Software Design Document (SDD)

Software design is a process by which the software requirements are translated into a representation of software components, interfaces, and data necessary for the implementation phase. The SDD shows how the software system will be structured to satisfy the requirements. It is the primary reference for code development and, therefore, it must contain all the information required by a programmer to write code. The SDD is performed in two stages. The first is a preliminary design in which the overall system architecture and data architecture is defined. In the second stage, i.e. the detailed design stage, more detailed data structures are defined, and algorithms are developed for the defined architecture.

This template is an annotated outline for a software design document adapted from the IEEE Recommended Practice for Software Design Descriptions. The IEEE Recommended Practice for Software Design Descriptions have been reduced in order to simplify this assignment while still retaining the main components and providing a general idea of a project definition report. For your own information, please refer to IEEE Std 10161998 for the full IEEE Recommended Practice for Software Design Descriptions.

Team1

Fatigue and somnolence detection

Software Design Document

Name (s): S.Baroudi & N.Fassi

Date: (11/12/2019)

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

This software design document describes the architecture and system design of the ADAS fatigue and somnolence detection system. It will help the different team members being updated about the system and knowing the steps to achieve nicely, precisely and on the schedule the requirements.

1.2 Scope

Drowsiness or fatigue is a major cause of road accidents and has significant implications for road safety. Several deadly accidents can be prevented if the drowsy drivers are warned in time. A variety of drowsiness detection methods exist that monitor the drivers' drowsiness state while driving and alarm the drivers if they are not concentrating on driving. The relevant features can be extracted from facial expressions such as yawning, eye closure, and head movements for inferring the level of drowsiness. In our case, our Somnolence Recognition System continually analyzes driver's grimaces via a camera installed on the driver's wheel, so
that it can recognize signs of sleepiness before the driver falls asleep.

1.3 Overview

This documents is organized in five main parts ; an introduction part, explaining the goals of the project and its scope, a system overview describing the context and the design of the system, a system architecture part to develop the relationship between the different modules of the system, and a data design part that would help developers handle the entries and the input/output data.

1.4 Definitions and Acronyms

ADAS: advanced assistance aide system
CAN-BUS : controller area network bus
ECU : electronic control unit
Req: short for requirement
DFD : data flow diagram
OS: operating system
DEV: development
FIFO: first in first out
ERA: Eye aspect ratio

SYSTEM OVERVIEW

The aim of this functionality is to prevent accident caused by somnolence. For this, the system will be designed as follows:

rotative cameras will detect the driver's face and eyes and decide, through image processing, if he is sleepy, if so, the driver will be alerted by a light sound signal.
This solution will be developed on an embedded platform to be easily integrated on the vehicle main system without disturbing the driver's comfort and its data will be transferred to the dashboard using a CAN-Bus, since it is he most used communication protocol for the automotive industry and that it presents some interesting advantages.

3. SYSTEM ARCHITECTURE

Hardware Environment

Pi Camera Device that deliver the live video feed of the driver to the different processing components, it is also automatically compatible to the Raspberry Pi microcontroller.

Rotative camera support Device that ensures the continuous mechanical tracking of the driver's face, in case it moves out of the camera's range.

Piezo Buzzer Device that generates a sound signal using the principle of piezoelectricity. It alarms the drivers in case he is sleeping.

CAN Bus module Device that ensures the communication of data using a CAN protocol between the different nodes.

Raspberry Pi 3 Microcontroller used for processing the data coming from the camera feed, and deciding of the driver's state, therefore alerting him in case he is sleeping. This device will ensure the link between the different nodes and the dashboard used for displaying messages and alerts to the driver.

Software Environment

Linux The OS or our DEV environment used in the Raspberry Pi card, as it is the most suitable OS for our applications. We will use the distribution with the Linux kernel, the Raspbian distribution.

Python This language ensures wide and rich libraries that will be used for data processing, image processing and data transfer. Therefore, offering pre-made functions for our applications. The main libraries are:

OpenCV Library offering functions for image processing with additional functions found in the Haar cascades XML files.

Numpy Library used for numerical and mathematical operations; it is useful for array processing. Image are made of pixels that can be stored in arrays and processed after.

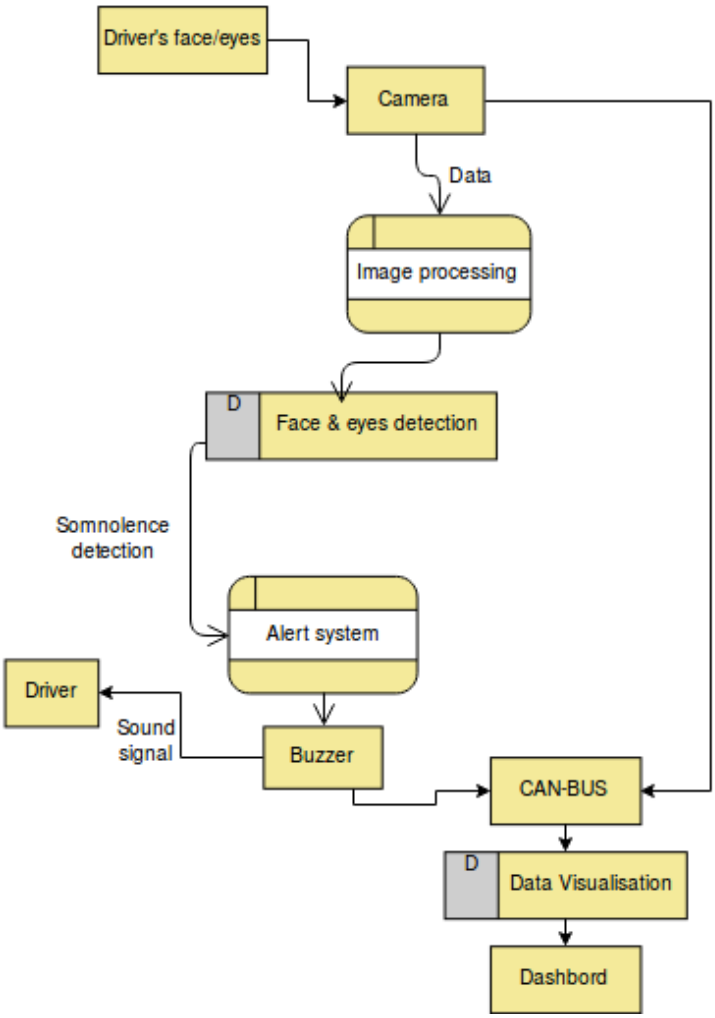
OpenCAN Library offering functions for the CAN protocol for our communication with the man dashboard.

3.1 Architectural Design

Specifications have been defined into distinct tasks:

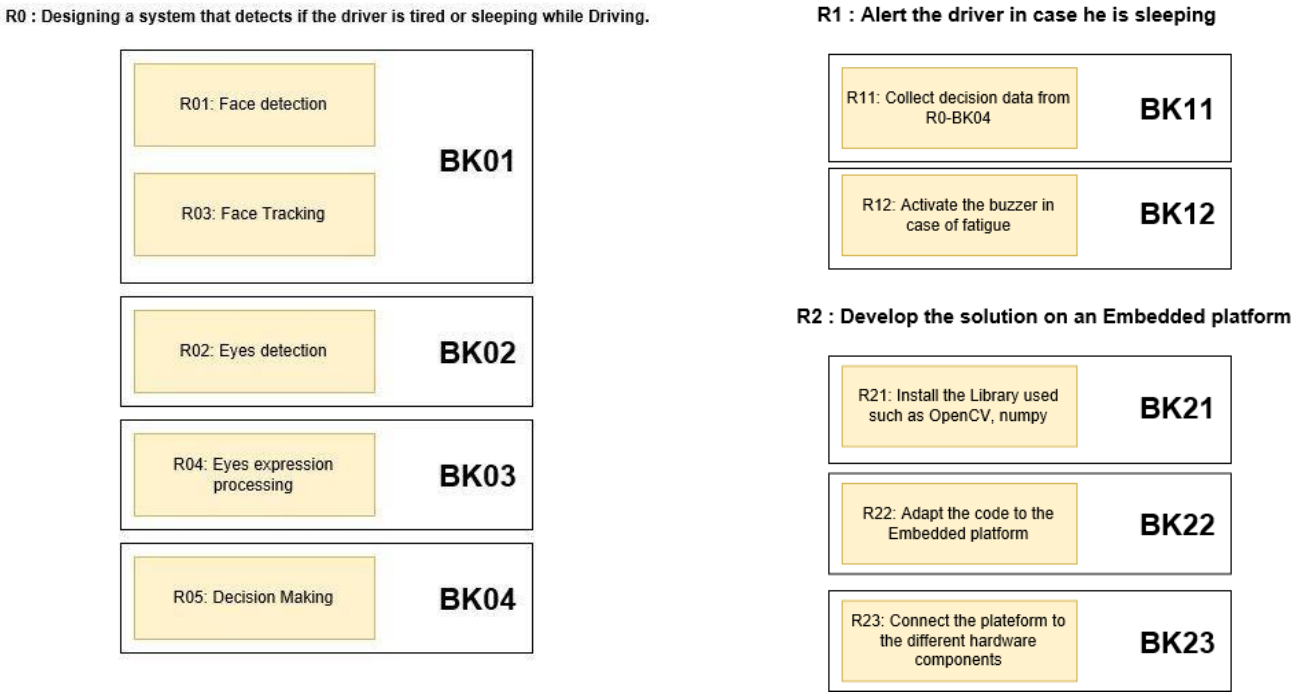
- R0: Designing a system that detects if the driver is tired or sleeping while driving, using eye tracking and face expressions processing.
- R1: Alert the driver if he is sleeping with a sound signal (not sudden or loud, so that the driver does not lose control over the vehicle)
- R2: Developing this solution on an embedded platform
- R3: transferring data into a dashboard using a CAN-BUS

Below is the top-level data flow diagram (DFD) to understand to functional description:



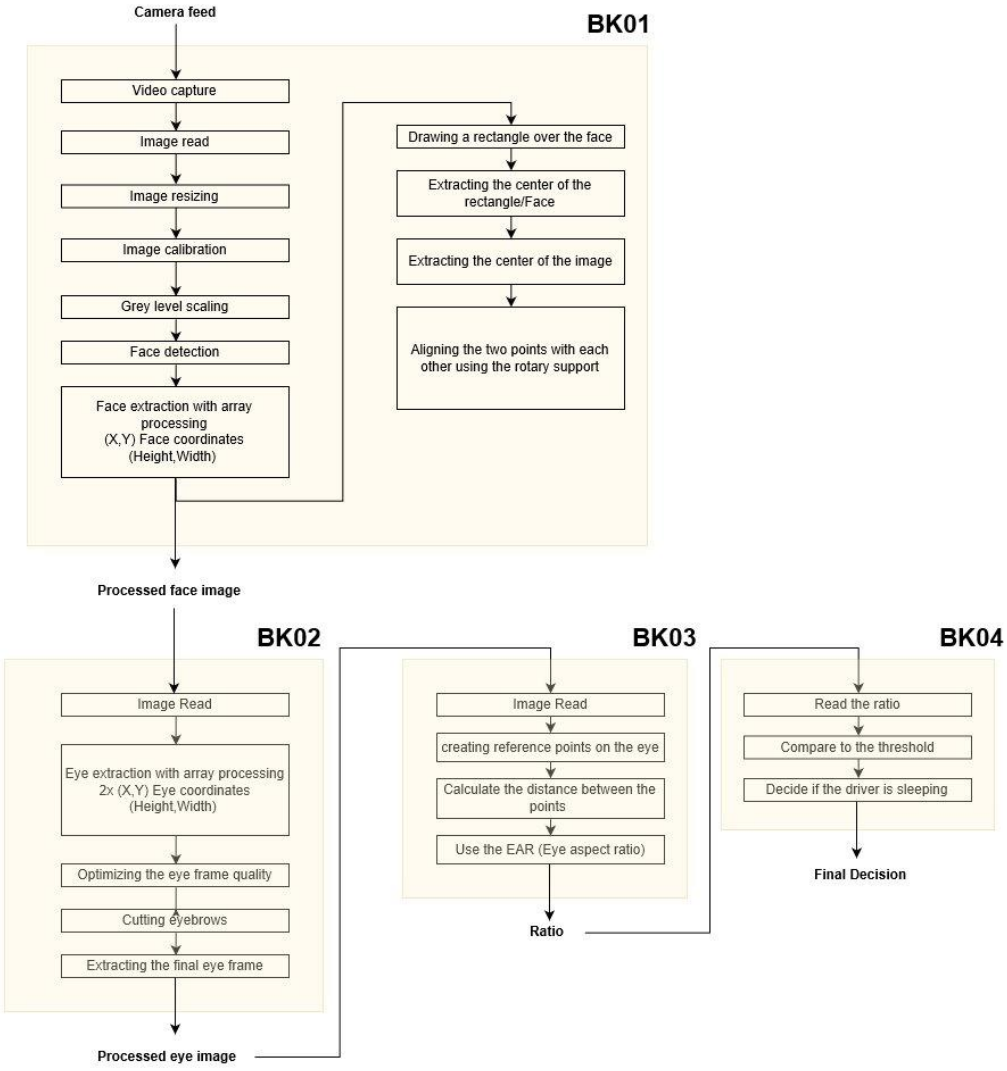
REQUIREMENTS BLOCKS

The requirements have been dissected into blocks. Each block has internal functions with specific inputs and outputs. Blocks can communicate between each other to create one distinct function. This step is important for structuring the functions and their implementation afterwards.



BLOCKS DESCRIPTOR

For the R1 and R2, the blocks are explicit enough but for the first requirement R0, each block should be explained separately with data flow (I/O) for each function.



4.1 Data Description

The data collected from the camera will be transferred to the image processing system, based in the Raspberry Pi card and displayed on the dashboard. It will be stored then in the ssd memory card of the Raspberry in a specified memory space and will be eliminated respecting a FIFO logic.

4. COMPONENT DESIGN

Bellow will be listed, in procedural description language (PDL), a summary of the function's algorithm:

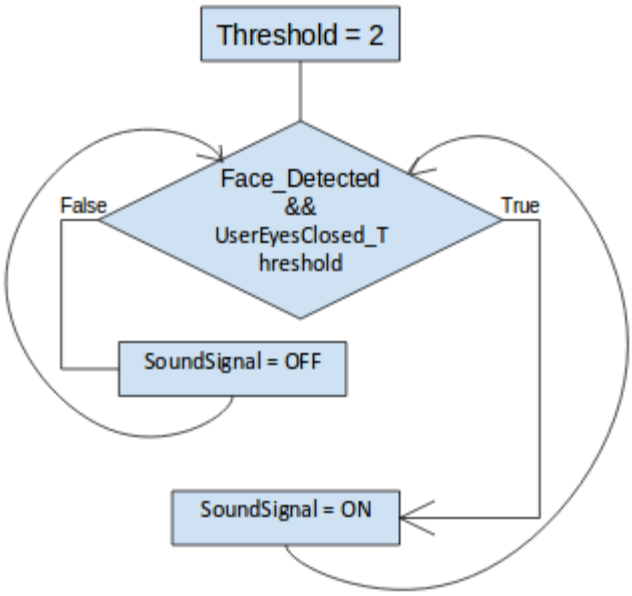
Threshold = 2

IF **Face_Detected** = TRUE

AND **UserEyesClosed_Threshold** = TRUE

THEN **SoundSignal** = ON

ELSE **SoundSignal** = OFF



6. REQUIREMENTS MATRIX

Requirements:

R0: Designing a system that detects if the driver is tired or sleeping while driving, using eye tracking and facial expressions processing.

BK01:

R01: face detection

after receiving the camera feed, the algorithm of the system will proceed by taking a video capture to read an image. This image will then be resized and calibrated before being scaled into grey level.

the face will be detected using the Haar cascade algorithm and extracted with the array processing principle (using the face coordinates that represents height and width) so as to be able to contour it with a rectangle.

R03: face tracking

now that the face is extracted and contoured with a rectangle, we will extract the center of the rectangle (the face) based on the height and the width. At the same time, the center of the image will be extracted also, so that the program could align the two extracted points (center of the image and the center of the face) using the rotary support of the camera.

BK02:

R02: eyes detection

same principle used in the face detection, eye detection algorithm will use array processing and extract the height and width of both eyes. The script cuts then the eyebrows to extract more precisely the finale eye frame. Haar Cascade algorithm will help achieve this requirement

BK03:

R04: eyes expressions processing

Same process as rectangle's centre extraction; the program will extract the centre of the upper side of the rectangle and the centre of its lower side. It will then calculate the distance of the two points so as to use the Eye aspect ratio algorithm (ERA)

BK04:

R05: decision making

The ratio generated in the previous step by the EAR will be read and compared to an agreed threshold of somnolence to decide if the driver is sleepy/ sleeping or not.

R1: Alert the driver if he is sleeping with a sound signal (not sudden or loud, so that the driver does not lose control over the vehicle)

R11: Collect the decision data from R0-BK04

R12: Activate the sound signal

In the case of fatigue, a synthetic alarm sound will be generated to help the driver awake.

R2: Developing this solution on an embedded platform

R21: install the libraries used such as OpenCV and numpy

So that the developing version on the computer and on the raspberry are compatible

R22: adapt the code to the embedded platform

to read the inputs and outputs from the raspberry GPIO

R23: connect the platform to the different hardware components

R3: Transferring data into a dashboard using a CAN-BUS

R31: specify the data to send

R32: put the data into a frame structure

Test protocol:

R0: Designing a system that detects if the driver is tired or sleeping while driving, using eye tracking and facial expressions processing.

BK01:

R01: face detection

T01.1: face detected and contoured

T01.2: face detected and not contoured

T01.3: face not detected
R03: face tracking
T03.1: the proper functioning of the rotative support
T02.2: the rotative support follows the center of the camera & thre rectangle
T03.3: the rotative support doesn't follow
BK02:
R02: eyes detection
T02.1: eyes detected and contoured
T02.2: eyes detected and not contoured
T02.3: eyes not detected
BK03:
R04: eyes expressions processing
T04.1: upper and lower rectangle side properly calculated, EAR generated
T04.2: algorithmic/ syntaxe error, EAR in not generated
BK04:
R05: decision making
T05.1: the decision is compatible with the actual state of the driver
T05.2: the decision is not compatible

R1: Alert the driver if he is sleeping with a sound signal (not sudden or loud, so that the driver does not lose control over the vehicle)
R11: Collect the decision data from R0-BK04
R12: Active the sound signal
T12.1 : Fatigue detected, sound generated
T12.2: Fatigue detected, sound not generated
T12.3: Sound generated without any detected fatigue
R2: Developing this solution on an embedded platform
R21: install the libraries used such as OpenCV and numpy
So that the developing version on the computer and on the raspberry are compatible
R22: adapt the code to the embedded platform
R23: connect the platform to the different hardware components
T23.1: connexion succeeded
T23.2: connexion failed
T23.3: working components
T23.4: probleme in components performances

R3: Transferring data into a dashboard using a CAN-BUS
R31: specify the data to send
R32: put the data into a frame structure
T3.1:the data transfer succeeded
T3.2:it doesn’t

Requirements	Testing bloc	Test Case
R0	R01 R03	T01.1 T01.2

	R02 R04 R05	T01.3 T03.1 T03.2 T03.3 T02.1 T02.2 T02.3 T04.1 T04.2 T05.1 T05.2
R1	R11 R12	T12.1 T12.2 T12.3
R2	R21 R22 R23	T23.3 T23.4
R3	R31 R32	T3.1 T3.2

8. APPENDICES