Bachelor Dissertation

Bachelor's degree in Industrial Technology Engineering

Test bench for mechanical characterization in reliability testing

PROJECT REPORT

Author:	Pedro Reyero Santiago
Director:	Héctor Alarcón Sánchez
Chairperson:	Josep Vilaplana Pastó
Call:	June 2018

ETSEIB

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

UPC

Summary

The objective of this project is to design an ‘automatic photography bench’ for reliability validations (mechanical stress tests, chemical tests, climatic tests…) of Telematic Control Units (TCUs) at the company Ficosa A.C., both at software and hardware levels. For each test sample, after each test, 6 photos must be taken from the 6 sides of a cube, in order to detect cracks and other defects produced as a result of these reliability tests. So far, this process was done manually, by laboratory technicians, which made the company waste money in each validation, so the process needed to be automatized.

In order to automatize this project, a device called “PhotoBench” has been designed and tested. It has six USB cameras, a cube-like structure to support them, with reflective panels and inner lighting, and connects to the laboratory technicians’ laptop or Raspberry Pi 3 through a USB HUB. An application with a fast and easy graphical user interface (GUI), programmed in Python and wxPython, lets the user control the PhotoBench comfortably.

Three design releases of the PhotoBench have been delivered with this project. The first one, designed experimentally by agreeing design parameters with laboratory technicians “by trial-and-error” and physically built as a prototype; the second one, designed through optics calculations as a proposed improvement of the first release without changing the already built prototype structure; the third and last one, redesigned completely through optics calculations as a proposed improvement to optimize the mechanical structure and image quality for a given high-megapixel USB camera.

This way, the project’s objective has been successfully fulfilled, with all three releases being directly usable in real reliability validations at Ficosa A.C. The application and the GUI let the user comfortably control the PhotoBench and remarkably reduce validation process times, showing to be a profitable investment for the company. Second and third releases were not fully physically implemented, but they are already being manufactured, or planned to do so in the near future. Other business units, who produce other products like mirrors or electric car components at Ficosa, have also shown interest in using this project’s result, as they have the same inefficiency problem with reliability validations.

Table of Contents

ETSEIB

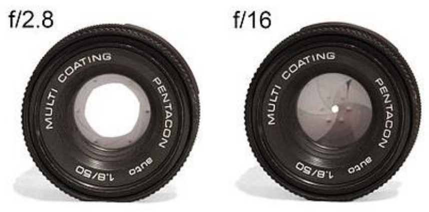
ETSEIB

1. Glossary

This chapter includes brief explanations about key concepts presented during this project's report that may need clarification for some general readers.

- **Angle of view (AOV):** Angular extent of a given scene that is imaged by a camera. It is used interchangeably with the term 'Field of view'.

- **Aperture (optics):** Magnitude that measures the dimension of the light beams that, coming from an object, penetrate an optical instrument. It is represented by the "f-number", which is a dimensionless number equal to the ratio of the system's focal length to the diameter of the entrance pupil.



f/2.8
f/16
MULTI COATING PENTACON auto 1.8/50
MULTI COATING PENTACON auto 1.8/50

Figure 1.1 A large (f/2.8) and a small (f/16) aperture [Source: Wikipedia]

- **Discounted Cash Flow (DCF):** Valuation method used to estimate the attractiveness of an investment opportunity. It uses future free cash flow projections and discounts them, using a required annual rate, to arrive at present value estimates.

- **ECU (Electronic Control Unit):** Any embedded system in automotive electronics that controls one or more of the electrical systems or subsystems in a vehicle.

- **Field of view (FOV):** Area under inspection that a camera can acquire.

- **Focal length (optics):** Measure of how strongly an optical system converges or diverges light. For an optical system in air, it is the distance over which initially collimated (parallel) rays are brought to a focus.

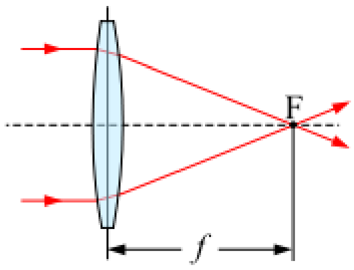


Figure 1.2 Focal point (F) and focal length (f) of a positive (convex) lens [Source: Wikipedia]

- **GUI (Graphical User Interface):** A type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators, instead of text-based user interfaces, typed command labels or text navigation.

- **HMI (Human-Machine Interface):** Set of parts of an interactive system (hardware or software) that provide the necessary information and control for the user to do some task with the interactive system. When the HMI is computer-assisted, it is a part of the program that communicates with the user. Ultimately, it is the point of action where a human stablishes contact with a machine.

- **HW (Hardware):** Interconnected electronic components which perform analog or logic operations on received and locally stored information to produce as output or store resulting new information or to provide control for output actuator mechanisms.

- **Image resolution:** Detail an image holds (higher resolution, means more image detail). The term resolution is often considered equivalent to pixel count in digital imaging.

- **Image sensor format:** Shape and size of an image sensor. Examples of it are ¼ inch, ⅓ inch, $1/2.5$ inch, etc.

- **IMEI (International Mobile Equipment Identity):** A number, usually unique, to identify mobile phones and some satellite phones.

- **IRR (Internal Rate of Return):** Rate of return that sets the NPV of all cash flows (both positive and negative) from the investment equal to zero.

- **MEC (Mechanical structure):** Mechanical structure for the PhotoBench, which includes supporting profiles, lighting, panels, etc.

- **NPV (Net Present Value):** In finance, a measurement of profit calculated by subtracting the present values (PV) of cash outflows (including initial cost) from the present values of cash inflows over a period of time.

- **Payback period:** In capital budgeting, refers to the period of time required to recoup the funds expended in an investment, or to reach the point at which total cost and total revenue are equal.

- **PB (PhotoBench):** Device designed in this project (at HW and SW levels) that automatizes most operations regarding TCU test samples' photo taking, processing and saving in Reliability validations.

- **Product lifetime / lifespan:** Time interval from when a product is sold to when it is discarded.

- **Reliability engineering:** A sub-discipline of systems engineering that is concerned with the ability of a system or component to perform its required functions under stated conditions for a specified time

- **SW (Software):** Refers to a collection of data or computer instructions that tell the computer how to work, in contrast to the physical hardware from which the system is built, which actually performs the work.

- **TCU (Telematic Control Unit):** In the automobile industry, refers to the embedded system on board a vehicle that controls tracking of the vehicle. It consists of a global positioning system (GPS), an external interface for mobile communication, an electronic processing unit, a microcontroller, a mobile communication unit, and some amount of memory for saving data.

ETSEIB

2. Preface

This chapter briefly presents the historical circumstances and market situation that motivated the project.

2.1. Project's origin

For some years, Ficosa's Advanced Communications business unit (Ficosa A.C.) has been producing Telematic Control Units (TCUs) for controlling telecommunications in cars, as well as other kinds of Electronic Control Units (ECUs). After a new TCU is designed, it has to be validated, so that the company can ensure it will work properly under in-vehicle conditions for the whole car's lifetime. As it would not be profitable to have a new product tested for 10 years / 150.000 km (habitual "target lifetime" agreed between Ficosa A.C. and its clients) before launching it onto the market, some special tests called 'Reliability' tests are performed in order to force TCUs' ageing and simulate their deterioration due to real in-vehicle conditions, by subjecting them to severe conditions, like operation at really high/low temperatures, high humidity, mechanical impacts, mechanical vibration, high electrical loads, corrosive atmospheres, etc.

After each of this tests is performed, photos need to be taken of the tested samples, in order to detect by comparison any kind of damage produced by each test to the TCUs, and include these photos in the final test reports. Up to now, in Ficosa A.C., these photos were taken manually by some engineer or laboratory technician, by placing the TCU on a table by hand, taking a photo with a mobile phone or a digital camera, and repeating the process for each of its 6 sides. This was repeated for each TCU being validated, and then photos were taken out from the device's memory, renamed, resized, organized into folders, and uploaded to the company's servers. As typical validations involve considerable amounts of TCUs and tests, this process was by no means practical. It implied having qualified workers, with high revenues, doing a repetitive process that generated no value added, which made the process strongly inefficient and a source of money waste. Apart from that, this manual and repetitive process usually led to mistakes, being really easy to forget to take a photo from 1 side of a TCU, to get a blurry or misaligned photo, to mix up photos from different samples, etc.

When a dedicated reliability team was built for TCU validation at Ficosa A.C., one of the team members, Héctor Alarcón, who had been part of previous reliability validations for several products and was aware of this inefficiency in the company's process, decided to propose the automation of the sample's photo taking as an improvement project, consisting on the design of a machine/bench (and its control program) that would automatically and robustly do as many previously manual tasks as possible, thus saving the company lots of money and giving a more professional image to clients.

2.2. Motivation

After analysing the existing market, both Ficosa A.C. and the student concluded that no suitable solution was available that could meet the company’s needs, so the full development of a new product was required. This resulted in a challenging and engaging project for the student, as he would need to use knowledge from many different engineering branches studied during his bachelor degree, as well as take plenty of design decisions, taking full responsibility for each and every of them, and manage the whole project himself.

3. Introduction

This chapter states this project's problem to solve, objectives, scope and design requirements, to serve as a starting point and basis for the rest of its development.

3.1. Project's problem

During reliability validations for TCUs in Ficosa A.C., laboratory technicians and engineers manually take a great amount of photos from lots of test samples after performing each test. This kind of approach is a source of time inefficiency and errors, which ultimately implies a waste of money for the company. In order to solve this problem, company's engineers proposed to design a 'photography bench' (PB) that would make this process as automatic, fast and robust as possible, and assigned this improvement project to Pedro Reyero, industrial engineering student working at the company as reliability intern. The student would perform the whole project himself, while working at the company, as part of his workday, inside of the company's facilities and using the company's resources whenever needed, in order to complete the project in accordance with agreed time and design requirements.

3.2. Project's objective

The objective of this project is to design an 'automatic photography bench' for reliability validations of TCUs in Ficosa A.C. The technician who uses the bench shall be able to take photos of each TCU test sample by introducing it in the bench and controlling the process through some graphical interface. This process must be easy, fast and robust (that is, it shall avoid any source of possible errors from the user, and it shall assure constant results' quality). In the end, the bench must reduce validations' process times.

3.3. Project's scope

This project's scope was fixed by the company Ficosa A.C. itself, as their engineer proposed the improvement project. This project is aimed to design the photography bench at two levels: software (SW) and hardware (HW).

At the end of this project, its main output must be a design of a system capable of guiding the entire TCU photo taking process for reliability validations. On the one hand, this design will specify a certain hardware capable of recognizing the test sample's information (recorded on text and 2D code on the sample's label), taking the different photos from the sample, editing them properly to fit into test reports' format needs, and saving them in an organized folder structure. On the other hand, this design will include a fully-working code for an application to control the abovementioned hardware, put together in the form of a graphical interface that will give the technician the right functionalities and information to use it without having to know anything about how it works internally, and that will help the technician to move through the process fast, easily and without mistakes.

This design does not need to include a well-defined mechanical structure (MEC) for its final implementation for validations, but some kind of physical prototype that includes a mechanical structure to support it must be built in order to validate the final design's correct operation (both at SW and HW levels) and to check that it meets all design requirements. If the hardware requires certain mechanical parameters (for instance, some certain distances) to fulfil a certain condition in order for the it to function properly, these mechanical requirements must be part of the project's design, and so are inside of the project's scope.

3.4. Design requirements

For this project, as it is meant for internal use, design requirements come directly and solely from requirements/specifications agreed with the company. Throughout several meetings with reliability engineers and laboratory technicians, several requirements were detected and were finally summarised and concretely defined in the form of User Stories [1], a technique used in the project management methodology called Agile [2]. User stories are short, simple descriptions of a feature, told from the perspective of the person who desires the new capability (usually a user or costumer of the system). They usually follow a simple template:

As a < type of user >, I want < some goal > so that < some reason >.

After the abovementioned meetings, project's tutor, company department's supervisor and student agreed and wrote an initial set of user stories, which was defined as open-to-change during project execution if some new need were to be detected or some need were to be considered as no longer applicable. Table 3.1 shows the final set of user stories that defines the design requirements for this project, ordered according to implementation order.

#	As a	, I want	so that	Main field(s)
1	**test operator**	to have a user-friendly HMI (including workflow)	I can control the PB.	SW
2	**test operator**	to take automatic photos of each TCU side	I can mechanically characterize the TCU.	SW, HW, MEC
3	**test operator**	to have the photos in a folder with the IMEI as folder name	I can trace photos to their corresponding TCU.	SW
4	**test operator**	to read the label's Data Matrix	I can have TCU traceability.	HW, SW
5	**test operator**	photos with high and reduced (72 dpi) quality	I can reduce the size of test reports.	SW
6	**test operator**	to be able to export photos taken from the PhotoBench	I can easily upload them to the company's server.	SW
7	**test operator**	to be able to use the application in different devices	I can use my own device to operate the app (laptop or Raspberry Pi 3).	SW, HW
8	**reliability engineer**	to have execution logs	I can trace test operator activities and debug errors.	SW
9	**test operator**	to have a fixed support to put test samples	they are always in the same position.	MEC
10	**test operator**	to power the PB connecting it to 230V AC line	I can use the PB in all laboratories.	HW, MEC
11	**company**	the PB to have a professional design/look	I can show it to clients if needed.	MEC
12	**test operator**	the PB to detect if photos fulfil similarity to example photos	I can detect when something unusual has happened when taking the photo.	SW
13	**department**	a MEC prototype of the PB	I can validate the operation of SW and HW together in real conditions.	MEC
14	**test operator**	the PB to take photos with a constant quality and size	I can directly use all the photos in test reports.	SW, MEC
15	**department**	a complete final prototype (SW, HW, MEC) of the PB	I can validate all design requirements once the project is completed.	SW, HW, MEC
16	**department**	the PB to take high quality photos	I can detect more clearly cracks and other defects produced during tests.	HW

Table 3.1 User stories that define project's design requirements

3.5. Project's solution

To fulfil the company's needs for the PhotoBench, expressed as design requirements, and solve the project's problem, the student proposed the following solution: a device with 6 cameras (one per photo, in order to have the possibility of independent configurations for each of the photos), one in the middle of each side of a cube-like metallic structure that will support the cameras. Cameras will be controlled by an application that acts as a Human-Machine Interface (HMI), which will be as simple, fast, user-friendly and robust as possible, in order for the laboratory technician to reduce process times and not make mistakes.

The control application, written in Python, will ask the user for test sample's information and project, by menu and by reading the label's DataMatrix with a USB barcode scanner, will execute the photo taking and saving process with one simple click, and will give its user the possibility of checking, exporting and clearing the obtained results.

The metallic structure, of which only a prototype will be made for this project, will have aluminium profiles to support the whole bench, and matt-white foamed PVC panels to close the interior of the bench, so that all light coming from a set of LED strips is as diffuse as possible and no external light noise influences the photos. The test samples will be placed on a transparent methacrylate support (which is transparent in order to be able to take the top and bottom photos), which will have adhesive marks on it to indicate where to place the TCU. The LED strips will be powered by a power supply, which will need an external power connection.

The cameras, as well as the USB barcode scanner, will be connected to a powered USB HUB, which will also need an external power connection. The USB HUB will be connected to the user's laptop or Raspberry Pi 3 by one USB type-A cable. Cameras' positions will be recognised by the application through the use of a "calibration cube", once each time the application is launched.

4. Project approach

This chapter states the different methodologies and strategies adopted in order to develop the whole project, as well as its overall planning and final approach.

4.1. Methodology

This project's development is organised in a way inspired by a project management methodology called Agile [2]. Agile is an approach to software or, in general, project development under which requirements and solutions evolve through the collaboration of self-organising and cross-functional teams and their end users.

One of Agile's main characteristics is the definition of a set of tasks, usually based on user stories, with an estimated amount of workload for each of the tasks (usually measured in hours). The project is then tackled sequentially in small time periods, called sprints. For each sprint, the available manpower is calculated (usually measured in hours), and a certain subset of tasks is selected from the remaining set of tasks, according to the tasks' estimated corresponding workload. During the sprint, each team member takes a task from the available sprint tasks list and works on it. Once he/she finishes the task, he can check the sprint remaining tasks list again and tackle another task. Priorities can be stablished and team members' dedication can be switched from one task to another according to current sprint needs and priorities. Thanks to this approach, the whole team is responsible for the fulfilling of all tasks, and any team member can directly see the current state of the sprint execution and task priorities at any moment.

With Agile as the project's main management inspiration, this project's development is tackled in a series of sprints, each one with a defined start and due date, which implies a defined available manpower (in hours). Project's task list comes mainly from user stories defined in Table 3.1, and Jira [3], an issue-tracking and project management software used in Ficosa A.C., tracks each of these project tasks, providing project members with time tracking capabilities, comments, descriptions, worklogs, etc. Each task has a clear task description with a 'Definition of Done' (DoD), that is, a list of the conditions to meet in order for the task to be considered as done/completed, as well as an estimated required workload (in hours) to fulfil it. A full report of each Jira task can be found in Appendix A.

4.2. Planning and scheduling

This project was initially agreed to start the 12th of February of 2018 and to end the 25th of

May of 2018, as this was the period of time for which the student had an internship agreement to work at Ficosa A.C.

In order to define project's different sprints, before each sprint the student held a meeting with the project tutor and the department's supervisor, in which they decided which user stories were going to be tackled next, in what order, and with what priority. This selection was made according to the current state of the project, always trying to define short sprints with a certain "core idea" or "main focus" (e.g.: user interface implementation, mechanical prototype design, etc.), making them similar to project "stages". They agreed the sprint's start and due dates, and planned a meeting at the very end of the sprint. The student then took the selected user stories and converted them into defined tasks, writing for each of them a clear Definition of Done and estimating their workload (taking into account that he had a maximum of 4h available per working day).

During Sprint 6, when the project was almost complete, the internship agreement was renewed, and a last user story was agreed and added (sixteenth design requirement in Table 3.1, which aimed for results' quality improvement), extending the project's deadline to the 18th of June of 2018.

4.3. Approach overview

The project was fulfilled in 8 sprints, plus an initial period of basic prototype development and design requirements definition. The first two sprints focused on software aspects, implementing the graphical interface for test operators. The next two sprints focused on defining and manufacturing a mechanical structure (MEC) prototype to validate SW and HW together, and defining the final camera to be used in the bench. The fifth sprint focused on solving an interaction problem between SW (control application) and HW (cameras) that appeared. The sixth sprint focused on delivering a complete prototype (SW, HW and MEC) of the PhotoBench that fulfilled all initially agreed requirements. The seventh sprint focused on improving the results obtained by the PhotoBench's full prototype, by performing adjustments to its HW (without changing anything from the MEC). The eighth and final sprint focused on improving the results obtained by the PhotoBench's full prototype, by redesigning the MEC to optimise conditions for a certain HW (using the acquired know-how of the project's problem and solution), and documenting the whole project. Figure 4.1 shows a simplified GANTT chart of the whole project, representing with colours each sprint's main focus/field (cyan for SW, magenta for HW, yellow for MEC, and their combinations for sprints combining different main fields). In Appendix A (Section 10), a detailed GANNT chart of the whole project is presented.

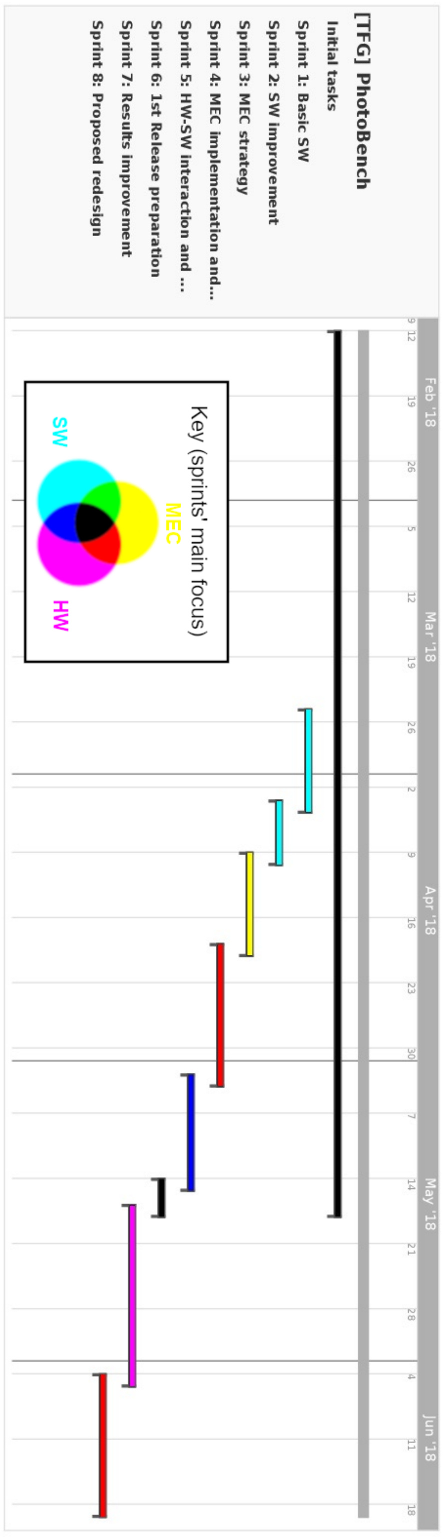


Figure 4.1 Simplified GANNT chart of the project. Key indicates field-colour equivalences

5. Project development

This chapter details the whole project's development, divided into an initial prototype and the project's consecutive sprints. For each sprint, an initial planning is included with its Jira tasks (ordered by priority and logical implementation order), start date, due date and time estimations. Then, the sprint's approach is briefly presented, with a description of the sprint's focus and an overview of the concrete tasks / actions for that sprint. Next, concrete aspects from the sprint's tasks are discussed, in order to describe the project's solution in detail. Finally, sprint's results are summarised before presenting the next sprint.

5.1. Initial prototype

Initial prototype's Jira task (Start date: 12/02/2018 , Due date (extended): 19/03/2018)	**Estimated time**	**Spent time**
Create initial prototype	14h	56h

Table 5.1 Jira task for the initial prototype

5.1.1. Task approach

Before defining the final product's requirements and developing the design, project's tutor suggested to implement a basic initial prototype of a photography bench, in order to detect possible sources of problems or difficulties and to get an idea of what was technically feasible and what not, so that later on the definition of the design requirements would be easier and would result in clearer user stories.

The initial prototype, as defined by project tutor's requirements, would be a system that could read the label's 2D code (with the info from the test sample), take 6 photos from a TCU with 1 camera (one photo at a time), and display the results of the process on a screen, with a basic control program to command it.

In order to tackle this task, first a functional analysis of the desired system was needed, then an alternatives analysis for each of the functions detected in the previous analysis, and finally the choice and implementation of a final solution.

5.1.2. Prototype's functional analysis

A functional analysis [4] consists in the division of a system into smaller parts, each of which performs a certain 'function'. This function defines the transformation from certain inputs into certain outputs, without specifying how this transformation is done. This analysis allows to split the problem of finding a global solution into finding solutions to smaller problems, without specifying any solution to the problems yet. Figure 5.1 shows the functional diagram for the initial prototype, in which the electric energy input as material resource for the system has been omitted to ease the diagram’s interpretation.

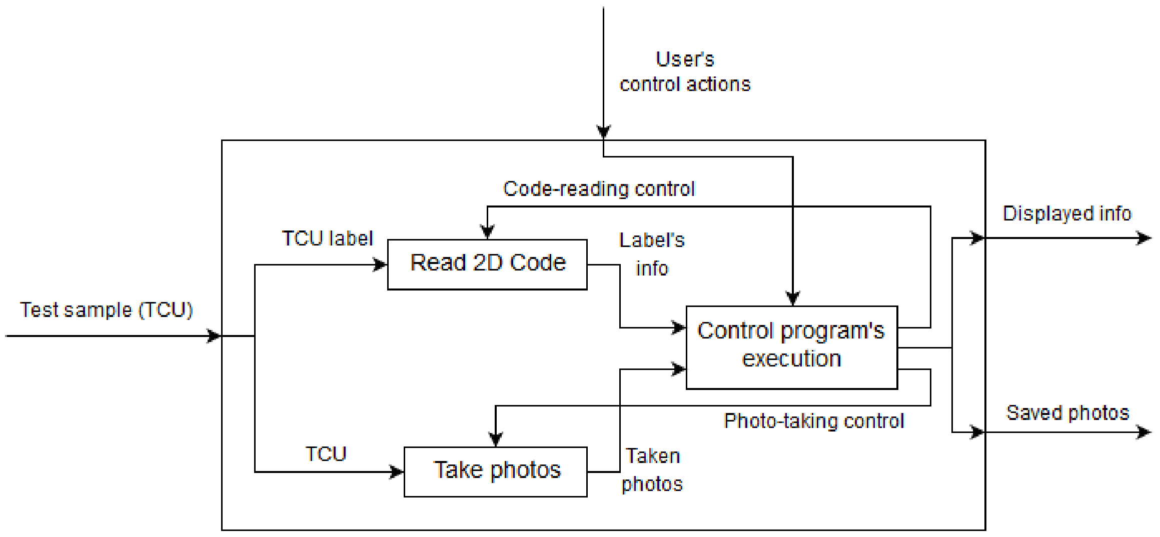


Figure 5.1 Initial prototype's functional diagram

5.1.3. Prototype’s alternatives analysis

After performing the functional analysis, the student did some research about possible alternatives that could work as solutions for each of the detected functions. For each function, alternatives were proposed, compared and, when possible, tested experimentally.

Before starting analysing each function, a couple of basic decisions were made regarding the initial prototype. Python2 was chosen as the programming language in which the control program’s code would be written, as the student had previous in-depth knowledge of the language, Ficosa A.C. also used Python2 for most of its TCU validation tests, and Python is a great language to code simple and clear applications. All the final control hardware was decided to be contained in a box, so that only inputs and outputs of the prototype were visible to the user (“black-box approach”).

First, an alternative was investigated for taking the photos. To begin with, a VGA Arduino Camera was tested, as it was a small and really cheap (around 1-5€) model, but the camera showed serious difficulties for its control using any Arduino board, as it lacked an electronic buffer circuit (FIFO), and its final image quality was very limited. Then, the Raspberry Pi V1 5MP camera was tested, which had an easy Python control in Raspberry Pi boards, and had a decent quality for photos from around a 10-15 cm distance. Problem with this camera was that it had no integrated focus adjustment, so it was unable to take good quality photos from close distances (around 1-2 cm, to capture the 2D code from the label), or from larger distances (around 30cm, which was the distance from which photos had been taken manually in previous reliability validations), apart from the fact that it could only be controlled by a Raspberry Pi (it had a special CSI serial cable). Finally, USB webcams were investigated, as they offered multiplatform support and were easy to control from Python. The requirement of having a manually adjustable focus was imposed, as that made the prototype versatile and let the student experiment with both close (for 2D code reading) and far (for test sample photos) distances. The webcam model Papalook PA187 was chosen, as it claimed to take “12MP FullHD 1080p quality photos” from 30 cm and had an adjustable focus (using an integrated focusing wheel). When tried experimentally, this model gave great close-up results, getting clear images from the 2D code, and also gave decent quality at further distances (around 30 cm), so it became the selected camera for the initial prototype. Further investigation on cameras stopped at this point, as, for the time being, the camera was only needed to experiment, and it was becoming a blocking issue, with lots of available hours spent in looking for and testing cameras (other Arduino modules, webcams…).

Next, an alternative was investigated for controlling the program execution. First, an available Arduino board was tested, as it was a cheap option for the prototype, but it lacked Operating System, internal memory and video output ports, as well as any CSI camera or USB ports to control the cameras, so it was not a very satisfactory control hardware for this application. Then, an available Raspberry Pi 3B board was tested, as it had its own Operating System (Raspbian, a dedicated distribution of Debian), internal memory, HDMI video output, USB ports and a CSI camera port, while being much cheaper than a desktop computer or a laptop. Those capabilities, together with Debian’s native compatibility and preference for Python, made the Raspberry Pi 3B the final choice to make an embedded initial prototype.

Finally, an alternative was investigated for reading the 2D code. Every TCU test sample at Ficosa A.C. has a label with some information that distinguishes the test sample from the rest, and part of this information is also stored in a 4-stacked DataMatrix [5] (4 industrial 2D codes stacked in a 2x2 matrix style, as shown in Figure 5.2) next to the written information in the label. Ficosa A.C. agrees with the client, prior to each project’s start, the information contained by the label and the DataMatrix, as well as its format and display arrangement. To read this label, each validation test bench (a setup with the devices needed to validate the TCU) includes a USB barcode scanner. Before running into using the scanner directly, the student investigated the technical feasibility of reading the DataMatrix through a photo taken with a camera, “similar to what a barcode scanner does”. Several Python open-source libraries were tried, such as pylibdmtx [6], ZBar [7], and ExactImage [8], as well as others in other programming languages (which could have been adapted into Python code), such as DmDecoder [9] and Freitag DataMatrix Decoder [10]. None of them proved to be useful for this project, as they were either too slow (taking more than 1 minute to analyse 1 clear photo with only the code), not robust enough (decoding succeeded or failed depending on slight variations on photo conditions) or could not decode DataMatrix codes stacked in 4 (some libraries could only read non-stacked DataMatrix codes). If one of these libraries were to be used, it would require the test operator to place the TCU near a close-up camera for quite a long time, and this placement would need to have high precision, as close-up photo taking tends to have a small field of vision in order to capture smaller details with higher definition, so all in all it would go against the PhotoBench's objective, which is process time reduction. For this reason, the initial prototype included a USB barcode scanner, which could read the DataMatrix fast and robustly with a small cost increase (50-100€ for a usable model).

Figure 5.2 4-stacked DataMatrix example [Source: Wikipedia]

5.1.4. Prototype's solution

With a chosen solution for each of the detected functions, the only thing left was to implement the control program and put together all the hardware. The control program was written in Python, and consisted on a simple application for the system console (in this case, for Linux's console, as it was implemented in a Raspberry Pi 3B, which uses Raspbian) that first asks the user to read the DataMatrix with the scanner, then displays a window with a picture with photo examples in the expected order and a window that shows what the camera sees at each moment, then takes a photo (in native camera resolution and FullHD) every time the user presses a key and then, when the 6 photos have been taken, saves all the photos (in a folder named after the test sample's IMEI and located at the same folder level as the python program file) and closes automatically in 10 seconds. Figure 5.3 shows a diagram of the initial prototype's hardware and information flux, with the prototype seen as a black box with only its inputs and outputs visible to the user, and Figure 5.4 shows a photo of the initial prototype taken the day a demo was performed for department colleagues and laboratory technicians. Chosen hardware's specifications are included in Appendix B (Section 1), while full control program's code can be found in Appendix C (Section 1).

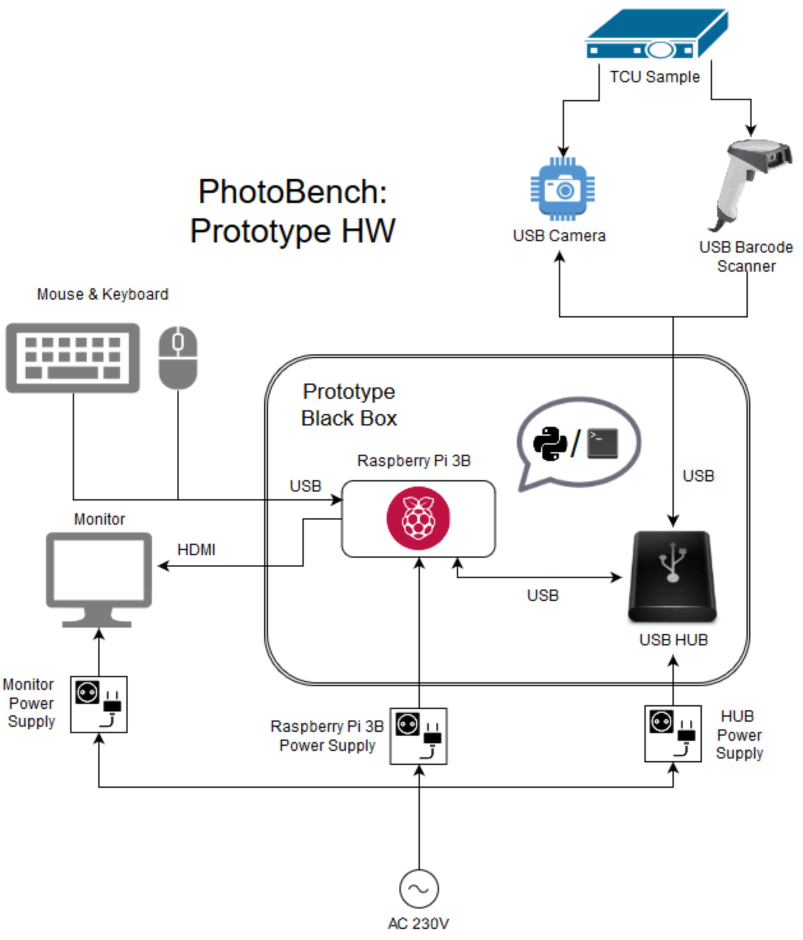


Figure 5.3 Initial prototype's hardware diagram

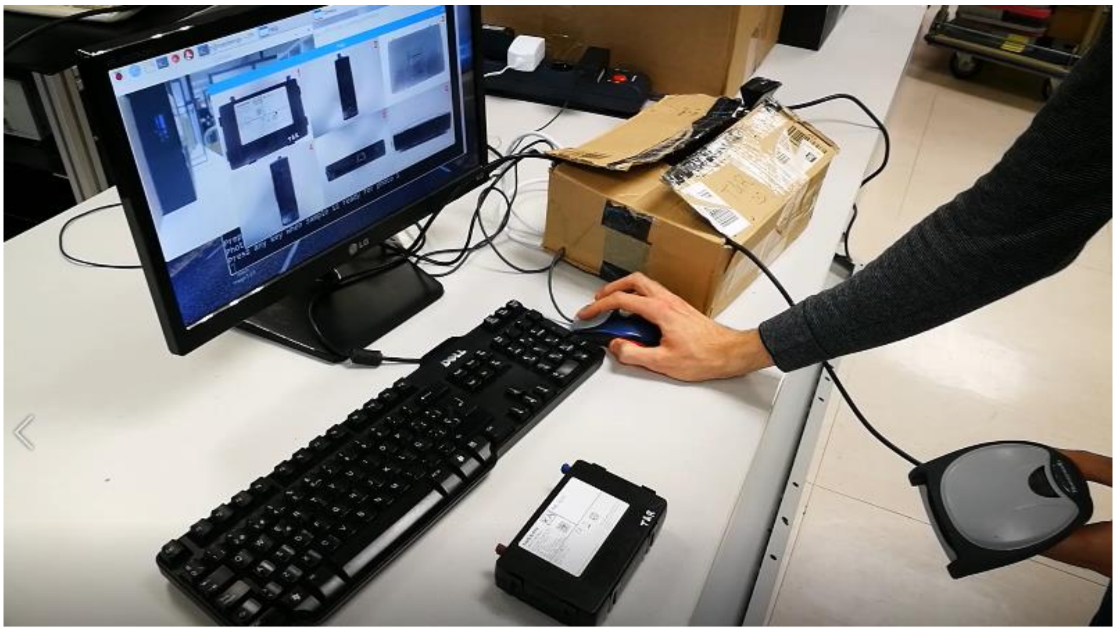


Figure 5.4 Initial prototype's actual implementation

5.2. Sprint 1: Basic SW

Jira tasks for the sprint (Start date: 23/03/2018 , Due date: 03/04/2018)	**Estimated time**	**Spent time**
Implement basic Human-Machine Interface (HMI)	9h	11h
Implement multi-camera photo taking	2h	3h
Implement automatic photo saving in folder	3h	1,5h
Implement DataMatrix information input with USB barcode scanner	3h	2,5h
Implement photo resolution adjustment	3h	2h
Implement basic photo exporting	5h	3h

Table 5.2 Jira tasks for Sprint 1

5.2.1. Sprint approach

The first sprint of the project focused on software development. Its main objective was to have a basic human-machine interface (HMI), which would be improved later on, adding extra functionalities to the core structure defined here.

The sprint's tasks consisted in researching on open-source Python libraries for graphical interfaces, implementing a basic HMI with displays and buttons, programming Python modules to tackle various functionalities (taking 6 photos with one or various camera, saving photos in a folder with a certain pixel density, reading DataMatrix information with a barcode scanner, and compressing photos from samples into a .zip file), and finally integrating those modules into the basic HMI. Modules are Python files that contain definitions of functions, while the HMI script is a Python file that imports the written modules and uses them to execute a set of orders written in it (whenever its main function is called), which conform the program's execution.

5.2.2. Language and framework choice

In order to implement the graphical interface for the user, a programming language must be chosen. For this project, this choice was rather simple, as Ficosa A.C. stated its preference for Python2 as the programming language for its new projects, as it had already spent lots of hours and economical effort into training its engineers on the language and adapting its testing environment to open-source Python2 libraries. The student also had a solid foundation on Python, acquired during his bachelor degree studies (specially on Python3, but differences were insignificant for this project's needs), so all in all Python2 was clearly a convenient choice.

After choosing the programming language, in order to avoid wasting lots of hours "reinventing the wheel", an open-source Python framework dedicated to the creation of graphical user interfaces (GUIs) was used. Through some research on the topic, the two most convenient options seemed to be PyQt and wxPython, both of which had already been used in Ficosa A.C. too. Taking into account implementation simplicity, cross-platform support and program O.S.-native appearance, wxPython was finally chosen for this project.

5.2.3. wxPython

wxPython is an open-source cross-platform GUI toolkit for Python, implemented as a set of Python extension modules that act as a wrapper of the popular C++ cross-platform library wxWidgets [11]. It allows for the creation of robust and highly functional GUIs that will run on different platforms (Microsoft Windows, Mac OS X, Linux…), using O.S.-native widgets in most cases to give applications a 100% native look.

The basic components of a wxPython GUI are an application object, instance of the *wx.App* class, which contains an event handler (which checks for user interactions with the GUI, like clicking on a GUI button, entering text from a keyboard, etc.), a window object, instance of the *wx.Frame* class, which represents the GUI's window, and a panel object, instance of the *wx.Panel* class, which acts as a container for any other objects in the GUI (buttons, text displays, drop-down lists, etc.). The application's event handler is run by a call to the application object's *MainLoop* method. The following code would be an example of it:

```
app = wx.App(False)
frame = wx.Frame(None,
title = "Photo Bench HMI",
size = (600,500))
panel = MainPanel(frame)
frame.Show()
app.MainLoop()
```

5.2.4. Sprint results

At the end of the sprint, a basic HMI was successfully implemented. It consisted on a main .py script for the HMI core, and the following modules:

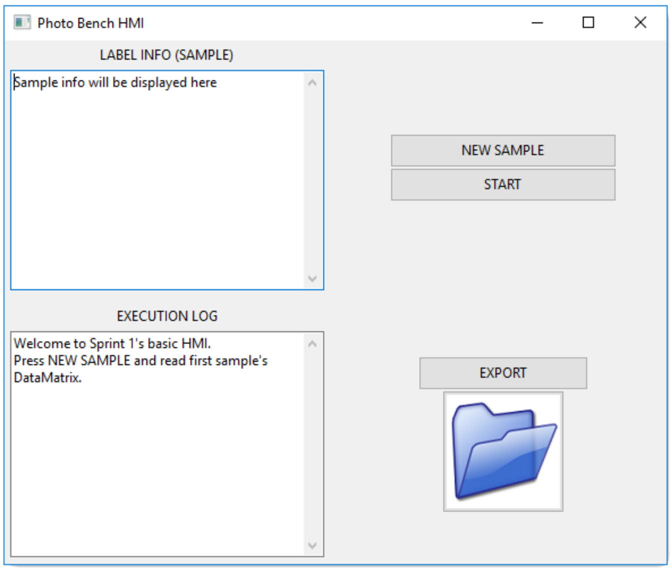
- **labelinput.py:** Allows to read (with a barcode scanner), interpret and store a test sample label's info.
- **multicamera.py:** Allows to take *n* photos with either *n* cameras, opened and closed sequentially (as the Raspberry Pi 3B can't open at the same time 6 USB cameras connected to a powered USB HUB due to technical limitations, they needed to be

activated one at a time). Also allows to take *n* photos with 1 camera, opening and closing it several times to simulate *n* cameras (this functionality was used while 6 cameras were not physically available). It can work with any camera recognisable by the OpenCV-Python library.

- **photosaving.py:** Allows to save photos from a certain test sample, with a certain resolution (pixel density) in a folder, with automatic naming.

- **photoexporting.py:** Allows to compress selected folders with test sample photos into a .zip file and save it.

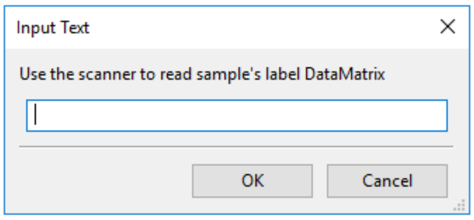
Figure 5.5 shows a screenshot of the resulting HMI's main screen. The "NEW SAMPLE" button triggers the DataMatrix reading routine, whose window can be seen in Figure 5.6. The "START" button triggers the photo taking and saving routine (which is mainly handled by Python open-source libraries OpenCV-Python and PIL), as well as the results displaying. The "EXPORT" button triggers the .zip exporting routine, whose windows can be seen in Figure 5.7. Finally, the button with a folder icon opens the results folder.



Sample info will be displayed here

Welcome to Sprint 1's basic HMI.
Press NEW SAMPLE and read first sample's DataMatrix.

Figure 5.5 Sprint 1's HMI



Use the scanner to read sample's label DataMatrix

Figure 5.6 HMI's label reading window (S1)

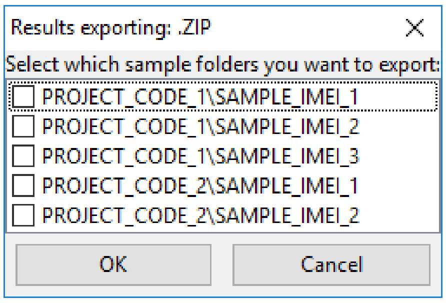
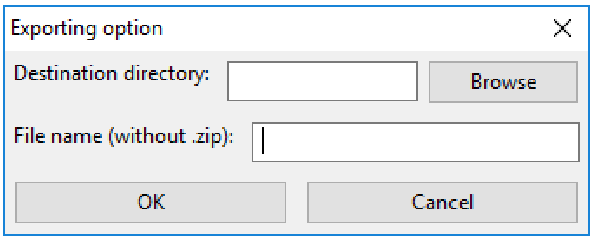

Results exporting: .ZIP
Select which sample folders you want to export:
☐ PROJECT_CODE_1\SAMPLE_IMEI_1
☐ PROJECT_CODE_1\SAMPLE_IMEI_2
☐ PROJECT_CODE_1\SAMPLE_IMEI_3
☐ PROJECT_CODE_2\SAMPLE_IMEI_1
☐ PROJECT_CODE_2\SAMPLE_IMEI_2
OK
Cancel


Figure 5.7 HMI's .zip exporting windows (S1)

5.3. Sprint 2: SW improvement

Jira tasks for the sprint (Start date: 03/04/2018 , Due date: 09/04/2018)	Estimated time	Spent time
Validate Sprint 1's HMI release	2h	1,5h
Fix known errors for Sprint 1's HMI release	2h	1,5h
Perform first HMI improvement loop	4h	6h
Improve logs display and implement log files generation	4h	4h
Implement current working project selection	3h	2h
Implement results erasing	2h	2h

Table 5.3 Jira tasks for Sprint 2

5.3.1. Sprint approach

The second sprint of the project also focused on software development. Its main objective was to improve the existing basic HMI in order to have a solid interface that could be used, with minor modifications, for the rest of the project and the PhotoBench's future laboratory use.

The sprint's tasks consisted in validating previous sprint's HMI (testing its functionalities in order to find errors/bugs), fixing its found errors, implementing new HMI control elements, redesigning HMI's display, adding log files generation, adding project selection and adding results erasing. The process of previous sprint's HMI validation and bug fixing was performed in each sprint until the first PhotoBench release.

5.3.2. HMI validation and bug fixing

At the beginning of each sprint, from Sprint 2 to Sprint 6, the student validated and fixed the previous sprint's HMI. Taking into account the seventh design requirement in Table 3.1, he tested the application in both a Windows laptop and a Raspberri Pi 3B. Each of the HMI's control elements (buttons, text displays, etc.) and functionalities (photo taking and saving, DataMatrix reading, results folder opening, etc.) were tested in search of errors/bugs that could make the application malfunction or crash. Appendix C (Section 2) presents a full report of the bugs found during each sprint's validation, as well as how they were fixed.

5.3.3. HMI improvement and changes

As new functionalities for the HMI were going to be implemented during this sprint, extra control elements were added: a button for file erasing and a drop-down list for project selection (both elements are available as native widgets in wxPython). In the 'Results' window, two small changes were made: a watermark was added to example photos, in order to clarify which of the displayed photos were the example ones, and photo numbering was no longer displayed under each photo pair. Figure 5.8 shows one of HMI's example photos for this sprint.

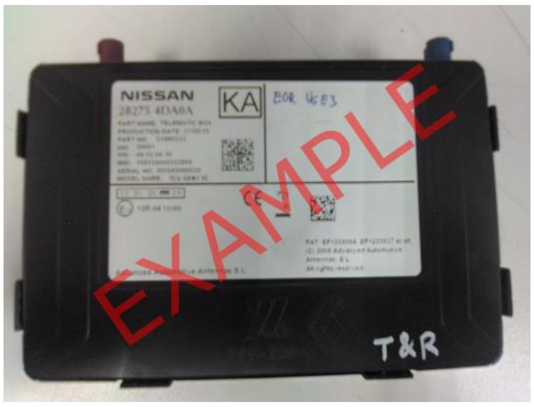
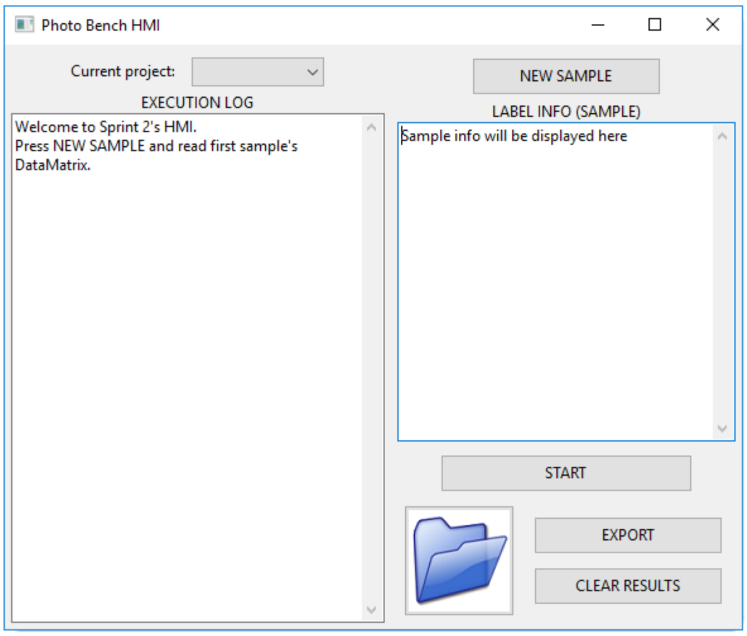


Figure 5.8 Sprint 2's example photo of a TCU's front side

Last but not least, HMI's display was restructured in order to show information "from top to bottom", that is, showing at the very top the information that the user first needs and progressing down with the next information he/she will need, so that the information flux follows the user's logical application use order, as can be seen in Figure 5.9.



Welcome to Sprint 2's HMI.
Press NEW SAMPLE and read first sample's DataMatrix.

Sample info will be displayed here

Figure 5.9 Sprint 2's HMI

5.3.4. Logging

Logging is a way of tracking different events that happen during software execution. This events are recorded in certain files, called "logs", including the most relevant information in them to be able to quickly recognise the event, the conditions in which it happened and to solve any issue related to it if necessary. Following the eighth design requirement presented in Table 3.1, log recording was implemented into Sprint 2's HMI using the built-in Python module logging, and saving the .log files in a dedicated folder. Following the style from existing applications in Ficosa A.C., log messages in this project obey the following format, where "Level" is a standard indicator of the event's level of severity [12]:

dd/mm/YYYY HH:MM:SS | *Level* | *Log message*

There are five levels of event severity (ordered from less to more critical): DEBUG, INFO, WARNING, ERROR and CRITICAL. Only log messages with an INFO level of severity or higher are recorded in PhotoBench's log files.

Apart from log files implementation, execution log display in HMI's main screen was changed in order to give its users only relevant information about the execution, following a clearer format.

5.3.5. Project selection and configuration files

At Ficosa A.C., the label's information and its arrangement, as well as how much of it goes into the DataMatrix and in what order, are agreed between the client and the company before the project's start. For that reason, the PhotoBench's HMI needs to know from which project a TCU test sample is before trying to interpret its DataMatrix information.

For the PhotoBench, the working project's automatic configuration is accomplished by a drop-down list in the HMI's main screen, used to select the project in which the laboratory technician is working on at each moment, together with a set of configuration ("config") files, one for each project the HMI will work on, with a format that the application can interpret, in order to know the current's project label configuration by only knowing the project's code.

Each config file has the project's code (which is agreed between Ficosa A.C. and the client) as its name, has .cfg file extension, and is stored in a dedicated folder. The HMI recognises its "supported projects" (projects for which it has a defined config file) once, when the application is being initialized, and then offers the user a list with those projects to choose from, each identified by its project code, every time he/she clicks the drop-down list element.

When a project code is selected, the HMI handles the configuration changes internally by interpreting the config file associated to the project. This config file also contains the information required to check if a code read by the user is valid for the working project (using a set of valid expected code lengths for that project, as the lengths of the codes for each project can only have a set of values agreed with client).

5.3.6. Sprint results

At the end of the sprint, a solid HMI with all needed control elements for this project was implemented and ready for later functionalities' integration. Two extra modules were added during this sprint:

- **fileerasing.py:** Allows to erase all files and subfolders in a certain folder.

- **projectselection.py:** Allows to generate program's appropriate configuration and supported projects list.

Figure 5.9 shows a screenshot of the resulting HMI's main screen. The "CLEAR RESULTS" button triggers the results clearing routine, which erases everything in the results folder (previously asking the user for confirmation), but leaves the logs intact. The file clearing is also recorded as an info-level log message.

5.4. Sprint 3: MEC strategy

Jira tasks for the sprint (Start date: 09/04/2018 , Due date (extended): 18/04/2018)	Estimated time	Spent time
Define MEC strategy	4h	7h
Validate Sprint 2's HMI release	2h	2h
Fix known errors for Sprint 2's HMI release	2h	3h
Implement basic image similarity detection	4h	9,5h
Perform second HMI improvement loop	1h	1,5h

Table 5.4 Jira tasks for Sprint 3

5.4.1. Sprint approach

The third sprint of the project focused on defining a clear strategy to build a mechanical structure prototype in order to test the PhotoBench's software and hardware together in fixed conditions and to have stable and homogenous light conditions for the photos, thus tackling the thirteenth design requirement defined in Table 3.1.

The sprint's tasks consisted in stablishing a clear strategy to generate the mechanical structure prototype, validating previous sprint's HMI and fixing its found errors, implementing a similarity check between taken photos and example photos (to detect possible problems in the taken photos) and implementing automatic 'Results' window closing when another 'Results' window is created or the application is closed.

5.4.2. Mechanical prototype strategy

For this project, in order to make a mechanical structure prototype that could give both support to the hardware and homogenous lighting, two available alternatives were studied: modifying and using a commercial photo light box, or building a custom structure.

The commercial photo light box was a cheap and direct option that consists on a box-like structure usually made of plastic, with its fabric walls covered interiorly with a reflective material (usually white), as shown in Figure 5.10. While this solution may be able to work as a structure prototype if modified properly, it wasn't robust enough to integrate the cameras and a test sample support permanently into it and didn't give the professional look that the company was looking for (stated in the eleventh design requirement in Table 3.1), as it looked cheap and fragile.

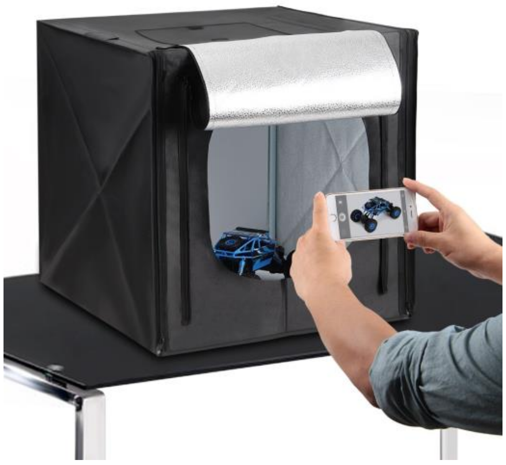


Figure 5.10 Commercial photo light box example [Source: Amazon]

After some meetings with laboratory technicians, department colleagues and supervisors, the project's tutor and the department supervisor, taking into account the abovementioned limitations of the commercial light box, agreed with the student to ask the company's Prototype department to manufacture a custom mechanical structure prototype (the detailed MEC design was out of this project's scope), which would need to be rigid, robust, cube-shaped, and with a professional look.

With that agreement, all that was left was to give Prototype department some indications of the mechanical structure's requirements, as well as assist them in its design and any issue related (which was part of the next sprint's main focuses). Ninth, tenth and eleventh design requirements in Table 3.1 were given to Prototype department as they are, as well as an extra parameter: the distance between the cameras and the TCU test samples' centre. This parameter was agreed between the student and the laboratory to be set around 30cm (each side of the TCU is actually at a slightly different distance from the cube), as it was the distance at which they were taking the photos manually. Apart from that parameter, some more general considerations were given: the structure must have a certain door-like system to introduce the test samples in the PhotoBench and then close it completely (to avoid light from escaping from the box and external light noise from entering it), the structure must be metallic (preferably a light metal, in order for the PhotoBench to be more portable) and have a cubic shape (same distance camera – sample centre for all cameras), the 6 cameras must be placed in the centre of each lateral panel and perpendicular to it (in order to get photos "from the 6 sides of a cube"), the lateral panels must have a white reflective interior (in order to have homogenous lighting conditions inside of the box), and the test sample support must be transparent (to be able to take photos from both the top and bottom cameras) and be placed centred in the middle of the box's height.

With the abovementioned indications written (and commented in person), a “service request” was formally performed to Prototype department and start and end dates were agreed, which would define most of next sprint’s task timings.

5.4.3. Image similarity

Following the twelfth design requirement in Table 3.1, the student implemented a similarity comparison between taken photos and a set of example photos. In order to compare a pair of photos, several approaches were available:

- Representing the images as arrays (using an open-source Python library like scipy or OpenCV-Python), calculating their pixel-by-pixel difference and then calculating the norm of that difference (several norms can be defined [13], such as the Manhattan norm, the Zero norm or the Euclidian norm).
- Computing some feature vector (like a histogram) for each of the images and then calculating their distance [14], or comparing the histograms with OpenCV-Python’s methods [15].
- Computing special indexes designed to measure image similarity, like the Structural Similarity Index, SSIM [16] (which can be computed easily using the open-source Python library scikit-image).

After testing the different options, colour histograms comparison with OpenCV-Python’s “Chi-Squared” method was chosen, as it showed the best results for this particular project, being able to tell when a foreign object was put instead of a TCU test sample, while not being very sensible to small variations in TCU’s design (which was really desirable, because that way any TCU test sample from any project could be directly compared to the same set of example photos without raising many false similarity mismatches if a certain small design aspect of the TCU changed between projects).

5.4.4. Sprint results

At the end of the sprint, a mechanical structure prototype was agreed to be implemented by Prototype department, following the general indications given by the student. A new module was also implemented:

- **similaritydetection.py:** Allows to compare two images, using a certain OpenCV-Python comparison method and checking if they’re similar enough according to a certain threshold.

5.5. Sprint 4: MEC implementation and HW selection

Jira tasks for the sprint (Start date: 18/04/2018 , Due date (extended): 02/05/2018)	Estimated time	Spent time
Follow MEC strategy	4h	3,5h
Validate Sprint 3's HMI release	1h	1h
Fix known errors for Sprint 3's HMI release	1h	4,5h
Define final camera HW	10h	2h
Perform third HMI improvement loop	4h	4,5h

Table 5.5 Jira tasks for Sprint 4

5.5.1. Sprint approach

The fourth sprint of the project had two main focuses: implementing the mechanical structure prototype and selecting (and buying) the final prototype's cameras, so that software and hardware could be tested together in the next sprint.

The sprint's tasks consisted in following the implementation of the mechanical structure prototype, assisting Prototypes department with any design and/or manufacturing aspect or issue that needed to be addressed, validating previous sprint's HMI and fixing its found errors, choosing the most suitable camera for the final PhotoBench prototype, and implementing a shortcut/launcher that does not make the system console appear during HMI's execution.

5.5.2. Mechanical prototype implementation

According to agreed implementation strategy, Prototype department implemented the mechanical structure prototype in 40h of technician work. During that time, the student assisted the technician by taking any design decision necessary, like deciding how to integrate the lateral panels with the aluminium profiles and the methacrylate support, how to place the cameras in the panel (outside of the interior cavity, in this case), how big should the panel holes for the cameras be, etc. The final duration of 40h was considered rather slow both by the student and his department, but the result had a professional look, so it was considered a satisfactory job. Nevertheless, as it was a prototype, the precision of the job was not the optimal, and things like profile perpendicularity or camera holes' centre's precise location were not accomplished as well as they could've been, so the mechanical structure prototype needed to be manually adjusted to be usable enough. Figure 5.12 and Figure 5.15 show different parts of the prototype's mechanical structure before and after integrating the cameras onto it.

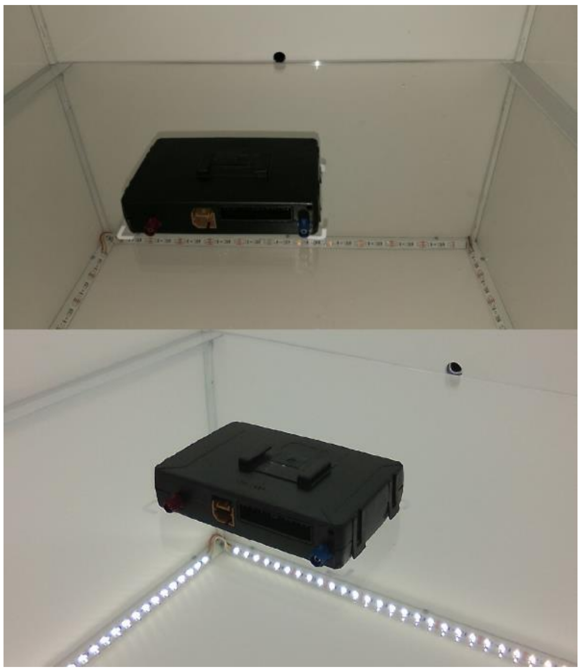


Figure 5.11 MEC prototype's interior, with a transparent support, LED strips and panel holes for the cameras. The upper image shows a TCU with the LEDs off, while the lower image shows a TCU with the LEDs on

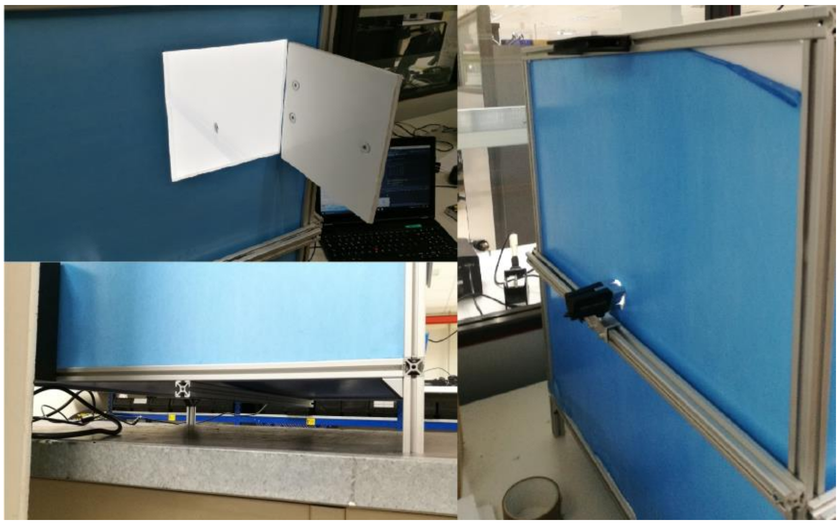


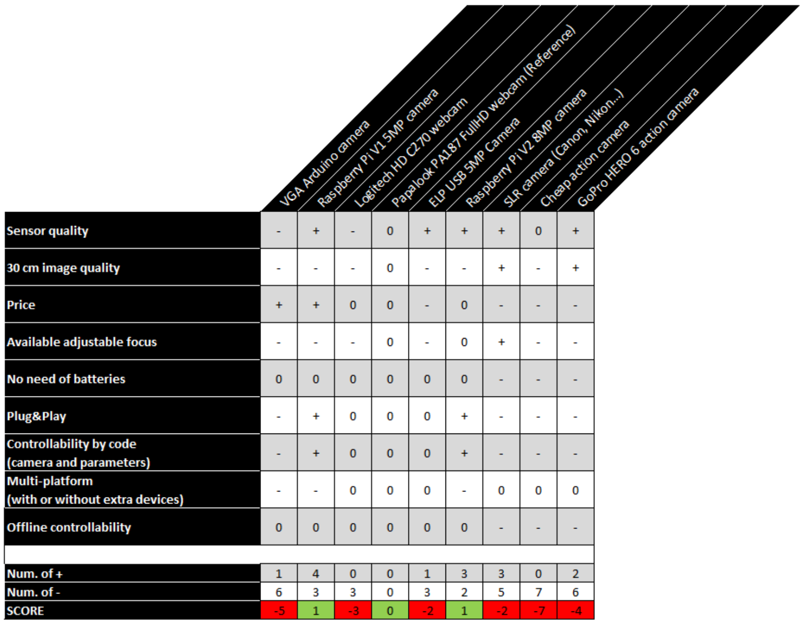
Figure 5.12 MEC prototype's exterior. The upper-left image shows the bench's door, the lower-left image shows the bench's legs and bottom without a camera integrated, and the right image shows one of the bench's handles and the blue film over its white panels

5.5.3. Final prototype’s camera selection

In order to reach a final prototype, the six cameras for the bench had to be chosen and integrated into the MEC prototype, thus tackling the fifteenth design requirement in Table 3.1. Before this decision was made, everything was being tested using either the initial prototype’s camera or a laptop’s internal camera, as they were both recognisable by OpenCV-Python library.

To achieve an acceptable choice, the student made another alternatives analysis, this time using two project management classic techniques: the filter matrix and the decision matrix [17]. Before applying those techniques, research was done again regarding available cameras in the market.

First, in order to perform a first filter, the student performed a filter matrix, which includes every alternative considered. A certain alternative is taken, more or less arbitrarily, as the “reference” alternative (in this case the camera from the initial prototype, as it had shown to be already a good starting point). Then, all alternatives are compared against the reference over different criteria (related to the expected performance of the chosen alternative), and each is given a qualification of ‘+’ if it performs better than the reference alternative in that aspect, ‘0’ if it performs more or less as good as the reference alternative, and ‘-‘ if it performs worse. Finally, a score is calculated for each alternative candidate, adding 1 point for each ‘+’ qualification and subtracting 1 point for each ‘-‘ qualification. The alternatives with negative scores are discarded directly. Figure 5.13 shows the results of this technique.

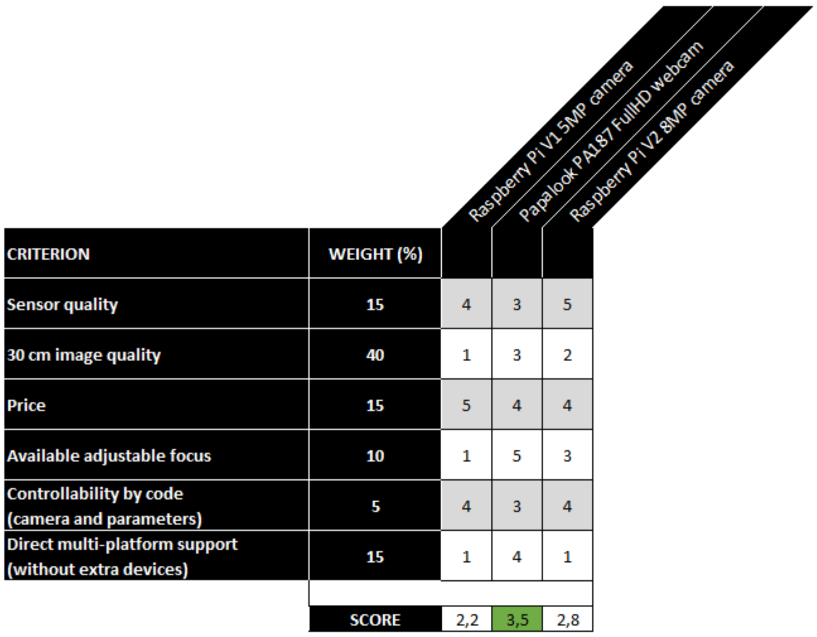


	VGA Arduino camera	Raspberry Pi V1 5MP camera	Logitech HD C270 webcam	Papalook PA187 FullHD webcam (Reference)	ELP USB 5MP Camera	Raspberry Pi V2 8MP camera	SLR camera (Canon, Nikon...)	Cheap action camera	GoPro HERO 6 action camera
Sensor quality	-	+	-	0	+	+	+	0	+
30 cm image quality	-	-	-	0	-	-	+	-	+
Price	+	+	0	0	-	0	-	-	-
Available adjustable focus	-	-	-	0	-	0	+	-	-
No need of batteries	0	0	0	0	0	0	-	-	-
Plug&Play	-	+	0	0	0	+	-	-	-
Controllability by code (camera and parameters)	-	+	0	0	0	+	-	-	-
Multi-platform (with or without extra devices)	-	-	0	0	0	-	0	0	0
Offline controllability	0	0	0	0	0	0	-	-	-
Num. of +	1	4	0	0	1	3	3	0	2
Num. of -	6	3	3	0	3	2	5	7	6
SCORE	🟥 -5	🟩 1	🟥 -3	🟩 0	🟥 -2	🟩 1	🟥 -2	🟥 -7	🟥 -4

Figure 5.13 Camera alternatives analysis: Filter matrix (in red, discarded alternatives; in green, alternatives that pass the first filter)

Then, the student performed a decision matrix, which only includes the alternatives that passed the first filter, in order to make the final decision. Different criteria are given different weights (in %, for a total sum of 100%), and each alternative is given an independent qualification from 1 to 5 for each criterion, being 5 the highest qualification, regarding its performance in that certain aspect. Finally, each alternative obtains a score, based on the sum of the products between each criterion qualification and its weight. The alternative with the highest score is chosen.

In this case, weights were assigned considering the quality that the camera could give with the agreed camera-sample distance (30 cm) as the most important factor (40% weight), and then camera general versatility (better sensor quality and multi-platform direct support) for possible future mechanical structure reworks as the second most important factor, together with price (15% weights). Figure 5.14 shows the results of this technique, where the USB webcam from the initial prototype showed the best results, being theoretically a non-optimal but good-enough solution.



CRITERION	WEIGHT (%)	Raspberry Pi V1 5MP camera	Papalook PA187 FullHD webcam	Raspberry Pi V2 8MP camera
Sensor quality	15	4	3	5
30 cm image quality	40	1	3	2
Price	15	5	4	4
Available adjustable focus	10	1	5	3
Controllability by code (camera and parameters)	5	4	3	4
Direct multi-platform support (without extra devices)	15	1	4	1
	SCORE	2,2	3,5 🟩	2,8

Figure 5.14 Camera alternatives analysis: Decision matrix (in green, the finally chosen solution)

5.5.4. Sprint results

At the end of the sprint, a mechanical structure prototype was successfully implemented, and the final prototype cameras' model was chosen and integrated into the structure, as can be seen in Figure 5.15. A shortcut to launch the application more easily was also created, as detailed in User Manual in Appendix D (Section 3.2).

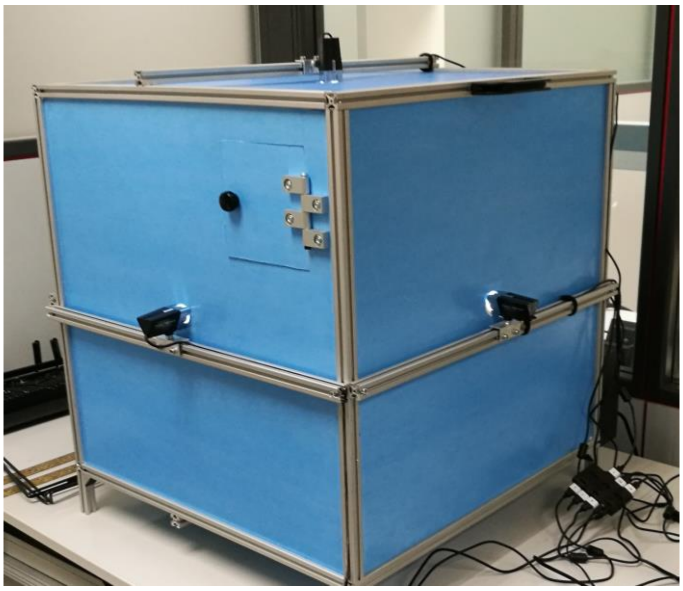


Figure 5.15 MEC prototype with cameras integrated

5.6. Sprint 5: HW-SW interaction and MEC adjustment

Jira tasks for the sprint (Start date: 02/05/2018 , Due date (extended): 14/05/2018)	Estimated time	Spent time
Validate Sprint 4's HMI release	1h	0,5h
Fix known errors for Sprint 4's HMI release	1h	1h
Implement automatic camera selection	12h	29,5h
Adjust cameras (bench structure and parameters)	6h	5,5h
Perform fourth HMI improvement loop	5h	5,5h

Table 5.6 Jira tasks for Sprint 5

5.6.1. Sprint approach

The fifth sprint of the project focused on testing the hardware and software together, through the use of the mechanical structure prototype, in order to detect and fix any problem and move towards a fully-functional setup for the PhotoBench full prototype (as requested by the fifteenth design requirement in Table 3.1).

The sprint's tasks consisted in validating previous sprint's HMI and fixing its found errors, implementing automatic camera selection (solving the technical problem that appeared with it), adjusting cameras' position, orientation and capture parameters, and configuring high and reduced resolution photos saving (as specified in the fifth design requirement in Table 3.1).

5.6.2. Automatic camera selection: problem and solution

When the six cameras, integrated in the MEC prototype, were tested together with the software, a technical problem appeared: the HMI needed to know which camera took each photo (“in which position was each camera”), in order to save the photos with the desired numbering and to display the photos in the desired order, as well as knowing with which example photo should each taken photo be compared (example photos’ naming is related to cube side positions’ desired order).

From the HMI’s point of view, cameras are identified by a “camera index” (which goes from 0 to any number) that OpenCV-Python assigns to each camera that is connected to the device that runs the application, so there was a clear need to determine the relation between camera indexes and the cameras they pointed to. In fact, the only important information about each camera for the HMI regarding this problem was its location in the cube, so it was enough if the HMI knew a relation between camera indexes and camera positions.

In order to determine that relation, OpenCV-Python’s camera index assignment system was investigated, both in dedicated webpages and forums, and in the library’s documentation and source code. After some experimental tests to prove what seemed to be the answer, the student concluded that the camera indexes’ assignment depends on the order in which the computer detects the cameras when the USB HUB is connected, and any camera connection/disconnection may alter that order (the library internally uses an O.S. device-identifying parameter that behaves this way, both in Windows and Raspbian). This meant that there was no way to robustly relate camera indexes and camera positions by software using OpenCV-Python directly as the photo taking library.

Several options were considered once this technical limitation was detected:

- Forcing the O.S. device-identifying parameter to behave in a different way, which after some research and testing was considered as something almost unfeasible and by no means comfortable, as it required manually modifying how the O.S. behaved, which was difficult, non-automatic and risky.

- Changing how the OpenCV-Python library referred to each camera and recompiling it, which after some research was considered too complicated and not very practical, as it was related with a lot of O.S. event handling programmed in C++ (the original language of the library) and changing it manually would be needed every time the OpenCV-Python library was updated in the control device or the control device changed, which would go against the seventh design requirement in Table 3.1. Apart from that limitation, the 6 USB chosen cameras had no characteristic constant parameter to distinguish them from the control device, as all their parameters were either identical or depended on USB connection order, so all in all this option was unfeasible.

- Changing the way photos were taken from the control device, which proved not to be a solution, as no characteristic parameter to robustly and permanently distinguish the cameras from the control device was available, so changing the Python library or taking the photo through command-line commands would encounter the same problem.

- Designing a cube-like “calibration object” that presented different recognisable patterns, shapes or colours in its sides. If the cube is always placed in the same “reference position”, by taking a photo from each camera (without previously knowing which camera is in which position), the HMI can know to which position of the cube structure the photo corresponds by recognising the pattern, shape or colour of the cube side in the photo. This was the chosen approach, as a quick provisional/improvable solution, as laboratory technicians agreed it had almost no impact in the process time (the cube has to be used only once if the application is not closed), so they were ok with it.

- Changing the hardware approach, by adding an extra controller device (like, for example, a Raspberry Pi Zero board) for each camera. Each extra controller device would only see 1 camera, so there would be no problem for them to take a photo of the only camera connected to them, which would always be the same (physical connection would not change). Then, each extra controller should send the photo to the main controller, “identifying himself” in this transmission (which is easily done, for example, by MAC address or other identifiers, depending on the transmission protocol used), so that the main controller knows exactly from which position comes each photo. This approach was left as a non-implemented alternative plan, as it would require extra economical investment, as well as a change in the photo taking Python module implementation, and there already was another approach that solved the problem more easily and at almost no cost.

The “calibration cube” was temporarily implemented as a fast prototype, using two small boxes and printed pieces of paper, joined together with adhesive tape, as can be seen in Figure 5.16. Each cube side had a letter printed on it (from A to F), following a 7-segment display style [18]. The cube’s detailed design can be found in the User Manual, in Appendix D (Section 2.4).

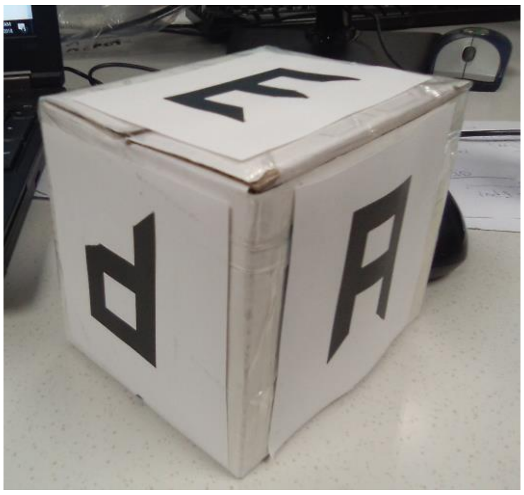


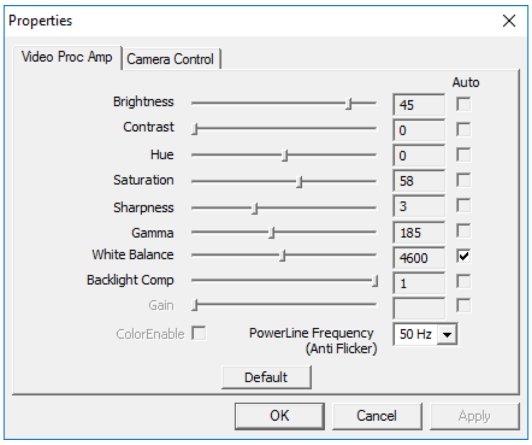
Figure 5.16 Calibration cube fast prototype

A computer vision algorithm (adapted from another algorithm solving a similar problem [19]) was implemented to locate and recognise the letter, using OpenCV-Python methods. First, the image is converted into grayscale and “cleaned up” from noise (using blurring, thresholding, morphological operations…). Then, contours (for instance, the outline of a letter) are found within the treated image. For each contour, a bounding box is calculated. If the bounding box’s size is within an experimentally calculated range (distance between cameras and the cube doesn’t change much, so it can be left as a constant in the program), the algorithm assumes it has found the letter. This recognition is robust enough if the 7-segment’s edges are sharp (they were designed to be sharp for this reason). Once the letter’s bounding box has been found, only that region of the image is taken into account, and is divided into rectangular regions representing the segments of the 7-segments. For each rectangular region, the percentage of black pixels is calculated, and the segment is considered to be ON/full if that percentage exceeds a set value, or OFF otherwise. When all segments’ states (ON or OFF) have been detected, the HMI knows which letter the 7-segment display is representing, and can relate it to a certain position through an internal constant dictionary, as the cube must always be placed with the A letter oriented towards the user (front side of the PhotoBench).

If the detected 7-segment pattern is not a letter from A to F, or no pattern is detected, the HMI ignores the camera that took that photo, thus letting it work with devices with extra internal cameras, like laptops, even if those cameras are activated during the calibration (internal cameras may get a low camera index, and camera indexes are tested sequentially, as the HMI does not know initially which camera indexes are “the right ones” to test).

5.6.3. Cameras adjustment

In order to adjust the cameras' position, orientation and capture parameters, they were activated one by one with the Windows program AMCap, which allows for real-time video display and capture parameters control. Different available parameters (not every parameter was controllable by software with the chosen cameras) were tested until a good-enough configuration was found, which worked with all the cameras (it can be seen in Figure 5.17). The cameras' position and orientation were adjusted changing the aluminium profiles' position and orientation, which were built by Prototype department to be adjustable through Allen screws and guidewires (integrated in the aluminium profile's shape).



Properties

Video Proc Amp | Camera Control

		Auto
Brightness	45	☐
Contrast	0	☐
Hue	0	☐
Saturation	58	☐
Sharpness	3	☐
Gamma	185	☐
White Balance	4600	☑
Backlight Comp	1	☐
Gain		☐

ColorEnable ☐ PowerLine Frequency (Anti Flicker) 50 Hz

Figure 5.17 Capture parameters configuration for the final prototype, with AMCap

5.6.4. Sprint results

At the end of the sprint, cameras were successfully adjusted (position, orientation and capture parameters) in order to give the best photos they could, and the 6 cameras could be controlled by the software with the only limitation of needing an initial calibration/recognition, which was implemented by a new module:

- **calibrationcube.py:** Allows to recognise a 7-segment letter from A to F from an image.

5.7. Sprint 6: 1st Release preparation

Jira tasks for the sprint (Start date: 14/05/2018 , Due date: 16/05/2018)	Estimated time	Spent time
Validate Sprint 5's HMI release	1h	1h
Prepare and deliver first PB release	6h	8h

Table 5.7 Jira tasks for Sprint 6

5.7.1. Sprint approach

The sixth sprint of the project focused on putting together everything designed and implemented so far in order to deliver a complete final prototype (hardware, software, and mechanical structure), called the “PhotoBench Release 1” (PB R1), to validate all initially agreed design requirements, as specified in the fifteenth design requirement in Table 3.1.

The sprint’s tasks consisted in validating previous sprint’s HMI and fixing its found errors, testing the prototype’s implemented software, hardware and mechanical structure together against the design requirements defined in Table 3.1, generating PB R1’s User Manual, and performing a demonstration and training of use to laboratory technicians.

5.7.2. Design requirements fulfilment

Having a complete prototype of the PhotoBench, it was necessary to check if it fulfilled all initially agreed design requirements defined in Table 3.1 (from first to fifteenth). The first six design requirements were fulfilled mainly in Sprint 1, with some extra work on the lasts sprints (and the second sprint, in the case of the first requirement). The seventh design requirement was fulfilled during each “previous sprint’s HMI validation and bug fixing”, as cross-platform module and function compatibility was the source of most of the bugs found. The eighth design requirement was fulfilled exclusively during Sprint 2. In Sprint 3, from the ninth to the twelfth design requirements were fulfilled, as three of them directly became design indications for Prototype department. The thirteenth design requirement was fulfilled between Sprints 3, 4 and 5, when the mechanical structure prototype was designed, implemented, and hardware was integrated onto it. Fourteenth design requirement was fulfilled during Sprint 5’s cameras adjustment. Fifteenth design requirement was fulfilled in Sprint 6, as it was its main objective.

So, all in all, every initially agreed design requirement had been successfully fulfilled by this project’s development strategy and implementation. The extra and last design requirement was fulfilled in the next sprints.

5.7.3. PhotoBench Release 1 documentation

For the first release of the PhotoBench (what we also call “complete final prototype”), a User Manual was created, which describes in detail everything related to using the abovementioned prototype (both its hardware and software). It can be found in Appendix D, and can be used for both company’s validations and as a reference to check for any implementation detail that has not been included or has not been explained in depth in this report.

The User Manual also includes the PB R1’s HMI workflow (through Section 3.3 and Chapter 4). Appendix B (Chapter 2) details PB R1’s hardware specifications, while Appendix C (Chapter 3) includes PB R1’s HMI full code.

5.7.4. Sprint results

At the end of the sprint, a fully-functional complete prototype of the PhotoBench was released and documented, which could already be used in Ficosa A.C.’s validations, although its image quality was just good-enough, far from the most desirable quality.

5.8. Sprint 7: Results improvement

Jira tasks for the sprint (Start date: 16/05/2018 , Due date: 04/06/2018)	**Estimated time**	**Spent time**
Improve photo quality results	20h	43h

Table 5.8 Jira tasks for Sprint 7

5.8.1. Sprint approach

The seventh sprint of the project focused on improving PB R1’s photo quality, as it was acceptable but not optimal, thus tackling sixteenth design requirement in Table 3.1 and obtaining a “second release” of the PhotoBench. As the mechanical structure prototype was rather expensive and had proved useful, this improvement had to be made, as agreed with the company, without manufacturing a new structure prototype.

This sprint’s tasks consisted in researching on ways to improve the taken photos’ quality by hardware, and testing each solution candidate until a remarkable quality improvement was achieved.

5.8.2. Quality improvement alternatives

In order to improve taken photos’ quality, the student performed a thorough research, in search of new cameras, new hardware approaches, or any other solution that could remarkably improve photos’ quality without modifying the mechanical structure prototype. He tested each possible solution idea found until a valid solution was obtained.

First, after examining the photos obtained with the PhotoBench’s first release, it appeared as if they had some light noise, as when the photo was zoomed in, details became a bit unclear. The student and the project tutor thought that might be due to non-optimal light conditions for the camera, even though the interior of the PhotoBench looked well lit to the naked eye. The student tried to make the light more diffuse, as they thought LED strips’ light may be too directional, by simulating light diffusors with high-density paper sheets, as can be seen in Figure 5.18. This option didn't improve results, so it was concluded that light conditions weren't probably the cause of the photo's noise / non-clarity.

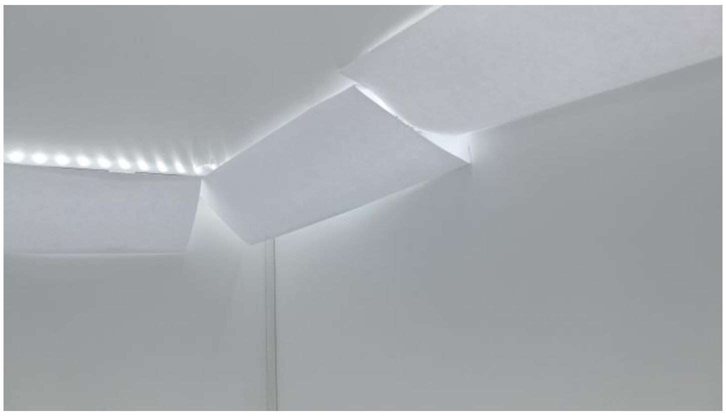


Figure 5.18 Light diffusors simulation with high-density paper sheets

Then, the student did a meeting with a department colleague that had previously worked with cameras and computer vision for an industry, who gave him the idea of changing the cameras' optics by adding extra lenses that could improve the results. In this case, telephoto lenses were needed, as the current cameras were capable of taking high quality photos in closer distances. Setting the camera – sample centre distance to 30 cm as agreed had become a design limitation when trying to improve photo quality, so it was considered and assumed to be a design mistake in the project (which was not so serious, as the initial quality requirements were fulfilled, as reported by laboratory technicians, and it only limited later improvements without building a new mechanical structure).

The student performed research on available telephoto lenses options and found a set of affordable mobile phone attachable lenses (which came with a special "clip"), which he tested on both the current webcam model and the Raspberry Pi V2 camera module (which was the second best option in the final camera alternatives analysis). As can be seen in Figure 5.19, the telephoto (x2 zoom) lenses substantially improved photo quality with the webcam, but a "window" effect appeared due to an optical mismatch between the webcam's lenses and the telephoto lenses. With the Raspberry Pi V2 camera module, as can be seen in Figure 5.20, photo quality was not increased substantially, and it was difficult to adjust the camera's lens focus with precision without a specific small tool for it. Apart from that, as can also be seen in Figure 5.20, distortion [20] appeared near the corners of the TCU's body.

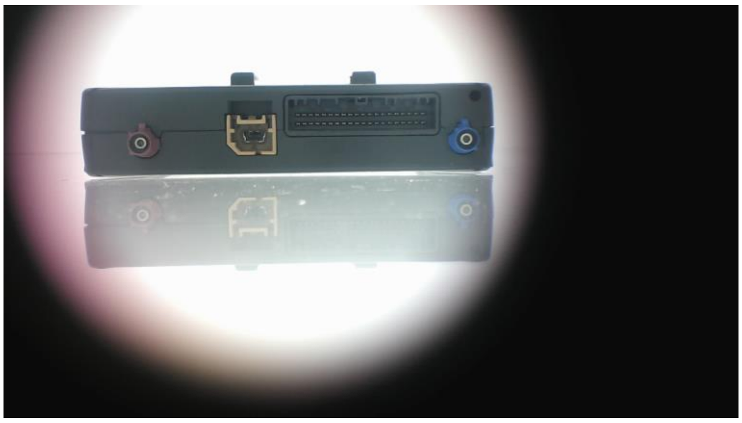


Figure 5.19 Photo taken with the PB R1's webcams + telephoto (x2) lenses

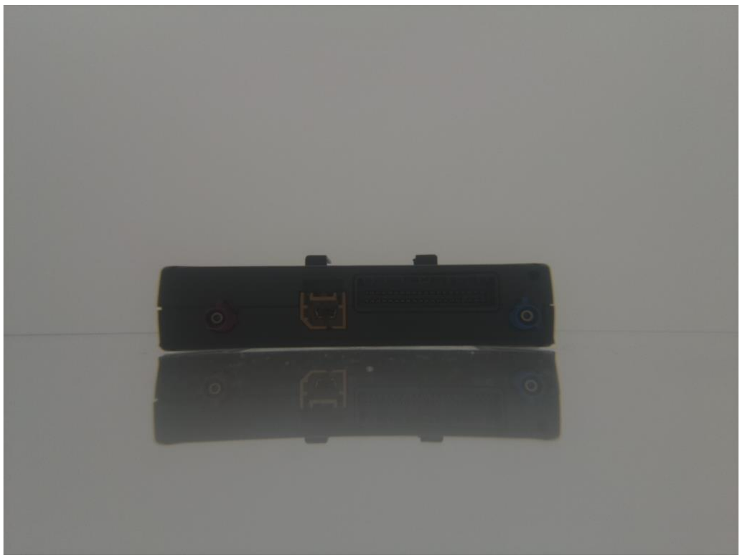


Figure 5.20 Photo taken with the RaspPi V2 camera + telephoto (x2) lenses

After the mobile phone attachable lenses' failure, the student took one of the webcams and opened it to see its parts (as the manufacturer didn't provide that information in the datasheets). The webcam had mainly 2 optical components: the image sensor and the lenses. The image sensor was integrated with the electronics of the camera, so the only part that could be replaced was the lenses. After some research (again, no information was provided on this in the datasheet), the student found they were normalized M12 CCTV board lenses, a type of lenses used in vigilance cameras and many other webcam models, with 'M12' designating the lenses' metric ISO screw thread diameter [21] (in this case, 12 mm). By looking at the webcam's specifications, the student found the lenses' most relevant parameters: an angle of view of 60º and a f/2.0 aperture (this optical parameters are explained in the Glossary). The product's specifications also included the required information about the image sensor, which was a ¼ inch 1080p sensor (which is equivalent to a sensor of around 2MP, as its maximum image resolution in pixels is 1920x1080). With this information and some research on optics basics, the right solution was found. Figure 5.21 shows how the cameras' lenses were changed.

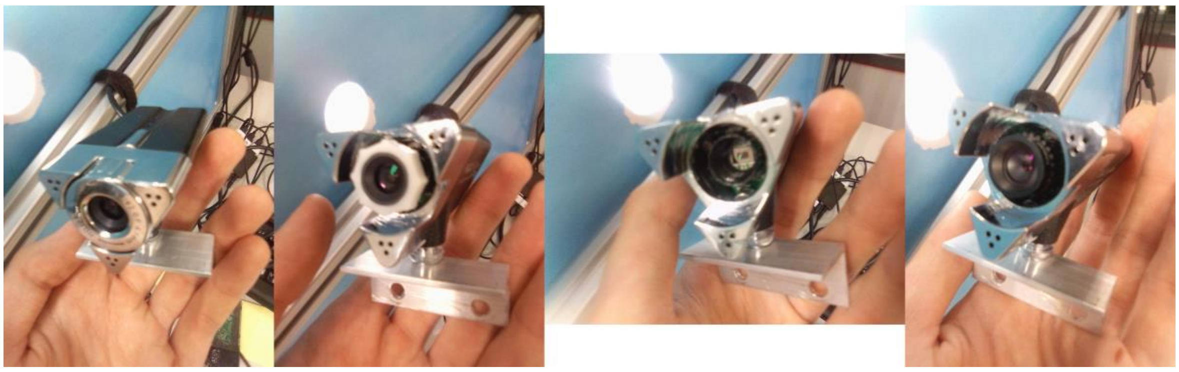


Figure 5.21 Camera lens changing procedure. From left to right: normal webcam, webcam with the focusing wheel detached, webcam with the lens detached (only the camera sensor remaining), webcam with the new lens attached

Calculations were performed, as described in Appendix F (Section 2), for each of the cube positions. For the left and right sides (taking the door side as the front side), the webcams' lenses were changed to 1080p lenses with a focal length of 8 mm (which acted like telephoto lens in this case), which greatly improved photo's test sample detail quality, as can be seen in Figure 5.22, where the TCU test sample can be seen with much more detail.

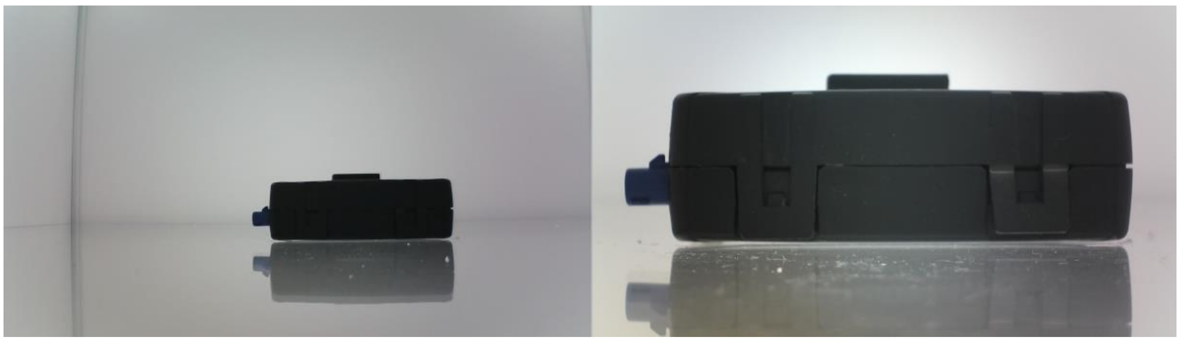


Figure 5.22 Raw images taken by the right camera before (left photo) and after (right photo) the lenses change

For the front and back sides, the webcams' lenses were changed to 1080p lenses with a focal length of 6 mm (which acted like telephoto lenses with less zoom than the 8 mm ones), which improved photo quality enough (not as much as with the left and right sides, but the 8 mm solution wasn't feasible here, as the zoom would be too much and the TCU would not be seen entirely). Results for these cameras can be seen in Figure 5.23.

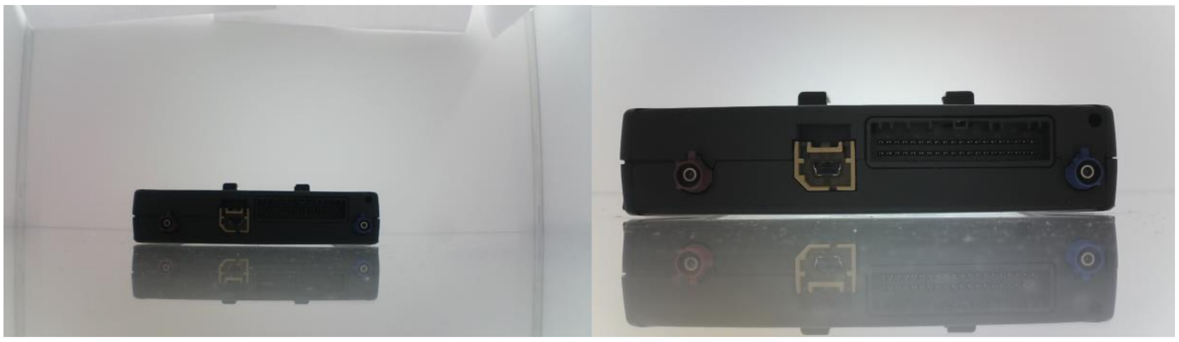


Figure 5.23 Raw images taken by the front camera before (left photo) and after (right photo) the lenses change

Finally, for the top and bottom sides, as the required vertical field of view was too high for the designed distance (those sides see both test sample's biggest dimensions), there was no way to improve the photo quality enough while not zooming too much at the same time, because of the webcam's cheap camera sensor. For this reason, these cameras were changed to USB camera modules with a ½$_{.5}$ inch 5MP sensor, whose M12 lenses were also changed to 5MP lenses with a focal length of 9,8 mm (as the ones that came with the camera modules were of much smaller focal length, needing telephoto lenses again for a good photo from the working distance). This last proposed improvement could not be experimentally tested, as lenses were not available on time, and so its quality improvement was only stated theoretically, but it will be tested as soon as the lenses are available at Ficosa A.C.

5.8.3. Quality improvement results

To evaluate the quality of the obtained "improved" results, and following the sixteenth requirement in Table 3.1, the new PhotoBench release was used to take photos from TCU test samples with cracks and other deformations.

Figure 5.24 shows a photo taken with the right camera (with the new 8 mm lens), where plastic deformation can be observed in the right hook.

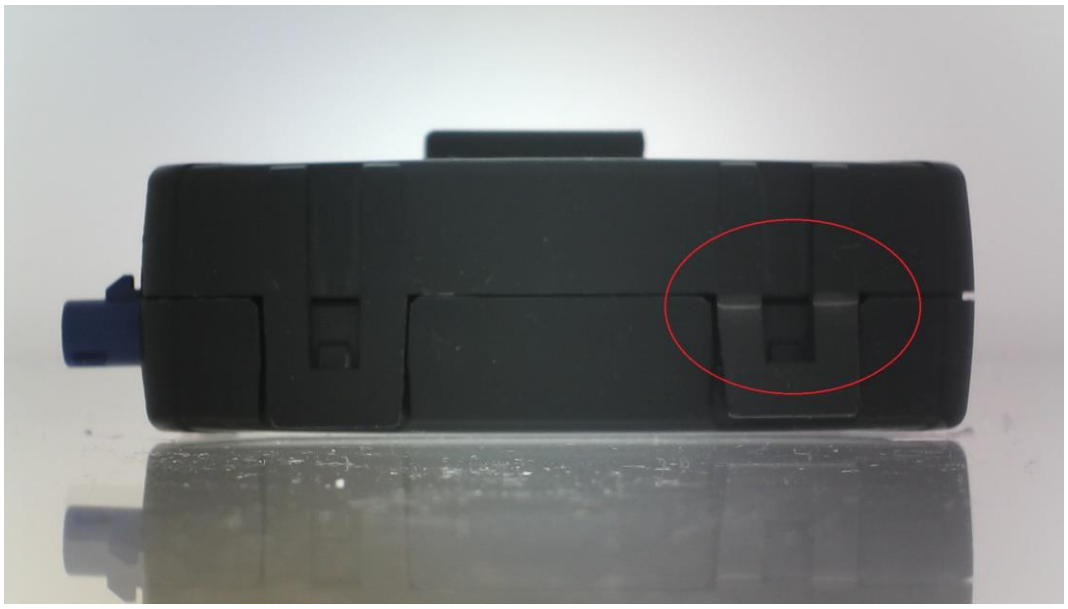


Figure 5.24 Plastic deformation in the right hook of a test sample's right side

As the improvement for the top camera could not be experimentally tested due to lenses' temporary unavailability, a photo of a crack in the union between the TCU and the mechanical test bracket was simulated using the front camera with its new 6 mm lens instead, which yielded inferior pixel density for the TCU details according to calculations in Appendix F (Section 2). This way, as the crack was easily detectable in this photo (Figure 5.25), it would also be detectable with the top camera (it has better pixel density, theoretically).

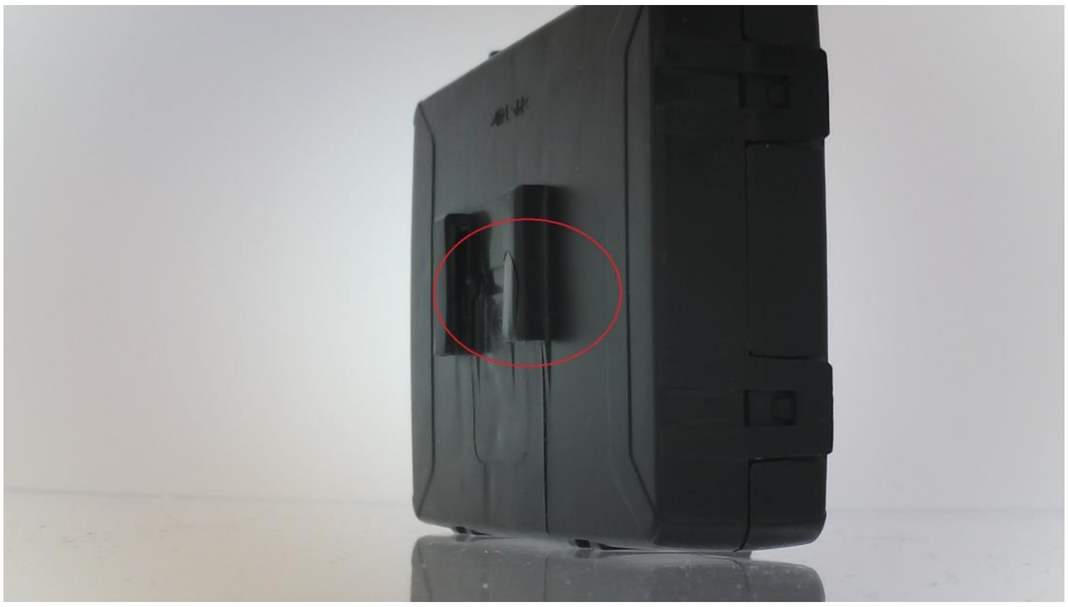


Figure 5.25 Crack in the union between test sample and mechanical test bracket

5.8.4. PhotoBench Release 2 documentation

In order to clearly reflect the changes made between PhotoBench's first and second releases, Appendix B (Section 3) details PB R2's new hardware, while Appendix E details PB R2's other changes (code, mechanic parameters, etc.).

5.8.5. Sprint results

At the end of the sprint, a fully-functional version of the PhotoBench with higher image quality was designed, called "PhotoBench Release 2", which could be used in Ficosa A.C.'s validations right away, as soon as all its new hardware was available and integrated.

5.9. Sprint 8: Proposed redesign

Jira tasks for the sprint (Start date: 04/06/2018 , Due date: 18/06/2018)	**Estimated time**	**Spent time**
Redesign mechanical structure	5h	8h
Document the project	50h	72h

Table 5.9 Jira tasks for Sprint 8

5.9.1. Sprint approach

The eighth and last sprint of the project focused on proposing a redesign for the mechanical structure of the PhotoBench, for any future unit of the device that may be produced. The mechanical structure was optimized for a suitable hardware, so that photo quality results are improved or, at least, quality needs stated in the design requirements in Table 3.1 are fulfilled. As agreed with the company, this proposed design must be directly compatible with the already implemented SW, without additional code reworking (except for, maybe, certain parameters' values). During this sprint, project documentation had to be generated.

This sprint's tasks consisted in calculating the new MEC design's parameters for a chosen HW (using the know-how acquired through the project's development) and generating appropriate full project documentation.

5.9.2. Proposed mechanical structure and hardware change

In order to further improve the photo quality results, this time being able to change the mechanical structure, the student decided to perform calculations for three cameras, finding for each one the optimal distances between camera and test sample (the "working distance") and the obtained pixel density (directly related to image detail quality, as the FOV shall only include the test sample and a bit of extra blank space).

With the calculations detailed in Appendix F (Section 3), the student decided to use the ELP 5MP USB camera partially used in PB R2 (for top and bottom camera positions), with bench's mechanical structure dimensions as drafted in Figure 5.26, where basic structure dimensions, methacrylate position (painted in light grey), position of the panel hole for the camera, and TCU test samples' position (painted in dark grey) are drafted. Further structure details or technical blueprints are not included, as the detailed MEC design is not a part of this project's scope, as stated in Section 3 of Chapter 3 (only those mechanical parameters determining proper hardware's functioning and results' quality, and a mechanical structure prototype for one release, in order to validate HW and SW designs, are included in this project's scope's mechanical aspects). Experimental checks were performed to check the designed working distances' validity, as also detailed in Appendix F (Section 3).

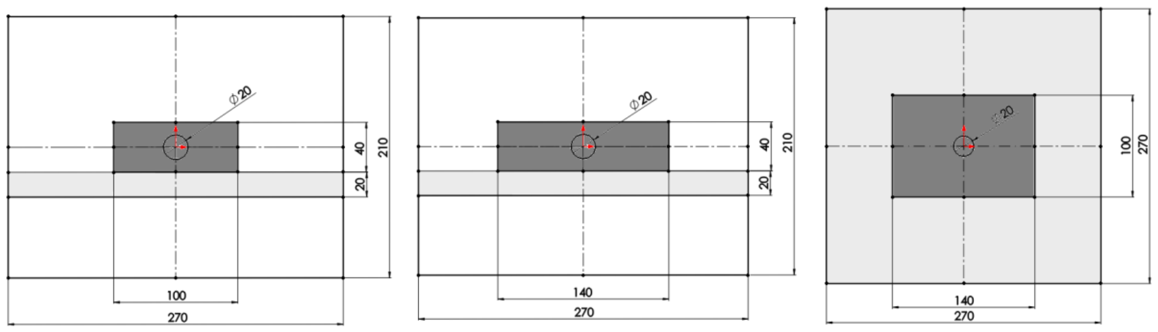


Figure 5.26 Draft of the essential mechanical parameters for PB R3. Drafts for "left to right" direction (left photo), "front to back" direction (centre photo) and "top to bottom" direction (right photo) are shown

5.9.3. Sprint results

At the end of the sprint, a new version of the PhotoBench was released (PhotoBench Release 3), this time only as a design for future units of the PhotoBench to be produced, which will be smaller, lighter and with higher photo quality than the prototype that became PB R1 and, later, PB R2.

With this, the project's development was completed, yielding as a result a fully-functional prototype of the PB R2 (made from adjusting the PB R1's prototype) and a redesign for future benches, the PB R3, a better solution for the problem, both successfully fulfilling all project's design requirements presented in Table 3.1.

ETSEIB

6. Economic study

This chapter analyses the economic consequences of the project for the company. The results of this section are especially important, as this is an improvement project with the objective of validations' cost reduction (through process time reduction).

6.1. Development quotation

Developing a project (studying the problem, analysing possible solutions and developing a final detailed solution design) always has some costs, usually high (specially for new products). The engineer(s) implied in the project have a cost per hour, and they usually need to buy products and devices during the project's development that end up not being used for the final product.

Table 6.1 includes a simplified quotation of the development costs, including the student's salary and all the test material (non-available at the company) he required but didn't end up being used in the final prototype (as, in this project, the final prototype ended up being the first execution of the design, thus being used in real Ficosa A.C.'s validations). Detailed development quotation can be found in Appendix G (Section 1).

Itemized costs	**Quantity**	**Unit price (€)**	**Amount (€)**
Student's working project hours	316	8,00	2528,00
Tested but not-chosen cameras and their tools	-	-	47,54
Mobile phone attachable lenses kit	1	26,00	26,00
TOTAL QUOTE			**2601,54**

Table 6.1 Projects' development quotation summary

6.2. Execution quotations

Once the project is developed and a solution design is reached, turning that design into a physical entity also implies costs, which can be higher or lower than development costs, depending on the project. All materials and manufacturing manpower must be taken into account.

Table 6.2 includes a simplified quotation of the execution costs for the PhotoBench Release 1. Table 6.3 and Table 6.4 include the same for PhotoBench Releases 2 and 3.

Detailed execution quotations can be found in Appendix G (Section 2).

Itemized costs	Quantity	Unit price (€)	Amount (€)
Prototype department's working hours	40	43,27	1730,80
1080p FullHD Webcam	6	21,99	131,94
USB Barcode Scanner	1	52,98	52,98
7-port powered USB 2.0 HUB	1	14,45	14,45
Mechanical structure materials	-	-	553,90
TOTAL QUOTE			**2484,07**

Table 6.2 PB R1 execution quotation summary

Itemized costs	Quantity	Unit price (€)	Amount (€)
Prototype department's working hours	40	43,27	1730,80
1080p FullHD Webcam	4	21,99	87,96
5MP USB Camera Module	2	57,00	114,00
Telephoto lenses, various types	6	-	108,62
USB Barcode Scanner	1	52,98	52,98
7-port powered USB 2.0 HUB	1	14,45	14,45
Mechanical structure materials	-	-	553,90
TOTAL QUOTE			**2662,71**

Table 6.3 PB R2 execution quotation summary

Itemized costs	Quantity	Unit price (€)	Amount (€)
Prototype department's working hours	40	43,27	1730,80
5MP USB Camera Module	6	57,00	342,00
USB Barcode Scanner	1	52,98	52,98
7-port powered USB 2.0 HUB	1	14,45	14,45
Mechanical structure materials	-	-	346,03
TOTAL QUOTE			**2486,26**

Table 6.4 PB R3 execution quotation summary

For PhotoBench's Releases 1 and 2 execution quotations, the mechanical structure quotation is made based on the prototype used during the project's development, which proved to be usable in real Ficosa A.C.'s validations., while for PhotoBench Release 3 execution quotation it is made based on material costs estimated from the other two releases (using proportionality factors for materials' used volumes/areas). Development and execution quotations show similar cost totals for this project. Naturally, development costs are paid only once, and any other time a new unit of the PhotoBench is produced, only the execution costs must be paid.

6.3. Economic feasibility

Almost every engineering project's success depends on its economic feasibility. Projects imply a series of costs, usually high, both for development and execution, and the project's outcome must compensate for those costs, usually economically. In order to determine if this project's resulting design was worth the company's invested effort, the student and the laboratory estimated process times with the PhotoBench (testing the device's prototype) and without it (based on previous experience).

Typical Reliability validations at Ficosa A.C. consist of initial and final functional tests, performed to every test sample, and a series of other tests (an average of around 60 tests of 3 samples per test). After each of these tests, photos are taken from all the samples involved in the test, from each of their 6 sides. If this process is made manually, an average of 20 min is spent for each test, except for functional tests, which require around 3 hours (as reported by laboratory technicians). If this process is made with the PhotoBench's prototype instead, an average of 6 min per test was estimated, except for functional tests, which were estimated to require around 1 hour.

With the abovementioned estimations, the time reduction is expected to be of around 18 hours per validation, which leads to saving 720€ per validation, given that laboratory technicians' work costs 40€/h. If each year 6 validations are performed (company's current estimation), with a project horizon of 2 years (not really big, as this was a small project from which the company expected to get good and fast results), the project is expected to have a Net Present Value (NPV, 'VAN' in Spanish) of over 3000€, positive, calculated with a discount rate of 2,5% (given by Ficosa A.C.), and an Internal Rate of Return (IRR, 'TIR' in Spanish) of around 40%, much higher than 2,5%. These two indicators confirm that the project is expected to be profitable for the company. The expected project's payback period is around 1 year and 3 months. Table 6.5 shows the detailed expected cash balance for this project, used to calculate the previous indicators. In it, company's investment is based on project's development quotation and project's execution quotations (combining their costs appropriately). Detailed economic calculations can be found in Appendix G (Section 3).

PhotoBench project	0	1	2
Total savings		*4320,00*	*4320,00*
Investment	5308,23		
Variable cost			
Total payments	5308,23	*0,00*	*0,00*
Cash flow	-5308,23	4320,00	4320,00
Discounted cash flow	-5308,23	4214,63	4111,84
Accumulated cash flow	-5308,23	-988,23	3331,77

Table 6.5 Project's estimated cash balance
(2 years horizon, 2,5% annual discount rate)

7. Project's impact

This chapter analyses the project's impact on company's internal environment and on the company's surrounding environment.

7.1. Social impact

Process times' reduction thanks to this project's designed product not only saves the company money in each of its Reliability validations, but also changes the worker's conditions during those validations. Thanks to the PhotoBench, workers no longer need to spend big amounts of hours doing repetitive and non-challenging tasks in order to take, modify and save the photos. This could improve their general opinion about their job and their position, as well as greatly increase their productivity during validations, as they would be able to dedicate more time to other tasks that generate value added. Laboratory management could also become less difficult, as smaller validation times make it easier to balance timings when different projects' validations are on-going simultaneously.

7.2. Environmental impact

For any project, it is always important to keep in mind and under control the impact it generates on the environment. In this project, the main materials used in the manufacturing of the mechanical structure prototype (aluminium, foamed PVC, methacrylate…) were recyclable. The main possible source of contamination in this project was the spray paint used to make sure the interior of the PhotoBench was all white, in order to diffuse the light coming from the LEDs better, but it was an isolated source of contamination, being used only for some minutes in a controlled environment in the company's facilities.

Apart from the environmental impact generated by the materials used in the PhotoBench MEC prototype's manufacture, energy consumption must be taken into account too. In this case, the main source of energy consumption is the MEC prototype's lighting, which is electrically powered by a power supply whose specifications are detailed in Appendix B (Section 2). This power supply supplies around 48 W with the lighting on (almost 5 m of LED strip), with a DC voltage of 12 V. Taking into account typical validations' total photo time with the PhotoBench, previously estimated to be around 8 h, total energy consumption for a typical validation would be around 0,384 kWh. With a current average of 6 validations per year, this yields a yearly consumption of 2,304 kWh for the PhotoBench. Producing, treating and using so little energy would have no serious impact on the environment.

Conclusions

The objective of this project was to design an ‘automatic photography bench’ for reliability validations of TCUs in Ficosa A.C., both at SW and HW levels. This objective has been successfully reached, as all three releases of the PhotoBench can be used in real validations. Releases 2 and 3 fulfil every design requirement stated by the company.

The HMI designed as the graphical interface for laboratory technicians provides with all functionalities required during validation processes. It is easy to use, fast and robust, giving the user no source of possible errors and assuring a stable and satisfactory image quality.

The PhotoBench greatly reduces validations’ process times, saving Ficosa A.C. considerable amounts of wasted money, thus being a profitable investment for the company.

PhotoBench Release 2’s top and bottom camera lenses could not be physically implemented and tested, but they will soon be available at Ficosa A.C., and their integration is planned as the next action to take, in order to leave 1 unit of the PB R2 ready for upcoming reliability validations.

PhotoBench Release 3’s design is a great improvement in performance at all levels, being a smaller, lighter and cheaper product than Release 2, and giving better image quality results. It has still not been produced, but it is already planned to.

After the PhotoBench Release 3’s design, there’s still room to improve the project’s solution. Automatic installation files could be created, or even an executable file that needed no Python installation on the target device. Code modularity could be increased, in order to simplify and ease its future maintainability. Camera recognition algorithm could be updated to recognize symbols written in the panels’, instead of using the provisional “calibration cube” solution. Changing the HW approach to use a RaspberryPi V2 camera module could also be another path to investigate.

In the future, this project could be followed up by other business units at Ficosa, which have the same problem with reliability validations, adapting the mechanical parameters of PB R3 (using the procedures stated in this project) in order to fit other products different than TCUs (automobile mirrors, electric car components, etc.).

ETSEIB

Acknowledgements

First of all, I would like to thank the company Ficosa A.C. for offering me the opportunity to lead, develop and execute entirely an engineering project while working for them as an intern. It was a really challenging and enriching experience and it has taught me so many valuable things on being a better and more complete engineer.

I would like to thank both this dissertation’s tutor, Héctor Alarcón, and this disseration’s chairperson, Josep Vilaplana, for all the support provided during the project’s development. Their suggestions and guiding made the whole process clearer, easier and more enjoyable.

Special thanks to Ficosa A.C. Reliability department’s supervisor, Sergio Díaz, for his constant support during project development, especially on the hardest times. He always believed in my capabilities and put his trust on me, and thanks to his continuous assistance and attention, I was able to take the project much further than initially expected. His vast knowledge and experience on project management, and his willingness to share it and help, were crucial to defining and following the project’s approach strategy and reaching the project’s final solution.

Finally, as this project marks the end of my bachelor’s degree, I would like to take this opportunity to wholeheartedly thank my family, specially my father, who always gave me his support both during the project and through the rest of my studies. He was always a shoulder to lean on.

Bibliography

In this chapter, bibliography for the project report and appendixes is detailed, following standard ISO 690.

Bibliographic references

In this section, documents referenced in the project report or its appendixes are cited.

[1] MOUNTAIN GOAT SOFTWARE. *User Stories* [online] [Date consulted: 23rd March 2018]. Available at: <https://www.mountaingoatsoftware.com/agile/user-stories>.

[2] WIKIPEDIA COLLABORATORS. *Agile software development* [online]. Wikipedia, The Free Encyclopedia, 2018 [Date consulted: 28th May 2018]. Available at <https://en.wikipedia.org/wiki/Agile_software_development>.

[3] WIKIPEDIA COLLABORATORS. *Jira (software)* [online]. Wikipedia, The Free Encyclopedia, 2018 [Date consulted: 28th May 2018]. Available at <https://en.wikipedia.org/wiki/Jira_(software)>.

[4] WIKIBOOKS COLLABORATORS. *4.1 - Functional Analysis & Allocate Requirements* [online]. Wikibooks, 2017 [Date consulted: 1st June 2018]. Available at <https://en.wikibooks.org/wiki/Seed_Factories/Functions>.

[5] WIKIPEDIA COLLABORATORS. *Matriz de datos* [online]. Wikipedia, La enciclopedia libre, 2018 [Data consulted: 28th February 2018]. Available at <https://es.wikipedia.org/wiki/Matriz_de_datos>.

[6] HUDSON, L. *pylibdmtx* [online]. Version 0.1.7. Python Software Foundation (US), 2018 [Date consulted: 2nd March 2018]. Available at <https://pypi.org/project/pylibdmtx/>.

[7] BROWN, J. *zbar* [online]. Version 0.10. Python Software Foundation (US), 2009 [Date consulted: 2nd March 2018]. Available at <https://pypi.org/project/zbar/>.

[8] EXACTCODE. *ExactImage. A fast, modern and generic image processing library* [online]. Version 1.0.1. [Date consulted: 7th March 2018]. Available at <https://exactcode.com/opensource/exactimage/>.

[9] TONXON. *DmDecoder* [online]. Version alpha. SourceForge, 2015 [Date consulted: 7th March 2018]. Available at <https://sourceforge.net/projects/dmdecoder/>.

[10] FREYTAG, J. *Freytag DataMatrix Decoder* [online]. Version 1.1 (beta). Free Software Directory, 2006 [Date consulted: 7th March 2018]. Available at <https://directory.fsf.org/wiki/Freytag_DataMatrix_Decoder>.

[11] WXPYTHON CONTRIBUTORS. *Overview of wxPython* [online] [Date consulted: 27th March 2018]. Available at <https://www.wxpython.org/pages/overview/>.

[12] SAJIP V. *Logging HOWTO* [online]. Version 2.7.15. Python Software Foundation (US), 2018. Basic logging tutorial. [Date consulted: 6th April 2018]. Available at <https://docs.python.org/2/howto/logging.html#logging-basic-tutorial>.

[13] WIKIPEDIA COLLABORATORS. *Norm (mathematics)* [online]. Wikipedia, The Free Encyclopedia, 2018 [Date consulted: 16th April 2018]. Available at <https://en.wikipedia.org/wiki/Norm_(mathematics)>.

[14] SASTANIN. *How can I quantify difference between two images?* [online]. Stack Overflow, 2008. General idea. [Date consulted: 16th April 2018]. Available at <https://stackoverflow.com/questions/189943/how-can-i-quantify-difference-between-two-images>.

[15] ROSEBROCK, A. *How-To: 3 Ways to Compare Histograms using OpenCV and Python* [online]. Pyimagesearch, 2014 [Date consulted: 18th April 2018]. Available at <https://www.pyimagesearch.com/2014/07/14/3-ways-compare-histograms-using-opencv-python/>.

[16] WIKIPEDIA COLLABORATORS. *Structural similarity* [online]. Wikipedia, The Free Encyclopedia, 2018 [Date consulted: 17th April 2018]. Available at <https://en.wikipedia.org/wiki/Structural_similarity>.

[17] DEPC-UPC. *Gestión de Proyectos (Ing. Ind.)* [ebook]. ETSEIB-UPC, 2018. Tema 3: Diseño técnico y análisis de alternativas. [Date consulted: 20th April 2018].

[18] WIKIPEDIA COLLABORATORS. *Seven-segment display* [online]. Wikipedia, The Free Encyclopedia, 2018 [Date consulted: 9th May 2018]. Available at <https://en.wikipedia.org/wiki/Seven-segment_display>.

[19] ROSEBROCK, A. *Recognizing digits with OpenCV and Python* [online]. Pyimagesearch, 2017 [Date consulted: 9th April 2018]. Available at <https://www.pyimagesearch.com/2017/02/13/recognizing-digits-with-opencv-and-python/>.

[20] WIKIPEDIA COLLABORATORS. *Distortion (optics)* [online]. Wikipedia, The Free Encyclopedia, 2018 [Date consulted: 23rd May 2018]. Available at <https://en.wikipedia.org/wiki/Distortion_(optics)>.

[21] WIKIPEDIA COLLABORATORS. *ISO metric screw thread* [online]. Wikipedia, The Free Encyclopedia, 2018 [Date consulted: 23rd May 2018]. Available at <https://en.wikipedia.org/wiki/ISO_metric_screw_thread>.

[22] WXPYTHON CONTRIBUTORS. *wx.BoxSizer* [online] [Date consulted: 12th April 2018]. Available at <https://wxpython.org/Phoenix/docs/html/wx.BoxSizer.html>.

[23] ROSEBROCK, A. *Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi* [online]. Pyimagesearch, 2017 [Date consulted: 22nd April 2018]. Available at <https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/>.

[24] OPENCV DEV TEAM. *Histogram Comparison* [online]. OpenCV 2.4.13.4 documentation, 2017. Theory. [Date consulted: 18th April 2018]. Available at <https://docs.opencv.org/2.4.13.4/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html>.

[25] NATIONAL INSTRUMENTS. *Calculating Camera Sensor Resolution and Lens Focal Length* [online] [Date consulted: 29th May 2018]. Available at <http://www.ni.com/product-documentation/54616/en/>.

[26] WIKIPEDIA COLLABORATORS. *Image sensor format* [online]. Wikipedia, The Free Encyclopedia, 2018. Table of sensor formats and sizes. [Date consulted: 29th May 2018]. Available at <https://en.wikipedia.org/wiki/Image_sensor_format#Table_of_sensor_formats_and_sizes>.

[27] INVESTOPEDIA. *Discounted Cash Flow (DCF)* [online] [Date consulted: 12th June 2018]. Available at <https://www.investopedia.com/terms/d/dcf.asp>.

[28] INVESTOPEDIA. *Internal Rate of Return - IRR* [online] [Date consulted: 12th June 2018]. Available at <https://www.investopedia.com/terms/i/irr.asp>.

Consulted bibliography

In this section, relevant documents, consulted during the project but not referenced in the project report nor in its appendixes, are cited.

KENLON, S. *Qt versus Wx: How do two of the most popular Python frameworks compare?* [online]. Opensource.com, 2017 [Date consulted: 22nd March 2018]. Available at <https://opensource.com/article/17/4/pyqt-versus-wxpython>.

PYTHON DOCS COLLABORATORS. *10. File and Directory Access* [online]. Version 2.7.15. Python Software Foundation (US), 2018 [Date consulted: 5th April 2018]. Available at <https://docs.python.org/2/library/filesys.html>.

PYTHON DOCS COLLABORATORS *12.4. zipfile — Work with ZIP archives* [online]. Version 2.7.15. Python Software Foundation (US), 2018 [Date consulted: 28th March 2018]. Available at <https://docs.python.org/2/library/zipfile.html>.

ZYGIMANTAS. *How to delete a file or folder* [online]. Stack Overflow, 2011 [Date consulted: 5th April 2018]. Available at <https://stackoverflow.com/questions/6996603/how-to-delete-a-file-or-folder>.

ROSEBROCK, A. *Basic Image Manipulations in Python and OpenCV: Resizing (scaling), Rotating, and Cropping* [online]. Pyimagesearch, 2014 [Date consulted: 14th May 2018]. Available at <https://www.pyimagesearch.com/2014/01/20/basic-image-manipulations-in-python-and-opencv-resizing-scaling-rotating-and-cropping>.

Complementary bibliography

In this section, documents related to this project's described solution are cited. They can be of great value for readers that want an in-depth explanation of some concepts presented and/or used during this project's solution description, but whose technical details are not necessary for the general reader to understand how the designed solution works and would break this project report's writing flow.

DUNN, R.; RAPPIN, N. *wxPython in action*. Greenwich: Manning Publications Co., 2016. ISBN 1-932394-62-1.

PETERS, J. F. *Foundations of Computer Vision: Computational Geometry, Visual Image Structures and Object Shape Detection*. Winnipeg: Springer International Publishing, 2017. Intelligent Systems Reference Library, Vol. 124. ISBN 9783319524818.