GLAS/LENSS Documentation

Login

All login credentials can be found on the ‘Logins’ Google Sheet.

Pi Image Note

We’ve moved to pre-made images for easier deployment. These links are intended for DEVELOPMENT ONLY. For setting up an SD card for deployment, follow the steps under ‘Pi Setup’ below.

- Updating an image with new software changes: https://docs.google.com/document/d/1s2-vkMSrkwbgb5MAWqxE9PrE1OzE30uQUt04AbRfaJQ/edit?usp=sharing
- Changes made to base Raspbian: https://docs.google.com/document/d/18rz8onVZbwCtVoKzKpbourAjxFcIXoXYdlEF-KKVyk0/edit?usp=sharing

Pi Setup

OS Install

- Grab a copy of the latest LENSS_Pi.img from LENSS/Engineering/Pi Images
- Get a microSD card, insert it into the computer, and format to Fat32.
- Following this guide, download and install the Raspberry Pi Imager.
- With Raspberry Pi Imager open, click “CHOOSE OS” and select “Use Custom” at the bottom of the list.
- Select the LENSS_Pi_x.x.x.img from your computer.
- Select the SD card under “CHOOSE STORAGE”.
- Click “WRITE” to write the image to the SD Card.

Post Install

Updates

- For actual deployments, do not update any software. When developing a new image, then you can update using `sudo apt update` and `sudo apt upgrade`. The reason is that if there are any updates that break compatibility, we'd rather have that figured out well before deployment.

SSH

- SSH is enabled, (note that keys are required!).
    - Look below at "If the file doesn't exist"
- If you wish to ssh:
    - From the Linux/Unix computer you will be using run 'cat ~/.ssh/id_rsa.pub' in a terminal to see if a public key has already been generated.
    - If the file doesn't exist:
        - Run 'ssh-keygen' and press [Enter] through all prompts.
    - Insert the SD card just created into your computer, and locate 'rootfs/home/pi/.ssh/authorized_keys'
    - Add the contents of ~/.ssh/id_rsa.pub (from your system) to the end of 'authorized_keys'' on the Pi SD card.
        - You can use any text editor
    - Save the file and unmount the SD card, inserting it into the Pi.
    - Start up the Pi, locate the IP, and try SSH.
        - If you cannot ssh into the pi, run ssh-keygen -f "/home/user/.ssh/known_hosts" -R "(pi ip address)"
- Easy adding keys (assuming you already have remote access)
    - `ssh-copy-id -f -i /path/to/persons_id_rsa.pub pi@<ip_address>`
- Potential Error:
    - `@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@`
    - `@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @`
    - `@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@`
    - `IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!`
    - If this happens you simply have to remove a previous instance of the ip address you're trying to use from the known_hosts in your .ssh folder (for mac, unsure about linux/windows)

Change default password

- By default, the user password is already set, but if you wish to change it from the default GLAS password, follow the instructions below.
- Run `sudo passwd` in a terminal and type the Pi password from the passwords Google Sheet. This is in the LENSS > Logistics but it is restricted you have to be added specifically. At this link https://docs.google.com/spreadsheets/d/15rKK04Ph-vCaVnnaP9LhBEcxGZmHeLjSlReYQii7XKA/edit#gid=0

Configure hostname

- Run `sudo raspi-config` in a terminal.
- Navigate to 'System Options' → 'Hostname' and agree to the prompt.
- Set Hostname to "Sensor####" with the #s being the unique 4-digit ID number of the newest sensor (record the name in the lenss sensor registry in Logistics)

  *sensor name is the same as the host name

Verify directory structure

- Under `/home/pi`, there should be the following folders:
  - SensorData
  - SensorDataArchive
  - Programs
  - logging
- In the `/home/pi/Programs` folder, there should be all necessary scripts and programs for data collection and upload.

Run configuration scripts

- After verifying the directory structure and setting the hostname, run `/home/pi/Programs/config.sh`
  - This will update serverconfig.ini to reflect the sensor number.

Enable set-ard-port.service and the lenss-collect.service

- The images contain an `set-ard-port.service` and `lenss-collect.service` for autorunning `ARDread.py` on startup. Enable them by running `sudo systemctl enable lenss-collect` and `sudo systemctl enable set-ard-port`.
- `set-ard-port.service` runs first on startup, and makes sure the port is set to the location of the Arduino.
- `lenss-collect.service` depends on `set-ard-port.service` and will not run until `set-ard-port.service` exits successfully.

    - NOTE: Due to a bug in Systemd, this actually does not work. The system runs lenss-collect regardless of whether set-ard-port fails or not. lenss-collect still fails when needed, though, because when an Arduino is not plugged in, ARDread.py cannot find /dev/ttyACM0 and throws an error terminating the script. This isn't great, but it works for now.
    - https://github.com/systemd/systemd/issues/11338

Verify crontab

- Run `crontab -l` and verify the following lines are there:
    - `30 01 * * * /home/pi/Programs/DataUpload.sh`
        - The five numbers/asterisks list the time to execute the command
        - The first parameter is the location for the DataUpload.sh script
    - `00 12 * * * /sbin/shutdown -r now`

Rclone

Unix command line tool to copy files to and from a google drive. Can also list content of files in the drive.

NOTE: We should really move to lenss@glaseducation.org instead of the starsatyerkes.net address.

Test by running `rclone ls GLAS:`

***** This is already set up on the base image *****
NOTE: Easy version of below. Copy the rclone.conf file from LENSS/Engineering/Pi Images/Current Pi Files/.config/rclone/ to ~/.config/rclone/rclone.conf. Replace any existing file there. Then copy lenss-data-upload.json from LENSS/Engineering/Pi Images/System/ to ~/Programs/lenss-data-upload.json on the Pi. Run 'rcrlone ls GLAS:' to verify things are working.

If for some reason you need to set a new remote, or edit the existing, the list below are the steps originally followed. To change the ID and Secret, you can edit (e) the GLAS remote and press [ENTER] until you reach the ID and Secret section, then use the defaults after.

- Follow the 'Script installation' at https://rclone.org/install/
- Read the Drive documentation at https://rclone.org/drive/
- Run `rclone config`
    - Make new remote (option -n)
    - Pick name and place it in scripts using Rclone. 'GLAS' is default for DataUpload.sh
    - Select Google Drive / drive (option 18) (not Google Cloud Storage)

    - Client ID: `294679892056-sv34dbse82a9pitpk17dbhoun9a4g0l6.apps.googleusercontent.com`
    - Client Secret: `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`
    - Scope: 1 (drive)
    - Root folder ID for LENSS > Data > Sensors Data : (paste only the key) `1qTskg1BMyn4uPyRk_mSlxJpSiM08p_93`
    - Service Account File: `/home/pi/Programs/yerkesstardataac-1611875130515-facbdd8aa5a0.json`
    - Say no to edit advanced config.
    - Say no to auto config.
    - Say no to configure as team drive.
    - Use defaults for any other prompts.
    - Check over the provided info at last prompt, then choose ‘yes’ if everything matches. If not, choose ‘edit’ and add the correct information.
- Currently installed on the raspberry pi we’re using for data collection
- To manually upload data use: bash /home/pi/Programs/DataUpload.sh /home/pi/Sensordata/ /home/pi/SensorDataArchive/ GLAS

Restart Pi

- Run `sudo shutdown -h now` and install Pi and Arduino into the sensor box for deployment.

Data Collection for Pi w/o WiFi

- For Pi’s without a network connection, an easy way to retrieve the data is by removing the microSD card from the Pi, putting it in a computer, and running a local rsync command. You could also drag and drop the folder, then say ‘no’ to copying duplicates.
- In the following command:
    - -a: archive mode. Recursive copy and preservation of various *NIX metadata.
    - -v: verbose mode. Display to standard output what is happening.
    - -z: enable compression. Reduces copy time.

    `rsync -avz <path_to_SD>/home/pi/SensorDataArchive <local_path_for_sensor_data>`

Servers @ Glas office

Han Server (PI 3)

- Currently lives at local IP address 192.168.7.177 - DHCP Reserved
- External ssh with ssh -p 425 pi@glasoffice.ddns.net
    - Set up in router to forward port 425 to its port 22 (SSH)
    - Use IP address to ssh from the GLAS office
- Login is pi and yerkes star
- Setup: Raspberry Pi 3 Model B, 32GB SD card, running Raspbian
- Web server: Apache is installed - files on /var/www/html
    - Set up in router to forward port 8080 to han server port 80 (HTTP)
    - For external access go to http://glasoffice.ddns.net:8080
    - Make your own folder on /var/www/html/users/yourusername so you will be able to see under http://glasoffice.ddns.net:8080/users/yourusername
- Please:
    - Put your stuff into the /home/pi/users/yourname folder (use mkdir “yourfolder” if it doesn’t exist)
    - For files you need for a short time, use the /home/pi/temp folder
- Programs that run over X11: xterm (terminal), thunar (file manager), lynx (text only web server)

LENSS Host (PI 4)

- Raspberry Pi 4B Rev 1.4, 4 core, 8GB Ram, 256GB SSD
- Server name is `lensshost` at IP 192.168.7.175
- Users: pi user has been deactivated, use individual login name
    - bb8: Marc Berthoud
    - obiwan: Adam McCulloch
    - masterjoeda: Joe Murphy
    - harrypotter: Chris Kirby?
    - Iroh: ?????????
    - kyloren: Griffin Moon
    - levi: Jaidyn Catherall
    - kirara: Charlie Carvajal
    - wanda: Syd Krause
    - rogerroger: Amanda
- SSH: it’s at port 426, so `ssh -p 426 <yourusername>@glasoffice.ddns.net`
    - If you do not have a user reach out to Adam
    - Set up in router to forward port 426 to its port 22 (SSH)

    - Use `ssh <yourusername>@192.168.7.175` to ssh from the GLAS office
- Webserver / apache setup:
    - Externally: https://glasoffice.ddns.net
        - You might get a warning message from your browser, check the certificate, the SHA-1 fingerprint is A2:0A:FF:D5:47:1E:EE:8F:A4:40:D3:D0:69:63:55:98:40:EE:4F:7B
        - Port forwarding for port 443 has been set up in the router.
    - Internally: https://192.168.7.175 - DHCP Reserved
    - Installed apache2 and openssl. Apache certificates in /etc/ssl/private and /etc/ssl/certs. Also edited default configuration to forward http to https
- Pipeline:
    - Warning: Python 2 and 3 are installed, please ensure you are using python3
    - Installing packages: darepype, drizzle, astropy, lmfit, ccdproc, PIL, sep, astroquery, lmfit, pandas (using pip3)
    - Pipeline installed under /opt/SEOpipeline/pipeline
    - Delta config file under /opt/SEOpipeline/configlocal/dconf_lensshost.txt
    - Image data under /opt/SEOpipeline/images
- Installing Astrometry: installed with apt-get install for package name “astrometry.net”
    - WHERE ITS INSTALLED:
    - WHERE .CFG file is: /etc/astrometry.cfg
    - WHERE INDEX FILES ARE: /usr/share/astrometry/data
- Linux:
    - For copying files from local to remote:
        - Use command: `scp -r <folder name> <username>@192.168.7.175:/opt/SEOpipeline/images/<name_of_new_folder_created>`
    - For copying files from remote to local
    - And more commands for copying files
    - **Note:** Make sure you use a screen if you are copying a lot of files
    - Using a Linux screen:
        - Commands
    -

Watchdog script (***Watchdog is for GLAS server only*** NOT on the Pi zero)

This script sends an email to registered accounts to state which data files from the previous day were successfully uploaded to the LENSS drive. The format to call the command is ‘python3 [watchdog.py address]’, making sure to have the credentials file, ‘key.json’, in the same directory as the script. For the current command it is:
`python3 /home/pi/Documents/watchdog.py`
To run it on a daily basis, put the above command into a crontab command after specifying the time to run it. For example:

```
0 11 * * * python3 /home/pi/Documents/watchdog.py
```

Local Data Mirror

A local data mirror was created to allow easier graph generation. A folder, ‘Sensor_Data_Mirror’, was created to contain the files. The Rclone command is ‘rclone copy GLAS: /home/pi/Sensor_Data_Mirror’ to update the existing file set. A file, ‘lenss-data-download.json’, contains the login credentials and must be located in ‘/home/pi/Documents/’. Rclone needs to be reconfigured if the location of this file changes. The command is run once a day using the following line in the crontab:

```
0 9 * * * rclone copy GLAS: /home/pi/Sensor_Data_Mirror
```

Optiplex (Dell Optiplex 3040)

New “temporary” donated home for hosting Streamlit and other programs that better run on x86.
Running Ubuntu Server 22.04
IP: 192.168.7.102 (DHCP reserved)
Login:

Username: glas
Pasword: <standard_glas_password>

There’s free reign over the ‘/opt/GLAS’ folder (and obviously the ‘/home/glas’ directory), but contact Alex Scerba’s GLAS email or Slack if other file modification/software installation is needed.

Local Data Mirror

A local data mirror was created to allow easier graph generation. A folder, ‘data’, was created to contain the files under ‘/opt/GLAS/streamlit’. The Rclone command is ‘rclone copy GLAS: /opt/GLAS/streamlit/data’ to update the existing file set. A file, ‘lenss-data-download.json’ under ‘/opt/GLAS/streamlit’, contains the login credentials and must be located there. Rclone needs to be reconfigured if the location of this file changes. The command is run once a day using the following line in the crontab:

```
0 9 * * * rclone copy GLAS: /opt/GLAS/streamlit/data
```

Streamlit (Local Plotting)

We are giving Streamlit a try for our data plotting needs. A script, Plotting_Site.py, was written, and it’s located under `/opt/GLAS/streamlit`. A system service, streamlitPlot.service, was created to auto-run the script on boot.
Start and stop service with:

```
sudo systemctl [start/stop] streamlitPlot
```

Access at: http://192.168.7.102:8501

Bumblebee

- No longer in service.
- https://sites.google.com/a/starsatyerkes.net/yerkesprojects/programs/lenss/lens-central-server-bumblebee

Network Setup

Last updated: 8/11/2022

IP Ranges

192.168.7.0 (Reserved)
192.168.7.1 (Router)
192.168.7.2 - 192.168.7.99 (Unused Range)
192.168.7.100 - 192.168.7.200 (DHCP Range)
Reservations:

- 192.168.7.102 → Optiplex 3040
- 192.168.7.148 → Printer
- 192.168.7.175 → LENSS Host
- 192.168.7.177 → Han Server

192.168.7.255 = Broadcast (Reserved)

Forwarded Ports

IP : Internal Port : External Port
192.168.7.102 : 22 : 114 (TCP)
: 25565 : 25565 (TCP/UDP)
192.168.7.175 : 22 : 426 (TCP)
: 443 : 443 (TCP/UDP)
192.168.7.177 : 22 : 425 (TCP)
: 80 : 8080 (TCP/UDP)

NoIP Information

About

A free alternative to registering a domain. Choose a custom prefix and a

predetermined suffix, and point it to a public IP. Configure dynamic DNS so the URL points to the updated IP when the ISP updates the public DHCP lease. **Every 30 days confirm the hostname through the provided email.**

Current Domains

glasoffice.ddns.net
2 free domains remaining

Data Information

Format

Data Format from Inside the files: UTC Time, Local Time, DS18B20 Temperature (Celsius), TSL237 Frequency, Photoresistor Voltage, Arduino ID #

- For arduino it's the same numbers but without the time at the beginning

SQMLU File Format (ref. SQM-LU_Users_manual.pdf p.49-50):
The full manual is at LENSS/Engineering/System Overview/SQM LU Software/UNIHEDRON/SQM-LU/SQM-LU_Users_manual.pdf.
HH:MM:SS,r, NN.NNm,FFFHz,NNNNNNNNc, SSS.SSSSs, TTT.TC

- Where HH:MM:SS is the current time
- r indicates that a reading is being returned
- NN.NNm is the magnitude per arcsecond squared
- FFFHz is the frequency of light counts hitting the SQM sensor
- NNNNNNc is counts of the sensor, where counts occur at a rate of 460.8 kHz
- SSS.SSSSs - is the period of the sensor with millisecond resolution
- TTT.TC Temperature in celsius (leading space is positive, leading - is negative)

Sample Data: 00:00:11,r, 18.98m,0000000002Hz,0000195848c,0000000.425s, 013.5C
For more information look at the Manual which Adam will upload to the google drive.

Arduino File Format:
HH:MM:SS,V.VV,F.FF,CC.C,ID

- Where HH:MM:SS is the current time
- V.VV is photoresistor voltage (from a CdS sensor with 100kΩ series resistor)
- F.FF is TSL237 frequency
- CC.C is the temperature in Celcius
- ID is a number specific to each sensor

Sample Data: 00:00:01,0.08,13,16.9,LENSS_TSL_0007

Locations

All files are on the Google drive under /LENSS/Data

- SQM data is under
  /LENSS/Data/Sensors Data/YYYY-MMmmmm/sqmdata_YYYY-MM-DD.txt
- Arduino data is under
  /LENSS/Data/Sensors Data/YYYY-MMmmmm/ardata_YYYY-MM-DD.txt
- Plots are under
  /LENSS/Data/Data Analysis/Plots_YYYY-MMmmmm

Pi Additional Details

Pi Login

Pi ip address: SEE ABOVE
Password found on computers

Working on the Pi

Downloading files to computer:
Done on standard terminal; do not connect to pi outside these commands

rsync -avz -e ssh pi@<ip_address>:/home/pi/Documents/LENSS/sensorsdata .
(**do not forget the dot**) (replace <ip_address> with the ip of the Pi)

Copy files after June 1st 2020
mkdir 2020-05
cp *2020-05-* 2020-05

Copy files before June 1st 2020
for i in $(echo 0 1 2 3 4 5 6 7 8 9) ; do mkdir 2020-05-0$i ; done #alter values corresponding to date

for i in $(echo 0 1 2 3 4 5 6 7 8 9) ; do cp *2020-05-0$i* 2020-04-0$i ; done #double check file and folder dates match to avoid trouble

Deleting incorrectly-made folders:

for i in $(echo 0 1 2 3 4 5 6 7 8 9) ; do rm 2020-04-0$i ; done

Unix Commands

Removing a directory with files inside on linux:
rm -rv <folder>

Screen commands:

Listing screens: screen -ls
Joining screens: screen -r <session id (found in output of screen -ls)>
Creating screens: screen -S <new session name>

Screen keyboard shortcuts:

Leaving screen: ctrl+a, ctrl+d
Kill screen: ctrl+a, k
Split screen: ctrl+a |
Start another terminal: ctrl+a c

More information on screen see the GNU screen manual at:
https://www.gnu.org/software/screen/manual/screen.html

Tutorials for Editors:
Vim: https://www.openvim.com/
Vi: https://staff.washington.edu/rells/R110/#basics7

AWK Language
Tutorial: https://www.tutorialspoint.com/awk/awk_overview.htm
Cheat Sheet:

- Formatting: awk 'BEGIN{header commands} {body commands} END{footer commands}' file
  - Semicolons separate commands
  - Command files format: awk -f command_file file
    - Command file lists the same commands normally put in quotes
- Options:
  - -f: use command file
  - -v: use a variable; awk -v var=VARIABLE …
  - --profile (-p): creates a nice-looking file showing what was executed in an awk command
- Isolating:
  - {print $n} prints the nth item in each line of a file (when separated by whitespace unless specified); n must be a number
    - awk -F "," changes this separator from whitespace to commas, and the syntax is the same for other separators
    - $0 represents the line as a whole
  - Print is the default executed when a body block is not specified in the command

    - Using awk '/PATTERN/ { }' [file] will only execute body when the line contains "PATTERN" somewhere (case sensitive)
        - awk '/PATTERN/{cnt++} END{print "Count = ", cnt' [file], for example, will print the number of lines where PATTERN appears
    - length($n) will return the length of the nth item
    - If the body is a boolean expression, then awk will only return lines where it is true
    - '$n ~ x' (match) functions similarly to pattern; the format provided finds if x is in the nth field
        - !~ is the logical opposite of match
- Built-in variables:
    - ARGC is the number of arguments in a command, and ARGV is the array of these arguments (just like in python sys library)
    - ENVIRON is the array of environment variables
    - FILENAME is self-explanatory
    - NF is number of fields per line
    - NR is the line (record) number
    - IGNORECASE is the case sensitivity of an awk expression
        - IGNORECASE = 1 makes it case insensitive
        - IGNORECASE = 0 (default) makes it case sensitive
- Patterns:
    - Dot operator (ex f.n) lets awk fill in blanks with a single character
    - Match character (f[aiu]n) limits scope of what can fill in the blank to what is specified
        - Using a ^ (ex f[^aiu]n) limits scope to all letters <u>except</u> those specified
    - ^ is an indicator of line start in patterns (ex /^The/)
    - $ is a line ending indicator for patterns (ex /ed$/)
    - | is a logical or within patterns (ex /Call|Ball/)
    - ? will match if there is one or no instances of the preceding character (/Colou?r/ matches Colour or Color)
        - * will match if there are many or no instances (Colouur can match as well from /Colou*r)
        - + will match if there is at least one instance (Color will not match from /Colou+r/)
    - Parenthesis groups alternatives for or operation (ex /Apple (Pie|Juice)/)
- Misc:
    - Strings automatically merge if only separated by space (ex str3 = str1 str2)
    - Arrays can have string indices (similar to dictionaries) and can only be one-dimensional
    - If statements and loops only execute the following command if the conditions are met; {} can be used to store multiple commands
    - rand() returns random numbers and srand(seed) uses seeds
    - String sorting functions alphabetize
    - Functions: function func_name(args) {body}

Ternary Operator:
(logical condition) ? statement1 : statement2

- Returns statement1 if the condition is true
- Returns statement2 if the condition is false

Python Notes

Splitting text:
<text>.split("<separator>")

now.strftime formatting:
https://devhints.io/strftime

Delay:
import time
time.sleep(<seconds>) #unlike arduino, this uses whole seconds. Use decimals for subseconds.

Logging library references:
https://stackoverflow.com/questions/338450/timedrotatingfilehandler-changing-file-name
https://docs.python.org/3/library/logging.handlers.html#timedrotatingfilehandler
https://medium.com/better-programming/tips-and-tricks-for-handling-logging-files-in-python-b48be3d553ad

Calculating median:
import numpy,
numpy.median(<array>)

Email via python:
https://realpython.com/python-send-email/