TOWARDS DEVELOPMENT OF A PORTABLE RAMAN SPECTROMETER

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

M.Tech.
in
Instrumentation and Electronics

by

Mr. KALLOL BERA

Reg. no. - *128055 of 2014-15*

Exam Roll no. – *M4IEE19008*

Dept. of Instrumentation and Electronics Engg.

Jadavpur University

Kolkata-700098

May 2019

CERTIFICATE

This is to certify that the Thesis entitled "***TOWARDS DEVELOPMENT OF A PORTABLE RAMAN SPECTROMETER***" is being submitted by **Mr. Kallol Bera** with Reg. No. *128055 of 2014-15*, Class Roll No. *001711103008* and Examination Roll No. *M4IEE19008* in partial fulfillment of the requirement for the award of the degree of M.Tech. in Instrumentation and Electronics to Jadavpur University, Kolkata is a record of bonafide work carried out by him under my guidance and supervision. The results presented in this thesis have been verified and are found to be satisfactory.

The results embodied in this thesis have not been submitted to any other University for the award of any other degree or diploma.

Rajib Bandyopadhyay 20/5/19
....
(Prof. Rajib Bandyopadhyay)
Supervisor
Dept. of Instrumentation and Electronics
Jadavpur University, Kolkata-98

Dept. of Inst. & Electronics Engg.
Jadavpur University
Salt Lake Campus
Kolkata-98

COUNTERSIGNED BY:

Runu Banerjee 20/05/19
....
(Prof. Runu Banerjee Roy)
Head of the Dept.
Dept. of Instrumentation and Electronics
Jadavpur University, Kolkata-98

Head
Dept. of Inst. & Electronics Engg.
Jadavpur University
Salt Lake Campus
Kolkata-98

C. Bhattacharya
....
(Prof. Chiranjib Bhattacharjee)
Dean - Faculty of Engineering and Technology
Jadavpur University, Kolkata-32

Abstract

This project aims to design, develop and build a low-cost portable Raman spectrometer. The spectrometer is based on a 100 mW, 785 nm red laser and an assembly of optical components to facilitate 180° back-scattering so that both solid and liquid samples can be used. An off-the-shelf detector setup was employed as a standard but due to its expensiveness an approach was made to design a detector with its associated optical and electronic components at an affordable cost. The complete design process, broadly categorised in - developing the laser power supply, assembling optical components for back-scattering and designing the interface electronics for the CCD detector, is explained in great details and accompanied with 3D models, wherever needed, for better understanding. The spectrum is projected with a diffraction grating on the detector, sampled by a 16-bit A/D converter to the micro-controller memory and finally transmitted to the computer via USB. In the computer it is digitally processed and displayed in a graphical user interface. Not only the optical setup and electronic circuit used to process the analog signal are described in this work, but also the software applications enabling the user to evaluate and save the measured spectra have been discussed.

Acknowledgements

While bringing out this thesis to its final form, I came across a number of people whose contributions in various ways helped my field of research and they deserve special thanks. It is a pleasure to convey my gratitude to all of them.

I take this opportunity to express a deep sense of gratitude towards my guide **Prof. Rajib Bandyopadhyay**, for providing excellent guidance, encouragement and inspiration throughout the project work. Without his invaluable guidance, this work would never have been a successful one. I feel proud to say that I had the opportunity to work with an exceptionally experienced Professor like him.

I am also highly grateful to **Prof. Kalyan Majumder**, Dept. of Instrumentation and Electronics, Jadavpur University, Kolkata for giving me suggestions and support from an early stage of this research and providing me extraordinary experiences throughout the work. I specially acknowledge him for his advice as and when required during this research.

I would also like to thank **Dilip Sing** and **Somdeb Chanda**, scholars at the same University who are recently working in this field of research, for their valuable suggestions and helpful discussions and lending me their invaluable knowledge and support. I am greatly indebted to all the people mentioned above.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AC	Alternating Current
ADC	Analog to Digital Converter
APB	Advanced Peripheral Bus
ARM	Advanced RISC Machines
ASIC	Application Specific Integrated Circuits
CCD	Charge Coupled Device
CDC	Communication Device Class
CLB	Configurable Logic Blocks
CMOS	Complementary Metal-Oxide Semiconductor
DAC	Digital to Analog Converter
DC	Direct Current
DMA	Direct Memory Access
DSLR	Digital Single-Lens Reflex
EM	Electro - Magnetic
EOS	Electro-Optical System
ESD	Electro-Static Discharge
FPGA	Field Programmable Gate Arrays
FWHM	Full Width at Half Maximum
GUI	Graphical User Interface
HDL	Hardware Description Language
HPLC	High Performance Liquid Chromatography
IDE	Integrated Development Environment
IR	InfraRed
LCD	Liquid Crystal Display
LUT	Look Up Table
MCU	Micro-Controller Unit
MSps	Million Samples per second
NA	Numerical Aperture
NTC	Negative Temperature Coefficient
OTG	On-The-Go
PC	Personal Computer
PCB	Printed Circuit Board
PWM	Pulse Width Modulation
RISC	Reduced Instruction Set Computer
RMSE	Root Mean Squared Error
SMA	SubMiniature version A
SMSR	Side-Mode Supression Ratio
SNR	Signal-to-Noise ratio
SPST	Single Pole Single Throw
SRAM	Static Random Access Memory
TE	Thermo-Electric
TEC	Thermo-Electric Cooling
USB	Universal Serial Bus
VHDL	Very High-speed-IC Hardware Desciption Language

Chapter 1

1 Introduction

1.1. What is Raman spectroscopy?

When light is scattered from a molecule or crystal, most photons are elastically scattered. The scattered photons have the same energy (frequency) and, therefore, wavelength, as the incident photons. However, a small fraction of light (approximately 1 in 10^7 photons) is scattered at optical frequencies different from, and usually lower than, the frequency of the incident photons. The process leading to this inelastic scatter is termed the Raman effect. Raman scattering can occur with a change in vibrational, rotational or electronic energy of a molecule. If the scattering is elastic, the process is called Rayleigh scattering. If it's not elastic, the process is called Raman scattering.

1.2. Birth and development of Raman Spectroscopy

The Indian scientist Chandrasekhara Venkata Raman was fascinated by the deep blue colour of the Mediterranean Sea. However, he was unsatisfied with Lord Rayleigh's explanation that the colour of the sea was just a reflection of the colour of the sky, which was explained by the classical theory of light scattering for unchanged frequency in 1871. Thinking on the matter and working extensively on a series of measurements of light scattered by liquids as well as by some solids, Raman and his group in Calcutta managed in 1928 to show that the colour of the sea was the result of the scattering of sunlight by the water molecules[1]. However, the inelastic light scattering phenomenon had already been predicted a few years earlier by the Austrian physicist Adolf Smekal[2]. Smekal proposed that photons could be scattered inelastically by molecules and would consist of shorter and longer wavelengths in addition to the origin wavelength. Smekal also showed that the frequency shift between the incident and scattered light is due to the energy difference between two states of the molecule. Meanwhile, two Russian physicists, Landsberg and Mandelstam observed light scattering with change of frequency by investigating Brillouin scattering from quartz. After all, the theoretical basis had been provided by Smekal, though his work was not widely known at that time. Moreover, the results of Landsberg and Mandelstam had been published after Raman's work was in print. For these reasons, Raman was considered to be the first who explained the phenomenon of the light scattering effect which was named after him and that earned him the Nobel prize in 1930.

The development of Raman spectroscopy, however, was relatively slow due to many reasons. First, the Raman effect is very weak. Typically, only one part in a thousand of the total intensity of the incident light is Rayleigh scattered, while for Raman scattering this value drops to one part in a million. Thus, as the Raman effect is quite weak, it is a major challenge to attenuate the light that is elastically scattered in order to detect the inelastically scattered Raman light. The second reason is that in all of the early light-scattering studies, the excitation source was sunlight and the samples used in the analysis were liquids. Furthermore, much of the early work in Calcutta was done by visual observation of colour rather than precise measurements of the light wavelength, which Raman has described as being a plentiful task to do. However, these difficulties have been overcome and things changed dramatically after the discovery of the lasers in 1960. The birth of the laser technique thus stimulated the traditional field of molecular spectroscopy and Raman spectroscopy in various ways. It also allowed the molecular spectroscopist to record and analyse Raman spectra of a great variety of compounds, from deeply coloured materials to highly fluorescent molecules.

1.3. Motivation

Raman spectroscopy is an ideal tool for chemical analysis due to its unique advantages over other analytical techniques. It is non-destructive and non-contacting method of obtaining the fingerprint spectrum of materials, requiring no special sample analysis. The sample can be as small as 1-2 μm across. A short amount of measuring time, normally a few seconds, is required to obtain a Raman spectrum. Thus, it can be used to monitor chemical reactions in real time.

Raman spectroscopy has become more popular due to its new prospective field applications in forensic sciences, war against terrorism, environment protection and other field chemical analysis. Despite the fact that Raman spectroscopy has so many advantages, it still is not widespread in use. The main reason is the high cost typically associated with Raman analyzer systems. Currently, high-resolution and high signal-to-noise ratio Raman analyser costs $4000 and above. On the other hand, there is a market need for low-cost Raman analyser systems as general laboratory tools. However, those systems usually are equipped with low-resolution, low-power visible lasers, and low signal-to-noise ratios, which is not adequate enough to perform any high-performance chemical analysis. Therefore, better-resolution and lower-cost Raman systems are key to increasing and enabling greater acceptance and usage of Raman spectroscopy. Hence, in this project an attempt was made to develop a low cost, portable Raman system along with a software application for spectral acquisitions.

Most of the Raman systems that are used today are either so large and bulky that it finds place only inside laboratories or are so expensive that it questions affordability. For instance, take the example of assessment tea quality. Instead of taking the tea leaves for quality testing at a laboratory using conventional methods like HPLC how nice it would be if the test can be done in-situ by just touching a probe over the leaves. Several such commercial devices are already in the market. A small comparison of them is given in Table 1-1. Most of them are above $30,000. Therefore, to make an affordable device with sufficient accuracy by using minimum number of components is the need of the hour so that it could serve satisfactorily both in industrial sectors and research laboratories. This thesis tries to offer a little contribution for this nobler cause.

Table 1-1 - Comparison of different Raman systems available in market [Price of some items may not have been updated][3]

Raman hand-held systems	Specifications	Price
NanoRam Handheld RAMAN System, B&W Tek [4] 	• ***Laser Output Power*** *- 300 mW Max Adjustable in 10% Increments* • ***Excitation Wavelength*** *- 785 nm* • ***Spectral Range*** *- 176 cm^{-1} to 2900 cm^{-1}* • ***Spectral Resolution*** *- About 9 cm^{-1} at 912 nm* • ***Detector*** *- TE Cooled Linear CCD Array* • ***Power requirements*** *- Battery Rechargeable Li-ion, Greater Than 4 hrs Operation* • ***Electrical*** *- AC Adaptor: 12V DC, 2A Minimum* • ***Dimensions*** *- 22 x 10 x 5 cm (8.8 x 3.9 x 2.0")*	$ 46,405

Thermo Scientific™ TruScan™ RM Handheld Raman Analyzer [5] 	• ***Laser Output*** - *250 mW ±25* • ***Laser Excitation Wavelength*** *– 785 nm ± 0.5* • ***Spectral Range*** *- 250 to 2875cm^{-1}* • ***Spectral Resolution*** *- 8 to 10.5cm^{-1}* • ***Collection Optics*** *- NA = 0.33, 18mm working distance; 0.2 to 2.5mm spot size* • ***External Power Supply*** *- DC Wall Adapter, 100-240VAC 50/60Hz* • ***Dimensions (L x W x H)*** *- 8.2 x 4.2 x 1.7 in.* • ***Weight*** *- 0.9kg*	$56,780
Raman Systems Portable-PinPointer™ [6] 	• ***Excitation Wavelengths*** *- 785 nm* • ***Laser Power*** *- 5-300 mW* • ***Spectral Range*** *- 200 - 3000 cm^{-1}* • ***Spectral Resolution*** *- 12 cm^{-1}* • ***Optics*** *- NA = 0.22; Sample size = 0.1 to 0.3mm* • ***Power*** *- Rechargeable battery, >4 hrs./charge* • ***Size*** *- 8.5" x 4.3" x 2.5"* • ***Weight*** *- 3 lbs.*	$26,912
Optosky ATR6500 4th Gen Handheld Raman Analyzer [7] 	• ***Operating System*** *- Android* • ***Excitation Wavelength*** *- 785 ± 0.5nm* • ***Wavenumber Range*** *- 200-4000 cm^{-1}* • ***Resolution*** *- 10 cm^{-1}* • ***Touch Screen*** *- 5.5", 1920×1080, c* • ***Weight*** *- 450g* • ***Size*** *- 6.7"×3.1"×1.2"* • ***Interface*** *- WIFI, USB Type-C, Bluetooth, GSM* • ***Battery*** *- 4-6 hour continuous operation*	$19,800
Metrohm Mira M-1 [6] 	• ***Barcode Scanner*** *- false* • ***Height (mm)*** *- 39.0* • ***Laser class according to EN 60825-1*** *- Mira M-1 Basic Class 1* • ***Laser output power (mW)*** *- ≤ 100 mW* • ***Laser Wavelength (nm)*** *- 785.0* • ***Length (mm)*** *- 131.0* • ***Width (mm)*** *- 85.0* • ***Resolution (cm^{-1}) (FWHM)*** *- 16-18* • ***Spectral Range (cm^{-1})*** *- 400-2300* • ***Touchscreen Display*** *- 2.7" Resistive oder 6.9 cm* • ***Weight (grams)*** *- 650.0*	> $17,500 (basic) – $32,000 (advanced)

1.4. Literature review

With the advent of the continuous wave laser, Raman spectroscopy has become a reliable and a more convenient tool for structural and analytical chemistry. Earlier designs of Raman systems generally include a double monochromator to get rid of the stray light. Each surface in the pathway of a light beam, both optical surfaces and light shields, causes elastic scattering of the radiation and the primary contribution derives from optical surfaces themselves. Marechal[8] has shown that scattering from optically polished mirror surfaces obeys the fourth power law as a function of frequency, whereas that from ruled gratings is proportional both to the square of the frequency and the square of the grating spacing error. Thus, independently of exciting frequency, the quality of the grating plays an important role in the minimization of stray light. Several optimizations were suggested by Allemand[9] for the collection of Raman light and relaying to the spectrometer with a double monochromator setup. It was found that if the slit width is less than 100 μm in a well-designed 1-m Czerny-Turner double monochromator of the subtractive type, the resolution is improved as compared to a single monochromator. However, portability of Raman systems does not go hand in hand with the use of double monochromator owing to its large size.

The optical setup can be done in two possible ways - 90° scattering and back-scattering. In former case the scattered radiation is taken perpendicular to the direction of the laser radiation while in the latter case it is taken at 180° to the laser radiation. Though the back-scattering layout is slightly more complicated than the 90°- scattering layout but the signal provided in this configuration is typically larger than in 90°-scattering because the excitation and collection volumes in the sample are automatically aligned in this case. But the back-scattering configuration has typically more influence from stray light due to reflections of the laser on the lenses and cuvette[10]. In this design no notch filters were used but colour filter. C. Mohr[11] and others made an inexpensive modular Raman spectrometer based on a 4 mW, 532 nm green laser pointer and a compact monochromator equipped with glass fiber optical connections, linear CCD detector array, and a USB computer connection and power supply costing about $5000. With back-scattering configuration and use of notch filter a spectral resolution of about 20 cm^{-1} was achieved with a cut off wave number below 340 cm^{-1}.

The most serious problems in design of the Raman spectrometers owing to low level of the useful Raman signal and existence of strong interfering optical signals, mainly external illumination and fluorescence induced by the laser beam can be overcome by proper choice of excitation laser wavelength, size of the sample and optical properties of investigated materials. Moreover, lasers for portable Raman spectrometers should have sufficient output power, narrow bandwidth of emission line, robust construction as well as small weight, size and moderate power consumption. Thus, diode lasers are the best solution. At present, most of the state-of-the-art portable Raman spectroscopic devices use the diode laser of the wavelength equal to 785 nm working in CW (continuous wave) mode as an excitation source. This near-infrared excitation wavelength, enabling quite efficient fluorescence reduction[12], is recognized as an industry standard, also this laser has sufficient output power (a few hundred milliwatts), compact size, long lifetime (at least 10,000 hours) as well as a relatively low price[13].

Some home-made approaches on the development of Raman systems are also worth mentioning. One of them is a low cost, about $3000, Raman spectrometer using consumer grade EOS DSLR camera, a green laser pointer at 532 nm for signal excitation of transparent liquid samples, in transmission mode. A spectral resolution of 15 cm^{-1} was achieved with image processing techniques. The author calls it an 'alternative scientific instruments' having good performance and at low cost though it does not fall strictly under the category of portable Raman systems.

1.5. Objective and scope of the Thesis

The objective of this thesis is to design, develop and build different components of a compact and portable Raman spectrometer. These components are –

- Laser power supply
- Optical assembly
- Detector and interface electronics
- A PC-based application for spectral acquisition

The design and development of various stages of a Raman spectrometer specially the scattering configuration hardware or the optical head along with the detector has been thoroughly discussed. For proper working of a Raman system a stable power supply for the excitation source is very important. After going through various literatures, it was found out that a 785 nm laser will suit our purpose and also fluorescence can be avoided. A user customizable constant current source with short-circuit protection was designed with proper temperature controller circuit for stable operation. In building the optical head 180° back-scattering technique is employed which greatly increases the signal-to-noise ratio. Though, problems may be caused by stray light due to reflections from surfaces of optical components but it can be removed using proper filters. The developed optical head measures 20 cm in length and around 5 cm in width and height. It is attached to the spectrometer by a small optical fibre. The system is portable but several modifications and addition of components are needed to actually make it a hand-held system.

A detector comprising of image sensor and its associated electronics also have been designed and built in our laboratory. Many challenges have been faced in finding a suitable image sensor for our application and in making it work properly without any added complex electronic circuits and components. The design process for the same has been discussed in great details in Chapter 4. A working software application for acquiring the spectra was also developed. Though it is the first version but due to its simple layout and user-friendliness using the application will be a piece of cake for any new user. Though there are many limitations both on the hardware and software ends but they still provide acceptable results. Upon successful completion of this work a low-cost, hand-held Raman system can be seen which will not only help greatly in agricultural industries for quality assessment but also serve academic scholars in their research.

A brief summary of the subsequent chapters is given below:

- **Chapter 2**: Discusses the theory behind Raman scattering in lucid language including Classical and Quantum explanations.
- **Chapter 3**: Discusses about the design methodologies of laser power supply, optical head for Raman back-scattering and spectral acquisition.
- **Chapter 4**: Discusses about the design and development of the interface electronics and a software application for spectral acquisition from a CCD detector.
- **Chapter 5**: Gives concluding remarks and talks about future scopes.

Chapter 2

2 Theory

2.1. Electromagnetic radiation

All light (including infrared) is classified as electromagnetic radiation and consists of alternating electric and magnetic fields and is described classically by a continuous sinusoidal wave like motion of the electric and magnetic fields. Typically, for IR and Raman spectroscopy we will only consider the electric field and neglect the magnetic field component. Figure 2.1 depicts the electric field amplitude of light as a function of time.

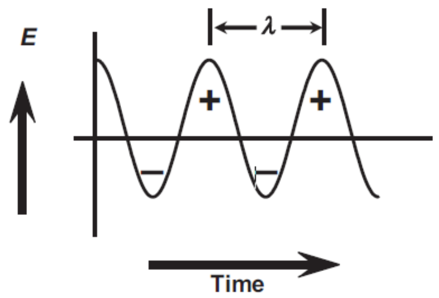


Figure 2.1 - The amplitude of the electric vector of electromagnetic radiation as a function of time. The wavelength is the distance between two crests.

The important parameters are the wavelength (λ, length of 1 wave), frequency ($\bar{\nu}$, number cycles per unit time), and wavenumbers (n, number of waves per unit length) and are related to one another by the following expression:

$$\bar{\nu} = \frac{\nu}{(c/n)} = \frac{1}{\lambda} \qquad (2.1)$$

where c is the speed of light and n the refractive index of the medium it is passing through. In quantum theory, radiation is emitted from a source in discrete units called photons where the photon frequency, ν, and photon energy, E_p, are related by

$$E_p = h\nu \qquad (2.2)$$

where h is Planck's constant (6.6256×10^{-27} erg sec). Photons of specific energy may be absorbed (or emitted) by a molecule resulting in a transfer of energy. In absorption spectroscopy this will result in raising the energy of molecule from ground to a specific excited state as shown in Figure 2.2. Typically, the rotational (E_{rot}), vibrational (E_{vib}), or electronic (E_{el}) energy of molecule is changed by ΔE:

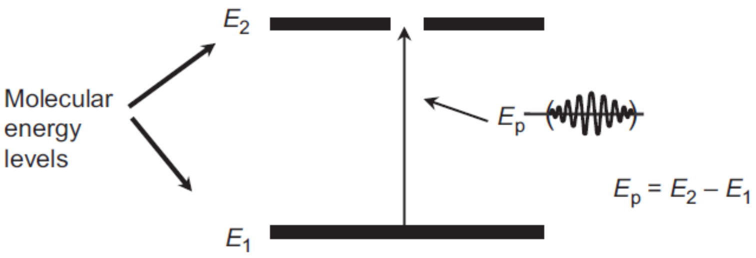


Figure 2.2 - Absorption of electromagnetic radiation

$$\Delta E = E_p = h\nu = hc\bar{\nu} \tag{2.3}$$

In the absorption of a photon the energy of the molecule increases and ΔE is positive. To a first approximation, the rotational, vibrational, and electronic energies are additive:

$$E_T = E_{el} + E_{rot} + E_{vib} \tag{2.4}$$

We are concerned with photons of such energy that we consider E_{vib} alone and only for condensed phase measurements. Higher energy light results in electronic transitions (E_{el}) and lower energy light results in rotational transitions (E_{rot}). However, in the gas-state both IR and Raman measurements will include $E_{vib} + E_{rot}$.

2.2. Internal Degrees of Freedom

The molecular motion that results from characteristic vibrations of molecules is described by the internal degrees of freedom resulting in the well-known $3n - 6$ and $3n - 5$ rule-of-thumb for vibrations for non-linear and linear molecules, respectively. Figure 2.3 shows the fundamental vibrations for the simple water (non-linear) and carbon dioxide (linear) molecules.

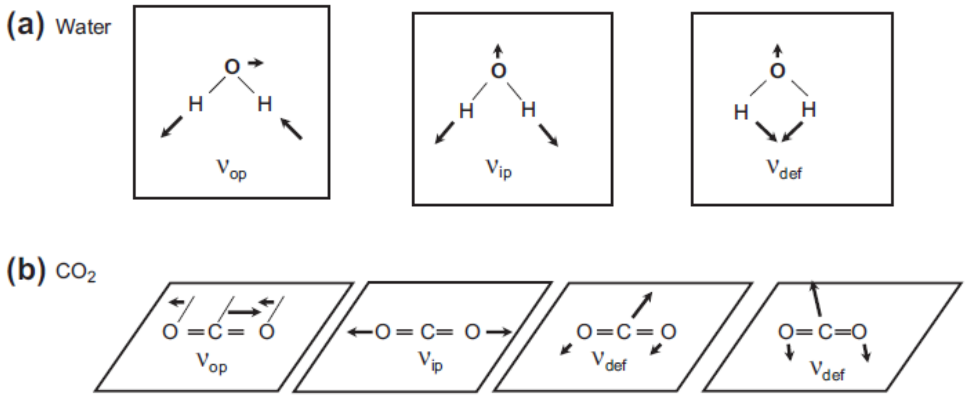


Figure 2.3 - Molecular motions which change distance between atoms for H2O and CO2.

The internal degrees of freedom for a molecule define n as the number of atoms in a molecule and define each atom with 3 degrees of freedom of motion in the X, Y, and Z directions resulting in $3n$ degrees of motional freedom. Here, three of these degrees are translation, while three describe rotations. The remaining $3n - 6$ degrees (non-linear molecule) are motions, which change the distance between atoms, or the angle between bonds. A simple example of the $3n - 6$ non-linear molecule is water (H_2O) which has $3(3) - 6 = 3$ degrees of freedom. The three vibrations include an in-phase and out-of-phase stretch and a deformation (bending) vibration. Simple examples of $3n - 5$ linear molecules include H_2, N_2, and O_2 which all have $3(2) - 5 = 1$ degree of freedom. The only vibration for these simple molecules is a simple stretching vibration. The more complicated CO_2 molecule has $3(3) - 5 = 4$ degrees of freedom and therefore four vibrations. The four vibrations include an in-phase and out of- phase stretch and two mutually perpendicular deformation (bending) vibrations.

The molecular vibrations for water and carbon dioxide as shown in Figure 2.3 are the normal mode of vibrations. For these vibrations, the Cartesian displacements of each atom in molecule change periodically with the same frequency and go through equilibrium positions simultaneously. The centre of the mass does not move and the molecule does not rotate. Thus, in the case of harmonic oscillator, the Cartesian coordinate displacements of each atom plotted as a function of time is a sinusoidal wave. The relative vibrational amplitudes may differ in either magnitude or direction.

Figure 2.4 shows the normal mode of vibration for a simple diatomic such as HCl and a more complex totally symmetric CH stretch of benzene.

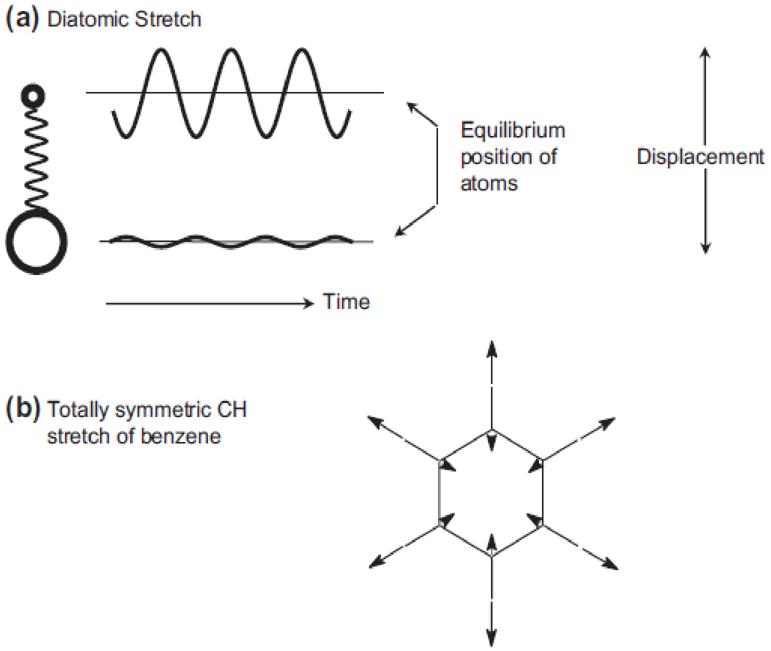


Figure 2.4 - Normal mode of vibration for a simple diatomic such as HCl (a) and a more complex species such as benzene (b). The displacement versus time is sinusoidal, with equal frequency for all the atoms. The typical Cartesian displacement vectors are shown for

2.3. Classic harmonic oscillator

The molecular vibrations responsible for the characteristic bands observed in Raman spectra can be understood by considering a simple model derived from classical mechanics. Figure 2.5 depicts a diatomic molecule with two masses m_1 and m_2 connected by a massless spring. The displacement of each mass from equilibrium along the spring axis is X_1 and X_2. The displacement of the two masses as a function of time for a harmonic oscillator varies periodically as a sine (or cosine) function. Although each mass oscillates along the axis with different amplitudes, both atoms share the same frequency and both masses go through their equilibrium positions simultaneously. The observed amplitudes are inversely proportional to the mass of the atoms which keeps the centre of mass stationary.

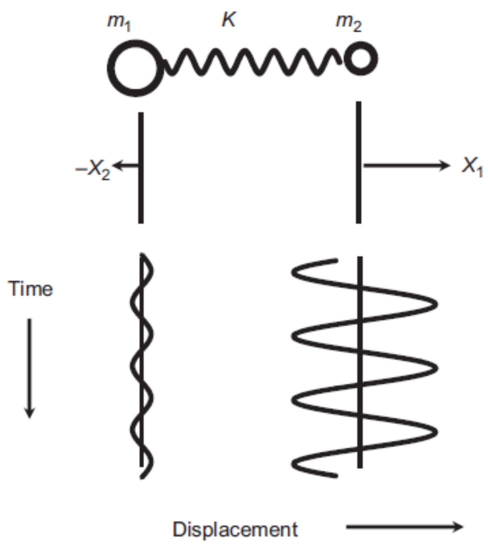


Figure 2.5 - Motion of a simple diatomic molecule. The spring constant is K, the masses are m1 and m2, and X1 and X2 are the displacement vectors of each mass from equilibrium where the oscillator is assumed to be harmonic.

The classical vibrational frequency for a diatomic molecule is:

$$\nu = \frac{1}{2\pi}\sqrt{K\left(\frac{1}{m_1}+\frac{1}{m_2}\right)} \qquad (2.5)$$

where K is the force constant in dynes/cm and m_1 and m_2 are the masses in grams and n is in cycles per second. This expression is also encountered using the reduced mass where

$$\mu = \frac{m_1 m_2}{m_1 + m_2} \qquad (2.6)$$

In vibrational spectroscopy wavenumber units, $\bar{\nu}$ (waves per unit length) are more typically used

$$\bar{\nu} = \frac{1}{2\pi c}\sqrt{K\left(\frac{1}{m_1}+\frac{1}{m_2}\right)} \qquad (2.7)$$

where $\bar{\nu}$ is in waves per centimeter and is sometimes called the frequency in cm^{-1} and c is the speed of light in cm/s.

If the masses are expressed in unified atomic mass units (u) and the force constant is expressed in millidynes/Ångström then:

$$\bar{\nu} = 1303\sqrt{K\left(\frac{1}{m_1}+\frac{1}{m_2}\right)} \qquad (2.8)$$

where $1303 = \frac{\sqrt{N_a \times 10^5}}{2\pi c}$ and N_a is Avogadro's number (6.023×10^{23} $mole^{-1}$).

This expression shows that the observed frequency of a diatomic oscillator is a function of

a) the force constant K, which is a function of the bond energy of a two-atom bond
b) the atomic masses of the two atoms involved in the vibration.

The general wavenumber regions for various diatomic oscillator groups are shown in Table 2-1, where Z is an atom such as carbon, oxygen, nitrogen, sulphur, and phosphorus.

Table 2-1 - General wavenumber regions for various simple diatomic oscillator groups

Diatomic oscillator	**Region (cm^{-1})**
Z–H	4000 – 2000
C≡C, C≡N	2300 – 2000
C=O, C=N, C=C	1950 – 1550
C–O, C–N, C–C	1300 – 800
C–Cl	830 – 560

2.4. Quantum mechanical harmonic oscillator

For the classical harmonic oscillation of a diatomic the potential energy (PE) is given by

$$PE = \frac{1}{2}kx^2 \qquad (2.9)$$

A plot of the potential energy of this diatomic system as a function of the distance, X between the masses, is thus a parabola that is symmetric about the equilibrium internuclear distance, X_e. Here, X_e is at the energy minimum and the force constant, k is a measure of the curvature of the potential well near X_e.

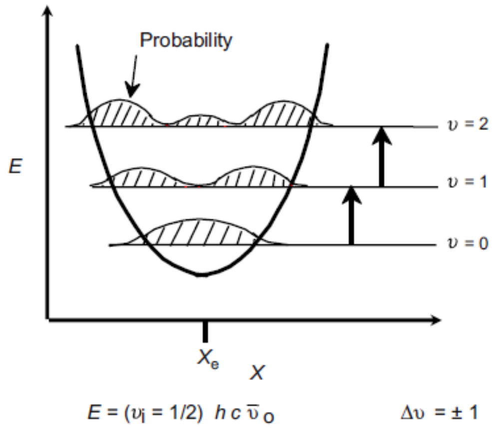


Figure 2.7 - Potential energy, E, versus internuclear distance, X, for a diatomic harmonic oscillator.

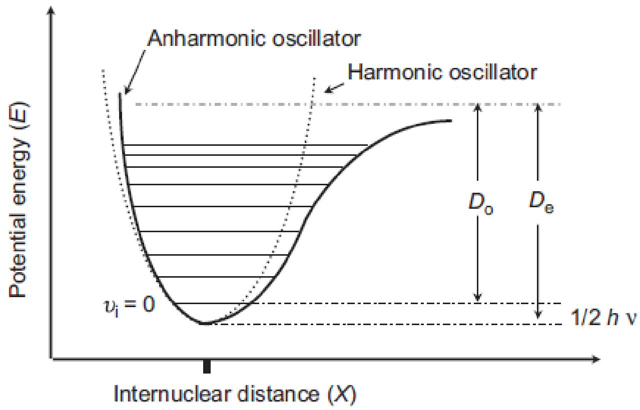


Figure 2.6 - The potential energy diagram comparison of the anharmonic and the harmonic oscillator. Transitions originate from the v=0 level, and Do is the energy necessary to break the bond.

From quantum mechanics we know that molecules can only exist in quantized energy states. Thus, vibrational energy is not continuously variable but rather can only have certain discrete values. Under certain conditions a molecule can transit from one energy state to another ($\Delta v = \pm 1$) which is what is probed by spectroscopy. Figure 2.7

shows the vibrational levels in a potential energy diagram for the quantum mechanical harmonic oscillator. In the case of the harmonic potential these states are equidistant and have energy levels E_i given by

$$E_i = \left(v_i + \frac{1}{2}\right) h\nu \quad , v_i = 0,1,2 \ldots \tag{2.10}$$

Here, ν is the classical vibrational frequency of the oscillator and v_i is a quantum number which can have only integer values. This can only change by $\Delta v = \pm 1$ in a harmonic oscillator model. The so-called zero-point energy occurs when $v = 0$ where $E = \frac{1}{2} h\nu$ and this vibrational energy cannot be removed from the molecule.

Figure 2.6 shows the curved potential wells for a harmonic oscillator with the probability functions for the internuclear distance X, within each energy level. These must be expressed as a probability of finding a particle at a given position since by quantum mechanics, we cannot be certain of the position of the mass during the vibration (a consequence of Heisenberg's uncertainty principle).

The anharmonic oscillator as shown in Figure 2.6 provides a more realistic model where the deviation from harmonic oscillation becomes greater as the vibrational quantum number increases. The separation between adjacent levels becomes smaller at higher vibrational levels until finally the dissociation limit is reached. In the case of the harmonic oscillator only transitions to adjacent levels or so-called fundamental transitions are allowed (i.e., $\Delta v = \pm 1$) while for the anharmonic oscillator, overtones ($\Delta v = \pm 2$) and combination bands can also result. Transitions to higher vibrational states are far less probable than the fundamentals and are of much weaker intensity. The energy term corrected for anharmonicity is

$$E_v = h\nu_e \left(v + \frac{1}{2}\right) - h\chi_e \nu_e \left(v + \frac{1}{2}\right)^2 \tag{2.11}$$

where $\chi_e \nu_e$ defines the magnitude of the anharmonicity.

2.5. The Raman scattering process

Light scattering phenomena may be classically described in terms of electromagnetic (EM) radiation produced by oscillating dipoles induced in the molecule by the EM fields of the incident radiation. The light scattered photons include mostly the dominant Rayleigh and the very small amount of Raman scattered light. The induced dipole moment occurs as a result of the molecular polarizability a, where the polarizability is the deformability of the electron cloud about the molecule by an external electric field.

In a typical Raman experiment, a laser is used to irradiate the sample with monochromatic radiation. Laser sources are available for excitation in the UV, visible, and near-IR spectral region (785 to 1064 nm). Thus, if visible excitation is used, the Raman scattered light will also be in the visible region. The Rayleigh and Raman processes are depicted in Figure 2.8.

No energy is lost for the elastically scattered Rayleigh light while the Raman scattered photons lose some energy relative to the exciting energy to the specific vibrational coordinates of the sample. In order for Raman bands to be observed, the molecular vibration must cause a change in the polarizability. Both Rayleigh and Raman are two photon processes involving scattering of incident light ($hc\bar{\nu}_L$), from a "virtual state." The incident photon is momentarily absorbed by a transition from the ground state into a virtual state and a new photon is created and scattered by a transition from this virtual state. Rayleigh scattering is the most probable event and the scattered intensity is about 10^{-3} less than that of the original incident radiation. This scattered photon results from a transition from the virtual state back to the ground state and is an elastic scattering of a photon resulting in no change in energy (i.e., occurs at the laser frequency).

Raman scattering is far less probable than Rayleigh scattering with an observed intensity that is about 10^{-6} that of the incident light for strong Raman scattering. This scattered photon results from a transition from the virtual state to the first excited state of the molecular vibration. This is described as an inelastic collision between photon and molecule, since the molecule acquires different vibrational energy ($\bar{\nu}_m$)and the scattered photon now has different energy and frequency.

As shown in Figure 2.8 two types of Raman scattering exist: Stokes and anti-Stokes. Molecules initially in the ground vibrational state give rise to Stokes Raman scattering, $hc(\bar{\nu}_L - \bar{\nu}_m)$ while molecules initially in vibrational excited state give rise to anti-Stokes Raman scattering, $hc(\bar{\nu}_L + \bar{\nu}_m)$. The intensity ratio of the Stokes relative to the anti-Stokes Raman bands is governed by the absolute temperature of the sample, and the energy difference between the ground and excited vibrational states. At thermal equilibrium Boltzmann's law describes the ratio of Stokes relative to anti-Stokes Raman lines. The Stokes Raman lines are much more intense than anti-Stokes since at ambient temperature most molecules are found in the ground state.

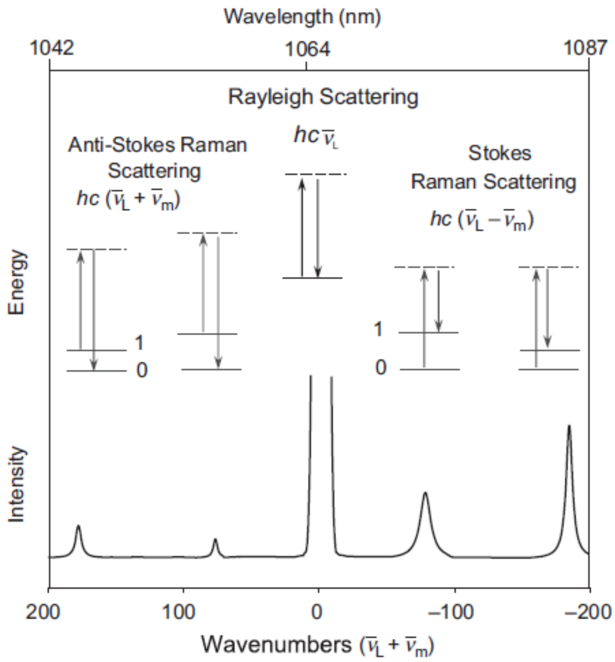


Figure 2.8 - Schematic illustration of Rayleigh scattering as well as Stokes and anti-Stokes Raman scattering. The laser excitation frequency (vL) is represented by the upward arrows and is much higher in energy than the molecular vibrations. The frequency of the

The intensity of the Raman scattered radiation I_R is given by:

$$I_R \propto \nu^4 I_o N \left(\frac{\partial \alpha}{\partial Q}\right)^2 \qquad (2.12)$$

where I_o is the incident laser intensity, N is the number of scattering molecules in a given state, ν is the frequency of the exciting laser, α is the polarizability of the molecules, and Q is the vibrational amplitude.

There is another classification of Raman scattering namely, *resonance scattering* and *non-resonance scattering*. Non-resonance Raman scattering occurs when the radiation interacts with a molecule resulting in polarization of the molecule's electrons. The increase in energy from the radiation excites the electrons to an unstable virtual state; therefore, the interaction is almost immediately discontinued and the radiation is emitted (scattered) at a slightly different energy than the incident radiation.[14]. Resonance Raman scattering occurs in a similar fashion. However, the incident radiation is at a frequency near the frequency of an electronic transition of the molecule of interest. This provides enough energy to excite the electrons to a higher electronic state. Being so close to an excited state makes the Raman process much more likely to occur. That's why the signal is so much stronger and can be used to detect much lower concentrations of a substance than conventional Raman can.

2.6. Classical description of the Raman effect

When charged particles of a molecule get perturbed by the electromagnetic field a dipole moment is induced given by

$$\mu = \alpha E \tag{2.13}$$

where α is the polarizability, E is the incident electric field, and m is the induced dipole moment. Both E and α can vary with time. The electric field of the radiation is oscillating as a function of time at a frequency ν_0, which can induce an oscillation of the dipole moment μ of the molecule at this same frequency, as shown in Figure 2.9a.

The polarizability α of the molecule has a certain magnitude whose value can vary slightly with time at the much slower molecular vibrational frequency ν_m, as shown in Figure 2.9b.

The result is seen in Figure 2.9c, which depicts an amplitude modulation of the dipole moment oscillation of the molecule. This type of modulated wave can be resolved mathematically into three steady amplitude components with frequencies ν_0, $\nu_0 + \nu_m$, and $\nu_0 - \nu_m$ as shown in Figure 2.9d. These dipole moment oscillations of the molecule can emit scattered radiation with these same frequencies called Rayleigh, Raman anti-Stokes, and Raman Stokes frequencies. If a molecular vibration did not cause a variation in the polarizability, then there would be no amplitude modulation of the dipole moment oscillation and there would be no Raman Stokes or anti-Stokes emission.

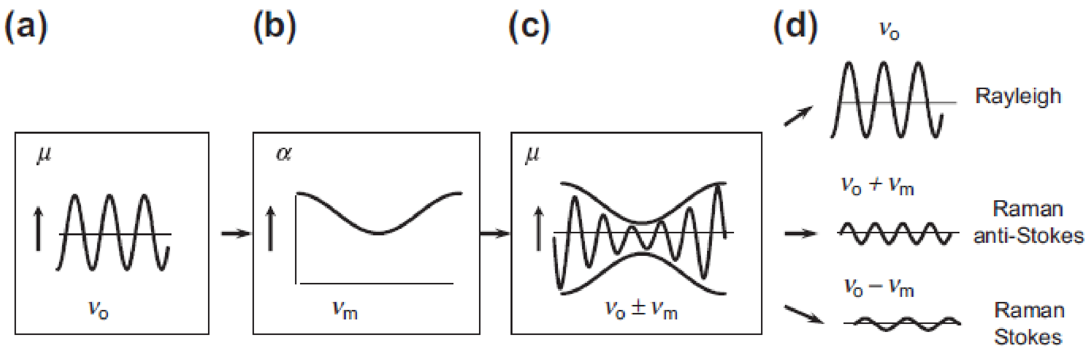


Figure 2.9 - Schematic representing Rayleigh and Raman scattering.

In (a) the incident radiation makes the induced dipole moment of the molecule oscillate at the photon frequency.

In (b) the molecular vibration can change the polarizability, a, which changes the amplitude of the dipole moment oscillation.

The result as shown in (c) is an amplitude modulated dipole moment oscillation.

The image (d) shows the components with steady amplitudes which can emit electromagnetic radiation.

Chapter 3

3 Development of portable Raman spectrometer using off-the-shelf optical assembly and CMOS linear image sensor

A portable Raman spectrometer has been designed and developed with minimum number of components keeping the procurement cost of the components and associated optics to a minimum. The entire design process can be broadly divided into three parts – 1) Laser power source, 2) Optical head, and 3) Detector comprising of high-sensitivity CMOS linear image sensor. In the following sections these three categories have been explained in details with 3D illustrations wherever needed for the reader to understand easily.

3.1. Components of laser power supply

The laser power supply gives a constant supply of current up to 180 mA to the 785 nm laser. It also consists of a temperature controller to control the operating temperature at 30 °C. The details of each component used in building the power source is described below.

3.1.1 Power Supply

The heart of the power supply is a MeanWell NET-50B triple channel SMPS power supply rated at 50W with output channels – 5V, 4A (CH1), 12V, 2A (CH2), -12V, 500mA (CH3) and takes 84-264 VAC or 120-370 VDC to drive itself. The output voltage tolerance is well within acceptable range with ±2% variation for CH1 and around ±5% variation for CH2 and CH3. Its line and load regulations are also within acceptable limits. The power supply has over-voltage protection as well. To drive the constant current source the output is reduced to ±9V with the help of voltage regulators LM7809 and LM7909 from STMicroelectronics.

Figure 3.1 - MeanWell NET-50B SMPS (left) and ±9V voltage regulator IC (right)

3.1.2 Constant current source

Figure 3.2 - LD1255R - 250 mA Precision Constant Current Laser Driver

The constant current source is a LD1255R laser diode constant current driver from Thorlabs Inc. which has low current noise, low temperature drift and comes with ESD protection. It can source a current ranging from 0.2-250 mA. Its operating voltage is from ±8-12V but for safety conditions we have used ±9V to drive the LD1255R. It has an external current control pin which is used to drive the laser based on the input current given by the user. The transfer function at the control pin is $V_{ctrl} = 50 \times I_{laser}$, where I_{laser} is the laser current in mA. The LD1255R operates the laser anode at ground potential for added protection against ESD. This requires that the LD1255R use a negative power supply to "pull" current from the ground-referenced laser anode. It has a warm-up time of 30 minutes.

3.1.3 Laser

The laser is 785±0.5 nm single mode spectrum stabilized laser from Innovative Photonic Solution with part #I0785S50100B. It comes with a circularized and collimated output beam, integral lase line filter, internal thermistor and ESD protection. Lasing wavelength can be accurately specified and repeatedly manufactured to within 0.1 nm. Its typical output power is 100 mW, optical beam diameter is 0.5-1.0 mm and maximum allowed current is 250 mA with a compliance voltage of 2.2 V within the typical temperature range of 30-35 °C. A Peltier cooler is fitted with laser housing so as to control its temperature through a TEC controller by measuring the 10kΩ NTC thermistor built inside the laser housing.

3.1.4 TEC controller

Figure 3.3 - TCM1000T - TEC Controller Module

The temperature controller is a TCM1000T 3W TEC controller from Thorlabs Inc. It controls current through the Peltier cooler attached to laser housing in order to maintain a constant temperature on the device. Operating from a +5 VDC power supply the module can provide up to 3 Watts of power and is current limited to 1 Amp. The module is designed to maintain temperature based on feedback provided from a 10 kΩ NTC type thermistor sensor. Temperature is set and monitored using a scaled voltage based on the thermistor resistance for a given temperature, and is available at test points

(T_{SET} & T_{ACT}) referenced to system ground. System response can be adjusted for various thermal loads using a Proportional Gain adjustment pot and an Integral Gain adjustment pot. Maximum adjustment range is 5 kΩ to 25 kΩ.

3.1.5 Micro-controller

The micro-controller is an ATmega328/P which is an 8-bit micro-controller from Atmel® having 32 Kbytes of flash memory and 2 Kbytes of SRAM. The micro-controller is driven by a crystal of 16 MHz. The ATmega328/P has a 10-bit ADC inside which is used to sense the temperature from the NTC thermistor attached to the laser housing. The transfer function for conversion from resistance to temperature is given in the datasheet of the laser source which is as follows-

$$Temp\ in\ °C = \frac{1}{A + B \times \ln(kOhm \times 1000) + C \times (\ln(kOhm \times 1000))^3} - 273.15 \qquad (3.1)$$

where, $A = 0.00113$, $B = 0.000234$, $C = 8.78 \times 10^{-8}$ and $kOhm$ in the above equation is calculated from the transfer function mentioned in the datasheet of TCM1000T, which is 1 V = 10 kΩ.

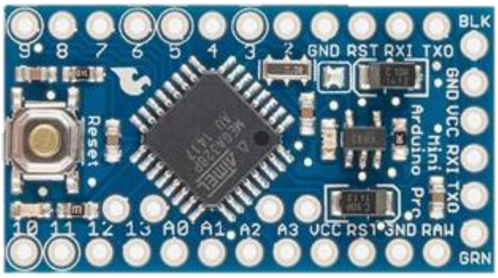


Figure 3.4 – Arduino Pro mini

The micro-controller accepts input from four SPST buttons which are used to adjust the current through laser and monitor the laser operating temperature. The user defined value of laser current is then set through an 8-bit DAC.

Figure 3.5 – Atmega 328/P micro-controller

3.1.6 DAC

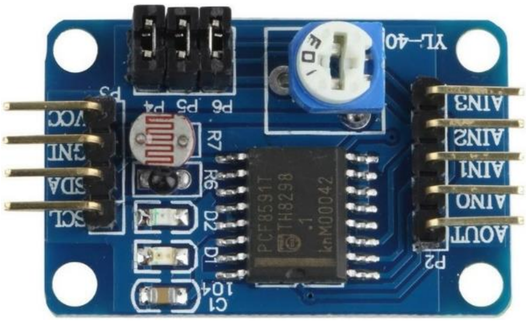


Figure 3.6 - PCF8591 8-bit DAC

The DAC used is a PCF8591 which is a 8-bit A/D & D/A converter from NXP Semiconductors. It gives an analog voltage tto the control pin of the constant current module based on the user input value of the laser current through four SPST buttons.

3.1.7 Display LCD

The display unit is a JHD162A series with 16x2 display. The LCD module has 16 pins and can be operated in 4-bit mode or 8-bit mode. Here we are using the LCD module in 4-bit mode. It is driven by the micro-controller. The user can interact with the display using the four push-buttons. The display shows the current passing through the laser diode and its temperature. A 10 kΩ trim-pot is used to set the contrast of the display.

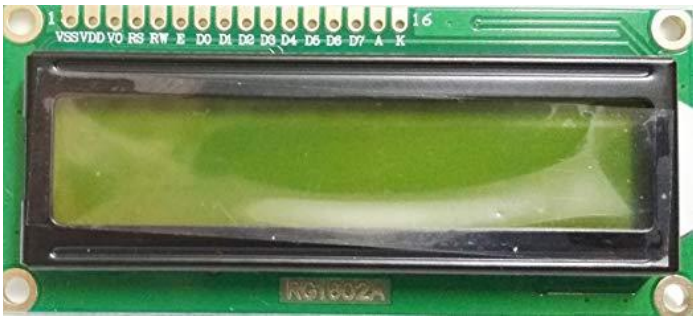


Figure 3.7 - JHD162A 16×2 LCD module

3.2. Design of laser power source

The laser used in this project is a 785 ± 0.5 nm, 100 mW laser with 10 kΩ NTC thermistor attached to the housing itself. The temperature of the laser is electronically controlled by checking the resistance of the thermistor with the help of Peltier cooler which is also attached to the laser housing. The laser temperature is roughly kept at 29 °C. The laser output power is limited by the current passing through it. This constant current is supplied by the LD1255R constant current module which is operated in external current control mode. The external current control pin can accept a voltage in the range of 0-5 V giving out maximum current at 5 V and minimum current at 0 V. For safe operation of the laser, the maximum value is chosen to be 180 mA and minimum current to be 5 mA. At first, polynomial fitting is employed and the result up to 3rd degree is tabulated below.

Table 3-1 - Polynomial fitting of laser current data

Degree	RMSE	R-square (%)
1	12.17	95.24
2	6.006	98.88
3	2.293	99.84

A cubic polynomial fit may seem to be a better a choice with an R-square value of 99.84 % with RMSE of about 2.3 but a careful observation of the data set in appendix 7.1 shows that the current tends to saturate beyond 165 mA as shown in Figure 3.8 and hence deviation increases. The laser current vs. DAC control byte data was taken a number of times and each time it was noticed that at the same set of control bytes, which was taken in earlier measurements, the laser current is same. Therefore, cubic spline data interpolation method, which is a piecewise polynomial fit method, was employed to get a function that can satisfactorily describe the variation of the output voltage of DAC with its control byte. The plot is shown in Figure 3.9. The data set for the interpolation and the associated function is given in appendix 7.1 and 7.2 respectively.

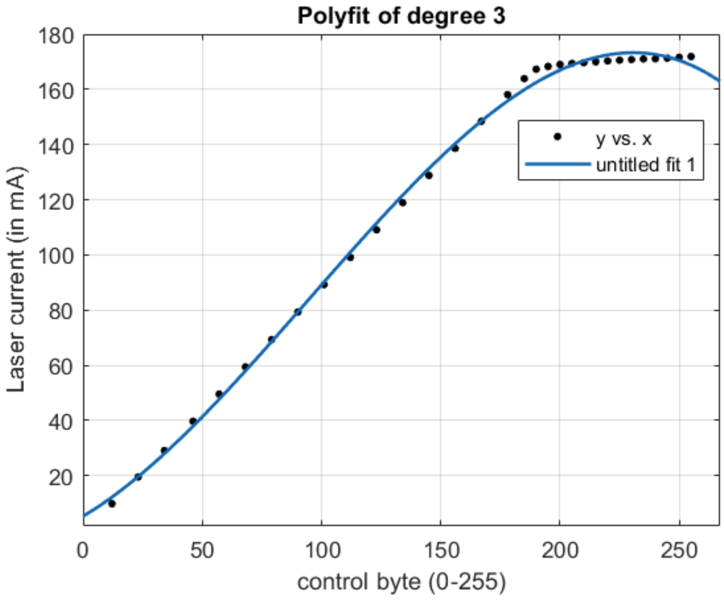


Figure 3.8 - 3rd degree polynomial fit of laser current data

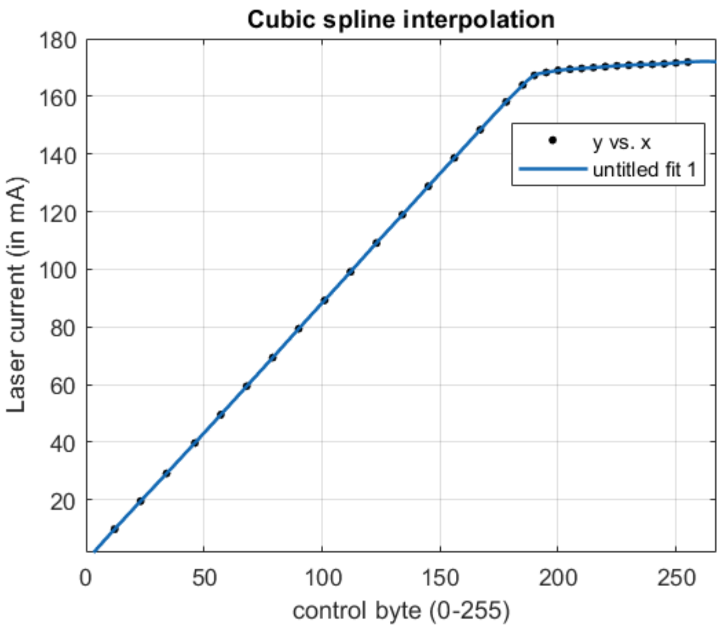


Figure 3.9 – Cubic spline interpolation of laser current data

Beside assembling the electronic components an interface is designed for the user to interact and change the laser current. For this purpose, a 16x2 LCD display was used along with four pushbuttons. The required code is provided in the appendix 7.4 which is written in Arduino IDE. A switch is also provided to disable the laser, if needed, without turning off the entire power supply.

Before procuring the components, a 3D model was developed to determine its capacity, size and portability so that all the components can be housed properly inside the enclosure. For convenience, only the main PCBs and other important components were placed inside the housing and also the wiring is not shown in the 3D model. A schematic of the entire design is provided in the appendix 7.3. Figure 3.10 show the proposed prototype of the laser power supply while Figure 3.11 shows the developed prototype.

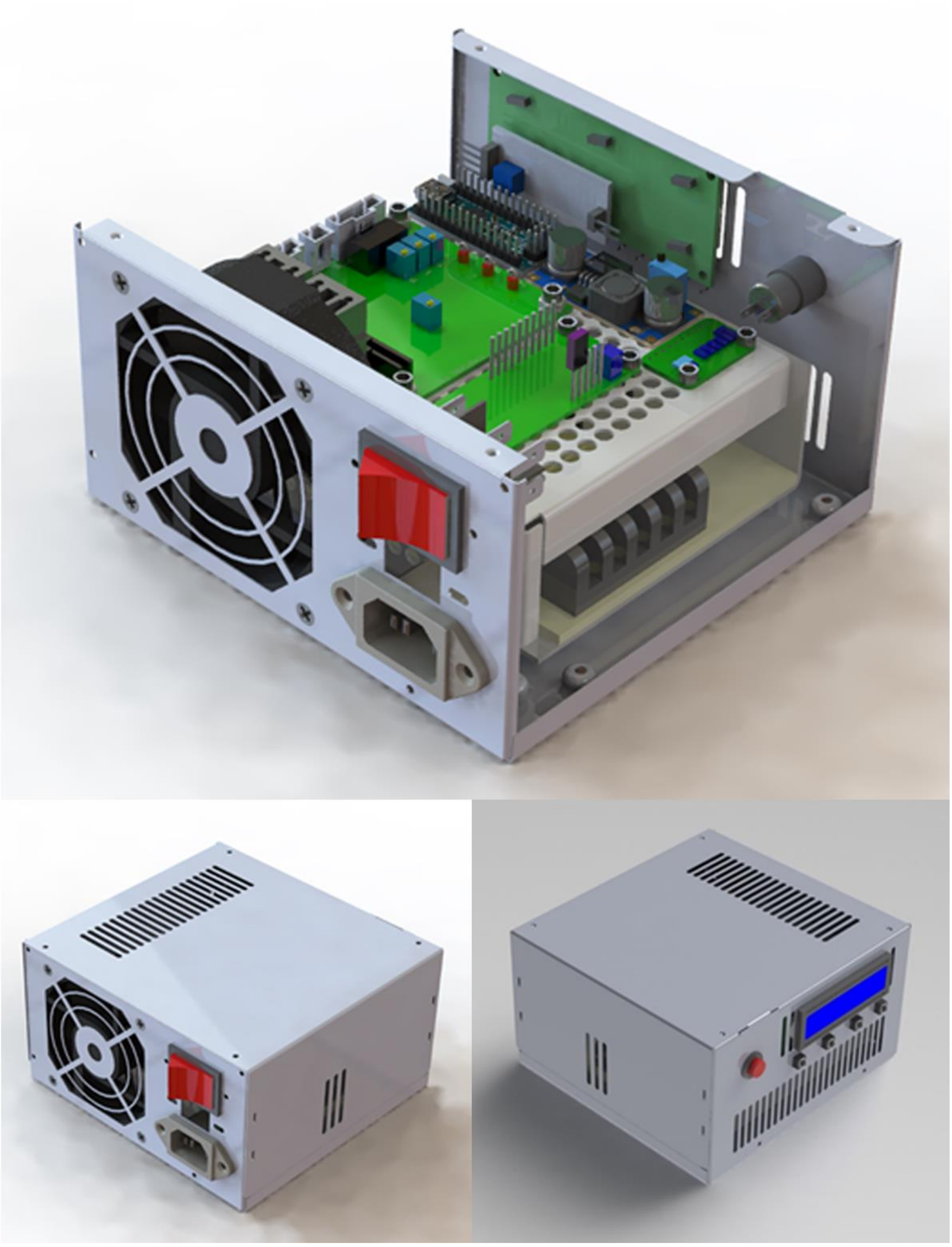


Figure 3.10 - 3D model of the proposed laser power supply

Four push buttons provided in front allows the user to change the laser current either in fine/coarse mode. A piezo-buzzer is added to give audible feedback upon successful button press by giving a short ‘beep’ sound. The large red coloured pushbutton in front allows the user to turn off the laser without turning off the entire power supply.

Figure 3.11 - A prototype of the laser power supply

3.3. Working principle of the laser power source

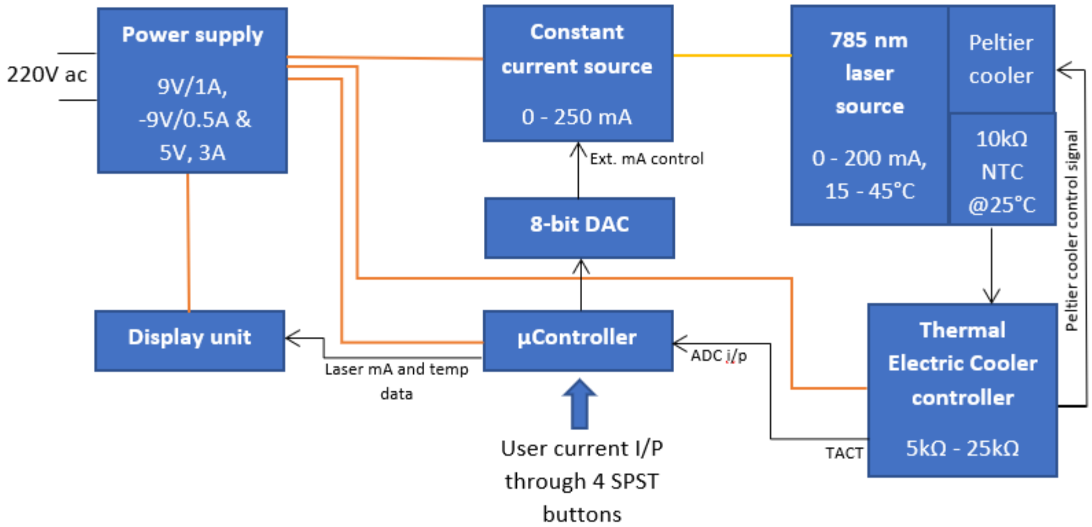


Figure 3.12 - Block diagram of the laser power source

The ±9V and 5V power rails supply power to all the modules and components which is shown by the red lines. The laser housing with built-in NTC thermistor is kept in contact with a Peltier cooler in order to control its temperature. This control signal is provided by the TEC controller in response to the temperature signal coming from the thermistor, thus forming a feedback loop. The user is provided with four SPST tactile buttons – Adjust (ADJ), SET, Increment (+) and Decrement (-), to set the laser current and monitor the laser temperature through a 16x2 LCD display unit. The SET button also acts as a fine/coarse adjustment knob which becomes useful when changing the current from a low to high value. Otherwise, for each step increase of 1 mA current the user has to press each time. Holding the SET button while at the 'Adjust Laser current' menu for 2 to 3 seconds turns on the 'coarse' mode and again holding the SET button for 2 to 3 seconds followed by a single tap returns the 'fine' mode. While at the main menu the laser temperature can also be monitored using the '+' or '-' button. After the user sets a current, the corresponding 8-bit control voltage value is transferred to the DAC by the micro-controller to produce a proportional voltage based on the spline interpolation coefficients given in appendix 7.2 which goes to the external control pin of the constant current module in order to produce a stable laser current as set by the user.

3.4. Components of optical head

The optical head is an assembly of different mechanical and optical parts. It is basically a long cylindrical tube consisting of necessary optical parts for irradiating the sample and collecting the resultant scattered radiation and transferring it to the detector using an optical fibre thereby measuring the Raman spectra. Numerous components have been used to develop the optical head. However, for sake of simplicity, only the important components have been explained here.

3.4.1 Laser

The laser is procured from Innovative Photonic Solutions having part #I0785S50100B which is shown in Figure 3.13 . Much of the details about the laser have been given in section 3.1.3. It is in a TO-56 package with laser line filter present inside the housing. It has wavelength tolerance of ±0.5 nm having a spectral linewidth of about 100 kHz. The SMSR with integrated laser line filter is very high at 70 dB. The beam exit angle is less than 3 degrees and beam divergence is about 2 mrad. The thermistor and Peltier cooler are also attached to the laser housing.

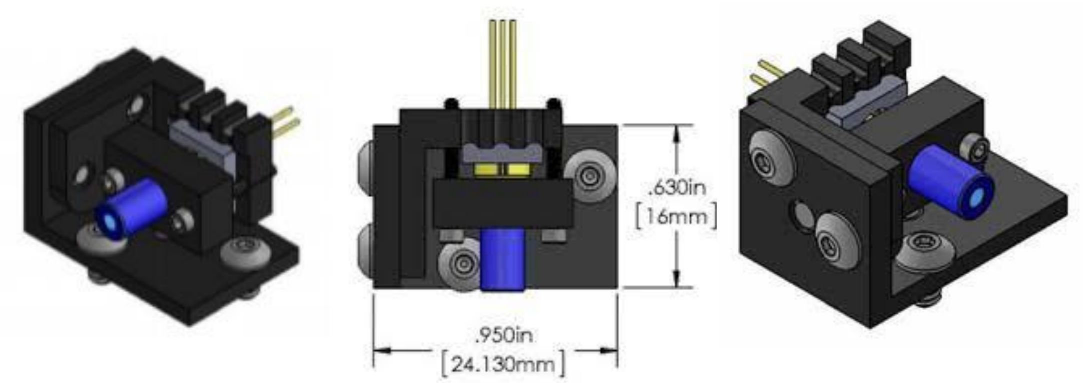


Figure 3.13 – 785 nm laser with thermistor and Peltier cooler

3.4.2 Mirror

The mirror is a MRA05-M01 right-angle prism mirror from Thorlabs Inc. The reflective surface is formed by the hypotenuse and is gold coated. The other two sides are having a length of 5 mm. The gold coating provides high reflectance with both S- and P-polarized light from 800 nm to 20 µm. The gold coating features a protective overcoat of SiO2. It has an average reflectance of greater than 96% in the range of 800 nm to 20 µm.

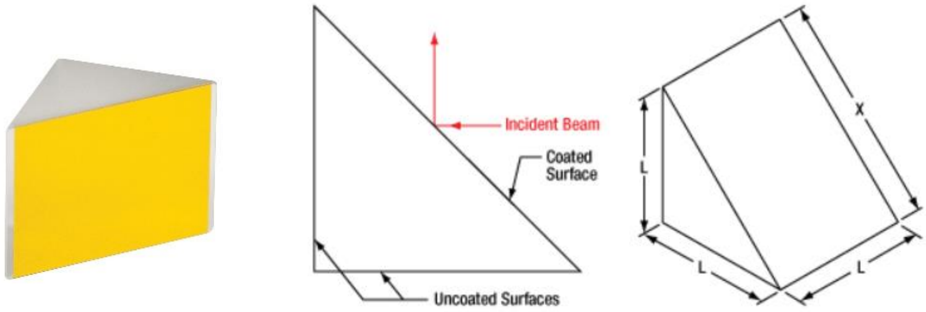


Figure 3.14 – Gold coated right angle prism mirror with L=5 mm

3.4.3 Focusing lens

Two focusing lens have been used, one with part #ACL25416U-B and another with part #LA1134-B from Thorlabs Inc. The former is an aspheric condenser lens of 1" diameter with 16 mm focal length and having a high NA of 0.79. The anti-reflective coating on this lens permits transmittance of about 70% for wavelength in the range 650 nm to 1050 nm. This type of lens is generally used in light collection. The latter is a plano-convex lens of 1" diameter with focal length of 60 mm. This lens also has anti-reflective coating which transmits 70% of light in the range of 650 nm to 1050 nm. It is used here to focus the collimated beam, coming from first lens, on the SMA connector.

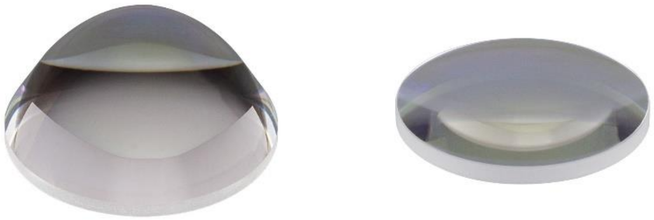


Figure 3.15 - Aspheric condenser lens (left) and Plano-convex lens (right)

3.4.4 Notch filter

Notch filters, also commonly referred to as band-stop or band-rejection filters, are designed to transmit most wavelengths with little intensity loss while attenuating light within a specific wavelength range (the stop band) to a very low level. They are essentially the inverse of bandpass filters, which offer high in-band transmission and high out-of-band rejection so as to only transmit light within a small wavelength range. Notch filters are useful in applications where one needs to block light from a laser. For instance, to obtain good SNR ratios in Raman spectroscopy experiments, it is critical that light from the pump laser be blocked. This is achieved by placing a notch filter in the detection channel of the setup. The one that is used in this project has a part #NF785-33 from Thorlabs Inc. which has a center wavelength of 785±2 nm. The passbands with more than 90% average transmittance are 590-760 nm and 810-1040 nm.

Figure 3.16 - 785 nm Notch filter

3.4.5 SMA connector and fiber optic cable

SMA is a fiber optic connector developed and manufactured by Amphenol Fiber Optic Products; it stands for SubMiniature version A. SMA connectors use a threaded plug and socket. It was the first connector for optical fibers to be standardized. In addition to their compact size, the SMA connector has exceptional mechanical durability. The SMA connector holds a single fiber. The one used here has a part #SM1SMA from Thorlabs Inc. The fiber optic cable is 1.5 m long cable with core diameter of 600 μm having NA of 0.22 having SMA905D on both ends.

Figure 3.17 - SMA connector (left) and Fibre optic cable (right)

3.5. Assembly of the optical head

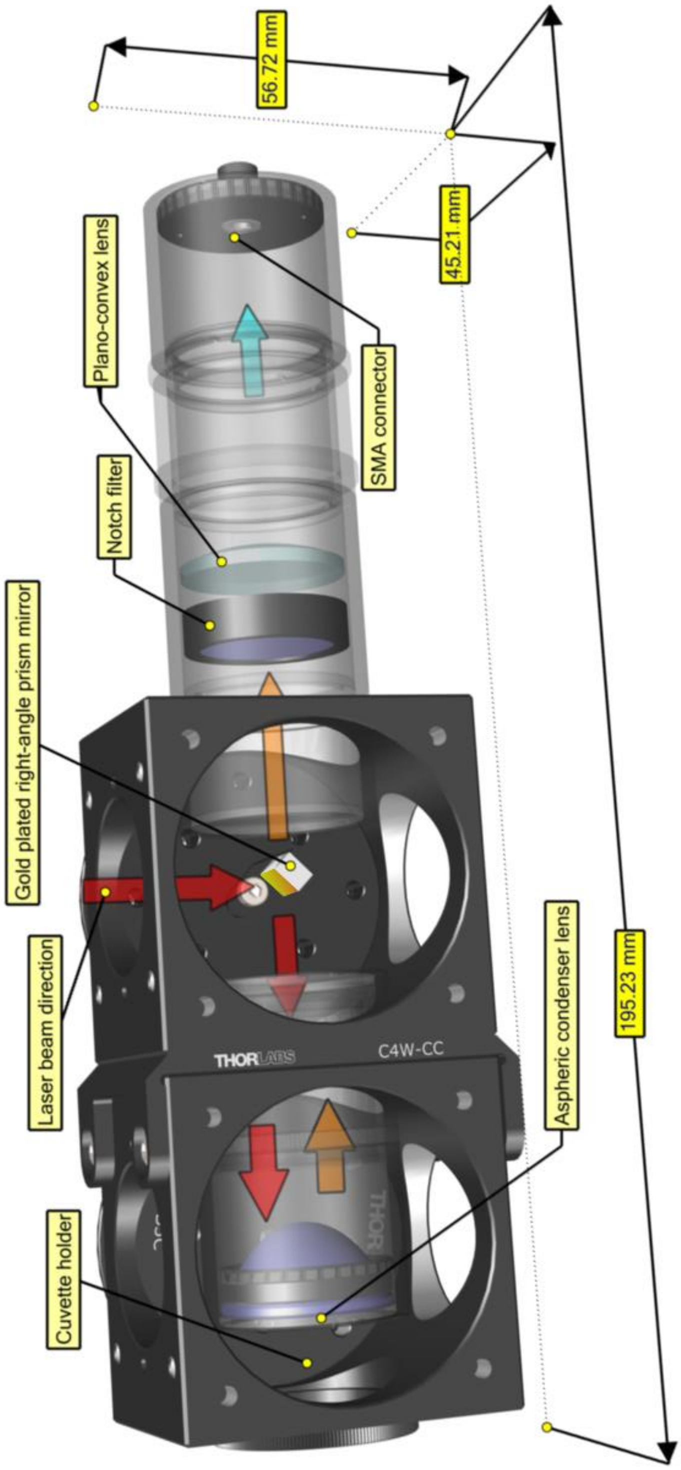


Figure 3.18 - The assembled optical head for Raman spectrometer. (Laser not shown)

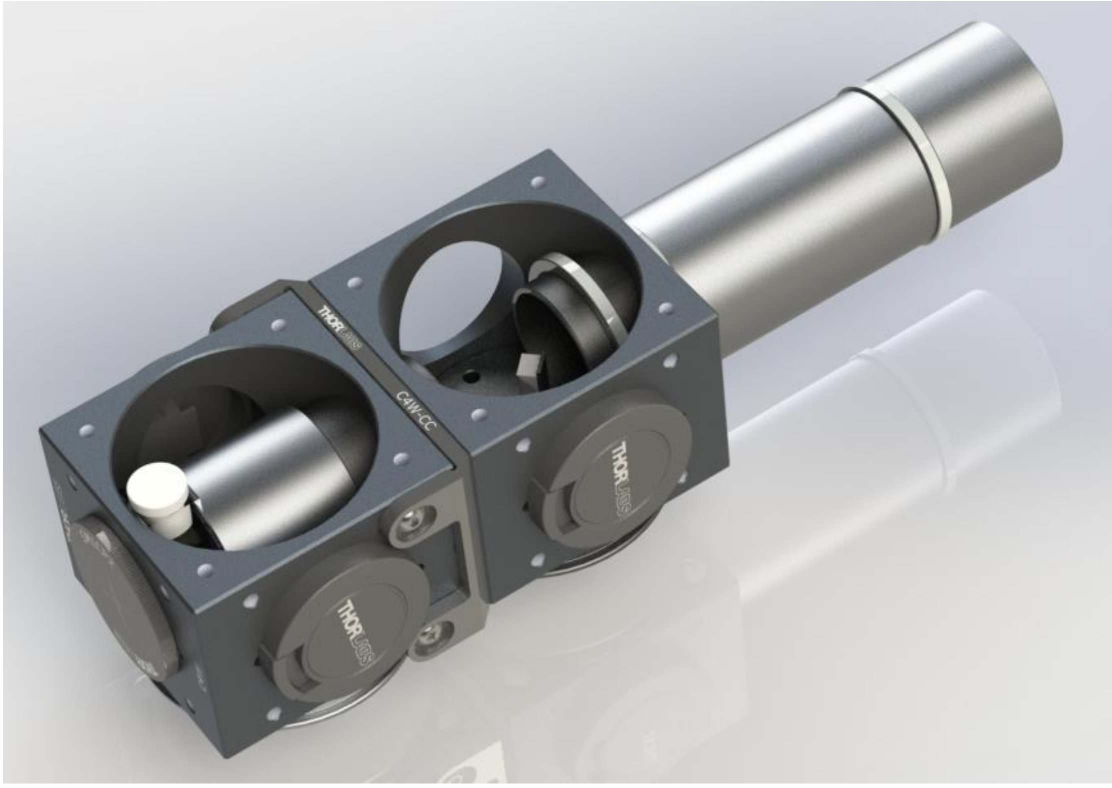


Figure 3.19 - 3D model of the proposed optical head. (In this picture the cuvette holding a liquid sample is shown in its place and the laser housing is not shown)

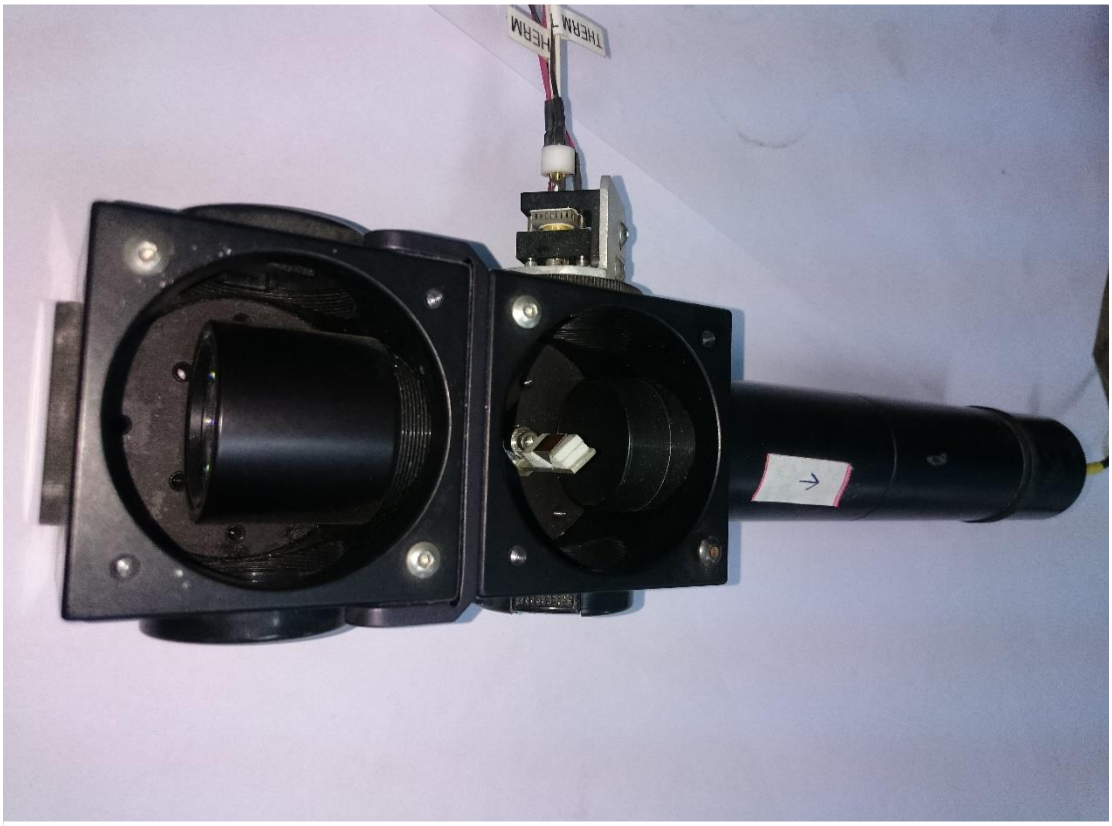


Figure 3.20 - A prototype of the optical head. (In this picture the laser housing is attached)

3.6. Working principle of the optical head

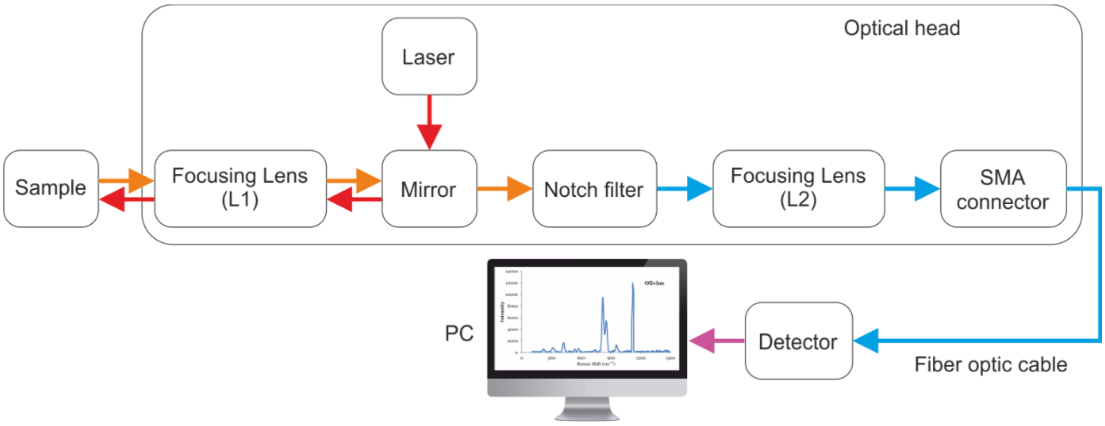


Figure 3.21 - Block diagram of the optical assembly with detector

A basic block diagram of the optical assembly is shown in Figure 3.21. A 785 nm laser along with Peltier cooler and associated NTC thermistor for temperature feedback is fitted at right angle to the optical head. The radiation from the laser falls on a mirror which is placed at 45°. The right-angle prism mirror reflects the radiation towards the sample. The focusing lens L1 focuses the beam on to the sample. The radiation scattered by the sample is collected by the focusing lens itself. Since the size of the mirror is very small compared to the optical tube, most of the scattered radiation bypasses the mirror and reaches the notch filter. The notch filter greatly reduces the intensity at laser wavelength which is 785 nm and transmits the remaining light towards focusing lens L2. This lens then focuses at the SMA connector and falls almost normally on the optical fiber. The detector housing consists of the CMOS image sensor and the grating and convex lens assembly. The grating breaks the individual wavelengths and forwards to the image sensor. With the help of necessary electronic interface, the information from the image sensor is read and displayed via a computer.

3.7. Acquisition of Raman spectra

The scattered radiation from the optical head goes via the optic fibre to the detector containing the necessary optical components and the high-sensitivity CMOS image sensor. The detector consists of a slit of dimension 10 x 400 µm, a collimating lens, a transmission grating, a focusing lens and a CMOS image sensor of 512 pixels as shown in Figure 3.22. The detector has a spectral resolution (FWHM) of 0.4 nm. The interface electronics has a 16-bit ADC and integrate from as low as 11 μs to 100 ms. It supports USB to facilitate connection with a PC and draws only a typical of 220 mA current.

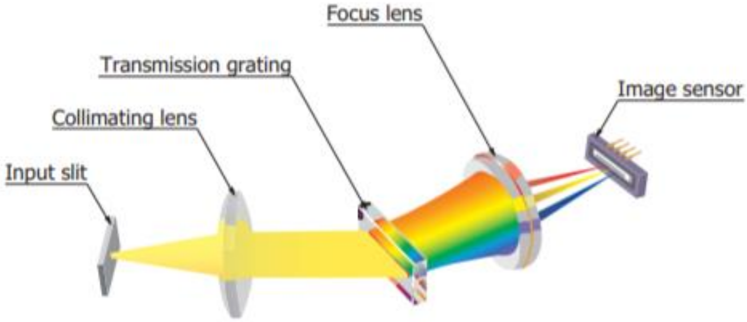


Figure 3.22 - Optical component layout of the detector

The data acquired at each pixel corresponds to each wavelength of the spectrum and is A/D converted in proportion to the light intensity. When saving this data with the evaluation software, the horizontal axis can be specified as pixels or wavelength. The following quantic equation can be used to calculate which pixel corresponds to the wavelength axis.

$$wavelength\ (nm) = a_0 + a_1 \cdot pix + a_2 pix^2 + a_3 pix^3 + a_4 pix^4 + a_5 pix^5 \tag{3.2}$$

, where pix is any pixel of the image sensor. The coefficients are as follows:

- $a_0 = 778.6461498$
- $a_1 = 0.3585042305$
- $a_2 = -0.0001745020601$
- $a_3 = 1.85457711 \times 10^{-7}$
- $a_4 = -3.284555756 \times 10^{-10}$
- $a_5 = 2.567910792 \times 10^{-13}$

With an integration time of 100 ms, 64 Raman spectra of pure benzene were averaged. Then it is subtracted from its dark spectrum and then plotted as it is. This can be seen in Figure 3.23.

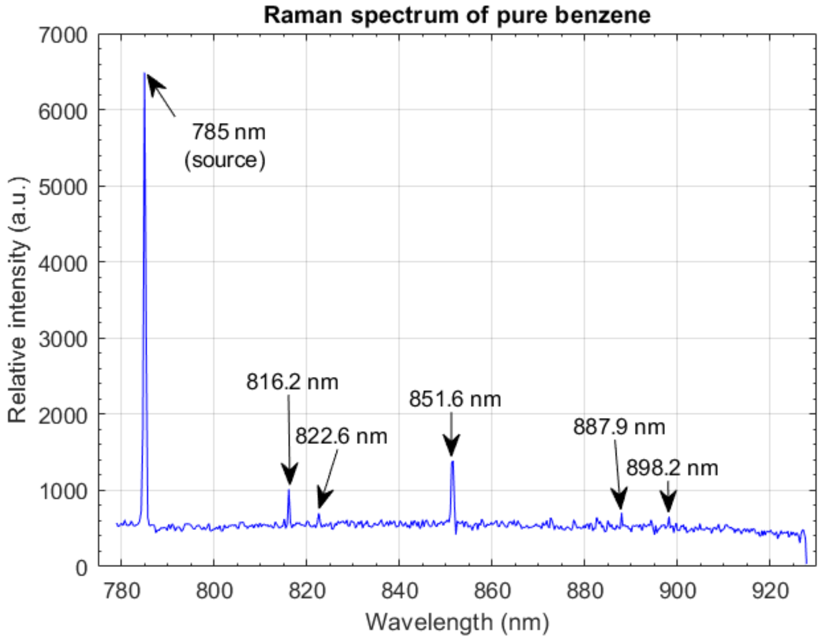


Figure 3.23 - Raman spectrum of pure benzene. (Here the source excitation is not suppressed)

Raman spectra generally consist in peaks and noise superimposed on a background. This background, or baseline can be either flat, linear with a positive or negative slope, curved or a combination of all three. It does not vary as quickly as the peaks do. This background is often due either to residual Rayleigh scattering at low Raman wavenumbers or to fluorescence of organic molecules intrinsic to the analysed sample or coming from contamination. Subtracting the estimation of the background from the raw spectrum gives a more interpretable signal, allowing to determine peak wavenumbers and to measure area and amplitude of peaks more accurately.

Background removal requires a cost function to be minimized e.g. Huber function or Truncated quadratic function. In Ref. [15], it was concluded that the asymmetric truncated quadratic function is the cost function which gives the best results to estimate the simulated background on spectra with only positive peaks. The asymmetric truncated quadratic function is given by

$$\forall x \in \mathbb{R}, \qquad \varphi(x) = \begin{cases} x^2 & \text{if } x < s, \\ s^2 & \text{otherwise} \end{cases} \tag{3.3}$$

,where s is the threshold value and x needs to be minimized.

After removing the background with an asymmetric truncated quadratic cost function with threshold of 2 and order of 8, we get the baseline corrected Raman spectra which is shown in Figure 3.24. In the spectra the source excitation peak corresponding to 785 nm was removed and presented in terms of Raman shift. Position of Raman peaks from various literatures have been listed in Table 3-2.

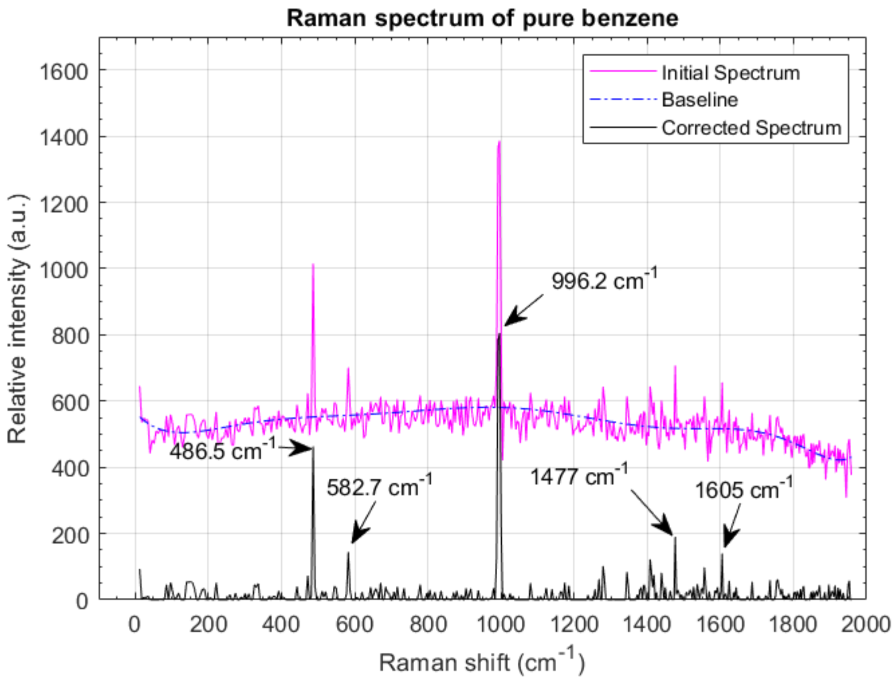


Figure 3.24 - Raman shift for pure benzene

Table 3-2 - Raman shift (cm^{-1}) of pure benzene from various literatures

Sl. No.	Lit. #1 [16]	Lit. #2 [17]	Lit. #3 [18]	Lit. #4 [19]	Lit. #5 [20]
1.	605.6	610	606	606	606
2.	848.9			850	849
3.	991.6	994	992	992	992
4.	1178	1180	1177	1178	1178
5.	1326				
6.	1584.6	1589-1609	1586	1585	1584
7.	1606.4				1603
8.					2460
9.					2542
10.					2597
11.					2617
12.					2784
13.					2928
14.			2950		2947
15.	3046.8	3048		3047	3046
16.	3061.9	3063	3063	3062	3060
17.					3164
18.					3183

3.8. Result and Conclusion

The values mentioned in Table 3-2 are taken as standard Raman shifts for further analysis. After comparing Figure 3.24 and Table 3-2, it was found that the strongest peak of pure benzene is close to the value obtained from various literatures. In this experiment, the strongest Raman peak was found at 996.2 cm^{-1} while literatures say, it should be 992 cm^{-1}. So, we have obtained this peak with 99.6 % accuracy. Another peak obtained at 1605 cm^{-1} is well within the range given by literatures. The second highest peak corresponding to 486.5 cm^{-1} does not belong to the Raman fingerprint. The peak at 1477 cm^{-1} is little off from literature value 1585 cm^{-1} showing only 93.2 % accuracy. All other shifts which are not in accordance with the standard values may have occurred either due to accidental entry of impurities while taking measurement or over-aging of the sample. It was also noted that a significant amount of noise is present in the Raman spectra as can be seen from Figure 3.24. This may have been caused by several factors. One of them may be accumulation of dust on the optical components or due to the thermal noise of the image sensor since it is of non-cooled type.

It can also be noted from Figure 3.23, that due to Rayleigh scattering the Raman scattering is greatly suppressed. The intensity of source excitation, though designed to be reduced by a factor of 10^6, is not getting suppressed properly. It may be due to a faulty notch filter which was supposed to suppress the 785 nm excitation wavelength. Also, the placement of the gold-plated mirror in the path of the scattered radiation coming from the sample towards the notch filter obstructs some of the scattered radiation thereby reducing the intensity of Raman peaks. It can be avoided by using a dichroic mirror in conjunction with the notch filter or a long-pass filter. The image sensor can also be sufficiently cooled by use of Peltier cells to avoid thermal noise though it would increase the current requirement of the power supply.

Chapter 4

4 Design and development of the detector with its interface electronics and a PC-based GUI for Raman spectrometer

4.1. Components of the detector

The detailed description of the components used in the design of the detector and its associated electronic parts are described in this section.

4.1.1 CCD detector

The charge coupled device, CCD, was originally designed not to be an optical detector, but a memory instead. It should work as an analog shift register, storing arbitrary charge in each of its cells and reading them sequentially. Later it was discovered that illumination causes internal photoelectric effect in the silicon substrate and fills up each memory cell with additional charge proportional to the incident light intensity. After that it found widespread application as a sensitive optical detector both in consumer electronics and in high-end scientific instruments such as astronomical telescopes.

CCD detectors may be divided into two groups: linear CCDs, where one row of pixels is shifted to the output, and matrix CCDs, where the bottom row is fully shifted out always when all columns are shifted down by one pixel. The one that is used here is TCD1304AP from Toshiba which is a linear non-cooled CCD image sensor with 3648 pixels each having a size of 8 µm x 200 µm.

Figure 4.1 - TCD1304AP from Toshiba

4.1.2 Micro-controller

The micro-controller used in this project is STM32H743ZI from ST Microelectronics. STM32 is a family of 32-bit microcontroller integrated circuits by STMicroelectronics. The STM32 chips are grouped into related series that are based around the same 32-bit ARM processor core, such as the Cortex-M7F, Cortex-M4F, Cortex-M3, Cortex-M0+, or Cortex-M0. Internally, each microcontroller consists of the processor core, static RAM, flash memory, debugging interface, and various peripherals. STM32H743ZI is based on the high-performance ARM® Cortex® - M7 32-bit operating at up to 480 MHz. It has a flash memory of 2 MB and a RAM of 1 MB along with extensive range of I/O peripherals. It has four DMA controllers to unload the CPU. It also has three 12-channels 16-bit ADCs with up to 3.6 MSps, two 32-bit timers with PWM feature, ten 16-bit general-purpose timers and supports USB OTG. The microcontroller comes with STM32 NUCLEO-144 development board.

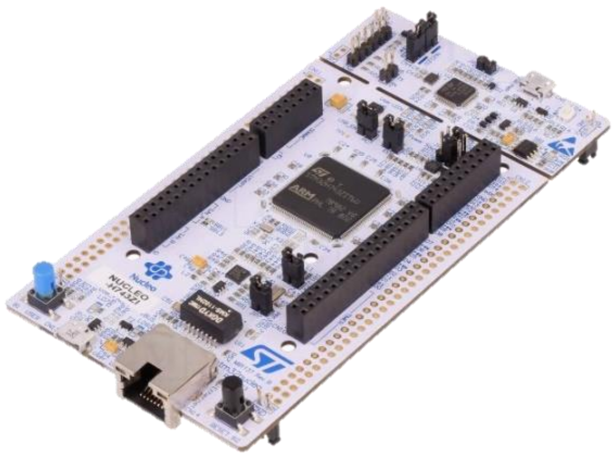


Figure 4.2 - STM32 Nucleo-H743ZI development board

4.1.3 FPGA

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory. Many FPGAs can be reprogrammed to implement different logic functions, allowing flexible reconfigurable computing as performed in computer software. The FPGA configuration is generally specified using a hardware description language (HDL) like Verilog or VHDL.

The one that is used here is of iCE40 LP8K from Lattice Semiconductor. It has 7680 logic cells (LUT + flip-flop) with 128Kb RAM. It has 22 PLLs and also features 178 maximum programmable I/O pins. It comes with TinyFPGA BX development board having its own bootloader where the user can program the FPGA through a simple micro-USB cable.

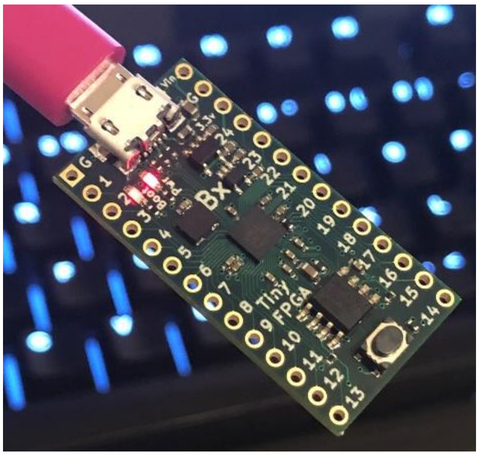


Figure 4.3 - TinyFPGA BX development board

4.2. Design of the detector

The image sensor and the interface electronics have been designed to obtain Raman spectra. Starting from the acquisition circuitry up to displaying the spectra in a custom designed GUI on PC has been thoroughly discussed in this section.

4.2.1 Driving the CCD image sensor

The TCD1304AP needs three driving pulses:

- fM - the master clock, running at 0.8-4 MHz
- SH - the shift gate
- ICG - the integration clear gate

For TCD1304AP the timing chart is given below.

TIMING CHART (Use electric shutter function)

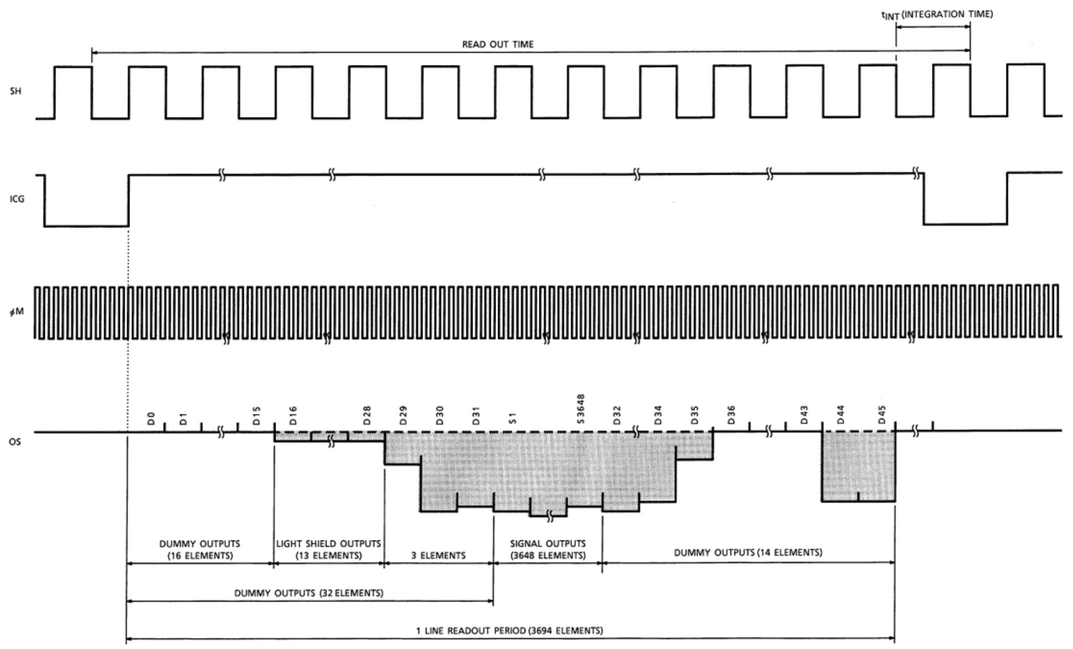


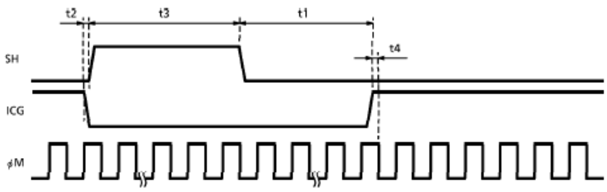
Figure 4.4 - The TCD1304AP's timing chart

From Figure 4.4 - The TCD1304AP's timing chart it is observed that

- The data-rate is one-fourth of fM.
- Pixels are only moved to the shift registers when ICG and SH coincide. If SH runs with a shorter period than ICG, the CCD runs in electronic shutter mode, and SH serves to control the integration time.
- The shortest integration time is 10 µs.

The datasheet gives the following timing requirements for the three pulses:

TIMING REQUIREMENTS



CHARACTERISTIC	SYMBOL	MIN.	TYP.	MAX.	UNIT
ICG Pulse DELAY	t1	1000	5000	—	ns
Pulse Timing of ICG and SH	t2	100	500	1000	ns
SH Pulse Width	t3	1000	—	—	ns
Pulse Timing of ICG and ϕ M	t4	0	20	*	ns

* : You keep ϕM "High" Level.
(Note) : If you use electronic shutter function. t_{INT} (MIN.) = 10µs

Figure 4.5 - Timing requirements for TCD1304AP

It translates to

1. SH must go high with a delay (t2) of between 100 and 1000 ns after ICG goes low.
2. SH must stay high for (t3) a minimum of 1000 ns.
3. ICG must go high with a delay (t1) of minimum 1000 ns after SH goes low.
4. ICG must go high when fM is high.

This is all handled by the FPGA.

4.2.2 Handling the output

CCDs are analog devices that collect photons and convert them into charge. The collected charge is then shifted to its internal pre-amplifier and the corresponding voltage signal occurs at OS pin. Sometimes the output voltage is stored in the sample-and-hold circuit. This output voltage then goes to the MCU, where an ADC digitizes it. The output signal from the TCD1304AP is provided in the figure below:

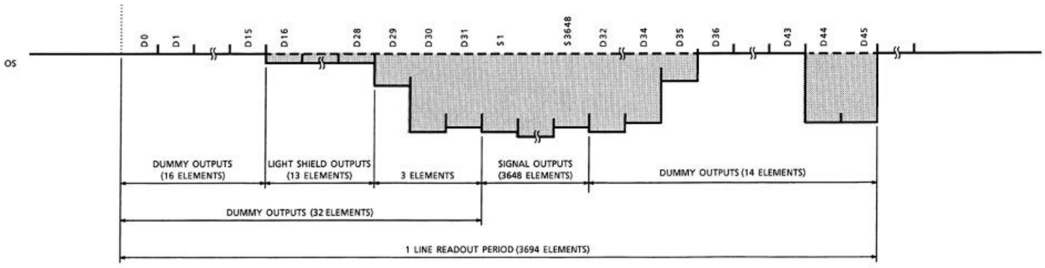


Figure 4.7 - Output signal of TCD1304AP

It is noted that the output signal is the reverse of the intensity of the light hitting the pixels. The higher the exposure the lower is the output voltage. There are 32 dummy pixels at the beginning and 14 at the end. The pixels giving useful information lies after the first 32 dummy pixels.

The CCD detector output shall depend on the illumination linearly. Under no illumination, the output is around 2.17 V, whereas under full saturation the output drops to 0.72 V. The influence of incident light on the CCD register itself and its supporting circuitry, along with thermal electrons, can interfere with measurement.

- **Blooming** occurs when excess charge concentrates in one cell and it leaks into neighbouring cells. It manifests as a flat, fully saturated regions with sharp edges, effectively obscuring the signal in the vicinity of the intensively illuminated spot. Overexposed areas must be avoided by either reducing the light intensity or integration time. For instance, the saturation exposure of the used linear CCD is about 0.004 lx·s.

- **Charge volatility:** The stored charge decays over time. This is expected to be caused by both thermal electrons and incident light. This presents a time constraint to the data acquisition and transmitting time.

- **Oversaturation:** When the CCD is illuminated by common daylight for a while (> 10 lx·s), the chip becomes so oversaturated that several full read-outs are needed to drain the integrated charge. Otherwise the output voltage may drop even under the saturation voltage. Since the occurrence of blooming is triggered by the saturation condition, it is useful to know the detector output voltage corresponding to its maximum charge capacity. This value, defined as the saturation voltage, represents the effective maximum output voltage of the CCD, and is calculated by multiplying the charge capacity by the charge-to-voltage conversion factor (the output sensitivity of the imaging device), as follows:

$$V_{sat} = N_{sat} \times \frac{dV}{dN} \qquad (4.1)$$

where is V_{sat} the saturation voltage, N_{sat} is the charge capacity, and $dV/_{dN}$ represents the charge-to-voltage conversion factor. This latter variable, which is equivalent to the CCD output sensitivity, is simply a ratio stating the change in output voltage for a given quantity of charge transferred onto the charge detection node of the device.

- **Thermal noise** is caused by random thermal electron-hole pairs. It is probably lower than electronic noise in the sampling circuit used.

Some of the aforementioned issues can be addressed by thermoelectrical cooling of the sensor, as a 20 °C temperature drop is expected to effectively suppress the thermal noise and conductivity. No cooling was used in this design because of rather high-power requirements of a Peltier cell.

4.2.3 Circuit diagram for TCD1304AP

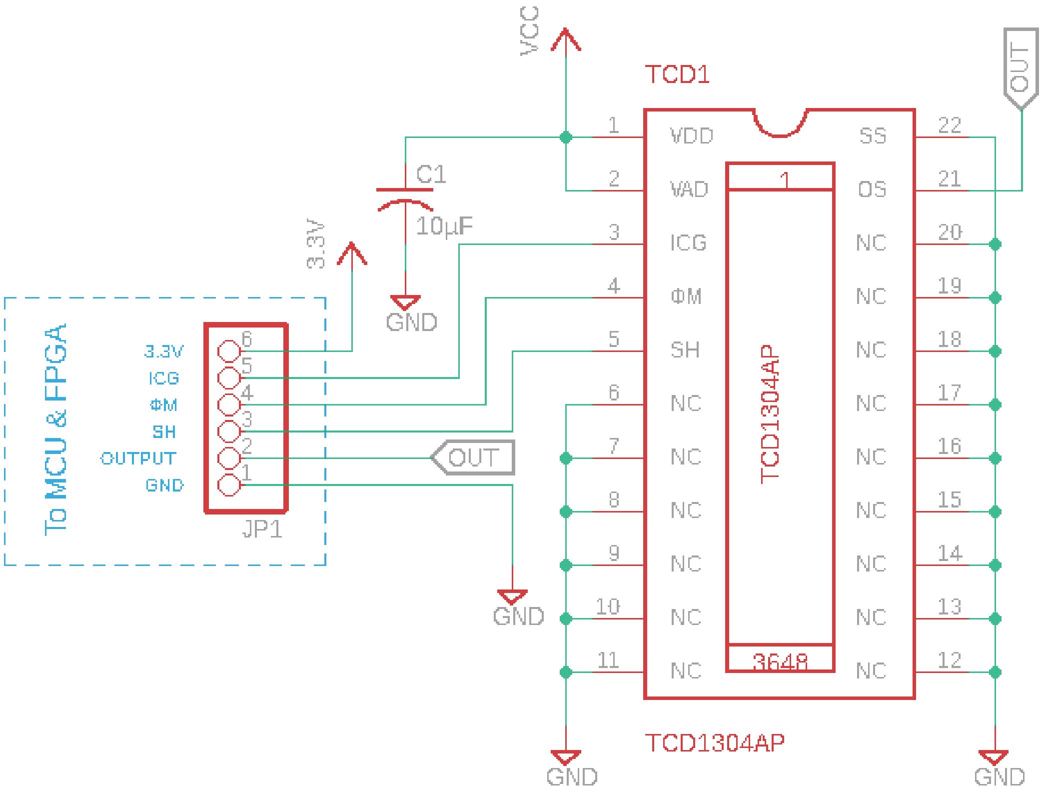


Figure 4.8 - Circuit diagram for TCD1304AP PCB

A 22 pin IC holder was soldered on a Veroboard shown in and a 6-pin relimate connector was provided for easy wiring. A 10 µF capacitor is used between the power rails to remove any noise from power supply.

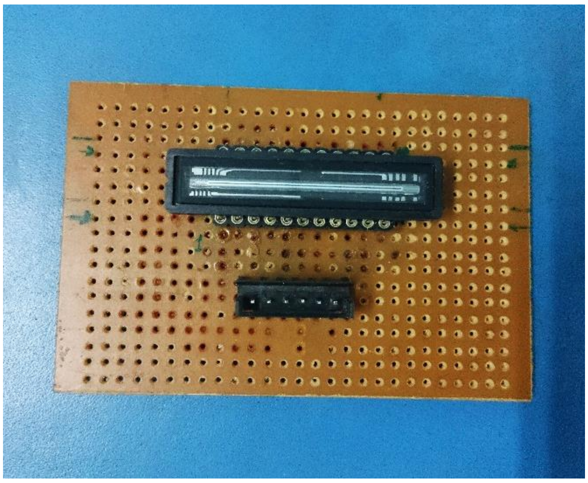


Figure 4.9 Circuit board for holding TCD1304AP

4.2.4 Programming the MCU

This section includes the programming and initialization of the main components inside the Micro-controller Unit (MCU).

4.2.4.1 The master clock signal

The TCD1304 requires a master clock (fM) with a frequency in the range of 0.8-4 MHz. The datasheet gives a typical value of 2 MHz. The CCD output data-rate is one-fourth of fM and the ADC must be able to keep up with this. Since, the MCU used has ADC with speed up to 3.6 MSps, it will be a piece of cake.

With fM = 2 MHz the data-rate is 0.5 MHz which gives a maximum conversion time of $(0.5\text{ MHz})^{-1} = 2$ µs. This is fine for the STM32H743ZI's 3.6 MSps ADC, which runs with speeds up to 240 MHz, so fM is set to 2.0 MHz.

Generation of fM is done by the FPGA. The FPGA system clock runs at 16 MHz. So, for generation of 2 MHz, the pre-scalar value should be 7 (since 0 to 7 is 8 counts) so that 16 MHz divided by 8 gives 2 MHz. Also, for 50% duty cycle the toggling of the clock is done at the exact half of pre-scalar value, i.e., 4. A snippet of the Verilog code is given below for better understanding.

```
always @(posedge CLK or negedge enable)
begin
  if(enable == 1'b0)
    temp2M <= 1'b0;
  else
  begin
    if (counter2M == 3'd7) // period, count from 0 to n-1
      counter2M <= 0;
    else
      counter2M <= counter2M + 1'b1;
    if (counter2M < 3'd4) // duty cycle, m cycles high, 50% if n/2
      temp2M <= 1'b1;
    else
      temp2M <= 1'b0;
  end
end
```

Code snippet 4.1 - Verilog code snippet for master clock of 2 MHz

It can be verified from the generated waveform in Figure 4.10 also. A closer look at the green waveform shows the period of the FPGA clock is 62.5 ns which is 16 MHz. Also, the orange waveform is the output 2 MHz master clock having time period 0.5 µs.

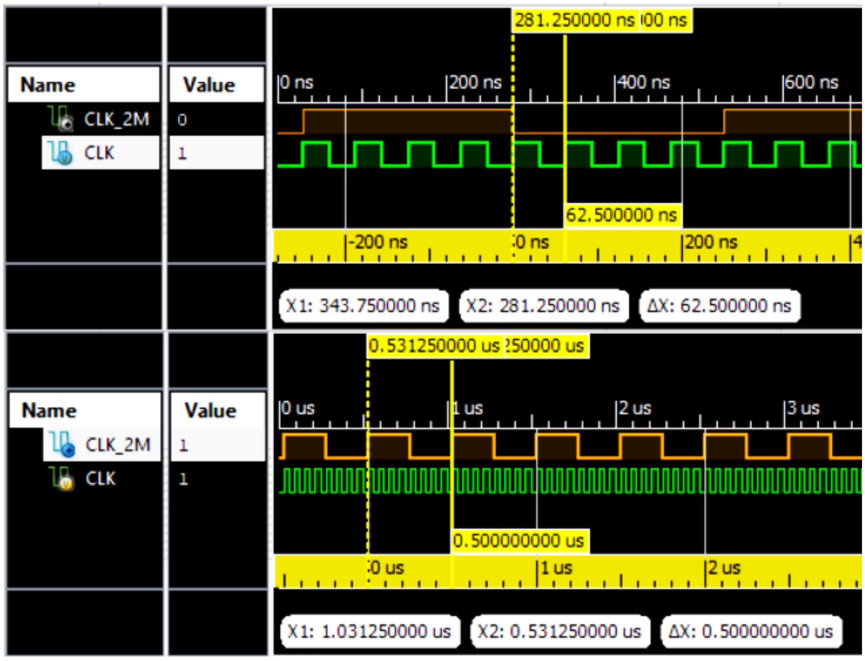


Figure 4.10 - Waveform of the generated 2MHz clock

4.2.4.2 The SH and ICG pulses

The SH-pulses control the integration time, and the ICG-pulses the moment the pixel-values are sent to the shift register. These pulses are also generated by the FPGA. The integration time depends on the period of SH pulse. The integration time can be as low as 10 µs (specified in the data sheet) but the highest integration time in this design is limited to 2 s, though it is customizable. The image sensor is operated in electronic shutter mode with time period of ICG being twice that of SH. This multiplication factor can be easily modified but higher the value of this factor more will be the acquisition time. Generally, higher multiplication factor is preferred when the intensity of light frequently changes. A snippet of the Verilog code with explicit description on how to change this multiplication factor (mentioned in comment) is given in Code snippet 4.2.

At each rising edge of the ICG pulse, a short pulse (ADC_start) is generated to initiate reading of the output voltage of the image sensor by the MCU. The generated output waveforms for SH and ICG satisfies the timing requirements as stated in the specification sheet for TCD1304AP (See Figure 4.5). This is shown in Figure 4.11.

```
/* The different integration time configurations
are done with electronic shutter mode ON and
keeping ICG=n*SH + 1, n being 2. User can change the
value of n by modifying the counter values indicated
by ICG and SH in comments.
*/
  always @(posedge CLK_2M or negedge INT10ms)
  begin
    if(INT10ms == 1'b0)
    begin
    counter <= 32'd0;
    counter1 <= 32'd0;
    end
    else
    begin
      if (counter == 32'd40001) //ICG
        counter <= 0;
      else
        counter <= counter + 1;
      if (counter < 32'd15)
        tempICG <= 1'b0;
      else
        tempICG <= 1'b1;
      if (counter1 == 32'd20000) //SH
        counter1 <= 0;
      else
        counter1 <= counter1 + 1;
      if (counter1 > 32'd0 && counter1 < 32'd5)
        tempSH <= 1'b1;
      else
        tempSH <= 1'b0;
    end
  end
```

Code snippet 4.2 - Verilog code snippet for generating SH and ICG pulses for 10 ms integration time

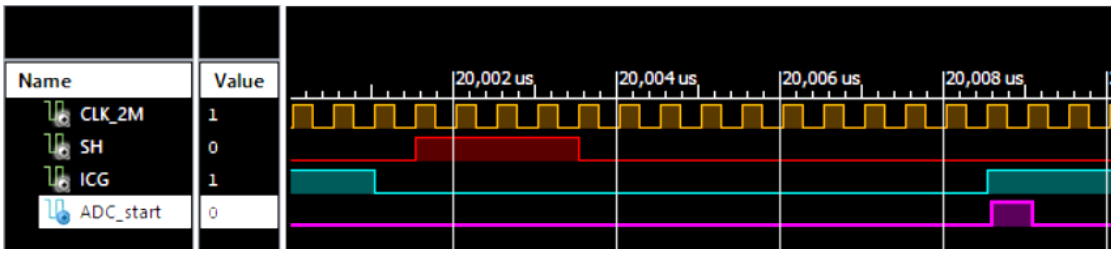


Figure 4.11 - Verification of timing requirements for SH and ICG pulses

4.2.4.3 The ADC trigger signal

The data-rate is 500 KHz, and to start the conversion of each pixel at the proper time, the ADC is triggered at the same rate. The ADC clock is generated by a 16-bit timer TIM4 of the STM32H743ZI MCU. Since TIM4 is present on APB1 bus which is running at 64 MHz (user-defined), so we need to divided it by 128 to get 500 KHz clock. Hence, the pre-scalar value is 127. TIM4 is enabled in an interrupt generated when the ICG-pulse is created, and disabled in a different interrupt created by the DMA-controller servicing the ADC.

4.2.4.4 ADC

The STM32H743ZI has a 16 bit 3.6 MSps ADC and it used to collect the pixel values from the CCD. The output from the CCD is clocked out at 1/4th of the frequency of the master clock, i.e., 0.5 MHz. To accurately sample at this rate the ADC is triggered by a timer (TIM4) running at this frequency.

The ADC runs with a typical frequency of 64 MHz. That means each cycle takes: $(64\text{ MHz})^{-1}$ = 15.625 ns. It's a 16-bit ADC, and each bit takes one cycle, so the conversion process itself of course takes: 16 x 15.625 ns = 0.25 µs. The output-rate is 0.5 MHz, so the total conversion time cannot exceed: $(0.50\text{ MHz})^{-1}$ = 2.0 µs. With 0.25 µs for a 16-bit conversion, there's (2 - 0.25) = 1.75 µs to sample in. This gives a sampling time (in ADC clock cycles) of: 1.75 µs / 15.625 ns = 112. Reading through the datasheet of STM32H743ZI it is found that the nearest available setting is 64.5 cycles. The converted values are stored in memory using DMA.

4.2.4.5 DMA

DMA stands for Direct Memory Access. It represents the fastest and easiest way of transferring data between memory and the peripherals. The DMA1 Stream 0 of the STM32H743ZI has been used in circular mode to store the ADC converted values. It is implemented as a double buffer, i.e., when the 1st half of the DMA gets filled up, data from the 2nd half is transferred to PC and when the 2nd half of the DMA gets filled up, data from the 1st half is sent to PC, thus avoiding any overwrite issue and simultaneously increasing the overall acquisition time.

4.2.4.6 USB

The data sent over USB are encoded using non-return-to-zero format: when a ”0” bit is transmitted, the D– and D+ differential lines switch their state. When a “1” bit is transmitted, no change occurs. However, when more than 6 consecutive “1” bits are to be transmitted, one “0” bit is stuffed after them to enable the receiver to synchronize its clock at an edge of incoming signal. The data lines do not have to be always differential. For several low-level signals, e. g. end of packet, the D– and D+ lines may go both to logical “0” or “1”.

All data are sent in packets. The packet contains an initial sync sequence, the packet identification number and optionally payload or control data along with its checksum. Some packets transmit payload data, while other packets are used for handshake and identification only. Three packets, token, data and handshake, usually form a transaction. The USB protocol is handled fully by the USB CDC library provided by ST Microelectronics.

After being requested by the host, each USB device identifies itself by a vendor number and an USB class. The communication device class at USB Full Speed (12 Mbps) is chosen for simplicity of coding.

4.2.5 Electronic interfacing of the detector

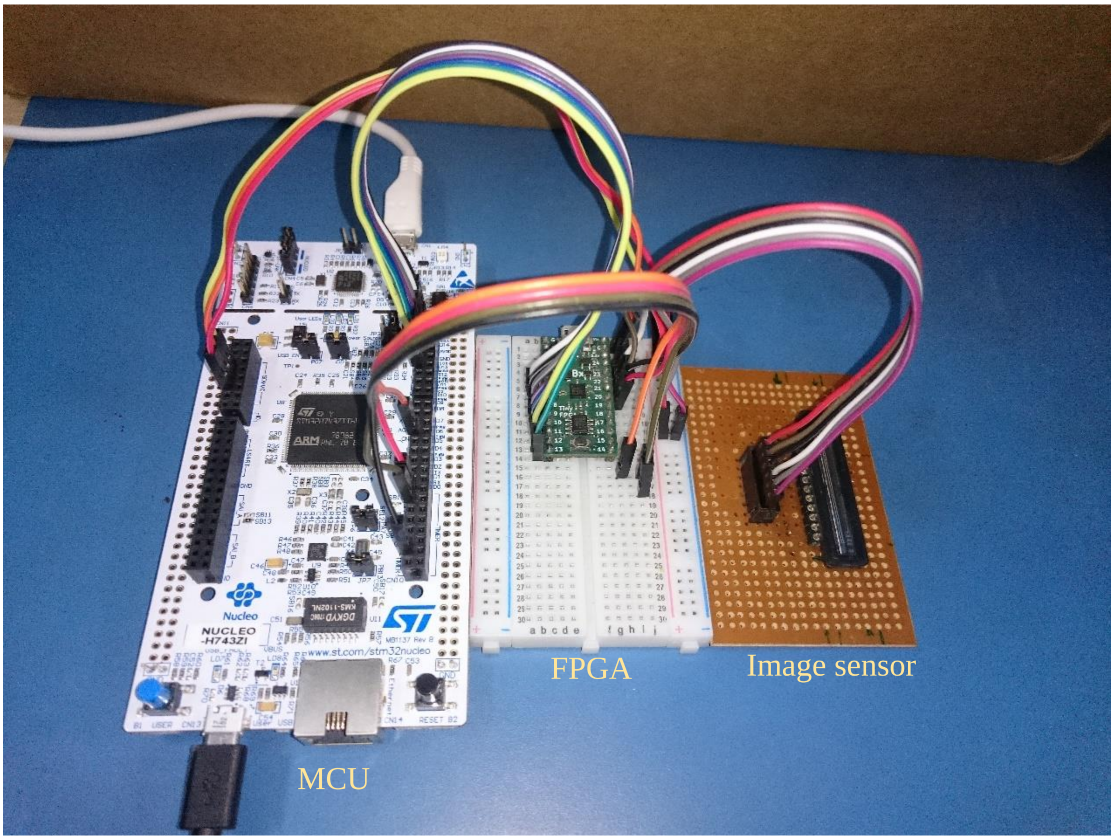


Figure 4.12 - Interface electronics for the CCD image sensor

4.3. Designing the GUI

The user interface on PC end has been designed with the help of App Designer by MATLAB. The user interface is designed is such a way that the user can set a wide range of integration time options – 10 ms, 100 ms, 200 ms, 500 ms, 1000 ms, 1500 ms and 2000 ms. Though no limit has been put on the number of sample averages but the user should remember that more the sample average more will be the acquisition time. For the sake of convenience, the y-axis of the plot is reversed since the CCD sensor gives low voltage on high intensity incident light. The user can take dark spectra or reference spectra and can zero it out. Also, two filtering algorithms have been implemented – Exponential Moving Average and Savitzky-Golay. The user can observe spectra either in one-shot mode or continuous mode by clicking on the 'Measure' or 'Monitor' respectively and can save it to a Microsoft Excel file. Various error messages and notification messages have been incorporated to aid the user operate the graphical interface properly. This interface is the not the final version and user may face some bugs here and there. Restarting the application may solve the issue. The user interface is shown below in Figure 4.13. The plot is not of an actual spectrum but obtained by using visible light and a very narrow slit almost at the middle position of the image sensor. The code for the same is attached in the appendix 7.8.

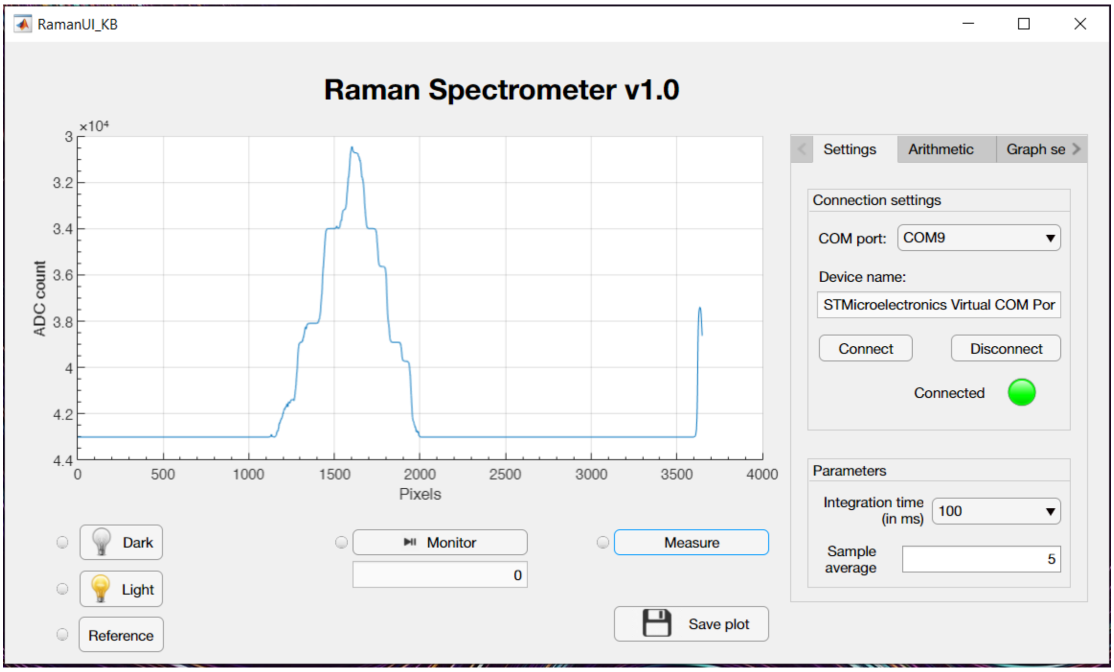


Figure 4.13 - The GUI for acquisition of Raman spectra

Chapter 5

5 Conclusion and Future scope

5.1. Summary of the work

The initial objective of the work was to design, develop and build a low-cost, portable Raman spectrometer. There are numerous portable and hand-held commercial Raman systems out there in the market as mentioned in section 1.3 of this report. A major drawback of these instruments are their sky-high prices. This is mainly because they can be used to identify and quantify various unknown samples. Now, this versatility may not be required in every sector. For example, a tea industry needs to only assess the quality of their tea production, they may not need a highly expensive versatile Raman device which could also work with other samples. Therefore, application-specific Raman systems are required. This would bring down the cost of the product. By proper choice of optical and electronic components such systems can be designed at an affordable cost.

We have been successful to a large extent in designing a portable Raman system. Still, much work is needed to materialise this idea and develop an actual commercial Raman device. In Chapter 3, the design methodologies of building a constant current source for the 785 nm laser used in this project were discussed. The built power supply not only sources constant current but also features a temperature feedback loop in association with a Thermo-electric controller to maintain the laser temperature at a constant value for its stable operation. Also, an optical head, consisting of least number of optical components, was set up to facilitate Raman back-scattering. This is also illustrated in details in section 3.5. Finally, a spectrum of pure benzene was taken with the help of an off-the-shelf detector using CMOS linear image sensor. Though some peaks corresponding to the Raman shift matched with the literatures but still there were many others that were absent. Also, there was significant amount of noise in the spectra. This goes to say that more fine-tuning is needed both on the hardware and software ends.

Keeping the afore-mentioned points in mind and also knowing the fact that CCD image sensors are superior to CMOS image sensors in terms of thermal noise, an electronic interface consisting of MCU and FPGA was developed for spectral acquisition. The CCD image sensor used in this project has complex timing requirements. The calculations and generation of pulses have been discussed in great details in section 4.2. Finally, a GUI was designed on the PC end with necessary filtering algorithms to get data from the image sensor and display the corresponding spectrum on PC.

5.2. Challenges faced

It is a very difficult job to design a Raman spectrometer from scratch. Many difficulties were faced right from the beginning of the project. The first challenge was how to build the basic structure of the optical head for back-scattering and finding the right components that can be procured from optical equipment companies online. Every component has to be compatible in size and specification for working properly. It was truly not an easy task to find compatible components online at the same place. Buying from multiple vendors will certainly increase the cost of the project, not to mention the different delivery delays. The second challenge was the procurement time. The work remained idle for a few months owing to delivery delays and other issues during the post-purchase days. More trouble was waiting till the actual design and development process started. Moreover, due to a limited time frame of around one year during M.Tech. and added complexity of the work, the project has been only partially completed and much work is remaining to achieve the set target, i.e., to witness a fully operational portable Raman system.

5.3. Future scope

The interface electronics for spectral acquisition along with the GUI can be used to develop a Raman spectrometer by using necessary optical components. Also, the assembly of the optical head being modular can be redesigned to suite various application. The long tubular part of the optical head also enables user to introduce various filters as and when required. The laser power source though looks bulky can be made to fit in one’s palm using Li-ion batteries and SMD components along with few minor changes. Since, the CCD image sensor has wide spectral range from visible to near-IR so one can get higher Raman shift by using different excitation source along with proper filters. Therefore, the flexibility that the design provides to a user is manifold.

5.4. Conclusion

The report demonstrates the feasibility of the implementation of a Raman system that combines many favourable qualities – low cost, ease of construction, operation etc. that can give satisfactory results. Though there are several limitations, and lots of room for improvement since the developed system is still a prototype still the strongest peak of pure benzene was achieved with 99.6% accuracy. This shows that that the system has potential and once perfected it may serve as a great tool not only in industrial sectors but also research labs.

6 References

[1] C. V. Raman and K. S. Krishnan, “A new type of secondary radiation [11],” *Nature*, vol. 121, no. 3048, pp. 501–502, 1928.

[2] A. Smekal, “Zur Quantentheorie der Streuung und Dispersion,” *Naturwissenschaften*, vol. 16, no. 31, pp. 612–613, 1928.

[3] “Serstech AB Business Intelligence. Robust for field use Please help us improve this material You are our ears and eyes in the ground/market Our competitive. - ppt download.” [Online]. Available: https://slideplayer.com/slide/10879706/. [Accessed: 15-May-2019].

[4] “NanoRam Handheld RAMAN System, B&W Tek | VWR.” [Online]. Available: https://us.vwr.com/store/product/13244534/nanoram-handheld-raman-system-b-w-tek. [Accessed: 14-May-2019].

[5] “TruScan^{TM} RM Handheld Raman Analyzer.” [Online]. Available: https://www.thermofisher.com/order/catalog/product/TRUSCANRM. [Accessed: 14-May-2019].

[6] “Welcome to Ramansystems - PinPointer.” [Online]. Available: http://www.ramansystems.com/english/pinpointer.htm. [Accessed: 14-May-2019].

[7] “The 4th Generation 785nm Handheld Raman Analyzer (450g) - Optosky.” [Online]. Available: https://optosky.com/raman-analyzer.html. [Accessed: 15-May-2019].

[8] A. Maréchal, “La diffusion résiduelle des surfaces polies et des réseaux,” *Opt. Acta Int. J. Opt.*, vol. 5, no. 3–4, pp. 70–74, 1958.

[9] C. D. Allemand, “Design Criteria for a Raman Spectrometer,” *Appl. Opt.*, vol. 9, no. 6, p. 1304, 2008.

[10] W. R. C. Somerville, E. C. Le Ru, P. T. Northcote, and P. G. Etchegoin, “High performance Raman spectroscopy with simple optical components,” *Am. J. Phys.*, vol. 78, no. 7, pp. 671–677, 2010.

[11] C. Mohr, C. L. Spencer, and M. Hippler, “Inexpensive Raman Spectrometer for Undergraduate and Graduate Experiments and Research,” *J. Chem. Educ.*, vol. 87, no. 3, pp. 326–330, 2010.

[12] A. Rahaman, N. R. Hoque, A. I. Talukder, K. M. Abedin, and A. F. M. Y. Haider, “Laser Raman Spectroscopy with Different Excitation Sources and Extension to Surface Enhanced Raman Spectroscopy,” vol. 2015, 2015.

[13] M. Gnyba, J. Smulko, A. Kwiatkowski, and P. Wierzba, “Portable Raman spectrometer - design rules and applications,” *Bull. Polish Acad. Sci. Tech. Sci.*, vol. 59, no. 3, pp. 325–329, 2011.

[14] R. S. Krishnan and R. K. Shankar, “Raman effect: History of the discovery,” *J. Raman Spectrosc.*, vol. 10, pp. 1–8, 1981.

[15] V. Mazet, C. Carteret, D. Brie, J. Idier, and B. Humbert, “Background removal from spectra by designing and minimising a non-quadratic cost function,” *Chemom. Intell. Lab. Syst.*, vol. 76, no. 2, pp. 121–133, 2005.

[16] T. Shimanouchi, “Tables of Molecular Vibrational Frequencies Consolidated Volume I,” pp. 1–160, 1972.

[17] J. Moreau and E. Rinnert, “Fast identification and quantification of BTEX coupling by Raman spectrometry and chemometrics,” *Analyst*, vol. 140, no. 10, pp. 3535–3542, 2015.

[18] X. Zhang, Q. Zhou, Y. Huang, Z. Li, and Z. Zhang, “Contrastive analysis of the Raman spectra of polychlorinated benzene: Hexachlorobenzene and benzene,” *Sensors*, vol. 11, no. 12, pp. 11510–11515, 2011.

[19] W. Krasser, H. Ervens, A. Fadini, and A. J. Renouprez, “Raman scattering of benzene and deuterated benzene chemisorbed on silica-supported nickel,” *J. Raman Spectrosc.*, vol. 9, no. 2, pp. 80–84, 1980.

[20] O. F. Benzene, “Wood-Phys Rev-1930-36-1431.pdf,” 1930.

7 Appendices

7.1. Laser current vs control voltage data set for cubic spline interpolation

Table 7-1 - Laser current vs control voltage data set for cubic spline interpolation

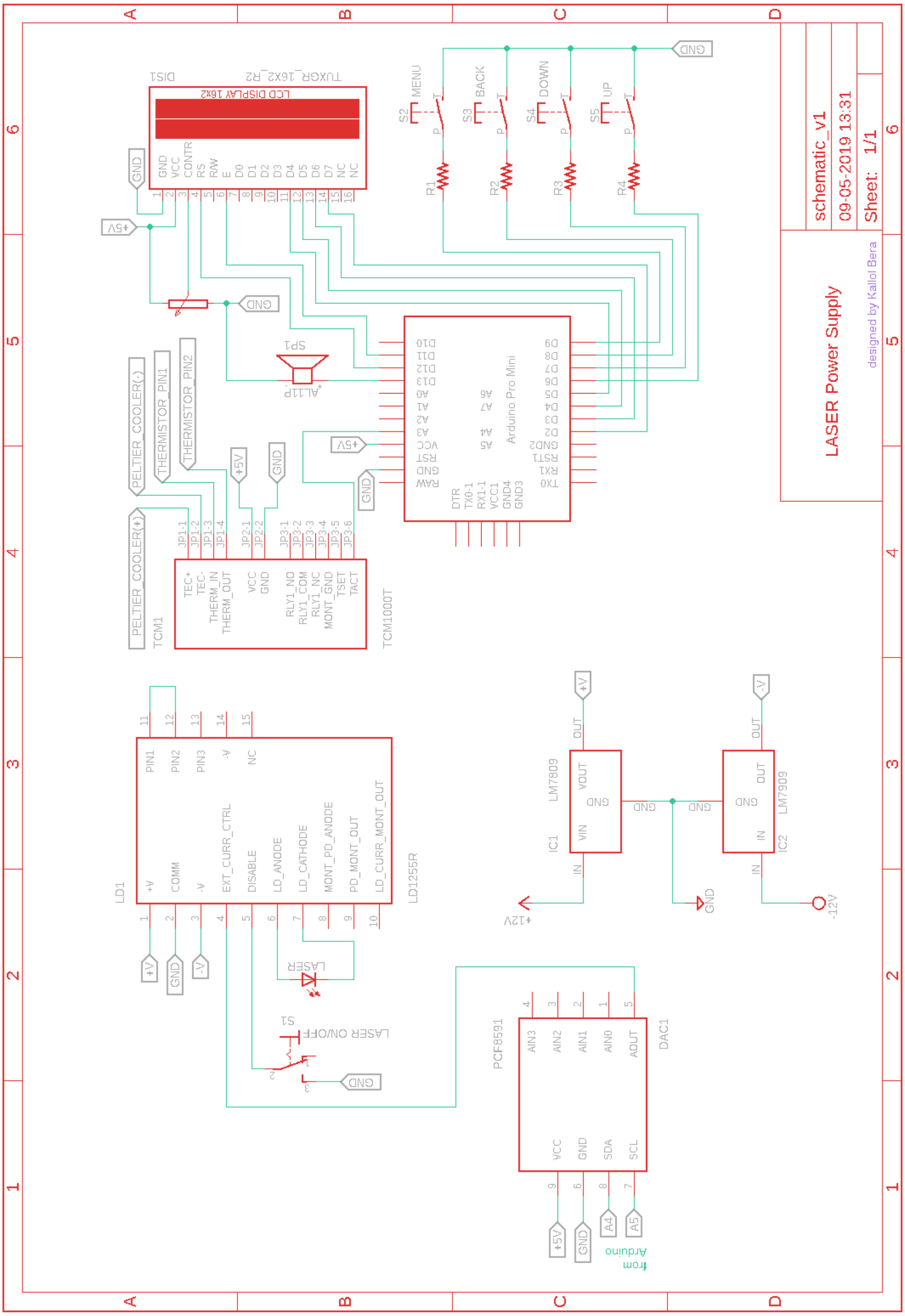
Sl. No.	8-bit control volt. level	Laser current (mA)	Sl. No.	8-bit control volt. level	Laser current (mA)
1	12	9.8	17	185	163.9
2	23	19.5	18	190	167.3
3	34	29.1	19	195	168.3
4	46	39.7	20	200	169
5	57	49.5	21	205	169.4
6	68	59.4	22	210	169.7
7	79	69.3	23	215	170
8	90	79.3	24	220	170.3
9	101	89.2	25	225	170.6
10	112	99.1	26	230	170.8
11	123	109.1	27	235	171
12	134	118.9	28	240	171.1
13	145	128.8	29	245	171.3
14	156	138.6	30	250	171.6
15	167	148.4	31	255	171.9
16	178	158.1			

7.2. Coefficients for the cubic spline interpolation of laser current

Table 7-2 - Coefficients for the cubic spline interpolation of laser current

Coeff. of x^3	Coeff. of x^2	Coeff. of x	Constant term
3.54E-05	-0.00158	0.894938	9.8
3.54E-05	-0.00041	0.872986	19.5
-1.73E-05	0.000756	0.876756	29.1
1.68E-05	0.000132	0.887416	39.7
-3.31E-05	0.000688	0.896439	49.5
4.04E-05	-0.0004	0.899556	59.4
-5.35E-05	0.00093	0.905337	69.3
2.33E-05	-0.00084	0.90637	79.3
3.52E-05	-6.55E-05	0.896455	89.2
-8.92E-05	0.001098	0.90781	99.1
9.62E-05	-0.00185	0.899577	109.1
-7.02E-05	0.001329	0.893883	118.9
3.45E-05	-0.00099	0.897619	128.8
7.53E-06	0.000148	0.888369	138.6
-0.00014	0.000397	0.894361	148.4
0.000116	-0.00421	0.85237	158.1
-0.00486	-0.00177	0.810472	163.9
0.005832	-0.07474	0.427913	167.3
-0.00166	0.012733	0.117876	168.3
0.000815	-0.01219	0.120583	169
1.66E-06	3.33E-05	0.059792	169.4
-2.16E-05	5.83E-05	0.06025	169.7
8.49E-05	-0.00027	0.059209	170
-0.00032	0.001007	0.062913	170.3
0.000387	-0.00376	0.049138	170.6
-0.00043	0.002042	0.040535	170.8
0.000532	-0.0044	0.028723	171
-9.79E-05	0.003574	0.024575	171.1
-0.00014	0.002106	0.052979	171.3
-0.00014	5.69E-16	0.063511	171.6

7.3. Schematic of laser power supply



```
////////////////////////////////////////////////////////////////////////////////
// 7.4. Arduino code for the user interface of the laser power supply
////////////////////////////////////////////////////////////////////////////////
#include <LiquidCrystal.h>
#include <Wire.h>
#include <math.h>

#define PCF8591 (0x90 >> 1) //I2C bus address

//LCD pins declaration
const int RS = 12; //Register Select
const int EN =11; //LCD Enable
const int D4 = 5; //Data pin 4
const int D5 = 4; //Data pin 5
const int D6 = 3; //Data pin 6
const int D7 = 2; //Data pin 7

//Button declaration
const int menuButton = 9;
const int backButton = 8;
const int upButton = 6;
const int downButton = 7;
const int laserStatusPin = 10;
const int speakerPin = 13;
//LCD instance creation
LiquidCrystal lcd(RS, EN, D4, D5, D6, D7);

//Input & Button Logic
const int numOfInputs = 4; //no. of buttons
const int inputPins[numOfInputs] = {downButton, upButton, menuButton, backButton};
int inputState[numOfInputs];
int lastInputState[numOfInputs] = {HIGH, HIGH, HIGH, HIGH};
bool inputFlags[numOfInputs] = {LOW, LOW, LOW, LOW};
long lastDebounceTime[numOfInputs] = {0,0,0,0};
long debounceDelay = 5; //debounce delay of 5 ms
long predefinedTime = 1000;
long lastMillis;

//Custom Character
byte leftArrow[] = {
  B00000,
  B00000,
  B00100,
  B01100,
  B11111,
  B01100,
  B00100,
  B00000
};
byte rightArrow[] = {
  B00000,
  B00000,
  B00100,
  B00110,
  B11111,
  B00110,
  B00100,
  B00000
};

byte continuousMode[8] = {
  B11111,
  B10001,
  B01110,
  B00100,
  B00100,
  B01110,
  B10001,
  B11111
};
```

```
//LCD Menu Logic
byte mA_min = 1;
byte mA_max = 254;
const int numOfScreens = 2;
int currentScreen = 0;
String screens[numOfScreens][2] = {{"Laser Current","mA"}, {"Laser temp", "degC"}};
float parameters[numOfScreens]={34.0,0.0};
byte setFlag = 0;
float kohm;
float mA_actual;
int manualFlag = 0;
int lastBackButtonState = HIGH;
int backButtonState;

void setup() {
  Wire.begin();
  lcd.begin(16, 2);
  lcd.createChar(0, rightArrow);
  lcd.createChar(1, leftArrow);
  lcd.createChar(2, continuousMode);
  pinMode(laserStatusPin, OUTPUT);
  for(int i = 0; i < numOfInputs; i++) {
    pinMode(inputPins[i], INPUT_PULLUP);
  }
  updateParameters();
  displayOnStart();
}

void loop() {
  laserStatus();
  int backButtonRead = digitalRead(backButton);
  if(backButtonRead != lastBackButtonState) {
  lastMillis = millis();
  }
  if(millis() - lastMillis > predefinedTime) {
    if(backButtonRead != backButtonState) {
      backButtonState = backButtonRead;
      if(backButtonState == LOW) {
        if(manualFlag == 0) {
          manualFlag = 1;
        }else {
          manualFlag = 0;
        }
      }
    }
  }
  lastBackButtonState = backButtonRead;
  setInputFlags();
  resolveInputFlags();
}

void setInputFlags() {
  if(manualFlag == 0) {
  for(int i = 0; i < numOfInputs; i++) {
    int reading = digitalRead(inputPins[i]);
    if (reading != lastInputState[i]) {
      lastDebounceTime[i] = millis();
    }
    if ((millis() - lastDebounceTime[i]) > debounceDelay) {
      if (reading != inputState[i]) {
        inputState[i] = reading;
        if (inputState[i] == LOW) {
          inputFlags[i] = HIGH;
          tone(speakerPin,1500);
          delay(50);
          noTone(speakerPin);
        }
      }
    }
```

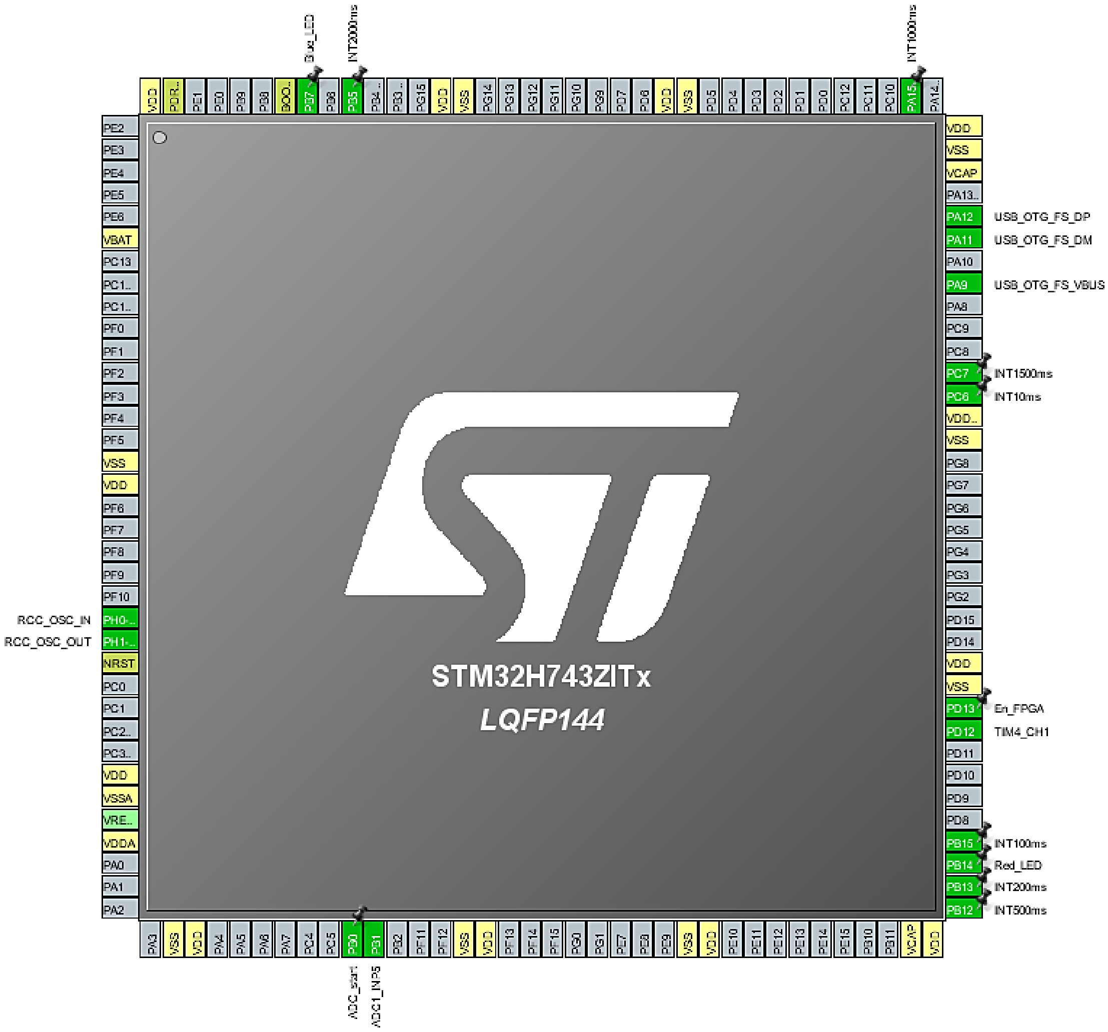
```
    lastInputState[i] = reading;
  }
  }else {
    for(int i = 0; i < numOfInputs; i++) {
      int reading = digitalRead(inputPins[i]);
      delay(debounceDelay);
      if (digitalRead(inputPins[i]) == LOW) {
        inputFlags[i] = HIGH;
      }
    }
  }
}

void resolveInputFlags() {
  for(int i = 0; i < numOfInputs; i++) {
    if(inputFlags[i] == HIGH) {
      inputAction(i);
      inputFlags[i] = LOW;
      printScreen();
      updateParameters();
    }
  }
}

void inputAction(int input) {
  if(input == 2) {
    if (currentScreen == 0) {
      setFlag = 1;
    }
  }
  if(input == 0 && setFlag == 1){
    parameters[currentScreen]--;
  }else if(input == 1 && setFlag == 1){
    parameters[currentScreen]++;
  }else if(input == 3){
    setFlag = 0;
    updateParameters();
  }
  if(input == 0 && setFlag == 0){
    if (currentScreen == 0) {
      currentScreen = numOfScreens-1;
    }else{
      currentScreen--;
    }
  }else if(input == 1 && setFlag == 0) {
    if (currentScreen == numOfScreens-1) {
      currentScreen = 0;
    }else{
      currentScreen++;
    }
  }
  if(input == 2 && currentScreen == 1){
    lcd.clear();
    lcd.print("Invalid input!");
  }
}

void printScreen() {
  lcd.clear();
  if(setFlag == 1){
    mA_actual = (float)calcActual_mA(parameters[0]);
    if(mA_actual <= 0.0) {
      lcd.print("Min mA reached!");
      tone(speakerPin,1500,100);
      noTone(speakerPin);
    }else if(mA_actual >= 255.0) {
      lcd.print("Max mA reached!");
      tone(speakerPin,1500,100);
      noTone(speakerPin);
    }else{
```

```
       lcd.print("Adj Laser Curr");
       lcd.setCursor(0,1);
       lcd.print((int)(round(mA_actual)));
      lcd.print(" ");
      lcd.print(screens[currentScreen][1]);
      lcd.setCursor(15,1);
      lcd.write((byte)0);
      lcd.setCursor(11,1);
      lcd.write((byte)1);
    }
    if(manualFlag == 1) {
      lcd.setCursor(15,0);
      lcd.write((byte)2);
    }
  }else{
    if(currentScreen == 0) {
      lcd.print(screens[currentScreen][0]);
      lcd.setCursor(0,1);
      lcd.print(mA_actual,1);
      lcd.print(" ");
      lcd.print(screens[currentScreen][1]);
      if(manualFlag == 1) {
        lcd.setCursor(15,0);
        lcd.write((byte)2);
      }
    }else if(currentScreen == 1) {
      lcd.print(screens[currentScreen][0]);
      lcd.setCursor(0,1);
      lcd.print(parameters[currentScreen],1);
      lcd.print(" ");
      lcd.print(screens[currentScreen][1]);
      if(manualFlag == 1) {
        lcd.setCursor(15,0);
        lcd.write((byte)2);
      }
    }
  }
}

void updateParameters(){
  //Control the laser current via DAC
  DACout();
  measureTemp();
}

void DACout() {
  if(parameters[0] <= mA_min) parameters[0] = mA_min;
  if(parameters[0] >= mA_max) parameters[0] = mA_max;
  Wire.beginTransmission(PCF8591); //wake up
  Wire.write(0x40); //control byte
  Wire.write((int)parameters[0]);
  Wire.endTransmission();
  mA_actual = (float)calcActual_mA(parameters[0]);
}

void measureTemp() {
  //Read the Laser temperature
  kohm = (5.0/1023.0) * analogRead(A3) * 10;
  parameters[1] = (1/(0.00113 + 0.000234*log(kohm*1000) + 8.78E-8*pow(log(kohm*1000),3))) - 273.15;
}

void displayOnStart() {
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Booting up...");
  lcd.blink();
  delay(2000);
  lcd.noBlink();
  delay(500);
```

```
  lcd.clear();
  lcd.print("Ready!");
  tone(speakerPin,1500,100);
  delay(150);
  tone(speakerPin,1500,100);
  noTone(speakerPin);
  delay(500);
  lcd.clear();
  lcd.print("Laser Current");
  lcd.setCursor(0,1);
  lcd.print(mA_actual, 1);
  lcd.print(" mA");
}

void alwaysOnDisplay() {
  if(manualFlag == 0) {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Laser Current");
    lcd.setCursor(0,1);
    lcd.print(mA_actual);
    lcd.print(" mA");
  }else if(manualFlag == 1) {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Laser Current");
    lcd.setCursor(14,1);
    lcd.write((byte)2);
    lcd.setCursor(0,1);
    lcd.print(mA_actual);
    lcd.print(" mA");
  }
}

void laserStatus() {
  if(parameters[1] >= 35.0) {
    delay(100);
    digitalWrite(laserStatusPin, HIGH);
    delay(100);
    digitalWrite(laserStatusPin, LOW);
    delay(100);
    digitalWrite(laserStatusPin, HIGH);
    delay(100);
    digitalWrite(laserStatusPin, LOW);
    tone(speakerPin,1500,100);
    delay(150);
    tone(speakerPin,1500,100);
  }
}

double calcActual_mA(float x) {
  double mA;
  if(x<12 || (x>=12 && x<23)) mA=(3.54304614237795e-05)*pow((x-12),3)+(-0.00158242836748060)*pow((x-12),2)+(0.894937808028191)*(x-12)+(9.80000000000000);
  if(x>=23 && x<34) mA=(3.54304614237832e-05)*pow((x-23),3)+(-0.000413223140495876)*pow((x-23),2)+(0.872985641440450)*(x-23)+(19.5000000000000);
  if(x>=34 && x<46) mA=(-1.73225107701979e-05)*pow((x-34),3)+(0.000755982086489008)*pow((x-34),2)+(0.876755989846374)*(x-34)+(29.1000000000000);
  if(x>=46 && x<57) mA=(1.68327847382305e-05)*pow((x-46),3)+(0.000132371698761841)*pow((x-46),2)+(0.887416235269385)*(x-46)+(39.7000000000000);
  if(x>=57 && x<68) mA=(-3.31000251940561e-05)*pow((x-57),3)+(0.000687853595123457)*pow((x-57),2)+(0.896438713502123)*(x-57)+(49.5000000000000);
  if(x>=68 && x<79) mA=(4.04358359478301e-05)*pow((x-68),3)+(-0.000404447236280354)*pow((x-68),2)+(0.899556183449396)*(x-68)+(59.4000000000000);
  if(x>=79 && x<90) mA=(-5.35118385071133e-05)*pow((x-79),3)+(0.000929935349998079)*pow((x-79),2)+(0.905336552700291)*(x-79)+(69.3000000000000);
  if(x>=90 && x<101) mA=(2.33485579003166e-05)*pow((x-90),3)+(-0.000835955320736760)*pow((x-90),2)+(0.906370333022167)*(x-90)+(79.3000000000000);
  if(x>=101 && x<112) mA=(3.52490869959979e-05)*pow((x-101),3)+(-6.54529100262930e-05)*pow((x-101),2)+(0.896454842483773)*(x-101)+(89.2000000000000);
```

```
  if(x>=112 && x<123) mA=(-8.92134257941381e-05)*pow((x-112),3)+(0.00109776696084166)*pow((x-112),2)+(0.907810297042742)*(x-112)+(99.1000000000000);
  if(x>=123 && x<134) mA=(9.62101759100796e-05)*pow((x-123),3)+(-0.00184627609036487)*pow((x-123),2)+(0.899576696617986)*(x-123)+(109.100000000000);
  if(x>=134 && x<145) mA=(-7.02328375757248e-05)*pow((x-134),3)+(0.00132865971466777)*pow((x-134),2)+(0.893882916485318)*(x-134)+(118.900000000000);
  if(x>=145 && x<156) mA=(3.44582142124956e-05)*pow((x-145),3)+(-0.000989023925331199)*pow((x-145),2)+(0.897618910168021)*(x-145)+(128.800000000000);
  if(x>=156 && x<167) mA=(7.53146081594648e-06)*pow((x-156),3)+(0.000148097143681126)*pow((x-156),2)+(0.888368715569870)*(x-156)+(138.600000000000);
  if(x>=167 && x<178) mA=(-0.000139715537566482)*pow((x-167),3)+(0.000396635350607431)*pow((x-167),2)+(0.894360773007043)*(x-167)+(148.400000000000);
  if(x>=178 && x<185) mA=(0.000116311422678802)*pow((x-178),3)+(-0.00421397738908643)*pow((x-178),2)+(0.852370010583774)*(x-178)+(158.100000000000);
  if(x>=185 && x<190) mA=(-0.00486459674824755)*pow((x-185),3)+(-0.00177143751283155)*pow((x-185),2)+(0.810472106270348)*(x-185)+(163.900000000000);
  if(x>=190 && x<195) mA=(0.00583155874637034)*pow((x-190),3)+(-0.0747403887365449)*pow((x-190),2)+(0.427912975023466)*(x-190)+(167.300000000000);
  if(x>=195 && x<200) mA=(-0.00166163823723385)*pow((x-195),3)+(0.0127329924590103)*pow((x-195),2)+(0.117875993635793)*(x-195)+(168.300000000000);
  if(x>=200 && x<205) mA=(0.000814994202565295)*pow((x-200),3)+(-0.0121915810994975)*pow((x-200),2)+(0.120583050433356)*(x-200)+(169);
  if(x>=205 && x<210) mA=(1.66142697235749e-06)*pow((x-205),3)+(3.33319389818884e-05)*pow((x-205),2)+(0.0597918046307782)*(x-205)+(169.400000000000);
  if(x>=210 && x<215) mA=(-2.16399104543147e-05)*pow((x-210),3)+(5.82533435672466e-05)*pow((x-210),2)+(0.0602497310435239)*(x-210)+(169.700000000000);
  if(x>=215 && x<220) mA=(8.48982148446742e-05)*pow((x-215),3)+(-0.000266345313247471)*pow((x-215),2)+(0.0592092711951228)*(x-215)+(170);
  if(x>=220 && x<225) mA=(-0.000317952948924613)*pow((x-220),3)+(0.00100712790942265)*pow((x-220),2)+(0.0629131841759986)*(x-220)+(170.300000000000);
  if(x>=225 && x<230) mA=(0.000386913580854273)*pow((x-225),3)+(-0.00376216632444653)*pow((x-225),2)+(0.0491379921008792)*(x-225)+(170.600000000000);
  if(x>=230 && x<235) mA=(-0.000429701374492978)*pow((x-230),3)+(0.00204153738836756)*pow((x-230),2)+(0.0405348474204844)*(x-230)+(170.800000000000);
  if(x>=235 && x<240) mA=(0.000531891917117915)*pow((x-235),3)+(-0.00440398322902712)*pow((x-235),2)+(0.0287226182171866)*(x-235)+(171); //27
  if(x>=240 && x<245) mA=(-9.78662939785445e-05)*pow((x-240),3)+(0.00357439552774160)*pow((x-240),2)+(0.0245746797107590)*(x-240)+(171.100000000000);
  if(x>=245 && x<250) mA=(-0.000140426741204191)*pow((x-245),3)+(0.00210640111806344)*pow((x-245),2)+(0.0529786629397842)*(x-245)+(171.300000000000);
  if((x>=250 && x<255) || x>=255) mA=(-0.000140426741204191)*pow((x-250),3)+(5.68989300120393e-16)*pow((x-250),2)+(0.0635106685301042)*(x-250)+(171.600000000000);
  return mA;
}
```

7.5. Programming the MCU



The STM32 Nucleo-H743ZI development board is programmed with the help of STM32 CubeMX and Keil µVision 5. Only those files are given here that are modified. Rest of the files generated by STM32 CubeMX are kept as it is. Though the I/O pin configurations can be understood from the above figure, still they are given below.

PA9: USB_OTG_FS_VBUS	PB13: INT200ms
PA11: USB_OTG_FS_DM	PB14: Red_LED
PA12: USB_OTG_FS_DP	PB15: INT100ms
PA15: INT1000ms	PC6: INT10ms
PB0: ADC_start	PC7: INT1500ms
PB1: ADC1_INP5	PD12: TIM4_CH1
PB5: INT2000ms	PD13: En_FPGA
PB7: Blue_LED	PH0: RCC_OSC_IN
PB12: INT500ms	PH1: RCC_OSC_OUT

```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  ** This notice applies to any and all portions of this file
  * that are not between comment pairs USER CODE BEGIN and
  * USER CODE END. Other portions of this file, whether
  * inserted by the user or by software development tools
  * are owned by their respective copyright owners.
  *
  * COPYRIGHT(c) 2019 STMicroelectronics
  *
  * Redistribution and use in source and binary forms, with or without modification,
  * are permitted provided that the following conditions are met:
  *   1. Redistributions of source code must retain the above copyright notice,
  *      this list of conditions and the following disclaimer.
  *   2. Redistributions in binary form must reproduce the above copyright notice,
  *      this list of conditions and the following disclaimer in the documentation
  *      and/or other materials provided with the distribution.
  *   3. Neither the name of STMicroelectronics nor the names of its contributors
  *      may be used to endorse or promote products derived from this software
  *      without specific prior written permission.
  *
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
  * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "usb_device.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "usbd_cdc_if.h"
#include "stm32h7xx_it.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;
DMA_HandleTypeDef hdma_adc1;

TIM_HandleTypeDef htim4;
```

```
/* USER CODE BEGIN PV */
uint8_t flag_ADCstart = 0;
uint8_t flag_DMAConvCmplt = 0;
uint8_t flag_DMAHalfConvCmplt = 0;
uint16_t adcVal[3648];
uint16_t adcBuff[7388];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_TIM4_Init(void);
static void MX_ADC1_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_TIM4_Init();
  MX_ADC1_Init();
  MX_USB_DEVICE_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    if(flag_ADCstart==1) //if interrupt received
    {
      HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1); //start ADC clock
```

```
      HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adcBuff, 7388); //start ADC+DMA
      flag_ADCstart = 0;
    }
    if((flag_DMAHalfConvCmplt || flag_DMAConvCmplt))
      {
      if(flag_DMAHalfConvCmplt == 1) //if DMA is 50% filled
      {
        flag_DMAHalfConvCmplt = 0;
        HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_1); //stop ADC clock
        CDC_Transmit_FS((uint8_t*)adcBuff,7388); //transmit to PC
      }
      else if(flag_DMAConvCmplt == 1) //if DMA is 100% filled
      {
        flag_DMAConvCmplt = 0;
        HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_1);
        for(int i = 3726; i < 7374; i++) adcVal[i-3726]=adcBuff[i];
        CDC_Transmit_FS((uint8_t*)adcVal,7388);
      }
    }
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};

  /**Supply configuration update enable
  */
  MODIFY_REG(PWR->CR3, PWR_CR3_SCUEN, 0);
  /**Configure the main internal regulator output voltage
  */
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);

  while ((PWR->D3CR & (PWR_D3CR_VOSRDY)) != PWR_D3CR_VOSRDY)
  {

  }
  /**Macro to configure the PLL clock source
  */
  __HAL_RCC_PLL_PLLSOURCE_CONFIG(RCC_PLLSOURCE_HSE);
  /**Initializes the CPU, AHB and APB busses clocks
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 1;
  RCC_OscInitStruct.PLL.PLLN = 48;
  RCC_OscInitStruct.PLL.PLLP = 6;
  RCC_OscInitStruct.PLL.PLLQ = 8;
  RCC_OscInitStruct.PLL.PLLR = 2;
  RCC_OscInitStruct.PLL.PLLRGE = RCC_PLL1VCIRANGE_3;
  RCC_OscInitStruct.PLL.PLLVCOSEL = RCC_PLL1VCOWIDE;
  RCC_OscInitStruct.PLL.PLLFRACN = 0;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /**Initializes the CPU, AHB and APB busses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2
                              |RCC_CLOCKTYPE_D3PCLK1|RCC_CLOCKTYPE_D1PCLK1;
```

```c
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.SYSCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB3CLKDivider = RCC_APB3_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_APB1_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_APB2_DIV1;
  RCC_ClkInitStruct.APB4CLKDivider = RCC_APB4_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_ADC|RCC_PERIPHCLK_USB;
  PeriphClkInitStruct.PLL2.PLL2M = 1;
  PeriphClkInitStruct.PLL2.PLL2N = 19;
  PeriphClkInitStruct.PLL2.PLL2P = 1;
  PeriphClkInitStruct.PLL2.PLL2Q = 2;
  PeriphClkInitStruct.PLL2.PLL2R = 2;
  PeriphClkInitStruct.PLL2.PLL2RGE = RCC_PLL2VCIRANGE_3;
  PeriphClkInitStruct.PLL2.PLL2VCOSEL = RCC_PLL2VCOMEDIUM;
  PeriphClkInitStruct.PLL2.PLL2FRACN = 0;
  PeriphClkInitStruct.UsbClockSelection = RCC_USBCLKSOURCE_PLL;
  PeriphClkInitStruct.AdcClockSelection = RCC_ADCCLKSOURCE_PLL2;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief ADC1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC1_Init(void)
{

  /* USER CODE BEGIN ADC1_Init 0 */

  /* USER CODE END ADC1_Init 0 */

  ADC_MultiModeTypeDef multimode = {0};
  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC1_Init 1 */

  /* USER CODE END ADC1_Init 1 */
  /**Common config
  */
  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
  hadc1.Init.Resolution = ADC_RESOLUTION_16B;
  hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  hadc1.Init.LowPowerAutoWait = DISABLE;
  hadc1.Init.ContinuousConvMode = DISABLE;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv = ADC_EXTERNALTRIG_T4_TRGO;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
  hadc1.Init.ConversionDataManagement = ADC_CONVERSIONDATA_DMA_CIRCULAR;
  hadc1.Init.Overrun = ADC_OVR_DATA_PRESERVED;
  hadc1.Init.LeftBitShift = ADC_LEFTBITSHIFT_NONE;
  hadc1.Init.BoostMode = DISABLE;
  hadc1.Init.OversamplingMode = DISABLE;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }
  /**Configure the ADC multi-mode
```

```
  */
  multimode.Mode = ADC_MODE_INDEPENDENT;
  if (HAL_ADCEx_MultiModeConfigChannel(&hadc1, &multimode) != HAL_OK)
  {
    Error_Handler();
  }
  /**Configure Regular Channel
  */
  sConfig.Channel = ADC_CHANNEL_5;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SamplingTime = ADC_SAMPLETIME_64CYCLES_5;
  sConfig.SingleDiff = ADC_SINGLE_ENDED;
  sConfig.OffsetNumber = ADC_OFFSET_NONE;
  sConfig.Offset = 0;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */

}

/**
  * @brief TIM4 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM4_Init(void)
{

  /* USER CODE BEGIN TIM4_Init 0 */

  /* USER CODE END TIM4_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM4_Init 1 */

  /* USER CODE END TIM4_Init 1 */
  htim4.Instance = TIM4;
  htim4.Init.Prescaler = 0;
  htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim4.Init.Period = 127;
  htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim4) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 16;
```

```
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM4_Init 2 */

  /* USER CODE END TIM4_Init 2 */
  HAL_TIM_MspPostInit(&htim4);

}

/**
  * Enable DMA controller clock
  */
static void MX_DMA_Init(void)
{
  /* DMA controller clock enable */
  __HAL_RCC_DMA1_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA1_Stream0_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Stream0_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Stream0_IRQn);

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, INT500ms_Pin|INT200ms_Pin|Red_LED_Pin|INT100ms_Pin
                          |INT2000ms_Pin|Blue_LED_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(En_FPGA_GPIO_Port, En_FPGA_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, INT10ms_Pin|INT1500ms_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(INT1000ms_GPIO_Port, INT1000ms_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : ADC_start_Pin */
  GPIO_InitStruct.Pin = ADC_start_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(ADC_start_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : INT500ms_Pin INT200ms_Pin Red_LED_Pin INT100ms_Pin
                           INT2000ms_Pin Blue_LED_Pin */
  GPIO_InitStruct.Pin = INT500ms_Pin|INT200ms_Pin|Red_LED_Pin|INT100ms_Pin
                          |INT2000ms_Pin|Blue_LED_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
```

```
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : En_FPGA_Pin */
  GPIO_InitStruct.Pin = En_FPGA_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(En_FPGA_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : INT10ms_Pin INT1500ms_Pin */
  GPIO_InitStruct.Pin = INT10ms_Pin|INT1500ms_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pin : INT1000ms_Pin */
  GPIO_InitStruct.Pin = INT1000ms_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(INT1000ms_GPIO_Port, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI0_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI0_IRQn);

}

/* USER CODE BEGIN 4 */
void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc)
{
  /* Prevent unused argument(s) compilation warning */
  UNUSED(hadc);
  flag_DMAHalfConvCmplt = 1;
  flag_DMAConvCmplt = 0;
}
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
  /* Prevent unused argument(s) compilation warning */
  UNUSED(hadc);
  flag_DMAHalfConvCmplt = 0;
  flag_DMAConvCmplt = 1;
}
void CDC_ReceiveCmpltCallback(uint8_t *buf, uint32_t len)
{
  /*
  The control bytes are used by the application on PC end
  to initiate and terminate data acquisition.
  */
  if(*buf == 255) //initiate at 10ms integ. time
  {
    HAL_GPIO_TogglePin(Red_LED_GPIO_Port, Red_LED_Pin);
    HAL_GPIO_WritePin(INT10ms_GPIO_Port,INT10ms_Pin,GPIO_PIN_SET);
  }

  if(*buf == 254) //terminate current initialization
  {
    HAL_GPIO_TogglePin(Blue_LED_GPIO_Port, Blue_LED_Pin);
    HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_1);
    HAL_GPIO_WritePin(INT10ms_GPIO_Port,INT10ms_Pin,GPIO_PIN_RESET);
  }
  if(*buf == 253) //initiate at 100ms integ. time
  {
    HAL_GPIO_TogglePin(Red_LED_GPIO_Port, Red_LED_Pin);
    HAL_GPIO_WritePin(INT100ms_GPIO_Port,INT100ms_Pin,GPIO_PIN_SET);
  }

  if(*buf == 252) //terminate current initialization
  {
```

```
    HAL_GPIO_TogglePin(Blue_LED_GPIO_Port, Blue_LED_Pin);
    HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_1);
    HAL_GPIO_WritePin(INT100ms_GPIO_Port,INT100ms_Pin,GPIO_PIN_RESET);
  }
  if(*buf == 251) //initiate at 200ms integ. time
  {
    HAL_GPIO_TogglePin(Red_LED_GPIO_Port, Red_LED_Pin);
    HAL_GPIO_WritePin(INT200ms_GPIO_Port,INT200ms_Pin,GPIO_PIN_SET);
  }

  if(*buf == 250) //terminate current initialization
  {
    HAL_GPIO_TogglePin(Blue_LED_GPIO_Port, Blue_LED_Pin);
    HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_1);
    HAL_GPIO_WritePin(INT200ms_GPIO_Port,INT200ms_Pin,GPIO_PIN_RESET);
  }
  if(*buf == 249) //initiate at 500ms integ. time
  {
    HAL_GPIO_TogglePin(Red_LED_GPIO_Port, Red_LED_Pin);
    HAL_GPIO_WritePin(INT500ms_GPIO_Port,INT500ms_Pin,GPIO_PIN_SET);
  }
  if(*buf == 248) //terminate current initialization
  {
    HAL_GPIO_TogglePin(Blue_LED_GPIO_Port, Blue_LED_Pin);
    HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_1);
    HAL_GPIO_WritePin(INT500ms_GPIO_Port,INT500ms_Pin,GPIO_PIN_RESET);
  }
  if(*buf == 247) //initiate at 1000ms integ. time
  {
    HAL_GPIO_TogglePin(Red_LED_GPIO_Port, Red_LED_Pin);
    HAL_GPIO_WritePin(INT1000ms_GPIO_Port,INT1000ms_Pin,GPIO_PIN_SET);
  }

  if(*buf == 246) //terminate current initialization
  {
    HAL_GPIO_TogglePin(Blue_LED_GPIO_Port, Blue_LED_Pin);
    HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_1);
    HAL_GPIO_WritePin(INT1000ms_GPIO_Port,INT1000ms_Pin,GPIO_PIN_RESET);
  }
  if(*buf == 245) //initiate at 1500ms integ. time
  {
    HAL_GPIO_TogglePin(Red_LED_GPIO_Port, Red_LED_Pin);
    HAL_GPIO_WritePin(INT1500ms_GPIO_Port,INT1500ms_Pin,GPIO_PIN_SET);
  }

  if(*buf == 244) //terminate current initialization
  {
    HAL_GPIO_TogglePin(Blue_LED_GPIO_Port, Blue_LED_Pin);
    HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_1);
    HAL_GPIO_WritePin(INT1500ms_GPIO_Port,INT1500ms_Pin,GPIO_PIN_RESET);
  }
  if(*buf == 243) //initiate at 2000ms integ. time
  {
    HAL_GPIO_TogglePin(Red_LED_GPIO_Port, Red_LED_Pin);
    HAL_GPIO_WritePin(INT2000ms_GPIO_Port,INT2000ms_Pin,GPIO_PIN_SET);
  }

  if(*buf == 242) //terminate current initialization
  {
    HAL_GPIO_TogglePin(Blue_LED_GPIO_Port, Blue_LED_Pin);
    HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_1);
    HAL_GPIO_WritePin(INT2000ms_GPIO_Port,INT2000ms_Pin,GPIO_PIN_RESET);
  }
  if(*buf == 241) //for checking connection
  {
    HAL_GPIO_TogglePin(Blue_LED_GPIO_Port, Blue_LED_Pin);
    CDC_Transmit_FS(buf, len);
  }
}
/* USER CODE END 4 */
```

```
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */

  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32h7xx_it.c
  * @brief   Interrupt Service Routines.
  ******************************************************************************
  *
  * COPYRIGHT(c) 2019 STMicroelectronics
  *
  * Redistribution and use in source and binary forms, with or without modification,
  * are permitted provided that the following conditions are met:
  *   1. Redistributions of source code must retain the above copyright notice,
  *      this list of conditions and the following disclaimer.
  *   2. Redistributions in binary form must reproduce the above copyright notice,
  *      this list of conditions and the following disclaimer in the documentation
  *      and/or other materials provided with the distribution.
  *   3. Neither the name of STMicroelectronics nor the names of its contributors
  *      may be used to endorse or promote products derived from this software
  *      without specific prior written permission.
  *
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
  * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32h7xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */

/* USER CODE END TD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
extern uint8_t flag_ADCstart;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

```

```c
/* USER CODE END 0 */

/* External variables --------------------------------------------------------*/
extern PCD_HandleTypeDef hpcd_USB_OTG_FS;
extern DMA_HandleTypeDef hdma_adc1;
extern TIM_HandleTypeDef htim4;
/* USER CODE BEGIN EV */

/* USER CODE END EV */

/******************************************************************************/
/*           Cortex Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */

  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */

  /* USER CODE END NonMaskableInt_IRQn 1 */
}

/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */

  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */

  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}

/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */

  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}
```

```c

/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */

  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}

/**
  * @brief This function handles System service call via SWI instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */

  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */

  /* USER CODE END SVCall_IRQn 1 */
}

/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */

  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */

  /* USER CODE END DebugMonitor_IRQn 1 */
}

/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */

  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */

  /* USER CODE END PendSV_IRQn 1 */
}

/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */

  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */

  /* USER CODE END SysTick_IRQn 1 */
}

/******************************************************************************/
```

```
/* STM32H7xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32h7xx.s).                    */
/******************************************************************************/

/**
  * @brief This function handles EXTI line0 interrupt.
  */
void EXTI0_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI0_IRQn 0 */
  flag_ADCstart = 1;
  /* USER CODE END EXTI0_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
  /* USER CODE BEGIN EXTI0_IRQn 1 */

  /* USER CODE END EXTI0_IRQn 1 */
}

/**
  * @brief This function handles DMA1 stream0 global interrupt.
  */
void DMA1_Stream0_IRQHandler(void)
{
  /* USER CODE BEGIN DMA1_Stream0_IRQn 0 */

  /* USER CODE END DMA1_Stream0_IRQn 0 */
  HAL_DMA_IRQHandler(&hdma_adc1);
  /* USER CODE BEGIN DMA1_Stream0_IRQn 1 */

  /* USER CODE END DMA1_Stream0_IRQn 1 */
}

/**
  * @brief This function handles TIM4 global interrupt.
  */
void TIM4_IRQHandler(void)
{
  /* USER CODE BEGIN TIM4_IRQn 0 */

  /* USER CODE END TIM4_IRQn 0 */
  HAL_TIM_IRQHandler(&htim4);
  /* USER CODE BEGIN TIM4_IRQn 1 */

  /* USER CODE END TIM4_IRQn 1 */
}

/**
  * @brief This function handles USB On The Go FS End Point 1 Out global interrupt.
  */
void OTG_FS_EP1_OUT_IRQHandler(void)
{
  /* USER CODE BEGIN OTG_FS_EP1_OUT_IRQn 0 */

  /* USER CODE END OTG_FS_EP1_OUT_IRQn 0 */
  HAL_PCD_IRQHandler(&hpcd_USB_OTG_FS);
  /* USER CODE BEGIN OTG_FS_EP1_OUT_IRQn 1 */

  /* USER CODE END OTG_FS_EP1_OUT_IRQn 1 */
}

/**
  * @brief This function handles USB On The Go FS End Point 1 In global interrupt.
  */
void OTG_FS_EP1_IN_IRQHandler(void)
{
  /* USER CODE BEGIN OTG_FS_EP1_IN_IRQn 0 */

  /* USER CODE END OTG_FS_EP1_IN_IRQn 0 */
```

```
  HAL_PCD_IRQHandler(&hpcd_USB_OTG_FS);
  /* USER CODE BEGIN OTG_FS_EP1_IN_IRQn 1 */

  /* USER CODE END OTG_FS_EP1_IN_IRQn 1 */
}

/**
  * @brief This function handles USB On The Go FS global interrupt.
  */
void OTG_FS_IRQHandler(void)
{
  /* USER CODE BEGIN OTG_FS_IRQn 0 */

  /* USER CODE END OTG_FS_IRQn 0 */
  HAL_PCD_IRQHandler(&hpcd_USB_OTG_FS);
  /* USER CODE BEGIN OTG_FS_IRQn 1 */

  /* USER CODE END OTG_FS_IRQn 1 */
}

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

```c
/**
  ******************************************************************************
  * @file           : usbd_cdc_if.c
  * @version        : v1.0_Cube
  * @brief          : Usb device for Virtual Com Port.
  ******************************************************************************
  * This notice applies to any and all portions of this file
  * that are not between comment pairs USER CODE BEGIN and
  * USER CODE END. Other portions of this file, whether
  * inserted by the user or by software development tools
  * are owned by their respective copyright owners.
  *
  * Copyright (c) 2019 STMicroelectronics International N.V.
  * All rights reserved.
  *
  * Redistribution and use in source and binary forms, with or without
  * modification, are permitted, provided that the following conditions are met:
  *
  * 1. Redistribution of source code must retain the above copyright notice,
  *    this list of conditions and the following disclaimer.
  * 2. Redistributions in binary form must reproduce the above copyright notice,
  *    this list of conditions and the following disclaimer in the documentation
  *    and/or other materials provided with the distribution.
  * 3. Neither the name of STMicroelectronics nor the names of other
  *    contributors to this software may be used to endorse or promote products
  *    derived from this software without specific written permission.
  * 4. This software, including modifications and/or derivative works of this
  *    software, must execute solely and exclusively on microcontroller or
  *    microprocessor devices manufactured by or for STMicroelectronics.
  * 5. Redistribution and use of this software other than as permitted under
  *    this license is void and will automatically terminate your rights under
  *    this license.
  *
  * THIS SOFTWARE IS PROVIDED BY STMICROELECTRONICS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS, IMPLIED OR STATUTORY WARRANTIES, INCLUDING, BUT NOT
  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
  * PARTICULAR PURPOSE AND NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY
  * RIGHTS ARE DISCLAIMED TO THE FULLEST EXTENT PERMITTED BY LAW. IN NO EVENT
  * SHALL STMICROELECTRONICS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
  * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
  * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
  * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
  * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
  * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/
#include "usbd_cdc_if.h"

/* USER CODE BEGIN INCLUDE */

/* USER CODE END INCLUDE */

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Private macro -------------------------------------------------------------*/

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/

/* USER CODE END PV */

/** @addtogroup STM32_USB_OTG_DEVICE_LIBRARY
  * @brief Usb device library.
  * @{
  */

```

```
/** @addtogroup USBD_CDC_IF
  * @{
  */

/** @defgroup USBD_CDC_IF_Private_TypesDefinitions USBD_CDC_IF_Private_TypesDefinitions
  * @brief Private types.
  * @{
  */

/* USER CODE BEGIN PRIVATE_TYPES */

/* USER CODE END PRIVATE_TYPES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Private_Defines USBD_CDC_IF_Private_Defines
  * @brief Private defines.
  * @{
  */

/* USER CODE BEGIN PRIVATE_DEFINES */
/* Define size for the receive and transmit buffer over CDC */
/* It's up to user to redefine and/or remove those define */
#define APP_RX_DATA_SIZE  2048
#define APP_TX_DATA_SIZE  2048
/* USER CODE END PRIVATE_DEFINES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Private_Macros USBD_CDC_IF_Private_Macros
  * @brief Private macros.
  * @{
  */

/* USER CODE BEGIN PRIVATE_MACRO */

/* USER CODE END PRIVATE_MACRO */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Private_Variables USBD_CDC_IF_Private_Variables
  * @brief Private variables.
  * @{
  */
/* Create buffer for reception and transmission           */
/* It's up to user to redefine and/or remove those define */
/** Received data over USB are stored in this buffer      */
uint8_t UserRxBufferFS[APP_RX_DATA_SIZE];

/** Data to send over USB CDC are stored in this buffer   */
uint8_t UserTxBufferFS[APP_TX_DATA_SIZE];

/* USER CODE BEGIN PRIVATE_VARIABLES */

/* USER CODE END PRIVATE_VARIABLES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Exported_Variables USBD_CDC_IF_Exported_Variables
  * @brief Public variables.
  * @{
```

```
  */

extern USBD_HandleTypeDef hUsbDeviceFS;

/* USER CODE BEGIN EXPORTED_VARIABLES */

/* USER CODE END EXPORTED_VARIABLES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Private_FunctionPrototypes USBD_CDC_IF_Private_FunctionPrototypes
  * @brief Private functions declaration.
  * @{
  */

static int8_t CDC_Init_FS(void);
static int8_t CDC_DeInit_FS(void);
static int8_t CDC_Control_FS(uint8_t cmd, uint8_t* pbuf, uint16_t length);
static int8_t CDC_Receive_FS(uint8_t* pbuf, uint32_t *Len);

/* USER CODE BEGIN PRIVATE_FUNCTIONS_DECLARATION */

/* USER CODE END PRIVATE_FUNCTIONS_DECLARATION */

/**
  * @}
  */

USBD_CDC_ItfTypeDef USBD_Interface_fops_FS =
{
  CDC_Init_FS,
  CDC_DeInit_FS,
  CDC_Control_FS,
  CDC_Receive_FS
};

/* Private functions ---------------------------------------------------------*/
/**
  * @brief  Initializes the CDC media low layer over the FS USB IP
  * @retval USBD_OK if all operations are OK else USBD_FAIL
  */
static int8_t CDC_Init_FS(void)
{
  /* USER CODE BEGIN 3 */
  /* Set Application Buffers */
  USBD_CDC_SetTxBuffer(&hUsbDeviceFS, UserTxBufferFS, 0);
  USBD_CDC_SetRxBuffer(&hUsbDeviceFS, UserRxBufferFS);
  return (USBD_OK);
  /* USER CODE END 3 */
}

/**
  * @brief  DeInitializes the CDC media low layer
  * @retval USBD_OK if all operations are OK else USBD_FAIL
  */
static int8_t CDC_DeInit_FS(void)
{
  /* USER CODE BEGIN 4 */
  return (USBD_OK);
  /* USER CODE END 4 */
}

/**
  * @brief  Manage the CDC class requests
  * @param  cmd: Command code
  * @param  pbuf: Buffer containing command data (request parameters)
  * @param  length: Number of data to be sent (in bytes)
```

```
  * @retval Result of the operation: USBD_OK if all operations are OK else USBD_FAIL
  */
static int8_t CDC_Control_FS(uint8_t cmd, uint8_t* pbuf, uint16_t length)
{
  /* USER CODE BEGIN 5 */
  switch(cmd)
  {
    case CDC_SEND_ENCAPSULATED_COMMAND:

    break;

    case CDC_GET_ENCAPSULATED_RESPONSE:

    break;

    case CDC_SET_COMM_FEATURE:

    break;

    case CDC_GET_COMM_FEATURE:

    break;

    case CDC_CLEAR_COMM_FEATURE:

    break;

  /*******************************************************************************/
  /* Line Coding Structure                                                       */
  /*-----------------------------------------------------------------------------*/
  /* Offset | Field       | Size | Value  | Description                          */
  /* 0      | dwDTERate   |   4  | Number |Data terminal rate, in bits per second*/
  /* 4      | bCharFormat |   1  | Number | Stop bits                            */
  /*                                        0 - 1 Stop bit                       */
  /*                                        1 - 1.5 Stop bits                    */
  /*                                        2 - 2 Stop bits                      */
  /* 5      | bParityType |  1   | Number | Parity                               */
  /*                                        0 - None                             */
  /*                                        1 - Odd                              */
  /*                                        2 - Even                             */
  /*                                        3 - Mark                             */
  /*                                        4 - Space                            */
  /* 6      | bDataBits  |   1   | Number Data bits (5, 6, 7, 8 or 16).          */
  /*******************************************************************************/
    case CDC_SET_LINE_CODING:

    break;

    case CDC_GET_LINE_CODING:

    break;

    case CDC_SET_CONTROL_LINE_STATE:

    break;

    case CDC_SEND_BREAK:

    break;

  default:
    break;
  }

  return (USBD_OK);
  /* USER CODE END 5 */
}

/**
  * @brief  Data received over USB OUT endpoint are sent over CDC interface
```

```
  *         through this function.
  *
  *         @note
  *         This function will block any OUT packet reception on USB endpoint
  *         untill exiting this function. If you exit this function before transfer
  *         is complete on CDC interface (ie. using DMA controller) it will result
  *         in receiving more data while previous ones are still not sent.
  *
  * @param  Buf: Buffer of data to be received
  * @param  Len: Number of data received (in bytes)
  * @retval Result of the operation: USBD_OK if all operations are OK else USBD_FAIL
  */
static int8_t CDC_Receive_FS(uint8_t* Buf, uint32_t *Len)
{
  /* USER CODE BEGIN 6 */
  USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
  USBD_CDC_ReceivePacket(&hUsbDeviceFS);
  CDC_ReceiveCmpltCallback(Buf, Len[0]);
  return (USBD_OK);
  /* USER CODE END 6 */
}

/**
  * @brief  CDC_Transmit_FS
  *         Data to send over USB IN endpoint are sent over CDC interface
  *         through this function.
  *         @note
  *
  *
  * @param  Buf: Buffer of data to be sent
  * @param  Len: Number of data to be sent (in bytes)
  * @retval USBD_OK if all operations are OK else USBD_FAIL or USBD_BUSY
  */
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
{
  uint8_t result = USBD_OK;
  /* USER CODE BEGIN 7 */
  USBD_CDC_HandleTypeDef *hcdc = (USBD_CDC_HandleTypeDef*)hUsbDeviceFS.pClassData;
  if (hcdc->TxState != 0){
    return USBD_BUSY;
  }
  USBD_CDC_SetTxBuffer(&hUsbDeviceFS, Buf, Len);
  result = USBD_CDC_TransmitPacket(&hUsbDeviceFS);
  /* USER CODE END 7 */
  return result;
}

/* USER CODE BEGIN PRIVATE_FUNCTIONS_IMPLEMENTATION */
__weak void CDC_ReceiveCmpltCallback(uint8_t *buf, uint32_t len)
{
}
/* USER CODE END PRIVATE_FUNCTIONS_IMPLEMENTATION */

/**
  * @}
  */

/**
  * @}
  */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/

```

```c
/**
  ******************************************************************************
  * @file           : usbd_cdc_if.h
  * @version        : v1.0_Cube
  * @brief          : Header for usbd_cdc_if.c file.
  ******************************************************************************
  * This notice applies to any and all portions of this file
  * that are not between comment pairs USER CODE BEGIN and
  * USER CODE END. Other portions of this file, whether
  * inserted by the user or by software development tools
  * are owned by their respective copyright owners.
  *
  * Copyright (c) 2019 STMicroelectronics International N.V.
  * All rights reserved.
  *
  * Redistribution and use in source and binary forms, with or without
  * modification, are permitted, provided that the following conditions are met:
  *
  * 1. Redistribution of source code must retain the above copyright notice,
  *    this list of conditions and the following disclaimer.
  * 2. Redistributions in binary form must reproduce the above copyright notice,
  *    this list of conditions and the following disclaimer in the documentation
  *    and/or other materials provided with the distribution.
  * 3. Neither the name of STMicroelectronics nor the names of other
  *    contributors to this software may be used to endorse or promote products
  *    derived from this software without specific written permission.
  * 4. This software, including modifications and/or derivative works of this
  *    software, must execute solely and exclusively on microcontroller or
  *    microprocessor devices manufactured by or for STMicroelectronics.
  * 5. Redistribution and use of this software other than as permitted under
  *    this license is void and will automatically terminate your rights under
  *    this license.
  *
  * THIS SOFTWARE IS PROVIDED BY STMICROELECTRONICS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS, IMPLIED OR STATUTORY WARRANTIES, INCLUDING, BUT NOT
  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
  * PARTICULAR PURPOSE AND NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY
  * RIGHTS ARE DISCLAIMED TO THE FULLEST EXTENT PERMITTED BY LAW. IN NO EVENT
  * SHALL STMICROELECTRONICS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
  * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
  * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
  * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
  * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
  * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *
  ******************************************************************************
  */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __USBD_CDC_IF_H__
#define __USBD_CDC_IF_H__

#ifdef __cplusplus
 extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "usbd_cdc.h"

/* USER CODE BEGIN INCLUDE */

/* USER CODE END INCLUDE */

/** @addtogroup STM32_USB_OTG_DEVICE_LIBRARY
  * @brief For Usb device.
  * @{
  */

/** @defgroup USBD_CDC_IF USBD_CDC_IF
```

```
  * @brief Usb VCP device module
  * @{
  */

/** @defgroup USBD_CDC_IF_Exported_Defines USBD_CDC_IF_Exported_Defines
  * @brief Defines.
  * @{
  */
/* USER CODE BEGIN EXPORTED_DEFINES */

/* USER CODE END EXPORTED_DEFINES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Exported_Types USBD_CDC_IF_Exported_Types
  * @brief Types.
  * @{
  */

/* USER CODE BEGIN EXPORTED_TYPES */

/* USER CODE END EXPORTED_TYPES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Exported_Macros USBD_CDC_IF_Exported_Macros
  * @brief Aliases.
  * @{
  */

/* USER CODE BEGIN EXPORTED_MACRO */

/* USER CODE END EXPORTED_MACRO */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Exported_Variables USBD_CDC_IF_Exported_Variables
  * @brief Public variables.
  * @{
  */

/** CDC Interface callback. */
extern USBD_CDC_ItfTypeDef USBD_Interface_fops_FS;

/* USER CODE BEGIN EXPORTED_VARIABLES */

/* USER CODE END EXPORTED_VARIABLES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Exported_FunctionsPrototype
USBD_CDC_IF_Exported_FunctionsPrototype
  * @brief Public functions declaration.
  * @{
  */

uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len);

/* USER CODE BEGIN EXPORTED_FUNCTIONS */
__weak void CDC_ReceiveCmpltCallback(uint8_t *buf, uint32_t len);
/* USER CODE END EXPORTED_FUNCTIONS */

```

```
/**
  * @}
  */

/**
  * @}
  */

/**
  * @}
  */

#ifdef __cplusplus
}
#endif

#endif /* __USBD_CDC_IF_H__ */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

```c
/**
  ******************************************************************************
  * @file           : usb_device.c
  * @version        : v1.0_Cube
  * @brief          : This file implements the USB Device
  ******************************************************************************
  * This notice applies to any and all portions of this file
  * that are not between comment pairs USER CODE BEGIN and
  * USER CODE END. Other portions of this file, whether
  * inserted by the user or by software development tools
  * are owned by their respective copyright owners.
  *
  * Copyright (c) 2019 STMicroelectronics International N.V.
  * All rights reserved.
  *
  * Redistribution and use in source and binary forms, with or without
  * modification, are permitted, provided that the following conditions are met:
  *
  * 1. Redistribution of source code must retain the above copyright notice,
  *    this list of conditions and the following disclaimer.
  * 2. Redistributions in binary form must reproduce the above copyright notice,
  *    this list of conditions and the following disclaimer in the documentation
  *    and/or other materials provided with the distribution.
  * 3. Neither the name of STMicroelectronics nor the names of other
  *    contributors to this software may be used to endorse or promote products
  *    derived from this software without specific written permission.
  * 4. This software, including modifications and/or derivative works of this
  *    software, must execute solely and exclusively on microcontroller or
  *    microprocessor devices manufactured by or for STMicroelectronics.
  * 5. Redistribution and use of this software other than as permitted under
  *    this license is void and will automatically terminate your rights under
  *    this license.
  *
  * THIS SOFTWARE IS PROVIDED BY STMICROELECTRONICS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS, IMPLIED OR STATUTORY WARRANTIES, INCLUDING, BUT NOT
  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
  * PARTICULAR PURPOSE AND NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY
  * RIGHTS ARE DISCLAIMED TO THE FULLEST EXTENT PERMITTED BY LAW. IN NO EVENT
  * SHALL STMICROELECTRONICS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
  * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
  * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
  * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
  * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
  * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/

#include "usb_device.h"
#include "usbd_core.h"
#include "usbd_desc.h"
#include "usbd_cdc.h"
#include "usbd_cdc_if.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/

/* USER CODE END PV */

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/

/* USER CODE END PFP */
```

```
/* USB Device Core handle declaration. */
USBD_HandleTypeDef hUsbDeviceFS;

/*
 * -- Insert your variables declaration here --
 */
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/*
 * -- Insert your external function declaration here --
 */
/* USER CODE BEGIN 1 */

/* USER CODE END 1 */

/**
  * Init USB device Library, add supported class and start the library
  * @retval None
  */
void MX_USB_DEVICE_Init(void)
{
  /* USER CODE BEGIN USB_DEVICE_Init_PreTreatment */

  /* USER CODE END USB_DEVICE_Init_PreTreatment */

  /* Init Device Library, add supported class and start the library. */
  USBD_Init(&hUsbDeviceFS, &FS_Desc, DEVICE_FS);

  USBD_RegisterClass(&hUsbDeviceFS, &USBD_CDC);

  USBD_CDC_RegisterInterface(&hUsbDeviceFS, &USBD_Interface_fops_FS);

  USBD_Start(&hUsbDeviceFS);

  /* USER CODE BEGIN USB_DEVICE_Init_PostTreatment */
  HAL_PWREx_EnableUSBVoltageDetector();
  /* USER CODE END USB_DEVICE_Init_PostTreatment */
}

/**
  * @}
  */

/**
  * @}
  */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

```
////////////////////////////////////////////////////////////////////////////////
//7.6 Programming the FPGA
////////////////////////////////////////////////////////////////////////////////
// look in pins.pcf for all the pin names on the TinyFPGA BX board
module ClkGenerator (
  input CLK, // 16MHz clock
  input INT10ms,
  input INT100ms,
  input INT200ms,
  input INT500ms,
  input INT1000ms,
  input INT1500ms,
  input INT2000ms,
  output CLK_2M, //2MHz clock
  output SH,
  output ICG,
  output ADC_start, //500KHz clock
  output USBPU  // USB pull-up resistor
  );

  // drive USB pull-up resistor to '0' to disable USB
  assign USBPU = 0;

  reg[31:0] counter = 32'd0;
  reg[31:0] counter1 = 32'd0;
  reg[31:0] counter2 = 32'd0;
  reg[31:0] counter12 = 32'd0;
  reg[31:0] counter3 = 32'd0;
  reg[31:0] counter13 = 32'd0;
  reg[31:0] counter4 = 32'd0;
  reg[31:0] counter14 = 32'd0;
  reg[31:0] counter5 = 32'd0;
  reg[31:0] counter15 = 32'd0;
  reg[31:0] counter6 = 32'd0;
  reg[31:0] counter16 = 32'd0;
  reg[31:0] counter7 = 32'd0;
  reg[31:0] counter17 = 32'd0;
  reg[2:0] counter2M;
  reg tempSH = 1'b0;
  reg tempICG = 1'b0;
  reg tempSH2 = 1'b0;
  reg tempICG2 = 1'b0;
  reg tempSH3 = 1'b0;
  reg tempICG3 = 1'b0;
  reg tempSH4 = 1'b0;
  reg tempICG4 = 1'b0;
  reg tempSH5 = 1'b0;
  reg tempICG5 = 1'b0;
  reg tempSH6 = 1'b0;
  reg tempICG6 = 1'b0;
  reg tempSH7 = 1'b0;
  reg tempICG7 = 1'b0;
  reg temp2M = 1'b0;
  reg tempADC = 1'b0;
  reg tempADC2 = 1'b0;
  reg tempADC3 = 1'b0;
  reg tempADC4 = 1'b0;
  reg tempADC5 = 1'b0;
  reg tempADC6 = 1'b0;
  reg tempADC7 = 1'b0;
  wire enable;

  assign enable = INT10ms | INT100ms | INT200ms | INT500ms | INT1000ms | INT1500ms | INT2000ms;

  always @(posedge CLK or negedge enable)
  begin
    if(enable == 1'b0)
      temp2M <= 1'b0;
    else
```

```
    begin
      if (counter2M == 3'd7) // period, count from 0 to n-1
        counter2M <= 0;
      else
        counter2M <= counter2M + 1'b1;
      if (counter2M < 3'd4) // duty cycle, m cycles high, 50% if n/2
        temp2M <= 1'b1;
      else
        temp2M <= 1'b0;
    end
  end

/* The different integration time configurations
are done with electronic shutter mode ON and
keeping ICG=n*SH + 1, n being 2. User can change the
value of n by modifying the counter values indicated
by ICG and SH in comments.
*/
  always @(posedge CLK_2M or negedge INT10ms)
  begin
    if(INT10ms == 1'b0)
    begin
    counter <= 32'd0;
    counter1 <= 32'd0;
    end
    else
    begin
      if (counter == 32'd40001) //ICG
        counter <= 0;
      else
        counter <= counter + 1;
      if (counter < 32'd15)
        tempICG <= 1'b0;
      else
        tempICG <= 1'b1;
      if (counter1 == 32'd20000) //SH
        counter1 <= 0;
      else
        counter1 <= counter1 + 1;
      if (counter1 > 32'd0 && counter1 < 32'd5)
        tempSH <= 1'b1;
      else
        tempSH <= 1'b0;
    end
  end

  always @(posedge CLK_2M or negedge INT100ms)
  begin
    if(INT100ms == 1'b0)
    begin
      counter2 <= 32'd0;
      counter12 <= 32'd0;
    end
    else
    begin
      if (counter2 == 32'd400001) //ICG
        counter2 <= 0;
      else
        counter2 <= counter2 + 1;
      if (counter2 < 32'd15)
        tempICG2 <= 1'b0;
      else
        tempICG2 <= 1'b1;
      if (counter12 == 32'd200000) //SH
        counter12 <= 0;
      else
        counter12 <= counter12 + 1;
      if (counter12 > 32'd0 && counter12 < 32'd5)
        tempSH2 <= 1'b1;
      else
```

```
        tempSH2 <= 1'b0;
    end
  end

  always @(posedge CLK_2M or negedge INT200ms)
  begin
    if(INT200ms == 1'b0)
    begin
      counter3 <= 32'd0;
      counter13 <= 32'd0;
    end
    else
    begin
      if (counter3 == 32'd800001) //ICG
        counter3 <= 0;
      else
        counter3 <= counter3 + 1;
      if (counter3 < 32'd15)
        tempICG3 <= 1'b0;
      else
        tempICG3 <= 1'b1;
      if (counter13 == 32'd400000) //SH
        counter13 <= 0;
      else
        counter13 <= counter13 + 1;
      if (counter13 > 32'd0 && counter13 < 32'd5)
        tempSH3 <= 1'b1;
      else
        tempSH3 <= 1'b0;
    end
  end

  always @(posedge CLK_2M or negedge INT500ms)
  begin
    if(INT500ms == 1'b0)
    begin
      counter4 <= 32'd0;
      counter14 <= 32'd0;
    end
    else
    begin
      if (counter4 == 32'd2000001) //ICG
        counter4 <= 0;
      else
        counter4 <= counter4 + 1;
      if (counter4 < 32'd15)
        tempICG4 <= 1'b0;
      else
        tempICG4 <= 1'b1;
      if (counter14 == 32'd1000000) //SH
        counter14 <= 0;
      else
        counter14 <= counter14 + 1;
      if (counter14 > 32'd0 && counter14 < 32'd5)
        tempSH4 <= 1'b1;
      else
        tempSH4 <= 1'b0;
    end
  end

  always @(posedge CLK_2M or negedge INT1000ms)
  begin
    if(INT1000ms == 1'b0)
    begin
      counter5 <= 32'd0;
      counter15 <= 32'd0;
    end
    else
    begin
      if (counter5 == 32'd4000001) //ICG
```

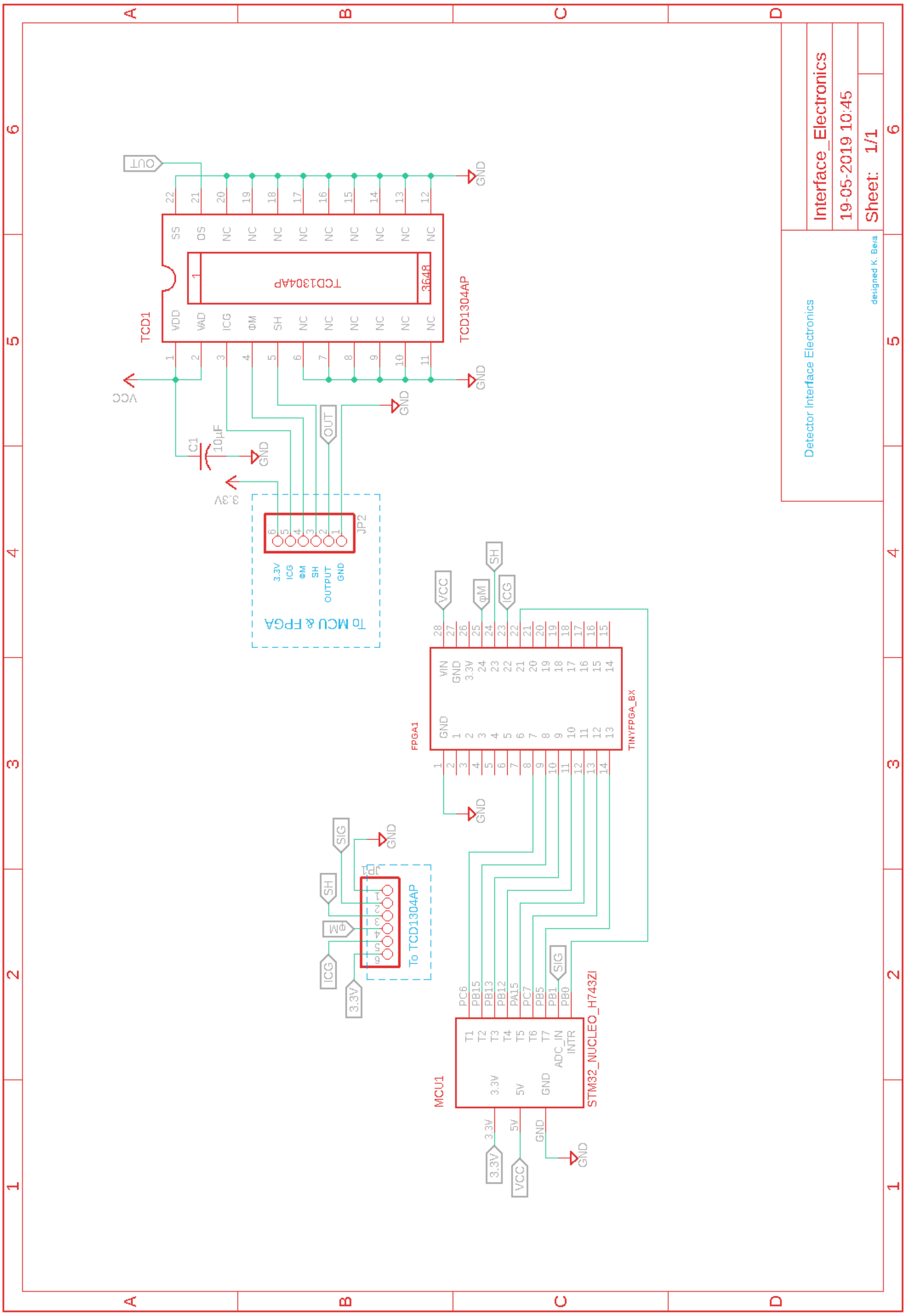
```
      counter5 <= 0;
    else
      counter5 <= counter5 + 1;
    if (counter5 < 32'd15)
      tempICG5 <= 1'b0;
    else
      tempICG5 <= 1'b1;
    if (counter15 == 32'd2000000) //SH
      counter15 <= 0;
    else
      counter15 <= counter15 + 1;
    if (counter15 > 32'd0 && counter15 < 32'd5)
      tempSH5 <= 1'b1;
    else
      tempSH5 <= 1'b0;
  end
end

always @(posedge CLK_2M or negedge INT1500ms)
begin
  if(INT1500ms == 1'b0)
  begin
    counter6 <= 32'd0;
    counter16 <= 32'd0;
  end
  else
  begin
    if (counter6 == 32'd4000001) //ICG
      counter6 <= 0;
    else
      counter6 <= counter6 + 1;
    if (counter6 < 32'd15)
      tempICG6 <= 1'b0;
    else
      tempICG6 <= 1'b1;
    if (counter16 == 32'd2000000) //SH
      counter16 <= 0;
    else
      counter16 <= counter16 + 1;
    if (counter16 > 32'd0 && counter16 < 32'd5)
      tempSH6 <= 1'b1;
    else
      tempSH6 <= 1'b0;
  end
end

always @(posedge CLK_2M or negedge INT2000ms)
begin
  if(INT2000ms == 1'b0)
  begin
    counter7 <= 32'd0;
    counter17 <= 32'd0;
  end
  else
  begin
    if (counter7 == 32'd8000001) //ICG
      counter7 <= 0;
    else
      counter7 <= counter7 + 1;
    if (counter7 < 32'd15)
      tempICG7 <= 1'b0;
    else
      tempICG7 <= 1'b1;
    if (counter17 == 32'd4000000) //SH
      counter17 <= 0;
    else
      counter17 <= counter17 + 1;
    if (counter17 > 32'd0 && counter17 < 32'd5)
      tempSH7 <= 1'b1;
    else
```

```
        tempSH7 <= 1'b0;
    end
  end

  always @(posedge CLK)
  begin
    if(counter == 32'd16) tempADC <= 1'b1;
    else tempADC <= 1'b0;
    if(counter2 == 32'd16) tempADC2 <= 1'b1;
    else tempADC2 <= 1'b0;
    if(counter3 == 32'd16) tempADC3 <= 1'b1;
    else tempADC3 <= 1'b0;
    if(counter4 == 32'd16) tempADC4 <= 1'b1;
    else tempADC4 <= 1'b0;
    if(counter5 == 32'd16) tempADC5 <= 1'b1;
    else tempADC5 <= 1'b0;
    if(counter6 == 32'd16) tempADC6 <= 1'b1;
    else tempADC6 <= 1'b0;
    if(counter7 == 32'd16) tempADC7 <= 1'b1;
    else tempADC7 <= 1'b0;
  end

  assign CLK_2M = temp2M;
  assign ADC_start = tempADC | tempADC2 | tempADC3 | tempADC4 | tempADC5 | tempADC6 | tempADC7;
  assign SH = tempSH | tempSH2 | tempSH3 | tempSH4 | tempSH5 | tempSH6 | tempSH7;
  assign ICG = tempICG | tempICG2 | tempICG3 | tempICG4 | tempICG5 | tempICG6 | tempICG7;
  endmodule
```

```
################################################################################
#
# TinyFPGA BX constraint file (.pcf)
#
################################################################################
#
# Copyright (c) 2018, Luke Valenty
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#
# 1. Redistributions of source code must retain the above copyright notice, this
#    list of conditions and the following disclaimer.
# 2. Redistributions in binary form must reproduce the above copyright notice,
#    this list of conditions and the following disclaimer in the documentation
#    and/or other materials provided with the distribution.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
# ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
# WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
# ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
# (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
# LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
# ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
# (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
# SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
# The views and conclusions contained in the software and documentation are those
# of the authors and should not be interpreted as representing official policies,
# either expressed or implied, of the <project name> project.
#
################################################################################

####
# TinyFPGA BX information: https://github.com/tinyfpga/TinyFPGA-BX/
####

# Left side of board
set_io --warn-no-port PIN_1 A2
set_io --warn-no-port PIN_2 A1
set_io --warn-no-port PIN_3 B1
set_io --warn-no-port PIN_4 C2
set_io --warn-no-port PIN_5 C1
set_io --warn-no-port PIN_6 D2
set_io --warn-no-port INT10ms D1
set_io --warn-no-port INT100ms E2
set_io --warn-no-port INT200ms E1
set_io --warn-no-port INT500ms G2
set_io --warn-no-port INT1000ms H1
set_io --warn-no-port INT1500ms J1
set_io --warn-no-port INT2000ms H2

# Right side of board
set_io --warn-no-port PIN_14 H9
set_io --warn-no-port PIN_15 D9
set_io --warn-no-port PIN_16 D8
set_io --warn-no-port PIN_17 C9
set_io --warn-no-port PIN_18 A9
set_io --warn-no-port PIN_19 B8
set_io --warn-no-port PIN_20 A8
set_io --warn-no-port ADC_start B7
set_io --warn-no-port ICG A7
set_io --warn-no-port SH B6
set_io --warn-no-port CLK_2M A6

# SPI flash interface on bottom of board
set_io --warn-no-port SPI_SS F7
set_io --warn-no-port SPI_SCK G7
```

```
set_io --warn-no-port SPI_IO0 G6
set_io --warn-no-port SPI_IO1 H7
set_io --warn-no-port SPI_IO2 H4
set_io --warn-no-port SPI_IO3 J8

# General purpose pins on bottom of board
set_io --warn-no-port PIN_25 G1
set_io --warn-no-port PIN_26 J3
set_io --warn-no-port PIN_27 J4
set_io --warn-no-port PIN_28 G9
set_io --warn-no-port PIN_29 J9
set_io --warn-no-port PIN_30 E8
set_io --warn-no-port PIN_31 J2

# LED
set_io --warn-no-port LED B3

# USB
set_io --warn-no-port USBP B4
set_io --warn-no-port USBN A4
set_io --warn-no-port USBPU A3

# 16MHz clock
set_io --warn-no-port CLK B2 # input

```

7.7. Schematic of the detector interface electronics



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 7.8. Designing the GUI for Raman spectra acquisition
%%      in App Designer byMATLAB
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
classdef RamanUIv1 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        RamanUI                         matlab.ui.Figure
        RamanSpectrometerv10Label       matlab.ui.control.Label
        DarkButton                      matlab.ui.control.Button
        LightButton                     matlab.ui.control.Button
        Button                          matlab.ui.control.Button
        TabGroup                        matlab.ui.container.TabGroup
        SettingsTab                     matlab.ui.container.Tab
        ConnectionsettingsPanel         matlab.ui.container.Panel
        COMportLabel                    matlab.ui.control.Label
        COMportDropDown                 matlab.ui.control.DropDown
        DevicenameLabel                 matlab.ui.control.Label
        ConnDeviceName                  matlab.ui.control.EditField
        ConnectButton                   matlab.ui.control.Button
        ConnStatusLamp                  matlab.ui.control.Lamp
        ConnStatusMsg                   matlab.ui.control.Label
        DisconnectButton                matlab.ui.control.Button
        ParametersPanel                 matlab.ui.container.Panel
        SampleaverageEditFieldLabel     matlab.ui.control.Label
        SampleAverageEditField          matlab.ui.control.NumericEditField
        IntegrationtimeinmsLabel        matlab.ui.control.Label
        IntegrationtimeDropDown         matlab.ui.control.DropDown
        ArithmeticTab                   matlab.ui.container.Tab
        FilteringalgorithmButtonGroup   matlab.ui.container.ButtonGroup
        EMAradioButton                  matlab.ui.control.RadioButton
        SavitzkyGolayButton             matlab.ui.control.RadioButton
        SmoothSlider                    matlab.ui.control.Slider
        SmoothnessfactorLabel           matlab.ui.control.Label
        OrderSpinnerLabel               matlab.ui.control.Label
        OrderSpinner                    matlab.ui.control.Spinner
        FramelengthLabel                matlab.ui.control.Label
        FramelengthSpinner              matlab.ui.control.Spinner
        GraphsettingsTab                matlab.ui.container.Tab
        XmaxEditFieldLabel              matlab.ui.control.Label
        XmaxEditField                   matlab.ui.control.NumericEditField
        XminEditFieldLabel              matlab.ui.control.Label
        XminEditField                   matlab.ui.control.NumericEditField
        YmaxEditFieldLabel              matlab.ui.control.Label
        YmaxEditField                   matlab.ui.control.NumericEditField
        YminEditFieldLabel              matlab.ui.control.Label
        YminEditField                   matlab.ui.control.NumericEditField
        XlimAuto                        matlab.ui.control.Button
        YlimAuto                        matlab.ui.control.Button
        ExportTab                       matlab.ui.container.Tab
        SelectdataButtonGroup           matlab.ui.container.ButtonGroup
        DarkdataButton                  matlab.ui.control.RadioButton
        LightdataButton                 matlab.ui.control.RadioButton
        ReferencedataButton             matlab.ui.control.RadioButton
        SelectfileextensionButtonGroup  matlab.ui.container.ButtonGroup
        xlsxButton                      matlab.ui.control.RadioButton
        matButton                       matlab.ui.control.RadioButton
        SaveButton                      matlab.ui.control.Button
        UIAxes                          matlab.ui.control.UIAxes
        MeasureButton                   matlab.ui.control.Button
        MonitorButton                   matlab.ui.control.Button
        SpectraCountEditField           matlab.ui.control.NumericEditField
        ReferenceButton                 matlab.ui.control.Button
        DarkLamp                        matlab.ui.control.Lamp
        LightLamp                       matlab.ui.control.Lamp
        ReferenceLamp                   matlab.ui.control.Lamp
        MonitorLamp                     matlab.ui.control.Lamp
        MeasureLamp                     matlab.ui.control.Lamp
        SaveplotButton                  matlab.ui.control.Button
```

```
    end


    properties (Access = public)
        ComPorts;
        ComDevs;
        NoFriendlyPortNames_flag = 0;
        SelectedPort;
        SampleAverage = 1;
        SerialPortObj;
        MonitorFlag = 0;
        MeasureFlag = 0;
        SmoothnessFactor = 1;
        ActiveConn_flag = 0;
        CurrData = 0;
        DarkData = 0;
        LightData;
        RefData;
        DarkDataSet_flag = 0;
        LightDataSet_flag = 0;
        RefDataSet_flag = 0;
        ResultData;
        Disconnected_flag = 0;
        StartCode = 253;
        EndCode = 252;
        InitiateCode = 241;
        SelectedIntegTime = '100';
        SGorder = 3;
        SGframeLen = 7;
        UpdatedData;
        RawData;
        FolderFlag=0;
        DataAvailableFlag = 0;
        userProfile;
        myDocsFolder;
        myFolder = 'RamanSpectrometer_v1';
        MinADCcount = 0;
        x = 1:3648;
        minY = 10000;
        maxY = 50000;
        minX = 0;
        maxX = 4000;
        AllData ;
    end


    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            instrreset;
            screenSize = get(groot,'ScreenSize');
            screenWidth = screenSize(3);
            screenHeight = screenSize(4);
            left = screenWidth*0.15;
            bottom = screenHeight*0.15;
            width = screenWidth*0.6;
            height = screenHeight*0.6;
            drawnow;
            app.RamanUI.Position = [left bottom width height];
            app.UIAxes.YDir = 'reverse';
            %% Find the serial port %%------------------------------------------
--
            Skey = 'HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\SERIALCOMM';
            % Find connected serial devices and clean up the output
            [~, list] = dos(['REG QUERY ' Skey]);
            list = strread(list,'%s','delimiter',' ');
            coms = 0;
            for i = 1:numel(list)
                if strcmp(list{i}(1:3),'COM')
```

```
                    if ~iscell(coms)
                        coms = list(i);
                    else
                        coms{end+1} = list{i};
                    end
                end
            end
            key = 'HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\';
            % Find all installed USB devices entries and clean up the output
            [~, vals] = dos(['REG QUERY ' key ' /s /f "FriendlyName" /t "REG_SZ"']);
            vals = textscan(vals,'%s','delimiter','\t');
            vals = cat(1,vals{:});
            out = 0;
            % Find all friendly name property entries
            for i = 1:numel(vals)
                if strcmp(vals{i}(1:min(12,end)),'FriendlyName')
                    if ~iscell(out)
                        out = vals(i);
                    else
                        out{end+1} = vals{i};
                    end
                end
            end
            % Compare friendly name entries with connected ports and generate output
            for i = 1:numel(coms)
                match = strfind(out,[coms{i},')']);
                ind = 0;
                for j = 1:numel(match)
                    if ~isempty(match{j})
                        ind = j;
                    end
                end
                if (ind == 0)
                    app.NoFriendlyPortNames_flag = 1;
                end
                if ind ~= 0
                    com = str2double(coms{i}(4:end));
                    % Trim the trailing ' (COM##)' from the friendly name - works on ports from 1 to 99
                    if com > 9
                        length = 8;
                    else
                        length = 7;
                    end
                    devs{i,1} = out{ind}(27:end-length);
                    devs{i,2} = com;
                    app.NoFriendlyPortNames_flag = 0;
                end
            end
            app.COMportDropDown.Items = seriallist;
            app.ComPorts = coms;
            if(app.NoFriendlyPortNames_flag == 0)
                app.ComDevs = devs;
            end
            app.SampleAverage = app.SampleAverageEditField.Value;
            app.SmoothnessFactor = app.SmoothSlider.Value;
            app.SelectedPort = cell2mat(coms(1));
            app.IntegrationtimeDropDown.Value = '100';
            app.userProfile = getenv('USERPROFILE');
            app.myDocsFolder = sprintf('%s\\Desktop', app.userProfile);
            [app.FolderFlag,~,~] = mkdir(app.myDocsFolder,app.myFolder);
        end

        % Value changed function: COMportDropDown
        function COMportDropDownValueChanged(app, event)
            value = app.COMportDropDown.Value;
            if(app.NoFriendlyPortNames_flag == 0)
                if(~isempty(app.ComDevs{find(strcmp(app.ComPorts,value)),1}))
```

```
                    app.ConnDeviceName.Value = cell2mat(app.ComDevs(strcmp(app.ComPorts,value)));
                else
                    app.ConnDeviceName.Value = 'No friendly port name(s)';
                end
            elseif (app.NoFriendlyPortNames_flag == 1)
                app.ConnDeviceName.Value = 'No friendly port names';
            end
            app.SelectedPort = value;
        end

        % Button pushed function: ConnectButton
        function ConnectButtonPushed(app, event)
            if(app.ActiveConn_flag == 0)
                d = uiprogressdlg(app.RamanUI,'Title','Connecting.. Please wait..','Indeterminate','on');
                serialPort = serial(app.SelectedPort);
                app.SerialPortObj = serialPort;
                fopen(serialPort);
                fwrite(serialPort, app.InitiateCode, 'uint8');
                checkConn = fread(serialPort, 1, 'uint8');
                if(checkConn == app.InitiateCode)
                    app.ConnStatusLamp.Color = 'green';
                    app.ConnStatusMsg.Text = 'Connected';
                    app.ActiveConn_flag = 1;
                else
                    app.ConnStatusLamp.Color = 'red';
                    app.ConnStatusMsg.Text = 'Connection failed';
                    instrreset;
                    app.ActiveConn_flag = 0;
                end
                pause(0.5);
                close(d);
            else
                uialert(app.RamanUI,'Connection is already established','Warning', 'Icon', 'warning');
            end
        end

        % Button pushed function: MeasureButton
        function MeasureButtonPushed(app, event)
            if(app.ActiveConn_flag == 0)
                uialert(app.RamanUI,'Please connect your device first!','Warning', 'Icon', 'warning');
            else
                app.MeasureLamp.Color = [0.93,0.69,0.13];
                app.MonitorFlag = 0;
                app.MeasureFlag = 1;
                pause(0.00001);
                SpectraCount = 0;
                serialPort = app.SerialPortObj;
                fwrite(serialPort, app.StartCode, 'uint8');
                app.AllData=zeros(1,3694);
                sum=0;
                flushinput(serialPort);
                avg = app.SampleAverage+1;
                for j=1:avg
                    for i=1:3694
                        app.AllData(j,i)=fread(serialPort,1,'uint16');
                    end
                end
                fwrite(serialPort, app.EndCode, 'uint8');
                flushinput(serialPort);
                avgData = mean(app.AllData((2:avg),(33:3680)),1);
                app.DataAvailableFlag = 1;
                if(app.EMAradioButton.Value == true)
                    alpha = app.SmoothnessFactor;
                    filt_data = filtfilt(alpha,[1,-(1-alpha)],avgData);
                elseif(app.SavitzkyGolayButton.Value == true)
                    filt_data = sgolayfilt(avgData,app.SGorder,app.SGframeLen);
```

```
                end
                if(app.DarkDataSet_flag == 0 && app.RefDataSet_flag == 0)
                    plot(app.UIAxes,app.x, filt_data);
                elseif(app.DarkDataSet_flag == 1 && app.RefDataSet_flag == 0)
                    for i=1:3648
                        app.ResultData(i) = filt_data(i) - app.DarkData(i);
                    end
                    plot(app.UIAxes,app.x, app.ResultData);
                elseif(app.DarkDataSet_flag == 0 && app.RefDataSet_flag == 1)
                    for i=1:3648
                        app.ResultData(i) = filt_data(i) - app.RefData(i);
                    end
                    plot(app.UIAxes,app.x, app.ResultData);
                elseif(app.DarkDataSet_flag == 1 && app.RefDataSet_flag == 1)
                    for i=1:3648
                        app.ResultData(i) = filt_data(i) - app.DarkData(i) - app.RefData(i);
                    end
                    plot(app.UIAxes,app.x, app.ResultData);
                end
                app.MeasureLamp.Color = [0.9 0.9 0.9];
                app.CurrData = filt_data;
                app.RawData = avgData;
                app.MeasureFlag = 0;
            end
        end

        % Value changed function: SampleAverageEditField
        function SampleAverageEditFieldValueChanged(app, event)
            value = app.SampleAverageEditField.Value;
            app.SampleAverage = value;
        end

        % Close request function: RamanUI
        function RamanUICloseRequest(app, event)
            if(app.ActiveConn_flag == 1 && app.Disconnected_flag == 0)
                serialPort = app.SerialPortObj;
                fclose(serialPort);
                instrreset;
            else
                instrreset;
            end
            delete(app)
        end

        % Button pushed function: MonitorButton
        function MonitorButtonPushed(app, event)
            app.SpectraCountEditField.Value = 0;
            if(app.ActiveConn_flag == 0)
                uialert(app.RamanUI,'Please connect your device first!','Warning', 'Icon', 'warning');
            else
                app.MonitorLamp.Color = [0.93,0.69,0.13];
                serialPort = app.SerialPortObj;
                if(app.MonitorFlag == 0)
                    app.MonitorFlag = 1;
                    SpectraCount = 0;
                elseif(app.MonitorFlag == 1)
                    app.MonitorFlag = 0;
                end
                pause(0.00001);
                while (app.MonitorFlag == 1)
                    SpectraCount = SpectraCount + 1;
                    app.SpectraCountEditField.Value = SpectraCount;
                    fwrite(serialPort, app.StartCode, 'uint8');
                    app.AllData=zeros(1,3694);
                    sum=0;
                    flushinput(serialPort);
                    avg = app.SampleAverage+1;
                    for j=1:avg
```

```
                        for i=1:3694
                            app.AllData(j,i)=fread(serialPort,1,'uint16');
                        end
                    end
                    fwrite(serialPort, app.EndCode, 'uint8');
                    flushinput(serialPort);
                    avgData = mean(app.AllData((2:avg),(33:3680)),1);
                    app.DataAvailableFlag = 1;
                    app.RawData = avgData;
                    if(app.EMAradioButton.Value == true)
                        alpha = app.SmoothnessFactor;
                        filt_data = filtfilt(alpha,[1,-(1-alpha)],avgData);
                    elseif(app.SavitzkyGolayButton.Value == true)
                        filt_data = sgolayfilt(avgData,app.SGorder,app.SGframeLen);
                    end
                    if(app.DarkDataSet_flag == 0 && app.RefDataSet_flag == 0)
                        plot(app.UIAxes,app.x, filt_data);
                    elseif(app.DarkDataSet_flag == 1 && app.RefDataSet_flag == 0)
                        for i=1:3648
                            app.ResultData(i) = filt_data(i) - app.DarkData(i);
                        end
                        plot(app.UIAxes,app.x, app.ResultData);
                    elseif(app.DarkDataSet_flag == 0 && app.RefDataSet_flag == 1)
                        for i=1:3648
                            app.ResultData(i) = filt_data(i) - app.RefData(i);
                        end
                        plot(app.UIAxes,app.x, app.ResultData);
                    elseif(app.DarkDataSet_flag == 1 && app.RefDataSet_flag == 1)
                        for i=1:3648
                            app.ResultData(i) = filt_data(i) - app.DarkData(i) - app.RefData(i);
                        end
                        plot(app.UIAxes,app.x, app.ResultData);
                    end
                    app.CurrData = filt_data;
                    drawnow;
                    if(app.MonitorFlag == 0)
                        break;
                    end
                end
                app.MonitorLamp.Color = [0.9, 0.9, 0.9];
                pause(0.0001); %must be in conjuntion with integration time
            end
        end

        % Button pushed function: DisconnectButton
        function DisconnectButtonPushed(app, event)
            if(app.ActiveConn_flag == 1)
                serialPort = app.SerialPortObj;
                fclose(serialPort);
                instrreset;
                app.Disconnected_flag = 1;
                app.ActiveConn_flag = 0;
                app.ConnStatusLamp.Color = [0.9 0.9 0.9];
                app.ConnStatusMsg.Text = '<not connected>';
            else
                uialert(app.RamanUI,'No active connection','Warning', 'Icon', 'warning');
                pause(0.5);
            end
        end

        % Button pushed function: DarkButton
        function DarkButtonPushed(app, event)
            if(app.ActiveConn_flag == 0)
                uialert(app.RamanUI,'Please connect your device first!','Warning', 'Icon', 'warning');
            else
                app.DarkData = app.CurrData;
                if(app.DarkDataSet_flag == 0)
```

```
                    app.DarkDataSet_flag = 1;
                    app.DarkLamp.Color = [0.07,0.62,1.00];
                else
                    app.DarkDataSet_flag = 0;
                    app.DarkLamp.Color = [0.9,0.9,0.9];
                end
            end
        end

        % Value changed function: SmoothSlider
        function SmoothSliderValueChanged(app, event)
            value = app.SmoothSlider.Value;
            app.SmoothnessFactor = value;
            if(app.EMAradioButton.Value)
                if((app.MonitorFlag == 0 || app.MeasureFlag == 0) && app.DataAvailableFlag == 1)
                    app.UpdatedData = filtfilt(value,[1,-(1-value)],app.RawData);
                    plot(app.UIAxes,app.x, app.UpdatedData);
                end
            end
            app.CurrData = app.UpdatedData;
        end

        % Value changed function: IntegrationtimeDropDown
        function IntegrationtimeDropDownValueChanged(app, event)
            value = app.IntegrationtimeDropDown.Value;
            app.SelectedIntegTime = value;
            if strcmp(value,'10')
                app.StartCode = 255;
                app.EndCode = 254;
            elseif strcmp(value,'100')
                app.StartCode = 253;
                app.EndCode = 252;
            elseif strcmp(value,'200')
                app.StartCode = 251;
                app.EndCode = 250;
            elseif strcmp(value,'500')
                app.StartCode = 249;
                app.EndCode = 248;
            elseif strcmp(value,'1000')
                app.StartCode = 247;
                app.EndCode = 246;
            elseif strcmp(value,'1500')
                app.StartCode = 245;
                app.EndCode = 244;
            elseif strcmp(value,'2000')
                app.StartCode = 243;
                app.EndCode = 242;
            end
        end

        % Value changed function: OrderSpinner
        function OrderSpinnerValueChanged(app, event)
            value = app.OrderSpinner.Value;
            app.SGorder = value;
            if(app.SavitzkyGolayButton.Value)
                if(app.MonitorFlag == 0 || app.MeasureFlag == 0)
                    if(app.SGframeLen  > app.SGorder)
                        app.UpdatedData = sgolayfilt(app.RawData,value,app.SGframeLen);
                        plot(app.UIAxes,app.x, app.UpdatedData);
                        app.CurrData = app.UpdatedData;
                    else
                        uialert(app.RamanUI,'Frame length must be greater than Order','Warning', 'Icon', 'warning');
                    end
                end
            end
        end

```

```
        % Value changed function: FramelengthSpinner
        function FramelengthSpinnerValueChanged(app, event)
            value = app.FramelengthSpinner.Value;
            app.SGframeLen = value;
            if(app.SavitzkyGolayButton.Value)
                if(app.MonitorFlag == 0 || app.MeasureFlag == 0)
                    if(app.SGframeLen  > app.SGorder)
                        app.UpdatedData =
sgolayfilt(app.RawData,app.SGorder,value);
                        plot(app.UIAxes,app.x, app.UpdatedData);
                        app.CurrData = app.UpdatedData;
                    else
                        uialert(app.RamanUI,'Frame length must be greater than
Order','Warning', 'Icon', 'warning');
                    end
                end
            end
        end

        % Button pushed function: SaveButton
        function SaveButtonPushed(app, event)
            if(app.DarkdataButton.Value)
                if(app.xlsxButton.Value)
                    filename =
strcat(app.myDocsFolder,'\',app.myFolder,'\',datestr(now,30),'_darkSpectra','.xlsx'
);
                    ADCvalue = app.DarkData';
                    Pixel = app.x';
                    T=table(Pixel,ADCvalue);
                    if(app.FolderFlag)
                        writetable(T,filename);
                        uialert(app.RamanUI,'File successfully saved','Success',
'Icon', 'success');
                    else
                        uialert(app.RamanUI,'File not saved','Error', 'Icon',
'error');
                    end
                elseif(app.matButton.Value)
                    filename =
strcat(app.myDocsFolder,'\',app.myFolder,'\',datestr(now,30),'_darkSpectra','.mat')
;
                    if(app.FolderFlag)
                        save(filename,'app.DarkData');
                        uialert(app.RamanUI,'File successfully saved','Success',
'Icon', 'success');
                    else
                        uialert(app.RamanUI,'File not saved','Error', 'Icon',
'error');
                    end
                end
            elseif(app.LightdataButton.Value)
                if(app.xlsxButton.Value)
                    filename =
strcat(app.myDocsFolder,'\',app.myFolder,'\',datestr(now,30),'_lightSpectra','.xlsx
');
                    ADCvalue = app.LightData';
                    Pixel = app.x';
                    T=table(Pixel,ADCvalue);
                    if(app.FolderFlag)
                        writetable(T,filename);
                        uialert(app.RamanUI,'File successfully saved','Success',
'Icon', 'success');
                    else
                        uialert(app.RamanUI,'File not saved','Error', 'Icon',
'error');
                    end
                elseif(app.matButton.Value)
                    filename =
strcat(app.myDocsFolder,'\',app.myFolder,'\',datestr(now,30),'_lightSpectra','.mat'
);
```

```
                    if(app.FolderFlag)
                        save(filename,'app.LightData');
                        uialert(app.RamanUI,'File successfully saved','Success', 'Icon', 'success');
                    else
                        uialert(app.RamanUI,'File not saved','Error', 'Icon', 'error');
                    end
                end
            elseif(app.ReferencedataButton.Value)
                if(app.xlsxButton.Value)
                    filename = strcat(app.myDocsFolder,'\',app.myFolder,'\',datestr(now,30),'_refSpectra','.xlsx');
                    ADCvalue = app.RefData';
                    Pixel = app.x';
                    T=table(Pixel,ADCvalue);
                    if(app.FolderFlag)
                        writetable(T,filename);
                    else
                        uialert(app.RamanUI,'File not saved','Error', 'Icon', 'error');
                    end
                elseif(app.matButton.Value)
                    filename = strcat(app.myDocsFolder,'\',app.myFolder,'\',datestr(now,30),'_refSpectra','.mat');
                    if(app.FolderFlag)
                        save(filename,'app.RefData');
                    else
                        uialert(app.RamanUI,'File not saved','Error', 'Icon', 'error');
                    end
                end
            end
        end

        % Button pushed function: LightButton
        function LightButtonPushed(app, event)
            if(app.ActiveConn_flag == 0)
                uialert(app.RamanUI,'Please connect your device first!','Warning', 'Icon', 'warning');
            else
                app.LightData = app.CurrData;
                if(app.LightDataSet_flag == 0)
                    app.LightDataSet_flag = 1;
                    app.LightLamp.Color = [0.07,0.62,1.00];
                else
                    app.LightDataSet_flag = 0;
                    app.LightLamp.Color = [0.9,0.9,0.9];
                end
                app.MinADCcount = min(app.CurrData);
            end
        end

        % Button pushed function: ReferenceButton
        function ReferenceButtonPushed(app, event)
            if(app.ActiveConn_flag == 0)
                uialert(app.RamanUI,'Please connect your device first!','Warning', 'Icon', 'warning');
            else
                app.RefData = app.CurrData;
                if(app.RefDataSet_flag == 0)
                    app.RefDataSet_flag = 1;
                    app.ReferenceLamp.Color = [0.07,0.62,1.00];
                else
                    app.RefDataSet_flag = 0;
                    app.ReferenceLamp.Color = [0.9,0.9,0.9];
                end
            end
        end
```

```

        % Value changed function: YmaxEditField
        function YmaxEditFieldValueChanged(app, event)
            value = app.YmaxEditField.Value;
            app.maxY = value;
            app.UIAxes.YLim = [app.minY app.maxY];
            plot(app.UIAxes,app.x, app.CurrData);
        end

        % Value changed function: YminEditField
        function YminEditFieldValueChanged(app, event)
            value = app.YminEditField.Value;
            app.minY = value;
            app.UIAxes.YLim = [app.minY app.maxY];
            plot(app.UIAxes,app.x, app.CurrData);
        end

        % Button pushed function: YlimAuto
        function YlimAutoButtonPushed(app, event)
            app.UIAxes.YLimMode = 'auto';
            app.UIAxes.YLim = [min(app.CurrData)*0.995 max(app.CurrData)/0.995];
            plot(app.UIAxes,app.x, app.CurrData);
        end

        % Value changed function: XmaxEditField
        function XmaxEditFieldValueChanged(app, event)
            value = app.XmaxEditField.Value;
            app.maxX = value;
            app.UIAxes.XLim = [app.minX app.maxX];
            plot(app.UIAxes,app.x, app.CurrData);
        end

        % Value changed function: XminEditField
        function XminEditFieldValueChanged(app, event)
            value = app.XminEditField.Value;
            app.minX = value;
            app.UIAxes.XLim = [app.minX app.maxX];
            plot(app.UIAxes,app.x, app.CurrData);
        end

        % Button pushed function: XlimAuto
        function XlimAutoButtonPushed(app, event)
            app.UIAxes.XLimMode = 'auto';
            app.UIAxes.XLim = [0 4000];
            plot(app.UIAxes,app.x, app.CurrData);
        end

        % Button pushed function: SaveplotButton
        function SaveplotButtonPushed(app, event)
            filename = 
strcat(app.myDocsFolder,'\',app.myFolder,'\',datestr(now,30),'_Spectra','.xlsx');
            ADCvalue = app.CurrData';
            Pixel = app.x';
            T=table(Pixel,ADCvalue);
            if(app.FolderFlag)
                writetable(T,filename);
                uialert(app.RamanUI,'File successfully saved on Desktop','Success', 
'Icon', 'success');
            else
                uialert(app.RamanUI,'File not saved','Error', 'Icon', 'error');
            end
        end
    end

    % App initialization and construction
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

```

```
            % Create RamanUI
            app.RamanUI = uifigure;
            app.RamanUI.Position = [100 100 832 467];
            app.RamanUI.Name = 'RamanUI_KB';
            app.RamanUI.CloseRequestFcn = createCallbackFcn(app, @RamanUICloseRequest, true);

            % Create RamanSpectrometerv10Label
            app.RamanSpectrometerv10Label = uilabel(app.RamanUI);
            app.RamanSpectrometerv10Label.HorizontalAlignment = 'center';
            app.RamanSpectrometerv10Label.FontSize = 24;
            app.RamanSpectrometerv10Label.FontWeight = 'bold';
            app.RamanSpectrometerv10Label.Position = [262.5 416 307 32];
            app.RamanSpectrometerv10Label.Text = 'Raman Spectrometer v1.0';

            % Create DarkButton
            app.DarkButton = uibutton(app.RamanUI, 'push');
            app.DarkButton.ButtonPushedFcn = createCallbackFcn(app, @DarkButtonPushed, true);
            app.DarkButton.Icon = 'bulbOff.png';
            app.DarkButton.Position = [63 88 70 30];
            app.DarkButton.Text = 'Dark';

            % Create LightButton
            app.LightButton = uibutton(app.RamanUI, 'push');
            app.LightButton.ButtonPushedFcn = createCallbackFcn(app, @LightButtonPushed, true);
            app.LightButton.Icon = 'bulbOn.png';
            app.LightButton.Position = [63 49 70 30];
            app.LightButton.Text = 'Light';

            % Create Button
            app.Button = uibutton(app.RamanUI, 'push');
            app.Button.Position = [601 86 100 22];

            % Create TabGroup
            app.TabGroup = uitabgroup(app.RamanUI);
            app.TabGroup.Position = [580 3 236 390];

            % Create SettingsTab
            app.SettingsTab = uitab(app.TabGroup);
            app.SettingsTab.Title = 'Settings';

            % Create ConnectionsettingsPanel
            app.ConnectionsettingsPanel = uipanel(app.SettingsTab);
            app.ConnectionsettingsPanel.Title = 'Connection settings';
            app.ConnectionsettingsPanel.Position = [15 143 207 202];

            % Create COMportLabel
            app.COMportLabel = uilabel(app.ConnectionsettingsPanel);
            app.COMportLabel.HorizontalAlignment = 'right';
            app.COMportLabel.Position = [4 150 62 22];
            app.COMportLabel.Text = 'COM port:';

            % Create COMportDropDown
            app.COMportDropDown = uidropdown(app.ConnectionsettingsPanel);
            app.COMportDropDown.ValueChangedFcn = createCallbackFcn(app, @COMportDropDownValueChanged, true);
            app.COMportDropDown.Position = [75 150 124 22];

            % Create DevicenameLabel
            app.DevicenameLabel = uilabel(app.ConnectionsettingsPanel);
            app.DevicenameLabel.Position = [9 118 79 22];
            app.DevicenameLabel.Text = 'Device name:';

            % Create ConnDeviceName
            app.ConnDeviceName = uieditfield(app.ConnectionsettingsPanel, 'text');
            app.ConnDeviceName.Editable = 'off';
            app.ConnDeviceName.Position = [8 94 191 22];
            app.ConnDeviceName.Value = '<not connected>';
```

```
            % Create ConnectButton
            app.ConnectButton = uibutton(app.ConnectionsettingsPanel, 'push');
            app.ConnectButton.ButtonPushedFcn = createCallbackFcn(app, @ConnectButtonPushed, true);
            app.ConnectButton.Position = [9 58 79 22];
            app.ConnectButton.Text = 'Connect';

            % Create ConnStatusLamp
            app.ConnStatusLamp = uilamp(app.ConnectionsettingsPanel);
            app.ConnStatusLamp.Position = [167 20 24 24];
            app.ConnStatusLamp.Color = [0.902 0.902 0.902];

            % Create ConnStatusMsg
            app.ConnStatusMsg = uilabel(app.ConnectionsettingsPanel);
            app.ConnStatusMsg.HorizontalAlignment = 'right';
            app.ConnStatusMsg.Position = [15 21 133 22];
            app.ConnStatusMsg.Text = '<not connected>';

            % Create DisconnectButton
            app.DisconnectButton = uibutton(app.ConnectionsettingsPanel, 'push');
            app.DisconnectButton.ButtonPushedFcn = createCallbackFcn(app, @DisconnectButtonPushed, true);
            app.DisconnectButton.Position = [120 58 79 22];
            app.DisconnectButton.Text = 'Disconnect';

            % Create ParametersPanel
            app.ParametersPanel = uipanel(app.SettingsTab);
            app.ParametersPanel.Title = 'Parameters';
            app.ParametersPanel.Position = [15 12 207 108];

            % Create SampleaverageEditFieldLabel
            app.SampleaverageEditFieldLabel = uilabel(app.ParametersPanel);
            app.SampleaverageEditFieldLabel.HorizontalAlignment = 'right';
            app.SampleaverageEditFieldLabel.Position = [9 8 48 30];
            app.SampleaverageEditFieldLabel.Text = {'Sample'; 'average'};

            % Create SampleAverageEditField
            app.SampleAverageEditField = uieditfield(app.ParametersPanel, 'numeric');
            app.SampleAverageEditField.Limits = [0 30];
            app.SampleAverageEditField.ValueDisplayFormat = '%.0f';
            app.SampleAverageEditField.ValueChangedFcn = createCallbackFcn(app, @SampleAverageEditFieldValueChanged, true);
            app.SampleAverageEditField.Position = [79 13 120 22];
            app.SampleAverageEditField.Value = 5;

            % Create IntegrationtimeinmsLabel
            app.IntegrationtimeinmsLabel = uilabel(app.ParametersPanel);
            app.IntegrationtimeinmsLabel.HorizontalAlignment = 'right';
            app.IntegrationtimeinmsLabel.Position = [8 51 89 27];
            app.IntegrationtimeinmsLabel.Text = {'Integration time'; '(in ms)'};

            % Create IntegrationtimeDropDown
            app.IntegrationtimeDropDown = uidropdown(app.ParametersPanel);
            app.IntegrationtimeDropDown.Items = {'10', '100', '200', '500', '1000', '1500', '2000'};
            app.IntegrationtimeDropDown.ValueChangedFcn = createCallbackFcn(app, @IntegrationtimeDropDownValueChanged, true);
            app.IntegrationtimeDropDown.Position = [104 53 95 22];
            app.IntegrationtimeDropDown.Value = '10';

            % Create ArithmeticTab
            app.ArithmeticTab = uitab(app.TabGroup);
            app.ArithmeticTab.Title = 'Arithmetic';

            % Create FilteringalgorithmButtonGroup
            app.FilteringalgorithmButtonGroup = uibuttongroup(app.ArithmeticTab);
            app.FilteringalgorithmButtonGroup.Title = 'Filtering algorithm:';
            app.FilteringalgorithmButtonGroup.Position = [20 94 192 253];
```

```

            % Create EMAradioButton
            app.EMAradioButton = uiradiobutton(app.FilteringalgorithmButtonGroup);
            app.EMAradioButton.Text = 'Exponential Moving Average';
            app.EMAradioButton.Position = [11 207 175 22];
            app.EMAradioButton.Value = true;

            % Create SavitzkyGolayButton
            app.SavitzkyGolayButton = uiradiobutton(app.FilteringalgorithmButtonGroup);
            app.SavitzkyGolayButton.Text = 'Savitzky-Golay';
            app.SavitzkyGolayButton.Position = [11 91 103 22];

            % Create SmoothSlider
            app.SmoothSlider = uislider(app.FilteringalgorithmButtonGroup);
            app.SmoothSlider.Limits = [0 1];
            app.SmoothSlider.ValueChangedFcn = createCallbackFcn(app, @SmoothSliderValueChanged, true);
            app.SmoothSlider.Tooltip = {'The lesser the value the more the filtering action'};
            app.SmoothSlider.Position = [21 168 150 3];
            app.SmoothSlider.Value = 0.25;

            % Create SmoothnessfactorLabel
            app.SmoothnessfactorLabel = uilabel(app.FilteringalgorithmButtonGroup);
            app.SmoothnessfactorLabel.Position = [42 180 107 22];
            app.SmoothnessfactorLabel.Text = 'Smoothness factor';

            % Create OrderSpinnerLabel
            app.OrderSpinnerLabel = uilabel(app.FilteringalgorithmButtonGroup);
            app.OrderSpinnerLabel.HorizontalAlignment = 'right';
            app.OrderSpinnerLabel.Position = [25 60 36 22];
            app.OrderSpinnerLabel.Text = 'Order';

            % Create OrderSpinner
            app.OrderSpinner = uispinner(app.FilteringalgorithmButtonGroup);
            app.OrderSpinner.Limits = [0 Inf];
            app.OrderSpinner.ValueDisplayFormat = '%.0f';
            app.OrderSpinner.ValueChangedFcn = createCallbackFcn(app, @OrderSpinnerValueChanged, true);
            app.OrderSpinner.Position = [76 60 100 22];
            app.OrderSpinner.Value = 3;

            % Create FramelengthLabel
            app.FramelengthLabel = uilabel(app.FilteringalgorithmButtonGroup);
            app.FramelengthLabel.HorizontalAlignment = 'right';
            app.FramelengthLabel.Position = [22 13 39 27];
            app.FramelengthLabel.Text = {'Frame'; 'length'};

            % Create FramelengthSpinner
            app.FramelengthSpinner = uispinner(app.FilteringalgorithmButtonGroup);
            app.FramelengthSpinner.Step = 2;
            app.FramelengthSpinner.Limits = [1 Inf];
            app.FramelengthSpinner.ValueDisplayFormat = '%.0f';
            app.FramelengthSpinner.ValueChangedFcn = createCallbackFcn(app, @FramelengthSpinnerValueChanged, true);
            app.FramelengthSpinner.Position = [76 16 100 22];
            app.FramelengthSpinner.Value = 7;

            % Create GraphsettingsTab
            app.GraphsettingsTab = uitab(app.TabGroup);
            app.GraphsettingsTab.Title = 'Graph settings';

            % Create XmaxEditFieldLabel
            app.XmaxEditFieldLabel = uilabel(app.GraphsettingsTab);
            app.XmaxEditFieldLabel.HorizontalAlignment = 'right';
            app.XmaxEditFieldLabel.Position = [21 323 39 22];
            app.XmaxEditFieldLabel.Text = 'X max';

            % Create XmaxEditField
```

```
            app.XmaxEditField = uieditfield(app.GraphsettingsTab, 'numeric');
            app.XmaxEditField.ValueChangedFcn = createCallbackFcn(app,
@XmaxEditFieldValueChanged, true);
            app.XmaxEditField.Position = [73 323 95 22];

            % Create XminEditFieldLabel
            app.XminEditFieldLabel = uilabel(app.GraphsettingsTab);
            app.XminEditFieldLabel.HorizontalAlignment = 'right';
            app.XminEditFieldLabel.Position = [21 285 35 22];
            app.XminEditFieldLabel.Text = 'X min';

            % Create XminEditField
            app.XminEditField = uieditfield(app.GraphsettingsTab, 'numeric');
            app.XminEditField.ValueChangedFcn = createCallbackFcn(app,
@XminEditFieldValueChanged, true);
            app.XminEditField.Position = [73 285 95 22];

            % Create YmaxEditFieldLabel
            app.YmaxEditFieldLabel = uilabel(app.GraphsettingsTab);
            app.YmaxEditFieldLabel.HorizontalAlignment = 'right';
            app.YmaxEditFieldLabel.Position = [19 247 39 22];
            app.YmaxEditFieldLabel.Text = 'Y max';

            % Create YmaxEditField
            app.YmaxEditField = uieditfield(app.GraphsettingsTab, 'numeric');
            app.YmaxEditField.ValueChangedFcn = createCallbackFcn(app,
@YmaxEditFieldValueChanged, true);
            app.YmaxEditField.Position = [73 247 95 22];

            % Create YminEditFieldLabel
            app.YminEditFieldLabel = uilabel(app.GraphsettingsTab);
            app.YminEditFieldLabel.HorizontalAlignment = 'right';
            app.YminEditFieldLabel.Position = [19 210 36 22];
            app.YminEditFieldLabel.Text = 'Y min';

            % Create YminEditField
            app.YminEditField = uieditfield(app.GraphsettingsTab, 'numeric');
            app.YminEditField.ValueChangedFcn = createCallbackFcn(app,
@YminEditFieldValueChanged, true);
            app.YminEditField.Position = [73 210 95 22];

            % Create XlimAuto
            app.XlimAuto = uibutton(app.GraphsettingsTab, 'push');
            app.XlimAuto.ButtonPushedFcn = createCallbackFcn(app,
@XlimAutoButtonPushed, true);
            app.XlimAuto.Icon = 'auto.png';
            app.XlimAuto.Position = [183 298 44 37];
            app.XlimAuto.Text = '';

            % Create YlimAuto
            app.YlimAuto = uibutton(app.GraphsettingsTab, 'push');
            app.YlimAuto.ButtonPushedFcn = createCallbackFcn(app,
@YlimAutoButtonPushed, true);
            app.YlimAuto.Icon = 'auto.png';
            app.YlimAuto.Position = [183 222 44 37];
            app.YlimAuto.Text = '';

            % Create ExportTab
            app.ExportTab = uitab(app.TabGroup);
            app.ExportTab.Title = 'Export';

            % Create SelectdataButtonGroup
            app.SelectdataButtonGroup = uibuttongroup(app.ExportTab);
            app.SelectdataButtonGroup.Title = 'Select data:';
            app.SelectdataButtonGroup.Position = [19 244 199 103];

            % Create DarkdataButton
            app.DarkdataButton = uiradiobutton(app.SelectdataButtonGroup);
            app.DarkdataButton.Text = 'Dark data';
            app.DarkdataButton.Position = [11 58 75 22];
```

```
            app.DarkdataButton.Value = true;

            % Create LightdataButton
            app.LightdataButton = uiradiobutton(app.SelectdataButtonGroup);
            app.LightdataButton.Text = 'Light data';
            app.LightdataButton.Position = [11 36 76 22];

            % Create ReferencedataButton
            app.ReferencedataButton = uiradiobutton(app.SelectdataButtonGroup);
            app.ReferencedataButton.Text = 'Reference data';
            app.ReferencedataButton.Position = [11 14 103 22];

            % Create SelectfileextensionButtonGroup
            app.SelectfileextensionButtonGroup = uibuttongroup(app.ExportTab);
            app.SelectfileextensionButtonGroup.Title = 'Select file extension:';
            app.SelectfileextensionButtonGroup.Position = [18 151 199 75];

            % Create xlsxButton
            app.xlsxButton = uiradiobutton(app.SelectfileextensionButtonGroup);
            app.xlsxButton.Text = '.xlsx';
            app.xlsxButton.Position = [11 29 47 22];
            app.xlsxButton.Value = true;

            % Create matButton
            app.matButton = uiradiobutton(app.SelectfileextensionButtonGroup);
            app.matButton.Text = '.mat';
            app.matButton.Position = [11 7 46 22];

            % Create SaveButton
            app.SaveButton = uibutton(app.ExportTab, 'push');
            app.SaveButton.ButtonPushedFcn = createCallbackFcn(app, @SaveButtonPushed, true);
            app.SaveButton.Position = [120 114 100 22];
            app.SaveButton.Text = 'Save ';

            % Create UIAxes
            app.UIAxes = uiaxes(app.RamanUI);
            xlabel(app.UIAxes, 'Pixels')
            ylabel(app.UIAxes, 'ADC count')
            app.UIAxes.PlotBoxAspectRatio = [1 0.471337579617834 0.471337579617834];
            app.UIAxes.XMinorTick = 'on';
            app.UIAxes.YMinorTick = 'on';
            app.UIAxes.XGrid = 'on';
            app.UIAxes.YGrid = 'on';
            app.UIAxes.Position = [22 134 550 280];

            % Create MeasureButton
            app.MeasureButton = uibutton(app.RamanUI, 'push');
            app.MeasureButton.ButtonPushedFcn = createCallbackFcn(app, @MeasureButtonPushed, true);
            app.MeasureButton.Position = [433 92 130 22];
            app.MeasureButton.Text = 'Measure';

            % Create MonitorButton
            app.MonitorButton = uibutton(app.RamanUI, 'push');
            app.MonitorButton.ButtonPushedFcn = createCallbackFcn(app, @MonitorButtonPushed, true);
            app.MonitorButton.Icon = 'play_pause.png';
            app.MonitorButton.Position = [291 92 70 22];
            app.MonitorButton.Text = 'Monitor';

            % Create SpectraCountEditField
            app.SpectraCountEditField = uieditfield(app.RamanUI, 'numeric');
            app.SpectraCountEditField.Limits = [0 Inf];
            app.SpectraCountEditField.ValueDisplayFormat = '%.0f';
            app.SpectraCountEditField.Editable = 'off';
            app.SpectraCountEditField.Position = [291 65 70 22];

            % Create ReferenceButton
```

```
            app.ReferenceButton = uibutton(app.RamanUI, 'push');
            app.ReferenceButton.ButtonPushedFcn = createCallbackFcn(app, @ReferenceButtonPushed, true);
            app.ReferenceButton.Position = [62.5 11 71 30];
            app.ReferenceButton.Text = 'Reference';

            % Create DarkLamp
            app.DarkLamp = uilamp(app.RamanUI);
            app.DarkLamp.Position = [44 98 10 10];
            app.DarkLamp.Color = [0.902 0.902 0.902];

            % Create LightLamp
            app.LightLamp = uilamp(app.RamanUI);
            app.LightLamp.Position = [44 59 10 10];
            app.LightLamp.Color = [0.902 0.902 0.902];

            % Create ReferenceLamp
            app.ReferenceLamp = uilamp(app.RamanUI);
            app.ReferenceLamp.Position = [44 21 10 10];
            app.ReferenceLamp.Color = [0.902 0.902 0.902];

            % Create MonitorLamp
            app.MonitorLamp = uilamp(app.RamanUI);
            app.MonitorLamp.Position = [277 98 10 10];
            app.MonitorLamp.Color = [0.902 0.902 0.902];

            % Create MeasureLamp
            app.MeasureLamp = uilamp(app.RamanUI);
            app.MeasureLamp.Position = [419 98 10 10];
            app.MeasureLamp.Color = [0.902 0.902 0.902];

            % Create SaveplotButton
            app.SaveplotButton = uibutton(app.RamanUI, 'push');
            app.SaveplotButton.ButtonPushedFcn = createCallbackFcn(app, @SaveplotButtonPushed, true);
            app.SaveplotButton.Icon = 'kisspng-computer-icons-download-favicon-save-icon-png-5ab0783fc5bea3.4073257715215145598l.jpg';
            app.SaveplotButton.Position = [433 20 130 30];
            app.SaveplotButton.Text = 'Save plot';
        end
    end

    methods (Access = public)

        % Construct app
        function app = RamanUIv1

            % Create and configure components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.RamanUI)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.RamanUI)
        end
    end
end
```