Brunel University London

College of Engineering, Design and Physical Science
Electronic and Computer Engineering

Assignment

Project Control and Managment

Distributed Computing Systems Engineering Msc

Authors: Christoph Gschrey
Dennis Müller
Antonio Parotta

Date: 23. March 2018

Supervisor: Dr. Alireza Mousavi

College of Engineering, Design and Physical Sciences

Department of Engineering and Design

ASSIGNMENT SUBMISSION FORM

Please note: that no course work will be accepted without this cover sheet.

Please ensure: that you keep a copy of work submitted and retain your receipt in case of query.

Student Number:	**SPO ID Number** (Office use only):	
Course:		**Level:**

MODULE	
Module Code:	**Module Title:**
Lab / Assignment:	**Deadline:**
Lab group (if applicable):	**Date Stamp** (Office use only):
Academic Responsible:	
Administrator:	

Please note: that detailed feedback will be provided on a feedback form.

✂.....

RECEIPT SECTION (Office Copy)	
Student Number:	**SPO ID Number** (Office use only):
Student First Name:	**Student Last Name:**
Module Code:	**Module Title:**
Lab / Assignment:	
Lab group (if applicable):	**Deadline:**
Academic Responsible:	**Number of Days late:**

DECLARATION	
I have read and I understand the guidelines on plagiarism and cheating in the Handbook and I certify that my contribution to this report fully complies with these guidelines. I confirm that I have kept a copy of my work and that I have not lent my work to any other students.	
Signed:	**Date Stamp** (Office use only):

✂.....

RECEIPT SECTION (Student Copy)	
Student Number:	**Student Name**:
Lab / Assignment:	
Lab group (if applicable):	**Module Title:**
Academic Responsible:	**Deadline:**
Module Code:	**Date Stamp** (Office use only):

The University penalty system will be applied to any work submitted late.

IMPORTANT: You **MUST** keep this receipt in a safe place as you may be asked to produce it at any time as proof of submission of the assignment. Please submit this form with the assignment attached to theDepartment of Design Education Office in the Michael Sterling Building, room MCST 055.

College of Engineering, Design and Physical Sciences

Department of Engineering and Design

ASSIGNMENT SUBMISSION FORM

Please note: that no course work will be accepted without this cover sheet.

Please ensure: that you keep a copy of work submitted and retain your receipt in case of query.

Student Number:	**SPO ID Number** (Office use only):	
Course:		**Level:**

MODULE	
Module Code:	**Module Title:**
Lab / Assignment:	**Deadline:**
Lab group (if applicable):	**Date Stamp** (Office use only):
Academic Responsible:	
Administrator:	

Please note: that detailed feedback will be provided on a feedback form.

✂.....

RECEIPT SECTION (Office Copy)	
Student Number:	**SPO ID Number** (Office use only):
Student First Name:	**Student Last Name:**
Module Code:	**Module Title:**
Lab / Assignment:	
Lab group (if applicable):	**Deadline:**
Academic Responsible:	**Number of Days late:**

DECLARATION	
I have read and I understand the guidelines on plagiarism and cheating in the Handbook and I certify that my contribution to this report fully complies with these guidelines. I confirm that I have kept a copy of my work and that I have not lent my work to any other students.	
Signed:	**Date Stamp** (Office use only):

✂.....

RECEIPT SECTION (Student Copy)	
Student Number:	**Student Name**:
Lab / Assignment:	
Lab group (if applicable):	**Module Title:**
Academic Responsible:	**Deadline:**
Module Code:	**Date Stamp** (Office use only):

The University penalty system will be applied to any work submitted late.

IMPORTANT: You **MUST** keep this receipt in a safe place as you may be asked to produce it at any time as proof of submission of the assignment. Please submit this form with the assignment attached to theDepartment of Design Education Office in the Michael Sterling Building, room MCST 055.

College of Engineering, Design and Physical Sciences

Department of Engineering and Design

ASSIGNMENT SUBMISSION FORM

Please note: that no course work will be accepted without this cover sheet.

Please ensure: that you keep a copy of work submitted and retain your receipt in case of query.

Student Number:	**SPO ID Number** (Office use only):	
Course:		**Level:**

MODULE	
Module Code:	**Module Title:**
Lab / Assignment:	**Deadline:**
Lab group (if applicable):	**Date Stamp** (Office use only):
Academic Responsible:	
Administrator:	

Please note: that detailed feedback will be provided on a feedback form.

✂.....

RECEIPT SECTION (Office Copy)	
Student Number:	**SPO ID Number** (Office use only):
Student First Name:	**Student Last Name:**
Module Code:	**Module Title:**
Lab / Assignment:	
Lab group (if applicable):	**Deadline:**
Academic Responsible:	**Number of Days late:**

DECLARATION	
I have read and I understand the guidelines on plagiarism and cheating in the Handbook and I certify that my contribution to this report fully complies with these guidelines. I confirm that I have kept a copy of my work and that I have not lent my work to any other students.	
Signed:	**Date Stamp** (Office use only):

✂.....

RECEIPT SECTION (Student Copy)	
Student Number:	**Student Name**:
Lab / Assignment:	
Lab group (if applicable):	**Module Title:**
Academic Responsible:	**Deadline:**
Module Code:	**Date Stamp** (Office use only):

The University penalty system will be applied to any work submitted late.

IMPORTANT: You **MUST** keep this receipt in a safe place as you may be asked to produce it at any time as proof of submission of the assignment. Please submit this form with the assignment attached to theDepartment of Design Education Office in the Michael Sterling Building, room MCST 055.

Contents

List of Figures

List of Tables

Abstract

ADD SOME TEXT HERE

1 Introduction

1.1 The Idea

The Digital Life Tracking App is designed to enable users to track their daily tasks and the time spent on them. To make it more convenient to trigger an activity, the start and end of an activity may be triggered in different ways. Possible activation mechanisms could be:

- Two claps
- moving the smartphone in a certain way (gestures)
- activation based on GPS position
- pressing a button

Users should be able to define their own activities and select an activation function from a list and assign it to one of their activities. The user should then be able to display his history in form of charts. A separate device (particle photon board) could also be used to trigger a specific activity.

1.1.1 The Deep Dive

For the prototype in the context of the assigment we decided to focus mainly on the detection of double clapping as an activity trigger. From the original idea a concept for the following app was developed:

Digital Life Tracking

The app should allow a user to divide his daily routine into categories of activities and to measure the time required for each activity. This enables him to get an overview of his time invested in various activities.

2 Data Analysis

This chapter is about the analysis of the gathered data. The amplitudes of the signals are recorded and stored. The amplitudes alone say nothing about the signal, except how strong it is. Therefore it is necessary to transform it into the frequency domain via *Fourier Transformation*[1]. This shows which frequencies are involved in the recordings.

2.1 First Prototype

The first prototype for the detection of clapping was created with Particle's Photon Board. For this purpose the circuit was rebuilt as shown in Figure 2.1[2] and a test program was loaded with the corresponding Web IDE with which it is possible to collect initial data.

1 https://betterexplained.com/articles/an-interactive-guide-to-the-fourier-transform/
2 Source: MSc_DCSE_Assignment_WS6_Smartphone_Sensing_meets_IoT_2018_final

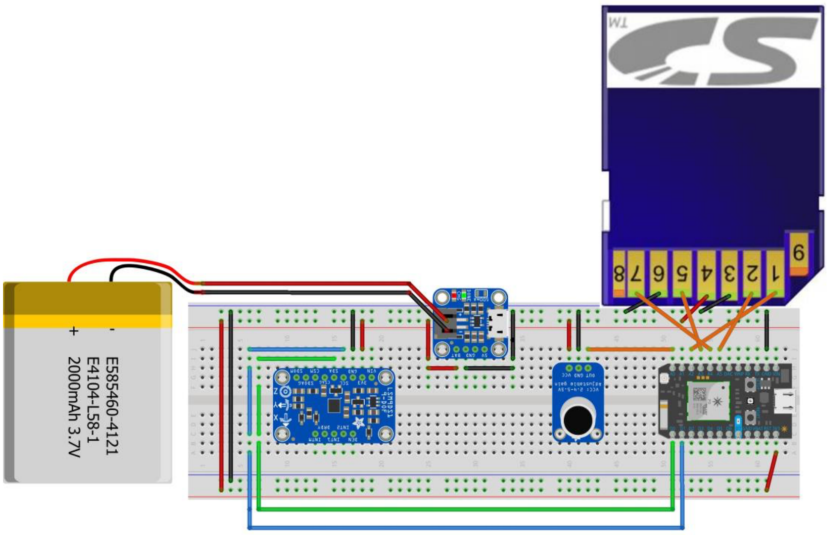

Figure 2.1: Particle's Photon Board

The first six measurements were made at a sampling rate of 10Hz. By performing the Fast Fourier transformation on the collected amplitudes, it was possible to examine the amplitude spectrum. Figure 2.2 shows one result that captured the clapping correctly. This was not the case with every measurement. It can be said that 10Hz is too little for sampling, since possible clapping cannot be measured correctly or only partially in this way. A possible solution is therefore to increase the sampling rate to 1kHz.

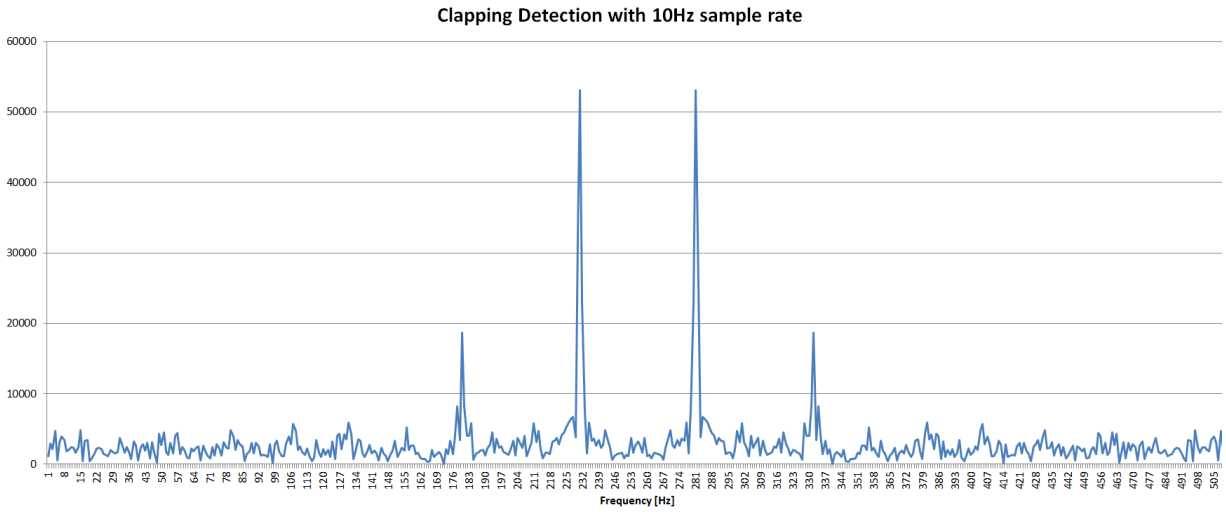

Figure 2.2: Amplitude Spectrum for Clapping with 10Hz sample rate

The following measurements have shown that writing to the SD card is too slow in each iteration, resulting in a reduced sampling rate of about 200Hz. It is therefore an idea to keep the data in the memory of the board until enough data has been collected to be written to the SD card at once.

By holding the data in the memory of the board, it is possible to obtain a higher sampling rate. However, the evaluation of the data is time-consuming, since they first have to be transferred from the SD card to a PC. With many measurements, the time adds up and makes the measurements no longer efficient.

Therefore, it makes more sense to use an Android smartphone with its microphone that can send the created CSV files directly, for example to the dropbox. Thus the data is quickly available for several persons and evaluations can be made directly. In addition to that a good debugger, exception handling, as well as reliable and sophisticated libraries for mathematical operations can be used. The implementation of the software for the Android phone and the used libraries are described in more detail in chapter 3.

2.2 Measurements with Android phone

According to figure 2.3[1] the frequency range of clapping lies between about 120Hz up to 10kHz. If a 10kHz clapping shall be captured right, the sampling rate according to shannon's law[2] must be double the size of the measured signal. Therefore roughly 20kHz.

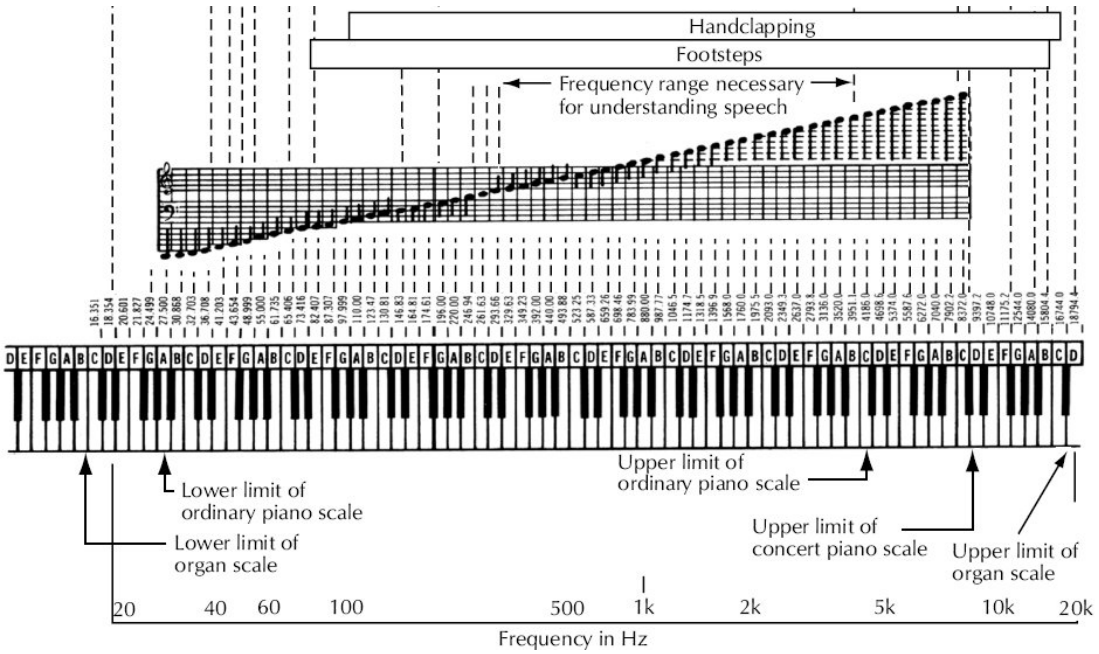

Figure 2.3: Frequency range of clapping

The first measurements with a sample rate of 22kHz generated the following figure (Figure 2.4). Compared to Figure 2.2, which shows the signal from the Particle Board, it can be said that the values for the frequency do not match (Table 2.1). This is logical, since the range of the clapping is large. It was tried to perform a nearly similar clapping to retrieve almost identical data. Since this was not the case, an idea was to check the Android

1 http://www.klangfuzzis.de/showthread.php?679817-Was-hat-in-etwa-wie-viel-hz
2 http://www.recordingblogs.com/wiki/nyquist-shannon-sampling-theorem

Software, if the data was calculated and transformed correctly.

Particle	Android
230Hz	160Hz

Table 2.1: Comparison clapping frequency of Particle Board and Android device

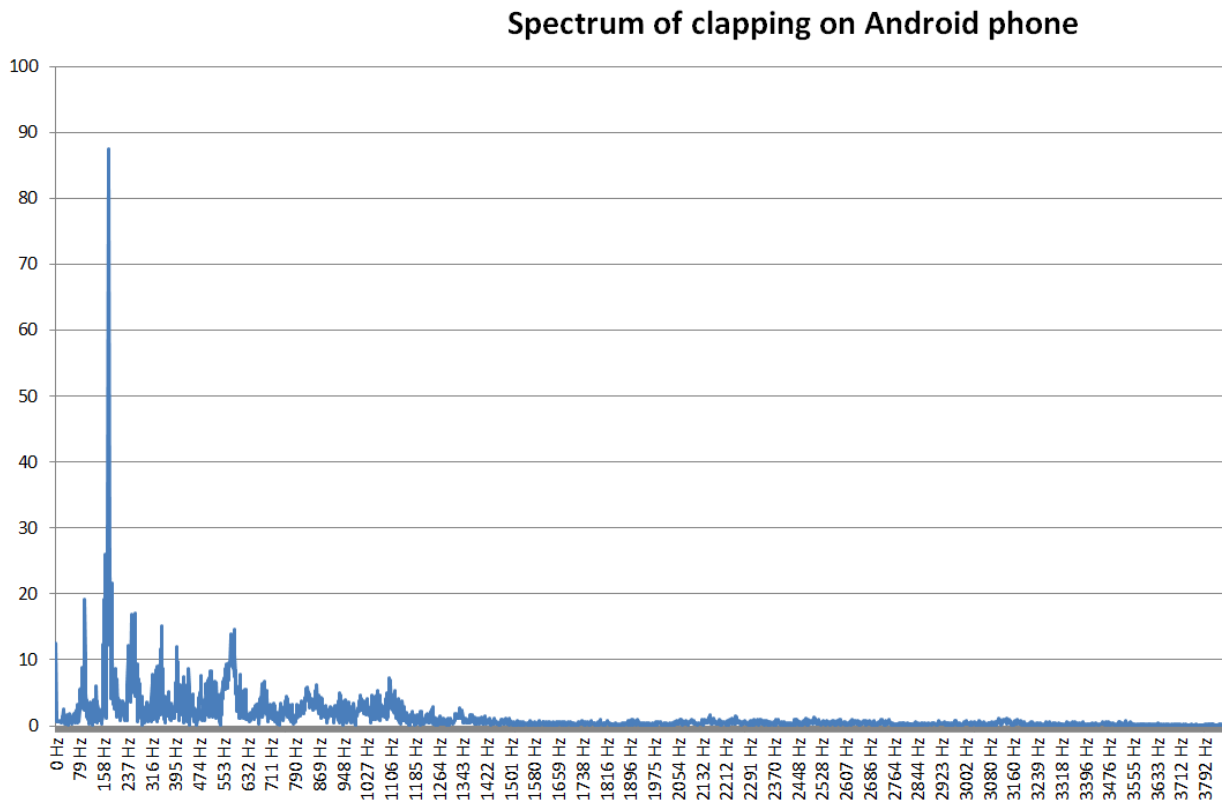

Figure 2.4: Amplitude Spectrum for Clapping with Android device

For this reason, the check of the calculations is done with another tool called iSpectrum, which can be installed on a MacBook. This makes it possible to collect live data from the laptops microphone to view the spectrum.

In order to compare the iSpectrum software and the Android device, a reliable source is needed that provides values for a specific frequency. The idea is to take a sound example

and play it to both systems. This makes it possible to observe which values match the frequency.

Instead of a single sound file, a video is used for testing, which plays all frequencies in the audible range of the human ear. This video is called **20Hz to 20kHz (Human Audio Spectrum)**[1] and emits an increasing frequency from 20Hz to 20kHz. It is thus possible to see whether the values are correctly interpreted with different frequencies in the Android device.

As figures 2.5 and 2.6 show, the tested frequency of 1200Hz is reliably detected by both systems. Live testing with iSpectrum makes it very convenient to make clapping and other noises visible. Therefore, different sounds were tried out to see what the spectrum looks like.

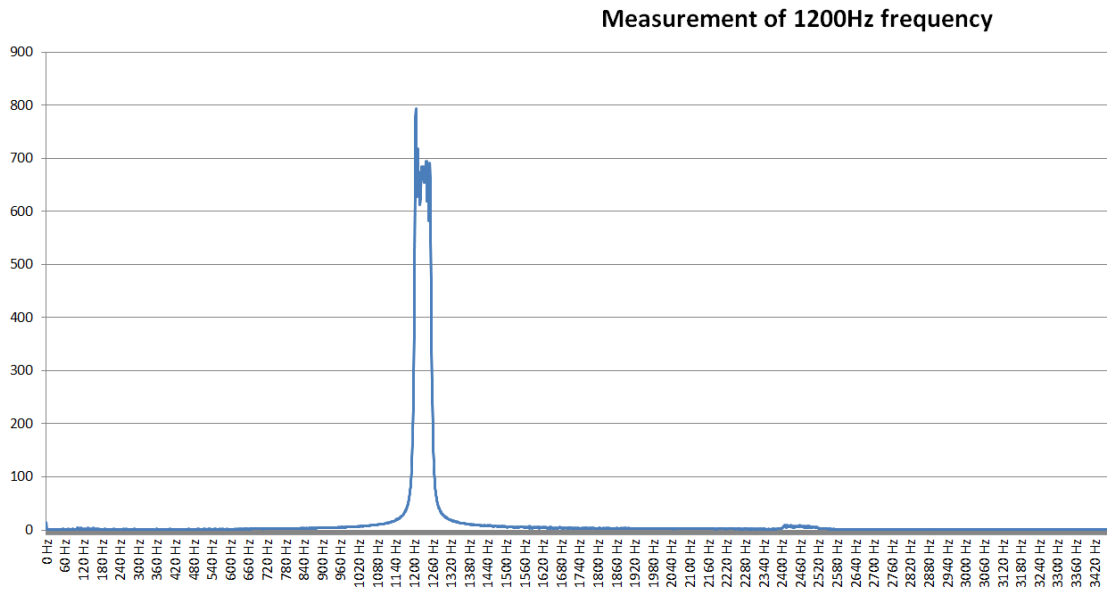

Figure 2.5: Amplitude Spectrum for 1200Hz (Android device)

1 https://www.youtube.com/watch?v=qNf9nzvnd1k

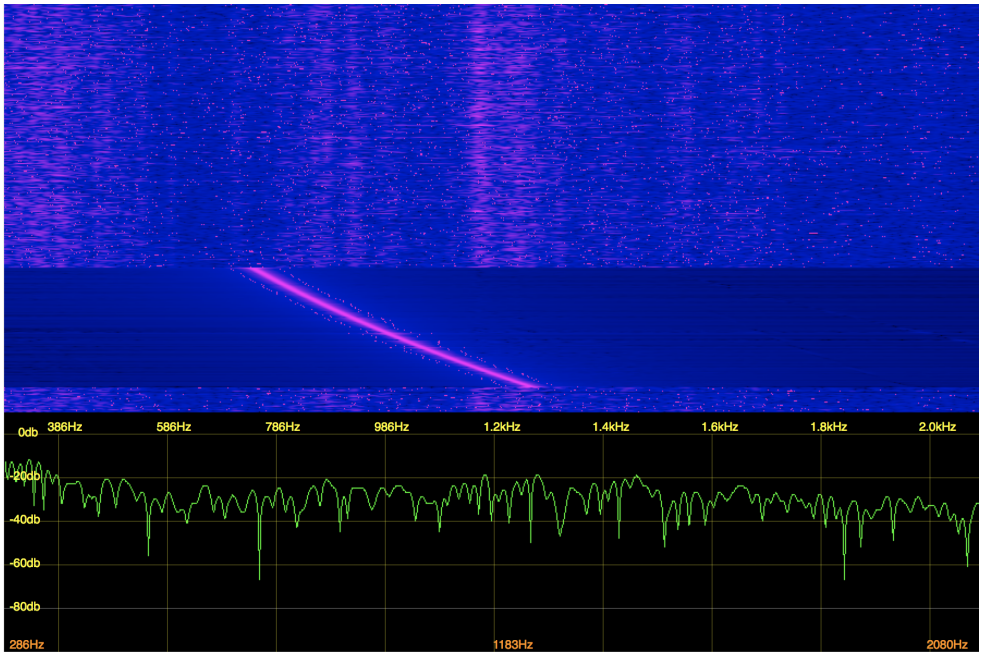

Figure 2.6: Amplitude Spectrum from 750Hz to 1200Hz (iSpectrum)

The yellow circle in Figure 2.7 shows the clapping in a small room. The three sounds detected, which have a green frame, are knocking on the table. It can be seen that these noises occur in different frequency ranges. The clapping echoes a little and the knocking stops abruptly. However, each clapping or tapping can be different from person to person, depending on how the clapping or tapping is performed, or where the person is. Further measurements with various test subjects have shown this. Figure 2.8 shows clapping from a test person in a big room. It can be obtained, that the clap has more echo in a big room compared to a small room. A distinction between clapping to other sounds could possibly be made with the duration of the echo.

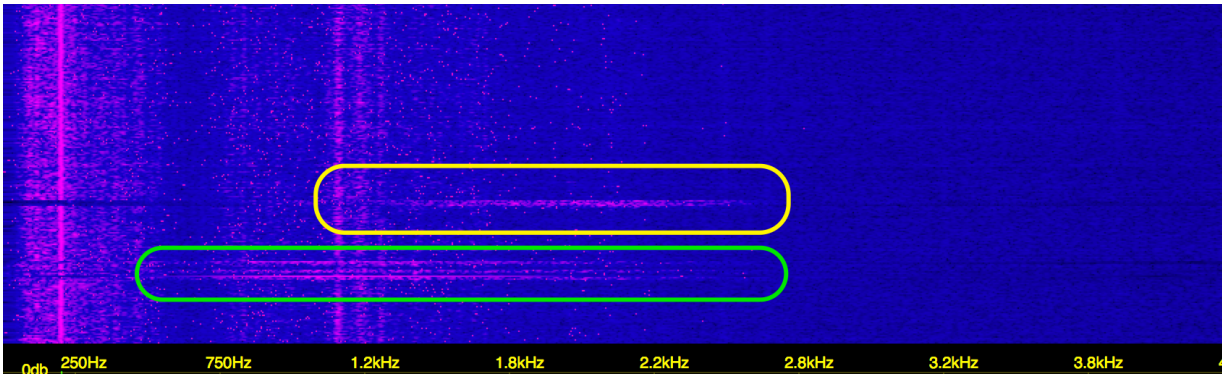

Figure 2.7: Spectrum of clapping and knocking

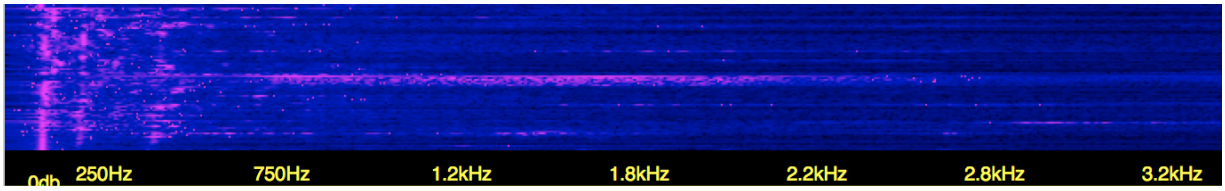

Figure 2.8: Spectrum of clapping in a big room

2.3 Generated insights

So far, the tests have shown that the clapping is in the range of 750Hz to 3kHz very consistently. That means that the first measurements with 10Hz and 1kHz are not usable, since according to Shannon's law it is not possible to measure this high frequencies with that small sample rate. That is why the first assumptions are completely wrong. Only the measurements with the Android device with a sampling rate of 22kHz provided clearer and more consistent results. Another assumption is that the measurements in Figure 2.4 give a false impression, since the strongest frequency is in the range around 160Hz. It is possible that the accumulated ambient noise has a higher amplitude than the clapping. This means that the frequency for the clapping is not displayed with a high amplitude. A possible solution for this scenario is to use a digital filter to capture and display only the

frequencies of the clapping range. One filter of interest in this case would be the bandpass filter[1], which provides exactly the needed behaviour. The band can therefore be limited from 700Hz to 3kHz.

It is of course very complicated to evaluate every single signal to determine if it is a clap. A single algorithm is therefore not capable of analyzing a wide range of clap frequencies. Therefore it makes sense to evaluate the filtered data with a neural network. However, it should be noted that it takes a lot of training to get a reliable data analysis. This means that many thousands of clap patterns must first be recorded in order to give the neural network the possibility to interpret the clap correctly. TensorFlow[2] offers a widely used open source framework for machine learning and provides a good solution strategy for analysing clap signals.

1 https://www.electronics-tutorials.ws/filter/filter_4.html
2 https://www.tensorflow.org

3 Second Prototype (Android)

3.1 Research for clap detection in android

Java is known to have many open source libraries. For the processing of audio signals, the open source library Tarsos[1] was a good choice. Among other things, this offers a ready-made percussion detection which enables to detect sudden peaks in a frequency. However, Tarsos also offers more basic functionalities, such as transforming an audio signal into the frequency domain using Fast Fourier Transformation.

In order to implement and test the necessary functionality for the state machine and the UI, the percussions detection from Tarsos was initially used to detect a loud noise.

3.2 Implementation for data analysis

Some code was soley written for getting data out of the app and was removed later for the final app.

1 https://github.com/JorenSix/TarsosDSP

3.2.1 Google Drive

When working on the first prototype with the Particle Photon Board, the roundtrip time required to get the data from the board to a PC, where we can analyze it, was particularly noticeable. For this reason, an automatic upload of the CSV data to Google Drive has been implemented. This made it possible to quickly transfer all recorded data to a central location and analyze it from there on a PC.

3.2.2 CSV Button

Another functionality that was only necessary for the initial phase is the recording of the audio signal for a certain period of time and afterwards writing the transformed (FFT) data into a CSV file.

An extra button was added to the page, which adds a new AudioProcessor to the AudioDispatcher, which then writes the data to the mobile phone. The existing CSVWriter class was used and adapted for this.

Unfortunately it turned out that you can't call the method AudioDispatcherFactory from the Default Microphone twice to run two dispatchers with different buffer sizes at the same time. For the recording of test data a longer period of time is useful over which the FFT is then applied. A buffer size of 3*sampling size was needed, to allow 3 seconds of audio recording per test. For real application, however, shorter periods of time are needed to detect a double clap and to keep the delay of the detection small.

3.3 Implementation

3.3.1 Clap Detector

The ClapDetector creates a new AudioDispatcher with a buffer size of 1024 bytes and a sample rate of 20kh and registers itself as AudioProcessor. Thus, the AudioDispatcher calls the AudioProcessor Process Handler with 1024 amplitude values every 0.02 seconds. For each time period, a Fast Fourier transformation is created using the Tarsos library. The number of peaks is counted for each new frequency spectrum. The peaks are detected by iterating over each frequency and calculating the decibel values for each magnitude. The decibel value is then compared with a fixed threshold. All decibel values that exceed the threshold are counted. The ClapDetector stores only the last two PeakCounter values. To detect a clap, the second last peak counter must be smaller than the last and the current one smaller than the last. If this is the case, an increase and decrease has been detected in several frequency ranges and the handler is called.

3.3.2 State-machine

A state machine was implemented to represent the different states of the app. The following diagram shows the implemented state machine.

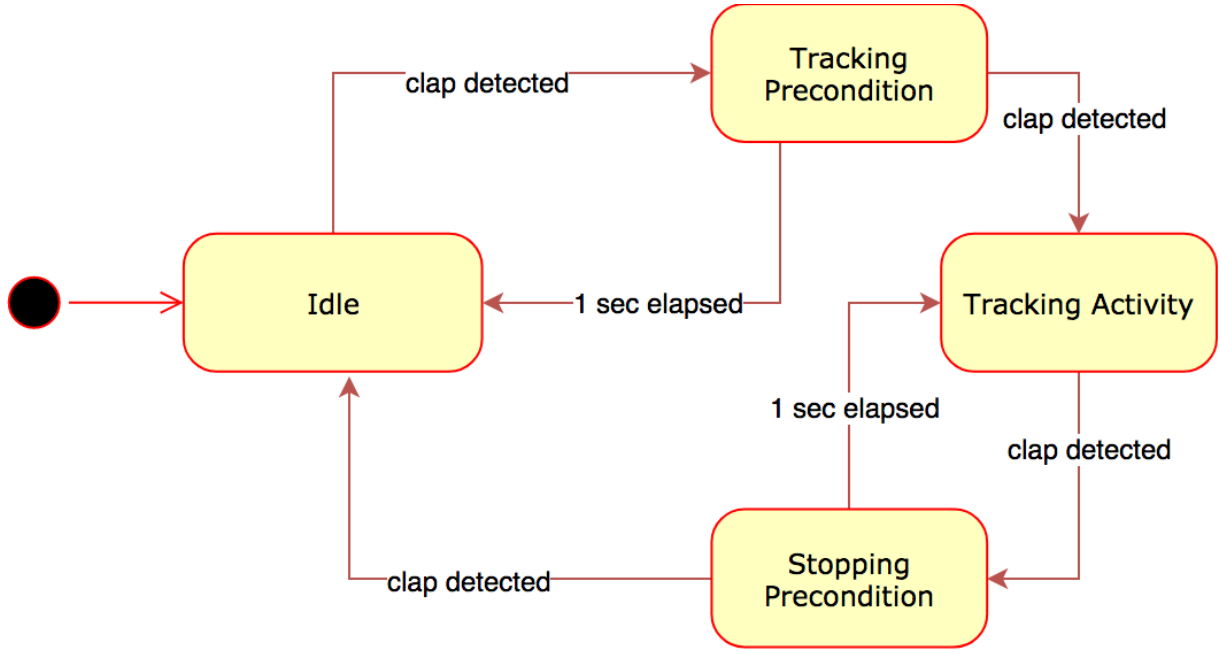

Figure 3.1: UML State Chart Diagram for the state machine

There are a total of four states in which the app can be in.

- Idle: The initial state when starting the app and the state after an activity tracking has been completed.
- StartPrecondition: If the app is in the idle state and a clap is detected, the state machine switches to this state. When switching to this state, a timer is started which defines the time window in which the second clap must occur in order to switch the state to TrackingActivity. If the timer expires before another clap is detected, the state machine switches back to the idle state.
- TrackingActivity: After a second clap is detected while the timer of the start precondition has not yet elapsed, the state machine changes to this state and starts capturing the time by saving a time stamp.

- StoppingPrecondition: If the state machine is in the TrackingActivity state and a clap occurs, then the state machine switches to this state, which behaves in the same way as the StartingPrecondition, except that on a successful second clap, it changes to the idle state and the tracking of the current activity is ended.

3.3.3 Architecture

This Chapter describes the architecture for the Digital Life Tracking App. The following figure 3.2 shows the class diagram with the most important classes:

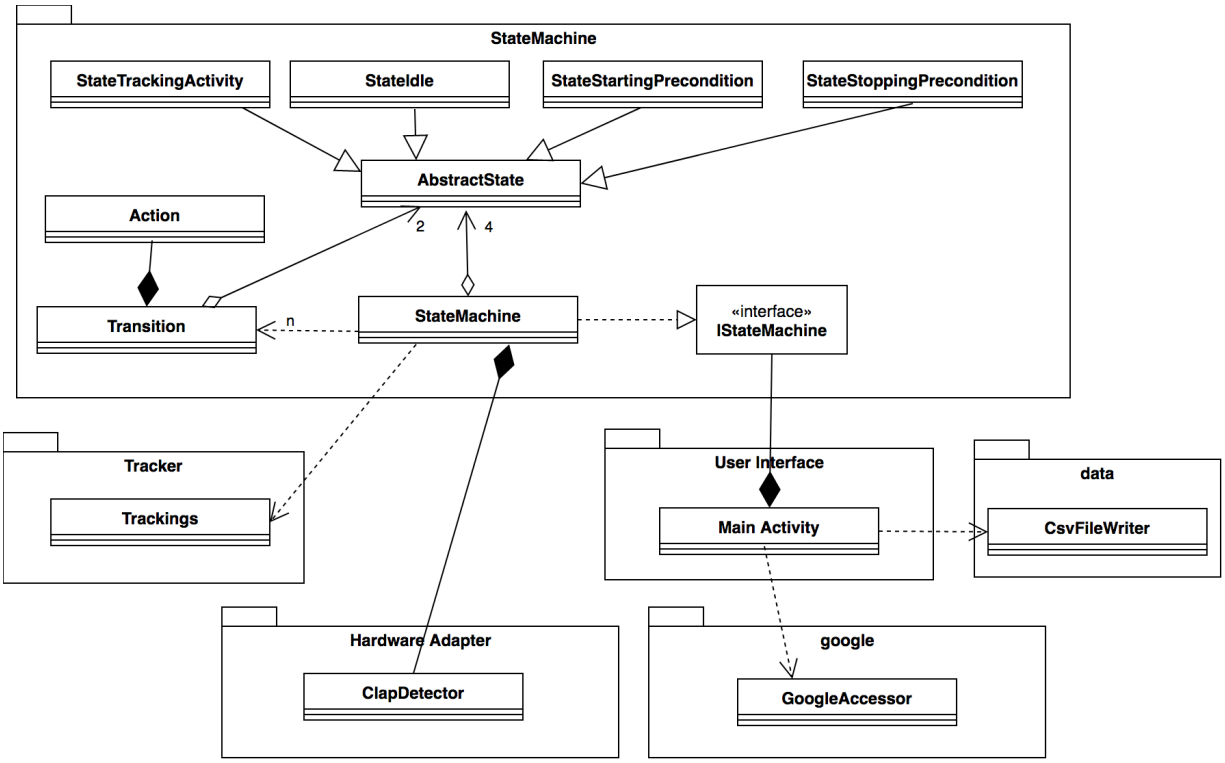

Figure 3.2: UML Class Diagram

StateMachine

The state machine presented in the previous chapter takes care of the logic of the app and switches back and forth between states when registering the corresponding event. The Statemachine class is the center of the package of the same name and performs the tasks described in the previous chapter. It holds instances of all four states and a configuration of all possible transitions between the individual states. Each transition in the configuration is also linked to an action that determines the event that triggers the transition. Each transition also defines the previous state and the next state. When a transition is triggered, other operations are also executed that either adapt UI elements to the new state or execute other functions that are important for parts of the business logic.

Hardware Adapter

The hardware adapter package includes the ClapDetector class, which uses the Smart-Phone's microphone to listen for ambient sounds and tries to detect a clap from the recorded frequencies. When a clap is detected, a handler is triggered in the state machine. This triggers a transition between two states in the state machine. This is illustrated in figure 3.1.

Tracker

One of the operations triggered by the transitions in the state machine is to start or stop the tracker, which stores the activities of the user together with the measured times in a list.

User Interface

The User Interface package contains the main activity of the app, which initializes the other functions of the program including the state machine. The goal was that the Main Activity class should contain as little business logic as possible. The business logic has been implemented almost completely in the state machine. If possible, the class should only provide and control the UI elements of the app. It also controls the permissions request to the user, which the user must give in order for the apps to work properly.

Google

This package contains the GoogleAccessor class, which was initially designed to upload the test data stored on the smartphone via the CSVWriter class to the Google Cloud. Once enough data has been collected, this functionality has been disabled for the release of the app. The reason for this was that access to the cloud required signed APK, which had nothing to do with the pure functionality of the app. Commenting out the code of this feature should avoid unnecessary problems when starting the app.

Data

The data package contains the CsvFileWriter class, which stores the frequencies recorded by the ClapDetector in CSV files on the memory of the smartphone. This functionality was used during development to collect test data for later analysis.

3.3.4 UI Design

In order to make faster progress in the development of the app, a user interface was initially implemented, which was designed for the developers' tasks. The interface should display the transitions between the individual states of the state machine and the events in an output field as logs. The interface should also support the collection of test data that was later uploaded to the Google Cloud for analysis.

At the same time, a design for the future user interface was also created, which was intended to replace the development interface after completion of the basic functionalities of the app. The following figure 3.3 shows mockups for both user interfaces:

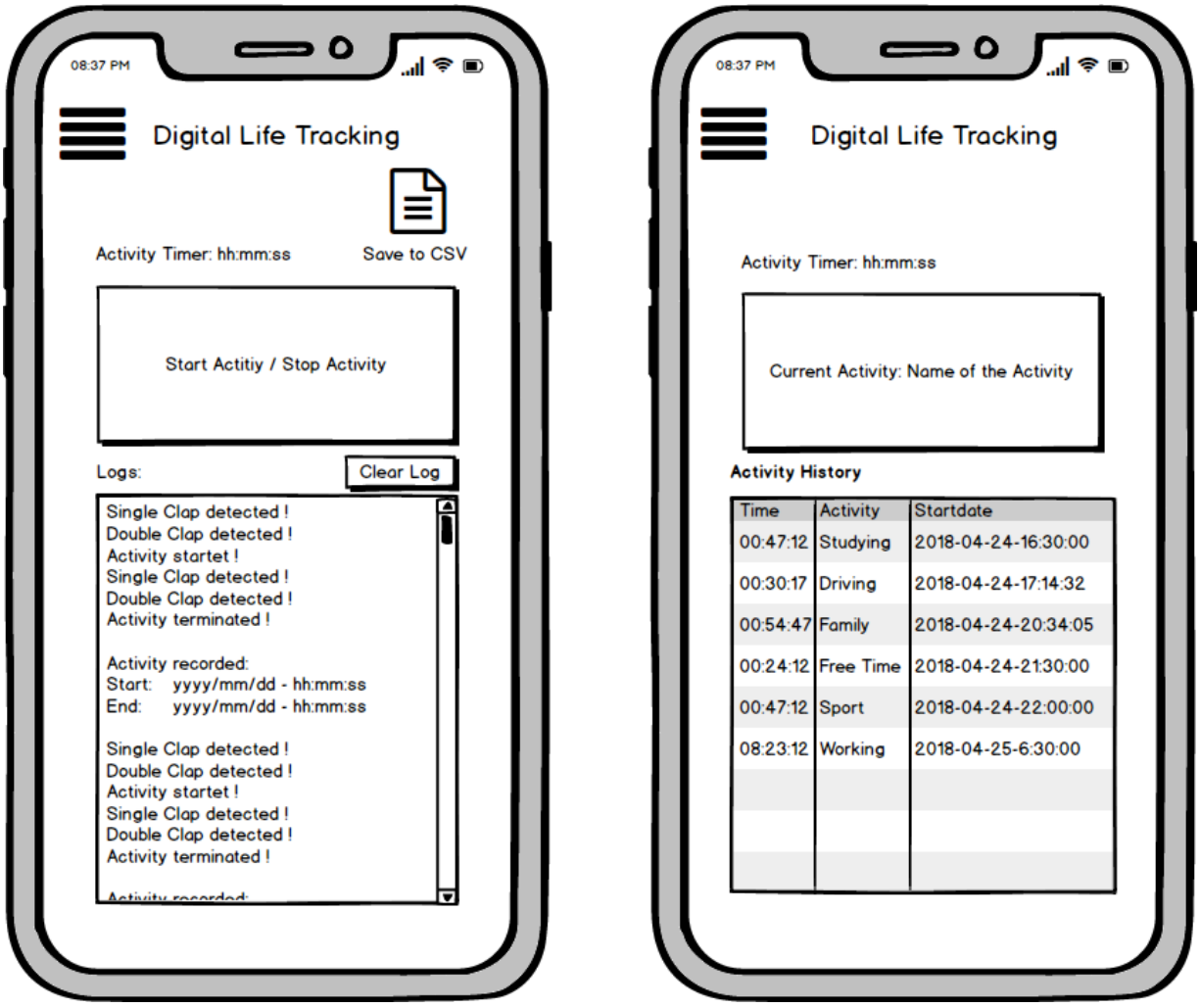

Figure 3.3: Mockups for development UI (left) and for the final UI (right)

The finished user interface should display a timer for the currently tracked activity and the name of the activity. In addition, it should display the last completed activities in a cronologically arranged list, whereby the name of the activity, the start time, as well as the duration of the activation should be visible.

Since there was not enough time to implement the actual user interface during development later on, the app only has the development UI as it is currently available. Figure 3.4 shows

a screenshot of the actual development UI:

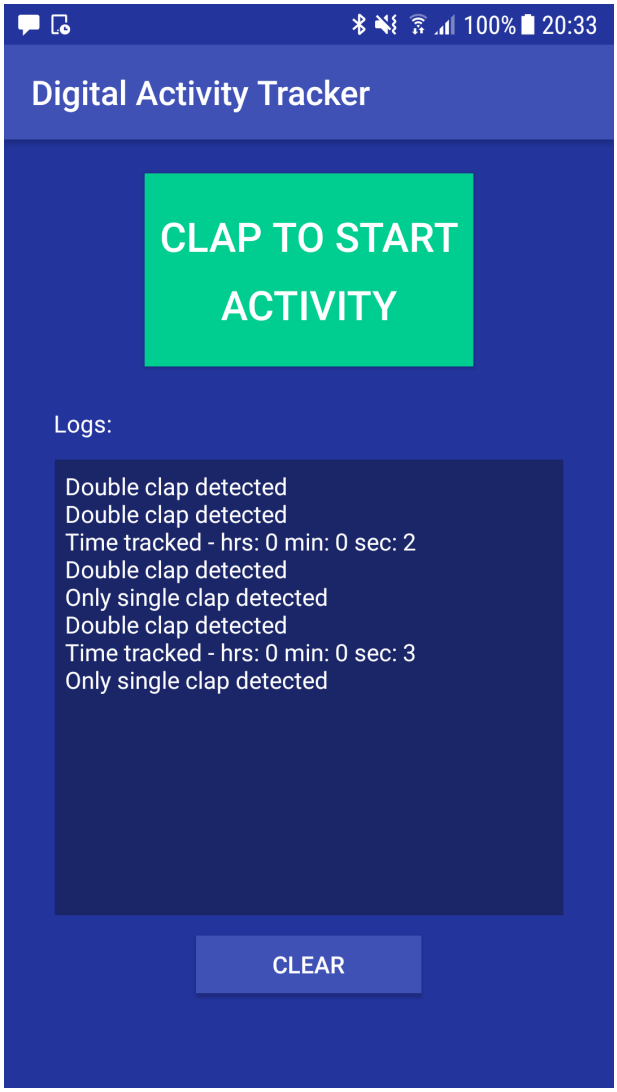

Figure 3.4: Screenshot of the development UI

4 Conclusion

4.1 Current State

4.1.1 Evalutation

- How reliable can our implementation detect clap.
- Benchmark
  - How many time false positives where detected (20x husten, 20x schnipsen, 20 klatschen)
  - Show statistics by trying it out (maybe in different environments (loud, silent rooms, outdoors)

Refer to to evalution part above. State how difficult this was and the time needed to try out more advanced solutions (AI) was not enough.

4.2 Project Outlook

Maybe add more debug functionallity inside the App, be able to not only tweak parameters inside the code, but also with UI Controls inside the app.

Whistling detection instead of clapping.