NTNU – Trondheim
Norwegian University of
Science and Technology

IT2901 - Informatics Project II

Project Report - Group 11
Colorophone

Lars-Kristian Dahl
Daniel Eriksen
Emil Orvik Kollstrøm
Daniel Martin Lundeby
Espen Sivertsen
Henry Skorpe Sjøen
Philipp Zirpins

July 14, 2017

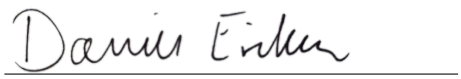
Abstract

In cooperation with Colorophone, a system to aid blind people in navigating using sound has been made. The project constitutes the course "IT2901 Informatics Project II", which is the final project for students doing a Bachelor's degree program in Informatics at the Norwegian University of Science and Technology (NTNU). The app works by the principle of color sonification, that is, conversion of colors into non-speech sound. The app processes color from an input source such as a camera, and produces sound that can be interpreted by humans, and ultimately be used by blind people for navigation and object recognition. Colorophone is developing an algorithm focusing on the encoding of color information. The primary focus has been developing an app for researchers, allowing them to tune parameters and test variations of the algorithm. Also, a system to detect Bluetooth beacons has been made as a proof of concept. The final product can be downloaded as an alpha version from Google Play Store [12]. A user manual for the app can be found in appendix G, and a development guide for future developers can be found in appendix H.

Preface

We would like to thank the Colorophone team and in particular Dominik Osiński, for the excellent cooperation and helpful feedback, and for his wholehearted commitment to the project. We wish all the best for the continuation of the Colorophone project. We also want to thank our supervisor Lemei Zhang for valuable help with the project and feedback on this report throughout the process. Special thanks also to the user testers who have contributed to improving the usability of the app, and to Nordic Semiconductor for sending beacon development kits free of charge.

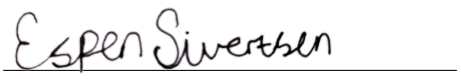
Lars-Kristian Dahl



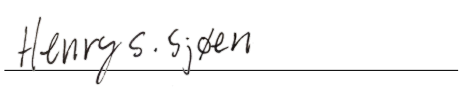
Daniel Eriksen

Emil Orvik Kollstrøm

Daniel Martin Lundeby

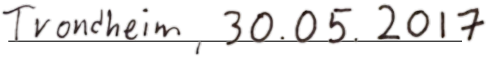


Espen Sivertsen



Henry Skorpe Sjøen

Philipp Zirpins



Place, date

Contents

List of Tables

List of Figures

1 Introduction

1.1 Glossary

Table 1: Glossary

Term	Definition
Android GC	Android Java Garbage Collector.
ARGB	RGB with additional channel "alpha", indicating level of transparency.
ARGB8888	Standard ARGB format indicating that each channel is represented by an 8-bit number.
CLRF	Codename for "Colorophone". Used as prefix for the subprojects, "CLRF Navigator" and "CLRF Beacons".
Developers	Software developers, that is, developers that have access to and can modify the source code.
End-users	Regular users that will use the products to help them perceive colors by sound.
Feature branch	Branch containing any code that is to be integrated to the master branch.
FR	Functional requirement.
FPS	Frames per second. Standard measurement of frame rate, that is, the frequency at which images are captured or displayed.
Gamification	The application of game-design elements and game principles in non-game contexts.
Master branch	Main Git branch used for development code.
NDK	Native Development Kit. Library that allows for use of native-code languages such as C and C++.
NFR	Non-functional requirement.
Researchers	Customer and researchers that will use the products for research and product development.
Proof of concept	A simple prototype of an idea, to demonstrate its feasibility.
RGB	The color channels "red", "green" and "blue", which can be combined to create any color.
SDK	Software Development Kit. A set of software development tools that allows for creation of applications.
Sonification	Use of non-speech audio to perceptualize information. "Color sonification" refers specifically to color information. In this report, these terms are used interchangeably.
UVC	USB Video Class, class of devices capable of streaming video, including web cameras.

1.2 Project goal

The overall goal of this project is developing an affordable system that will aid blind people in overcoming difficulties caused by their blindness. The system should utilize the principle of sensory substitution, by converting visual information into sound using algorithms developed by Colorophone.

The system should be centered around an Android application which converts live video into sound, thus enabling users to interpret visual expressions by hearing. The application should contain experimental features utilizing this principle, and it should allow researchers to experiment with parameters affecting the conversion process. Also, a system for integration with a set of distributed Bluetooth beacons should be implemented, allowing the user to obtain information about location which is normally perceived by vision.

The complete scope of the project was not defined before the beginning of the project. Rather, the scope was explored in the first part of the project in collaboration with the customer, and then more or less settled at a meeting with the customer in the middle of the project period, on March 15th. The scope was affected by the fact that the development should be continued after the termination of this project, which has stressed the importance of writing reusable and understandable code as well as understandable documentation.

1.3 Background

There are 39 million blind people in the world [38]. Easing daily tasks like object recognition and independent navigation for blind people can improve life quality significantly. The ability to identify colors can be helpful for several reasons: it gives clues about object identity, it enables blind and sighted to communicate about the visual world and it is important for figure-ground segmentation[26].

According to [52] retinal prostheses that can partially restore vision not only has technical challenges but also economic ones as well. Cheaper alternatives seem to be needed. Sensory substitution devices (SSDs) can be used to implement effective and user-friendly coding methods for transferring sensory information from one sense to the other. The Colorophone method is one such method that codes color and light intensity information into sound[37].

1.4 Colorophone

Colorophone is a team working in the field of sensory substitution to give visually impaired people the ability to experience colors. Colorophone is led by Dominik Osiński, Assistant Professor of the Department of Electronic Systems, who is also the originator

of the product idea. They have previously worked on other sensory substitution solutions like the ISENSE system, which is a project making echolocation easier for humans.

Colorophone develops a device consisting of a pair of glasses connected to a processing unit, for instance, an Android mobile phone. The glasses are equipped with a camera and a bone-conducting headset. When a user connects the glasses to the processing unit, the user experiences the color in front of him in the form of non-speech audio.

The sonification algorithm takes an image as input, processes the color data and uses the result to encode audio channels with different sounds and volume. The algorithm is under development, and needs a way to be tested efficiently.

The Colorophone team consists of more than 20 people in different groups. These groups have different academic background and focus on different aspects of the Colorophone project. These simultaneous projects are under the supervision of leader Dominik Osiński. In addition to the group writing this report, the Colorophone team has groups working with business and marketing, product design, hardware development and scientific research. Regular meetings between these groups are not conducted and the groups are not dependent on each other to finish each group's respective projects successfully.

1.5 Domain research

The customer has already done considerable research on the domain and made available a paper detailing their previous work, experiments and test results [37], which was very important for the group to understand the project. He also made available a Dropbox folder which contained a lot of domain related articles and books. In addition to this, the customer gave the group links to YouTube videos he considered to be relevant, for instance, a video series on color vision by Craig Blackwell, a TEDx video about blindness and a video on describing colors to a blind person by Tommy Edison. All these sources together created a good picture of what the difficulties with being blind is, and how a sensory substitution system could help ease some of them.

1.6 Stakeholders

It is important to be aware of relevant stakeholders, to ensure that the final product is satisfactory for everyone who may have an interest in the project. The most important stakeholders are described below.

Customer Dominik Osiński represents the Colorophone team and acts as the project customer to the development team. Researchers and developers in Colorophone have an interest in the researcher app, which allows experimenting with algorithms and parameters.

End-users End-users are visually impaired people who are using the Colorophone system as an aid in their everyday life. The end-users have a natural interest in the project and should be kept in mind, but they are not the main focus for this project.

Project team The project development team will be evaluated and graded on the project, and the project serves as a showcase and portfolio for the future.

Norwegian University of Science and Technology (NTNU) The Colorophone team is associated with NTNU and could potentially attract some attention to the university.

1.7 Report structure

Section 2 starts out with a description of the group process. The section starts with the prestudy phase, followed by a detailed description of the chosen development methodology and tools. In the end, import changes to the process are discussed. Then, section 3 gives an overview of the project structure, while section 4 and 5 describe each of the separate subprojects. These sections contain information about the prestudy phase, requirements, architecture and design and possible extensions for the future. In section 6, the testing phase is described, including the test strategy and important test results. A detailed description of the progress in each sprint is included in section 7. Finally, in section 8, important aspects of the product and process are discussed and reflections are made.

2 Process

This section describes the process of the project. It starts out with the process prestudy phase, followed by a detailed description of the chosen methodology. Then, the project management tools and development tools are explained.

2.1 Process prestudy

At the initial phase of the project, a prestudy phase was conducted. The group discussed several different approaches to the development process and quickly agreed to use an agile development process. The product should be utilized in a research and product development setting and therefore needs to be able to support rapidly changing requirements. Requests for additional or changing features must be handled in an agile manner, and some of the features should ideally be prototyped first.

There are many different agile development methods, such as Extreme Programming (XP), Scrum, Dynamic Systems Development Method (DSDM), Lean Development and Feature-Driven Development (FDD). The group members had good prior experiences with XP and Scrum, so it was decided to use these as a base to avoid having to learn an entirely new process during the project, and they are also easier to adapt than the other more rigorous frameworks.

The group then continued to discuss working processes, version control, project management and supporting tools. The outcome of this phase was a project guideline document, which is attached in Appendix B. This document was intended to always be updated to reflect the current working process and policies, and it has helped the group in achieving consistency in the work and process. Having a prestudy phase and getting an overview of different alternatives before the real planning and implementation started, made it easier to make decisions together with the customer and in the planning phase.

2.2 Project team

The project team consists of 7 developers. Most group members knew each other beforehand from earlier group work. All members have attended basic computer science courses, and have basic programming experience. Each group member's relevant experience prior to the project is given below.

Lars-Kristian Dahl Expert knowledge of Java, some prior experience with Android. Theoretical and practical experience with process methodologies, architecture and system design from university courses, open source and personal projects.

Daniel Eriksen Experience with Java programming, agile methodologies and some experience with Android programming from earlier courses.

Emil Orvik Kollstrøm Experience with agile methodologies, Java programming and system design from earlier courses and personal projects.

Daniel Martin Lundeby Experience with Java programming from private projects and university courses, as well as agile methodologies.

Espen Sivertsen Experience with Java programming and agile methodologies from earlier courses. Some experience with Android from personal projects.

Henry Skorpe Sjøen Experience with programming and teamwork, gained from previous informatics courses at NTNU. Some personal projects in Java, Python, and JavaScript, but limited experience with the Android ecosystem.

Philipp Zirpins Experience with Java, databases and software development related practices, tools and frameworks gained from university courses and non-course related projects. No prior experience with Android or Bluetooth low energy beacons.

2.3 Responsibilities

All team members are responsible for the success of the project and their own contribution. All team members have agreed upon and signed a team contract, attached in appendix A. However, some group members have been assigned additional responsibilities, described below.

Project manager Lars-Kristian Dahl

- Overall project management and coordination
- External contact with customer and other external entities
- Handle external issues

Team manager Henry Skorpe Sjøen

- Meeting leader and coordinator
- Internal issues and conflicts

- Team-building

Report manager Daniel Martin Lundeby

- Maintain overview of project report
- Manage report-related deadlines
- Overall quality control for the report
- Write and publish meeting minutes

Team leader CLRF Navigator Lars-Kristian Dahl

- Manage overall quality and control of the subproject

Team leader CLRF Beacons Emil Orvik Kollstrøm

- Manage overall quality and control of the subproject

2.4 Development methodology

The group and customer agreed on using an agile development methodology mainly inspired by Scrum, customized to fit the project needs. It also has some features and inspiration borrowed from XP, as described in the following sections. Due to customer request, a Trello Kanban board was also used initially, serving as a product backlog. It was however challenging and tedious to manage and synchronize all the various tools. In agreement with the customer, all product backlog items were moved to Waffle, which used a structure similar to Kanban, and Trello was discarded. This process has been described in detail in section 2.10.1.

2.4.1 Management practices

Though some specific management practices may vary between the different variations of agile methods, there are also many standard practices and principles [56]. All agile processes have iterations of some defined length, and these iterations must be planned. The team began the project by identifying and prioritizing a superset of features in close cooperation with the customer. This was followed by planning a roughly estimated release date and the first iteration. From then on, iteration by iteration, continuous planning was used to refine the release plan as new information was discovered and new features were requested or existing ones changed.

This section describes some of the common practices and principles used in the process, and how they were implemented and adapted to the project.

Feature estimation Feature estimation was based mostly on Scrum, where a feature is called a Backlog Item and allows non-feature items such as "configure y" and "investigate option x". Scrum also tends to allow items to be larger-grained than some other more feature-oriented methodologies [55]. In terms of estimation units, work units were used, specifically points. The team used the practice of relative estimation for features, using a set of predefined estimation categories {1,2,3,5,8,13}. All features were estimated in terms of these categories, and if any features were deemed to be larger than 13 they had to be broken up into smaller items. The significant time and effort saved by planning with this type of process often outweigh any costs of imprecise estimates [54].

Iteration planning Each iteration initially lasted one week, to allow the team more rapid prototyping and feedback from the customer. As the project evolved, this was adjusted to two weeks. The team had a planning meeting at the start of every iteration. During this session, the most prioritized items from the product backlog were broken down and split up into smaller and more specific technical tasks. The meetings, combined with retrospective and sprint review, usually lasted 2 hours and was held on-site at the customer's location, sometimes including the customer.

Release planning A preliminary release planning meeting was held at the start of the project, where the team planned a rough estimate for deadlines and which features could be delivered within these deadlines.

Iteration tracking Iteration tracking was done through Waffle and Toggl. Waffle tracks the points of each task, as well as when tasks are started and completed. Toggl was used to monitor the actual time spent on each task, which could then be used to compare and improve estimates for the following iterations.

2.4.2 Programmer practices

A set of agile programmer practices has emerged and proven to enable more frequent delivery with higher quality [57]. These programmer best practices help programmers and even the code itself to become more agile. This section describes some of the best practices that were used during this project, and how they were used and implemented in practice. A more detailed description of the actual guidelines followed by the team can be found in Appendix B.

Refactoring Code refactoring is the process of clarifying and simplifying the design of existing code, without changing its behavior [58]. The team performed continuous refactoring of the code throughout the project. Several parts of the code have been refactored multiple times, slightly improving the system design each time. Examples of this are refactoring to Model-View-Presenter (MVP) architecture, the introduction of the Dagger2 dependency injection library and continuous improvement and separation of concerns in the sonification pipeline.

Continuous integration Continuous integration (CI) involves automatic testing and producing a clean build of the application, usually at fixed times or through some trigger mechanism. The team used Travis CI to do automatic builds and testing for each pull request and change to the master branch on GitHub.

Simple design As an agile team, the team was under pressure to deliver working code at the end of each iteration, while also allowing potentially radical requirement changes at any point during the process. For this reason, it was critical that the code was extensible and easily modifiable, that is, the extent to which the code can be easily modified, maintained and extended. Previously, refactoring was mentioned as one factor of keeping the code extensible. The other key factor is code simplicity. Extensibility seems to be inversely proportional to design complexity [61]. The code was kept as simple as possible, and the team heeded the saying "You Aren't Gonna Need It" (YAGNI) from Extreme Programming. If it was discovered that a more complex implementation was needed, the relevant code was refactored to provide the needed functionality. An example of this is the sonification pipeline, which initially was implemented as the simplest version that could work. This was later split into separate components with configuration, callbacks and more design complex pattern as needed. Some of these needs were the need to use the process in other scenarios such as the Image Viewer or the Game feature or to provide information about the input and output to the GUI.

Pair programming Pairing is by far the most controversial and least universally-embraced of the agile programmer practices [60]. This is a practice most prevalent in XP, but the team has had good experiences with this practice in previous projects and wanted to utilize it for this project too. It was decided to use it on occasion as both a learning tool and to improve code design and robustness for vital parts.

Coding standard Programmers in a group should all adhere to a single agile coding standard, including everything from tabs vs. spaces and curly bracket placement to naming conventions for things like classes, methods, and interfaces. It is easier to maintain and extend the code, to refactor it, and to reconcile integration conflicts, if a common standard is applied consistently throughout[59]. The team created a common coding standard at the start of the project, see Appendix B. They were based on the concepts in [30].

2.5 Definition of done

A feature is considered done when the following steps have been completed in order:

1. Feature has been implemented on a separate branch
2. Pull request has been created
3. Pull request has passed all tests
4. Pull request has been manually tested and approved by at least one team member
5. Pull request has been merged to master

Step 2 is enforced by the tools by disallowing direct pushes to the master branch. Step 3 and 4 are enforced by requiring all tests to pass and at least one approved review before a pull request can be merged to the master branch.

2.6 Iteration breakdown

2.6.1 Sprint meetings

Sprint meetings were held every Wednesday 08:15-10:00 at Kalvskinnet, in proximity of the customer. The sprint meetings featured a sprint retrospective, sprint review and sprint planning. The customer was welcome to join the meeting and participated roughly every other week.

2.6.2 Status meetings

Status meetings were held every Friday at 12:15-13:00. These meetings were used for quick updates on the work over the last two days since the sprint meeting and used to resolve any potential issues that had come up during that time.

2.6.3 Supervisor meetings

Meetings with the supervisor were held every other Friday at 15:00. During these meetings, the group received advice on administrative matter, and the team's progress was discussed.

2.6.4 Work meetings

As a main rule, the group worked together on Wednesdays after the sprint meeting. Extra meetings were scheduled when needed, mainly for people working on the same feature.

2.6.5 Project leader meetings

Project leader meetings were held roughly every other week, but less frequently in the last part of the project period. Members from different informatics bachelor projects participated, and the meeting were led by the lecturer of the course. The purpose was to discuss the project progress, identify problems and how they could be solved, as well as asking for advice.

2.7 Development environment and tools

2.7.1 Development environment

The product is based on the Android ecosystem, and thus Java has been the primary programming language. A range of libraries have been used to ease the process and improve and simplify the code. Following is a list of libraries that have been used.

RenderScript Offers increased performance of computationally intensive tasks through low-level libraries. [6]

ButterKnife Used to inject Android Views and bind them to fields and methods using annotations. It is used to avoid writing boilerplate code. [62]

UVCCamera Used to access USB video device class (UVC) web cameras on Android phones [45]. Enables the use of an external web camera in addition to the built-in Android camera.

Dagger2 A dependency injector for Android and Java [48]. This lets the programmer use a concept called Inversion of Control where class dependencies are configured outside of the class. This makes it easier to reuse classes and test them. [63]

Glide A media management and image loading framework that can be used for media decoding, disk caching, and memory and resource pooling [28]. The library is used to avoid having to implement these features from scratch.

JUnit Testing framework for Java. Used for software tests of small units of the program to verify that individual classes and their methods are logically correct and work as they should. [64]

Mockito Mocking framework for use in software tests. When testing classes that are dependent on other classes it is desirable to use mock objects for the classes that are not being tested, to isolate the test for the unit in question. Mockito makes it easy to create mock objects with minimal boilerplate code. [65]

Robolectric Test framework that enables Android tests to be run within the IDE. Running tests on devices or emulators imply building, deploying and launching the application, which can take minutes. With Robolectric the hassle of running tests on an Android emulator or physical device is avoided. [19]

LeakCanary A memory leak detection library. While running the app in developer mode on a device and if a memory leak is detected a notification is shown on the device that explains what Android activity caused the leak. This makes it easy to discover root causes to memory errors and fix them. [44]

Android Beacon Providing APIs to interact with Beacons from Android phones. Was chosen in favor of others because of its compatibility with different beacon protocols. [43]

2.7.2 Development tools

Android Studio A comprehensive, integrated development environment (IDE). A powerful development tool tailored for Android programming, based on the IntelliJ platform. It was chosen for its features and ease of use.

Git Version control system (VCS). A tool for a group to work together in the same code. Features like branching and merging make collaboration much easier.

GitHub Hosting of VCS. An excellent, free of charge service that all the group members had used before. It is easy to integrate with other development tools, such as Travis CI

and Waffle.

Travis CI Continuous integration service. A powerful tool for testing code as it incorporates automated builds and testing. It has been integrated with GitHub, which allows builds to be triggered by certain Git events. It was configured to run automated builds for all pull requests, and also for each update to the master branch.

Gradle Build tool. The default build system for Android projects.

2.8 Project management tools

Slack Main communication channel within the team and with the customer. This is a popular communication tool which makes it is easy to create channels for different purposes and people, and has the possibility of integration with services like Waffle, Google Calendar, GitHub and Travis CI. It also has a quite good mobile app to keep in touch on the go.

Trello Product backlog. This is a very easy to use tool, which works as a Kanban Board where one can create tasks and move them to different lanes depending on the status. It was used initially to keep track of the biggest features of a subproject, but was later discarded.

Waffle Sprint backlog and planning, integrated with GitHub. This is a more advanced Kanban tool to keep track of issues. It was chosen because of the possibility of GitHub integration and automatic issues tracking.

Google Drive Storage of project-related files. This is a free and easy to use service for hosting a project folder to share and edit files.

Google Docs Informal documents. This makes creating files and writing in the same document within a group very easy.

ShareLatex Formal documents. This tool combines the features of LaTeX and simultaneous work. ShareLatex is an advanced text editor which makes it easier to create and maintain large documents.

Toggl Time tracking. This is a tool to track the time used on different tasks individually and in a team. It was used to see how time spent was split between meetings, implementation, and the report. The main feature is a stopwatch that is started and stopped when working, with the possibility to add work sessions manually.

Google Calendar Events and meeting planning. This tool lets the user create a custom calendar for a team, and integrated with Slack it gives notifications at the start of the day and 15 minutes before an event.

Draw.io Used for making diagrams. This is a free and simple to use tool which lets the user create quite advanced diagrams and charts, like flowcharts or electrical circuits.

SimpleUML UML diagram generator. This is a plugin for Android Studio which can assist the process of creating certain UML diagrams.

Sequence Diagram Generating sequence diagrams. This is a plugin to Android Studio which speeds up the process of creating sequence diagrams.

2.9 Use of project management artifacts

In order to organize the project work well and work efficiently, it is important to use available tools in an appropriate way. This section describes the most important project management tools, and how the group has used them.

2.9.1 Activity plan

During the project, boards on Waffle have been used as activity plans. Waffle is a project management tool integrated with GitHub, which provides automated issue tracking. Waffle organize items into columns in boards much like Trello, with the exception that the process is automated. A screenshot of the Waffle board as of February 15th is shown in figure 1.

Activity plan items may be features, enhancements or bugs. The items were in any of five columns corresponding to the state of the item. Each item had an assignee, a size estimate, and a description. Each item was also assigned a GitHub milestone, described below. An item shown in greater detail, in the context of bug submission, is shown in Appendix C.3.

The activity plan cycle is explained in short here. The "Backlog" column contained all new suggestions for features and enhancements. The items were submitted by the

group, mainly from feature suggestions by the customer. Items that were considered ready for development were broken down into smaller items and moved manually to the "Ready" column. This happened mainly on the weekly sprint planning meetings, but also continuously during each sprint. When a developer started working on an item, the item was referenced in the branch name and automatically moved to the "In Progress" column. Once fulfilled, a pull request was submitted in GitHub. The item then moved to the "Needs Review" column. For the pull request to approved, at least one review by another group member was required. In case something needed improvements, the author was notified, and the code was rewritten. Once approved and merged, the item moved automatically to the "Done" column. This column contained a list of all integrated features, for the benefit of both the group and customer.

GitHub milestones Milestones on GitHub were used to gather backlog items under a product feature, functionality or property. Thus, they had no direct impact on the duration of a subproject, but instead represented major progress points that had to be reached to make sure that the product delivered would meet the customer's requirements and fit the intended use cases. Examples are "Sonification pipeline", "Practice game" and "Image viewer".

Refer to section 8.3.3 for a more in-depth evaluation of the use of Waffle as activity plan.

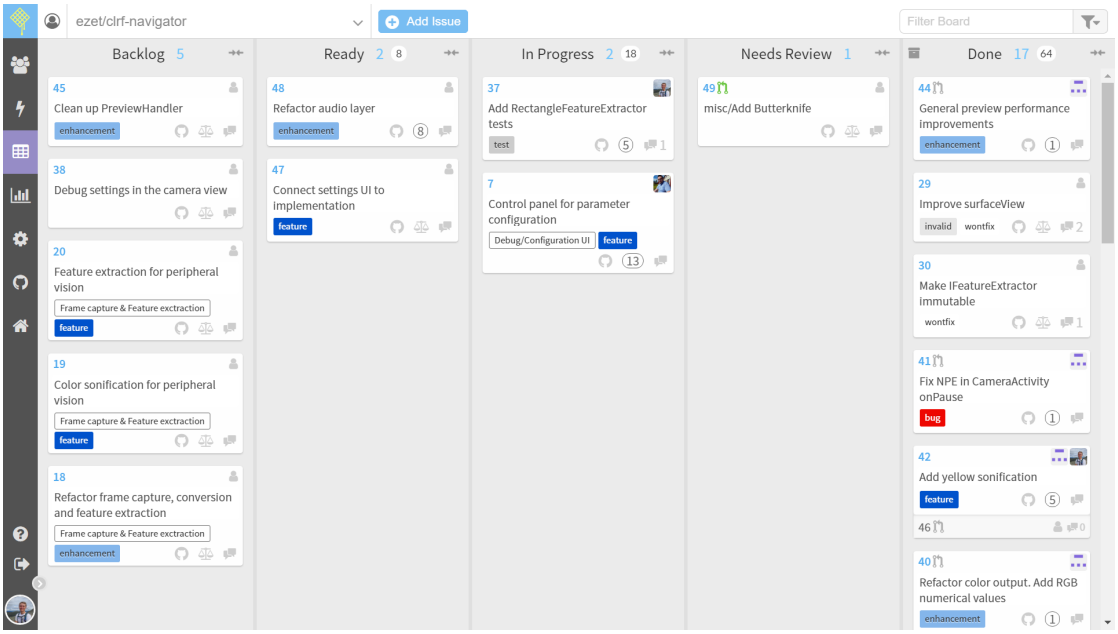


Figure 1: Waffle board as of week 7.

2.9.2 Status reports

Status reports contain information about the current project status, problems and planned work. The status reports have mainly been used as an internal tool to keep track of the development and to communicate the process to the supervisor in an understandable way. They have been delivered ahead of the meetings with the supervisor every other week. See Appendix C.2 for an example of a status report from March 17th.

2.9.3 Work Breakdown Structure (WBS)

WBS is a tool to organize and get an overview of a project. The project has been divided into packages with deliverables and subtasks, see figure 2. The Gantt diagram for this project, shown in Appendix C.1, may not look very typical as the project is split up into several subprojects. Therefore, the diagram differentiates between work packages for the whole project and those that are done in subprojects. This is the most accurate way to show a correct timeline of the project as a whole. The Gantt diagram also shows milestones or dates that have been important for the project, like delivery dates and start or stop events for subprojects.

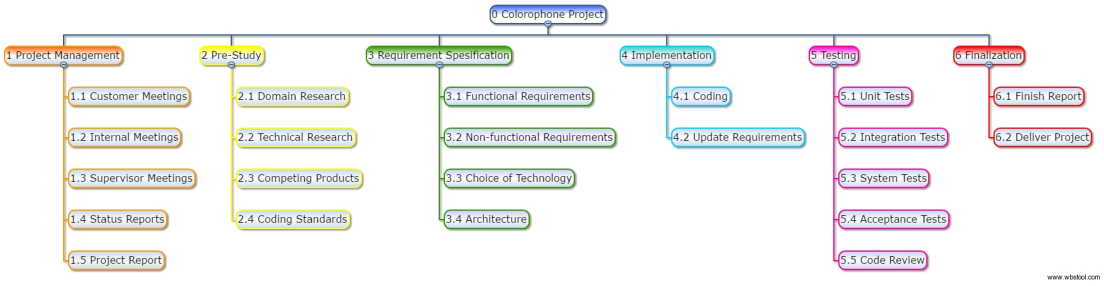


Figure 2: WBS Graph

2.9.4 Risk assessment

The risk assessment is split into two categories. The team risk assessment contains risks related to team members and project management, and can generally be controlled or managed. The project risk assessment includes risks associated with development and external sources and are not as easily controlled. The final risk assessments are shown in figure 3 and 4. The risk assessment applies equally well to both subprojects.

ID	Risk	Mitigation	Action	Responsible	Rank
T7	Team member is ill	If possible, notify team in advance.	If possible, work from home or redelegate tasks.	Every team member	1
T5	Temporary unavailability	If possible, notify team in advance.	Notify team, work remote, redelegate tasks. Add the timespan to calendar with your name.	Every team member	2
T6	Technical software issues	Ask team if in doubt, keep platform up to date, use recommended tooling	Notify and discuss solutions with team,	Lars-Kristian	3
T3	Internal conflicts	Team-building	Talk w/person involved, then Henry. Meeting between the involved and Henry. Then supervisor.	Henry	4
T2	People not showing up at all	No rescheduling of meetings later than 17:00 the day before	Talk w/person involved. Then supervisor.	Henry	5
T1	Late arrival over a period of time	Team-building, notice in plenum	Talk w/person involved	Henry	6
T4	Permanent unavailability	Team-building	Notify supervisor	Henry	7

Figure 3: Final team risk assessment

ID	Risk	Mitigation	Action	Responsible	Rank
P4	Customer unavailability	Plan ahead, establish alternate communication	Reschedule, use alternate communication,	Lars-Kristian	1
P2	Change in project requirements	Thorough initial project analysis and planning, customer contact	Reevaluate requirements, modify project plan	Lars-Kristian	2
P6	Customer changes requirements drastically	Make customer aware that changes	Discuss feasible solutions, present them to the customer.	Henry	3
P3	VCS issues	Good VCS guidelines, guidance	Notify team	Lars-Kristian	4
P5	Technical hardware issues	Communication w/customer and research	Request replacement, adapt	Emil	5
P1	Missed deadline	Progress planning, progress follow-up	Communicate to customer, set new deadline,	Lars-Kristian	6

Figure 4: Final project risk assessment

The risks have been sorted based on what is most relevant for the group, by weighing up consequence and probability. Separate scores for consequence and probability have intentionally not been assigned, as such numeric scores are considered not to have any practical value for the team. The risk assessment is minimal, in the sense that it only contains risks that are considered really relevant. The risk assessment has been updated when needed by adding new risks as new problems have been identified. Occasionally, the risks have been reprioritized to reflect the current state of the project.

Use of the risk assessments The risk assessment has not been used extensively, but it has made group members more aware of which risks may happen and when risks actually happen. Fortunately, only few risk events occurred in practice. In some cases, team members were ill and unable of attending meetings. In those cases, they did the planned action of notifying the team in advance and working from home if able.

Changes to the risk assessment Initially, the risk items had no priority. Priorities were later assigned to make it easier to understand which risks are most severe for the team.

In some cases, risks occurred that were not identified initially. In those cases, the risks were added to the risk assessment to minimize their consequence later in the process. This was the case for the following risks:

- P4 Customer unavailability
- P6 Customer changes requirements drastically
- T7 Team member is ill

2.10 Major project management changes

At the beginning of the project, the group agreed on several project management tools that served all necessary purposes. It quickly became apparent that the choices were well made and that major changes in which tools to use were unlikely to happen. In fact, only one significant project management change was made during the project.

2.10.1 Moving away from Trello

At the first meeting with the customer, the group was introduced to the project and possible features. The project was divided into Trello boards, one for each potential feature, which already contained cards defined by the customer. After having used Trello for a while, the team had some issues with the tool. The team had a discussion with the customer about the usability of Trello, and it was agreed on not to use it since it required too much manual maintenance with regards to keeping it up to date.

The customer understood that the team might know tools that would fit the current situation better than Trello. Thus, it was agreed on using Waffle as activity plan, which allowed for easier maintenance of the backlog and keeping everything up to date and synchronized.

3 Project Overview

This section gives an overview of the project structure and its evolution since the start of the project. As previously noted, the customer initially proposed many different ideas for projects and features. During the lifetime of the project, some ideas and features have been discarded due to technical limitations, feasibility, customer value and so on. At the same time, new ideas and features have been proposed or requested, which have been considered and in some cases implemented in the final product.

3.1 Structure

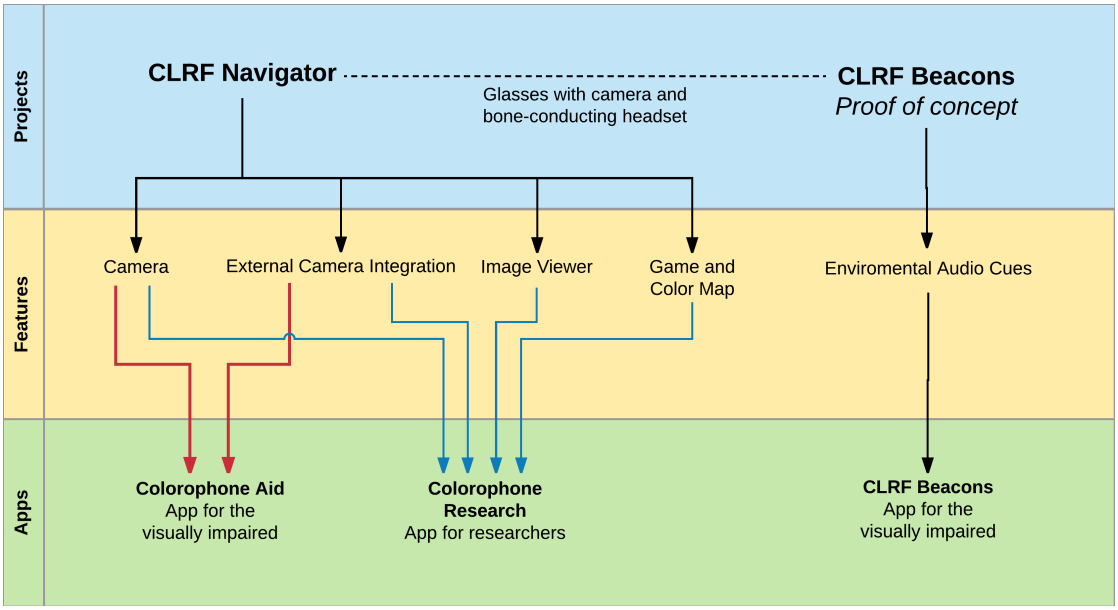


Figure 5: Colorophone final project overview

The final project structure is depicted in figure 5, and was finalized roughly six weeks after the project started. Refer to section 3.3 for a description of the changes. The final structure consists of two separate subprojects, internally named ”CLRF Navigator” and ”CLRF Beacons”.

The decision to split the project into two completely separate subprojects was made because the customer had ideas for two different concepts, and combining them in the same application would introduce unnecessary technical complications to the development process. The CLRF Beacons project relies on completely different technologies when compared to CLRF Navigator, such as Bluetooth, Text-to-Speech and databases. The applications themselves would also serve very different purposes, with CLRF Beacons being a prototype while CLRF Navigator is meant as a production application and

platform for further development and research.

This also made it easier to split the work within the group, as it introduced the possibility of having two completely separate teams that could work more or less independently of each other on separate projects. This was especially helpful during the early stages of the projects as it would have been difficult to effectively utilize a comparatively large team of 7 developers for the CLRF Navigator project while the core components were still being developed.

It was clear that the CLRF Beacons subproject was meant as a proof of concept, and therefore had a clear goal and likely would require less time to complete than the CLRF Navigator subproject. The decision was made to assign three developers to work on this subproject until completion, while the rest were assigned to CLRF Navigator.

After the CLRF Beacons subproject successfully finished on March 8th, CLRF Navigator had reached a stage where the core platform was stable, and independent features were being added to the application. This let teams of two and three developers work on separate features independently, as can be seen in the Features section of Figure 5.

3.2 Projects

3.2.1 CLRF Navigator

CLRF Navigator is the most important subproject as desired by the customer, accounting for roughly 80% of the group's effort. The core of this project is color sonification, and a range of features are built around this concept.

The customer initially proposed two different product flavors:

Colorophone Research is meant for further research and development, with the option to change all parameters related to the sonification and other relevant options process through run-time settings. It is a superset of all other flavors.

Colorophone Aid is meant for visually impaired users, providing simplified access to the Camera feature with limited or no user settings.

As the idea of a sonification game was introduced, this also introduced the concept of a third application flavor:

Colorophone Vision is meant for regular users, that might want to use the application for entertainment or demonstration to other users. This version can also be a valuable tool for collecting data and feedback from other users. It should focus on the sonification

game, but also provide access to Camera and Image Viewer and a limited set of options and good default settings.

The focus of this project has been the Research flavor of the application as this provides a superset of all other flavors, and was of highest priority to the customer. The result is an application and a platform for further development and research in the area. The application has been made with modifiability and extensibility in mind, and with a build system that allows easy configuration and compilation of different configurations and product flavors.

3.2.2 CLRF Beacons

The CLRF Beacons project is a proof of concept for the application of location-aware apps for visually impaired users. It is meant to provide its users with auditory, rather than graphical, cues about the environment and location. Its intention was to provide an understanding of the different beacon technologies and protocols, and which advantages and drawbacks each would contribute.

It was developed as a standalone application for this project, for reasons described above. It is, however, possible that a production implementation of this feature could be added to the "Colorophone Aid" version of the Colorophone application, to further improve the usability and utility of the system for visually impaired users.

3.3 Initial structure and changes

The customer proposed a range of different ideas at the start of the project period, as listed below in prioritized order. The customer wanted the group to explore what was possible for some of these ideas, with an actual Android application as a result. No strict requirements were given.

1. Color Sonification
2. Beacons
3. Image Viewer
4. External Camera Integration
5. Custom Android ROM
6. Shape Sonification
7. Color Constancy Research

The group initially split in two, one focusing on the color sonification and the other on exploring the use of beacons. As the work on the sonification proceeded, also the Image Viewer and External Camera Integration were started as features of CLRF Navigator.

The CLRF Beacons subproject succeeded as a proof of concept, but the customer decided to end this subproject and instead focus on gamification of the sonification application. The resulting structure thus had two separate subprojects, with CLRF Navigator being largest and including a range of complementary features.

4 CLRF Navigator

The main project is referred to by code name CLRF Navigator. The goal of this subproject is to develop and implement an algorithm for color sonification running on Android devices. The application should capture input from the camera, process it and produce audio corresponding to the input colors from the camera. The application should support further research by providing options to tune parameters related to the algorithm and application in general. Some of these configuration options should only be available for product development and prototyping.

The section starts with a description of the prestudy phase. A range of typical usage scenarios are then described, leading to the requirements specification. Then, an overview of the application is given. Following, the architecture and design are explained in detail. In the end, possible future extensions are discussed, which is of high importance as the application aims to be developed further.

4.1 Prestudy

The primary requirement for the CLRF Navigator is for the user to be able to perceive visual information from a live video through sound, using technology and algorithms developed by Colorophone. Researchers should also easily be able to change settings. As the technology is developed by Colorophone, the initial thought was to develop the application from scratch. However, a short analysis was done to check whether similar solutions exists.

There proved to exist a couple of solutions based on the conversion of visual information into sound. A sonification technology was developed by Meijer in 1992, which relies on scanning images from one side to another [31]. The technology has been applied to several platforms, including an Android application [32]. Another research group led by Amir Amedi has developed the Android application EyeMusic [1], using another sonification algorithm based on image scanning. Also, a range of other applications converting images into audio have been made, which are not for the purpose of aiding blind people [15, 16, 40, 46].

As noted, the apps of Meijer and Amedi work by fundamentally different principles as compared to Colorophone. Thus, as initially proposed, it was decided to develop the application entirely from scratch, only based on the Colorophone algorithm. Concerning the user interface, the menu system of both apps seems not to be very user-friendly and up-to-date. Additionally, none of them followed the Material Design guidelines that had been decided to follow. Thus, the group concluded also to develop debugging output and settings options from scratch. However, the existing solutions emphasized the importance of the sound being comfortable to listen to. In particular, the app by Meijer had very artificial sounds, which was unpleasant in the long run. Thus, the group decided to keep a focus on comfort when using the app.

4.2 Scenarios

In this section, a couple of potential ways of using the system have been described in terms of scenarios. This should give a clearer picture of who might be using the app, and how the app might be used. Note that the persons described here are fictive.

As the app intended for researchers has been most important, scenario 1 to 3 are most important for the requirements stated below. Scenario 4 to 7 mainly illustrate future users of the system. They should be kept in mind also while developing for researchers, and have thus been included.

Scenario 1: Experienced sonification algorithm researcher Nils is a developer of the sonification algorithm. Having worked with it for two years, he knows the algorithm and its background well, and he is keen to test it out on real persons. He has some basic computer skills, but he is not an experienced programmer.

Scenario 2: Psychology researcher Anne has been hired to the Colorophone team as an experienced psychology researcher, although with minimal sonification and programming experience. Her main focus is psychological aspects of the process of recognizing colors as sound. She should be perfectly able to use all features of the app, to perform her studies efficiently.

Scenario 3: Developer Per has recently finished his computer science studies and has been hired to Colorophone to develop the Android app further. In particular, he will be work with different algorithm implementations and sound effects. He has much experience with programming and using mobile devices.

Scenario 4: Visually impaired young student Kristin is a young, tech-oriented student who has been blind for all her life. She uses all blind aid systems available, in particular the well-established Talkback feature for Android. She has heard about the Colorophone concept and is eager to try it out on her Android phone.

Scenario 5: Visually impaired retiree Ole is a retiree who has been blind for most of his life, but his hearing is excellent. His wife got a smartphone last Christmas, but it has barely been used. Neither of them is experienced with any technology devices. He has been hired as a tester of the Colorophone app and is very interested in how it works and whether he will be able to use it.

Scenario 6: Friend of visually impaired people John is a student at NTNU, having downloaded a Colorophone app during a conference. He has been completely addicted

to the game and can tell the sound of just about any color intuitively. Now, he wants to recommend the concept to his two visually impaired friends.

Scenario 7: Visually impaired person from Nigeria Sarah is a thirty-year-old, blind woman from Nigeria. She lives in a poor region and does not have many opportunities due to her visual impairment. However, by being able to navigate using the Colorophone system, she would be able to obtain a job and improve her life quality significantly.

4.3 Requirements

The system has been developed in a highly agile way with requirements constantly evolving. The customer, as well as the project team, have come with new suggestions and priority changes as the project has proceeded. The group and the customer have, however, agreed upon a set of minimal requirements for a working system, related to scenarios 1, 2 and 3 as described above. Due to the nature of the project, these requirements have been subject to significant changes during the process, outlined in section 4.3.5. Also, it must be emphasized that the project has not been limited to these requirements, but extensions have been made in collaboration with the customer. The most important optional features are shown in section 4.3.3.

4.3.1 Functional requirements

Researchers must be able to:

ID	FR-NAV.01
Title	Change sound
Role	Researchers
Description	Change the sound of each channel individually. Required sounds: Sine waves, instruments and white noise. Optional sounds: Square waves, triangle waves and variations of white noise.

ID	FR-NAV.02
Title	Change frequency
Role	Researchers
Description	Change the audio signal frequency for each output channel individually for generated sounds

ID	FR-NAV.03
Title	Disable soundchannel
Role	Researchers
Description	Disable each channel individually

ID	FR-NAV.04
Title	Show RGB input values
Role	Researchers
Description	See algorithm input RGB values

ID	FR-NAV.05
Title	Show RGB output values
Role	Researchers
Description	See algorithm output RYGBW values

ID	FR-NAV.06
Title	Show frames per second (FPS)
Role	Researchers
Description	See number of frames processed per second

ID	FR-NAV.07
Title	Adjust volume
Role	Researchers
Description	Adjust the volume of each channel individually

ID	FR-NAV.08
Title	Change algorithm
Role	Researchers
Description	Change between algorithm implementations

ID	FR-NAV.09
Title	Adjust noise
Role	Researchers
Description	Change parameters related to noise generation

ID	FR-NAV.10
Title	Adjust feature extraction
Role	Researchers
Description	Change the size and shape of feature extraction

Developers must be able to:

ID	FR-NAV.18
Title	Add and remove algorithms
Role	Developers
Description	Easily add or remove implementation variations of the algorithm

ID	FR-NAV.19
Title	Add and remove sounds
Role	Developers
Description	Easily add or remove audio sounds

4.3.2 Quality attributes (non-functional requirements)

Performance

ID	NFR-NAV.01
Title	FPS
Role	System
Description	The system should be able to process 15 frames per second in good light conditions

Power efficiency

The system should minimize power usage.

ID	NFR-NAV.02
Title	Sonification with preview
Role	System
Description	Sonification should be operable for at least 2 hours while preview is enabled

Usability

ID	NFR-NAV.05
Title	API version
Role	System
Description	The application should be usable on all phones with Android API 17 or higher

ID	NFR-NAV.06
Title	External camera
Role	System
Description	The application should select any connected external camera as the default input, provided that the external camera is supported by the device.

Modifiability

The product will be used in a research and product development setting which requires rapid prototyping and a dynamic development model. The product architecture and

implementation should support this.

ID	NFR-NAV.07
Title	Add new sounds
Role	Developers
Description	Developers should be able to add new sounds permanently within 10 minutes of work

ID	NFR-NAV.08
Title	Add new algorithm
Role	Developers
Description	Developers should be able to add a new algorithm implementation within 30 minutes of work

ID	NFR-NAV.10
Title	Change parameters
Role	Researchers
Description	Researchers should be able to change any parameter related to the camera input, sonification algorithm or audio output within 30 seconds

4.3.3 Optional features

These are features that have been implemented to improve the application further, however, they have not been required.

ID	OPR-NAV.01
Title	Image Viewer
Role	Researchers
Description	Be able to choose an image from the phone and swipe to sonificate it.

ID	OPR-NAV.02
Title	Game
Role	End-users
Description	The app should have a game, challenging the user to guess a hidden image based on the sound from swiping.

ID	OPR-NAV.03
Title	Learning
Role	End-users
Description	The app should have a function teaching the user the mapping between color and sound.

ID	OPR-NAV.04
Title	Feature extractor scaling
Role	End-users
Description	The feature extractor should automatically scale to a reasonable size.

ID	OPR-NAV.05
Title	Extra settings
Role	Researchers
Description	Be able to change individual channel volume, change filter of white noise, change the value of Stevens Power Law and change the sample rate of the sound.

ID	OPR-NAV.06
Title	Extracted color
Role	Researchers
Description	Be able to see which color is extracted from the camera in the navigator.

4.3.4 Architecturally significant requirements (ASRs)

Architecturally significant requirements (ASRs) are those requirements that have a measurable effect on a software system's architecture. They are a subset of requirements, the subset that affects the architecture of a system in measurably identifiable ways. [11]

Performance There is a hard performance requirement of being able to process at least 15 and optimally 30 frames per second. This has a major impact on the architecture and software design. There is a long pipeline of components that are involved in the sonification process, from the input, through processing, to output. This pipeline has been optimized for performance, which to allow it to process 30 frames per second easily. The most resource-intensive task by far is the conversion to RGB, so most of the optimization efforts has gone into the conversion algorithm. This requirement also limits the amount of memory allocations that can be performed during the sonification process due to how the garbage collector (GC) works, and the architecture, therefore, minimizes memory allocations by utilizing predefined memory buffers and object pools.

Modifiability Because the application is intended as a platform for further research and prototyping, modifiability is also essential and has a significant impact on the architectural decisions. However, high modifiability may also conflict with performance in some cases, especially in regards to the sonification pipeline. Common modifiability tactics such as virtual dispatch through polymorphism, observer patterns and other forms of indirection may have negative effects on performance due to increased complexity or an increase in method calls. In such cases, performance has a higher priority.

4.3.5 Changes to requirements

Due to the nature of this project, changes to the requirements have been extensive. This section describes requirements that have been changed or removed.

ID	FR-NAV.11
Title	Add and remove feature extractor
Role	Researchers
Description	Add or remove feature extraction objects

The customer has emphasized that additional feature extraction objects are not important, thus this requirement has been removed.

ID	FR-NAV.12
Title	Connect feature extractor
Role	Researchers
Description	Connect feature extraction objects to algorithm implementations and audio channels

The customer has emphasized that additional feature extraction objects are not important, thus this requirement has been removed.

ID	FR-NAV.13
Title	Add sounds
Role	Researchers
Description	Add pre-recorded sound samples for channels

This has not been done due to priority restrictions.

ID	NFR-NAV.01
Title	FPS
Role	System
Description	The system should be able to process 30 frames per second in good light conditions

This was changed from 30 FPS to 15 FPS, which is the absolute minimum requirement.

ID	FR-NAV.20
Title	Preset configuration
Role	Developers
Description	Easily create preset configurations of sounds and frequencies

This has not been done due to priority restrictions.

ID	NFR-NAV.03
Title	Sonification without preview
Role	System
Description	Sonification should be operable for 5 hours with preview disabled

Preview disabling has not been done due to priority restrictions.

ID	NFR-NAV.05
Title	API version
Role	System
Description	The application should be usable on all phones with Android API 16 or higher

The API level has been changed from 16 to 17 due to use of the Renderscript framework, described in section 4.6.3.

ID	OPR-NAV.03
Title	Tutorial
Role	Researchers
Description	The app should have a tutorial-function that teaches the user how to use the app.

This has not been done due to priority restrictions.

4.3.6 Requirements for Colorophone Aid

These requirements are for the Colorophone Aid app, which is intended for visually impaired users. This app has not been implemented. The requirements are included here for reference in case the app will be implemented in the future, see section 4.8.6.

Functional requirements

End-users must be able to:

ID	FR-NAV.14
Title	Hear colors
Role	End-users
Description	Hear colors as instruments

ID	FR-NAV.15
Title	Start sonification
Role	End-users
Description	Start sonification directly by opening the application

ID	FR-NAV.16
Title	Turn off sonification
Role	End-users
Description	Turn off sonification by using a gesture or pressing back

ID	FR-NAV.17
Title	Adjust sonification sound
Role	End-users
Description	Adjust sonification volume using external volume button

Quality Attributes (Non-functional requirements)

ID	NFR-NAV.04
Title	Sonification start
Role	System
Description	Sonification should start within 2 seconds after opening the Sonification application

4.4 Application overview

The application is centered around the conversion of colors into sound. The app uses a live camera or image as an input source. Pixels are extracted from a part of the image known as a "feature extractor". The red, green and blue channels are used as input to an algorithm, which processes the data and encodes audio channels with different sounds and volumes. This sonification processing is at the very core of the CLRF Navigator project and is the basis for a range of features described below.

Camera The Camera feature uses the phone's internal camera as an input source for the sonification process. This feature provides an easy way of testing how sonification is experienced by end-user. Also, a sidebar showing input and output values is provided.

Image Viewer The Image Viewer feature lets the user select an image, and then provides color sonification for the image. The sonification works by playing sound from the area on the image where the user touches.

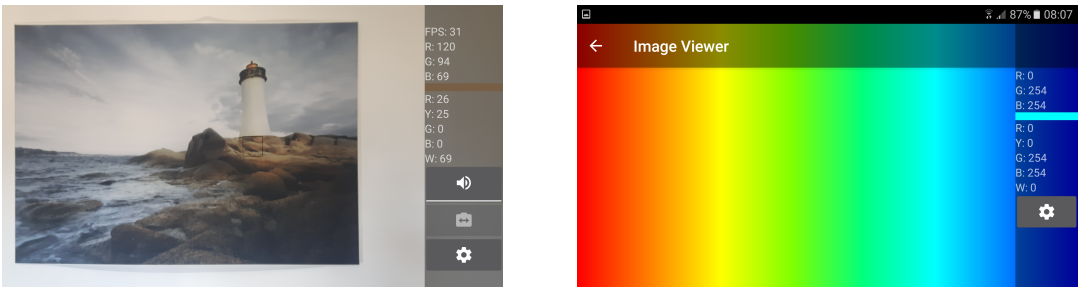
Game The Game is an extension of the Image Viewer. It lets the user use the touch screen to provide input to the sonification process without revealing the image visually. It provides a means for the user to guess the object or color in the image, along with feedback as to whether the answer given was correct or wrong.

Color Map

The Color Map provides a way of learning which color corresponds to which sound. The feature lets users listen to different colors by pressing buttons corresponding to colors.

External Camera Integration This feature allows for the integration of an external camera with the Android application and sonification process. It involves low-level integration using UVC drivers for Android, obtaining the preview frames and converting them to an appropriate format. This allows for connecting the app to hardware made by other groups of the Colorophone project.

Screenshots of the Camera and Image Viewer are shown in figure 6a and 6b, respectively. A detailed description of how to use the app can be found in the user guide, attached in Appendix G.



(a) Screenshot of the live camera. Sound is extracted from the black square in the center.

(b) Screenshot of the Image Viewer. Sound is extracted from the area where the user touches.

Figure 6: Screenshots of the CRLF Navigator app

4.5 Architecture

The application is built on top of the Android architecture stack, and is limited by the restrictions imposed by the Android SDK. The application has been built with a layered architecture. All interaction with audio and the camera has been placed in the sonification layer, which internally uses a pipe-and-filter pattern. UI logic and views have been placed in a separate layer, the UI layer, and utilizes the MVP pattern. All communication between the UI layer and the Audio and Camera APIs goes through the sonification layer.

Figure 7 shows how the layers communicate with the other layers of the Android architecture which the application is built upon. Every layer uses the layer immediately below it and may use other layers further down the hierarchy. The sonification layer accesses the Hardware Abstraction Layer (HAL) to implement support for external UVC cameras. It also utilizes the JNI (Java Native Interface) for efficient processing of the feature extraction through the Native C/C++ layer. It uses the common Java API framework

for managing the audio and internal camera, as well as other APIs for managing threads and so on. The GUI layer uses the sonification layer immediately below it, and the Java API Framework for most of its logic. It also uses the System Apps layer to, for example, select an image to open.

The following sections provide a view of the architecture based on the 4+1 View Model as presented by Sommerville [47, p. 153].

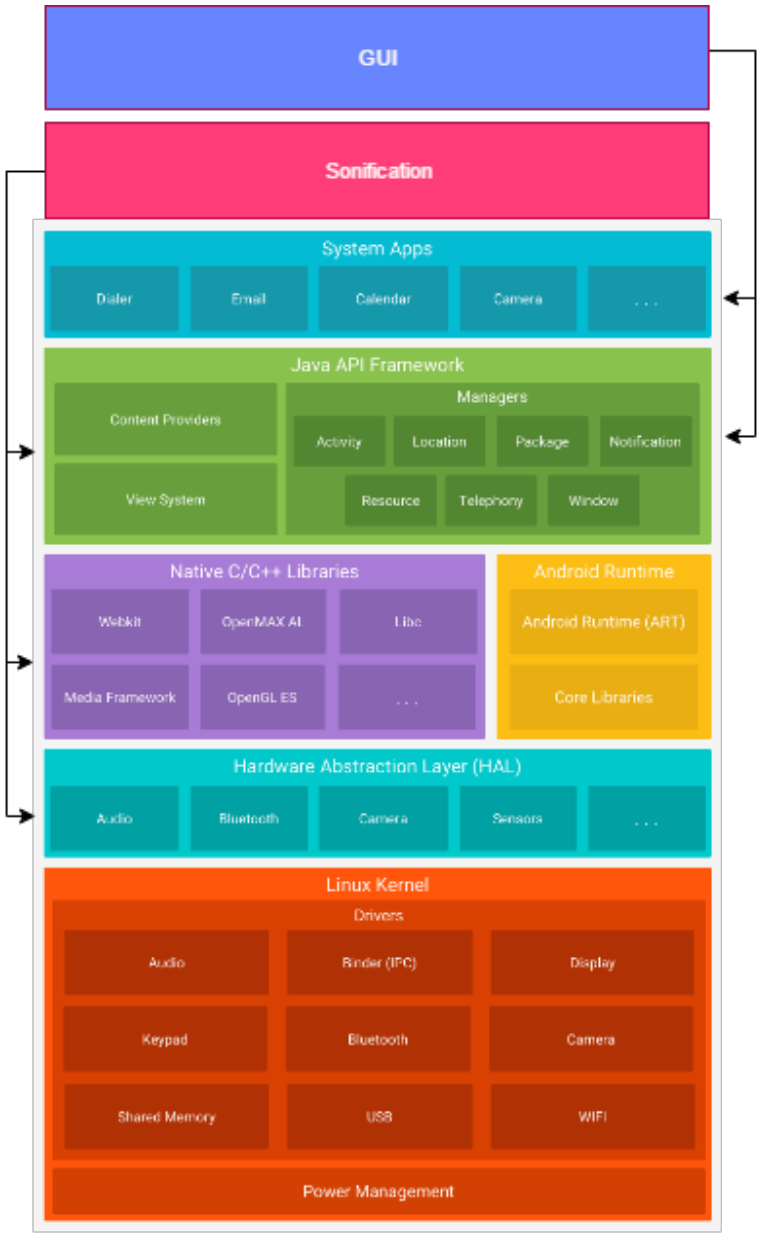


Figure 7: Application architecture. GUI and Sonification layer is created on top of the Android framework stack.

4.5.1 GUI layer

The application uses a basic Model-View-Presenter (MVP) architecture for the GUI. The Views are represented by Android Activity classes. A View holds a reference to a Presenter, which it can call into and notify of user actions. A View must implement a Presenter specific interface. The Presenter holds a reference to an implementation of its View interface and can perform UI operations on this View. Models are observable through a Model specific observer pattern, which a Presenter can observe by implementing a Model listener interface. The Presenter is responsible for mediating data between the View and the Model.

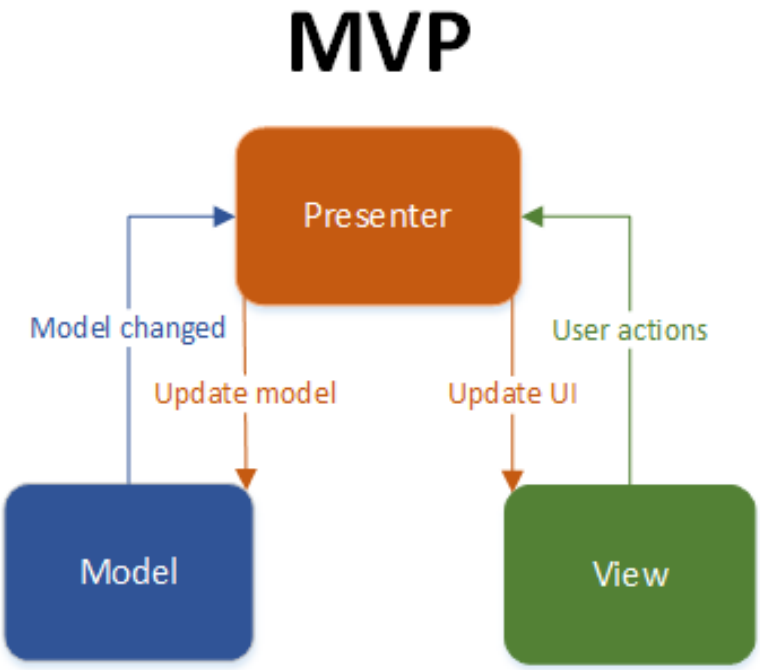


Figure 8: Generic Model-View-Presenter architecture

4.5.2 Sonification layer

The sonification layer uses a pipe-and-filter architecture pattern, as described by Sommerville [47, p. 163]. Figure 9 illustrates how data flows through the pipeline, and in which order the components are connected. Each component takes the output from the previous component in the pipeline as its input. The passing of data between components is not direct but is handled by the `SonificationManager`. Direct piping of data between components was considered as an option, but a decision was made to let a separate object handle the flow of data to increase modifiability and provide better control of the data flow.

Figure 9: Sonification layer

4.5.3 Development view

The application consists of several packages, which contain coherent modules and classes that combine to a component of the application. Figure 10 shows a simplified view the various packages, and how they communicate among each other. The name of the package is displayed in the header, and the fields in each package display the most prominent classes and interfaces in the package. Some dependencies have been left out to keep the figure readable, such as the dependencies of the top-level package. The top-level package, app, depends on all the other packages. As can be seen from the figure, all dependencies are one-way dependencies such that no two components mutually depend on each other, which in turn helps achieve modifiability.

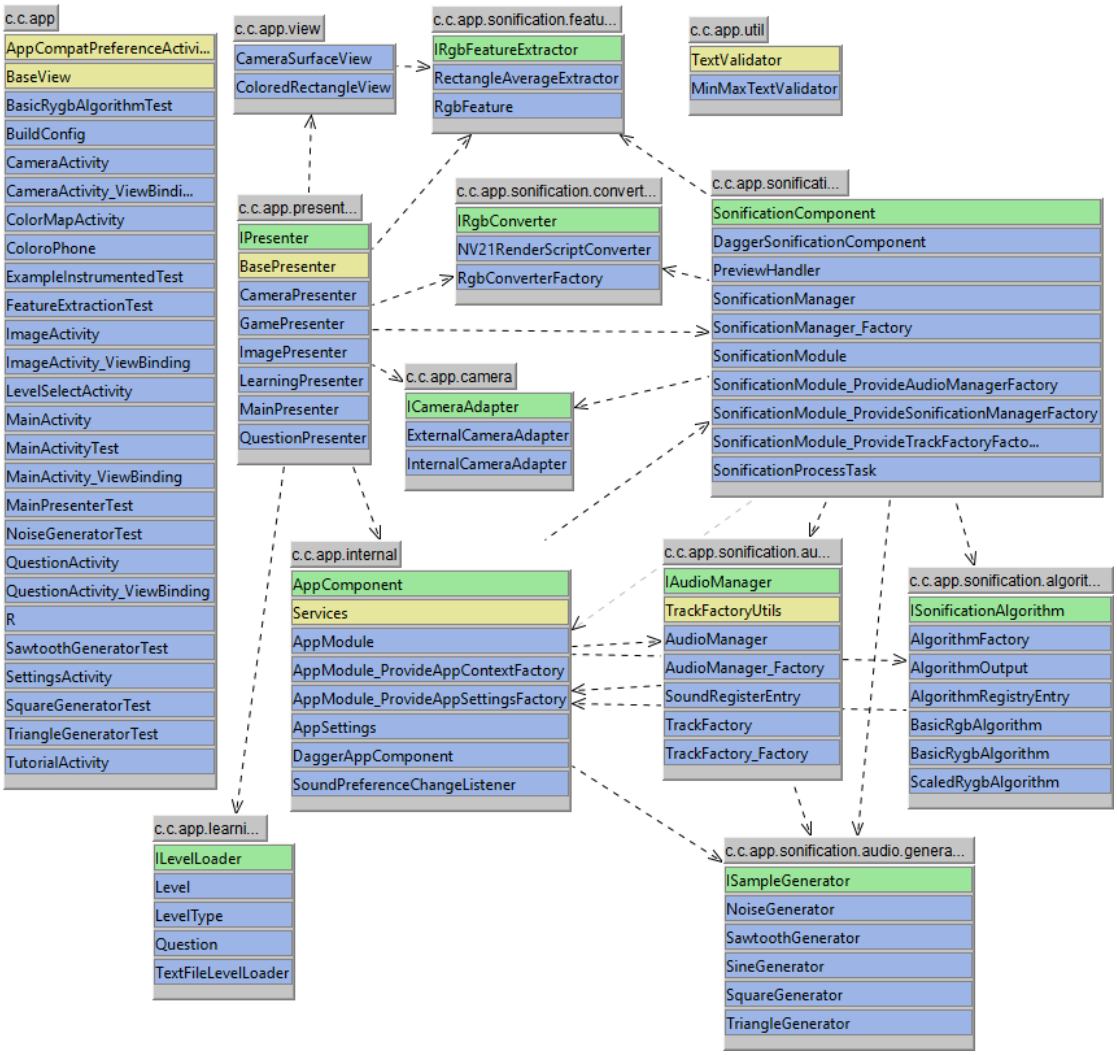


Figure 10: Application packages. The name of the package is displayed in the header, and the fields in each package display the most prominent classes and interfaces in the package.

app The main package. This package contains the Application class, as well as all Activities and other Android specific classes.

app.presenter Contains a `IPresenter` and `BasePresenter` as well as their implementations. The interface and base class provides Android lifecycle management.

app.view Contains custom made view components, sometimes called widgets. `CameraSurfaceView` is used to display the camera preview frames on the device display. `ColoredRectangleView` displays a configurable solid color rectangle, which is used to display the RGB value extracted by the feature extractor.

app.learning Contains models and logic related to the Color Map and Game activities.

app.sonification Contains all classes related to the sonification layer. Most notably the `SonificationManager`, `PreviewHandler` and `SonificationProcessTask`.

app.sonification.feature Contains the feature extractor interface, its implementations and a poolable `RgbFeature` class which is used for output from the extractors.

app.sonification.converter Contains a converter interface and converters that convert camera input to ARGB8888. Currently, only the YV21 format is supported, but that should be enough as all devices should support it by default.

app.sonification.algorithm Contains an `ISonificationAlgorithm` interface and various algorithm implementations. `AlgorithmOutput` is a poolable class used for algorithm output.

app.sonification.audio Contains the `IAudioManager` interface and an implementation. The `AudioManager` simplifies managing the audio tracks used for sonification. `TrackFactory` is used to instantiate tracks with sounds based on user settings. `SoundRegisterEntry` is used to register sounds and sound factories with the application.

app.sonification.audio.generator Contains audio sample generators used by the audio package.

app.camera Contains the `ICameraAdapter` interface, as well as two implementations `ExternalCameraAdapter` and `InternalCameraAdapter`, which handles logic specific to the camera that is being used.

app.internal Internal helper classes specific to the application, such as `AppSettings` which simplify use of shared preferences, and the `Services` class which acts as a ServiceLocator for the application.

app.util Generic utility classes.

4.5.4 Process view: Activities

The application consists of several activities, which are implementations of an Android `Activity`. As the application has to be available in two different versions for blind users and researchers, a build flag can be used to deactivate or activate certain activities and alter their behavior when building the application. This makes it easy to build several different versions of the application using the same code base. The version intended for blind users is a simplified version of the full-fledged application, and only provides access to MainActivity and CameraActivity.

The diagram in figure 11 shows the basic navigation structure of the application. Pressing the back button on the device at any point will always lead back to the previous activity, and may in some cases close the application if the previous activity is in another process. The SettingsActivity has an explicit back button that will always take the user back to the previous activity in the same process. The user can also exit the application at any time using the Android navigation, but this is implicit and handled by the Android platform, and therefore not shown in the diagram. The application provides no explicit way to quit the application, as per the Android navigation guidelines[20].

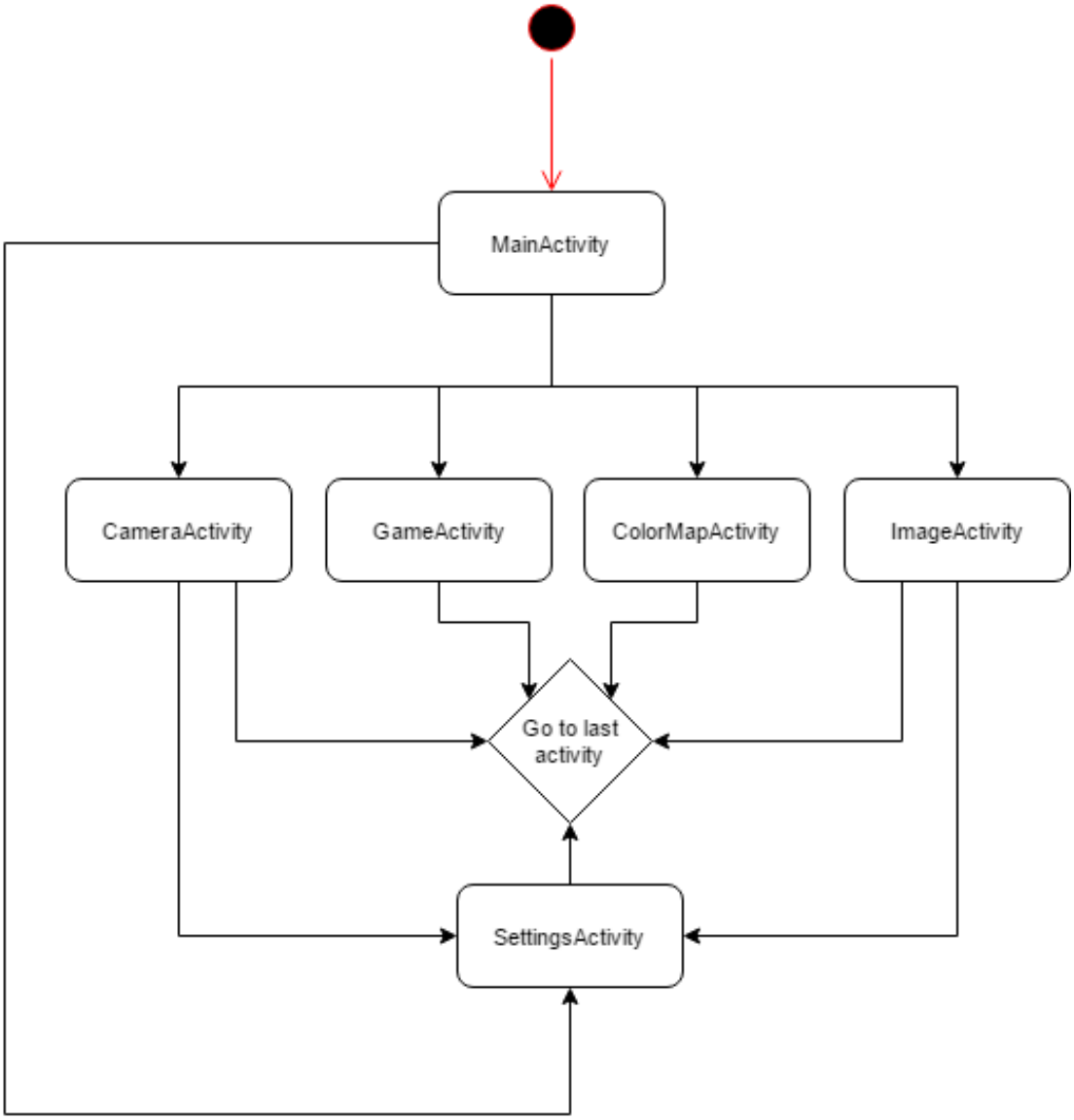


Figure 11: Activity diagram

MainActivity The main activity is the entry point for the application and provides the user with a navigation drawer which lets a user navigate to any other activity in the application, including settings. The main activity currently displays a web view of the Colorophone web page, but this is intended as a temporary solution.

CameraActivity The camera activity lets a user use sonification through the internal, and possibly external, camera. The activity initializes the camera using the Camera API and sets up the sonification process based on user settings and default configurations. The activity provides quick access to the user settings activity, and has an info panel that displays information regarding the sonification process, such as input, algorithm

and output values.

ImageActivity The image activity lets a user select any image available on the device, and loads it for sonification. The sonification input is guided by screen touch events, which manipulates the position of the feature extractor. Due to the loose coupling between the input and the sonification process, the application can use the `SonificationManager` without any modifications except the manipulation of the feature extractor position. The selected image is simply converted to a bitmap and provided to the `SonificationManager`.

GameActivity The game activity lets a user play a game which provides a mixture of learning and entertainment for the user. The user can select between different difficulties and is tasked with identifying the correct color or object through color sonification. Sonification is performed by swiping the finger across the screen, and when finished the user is provided with four alternatives, of which one is correct. After each answer, the correct answer is given, and a total score is displayed at the end.

ColorMapActivity The Color Map provides a learning environment to the user. It lets the user listen to a selection of sample colors by pressing a color in a Color Map, providing a simple training and learning opportunity to the user.

SettingsActivity The settings activity utilizes the Android Shared Preferences API to store and manage user preferences. This activity follows all the best practices regarding Android preferences. The Preferences API is wrapped in a custom wrapper API named `AppSettings` to simplify the use of the preferences in the application code, and to manage registering sounds, algorithms and other objects at run-time.

4.5.5 Process View: Sonification

The sequence diagram is shown in figure 12 depicts the object and classes involved in the sonification process, and the sequence of messages exchanged between them. The sonification of the transition area has been removed for simplicity as it is almost identical to the peripheral sonification. The diagram provides a detailed view of the flow of messages between the components in the sonification pipeline. It shows how the image is converted before being handed over to the `SonificationManager`, which then takes care of the rest of the process. It also shows how the returned data objects are being obtained and released throughout the process to avoid object allocation.

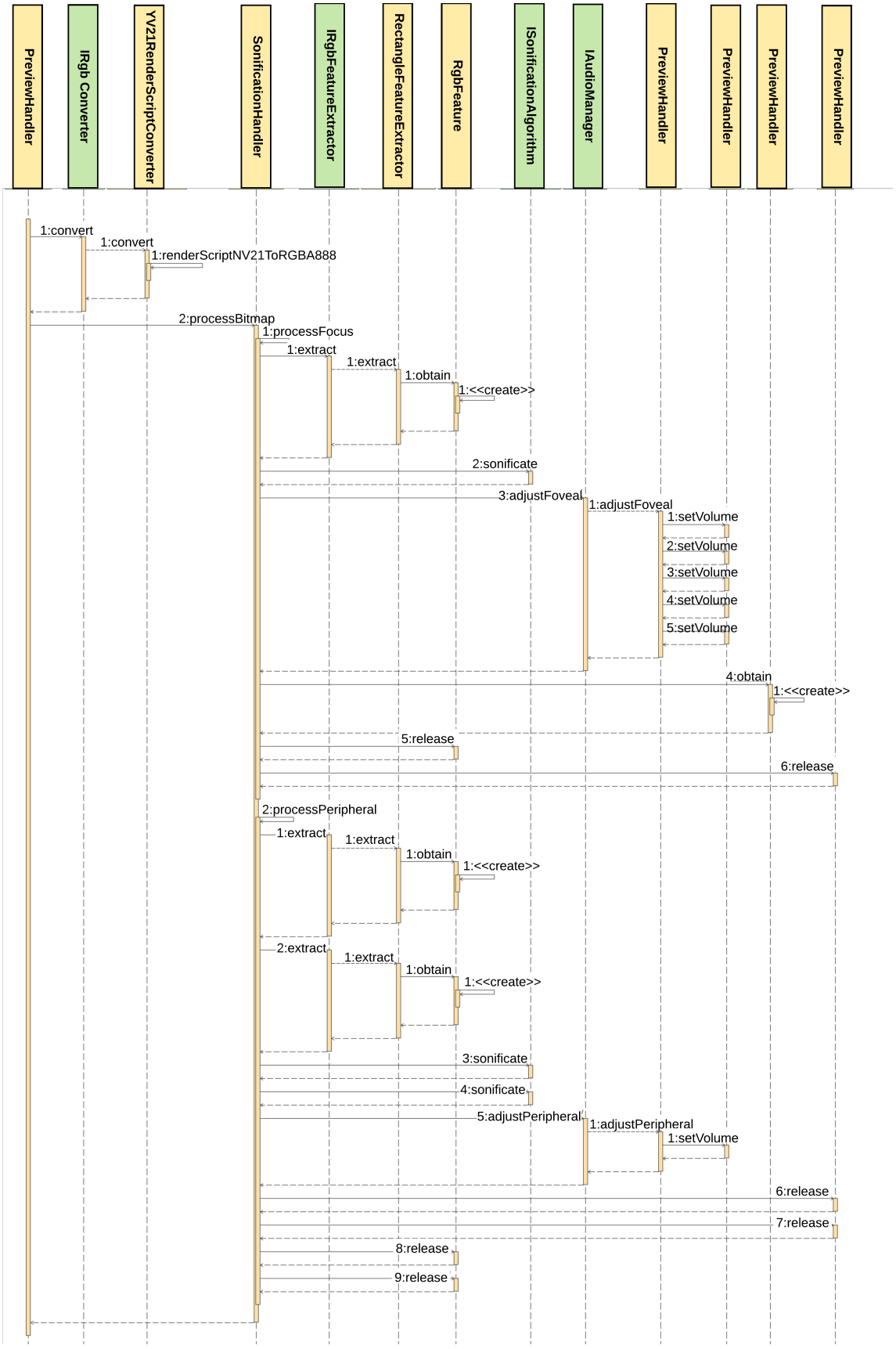


Figure 12: Sonification sequence diagram

4.5.6 Physical view

The application can be deployed to Android devices through Google Play Store, email, USB or any other media that can carry data. It supports all devices and configurations that satisfy the minimum requirement of Android version 4.2, including smartphones, tablets, and PCs. That is, the execution environment must support API 17 or higher. Figure 13 shows how the application is compiled and deployed to a device.

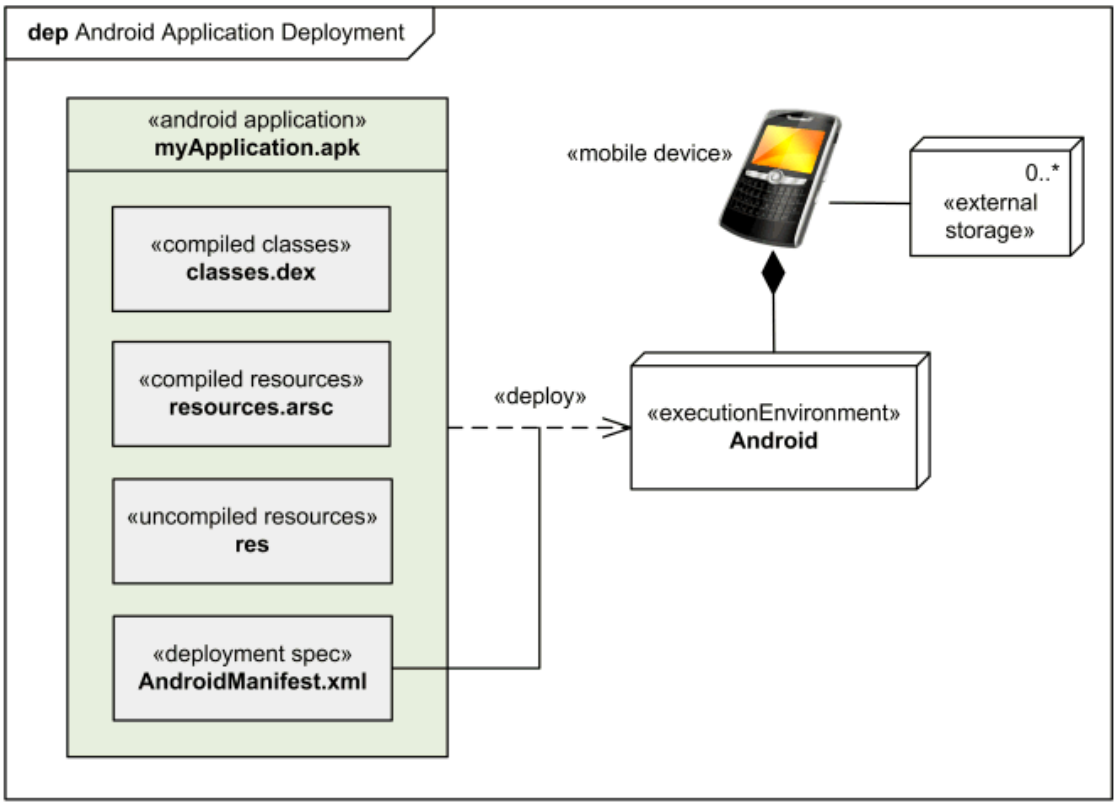


Figure 13: Android deployment[53]

4.5.7 Logical view

The application consists of several different components as described in previous sections and figure 9. This section provides a detailed description of the most interesting components of the system, mostly in the sonification layer. Figure 14 shows the overall structure of the components in the sonification pipeline, as well as the `SonificationManager` which acts as a facade to the other components and simplifies their use. `SonificationManager` mostly has references to interfaces so that any component in the pipeline can be replaced with another implementation at run-time. This greatly simplifies the work of making changes to the pipeline, and therefore greatly improves the modifiability of the pipeline. This also allows the pipeline to easily be adapted to other inputs, such as static images. Any image, regardless of source, can

be converted to the ARGB8888 format and processed by the `SonificationManager` . When operating in camera mode, the `PreviewHandler` receives a callback with the preview data on each preview frame capture. In camera mode, this is the object that references and uses the `SonificationManager`, while in image mode this would be a different class. If the application is using the internal camera, the `PreviewHandler` will first convert the frame using a reference to an `IRgbConverter`. The frame is then passed to the `SonificationManager` for further processing. From there, it is passed to one or more instances of an `IRgbFeatureExtractor`, depending on if it is operating in foveal or peripheral mode, which each returns an instance of a `RgbFeature` data object. Next, the `SonificationManager` extracts this data and passes it to an instance of `ISonificationAlgorithm`, which returns an instance of `AlgorithmOutput`. Again, the `SonificationManager` extracts the data from the returned object and passes it to the last step in the pipeline, an instance of `IAudioManager`. When the `AudioManager` has processed the data, the chain is complete and control returns to the caller. An instance of `SonificationProcessTask` may be used to notify possible observers of the results of the individual steps in the chain, such as the GUI.

Lots of thought and effort has gone into creating a robust platform for further development, and the pipeline has gone through several iterations of refactoring, improvements, and testing.

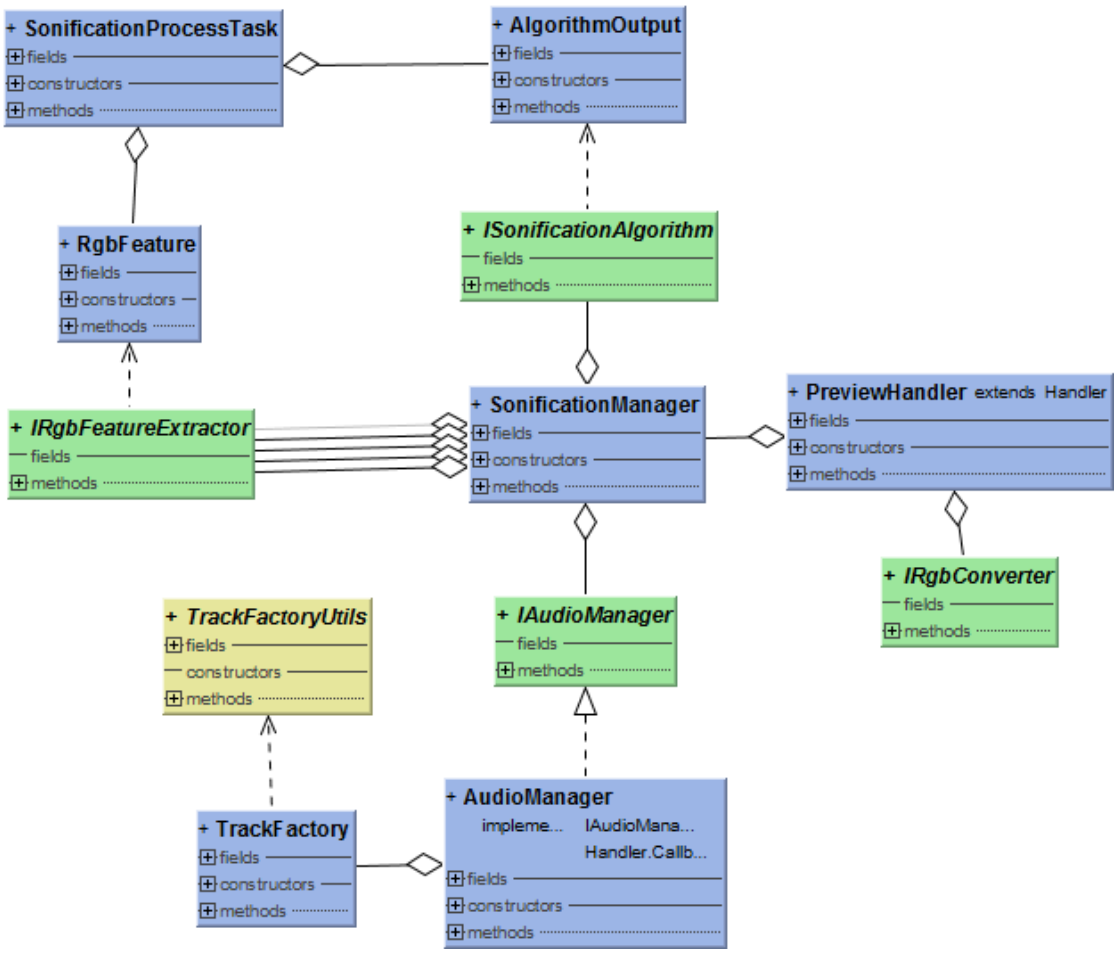


Figure 14: Camera sonification overview

Camera The application supports using either the internal camera or a supported external USB UVC camera. External USB cameras are not supported by all devices and depends on the manufacturer's modifications to the Android OS.

Android does not provide native support for external cameras, so all interaction with the underlying camera implementations has therefore been wrapped in an interface, `ICameraAdapter`. This interface has two implementations, `InternalCameraAdapter` and `ExternalCameraAdapter`. This relieves the client code of having to differentiate between which camera is being used and provides a uniform interface for all cameras. Figure 15 displays how the classes in the camera component are connected.

The `PreviewHandler` contains a reference to an instance of `ICameraAdapter`, and is responsible for handling preview frame callbacks from the camera. The `ICameraAdapter` provides methods to manage the camera, as well as a wrapper, `CameraListener` for the native camera preview frame callback, which is different for the internal and external camera. This makes it possible to define a single callback handler for preview frames, regardless of its source. The callback wrapper converts the callback data to a byte

array before passing it to its handler. The `ICameraAdapter` extends `SurfaceHolder.Callback` so that its implementations can manage how the preview is drawn on the screen.

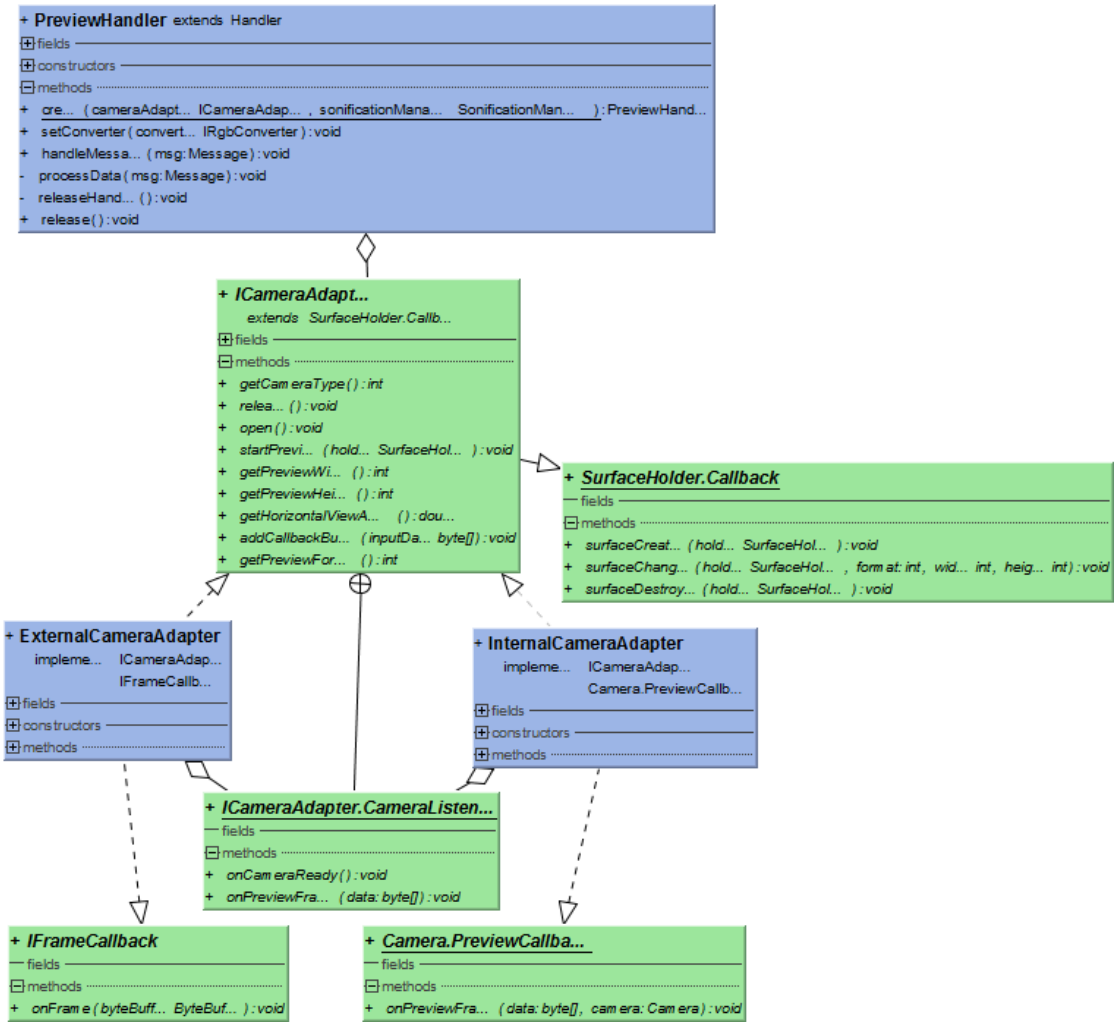


Figure 15: Camera component

RGB conversion Figure 16 shows the `IRgbConverter` interface and one implementation of it. The `RgbConverter` takes as parameters an input byte array, the image width, image height and a byte output array. `YV21RenderScriptConverter` converts a byte buffer from YV21 to RGBA8888 format. The conversion step is optional and is only performed if needed. It is not used when processing stored images or when operating with an external camera.

After the conversion has completed, the `PreviewHandler` passes the newly converted buffer to the `SonificationManager` for further processing.

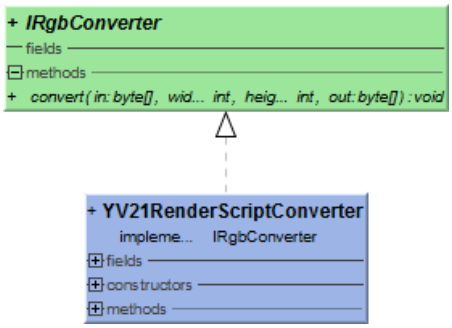


Figure 16: Bitmap converter component

Feature extraction Figure 17 shows the `IRgbFeatureExtractor` interface and an implementation, as well as a data class used to return the extractor output. A feature extractor can be used to extract a single RGB feature from a bitmap, or in other words, reduce a bitmap to a single RGB value. The size and position of the rectangle can be set through mutators, as well as the size of the frame it operates on. This makes it easy to resize or reposition the extractor dynamically, either through settings or other means such as the touch position on a screen as done with the Image Viewer.

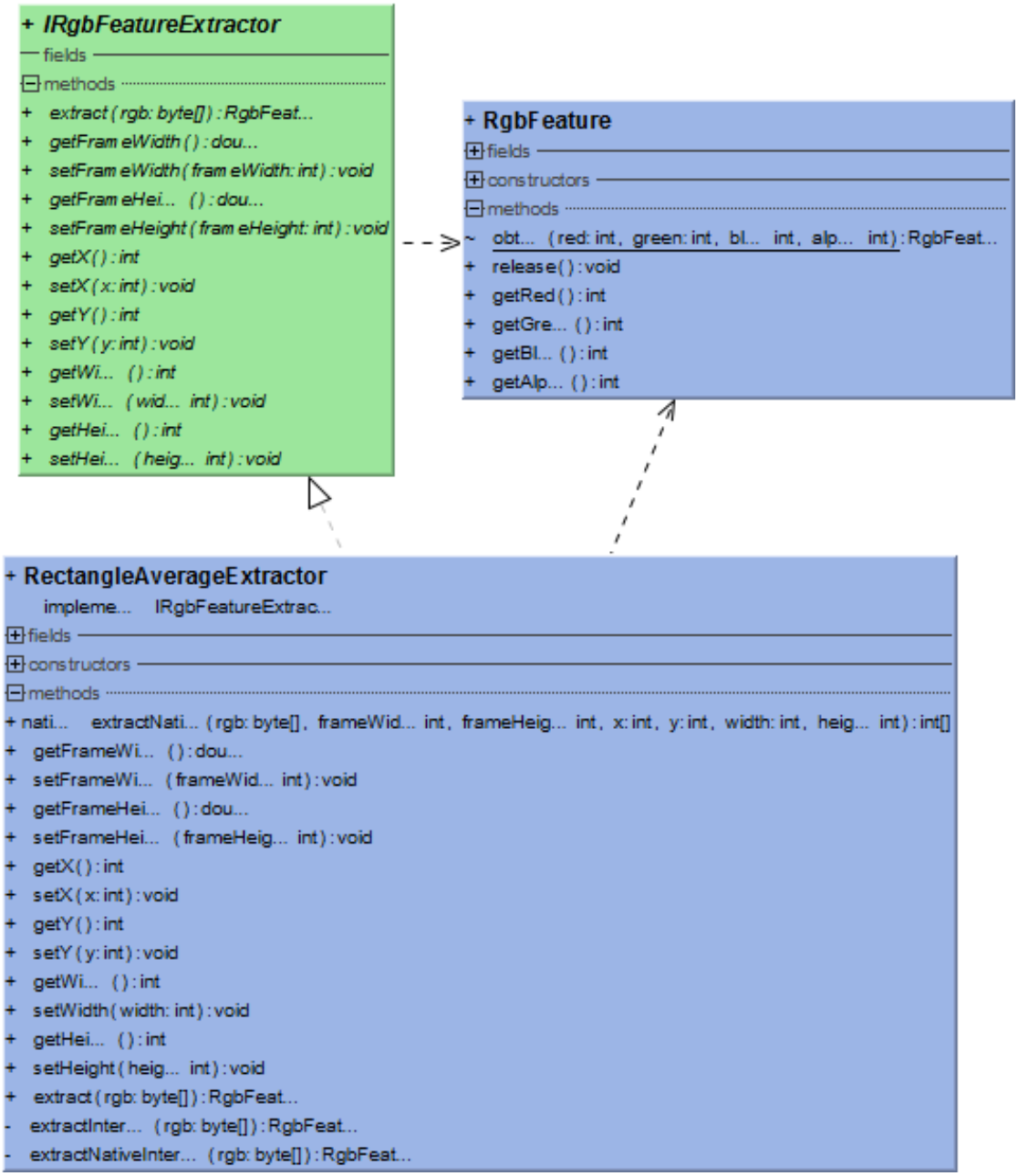


Figure 17: Feature extractor component

Sonification algorithm Figure 18 shows the `ISonificationAlgorithm` interface and some implementations, as well as the algorithm output class. The interface is very simple, and defines a single method that takes as input the integers representing red, green and blue, and returns an instance of `AlgorithmOutput`. The output object provides accessors for red, green, blue, yellow and white channel information. Like the `RgbFeature` objects, these objects are obtained from an object pool and must be released when it is no longer needed. The sonification algorithm is explained in further detail in the section 4.6.6. The specific algorithm implementation is instantiated by the `AlgorithmFactory` based

on user settings, and available algorithms are provided to the system through instances of `AlgorithmRegistryEntry`. This design and its implementation supports all related requirements as specified by FR-NAV.08, FR-NAV.18 and NFR-NAV.08.

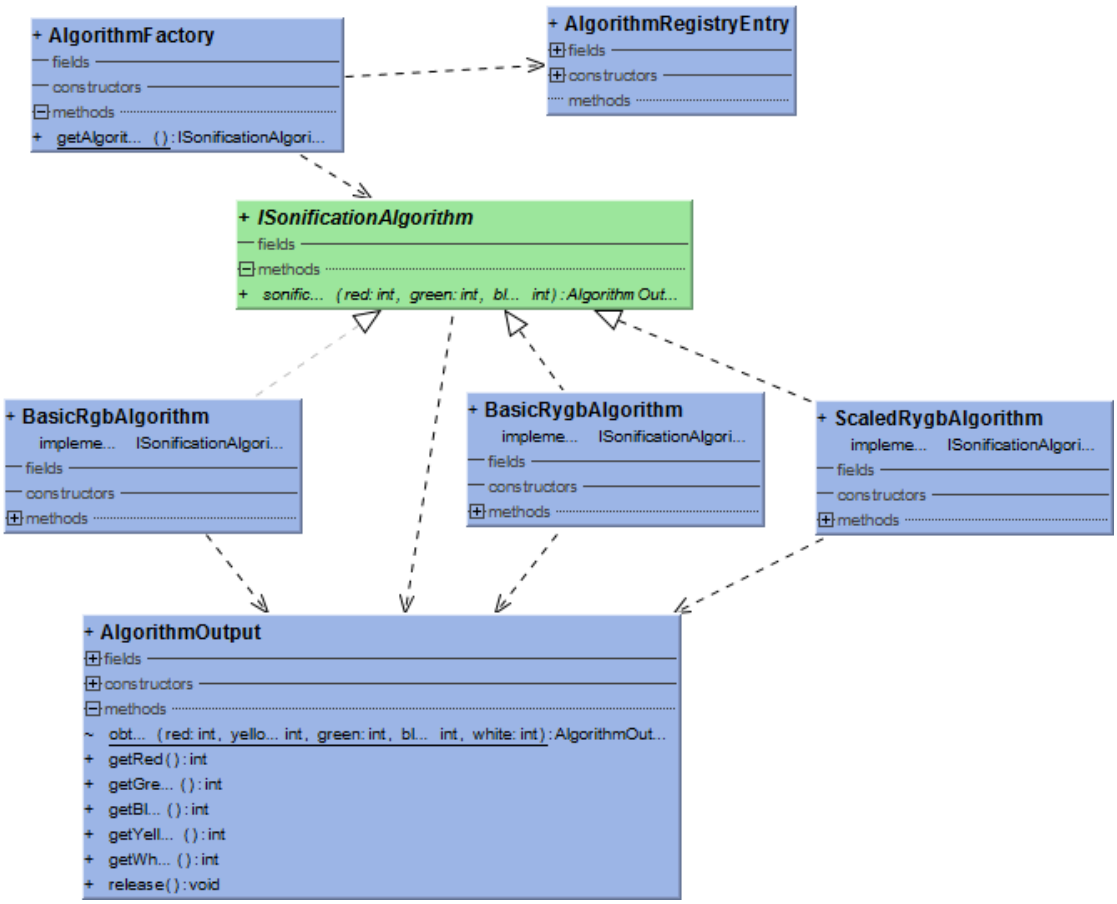


Figure 18: Sonification algorithm component

Audio The audio module is by far the most complex component in the sonification pipeline, as can be seen in figure 19. `ISampleGenerator` defines an interface for sound generators, and contains a single definition for a method `nextSample()` which returns an audio sample in the range [-1 : 1]. The sample generators can be constructed with parameters for sample rate and frequency. The `NoiseGenerator` also implements this interface but has different constructor parameters.

`TrackFactory` is a factory class for audio tracks and creates tracks and audio content based on user settings. It is possible to dynamically add sounds at run-time using `SoundRegisterEntry` objects, which specify a sound name, key and sound factory for the sound. The `TrackFactory` can match the keys against the keys specified in user settings, and then instantiate the correct sound factory based on the registry. The factory supports creating generated sounds using the `ISampleGenerator` interface, or it can load prerecorded PCM samples, depending on user settings. `TrackFactoryUtils` provides utility methods for creating and writing to audio tracks.

The `IAudioManager` interface is the interface referenced and used by the `SonificationManager` to start, pause, stop and adjust the volume during sonification.

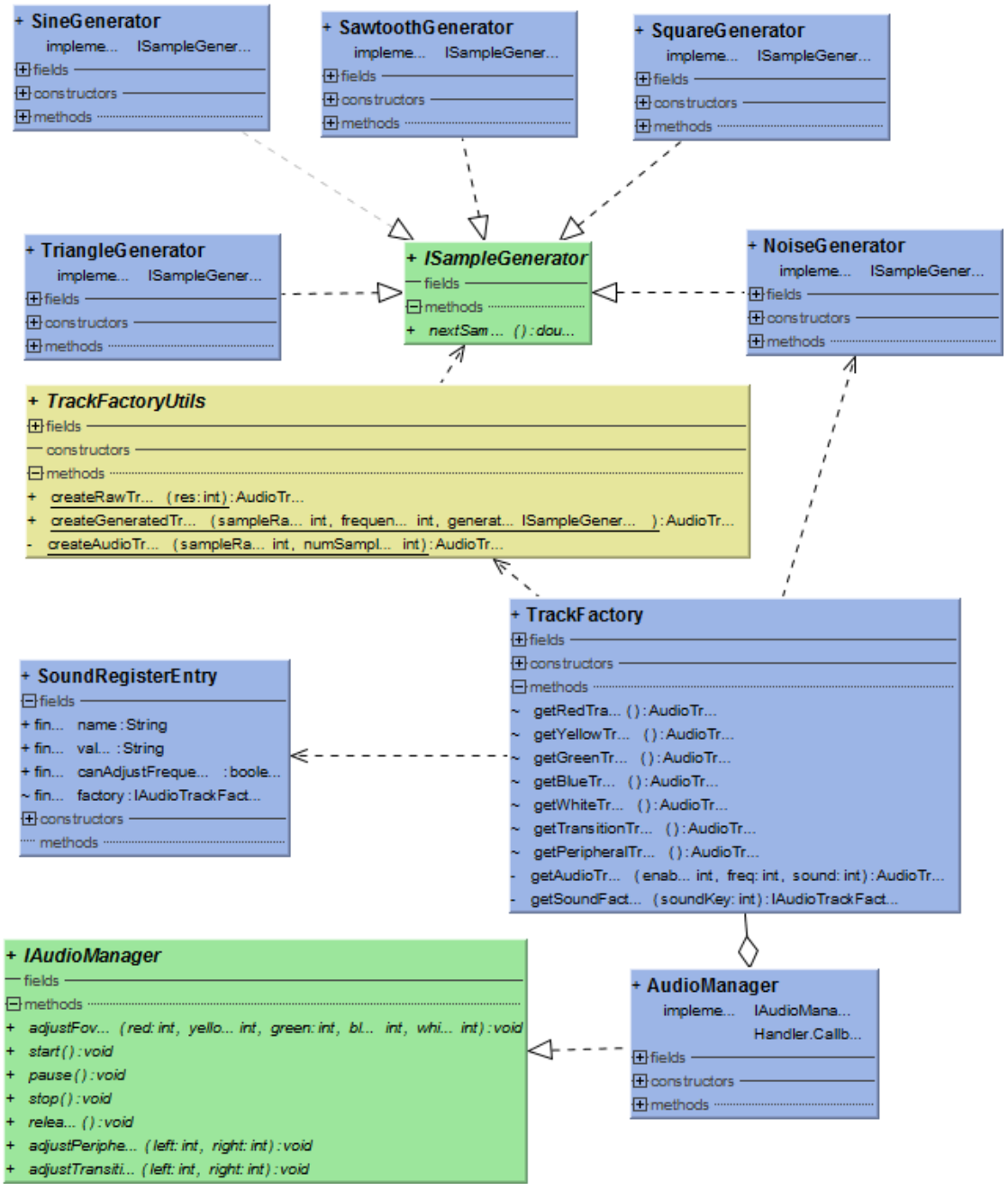


Figure 19: Audio component

4.6 System Design

The application design follows Android best practices as closely as possible, as they are described in the official Android best practice documents[3, 5, 7], and by Future developers[18]. The application also closely follows the Material Design[21] guidelines. Also, it utilizes several state of the art Android libraries such as Dagger2 for dependency injection and Glide for managing media.

4.6.1 Object pools

Considering the small amount of memory available on most devices and the way the Android Garbage Collection (GC) functions, a decision was made to implement custom object pools for all frequently used objects in the sonification pipeline. This was done to reduce the overhead of creating new objects and more importantly reduce the workload of the GC[8]. This has a measurable positive impact on NFR-NAV.01 and NFR-NAV.02.

4.6.2 Camera implementation

The `ICameraAdapter` API provides a hook which allows the application to receive a callback with the preview frame data on each update. The data is supplied through a user-specified byte buffer, and the default image format is YV21 for the internal camera and RGB888 for the external camera. A decision was made to use the default format as it is guaranteed to be supported by all devices, unlike the other formats. The frame is then displayed on the device screen through a custom implementation of a `SurfaceView`, which adds a visible rectangle indicating the position and size of the feature extractor as well as the number of frames processed per second, as per FR-NAV.06.

The byte buffer is then passed on to an Android `Handler` object which has a `Looper` on a separate high priority thread, by placing a `SonificationTask` with a reference to the buffer in the handler's message queue. The queue functions as a LIFO queue with a size of 1. This ensures that the application only processes the most recent frame in cases where processing is slow and avoids build-up of a queue of old frames. In such cases, it is preferable to lower the number of samples processed per second, and ensuring that the frames that do get processed are as up to date as possible.

If the system is operating with the internal camera, the handler then uses an instance of `IRgbConverter` to copy and convert the buffer to a new preallocated conversion buffer, before making the preview buffer available again for the camera. The application uses manually preallocated preview- and conversion-buffers and double buffering to allow the camera to begin capturing the next frame while the application continues processing the current frame. The external camera manages its own buffers, and the preview frame does not need to be converted. The frame is eventually passed on to the `SonificationManager` for further processing.

External camera The application uses a 3rd party library for external camera support, UVCCamera, which offers an API that is different but similar to the Android Camera API.

Internal camera The application uses the Android Camera API to set up the internal camera and process preview frames. A decision was made to use the original Camera API rather than the newer Camera2 API. to support devices running API 16 instead of requiring API 23. This also simplifies the `ICameraAdapter` implementations as the original Camera API is similar to the external `UVCCamera` API.

4.6.3 Bitmap conversion

The bitmap conversion is performed using an API called RenderScript. This API provides access to highly optimized algorithms that are often written in low-level languages and can possibly be run on the GPU. The initial implementation of the conversion was implemented manually in Java and proved to be a major bottleneck in the system and a limiting factor for the FPS in good light conditions. The current YV21 to RGB conversion algorithm is partly written in Neon assembly, C99 kernel language and C++ for the NDK, and provides performance that are several orders of magnitude better than a Java implementation could possibly be. RenderScript was introduced in API23 but can be utilized on older devices through the RenderScript Support Library.

This component has been extensively tested to ensure it meets the performance requirements as described by NRF-NAV.01. The conversion algorithm also has a major impact on NFR-NAV.02 and NFR-NAV.03 as it can offload work to the more power efficient GPU and also minimizes the time spent in high-energy modes.

4.6.4 Feature extraction

The `RectangleFeatureExtractor` implementation performs feature extraction by calculating the average of the RGB values of pixels defined by a rectangle anywhere on the bitmap. The initial implementation of the feature extractor was fully implemented in Java. The CPU-intensive part of the feature extractor has later been refactored and implemented in native C, and is being called through the JNI. This results in a significant improvement in processing time and CPU utilization, which also benefits battery performance.

The extractor returns the result as a `RgbFeature` object, which is stored in an object pool for performance reasons. The objects are obtained from the pool when needed, and client code releases the object back to the pool when it is no longer needed. The implementation supports the requirements related to feature extraction as defined by FR-NAV.10, FR-NAV.11 and FR-NAV.12.

4.6.5 Audio

`AudioManager` is an implementation of the `IAudioManager` interface that uses one Android `AudioTrack` object for each channel. Internally it uses an instance of `TrackPlayer` to wrap each `AudioTrack`, which implements `Runnable`. Each channel is associated with a separate `Handler` object, but they are all handlers for the same `Looper` object which runs on a background thread. The `TrackPlayer` can be submitted to a Handler object to start the playback, and also has methods to stop and pause playback, which can be called through the `IAudioManager` interface.

The `AudioManager` also implements a Shared Preference change listener, so it can be notified of configuration changes during playback and act accordingly, for example, disable channels, change sounds, change frequencies and so on.

Much thought has gone into designing and implementing the `AudioManager`, making it as performant, modifiable and easy to use as possible. It supports playing back any arbitrary `AudioTrack` supplied to it from the `TrackFactory`, whether the audio content is generated by a generator or loaded from a prerecorded PCM sample. The implementation supports all requirement as specified by FR-NAV.01, FR-NAV.02, FR-NAV.03, FR-NAV.07, FR-NAV.09, FR-NAV.13, FR-NAV.14, FR-NAV.17, FR-NAV.019, NFR-NAV.06 and NFR-NAV.08.

4.6.6 Algorithm details

The actual sonification algorithm, which transforms an input RGB value to output values is rather simple. The following describes a simple conversion from RGB to RYGBW values.

1. Set Y to minimum of R and G
2. Set MIN to minimum of R, G and B
3. Subtract MIN from R, Y, G, B
4. Set W equal to MIN
5. Return RYGBW

The optimal algorithm is still being researched, and this is just an example of the most basic version. The application supports changing algorithms at run-time, but they have to be implemented and compiled beforehand. The algorithm interface has been made as simple as possible, to make it easy to write and add new algorithms.

Code 1: Implementation of the BasicRygbAlgorithm

```
1 public class BasicRygbAlgorithm implements ISonificationAlgorithm {
2
3     @Override
4     public AlgorithmOutput sonificate(int red, int green, int blue) {
5         int yellow = Math.min(red, green);
```

```
6           int min = Math.min(yellow, blue);
7           yellow -= min;
8           red -= (yellow + min);
9           green -= (yellow + min);
10          blue -= min;
11
12          return AlgorithmOutput.obtain(red, yellow, green, blue, min);
13      }
14  }
```

The code required to add a new algorithm to the application has been made as simple as possible. The following listing shows how algorithms are registered with the application in a way which lets the settings activity display the algorithm as a selectable item, and also lets the `AlgorithmFactory` instantiate the algorithm when needed.

Code 2: Registering an algorithm

```
1       registerAlgorithm(getStringRes(R.string.algorithm_basic_rygb),
            getStringRes(R.string.algorithm_basic_rygb_key), new
            AlgorithmFactory.IAlgorithmFactory() {
2               @Override
3               public ISonificationAlgorithm create() {
4                   return new BasicRygbAlgorithm();
5               }
6           });
```

The parameters to `registerAlgorithm` in order are:

1. A human readable name for the algorithm
2. A unique key for the algorithm used by the Preference framework
3. An instance of IAlgorithmFactory which instantiates and returns the algorithm

This exact pattern is also used for sounds, which makes it equally as easy to add new sounds to the application.

4.7 Known issues

Although the product is considered complete, the group did not have time and resources to resolve all problems with the system. The list shown below contains known lacks and bugs of the system. Note that all of the bugs are reported to Waffle for future system development.

- The start screen shows the content of the Colorophone website, but does not indicate it. It also outputs an error message in case of no Internet connection, and has no option to reload.
- The app crashes when tapping the ”switch camera” button repeatedly.
- The Image Viewer and Game features have a memory leak.

- The specified email address is not real.
- Sometimes after having used the app for a long time, the sound disappears. The app needs to be restarted to work again.
- Changing frequency generated sound channels is inconvenient, because the system defaults to the upper and lower limits.
- The game has incorrect alignment of the content in certain states.
- Some game images do not scale correctly.
- The game does not warn user if touching outside image boundaries.
- In the game, for some screen resolutions in landscape mode, text on answer options cannot be seen.
- In the game, it is possible to press behind the confirmation box.

4.8 Future development

Many features could be implemented or improved, and parts of the code that could or should be refactored. This section contains some of the most prominent improvements that could be made to the application in the future.

4.8.1 Test coverage

Because the application has been developed as a rapid prototype with continuously changing requirements and features, it does not have a very good test coverage. This is usually considered acceptable for such prototypes, but the application is currently at a level where the core functionality is becoming more stable, and a natural next step is to implement additional tests and improve test coverage. This includes both unit tests and instrumentation tests, and should likely be done before any new features are added.

4.8.2 Sonification as background service

One of the most important features for the application to be usable in a real world scenario is to be able to run the sonification process as a background service while the phone is locked and without a preview. This is needed both to let a user use the device for other tasks without interrupting the sonification, but also to preserve battery since the preview has no purpose for the target demographic. Android does not easily allow the camera to run in preview mode in the background without a display for privacy reasons, so certain workarounds would need to be implemented, but it is certainly possible. This would be a natural progression of the application.

4.8.3 Peripheral support

The original paper written by the customer[37] discusses the use of peripheral input to enhance the sonification, by letting the peripheral parts in the horizontal plane of the input translate to white noise in stereo. This noise was separated into a peripheral and transitional area, with varying intensity. The current application has been built with support for peripheral sonification in mind but does not implement all the needed parts. Most importantly, the algorithm for peripheral and transitional sonification has not been implemented. This is per customer request, as this part of the process is still under research and not considered as important for the system as a whole. That said, complete support for peripheral sonification could probably be implemented in a few hours granted the algorithms were defined.

4.8.4 Custom Android ROM

It is theoretically possible to heavily optimize the battery performance by providing a custom distribution of the Android OS which disables all unnecessary components at the hardware level, such as the screen and sensors. This would, however, make the device unable to perform any other tasks. This could be considered as an option in the long term to make the application usable on older, cheaper devices that can be supplied to less wealthy populations.

4.8.5 Extending game

Extending the game and making it more professional would be a priority continuing this project, but due to time restrictions finishing this game proved not to be feasible. For this project, it was only time to create the first prototype showing the possibilities of the game, where users try to interpret an image using the Image Viewer feature, then guess which image it is among a list of alternatives.

There are three ideas behind this game. The first one is to collect scientific data from the user, by looking at their games. This can be done by setting up a server and having the app send the data over the internet. The second idea is to draw attention to the Colorophone project by letting everyone try out the game and see the possibilities of this technology. To do this one would have to add some gamification to keep the users interested. The third is a way for the target demographic to train and familiarize themselves with the different sounds before using it in the real world. To do this, the game has to be optimized for visually impaired users, either by using Androids built-in talkback-function or by giving the user feedback another way. What these three ideas have in common is that creating a database to store all the images in is a big priority, this will make it much easier to add new levels to the game, and it scales much better overall.

4.8.6 Application flavors

The focus of this project has been on the Colorophone Research version of the application, but there is an intention to eventually release multiple flavors of the application, including a version for users with vision, and more importantly a version to aid visually impaired users.

The build files and build script has been set up to support multiple flavors within the same code base, but these flavors have not been implemented yet. The work mainly revolves around limiting functionality or removing features from the Research version, which can be done through if statements or similar constructs based on the current build flavor.

5 CLRF Beacons

CLRF Beacons is a subproject featuring Bluetooth low energy beacons. The goal of this project is to develop an application for Android devices that scans for nearby beacons with Bluetooth to deliver location awareness to the user via audio cues. Each beacon should contain an identifier that maps to beacon information. That information should be stored in a local database on the phone and be updated monthly via an external database. The application should send the message from beacons in range to the phone as a push notification and read it out loud to the user via Text-to-Speech. The beacon should broadcast its identifier as a signal to nearby devices that are beacon compatible. The purpose of the implementation is to explore the possibilities and limitations of Bluetooth beacons, and present these to the customer as a proof of concept.

The section starts by describing the prestudy phase, followed by a set of requirements. Then, the system architecture and design are explained. In the end, proposals for future development are discussed.

5.1 Prestudy

The prestudy phase consisted of research of relevant technology and a comparison of this project to similar existing systems. It was also vital to decide whether these similar systems or parts of them could be used as a code base for this project and if those systems were competitors to the system developed in this project.

5.1.1 Technology research

Beacon technology had to be researched. The article *How do beacons work? The physics of beacons*[9] provided insights into the fundamentals of beacons and how they work. Several libraries for beacon interaction were considered. It was decided to use *Android Beacon Library*[43] - a library that provides an API to interact with beacons. It was chosen because its ability to be used with different beacon protocols and because it could be used to implement a lot of the requirements for the system. The other options included using Nordic Semiconductor's beacon application for Android and use its open source code[34] as base and inspiration. Those options would not be as flexible as the one chosen.

5.1.2 Existing solutions

Bluetooth beacons are implemented in many different domains, included in blind aid systems. Only beacon systems that are used in the blind aid domain have been considered

here.

Wayfindr is a navigator system that uses beacons to help the user navigate. It is an open standard that has been implemented as an iOS-app for testing purposes. The iPhone app uses Text-to-Speech to tell which directions to take next based on the location of beacons and the information stored in the app associated with those beacons.[33] The system has been tested on locations in London[39] and Sydney[14] for navigating in railroad stations.

Another similar system has been tested in Strasbourg in France. Beacons connected to buses are used to inform users about incoming buses to help the user know which bus to take. This system also uses Text-to-Speech to communicate this information. [29]

Because the purpose of this subproject is to explore the technical limitations and possibilities of Bluetooth beacons, these similar systems are not competitors. The code from either of the systems could not be used as a base for this project because the Wayfindr app is implemented in iOS, and the system in Strasbourg is not open source, so the code is not openly available.

5.2 Requirements

Following is a list of requirements that have been created during the process by the customer and project group. These requirements are not final, due to the beacon project being terminated. They should rather be used as a framework for how the system should work in the future.

5.2.1 Functional requirements

ID	FR-BCN.01
Title	Receive information
Role	End-user
Description	Must be able to get information about nearby beacons to the phone

ID	FR-BCN.02
Title	Manage database
Role	Developers
Description	Must be able to add, remove and update beacons from database

ID	FR-BCN.03
Title	Background scan
Role	System
Description	Must be able to scan for beacons in the background

ID	FR-BCN.04
Title	Bluetooth scan
Role	System
Description	Must be able to scan for beacons via Bluetooth

ID	FR-BCN.05
Title	Text-to-speech
Role	System
Description	Must be able to read out information associated with beacons through text-to-speech

5.2.2 Quality attributes (non-functional requirements)

5.2.3 Performance

ID	NFR-BCN.01
Title	Scan time
Role	System
Description	Must be able to scan for beacons every 10 seconds for 2 seconds

ID	NFR-BCN.02
Title	Send notification
Role	System
Description	Must be able to send notification when user is in a 15 meter range of beacon

ID	NFR-BCN.03
Title	Start text-to-speech
Role	System
Description	Must be able to start Text-to-Speech 3 seconds after a beacon is found

5.2.4 Power efficiency

ID	NFR-BCN.04
Title	Operable time
Role	System
Description	Must be able to be operate for at least 7 hours, and 12 hours in background mode

5.2.5 Usability

ID	NFR-BCN.05
Title	Android API
Role	System
Description	Must be able to support at least Android API 18

5.3 Architecture

Figure 20 shows the planned overall architecture of the CLRF Beacons subproject. An Android device interacts with Bluetooth low energy beacons and with an integrated, local database and an external database. This architecture pattern is a type of a client-server architecture as described by Sommerville [47, p. 162] where the external database is the server and the Android device is the client.

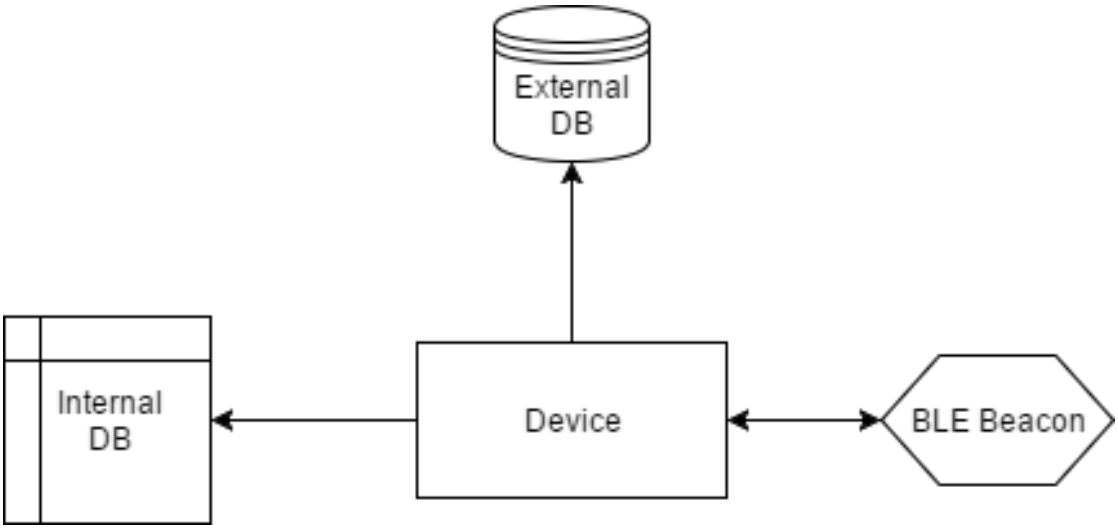


Figure 20: CLRF Beacons architecture

The following sections provide a view of the architecture based on the 4+1 View Model as presented by Sommerville [47, p. 153].

5.3.1 Development view

Figure 21 depicts an overview of the packages with the package names, the classes, and interfaces they contain as well as their communication among each other.

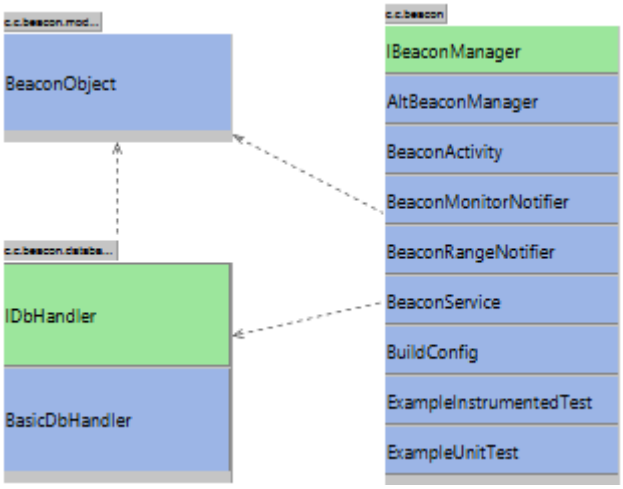


Figure 21: CLRF Beacons packages

com.colorophone.beacon Contains the main part of the functionality. This includes Text-to-Speech, beacon detection and Bluetooth setup, notification logic for when a beacon is in range and more.

com.colorophone.beacon.database Contains the database logic that is used to hold data about beacons.

com.colorophone.beacon.model Contains the beacon model which is used when retrieving data from or saving data to the database.

5.3.2 Process view

Figure 22 shows the workflow of the beacon application. As the app is started, the user is asked to allow enabling of Bluetooth connections. If the access to this function is denied, the application shuts down. Otherwise Bluetooth is activated, and the search for Bluetooth low energy beacons starts automatically and runs in the background. In case one or more beacons are detected the app will receive their UUID and search for a match in the local database. Nothing happens unless a match is found. In this case, the beacons type, and its related message will be retrieved from the local database and, together with the UUID, presented on the screen. Additionally, the message will be read out loud by the applications Text-to-Speech function.

The sequence diagram shown in figure 23 depicts object interactions in a time sequence for the application. Some method calls are excluded for clarity. The diagram shows how the `BeaconActivity` class sets up the services, makes use of the other classes and how the client-server architecture is used in practice.

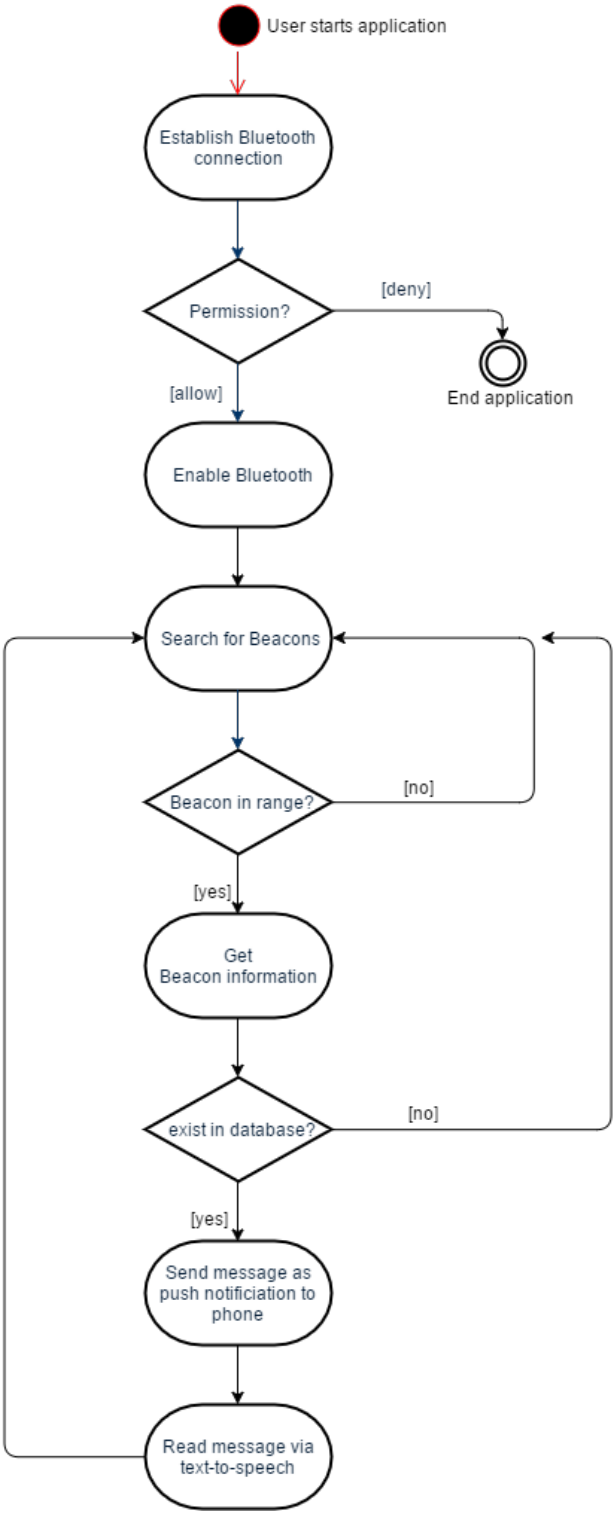


Figure 22: CLRF Beacon activity diagram

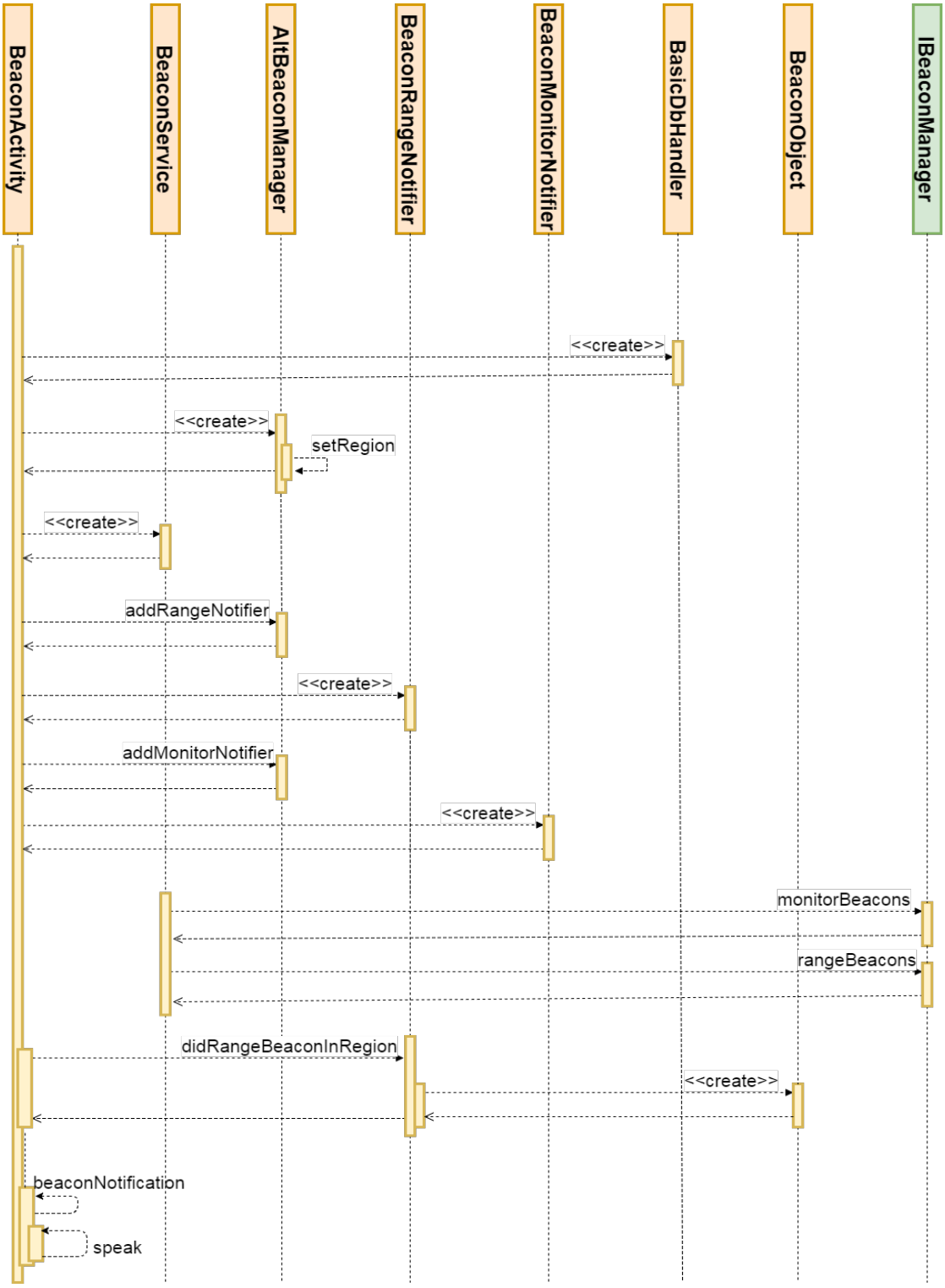


Figure 23: CLRF Beacon sequence diagram

5.3.3 Physical view

Since the application is only a proof of concept, it is not available through Google Play Store. It can, however, be installed via USB, email or any other media that can carry data. Android version 4.3 is the minimum requirement for all devices.[2] Thus, the execution environment must support API 18 or higher.[22]

5.3.4 Logical view

Figure 24 depicts the class diagram for the entire subproject. The central part is the `BeaconActivity`. Here, the search using Bluetooth and Text-to-Speech service is started. In addition to that, the `BeaconService`, that implements the BeaconConsumer interface that is part of the AltBeacon Library [42], must be used since it enables, together with the BeaconManager interface in the `AltBeaconManager` class, the use of the `BeaconMonitorNotifier` and `BeaconRangeNotifier`. These allow the user to monitor how many beacons are in range and which of them are in the database, respectively. Despite its name, the `AltBeaconManager` class also provides the functionalities to find both iBeacon and AltBeacon Bluetooth beacon types. The `BeaconActivity` class also sets up the `BasicDbHandler` for a consistent and efficient use of the local database which is filled with `BeaconObject`s.

Figure 24: CLRF Beacons class diagram

5.4 System design

5.4.1 Detection notifications

The application currently notifies the user about nearby beacons that are also in the local database in three ways:

- TextToSpeech is an Android class [24] that is used to give audio feedback by reading out loud a beacon's associated message.
- Push notifications are used to see whether the application continues to detect beacons while the user is not inside the app. In this case the Text-to-Speech function is not activated but the user gets a push notification with some information about the detected beacon. [23]
- Figure 25 depicts the screen of the CLRF Beacons prototype app, when no beacons are detected. As the beacons get detected, the screen will be filled with a list of beacons showing their UUID, beacon type and message. This functionality is used for development purposes only.

Figure 25: CLRF Beacons screenshot

5.4.2 Beacon detection

The application uses a 3rd party library [42] to detect Bluetooth beacons that use the AltBeacon and iBeacon protocols.

5.4.3 Databases

Local database SQLite and SQL were used to provide database services to the application and are implemented through the `BasicDbHandler` class. Each row in the database consists of the following four fields:

- Primary Key - an integer identifying the beacon in the database
- Beacon ID - a field of thirty-six characters representing a beacon's UUID, used as index
- Beacon Type - a string/text field representing the beacon type
- Message - a string/text field for the message related to that beacon

The purpose of the local database is to hold information about a number of physical low energy Bluetooth beacons in a manageable way. Thus, when a beacon is detected, the application will use the detected UUID and search for it in the database. If a match is found, the corresponding data is retrieved and presented to the user.

External database The internal database would download content from an external database to populate or update its own content. This was not implemented before the termination of this subproject. See section 5.6.1 for a description of how this could have been done.

5.5 Termination of CLRF Beacons subproject

In Sprint 5, the customer decided to discontinue the work on this subproject because he wanted to focus development on a practicing game in the Researcher app. See the summary of sprints 4 and 5 in section 7.3.4 for details. The group was told that the current implementation served as a proof of concept. The app proved a way to do what the customer asked for and fulfilled the functional requirements. Finishing this subproject by refactoring code, properly testing it, fulfilling the non-functional requirements and adding use cases, optional features and a user guide to the documentation had no real value to the customer. As a consequence, no formal tests were done by the end of the project. The only testing outcome of this subproject was the unit tests that were written during development. The following functional requirements were finished: FR-BCN.01, FR-BCN.02, FR-BCN.03, FR-BCN.04, FR-BCN.05 and NFR-BCN.05. The remaining functional requirements were not tested and verified formally.

5.6 Future development

Features that were discussed but not implemented are listed below, together with an explanation of the intended functionality as well as possible implementations and their

limitations. Also, some aspects of the application design were not finished due to the discontinuation of development. The application needs further focus on user interface design and user experience design.

5.6.1 External database

The implementation of an external database would serve multiple purposes. The external database would reside on an online server. The customer would add and remove beacons when appropriate. The local database could use the external database to update itself according to some schedule automatically. As the external database would grow in size, containing beacons from multiple geographical regions, it could be useful to only have region-related beacons in the local database in order to save space and keep performance stable.

5.6.2 Distance indication for beacons

It is recommended to add some indication of distance. That way, the user will know how far he or she is away from where the beacon is installed. This could potentially be done using a beacon's signal strength. The stronger the signal, the closer the user is to the beacon.

For the implementation to provide an efficient and useful solution, multiple obstacles would be encountered:

Signal strength altering There are many conditions that either amplify or reduce a beacon's signal intensity like the weather, objects, walls and so on. Thus, distance indication algorithms could deliver wrong results. One possible implementation could use not only a single beacon's signal strength, but also the nearby beacons to keep track of where the user is relative to a specific beacon. This comes with four disadvantages. The first being that each beacon would need more space in the internal and external databases by adding beacon-to-beacon distances to some other nearby beacon. The second problem would be the increased battery usage. An algorithm would need to be run to check whether the distance indicated matches with the beacon's actual distance by using the relative distances between the targeted beacon, the beacon closest to the user and the beacons in between those two. Third, the maintainability. The internal database must constantly be up to date to make this work. Lastly, this will not function as well in locations where beacons are thinly dispersed.

Differences in height If the beacons are on different floors in a building, the user must be made aware of that. The easiest way is to simply add a field to the internal and external database containing the beacons floor number. This comes with the cost of increasing the database sizes when many beacons might not need that field.

Avoid spamming distances In order to let a user have a good experience with this feature, it is important to implement vocal indication in a smart way. Once a beacon is detected the user should not constantly be updated with the distance to that beacon. A good start would be to mention a detected beacon once. If the beacon is relevant to the user, he or she could shake the smartphone to "lock-on" to that beacon and regularly get distance updates for it.

Moving targets Beacons installed on moving targets like vehicles for public transport would need special treatment regarding their distance indication. Multiple important situations need to be taken care of in an efficient and user friendly way. The app should be able to handle bus stops where multiple buses stop while others pass by. Also, the app must not spam messages about detected buses when the user sits on a bus that drives through the city.

5.6.3 Database migration tools

To be suited for future use, the internal database must be able to automatically manage changes in the database schema of the external database. This means that in case fields are added or removed from the external database schema the internal database has to automatically make necessary adjustments to existing elements and make sure that new ones have the right set of fields.

5.6.4 Eddystone compatible search

At this point in development, the app can detect beacons of the iBeacon and AltBeacon types but not the Eddystone type. For increased flexibility, it should be able to find all three types of Bluetooth low energy beacons.

5.6.5 Handling change of UUIDs

For security purposes, the beacons the team worked with automatically changed their UUID after some time. If the beacons used in the future are configured to do this, it has to be handled properly. Beacons not intended for use with the app could further complicate the matter. If an unrelated beacon changes its UUID to one that is in the database of the app, it could mislead the user to a potentially dangerous degree, and relevant beacons could not be found anymore. Future developers should determine when and to what a beacon's UUID will change or determine it with an algorithm in order to keep the security aspect of this property. Additionally, changes have to be made to the internal database, so it stays up to date as the physical beacons change their UUID.

6 Testing

Testing is a vital part of all software development projects, and it is important to have a test strategy describing how the testing is planned to be conducted. Tests have been planned at different levels, all the way from unit tests to acceptance tests, to ensure that the final product will be reliable and of high quality. This section contains a description of the types of tests that have been used, how they have been used, and important results.

The complete test stack has been applied to the CLRF Navigator subproject. However, as the CLRF Beacons subproject functions more as a proof of concept, the customer did not want the group to spend time conducting a final testing phase for this subproject.

6.1 Continuous integration

The group has chosen to adapt to the practice of continuous integration, which involves frequent merging of feature branches into the master branch [17]. Travis CI has been used as a client, see section 2.7.2. As a part of this practice, every time a branch is merged into master, the entire system is built from scratch, and all automated tests are run. In principle, all code needs to be working before integration into the master branch. This way, it is ensured that the system is always on track, and errors can be detected and fixed as early as possible.

6.2 Manual inspection

Manual inspection of code is a useful addition to code-based tests. Manual inspection will eventually lead to more readable code as non-consistent code and confusions are pointed out. Additionally, it allows for removal of certain bugs that would be hard to detect otherwise. In the development process, manual inspection has been used actively. Before any integration into the master branch, a review of the pull request has been required. The code should be inspected manually as part of this review.

6.3 Manual testing

Manual testing has been done through the whole process by reviewing pull requests or testing the current master branch thoroughly. Manual testing was done extensively in the latter part of the project, about the time of camera integration and before the product demonstration. This was done informally by individual group members, with the result of submitting bugs and improvement suggestions to the Waffle issue tracker system. An example of how bugs were reported and corrected using Waffle is shown in

appendix C.3.

6.4 Unit testing

Unit tests focus on the testing of small, individual program components like methods and object classes [47, p. 211]. For this project, JUnit has been used as testing framework. All of the unit tests are automated, and they execute automatically by Travis before any changes to the master branch. As a main rule, unit tests should be written at the same time as the code or shortly after. However, when the specification of methods or classes is unclear or likely to change, the implementation of unit tests has been postponed. Also, much of the code has been difficult or inconvenient to test properly by unit tests. Then, the code has instead been tested manually.

An example unit test of the BasicRygbAlgorithm is shown in code listing 3.

Code 3: Example unit test of the BasicRygbAlgorithm

```
1       @org.junit.Test
2       public void mixedValueSingleMinimum() {
3           AlgorithmOutput result = fixture.sonificate(255, 150, 100);
4           assertEquals(155 - 50, result.getRed());
5           assertEquals(50, result.getYellow());
6           assertEquals(0, result.getGreen());
7           assertEquals(0, result.getBlue());
8           assertEquals(100, result.getWhite());
9       }
```

6.4.1 Instrumentation unit testing

Instrumented tests differ from normal tests in that they depend on the APIs provided by the Android platform, and are usually run on physical devices or emulators instead of on the local development environment [4]. As much of the code is dependent on Android code which is unavailable outside the device, instrumented unit tests have been necessary to use. Instrumented unit tests are automated as well, and are run regularly by Travis. Some instrumentation tests can be run without a physical device or emulator, instead using a library named Robolectric and its test runner. Robolectric provides shadow implementations of certain common Android APIs such as the Android context, but provides no real implementation.

An example instrumentation test of the MainActivity is shown in code listing 4.

Code 4: Example instrumentation test of the camera button in MainActivity

```
1       @Test
2       public void clickCamera_shouldStartCameraActivity() {
3           MainActivity mainActivity = Robolectric.buildActivity(MainActivity.
                class).get();
4           MenuItem menuItem = new RoboMenuItem(R.id.nav_navigate);
```

```
5           ShadowActivity shadowActivity = Shadows.shadowOf(mainActivity);
6           mainActivity.onNavigationItemSelected(menuItem);
7           Intent expectedIntent = new Intent(mainActivity, CameraActivity.
                class);
8           Intent startedIntent = shadowActivity.getNextStartedActivity();
9
10          assertTrue(expectedIntent.filterEquals(startedIntent));
11      }
```

6.5 System testing

System testing focuses on the complete system, and in particular the integration of subcomponents into a total system [47, p. 42]. A formal system test was done in the late part of the project, after the coding deadline. The focus was testing whether the system behaves according to the customer requirements, and finding bugs or erroneous information. Any comments were noted and added to Waffle, mainly as a note for future development. There was not time to actually correct any bugs. The plan and results are shown in appendix E.1 and E.2, respectively. Only few improvements were detected and noted, as improvements in general have been proposed and corrected continuously during the implementation phase.

6.6 User testing

In user testing, potential users of the system can give input and advice on the current product [47, p. 228], which is of high value for improving the system. User testing may be important because users often use the system in a different way than expected by the developers, and they often have other prerequisites and expectations. User testing can be done both formally and informally [47, p. 228]. Informal user tests have been done regularly, mainly when the customer has attended meetings with the group and has got the possibility to use the current version. A formal user test of the researcher app was performed late in the process, and is described below.

The user tests were conducted at May 5th, just after the feature freeze deadline, and 6 persons tested the app. The test persons were provided by the customer. All of them had a relation to the Colorophone project, and were potential users of the researcher app. During the test, a list of tasks was performed by the users, monitored by the test leader and a referee. The tasks, attached in appendix D.2, mainly consisted of a selection of the requirements, written in a more natural language. A detailed plan is attached in appendix D.1.

6.6.1 Results

This section contains a summary and analysis of the results of the users tests. The detailed results can be found in appendix D.3. Most importantly, many small usage problems were identified, added to the Waffle tracker system and corrected. This includes:

- In the Camera, the sound should be turned on by default.
- The structure of the Learning feature should be changed to make it easier to find the game.
- The app should give a confirmation message when trying to reset all settings to default.
- Some of the names on menus and settings should be renamed to be more intuitive.

In addition, a couple of bugs were discovered and added to Waffle. Also, positive user experiences were identified. This includes the clicking and swiping behavior of the Image Viewer, and a smooth and intuitive transition to the external camera.

Interestingly, many people expected to find color-related settings in the "general" category instead of in the "audio" category. They did not find the link between colors and sounds to be intuitive at first glance. However, an important concept of the Colorophone project is this particular link, and thus the menu was not changed.

In general, after having tried doing the tasks and struggled a bit using the app for about 15 minutes, most people told that they found the app intuitive and easy to use. The app needs only a quick learning phase for the users to be comfortable in using it. Also, the app proved to be very stable, with a crash happening only once. Thus, the group considers the app at the current stage to be a success.

As a side note, almost all participants succeeded in the last task of guessing which color corresponding to a given sound. This happened even though the app had been used for only 15 minutes, with a main focus on usability of the app. This indicates the success of the sensory substitution concept.

6.7 Acceptance testing

Acceptance tests are the last tests to be conducted. In these tests, the customer tests the product in his own environment and decides whether the product is satisfactory or not [47, p. 42]. An acceptance test was conducted about May 24th, some days after the code freeze. The customer downloaded the app from Google Play, and was free to use it for himself for some time and write some feedback, both general and related to the requirements. The result was positive, and the customer was seemingly very pleased with the product. The resulting document can be found in appendix F.

7 Summary of iterations

This section contains a chronological summary of the progress of the group, divided into sections of roughly two sprints. Most sprints lasted for one week, starting on Wednesdays. The complete documentation of the process can be found in the separate attachments "Meeting Minutes" and "Meeting Agendas".

7.1 Planning and sprint 1 (week 2-5)

7.1.1 Planning

A lot of the planning in the start was used to decide the workflow of the project. Before the group could start on the actual project, it was important to define rules, meetings, and tools. It was also important to learn more about Colorophone and their product.

7.1.2 Results

This phase mainly consisted of project planning and getting started working on the project. The group finalized a team contract, see Appendix A, to help the group from falling into common mistakes. A Slack channel was created for communication, and most tools to be used during the project were decided, see section 2.7 and 2.8. The group members' expectations were discussed, and their schedule was surveyed. It was decided to meet twice a week, with optional work meetings. Sprint meetings were decided to be held on Wednesdays, and status meetings on Fridays. In addition to these meetings, there would also be occasionally group leader meetings at Wednesdays, and every other week supervisor meetings on Fridays. It was also decided that the length of the sprint iteration should be one week. A project guidelines document was written, see Appendix B, to keep development consistent between group members. The first meeting with the customer was held, and sprint planning and requirement specification was conducted. After the customer meeting the group divided into two teams and started working on the two subprojects CLRF Navigator and CLRF Beacons, as instructed by the customer. A very simple interface and implementation of the core logic was written for the Colorophone Research application by the CLRF Navigator team.

7.1.3 Reflections

The positive things during this phase were that the group had a good start. Much documentation was written early, and the group came to an agreement about important things regarding the project's workflow. It was also positive to get in contact and arrange a meeting early with the customer. A few of the negative things were that the beacon

team could not get in touch with Comarch[13], the producers of the beacons assigned from the customer, to get technical help with the beacons. Also, none of the beacon team members had any previous experience with Android development. At this stage the group could improve on teamwork, working together in smaller groups and pair programming. It was decided that there should be more meetings to work together, and the meeting times should be clear and concise for everyone. Moreover, if group members are unable to meet for additional meetings, work should be coordinated accordingly.

7.1.4 Summary

The first sprint ended overall very well, most of the tools were decided, and the group as a unit gained a clear vision of what to achieve during the project. The plan for the next sprint was for each team to familiarize with their projects and write requirements before starting to work.

7.2 Sprint 2 and 3 (week 5-7)

7.2.1 Planning

After the first sprint, the group learned a lot about the Colorophone project. The customer wanted the group to work on two separate projects in the start (CLRF Navigator and CLRF Beacons). Thus, it was naturally to divide the group into two teams. The first week went into researching possibilities within the two teams, in addition to writing requirement specification for each team project. The group planned on having more frequently meetings with the customer, in the beginning, to help avoid misunderstandings. It was also planned to keep contact regularly with the customer, preferably via Slack. It was decided that available free time should be spent on the project report, the more work that could be done early would only help later. There was also a preliminary version of the report due to deliver during the next sprint.

7.2.2 Results

The requirement specification was completed for both the current projects so that each group member had a clear vision of what to achieve. After finishing all documentation, the group had much time working on new features.

CLRF Navigator This team worked on the basic algorithm that was to be used in the application. The algorithm now handled input, process, and the output pipeline. A settings menu and a debug user interface (UI) was made to complete a fully working prototype, which was demonstrated for a satisfied customer.

CLRF Beacons This team, on the other hand, experienced a few problems during this sprint. Only one out of three smartphones from group members were compatible with the Comarch beacons received. The batteries on the beacons were also dead and since they were a unique type of batteries the team had problems finding compatible ones. As soon as these problems were fixed, the team was able to get work done. All of the beacons were set up and configured correctly, and a separate application was made. The application now handled searching and discovering beacons, as well as reading messages from beacons via Text-to-Speech.

New tools for the project were introduced to help the group's efficiency and workflow. Toggl [50], a time tracking tool, was introduced to measure how much time the team spends on different aspects of the project. Continuous Integration by Travis [51] was also introduced to require group members to review written code before merging pull requests. The group decided to use Material Design, Google's design guidelines, for the created applications. It was also agreed to create a risk assessment (see section 2.9.4) to identify potential risks that could cause harm to the group.

7.2.3 Reflections

The positive things during these sprints were that the group had a lot of exciting features to work on. It was also done much research on both projects, which made the teams excited about future sprints. A few negative things were the troubles the beacon team experienced with incompatible smartphones, which delayed the development. The group realized it could improve on making the estimations for tasks to be more accurate, which was difficult due to the group not having experience with these types of tasks and it was hard to estimate how much time it would take. Another suggested improvement was that the team could work more together. The teamwork was good, and the group members worked efficiently together, but finding the time for additional meetings was difficult due to different schedules.

7.2.4 Summary

Sprint 2 started a little troublesome, but the group kept the motivation up and managed to handle the challenges. During this sprint, the main focus was completing a lot of the documentation for the project, but at the same time making sure that features were implemented and having a working prototype by the end of the sprint. The plans for the next sprint involved implementing new features for both the Colorophone Research app developed by the CLRF Navigator team and the beacon discovery application developed by the CLRF Beacons team. The CLRF Navigator team planned to improve the settings and debug UI, while also add support for peripheral mode and channel audio sample files. The CLRF Beacons team planned on adding a database which would contain the beacons, add search for beacons in the background, add range detection for beacons and improve the UI according to guidelines.

7.3 Sprint 4 and 5 (week 7-9)

7.3.1 Planning

Sprint 4 started with the preliminary report being due at the beginning of the sprint. Tasks regarding the report had already been delegated from the previous sprint. Now it was all about finishing up and continue spending more time on the report whenever time would allow it. The group planned to showcase both of their applications for the customer. During this meeting, the plan was to receive feedback and directions on where to take the project in the future. It would also give the opportunity to show implemented features, which features should be prioritized and open up for discussion. The group was invited to a conference, Technoport, that would occur the next sprint. The group planned to attend a stand during the conference. The focus for this sprint was to make sure that the Colorophone Research application was ready for the conference. At this stage, the Image Viewer was the most important feature to implement.

7.3.2 Results

The project continued to progress well during this phase, as team CLRF Beacons prepared to stop working on the subproject and rather start working on the external camera integration. Before ending the work on the subproject, new features were implemented. The beacon application now had a local database for beacons and could search for beacons in background mode. The team also received beacon development kits from Nordic Semiconductor[35], which were configured as Eddystone beacons for future development. Unit tests were written, and refactoring of beacon handling was done. The CLRF Navigator team started implementing the Image Viewer feature for the Colorophone Research application. The team also continued implementing settings to change between different generated sounds, change frequency and disable each channel. Support for peripheral vision was added to make it easier in the future to complete this feature. Also, overall refactoring for better performance and implementation of unit tests was done.

7.3.3 Reflections

The positive things during these sprints were the progression that the group achieved. Both applications were working well, and the group members met physically to work together. It was also positive having a social gathering with everyone from the extended Colorophone team. Negative things during the sprints were the amount of time spent on some tasks, which made the group too dependent on these tasks. Therefore, group members had to wait for the task to be completed before starting another task. The CLRF Beacons team received a different beacon kit than expected, which caused some extra work. It was discussed that the group as a whole could improve on creating new tasks, or break them up into smaller tasks. It was also brought up that improving the way the team did testing could be beneficial, where it is spent more time writing tests,

and less time waiting for them to run. It was also decided that additional meetings should be added to the group calendar.

7.3.4 Summary

These sprints were quieter than the previous ones, due to the team being more settled in. This gave the group the chance to focus on working on new and existing features fully. Different sections in the project report were delegated to team members. The customer requested the subproject CLRF Beacons to be terminated. The proof of concept was approved, and it was decided that it was more important to integrate it with the CLRF Navigator application instead. The customer also requested that the group should make a gamification concept of the Image Viewer. The group would have to make a decision the upcoming sprint whether this should be done or not.

It was planned to prepare the CLRF Navigator application in the next sprint for the Technoport conference. It was also planned to write and deliver the midterm report, which was due the week after Technoport. The remaining time would go into working on the external camera integration.

7.4 Sprint 6 (week 9-11)

7.4.1 Planning

During the planning for this sprint, it was decided to change the sprint duration from one week to two weeks. The group was invited by the customer to attend the Technoport conference at Clarion Hotel in Trondheim and attend a stand. Therefore, the main focus for the first week of the sprint was to prepare the CLRF Navigator application for public viewing. The second week was focused on report writing, and to prepare the project report for the midterm delivery.

The team now was working on the external camera integration planned to do some research on how to complete the task. The main purpose of this project was to make it work however possible. The group also needed to decide whether to create a game as suggested by the customer, where the user tries to identify objects in images by sonification similar to the Image Viewer but with the image hidden. The customer only wanted the game to be made if there was time available for it. A meeting to discuss the game and its requirements was needed before the group could make a decision. However, either way, the team would not start on the game during this sprint.

7.4.2 Results

Navigator The first week of this sprint was used to prepare the Colorophone Research application for the conference. No new features were started on before the conference, as it was a focus on bug-fixing and polishing the application. After the conference work on prerecorded audio started, where the color sounds are made into instrumental sounds instead of just sine waves. It was close to being complete, for now, it contained violin and cello. More testing with different samples and settings was needed, the group planned to finish this before the next sprint.

External camera integration Progress on the new feature was held back due to not having much time left in the sprint. A known problem also occurred; no team member had a compatible phone that would work. The group solved this issue by bringing over a group member from the CLRF Navigator team, that had a compatible phone. The external camera was confirmed to work when plugged into the phone via an on-the-go (OTG) cable.

The group also held a meeting with the customer to update the requirement document. For the most part, the group had proposed requirements themselves, and the discussion helped to get everyone to agree on each requirement. On the same meeting, it was decided to split the main application into two applications. One application would target end-users. It would be minimalistic with only the necessary features, such as the internal and external camera with sonification. The second application named Colorophone Research would contain every feature that was made. This application would be made for researchers and future developers, and its main purpose was to be used as a researching tool to gather more data in future user tests. The rest of the sprint was focused writing of the project report for the midterm delivery. Again, different sections within the report were delegated between the group members.

7.4.3 Reflections

During this sprint, the focus was on bug fixing, fixing existing features and the project report. The report required much time and slowed down the development of the application. A few negative things during the sprint was the small amount of new features to work on, much time went into writing on the report, but that also gave the group more control over it for impending deadlines. The external camera also gave the group a few headaches, as many team members did not have compatible smartphones.

7.4.4 Summary

The two-week sprint ended well. The group had much fun at the Technoport event, showcasing the application for interested attendants. The customer who also attended

the stand got to see the progress of the application and new features. The rest of the sprint was all about getting the report ready for the midterm delivery. Work delegation of the report made it much easier for the group to finish on time. The group planned to continue working on support for prerecorded audio for the next sprint and continue fixing bugs that were found throughout this sprint. It was also planned to start working on a tutorial/game implementation based on the Image Viewer. There was still some discussion to be made on what the best solution for this could be. Lastly, the group planned to continue the external camera integration.

7.5 Sprint 7 (week 12-13)

7.5.1 Planning

The group decided to create the game suggested by the customer, but not make it as big as originally intended. The group wanted to make the game into a tutorial for blind people, while still play and feel like a game for everyone. It was decided to start with a small scope, and at first, only make it so the user could learn the different colors and their respective sound. Then, if time, make the game more complex with help from the Image Viewer and sonification. Either way, the group would end up with a simple tutorial/game that could easily be developed further in the future to meet the original expectations from the customer.

7.5.2 Results

External camera integration Since the group did not have time to write its own library for external camera support, a few libraries were looked into. There were not any Android Studio built-in solutions and the libraries that were found only promised to work in the future. After some research, the group managed to find a library that worked this time, a library called UVCCamera[45], a library that enables the use of UVC web cameras on Android devices. There were some issues in getting the library to work correctly.

Game Two developers focused on the new Tutorial and Game features that were to be implemented in the Colorophone Research app. Much time, in the beginning, went into planning and brainstorming how the game should work and which features it should have. The team decided on three main features: a tutorial, to explain to the user how the game and the application in general works (a feature later discarded). The Color Map, a feature where the user can listen to different colors to learn their sound. Moreover, the actual game, where the user swipes the screen to hear the sound of a hidden image and guess what it is. A prototype of the solution was made on with the prototyping tool Proto.io[41]. The prototype was sent to the customer, and the team now waited for it to be reviewed, but since the customer was in China communication was difficult. The group started implementing their solution while awaiting feedback so that there was at

least some progression.

Image Viewer The rest of the team focused on refactoring and enhancing the Image Viewer. A side menu displaying relevant information about the color extraction was added, and the scaling between the image and the screen was fixed. The team also completed the first implementation of prerecorded sound, with a short sound to be used.

7.5.3 Reflections

The group was excited about starting the development of the game/tutorial. The decision to create the game brought out much discussion between team members about how it should work. The discussion was positive, and a lot of good ideas were made because of this. The team was eager to get feedback from the customer about the game, to see if the new features they came up with would be approved. The group was now divided into three smaller teams working on different features for the Colorophone Research app. This made coordinating additional meetings easier, and it would hopefully improve the group's efficiency overall.

7.5.4 Summary

Overall this sprint ended positive, there were many great discussions about the game. There were also much discussion regarding how the end-users should operate the application and receive feedback. There was some testing with Google's TalkBack mode[27] enabled and how the application should handle button presses. The group planned in the next sprint to start the implementation of the external camera library into the Colorophone Research application. In the start, it should be added to an extra activity, and later alter it, so it works with the application whenever it is plugged in or out. For the game, it was planned to implement the Color Map. The team focused on this feature because it would help the user learn how to use the whole application, not just the game.

7.6 Sprint 8 (week 13-14)

7.6.1 Planning

The group had a lot of different feature implementations going on at the same time, and all of them were progressing well. The one group of members focused on usability and refactoring to improve the Colorophone Research application further. It was planned to finish the Color Map feature and start implementing the game. A working library had been found for the external camera, and the group hoped to get the external camera working. The group also planned to have a meeting with the supervisor to receive

feedback regarding the report which would then be corrected and delegated between team members. Easter was getting closer, and group members would leave for the holiday at different times. Therefore, some less progress was expected.

7.6.2 Results

CLRF Navigator In this sprint the team worked on adding support for prerecorded sounds, this was one of the optional features discussed in 4.3.3. One of the struggles with this was to create a sound clip to be repeated over and over again and creating a nice sound to listen to. Significant parts of the application were refactored into a Model-View-Presenter architecture, and specific tests for this was written.

Game In this sprint the Color Map was finished. After this, the team began working on the game and the tutorial. For the game, the task was to set up the logic and navigation as decided in the prototype. For example, the questions should load randomly, and the user should double-click when ready to answer a question. For the tutorial, the task was to create a media player that the user would use to get information about how to use the application.

External camera integration In this sprint a library that worked with the camera was implemented. The task of making a new activity to get the external camera integrated with navigator was started and completed but crashed whenever it was connected to the external camera.

7.6.3 Reflections

The customer was in China during this sprint and was not able to review the game prototype. That means the team might have spent time on some features that were not approved by the customer.

7.6.4 Summary

By the end of the sprint, the group had a meeting with the customer to see the progression of the application. The customer was satisfied with the decisions about the game and the implementation so far, as well as the progression of instrumental sounds. The group planned to have more tests added in the next sprint for the Colorophone Research app, and refactor the existing code. The team planned to complete the game logic and add sound to make the first prototype. The team wanted to get the camera working correctly and implement it with the application.

7.7 Sprint 9 (week 16-17)

7.7.1 Planning

The group returned from Easter break and was now focusing on testing of the application and arranging a user test. Because the group had created an application for researchers, it felt natural that the user tests were focused on the usability of the application. As much as the group wanted to test the application on end-users, which are primarily blind persons, there were not enough time to gather people for this. The group discussed this with the customer, and the customer would find persons that could attend the user test. There was also a high focus on writing unit tests and complete a system test when the application was near completion. The customer gave the green light on the game implementation and thought the prototype was very interesting. The plan now was to continue working on the implementation of the game and have it ready for "Åpen dag" at NTNU. The group was invited to attend a stand during "Åpen dag" at NTNU which occurred at the end of the sprint, April 26th.

7.7.2 Results

CLRF Navigator In this sprint JavaDoc was added for all public APIs, and the code was refactored so that all activities were in line with the MVP pattern. More unit tests were created.

External camera integration The library was implemented with the application. So the external camera was now recognized whenever connected.

Game In this sprint the group got the prototype approved by the customer. The implementation started quickly since most of the planning had already been done. The team managed to implement the game logic for the game, and a tutorial for the game was created. The group also managed to start the work on state saving and the Image Viewer for the game but would complete these in the next sprint.

7.7.3 Reflections

The customer returned from China, as well as the remaining group members returned from Easter break. It was impossible to get contact with customer due to poor internet and Google being blocked there. Some team member had traveled to places without an internet connection, so it was tough staying in touch. Much work was therefore done were internet connection was not necessary, such as documentation. The group had a meeting with the customer to discuss the game, progression and what to do next.

7.7.4 Summary

With a little over one month left on the project, the group focused more on the testing aspect. The group hoped to have a user test sometime during the next two weeks. In this sprint, the external camera project saw some progress, as well as the game implemented a lot of new features. These features were showcased at the stand on "Åpen dag" and were very popular by attendants. The experience from "Åpen dag" was positive, the group members attending received much useful feedback that would help develop the game further. The group planned to continue adding more unit tests and fix bugs in the next sprint. The team also prepared to release an alpha version of the application on Google Play Store. For the external camera feature it was planned to show a preview of the camera on screen and later implement sonification so that it would function the same way as the internal camera. For the game, it was planned to finish implementing state saving and Image Viewer for the game. If there was enough time, it was planned to start working on a database to store the images used in the game.

7.8 Sprint 10 (week 17-18)

7.8.1 Planning

During this sprint, the group had reached an important milestone, feature deadline May 1st, where the group planned not to implement any new features into the application. All features that had been started before this would be completed as intended. The external camera team planned to restart the implementation of the external camera library, in hope to overcome error messages. It was crucial to complete it before the user test and the demonstration day. The customer gathered persons that were willing to perform the user test, that would occur next sprint. During the user tests, it was planned to evaluate the usability of the application. The group planned to spend the last week of the sprint to focus on the report. A deadline for the prefinal report delivery was set to the last day of the sprint and would be sent to the supervisor to receive feedback.

7.8.2 Results

The group decided to create a Slack channel for report related questions and suggestions. This was created to avoid spamming other Slack channels, and sending personal messages across group members. This way it was possible for everyone to see it and give their feedback.

External camera integration The feature was finally completed, and the library was implemented without any error messages. This meant the camera could connect within the application and work in the same way as the internal camera.

7.8.3 Reflections

The external camera integration proved to be more complicated than initially assumed, but was finally resolved during this sprint. The progress of this feature was greatly enhanced by getting help from group members working on other features.

7.8.4 Summary

The external camera integration was now working correctly, the library was implemented and was working the same way as the internal camera. The game had been polished for the demonstration day that would occur next sprint. The group also planned to perform the user test in the next sprint.

7.9 Sprint 11 (week 18-19)

7.9.1 Planning

During this sprint, the group would run user tests. The test would last approximately fifteen minutes per person, with a five-minute talk with feedback and experiences with the application. Each test person would receive a document with a short section of necessary information about the application, and twelve tasks that should be solved. A plan for the user test can be found in Appendix D.1. The group also planned to prepare for the presentation. The team manager was given the responsibility for it and would perform a one-minute 'sales pitch' trying to sell the product and later a five-minute presentation of the whole product. The presentation took place in Drivhuset at NTNU, and after the presentation, the group would maintain a stand. The group would also release an alpha version of the application on the Google Play Store before the presentations took place.

7.9.2 Results

In this sprint, the group prepared both user tests and the presentation. The user test consisted of twelve tasks; the tasks can be found in Appendix D.2. Each participant gave feedback and suggestions; a complete list can be found in Appendix D.3. Later in the sprint, the group released an alpha version of the application on Google Play Store [12], which was downloadable during the demonstration.

For the remaining of the sprint, time was spent on bug fixing and improving the application ahead of the demonstration. The game had state handling being implemented so that it behaved correctly when tilting the screen. Logic from the Image Viewer was

implemented as well, so the user can feel around with their finger to hear a sound and then make a guess of which image it is. Another game level was introduced during this sprint, instead of just simple colors the user could now guess what flags are hidden. This makes the game more challenging because there was now more than one color the user had to consider.

7.9.3 Reflections

The group was satisfied with the experience of the presentation and was always thrilled to show the application for those interested. The customer had to travel and therefore missed the presentation. The group wished the customer could attend the demonstration, to see how much people enjoyed the idea the customer had put so much passion into.

7.9.4 Summary

User tests were performed, and the group received much good feedback. The presentation and demonstration were a success, and there was much excitement around the event. The group also managed to release an alpha version of the application. The group planned in the next sprint to spend the remaining time on the project on finishing the report and any necessary documentation that was needed. There was also planned to have a meeting with the supervisor to receive final feedback on the report that was submitted to the supervisor the last sprint.

7.10 Sprint 12 (week 19-22)

7.10.1 Planning

This would be the last sprint for the project, and it lasted from demonstration day, May 12th, until the deadline of the report, May 30th. During this sprint, the main focus would be completing the report and finish writing other necessary documentation. The group arranged a meeting with the supervisor to receive feedback on the prefinal report that was submitted the last sprint. It was also planned to complete a system test of the application May 26th, mainly to identify any issues for future development. The plan for the system test can be found in Appendix E.1. The last objective the group would perform during this sprint was to proofread the whole report before the deadline.

7.10.2 Results

This sprint lasted a long time and resulted in much work being done. The group finished all of the remaining documentation that was needed. Refactoring of the code and bug fixing of the application was also done. For the documentation part, the group finished writing a user manual (see Appendix G), developer guide (see appendix H) and reflections of the whole project. The meeting with supervisor regarding feedback on the prefinal report yielded much good feedback that helped with the completion of the report. The group performed a system test, and the application met all requirements. The group did, however, find a few bugs, a list of the results can be found in Appendix E.2. The group also sent an acceptance test to the customer, where all the requirements were tested, the results can be found in Appendix F. Before deadline, the group met up to proofread the report to ensure that there were no spelling errors or missing sections and similar issues with the report, before finally delivering the final report.

7.10.3 Reflections

During this sprint, it was impossible to have regular meetings, due to group members having exams on different dates. However, the work was delegated in a good way, so each team member had something to complete before everyone met for the system test.

7.10.4 Summary

This sprint lasted longer than any previous sprints; the group managed to do everything that was on the plan. There were, fortunately, no problems reaching each deadline that was set. During this sprint, the group performed a system and acceptance test. All of the documentation including the report was completed.

8 Discussion and reflection

This section contains a final reflection of the project work. It starts out with some observations and thoughts regarding the project process, including project management, process methodology, and customer involvement. Then, the product quality and issues related to hardware and software are discussed. In the end, a selection of important aspects that have influenced the project work is presented.

8.1 Process reflection

8.1.1 Cooperation and communication within group

The group is satisfied with the communication and cooperation throughout the project. Communication was mainly done through Slack, which has proven to be a very handy tool with useful features. All group members have been available on Slack virtually anytime, which has significantly eased the cooperation.

There have been weekly meetings on Wednesdays and Fridays, which of course have been crucial for communication and collaboration. The meetings were generally efficient and well attended and were followed by a useful work session for as long as desired. There has been a favorable environment for creativity and discussing new ideas, which also has been necessary as the group has been forced to find solutions themselves. However, some meetings were too inefficient, in particular in the latter phase of the process. The meetings could in these cases have been led more strictly, forcing group members to keep to the point.

There have been quite significant differences in the level of experience within the group. This has to some degree been a limiting factor, but the team believes to have handled this in a good way. Experienced people have been eager to spend time teaching other group members, and less experienced people have been interested in learning. This learning process has happened through work sessions - some of them with pair programming, mandatory code reviews, and discussions on Slack. The difference in experience has never caused any bad atmosphere in the group.

The primary factor hindering cooperation was the group size. The group consisted of seven persons, which in particular has made it difficult to distribute work well. The work has been divided by splitting the application into a set of features, with some people being responsible for each feature, which generally has worked out well. Communication has also worked well despite the high number of group members, mainly due to Slack.

8.1.2 Cooperation with customer

Overall, the cooperation with the customer and product owner has been excellent. Contact was established immediately, and the first meeting was held only a few days after the projects had been assigned. The customer has made himself accessible throughout the project by email, phone and Slack, and has been very cooperative in general.

The most challenging part in regards to customer relations has been related to discovering the actual needs and requirements of the customer, as it has often not aligned with what is being expressed initially. This is a common problem in such projects, and knowledge and experience gained from previous courses have been very helpful in this regard.

8.1.3 Project management

In general, the project management has worked out very well. Most of the decisions related to management and roles had been decided upon before the project assignment, and everyone had a clear understanding of their roles once the project started.

Roles One of the most successful practices put to use was to delegate as much responsibility as possible among the team members, in order to give everyone a sense of ownership of the project or specific parts of it. This relieved the project manager of some work and let him focus on overall management and external contact, while also making sure the other team members were included in all decisions. This might not work as well in a professional setting where the team members might have specialized expertise but has worked out particularly well in an educational setting where the team is made up of students.

Team building Team building events could have been used more efficiently. All members of the team had a positive attitude towards team building events and suggestions were proposed, but none were officially organized by the team. One was arranged by the customer, where unfortunately not everyone could attend, but this was highly successful and great fun.

Project guidelines The early establishment of project guidelines, attached in Appendix B, worked out very well. It provided a common ground for all things related to the project, such as technical and theoretical practices to be used. This document was updated a few times throughout the project with new additions or changes as the practices changed and overall proved to be a very useful guidance throughout the project.

Team contract A team contract was also established early, which everyone signed. This is a good and recommended practice, even if it does not need to be used later

during the project, which usually is a good thing.

Milestones The management of project milestones could possibly have been better, but it was tough to make predictions regarding technical milestones at the early stages of the project. Some milestones were set early, such as the final delivery, mid-term delivery, presentation and so on, while others related to the stages of the system implementation were not. The lack of technical milestones has however not had any apparent drawbacks on the project, and some were added as the project evolved and the team gained a better understanding of the final product.

8.1.4 Process methodology

The customization and implementation of a Scrum-inspired method has worked out very well overall. The combination of sprint planning, sprint review and retrospective meetings on Wednesdays has been a success, and has helped the group keeping progress and resolving problems. Also, it has been convenient not having formal meetings too frequently. As desired by all agile methodologies, the customer has been highly involved in the process, and he has tested app prototypes in all stages of the process. The group has chosen to exclude certain Scrum artifacts like Planning Poker and daily scrum meetings, and was satisfied with this choice at the end of the semester. Some suggestions for improvement are described below.

Sprint length The initial sprint length was one week and later changed to two weeks. The sprint length should possibly have been two weeks from the beginning of the project, as that would force the team to be more accurate with sprint planning and estimations, and also reduce the number of items that had to be extended across several sprints. On the other hand, one-week sprints worked very well in that it allowed the team to be extremely agile in the rapid prototyping and discovery phases early in the project.

Sprint planning The sprint planning could have been more accurate. Most sprints had additional items added to the sprint backlog during the sprint, often due to not having enough work or changes in requirements or priorities.

8.1.5 Group development

The group has learned much and gained useful experience during the project. Many team members have learned much Android programming and new Git practices, but even more important, got experience with working in a large group in cooperation with a real customer. The group has got to know what it means that the clients may change requirements and priorities, and the importance of having a good team spirit and contributing to the project. An important lesson learned is the importance of planning.

This applies both to the initial phase, and during the project work. Also, cooperation regarding code review and pair programming have proved to be very useful.

8.2 Product reflection

8.2.1 Compliance with customer requirements

The acceptance test, described in section 6.7, shows that the customer was very satisfied with the final product. Also, during the project work, the customer has seemed pleased with the group's effort and progress. The requirements that the group and customer agreed on early in the process, described in section 4.3, have changed dramatically and have been of limited importance. It has been much more important for the customer that the group should be agile and adapt to the changing requirements. The team is satisfied with how this has been handled, which the customer also confirmed in the acceptance test.

8.2.2 Quality of product

The overall quality of the final product is considered to be high, especially considering the requirements, timeline, and goals of the project. The quality of the CLRF Navigator project is greater than the CLRF Beacons project, mainly because more time has been spent on the former, and the latter is a proof of concept, and the rest of this section is concerned about CLRF Navigator.

The system utilizes many of the current best practices for Android development, such as the separation of concerns through MVP, inversion of control and dependency injection with Dagger, and a coherent design with well-known and tested design patterns. The sonification layer has been made very modular and extensible, to allow rapid changes and tweaks, both through run-time settings and modification to the code.

Most of the public interfaces have been documented with JavaDoc as to make it easier for future developers to understand and utilize the interfaces created. Implementation specific documentation has mostly been avoided as such documentation quickly becomes out-of-date and misleading, instead focusing on writing self-documenting code that is easy to comprehend. The reasoning behind this is that if developers want to understand the implementation details of code, they would need to read and understand the implementation itself regardless, as there is no guarantee the documentation is accurate enough to be the single source of information, while the code itself is.

The application mostly follows the Material Design guidelines as specified by Google, but some activities still need some work and polish. Because it is time-consuming, it has been left until the activities have been properly prototyped and a decision has been made on how it is going to function and look. This means that the first activities that

were implemented, such as camera, image sonification and settings, follow the guidelines more closely than the more recent additions to the application such as the Color Map and Game features.

8.2.3 Tooling

The tooling overall worked out very well, with Android Studio providing a very polished developer experience for Android. There are however still some issues with build and deployment times, even though this has been improved massively over the last year. The introduction of Instant Run has reduced the build times dramatically, but the feature is unfortunately not completely reliable, and Gradle often needs to perform a full build regardless. This has been particularly prominent after the JNI implementation of the feature extractor was added, as this requires the NDK and several extra build steps using CMake, which increased build times considerably.

8.2.4 Hardware

There were some challenges related to hardware during the project. Some team members did not have access to devices that could run the application for the duration of the project, which likely was a limiting factor. Adding to that, some of the phones that could execute the application did not support external USB camera due to manufacturer modifications to the Android distribution, further limiting the number of available devices. This was further complicated by the fact that the external camera could not be tested with an emulator.

A group member had his laptop break during the project and had to use an old laptop for the remainder of the project, slowing down the developer's efficiency considerably.

8.2.5 Testing

Testing could have been done better, particularly the practice of writing unit tests, instrumentation tests, and regression tests. This comes mostly from a lack of experience with testing in general, and specifically testing on Android, which is generally considered harder to test relative to some other platforms by the industry[25]. But considering the goal of the project, which has been to provide a platform for rapid prototyping and research, and the generally accepted practice of focusing less on tests during the early phases of a prototyping project, the testing has been acceptable. However, given more time, tests should and would be a focus for further development.

User testing, system testing, acceptance testing and manual testing could possibly have been planned better in advance. However, tests were still performed adequately, effectively and had great value, and will have continued value for future developers and

system development.

8.3 Selected aspects

A selection of aspects which have been important for the group work are described in this section.

8.3.1 Divergent and convergent phases

The overall group process had similarities to figure 26, having separate phases dominated by divergence and convergence, respectively. The start of the process was dominated by new ideas, suggestions of features, and explorations of possibilities. Midways in the process, the group had an important meeting with the customer, discussing the current progress and which priorities should be made for the rest of the project. This resulted in a final requirements document and can be considered the peak in the figure. The last part of the project was focused on completing the features, and no more major changes to the system were planned. This initial phase of exploration has affected the process greatly, and has made the project differ significantly from projects where all requirements are known beforehand.

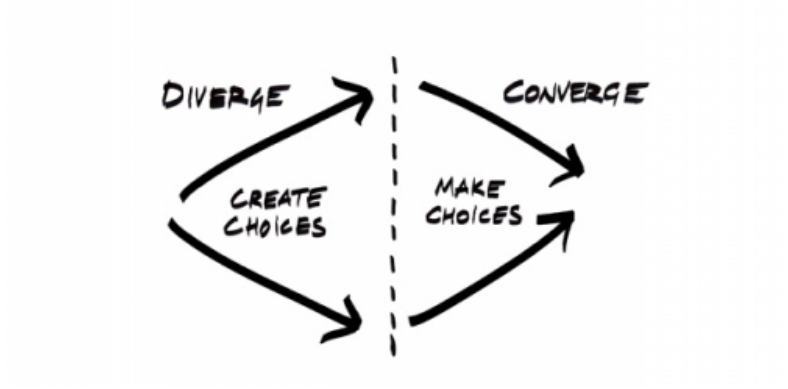


Figure 26: Phases of divergence and convergence, inspired by Design Thinking [10]

8.3.2 Participation in events

Group members have participated in several events together with the customer, presenting a prototype of the app. These include the Technoport conference [49], a conference for innovation and technology, where all group members attended and presented the current app prototype. Also, some team members attended "Åpen dag" at NTNU [36], where college students get information about NTNU. A product demonstration was also held as a part of the course, at May 11th. A 1-minute sales pitch and a 5-minutes' presentation took place, in addition to presenting and demonstrating the project on a

stand. All of these events have pushed the group to make runnable prototypes of the system.

8.3.3 GitHub and Waffle integration

Waffle has been used as the activity plan and has provided integration with GitHub and automated issue tracking. This has been critical for the structure of the work, and thus a separate section has been dedicated to the evaluation of it.

The group initially adopted a rather strict set of guidelines on the use of Waffle, as dictated by the project guidelines, see Appendix B. Having standardized branch names made the project very well organized, even though it ended up consisting of more than a hundred different branches. In a few cases, branch names were not according to the standard. This occurred rarely enough that it did not cause any problems. The fact that Waffle automates issue tracking is a major advantage. Issues may be directly coupled to branches, and they are moved to the appropriate column by correct referencing in commits and pull requests.

Pull requests have usually been submitted daily, which has resulted in a very transparent process. Because of this transparency, there have not been any notable cases of double work, where several people have worked on the same issue. Also, the customer has had access to the Waffle board, which has continuously provided him information about the state of the project.

Importantly, Waffle has allowed for easy addition of issues for new features and bugs as the development proceeded. This has been very helpful for having a flexible process.

A drawback is that many Waffle items may be difficult to understand for non-developers. Also, the Waffle board may be too detailed and specific if the aim is getting a proper overview of the project state. However, the advantages mentioned above are believed to be even more important.

By using Waffle, the group has learned a lot about how to organize large projects and split features into a set of smaller tasks. Also, as the team adapted this way of organizing tasks, it gradually became a more natural way of working. As a conclusion, the combination of using continuous integration and Waffle as task organizer worked very well.

8.4 Conclusion

This report describes the final project for a group of bachelor students of informatics at NTNU, which has been done in cooperation with Colorophone. The project team has made an Android-based system to help researchers develop a method to help blind people experience colors through sound. The app converts colors captured by a live

camera or an image to sound and provides customization of a range of parameters in the process. A range of features utilizing this principle have been implemented, including an Image Viewer and a Game. The group has followed a highly agile process, with some requirements changing dramatically. However, the team has learned a lot during the process.

A fully functional app prototype has been developed as desired by the customer. Also, there are excellent opportunities to develop the app further. The application has been developed in a modular way, making extensions convenient. Suggestions for future development, a development guide and a list of known issues are contained in this report for future developers. Both the group and the customer are satisfied with the work and considers the project a success.

9 References

[1] Amir Amedi's Lab. EyeMusic (version 1.2). Android application, 2015. URL https://play.google.com/store/apps/details?id=com.quickode.eyemusic.

[2] Android Developers. Bluetooth Low Energy, 2017. URL https://developer.android.com/guide/topics/connectivity/bluetooth-le.html.

[3] Android Developers, 2017. URL https://developer.android.com/distribute/best-practices/index.html.

[4] Android Developers. Building instrumented unit tests — android developers, 2017. URL https://developer.android.com/training/testing/unit-testing/instrumented-unit-tests.html.

[5] Android Developers. Best practices, 2017. URL https://developer.android.com/guide/practices/index.html.

[6] Android Developers. Renderscript, 2017. URL https://developer.android.com/guide/topics/renderscript/compute.html.

[7] Android Developers. Getting started — android developers, 2017. URL https://developer.android.com/training/index.html.

[8] Android Developers. Performance tips, 2017. URL https://developer.android.com/training/articles/perf-tips.html#ObjectCreation.

[9] W. Borowicz. How do beacons work? the physics of beacon tech, 2017. URL http://blog.estimote.com/post/106913675010/how-do-beacons-work-the-physics-of-beacon-tech.

[10] T. Brown. *Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation*. HarperCollins e-books, 2009. URL https://www.amazon.com/Change-Design-Transforms-Organizations-Innovation-ebook/dp/B002PEP4EG%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3DB002PEP4EG.

[11] L. Chen, M. A. Babar, and B. Nuseibeh. Characterizing architecturally significant requirements. *IEEE Software*, 30(2):38–45, 2013. doi: 10.1109/ms.2012.174.

[12] Colorophone. CLRF Research (version 1.0-alpha2). Android application, 2017. URL https://play.google.com/store/apps/details?id=com.colorophone.app.research.

[13] Comarch Technologies, 2017. URL http://www.beacon.comarch.com/.

[14] A. Devine. Wayfindr - sydney town hall trial - wayfindr, 2017. URL https://www.wayfindr.net/blog/wayfindr-sydney-town-hall-trial.

[15] dtg-android. Spectral - images into sound (version 1.0). Android application, 2009. URL https://play.google.com/store/apps/details?id=uk.ac.cam.cl.dtg.android.audionetworking.spectral.

[16] FlexibeatzII. Paint2Sound. Windows desktop program. URL http://flexibeatz.weebly.com/paint2sound.html.

[17] M. Fowler. Continuous integration, 2017. URL https://www.martinfowler.com/articles/continuousIntegration.html.

[18] Futurice, 2017. URL https://github.com/futurice/android-best-practices.

[19] J. Gerrish and C. Williams. Robolectric 3.3 and roadmap — robolectric, 2017. URL http://robolectric.org/blog/2017/03/02/robolectric-3-3-and-roadmap/.

[20] Google. Android navigation, 2017. URL https://developer.android.com/design/patterns/navigation.html.

[21] Google Inc., 2017. URL https://material.io/guidelines/.

[22] Google Inc. Android 4.3 apis — android developers, 2017. URL https://developer.android.com/about/versions/android-4.3.html.

[23] Google Inc., 2017. URL https://developer.android.com/guide/topics/ui/notifiers/notifications.html.

[24] Google Inc., 2017. URL https://developer.android.com/reference/android/speech/tts/TextToSpeech.html.

[25] C. Greb. Two reasons testing is hard on android, 2016. URL https://medium.com/android-testing-daily/two-reasons-testing-is-hard-on-android-78e4afeb0cba.

[26] G. Hamilton-Fletcher and J. Ward. Representing colour through hearing and touch in sensory substitution devices. *Multisensory Research*, 26(6):503–532, 2013. doi: 10.1163/22134808-00002434.

[27] J. Hildenbrand. What is google talkback?, 2014. URL http://www.androidcentral.com/what-google-talk-back.

[28] S. Judd. bumptech/glide: An image loading and caching library for Android focused on smooth scrolling, 2017. URL https://github.com/bumptech/glide.

[29] F. Marcellin. smartphones, bluetooth beacons: The pairing that could help the blind catch the right bus — zdnet, 2017. URL http://www.zdnet.com/article/smartphones-bluetooth-beacons-the-pairing-that-could-help-the-blind-catch-the-right-

[30] R. C. Martin. *Clean code*. Prentice Hall, 1 edition, 2012.

[31] P. Meijer. An experimental system for auditory image representations. *IEEE Transactions on Biomedical Engineering*, 39(2):112–121, 1992. doi: 10.1109/10.121642. URL https://doi.org/10.1109%2F10.121642.

[32] P. Meijer. The vOICe for Android (version 2.0.5). Android application, 2017. URL https://play.google.com/store/apps/details?id=vOICe.vOICe.

[33] millsustwo. Wayfindr – how it works (ustwo + rlsb), 2017. URL https://www.youtube.com/watch?v=mc3KmbfxuUQ.

[34] Nordic Semiconductor. Android-nrf-beacon - github, 2017. URL https://github.com/NordicSemiconductor/Android-nRF-Beacon.

[35] Nordic Semiconductor. nrf52 dk, 2017. URL https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52-DK.

[36] NTNU. Åpen dag ntnu i trondheim, 2017. URL https://www.ntnu.no/moetntnu/apendag.

[37] D. Osinski and D. R. Hjelme. A real time sensory substitution device inspired by the human visual system. Unpublished article, 2017.

[38] D. Pascolini and S. P. Mariotti. Global estimates of visual impairment: 2010. *British Journal of Ophthalmology*, 96(5):614–618, 2011. doi: 10.1136/bjophthalmol-2011-300539.

[39] K. Payne. How wayfindr guided my first steps to independence on the tube - wayfindr, 2017. URL https://www.wayfindr.net/blog/wayfindr-guided-independence-tube.

[40] Photosounder. Photosounder. Desktop program. URL http://photosounder.com/.

[41] Proto.io. Proto.io - prototypes that feel real, 2017. URL https://proto.io/.

[42] Radius Networks, 2014. URL https://github.com/AltBeacon/android-beacon-library.

[43] Radius Networks. Android beacon library, 2017. URL https://altbeacon.github.io/android-beacon-library/index.html.

[44] P.-Y. Ricau. Leakcanary: Detect all memory leaks! – square corner blog – medium, 2017. URL https://medium.com/square-corner-blog/leakcanary-detect-all-memory-leaks-875ff8360745.

[45] saki4510t. UVCCamera, 2017. URL https://github.com/saki4510t/UVCCamera.

[46] Skytopia. SonicPhoto. Windows desktop program, 2013. URL http://www.skytopia.com/software/sonicphoto/.

[47] I. Sommerville. *Software Engineering*. Pearson, 2011. URL https://www.amazon.com/Software-Engineering-Ian-Sommerville-ebook/dp/B008VIMCB8%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3DB008VIMCB8.

[48] Square, Inc. Dagger, 2017. URL http://square.github.io/dagger/.

[49] Technoport, 2017. URL http://conference.technoport.no/.

[50] Toggl OÜ, 2017. URL https://toggl.com/.

[51] Travis CI, 2017. URL https://travis-ci.org/.

[52] J. B. Troy. Visual prostheses: Technological and socioeconomic challenges. *Engineering*, 1(3):288–291, 2015. doi: 10.15302/j-eng-2015080.

[53] uml-diagrams.org, 2017. URL http://www.uml-diagrams.org/examples/deployment-example-android.png.

[54] VersionOne, 2017. URL https://www.versionone.com/agile-101/agile-development-success/.

[55] VersionOne, 2017. URL https://www.versionone.com/agile-101/agile-management-practices/agile-feature-estimation/.

[56] VersionOne, 2017. URL https://www.versionone.com/agile-101/agile-management-practices/.

[57] VersionOne, 2017. URL https://www.versionone.com/agile-101/agile-software-programming-best-practices/.

[58] VersionOne, 2017. URL https://www.versionone.com/agile-101/agile-software-programming-best-practices/refactoring/.

[59] VersionOne, 2017. URL https://www.versionone.com/agile-101/agile-software-programming-best-practices/coding-standard/.

[60] VersionOne, 2017. URL https://www.versionone.com/agile-101/agile-software-programming-best-practices/pair-programming/.

[61] VersionOne, 2017. URL https://www.versionone.com/agile-101/agile-software-programming-best-practices/simple-design/.

[62] L. Vogel. Using Butterknife for view injection in Android - Tutorial, 2016. URL http://www.vogella.com/tutorials/AndroidButterknife/article.html.

[63] L. Vogel. Using dependency injection in java - introduction - tutorial, 2016. URL http://www.vogella.com/tutorials/DependencyInjection/article.html.

[64] L. Vogel. Unit testing with junit - tutorial, 2016. URL http://www.vogella.com/tutorials/JUnit/article.html.

[65] L. Vogel and F. Pfaff. Unit tests with mockito - tutorial, 2016. URL http://www.vogella.com/tutorials/Mockito/article.html.

Appendix A Team contract

NTNU – Trondheim
Norwegian University of
Science and Technology

IT2901 - Informatics Project II

Colorophone Team 11 Contract

Lars Kristian Dahl, Daniel Martin Lundeby, Philipp Zirpins, Daniel Eriksen, Emil Orvik Kollstrøm,Henry Skorpe Sjøen, Espen Sivertsen

May 7, 2017

1 Commitments

1. Only agree to do work that we are qualified and capable of doing.
2. Be honest and realistic in planning and reporting project scope, schedules, tasks and time commitments and time spent on the project.
3. Operate in a proactive manner, anticipating potential problems and working to prevent them before they happen.
4. Promptly notify our customer of any change that could affect them.
5. Keep other team members informed.
6. Keep proprietary information about our customer in strict confidence.
7. Focus on what is best for the project as a whole.
8. See the project through to completion.
9. Note and report hours spent on the project to the team and the customer.
10. Allot an average of 20 hours per week to the project when required.

2 Communication

1. Keep issues that arise in meetings in confidence within the team unless otherwise indicated.
2. Be clear and to the point.
3. Keep discussions on track.
4. Be honest and open during meetings.
5. Encourage a diversity of opinions on all topics.
6. Give everyone the opportunity for equal participation.
7. Be open to new approaches and listen to new ideas.
8. Avoid placing blame when things go wrong. Instead, we will discuss the process and explore how it can be improved.

3 Decision making

1. Make decisions based on data whenever feasible.
2. Seek to find the needed information or data.
3. Discuss criteria for making an important decision before choosing an option.
4. Get input from the entire team before important decisions are made.
5. Ask all team members if they can support a decision before important decisions are made.
6. Simple decisions can be made by individuals when needed, based on the information they have available.
7. Accept individual decisions made by others for simple problems, including when they are not optimal.

4 Meeting procedures

1. Meetings will begin and end on time.
2. Team members will come to the meetings prepared.
3. Agenda items for the next meeting will be discussed at the end of each meeting.
4. If a team member can not attend a meeting, he/she will notify the team in advance, and provide a written note of his/her opinions on the agenda items.
5. Unresolved issues will be added to the issues list.
6. A summary should be written and published for every meeting.

5 Conflict

1. Regard conflict as normal and as an opportunity for growth.
2. Choose an appropriate time and place to discuss and explore the conflict.
3. Listen openly to other points of view.

4. State our points of view and our interests in a non-judgemental and non-attacking manner.

5. Seek to find some common ground for agreement

6 Signatures

Team Member	Date
Team Member	Date
Team Member	Date
Team Member	Date
Team Member	Date
Team Member	Date
Team Member	Date

Appendix B Project guidelines

Colorophone - Project Guidelines

1 Team Roles

Project manager, scrum master: Lars-Kristian Dahl
Team manager, meeting leader: Henry Skorpe
Report manager: Daniel Lundeby

2 Workflow

1. Sprint Meeting every Wednesday, at 08:15.
2. Status Meeting every Friday, at 12:15
3. Sub-teams manage themselves.
4. Pair-programming is recommended.
5. All decisions that affect the team are to be made between 08:00 and 17:00.

3 Engineering and design

1. Use known design patterns where appropriate.
2. Follow the SOLID principles for class design.
3. Follow the RCC and ASS principles for packaging and name-spaces.
4. Favor composition over inheritance.
5. Favor polymorphism over control structures.

4 Continuous Integration

- Travis CI is used for continuous integration.
- Builds are triggered by pull requests and merges to master or stable.
- If master fails to build, the error should be fixed as soon as possible.

5 Tools

Development:

- JVM based languages: Android Studio
- Version control system (VCS): Git
- Hosting of VCS: GitHub
- Continuous Integration: Travis CI
- Build tool: Gradle

Project management:

- Communication: Slack
- Sprint backlog and planning: Waffle.io, Github
- Storage of project-related files: Google Drive
- Informal documents: Google Docs
- Formal documents: ShareLatex
- Time tracking: Toggl.com
- Event and meeting planning: Google Calendar

6 Time tracking

Main activities:

- Development, sprint-related work
- Research, general domain research
- Organization, project organization, report, meetings, misc

7 Development Methodology

The project will use a simplified version of Scrum.

1. Sprint duration: Two weeks
2. Sprint planning, review and retrospective meetings: Wednesday 08:15 - 10:00.
3. Status meetings: Friday 12:15 - 13:00
4. Sprint backlog maintained on waffle.io
5. Task scores higher than 13 should be divided into smaller tasks

8 Version Control System

Git with a Merge based Feature Branch Workflow.

1. Branch 'master' is considered a special public branch.
2. Branch 'master' should always be in a consistent, stable state.
3. All development should take place on separate private branches.
4. Sprint-related branches should be named using the story ID, branch type and a descriptive title. eg. 313-feature/simple-ui
5. Other branches should be named with type misc and a name, eg. misc/changed-config
6. Do not push directly to master, use pull requests.
7. Pull requests should be reviewed by at least 1 person before merging.
8. Use interactive rebase to clean the branch before pushing. (git rebase -i HEAD~x)
9. Branches should be cleaned before issuing a pull request, usually to 1-3 commits.
10. Never use 'rebase' on public commits. ("Golden Rule of Rebasing").
11. Pull requests for bugfixes should consist of exactly one commit, describing the fix.

9 Github.com and Waffle.io integration

When starting work on a task either:

- Publish a branch that refers to the task by id in it's name
- Manually assign yourself and move the task to "In Progress" on Waffle.

Branch names should be on the form 'issueId'-'branchType'/branch-name, where:

- 'issueId' refers to a Waffle/Github issue WITHOUT leading #
- branch-name is a short, dash-separated, descriptive name of the branch
- 'branchType' is one of the following:
    - feat: a new feature
    - fix: a bug fix
    - refactor: refactoring production code
    - misc: updating build tasks, package manager configs, etc; no production code change
    - test: adding tests, refactoring test; no production code change

- style: formatting, missing semi colons, etc; no code change
- docs: changes to documentation

Example: 3-feature/my-feature

Pull requests must refer to related issues in the PR title or body by
- including #id in title
- including fix, close or resolve #id in the body
- connect #id in the body will relate the PR to the git issue without closing it

Single commits should not reference issues.

10 Design guidelines

https://material.io/guidelines/

11 Programming guidelines

Always leave the checked in code cleaner than it was checked out!

11.1 Code style for Android

https://github.com/ribot/android-guidelines/blob/master/project_and_code_guidelines.md

11.2 Code style for C/C++

https://google.github.io/styleguide/cppguide.html

11.3 Functions and methods

1. Clear and single purpose.
2. Short, rarely over 20 lines, hardly ever over 100 lines.
3. Convey intent with a concise name.
4. Avoid nesting control structures if possible.
5. Minimize the number of arguments.
6. Prefer pure functions.
7. Avoid output arguments.

11.4 Naming

- Give intuitive names to methods and variables.
- Use verbs for function names and nouns for classes and attributes.
- Use pronounceable names.
- Use solution domain names.
- Use one word per concept, and be consistent.
- Don’t be cute or clever.
- Use name-space instead of prefixing names.

11.5 Comments

- Prefer expressive code over comments.
- Don’t comment bad code, rewrite it.

- If code is readable, you don’t need comments.
- Explain your intentions in comments.
- Warn of consequences in comments.
- Emphasize important points in comments.
- Temporary TODO comments are acceptable, but prefer using the VCS.
- Any comments other than the above should be avoided.
- Never leave commented code.

11.6 Be wary of the following:

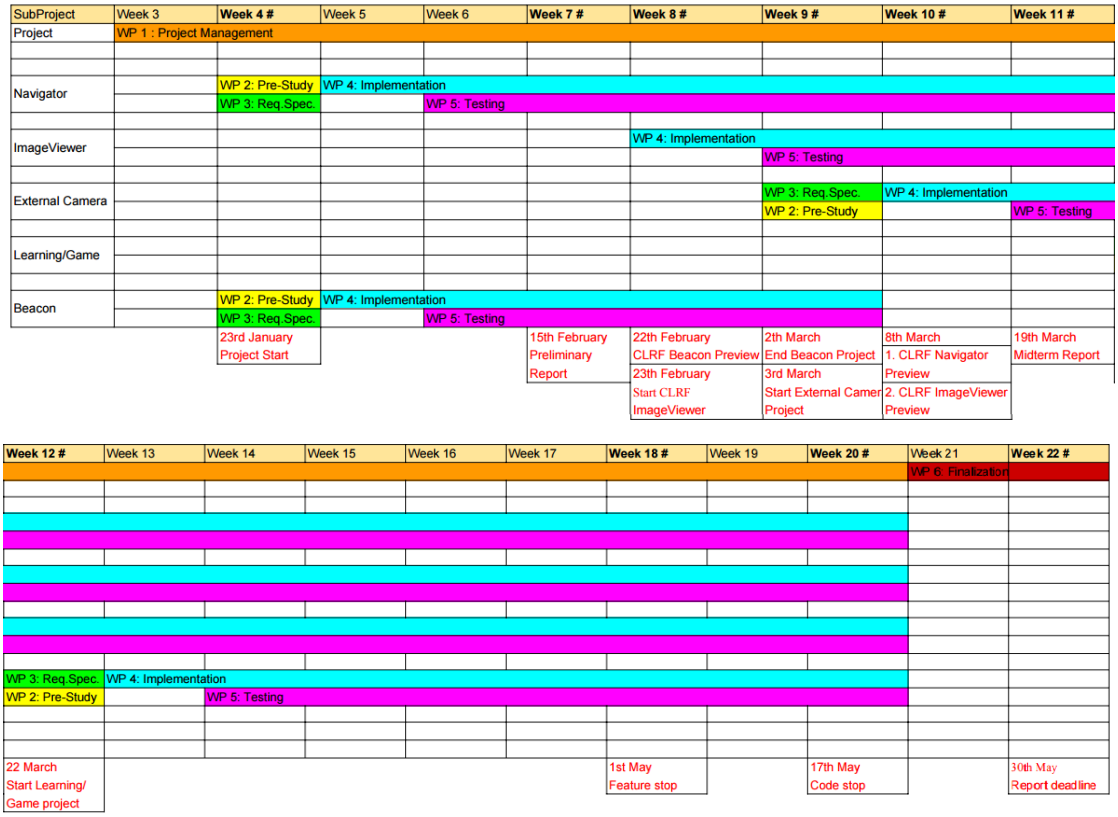
- Dead or commented code
- Large classes
- Speculative generality
- Framework code modifications
- Circular dependencies
- Circular references
- Sequentual coupling
- Deep call hierarchies
- Global variables and state
- God objects
- Magic numbers
- Long if conditions
- Deep inheritance hierarchies
- Hard-coded values

References

[1] http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod.
[2] http://www.planetgeek.ch/wp-content/uploads/2013/06/Clean-Code-V2.1.pdf.
[3] Clean, high quality code:
a guide on how to become a better program-
mer.
https://www.butterfly.com.au/blog/website-development/clean-high-quality-
code-a-guide-on-how-to-become-a-better-programmer.
[4] Atlassian. Feature branch workflow. https://www.atlassian.com/git/tutorials/comparing-
workflowsfeature-branch-workflow. Accessed: 2017-01-25.
[5] R.C. Martin.
Clean Code: A Handbook of Agile Software Craftsmanship
. Robert C.
Martin series. Prentice Hall, 2009.

Appendix C Project management attachments

C.1 Gantt diagram



C.2 Status report example

Group 11
Project Colorophone

Summary Status Report
17.03.2017

1 Introduction

[Any general information that can be useful]
Colorophone had a stand at Technoport 2017. We held the stand with the customer, divided in two groups, from 0800 until 1400. The stand had a lot of traffic and we showcased our work and android app to several interested parties.
Since Technoport, we have shifted our focus to writing the project report.

2 Progress summary

CLRF Navigator

Major improvements to settings UI
- disallow invalid values
- disable irrelevant setting dependencies
- display toast notifications for invalid values
- add new settings for algorithms, pre-recorded samples

Improve info panel in camera mode
- Display input RGB values and visible color
- Display output RYGBW values
- Display number of frames processed per second
- Add settings button to open settings

Add support for pre-recorded audio samples
Bug fixes

[Important: the activity plan for the two-week period, with the actual work performed by your team members and the actual completion of the activities, must be related to a milestone plan.]

3 Open / closed problems

Navigator

Need a 16-bit PCM sample at 44100 to continue development of pre-recorded sounds.

[What problems have caused delays? What have been solved and what are still open?]

External camera integration

- Uvcvideo driver missing from some vendor-specific Android kernels. Couldn't develop with the smartphones that the people in our group had. Solved by Henry joining our group since he has a phone with that driver.

4 Planned work for next period

[Detailed activity plan for the coming period]
(Refer to waffle.io for a detailed plan. Here are the main points)

Navigator

Continue working on support for pre-recorded audio
Bugfixes
Start on tutorial subproject.

Implement external camera integration.
Unit tests

5 Updated risks analysis

[Risk list; see RiskList_guidelines. Comment on changes to the list that have influenced your activity planning]

Customer is going away for a two week period. From Saturday 18. March to 1. April. We have planned what to do in the period. We don't know how accessible he will be, but he will try to answer important messages if needed.

C.3 Example session of manual testing

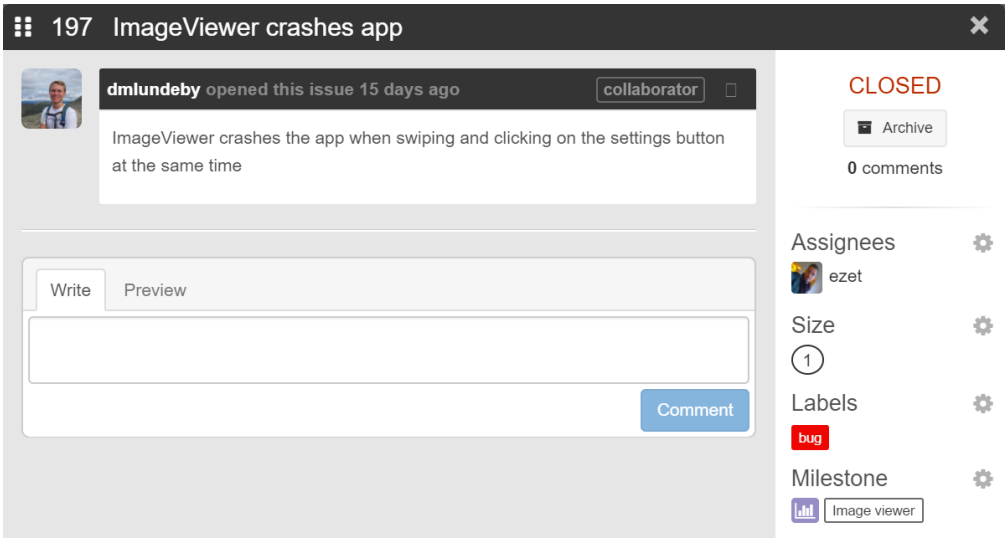


Figure 27: Bug report in Waffle.io. Note that this issue is in a closed state, as it has been resolved. The size was determined after the bug was resolved.

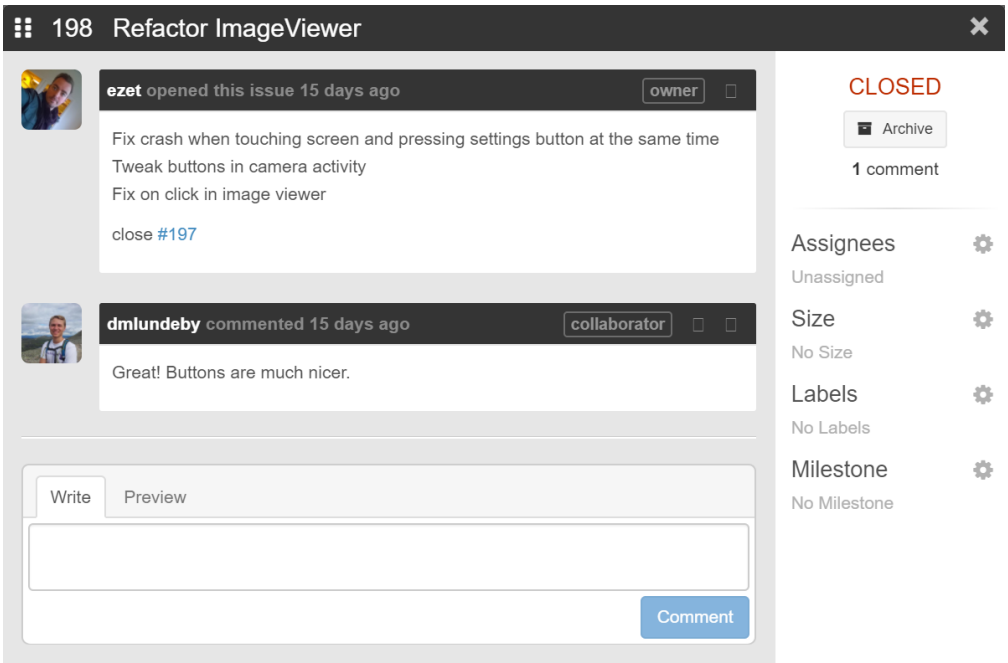


Figure 28: Pull request in Waffle.io, resolving the bug from figure 27

Appendix D User testing

D.1 User testing - Plan

User testing - Plan

Purpose

The purpose of the user tests is getting feedback on the usability of the researcher app, in addition to discovering bugs.

Link to Doodle

http://doodle.com/poll/gcresv8ebz58q3d4

Time and place

5/5-17, 10:00-16:00, at Kalvskinnet.

Schedule

- Present the test and what is to be done. (1 min)
- Give a brief introduction to the system (1 min)
- Do the test (~15 min)
- General feedback (~5 min)

Participants

There will be about 5 participants. Our customer will provide relevant participants who have some background in the Colorophone project. The participants are assumed to have some knowledge on the sonification algorithm and background knowledge on the domain, or will otherwise be given a brief introduction.

Environment

The tests will be conducted in a closed and silent room at Kalvskinnet, with minimal distractions.

Necessary equipment

- Android phone and charger. The phone should have the researcher app installed, as well as screen recording software. Also a reserve phone.
- Headphones
- External camera and USB OTG cable
- Printed forms for the user and optionally for the referee.

Git SHA at time of user testing: f7c92772ca7ffc595abd70ed9f801703cb1c3b4c

System introduction

Given at the “User Testing - Tasks“ document.

Roles

- Test leader: Presents the test, the system and tasks, and answers questions.
- Referee: Responsible for camera and taking notes.
- Participant

Performing the test

- The participant should try to solve the tasks described in the document “User testing - Tasks” by himself. The tasks are based on the requirements document.
- The test should be recorded by screen recording software, if the participant permits.

Guidelines:

- In order to have an objective comparison, the users should not test anything on the app before trying to solve the tasks.
- The participant should try to solve the task entirely by himself. Do not provide any help except for clarification.
- We should encourage the participants to speak loud what he/she is thinking.
- We should point out that we are testing the app, not the participant.
- The user should feel free to go to the next task if stuck on the current.
- Inform the participant that we will take some notes during the test.

Evaluation

- We should document basic user information, such as background (student, PhD, worker, etc.) and whether he is used to Android.
- During the test, note any problems or misunderstandings. Try to find the cause of problems. In particular, pay attention to:
    - Whether each module is intuitive.
    - Click behavior on Image Viewer - whether it's intuitive.
- After the test, save time for general feedback, Then, have a great focus on what the participant thinks. Try to keep an open dialogue.

Analysis

We will go through the videos and evaluation forms. We will focus on find causes of any problems, and propose improvements. A summary of the observations are written in the document "User testing - Observation", while the results and analysis are written in the report.

As the test is done on few people, we focus on qualitative rather than quantitative data. Each test has been done thoroughly, and the participants have had a good opportunity of giving high quality feedback.

Pilot test

A pilot test should be done before the actual test, to ensure everything works as planned.

An important result from the pilot test was a decision to use screen recording software instead of capturing use video camera.

D.2 User testing - Tasks

Colorophone Research App: User testing

The app converts colors captured by a camera or image into sound. Pixels are extracted from a square of the image known as a “feature extractor”. The red, green and blue channels are used to encode audio channels with different sounds and frequency.

The app has three modules: **Navigate**: Uses a live camera. **Image viewer**: Sound is played when swiping an image. **Learning mode**: Tutorial and game for recognizing sound colors. In addition, there is a **settings** menu for tuning parameters.

Try to do the tasks below as best as you can.

Playing sounds

1	Start playing some sound from the mobile camera.
2	Use the camera, and compare the frames per second (FPS) of the table and the wall.
3	Use the Image viewer to pick an image. Swipe the screen, and note the green output value.

Settings

4	Change the size of the feature extractor to 40 x 40.
5	Disable the red channel.
6	Lower the gain of the alpha channel.
7	Change the app to only output sine wave sounds.
8	Change the sonification algorithm to “Advanced RYGB”.
9	Reset all settings to default.

External camera

10	Connect an external camera, and play some sound from it.

Learning

Now, open the “Learning” module of the app.

11	Play the sound of a purple color.
12	Start a game on medium mode, and try one question.

D.3 User testing - Observations

User testing - Observations

This document contains a summary of important observations from the user tests, sorted on question. In addition, there is a summary of the general feedback. The observations are listed as points prefixed by “+” if something is positive,, or “-” if it is counterintuitive or may be improved.

1) Start playing some sound from the mobile camera.

- + Easy to locate sound toggle button.
- - The sound should be turned on by default.
- - The camera view doesn’t rotate.

2) Use the camera, and compare the frames per second (FPS) of the table and the wall

- + Most people found the FPS value quickly and were able to compare the values.

3) Use the Image viewer to pick an image. Swipe the screen, and note the green output value.

- - Many started trying to click the image to hear sounds, instead of continuously swiping (might be an idea with a toast message telling to swipe).
- + Generally, it doesn’t take long time to understand that swiping is the correct way of using the feature. Once discovered, it is very intuitive to swipe the image.
- + Easy to find output values when listening.
- - The difference between “in” and “out” wasn’t clear to everyone.
- - Name on “View Image” doesn’t indicate that it is possible to get any sound out of it.

4) Change the size of the feature extractor to 40 x 40.

- + Easy to locate in settings.
- - Name is confusing; whether the size of camera changes or the square in the middle.

5) Disable the red channel.

- - Many think it exists in “general” settings under the “algorithms” category. In general, many look for color-related settings are under the “general” category instead of “audio”.
- + Makes sense that it is in audio settings.

6) Lower the gain of the alpha channel.

- + Easy to find after learning locating the channel settings.
- - Difficult to understand what it means

7) Change the app to only output sine wave sounds.

- - When changing sound, it starts at the currently chosen list element and not at the top. One person didn’t realize it was possible to scroll upwards.
- - Not so easy to locate where to change sounds.
- - Must change sound for each color individually. Might be an idea to have presets of sounds.

8) Change the sonification algorithm to “Advanced RYGB”.

- + Most people found this option quickly, after getting known to the menu system.

9 Reset all settings to default.

- - App gives no confirmation message when resetting. Resetting did once happen unintentionally.
- + Easy to locate

10) Connect an external camera, and play some sound from it.

- + Transition to external camera is smooth and intuitive. No problems encountered.

11) Play the sound of a purple color.

- - Sound plays only once on button click. Many would have it play as long as the button is pressed.
- + Good place to practice and learn sounds.
- + Intuitive name

12) Start a game on medium mode, and try one question.

- - Difficult to find the game, when there are so many different words in the activity. Learning, tutorial, practice. Think of renaming to game.
- + Most people managed to get a correct answer on the question.
- + Works like image viewer

General feedback

- - The start screen is white. That makes people think that something is loading. Almost everyone requested some content here.
- - Hard to find the Camera activity, called “Navigator”. Consider renaming.
- + Drawer-menu at start-screen is easy to navigate through.
- + Instrument sounds makes it more comfortable to listen to.
- - On navigating back from the Image Viewer, one person noted that it would be more intuitive to get to the image chooser menu.
- - Text input boxes sometimes have Norwegian labels like “OK”, “Avbryt” and “Utført” - is that caused by the app or the OS?

Other notes

- + In general, the app appears to be very stable. It crashed once during the tests (with about 90 minutes of running the app), as described below.
- - A strange bug was discovered, may be reproduced this way: (1) Shut down the app. (2) Open Image Viewer, and choose an image from a folder that is not the camera folder. (3) Return to the main menu, and open Image Viewer with the same image as in (2). This causes the app to crash.
- - Choosing image in image viewer in landscape mode is sometimes loading slowly. That could also be caused by the recording software running in background.
- - On navigating from camera to settings, and then using back button, one ends up in the main menu instead of the camera.

Appendix E System testing

E.1 System testing - Plan

Colorophone Research App: System testing Plan

Purpose

The purpose is testing whether the system behaves as expected. The app should (1) behave according to the customer requirements, and (2) shouldn't contain bugs or erroneous information.

Execution

- During the test, all developers should do the tasks as specified below, and note any comments. They do not need to do the tasks at the same time.
- There should be an open environment for discussion during the tests.
- The system test should be done before the app is sent for acceptance testing to the customer.

Evaluation

A summary of the results will be a document showing what needs to be done for the system to behave as expected. Suggestions for improvement will be added to the Waffle.io issue tracker system. If time, improvements will actually be implemented.

Tasks

- Go through all requirements: functional, non-functional and optional. Note any comments related any of the requirements.
- Test each single feature as listed below. Try to note any error or unexpected behavior.

Features to be tested:

- Start page and drawer menu
- Camera
- Image Viewer

- Game
- Color map
- Settings

E.2 System testing - Results

Colorophone Research App: System testing
Summary of results

Requirements

Functional requirements

ID	Comment
FR-NAV.02	Changing frequency on any channel is inconvenient. When trying to change number, it defaults to 100 Hz or 2000 Hz, which are the limits.

Non-functional requirements

Nothing detected

Optional requirements

Nothing detected

Features

Start page and drawer menu

Nothing detected

Camera

- Sometimes after having used the app for a long time, the sound disappears. The app needs to be restarted to work again.

Image Viewer

Nothing detected

Game

- For some screen resolutions, text on answer options cannot be seen in landscape mode.
- It is possible to press behind the confirmation box.

Color map

Nothing detected.

Settings

Nothing detected.

Other comments

Nothing detected.

Appendix F Acceptance testing

Colorophone Research App: Acceptance testing

Functional requirements

ID	Description	Accepted (yes/no)	Comment (optional)
FR-NAV.01	Change the sound of each channel individually. Required sounds: Sine waves, instruments and white noise. Optional sounds: Square waves, triangle waves and variations of white noise.	yes	Variations of white noise: more specifically low pass filtered white noise.
FR-NAV.02	Change the audio signal frequency for each output channel individually for generated sounds	yes	
FR-NAV.03	Disable each channel individually	yes	
FR-NAV.04	See algorithm input RGB values	yes	
FR-NAV.05	See algorithm output RYGBW values	yes	
FR-NAV.06	See number of frames processed per second	yes	
FR-NAV.07	Adjust the volume of each channel individually	yes	
FR-NAV.08	Change between algorithm implementations	yes	
FR-NAV.09	Change parameters related to noise generation	yes	
FR-NAV.10	Change the size and shape of feature extraction	yes	

Quality Attributes (Non-functional requirements)

Performance

ID	Description	Accepted (yes/no)	Comment (optional)
NFR-NAV.01	The system should be able to process 15 frames per second in good light conditions	yes	

Power efficiency

ID	Description	Accepted (yes/no)	Comment (optional)
NFR-NAV.02	Sonification should be operable for at least 2 hours while preview is enabled	yes	

Usability

ID	Description	Accepted (yes/no)	Comment (optional)
NFR-NAV.05	The application should be usable on all phones with Android API 17 or higher	yes	
NFR-NAV.06	The application should select any connected external camera as the default input, provided that the external camera is supported by the device.	yes	

Modifiability

ID	Description	Accepted (yes/no)	Comment (optional)
NFR-NAV.10	Researchers should be able to change any parameter related to the camera input, sonification algorithm or audio output within 30 seconds	yes	

Optional features

ID	Description	Accepted (yes/no)	Comment (optional)
OPR-NAV.01	Be able to choose an image from the phone and swipe to sonificate it.	yes	
OPR-NAV.02	The app should have a game, challenging the user to guess a hidden image based on the sound from swiping.	yes	

OPR-NAV.03	The app should have a function teaching the user the mapping between color and sound.	yes	
OPR-NAV.04	The feature extractor should automatically scale to a reasonable size.	yes	Does it mean a default value?
OPR-NAV.05	Be able to change individual channel volume, change filter of white noise, change the value of Stevens Power Law and change the sample rate of the sound.	yes	
OPR-NAV.06	Be able to see which color is extracted from the camera in the navigator.	yes	

General feedback

The project team has performed a magnificent job. During the project period, some requirements had been changed dramatically, which the group has handled in a very professional way. The project team delivered a fully functional app prototype as planned.

Appendix G User manual

User Manual

Colorophone Research App

Introduction

This user manual shows how to use the Colorophone Researcher App. The researcher version permit tuning of a range of parameters, and consists of several modules:

1) **Camera**: Uses a live camera as input source.
2) **View Image**: Uses an image stored on the phone as input source. Sound is played when swiping the image.
3) **Play**: Play a game to practice recognizing colors. Swipe a black screen and hear what it sounds like. Guess what image it is.
4) **Color Map**: Tutorial for recognizing which sound corresponds to which color.

In addition, there is a **settings** menu for tuning parameters.

Installation

The app can be downloaded as an alpha version from the following link on Google Play: https://play.google.com/store/apps/details?id=com.colorophone.app.research

Alternatively, just search Google Play store for “CLRF Research”.

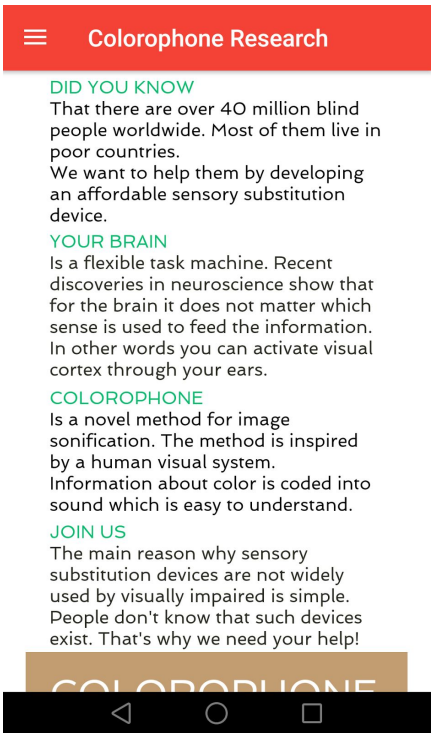
The app requires a device running at least Android 4.2 (API level 17).

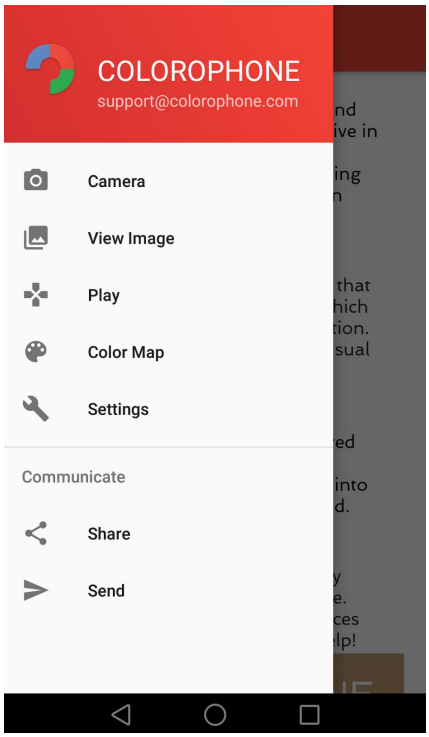
Working principle

The app works by the principle of color sonification, that is, the conversion of colors into sound. The app uses a live camera or an image as an input source. Pixels are extracted from a square of the image known as a “feature extractor”. The red, green and blue channels are used to encode audio channels with different sounds and frequency. By learning to recognize which sounds correspond to which color, it is possible to “hear” colors. The ultimate goal is making a complete sensory substitution device based on this principle.

Start screen

The start screen consists of a web view that displays the customer's product website: https://www.colorophone.com. The site contains information about the Colorophone project. The developers of the Colorophone Researcher Android app has not participated in developing or designing the website displayed.





The screen also has a navigation bar where a menu can be accessed by pressing the hamburger menu icon. From this menu, all of the different modules can be navigated to.

Camera

The camera module uses a live camera as an input source and outputs sound from the square in the middle of the image. A range of debugging information is shown on the sidebar, and some buttons are available.

The sidebar can be toggled on and off by pressing anywhere on the screen.

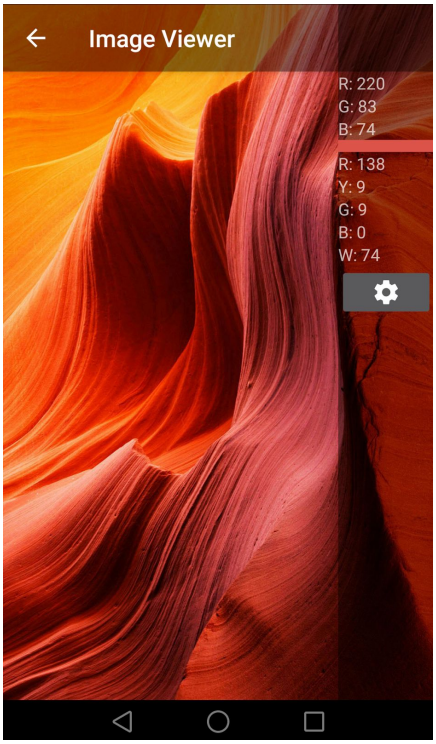
The following buttons are available:

- **(1) Toggle sound**. Sound is turned on or off.
- **(2) Switch between internal and external camera**. Only available if an external camera is connected.
- **(3) Go to settings menu**. Shortcut to the settings menu.

The following debug information is shown at the screen:

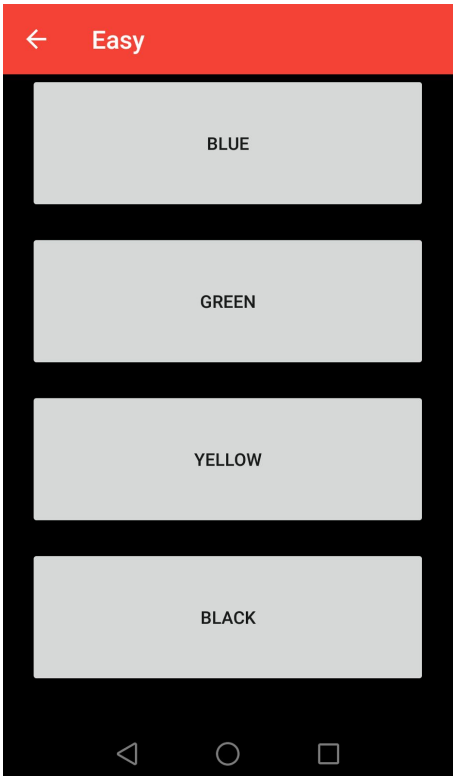
- FPS: Frames per second that the app is available of processing.
- R, G, B: Input values of the red, green and blue channels, ranging from 0 to 255.
- R, Y, G, B, W: Output values of the red, green, blue and white channels, ranging from 0 to 255.
- Color bar: Display of the extracted color.

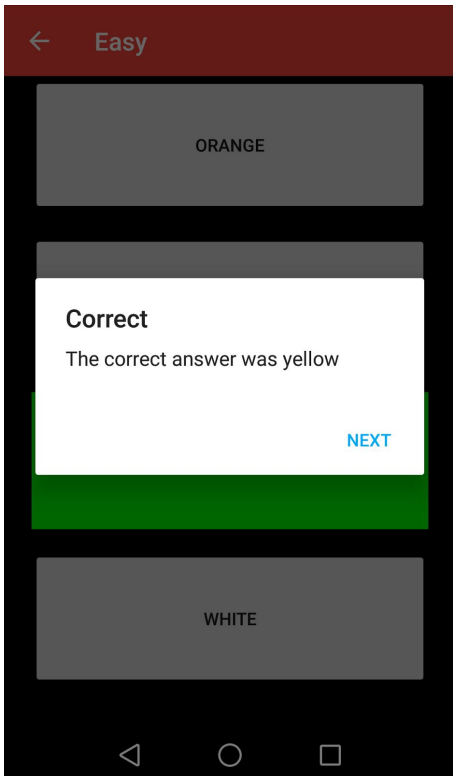
View Image



When opening this module, the user is prompted to select an image from the phone's memory. Sounds from the image is played when swiping the image. The same debug information shown in the Camera module is displayed. There is also a button which is a shortcut to the settings menu. The sidebar and navigation bar can be toggled by pressing anywhere on the screen.

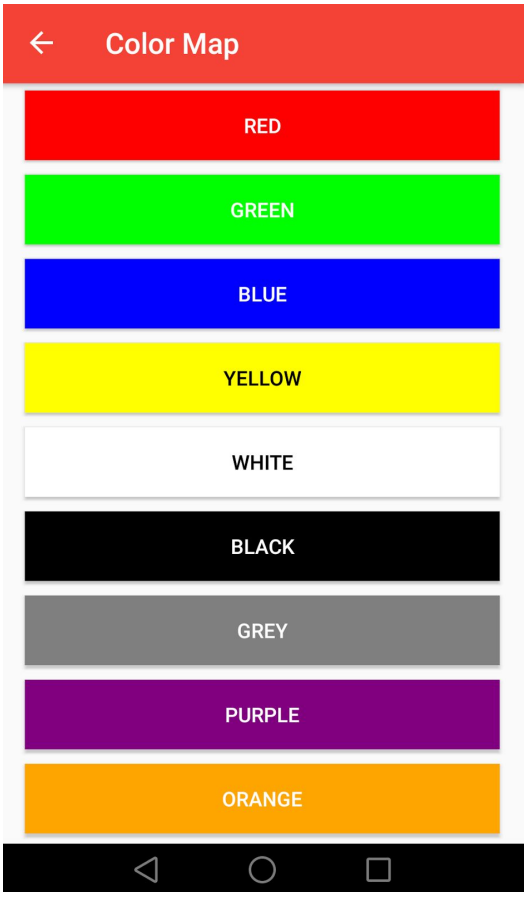
Play





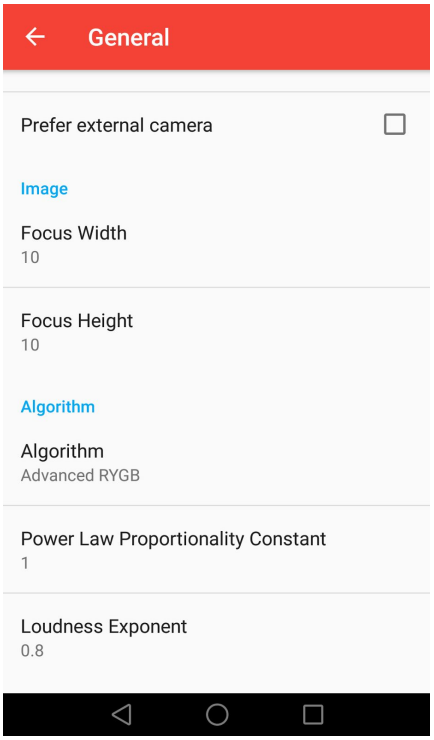
In the first menu, the user can select between different levels (Easy, Medium, Hard, Flags). After choosing a level, a black screen is shown for the first question of that level. Hidden underneath lies an image that is swipeable identical to the way the image viewer works. Single-tapping anywhere on the screen toggles the navigation bar. Double-tapping anywhere on the screen displays the answering options for that question. After answering a pop-up dialog will provide information about what color was correct. Tapping the Next button will make take the user to the next question. After going through all of the questions, information about the number of right and wrong guessed will be displayed.

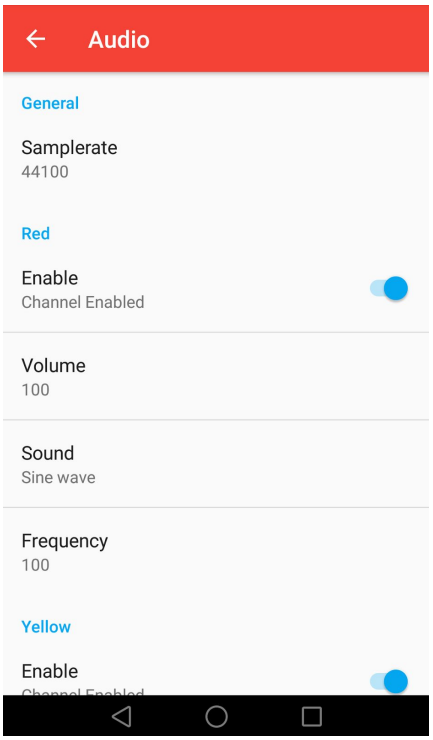
Color Map



The Color Map module has buttons corresponding to their names and colors. When held down the sound for that button's color is played. More colors are found by scrolling downwards.

Settings





When entering the settings menu two different settings can be chosen:

- **(1) General**: camera, image and algorithm settings. For the camera settings, the user can choose between center and peripheral mode, which camera to prefer if an external one is plugged in and the height and width of the extractor (Focus Width and Focus Height). The size of the extractor can be changed for the image viewer as well. Algorithm settings let the user choose between the different implemented algorithms: Basic RYGB, Advanced RYGB, and Basic RGB. Steven's power law proportionality constant can be set, and the loudness exponent can be adjusted, both to a value between 0 and 1. There is also an option to restore all settings to the defaults.
- **(2) Audio**: settings for the different audio channels. Three different sample rates for the sound are available: 11025, 22050 and 44100. For each of the five audio channels (red, green, blue, yellow, white) the volume (ranging from 0-100) can be set

and the sound channels can be enabled and disabled. For the four color channels the sounds (different types of waves like sine, sawtooth, etc. and instrument sounds like a violin, flute, organ), and frequency (if one of the wave sounds are chosen) can be set in addition. For the white channel, the Noise Alpha can be adjusted as well (a value between 0 and 2).

Connecting external camera

Plugging in a compatible USB camera will prompt the user to make Colorophone Research the default app to open when plugging in the camera. The app does not have to be open for this to happen. If the user chooses “Remember my choice” the app will open when the camera is plugged into the phone without asking in advance the next time. The Camera module in the app will then be opened, and the input from that camera will be used instead of the internal camera on the phone.

Appendix H Development guide

Development guide

Colorophone Research App

Introduction

This guide is intended to help developers get started with further development of Colorophone Research. Readers are assumed to be familiar with programming in Java for Android.

Setup

Recommended setup:

1. Clone the repository https://github.com/ezet/clrf-navigator
2. Install the latest version of Android Studio.
3. Import the project as a Gradle Project.
4. Use Gradle sync to install all required dependencies.

Common use cases

Implementing a sonification algorithm

1. Create a class implementing *sonification.algorithm.ISonificationAlgorithm.*
2. Implement the *sonificate()* method. The method takes 3 arguments: *red*, *blue* and *green*, which are the color values returned by the feature extractor in range [0-255]. The return value should be an instance of *AlgorithmOutput*, retrieved using the static *AlgorithmOutput.obtain()* method, with values representing the output value for each channel in the range [0 - 255].
3. Register the new algorithm with the application using *AppSettings.registerAlgorithm()* in *AppSettings.registerAlgorithms()*. The method requires 3 arguments; a human readable name, a unique key to identify the algorithm, and an implementation of *IAlgorithmFactory* that returns an instance of the algorithm.

Adding a pre-recorded sound

1. Place the sound file in *res/raw*.
2. Register the new sound with the application using *AppSettings.registerSound()* in *AppSettings.registerSounds()*. The method takes 4 arguments; a human readable name, a unique key to identify the sound, a boolean stating whether the sound support custom frequency, and and implementation of *IAudioTrackFactory* that returns an instance of *AudioTrack* with the new sound. *TrackFactoryUtils.createRawTrack()* can be used to simplify track creation, and takes the sound resource as an argument.

Only *.wav* files are supported, and they need to be 16-bit PCM, 44100Hz samples. The sound should have a smooth transition when repeated.

Adding a sound generator

1. Create a new class that implements *ISampleGenerator*.
2. Implement *nextSample()*, returning an integer in the range [-1, 1] that represents the next audio sample.
3. Register the new generator with the application using *AppSettings.registerSound()* in *AppSettings.registerSounds()*. The method takes 4 arguments; a human readable name, a unique key to identify the sound, a boolean stating whether the sound supports custom frequency, and and implementation of *IAudioTrackFactory* that returns an instance of *AudioTrack* with the new sound. *TrackFactoryUtils.createGeneratedTrack()* can be used to simplify track creation.