Faculty of Engineering
Systems & Computers DEPT.

AL-Azhar University
Faculty of Engineering
Systems & Computers Department

كلية الهندسه
قسم هندسة النظم و الحاسبات

Graduation Project Documentation

Sky Guide / Tracker

A Project Submitted in partial fulfilment of the requirements for the Degree of Bachelor of Science in Systems and Computers Engineering

Submitted By

Sherif Gamal Abdellatif	**404045**
Abdelrahman Farouk Sayed	**404052**
Attia Sayed Attia	**404060**
Ahmed Sedek Ali	**404010**
Ahmed Ragab Abdelkader	**4040111**

Supervised by
Dr. Khalid Elshafey

2020/2021

Sky Guide / Tracker

2020/2021

Each graduation project is assigned to an examiner committee. The committee has three members nominated by the graduation project committee. The examiner committee has a president, and two more members such that the project supervisor is one of the two members. The name, rule, and signature of each examiner is shown in the following table. The examiners' signatures are required before students can submit their project final report/documentation. Therefore, students are responsible for asking each examiner to sign next to his name in the following table.

Examiner Committee

Name	**Rule**	**Signature**
Prof.	President	
Dr.	Supervisor	
Dr.	Member	

ABSTRACT

For most people, the night sky is a mystery because it’s really difficult to find a certain object in the night sky using the naked eye, however, in reality the night sky is a detailed map;

Just like the coordinates of any place on earth, each celestial object can be located using specific coordinates, our project aims to exploit this fact.

Sky Guide consists of two main parts that enable the user to find and track any object in the sky, these are:

- *Desktop application*:

  We’ve established a moderate database, that contains a set of coordinates and information for some of the most famous celestial objects (stars, nebulas, etc); From the desktop application we access this database to retrieve the coordinates of the required object, and send it to the physical device.
- *Physical device:*

  The device receives the coordinates of the required object -from the desktop application- and points toward it.

This project is targeting the amateur astronomers and the astrophotography community, and it can be used as a Telescope German Equatorial Go-To Mount.

KEYWORDS: Astronomy; Astrophotography; Telescope Mount;

ACKNOWLEDGEMENTS

"Praise be to Allah, who guided us to this and we would never have been guided if Allah had not guided us."

We would like to thank Dr. Khaled Al-Shafei for advice, encourage, support and mentoring has offered us during project also in the class.

We would like to thank our friend and colleague Eng. Mostafa Arafa from the department of mechanics, for helping us greatly in the design of the physical device.

Also, we would like to thank staff doctors and Engineers of facility members of the department of systems and computer engineer for their help and encouragement.

Finally, we would like to thank our parents for support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY AND LIST OF ABBREVIATION

RA: Right Ascension

Dec: Declination

LST: Local Sidereal Time

AZ: Altitude Azimuth

CHAPTER 1 INTRODUCTION

1.1 Background and Motivation

The sky has always been our passion and having the opportunity to combine our interest in the sky and our knowledge in computer science was quite a motivating experience. We have found that in the amateur astronomy and astrophotography communities, they face quite a challenge in locating and tracking the celestial objects that they wish to observe or study. Thus, the aim of the work described in this report is to provide a software tool and a device controlled by this software, to find and track any celestial object across the sky.

1.2 Scientific Idea

Similar to the navigation on earth, where to find a position on earth, a one must know its coordinates -longitude and latitude, there must be a celestial coordinate system to enable us to navigate the sky, this introduces us to the concept of “Celestial sphere” & “Celestial coordinate systems”.

1.2.1 Celestial sphere

Celestial Sphere is an abstract sphere that has an infinite radius and shares the same center with the Earth. We imagine that the stars and planets are attached to the inside surface of the celestial sphere.
Standing outside on a clear moonless night far from city lights, it is easy to imagine that one is at the center of such a sphere and that the stars and planets are attached to its inside surface.

Extending the Earth's axis of rotation in both directions onto the celestial sphere determines two points, the north celestial pole and the south celestial pole, similarly, projecting the Earth's equator onto the celestial sphere determines the celestial equator.

As the Earth revolves around the Sun each year, we see the Sun seeming to travel across the celestial sphere. As it does, it traces out an imaginary great circle, which is called the ecliptic (Ford, 2014).

The plane determined by the Earth's equator is tilted with respect to the plane determined by the ecliptic, so the Sun is north of the equator for 6 months of each year and south of the equator for the other 6 months. The ecliptic and the equator cross at two points, the vernal equinox and the autumnal equinox. The vernal equinox is the point where the Sun crosses the equator on its way north each year, marking the first day of spring in the Northern Hemisphere. See illustration fig 1.1.

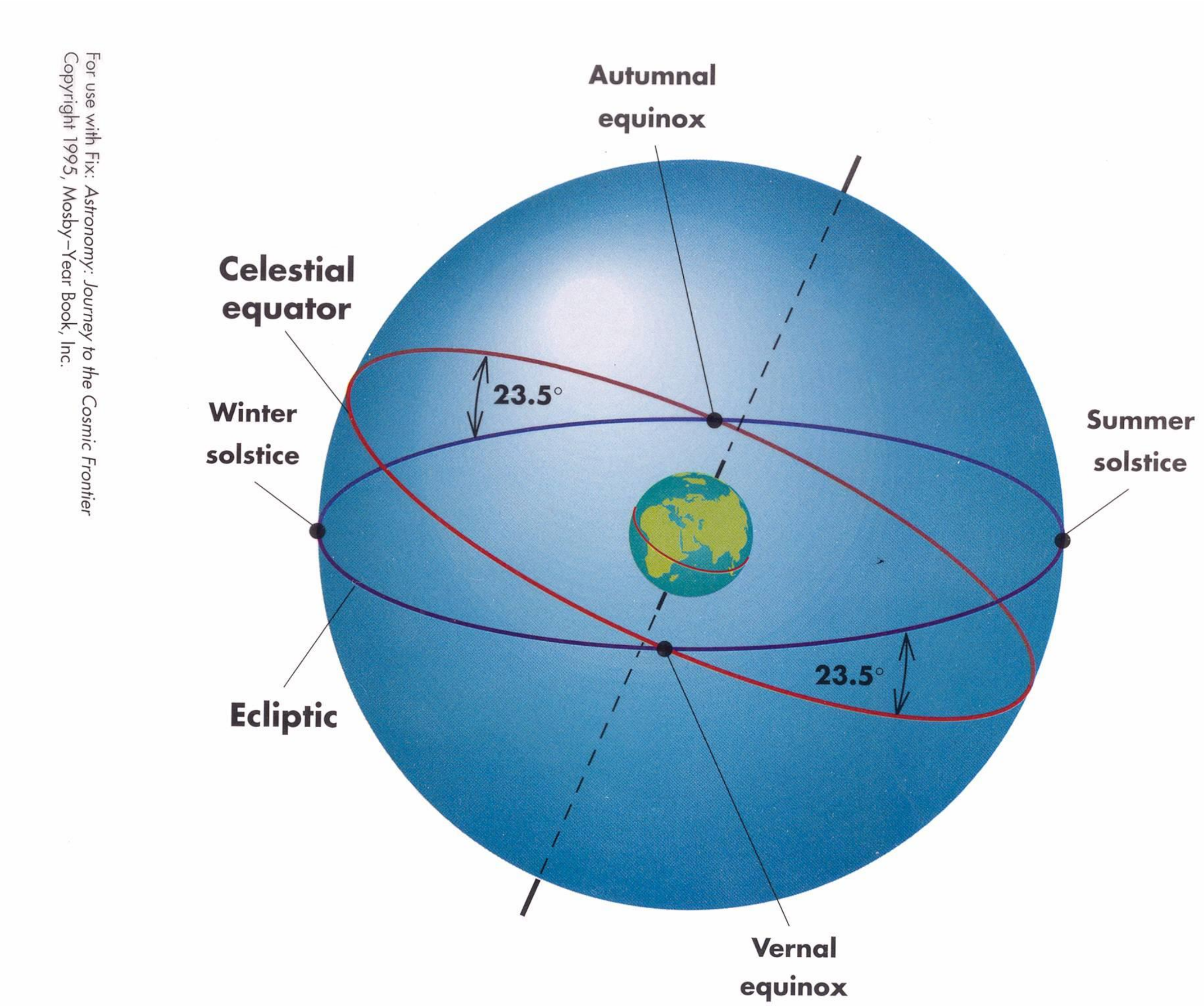


Fig 1.1: Visualization of celestial sphere around the earth

1.2.2 Celestial coordinate systems

There are many celestial coordinate systems, for example:
Horizontal system, Galactic system, Ecliptic system and Equatorial system -which is used in this project.

Equatorial coordinate system:
As the distances to the stars and planets are so great, two different observers see the same star in the same direction and thus the star can be thought of as being at a specific “position” on the celestial sphere.

Using the two poles, the equator, and the ecliptic, it is straightforward to establish a coordinate system that makes it possible to determine the position of any object in the sky. The system is similar to the system of latitudes and longitudes used for the locations of objects on the surface of the Earth. Right ascension -abbreviated RA or with the Greek letter α- is analogous to longitude. Declination -abbreviated Dec or with the Greek letter δ- is analogous to latitude.

As with longitude on the Earth, it is necessary to choose a “zero” point for right ascension, we use on earth a meridian -a great semicircle from one pole to the other, perpendicular to the equator- passing through Greenwich, England. On the celestial sphere we use the meridian that passes through the vernal equinox. Every point on this prime meridian has a right ascension of zero.

Right ascension is measured in hours and minutes, ranging from 0 hours, 0 minutes to 23 hours, 59.999… minutes. Hours are used rather than degrees because the entire sphere seems to rotate once per day. Each hour of right ascension corresponds to 15°. Right ascension increases toward the east.

Declinations are measured north and south of the equator with angles between 0° and 90°, measured in degrees and minutes. Northern declinations are considered positive, southern declinations negative.

For example: If a star is located at RA = 4.1h and Dec = 58.8°, figure 1.2 illustrates the position of this star.

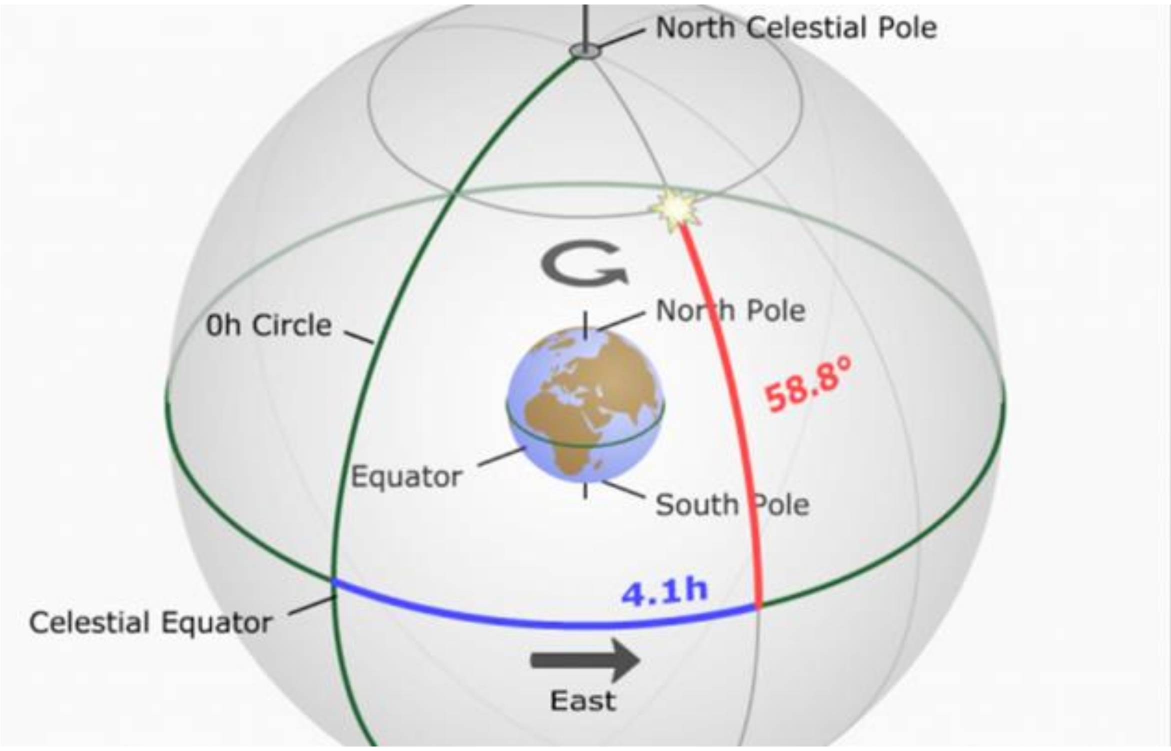


Fig 1.2: Finding star given its RA and Dec

1.2.3 Local sidereal time

Sidereal time is a timekeeping system that is used to locate celestial objects, astronomers use Sidereal Time to measure the movement of the celestial sphere.

Local sidereal time (LST) is the sidereal time where the observer's located, it is equal to the RA of any celestial object that is transiting the observer's meridian -the great circle passing through the celestial poles, as well as the zenith and nadir of an observer's location (see Fig 1.3)- at this particular moment, for example the time when the vernal equinox passages over the observer this marks the zero hour of the local sidereal time.

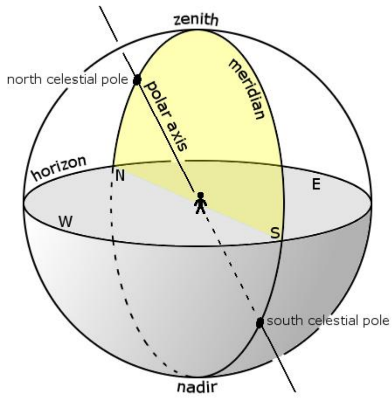


Fig 1.3: Observer meridian

1.3 Problem definition

As the earth revolves around itself a full revolution per day, any celestial object seems to be moving across the sky -from the east towards the west, due to this fact, it’s very difficult to find the exact RA of a particular celestial object, and to track this object across the sky. So, for a given celestial object, we shall have its coordinates -RA and Dec- and it’s required to find this particular object and track it across the sky.

1.4 Survey

There are many companies develop smart mounts in order to solve these problems
Among these companies:

- Celestron:

Celestron is a company that manufactures and distributes telescopes and telescope mount.
Examples of their telescope mounts: CGX EQUATORIAL MOUNT, Advanced VX Go-To German EM.

- iOptron:

iOptron is a global company specializing in the development, manufacturing, and marketing of innovative astronomical telescopes, mounts & accessories, and cutting-edge optical instrument for multiple applications.
Examples of their telescope mounts: Urban 90, Versa AZ

- Orion:

Examples of their telescope mounts: Orion SpaceProbe II

Unfortunately, there aren’t any local companies that make similar device.

CHAPTER 2 PROPOSED APPROACH

2.1 Introduction

To solve the problem discussed in the previous chapter, there were many challenges that faced us, some of these challenges are as follows:

- How to make the user select a particular celestial object?
- How to make the user input the coordinates of any celestial object?
- How to send these coordinates to the device to move towards the object?
- How to make the device receives coordinates of an object and moves towards it?
- How to make sure that the device reached its destination?
- How to make the device track the object across the sky?

To solve these challenges, we settled on the following approach.

2.2 Proposed approach

Our approach consists of two parts:

2.2.1 Software application

To solve the problem of enabling the user to select a particular celestial object, we have to store the positions and some information related to the most famous celestial objects in a database, which will be available for the user to access remotely through our desktop application.

The desktop application will communicate with the hardware device through a wireless communication to send the positions selected by the user. Also, it will enable the user to perform the following actions on Sky Guide database:

- Search for a celestial object by name.
- Explore specific coordinates.
- Add a new celestial object to the database.

The following celestial objects are supported by Sky Guide database: constellations, stars, nebulas, supernova remnants and the most famous objects in our solar system.

Finally, the desktop application will have the responsibility of updating the coordinates of the celestial object, thus, enabling the hardware device to track it across the sky.

2.2.2 Hardware device

We wanted to make an actual telescope-camera mount to point the telescope or the camera to the specified celestial object, like the one in Fig 2.1.

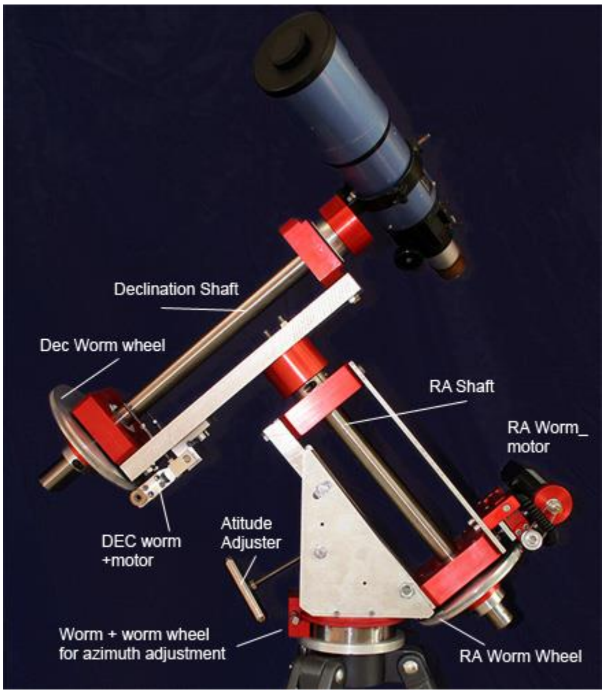


Fig 2.1: Equatorial mount design

Unfortunately, we are limited in the hardware resources and materials. So, we made a small prototype in which instead of controlling a telescope-camera we will control a laser which will be pointing at the celestial object.

The hardware device will communicate with the desktop application through a wireless communication to receive the required angles to which the laser will be pointing at.

CHAPTER 3 SYSTEM DESIGN AND IMPLEMENTATION

As mentioned previously, our project consists of two parts, software application and hardware device.

3.1 Software part

Sky Guide desktop application consists of three main parts, database, backend and frontend.

3.1.1 Database

The process of establishing Sky Guide database was done in three phases,
ERD, Data collection and uploading the database on clever cloud online server.

3.1.1.1 Entity relationship diagram (ERD) (ERDPlus, n.d.)

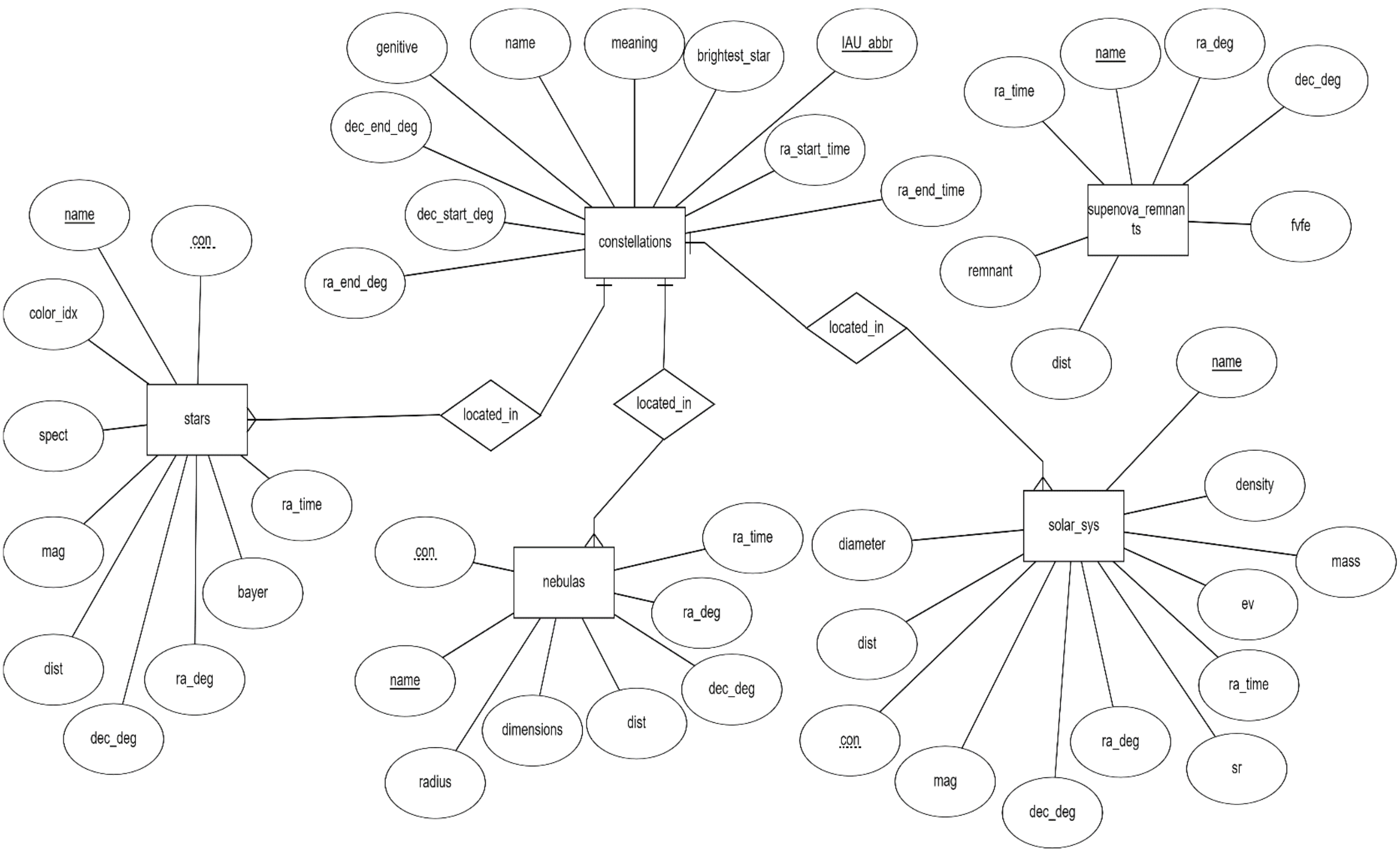


Fig 3.1: Sky Guide database ERD

3.1.1.2 Entities description

As shown in the previous diagram, Sky Guide database consists of five entities.

- Constellations: this table contains information about all the known 88 constellations, the description of these attributes are as follows (Wikipedia, n.d.).
  - *IAU_abbr*: the IAU (International Astronomical Union) abbreviation, which is a three-litter abbreviation of any constellation.
  - *name*: the Latin name of the constellation.
  - *meaning*: the meaning of the constellation name.
  - *genitive*: the name of the constellation which is used in its stars' bayer names.
  - *ra_start_time*: the RA position of the start point of the constellation in time notation (h m s).
  - *ra_end_time*: the RA position of the end point of the constellation in time notation (h m s).
  - *ra_start_deg*: the RA position of the start point of the constellation in degrees.
  - *ra_end_deg*: the RA position of the end point of the constellation in degrees.
  - *dec_start_deg*: the Dec position of the start point of the constellation in degrees.
  - *dec_end_deg*: the Dec position of the end point of the constellation in degrees.
  - *brightest_star*: the brightest star in the constellation.

- Supernova remnants: this table contains information about some of the most famous supernova remnants, the description of these attributes are as follows.
  - *name*: the name of the supernova remnant.
  - *ra_time*: the RA position of the supernova remnant in time notation.
  - *ra_deg*: the RA position of the supernova remnant in degrees.
  - *dec_deg*: the Dec position of the supernova remnant in degrees.
  - *dist*: the approximate distance between the earth and the supernova remnant in light years
  - *remnant*: what it is remnant of.
  - *fvfe*: first visible from earth.

- ❖ Stars: this table contains information about some of the most famous stars, the description of these attributes are as follows (Nash, 2011).
  - *name*: the name of the star.
  - *ra_time*: the RA position of the star in time notation.
  - *ra_deg*: the RA position of the star in degrees.
  - *dec_deg*: the Dec position of the star.
  - *dist*: the distance between the star and the earth.
  - *mag*: the apparent visual magnitude of the star.
  - *spect*: the spectral type of the star, if known.
  - *color_idx*: the color index of the star, if known.
  - *bayer*: the Bayer designation -a stellar designation in which the star is identified by a Greek or Latin letter followed by the genitive form of its parent constellation's Latin name.
  - *con*: the IAU abbreviation of the parent constellation of the star.

- ❖ Nebulas: this table contains information about some of the most famous nebulas, the description of these attributes are as follows.
  - name: the name of the nebula.
  - *ra_time*: the RA position of the nebula in time notation.
  - *ra_deg*: the RA position of the nebula in degrees.
  - *dec_deg*: the Dec position of the nebula.
  - *dist*: the distance between the nebula and the earth.
  - dimensions: the apparent dimensions of the nebula.
  - radius: the approximate radius of the nebula in light years
  - con: the IAU abbreviation of the parent constellation of the nebula.

- ❖ Solar system objects: this table contains information about some of solar system objects, the description of these attributes are as follows (TheSkyLive, n.d.).
  - *name*: the name of the solar system object.
  - *ra_time*: the RA position of the solar system object in time notation.
  - *ra_deg*: the RA position of the solar system object in degrees.
  - *dec_deg*: the Dec position of the solar system object.
  - *dist*: the distance between the solar system object and the earth.
  - *mag*: the apparent visual magnitude of the solar system object.

- *mass*: the mass of the solar system object in kilograms.
- *ev*: the required escape velocity of the solar system object in km/s
- *sr*: the solar system object sidereal rotation in hours.
- *diameter*: the solar system object diameter in Kilometers.
- *density*: the density of the solar system object in gr/cm^3.
- *con*: the IAU abbreviation of the parent constellation of the solar system object.

3.1.1.3 Uploading on CleverCloud server

To make the database of Sky Guide accessible remotely, we uploaded it on an online MySQL server supported by clever cloud (CloudClever, n.d.).

Clever Cloud is a Europe-based PaaS company, that help developers deploy and run their apps with bulletproof infrastructure, automatic scaling, fair pricing and other features. Among those features, free MySQL hosting, of course with some limitations.

3.1.2 Backend

In this section we'll discuss Sky Guide desktop application backend, and some of the important points regarding the application.

3.1.2.1 Programming language used

In the development of Sky Guide application, we used Python-3.9 as the backend programming language, we chose Python because it's a simple and powerful language, and recently we have studied it, and all of Sky Guide team are familiar and have some experience with it.

3.1.2.2 Device connection

Sky Guide mount wireless communication with the application is done using a Bluetooth connection, we used a serial communication module called PySerial (PyPi, n.d.), to configure the app to communicate with the mount which will be connected to one of the OS Bluetooth COM ports.

Once the application starts, it tries to connect with the mount, and sends the appropriate angles to it, then it waits for the confirmation, if the confirmation wasn't sent back, the app will disconnect, and retry to initiate the connection again, this process is done until the application is closed by the user.

3.1.2.3 LST calculations

As the application is responsible to send the correct angels to the mount of the celestial object, it has to calculate the difference between the current LST and the object's RA, and then sends an updated angel of this object's RA to the mount, this is done continuously to ensure that the pointer finds and follows the celestial object in the sky. The calculation of the local sidereal time was taken form "Astronomical algorithms 2nd edition" by Jean Meeus, chapter 12 (Meeus, 1998).

3.1.2.4 Application threads

Sky Guide application consists of three threads: main thread, internet checking thread and mount communication thread.

In the internet checking thread, we continuously check for the internet connection, as it is needed to connect and execute queries on Sky Guide online database.

In the mount communication thread, we continuously send the Dec and the updated RA angles to the mount, thus, enabling the mount to find and track the celestial object in the sky.

3.1.2.5 UML diagram

Sky Guide backend is a three-layered architecture: Data access layer, Business layer and Application layer. Following is the UML of these layers.

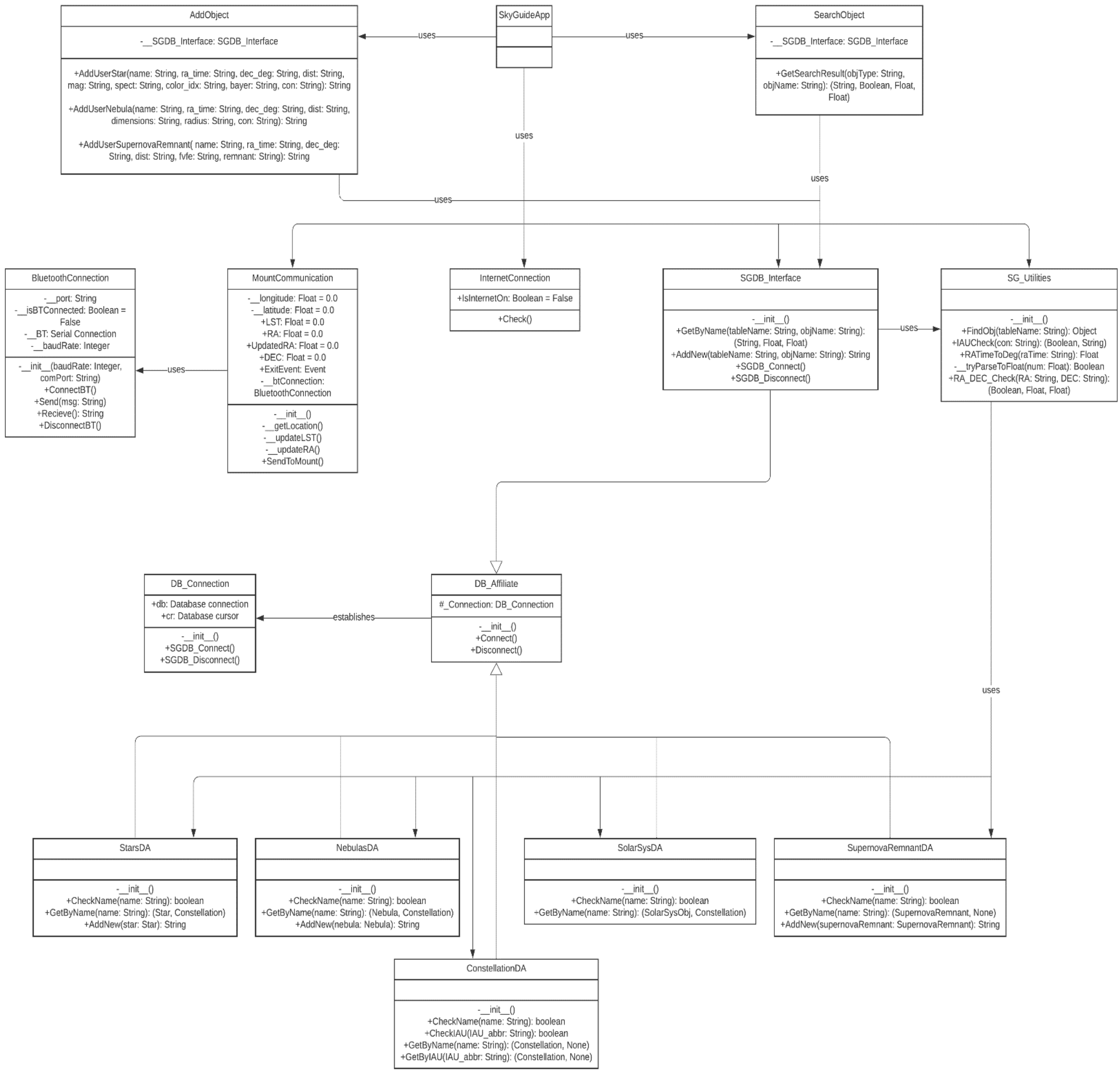


Fig 3.2: Sky Guide backend UML diagram

The “SkyGuideApp” class contains all the signals and the slots. Which will be used to connect with the frontend of the app-we didn’t add it’s details to avoid complexity in the uml design, following is the uml of the class,

SkyGuideApp
+AddObject: AddObject +SearchObject: SearchObject +SGDB_Interface: SGDB_Interface +netConnection: InternetConnection +mountCommunication: MountCommunication +NewConnectionState: Boolean = False +UserRA: Float = 0.0 +UserDEC: Float = 90.0 +internetConnected: Signal(Boolean) +starAddResult: Signal(String) +nebulaAddResult: Signal(String) +supernovaRemnantAddResult: Signal(String) +searchResult: Signal(String) +searchStatus: Signal(Boolean) +exploreResult: Signal(String)
-__init__() +InternetCheck() +addUserStar(name: String, ra_time: String, dec_deg: String, dist: String, mag: String, spect: String, color_idx: String, bayer: String, con: String) +addUserNebula(name: String, ra_time: String, dec_deg: String, dist: String, dimensions: String, radius: String, con: String) +addUserSupernovaRemnant(name: String, ra_time: String, dec_deg: String, dist: String, fvfe: String, remnant: String) +getSearchResult(objType: String, objName: String) + sendToMount_Search() +expolreObject(RA: String, DEC: String)

Fig 3.3: SkyGuideApp class UML diagram

3.1.3 Frontend

Sky Guide frontend is developed using Qt, Qt is a widget toolkit for creating graphical user interfaces as well as cross-platform applications that run on various software and hardware platforms.

Sky Guide application user interface will be simple and easy to use, it will enable the user to perform the following actions on Sky Guide database:

- Search for a celestial object by name.
- Explore specific coordinates.
- Add a new celestial object to the database.

Also, will enable the user to control the celestial coordinate that the mount is pointing at.

3.1 Hardware part

Following we discuss some important points regarding Sky Guide laser mount.

3.1.1 Device body

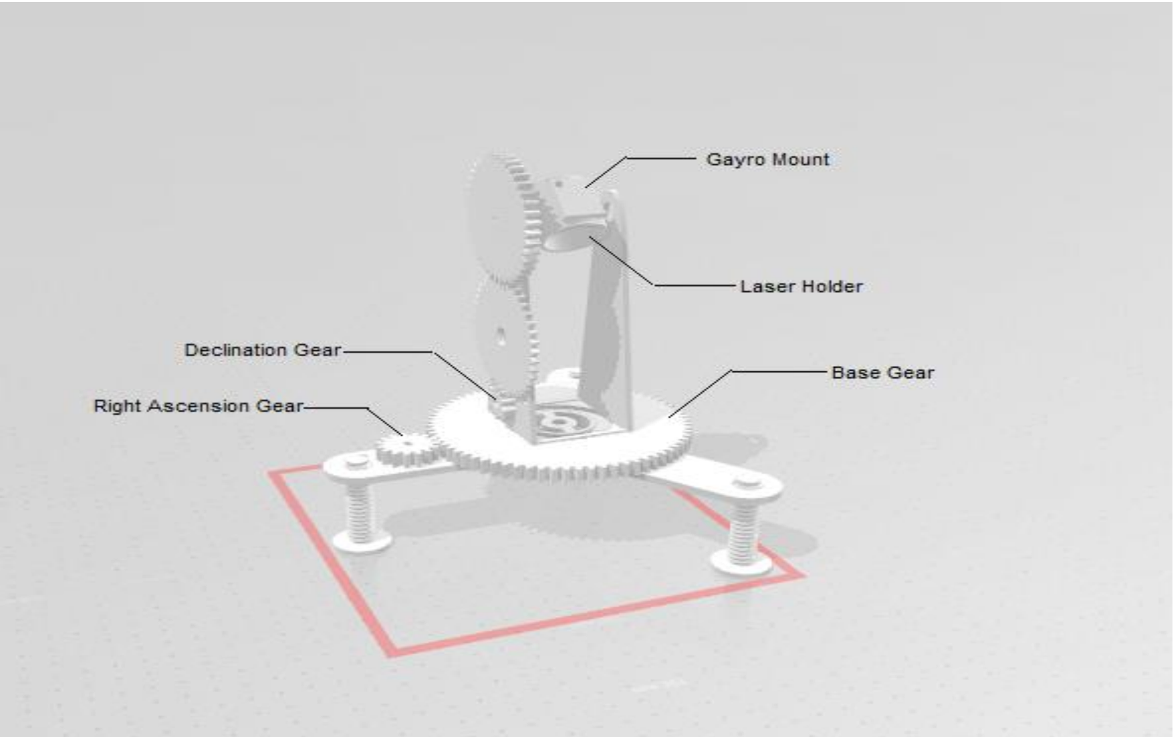


Fig 3.4: Sky Guide mount design

The design of the device body was created using Tinkercad, Tinkercad is a free-of-charge, online 3D modelling program that runs in a web browser, known for its simplicity and ease of use.

3.1.2 Used components

To execute the logic of the mount we will use the following components: Arduino uno, stepper motor, stepper motor driver, gyroscope sensor and Bluetooth module.

- *Arduino uno (x2)*: we shall be using two Arduino Unos, as master and slave,
    - The master will be responsible for receiving the RA and Dec angles from the desktop app, comparing them with the gyroscope sensor readings, and sending the appropriate control signals to the slave.
    - The slave will be responsible for receiving the control signals from the master and accordingly control the 2 stepper motors.

- *Stepper motor (28BYJ-48) (x2):* we shall be using two stepper motors to control both the right ascension gear and the declination gear.

- *Stepper motor driver (ULN2003AN) (x2)*: we shall be using two drivers to interface the slave Arduino uno with the stepper motors.

- *Gyroscope sensor (MPU6050)*: MPU6050 is a Micro Electro-mechanical system (MEMS), it consists of three-axis accelerometer and three-axis gyroscope. It helps us to measure velocity, orientation, acceleration, displacement, and other motion like features. We shall be using it as a feedback sensor representing the current positions of the stepper motors, it will be connected to the master Arduino uno and used to compare the angles sent by the app and the current positions of the stepper motor

- *Bluetooth module (HC-50)*: we have changed the module’s name to SkyGuide, and this module will be used to communicate with Sky Guide application.

3.1.3 Components’ connection

Following is the component connection for Sky Guide mount.

3.1.4 Device firmware

In Appendix A you will find the firmware of both the master and the slave Arduino uno(s).

CHAPTER 4 RESULTS AND DISCUSSION

4.1 Results

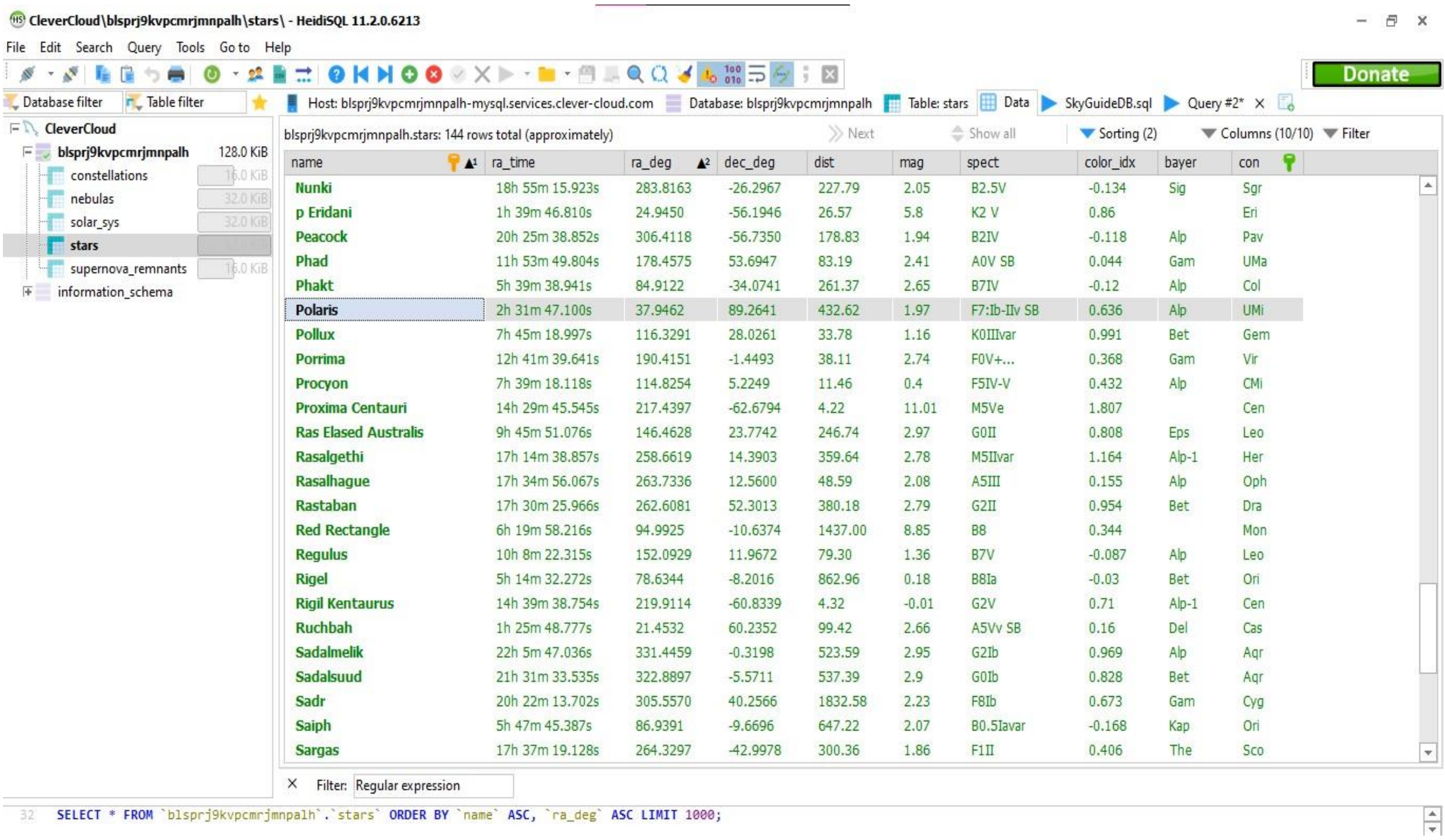
Following are the results of our project both software and hardware.

4.1.1 Software result

We have successfully built the desktop application as discussed in the previous chapter. All the source code of our application can be found on SkyGuide GitHub repository (SkyGuide-Azhar, 2021).

4.1.1.1 Database result

The following figure is a screenshot of the "stars" table in Sky Guide database -we used HeidiSQL application to create and display the database- all the SQL code of the database can be found on SkyGuide GitHub repository (SkyGuide-Azhar, 2021).



name	ra_time	ra_deg	dec_deg	dist	mag	spect	color_idx	bayer	con
Nunki	18h 55m 15.923s	283.8163	-26.2967	227.79	2.05	B2.5V	-0.134	Sig	Sgr
p Eridani	1h 39m 46.810s	24.9450	-56.1946	26.57	5.8	K2 V	0.86		Eri
Peacock	20h 25m 38.852s	306.4118	-56.7350	178.83	1.94	B2IV	-0.118	Alp	Pav
Phad	11h 53m 49.804s	178.4575	53.6947	83.19	2.41	A0V SB	0.044	Gam	UMa
Phakt	5h 39m 38.941s	84.9122	-34.0741	261.37	2.65	B7IV	-0.12	Alp	Col
Polaris	2h 31m 47.100s	37.9462	89.2641	432.62	1.97	F7:Ib-IIv SB	0.636	Alp	UMi
Pollux	7h 45m 18.997s	116.3291	28.0261	33.78	1.16	K0IIIvar	0.991	Bet	Gem
Porrima	12h 41m 39.641s	190.4151	-1.4493	38.11	2.74	F0V+...	0.368	Gam	Vir
Procyon	7h 39m 18.118s	114.8254	5.2249	11.46	0.4	F5IV-V	0.432	Alp	CMi
Proxima Centauri	14h 29m 45.545s	217.4397	-62.6794	4.22	11.01	M5Ve	1.807		Cen
Ras Elased Australis	9h 45m 51.076s	146.4628	23.7742	246.74	2.97	G0II	0.808	Eps	Leo
Rasalgethi	17h 14m 38.857s	258.6619	14.3903	359.64	2.78	M5IIvar	1.164	Alp-1	Her
Rasalhague	17h 34m 56.067s	263.7336	12.5600	48.59	2.08	A5III	0.155	Alp	Oph
Rastaban	17h 30m 25.966s	262.6081	52.3013	380.18	2.79	G2II	0.954	Bet	Dra
Red Rectangle	6h 19m 58.216s	94.9925	-10.6374	1437.00	8.85	B8	0.344		Mon
Regulus	10h 8m 22.315s	152.0929	11.9672	79.30	1.36	B7V	-0.087	Alp	Leo
Rigel	5h 14m 32.272s	78.6344	-8.2016	862.96	0.18	B8Ia	-0.03	Bet	Ori
Rigil Kentaurus	14h 39m 38.754s	219.9114	-60.8339	4.32	-0.01	G2V	0.71	Alp-1	Cen
Ruchbah	1h 25m 48.777s	21.4532	60.2352	99.42	2.66	A5Vv SB	0.16	Del	Cas
Sadalmelik	22h 5m 47.036s	331.4459	-0.3198	523.59	2.95	G2Ib	0.969	Alp	Aqr
Sadalsuud	21h 31m 33.535s	322.8897	-5.5711	537.39	2.9	G0Ib	0.828	Bet	Aqr
Sadr	20h 22m 13.702s	305.5570	40.2566	1832.58	2.23	F8Ib	0.673	Gam	Cyg
Saiph	5h 47m 45.387s	86.9391	-9.6696	647.22	2.07	B0.5Iavar	-0.168	Kap	Ori
Sargas	17h 37m 19.128s	264.3297	-42.9978	300.36	1.86	F1II	0.406	The	Sco

Fig 4.1: Stars table from Sky Guide database

4.1.1.2 User experience result

Screenshot of the SkyGuide application Search page

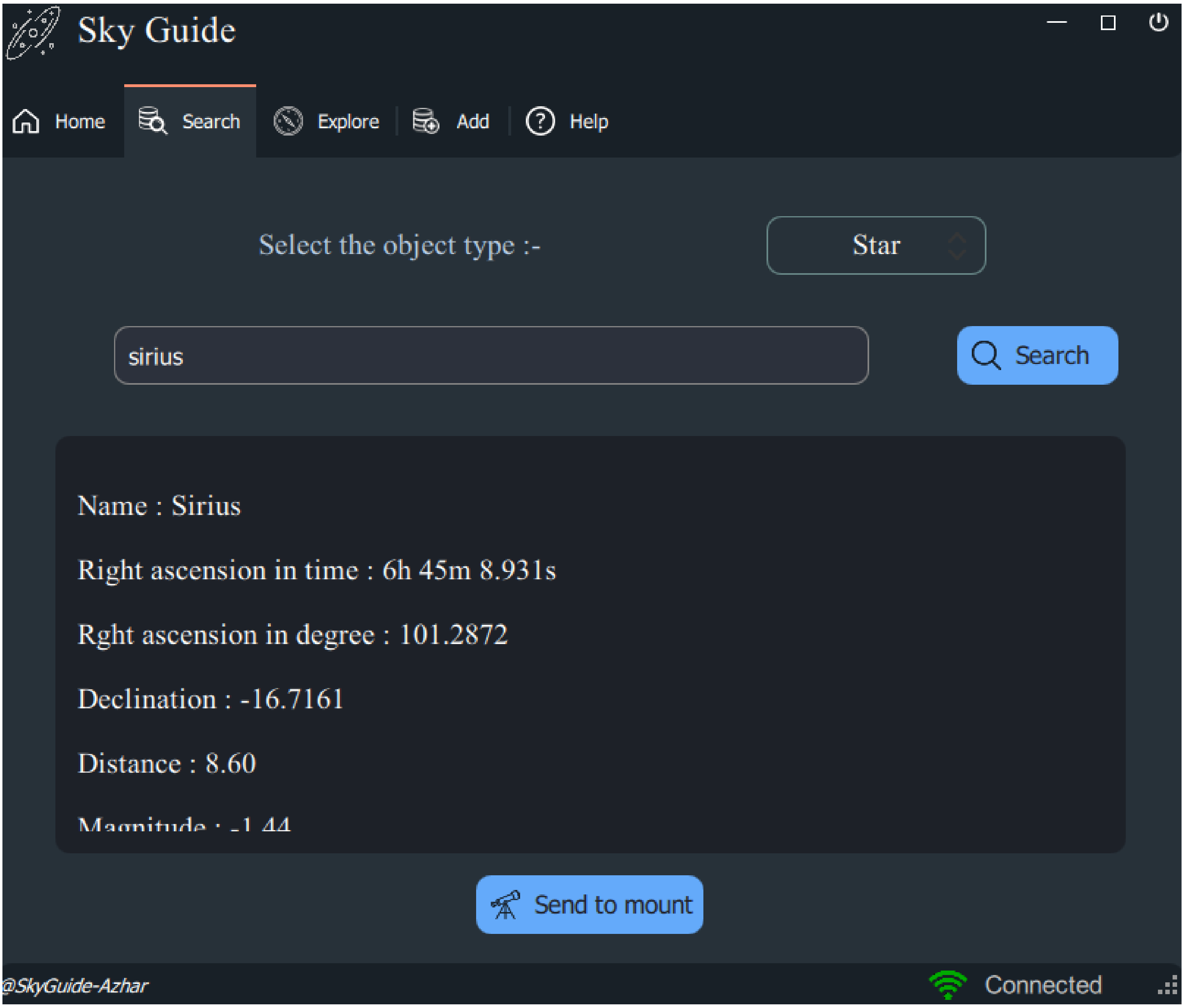


Fig 4.2: Example of search page in Sky Guide app

Screenshot of the SkyGuide application Expolre page

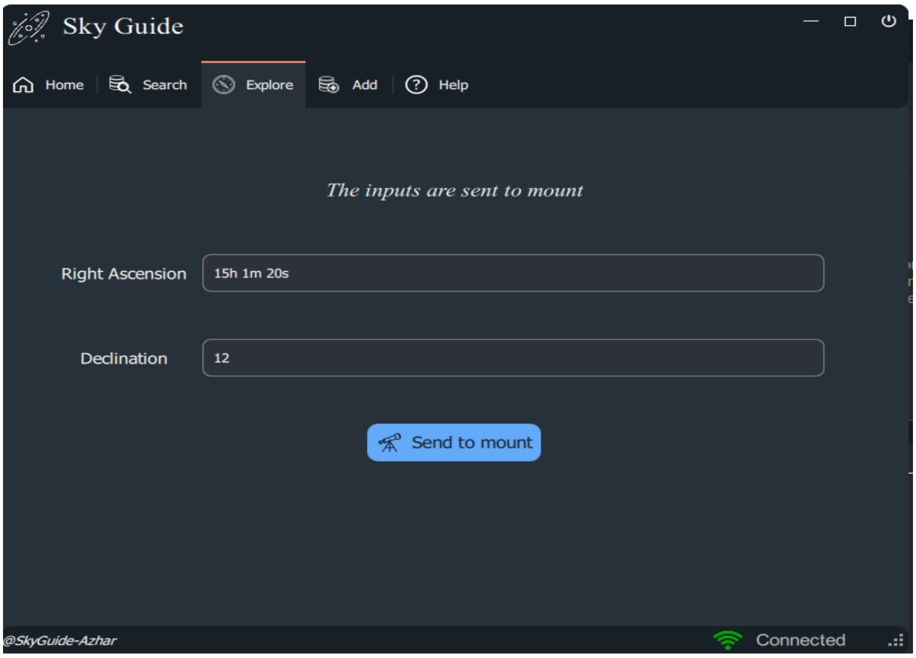


Fig 4.3: Example of explore page in Sky Guide app

Screenshot of the SkyGuide application Add page

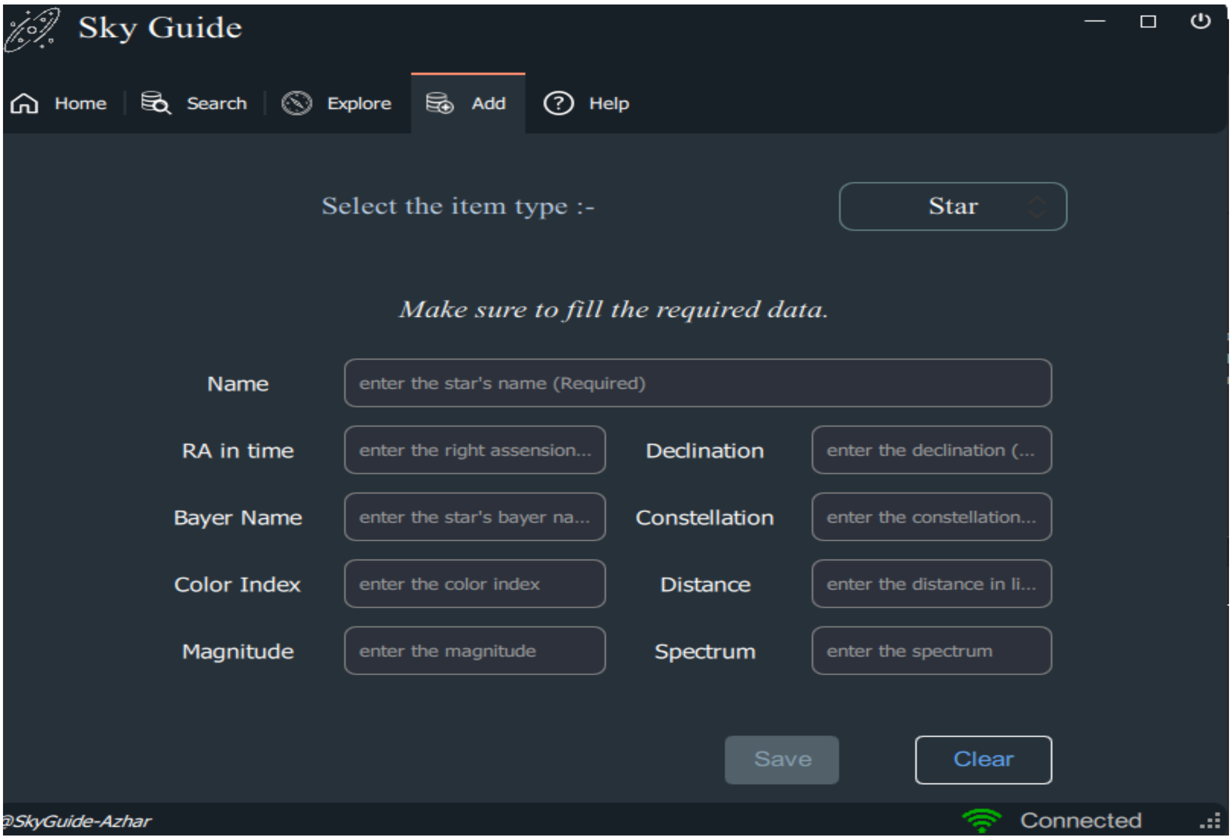


Fig 4.4: Example of add page in Sky Guide app

4.1.2 Hardware result

We have printed the design and connected the components, following are images of the device.

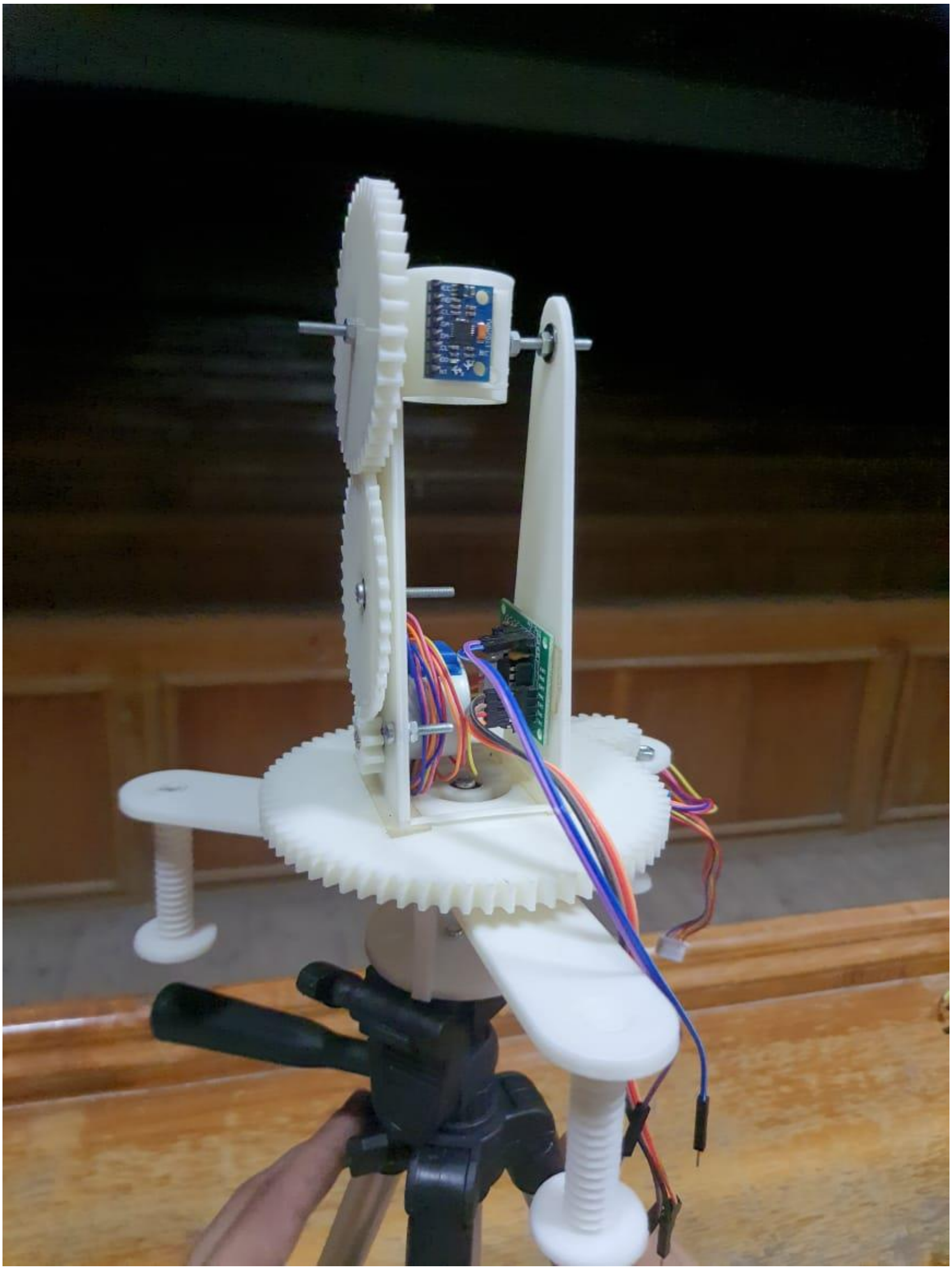


Fig 4.5: Image 1 of Sky Guide device

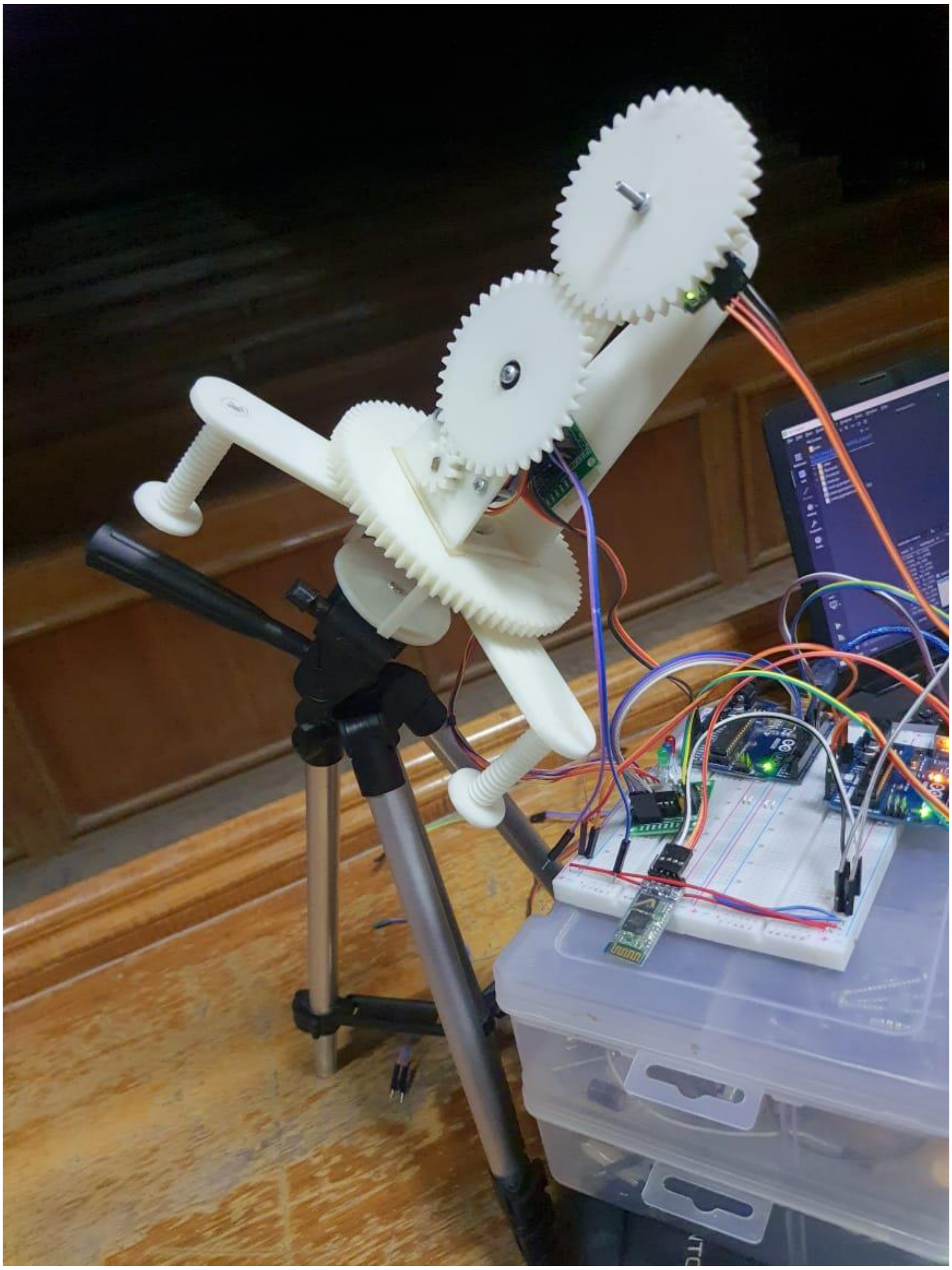


Fig 4.6: Image 2 of Sky Guide device

4.2 Discussion

We have successfully uploaded Sky Guide database on Clever Cloud online MySQL host, and executed the application and mount connection using Bluetooth as intended, however, the components' connection was very challenging, and we could not improve it to make the mount smooth as intended.

CHAPTER 5 CONCLUSION AND FUTURE WORK

5.1 Conclusion

Sky Guide device was able to solve the problem that we have described in the previous chapters, we were able to:

- Enable the user to select a celestial object by its name and send its coordinates to the mount to point at that object.
- Enable the user to add a new celestial object to Sky Guide database in a robust fashion and access it remotely later.
- Enable the user to explore a particular coordinate on the celestial sphere.
- Enable the user to track the celestial object across the sky.

5.2 Future work

Following is our recommendation for the future work on Sky Guide project.

- ❖ Regarding software
  1. Enable the user to view the content of a table in Sky Guide database to choose from.
  2. Enable the user to update the solar system table data from the app on command.
  3. Substitute the Bluetooth connection with a more robust wireless connection.
  4. Enable the user to view the current positions of the mount from Sky Guide application.
- ❖ Regarding hardware
  1. Build a mount that will be able to carry a telescope or a camera and make them point at the sky.
  2. Build a firmware based on a microcontroller other than Arduino.
  3. Solve the issues of wires connection.
  4. Add the tracking feature to the hardware, to make it less error-prone.

REFRENCES

CloudClever. (n.d.). *Clever Cloud*. Retrieved from https://www.clever-cloud.com/en/

ERDPlus. (n.d.). *ERDPlus*. Retrieved from https://erdplus.com/

Ford, D. (2014). *The Observer's Guide to Planetary Motion.* New York: Springer.

Meeus, J. (1998). *Astronomical Algorithms* (2 ed.). Virginia: Wilmann-Bell, Inc.

Nash, D. (2011). *HYG-Database*. Retrieved from Github: https://github.com/astronexus/HYG-Database

PyPi. (n.d.). *PySerial*. Retrieved from https://pypi.org/project/pyserial/

SkyGuide-Azhar. (2021). *SkyGuide*. Retrieved from Github: https://github.com/SkyGuide-Azhar/SkyGuide

TheSkyLive. (n.d.). *TheSkyLive*. Retrieved from https://theskylive.com/

Wikipedia. (n.d.). *IAU_designated_constellations*. Retrieved from https://en.wikipedia.org/wiki/IAU_designated_constellations

APPENDICES

Appendix A: Arduino Master Code:

```
#include <Wire.h>
#include <MPU6050.h>
#include <SoftwareSerial.h>

#define RA_Stop  7
#define RA_CW    6
#define RA_CCW   8

#define DEC_Stop 10
#define DEC_CW   9
#define DEC_CCW  11


SoftwareSerial BTSerial(3, 4); // RX , TX

MPU6050 mpu;

boolean RAStopped = false;
boolean DECStopped = false;

unsigned long timer = 0;
float timeStep = 0.01;


double pitch, roll;

double in_DEC, in_RA;  //variable to store the user input DEC and RA

void setup()
{
    Serial.begin(115200);
    BTSerial.begin(9600);

    pinMode(RA_Stop,OUTPUT);
    pinMode(RA_CW,OUTPUT);
    pinMode(RA_CCW,OUTPUT);
    pinMode(DEC_Stop,OUTPUT);
    pinMode(DEC_CW,OUTPUT);
    pinMode(DEC_CCW,OUTPUT);

    while(!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G))delay(500);

    mpu.calibrateGyro();
    mpu.setThreshold(3);
}

void loop()
{

    recvdata();

    timer = millis();
    Vector norm = mpu.readNormalizeGyro();


    roll  = roll + norm.YAxis * timeStep;
    pitch = pitch + norm.XAxis * timeStep;

    Serial.print(" roll = ");
    Serial.print(roll);
    Serial.print(" Pitch = ");
    Serial.println(pitch);

    DecCheck();
    RACheck();


    Serial.print(" RA = ");
    Serial.print(in_RA);
    Serial.print(" DEC = ");
    Serial.println(in_DEC);

    delay(abs((timeStep*1000) - (millis() - timer)));//timer for the gyro.
}
```

```
void recvdata()
{
    Serial.println(BTSerial.available());
    if (BTSerial.available())
    {
        String UserInput = BTSerial.readString();
        String UserRA, UserDec;

        Serial.println(UserInput);

        for (int i = 0; i < UserInput.length(); i++)
        {
            if (UserInput.substring(i, i+1) == ",")
            {
                UserRA = UserInput.substring(0, i);
                UserDec= UserInput.substring(i+1);
                break;
            }
        }

        in_DEC = 90 - UserDec.toFloat(); // "90 - " cuz the pitch's zero is 90(polaris)
        in_RA  = UserRA.toFloat();

        BTSerial.println("1");
    }
}


void DecCheck()
{

    if(floor(pitch) == floor(in_DEC))
        digitalWrite(DEC_Stop,HIGH);
    else
        digitalWrite(DEC_Stop,LOW);

    if(floor(pitch) < floor(in_DEC))
        digitalWrite(DEC_CW,HIGH);
    else
        digitalWrite(DEC_CW,LOW);

    if(floor(pitch) > floor(in_DEC))
        digitalWrite(DEC_CCW,HIGH);
    else
        digitalWrite(DEC_CCW,LOW);

}


void RACheck()
{

    if(floor(roll) == floor(in_RA))
        digitalWrite(RA_Stop,HIGH);
    else
        digitalWrite(RA_Stop,LOW);

    if(floor(roll) < floor(in_RA))
        digitalWrite(RA_CW,HIGH);
    else
        digitalWrite(RA_CW,LOW);

    if(floor(roll) > floor(in_RA))
        digitalWrite(RA_CCW,HIGH);
    else
        digitalWrite(RA_CCW,LOW);
}
```

Slave Code:

```
#include <AccelStepper.h>
#include <Wire.h>

#define HALFSTEP 8

#define motorPin1  2       // IN1 on the ULN2003 driver 1 (RA)
#define motorPin2  3       // IN2 on the ULN2003 driver 1 (RA)
#define motorPin3  4       // IN3 on the ULN2003 driver 1 (RA)
#define motorPin4  5       // IN4 on the ULN2003 driver 1 (RA)

#define motorPin5  A0      // IN1 on the ULN2003 driver 2 (DEC)
#define motorPin6  A1      // IN2 on the ULN2003 driver 2 (DEC)
#define motorPin7  A2      // IN3 on the ULN2003 driver 2 (DEC)
#define motorPin8  A3      // IN4 on the ULN2003 driver 2 (DEC)

#define RA_Stop  7         // Master RA stop control signal
#define RA_CW    6         // Master RA CW control signal
#define RA_CCW   8         // Master RA CCW control signal

#define DEC_Stop 10        // Master DEC stop control signal
#define DEC_CW   9         // Master DEC CW control signal
#define DEC_CCW  11        // Master DEC CCW control signal

AccelStepper RA_Stepper  (HALFSTEP, motorPin2, motorPin4, motorPin3, motorPin1);
AccelStepper DEC_Stepper (HALFSTEP, motorPin6, motorPin8, motorPin7, motorPin5);


boolean RA_Stopped = false;
boolean DEC_Stopped = false;

void setup()
{
    RA_Stepper.setMaxSpeed(1000.0);
    DEC_Stepper.setMaxSpeed(1000.0);

    pinMode(DEC_Stop,INPUT);
    pinMode(DEC_CW,INPUT);
    pinMode(DEC_CCW,INPUT);

    pinMode(RA_Stop,INPUT);
    pinMode(RA_CW,INPUT);
    pinMode(RA_CCW,INPUT);
}

void loop()
{
    DEC_MotorUpdate();
    RA_MotorUpdate();

    if(RA_Stopped==false)
        RA_Stepper.run();

    if(DEC_Stopped==false)
        DEC_Stepper.run();

}


void RA_MotorUpdate()
{
    if(digitalRead(RA_Stop)==HIGH)
        RA_Stopped = true;
    else if(digitalRead(RA_CW)==HIGH)
    {
        RA_Stepper.setSpeed(500);
        RA_Stopped = false;
    }
    else if(digitalRead(RA_CCW)==HIGH)
    {
        RA_Stepper.setSpeed(-500);
        RA_Stopped = false;
    }
}


void DEC_MotorUpdate()
{
    if(digitalRead(DEC_Stop)==HIGH)
        DEC_Stopped = true;
    else if(digitalRead(DEC_CW)==HIGH)
    {
        DEC_Stepper.setSpeed(500);
        DEC_Stopped = false;
    }
    else if(digitalRead(DEC_CCW)==HIGH)
    {
        DEC_Stepper.setSpeed(-500);
        DEC_Stopped = false;
    }
}
```