Viasat AMEND - Machine Learning Team

ECE4871 Senior Design Project

Viasat AMEND – Machine Learning Team
Xiaoli Ma
Gregory Kiesel

Shreyas Mhasawade - Computer Engineering - smhasawade3@gatech.edu
Adrija Bhattacharya - Electrical Engineering - adrija3@gatech.edu
Tyler Cole - Computer Engineering – tcole33@gatech.edu
Chris Rothmann - Electrical Engineering – crothmann3@gatech.edu
Mikias Balkew - Electrical Engineering - mbalkew24@gatech.edu

Submitted 28th April, 2022

Table of Contents

Executive Summary

Tracking satellite trajectories is a critical step for ensuring stable global communication and accurate data transfer between ground and low orbit satellites. One of the primary purposes of highly accurate tracking is to ensure ground dishes are pointed in the exact direction necessary to maximize Gain-to-Noise Temperature. Thus, a predictive tracking system is needed to steer ground dishes with a minimum accuracy within one tenth of a degree. This project specifically focuses on taking various inputs from a wide range of scenarios and training a machine learning algorithm to predict PI parameters based on previous orbit trajectories.

This project is the continuation of a previous team's work. This specific team will utilize machine learning (ML) techniques to efficiently produce control system parameters of an improved RF simulation model that can be used with Viasat systems. There is also a hardware team, that will aim to improve upon existing RF models. Both teams will work in close conjunction with one another.

The project involves using an existing Simulink model, which was left behind by the previous team. The hardware team focused on increasing the fidelity of the model, while the ML team placed emphasis on improving the accuracy of PI-controller parameters. The ML team developed scenarios representative of various possible configurations of control parameters to generate data with the Simulink model. This data was used to populate a database that serves as an input to an advanced ML algorithm, specifically a Recursive Neural Network (RNN).

The ML model analyzed the data being fed into it and predicted the RMS error between the satellite's angular position and the elevation angle of the ground station as a function of the eigenvalue lambda. The eigenvalues are directly correlated to control parameters via classical state-space control system design. Ideally, this analysis will allow a prospective control engineer to predict optimized eigenvalues (system pole placement) with respect to various system parameters of choice, such as RMS error, power consumption, etc. (only the former has been incorporated into the model thus far).

Viasat AMEND Machine Learning

1. Introduction

The Viasat AMEND Machine Learning team will not require any funding to complete its project of creating a machine learning algorithm that will generate reliable PI controller parameters. The end goal of the project is to generate reliable PI parameters that allow Viasat ground stations to reestablish communication with lost satellites, minimizing the time spent not communicating with the satellite. The existing model will produce some representative data of current scenarios, and the ML team will analyze this data using machine learning techniques to create more accurate inputs for the PI controllers. The main tasks for this project were the following: 1. Create a populated database with representative scenarios. 2. Utilize machine learning techniques to correctly parse the data and calculate improved input parameters. Future opportunities include incorporating other system non-idealities into the ML algorithm, such as power consumption, overshoot in the driven plant output, total energy dissipated, etc. The remainder of this paper will be laid out as follows: section 2 – background, section 3 - technical specifications, section 4 - design approach, section 5 - schedule, section 6 - demonstration details, section 7 – market strategy, section 8 – current status, section 9 – leadership roles.

2. Project Description, Customer Requirements, and Goals

The AMEND ML Team designed and implemented RNN code capable of accepting a wide range of RF inputs and generating PI parameters that are used to take predictive actions. These actions will correct for errors in the relative position between the actively tracked low-Earth orbit satellite and the grounded parabolic dish. Since the algorithm implements machine learning techniques, the goal is

to create increasingly accurate PI parameters (with more cycle runs) that produce improved corrective actions for the tracking system.

The team also created a Matlab database that holds data points generated from the existing RF (slow time) model. This database was used to feed input data into the ML algorithm. Primary tools that were developed included a method of extracting data from the RF model [1] (automatic population of the allocated database is left for future work). Lastly, we implemented a way to transfer this data into the ML algorithm.

This project's only stakeholder is Viasat, as they are the corporate sponsor who intend to use this technology. Some customer needs include the desire to create an ML solution that works with the existing Simulink model. This was accomplished by linking MATLAB code used to run the control system to RNN code run in Python.

There are several design constraints that we will have to keep in mind while developing our solution. Primarily, we will focus on keeping the algorithm latency to a manageable value. Having too high a latency will mean that the parameters will not be generated at the desired update rate [2]. We will also need to ensure that the results that we generate will be within the desired root mean squared error that will be specified by the consumer.

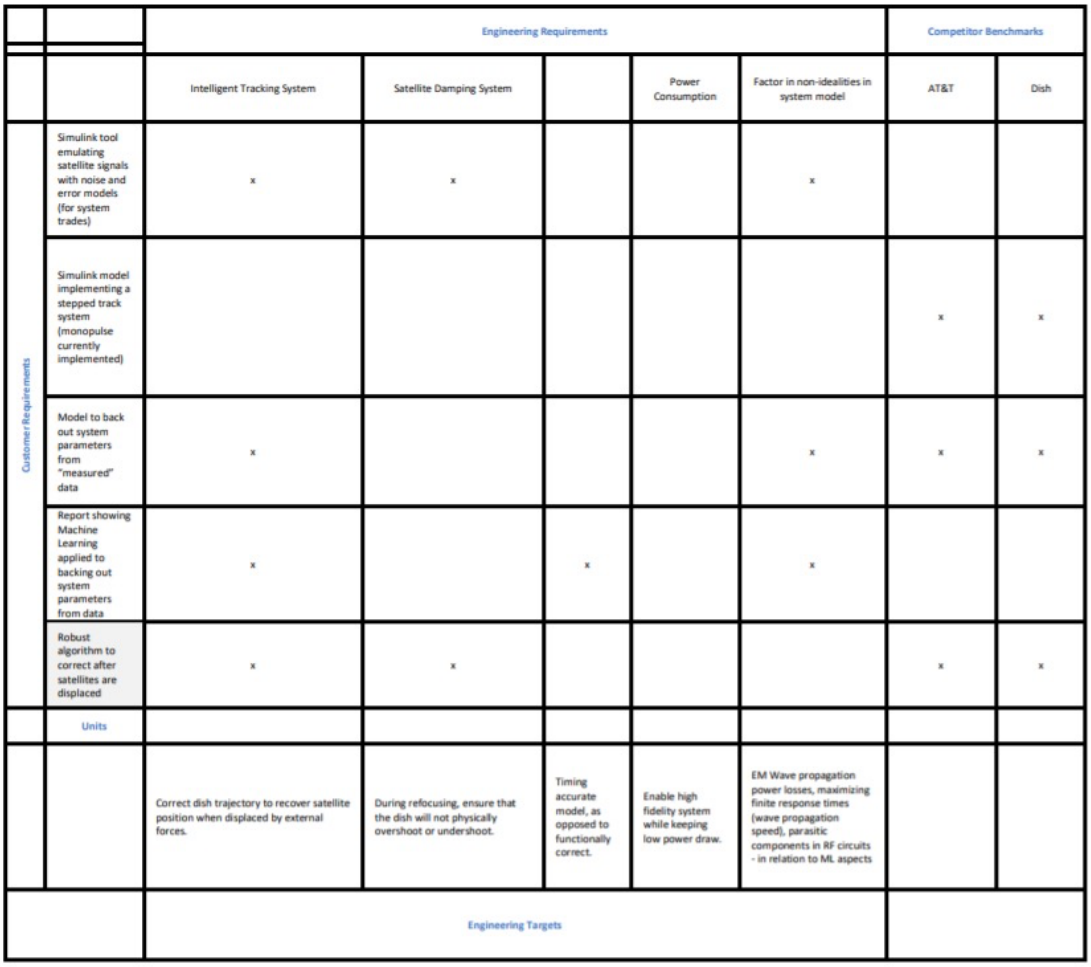


Figure 1. Quality Function Diagram based on Customer Needs (Column 1) and Engineering Requirements (Row 1)

3. Technical Specifications

The following design specifications were mandated for last year's Viasat AMEND team and was utilized for the development and implementation of the ML algorithm.

Minimum RF Input Bandwidth	200 MHz
Desired RF Input Bandwidth	3 GHz
I/Q Signal Bandwidth	10 MHz
Minimum SNR on Main Horn After Processing	7dB
Maximum Latency	0.5s
Control System Update Rate	50 Hz – 1kHz
Maximum Trackable Satellite Speed (Azimuth)	$15°/s$

Maximum Trackable Satellite Speed (Elevation)	$3°/s$
Maximum Mean Squared Tracking Error	0.01°RMS

Table 1. Technical specifications as requested by Viasat

4. Design Approach and Details

4.1 Design Concept Ideation, Constraints, Alternatives, and Tradeoffs

Our solution is implemented within MATLAB/Simulink, and we used Python for some parts. Initially, we were going to utilize the Matlab Statistics and Machine Learning toolbox in order to stay within the existing model's environment [3]. This would make it easier for interaction with the existing RF model and the controller. However, we have decided to utilize easier-to-use Python toolboxes, even though it will add some effort to make the ML synthesize with the existing model. The project will be an improvement to the existing slow-time model, generating updated and optimized PI parameters for the ground station controllers (servo motors).

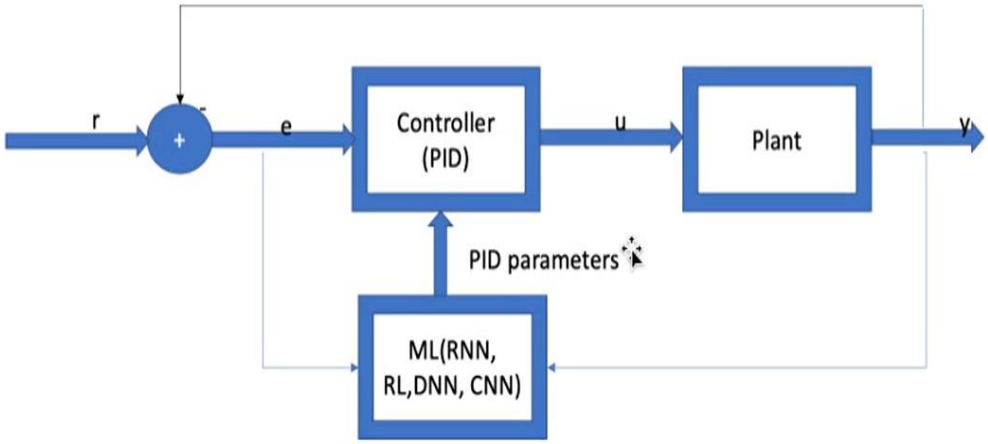


Figure 2. Block Diagram for proposed solution

Within our project, there exists two primary blocks. The first block is a MATLAB controller that will generate PI parameters. The controller accepts an “angle off boresight” as input, or an angle between the satellite and the ground station. Using this error e(t), it will calculate the input to the plant. The plant is a block that will take an input current and perform some calculations, outputting parameters like motor rate and load angle, defining instructions for the motor to minimize angle off boresight. The plant has been provided in the previous model, and it behaved as a black box for us. Based on the plant’s output, we will be able to calculate the difference between the output and the next input, yielding the error that will continuously feed into the controller.

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_p \frac{de}{dt}$$

Figure 3. Equation that will be outputted from controller, using feedback error

The second block was where the machine learning techniques were utilized. The second block took as input, the error that is being fed into the controller, as well as the plant output. Using this information, the ML block calculates new and improved PID parameters. These new PID parameters are propagated to the controller, which calculates a new plant input based on the new data. As more data is used to teach the algorithm offline, it will be able to generate more accurate PID parameters, minimizing error. This block was implemented in Python, using publicly available ML toolboxes.

One challenge we faced was connecting the ML algorithm to the controller. We wanted to be able to change the PID parameters in the MATLAB model with values calculated in Python. Getting the two platforms to communicate efficiently with one another required some additional engineering. It would have been easier to synchronize if the ML was coded within the MATLAB environment, but the computational overhead that would make the system slower. Additionally, there is much more documentation and many more resources that assist with implementing ML in Python than in MATLAB.

Another tradeoff lay in the fact that there is only a certain point up to which DSP systems can be added to the model before computational limits and latency start becoming a major issue. Thus, while higher signal processing power gives us more accurate computation for angle correction from errors, the higher processing time can increase the period of time between successive corrections of the base station dish. We used limited DSP system power to make up for the lost speed in using MATLAB over other programming languages. This leaves room for improving the efficiency of the machine learning algorithm by increasing signal processing power in the future.

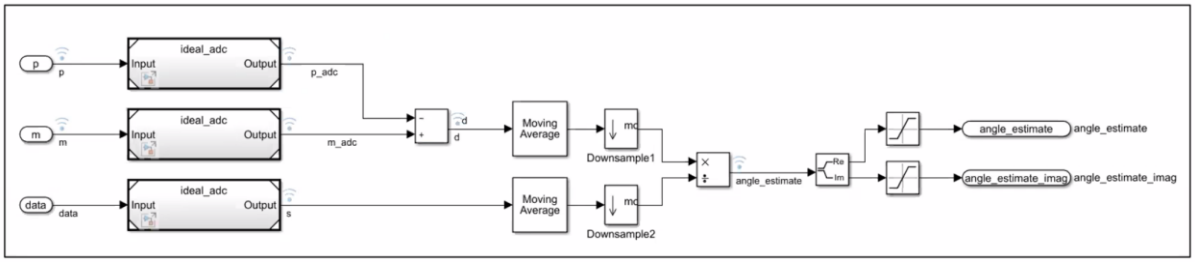


Figure 3. The DSP Subsystem

4.2 Preliminary Concept Selection and Justification

We selected our current path based mainly on what existing resources we had access to. The previous Viasat AMEND team, who had worked on this project during the 2020-2021 academic year,

had created a Matlab/Simulink model that would simulate the dynamic response of the antenna tracking system in the time domain.

This existing model is the main tool that Viasat has provided us with to assist with our project. By utilizing it, we know that we can reliably yield data that can be used to train an ML algorithm. Not only did the RF model exist, but there was also another Viasat team, the Viasat AMEND – Analog team, that continuously helped improve this model as our project progressed. We were provided with both a tool that will yield us measurements that would be too difficult to acquire otherwise, and a team that would work to improve the measurements given to us by the existing tool.

We modified the existing controller by setting up a PI control system in this block to synchronize it with the current model. The ML algorithm will simply generate new values for this controller block, which will be integrated into the existing system.

The model also provided us with a Simulink construction of the plant. The plant was given by the Viasat team to the previous GT AMEND team, who were able to implement the solution in Simulink. For our purposes, we will treat the plant as a black box, looking only at the input we feed it and the output we receive from it.

The design system for the database relied heavily on receiving data from the existing RF model. We carefully selected realistic input parameters that would create good data points to be stored within the database. The more representative of the real world the scenarios, the more accurate the output we will receive from the model. With the analog team working to make improvements to the model, we can be confident that our base data set is strong. If we did not have access to this tool, we would need to acquire Viasat's historical data of how they reestablished communication between ground stations and satellites.

4.3 Engineering Analyses and Experiment

Since the equipment that we worked with were expensive ground stations, it will not be viable to test our outputs on the ground stations themselves. If our results were not correct, they could potentially cost Viasat a large amount of money. The way we tested the results of our ML algorithm was by comparing our updated PI parameters to a truth value for RMS values, and going forward, the actual PI parameters that Viasat has historically created through their existing tracking technology can be used for this comparison. For example, we will have some test case of a scenario that Viasat had actually run into before. We would then run the input parameters for that scenario through the RF model, generate an output, and run that through the ML model to yield some updated PID parameters. We would then need to compare these parameters to the data that Viasat provided. If our updated parameters would yield more accurate (lower root mean squared error) results than the test cases, it would be safe to assume that the task is being done correctly.

4.4 Codes and Standards

FCC standard 15.209 regulates intentional emissions of RF energy. Although the auto tracking system will ideally not radiate any emissions, there was a chance that some RF energy would be unintentionally emitted. Unintentional emissions are exempt from the standard, so this standard didn't significantly affect the design [5].

The team also needed to comply with the FCC part 25 standard which contains regulations on carrier frequency tolerance, power radiation, frequencies used, as well as the angle of antenna elevation.

5. Project Demonstration

5.1 Data Generation

Data was generated using the MATLAB/Simulink model provided from last year. Variations were

applied to the eigenvalue “Lambda”. There is a direct dependence between lambda and the controller parameters Kp and Ki. Varying the lambda values allowed us to test new Kp and Ki values and observe their effects on the outputs of the model.

A reference angle of .25 degrees was chosen. After this, the maximum and minimum values of lambda for a stable result were chosen (physical limits on controllability of system for various eigenvaues). In the figure below, we can see that the lower bound for lambda is 10 and the upper bound is 33. Every lambda at a step of .01 is chosen, and the simulation is run again.

```
%loop generating training data
z = 10:0.01:33;
for i=1:length(z) %stopped at 2990
    [K1, K2] = init_controller_params(z(i)); %run params with new Lambda, get back K1, K2
    out = sim('slow_time'); %run simulation with new K1, K2
    gen_data(out,K1,K2);    %generate data with new output values
    disp(z(i))
end
```

Figure 4: Data generation code

Every time the simulation is run, a new set of output time series is generated. The elevation angle is the output that interests us. After every simulation, we write the RMS error of elevation angle, lambda, Kp, and Ki to an excel file. This data is the training data that is utilized by machine learning.

5.2 Machine Learning Code

The machine learning code is a python based recurrent neural network running libraries from *keras* and *sklearn* for the machine learning processing. The algorithm reads in a specified 10000 data point csv file provided by the MATLAB code and outputs a graph predicting specific parameters relevant to the Kp and Ki values. A time step of 30 was chosen for the training and 4 LSTM layers were added to improve the robustness of the model. For future variations, the graph can be changed to display predictions for any of the values held within the csv file. Furthermore, more features should be accounted for in the training to avoid potential overfitting of the data.

6. Schedule, Tasks, and Milestones

A detailed GANTT chart and PERT analysis has been included in **Appendix A**.

The analysis contains both the tasks that were completed in the current semester as well as for the next semester. The first semester's tasks include documentation and logistical assignments, while the next semester's tasks included tasks like software development, testing, and analysis. With each team member doing their tasks, we completed the project on time.

7. Marketing and Cost Analysis

7.1 Marketing Analysis

A competitor's product is not available for this project since it is internal to Viasat, and is thus specific to their ground stations. However, simulink models have been created by the previous team for the control system and tracking receiver. Our project will extend the model to better examine system performance trades and will add scenario modeling to support Machine Learning.

7.2 Cost Analysis

The Viasat AMEND Machine Learning work involved minimal physical hardware and materials. Any physical material necessary was equipment inherited from industry partner Viasat. The nature of the project means that this will not be a product for sale, but an internal tool that Viasat can use to save costs during their normal operations.

The estimated labor cost will be calculated as follows:

Estimated Hourly Pay Rate (via Payscale.com)
Entry Level Software Engineering salary = $91,000 per year
Hourly Rate = 91,000 / 2080 hours = $44 per hour

First Semester - 12 weeks	Total Number of Hours	Cost
Assignment Completion	24	$1056
Industry Sponsor Meetings	6	$264
Faculty Advisor Meetings	6	$264
Second Semester - 15 weeks		
Coding/Engineering	90	$3960
Industry Sponsor Meetings	7.5	$330
Faculty Advisor Meetings	7.5	$330
	Total Labor Cost per Engineer	**$6204**
	Total Labor Cost for 5 Engineers on Team	**$31020**

As can be seen from the above calculations, the total labor cost for our group working on this project was forecasted to be around **$31,020**. Our industry project partners, Viasat, provided us with a financial principle used to estimate the development cost of a program depending on the number of labor hours put in. The principle is called the "Rule of Three," and it takes the following factors into account:

1. **Overhead –** Accounts for costs like software, licenses, stationary, power, etc. (120% - 150% Labor Cost) Fringe Benefits – (30% Labor cost)
2. **Fees –** Covers unallowable costs (expenses acquired by contractor that do not meet the criteria to be billed) and overruns (5% - 10% Labor cost)
3. **Profits –** bonuses & shareholder dividends, cost necessary for program to turn a profit (10% - 20% Labor Cost)

Accounting for the above factors together yields a 200% increase from just the labor cost. As a result, our final cost calculation is as follows.

Estimated Program Cost = $31,020 * 3 **= $93,060**

Our estimated program cost utilized the "Rule of Three" that we were given, which accounted for all costs supplementary to our labor cost.

The nature of our partnership with Viasat meant that if this project was being completed in industry, rather than as a senior design project, Viasat would foot the bill for any related costs and expenses.

8. Results

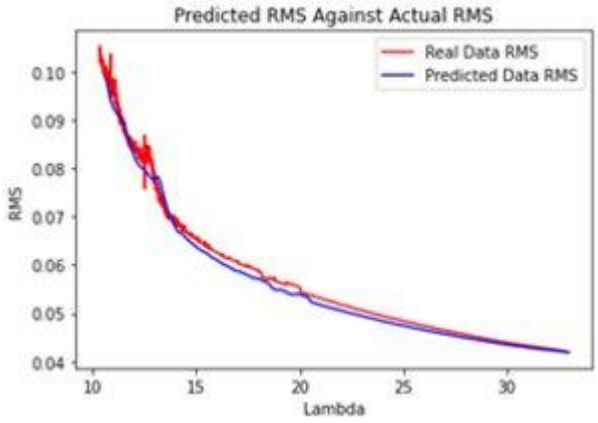


Figure 5: Graph showing predicted RMS plotted against the Lambda values to demonstrate improved RMS error

Graph of one of the possible outcomes from the machine learning model. This graph demonstrates the RMS value against the Lambda value. As shown, the model can closely predict the values which saves significant time over using the MATLAB code for the same output generation to find optimal Ki and Kp values.

9. Adding dynamic input

In addition to using constant reference, we decided to add dynamic input capability in the model for practical, real-time tracking. We tried to use a square function as input (left), and achieved some level of tracking on the right. This is quite imperfect at the moment and we intend to achieve better results in the future with appropriate eigenvalue assignment and timestep initialization.

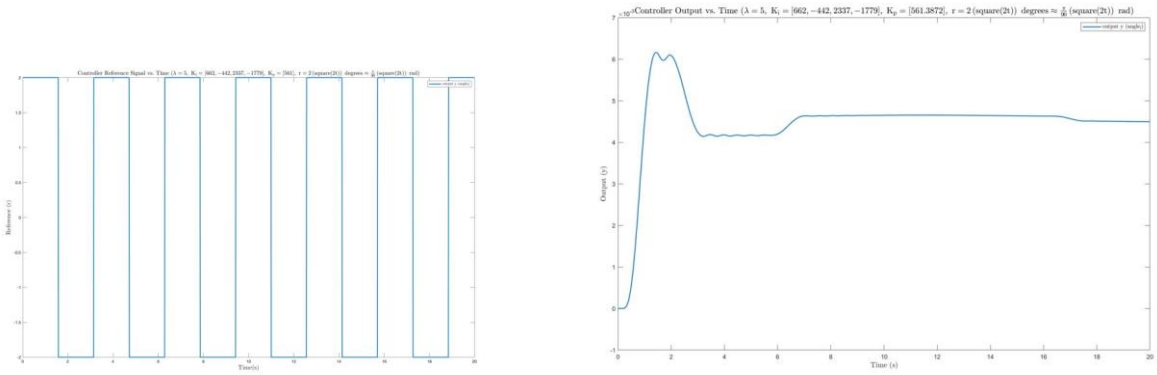


Figure 6: Square wave input (left) and the model ouput to the dynamic input (right)

10. Leadership Roles

We have split our team roles into technical interests and expertise. Although all people will spend time developing our product next semester, certain people will insight into their respective fields

Our team roles have been divided in the following manner:

1. Webmaster/ Git Coordinator – Tyler Cole
2. RF Insider – Mikias Balkew
3. Documentation/Viasat Coordinator - Shreyas Mhasawade
4. DSP insider - Adrija Bhattacharya
5. Expo Coordinator - Chris Rothmann

11. References

[1] R. Ranjan, “Calibration in Machine Learning,” 14-Sep-2019. [Online]. Available: https://medium.com/analytics-vidhya/calibration-in-machine-learning-e7972ac93555. [Accessed: 08-Oct 2021]

[2] M. Bkassiny, Y. Li and S. K. Jayaweera, "A Survey on Machine-Learning Techniques in Cognitive Radios," in IEEE Communications Surveys & Tutorials, vol. 15, no. 3, pp. 1136-1159, Third Quarter 2013, doi: 10.1109/SURV.2012.100412.00017.

[3] MathWorks, “Pricing & Licensing.” [Online]. Available: https://www.mathworks.com/pricing-licensing.html.

[4] M. Aorpimai, V. Malayavej and P. Navakitkanok, "High-fidelity orbit propagator for precise antenna pointing in LEO satellite operation," The 20th Asia-Pacific Conference on Communication (APCC2014), 2014, pp. 223-226, doi: 10.1109/APCC.2014.7091637.

[5] “47 CFR 15.209 - Radiated emission limits; general requirements.,” Govinfo. [Online]. Available: https://www.govinfo.gov/app/details/CFR-2010-title47-vol1/CFR-2010-title47-vol1-sec15-209. [Accessed: 22-Nov-2021].

[6] "The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition," in IEEE Std 100-2000 , vol., no., pp.1-1362, 11 Dec. 2000, doi: 10.1109/IEEESTD.2000.322230.

Appendix A: GANTT Chart & PERT Analysis

Paths

1-2-4-5-8-9-10-12-13-14-15-16-17
1-2-4-6-8-9-10-12-13-14-15-16-17
1-2-4-5-8-9-11-12-13-14-15-16-17
1-2-4-6-8-9-11-12-13-14-15-16-17
1-3-4-5-8-9-10-12-13-14-15-16-17
1-3-4-6-8-9-10-12-13-14-15-16-17
1-3-4-5-8-9-11-12-13-14-15-16-17
1-3-4-6-8-9-11-12-13-14-15-16-17

Critical Path:

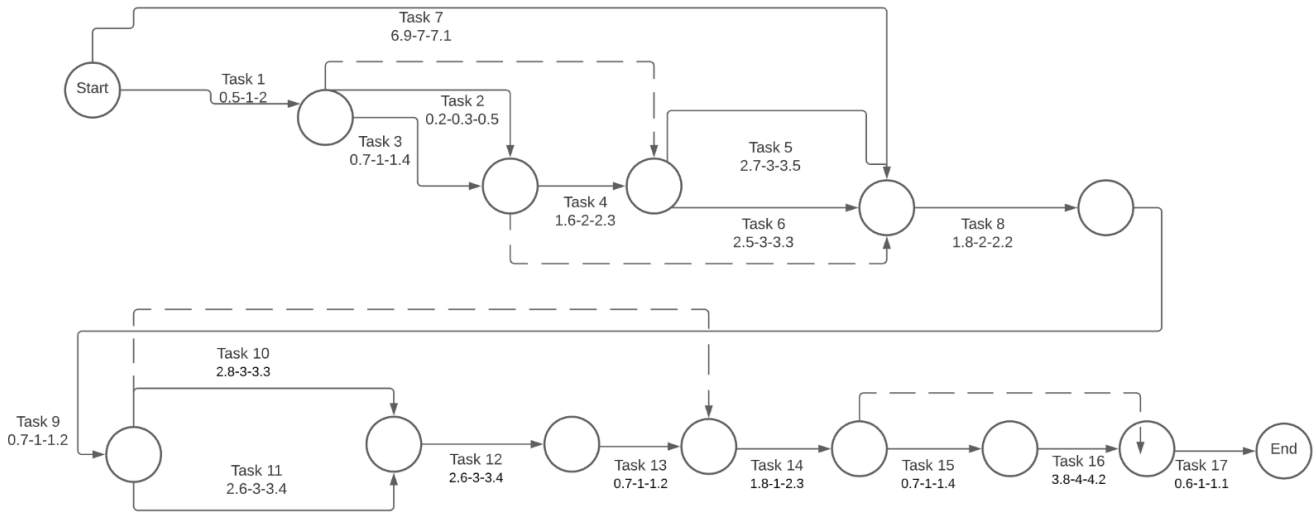
1-3-4-5-8-9-10-12-13-14-15-16-17

Time calculations for the critical path:

Critical Path

Task	Optimistic time	Most likely time	Pessimisic time	Expected Time	Dependencies	Std Dev	Variance
1	0.5	1	2	1.08		0.25	0.0625
3	0.7	1	1.4	1.016666667	3	0.116667	0.013611
4	1.6	2	2.3	1.983333333	4	0.116667	0.013611
5	2.7	3	3.5	3.033333333	4	0.133333	0.017778
8	1.8	2	2.2	2		0.066667	0.004444
9	0.7	1	1.2	0.983333333	8	0.083333	0.006944
10	2.8	3	3.3	3.016666667	9	0.083333	0.006944
12	0.8	1	1.3	1.016666667	10,11	0.083333	0.006944
13	0.7	1	1.2	0.983333333	12	0.083333	0.006944
14	1.8	2	2.3	2.016666667	13	0.083333	0.006944
15	0.7	1	1.4	1.016666667	14	0.116667	0.013611
16	3.8	4	4.2	4	15	0.066667	0.004444
17	0.6	1	1.1	0.95	17	0.083333	0.006944
Sums			27.4	23.09666667		1.366667	0.171667
							0.414327

PERT CHART from GANTT CHART:



From PERT analysis, the critical path did not change.

1. Expected duration of the project: **23.09 weeks**
2. Expected standard deviation for the critical path: **.414**
3. Probability of finishing project one week prior to design expo: **99.99999999987202 %**

a. Mathematics below

P(Tproject < Tleft-before-1-week-before-expo) = probability of completing the project 1 week before the design expo.

$\sigma_T = (\sigma_A + \sigma_B + \sigma_{C+} \sigma_D + \sigma_E + \sigma_F + \sigma_G + \sigma_H + \sigma_I + \sigma_J + \sigma_H + \sigma_K + \sigma_L + \sigma_M + \sigma_N + \sigma_O + \sigma_P)^{\wedge}(1/2)$

$z_S = (T_s - T_e) / \sigma_T$

where:

T_s = 26 weeks = time before 1 week before design expo

T_e = 23.09 weeks = sum of expected times from each task (expected duration, critical path)

σ_T = 0.414327 = total standard deviation of completion time (expected)

$z_S = (T_s - T_e) / \sigma_T$ 7.0234

using a standard z table:

P(Tproject < Tleft-before-1-week-before-expo) = 1 – P(Tproject > Tleft-before-1-week-before-expo) = 1 – 0.00000000000127981 **= 99.99999999987202 %**