Astronomy Club, IIT Kanpur

Summer Project 2017 – OAAR Automation

Background

- **Telescope basics: Celestron CGEM DX 1400 HD**

  Aperture: 14 inches

  Weight: approx. 80 kg

  Mount: Equatorial (motorised)

  Dome: Retractable, Alt-Az

- Utility: Astrophotography
- Other uses: Too sluggish.

About the Project

- Long-term
- Aim: to automate the various processes of telescope and observatory to the extent such that basic night sky observation and astrophotography can be done remotely with ease.
- Target for summer: 70%

Modules

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

Benefits

- Automation of astrophotography processes
- Human presence-independent
- Remote access
- Wider reach to the public
- Faster means of data collection

Workplan

- Programs for each of the modules
- Code-hardware syncing
- Debugging

1. Weather Monitoring

Editor - Canopy

File Edit View Search Run Tools Window Help

File Browser

Filter: All Files (*)

- hp
- Recent Files
  - weather.py
  - code.py
  - bridge.py
  - untitled-1.py
  - bridge.py

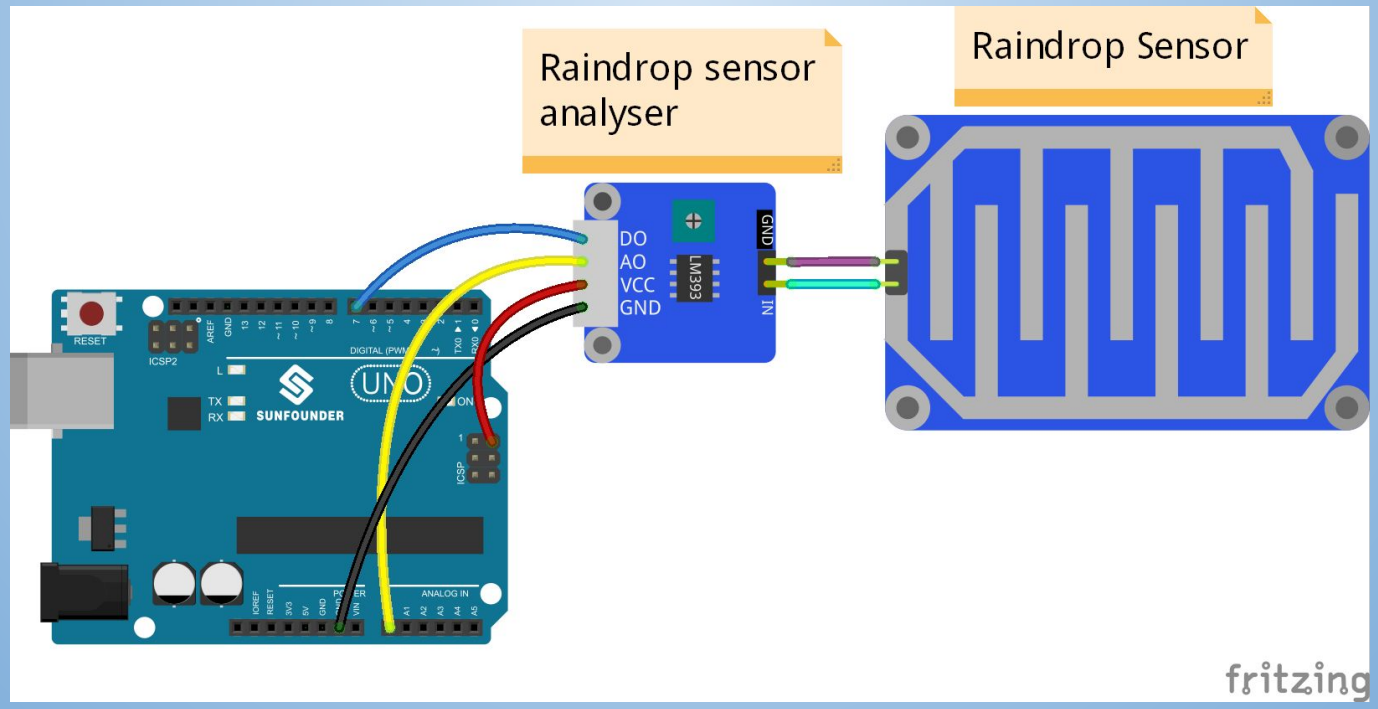
weather.py

```python
import json, requests

url = 'http://dataservice.accuweather.com/currentconditions/v1/3125605'

params = dict(
    apikey='ArPJyxYlaQTyxoftoL6WtrwUx7rSoMEE',
    details='true'
)

resp = requests.get(url=url, params=params)
data = json.loads(resp.text)

cloud = data[0]["CloudCover"]
temp = data[0]["Temperature"]["Metric"]["Value"]
wind_dir = data[0]["Wind"]["Direction"]["Degrees"]
wind_speed = data[0]["Wind"]["Speed"]["Metric"]["Value"]
Visible = data[0]["Visibility"]["Metric"]["Value"]
Humidity = data[0]["RelativeHumidity"]
Precipitation = data[0]["PrecipitationSummary"]["Precipitation"]["Metric"]["Value"]


print 'cloud cover =', cloud,'%'
print 'Temperature =', temp,'Deg Cent.'
print 'Wind direction =', wind_dir,'Deg'
print 'Wind Speed =', wind_speed,'km/h'
print 'Visibility =', Visible,'Km'
print 'Relative Humidity =', Humidity,'%'
print 'Precipitation =', Precipitation,'mm'
```

Python

2. Raindrop Sensor

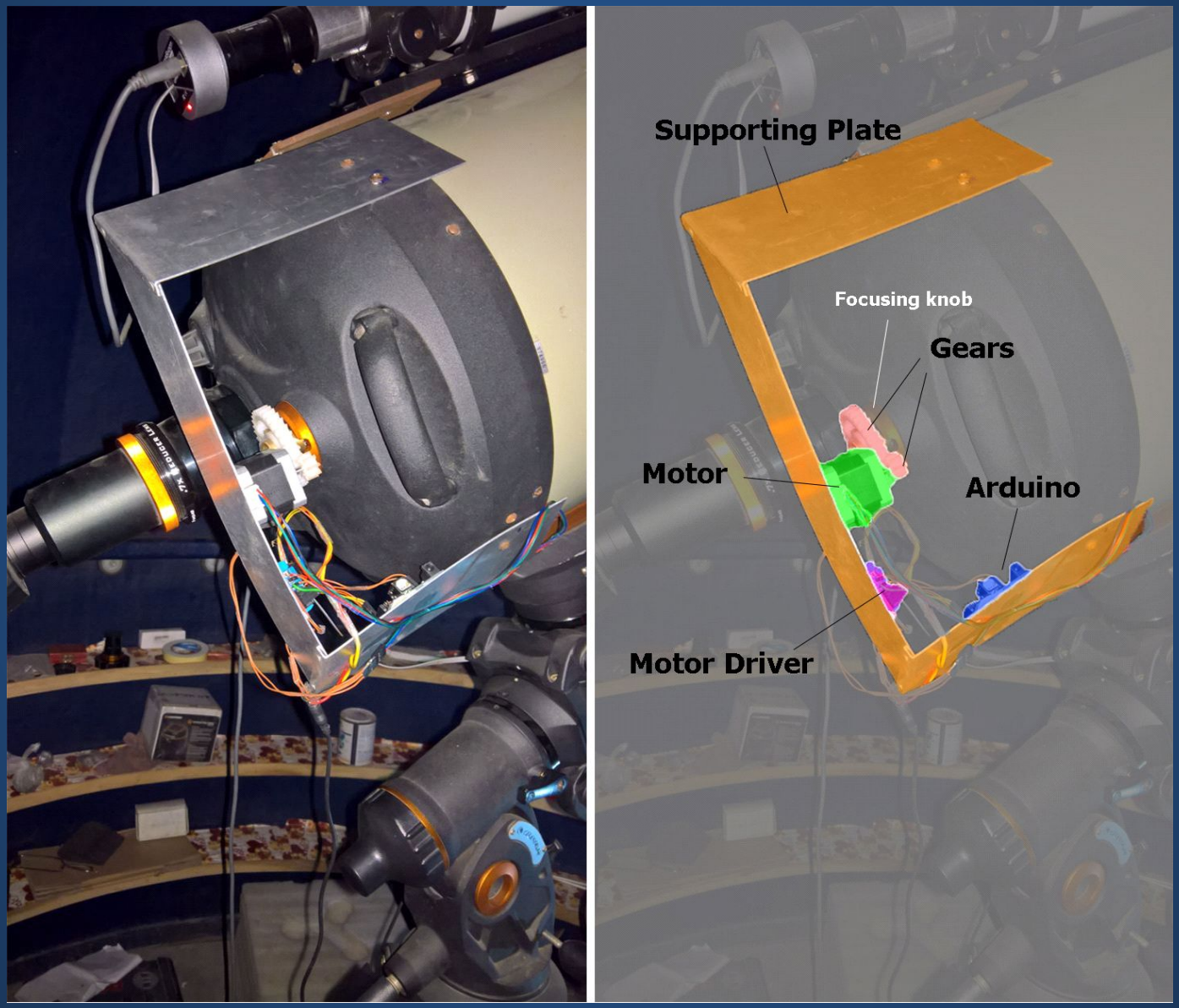
- Droplet detector
- Responds to change in resistance



3. Dome-Telescope Alignment

- Set of Data Points to determine Dome-Telescope relative positions
- Extrapolation and Graphing - MATLAB
- Issues:
  - Software updates
  - PC-Dome connection

4. Autofocuser



code.py

```python
import serial
from win32com.client import Dispatch

foc = Dispatch("ASCOM.Simulator.Focuser")
ser = serial.Serial("COM3",9600)# write com port

foc.Connected = True #check

inipos = foc.Position
oldpos = inipos
c = True

while(c):
    while(foc.IsMoving):
        delay(100)
    newpos = foc.Position
    sign = 1
    steps = newpos - oldpos
    if(steps < 0):
        sign = -1
        steps = -steps
    rev = int(steps/200)
    steps = steps%200
    if(steps<>0 or rev<>0):
        line=""
        while(line==""):
            ser.write("a")
            line=ser.readLine(
        ser.write(sign)
        line=""
        while(line==""):
```

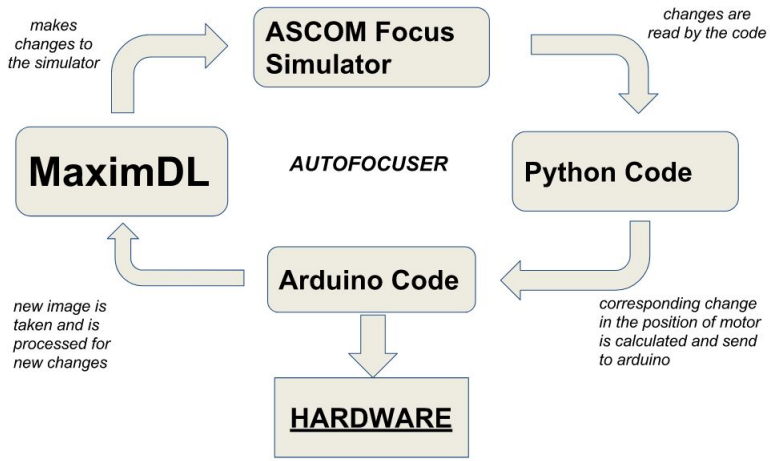
Python

File Edit Sketch Tools Help

sketch_jun10a

```arduino
#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) {
```

Basic Flow of Control



5. Astrometry

- Get **RA** and **DEC** of the **centre of the image**
- Reference database: Nova Astrometry
- **client.py** file with some changes from nova.astrometry.net

NOVA ASTROMETRY CODE

IDLE File Edit Format Run Options Window Help

nova-2.py - /Users/spoorthi/Download

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

from Tkinter import Tk
from tkFileDialog 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("kmhjxpnvsmeycapf")
    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.")
```

IDLE File Edit Format Run Options Window Help

nova-2.py - /Users/spoorthi/Downloads/nova-2

```
    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
```

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

- Photometry
- Spectroscopy
- Can be upgraded to calculate
  - magnitudes, distances, luminosities
  - masses, speeds, sizes, time periods of motion
  - surface temp. & atmospheric composition of any unidentified objects in the sky.

Thank You!

Documentation link:
https://docs.google.com/document/d/152W9Zm6HMiCJ8ytvQfFaABruLJzFg_coycztATT1qvk/edit?usp=sharing

Sandarsh Gupta **Jaidev Ashok** **Kishan Sankharva**

Spoorthi S.L. **Malay Kumar Mohanta**