Bachelor's Thesis

Information and Communications Technology

2022

Otto Heldt

Testing of LED panels using computer vision

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2022 | 36 pages

Otto Heldt

Testing of LED panels using computer vision

This Bachelor’s thesis covers some of the technologies used in a computer vision system and the process of building a computer vision system from the ground up that can be used to test the functionality of a LED panel.

The goal of this thesis was to find a good method of building a computer vision system for LED panel testing, and to see if it is possible to have a reliable method of testing LED panel functionality with computer vision. A good computer vision system should be easy to use, and it should give out reliable results and be adaptable for different kinds of testing scenarios.

This thesis also aimed to have a working computer vision system setup with results gathered from testing that show a functioning computer vision script detecting LEDs on a LED panel and working according to the requirements set during the planning process. These results were collected by implementing real-life testing with a 64x32 LED panel and a computer vision script that detected how many LEDs were functioning as intended.

This goal was achieved, and the results gathered from the testing process show that LED panels can be tested in a reliable way using computer vision.

Keywords:

Computer vision, LED panel, automation, OpenCV

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2022 | 36 sivua

Otto Heldt

LED-paneelien testaaminen konenäön avulla

Tässä opinnäytetyössä tarkastellaan joitain niistä teknologioista, joita tarvitaan konenäköjärjestelmän rakentamiseen, sekä konenäköjärjestelmän rakentamista alusta loppuun LED-paneelien toimivuuden testaamiseen.

Opinnäytetyön tarkoituksena oli löytää hyvä tapa konenäköjärjestelmän rakentamiseen LED-paneelien testaukseen, ja kokeilla onko luotettavaa tapaa testata LED-paneelin toimivuutta konenäön avulla. Hyvän konenäköjärjestelmän tulee olla helppo ja intuitiivinen käyttää, sen tulee myös antaa luotettavia tuloksia ja olla mukautuvainen moniin erilaisiin testaustilanteisiin.

Opinnäytetyössä oli myös tarkoitus tuottaa toimiva konenäköjärjestelmä, ja osoittaa sen toimivuus. Konenäköjärjestelmän tulee toimia asetettujen vaatimusten mukaisesti ja havaita ledit paneelista, jos ne ovat päällä. Nämä tavoitteet saavutettiin ja tulokset kerättiin testausprosessin aikana testeillä, joissa konenäköohjelma sai kuvaa 32x64 LED-paneelista.

Testauksen aikana todettiin, että konenäköohjelma toimi asetettujen vaatimusten mukaisesti.

Asiasanat:

Konenäkö, automaatio, OpenCV, LED-paneeli

Contents

Figures

Pictures

Tables

List of abbreviations

LED	Light-emitting diode [1]
RGB	Red, green, and blue primary colors [2]

1 Introduction

Automating manufacturing processes is an interesting challenge faced in the modern world. With properly automated production lines companies are able to significantly speed up the production process. When humans are being aided by computers and robots, the outcome is often far more accurate thus faulty products are less likely to appear.

The manufacturing of LED panels is one area where automation of the production lines can help remove some human error. Using computer vision is one of the ways manufacturing can be automated. It can be beneficial to utilize it in quality control when producing LED panels considering that humans cannot check for quality issues on LED panels as accurately and as fast as computers can.

For example, if there is a LED panel with 64x64 LEDs, this means there is a total of 4096 LEDs. A computer vision program could run through a combination of different lighting patterns on this panel in seconds and report accurately if the panel is functioning properly. For a human this task would require a fair bit of more time and the possibility of missing a non-functioning LED would be greater than for a well-tested computer vision program.

The goal of this thesis is to:

(1) Plan a computer vision system that includes:

- a method for capturing video
- a computing platform where captured data is processed
- a method for processing the data
- a LED panel for testing
- a microprocessor that drives the LED panel

(2) Build a computer vision system that:

- accurately detects faults in LED panels
- could be adapted into a production line in the future

- is simple to use

The structure of this thesis is as follows: Chapter two is an overview into the methods and technologies used in this project. Chapter three covers the requirements for this project, followed by Chapter four which is dedicated to the architecture of the project, i.e., it discusses the selected technologies and why they were selected. Chapter five consists of the implementation of this computer vision system and the implementation is compared to the requirements, then chapter six reports on the testing process where the computer vision system is being tested against the requirements to determine if they were met. Finally in Chapter seven the results are reflected upon, and possible future work and improvements are discussed.

2 Methods and technologies

2.1 Microcontroller platform

Microcontrollers are small scale, inexpensive computers well suited for embedded applications [3]. Typically, a microcontroller consists of a processor, memory, and input/output peripherals all combined on a single small chip [4]. These devices have significantly less computing power and memory than an average personal computer would have. Microcontrollers can be found on a wide range of different devices, all the way from consumer electronics to the automotive industry.

There are many different microcontroller platforms to choose from and picking the right one is an integral part of any project. It may determine how the whole project is going to be constructed and many important functions can be built around the microcontroller and its capabilities. Some of the most widely used microcontrollers are:

- ATmega328 [5]
- PIC16F877A [6]
- MSP430 [7]
- ESP32 [8]
- ATmega32U4 [9]
- STM32 (ARM cortex-based) [10]

Not all microcontroller platforms are built the same and depending on the requirements, options should be narrowed down so that the best possible microcontroller platform can be chosen. If internet connection is required one might consider choosing ESP32 or ESP8266 since they both have Wi-Fi capabilities. On the other hand, if raw processing power is needed something like the STM32F7 [11] might be a good choice. Ease of use is also crucial and should be examined especially if time limitations apply. For prototyping purposes Arduino boards are an excellent choice since there are a lot of

resources available for them and developing on them is simple. Arduino boards use ATmega microcontrollers. However, Arduino boards are usually not used for final production since they are larger in size and cost than other options.

2.2 LED technology

In lighting technology, light emitting diodes (LEDs) are the preferred light source. For the past few decades, LED technology has significantly grown [12]. The fast growth of solid-state lighting based on high-brightness visible LEDs is fueled by high efficiency, dependability, robust construction, low power consumption, and longevity.

Incandescence or gas discharge are used in traditional light sources like filament light bulbs and fluorescent lamps. Due of the high temperatures and substantial Stokes shift characteristics (Figure 1) [13], these two processes are accompanied by considerable energy losses. Semiconductors, on the other hand, enable for more efficient light production. Semiconductor-based LEDs have the capacity to convert power to light with a near-one-to-one efficiency unlike traditional filament light bulbs.

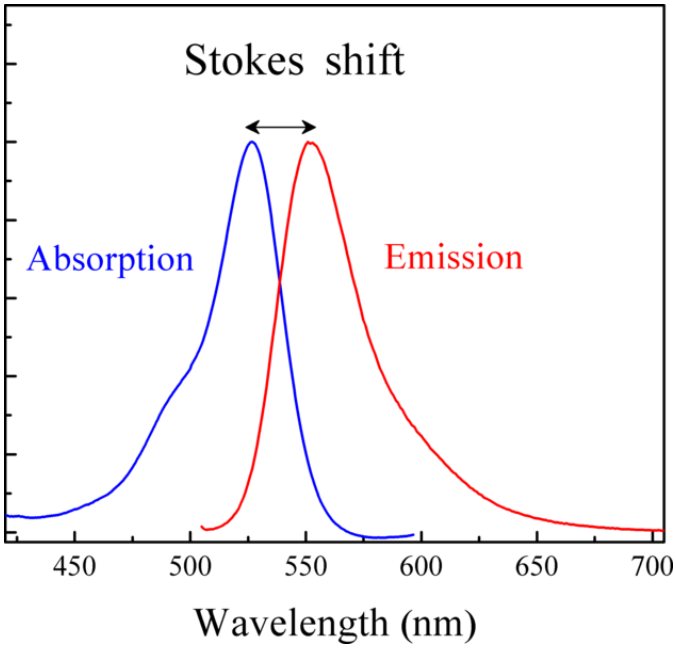


Figure 1. Example of Stokes shift.

The LED panel used in this project is an RGB LED panel. This means that every RGB LED in the panel has three LEDs inside with the three primary colors of red, green, and blue, that can be used to produce a multitude of colors. These colors can be changed with pulse width modulation [14] by controlling the intensity of each of the LEDs inside the RGB LED. This however means that the LED panel flickers at the duty cycle [15] of the PWM. The flickering of the LEDs is not noticeable by humans, but a camera might be able to see it, causing the computer vision system to not detect the LEDs properly. However, the LED panels this computer vision system aims to help test in a production line are not RGB LED panels. Therefore, even if the flickering of the LED panel seems to be an issue during testing, it can be assumed that it will not affect the performance during the testing of the actual products.

2.3 Computer vision

Research on computer vision started back in the 1960s when Larry Roberts considered the prospect of extracting three-dimensional information from two-dimensional perspective views of blocks in his Ph.D. thesis “Machine Perception of Three-Dimensional Solids" [16]. Many Artificial Intelligence researchers at MIT and elsewhere researched computer vision in the setting of the blocks world because of this study. Later, researchers found that photographs from the real world were required. As a result, substantial research was required in "low-level" vision tasks including edge detection and segmentation.

The human view of the world is three dimensional [17]. It is easy for humans to name and count all the people they know when looking at group photograph. Humans can even have an educated guess about the feelings of others when seeing their faces. Over the years research done in the field of computer vision has made strides forward and now it is possible to find a person’s name by using hair detection and recognition and combining it with face, and clothing detection. Even with all these advancements computers are not yet capable of understanding visual data at the same level as humans. Since vision is an

inverse problem, meaning the goal is to find unknowns with inadequate data to completely identify an answer, utilizing probabilistic and physics-based models to come to a solution is a must.

In the modern world where the quality standards of production are high, computer vision systems offer a way for automated quality control. However, the standard for these systems grows higher as they get implemented into production facilities where extreme accuracy is needed, such as when producing measurement systems or machine parts.

Despite the challenges computer vision has it is extensively used for a wide range of applications in the real world, which include:

- **Automotive safety:** detecting obstacles, pedestrians, and vehicles on the road.
- **Retail:** recognition of objects for automated checkouts.
- **Motion capture:** this process captures the movements of a subject in real life and can use multiple cameras to track markers placed on the body [18]. This material can then be combined with computer generated imagery (CGI) for fluent and realistic animations.
- **Surveillance:** motorway traffic analysis, intruder monitoring.
- **Fingerprint recognition:** forensic applications as well as for automatic access authentication situations.
- **Quality inspection for production:** automated inspecting of products for quality issues.

2.4 OpenCV

There are multiple tools to choose from when it comes to building computer vision solutions. The most popular choice as of writing is called OpenCV (Open-Source Computer Vision Library). OpenCV first started in 1998 as a research

project at Intel and it has been publicly available under the BSD open-source license from 2000 [19]. It is a machine learning and computer vision software library that contains over 2500 different algorithms, that include both classic and modern machine learning and computer vision algorithms [20].

Developing programs and scripts that utilize OpenCV can be written in a multitude of different programming languages and currently OpenCV supports C++, Java, Python and MATLAB. It also supports a host of different operating systems that include Windows, Linux, Mac OS, and Android.

OpenCV is free to use, even for commercial applications and does not require for projects that use it to be open source. With over 9 million users, developers have a strong support network and that is a big advantage of OpenCV compared to other computer vision libraries. OpenCV also receives funding from large corporations which allows for it to continue developing and evolving at a fast pace.

3 Requirements

Every project needs a set of requirements that act as a guideline for the project. This chapter goes over those requirements set for the computer vision system using the Moscow method [21]. These requirements can be seen listed in Table 1 and include the functionality needed for the computer vision system to be a success, as well as some functionality that could be added later.

Table 1. Listing of requirements using the Moscow method.

Requirements	Description	Priority
1	Determining the LED state using computer vision	Must
2	Warn the user if an LED is off	Must
2.1	Warn the user if LED intensity is incorrect	Could
2.2	Take a picture of the LED panel that can be viewed in case faulty LEDs are detected	Must
3	Produce consistent and accurate results	Must
4	Test enclosure with controlled lighting	Could
5	Visually show the locations of LEDs that are turned on	Should
6	Detect multiple LED colors	Will not

In the following sections some of the requirements are explained in more detail.

3.1 Determining LED state

When it comes to the requirements of this project, arguably the most important function this computer vision system needs to do is to determine the state of the LEDs on the LED matrix. This requires a way for the computer vision program to see the LED matrix and could be done with an external camera system. Detecting the LEDs from the video feed outputted by the camera system is the next step in determining the LED state. This step will require programming or a camera system that works with a computer vision software e.g., Keyence or Teledyne FLIR. Programming the script in-house allows for a more specialized and affordable solution then going for a ready built multi-functional computer vision system and is the better solution for this project. When writing the computer vision program, there are multiple different ways to do object detection. Selecting the most reliable way is crucial and testing is likely required for finding the best solution. Determining the LED state is presumably the most demanding aspect of this project because of the need for it to be as accurate as possible to avoid any false positives or negatives.

3.2 Warning the user if an LED is off

The user of this computer vision system should have some way of knowing when the LED matrix is faulty. The program could give out an error when it detects a faulty LED and possibly show the location of this faulty LED, if that is something that can be done within the time frame of this project. The most straight forward approach would be to have a predetermined integer set to a variable which would be the size of the number of LEDs that should be turned on. This integer could then be compared to the number of LEDs the computer vision program detects are in the on state and if they match the LED panel is working as intended. If the program determines they do not match, a warning would be given out to the user and the number of LEDs that are faulty along, with a picture taken of the panel on the moment of testing would be displayed. This would allow the user to easily determine if the LED panel is faulty.

3.3 Consistency with the results

Consistency is key, the computer vision system needs to be reliable for it to be used in a production setting. If false negatives are given by the system, it would equate to faulty products leaving the production facility which is not ideal. Reducing outside variables to a minimum would guarantee consistent results if adequate testing has been done previously. These variables could be related to the ambient lighting of the production facility or the position of the LED panel on the testing line. Building a testing enclosure or a testing space with controlled lighting and a marked area for the LED panel ensures that the effects of these variables can be mostly negated allowing for more accurate and predictable results, making the process consistent.

4 Architecture

The architecture needs to be in line with the requirements set for this project. The requirements act as a guideline when choosing the critical parts for this project. These following subchapters cover the chosen components and the process of why those components were selected for this project and how they are used to accomplish the wanted results set by the requirements.

4.1 Computing platform

The computing platform is what runs the computer vision program used for LED panel testing, this is a combination of an operating system together with hardware that the operating system is running on. With that in mind there are a few possible options of operating systems to select from, most notably Microsoft Windows platform and Linux distributions such as Ubuntu [22] or Manjaro [23]. Using a Raspberry Pi [24] running Raspbian [25] , a Linux distro made for the Pi would be an affordable solution, offering an accessible hardware and software combination. Only the performance capabilities of the Raspberry Pi platform are not best suited for more computationally demanding functions such as real time computer vision tasks with hundreds of detected objects. Utilizing a device with more computing power is necessary for fast performance and a modern laptop with a multi-core CPU serves this purpose. Table 2 shows the differences in device specifications between a modern laptop and a Raspberry Pi device.

Table 2. Comparison of device specifications between Lenovo P14s and Raspberry Pi 4.

Device	Lenovo P14s	Raspberry Pi 4
CPU	i7-1165G7	Quadcore Cortex-A72
CPU Clock Speed	2.8 GHz base, 4.8GHz turbo	1.5GHz
Number of CPU cores	4	4
Number of threads	8	4
RAM	16GB	Maximum of 8GB
Operating system	Windows 10	Raspbian (Linux)
CPU Benchmark score [26] (higher is better)	10596	834

This project will use a Lenovo P14s with an Intel i7-1165G7 CPU as it offers enough computing power for fast performance, which is important in production to avoid any delays that would slow down the testing process, hence slowing down production. It also allows for internet connectivity via ethernet, this can be important in a production facility for data transferring with high speeds and secure connection.

4.2 Sensor subsystem

The LED panel must be controlled with a microcontroller so that different LED lighting patterns can be tested on it to ensure the functionality of every LED. Going with a microcontroller board is a good option for this project, due to them allowing a fast and simple way for controlling the LED panel. Arduino, which is an open-source software and hardware company, designs and produces boards that are well suited for this application.

Arduino boards can be programmed using the Arduino IDE, which is free to use and allows for a simple and easy way to write and upload code to the board.

Traditionally a microcontroller needs to be connected to an external programmer through its programming pins for code to be uploaded to it, but with Arduino this is done using a USB connection from a PC running the Arduino IDE and an Arduino board, simplifying the process significantly.

Since this project uses a 64x32 LED panel, a microcontroller with enough flash program memory to drive it is needed. Arduino Uno [27], which is the most used board from Arduino uses an ATmega328P [5] microcontroller. This microcontroller has 32kB of flash memory. This is not sufficient for this project, because of the amount of flash memory required to store all the programmed LED data for a 64x32 LED panel.

Arduino Mega [28] is another board produced by Arduino. It has an ATmega1280 microcontroller which offers 256kB of flash memory. This allows for the controlling of the whole 64x32 LED panel. This alongside with the simplicity of using an Arduino board makes the Arduino Mega a good choice for this project.

4.3 Programming language

There are multiple different programming languages that can be used for computer vision applications. Considering this project relies on OpenCV as its computer vision library the four main options for a programming language are C++, Python, Java, and MATLAB due to them being the only supported programming languages by OpenCV as of writing.

MATLAB is often used for academic purposes when testing new ideas and prototyping. MATLAB however is extremely slow in executing code and its syntax is quite different from general-purpose languages, such as Python and C++. MATLAB coder does offer the capability to convert MATLAB code to C and C++ negating the issue with its speed. It is also not a great option for production as it is not designed to be used for such purposes. MATLAB also requires a license payment and yearly fees making it not the most desirable option for this computer vision system.

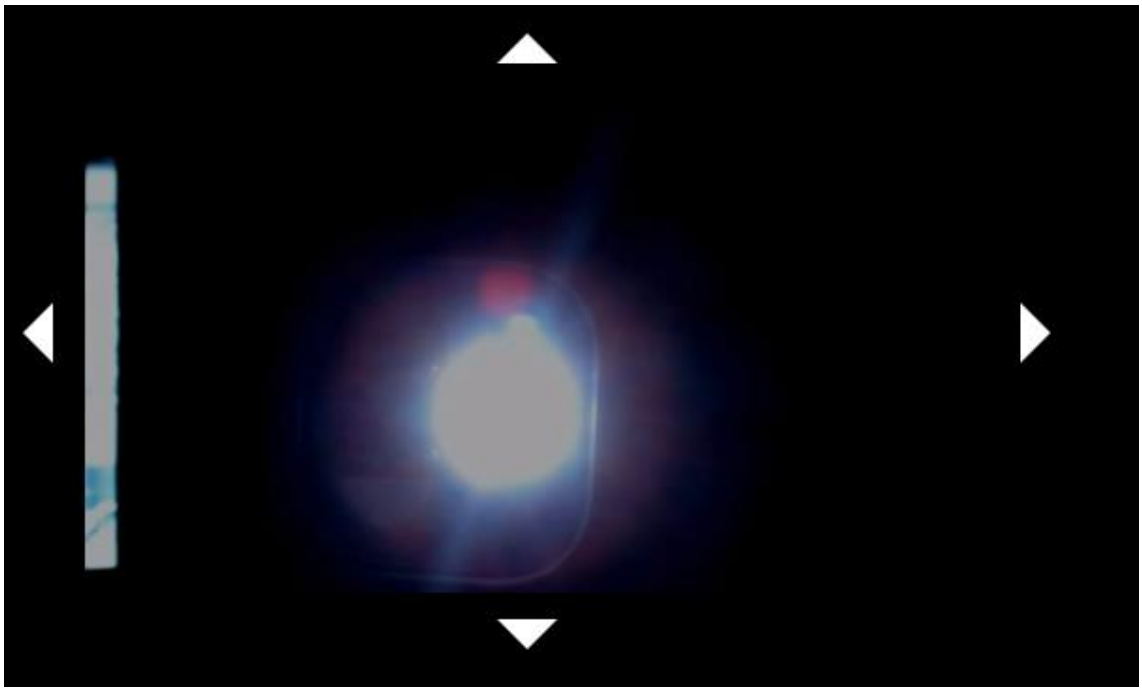
C++ and Python are both great options for a computer vision system. Being a compiled language C++ offers faster performance when compared to Python, which is an interpreted language [29]. Programming with C++ is often said to be more difficult when comparing it with Python thanks to Pythons easy-to-read syntax making Python simpler to develop programs with. Alongside with Pythons simple syntax, Python is more widely used for machine learning applications making it the better option for computer vision purposes out of the two, therefore this project will use Python as its programming language for the computer vision program.

4.4 Camera system

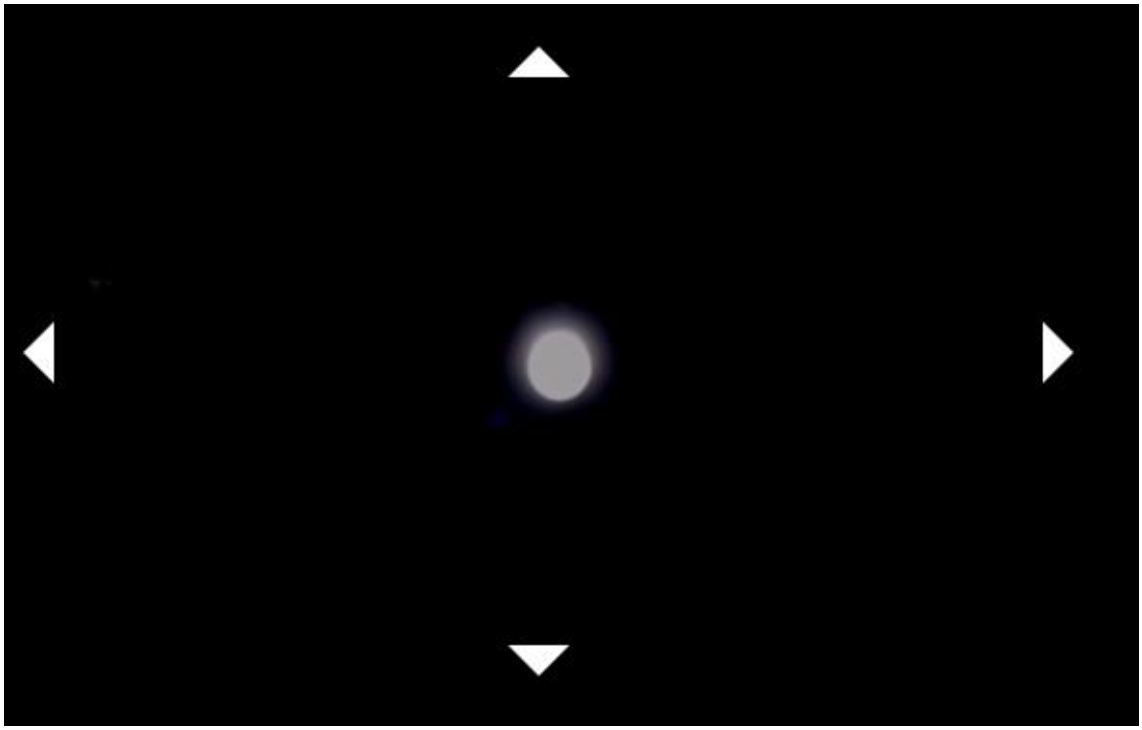
There are a wide variety of different camera systems available for computer vision use, but it is also possible to use cameras that are not specifically designed with computer vision in mind. Camera systems made for computer vision offer advanced industry standard sensors and on-camera pre-processing with up-to 4K 120FPS video. Using a camera specialized for computer vision means that to get the most out of that hardware, the use of software's that are built for those camera systems is recommended. This would add another layer of complexity to the computer vision system and an added learning phase of the software. Cameras specialized for computer vision are also expensive and offer extra functionality that is redundant for this computer vision system, making them a suboptimal choice.

Looking at more widely available easy to access camera offerings, that are inexpensive as well as easy to use and setup, steers the camera selection towards webcams. Webcams are portable, relatively inexpensive when compared to other cameras such as DSLRs or point and shoot cameras, and they are built specifically to be used concurrently with a computer. They are also lightweight and often offer adjustable mounting which allows for easy adjustment of the picture frame. Easy frame adjustment is helpful when aligning the LED panel for testing and makes the testing procedure faster.

Webcams often come with their own software that makes it easy to change the brightness and exposure of the video feed that the webcam is capturing. These brightness and exposure settings can be a useful tool when fine-tuning the LED panel testing environment as they affect the way that the light produced by the LEDs shows on the captured video making it possible to have the LED light be the only light that is properly visible on the video feed. Pictures 1 and 2 show the difference between exposure that is set to 50% and 0%.



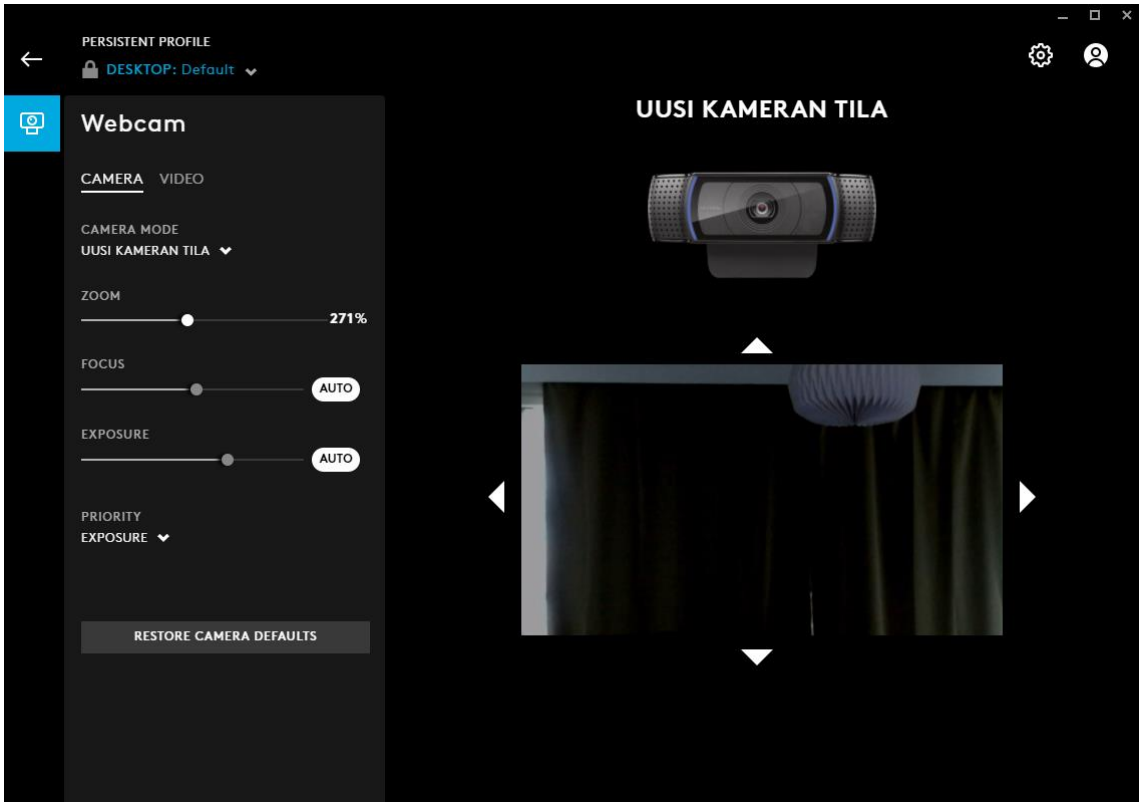
Picture 1. Web cam video feed of a single LED with exposure set to 50% and brightness set to 0%.



Picture 2. Web cam video feed of a single LED with exposure set to 0% and brightness set to 0%.

As the pictures show, changing the exposure makes the single LED show up on the video feed with a clearer outline. This allows the computer vision program to distinguish the line between where one LED ends and another starts, making the detection process more reliable. With the exposure set to 0%, the effects of unwanted light sources such as room lights is also mitigated since they are less likely to appear on the video feed due to them not being as bright as the LEDs. This can also be seen from pictures 1 and 2 as the light that was visible on picture 1 is no longer there in picture 2, and the only visible bright spot is the LED light.

There is a wide selection of different webcams currently on the market. Logitech has great options to choose from due to the availability of their products and the computer software (Logitech G Hub), that allows for the fine tuning of the video signal captured by the webcam. Picture 3 shows an overlook of the Logitech G Hub software and some of the options that are available on it.



Picture 3. Logitech G Hub software.

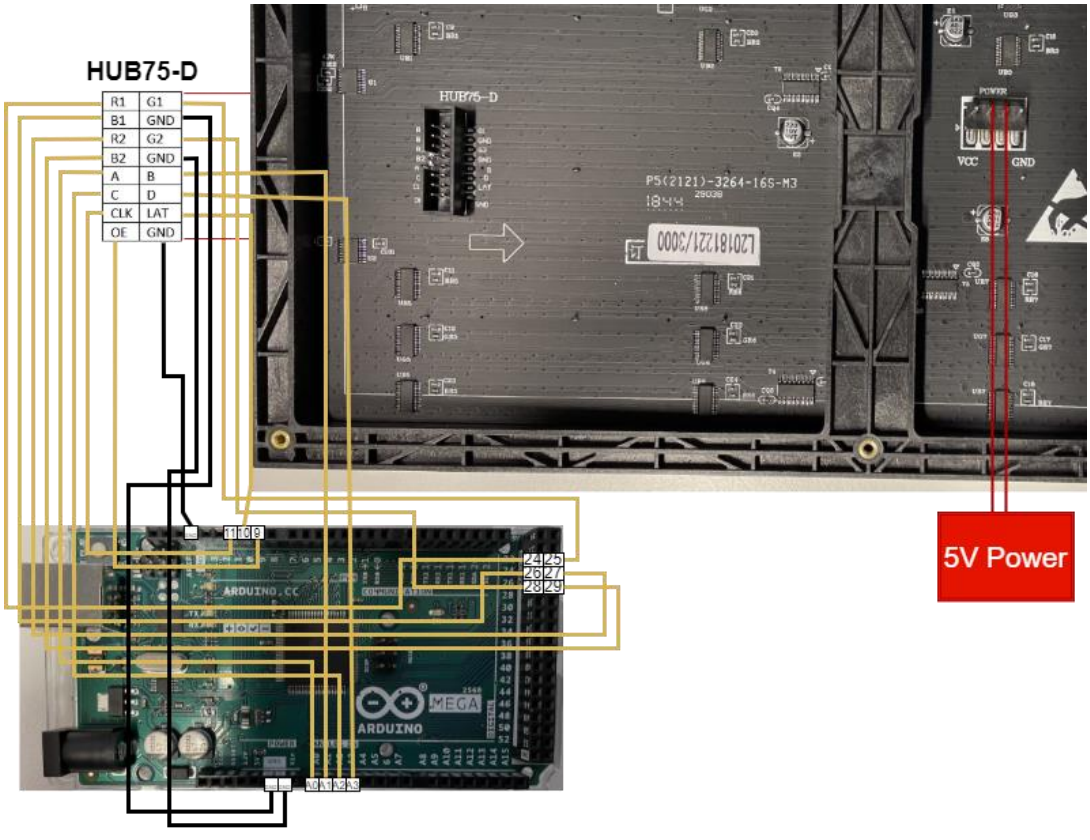
The webcam used for this project is the Logitech C920 as it offers a good balance between video quality, adjustable focus, brightness, and exposure, and it is affordable, making it a great choice for testing the functionality of this project before committing to a more permanent solution set up on a production line.

5 Implementation

This implementation chapter covers the process of utilizing the parts selected in Chapter 4. The implementation must match with the architecture of the project so that the desired results can be achieved. The following two subchapters explain in more detail how the selected components were used in the project, firstly going over the hardware section and then the software section of the project.

5.1 Hardware

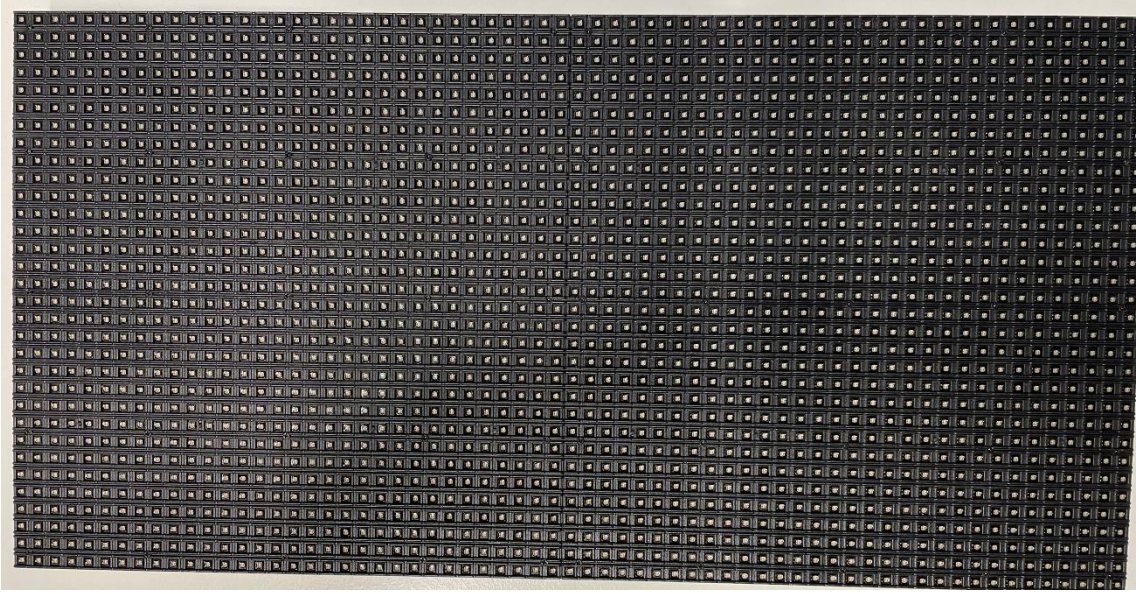
An Arduino Mega was chosen to control the LED panel. This allowed for a simple and easy way to display a multitude of differing lighting patterns on the LED panel. The connections between the Arduino Mega and the LED panel can be seen in picture 4.



Picture 4. Connection between an Arduino Mega and a 64x32 LED panel.

These connections were made by soldering the wires between the corresponding pins on the Arduino Mega and the LED panel to make sure no wires would come loose, stopping the panel from working properly. For the 5V power an external power supply was used. This also allowed for current monitoring straight from the power supply during the testing phase. Power for the Arduino board was delivered through a USB connection to a Lenovo P14s laptop. This could have also been done with an external power source, but since the code controlling the lighting patterns on the LED panel is uploaded using the USB connection, this way of power delivery was optimal.

The LED panel used in the project has a resolution of 64x32 dots, meaning it has 64 LEDs on the x-axis and 32 LEDs on the y-axis. Its dimensions are 320 x 160 x 15mm. The chosen panel has a pixel pitch of 5mm, which is like the panels in production, in which this project would be utilized for. This panel was chosen due to its availability and size. There is no reason why this project would not work with other LED panels with a similar pixel pitch, assuming they can be controlled with an Arduino board. Picture 5 shows what the LED panel looks like when no LEDs are being powered.



Picture 5. 64x32 LED panel with all LEDs off.

5.2 Software

Before focusing on the computer vision aspect of this project, a method of controlling the LEDs on the panel was needed. Arduino IDE was used to write and upload code to the Arduino Mega board and that allowed for the controlling of the LEDs in any manner that was needed. Three additional libraries were installed for the Arduino IDE that provided needed functionality that allowed for the controlling of the LED panel. These libraries were:

- Adafruit GFX
- RGB Matrix panel
- Adafruit BusIO

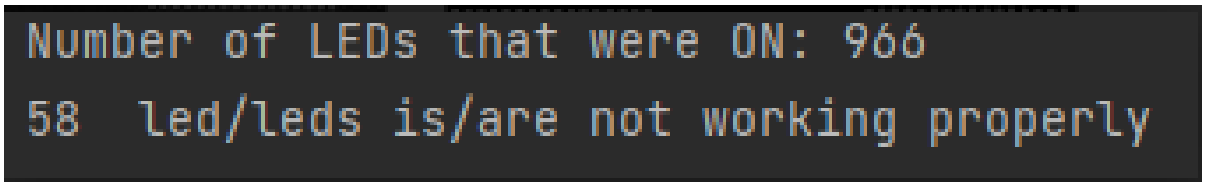
These libraries can be found from the library manager of the Arduino IDE and are free to use for anyone. After the code was ready, it was uploaded to the Arduino Mega by first selecting the right board from Arduino IDEs boards manager and then simply uploading the code to the board by clicking the upload button.

Python code was written to do the computer vision tasks. The IDE that was used is called PyCharm [30]. It was selected because of it being purposefully build for Python programming. Python version used in this project was Python 3.8 and it was selected due to it being the latest release of Python when the project started. Migrating the project to a newer Python interpreter should not cause any issues, but older Python version might not be compatible with the version 4.5.1.48 of OpenCV that was used.

OpenCV is not a package that comes with Python and to use it, it needs to be installed using pip [31]. After installing, all the functionality of OpenCV is available to use with Python, but it is still necessary to import the OpenCV package to the Python project where it is needed. OpenCV offers functionality to change the color space of the video feed it receives [32]. This functionality was first used to change the BGR (blue, green, red) color spectrum of the video feed to grayscale to simplify the process of detecting LEDs. After the video

signal was converted to grayscale a threshold function [33] was used to produce a video signal that had all the pixel values under the threshold value turned to black. This means that by finding the right threshold range, the only visible spots in the picture would be the LEDs as they were the brightest objects in front of the webcam and everything else would be filtered out. Since it was determined that it would be beneficial to indicate the locations of the LEDs that were turned on, connected components with stats [34] function was used to find all the points from the filtered boolean video signal that were similar in size. This function provided centroids with x-axis and y-axis values that correspond to the location of the bright spots of the boolean video signal. This means that by giving those values to a function called circle [35] that takes in values for x-axis and y-axis it is possible to draw a circle around the location of every LED that the script determines are on.

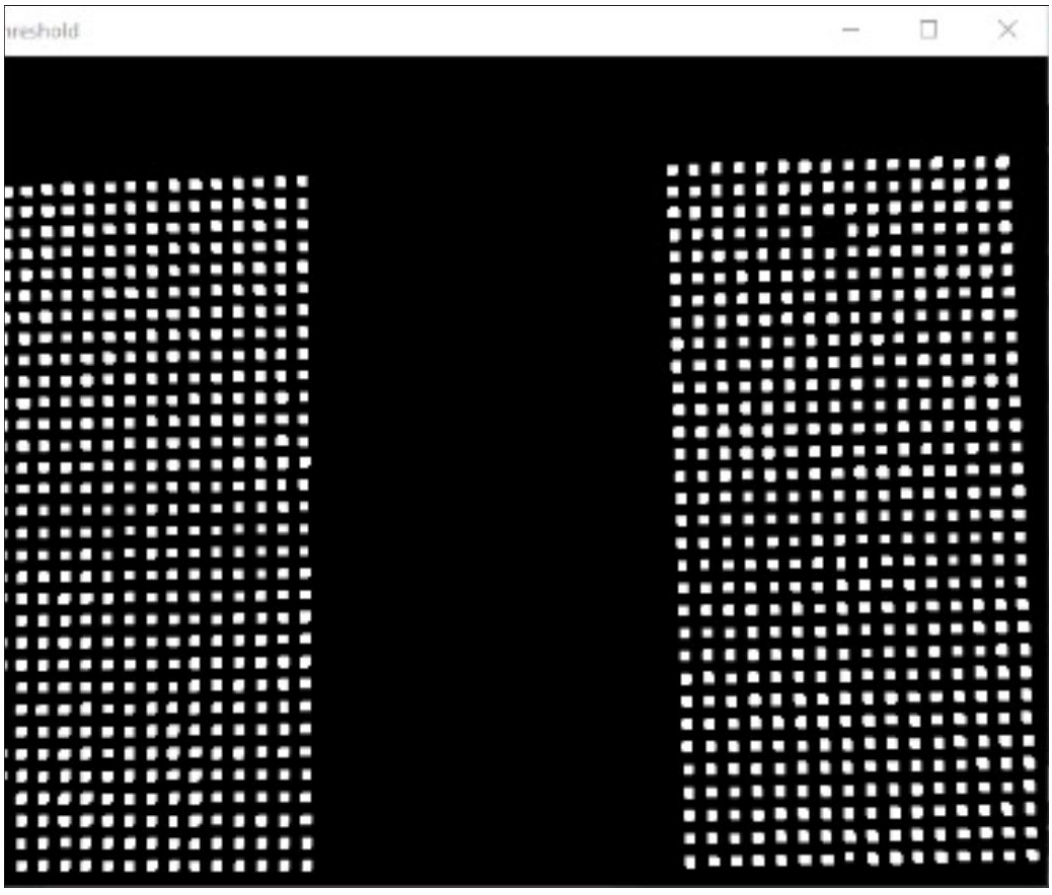
To test the panel, a user needs to input the number of the panel that is tested. There is no real functionality for it currently, since there was only one size of LED panel that was tested, but for future use it was important to have. Now the panel number given only provides the number of LEDs that should be on in one of the lighting tests patterns. After the user has given the panel number the script starts the video feed and starts detecting LEDs. When the user is ready to test the panel, they press q to stop the test and save an image of the panel that shows the locations of the LEDs marked with a green circle. It also provides them details of how many LEDs it detected were on and if there were any problems with the panel. An example of this can be seen in Picture 6 below.



Picture 6. Output of the computer vision program with faulty LEDs detected.

6 Testing

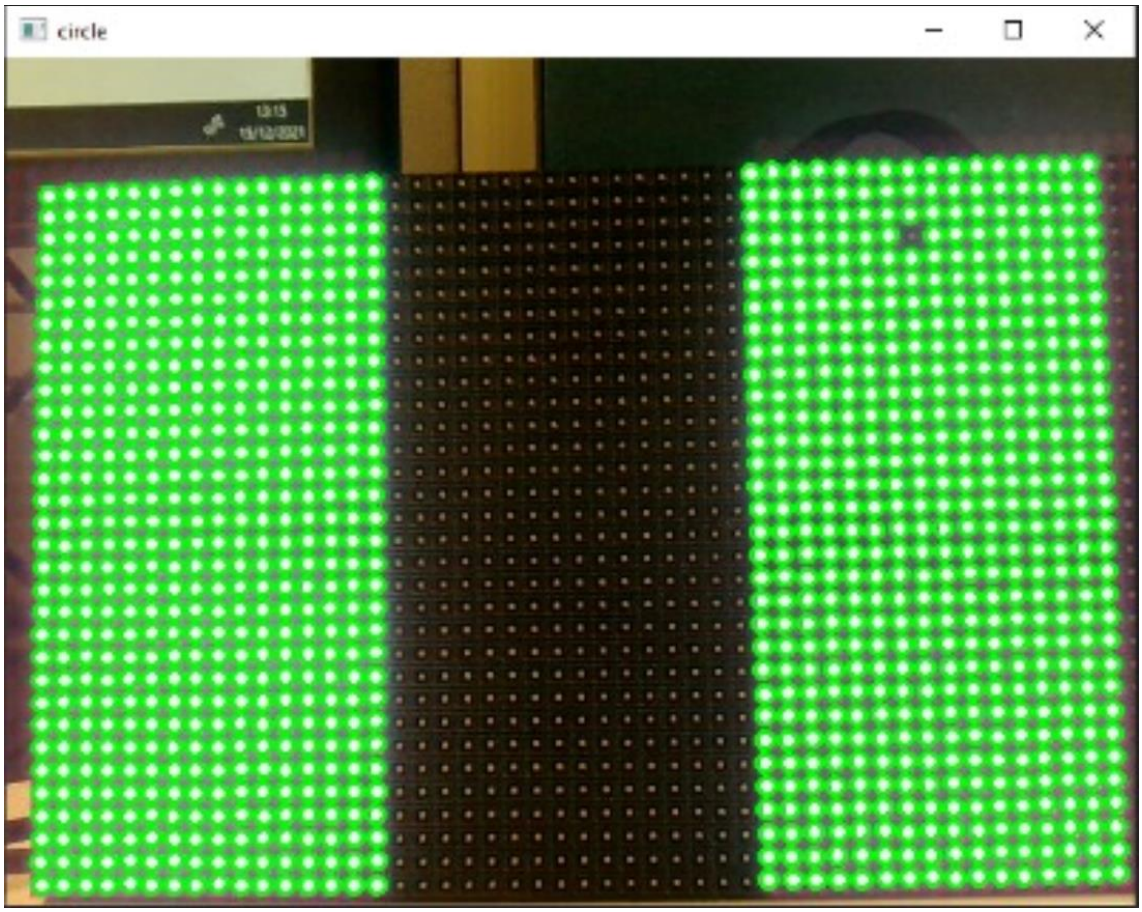
The testing process of the computer vision system started after the hardware and software parts of the system were built. A simple testing set up was built on a table where the webcam was pointed at the LED panel and the test was ran multiple times with different threshold values to determine the best range. During testing the optimal threshold was found to be 245 for the lower limit and 255 for the upper limit, which is also the maximum threshold value. Picture 7 shows the boolean video signal which this threshold produces when pointed at the LED panel with a lighting pattern that emulates half of the LEDs being turned off and a single LED being faulty.



Picture 7. Boolean video signal feed with a single faulty LED.

To better show the location of the LEDs on the LED board circles were drawn on top of the original video feed. This was tested to make sure that the circles

were drawn correctly. First there was a problem with a circle being drawn to an initial location on the video feed where no actual LEDs were located, but this was fixed by changing the code so that the drawing of circles did not start from 0, but instead 1, skipping the first initial drawing of a circle. Picture 8 shows how the circles are drawn on the same test that picture 7 is depicting, but now with the original video feed instead of the filtered boolean one.



Picture 8. Drawing of circles on top of the detected LEDs.

No enclosure was built for the panel because the lighting at the testing area was constant and during testing the LED detection was functioning as intended so there was no need to build a dedicated testing enclosure. During the testing process the LED panel became faulty. This was however a blessing in disguise as it allowed for the testing of an LED panel with LEDs that were still on but did not produce the desired amount of light making them less bright than the properly functioning LEDs. It was found that the program handled this quite well

and did not detect the dimmer LEDs to be functional. Picture 9 shows how these LEDs were not detected as they do not have circles drawn around them.

Picture 9. LED panel with faulty LEDs not being detected as functional.

The testing process proves that the computer vision system works as intended and all the requirements set for the project were met. Even some functionality that was not a must have works as the program does not detect dimmer LEDs to be functional, giving out a warning to the user of faulty LEDs when they are not producing enough light.

7 Conclusion and future development

This final chapter covers the results of this thesis and goes into more detail about the future of this project and possible improvements that could be made to the computer vision system in the future and how it could be improved currently.

7.1 Results

The goal of this thesis was to have a functioning computer vision system built that could be used for detecting faults in a LED panel. The computer vision system built for this thesis can now detect faults in LED panels with reliable results and the functionalities that were determined to be a must using the MoSCoW method were achieved. The computer vision system warns the user when it detects that the LED panel tested is not functioning properly and visually shows the user the locations of the LEDs that are on by drawing indicating circles around them. It also provides a picture of the LED panel taken during testing, providing an easy method for finding the faulty LEDs without the need for constant monitoring. This thesis also covers the technologies used for this computer vision system as well as other options for building a computer vision system and documents the process of building it.

7.2 Possible improvements

Even though the functionality that was required was achieved, there are still improvements that can be made to the computer vision system. The most important improvement that could be implemented would be to have the faulty LEDs be circled with an indicator as well. This would allow for an even faster testing process as the faulty LEDs would be faster to find by a human looking at the picture provided by the computer vision system. A camera with a wider angled lens would also be a great and easy improvement that could be implemented to accommodate for a multitude of different sized LED panels, this

would however be more expensive and would only be a good option if this computer vision system is used in a production facility.

7.3 Future work

In the future this computer vision system could be implemented into an actual production facility where LED panels are being produced and need to be tested. This would mean improving the code to allow for more LED panel types to be selected. It would also require modifying the test so that it automatically runs again after it has finished testing a LED panel, if the system must be fully autonomous. There are possibly other considerations that would need to be taken into account, such as a testing enclosure if the lighting in the production facility is as bright as the LED panel that is being tested. In any case, the system would need further testing and calibration in the actual facility where it is being implemented, if it were to be adequately accurate to be used autonomously.

References

[1] Wikipedia contributors, "wikipedia.com," 18 Aprila 2022. [Online]. Available: https://en.wikipedia.org/wiki/Light-emitting_diode. [Accessed 5 May 2022].

[2] Wikipedia contributors, "wikipedia.com," 4 May 2022. [Online]. Available: https://en.wikipedia.org/wiki/RGB_color_model. [Accessed 5 May 2022].

[3] I. D, Microcontroller based applied digital control, John Wiley & Sons, 2006.

[4] T. Wellem and B. Setiawan, "A microcontroller-based room temperature monitoring system," *International journal of computer applications,* vol. 53, no. 1, 2012.

[5] Atmel, "ATmega328P 8-bit AVR Microcontroller with 32K Bytes In-System DATASHEET," 2015. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. [Accessed 25 April 2022].

[6] Microchip, "PIC16F87XA," 2003. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf. [Accessed 25 April 2022].

[7] Texas Instruments, "MSP430x11x MIXED SIGNAL MICROCONTROLLERS," 2004. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/27250/TI/MSP430.html. [Accessed 25 April 2022].

[8] Espressif Systems, "ESP32 Datasheet," 8 October 2016. [Online]. Available: https://cdn.sparkfun.com/datasheets/IoT/esp32_datasheet_en.pdf. [Accessed 25 April 2022].

[9] Atmel, "ATmega16U4/ATmega32U4 8-bit Microcontroller with 16/32K bytes of ISP Flash and USB Controller DATASHEET," April 2016. [Online]. Available: https://ww1.microchip.com/downloads/en/devicedoc/atmel-7766-8-bit-avr-atmega16u4-32u4_datasheet.pdf. [Accessed 25 April 2022].

[10] STMicroelectronics, "STM32F405xx STM32F407xx Datasheet - production data," August 2020. [Online]. Available: https://www.st.com/resource/en/datasheet/DM00037051.pdf. [Accessed 25 April 2022].

[11] STMicroelectronics, "STM32F745xx/STM32F746xx Datasheet - production data," February 2016. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f745ie.pdf. [Accessed 25 April 2022].

[12] F. Yam and Z. Hassan, "Innovative advances in LED technology," *Microelectronics Journa,* vol. 36, no. 2, 2005.

[13] Wikipedia contributors, "Wikipedia," 27 May 2021. [Online]. Available: https://en.wikipedia.org/wiki/Stokes_shift. [Accessed 11 February 2022].

[14] M. Barr, "Pulse width modulation.," *Embedded Systems Programming,* vol. 14, no. 10, pp. 103-104, 2001.

[15] Wikipedia contributors, "Wikipedia.com," 18 December 2020. [Online]. Available: https://en.wikipedia.org/wiki/Duty_cycle. [Accessed 29 April 2022].

[16] T. Huang, "Computer vision: Evolution and promise.," CERN European Organization for Nuclear Research-Reports-CERN, 1996.

[17] R. Szeliski, Computer vision: algorithms and applications., Springer Science & Business Media, 2010.

[18] G. Guerra-Filho, "Optical Motion Capture: Theory and Implementation.," *RITA,* vol. 12, no. 2, pp. 61-90, 2005.

[19] K. Pulli, A. Baksheev, K. Kornayakov and V. Eruhimov, "Real-time computer vision with OpenCV.," *Communications of the ACM,* vol. 55, no. 6, pp. 61-69, 2012.

[20] OpenCV, "About - OpenCV," 2021. [Online]. Available: https://opencv.org/about/. [Accessed 10 December 2021].

[21] Wikipedia contributors, "Wikipedia," 15 April 2022. [Online]. Available: https://en.wikipedia.org/wiki/MoSCoW_method. [Accessed 25 April 2022].

[22] Canonical Ltd, "ubuntu.com," 2022. [Online]. Available: https://ubuntu.com/. [Accessed 25 April 2022].

[23] Manjaro GmbH, "manjaro.org," 2022. [Online]. Available: https://manjaro.org/. [Accessed 25 April 2022].

[24] Raspberry Pi Foundation, "raspberrypi.org," 2022. [Online]. Available: https://www.raspberrypi.org/. [Accessed 25 April 2022].

[25] Raspberry Pi Foundation, "raspbian.org," 2022. [Online]. Available: https://www.raspbian.org/. [Accessed 25 April 2022].

[26] PassMark Software, "cpubenchmark.net," 2022. [Online]. Available: https://www.cpubenchmark.net/. [Accessed 25 April 2022].

[27] Arduino, "Arduino® UNO R3 Product Reference Manual," 1 April 2022. [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf. [Accessed 25 April 2022].

[28] Arduino, "Arduino® MEGA 2560 Rev3 Product Reference Manual," 25 April 2022. [Online]. Available: https://docs.arduino.cc/static/e15cb755cdf3a1903389ecd1032326b7/A000067-datasheet.pdf. [Accessed 25 April 2022].

[29] F. Zehra, M. Javed, D. Khand and M. Pasha, *Comparative analysis of C++ and Python in terms of memory and time.,* 2020.

[30] JetBrains, "jetbrains.com," 2022. [Online]. Available: https://www.jetbrains.com/pycharm/. [Accessed 29 April 2022].

[31] Python Software Foundation, "pypi.org," 2022. [Online]. Available: https://pypi.org/project/pip/. [Accessed 29 April 2022].

[32] OpenCV, "docs.opencv.org," 29 April 2022. [Online]. Available: https://docs.opencv.org/4.x/d8/d01/group__imgproc__color__conversions.html. [Accessed 29 April 2022].

[33] OpenCV, "docs.opencv.org," 29 April 2022. [Online]. Available: https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html. [Accessed 29 April 2022].

[34] OpenCV, "docs.opencv.org," 29 April 2022. [Online]. Available: https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#gae57b028a2b2ca327227c2399a9d53241. [Accessed 29 April 2022].

[35] OpenCV, "/docs.opencv.org," 29 April 2022. [Online]. Available: https://docs.opencv.org/3.4/d6/d6e/group__imgproc__draw.html#gaf10604b069374903dbd0f0488cb43670. [Accessed 29 April 2022].