

Science and Technology Council, IIT Kanpur

Astronomy Club, IIT Kanpur - Summer Project 2017

OAAR Automation

Students:

1. Sandarsh Gupta (160620)
2. Jaidev Ashok (160303)
3. Kishan Sankharva (160332)
4. Spoorthi S.L. (160704)
5. Malay Kumar Mohanta (160377)

Mentors:

1. Satish Nande (12632)
2. Ankit Bende (16112001)
3. Pranav Kulkarni (150346)

Background

The student-made Observatory for Amateur Astronomical Research (OAAR) at the Airstrip, IIT Kanpur, comprises a Celestron CGEM DX 1400 HD telescope housed inside a retractable Alt-Az dome. It is a 14-inch aperture telescope resting on a computerized Equatorial mount on a tripod. Currently, an autoguiding scope, the autofocuser hardware, and a DSLR camera are attached to the telescope. The system weighs an approximate 80 kilograms.

Currently, all motion is motorized but must be triggered and operated by hand at will. This includes moving the telescope and dome with remotes, triggering the autoguiding sequences, focusing, commanding image capture, and image processing. The observatory telescope is generally used to perform astrophotography. Observations with this telescope are possible but highly cumbersome due to the sluggish nature of the system unsuitable for efficient manual handling.

Benefits of the Project

- All processes of astrophotography that were initially done manually, including internal error corrections, will be fully automated.
- It will overcome the need to be in the observatory for the duration of the entire procedure.
- It will allow remote access to the observatory from anywhere within and outside the campus.
- Completion of this project will enhance reach, giving access to a wider group of people, so one can have the OAAR capture a photograph and send it back to the person in question, in real time.
- Basically, it will result in a faster and more efficient means of data collection for amateur and research purposes without the demand for manual supervision.

Modules

1. Weather monitoring
2. Raindrop sensor
3. Dome-telescope alignment
4. Autofocuser
5. Astrometry
6. Autoguider
7. Image Processing
8. Database management

1. Weather Monitoring

Weather monitoring is very important in the context of an automated observatory. so as to protect the equipments of the observatory. Obviously, an optical telescope is useless under a cloudy sky, and furthermore, weather phenomena such as precipitation can damage the optics, the electronics and mechanical equipment.

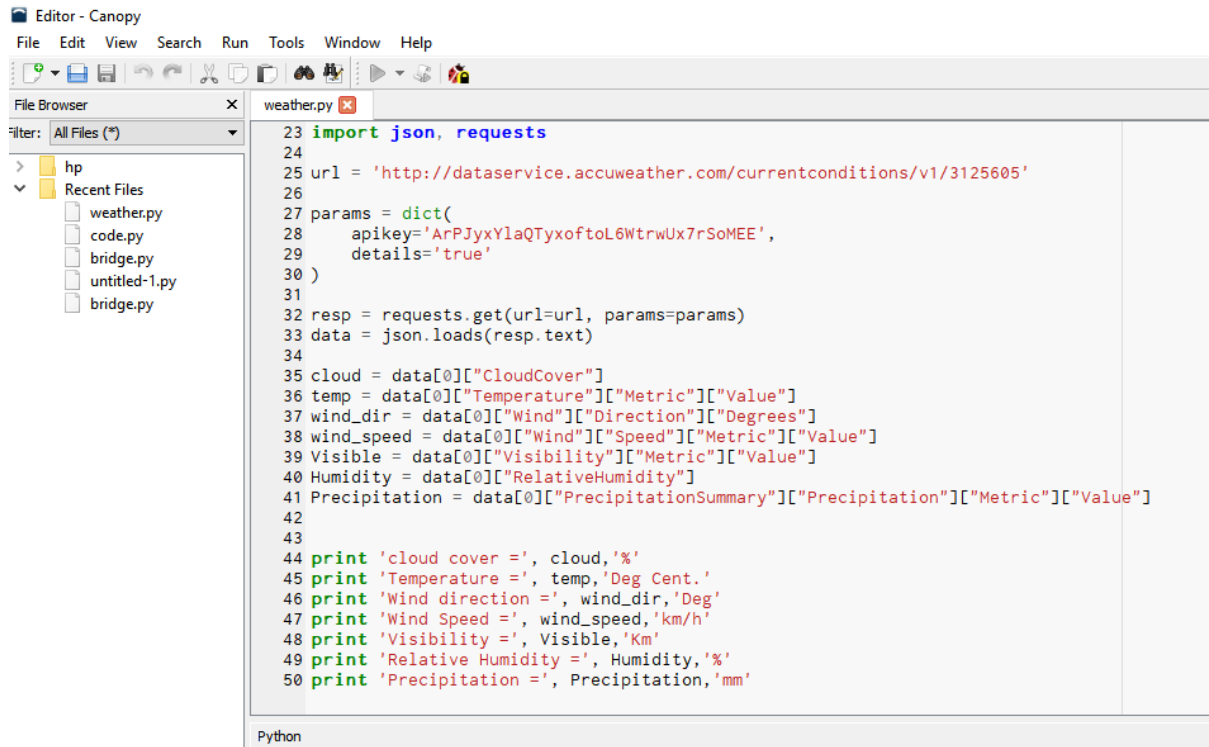
We will be using real time data from accuweather to monitor wind speeds, humidity, rain, temperature, daylight, dew point and cloud conditions and use the data to create a standard one-line weather.

Working:

- We have written a python code that will collect data about cloud cover , temperature, wind direction, wind speed and precipitation from the website [AccuWeather](#).
- Under unfavourable weather conditions, the observatory will be closed.

Online Databases used:

- AccuWeather



```

23 import json, requests
24
25 url = 'http://dataservice.accuweather.com/currentconditions/v1/3125605'
26
27 params = dict(
28     apikey='ArPJyxYlaQTyxoftoL6WtrwUx7rSoMEE',
29     details='true'
30 )
31
32 resp = requests.get(url=url, params=params)
33 data = json.loads(resp.text)
34
35 cloud = data[0]["CloudCover"]
36 temp = data[0]["Temperature"]["Metric"]["Value"]
37 wind_dir = data[0]["Wind"]["Direction"]["Degrees"]
38 wind_speed = data[0]["Wind"]["Speed"]["Metric"]["Value"]
39 Visible = data[0]["Visibility"]["Metric"]["Value"]
40 Humidity = data[0]["RelativeHumidity"]
41 Precipitation = data[0]["PrecipitationSummary"]["Precipitation"]["Metric"]["Value"]
42
43
44 print 'cloud cover =', cloud, '%'
45 print 'Temperature =', temp, 'Deg Cent.'
46 print 'Wind direction =', wind_dir, 'Deg'
47 print 'Wind Speed =', wind_speed, 'km/h'
48 print 'Visibility =', Visible, 'Km'
49 print 'Relative Humidity =', Humidity, '%'
50 print 'Precipitation =', Precipitation, 'mm'
  
```

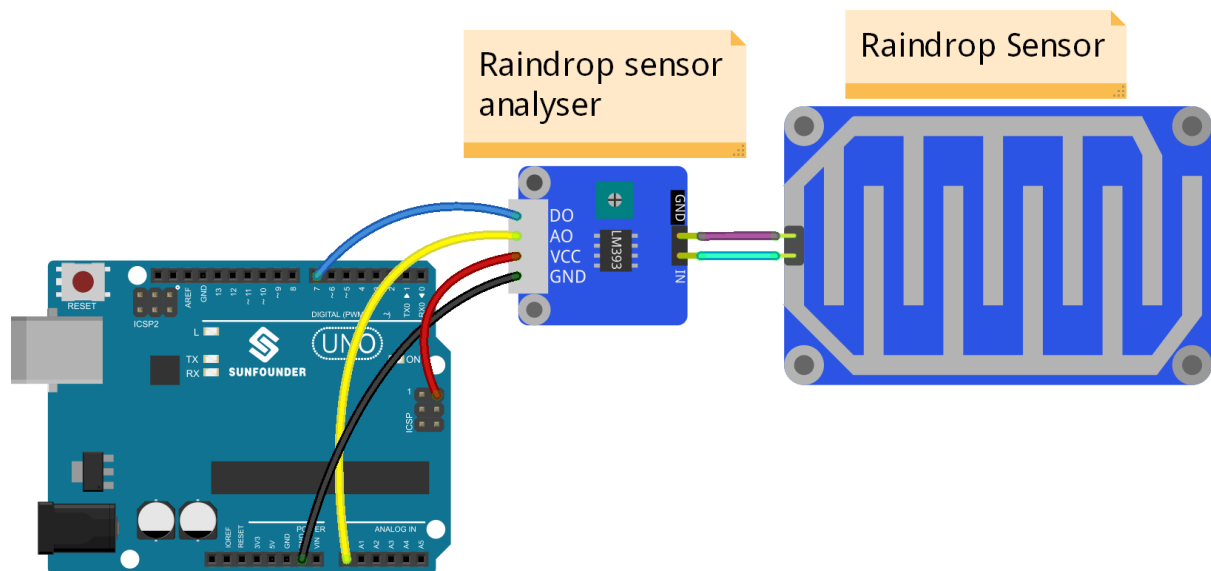
2. Raindrop Sensor

Even though we will get weather predictions from accuweather, it is not hundred percent foolproof. Any sudden change in weather conditions will be harmful, most notably precipitation. For this reason we have installed a rain drop sensor.

Working:

- A raindrop sensor or rain switch is a switching device activated by rainfall.
- The module includes a rain board and a control board that are separate for more convenience.
- It has a power indicator LED and an adjustable sensitivity through a potentiometer. The module is based on the LM393 op amp. It includes a printed circuit board(control board) that “collects” the rain drops.
- As rain drops are collected on the circuit board, they create paths of parallel resistance that are measured via the op amp.

- The lower the resistance (or the more water), the lower the voltage output. Conversely, the less water, the greater the output voltage on the analog pin.
- A completely dry board for example will cause the module to output five volts.



fritzing

Specifications of Raindrop sensor:

- Operating voltage: 5V
- Provide both digital and analog output
- Adjustable sensitivity
- Output LED indicator
- Compatible with Arduino
- TTL Compatible
- Bolt holes for easy installation

Dimensions:

- 3.2 cm x 1.4 cm x 1.0 cm

Materials Used:

- Arduino UNO microcontroller
- Raindrop sensor
- Raindrop sensor analyser
- 2 Female/Female jumper wires

Software used:

- Arduino editor

3. Dome and telescope alignment

To automate an observatory, dome telescope alignment is a very important step. In most cases, an observatory is a small, finite sized dome that contains a telescope that along with its mount covers a major volume inside of it. The consequence is that one cannot rely on simple mathematical conversions from the Equatorial system to the Horizontal coordinate system and vice-versa to determine the movement of the dome with respect to the telescope. The weighting factors that come into play are the dimensions of the mount and the telescope and have to be accounted for in the calculations. To ease out this process, we tabulate a sufficient number of data points between the coordinates of the telescope and the azimuth of the dome to be able to extrapolate a continuous, approximate graph to use as reference for all movements.

Working:

- We have written the code for data collection of dome telescope alignment.
- We will extrapolate the data points using MATLAB.
- The final code for dome telescope alignment will be prepared by using data from MATLAB.

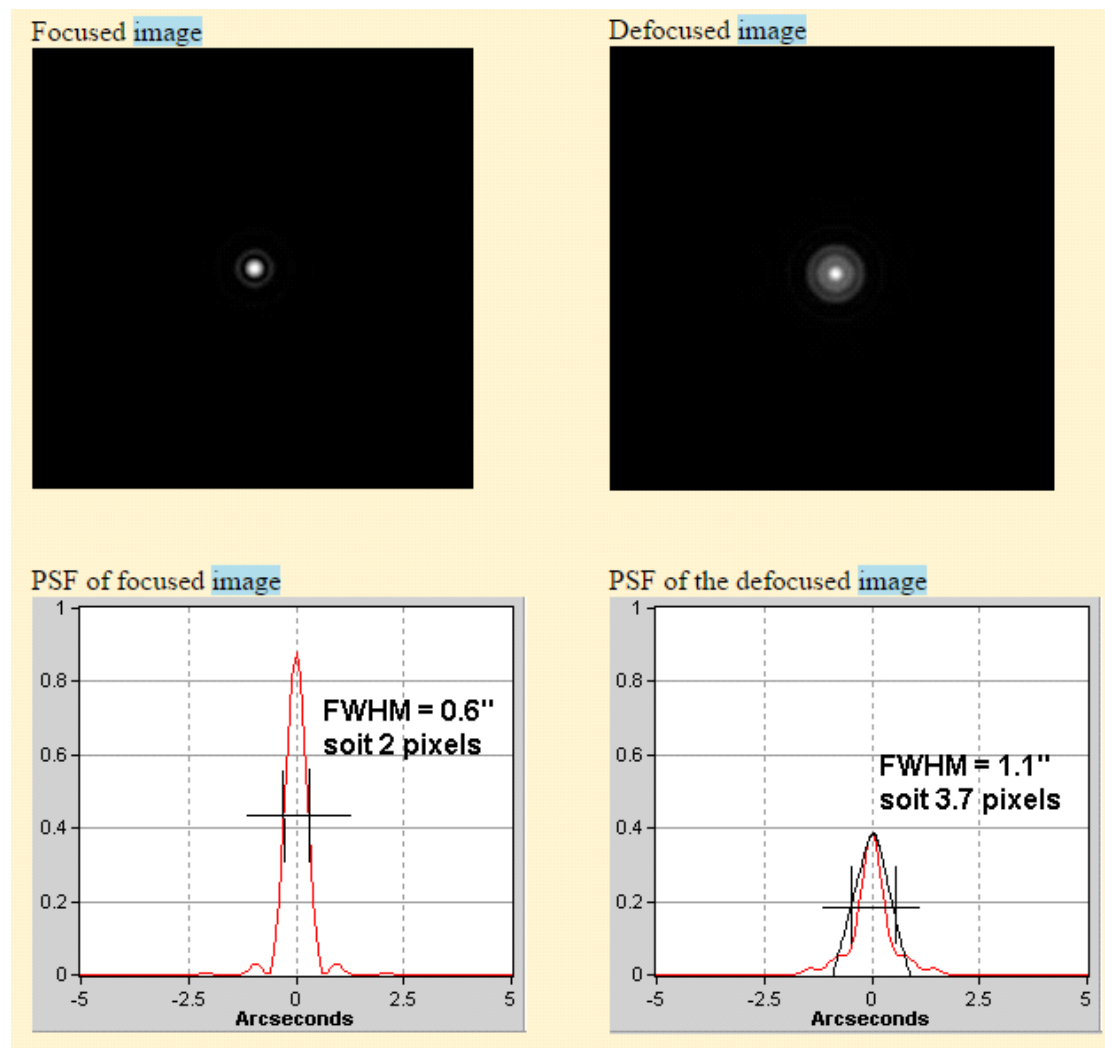
Problems:

- Syntax of ASCOM functions were unclear.
- Implementation of code on hardware cannot be done due to dome and telescope's connection issues with the computer.
- There are some problems with polar alignment of the telescope

4. Autofocuser

One of the fundamentally essential processes in astrophotography, like in regular photography, is the focusing of celestial objects to obtain a crisp image after capture. This generally requires a manual influence over the focusing knob of the telescope, and is limited by the capability of the eye to see through an eyepiece or watch a computer screen to judge the sharpest point-image over a range of the turn of the knob. The creation of an autofocuser drives us closer to the ideal level of telling objects apart, or visual resolution, with objective machine-level accuracy.

The Autofocuser utilizes the CCD (Charge-Coupled Device) attached to the telescope, or equivalently a DSLR camera, to consistently image the field of view at different focal positions using an imaging software like FocusMax or MaxIm DL. The data of illumination over the range of pixels is graphed each time (called a Point Spread Function), and the Full Width at Half Maximum (FWHM) for every instance is calculated alongside. When the image is sharpest, it is trivially understood that the FWHM is minimum. A Python-driven code controls the motor and triangulates right to this point.



credit: www.astrosurf.com

Materials used:

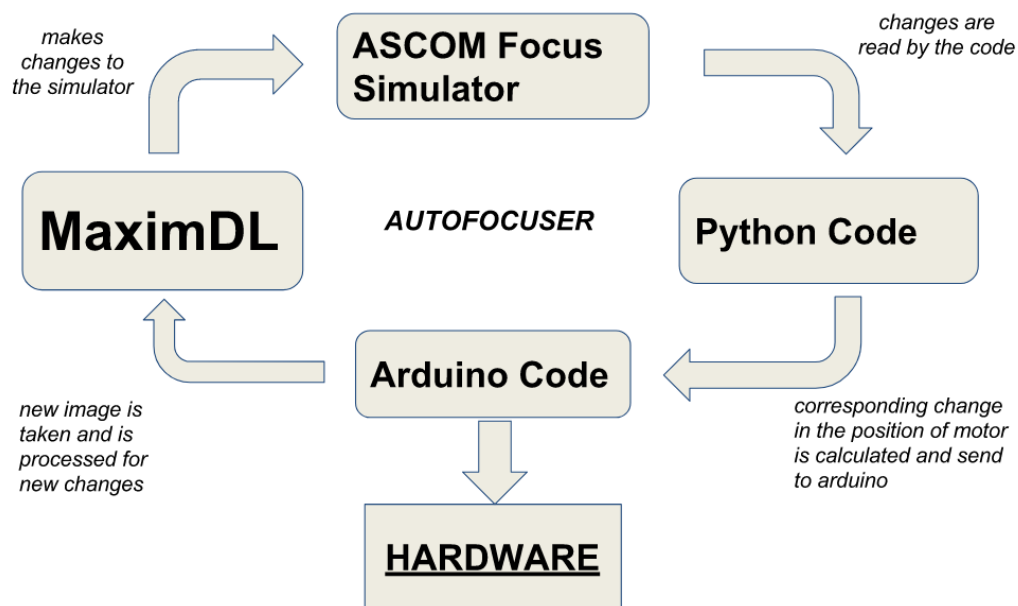
- A stepper motor
- Motor driver
- PTFE - cut into an 8-tooth gear
- PTFE - cut into a 33-tooth gear
- Arduino UNO microcontroller

- Metal sheet (for the supporting plate)
- Wires
- Power adapter and jumpers

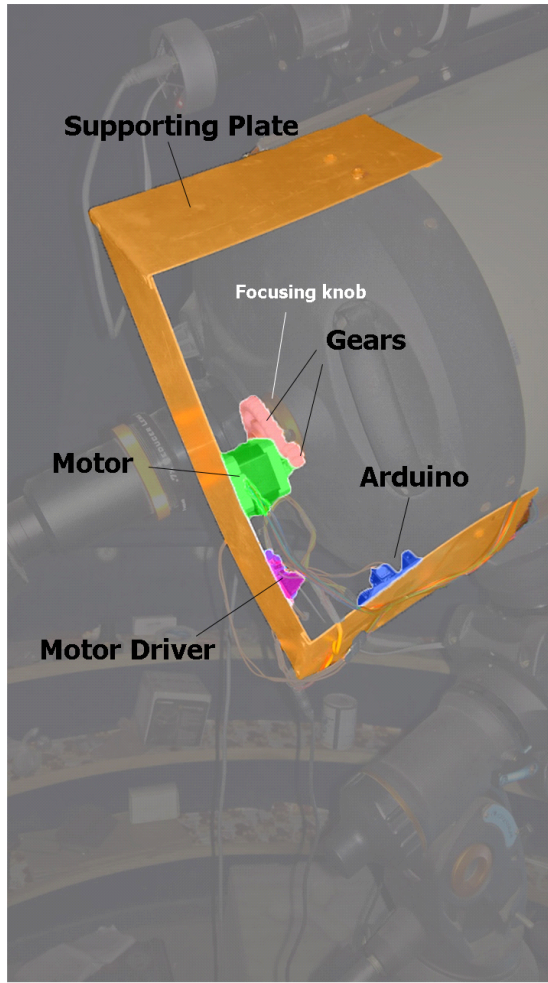
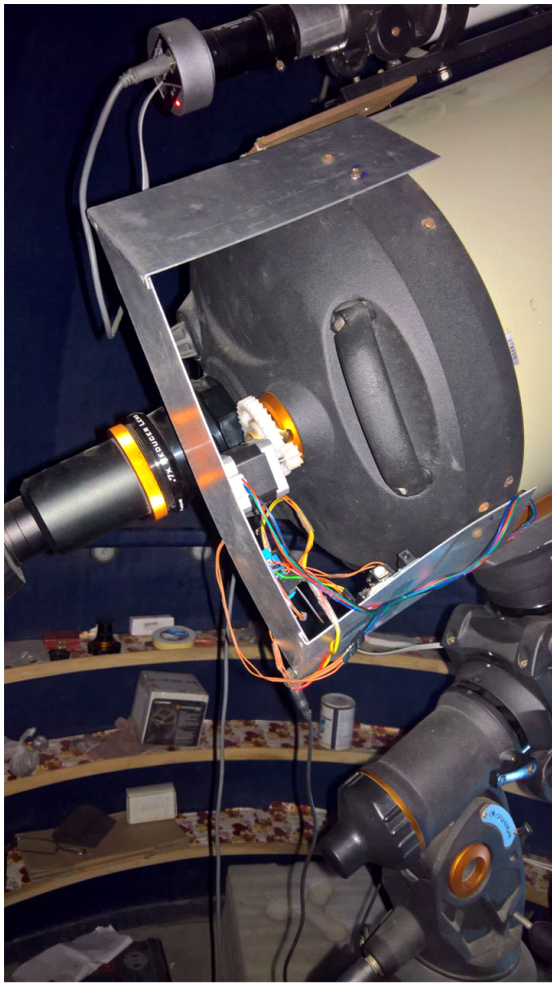
Softwares used:

- Maxim DL
- ASCOM simulator
- Arduino editor

Basic Flow of Control



The entire autofocusing equipment would be attached onto the telescope body. This could be achieved by a metallic plate onto which the tinier individual components are strapped on. The knob for manual focus must be clipped on to by a gear or a combination of gears, the genesis of which would be at the axle of a motor. PTFE (Poly-tetrafluoroethane) slabs were cut into two gears of 9 and 33 teeth. The 9-tooth gear would be on the motor's axle, while the larger 33-tooth gear would be on the focuser knob for increased sensitivity and finer focus adjustment. The motor would be wired to a motor driver controlled by an Arduino microcontroller board, which would ultimately be wired to the USB port of the telescope's accompanying PC.



The Autofocuser was a 2016 Winter project of the Astronomy Club. In its current state, its mechanics have been implemented and fully assembled, and the autofocuser has been affixed to the observatory telescope. The motor has been run by power and the focuser tested for its sensitivity. The code for its working has been written and functioning of the code has been determined fine after running it with a simulator.


```
code.py
1 import serial
2 from win32com.client import Dispatch
3
4 foc = Dispatch("ASCOM.Simulator.Focuser")
5 ser = serial.Serial("COM3",9600)# write com port
6
7 foc.Connected = True #check
8
9 inipos = foc.Position
10 oldpos = inipos
11 c = True
12
13 while(c):
14     while(foc.IsMoving):
15         delay(100)
16         newpos = foc.Position
17         sign = 1
18         steps = newpos - oldpos
19         if(steps < 0):
20             sign = -1
21             steps = -steps
22         rev = int(steps/200)
23         steps = steps%200
24         if(steps<>0 or rev<>0):
25             line=""
26             while(line==""):
27                 ser.write("a")
28                 line=ser.readLine()
29             ser.write(sign)
30             line=""
31             while(line==""):
```

Python Change

Above: Python code for the autofocuser

```
File Edit Sketch Tools Help
sketch_jun10a
#include <Stepper.h>

const int stepsPerRevolution = 200;

Stepper motor(stepsPerRevolution, 8, 9, 10, 11);

int index=0;
int value[]={0, 0, 0};

void setup() {
    motor.setSpeed(60);//get input
    Serial.begin(9600);
}

void loop() {
    if(Serial.available()) {
        Serial.println("a");
        value[index] = Serial.read();
        index++;
        if(index==3) {
            index = 0;
            runmotor();
        }
    }
}

void runmotor() {
```

Above: C code for Arduino controller

5. Astrometry

Even motorised telescopes like the one in OAAR, are not perfectly calibrated. So, they need to be corrected for minuscule errors everytime. The website nova.astrometry.net helps in this purpose. Image of any part of the sky can be solved for the RA and Dec of the centre of image using this website. The telescope is then synced with this information to correct errors.

Working:

- www.nova.astrometry.net uses the star patterns in the image to find the part of the sky in the image. This is how it finds the RA and Dec of centre of the astronomical image.
- We have written a python code using nova API. The code will automatically login to the website and upload the image to be solved.
- The code will then monitor the progress in solving of the image. Once solved, it will retrieve the data of RA and Dec and automatically sync the telescope to given coordinates.
- The code is designed to report any errors, if any, while solving the image.

Online Databases:

- Nova Astrometry

```

IDLE File Edit Format Run Options Window Help
nova-2.py - /Users/spoorthi/Downloads/nova-2.py (3.6.1)

import client
import time
import sys
import win32com.client
import winsound

from Tkinter import Tk
from tkinterFileDialog import askopenfilename

#uncomment line 14 and comment line 13 to do actual alignment of our celestron Telescope
#replace the line 29 apikey to your account instead of club astrometry account if you want that

tel = win32com.client.Dispatch("ASCOM.Simulator.Telescope")
#tel = win32com.client.Dispatch("ASCOM.Celestron.Telescope")
tel.Connected = True

Tk().withdraw() # we don't want a full GUI, so keep the root window from appearing
filename = askopenfilename() # show an "Open" dialog box and return the path to the selected file
print(filename)
print "\n"

nova = client.Client()

approx_ra = tel.RightAscension
approx_dec = tel.Declination

for i in xrange(5):
    log_result = nova.login("kmhxpnmvsmeycapf")
    if log_result["status"] == "success":
        break
if (i == 5):
    print("Login failed.\n")
    sys.exit()
else:
    session_id = log_result["session"]
    print("\nLogin Succeeded.")
    print("Session ID: %s\n" %(session_id, ))

jobs = nova.myjobs()
print "\n"
last_job_id = jobs[0]

for i in xrange(5):
    sub_result = nova.upload(filename)
    if sub_result["status"] == "success":
        break
if (i == 5):
    print("Image upload failed.\n")
    sys.exit()
else:
    subid = sub_result["subid"]
    print("\nImage upload succeeded.")
```

Above and below: nova.py codes for astrometry

```

IDLE File Edit Format Run Options Window Help
nova-2.py - /Users/spoorthi/Downloads/nova-2.py (3.6

print("Login failed.\n")
sys.exit()
else:
    session_id = log_result["session"]
    print("\nLogin Succeeded.")
    print("Session ID: %s\n" % (session_id, ))

jobs = nova.myjobs()
print "\n"
last_job_id = jobs[0]

for i in xrange(5):
    sub_result = nova.upload(filename)
    if sub_result["status"] == "success":
        break
    if (i == 5):
        print("Image upload failed.\n")
        sys.exit()
    else:
        subid = sub_result["subid"]
        print("\nImage upload succeeded.")
        print("Submission ID: %d\n" % (subid, ))

while (jobs[0] == last_job_id):
    time.sleep(30)
    jobs = nova.myjobs()
    print "\n"

job_id = str(jobs[0])
job_result = nova.job_status(job_id)

if (job_result["status"] == "failure"):
    print("Solving image failed.\n")
    sys.exit()
elif (job_result["status"] == "success"):
    print("\nImage solved for RA and Dec.\n")
    print ("RA = %f, Dec = %f" % (job_result["calibration"]["ra"], job_result["calibration"]["dec"]))

    print("Objects in field: ")
    no_of_objects = len(job_result["objects_in_field"])
    for i in xrange(no_of_objects):
        print ("%d. %s" % (i+1, job_result["objects_in_field"][i]))

tel.SyncToCoordinates(job_result["calibration"]["ra"]/15.0, job_result["calibration"]["dec"])

Freq = 1000 # Set Frequency To 2500 Hertz
Dur = 800 # Set Duration To 1000 ms == 1 second
winsound.Beep(Freq,Dur)

tel.Connected = False
```

Resources

Hardware:

- Arduino UNO microcontrollers
- PTFE slabs cut into 8-tooth and 33-tooth gears
- Motor
- Motor Driver
- Raindrop sensor kit

Software:

- MaxIm DL
- ASCOM Simulator
- Arduino Editor
- Digital Dome Works

- MATLAB
- Adobe Photoshop
- DeepSkyStacker

Online Databases

- AccuWeather
- Nova Astrometry

Timeline

Week	Action	Status
Week 1	Weather monitoring	completed
Week 2	Rain drop sensor	nearing completion
Week 3	Fixing and utilizing the Autofocuser	completed
Week 4	Astrometry	completed
Week 5	Dome-telescope alignment	not completed
Week 6	Autoguider	not completed
Week 7	Image capture and processing, database mgmt.	not completed
Later times (if required)	Error handling, debugging	-

Possible Future Additions

- This project can be upgraded to perform **photometry** and **spectroscopy** of celestial objects and return useful data of high research value.
- It can also be upgraded to calculate magnitudes, distances, luminosities, masses, speeds, sizes, time periods of motion, surface temperature and atmospheric composition (if any) of unidentified objects in the sky.

(as on 20 June, 2017)