Western Norway University of Applied Sciences

BACHELOR THESIS:

BO21E-16 Tracking System for the Behavior and Activity of Ciona Intestinalis

Simen Fuglestad
Hector Fabian Moya
Endre Røberg Løseth

31. May. 2021

Document Control

Report title BO21E-16 Tracking System for the Behavior and Activity of Ciona Intestinalis	*Date/Version* May 30, 2021 *Report number:* B021E-16
Author(s): Simen Fuglestad Hector Fabian Morales Endre Røberg Løseth	*Course:* ELE150 *Number of pages including appendixes* *54*
Supervisor at Western Norway University of Applied Sciences Yngve Thodesen	*Security classification:* Open
Comments: We, the authors, allow publishing of the report.	

Contracting entity: Sars center	*Contracting entity's reference:*
Contact(s) at contracting entity, including contact information: Jørgen Høyer Tlf: 41636030 E-mail: Jorgen.Hoyer@uib.no	

Revision	Date	Status	Performed by
0.11	29.01.21	Preliminary study	Hector, Simen and Endre
0.12	15.05.21	First full issue	Hector, Simen and Endre
0.13	25.05.21	Second issue, revised and expanded	Hector, Simen and Endre
0.14	28.05.21	Added appendixes and minor corrections.	Hector, Simen and Endre
0.15	30.05.21	Final revision	Hector, Simen and Endre

Index

Preface

The project described in this report is the result of the work carried out by a group of 3 students at the Western Norway University of Applied Sciences, enrolled at the Electronic Engineering program. It amounts to our final year project and as such serves as the conclusion of our studies. The project itself has been carried out on behalf of the Sars International Centre for Marine Molecular Biology, a prominent research institution situated in Bergen, Norway.

We would like to extend our heartfelt gratitude towards our educational institution, our supervisor, and of course the good people over at the Sars center for providing us with the opportunity to work on this exciting project. As is hopefully evident from this report, we truly enjoyed it.

Summary

This document gives a complete description of the work carried out with regards to the bachelor project BO21E-16. The project was commissioned by the Sars International Center for Marine Molecular Biology, with the goal of designing and implementing a fully working system for experimentation and visual data collection on marine micro-organisms, namely the Ciona Intestinalis, from a sideways perspective. This stands in parallel to a similar project already carried out by a researcher and employee at Sars that provided this exact functionality except from a top-down perspective. Required critical features included a 3D-printable container, video recording, custom experimentation design and video processing to obtain tracking data. The requirements were obtained qualitatively with no clear or significant budget limitations, resulting in a high degree of freedom and flexibility in terms of design, implementation, and resource usage. Keeping frequent communications with our contacts at Sars, work was carried out continuously and dynamically as features were added, extended, simplified, clarified, or even cut altogether. The product is now at a functioning prototype stage and is being put to use by biology students from HVL, brought on by Sars for various academic and educational pursuits. Though certain aspects and features of the prototype could benefit from enrichment, we are under the impression that Sars are satisfied with our efforts and thus we feel it apt to proclaim our work as successful. Documentation and usage of version control was employed to simplify future development independently from our efforts, and it is our ambition that this will enable the researcher who built the previous system to continue development in the future. Finally, the project is publicly available as open-source and licensed accordingly.

1 Introduction

Before we dive into analysis or technical details, we would like to start off by establishing the background of the project, the contracting entity, as well as a “nutshell” version of how we intend to approach the task at hand.

1.1 Contracting entity

The Sars International Center for Marine Molecular Biology is a research facility established in 1997 (UIB, 2021). The Sars center, which is organized in eight different research groups, is located at the “High Technology Center” in Bergen and studies biological processes in marine organisms with a focus on evolution and development.

Figure 1-1 Sars Center

1.2 Problem description

Currently, the Sars center is engaged in research regarding how groups of interacting cells in a nervous system can manifest as behavior and movement. A key part of the data gathering process involves studying a marine animal known as *Ciona Intestinalis* (CI), which serves as a “model organism” due to their comparatively simple neurological structure. These are of microscopic size and require specialized setups to study properly. The researchers have come up with a design that makes use of specialized cameras tracking the organisms from a top-down perspective. This will be referred to as the “existing solution” further on in this document. Below are two pictures showing implementations of this design: The first is a stationary setup from the lab, while the second is a portable version utilizing the same concept but with less infrastructure.

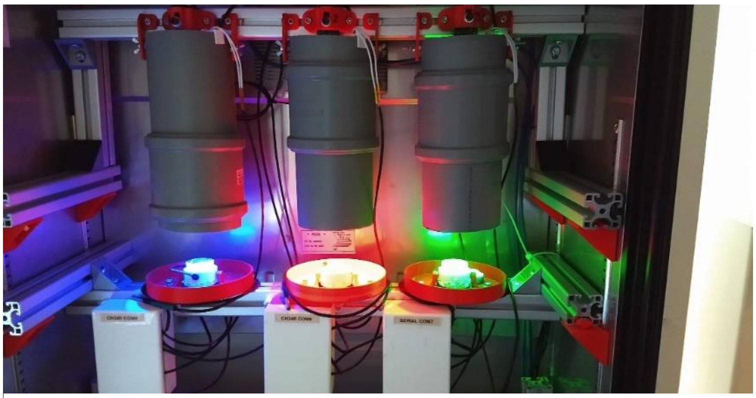


Figure 1-2 Existing solution, stationary setup

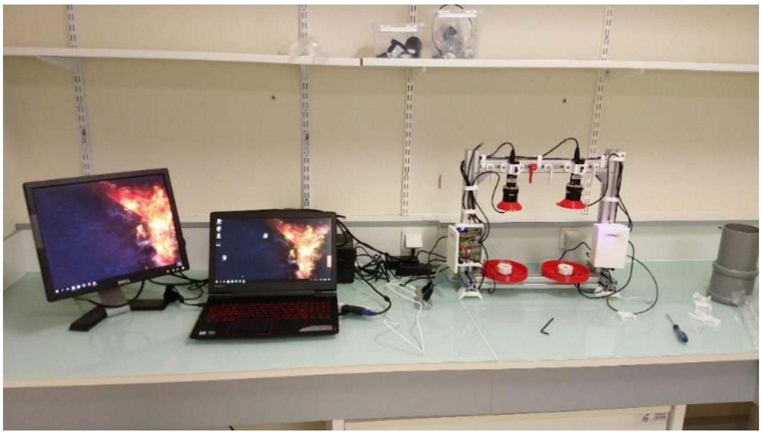


Figure 1-3 Existing solution, portable setup

These solutions, however, does not provide a full view of motion displayed by the *Ciona Intestinalis* as they are submerged in liquid and can move in a three-dimensional space. The problem, as it currently stands, constitutes designing and implementing a setup that either improves upon or reinvents the current solution to allow gathering of data describing the vertical motion of the *Ciona Intestinalis*.

1.3 Main idea of solution

The main idea of our solution is to redesign the current solution to work from a vertical perspective, while adding to or improving upon the design, as necessary. More specifically, it will consist of a closed system where the frame will be 3D printed and there will be a slot for the cuvette, camera, lens, electronics, and Light emitting diodes (LEDs). The entire system should be modular, expandable and controlled through a Graphical user interface (GUI).

2 Specification of requirements

Below is a table giving brief summaries of the required specifications that the finished product will adhere to. These were worked out in collaboration with a representative from Sars. To give an idea of their respective importance, keywords *must* and *should* have been used to describe the level of severity per requirement. *Must* indicates a critical "must-have" requirement, while *should* indicates a requirement strongly desired but not critical to final delivery.

Requirements	**Specification**
Vertical Tracking/Acquisition Control	Must be able to track multiple CI from the *side* and record tracking data.
Stimulus Control	Must be able to supply stimulus control to the organisms by making use of LED(s)
Functional in low light/darkness	Must be able to read tracking data in environments with sparse illumination
Infrared Capabilities	Must be able to obtain tracking data using infrared
Maintainable Code	Must be written in python, should reuse existing solution as much as possible
Version Control	Must use Git and GitHub for version control
3D-printable components	Must be comprised of 3D-printable components as much as possible
GUI	Must be able to interact with system through a GUI
Visualize Ciona Intestinalis	Must be able to provide visualization of CI behavior
Size constraints	Should fit inside an incubator for temperature control.

Figure 2-1 Product requirements

To ensure that the system will indeed end up as a satisfactory product, we will rely heavily on feedback from our contacts at Sars to provide us with directions as development progresses. The next chapter provides additional in-depth analysis of each requirement, breaking them into more manageable tasks in preparation for the development process.

3 Problem analysis

In this section we will further investigate the requirements specified above to provide a more detailed description of how to meet them. A "divide and conquer" approach will be employed to modularize each requirement into more concise pieces that are easier to assess in isolation. We will refer to these new pieces as *sub-requirements* and aim to formulate them in a way that allows them to be used as minor milestones to track the progress of the project. When all sub-requirements of a single requirement are completed, this can then be referred to as a major milestone as it marks the completion of a major requirement. In addition, some of the original requirements are already quite modular and have therefore been redefined as sub-requirements where appropriate. For instance, "Infrared Capabilities" is an essential yet quite specific part of providing successful tracking and has therefore been reassigned as a sub-requirement of "Vertical Tracking". A table is provided to give an overview of regular and sub-requirements. More detailed descriptions with accompanying justifications can be found below the table itself.

Requirements: Major Milestones	Sub-Requirements: Minor Milestones
Vertical Tracking	• Camera with infrared capabilities • Cuvette container for the CI • Functional in low light/darkness • Collect tracking data
Stimulus Control	• LEDs for light stimuli • Microcontroller for controlling light stimuli • Interact with system through GUI
Visualize CI	• View CI response to stimulus as visualizations • Interact with visualization through a GUI
3D-printable Components	• Designing and prototyping modular components

Figure 3-1 Requirements as milestones

3.1.1 Vertical Tracking

The Ciona Intestinalis being micrometers in size, transparent in their physical nature and their response to light makes the use of infrared lighting a necessary feature in the setup. The Ciona Intestinalis live in a dark natural habitat and therefore the system is required to function in low light/darkness to simulate their natural environment. The vertical tracking system will serve as an acquisition control, capturing image data for further use in other parts of the system.

3.1.2 Stimulus control

CI responds to light. The stimulus will be in the form of LEDs that can emulate daylight, that we can turn on and off with the GUI. The stimulus control will be linked to the acquisition control.

3.1.3 3D-printing

The frame and container will be 3D printed in Polylactic acid (PLA). This is an easy and cheap way to prototype. It also allows parts to be replaced if damaged. It will also reduce the size of the system so that it can fit inside an incubator for temperature control.

3.1.4 Visualize CI

For the purpose of studying the behavior of CI, visualizing the image data provided by the vertical tracking system is hugely beneficial compared to reading the data directly. Therefore, the finished product will need to supply the user with a rich interface to view and control visualizations depicting the movement of CI.

3.2 Description of initial solution

Based on the previous analysis and specification of requirements, we have come up with a sketch of a solution. Firstly, we would like to present the physical structure of the setup via the computer-rendered images below.

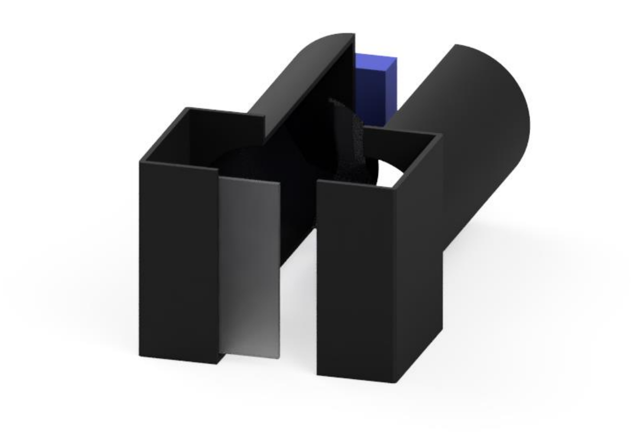


Figure 3-2 Computer-rendering of current solution 1

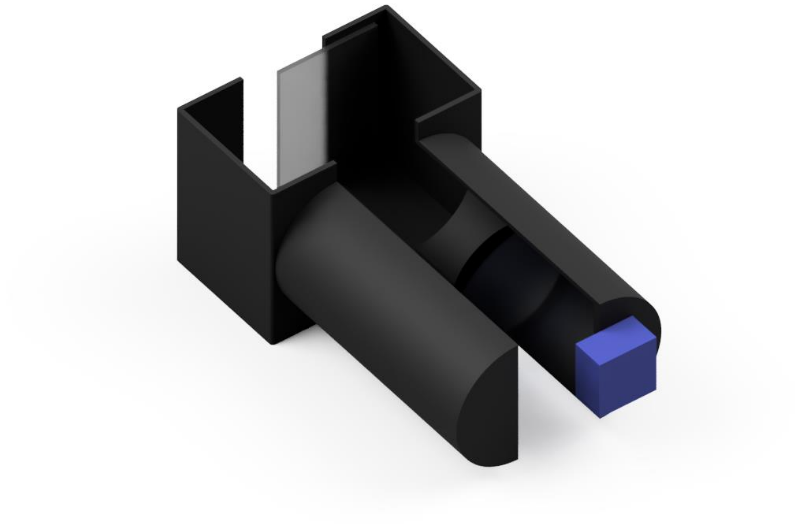


Figure 3-3 Computer-rendering of current solution 2

While inspired by the existing solution, this new design aims to provide sideways perspective in a closed setup. The camera, indicated by the blue cube, sits at the end of a closed cylindrical chamber. On the opposite end a cuvette containing the organisms will be held and the lens can be found between it and the camera. A plexiglass plate is situated behind the cuvette itself, allowing infrared backlighting. A benefit of this design is that it can be expanded upon by adding a structure similar to the existing solution to record tracking data from the top, thus keeping the possibility of the aforementioned bonus requirement available. If, for any reason, this bonus requirement is scrapped, then the top can simply be a lid through which the cuvette is inserted. If this bonus requirement is implemented, then the cuvette will necessarily have to be inserted from the side. It should also be mentioned that LEDs will be fitted inside the structure to provide the stimulus control.

Another aspect of the design that is still in the brainstorming phase, though worthy of mention, is the idea of keeping the CI in a liquid in the structure itself, scrapping the use of a cuvette. The purpose of this would be to try and reduce light refraction through unclear surfaces such as the cuvette walls and record the CI directly. This hinges on quite a few factors: Liquid pressure against the lens, waterproofing electronics in case of leakage, a pumping system to supply and withdraw liquid, getting the CI in and out of the structure etc. For the time being, we will consider this an experimental alternative (as opposed to an actual solution alternative) that can be tried out during testing in case the original variant falls short.

3.2.1 Hardware

The contracting entity has not made any specific requirements for hardware components, which gives the opportunity to use and assemble the hardware components as we see fit. For the microcontroller that is going to control the LED lighting we went with the Arduino NANO, which were used by Sars for their previous horizontally oriented system and thus they had several units readily available. Warm white light LEDs for simulating daylight, and infrared LEDs for lighting up the CI. The Sars center has an available 3D-printer, we also have our own private 3D printer which we will use when necessary. PLA filament will be used for 3D-printing because it is inexpensive and easy to print with. The Sars center will supply us with a DMK-UP1300 USB-3 camera from "The Imaging Source" (DMK 33UP1300, 2021), using a MVL35M1 objective lens (Camera Lenses for Machine Vision, 2021).

3.2.2 Purchases

The Sars center has stated that they will not be providing any explicit details of a budget, and that we are free to request any purchases that we deem required or useful (within reason). The only constraint we were given explicitly were that we should, to the best of our ability, make use of domestic retailers. Since cameras and lab equipment are already provided, this leaves mostly inexpensive electronics and various structural components to be purchased. Below is an overview of the parts we require to commence building an initial prototype, with corresponding retailers.

MOSFET	IRL510PBF | Vishay MOSFET N, 100 V 5.6 A 43 W TO-220 | Elfa Distrelec Norge
Custom circuit board	PCB Prototype & PCB Fabrication Manufacturer - JLCPCB
White light LED	JU1215-KM277K4-09804-450T | Everlight Electronics LED COB Varm hvit 11 V 3.7 W | Elfa Distrelec Norge
Miscellaneous	This will include wires, soldering tin, glue, resistors, paint, PLA, sandpaper and so on.

Figure 3-4 Purchases

3.2.3 Software components

As covered in section 1.3, the Sars center has already come up with and followed through on a design that allow tracking the CI from a top-down perspective. The implementation includes a somewhat extensive repository of source code that can be found on its respective GitHub page (imMobilize, 2021), and we have been encouraged by representatives from Sars to reuse this code to the degree that it is beneficial. The codebase itself can be split into two categories: Interaction with microcontroller and interaction with user (interface), and both employ the following free-to-use software libraries and languages:

- Python3
- PyQt5 (PyQt5, 2021)
- C++ (for Arduino)
- OpenCV (OpenCV, 2021)

Using these same technologies, we will work to adopt the existing codebase to fit the needs of our project. The software development will make extensive use of Git and GitHub, which opens up the possibility of simply forking (Forking, 2021) the existing repository. Finished software components will also aim to be cross-platform, functioning on most systems that can readily install the required software and libraries.

3.2.4 Documentation and development practice

We will aim to document all parts of the development process where it is deemed necessary or practical. Most of these documents, such as design plans, meeting journals, work logs and so on, will be hosted on a cloud-based document-sharing service such as OneDrive. The only exception is a detailed work log of the software development process, which is tightly integrated with the usage of Git. This process will follow the renowned practice of GitFlow (atlassian, 2021), which should result in a rich commit log that can be evaluated during and post-development. When it comes to developing hardware components, most of the work will take place in a lab setting, either at the school facilities or at the Sars center. It is somewhat counterintuitive to apply the concept of GitFlow to hardware development, yet we will strive to keep both software and hardware development as agile (Atlassian, 2021) as possible.

3.3 Problem analysis conclusion

In this section we have provided an overview of the problem at hand and how we intended on tackling it. The current solution, based on the specification of requirements, is very similar in spirit to the existing top-down solution presented to us, yet allows for expansion and provides the requested feature of acquiring tracking data from the side. It should also be relatively inexpensive, making use of provided cameras and requiring mostly low-cost purchases. In light of this, as well as our usage of open-source software languages and libraries that the contracting entity is familiar if, we would like to make the case that our design is justified in that it could serve as a useful solution while also keeping within the ramifications provided by the Sars center.

4 Realization of selected solution

This section aims to give an overview of how the various parts of the system and their interactions were realized. The bulk of our attention will be directed towards implementation details and the employed technologies. The only exception to this is section 4.3.2, *Designing the layout*, as it is quite impractical to give a strictly technical view of design choices in terms of user interaction. See section 6 for a more prosaic description of the system as well as reflections pertaining to the development process.

4.1 The system: An illustrative summary

The various components that comprise the system and their interconnections are illustrated in the figure below.

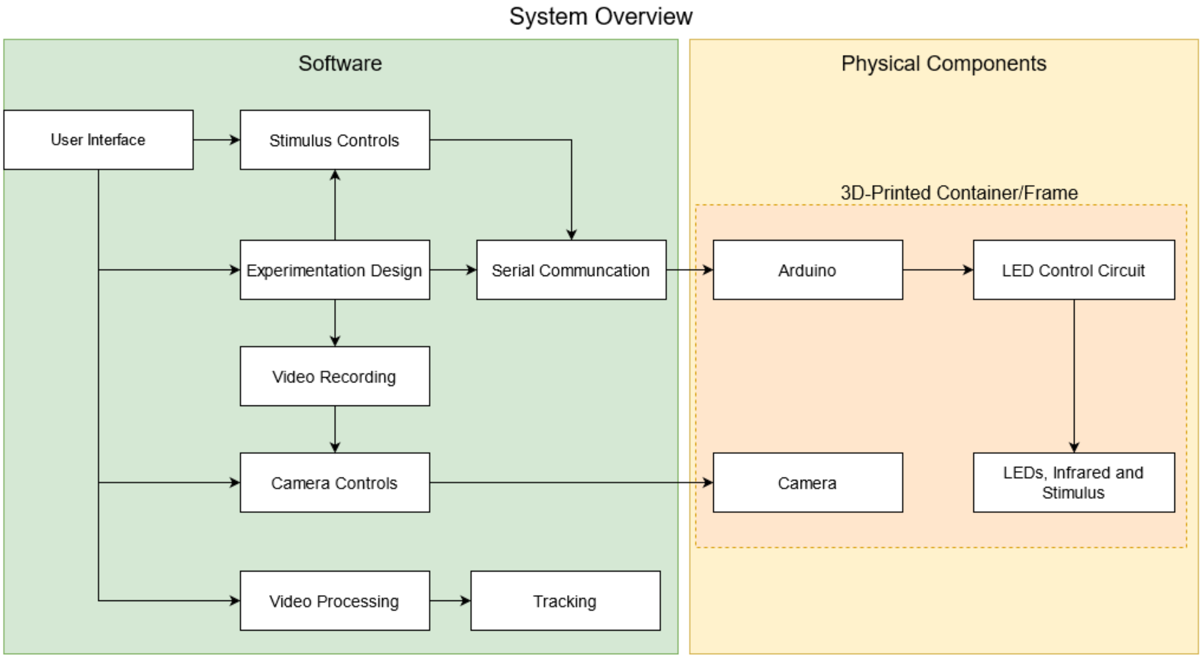


Figure 4-1 System Overview

In essence, the system can be grouped into “software” components and “physical components”, each with their own respective sub-components. Anything under software simply implies that the feature is entirely software-centric: An application running on a user’s computer. The physical components are namely just that, and they are all designed to sit inside or adjacent to the 3D-printed container. See section 6.1 *Product Overview* for a more thorough discussion of the design from a more feature-centric perspective.

4.2 Physical frame

This section is aimed to get an overview of the different materials and tools we used to design and build the physical frame. As you can see below, the finished product consists of multiple small sections.

4.2.1 3D modelling

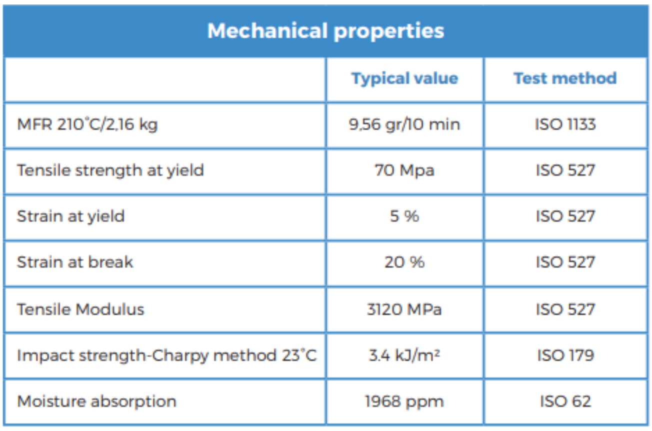
The software we used is called Fusion 360 by Autodesk. It is an excellent tool for precise modeling of 2D and 3D objects. Fusion 360 offers CAD, CAM and CAE possibilities while still being licensed as free to use for educational and hobbyist purposes.

4.2.2 Materials

To print the physical frame we used PLA, which is a biodegradable, sustainable and food safe polymer made from organic sources. It is the most common used filament in Fused Filament Fabrication (FFF) 3D printers for its ease of use and a wide range of applications, especially those not mechanically or thermally demanding (P, 2021).

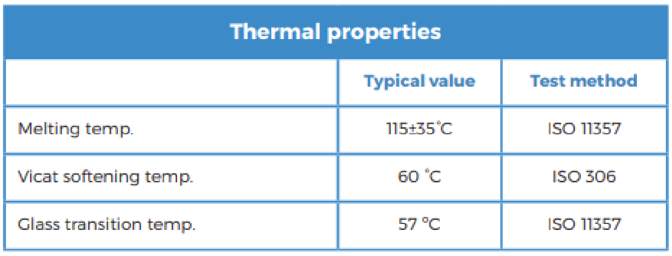
Properties of PLA:

- Detailed and glossy surface quality
- Good tensile strength
- Rigid, fragile behavior
- Good UV resistance
- Withstand operating temperatures up to 50 °C.
- Odor-free, ideal for educational and office environments
- Compatible with PVA supports
- Low humidity resistance.



Mechanical properties		
	Typical value	Test method
MFR 210°C/2.16 kg	9.56 gr/10 min	ISO 1133
Tensile strength at yield	70 Mpa	ISO 527
Strain at yield	5 %	ISO 527
Strain at break	20 %	ISO 527
Tensile Modulus	3120 MPa	ISO 527
Impact strength-Charpy method 23°C	3.4 kJ/m²	ISO 179
Moisture absorption	1968 ppm	ISO 62

Figure 4-2 Mechanical Properties (BCN3D, 2021)



Thermal properties		
	Typical value	Test method
Melting temp.	115±35°C	ISO 11357
Vicat softening temp.	60 °C	ISO 306
Glass transition temp.	57 °C	ISO 11357

Figure 4-3 Thermal Properties (BCN3D, 2021)

4.2.3 Printer

The 3D printer we used is Ender 3 Pro by Creality (creality3d-ender-3-pro-high-precision-3d-printer, 2021). This is a popular printer due to its price, ease of use and availability of spare parts. The Ender 3 Pro has 220 x 220 x 250 mm print volume. And a ± 0.1 mm printing precision. To generate files we use Cura open source software (Ultimaker-cura, 2021) with the following settings:

- 0.16mm Layer height
- 20% Infill density
- 220 ^{o}C Printing temperature
- 60 ^{o}C Build plate temperature
- 25 mm/s Retraction speed
- 6mm Retraction distance

4.3 Control circuit

This section will give more technical details and specification about the components used in designing the control circuit.

4.3.1 Arduino NANO

Figure 4-4 Arduino NANO

An Arduino NANO serves as the intermedium between the software and the controlling circuitry. It is a small yet capable microcontroller operating at 5V with a clock speed of 16 MHz, providing a total of 22 input/output (IO) pins. (Arduino Nano , 2021). Of particular interest to us are the 6 pins that support 8-bit pulse-width modulation (PWM) (Barr, 2021), which outputs a voltage of 5V with a varying duty cycle, will be used in conjunction with the control circuitry to regulate the intensities of the LEDs. Programming the Arduino is performed using the Arduino IDE, a graphical environment providing code editing of the Arduino Language (Language Reference, 2021) as well as graphical wrappers to lower-level tasks such as compiling and uploading program code to Arduino devices (Arduino IDE 2, 2021). Implementing a serial communication interface (Serial, 2021), the Arduino receives signals from the software and adjusts the duty cycle on its PWM pins accordingly to regulate the intensity. The current implementation as the Arduino continuously checking for input signals in its idle state, implying that a single successful upload is necessary yet sufficient for properly operating the system (given that power has not been disconnected from the Arduino for an extended period of time).

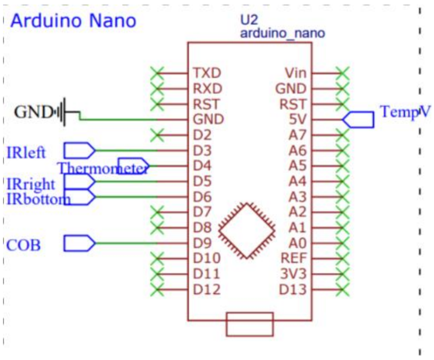


Figure 4-5 Arduino NANO pin layout

4.3.2 Power supply

The power supply used in the circuit is an adapter with 12V and up to 2.0A output supply and can be plugged in any standard European 50/60Hz, 100V-240V AC electrical outlet. The reason for choosing this power supply is simply because it was what we had available, and since it covers the power requirements needed for our setup, it seemed like the obvious choice.

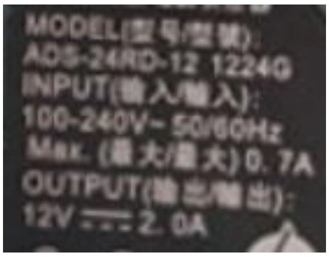
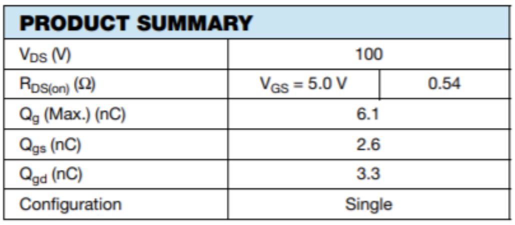

MODEL(型号/型號):
ADS-24RD-12 1224G
INPUT(输入/輸入):
100-240V~ 50/60Hz
Max. (最大/最大) 0.7A
OUTPUT(输出/輸出):
12V ⎓ 2.0A

Figure 4-6 Power supply

4.3.3 MOSFET transistor

MOSFETs or "*Metal Oxide Semiconductor Field-Effect Transistor",* are voltage-controlled devices consisting of three terminals, gate, source and drain (ElProCus, 2021). The drain terminal is controlled by the voltage of the gate terminal and makes the MOSFET a suitable switch for the circuit. While the BJT transistor might be more suitable in low current applications, the MOSFET will give the light circuits more scalability in terms of light intensity since it can withstand greater voltages and currents. The MOSFET transistors are available in 4 different types, such as P-channel or N-channel with either an enhancement mode or depletion mode. For the enhancement mode MOSFET, which is the type of transistor we chose, if the voltage on the 'G'-terminal is low then the device does not conduct. When more voltage is applied to the gate terminal, then the conductivity of this device is good. This makes the enhancement mode MOSFET suitable to use as a switch for the light circuit. The MOSFET we chose for the circuit is the IRL510 MOSFET. A total of four N-channel enhancement MOSFETs with logic level gate drive are used in the control circuit as switches for each of the LED light circuits.



PRODUCT SUMMARY		
V_{DS} (V)	100	
$R_{DS(on)}$ (Ω)	V_{GS} = 5.0 V	0.54
Q_g (Max.) (nC)	6.1	
Q_{gs} (nC)	2.6	
Q_{gd} (nC)	3.3	
Configuration	Single	

Figure 4-7 MOSFET characteristics

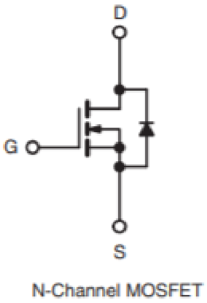


Figure 4-9 symbolic representation of N-channel MOSFET

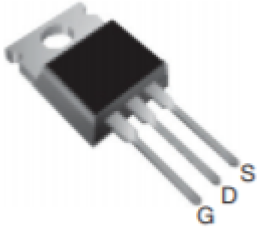


Figure 4-8 Pin layout of MOSFET

4.3.4 Chip On Board (COB) LED

For the light stimulus we chose a white-light COB to simulate sunlight and to ensure sufficient lighting. “Chip On Board” refers to the mounting of a LED chip in direct contact with a PCB substrate to produce LED arrays. COB technology allows for a much higher packing density than other LED technologies such as Dual In-line Package (DIP) or Surface Mounted Device (SMD) and therefore gives a higher light intensity from a smaller surface area. (Rice, 2021) Below is an illustration of this.

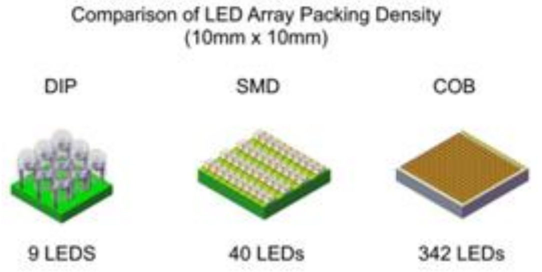


Figure 4-10 Comparison of LED Array Packing Density

Other advantages of the COB LEDs are that they have a superior thermal performance for increased life cycle, stability, reliability and design simplicity because only one circuit and two contacts are required, as well as a larger cooling area. The COB can also be paired with a heatsink which gives an even greater thermal resistance for running on higher light intensities (Rice, 2021). The COB of choice is the JU1215 produced by *Everlight Electronics*. It emits wavelengths in the range from ~420nm to ~700nm as shown in the figure below. (JU1215, 2021)

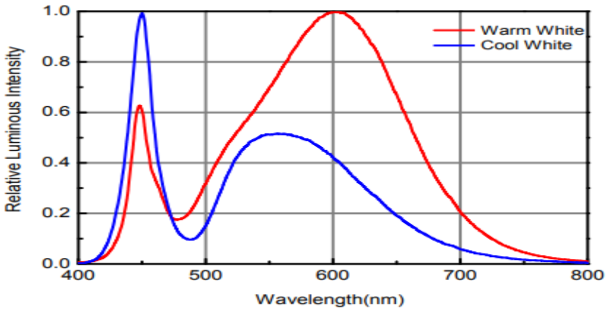
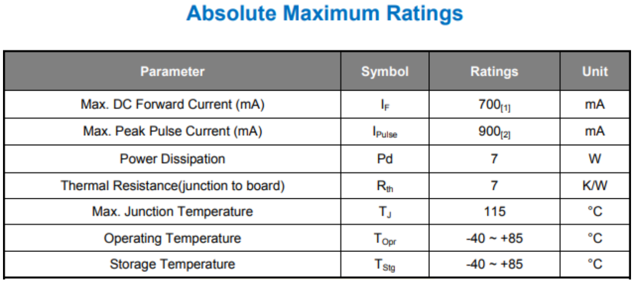


Figure 4-11 COB Wavelengths



Absolute Maximum Ratings

Parameter	Symbol	Ratings	Unit
Max. DC Forward Current (mA)	I_F	700[1]	mA
Max. Peak Pulse Current (mA)	I_{Pulse}	900[2]	mA
Power Dissipation	Pd	7	W
Thermal Resistance(junction to board)	R_{th}	7	K/W
Max. Junction Temperature	T_J	115	°C
Operating Temperature	T_{Opr}	-40 ~ +85	°C
Storage Temperature	T_{Stg}	-40 ~ +85	°C

Figure 4-12 COB ratings

The dimension of the chip is 12mmx1,3mmx15mm and requires little space to incorporate in our setup. The COB LED proved to be more than powerful enough to supply sufficient stimulus lighting, with a maximum forward current of 700mA, the COB only required a maximum input of 7mA to get the desired lighting intensity for the stimulus.

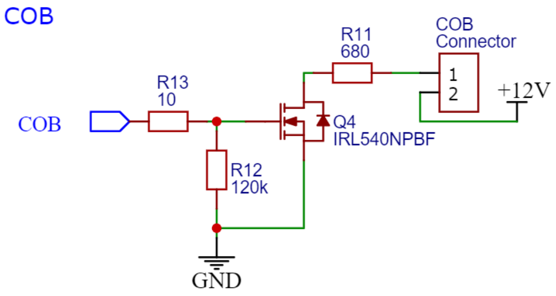


Figure 4-13 COB circuit drawing

4.3.5 Infrared LEDs

The infrared (IR) LEDs used in the setup were supplied by the contracting entity and did not come with any form of identification other than the forward voltage and wavelength of the LEDs written on a container. As shown in the image below the LEDs have a forward voltage of 1.4 – 1.5V and emits light with a wavelength of about 850nm. The infrared LED circuits are controlled separately with eight lights on three sides of the cuvette, left, right and bottom, where each of the sides are supplied with 12V giving each of the diodes a voltage drop of 1.5V. For details about the current through the IR LEDs, see section 5 since the current flow through the circuits is adjusted relative to the image results. The circuit for each of the infrared light circuits are identical and is shown in the circuit image below.

Figure 4-15 Infrared diodes container

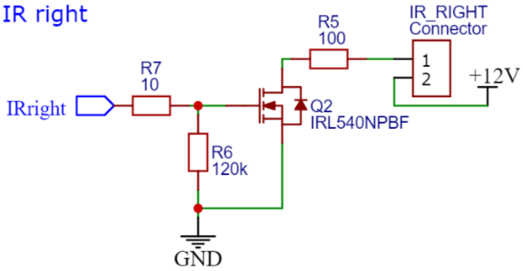


Figure 4-14 Sample IR circuit

4.4 Serial Communication

The direct link between the software and the Arduino in the system is the serial communication channel. It is, in its current state, a one-way channel that takes data of a format that specifies a number value (floating point or integer) along with a string that serves as a tag. The number decides the intensity of the LED light and the tag specifies which portion of the LEDs that should take on this value. On the software application side of things, the file under *srd/serial_interface/serial_interface.py* implements methods to handle the connection to the Arduino as well as a simplistic way of sending a single data point. It should also be noted that we have made use of the module *pySerial* to deploy the serial communication (pySerial, 2021).

4.5 User interface

This sub-section gives a brief description on the technologies employed to create a GUI for end user to interact with the system. It is important to note that there will not be any detailed reasoning of the actual *user experience* (UX) or any justification of design: These can be found in section 6.2 *Design choices and user experience.*

4.5.1 Python, PyQt and pyqtgraph

As mentioned in section 3.1.3, the programming language of choice going into development is python3, referred to as python henceforth, which is high-level and supports a number of programming paradigms (Python 3.9.5 documentation, 2021). The selected technological framework for this project was initially PyQt5, however the decision was made to go with PySide6 (PyPi, 2021) as it is more recent and thus has longer projected life cycle. It is also worth noting that PySide6 is a full rebinding of the original Qt, which uses C++ as its primary programming language and is intended for full-scale application development across a range of systems (qt.io, 2021) and thus provides a more than sufficient platform for developing a user interface. For plotting purposes, the group have made use of pyqtgraph, a popular tool for displaying various graphical elements that also integrates nicely with PySide6 (pyqtgraph.org, 2021).

4.5.2 Designing the layout

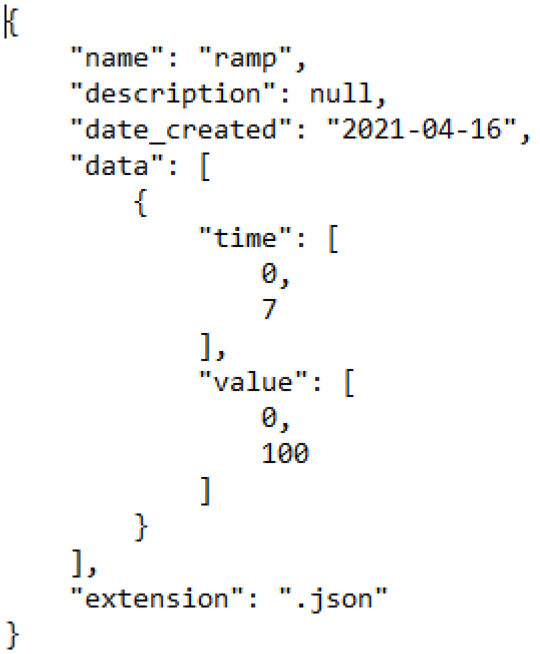
As mentioned directly above, PySide6 was selected as the principal tool for developing the user interface. An important factor in making this decision is the fact that PySide6 is compatible with Qt Designer (qt.io, 2021), a powerful utility for designing interfaces compatible with various Qt frameworks. Qt designer has been used extensively and has allowed for rapid initial designs, as well as simplified adjustments later during development. Designing using this tool results in the generation of a ".ui" file that describes the layout of a Qt application via a particular kind of structured markup language. This file can then be used to generate python code compatible with an application written in PySide6, using the command line tool *Pyside6-uic* (doc.qt.io, 2021). Thus, during the development process, iterating on the design simply consisted of editing the UI in the designer, recompiling to python code, and subsequently verifying changes in the application.

4.6 Experimentation design and stimulus control

This section will detail the underlying logic and storage functionality that enables users to work with controlled experiments.

4.6.1 Stimulus control

As described in section 4.4, it is the job of the serial communication interface to relay values to the Arduino that controls the LED circuit. These values must, however, be entered by a user and collected in a controlled fashion. Whenever a user adds stimulus with a given value-time range, this is stored in working memory as a list of data points with 4 entries: a start time, an end time, a start value, and an end value. These values can be saved in a json file (json.org, 2021) for later usage. Files saved like this are referred to as "stimulus profiles", below is an image of sample file that illustrates the structure.



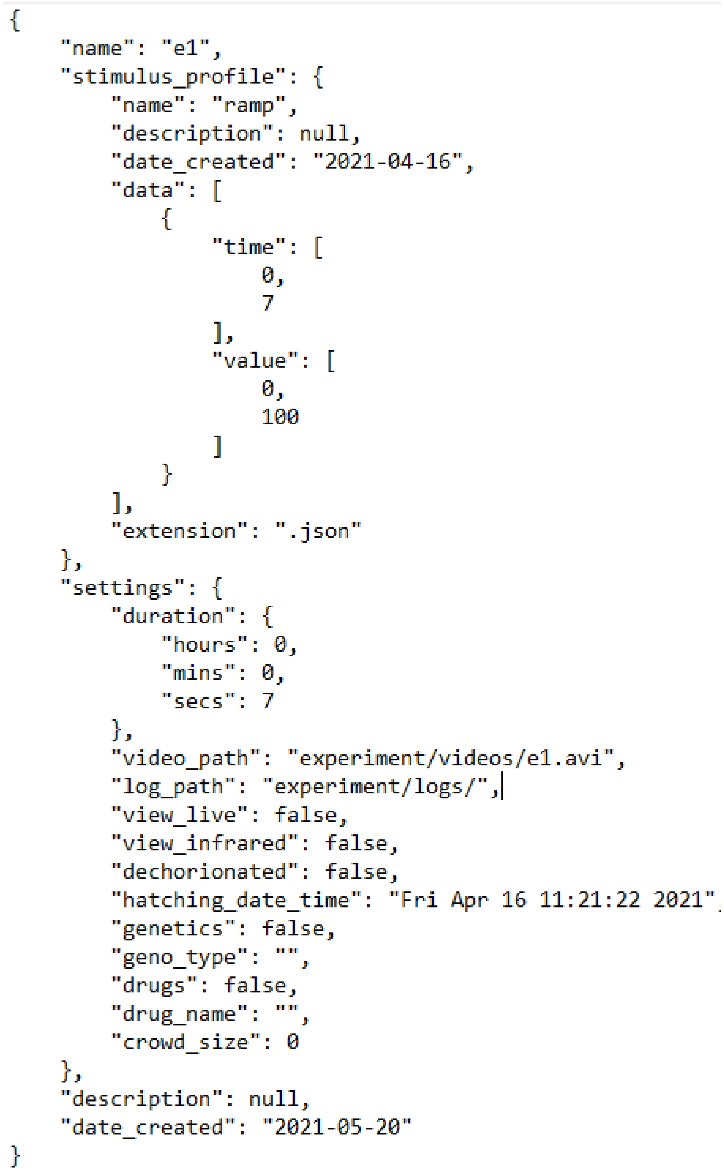
```
{
    "name": "ramp",
    "description": null,
    "date_created": "2021-04-16",
    "data": [
        {
            "time": [
                0,
                7
            ],
            "value": [
                0,
                100
            ]
        }
    ],
    "extension": ".json"
}
```

Figure 4-16 Sample stimulus profile

The data points that constitute the values to be plotted are stored under “data”, the other entries are meta-data and are created automatically upon saving, as well as inputted into their respective fields in the user interface when loaded. Saving and loading is implemented using python libraries available with the standard distribution, namely *json* (docs.python.org, 2021) and *os* (docs.python.org, 2021).

4.6.2 Experiment configuration

Designing experiments amounts to configuring data that are stored in working memory pertaining to the experiment, then this data can be saved in much the same manner as the stimulus profiles discussed in the previous section. These are accordingly referred to as “experiment profiles” and saved as json files. Below is a screenshot taken from a sample of such a file.



```
{
    "name": "e1",
    "stimulus_profile": {
        "name": "ramp",
        "description": null,
        "date_created": "2021-04-16",
        "data": [
            {
                "time": [
                    0,
                    7
                ],
                "value": [
                    0,
                    100
                ]
            }
        ],
        "extension": ".json"
    },
    "settings": {
        "duration": {
            "hours": 0,
            "mins": 0,
            "secs": 7
        },
        "video_path": "experiment/videos/e1.avi",
        "log_path": "experiment/logs/",
        "view_live": false,
        "view_infrared": false,
        "dechorionated": false,
        "hatching_date_time": "Fri Apr 16 11:21:22 2021"
        "genetics": false,
        "geno_type": "",
        "drugs": false,
        "drug_name": "",
        "crowd_size": 0
    },
    "description": null,
    "date_created": "2021-05-20"
}
```

Figure 4-17 sample experiment profile

Here we can see that there several more fields related to data about the experiment, which are all either entered from the user interface of generated automatically (i.e. "date_created" is generated upon creation). Also note that an experiment contains a stimulus profile, the very same data structure as we saw in the previous section. Loading or saving an experiment is done in much the same way as a stimulus profile, except for the added data points and the need to add or extract a stimulus profile. The same libraries as before are used in this regard.

4.6.3 Running an experiment

Once an experiment has been configured and, alternatively, saved, a user running the experiment results in the data being passed along to a class made specifically for dealing with ensuring correct execution of experiments (at least on the software side of things). There are essentially two steps to the running process: Compute intervals of stimulus values and sending the correct number of inputs to the serial communication channel with proper timing. The first point is necessary because stimulus values only have pairs of start and end values, so to ensure a proper gradient a step size must be computed if the user wants a linear gradual change of stimulus over time. The second point is crucial to ensure that the stimulus is timed according to the user's design, and is handled with QTimer, a class in Qt that supports timing controls with an ideal margin of error 1 millisecond (doc.qt.io, 2021). Timing events, fired on the Qt event loop in the application (wiki.qt.io, 2021), signals that the next value should be passed to the serial communication interface. This is done progressively until all values are sent or the duration of the experiment is over (which is part of the timing calculation). An experiment going for longer than the provided stimulus will simply signal an "off" value to the serial communication interface. Also note that running an experiment provides a simple Boolean locking mechanism to prevent multiple experiments from running on top of each other.

4.7 Tracking and analysis

This section will detail the various tools and technologies we used to achieve the collection of data from the movement of Ciona Intestinalis, and our reasoning and logic behind our decisions. As we have detailed earlier in this report the animal CI is micrometer in size and quite transparent. This is a problem when we need a clear image of the animal to collect reliable data that can be used for scientific purposes. There is also the problem of animals moving past one another, this can confuse the algorithm that is handling the identification of each individual in the container.

4.7.1 Image processing

OpenCV is an extensive computer vision library containing, amongst other things, a multitude of algorithms and routines for image capture and processing (OpenCV, 2021). It also has python bindings readily available for easy integration with the software and provided a platform-agnostic interface for working with the camera.

4.7.2 Camera

To record the Ciona Intestinalis, we were provided with several units of the DMK 33UP1300 USB3.0 monochrome industrial camera (DMK 33UP1300, 2021). With a 1.3 megapixel (MP) resolution and a cap of 210 frames per second (FPS), this camera proved more than satisfactory.

4.7.3 Optics

The optics we used to get a clear image of the test subjects is MVL35M1 from Thorlabs (Camera Lenses for Machine Vision, 2021).

The specifications are:

- 35 mm Focal Length
- f/1.4 Aperture(max)
- 300 mm Min Object Distance
- 26.1 ^{0}Field of View (1")^{c}

Field of view and object distance can be altered with the use of C-mount spacers.

Figure 4-18 MVL35M1

5 Testing

This section details our approach to testing the various parts of the system, and how it has affected the final product as well as the development process.

5.1 Justification of testing practices

As this is a project that combines many fields of engineering and development, we can naturally assume that many types of testing are required to ensure proper functionality of the final product. However, there are a few considerations we would like to emphasize: Firstly, some parts of the system are less comprehensive and/or critical than others, and as such testing should be favored among the most important and error-prone components of the product. This can be clearly illustrated when comparing the 3D-printed container vs the control circuity: A faulty container will most likely still provide a decent platform for performing experiments, while a faulty circuit could render the entire product useless. Secondly, sophisticated components can be difficult and quite time consuming to test thoroughly. In our case, these kinds of components would be the Arduino NANO and the DMK 33UP1300 camera (see section 4.7.2 for more details on the camera used for this product). Testing these have merely involved verifying that they work as expected in the context of their functionality (for instance viewing images captured by the camera), any other significant testing has intentionally been left out. We have, however, directed our testing efforts towards the software and the circuitry of the system. Below are sections detailing the process of each.

5.2 Testing software

When it comes to software, large systems usually incorporate some sort of testing framework with varying degrees of complexity. For this project however, we have gone with an alternate and, probably, semi-controversial approach. Rather than spending significant time performing unit tests, we have made extensive use of the human factor during testing. In practice, this means that we have handed over a working prototype somewhat early, with a mutual agreement that users would report back flaws and bugs they found within the software. This has rendered us free to pursue further development of features, and as such we have relied heavily on communication to ensure that the product upholds the standards of its users. One could of course argue that this testing does virtually nothing to prevent edge cases, and we do not disagree in the slightest. However, we would like to point out that this agreement was worked out in collaboration with our contractor and as such served as a compromise between getting more features up and running and strict verification. It was the wishes of our contractor that a working prototype be ready and available to biology students performing experiments at the Sars center within a shorter time frame, and as such rigorous testing before initial delivery would be too time consuming. We are, however, under the impression that this arrangement has worked well, and software errors reported were in all cases handled either remotely or on site. There was also an intent to systemize this testing methodology by constructing test cases for users, simplifying the recreation process of eventual errors. The idea was to provide users with a spreadsheet containing sets of instructions to complete inside the application, accompanied by descriptions of expected application behavior. A user could then report whether going through the instructions resulted in the expected behavior, as well as adding notes or comments. Unfortunately, this idea was never further pursued, and as such all feedback were handled in less formal means via direct communications. At this point, the reader might be under the impression that software was simply written and shipped with no testing whatsoever, but this is also not the case. Software features were tested against all aspects of functionality before being uploaded to the central code base intended for users, though this testing process was more empirical in terms of simply verifying that new features or improvements did not break existing code. For cases were this happened, the version control system employed made it a simple task to revert to an older build and identify the error. Such testing, however, is of course not up to the standards of the rigorous manner often required by commercial grade software. With an extended time period, however, such testing frameworks could and should be put into place. This, however, is a decision we leave with SARS (see section 6.4 *Future prospects: The road ahead* for more info on development going forward.*)*

5.3 Testing Circuitry and Arduino

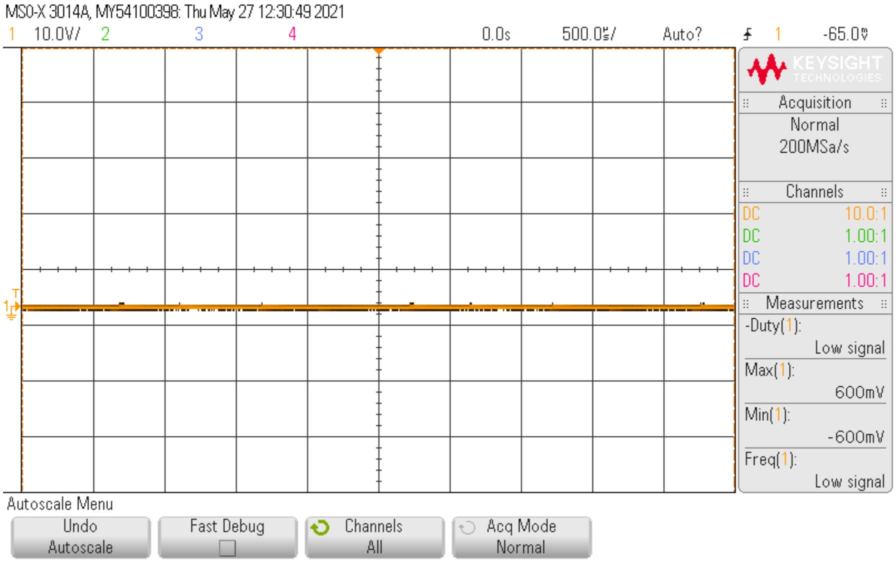
Testing of the circuitry amounted to verifying two aspects of the system: Correct values of currents and voltages throughout the system, and correct operation of signals. Each aspect is covered separately below.

5.3.1 Currents and voltages

In section 4.3.4 and 4.3.5, we covered the various voltage and current levels we expect to operate on with regards to the LEDs. Briefly summarized, we concluded that each cluster consisting of 8 IR diodes each requiring a forward voltage of 12V. The current levels would have to be adjusted to accommodate the application: Proper light intensity for backlighting and stimulus. Through collaboration with our contractor, we adjusted the respective resistors of each signal path to get a lighting they deemed satisfactory. To be more precise, the IR diode clusters were each given an additional 100ohm of resistance while the COB for stimulus lighting were given an additional 680ohm. With this complete, we took measurements to see if the adjusted values corresponded to our expectations and to verify if the adjustments impacted the circuit in any unexpected ways. Our choice of instrument for measuring this was the Keysight U1242C TRUE RMS (Keysight Technologies U1242C, 2021). The voltage values across each diode cluster matched our expectation of approximately 12 volts, with deviant values ranging from 0.1-0.3 volt. The current through each cluster, with intensity set to 100 percent, were measured to be 6.953mA, which is satisfactory and far beneath any maximum threshold values. The voltage across the output resistor of the COB were measured to be 4.735 volt, which at a 100 percent intensity amounts to a current of 6.953mA. This is also a safe value for the COB to operate as it is designed to handle far more current and still operate properly.

5.3.2 PWM Signal operation

Using a KEYSIGHT InfiiVision MSOX3014A oscilloscope (InfiniiVision 3000A X-Series Oscilloscopes, 2021) to probe the final version of the circuit, we sought to verify that the PWM signals emitted by the Arduino were indeed causing the infrared LEDs to behave as expected. Using the newest version of the software, infrared backlighting was adjusted in the software while the probe was connected to provide real-time insight into the operation. While several levels of intensity were tested during this process, we have included the measurements that we believe are most illustrative to display the results. Below is an image displaying the voltage operation across the left infrared diode cluster with light intensity set to 0 percent (meaning off).



MSO-X 3014A, MY54100398: Thu May 27 12:30:49 2021

1 10.0V/ 2 3 4 0.0s 500.0s/ Auto? 1 -65.0V

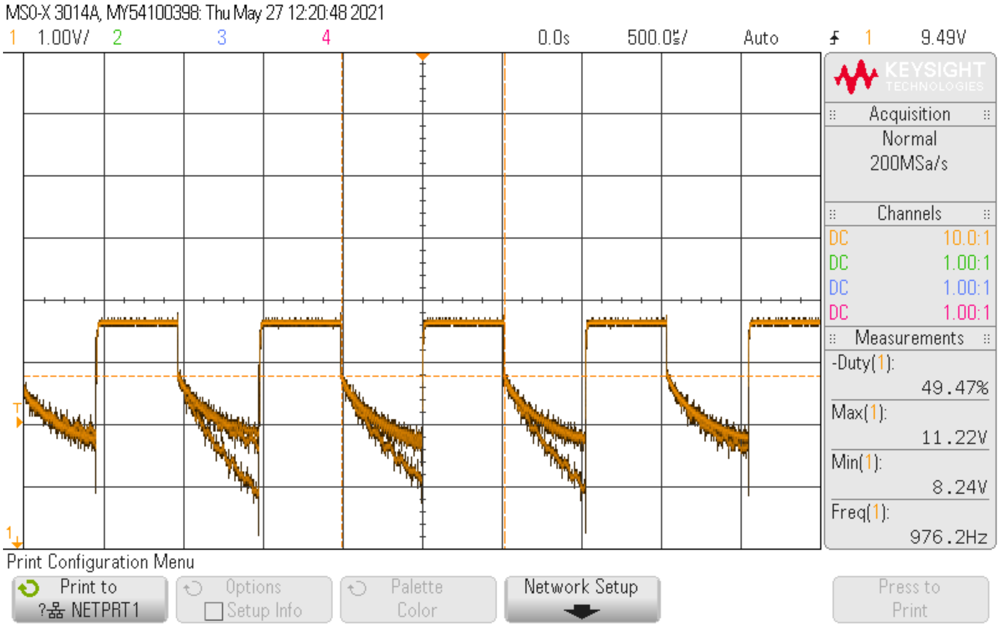
Acquisition
Normal
200MSa/s

Channels
DC 10.0:1
DC 1.00:1
DC 1.00:1
DC 1.00:1

Measurements
-Duty(1):
Low signal
Max(1):
600mV
Min(1):
-600mV
Freq(1):
Low signal

Figure 5-1 Measurement of 0% intensity over IR diode cluster

As can be seen, the scope is unable to measure the duty cycle, while also displaying trivial maximum and minimal values. This behavior is completely expected and persisted after repeatedly increasing and then decreasing the intensity back to 0. Next, we provide an image taken from the probe with intensity set to 50 percent.



MSO-X 3014A, MY54100398: Thu May 27 12:20:48 2021

1 1.00V/ 2 3 4 0.0s 500.0s/ Auto 1 9.49V

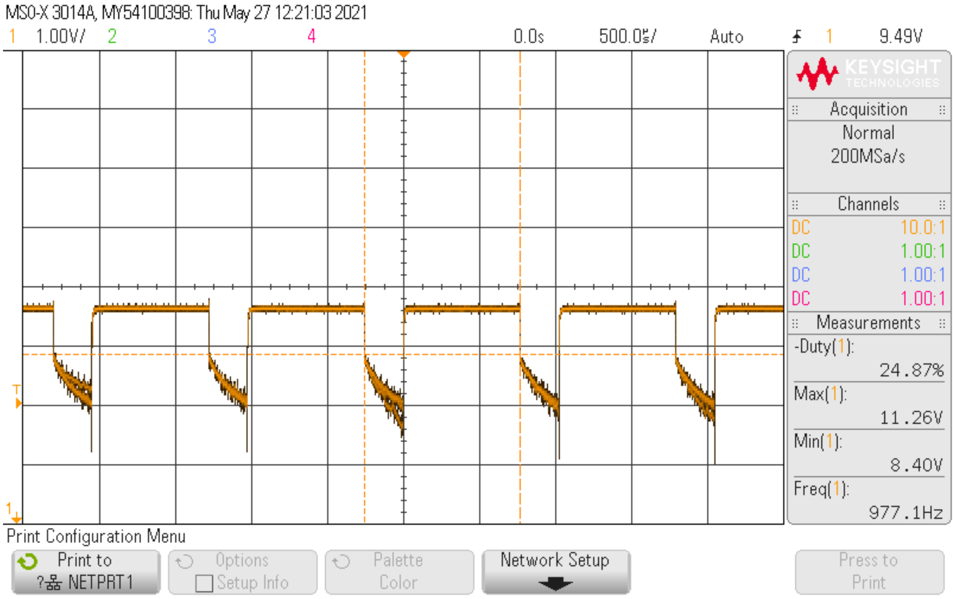
Acquisition
Normal
200MSa/s

Channels
DC 10.0:1
DC 1.00:1
DC 1.00:1
DC 1.00:1

Measurements
-Duty(1):
49.47%
Max(1):
11.22V
Min(1):
8.24V
Freq(1):
976.2Hz

Figure 5-2 Measurement of 50% intensity over IR diode cluster

The scope informs us of a duty cycle at approximately 50 percent, which also can be gleaned from the image, as well as maximum and minimal values respectively at 11.22 volt and 8.24 volt. Accounting for the slopes that occur at the falling edge of each cycle is somewhat more difficult. While we were expecting some non-linear behavior given the non-linearity of diodes, this seemingly noisy and gradual curve does not match our expectations of a square signal. While this is tolerable, considering that it seemingly has no effect on the functionality, further clarification of this phenomenon would likely be investigated given more time. Next, we look at the signal operating at 75 percent and compare it to the previous analysis.



MSO-X 3014A, MY54100398: Thu May 27 12:21:03 2021

1 1.00V/ 2 3 4 0.0s 500.0ᵘ/ Auto 1 9.49V

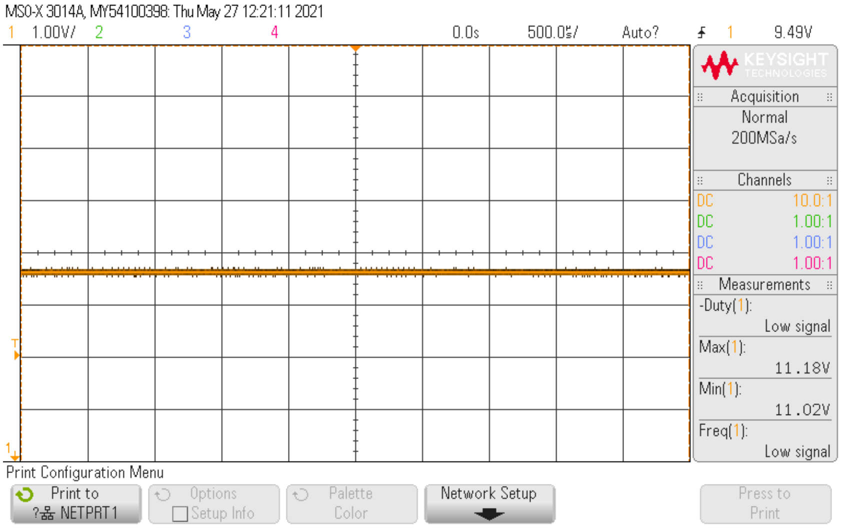
Acquisition
Normal
200MSa/s

Channels
DC 10.0:1
DC 1.00:1
DC 1.00:1
DC 1.00:1

Measurements
-Duty(1): 24.87%
Max(1): 11.26V
Min(1): 8.40V
Freq(1): 977.1Hz

Figure 5-3 Measurement of 75% intensity over IR diode cluster

Given that 75 percent intensity would imply a duty cycle of 25 percent, we find the result of 24.87 percent quite satisfactory. Finally, we present a measurement of the probe with intensity set to 100 percent.



MSO-X 3014A, MY54100398: Thu May 27 12:21:11 2021

1 1.00V/ 2 3 4 0.0s 500.0ᵘ/ Auto? 1 9.49V

Acquisition
Normal
200MSa/s

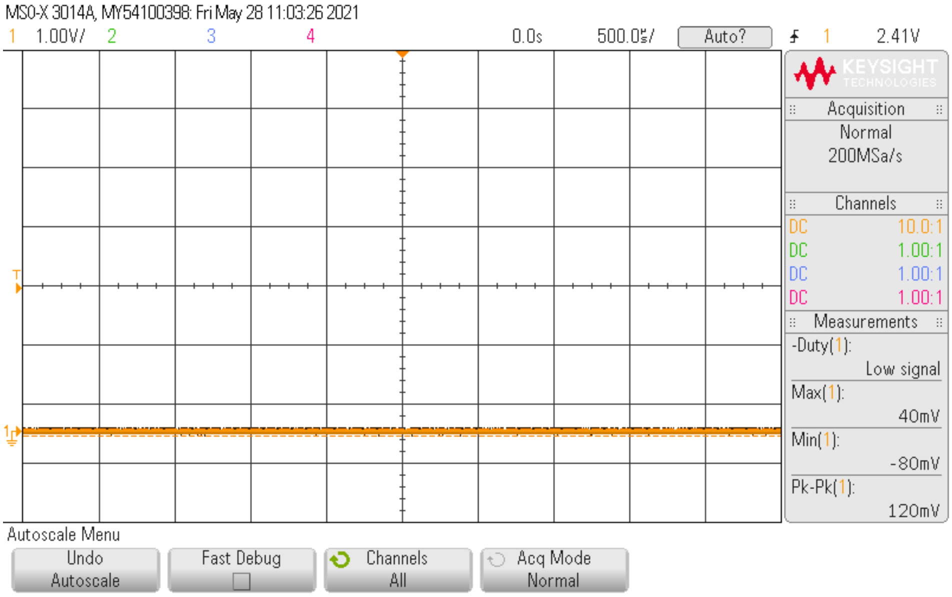
Channels
DC 10.0:1
DC 1.00:1
DC 1.00:1
DC 1.00:1

Measurements
-Duty(1): Low signal
Max(1): 11.18V
Min(1): 11.02V
Freq(1): Low signal

Figure 5-4Measurement of 100% intensity over IR diode cluster

Like the first in this series of image, we can see that the duty cycle is unreadable. In this case, however, the voltage values are the same as in preceding three measurements, meaning that the transistors are in fact letting the signal through. As such, we now have a PWM signal with a duty cycle of 0: Which is just direct current and wholly according to expectations. In conclusion, we would like to state the operation of the PWM across the diodes works as intended, with the oddity of the slopes still requiring further investigation for an even more thorough verification process.

Next, we use the same principal method to verify that the Arduino outputs the correct PWM signals. Below are images with intensity set to 0, 50, 75, and 99 percent respectively.



MSO-X 3014A, MY54100398: Fri May 28 11:03:26 2021

1 1.00V/ 2 3 4 0.0s 500.0µs/ Auto? 1 2.41V

Acquisition Normal 200MSa/s

Channels: DC 10.0:1, DC 1.00:1, DC 1.00:1, DC 1.00:1

Measurements

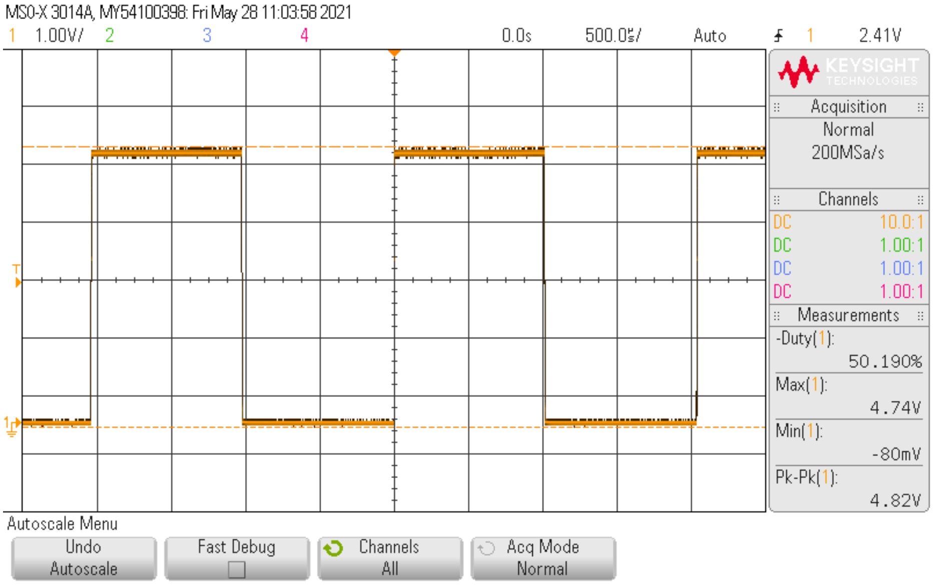
-Duty(1): Low signal

Max(1): 40mV

Min(1): -80mV

Pk-Pk(1): 120mV

Figure 5-5 Arduino PWM at 0% intensity



MSO-X 3014A, MY54100398: Fri May 28 11:03:58 2021

1 1.00V/ 2 3 4 0.0s 500.0µs/ Auto 1 2.41V

Acquisition Normal 200MSa/s

Channels: DC 10.0:1, DC 1.00:1, DC 1.00:1, DC 1.00:1

Measurements

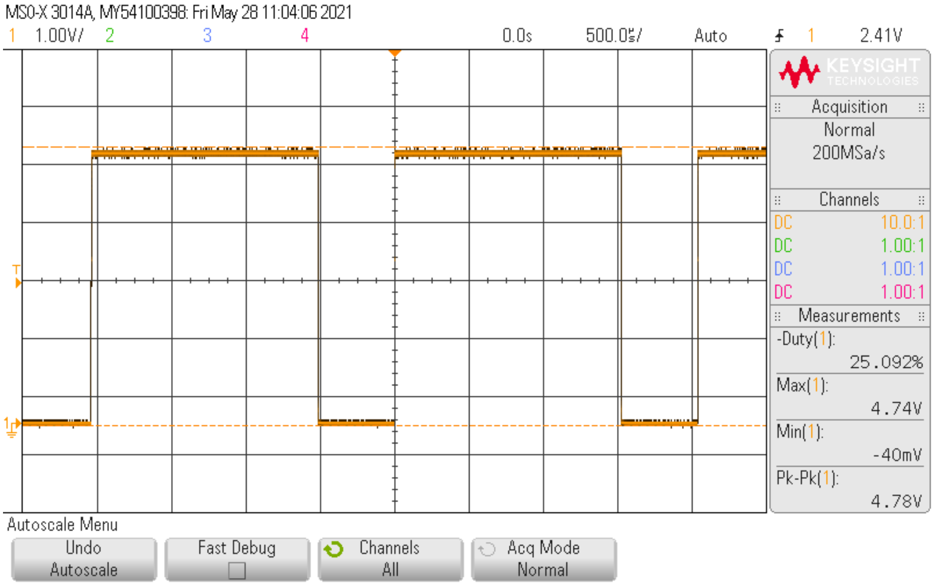
-Duty(1): 50.190%

Max(1): 4.74V

Min(1): -80mV

Pk-Pk(1): 4.82V

Figure 5-6 Arduino PWM at 50% intensity



MSO-X 3014A, MY54100398: Fri May 28 11:04:06 2021

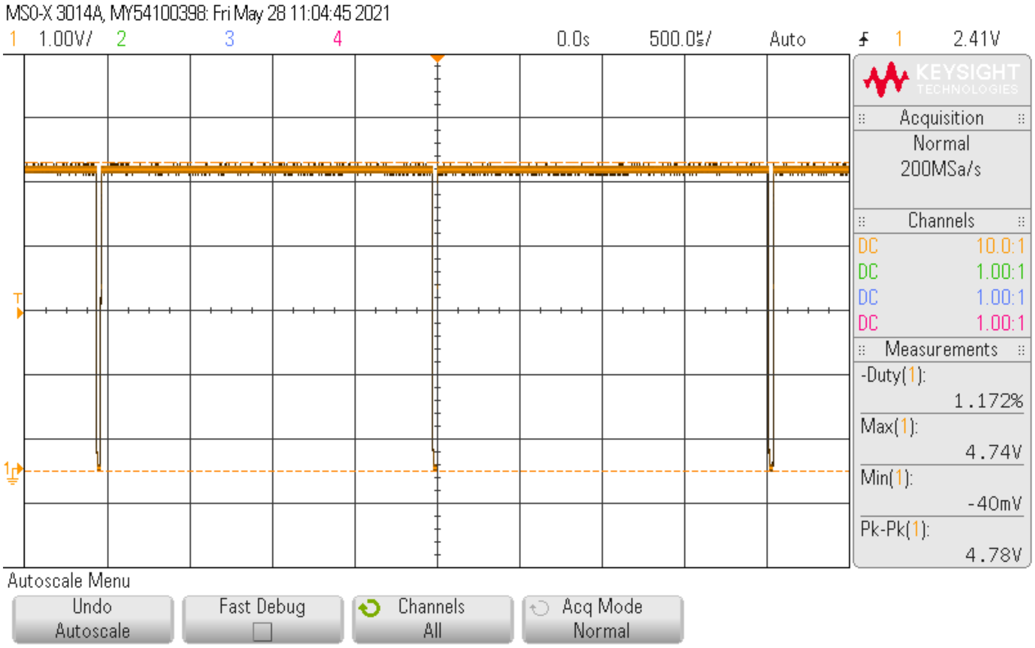
1 1.00V/ 2 3 4 0.0s 500.0㎲/ Auto 1 2.41V

Acquisition
Normal
200MSa/s

Channels
DC 10.0:1
DC 1.00:1
DC 1.00:1
DC 1.00:1

Measurements
-Duty(1): 25.092%
Max(1): 4.74V
Min(1): -40mV
Pk-Pk(1): 4.78V

Figure 5-7 Arduino PWM at 75% intensity



MSO-X 3014A, MY54100398: Fri May 28 11:04:45 2021

1 1.00V/ 2 3 4 0.0s 500.0㎲/ Auto 1 2.41V

Acquisition
Normal
200MSa/s

Channels
DC 10.0:1
DC 1.00:1
DC 1.00:1
DC 1.00:1

Measurements
-Duty(1): 1.172%
Max(1): 4.74V
Min(1): -40mV
Pk-Pk(1): 4.78V

Figure 5-8 Arduino PWM at 99% intensity

Here we can see the same patterns as before: 0 intensity is no signal, 50 and 75 percent intensities matches their respective 50 and 75 percent duty cycles, while 99 percent is practically direct current with minor "dips" in the signal. At 100 percent intensity, the signal operated wholly as direct current.

6 Discussion

In this section we will review and reflect upon the system in its current state, as well as providing accounts for the various design and implementation choices. Like section 4, we will view the system as a whole before delving more extensively into each component.

6.1 Product overview

Below is a diagram illustrating the final system using a higher level of abstraction in terms of components.

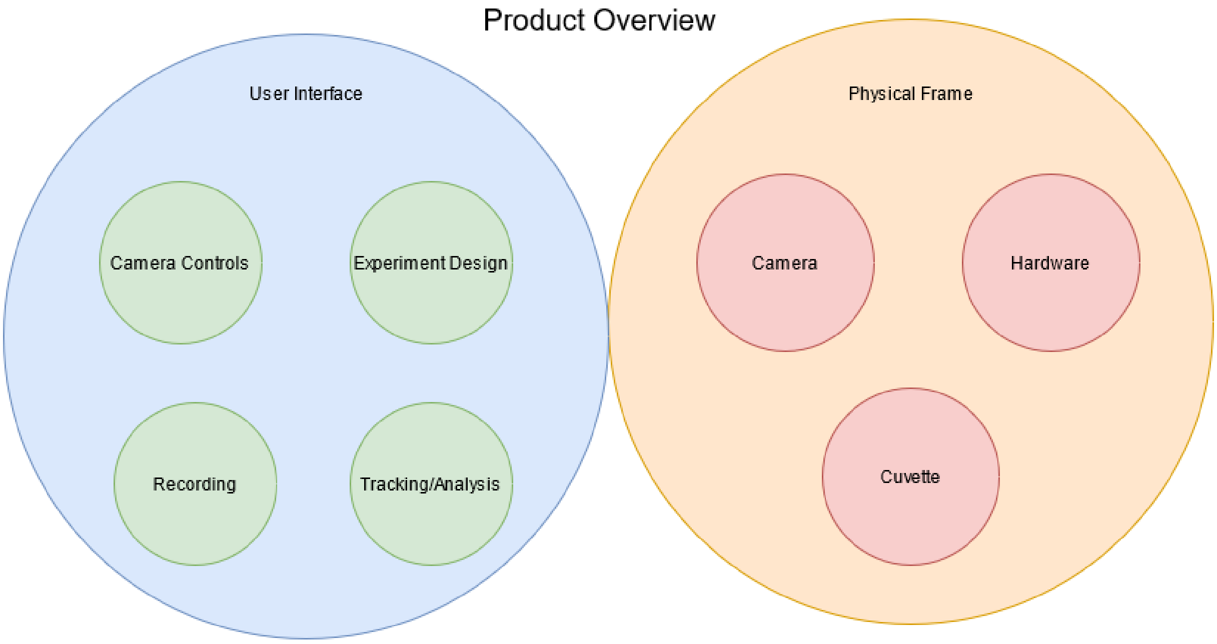


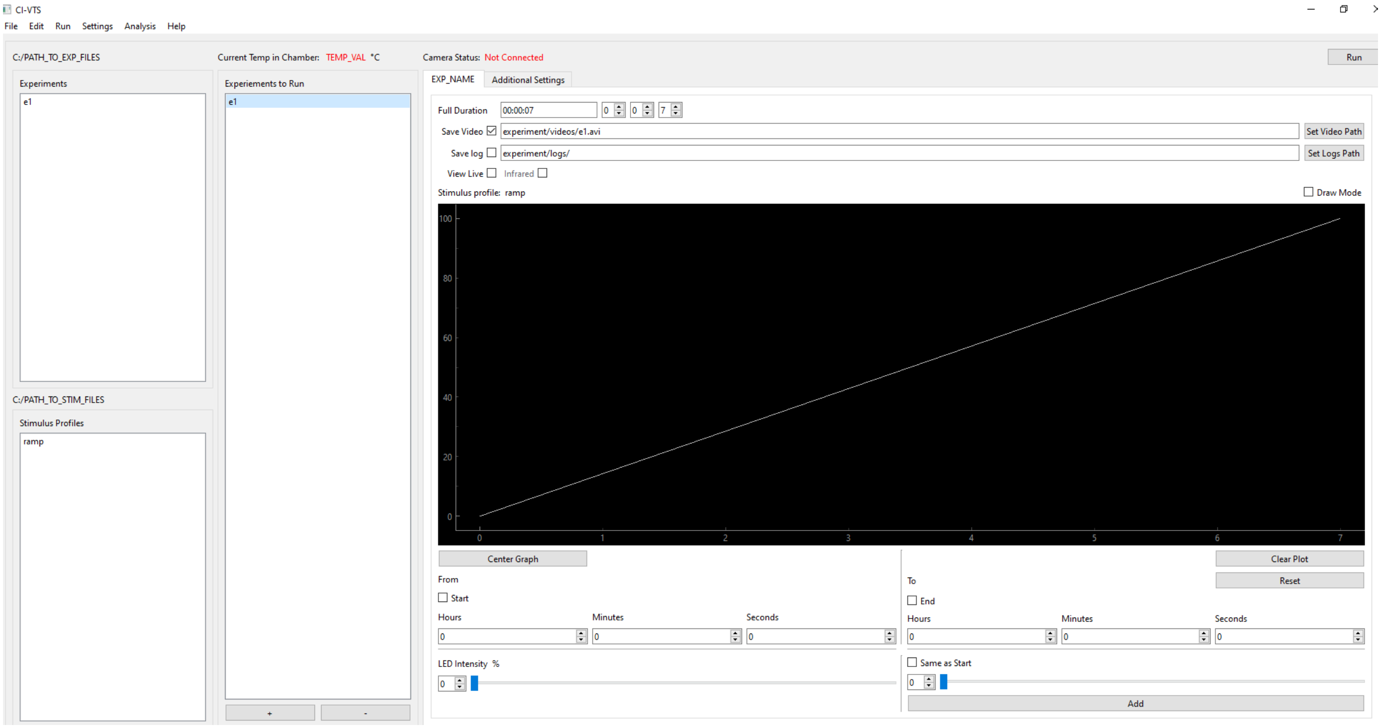
Figure 6-1 Product overview

The user interface acts the as the "container" and gateway of the user's interaction with the camera, experiment design, video recording and the tracking features. The physical frame does indeed act as an actual container to all the physical components. While each subsequent section does not serve to give specific technical accounts of these component, they will be discussed and reflected upon throughout to provide the reader with insight into our endeavors.

6.2 Design choices and user experience

We start this sub-section with something of a disclaimer: Readers should note that the term "design" is here reserved for the design of the user experience: Layout of user interface components, intended workflow etc. and should not be confused with the overall design of the entire system as otherwise described throughout this document.

It was stressed from the onset of this project that the user interface is a critical feature of this product. While there were no expectations to provide a commercial grade experience (typically delivered by front-end developers and user experience designers), an important specification was the system be fully interactable through graphical means, which implies that some sort of user experience design would be involved. None of the group members on this project have any significant experience with such designing, and as such we relied heavily upon feedback from Sars to ensure that the product would look and feel according to their tastes. Given frequent communications, where ideas and images were shared and critiqued, a design was produced early in the project development period. Completing the design early was a highly deliberate choice: It allowed us to clearly visualize the features that our contractors would be working with, while also allowing them to give early feedback and thus provide clarifications on how they would like the software to operate. Below is an image taken as a screenshot of the main window of the application.



C:/PATH_TO_EXP_FILES

Current Temp in Chamber: TEMP_VAL °C

Camera Status: Not Connected

Experiments: e1

Experiements to Run: e1

C:/PATH_TO_STIM_FILES

Stimulus Profiles: ramp

Full Duration 00:00:07 0 0 7

Save Video ☑ experiment/videos/e1.avi

Save log ☐ experiment/logs/

View Live ☐ Infrared ☐

Stimulus profile: ramp

☐ Draw Mode

From

☐ Start

Hours 0 Minutes 0 Seconds 0

To

☐ End

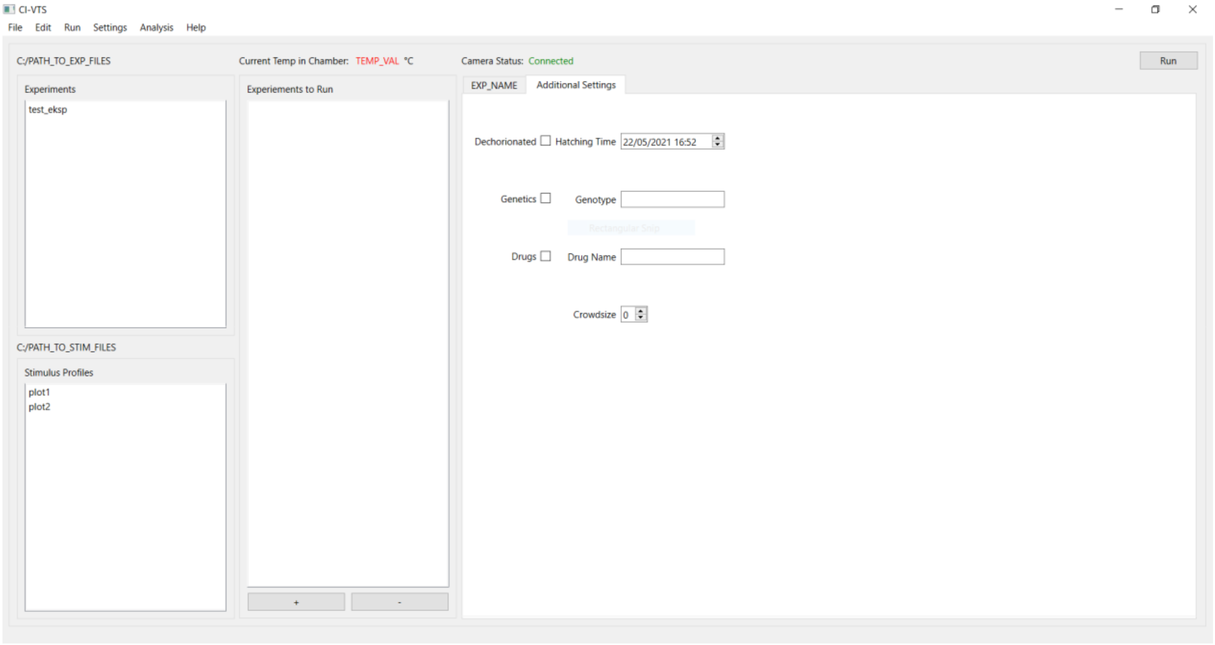
Hours 0 Minutes 0 Seconds 0

LED Intensity % 0

☐ Same as Start 0

Figure 6-2 main window of application

The goal behind this design is to provide a "classical desktop" feel, with clearly identifiable functionality that can be navigated intuitively. Another essential idea was to uphold a "left-to-right" workflow: Users can start by selecting their experiments, optionally select a stimulus profile, and then proceed to make any modifications to the experiment before selecting run in the top right corner. In the image above we can see a user that has selected an experiment named "e1" and added a stimulus profile "ramp" (for a technical discussion of these, see section 4.6). By keeping all the central features of experiment modification in clear view, users should hopefully be able to make rapid customization of experiments. Here it is prudent to mention that some of the more detailed experiment configurations were moved to a separate tab. The image below shows how the main window appears if this tab is selected.



C:/PATH_TO_EXP_FILES

Current Temp in Chamber: TEMP_VAL °C

Camera Status: Connected

test_eksp

Dechorionated □ Hatching Time 22/05/2021 16:52

Genetics □ Genotype

Drugs □ Drug Name

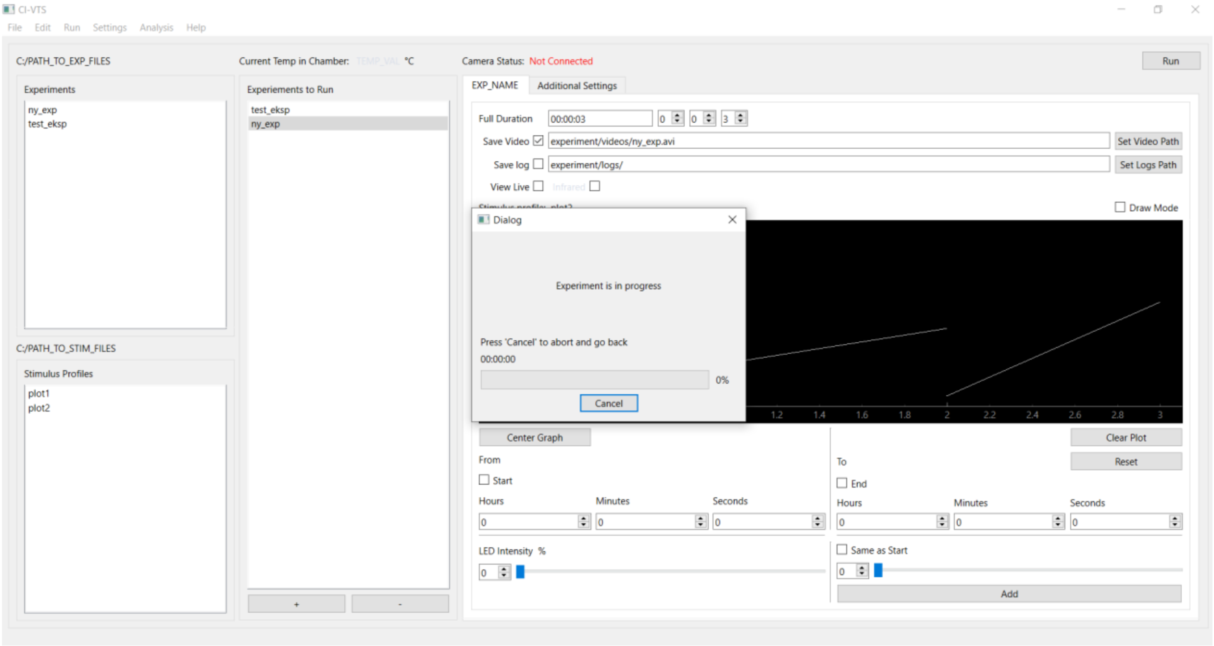
Crowdsize 0

C:/PATH_TO_STIM_FILES

plot1
plot2

Figure 6-3 Main window with additional settings

It should also be noted that some elements in the main window are not interactable: For instance, there were some brief discussions within the group to add a logging feature which would user interactions and write all observations to a file. While this feature never made it out of the "casual conversation stage", adding the necessary user interface components was merely a matter of replicating the same design as for the setting the video path, so it was added on to avoid reworking the design later on. This also applies to certain tabs in the menu on the top left: For instance, the "help" drop-down does, unfortunately, provide no such thing. However, there are no plans to remove them either: They will remain as shells for future feature development, a decision that is at the sole discretion of our contractor. Finally, we direct our attention to a feature of the user interface that was highly requested after the delivery of the initial prototype: A progress indicator for running experiments. Once again, a screenshot is included below.



Full Duration 00:00:03 0 0 3

Save Video ☑ experiment/videos/ny_exp.avi

Save log □ experiment/logs/

View Live □ Infrared □

Experiment is in progress

Press 'Cancel' to abort and go back

00:00:00

0%

□ Start

□ End

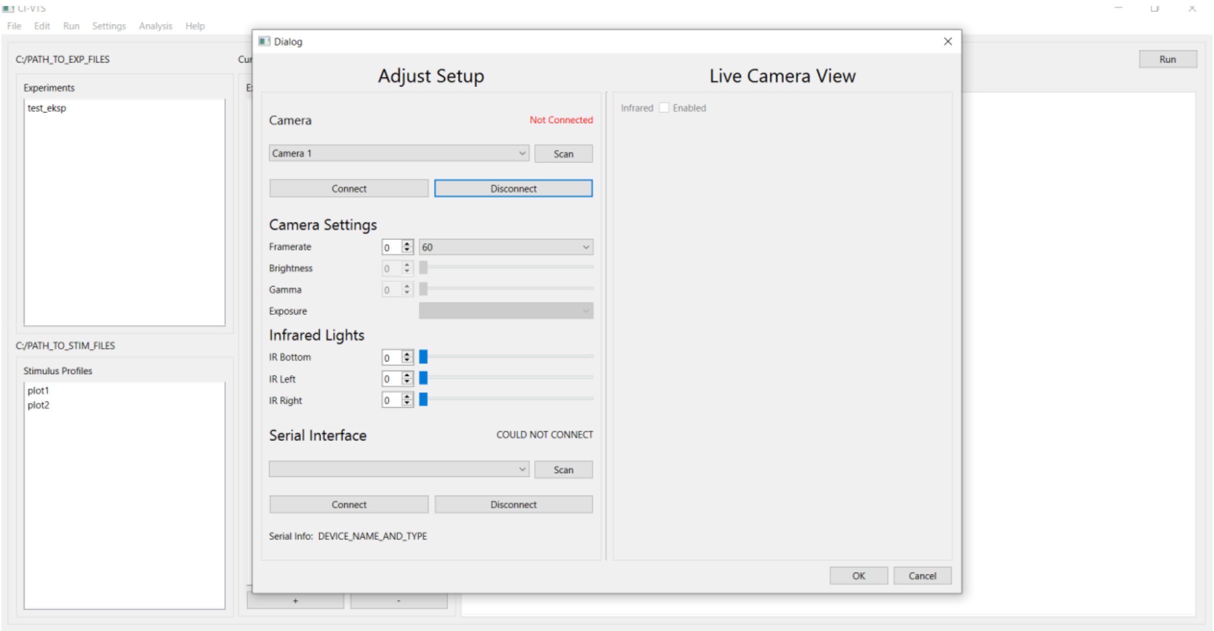
Hours 0 Minutes 0 Seconds 0

□ Same as Start

Figure 6-4 Progress bar of the experiment

As can be seen in the center of the image, a popup dialog with a duration field and a progress bar will appear when a user runs an experiment. Upon completion, a user will be notified in the text field that the experiment is indeed done.

Whereas the main window allows users to perform experimentation design and running, there are two other windows that also comprise the user interface: The “setup” window and the “analysis” window. A screenshot of the former is shown below.



Camera Not Connected

Camera 1

Framerate 0 60

Brightness 0

Gamma 0

IR Bottom 0

IR Left 0

IR Right 0

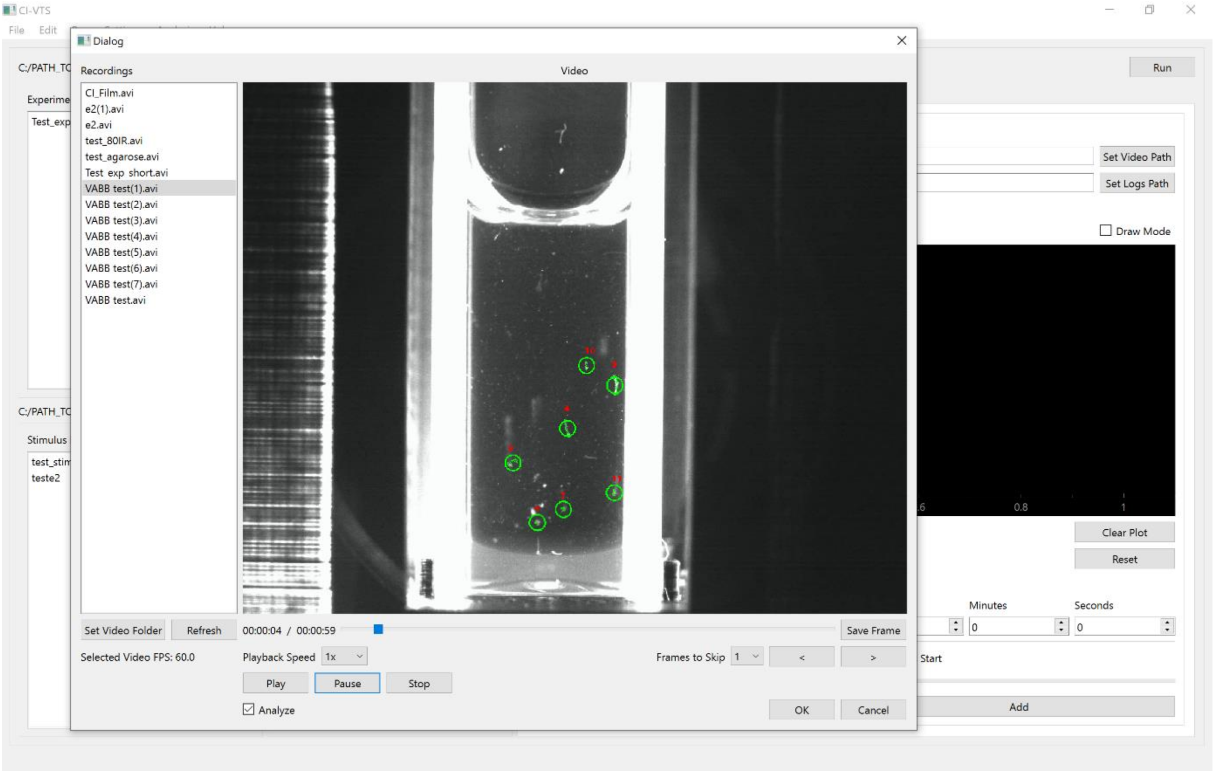
Serial Interface COULD NOT CONNECT

Serial Info: DEVICE_NAME_AND_TYPE

Infrared □ Enabled

Figure 6-5 Setup window

The setup menu overlays the main window and provides users with the ability to adjust the camera settings as well as the infrared backlighting. On the left is a live camera view (which in this particular image is empty because no camera is connected at the time of taking this screenshot), enabling users to see how altering the settings will affect the footage. It is also possible to select between multiple cameras. Next, we look at the analysis window.



00:00:04 / 00:00:59

Selected Video FPS: 60.0

Playback Speed 1x

Frames to Skip 1

☑ Analyze

Figure 6-6 Analysis window

This window provides a list of recordings that appears in the left column, while also providing with an interactive video player. Users can also perform the tracking analysis by checking the analysis box and then playing the video, which provides a visualized overlay on the video. Here we feel it worth mentioning that this checkbox design, at the behest of our contractor, is in the process of being altered to a more streamlined approach: A button, that when pressed runs a full analysis on the entire video file, much in the same way as running an experiment.

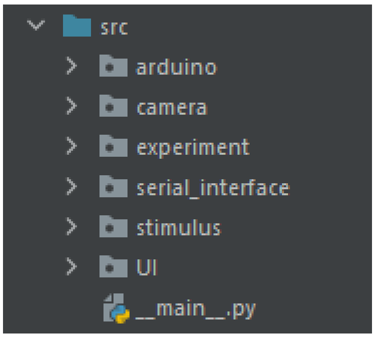
We are under the impression that our contractor is, in general, happy with the design choices we made. Very few alterations were made throughout the development process, and such we consider it a successful part of the system.

6.3 The software

This section will discuss the final version of the software, with added emphasis on its organization, structure, architecture, maintainability and future development considerations.

6.3.1 Organization and structure

The software was structured in accordance with its components as illustrated in section 6.1, with some minor adjustments. The idea was to modularize as much as possible with the goal of simplifying development abstractions. Below is a screenshot, taken from a development tool, that illustrates the hierarchy of the software components.



```
src
  arduino
  camera
  experiment
  serial_interface
  stimulus
  UI
  __main__.py
```

Figure 6-7 Software structure

These folders, which in fact are defined as python packages, each comprise a separate part of the software in the system (note that the “arduino”-folder contains program code for uploading to the Arduino and is decoupled from the rest of the system). Consider the figure in section 6.1, it is prudent to note that “experimentation design” has been divided into “stimulus” and “experiment”, with the reason being to enhance the modularity mentioned previously. This has been highly beneficial in terms of working with parts separately and eased the use of version control since it inherently avoids more merge conflicts. For instance, a developer working with the camera controls can completely disregard another developer working with the experiment configurations. This has proved very fruitful to the development process at large, enabling flexible coding practices.

6.3.2 Components and inner workings

Whereas software organization is good for getting an overview of the larger abstractions within the software, this section delves a little deeper with the aim to give further insights. That is, a more extensive account of its various components, like how modules and classes behave and interact. Below is a component diagram made, somewhat loosely, according to the principles of the Unified Modeling Language, most often abbreviated as UML (What is UML, 2021), which provides an overview of the various software components and how they interact in the system.

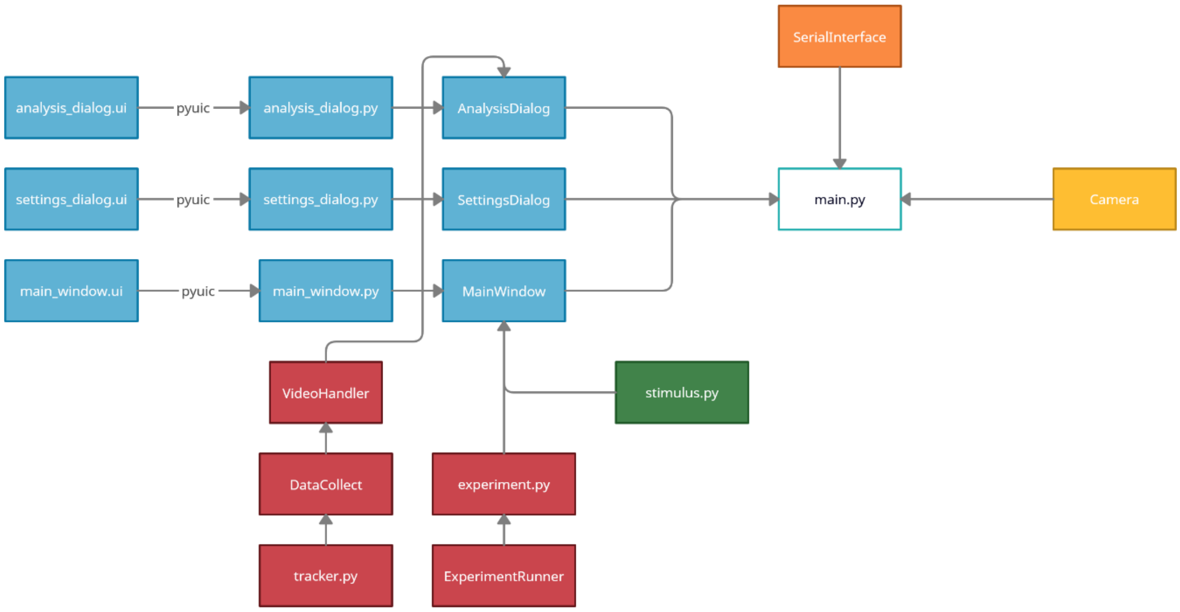


Figure 6-8 Software component diagram

Note that the arrows indicate direction of usage: An arrow pointing from one component to another means that the latter makes direct usage of the former, though not the other way around. The color coding, though chosen arbitrarily, have been selected in accordance with the organization discussed in the previous sub-section. As such, blue is the user interface, red is experimentation, green is stimulus management, yellow is the camera, orange is the serial interface, and finally the main application is white. The latter of these can be found at the crux of the software and handles proper initialization and, of course, running the application. During the initialization process, single instances of the classes that handle the camera and the serial interface are passed to the interface components that require them. That way, a single running instance in memory is shared among several user interface components. The user interface itself originates from .ui-files, which are made and generated by Qt-Designer (see section 4.5 for more details on Qt-Designer). These are then compiled to python code using the command line tool pyuic (also see section 4.5) and then inherited by their respective developer-defined classes which in turn adds functionality and interaction logic on top of the user interface, making use of the single instances of the camera and serial interface passed down from main.py as well as the modules experiment.py and stimulus.py. As such, we separate complicated logic pertaining to the features of the software into separate modules and classes and use the user interface as wrappers around them. These wrappers also implement the user's interaction with the system and as such serve as data entry points. For instance, a user plotting some stimulus values and running will result in those values being passed through MainWindow and down to experiment.py and ExperimentRunner. Since the user interface is, quite naturally, the preferred way of interacting with the system, this is true for all components of the system. An advantage of this approach, as opposed to implementing feature logic directly into the user interface wrappers, is that it increases modularity and makes code more maintainable through less coupling. While we are in general happy with the architecture of the software, we take self-critique in the way that there are cases where a clearer distinction between the use of modules and classes are prudent. For instance, the stimulus.py, which handles loading and saving of stimulus data, is comprised entirely of functions, while its counterpart experiment.py contains both standalone functions and the class ExperimentRunner. Further time investment in this project would likely result in an overhaul and standardization of these code abstractions. Nevertheless, we feel that the overall structure is satisfactory and in line with sound development practices.

6.3.3 Tracking

When we started working on the tracking software, we quickly realized that it would be problematic to get consistent data collection from the movements of Ciona Intestinalis inside of the cuvette. Some of the obstacles was to distinguish contaminants and bubbles in the water from the actual specimen. The size of those objects are similar and the shape can be similar as well, depending on the angle of the animal. We decided to use frame difference as a base image processing. We took a number of previous frames and compared the average to the current frame. Everything that was the same got removed and the rest remained. This was then used to get the contours of each animal and given different IDs based on a few parameters.

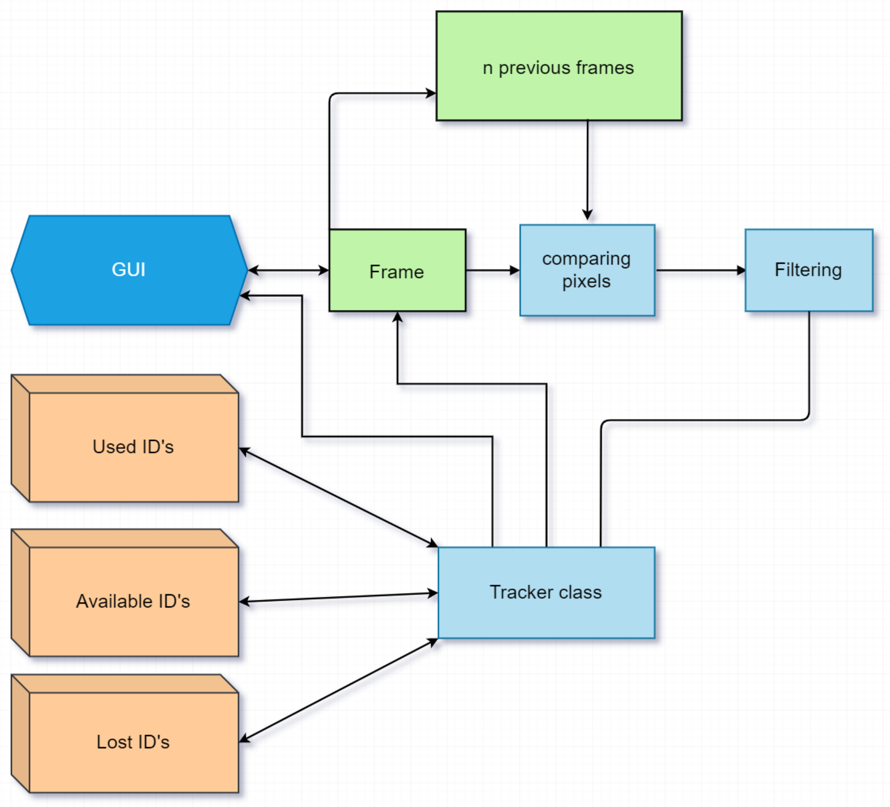


Figure 6-9 Tracking diagram

As can be seen from the flowchart the GUI sends a frame into the DataCollect class. There the frame is saved so that it can be used later for comparison. The pixels of the current frame and a number of previous frames are then compared to detect differences. This is then filtered and sent into a class called Tracker. In the tracker class we compare center points and size of each contour to see if they should be given a new ID from available ID's or if its already an ID assigned to that exact point, or if there was a previous ID assigned in near proximity that was lost. Each ID with location is then sent back to DataCollect class which in turn sends this data together with information about current frame to the GUI. In the GUI this data is printed to a JSON file.

6.4 Thermometer (incomplete feature)

During the development process, it came to our attention that our contractor desired temperature readouts from inside the physical frame. While this feature was investigated, it unfortunately has not made it into the final product. The intention was to expand the serial communication interface to allow a two-way communication stream, where the Arduino could receive stimulus values while also sending temperature readouts to be displayed in the user interface. We were supplied with the simple digital thermometer, namely the VMA324 DS18b20 (Temperature Probe - DS18b20, 2021), which can handle temperature readings from negative 55 degrees Celsius to 125 degrees Celsius with 9 to 12 bits of resolution. Considering that the product is intended to be situated in a lab environment, this seems more than sufficient. To handle the input of this probe, the library DallasTemperature was selected as it explicitly provides support for the DS18b20 (DallasTemperature, 2021). However, due to time constraints and the continuous effort to support features of higher priority, temperature readouts have not made into the product yet.

6.5 Current overall state of the system

The software is currently capable of performing all the tasks described as requirements in chapter 2: Record videos, control camera and stimulus, perform visualized tracking, and run experiments, though the robustness of these features is yet to be put fully to the test. Some, like the tracking feature, were completed recently and as such it is still in its initial phase. The software component of the product can therefore only be described as in an early alpha stage, though for the purpose of handing over development to our contractor we consider this satisfactory. Of course, we would very much enjoy seeing the software reach full maturity, but that seems to be off the horizon for us as students working on the project.

6.6 Collaboration, communication, and version control

As suggested throughout the text, collaboration and communication both within the group and with Sars have been absolutely critical to the success of this project. Early on, communications were streamlined through the use of Slack (Slack, 2021). While this may seems like a minor thing to comment on, we believe that having this platform available has made a huge difference: Fast and easy conveying of ideas and thoughts have made development and planning much easier. As such it is only fair to assume that productivity have been boosted by this.

While slack was used for communicating with our contacts at Sars, we the students used git and GitHub extensively to keep track of development within the project. This is especially prevalent on the software side of things, though all relevant files required to reproduce the system are available on the GitHub page of our repository (see appendix B). The project management feature on GitHub was used to keep track of tasks, and, as mentioned in section 3.1, development was carried out in the spirit of gitflow. This also turned out to be a successful endeavor, with only minor setbacks in terms of merge conflicts in the code base.

6.7 Experimentation design and usefulness

Experimentation design was never explicitly mentioned as part of the initial requirements listed in chapter 2 and was left as a more implicit factor (after all, the product is intended for research). This observation, however, eluded the authors for a little while as we merely intended on implementing the hard requirements that were given to us. When it evidently became clear that some mechanism of tying information to stimulus and recording was a highly sought-after feature, we came up with the design described in section 4.6. Using JSON, which is ubiquitous in modern data transfer and storage, the system handles the experimentation settings in a straight-forward manner by simply writing data to a file with descriptive fields. These files can then be loaded on demand by the software to reprepare experiments at the user’s discretion, making it easy to switch between different experiments. Running the analysis functionality on recorded videos also produces JSON-files that contain simplistic tracking data. While there are no features yet that deal with the illustration of this data, our contractor can readily implement data visualization of this tracking data as they see fit when development is passed passer over to them. In summary, the product is useful for viewing and recording tracking data, but data representation is far from mature.

6.8 Physical frame

When designing the physical frame, we focused on modularity and size. The modularity is important so that the contractor could make small changes to the frame without printing out large modules. It is also easier to make changes while prototyping. The size is important so that the entire system could fit inside an incubator.

The first version is illustrated and detailed in section 3.2. This was a simple sketch of our starting idea. Since then we have made a total redesign. We have however kept our core design parameters of modularity and size.

The thoughts behind the second version are that we would have the camera inside its own casing, connecting it to the main chamber where we would have the background lighting and the cuvette. The oval wall between the main chamber and the camera housing is designed to hold the infrared lens. Here we placed reflectors with IR LEDs along the wall inside the main chamber. The idea was that we could then control each LED individually to get the best possible light for our image. We would also have one LED placed below the cuvette. This did however not get us the desired result. When controlling each LED individually we did not see a big change in picture quality. And the LED that was placed at the bottom did not do anything to help. This is because there was not enough directionality or intensity of the light. The placement of the cuvette was too cumbersome and required steady hands. We had to find a new way to place the cuvette, and we needed more control of the background light. The camera placement was still good.

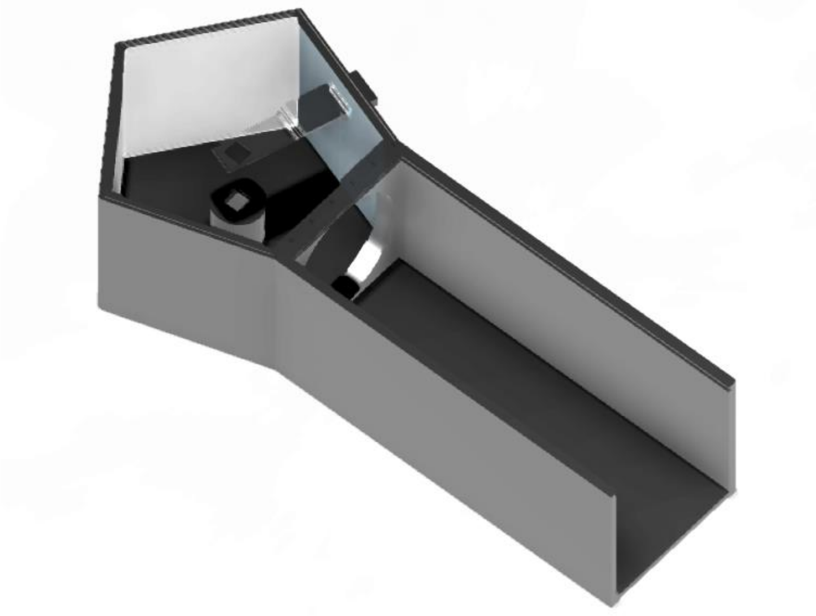


Figure 6-10 Casing version 2

The third version, which can be seen below, got us a lot closer to our goal. We reduced the size of the main chamber and created a separate light module that can be placed around the cuvette. We also added a slot in the lid for the main chamber that can house a COB LED for stimulus. The camera chamber was mostly unchanged, the only difference is that we added a sliding lid that was intended to lightproof the camera chamber. This version was sent to the Sars center for testing. They were overall happy with the design and functionality of the system; they did however want some changes to part of the design. Most notably, they wished for a redesign of the cuvette holder as it was too difficult to put the cuvette in place.

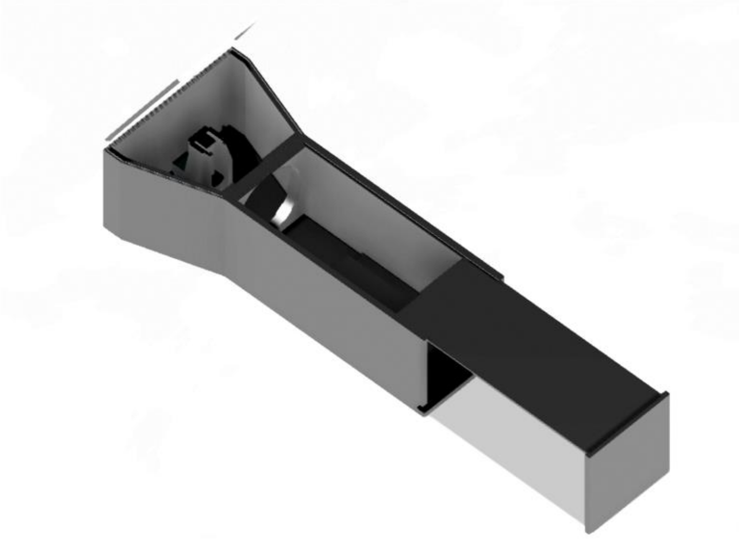


Figure 6-11 Casing version 3

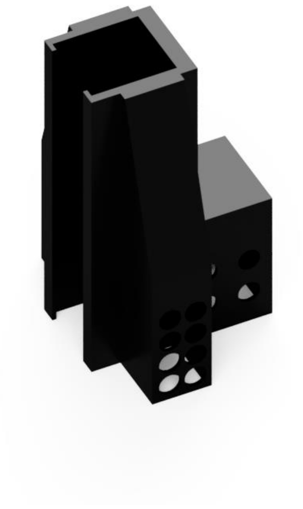


Figure 6-12 Light module

The light module that we designed can be seen in the picture to the left. This houses 24 IR LEDs that is divided into 3 zones. Each zone can be individually adjusted through our GUI. The walls on the inside are 0.4 mm thick where the cuvette is sitting in between. This is to diffuse the light and make it more evenly distributed along the side of the cuvette. After testing on live specimens this module was considered a success and has been unchanged since then.

Our final version takes the best parts of the previous models and incorporates it into a modular, functional and compact design that the scientists at Sars center can use for their experiments. It is also designed to be easily upgraded if they need additional features in the future. We have added spacers so that it can be used with a wide variety of lenses and cameras. We have designed a container for all the electronics and wiring in the back of the main chamber. This is connected with two 8mm screws, washers and nuts. We have redesigned most of the system so that it is made out of multiple smaller pars, this is to increase modularity and ease of repairs or upgrades.

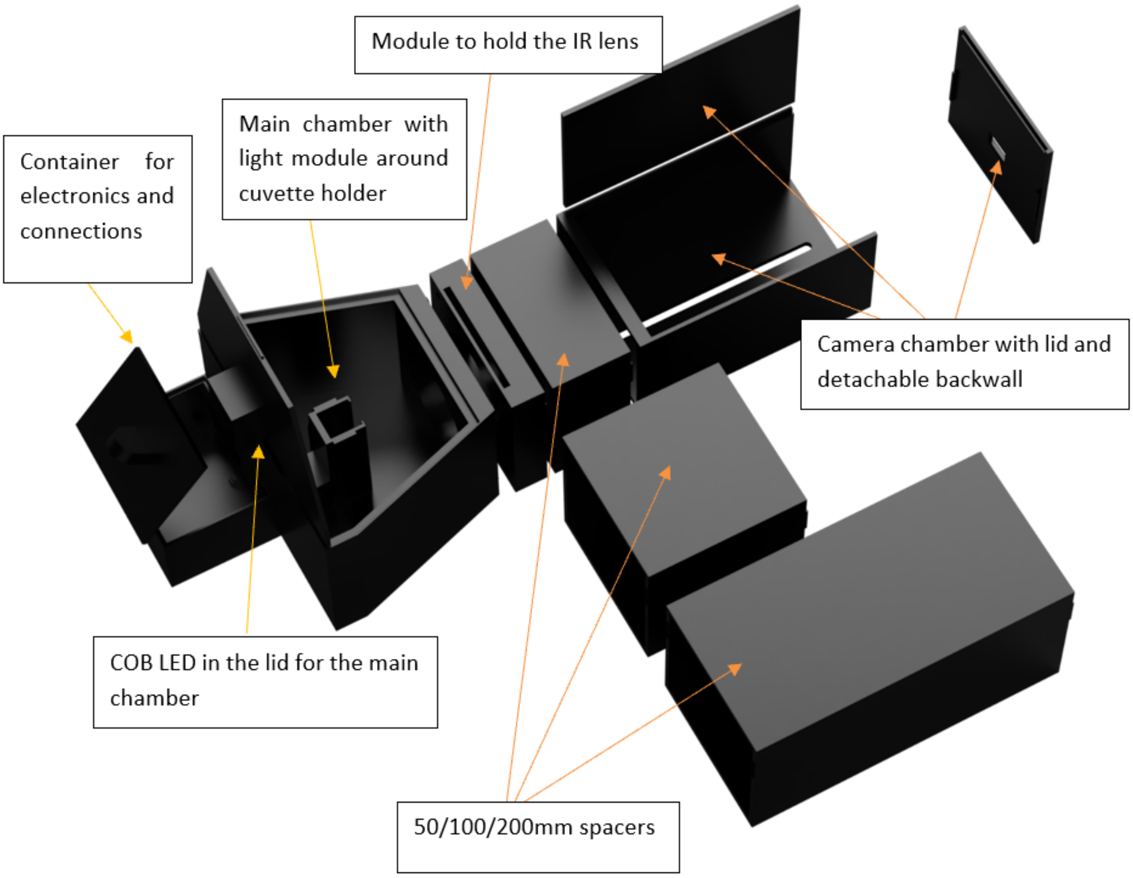

Module to hold the IR lens
Main chamber with light module around cuvette holder
Container for electronics and connections
Camera chamber with lid and detachable backwall
COB LED in the lid for the main chamber
50/100/200mm spacers

Figure 6-13 Casing final version

The box located at the back of the main chamber is as mentioned the container for electronics. It allows wires to go directly from the main chamber and connect it to the circuit. On the side there is a slot where we can connect the USB to the Arduino and a power supply for the circuit.

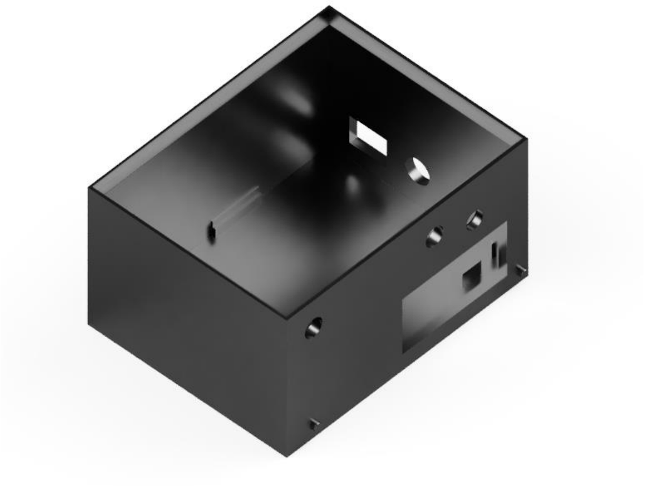


Figure 6-14 Container for electronics

The location of the lens is the same, but we separated it into a standalone module. This is because the lens should be easy to remove or replace. The lens is also a very expensive component and it is in everyone's best interests that it can easily be removed for safekeeping. This also gives the opportunity to print out a new module if another lens with different measurements is implemented.

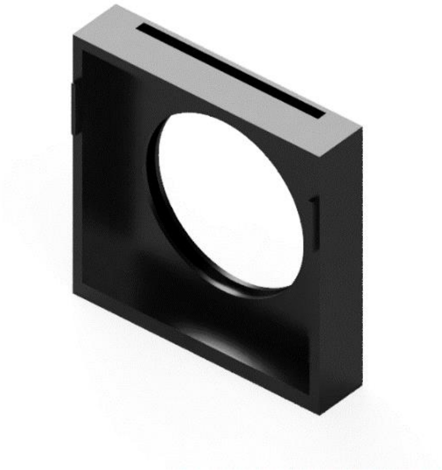


Figure 6-15 IR filter holder

The main chamber was left mostly unchanged since the third version. The only changes was that we added compatibility with the electronics container and some minor changes to the cuvette holder. The cuvette holder was enlarged and reshaped to make it easier to place the cuvette.

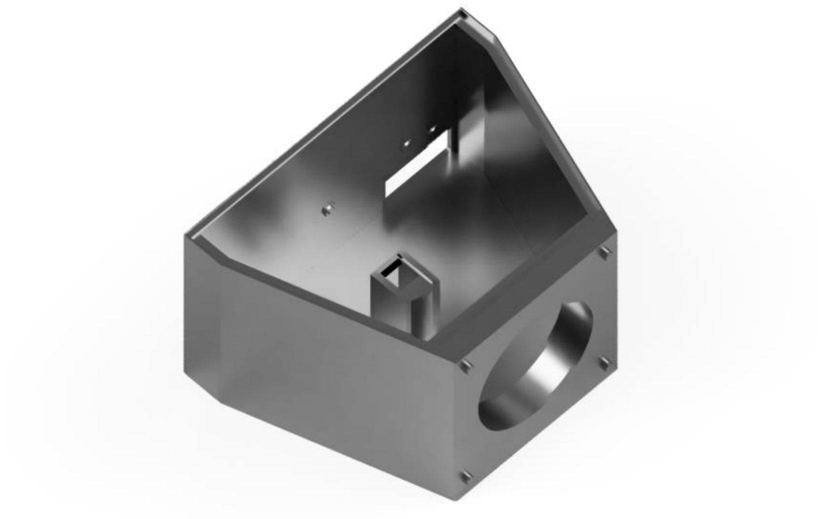


Figure 6-16 Main chamber

6.9 Licensing

This project is licensed under the GNU Lesser General Public License (GNU Lesser General Public License, 2021), also known as LPGL, which is a less strict version of the more common General Public License, known as GPL (GNU General Public License, 2021). LGPL allows proprietary usage or modification of the program, both open and closed source. While this can be a detriment to open-source projects, we felt it prudent to give our contractors the option to back out of the open-source paradigm in case they should wish to do so (though it has eventually become clear that this is highly unlikely).

6.10 Future Prospects: The road ahead

Our contacts at Sars made it clear early on that they wished us to make a product that they could carry on developing after our work was done. Thus, it has been an important consideration for us to develop the project in a manner that enables them to pick up where we would eventually leave of. A corner stone in this process has been the use of version control, namely Git (git, 2021) and GitHub (github.com, 2021), which now serves as the center of the code base, relevant design files, project tracking and, of course, versioning. Ownership of the repository will be transferred to an employee at Sars while remaining open-source. As such, we or anyone else inclined can contribute or continue development, though it is expected that the bulk of future plans will be carried out by an employee at Sars.

7 Conclusion

Even through many major and minor alterations to the project, the result still very much resembles the core elements of the initial design (with the exception of the experimental feature described at the end of section 3). Working incrementally, improvements were made in a most modular fashion while always keeping the system as a whole in mind. The requirements, as described in chapter 2, have been implemented, though their robustness is something that needs to be addressed in future development. The next and probably final step is to have talks with our contacts at Sars where we will brief them on every detail of the system to ensure that a hand-off can be carried out with minimal effort. While the system cannot be described as entirely complete, we feel that it is ready for such a hand-off and we would therefore like to conclude that the product development was, to a greater extent, successful.

8 References

Arduino IDE 2. (2021, 05 26). Retrieved from docs.arduino.cc: https://docs.arduino.cc/software/ide-v2

Arduino Nano . (2021, 05 26). Retrieved from www.arduino.cc: https://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardNano

Atlassian. (2021, 05 30). *Atlasian*. Retrieved from agile: https://www.atlassian.com/agile

atlassian. (2021, 05 30). *gitflow-workflow*. Retrieved from https://www.atlassian.com/: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Barr, M. (2021, 05 26). *Introduction to Pulse Width Modulation (PWM)*. Retrieved from barrgroup.com: https://barrgroup.com/Embedded-Systems/How-To/PWM-Pulse-Width-Modulation

BCN3D. (2021, 05 28). *Technical data sheet PLA.* Retrieved from www.bcn3d.com: https://www.bcn3d.com/wp-content/uploads/2019/09/BCN3D_FILAMENTS_TechnicalDataSheet_PLA_EN.pdf

Camera Lenses for Machine Vision. (2021, 05 30). Retrieved from www.thorlabs.com: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=1822

creality3d-ender-3-pro-high-precision-3d-printer. (2021, 05 30). Retrieved from https://creality3d.shop: https://creality3d.shop/products/creality3d-ender-3-pro-high-precision-3d-printer

DallasTemperature. (2021, 05 24). Retrieved from www.arduino.cc: https://www.arduino.cc/reference/en/libraries/dallastemperature/

DanielDondorp/ChatzigeorgiouGroup. (2021, 01 12). *GitHub*. Retrieved from https://github.com/ChatzigeorgiouGroup/imMobilize

DMK 33UP1300. (2021, 05 23). Retrieved from theimagingsoruce.com: https://www.theimagingsource.com/products/industrial-cameras/usb-3.0-monochrome/dmk33up1300/

doc.qt.io. (2021, 05 20). Retrieved from Using .ui files from Designer or QtCreator with QUiLoader and pyside6-uic: https://doc.qt.io/qtforpython/tutorials/basictutorial/uifiles.html

doc.qt.io. (2021, 05 21). Retrieved from QTimer Class: https://doc.qt.io/qt-5/qtimer.html

docs.python.org. (2021, 05 21). Retrieved from json - JSON encoder and decoder: https://docs.python.org/3/library/json.html

docs.python.org. (2021, 05 21). Retrieved from os — Miscellaneous operating system interfaces: https://docs.python.org/3/library/os.html

ElProCus. (2021, 05 27). *What are the Differences between BJT and MOSFET?* Retrieved from www.elprocus.com: https://www.elprocus.com/difference-between-bjt-and-mosfet/

Forking. (2021, 05 30). Retrieved from https://guides.github.com: https://guides.github.com/activities/forking/

git. (2021, 05 21). Retrieved from git-scm.com: https://git-scm.com/

GitHub. (2021, 01 29). *GitHub*. Retrieved from https://guides.github.com/activities/forking/

github.com. (2021, 05 21). Retrieved from github: https://github.com/

GNU General Public License. (2021, 05 21). Retrieved from gnu.org: https://www.gnu.org/licenses/gpl-3.0.en.html

GNU Lesser General Public License. (2021, 05 21). Retrieved from gnu.org: https://www.gnu.org/licenses/lgpl-3.0.en.html

https://www.atlassian.com/agile. (n.d.). Retrieved from Atlassian: https://www.atlassian.com/agile

imMobilize. (2021, 05 30). Retrieved from Github: https://github.com/ChatzigeorgiouGroup/imMobilize

InfiniiVision 3000A X-Series Oscilloscopes. (2021, 05 28). Retrieved from /www.keysight.com: https://www.keysight.com/zz/en/products/oscilloscopes/infiniivision-2-4-channel-digital-oscilloscopes/infiniivision-3000a-x-series-oscilloscopes.html

json.org. (2021, 05 21). Retrieved from json.org: https://www.json.org/json-en.html

JU1215. (2021, 05 30). Retrieved from https://www.elfadistrelec.no: https://www.elfadistrelec.no/Web/Downloads/_t/ds/JU1215%209V-4W%203S8P-EU_eng_tds.pdf

Keysight Technologies U1242C. (2021, 05 28). Retrieved from www.alliedelec.com: https://www.alliedelec.com/product/keysight-technologies/u1242c/70709271/

Language Reference. (2021, 05 26). Retrieved from www.arduino.cc: https://www.arduino.cc/reference/en/

OpenCV. (2021, 05 24). Retrieved from opencv.org/: https://opencv.org/

P, A. (2021, 05 28). *3D Printing Materials Guide: Plastics*. Retrieved from www.3dnatives.com: https://www.3dnatives.com/en/plastics-used-3d-printing110420174/

PyPi. (2021, 05 20). Retrieved from PyPi: https://pypi.org/project/PySide6/

Pypi. (2021, 01 29). *www.pypi.org*. Retrieved from https://pypi.org/project/PyQt5/

PyQt5. (2021, 05 30). Retrieved from www.pypi.org: https://pypi.org/project/PyQt5/

pyqtgraph.org. (2021, 05 20). Retrieved from pyqtgraph: http://www.pyqtgraph.org/

pySerial. (2021, 05 20). Retrieved from pySerial: https://pyserial.readthedocs.io/en/latest/pyserial.html

Python 3.9.5 documentation. (2021, 05 26). Retrieved from https://docs.python.org: https://docs.python.org/3/index.html

qt.io. (2021, 05 20). Retrieved from qt.io: https://www.qt.io/design

qt.io. (2021, 05 20). Retrieved from qt.io: https://www.qt.io/

Rice, C. (2021, 05 27). *Silicon lightworks*. Retrieved from Silicon lightworks: https://siliconlightworks.com/resoures/what-are-cob-leds

Serial. (2021, 05 26). Retrieved from www.arduino.cc: https://www.arduino.cc/reference/en/language/functions/communication/serial/

Slack. (2021, 05 22). Retrieved from slack.com: https://slack.com/intl/en-no/

Temperature Probe - DS18b20. (2021, 05 24). Retrieved from rpelectronics.com: https://www.rpelectronics.com/vma324.html

UIB. (2021, 05 30). *Sars International Centre for Marine Molecular Biology*. Retrieved from www.uib.no: https://www.uib.no/en/sarssenteret

Ultimaker-cura. (2021, 05 30). Retrieved from https://ultimaker.com: https://ultimaker.com/software/ultimaker-cura

What is UML. (2021, 05 28). Retrieved from www.uml.org: https://www.uml.org/what-is-uml.htm

wiki.qt.io. (2021, 05 21). Retrieved from Threads Events and QObjects: https://wiki.qt.io/Threads_Events_QObjects#Events_and_the_event_loop

Appendix A. Development and collaboration resources

GitHub repository containing code base, files, and project tracking for our solution–

https://github.com/simenfuglestad/CI-VTS

GitHub repository with code base of existing solution –

https://github.com/ChatzigeorgiouGroup/imMobilize

Slack, the communication tool of choice –

https://slack.com/intl/en-no/

Appendix B. Abbreviations

CI - Ciona Intestinalis

LED - Light Emitting Diode

GUI - Graphical User Interface

PLA - Polylactic Acid

USB - Universal Serial Bus

FFF - Fused Filament Fabrication

PWM - Pulse Width Modulation

COB - Chip on Board

DIP - Dual In-Line Package

SMD - Surface Mounted Device

IR – Infrared

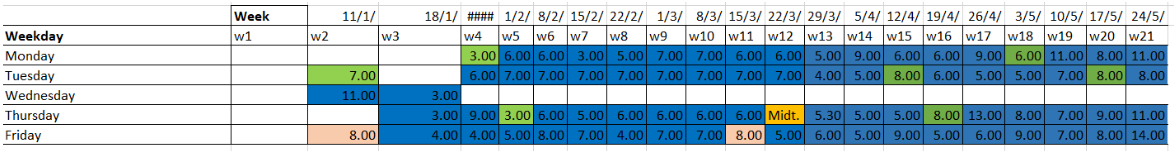
LGPL - Lesser General Public License

GPL - General Public License

Appendix C. Project management

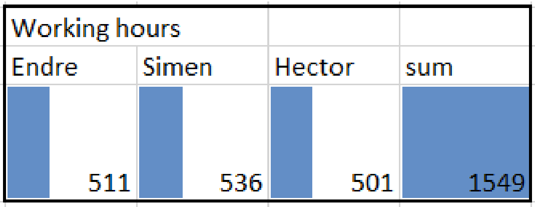
				Week	w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11	w12	w13	w14	w15	w16	w17	w18	w19	w20	w21	w22
				Man	4/1	11/1	18/1	25/1	1/2	8/2	15/2	22/2	1/3	8/3	15/3	22/3	29/3	5/4	12/4	19/4	26/4	3/5	10/5	17/5	24/5	31/5
				Fre	8/1	15/1	22/1	29/1	5/2	12/2	19/2	26/2	5/3	12/3	19/3	26/3	2/4	9/4	16/4	23/4	30/4	7/5	14/5	21/5	28/5	4/6
	Estimated Start Date	Estimated End Date	Progress	Assigned team member																						
Modeling the frame and container	1/2	5/2	100%	Endre					🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟩	
Integrating camera and lens	8/2	19/2	100%	Endre													🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟩	
Testing physical frame /w cuvette	22/2	26/2	100%	Endre and Hector								🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟩	
Integrate control system (arduino)	1/3	5/3	100%	Hector									🟨	🟩												
Integrate stimulus hardware	8/3	12/3	100%	Hector										🟨	🟩											
Write control software	15/3	26/3	100%	Hector and Simen									🟨	🟨	🟨	🟨	🟩									
Test and verify control software	29/3	2/4	100%	Simen															🟨	🟨	🟨	🟨	🟨	🟨	🟩	
Write interface software	5/4	16/4	100%	Simen					🟨	🟨	🟨	🟨	🟨	🟨	🟩											
Test and verify interface software	19/4	23/4	100%	Simen																🟨	🟨	🟨	🟩			
Write tracking software	1/3	7/5	100%	Endre and Simen								🟨	🟨	🟨	🟨	🟨	🟩									
Test and verify tracking sw	10/5	14/5	100%	Endre and Simen									🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟩						
Final testing rounds	17/5	28/5	100%	Everyone																		🟨	🟨	🟨	🟩	
Preliminary study	1/2	9/4	100%	Everyone	🟨	🟨	🟨	🟩																		
Final Report	10/4	28/5	100%	Everyone											🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟩	
🟨 Work in progress																										
🟩 Completed																										

Figure 17 GANT



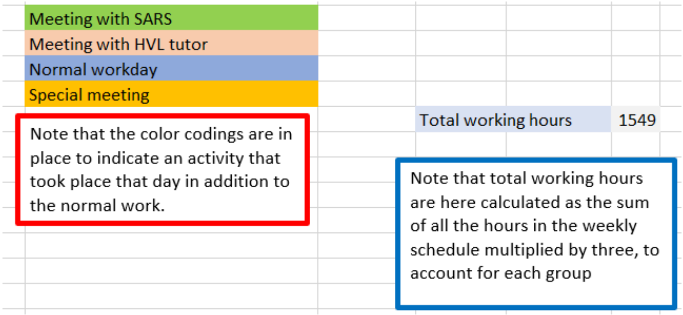
	Week	11/1/	18/1/	####	1/2/	8/2/	15/2/	22/2/	1/3/	8/3/	15/3/	22/3/	29/3/	5/4/	12/4/	19/4/	26/4/	3/5/	10/5/	17/5/	24/5/
Weekday	w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11	w12	w13	w14	w15	w16	w17	w18	w19	w20	w21
Monday				3.00	6.00	6.00	3.00	5.00	7.00	7.00	6.00	6.00	5.00	9.00	6.00	6.00	9.00	6.00	11.00	8.00	11.00
Tuesday		7.00		6.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	4.00	5.00	8.00	6.00	5.00	5.00	7.00	8.00	8.00
Wednesday		11.00	3.00																		
Thursday			3.00	9.00	3.00	6.00	5.00	6.00	6.00	6.00	6.00	Midt.	5.30	5.00	5.00	8.00	13.00	8.00	7.00	9.00	11.00
Friday		8.00	4.00	4.00	5.00	8.00	7.00	4.00	7.00	7.00	8.00	5.00	6.00	5.00	9.00	5.00	6.00	9.00	7.00	8.00	14.00

Figure 18 weekly schedule



Working hours			
Endre	Simen	Hector	sum
511	536	501	1549

Figure 19 Working hours



Meeting with SARS

Meeting with HVL tutor

Normal workday

Special meeting

Note that the color codings are in place to indicate an activity that took place that day in addition to the normal work.

Total working hours	1549

Note that total working hours are here calculated as the sum of all the hours in the weekly schedule multiplied by three, to account for each group

Figure 20 working hours total and information

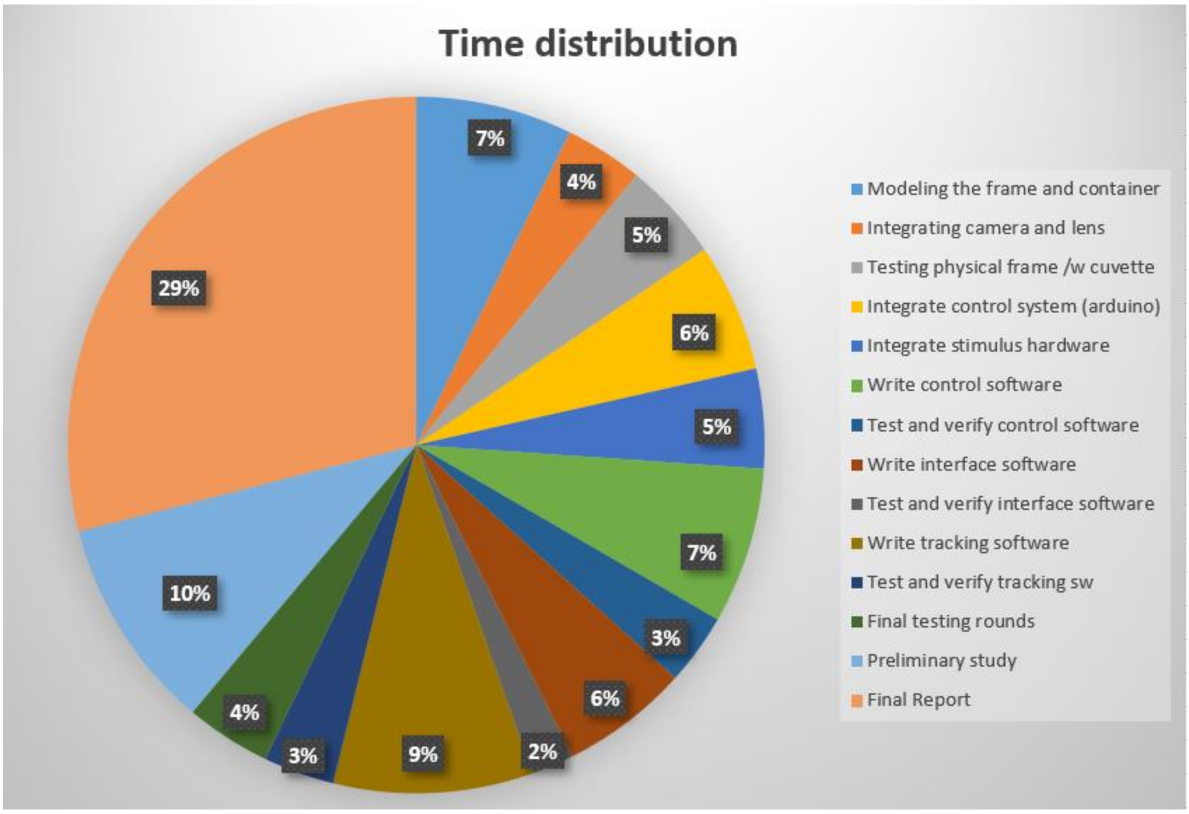


Figure 21 Time distribution