STATENS
NATURHISTORISKE
MUSEUM

KØBENHAVNS
UNIVERSITET

Workflows for Digitization of Insect Collections

Internship at the Natural History Museum of Denmark

Final Report
By: Roberta Hunt

15 ECTS Credit course
Department of Humanities
UNIVERSITY OF COPENHAGEN

SEPTEMBER - DECEMBER 2018

Abstract

Digitization of natural history collections has become of increasing interest in the past few years as image analysis techniques progress and as the technology to store large quantities of data becomes cheaper and more accessible. Many museums desire to open their collections to researchers and the public all around the world. Doing this requires two major components: digital versions of the collections including auxiliary data about the specimens, and a server to host the data. Here we present a prototype digitization station for imaging insects created for the Natural History Museum of Denmark and our choice of IIIF server to allow access to the data. The prototype digitization station includes the design of an insect tray, the automatic control of the camera height to allow focus stacking of the images, and a user interface for the digitization officers. It currently has a digitization rate of 8 specimens per hour on average, with a theoretical maximum of 18 specimens per hour. This is a first attempt, and is not yet state of the art, nor indeed finalized, but may help other museums in planning their digitization projects.

Acknowledgements

Thanks to Martin Stein for sharing all his knowledge about specify and the collections with me, and being a sounding board for ideas. Thanks to Kim Steenstrup Pedersen, Anders Drud Jordan and Isabel for all the brainstorming and guidance. Thanks to Daniel Ashbrook for taking the time to advise us about 3d printing.

Colophon

This thesis was typeset with $\LaTeX\,2_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Contents

List of Figures

List of Tables

1 Introduction

The natural history museum of Denmark recently began a project to digitize their insect collections and digitally open the collections to the public, starting with their butterfly collection which includes approximately 2 million butterflies. The first step of this is to create a workflow where an individual specimen can be imaged alongside its labels and added to the museum's database, specify, as an individual specimen.

In this introduction section I will briefly introduce the museum's current systems and practices as they relate to this internship.

In the project requirements section I will briefly explain what the project desired of different systems. These requirements were generally given to me by my supervisors, or technical experts who work with the databases / specimens.

Then in sections 3-4 I go into the details of the design and implementation of each of the project requirements, and how they were solved.

In section 5 I discuss the preliminary results of the system.

Finally, in section 6 I summarize the results and workflows, and discuss what has yet to be completed, and possible next steps to improve the digitization workflow.

1.1 Insect collections at the museum

The Natural History Museum of Denmark houses millions of specimens from around the world. Most of the insect collections are housed in drawers like the one shown in Fig. 1.1 below.

Each insect specimen will typically have some metadata associated with this, and this is kept with the specimen by putting the labels on the needle beneath the specimen, as shown in Fig. 1.2. The location of the metadata beneath the specimen itself makes it difficult to quickly and easily transcribe or read the specimen labels.

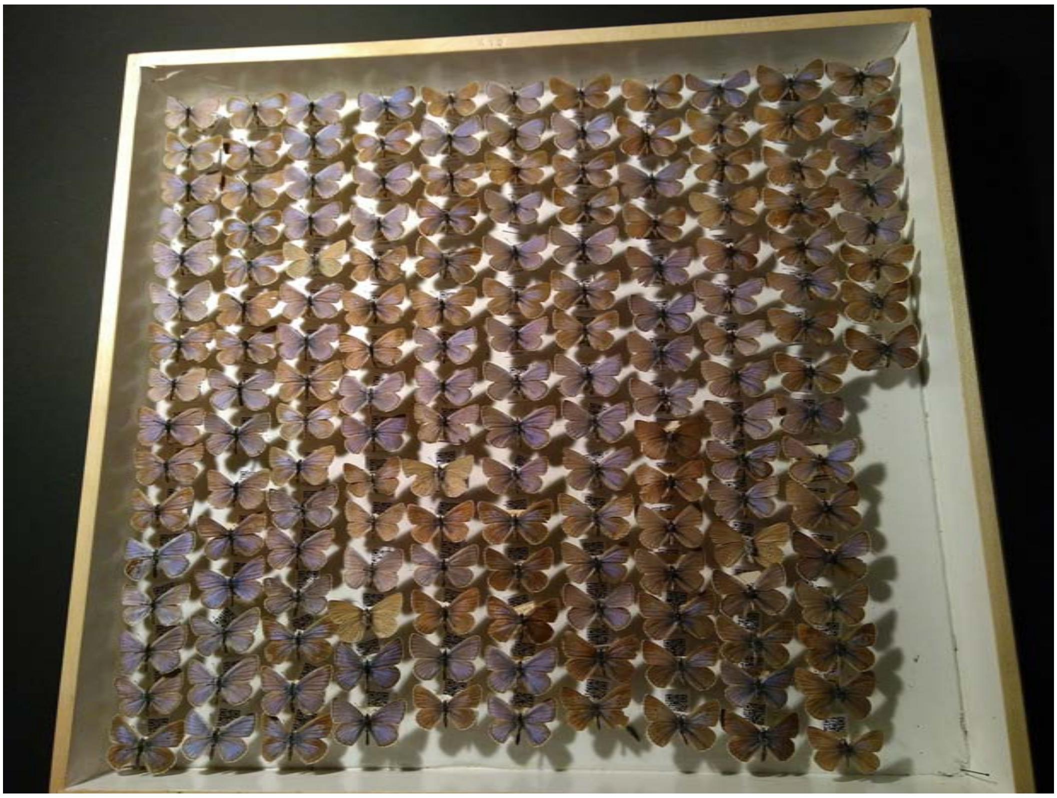


Fig. 1.1.: Example butterfly drawer containing over 100 butterflies.

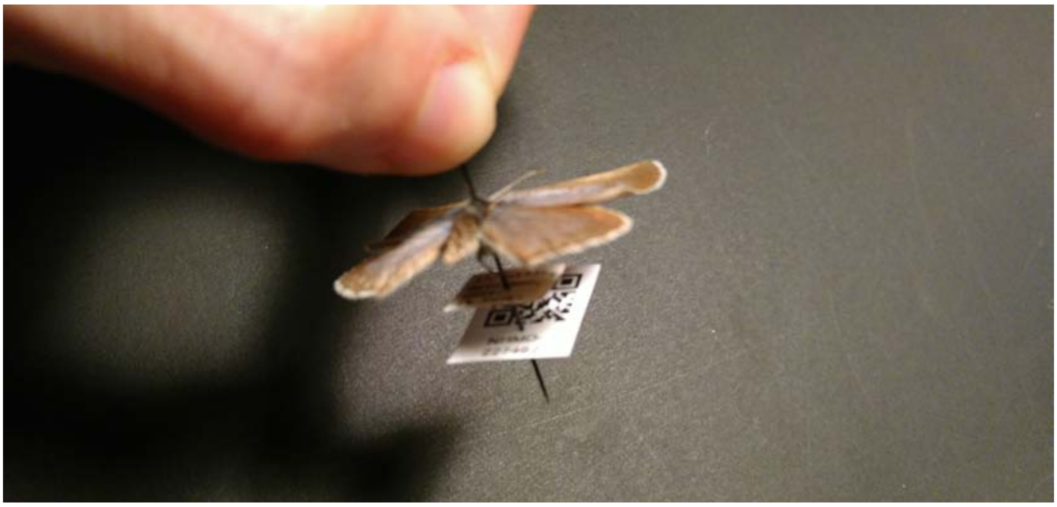


Fig. 1.2.: Butterfly specimen on needle with labels

1.2 Collections Database: 'Specify'

The natural history museum of Denmark uses an open source collections management system called Specify. More detailed information about this system can be found on their mainpage [3] and github [4]. Two versions of specify are currently in place at the museum, the legacy Specify 6, which is written in java, and the latest Specify 7 which is written in python with a django framework. Due to the large difference between Specify 6 and 7, both are still in use at the museum.

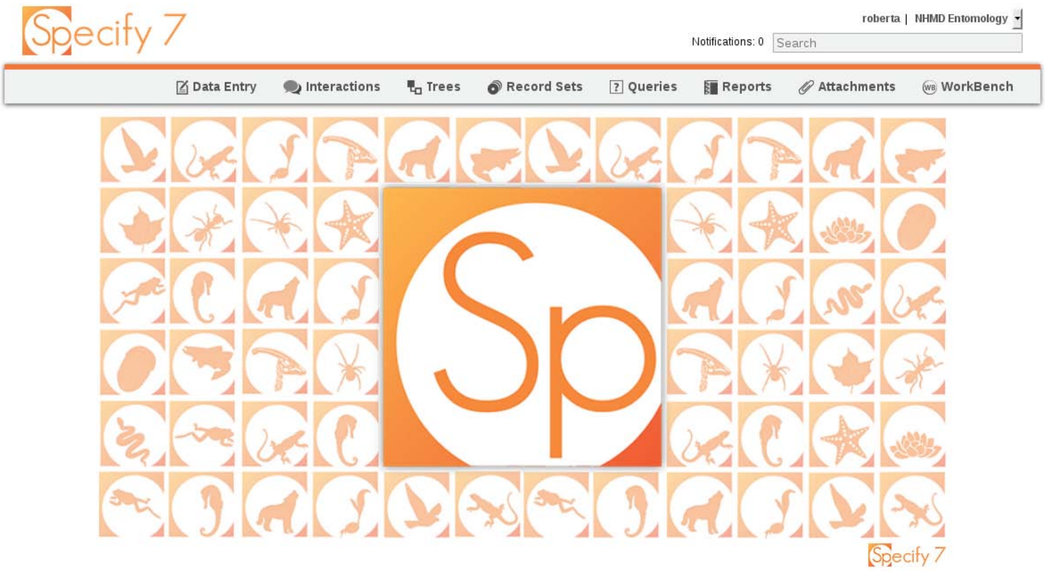


Fig. 1.3.: Specify 7 interface

2 Project Requirements

The main purpose of this project was to create workflows for digitizing the insect collections at the museum and making the data available to the public and researchers around the world. We planned to do this by hosting a permanent and public IIIF compliant image server and building a prototype digitization station. The image server should provide images of the collections to researchers and the public, and the digitization station should collect high-quality images of the insects and labels as quickly as possible, and allow each insect to be individually catalogued at the same time. Some more detailed requirements are listed below as well as some nice-to-haves for both the server and digitization station.

Image Server Requirements:

1. Must be at least minimally compliant with the IIIF Image API, but preferably also implementing as many of the IIIF features as possible
2. Fast to download images
3. Easily customizable for adding features like IIIF presentation API, ability to download raw files, etc.
4. Simple to add new images to
5. Simple to set up
6. Wide-spread usage (more likely to be maintained and updated in the future)

Digitization Station Requirements:

1. Will use a Sony alpha RII camera on a kaiser workbench - These had already been obtained by the museum, This allows digital control of the camera's vertical position
2. Must upload the images securely to a new IIIF compliant server where the images are publicly readable
3. Must interface with the current collections management database, Specify, and automatically upload the images to the database with the correct catalog number
4. Must be as user-friendly and automated as possible

5. Must include an insect-tray which will hold the insect needles in place, and hold the color target and labels, preferably all at the same height so that the insect, labels and color target will be on the same focal plane.

Digitization Station nice-to-haves:

1. Uses the digital control of the camera's position to take multiple shots at different heights to allow for focus stacking to create one image with the entire insect in focus
2. Software automatically reads the QR code/catalog number from the image, names file accordingly
3. Ability to use GUI to control camera settings, such as focus, white balance, F-number, color temperature

3 IIIF Compliant Image Server

IIIF (International Image Interoperability Framework) is an image API standard which is gathering popularity, especially as public institutions further digitize their data and want to make it available to the public. More about the details of this specification can be found on their website.

3.1 Choice of Image Server

Many servers already exist which implement IIIF. A list of the most widely used ones is available on the IIIF website. Of these, we decided to focus on comparing two: 'loris', because it uses python which is easy for us to modify and change, and IIP, because it is noted as high-performance, and we want the images to be as quickly accessible as possible.

Given the requirements listed in chapter 2, we made a comparison of the two servers, loris and IIP, resulting in a decision matrix which can be found in Appendix B. Using this we chose to implement the IIP Server going forward, mainly because it is faster (up to 1000x faster for smaller thumbnail images), even though it requires approximately 40% more storage space to store the images in multi-resolution tiff format.

3.2 Integration with Collections Database

Part of this project required the image server to integrate with the collections database 'Specify' so that collections managers can link the images with the catalog numbers in the database, and their associated metadata including species information, label transcriptions, etc. Specify already comes with their own attachment server which can include images, and the intention was to keep this attachment server separate from the IIIF server, so that documents and images that need to be kept private could still be stored on the attachment server, and images that could be publicly available could be stored on the IIIF server.

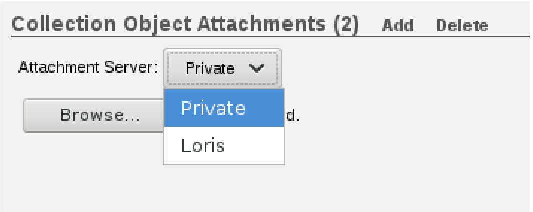


Fig. 3.1.: Added dropdown menu to choose server.

There are two versions of Specify currently in use by the museum: Specify 6 and 7. However, the museum had decided to phase out the usage of the java-based Specify 6, and since Specify 7 uses a python backend, we decided to focus on development in Specify 7.

During the internship I modified the Specify 7 codebase to allow easy configuration of new attachment servers to be added to the collections management system, and added a dropdown menu in the specify interface allowing collection managers to easily choose where the attachment should go. The interface modification can be seen in Fig. 3.1 below. These modifications are included in a pull request (which is still open) on github. The added code for the pull request is also included in appendix E.

4 Digitization Station

The physical station itself had already been chosen as a Kaiser RSD with Copy Stand (Model 5603) with a wired remote control and a Sony Alpha 7R II camera. The camera is connected to a mac which can be used to control certain features of the camera through a software called gphoto2. This was all purchased and determined before the internship's start.

Gphoto2 allows for remote control of many features of the camera - Unfortunately not the adjustment of the focus to a specific focal length, but capturing and downloading images from the camera, capturing the liveview of the camera, and adjusting numerous camera settings such as white balance, F-Number, etc.

4.1 Physical components

As indicated in the requirements for the project (see Sec. 2), ideally we wanted to be able to take stacked photos of the insects, in order to produce images where all parts of the insect are in focus. Since the camera is not capable of digitally controlling the focal length to do focus stacking, we chose to try automatically controlling the camera height on the Kaiser stand. The camera station came with a hand-operated controller for the height, but since we wanted this to be machine controllable, we decided to try controlling it with an Arduino. A sonar sensor to measure the camera's height was also included in the design. However, the sonar sensor ended up not being sensitive enough to provide precise (within mm) measurements of the height. Therefore the sonar sensor is not currently used, but is included in this report in the hopes that if the museum wishes to replace the sonar sensor with a more precise one in the future, this can be more easily done.

Fig. 4.1 below shows how the arduino was connected to the motor controller and sonar sensor. The arduino is programmed to send 5V signals to the motor controller when the gui requests the camera to move. Fig. 4.2 below shows how the physical system itself looks. This allows movement of the camera in $<$ 1mm increments.

The code for the arduino is provided in appendix C and also available on github.

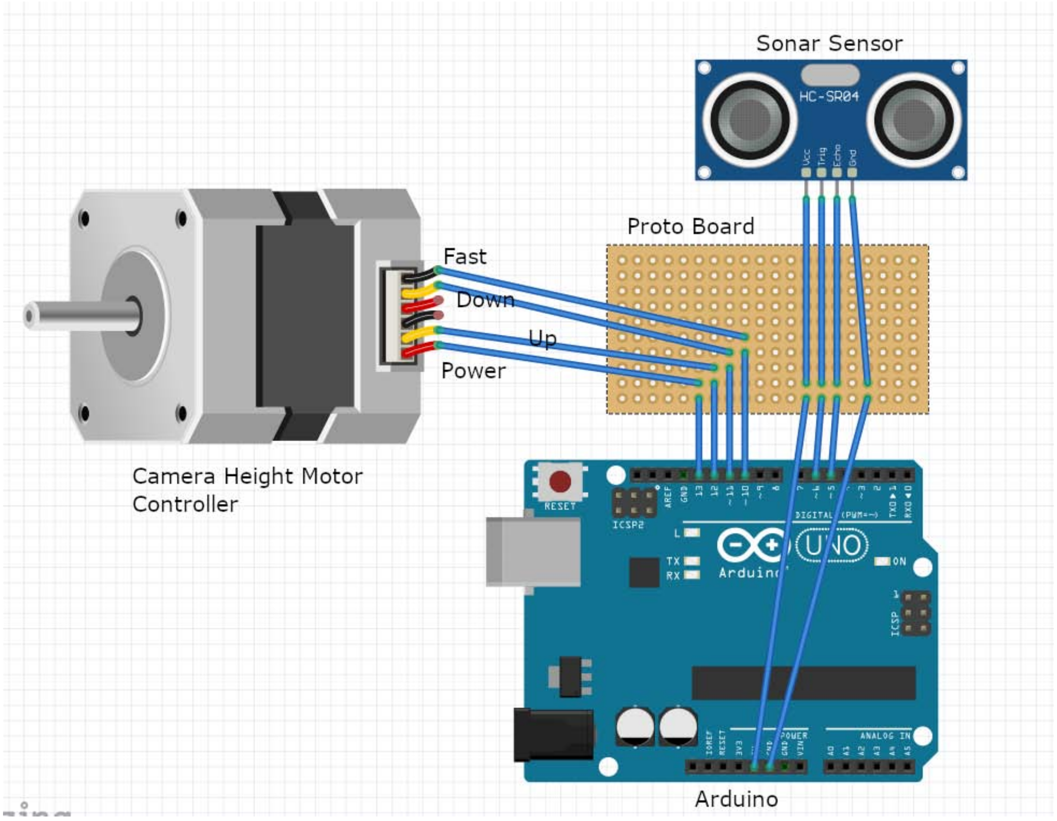


Fig. 4.1.: Wiring of arduino to motor controller. Pins 10-13 control the height of the camera. One pin for up, one pin for down, one to increase the speed on the motor, and one to provide power. Pins 5 and 6 are the echo and trigger pins for the sonar sensor, and the 5V and GND pins provide power to the sensor. In reality these pins are connected using DIN connectors to make it easy to connect and disconnect the sensors and arduino.

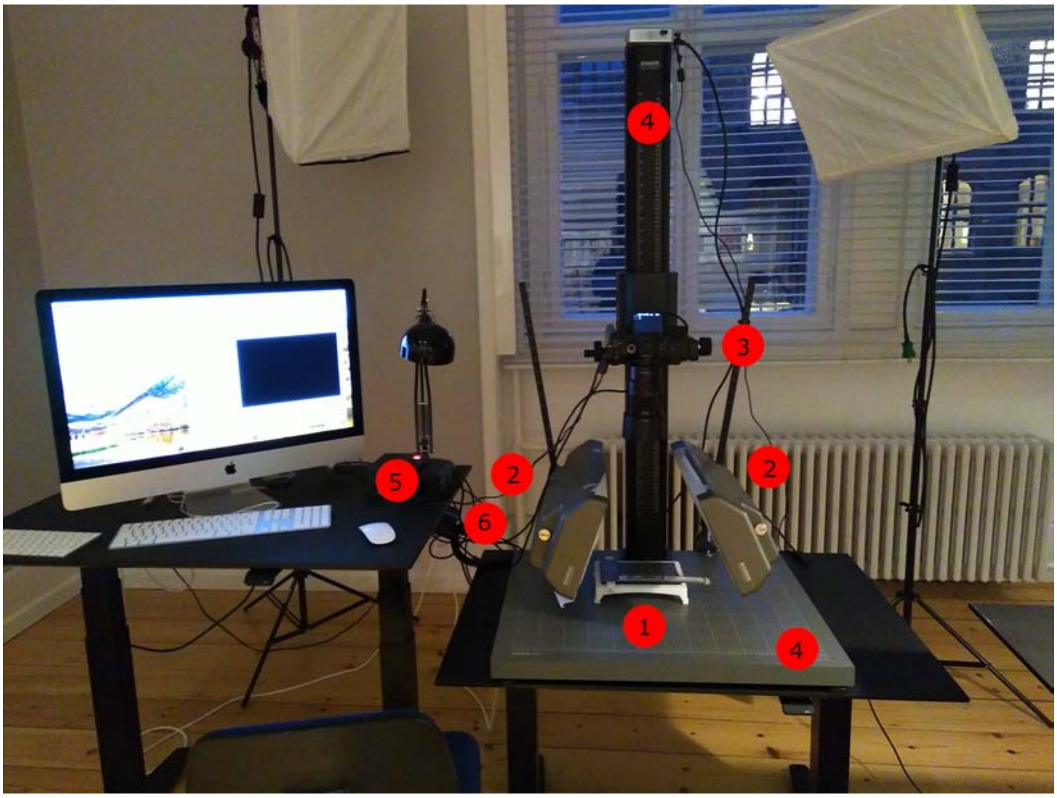
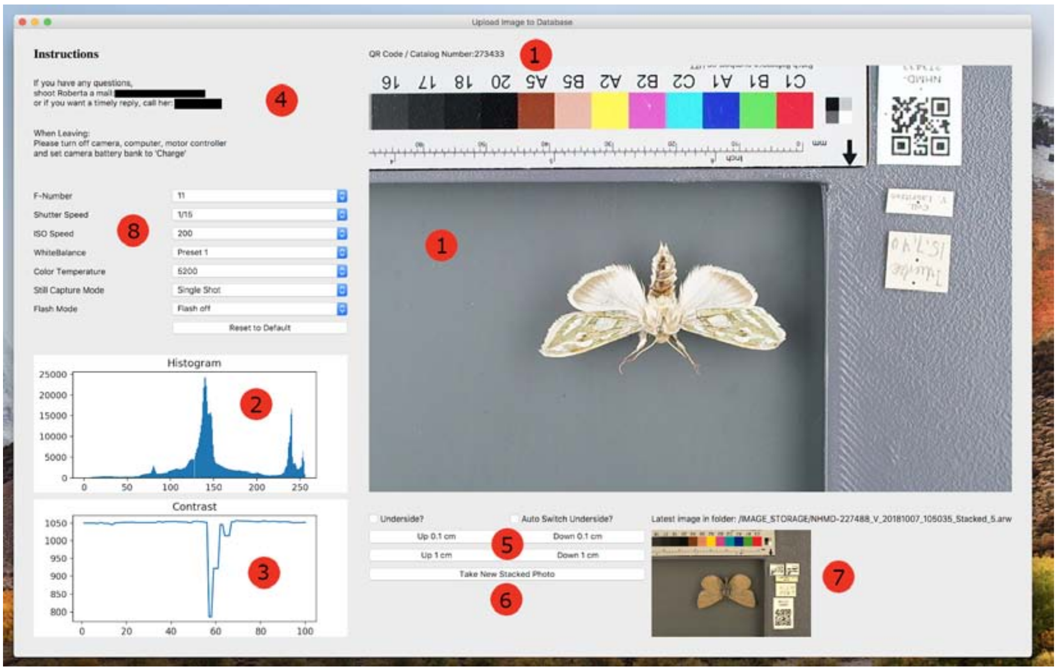


Fig. 4.2.: Physical set up of the digitization station. This includes the (1) insect tray itself, (2) two florescent lamps on either side of the insect tray to try to keep even illumination, (3) the camera attached to the (4) kaiser stand, and the (5) kaiser motor controller and (6) arduino box.

4.2 Graphical User Interface

A GUI was created to improve the ease of use. I learnt to use the package pyqt to complete this. An screenshot of the gui is shown in Fig. 4.3 below. The gui has the following features:

1. Live preview from the camera - updated every second with automatic QR Code reading
2. Live luminosity histogram - updated every second. Used to indicate if the image is over/under exposed.
3. Live history of contrast - updated every second. Calculated using the variance of the laplacian of the image - This is used to focus the image. As the image becomes more focused, the contrast increases.
4. Small information area, giving my contact information if there are problems. If this digitization station becomes a permanent fixture at the museum, this will be replaced with a permanent IT person's contact information
5. Controls for the camera height - able to move it up and down as necessary.
6. Button to take new stacked photo
7. Preview of last image taken - This is clickable so the user can quickly open the last picture as necessary
8. Controls for camera settings - F Number, Shutter Speed, ISO Speed, WhiteBalance, Color Temperature, Capture Mode and Flash Mode.



Upload Image to Database

Instructions

If you have any questions,
shoot Roberta a mail: [redacted]
or if you want a timely reply, call her: [redacted]

When Leaving:
Please turn off camera, computer, motor controller
and set camera battery bank to 'Charge'

F-Number: 11
Shutter Speed: 1/15
ISO Speed: 200
WhiteBalance: Preset 1
Color Temperature: 5200
Still Capture Mode: Single Shot
Flash Mode: Flash off

QR Code / Catalog Number:273433

Latest image in folder: /IMAGE_STORAGE/NHMD-227488_V_20181007_105035_Stacked_5.arw

Fig. 4.3.: Digitization station user interface.

The code for the GUI is provided in appendix D and also available on github.

4.3 Insect Tray

The insect tray was designed mainly to be 3D printed, with some components purchased where necessary (or cheaper).

As listed in the project requirements (section 2), the digitization station needs to include a structure to hold the insect needle, color target, and insect labels while the photos are being taken. In addition, the following requirements were added:

1. As easy as possible to calibrate the camera - for this reason we chose to use a gray card in the background of the insect tray
2. As easy as possible to automatically segment the insect and labels from the background - the 18% gray card background also helps with this, and the label holder
3. A durable material for the needle holder which could remain tightly hold needles, even after a thousand profusions - for this reason Martin Stein had the idea to use the bristles of a tightly compressed barber's brush (shown in figure 4.5. This also allows us, if necessary, to unscrew the brush and replace it without replacing the entire tray.
4. Possibility on an ad-hoc basis to lower/raise the label holder, depending on the size of the insect, and the insect's location on the needle - for this reason, the label holder piece 'floats' over top of the tray, and can be raised / lowered in 3 mm increments using small lego pieces.
5. Possibility to move where the label holder is in relation to the needle holder to allow for imaging of insect of varying sizes - this is another reason that the label holder 'floats' above the insect tray. Small M3 screws attached to the sides of the label holder can be loosened to allow the label holder to move - it can therefore slide in all directions.

List of components for the insect tray:

1. 3D printed tray
2. 3D printed color target holder
3. 3D printed slider
4. 18% gray card
5. 30 small lego pieces
6. Color target reference
7. Barber's brush compressed with O-ring clamp to hold needles

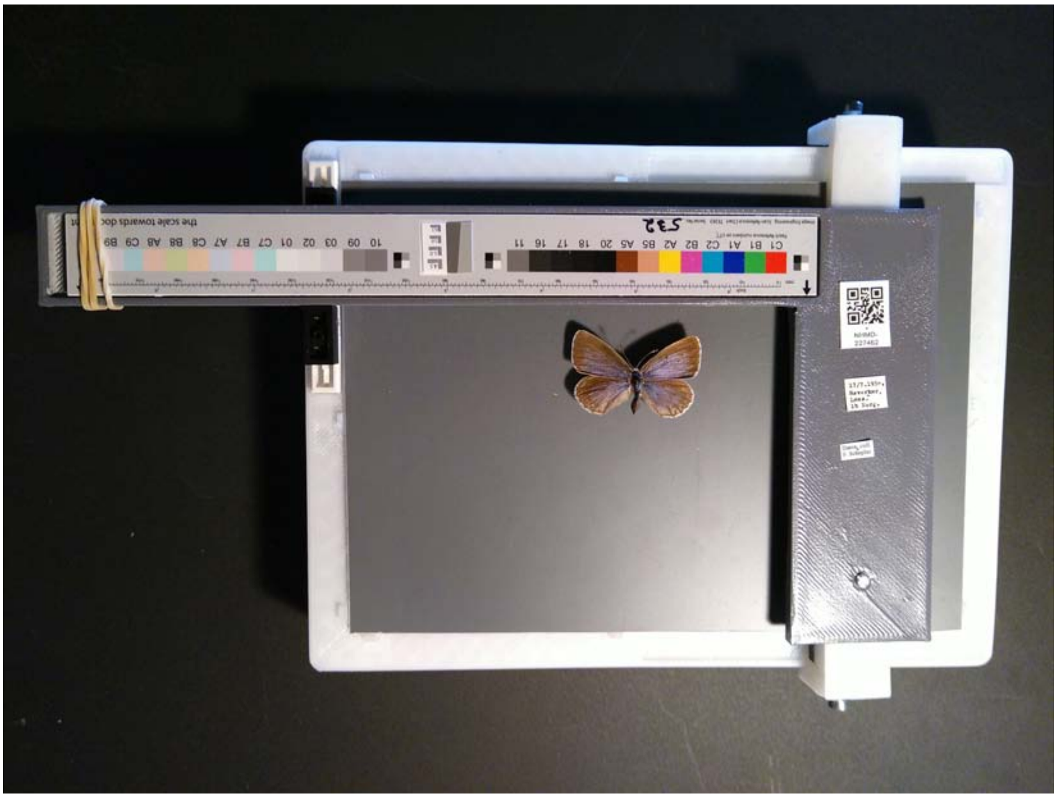


Fig. 4.4.: Assembled insect tray with butterfly.

8. 1 M2 screw with nut
9. 2 M3 screws with nuts

The models for the 3D printable components are available for download on github.

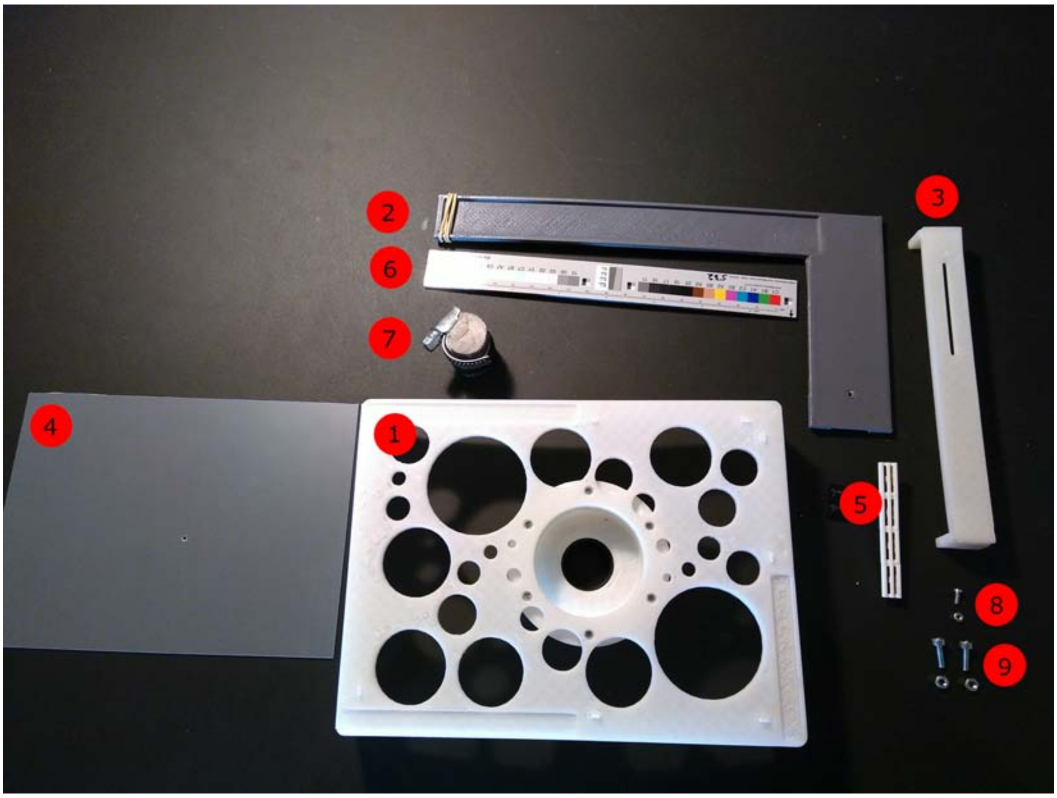


Fig. 4.5.: All components of disassembled insect tray.

Results 5

5.1 IIIF Server

The IIIF server is just now moving from the development phase to production, but the download times for the development server are as low as 0.006673 seconds for smaller images, which is quite promising. In addition the IIP server that was chosen seems to satisfactorily implement the IIIF Image API specifications, including on the fly resizing, rotation and cropping. However, automated uploading of images still needs to be implemented.

5.2 Specify Integration

The IIIF Development Server has been successfully integrated into a local copy of the specify server. The changes made to specify are currently included in a pull request to the specify team, which is awaiting their approval / comments. The code for the pull request can be found in appendix E.

5.3 Digitization Station

The digitization station has been running for 2 months as a part of the butterfly exhibit at the geological museum (A guide made for running the digitizaiton station can be found in Appendix A). Approx. 650 specimens have been digitized to date (as of Dec 19, 2018). Due to the time it takes to move the camera during focus stacking, the requirement that the labels be removed and imaged alongside the butterfly, and the fact that this is part of an exhibition which allows interaction with the public, the digitization rate is currently only around 8 specimens per hour, with a theoretical maximum of 18 specimens per hour (assuming no interruptions). The requirement for direct integration of the digitization station with specify has not yet been achieved. Currently the digitization station uploads the images to an intermediary server, and from there they need to be added to specify via batch script. Direct integration will be explored in the future.

6 Conclusions

6.1 Discussion

6.1.1 IIIF and Specify

The IIIF server and the Specify modifications seem to meet the requirements of this project in the development environment. It remains to be seen if this will work as expected in production.

6.1.2 Digitization Station

The throughput of the digitization station, theoretically 18 per hour at best and 8 per hour on average, is relatively low, especially compared to systems such as ALICE [2] which boast average rates of 140 specimens per hour. However, it is a starting point, and allows for focus stacking which is not included in most systems such as ALICE.

6.2 Next Steps

6.2.1 IIIF Image Server

The IIIF image server still needs to be moved from the development environment to a production server. Possible future improvements to the system after that include adding the IIIF presentation API and content search capabilities.

6.2.2 Specify Integration

The Specify code changes need to be approved by the Specify team before it can be integrated into the production system. Once this is completed, no further changes are expected.

6.2.3 Digitization Station

The prototype of the digitization station is complete, however, several possible improvements are being discussed, such as:

1. Increased throughput by removing focus stacking, or by adding angled imaging like ALICE [2], or through drawer-level imaging, like Inselect [1]
2. Direct integration of the digitization station with Specify so QR codes/catalog numbers can be printed on demand, and images can be directly uploaded into specify/the image server
3. Replace sonar sensor with more precise distance sensor to ensure camera always moves back to starting point

6.3 Conclusion

A prototype digitization station for individual insect specimens was designed, programmed and built. This station allows automatic focus-stacking of images and provides a user interface which allows easy modification of the camera settings. The normal throughput of the digitization station is 8 specimens per hour. Many possible improvements for the system have been identified and perhaps the museum will pursue them depending on funding and priorities.

A development version of a IIIF Image API Server was setup and integrated into Specify. This allows images to be available via the IIIF API. The image server chosen is fast and allows easy downloading of images from the museum collections.

Bibliography

[1] Lawrence N. Hudson, Vladimir Blagoderov, Alice Heaton, et al. „Inselect: Automating the Digitization of Natural History Collections“. In: *PLOS ONE* 10.11 (Nov. 2015), pp. 1–15 (cit. on p. 22).

[2] Benjamin Price, Steen Dupont, Elizabeth Allan, et al. „ALICE: Angled Label Image Capture and Extraction for high throughput insect specimen digitisation“. In: (Nov. 2018) (cit. on pp. 21, 22).

[3] Specify Collections Consortium. *Specify*. URL: `http://www.sustain.specifysoftware.org/` (visited on Nov. 8, 2018) (cit. on p. 5).

[4] Specify Collections Consortium. *Specify 7*. URL: `https://github.com/specify/specify7` (visited on Nov. 8, 2018) (cit. on p. 5).

A

Guide to Operating Digitization Station

Guide to Digital Lab - Butterfly Digitization

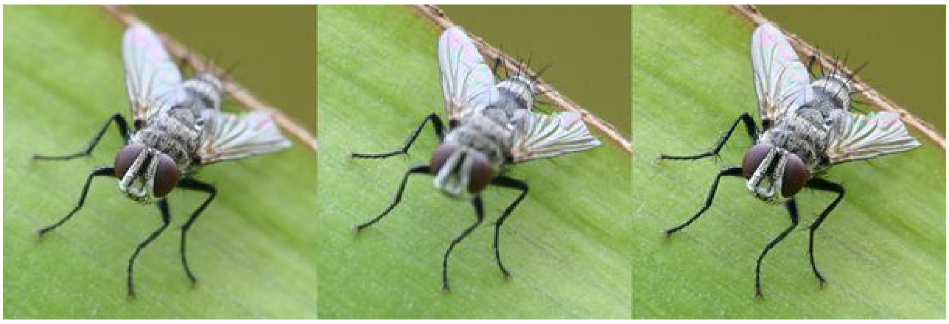
Thanks to all students for helping on this project.

We have tried to make this process as smooth as possible, but as you will notice there are still some kinks to work out. If you encounter any technical problems not explained here, first try restarting the image_gui application, then try restarting the camera, then restarting the mac. If none of that works, or the problem lies elsewhere, please call Roberta:, and hopefully we can sort it out as soon as possible. If you have any suggestions for new features, things that would make the work easier, or ideas for how to improve the process, please e-mail Roberta: - it may not be possible to implement everything, but if it is easy to, she might be able.

Some general notes about what we are doing

This project is an attempt at further digitizing the collections in the Natural History Museum of Denmark. Digitizing natural history museum collections is particularly hard due to the varied nature of the collections: from enormous whale skeletons, to jars of fish preserved in alcohol, and of course, various insects. We have already digitized many plant specimens, however these are somewhat easier because they are pressed and mounted on sheets and can be placed on conveyor belts, and scanned. Insects and butterflies are considerably more difficult due to their 3D nature. Ideally we would take high resolution 3D scans of the insects, and save those and make them available online. However, this is considerably more labor intensive, so as a first step, we will take 6 shots of the top of each butterfly. Each successive shot, the camera will move slightly closer to the butterfly, in order to focus on a slightly different part of the butterfly (this is done automatically once the first shot is set up). The idea is that we can then combine the 6 images into one, to get a super-focused image.

This is called focus stacking. An example is below:



(image from: https://en.wikipedia.org/wiki/Focus_stacking)

Currently we take the photos and then a background script runs which saves those on a server (ERDA). The next steps for us are to combine those images into a focus-stacked image, and make all the images publically available. This will hopefully be done before the exhibition is over, but that is a bit ambitious, so no promises.

Join us and take part in digitising our collections

A masters student, Philip, will be taking many of the first digitization shifts. He is working on his thesis, and imaging both the top and underside of the butterflies, and taking some genetic samples at the same time.

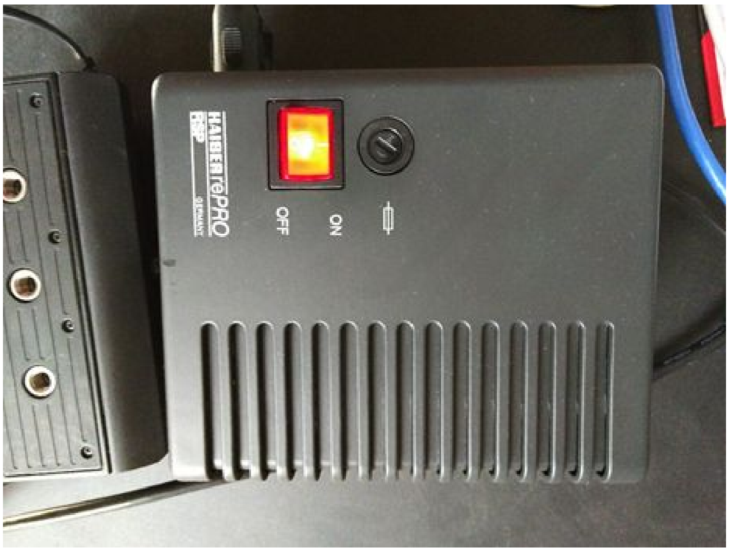
If you have some masters/bachelors project or thesis idea you think could be relevant to this digitization project, please let us know, and we can see if we can accommodate that. You can contact either Roberta, or Anders .

Another masters student, Roberta, will be doing a free topic on automatic transcription of the labels, so there are any potential possibilities.

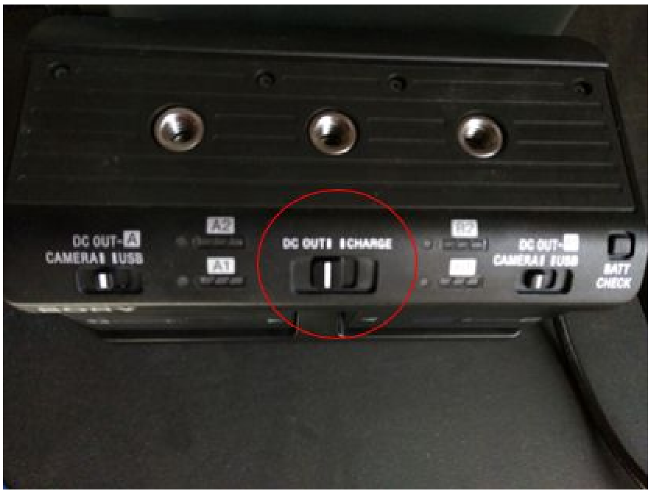
The image_gui application is completely open source, and is available on github (excluding passwords etc) in case you are curious or really ambitious with bug-fixes: https://github.com/robertahunt/image_gui

Instructions – Start of day

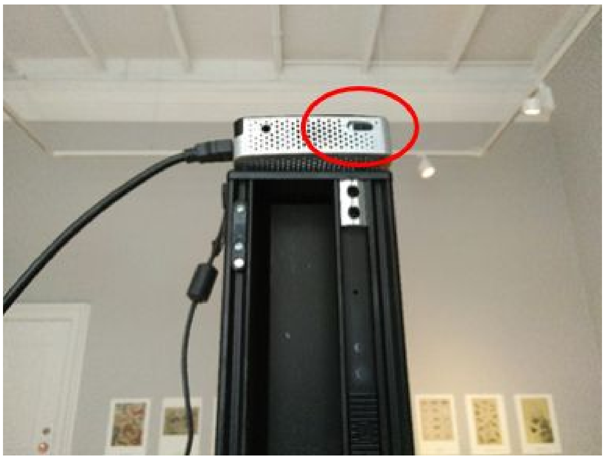
1. If the computer is off (although it shouldn't be, because it runs some programs in the background overnight), Turn on computer (power button is on the back of the mac) and login – username 'student'. (password sent in mail)
2. Turn on motor controller



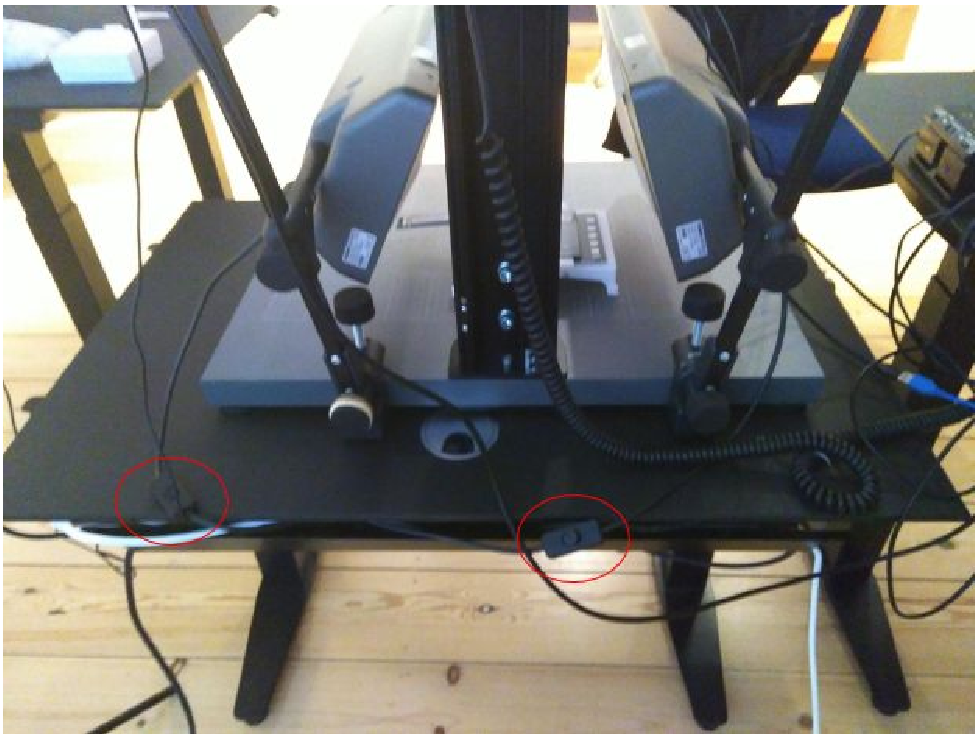
3. Turn camera battery bank to DC-Out



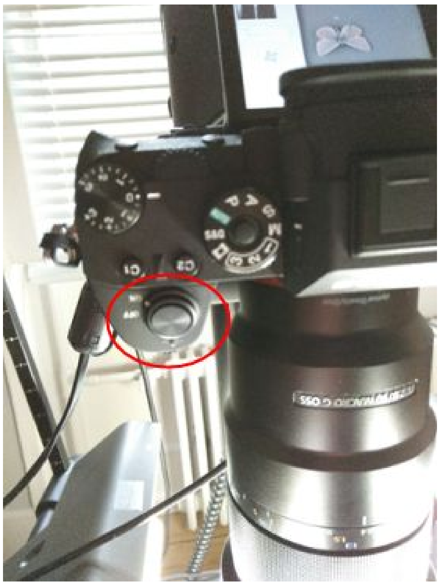
4. Turn on projector (on top of camera stand). Power button is on the back of the motor controller.



5. Turn on lights to right and left of specimen area (also behind the setup)

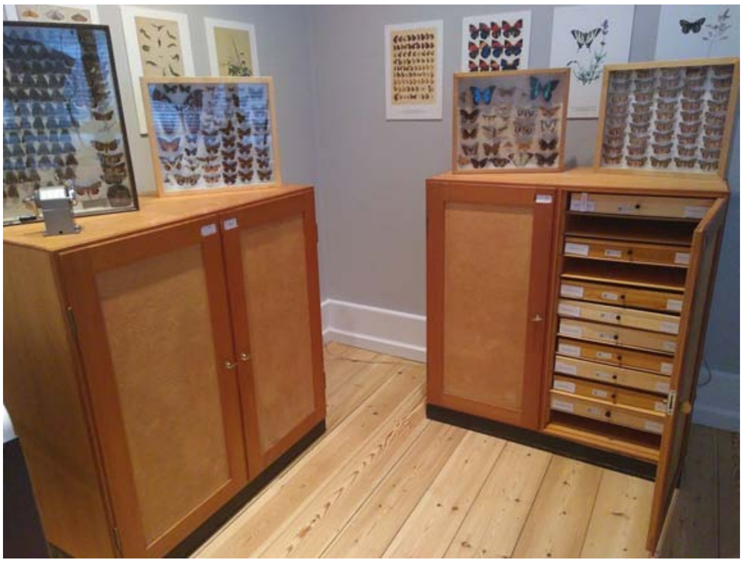


6. Turn on camera

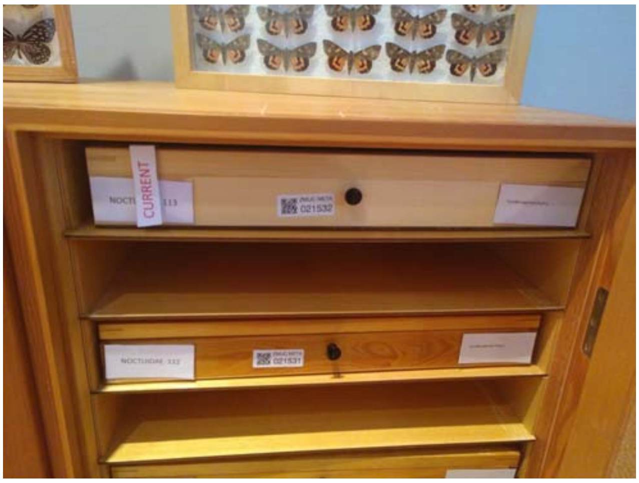


Instructions – Preparing each specimen

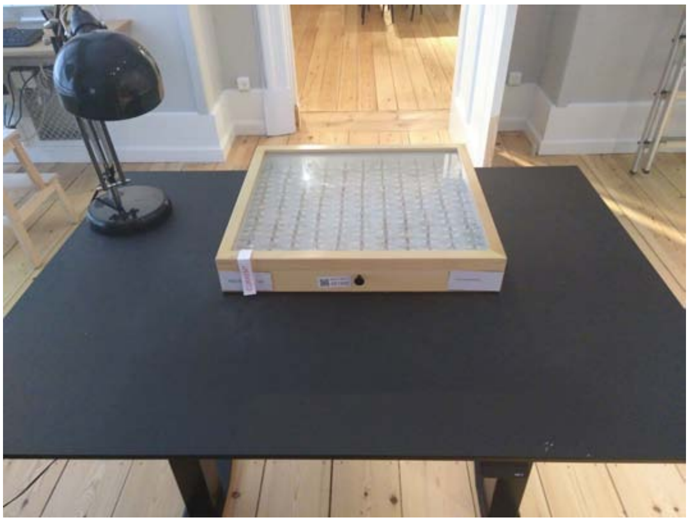
1. Take out current drawer:
    a. Open cabinet to the far right.



    b. Start from the top and go down. Skip any drawers which are labelled 'Done'. Find drawer that is marked 'Current'.

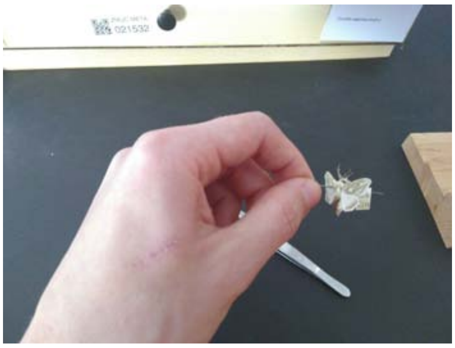


c. **Take this drawer out** and place it on specimen prep table.



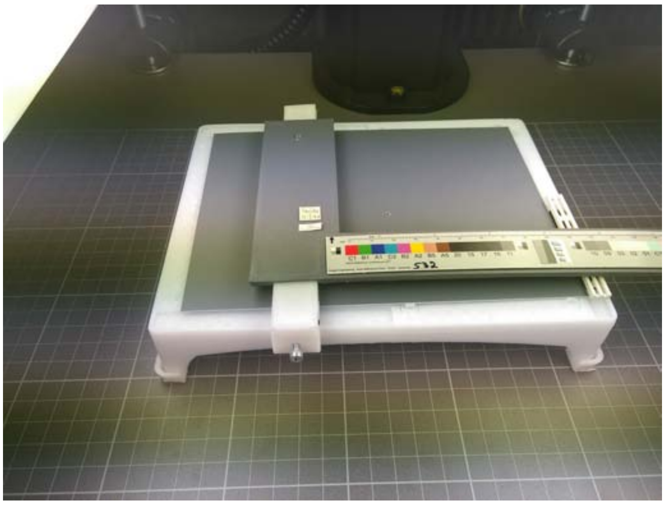
Avoid putting the lid on the table upsidedown, as this may let dust and other particles into the specimen tray when it is flipped back on top. Instead, place the lid, right side up, beneath the tray, as shown.

2. Take out next specimen
    a. Find next specimen (one without a QR code under it),
    b. **Take out next specimen**, holding them by the top of the pin and being careful not to touch the specimen

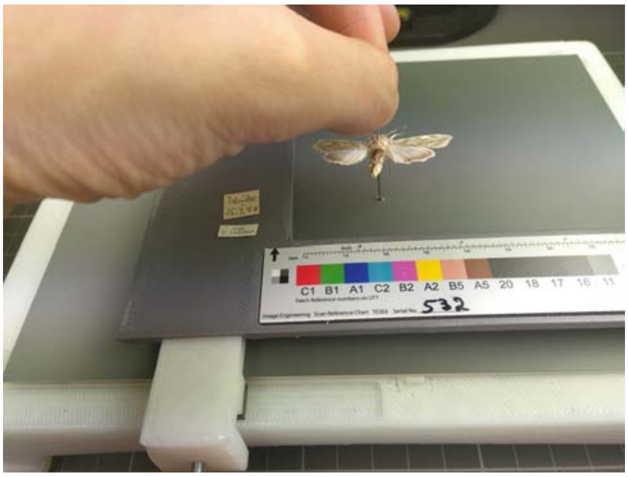


3. Using tweezers, **remove the label(s)** from the specimen. Be careful here, there are often many labels very close together, so please make sure they are all split apart and visible on the label holder as you take them off.

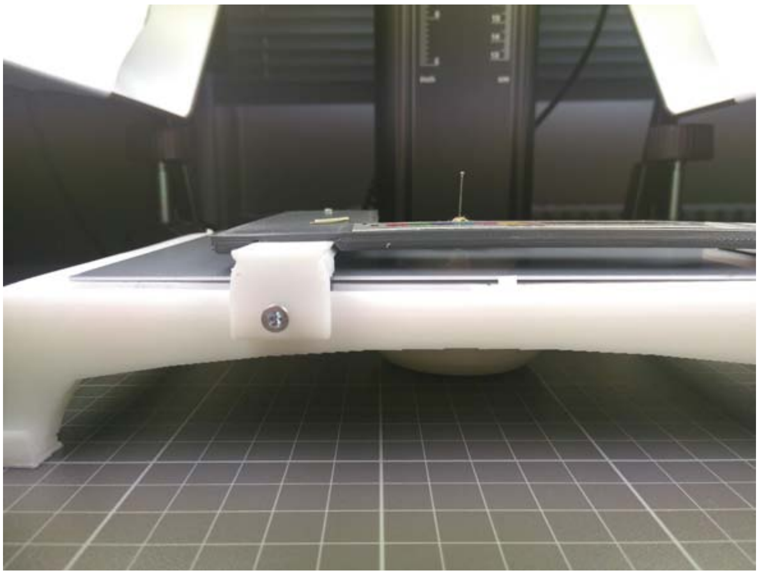
4. **Arrange the labels on the tray** so they would be read right side up.



5. **Put specimen into digitization tray, with the Venstral side up**. If the specimen is too wide or tall, you can adjust where the label holder and color target sits by unscrewing the screws (use the screwdriver and allen key provided, sitting at the bottom of the cabinet next to the labels). The sliding mechanism is snug, so it may take some effort to slide properly.

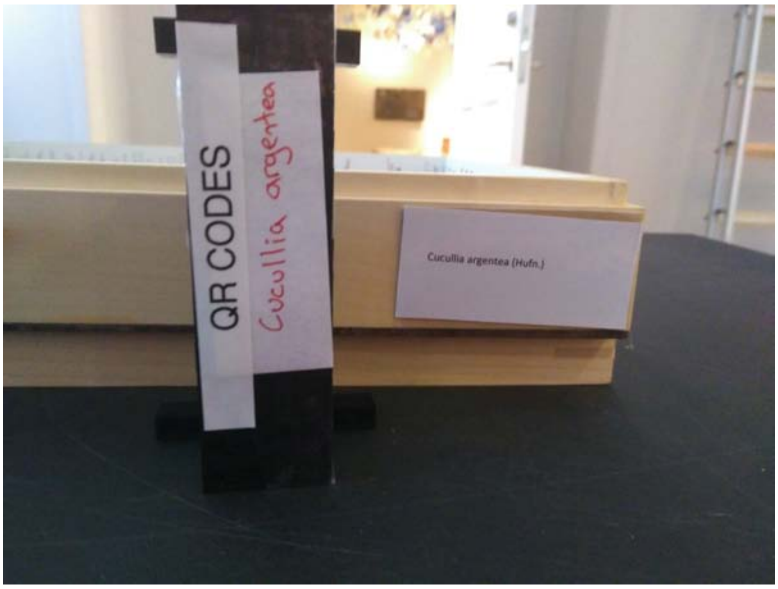


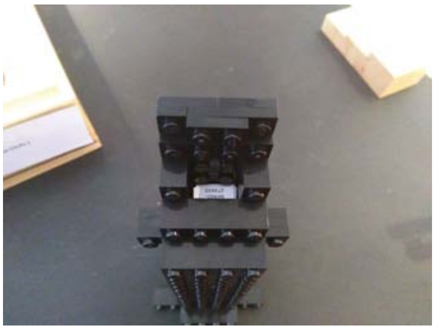
6. **Adjust specimen height** - Ideally all parts of the specimen should be just below the label / color target holder, as shown. If the specimen is too tall to fit between the tray and the gray card beneath it, please see instructions ‘Adjusting the height of the label holder’ below.



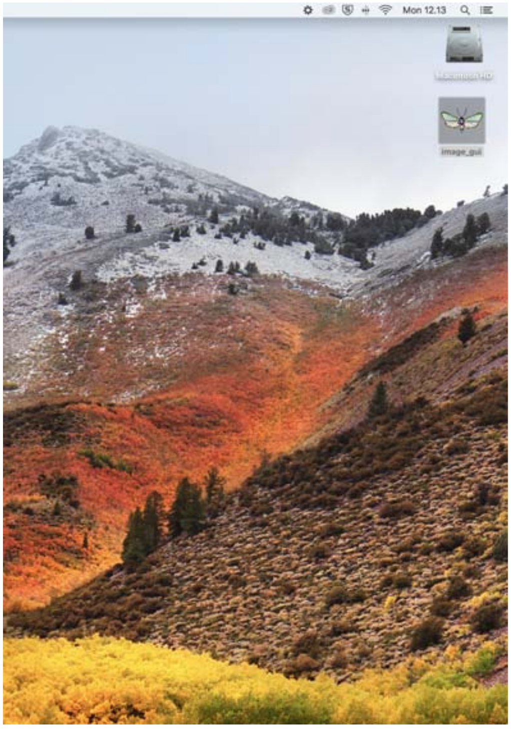
7. **Put a QR Code on label holder** from the QR Code pile on the label holder. This is the new catalog number for the specimen, and should be automatically read by the image-capture software. Please note that there are two piles of QR Codes. One is for the thesis student who needs two of each QR code. You should use the ones shown below. Make sure they match the species name on the drawer. QR Code holders are in cabinet

132, and labelled with the species that is associated with them. Please ensure you are using the right QR codes.

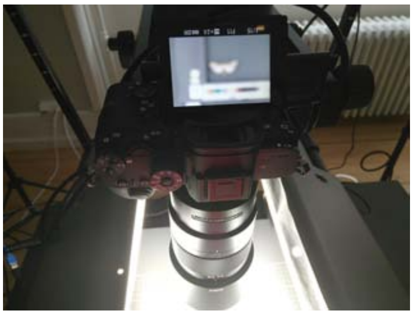


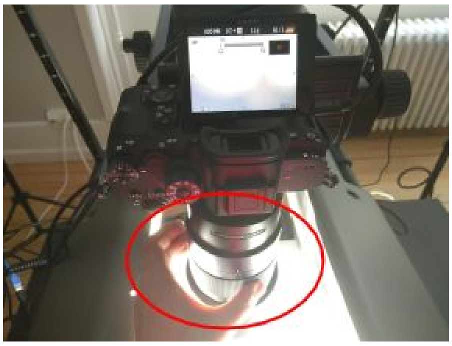


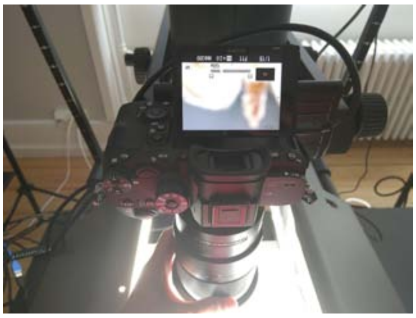
8. Once the specimen is on the tray and the label holder is just above the specimen, go to the Mac and **open the image_gui application**, if not open already (on the desktop)


Mon 12.13
Macintosh HD
image_gui

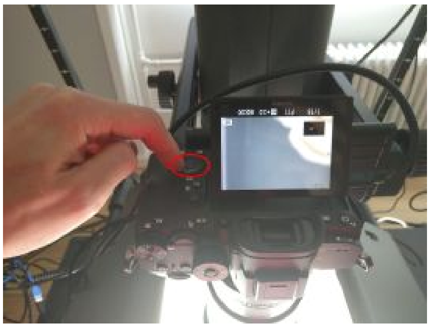
9. **Adjust the camera height** using the ‘Up’ and ‘Down’ buttons on the gui, until the labels, color target and butterfly are in view, and take up most of the image - remember what height this is. If you ensure each time that you return to this height, you will only need to manually focus once.
10. **Manually focus the camera** onto part of the label holder. This is important that all of the butterfly/moth is slightly below the surface of the label holder, and that the camera is focused on the label holder, so when the camera moves down as it takes photos it will focus on different parts of the butterfly’s body.
    a. Adjust the focus slightly by rotating the textured rotar on the lens - this will create a zoomed preview.

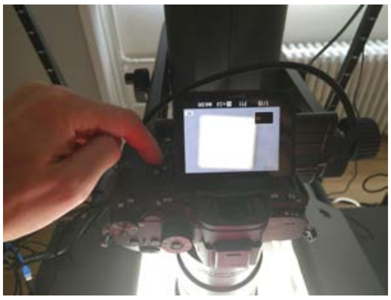






b. This will automatically focus in the middle of the picture, move the 'zoomed' location over to look at one of the labels. You can move it by pressing the arrow buttons on the top of the camera

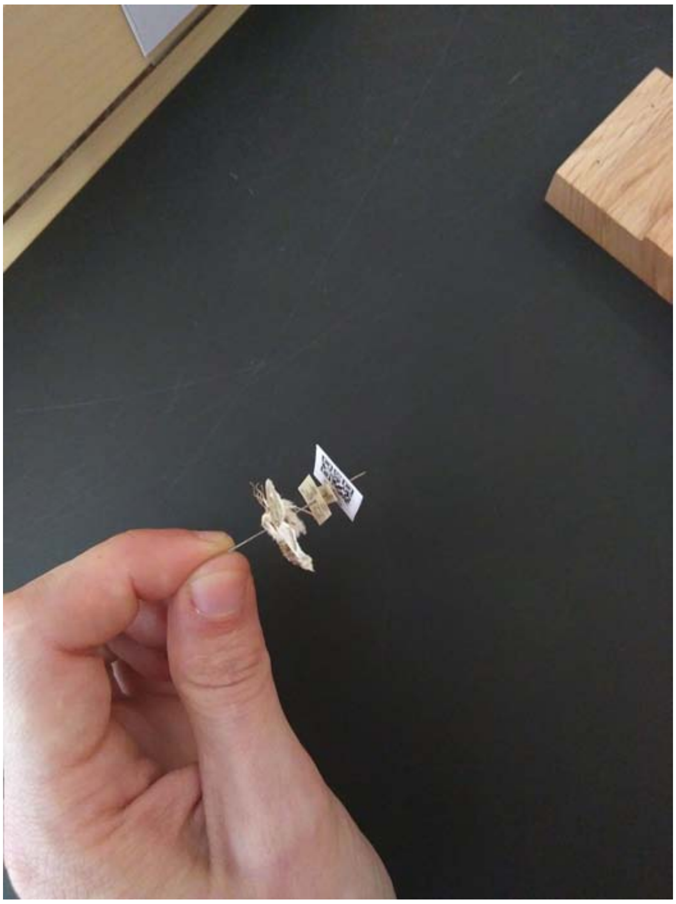




c. Adjust the focus by rotating the textured rotar on the lens until the label is in the best focus possible. You can then get out of the zoomed preview by lightly pressing the 'take photo' button on the camera. This is however, not required (it will take un-zoomed photos regardless).

11. Go back to the computer and **make sure 'underside?' and 'auto-underside?' in the image_gui application are unchecked.** This is for the thesis student who is also imaging the underside of the butterflies.
12. Click **'Take stacked photo'.** This should take about 45 seconds to take 6 shots, moving down approximately 1 mm for each shot. Through this process it will also automatically open the first shot you took and the last shot, so you can see the range of focal planes. Pressing Command+0 on the keyboard will zoom the shot into max resolution, so you can see more clearly.
13. If the image looks too overexposed, or underexposed, you can adjust the shutter speed and iso speed on the camera using the dropdown menus (please do not change the f number). If you mess something up, you can always return to the defaults by clicking reset to default.
14. If you like, once that is completed, you can look at the images by opening Finder and going to 'IMAGE_DUMP' on the quick access bar. Here the latest raw files are stored locally so you can look at them. The IMAGE_STORAGE folder keeps a local copy of all the raw image files, and the IMAGE_CACHE is used as an intermediary store for the raw files as they are converted to tiff and uploaded to the database.
15. Take digitization tray back to specimen preparation area.

16. Using tweezers, **put labels back onto the needle beneath the specimen** – ideally using the same hole that the needle was in originally.



17. **Put QR Code, facing up, onto needle** beneath the specimen. Try to put the needle between the QR code and the written number, so everything can be read easily later.

18. **If you just digitized the last specimen in a drawer**, congrats! Please place a 'Done' label on the drawer. (tape and labels are provided next to the lego, at the top of the left

19. If you are running out of QR codes / specimens please email and

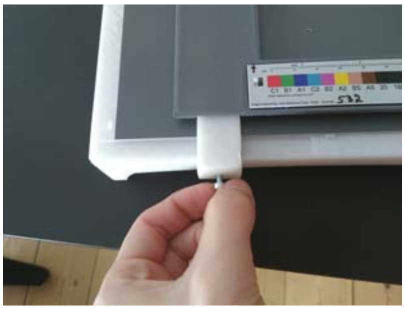
side of cabinet 132.

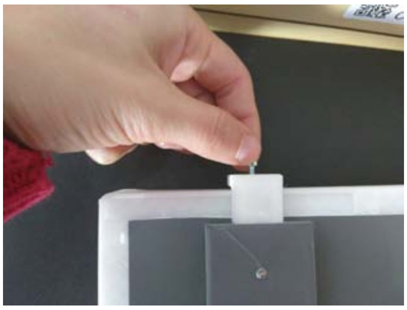
Instructions – End of day

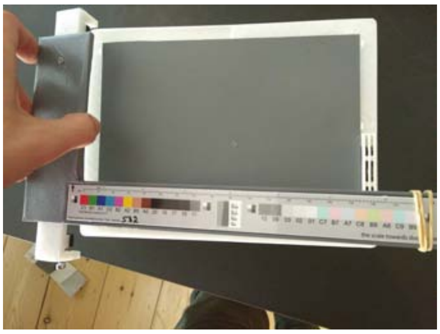
1. Make sure the drawer you were working on is labelled with ‘Current Drawer’ Marker.
2. If you finished an entire drawer, make sure to mark it as finished by putting a ‘Done!’ paper on it, as shown.
3. Open ‘Finder’ on the desktop and go into the image_cache folder. If there are any images there from before today (they are named with a time stamp), please email robert. This means there is something wrong with the upload proces
4. Pack away QR code holder using lego blocks to your heart's content.
5. Turn off motor cotroller
6. Turn camera battery bank to charge
7. Turn off projector (on top of camera stand)
8. Turn off lights to right and left of specimen area (also behind the setup)
9. Turn off camera

Instructions – Adjusting the height of the label holder

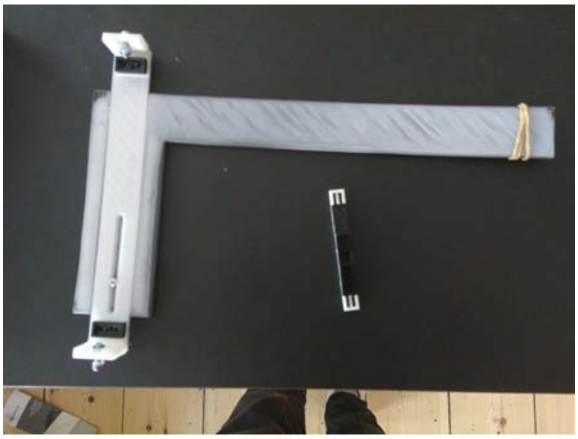
1. Move labels to the side (on the gray card, or somewhere safe)
2. Take off label holder – by unscrewing the label holder from the digitization tray (the screws should just be finger-tight)





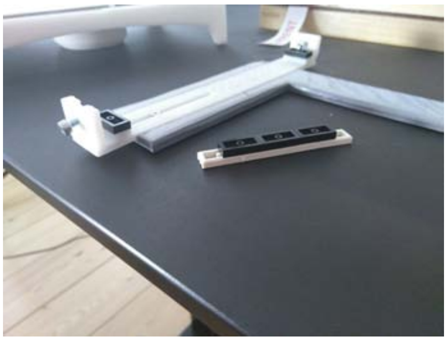


3. Flip label holder over

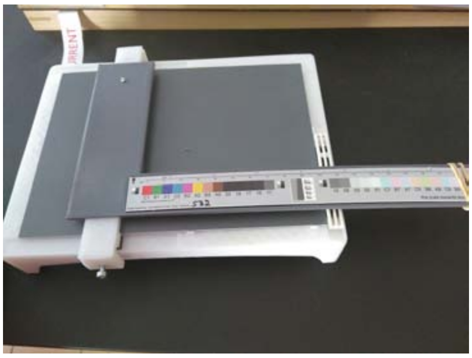


4. Add or remove black lego pieces to the top and bottom areas as shown, they should have the same number of lego pieces. Extra lego pieces are stored in the cabinet.
5. Add or remove black lego pieces from the side lego. This should always have one more white lego piece (flat topped) more than the other two lego areas. So if each of the top and bottom have one black lego piece, the side lego area should have one black lego

layer, and one white lego layer on top of that.

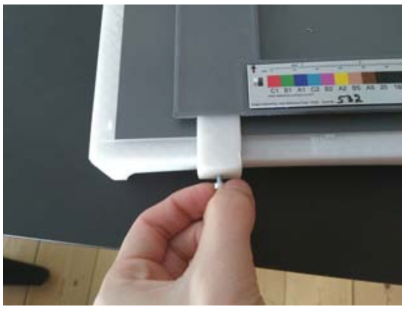


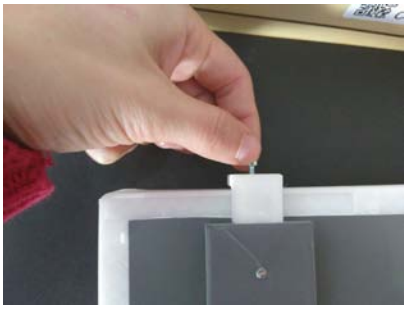
6. Put the label holder back on and screw it in finger-tight



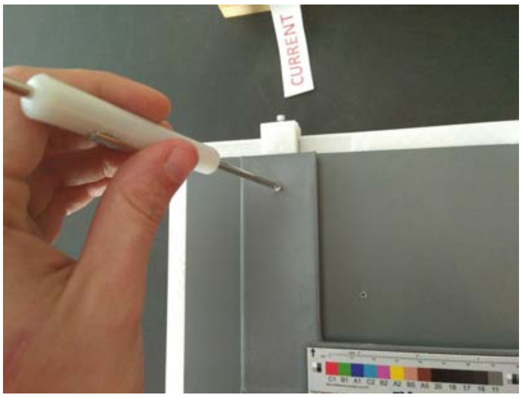
Instructions – Adjusting the location of the label holder

1. Slightly unscrew the screws at the top and bottom of the label holder





2. Use the screwdriver in the cabinet to unscrew the label slider



3. Adjust the location by sliding the sliders.
4. Screw the label slider back in with the screwdriver, and the other bolts with your fingers

B

IIIF Server Decision

B.1 Decision Matrix

		Servers				
		Loris		IIP		
Criteria	Weight	Score	Total	Score	Total	Justification
Fast	5	2	10	5	25	A comparison was completed between loris and IIP for downloading images of different sizes. It can be found in table B.2 below. For thumbnail size images IIP was over 100 x faster in some cases. This could be solved by pre-making thumbnails for all images, but IIP seems like a cleaner option. IIP is faster for thumbnail images because it uses multiresolution tiled-tiff format.
IIIF Options	1	4	4	3	3	Loris allows rotation at any angle, IIP only allows rotation at 0,90,180 and 270 degrees
Customizable	2	5	10	2	4	Python is much more easily customizable for the team at NHMD, However we should be able to use IIP for the IIIF part, and create new servers for other functionality, such as the presentation API, uploading images, etc.
Simple image addition	3	5	15	5	15	Both servers just find images in a image dump folder
Widely used	3	2	6	5	15	IIP has a long list of usage in other institutions, including many museums. Loris is just gaining footing in these areas as far as we can tell.
Total			45		62	

Tab. B.1.: Decision matrix to decide which IIIF compliant server to use.

B.2 Download Times

Image Resolution	Server	
	IIP	Loris
full	0.946592	1.458021
!100x100	0.006673	0.653253
!200x200	0.008482	0.833460
!400x400	0.018264	0.828187
!1000x1000	0.167003	0.989094
!2000x2000	0.649720	1.345052
!3000x3000	0.703145	1.126506

Tab. B.2.: Comparison of image download times (in seconds) for two different IIIF compliant servers, in a development environment.

C Arduino Code

link.

```
//Set Up Sonar Sensor
#include "SR04.h"
#define TRIG_PIN 6
#define ECHO_PIN 5
SR04 sr04 = SR04(ECHO_PIN, TRIG_PIN);


//Set Up Motor Controller
int data;

float distance = 1;
float adjustment = 1;

boolean newCommand = false;
float ms_per_cm_up = 750;
float ms_per_cm_down = 890;

int CHECK_CONTROLLER_PIN = 13;
int UP_PIN = 12;
int DOWN_PIN = 11;
int controllerOff = 1;

static byte nowReadingDistance = 0;
static byte ndx = 0;
static char endMarker = ’\n’;
static char splitMarker = ’ ’;


char rc;
int numChars = 10;
char receivedDirection[10]; // an array to store the received data
char receivedDistance[10];
```

```
void setup() {
  Serial.begin(9600); //initialize serial COM at 9600 baudrate
  pinMode(LED_BUILTIN, OUTPUT); //make the LED pin (13) as output
  digitalWrite (LED_BUILTIN, LOW);

  pinMode(CHECK_CONTROLLER_PIN, INPUT);
  pinMode(UP_PIN, OUTPUT);
  pinMode(DOWN_PIN, OUTPUT);
  digitalWrite(UP_PIN, HIGH);
  digitalWrite(DOWN_PIN, HIGH);
  controllerOff = digitalRead(CHECK_CONTROLLER_PIN);
}

void loop() {

  if (Serial.available() && newCommand == false){
    static byte nowReadingDistance = 0;
    static byte ndx = 0;
    char endMarker = '\n';
    char splitMarker = ' ';
    char rc;

    rc = Serial.read();


    if (rc == splitMarker) {
        receivedDirection[ndx] = '\0';
        nowReadingDistance = 1;
        ndx = 0;
        }
    else if (rc == endMarker){
      nowReadingDistance = 0;
      receivedDistance[ndx] = '\0'; // terminate the string
      ndx = 0;
      newCommand = true;
      runCommand();
      }
    else {
        if (nowReadingDistance == 1) {
          receivedDistance[ndx] = rc;
          ndx++;
```

```
     if (ndx >= numChars) {
      ndx = numChars - 1;
     }
    }
    else {
     receivedDirection[ndx] = rc;
     ndx++;
     if (ndx >= numChars) {
      ndx = numChars - 1;
     }
    }

   }
 }
 else{
  //long sonar_distance = sr04.Distance();
  //Serial.print(sonar_distance);
  //Serial.println("mm");
  //delay(500);
 }
}

void runCommand() {
  digitalWrite(UP_PIN, HIGH);
 digitalWrite(DOWN_PIN, HIGH);
 controllerOff = digitalRead(CHECK_CONTROLLER_PIN);

 if (newCommand == true) {
  if (!controllerOff){
   float distance = atof(receivedDistance);
   if (receivedDirection[0] == ’u’) {
    Serial.print("Going up ");
    Serial.print(distance);
    Serial.println(" cm");
    if (distance > 0.5){
     adjustment = 1.115;
     }
    digitalWrite(UP_PIN, LOW);
    delay(distance*adjustment*ms_per_cm_up);
    digitalWrite(UP_PIN, HIGH);
   }
```

```
      else if (receivedDirection[0] == 'd'){
        Serial.print("Going down ");
        Serial.print(distance);
        Serial.println(" cm");

        digitalWrite(DOWN_PIN, LOW);
        delay(distance*adjustment*ms_per_cm_down);
        digitalWrite(DOWN_PIN, HIGH);
      }

      else{
        Serial.print(receivedDirection);
        Serial.println("Command not understood");
      }
    newCommand = false;
    }
  }
}
```

D GUI Code

link.

D.1 gui.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon Aug 27 09:59:05 2018

@author: ngw861
"""
import os
import sys
import time

from PyQt5 import QtGui, QtCore, QtWidgets

from guis import captureThread
from guis.plotsGUI import plotsGUI
from guis.basicGUI import basicGUI
from guis.checksGUI import checksGUI
from guis.configGUI import configGUI
from guis.liveViewGUI import liveViewGUI
from guis.takePhotosGUI import takePhotosGUI
#from guis.calibrateGUI import calibrateGUI
from guis.progressDialog import progressDialog
from guis.instructionsGUI import instructionsGUI



class GUI(basicGUI):
    def __init__(self):
        super(GUI, self).__init__()
        self.progress = progressDialog()
        self.progress._open()
        self.progress.update(20,'Getting␣Coffee..')
        self.instructions = instructionsGUI()
        self.progress.update(30,'Checking␣Appendages..')
        self.checks = checksGUI()

        self.progress.update(50,'Getting␣Dressed..')
        self.live_view = liveViewGUI()
        #self.plots = plotsGUI()
        self.takePhoto = takePhotosGUI()
        self.config = configGUI()
        #self.calibrate = calibrateGUI()
        #self.progress.close()
        self.initUI()


    def initUI(self):
        self.setWindowTitle('Upload␣Image␣to␣Database')
        self.setWindowIcon(QtGui.QIcon('icon.png'))

        self.grid.addWidget(self.instructions, 0, 0)
        self.grid.addWidget(self.config, 1, 0, 2, 1)
        #self.grid.addWidget(self.plots, 3, 0, 6, 1)
        self.grid.addWidget(self.live_view, 0, 1, 8, 1)
        self.grid.addWidget(self.takePhoto, 8, 1, 1, 1)

        self.setLayout(self.grid)
        self.show()
        self.progress._close()



if __name__ == '__main__':
    global captureThread
    captureThread.init()

    QtCore.QCoreApplication.addLibraryPath(os.path.join(os.path.
        ↪ dirname(QtCore.__file__), "plugins"))

    app = QtWidgets.QApplication(sys.argv)
    app.setWindowIcon(QtGui.QIcon('icon.png'))
    gui = GUI()
    sys.exit(app.exec_())
```

D.2 erdaBackgroundUpload.py

```
#Background process

#run on infinite loop inside try function
#Make sure sftp is mounted
#Open local storage
#Create tiff from img
#copy from local to erda
#Do checksum
#delete local copy
import os
import pysftp
import logging
import paramiko
import subprocess

from time import sleep
from base64 import b64decode
from guis.settings.local_settings import (SFTP_PUBLIC_KEY,
    ↪ ERDA_USERNAME,
                                          ERDA_SFTP_PASSWORD, ERDA_HOST,
                                          ERDA_PORT, ERDA_FOLDER, DUMP_FOLDER
                                              ↪ ,
                                          CACHE_FOLDER, STORAGE_FOLDER)




class ERDA():
    def __init__(self):
        self.sftp = self.connectSFTP()

    def connectSFTP(self):
        key = paramiko.RSAKey(data=b64decode(SFTP_PUBLIC_KEY))
        cnopts = pysftp.CnOpts()
        cnopts.hostkeys.add(ERDA_HOST, 'ssh-rsa', key)

        sftp = pysftp.Connection(host=ERDA_HOST, username=
            ↪ ERDA_USERNAME,
                                    password=ERDA_SFTP_PASSWORD, cnopts=
                                        ↪ cnopts)
        return sftp

    def upload(self, localPath, remotePath):
        self.sftp.put(localPath, remotePath)

    def getFiles(self, folder):
        return self.sftp.listdir(folder)

    def checkUploaded(self, erdaPath, cachePath):
        erdaFolder = ’/’.join(erdaPath.split(’/’)[:-1])
        files = self.getFiles(erdaFolder)
        cacheFile = cachePath.split(’/’)[-1]

        if cacheFile in files:
            logging.info(’ERDA␣Upload␣okay␣for␣%s’%cacheFile)
            return True
        else:
            logging.info(’Something␣messed␣up␣%s␣’%cacheFile)
            return False

    def close(self):
        self.sftp.close()



def createTiff(arwPath):
    name = arwPath.split(’/’)[-1].split(’.’)[0]
    tiffFile = name + ’.tiff’
    tiffPath = os.path.join(CACHE_FOLDER, tiffFile)

    subprocess.check_output([’sips’, ’-s’,’format’,’tiff’, arwPath, ’
        ↪ --out’, tiffPath])
    return tiffPath

def getARWFiles(folder):
    ARWFiles = [f for f in getFiles(folder) if f.endswith(’.arw’)]
    if len(ARWFiles):
        paths = [os.path.join(folder, image) for image in ARWFiles]
        paths.sort(key=os.path.getctime)
        return [path.split(’/’)[-1] for path in paths]
    else:
        return ’’, ’’

def getFiles(folder):
    return os.listdir(folder)

def checkSum(file1, file2):
    pass

def deleteFile(_file):
    return os.remove(_file)

def deleteNonARWFilesFromLocalCache():
    local_files = getFiles(CACHE_FOLDER)
    for local_file in local_files:
        if (not local_file.endswith(’.arw’)) & (not local_file.
            ↪ endswith(’.DS_Store’)):
            path = os.path.join(CACHE_FOLDER, local_file)
            logging.info(’Deleting␣%s’%path)
            deleteFile(path)

if __name__ == ’__main__’:
    while True:
        sleep(1)
        try:
            logging.basicConfig(filename=’/Users/robertahunt/
                ↪ Documents/gui/logs/erda_upload_log.txt’, level=
                ↪ logging.INFO)

            deleteNonARWFilesFromLocalCache()

            logging.info(’Getting␣lists␣of␣files’)
            local_files = getARWFiles(CACHE_FOLDER)
            logging.info(local_files)

            if len(local_files):
                logging.info(’Starting␣ERDA’)
                erda = ERDA()

                erda_files = erda.getFiles(ERDA_FOLDER)
            for local_file in local_files:
                logging.info(’looking␣at␣file:␣%s’%local_file)
                arwCachePath = os.path.join(CACHE_FOLDER,
                    ↪ local_file)
                logging.info(’creating␣tiff’)
                tiffCachePath = createTiff(arwCachePath)
                tiff_name = tiffCachePath.split(’/’)[-1]

                arwLocalPath = os.path.join(STORAGE_FOLDER,
                    ↪ local_file)
                tiffLocalPath = os.path.join(STORAGE_FOLDER,
                    ↪ tiff_name)

                #what happens if file already uploaded


                arwERDAPath = os.path.join(ERDA_FOLDER, local_file
                    ↪ )
                tiffERDAPath = os.path.join(ERDA_FOLDER, tiff_name
                    ↪ )

                erda.upload(tiffCachePath, tiffERDAPath)
                uploadedTiff = erda.checkUploaded(tiffERDAPath,
                    ↪ tiffCachePath)

                erda.upload(arwCachePath, arwERDAPath)
                uploadedARW = erda.checkUploaded(arwERDAPath,
                    ↪ arwCachePath)


                if uploadedARW & uploadedTiff:
                    logging.info(’Deleting␣cached␣files’)
                    deleteFile(arwCachePath)
                    deleteFile(tiffCachePath)

            erda.close()
    except Exception as ex:
        if ’erda’ in locals():
            if hasattr(erda, ’sftp’):
                erda.sftp.close()

        logging.warning(’Exception␣encountered:␣’+str(ex))
```

D.3 guis/basicGUI.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sun Sep  2 19:27:30 2018

@author: robertahunt
"""
import sys
import time
import warnings
import subprocess

from guis import captureThread
from serial import Serial
from serial.tools import list_ports
from PyQt5 import QtGui, QtWidgets, QtCore

class ClickableIMG(QtWidgets.QLabel):
    clicked = QtCore.pyqtSignal(str)

    def mousePressEvent(self, event):
        self.clicked.emit(self.objectName())

class basicGUI(QtWidgets.QWidget):
    def __init__(self):
        super(basicGUI, self).__init__()
        self.grid = QtWidgets.QGridLayout()
        self.grid.setSpacing(10)

    def headerLabel(self, text):
        headerLabel = QtWidgets.QLabel(text)
        headerFont = QtGui.QFont("Times", 20, QtGui.QFont.Bold)
        headerLabel.setFont(headerFont)
        return headerLabel


    def commandLine(self, args):
        assert isinstance(args,list)
        print(’Sent␣command:␣’ + ’␣’.join(args))
        if (args[0] == ’gphoto2’):
            captureThread.captureThread.pause()

        try:
            output = subprocess.check_output(args)
            if (args[0] == ’gphoto2’):
                captureThread.captureThread.resume()
            return output
        except Exception as ex:
            #self.warn(’Command %s failed. Exception: %s’%(’ ’.join(
                ↪ args), ex))
            if (args[0] == ’gphoto2’):
                captureThread.captureThread.resume()
            return ex

    def warn(self, msg, _exit=False):
        warnings.warn(msg)
        warning = QtWidgets.QMessageBox()
        warning.setWindowTitle(’Warning␣Encountered’)
        warning.setText(msg)
        warning.exec_()
        if _exit:
            sys.exit()


    #def closeEvent(self, event):
        #reply = QtGui.QMessageBox.question(self, ’Message’,
        # "Are you sure to quit?", QtGui.QMessageBox.Yes |
        # QtGui.QMessageBox.No, QtGui.QMessageBox.No)

        #if reply == QtGui.QMessageBox.Yes:
        # event.accept()
        #else:
        # event.ignore()

class Arduino():
    def __init__(self):
        self.port = self.getArduinoPort()
        self.ser = Serial(self.port, 9600, timeout=0.1)
    def getArduinoPort(self):
        ports = list(list_ports.comports())
        for p in ports:
            if 'Arduino' in str(p.manufacturer):
                return p.device
        else:
            return None

    def moveCamera(self, direction, cm):
        i = 0
        assert direction in ['u','d']
        self.ser.write("%s␣%s\n"%(direction,cm))
        #while True:
        # print('Trying to write to arduino')
        # if (self.ser.inWaiting()>0): # Check if board available
        # self.ser.write("%s %s\n"%(direction,cm))
        # break
        # i+=1
        # time.sleep(0.1)
        # if i > 50:
        #
        # self.ser.write("%s %s\n"%(direction,cm))
        # self.warn('Could not move camera... try restarting arduino
            ↪ ')
        # break

    def readline(self):
        return self.ser.readline()

    def getHeight(self):
        while True:
            line = self.ser.readline()
            if 'mm' in line:
                return float(line.split('mm')[0])
            print(line)

    def cameraUpMm(self):
        self.moveCamera('u','0.1')
    def cameraUpCm(self):
        self.moveCamera('u','1')
    def cameraDownMm(self):
        self.moveCamera('d','0.1')
    def cameraDownCm(self):
        self.moveCamera('d','1')

    def close(self):
        self.ser.close()
```

D.4 guis/captureThread.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Tue Sep  4 10:28:02 2018

@author: robertahunt
"""
import os
import subprocess

from time import sleep
from PyQt5 import QtCore
from guis.settings.local_settings import DUMP_FOLDER

#from basicGUI import basicGUI

class capturePreview(QtCore.QThread):
    def __init__(self):
        QtCore.QThread.__init__(self)
        self.paused = False

    def run(self):
        while( self.isRunning() ):
            if self.paused == False:
                self.running = True
                try:
                    subprocess.check_output(['gphoto2','--capture-
                        ↪ preview',
                                              '--force-overwrite','--
                                                  ↪ filename',
                                              os.path.join(DUMP_FOLDER,
                                                           'preview.jpg')]
                                              )
                except Exception as ex:
                    pass
                self.running = False
                #print('Updated Preview')
            sleep(1)

    def pause(self):
        #print('Capture Paused')
        self.paused = True
        while self.running == True:
            sleep(1)
        return True

    def resume(self):
        #print('Capture Resumed')
        self.paused = False

def init():
    global captureThread
    captureThread = capturePreview()
    captureThread.start()
```

D.5 guis/checksGUI.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 7 17:28:17 2018

@author: robertahunt
"""
import time

from serial import Serial
from PyQt5 import QtCore, QtWidgets

from guis.basicGUI import basicGUI, Arduino
from guis.progressDialog import progressDialog



class checksGUI(basicGUI):
```

```
def __init__(self):
    super(checksGUI, self).__init__()

    self.progress = progressDialog()
    self.progress._open()

    self.progress.update(0,'Trying␣to␣Detect␣Camera.')
    self.testDetectCamera()

    self.progress.update(20,'Testing␣Camera␣Connection')
    self.testCameraConnection()

    self.progress.update(40,'Testing␣Arduino␣Connection')
    self.testArduinoConnection()

    #self.progress.update(60,'Testing Sonar Data Collection')
    #self.testGetSonarData()

    #self.progress.update(80,'Testing Camera Movement')
    #self.testMoveCamera()

    self.progress.update(100,'Done')
    self.arduino.close()
    self.progress.close()


def testDetectCamera(self):
    try:
        auto_detect_output = 'Cameras␣Detected:␣\n%s'%self.
            ↪ commandLine(['gphoto2','--auto-detect'])
        if len(auto_detect_output) <= 126:
            warning = 'Error␣xkcd1314:␣No␣cameras␣detected.␣Please
                ↪ ␣make␣sure␣camera␣is␣on␣and␣connected␣to␣the␣
                ↪ computer.'
            self.warn(warning, _exit=True)
    except Exception as ex:
        print(str(ex))

def testCameraConnection(self):
    try:
        output = self.commandLine(['gphoto2','--list-all-config'])
            ↪
        if ’Error’ in str(output):
            self.warn(’’’Error xkcd806: There is a problem sending
                ↪  commands to camera. Please check usb cable,
                ↪ unplug and replug and try again.’’’, _exit=True)
                ↪
    except Exception as ex:
        print(str(ex))



def testArduinoConnection(self):
    try:
        self.arduino = Arduino()
    except Exception as ex:
        print(str(ex))
        self.warn(’Error␣xkcd730:␣Unable␣to␣connect␣to␣Arduino.␣
            ↪ Please␣make␣sure␣arduino␣has␣power␣and␣is␣connected
            ↪ ␣to␣the␣computer.’,_exit=True)



def testGetSonarData(self):
    try:
        height = self.arduino.getHeight()
    except Exception as ex:
        print(str(ex))
        self.warn(’Error␣xkcd1590:␣Unable␣to␣read␣data␣from␣sonar
            ↪ ␣sensor.␣Please␣make␣sure␣it␣is␣connected␣to␣the␣
            ↪ big␣black␣box.’,_exit=True)

def testMoveCamera(self):
    try:
        height_1 = self.arduino.getHeight()
        print(height_1)
        time.sleep(4)
        self.arduino.moveCamera(’u’,’3’)
        time.sleep(5)
        height_2 = self.arduino.getHeight()
        print(height_2)
        if height_2 - height_1 < 0.9:
            print(height_1, height_2)
            self.arduino.moveCamera(’d’,’3’)
            self.warn('Error␣xkcd1428:␣It␣seems␣we␣cannot␣move␣the
                ↪ ␣camera␣using␣the␣arduino....␣That␣is␣not␣good.'
                ↪ , _exit=True)
        self.arduino.moveCamera('d','4')
    except Exception as ex:
        print(str(ex))
        self.arduino.moveCamera('d','3')
        self.warn('Error␣xkcd510:␣Something␣went␣horribly␣wrong.␣
            ↪ Goodbye', _exit=True)
```

D.6 guis/configGUI.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 2 20:03:22 2018

@author: robertahunt
"""
import functools
import subprocess

from time import sleep
from PyQt5 import QtWidgets
from guis.basicGUI import basicGUI

defaultConfig = {'/main/capturesettings/expprogram':'M', #not
    ↪ controllable
                 '/main/status/vendorextension':'Sony␣PTP␣Extensions',
                     ↪  #not controllable
                 '/main/capturesettings/imagequality':'RAW', #
                     ↪ Controllable
                 '/main/actions/opcode':'0x1001,0xparam1,0xparam2', #
                     ↪ not controllable
                 '/main/capturesettings/flashmode':'Flash␣off', #
                     ↪ Somewhat controllable
                 '/main/imgsettings/whitebalance':'Preset␣1', #
                 '/main/imgsettings/colortemperature':'5200', #
                     ↪ controllable if whitebalance set to 'Choose
                     ↪ Color Temperature'
```

```
               '/main/capturesettings/exposurecompensation':'0', #
                   ↪ not controllable
               '/main/capturesettings/exposuremetermode':'Unknown␣
                   ↪ value␣8001', #Seems controllable
               '/main/status/cameramodel':'ILCE-7RM3', #not
                   ↪ controllable
               '/main/status/batterylevel':'98%', #not controllable
               '/main/capturesettings/f-number':'11', #Controllable
               '/main/imgsettings/imagesize':'Large', #Controllable
               '/main/capturesettings/aspectratio':'3:2', #
                   ↪ Controllable
               '/main/status/deviceversion':'1.0', #not controllable
               '/main/actions/capture':'2', #not controllable
               '/main/status/serialnumber':'
                   ↪ 00000000000000003282933003783803', #not
                   ↪ controllable
               '/main/capturesettings/shutterspeed':'1/15', #
                   ↪ Controllable
               '/main/actions/movie':'2', #not controllable
               '/main/actions/bulb':'2', #not controllable
               '/main/capturesettings/focusmode':'Manual', #not
                   ↪ controllable
               '/main/actions/manualfocus':'0', #not controllable
               '/main/status/manufacturer':'Sony␣Corporation', #not
                   ↪ controllable
               '/main/imgsettings/iso':'200', #Controllable
               '/main/actions/autofocus':'2', #not controllable
               '/main/capturesettings/capturemode':'Single␣Shot'} #
                   ↪ Controllable

fNumbers = ['2.8', '3.2', '3.5', '4.0', '4.5', '5.0', '5.6', '6.3', '
    ↪ 7.1',
           '9.8', '10', '11', '13', '14', '16', '18', '20', '22']

shutterSpeeds = ['1/8000','1/6400','1/5000','1/4000','1/3200','
    ↪ 1/2500',
               '1/2000','1/1600','1/1250','1/1000','1/800','1/640',
               '1/500','1/400','1/320','1/250','1/200','1/160','
                   ↪ 1/125',
               '1/100','1/80','1/60','1/50','1/40','1/30','1/25','
                   ↪ 1/20',
               '1/15','1/13','1/10','1/8','1/6','1/5','1/4','1/3','
                   ↪ 0.4',
               '0.5','0.6','0.8','1','1.3','1.6','2','2.5','3.2','4'
                   ↪ ,'5',
               '6','8','10','13','15','20','25','30']

colorTemperatures = ['0'] + [str(x) for x in list(range
    ↪ (3000,8000,100))]

class configGUI(basicGUI, QtWidgets.QMainWindow):
    def __init__(self):
        super(configGUI, self).__init__()
        self.configOptions = self.getConfigOptions()
        self.widgets = {}
        self.initUI()



    #def colorTempWidgets(self):
    # path = '/main/imgsettings/colortemperature'
    # config = self.configOptions[path]
    # labelWidget = QtWidgets.QLabel(config['Label'])
    #
    # editWidget = QtWidgets.QLineEdit()
    # editWidget.setText(config['Current'])
    # editWidget.editingFinished.connect(functools.partial(self.
        ↪ updateValue, path))
    #
    # editWidget.name = path
    # self.widgets[path] = editWidget
    # return labelWidget, editWidget
    def colorTempWidgets(self):
        path = '/main/imgsettings/colortemperature'
        return self.makeWidgetsFromPath(path)



    def fNumberWidgets(self):
        path = '/main/capturesettings/f-number'
        return self.makeWidgetsFromPath(path)

    def imageSizeWidgets(self):
        path = '/main/imgsettings/imagesize'
        return self.makeWidgetsFromPath(path)
```

```
def shutterSpeedWidgets(self):
    path = '/main/capturesettings/shutterspeed'
    return self.makeWidgetsFromPath(path)

def isoWidgets(self):
    path = '/main/imgsettings/iso'
    return self.makeWidgetsFromPath(path)

def captureModeWidgets(self):
    path = '/main/capturesettings/capturemode'
    return self.makeWidgetsFromPath(path)

def imgQualityWidgets(self):
    path = '/main/capturesettings/imagequality'
    return self.makeWidgetsFromPath(path)

def whiteBalanceWidgets(self):
    path = '/main/imgsettings/whitebalance'
    return self.makeWidgetsFromPath(path)

def flashModeWidgets(self):
    path = '/main/capturesettings/flashmode'
    return self.makeWidgetsFromPath(path)

def makeWidgetsFromPath(self, path):
    config = self.configOptions[path]
    labelWidget = QtWidgets.QLabel(config['Label'])

    editWidget = QtWidgets.QComboBox()
    editWidget.addItems(config['Choices'])

    currentIndex = self.getCurrentOptionIndex(path)

    editWidget.setCurrentIndex(currentIndex)
    if path in ['/main/capturesettings/f-number',
               '/main/capturesettings/shutterspeed',
               '/main/imgsettings/colortemperature']:
        editWidget.activated.connect(functools.partial(
               self.updateValueByIndex, path))
    else:
        editWidget.activated.connect(functools.partial(
               self.updateConfigOptionByIndex, path))
    editWidget.name = path
    self.widgets[path] = editWidget
    return labelWidget, editWidget


def initUI(self):
    iso, self.isoEdit = self.isoWidgets()
    fNumber, self.fNumberEdit = self.fNumberWidgets()
    flashMode, self.flashModeEdit = self.flashModeWidgets()
    colorTemp, self.colorTempEdit = self.colorTempWidgets()
    #imageSize, self.imageSizeEdit = self.imageSizeWidgets()
    #imgQuality, self.imgQualityEdit = self.imgQualityWidgets()
    captureMode, self.captureModeEdit = self.captureModeWidgets()
    whiteBalance, self.whiteBalanceEdit = self.
        ↪ whiteBalanceWidgets()
    shutterSpeed, self.shutterSpeedEdit = self.
        ↪ shutterSpeedWidgets()

    setDefaultButton = QtWidgets.QPushButton(’Reset␣to␣Default’)
    setDefaultButton.clicked.connect(self.setDefaultOptions)
    showCurrentButton = QtWidgets.QPushButton(’Show␣Current’)
    showCurrentButton.clicked.connect(self.showCurrent)

    #self.grid.addWidget(imgQuality, 0, 0)
    #self.grid.addWidget(self.imgQualityEdit, 0, 1)
    #self.grid.addWidget(imageSize, 1, 0)
    #self.grid.addWidget(self.imageSizeEdit, 1, 1)
    self.grid.addWidget(fNumber, 2, 0)
    self.grid.addWidget(self.fNumberEdit, 2, 1)
    self.grid.addWidget(shutterSpeed, 3, 0)
    self.grid.addWidget(self.shutterSpeedEdit, 3, 1)
    self.grid.addWidget(iso, 4, 0)
    self.grid.addWidget(self.isoEdit, 4, 1)
    self.grid.addWidget(whiteBalance, 5, 0)
    self.grid.addWidget(self.whiteBalanceEdit, 5, 1)
    self.grid.addWidget(colorTemp, 6, 0)
    self.grid.addWidget(self.colorTempEdit, 6, 1)
    self.grid.addWidget(captureMode, 7, 0)
    self.grid.addWidget(self.captureModeEdit, 7, 1)
    self.grid.addWidget(flashMode, 8, 0)
    self.grid.addWidget(self.flashModeEdit, 8, 1)
    self.grid.addWidget(setDefaultButton, 9, 1)
    #self.grid.addWidget(showCurrentButton, 10, 0)

    self.showCurrent()

    self.setLayout(self.grid)

    #self.widgets[’/main/imgsettings/colortemperature’].
        ↪ clearFocus()


def showCurrent(self):
    sleep(0.5)

    self.configOptions = self.getConfigOptions()

    for path in self.widgets.keys():
        widget = self.widgets[path]
        widget_class = widget.__class__.__name__
        if widget_class in [’QComboBox’, ’QLineEdit’]:
           path = widget.name
           actual = self.configOptions[path][’Current’]
           if False:#path == ’/main/imgsettings/colortemperature
               ↪ ’:
               widget.setText(actual)
           else:
               widget.setCurrentIndex(self.getCurrentOptionIndex(
                   ↪ path))

def getCurrentOptionIndex(self, path):
    return self.configOptions[path][’Choices’].index(
                   self.configOptions[path][’Current’])

def setDefaultOptions(self):
    self.configOptions = self.getConfigOptions()
    path = ’/main/imgsettings/whitebalance’
    if defaultConfig[path] != self.configOptions[path][’Current’]:
        ↪
        index = self.widgets[path].findText(defaultConfig[path])
        self.updateValueByIndex(path, index, showCurrent=False)

    for path in self.widgets.keys():
```

```
        if defaultConfig[path] != self.configOptions[path]['
            ↪ Current']:
            if False:#path in ['/main/imgsettings/colortemperature
                ↪ ']:
                self.updateValue(path, defaultConfig[path],
                    ↪ showCurrent=False)
            elif path in ['/main/capturesettings/f-number',
                        '/main/capturesettings/shutterspeed',
                        '/main/imgsettings/colortemperature']:
                index = self.widgets[path].findText(defaultConfig[
                    ↪ path])
                self.updateValueByIndex(path, index, showCurrent=
                    ↪ False)
            else:
                self.updateConfigOptionByName(path, defaultConfig[
                    ↪ path], showCurrent = False)
    self.showCurrent()

def updateConfigOptionByIndex(self, path, choice_index,
    ↪ showCurrent = True):
    self.commandLine(['gphoto2','--set-config-index',path+'='+str
        ↪ (choice_index)])

    if path == '/main/imgsettings/whitebalance':
        if self.widgets[path].itemText(choice_index) != 'Choose␣
            ↪ Color␣Temperature':
            pass
            #self.updateValue('/main/imgsettings/colortemperature
                ↪ ', '0')
    if showCurrent == True:
        self.showCurrent()

def updateConfigOptionByName(self, path, choice, showCurrent =
    ↪ True):
    choice_index = self.configOptions[path]['Choices'].index(
        ↪ choice)
    return self.updateConfigOptionByIndex(path, choice_index,
        ↪ showCurrent)

def updateValueByIndex(self, path, index, showCurrent = True):
    value = self.widgets[path].itemText(index)
```

```
        self.commandLine(['gphoto2','--set-config-value',path+'='+str
            ↪ (value)])
        if showCurrent == True:
            self.showCurrent()

    def updateValue(self, path, value = None, showCurrent = True):
        if value is None:
            value = self.widgets[path].text()
        if path == '/main/imgsettings/colortemperature':
            if self.widgets['/main/imgsettings/whitebalance'].
                ↪ itemText(self.widgets['/main/imgsettings/
                ↪ whitebalance'].currentIndex()) != 'Choose␣Color␣
                ↪ Temperature':
                value = 0
        self.commandLine(['gphoto2','--set-config-value',path+'='+str
            ↪ (value)])
        if showCurrent == True:
            self.showCurrent()
        self.widgets[path].clearFocus()

    def getConfigOptions(self):
        raw_config = self.commandLine(['gphoto2','--list-all-config'
            ↪ ])
        config = {}
        if isinstance(raw_config, subprocess.CalledProcessError):
            self.warn('Could␣not␣get␣config␣params␣from␣camera.␣
                ↪ Restart␣camera␣and␣try␣again', _exit=True)

        for line in raw_config.split('\n'):
            if 'other' in line:
                break

            if line.startswith('/'):
                name = line
                config[name] = {}
                config[name]['Choices'] = []
            elif line.startswith('Label:'):
                config[name]['Label'] = line[7:]
            elif line.startswith('Readonly:'):
                config[name]['Readonly'] = line[10:]
            elif line.startswith('Type:'):
                config[name]['Type'] = line[6:]
            elif line.startswith('Current:'):
                config[name]['Current'] = line[9:]
            elif line.startswith('Choice:'):
                choice = '␣'.join(line[8:].split('␣')[1:])
                config[name]['Choices'] += [choice]
            elif line.startswith('Bottom:'):
                config[name]['Bottom'] = float(line[8:])
            elif line.startswith('Top:'):
                config[name]['Top'] = float(line[5:])
            elif line.startswith('Step:'):
                config[name]['Step'] = float(line[6:])

        config['/main/capturesettings/f-number']['Choices'] =
            ↪ fNumbers
        config['/main/capturesettings/shutterspeed']['Choices'] =
            ↪ shutterSpeeds
        config['/main/imgsettings/colortemperature']['Choices'] =
            ↪ colorTemperatures
        return config
```

D.7 guis/imageViewGUI.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 2 20:43:32 2018

@author: robertahunt
"""
import os
import cv2
import time
import Queue
import rawpy
import serial
import pysftp
import paramiko
import threading
import numpy as np
import pandas as pd
```

```
from serial import Serial
from pyzbar import pyzbar
from base64 import b64decode
from PyQt5 import QtWidgets, QtCore, QtGui
from basicGUI import basicGUI, ClickableIMG, Arduino
from settings.local_settings import (SFTP_PUBLIC_KEY, ERDA_USERNAME,
                                    ERDA_SFTP_PASSWORD, ERDA_HOST,
                                    ERDA_PORT, ERDA_FOLDER,
                                        ↪ TEMP_IMAGE_CACHE_PATH,
                                    LOCAL_IMAGE_STORAGE_PATH)

global start_time

def start_timer():
    global start_time
    start_time = pd.Timestamp.now()

def tick(msg = ’’):
    global start_time
    print(msg + ’,␣Time␣Taken:␣%s’%(pd.Timestamp.now()-start_time))

class imageViewGUI(basicGUI, QtWidgets.QMainWindow):
    def __init__(self):
        super(imageViewGUI, self).__init__()
        self.arduino = Arduino()
        self.PREVIEW_WIDTH = 1024//2
        self.PREVIEW_HEIGHT = 680//2

        self.TEMP_FOLDER = TEMP_IMAGE_CACHE_PATH
        self.tempPath, self.tempName = self.getLatestImageName(self.
            ↪ TEMP_FOLDER)
        self.newImgName = ’’
        self.imgSuffix = ’0’
        self.initUI()


    def initUI(self):
        self.imgView = ClickableIMG(self)
        self.imgView.setMinimumSize(self.PREVIEW_WIDTH, self.
            ↪ PREVIEW_HEIGHT)
        self.imgView.clicked.connect(self.openIMG)
        header = self.headerLabel('Latest␣image')
        self.imgDesc = QtWidgets.QLabel('Latest␣image␣in␣folder:␣%s'%
            ↪ self.tempPath)
        self.newImgNameLabel = QtWidgets.QLabel('New␣image␣name:␣%s'%
            ↪ self.newImgName)
        sendButton = QtWidgets.QPushButton("Send␣to␣Datastorage")
        sendButton.clicked.connect(self.sendToERDA)
        sendButton.clicked.connect(self.sendToLocalDir)

        self.QRCodeLabel = QtWidgets.QLabel()

        label = 'Manual␣Catalog␣Number␣Entry␣(used␣only␣if␣QR␣Code␣
            ↪ not␣found):␣'
        QRCodeManualLabel = QtWidgets.QLabel(label)
        self.QRCodeManualEdit = QtWidgets.QLineEdit()

        self.grid.addWidget(header, 0, 0, 1, 2)
        self.grid.addWidget(self.imgDesc, 1, 0, 1, 2)
        self.grid.addWidget(self.QRCodeLabel, 2, 0, 1, 2)
        self.grid.addWidget(QRCodeManualLabel, 3, 0, 1, 1)
        self.grid.addWidget(self.QRCodeManualEdit, 3, 1, 1, 1)
        self.grid.addWidget(self.imgView, 4, 0, 1, 2)
        self.grid.addWidget(self.newImgNameLabel, 5, 0, 1, 2)
        self.grid.addWidget(sendButton, 6, 0, 1, 2)
        self.setLayout(self.grid)

    def getIMG(self):
        _format = self.tempPath.split('.')[-1]
        #if _format == 'jpg':
        # return cv2.imread(self.tempPath)
        if _format == 'arw':
            with rawpy.imread(self.tempPath) as raw:
                return raw.postprocess()
        else:
            self.warn('Image␣format␣in␣folder␣not␣understood.%s'%
                ↪ _format)

    def sendToERDA(self, tempPath=None, imgName=None):
        if not len(tempPath):
            tempPath = self.tempPath
        if not len(imgName):
            imgName = self.newImgName
        if len(str(int(self.QRCode))) != 6:
            try:
                if len(str(int(self.QRCodeManualEdit.text()))) == 6:
                    self.QRCode = len(str(int(self.QRCodeManualEdit.
                        ↪ text())))
                else:
                    self.warn('QR␣Code␣not␣recognized␣in␣image,␣and␣
                        ↪ manual␣catalog␣number␣entry␣invalid')
                    return None
            except:
                self.warn('QR␣Code␣not␣recognized␣in␣image,␣and␣manual
                    ↪ ␣catalog␣number␣entry␣invalid')

        key = paramiko.RSAKey(data=b64decode(SFTP_PUBLIC_KEY))
        cnopts = pysftp.CnOpts()
        cnopts.hostkeys.add(ERDA_HOST, 'ssh-rsa', key)

        sftp = pysftp.Connection(host=ERDA_HOST, username=
            ↪ ERDA_USERNAME,
                               password=ERDA_SFTP_PASSWORD, cnopts=
                                   ↪ cnopts)

        remote_path = os.path.join(ERDA_FOLDER, imgName)
        sftp.put(tempPath,remote_path)
        sftp.close()
        self.close()

    def sendToLocalDir(self):
        pass


    def getQRCode(self):

        decoded_list = pyzbar.decode(cv2.resize(self.img,(1024,680)))
        for decoded in decoded_list:
            if decoded.type == 'QRCODE':
                return decoded.data
        else:
            return ''

    def openIMG(self):
        self.commandLine([’open’, self.tempPath])

    def getLatestImageName(self, image_folder):
        images = [image for image in os.listdir(image_folder) if
            ↪ image.split(’.’)[-1] in [’arw’]]
        if len(images):
            latest_image_path = max([os.path.join(image_folder, image)
                ↪  for image in images], key=os.path.getctime)
            return latest_image_path, latest_image_path.split(’/’)
                ↪ [-1]
        else:
            return ’’, ’’

    def displayLatestImg(self):
        self.tempPath, self.tempName = self.getLatestImageName(self.
            ↪ TEMP_FOLDER)
        self.imgDesc.setText(’Latest␣image␣in␣folder:␣%s’% self.
            ↪ tempPath)
        self.img = self.getIMG()

        _format = self.tempPath.split(’.’)[-1]
        #if _format == ’arw’:
        imgResized = QtGui.QImage(self.img.data, self.img.shape[1],
            ↪ self.img.shape[0],
                            self.img.shape[1]*3, QtGui.QImage.
                                ↪ Format_RGB888)
        imgResized = QtGui.QPixmap.fromImage(imgResized).scaled(self.
            ↪ PREVIEW_WIDTH, self.PREVIEW_HEIGHT,
                                              QtCore.Qt.
                                                  ↪ KeepAspectRatio)
        #elif _format == ’jpg’:
        # imgResized = QtGui.QPixmap(self.tempPath).scaled(self.
            ↪ PREVIEW_WIDTH, self.PREVIEW_HEIGHT,
        # QtCore.Qt.KeepAspectRatio)
        self.imgView.setPixmap(imgResized)

        self.QRCode = str(self.getQRCode())
        self.QRCodeLabel.setText(’QR␣Code␣/␣Catalog␣Number:␣’ + self.
            ↪ QRCode)
        self.newImgName = self.QRCode + ’_’ + self.tempName
        self.newImgNameLabel.setText(’New␣image␣name:␣%s’% self.
            ↪ newImgName)
    def takePhoto(self, imgName=None):
        start_timer()
        print(imgName)
        if (imgName is None) | (imgName == False):
            imgName = 'capture.arw'

        os.chdir(self.TEMP_FOLDER)
        print(imgName)
        self.commandLine(['gphoto2', '--capture-image-and-download',
                        '--force-overwrite', '--filename', imgName])
        tick('Done␣Taking␣Photo')

        #start_timer()
        #self.tempPath, self.tempName = self.getLatestImageName(self.
            ↪ TEMP_FOLDER)
        #name, fileformat = self.tempName.split('.')
        #if len(name):
        # new_name = os.path.join(TEMP_IMAGE_CACHE_PATH, name + '.
            ↪ tiff')
        # self.commandLine(['sips', '-s','format','tiff',self.
            ↪ tempPath, '--out',new_name])
        #tick('Done Converting Photo to tiff')



    def takeStackedPhotos(self):
        n_photos = 10
        QRCode = ''
        timestamp = time.strftime('%Y%m%d_%H%M%S', time.gmtime())

        #Start taking photos
        print('Taking␣pics')
        for i in range(n_photos):
            start_timer()
            tempImgName = 'Stacked_'+str(i)+'.arw'
            self.takePhoto(imgName=tempImgName)
            time.sleep(0.1)
            self.arduino.moveCamera('d','0.2')


            self.img = self.getIMG()
            self.QRCode = self.getQRCode()
            if len(self.QRCode):
                QRCode = self.QRCode
            print(self.QRCode)
            tick('Done␣taking␣one␣photo␣for␣stack')
#
# print('Copying to Local Storage')
# if len(QRCode):
# for i in range(n_photos):
# start_timer()
# tempRawPath = os.path.join(self.TEMP_FOLDER, 'Stacked_'+str(i)+'.
   ↪ arw')
# tempTiffPath = os.path.join(self.TEMP_FOLDER, 'Stacked_'+str(i)+'.
   ↪ tiff')
#
# rawImgName = 'NHMD' + QRCode + '_' + timestamp + '_' + str(i) + '.
   ↪ arw'
# tiffImgName = 'NHMD' + QRCode + '_' + timestamp + '_' + str(i) + '.
   ↪ tiff'
#
# rawImgPath = os.path.join(LOCAL_IMAGE_STORAGE_PATH, rawImgName)
# tiffImgPath = os.path.join(LOCAL_IMAGE_STORAGE_PATH, tiffImgName)
#
# self.commandLine(['cp',tempRawPath,rawImgPath])
# tick('Done copying one arw locally')
# start_timer()
# self.commandLine(['cp',tempTiffPath,tiffImgPath])
# tick('Done copying one tiff locally')
#
# print('Sending to ERDA')
# if len(QRCode):
# for i in range(n_photos):
# start_timer()
# print('Photo %s'%i)
# tempRawPath = os.path.join(self.TEMP_FOLDER, 'Stacked_'+str(i)+'.
   ↪ arw')
# tempTiffPath = os.path.join(self.TEMP_FOLDER, 'Stacked_'+str(i)+'.
   ↪ tiff')
#
# rawImgName = 'NHMD' + QRCode + '_' + timestamp + '_' + str(i) + '.
   ↪ arw'
```

```
# tiffImgName = 'NHMD' + QRCode + '_' + timestamp + '_' + str(i) + '.
    ↪ tiff'
#
# self.sendToERDA(tempRawPath, rawImgName)
# tick('Done sending one arw to ERDA')
# self.sendToERDA(tempTiffPath, tiffImgName)
# tick('Done sending one tiff to ERDA')

        print('Moving␣camera␣back␣to␣place')
        #Move camera back to place
        start_timer()
        for i in range(n_photos):
            self.arduino.moveCamera('u',str(n_photos*0.2))
            time.sleep(0.25)
        tick('Done␣Moving␣Camera␣Back')




class takePhotoGUI(basicGUI):
    def __init__(self):
        super(takePhotoGUI, self).__init__()
        self.initUI()

    def initUI(self):
        self.dialog = imageViewGUI()
        self.moveCameraUpMm = QtWidgets.QPushButton('Up␣0.1␣cm')
        self.moveCameraUpCm = QtWidgets.QPushButton('Up␣1␣cm')
        self.moveCameraDownMm = QtWidgets.QPushButton('Down␣0.1␣cm')
        self.moveCameraDownCm = QtWidgets.QPushButton('Down␣1␣cm')

        self.moveCameraUpMm.clicked.connect(self.dialog.arduino.
            ↪ cameraUpMm)
        self.moveCameraUpCm.clicked.connect(self.dialog.arduino.
            ↪ cameraUpCm)
        self.moveCameraDownMm.clicked.connect(self.dialog.arduino.
            ↪ cameraDownMm)
        self.moveCameraDownCm.clicked.connect(self.dialog.arduino.
            ↪ cameraDownCm)

        self.takePhotoButton = QtWidgets.QPushButton('Take␣New␣Photo'
            ↪ )
        self.takePhotoButton.clicked.connect(self.dialog.takePhoto)
        self.takePhotoButton.clicked.connect(self.dialog.
            ↪ displayLatestImg)
        self.takePhotoButton.clicked.connect(self._open)

        self.takeStackedPhotoButton = QtWidgets.QPushButton('Take␣New
            ↪ ␣Stacked␣Photo')
        self.takeStackedPhotoButton.clicked.connect(self.dialog.
            ↪ takeStackedPhotos)

        self.grid.addWidget(self.moveCameraUpMm, 0, 0)
        self.grid.addWidget(self.moveCameraDownMm, 0, 1)
        self.grid.addWidget(self.moveCameraUpCm, 1, 0)
        self.grid.addWidget(self.moveCameraDownCm, 1, 1)
        self.grid.addWidget(self.takePhotoButton, 2, 0, 1, 2)
        self.grid.addWidget(self.takeStackedPhotoButton, 3, 0, 1, 2)

        self.setLayout(self.grid)

    def _open(self):
        self.dialog.raise_()
        self.dialog.show()
```

D.8 guis/instructionsGUI.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 2 20:01:42 2018

@author: robertahunt
"""

from PyQt5 import QtWidgets
from guis.basicGUI import basicGUI
from guis.progressDialog import progressDialog

class instructionsGUI(basicGUI):
    def __init__(self):
        super(instructionsGUI, self).__init__()
        self.inst_title = self.headerLabel('Instructions')
        self.inst_desc = QtWidgets.QLabel(
'''
If you have any questions,
shoot Roberta a mail: XXX@XXX.ku.dk.
or if you want a timely reply, call her: XX XX XX XX



When Leaving:
Please turn off camera, computer, motor controller
and set camera battery bank to 'Charge'
''')
        self.initUI()

    def initUI(self):
        self.grid.addWidget(self.inst_title)
        self.grid.addWidget(self.inst_desc)
        self.setLayout(self.grid)
```

D.9 guis/liveViewGUI.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 2 20:40:17 2018

@author: robertahunt
"""
import os
import cv2

from pyzbar import pyzbar
from PyQt5 import QtWidgets, QtCore, QtGui
from guis.basicGUI import basicGUI, ClickableIMG
from guis.settings.local_settings import DUMP_FOLDER

class liveViewGUI(basicGUI):
    def __init__(self):
        super(liveViewGUI, self).__init__()

        self.preview_path = os.path.join(DUMP_FOLDER,'thumb_preview.
            ↪ jpg')
        self.timer = QtCore.QTimer()
        self.timer.timeout.connect(self.updatePreview)
        self.timer.start(500)
        self.initUI()

    def initUI(self):
        #self.title = self.headerLabel('Camera Preview:')
        self.preview = ClickableIMG(self)
        self.preview.setMinimumSize(1024,680)
        #self.preview.clicked.connect(self.openIMG)

        self.QRCode = QtWidgets.QLabel()

        #self.grid.addWidget(self.title, 0, 0, 1, 2)
        self.grid.addWidget(self.QRCode, 0, 0, 1, 2)
        self.grid.addWidget(self.preview, 3, 0, 1, 2)

        self.setLayout(self.grid)

    def openIMG(self):
        self.commandLine(['open',self.preview_path])

    def updatePreview(self):
        try:
            preview_img = QtGui.QPixmap(self.preview_path).scaled
                ↪ (1024,680,
                                                      QtCore.Qt.
                                                          ↪ KeepAspectRatio
                                                          ↪ )
            QRCode_text = self.getQRCode(self.preview_path)

            self.preview.setPixmap(preview_img)
            self.QRCode.setText('QR␣Code␣/␣Catalog␣Number:' +
                ↪ QRCode_text)
        except Exception as ex:
            pass
            #self.warn('Error reading' + self.preview_path + 'file. \
                ↪ n%s'%ex)

    def getQRCode(self, img_path):
        decoded_list = pyzbar.decode(cv2.imread(img_path))
        for decoded in decoded_list:
            if decoded.type == 'QRCODE':
                return decoded.data
        else:
            return 'XXXXXX'
```

D.10 guis/plotsGUI.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Sep 10 09:18:50 2018

@author: robertahunt
"""
import os
import gc
import cv2
import time
import numpy as np
import matplotlib.pyplot as plt

from PyQt5 import QtCore
from matplotlib.figure import Figure
#from matplotlib.backends.qt_compat import QtCore#, QtWidgets,
    ↪ is_pyqt5
from matplotlib.backends.backend_qt5agg import FigureCanvas

from guis.settings.local_settings import DUMP_FOLDER



from guis.basicGUI import basicGUI

class plotsGUI(basicGUI):
    def __init__(self):
        super(plotsGUI, self).__init__()

        self.width = 400
        self.height = 150
        self.path = os.path.join(DUMP_FOLDER, 'thumb_preview.jpg')
        self.n_contrast_history = 100
        self.contrast_history = np.zeros(self.n_contrast_history)
        self.contrast_x_axis = np.linspace(1,self.n_contrast_history,
                                       self.n_contrast_history)

        self._img_timer = QtCore.QTimer()
        self._img_timer.timeout.connect(self._update_img)
        self._img_timer.start(500)

        self.hist_timer = QtCore.QTimer()
        self.hist_timer.timeout.connect(self._update_hist)
        self.hist_timer.start(600)

        self.initUI()

    def initUI(self):
        self.hist_fig = Figure(figsize=(5,3))
        #self.hist_fig, self.hist_ax = plt.subplots(figsize=(5,3))
        self.hist_canvas = FigureCanvas(self.hist_fig)
        self.hist_canvas.setMinimumSize(self.width, self.height)
        self.hist_ax = self.hist_canvas.figure.subplots()
        #contrast_canvas = FigureCanvas(Figure(figsize=(5, 3)))

        #contrast_canvas.setMinimumSize(self.width, self.height)

        #self.histTimer = hist_canvas.new_timer(
        # 500, [(self._update_hist, (), {})])
        #self.histTimer.start()

        #self._contrast_ax = contrast_canvas.figure.subplots()
        #self.contrastTimer = contrast_canvas.new_timer(
        # 500, [(self._update_contrast, (), {})])
        #self.contrastTimer.start()

        #self.grid.addWidget(contrast_canvas, 1, 0)
        self.grid.addWidget(self.hist_canvas, 0, 0)
        self.setLayout(self.grid)

    def _update_img(self):
        try:
            self.img = cv2.imread(self.path)
        except:
            pass
    def _update_hist(self):
        try:
            gray = cv2.cvtColor(self.img, cv2.COLOR_BGR2GRAY).ravel()

            #self.rmhist()
            self.hist_ax.clear()
            #self.hist_ax.figure.canvas.clear()
            #self.hist_ax = self.hist_canvas.figure.subplots()

            self.hist_ax.hist(gray, 256, [0,256])
            self.hist_ax.set_title('Histogram')
            self.hist_ax.figure.canvas.draw()
        except:
            pass

    def _update_contrast(self):
        try:
            self._contrast_ax.clear()

            contrast = cv2.Laplacian(self.img, cv2.CV_64F).var()

            self.contrast_history = np.append(self.contrast_history
                ↪ [1:], [contrast])

            self._contrast_ax.plot(self.contrast_x_axis, self.
                ↪ contrast_history)
            self._contrast_ax.set_title('Contrast')
            self._contrast_ax.figure.canvas.draw()
        except:
            pass

    def rmhist(self):
        self._hist_ax.close()
        self._hist_ax.deleteLater()
        gc.collect()
```

D.11 guis/progressDialog.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
```

```
Created on Sun Sep 24 20:43:32 2018

@author: robertahunt
"""

from PyQt5 import QtWidgets, QtCore, QtGui

from guis.basicGUI import basicGUI

class progressDialog(basicGUI, QtWidgets.QMainWindow):
    def __init__(self, initial_msg = "Working."):
        super(progressDialog, self).__init__()
        self.initial_msg = initial_msg
        self.resize(len(initial_msg)*10,48)
        self.initUI()

    def initUI(self):
        self.text = QtWidgets.QLabel(self.initial_msg)
        self.progressBar = QtWidgets.QProgressBar(self)
        self.progressBar.setRange(0,100)
        self.grid.addWidget(self.text)
        self.grid.addWidget(self.progressBar)
        self.setLayout(self.grid)

    def _open(self):
        QtWidgets.QApplication.processEvents()
        self.raise_()
        self.show()
        QtWidgets.QApplication.processEvents()

    def update(self, value, text = None):
        if text is not None:
            self.text.setText(text)
            self.resize(len(text)*10,48)
        else:
            newText = self.text.text() + '.'
            self.text.setText(newText)
            self.resize(len(newText)*10,48)

        self.progressBar.setValue(value)

        QtWidgets.QApplication.processEvents()
    def _close(self):
        self.update(100)
        self.progressBar.close()
        self.close()
```

D.12 guis/takePhotosGUI.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 2 20:43:32 2018

@author: robertahunt
"""
import os
import cv2
import time
import rawpy
import serial
import pysftp
import paramiko
import threading
import numpy as np
import pandas as pd


from pyzbar import pyzbar
from base64 import b64decode
from functools import partial
from PyQt5 import QtWidgets, QtCore, QtGui
from guis.basicGUI import basicGUI, ClickableIMG, Arduino
from guis.settings.local_settings import (SFTP_PUBLIC_KEY,
    ↪ ERDA_USERNAME,
                                          ERDA_SFTP_PASSWORD, ERDA_HOST,
                                          ERDA_PORT, ERDA_FOLDER, DUMP_FOLDER
                                              ↪ ,
                                          CACHE_FOLDER, STORAGE_FOLDER)
from guis.progressDialog import progressDialog
```

```
global start_time

def start_timer():
    global start_time
    start_time = pd.Timestamp.now()

def tick(msg = ’’):
    global start_time
    print(msg + ’,␣Time␣Taken:␣%s’%(pd.Timestamp.now()-start_time))



class takePhotosGUI(basicGUI):
    def __init__(self):
        super(takePhotosGUI, self).__init__()

        self.arduino = Arduino()
        self.PREVIEW_WIDTH = 1024//4
        self.PREVIEW_HEIGHT = 680//4

        self.newImgName = ’’
        self.imgSuffix = ’0’

        self.previewPath = os.path.join(DUMP_FOLDER,’thumb_preview.
            ↪ jpg’)
        self.initUI()
        self.displayLatestImg()



    def initUI(self):
        self.imgView = ClickableIMG(self)
        self.imgView.setMinimumSize(self.PREVIEW_WIDTH, self.
            ↪ PREVIEW_HEIGHT)
        self.imgView.clicked.connect(self.openLatestIMG)

        header = self.headerLabel(’Latest␣image’)
        self.imgDesc = QtWidgets.QLabel(’Latest␣image␣in␣folder:␣%s’%
            ↪  DUMP_FOLDER)



        self.moveCameraUpMm = QtWidgets.QPushButton(’Up␣0.1␣cm’)
        self.moveCameraUpCm = QtWidgets.QPushButton(’Up␣1␣cm’)
        self.moveCameraDownMm = QtWidgets.QPushButton(’Down␣0.1␣cm’)
        self.moveCameraDownCm = QtWidgets.QPushButton('Down␣1␣cm')

        self.moveCameraUpMm.clicked.connect(self.arduino.cameraUpMm)
        self.moveCameraUpCm.clicked.connect(self.arduino.cameraUpCm)
        self.moveCameraDownMm.clicked.connect(self.arduino.
            ↪ cameraDownMm)
        self.moveCameraDownCm.clicked.connect(self.arduino.
            ↪ cameraDownCm)

        self.undersideCheckBox = QtWidgets.QCheckBox('Underside?')
        self.autoUndersideCheckBox = QtWidgets.QCheckBox('Auto␣Switch
            ↪ ␣Underside?')
        #self.takePhotoButton = QtWidgets.QPushButton('Take New Photo
            ↪ ')
        #self.takePhotoButton.clicked.connect(self.takeSinglePhoto)

        self.takeStackedPhotoButton = QtWidgets.QPushButton('Take␣New
            ↪ ␣Stacked␣Photo')
        self.takeStackedPhotoButton.clicked.connect(self.
            ↪ takeStackedPhotos)

        self.grid.addWidget(self.undersideCheckBox, 0, 0)
        self.grid.addWidget(self.autoUndersideCheckBox, 0, 1)
        self.grid.addWidget(self.moveCameraUpMm, 1, 0)
        self.grid.addWidget(self.moveCameraDownMm, 1, 1)
        self.grid.addWidget(self.moveCameraUpCm, 2, 0)
        self.grid.addWidget(self.moveCameraDownCm, 2, 1)
        #self.grid.addWidget(self.takePhotoButton, 2, 0, 1, 2)
        self.grid.addWidget(self.takeStackedPhotoButton, 3, 0, 1, 2)
        self.grid.addWidget(self.imgDesc, 0, 2)
        self.grid.addWidget(self.imgView, 1, 2, 7, 1)

        self.setLayout(self.grid)


    def readQRCode(self, imgPath):
        _format = imgPath.split('.')[-1]
        if _format == 'arw':
            with rawpy.imread(imgPath) as raw:
                img = raw.postprocess()
        elif _format == 'jpg':
            img = cv2.imread(imgPath)
        else:
            self.warn('Image␣format␣at␣%s␣not␣understood.␣Got␣%s,␣
                ↪ should␣be␣arw␣or␣jpg.'%(imgPath,_format))

        decoded_list = pyzbar.decode(cv2.resize(img,(1024,680)))
        for decoded in decoded_list:
            if decoded.type == 'QRCODE':
                return decoded.data
        else:
            return ''

    def takePhoto(self, imgName):
        os.chdir(DUMP_FOLDER)

        self.commandLine(['gphoto2', '--capture-image-and-download',
                         '--force-overwrite', '--filename', imgName])

    def takeSinglePhoto(self):
        progress = progressDialog('Taking␣Single␣Photo')
        progress._open()
        timestamp = time.strftime('%Y%m%d_%H%M%S', time.gmtime())
        imgName = 'singlePhoto.arw'

        progress.update(20, 'Taking␣Single␣Photo..')
        self.takePhoto(imgName)

        dumpPath = os.path.join(DUMP_FOLDER, imgName)
        QRCode = self.readQRCode(self.previewPath)
        QRCode = self.checkQRCode(QRCode)

        underside = self.toggleAndCheckUnderside()

        if len(QRCode):
            newImgName = 'NHMD-' + QRCode + underside + '_' +
                ↪ timestamp + '.arw'
            progress.update(90, 'Copying␣file␣to␣cache␣as␣%s␣'%
                ↪ newImgName)
            cachePath = os.path.join(CACHE_FOLDER, newImgName)

            self.commandLine(['cp',dumpPath,cachePath])
            self.warn('Done␣Taking␣Photo')
        progress._close()
```

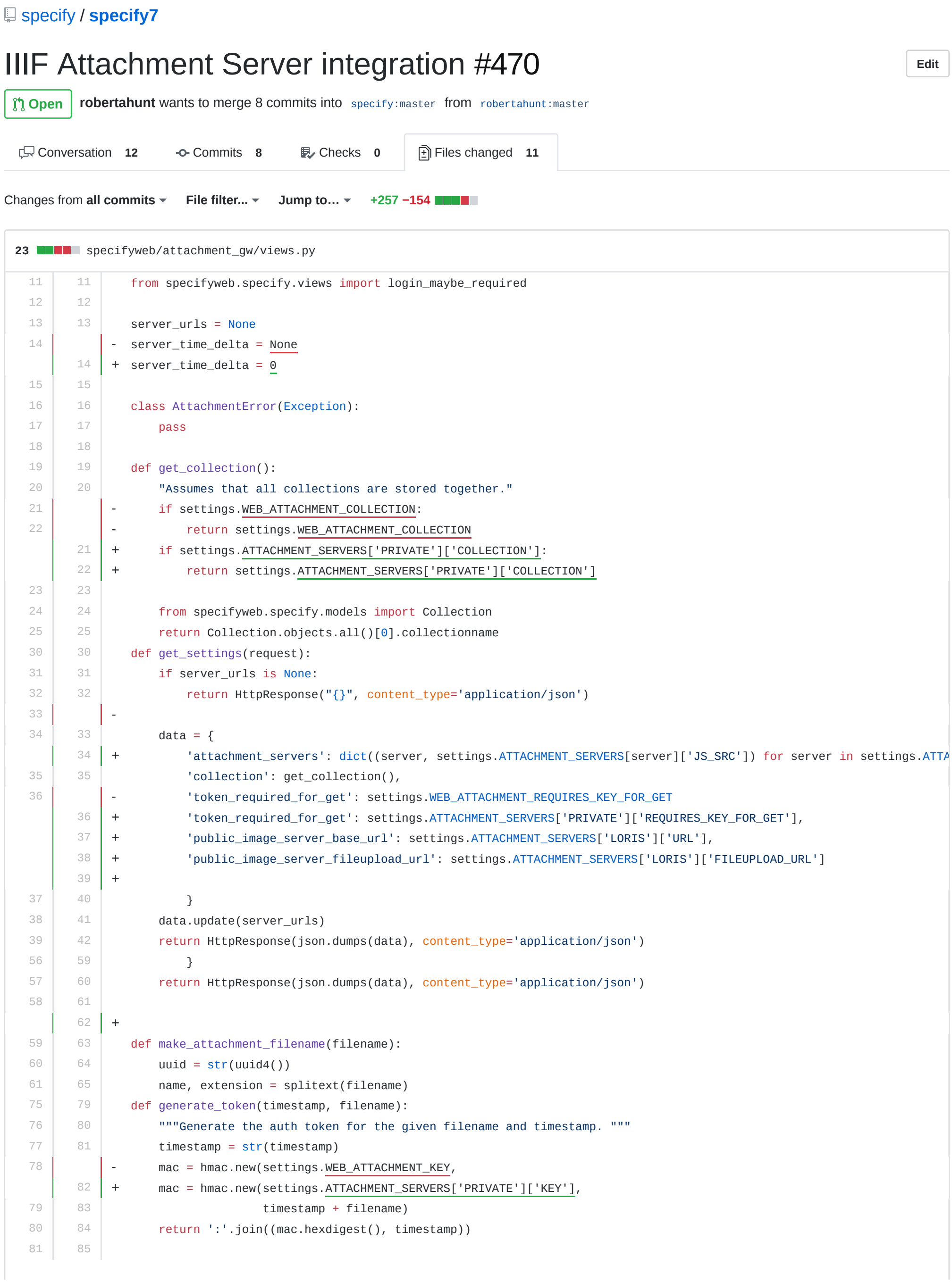
```
def checkQRCode(self, QRCode):
    try:
        _len = len(str(int(QRCode)))
    except:
        _len = 0

    if _len == 6:
        return str(QRCode)

    else:
        QRCode, okPressed = QtWidgets.QInputDialog.getInt(self, "
            ↪ QR␣Code␣not␣found","QR␣Code␣not␣found␣in␣image,␣
            ↪ please␣manually␣input␣6-digit␣Catalog␣Number:")
        if okPressed:
            try:
                _len = len(str(int(QRCode)))
                if _len == 6:
                    return str(QRCode)
            except:
                pass
            return self.checkQRCode(QRCode)
        else:
            self.warn('No␣Photo␣Taken')
            return ''

def toggleAndCheckUnderside(self):
    if self.autoUndersideCheckBox.isChecked():
        if self.undersideCheckBox.isChecked():
            self.undersideCheckBox.setChecked(False)
        else:
            self.undersideCheckBox.setChecked(True)

    if self.undersideCheckBox.isChecked():
        underside = '_V'
    else:
        underside = '_D'

    return underside

def copyToLocalStorage(self, currentPath, newPath):
    self.commandLine(['cp', currentPath, newPath])
```

```
def takeStackedPhotos(self):
    n_photos = 6
    progress = progressDialog(’Taking␣%s␣Stacked␣Photos’%n_photos
        ↪ )
    progress._open()


    progress.update(5,’Checking␣QR␣Code..’)
    QRCode = self.readQRCode(self.previewPath)
    QRCode = self.checkQRCode(QRCode)

    underside = self.toggleAndCheckUnderside()

    if len(QRCode):
        timestamp = time.strftime(’%Y%m%d_%H%M%S’, time.gmtime())

        for i in range(0,n_photos):
            progress.update(100*i/n_photos, ’Taking␣Photo␣%s␣of␣%s
                ↪ ’%(i+1,n_photos))
            tempName = ’Stacked_’+str(i)+’.arw’
            self.takePhoto(imgName=tempName)
            time.sleep(0.1)
            self.arduino.moveCamera(’d’,’0.2’)
            time.sleep(1)

            newImgName = ’NHMD-’ + QRCode + underside + ’_’ +
                ↪ timestamp + ’_Stacked_’ + str(i) + ’.arw’
            progress.update(100*i/n_photos + 5)

            dumpPath = os.path.join(DUMP_FOLDER, tempName)
            cachePath = os.path.join(CACHE_FOLDER, newImgName)
            storagePath = os.path.join(STORAGE_FOLDER, newImgName)

            self.copyToLocalStorage(dumpPath, cachePath)
            self.copyToLocalStorage(dumpPath, storagePath)

            if i == 0:
                self.openIMG(dumpPath)
            if i == 5:
                self.openIMG(dumpPath)
         progress.update(99, 'Moving␣Camera␣Back␣Into␣Place')

         self.arduino.moveCamera('u',str(n_photos*0.2))
         self.warn('Done␣Taking␣Photos')

     progress._close()
     self.displayLatestImg()

 def displayLatestImg(self):
     path, name = self.getLatestImageName(STORAGE_FOLDER)
     self.imgDesc.setText('Latest␣image␣in␣folder:␣%s'% path)
     img = self.getIMG(path)

     imgResized = QtGui.QImage(img.data, img.shape[1], img.shape
         ↪ [0],
                          img.shape[1]*3, QtGui.QImage.
                              ↪ Format_RGB888)
     imgResized = QtGui.QPixmap.fromImage(imgResized).scaled(self.
         ↪ PREVIEW_WIDTH, self.PREVIEW_HEIGHT,
                                            QtCore.Qt.
                                                ↪ KeepAspectRatio)

     self.imgView.setPixmap(imgResized)



 def getLatestImageName(self, folder):
     images = [image for image in os.listdir(folder) if image.
         ↪ split('.')[-1] in ['arw']]
     if len(images):
         latest_image_path = max([os.path.join(folder, image) for
             ↪ image in images], key=os.path.getctime)
         return latest_image_path, latest_image_path.split('/')
             ↪ [-1]
     else:
         return '', ''

 def getIMG(self, path):
     _format = path.split('.')[-1]
     if _format == 'arw':
         with rawpy.imread(path) as raw:
            return raw.postprocess()
     else:
         self.warn('Image␣format␣in␣folder␣not␣understood.%s'%
             ↪ _format)

 def openIMG(self, path):
     self.commandLine(['open', path])



 def openLatestIMG(self):
     path, name = self.getLatestImageName(STORAGE_FOLDER)
     self.commandLine(['open', path])

 def closeEvent(self, event):
     #reply = QtGui.QMessageBox.question(self, 'Message',
     # "Are you sure to quit?", QtGui.QMessageBox.Yes |
     # QtGui.QMessageBox.No, QtGui.QMessageBox.No)

     #if reply == QtGui.QMessageBox.Yes:
     # event.accept()
     #else:
     # event.ignore()
     self.arduino.close()
```

E

Specify Pull Request

link.



specify / **specify7**

IIIF Attachment Server integration #470

Edit

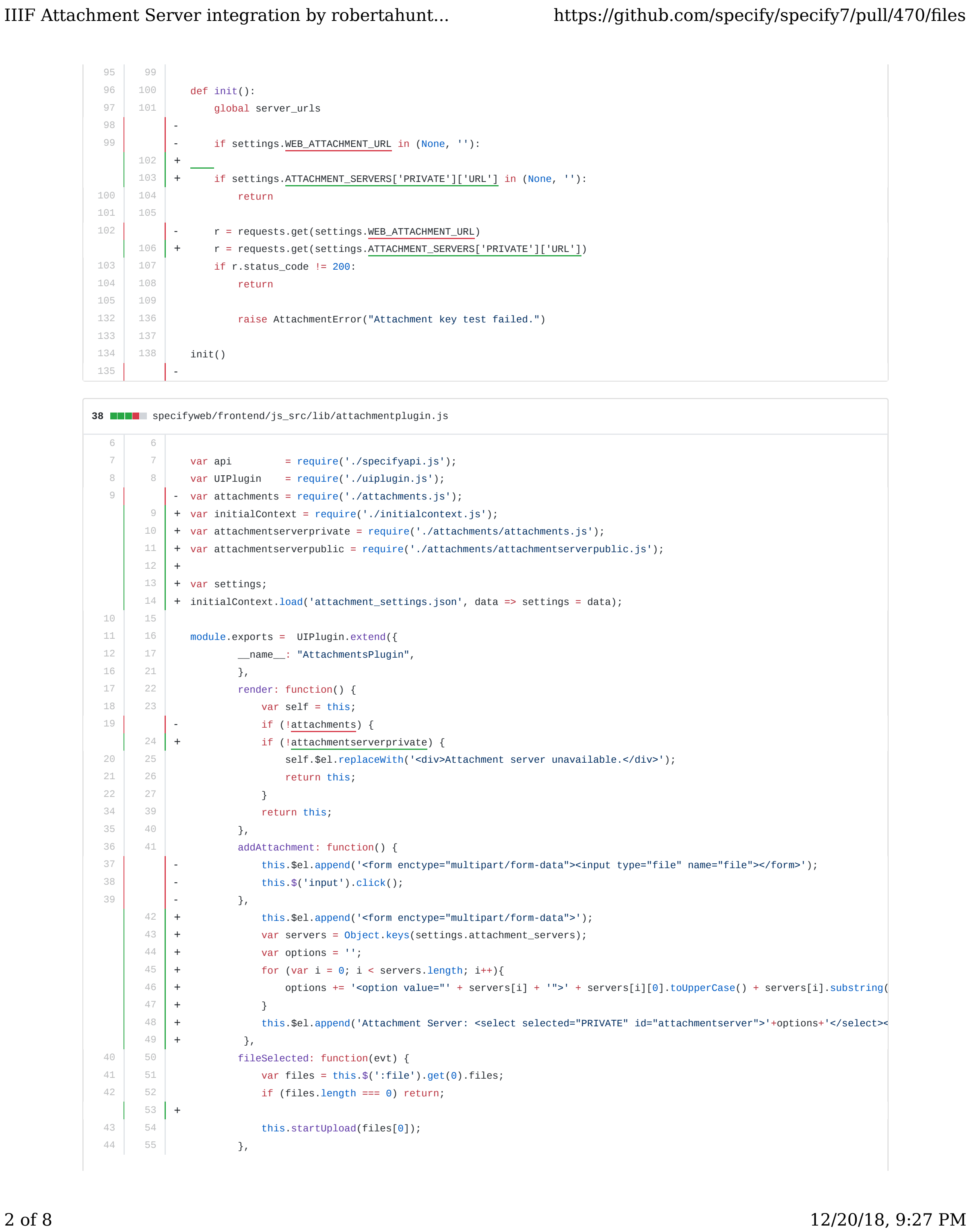
Open **robertahunt** wants to merge 8 commits into `specify:master` from `robertahunt:master`

Conversation 12 | Commits 8 | Checks 0 | Files changed 11

Changes from **all commits** ▾ **File filter...** ▾ **Jump to...** ▾ +257 −154 🟩🟩🟥🟥⬜

23 🟩🟩🟥🟥⬜ `specifyweb/attachment_gw/views.py`

```
  from specifyweb.specify.views import login_maybe_required

  server_urls = None
- server_time_delta = None
+ server_time_delta = 0

  class AttachmentError(Exception):
      pass

  def get_collection():
      "Assumes that all collections are stored together."
-     if settings.WEB_ATTACHMENT_COLLECTION:
-         return settings.WEB_ATTACHMENT_COLLECTION
+     if settings.ATTACHMENT_SERVERS['PRIVATE']['COLLECTION']:
+         return settings.ATTACHMENT_SERVERS['PRIVATE']['COLLECTION']

      from specifyweb.specify.models import Collection
      return Collection.objects.all()[0].collectionname
  def get_settings(request):
      if server_urls is None:
          return HttpResponse("{}", content_type='application/json')
-
      data = {
+         'attachment_servers': dict((server, settings.ATTACHMENT_SERVERS[server]['JS_SRC']) for server in settings.ATTA
          'collection': get_collection(),
-         'token_required_for_get': settings.WEB_ATTACHMENT_REQUIRES_KEY_FOR_GET
+         'token_required_for_get': settings.ATTACHMENT_SERVERS['PRIVATE']['REQUIRES_KEY_FOR_GET'],
+         'public_image_server_base_url': settings.ATTACHMENT_SERVERS['LORIS']['URL'],
+         'public_image_server_fileupload_url': settings.ATTACHMENT_SERVERS['LORIS']['FILEUPLOAD_URL']
+
          }
      data.update(server_urls)
      return HttpResponse(json.dumps(data), content_type='application/json')
          }
      return HttpResponse(json.dumps(data), content_type='application/json')

+
  def make_attachment_filename(filename):
      uuid = str(uuid4())
      name, extension = splitext(filename)
  def generate_token(timestamp, filename):
      """Generate the auth token for the given filename and timestamp. """
      timestamp = str(timestamp)
-     mac = hmac.new(settings.WEB_ATTACHMENT_KEY,
+     mac = hmac.new(settings.ATTACHMENT_SERVERS['PRIVATE']['KEY'],
                     timestamp + filename)
      return ':'.join((mac.hexdigest(), timestamp))

```



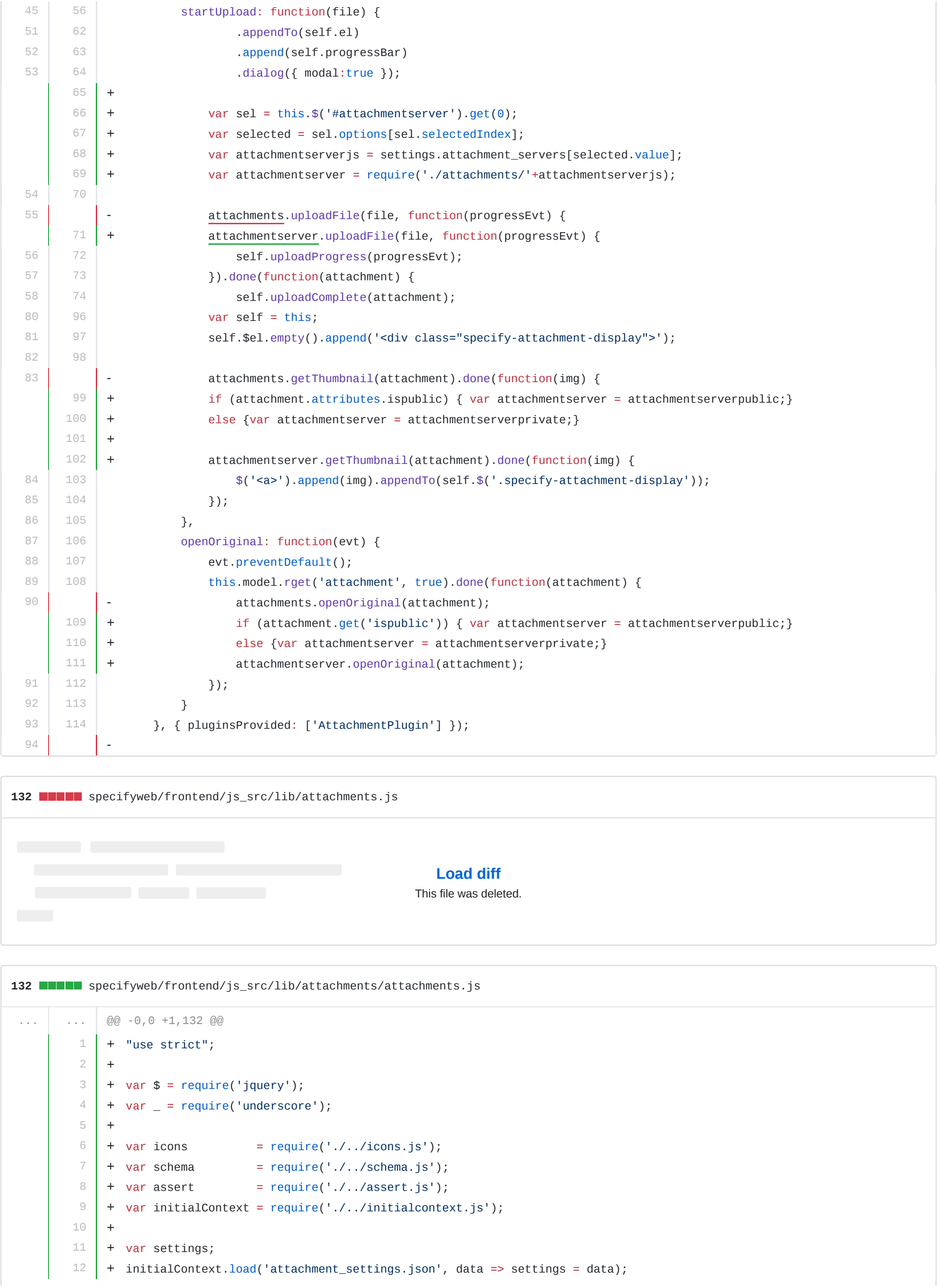
IIIF Attachment Server integration by robertahunt... https://github.com/specify/specify7/pull/470/files

```
     def init():
         global server_urls
-
-        if settings.WEB_ATTACHMENT_URL in (None, ''):
+
+        if settings.ATTACHMENT_SERVERS['PRIVATE']['URL'] in (None, ''):
             return

-        r = requests.get(settings.WEB_ATTACHMENT_URL)
+        r = requests.get(settings.ATTACHMENT_SERVERS['PRIVATE']['URL'])
         if r.status_code != 200:
             return

             raise AttachmentError("Attachment key test failed.")

     init()
-
```

38 🟩🟩🟩🟥⬜ specifyweb/frontend/js_src/lib/attachmentplugin.js

```

   var api         = require('./specifyapi.js');
   var UIPlugin    = require('./uiplugin.js');
-  var attachments = require('./attachments.js');
+  var initialContext = require('./initialcontext.js');
+  var attachmentserverprivate = require('./attachments/attachments.js');
+  var attachmentserverpublic = require('./attachments/attachmentserverpublic.js');
+
+  var settings;
+  initialContext.load('attachment_settings.json', data => settings = data);

   module.exports =  UIPlugin.extend({
           __name__: "AttachmentsPlugin",
           },
           render: function() {
               var self = this;
-              if (!attachments) {
+              if (!attachmentserverprivate) {
                   self.$el.replaceWith('<div>Attachment server unavailable.</div>');
                   return this;
               }
               return this;
           },
           addAttachment: function() {
-              this.$el.append('<form enctype="multipart/form-data"><input type="file" name="file"></form>');
-              this.$('input').click();
-          },
+              this.$el.append('<form enctype="multipart/form-data">');
+              var servers = Object.keys(settings.attachment_servers);
+              var options = '';
+              for (var i = 0; i < servers.length; i++){
+                  options += '<option value="' + servers[i] + '">' + servers[i][0].toUpperCase() + servers[i].substring(
+              }
+              this.$el.append('Attachment Server: <select selected="PRIVATE" id="attachmentserver">'+options+'</select><
+           },
           fileSelected: function(evt) {
               var files = this.$(':file').get(0).files;
               if (files.length === 0) return;
+
               this.startUpload(files[0]);
           },
```

2 of 8 12/20/18, 9:27 PM



```
          startUpload: function(file) {
                  .appendTo(self.el)
                  .append(self.progressBar)
                  .dialog({ modal:true });
+
+             var sel = this.$('#attachmentserver').get(0);
+             var selected = sel.options[sel.selectedIndex];
+             var attachmentserverjs = settings.attachment_servers[selected.value];
+             var attachmentserver = require('./attachments/'+attachmentserverjs);

-             attachments.uploadFile(file, function(progressEvt) {
+             attachmentserver.uploadFile(file, function(progressEvt) {
                  self.uploadProgress(progressEvt);
              }).done(function(attachment) {
                  self.uploadComplete(attachment);
              var self = this;
              self.$el.empty().append('<div class="specify-attachment-display">');

-             attachments.getThumbnail(attachment).done(function(img) {
+             if (attachment.attributes.ispublic) { var attachmentserver = attachmentserverpublic;}
+             else {var attachmentserver = attachmentserverprivate;}
+
+             attachmentserver.getThumbnail(attachment).done(function(img) {
                  $('<a>').append(img).appendTo(self.$('.specify-attachment-display'));
              });
          },
          openOriginal: function(evt) {
              evt.preventDefault();
              this.model.rget('attachment', true).done(function(attachment) {
-                 attachments.openOriginal(attachment);
+                 if (attachment.get('ispublic')) { var attachmentserver = attachmentserverpublic;}
+                 else {var attachmentserver = attachmentserverprivate;}
+                 attachmentserver.openOriginal(attachment);
              });
          }
      }, { pluginsProvided: ['AttachmentPlugin'] });
-
```

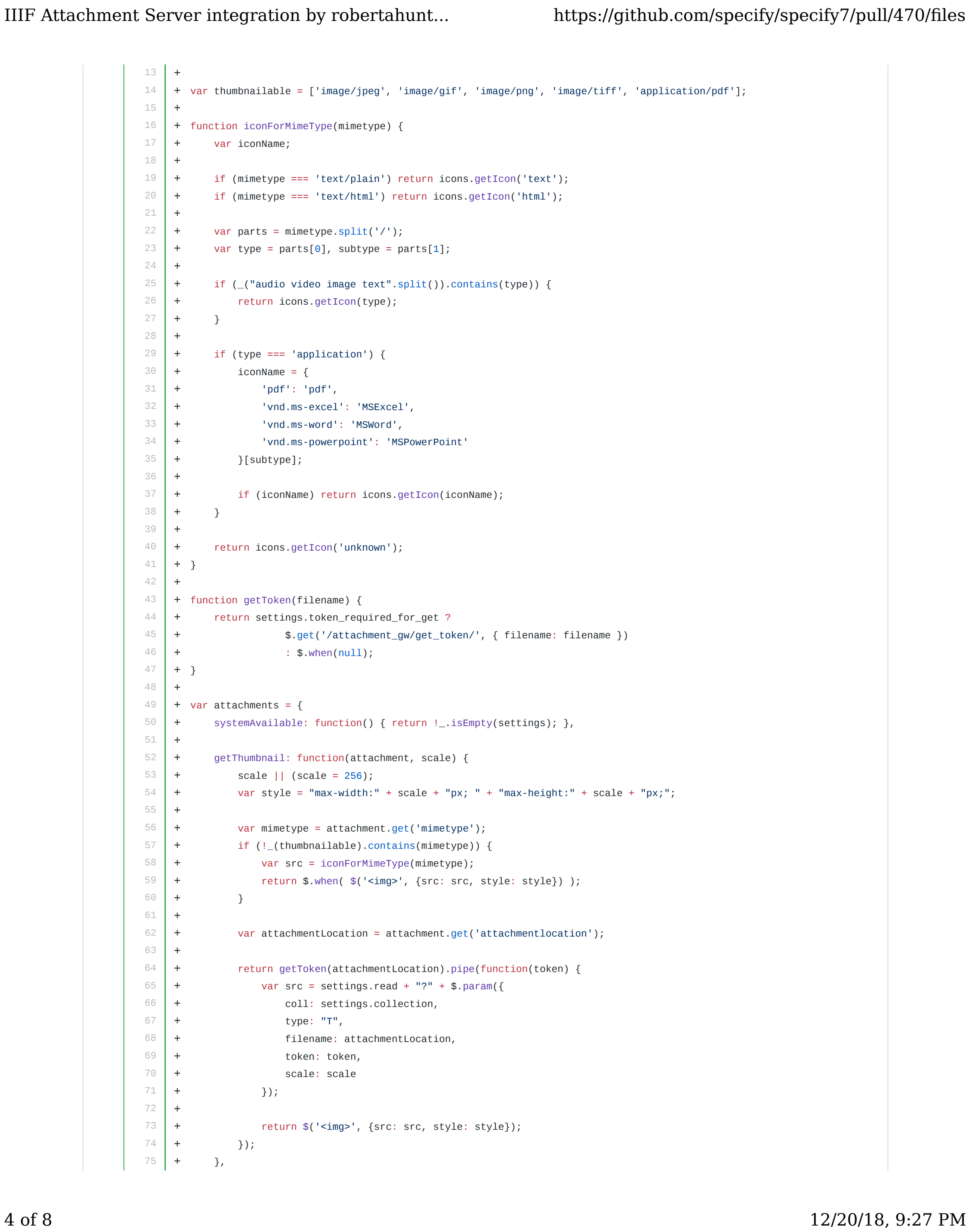
132 🟥🟥🟥🟥🟥 specifyweb/frontend/js_src/lib/attachments.js

Load diff

This file was deleted.

132 🟩🟩🟩🟩🟩 specifyweb/frontend/js_src/lib/attachments/attachments.js

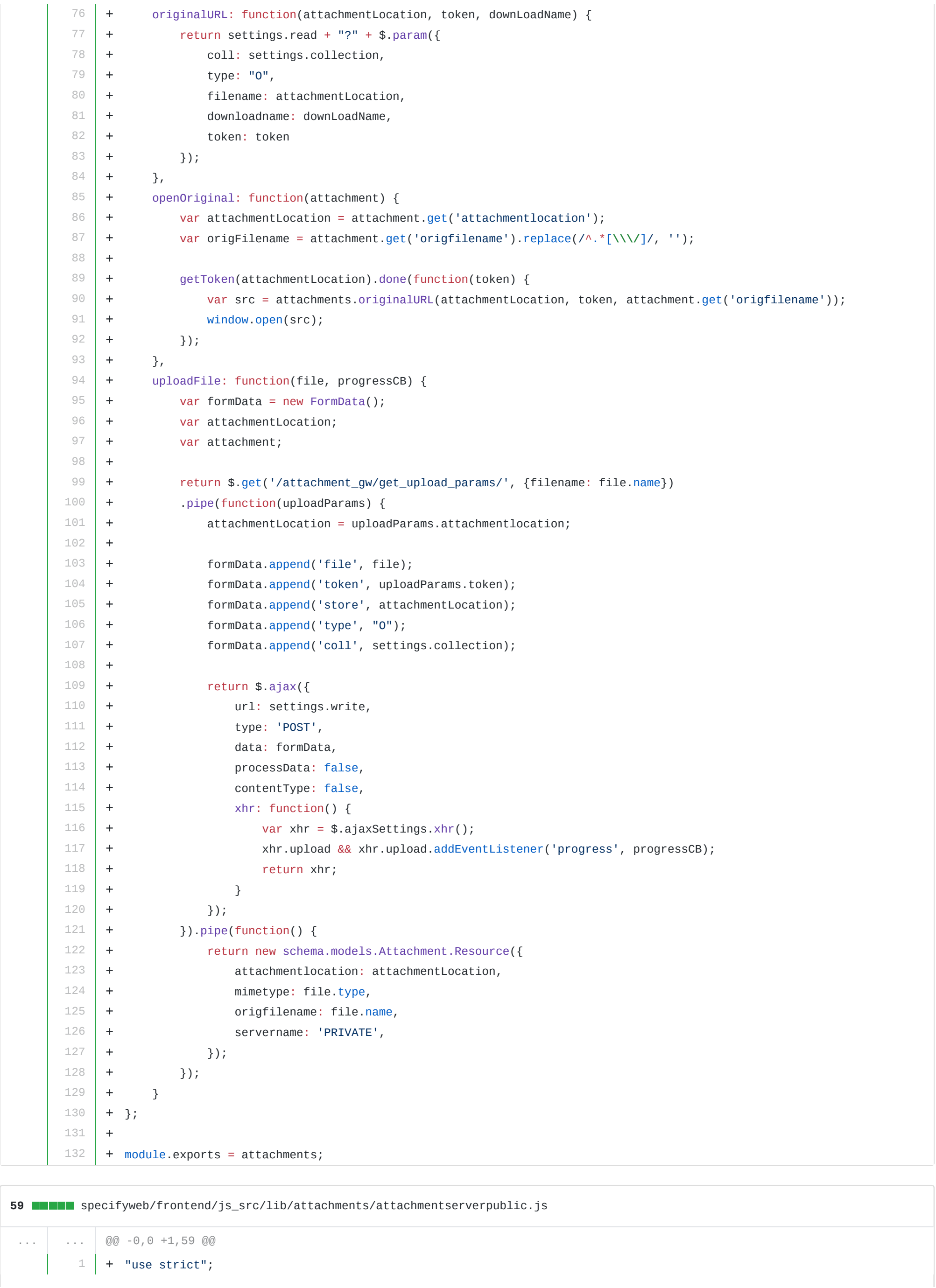
```
@@ -0,0 +1,132 @@
+ "use strict";
+
+ var $ = require('jquery');
+ var _ = require('underscore');
+
+ var icons          = require('./../icons.js');
+ var schema         = require('./../schema.js');
+ var assert         = require('./../assert.js');
+ var initialContext = require('./../initialcontext.js');
+
+ var settings;
+ initialContext.load('attachment_settings.json', data => settings = data);
```



IIIF Attachment Server integration by robertahunt... https://github.com/specify/specify7/pull/470/files

```
+
+ var thumbnailable = ['image/jpeg', 'image/gif', 'image/png', 'image/tiff', 'application/pdf'];
+
+ function iconForMimeType(mimetype) {
+     var iconName;
+
+     if (mimetype === 'text/plain') return icons.getIcon('text');
+     if (mimetype === 'text/html') return icons.getIcon('html');
+
+     var parts = mimetype.split('/');
+     var type = parts[0], subtype = parts[1];
+
+     if (_("audio video image text".split()).contains(type)) {
+         return icons.getIcon(type);
+     }
+
+     if (type === 'application') {
+         iconName = {
+             'pdf': 'pdf',
+             'vnd.ms-excel': 'MSExcel',
+             'vnd.ms-word': 'MSWord',
+             'vnd.ms-powerpoint': 'MSPowerPoint'
+         }[subtype];
+
+         if (iconName) return icons.getIcon(iconName);
+     }
+
+     return icons.getIcon('unknown');
+ }
+
+ function getToken(filename) {
+     return settings.token_required_for_get ?
+                 $.get('/attachment_gw/get_token/', { filename: filename })
+                 : $.when(null);
+ }
+
+ var attachments = {
+     systemAvailable: function() { return !_.isEmpty(settings); },
+
+     getThumbnail: function(attachment, scale) {
+         scale || (scale = 256);
+         var style = "max-width:" + scale + "px; " + "max-height:" + scale + "px;";
+
+         var mimetype = attachment.get('mimetype');
+         if (!_(thumbnailable).contains(mimetype)) {
+             var src = iconForMimeType(mimetype);
+             return $.when( $('<img>', {src: src, style: style}) );
+         }
+
+         var attachmentLocation = attachment.get('attachmentlocation');
+
+         return getToken(attachmentLocation).pipe(function(token) {
+             var src = settings.read + "?" + $.param({
+                 coll: settings.collection,
+                 type: "T",
+                 filename: attachmentLocation,
+                 token: token,
+                 scale: scale
+             });
+
+             return $('<img>', {src: src, style: style});
+         });
+     },
```

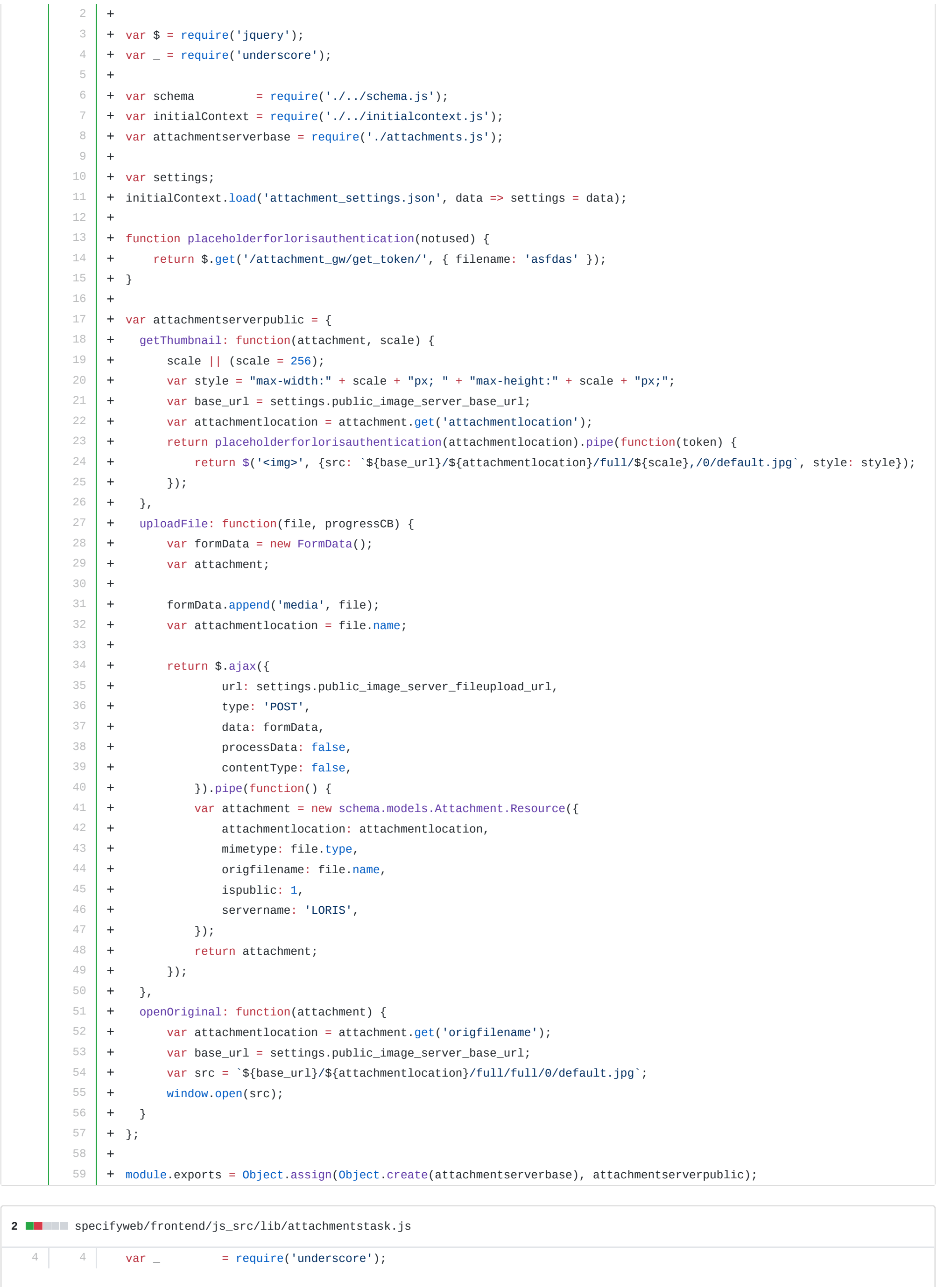
4 of 8 12/20/18, 9:27 PM



```
+     originalURL: function(attachmentLocation, token, downLoadName) {
+         return settings.read + "?" + $.param({
+             coll: settings.collection,
+             type: "O",
+             filename: attachmentLocation,
+             downloadname: downLoadName,
+             token: token
+         });
+     },
+     openOriginal: function(attachment) {
+         var attachmentLocation = attachment.get('attachmentlocation');
+         var origFilename = attachment.get('origfilename').replace(/^.*[\\\/]/, '');
+
+         getToken(attachmentLocation).done(function(token) {
+             var src = attachments.originalURL(attachmentLocation, token, attachment.get('origfilename'));
+             window.open(src);
+         });
+     },
+     uploadFile: function(file, progressCB) {
+         var formData = new FormData();
+         var attachmentLocation;
+         var attachment;
+
+         return $.get('/attachment_gw/get_upload_params/', {filename: file.name})
+         .pipe(function(uploadParams) {
+             attachmentLocation = uploadParams.attachmentlocation;
+
+             formData.append('file', file);
+             formData.append('token', uploadParams.token);
+             formData.append('store', attachmentLocation);
+             formData.append('type', "O");
+             formData.append('coll', settings.collection);
+
+             return $.ajax({
+                 url: settings.write,
+                 type: 'POST',
+                 data: formData,
+                 processData: false,
+                 contentType: false,
+                 xhr: function() {
+                     var xhr = $.ajaxSettings.xhr();
+                     xhr.upload && xhr.upload.addEventListener('progress', progressCB);
+                     return xhr;
+                 }
+             });
+         }).pipe(function() {
+             return new schema.models.Attachment.Resource({
+                 attachmentlocation: attachmentLocation,
+                 mimetype: file.type,
+                 origfilename: file.name,
+                 servername: 'PRIVATE',
+             });
+         });
+     }
+ };
+
+ module.exports = attachments;
```

59 ■■■■■ specifyweb/frontend/js_src/lib/attachments/attachmentserverpublic.js

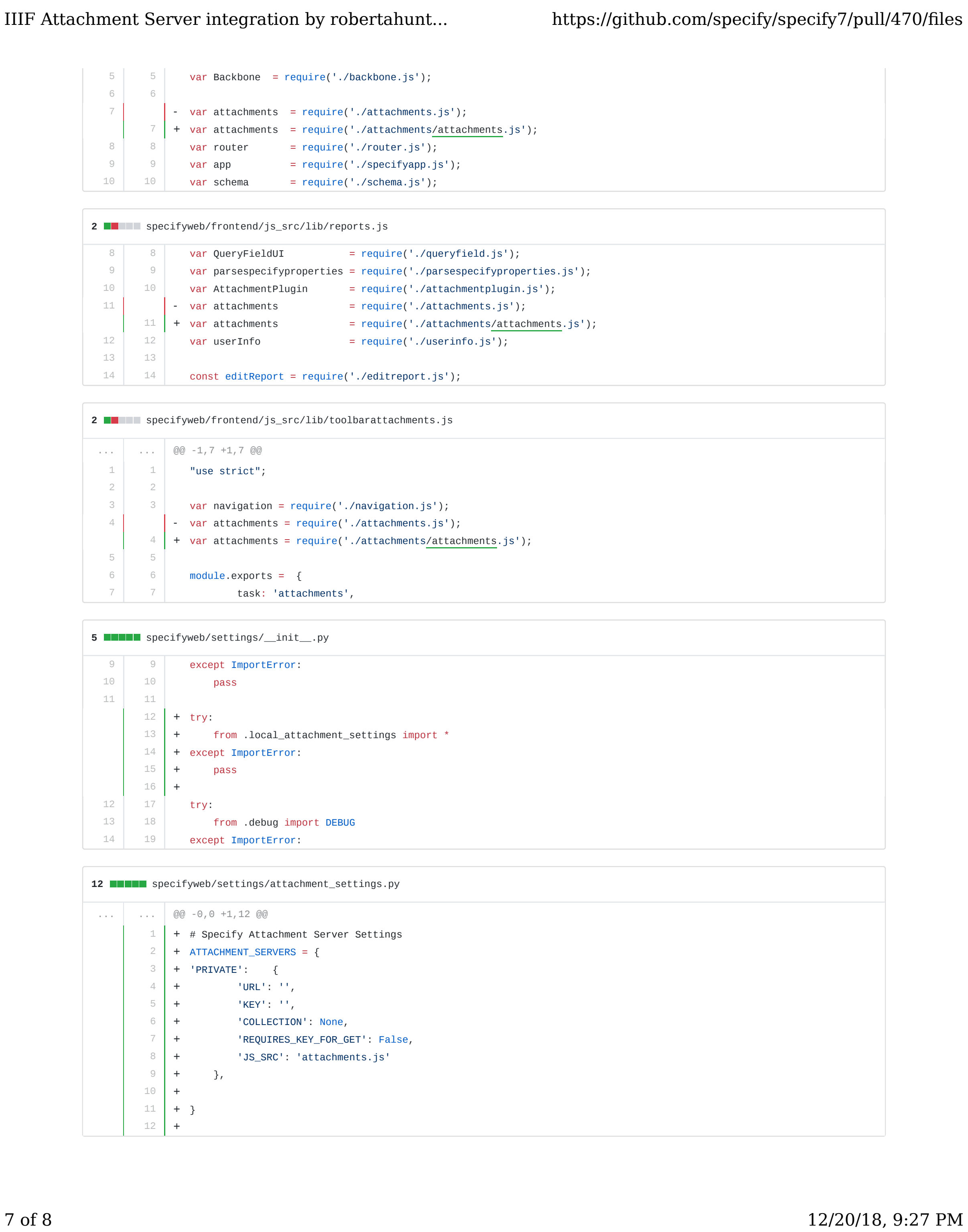
```
... ...  @@ -0,0 +1,59 @@
+ "use strict";
```



```
+
+ var $ = require('jquery');
+ var _ = require('underscore');
+
+ var schema         = require('./../schema.js');
+ var initialContext = require('./../initialcontext.js');
+ var attachmentserverbase = require('./attachments.js');
+
+ var settings;
+ initialContext.load('attachment_settings.json', data => settings = data);
+
+ function placeholderforlorisauthentication(notused) {
+     return $.get('/attachment_gw/get_token/', { filename: 'asfdas' });
+ }
+
+ var attachmentserverpublic = {
+   getThumbnail: function(attachment, scale) {
+       scale || (scale = 256);
+       var style = "max-width:" + scale + "px; " + "max-height:" + scale + "px;";
+       var base_url = settings.public_image_server_base_url;
+       var attachmentlocation = attachment.get('attachmentlocation');
+       return placeholderforlorisauthentication(attachmentlocation).pipe(function(token) {
+           return $('<img>', {src: `${base_url}/${attachmentlocation}/full/${scale},/0/default.jpg`, style: style});
+       });
+   },
+   uploadFile: function(file, progressCB) {
+       var formData = new FormData();
+       var attachment;
+
+       formData.append('media', file);
+       var attachmentlocation = file.name;
+
+       return $.ajax({
+               url: settings.public_image_server_fileupload_url,
+               type: 'POST',
+               data: formData,
+               processData: false,
+               contentType: false,
+           }).pipe(function() {
+           var attachment = new schema.models.Attachment.Resource({
+               attachmentlocation: attachmentlocation,
+               mimetype: file.type,
+               origfilename: file.name,
+               ispublic: 1,
+               servername: 'LORIS',
+           });
+           return attachment;
+       });
+   },
+   openOriginal: function(attachment) {
+       var attachmentlocation = attachment.get('origfilename');
+       var base_url = settings.public_image_server_base_url;
+       var src = `${base_url}/${attachmentlocation}/full/full/0/default.jpg`;
+       window.open(src);
+   }
+ };
+
+ module.exports = Object.assign(Object.create(attachmentserverbase), attachmentserverpublic);
```

2 🟩🟥⬜⬜⬜ specifyweb/frontend/js_src/lib/attachmentstask.js

```
  var _          = require('underscore');
```



```
  var Backbone  = require('./backbone.js');

- var attachments  = require('./attachments.js');
+ var attachments  = require('./attachments/attachments.js');
  var router       = require('./router.js');
  var app          = require('./specifyapp.js');
  var schema       = require('./schema.js');
```

2 🟩🟥⬜⬜⬜ specifyweb/frontend/js_src/lib/reports.js

```
  var QueryFieldUI           = require('./queryfield.js');
  var parsespecifyproperties = require('./parsespecifyproperties.js');
  var AttachmentPlugin       = require('./attachmentplugin.js');
- var attachments            = require('./attachments.js');
+ var attachments            = require('./attachments/attachments.js');
  var userInfo               = require('./userinfo.js');

  const editReport = require('./editreport.js');
```

2 🟩🟥⬜⬜⬜ specifyweb/frontend/js_src/lib/toolbarattachments.js

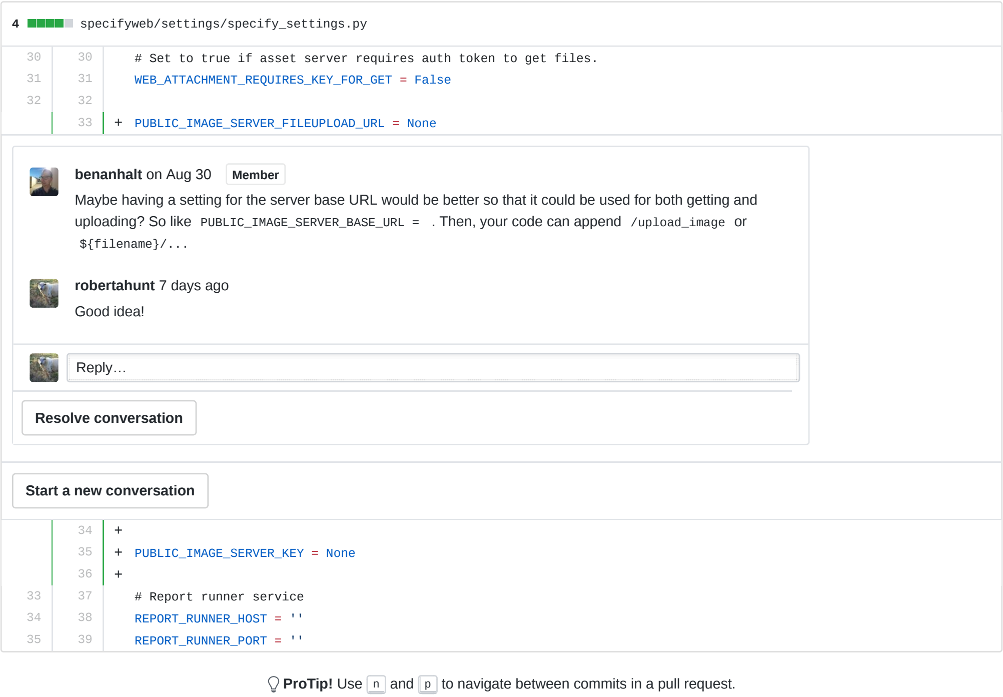
```
@@ -1,7 +1,7 @@
  "use strict";

  var navigation = require('./navigation.js');
- var attachments = require('./attachments.js');
+ var attachments = require('./attachments/attachments.js');

  module.exports =  {
          task: 'attachments',
```

5 🟩🟩🟩🟩🟩 specifyweb/settings/__init__.py

```
  except ImportError:
      pass

+ try:
+     from .local_attachment_settings import *
+ except ImportError:
+     pass
+
  try:
      from .debug import DEBUG
  except ImportError:
```

12 🟩🟩🟩🟩🟩 specifyweb/settings/attachment_settings.py

```
@@ -0,0 +1,12 @@
+ # Specify Attachment Server Settings
+ ATTACHMENT_SERVERS = {
+ 'PRIVATE':    {
+         'URL': '',
+         'KEY': '',
+         'COLLECTION': None,
+         'REQUIRES_KEY_FOR_GET': False,
+         'JS_SRC': 'attachments.js'
+     },
+
+ }
+
```



4 specifyweb/settings/specify_settings.py

```
    # Set to true if asset server requires auth token to get files.
    WEB_ATTACHMENT_REQUIRES_KEY_FOR_GET = False

+   PUBLIC_IMAGE_SERVER_FILEUPLOAD_URL = None
```

benanhalt on Aug 30 Member

Maybe having a setting for the server base URL would be better so that it could be used for both getting and uploading? So like `PUBLIC_IMAGE_SERVER_BASE_URL =` . Then, your code can append `/upload_image` or `${filename}/...`

robertahunt 7 days ago

Good idea!

```
+
+   PUBLIC_IMAGE_SERVER_KEY = None
+
    # Report runner service
    REPORT_RUNNER_HOST = ''
    REPORT_RUNNER_PORT = ''
```

ProTip! Use n and p to navigate between commits in a pull request.