

---

# **Camera Network Documentation**

***Release 1.0.0***

**Amit Aides**

**Sep 30, 2021**



---

# Contents

---

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>7</b>
2.1	Installation . . . . .	7
2.1.1	Installation - Client . . . . .	8
2.1.2	Installation - Server . . . . .	9
2.1.3	Installation - Proxy . . . . .	9
2.1.4	Installation - Calibration Station . . . . .	9
2.1.5	Camera setup . . . . .	9
2.1.5.1	Arduino connections . . . . .	9
2.2	Installation - OLD . . . . .	9
2.2.1	Prerequisites . . . . .	9
2.2.2	CameraNetwork Installation . . . . .	10
2.2.3	Installing the Client . . . . .	11
2.2.4	Installing the Calibration Station . . . . .	11
2.2.5	Shubi reference . . . . .	11
<b>3</b>	<b>Calibrating the Camera Network</b>	<b>13</b>
3.1	Geometric Calibration . . . . .	13
3.1.1	Intrinsic Calibration . . . . .	14
3.1.2	Extrinsic Calibration . . . . .	14
3.2	Radiometric Calibration . . . . .	14
<b>4</b>	<b>Camera</b>	<b>15</b>
<b>5</b>	<b>Using the Camera Network Package</b>	<b>17</b>
5.1	Client . . . . .	18
5.1.1	sun Shader . . . . .	21
5.1.2	Sprinkler . . . . .	22
5.2	Camera (server) . . . . .	22
5.2.1	Field Deployment . . . . .	22
5.2.2	Code . . . . .	22
5.2.3	Connection . . . . .	22
5.2.3.1	Serial connection . . . . .	23
5.2.3.2	SSH . . . . .	23
5.2.3.3	GUI . . . . .	23
5.3	Proxy . . . . .	23
5.3.1	To connect to the proxy . . . . .	23
5.3.2	Noticable stuff . . . . .	23
5.4	Others . . . . .	24
5.4.1	Image Acquisition flow . . . . .	24

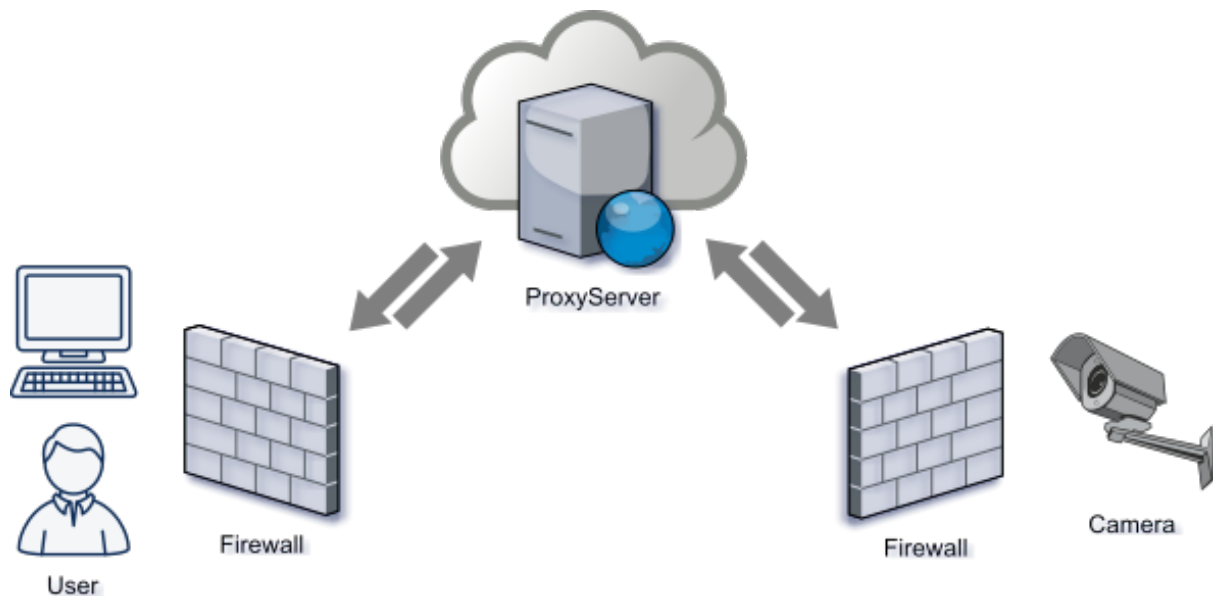
5.4.2	Useful commands . . . . .	24
5.4.3	Data Structures . . . . .	24
5.4.4	Analyzing Results . . . . .	26
<b>6</b>	<b>Indices and tables</b>	<b>27</b>

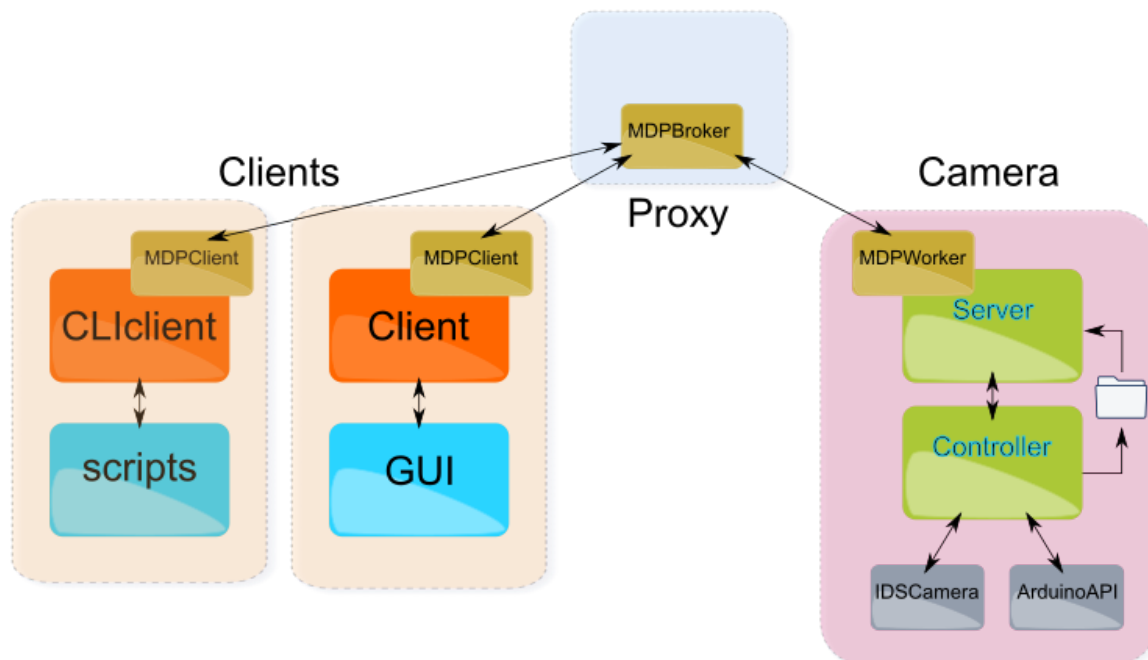
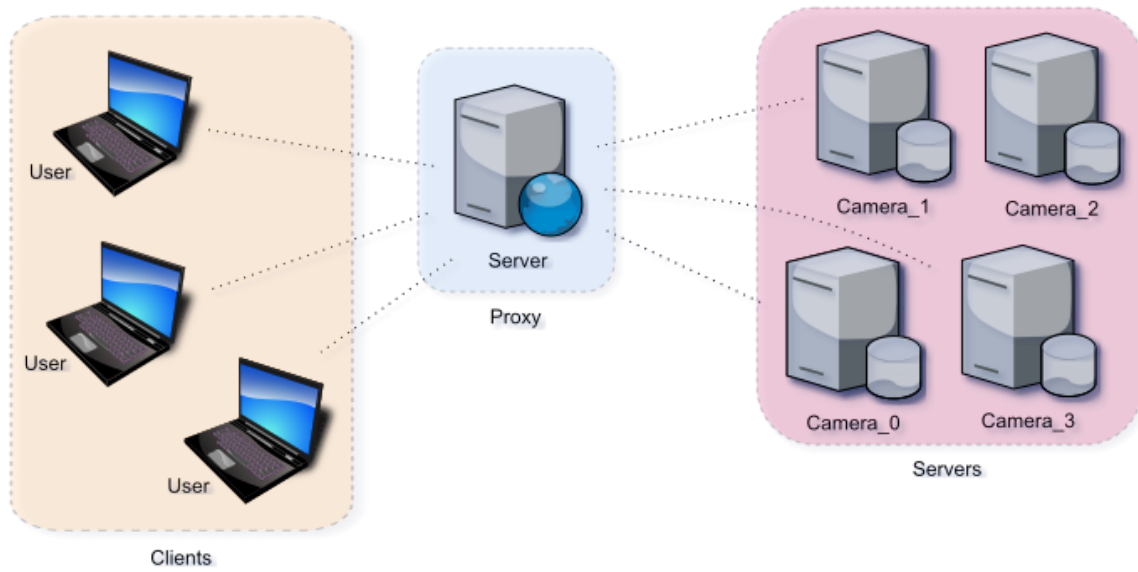
Contents:



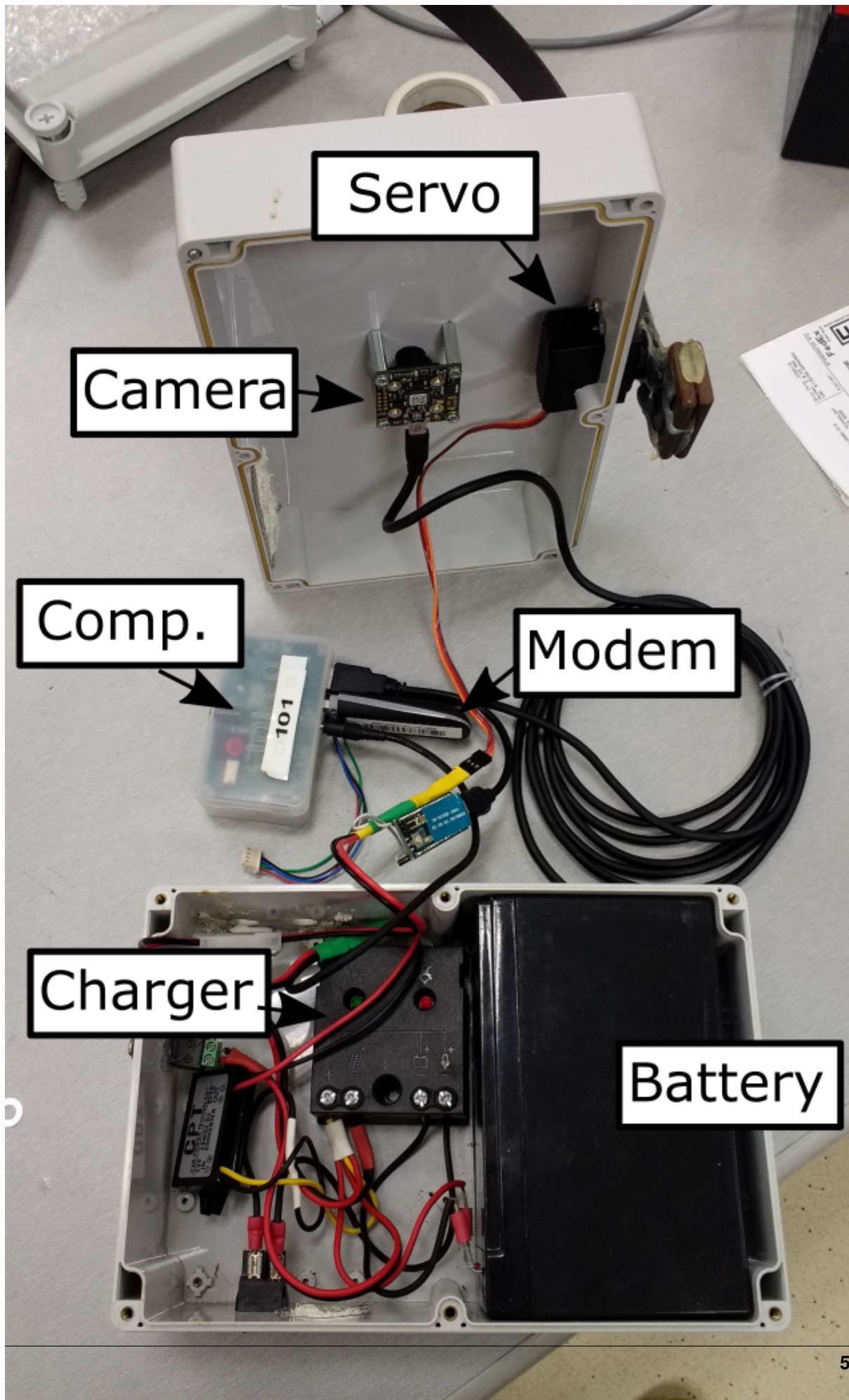
# CHAPTER 1

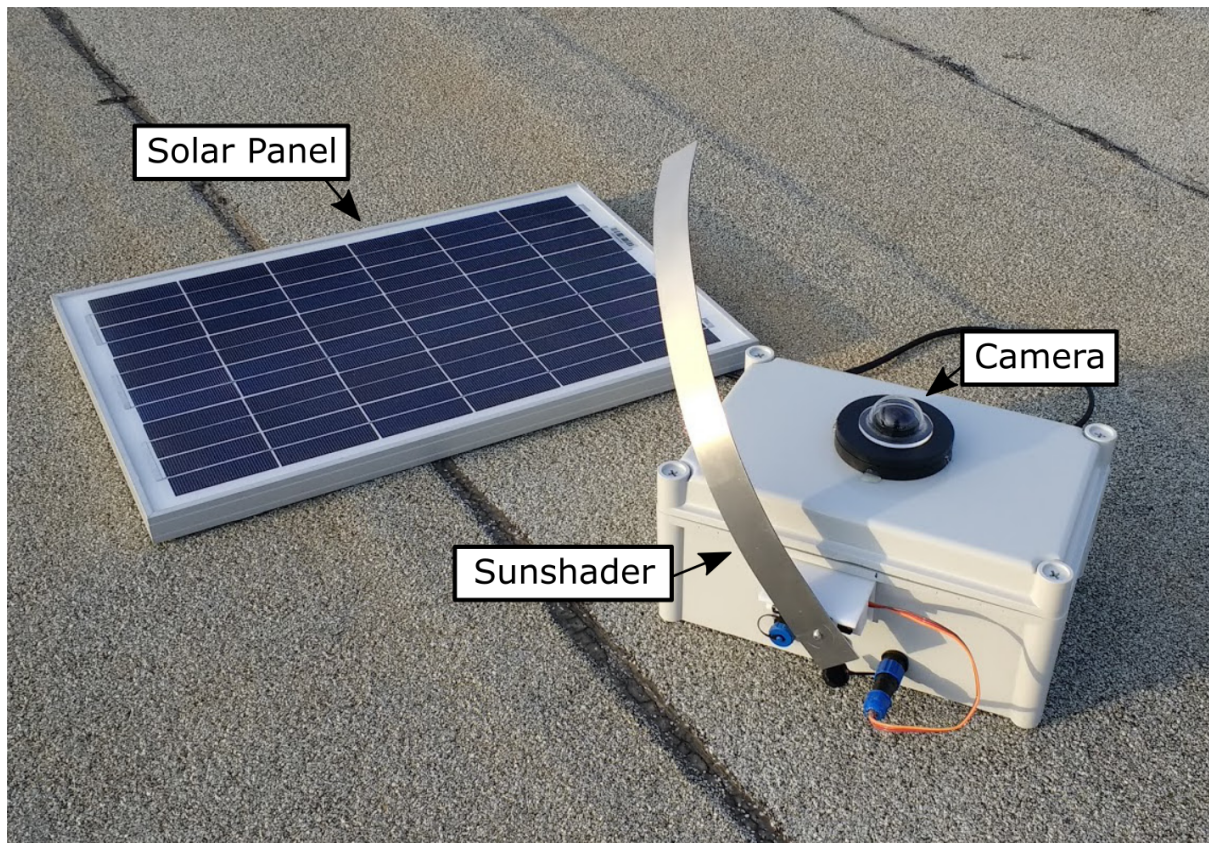
## Introduction











### Table of Contents

- *Installation*
  - *Installation*
    - \* *Installation - Client*
    - \* *Installation - Server*
    - \* *Installation - Proxy*
    - \* *Installation - Calibration Station*
    - \* *Camera setup*
      - *Arduino connections*
  - *Installation - OLD*
    - \* *Prerequisites*
    - \* *CameraNetwork Installation*
    - \* *Installing the Client*
    - \* *Installing the Calibration Station*
    - \* *Shubi reference*

## 2.1 Installation

The `CameraNetwork` system is made of three logical parts:

1. *Server*: The camera unit. The server performs the actual measurements.
2. *Client*: A program that enables remote control of servers.
3. *Proxy*: A program that bridges and manages the communication between the *Servers* and *Clients*.

There can be multiple *Servers* and *Clients* but only one *proxy*.



The CameraNetwork package contains the code for both the *Server*, *Client* and *Proxy* subsystems. This simplifies the deployment and enables code reuse. The installation procedures is similar for the three components but differs due to the different platforms.

The CameraNetwork is implemented completely in *Python* <<http://www.python.org/>>\_.

### 2.1.1 Installation - Client

1. Install conda. Tested on conda 4.7.11
2. Clone the cameranetwork package:

```
git clone https://github.com/Addalain/cameranetwork.git
```

3. Navigate to it:

```
cd cameranetwork
```

4. Create virtual env:

LINUX: Create conda virtual environment from *cn\_client\_ubuntu18.yml*

```
conda env create -f cn_client_ubuntu18.yml
```

---

**Note:** The first line of sets the new environment's name (currently *cn\_client*)

---

WINDOWS (exact procedure):

```
# Create new environment with name: cn_client

conda create -n cn_client --yes

conda activate cn_client

conda config --env --set restore_free_channel true

conda config --env --append channels conda-forge

conda install python=2.7 pip paramiko cython tornado=4.5.3 futures_
↳numpy scipy matplotlib beautifulsoup4 scikit-learn scikit-image_
↳pyside requests ephem pandas=0.19.2 ipython pyfirmata joblib pyzmq_
↳enaml pillow traits pyqtgraph pyopengl vtk mayavi opencv git_
↳mercurial

# pip install pyp3d, traits-enaml and pyfisheye
# Note, this installs pyfisheye without cloning it. For development of
↳pyfisheye clone and install manually from https://bitbucket.org/
↳amitibo/pyfisheye (TODO: migrate pyfisheye codebase to github)

python -m pip install pyp3d==1.1.1 git+https://github.com/enthought/
↳traits-enaml.git@update-data-frame-table hg+https://bitbucket.org/
↳amitibo/pyfisheye
```

5. Activate the environment:

```
conda activate <venv_name>
```

6. Install the cameranetwork package

```
python setup.py develop --user
```

---

**Note:** without `--user` it installs the scripts for all users (Windows: `C:\ProgramData\Anaconda2\Scripts`)

---

7. Verify successful installation by opening the GUI:

```
python scripts_client/camera_client.py
```

## 2.1.2 Installation - Server

The server software is run on an [Odroid U3](#) as at the time of selection it offered a unique balance between capabilities and cost. Nonetheless it should be straight forward to install the `CameraNetwork` package and its prerequisites on other platforms like newer Oroids and even on the [RaspberrPi](#).

In the following we detail the procedure of installing the required prerequisites and main package. Note that once the package is installed on one computer, it is much more time effective to create an image of the Odroid memory card and duplicate it as needed.

## 2.1.3 Installation - Proxy

Currently the code assumes that the proxy server is run on an ec2 instance. Installation on the proxy follows the same steps of installation on the client.

To run the proxy program, do:

```
start_proxy.py
```

## 2.1.4 Installation - Calibration Station

## 2.1.5 Camera setup

### 2.1.5.1 Arduino connections

Savox SunShader Servo pins:

1. Brown (Gnd) = Gnd
2. Red (5V) = 5V
3. Orange (Signal) = PIN NUM

## 2.2 Installation - OLD

### 2.2.1 Prerequisites

To use *CameraNetwork* several software package are needed. This can be installed using the following commands. Copy paste these to a commandline:

```
> sudo apt-get install python-pip git mercurial screen autossh
> sudo pip install paramiko
> sudo pip install cython
> sudo pip install pyzmq --install-option="--zmq=bundled"
```

(continues on next page)

(continued from previous page)

```
> sudo pip install tornado==4.5.3
> sudo pip install futures
> sudo apt-get install python-numpy python-scipy python-matplotlib
> sudo pip install beautifulsoup4
> sudo pip install sklearn
> sudo pip install skimage
> sudo pip install ephem
> sudo pip install pandas
> sudo pip install pymap3d
> sudo pip install ipython
> sudo pip install pyfirmata
> sudo pip install joblib
```

To install opencv3 follow a tutorial relevant to your system, e.g. on Odroid XU4 the following tutorial was useful [opencvsh\\_for\\_ubuntu\\_mate](#).

Install the python wrappers to the ids SDK:

```
> mkdir code
> cd code
> git clone https://github.com/amitibo/ids.git
> cd ids
> sudo python setup.py install
```

Install the pyfisheye module:

```
> cd ~/code
> hg clone https://amitibo@bitbucket.org/amitibo/pyfisheye
> cd pyfisheye
> sudo python setup.py install
```

Some platforms might require the installation of modem software:

```
> sudo apt-get install network-manager
> sudo apt-get install network-manager-gnome
```

The first instal *nmcli* (used for activating the connection). The second intalls *nmcli-connection-editor* used for defining the mobile network connection.

Install a recent version of *usb\_modeswitch* (required on raspberryPi). Follow the [usb\\_modeswitch tutorial](#). To compile the above code you will need to install the *libusb-1* dev files:

```
> sudo apt-get install libusb-1.0-0-dev
```

Prepare a device reference file from the following [device reference file](#) and run it using the command:

```
> sudo usb_modeswitch -c <path to device file>
```

## 2.2.2 CameraNetwork Installation

Download and install the package:

```
> git clone https://amitibo@bitbucket.org/amitibo/cameranetwork_git_
↪cameranetwork
> cd cameranetwork
> python setup.py develop --user
```

---

**Note:** The first command downloads a *slim* version of the code that only includes the *Server* components.

---

To make the system start automatically at boot time, we use the *rc.local* script:

```
> sudo cp cameranetwork/scripts/rc.local/rc.local /etc/rc.local
```

Run the camera setup script to setup the camera environment.

```
> setup_camera.py
```

You will be asked for a camera id. Enter a unique camera id number.

### 2.2.3 Installing the Client

It is recommended to install python using the [Anaconda](#) distribution. Install the CameraNetwork package:

```
> git clone https://amitibo@bitbucket.org/amitibo/cameranetwork_git.git_
↪cameranetwork
> cd cameranetwork
> python setup.py develop --user
```

### 2.2.4 Installing the Calibration Station

It is recommended to install python using the [Anaconda](#) distribution. Install the CameraNetwork package:

```
> git clone https://amitibo@bitbucket.org/amitibo/cameranetwork_git.git_
↪cameranetwork
> cd cameranetwork
> python setup.py develop --user
```

### 2.2.5 Shubi reference

1. Create conda virtual environment:

```
conda create --name <venv_name> --no-default-packages
conda config --add channels conda-forge
conda activate cnenv
```

2. Install prerequisites:

```
conda install python=2.7 pip paramiko cython tornado=4.5.3 futures numpy scipy_
↪matplotlib beautifulsoup4 scikit-learn scikit-image ephem pandas ipython_
↪pyfirmata joblib
pip install pyzmq --install-option="--zmq=bundled"
pip install pymap3d
conda install enaml pillow traits pyqtgraph pyopengl vtk mayavi opencv
```

3. Install additional modules:

```
pip install ephem
conda install -c anaconda pil
conda install -c anaconda enaml
conda install -c anaconda traits pyqtgraph pyopengl
conda install -c anaconda vtk
pip install mayavi
```

4. Install traits-enaml:

```
git clone https://github.com/enthought/traits-enaml.git --branch update-data-  
↪frame-table  
cd traits-enaml  
python setup.py install  
cd..
```

### 5. Install the cameranetwork package

1. Navigate back to cameranetwork:

```
cd ..
```

2. Install the cameranetwork package:

```
python setup.py develop --user
```

---

**Note:** without `--user` it installs the scripts for all users (Windows: `C:\ProgramData\Anaconda2\Scripts`)

---



---

### Calibrating the Camera Network

---

Calibration of the camera network is done in several stages. Part of it is done in the lab using a custom calibration setup. Another part is done in the field using the sunphotometer of the [Aeronet system](#).

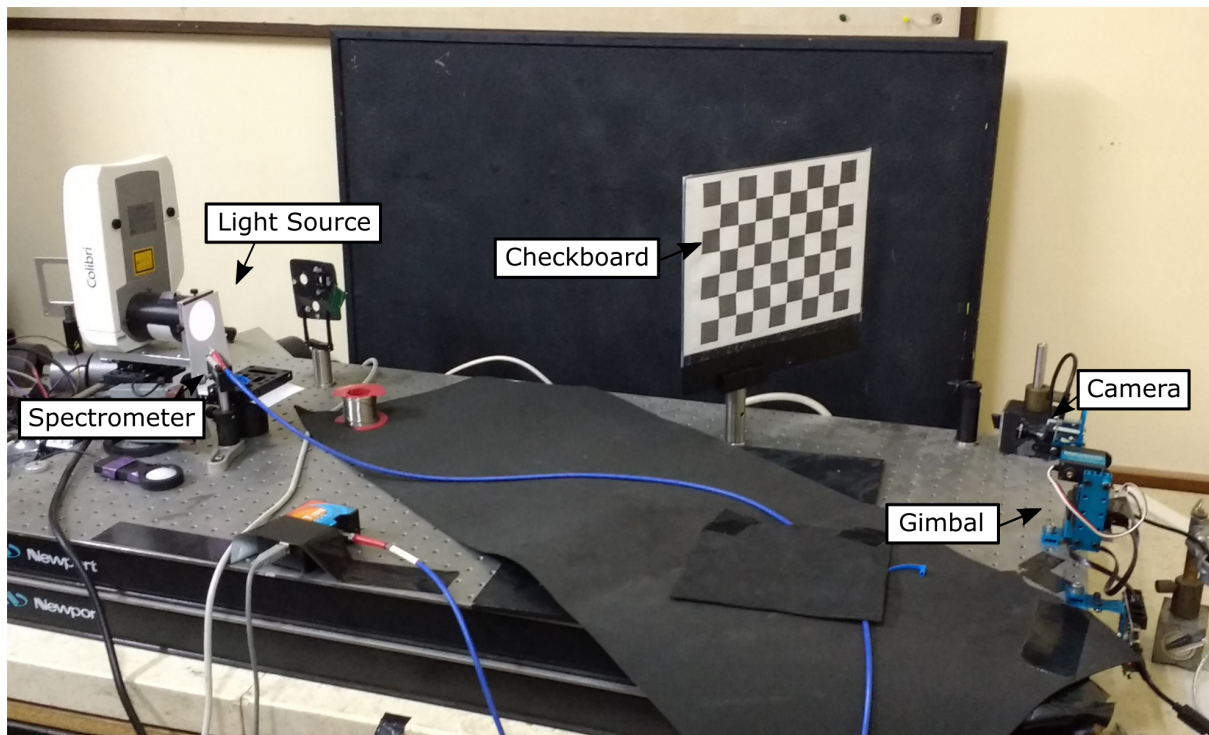


Fig. 3.1: Camera calibration setup.

### 3.1 Geometric Calibration

Geometric calibration is done

### 3.1.1 Intrinsic Calibration

Intrinsic calibration is done using a checkboard target and using the `pyfisheye` package which is based on the `opencv fisheye` calibration toolbox.

### 3.1.2 Extrinsic Calibration

Extrinsic calibration is the determination of the orientation of the camera in the global coordinates system. It is based on tracking the sun position.

## 3.2 Radiometric Calibration

Each camera is calibrated relation the a custom calibration setup (Figure :num:'calibration-setup').

## CHAPTER 4

---

Camera

---



---

## Using the Camera Network Package

---

### Table of Contents

- *Using the Camera Network Package*
  - *Client*
    - \* *sun Shader*
    - \* *Sprinkler*
  - *Camera (server)*
    - \* *Field Deployment*
    - \* *Code*
    - \* *Connection*
      - *Serial connection*
      - *SSH*
      - *GUI*
  - *Proxy*
    - \* *To connect to the proxy*
    - \* *Noticable stuff*
  - *Others*
    - \* *Image Acquisition flow*
    - \* *Useful commands*
    - \* *Data Structures*
    - \* *Analyzing Results*

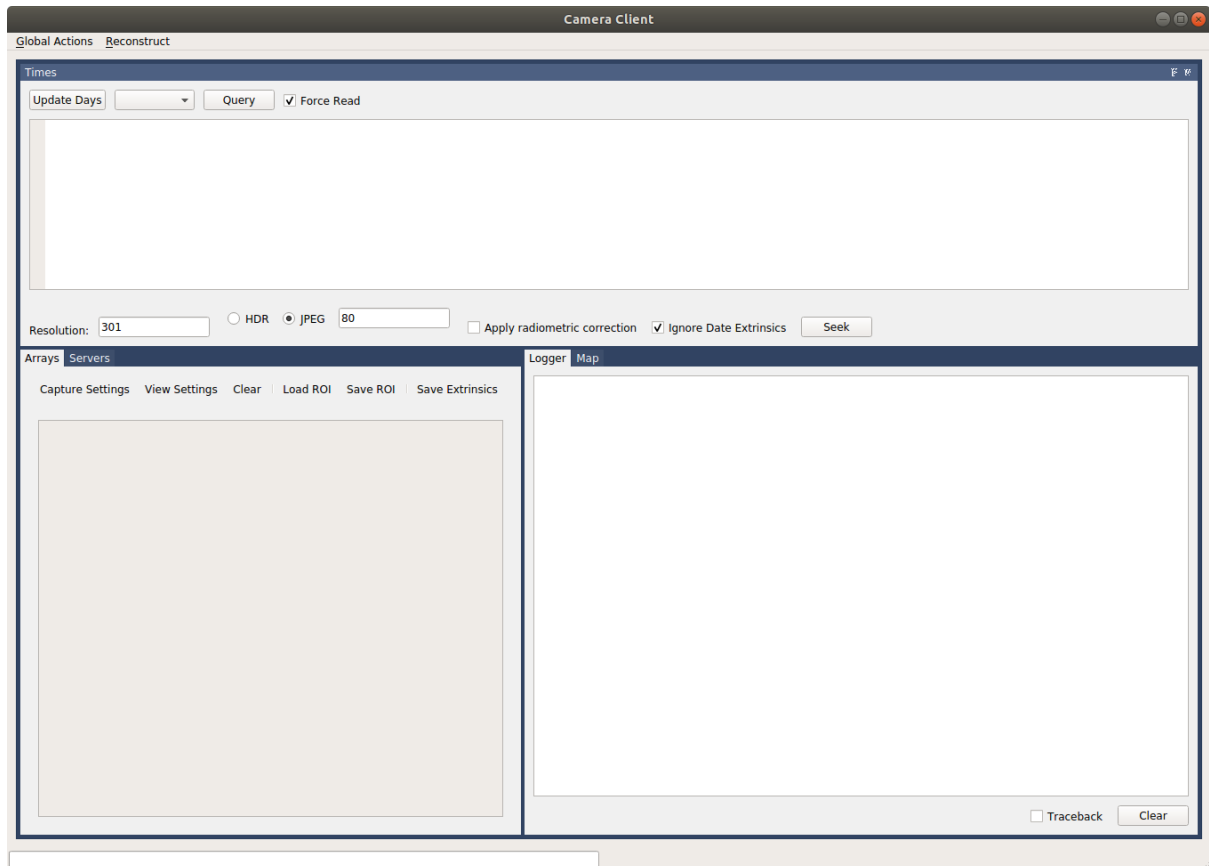
## 5.1 Client

After successful installation, start the Client GUI by navigating to:

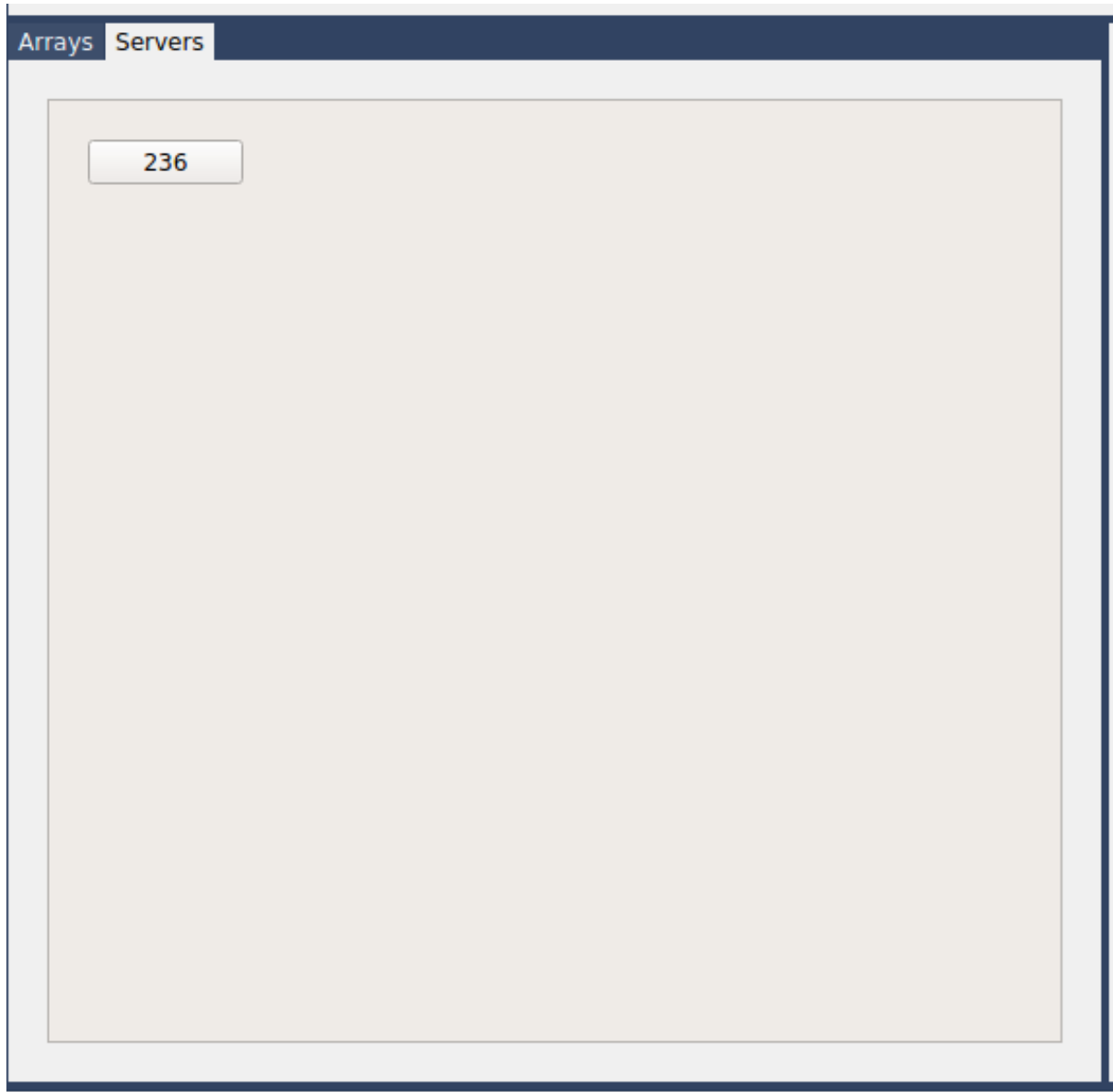
```
cd cameranetwork/scripts_client
```

then run `python camera_client.py`

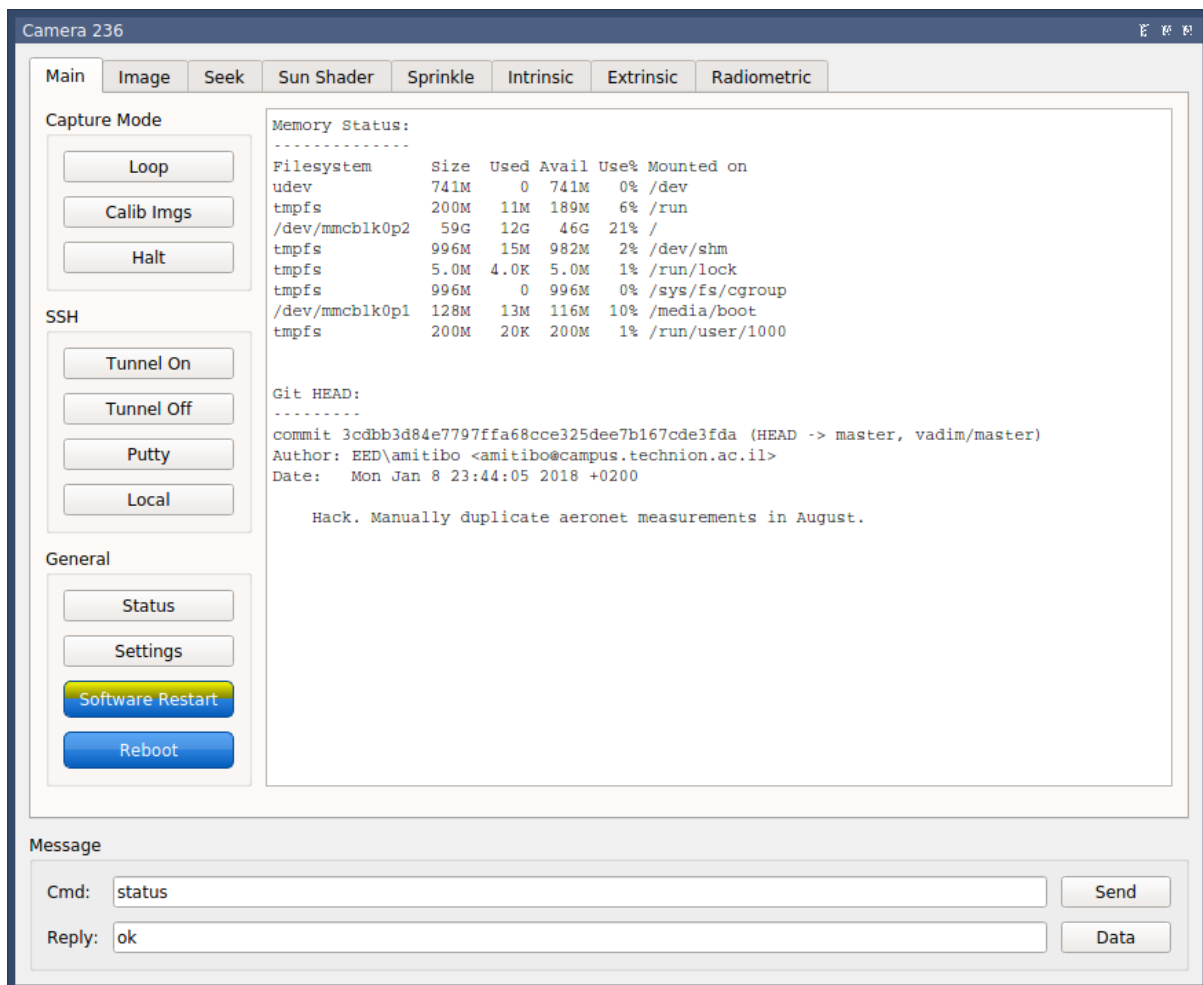
You should now see



after pressing on servers, you should see all connected cameras, in this case camera id 236.

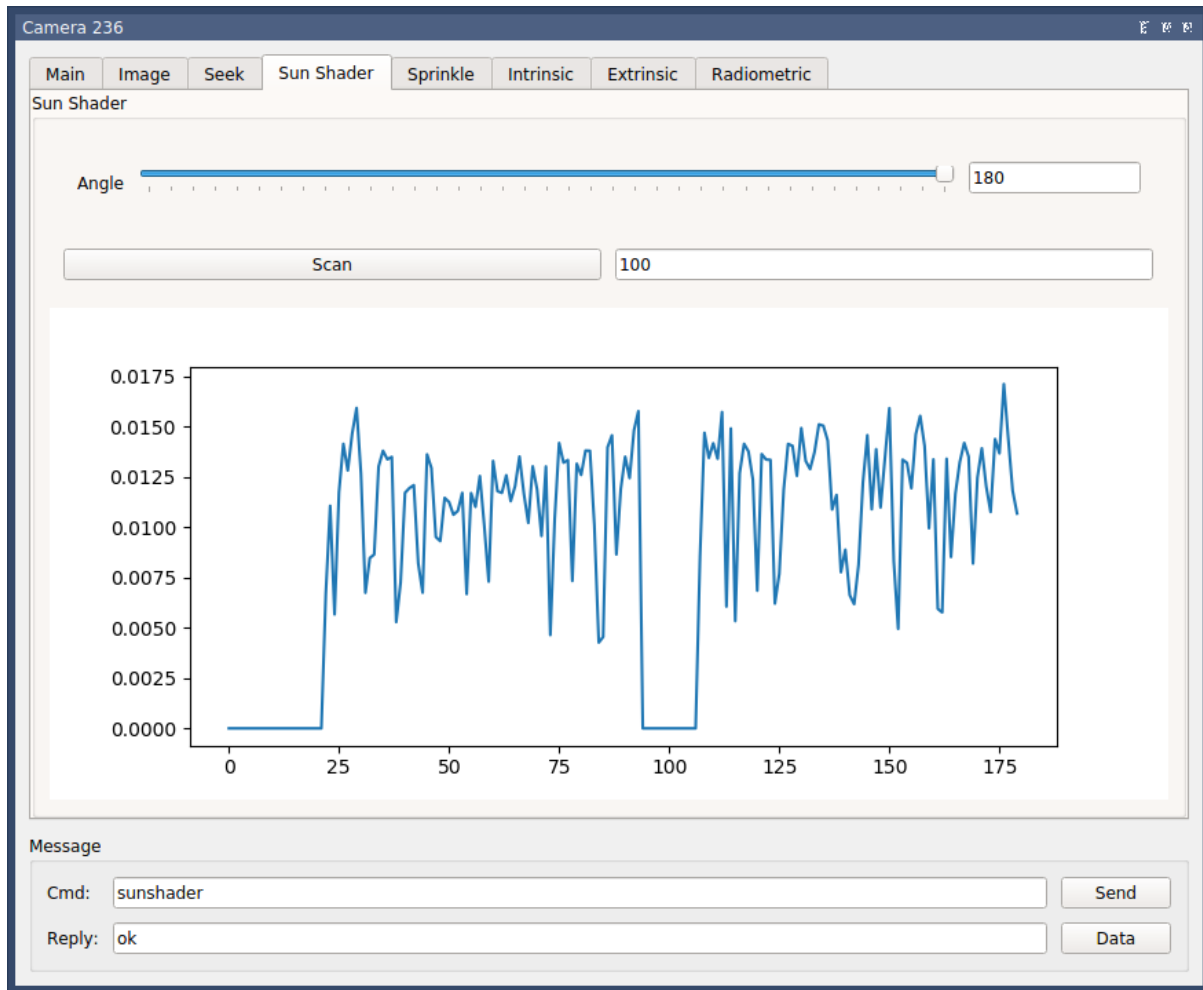


pressing on the camera ID should lead to the camera interface screen



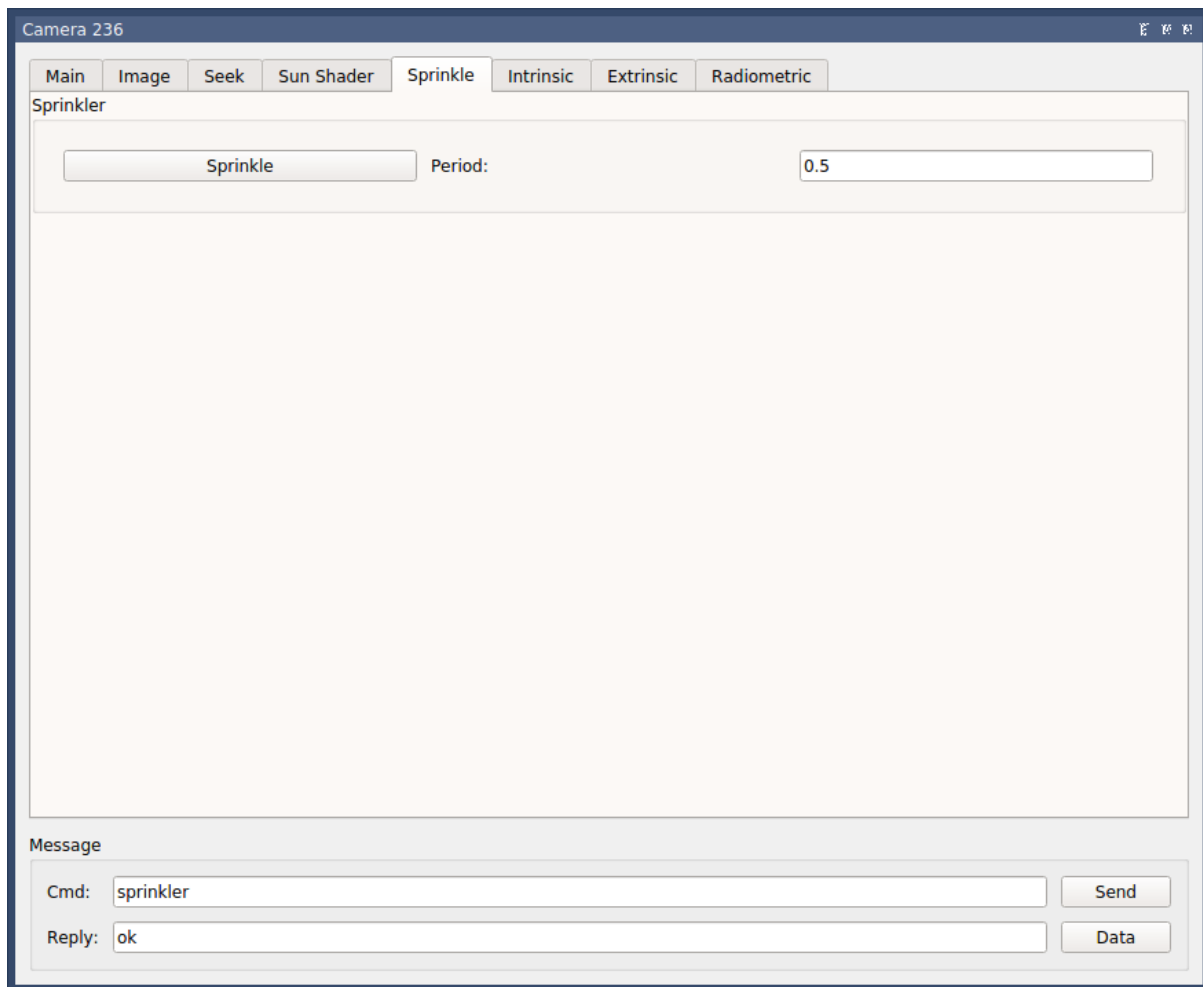


### 5.1.1 sun Shader



The angle slider allows manual setting of sunshader angle. The scan button moves the sunshader throughout it's whole range, then according to the point with least sunlight (as shown in the graph) determines the sunshader optimal angel.

## 5.1.2 Sprinkler



The Sprinkle tab and corresponding button allows to manually activate the sprinkler system in order to clean the camera len dome. Period Refers to activation time in seconds.

## 5.2 Camera (server)

### 5.2.1 Field Deployment

1. Verify Case Screws fully tightened.
2. Verify plugs fully screwed.
3. Verify sprinklers are pointing in the right direction
4. Verify camera alignment with north-south

### 5.2.2 Code

After changing any `global_setting.py` parameter, need to run `setup_camera.py` again.

### 5.2.3 Connection

There are options to connect to the camera

### 5.2.3.1 Serial connection

On the Client's PC, from cameranetwork folder:

`bash scripts/listusb.sh` to list all connected usb devices and to find the relevant one. Should be `/dev/tty/USB0` (replace '0' with relevant number)

1. Follow [driver installation instructions by Odroid](#).
2. Linux: Run `sudo minicom` in Client PC's terminal. Windows: Use Teraterm.
3. Enter odroid username
4. Enter odroid password

### 5.2.3.2 SSH

1. Via SSH

### 5.2.3.3 GUI

1. Via GUI (as mentioned in the client section)

## 5.3 Proxy

### 5.3.1 To connect to the proxy

```
sudo ssh -i <path_to_key> ubuntu@<proxy_ip>
```

---

**Note:** `sudo chmod 400 <path_to_private_key>` if permission error is encountered.

---

---

**Note:** `<path_to_key>` is the path and name of the proxy's private key `<proxy_ip>` is defined in `global_settings.py`. Currently `3.123.49.101`

---

If this is the initial setup of the proxy server:

```
screen _S session_name
python ./code/cameranetwork/scripts_proxy/start_proxy.py --log_level info
```

Should be run from the root of the server, otherwise the logs would be put in a different location each time. Screen is used to be able to detach and retrieve when ever needed.

- Press `ctrl+a` then `ctrl+d` to detach the `start_proxy.py` from the terminal
- `screen -ls` to see detached processes. then `screen -r <name>` to bring it back.

### 5.3.2 Noticable stuff

`tunnel_port_<camera_id>.txt` stores the odroid's password and `tunnel_port` (random int between 20,000 and 30,000).

`/proxy_logs/cameralog_<date+time of ____ initialization>_proxy.txt` is a log. Mainly shows Heartbeats from connected cameras and notification of message transmissions to/from the client.

## 5.4 Others

### 5.4.1 Image Acquisition flow

On Odroid: rc.local -> main(start\_server.py) -> start(server.py).278 -> loop\_timer(server.py) -> handle\_loop(controller.py) -> safe\_capture(controller.py) -> IDSCamera.capture (cameras.py)

### 5.4.2 Useful commands

- `ps -ef | grep python` to view running python processes (should see start\_proxy.py!)
- `sudo netstat -a -nlp -o | grep 198` to see status of relevant ports
- adding ssh key to ssh-agent.
- [How to use scp to transfer files](#). For example to retrieve proxy log from proxy to client: `scp ubuntu@3.123.49.101:/home/ubuntu/proxy_logs/cameralog_190929_092735_proxy.txt /home/shubi/Desktop/log`
- gparted for microsd / eMMC partitioning & direct copying.
- `sudo dd if=/dev/sdb of=~ /xu4_lab.img status=progress` to create an image of odroid
- [etcher](#) to flash image onto the SD card
- `grep -a -e "Unregistering worker 236" -e "Registering new worker 236" cameralog_190929_092735_proxy.txt` to see connections and disconnections. replace log.txt with \* for all logs in folder.

### 5.4.3 Data Structures

When looking at a specific camera, under *captured\_images*, for each that the camera recorded a folder `<%Y-%M-%D>` is created. Inside, the images are stored as *.mat* files. In addition there is a thumbnail *.jpg* version, add metadata as *.pkl*. The name is *utctime\_date+exact time*. The *.pkl* file stores the following data:

```
img = pd.read_pickle('~ /captured_images /2019_10_02 /1570011900.0_2019_10_02_10_25_00_00_3.pkl')
```

```

▼ img_data = {DataObj} <CameraNetwork.utils.DataObj object at 0x7f0ebb338590>
  01 altitude = {int} 229
  ▼ camera_info = {dict} {'global_shutter': False, 'sensor_name': 'UI155xLE-C', 'l... Vi
    01 'global_shutter' (139701247260352) = {bool} False
    01 'sensor_name' (139701247256304) = {str} 'UI155xLE-C'
    01 'hw_version' (139701247012960) = {str} 'V1.0'
    01 'color_mode' (139702319758912) = {str} 'Bayer'
    01 'master_gain' (139701247013056) = {bool} True
    01 'max_height' (139701247256832) = {int} 1200
    01 'blue_gain' (139701247256640) = {bool} True
    01 'red_gain' (139701247013920) = {bool} True
    01 'sensor_id' (139701246946448) = {int} 39
    01 'max_width' (139701247015504) = {int} 1600
    01 'green_gain' (139701247256496) = {bool} True
    01 'manufacture_date' (139701247259960) = {str} '16.01.2017'
    01 'serial_num' (139701247013296) = {str} '4103098529'
    01 'type' (139701247013104) = {str} 'USB uEye LE'
    01 'id' (139701246972672) = {int} 1
    01 'pixel_size' (139701247256544) = {float} 2.8
    01 'manufacturer' (139701247260296) = {str} 'IDS GmbH'
    01 __len__ = {int} 17
  ▶ capture_time = {datetime} 2019-10-02 10:25:04.264311
  01 color_mode = {unicode} u'raw'
  01 exposure_us = {float} 522.64
  01 gain_boost = {bool} True
  01 gain_db = {long} 0
  01 latitude = {float} 32.775776
  01 longitude = {float} 35.024963
  ▶ name_time = {datetime} 2019-10-02 10:25:00.001113

```

In addition, one *database.pkl* is created and stored per day:

```

database = pd.read_pickle('~/.captured_images/2019_10_02/database.pkl')
database.head()

```

Time	hdr	path	longitude	latitude	altitude	serial_num
2019-10-02 00:00:00	0	/home/odroid/captured_images/2019_10_02/1569974400.05_				
↪ 2019_10_02_00_00_00_0.mat			35.024963	32.775776	229	4103098529
2019-10-02 00:30:00	0	/home/odroid/captured_images/2019_10_02/1569976200.05_				
↪ 2019_10_02_00_30_00_0.mat			35.024963	32.775776	229	4103098529
2019-10-02 01:00:00	0	/home/odroid/captured_images/2019_10_02/1569978000.05_				
↪ 2019_10_02_01_00_00_0.mat			35.024963	32.775776	229	4103098529
2019-10-02 01:30:00	0	/home/odroid/captured_images/2019_10_02/1569979800.05_				
↪ 2019_10_02_01_30_00_0.mat			35.024963	32.775776	229	4103098529
2019-10-02 08:48:03	0	/home/odroid/captured_images/2019_10_02/1570006083.33_				
↪ 2019_10_02_08_48_03_0.mat			35.024963	32.775776	229	4103098529

### 5.4.4 Analyzing Results

On Client PC:

```
cd /cameranetwork/scripts_client
python start_local.py <path_to_experiment_data>
```

---

**Note:**

- Make sure to activate environment beforehand: `conda activate cn_client`
  - `-l` flag is used for local proxy (instead of real proxy server)
  - `-d` flag is for opening gui separately.
- 

workflow + data structure:

#. Run `python start_local.py -d ~/experiment_23_09_2019` in the background where *experiment\_date* is a folder containing *cam\_ID* folder for each camera involved. Each *cam\_ID* should consist of

1. *captured\_images* folder which stores inside a folder with images(.jpg, .mat & .pkl versions) and database.pkl for each day that the camera recorded.
  2. *dark\_images* folder
  3. *sun\_positions* folder, containing a folder with .csv containing the positions of the sun (and moon!) with format: timestamp, object, pos\_x, pos\_y, sunshader\_angle, row each 6 minutes for sun and every 1 minute for moon.
  4. Additional pkl's and json's and other (important!) files.
1. Run `python camera_client.py`
  2. You should see a list of all real & virtual cameras.

## CHAPTER 6

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`