alpacadsc

Oct 21, 2020

Contents:

CHAPTER 1

Introduction

The Alpaca digital setting circles driver (**alpacadsc**) allows connecting client software like planetariums to ALT/AZ mounted telescopes like dobsonians with the appropriate hardware. The software supports tracking the position of the telescope as it is moved across the sky, allowing easy acquisition of sky targets.

The basic theory of operation is for the user to locate a sky target such as a bright star and center the telscope on the target. Then using the planetarium software the user send a "Sync" command to the driver with the coordinates of the target. In Cartes du Ciel, for example, the user would right click on the target and select "Sync". This tells the driver where it is currently pointing in the sky. Using the configured latitude and longitude and the current time **alpacadsc** can then compute the altitude and azimuth of the target. From this it can then compute the altitude and azimuth of the telescope from the changes in the encoder values.

This simple "1 star" synchronization works well over a small part of the sky (say 30-50 degrees) as long as the telescope is fairly level. If you find the pointing of the scope is poor as you get farther from the original sync target then simply sync on a new target closer to your desired desination target. This should improve the pointing accuracy. Do this as necessary as you move around the sky.

Before using **alpacadsc** it is necessary to configure a profile for your telescope. This an other usage details are covered in the *Usage* section.

CHAPTER 2

Usage

2.1 Installation

The Alpaca digital setting circles driver (**alpacadsc**) can be installed from source. It supports setup.py so the package can be installed using the command:

```
python3 setup.py install
```

Alternately a distribution package can be created with:

```
python3 setup.py bdist_wheel
```

The resulting package can be installed with:

```
python3 -m pip install <bdist_file>
```

where <*bdist_file*> will be the newly created package in the "dist/" folder.

Other options available are:

- Rebuilding the documentation into the directory docs/build/html.

  ```
  python3 setup.py build_sphinx
  ```

- Run several tests on the code base.

  ```
  python3 setup.py test`
  ```

2.2 Starting The Alpaca Service

You will need to start the Alpaca service which will allow software to connect with your setting circles. The command to do this on Linux is:

```
alpacadsc
```

and on Windows would be:

```
alpacadsc.exe
```

You can also start the service by invoking the module via python:

```
python -m alpacadsc.startservice
```

The service will start and by default listens to the port 8000 on the local host.

Warning: The service will run a web server on your system that will listen for incoming connections from Alpaca clients. It should only listen for connections from your local computer. At this point *alpacdsc* is still in development and such the test server built into Flask is being used. You will see a warning when you start the service that says this. The intention long term is to move off the internal Flask server.

2.2.1 Command Line Options

The service accepts several command line options:

--port port
: Sets the port that the Alpaca service will listen to for client connections. The default value is 8000.

--profile PROFILE
: Use the configuration profile **PROFILE**. If none is supplied then the last profile used will be loaded.

--listprofiles
: List all profiles which are currently defined.

--debug
: Show additional debugging information in log file.

2.2.2 Log File Output

A log file will be created in the directory from which the service was started and has a filename of the format *alpacadsc-<dateime>.log* where *datetime* is a timestamp of when the service was started. This file can be helpful when trying to track down problems or reporting an issue you may encounter.

2.3 Configuration

Before connecting to the Alpaca service you will need to configure a profile for your equipment.

Note: You cannot configure the Alpaca server if a program is currently connected to the service so be sure to disconnect all clients before attempting configuration.

The configuration page is available by connecting a browser to:

http://localhost:8000/setup/v1/telescope/0/setup

As a convenience if you connect to:

http://localhost:8000

or:

http://localhost:8000/setup

a link will be provided to get to the actual configuration page.

2.3.1 Profiles

The first step is to create a new profile for your equipment. This is done using the "Create New Profile" button. Fill in the box next to the button with the name of the new profile and click the button. If successful a new page will load confirming the new profile has been created. Use the link to return to the configuration page.

When a new profile is created the current profile used for the service will be set to the new profile. If you want to change the current profile to a previously created profile use the "Change Profile" button. A new page will load showing all the available profiles with a checkbox next to each one. Select the checkbox for the profile you want to switch to and then click the "Change Profile" button.

The current profile will automatically be loaded whenever the service is started. Optionally the *–profile* command line option can be used to specify the profile to be used. To get a list of available profiles use the *–listprofiles* command line option.

Profiles are stored as YAML formatted files. The location of the profile files depends on the platform:

Linux	$(HOME)/.config/alpacadsc
Windows	%APPDATA%/alpacadsc

If you want to backup your settings or move them to another computer you can copy the profiles stored here. The current profile name is stored in the file "current_profile.yaml".

The location configuration in the YAML file are stored in an array called "location" with the following keys:

Key	Data Type	Notes
obsname	String	Human readable name of location
longitude	Float	Longitude as decimal degrees
latitude	Float	Latitude as decimal degrees
altitude	Float	Altitude in meters

An example is:

```
location:
    obsname: Observatory
    longitude: 100.0
    latitude: 30.0
    altitude: 450.0
```

The encoder configuration in the YAML file are stored in an array called "encoders" with the following keys:

Key	Data Type	Notes
driver	String	Name of driver - currently "DaveEk" is only allowed
serial_port	String	Serial port device name
serial_speed	Integer	Serial port speed
alt_resolution	Integer	Tics per revolution for alt encoder
az_resolution	Integer	Tics per revolution for alt encoder
alt_reverse	Boolean	If true then reverse alt axes
az_reverse	Boolean	If true then reverse alt axes

An example is:

```
encoders:
  alt_resolution: 4000
  alt_reverse: false
  az_resolution: 4000
  az_reverse: false
  driver: DaveEk
  serial_port: /dev/ttyUSB1
  serial_speed: 9600
```

2.3.2 Location

The observing location needs to be set for each profile. This consists of the name of the location (a string) as well as the latitude, longitude and altitude (meters). Specify the latitude and longitude as decimal degrees and use a negative longitude for Western latitudes.

For example, if the location is latitude equal to 36d40m20s North and longitude was 30d30m10s West, first convert the sexagesimal degrees to decimal degrees yielding 36.67222 North, 30.502778 W. Since the longitude is a Western one then convert it to a negative value so you would use "36.67222" for the latitude and "-30.502778" as the longitude.

There are websites that can convert sexagesimal degrees to decimal degrees as well as many calculators have a function to perform this conversion.

Once these settings are entered use the "Save Changes" button to make them permanent. The button only saves the location settings.

Encoders

The encoders used by the digital setting circles (DSC) also need to be configured.

Currently the Alpaca service only supports DSC which use the "Dave Eks" protocol so the "Driver" should be set to "DaveEk".

The serial port should be configured to match the port the DSC is connected to - there will be some suggested ports based on the available ports on the computer.

The serial speed must match that of the DSC - 9600 is typical.

The resolution of the encoders on the altitude and azimuth axes must also be specified. Common values are 4000, 8000 or 10000. If this value is wrong then the service will not properly track the scope as it is moved.

Finally two checkboxes are available to tell the service the altitude and/or azimuth encoder outputs need to be reversed. If you move the scope one way and it moves the opposite direction in your software connected to the service then try reversing the axis.

Once these settings are entered use the "Save Changes" button to make them permanent. The button only saves the encoder settings.

2.4 Using With Planetarium Software

First start the Alpaca DSC driver service as shown in the section *Starting The Alpaca Service*.

Then use your software to connect to the service. The software must support Alpaca to work with this driver. You will want to configure the server IP as 127.0.0.1 or "localhost" and the server port as 8000.

Once connected to the Alpaca DSC driver service the driver will still need to be synchronized with the sky before it can report the position of the telescope. This is done by finding a star in your planetarium program and then manually pushing the telescope so the same star is centered in the eyepiece. Now use the "Sync" command in your program to tell the driver to sync on the current position. This will let the driver know the current telescope position and from then on the driver will report the ALT/AZ and RA/DEC values as the telescope is moved around.

For best results choose a star to synchronize on which is close to the area of the sky you will be observing. If you move to another part of the sky then you can synchronize on a new star in that region. The sync operation will override the previous one.

The synchronization with the sky is lost when the driver exits.

2.5 Debugging Encoders

There is a debugging web page generated by the driver which reports the current encoder raw counts if the driver is connected. If the driver has been synchronized with a star then it will also report the current ALT/AZ and RA/DEC position.

CHAPTER 3

Writing A New Driver

The **alpacadsc** package supports adding new drivers for digital setting circles/encoders. Currently any digital setting circles which use a serial port interface and report the raw encoders counts should be able to made to work. Digital setting circles that work in celestial coordinates (RA/DEC) will not work with the current alpacadsc implementation.

To add a new driver create a new source file with a name following the pattern "encoders_altaz_<drivername>." The package includes a reference driver for encoders supporting the "Dave Ek" protocol as well as a simulator. The "Dave Ek" driver would be a good starting point. Simply copy the driver source and then edit to change the various methods to use the protocol commands for the encoders in question and parse the return values. Also change the name of the driver in the "name()" method to be a human readable name for your new driver.

To test simply put the new source file in the **alpacadsc** package location. When alpacadsc loads it will scan for modules following the pattern given above for encoders driver plugin and detect it. It will also be given as an option on the configuration page.

CHAPTER 4

alpaca package

4.1 Submodules

4.2 alpacadsc.alpaca_controller module

4.3 alpacadsc.alpaca_models module

4.4 alpacadsc.alpaca_service module

4.5 alpacadsc.altaz_dsc_profile module

class alpacadsc.altaz_dsc_profile.**AltAzSettingCirclesProfile**(*reldir*, *name=None*)

Bases: *alpacadsc.profiles.Profile*

class Encoders(*_sectionname: str = 'encoders'*, *driver: str = None*, *serial_port: str = None*, *serial_speed: int = 9600*, *alt_resolution: int = None*, *az_resolution: int = None*, *alt_reverse: bool = False*, *az_reverse: bool = False*)

Bases: *alpacadsc.profiles.ProfileSection*

alt_resolution = None
Alt axis resolution

alt_reverse = False
Reverse ALT?

az_resolution = None
AZ axis resolution

az_reverse = False
Reverse AZ?

driver = None

serial_port = None

serial_speed = 9600

class Location(*_sectionname: str = 'location'*, *obsname: str = None*, *latitude: float = None*, *longitude: float = None*, *altitude: float = None*)
Bases: *alpacadsc.profiles.ProfileSection*

altitude = None
Altitude in meters

latitude = None
Latitude in degrees

longitude = None
Longitude in degrees

obsname = None
Name of observing location

read()
Read profile config file.

Returns (bool) Whether or not read succeeded.

4.6 alpacadsc.baseencoders module

class alpacadsc.baseencoders.**A**
Bases: *alpacadsc.baseencoders.EncodersBase*

class alpacadsc.baseencoders.**EncodersBase**
Bases: object

Base class for all encoder drivers.

connect()
The driver should connect to the digital setting circles hardware when this method is called.

disconnect()
The driver should disconnect to the digital setting circles hardware when this method is called.

get_encoder_position()
Read the encoders resolution from the digital setting circles hardware.

Returns (tuple) The position of the altitude and azimuth encoders.

get_encoder_resolution()
Read the encoders resolution from the digital setting circles hardware.

Returns (tuple) The resolution of the altitude and azimuth encoders.

name()
Returns the human readable name for this driver.

set_encoder_resolution(*res_alt*, *res_az*)
Read the encoders resolution from the digital setting circles hardware.

Parameters

- **res_alt** – Resolution (steps/rev) of altitude encoder.
- **res_alt** – Resolution (steps/rev) of azimuth encoder.

4.7 alpacadsc.encoders_altaz_daveek module

class alpacadsc.encoders_altaz_daveek.**EncodersDaveEk**(*res_alt=4000*, *res_az=4000*, *reverse_alt=False*, *reverse_az=False*)

Bases: *alpacadsc.baseencoders.EncodersBase*

Parameters

- **res_alt** (*int, optional*) – Altitude encoder resolution, defaults to 4000
- **res_az** (*int, optional*) – Azimuth encoder resolution, defaults to 4000
- **reverse_alt** (*bool, optional*) – Reverse altitude axis, defaults to False
- **reverse_az** (*bool, optional*) – Reverse azimuth axis, defaults to False

connect(*port*, *speed=9600*)

The driver should connect to the digital setting circles hardware when this method is called.

Parameters

- **port** – Serial device to which digital setting circles is connected.
- **res_alt** – Speed for serial connection.

Returns True is successful.

Return type bool

disconnect()

Disconnect.

Returns True is successful.

Return type bool

get_encoder_position()

Read the encoders resolution from the digital setting circles hardware.

Returns (ttuple) The position of the altitude and azimuth encoders.

get_encoder_resolution()

Read the encoders resolution from the digital setting circles hardware.

Returns (tuple) The resolution of the altitude and azimuth encoders.

name()

Returns the human readable name for this driver.

set_encoder_resolution(*res_alt*, *res_az*)

Read the encoders resolution from the digital setting circles hardware.

Parameters

- **res_alt** – Resolution (steps/rev) of altitude encoder.
- **res_alt** – Resolution (steps/rev) of azimuth encoder.

4.8 alpacadsc.encoders_altaz_simulator module

class alpacadsc.encoders_altaz_simulator.**EncodersAltAzSimulator**(*res_alt=4000*, *res_az=4000*, ***, *reverse_alt=False*, *reverse_az=False*)

Bases: *alpacadsc.baseencoders.EncodersBase*

Parameters

- **res_alt** (*int, optional*) – Altitude encoder resolution, defaults to 4000
- **res_az** (*int, optional*) – Azimuth encoder resolution, defaults to 4000
- **reverse_alt** (*bool, optional*) – Reverse altitude axis, defaults to False
- **reverse_az** (*bool, optional*) – Reverse azimuth axis, defaults to False

connect(*port*, *speed=9600*)

The driver should connect to the digital setting circles hardware when this method is called.

Note: port and speed ignored in this simulator driver.

Parameters

- **port** – Serial device to which digital setting circles is connected.
- **res_alt** – Speed for serial connection.

Returns (bool) True is successful.

disconnect()

The driver should disconnect to the digital setting circles hardware when this method is called.

get_encoder_position()

Read the encoders resolution from the digital setting circles hardware.

Returns (ttuple) The position of the altitude and azimuth encoders.

get_encoder_resolution()

Read the encoders resolution from the digital setting circles hardware.

Returns (ttuple) The resolution of the altitude and azimuth encoders.

name()

Returns the human readable name for this driver.

set_encoder_resolution(*res_alt*, *res_az*)

Read the encoders resolution from the digital setting circles hardware.

Parameters

- **res_alt** – Resolution (steps/rev) of altitude encoder.
- **res_alt** – Resolution (steps/rev) of azimuth encoder.

4.9 alpacadsc.profiles module

class alpacadsc.profiles.**Profile**(*reldir*, *name=None*)

Bases: object

Stores program settings which can be saved persistently. Supports ProfileSection's which allow a hierarchical namespace for parameters.

Set some defaults for program settings

Parameters

- **reldir** (*str*) – location relative to top of default config location If None then will be relative to current working directory.
- **name** (*str*) – name of profile config file

Note: reldir = "hfdfocus/" and name = "C8F7.yaml" would create a file <configbasedir>/hfdfocus/C8F7.yaml

add_section(*sectionclass*)
Add a section to Profile.

Parameters **sectionclass** (ProfileSection) – Section to be added.

filename()
Return profile config filename.

Returns (Path) Profile filename

read()
Read profile config file.

Returns (bool) Whether or not read succeeded.

write()
Write profile config file.

Returns (bool) Whether or not write succeeded.

class alpacadsc.profiles.**ProfileSection**
Bases: object

A ProfileSection is a subtree member of a Profile and contains its own set of key/value pairs. Multiple ProfileSection's can be added to a Profile to give parameters different namespaces in the Profile.'

get(*key*, *default=None*)
Retrieve parameter from ProfileSection by key name. Default value used if key not found in ProfileSection.

Parameters

- **key** (*str*) – Name of parameter to retrieve
- **default** – Optional default value if key not present

Returns Parameter value or default value if not present.

alpacadsc.profiles.**find_profiles**(*loc*)
Return list of existing profiles in given location loc. The location loc is relative to the base path for config files for the given platform.

Parameters **loc** (*str*) – Directory relative to base config path to search for profiles

Note Assumes profile configuration files end with '.yaml'

Returns (List[str]) List of profiles found or [] if none available.

alpacadsc.profiles.**get_base_config_dir**()
Find base path for where to store config files depending on platform.

Returns (Path) Root path of where config files are stored

Raises **FileNotFoundError** – If base path cannot be determined.

alpacadsc.profiles.**get_current_profile**(*loc*)
Read current_profile.yaml file to get name of current profile.

Parameters **loc** (*str*) – Directory relative to base config path to search for profiles

Returns (str) Name of currently active profile or None if none defined.

alpacadsc.profiles.**set_current_profile**(*loc*, *current_profile_name*)
Write current_profile.yaml file with name of current profile.

Parameters

- **loc** (*str*) – Directory relative to base config path to search for profiles
- **current_profile_name** (*str*) – Name of current active profile - do NOT include a '.yaml' extension on the profile name.

Returns (bool) Whether operation was successful or not

4.10 alpacadsc.setup_controller module

4.11 alpacadsc.startservice module

4.12 Module contents

CHAPTER 5

Design Specification

Author: Michael Fulbright

Contact: mike.fulbright@pobox.com

Status: Initial draft

Date: 2020-09-04

5.1 Introduction

5.1.1 Purpose

The main purpose of this document is to outline the requirements of the Alpaca Digital Setting Circles Driver (hereafter "DSC driver"). This specification will cover the user experience as well as address some details of the technical implementation.

The digital setting circles (DSC) is a device which interfaces which each movable axis of a telescope and tracking change sin position. The most common use is on a dobsonian telescope which has altitude (ALT) and azimuth (AZ) axes. The ALT axis moves from the horizon to straight overhead (the zenith), while the AZ axis corresponds to the distance from North around the horizon. These together allow the specification of any point in the sky.

By reading the changes in the ALT and AZ position of the telescope a program can track the telescope and determine where it is pointing in the night sky. A planetarium program can be used which also has a database of sky objects and so the telescope position can be plotted against the known objects. This allows user ("observer") to find objects by moving the telescope until it is at the desired object.

5.1.2 Scope

The DSC Driver will allow users to:

- connect to a DSC using an application which supports Alpaca
- edit the configuration of the DSC (serial port, encoder resolution, etc)

- edit the geographical location where the observer is location
- synchronize the telescope location via a planetarium program
- once synchronized the user can get locate objects using the planetarium program

5.1.3 Technical Overview

Supported Platforms

The DSC driver will support Windows and Linux targets initially. In theory the driver should work on any platform which supports Python 3.

Communication Interfaces

The DSC driver will listen on a TCP port for REST API requests which are how an Alpaca driver communicates with clients.

The DSC driver will also communicate with the DSC device. This is normally a serial interface such as RS232 or a USB<->RS232 adapter. Wireless operation using a serial stream via Bluetooth or WiFi is also possible.

Assumptions and Dependencies

The DSC driver uses Python 3.7 and depends on <insert python web framework> web framework to implement the REST API server needed for Alpaca. The *pyserial* module is used for communicating with serial devices.

Currently the driver assumes an ALT/AZ arrangement for the telescope. It would be possible to support a RA/DEC arrangement (like a German Equatorial Mount (GEM)), but this is beyond the scope of the current implementation.

5.1.4 Requirements

Functions

Alpaca Telescope Driver

The Alpaca driver listens on a TCP socket for REST API requests from clients. There are several classes of devices supported by Alpaca such as cameras, telescope, filter wheels, etc. The DSC Driver is a telescope device. The entire API for a telescope device is not implemented however, as this is not required for a DSC device.

The entire Alpaca API for a telescope device is NOT implemented, but only those sufficient to get a planetarium program working. The program *Cartes du Ciel* was used for testing - other programs may require additional API components to be implemented.

The following Alpaca API interfaces are implemented:

Name	GET	SET	Notes
alignmentmode	YES	NO	Returns ALPACA_ALIGNMENT_ALTAZ
altitude	YES	NO	Need to implement SET
aperturearea	YES	NO	Need to implement SET
aperturediameter	YES	NO	Need to implement SET
athome	YES	NO	Returns FALSE
atpark	YES	NO	Retursn FALSE

Continued on next page

Table 1 – continued from previous page

Name	GET	SET	Notes
axisrates	YES	NO	Returns empty list
azimuth	YES	NO	
canfindhome	YES	NO	Returns FALSE
canmoveaxis	YES	NO	Returns FALSE
canpark	YES	NO	Returns FALSE
canpulseguide	YES	NO	Returns FALSE
cansetdeclinationrate	YES	NO	Returns FALSE
cansetguiderates	YES	NO	Returns FALSE
cansetpark	YES	NO	Returns FALSE
cansetpierside	YES	NO	Returns FALSE
cansetrightascensionrate	YES	NO	Returns FALSE
cansettracking	YES	NO	Returns FALSE
canslew	YES	NO	Returns FALSE
canslewaltaz	YES	NO	Returns FALSE
canslewaltazasync	YES	NO	Returns FALSE
canslewasync	YES	NO	Returns FALSE
cansync	YES	NO	Returns TRUE
cansyncaltaz	YES	NO	Returns FALSE
connected	YES	NO	
declination	YES	NO	
description	YES	NO	
destinationsideofpier			
doesrefraction	YES	NO	Returns FALSE
driverinfo	YES	NO	
equatorialsystem	YES	NO	Returns FALSE
focallength	YES	YES	
interfaceversion	YES	NO	
ispulseguiding	YES	NO	Returns FALSE
rightascension	YES	NO	
sideofpier	YES	NO	Returns 0
sideraltime	YES	NO	
siteelavation	YES	NO	Need to implement SET
sitelatitudev	YES	NO	Need to implement SET
sitelongitude	YES	NO	Need to implement SET
slewing	YES	NO	Returns FALSE
supportedactions	YES	NO	
synctocoordinates	NO	YES	
targetdeclination	YES	NO	
targetrightascension	YES	NO	
tracking	YES	NO	
trackingrate	YES	NO	
trackingrates	YES	NO	
utcdate	YES	NO	

Interface to DSC Encoders

The DSC driver also maintains communication with the encoders of the DSC device. This gives the position of the ALT and AZ axes of the telescope. The DSC is polled at regular intervals for its current position and the driver then recomputes the sky position that the telescope is currently pointed. This computation depends upon the user first

performing a *synchronize* (or *sync*) operation which involves pointing the telescope at a known star or sky objects and telling the planetarium program to synchronize the mount. The DSC driver uses the raw ALT/AZ encoder positions and the RA/DEC coordinates of the target chosen in the sky for syncing and computes a mapping from raw encoder position to sky position.

Encoder Synchronization

The DSC encoders report back the change in the position of each axis since the DSC was powered on. The changes are relative to the original position. In order to map these values to the position of the telescope in the sky the user must synchronize the DSC encoders. The process is as follows:

- User points the telescope to a star or sky object in the planetarium catalog.
- Once the object is centered in the field of view (FOV) of the telescope the planetarium program is told to "sync" the position of the telescope.
- The DSC driver receives the sync request and records the raw DSC encoder values.
- Using the encoder resolution for each axis the raw encoder values are converted to degrees.
- Using the geographic location of the observer site and the current time the current position of the object in the sky (ALT/AZ) is computed.
- A linear mapping between the raw encoder values and the actual ALT/AZ position is computed and applied to future value read from the DSC device.
- It is assumed the dobsonian base is level for this simple synchronization to work.

This simple mapping works well within the vicinity of the object chosen for synchronization but will become more inaccurate as the observing position is farther and farther from the synchronization position. The easy remedy is to chose a new synchronization point when moving to a new area of the sky.

Observing Profile

A profile is stored for each observing configuration which contains the following information:

- **location**
  - location name
  - latitude (decimal degrees)
  - longitude (decimal degrees)
  - altitude (meters)
- **DSC configuration**
  - serial port for DSC
  - communication speed
  - ALT/AZ encoder resolution
  - Whether ALT and/or AZ direction is reversed
- **equipment information**
  - aperture of telescope
  - focal length of telescope

This profile is stored as a YACC file under a system configuration directory which depends on the system platform. For Linux is it stored in the ".config/AlpacaDSCDriver" directory in the user's home directory. In Windows it is stored in the directory "%APPDATA"/AlpacaDSCDriver".

Web dashboard

The DSC driver also has a built in web server which is used to monitor the current status of the DSC driver as well as config the driver. The status page displays the following information:

- current raw encoder counts
- current sky position as ALT/AZ (if synchronized)
- current sky position as RA/DEC (if synchronized)
- whether the mount is tracking (for dobsonians on an equatorial platform - not currently implemented)
- current observational profile

A button exists that will lead to an alternate web page allows configuring the observing profile mentioned in the previous section "Observing Profile". The user can also create a new profile, load an existing profile, or save the current profile under a new name.

CHAPTER 6

Tests

6.1 Test cases

The test cases can be executed using the command:

```
python setup.py test
```

6.1.1 test_server_basic

Tests various HTTP endpoints served by the driver work properl including pages supporting the configuration of the driver.

6.1.2 test_server_alpaca

Tests basic Alpaca REST API calls.

6.1.3 test_server_pointing

Tests reading encoders value and that syncronizing the driver and moving scope to a new position tracks in ALT/AZ and RA/DEC properly.

6.2 utils module

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

Symbols

A

C

D

E

F

G

L

N

O

P

R

S

W