**3 Dimensional Fibre Alignment System**

**System Overview**

The basis of the alignment system is a Windows Forms based application. Windows Forms uses the .NET framework in order to generate a graphical user interface. The application is split into two main subsystems: Motor Control and Data Collection. The Motor Control sub-system is C# based and uses the libraries (or application extensions) provided by Thorlabs to communicate with the hardware. The Data Collection sub-system is Python based and again uses the manufacturer libraries to communicate with the hardware via a GPIB gateway. These two sub-systems are almost always running “simultaneously” (in reality the program is switching back and forth between the processes very quickly but to an observer this happens so quickly that it appears simultaneous) and so the application makes use of a Named Pipe to keep both processes synced correctly with another. A Named Pipe simply allows the user to pass low-level data (binary streams) between processes running independently of one another. This enables the C# and Python elements of the application to communicate with another rapidly.

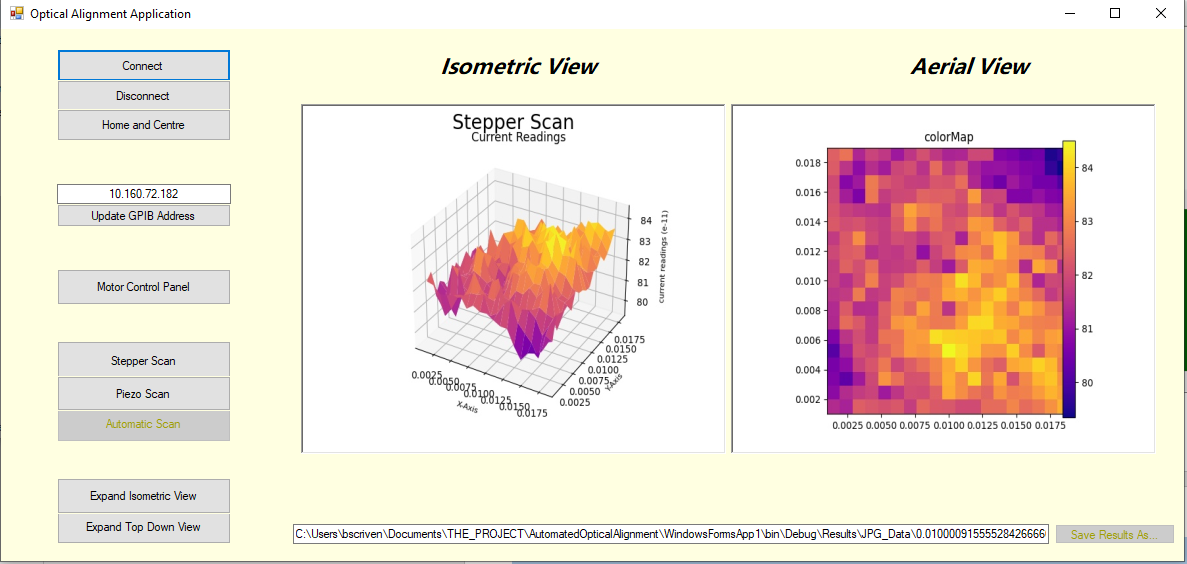
**Outlining the Problem**

The objective of this project is simply to align a light source with the active area on a PCB as quickly and accurately as possible. The light source is attached to a collimator in order to collimate the light source and thus make it easier to align.

To align the light source manually can often take hours and so automating this process will save lots of time and manual labour. The process of aligning the light source and target area is simply a matter of scanning a small 2D area and gradually refining the scan until a single high current point is identified. Once the 2D position is found the system should try moving up and down in the Z-axis to try and maximise the current output.

**Application Layout and Features**

The application contains many features to help align the system as accurately as possible and also to help with debugging if issues are encountered during alignment.

**Image 1: Main Form for Optical Alignment Application**

1. **Connect, Disconnect and Home and Centre**

These buttons are self-explanatory - The Connect and Disconnect buttons communicate with the motor controllers for the Steppers and Piezos while the Home and Centre button resets the position of all motors and then centres all of them, ready to start a new scan.

1. **Update GPIB Address**

This button is used to update the GPIB address that Python uses to communicate with the Keithley Source Meter. A .txt file is held at:

***C:\... \AutomatedOpticalAlignment\WindowsFormsApp1\bin\Debug\GPIBAddress.txt***

When the button is clicked the text within this file is replaced with the text in the adjacent text box.

***It is important that this box is updated with the correct address. Otherwise the python script will crash as it is unable to establish a connection with the Keithley. This may be improved with some sort of cross process error handling as an extension.***

1. **Motor Control Panel**

This button just launches a new form with lots of controls for operating the motors manually. These controls include stepping, connecting, disconnecting, moving to precise positions and positional feedback (for the piezos only).

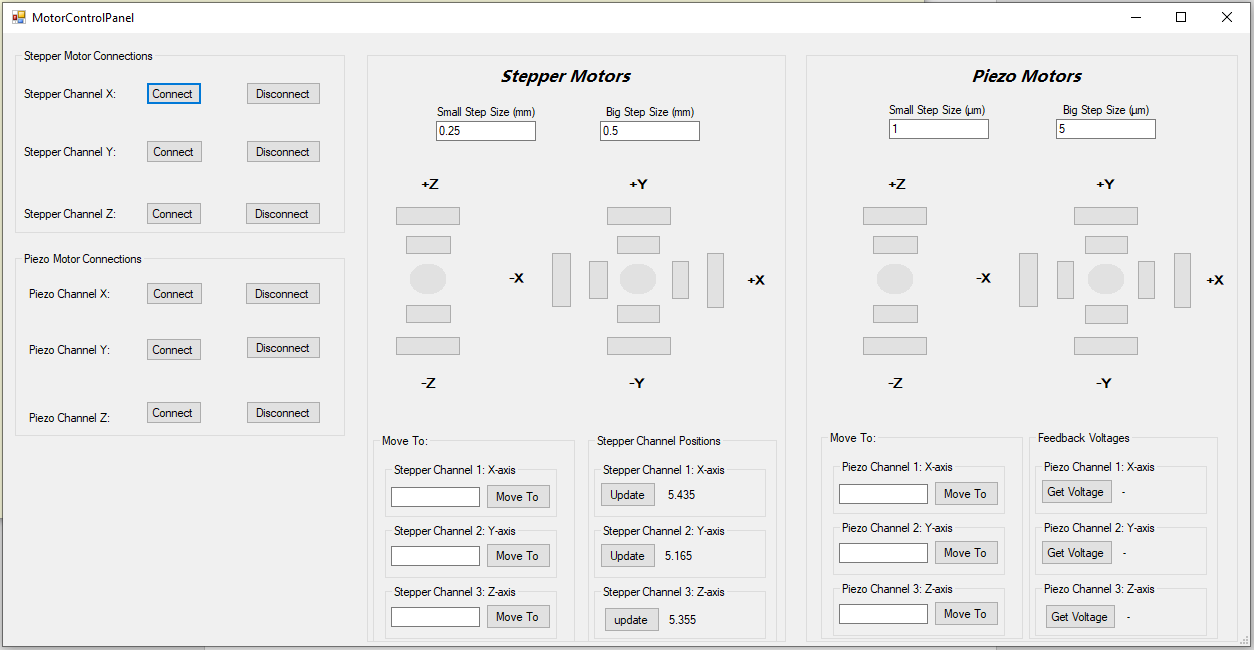
1. **Stepper Scan and Piezo Scan and Automatic Scan Buttons**

The stepper and piezo buttons just open a new form with controls and input options for running a scan. The automatic scan button is currently disabled as it is incomplete however the existing code can be accessed and updated by opening the project in an IDE which supports the .NET Framework such as Visual Studio.

1. **Expand View Buttons**

The expand isometric view button runs a python script to display the 3D surface plot using the Python Matplotlib library. This is because 3D graphics are better supported in Python than in C#.

The Expand top down view button again expands the view already shown however this one allows the user to hover over the scan and determine the X and Y motor coordinates for a point at which a high current was identified. These positions can then be used in the motor control panel to run another scan at that point.



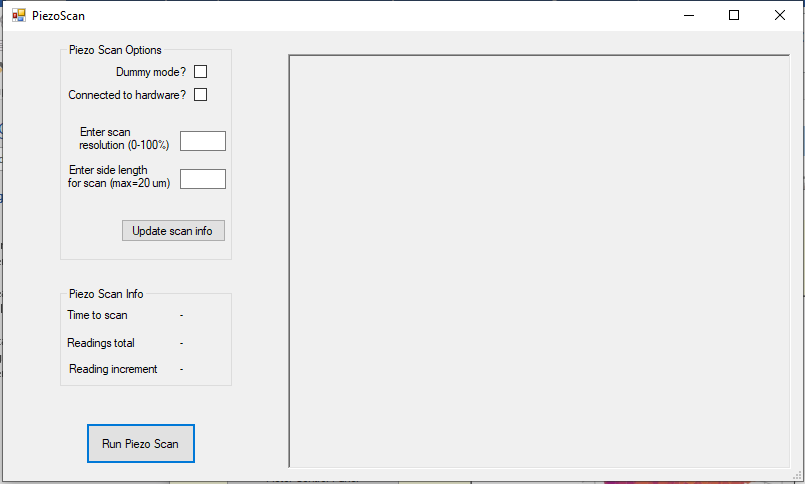
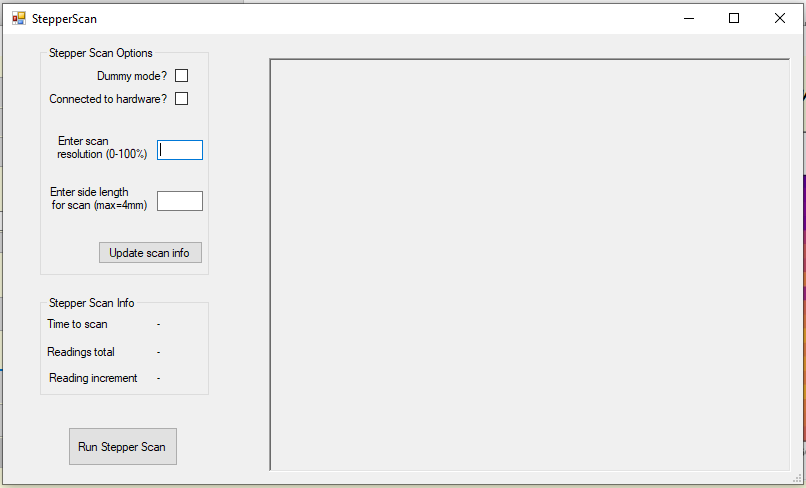
**Image 2: Motor Control Panel**

1. **Motor Control Panel**

The left of this panel allows each motor channel to be connected and disconnected individually.

The Stepper Motors panel contains buttons for moving relative to the current position. The small buttons take their step size from the box labelled Small Step Size and the same with the big buttons. The circular buttons in the centre can be used to home the channels back to their centre point. Below these buttons, there are three boxes for moving the stepper channels to exact positions and adjacent to these are labels which display the stepper channel positions. These do not have positional feedback and so the position is stored programmatically and updated when the motors are moved.

The Piezo Motor panel is identical to the stepper panel in terms of the movement buttons at the top of the panel. There are also inputs for moving the motors to precise positions and buttons for reading the position. The piezo motors do offer positional feedback.

**Image 3: Stepper Scan Panel Image 4: Piezo Scan Panel**

1. **Run Stepper Scan and Run Piezo Scan**

These buttons can be used to run a scan of a square area. The scan resolution input box varies the step size of the motors where 0% corresponds to a large step and 100% is the minimum step size of the motors being used (stepper or piezo). The side length input box lets the user choose the edge length of the square area that the user would like to scan. The 2D and 3D results windows will show where high current points are encountered however the system will not continue scanning after the given square area is finished being scanned. This feature can be useful for debugging if the automatic scan button is struggling to find the correct position.

The Scan Options group boxes also contain optional checkboxes:

* *Dummy Mode* tells the program not to start the python script automatically. This allows the user to start the python script manually inside an IDE and add breakpoints etc. for debugging.
* *Connected to Hardware* tells the application whether the measurement instruments and motors are connected or if you would like to run in it without connecting hardware. In this case the application generates sample data of integer’s incrementing by 1.

The Scan Info group boxes display information about the scan using the inputs provided in the Scan Options. The *Reading Increment* data will be exactly correct however the *Time to Scan* and *Reading Data* is only approximately correct. This is because the application runs extra measurements when a point of interest is identified.

1. **Automatic Scan**

Here, the *Dummy Mode* and *Connected to hardware* buttons have the same function as described in the stepper and piezo scan section.

In this case the application starts scanning from the centre outwards until it identifies a point where the current is greater than the specified target current. When this happens the application will run a local scan of 9 points in a square surrounding the original high point. If the average of these 9 readings exceeds the target then the motor remains in this position and begins a second scan. Before the second scan commences, the target current is incremented by 10% and the step size is reduced by 10%. This means that the second scan will start from the high current point that was identified and start a new *finer* scan of the same area. This process of increasing the target current and reducing the step size continues until either a target current is achieved or the step size is smaller than the minimum step size of the piezo motors. In either case the fibre should now be aligned very accurately.

Starting step size should be determined by the diameter of source collimator and the diameter of the active area as this will determine whether the motors skip straight over the active area when scanning.

Input boxes added but functionality not yet in place.

**Functionality**

This section will briefly explain how the application operates. When running a fixed area scan there are multiple steps. Firstly, a 2D array is generated in the pattern of a serpentine square. The C# then runs a command to launch the Python and begin communicating with the Keithley. Once a connection has been established, the motor positions begin generating. A motor position is generated using 3 variables: The serpentine pattern matrix, the step size of the scan and the current iteration within the scan. By passing these 3 variables into a MotorPosition function, an actual position is returned and the motors are incremented. After each motor increment the Python receives an instruction to take a current reading and store it in a 2D array of results. This continues until the fixed area has been scanned. Once the scan is complete, the Python script generates 2 graphs and saves these .jpg files along with the raw results array which is saved as a .csv file. The Python then terminates and the named pipe allowing communication between the processes is closed. The application then displays the 2 .JPG files within the main form so that the user can observe the results of the scan.

**Results**

The results that are produced by the Python script take the form of an extremely large matrix of current readings. The single area scan results are stored in a 2D matrix. These matrices are then saved as CSV files so that the C# application may take the CSV file and display the results in 3D. These results are also used to generate an Isometric view of the results in 3D and a top down view shaded with a heat map to aid in identifying points of high current.

All **raw** results are stored in this file space:

***C:\... \AutomatedOpticalAlignment\WindowsFormsApp1\bin\Debug\Results\CSV\_Data***

All **.jpg** results are stored in this file space:

***C:\... \AutomatedOpticalAlignment\WindowsFormsApp1\bin\Debug\Results\JPG\_data***

**References and Libraries**

The libraries and references used in the application are split between the C# Windows Form and the Python script.

The Windows Form Application uses .dll (Dynamic Linked Libraries) files provided by Thorlabs in order to communicate with the Stepper and Piezo motor controllers. It also uses various OpenTK and OpenControls libraries for displaying the 3D plots using CSV data.

These Libraries can be found within the Solution Explorer in Visual Studio. Alternatively all of these libraries are stored in the application working directory:

***C:\…\AutomatedOpticalAlignment\WindowsFormsApp1\bin\Debug***

This is also the file space from which the Python script is called and also where the Application Results folder and Measurement Instrument libraries are stored.

The Python script uses a Keithley 2400 Sourcemeter in order to take current measurements. The libraries for this meter are stored here:

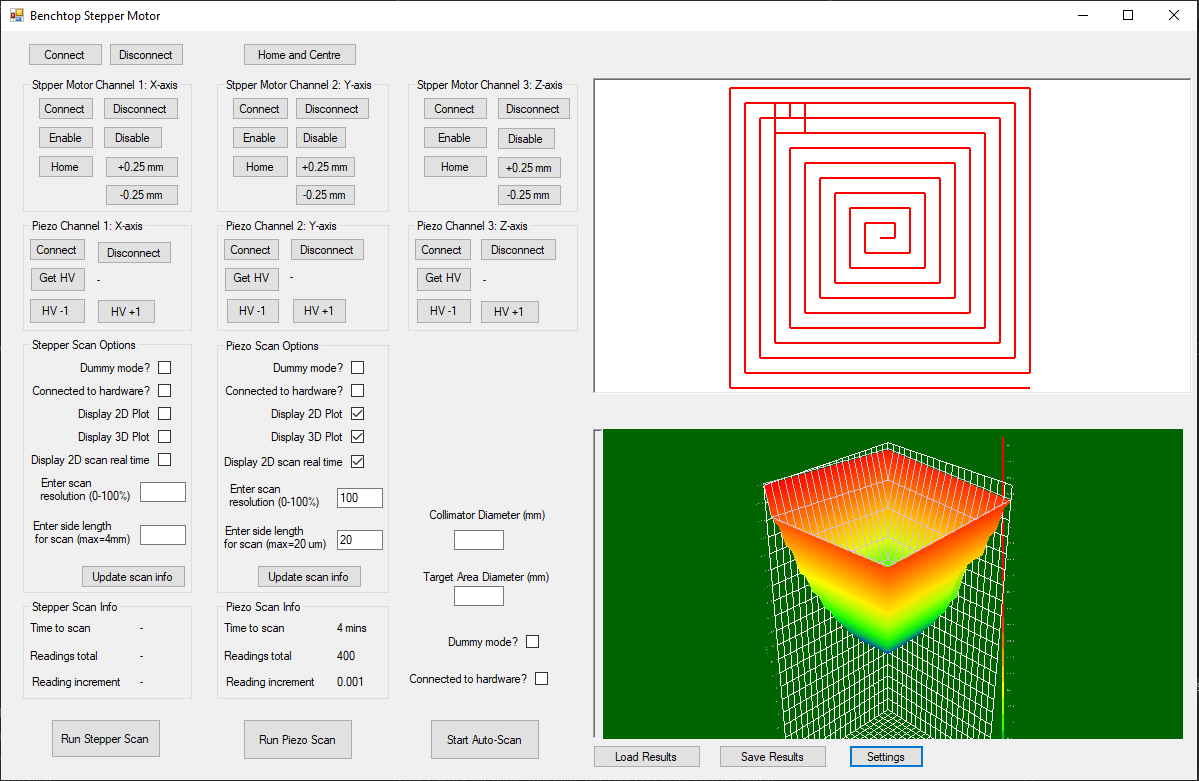
***C:\...\AutomatedOpticalAlignment\WindowsFormsApp1\bin\Debug\COMMON***

The COMMON folder also contains many other measurement instrument libraries however the application is only coded to work with the Keithley 2400 currently. This would be a good extension to the application.

**Extensions to the Project**

* Currently the C# and Python operate as separate processes within Windows. This means that when errors occur in either the Python or C# it is difficult to catch and handle the errors in both processes at once. The only way to deal with it currently is using a time out error catch however this is an inelegant solution. Ideally the host application would be able to handle all errors from both processes.
* The application is currently only setup to communicate with a Keithley 2400 Sourcemeter. A possible improvement would be add options for using different sourcemeters/multimeters/measurement devices. An oscilloscope may be a prime candidate for replacing the Keithley as this would allow the computer to poll the current readings at a much a faster rate than with the Keithley.

**Appendix**



**Image 5: Previous Development of Main Panel**