**<Add an attractive title page>**

**<Name of your project, the project name should have a direct link with the content of your project. The name of your project has a maximum length of 40 characters. Use a picture if possible>**

Name of the group: <name>  
Name of the group members: **<name, student number>**  
Document ID: 2017S\_TEMPLATE SYSTEM DESIGN DOCUMENT\_SDD.DOCX . <update this name!>

|  |  |  |  |
| --- | --- | --- | --- |
| **Release** | **Date** | **Author** | **Main changes** |
| R01 |  |  |  |
| R02 |  |  |  |
| R03 |  |  |  |

Authenticated by: <name and \_\_\_\_\_\_//SIGNED//\_\_\_\_\_\_\_\_>   
Approved by: <name and \_\_\_\_\_\_//SIGNED//\_\_\_\_\_\_\_\_>

Table of Contents

[1. Introduction 3](#_Toc471744746)

[1.1. Background and context 3](#_Toc471744747)

[1.2. Definitions and Abbreviations 3](#_Toc471744748)

[1.2.1. Definitions 3](#_Toc471744749)

[1.2.2. Abbreviations 3](#_Toc471744750)

[2. Functional breakdown 3](#_Toc471744751)

[3. Concept selection (the morphological overview) 3](#_Toc471744752)

[4. Architecture 4](#_Toc471744753)

[4.1. Hardware design 4](#_Toc471744754)

[4.2. Software design 5](#_Toc471744755)

[5. System test plan 5](#_Toc471744756)

[References 5](#_Toc471744757)

**List of figures**

**List of tables**

# Introduction

This document contains the choices for the key components and design considerations of the Fitbit interval training app.

## Background and context

The [Fitbit interval training app] is intended to be used by users of Fitbit devices, namely the devices that are compatible with Fitbit OS 5: Ionic, Versa, Versa 2, Versa lite, Versa 3 and Sense. In particular Fitbit users that perform interval training in running and cycling. However the intent of developing this app is mainly fuelled by the desire to improve interval training with the Fitbit for myself. The basic app should have options for interval training in running and cycling. Interval training for other sports can be added after the basic version of this app is up and running.

### Definitions

### Abbreviations

# Architecture

<This chapter has the following contents:

* System overview (context diagram)

< The context diagram should show the general information flow and components of the system and will be divided up into the hardware and software diagrams in the next step>

* More detailed block diagrams (functional decomposition) for Hardware and Software (related to the chosen concept)
* Name and definition of all design blocks and their connections (inputs and outputs such as wiring, digital, protocols, analogue, voltages, etc).

The system is a black box; you define all these in- and outputs in a data dictionary!

So for this chapter: visualize the system design in function blocks starting at the top level (see Figure 1, DFD\_0 heart beat)

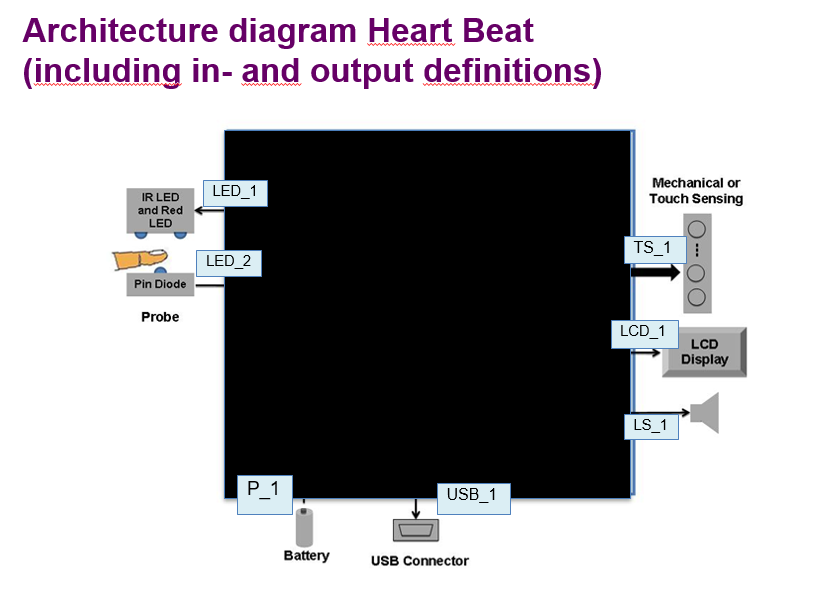


Figure 1, DFD\_0 heart beat

First define the in- and outputs (see Figure 2, A Data Flow Diagram (DFD) table).

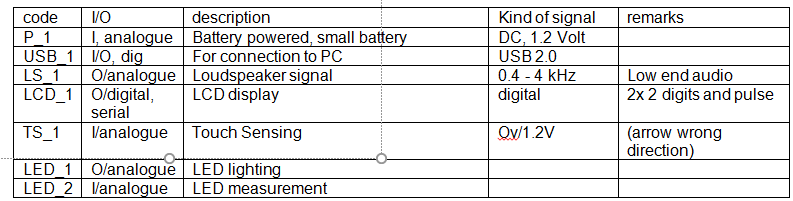


Figure 2, A Data Flow Diagram (DFD) table

## Hardware design

<Describe the overall system hardware and organization. Include a list of hardware blocks with a brief description of each block, input and output definitions and diagrams showing the connectivity between the blocks ([1], [2]). If appropriate, use subsections to address more subsystems. Assign for each block (also for the input and output functions) a target cost price in such a way that, when adding these separate cot price targets, you meet the target cost price as mentioned in the SRD> So the next step is zooming in for the modules in the Blackbox and go over the data flow diagrams again but now for all blocks defined.

2.1.1 Block diagram Hardware (go into the Black box)

2.1.2 Description per block, definition of the interfaces (inputs and outputs) for each block, needed for your MDD assignments

# 3. Software design

Fitbit allows for application development for Fitbit OS through the Fitbit software development kit. Applications for Fitbit OS are developed using Javascript, CSS and SVG. Javascipt is run on the device using the Jerryscript engine. Aside from the device API for Fitbit OS apps, there is also a companion API that allows for the creation of a companion app on a mobile device. The companion app runs within the Fitbit mobile application. The application will be developed using Fitbit studio. The Fitbit build folder must have the following structure:

/app/

/common/

/companion/

/resources/

/settings/

* /app/ contains the application logic which executes on the Fitbit device. It has access to the device API.
* /common/ contains code that is shared between the Fitbit application and the companion. Files are created as ES6 modules.
* /companion/ contains the companion logic which executes on the mobile device. It has access to the companion API.
* /resources/ contains CSS/SVG resources which make up the user interface of the device and companion applications.
* /settings/ contains application settings, written using React JSX. Code within this folder has access to the settings API.

<Describe the overall system software and organization. Include a list of software modules (this could include functions, subroutines, or classes), computer languages, and programming computer‐aided software engineering tools (with a brief description of the function of each item). Use structured organization diagrams/object‐oriented diagrams that show the various segmentation levels down to the lowest level. All features on the diagrams should have reference numbers and names. Include a narrative that expands on and enhances the understanding of the functional breakdown. If appropriate, use subsections to address each module.

## 3.1 Device application

The architecture of the device application is seen in figure 3. The functionality of the application is divided into distinct function blocks, which are described below.

Diagram

Description automatically generated

Figure 3: Device application architecture

### 3.1.1 UI Renderer

The UI renderer is responsible for rendering the user interface on the screen of the device. It has the following tasks:

* Render the device UI

### 3.1.2 UI Controller

The UI controller is responsible for interactions with the user interface. It is also the messenger between the user interface and the other function blocks. It has the following tasks:

* Control button clicks
* Cycle through different screens
* Update display when data changes
* Start workout
* Display workout analysis

### 3.1.3 Active workout controller

The active workout controller tracks an active workout. It has the following tasks:

* Start next interval
* Track distance
* Track heartrate
* Track time
* Stop interval with vibration signal
* Stop workout with vibration signal

### 3.1.4 Workout loader and saver

The workout loader and saver loads a programmed workout and saves a completed workout. It has the following tasks:

* Save completed workout
* Load workout template

### 3.1.5 Workout file retriever and sender

The workout file retriever and sender send a completed workout to the companion application and receives a workout template from the companion application. It has the following tasks:

* Send completed workout to the companion application
* Receive workout template from the companion application

## 3.2 Companion application

The architecture of the companion application is seen in figure 4. The functionality of the application is divided into distinct function blocks, which are described below.

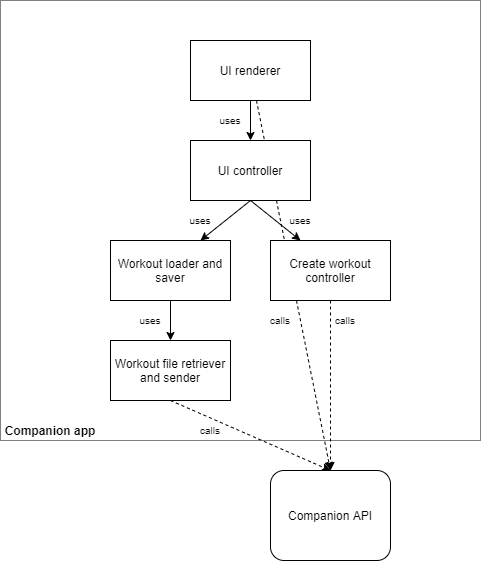


Figure 4: Companion application architecture

### 3.1.1 UI Renderer

The UI renderer is responsible for rendering the user interface. It has the following tasks:

* Render the companion UI

### 3.1.2 UI Controller

The UI controller is responsible for interactions with the user interface. It is also the messenger between the user interface and the other function blocks. It has the following tasks:

* Control button clicks
* Cycle through different screens
* Update display when data changes
* Display workout analysis
* Adjust settings
* Share a completed workout on social media

### 3.1.3 Create workout controller

The create workout controller creates a new workout template that can be send to the device. It has the following tasks:

* Modify workout template
* Add new workout template

### 3.1.4 Workout loader and saver

The workout loader and saver loads stored workout templates and stored completed workouts. It also saves a new workout template. It has the following tasks:

* Load workout templates and completed workouts
* Save new workout template

### 3.1.5 Workout file retriever and sender

The workout file retriever and sender sends a workout template to the device and receives a completed workout from the device. It has the following tasks:

* Send workout templates to the device
* Load completed workout from the device

# 4. System test plan

<In this section, system test plan, you describe how the specifications of your system design will be tested to show, after prototyping your design, you have reached all the specifications of the system given to you by the product manager; This all according the V-model guidelines and according the test plan template. Apply test plan only on the chosen concept.  
Indicate more precise how the test has to be carried out; there should be guidance for the tester>

# References

For your SDD schematics:

[1] <http://www.industrial-electronics.com/drafting-for-electronics-12.html>

[2] <https://syntherjack.net/synth-block-diagram-guidelines/>