**Embedded Systems Engineering**

**Product report**

A face of a person

Description automatically generated

*Facemesh facial expression detection*

**Embedded Systems Engineering  
Academy Engineering and Automotive  
HAN University of Applied Sciences**

Authors

|  |  |
| --- | --- |
| ***1548148 2188888 1671261  1659237*** | ***Jaap Jan Groenedijk Kevin van Hoeijen Enes Çaliskan William Hak*** |

Date

|  |
| --- |
| ***January 2025*** |

Version

|  |
| --- |
| **0.2** |

# Revisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **When** | **Who** | **What** |
| 0.1 | 14-12-2024 | J.J.Groenendijk | Technical Design init |
| 0.2 | 12-1-2025 | K van Hoeijen | Front page, introduction and functional design |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Preface

*Looking back on the course of the project, how did the project group experience the project, what did the group learn from it, what will the group do better next time; in short, the preface can contain all kinds of personal reflections on the project and its course.*

# Summary

*Description of the starting points, goals to be achieved, what was and was not achieved,* ***results achieved****; the summary should give an overall impression of the whole assignment and is maximum 1 A4.*

# Contents

[Revisions 2](#_Toc190277315)

[Preface 3](#_Toc190277316)

[Summary 4](#_Toc190277317)

[Contents 5](#_Toc190277318)

[1 Introduction 7](#_Toc190277319)

[1.1 Reason 7](#_Toc190277320)

[1.2 Objective 7](#_Toc190277321)

[1.3 Report structure 7](#_Toc190277322)

[2 Functional design 8](#_Toc190277323)

[2.1 Functional specifications 8](#_Toc190277324)

[2.2 Technical specifications 9](#_Toc190277325)

[2.3 User interface 9](#_Toc190277326)

[3 Technical Design 10](#_Toc190277327)

[3.1 Host hardware 10](#_Toc190277328)

[3.2 Host software 10](#_Toc190277329)

[3.3 Architecture 10](#_Toc190277330)

[3.3.1 Application container 10](#_Toc190277331)

[3.4 System Flow 10](#_Toc190277332)

[3.4.1 Technology Stack 10](#_Toc190277333)

[3.4.2 Project Structure 11](#_Toc190277334)

[3.5 Interfaces 12](#_Toc190277335)

[3.5.1 Power Supply 12](#_Toc190277336)

[3.5.2 USB Interface: PC-Camera 12](#_Toc190277337)

[3.5.3 ROS 2 Communication 13](#_Toc190277338)

[4 Realisation 14](#_Toc190277339)

[4.1 Hardware 14](#_Toc190277340)

[4.2 Software 14](#_Toc190277341)

[5 Testing 15](#_Toc190277342)

[6 Conclusions and recommendations 16](#_Toc190277343)

[7 References 17](#_Toc190277344)

[Appendix A 19](#_Toc190277345)

[Appendix B 20](#_Toc190277346)

[Appendix n 21](#_Toc190277347)

# Introduction

This project is conducted to realise a product that is able to process image data and predict what kind of expression is shown at a person's face.

## Reason

Whilst the shortage of trained professionals rises, the demand for good quality care keeps rising. To reduce the workload of care professionals our team was asked to provide a solution that requires less human intervention, so that professional care can be given to whom who needs it the most while mitigating the loss of costly human checkups.

## Objective

Our objective is to feed back a single stream of human facial expressions back to an interface so that that information, and that information only, can be used to trigger a checkup on a patient. This wile guaranteeing the privacy of the patient in question, by not storing any visual data of the video stream.

This goal is achieved by using 3D-coordinates provided by the FaceMesh model by Mediapipe.

## Report structure

The project document is oorganized as followed;

**Chapter 2:** Provides an overview of the functional design, what the end product should and should not do. Specified by the SMART and MoSCoW method.

**Chapter 3:** Outlines the technical design, detailing the hardware and software specifications, system architecture, technology stack, project structure, and interfaces.

**Chapter 4:** Describes the implementation phase, focusing on the development process, tools, and methodologies used to build the system.

**Chapter 5:** Explains the system testing and validation process, including the testing framework, procedures, and results.

**Chapter 6:** Discusses the evaluation of the project, highlighting performance analysis, system limitations, and areas for improvement.

# 

# Functional design

The function design chapter servers as a blueprint for the development of the system. Here lies the functional and technical aspects of what the system must, would, should and will not do.

The requirements are, when applicable, noted in the SMART method.

In this chapter, we define detailed functionalities as well as optional features that provide usability and integration. By sorting the requirements hierarchically it provides a structural approach of realising the project whilst maintaining flexibility for future enhancements.

## Functional specifications

| **SMART functional specification** | | |
| --- | --- | --- |
| **#** | **MoSCoW** | **Description** |
| **F1** | **M** | **The system will process a live video stream in real-time.** |
| F1.1 | M | The system will render a Face Mesh on the detected face during analysis. |
| F1.2 | M | The system will detect and process eyebrows and mouth movements to recognize facial expressions. |
| **F2** | **M** | **The system will detect and track changing facial expressions sequentially within a single video stream automatically.** |
| F2.1 | M | The system will recognize: “Neutral” “Smiling” “Frowning” and “Big smile with visible teeth” |
| F2.1.1 | M | The system will recognize different specified facial expressions with an accuracy of at least 90% |
| **F3** | **M** | **The system logs facial expressions in a file** |
| **F4** | **M** | **The system logs facial expressions through an API** |
| F4.1  F4.2 | S  S | The system uses Flask as API  The system uses ROS-2 as API |
| **F4** | **M** | **The system will detect when there is no face present** |
| **F5** | **M** | **The system will project the FaceMesh on the face in the video stream for live feedback** |
| **F6** | **M** | **The system provides the accuracy of the prediction in the videostream and in the output** |
| **F7** | **C** | **The system detects when a face is not straight for the camera** |
| **F7** | **W** | **The system provides a possibility to save the videostream** |

## Technical specifications

| **SMART technical specification** | | |
| --- | --- | --- |
| **#** | **MoSCoW** | **Description** |
| **T1** | **M** | **The system will work on a generic laptop** |
| T1.1 | M | The system will work on Ubuntu operating system |
| T1.2 | M | The system will work on OUMAX Mini PC Intel N100 16GB 500GB or Max N95 8GB 256GB hardware. |
| **T2** | **M** | **The system will operate effectively up to 2 meters away from the camera.** |
| **T3** | **M** | **The system will operate effectively up to 2 meters away from the camera.** |
| **T4** | **M** | **The system will work under diffused LED lighting conditions.** |
| **T5** | **M** | **The system will remain operational when a face is tilted up to 45 degrees relative to the camera.** |
| **T6** | **M** | **The system will process facial expression changes with a delay of no more than 1000 ms.** |
| **T7** | **M** | **The system will visually display emotion scores with a delay of no more than 1000 ms.** |
| **T8** | **S** | **The system should utilize hardware acceleration via a TPU** |
| **T9** | **S** | **The system should utilize TensorFlow Lite for microcontroller implementations** |
| **T10** | **C** | **The system could support various input resolutions for training and live feed** |
| **T11** | **C** | **The system could be developed in C++** |

## User interface

Debug user interface for checking video streaming and prediction of emotions. Inputs are video streams only from the built-in or plugged in webcam of the operating system it is running on. From here the only user interface that is usable on UI level is as is below:

A screenshot of a computer

AI-generated content may be incorrect.

The used face mesh landmarks are drawn on the face in the video input, and the emotion and the emotion score is displayed

A black background with white text

AI-generated content may be incorrect.

# 

# Technical Design

## Host hardware

The client for the Facemesh project already defined which hardware is preferred. The system will run on a single board computer (SBC) with a low power CPU. System specifications of the host is as follows:

|  |  |
| --- | --- |
| CPU | Intel N100, 4 cores/threads, max turbo: 3.4GHz, Maximum Memory Speed  4800 MHz |
| PSU | 12V, 3.75A |
| RAM | 8 GB |
| Storage | 128 GB |

## Host software

Ubuntu LTS will be used as the host OS. The latest stable version of the docker engine is used as container engine.

## Architecture

The Facemesh solution consists of multiple microservices that are hosted in containers. Docker is used as container engine, and the orchestration of the services is done with a single Docker Compose file. Docker is used to make the software more portable. Splitting software in multiple services is not an active goal in the Facemesh project.

### Application container

The application container is based on a minimal python container. The application has the following functions

* Hosts a Flask server.
* Streams webcam video feeds to the backend.
* Processes video frames using OpenCV.
* Displays processed frames on the client-side web interface

The api container does the following:

* Connect to the application container
* Collect system logs
* Display the logs on a website hosted within the api container.
* Publish the logs on ROS2 protocol

## System Flow

Steps in the system:

The user accesses the Flask app through http://localhost:9009.

Webcam video feeds are streamed and processed in real-time using OpenCV and FaceMesh.

System logs are generated by the API container and displayed at http://localhost:9010.

### Technology Stack

* Programming Languages: Python
* Frameworks: Flask, OpenCV, Mediapipe
* Containerization: Docker, Docker Compose
* Protocols: HTTP, ROS2 for logging

### Project Structure

The project is organized into several directories and files:

1. **.vscode**
   * tasks.json: Defines tasks for building and running Docker containers.
2. **api**
   * app.py: Main API application file.
   * Dockerfile: Dockerfile for building the API container.
   * requirements.txt: Python dependencies for the API.
   * ros\_entrypoint.sh: Entrypoint script for the Docker container.
   * templates/
     + index.html: Main HTML template.
3. **application**
   * app.py: Main application file.
   * Dockerfile: Dockerfile for building the application container.
   * frame\_editor.py: Contains functions for manipulating video frames.
   * requirements.txt: Python for the application.
   * templates/
     + index.html: Main HTML template.
4. **Data**
   * Contains data-related files and directories (ignored in this documentation).
5. **dataset**
   * Contains dataset-related files and directories (ignored in this documentation).
6. **docker-compose.yml**
   * Docker Compose configuration file.
7. **output**
   * Contains output files and directories.
8. **pipeline**
   * fileHandler.py: Handles file operations.
   * imagePrep.py: Contains functions for image preprocessing and landmark detection.
9. **README.md**
   * Project README file.

## Interfaces

For each interface in the system, we describe its electrical and data communication properties. These specifications are determined based on technical requirements and design choices. Below, we outline the specifications and design considerations for the power supply, USB interface (PC-Camera), and ROS 2 communication.

## 3.5.1 Power Supply

The system is powered by a mini-PC with a built-in Intel Alder Lake N100 processor and an ASRock N100DC-ITX motherboard, designed for low power consumption and stable operation. The power supply ensures sufficient and reliable power delivery to all components.

**Specification:**

* **Voltage Requirements:**
  + AOOSTAR Mini-PC: 19V DC input via external power adapter.
  + Peripheral Devices: 5V or 3.3V (USB-powered or onboard power regulation).
* **Power Source:**
  + The system uses an external EU power adapter with a 19V DC output for the mini-PC.
  + Peripheral devices such as the USB camera draw power directly from the mini-PC’s USB 2.0 or 3.0 ports.
* **Maximum Current:**
  + The total system power budget must accommodate up to 65W (based on the mini-PC's power rating and connected peripherals).

## 3.5.2 USB Interface: PC-Camera

The USB interface connects the camera to the PC, enabling real-time video capture for the Face Mesh application. This interface also serves as the primary communication channel for transferring image data to the microcontroller for processing.

**Specification:**

* **Communication Protocol:** USB 2.0 High Speed (480 Mbps).
* **Driver Requirements:** The USB driver ensures reliable data transfer between the camera and the PC, enabling consistent frame rates for video streaming.
* **Voltage Levels:** The USB interface operates at 5V, delivering both data and power to the camera module.
* **Frame Rate and Resolution:** The camera supports up to 30 FPS at a resolution of 1280x720 pixels, meeting the requirements of the Face Mesh algorithm.

## 3.5.3 ROS 2 Communication

The system utilizes ROS 2 (Robot Operating System) to facilitate data exchange between the microcontroller and the PC. ROS 2 serves as the middleware for processing and publishing the detected facial expression data to other modules.

**Specification:**

* **Communication Medium:** Ethernet or Wi-Fi, depending on the system’s deployment environment.
* **Message Format:** The data is exchanged using ROS 2 messages in a standard format. For example, detected facial landmarks and expressions are published as sensor\_msgs or custom-defined message types.
* **Frequency:** The microcontroller publishes data at a rate of 10 Hz, ensuring near real-time updates.
* **Middleware Layer:** ROS 2’s DDS (Data Distribution Service) ensures reliable communication and supports Quality of Service (QoS) policies to handle varying network conditions.

## 3.5.4 Frontend

The frontend interface provides a real-time display of facial expression detection through a locally hosted web application (accessible via http://localhost:9009) running in Docker. The interface is designed to visualize both the original and manipulated video feeds side-by-side while displaying detected emotions.

1. **Interface Layout**

* **Left Panel:** Displays the original video stream from the webcam, processed via WebRTC.
* **Right Panel:** Shows the manipulated video feed with real-time facial landmarks overlaid using MediaPipe FaceMesh.
* **Emotion Label:** The detected emotion (e.g., 'Smiling,', ‘Smiling with teeth’, 'Frowning,' or 'Neutral') appears in the top-right corner of the manipulated video.
* **Accuracy Score:** Displays the confidence score of the detected expression.

1. **Functionality**

* **Real-Time Landmark Detection:** The application processes the video stream frame-by-frame using MediaPipe FaceMesh.
* **Dual Video Feed:** Displays both original and processed streams simultaneously.
* **Live WebSocket Logs:** Forwards detection logs to the API service (http://localhost:9010) using WebSockets and ROS2.
* **API Integration:** Logs are published to the ROS2 network and made available via REST endpoints.

# 

# Realisation

*Details of the realized hardware and software with accompanying explanations and calculations (such as power consumption, values of components, etc.). Complete and detailed diagrams of the hardware and listings of the software are included in the appendices.*

## Hardware

## Software

*The realized software is explained by means of code snippets. Make sure the code is easy to read by using syntax highlighting. Use code snippets that are no longer than 20 lines and that each line of code fits on one line in the report. Not all of the realized code needs to be explained. Choose two or three of the most relevant subsystems. The full code is included as an appendix. Also pay attention to the software development environment. In doing so, ask yourself what is important information for a fellow engineer who will be using the same development environment for the first time.*

Model tuning

# 

# Testing

*Unambiguous representation of how the system, hardware and/or software was tested. What hardware and or software modules were tested, how were the functional specifications tested during the acceptance test? What test setup was used and what were the final results. Did the tests meet the functional and technical specifications? The results are accompanied by a clear description of any remaining problems and how they might be explained. Were any 'work arounds' performed during testing? The tests must be described in such a way that each test can be reproduced by others.*

# Conclusions and recommendations

*Reflection on the goals of the project. What are the results? What has been achieved and what has not been achieved? What can be added to, expanded on, improved?*

# References

Adrián Sánchez Cano. (2013, 3 5). *Using RTC module on FRDM-KL25Z.* Retrieved from https://community.nxp.com/docs/DOC-94734

ARM. (2022, 04 26). *µVision® IDE*. Retrieved from https://www2.keil.com/mdk5/uvision/

ARM Developer. (2022, 04 26). *KAN232 - MDK V5 Lab for Freescale Freedom KL25Z Board*. Retrieved from https://developer.arm.com/documentation/kan232/latest/

Berckel, M. v.-v. (2017). *Schrijven voor technici.* Noordhoff Uitgevers B.V.

contributors, W. (2022, 07 06). *MoSCoW method*. (Wikipedia, The Free Encyclopedia) Retrieved 07 06, 2022, from https://en.wikipedia.org/w/index.php?title=MoSCoW\_method&oldid=1091822315

contributors, W. (2022, 05 25). *SMART criteria*. (Wikipedia, The Free Encyclopedia) Retrieved 07 07, 2022, from https://en.wikipedia.org/w/index.php?title=SMART\_criteria&oldid=1089766780

ELECFREAKS. (2022, 04 19). Retrieved from Ultrasonic Ranging Module HC - SR04: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

Freescale Semiconductor, I. (n.d.). *FRDM-KL25Z Pinouts (Rev 1.0).* Retrieved 3 31, 2023, from https://www.nxp.com/document/guide/get-started-with-the-frdm-kl25z:NGS-FRDM-KL25Z

Freescale Semiconductor, Inc. (2012, 9). *KL25 Sub-Family Reference Manual, Rev. 3.*

Freescale Semiconductor, Inc. (2013, 10 24). *FRDM-KL25Z User's Manual, Rev. 2.0.* Retrieved from https://www.nxp.com/document/guide/get-started-with-the-frdm-kl25z:NGS-FRDM-KL25Z

Freescale Semiconductor, Inc. (2014, 08). *Kinetis KL25 Sub-Family, 48 MHz Cortex-M0+ Based Microcontroller with USB, Rev 5.*

Hmneverl. (2015, 11 18). *De beslismatrix, het maken van keuzes*. (Info.NU.nl) Retrieved 07 06, 2022, from https://mens-en-samenleving.infonu.nl/diversen/164525-de-beslismatrix-het-maken-van-keuzes.html

NXP. (2022, 04 19). *Kinetis® KL2x-72/96 MHz, USB Ultra-Low-Power Microcontrollers (MCUs) based on Arm® Cortex®-M0+ Core*. Retrieved from https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/kl-series-cortex-m0-plus/kinetis-kl2x-72-96-mhz-usb-ultra-low-power-microcontrollers-mcus-based-on-arm-cortex-m0-plus-core:KL2x?tab=Buy\_Parametric\_Tab#/

NXP. (2022). *OpenSDA Serial and Debug Adapter*. Retrieved from https://www.nxp.com/design/software/development-software/sensor-toolbox-sensor-development-ecosystem/opensda-serial-and-debug-adapter:OPENSDA?&tid=vanOpenSDA

Solomon Systech Limited. (2008, 4). *SSD1306: Advanced Information.* Retrieved from https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf

Vishay Semiconductors. (2017, 8 9). *TCRT5000(L), Reflective Optical Sensor with Transistor Output, Rev. 1.7*.

# Appendix A

# Appendix B

# Appendix n