This document presents the work carried out for the installation, configuration and use of the application developed in this final degree work, whose objective is detect fluctuations in the perceived lighting of a spotlight (either through a live recording camera or through video) following profiles in different ROIs generating HTML reports.

Throughout this work, different phases of the software development are presented: analysis and planning, listing of requirements, reverse engineering, design, implementation, testing and evaluation of the software and hardware to achieve the objective of the project. In addition to a conclusion on my opinion about this project, future improvements and the knowledge learned.

Javier Expósito Martínez is the student in charge of the implementation and configuration of the project, and with this work he finishes his degree in Computer Engineering with a specialisation in Information Technologies.

Andrés María Roldán Aranda is the academic head of the present project, and the student's tutor. He is a professor in the Departament of Electronics and Computers Technologies at University of Granada

BACHELOR THESIS

OpenCV and Python application for automotive spotlight image processing

Javier Expóstio Martínez

COMPUTER ENGINEERING

UNIVERSITY OF GRANADA

Degree in Computer Engineering

OpenCV and Python application for automotive spotlight image processing

Javier Expósito Martínez

2022/2023

Tutor: Andrés María Roldán Aranda

Printed in Granada, July 2023.

"OpenCV and Python application for automotive spotlight image processing."

BACHELOR'S DEGREE IN COMPUTER ENGINEERING

Bachelor's Thesis

"OpenCV and Python application for automotive spotlight image processing."

ACADEMIC COURSE: 2023/2024

Javier Expósito Martínez

BACHELOR'S DEGREE IN
COMPUTER ENGINEERING

"OpenCV and Python application for automotive spotlight image processing."

AUTHOR:

Javier Expósito Martínez

SUPERVISED BY:

Prof. Andrés Roldán Aranda

DEPARTMENT:

Electronics and Computer Technologies

“OpenCV and Python application for automotive spotlight image processing.”

Javier Expósito Martínez

KEYWORDS:

Python, PyQt5,QtDesigner, DUT, EMC,ROI, GranaSAT, UGR, Functional Requirements, Non Functional Requirements, Reverse engineering, Arduino, Thread, Framerate, Keyframe, jinja2, OpenCV, PyArmor, CX-freeze, PyInstaller.

ABSTRACT:

The main objective of this Bachelor’s thesis is the development of an application that aims to detect fluctuations in the perceived lighting of a spotlight (either through a live recording camera or through video) following profiles in different ROIs generating HTML reports in base64 during the process. To achieve this, the application consists of three main functionalities:

The first one consists of controlling the on and off of car lights using one or two Arduinos that are configured to perform the task correctly.

The second one consists of performing a DUT test, through which we define the aforementioned ROIs.

The third one performs the test that controls the luminosity fluctuation and generates the report, called EMC

This Bachelor’s thesis is part of one of the TFGs offered by the Aerospace Electronics group, GranaSAT.

The final goal of this project is to develop a functional, complete, and commercial application that demonstrates the author’s knowledge and capabilities in analysis, design, and development.

“Aplicación OpenCV y Python para el procesado de imágenes de focos de automoción”

Javier Expósito Martínez

PALABRAS CLAVE:

Python, PyQt5, PyQt5,QtDesigner, DUT, EMC,ROI, GranaSAT, UGR, Functional Requirements, Non Functional Requirements, Reverse engineering, Arduino, Thread, Framerate, Keyframe, jinja2, OpenCV, PyArmor, CX-freeze, PyInstaller.

RESUMEN:

El objetivo principal de este Trabajo de Fin de grado es el desarrollo de una apliación que pretende detectar fluctuaciones en la iluminación percibida del foco(ya sea mediante una cámara que graba en vivo al foco o mediante vídeo) siguiendo unos perfiles en las diferentes ROIs generando en el proceso informes HTML en base64. Para cumplir esto, la aplicación consta de tres funcionalidades principales:

La primera consiste en el control del encendido y apagado de las luces de los atomóviles mediante uno o dos arduinos que están configurados para llevar a cabo correctamente dicha tarea.

La segunda consiste el realizar un test DUT mediante el cual definimos las ROIs mencionadas anteriormente.

La tercera realiza ahora si el test que controla la fluctición de luminosidad y genera el reporte, denomicado EMC

Este Trabajo de fin de Grado forma parte de uno de los TFGs ofertados por el grupo de Electrónica Aeroespacial,GranaSAT.

Este proyecto tiene como objetivo final el desarrollo de una aplicación funcional, completa y comercial que demuestre los conocimientos y capacidades de análisis, diseño y desarrollo del autor.

"Not giving up is my best weapon"

Asta
***Black clover*, 2017.**

Acknowledgements

When I was a child my mother always told me the saying "it is well-born to be grateful". [1], and although we rarely stop to be thankful for things, today is a good day to do so:

Thank you to my family, because even though you can't choose, I wouldn't have chosen another. Thank you for the patience, advice, support and love you have given me. I am proud to be your son and brother, thank you for everything.

Thank you, Marina, because you are the nicest person I have ever met and I hope you will never fail me. Thank you for understanding me and supporting me every day, because the good times, with you, are better and the bad times are not so bad by your side. You are the best person, girlfriend and friend I could have.

Ivan, being neighbors was the best thing that happened to me in a long time, I wish I had had a last summer to enjoy it as I would have liked.

Enrique and company, thanks for all the plans, it's incredible to have so many aches and pains when we only played videogames, but that's the bad thing about laughing so much.

Thanks to all my friends, for making the classes so incredible, because giving technical drawing class at home for the exam the next day was the best, I wish I could do it again. So many plans, so many crazy adventures, you are the best, no doubt.

To my roommates throughout the years, Dani, Fred, Carmen and Felipe, without a doubt, you are not only roommates, but also great friends that I will never forget. Because playing fornite or The Lost Vikings are unique experiences with you.

Inés, I met you doing a job for FR, and what a joy to see that you are not only the most responsible to work with, but you are also one of the best people I have ever met, grateful to be able to call you friend, my only regret, not being able to see each other more.

Finally, although I could be thanking for days, I want to thank GranaSAT for all the help and for giving me the opportunity to work on this project. And of course thanks to Andrés Roldán, for his attention, direction and dedication, as well as his treatment towards me.

Thank you all for being part of my life.

Agradecimientos

De pequeño mi madre siempre me decía el refrán "es de bien nacidos, ser agradecidos" [1], y aunque pocas veces nos paramos a agreder las cosas, hoy es un buen día para hacerlo:

Gracias a mi familia, porque aunque no se puede elegir, yo no habría escogido a otra. Gracias por la paciencia, consejos, apoyo y amor que me habeis dado. Estoy orgulloso de ser vuestro hijo y hermano, gracias por todo.

Gracias, Marina, porque eres la persona más buena que he conocido y espero que nunca me faltes. Gracias por comprenderme y apoyarme todos los dias, porque los momentos buenos, contigo, son mejores y los malos, no lo son tanto a tu lado. Eres la mejor persona, novia y amiga que podría tener.

Ivan, ser vecinos fue lo mejor que me pasó en mucho tiempo, ojala haber tenido un último verano para disfrutarlo como me hubiese gustado.

Enrique y compañía, gracias por todos los planes, es increible tener tantas agujetas cuando solo hemos jugado videojuegos, pero es lo malo de reir tanto.

Gracias a todos mis amigos, por hacer las clases tan increibles, porque dar clase de dibujo técnico en mi casa para el examen del día siguiente era lo mejor, ojala poder repertirlo. Tantos planes, tantas aventuras alocadas, sois los mejores, sin duda.

A mis compañeros de piso a lo largo de los años, Dani, Fred, Carmen y Felipe, sin duda alguna, no sois solo compañeros, sino tambien grandes amigos que jamás olvidaré. Porque jugar a fornite o The Lost Vikings son experiencias únicas con vosotros.

Inés, te conocí haciendo un trabajo para FR, y que alegría ver que no solo eres la más responsable para trabajar, sino que tambíen eres de las mejores personas que he conocido, agradecido de poder llamarte amiga, mi unico arrepentimiento, no poder vernos mas.

Por último, aunque podría estar agradeciendo durante días, dar las gracias a GranaSAT por toda la ayuda y por darme la oportunidad de trabajar en este proyecto. Y por supuesto gracias a Andrés Roldán, por su atención, dirección y dedicación, además de su trato hacia mi.

Muchas gracias a todos por formar parte de mi vida.

Contents

List of Figures

List of Tables

Glossary

A | C | D | E | F | G | J | K | N | O | P | Q | R | T | V

A

Apache is a widely used open-source web server software. It is one of the most popular web servers in the world and is known for its stability, security, and flexibility.

Arduino Is an open-source hardware and software platform designed for creating interactive projects and prototypes. It consists of a microcontroller board that can be programmed to control various electronic components and sensors. Arduino boards are widely used by hobbyists, students, and professionals for a wide range of applications, including robotics, home automation, art installations, and more..

C

CX-freeze is a library to creates standalone executables from Python scripts, with the same performance, is cross-platform and should work on any platform that Python itself works on.

D

DUT Refers to the device or component that is being tested in an experiment or evaluation.In our context, is a test performed on the spotlight or lighting system being evaluated in the application..

E

EMC Refers to the ability of electronic devices or systems to function properly in their electromagnetic environment without causing electromagnetic interference to other devices or systems. In our context , the EMC test is a test performed to assess the ability of the spotlight or lighting system to function properly in its electromagnetic environment .

F

Framerate number of individual frames or images displayed per second (fps). .

Functional Requirements refer to the specific tasks, actions, or behaviors that a software system or application must perform to fulfill its intended purpose. .

G

Gantt chart is a management tool in which a list of tasks is outlined in a timeline. Color bars represent working on tasks. The balloons indicate milestones, and dependencies between tasks are denoted with arrows. .

GranaSAT *Electronics Aerospace Group*. An academic project from the UGR. This organization has an electronics laboratory where students from different degrees and education levels develop multidisciplinary projects [2] .

J

jinja2 is a popular templating engine for Python. It is used for generating dynamic content and rendering templates in web applications, but can also be used for other text-based generation tasks. .

K

Keyframe frame that defines a significant point in a sequence of frames. .

N

Non Functional Requirements refer to the qualities or attributes that define how a software system should behave or perform. .

O

OpenCV OpenCV (Open Source Computer Vision) is an open-source library of computer vision and image processing functions. It provides a comprehensive set of tools and algorithms for analyzing, manipulating, and understanding visual data, including images and videos. .

P

PowerShell is a command-line shell and scripting language developed by Microsoft for automating administrative tasks and managing system configurations. .

PyArmor is a tool and library used for protecting Python applications by obfuscating and encrypting the source code. It is designed to add an additional layer of security to Python programs to prevent unauthorized access and Reverse engineering. .

PyInstaller bundles a Python application and all its dependencies into a single package. The user can run the packaged app without installing a Python interpreter or any modules. .

PyQt5 Python binding for the Qt framework, which is a powerful cross-platform application development framework widely used for developing graphical user interfaces (GUIs). PyQt5 allows developers to create desktop applications with a rich set of features and functionality. .

Python Is a high-level programming language known for its simplicity, readability, and versatility. .

Q

QtDesigner is a graphical user interface (GUI) design tool that is part of the Qt framework. It allows developers to create and design user interfaces for Qt-based applications visually, without the need to write code manually.. .

R

Reverse engineering Process through which it is attempted to understand through deductive reasoning how an already designed device, process or system works or was designed .

T

Thread Refers to a sequence of instructions that can be executed independently within a program. It is a lightweight unit of execution that allows concurrent or parallel processing of tasks within a single program. .

V

Valeo Relay Shield box composed of an Arduino map and relays..

Virtual Box is a virtualization software that allows you to create and run Virtual Machines on your computer. .

Virtual Environment A virtual environment is a Python environment such that the Python interpreter, libraries and scripts installed into it are isolated from those installed in other virtual environments, and (by default) any libraries installed. .

Acronyms

P | **R** | **U** | **V** | **W**

P

PHP Hypertext Preprocessor.

R

ROI Region Of Interest.

U

UGR University of Granada.

V

VM Virtual Machine.

W

WBS Work Breakdown Structure.

WSL Windows Subsystem for Linux.

Chapter 1

Introduction

This Bachelor Thesis shows the result of knowledge and skills acquired by the student in the Bachelor's Degree in Computer Engineering which has been tested during the development process of this project.

This document aims to reflect the engineering process behind the updating application development: since reverse engineering to know how it works until the creation development of new functionalities and improvements. The project's main goal is to update and improve an already created application which measures the fluctuations of automobile's spotlights Through the performance of an EMC test.

This Final Degree Project is carried out in collaboration with the academic project GranaSAT. This is an aerospace development group of the University of Granada (UGR), formed only by students from different fields of Engineering, such as Aerospace Engineering, Electronic Engineering, Computer Engineering or Telecommunications Engineering among others, under the supervision of Professor Dr. Andrés María Roldán Aranda.

1.1 Motivation

The main reason for choosing this project was my growing interest in the field of image treatment and processing arose as a result of taking electives subjects in my career which deal with this subject. Learning about the process of creating and developing an application also sparked my interest, as I consider that programming applications is one of the most important tasks expected of a programmer. However, I believe that the career does not realistically prepare us to carry out this work.

Another reason for my interest was the Python programming language. Not only is it comfortable to program with, but it is also a widely used language that is not outdated or obsolete, unlike other languages I have had to learn.

Therefore, I thought that my bachelor's thesis would not only help me to expand my current knowledge, but also to give it realistic functionality in the professional world. I am pleased to think that the development of this software has a practical application, since it is an application requested from a company. Thanks to this project, I will finish my degree knowing that I can develop applications.

1

Figure 1.1 – *The GranaSAT logo.*

1.2 Project goals and objectives

In this section outlines the main top-level non-technical goals of the project. Objectives listed in Table 1.1 must be understood as the author's expected results in academic and professional terms of the execution of this project.

Obj. Nº	Description
Obj. 1	Successfully migrate and upgrade the application given to the learner: from migrating libraries, to adding new functionalities, to essential functionality and performance improvements.
Obj. 2	Acquire familiarity, skill and confidence with the professional software for the logic and design of this application.
Obj. 3	To prove the capabilities of organizing and carrying out an engineering project.
Obj. 4	To document the entire process, which may be necessary during the development itself or useful for the future of the project.
Obj. 5	To demonstrate the knowledge acquired during the Bachelor studies in Computer Engineering, as well as the multidisciplinary abilities gathered during the development of this Thesis.
Obj. 6	To participate into the GranaSAT laboratory work environment to consolidate the training of the Bachelor's Degree.
Obj. 7	To successfully conclude the Bachelor's Degree with this Thesis.

Table 1.1 – *Top-level objectives of this Bachelor Thesis.*

1.3 Project structure

This document is divided into seven chapters and eight addenda. The chapters progressively expound all the stages of the development of the proposed device, including the analysis of signals and of the competing products, tackle specification, design, fabrication and validation tasks; and finalizes with the successful completion of the product.

The chapters included in this report are:

1. **Chapter 1: Introduction.** This first chapter is intended as groundwork to the subject at hand, and to show the objectives and motivations of this project. It includes some definitions, the state of the art and an introduction to the engineering methodology followed throughout this project.

2. **Chapter 2:Analysis.** Section presenting the different requirements necessary to achieve the project's purpose, the analysis and organization of the project.
3. **Chapter 3: Reverse Engineering.**

   This chapter offers a process of Reverse Engineering in order to gain understanding of the advantages and limitations of its technology so we can synthesize our own and superior product.
4. **Chapter 4: System design.** This chapter describes all the aspects of the system design.
5. **Chapter 5: Implementation and configuration.**

   Section showing what aplication is,how it works, its installation and configuration along with the chosen devices.
6. **Chapter 6: Testing and validation.** This sixth chapter details the process and testing of the application and verification of the systems' correct operation.
7. **Chapter 7: Conclusion and future lines.** Lastly, the final chapter brings to an end the main contents of this Bachelor's Thesis, and establishes some future lines of work that have emerged naturally during this long development process.

On the other hand, the addenda is divided in:

A. **Appendix A: How to install application.**

B. **Appendix B: How to install and configure Valeo Relay Shields.**

C. **Appendix C: How to install and configure a Local Server.**

D. **Appendix D: Detailed application structure.**

E. **Appendix E: Graphical visualization of improvements**

Chapter 2

Analysis

The aim of this section is to present in its entirety the list of requirements that has been elaborated during the numerous interviews with the client. The main purpose of this list is to define what updates and improvements needs our application.
The client explained his ideas in mind which he wanted, to improve the application.This is followed by the analysis and organization of the project as shown below.

2.1 Functional Requirements

Ref.	Description
RF. 1	The software must allow the user to create a new DUT Test.
RF. 2	The software must allow the user to create a new EMC Test.
RF. 3	The software must allow the user to open a DUT file.
RF. 4	The software must allow the user to open an EMC file.
RF. 5	The software must allow the user to clone an EMC file.
RF. 6	The software must allow the user to open a recent DUT or EMC file.
RF. 7	The software must allow the user to configure car's spotlights using one or two Valeo Relay Shields, and save this light's configuration.
RF. 8	The software must allow the user to change between white and black theme.
RF. 9	The software must allow the user check about additional information.
RF. 10	The software must allow the user check updates and update the application if it is necessary.
RF. 11	The software must allow the user to select one of several languages: english, spanish and french
RF. 12	The software must allow the user to create a DUT file since EMC test
RF. 13	The software must allow the user to create a light configuration file since EMC test
RF. 14	The software must allow the user to generate a report.

Table 2.1 – *Functional Requirements*

2.2 Non Functional Requirements

Ref.	Description
NFR. 1	The software must be able to detect when an Valeo Relay Shield is disconnected, at any time.
NFR. 2	The software must be able to reconnect an Valeo Relay Shield for a short period of time since it was disconnected.
NFR. 3	The software must be able to work correctly without an Valeo Relay Shield connected.
NFR. 4	The software must be able to save the last configuration which it had when it was closed and starts with this.
NFR. 5	All software's interfaces must be responsives.
NFR. 6	The software must has a status bar which reports user's operations and application's status.
NFR. 7	The software must be save datas in xml files, unlike before, it was done in txt format.
NFR. 8	In DUT and EMC tests' forms, done's bottoms must be disable when user delete any required field. Done's bottoms only must be enabled when all required fields are filled.
NFR. 9	In DUT test's forms, when the user selects a car spotlight, it should highlights.
NFR. 10	When the user open DUT of EMC file, in DUT and EMC tests' forms, must appear a new field which contains the video's path what was used in test.
NFR. 11	When the user open DUT file,crop points must be show in window where the user crop the image.
NFR. 12	When the user open DUT file, ROIs must be show in crop window and create the ROIs.
NFR. 13	In DUT Test, in windows where the user crop the image and create the ROIs, the software must be able to separate the camera view and zoom view, allowing to work with two or even three screens.
NFR. 14	In EMC Test,when the analysis starts, if the user select a ROI, this must be highlight in Keyframe view.
NFR. 15	In EMC Test,when the analysis starts, if the user select an alert, this must be shown.
NFR. 16	The software must be able to add a graph which shows the lumination's difference between Keyframe and video for each ROI in the generated report.
NFR. 17	The software must read and process the videos efficiently and quickly.

Table 2.2 – *Non Functional Requirements*

2.3 Analysis

The first thought was about the programming language, continue with Python or look for a better alternative. And after analyzing several programming languages such as C++ or Java, which are more resource efficient than Python, it was decided to continue with this language because of the many advantages it had. Not only its ease of use and ease of programming, in addition to a very high number of useful libraries, but also because it already had very interesting tools from the previous version as PyQt5 for the interface or CX-freeze to create the executable, which meant a shorter development time of the application and cost savings (the latter to give maximum realism to the project).

Knowing what language to use, some useful tools and the client's requirements, it is time to proceed to organize the work throughout the course, as shown in the next section.(Project tasks and organization)

2.4 Project tasks and organization

A Work Breakdown Structure is a hierarchical decomposition of the tasks of a project in order to accomplish the desired objectives. In figure 2.1 the Work Breakdown Structure of the project is presented. This schema is product of the definition of system requirements in Chapter 2: Analysis and iteration with the design process elaborated on Chapter 5: System design.

On the other hand, the figure 2.2 shows the Gantt chart of the project's development process. A Gantt chart is a management tool in which a list of tasks is outlined in a timeline. Color bars represent working on tasks. The balloons indicate milestones, and dependencies between tasks are denoted with arrows.

It is important to point out that besides the tasks, meetings are also included since they were a fundamental part of the development process. These meetings not only served as a form of reviewing results and controlling the development, but they were essential to define the project requirements (elaborated in chapter 2).

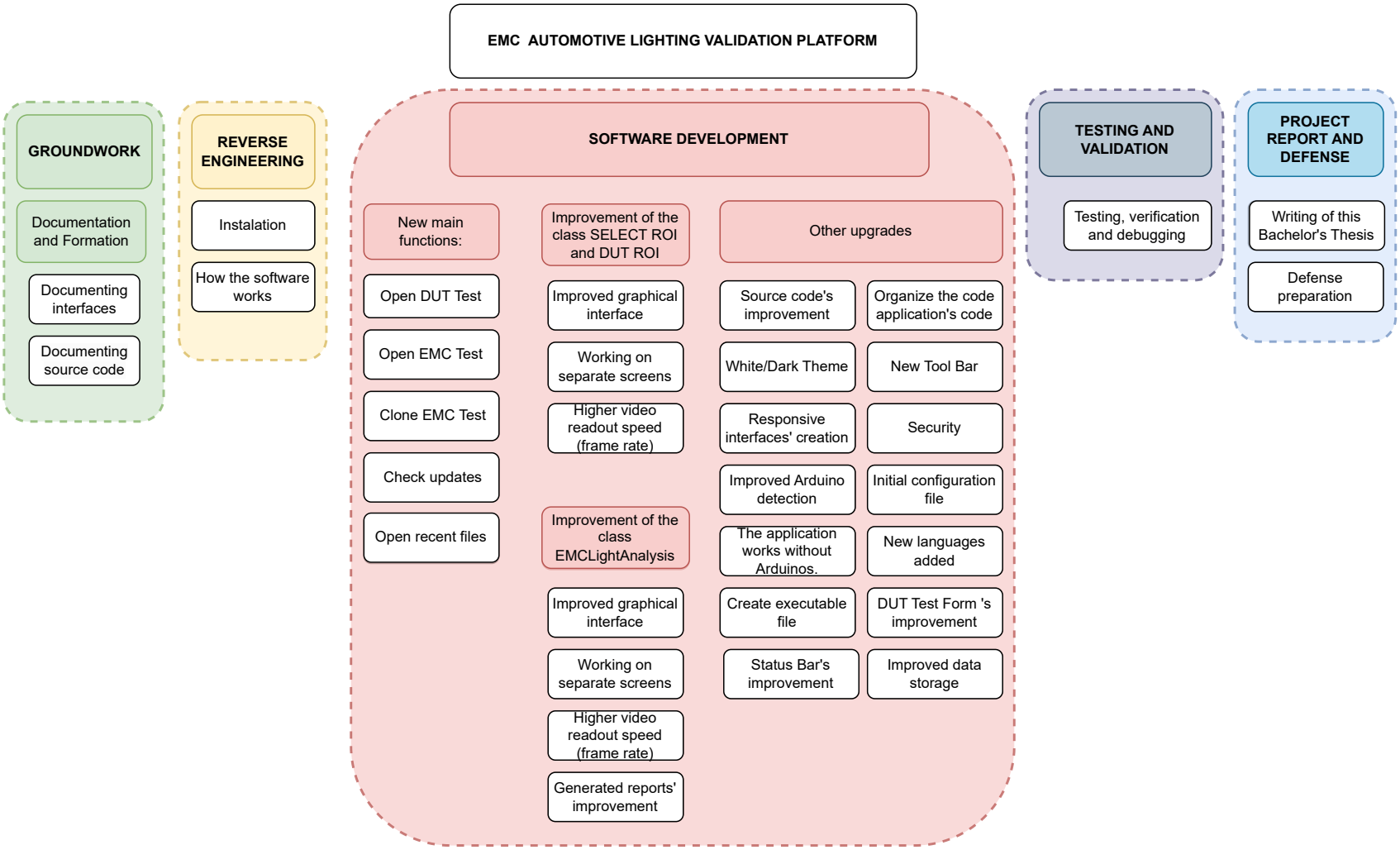
EMC AUTOMOTIVE LIGHTING VALIDATION PLATFORM
GROUNDWORK
Documentation and Formation
Documenting interfaces
Documenting source code
REVERSE ENGINEERING
Instalation
How the software works
SOFTWARE DEVELOPMENT
New main functions:
Open DUT Test
Open EMC Test
Clone EMC Test
Check updates
Open recent files
Improvement of the class SELECT ROI and DUT ROI
Improved graphical interface
Working on separate screens
Higher video readout speed (frame rate)
Improvement of the class EMCLightAnalysis
Improved graphical interface
Working on separate screens
Higher video readout speed (frame rate)
Generated reports' improvement
Other upgrades
Source code's improvement
Organize the code application's code
White/Dark Theme
New Tool Bar
Responsive interfaces' creation
Security
Improved Arduino detection
Initial configuration file
The application works without Arduinos.
New languages added
Create executable file
DUT Test Form 's improvement
Status Bar's improvement
Improved data storage
TESTING AND VALIDATION
Testing, verification and debugging
PROJECT REPORT AND DEFENSE
Writing of this Bachelor's Thesis
Defense preparation

Figure 2.1

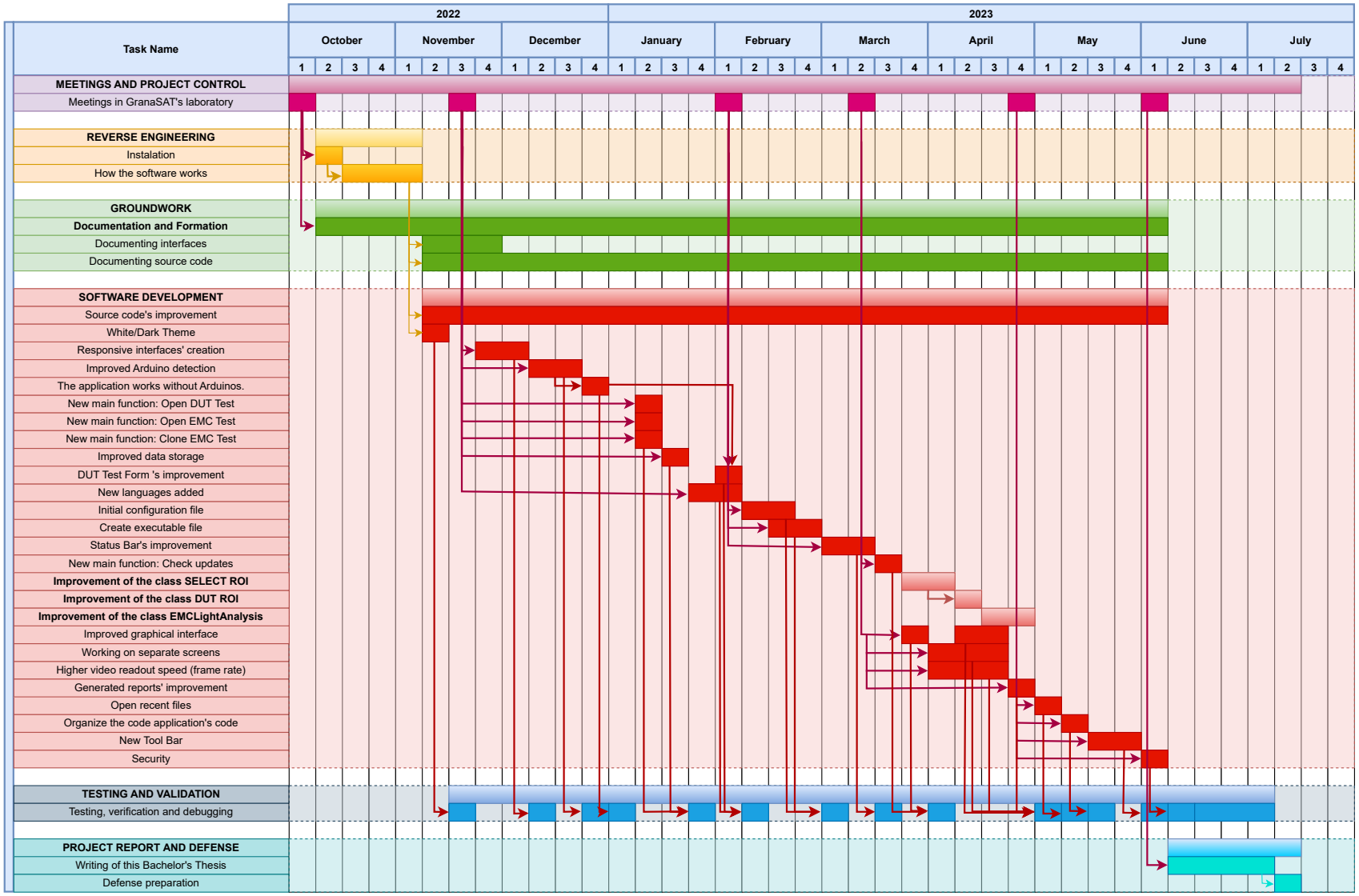

Figure 2.2 – *Gantt chart of the project.*

Chapter 3

Reverse engineering

3.1 What is reverse engineering?

Reverse engineering: Process through which it is attempted to understand through deductive reasoning how an already designed device, process or system works or was designed.

The limited usefulness of the comments in the source code of the application and the non-existent documentation about its operation has made Reverse engineering plays a fundamental role in the development of this application. It must be remembered that the development of this application is based on an older version of the same and I had not previously worked on the application.

3.2 Original software's analysis

The first step was understand how to install the application, because the software was a .zip file which contained a huge files. Some of them showed errors, empty folders, junk files, etc.

The only useful file for figuring out how to install the software was the requirements.txt file. This file contains the libraries and the version of these libraries that are used by the application. So after finding this file, I created a Virtual Environment and installed the libraries.

The installation process also gave error due to the incompatibility of versions of some libraries, but after updating them the software started.

After starting the software, I started by analyzing the files and the source code. To begin with there were three main.py files, which was not logical since there should be only one because this file (main.py) is the one that starts the whole application.

After some analysis, one of the main.py files was discarded and I proceeded to analyze the other two more deeply. For this, I created two simple diagrams that show the software's flow, put another way, classes' calls to others.

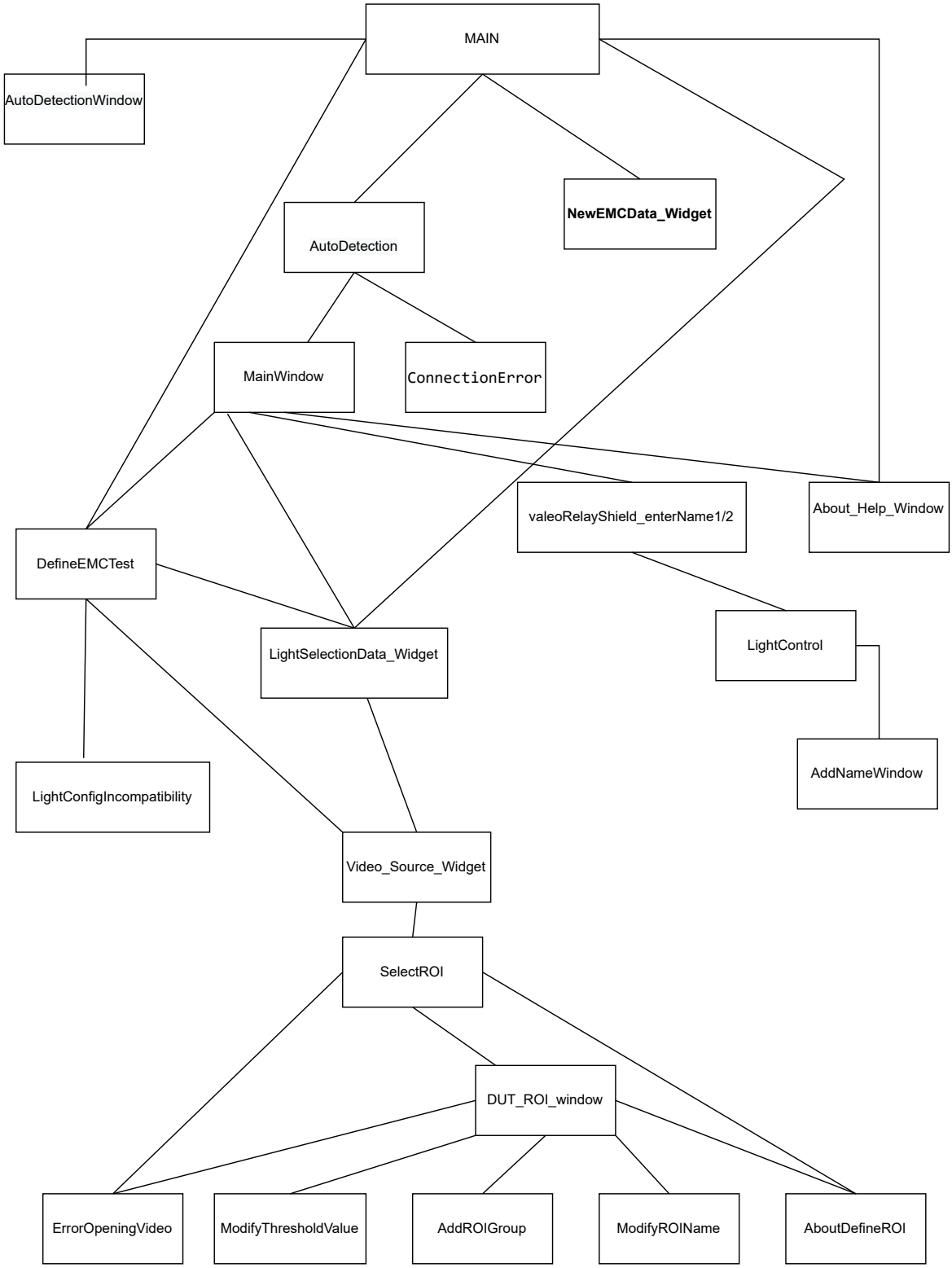

Figure 3.1 – *The class map of first main.py*

Figure 3.2 – *The class map of second main.py*

After seeing both diagrams and from my point of view, I considered that the main.py more suitable was the one whose diagram is more organized, the one in the second diagram to be exact.

3.3 Main and secondary functions' analysis

Through the exhaustive analysis of the source code and test the application, I identified several functions, which I will divide into :

- Primary:
  - Open Light Controller: To perform the configuration of the lights by means of the Valeo Relay Shields.
  - Create new DUT Test: To create a new DUT Test.
  - Create new EMC test: To create a new EMC Test.
  - Open new DUT Test: To open a saved DUT file.
  - Open new EMC Test: To open a saved EMC file.
- Secondary:
  - Clone EMC Test: To clone EMC Test saved file.
  - Select Theme:To select theme (light or dark).
  - Help about: To show contact window.
  - Check Updates: To check if there is a new version and if so, to update the software.

3.4 How the application should be works?

After identifying the main and secondary functions, I proceeded to relate the different classes to the previously mentioned functionalities, and show how the whole application works.

The first class is **Main**, which simply starts the application. Then we have the classes **AutoDetectionWindow** that shows the detected Valeo Relay Shields and **AutoDetection** that is in charge of the detection of these.

The next classes to be called are: **connectionError**, to show an error window if there are no Valeo Relay Shields connected or if they are disconnected at any time and the **mainWindow** class that shows the main window of the application.

In the main window, there are three main functions that call other classes:

- **Open Light Controller:** To set up a save file with the status (on or off) of the car headlights. The process is as follows: The class **ValeoRelayShieldName1** or **ValeoRelayShieldName2** is called depending on the number of connected Valeo Relay Shields. These classes only show a window to name the tabs of the next window that will appear, namely the lights configuration window, the **LightControl** class. In addition this class calls the **AddNameWindow class**, which displays a window to add another type of headlight that the user wants.

Figure 3.3 – *Create Light Configuration Process*

- **Create/Open DUT Test:** Subjects a car headlight to a DUT test. To do this, the **LightSelectionDataWinget** class is called, which displays a window with a form about the car headlight data and the test.

  After filling out the form completely, **VideoSourceWidget** displays a window to choose between using a camera to test it or an already recorded video.

  SelectROI then displays a window that allows you to crop the video using a view of the video, and an auxiliary view of the video but with a zoom factor applied. Then after doing this, **DutROI** is in charge of drawing the ROIS showing a window similar to SelectROI, being able to change its color, threshold value, color, add custom ROI groups... by means of the classes **ModifyThresholdValue**, **AddROIGroup** and **ModifyROIName**. Additionally **ErrorOpeningVideo** will show if there has been any error when opening the video and **AboutDefineROI** will show a help window on how to define an ROI.

Figure 3.4 – *Create DUT Test Process*

- **Create/Open EMC Test:** Subjects a car headlight to a EMC test. This task is performed by the following classes: **DefineEMCTest** displays a window with a form to be filled in with all necessary data. It also allows to create/open a DUT Test, and to create/open an automotive light configuration. The class **lightConfigIncompatibility** will show an error if we have only one Valeo Relay Shield connected and we try to open a light configuration file that uses two Valeo Relay Shields.Similarly, in DUT Test, the **VideoSourceWidget** class is called, and after selecting an option, the **EMCLightAnalysis** class will perform the EMC test and generate the report.

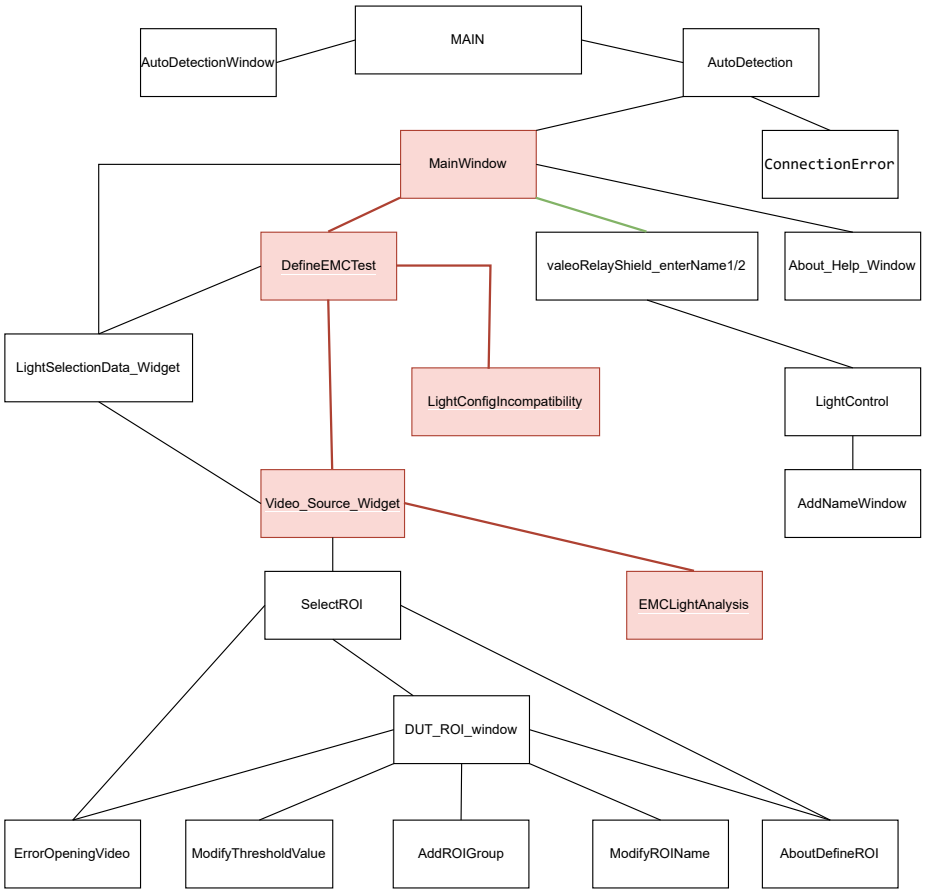

Figure 3.5 – *Create EMC Test Process*

In addition the **AboutHelpWindow** class displays a contact window and information about GranaSAT and the application developer.

3.5 How did the application work?

After explaining how the application should work, I will now describe how the original application functioned before working on it.

- **Select Window > WhiteTheme** caused the application to abort.
- **Help > check updates** did not work as it was not implemented.
- **Tools > OpenLightController** and **File > New EMC test** did not work without the Valeo Relay Shield connected, resulting in the application aborting.
- **File > Open > Dut, File > Open > EMC, and File > Clone > EMC** did not work as they were not implemented.

Upon connecting the Valeo Relay Shield, the functionalities **Tools > OpenLightController** and **File > New EMC test** did work. However, the latter, **File > New EMC test** had issues such as opening an EMC file causing the application to abort, and **"create new Light Configuration"** did not work as it was not implemented. Additionally, some interfaces, due to their non-responsive nature, had display problems.

3.6 Conclusion

To summarize, I will show the whole process graphically:

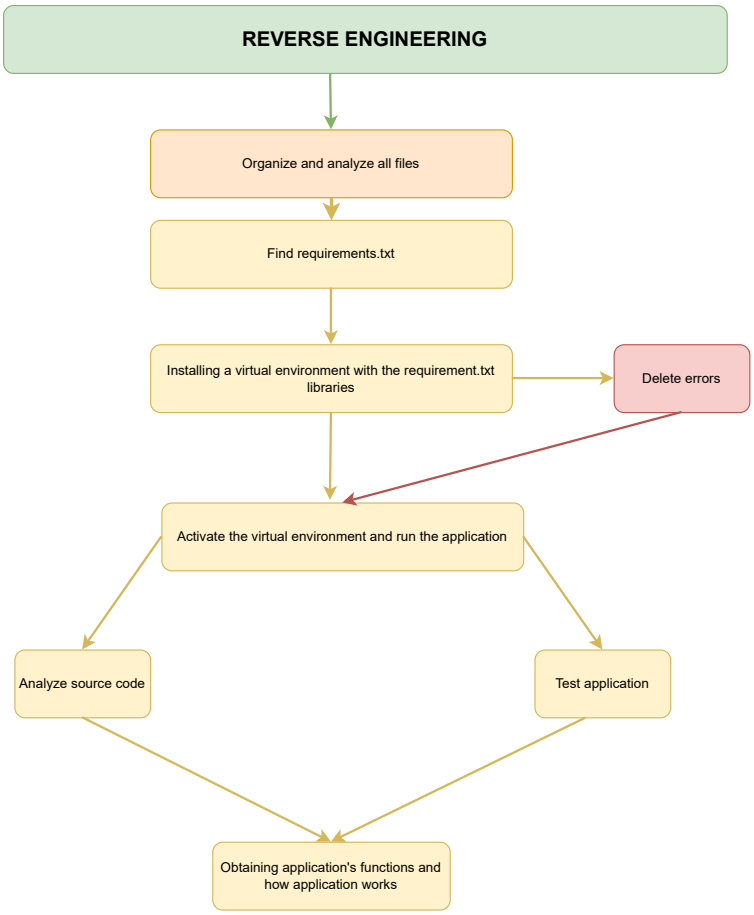

Figure 3.6 – *Reverse engineering Process*

Chapter 4

System design

In the following section, once all the comparisons and decisions from the analysis phase have been made, I'm going to explain in more detail the design of the application, as well as its directories hierarchy.

4.1 Applications' directories hierarchy

Starting with the directories' hierarchy of the application, it is organized into different folders and files. To summarize, the application is divided into the following:

- Files:
  - **main.py**, which starts the application.
  - **configuration.xml**, which contains the initial configuration of the application.
- Secondary:
  - **reports**, which contains the HTML template for generating reports.
  - **images**, which contains various images related to the application and others.
  - **languages**, which contains different language files.
  - **datas**, which stores saved files and videos.
  - **gui**, which contains all files related to the graphical user interface.
  - **src**, which holds all the application code that is used by main.py (this folder is not present in the client-installed application for security reasons).

To display the structure more clearly, here it is shown graphically:

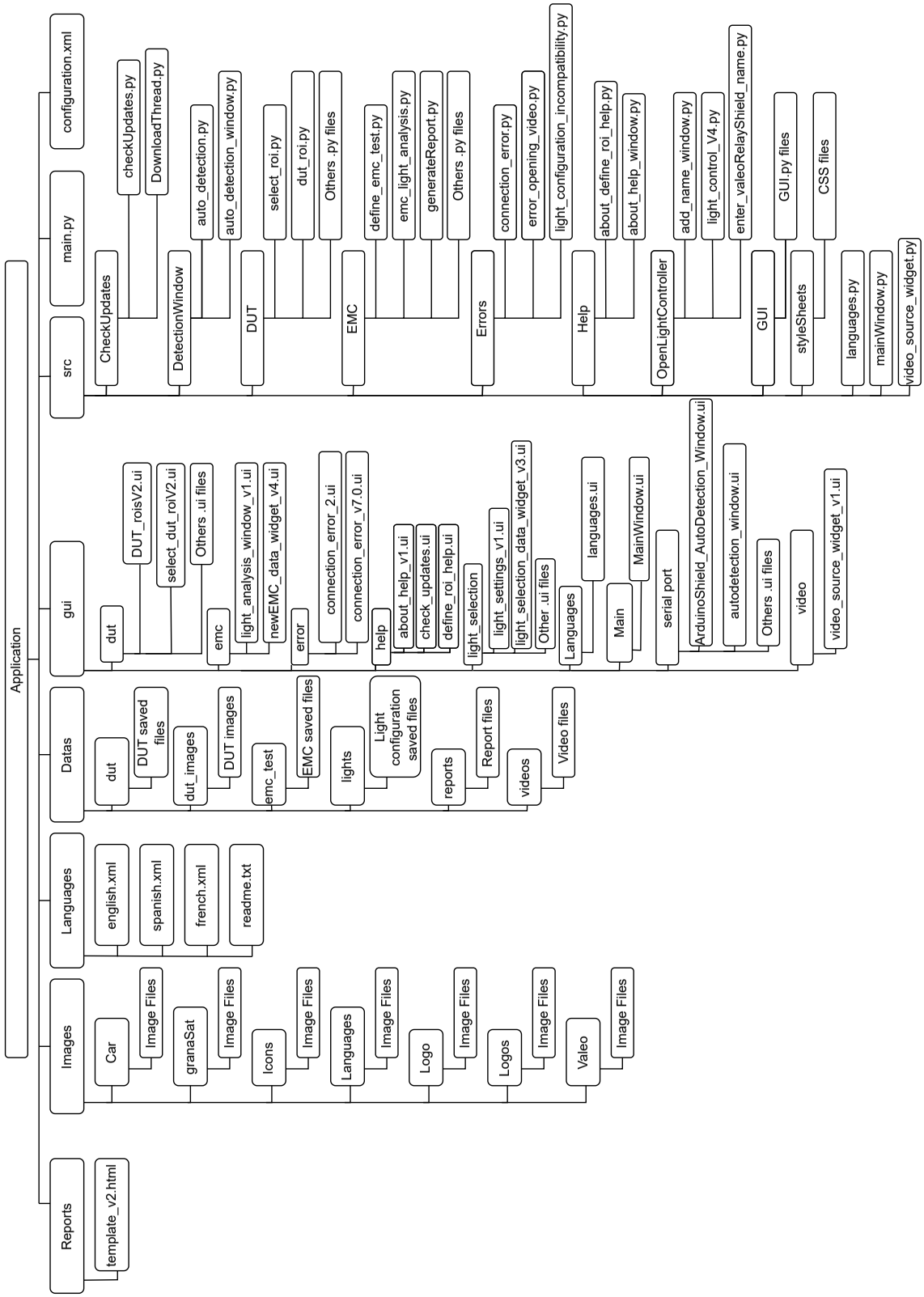

Figure 4.1 – *Application's Directorieshierarchy*

4.2 Applications' structure

For the development of the project, figure 4.2 has been followed, where on the right side, the connection of the application with the GranaSAT server is shown, to update the application version if necessary (whenever there is an internet connection). On the left side, the execution of DUT and EMC tests is shown, either through a video file or by using a camera connected to the application. At the same time, the Valeo Relay Shield hardware is connected to the computer to control the lighting of the spotlights.

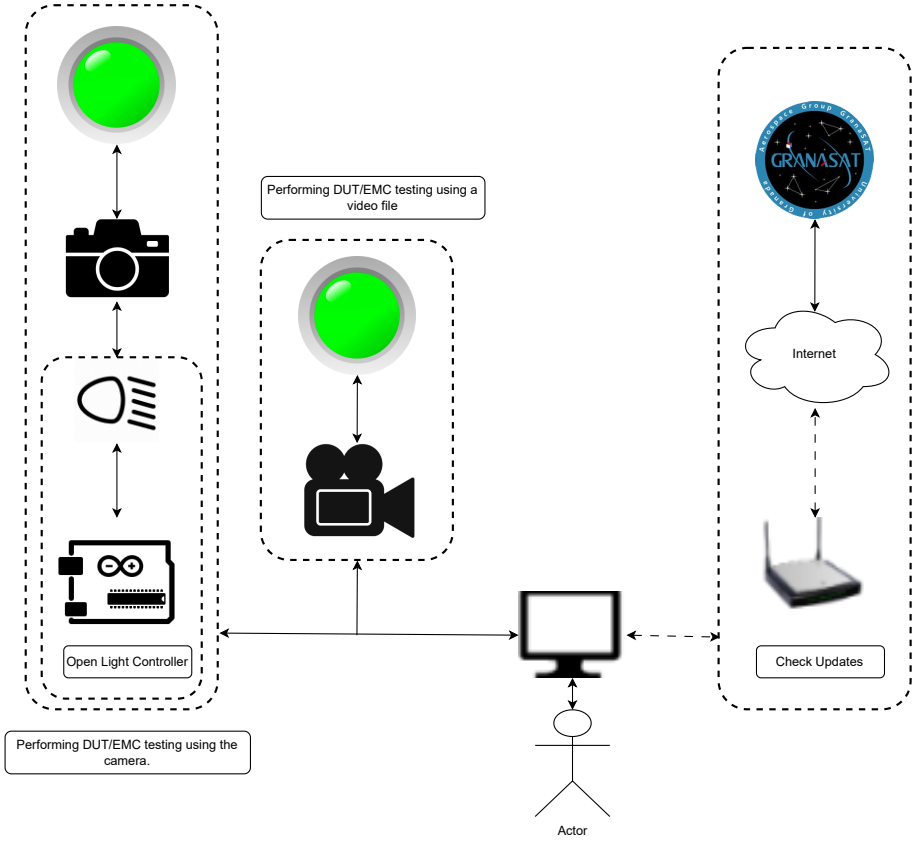

Figure 4.2 – *Application's Structure*

4.3 Class diagram

To conclude this chapter, it proceeds to show the complete class diagram of the application divided into parts due to its extension and high complexity, thus showing the final structure of the application. Below I only put the main diagram showing the connection between classes. The class diagrams detailing each part of the application are in the appendix(Appendix D: Detailed application structure.).

4.3.1 Application's class diagram

It proceeds to show two diagrams where all the classes are shown, joined by arrows to show the class calls and the communication of some classes with others.

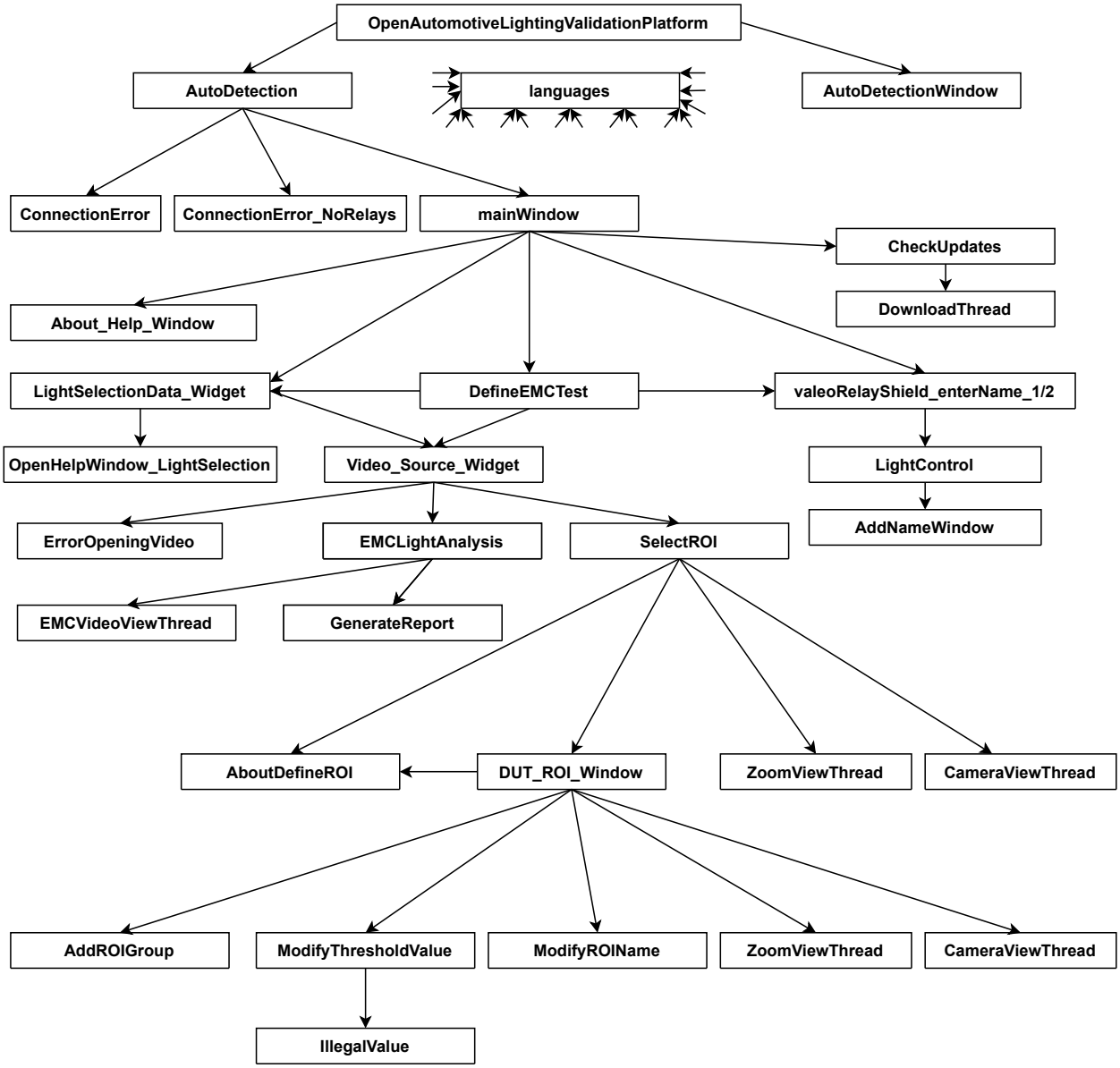

Figure 4.3 – *Application's diagram class*

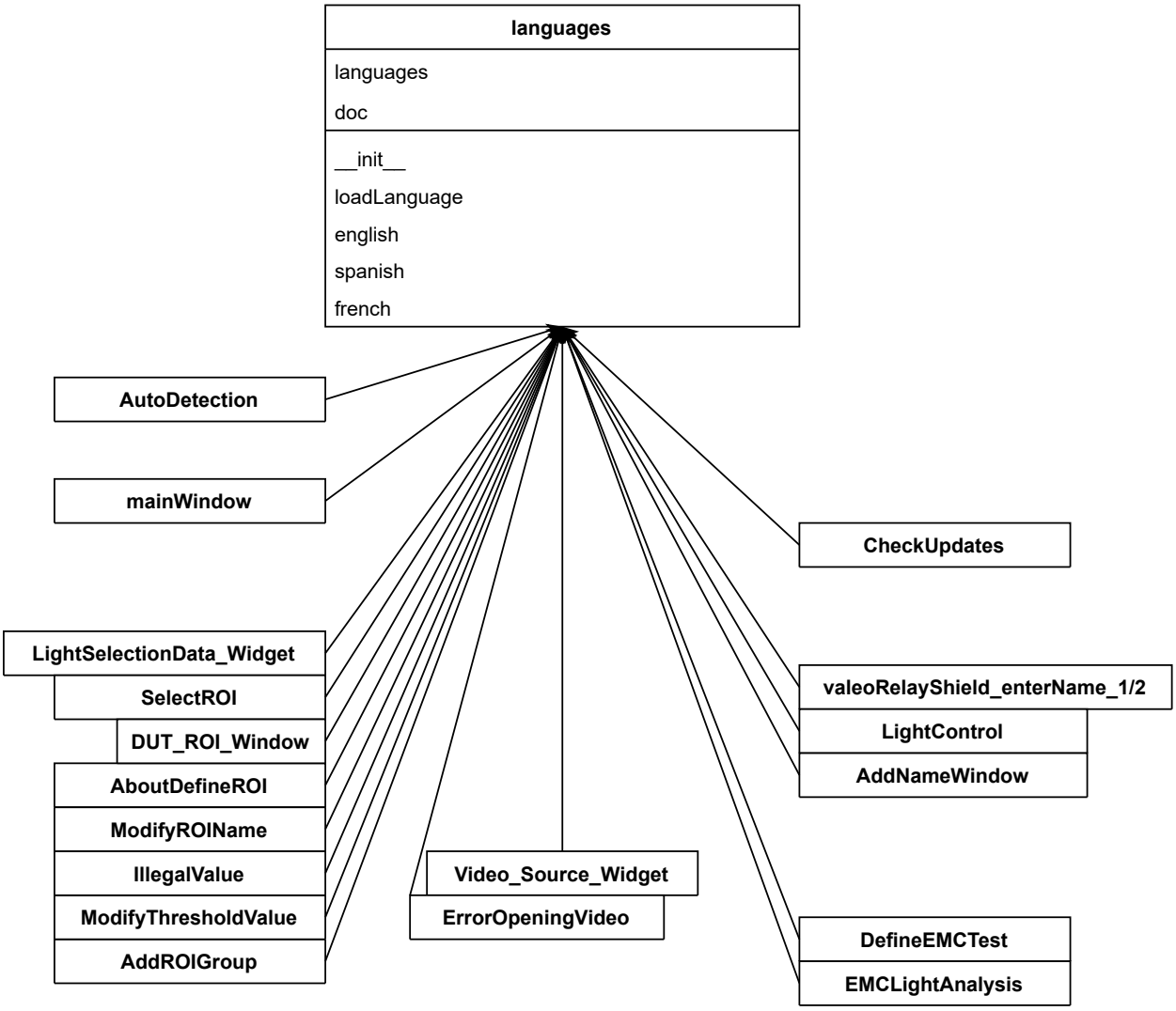

Figure 4.4 – *Application's diagram class2*

Chapter 5

Implementation and configuration

In this chapter, I will talk in detail about the entire configuration of the application, as well as the modifications and upgrades made to the project assigned to me in September.

5.1 Application's upgrades

Once the Valeo Relay Shield was configured and connected, after the Reverse engineering process, It started to fix the bugs in the application and to make the upgrades requested by the customer. In this section, therefore, the upgrades made to the application will be explained, and these changes will be visually displayed in the section Appendix E: Graphical visualization of upgrades

5.1.1 Responsives interfaces

The application used interfaces that could not be resized, causing problems when the monitor size was not large enough to display the entire window. Therefore, the first improvement to be implemented was the creation of responsive interfaces and in some of them, even the possibility of scrolling to display them completely.There were also some other changes, especially in the main window.

On the main window interface, the buttons that contained the main actions performed by the application were removed, as well as some other unusable elements. An error in the AutoDetection Window class was also eliminated., which consisted in creating a second mainWindow just before displaying it, thus avoiding the display of messages such as whether the Valeo Relay Shield were connected or not.

For the rest of the interfaces as well as for the main window, the resizeWindow() method that set a fixed window size was eliminated.

To create the responsive interfaces, we used QtDesigner, a tool that comes with the installation of the application in developer mode. To make an interface responsive, the grid layout, horizontal layout and vertical layout elements are used, in addition to giving a minimum and maximum size to each element.

In addition, as mentioned above, scrolling capability has also been added to some interfaces, specifically the two interfaces that are the forms for the DUT and EMC tests. This possibility is done by adding the scroll area element, and inside the rest of the elements so that if the size of the elements is bigger than the size of the window, the scroll bars appear. A new field was also added to show in case of opening a file, which video file was used to make the video.

The images below are examples of the difference between before and after this improvement.

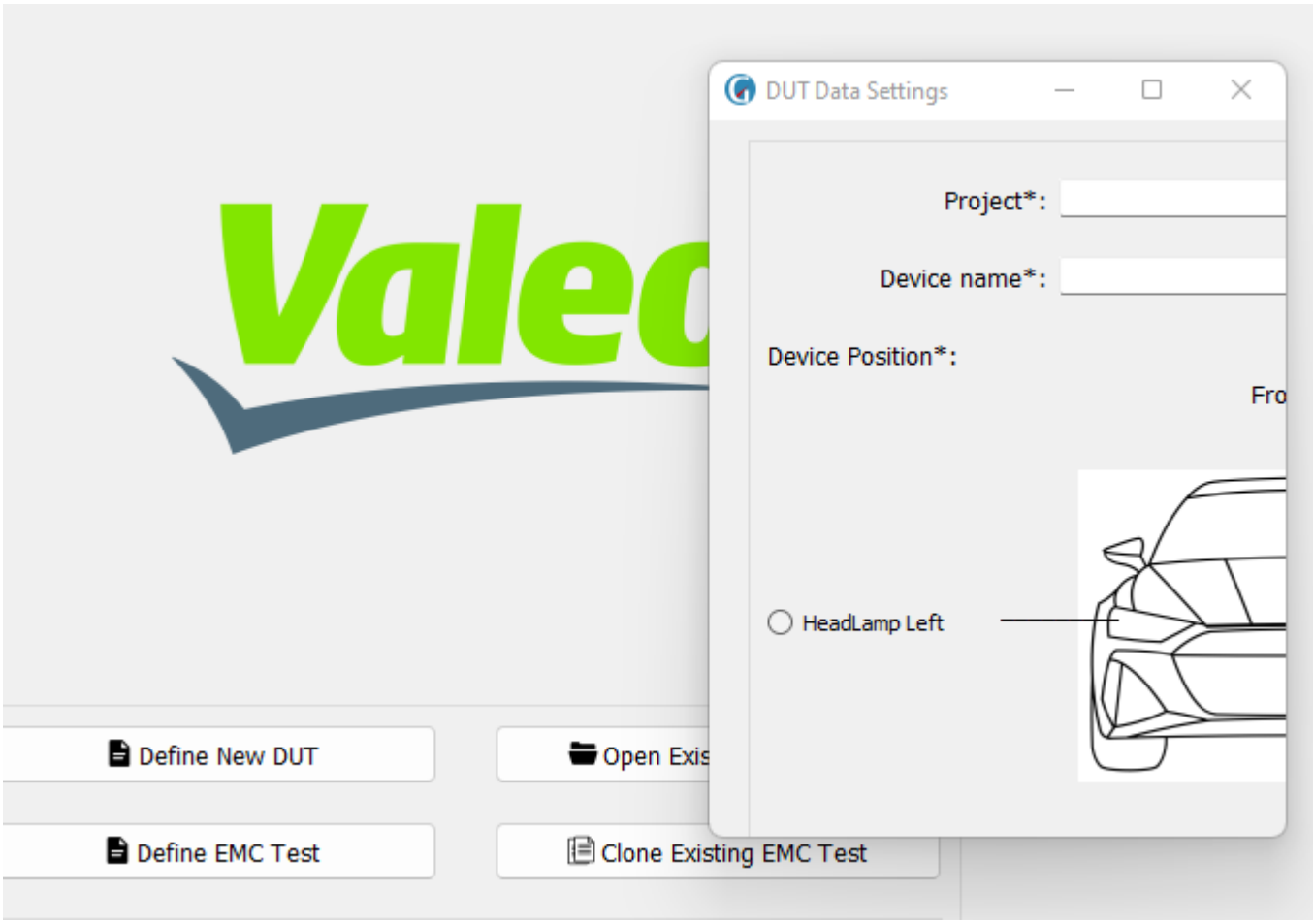

Figure 5.1 – *Before*

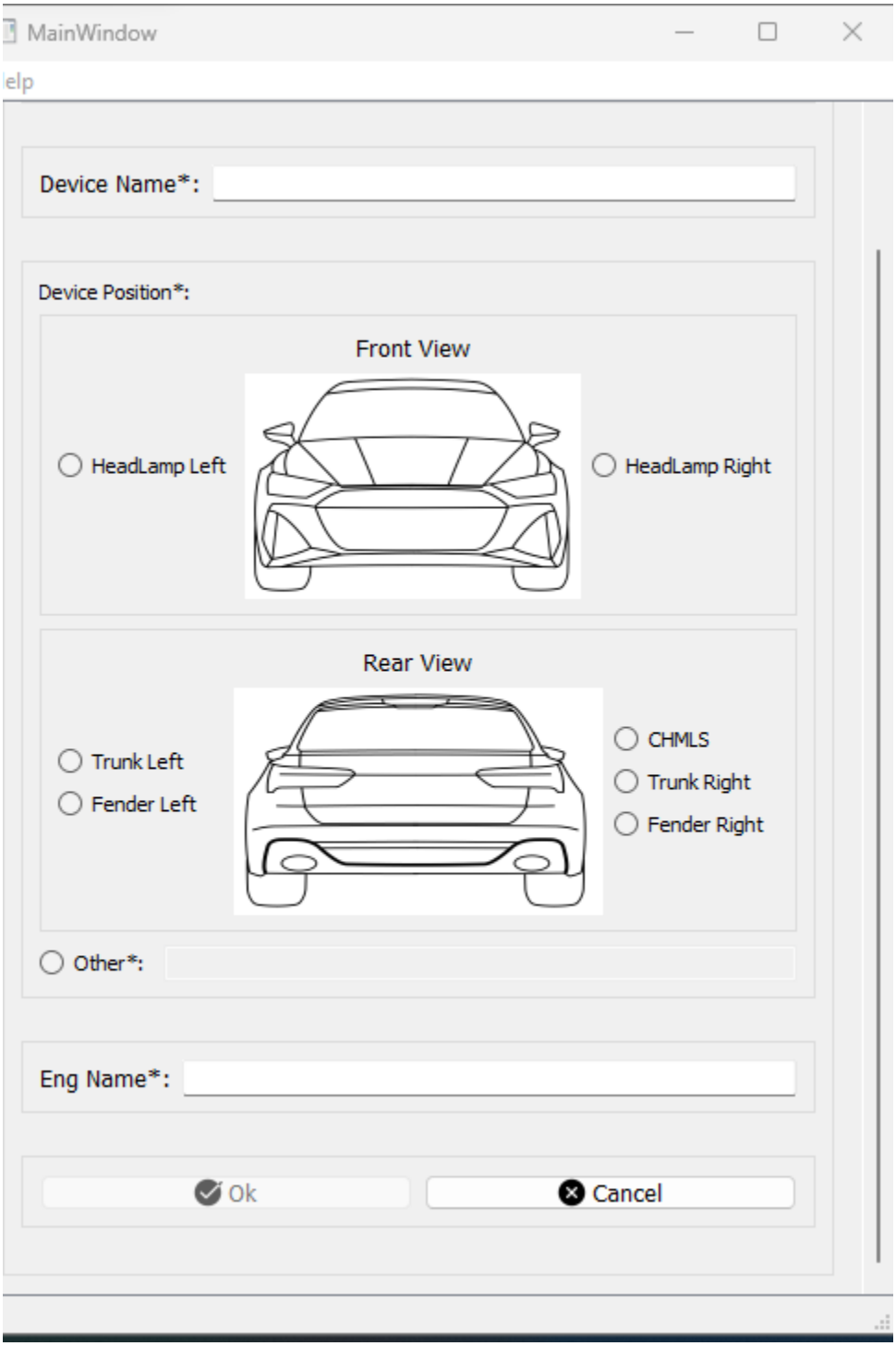

Figure 5.2 – *After*

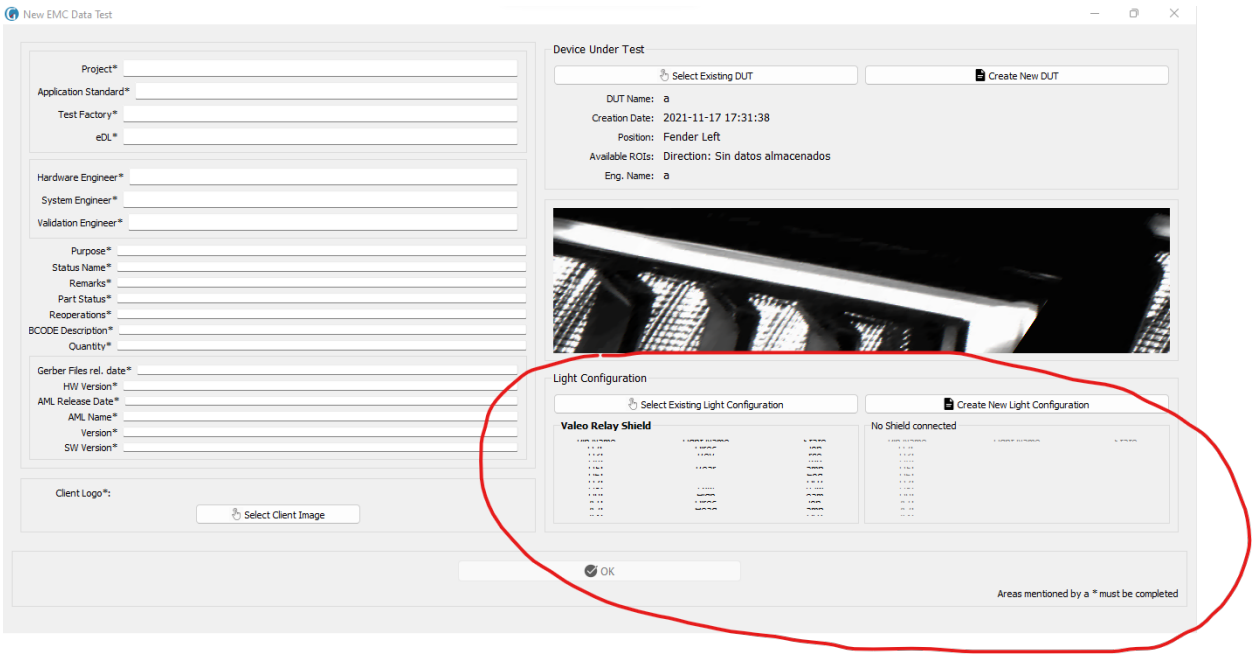

Figure 5.3 – *Before*

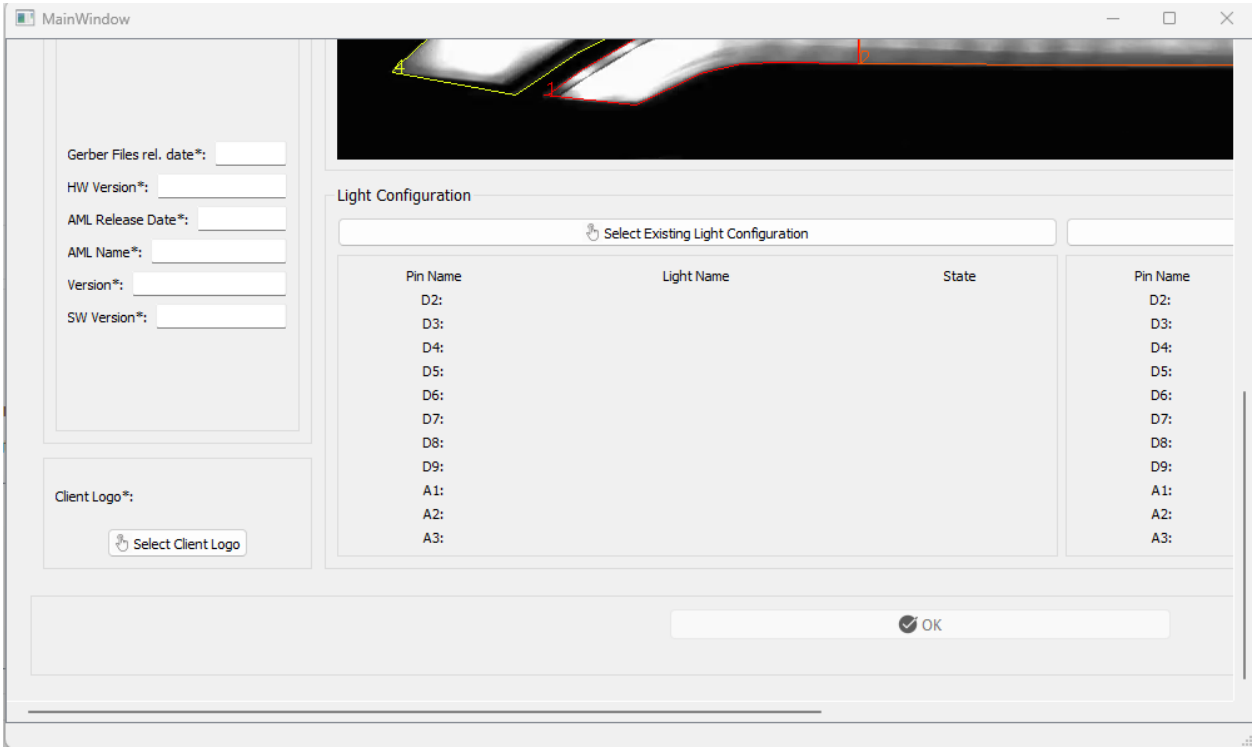

Figure 5.4 – *After*

5.1.2 Valeo Relay Shield detection and source code improvement

The next step was to improve the detection of the Valeo Relay Shields. All the code was restructured and separated into methods, easier to understand and control. In addition, errors such as the same name of two different variables or hidden windows were corrected.

In addition, an improvement was implemented so that the Valeo Relay Shield can be detected a few seconds after the application starts or when it is disconnected, unlike before, when the Valeo Relay Shield had to be connected before the application started.

5.1.3 The application works without Valeo Relay Shields

The Valeo Relay Shields in the application are used to control the lights of the car headlights, and thus record a video testing these. However, if we already have the video recorded, there is no need to have the Valeo Relay Shields connected and they are completely useless. So why do they always have to be connected if in most situations they are not even used?

After this improvement, the application works with or without connected Valeo Relay Shields. Although it warns you with alerts that the Valeo Relay Shields have not been connected, you can still work perfectly with the application.

5.1.4 Source code upgrades

In the same way that the code was improved in Valeo Relay Shields detection, other important classes were redesigned by adding and removing attributes, adding new methods to simplify and control the code, etc.

5.1.5 New functionality: open a DUT test

The possibility to open a DUT test, store the data, and perform the test without any issues has been added to the application. Changes were made to resolve crashes and forced closures in Select ROI and DUT ROI. Code was added to ensure that the camera does not remain open when closing the window unexpectedly after selecting the camera option. Additionally, a new menu with actions such as tools, zoom, help, etc. has been added.

5.1.6 New functionality: open and clone a EMC test

The possibility to open a EMC test, store the data, and perform the test without any issues has been added to the application. The entire logic behind an EMC analysis has been developed because the class was empty with no functionality apart from displaying the window. Therefore, not only has the entire class been programmed to perform the necessary calculations, but also a completely new class called "generateReport" has been created to generate a report based on the results.In addition, a menu with actions such as "revert alerts," "generate report," etc., has been added.

5.1.7 Data storage

The old version of the application stored data in files with a .txt format. This format, besides being very primitive, presents a problem: it is not scalable and can easily lead to errors. For example, if you change a line by pressing enter, it could cause a failure. This makes it tedious to modify the file in the future if additional data needs to be stored at the beginning, and it can generate errors. Therefore, the data is now

stored in files with the .xml format, where we access the content using tags. This is a simple, easy, and scalable solution.

The syntax of an XML file is simple:

```
<?xml version='1.0' encoding='utf-8'?>
<root-tag>
    <tag1>Data saved 1</tag1>
    <tag2>Data saved 2</tag2>
    <tag...>Data saved ...</tag...>
</root-tag>
```

Figure 5.5 – *XML syntax*

To read these files, it use "from xml.dom import minidom," which allows read the data like the following picture:

```
def update_DUT_data(self):
    if(self.fileOpen(self.dut_file_name)):
        doc=minidom.parse(self.dut_file_name)
        self.date = doc.getElementsByTagName("Date")[0].firstChild.data

        # Read DUT datas
        self.projectName = doc.getElementsByTagName("Project")[0].firstChild.data
        self.deviceName = doc.getElementsByTagName("DeviceName")[0].firstChild.data
        self.devicePosition = doc.getElementsByTagName("DevicePosition")[0].firstChild.data
        self.engName = doc.getElementsByTagName("EngName")[0].firstChild.data
        self.VideoNameDUT=doc.getElementsByTagName("Video")[0].firstChild.data
```

Figure 5.6 – *Read XML document*

5.1.8 DUT test and EMC test forms' improvement

Both for the DUT test form and the EMC test, in order to press the "Done" button, all fields must be filled out completely. If any field is deleted, the "Done" button becomes disabled (previously this was not the case, allowing empty fields to be submitted).

Additionally, for the DUT test, now when you select a focus, it turns a light red color, making it easier for the client to know which headlamp or rearlamp is being selected. There is also a help menu that displays two images, where the focuses are connected to their names with arrows.

5.1.9 New languages in application

A new menu has been added, with a new action, "Languages -> Select languages," which displays a window where we can select the language of the application: English, French, and Spanish.

For this enhancement, three options were considered:

1. **Using .txt files** to store words in different languages and reading the appropriate file based on the selected language. However, this solution was quickly dismissed because it is not scalable and prone to many errors, as mentioned in the section discussing upgrades in file saving.
2. **Use an existing library**, which functions as a translator. It had clear advantages, such as not requiring language files and being much more flexible. However, it significantly decreased the application's performance, leading to a state of unresponsiveness. Additionally, it required an internet connection. Further research led to the discovery of the "argostranslate" library, which resolved the internet connection issue, but its performance was even worse.
3. **Using XML language files**.This solution sacrificed some adaptability compared to the second option but greatly improved performance, almost on par with the first option. Use XML language files, allows access through tags instead of positions in the code. This allows for future changes in language files without affecting the existing code.

To provide a more visual analysis, we created a table:

Option	Scalability	Flexibility	Performance	Internet Conection
Use .txt language files	Poor	No	Good	Not required
Use a translator library	Excellent	Yes	Poor	Required
Use .xml language files	Sí	Good	Excellent	Not required

5.1.10 Initial configuration file

A file (.xml) has been created that stores the configuration (theme, size of the main window and language currently used) set in the application at the moment we close the main window and that is used to, the next time we start the application, have the same as when it was closed.

5.1.11 Create executable file

To create the executable file and installer, a search for alternatives to CX-freeze, the library used to generate the executable in the previous version of the application, was undertaken. So after searching it found the PyInstaller library where an executable was created in a very simple way with the command:
pyinstaller –onefile –windowed –name=ValeoApp main.py
where:

- **–onefile:** generates only an executable that includes everything needed to run it, and not a folder of files.
- **–windowed:** windowed not console application.
- **–name:**name of the app.
- **main.py:**code to get the executable from

However, there were many problems about how to import the libraries and dependencies of the application, being that the idea that this library was simpler than CX-freeze discarded and therefore discarded.

So after this, I investigated about the CX-freeze library, library that is better in terms of speed and its use is more extended than PyInstaller.

To create the installer of an application with CX-freeze two things are needed, once the library has been installed: the file containing all the configuration to create the installer (setup3.py) and execute the Python

command Python **setup3.py bdist_msi**. Running this command will create both the installer and the application executable.

As for the setup3.py file, an image with its contents is shown below:

```
import sys
from cx_Freeze import setup, Executable
import os
import tkinter

# Dependencias del proyecto

#CREAR ACCESO DIRECTO AL ESCRITORIO
# Shortcut,Directory_,Name,Component, Target,Arguments,Description,Hotkey,Icon,IconIndex,ShowCmd,WkDir
data_options={"Shortcut": [("DesktopShortcut","DesktopFolder","EMC Automotive Lighting Validation Platform", "TARGETDIR","[TARGETDIR]main.exe",
    None,None,None,None,None,None,'TARGETDIR')]}

bdist_msi_options={'data':data_options,'upgrade_code': '{66620F3A-DC3A-11E2-B341-002219E9B01E}','add_to_path': True,
                   'initial_target_dir': r'[ProgramFilesFolder]\%s\%s' % ("GranaSAT", "ALVP")}
# Ejecutable principal
base = None
if sys.platform == "win32":
    base = "Win32GUI"

executables = [Executable("main.py", base=base,icon="Images/granasat/GranaSDR.ico",copyright="GranaSAT 2020",trademarks="GranaSAT 2020")]


# Archivos de datos
include_files = ["Reports/","Images/","Languages/","gui/","datas/","fixPermission.bat","configuration.xml",
                 ("lib/libHDSN",os.path.join('lib', 'libHDSN')),
                 ("lib/opencv_videoio_ffmpeg454_64.dll",os.path.join('lib', 'opencv_videoio_ffmpeg454_64.dll')),
                 ('lib/tcl86t.dll',os.path.join('lib', 'tcl86t.dll')),
                 ('lib/tk86t.dll',os.path.join('lib', 'tk86t.dll')),
                 ('lib/matplotlib.libs',os.path.join('lib', 'matplotlib.libs')),
                 ('lib/zlib.dll','zlib.dll'),
                 ('pytransform/_pytransform.dll',os.path.join('lib', '_pytransform.dll')),
                 ('lib/Qt5Core.dll',os.path.join('lib', 'Qt5Core.dll')),
                 ('lib/Qt5Gui.dll',os.path.join('lib', 'Qt5Gui.dll')),
                 ('lib/Qt5Widgets.dll',os.path.join('lib', 'Qt5Widgets.dll'))]

build_exe_options = {"packages": ['numpy','os','cv2','winsound','pytransform','PyQt5','src','serial','xml.dom','requests','base64','jinja2','PIL','matplotlib'],
                     'include_files':include_files}
# Configuración general del proyecto
setup(
    name="EMC Automotive Lighting Validation Platform",
    version="2.0",
    description="EMC Automotive Lighting Validation Platform",
    author="Javier Exposito Martinez",
    options={"build_exe": build_exe_options,'bdist_msi':bdist_msi_options},
    executables=executables
)
```

Figure 5.7 – *Setup3.py's content*

where:

- **import sys:** Imports the sys module, which provides access to Python interpreter-specific variables and functions.
- **from cx_Freeze import setup, Executable:** Imports the setup and Executable functions of the cx_Freeze module. These functions are used to configure and create executables of the installation package.
- **import os:**Imports the os module, which provides functions to interact with the operating system.
- **import tkinter:**Imports the tkinter module, which is used to create graphical user interfaces.
- **data_options="Shortcut": [...]:** Defines the options for creating desktop shortcuts.
- **bdist_msi_options=...:**Defines options for the creation of the installation package in MSI (Microsoft Installer) format. These options include the configuration of directories, update codes, initial destination directory and more.
- **base = None:**Initializes the base variable as None.
- **if sys.platform == "win32": base = "Win32GUI":**Checks if the operating system on which the script is running is Windows. If so, set the value of 'base' to "Win32GUI". This indicates that the GUI should be used when running the program.

- **executables = [...]:**Defines the program executables to be included in the installation package. In this case, a file named "main.py" is specified as the main executable, along with some additional options such as the icon, copyright and trademarks.
- **include_files = [...]:**Defines the files and directories to be included in the installation package.Include in this section the .dll files are important because without them the executable cannot make use of the libraries it needs.

  These .dll files were obtained by searching for them in the local files of the computer, namely:

  - tcl86t.dll and tk86t.dll were found in 'C:\Users\javie\anaconda3\envs\TFG\Library\bin'.
  - zlib.dll was located at 'C:\Users\javie\anaconda3\envsTFG'
  - PyQt5 .dll can be found in 'C:\Users\javie\anaconda3\envs\TFG\Lib\site-packages\PyQt5\Qt5\bin
  - _pytransform.dll is in the folder generated by obfuscating the code (to be explained later, specifically in the section on security enhancement)

  These files and directories will be copied to the installation directory during installation.
- **build_exe_options = ...:** Defines the build options for the run package. These options include the required Python packages (libraries), the files and directories to be included, among others.
- **setup(...):** Calls the setup function of cx_Freeze to configure the installation package. The project name, version, description and author are specified, along with build options and installation options in MSI format.
- **executables=executables:**Specifies the executables to include in the installation package.

5.1.12 Statusbar

A status bar has been created for the mainWindow class as well as for SelectROI, DUT_ROI_window and EMCLightAnalysis. It displays messages about the status of the application, the selected action and more information. In addition, depending on the type of message it will change color, that is, if it is an error message, the status bar changes to red, if it is an action it changes to white, etc.

In addition the status bar of the main window, not only shows the above, but also the current date, which is updated every second thanks to a timer and also shows a progress bar about the process being carried out (Test DUT or Test EMC).

5.1.13 New functionality: Check updates

In this version, the new functionality "Check Updates" has been added, which shows whether the version of the application installed by the client is the current version or an older one. Whether there is no connection available, the application is already installed, or an update is required, the corresponding window and text informing the client are displayed. If an update is necessary, pressing the download button will download the installer with the latest version of the browser, and the download process can be observed with a progress bar in the status bar of the mainWindow class.

To develop this functionality, we need to make changes in two parts: the client and the server.

Regarding the server part, the "version.php" file had to be edited and re-uploaded to the server. Editing it is straightforward; you just need to change the last line that prints the version to the current version of the application. After that, it was re-uploaded to the server.

However, to perform all the necessary tests, a local server was created to function as the real server. How

to install a local server is explained in Appendix: How to install a Local Server .

Regarding the client part, the code changes are shown in more detail in the corresponding section. However, it's worth mentioning the use of a Thread to perform the installer download process. This is because when the download is done in the main Thread, only the progress bar gets updated, but the rest of the application remains blocked until the process finishes. By using a Thread, when the download button is pressed, the Thread takes care of downloading the installer and updating the download progress bar, allowing the client to continue performing other actions in the application. Therefore, this is the graphical representation of the "Check Updates" functionality:

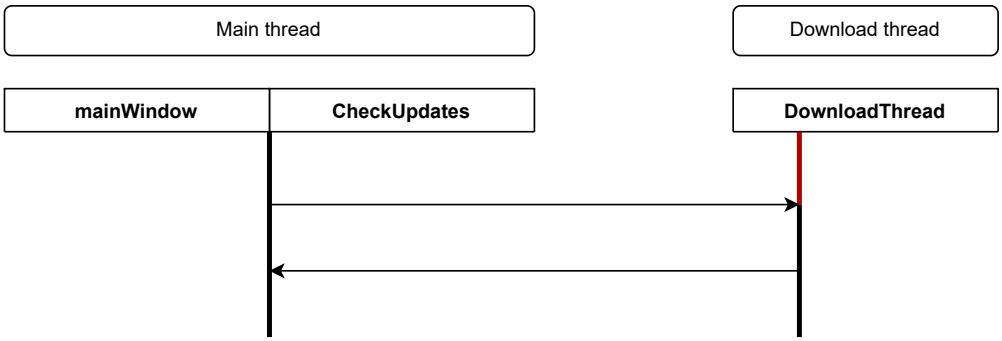

Figure 5.8 – *Check updates functionality*

5.1.14 Improvements of the classes SELECT ROI, DUT_ROI and EMCLightAnalysis

5.1.14.1 Improved graphical interface

The interfaces of the Select_ROI, DUT_ROI and EMCLightAnalysis classes have been improved so that now the camera and zoom view, or the Keyframe and video view can be decoupled and coupled allowing to work on one, two or even three screens, allowing also the resizing of each of the windows without any problem (ROIs are drawn by joining the points whose coordinates are calculated taking into account the image resolution, and recalculated in case the resolution changes). Unlike the previous version, the camera and zoom windows have a very small and fixed size, generating low quality images, problems with the visualization of the views, etc.

To undock the windows, it is as simple as right clicking with the mouse and dragging, while to dock the windows just close them.

In addition the mouse pointer changes showing graphically when you can paint an ROI, when you can undock a window or when you can do nothing. Now I proceed to show the difference between the previous interface and the current one:

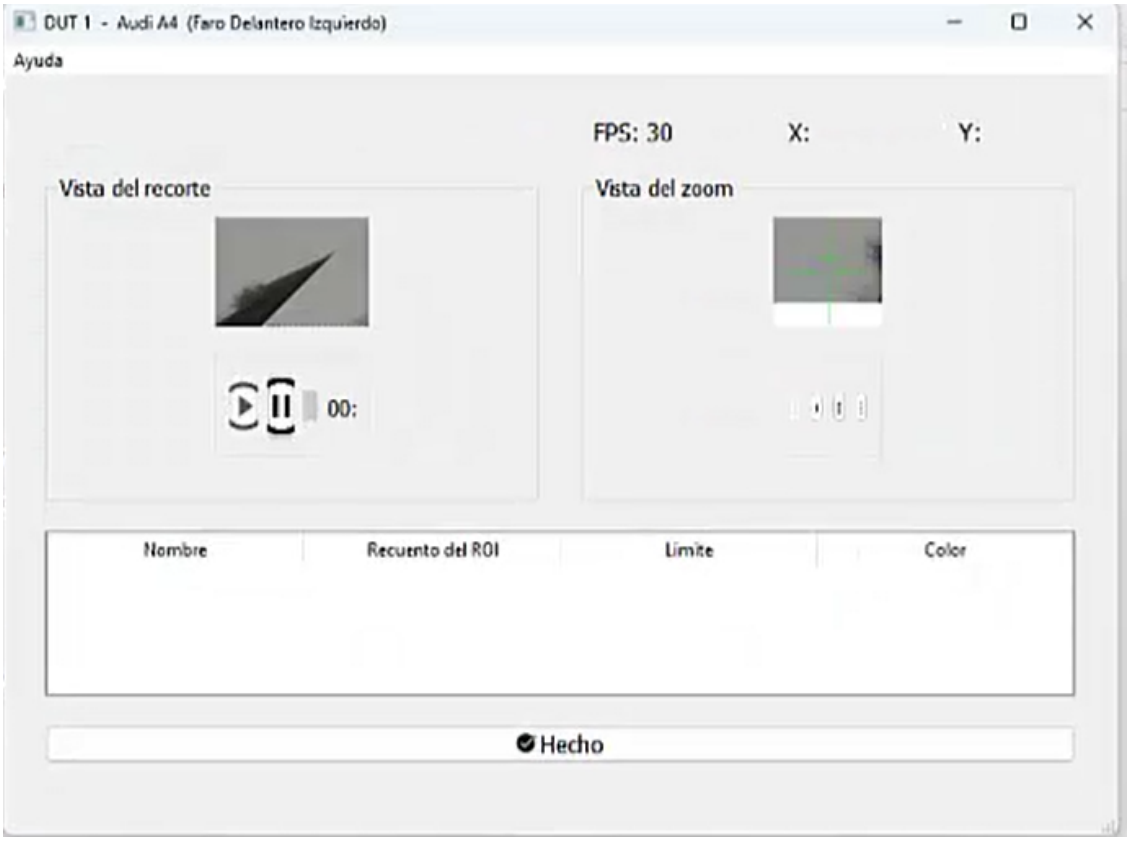

Figure 5.9 – *Before*

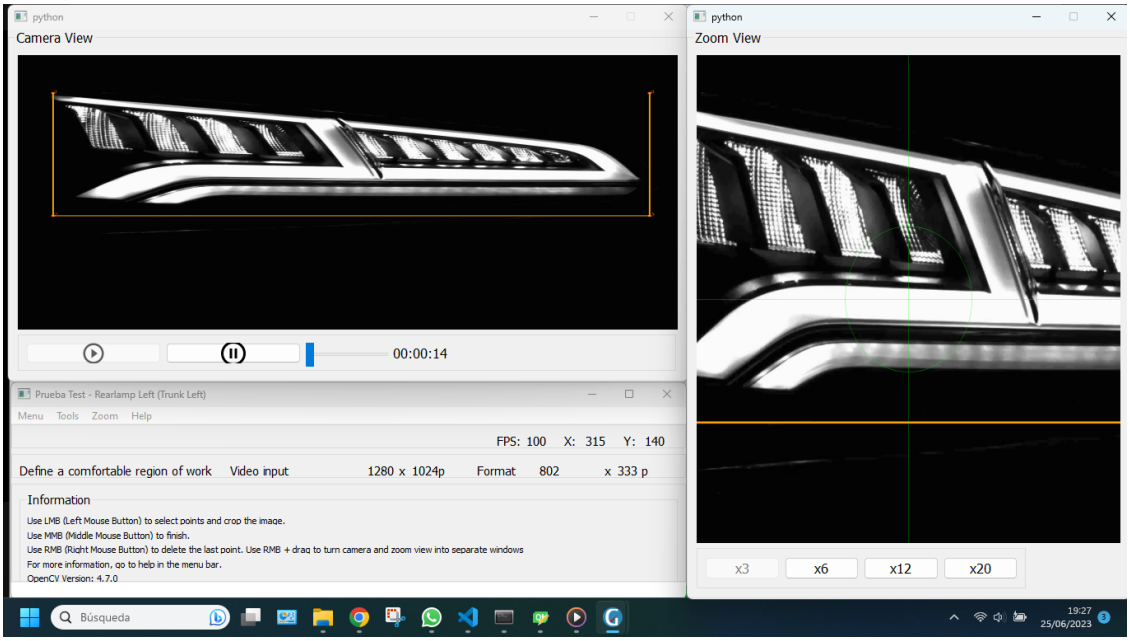

Figure 5.10 – *After*

Also now, if you open a file, the crop points and ROIs that were drawn when the test we have opened is performed are displayed in the interface.

In EMC Analysis, when you select an ROI, it is now displayed in the Keyframe view. Additionally, when you select an alert, a pop-up window appears showing which alert it is as shown below:

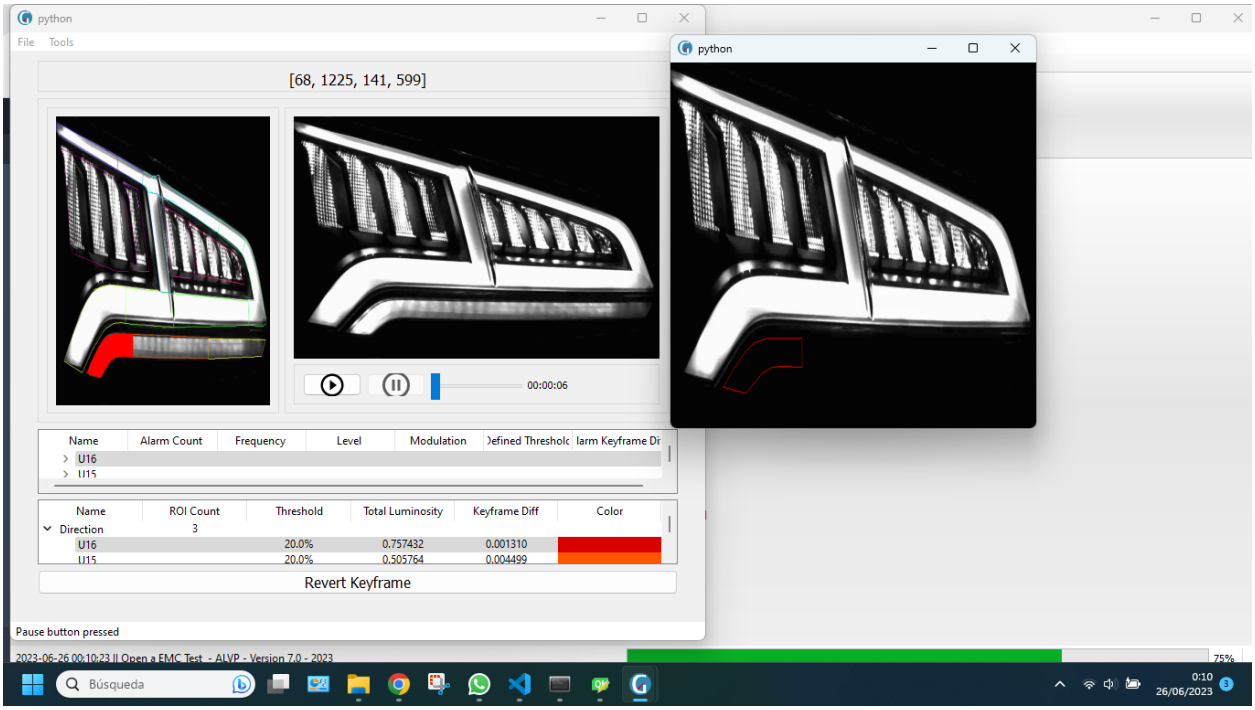

Figure 5.11 – *show ROIs and alerts in EMC analysis*

5.1.14.2 Higher video readout speed (frame rate)

Another important aspect was optimizing performance and improving efficiency, as the customer's recorded videos have a high Framerate (one hundred frames/s). This fact was extremely important because if the customer wants to analyze a complete video, this analysis should take as little time as possible. For example, before implementing this improvement, a video with a duration of approximately six seconds took about sixty seconds to process the entire video, ten times slower. After this improvement, the same video takes between ten and thirteen seconds to be fully analyzed, representing an 83.33% improvement.

To achieve this goal, the first step has been to optimize the entire code by deleting unnecessary parts and simplifying it, avoiding the use of nested loops as much as possible.

However, the main change that has had the greatest impact on performance is the utilization of Threads, allowing certain parts of the code to be executed simultaneously. In summary, we have the main Thread, which carries out the overall functioning of the application, and two additional Threads, one responsible for the logic behind the camera view and the other for the zoom view. This way, all the processing is done by the Threads in a synchronized manner, using a queue, similar to the producer-consumer problem. Now, we proceed to graphically illustrate the process of reading and processing the video:

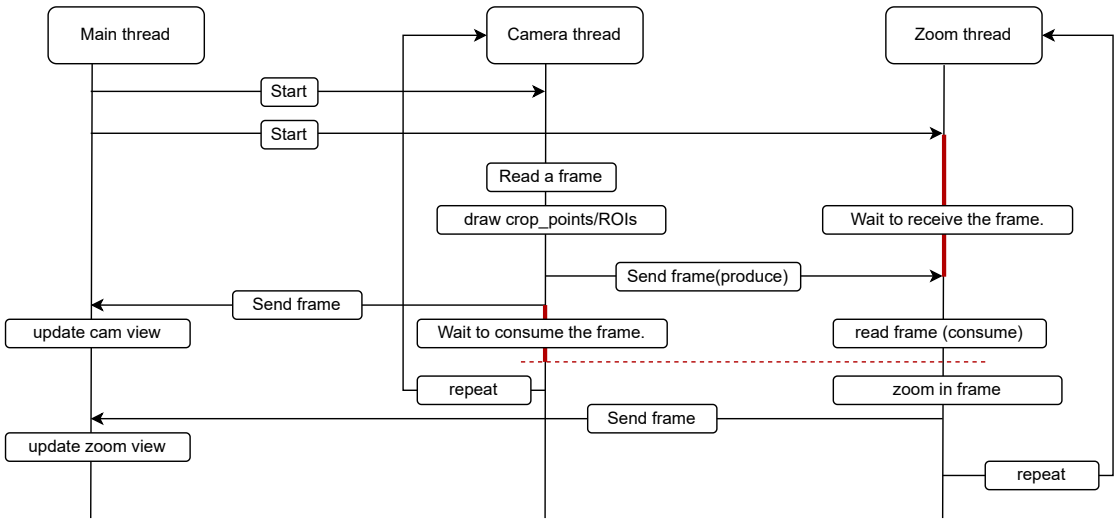

Figure 5.12 – *Creation of views using Threads*

5.1.14.3 Generated reports' improvement

In EMC analysis,a new class, generateReport, has been created for the creation of reports after EMC analysis and a new template used by jinja2 to generate a report has been created. Additionally the report now has a section where the graphs containing the brightness values corresponding to each ROI during the whole test are displayed. You can visualize at what moment the brightness value triggered the alert or, in case it did not trigger the alert, how close it came to triggering it. An example of a current report is shown below in the following pictures:

Valeo EMC CV ADVP - Version 7.0 - 2023 LICENSED

DUT Info

Device name:	Rearlamp Left
Device position:	Trunk Left
Creator:	Javivi

Defined ROIs:

Direction

ID	ROI Name	ROI Threshold	Color
0	U10	20.00%	
1	U15	20.00%	
2	U14	20.00%	

Stop

ID	ROI Name	ROI Threshold	Color
3	U11	20.00%	
4	U0	20.00%	
5	U9	20.00%	
6	U10	1.00%	
7	U1	20.00%	
8	U4	20.00%	
9	U3A	20.00%	
10	U3B	20.00%	
11	U2	20.00%	

Rearlamp

ID	ROI Name	ROI Threshold	Color
12	U6	20.00%	
13	U7A	20.00%	
14	U7B	20.00%	
15	U5	20.00%	

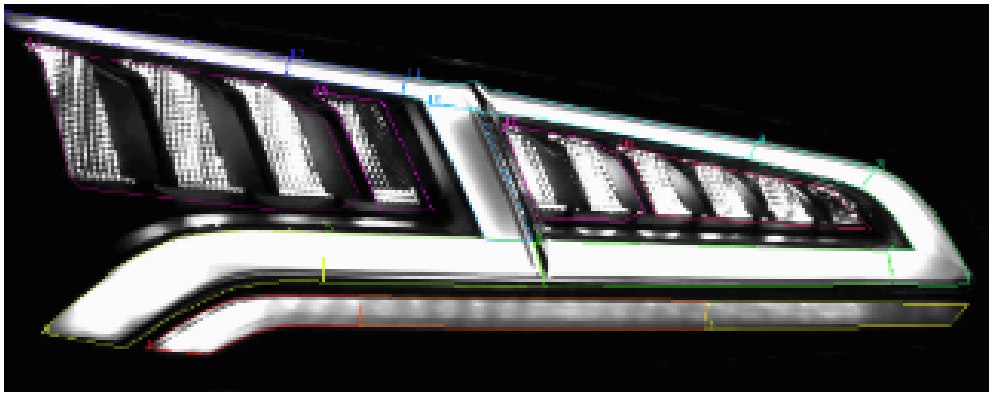

Figure 5.13 – *Report: DUT information*

BCI Test Info

Project:	
Application Standard:	
Test factory:	
System Engineer:	
Hardware Engineer:	
eOL:	
Validation Engineer:	
Part status:	
Purpose:	
Status Name:	
BCODE Description:	
Version:	
AML name:	
AML release date:	
Gerber files release date:	
IW Version:	
Software Version:	
Recipe options:	
Remarks:	
Quantity:	
Keyframe:	

Figure 5.14 – *Report: BCI information*

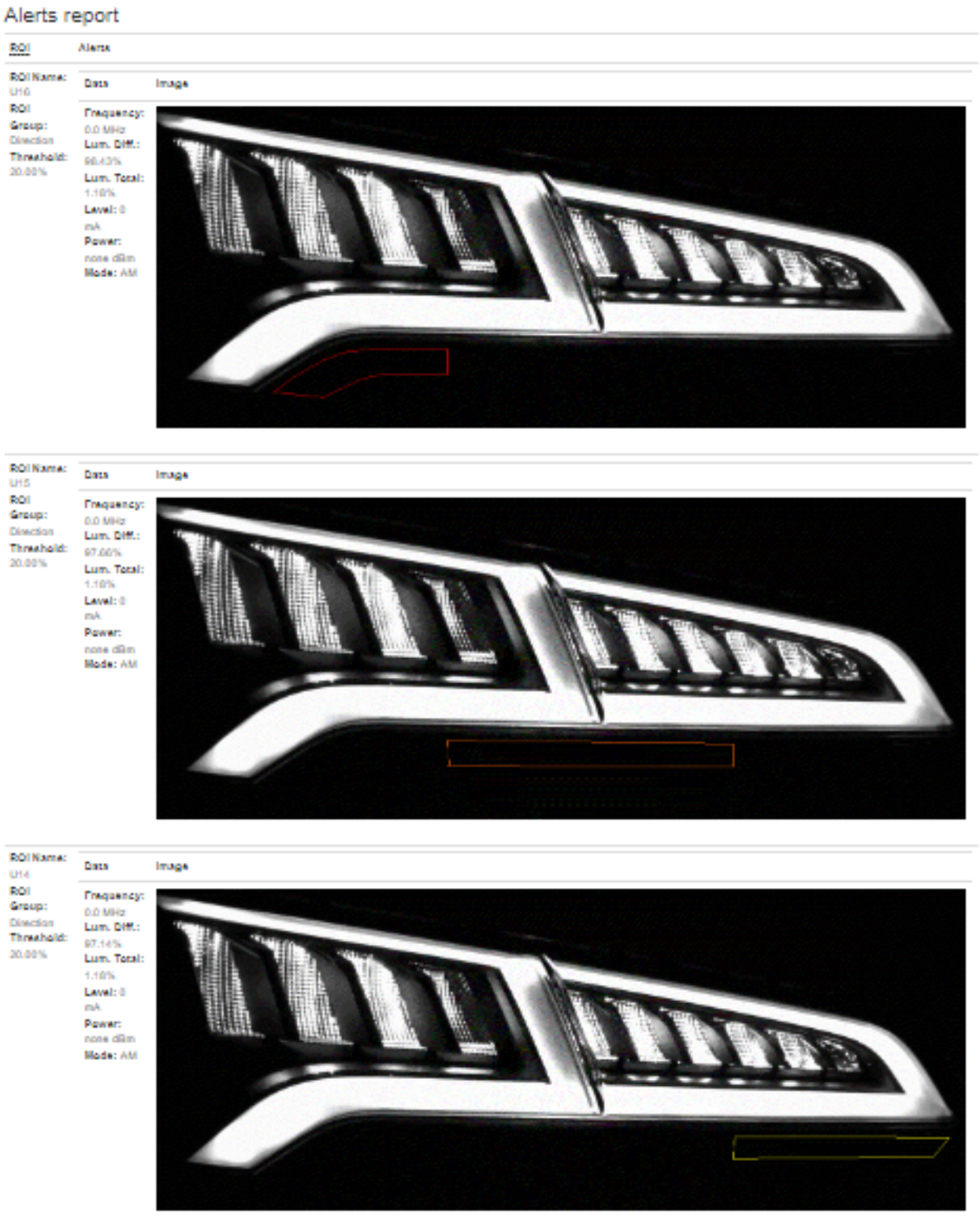

Figure 5.15 – *Report: Alerts*

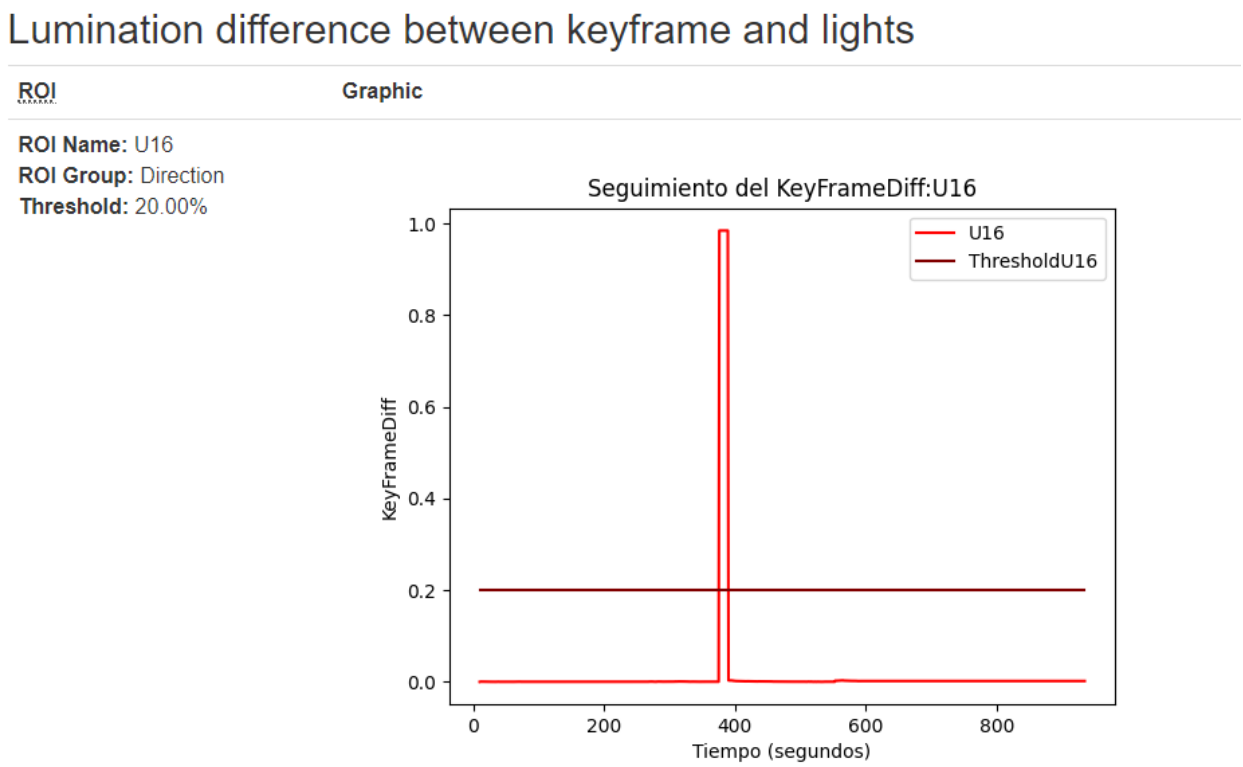

Figure 5.16 – *Report: Diagram of difference of ilumination*

5.1.15 New function: open recent file

Now, when the user is browsing the menu and presses open file, not only the option to open a new file but also the last file that was opened by the application will appear.

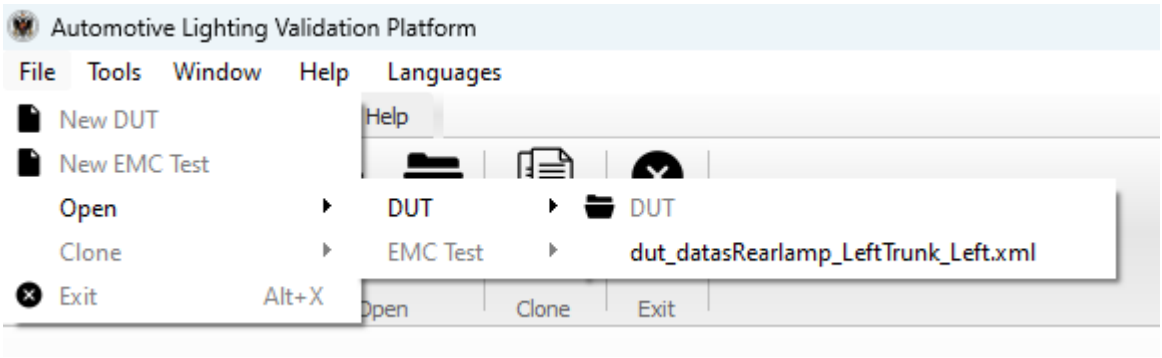

Figure 5.17 – *Open Recent File*

5.1.16 Organize the code application's code

The code has been organized in folders according to the functionality to which they belong in the application. So the directory containing the code (the src folder) is now much easier to understand.

The changes that were made in the code, were modifications in the imports, using now relative path to include a class.

5.1.17 New toolbar

A toolbar has been created in the main application window to perform the same actions as the menu in a simple and visual way. This way, the application has a more modern and updated visual aspect, and not obsolete and belonging to old applications.

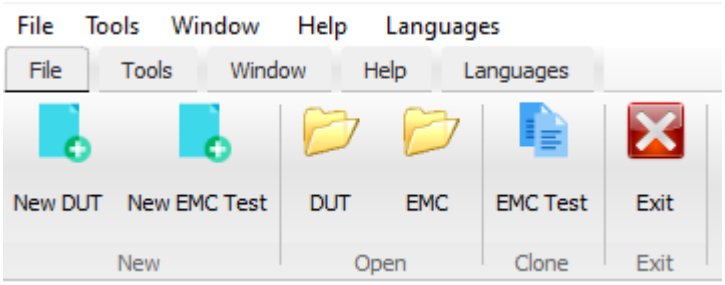

Figure 5.18 – *Toolbar*

5.1.18 Security

This section addresses the issue of security with respect to obtaining the source code from the executable. After a thorough investigation, it was concluded that it is not possible to obtain the source code from the executable but from files that are generated along with it, using tools such as decompyle3 or decompyle6. This fact shows the urgency and necessity of using security methods to avoid this.

The security method to avoid this is obfuscation, which converts the code of a software or project into a type of code that is more difficult for humans to understand. It achieves this goal by applying encryption mechanics and patterns to prevent access to critical sections of the code. And to achieve this, the PyArmor library is used.

Although obfuscation can be reversed with Reverse engineering, it is a very slow and complex process that only an expert could perform. The PyArmor documentation itself states the following: "PyArmor focus on protecting Python scripts, by several irreversible obfuscation methods, now PyArmor make sure the obfuscated scripts can't be restored by any way." [3]

On the other hand, a license has been created for the application, depending on the expiration date and the serial number of the hard disk, the client will be able to use the application or an error window will appear saying that the license has expired or is incompatible on the device that is running the application.

For the process of creating the license, at first, it was thought to use the PyArmor library itself, whose main advantage was the security it provided, and the simplicity to create it. However, it was discarded due to incompatibility problems with the CX-freeze library, causing problems in the executable.Therefore, only Python libraries are used to access the hard disk serial number and check the expiration date. The license logic is programmed in the code itself.

Chapter 6

Testing and validation.

In this chapter, all the requirements imposed by the customer, both functional and non-functional, are validated. Therefore, it will be analyzed requirement by requirement, verifying that everything works correctly.

6.1 Functional Requirements

6.1.1 RF.1

Description	RF. 1 The software must allow the user to create a new DUT Test.
Analysis	Requirement satisfied by correcting and redesigning the classes involved in the process.
Evaluation	Validated

Table 6.1 – *RF.1*

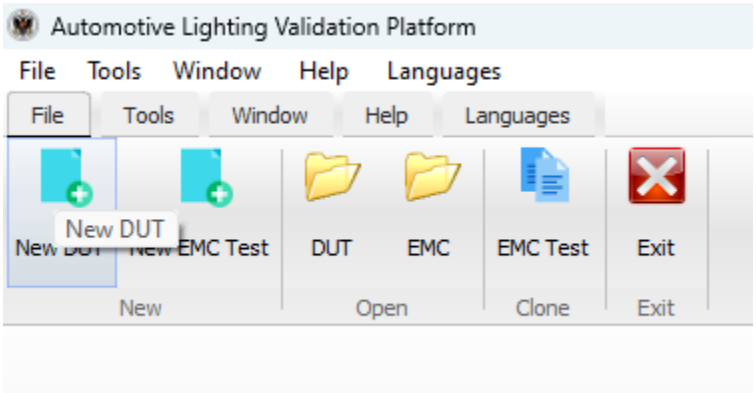

Figure 6.1 – *New DUT: step one*

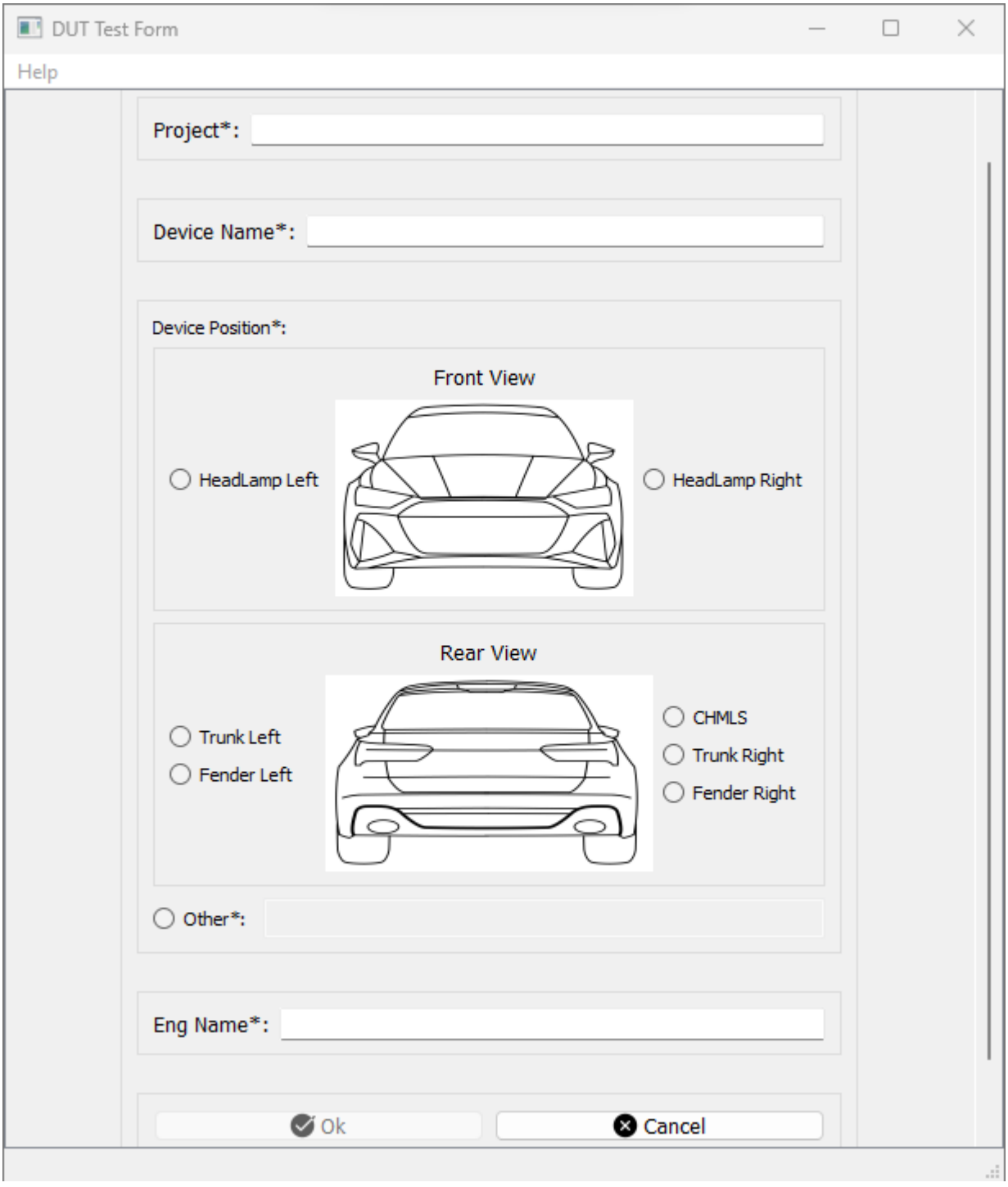

Figure 6.2 – *New DUT: step two*

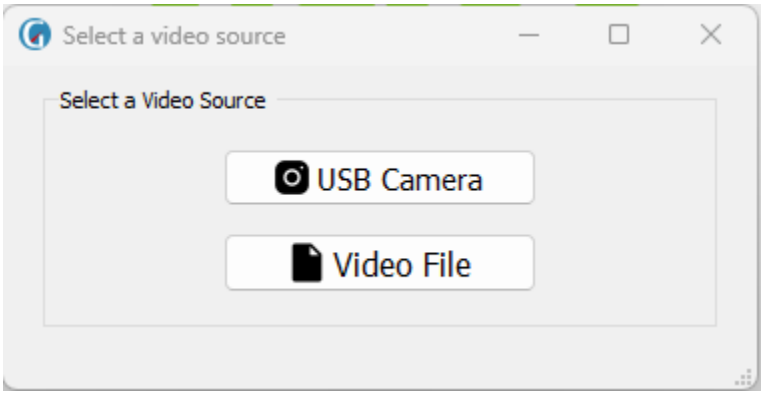

Figure 6.3 – *New DUT: step three*

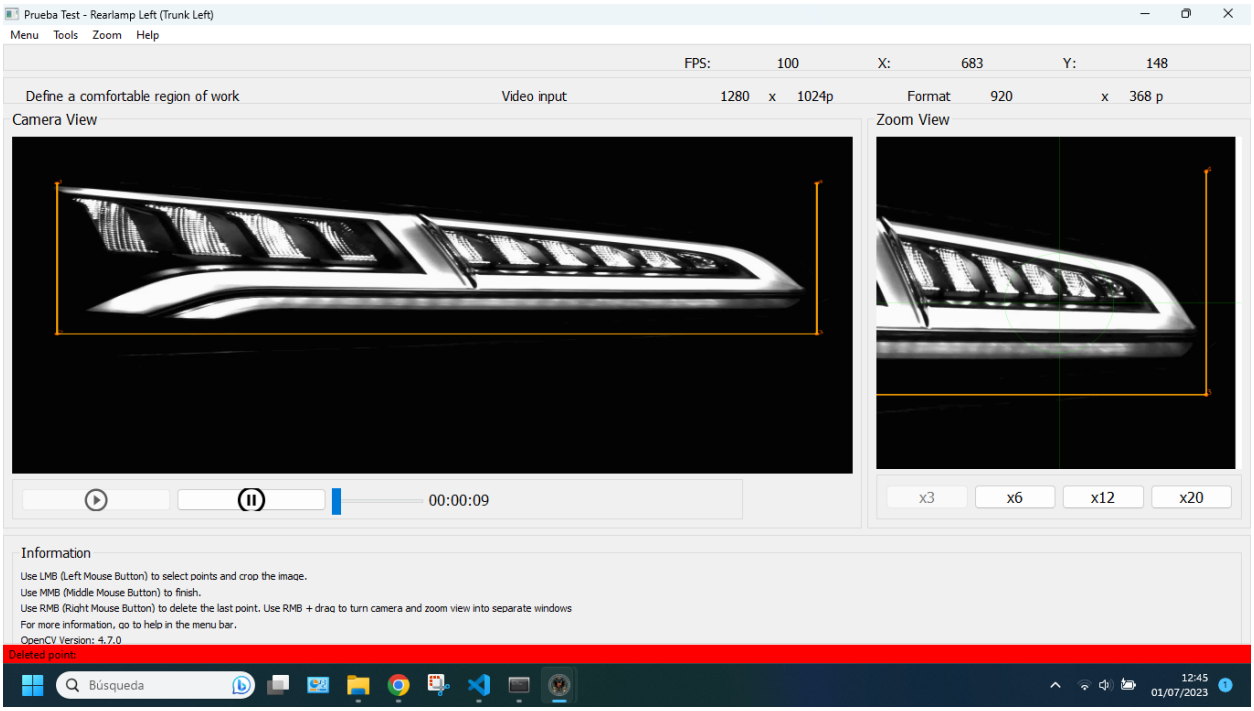

Figure 6.4 – *New DUT: step four*

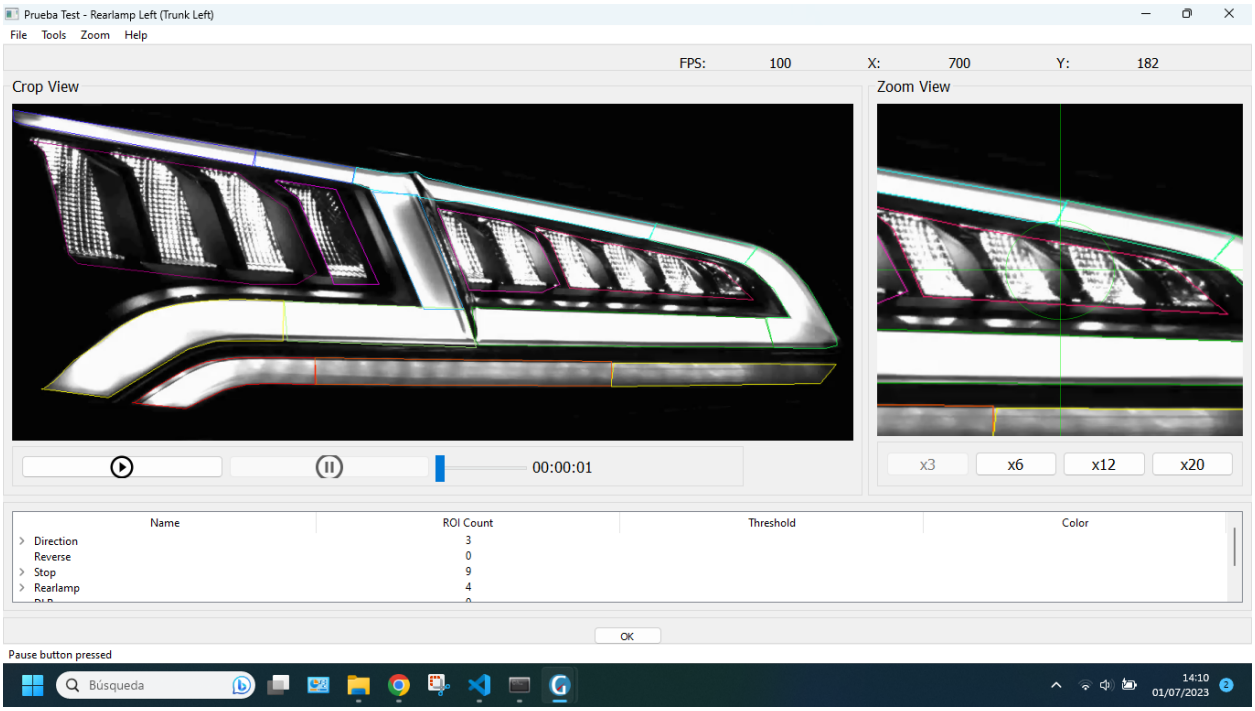

Figure 6.5 – *New DUT: step five*

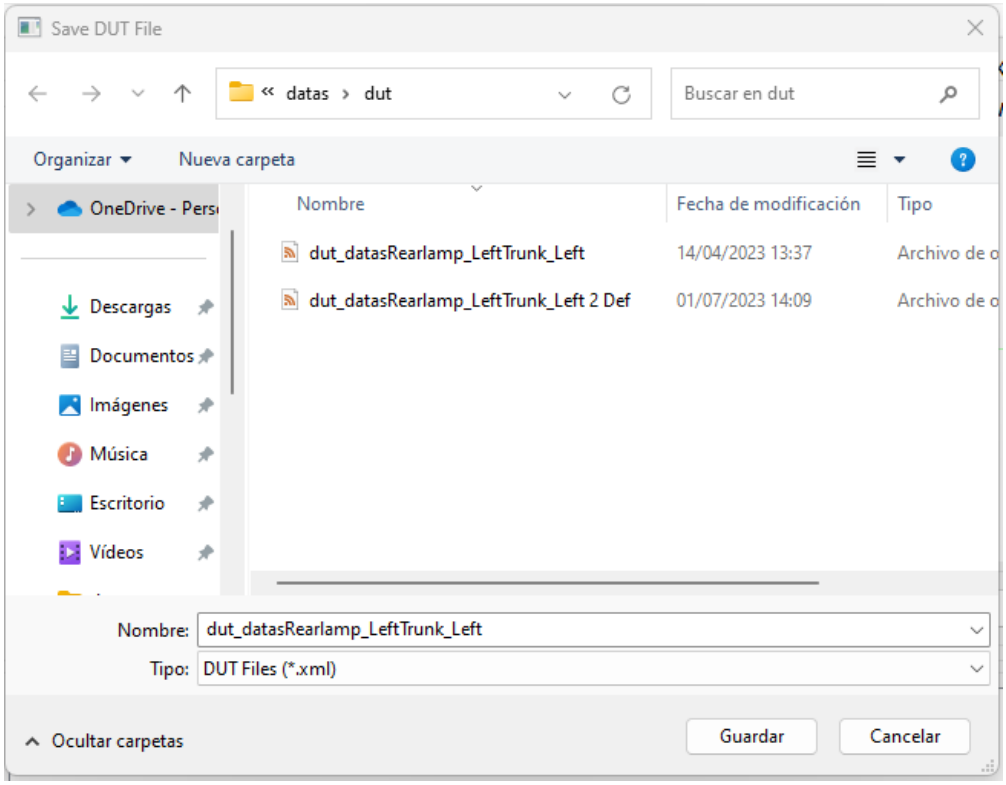

Figure 6.6 – *New DUT: step six*

6.1.2 RF.2

Description	RF. 2 The software must allow the user to create a new EMC Test.
Analysis	Requirement satisfied by correcting,redesigning and creation of the classes involved in the process.
Evaluation	Validated

Table 6.2 – *RF.2*

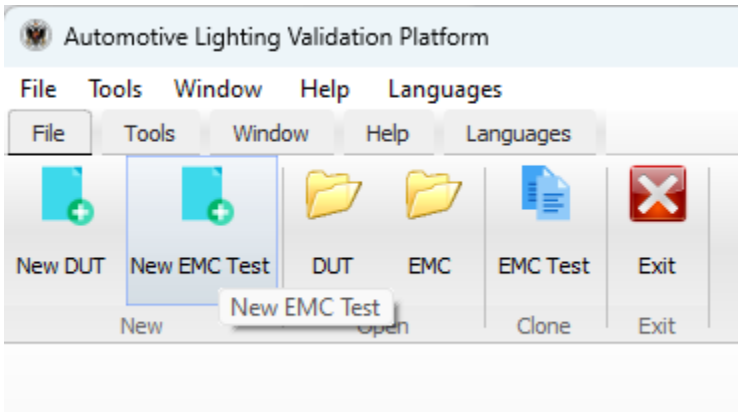

Figure 6.7 – *New EMC: step one*

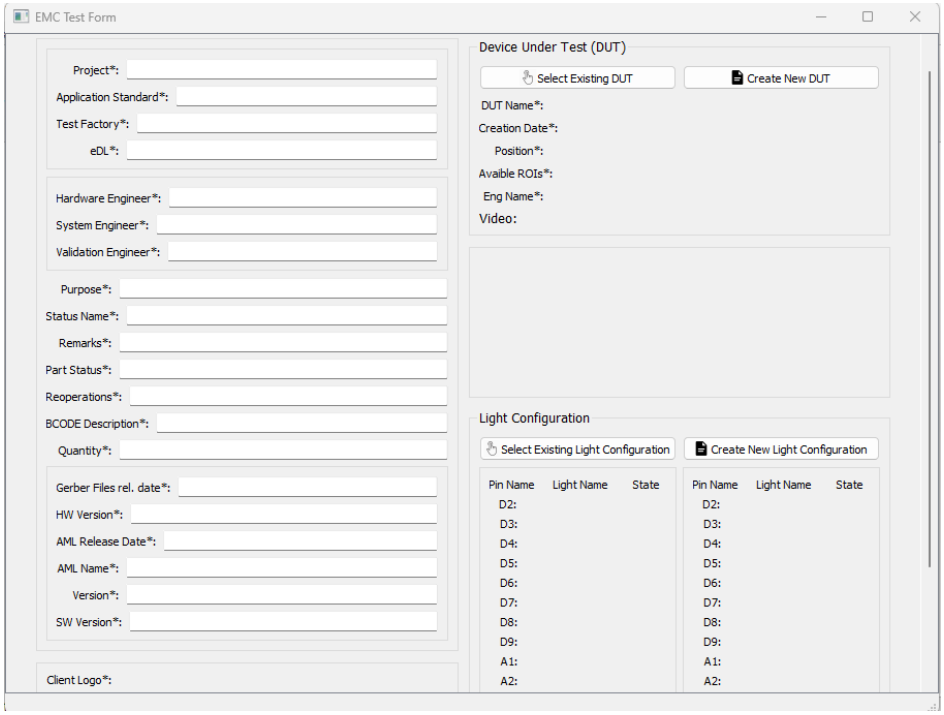

Figure 6.8 – *New EMC: step two*

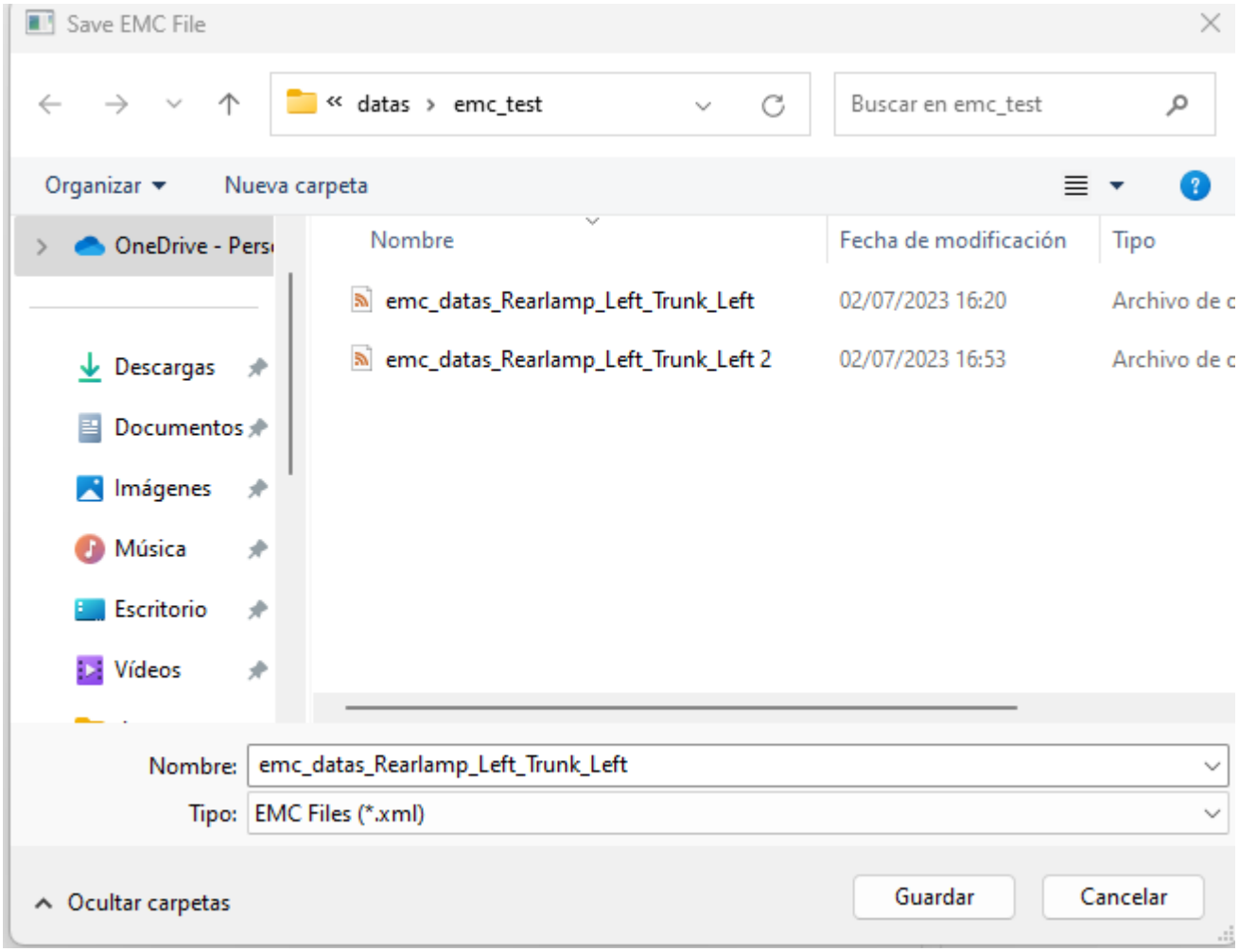

Figure 6.9 – *New EMC: step three*

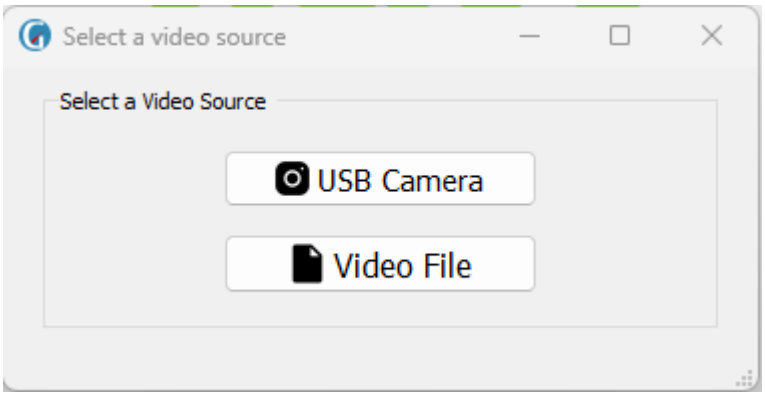

Figure 6.10 – *New EMC: step four*

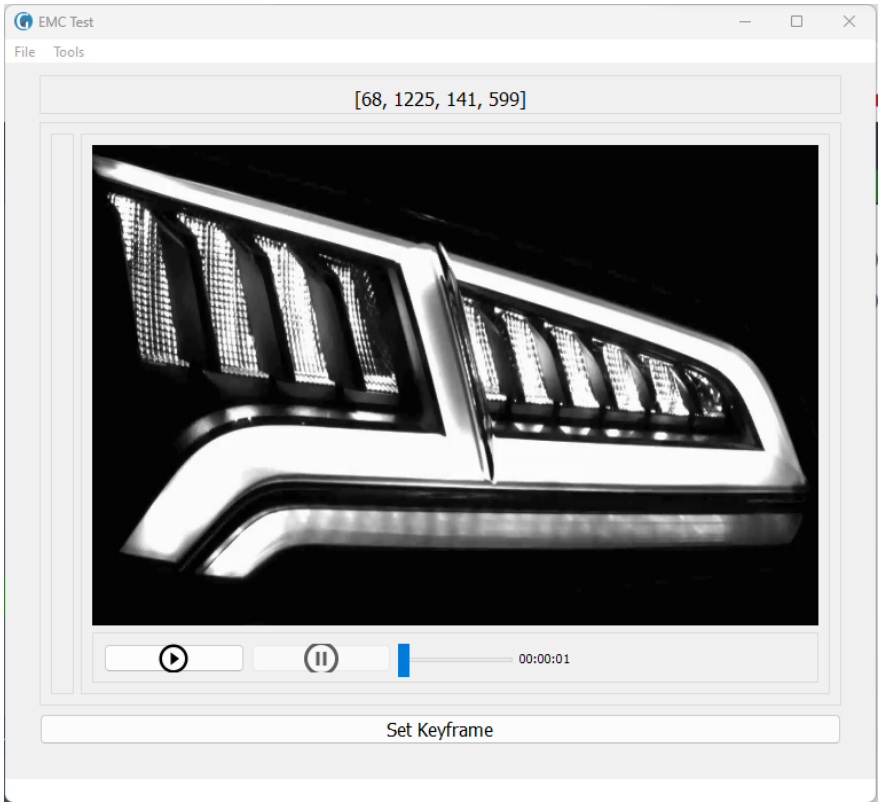

Figure 6.11 – *New EMC: step five*

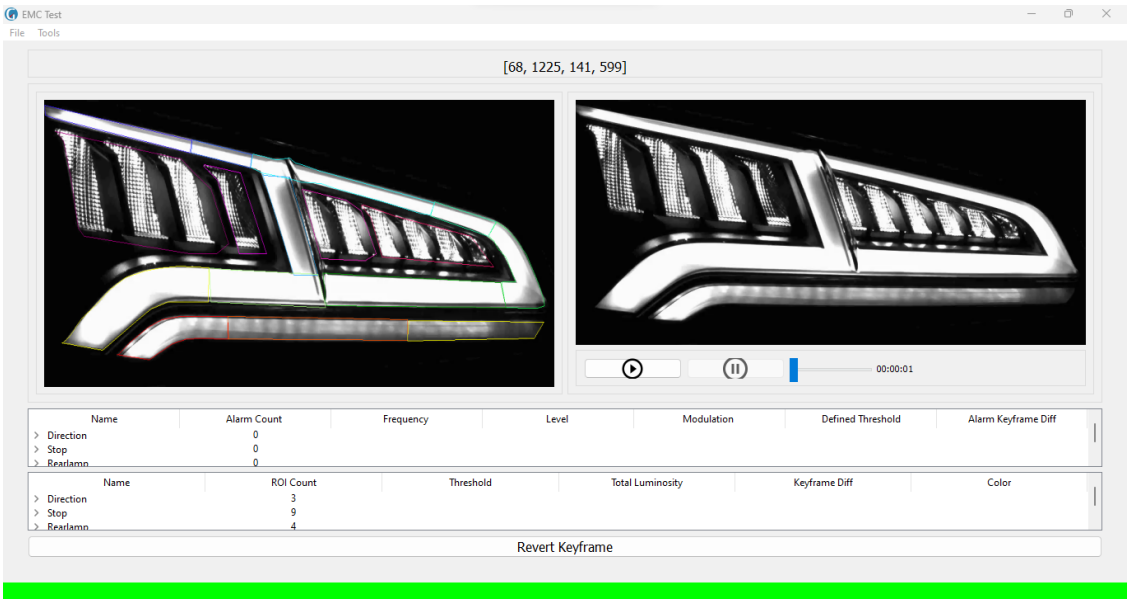

Figure 6.12 – *New EMC: step six*

6.1.3 RF.3

Description	RF. 3 The software must allow the user to open a DUT file.
Analysis	Requirement satisfied by creating a new action in the menu that allows to open a file, and then perform the whole process similar to RF.1
Evaluation	Validated

Table 6.3 – *RF.3*

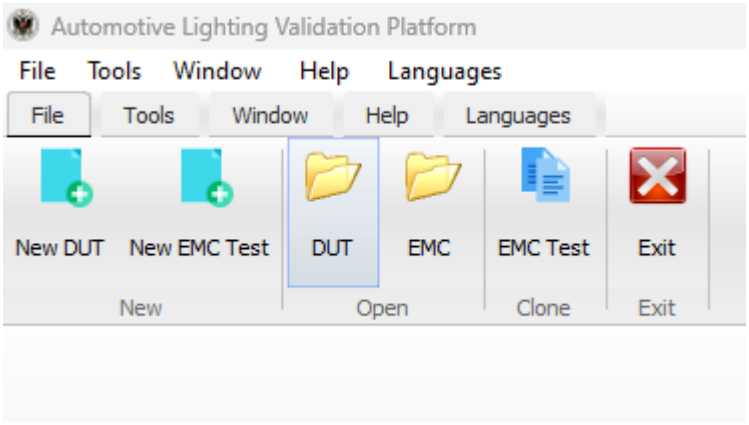

Figure 6.13 – *Open DUT: step one*

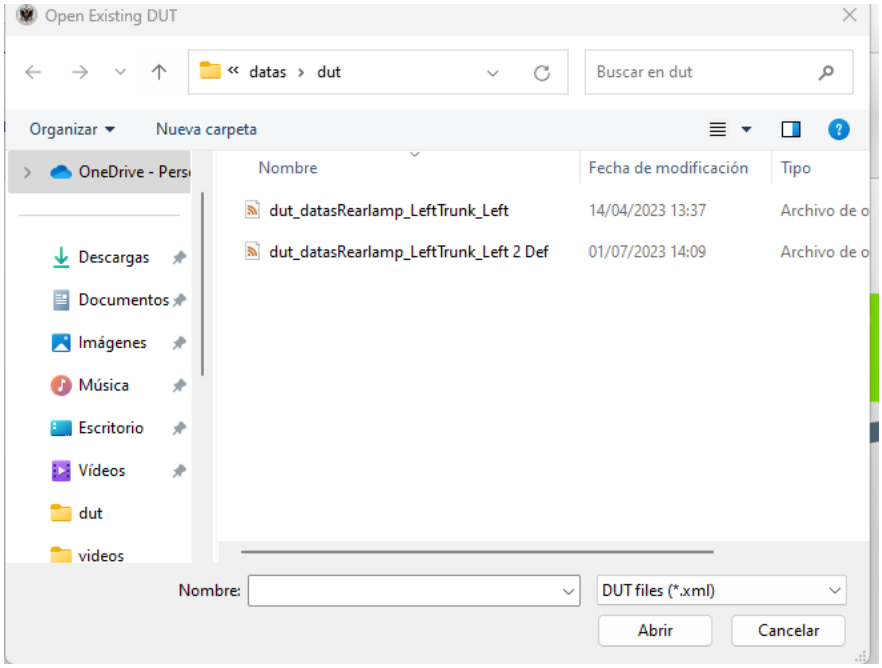

Figure 6.14 – *Open DUT: step two*

6.1.4 RF.4

Description	RF. 4 The software must allow the user to open an EMC file
Analysis	Requirement satisfied by creating a new action in the menu that allows to open a file, and then perform the whole process similar to RF.2
Evaluation	Validated

Table 6.4 – *RF.4*

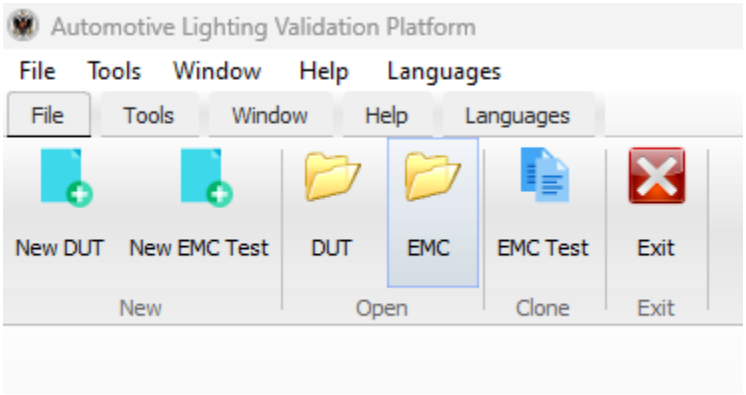

Figure 6.15 – *Open EMC: step one*

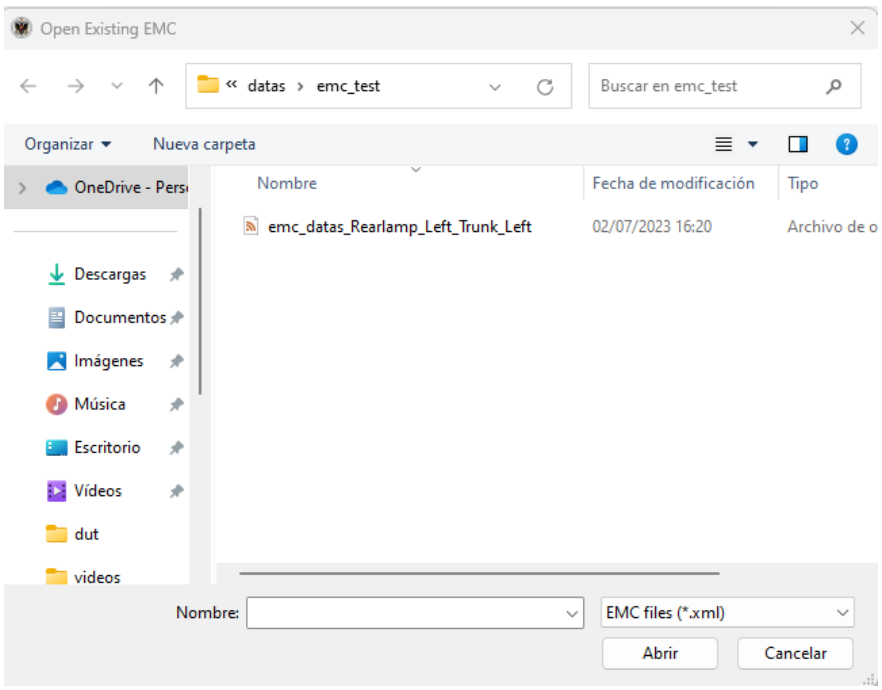

Figure 6.16 – *Open EMC: step two*

6.1.5 RF.5

Description	RF. 5 The software must allow the user to clone an EMC file
Analysis	Requirement satisfied by creating a new action in the menu that allows to clone a file, and then perform the whole process similar to RF.2
Evaluation	Validated

Table 6.5 – *RF.5*

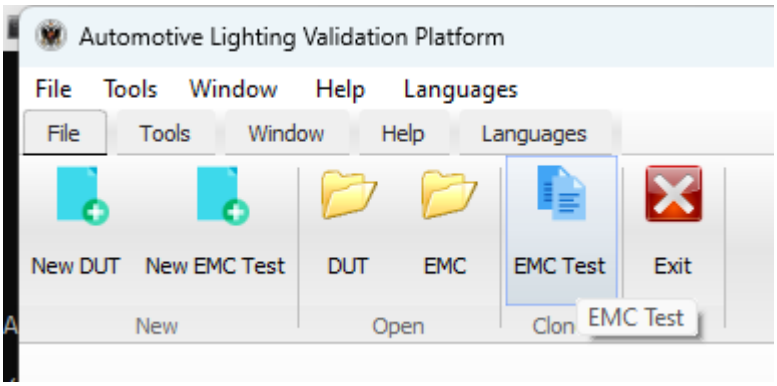

Figure 6.17 – *Clone EMC: step one*

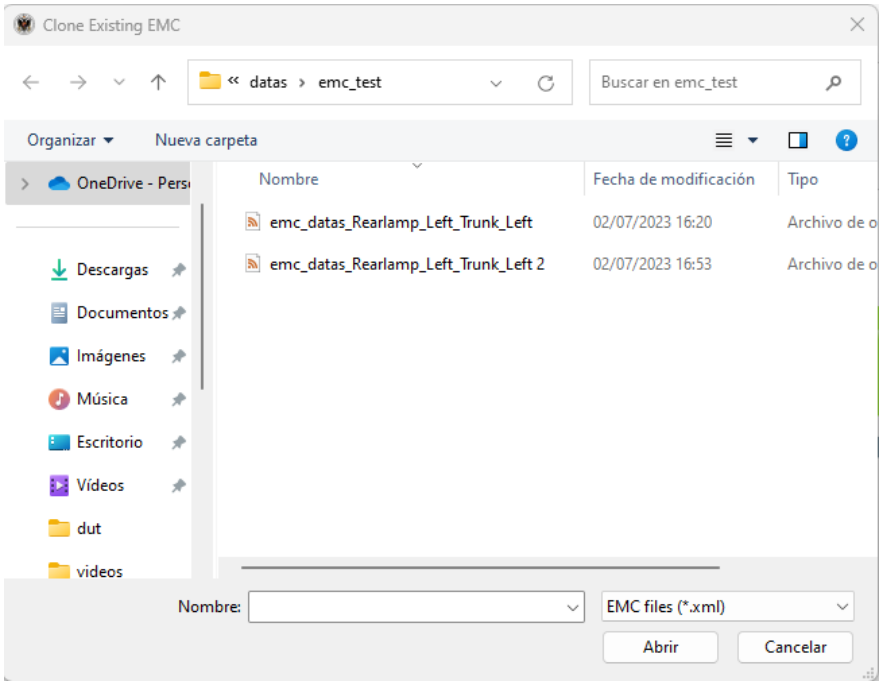

Figure 6.18 – *Clone EMC: step two*

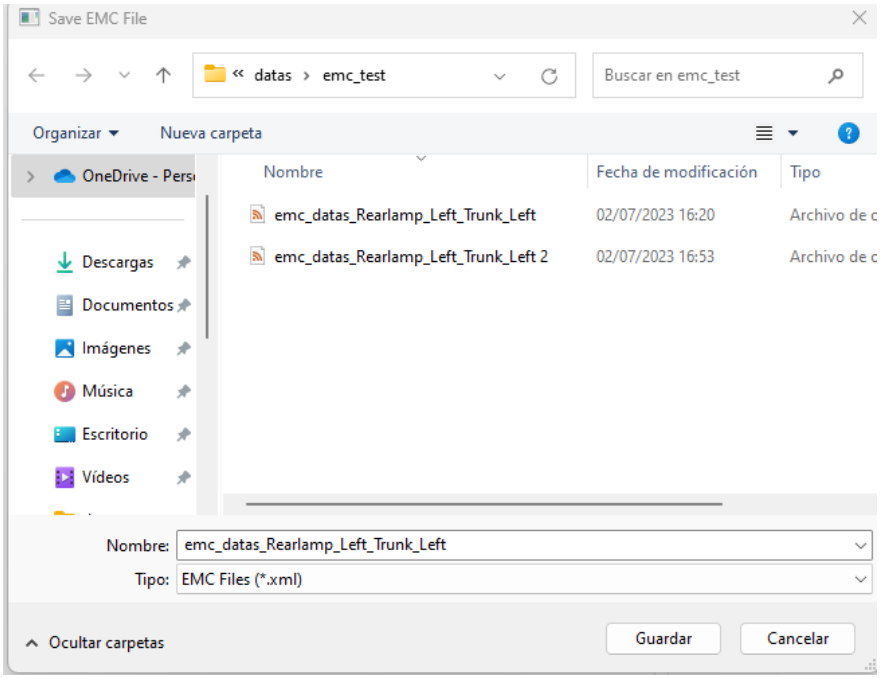

Figure 6.19 – *Clone EMC: step three*

6.1.6 RF.6

Description	RF. 6 The software must allow the user to open a recent DUT or EMC file.
Analysis	Requirement satisfied by creating a new action in the menu that allows to open a recent file, and then perform the whole process similar to RF.1 or RF.2
Evaluation	Validated

Table 6.6 – *RF.6*

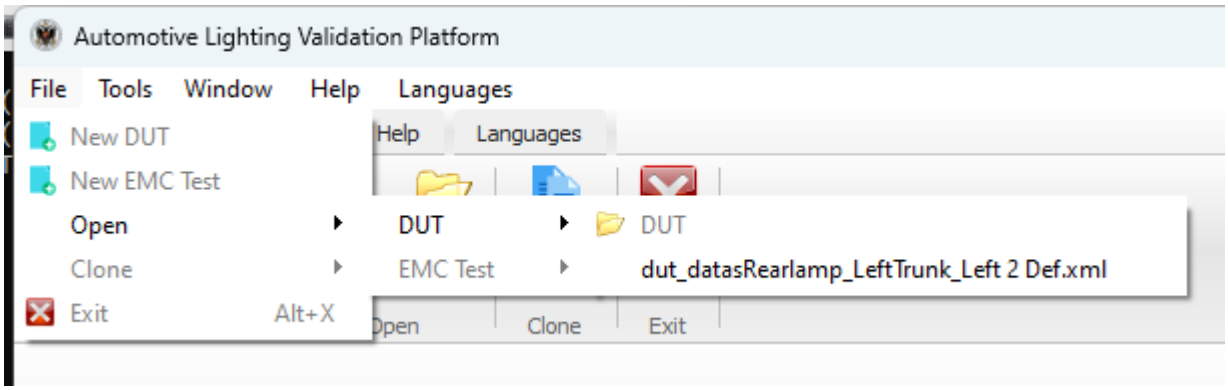

Figure 6.20 – *Open recent DUT*

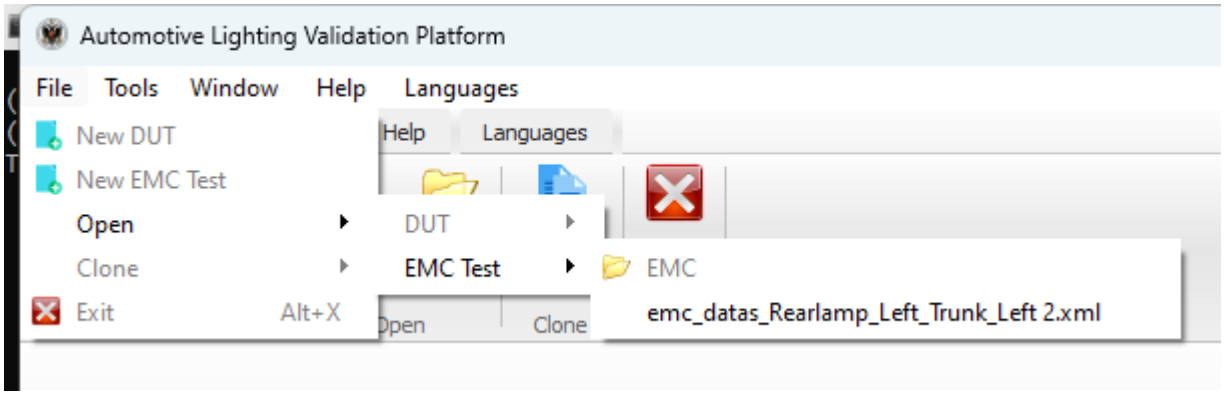

Figure 6.21 – *Open Recent EMC*

6.1.7 RF.7

Description	RF.7 The software must allow the user to configure car's spotlights using one or two Arduinos, and save this light's configuration.
Analysis	Requirement satisfied by correcting and redesigning the classes involved in the process.
Evaluation	Validated

Table 6.7 – *RF.7*

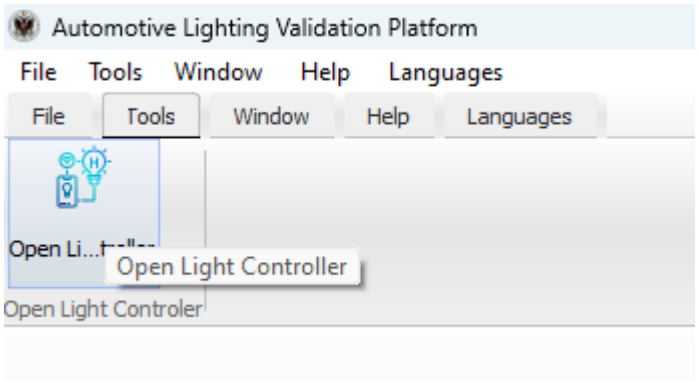

Figure 6.22 – *Open light controller: step one*

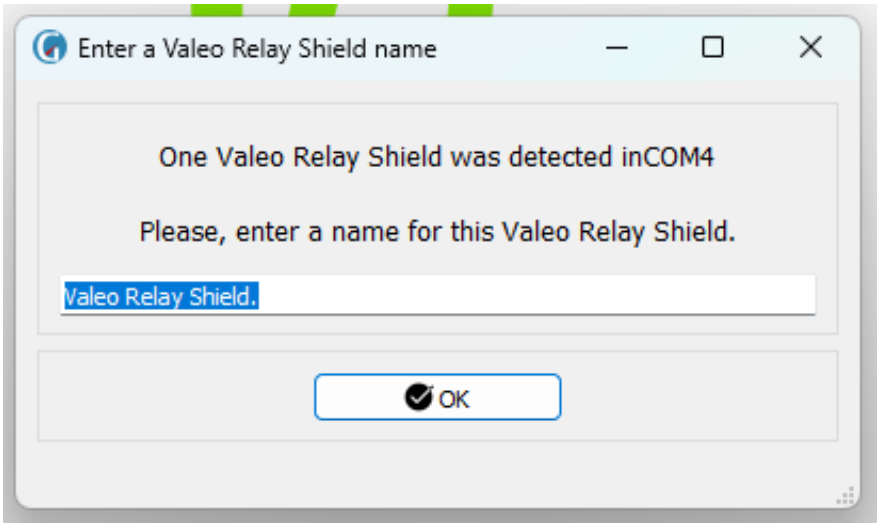

Figure 6.23 – *Open light controller: step two*

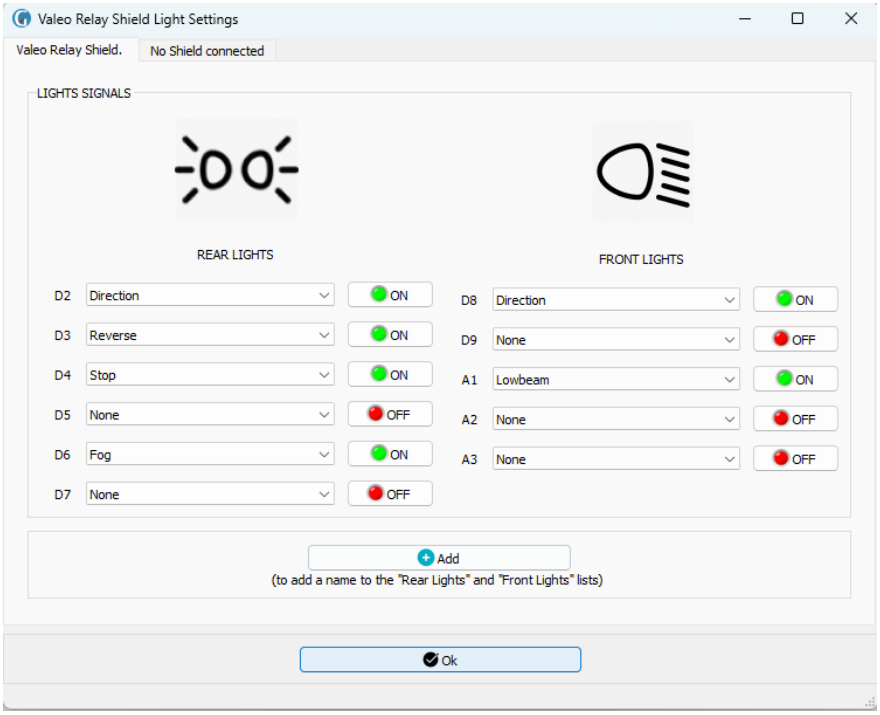

Figure 6.24 – *Open light controller: step three*

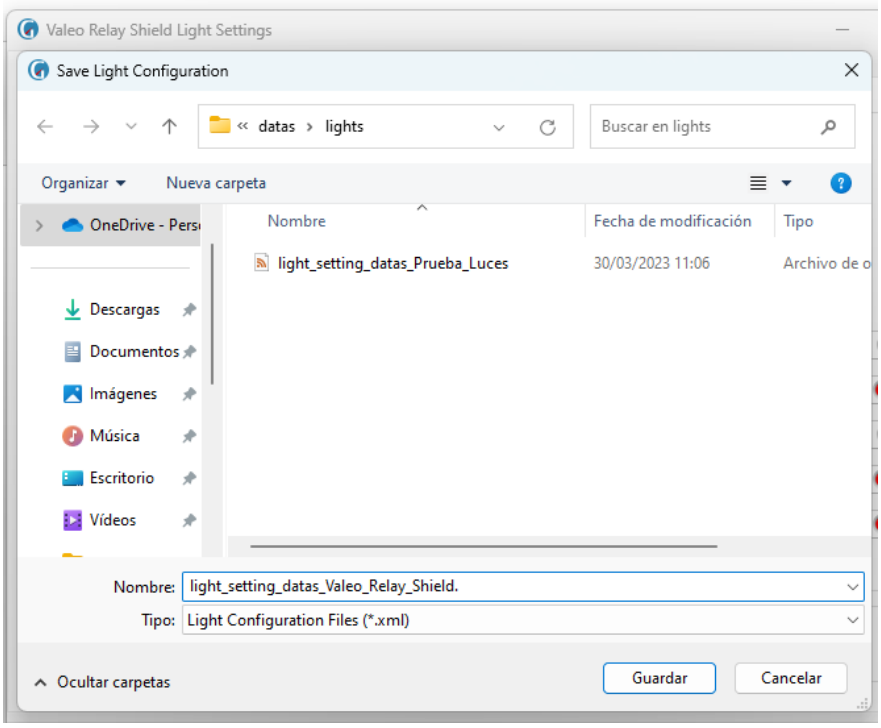

Figure 6.25 – *Open light controller: step four*

6

6.1.8 RF.8

Description	RF.8 The software must allow the user to change between white and black theme
Analysis	Requirement satisfied by redesigning the corresponding methods, where we now have the incorporation of css files that give appearance and style to the interfaces.
Evaluation	Validated

Table 6.8 – *RF.8*

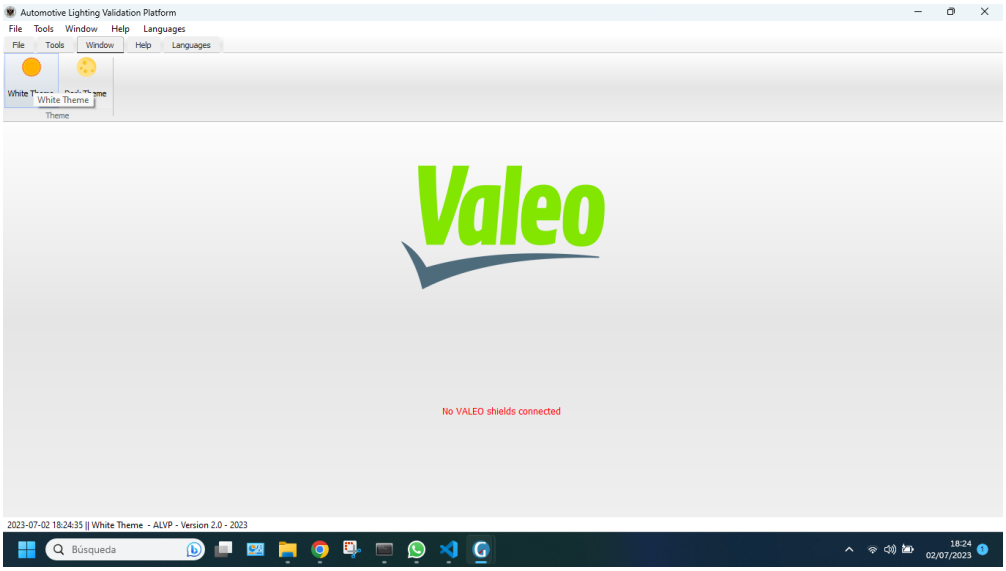

Figure 6.26 – *White theme*

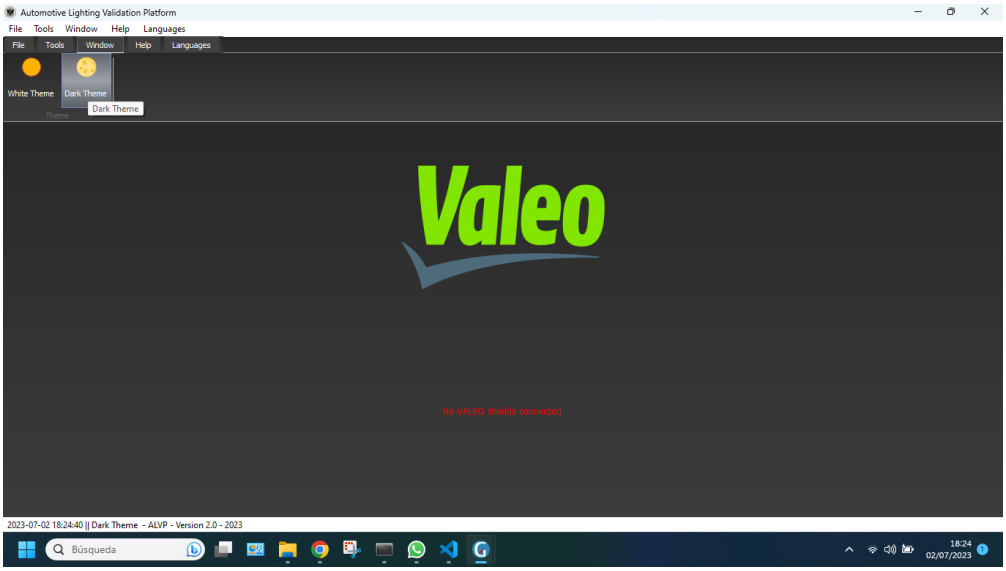

Figure 6.27 – *Dark theme*

6.1.9 RF.9

Description	RF.9 The software must allow the user check about additional information.
Analysis	Requirement satisfied by redesigning the corresponding window, which shows contact information.
Evaluation	Validated

Table 6.9 – *RF.9*

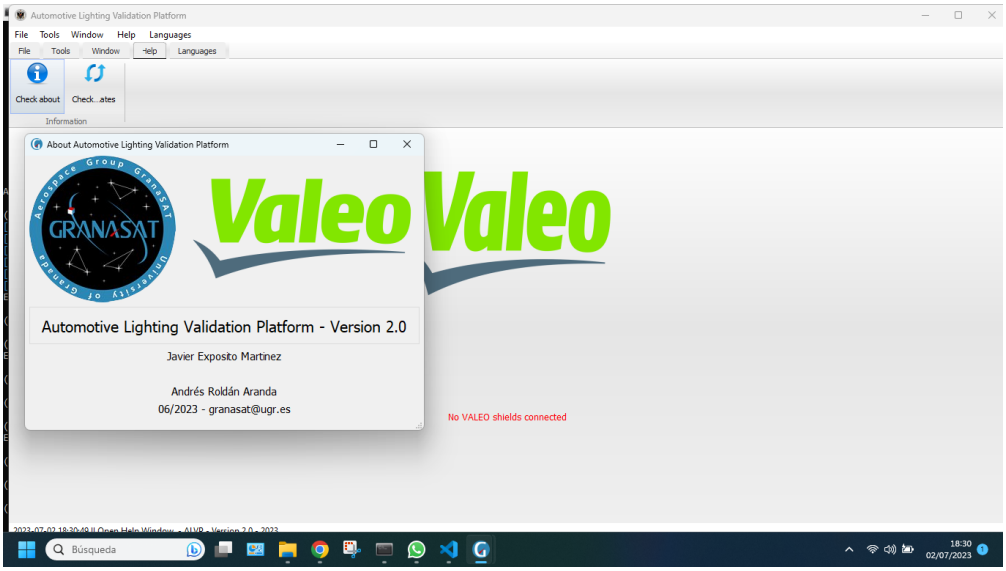

Figure 6.28 – *Check about*

6.1.10 RF.10

Description	RF.10 The software must allow the user check updates and update the application if it is necessary
Analysis	Requirement satisfied by creating methods and classes that manage the entire process of querying and updating application versions. Additionally, the file stored in the server has been updated with the current version of the application.
Evaluation	Validated

Table 6.10 – *RF.10*

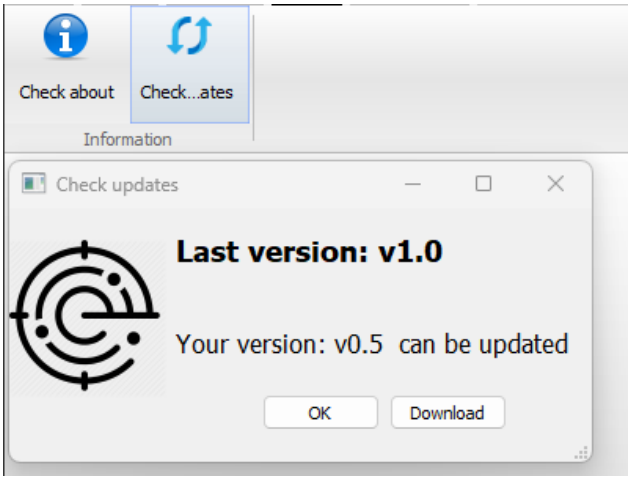

Figure 6.29 – *Check updates*

6.1.11 RF.11

Description	RF.11 The software must allow the user to select one of several languages: english, spanish, and french
Analysis	Requirement satisfied by creating an action in the menu, a new class, modifying and creating some methods and creating a new window. In addition to the creation of .xml files, one per language.
Evaluation	Validated

Table 6.11 – *RF.11*

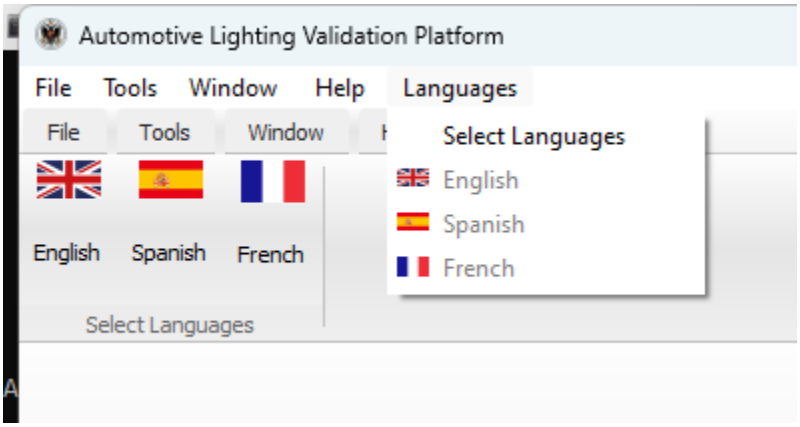

Figure 6.30 – *Select Languages: step one*

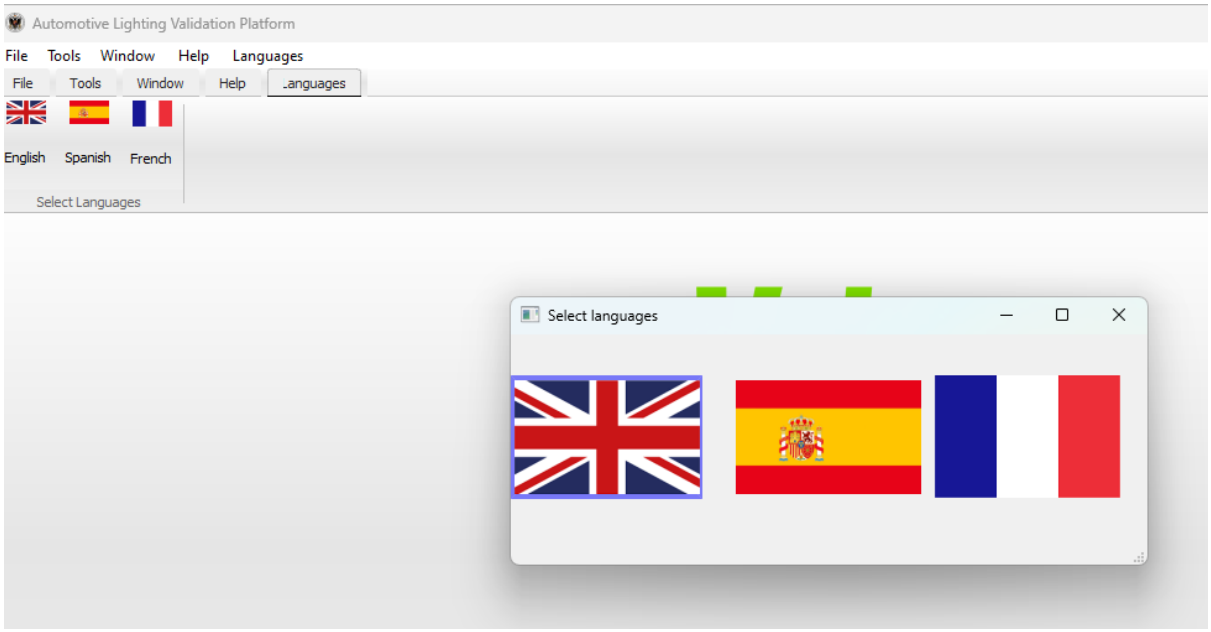

Figure 6.31 – *Select Languages: Step two*

6.1.12 RF.12

Description	RF.12 The software must allow the user to create a DUT file since EMC test
Analysis	Requirement satisfied by creating the method which manages the creation of the DUT process from the DefineEMCTest class.
Evaluation	Validated

Table 6.12 – *RF.12*

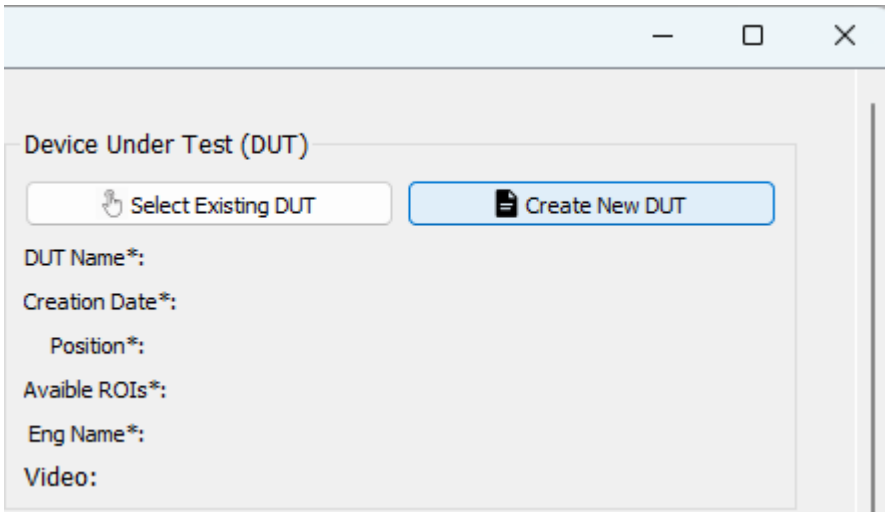

Figure 6.32 – *new DUT from EMC: step one*

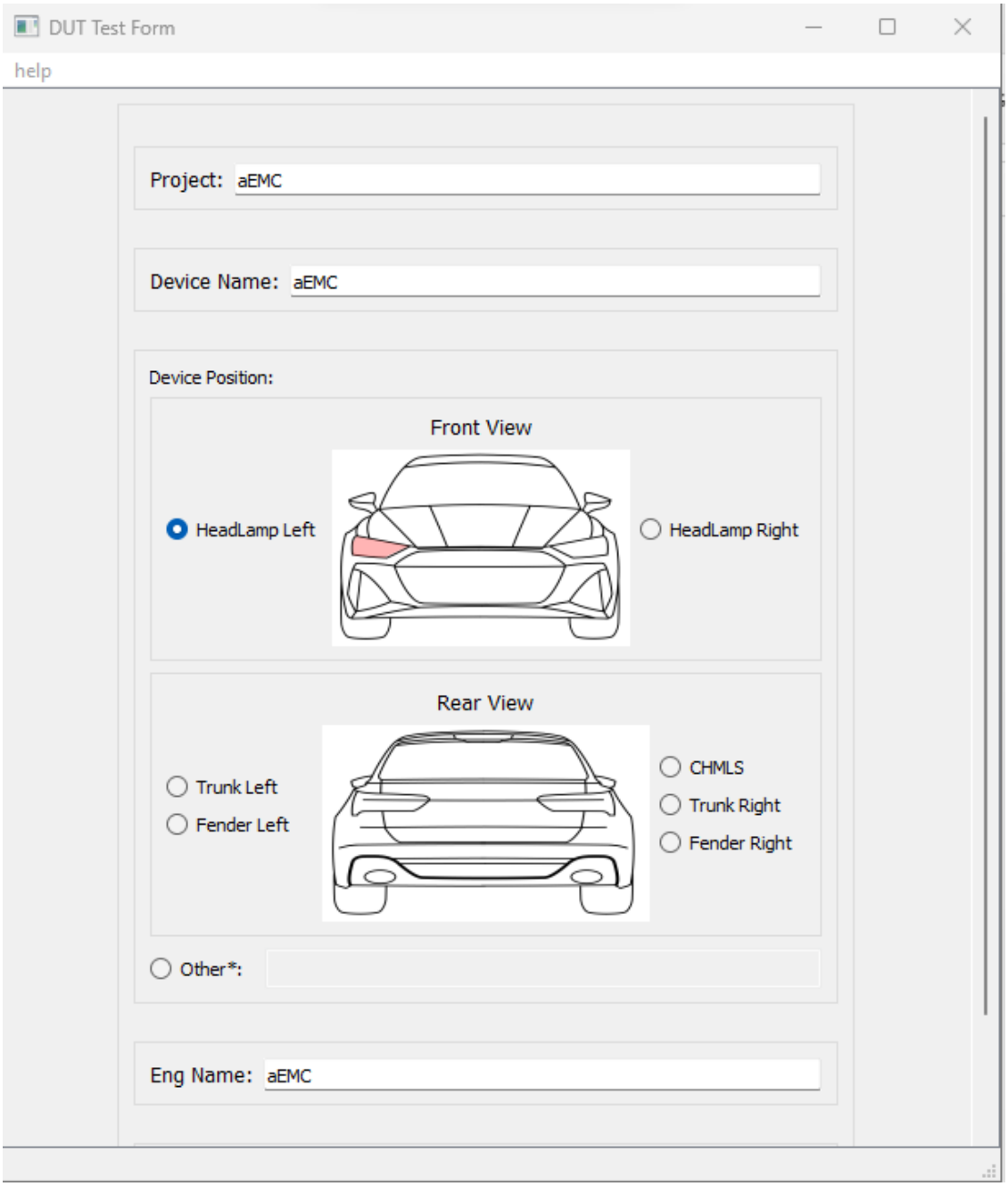

Figure 6.33 – *new DUT from EMC: step two*

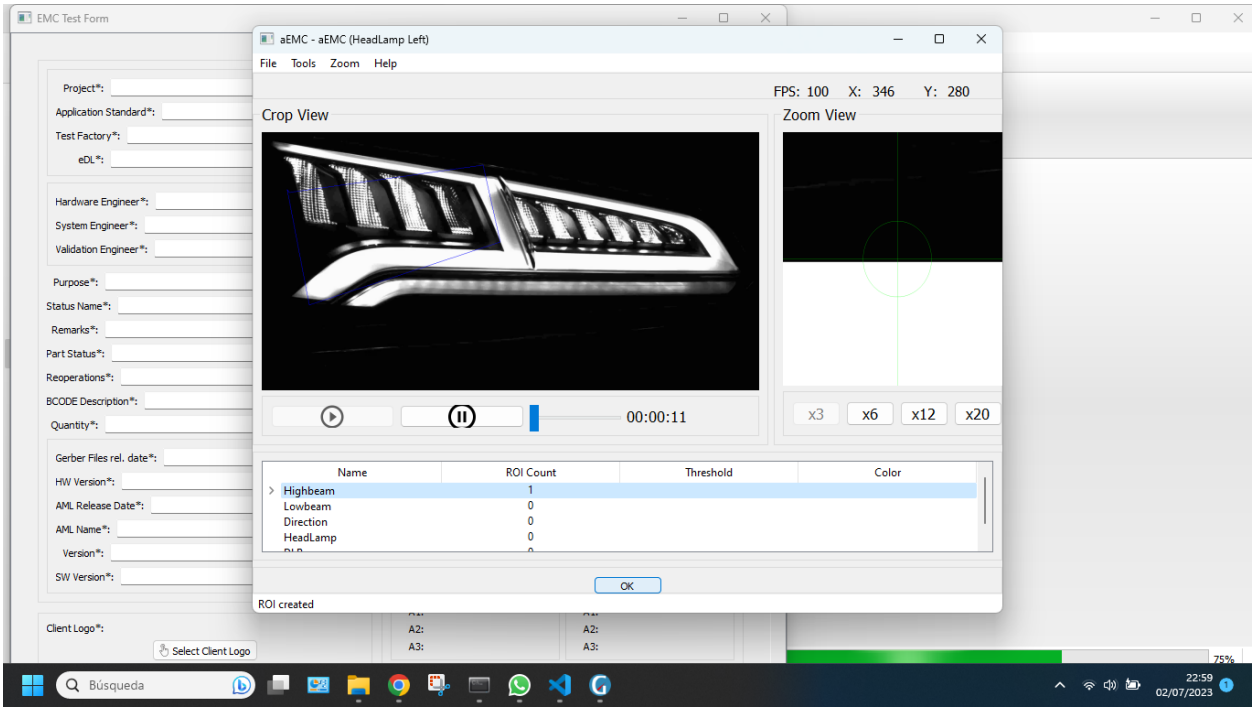

Figure 6.34 – *new DUT from EMC: step three*

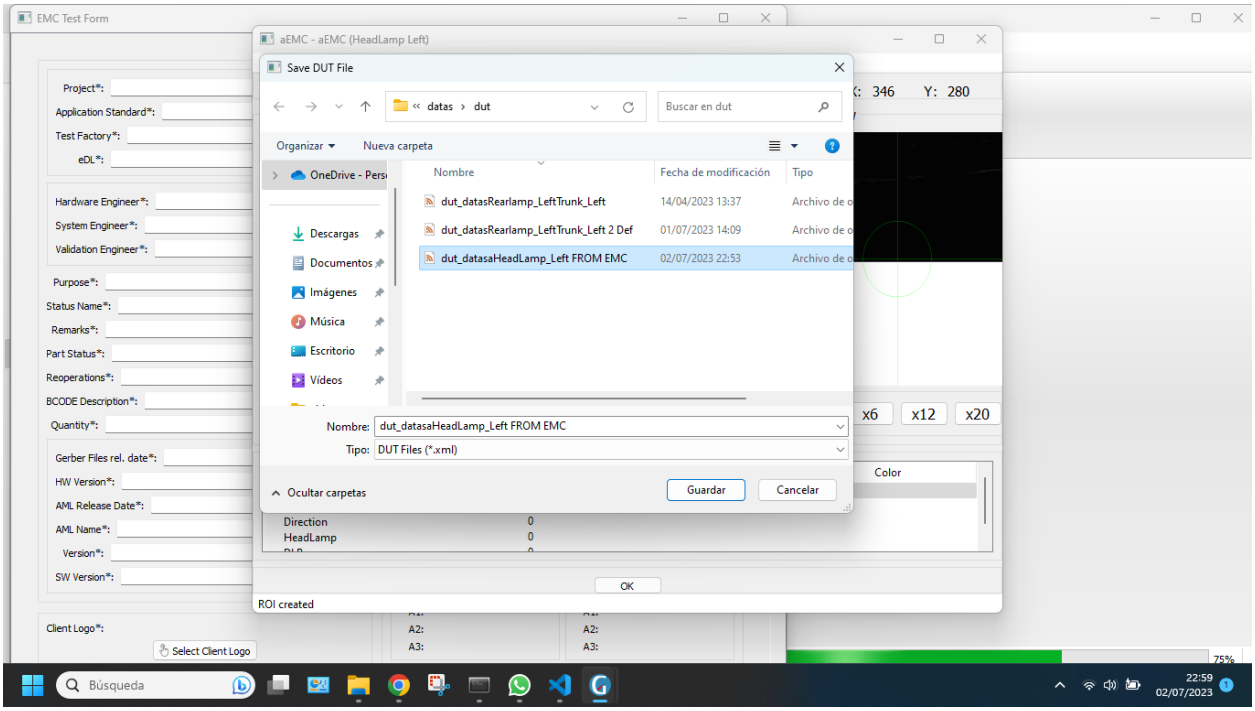

Figure 6.35 – *new DUT from EMC: step four*

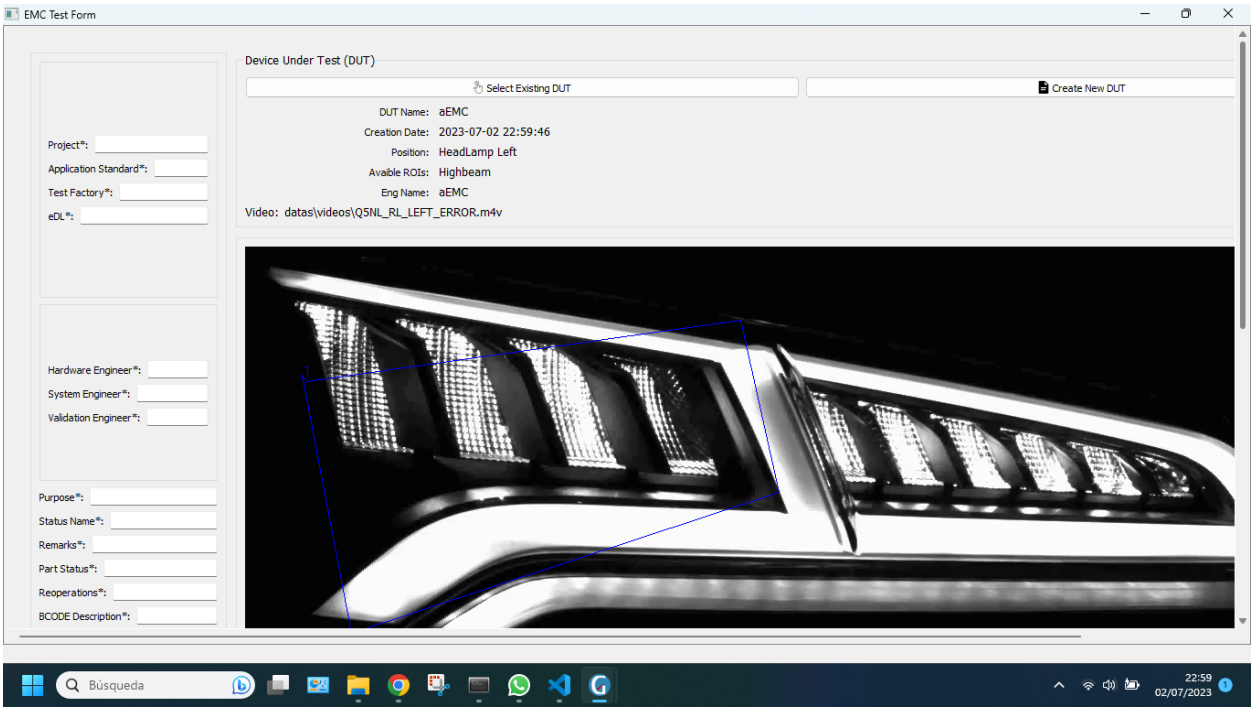

Figure 6.36 – *new DUT from EMC: step five*

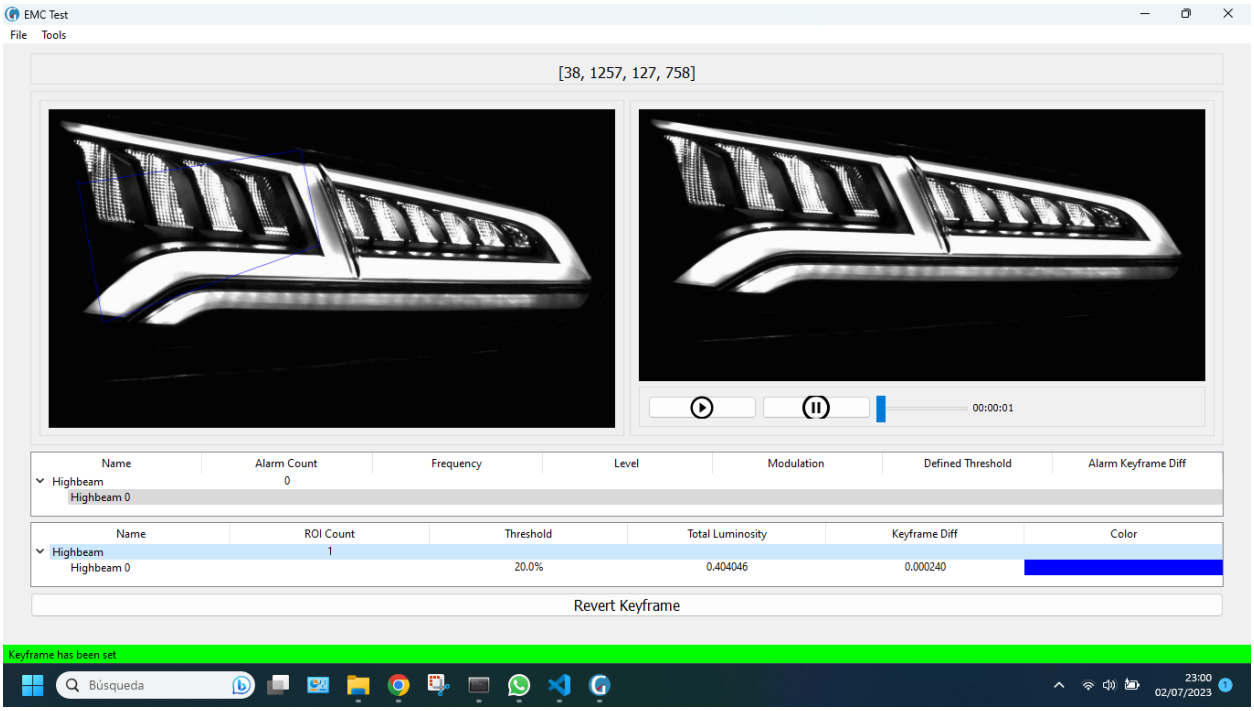

Figure 6.37 – *new DUT from EMC: step six*

6.1.13 RF.13

Description	RF.13 The software must allow the user to create a light configuration file since EMC test
Analysis	Requirement satisfied by creating the method which manages the creation of the light configuration file from the DefineEMCTest class.
Evaluation	Validated

Table 6.13 – *RF.13*

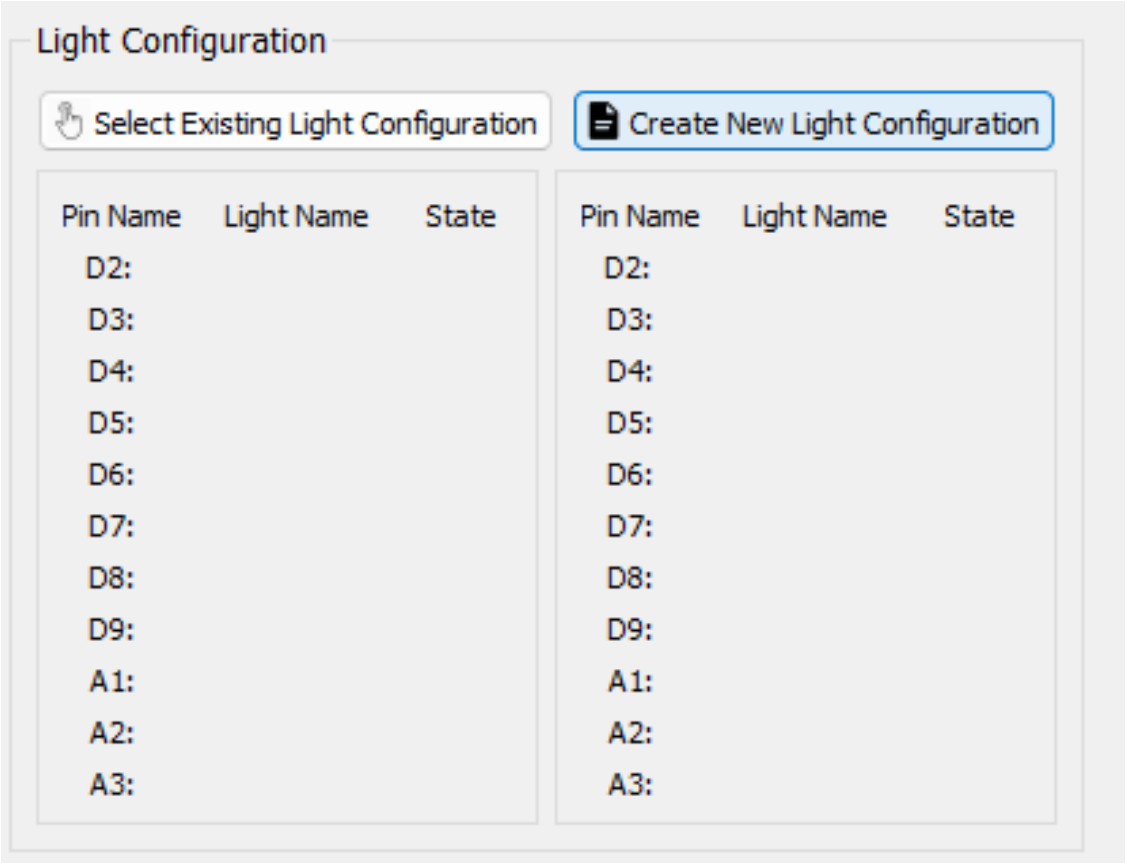

Figure 6.38 – *new Light configuration from EMC: step one*

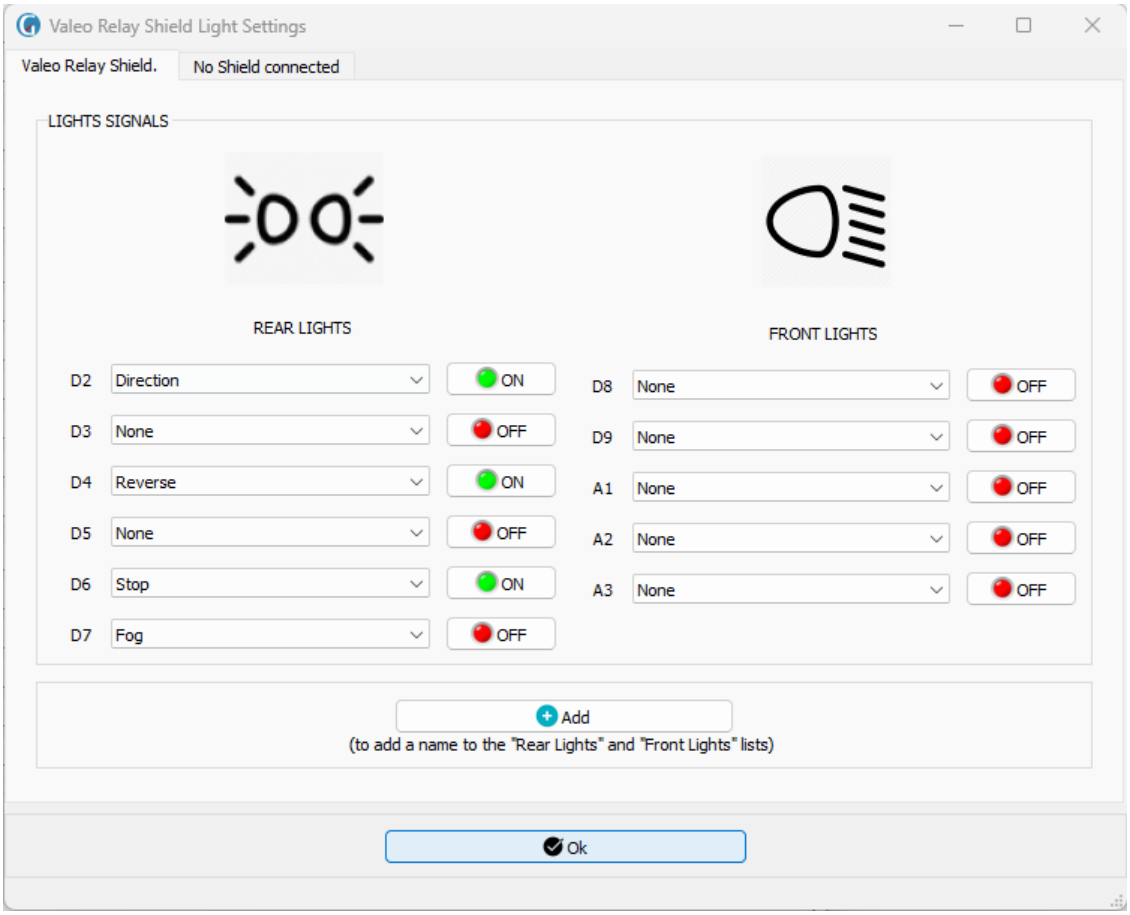

Figure 6.39 – *new Light configuration from EMC: step two*

6

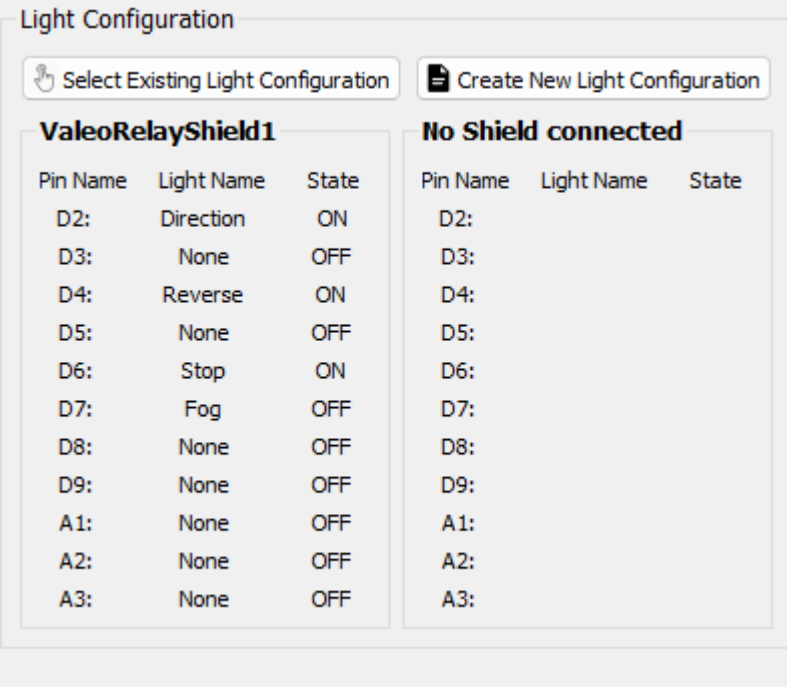

Figure 6.40 – *new Light configuration from EMC: step three*

6.1.14 RF.14

Description	RF.14 The software must allow the user to generate a report.
Analysis	Requirement satisfied by creating a new action in the menu, a class that manages the whole process to create the report, as well as creating a new template for the report.
Evaluation	Validated

Table 6.14 – *RF.14*

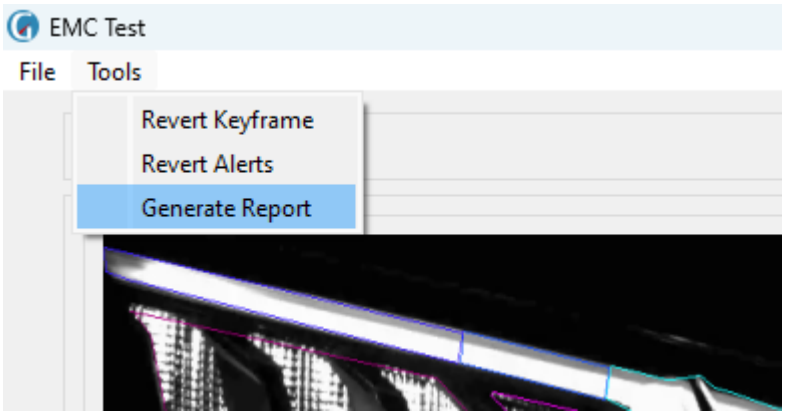

Figure 6.41 – *Generate report: step one*

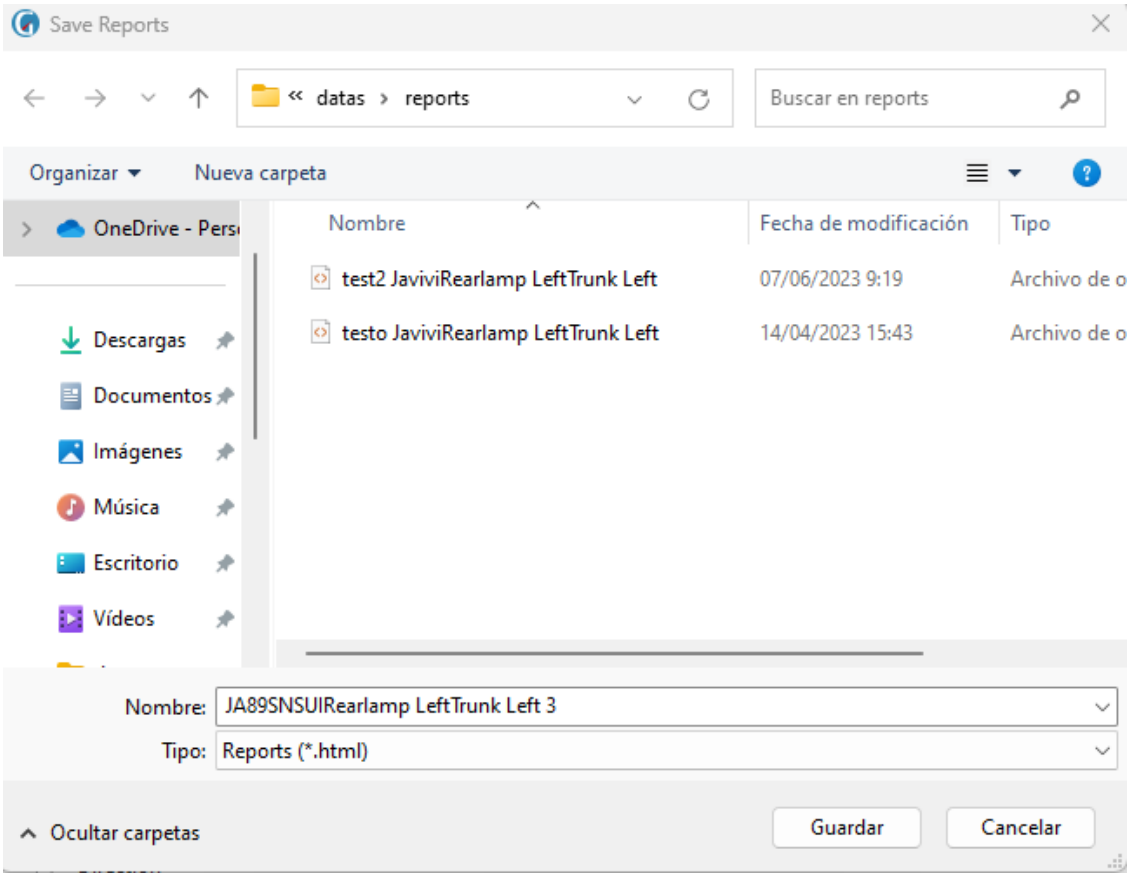

Figure 6.42 – *Generate report: step two*

6.2 Non-Functional Requirements

6.2.1 NRF.1

Description	NRF.1 The software must be able to detect when an Arduino is disconnected, at any time.
Analysis	Requirement satisfied by creating a timer that calls every second a method that checks the disconnection of the Arduinos.
Evaluation	Validated

Table 6.15 – *NRF.1*

Figure 6.43 – *Arduino disconection*

6.2.2 NRF.2

Description	NRF.2 The software must be able to reconnect an Arduino for a short period of time since it was disconnected.
Analysis	Requirement satisfied by creating a timer that is triggered when an arduino is disconnected, calling every second a method that reconnects the arduinos and checks if they have been reconnected correctly.
Evaluation	Validated

Table 6.16 – *NRF.2*

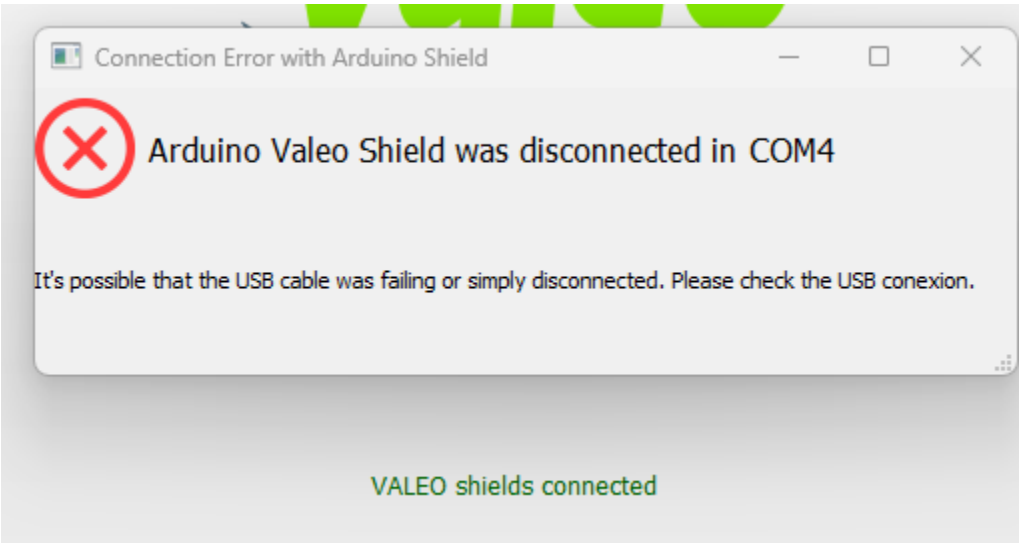

Figure 6.44 – *Arduino connection after its disconection*

6.2.3 NRF.3

Description	NRF.3 The software must be able to work correctly without an Arduino connected.
Analysis	Requirement satisfied by modifying variables and methods to prevent the application from aborting.
Evaluation	Validated

Table 6.17 – *NRF.3*

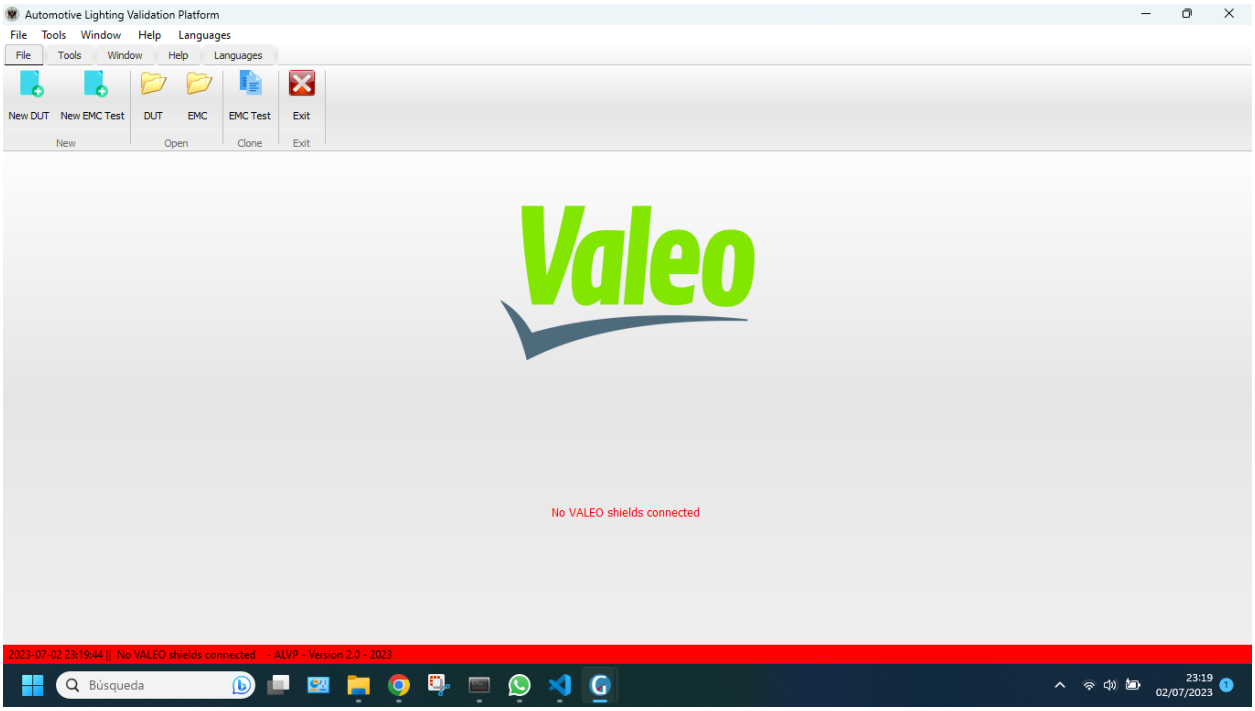

Figure 6.45 – *Application without Valeo Relay Shields*

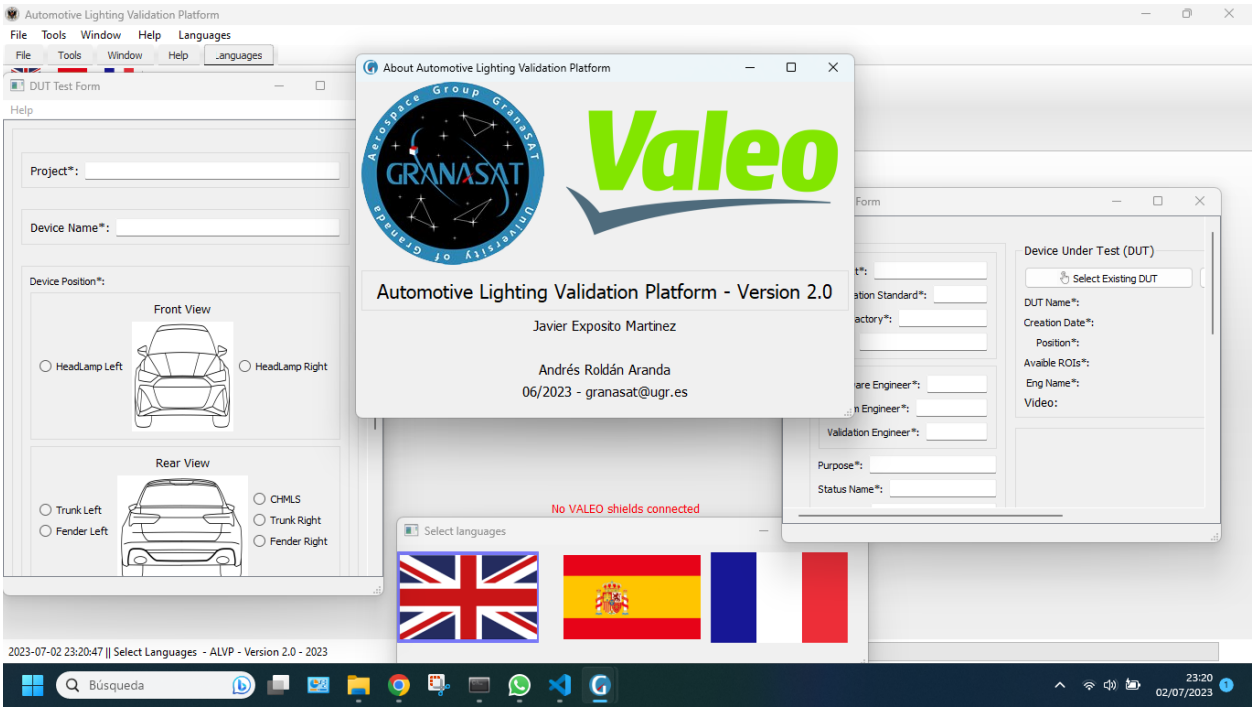

Figure 6.46 – *Application without Valeo Relay Shields 2*

6.2.4 NRF.4

Description	NRF.4 The software must be able to save the last configuration which it had when it was closed and starts with this.
Analysis	Requirement satisfied by creating write and read methods from an xml file, called configuration file. As well as the creation of variables that store the data we want to save.
Evaluation	Validated

Table 6.18 – *NRF.4*

6.2.5 NRF.5

Description	NRF.5 All software's interfaces must be responsives.
Analysis	Requirement satisfied by modifying some interfaces, and creating new ones using QtDesigner.
Evaluation	Validated

Table 6.19 – *NRF.5*

6.2.6 NRF.6

Description	NRF.6 The software must has a status bar which reports user's operations and application's status.
Analysis	Requirement satisfied by adding new actions and statuses to the status bar, changing its color depending on the action/status and creating new status bars in other windows. In addition, each action has the text with its respective languages.
Evaluation	Validated

Table 6.20 – *NRF.6*

Figure 6.47 – *Statusbar - No arduino detect*

Figure 6.48 – *Statusbar - New DUT*

Figure 6.49 – *Statusbar - Error opening video*

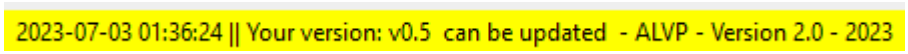

Figure 6.50 – *Statusbar - No arduino detect*

6.2.7 NRF.7

Description	NRF.7 The software must be save datas in xml files, unlike before, it was done in txt format.
Analysis	Requirement satisfied by creating new save files in .xml format and using a library to write to and read tags from these files.
Evaluation	Validated

Table 6.21 – *NRF.7*

6.2.8 NRF.8

Description	NRF.8 In DUT and EMC tests' forms, done's bottoms must be disable when user delete any required field. Done's bottoms only must be enabled when all required fields are filled.
Analysis	Requirement satisfied by creating and modifying attributes and methods belonging to these classes.
Evaluation	Validated

Table 6.22 – *NRF.8*

6.2.9 NRF.9

Description	NRF.9 In DUT test's forms, when the user selects a car spotlight, it should highlights.
Analysis	Requirement satisfied by creating and modifying attributes and methods belonging to these classes. Qt objects such as Qpen or QPainterPath are used.
Evaluation	Validated

Table 6.23 – *NRF.9*

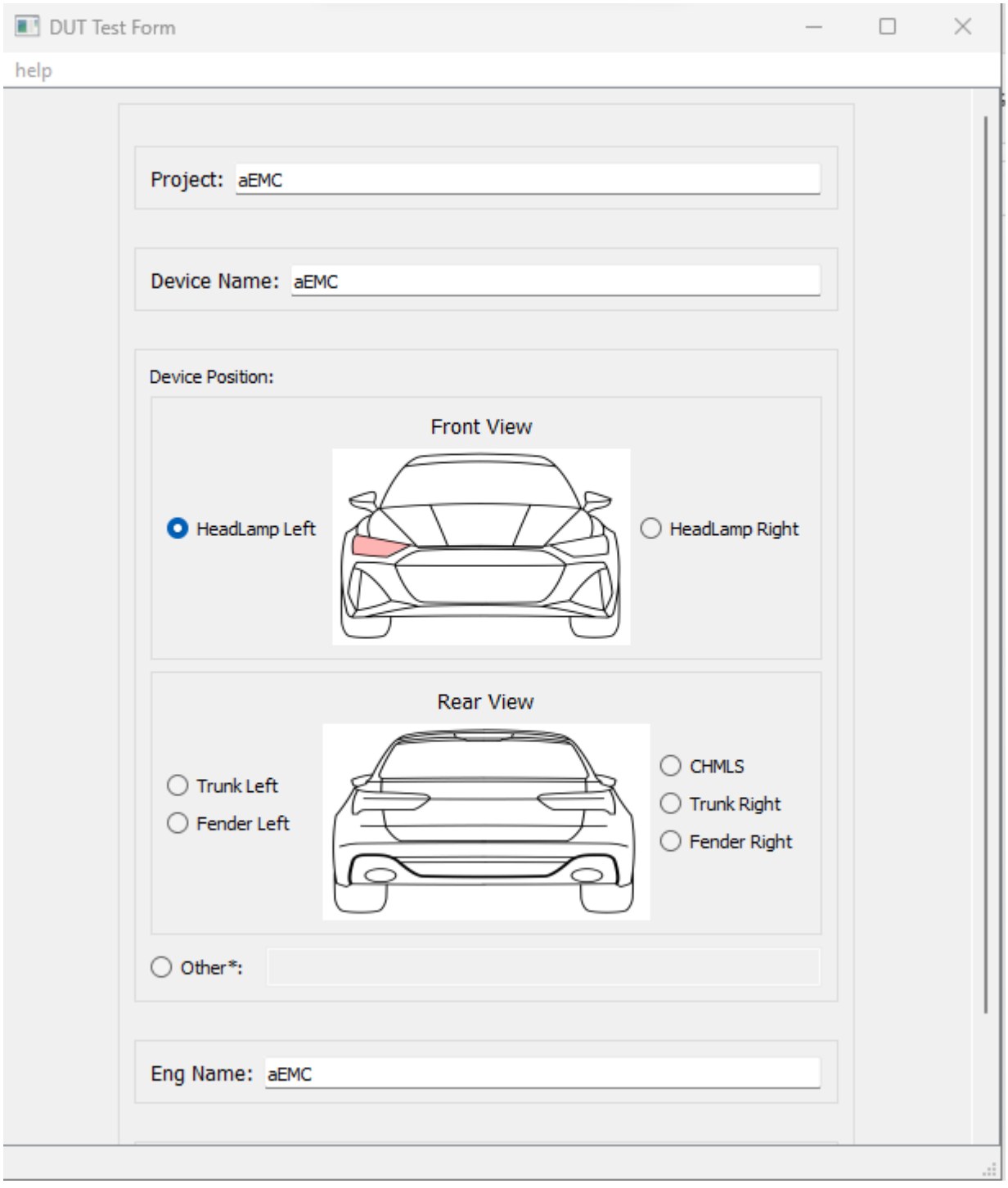

Figure 6.51 – *Select a car spotlight*

6.2.10 NRF.10

Description	NRF.10 When the user open DUT of EMC file, in DUT and EMC tests' forms, must appear a new field which contains the video's path what was used in test.
Analysis	Requirement satisfied by modifying the interfaces that show the forms, added to the creation and modification of the methods and attributes in charge of storing and showing the label with the video path. In addition, the video path is now saved in the saving files.
Evaluation	Validated

Table 6.24 – *NRF.10*

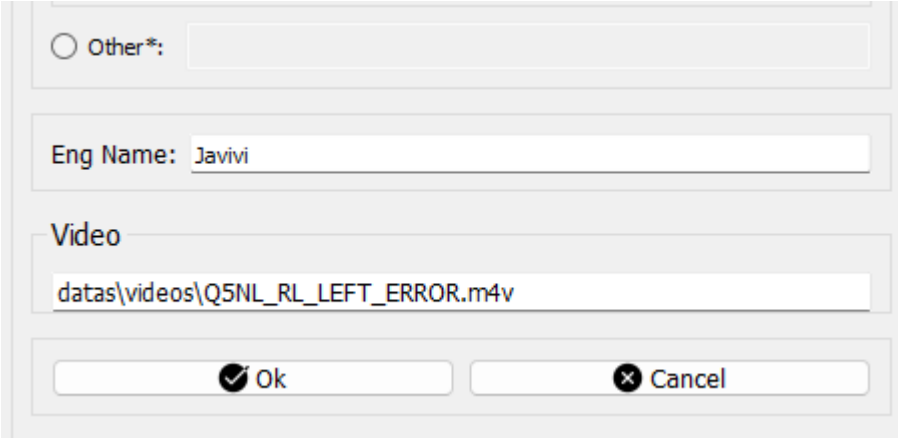

Figure 6.52 – *Video path in DUT and EMC Test*

6

6.2.11 NRF.11

Description	NRF.11 When the user open DUT file,crop points must be show in window where the user crop the image.
Analysis	Requirement satisfied by the creation and modification of methods that read the data and store it in the attributes that are used to draw and store the crop points.
Evaluation	Validated

Table 6.25 – *NRF.11*

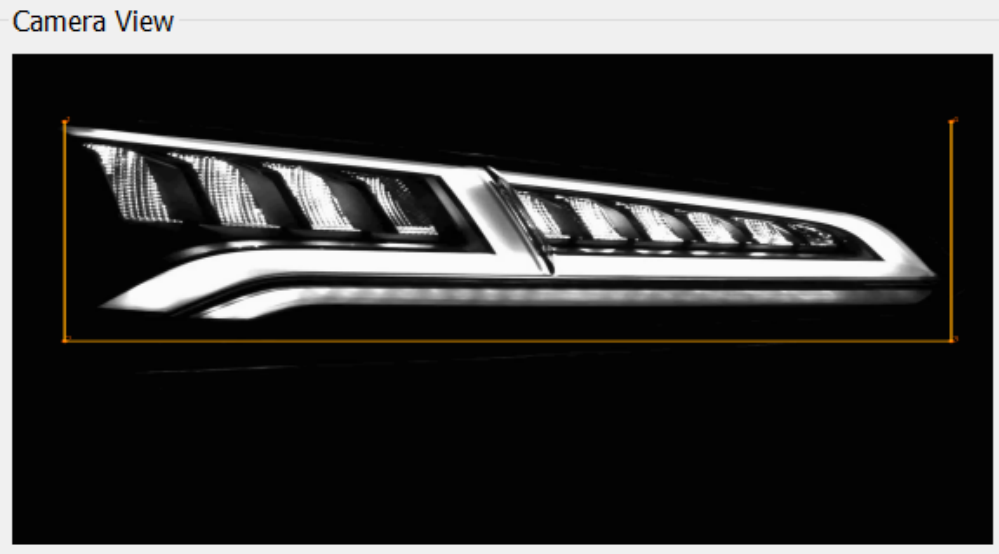

Figure 6.53 – *Crop points in camera view*

6.2.12 NRF.12

Description	NRF.12 When the user open DUT file, ROIs must be show in crop window and create the ROIs.
Analysis	Requirement satisfied by the creation and modification of methods that read the data and store it in the attributes that are used to draw and store the ROIs
Evaluation	Validated

Table 6.26 – *NRF.12*

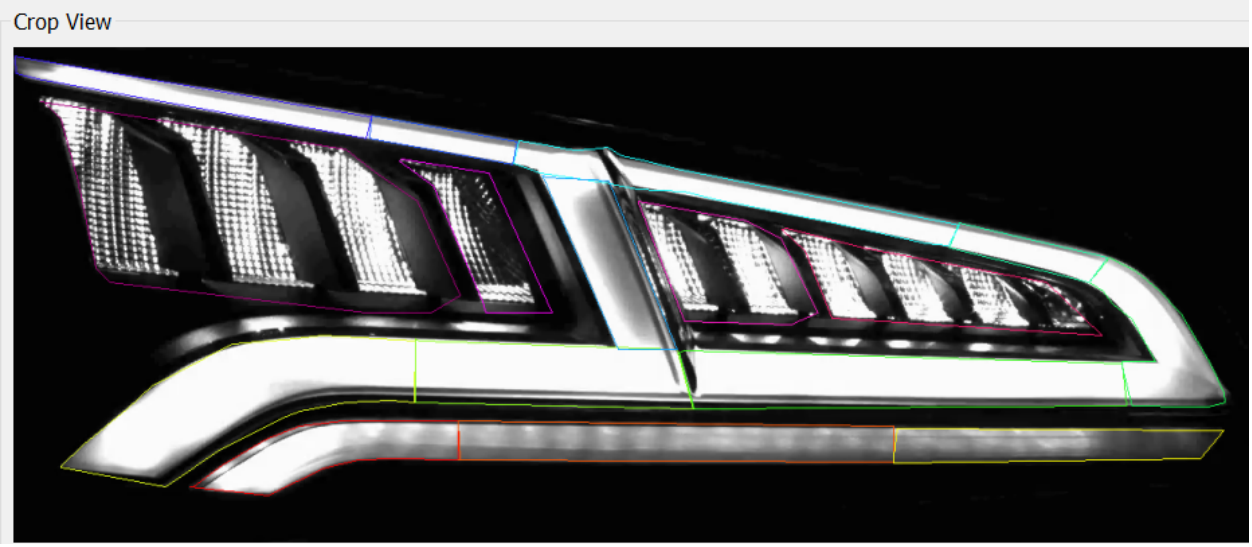

Figure 6.54 – *ROIs points in crop view*

6.2.13 NRF.13

Description	NRF.13 In DUT Test, in windows where the user crop the image and create the ROIs, the software must be able to separate the camera view and zoom view, allowing to work with two or even three screens.Also in EMC Test, for the class in charge of performing the EMC analysis.
Analysis	Requirement satisfied by the creation of methods and variables that allow this requirement to be met.
Evaluation	Validated

Table 6.27 – *NRF.13*

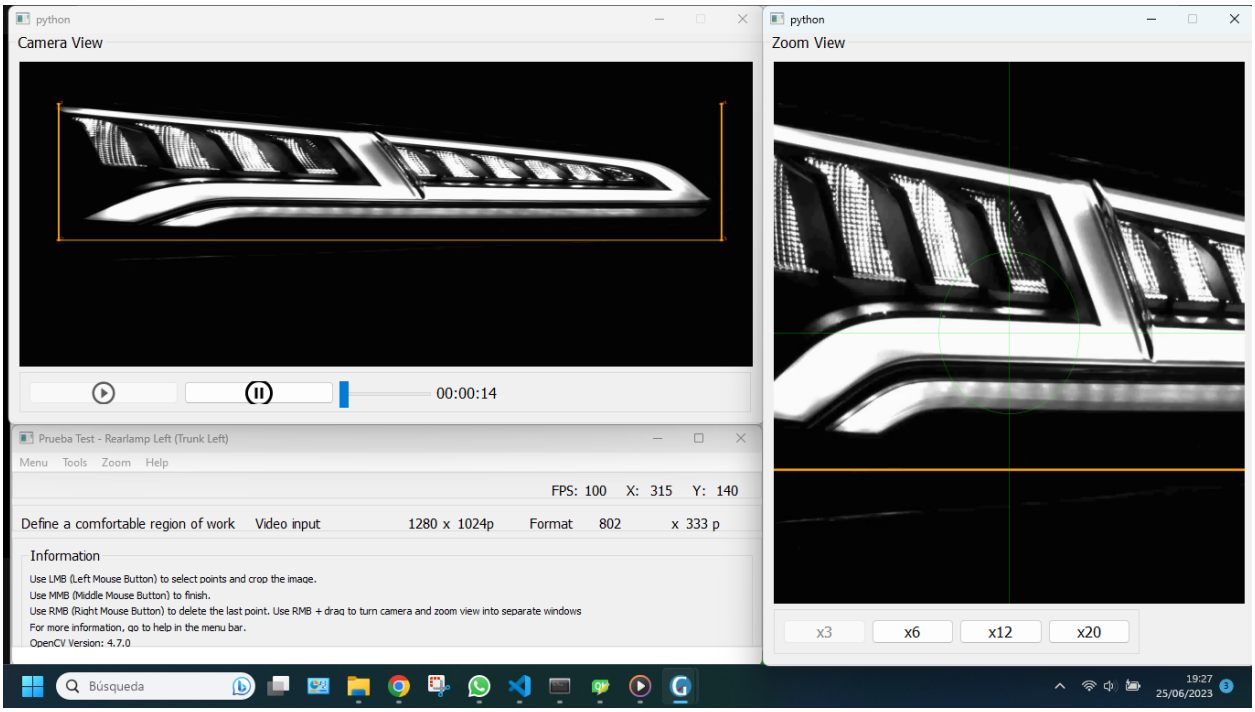

Figure 6.55 – *Separate views*

6.2.14 NRF.14

Description	NRF.14 In EMC Test,when the analysis starts, if the user select a ROI, this must be highlight in Keyframe view.
Analysis	Requirement satisfied by creation and modification of methods and variables that allow this requirement to be met.
Evaluation	Validated

Table 6.28 – *NRF.14*

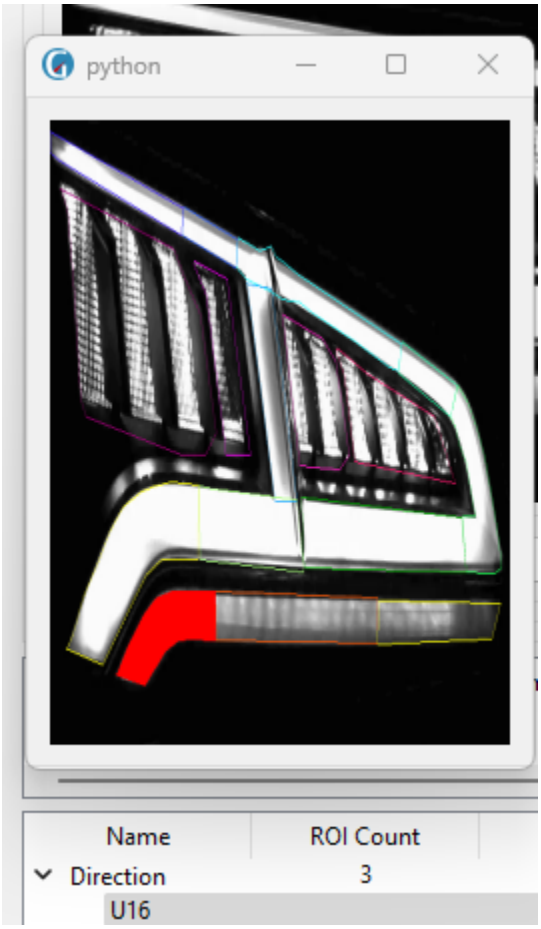

Figure 6.56 – *Show a selected ROI in EMC analysis*

6.2.15 NRF.15

Description	NRF.15 In EMC Test,when the analysis starts, if the user select an alert, this must be shown.
Analysis	Requirement satisfied by creation and modification of methods and variables that allow this requirement to be met.A timer is created whose functionality is to show the alert in the form of a "gif"
Evaluation	Validated

Table 6.29 – *NRF.15*

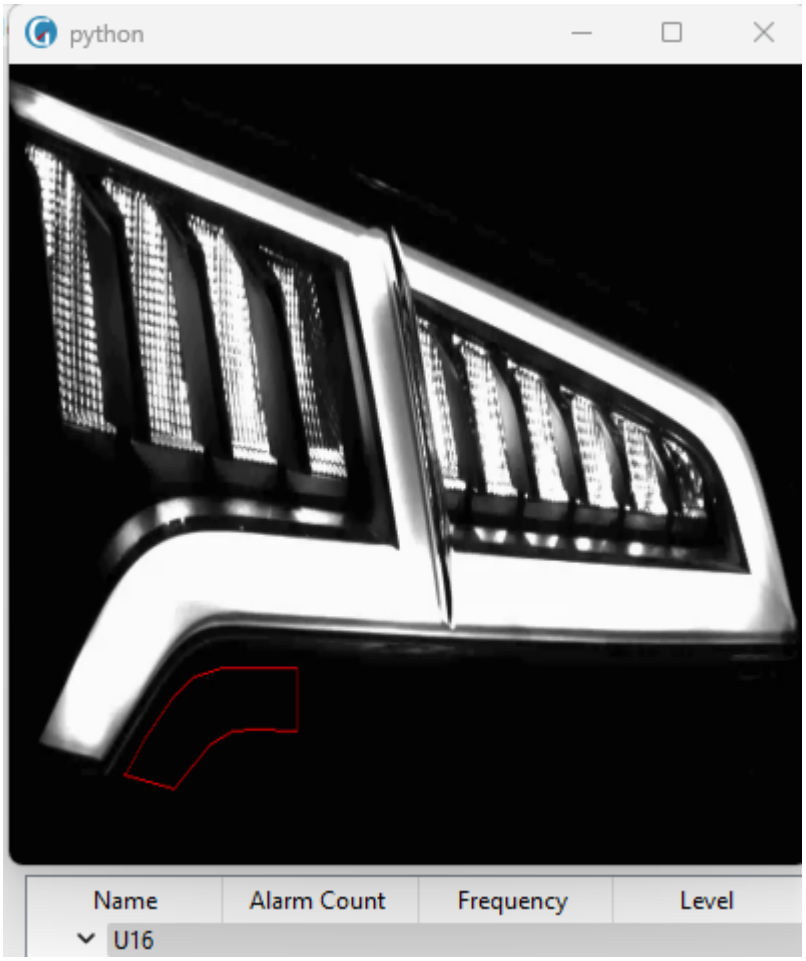

Figure 6.57 – *Show a selected alert in EMC analysis*

6

6.2.16 NRF.16

Description	NRF.16 The software must be able to add a graph which shows the lumination's difference between Keyframe and video for each ROI in the generated report.
Analysis	Requirement satisfied by creation and modification of methods and variables that allow this requirement to be met.In addition, the template that is used to generate the report must also be modified.
Evaluation	Validated

Table 6.30 – *NRF.16*

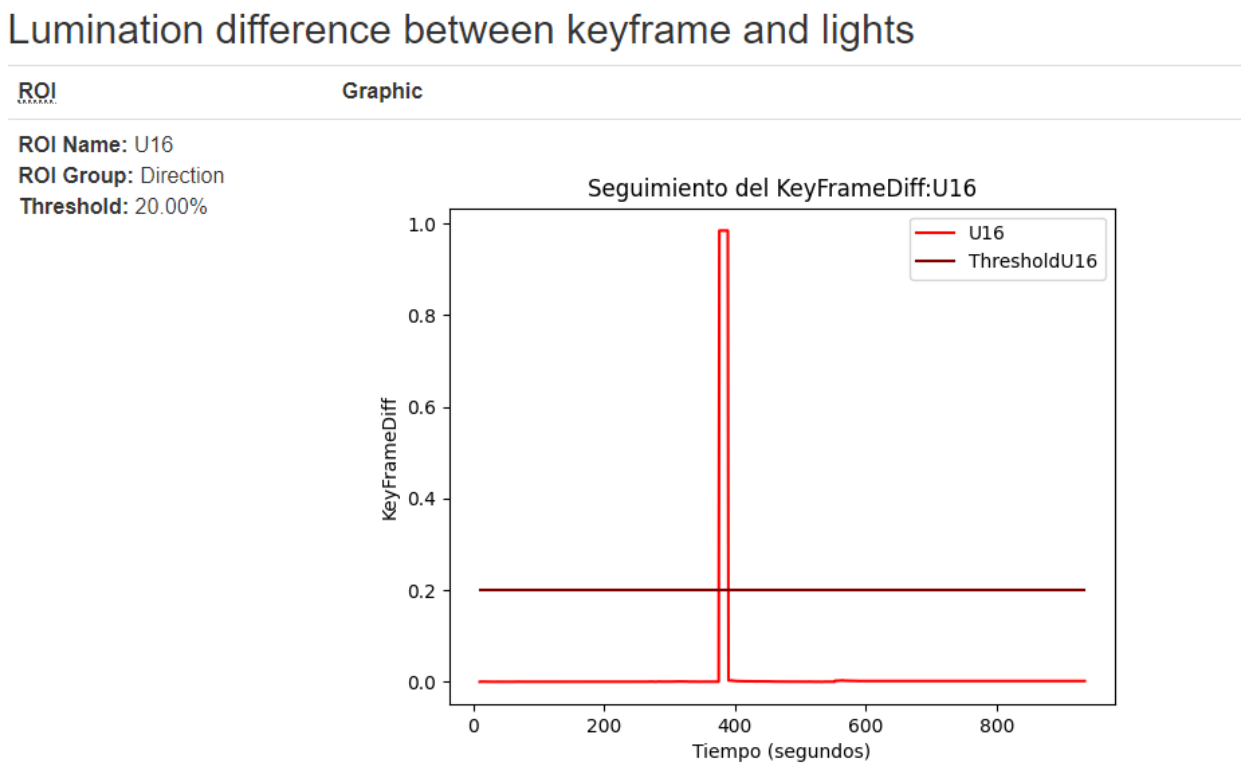

Figure 6.58 – *Diagram of difference of ilumination*

6.2.17 NRF.17

Description	NRF.17 The software must read and process the videos efficiently and quickly.
Analysis	Requirement satisfied by by simplifying code and using Threads that allow code concurrency.
Evaluation	Validated

Table 6.31 – *NRF.17*

Chapter 7

Conclusions and future lines

7.1 Conclusions

In this document it has tried to explain in detail the installation, operation of the application, as well as all the evolution and differences with respect to its previous version.

The beginnings, above all, were very hard, not only because of the lack of documentation and file organization, or because the libraries used by the application as PyQt5 were totally unknown to me at the beginning, but also because during the degree in Computer Engineering does not teach to create large projects that resemble real projects developed in a company.

From the beginning and throughout the development of the application, it has been a challenge, as everything except the programming language (Python) was new and unfamiliar.

For example, learning how to create the application using the interfaces created by QtDesigner and understanding the code that managed the interface with the operation of the application, while understanding how the application worked, was one of the most complicated and problematic things I had to solve. Another challenge was to ensure the correct migration from Python 2.7 to version 3.10 and higher, as part of the application became unusable due to method calls that did not work. There were also problems with reading and opening saved files, understanding and resolving window resizing and resolution for video display during testing, learning to use Threads, etc.

Today, I can conclude that the application is not only 100% functional, but includes a number of enhancements that take it to the next level, meeting all of the client's requirements in the process.

This project has not only been a demonstration of the knowledge and skills acquired in the computer engineering degree but also a continuous and practical learning on how to develop an application in Python.

As a final comment, I am proud of the work done and the outcome of this project, despite the difficulties and frustrations encountered during its development. I think it is a very good final product, and I was up to the challenges that were presented to me. I am proud of all the skills and knowledge acquired, which I am sure will be very useful for the future.

7.2 Proposed future upgrades

Although this project has met the requirements proposed by the client, there is a list of upgrades that could be made to the application:

7.2.1 Improve source code

Although the main classes were simplified, reducing in some cases to half the number of lines of code and separated into methods, the code is chaotic, the names of some methods and variables are in English and others in Spanish. Some methods do not have a clear function, other methods could simply avoid being created with a better programming approach. More classes should be created with functions or methods common to many classes, instead of writing the same method in all classes.

Classes should be better organized, being clear about the tributes and methods needed and their visibility (private, public minimum). For example, I have had to remove many attributes of classes that were not used or were really local variables of a particular method but not an attribute. In addition, the visibility of all attributes and methods of the classes is public, which to summarize, is a mistake that can lead to problems of design, maintenance and security of the application.

7.2.2 Complete the application

Although new functionalities have been implemented, there are certain incomplete actions in the application. For instance, enhancing the status bar by adding messages about the application's state or indicating missing actions. Another example is programming the behavior when performing a DUT test and selecting "Other" in the "Device Position" field.

7.2.3 Improvement in the structure and control of the application.

The main improvement is a change in the application's structure. Currently, the structure is fine; it has been updated (Application's Structure) to fix errors and incorporate the newly created classes. However, I don't consider it to be optimal. For example, since the application can function without Valeo Relay Shield devices initially, it doesn't make sense for the application to go through the Valeo Relay Shield detection process, causing the client to wait for a few seconds when launching the application.

It would also be more suitable to have the ability to connect Valeo Relay Shield devices whenever desired, using an action such as "Connect Valeo Relay Shields," rather than having only a few seconds to connect them, and requiring a restart of the application if not connected in time. The calls to the language class can be managed in a better way, etc. To provide a better illustration of all these points, let's proceed to show it graphically:

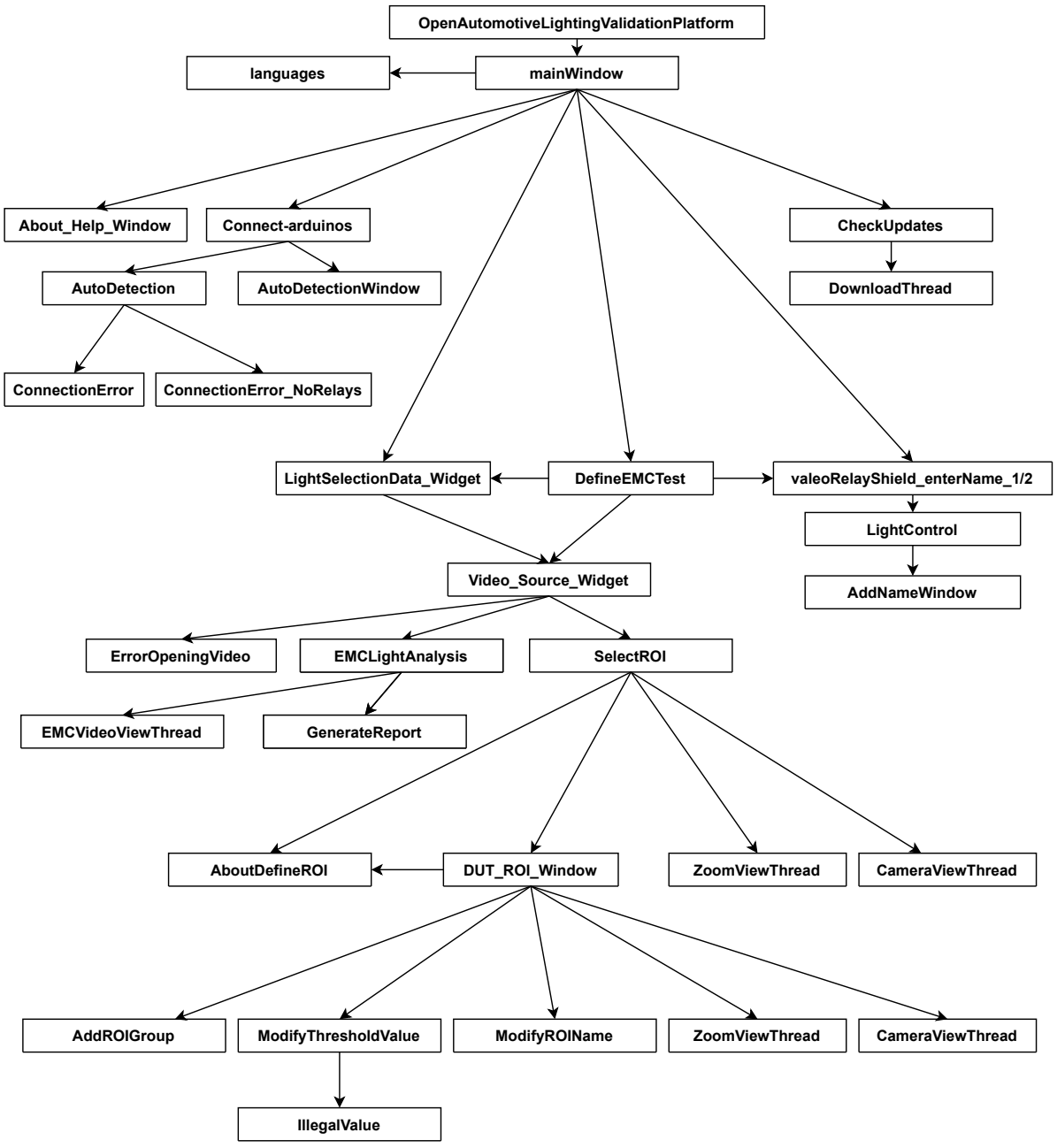

Figure 7.1 – *Future Upgrade Structure's Class Diagram*

7.3 Lessons learned

I wanted to conclude this project with a list of the lessons which I have learned during the development of this thesis:

- Migrate an application.
- Installing and configuring an Valeo Relay Shield board at a basic level.
- Use QtDesigner and the PyQt5 library to create the interfaces of an application.
- How to create, open and write to an .xml file.
- My knowledge about the OpenCV library has increased.
- To use the PyArmor and pytransform libraries to obfuscate, encrypt the code and create a license.
- To create an executable (.exe) of the application, as well as its installer.
- Learn about timers and Threads.
- Creation of an .html file with the template designer: jinja2 and how to integrate a photo and/or gif in this file.
- Learn by Reverse engineering how the application and libraries work and know how to apply it to the new version of the application.
- Study the different methods for implementation and choose the one which I think is the best for the incorporation of the device in the system.

7

Addenda

Appendix A

How to install application

A.1 Developers' installation

For developers, the installation process is as follows:

1. Create a virtual environment with Python version 3.10 or higher. In my case, I use the Anaconda environment, so to create the virtual environment, I use the command **conda create -n VirtualEnvironmentName python=3.10**.
2. Next, install the required dependencies by running **pip install -r requirements.txt**.
3. Once the dependencies are installed, the application can be launched smoothly. Open the terminal, navigate to the working directory using the command **cd workingpath**, and execute **python main.py**.

A.2 Clients' installation

For clients, the application installation process is as simple as opening the application installer and clicking on "Install."

Appendix B

Hardware Configuration: How to install and configure Valeo Relay Shields

The first thing to do is to download the application that we will use to configure it: `https://www.arduino.cc/en/software`. Next, We install it, open it (the necessary packages will be installed).Then we click on yes, to everything and connect the Valeo Relay Shield.

When connecting the Valeo Relay Shield to the computer, in the tools menu bar appears a section called ports (before connecting the Valeo Relay Shield did not appear). Select tools > ports > port: COM 3 (in my case), which is the only one that appears.

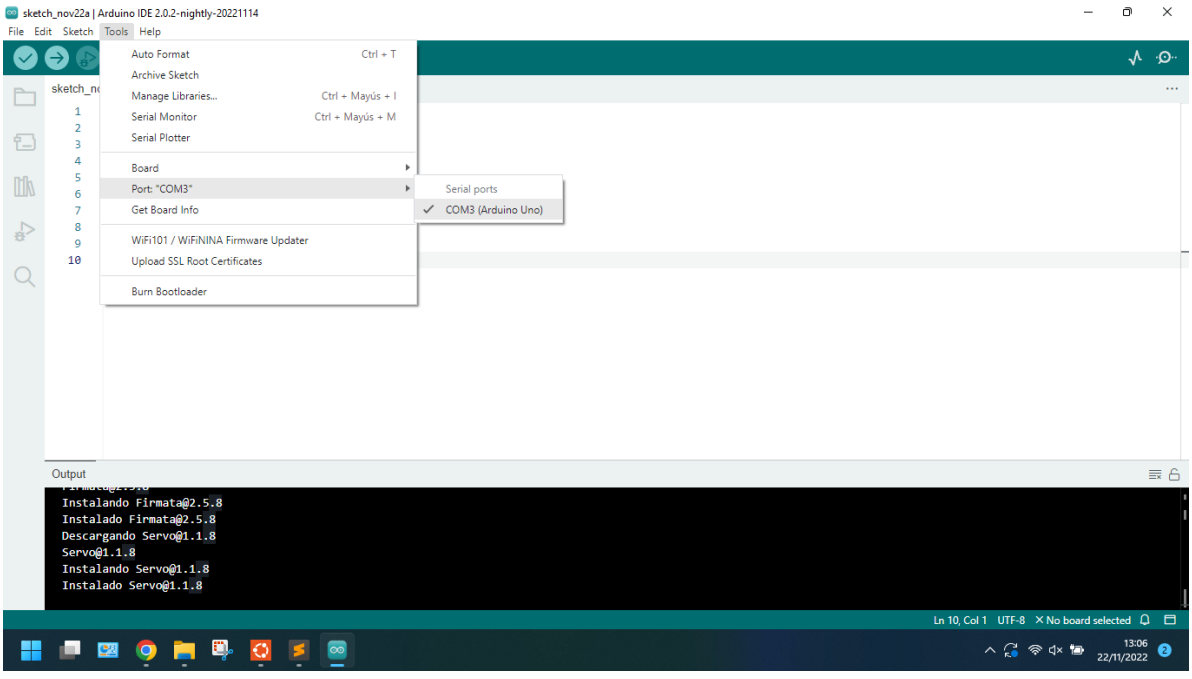

Figure B.1 – *How to set up Valeo Relay Shield: step One*

We also need to select a board, so we go to: tools > board > Arduino AVR boards > Arduino Uno.

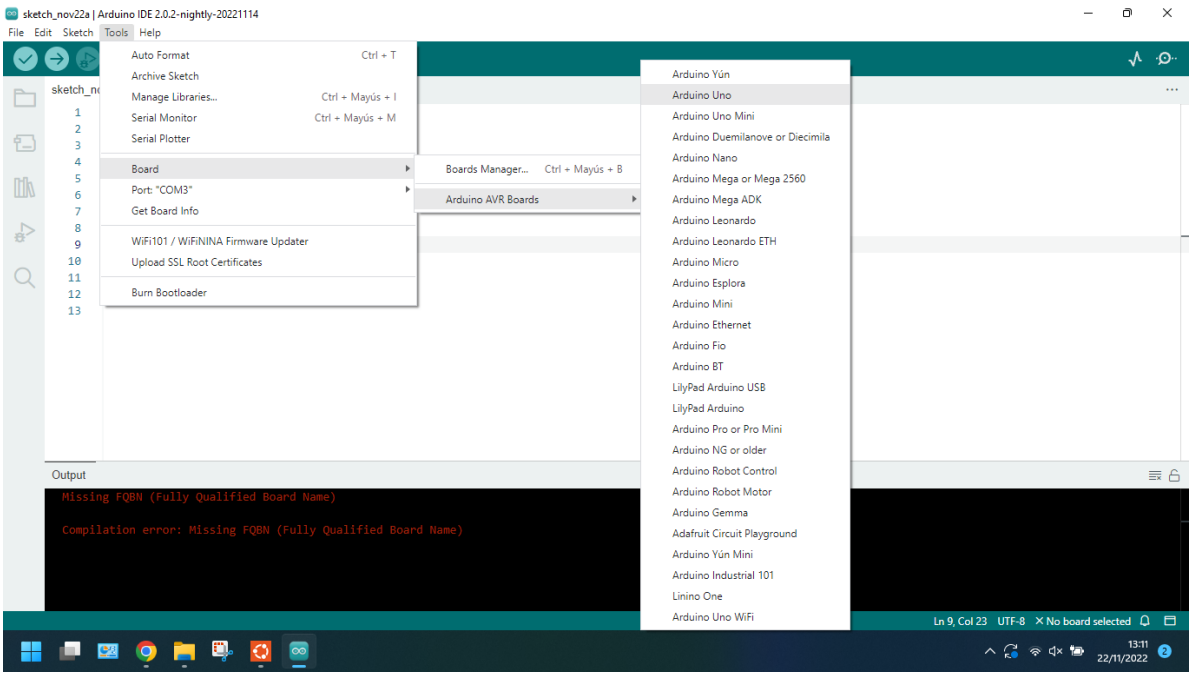

Figure B.2 – *How to set up Valeo Relay Shield: step Two*

Now we create a basic program to check if it works correctly or not, before configuring it for our application:

```
void setup() {
  // put your setup code here, to run on
  pinMode(13,OUTPUT);
}

void loop() {
  digitalWrite(13,HIGH);
  delay(1000)
  digitalWrite(13,LOW);
  delay(1000)

}
```

Figure B.3 – *How to set up Valeo Relay Shield: step Three*

The program turns on and off a led through port 13, every second.

Now we click on verify, located at the top, which has a tick/check symbol, which compiles and displays any syntactic errors in the code, as well as additional data that appears in the terminal:

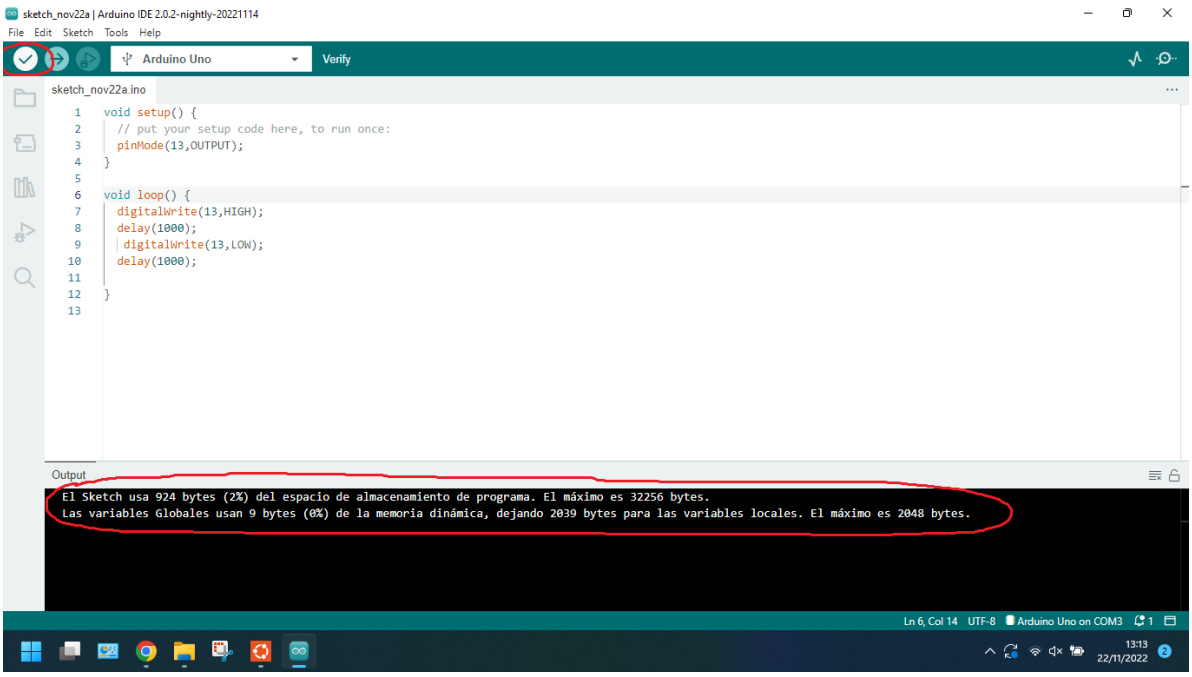

Figure B.4 – *How to set up Valeo Relay Shield: step four*

And finally, to run it, save the file and click on the small arrow located just to the right of the check mark symbol that we pressed earlier. Then, we can observe how the Valeo Relay Shield board's amber LED turns on and off.

(This basic program can be found within the application in the Valeo Relay Shield folder, along with the final version of the Valeo Relay Shield configuration).

Now, let's navigate to our application's Valeo Relay Shield folder and open the `"relay_shield_automated_serial_V0_3.ino"` file with the program. Press the tick/check symbol to verify that there are no errors and then execute it. This way, the Valeo Relay Shield will be configured for our application.

To reset this configuration, simply run the program with an empty setup and loop, as they come by default.

Appendix C

How to install and configure a Local Server

To install a local server, it can be done using a VM or WSL (installation is done by executing the following command in PowerShell: **wsl –install**). Then, follow the tutorial from these links, one for VM and another in case of using Virtual Box:
- VM: Install Apache in WSL.
- Virtual Box:Install Apache in virtual box.

After that, run the following commands to update libraries and install PHP:

sudo apt-get update and **sudo apt-get php8.1**

Move the "version.php" file to the local server (www/var/html).

Appendix D

Detailed application structure

D.0.1 Valeo Relay Shield Detection Class Diagram

Now it proceeds to show the class diagram, related to the Valeo Relay Shield's detection process.

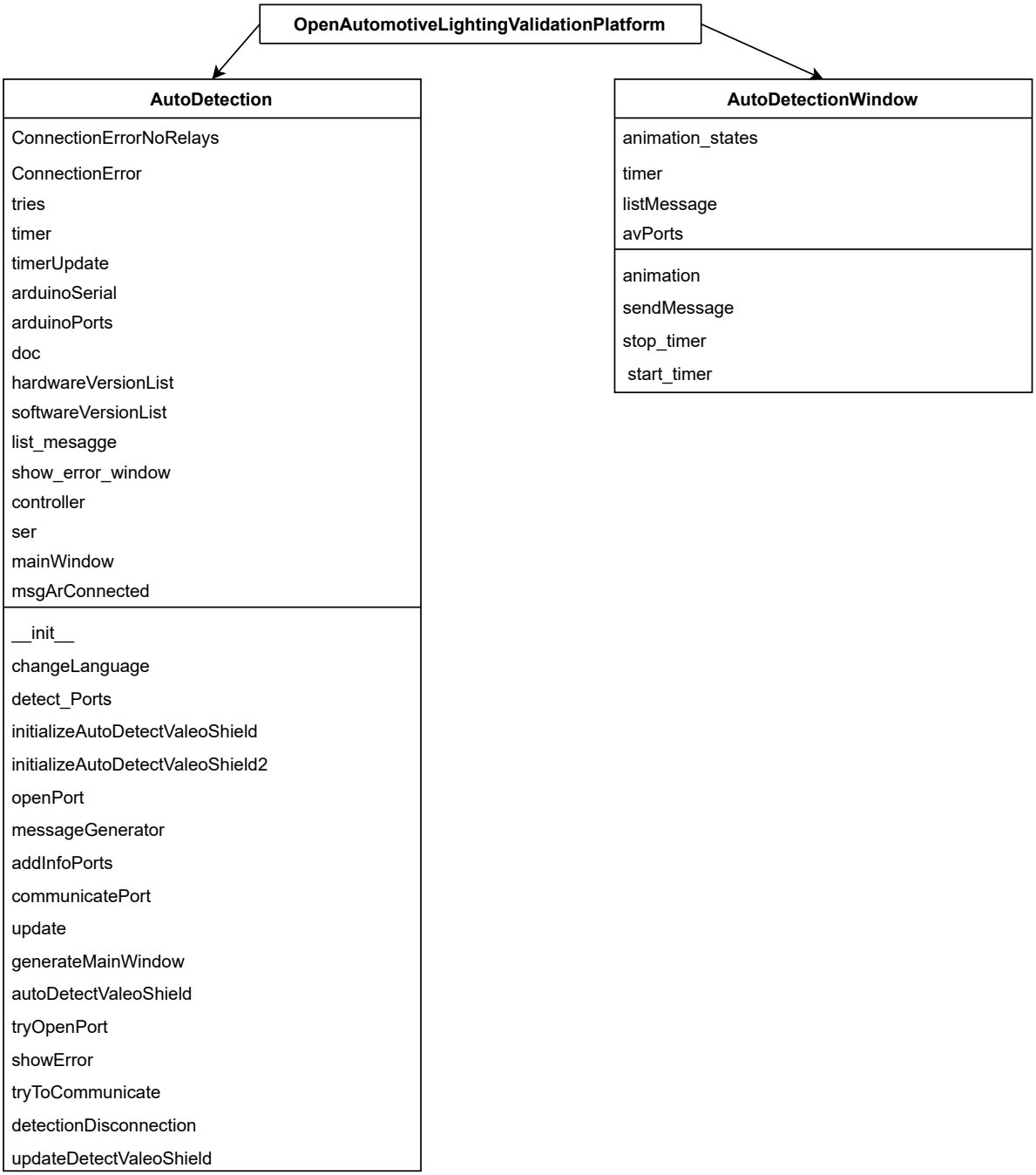

Figure D.1 – *Valeo Relay Shields' diagram class*

D.0.2 MainWindow Class Diagram

Next, it proceeds to show the mainWindow class which is used to access the various functionalities of the application.

Figure D.2 – *MainWindow diagram class*

D.0.3 Functionalities' Class Diagrams

The following diagrams show the different functionalities that the application has, starting with a window to contact the application developer and GranaSAT and the functionality to check for new software updates.

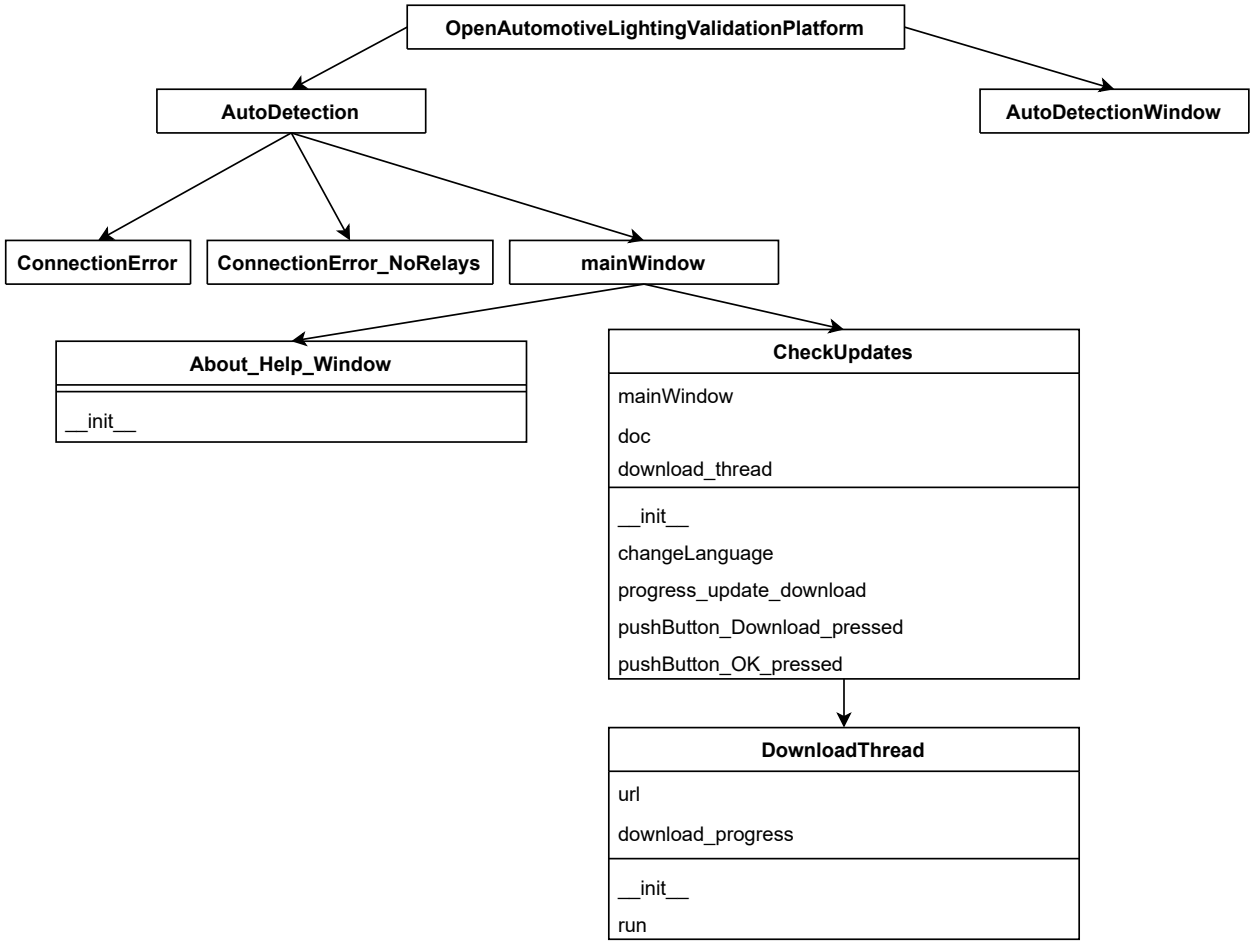

Figure D.3 – *CheckUpdate and AboutHelp diagram class*

The next diagram will show the openLightController functionality.

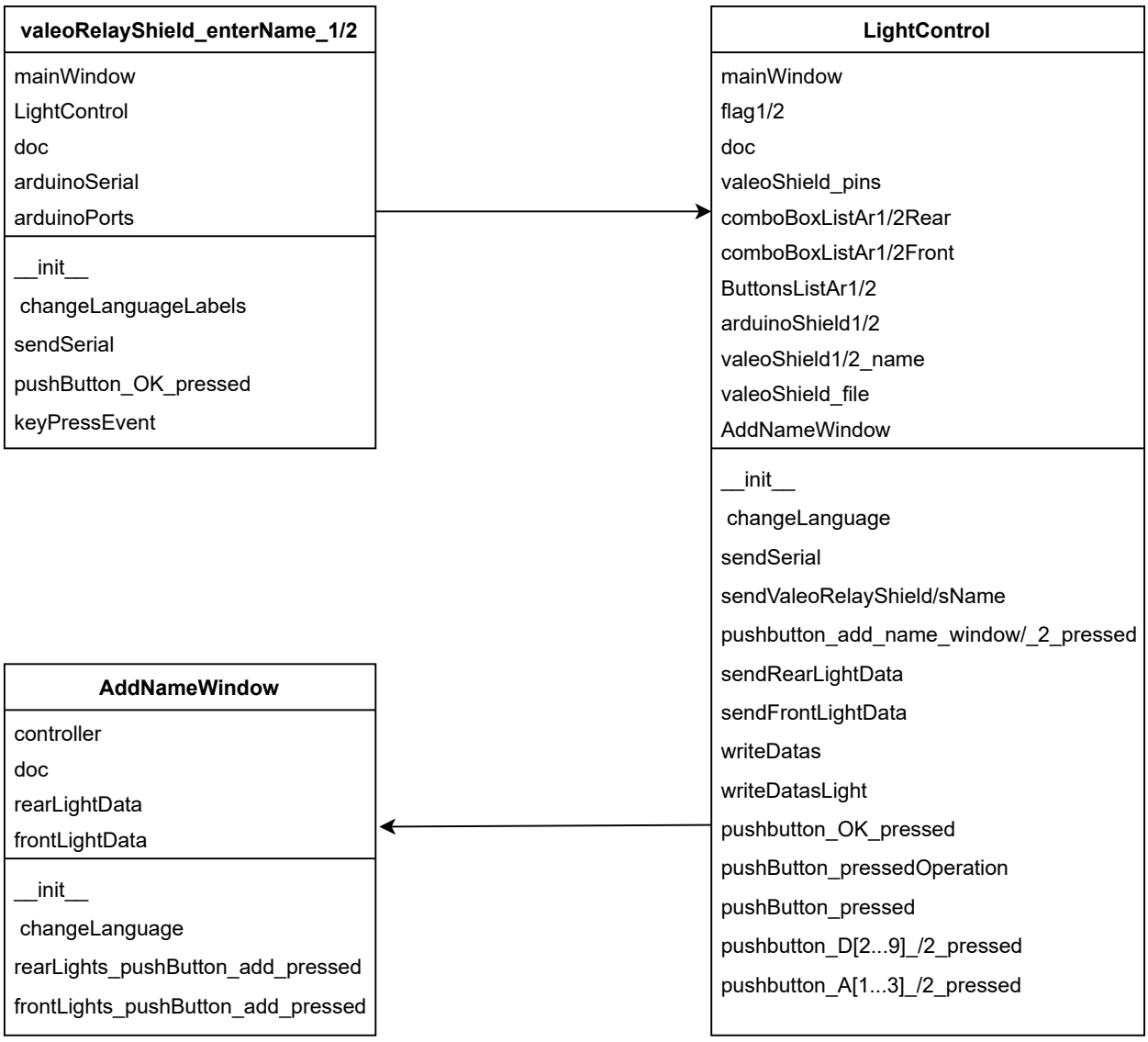

Figure D.4 – *Open light controller diagram class*

The last two main functionalities are divided into several diagrams. The first diagram (D.5) shows the classes that are in charge of displaying the forms to fill in the test fields and the VideoSourceWidget class is in charge of selecting whether a video or the camera will be used to perform the test (DUT or EMC).

The second diagram (D.6) shows the structure of the SelectRoi class, which is in charge of trimming the video to later create the ROIs in the DUTROI class (D.7) thus finalizing the process of performing a DUT test.

The last diagram(D.8) shows the EMClightAnalysis class that performs the EMC test and generates the report finalizing the process of performing an EMC test.

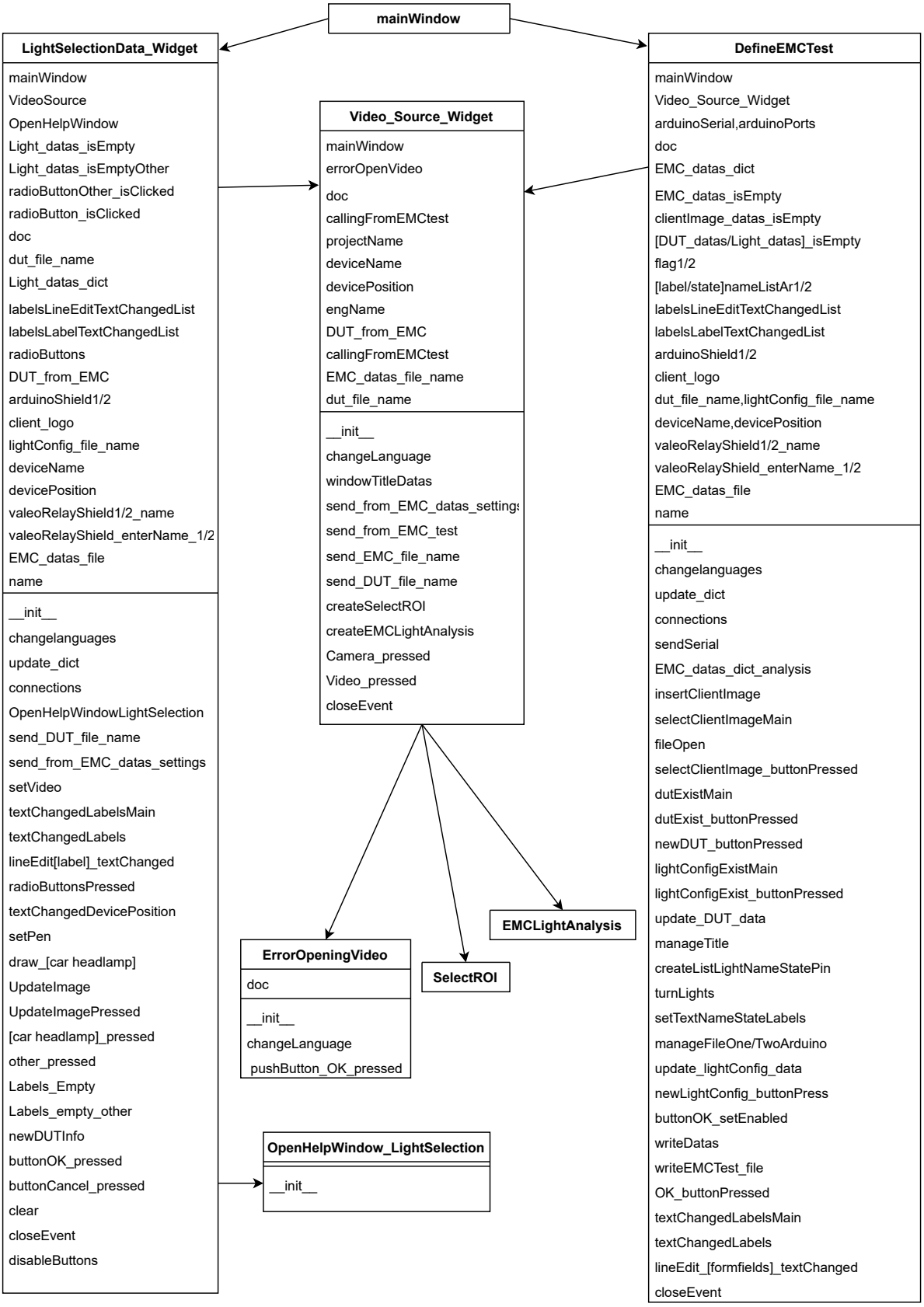

Figure D.5 – *Forms and Video source widget diagram class*

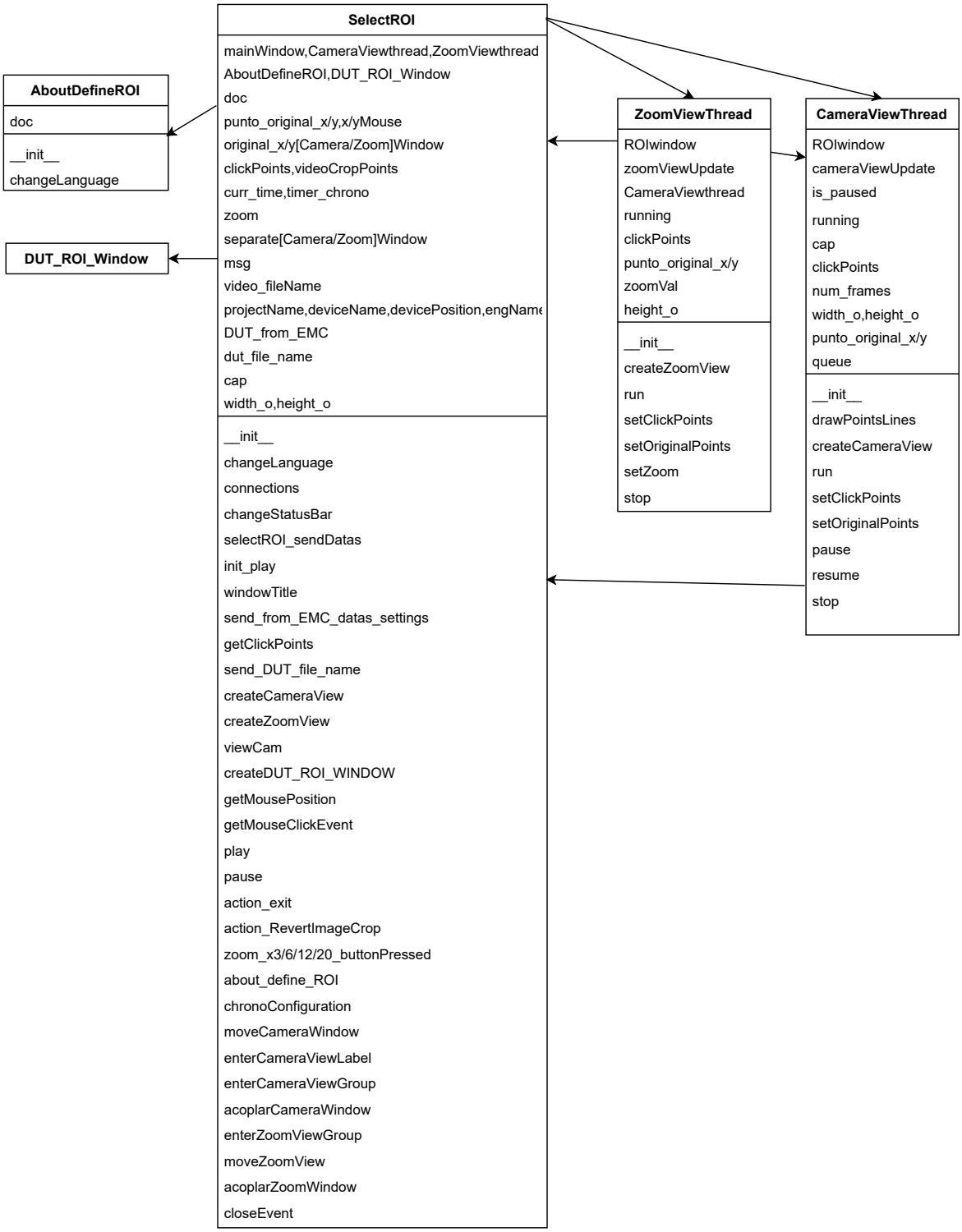

Figure D.6 – *Select ROI diagram class*

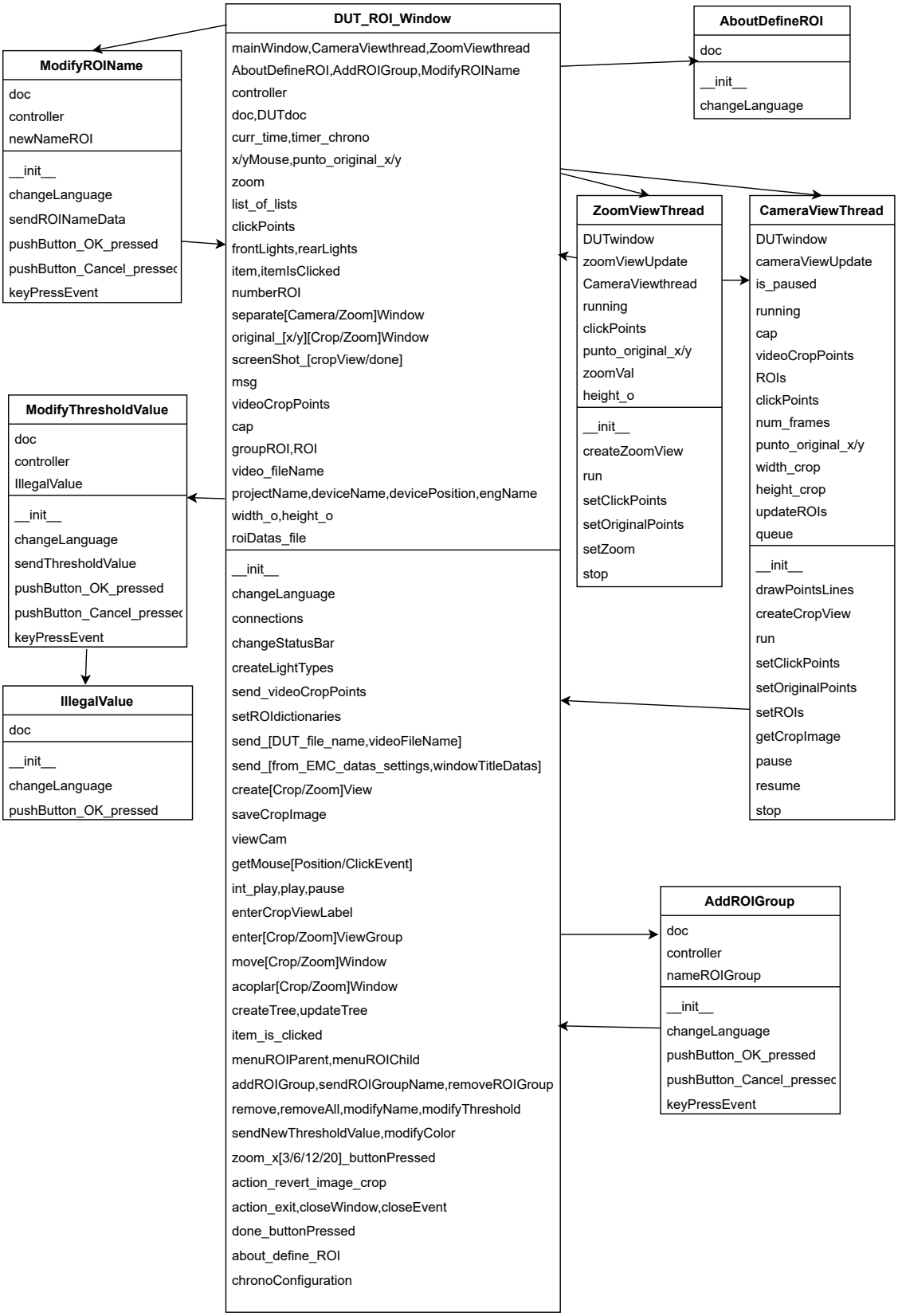

Figure D.7 – *DUT ROI diagram class*

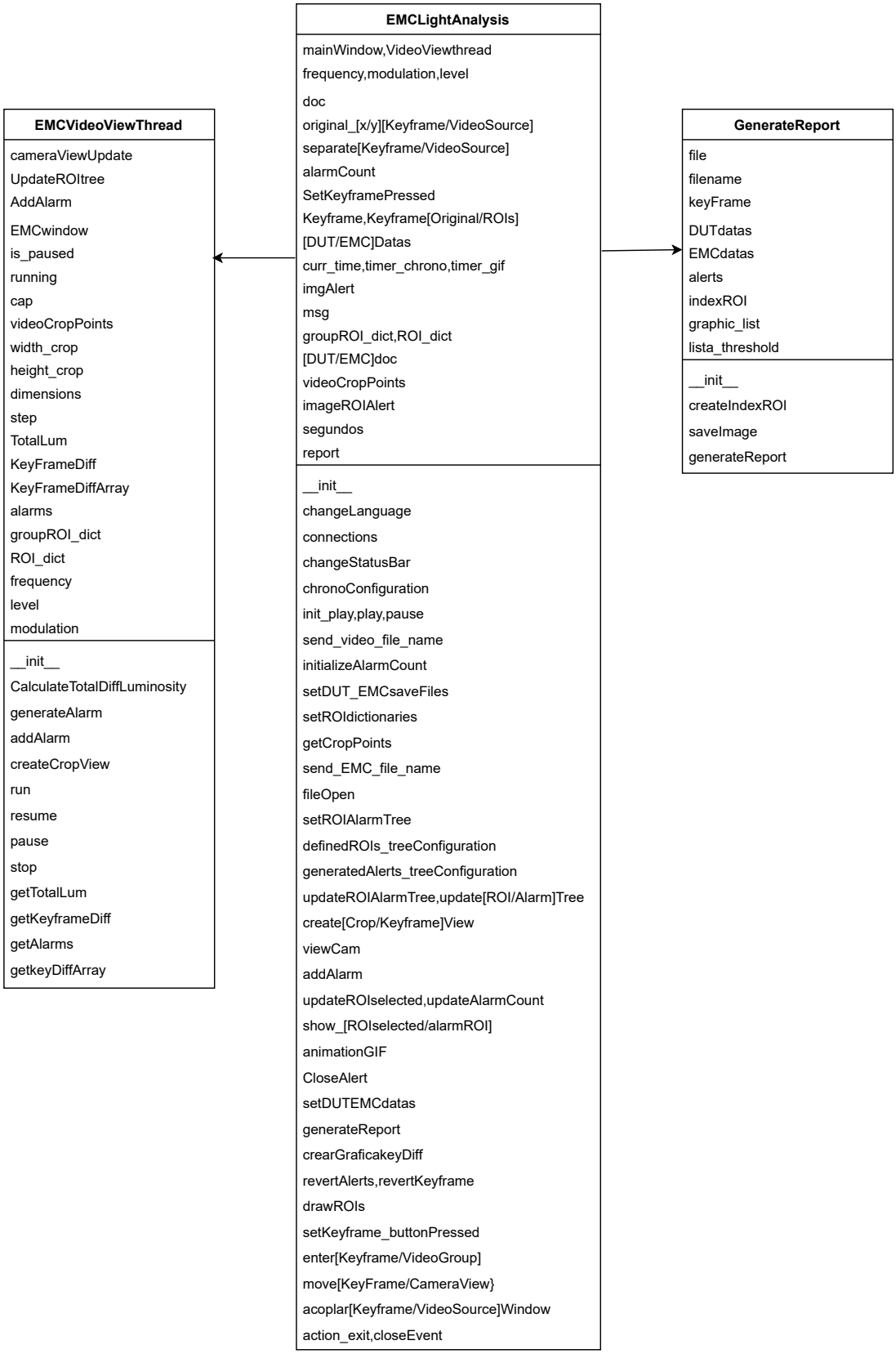

Figure D.8 – *EMCLightAnalysis diagram class*

Appendix E

Graphical visualization of upgrades

The classes which have been altered by this upgrades are shown in the following chart:

E.1 Interface Resposive

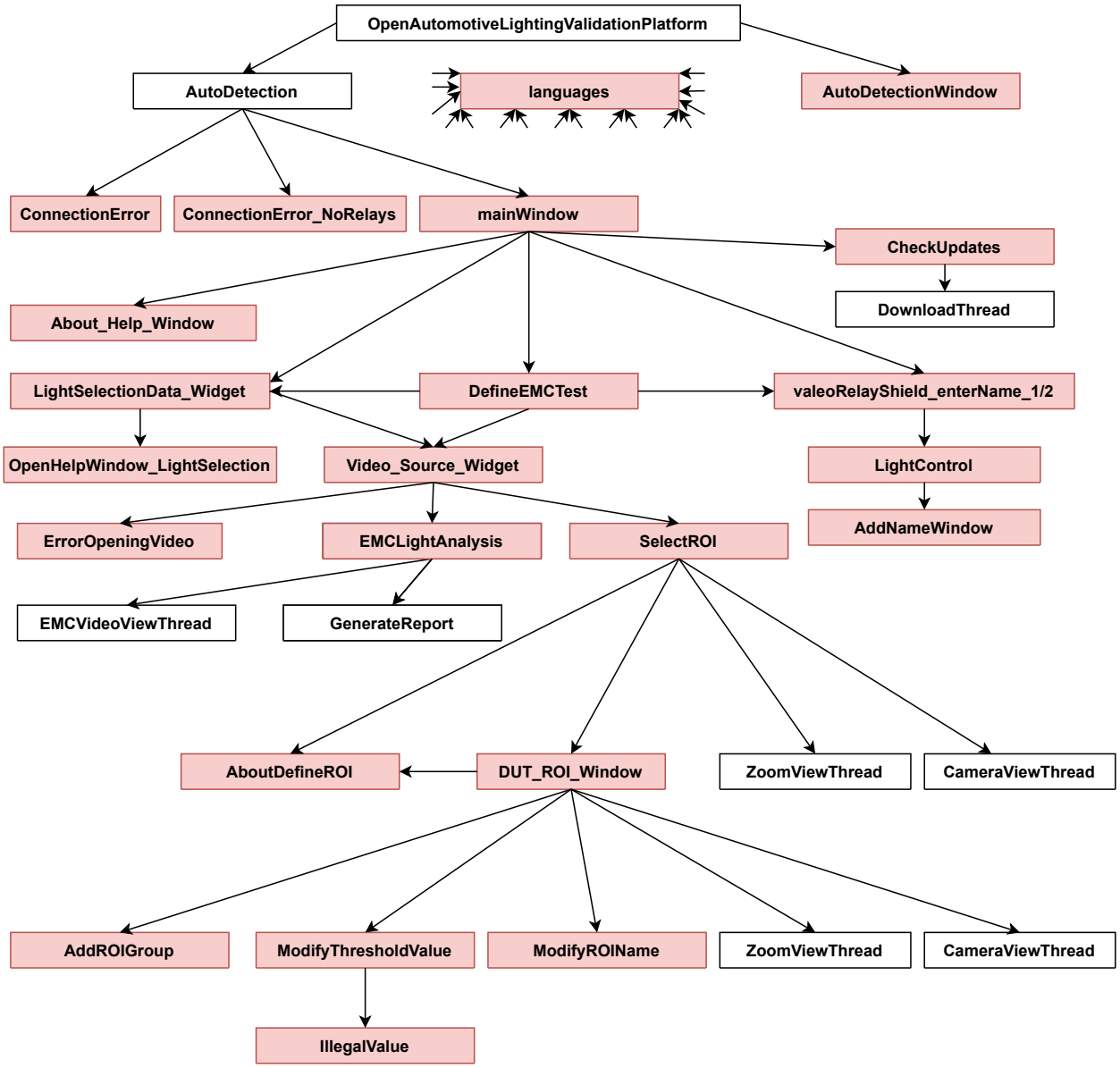

Figure E.1 – *Responsive Interfaces' Upgrade*

E

E.2 Valeo Relay Shield Detection's Upgrade

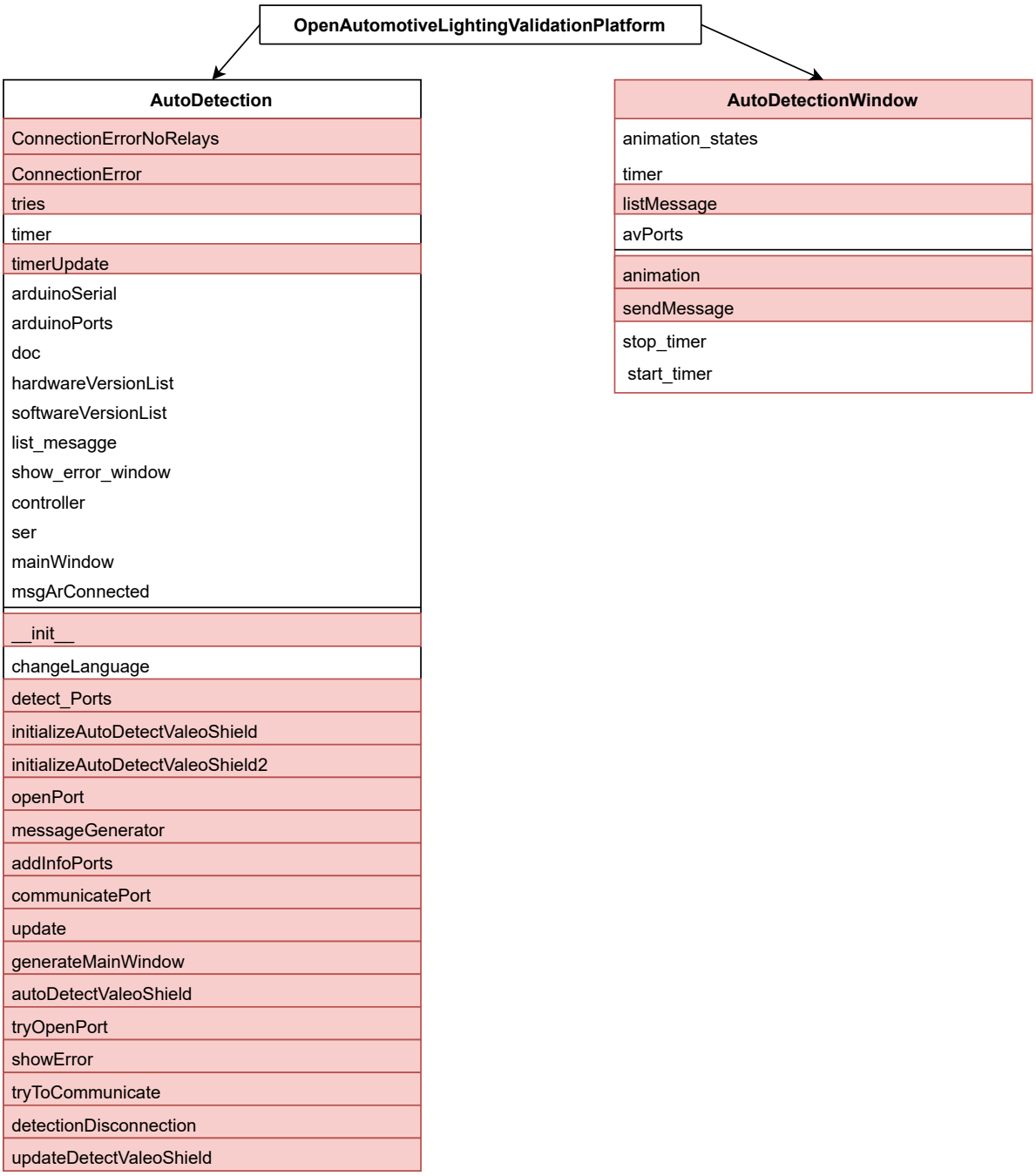

Figure E.2 – *Valeo Relay Shield Detection's upgrade*

E.3 Works without Valeo Relay Shields' upgrade

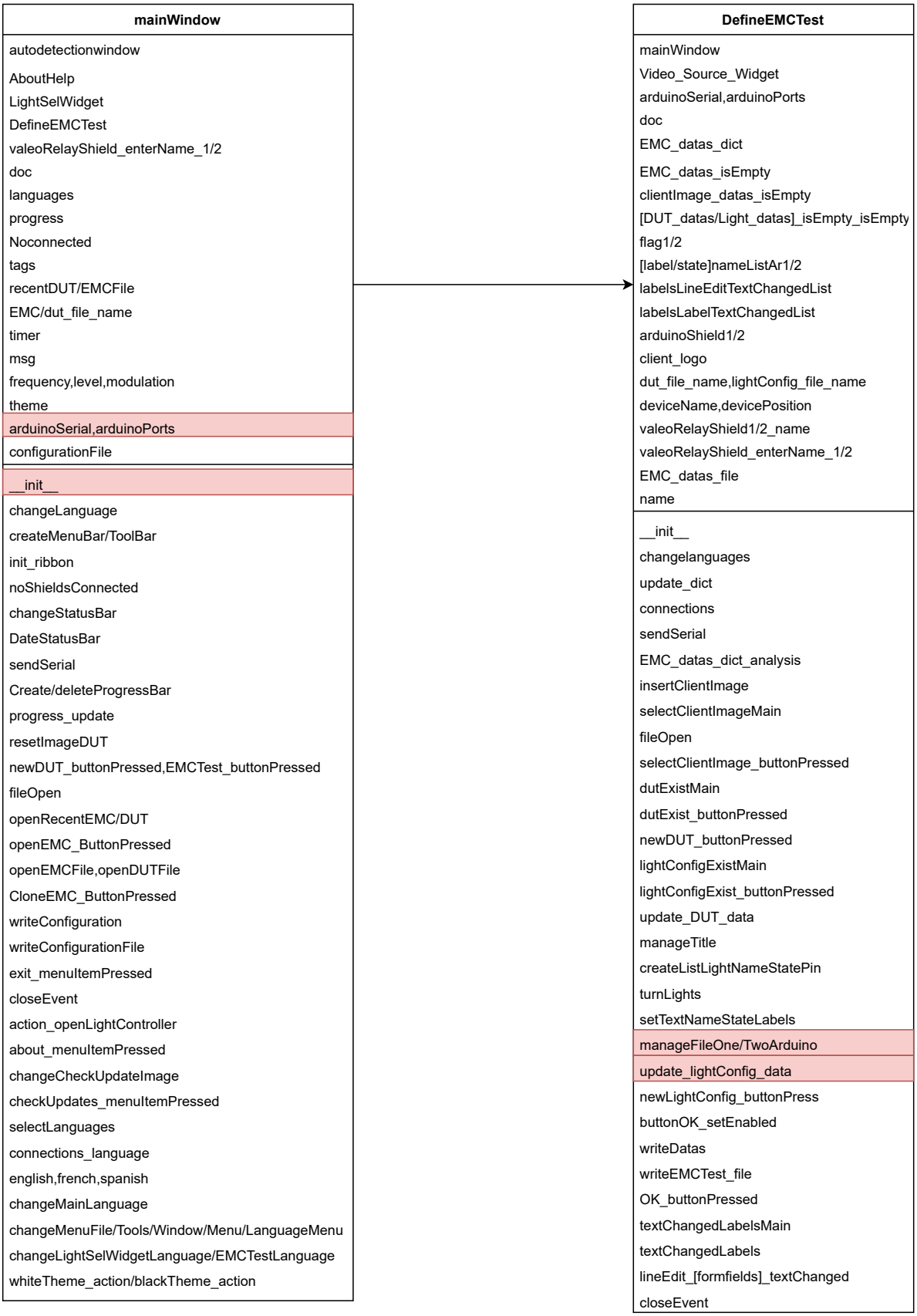

Figure E.3 – *Works without Valeo Relay Shield' upgrade*

E.4 Source code upgrade

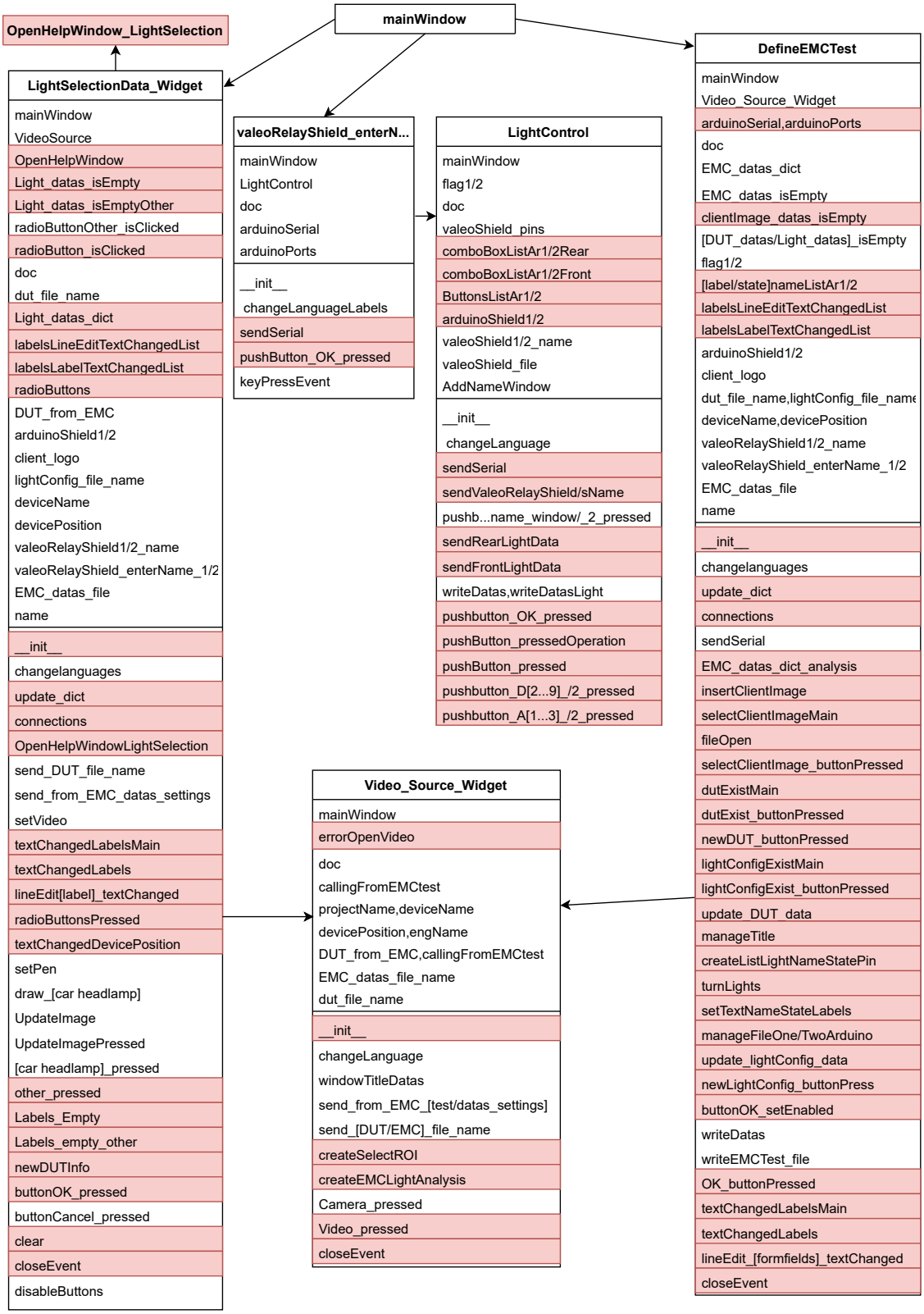

Figure E.4 – *Source Code's upgrade*

E.5 Open DUT 's upgrade

Figure E.5 – *Open DUT's upgrade*

E.6 Saved datas' upgrade

Figure E.6 – *Saved datas' upgrade*

E.7 DUT and EMC Tests' upgrade

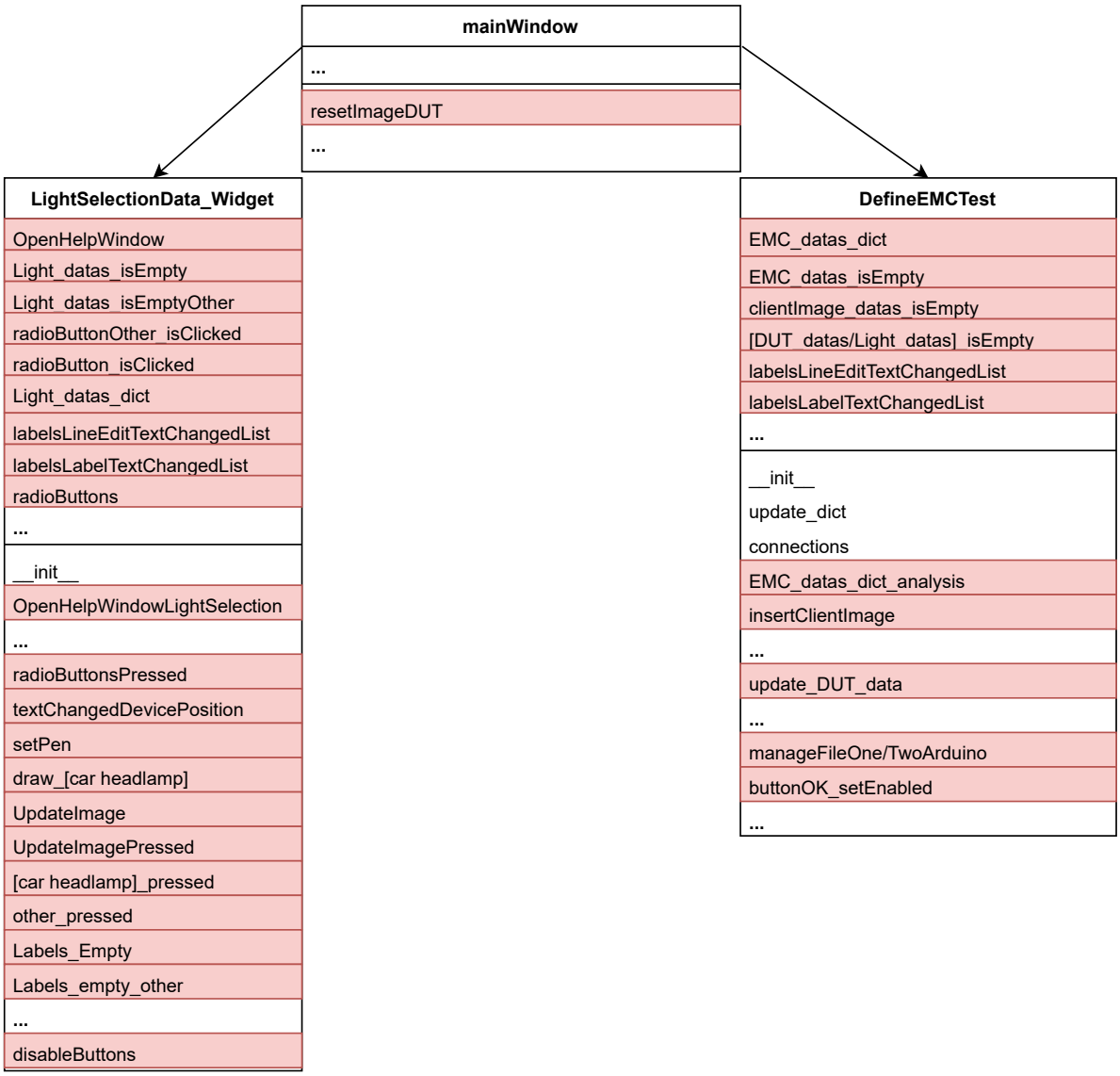

Figure E.7 – *DUT and EMC Tests' upgrade*

E.8 Languages' upgrade

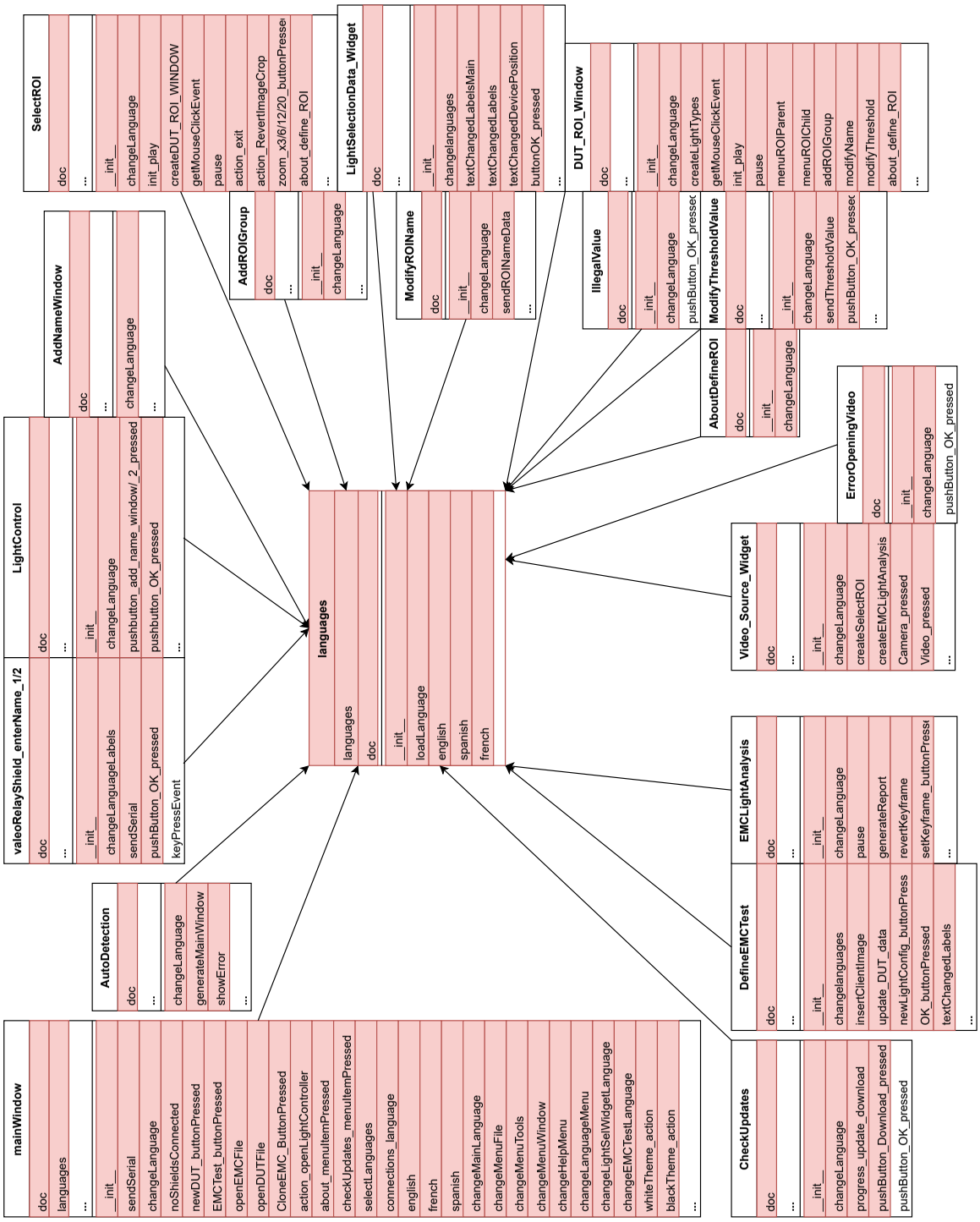

Figure E.8 – *Languages' upgrade*

E

E.9 Configuration file's upgrade

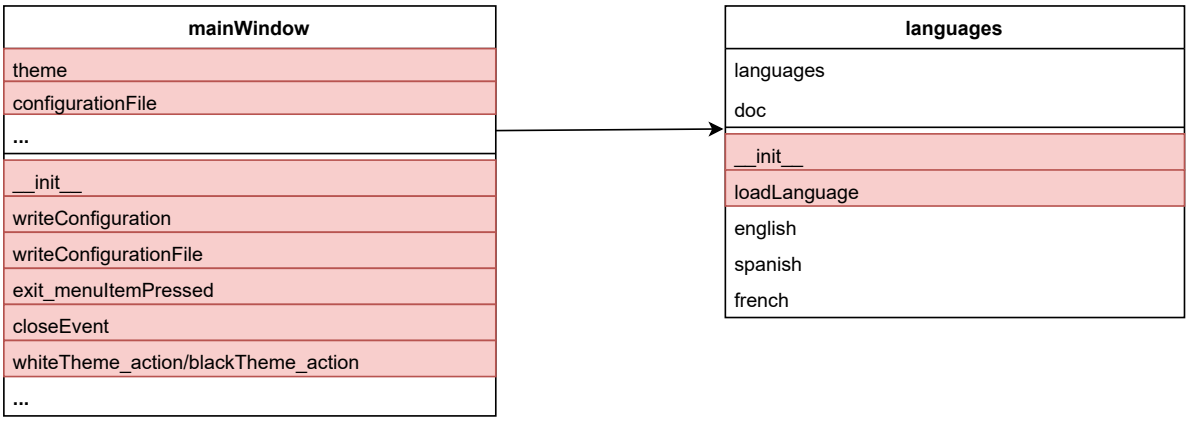

Figure E.9 – *Configuration file's upgrade*

E.10 Status Bar's upgrade

Figure E.10 – *Status Bar's upgrade*

E.11 Check Updates's upgrade

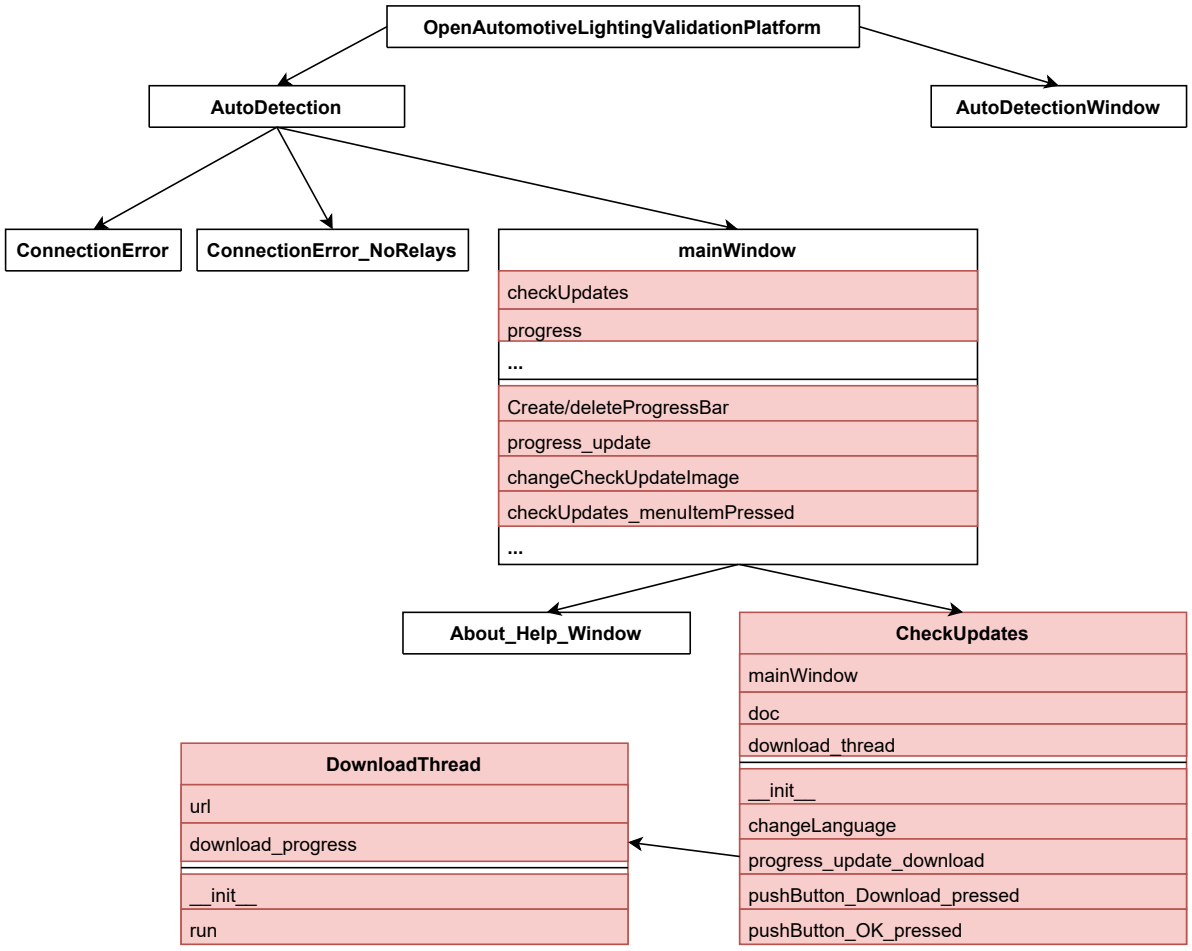

Figure E.11 – *Check Updates's upgrade*

E.12 Upgrades of the classes Select_ROI and DUT_ROI.

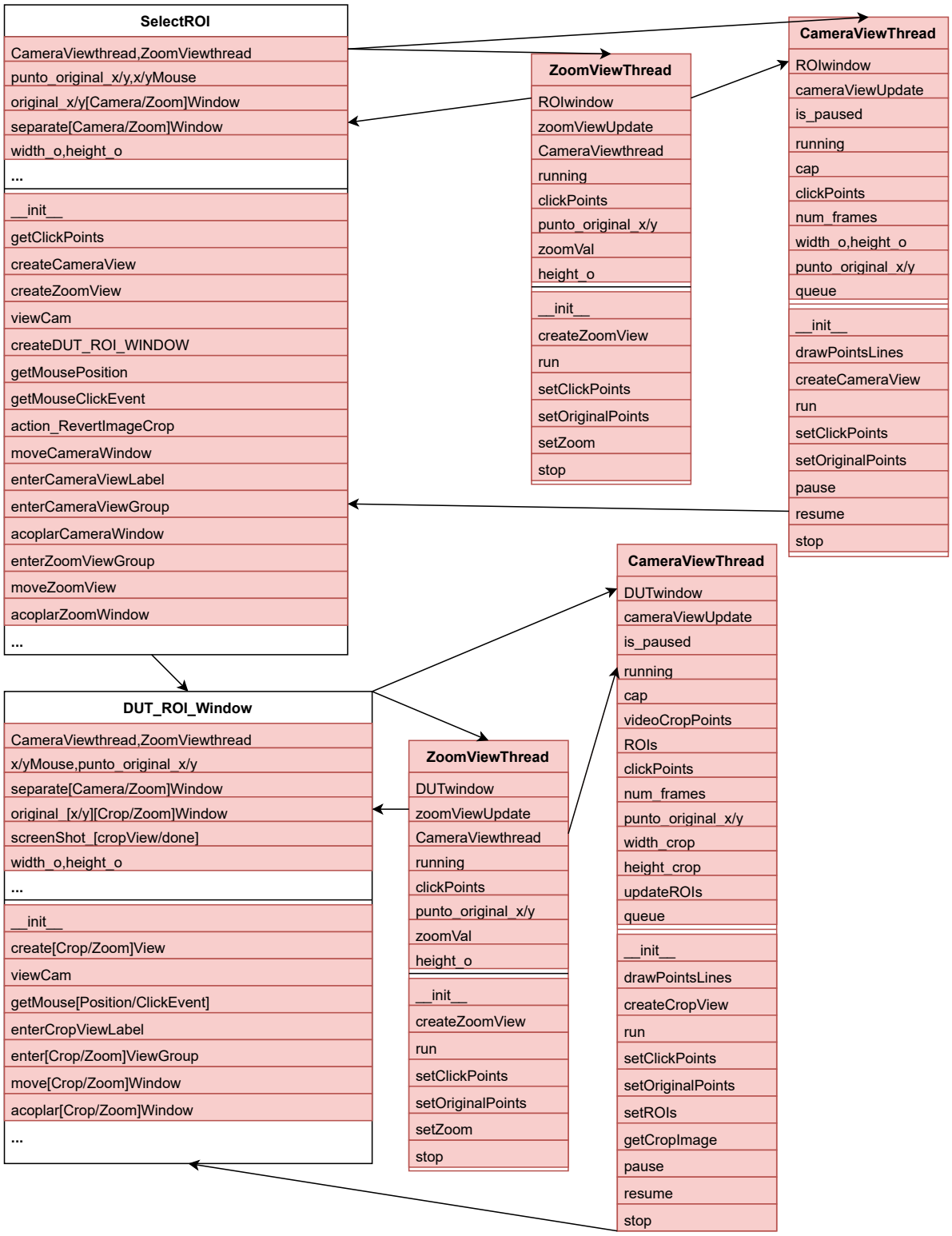

Figure E.12 – *Upgrades of the classes Select_ROI and DUT_ROI.*

E.13 Upgrades of the classes Select_ROI and DUT_ROI.

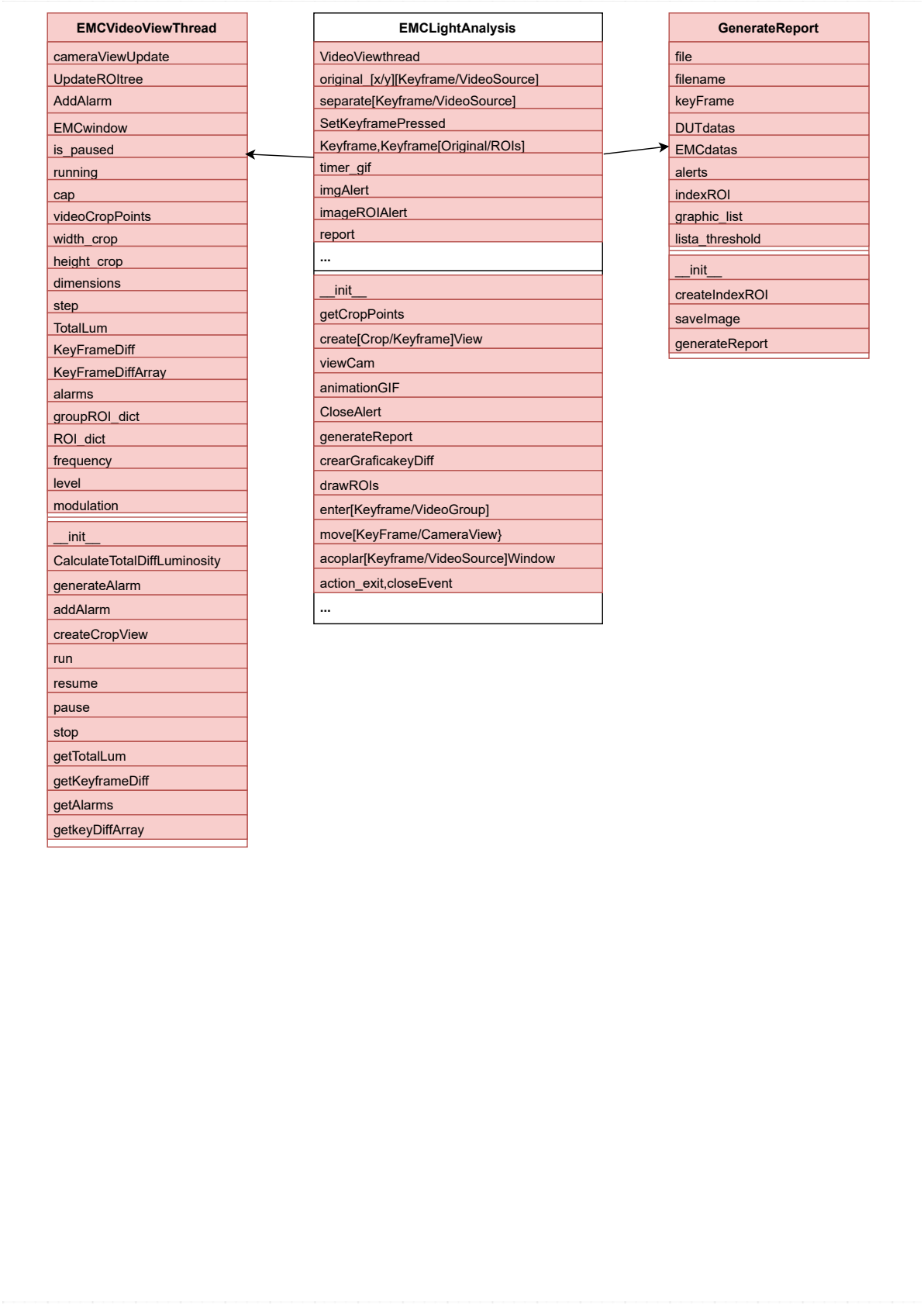

Figure E.13 – *Upgrades of the class EMCLightAnalysis*

E.14 Toolbar's upgrade

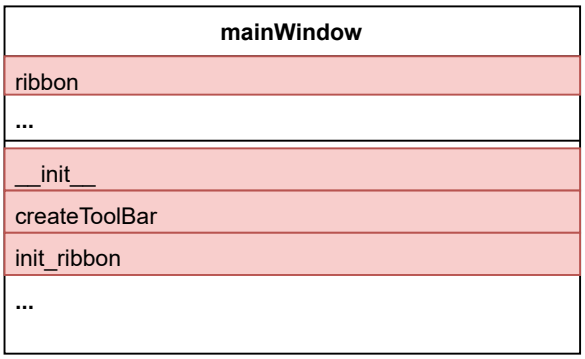

Figure E.14 – *Toolbar's upgrade*

Thank you for reading this Bachelor's Thesis.

Bibliography

[1] "Es de bien nacidos ser agradecidos." Refrán popular.

[2] GranaSat webpage. https://granasat.ugr.es Accessed: October 2023.

[3] "Pyarmor focus on protecting python scripts, by several irreversible obfuscation methods, now pyarmor make sure the obfuscated scripts can't be restored by any way." PyArmor.

[4] PyQt5 Documentation. https://doc.qt.io/qtforpython-5/.

[5] xml.dom.minidom Documentation. https://docs.python.org/es/3/library/xml.dom.minidom.html.

[6] Argos translate Documentation. https://github.com/argosopentech/argos-translate.

[7] Argos translate Documentation. https://pypi.org/project/argostranslate/1.4.0/.

[8] CX-Freeze documentation https://cx-freeze.readthedocs.io/en/stable/.

[9] PyInstaller documentation https://pyinstaller.org/en/stable/operating-mode.html.

[10] cxfreeze vs pyinstaller. https://coderslegacy.com/cx_freeze-vs-pyinstaller-comparison/.

[11] cxfreeze vs pyinstaller. https://www.slant.co/versus/9252/9256/~cx_freeze_vs_pyinstaller.

[12] Pyarmor documentation. https://pyarmor.readthedocs.io/en/stable/part-1.html.

[13] Pyarmor documentation. https://pyarmor.readthedocs.io/en/stable/questions.html?highlight=reverse#how-easy-is-to-recover-obfuscated-code.

[14] Pyarmor documentation. https://pyarmor.readthedocs.io/en/stable/topic/performance.html?highlight=obfuscation.

[15] Can EXE generated by cxfreeze be completely decompiled. https://stackoverflow.com/questions/5497399/can-exe-generated-by-cx-freeze-be-completely-decompiled-back-to-readable-python.

[16] Obfuscation process information. https://resources.infosecinstitute.com/topic/encrypted-code-reverse-engineering-bypassing-obfuscation/.

[17] How to install apache with WSL. https://www.how2shout.com/how-to/install-apache-on-windows-10-wsl-http-server.html.

[18] How to install apache in Ubuntu. https://ubunlog.com/servidor-web-apache-instalacion-conceptos-basicos-ubuntu-20-04/.