music recommendations** and much more. Depending on the needs, a range of audio features is available: Rhythm Patterns, Rhythm Histograms (i.e. a rough BPM peak histogram), Spectrum Descriptors and more. The library is available for Python, Matlab and Java.<br><br>In the final release, as of December 2015, a classification feature for both single-label and multi-label classification (recognition of genre, mood, etc.) has been added (rp_classify) and evaluated, which also supports import and export of (pre-)annotated csv file for training and predicted output. |

**Search by Sound Music Similarity**

The Search by Sound online system is based on the Rhythm and Timbre Analysis (see above) and provides a system which can be used via a **REST Web API (called SMINT API)** to upload, find and match acoustically similar songs in terms of rhythm and timbre – without the need to install any prerequisite or run the analysis on your own. It can be used with your own custom music dataset or the readily available content from freemusicarchive.org that was already pre-analyzed by rhythm and timbre, to find music matching a particular rhythm or timbre from that archive.

### *3.1.1 Sonarflow*

The world's first visual music discovery app for iPad, iPhone and Android is now available open-source for hacking and extending it. It features a slick UI for browsing music by zooming into a colourful world of bubbles which represent genres, artists or moods and allows discovering new music online from various sources. It is available for iOS and Android, with APIs connecting to 7digital, last.fm, Youtube, Spotify, etc. Demo videos at http://www.sonarflow.com , Demo App in Google Play Store. , Full Source Code: https://github.com/spectralmind

**MusicBricks Transcriber**

The MusicBricks Transcriber (Melody, Chord & Bass Transcription + Beat & Key & Tempo Estimation) provided by Fraunhofer IDMT is an executable that allows to transcribe the main melody, chords and bass line of a given audio file. Also, the beat times, the key, and the average tempo are estimated. The results can be provided as MIDI, MusicXML, ABC Notation, or plain XML files. In addition, a Python wrapper is included to further process the analysis results.

**Real-time Pitch Detection**

The real-time pitch detection allows to estimate the predominant melody notes (monophonic) or multiple notes (polyphonic) from a consecutive audio sample blocks. This allows to transcribe the currently played / sung note pitches from a recorded instrument / vocal performance. The monophonic version also estimates the exact fundamental frequency values. Typical applications are music games and music learning applications. Fraunhofer IDMT provides a C++ library as well as sample projects that show how to include the functionality.

**Real-time Pitch-Shifting & Time-Stretching**

The real-time pitch shifting library allows changing the pitch of audio material while keeping the tempo. It allows enabled changing the tempo without changing its pitch. Typical applications are music games and music learning applications as well real time performances. Fraunhofer IDMT provides a C++ library as well as sample projects that show how to include the functionality.

| | |
|---|---|
| | **Goatify**<br>The Goatify tool provided by Fraunhofer IDMT is an executable that automatically replaces the main melody in a song with a given sample. Therefore the main melody is extracted and removed from the song. Then the sample is placed and pitched according to the melody notes in the song. For proper pitching of the sample, the pitch of the sample itself is extracted beforehand. The tool is delivered with free sound samples (goat, etc.) from www.freesound.org for direct use. |
| | **Freesound API**<br>Freesound (www.freesound.org) is a state-of-the-art online collaborative audio database that contains over 200K Creative Commons licensed sound samples. All these sounds are annotated with user-provided free-form tags and textual descriptions that enable text-based retrieval. Content-based audio features are also extracted from sound samples to provide sound similarity search. Users can browse, search, and retrieve information about the sounds, can find similar sounds to a given target (based on content analysis) and retrieve automatically extracted features from audio files; as well as perform advanced queries combining content analysis features and other metadata (tags, etc…). The Freesound API will provide access to a RESTful API with API Clients in Python, Javascript, Objective-C. |
| | **Spectral Synthesis API**<br>Spectral Synthesis API is a new set of synthesis algorithms integrated in the Essentia library (www.essentia.upf.edu). Its purpose is to support the transformation of audio samples in the context of audio mosaicing, or concatenative synthesis. Specifically, the integrated algorithms are: Sinusoidal Model Analysis / Synthesis; Sinusoidal Plus Residual Model Analysis / Synthesis; Sinusoidal Plus Stochastic Model Analysis / Synthesis; and Harmonic Plus Stochastic Model Analysis / Synthesis. |

Next sections provide detailed information of the different technologies provided by research partners.

## 3.2 MusicBricks Transcriber - Melody, Chord & Bass Transcription, Beat & Key & Tempo Detection API (by Fraunhofer IDMT)

### *3.2.1 Description*

The first binary executable allows performing different music analysis tasks for a given WAV or MP3 audio file of 15 minutes length maximum:

- **Bass Transcription**
    - This algorithm transcribes the bass line of a song as a sequence of note events
- **Melody Transcription**

- This algorithm transcribes the predominant melody of a song as a sequence of note events
- **Chord Transcription**
    - This algorithm transcribes the basic chords of a song including their onsets
- **Key**
    - This algorithm estimates the most likely major or minor key of a given song
- **Beat Grid**
    - This algorithm estimates the metric structure of a given song and returns the beat times and beat numbers
- **Tempo**
    - This algorithm estimates the average tempo of a song in beats per minute (bpm)

### *3.2.2 Platforms*

- Windows
- Mac OSX
- Linux

### *3.2.3 Interface Description*

The executable allows selecting the **analysis tasks** to be performed using **commandline parameters**.
The analysis results (transcription, beat grid, etc.) can be exported as

- **MIDI** (melody & bass transcription results)
- **MusicXML** (melody & bass transcription result)
- **MXL** (zipped MusicXML)
- **ABC** (melody & bass transcription result)
- **XML** (easy-to-read format with all results)

for further use in other programs.

Usage example:
**MusicBricksIDMTTranscriber.exe -i** *input audio file (wave or mp3)* **-o** *output name* **--xml --midi --melody**

### *3.2.4 Readiness level and IPR*

- Stable and reliable. Tested in scientific and commercial settings throughout many years.
- Overview and documentation will be included
- Royalty-free, limited, non-transferable license

## 3.3 Real-time Pitch Detection - Monophonic & Polyphonic API (by Fraunhofer IDMT)

### *3.3.1 Description*

The real-time melody transcription technology will be provided as dynamic libraries (DLL / SO files) to be included into the user's programs.

### *3.3.2 Platforms*

- Windows, Mac OSX, Linux
    - Sample-Code in C++ will be provided
- iOS, Android
    - A sample application for both operating systems will be provided as demo

#### 3.3.3 *Interface Description*

The interface will provide methods to

- switch between the pitch detection version (monophonic, polyphonic)
- create / delete a pitch detection object
- set reference frequencies for polyphonic pitch detection (to limit the search range)
- get one / multiple detected fundamental frequencies for a given sample buffer

#### 3.3.4 *Readiness level and IPR*

- Stable and reliable. Tested in scientific and commercial settings throughout many years.
- Overview, documentation, and sample application will be included
- Royalty-free, limited, non-transferable license

---

## 3.4 Real-time Pitch-Shifting & Time-Stretching API (by Fraunhofer IDMT)

#### 3.4.1 *Description*

The pitch-shifting & time-stretching technology will be provided as dynamic libraries (DLL / SO files) to be included into the user's programs.

#### 3.4.2 *Platforms*

- Windows, Mac OSX, Linux
    - Sample-Code in C++ will be provided
- iOS, Android

#### 3.4.3 *Interface Description*

The interface will provide methods to

- create / delete processing object
- set pitch shifting by factor or semitones
- set time stretching by factor from half to double tempo
- pitch-shift / time-stretch a given sample buffer

#### 3.3.4 *Readiness level and IPR*

- Stable and reliable. Tested in scientific and commercial settings throughout many years.
- Overview, documentation, and sample application will be included
- Royalty-free, limited, non-transferable license

## 3.5 Goatify API (by Fraunhofer IDMT)

### *3.5.1 Description*

The binary executable removes the main melody from WAV or MP3 audio files and replaces it with a given sample. The result is saved as WAV file. If the midi pitch of the sample is not set, it will be automatically extracted as well.

### *3.5.2 Platforms*

- Windows
- Mac OSX
- Linux

### *3.5.3 Interface Description*

The executable allows selecting the input audio file and the sample to be using **commandline parameters**. Additionally the midi pitch of the sample can be set.

Usage example:
**Goatify_cmd.exe -i** *input audio file (wave or mp3)* **-o** *ouput audio file (wave)* **-s** *sample file* -**m** *midi pitch of sample*

### *3.2.4 Readiness level and IPR*

- Stable and reliable. The used technologies are tested in scientific and commercial settings throughout many years.
- Overview and documentation will be included
- Royalty-free, limited, non-transferable license

## 3.6 Rhythmic and Timbre - RP extract: Audio Feature Extraction (by TU Wien)

### *3.6.1 Description*

RP_extract is a library that processes audio data as input and analyzes the spectral rhythmic and timbral information in the audio to create different audio descriptors (a.k.a. features).

These extracted audio descriptors can be used to find similar sounding songs, create automatic playlists, make music recommendations, categorize music etc. Depending on the needs, a range of audio features is available:

**Rhythmic descriptors:**

- RP: captures Rhythm Patterns (see picture at the right - audible frequency vs. repetition frequency) and is able to find songs with similar rhythm
- RH: Rhythm Histogram: simplified rhythm descriptor (roughly containing rhythmic strength for different bpm values)

**Timbral descriptors:**

- SSD: Statistical Spectrum Descriptor: describing timbral aspects of the audio Sonogram; by that it is possible to find songs with similar timbral characteristics
- MVD: The Modulation Frequency Variance Descriptor measures variations over critical audible frequency bands for a specific rhythmic repetitions (derived from a rhythm pattern).

**Temporal descriptors:**

- TSSD: Timbral variations over time (based on SSD)
- TRH: Temporal Rhythm Histograms - rhythmic variations over time (based on RH)

Features can also be combined (e.g. to cover rhythmic **and** timbral aspects).

The output is a corresponding list of feature vectors for the supplied input data, in the following formats:

- Matlab: SOMlib file format (an extended CSV file format with headers)
- Java: SOMlib file format or Weka ARFF
- Python: CSV or Python dictionary (with the feature abbreviations as dictionary keys and the feature vectors as values)

In the final version of the API, the following new capabilities were added, due to needs identified from the Creative Testbeds and the Industry Testbeds:

**New Input file formats:**
The Rhythm Timbre library can now handle:

- WAV
- MP3
- M4A (new)
- AIF(F) files (new)

Furthermore, a completely new program function was added to classify and predict high-level music concepts such as genre, mood etc.:

**rp_classify**

which supports 3 modes:

- **train** a new model: providing audio files and annotated categories, a model is built
- **cross-validate:** gives a performance self-evaluation of the trained model
- **classify**: predict the categories (genres, moods) on new music

This was due to popular request to directly detect a genre or mood in music, instead of working with the high-dimensional low-level features.

It supports single and multi-category prediction and import and export of annotations in CSV files.

**This complete library is available free and open source:**

Python: https://github.com/tuwien-musicir/rp_extract
Matlab or Java: http://ifs.tuwien.ac.at/mir/downloads.html

**Documentation:**

Quick description: http://ifs.tuwien.ac.at/mir/audiofeatureextraction.html
Tutorial: http://ifs.tuwien.ac.at/mir/audiofeatureextraction/tutorial/RP_extract_Tutorial.html
Research paper: [1]

### *3.6.2 Platforms*

Implementations:

- Python
- Matlab
- Java

All platforms running Python or Matlab or Java are supported:

- Windows, Mac OS X, Linux

All versions include decoding functionality for MP3 compressed audio through an external program (mpg123, ffmpeg).

### *3.6.3 Interface Description*

**Function name: rp_extract**

- **data**: pcm signal data
- **samplerate**: signal sampling rate
- **extract_rp:** extract Rhythm Patterns features
- **extract_ssd:** extract Statistical Spectrum Descriptor
- **extract_sh:** extract Statistical Histograms
- **extract_tssd:** extract temporal Statistical Spectrum Descriptor
- **extract_rh:** extract Rhythm Histogram features
- **extract_trh:** extract temporal Rhythm Histogram features
- **extract_mvd:** extract modulation variance descriptor
- **resample:** do resampling before processing: 0 = no, > 0 = resampling frequency in Hz
- **skip_leadin_fadeout:** how many sample windows to skip at the beginning and the end
- **step_width:** >=1 each step_width'th sample window is analyzed
- **n_bark_bands:** 15 or 20 or 24 (for 11, 22 and 44 kHz audio respectively)
- **mod_ampl_limit:**
- **spectral_masking** use Spectral Masking
- **transform_db:** use transformation into DeciBel
- **transform_phon:** use transformation into Phon
- **transform_sone:** use transformation into Sone (requires transform_phon enabled)
- **fluctuation_strength_weighting:** apply Fluctuation Strength weighting curve

**Program name: rp_classify**

To be used and controlled via command line parameters:

- rp_classify.py -t input_path model_file
- rp_classify.py -t -c classfile input_path model_file
- rp_classify.py -t -m multiclassfile input_path model_file
- rp_classify.py -cv input_path model_file
- rp_classify.py input_path model_file output_filename

#### *3.6.4 Readiness level and IPR*

***Java version:***

- Version 0.7 available
- Not fully implemented and supported
- Free and open source (further development is encouraged)

***Python, Matlab versions:***

- Stable and reliable. Tested in scientific settings throughout many years.
- Download code and examples from GitHub: https://github.com/tuwien-musicir/rp_extract
- Overview and documentation provided on: http://ifs.tuwien.ac.at/mir/musicbricks
- Free and open source

***Python version:***

- Readme for quick setup and usage
- Tutorial provided (available as interactive iPython notebook)
- new classification library for prediction of genre or mood
- stable in 1.0
- ready for productive usage

(Note: Python version is recommended as it poses the ideal combination of open-source, readiness, ease of use and performance.)

---

## 3.7 Search by Sound: Music Similarity Retrieval API (by TU Wien / Spectralmind)

#### *3.7.1 Description*

The "Search by Sound" system can analyze music tracks and store their fingerprints (features) in a database. Via an API, one can query for similar sounding songs, using a combination of audio-feature based and meta-data query (e.g. "give me rhythmically similar songs from the genre 'Electronic'").

The system can be set up with a custom (or empty) music library where additional songs can be added via the API. We also provide a pre-analyzed library with 50,000 songs from http://freemusicarchive.org

"Search by Sound" consists of:

- SMAFE Audio Feature Extractor, a server side application that analyzes rhythmic and spectral content of MP3 files and stores them in a database
- SMINT Audio Similarity System which computes similarities between the different feature vectors of all songs in the database
- SMINT API: A REST Web API to query for similar songs, as well as adding and removing songs (see Interface Description below)
- Search by Sound Web Frontend: A web frontend to interactively search for songs in the database, retrieve and listen to similar songs and perform segment searches based on a waveform segment of a song.

#### *3.7.2 Platforms*

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

- SMAFE and SMINT are implemented in C++ and run as binary libraries on Linux systems (optionally Mac OS, Windows with adaptations)
- SMINT API runs on Apache Web Server using PHP 5.3
- all this is ready hosted at: http://musicbricks.ifs.tuwien.ac.at [2]

- SMINT API can be queried by any application capable of accessing **REST APIs**
- Search by Sound HTML/PHP Web Frontend is provided for testing and evaluation purposes (can be also customized / extended)

### *3.7.3 Interface Description*

SMAFE and SMINT are available as C++ source code with respective function calls available.

The SMINT API is a REST API which can be queried by using HTTP requests.

The API methods are:

- **track/add ...** Add a Track by sending a URL where the track a) can be downloaded or b) providing a local file location (on the same server as the API is running).
- **track/delete/:smint_track_id …** Removes a track from the system.
- **track/:smint_track_id** … Returns a list of tracks that are similar to the given trackid.
- **track_external_key/:external_key** … Returns a list of tracks that are similar to the given external key.
- **version** … Returns version information on the API.

The API returns a proper HTTP status code and an XML document.

### *3.7.4 Readiness level and IPR*

- Stable and tested in industrial environments.
- Service is running as server backend on server http://musicbricks.ifs.tuwien.ac.at
- The API is accessible via endpoint: http://musicbricks.ifs.tuwien.ac.at/smintapi/
- Overview and documentation provided on: http://ifs.tuwien.ac.at/mir/musicbricks
- Full API documentation provided: http://ifs.tuwien.ac.at/mir/musicbricks/SMINT/SMINT-API-Documentation_v1.1.1c-SbS-rel1.3.pdf
- Binaries of backend implementation only available under commercial license (on request)
- API examples provided
- free usage of API and server time on demo server
- fully open-source under MIT license

## 3.8 Sonarflow: Visual Music Discovery Apps (by TU Wien / Spectralmind)

### *3.8.1 Description*

Sonarflow is a compelling visual music discovery app for iPad, iPhone and Android. Through MusicBricks, it is now available open-source for hacking and extending it.

It features a slick UI for browsing music by zooming into a colourful world of bubbles which represent genres, artists or moods and allows discovering new music online from various sources. It is available for iOS and Android, with APIs connecting to 7digital, last.fm, Youtube, Spotify, etc.

Demo videos: http://www.sonarflow.com
Demo App in in Google Play Store.

### *3.8.2 Platforms*

- Android
- iOS (built for iOS 6, still compatible with iOS 9)

### *3.8.3 Interface Description*

See source code on GitHub:
- https://github.com/spectralmind/sonarflow-ios
- https://github.com/spectralmind/sonarflow-android

Documentation also available on GitHub (see Readme)

### *3.8.4 Readiness level and IPR*

- stable and tested in commercial environments
- App has been downloaded by ~ 150,000 users
- was under heavy development for about 3 years
- now fully open-source on GitHub under MIT license

---

## 3.9 Freesound API (MTG-UPF)

### *3.9.1 Description*

**Freesound** (www.freesound.org) is a state-of-the-art online collaborative audio database, built utilising UPF-MTG's technologies, that contains over 200K sounds uploaded and annotated by registered web users. Freesound is the only EU academic database with an API used by commercial organizations as a resource of Creative Commons licensed sound samples. Freesound is continuously growing at a rate of ~120 new sounds and ~1000 new registered users per day. The contents of Freesound are very heterogeneous, ranging from environmental recordings to instrument samples, including voice, foley, music loops and sound effects. All these sounds are annotated with user-provided free-form tags and textual descriptions that enable text-based retrieval. Content-based audio features are also extracted from sound samples to provide sound similarity search.

The Freesound API will provide access to:
- Collaborative repository
- CC licensed sounds +200k
- RESTful API
- API Clients:
    - Python, Javascript, Objective-C

With the Freesound API users can browse, search, and retrieve information about Freesound users, packs, and the sounds themselves of course. Users can find similar sounds to a given target (based on content analysis) and retrieve automatically extracted features from audio files, as well as perform advanced queries combining content analysis

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

features and other metadata (tags, etc...). With the Freesound API, you can also upload, comment, rate and bookmark sounds.

### *3.9.2 Platforms*

Freesound API can be queried by any application capable of accessing **REST APIs.**

### *1.1.1 Interface Description*

The following functionalities will be supported by the defined interface:

- **Authentication**: Supports standard token (api key) authentication or OAuth2 for post requests that need to be linked to a Freesound user account.
- **Searching**: Search based on textual terms, audio features, geo-tagging information, or a combination of these.
- **Downloading sounds**: Download original quality sounds or lossy previews with different qualities in a unified format (either mp3 or ogg).
- **Uploading sounds**: Use Oauth2 authentication to link API requests to a Freesound user account and upload sounds to Freesound from a third party application using the API.

Additionally, a set of Client Libraries for Freesound API will be included:

- Python
- Javascript
- Objective-C (iOS)

For example, using the Python client for Freesound API, one could make a query and retrieve a list of files using the example code below:

```
import freesound, sys,os
c = freesound.FreesoundClient()
c.set_token("<your_api_key>","token")
results = c.text_search(query="dubstep",fields="id,name,previews")
for sound in results:
    sound.retrieve_preview(".",sound.name+".mp3")
    print(sound.name)
```

By the end of the tasks in WP3 (M12), we shall be able to provide accompanying example applications such as:

1. Freesound Drum Machine
    a. Web app
    b. Assign FS samples to an online step sequencer

2. Freesound SampleBank Creation
    a. Web app
    b. Specify tags and build a SampleBank archive
    c. Compatibility with common sampler formats (e.g. soundfonts)

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

## 3.10 Melody Extraction API (MTG-UPF)

### *3.10.1 Description*

The Melody Extraction API is based on the Essentia Library (http://essentia.upf.edu) [3,4], developed at the Music Technology Group of Universitat Pompeu Fabra in the last years.

Essentia is an open-source C++ library for audio analysis and audio-based music information retrieval released under the Affero GPLv3 license. It contains an extensive collection of reusable algorithms which implement audio input/output functionality, standard digital signal processing blocks, statistical characterization of data, and a large set of spectral, temporal, tonal and high-level music descriptors.

For #MusicBricks, we will provide a specific library for melody extraction, both from solo or a cappella recordings, as well as for predominant melody extraction from polyphonic audio.

Melody is the most salient and identifiable element in a musical piece. It had a prominent role in the origins of music, which relied on oral tradition or the usage of rudimentary instruments (e.g. prehistoric flutes). But still today, melodies are at the musical forefront from an educational context to the latest commercial song hit. In the scope of MusicBricks, we aim at promoting music creation and therefore we consider the use and reuse of melodies as a fundamental step. Two existing and complementary technologies for melody extraction will be integrated in MusicBricks and accessed through an API. The first technology component inputs a cappella recordings by users and the second technology component processes polyphonic mixtures with predominant vocals. As part of the Ideas Incubation stages of MusicBricks (WP6), these melodies could then control external instrument synthesizers and remixed with other musical accompaniment.

### *3.10.2 Platforms*

The MelodyExtraction library is developed in C++. Python bindings are also provided in order to be able to use Essentia in an interactive development environment, and they fit naturally with the IPython/NumPy/Matplotlib environment (similar to Matlab).

The library is cross-platform and currently supports:

- Linux
- OSX

### *3.10.3 Interface Description*

- Inputs:
  - audio filename (WAV, 44.1kHz, 16bit)
- Output:
  - YAML file with pitch values in Hz and pitch confidence

### *3.10.4 Readiness level and IPR*

- Technology is ready to be tested. Download code and examples from http://essentia.upf.edu
- Fully documented algorithms and tutorial examples as part of Essentia library
- Stable version 2.1: library used in several projects

- Affero GPL License: source code available. Github repository

---

## 3.11 Onset Description API (MTG-UPF)

### *3.11.1 Description*

Onset Description is part of the EssentiaRT~, and it is provided as a binary file compiled for Mac OSX, Linux and Windows. Additionally, it comes with a collection of abstractions designed to make interaction with the object easier. A number of help files and use examples complete the package. EssentiaRT~ is offered free of charge for non-commercial use only.

### *3.11.2 Platforms*

- Max OS X (10.7 or newer), a recent version of Debian/Ubuntu, Microsoft Windows 7/8 32-bit or 64-bit)
- Software
    - Pd-extended (version 0.42.5 or newer) or Max (Version 5 or newer).
    - On Mac and Windows please make sure you use 32-bit versions of Pd and Max.

Please note: examples are note provided for Pd and not Max (save for a help file). Please consult the Pd patches and adopt them for your own needs in Max.

### *3.11.3 Interface Description*

### *3.11.4 Readiness level and IPR*

- Technology is ready to be tested. Download code and examples from http://mtg.upf.edu/technologies/EssentiaRT~?p=Download%20and%20installation
- Fully documented algorithms and tutorial examples as part of EssentiaRT~ library
- Development version v0.2: library used in several projects
- Affero GPL License: source code available. Github repository

---

## 3.12 Spectral Synthesis API (MTG-UPF)

### *3.12.1 Description*

The Spectral Synthesis API is integrated in the Essentia Library (http://essentia.upf.edu) [3,4], developed at the Music Technology Group of Universitat Pompeu Fabra in the last years. Its purpose is to support the transformation of audio samples in the context of audio mosaicing, or concatenative synthesis.

Essentia is an open-source C++ library for audio analysis and audio-based music information retrieval released under the Affero GPLv3 license. It contains an extensive collection of reusable algorithms which implement audio input/output functionality, standard digital signal processing blocks, statistical characterization of data, and a large set of spectral, temporal, tonal and high-level music descriptors.

For #MusicBricks, we integrated in the Essentia Library a specific set of algorithms for spectral analysis/synthesis for monophonic or polyphonic sounds. Specifically, the integrated algorithms are:

- Sinusoidal Model Analysis / Synthesis
- Sinusoidal Plus Residual Model Analysis / Synthesis
- Sinusoidal Plus Stochastic Model Analysis / Synthesis
- Harmonic Plus Stochastic Model Analysis / Synthesis

### 3.12.2 *Platforms*

The Spectral Synthesis library is developed in C++. Python bindings are also provided in order to be able to use Essentia in an interactive development environment, and they fit naturally with the IPython/NumPy/Matplotlib environment (similar to Matlab).

The library is cross-platform and currently supports:

- Linux
- OSX

**2. *Interface Description***

- Inputs:
  - audio filename (WAV, 44.1kHz, 16bit)
- Output:
  - audio filename (WAV, 44.1kHz, 16bit)

### 3.12.3 *Readiness level and IPR*

- Technology is ready to be tested. Download code and examples from http://essentia.upf.edu
- Fully documented algorithms and tutorial examples as part of Essentia library
- Stable version 2.1: library used in several projects
- Affero GPL License: source code available. Github repos

## 4. Download and installation

Next table compiles the #MusicBricks included in the Final Version of the APIs, including the download links, system requirements and licensing terms.

| Tool | Platform | Programming Environment | Real-Time Usage | Links for Installation and Usage | Provider | License |
|---|---|---|---|---|---|---|
| Melody Extraction | Linux<br>Mac OS X<br>Windows | C++<br>Python<br>Executables | No | OverviewDownloadTutorial (Python code and Binay executables) | UPF MTG | Affero GPLv3 |
| Onset Description | Linux<br>Mac OS X<br>Windows | PureData<br>MaxMSP | Yes | Documentation and download | UPF MTG | Open source |
| Rhythm and Timbre Analysis | Linux<br>Mac OS X<br>Windows | Python<br>Matlab<br>Java | Possible | Overview Downloads GitHubTutorial Feature Description | TU Wien | Open source |
| Search by Sound Music Similarity Web API | any | REST API | No | OverviewAPI Documentation Frontend tutorial | TU Wien – Spectralmind | Free usage of API (Binaries and source subject to licensing) |
| MusicBricks Transcriber | Linux<br>Mac OS X<br>Windows | Excecutable & Python Wrapper | No | Documentation & Download | Fraunhofer IDMT | Royalty-free, limited, non-transferable license |
| Real-time Pitch Detection | Linux<br>Mac OS X<br>Windows<br>iOS<br>Android | C++ Library | Yes | Documentation & Download | Fraunhofer IDMT | Royalty-free, limited, non-transferable license |
| Real-time Pitch Shifting & Time Stretching | Linux<br>Mac OS X<br>Windows<br>iOS<br>Android | C++ Library | Yes | Documentation & Download | Fraunhofer IDMT | Royalty-free, limited, non-transferable license |
| Goatify | Linux<br>Mac OS X<br>Windows | Excecutable | No | Documentation & Download | Fraunhofer IDMT | Royalty-free, limited, non-transferable license |

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

| | | | | | | |
|---|---|---|---|---|---|---|
| Gesture Sensors and Gesture Analysis | Any platform with OSCfor receiving raw dataMacOS or Windows for Gesture Analysis with MaxMSP | MaxMSP | Yes | http://ismm.ircam.fr/devices/ | IRCAM | Sensor should be returned after the hackathon,Max external and patches free |
| FreeSound API | Web API | Python, Javascript and Objective C clients | No | Documentation | UPF MTG | Terms of use |
| Spectral Synthesis API | Linux<br>Mac OS X<br>Windows | C++<br>Python<br>Executables | No | Overview<br>Download | UPF MTG | Affero GPLv3 |

# 5. Extending and combining #MusicBricks

We envision novel Music Interaction systems to combine the aforementioned technologies. But a major challenge today is how to effectively and rapidly integrating multiple technologies, allowing creative users and makers to build prototypes.

For example, we see that new systems are shifting towards screen-less devices, taking advantage of gestural controllers or objects augmented with sensors (e.g. IoT). Especially in performance situations it allows to escape from the image of the 'computer' musician sitting in front of a screen. These systems can also take advantage today of online connectivity through cloud-based services. Big data repositories of music and sounds with permissive licenses are easily available and data bandwidth is no longer a bottleneck for downloading and transmitting sounds online. (Research on audio analysis, classification and processing).

One of the goals of the project was to demonstrate the capabilities of combining technologies from different partners in a creative project. This section presents a Demo Application as an example of extending the potential of MusicBricks with new projects beyond the supported creative and industry testbeds. These two prototypes are a clear example of the interoperability and ease of integration of the provided MusicBricks portfolio. This work is part of the tasks T 3.2 and T3.3.

The first one (*HandsFreesoundMachine*) uses voice and gesture inputs to control timing and timbre of a drum machine. Voice input is used to query sounds from an online repository, while gesture analysis is used to define the rhythm of a 4-track step sequencer. This system combines hardware and software tools together with online web services. It integrates various systems seamlessly: rIOT sensor, FreeSound API, Creative Commons Licensed sounds and Google Speech API. The prototype is fully functional and publicly available.

The second one *(Goatify Web Service*) is a web service that replaces the main melody in a song with a sample, the sample will be pitched according to the melody. The sample is retrieved from the Freesound repository using the Freesound API. Currently this demo application is in development phase, and in this section we provide the details at concept level.

## 5.1 Demo application 1: HandsFreesoundMachine

This demo application combines two "music bricks": the rIOT Gesture sensor (IRCAM), and the Freesound API (UPF). Additionally, we show how we can extend the concept of MusicBricks with external tools. In this case we use the Google Speech Recognition API, in combination with the partners technologies.

Next we provide the concept and the details of the steps followed. This demo application can be regarded also as a validation of the Final Version of the #MusicBricks API.

### *5.1.1 Concept*

The idea of this demo application can be summarized as follows:

- a web-based hands-free DRUM-MACHINE controlled by voice and gesture inputs
- a 4-track drum machine, using a 16 steps sequencer
- It retrieves samples directly from Freesound using the FS API v2
- Voice interface for specifying the search query (using Google API)
- A gesture sensor to toggle on/off the step sequencer in multiple drum tracks
- Tempo set by gesture recognition: headbanging

### *5.1.2 Technical requirements*

We identify four different aspects with regard to the technical requirements:

- Gesture recognition: we need a light wireless sensor to detect a 'drum hit' gesture, to set the tempo and toggle on/off the drum machine steps
- Speech recognition: we need an automatic speech recognition to get the user voice query to a text query to be passed to the FreeSound API.
- Sound retrieval: we need an online repository of sounds, accessible through a web API.
- Web-based audio playback: we need a web technology to playback audios downloaded from the internet in a web browser.

### *5.1.3 Implementation*

#### *Gesture sensor and analysis*

The gesture analysis component allows to recognize user gestures. We are uniquely interested in detecting one type of gesture: a 'drum hit' or 'kick'. It is a short discrete and impulsive gesture that is used to control the rhythmic aspects of the drum machine. First it is used to set the tempo by continuously tapping the target tempo. Second, during playback it allows to toggle on/off the each step in the drum machine sequencer in real-time.

We use the RIoT sensor, developed by IRCAM, and provided as one of the MusicBricks. This is a wireless sensor board that pre-processes sends gesture data in OSC format through a Wifi connection. We need to create a private Wifi network to receive the data from the sensor. More technical details on the RIoT sensor are available in this web page: http://ismm.ircam.fr/devices/

The current version of the RIoT firmware already supports the detection of 'drum hits', which are sent with the OSC message 'kick'. Hence our web server listens only to incoming '/kick' OSC messages.

The code is publicly available in the Github repository:
*https://github.com/Ircam-RnD/RIoT*

#### *Navigation and user control*

Users can interact with the Hands-free Sound Machine through gestures and voice commands. An Automatic Speech Recognition (ASR) system is used to provide voice control which allows users to change the states of the application (see Implementation section for more details on the ASR).

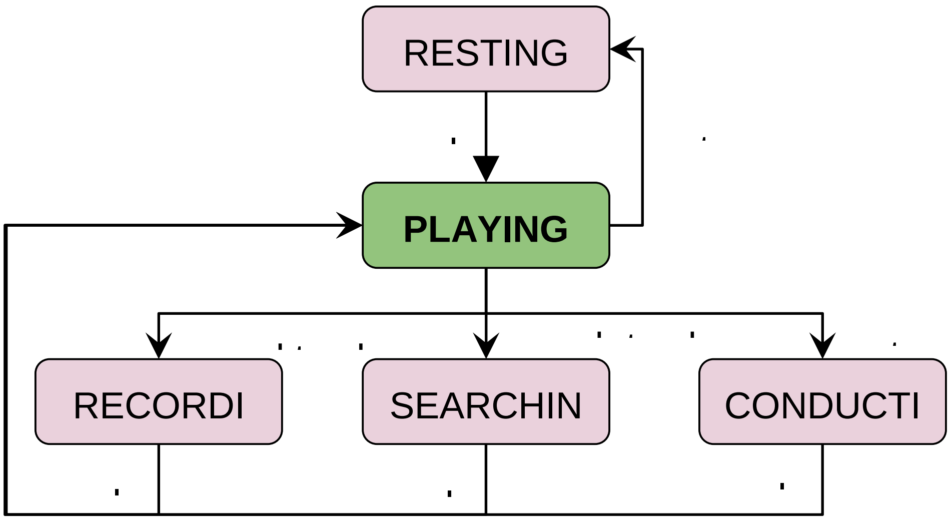
The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

The different voice commands recognised by the ASR are:

- PLAY
- STOP
- [SET] TEMPO
- RECORD [TRACK] #n {ONE; TWO THREE; FOUR}
- CLEAR [TRACK] #n {ONE; TWO THREE; FOUR}
- SEARCH [TRACK] #n {ONE; TWO THREE; FOUR}

Using these commands, users can navigate across the different states of the application as follows:

1. Resting:
    a. Drum machine is stopped. On page load, default sounds taken from Freesound and a random sequence are loaded in the 4-track sequencer. The default sounds are:
        i. kick drum
        ii. snare drum
        iii. closed hi-hat
        iv. ride cymbal
    b. ASR listening mode for commands: PLAY, SET TEMPO, SEARCH TRACK
2. Playing
    a. Drum machine is running
    b. ASR is listening for commands: STOP; RECORD TRACK; CLEAR TRACK; SEARCH TRACK, SET TEMPO
3. Searching
    a. ASR is listening for new sound query: “free words + pause”
    b. Call FS API to search new sound and changes it automatically upon receiving results
4. Recording (default playing mode)
    a. Gesture detection for step activation/deactivation
    b. Drum machine is running
    c. ASR is listening to commands: PLAY; STOP, CLEAR TRACK,
5. Tempo
    a. Gesture detection for changing the tempo
    b. Drum machine is running with the detected tempo (slow adaptive window)
    c. ASR is listening to commands: PLAY; STOP,



The MusicBricks project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement n°644871

***Web application: server and browser***

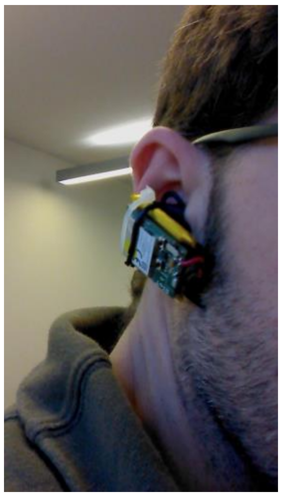
The Hands-free Sound Machine combines several technologies to build a voice and gesture controlled drum machine that plays sounds from Freesound. On the one hand, a web server built using Python's Flask[1] package establishes a web sockets connection with the client browser. Furthermore, the web server includes an Open Sound Control[2] (OSC) module that processes OSC messages received from the RiOT sensor and sends them to the web browser using the web sockets connection. On the other hand, the client (Web browser) shows the UI (using JQuery[3] and Bootstrap[4]), interprets messages from the RiOT sensor (forwarded by the web server), communicates with the Freesound API[5] to retrieve sounds and uses the Web Audio API[6] to play them in a time-accurate manner[7]. Finally, the client is also in charge of processing and interpreting the voice control (ASR) input from the user using the Web Speech API[8]. Voice signal is feeded to the computer using a Bluetooth microphone that is attached to the RiOT sensor to provide a single object to interface with the drum machine. By using a Bluetooth microphone that the user can hold in his hand, we can adjust the sensitivity in a way that allows us reduce potential voice recognition errors caused by audio feedback from the speakers. The diagram below shows the architecture of the system and how the different components are interconnected.

### *5.1.4 Prototype*

For building the prototype we use a commercial bluetooth headset as voice input (Sennheiser VMX 200-II). It allows us to attach the headset and the rIOT sensor together and use it as a normal on-ear headset. We selected this model for its reasonable cost (80EUR) and high quality.

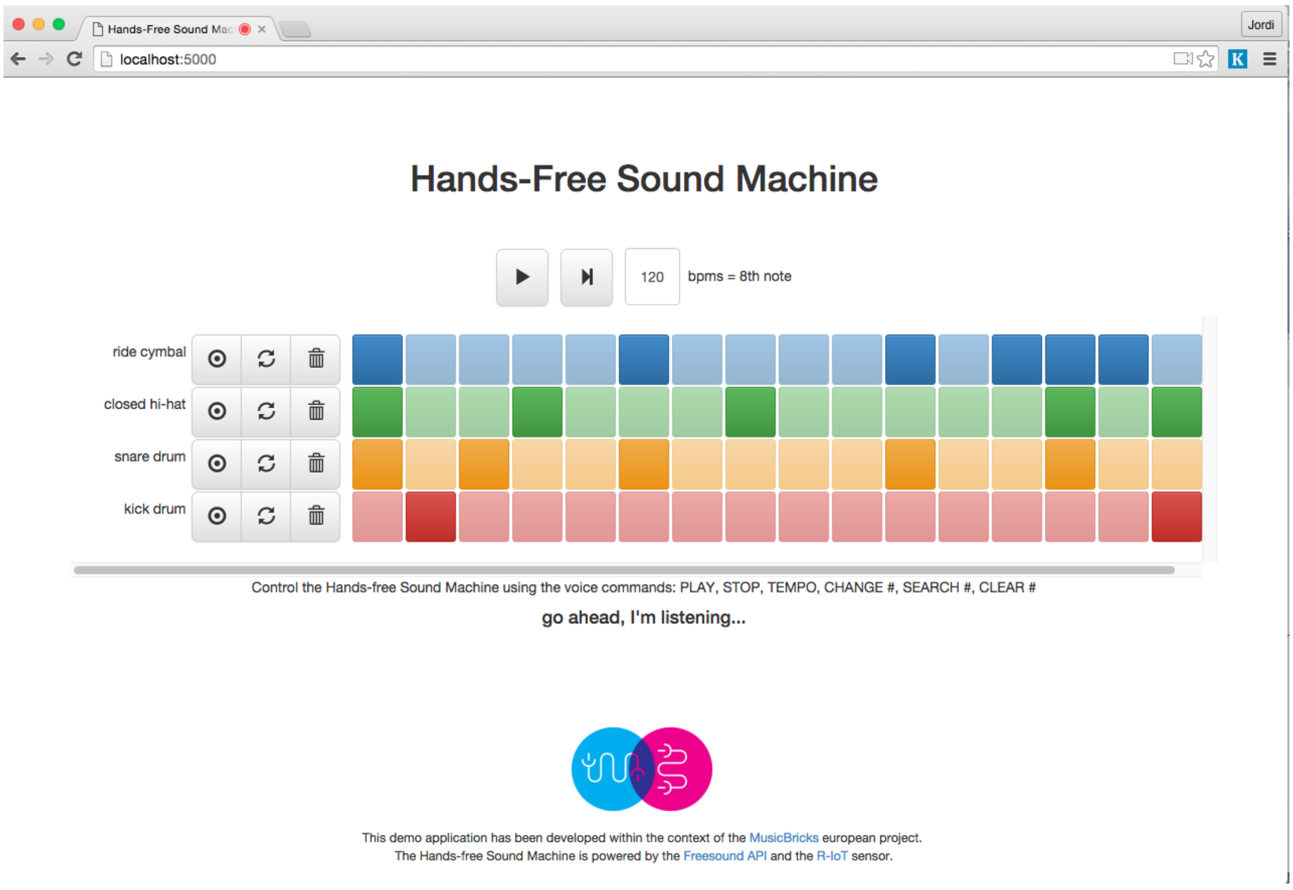
[1] http://flask.pocoo.org
[2] http://opensoundcontrol.org/introduction-osc
[3] https://jquery.com
[4] http://getbootstrap.com
[5] http://freesound.org/docs/api
[6] http://www.w3.org/TR/webaudio
[7] http://www.html5rocks.com/en/tutorials/audio/scheduling
[8] https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

| | | |
|---|---|---|
| Bluetooth headset VMX-200 | Attached headset and RIOT sensor |  Wearing the sensors |

The web browser application is in charge of the audio playback and provides visual feedback to the user. Although the application can be used without display, we provide a GUI in the web browser. The visual feedback is important to get familiar with the system, but it is not necessary once the user has been trained.

Next figure provides the GUI of the application. The step sequencer shows the 4 tracks, the current tempo, the toggled steps, and provides description of the selected sounds for each track. When loading the page the default sounds queried: {ride cymbal, closed hihat, snare drum and kick drum}.


Hands-Free Sound Machine

120 bpms = 8th note

ride cymbal
closed hi-hat
snare drum
kick drum

Control the Hands-free Sound Machine using the voice commands: PLAY, STOP, TEMPO, CHANGE #, SEARCH #, CLEAR #

go ahead, I'm listening...

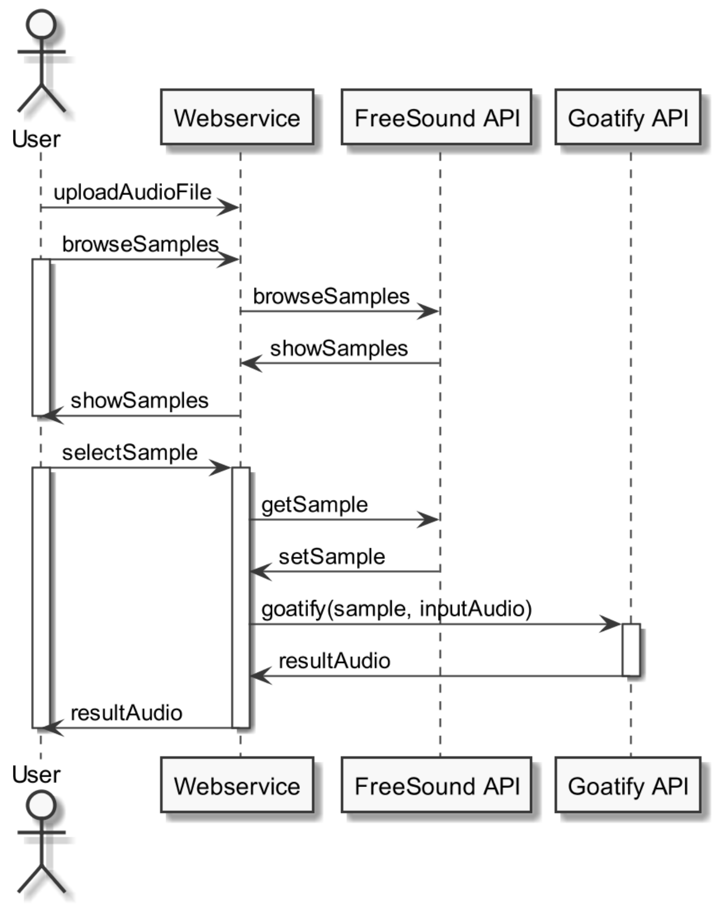
This demo application has been developed within the context of the MusicBricks european project.
The Hands-free Sound Machine is powered by the Freesound API and the R-IoT sensor.

All source code for the web application is available online in this github repository:
https://github.com/MTG/hands-free-sound-machine

## 5.2 Demo application 2: Goatify Webservice

### *5.2.1 Concept*

This web services allows to replace the main melody in a song with a sample. The target sample will be pitched according to the melody. This webservice combines the two “music bricks” Goatify API (Fraunhofer IDMT) and the Freesound API (UPF). Instead of manually downloading samples one could directly browse in the Freesound library for samples to use for Goatify. Therefore the user must upload his music track for which he wants the melody to be replaced by the webservice. Then the Freesound Library will be displayed and the user can select the sample to be used. Both files (audio from user and selected sample) will be handed over to the Goatify command line tool. The resulting audio file can then be downloaded from the user. The use-case is shown in the following diagram.



### *5.2.2 Technical requirements*

Both Music Bricks offer the required interfaces for getting samples and creating the desired output file. Therefore one has to create the webservice which calls the APIs. The user can access the webservice with a web front-end which has to be created.

### *5.2.3 Implementation*

The webservice can run on any Windows, Linux or Mac OSX Server since both “music bricks” support these platforms. Therefore one could freely choose which server framework to use.

### *5.2.4 Prototype and evaluation*

This is still in a concept stage since the Goatify API was added recently to the MusicBricks API portfolio as part of this deliverable.

## 6. Future steps

The Final Version of the APIs will be integrated by the selected projects in the Industry Testbeds during the next months. With the finalization of the tasks of WP3 (APIs), all partners have made sure that the provided technologies have been tested, all properly documented and all tools made available online.

On the technology side, this is the last deliverable of this work package. All Tasks T3.1 T3.2 and T3.3 have been successfully carried out. As future steps, the technology partners in the Consortium will be available to the participants in the incubated projects, providing their technical support and guidance when required until the end of the project (June 2016).

## 7. Conclusions

This document concludes the work related to the development, documentation and distribution of the MusicBricks technology portfolio. We successfully identified and integrated under the same umbrella various relevant and complementary technologies by the research partners (TUW, Fraunhofer, UPF).

All these technologies have been widely used in the Creative Tests in the events throughout 2015. Feedback from creative users (hackers) has allowed improving certain aspects of the tools. For example, several users requested more real-time and easy to run tools.

With the goal to put these technologies available to a wider audience, we have a public web page (http://musictechfest.net/musicbricks/#) with a description of all technologies. It includes a more detailed technical description of the portfolio (http://musictechfest.net/technicaldata/), providing detailed information about Tool name, Purpose, Platform, Programming Environment, Real-Time Usage, Links for Installation and Usage, Provider and License schema.

## References

[1] Thomas Lidy and Andreas Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Proceedings of the Sixth International Conference on Music Information Retrieval*, pages 34--41, 2005. [ pdf ]
[2] "Search by Sound" Web frontend, SMINT Web API, available from http://musicbricks.ifs.tuwien.ac.at
[3] Bogdanov, D., Wack N., Gómez E., Gulati S., Herrera P., Mayor O., et al. (2013). ESSENTIA: an Audio Analysis Library for Music Information Retrieval. International Society for Music Information Retrieval Conference (ISMIR'13). 493-498.
[4] Essentia Library, MTG-UPF, http:// essentia.upf.edu, http://github.com/MTG/essentia