Design and Integration of Portable Optical Spectrometer

Submitted by

Aung Myin Kyaw

A0160121A

Department of Electrical & Computer Engineering

In partial fulfilment of the
requirements for the Degree of
Bachelor of Engineering
National University of Singapore
B.Eng. Dissertation

Design and Integration of Portable Optical Spectrometer

By Aung Myin Kyaw

National University of Singapore

2018/2019

Project ID: H531300

Project Supervisor: Prof Hong Minghui

Deliverables:

Report: 1 Volume

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Spectrometers are commonly used for spectroscopic analysis of sample materials. However, the sizes of the conventional spectrometers are large, and it is restricted only in the laboratory. This project is to design and integrate portable optical spectrometer.

This part of the project is the development and implementation of a user interface component of the portable optical spectrometer. Though there are some existing user interfaces, it has limitations such us customisation, controlling and communicating with different components of the portable spectrometer. Thus, the report will discuss the implementation using C# programming language to integrate the user interface.

1. INTRODUCTION

1.1 Objective

Optical spectrometer is a tool used to measure the intensity of light over a specific portion of electromagnetic wave spectrum. Raman spectroscope is widely used to observe different materials, chemical substances and many more as it has many advantages depending on the purposes of the analysis. However, size of conventional Raman spectroscope is rather large, and it is inconvenient for measurements outside the laboratories. This project is to develop a multifunctional portable Raman spectroscope for point-of-care diagnostics. This report will discuss the implementation and integration of the user interface using C# programming language. Firstly, for the background information this report will provide the overview and properties of Raman scattering. It will then discuss different programming languages available for the implementation and the reason behind choosing C# for the implementation. Next, it will discuss the documentation of the implementation methods and difficulties that encountered during the process. Lastly, the report will present the outline of what the end-product will look like.

2. BACKGROUND OF RAMAN SCATTERING

Different spectroscopy methods are often used for the classification of a wide range of samples. These methods are used for qualitative and quantitative analyse of the samples. The qualitative analysis is to establish the identity of sample while quantitative analysis is to estimate the concentration of analyte in sample. Raman spectroscopy method can be used for both qualitative and quantitative purposes. Qualitative analysis can be performed by measuring the frequency of scattered radiations while quantitative analysis can be performed by measuring the intensity of scattered radiations [1][2]. Thus, nowadays Raman spectroscopy has become highly popular for many different purposes. It is well-suited to analysis different solids, liquid sor gases, depending on the set up and it is relatively fast process. Raman spectroscopy is a technique which detects photons that have undergone Raman scattering. Raman spectroscopy can be determined by observing Raman scattering, or inelastic scattering of monochromatic light. Scattering occurs when an incident photon collides with the target molecule along the direction of travelling (Figure 1.1). The target molecule is excited and then relaxes and emits a photon immediately.

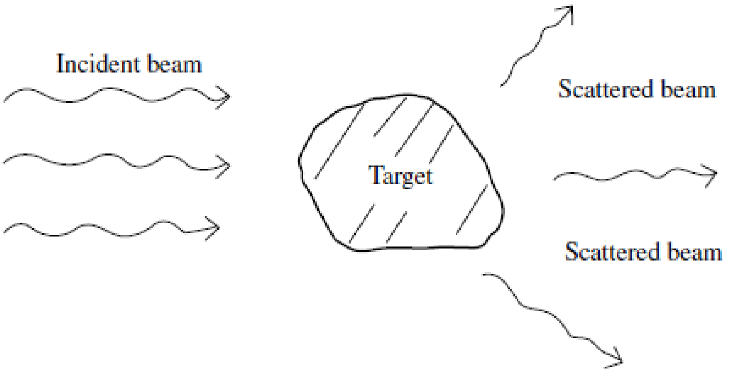


Figure 1.1 Diagram of scattering phenomena. Reprinted from S.-L. Zhang, Raman Spectroscopy and Low-dimensional nanoscale Semiconductors, Science Press, (2008)

There are many different scatterings, such as Rayleigh scattering, Brillouin scattering and Raman scattering. The main different between above then scattering is the change of energy when the collision occurs. It is known as inelastic scattering when energy of the emitted photon is either lower or higher than the incident photon. Thus, Raman scattering (Raman shift) is an inelastic scattering with the change of energy greater than 1 cm^{-1}[1]. Raman shift is expressed in wavenumber (em^{-1}, wavenumber multiplied by the velocity of light, 3 X 10^{10} cm/sec, gives the actual frequency) [3]. As the energy change in Rayleigh scattering is caused by the recoil of the target, it is considered as elastic scattering [4].

Raman scattering spectra have their own characteristics. Usually, Raman shifts are naturally in wavenumbers, which have units of inverse length (cm^{-1}). During scattering, the frequencies emitted are used as chemical fingerprints for the identification of molecules. Raman wavenumber (ω_R) can be calculated by using equation (1.1a), where ω_0 is the wavenumber of the incident light and ω_S is the wavenumber of the Raman spectrum [4].

$$\omega_R = \omega_0 - \omega_S \tag{1.1a}$$

Use the following equation (1.1b) to convert between spectral wavelength and wavenumbers of shift in the Raman spectrum, where λ_0 is the wavelength of the incident light and λ_S is the wavelength of Raman spectrum [4].

$$\omega_R = \frac{1}{\lambda_0} - \frac{1}{\lambda_s} \tag{1.1b}$$

The intensity of Raman scattering is a lot weaker than the intensity of Rayleigh scattering [4]. However, it can be enhanced using Resonance Raman Spectroscopy (RRS) and Surface Enhanced Raman Spectroscopy (SERS) [4]. SERS is a technique in which sample is adsorbed on a colloidal metallic surface such as silver, gold or copper. Hence, this improves the intensity of Raman signals [6] [7]. In addition, the low intensity can be easily improved with a notch or edge-pass filters. Similarly, recording the Raman spectrum has become easier by the advancement of Charge-Coupled Device (CCD) spectrometers.

CCD spectrometers are a multichannel detector which consists of a 2-dimensional array of light-sensitive elements. There are many different CCD spectrometers in the market. However, Compact CCD Spectrometers 100 (CCS100) by Thorlabs will be used for the implementation of the project. After considering the different parameters such as overall Size, Cost, Speed, Customization, Sensitivity, Signal-to-Noise Ratio, Dynamic range, Linearity, Thermal Stability, and Robustness CCS100 fits almost all the requirement to build a portable optical spectrometer (See Appendix A for Technical Data of CCS100).

3. IMPLEMENTATION

For the developers, Thorlabs provides different headers and libraries of different programming languages such as C, C++, C# and LabVIEW. Though this could be considered as a limitation since it will be difficult to use other programming languages which are not permitted by Thorlabs. However, above four languages are comprehensive enough to program a user interface or a controller to the spectrometer. Hence, the choice of the programming language will be selected from C, C++, C# and LabVIEW. When to choose which programming language to be used there are different parameters such as Cost of the Integrated Development Environment (IDE), Performance and Resources are to be considered.

3.1 Cost of the IDE

In term of cost, three languages C, C++ and C# can use free IDE such as virtual studios, code block or eclipse for implementation. For LabView, though there is a free license for students it is only for first 6 months. After the end of 6 months it will charge SGD486.0 to purchase the Switch Executive combo of National Instrument which include LabVIEW. Thus, using LabVIEW for the implementation is an expensive option.

3.2 Performance and Resources

Table 1.1 Table of all 4 different programming and their properties

	C [8]	C++ [8]	C# [8]	LabVIEW [9]
Type	Procedural and text base	Object-oriented and text base	Object-oriented and text base	Object-oriented and graphical
Memory management	Manual memory management	Manual memory management	Garbage collection	No memory management system
Performance	Fast	Fast	Standard performance	Standard performance
Purposes	Can code for any platform	Can code for any platform	Can code for any platform	LabVIEW application
Compilation	Given the syntax is right, allows coding almost anything	Given the syntax is right, allows coding almost anything	Warning messages will show during compilation if there is an error	It is easy to program, and errors will only display when there are some nodes not connected
Purposes	Good for system level code, and embedded devices	Good for server-side applications, networking, gaming, and device drivers	Good for simple web, mobile and desktop application	Only on LabVIEW platform

By observing the table above, LabVIEW is not the most suitable to create a desktop application. Both C++ and C languages are faster than C# in the performance however C# makes up its shortcoming by having better memory management and warning messages during compiling so that it is faster and easier to debug. Moreover, C is a procedural language which has limitations with the tasks that it can carryout compare to C++ and C#. Thus, both C# and C++ are the most suitable languages for the implementation of user interface for the portable spectrometer however C# is usually a more popular choice by the developer’s community when designing a desktop application. Though the only limitation is that the documentations that are provided by Thorlabs is in LabVIEW version. Since the fundamentals are the same in most of the programming languages, it is still possible to create a desktop application for the portable spectrometer by understanding the LabVIEW version of the documentation. In conclusion, C# language is used to design, implement and control the portable spectrometer.

3.3 Architecture Design

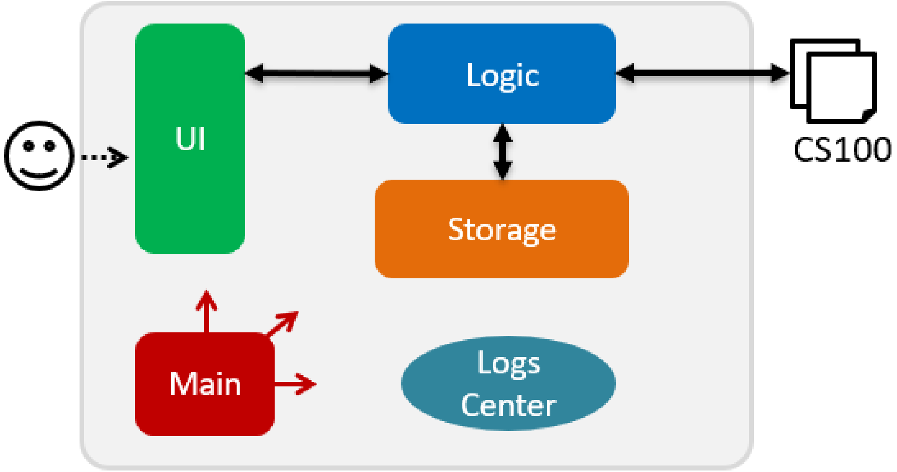


Figure 3.1 Architecture Diagram

Main has only one class which is responsible for,

- When the app launches: Initializes the components in the correct sequence and connects them up with each other.
- When shut down: Shuts down the components and invokes clean-up method where necessary.

LogsCenter: Used by many classes to write log messages to the App's log file. UI: The UI of the application. Logic: The command executor where most of the logical decision making will process. Thorlabs has 25 different logical functions designed for controlling of any Thorlabs products (See Appendix B for the list of functions and their purposes). Each function has a different characteristic which can be considered as pieces of jigsaw puzzles which will be used in the implementation of the program.

4. CURRENT IMPLEMENTATION AND PROGRESS

For the understanding of the different available functions and workability of the program; it is in command lines console. At the moment there are three different functions which the program can carry out. The flow of the program is as follow –

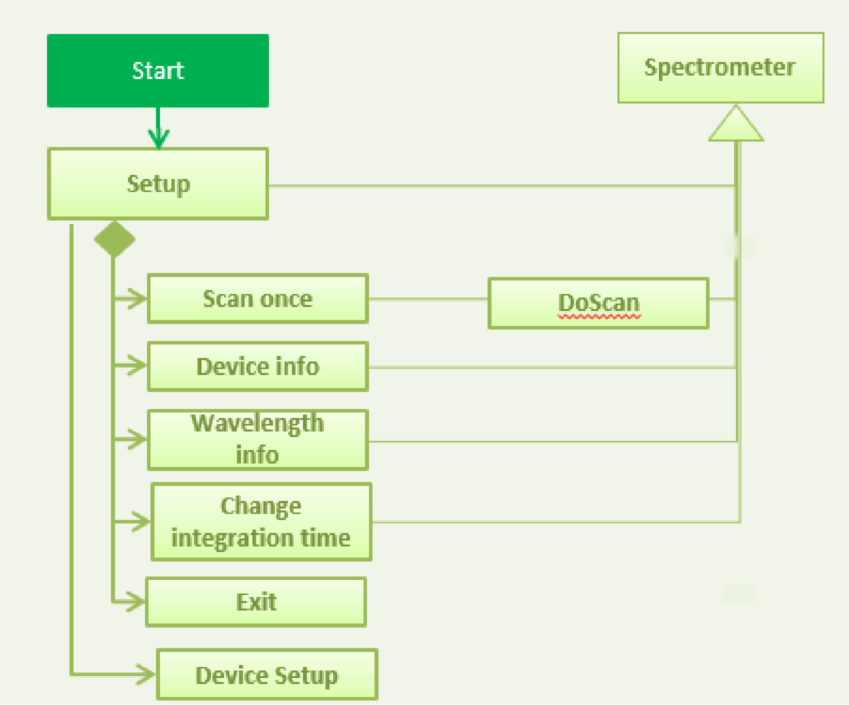
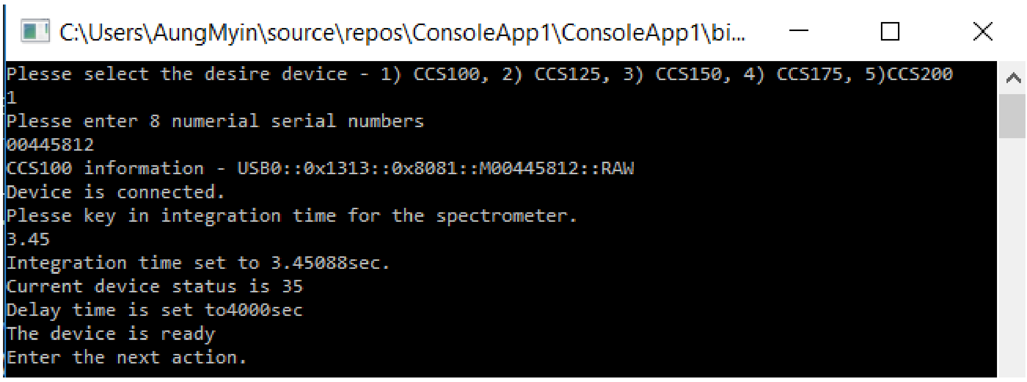


Figure 4.1 Diagram of the flow of the program.

4.1 Device Setup Function

Initial set up is needed by the user to key in the integration time, device model and serial numbers. Based on the models there are 4 choices CCS100, CCS125, CCS150, CCS175 and CCS200 are available for the user to choose. In addition, the serial numbers of the CCD are needed. This information is used to establish a connection with the CCD. Once the connection is established, program will prompt the user to key in the desired integration time. The integration time determines how long the CCD interacts with incoming light. For very bright sources, low integration time are required, while for weak sources, larger integration time should be used. Using the integration time entered, the program will produce the delay time.

This delay time will be used to pause the system to allow the CCD to complete the scanning. This is to avoid the data collected from being incomplete and corrupted as the program do not allow the CCD to fully process. Thus, delay time is required, and it will always be greater than the integration time.


C:\Users\AungMyin\source\repos\ConsoleApp1\ConsoleApp1\bi...

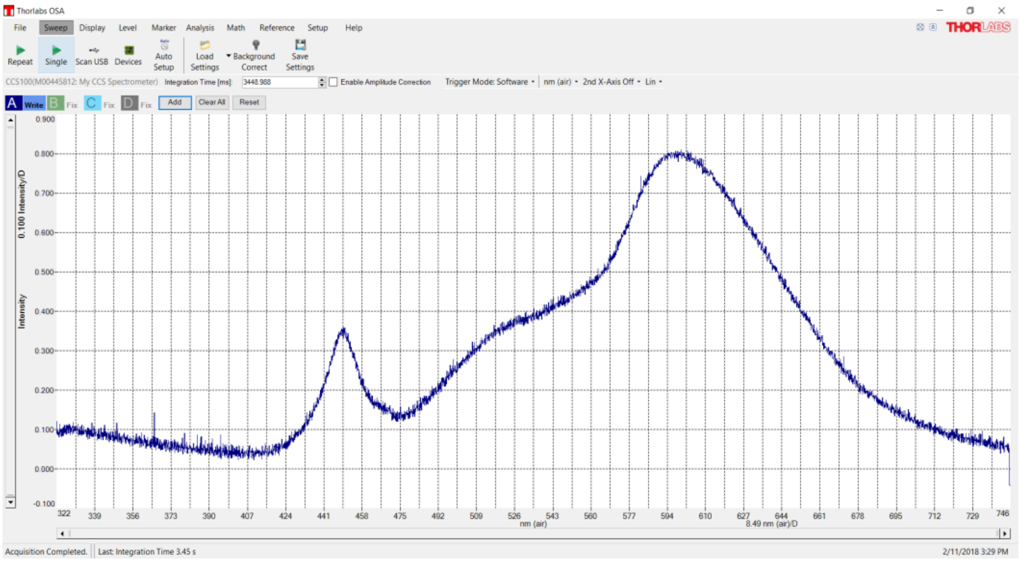
```
Plesse select the desire device - 1) CCS100, 2) CCS125, 3) CCS150, 4) CCS175, 5)CCS200
1
Plesse enter 8 numerial serial numbers
00445812
CCS100 information - USB0::0x1313::0x8081::M00445812::RAW
Device is connected.
Plesse key in integration time for the spectrometer.
3.45
Integration time set to 3.45088sec.
Current device status is 35
Delay time is set to4000sec
The device is ready
Enter the next action.
```

Figure 4.2 Picture of the console after setting up.

Main Functions

There are 4 different functions which the user can execute – scan once, device info, change integration time and wavelength info.

Scan once function: This function enables the spectrometer to scan once and save the processed data in a double variable type array with 3648 different points. User can execute this function by simply typing “scan once” in the command line. Scanned data will be saved in an excel file called ‘Scan.csv’. The 2 diagrams below show the graph produced using the original Thorlabs default program and the currently implemented program. Both scanned the same object; under same condition and with the integration time of 3.45 seconds.



CCS100(M00445812: My CCS Spectrometer) Integration Time [ms]: 3448.988 □ Enable Amplitude Correction Trigger Mode: Software nm (air) 2nd X-Axis Off Lin

Acquisition Completed. | Last Integration Time 3.45 s 2/11/2018 3:29 PM

Figure 4.3 Graph produced using the original Thorlabs default program.

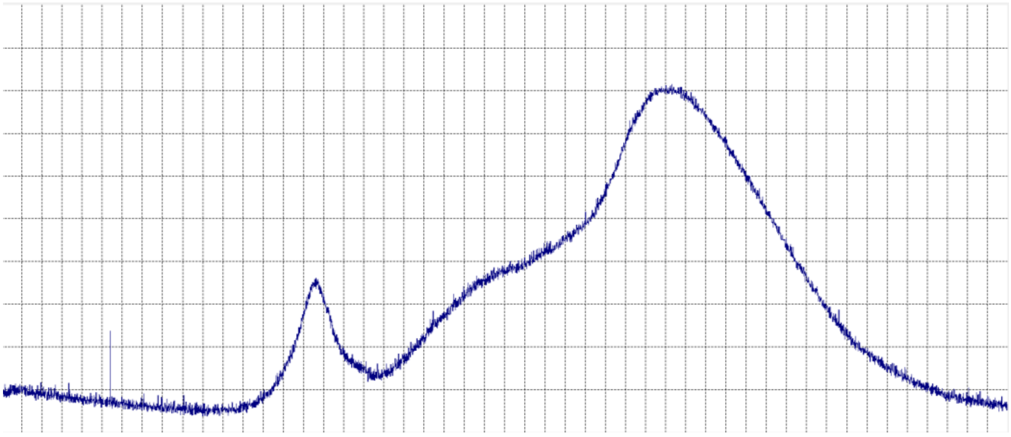
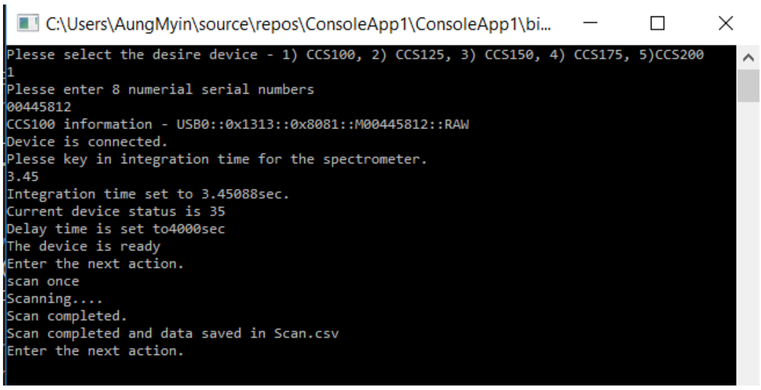


Figure 4.4 Graph produced using the currently implemented program.



C:\Users\AungMyin\source\repos\ConsoleApp1\ConsoleApp1\bi...

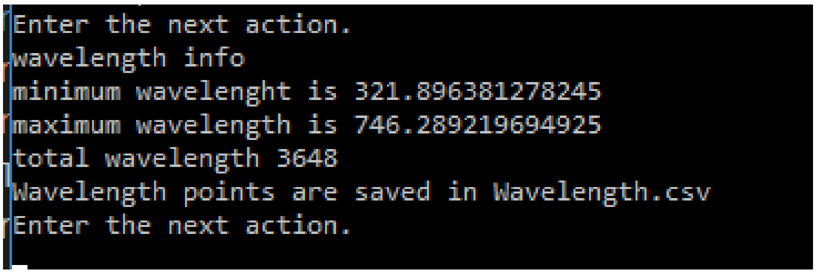
```
Plesse select the desire device - 1) CCS100, 2) CCS125, 3) CCS150, 4) CCS175, 5)CCS200
1
Plesse enter 8 numerial serial numbers
00445812
CCS100 information - USB0::0x1313::0x8081::M00445812::RAW
Device is connected.
Plesse key in integration time for the spectrometer.
3.45
Integration time set to 3.45088sec.
Current device status is 35
Delay time is set to4000sec
The device is ready
Enter the next action.
scan once
Scanning....
Scan completed.
Scan completed and data saved in Scan.csv
Enter the next action.
```

Figure 4.5 Picture of the output of command prompt when “scan once” function is executed.

Thus, this show that the currently implemented program is able to collect the data correctly. In addition, we can also conclude that the delay time set by the program allows the CCD to scan and process the data before saving it into an excel file.

Device information Function: This function enables the user to see the device information and the set up if there is a need to change certain settings. By typing “device info” in the command, the program will display the current device information such us model number, integration and delay time set for the device.

Wavelength information Function: This function enables the user to see the maximum and minimum wavelength of the CCD. In addition, each wavelength points will be saved in an excel file called ‘Wavelenghts.csv’. By typing “wavelength info” in the command prompt the program will display the maximum and minimum wavelength that can be used to scan with the current CCD and save the data into an excel file. The picture below shows the output of the program when the user run wavelength information function.



```
Enter the next action.
wavelength info
minimum wavelenght is 321.896381278245
maximum wavelength is 746.289219694925
total wavelength 3648
Wavelength points are saved in Wavelength.csv
Enter the next action.
```

Figure 4.6 Picture of the output when “wavelength info” function is executed.

Change integration time Function: By keying in “change integration”, the program allows the user to change its desired integration time. It is important to be able to change the integration time without restarting the program. When this function is invoked by the user, the program will change the CCD to new integration time and update the delay time based on the integration time by the user.

Thus, with current implementation, we can conclude that the CCD can be programmed using C# language and is able to communicate.

5. MOVING FORWARD

Current implementation is to test controlling and communicating CCD by using C# language. Furthermore, since console-based program is used for the testing, the scanned data cannot be printed out in the console and instead saved inside an excel file. Thus, the next stage of the project will be the implementation of the UI base on the current logics.

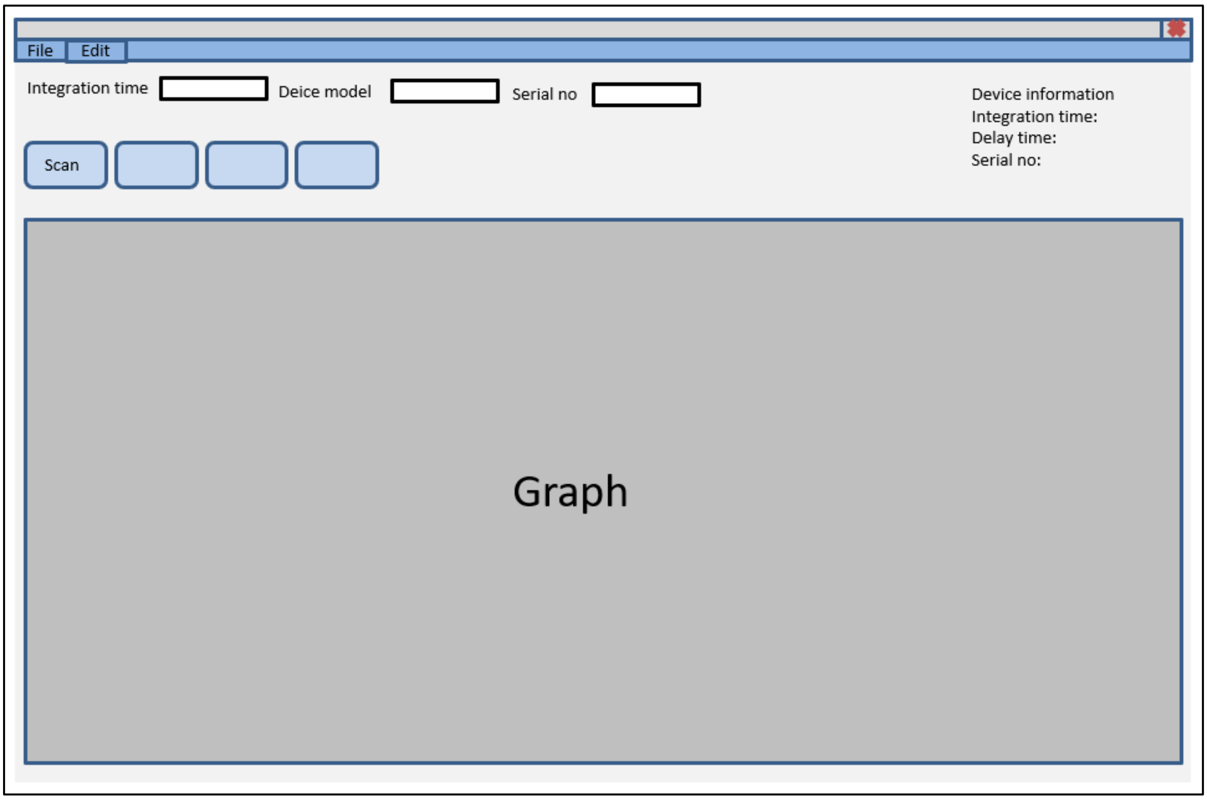


Figure 5.1 Picture of the intended UI layout.

The picture above shows the intended UI design for the future. The data scanned can be printed on the UI once the process is done by the CCD.

The other improvement which can be considered for the future instead of using delay time to pause the thread; using other methods such as priority queues. This is because currently by pausing the thread it may also cause the UI to be unresponsive for other functions. This is one of the limitations which can be overcome by using different priority level.

Rough Timeline

	Dec 17	**Jan 18**	**Feb 18**	**March 18**	**Apr 18**
Designing of UI	◆	◆			
Integration of the logical and UI		◆	◆		
Testing and Debugging			◆	◆	◆
Completion of final report					◆

Table 5.1 Proposed schedule for the remaining half of the year.

REFERENCES

[1] Skoog DA, Holler FJ, Crouch SR. Principles of instrumental analysis. 6th ed. Cengage Learning, 2006.

[2] Willard HH, Meritt Jr LL, Dean JJ, Settle Jr FA. Instrumental methods of analysis. 7th ed. New Delhi: CBS Publisher & Distributors; 1988.

[3] Wu Guozhen. RAMAN SPECTROSCOPY an Intensity Approach, Science Press, Beijing, 2014

[4] S. Zhang, Raman spectroscopy and its application in nanostructures. Hoboken, N.J.: Wiley, 7-8, 2013.

[5] Zhang, Shu-Lin. Raman Spectroscopy and Low-dimensional Nano-semiconductors, Science Press, Beijing, 2008.

[6] Settle FA. Handbook of instrumental techniques for analytical chemistry. New Jersey: Prentice, Inc., 1997.

[7] G. Bumbrah and R. Sharma, "Raman spectroscopy – Basic principle, instrumentation and selected applications for the characterization of drugs of abuse", Egyptian Journal of Forensic Sciences, vol. 6, no. 3, pp. 209-215, 2016.

[8] Anon, (2018). [online] Available at: https://www.upwork.com/hiring/development/c-sharp-vs-c-plus-plus/ [Accessed 24 Oct. 2018].

[9] Travis, J. and Kring, J. (2018). What Exactly Is LabVIEW, and What Can It Do for Me? | Introduction to Graphical Programming with LabVIEW | InformIT. [online] Informit.com. Available at: http://www.informit.com/articles/article.aspx?p=662895&seqNum=3 [Accessed 24 Oct. 2018].

Appendix A

Technical Data of CCS100

Optical Specs	
Wavelength Range	350 – 700nm
Spectral Resolution	<0.5nm FWHM @ 435nm
Slit (WxH)	20 µm x 2 mm
Grating	1200 Lines/mm, 500nm Blaze
Sensor Specs	
Detector Range (CCS Chip)	350 - 1100nm
CCD Pixel Size	8 µm x 200 µm (8 µm pitch)
CCD Sensitivity	160 V / (lx · s)
CCD Dynamic Range	300
CCD Pixel number	3648
Resolution	10 px/nm
Integration Time	10 µs – 10 s [1]
Scan Rate Max.	200 Scans/s [2]
S/N ratio	≤ 2000:1
External Trigger	
Fibre Connector	SMA 905
Trigger Input	SMB
Trigger Signal	TTL
Trigger Frequency Max.	100 Hz
Trigger Pulses Length Min.	0.5 µs
Trigger Delay	8.125 µs ± 125 ns
General Specs	
Interface	Hi-Speed USB2.0 (480 Mbit/s)
Dimensions (LxWxH)	122 x 80 x 30 mm
Weight	< 0.4 kg

All technical data are valid at 23 ± 5°C and 45 ± 15% rel. humidity (non-condensing)

1) integration time 5 ms

3) software allows to set up to 60 s. Hot pixels and noise may increase drastically. See section Integration Time

Operating Temperature	0 ... +40 °C
Storage Temperature	-40 ... +70 °C
Relative Humidity	Max. 80% up to 31 °C; decreasing to 50% at 40 °C
Operation Altitude	< 3000 m

APPENDIX B

Configuration Functions –

tlccs_init

tlccs_setIntegrationTime

tlccs_getIntegrationTime

Action/Status Functions -

tlccs_startScan

tlccs_startScanCont

tlccs_startScanExtTrg

tlccs_startScanContExtTrg

tlccs_getDeviceStatus

Data Functions -

tlccs_getScanData

tlccs_getRawScanData

tlccs_setWavelengthData

tlccs_getWavelengthData

tlccs_getUserCalibrationPoints

tlccs_setAmplitudeData

tlccs_getAmplitudeData

Utility Functions -

tlccs_identificationQuery

tlccs_revisionQuery

tlccs_reset

tlccs_selfTest

tlccs_errorQuery

tlccs_errorMessage

tlccs_setUserText

tlccs_getUserText

tlccs_setAttribute

tlccs_getAttribute

tlccs_close