Synchronous X-Ray Diagnostics Measurements Help Guide

Created by Codrutza Dragu, last modified just a moment ago

This help guide is to assist the users of the summer project Synchronous X-Ray Diagnostics Trigger System, developed by Codrutza Dragu, University of Oxford (codrutza-maria.dragu@trinity.ox.ac.uk)

The following components of the system will be discussed in detail:

Software Monitoring + Analysis Package

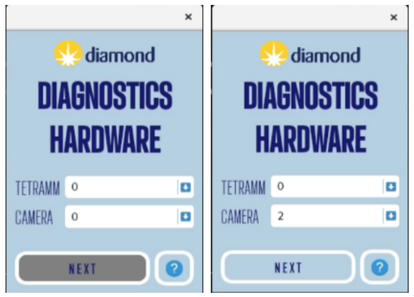
Experiment Set-up Software Package:

To begin running the experiment set up sequence of GUIs, in the python terminal, run the **Main_Experiment_Set_Up.py** script. Check the read-me for script pre-requisites.

The order of the screens appearing as generated by the script are as such:

Set_Up_Main.py

(a) (b)



This GUI allows you to pick how many of each type of acquisition software you want to use.

- **TetrAMM Dropdown Combo box:** Select the number of **TetrAmm** devices that are installed on the beamline, and whose data you are interested in acquiring.
- **Camera Dropdown Combo box:** Select the number of **Camera** devices that are installed on the beamline, and whose data you are interested in acquiring.
- **'Next' button:** Starts is disabled (as shown in (a)) until at least one of either types of hardware is selected, in which case it enable (as shown in (b)). Pressing this moves on to the next GUI in the set up secquence
- **'?' button:** Contains hyperlink to this confluence page

Set_Up_TetrAMM.py

(a) (b)

This GUI allows you you to set up the TetrAMM object's HDF section, and will acquire all the data from the QE1 section as it is in the moment you run the GUI from it's EPICS GUI.

- **The Title:** In the above case TetrAMM (1/1) – is of the format **TetrAMM (k/n)**, where **k** is the current number of the TetrAMM you are setting up, and **n** is the total number of TetrAMMs you have to set up based on how many TetrAMMs were selected in **Set_Up_Main.py**
- **PV:** Input the PV of the EPICS GUI linked to that hardware
- **FilePath:** Input the folders where the files should be saved. Note: This folder must be previously created by you and have the appropriate write access (chmod a+rwx folder_name). If the folder does not exist or the directory is wrong, it will error and not create the folder.
- **FileName:** This is the both the name of the file (eg. files will be saved as T1_000x in the above screenshot) but also considered to be the name of the hardware itself and the key for how the full data from the matfiles (produced at the end) will be accessed
- **FileFormat:** This comes pre-written, as follows the 4 digit with leading 0's format (eg. 0001, 0004, 0077)
- **Seconds:** Type the number of seconds of data will be acquired per interval.
- **Capture Num:** Automatically calculates the number of captures required to input to the HDF5 set up in the EPICS GUI. Note that if your 'Seconds' input is not valid or the PV is wrong, Capture_Num will show **'Invalid'** like in (b)
- **Next Button:** This will either progress to the next hardware set up GUI, or to the **Set_Up_Beamline.py** GUI

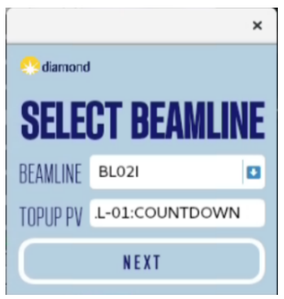
Set_Up_Camera.py

(a) (b)

This GUI allows you you to set up the Camera object's HDF section, and will acquire all the data from the CAM section as it is in the moment you run the GUI from it's EPICS GUI.

- **The Title:** In the above case Camera (1/1) – is of the format **CamerA (k/n)**, where **k** is the current number of the Camera you are setting up, and **n** is the total number of Cameras you have to set up based on how many Cameras were selected in **Set_Up_Main.py**
- **PV:** Input the PV of the EPICS GUI linked to that hardware. Note: Once the PV is inputted and you 'click off' that editing box, the script will automatically try to acquire the FPS. If it cannot do so because the PV is wrong, then the **'Invalid'** message will show up in the FPS box. If the FPS shows up correctly, be aware that we want the **Lowest FPS**. Usually, in EPICS, the fps tends to fluctuate between two values (eg. 708 and 709). We want the fps box to display the **lowest** of the 2, so if it does not, click on and off the PV box a few times until the FPS box updates with the lower value.
- **Fast Cam Button:** This button will bring up the Fast_Cam GUI to position the region start and size for the camera. This script was written by Claire Houghton.
- **FilePath:** Input the folders where the files should be saved. Note: This folder must be previously created by you and have the appropriate write access (chmod a+rwx folder_name). If the folder does not exist or the directory is wrong, it will error and not create the folder.
- **FileName:** This is the both the name of the file (eg. files will be saved as C1_000x in the above screenshot) but also considered to be the name of the hardware itself and the key for how the full data from the matfiles (produced at the end) will be accessed
- **FileFormat:** This comes pre-written, as follows the 4 digit with leading 0's format (eg. 0001, 0004, 0077)
- **Seconds:** Type the number of seconds of data will be acquired per interval.
- **Capture Num:** Automatically calculates the number of captures required to input to the HDF5 set up in the EPICS GUI. Note that if your 'Seconds' input is not valid or the PV is wrong and hence the FPS is invalid, Capture_Num will show **'Invalid'** like in (b)
- **Next Button:** This will either progress to the next hardware set up GUI, or to the **Set_Up_Beamline.py** GUI

Set_Up_Beamline.py

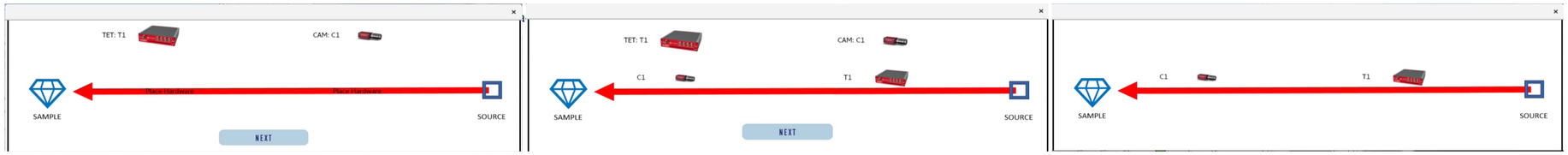


This GUI allows you to set up some beamline parameters that will be carried forward to the next GUIs

- **Beamline drop down:** Select the name of the Beamline that you are on / want to use
- **Topup PV:** The PV for the 'time left to beam topup' is automatically included
- **Next Button:** This will open the **Set_Up_Beamline_Schematic.py** GUI

Set_Up_Beamline_Schematic.py

(a) (b) (c)

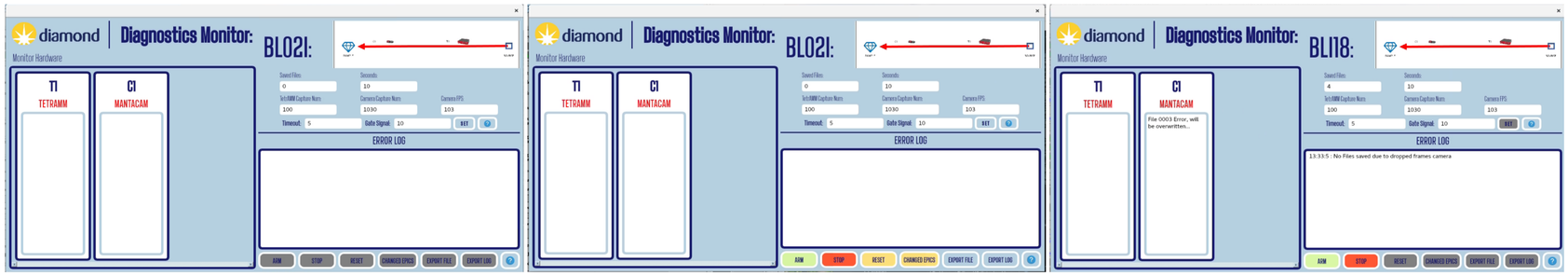


This GUI allows you to place the available hardware in order of where they should go along the beamline from source to sample.

- **Hardware Images + Labels along the top:** The hardware images along the top (shown in (a)) display the available hardwares you have set up and their names. These lables are draggable and can be placed on the **'Place Hardware'** labels on the beamline
- **'Place Hardware' labels on the beamline:** These labels are 'droppable', meaning that if one of the hardware is dragged on them, they will assume the appearance of that hardware label (as shown in (b))
- **Next Button:** This button will clear the top hardware labels and itself (as shown in (c)), take a 'screenshot' of this schematic to be saved as **BeamSchemAnno.png** in the same folder as the code is located, and automatically close this GUI and move on the next GUI

Set_Up_Diagnostics.py

(a) (b) (c)



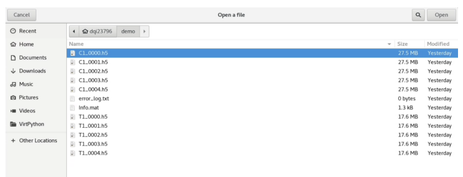
This is the main monitoring GUI that allows you to to observe the state of the hardware and whether any / which errors occurred.

- **Monitor Hardware Rectangular Modals:** The white modals with the available hardware located on the left of the screen display the personal errored files for each of the hardware (as shown in (c), where there was an error with file 0003 for the camera)
- **Beamline name and schematic:** The beamline name and schematic located on the top right of the screen were set up from the previous **Set_Up_Beamline.py and Set_Up_Beamline_Schematic.py** GUIs and display the current beamline name and the location of the hardware devices with respect to the sample and the source.
- **Saved Files**: Automatically updating counter of the number of files that have been saved so far
- **Seconds**: Displays the seconds of data that you want to acquire, as specified in the previous **Set_Up_TetrAMM.py** and **Set_Up_Camera.py** GUIs
- **TetrAMM Capture Number:** Displays the number of captures the TetrAMM has to do. This updates if you change the HDF section of the EPICS GUI
- **Camera Capture Number:** Displays the number of captures the Camera has to do. This updates if you change the HDF section of the EPICS GUI
- **Camera FPS:** Displays the FPS of the camera. Will update if you change the EPICS GUI
- **Timeout:** This come pre-populated with a value. This indicated the wait time for the cothread caget and caput. This timeout time is proportional to file size, which is in turn proportional to the number of seconds of data that is being acquired. The way that the timeout is calculated (and a good rule of thumb) is as such:
  - For time(s) < 30, timeout = 5
  - For 30 <= time(s) < 60, timeout = 6
  - For 60 <= time(s) < 90, timeout = 7
  - For 90 <= time(s) < 120, timeout = 8
  - For 120 <= time(s) , timeout = 10
  - After 30 seconds, the timeout values increment by 1 until they are greater than 2 mins, after which the value remains constant as

  The timeout value can be manually edited if needed. This could be for example, if the network is particularly slow and the one off trigger test shows that files are not saving properly.

- **Gate Signal:** This is the number of seconds which the data is being acquired for. If this changes, you can manually update this here. It is the value used to determine whether or not there was top up in the acquisition or not. Initially (as in (a)) the gate signal value and the seconds value should be the same, but as there is no seconds to update in the EPICS GUIs, if you do want to update the acquisition time, do it in the Gate Signal box.
- **Set**: Lock in the Timeout and Gate Signal values. Note: Until this is pressed at least once, the array of buttons at the bottom do not become available. This button itself becomes disabled when the 'Arm' button is pressed and the hardware is acquiring data. When the loop has been stopped however, the button is enabled again in case any updates need to be made.
- **'?':** Hyperlinks to this confluence page
- **ARM Button:** Starts the loop code. Starts the HDF5 acquisition to wait for the trigger signal
- **STOP Button:** Stops the hardware from acquiring HDF data. Note: This will stop at the end of the current. This means that if the loop is currently saving files or resetting a hardware, this will be executed before stopping. THERE IS NO NEED TO SPAM THE STOP BUTTON IT HEARD YOU THE FIRST TIME I PROMISE
- **RESET Button:** This will reset the GUI and the hardware to their initial state - ie, clear all error messages and set the increment back to 0
- **CHANGED EPICS Button:** This button must be pressed **ANY TIME AFTER ANY CHANGE HAS BEEN MADE IN THE EPICS GUI.** This to ensure that the hardware objects have the most up to date values such that when they need to be reset during the loop code (if, for example, the fps is bad) then the update values will be re-inputted to the EPICS GUIs as opposed to the old ones
- **EXPORT FILE Button:** This button **MUST BE PRESSED** after the GUI otherwise you wont be able to close the GUI. The first time you press the close on the top right, the Export File Button will turn red and the GUI will not close. If you press the x one more time without saving the file, the GUI will close. This will save a .mat file of important information for the data conversion and analysis portion of the software
- **EXPORT LOG Button:** This button will create a txt file of the error messages in the error log, to be saved in the same location as the HDF5 files, and also where the Info file is located
- **'?':** Hyperlinks to this confluence page

At the end of the experiment, the directory containing your hdf5 files should look like this:

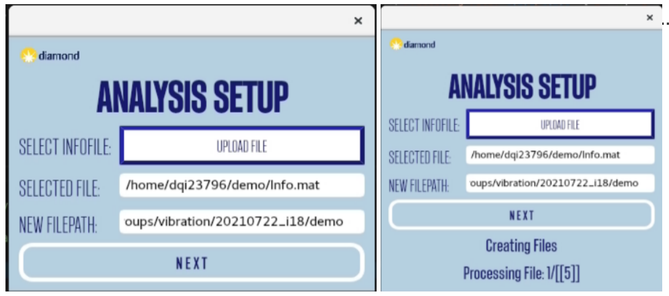


Analysis Software Package (post experiment):

This part of the software package provides the functionality of processing the data from the HDF5 files into mat files for easy access and analysis To run the Data conversion script, **Main_Data_Converter_Set_Up.py,** then to run the analysis on the .mat files, run the **Main_Analysis_Set_Up.py** script.

Set_Up_DataConversion.py

(a) (b)

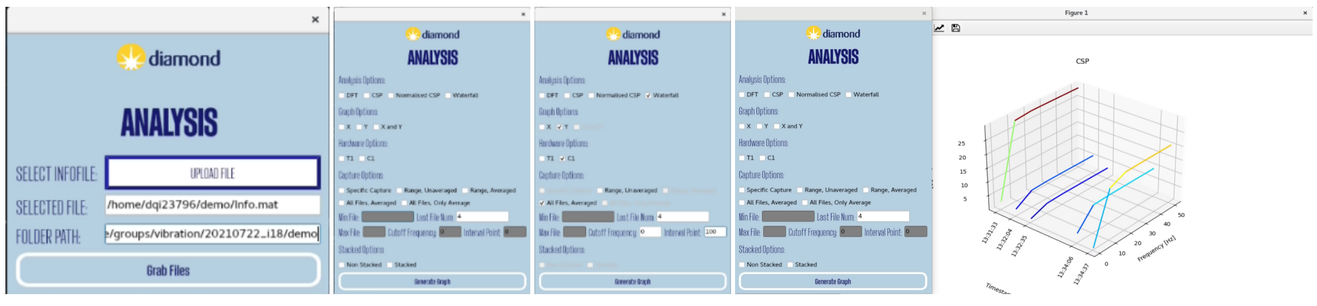


This GUI script (initially looking like (a)) is opened by **Main_Data_Converter_Set_Up.py** and allows you to set up the .mat file creation and HDF5 processing to run after the experiment, whilst also displaying the state of processing

- **Select Infofile Button:** Select the Info.mat file produced by the main diagnostics GUI at the end of the experiment
- **Selected File:** Displays the file path of the Info File as selected by the Upload Button. This is editable, so you can manually input the file path if you do not want to use the button
- **New File Path:** Type the file path of where you want to save the .mat files created from this script. Note: Make sure that the directory you want to save the mat files to already exists (it will not be created) and has the appropriate access set up. The directory must also have enough space to save the matfiles otherwise it will error and crash before it has produced and saved all of them
- **Next Button:** When this is pressed, the location of the HDF5 files from the experiment is extracted from the Info.mat file, and those files accessed. The HDF5 data sets are processed and fitted to get X and Y position data for future fourier analysis. The GUI will extend (as shown in (b)) to display the progress of the processing. The GUI is done processing when the bottom says : Processing (including XFile): Done! Then the GUI can be closed and the next analysis GUI run

Set_Up_Analysis_GUI.py

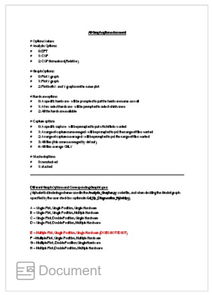
(a) (b) (c) (d)



This GUI script allows you to specify the location of the mat files produced from the previous GUI and produces a photoshop-style analysis 'toolbar' where you can select graph options. More than one graph can be produced without closing the previous graphs, and this script can be run from **Main_Analyisi_Set_Up.py**.

- **Select Infofile Button (a):** Select the Info.mat file produced by the main diagnostics GUI at the end of the experiment
- **Selected File (a):** Displays the file path of the Info File as selected by the Upload Button. This is editable, so you can manually input the file path if you do not want to use the button
- **Folder Path (a):** This is the filepath to the folder where the .mat files produced by the data converter are stored
- **Grab Files Button (a):** This will grab the .mat files from the folder (Note: the GUI will crash if that folder does not exist) and open the real analysis toolbar interface (b)
- **Analysis Options checkboxes (b):** This allows you to select the kind of analysis you want. You can only select one option at a time, and if you select certain options (eg Waterfall) other options down the GUI become disabled (as shown in (c)). Please fill the GUI in from top to bottom to allow these options to enable and disable. If you want to change something, untick them and go to the top.
- **Graph Options checkboxes (b):** This allows you to select what kind of position data you want to be displayed
- **Hardware Options checkboxes (b):** This allows you to select however many and which of the available hardware you would like to perform fourier analysis on
- **Capture Options checkboxes (b):** This allows you to select which and how many of the data files you want to display the analysed data of. Depending on which option is chosen , the max and min editor boxes will become enabled.
- **Min File (b):** If a range is selected, this will be the first file (inclusive) that you want to analyse. If the single capture option is selected, this will be the file capture number. Note: The files start from 0 and the file number of the last file is displayed in the **Last File Number Box.** Common sense suggests you do not put any number higher than the last file number in the Min File box, or the GUI will crash
- **Max File (b):** If a range is selected, this will be the last file number (inclusive) that you want to analyse. Note: The files start from 0 and the file number of the last file is displayed in the **Last File Number Box.** Common sense suggests you do not put any number higher than the last file number in the Max File box, or the GUI will crash
- **Last File Num (b):** The number of the last file. This is **NOT** how many files there are total. If this box displays the number 4 (as in (c)), then that means that files available are 0000, 0001, 0002, 0003, 0004, so there are actually 5 files as the count starts from 0, but 4 is the last file's number
- **Cutoff Frequency (c):** This option is only available when the waterfall display has been selected. It allows you to select the cutoff frequency at which the waterfall plot will stop displaying. **Putting 0 will default to displaying all the frequencies**
- **Interval Point (c):** This option is only available when the waterfall display has been selected. It allows you to specify how many nth points to take as a subset from the main data set to analyse and produce the graph for. The higher the number, the faster the processing time, but the lower the accuracy/resolution of the graph. As shown in (c), if the data set has, for example, 10,000 data points, the anlaysis will only be performed on a subset of every 100th point, and this is what will be displayed. **Putting 0 will default to using the full data set**
- **Stacked Options checkbox (b):** This allows you to select whether you want all the data displayed on the same plot (non stacked) or on subplots (stacked)
- **Generate Graph button (b):** This button will call external scripts to perform and display the analysis based on the options that you have selected. Multiple graphs can be produced each time you press the button. Note: If at least one check box from all the sectioned available options are not selected, the GUI will crash

Use the following document to check the different graph options available and how they look like:


Document

CODE_ZIP_FILES:

VirtPython.7z

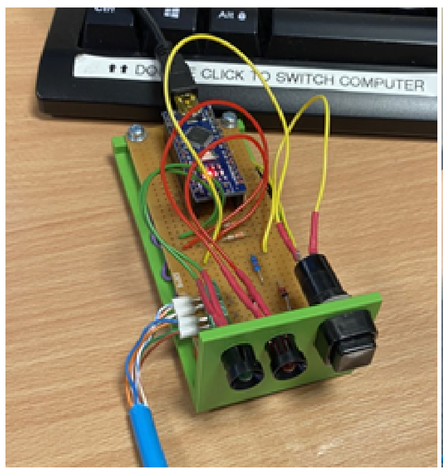
Physical Trigger Software + Hardware Package

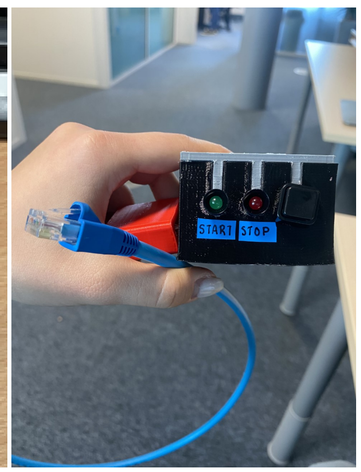
Hardware Guide:

This section will elaborate on how to use the hardware portion of the trigger system.

UserSide_Trigger:

(a) (b)



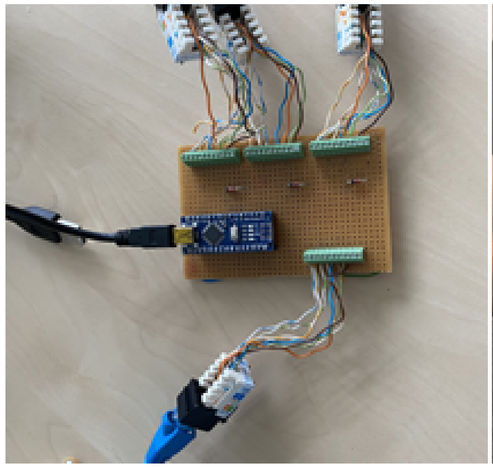


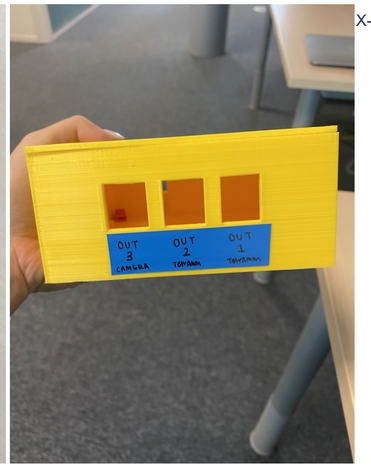
This is what the Diagnostics Team member will be controlling in the control room to be able to trigger the hardware located in the hutch

- **Push Button:** Once this is pressed, the trigger signal will be sent to the hardware. To use this in parallel with the diagnostics software package, make sure the 'ARM' button has been pressed on the Diagnostics GUI prior to trigger button press
- **Stop LED:** This indicates whether the trigger is disabled or not. If the red light is on, the trigger is not sending any signal. Note: For the continuous trigger mode (mode 1), if neither the red or green light is not, the trigger is not off, it is simply between triggers. For the one-off mode, if both the red and the green lights are on, the trigger is either sending a signal or has finished sending a signal. Push the button one more time so that ONLY the red light is on to put the trigger in disabled mode.
- **Start LED:** This indicated that a trigger pulse has been sent at that point in time. Note: For the continuous trigger mode (mode 1), if neither the red or green light is not, the trigger is not off, it is simply between triggers. For the one-off mode, if both the red and the green lights are on, the trigger is either sending a signal or has finished sending a signal. Push the button one more time so that ONLY the red light is on to put the trigger in disabled mode.
- **Ethernet Connector:** This connects to an ethernet port in the wall which has a direct patch panel link to the input of the **MidBox Arduino**.
- **USB Port:** This is to upload the arduino code to the arduino hardware, and also to supply power to the arduino from any power source once the code is uploaded

Midbox_Repeater:

(a) (b)





X-ray diagnostic measurements

This hardware intercepts the button signal from the UserSide arduino and distributes that signal to the available hardware, taking into account hardware delays

- **Input Ethernet Keystone (the single ethernet connectors in (a)):** This is where an ethernet cable connected to the patch panel port to which the UserSide arduino is connected to will be inputted.
- **Output Ethernet Keystones (the set of three ethernet connectors in (a)):** This is where the outputs to the hardware devices are connected. Note: to account for the specific hardware delays, **Output 3** should always be used for the camera, and the other two outputs for the Arduino
- **USB Port:** This is to upload the arduino code to the arduino hardware, and also to supply power to the arduino from any power source once the code is uploaded

Arduino_Converter:

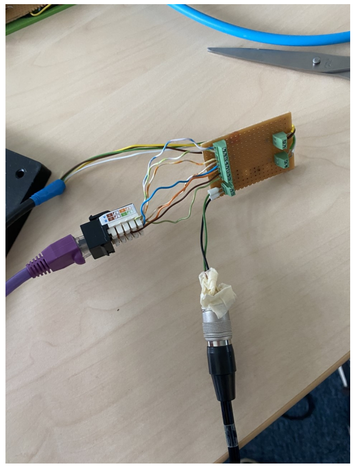
(a) (b)

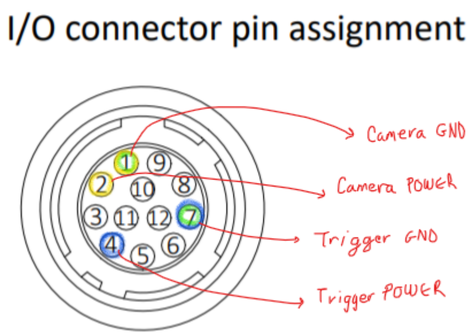
This converter connects to the Input 1 slot on the back of the TetrAMM and converts the ethernet cable to a Lemo-connector-ended-cable

- **Input Ethernet Keystone:** This is where an ethernet cable connected to the patch panel port to which the Midbox arduino output is connected to will be inputted.
- **Output Lemo Cable:** This connects to the Line 1 Trigger input of the TetrAMM

Camera_Converter:

(a) (b)





This convertor will both charge and send trigger signals to the camera using a Hirose 12pin plug

- **Input Ethernet Keystone:** This is where an ethernet cable connected to the patch panel port to which the Midbox arduino output is connected to will be inputted. The signal received from the 2 Pins will be routed to another Hirose12pin's respective pins (b)
- **Input Power Hirose 12Pin, Male:** This is where the original camera power cable will be plugged into. The power received from the 2 Pins will be routed to another Hirose12pin's respective pins (b)
- **Output Power & Trigger Hirose 12Pin, Female :** This will be plugged into the back of the camera (c) as per usual

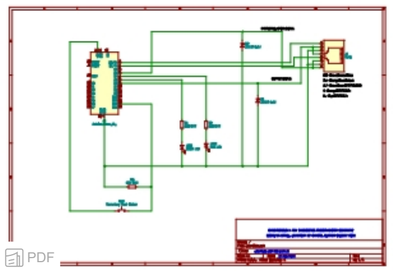
SCHEMATIC_ZIP_FILES:

KiCad Schematics:

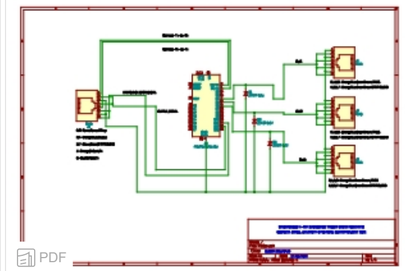
FinalSchematics.7z

PDF Schematics:

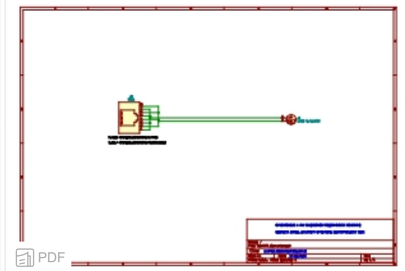
UserSide



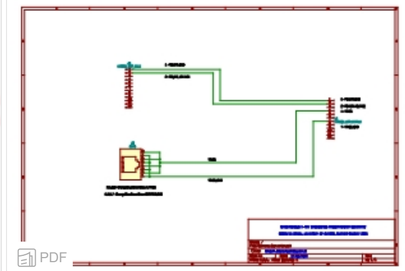
MidBox



TetrAMM Converter



Camera Converter



3D_ENCLOSURE_ZIP_FILES:

Trigger_Enclosure_Stl_Files.7z

Software Guide:

This section will elaborate on how to use the software portion of the trigger system

User_Side_Code:

(a)

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

// TO BE EDITED BY USER // --> EG. collect 10 seconds of data every 20 seconds
unsigned long interval_time = 20000; // Length of time between data collection intervals // EG. 20 seconc
// This will be transmitted to the other arduino (to communicate via serial port)
unsigned long gate_interval = 10000; // Length of time for which the trigger on signal is held (length of
// MODES
int mode = 1; // Choice of Mode: 0 -> One off trigger, 1 -> Conitnuous trigger

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

This portion of the code is editable. If anything is edited, unplug the arduino from the power and from the ethernet panel, and reupload the code. Only then plug it back in to the ethernet panel.

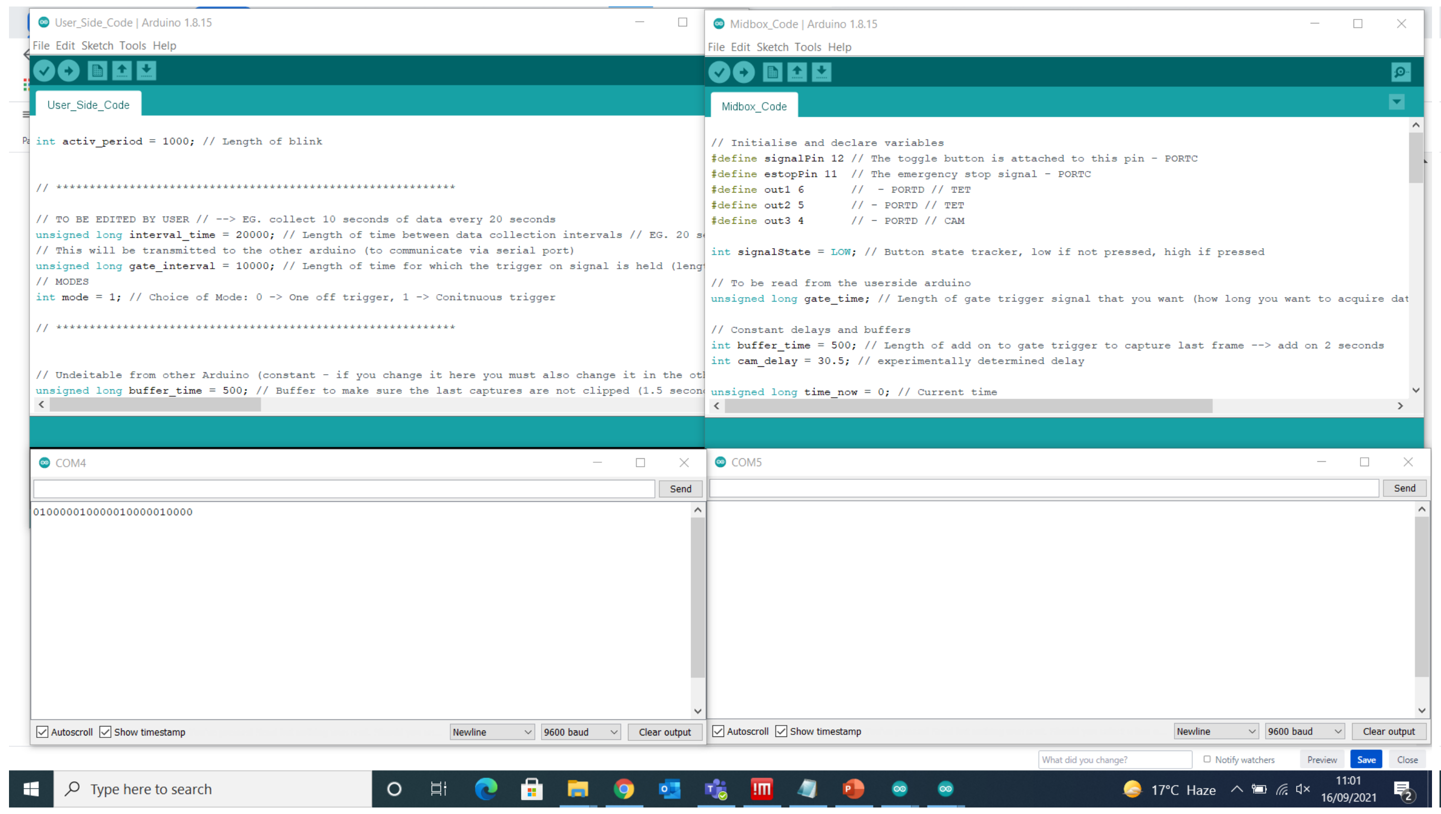
- **Interval Time:** Length of time between trigger signals
- **Gate Interval:** Length of time for which the trigger signal to trigger the acquisition hardware
- **Mode:** Select one-off experiment mode (0) or continuous trigger mode (1)

Midbox_Code:

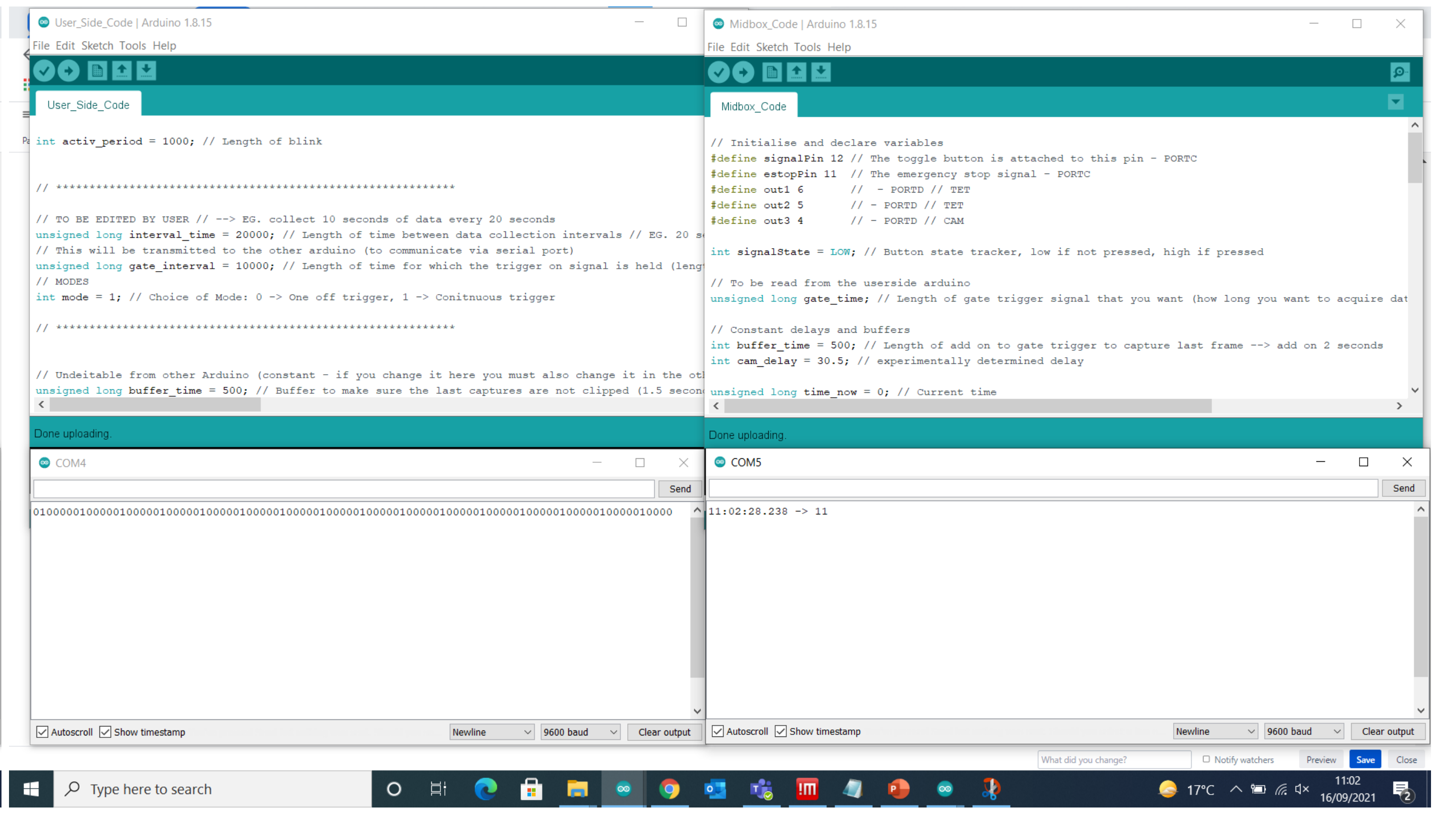
This code is uneditable, and will wait to receive the gate interval time from the Userside arduino

How To: SERIAL CONNECTION:

(a)



(b)



To Initialise the serial connection, do the following:

1. Connect both the Midbox and Userside to the computer with the arduino code, but **DO NOT CONNECT THE ETHERNET OUTPUT OF THE USERBOX TO THE INPUT OF THE MIDBOX YET**
2. Upload the respective code files to each of the arduinos
3. Move the MidBox Arduino to it's final location, plug in the power, and plug in the ethernet input connection
4. The Userside Arduino should begin sending the gate interval parameter continuously via serial as shown in (a)
5. **Connect the ethernet output of the UserBox to the MidBox's input - via the patch panel network**
6. The MidBox Arduino will send '11' across the Serial Port when it has received the parameter, and the UserSide will stop sending the signals
7. **Done!**

If you want to change the parameter while the beam is on, simply unplug the UserSide arduino's ethernet connection to the MidBox, edit the value you want, and steps 4-7 are identical.

ARDUINO_CODE_ZIP_FILES:

Arduino Code.7z

No labels