# Sensor Music Player

István Szőllősi

*Faculty of Sciences and Letters, "Petru Maior" University of Târgu Mureș*

August 25, 2018

# Contents

# 1 About

I would like to present how to gain values from an accelerometer and process that data on a separate server.
The scopes of the project:

- main scope - make a music player that can be controlled with gestures
- understable and optimized signal processing (FastDTW, Dtaidistance, etc)
- easily buildable/usable server side (Node.js, Express.js, PythonShell)

Listen to music can be a relaxing activity or source of motivation for some people. Many people always listen to music when he/she has possibility. Nowadays almost everybody use a smart device to listen to music. People listen to music in there cars, in there homes, in the office, on the bus/train almost in every places.

To control the music player you need to touch the device at least once a time if you are in the car or if the screen of a smart device is open. But, if your music player device is in your pocket, you at first need to pull out, to unlock the screen, to open the music player app and for the last time to change your track. This is a long way to change a track and in some case it is hard to execute for example on a bus/train with many people.

To resolve these problems we can use a music player which can be controlled with gestures. From this time we do not need to unlock everytime a smart device to change a track or we do not need to pull out from our pocket. We can easily change a track if the device is in our pocket by double tap the topside of the device.

The player monitors the state changes of the accelerometer and with a help of a server can analyze in almost real-time what type of gesture is and what to do.

## 1.1 Technical about

The main structure of the repository is *a valid Android project* with several additionals folders, like the:

- **backend** folder where the *Python* and *JavaScript* code is stored
- **docs** folder where the documents about the project are stored

Source code available: https://github.com/I-sty/SensorMusicPlayer/.

## 1.2 Workflow

1. Use the Android application to get the values from the accelerometer
   (a) Navigate to *Config activity*
   (b) Press the *Record* button to start recording the accelerometer
      - The text of the button will be changed to *Stop*
   (c) Press the *Stop* button to stop recording
   (d) Press the *Send* button to send the recorded signal to the Node.js server
   (e) Optionally, you can "analyze" your recorded array by pressing the *Analyze* button
      - This feature checks your items in the array if they are greater than a predefined threshold.
      - This feature eliminates the ability to send almost blank lists to the server
2. By pressing the *Send* button in Android application, the application sends the recorded list to the server using a REST POST call.
3. The Node.js server receives the request from the application and process it.
   (a) **REST API level** - The request arrives to the right URI path, that was registrated to receive request.
   (b) **Controller level** - The *REST API level* forwards the request to the right function of the controller.
      i. The body of the request is forwarded to the *Model level* to parse. The return object will be saved in MongoDB.
      ii. The body of the request is forwarded to another function where the DTW's distance will be calculated using the PythonShell module.
   (c) **Model level** - The body of the request will be interpreted as a *Mongoose* object and returned to the *Controller level* so it can be saved.
4. The calculated distance will be send to the Android application as a respond for the POST request.

# 2 Node.js

In Node.js is very simple to create a small web server for REST calls.

## 2.1 Installation

```
sudo apt install nodejs
```

## 2.2 Usage

To start use the follow command in the project's folder:

```
npm run start
```

## 2.3 Configuration

Used tutorial: Build Node.js RESTful APIs in 10 Minutes

### 2.3.1 Mongoose

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.

#### 2.3.1.1 Installation

```
npm install mongoose
```

### 2.3.2 Express

Express.js, or simply Express, is a web application framework for Node.js. It is designed for building web applications and APIs

#### 2.3.2.1 Installation

```
npm install express --save
```

### 2.3.3 Nodemon

*nodemon* is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected.

#### 2.3.3.1 Installation

```
npm install --save-dev nodemon
```

# 3 MongoDB

MongoDB to store signal data from the *Y axis* of the accelerometer from the Android devices.

## 3.1 Installation

```
sudo apt install mongodb
```

## 3.2 Runing

To start mongo service use the follow command:

```
sudo mongod --config /etc/mongodb.conf
```

## 3.3 Drop a collection

In some cases you need to drop collections, for example when the collections contains random test cases what were introduced during the development process.

To drop/clear a collection introduce the following codes, in the Mongo's shell.

To access the Mongo's shell enter the *mongo* command in terminal.

```
show dbs
use <db>
show collections
db.<collection>.drop()
```

Listing 1: MongoDB shell commands to drop a collection

# 4 Matplotlib

Matplotlib is a plotting library for the Python programming language. I used *matplotlib* to generate graphical view of my series.

## 4.1 Installation

Install module using this tutorial.

Optionally, you need to install the **python-tk** package also.

## 4.2 Examples

According to this official tutorial you can easily generate a plot about an array using this Python script:

```
1 import matplotlib.pyplot as plt
2 plt.plot([ 5.733, 1.704, -2.713, -1.343, 2.604, 3.922, 2.157, -0.910, -2.414,
      -2.943, -1.526, -0.823])
3 plt.ylabel('some numbers')
4 plt.show()
```

Result:

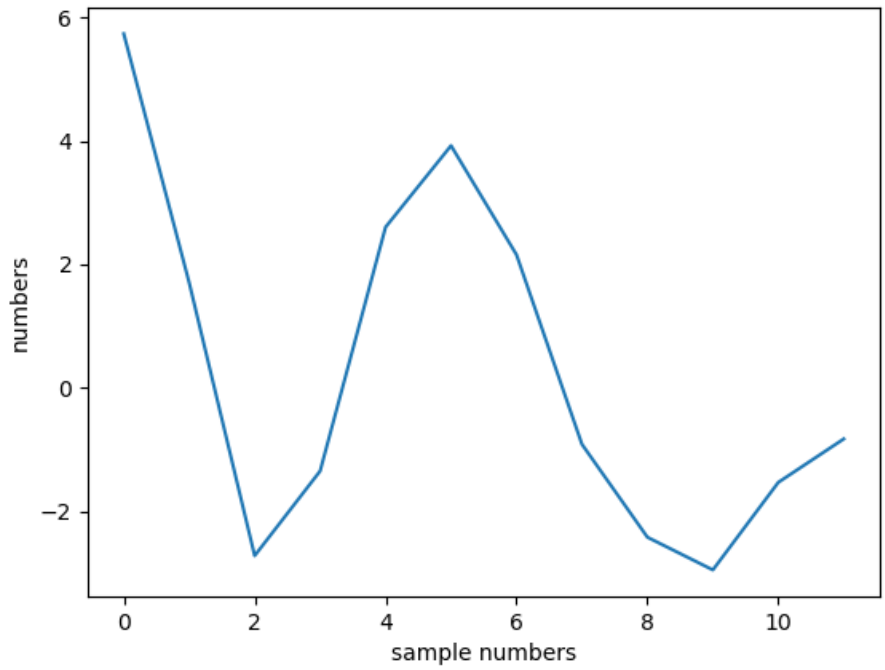


Figure 1: The result of the script

This example was very easy, so here is a *normal* signal from the accelerometer:

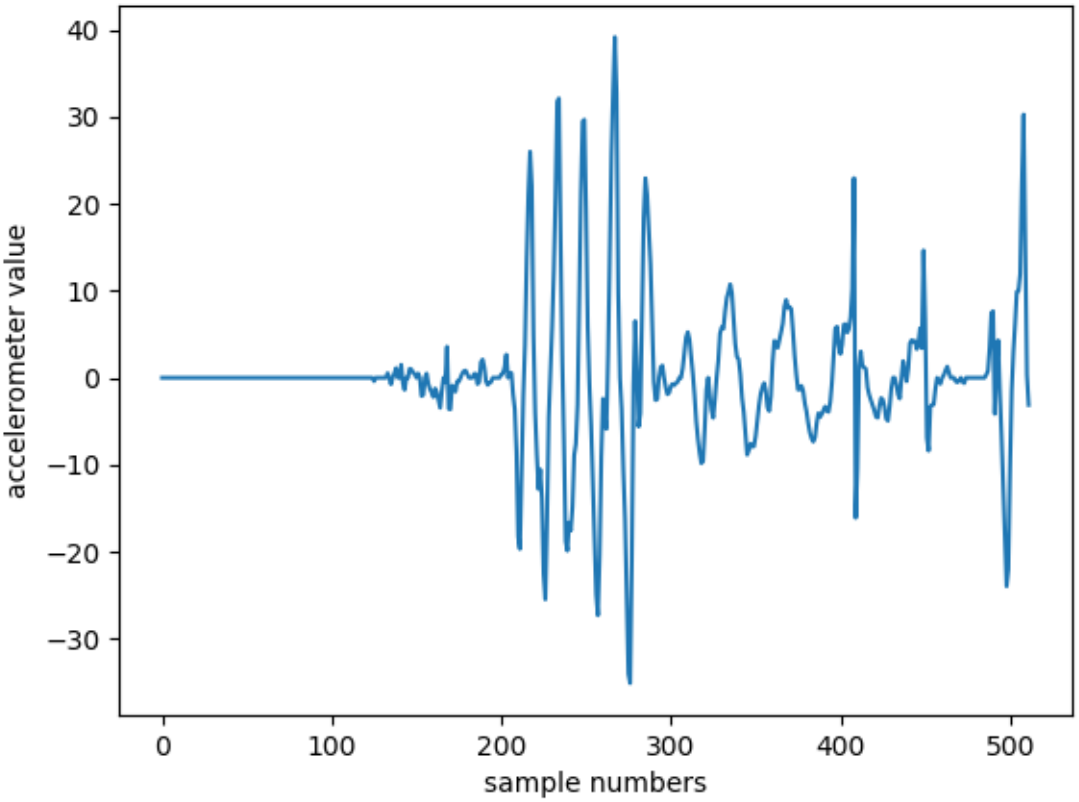


Figure 2: A section of the signal of accelerometer in real usage

# 5 DTAIDistance

Library for time series distances (e.g. Dynamic Time Warping) used in the DTAI Research Group.

DTAIDistance official documentation: dtaidistance.readthedocs.io

Source available on https://github.com/wannesm/dtaidistance.

## 5.1 Installation

Run the follow codes in terminal to install the module:

```
sudo apt install python3-pip python3-setuptools  python3-dev python3-tk
pip3 install wheel
pip3 install dtw
pip3 install dtaidistance
```

Listing 2: Used Linux Mint 19

## 5.2 Usage

```
from dtaidistance import dtw
from dtaidistance import dtw_visualisation as dtwvis
import numpy as np

s1 = np.array([0, 1, 2, 1, 0, 2, 1, 0, 0])
s2 = np.array([0, 1, 2, 1, 0, 0, 1, 2, 1])

d, matrix = dtw.warping_paths(s1, s2, window=25, psi=2)
print('DTW distance = ', d)

dtwvis.plot_warpingpaths(s1, s2, matrix, best_path, "image.png")
```

Listing 3: Script to calculate DTW distance and plot warping paths matrix

The result will be put in the newly created **image.png** file.

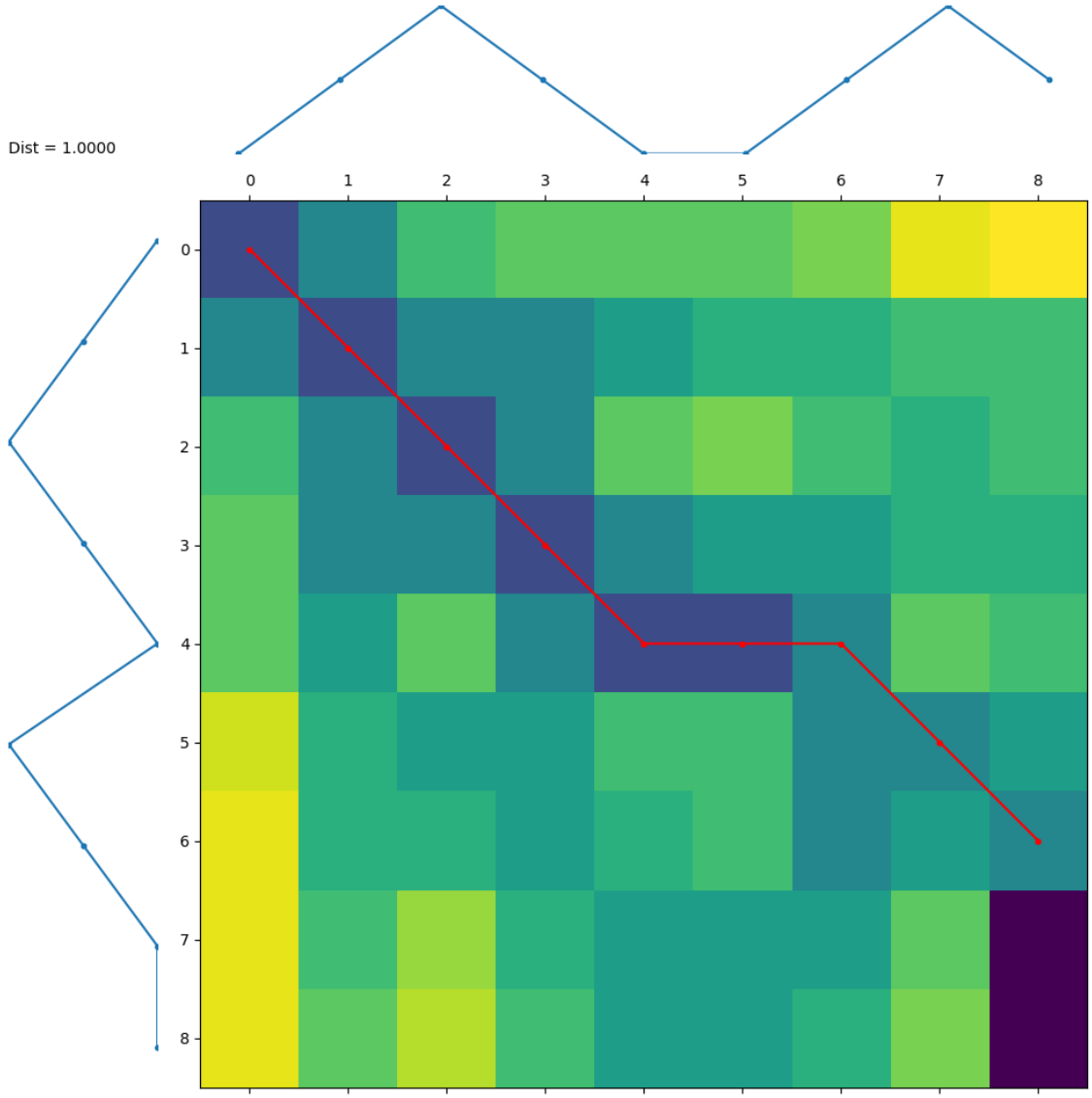


Figure 3: DTAIDistance's warping paths matrix

## 5.3 Documentation

```
import numpy as np
s1 = np.array([0, 1, 2, 1, 0, 2, 1, 0, 0])
```

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

The *array()* function creates an array from a given list. See more details here.

```
from dtaidistance import dtw
d, matrix = dtw.warping_paths(s1, s2, window=25, psi=2)
print('DTW distance = ', d)
print('DTW matrix = ', matrix
```

The *warping_paths()* function calculates the DTW distance and the DTW matrix for the two given series. See more details here.

```
from dtaidistance import dtw_visualisation as dtwvis
dtwvis.plot_warpingpaths(s1, s2, matrix, best_path, "image.png")
```

The *plot_warpingpaths()* method plots the warping paths matrix into the *image.png* file. See more details here.

# 6 Postman

## 6.1 Installation

Installed according to this article: How to install Postman native app in Linux Mint 18.3

Used to test the main functionalities of the Node.js server.

## 6.2 Usage

### 6.2.1 GET

To get all buffers from database run this code in Postman/Linux terminal

```
curl -X GET http://localhost:3000/buffers
```

Listing 4: Get all buffers

The response is or an empty list, if no items in the database or a list like this:


```
[
    {
        "value": [
            5.733050346374512,
            1.704751968383789,
            -2.7134790420532227,
            -1.343064308166504,
            2.6042985916137695,
            3.92281436920166,
            2.15725040435791,
            -0.9106369018554688,
            -2.4146032333374023,
            -2.943338394165039,
            -1.5269522666931152,
            -0.8230304718017578
        ],
        "_id": "5b82607f5601ec575d3bf0e4",
        "__v": 0
    }
]
```


Listing 5: A sub section of the signal to process

### 6.2.2 POST

Post a new buffer a.k.a a sub section of the signal to store and process. Run this code in Postman or in a Linux terminal to post a new buffer to the Node.js server.

```
curl -X POST http://localhost:3000/buffers  -d '{
  "value":[-2.2, -1.1, 0, 1.1, 2.2]
}'
```

Listing 6: Send signal data via REST

# 7 PyMongo

Use PyMongo if you want to get items from your MongoDB and want to parse in Python environment.

## 7.1 Installation

```
pip3 install pymongo
```

Installing with pip: http://api.mongodb.com/python/current/installation.html

## 7.2 Usage

```
import pymongo
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
client.database_names()
db = client['<db_name>']
db.collection_names()
coll = db['<collection_name>']

for col in coll.find({}):
  for keys in col.keys():
    if keys == "value":
      print ('{', keys, ":" , col[keys] , '}' )
    print('\n')
```

# 8 PythonShell

Use PythonShell if you want to call a Python script in Node.js environment. For example when you receive a POST request in the Node.js server and you want to process your data using a Python script.

## 8.1 Installation

```
sudo npm install python-shell
```

## 8.2 Usage

```
var PythonShell = require('python-shell');

function callPythonMethod(req, res){
  var options = {
    mode: 'text',
    pythonPath: 'python3', // or use the command
    pythonOptions: ['-u'], // get print results in real-time
    scriptPath: '<fullPathToTheFolder>/backend/python/',
    args: [req]
  };
  PythonShell.run('<script>.py', options, function (err, results) {
    if (err) res.send(err);
    res.json(results);
  });
};
```

## 8.3 Integrate into the project

You need to add these lines of code into the method where you create and save a new MongoDB item. The first step is to convert the input paramater a.k.a the body of the request to JSON object. From this JSON object you can get the array, convert to string and pass to the *callPythonMethod()* as first parameter.

```
exports.create_a_buffer = function(req, res) {
  var new_buffer = new Buffer(req.body);
  new_buffer.save(function(err, task) {
    if (err) res.send(err);
    //res.json(task);
```

```
  });

  var stringify = JSON.stringify(req.body);
  var body = JSON.parse(stringify);
  callPythonMethod(body.value.toString(), res);
};
```

# 9 Plot everything

This is a short Python script that plots every item in the MongoDB to a separate *png* file.

```
import pymongo
import os
import shutil
import matplotlib.pyplot as plt
from pymongo import MongoClient

client = MongoClient('localhost', 27017)
print('Databases: ', client.database_names())
db = client['Tododb']
print('Collections: ', db.collection_names())
coll = db['buffers']

dir = 'asimage'
if not os.path.exists(dir):
    os.makedirs(dir)
else:
    shutil.rmtree(dir)
    os.makedirs(dir)
os.chdir(dir)

for col in coll.find({}):
  plt.plot(col["value"])
  plt.ylabel('accelerometer value')
  plt.xlabel('sample numbers')
  plt.savefig(str(col["_id"]))
  plt.gcf().clear()
```