Format: some projects are well suited to keeping a lab book, either physical or electronic, which records all your daily tasks, recorded values, results, plots, thoughts, useful numbers, etc. For other projects, this documentation may come in a different form, e.g. organised notes or minutes from project meetings, GitHub commit records, or progress reports.

Whatever form you and your supervisor think is appropriate, your project documentation should:

- Include a project plan formed during the first few weeks of the project. This should include key milestones and dates.
- Be legible and clearly laid out, with dates against every entry/contribution.
- Include regular entries or updates. Most projects will make weekly progress, and so should have weekly updates to the documentation collection.

Lachlan Catto and Glen Pearce's Lab Record

2/8/24

We went through the Risk Assessment
This included emergency exits, lab hazards and risk control strategies

Today we familiarised ourselves with the microscope apparatus and SDK software. We setup a GitHub folder and organised the folders to suit the project. We commenced a project plan for how we are going to approach the project. We created a python file and created our own code from the ground by modifying the code that Matt had setup for us. A python file was created for each task, namely camera viewing and stage movement. Within the stage movement file, we were able to move the stage in all 3 directions the desired step amount and make it go to the home positions.

We started considering how the monolayer locating algorithm was going to work. Some ideas included sweeping across and moving down once it detects an edge through colour or contrast recognition; taking photos as it does so including overlap and stitching the photos together at the end.

Next week

- [x] Check FOV
- [x] Start scanning algorithm
- [x] Take photos
- [x] Save Photos
- [x] Get accurate location + coord system
- [ ] Formalise project plan and steps

9/8/24

We worked on the project plan together, brainstorming our ideas on the whiteboard. We established the key steps that need to be done in the project including creating a GUI, calibrating the origin, creating a conversion between nm and pixels, and a scanning algorithm.

Glen

I developed the fundamental backend basics of the program. I defined many useful functions such as converting the stage encoder to nm, an easy to use movement function which moves the stage to specified coordinates and waits until the movement completes. I added a basic nm -> pixel conversion for the coordinate system. This was achieved by moving the stage one camera screen length and measuring the distance with the nm coordinate system and using the known screen size to obtain a ratio. I added a basic terminal input which allows the user to specify the location of the opposite corners of the sample which enabled me to develop a simple scanning movement algorithm to sweep across the sample. Using the basic conversion obtained, I was able to ensure a slight overlap of the images when scanning the sample and I adjusted the algorithm to take images along the path. I ran into an issue where images of sample were skipped when changing direction, this was fixed by adding additional image acquisition step edge cases to the algorithm. I then added an additional thread which stitches the images together into one large image as the frames are passed in. This is quite slow and will need to be addressed next week. Maybe a different multiprocessing or more efficient overlay method is required. I also ran into an issue with the images not stitching correctly, however this was just do to a 90$\degree$ rotation of the camera which was fixed by adding a counter rotation step. It would be good if this was added as on option to the calibration window. This all resulted in the first basic working implementation of the backend code, allowing us to scan a whole sample easily. Once this implementation is refined we can move on to basic image processing to identify monolayers.

Lachlan

I started designing and coding the GUI for viewing the image and controlling the stage. I spent today familiarising myself more with git including how commits work as well as Tkinter which is used for creating a GUI. I learned how to produce frames and tabs in a GUI and display images, text and buttons. I then implemented what I learnt to display a test image of a monolayer in the GUI as well as a few buttons which don't do anything yet.

Next week I will continue working on the GUI, creating buttons used for navigation on the image and text displaying the position that the image will be in based off its locating from ThorLabs. I will also create a calibration tab that is used for manually calibrating the start and end points for the sweeping algorithm.

Next week

- [x] Formalise Project Plan
- [x] Speed up image stitching
- [x] Finalise basic implementation
- [x] Fix disgusting algorithm code
- [x] Offset coordinate system such that pixel coords start at 0
- [x] Predefine canvas size for stitched image
- [ ] Create GUI
- [x] Add movements buttons to GUI
- [x] View live position on GUI
- [ ] Add camera rotation calibration
- [ ] Update License
- [x] Fix color conversion

16/8/24

We created a more detailed project plan

Glen

This week we wrote up a detailed project plan and discussed the timeline. I then continued to work on the backend code where I added detailed comments to the main code. I then modified the image stitching and generation code to place images more accurately and blend them better. I fixed the color conversion and made the algorithm function much neater and professional.

Lachlan

I continued working on the GUI, adding buttons for movement and labels for viewing live positions in nm. I attempted to add a zoomed-in photo that you could move around using crop and resize functions in PhotoImage. I successfully created an image that was zoomed in to the desired amount, but the buttons I made did not move the viewing position of the photo. Part of the problem was the image was only called once but never updated. I had to add an infinite loop that constantly updated the photo. This still did not work but I used this as a solution to the live position text. I had a similar problem with the live view not displaying so I created an infinite loop through a function that called itself after 100ms. This function was called once to initiate it. I coded on my personal computer so the code did not immediately work when we transferred it to the desktop connected to the ThorLabs equipment and tested it on that. We were quickly able to identify the errors and missing parts to complete the code.

Next Week

- [ ] Add camera rotation calibration
- [x] Add live camera view to GUI in main and calibration tabs
- [ ] Start on a basic flow for the GUI
- [ ] Update License
- [x] Post processing with OpenCV
- [x] Create contours with OpenCV
- [x] Monolayer Class
- [ ] Speed up processing

23/08/24

Glen

This week I managed implement the post processing of the stitched image to enable us to identify monolayers. I added a monolayer class as to have a neat way of storing the data, this also contains functions to assess the quality of the monolayer as they are identified. This class also stores images of each individual monolayer so that they can be assessed visually later. One issue I had with the post processing for the monolayer detection was the conversion to greyscale. To be efficient with detection I wanted to bias the greyscale such that monolayers (mostly red) would appear brighter than everything else, particularly the green background. The simple approach would have been to just consider the red chanel of the image when converting to greyscale however objects that were white in the original image (such as other debris) would be just as bright. To account for this, I introduced a negative offset for any green and to a lesser extent blue pixels, clamping the result to the allowed range of 0 to 255. This meant that areas with significant green or

blue components would appear darker on the greyscale image, allowing for more contrast against the monolayers for easier detection.
Additionally, this week I also spent time refactoring and speeding up the image stitching and processing code as this was a major sink of computation time and resulted in the program taking a long time to run. To address this I implemented several steps, firstly removing excess image saves, next enabling concurrent processing of images, using some masking techniques and finally switching some of the larger image calculations to PyTorch which allowed for GPU based CUDA acceleration. All in all this reduced the image stitching computation time by approximately 100 times. I also changed the saving and post processing steps to use scaled down versions of the canvas, speeding up the code by a further 60 times at least and saving a large amount of storage space.

Lachlan

On the GUI I created a canvas which displayed a live view of the image from the camera. I ran into image queueing and threading issues so I fixed this by changing where the threading was being called. I tested everything in a test GUI file before trying to implement it into the main GUI file. One thing I was able to successfully do in the test GUI was move around a stock photo with buttons. This was not working last week because of threading and garbaging issues. I implemented a slider which will be used for changing the zoom and a 360 degree interactive wheel which will be used for rotating the camera. I had to change everything to be inside a class so it is easier to call functions that will update the windows such as the wheel.

Next Week

- [x] CAMERA SETTINGS (basic)
- [x] Make move and wait (& relative) generalised for z axis too.
- [ ] Add pixel -> location function
- [ ] Fix Monolayer class image color conversion in quality functions
- [x] Add move specific distance function
- [x] Added a goto monolayers
- [ ] Add Autofocus
- [x] Speed up
- [ ] Table of statistics
- [x] Add a thread which is given the canvas and scales it down during image processing and ~~displays it~~
- [ ] ~~Threaded final image save~~
- [x] Add a save and downscale image function
- [ ] Stage accuracy
- [ ] ~~Save file system~~ GUI
- [x] Wheel for camera rotation calibration
- [x] Z position slider for focus calibration
- [x] Enterables for position change on main tab
- [ ] Integration with main algorithm
- [ ] Pop up cat image

30/08/24

Glen

This week I tested my speed upgrade changes as well as generalized some of the functions within the code. The camera parameters can now be set via the software. I had issues with modifying the camera rotation as I had to change one of the classes given by the example documentation. However this didn't prove too difficult. I managed to create a function which converts the coordinates of the monolayers in pixels back to the stage coordinates, allowing us to find particular monolayers after we have processed them. This will probably need more refining next week. We also modified some of the functions such that we could run them without threads by removing the waits. This could be generalised into one function which is a next week job. I did encounter an error where the stage would occasionally move to unexpected locations on the opposite side of where it was currently processing. I think this is fixed (hopefully) otherwise it may be an issue with the stage boundaries. It would be good to keep an eye on this. Many other minor fixes and cleaning of the code. Next week will be focusing on making these last few changes and working on the lab report. From there we can finally merge our code and begin on the next lot of goals and improve the usability.

Lachlan

This week was focused on usability of the GUI. I added several features to the main and calibration tab. These features included a usable wheel for rotating the image from the sensor, buttons for moving around in the live image and changing the focus, and enterables for changing the position of the live image in nm. The rotation of the live image from the sensor produced an actual rotate image on the window. This was different to what was expected. I expected the rotation to physically rotate the camera sensor and give a different angle of the image but instead it just rotated the digital copy of the image already captured. I was able to implement these changes with the use of the bind() command in Tkinter, which I learned to use for the buttons and enterables.

One issue we had was a really laggy window, regardless of whether the stage and thus positions were being changed. The issue was two versions of the live image under the LiveViewCanvas() class was being produced by the code at the same time, but only one was ever being displayed (two images for two different tabs). The way we overcame this was checking which tab was in current use and only displaying the image on that tab.

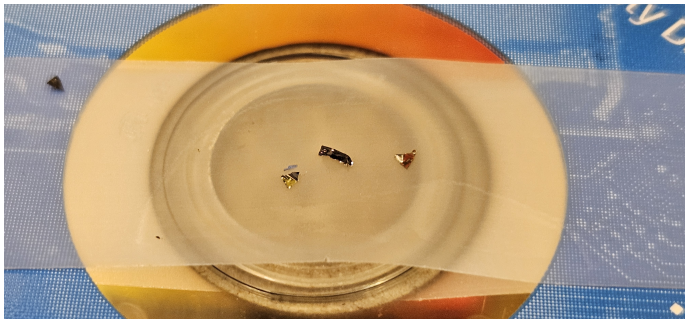
Next Week

- [ ] Lab Report Drafting
- [ ] Camera settings (advanced)
- [ ] Have mm, um, and nm option for position on main tab
- [ ] Create better framing for calibration tab
- [x] Check: Test enterables for X,Y,Z
- [x] Camera settings (advanced)
- [ ] Autofocus
- [x] Check stage coords function (more precise)
- [x] Generalise move_(and/not)_wait
- [x] Progress bar

06/09/24

Lachlan & Glen

Today we combined the algorithm and GUI code. The GUI got a rework to have a theme and layout. Added a progress bar, live stitched view, and results view. We ran into an error with the corners which was caused by the end values being larger than the start values, this was introduced when we make the corners defined through the GUI and was later fixed.
We also did mechanical exfoliation to obtain hour own samples and tested our code, identifying issues and fixing it. We didn't observe any monolayers on our samples. We need to focus on getting the exposure automated for next week.



Next Week

- [ ] Focus Plane
- [ ] Exposure and Gain
- [ ] Workflow
- [ ] Plain Camera Tab
- [ ] Scale Bar
- [ ] Monolayer Results List
- [ ] Postprocessing calibration and view
- [ ] Cat Picture
- [ ] Read the filter setting

Later

- [ ] Save file so we can load previous results.

13/09/24

Lachlan & Glen

Today we spent most of the day working on the lab report. We focused on the relevance (/introduction) section and the methods. We also added images of our first real monolayer detection test on an actual

sample. We moved and rewrote sections of our report in LaTex (Overleaf) and included references and contents etc.

We also performed our first test of the software on a real sample. This detection went quite well apart from needing to adjust values in the code to account for a different lens since we have not yet included this in the GUI. The code sucessfully scanned the section of the sample and identified several monolayers including he main one of interest.

23/09/24

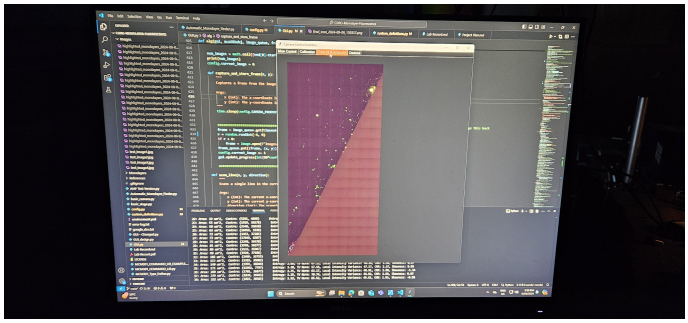
Lachlan

The very important cat pop up image was missing from the main GUI tab so that was first on the agenda for the week. To analyse the quality of the monolayers, certain statistics are use for each identified monolayer from the algorithm. I created histograms of the obtained statistics to display the distribution of area and quality of the monolayers for analysis. It was relatively easy to integrate matplotlib into tkinter, however, I kept running into a text scaling issue where matplotlib would change the global text fontsize. This was difficult to resolve and whilst I was able to fix it several times, it kept creeping back. It could not be implemented into the main GUI without having to entirely rewrite the code.

Glen

- Redid layout for live camera
- Added camera adjustment boxes
- Added Results Treeview and go to
- Redid much of the layout
- Made the goto positions work for X, Y, Z individually

Image of a test scan done prior:



24/09/2024

Lachlan

Calibration and image viewing were the focus for today. Since different monolayers could produce different fluorescence colouring, calibration for monolayer post processing was required. I embedded a colour picker for the expected monolayer fluorescence to be used for post processing. This was done in the test GUI before being implemented into the main GUI. I also created a scalable and pannable image in the test GUI which will

allow the final stitched image to be zoomed and panned to the desired locations. One issue with the zoomed image is the sheer size of the file. Since the resolution is so high, an image pyramid was created to reduce lag.

Glen

- Added automatic exposure
- Refined the Monolayer Colour detection
- Reformatted calibration tab
- Enabled rerunning of post processing
- Added lens calibration and made that nicer.

To implement the lens calibration I used the data supplied from the camera to determine the pixel dimensions. From here the nm to px conversion factor could be determined by dividing the size of the pixels by the magnification factor. A box to enter the magnification factor was added in the calibration tab.

25/09/2024

Lachlan

To provide some motivation for the design of the image zooming, I created a second option for how images are zoomed in through hovering over the larger stitched image and a zoomed in image displaying where the mouse is located. I then attempted to initialise the first zoomed image option to become unzoomed without success. The image pyramid and tiling was proving to be difficult to set the desired initial zoom. I created a double click option to return the position on the zoomed image which will then be used to move the stage to the desired location for live viewing. This was based on the distance of the mouse from the center of the canvas. I did run into some positioning issues from coordinates and zoom factors but I resolved these by changing the mouse click to give the canvas position, not the image pixel location. This will be implemented into the main GUI to move the live image in the direction and distance of the mouse click from the center of the image canvas. Today I also commenced drawing a scale for the live view image to tell us the true distance of the image being shown.

Glen

- Autofocus
- Fixed Auto Exposure
- Many bug fixes including the camera not appearing
- Made the autofocus adapt to the zoom

The main goal for today was to implement auto focus routines. To achieve this I first needed to find a good quantitative measure of the image focus which was exposure invariant, ignored image noise, and was sensitive to small changes in focus. After several tests I settled on the Power Spectrum Slope method. This involved taking the 2D fourier transform of the image and computing the magnitude of each frequency component to get the Power Spectrum of the image where high frequency data corresponds to sharp edges. The radial components were averaged to remove the radial dependance which also contracts the noise. The log of frequency was then compared against the log of power to obtain a linear relationship which a line was fit too. The slope of this line acted as our focus measure. The shallower the line, the greater the abundance of high frequency components and thus the sharpness or focus of the image. This magnitude of this value was inverted such that higher values indicated better focus.

Changes were also made to the auto expose routine, increasing the accuracy and speeding up the process. The changes included limiting the proportional increase/decrease in gain/exposure to 20% per step as to avoid overshooting and adding extra criteria for when the exposure is way too high or low. It also now tries to converge to the closest allowed value in the image intensity (exposure) range.

TODO:

- [x] Disable autofocus button when focusing
- [ ] Allow adjustment of other config parameters in extra window
- [ ] Refine quality parameters
- [x] Begin tidying

26/09/2024

Lachlan

We showed Matt the progress of our project, highlighting all the new features we added. I continued working on the scale showing measurement similar to a ruler but adding a dynamic feature so it will change based on the scale factor of nanometers to pixels. I adjusted the tick spacing and units based on the zoom. All that needs to be done with this is to implement it into the main GUI with scale conversion based on the zoom of the microscope and positioned below the canvas appropriately.

Glen

I added an advanced parameter option allowing the configuration of several behind the scenes parameters in the GUI. Additionally I refined the focus and exposure routine and bug fixed. We found there was a major issue with the software becoming unresponsive at large scan sized. This is almost certainly due to the large image handling. We already process the images down but I believe that downscaling the images earlier in the process or using the camera hardware binning may be a fix for this. This will be next week.

27/09/2024

Glen

- Speed
- Binning
- Rotation of view

Today I did a lot. The main issue I needed to solve was identified last week and was issues with computer power and memory for large scale scans. To combat this I tried to include hardware binning however the thorlabs camera does not provide a simple way to do this without impacting the colour processing. As a compromise I (with the help of ChatGPT, particular when optimising the performance) wrote custom software binning functions which would scale down the live image before it was passed to the other various functions. This was achieved by a combination of summing and averaging groups of nearby pixels to produce an image of lower resolution and higher exposure whist still preventing excess noise. This worked very well and in combination with some other minor changes the code was much faster, more responsive and CPU/RAM efficient. This took the majority of the day. I then added an option to rotate the live views by multiples of 90

degrees such that the camera lined up with the microscope. This required having to adjust the movement code to work at any rotation. This was particularly difficult and I ended up needed to use formulas incorporating modulo arithmetic and lists of direction coefficients for different rotations.

04/10/2024

- Backup Image
- Better exposure binning
- Testing and Bug fixing Today was mostly testing and many various bug fixes from the other week. This included adjusting the magnification logic to work again with the new resized images and enabling and disabling the exposure binning depending on the light levels. There was a glitch in one test where the code stopped unexpectedly half way through. As a simple backup measure added an image that is occasionally stored partway through a scan.

10/10/2024

Glen

- Redid the LICENCE

11/10/2024

Glen

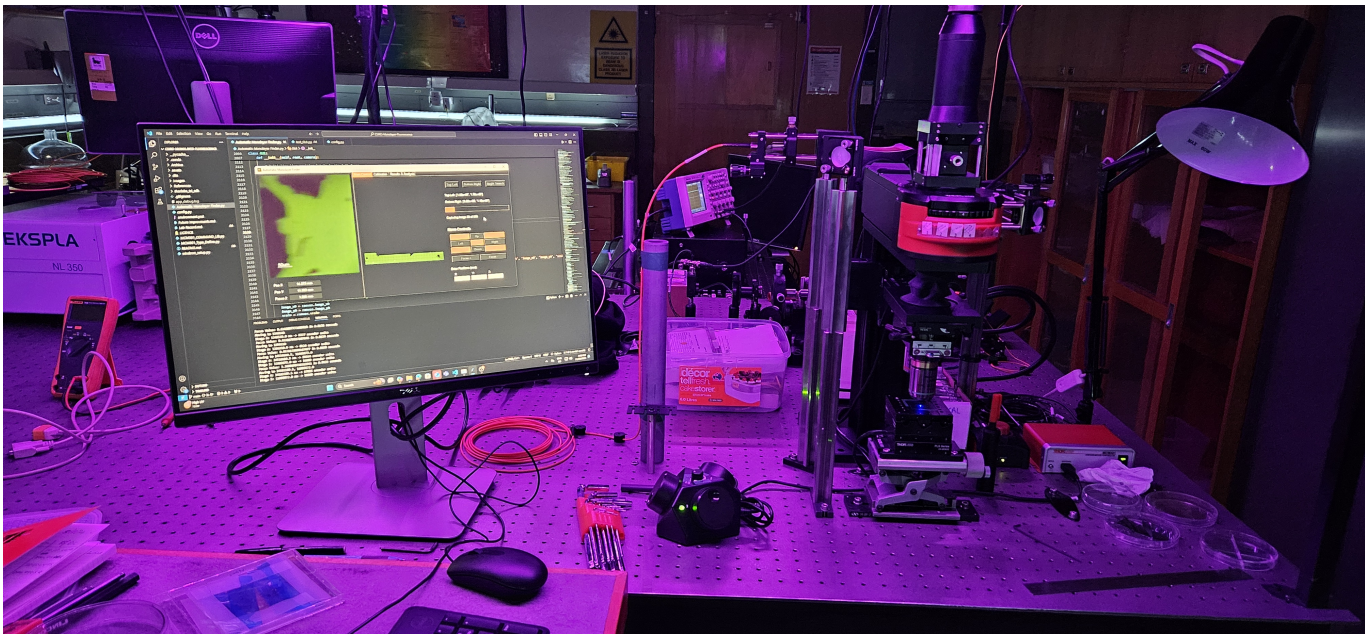
Wrapping up the project, bug fixing and implementing a final few features an getting results. Matt added some extra move buttons and a scale. Final things I worked on included quick bugs fixes, numbered monolayers (this involved calibrating for the view rotation again) and rotating the saved image. Additionally implementing an adjustble file path for the image saving. I also wrote down ideas and future improvements to share with Matt and the team. Finally we used the software to gather results.

Lachlan

One thing to implement for usability of the program was a save button. The final image needed to be saved so that the material could be easily located in future. The images were already being saved, I just needed to implement a browsing button to choose the folder which saved the final image and other various images. I ran into the problem of requiring Ghostscript to be installed to save the canvas images. I solved this by using the save file function already found the code and replacing it with an askdirectory function instead. I also completed the instructions for the GUI after all the updates had been made.

This concludes the 2024 AMF project at CSIRO!

The final setup with our code:



Checklist of action items for the last few weeks

- [x] Add cat in a Santa hat
- [x] Add camera parameter adjustment (done on backend, needs GUI)
- [x] Add a focus routine
- [x] List the located monolayers and stats and add an option to take you to a specific one (done on backend, needs GUI)
- [x] Different lens calibration
- [x] Rerun Results Processing Button
- [x] Monolayer detection settings, so we can calibrate the detection for different materials (done on backend, needs GUI)
- [x] Autofocus adapt to zoom
- [x] Disable autofocus button when focusing
- [x] Allow adjustment of other config parameters in extra window
  ~~Add a threshold slider for monolayer detection~~
- [x] Allow zoom on the results image (and click to go to location)
- [x] Add more instructions to the GUI and redo layout as needed
- [x] Add an intuitive click to move on the live view image
- [x] Tidy up our directory and to make it neater
  ~~- List the connected devices in the calibration tab (done on backend, needs GUI)~~
- [x] Add automatic focus adjustment (every n frames)
  ~~- Select Camera~~

- [x] Rescale stichable image
- [x] Make larger ML outline
- [x] Give the window a logo and ~~package it into a .exe if possible~~
- [x] Redo environment.yml
- [x] Instructions in the README
- [ ] Add advanced parameters
  ~~Refine monolayer quality parameters~~
- [x] Documentation
  ~~Comment GUI~~
- [ ] Add a scale to the results (&live?) image

For others:

- [ ] Add a save and load for settings etc.
- [ ] Check all relevant buttons get disabled during potentially colliding processes
- [ ] Stop button
- [ ] Autofocus bias towards the monolayer colour (focus when the color is found and determine if there is something to focus on using colour instead of intensity)

GET RESULTS FOR REPORT

Things to fix: (DONE 4/10/24)
Still Jumping to random other parts
Still occasionally doesn't load camera
The corners need to be labeled so that it starts on a focusable part
FASTER!