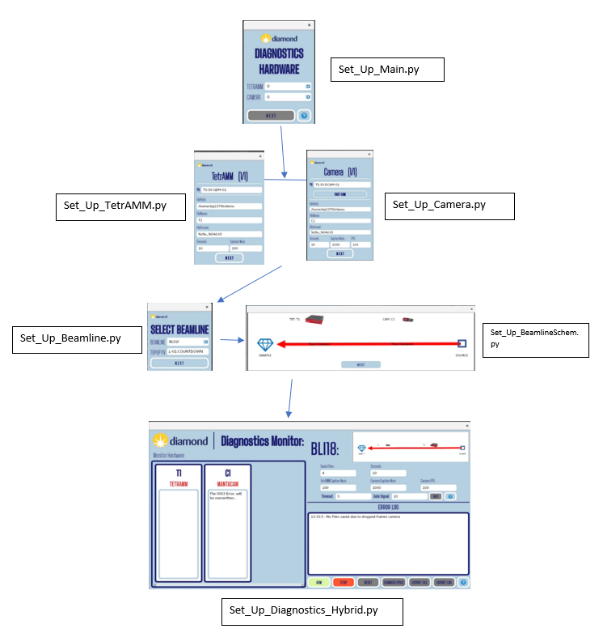
| Technical Division, Diagnostics |  | Summer Student Project  Date: September 17, 2021 |
| --- | --- | --- |

Synchronous X-Ray Diagnostics System

*Codrutza Dragu*, 

**Supervisor: Claire Houghton**

September 17, 2021

# Abstract / Summary

This project is centered around helping the Diagnostics team acquire better data during their beamline vibrations analysis. This is done through a two-fold system, consisting of a ‘button-press’ hardware element to synchronously trigger the diagnostics hardware on the beamline, and a software package to fix any issues during data acquisition, and provide a user-friendly analysis interface. The system was tested on the i18 beamline in late June, and approximately 7hours worth of synchronous data was acquired and analyzed. The graphs produced at the end of the experiment indicate possibly uncorrelated vibrations coming from the horizontal focusing mirrors closest to the sample.

**Contents**

[Abstract / Summary 1](#_gjdgxs)

[1. Introduction 3](#_30j0zll)

[2. Project Deliverables 3](#_1fob9te)

[3. Hardware Solution 4](#_3znysh7)

[4. Software Section 11](#_2et92p0)

[5. Beamline Testing: I18 16](#_tyjcwt)

[6. Conclusions 19](#_3dy6vkm)

[References 20](#_1t3h5sf)

# Introduction

An issue many beamlines face, and one that affects the quality of the data acquired on the beamline, is if there are significant unwanted vibration present during beam time. These vibrations could be internal and come from the optical elements along the beamline, or from elements in the ring. They could even be external and come from overhead motion or from the air conditioning system. Identifying the source of unwanted vibration is critical in maintaining the quality of the beamline and the stability of the beam. Use of the diagnostics hardware present on the beamline allows data to be taken at beamline positions, and compared to determine whether there is uncorrelated motion present between the different acquisition devices, and if so, then further identification of the component causing the vibration. To accurately determine non-correlated motion, data should be acquired synchronously across all the hardware. The current limitation is that the EPICS interface controlling the hardware allows only manual acquisition for each hardware, and has no way to coordinate grabbing data from multiple hardware at the same time. The scope of this project is to produce a physical trigger system to control the hardware, and a software package that allows the trigger experiment to be monitored and has post-experiment Fourier analysis capabilities.

# Project Deliverables

The project scope presented at the beginning of the placement specifies the following targets:

1. Develop a system to acquire 30s of synchronous data from a CANels TetrAMM (four channels) and a AVT Manta G235b camera (60 X 60 pixel images) at the push of a button. The acquisition systems should be triggered to begin acquisition to within 0.1 ms of each other. The system should be able to operate with difference data rate from each device.
2. A graphical user interface (GUI) should be developed to display the measured beam motion over the 30s period acquired from both the CAENels TetrAMM and the AVT Manta G235b camera. A method for highlighting correlated motion and uncorrelated motion should be developed. This should show the user if a particular motion is correlated, and seen in both measurements or uncorrelated, and only seen in one instrument.
3. An indication on the graphical user interface to direct users to a specific component or components that could be the cause of the correlated or uncorrelated motion.
4. The ability for the GUI to display spectral data, using fast Fourier transforms to display the 30s period acquisition in the frequency domain. Highlight the largest frequency contributions in both directions (horizontal and vertical) and indicate if this is correlated or uncorrelated between devices.
5. A project report, detailing the process used to develop and test the system complete with instructions for the continued use of the system. A presentation of the main results should be given before the close of this project discussing the deliverables above and how well these were met.

Each deliverable and the success to which it was delivered will be outlined at the end of this report, however, throughout the project, further targets were identified as follows:

1. Allow the trigger system to have selectable modes: it should be able to send a one-off trigger signal, and should also be able to continuously acquire data
2. Allow the continuous trigger system to be self-fixing when issue with file-saving Camera/TetrAMM network issues arise

# Hardware Solution

This section outlines the development of the hardware (and accompanying software) solution to synchronously trigger the acquisition devices. A few different trigger-transmission solutions were explored and evaluated before selecting the final idea. The table below outlines the alternative hardware solutions, and comments on their suitability:

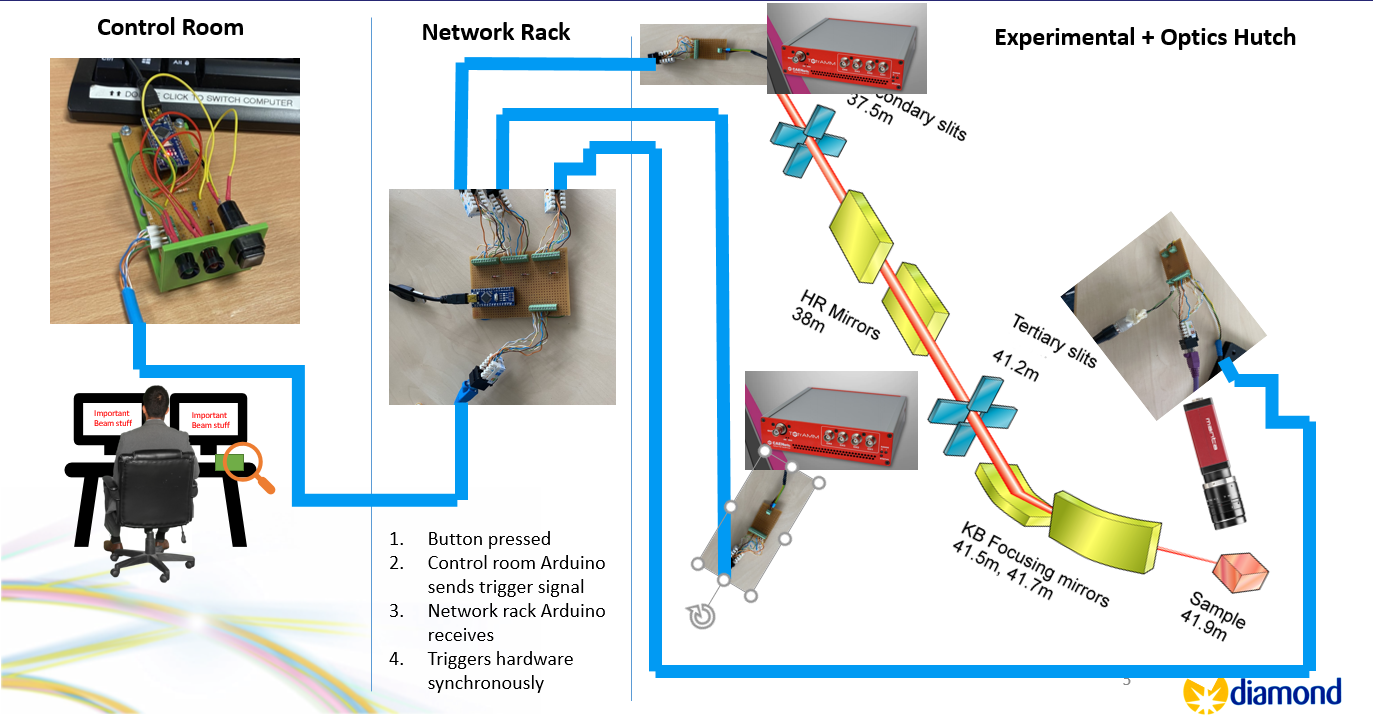
| **Solution Description** | **Pros** | **Cons** |
| --- | --- | --- |
| **1 Arduino Solution, Ethernet Cable Voltage Transmission:**  Control End: 1 Arduino on the control side, with separate pins connected to 3 separate outputs to be connected to the acquisition hardware  Middle Box: Ethernet cable spliced and wire pairs connected to separate ethernet cables before the pre-patch panels.  Receiver End: Ethernet cable spliced to TetrAMM and Camera connection | * Less resources used (only one Arduino) * Makes use of the existing infrastructure * Everything is control-side, makes maintenance easier * Cheap solution as it makes use of pre-existing resources * Allows different time triggering | * More susceptible to timing differences due to varying cable length * More likely to strain the Arduino from drawing more current |
| **2 Arduino solution, Ethernet Cable Voltage Transmission:**  Control End: 1 Arduino to send the trigger signal upon button press. Signal passed through two out of the 4 ethernet cable wire pairs (other 2 are ground) to ensure that if a wire is tampered with there at three more still sending the signal.  Middle Box: Trigger received by another Arduino which synchronously triggers three outputs and accounts for the delays between different types of hardware.  Receiver End: Ethernet cable spliced to TetrAMM and Camera connection | * Better proof of concept for panda box (repeater + panda) * Due to less cable between the second Arduino and devices, less resistance, more reliable voltage output * Arduino box is moveable to hutch if all components there (less resistance, better signal quality) * More reliable half-way point * Hardware is triggered with voltage, no need for extra components | * Uses more resources (2x Arduino nano) * Maintenance a bit more difficult * Takes longer to build / program two sets of Arduinos * Arduino may not be able to generate enough voltage to trigger the devices |
| **2 Arduino solution, Ethernet-Sheild transmission:**  Control End: 1 Arduino to send the trigger signal upon button press. Signal is passed through the data lines of the Ethernet cable using the Arduino Ethernet Sheild  Middle Box: Trigger received by another Arduino’s Ethernet Sheild which decodes the trigger data and synchronously triggers three outputs (via voltage) and accounts for the delays between different types of hardware.  Receiver End: Ethernet cable spliced to TetrAMM and Camera connection | * Better proof of concept for panda box (repeater + panda) * Due to less cable between the second Arduino and devices, less resistance, more reliable voltage output * Arduino box is moveable to hutch if all components there (less resistance, better signal quality) * Likely to lengthen the life of the ethernet cables, as they were meant more for data transmission as opposed to voltage | * Uses more resources (2x Arduino nano) * Maintenance a bit more difficult * Takes longer to build / program two sets of Arduinos * Arduino may not be able to generate enough voltage to trigger the devices * Mixing trigger signal types may be confusing * The ethernet shield is very expensive [1] |

*Table 3.1: Outlining the different solution options for the hardware portion of the trigger*

Considering the points above, the solution chosen to be developed is the **2 Arduino Solution, Ethernet cable voltage transmission.** This was chosen due to the ease of implementation with the existing network rack architecture, and due to proof of concept for a future endeavor: The ‘PandaBox’, which in a sense acts like the second ‘middle man’ Arduino.

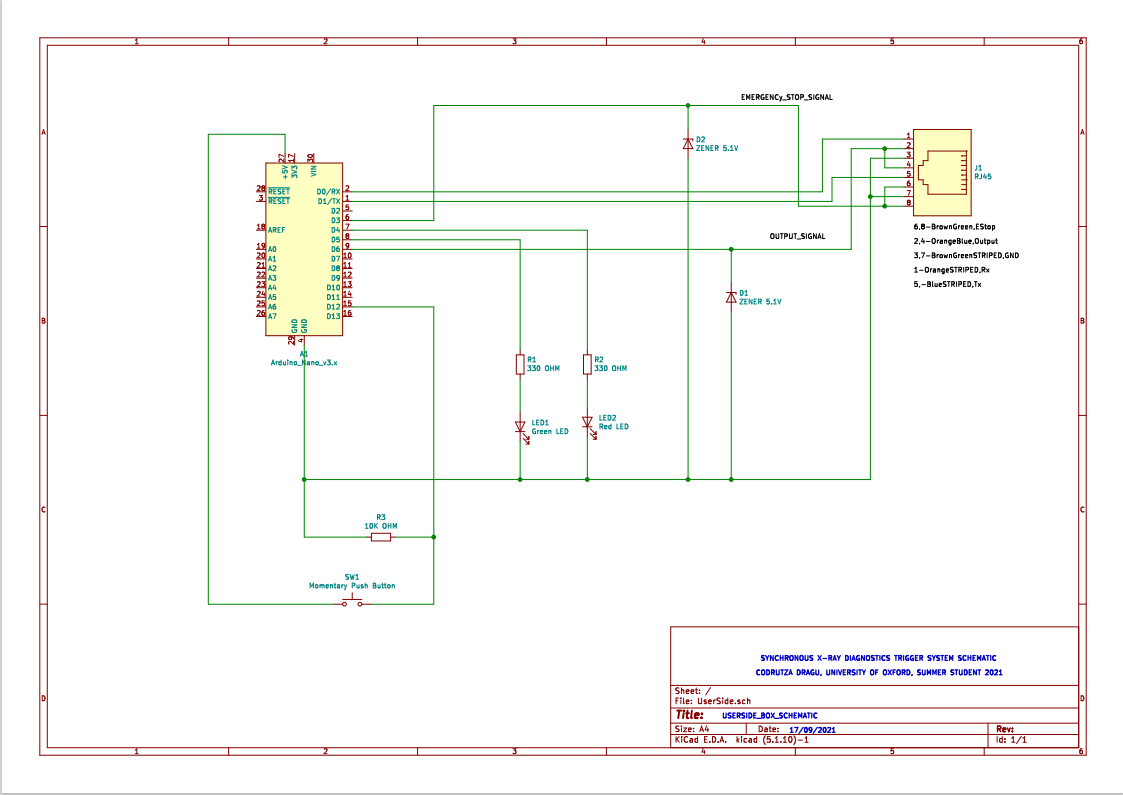
**Hardware Details and Features:**

The hardware package consists of 4 bespoke schematic designs for a **UserSide Box, Midbox, TetrAMM Convertor** and **Camera Convertor** to be used on the beamline. Figure 3.1 outlines the placement and use of these separate hardware components on a typical beamline:



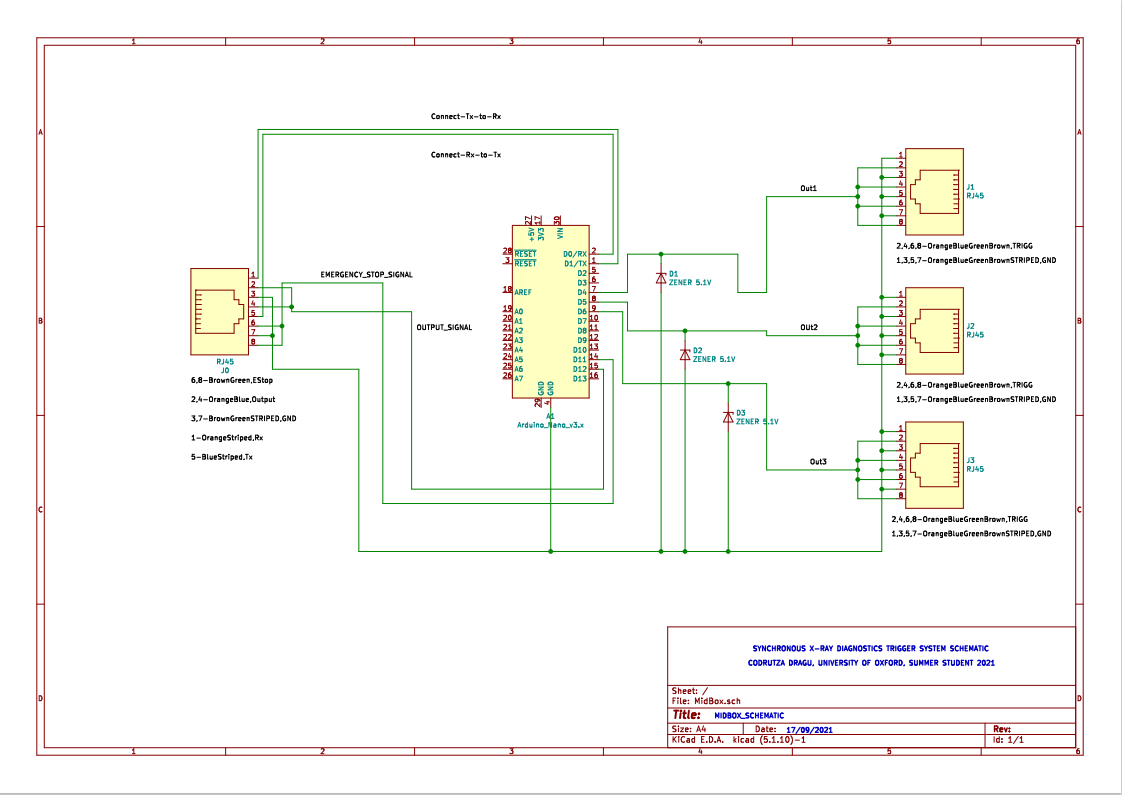
*Figure 3.1: Showing the UserSide Arduino with the trigger button placed in the Control Room, the MidBox Arduino placed in the Network Rack, and the converters placed near the trigger port of their respective hardware on the beamline. All these devices are connected via traditional Cat6 type B wires.*

Detailed information on each of the trigger hardware components and their features are outlined in the Confluence Help Guide [2]. The design of each of the circuit boards are elaborated on below:



*Figure 3.2: UserSide Schematic Diagram*

This Arduino features two LEDs to indicate the state of a trigger, their appropriate resistors, and a button with a pull up resistor as the main interaction interface. The pull-up resistor of the button was chosen such that it would record state HIGH when being pressed. The Zener diodes connected to ground before the output are a safety feature to prevent any back-current surge from affecting the Arduino pins. The output to ethernet cable transmits the voltage trigger, its ground, and also forms an RX/TX (read/write) connection with the other Arduino to enable serial communication. This was done to be able to set the Midbox Arduino up from the UserSide Arduino, thus making the system more convenient.

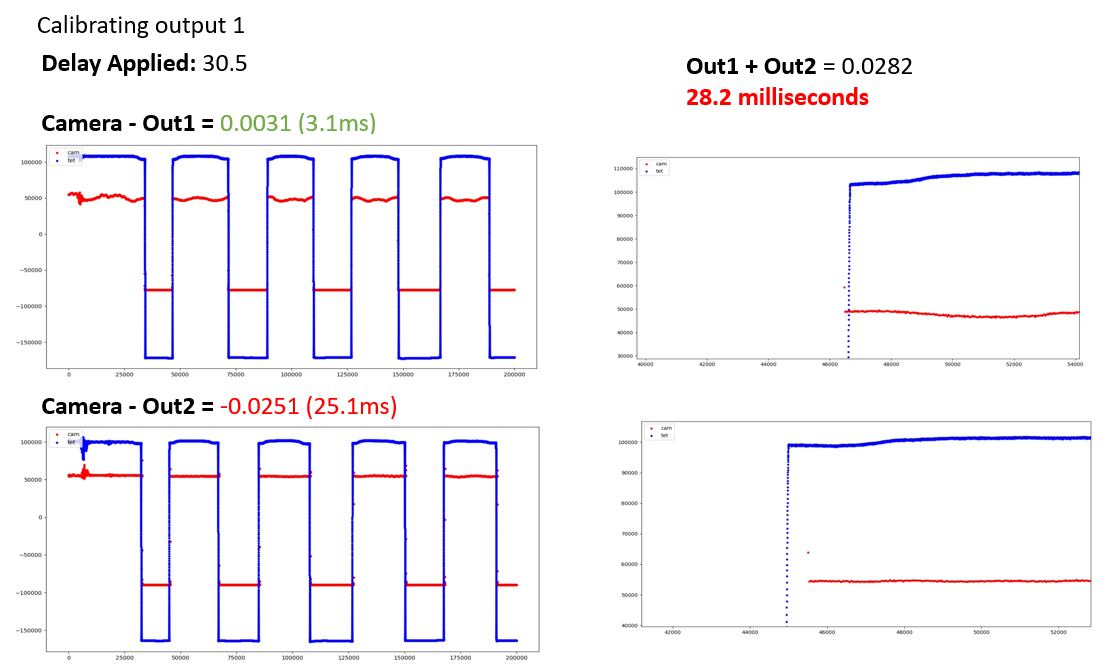


*Figure 3.4: Midbox Schematic Diagram*

The Midbox Arduino receives the voltage and serial signal via ethernet cable, and uses port register manipulation to send voltage from the required pins simultaneously, but also taking into account the delay of the Camera with respect to the TetrAMM. Port register manipulation is used as opposed to the normal DigitalPinWrite because it allows a series of pins to be turned on with higher precision.

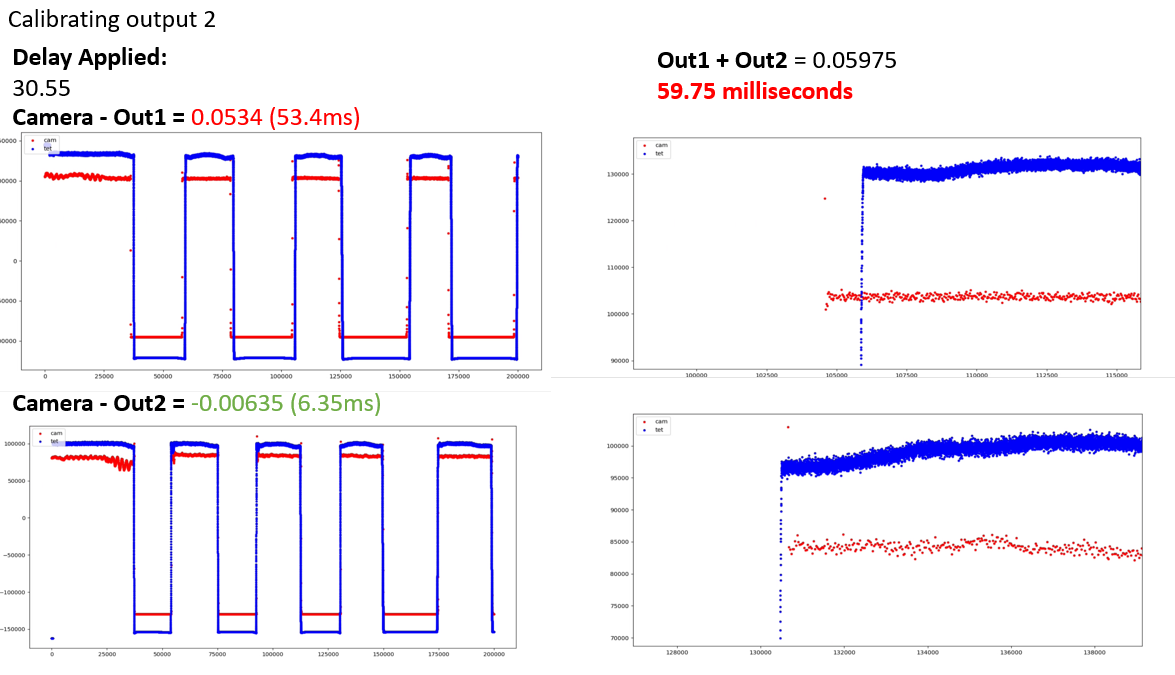
*Pin Triggering:*

When experimenting with the delay calculation between the different hardware devices, both the classing DigitalPinWrite and Register manipulation methods were evaluated. The delay between the Camera and TetrAMM was experimentally determined to be 30.5ms. This was done by comparing the spikes in data received by both hardwares on an optical table where a beam-split laser light was shinning on them and being periodically blocked. When applying this delay with the DigitalPinWrite method and calibrating for the first TetrAMM output that would be turned on, it was observed that the discrepancy between the Camera and the first output TetrAMM was very small, whilst the discrepancy between the Camera and the second output TetrAMM, and between both TetrAMMs was very big:



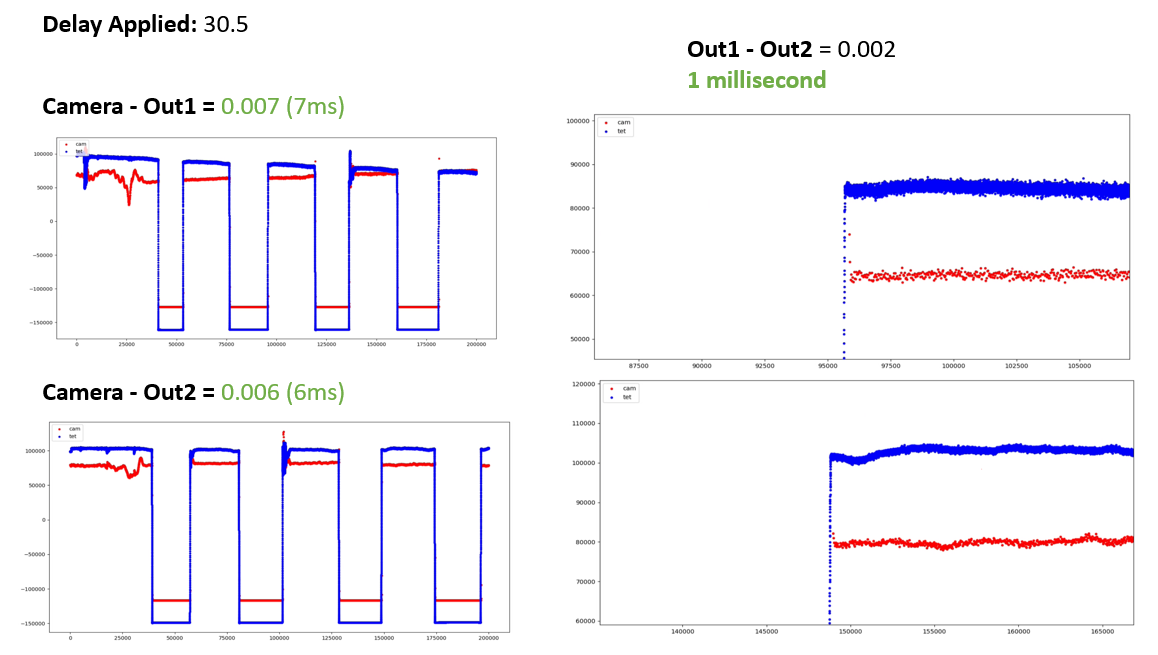
*Figure 3.5: Showing the level of accuracy the DigitalPinOut method would produce when calibrating for the first TetrAMM output*

As the DigitalPinWrite turns the outputs ‘on’ in sequence, though the first TetrAMM output waveform and the camera waveform are close in phase, the same cannot be said between the camera and the second TetrAMM output. The reason for this discrepancy is that it takes time for the C++ code to be converted to machine code, causing this 25.1ms delay in between DigitalPinWrite statements. Calibrating for the second output has a similar detrimental affect on the phase of the camera and the first TetrAMM output’s waves:



*Figure 3.6: Showing the level of accuracy the DigitalPinOut method would produce when calibrating for the second TetrAMM output*

In comparison, the Port Register manipulation method uses the lowest level of pin/port manipulation to turn pins on simultaneously, This method proved more accurate than the last, and precision of 7ms between the camera and the first TetrAMM output, 6ms between the camera and the second TertAMM output, and only 1ms discrepancy between the two TetrAMM outputs, which is the highest degree of accuracy that the Arduino hardware can maintain:

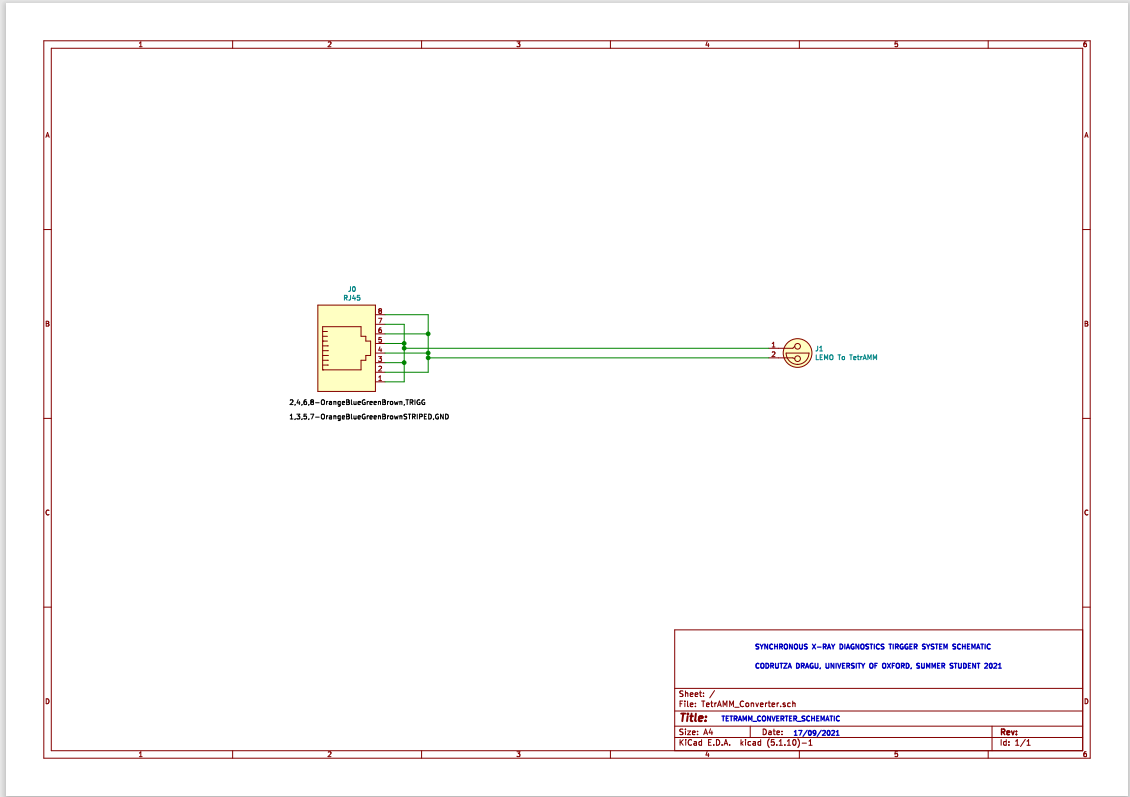


*Figure 3.7: Showing the level of accuracy the port register manipulation method would produce between the camera and the TetrAMMs*

A setback when developing this method was when the pin triggered by port manipulation caused a voltage of 3.8V to be outputted as opposed to 5V. This was problematic, as the TetrAMM cannot trigger at this level. The reason for the low output was an issue with the code, and the oscilloscope revealed that the output was fluctuating between 5V and 0, giving an average of 3.8V.

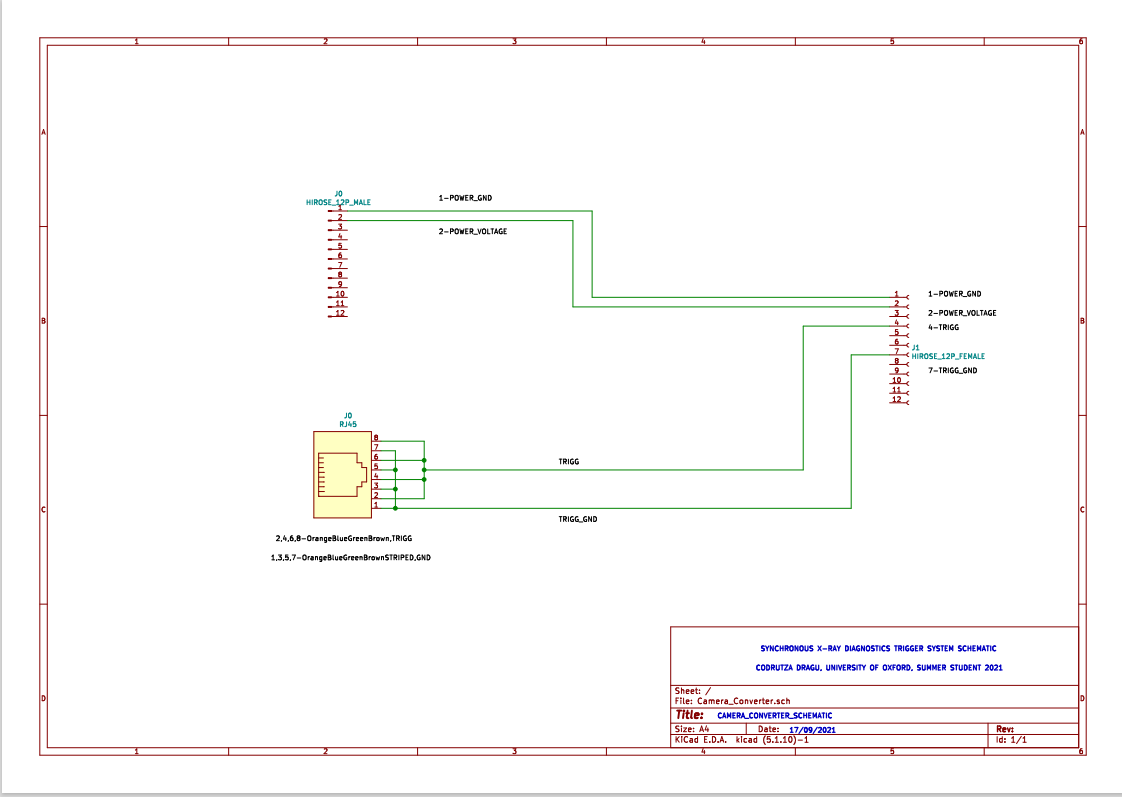


*Figure 3.8: Showing the oscilloscopes before and after the code issue was fixed*

**

*Figure 3.9: TetrAMM Converter Schematic*

This is a simple circuit which routes the voltage and ground into the appropriate inner/outer wires of the Lemo connector.



*Figure 3.9:Camera Converter Schematic*

This is a simple circuit connects the power and ground pins of the original camera power cable and the trigger pins to the hybrid Hirose 12Pin connector for the camera.

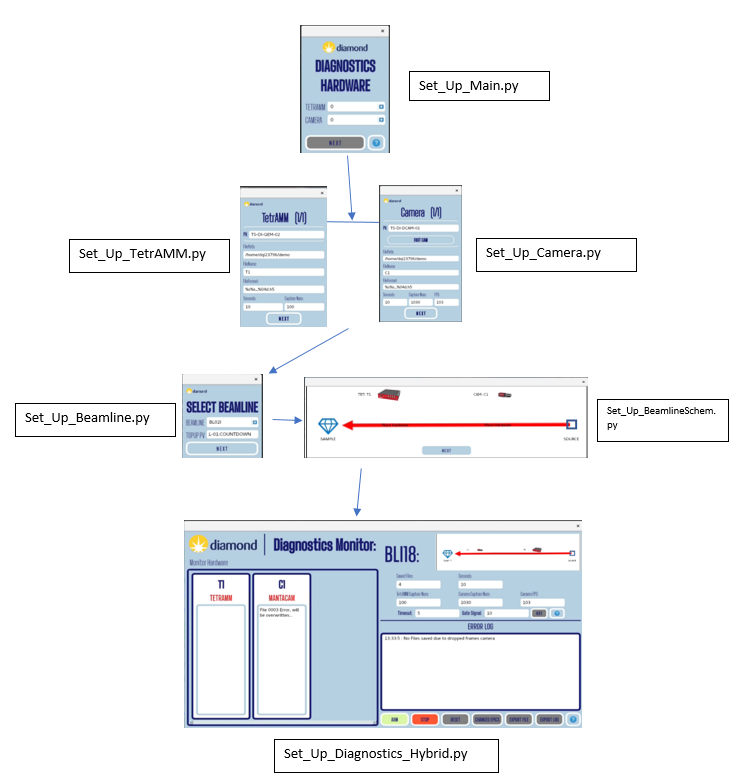
The final piece of the hardware package is the 3D printed enclosures to put the circuits in, as shown below:



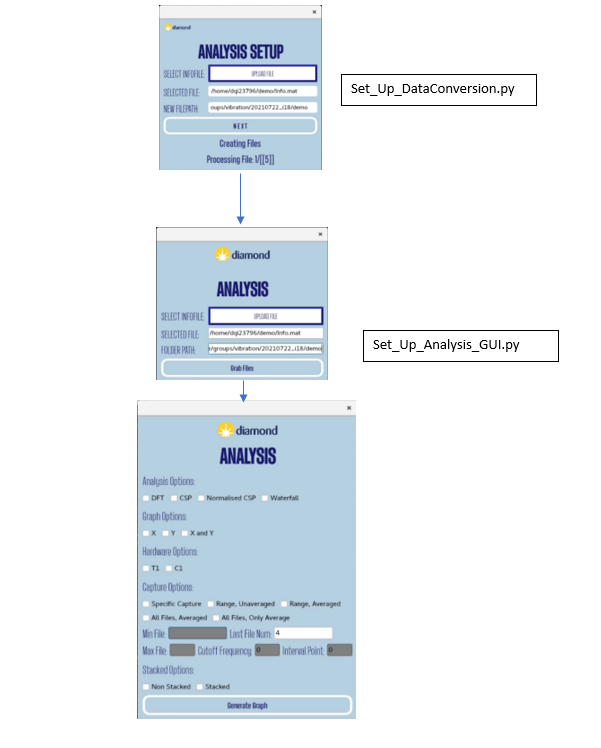
*Figure 3.10: 3D printed enclosures for the trigger system*

# Software Section

This section outlines the development of the software package both for the monitoring system during the experiment and the Fourier Analysis GU. Detailed descriptions of each of the GUIs is available on the confluence help guide [2], but a top down diagram of the GUIs and of the coding packages is shown below:



*Figure 4.1: Drop down diagram of the software to use when setting up the experiment and during the experiment.*



*Figure 4.2: Drop down diagram of the software to use when converting the data and choosing the analysis options. A document is available on the confluence help page [2] outlining the different analysis options*

Though more detail is available in the help guide, a brief summary of the code can be done for the experiment package and the analysis package. For the experiment package, after setting up the hardware and the beamline order schematic, the main diagnostics page displays real time error log data and a file increment counter. The loop code behind this (Main\_Loop\_Code.py) is a self sufficient code that can rectify problems as they arise. There are three kinds of problems identified from the i18 beamline experiment in June which the code can fix:

1. **TOPUP:** Any file containing data corrupted by the beam topup will not be saved. Based on the length of time the data is taken for and the feedback from the topup PV, the code will determine whether the file has caught any topup, and if so, clear the HDF5 buffer and restart the data acquisition without saving the file
2. **CAMERA DROPPED FRAMES:** If the frames per second has reduced due to network instability, the dropped frames incrementor will be updated. If the code senses that the file has dropped frames, it will not save this file and will reset the camera. The reset function usually reboots the camera and fixes the network issue.
3. **FILES NOT SAVED PROPERPY:** If the file has not been saved properly, the hardware with the issue will undergo a reset before the next trigger acquisition start. This may be due to network instability or lag causing the cothread timeout function to expire. There is a way to manually increase the timeout time on the Diagnsotics\_Hybrid GUI if necessary.

Other features of the main diagnostics page include the ‘Save File’ button, which saves a copy of the hardware objects created and other important beam experiment metadata, eg the frames per second and the file path of the saved HDF5 files. This file is needed to perform the data conversion and the analysis in the post-experiment processing

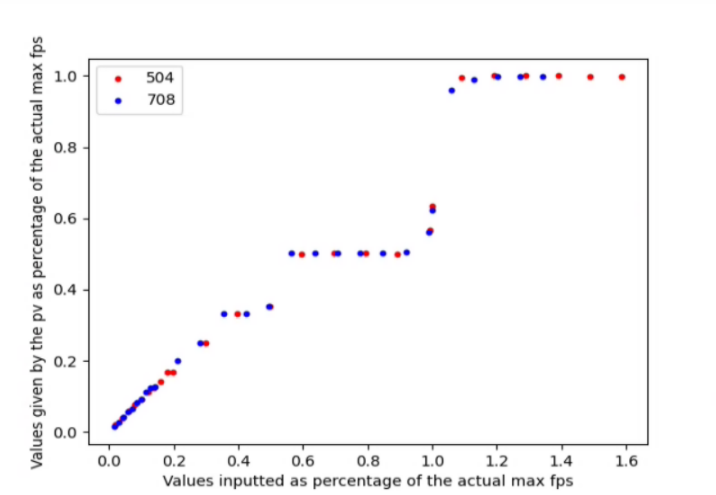
For the post-experiment package, the first thing the code will do is to convert the HDF5 files into .mat files. During this process, the HDF5 files will be processed (but not fourier transformed) and gaussian fitted in necessary to produce analyzable data sets. This is done because the fitting and reshaping of the HDF5 data takes a long time, and the matfiles are easy accessible can be saved for other analysis. Once these matfiles are done, the analysis toolbar options GUI can be initialized. This will produce different graphs based on the options selected. Multiple graphs can be produced without being cleared, and a document outlining the appearance of the graphs based on the choices selected is available on the confluence help page [2].

**Other Features of the Software Package**

When developing the software package, two side-tasks also explored was to determine an equation (if any) for the max frame rate PV, and to explore windowing functions and DC drift.

*Max Frame Rate PV*

When exploring the max frame rate PV, the values suggested by EPICS versus the value put in by the user was plotted for two tests with different frames per second in the graph below:



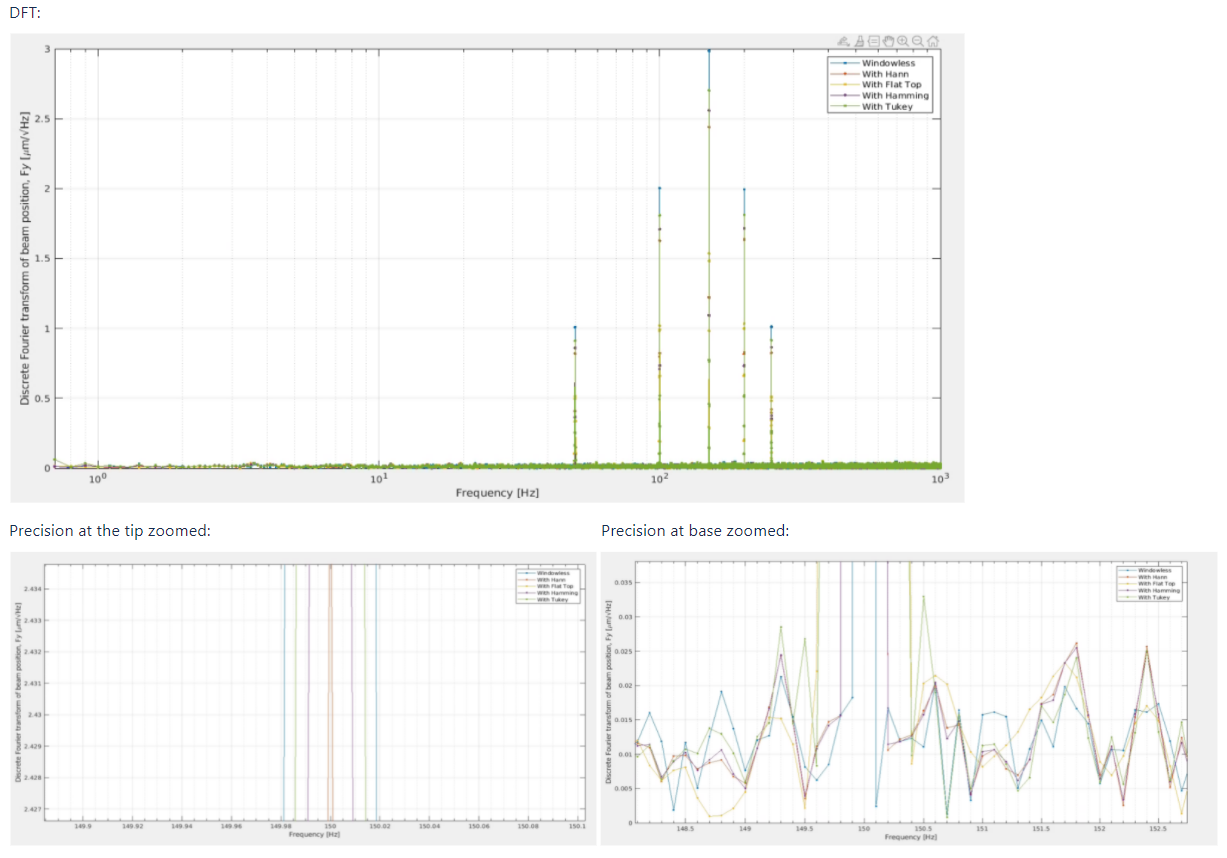
*Figure 4.3: Graph outlining the discrepancy of the max frame rate [FPS] suggested value versus the one that user wanted, for frames per second of 504 and 708.*

The x-axis of Figure 4.3 is the value manually inputted by the user into the mac frame rate PV as a percentage of the actual maximum frame rate, and the y-axis is the corresponding value that the fps was changed to as a percentage of the actual maximum frame rate. Results show that inputting small fps gives accurate values, as there is a linear relationship with a gradient of approximately 1 as the value inputted become the value the fps was changed to. After inputting a value approximately of 50% of the maximum frame rate, this linear relationship no longer holds, and the following values up until just before inputting the max frame rate itself plateau at 55% percent of the actual max fps. Inputting the exact max fps into the max array PV will not actually yield the same value, instead, it will yield 63% of the max fps which rises steeply. This plateaus very quickly to the actual max fps and remains there from an inputted value 1.1 times the max fps onwards.

In summary, the good performance at low fps inputs is not useful for our purposes. We are more interested in the relationship closer to the max fps which could be used to adjust this value during bad frames.

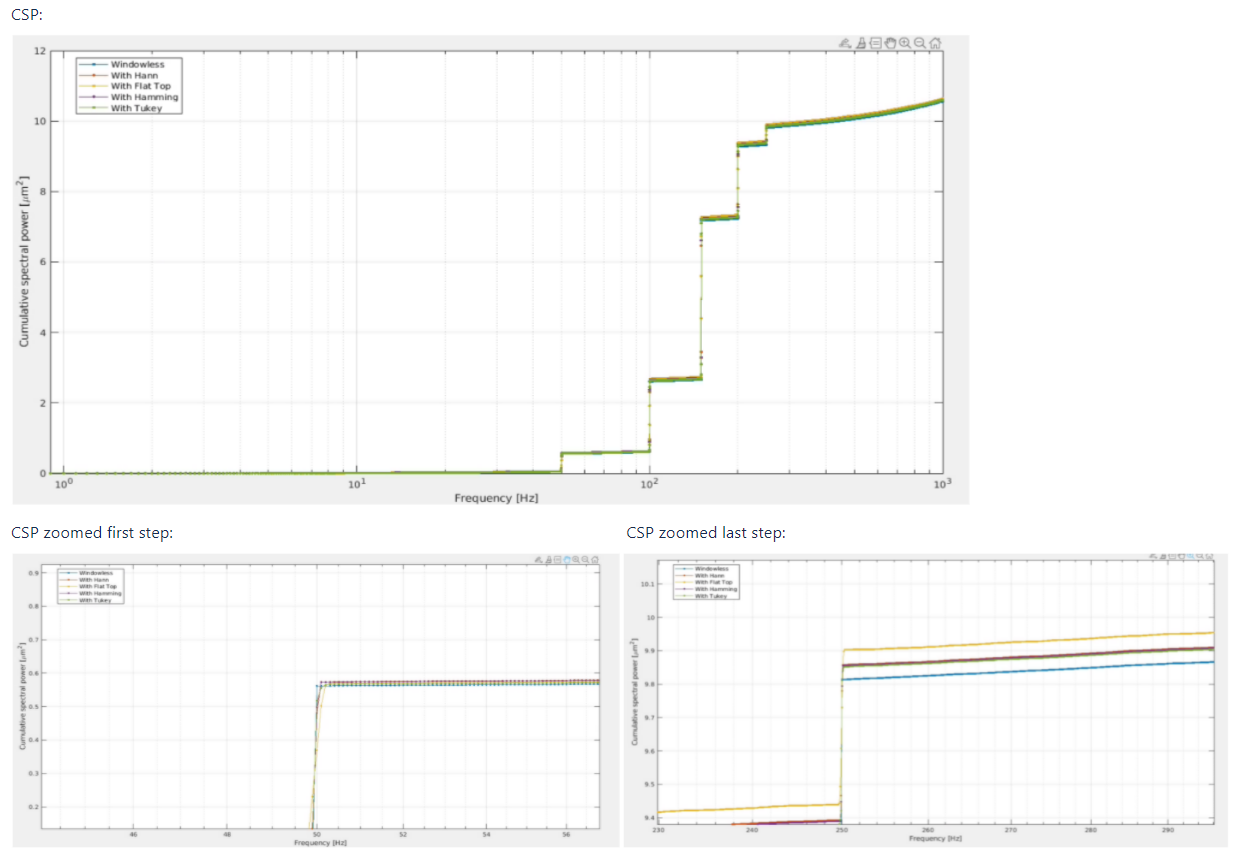
*Windowing Functions:*

When exploring windowing, various windowing functions including the Hann (currently used), Hamming, Flattop, and Tukey functions. Detailed descriptions of each of these functions and why they were considered are included on the windowing functions help guide on confluence [5]. In summary, dummy beam data was generated to test the performance of the windowing functions, and the Hann window was deemed to be the most appropriate windowing function for this type of data.



*Figure 4.4: DFT plots both zoomed out (top) and zoomed in (bottom) of dummy beam data for each of the windowing functions*

All the windowing functions can identify the present frequencies, but some to a higher degree of accuracy than the others. For this test, the graph zooming in towards the tip of the peak shows that the Hann window has the greatest frequency accuracy, and seems to be most uniform (along with the Hamming) at the base. The Tukey still has the best amplitude accuracy, but shows significant amplification of noisy data at the base (the large spikes on either side of the main peak), which is unideal. In this scenario, the Hann is most appropriate.



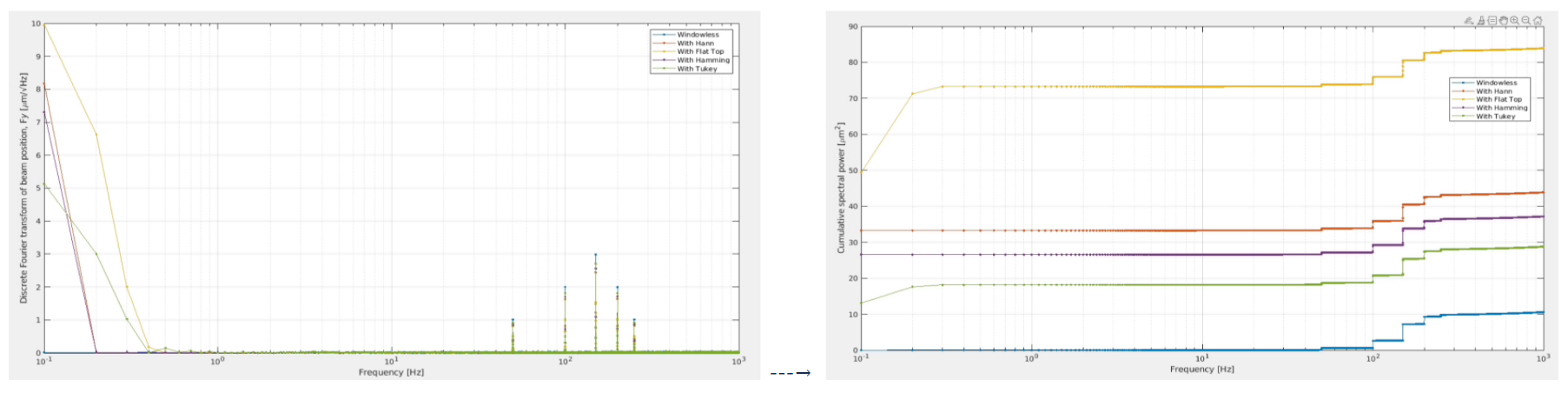
*Figure 4.5: CSP plots both zoomed out (top) and zoomed in (bottom) of dummy beam data for each of the windowing functions*

The CSP graph shows that all the windows follow the original shape quite well, with the major discrepancy being the last step (variance) at the end. The window producing the largest variance, which is unideal, is the flat top. This window has failed every single matlab test thusfar, and it should in no way be considered for application to real beam data. The Hamming, Hann and Tukey windows all have the same or similar final variance. This is approximately 0.5um^2 higher than the windowless function, which is a decent degree of accuracy.

In conclusion, the flat top window is completely unsuitable for this kind of analysis and will not be investigated further. The Tukey window performed the best from an amplitude point of view, but it amplified the most spectral noise, and is not super accurate frequency-wise. The final verdict lies between the Hamming and Hann windowing functions. Though the Hamming may provide slightly better amplitude accuracy, it is my opinion that the benefits from this are not significant enough to change the Diagnostics team's windowing function standard, and the Hann already performs very well and to a high degree of accuracy. Therefore, the Hann window is still the most suitable windowing function for the diagnostics team. The confluence windowing help guide [5] should also be referred to for a more detailed overview of how the windowing function was chosen.

*DC Drift: Windowing Functions Phenomenon:*

The phenomenon of DC drift was observed for certain windowing functions but not for others:

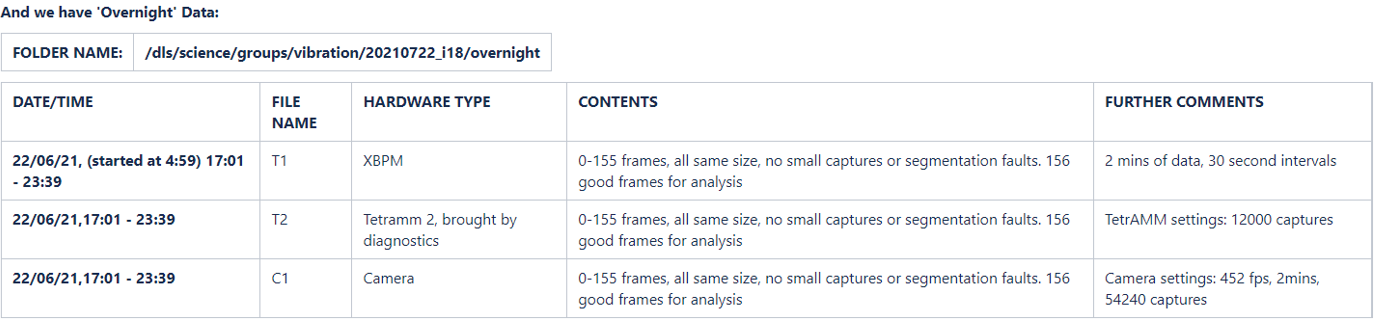


*Figures 4.4: Graph showing the effect of DC drift after windowing for (left) the Discrete Fourier Transform (DFT) and (right) the Cumulative Spectral Power (CSP).*

This problem was rectified by cutting off the first few points at the lowest frequency to remove the drift effect. More information on this is available on the confluence windowing help guide [5].

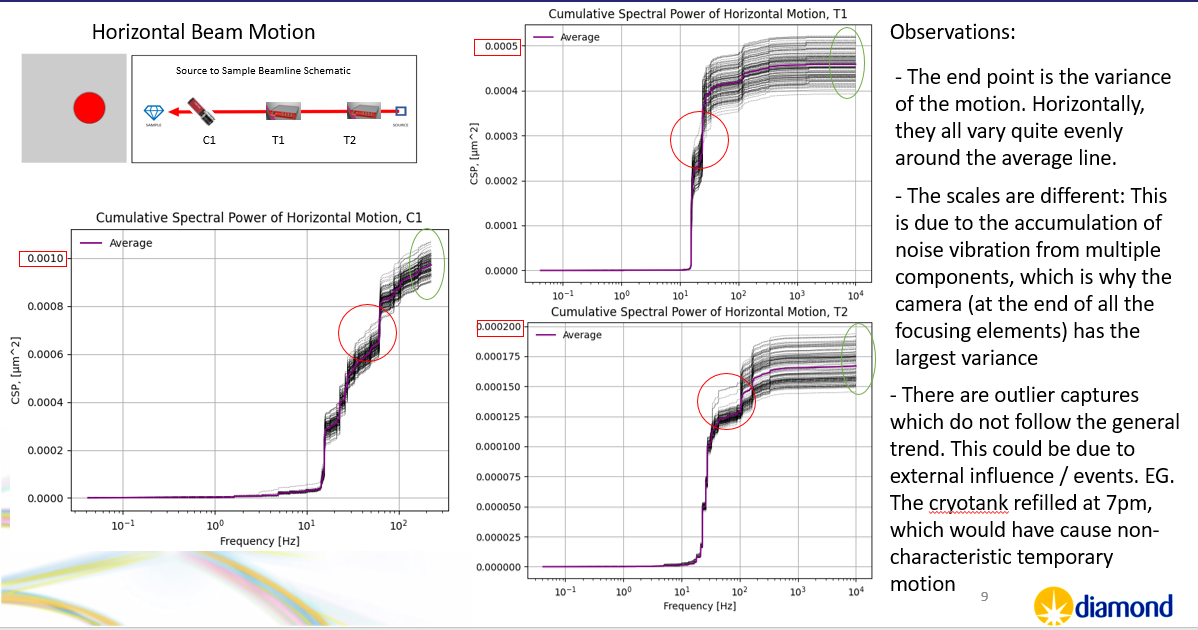
# 5. Beamline Testing: I18

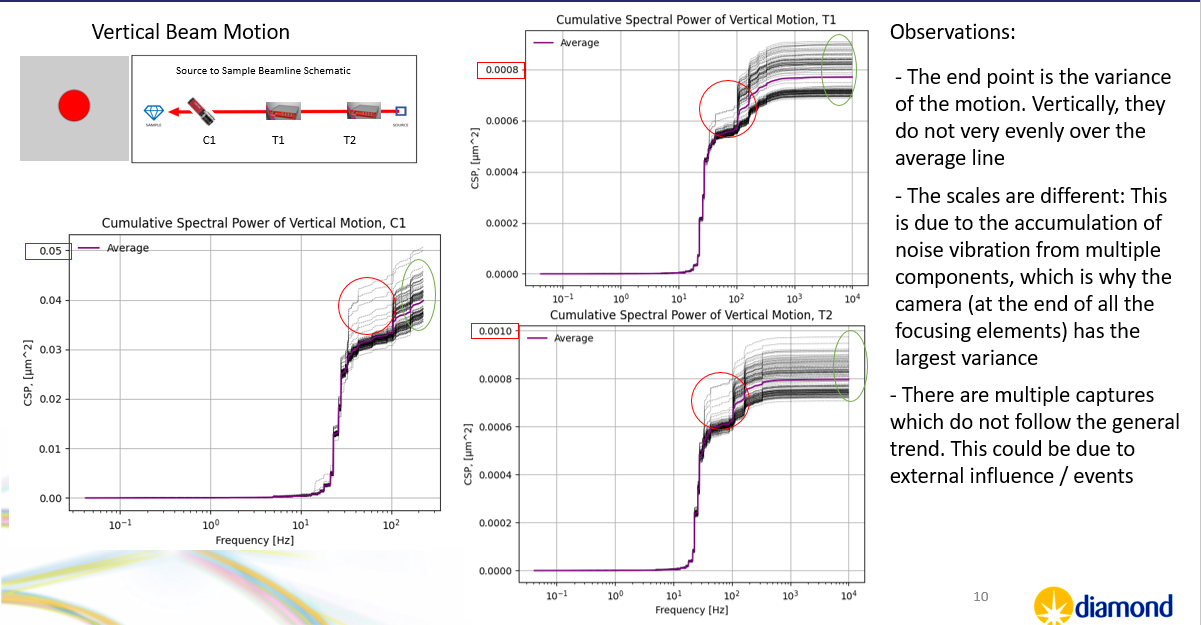
After the system was designed and build, it was tested on the I18 beamline in June and acquired 7 hours’ worth of overnight data being triggered for 2 minutes every 30 seconds. The event log and comments about problems which arose during this experiment is available on confluence [3].



*Figure 5.1: Summary of beamline data acquired during the I18 Trigger Testing*

This data was then processed and analyzed, the outcomes of which will be discussed below:





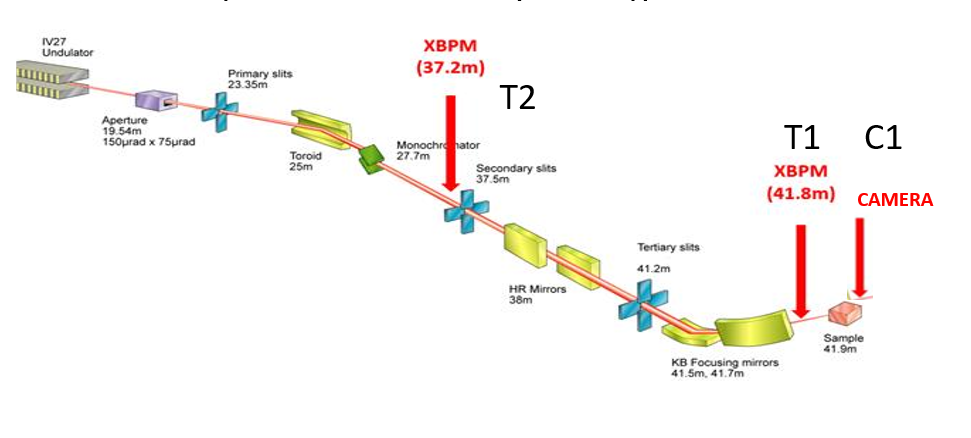
*Figures 5.2: Summary of Horizontal and Vertical beam motion analysis for the I18 experiment data*

Plotting the normalized Cumulative Spectral Power plots allow us to more easily see and determine the sources of correlated and uncorrelated vibrations

*Figures 5.3: Normalised CSP plot for the TetrAMMS and the Camera for the Horizontal motion (left) and Vertical Motion (right)*

From Figure 5.3, we can more clearly observe uncorrelated frequency steps. First looking at the horizontal motion, we can see that the red and black lines (C1 and T1) both experience a frequency step at ¬15Hz (0.07s), but not T2. Looking at the beamline schematic diagram in Figure 5.4, we can speculate that this is most likely due to the horizontal focusing mirrors, which are located between T2 and T1/C1. Furthermore, we can also observe that only C1 sees a rise at about 80Hz (0.0125s). This is not present in T1, and as there is nothing between them except the sample mount (where the camera was mounted) the motion is likely coming from there.

Looking at the vertical motion, we can observe that all the hardware show quite correlated motion and similar steps/amplitudes. This indicated that there isn’t significant uncorrelated vertical motion, and that because each hardware can produce the same/similar data, the frequency steps from the horizontal motion graph must be coming from the beamline as opposed to from any of the acquisition hardware. Furthermore, the lack of frequency step at 15Hz further supports the theory that the horizontal focussing mirrors are causing the uncorrelated motion, as they should not affect the vertical motion, which they indeed do not.



*Figure 5.4: I18 beamline schematic showing the placement of the data acquisition hardware along the beamline and the focusing elements between them*

**Overall, it can be concluded that the I18 beamline sees uncorrelated horizontal motion, most likely due to the horizontal focussing mirrors between the two XBPMs and from the sample mount**

# 6. Conclusions

The aim of the final section of this report is to evaluate how well my system met the specification criteria put forward at the beginning of the project. This will be explored in Table 6.1 below:

| **No.** | **Deliverable** | **Met?** | **Comments** |
| --- | --- | --- | --- |
| **1** | Develop a system to acquire 30s of synchronous data from a CANels TetrAMM (four channels) and a AVT Manta G235b camera (60 X 60 pixel images) at the push of a button. The acquisition systems should be triggered to begin acquisition to within 0.1 ms of each other. The system should be able to operate with difference data rate from each device. |  | The system can acquire synchronous data at the push of a button for 30s or more (and less) and they can operate with the different data rates. The degree of accuracy value (0.1ms) **could not be met** due to the hardware limitation of the Arduino itself. The most accurate trigger error is 7/6 ms between the camera and the TetrAMM outputs, and between the TetrAMMs itself, 1ms. The 1ms value is the fastest that a register can ‘synchronously’ trigger 2 pins, and cannot be improved upon. |
| **2** | A graphical user interface (GUI) should be developed to display the measured beam motion over the 30s period acquired from both the CAENels TetrAMM and the AVT Manta G235b camera. A method for highlighting correlated motion and uncorrelated motion should be developed. This should show the user if a particular motion is correlated, and seen in both measurements or uncorrelated, and only seen in one instrument. |  | The GUI can display the acquired data in various different ways and configuration in the post-experiment analysis package. Features such as windowing and the removal of DC drift have been accounted for to produce helpful graphs.  **The GUI cannot indicate whether motion is correlated or uncorrelated,** however, it can present the data in a way that makes this easy to read for the human eye: |
|  | An indication on the graphical user interface to direct users to a specific component or components that could be the cause of the correlated or uncorrelated motion. |  | This proved more difficult than it sounded and was unable to be achieved in the given timeframe. However, the diagnostics GUI has a feature to place the hardware in order on the beamline. This could be further developed to add beamline elements between the hardware for easier speculation as to which element is producing the unwanted vibration |
| **3** | The ability for the GUI to display spectral data, using fast Fourier transforms to display the 30s period acquisition in the frequency domain. Highlight the largest frequency contributions in both directions (horizontal and vertical) and indicate if this is correlated or uncorrelated between devices. |  | The GUI can display the processed data, but it **unable** to highlight the largest frequency contributions and determine if it is correlated or uncorrelated |
| **4** | A project report, detailing the process used to develop and test the system complete with instructions for the continued use of the system. |  | The report is to be used in conjunction with the confluence help guide [2] for a well-rounded understanding of how to use the system |
| **5** | A presentation of the main results should be given before the close of this project discussing the deliverables above and how well these were met. |  | The presentation was completed on 10/09/2021 and is available on confluence [4] |
| **6** | Allow the trigger system to have selectable modes: it should be able to send a one-off trigger signal, and should also be able to continuously acquire data |  | This is an implemented feature that works well and has been tested |
| **7** | Allow the continuous trigger system to be self-fixing when issue with file-saving Camera/TetrAMM network issues arise |  | This required much trial and error, but the code is now able to fix the three main errors identified that could come up during the experiment. For more information on this, please refer to Section 4 |

Overall, the project has met most of the deliverables, with the more complex tasks not being met due to lack of time within the given period. The project ends with a detailed user help guide on confluence [2], a portable hardware system, and an extendable software package that can be update with new hardware in the future. Other things that could be developed on the project include:

Software:

* Make Fourier analysis take place in "Conversion".py file such that the Fourier transform is done at the same time as the .mat files. This will be beneficial for long experiments with many files, as the graphs can be displayed quicker.
* Add other beamline components into the GUI components sketch, so you can see roughly what could be causing issues (DCM / slits / mirrors).
* Highlight main frequency contributions in GUI. Can the biggest spectral line be highlighted, and shown where along the beam path it's introduced?

Hardware:

* Replace "trailing" Ethernet cable from Arduino box with an Ethernet socket.
* Hardware button or rocker switch to send new information to second Arduino box (instead of unplugging Ethernet cable).
* User-selectable acquisition times using some LED / button display on the Arduino box, rather than having to use a USB cable to reprogram the Arduino.
* Power second Arduino via Ethernet cable / first Arduino, or via a battery, if possible.
* Tidy up boxes / install in metal boxes.
* PANDA boxes? This could do everything that our Arduinos do. Not all beamlines have PANDAs though.

# References

[1] Arduino Ethernet Shield:

<https://www.digikey.co.uk/en/products/detail/seeed-technology-co.,-ltd/103030021/5774153?utm_adgroup=Evaluation%20Boards%20-%20Expansion%20Boards%2C%20Daughter%20Cards&utm_source=google&utm_medium=cpc&utm_campaign=Shopping_Product_Development%20Boards%2C%20Kits%2C%20Programmers_NEW&utm_term=&productid=5774153&gclid=EAIaIQobChMI8a3d56uE8wIVxOjmCh1arAsGEAQYAyABEgIkfPD_BwE>

[2]: Confluence Help Guide:

<https://confluence.diamond.ac.uk/display/DIAGTECH/Synchronous+X-Ray+Diagnostics+Measurements+Help+Guide>

[3]: I18 Experiment Log:

<https://confluence.diamond.ac.uk/display/DIAGTECH/21-08-21%2C+22-08-21%2C+23-08-21%2C+24-08-21+BEAMLINE+TESTING+NOTES>

[4]: Final Presentation:

<https://confluence.diamond.ac.uk/display/DIAGTECH/15-09-2021+Wrap-up+Meeting>

[5]: Windowing Help Guide:

<https://confluence.diamond.ac.uk/display/DIAGTECH/How+to+choose+Windowing+Functions>