# Practical Work Report

**Software Engineering (Department of Electrical & Computer Engineering)**

**RisosEnterprises Ltd.**

**Unleash Space 20 Symonds Street, Auckland CBD, Auckland 1010**

**Dates Of Work Period: May 15th 2023 - 25th September 2023**

**Date Of Report: 25th September 2023**

## Summary

RisosEnterprises Ltd. is a small start-up Electronics Consumer-based company focusing on revolutionising the water industry. Currently, the company is based in Auckland CBD, and most of the employees work at the unleashed space at the University Of Auckland, where they have been so lovely to let us stay there for free. The company's main product they are developing is the Waicorder, a revolutionary handheld device to detect waterborne particulates (E. coli, protozoa, microplastics, sediments, algae, and many more) as well as chemicals (Nitrates, Phosphates, Hormones, and many more) in under 10 seconds, on-site.

During my internship, I was under the guidance of Dr. Alex Risos, who was my mentor when I needed help or had any queries about the project. During my entire time there, I was tasked with making a Camera app that could be used to display a live video feed from a high-resolution microscope camera that could zoom in, zoom out, take pictures and store them, change the resolution, and get the IP address. The device also had to be compatible with the Raspberry Pi and be optimised not to lag at high solutions, which proved to be a very challenging task later on that required compromise.

The valuable skills that I learnt during this internship were learning the C++ programming language, the Qt Framework, how to cross-compile and practical research skills.

Overall, the project was very challenging and required a lot of perseverance and determination to finish. Still, in the end, it was gratifying, and I learnt a lot of valuable things that will be carried with me throughout my career.

# Acknowledgements

Risos Enterprises Ltd. ensures that their internships are helpful for anyone who participates, and those who come out always learn some essential skills. But I do want to acknowledge some people who went above and beyond.

First and foremost, I would like to thank Dr. Alex Risos for his mentorship and guidance throughout the internship, it did help. He motivated me and taught me essential skills about research, determination and perseverance to achieve my goals. He was patient and never seemed annoyed or frustrated when things went south with our projects. So I am very grateful.

Secondly, I would like to acknowledge the other interns who also worked at the company; with them, I was able to finish this project. they aided with much of the technical side when I couldn’t figure out specifics. They were also very friendly and great to talk to. There was not a day that I left there not smiling or happy about the fruitful conversations we had. So for that, I have something to cherish about.

# Table Of Contents

# Table Of Figures

# 1 Introduction

This report covers my practical work experience as a Software Engineer at Risos Enterprises Ltd. from 15 May 2023 to 25 September 2023. This report will give details about the work structure at Risos Enterprises Ltd., the project I worked on, the skills I learned and the knowledge I gained. I will summarise this report with the appraisals of my time at the company and what I thought of the whole experience.

Risos Enterprises Ltd. was established in 2021 by Dr. Alex Risos. The company aims to create a product that could find microbes in water without waiting weeks in the lab for the results to return. This is something revolutionary. What we are currently doing now is making the prototype of the Waicoder, which is a revolutionary handheld device to detect waterborne particulates (E. coli, protozoa, microplastics, sediments, algae, and many more) as well as chemicals (Nitrates, Phosphates, Hormones, and many more) in under 10 seconds, on-site. They are a small startup company with approximately 10-15 interns working on this project, and hopefully get a working prototype by the end of this year.

I was employed at Risos Enterprises Ltd. as a Software Engineer intern, where my primary duties were to apply for the digital microscope camera that could zoom in and zoom out, take pictures and store them, change the resolution, and get the IP address. I also had to assemble parts of the microscope as well. The device also had to be compatible with the Raspberry Pi and be optimised not to lag at high resolutions, which proved to be a very challenging task later on that required compromise. During my time there, I spent a lot of time on this project, which required a lot of perseverance and determination to finish as I met many obstacles that would leave me stumped for weeks. The work included research and development, where I would have to research a framework or programming language, how to implement it and use it for this project. I used simple tools to assemble the microscope, such as screws, Allen keys, screwdrivers, bolts, and nuts. The software skills I learned during this time were the programming language C++, the Qt Framework, how to cross-compile between different OS systems, and how to set up a Raspberry Pi and use one. The soft skills I learned during my time were perseverance, determination, problem-solving, communication, and valuable research skills.

# 2 Early Sections: Company Information

## 2.1 Layout Of Office

Risos Enterprises Ltd. is situated at Unleash Space, 20 Symonds Street, Auckland CBD, Auckland 1010, where the university has let us stay for the past few years. Although we have yet to have an official office, we are expected to move out of this facility by the end of this year. With promising investments coming in, we are hopeful to get our own office space and facilities for the staff. there are many lights and a great window view from within the space. Plenty of tables are in the space, but we have allocated a specific section for our company, which is nice. Although allocated space, we could move to any other table to feel more comfortable. There is also a workshop area with many tools to help us build things, a 3D printer, and perfect storage spaces to put our things in, so this place is excellent for a startup like us. The area is relatively small, but we are still grateful to be provided with this space for free to continue our work.

## 2.2 Staff Organisation And Structure

Because the company is relatively small, there is no textbook company structure, as our team comprises around 15 interns/full-time/part-time employees. The structure is relatively flat, with just Dr. Alex Risos as the CEO and everyone else on the same level. If we have any queries, we can flick him a text, or if he is in the space, we can walk up to him and talk to him about any concerns we may have. We never feel he is holding any power up against us and that we are all treated equally as individuals and employees, which I appreciate about working at Risos Enterprised Ltd.

## 2.3 Comments On Building

The unleashed space area has many amenities for us while we work there. There is a small kitchen area to heat food and get water. There is also a fridge for us to keep our food in. If we are tired, there are bean bags for us to sit on and relax. In addition, we can use a coffee and hot chocolate machine to get drinks in the morning so we always feel satisfied. Surround the building are also many food stores and places to buy things if we are hungry, so the locations are also great. Every once in a while, if we had a breakthrough or outstanding work, Dr. Risos would take us out for lunch, which we appreciate as it's just a startup, and there needs to be more cash flow. The Unleashed space area is just one big room with all these amenities and facilities, and outside that is just the Engineering building at the University of Auckland

# 3 Work Experience

Upon arriving on my first day at Risos Enterprises Ltd. I was greeted by Dr. Alex Risos, who showed me around the space and told me how he was grateful to the University for letting them use it for free as they are just a small start-up company. He then assigned me tasks to prepare for the project I was about to take up. So he first told me to research about displaying a live video feed from a camera on an application in a Linux environment. He then told us he wanted us to create an app to display the video feed. So, I did my research and presented him with my application, which was pretty simple. After that, he told me what my project would be for the following few weeks, where I learned how to ask elicitation questions, follow up, and confirm the project specifications with clients (Dr. Risos). Dr Risos then told me to present my research and planning before I began the project. He was very open about using AI to aid us in this new day and age, as it is a precious tool.

## 3.1 Research And Planning

To begin, I asked ChatGpt what the best way to make a Camera app that displayed a live video feed that could capture and store images that were compatible with Linux and would be running on a Raspberry Pi. I was given an excellent response that told me about all the pros and cons of using which programming language and what approach I should take. ChatGpt told me that the best solution was to Use OpenCV with Python, or I could use C++ and the Qt Framework to help me build my app. Looking back, I should have researched more, not just trust the AI. Still, because I already knew how to code in Python, I decided to dive straight into it because I was more comfortable in that language rather than learning a whole new framework and Language to start creating the app without having a more thorough look. Then I presented my findings and weighed the pros and cons of the ChatGpt response to Dr Alex Risos. He approved it, so I continued with the Python approach

## 3.2 Python Approach

I started this approach with two options: the OpenCV library or the Kivy framework. Since I was familiar with the OpenCV library, I decided to use it since Dr. Alex Risos wanted to use a pinch on the touch screen to zoom in. I assumed that would be the hardest thing to do, so I started with that functionality. What I came to realise is that with OpenCV, it's not possible to do a pinch to zoom because even if it recognises the pinch, there is no inbuilt functionality in OpenCV to be able to crop up the video feed and only display a specific section of the video feed, so I had to have another look at my approach. This would be a much better solution if I had started with the Kivy Framework. Implementing the pinch-to-zoom functionality in the Kivy Framework with Python was very difficult because of all the math that was involved in cropping the image screen and just the rendering of what a pinch was and what was not a pinch and rather a swipe to traverse the image made it hard to implement. In the end, though, I managed to succeed, although there were many bugs as to how far you could swipe and the functionalities to not being able to zoom past the borders, which took a lot of effort to fix these edge cases. After fixing these bugs, adding the remaining functionalities, like capturing the image, switching cameras, changing resolution, and the IP address, was a breeze. I didn't have to worry about the edge cases for these because Python and the Kivy framework handled all the errors for me without worrying too much about it. There were a few limitations with the Kivy framework, though. I could add menus, and the spinners were limited, which disappointed me. But in the end, I managed to make the app shown in Figure 1 that had all the functionalities required. However, there were some problems that I didn't account for; I didn't realise that since Python was such a slow programming language, it took almost 30 seconds for the app to initialise and also 30 seconds for the camera to initialise in the app when switching to higher resolutions as most cameras require you to close the camera and open it to get a new solution. This increases to about 5 minutes on the Raspberry Pi due to its slower processing power. This was bad as people using the microscope couldn’t afford to wait so long to use the app, and there wasn’t any other way around it because I always had to reinitialise the camera to change the resolution, so I had to rethink my approach. I decided that the Python programming language was too slow for what Dr. Risos was looking for, so I decided to go outside my comfort zone and learn C++ with the Qt framework as C++ is a lower-level programming language and, as a result, is much faster than Python.

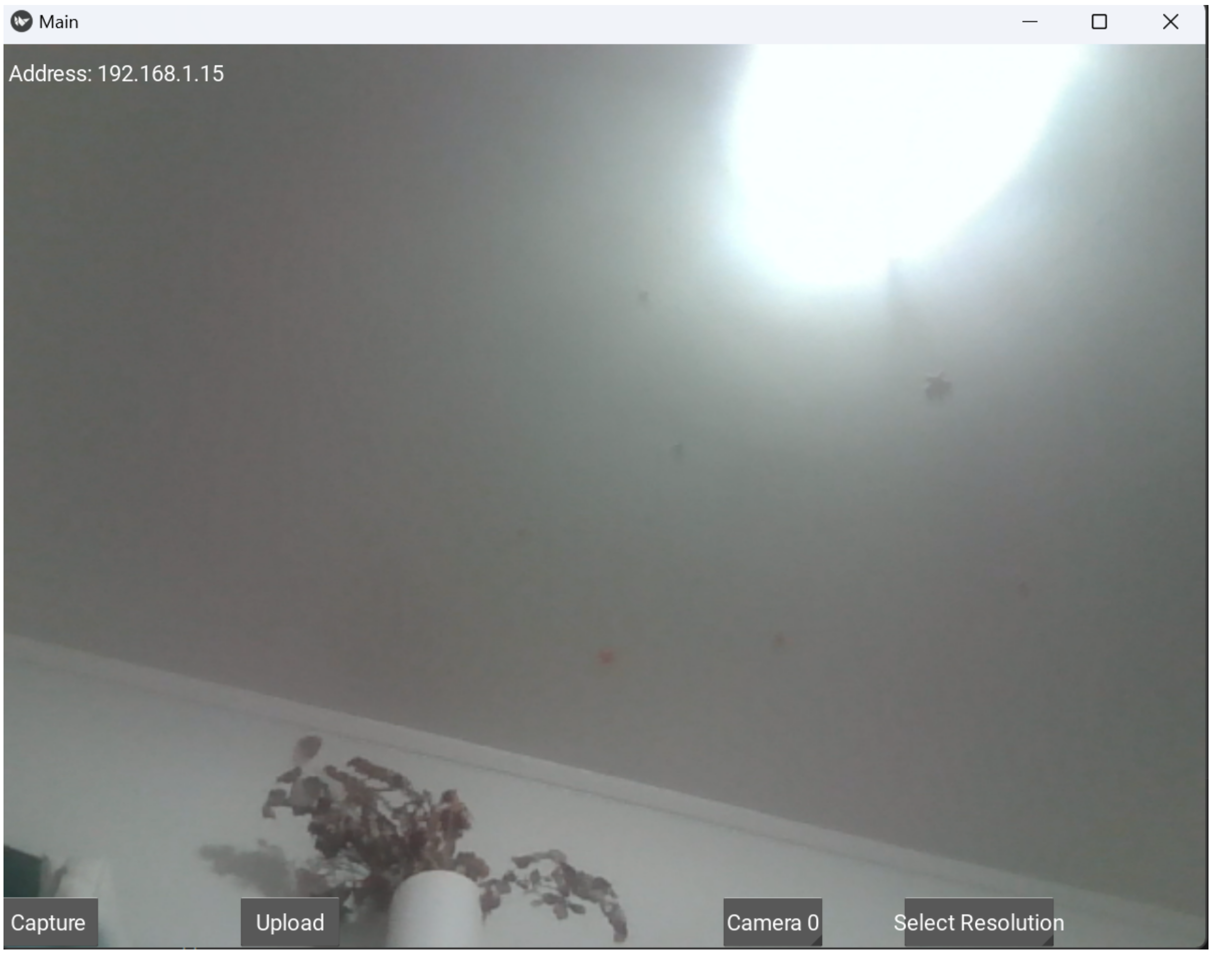
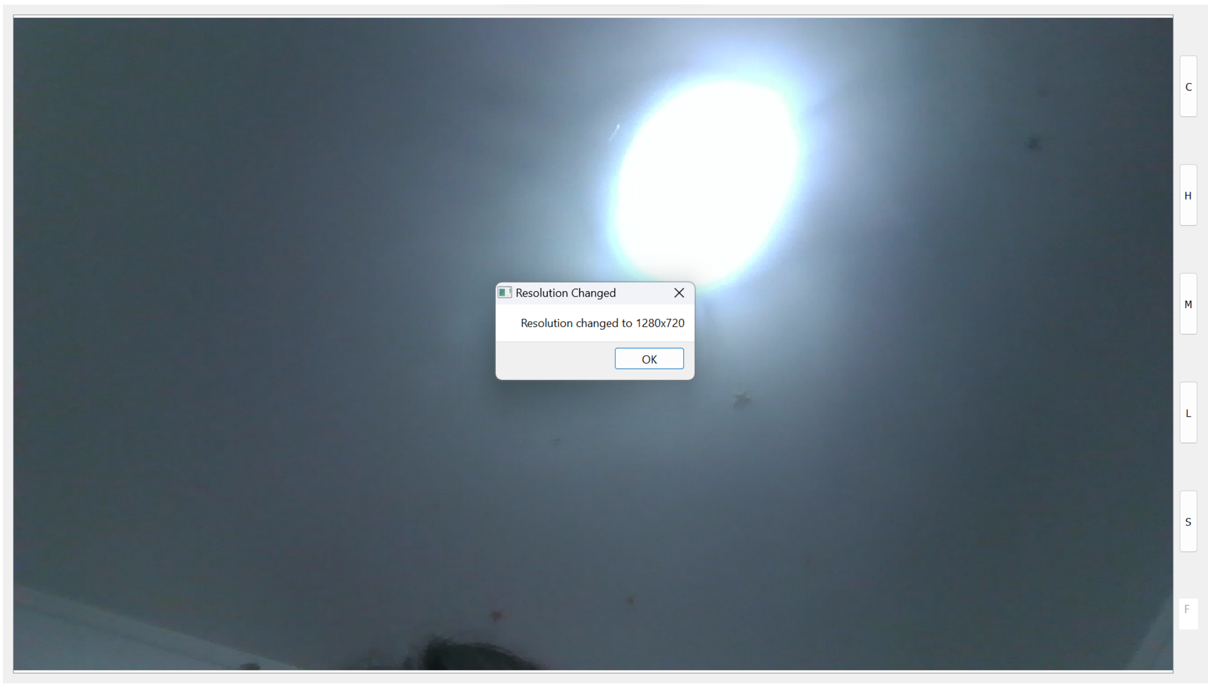


Figure 1: Python App

## 3.3 C++ Using Qt Framework Approach

With the Qt FrameWork and C++ approach, I spent many weeks researching the C++ programming language and Qt framework, how everything worked through online tutorials and documents, and asking questions within the Reddit and Qt Forums community. Since I know Java quite well, the syntax and Type safety in C++ were familiar, and I could easily transition. After finding I was comfortable with C++ and the Qt Framework, I decided to start the app. I couldn't reuse the code from the Python app for the pinch-to-zoom because it was a completely different framework compared to Kivy, and it didn't transition over smoothly, so I had to start again with that functionality, although the basics were there. I was lucky, though; that was an opensource GitHub project that already implemented a pinch to zoom feature in their game, and I used their pinch to Zoom code and changed a few things so that it could be implemented into my application on Qt, which saved me a lot of time. After I could do that, I started implementing all the other features Alex wanted on the app, which was relatively straightforward to do as that part was very similar to the Python approach, so it was entirely transferable. I found using the Qt designer quite interesting as I didn't need to guess where to put everything and could visually see it before I tested it. After I created the app on Windows (Figure 2), I decided to send the code to the Raspberry Pi, but to my dismay, I couldn't install Qt6 using Apt or any other package management system as Qt6 recently came out. So, I had to learn to cross-compile as the Raspberry Pi was too slow to compile it from the source code itself.

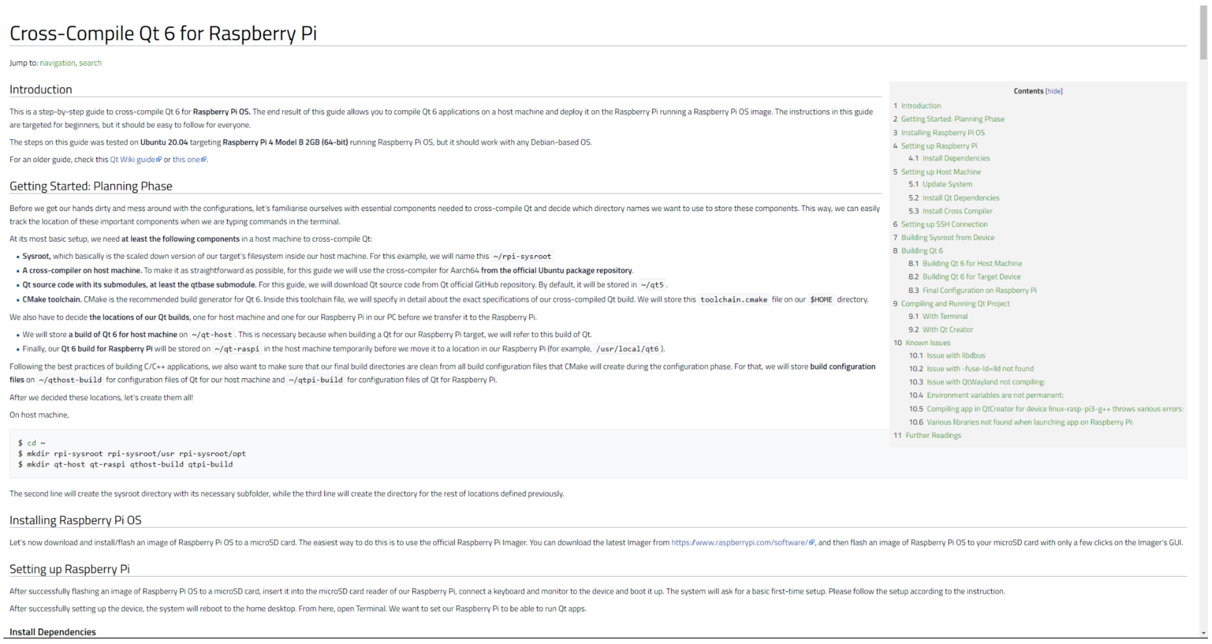

Resolution Changed

Resolution changed to 1280x720

**Figure 2: C++ with Qt Framework Application**

## 3.4 Cross-Compiling

Cross-compiling was one of the most challenging things for me because I’d never used a Linux system before, and I was entirely new to all of this. Hence, it took me a few weeks to learn about cross-compiling, but it was ultimately worth it. I learned some precious tools and skills. The tutorial provided by the Qt community aided me in cross-compiling in Qt6, as shown below in Figure 3. It was beneficial, but I encountered many problems with the tutorial as my Raspberry Pi didn't precisely match theirs, so there were often errors that resulted in me hitting bumps and getting stuck for a while. When I hit these bumps, what would I do? I would go to Dr. Risos for advice on overcoming these challenges, and he would point me to the forums and online community to ask for help. This was great advice as after a few days, I got the answer I was looking for and successfully cross-compiled Qt6 to the Raspberry Pi, which was terrific.


Cross-Compile Qt 6 for Raspberry Pi

Jump to: navigation, search

Introduction

This is a step-by-step guide to cross-compile Qt 6 for **Raspberry Pi OS.** The end result of this guide allows you to compile Qt 6 applications on a host machine and deploy it on the Raspberry Pi running a Raspberry Pi OS image. The instructions in this guide are targeted for beginners, but it should be easy to follow for everyone.

The steps on this guide was tested on **Ubuntu 20.04** targeting **Raspberry Pi 4 Model B 2GB (64-bit)** running Raspberry Pi OS, but it should work with any Debian-based OS.

For an older guide, check this Qt Wiki guide or this one.

Getting Started: Planning Phase

Before we get our hands dirty and mess around with the configurations, let's familiarise ourselves with essential components needed to cross-compile Qt and decide which directory names we want to use to store these components. This way, we can easily track the location of these important components when we are typing commands in the terminal.

At its most basic setup, we need **at least the following components** in a host machine to cross-compile Qt:

- **Sysroot,** which basically is the scaled down version of our target's filesystem inside our host machine. For this example, we will name this `~/rpi-sysroot`
- **A cross-compiler on host machine.** To make it as straightforward as possible, for this guide we will use the cross-compiler for Aarch64 **from the official Ubuntu package repository**.
- **Qt source code with its submodules, at least the qtbase submodule.** For this guide, we will download Qt source code from Qt official GitHub repository. By default, it will be stored in `~/qt5`.
- **CMake toolchain.** CMake is the recommended build generator for Qt 6. Inside this toolchain file, we will specify in detail about the exact specifications of our cross-compiled Qt build. We will store this `toolchain.cmake` file on our `$HOME` directory.

We also have to decide **the locations of our Qt builds**, one for host machine and one for our Raspberry Pi in our PC before we transfer it to the Raspberry Pi.

- We will store **a build of Qt 6 for host machine** on `~/qt-host`. This is necessary because when building a Qt for our Raspberry Pi target, we will refer to this build of Qt.
- Finally, our **Qt 6 build for Raspberry Pi** will be stored on `~/qt-raspi` in the host machine temporarily before we move it to a location in our Raspberry Pi (for example, `/usr/local/qt6`).

Following the best practices of building C/C++ applications, we also want to make sure that our final build directories are clean from all build configuration files that CMake will create during the configuration phase. For that, we will store **build configuration files** on `~/qthost-build` for configuration files of Qt for our host machine and `~/qtpi-build` for configuration files of Qt for Raspberry Pi.

After we decided these locations, let's create them all!

On host machine,

```
$ cd ~
$ mkdir rpi-sysroot rpi-sysroot/usr rpi-sysroot/opt
$ mkdir qt-host qt-raspi qthost-build qtpi-build
```

The second line will create the sysroot directory with its necessary subfolder, while the third line will create the directory for the rest of locations defined previously.

Installing Raspberry Pi OS

Let's now download and install/flash an image of Raspberry Pi OS to a microSD card. The easiest way to do this is to use the official Raspberry Pi Imager. You can download the latest Imager from https://www.raspberrypi.com/software/, and then flash an image of Raspberry Pi OS to your microSD card with only a few clicks on the Imager's GUI.

Setting up Raspberry Pi

After successfully flashing an image of Raspberry Pi OS to a microSD card, insert it into the microSD card reader of our Raspberry Pi, connect a keyboard and monitor to the device and boot it up. The system will ask for a basic first-time setup. Please follow the setup according to the instruction.

After successfully setting up the device, the system will reboot to the home desktop. From here, open Terminal. We want to set our Raspberry Pi to be able to run Qt apps.

**Install Dependencies**

**Figure 3: Cross Compile Qt6 To Raspberry Pi.**

However, even after reaching this milestone, I faced further challenges. I realised that Gstreamer, the backbone of video processing on the Raspberry Pi, was not yet fully compatible with Qt6 because of how recently it has just come out. Hence, I ended up with the videos not going through the app on the Raspberry Pi, which was very frustrating. This served as a learning experience that I need to thoroughly do my research before jumping into anything because if I had correctly done my research, I could have saved a lot of time. The image in Figure 4 shows what I was left with after cross-compiling, so it didn't work.

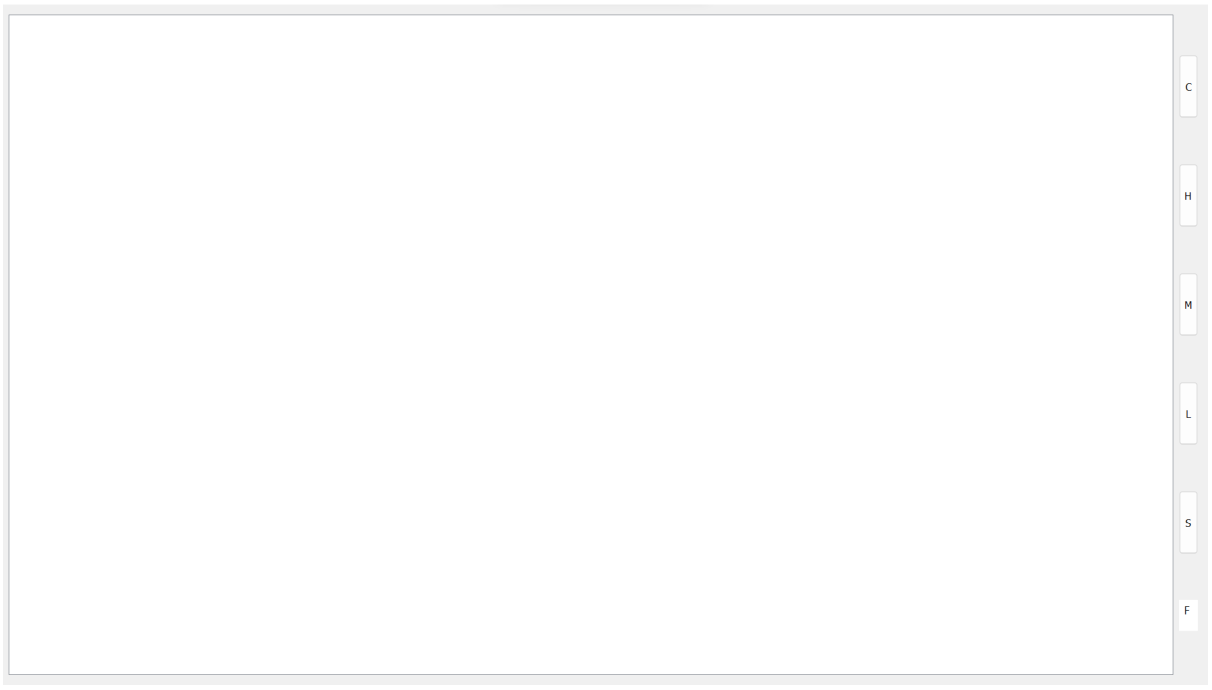


**Figure 4: Issues With Qt6**

With these pressing issues, I had to port my code from Qt6 to Qt5; this was not that hard of a task because Qt5 was very similar to Qt6, but with tiny changes between Qt6 and Qt5, I could easily port the code within a day. After porting the code, I didn't need to cross-compile anything to the Raspberry Pi because Qt5 was quite old, so It was already available from Apt, and I could quickly download everything and get the app to work.

## 3.5 Issues With Raspberry/Linux OS

Although the app was working great, there was a massive problem. Because the Raspberry Pi had such weak processing power, even though I used C++ and a USB 3.0 cable to transfer the data from the camera to the Raspberry Pi, When the camera was running at 8MP resolutions, it was highly laggy running at 1 to 10fps, which was not good enough and would not meet the required specifications given to my by Dr Alex Risos. I talked with him about my issues, and then we had to compromise. Due to the limitations of a Raspberry Pi and the limitations of using Linux on Linux, there were issues with processing Raw and compressed video formats that caused it to be so laggy. However, these issues were not prevalent in Windows as we handled them all. Still, in Linux, we had to handle it ourselves, and neither I nor I were technically able to handle these issues as they require extreme expertise to handle these cases. As a result, we agreed that we wouldn't be able to use the Raspberry Pi, so we had to switch to a Windows computer to process and display the videos.

## 3.6 Reverting To Windows

After reverting to Windows, I successfully created an app that Dr. Alex Risos was happy with that could handle the videos with high resolutions without being too laggy. Below in Figure 5 shows the final product of the app that was to be used with his microscope. Where C represents the Capture Image function. H stands for high resolution, M is Mid resolution, L is Low resolution, S is to Switch cameras, and F is the files button for opening up a menu with more options, as shown in Figure 6.

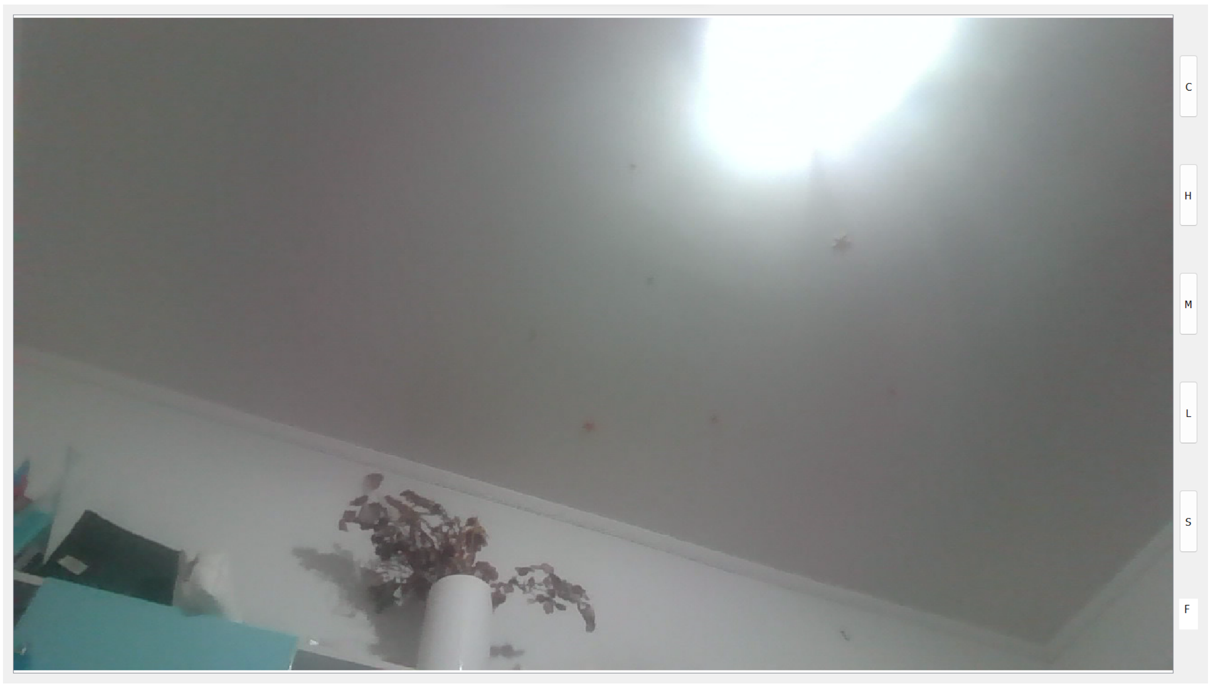


**Figure 5: Final App**

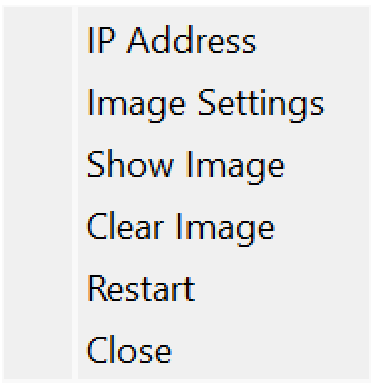


**Figure 6: Menu For Files Button**

# 4 Reflection Appraisal

## 4.1 Impressions And Performance

Risos Enterprise Ltd. is a startup taking the industry by storm with its fast-paced environment with a revolutionary product that will have a prototype in a few months. We all take pride in our work here and believe we should challenge the status quo. To go above and beyond, put 100% effort into everything we do, and never give up. This is especially important because of my youth and having so much more to learn from. Life is always about learning, which I genuinely believe in and will try to live by.

Although the company is relatively small, it does feel like a family. All the employees are friendly and always greet me with a smile, even when I'm stressed or down from the load of my coursework. Alex was approachable during this entire time, and he never would put us down, always held us up and motivated us to keep pushing forward and to give up. I have much to thank for this internship; it was a great experience.

## 4.2 Skills Gained

Throughout this internship, I learned many excellent skills. Some of them were:

- **C++ Language:** I had to get out of my comfort zone to learn a new language; although I was a bit hesitant at first because I had to face a new, scary challenge, I became proficient in C++, which is excellent for me. And I learnt that you should always go outside of your comfort zone if it will benefit your future.
- **Qt Framework:** The Qt framework was quite challenging for me primarily since I've never handled any UI applications before, so it was a huge learning curve. Some aspects of Qt were not entirely intuitive, so I did have to put in a bit of extra effort to get used to Qt, but in the end, it was all worth it.
- **Cross-Compiling:** This was especially the hardest out of everything I had to learn because, before this project, I had never heard what Cross-Compiling was and its purpose. However, after learning about it, I understand its uses and can quickly cross-compile projects, which is excellent for me. This is a great skill to have in my arsenal.
- **Research And Development:** Throughout this project, one of the key things I've learnt from Alex himself was how to be independent, do your research and ask thoughtful and insightful questions. This was a key thing I learned here at my internship that is transferable to any

job or career I may choose and is something I appreciate that I was taught.

## 4.3 Lessons Learnt

At this internship, I learnt a precious lesson about taking time to rush into things. To take a calm and settled approach. To avoid getting too excited about a project and not think things through. To research a project thoroughly before making any advances in writing the code.

If I had known these things before I started this project, I would have saved a lot of time and finished the project much quicker. But this valuable lesson I learned will stick with me for the rest of my career.

Another lesson I also learned is that no matter how hard things get, you should never give up because you only lose when you give up. If you keep trying, then at some point, you will get there. This was a precious lesson taught to me by Dr. Alex Risos, and in the end, I honestly did get there, so for that, I am genuinely thankful for his guidance and mentorship.

# 5 Conclusions

In conclusion, I am extremely lucky to have found this internship and learned some valuable skills from my peers to the CEO, Dr. Alex Risos. These skills will be carried through with me for the rest of my life. Despite his highly fast-paced environment and stress, Alex always took the time to tend to the interns, catch up with them, and ensure they could keep up and provide assistance when needed. His patience and kindness were highlighted during this time, making this internship enjoyable.

Being part of such a revolutionary company that aims to change the world and improve the lives of others has indeed had an impact on me, even if my project is just a tiny part of this extensive mission. I've learnt what it means to be part of a team, work hard, and watch the fruits of my efforts grow, which was amazing to see. I learned how to solve problems effectively and the hard way to take your time and not rush into things because it's better to have a clear map of what you're doing than to run into the woods blindly.

Another small but valuable lesson I learned during my time at Risos Enterprise Ltd. was the importance of community and communication. With the assistance of my fellow interns at the company, I achieved my goals. The acceleration of my growth in learning new frameworks and the programming language and having people around to assist me was great, which opened my eyes to its importance. I'll take this back when I move on to my career. Always be communicative and transparent when you need help or when you can help others, as they have done with me.

I am grateful for this internship that lasted over 5 months while also studying at the University. So I'm genuinely grateful that Dr. Alex Risos was flexible with my hours and accommodated my exams and University commitments. The length of this internship shows the robust and integral knowledge I learned while at Risos Entreprises Ltd.