Satellite Tracker utilizing SatPC32 software and an Arduino Microprocessor

Christian Pack
Tickle College of Engineering
University of Tennessee, Knoxville
Tennessee, U.S.A
chrspack@vols.utk.edu

***Abstract*— Satellites and antenna communication rely on clear access to the electromagnetic waves that are being transmitted and received. This requires antennas to correct themselves towards satellites that may be orbiting faster or slower than earth's rotational velocity. Computer software such as SatPC32 provide two-line element data for tracking satellites and providing commands to a variety of rotor interfaces. Using this software in conjunction with an Arduino Microprocessor, a servo motor, and a stepper motor, a satellite tracking device can be made. The Arduino microprocessor receives and converts the incoming serial data into rotation commands for the servo and stepper motors. These commands allow for the proper orientation of the device in both the azimuth (stepper) and elevation (servo) axes towards orbiting satellites. This paper explores the relationship between using the SatPC32 software and an Arduino Microprocessor to create a compact satellite tracker and the many applications that may benefit from this device.**

Keywords*— *Arduino microprocessor, servo motor, stepper motor, elevation, azimuth, amateur radio, perturbation models, doppler correction, SatPC32

***Abbreviations*—**

SGP4: Standard General Perturbations Version 4
SDP4: Simplified Deep-space Perturbations Version 4
LEO: Low Earth Orbit
TLE: Two-line element set.
DDE: Dynamic Data Exchange
CAT: Computer Aided Transceiver

I. Introduction

After the discovery of radio waves in 1888, radio communication technology had a huge increase in interest and research. Amateur radio, or ham radio, became a popular pastime beginning in the early 1900s and led to technical advances towards the late 1900s [1]. Namely, researcher and scientist Erich Eichmann, or DK1TB, developed a satellite tracking software that utilizes radio technology titled SatPC32. This software essentially allows for fast and precise tracking of satellite systems using perturbation models and can interface with rotors for satellite tracking. This paper will explore using the SatPC32 software in conjunction with a servo and stepper motor to create a rotating antenna that orients towards orbiting satellites.

II. Materials

A. Hardware Overview

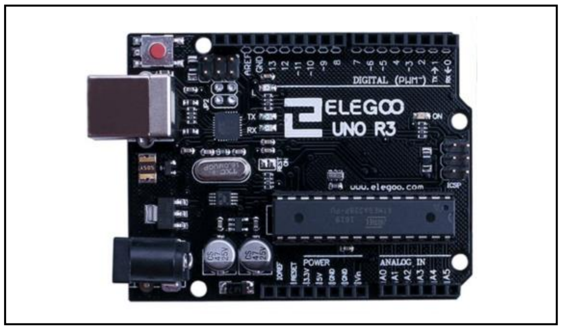


Figure 1. Picture of Arduino Board

Arduino Microprocessor: A microcontroller board that can be programmed to control electronic devices, receive input from various sensors, and communicate with other devices. For this specific project, the Arduino board is used to control a stepper and servo motor while receiving communicated input from a laptop.

Figure 2. Picture of Stepper Motor and ULN2003 Motor Driver

Stepper Motor & ULN2003 Motor Driver (Azimuth Axis): A stepper motor is a type of motor that can rotate in precise steps. This is especially useful for obtaining the correct orientation as opposed to an inexact DC motor. The stepper motor is used for rotating along the azimuth axis, or across the horizon. The ULN2003 motor driver is a chip that provides power and control signals to the stepper motor, allowing it to move in a specific direction and angle. The motor driver provides a higher current gain than a single transistor could; it also allows a microcontroller's low voltage low current output to drive a high current stepper motor.

Servo Motor (Elevation Axis): A servo motor is a type of motor that can rotate to a precise angle and hold that position, making it useful for controlling the elevation axis of this device. It can be pivoted in angles ranging from 0 degrees to 180 degrees.

Figure 3. Picture of Servo Motor

Power Supply Module: A device that provides electrical power to other components or devices. For this project, it receives an AC input from a power adaptor and converts it into a stable DC output voltage. On this specific power supply module, it allows for two customizable output voltages: 5V or 3.3 V. The servo motor requires a 3.3V input while the stepper motor and driver requires a 5V input, so this module is useful in mitigating the difference.

Figure 4. Power Supply Module

Breadboard: A board with a grid of conductive holes used for prototyping and testing electronic circuits. This board is used so that components can be connected and disconnected without the need for soldering.

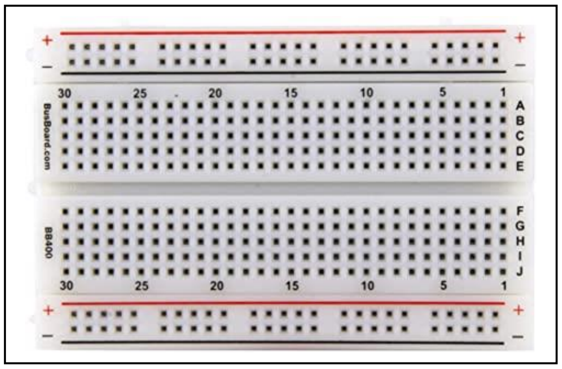


Figure 5. Breadboard

16x2 LCD: A liquid crystal display with 2 rows of 16 characters each, used for displaying text or simple graphics. For this project, the LCD will be displaying the current orbital location of the satellite as well as the azimuth and elevation axes.

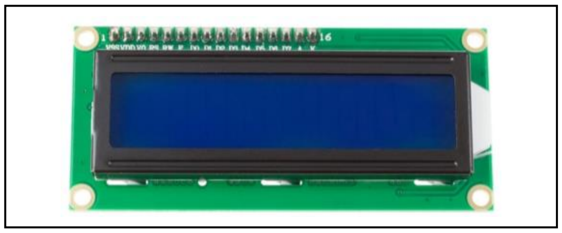


Figure 6. 16x2 Liquid Crystal Display used.

3D Printed Design: A three-dimensional designed piece was used to mount the servo on top of the stepper motor.

Figure 7. Fusion360 software used to create 3D objects.

B. *Software Overview*

SatPC32 [3]: SatPC32 is a software program designed for communication with amateur radio satellites. It provides tracking information and real-time visualization of the satellite's position, as well as automated doppler frequency correction and antenna steering.

Yaesu GS-232A Emulation [3]: This emulation method is the setting established on the SatPC32 software for communication with our Arduino board. This protocol transcribes commands for rotor controllers that are used to adjust the direction of an antenna. By emulating that we have a Yaesu GS-232A (a rotor controller) we can receive commands from SatPC32 and interpret them for a stepper and servo motor rather than an actual rotor.

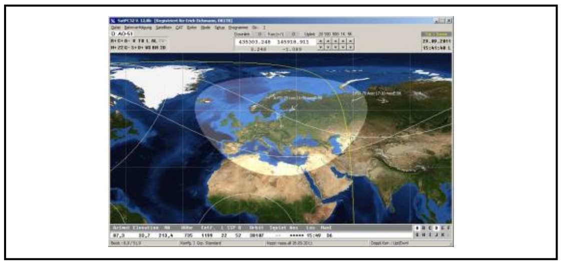
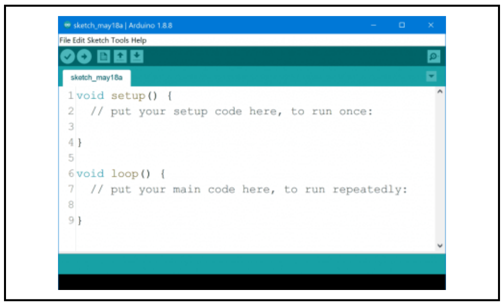


Figure 8. SatPC32 Software Interface

Arduino IDE [4]: The Arduino Integrated Development Environment (IDE) is software used to seamlessly communicate with the Arduino environment. This application includes a text editor for writing code, a compiler for converting code, and a firmware uploader for transferring code. This is the main application used for communication between the PC and the Arduino Microcontroller.

Arduino Libraries [4]: Since we are using a few electrical components for our circuit, we need the proper libraries to control them. More specifically, the Servo.h, AccelStepper.h, and LiquidCrystal.h libraries should be installed.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 9. Arduino IDE Interface

C. *Satellite Tracker Concept*

As previously mentioned, this paper explores combining both satellite tracking software technology with a mechanical rotor to create a model antenna that orients itself towards orbiting satellites.

The SatPC32 software allows for retrieval of satellite location using radio technology and can translate that data into rotor controls for a variety of antenna brands and interfaces. For this project, the Yaesu GS-232A rotor interface was chosen for the driver program; this driver formats the data into usable data exported out of a computer's serial port. The reason the Yaesu GS-232 was chosen was because it exports data that is the most compatible for an Arduino microprocessor. This data can be utilized with a little bit of code manipulation for rotating our homemade antenna in place of an expensive Yaesu Antenna.

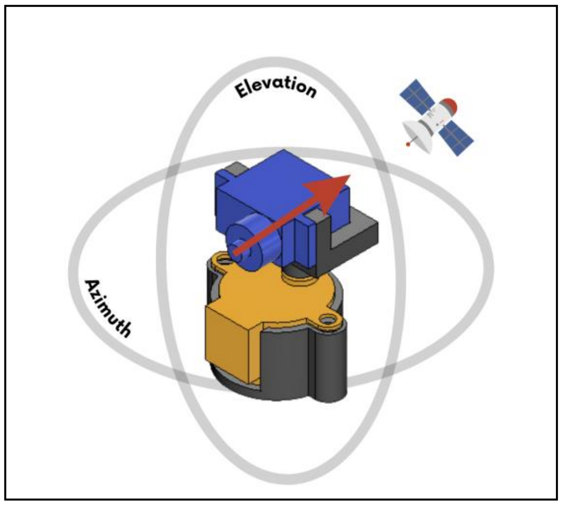


Figure 10. Rendering of Stepper and Servo motors being used as Azimuth and Elevation axes respectively.

As the data is retrieved and sent to the PC, the Arduino sketch calculates the proper degree of turn for the servo as well as the correct number of steps for the stepper motor needed for proper orientation. The servo motor is aligned with the elevation axis to provide 180 degrees of freedom and the stepper motor is aligned with the azimuth axis to provide 360 degrees of freedom. The azimuth axis operates along the horizon and the elevation axis operates perpendicular to the horizon as seen below [5].

D. *Orbital Perturbation Models*

Together, the servo and stepper motors allow for a complete range of freedom when pointing towards the passing satellite. By pointing towards the projected satellite's orbit, the antenna can be used for a more precise data collection point compared to the PC radio receiver, or Computer Aided Transceiver. The availability to a satellite during orbit can be mitigated by updating the antenna orientation to prevent a doppler shift. A doppler shift is a change in a radio wave frequency in relation to a wave source moving relative to an observer, essentially a satellite moving away from an antenna [6]. As the satellite orbits, a satellite tracker device can help mitigate the doppler shift through orienting itself to receive a proper radio frequency.

The satellite's location can be predicted by using a mathematical concept called simplified perturbation models. Perturbations, in astronomy, is a deviation in the motion of a celestial object caused either by the gravitational force of a passing object or by a collision with it [7]. For example, a satellite's orbit around earth is not a direct orbit as there are interferences from the Earth's gravitational field, the effects of the Sun and Moon, and the atmospheric drag. By using highly computational and derived formulas, scientists have come up with perturbation models that can predict a satellite's orbit around Earth that accounts for these interferences.

These models calculate the orbital vectors of satellites relative to the Earth-centered coordinate system and deliver them in two-line element sets. Beginning in the 1900s, these perturbation models have allowed space agencies and governments to accurately track satellites and even discover new planets due to deviations of orbits.

There are two types of perturbation models used for satellites that depend on the orbital period of a satellite, or the time it takes to orbit the earth [8]. The orbital period increases with altitude due to a decrease in gravity, as well as the increased circumference of travel. Simplified General Perturbations (SGP) models are used for objects with an orbital period of less than 225 minutes, while Simplified Deep Space Perturbations (SDP) models are used for objects with an orbital period greater than 225 minutes. The SGP4 and SDP4 models are used to calculate the satellite position regardless of the dead spots that we encounter as aforementioned. SatPC32 utilizes radio technology to communicate with these satellites while simultaneously utilizing SGP4 and SDP4 perturbation models to calculate the orbits of amateur radio satellites. SatPC32 accurately predicts the satellite's position and movement in real-time which is the first component of commanding the satellite tracker [3].

III. METHODS

A. *Wiring*

After acquiring all the material previously mentioned in the hardware section, it is now possible to assemble. You may be able to get away with powering the servo and stepper motor, however, since this is designed to stay on for long periods, I decided to add a power supply module to regulate the voltage more efficiently.

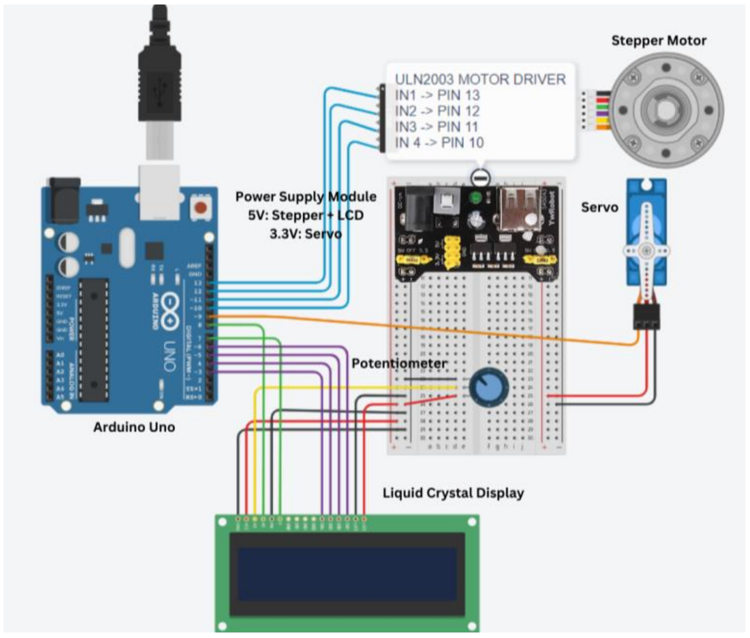


Figure 11. Wiring Diagram for the Arduino Satellite Tracker Project

1. Plug the power supply module onto a breadboard. Adjust one power column to supply 5V and the other supply 3.3V.
2. Connect the male pins from the Stepper motor to the female pins on the ULN2003 Motor Driver.
3. Connect the ULN2003 Motor Driver pins to the Arduino board in the following combination:
    a. ULN2003 pin IN1 to Arduino pin 13
    b. ULN2003 pin IN2 to Arduino pin 12
    c. ULN2003 pin IN3 to Arduino pin 11
    d. ULN2003 pin IN4 to Arduino pin 10
4. Connect the 5V and GND from the ULN2003 Motor Driver Pins to the 5V power column of the breadboard. The stepper motor is now wired.
5. Connect the Servo motor 3.3V and GND to the 3.3V column of the breadboard.
6. Connect the Servo signal pin to pin ~9 (PWM) on the Arduino Board. The servo motor is now wired.
7. Connect the LCD display pins to the Arduino board and breadboard in the following combinations. All six Arduino pins must be digital outputs.
    a. LCD pin 1 (VSS) to GND on power column.
    b. LCD pin 2 (VDD) to 5V on power column.
    c. LCD pin 3 (V0) to potentiometer's middle signal pin, and the potentiometer's outer pins to GND and 5V respectively.
        i. The potentiometer is used for controlling the backlit of the LCD.
    d. LCD pin 4 (RS) to Arduino pin 8
    e. LCD pin 5 (RW) to GND on power column
    f. LCD pin 6 (E) to Arduino pin 7
    g. LCD pins 11 to 14 (DB4-DB7) to Arduino pins 3-6.
    h. LCD pin 15 (A) to 5V on power column.
    i. LCD pin 16 (K) to GND on power column.
8. Connect the 5V and GND from the LCD display to the 5V power column of the breadboard.

The wiring for the servo, stepper, LCD display, and potentiometer are now all complete.

B. *Assembly*

When initializing the rotor, it must be orientated North when starting the azimuth rotations. For this reason, a small compass was attached to the project to help reference North. A small plastic container was used to house the parts and protect the connections from outside interference. For the rotor mechanism, I designed, and 3-D printed a mount for the Servo motor to attach to the stepper motor's spindle.

C. *SatPC32 Configuration*

Download and install SatPC32 using the setup.exe provided from the creator's website, www.dk1tb.de/indexeng.htm. For our setup, Knoxville, TN was used as the observer location with longitude of -83.9207 (West) and a latitude of +35.9606 (North). These coordinates use the station grid EM85AX. Make the following changes to the sections along the top of the program window.

- Satellites:
    - Select the source file nasa.all. Press "Update Keps" at the lower bottom of the screen and

select the appropriate nasa.all file to update with Celestrak data.
    - Select the desired satellites (192 available) and SatPC32 stores them in Letters A-X.
    - There are also files that can be used in place of nasa.all that can track other LEO objects such as cable and internet satellites or various CubeSATs.
    - Press the OK button to save changes.
- Setup:
    - Observer: Calculate the proper altitude of your location. For Knoxville, TN it is 150 m. Other Values should be automatically included when prompted for coordinates at start up.
    - Rotor Setup: Select Yaesu GS-232 rotor interface/controller. Make the following changes and press store then OK.
        - LPT = 3 (COM3), Delay = 30, Turning point = N, Min. Elevation = -3.
        - Update: at sat. pos. change, Max Elevation = 90 deg., Pos Change = 5 deg.
        - Check the box gain related.
    - Radio Setup: Make sure the radio is not using the COM3 port we will be using for serial output. Put COM port 0 and press store.
- ServerSDX:
    - When an Arduino board is connected, ServerSDX will launch since a COM port is detected. Press Setup, and select COM Port #3, and a Baud rate of 9600. Make sure that Az./El is selected. Click Store and exit.

Once finished with the above changes, close and reopen SatPC32 to initiate the changes. The new Kepler data as well as COM port configurations should have SatPC32 ready to interact with the Arduino.

D. Sketch

Initialization: Include Servo, AccelStepper, LiquidCrystal libraries. Initialize servo, stepper, and LCD objects with the correct Arduino pin numbers. Initialize global variables to be used in later functions.

Setup Function: attach the servo to pin 9, initialize stepper parameters, and begin the Serial port. Also run the functions set_azimuth and set_elevation to 0 to reset to initial positions. Before the stepper is engaged, ensure that the servo is positioned towards North.

Loop Function: Check if there is serial data available, if so, read it. If there is not any serial data available and it has been more than 10 seconds since that last input of data, reset the rotor and print that no satellite is above the horizon for tracking. The Yaesu-GS232 interface does not send any data unless a satellite has positive elevation (above the horizon).

Read_data Function: this manipulates the input data from SatPC32 which is exported as “Waaa eee.” We extract the “aaa” and “eee” into buffer strings that can then be turned into integers for later use. Run the set_azimuth and set_azimuth functions using the new integer values.

Set_azimuth Function: use the map() function to convert the decoded positioning information from a range of degrees (0-360) to a range of steps (0-2048). Update the azimuth position on the lcd screen.

Set_elevation Function: Use the decoded integer and write it to the servo motor. The servo should initially be horizontal when at 0 degrees. Update the elevation position on the lcd screen.

IV. RESULTS AND DISCUSSION

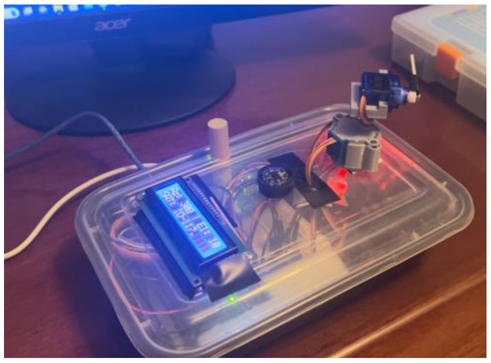


Figure 12. Finished project housed in plastic container.

SatPC32 calculates the orbital path of satellites and connects with many radio communications interfaces to allow rotor control for antennas. In our case, our homemade antenna utilized a servo for elevation and a stepper for azimuth orientation towards passing satellites. Orienting myself towards true north and utilizing the global map on SatPC32 the elevation and azimuth commands seemed to be accurate when compared to various satellite positions. The LCD screen displayed the current azimuth and elevation commands in degrees as well as the current satellite it is tracking.

Although this is a very small-scale prototype, it would be interesting to pursue this project further into a large-scale antenna with radio communications and various improvements. Our prototype does not communicate to a satellite for direct communication but rather utilizes its location for tracking through SatPC32 and perturbation models. With a functioning antenna attached, we can then introduce radio communications to our device with an accurate, auto-adjusting reception. In addition to this, due to the simplicity of the circuit, the power supply module could very easily be adjusted to run off a battery pack that can be seated within the plastic container for mobile use. Since SatPC32 runs off downloaded Kepler data, a computer could theoretically use this antenna project to connect to a robust internet satellite when overhead.

The reason I pursued this project was to understand the relationship between satellites and their antenna counterparts. Through this endeavor I discovered their complex communication systems as well as the history within amateur radio [1]. With the use of SatPC32, this rotor device can utilize satellites to their full potential when within our range of elevation.

Satellites have a variety of purposes such as earth observation, communication, and navigation. Relaying potential weather hazards or patterns help humans prepare for the worst. The National Ocean and Atmospheric Administration operates a total of 17 satellites all of which provide crucial weather information that can help mitigate hurricanes, understand peak climate conditions for farmers, or even help navigate airliners around foggy conditions [9]. In addition to weather information, satellites can also provide key communications nodes for telephones, television, internet, radio broadcasting, amateur radio, and military personnel [10].

Satellites are important components within our daily lives, and they are credited with supporting many key aspects of any nation's infrastructure. A plethora of companies utilize satellites to provide services to society. To name a few in the U.S., Sirius XM provides satellite radio services for vehicle applications, DirecTV and Dish provide satellite television streaming services, and even SpaceX provides internet services through a total of 3,000 satellites in LEO [11]. This is why this project is an important piece of technology that is beneficial to not only amateur radio enthusiasts, but to the proper functioning of society.

V. CONCLUSION

By utilizing SGP4 models and SatPC32 software to orient a mock antenna towards a satellite, we can improve the reception for capturing radio electromagnetic waves. These waves transmit valuable data that can be used for the purposes mentioned above. There is a vast array of applications that satellites can be used for, and the proper connection to them through satellite tracking is crucial in facilitating those efforts. This project oriented a mock antenna towards a satellite that could be useful in extreme measures. With self-sustaining power supply and downloaded satellite orbit patterns, a laptop could be used remotely for radio transmissions or internet connections that utilize the satellites overhead. Even in less drastic cases, this device could also be used for a better picture for satellite television or a clearer reception for satellite music. Regardless, the applications are limitless and eventually I hope to scale this project into a larger, radio functional antenna communications system.

ACKNOWLEDGMENT *(Heading 5)*

I would like to acknowledge a few key contributors to the development and inspiration of this project. Erich Eichmann, the creator of SatPC32, developed important software that made satellite tracking and antenna control possible. He has also made significant contributions to the amateur radio community through his DK1TB callsign and satellite station. Glen Popiel (KW5GP), the author of Arduino for Ham Radio, was also a key inspiration to this project through his book. This book provided the initial Arduino sketch for a satellite tracker that is like this project. A few other space enthusiasts created variations of this project on Instrucables.com which provided some insight into the missing gaps between my design and Arduino sketch that Popiel did not cover. Alex Chang developed a similar project utilizing an Arduino MKR 1000 with Wi-Fi, and Russell Grokett developed one with an ESP8266 microprocessor, both of which had additional considerations in their sketch code that I utilized.

REFERENCES

[1] Encyclopedia.com, "Heinrich Hertz Produces and Detects Radio Waves in 1888." Encyclopdia.com, https://www.encyclopedia.com/science/encyclopedias-almanacs-transcripts-and-maps/heinrich-hertz-produces-and-detects-radio-waves-1888, (accessed May 1, 2023).

[2] T. Olujinmi, "What's in Your Arduino Starter Kit?", Make Use Of, https://www.makeuseof.com/tag/whats-arduino-starter-kit-arduino-beginners/, (accessed May 1, 2023).

[3] E. Eichmann,"SatPC32", DK1TB, http://www.dk1tb.de/indexeng.htm, (accessed May 1, 2023).

[4] Arduino, "Arduino Integrated Development (IDE) v1," Arduino, https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics, (accessed May 1, 2023).

[5] Celestis Memeroail Spaceflights, "What are the 'Azimuth and Elevation' of a Satellite?" Celestis, https://www.celestis.com/resources/faq/what-are-the-azimuth-and-elevation-of-a-satellite/, (accessed May 1,2023).

[6] R. Christy, "Doppler Curves in Satellite Tracking," Orbital Focus UK, https://www.orbitalfocus.uk/Tracking/Doppler.php#:~:text=Measuring%20the%20Doppler%20Shift%20is,to%20the%20actual%20transmission%20frequency., (accessed May 1, 2023).

[7] Britannica, "Perturbation," Britannica, https://www.britannica.com/science/perturbation-astronomy, (accessed May 1, 2023).

[8] DBpedia, "Simplified Perturbation Models," DBpedia, https://dbpedia.org/page/Simplified_perturbations_models, (accessed May 1, 2023).

[9] National Environmental Satellite, Data, and Information Science, "Current Satellite Missions, Current Flying," NOAA, https://www.nesdis.noaa.gov/current-satellite-missions/currently-flying, (accessed May 1, 2023).

[10] Toppr, "Communication Systems, Satellite Communication," Toppr, https://www.toppr.com/guides/physics/communication-systems/satellite-communication/#Advantages_of_Satellite_Communication, (accessed May 1,2023).

[11] A. Cavalier, "7 Influential Satellite Communication Service Providers to Keep an Eye On in 2023," ABIresearch, https://www.abiresearch.com/blogs/2022/12/12/satellite-communication-service-providers/, (accessed May 1, 2023).