Control Room Final Report

Tyler Franks
Liam Bradly
Joshua Snyder

YCAS Radio Telescope Project
Senior Software Design Project II, Spring 2022
Professor Donald J. Hake II
York College of Pennsylvania

Table of Contents

Abstract

The following is the technical report for the Control Room team written during the Spring Semester of the fourth year of development on the YCAS Radio Telescope Project. Over the course of the Spring 2022 semester, the Control Room team was able to make significant improvements and enhancements, finalize certain aspects, add new functionality, and make the Control Room Application more stable, testable, accessible, and functional. The team has improved testing to cover a wider range of functionality, completed a wide variety of tasks and fixed bugs, improved interaction with the ESS, and implemented encryption for the TCP[9] Communication Protocol to be used with the Mobile App.

The current plan is to deploy the telescope at John C. Rudy County Park at the end of the Summer 2022 semester. Further real-world systems-level acceptance testing will need to be done once this is completed, which future team members will be responsible for. The Control Room team created numerous issues on GitHub that will help future members understand which areas of the software need more work.

Introduction

York College of Pennsylvania (YCP), the York County Astronomical Society (YCAS), and the York County Park System (YCPS) collaborated in an effort to develop and design, from the ground up, a radio telescope to be installed at the York County Astronomical Society (YCAS) observatory located at John C. Rudy County Park. This telescope has the capabilities of remote access, auto location, and auto-tracking.

Some of the Control Room team's tasks this semester include continuing the development of the Control Room Application for the radio telescope, and adding any last requested features, such as:

- Integrate a DHT22[21] temperature and humidity sensor
- Implement internal fan control
- Add customizable settings for the Sensor Network[1]
- Perform inertial testing between the Control Room App and the RT HW mount and analyze collected data
- Encrypt communications between the Control Room App and the Mobile App
- Update the custom orientation input dialog to use the appropriate software stops
- Add a command to reset the MCU error bit
- Update the appointment calibration routine
- Update the backend to include all sensors and their statuses

Other tasks of the Control Room team include improving upon and testing the work of previous teams, such as:

- Finalize existing functionality
    - Improve testing
    - Fix bugs
    - Log any unfixed bugs to the issue tracker so they do not get forgotten
    - Document how different parts fit together and interact with each other
- Verify that functionality that was instantiated from previous teams can be replicated reliably and consistently

The Control Room team faced many challenges and obstacles this semester including onboarding new team members, having to learn about the hardware on the fly, and troubleshooting and accounting for new hardware issues as they arose. Perhaps the most difficult obstacle the team faced this semester, however, would be the downtime we had to deal with, leaving the team with roughly only 3 weeks to test and run the telescope. Even so, the Control Room team was able to persevere and succeed in the objectives that were defined at the beginning of the semester and are well-positioned to onboard new members and continue work into the Fall 2022 semester.

Background

Throughout the semesters, each new team has brought something new to the Control Room Application in terms of functionality and usability. The team previous to this semester, in the Fall of 2021, reimplemented TCP communications with the Mobile App and Front End, implemented software stops that act as another layer of stop protection, and performed full integration testing on the hardware[11].

As the Control Room team continues to expand the functionality of the Radio Telescope, the Mobile and Front End teams must also accommodate any of the updates.

The Control Room Application software (CR) drives the telescope's hardware, communicating with a Programmable Logic Controller[2] and Motor Controller Unit (MCU)[3], and receives radio frequency (RF) data through the SpectraCyber[4] which is then saved to a MySQL[5] database. The database, using the Entity framework[6] will save relevant data that the Mobile and Front End teams can access and present for user consumption, such as:

- Appointment RF data
- Sensor statuses
- Temperatures
- Acceleration data blobs
- Weather information
- Appointment statuses

Because of this communication with the database, the YCAS members who will be administering the telescope will be able to effectively and efficiently control the telescope, maintain it, and whenever it enters an unsafe condition, return it to a safe and stable state.

The Control Room team's main tasks this semester included:

- Conduct Inertial Testing and analyze collected data
- Encrypt Mobile App communications
- Integrate a DHT22 temperature and humidity sensor
- Implement internal fan control
- Make Sensor Network customizable
- Reimplement appointment calibration
- Stow the telescope correctly even if it hasn't been homed
- Speed up Sensor Network position updates
- Make software stops toggleable
- Add a Stop button
- Check for absolute encoder discrepancies and timeouts
- Implement sensor statuses

- Add new ResetMCU Bit command for Mobile TCP
- Fix azimuth discrepancy
- Reimplement stopping deceleration
- Remove unused and obsolete code
- Verify that the connection with the hardware is still possible
- Verify that connection with other teams is still possible
- Rigorously test the software (alongside the hardware)
    - Record any bugs or abnormalities
- Fix as many bugs as possible
- Expand testing so that it encompasses more functionality

Implementation

Sensor Network

The Sensor Network[1] gathers sensor data from various sensors and sends that data, at regular intervals, to the Control Room software. This data is received in a packet that must be decoded on the Sensor Network Server end and then parsed into a representation that can be displayed on the Control Room software's user interface. The Control Room software and/or the user can then make decisions based on that decoded data, if necessary. Requests to turn the fan on and off are also sent from the Control Room to the Sensor Network. A separate Sensor Network Testing Suite application that facilitates troubleshooting and generating test CSVs and packets has also been developed alongside the Control Room software.

Here are the various types of sensor data being retrieved from the Sensor Network:

- Azimuth Motor Temperature
- Elevation Motor Temperature
- Azimuth Motor Acceleration
- Elevation Motor Acceleration
- Counterbalance Acceleration
- Elevation Absolute Position
- Azimuth Absolute Position
- Internal Temperature and Humidity

DHT22 Integration

A new DHT22[21] temperature and humidity sensor was added to the Embedded Sensor System and was integrated into the Sensor Network[1]. The internal ambient temperature and humidity read from the sensor are used to determine whether or not a fan should be turned on or off. Integrating this sensor required many changes to the Control Room software, Testing Suite, and Backend.

A new data type, Humidity, was added to the Control Room application and to the Backend database. Humidity data is very similar to how temperature data is formatted. Humidity data contains the relative ambient humidity, time samples, and location sampled should we ever want humidity read elsewhere.

Both internal ambient temperature and humidity are displayed on the UI as shown below. The ambient dew point is also computed and displayed on the UI. Being a new sensor in the sensor network, there is now an initialization checkbox used to enable or disable the sensor data, as well as sensor statuses and error codes for the senso.

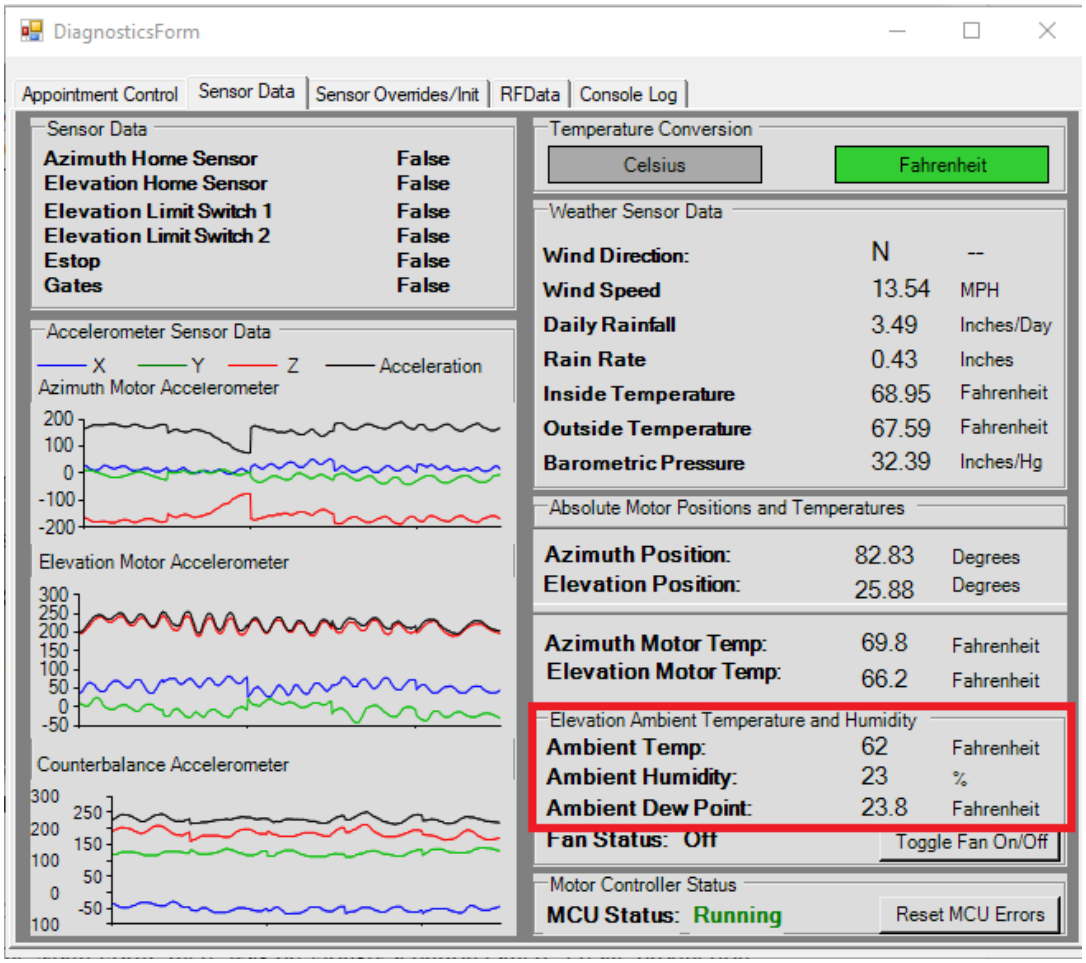

Figure 1. Ambient Temperature, Humidity, and Dew Point on the UI

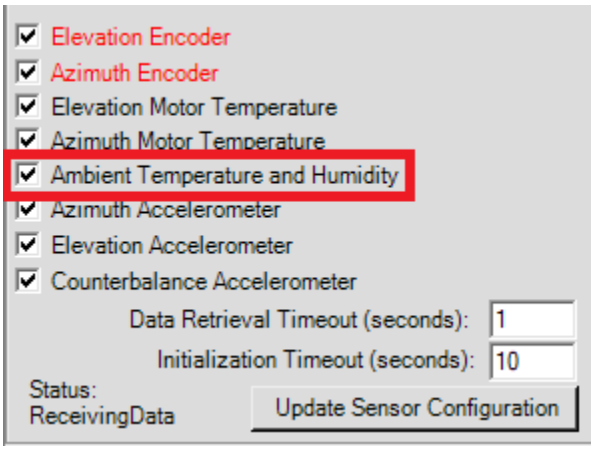

Figure 2. Checkbox for initializing ambient temperature and humidity sensor

Adding the DHT22[21] also required changes to the Sensor Network packets received from the Embedded Sensor System. Temperature and humidity data were added to the end of the packet, and statuses and error codes were added to the beginning.

Fan Control

A new fan was added to the Embedded Sensor System, used to cool down internal ambient temperatures and reduce condensate from building up on the inside of the telescope. To monitor and understand how condensate could build up on the inside of the telescope, the fan control needed to know both the outside temperature and dew point. This ultimately led to the decision to have the fan control logic take place on the Control Room application side of things.

The fan's primary purpose is to reduce heat and moisture inside the elevation frame. To do this, customizable upper and lower temperature and humidity thresholds were added to the diagnostics form (**Figure 3**). These thresholds are stored in the database and are used to determine at which temperatures to turn the fan on and off, and at which relative humidity levels to turn the fan on and off. The flow diagrams for the fan control logic are shown below.

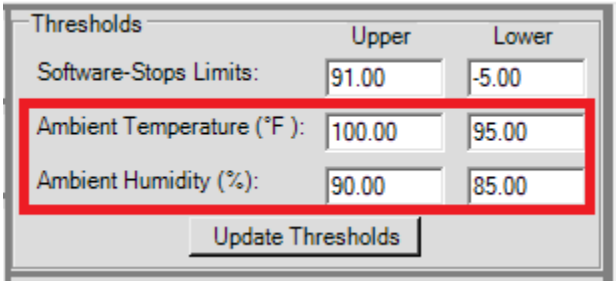

Figure 3. Ambient temperature and humidity thresholds

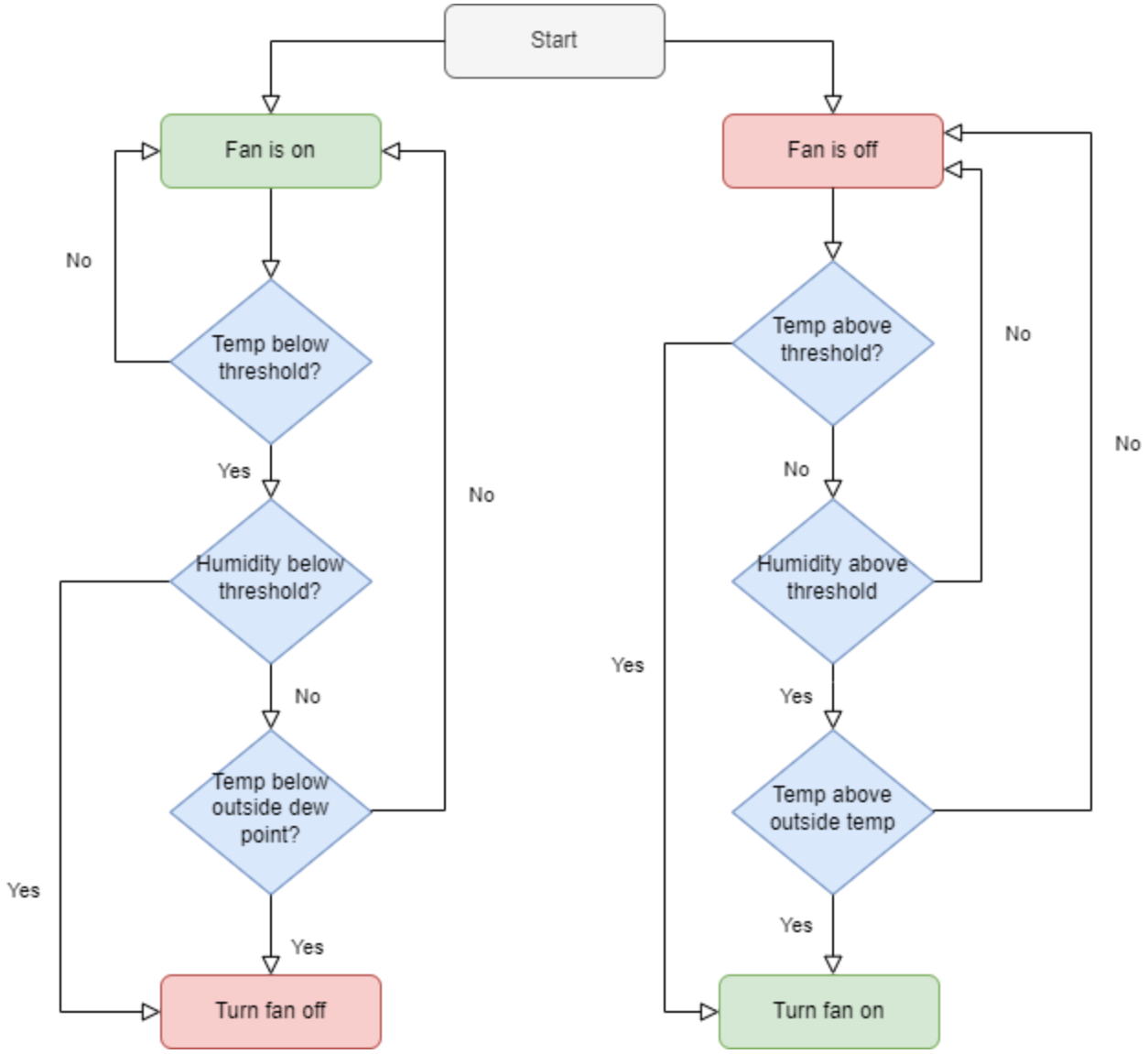

Figure 4. Fan control logic

In the case that the fan is desired to be permanently on or off, a new override button was created that would override the fan logic and disallow any further changes to the fan state. Additionally, a fan toggle button was also added to manually change the fan state. The status of the fan is always displayed on the diagnostics form so that the user may always know what the fan is doing.

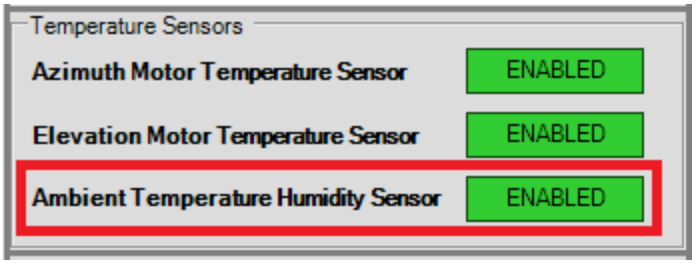

Figure 5. Ambient temperature and humidity override

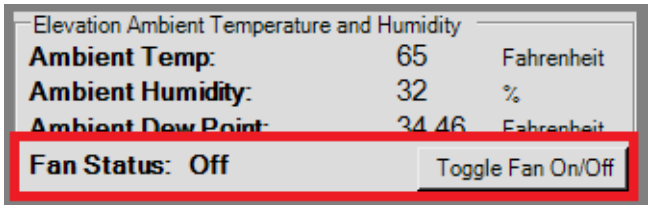

Figure 6. Fan toggle button and status

Adding the fan logic to the Control Room application side of the Sensor Network required that a new two-way communication protocol be established. After every Sensor Network packet is received, the Control Room application immediately sends back a packet containing a byte indicating whether or not the fan should be on. The Sensor Network Packet received by the Control Room application also contains the state of the fan, so that in the case of a disconnection, the fan state can be known immediately.

Sensor Customizability

Once the telescope is moved to the park, there may be many settings and/or values that we may want to tweak within the Embedded Sensor System code, but we may not have the ability to flash the Teensy[14] remotely. To get around this, certain characteristics of the Embedded Sensor System code have been made customizable through the initialization packet. Mainly, all three ADXL345 3-axis accelerometers[12] and multiple timing periods have been made customizable from the diagnostics form.

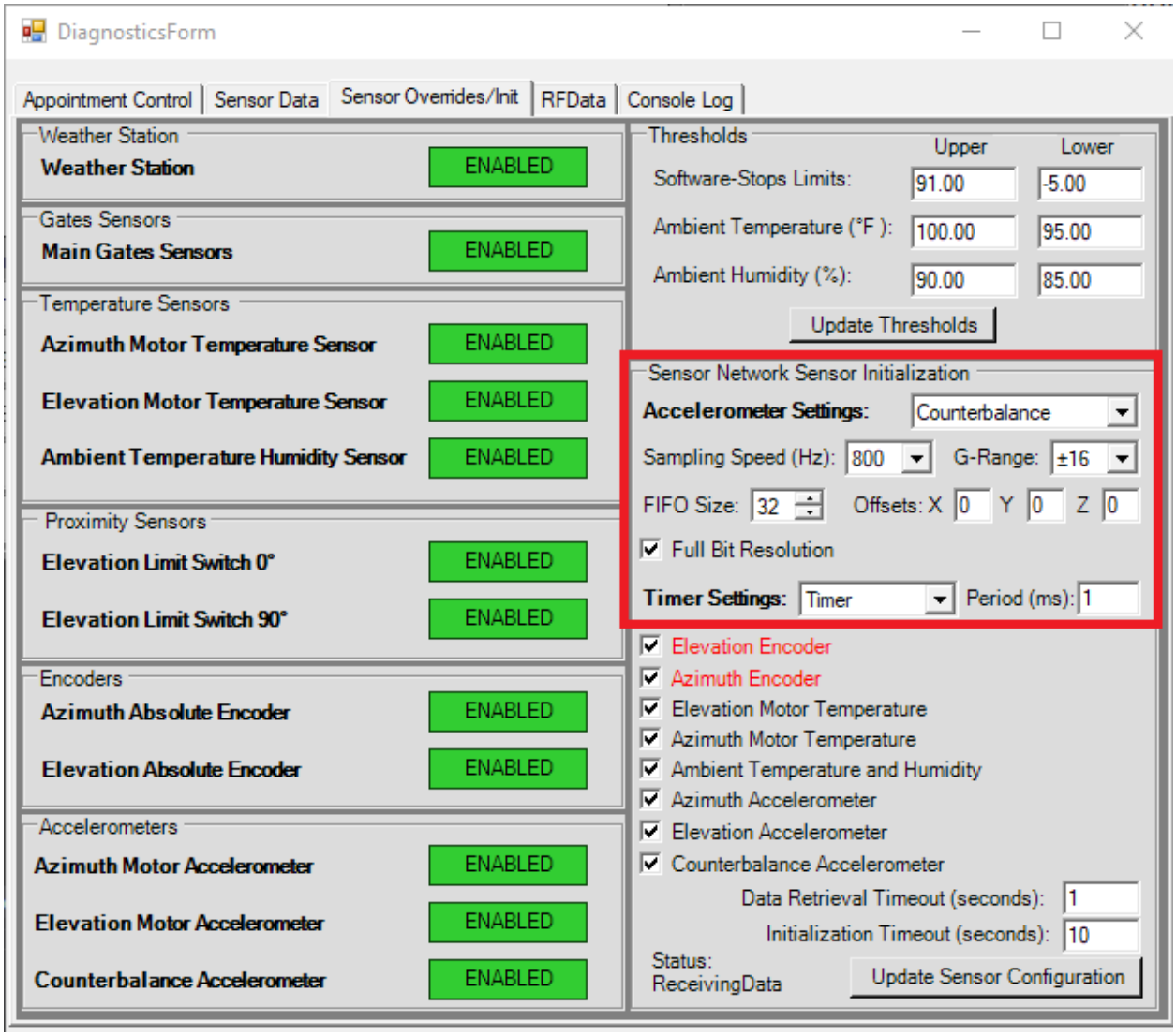

Figure 7. Sensor customizability options

For the accelerometers[12], there are multiple different options for customizations. Each accelerometer has its own sampling frequency, FIFO size, g-range, XYZ offsets, and can be full bit resolution or 10-bit resolution. The sampling frequency determines how fast the acceleration values are sampled. The FIFO size determines how many samples are stored on the accelerometer until it is ready to be emptied. The g-range is the range of g-forces that the raw acceleration data is mapped to. The XYZ offsets are the offsets applied to the raw acceleration data and are used for calibration. Finally, the full bit resolution option is used to set the accelerometer to full bit resolution or 10-bit resolution, which affects the precision of the acceleration data. Refer to the ADXL345 datasheet for more information.

There are also multiple different timing periods that can be set. All periods are set based on milliseconds. The timer option sets how often the timer ISR on the Teensy[14] will interrupt, the ethernet option sets how often the main packet data will be sent to the Control Room application, the temperature option sets how often the temperature sensors will be sampled, and the encoder period sets how often the absolute encoders[13], [22] and counterbalance accelerometer are sent.

All areas in the Control Room application which used static sensor settings have been updated to use the new customizability options. These options are stored alongside the Sensor Network[1] Configs stored in the database so the settings will persist from session to session.

Sensor Status

We want to know what sensors are functioning and have records of the sensor statuses in the database so we may understand what has caused an error. To do this, it involved a modification of the sensor monitoring routine to set statuses appropriately which included absorbing our weather monitoring routine.

The sensor_status table was updated to include statuses for all sensors. It includes timestamps for when the statuses are updated and a status for each sensor in the network. For most sensors the database can be updated through the Sensor Network[1] server, some logic was provided for calculating a sensor timeout for the remaining sensors though some future work will involve computing statuses from the raw Sensor Network data.

Stow Without Homing

When using the radio telescope, there may be times when the telescope will stow even though the motors have not been homed yet. To address this problem, the Control Room team utilized the Sensor Network's[1] absolute encoders[13], [22] to determine the position of the telescope, run the stow command, and verify that the telescope was successfully stowed with the absolute encoders. Due to the simulation's absolute encoders not reflecting the actual position of the simulation telescope, Stow Without Homing is disabled on the simulation.

Inertial Testing Software

To support the inertial testing done by the radio telescope team, the Control Room team developed new inertial testing tools[18] using Jupyter Notebooks and Python. The software allows the user to pull data from the database based on the inputted datetimes, and graph the pulled data for further analysis. Temperature data is pulled for both the azimuth and elevation motor and graphs for each are generated (**Figure 8**). Accelerometer data is pulled for each of the ADXL345

accelerometers[12] (elevation motor, azimuth motor, and counterbalance) and each axis is graphed as well as the acceleration magnitude (**Figure 9**). Each of the acceleration graphs also has its own frequency analysis applied to them. FFT and Lomb-Scargle analysis are used to extract frequency data from the raw acceleration data (**Figure 10**). FFT generates a graph much faster than Lomb-Scargle does, but Lomb-Scargle produces more accurate results due to the non-uniform acceleration sampling.

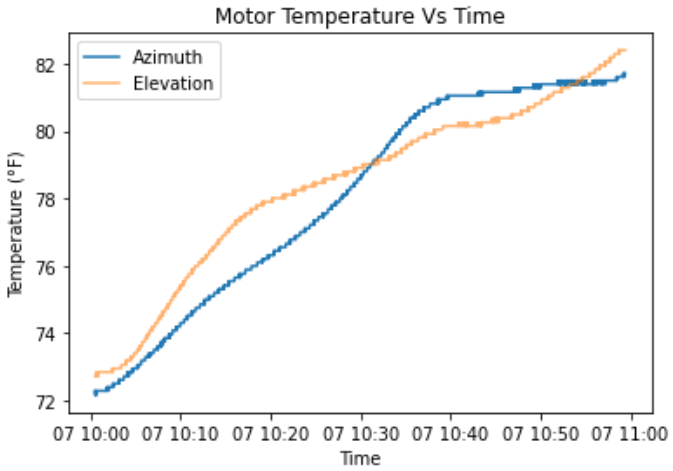

Figure 8. Motor temperature graph

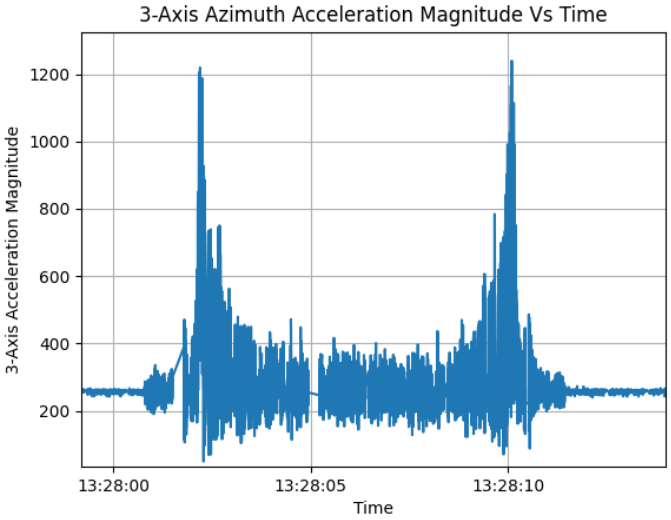

Figure 9. Acceleration graph

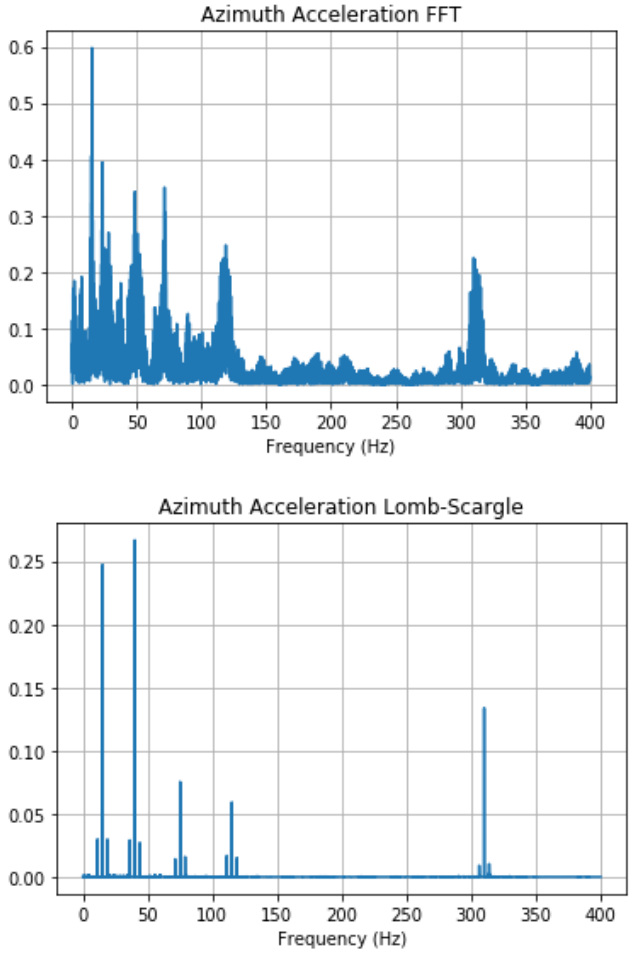

Figure 10. Frequency analysis graphs (simulated data)

A large benefit to the inertial testing tools[18] is that it serves as a framework and proof of concept for integrating these analyses into the Control Room application.

AES-256 Encryption

Data sent between the Mobile App and the Control Room is now encrypted using AES-256 encryption. This includes all data that is sent between the two platforms, such as commands, output log messages, etc. Encryption only operates on versions 1.1+ of our TCP[9] protocol. However, older versions of the protocol will still work with the Control Room as normal, but any data sent between the two platforms will not be encrypted.

The process of sending and receiving data from the Mobile App is as follows: Data is received from the Mobile App and tested to see if the data was encrypted. This is done by checking the first token of the data string, which contains the version number. If the version number is 1.1 or above, then the Control Room knows that the data is encrypted and proceeds to

decrypt the data. Once the data is decrypted, the data is parsed and processed like normal. To send data back, the control room encrypts anything it sends back since it knows that the version of TCP the Mobile App uses supports encryption. After the data is encrypted, it's sent like normal to the Mobile App. Figures 11 and 12 below visualize this process.

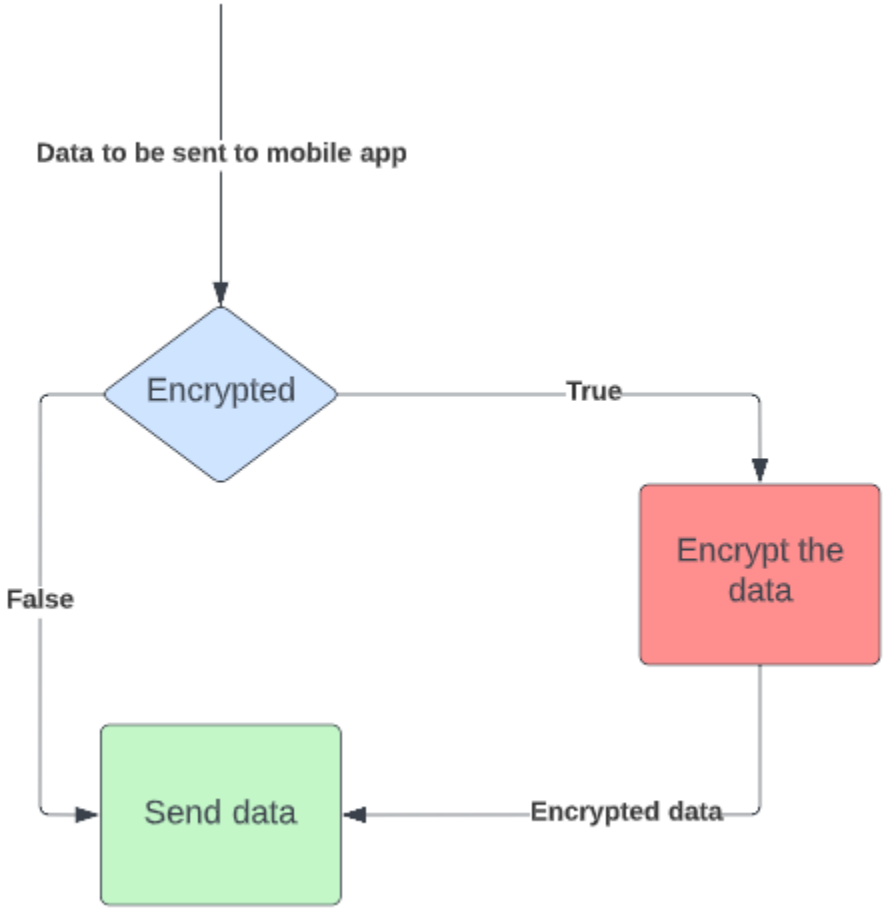

Figure 11. Receiving data from the Mobile App

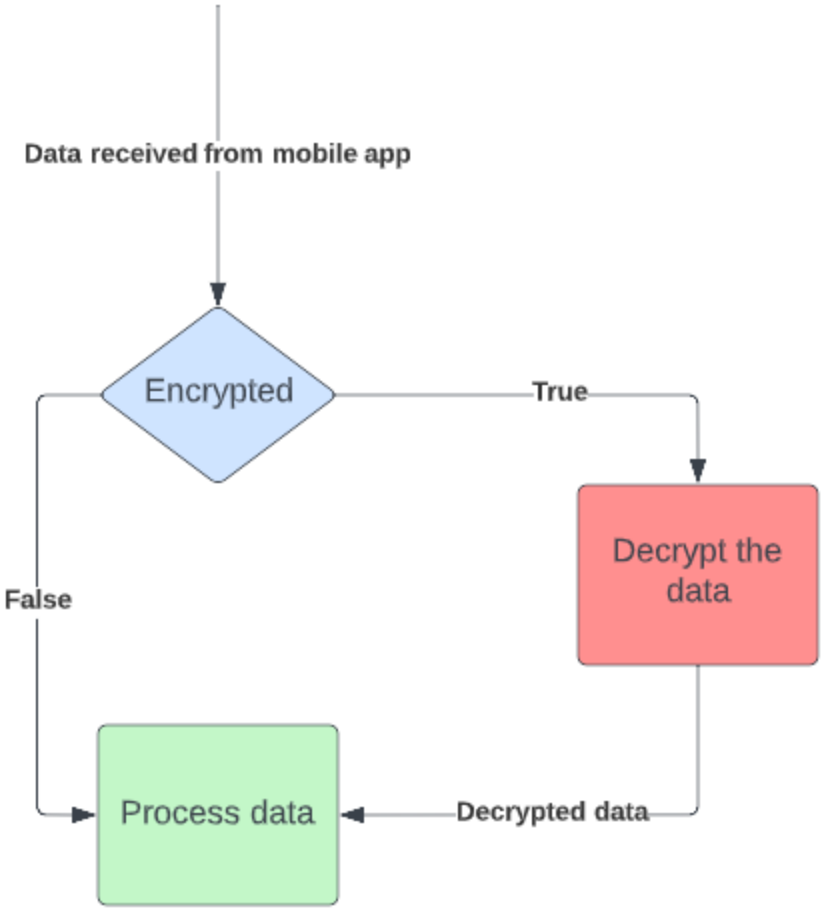

Figure 12. Sending data to the Mobile App

Appointment Calibration

The existing calibration routine was scrapped for a new calibration routine that accomplished what was originally intended. The new calibration routine is as follows:

1. Home the telescope
2. Perform two SpectraCyber[4] reads for calibration before the appointment
3. Run the appointment
4. Run another appointment calibration in reverse order.

The calibration data is stored in CSV files in addition to the appointment data and then sent as part of an email once the appointment has finished.

Each calibration involves two SpectraCyber reads: the first points up to the sky to get a neutral reading; the second points at a specific tree on the property that outputs a specific radio frequency associated with that tree. Running calibrations both before and after an appointment

can tell users if any discrepancy has occurred during the appointment, and can be factored into the data accordingly.

To uphold industry standards, a full calibration must be performed before and after an appointment to ensure that data is accurately read. To slightly improve the timeliness of this process, the beginning calibration and the ending calibration work in reverse order from one another to prevent multiple unnecessary movements, although this is only partially effective as we must home the telescope between appointments to ensure no azimuth discrepancies.

Software-stop Toggles

The Control Room is now able to switch between using the absolute elevation encoder[13] and the counterbalance accelerometer[12] to read elevation data. The user is also able to change between the two as well by using a checkbox on the main control form. In the event that either fails, the Control Room will switch the device used to read elevation data. Besides failure, the Control Room will switch from the elevation absolute encoder to the counterbalance accelerometer in the event that the encoder is out of range. It should be noted that the user won't be able to manually switch to a device that can't be used. For instance, if the counterbalance accelerometer is experiencing an error, only the elevation absolute encoder can be used, and the user will be notified about this when attempting to change devices.

Custom Orientation Input Form

The previous input form for the user to enter a custom orientation to move the telescope was poorly designed and needed revision. We also needed to ensure that the correct software stops were used when validating the input into the form. Changes have been implemented so that the correct software stops are used when validating input. The form also updates per character entered, hence if invalid input is entered, a message will be shown to the user letting them know. More changes have been made to the form that allows for a much cleaner appearance, as well as making use of abstraction so that it can be used at other points in the Control Room if needed. Figures 13 shows the new input form.

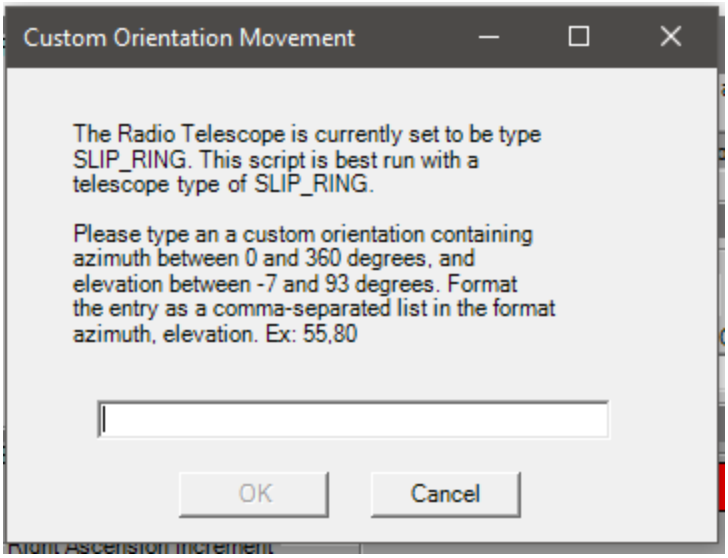

Figure 13. The new custom orientation input form

Reset MCU Error Bit Command

The MCU[3] error bit can now be reset via the Mobile App remotely. This involved implementing a new command parse and execution method since a new command needed to be created. This command is only supported with versions 1.1+ of our TCP protocol[10].

Check for Absolute and Motor Encoder Discrepancy

Previously there was no check to ensure that absolute and motor encoders were reading similar data. A check was implemented that uses a discrepancy constant to ensure that the motor encoders aren't falling behind the absolute encoders[13], [22]. If that were to happen, it would indicate that a motor was skipping steps, and likely experiencing an undesirable level of stress.

Stop Button Addition on Control Form

We have previously had stop functionality for the telescope, but there was no implementation on the control form to stop the telescope. Previously one would have to hit the red emergency stop button to stop the telescope. With this addition it uses the same stop scripts as the Mobile App and allows the user to stop the telescope from software instead of using the hardware to stop the telescope.

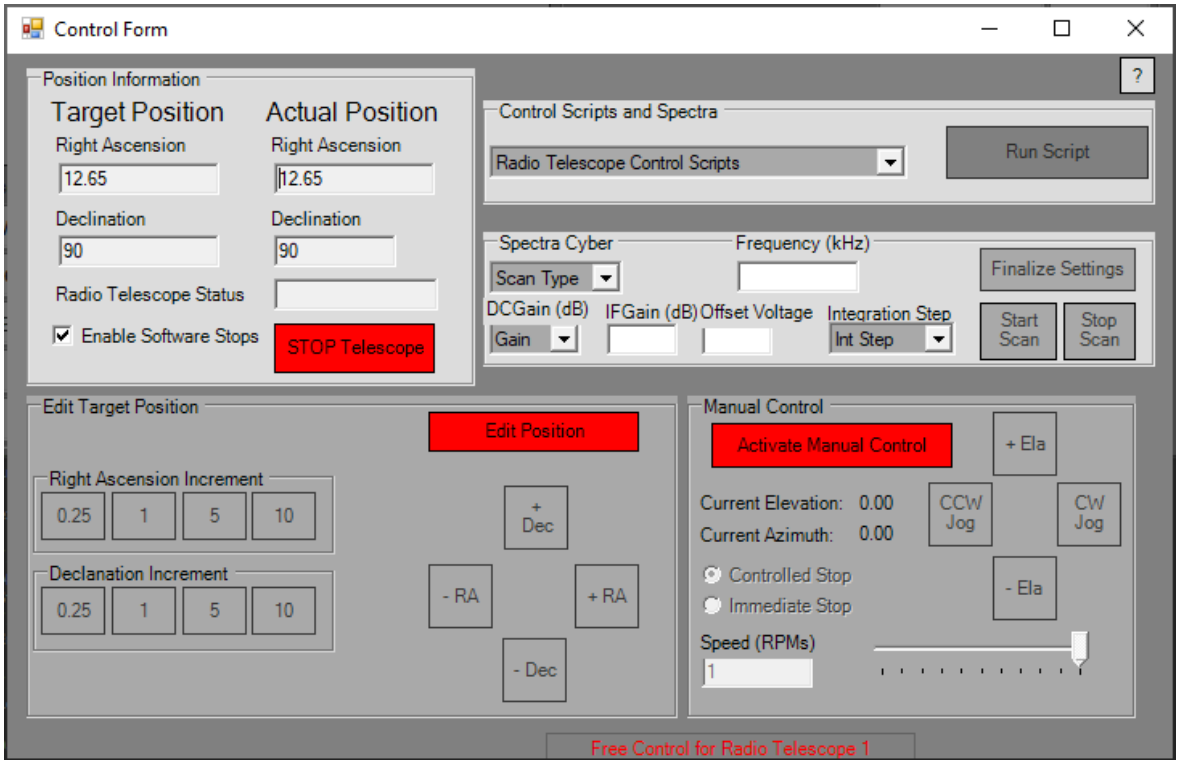

Figure 14. Updated Control Form

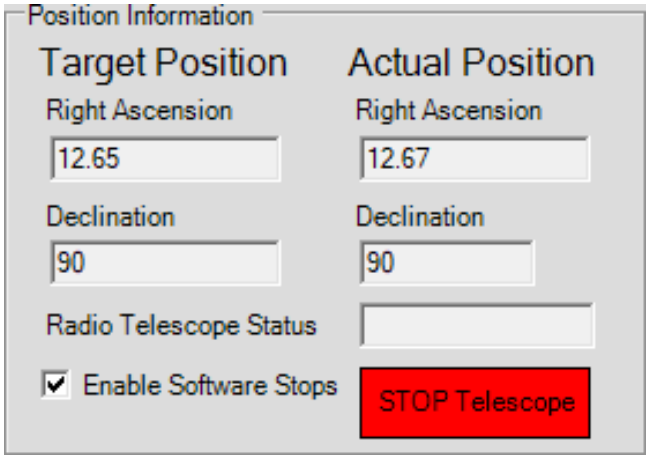

Figure 15. Stop Button Implementation

New Immediate Stop Implementation

While conducting inertial testing, the Control Room team discovered that the immediate stop command worked too well, which could result in damage to the radio telescope mount if an immediate stop was executed while the mount was operating at a sufficiently high rate of rotation. To get around this problem, controlled stops are now used for all movements and an

adequate amount of deceleration is computed by squaring the input velocity and dividing by the distance.

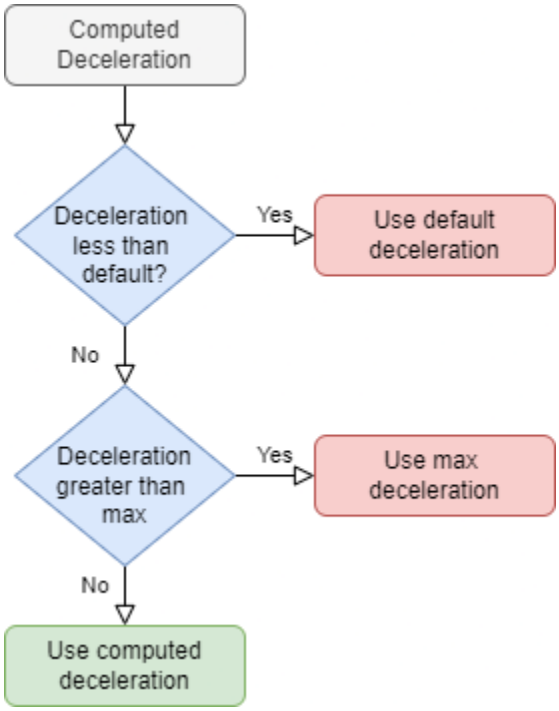

Figure 16. Deceleration logic

The deceleration will stop the telescope within the specified stopping distance. All stop commands now call upon the new controlled stops instead of the immediate stops except the homing command. Homing commands cannot be stopped by traditional controlled stops, so an immediate stop is used instead for the time being. Homing commands run at a sufficiently slow rotation rate, so an immediate stop will not damage the RT mount.

User Manual Updates

To ensure that any user knows how to operate the Control Room application, a new updated User Manual was added to each of the forms describing how to operate the software. Any outdated information was removed and updated.

Connection With the Hardware (Team)

Throughout this semester, the hardware and hardware team played an incredibly important role in the development of the Control Room application. The Control Room team made it a priority to test each and every development build on the hardware before pushing it to the master branch. Through this testing with the hardware, more bugs were found that would not have been found otherwise.

Understanding where the hardware team was with their development on certain aspects of the telescope was an extremely critical part of communication between teams as well because there were times when something was incomplete and the telescope was not in a good state to run CR tests on it.

Bug Fixes and Additional Tasks

There were several bugs that the previous team documented for us to begin with this semester. Another of the Control Room team's goals this semester was to make note of every bug that was found and fix as many of them as possible. New bugs were found through numerous sessions of rigorous testing, as well as normal use of the Control Room Application. The presence of these bugs, and the ease of discovering some of them, exemplified the Control Room Application's strong need for testing.

The Control Room team made a significant effort to not allow these bugs to be forgotten, adding an issue for each one to the GitHub issue tracker. While many of the existing bugs were able to be fixed, new ones were still found that ended up getting leftover and will be a priority for the next team. Any that were impeding progress were fixed, though each and every one will need investigation in the future.

Fix Simulation Acceleration Timestamps

While developing the inertial testing tools, the Control Room team discovered that the timestamps generated by the Simulation Sensor Network[1] were incorrect and overlapping. This caused the acceleration data to appear overlapped and a mess. The issue was fixed by applying an offset that simulated time passing between acceleration dumps.

Receive Acceleration Time Captured from the Sensor Network

Wrapping up work from the Fall 2021 semester, the acceleration time captured was unintegrated work that contained a bug. The Sensor Network[1] packet did not contain the correct amount of acceleration values when an accelerometer[12] was disconnected. This was fixed in the Embedded Sensor System code by guaranteeing that the number of samples to send matches the samples collected.

Fix Azimuth Discrepancy

Many issues appeared with the hardware when assembling the telescope back together and conducting inertial testing. One of the major issues discovered during testing was that the azimuth did not rotate the correct number of degrees. All of these

hardware-specific issues were fixed on the azimuth-discrepancy-fix branch, which includes motor direction changes, port text boxes, and an azimuth discrepancy constant. These changes were left unmerged so that the simulation is still usable. The port text boxes for the SpectraCyber[4] and weather station were swapped around and the motor directions were switched.

The azimuth discrepancy was a roughly 10.03 degree offset between the azimuth motor encoder and absolute encoder[22] for every 360-degree rotation. It was a mostly consistent discrepancy but further testing revealed that the discrepancy varied by about 0.5 degrees every few rotations. A constant was applied to all computed motor ticks so that the offset became minimal, but further testing is still required to characterize the discrepancy.

Fix Incorrect Position Errors While Running Simulation

When running the simulation, every movement made would result in “Incorrect Position” errors. After some investigation, the source of these errors was determined to come from an absolute encoder[13], [22] check. Since the simulation absolute encoders do not reflect the simulation movement of the telescope, the movement result always appeared to be an incorrect position.

Jog Stop Fix

Previously, Jog commands could not be interrupted using traditional stop methods. This was because Jog commands are not monitored movements, so the only way to stop them is to do so manually, not by using the movement monitor method. The Control Room team changed the Jog movements to be stopped using controlled stops manually.

Fix Timestamp Issue

Initially we had tests to check that our timestamps were valid, but they were created during Eastern Standard Time, meaning the only time stamp to receive was EST. This semester we ensured that all units of time within the Control Room were using UTC time so that communication with the database would work properly regarding appointments. We also updated the times displayed to the user to display in local time, so that regardless of where the telescope is, the time displayed will always be that of the user’s timezone.

Fix memory leak in RemoteListener

Originally, objects that implemented the IDisposable interface in the RemoteListener[8] class were not being properly disposed of in the event of an exception

being thrown. We modified the class to encompass the objects within a “using” statement, which made sure that each instance declared within the statement was properly disposed of, regardless of exceptions being thrown.

Fix DeleteCSVFile Failure Message

The logic in the DataToCSV class was malformed and resulted in a failure message not being printed despite being supposed to print. This issue was caused by the number of delete attempts being checked was not the same as the number of attempts the DeleteCSVFileWhenDone() method took to delete the file. We fixed this by adding a constant in the MiscellaneousConstants class so that both routines check for the same number of attempts. Once this was done, all tests relating to the issue were passed.

Update Dropdowns for Default Values Checkbox

Since we’ve been able to test on the physical hardware of the telescope more often this semester, we wanted to update the “Default Vals (for production)” checkbox. Originally, the default values were set for the simulation. We updated the routine by having the checkbox update the dropdown values to values corresponding to the physical hardware. This includes the IP addresses and port numbers of the PLC[2] and MCU, as well as the IP addresses of the Sensor Network[1] client and server.

Improved Testing Coverage

There were several areas highlighted both by the past semester’s team, as well as new areas discovered that were in need of unit tests, or more testing in general. Over the course of the semester, the Control Room team brought the unit test count up from 344 to 400. The Control Room team is feeling more confident about the code coverage, though there are still several areas that are in need of improvement.

Added Tests

- Sensor Network tests for DHT22, fan control, and sensor configurations
- RemoteListener test for AES-256 encryption
- Motor and Absolute Encoder discrepancy tests

Documentation

As new features were added over the course of the semester, documentation was also updated alongside these features to encompass and explain the new functionality. This functionality includes, but is not limited to, Sensor Network[1] updates, UI updates, appointment calibration, and description of various parts of the software that the user interacts with and how they work. All diagrams that were seen in this document are also present in the Control Room team's documentation folder so that they can be referenced and edited by future teams if anything happens to change.

Inertial Testing

Extensive inertial testing documentation was created to explain all tools and parts of inertial testing. An Inertial Testing Procedure[16] was created to document the testing procedures put in place for inertial testing. Instructions for using TeamViewer Remote Desktop[17] were added so that the Control Room software could be monitored remotely during tests or in the future. Setup documentation for the Python Testing Tools[18] was also added so that any person can set up the analysis tools on their machine and analyze their local database data.

Remaining Tasks

At the beginning of the semester, a Remaining Tasks[19] document was created to better plan out and describe the issues listed on GitHub. This document took all of the GitHub issues and labeled them with priority and when they should be completed by. Future teams can use this document to plan out and gauge what their goals should be for each milestone.

Workflow Walkthrough

Created at the beginning of the semester, a Workflow Walkthrough[20] document was created to serve as guidelines and demonstrate typical Control Room workflow. This document is especially helpful for new team members getting up to speed and for setting general guidelines for Control Room team members to follow.

User Manual

The Control Room team updated the User Manual[15] document with new features and UI components. The updated manual was also added to the Control Room software and is accessible from all forms. Some parts of the manual are still outdated because certain components need to be removed or reworked, and that decision is up to our clients. The User Manual document contains comments highlighting each of these outdated parts.

GitHub

The current Control Room team also made a significant effort to document any possible issue that the Control Room software would need on GitHub. GitHub issue tracking has been a critical part of the organization of the Control Room team's inner workings and proved to be valuable for finding information quickly. Many of the uses for the Control Room GitHub repositor involve, but are not limited to:

- Separating milestones into projects, and organizing issues into each milestone
- Always making sure that all completed and/or duplicated issues have been closed
- Conducting code reviews using GitHub's integrated review functionality, requesting changes when necessary

Future Work

As illustrated by the Improved Testing and Bug Fixes sections, there is a lot of future testing that will need to be done. It is important for the Control Room team to set up future team members with a strong and clear path to follow regarding what needs to be done next.

Currently, during the Spring 2022 semester, we had the Capstone Engineering team for support, but in Fall 2022, there will not be a hardware team to help support hardware issues as they arise. It will be important for all members of the future Control Room team to become familiar with the hardware and be able to troubleshoot problems if something goes wrong.

As development continues, it will be important to maintain close communication with all teams. While the Front End and Mobile teams continue development on the MS, it will be important for everybody to be on the same page regarding commands, and what the Control Room team expects.

While priorities will have to be adjusted to match the engineering teams' demands, as well as the priorities of the York County Astronomical Society, a safe starting point would be the existing issues in the GitHub issue tracker. Those issues contain bugs, functionality that was not able to be completed, various areas of improvement in the codebase, and areas that need more unit tests. Areas of improvement regarding unit tests are mentioned above; those are some of the tasks that are left to accomplish (which are documented in GitHub). Please note that this is *not* a comprehensive list, and only covers the *major* areas that are still in need of attention.

SpectraCyber Spectral Scan CSV

Spectral scan CSV output does not include the frequency scanned, which is crucial to spectral scan data. The RF data stored from the SpectraCyber[4] must include the frequency scanned during each spectral scan, and the CSV files will need to show the frequency of each RF data sample. The clients are planning to do more with the CSV files generated from the scans, so SpectraCyber CSV files will need to conform to the format they wish to use.

Secondary Appointments

Our clients will likely want to operate the telescope as much as possible. If there is a public appointment running, they need to make sure that their appointments do not interfere with it. Secondary appointments solve this problem by being a lower priority appointment that our clients can run, and will be interrupted by any primary appointments set by the public. The secondary appointment type already exists in the Control Room application, but there is no logic implementing the intended behavior. The future team will need to implement the interrupting behavior and the logic to run the secondary appointments whenever primary appointments are not running.

Send Appointment Data to a User if their Appointment is Canceled

In the case that an appointment needs to be interrupted, all data collected during the appointment should still be sent to the appointment scheduler. Currently, the data is just dropped and never sent. The appointment is also scheduled to run immediately when the Control Room software is opened. The future Control Room team will need to implement a way for schedulers to get their appointment data should the appointment be interrupted before completion.

Shutdown RT Occasionally Freezing

Shutdown RT aims to call corresponding "bring down" functions of all the pieces. This includes ending TCP clients and servers, disposing of objects, and so on. That process still occasionally hangs up on the RadioTelescopeControllerManagementThread and will need to be investigated.

Allow Motors to Maintain Movement Between Orientations

This is a challenging problem that will require a lot of different ideas to come together to form a solution. The current Control Room team investigated how this might look, and found that Assembled Moves, mentioned in the MCU's manual[3], may be the solution. There are two assembled move types: Blend Moves and Dwell Moves. Blend Moves allow the motors to maintain movement between orientations, but the motors cannot change direction, so multiple Blend Moves may need to be sent to the MCU in order to maintain motion, stopping briefly if direction needs to be changed. An example of how Blend Moves works can be seen in Figure 17.

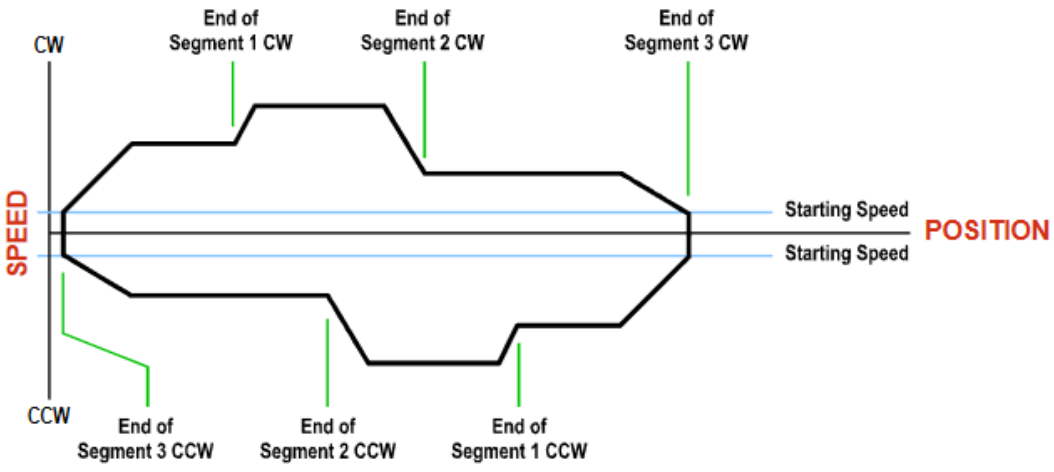

Figure 17. Blend Move diagram from the MCU's manual[3], showing motion being maintained but stopped if the direction needs to be changed

Implement Sensor Network Sensor Statuses

Currently, the Sensor Network[1] Server receives sensor statuses from the Sensor Network, but it does not do anything with those bytes. This would involve implementing logic around these bytes to output a sensor's status on the user interface, where action can be taken if a sensor fails. This also involves implementing the statuses in the Simulation Sensor Network, which does not simulate any kind of sensor statuses and returns that all sensors are in a failure state.

Characterize Accelerometer Data and Act Upon It

Accelerometer[12] data can be stored and displayed on the user interface, but nothing else is being done with it. The team still needs to determine what "bad" accelerometer data looks like, and how to act upon that in the sensor checks in **RadioTelescopeController**. To do this, perform an FFT on each axis of the data and identify normal operating frequencies and their relations to motor speeds. After that, come up with an upper bound of normal frequency amplitude and use that to detect "bad" accelerometer data.

Home Telescope Before Automated Movement

The Control Room application should only allow automated movements to occur if the telescope has been homed. Without being homed, the telescope will not be correctly oriented. Currently, if an automated movement is requested, the telescope will attempt to perform that movement based on wherever the motor encoders think they are, which will lead to incorrect results. Every automated movement should check if the telescope has been homed first and if it hasn't, home first.

Integrate Current Transducers

The HW team aims to integrate current transducers into the telescope during summer 2022. Those transducers will interface with the PLC[2], which means the Control Room application will need to collect, store, monitor, display, and act upon that new transducer data. The Control Room team does not know exactly what this will look like, but future teams need to be ready to support the integration should they need to.

Store Admin TCP Commands and Verify Timestamps

The Mobile App team will be sending admin information along with the TCP commands[10] so that the Control Room application can log it. They are also sending timestamps of when the commands were issued. These timestamps can be used for an extra level of verification so that the Control Room application only accepts commands that have valid timestamps. This way, packet sniffers cannot retransmit captured TCP packets to execute

commands. There will be thresholds around the timestamps to determine whether or not the commands are valid.

Interrupt Homing by Triggering Homing Sensor Signal

Homing moves can only be interrupted with immediate stop commands, which is bad for the life of the telescope's motors and gearboxes. There could be a way to trick the telescope into decelerating if its homing sensor signal is triggered manually, and an immediate stop is called right after. There is no confirmation if this is possible yet and would involve working with the MCU[3] and PLC[2]. If this is possible, stopping the telescope while it is homing would be much safer.

Make Final Position Offset Values Setable from Control Room

The final position offset is applied to the telescope's home position. It is how the Control Room application determines the physical orientation of the telescope. Currently, the only way to set the final position offset values is to manually SQL inject them. For the users' sake, it would be easier for them to set the final position offset through the Control Room application instead. The final position offset

References

[1] (n.d.) SensorNetwork embedded software repository
- https://github.com/YCPRadioTelescope/YCP_RT_SensorNetwork

[2] (n.d.) Programmable Logic Controller quick start guide
- https://drive.google.com/file/d/1Ehr0wrP6aTUBFwSFX0TD-wXjXachdyYj/view?usp=sharing

[3] (n.d.) Motor Controller Unit manual
- https://drive.google.com/file/d/1FZIEM_U2YgjXn-HmcZCzW56SdwLZef_1/view?usp=sharing

[4] (n.d.) SpectraCyber documentation
- http://www.ncra.tifr.res.in/rpl/facilities/4m-srt/sci_iimanual.pdf

[5] (n.d.) Entity Framework Overview
- https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/overview

[6] (n.d.) MySQL Official Site
- https://www.mysql.com/

[7] (n.d.) Amazon Web Services documentation
- https://docs.aws.amazon.com/index.html?nc2=h_ql_doc_do

[8] (n.d.) RemoteListener Documentation
- https://docs.google.com/document/d/16_HVjoNUsJ0Viq6EgDgaGzdX-fo2Y_0RmtDoi1egdQU/edit

[9] (n.d.) TCP Protocol Overview
- https://drive.google.com/file/d/1cV8tHFrGfRm__4U1mG9G2fSVQmTVpjK5/view?usp=sharing

[10] (n.d.) TCP Protocol In-depth documentation
- https://docs.google.com/document/d/1xSQuwcbpDC_5UnIyre336b_8t7jicmA4HsSlqfWuxt4/edit?usp=sharing

[11] (n.d.) Fall 2021 Control Room Team Final Tech Report
- https://docs.google.com/document/d/10RPAiB_wCfEqlzePWlvaZsp6BjeSuPRP_TsW9eVkqEU/edit?usp=sharing

[12] (n.d.) ADXL345 3-axis Accelerometer Data Sheet
- https://drive.google.com/file/d/1wJvjnoiX1-KQEXDOtdXxK6EDPpy6VVqw/view?usp=sharing

[13] (n.d.) Honeywell SMART Position Sensor 100 deg Data Sheet
- https://drive.google.com/file/d/1n9GoiZGoPCNxCLIzkwQu4hh-jnUnLp0B/view?usp=sharing

[14] (n.d.) Teensy 4.1 Microcontroller documentation
- https://www.pjrc.com/store/teensy41.html

[15] (n.d.) User Manual

- https://docs.google.com/document/d/1mzbJ0J7MzUnVulLPzXmTxSjKzZEhG1Y5/edit?usp=sharing

[16] (n.d.) Inertial Testing Procedure
- https://docs.google.com/document/d/1kJurZy5Rmlt9pgg3f9qcnR7MGIRgR46dzYnf-0yaomc/edit?usp=sharing

[17] (n.d.) Team Viewer Remote Desktop
- https://docs.google.com/document/d/1kJurZy5Rmlt9pgg3f9qcnR7MGIRgR46dzYnf-0yaomc/edit?usp=sharing

[18] (n.d.) Python Testing Tools Setup
- https://docs.google.com/document/d/1bwCuV5lJX-5D1dGGSGY11Q_mC1pPDzClWTKUtc605HU/edit?usp=sharing

[19] (n.d.) Remaining Tasks
- https://docs.google.com/document/d/1ecdeRKI8cJe0zdkzhhsniZOlR7Ki8IYFYQ0V4hHgrxc/edit?usp=sharing

[20] (n.d.) Workflow Walkthrough
- https://docs.google.com/document/d/1C_X7ITpnPVH8c_-ZnJ_Rbwotm3vXP_cXGYUhj2JmJFs/edit?usp=sharing

[21] (n.d.) DHT22 Datasheet
- https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[22] (n.d.) Midi Range Inductive Angle Encoder
- https://drive.google.com/file/d/13ax3rYI_46NK4k0j9LQoruXowl3f3kFk/view?usp=sharing