BRAD Station Software Overview

Mathew Clutter

April 19, 2022

1 Introduction

This document offers an overview of the BOREALIS BRAD station software. This software is written primarily in Python, along with Arduino code written in C++. There are two major versions of this software: a non-IMU version, and a version that utilizes an IMU. Each version has their respective advantages and disadvantages. This program utilizes the PyQt5 framework. This framework allows for the creation of a GUI using the Qt Designer program, and the pyuic command. This allows for the visual creation of the UI. An overview of both versions, along with technical details of each will follow.

2 Main Program Overview

The main program is constructed as a main class, that connects elements of the UI to functions, as well as instantiates other classes.

A high level overview of the Python and Arduino code for the basic tracking software follows:

On the Python side, the main tracking loop works by grabbing the latest balloon coordinate information from the server every 5 seconds. The latitude, longitude, and altitude of the balloon are then passed to the trackMath class, which, along with the ground station latitude, longitude, and altitude, computes the line-of-sight distance, azimuth, and elevation from the ground station location to the balloon. The azimuth and elevation calculated are then sent over serial to the Arduino. This basic loop is repeated until the tracking is stopped.

On the Arduino side, a loop is constantly checking for new serial data. If new serial data is available, the first character of the transmission encodes what operation is requested. While tracking, the Arduino code will parse the serial data sent from the tracking loop in Python and separate out the new azimuth and the new elevation that was sent. After parsing out the values, the Arduino will calculate the difference between its current azimuth and elevation, and the new azimuth and elevation that is requested. It will then convert the difference in azimuth and elevation into steps for the stepper motors that controls the pan and tilt of the ground station. The steps are then sent to the motor controls over I2C, and the new position is set. This process repeats while the tracking is active.

There is also a fairly large number of comments placed in the source code. The source code will hopefully be easy enough to follow and read. (The PyQt5 program may be a bit confusing at first, if unfamiliar with the framework. Don't worry, it's not too complicated once you familiarize yourself with the library a bit.)

3 Main Program Overview (IMU)

The IMU program works similarly to the basic tracking program. However, instead of entering a manual calibration, the IMU position is used to determine the azimuth and elevation that the ground station is pointed.

The Python tracking is also slightly altered. Before the ground station moves to the updated position of the balloon, it reads in the current azimuth and elevation reported by the IMU. This allows the program to still point correctly if the ground station is bumped or otherwise moved. It also helps to eliminate the issue of backlash.

The Arduino code that connects to the motors is identical to the non IMU version of the software. There is an additional Arduino used to read data from the IMU and report the yaw and pitch over serial, which is read in to the Python program with PySerial. This Arduino Code is a slightly modified version of the TransducerM TM200 Arduino Library example, generated from their IMU Assistant program.

4 Github and Continuing to Work on this Program

Currently, this codebase lives on my personal Github account. There are 2 separate branches, one for the basic tracker and one for the IMU based program. Feel free to fork, or clone any of these branches and continue working on them. One note is that the repository is fairly large, and so if you would like to save some space on your disk, I'd recommend doing a shallow clone if cloning the repository. Basically, feel free to continue development in whatever way you see fit. If you would like me to add you to the current repository, let me know. Or, if you would like to fork the repository, or take the code and start your own repository, feel free.

A link to the current Github repository is `https://github.com/mathias314/Ground-Station-Tracker/tree/main`

5 Creating the Program

5.1 Creating / Editing the GUI

The GUI for this software was created using the PyQt5 library and the Qt Designer software. Qt Designer allows the creation of a GUI with a visual editor, which can be saved as a .ui file. After creating/editing the GUI using Qt designer, in a command line, run the following command:

```
pyuic5 uiFile.ui -o designerFile.py
```

This will generate a .py file that will contain all of the information regarding the UI layout.

You may need to install the pyqt5 library if this command doesn't work. To do so, run the following pip install:

```
pip install pyqt5
```

For more information, see the PyQt5 documentation.

5.2 Creating .exe file

Normally, Python is an interpreted language, which requires a Python interpreter to be installed, and the proper libraries to be installed to run a Python program. However, in the interest of making this program more portable and easy for end users to utilize, a .exe file can be created from the Python program. This is done using the pyinstaller package. In order to create an exe from a Python source file, run the following command:

```
pyinstaller --onefile main.py
```

This will generate an exe file from the Python source file, and link all of the other files and libraries called by the program into a single executable file, that can be run without a Python interpreter.

If needed, pyinstaller can be installed using the following pip command:

```
pip install pyinstaller
```

6 Ideas and Improvements

What follows are various suggestions for things that need to be implemented, or could be improved upon:

6.1 Declination Correction

The IMU reports the yaw according to magnetic north. Thus, it is necessary to correct the azimuth to true north. An automatic way to set the declination for whatever location the ground station is located at would be greatly beneficial. As of now, this is hardcoded into the IMU class in the Python program. A more elegant and flexible solution to set the declination automatically would be a great improvement.

6.2 Bad Readings from IMU

Occasionally, the IMU will output incorrect yaw values. This is causing the ground station to occasionally pan to an incorrect position, which will then self correct. I've attempted to remedy this issue by taking an average of a few points to average out the occasional bad reading. It may be good to alter the amount of points that are being averaged to find a good amount of points that filters out the bad readings, but still reports quickly. There may also be other viable methods to remove the bad readings. Ensuring that the IMU doesn't send out errant values would be a good improvement.

6.3 Interfacing with IMU

Currently, the IMU utilizes another Arduino to send the yaw and pitch values to the ground station software. It may be possible to interface with the sensor directly over the USB port, and get rid of the extra Arduino.

6.4 UI Improvements

Currently, the UI is not the prettiest, and could use some love. There also may be some display issues on certain screens. If you'd like to improve how the program displays and looks, that would be a good improvement.

6.5 Better Calibration Method (non-IMU)

Currently, the non-IMU version requires the sun to be visible to calibrate. This is not ideal, as the sun must be visible in order to calibrate accurately. Currently, a workaround method is to use a compass (ensure that it is set to true north, or you account for magnetic declination) and a level to manually enter the starting location. While this gets the calibration relatively close, it is not super accurate. A better calibration method that does not depend on the sun would be beneficial in making this version more robust.

7 Advantages and Disadvantages of Each System

The non-IMU system is cheaper (as it does not require an IMU, or the second Arduino.) However, it does have some more limitations in terms of calibration and ease of use. If the sun is not visible, accurate calibration is very difficult. Additionally, it takes an extra step to calibrate compared to the IMU version.

The IMU version, while a bit more expensive, does offer some advantages. These include easier calibration/setup, as the user simply needs to connect to the Arduinos, set the ground station location, and select the balloon (they avoid the step of entering/setting the initial starting position of the ground station). Additionally, the IMU tracking version can more accurately utilize the predictive tracking algorithm, as the IMU will negate the impact of backlash in the gears. However, the IMU system is not without its disadvantages. In addition to the cost, the software is a bit more complex. There is the need to account for declination, as the IMU zeros based on magnetic north, not true north. Thus, there needs to be code in place to account for this difference in magnetic zero points.

Overall, each system has their own respective advantages and disadvantages.

8 Contact Information

If you have any questions, feel free to contact me at mathew.clutter@mines.sdsmt.edu. I'll do my best to get back to you and answer any questions you may have about this software. Best of luck with working with

this software!