The INDI protocol

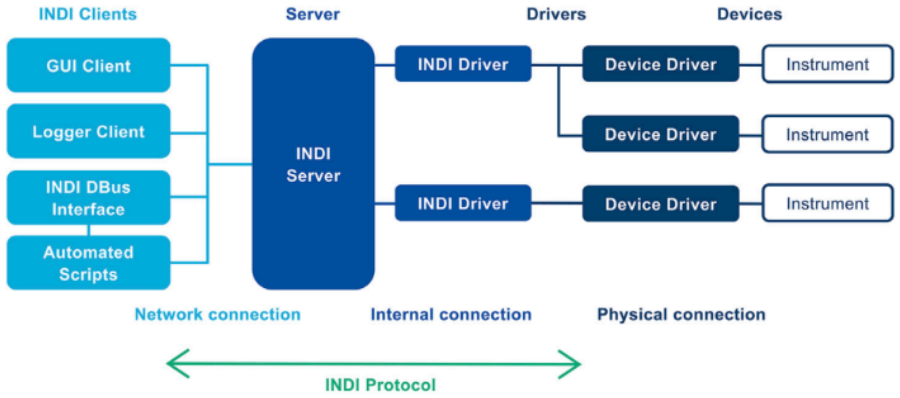
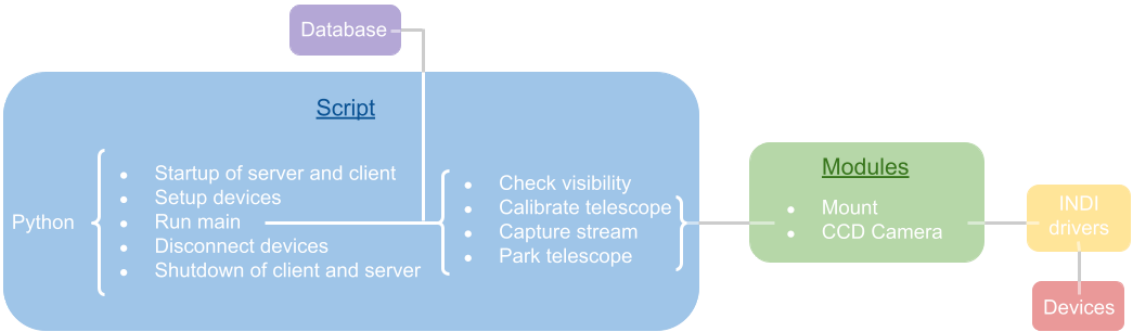


FIGURE 2
Schematic representation of the operation principle of INDI from typical clients to the different telescope instruments with the INDI Server and Drivers controlled by the INDI Protocol and their connection types.

https://www.frontiersin.org/articles/10.3389/fspas.2022.895732/full

Structure of the installation



What does the code do?

Summary:

1. initializing server and client
2. setting up devices and their properties
3. running main function
    a. calibrating telescope
        i. checking star visibility
        ii. checking telescope direction accuracy
        iii. taking control picture
    b. capturing stream
        i. setting stream parameters
        ii. recording stream
    c. parking telescope
4. disconnecting devices
5. terminating server

I/ Setting up the installation

1. initializing the server and client

This code is using the open source software INDI to control the telescope orientation and camera trigger. So before anything else, an INDI server needs to be launched and an INDI client needs to be initialized: the server will run the commands, and the client will control the devices from the server.

To initialize the server, the command 'indiserver' has to be run in a terminal, followed by the initialization of the devices that we want to control. In this case, we are using a QHY CCD camera, and a SynScan telescope, so the full command to run is as follows:

```
~ $ indiserver -v indi_synscan_telescope indi_qhy_ccd
```

To initialize the client, we need to create a subclass to the original INDI class responsible for creating clients : BaseClient. The subclass (called IndiClient) inherits the properties and methods of BaseClient (e.g. newDevice() etc). None of the methods need to be overridden in theory, but overriding the newBLOB method enables the use of Python module threading, which will prove useful to be able to continue capturing new data while processing data that has just been captured. Once the client subclass has been created, we need to create an instance of IndiClient, and link it to the server. Once the client is connected to the server, the set up of the devices can begin.

2. setting up the devices

To be able to control the doings of the telescope and the camera, they first have to be recognized by the client.

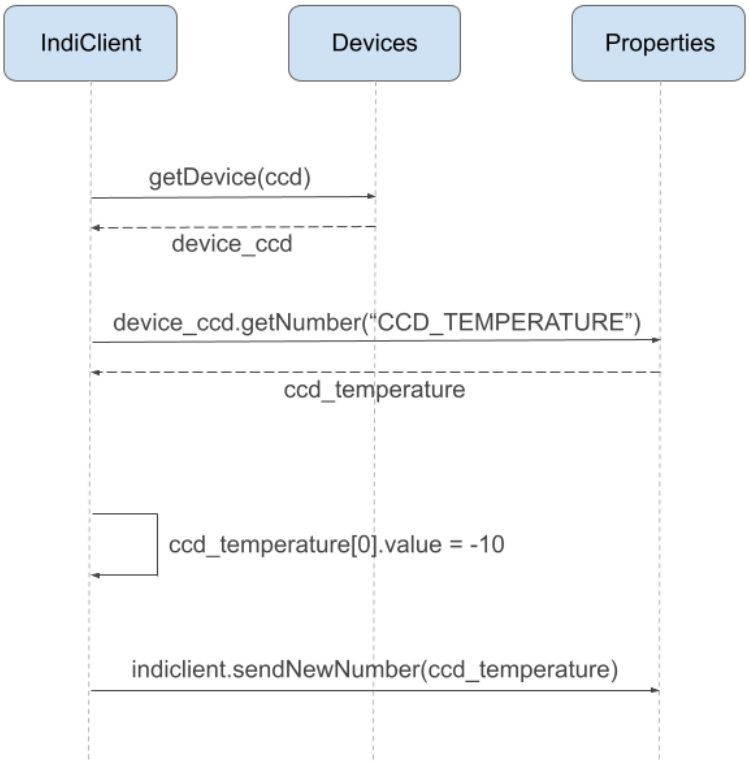
```
ccd = "QHY CCD QHY174M-7a6fbf4"
device_ccd = indiclient.getDevice(ccd)
```

Once they are, they have to be connected to the server, after which all properties will be controllable through the server with predefined commands. Here is how setting the properties works, illustrated by the example of the CCD's temperature property.

```
ccd_temperature = device_ccd.getNumber("CCD_TEMPERATURE")
while not ccd_temperature:
    time.sleep(0.5)
    ccd_temperature = device_ccd.getNumber("CCD_TEMPERATURE")
ccd_temperature[0].value = -10
indiclient.sendNewNumber(ccd_temperature)
```

For every device, there is a defined list of properties, with specific types and names (most of them can be found under Standard Properties on the Indilib website. The CCD's temperature is a type *Number* property and its name is '*CCD_TEMPERATURE*'. Other types include *Text* and *Switch* (ON/OFF). In order to change a property, it has to be accessed first with the *getType("NAME_OF_PROP")* method. For example the CCD's temperature is accessed by *getNumber("CCD_TEMPERATURE")*.

Then, once the server was able to get the property, we can set its value, text, or position depending on whether the property type is *Number*, *Text* or *Switch*. Each property is represented as an array in Python (it can be an array of 1), so in order to set values for a property, it can simply be called with indexes. The CCD temperature property is a single value, so *ccd_temperature[0].value = XX* will set the CDD temperature to *XX*. Finally, for the new set value to be taken into account, the redefined array has to be sent to the device by the client, using the *sendNewType* method. For the CCD temperature property, it would be *sendNewNumber(ccd_temp*).



II/ The main function

The main function is the function that will trigger the pictures and streams of a chosen star at a chosen time. It takes in argument a text file containing information on stars to captured and when and how to capture them. This text file changes every night. Each line of the file represents a star.

The main function is divided in three parts for each star:

- calibration of the telescope
- capture of the stream
- parking the telescope before next star

The calibration of the telescope and the capture of the stream are set to start respectively five minutes (see value of DELAY) before and at the time given in the text file. The telescope is parked once the stream recording has ended.

1. calibration of the telescope

The calibration itself is divided into three (or four) parts (see Figure):

- making sure the star is visible,
- making sure the telescope is pointing in the correct direction
- (and recalibrating it if not)
- capturing a control picture for future analysis

The first step to calibrating the telescope is making sure the star we are directing it to is in fact visible at this time of night. To do so, we are using the astroplan library, and giving it the scope's coordinates on Earth, the coordinates of the target (star), and the time.

If the star is not yet visible, the program will stop the calibration and move on to the next target in the text file.

If the star is visible, however, the program carries on with the next step: checking the accuracy of the telescope's coordinates. The coordinates of the star are given to the telescope for it to move to them. Once the telescope is locked on a direction (and tracking it), a picture is captured. This picture is then analyzed by astrometry.net, which will return (if the stars are recognizable) the actual coordinates of the portion of the sky the telescope is pointing to.

If the difference - between the coordinates given to the telescope and the actual ones returned by astrometry.net - is small enough, the telescope is considered already calibrating.

If not, the telescope is then synced to the coordinates returned by astrometry.net

In any case, the program then moves on to capturing a picture of the sky, which will serve as a control image.

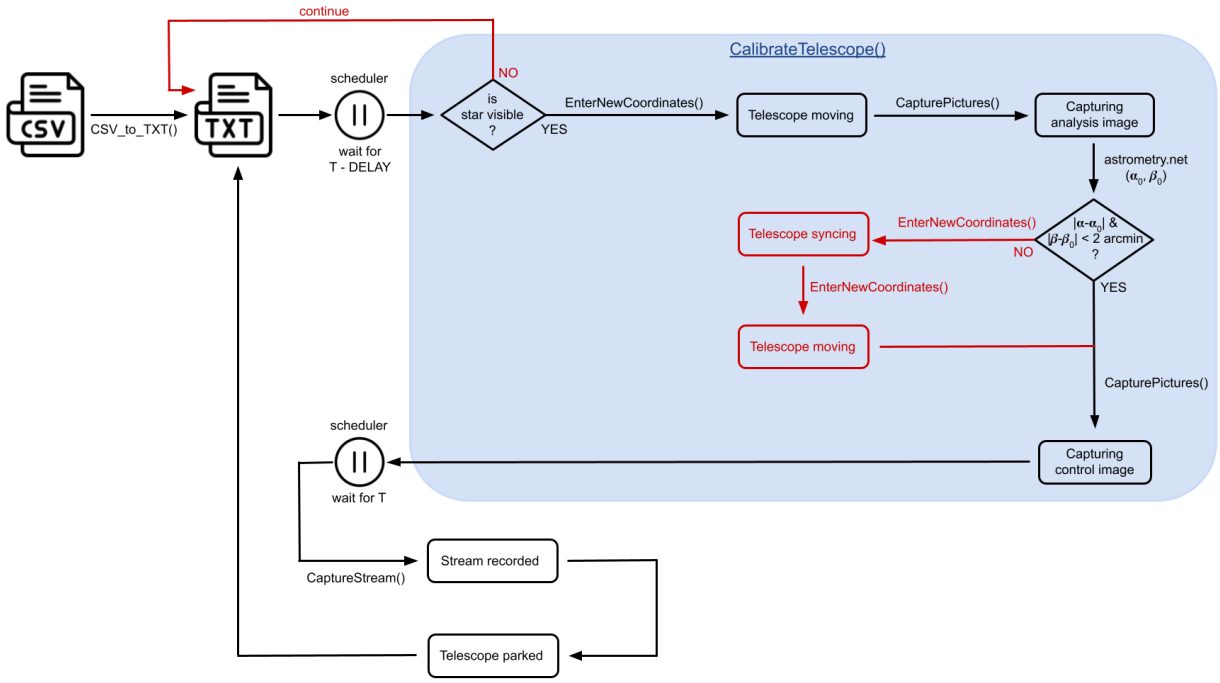
2. capture of the stream

The steps to capturing the stream are fairly straightforward:

- starting the stream (setting the streaming property switch ON)
- setting the streaming exposure
- setting the duration of the recording
- starting the recording of the stream
- turning the stream of once the duration of the recording's passed

3. parking of the telescope

To park the telescope, the telescope property for parking is switched on, and the telescope will move to predefined parking coordinates. Then, we make sure the tracking mode of the telescope is disabled so that the telescope does not move until the next target's time has come.



III/ Details for secondary functions

1. EnterNewCoordinates

This function gives the telescope new equatorial coordinates to track, and waits for it to be locked on its new position (by interrogating the state of the property and waiting until it is not BUSY anymore).

2. CapturePictures

This function takes in argument a star and a list of exposures to capture the star with. For every given exposure, the program will wait for the previous picture to have been captured before taking a new one, while the first one is being processed and saved to a fits file, the name of which contains information on the star itself and the time of the capture. All pictures are saved into a folder named with the current date.

The function returns a list of the names of the paths. The use of this function in the main function is restricted to one exposure each time it is called, and the path returned is used for astrometry.net.

3. MagnitudeToExposure

The aim of this function is to give an equivalent of exposure for magnitude given, in a way that the relation between magnitude and exposure is the same for every star (no matter the magnitude).

IV/ Terminating the program

Once the all the stars have been captured, it is time to properly stop the program. First, the devices have to be disconnected from the server (the same way they were connected to it). NB: the CCD camera's cooler should be turned off before disconnecting the camera. Once the devices are disconnecting, the server can be killed through a terminal, and the Raspberry PI can be shutdown.