The OpenFlexure Microscope

Software Requirements Specification

Mehmet Alper Yilmaz - 2405611

Alparslan Yeşilkaya - 2237923

Spring 2021

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1 Introduction

This documentation is the SRS (Software Requirements Specification) of The OpenFlexure Microscope which is a modern microscope supported with “Web of Things” (WoT).

1.1 Purpose

In this project, an open source microscope will be developing a lab microscope which is open source, 3D printed, fully automatic motorized sample positioning and auto focused. Users will be able to access the microscope not only physically but also anywhere via the Internet, with a low cost and extensibility features.

1.2 Scope

In this project, users will be able to use 3D printed microscopes thanks to Web of Things (WoT) over the internet, utilizing many client side environments with the support of open source software. Scientists will be able to extend and modify its source code due to its open source nature. More specific objectives include followings.

- Device will include a raspberry pi computer containing device control code for the operation of the camera, leds, motor movements, cabled connections, networking, image processing to enable client and server side applications functioning.

- The system will have GUI and APIs for scripting experiments which provide simultaneous connection of users to microscope, real-time camera feed, sample manipulation, data acquisition, data analysis.

- The system will enable a single client to access multiple microscopes. As a result; multiple image samples obliquely, increment in the throughput will be obtained.

- The system will be built as a server client architecture. As a result, client side applications can use any modern programming language.

- The system will provide a web application, a desktop application, a Python client, and a Matlab client.

- The system will be able to live-stream real time Motion JPEG (MJPEG) with auto focus features, data compression,image correction, and post- processing algorithms.

- The system will also provide automatic calibration features.

1.3 System Overview

1.3.1 System Perspective

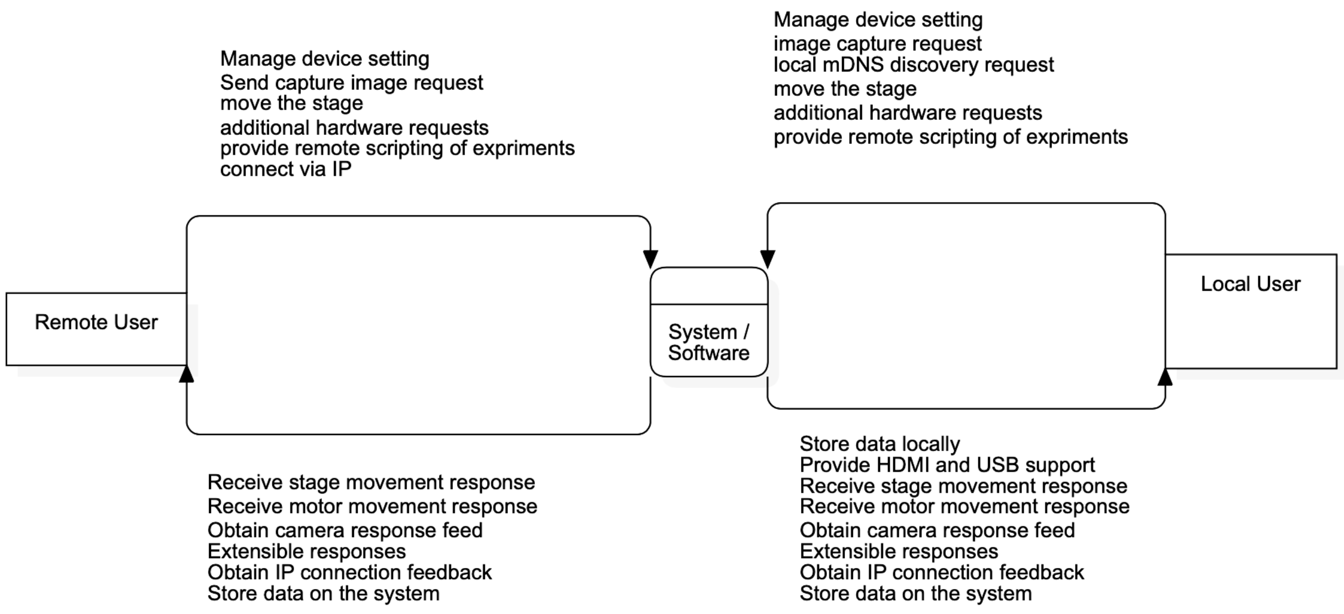

Manage device setting
Send capture image request
move the stage
additional hardware requests
provide remote scripting of expriments
connect via IP
Manage device setting
image capture request
local mDNS discovery request
move the stage
additional hardware requests
provide remote scripting of expriments
Remote User
System / Software
Local User
Receive stage movement response
Receive motor movement response
Obtain camera response feed
Extensible responses
Obtain IP connection feedback
Store data on the system
Store data locally
Provide HDMI and USB support
Receive stage movement response
Receive motor movement response
Obtain camera response feed
Extensible responses
Obtain IP connection feedback
Store data on the system

Figure 1: Context Diagram

1.3.1.1 System Interfaces

Data Storage Interface: This interface stores the data locally either on an external hard disk or SDD. If the stored data is needed it is fetched from external storage devices by accessing this interface. Both remote and local users can use this interface to access locally connected data storage units.

Error Management Interface: This interface runs as a subroutine on the device checking system failures constantly. It includes error handling methods for either hardware or software operations. Clients can access this interface to examine the system in a fast fashion. It detects system failures and abnormal activities.

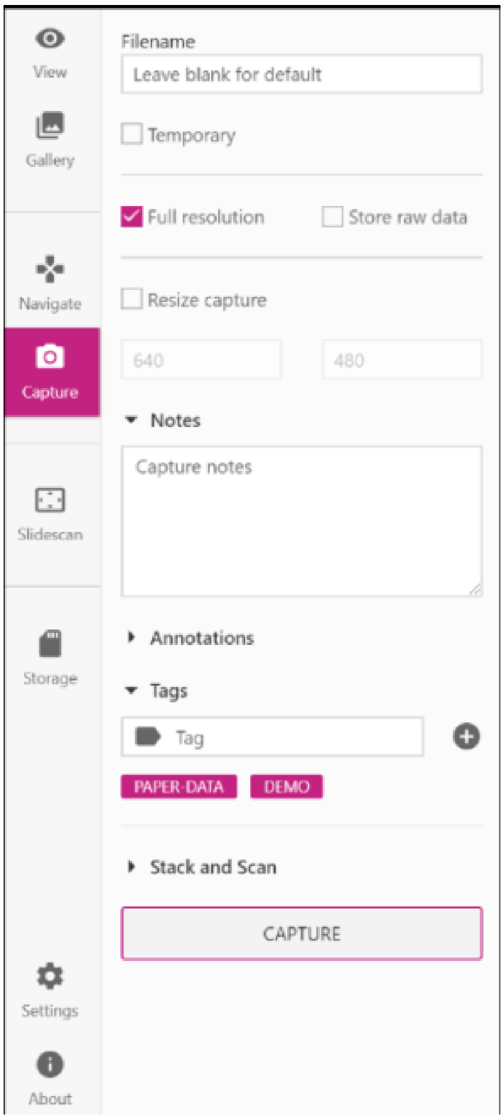
Server API: This interface contains the core logic of the system. This interface manages all the other interfaces like a maestro. It provides functionalities to the client interfaces, uses system interfaces and hardware interfaces. Lastly it communicates with other interfaces by using communication interfaces. The server API is written by using python Flask. Thread based concurrency and hardware synchronization are also the functionalities it provides.

1.3.1.2 User Interfaces

Web Application Interface: This interface provides GUI to the users. Thanks to its being a web application it can be accessed by different client systems. It also includes live stream of camera, image capture, data management, and extension support functionalities.

Users can access the OpenFlexure graphical interface from any web browser which is either connected to the same local network or the internet. If the microscoscope is in the local network, the user can access it by typing “http://microscope.local:5000” to the web browser. If the OFM in the external network, the user can access it by typing “http:/microscopesIPadress”, for example “http://144.122.171.83:5000”. The interface of the web application runs on port 5000.

Desk Application Interface: OpenFlexure Connect desktop application provides this interface with GUI support. It provides mDNS discovery and manual connection features. It also saves commonly accessed microscopes in a list. This interface is responsive with general purpose interface frameworks.



Filename: Leave blank for default

[ ] Temporary

[✓] Full resolution [ ] Store raw data

[ ] Resize capture

640 480

Capture notes

Tag

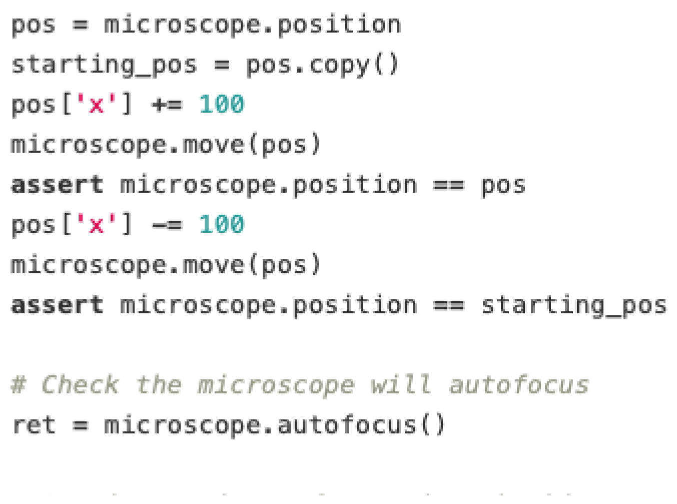
PAPER-DATA DEMO

Figure 2: OpenFlexure Connect Desktop Application Interface

Python Client Interface: This interface enables users with experiment scripting. It achieves its goal by mapping web API functions to python functions. As a result, interactive sessions are possible through scripting. Thanks to this interface the microscope can be controlled from a Python script that can be run on the Raspberry Pi itself or remotely. Jupyter notebook can be used to run the script remotely allowing users to plot graphs and display images.

```python
import openflexure_microscope_client as ofm_client
microscope = ofm_client.find_first_microscope()
```

Figure 3: Python Client import

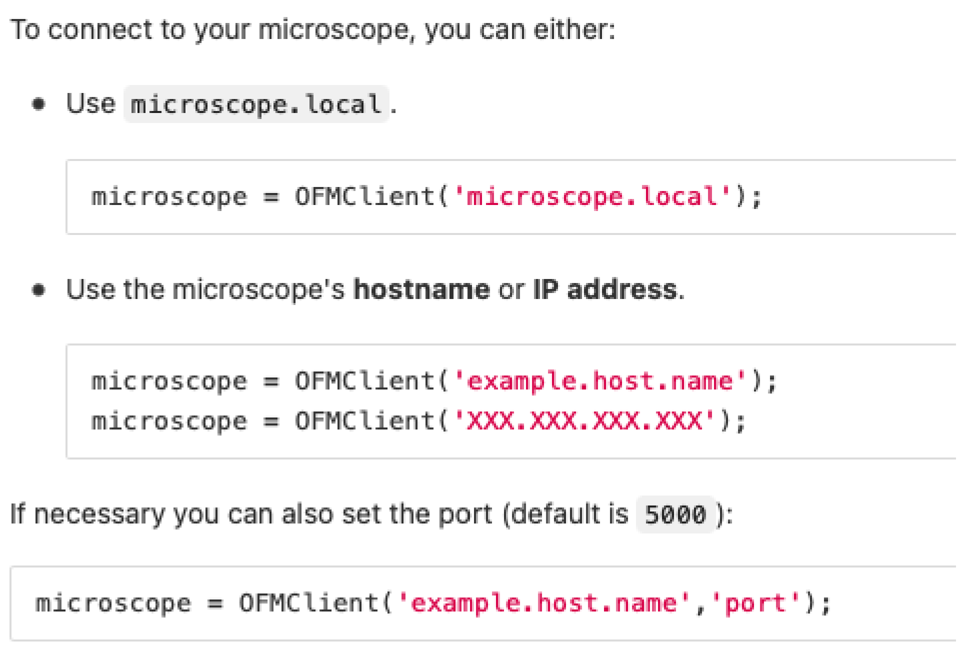


```python
pos = microscope.position
starting_pos = pos.copy()
pos['x'] += 100
microscope.move(pos)
assert microscope.position == pos
pos['x'] -= 100
microscope.move(pos)
assert microscope.position == starting_pos

# Check the microscope will autofocus
ret = microscope.autofocus()
```

Figure 4: Python Client OFM Control example

MATLAB Client Interface: Users can access and control the microscope through MATLAB over a network or locally. Users are able to control the motors and access to the camera feed as well as extend its functionalities.



To connect to your microscope, you can either:

- Use `microscope.local`.

```matlab
microscope = OFMClient('microscope.local');
```

- Use the microscope's **hostname** or **IP address**.

```matlab
microscope = OFMClient('example.host.name');
microscope = OFMClient('XXX.XXX.XXX.XXX');
```

If necessary you can also set the port (default is `5000`):

```matlab
microscope = OFMClient('example.host.name','port');
```

Figure 5: MATLAB Client Connection Establishment

1.3.1.3 Hardware Interfaces

External IO device Interfaces: This interface enables connection of external IO devices to the microscope. Since OFM contains client applications also in its local Raspberry pi computer, local IO device connection and through this connection direct interaction with the client applications is possible.

Motor Control Interface: This interface provides Raspberry pi with the control of motors through an Arduino board.

1.3.1.4 Software Interfaces

Operating System: The operating system is automatically built and distributed by an SD card image. Client and server applications are preinstalled in the OS.

1.3.1.5 Communication Interfaces

Users can establish communication with the microscope in 2 different ways. First, users can use USB peripherals and HDMI output to directly control the microscope. Secondly, users can connect to the server host, which controls the microscope, over any IP network (Internet, Wireless or Ethernet) and hosts can send request/response messages to each other through HTTP application layer protocol.

1.3.1.6 Memory Constraints

The memory usage of the system isn't very high due to its limited operations so memory is not a big problem for the openflexure microscope system. However, depending on the researcher, the storage can be a problem because researchers may want to store large amounts of data from the system. In that case larger storage devices should be preferred in Raspberry Pi.

1.3.1.7 Operations

The operations of the OpenFlexure Microscope software can be divided into 2 group:

Local User Operations:

- Connect to the server by USB and HDMI
- Control Motor
- Control Camera
- Tile Scan
- Stream Video
- Store Image/Video

Remote User Operations

- Connect to the server by IP
- Control Motor
- Control Camera
- Tile Scan
- Stream Video
- Store Image/Video
- Extend Functionality
- Perform Remote Script

All of these operations will be covered more deeply in Functions section (3.2)

1.3.2 System Functions

Function	**Summary**
Connect by HDMI and USB	User can connect to the OpenFlexure Microscope system by using USM and HDMI cables.
Connect by IP	The OpenFlexure Microscope system can be connected via IP network.
Control Motor	Users can control the motors on the OpenFlexure Microscope in order to move the translation stage to the desired position.
Control Camera	Users can make camera perform some functionalities such as auto focus, calibrating images and camera-stage mapping.
Calibrate Image	After an image captured by the camere, it needs to be calibrated. Therefore, the captured image go through some process which are applying flat-field correction,

	correcting vignetting and post process.
Post Process Image	The saturation of captured images are corrected
Correct Vignetting	Vignetting which occurs in raw images is corrected.
Apply Flat-Field Correction	Captured images can suffer from some optical artifacts like shading or lens cast. Flat-Field correction removes these kinds of artifacts.
Camera Stage Mapping	The translation stage is mapped to camera location in accordance with axes and step sizes of the translation stage.
Auto Focus	The quality of captured images and videos is increased
Stream Video	The system provides a real time video stream for the user.
Run MJPEG Algorithm	Each captured video frame compressed by a high quality video compression format called MJPEG.
Tile Scan	Users can create images which are much larger than the camera’s field of view via capturing images at multiple locations.
Extend Functionality	With this functionality, users can add their own functionalities to the system.
Perform Remote Scripting of Experiments	This functionality enables users to perform scripted experiments via converting the web API into Python or MATLAB functions.
Store Data	Users can store images or videos captured from the microscope and they can access these data from the application whenever they want.

Table 1 : System Functions

1.3.3 User Characteristics

There are 2 types of users in the OpenFlexure Microscope System: Local Users and Remote Users.

Local users connect to the system via USB and HDMI cables. These types of users are not expected to have programming skills. Therefore, They perform all fundamental operations on graphical user interfaces.

Remote users connect to the system via IP network. There is no strict obligation about having programming skills. If they don't, they can still use the system with gui and perform basic operations. If they have programming skills, They can extend the functionality of the system by adding new functions or they can conduct experiments by writing proper experimental scripts.

1.3.4 Limitations

Regulatory policies: Since OFM systems don't collect personal data of users or any bank account, there is no strict regulatory policy which the system must obey.

Hardware limitations: Most of the operations which the OFM system may manage to perform depend on the hardware. The general structure of the translation stage and the motors , which move the stage, determines the motion capability. Used cameras determine the image and video quality. The memory size, which the server has, restricts its memory usage amount.

Interfaces to other applications: OFM systems should be compatible with different types of microcontrollers.

Parallel operation: Parallelization is the very significant part of the OFM system because server application should be capable of broadcasting video and performing desired operations concurrently.

Audit and Control functions: OFM system has an inner control mechanism which rejects all operation requests in case the microscope is used by a client.

Higher-order language requirements: For client application Python or MATLAB can be used for experiment scripting. Server application is implemented with Python.

Signal handshake protocols: Remote client application and server application communicate with each other via HTTP protocol.

Quality requirements: Since users may want rapid response of their operations and fast high quality video stream at the same time, both client host and server host should have fast high bandwidth network.

Criticality of application: System failure causes interruption of the experiment only. It doesn't induce serious problems.

Safety and security considerations: It is not required to take security or safety measures because OFM systems can't hurt anyone and it doesn't collect any valuable data which can be stolen. However, if the secure connection is desired, instead of http, https protocol may be preferred.

Physical/mental considerations: Any one having access to the internet, or direct access to the microscope via cable can use the OpenFlexure system. As long as the microscope is ready for use, only making the connection can be problematic. If this is the case, the person should request help from other people.

1.4 Definitions

Term	**Definition**
API	Application Programming Interfaces
GUI	Graphical User Interface
HTTP	Hyper-Text Transfer Protocol
IP	Internet Protocol
LAN	Local Area Network
OFM	OpenFlexure Microscope
TCP	Transmission Control Protocol

Table 2: Definitions

2 References

This document is prepared with respect to the specifications of IEEE 29148-2018 standard:
"ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering," in ISO/IEC/IEEE 29148:2018(E) , vol., no., pp.1-104, 30 Nov. 2018, doi: 10.1109/IEEESTD.2018.8559686. from https://standards.ieee.org/standard/29148-2018.html

Other sources:
Collins, Joel & Knapper, Joe & Stirling, Julian & McDermott, Samuel & Bowman, Richard. (2021). Modern Microscopy with the Web of Things: The OpenFlexure Microscope Software Stack.

Collins, Joel & Knapper, Joe & Stirling, Julian & Mduda, Joram & Mkindi, Catherine & Mayagaya, Valeriana & Anyelwisye, Grace & Nyakyi, Paul & Sanga, Valerian & Carbery, Dave & White, Leah & Dale, Sara & Lim, Zhen Jieh & Baumberg, Jeremy & Cicuta, Pietro & McDermott, Samuel & Vodenicharski, Boyko & Bowman, Richard. (2020). Robotic microscopy for everyone: the OpenFlexure Microscope. Biomedical Optics Express. 11. 10.1364/BOE.385729.

Stirling, Julian & Sanga, Valerian & Nyakyi, Paul & Mwakajinga, Grace & Collins, Joel & Bumke, Kaspar & Knapper, Joe & Meng, Qingxin & McDermott, Samuel & Bowman, Richard. (2020). The OpenFlexure Project. The technical challenges of Co-Developing a microscope in the UK and Tanzania. 1-4. 10.1109/GHTC46280.2020.9342860.

3 Specific Requirements

3.1 External Interfaces

Following diagram explains the relationship between functionalities of the interfaces and their interaction with each other.

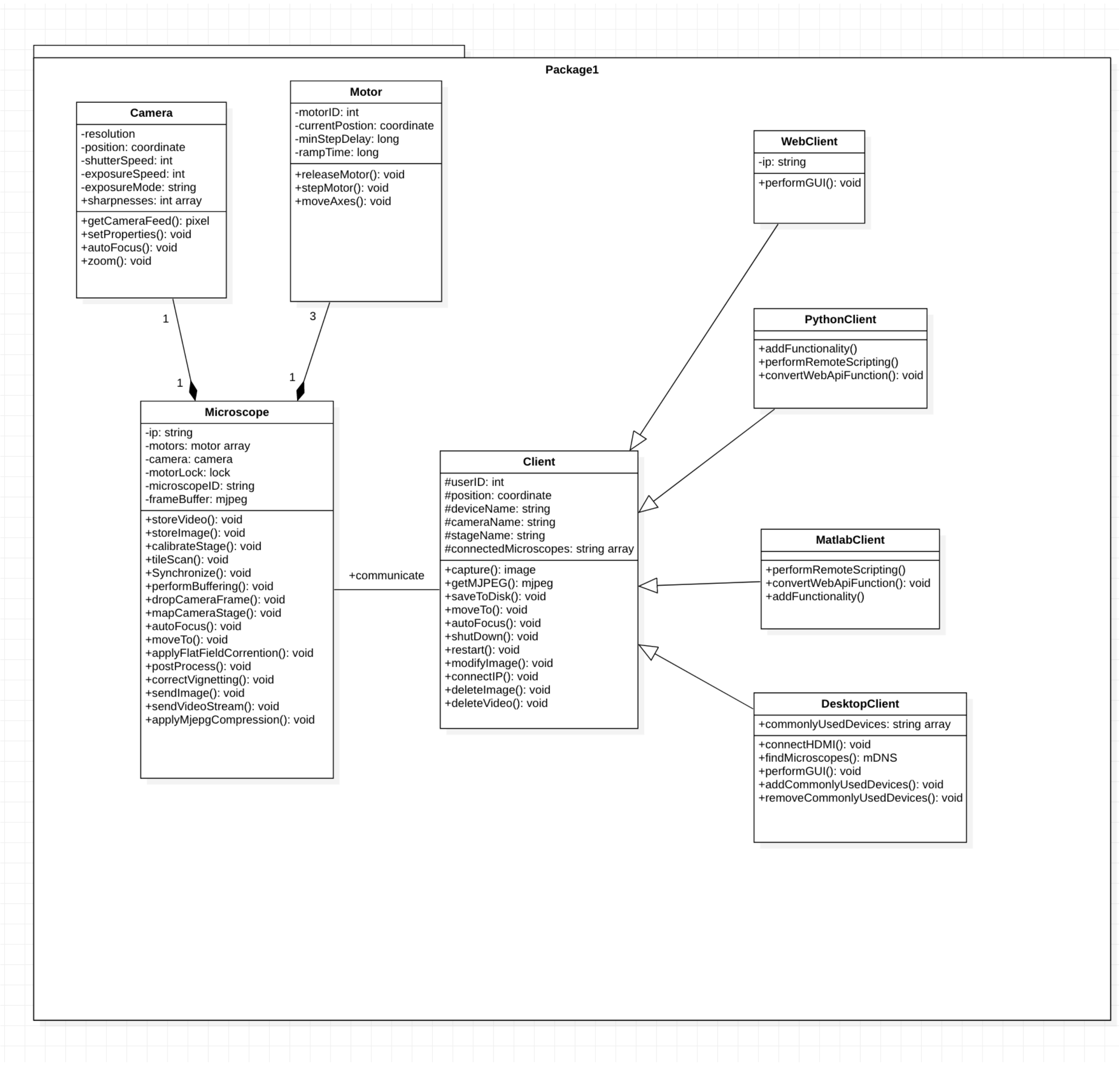


Figure 6: Class Diagram

3.2 Functions

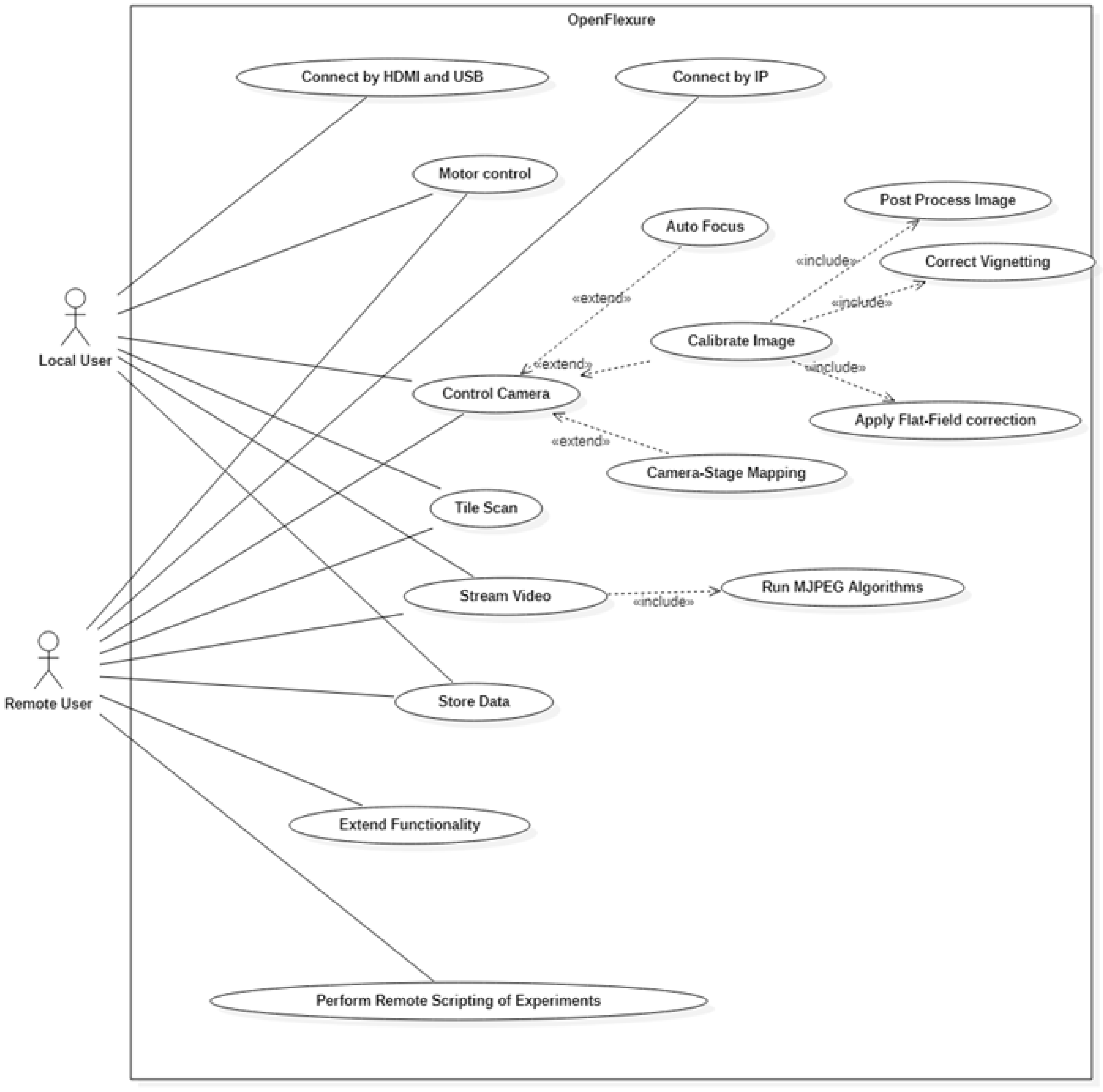


Figure 7: Use-case Diagram

Table 3: Use Case 1

Use Case ID	1
Use Case Name	Connect by HDMI and USB
Actors	Local User
Descriptions	The user connects to the system in order to use its functionalities.
Preconditions	The user has the access to the microscope physically
Postconditions	The user is successfully connected to the OpenFlexure Microscope system.
Normal Flow	1. Microscope is found using mDNS queries in the user’s local network or the user directly connects to the microscope hardware 2. System accepts user connection
Exceptions	1a. If the user’s LAN does not provide mDNS support system will not answer and client side applications will provide error messages. 1a. In case of a hardware failure between the end systems connection fails.

Table 4: Use Case 2

Use Case ID	2
Use Case Name	Connect by IP
Actors	Remote User
Descriptions	The user can connect to the microscope control server application over the IP network.
Preconditions	Server application should be ready for connection.
Postconditions	The user is successfully connected to the server application and now the user can send HTTP request packet to the server and can receive HTTP response packet from the server.
Normal Flow	1. The client sends a small TCP segment to the IP address of the server application. 2. The server application acknowledges back to the client and it means TCP connection is established.
Exceptions	2a. If the connection is lost with the web application, the user can’t control the microscope.

Table 5: Use Case 3

Use Case ID	3
Use Case Name	Control Motors
Actors	Local User and Remote User
Descriptions	The user controls motors on the OFM by interacting with one of the APIs or with a local client.
Preconditions	Connection is established with the OFM by either remotely or locally
Postconditions	Motors are moved as desired.
Normal Flow	1. The user sends movement data by using APIs or the local server. 2. Server side receives the data 3. Server side processes the data. According to this process, the server sends signals to the motors.
Exceptions	3a. If another user that is currently using the OFM already sent a signal to the motors, the signal which is sent later is dropped. User is informed accordingly.

Table 6: Use Case 4

Use Case ID	4
Use Case Name	Control the Camera
Actors	Local User, Remote User
Descriptions	The user controls camera specific actions
Preconditions	The user is connected to the OFM.
Postconditions	Tasks such as auto focus, calibration are performed.
Normal Flow	1. In the graphical interfaces user clicks the related button for update camera settings. In the text based interfaces such as Python scripting API, client calls related functions to control the camera. 2. OFM processes the action internally. 3. If the action is valid camera control actions performed.
Exceptions	1a. Network errors result in error notifications on both server side and client side. 3a. Actions that require reentrant locks result in exceptions sent to the client side.

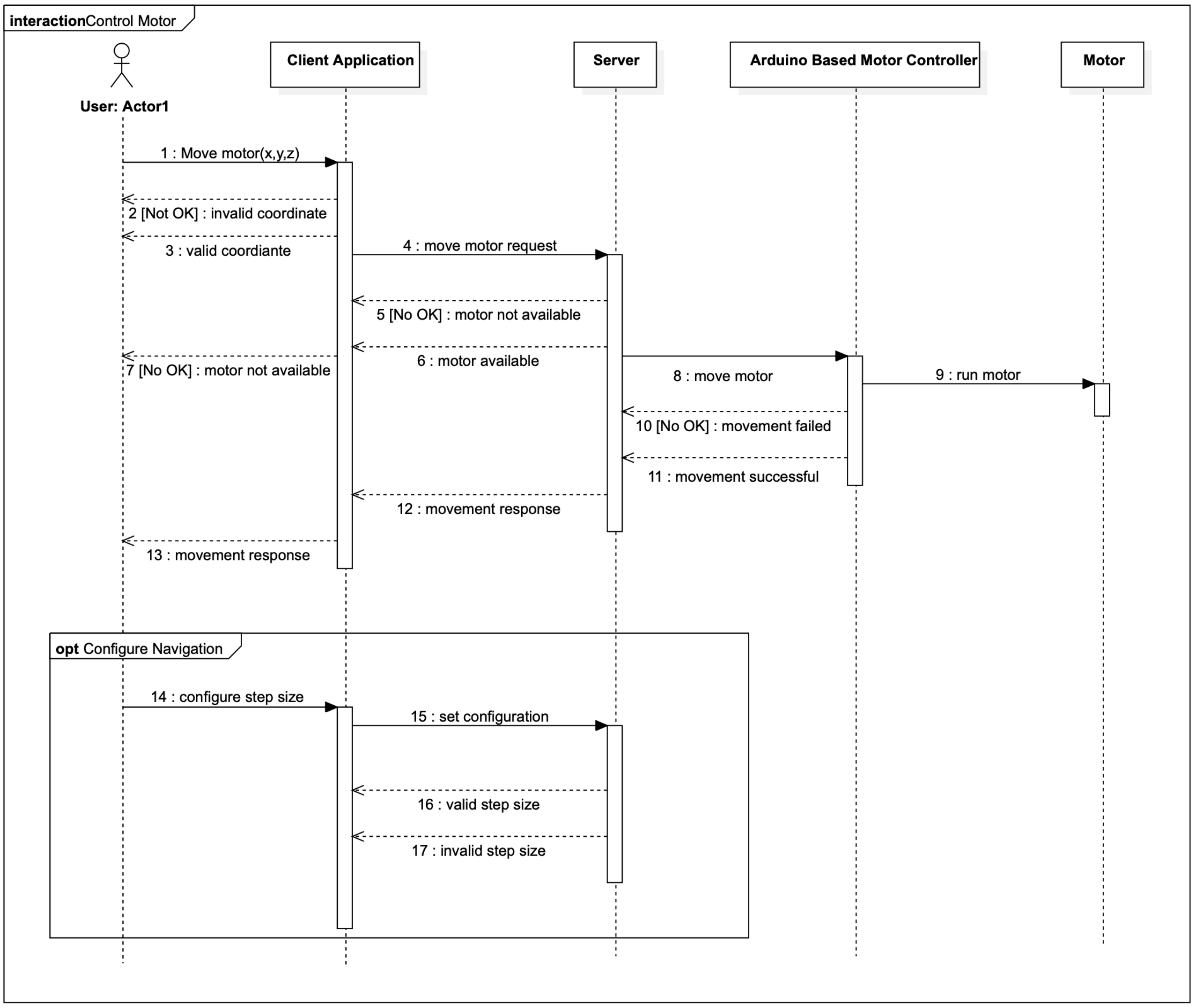


Figure 8: Sequence Diagram of “Control motors”

Table 7: Use Case 5

Use Case ID	5
Use Case Name	AutoFocus
Actors	Local User, Remote User
Descriptions	The Camera focuses on the sample and much better image and video quality is obtained.
Preconditions	The user is connected to the OFM. Use case number 4 is performed.
Postconditions	A focused and clearer camera feed level is achieved.
Normal Flow	1. Capture z stack images. 2. Z stack images are converted to gray scale. 3. By assigning higher values to higher spatial brightness variance, Laplacian convolution is applied. 4. Values are raised to fourth power and summed over image to provide sharpness value. 5. Returned to the z position with the highest sharpness.
Exceptions	1a. By capturing z stack images lock for stage movement is required. If lock cannot be obtained causing an exception. 3a. If a malfunction of the led bulb that is used to light the sample surface occurs, brightness value cannot be computed correctly. As a result, incorrect autofocusing occurs.

Table 8: Use Case 6

Use Case ID	6
Use Case Name	Calibrate Image
Actors	Local User, Remote User
Descriptions	Certain corrections techniques applied to remove imperfections on the image.
Preconditions	Camera must be functioning properly. The user must satisfy the use case 4.
Postconditions	An image is obtained which is cleaned from negative effects of the Sony IMX 219 image sensor.
Normal Flow	1. Take an image 2. Apply flat field correction 3. Correct Vignetting 4. Perform post processing
Exceptions	1a. Taken image must be focused. If they are not focused, images must be retaken.

Table 9: Use Case 7

Use Case ID	7
Use Case Name	Post Process the Image
Actors	Local User, Remote User
Descriptions	The image that is acquired using the camera is post processed to correct its

	saturation.
Preconditions	The image must be stored internally or remotely.
Postconditions	Reduction in saturation at the edges of the image is eliminated.
Normal Flow	1. Acquire and store the image. 2. Enter the interface of control camera either graphically or using scripting 3. Run the post processing algorithm.
Exceptions	1a. failure to access the image causes an error.

Table 10: Use Case 8

Use Case ID	8
Use Case Name	Correct Vignetting
Actors	Local User, Remote User
Descriptions	Live streams and raw images that are captured by OFM suffer from vignetting. Vignetting is corrected by some techniques in software.
Preconditions	The image must be present or real time streaming must be started.
Postconditions	Vignetting is removed from the image or real-time preview.
Normal Flow	1. The image is taken or streaming is started

	2. Access to the lens shading table in the camera's GPU-based image processing pipeline by the help of "picamera" library. 3. Correct vignetting by flat-field correction techniques.
Exceptions	2a. Shading table access requires reentrant locks. If the lock has not been acquired, the sequence results in an exception.

Table 11 : Use Case 9

Use Case ID	
Use Case Name	Apply Flat-Field Correction
Actors	Local User and Remote User
Descriptions	Optical artifacts such as shading or lens cast are removed from the image.
Preconditions	The image must be present or real time streaming must be started.
Postconditions	Optical artifacts are successfully removed from the image.
Normal Flow	1. A flat-field array of the image, which consists of each pixel's dark current and their gain, is created. 2. Using a flat-field array, the correction equation is applied for each pixel. 3. At the end, resulting image will be calibrated image
Exceptions	1a. In case flat-field array isn't created

	properly, optical artifacts in the image may still exist.

Table 12: Use Case 10

Use Case ID	9
Use Case Name	Camera-stage Mapping
Actors	Local User, Remote User
Descriptions	The relationship between the axes and step size of the translation stage is calibrated and the stage is mapped to camera location accordingly.
Preconditions	The user is connected to the OFM. Use case number 4 is performed.
Postconditions	The translation stage is successfully mapped to the camera location.
Normal Flow	1. The translation stage is moved back and forth along its x and y axes in order. 2. The displacement in the image ,captured by camera, is analyzed and translation stage is calibrated. 3. The calibrations are combined into a 2x2 affine transformation matrix which is used in mapping stage to camera coordinates.
Exceptions	3a. In case the stage isn’t appropriately mapped to camera location, the user can’t move the camera in the desired direction.

Table 13: Use Case 11

Use Case ID	10
Use Case Name	Scan (Tile Scan)
Actors	Local User and Remote User
Descriptions	Tile scan provides images with larger fields of view constructed by combining normal images for users.
Preconditions	The user is connected to the OFM. Auto focus, Image calibration and camera calibration works very well.
Postconditions	An image with a large field of view is created.
Normal Flow	1. Camera moves around the sample. 2. Before taking each image, the microscope is brought back into focus. 3. Camera captures images at a series of locations. 4. The system combines the captured images and reconstructs a complete image by using overlapped regions.
Exceptions	1a. In case the camera doesn’t move to the appropriate location, the camera can’t take pictures in these areas. This might cause some problems in creating the final image. 2a. If the microscope is not brought back to the focus, the captured image might be of poor quality.

	3a. In case the camera doesn't take images in some locations, this might cause some problems in creating the final image. 4a.If there are some locations in which camera doesn't take image : 1. The final image may not be created. 2. The final image might include some gaps. 4b. If there are some low-quality images: 1. The final image may not be created. 2. The final image might include poor quality regions.

Table 14: Use Case 12

Use Case ID	11
Use Case Name	Stream Video
Actors	Local User and Remote User
Descriptions	The OFM host provides real-time MJPEG live stream of the camera for the users.
Preconditions	The user is connected to the OFM.
Postconditions	The users can display live video streams from the microscope interface.

Normal Flow	1. A background thread recording JPEG (MJPEG) frames from the camera into a buffer is run on startup. 2. In case users connect to the system, the server starts to send a multi-part stream of frames, stored in the buffer, to users.
Exceptions	1a. In case the thread doesn't run or it is terminated, then JPEG frames can't be recorded. 2a. In case user disconnected from the server, live stream stops

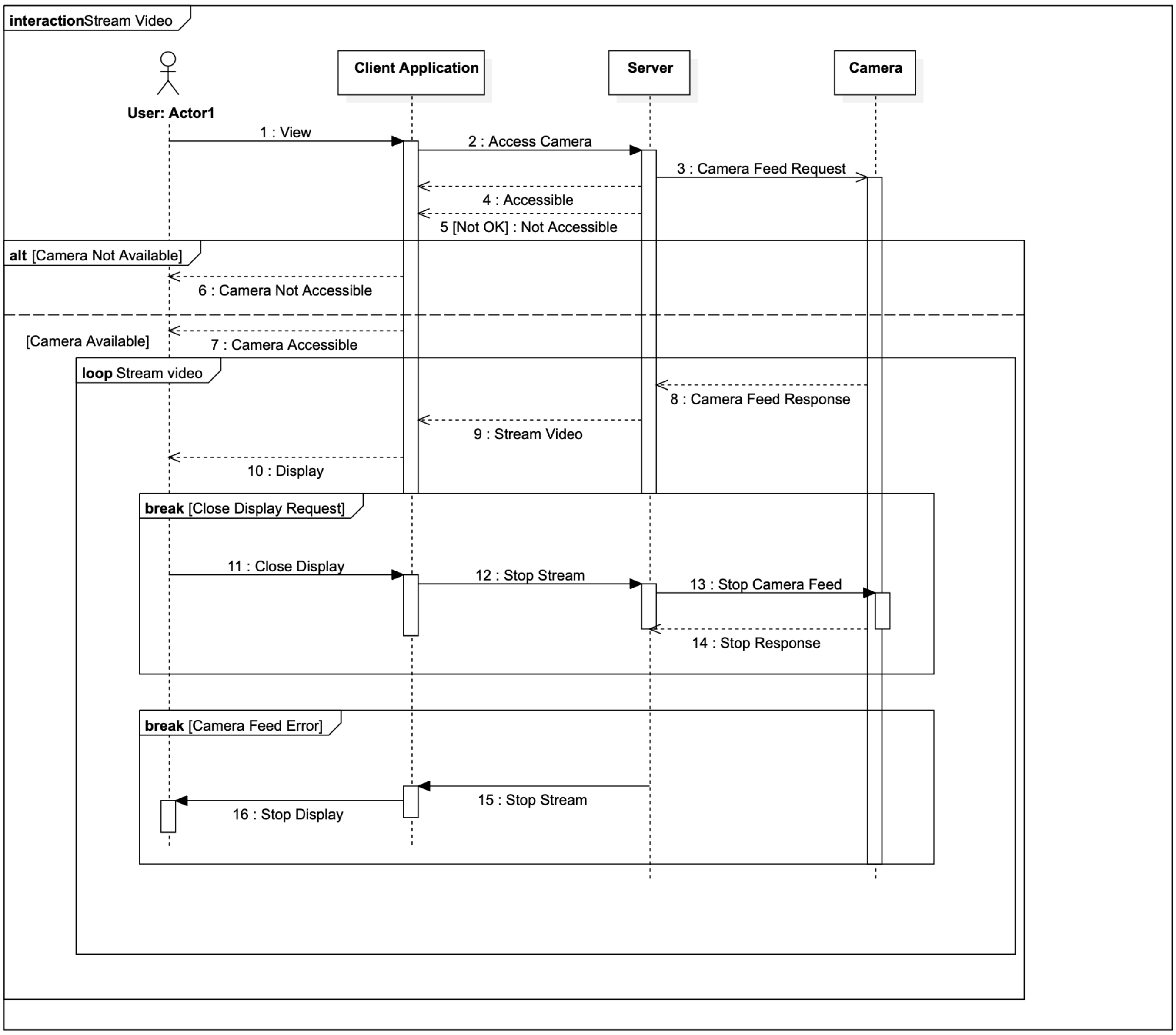


Figure 9: Sequence Diagram of “Stream Video”

Table 15: Use Case 13

Use Case ID	12
Use Case Name	Run MJPEG Algorithms
Actors	Local User and Remote User
Descriptions	MJPEG is a high quality video compression

	format in which each frame is compressed as a JPEG image.
Preconditions	The OFM streams video.
Postconditions	OFM successfully streams video as MJPEG format.
Normal Flow	1. Each frame in the video is compressed separately as a JPEG image format. 2. Resulting JPEG images are combined and the resulting video will be in MJPEG format.
Exceptions	1a. If video frames couldn't be compressed in JPEG format, MJPEG video won't be created.

Table 16: Use Case 14

Use Case ID	13
Use Case Name	Store Data
Actors	Local User and Remote User
Descriptions	The OFM system stores images and microscope state data for analysis and the user can access this data through HTTP API or data cable.
Preconditions	The user is connected to the OFM.
Postconditions	The User successfully accesses the every available data he/she wants.
Normal Flow	1. In case a picture is captured by the camera, the OFM system stores metadata about the state of the

	microscope during capturing picture including stage position,camera settings, calibration data, and custom metadata added by the user. The metadata is stored as JSON (JavaScript Object Notation) formatted string in the “UserComment” EXIF field. Furthermore, JPEG images and raw 8MP Bayar data collected from the camera are also stored for advanced analysis. Raspberry Pi stores these data on SD card or external USB storage device. 2. Through a data cable or HTTP API, the stored data can be accessed by users.
Exceptions	1a. In case of data corruption or hardware failure, users can’t access the data.

Table 17: Use Case 15

Use Case ID	14
Use Case Name	Extend Functionality
Actors	Remote User
Descriptions	Microscope Functionalities other than the most basic ones are provided with extensions.
Preconditions	The user is connected to the OFM.
Postconditions	OFM has a new functionality

Normal Flow	1. A Python script providing different functionality for the OFM is written. 2. OFM runs the script and now it has an additional functionality.
Exceptions	2a. In case the script includes a command which OSM can’t perform, desired functionality can’t be added.

Table 18: Use Case 16

Use Case ID	15
Use Case Name	Perform Remote Scripting of Experiments
Actors	Remote User
Descriptions	Python/MATLAB clients can transform web API into native Python/MATLAB functions. In that way, experiments can be performed remotely via scripts.
Preconditions	The user is connected to the OFM.
Postconditions	The User successfully makes OSM perform particular experiments via scripts .
Normal Flow	1. The user writes a python/MATLAB script which will perform a particular experiment. 2. The user sends the script into the server application. 3. OFM receive the script, process it, and perform the desired experiment.
Exceptions	2a. If the client lost connection between the server application while sending the packet,

	experiments can’t be performed 3a.In case the script includes a command which OSM can’t perform, experiment fails.

3.3 Usability Requirements

Usability requirements that eases the use of the OpenFlex Microscope are as following:

- Web and Desktop client applications should have simple and easy to use GUIs.
- The user shall use the microscopes features when the Internet is available.
- Users shall wait for the access to the motors due to its thread based accessibility.
- Camera feed shall be accessed with a low latency in the networking.
- Users shall be able to use open source libraries or the features that they developed to extend functionality of the OpenFlex Microscope.
- Users shall be provided with the support of various kinds of IO device integration. A monitor, a mouse, a gamepad, a keyboard shall be added to the system and configured according to the user needs.
- The system shall serve in a concurrent manner such that multiple users may access the same OFM at the same time.
- While providing concurrency, the system shall also prevent critical operations to take place in a synchronized way to prevent the disruption of these operations by using locks.
- MJPEG frames shall be synchronized to prevent the user getting the same frame twice.
- Running processes such as large tile scans must run in the background without blocking coming API requests.

3.4 Performance Requirements

- The response time should be less than 1 second to enable the user to access the real time camera feed and control motors.
- Connection to the OFM shall be less than 2 seconds.
- Post processing shall take less than 300msec to enable user access real time MJPEG live stream
- Raspberry pi camera must capture 8 megapixel bayer data at one shot.
- The internet connection must be over 10 Mbps.
- The microscope must support the connection of 50 users at the same time. Provided that it complies with the other requirements.
- The local storage must have enough capacity for the data that is acquired by the microscope.
- For the clients that are not able to receive frames at full rate, frame dropping shall be applied to reduce the latency.

3.5 Logical Database Requirements

There is no actual database in the system. A primitive level of storage units are present just to perform buffering of acquired data. Outside the system data can be stored by the client if desired but the system itself does not have a database. However, external storage units can be added.

3.6 Design Constraints

- All the user data must be kept separate from users data that has access to the microscope.
- The system shall be designed to conform to laws and privacy.

3.7 Software System Attributes

3.7.1 Reliability

- The system must be tested before deploying the application for the usage of the clients. Error probability should not be more than 0.0001.
- Data corruption must be minimized. To satisfy this goal, the system must buffer tha data of users individually and user capacity should not be exceeded.
- Every new extension should be tested before putting it into usage.

3.7.2 Availability

- The system should be available during the operating hours of the clinicians. In order to satisfy this goal, multiple server instances in the Raspberry Pi shall be used.
- The system shall have error handling functionalities to recover from possible erroneous situations.

3.7.3 Security

- Since the number of connections must be kept limited by the project specification, this shall not be a system weakness. As a result, the system security shall be ensured against SYN flood attacks.
- Security of the user data shall be ensured by internal synchronization and separate buffering mechanisms.

3.7.4 Maintainability

- An expressive documentation of the system should be provided.

- Git version control shall be used publicly enabling any contributions from the open source world.
- Developers shall follow a coding style which is extensible, and has low coupling and high cohesion.

3.7.5 Portability

- The system shall be available for the usage of the user that want to access via Web, Desktop environment and remote scripting which may have any OS with modern features.
- The system shall use multi platform supported components
- The extensions shall be produced to be compatible with major operating environments.
- System client applications for different operating systems shall be available.

3.8 Supporting Information

The OFM is a customizable optical microscope using microscope objectives or an inexpensive webcam and a mechanical stage that can be moved by motors. The OFM uses elastic material which is vibration absorbing and frictionless. If mini-stepper motors are used , its size of step is around 100 nanometers which is well enough in microscopic scale. It is also stable for many days within a few microns.