**End of Term Report: Communications**

**Table of Contents –**

1. **Overview**
2. **Investigation of a smaller antenna**
3. **Transmitting Video Stream**
4. **Hardware Changes Made**
5. **ROS2 Communications**
6. **Path Forward**

**List of Figures –**

1. **8dBi 2.4GHz antenna (top) and 50-ohm load calibration reference**
2. **8dBi to 17dBi signal power at 2.4GHz quality over distance**
3. **Two Rocket M2 radios bridged as a P2P network**
4. **Ping test from station radio**
5. **Gantt chart for communications with updated progress**
6. **Station settings for the network**
7. **Access point settings for the network**
8. **Overview**

The communications team consists of solely Ian Dwyer. The system is still using the 2.4GHz 802.11bgn unlicensed band with Ubiquiti Rocket M2 radios. The rover radio uses a passive PoE injector that is ideal for DC battery power. A 10dBi omnidirectional OEM antenna for the rover, a 17dBi 90 sector OEM base station antenna, and another set of 8dBi antennas for the rover will be used. The control side will have multiaccess through a LAN router. The GNSS system is the Sparkfun XA1110 GPS breakout used for the autonomous mission. Radios and other essential equipment have been acquired, now allowing for further testing. Radio testing of network, video and antenna options has already commenced which will be the discussion of the report.

1. **Investigation of a Smaller Antenna**

A smaller antenna has been requested for the rover due to the center of gravity being so high it is a point of instability. The current antenna proposed is an OEM 10dBi omni antenna. The best quality replacement is only 8dBi. Characterizations were made of the new antennas to ensure quality across the 2.4GHz unlicensed band.

Chart, diagram

Description automatically generated

Figure 1: 8dBi 2.4GHz antenna (top) and 50-ohm load calibration reference (bottom)

It should be noted that the load does not sit at the matched 50-ohm line due to using an RP-SMA (Reverse-Polarity Subminiature version A) adapter in which required a unique method of calibration to acquire accurate results by alternating calibration standards to incorporate the adaptor whenever possible.

The calibration load standard provided a consistent return loss of -18.36dB at 46.2 ohms with a VSWR (Voltage Standing Wave Ratio) of 1.275. This provides an offset of 1.45 signal loss to the antenna measurement. The antenna provided a return loss average of -15dB at 41.8 ohms with a VSWR of 1.421. This provides an initial measurement of 3.16 return loss, resulting in an estimated 1.71% return loss total from the antenna. Furthermore, the VSWR of the antenna is closer to 1.26 with offset taken into consideration, which is substantially closer to 1, suggesting a good quality candidate as a substitute.

Chart, line chart

Description automatically generated

Figure 2: 8dBi to 17dBi signal power at 2.4GHz quality over distance

The above graph shows the transmit signal seen at the base station from the rover, and 8dBi transmit to 17dBi receive network connection. From the graph, it is noticeable in the worst-case function that the signal required for high-bandwidth applications, above 30dBm, is estimated to only reach 40% the distance of the 10dBi antenna. The best-case scenario shows that 3km is plausible, but highly unlikely to carry high bandwidth due to unforeseeable losses over such distances. These results suggest that investigation into this antenna could be worthwhile but could also be risky for missions out of line of sight of any of the events’ specified maximum distances in the guidelines, 1km for most and 2km for the autonomous mission.



Figure 3: Two Rocket M2 radios bridged as a P2P network

The goal was to estblish a point-to-point (P2P) communication between the two machines. A static IPv4 was assigned at 198.162.42.99 in the network tab to the access point machine and the static IPv4 of the station was manually assigned 198.162.42.98 via the ethernet settings of each device before configuration in AirOS. This allowed for communication and prevented the AirOS gateway from being blocked. The only devices to be usng the wireless conection is the radio link to remove any possible network congestion. Instead, all devices will be ethernet LAN to each node of the access point or station. Although a USB camera is currently proposed, and IP camera may be suggested since the computer at the rover would not need to process any data, freeing processing threads for controls and likely reducing latency due to extra hardware and software from processing.

Text

Description automatically generated

Figure 4: Ping test from station radio

The ping test only indicates a 0.003 to 0.034 second ping which is classified as real-time and qualifies us for this competition regarding the radio network itself. Limitations of hardware, software, compression, and other DSP encoding can still affect qualification for the event and must be considered given the real-time guideline provided by the URC. Minor changes will be made to this network as we progress, and most of the communications work will be optimizing the hardware and servers for our configuration.

1. **Transmitting Video Stream**

To establish a quality stream, RTSP (real-time streaming protocol) was used as it is the fastest heavily supported streaming protocol which makes it ideal for testing. SRT (secure reliable transport) is still expected to be the service used, but due to still having limited support is not suggested for initial video testing.

FFMPEG over Ubuntu bash was used with a few predesigned GitHub repositories. To get the best streaming rates many different parameters were considered. We do not need audio, so audio was excluded from the stream. Saving the stream at the controller’s PC will slow down the computer and will have to be saved at a different node, so stream saving was disabled. It is suggested for only 1 camera stream to be used at controller, even if 2 streams are concurrently active. A method for startup delay will need to be incorporated. The fastest and highest quality codec for video is h264 and x264 formatted for mpeg providing optimal packaging of the video data. It is suggested to operate at a 256x144 or 640x360 pixel frame at 10 frames per second with a dpi of 8 to 16 for RTSP. Each of these settings provided a functional image while greatly reducing the payload. UDP is still suggested so that dropped packets do not break the stream as TCP transport is known for. The initial testing is thanks to user Aler9 on GitHub for their RTSP simple server repository [1].

This current filtering and compression method has a few pitfalls observed. First is the latency in video has been subjectively observed to be approximately 2 seconds. Next, the video quality is less pixelated than other codecs but still may be too low of quality for functional use in comparison to the local video quality using the local camera application.

The throughput required for a single video stream averaged 4.6Mbps and peaked at 5.4Mbps. This implies that 6Mbps should be reserved for each video stream and with requiring a QPSK modulation scheme for long distances we are limited to MCS 3 or 4 providing 26 to 39Mbps, respectively, before channel congestion which should be more than appropriate. For lowest noise floor, given the figures outlined in the midterm report, MCS 1 or 2 is preferred giving 13 to 21.7Mbps, respectively, but also proving an extra 5 to 8dBm on our signal which could provide a significant boost in quality or distance as needed. If we do use these parameters, we may only have 2 concurrent video streams without compromising controls with greater delay.

These tests were done using the 8dBi antennas at close range, 10m with low power output set and low distance manually set in the gateway GUI. For distance testing, these will all be maximized given FCC regulations.

SRT will allow for significantly faster transmission at a minimum of 2.8 to 12 times faster depending on the transcoding method implemented [2] where all the parameters previously discussed still apply. The future testing should require establishing a reliable, dedicated server customized for local network needs. The company who developed SRT, Haivision, provides a GitHub repository for establishing the server, managing the connection, and parameterizing the packet transport protocols [3]. Developing a quality streaming method for high-bandwidth communications is an essential focus to ensure that the latency for the controls is as close to our measured ping value as possible to ensure that the rover qualifies for competition. Video streams and ROS will be the only communication streams between the controller and the rover where reliable, latency free transmissions will be a focus for communications.

1. **Hardware Changes**

Given the previous results, it may be suggested to use the 8dBi antennas for short range missions and still utilize the 10dBi OEM antenna for the long-range autonomous mission. The second notable hardware change is a fuse box implemented to prevent destruction of the hardware given charges may be stored due to the inductive motors and kick back high voltages into other components within the system which would burn out sensitive equipment such as the communications equipment.

As previously mentioned, an IP camera will be investigated for reducing latency. The USB camera will be good for the autonomous portion of the competition where video data is used for completion of certain tasks such as passing through marked goalposts, but any means to reduce latency should be utilized

Integration of a 9 degree of freedom IMU (inertial measurement unit) incorporating acceleration, magnetic orientation, and a gyroscope has been suggested for extra tracking and arm/hand control feedback from communications. Accuracy from the GPS and control program should provide enough redundancy, so the IMU may not be necessary but is still to be considered for characterization of the rover’s performance.

1. **ROS2 Communications**

The mutually agreed platform for controls will be the ROS2 Humble package in which the official ROS site suggests the latest Ubuntu distribution, explicitly 22.04, for the most reliable communications [4]. ROS is a publish-subscribe communications protocol where sensor data and remote actuation commands are packaged into open-source packages with protocols being customizable on high-level languages including C++ and python, with data serialization provided in any of the common formats YAML (Yet Another Markup Language), JSON (JavaScript Object Notation) and XML (Extensible Markup Language). The ROS is controlled through the Linux bash and require minimal software overhead for execution. Reliable delivery of these packets is essential, and communications will be exploring the impact of these packets over the network. Each condition should be extensively reviewed. First condition being heavy ROS traffic with two concurrent video streams. The second condition, this load over the maximum manual mission distance, 1km. The third is this load out of line of sight. The final condition being able to achieve reliable monitoring conditions for testing of the autonomous mission’s distance of 2km.

The network load will likely need to be simulated to do timely testing where the actual network load from controls can be slowly incorporated for more accurate measurement as controls prototypes. The load should be simulated with similar conditions as control requirements. Analog readouts equivalent to the number of encoders should be sending signals from the rover. A larger bitstream for GPS tracking is required but only frequents 10 per second. Three to five ultrasonic sensors will be sending analog data as well. Actuation controls with equivalent bitstreams to the number of concurrent commands that the rover could receive, which should be 4 to 6 bytes, needs to also be used to simulate maximum network congestion. The data will likely be transmitted at 9600 Baud to 115200 Baud for reliability, which means that the channels will be required to dynamically allocate for concurrent high and low bandwidth streams which will likely prove that the video will choke out the channels when signal is not strong enough for high-bandwidth applications. With all videos, analog and digital signals, we will attempt to find the limits of the system and adjust accordingly. We hope this testing will provide data allowing us to discontinue the large 10dBi antennas and all traffic for ROS and video to be managed through the substantially smaller 8dBi antennas.

1. **Path Forward**

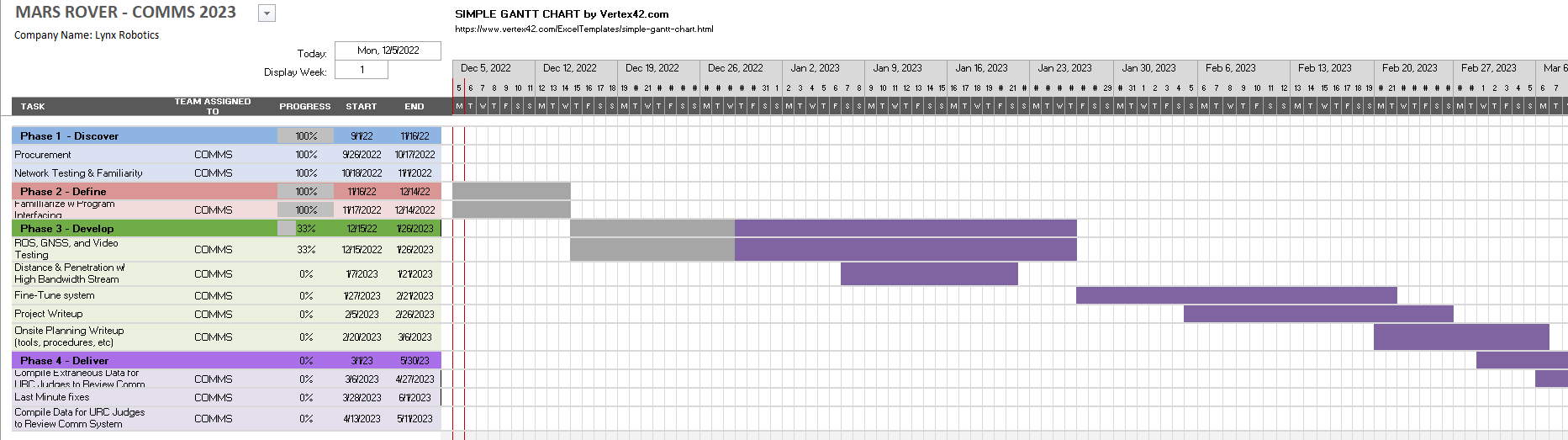


Figure 5: Gantt chart for communications with updated progress

Given the initial projections for the timeline, communications are slightly ahead of schedule, and we hope to continue in this fashion to allocate more manpower for assisting controls and power. After ROS is tested, distance with the communication methods will be tested. Distance testing will be conducted at incremental distances measured preferably measured with the XA1110 GPS for faster testing procedures and measuring the signal strength at each increment. The goal is to have communications established and fully integrated before beginning the second term of the project. At that point we should be ready for concurrent debugging and reallocation of workforce. Given current developments, this seems attainable.

**Bibliography**

[1] Ros, A. (2022, November 15). *Aler9/RTSP-simple-server: Ready-to-use RTSP / RTMP / LL-HLS server and proxy that allows to read, publish and proxy video and audio streams*. GitHub. Retrieved November 26, 2022, from https://github.com/aler9/rtsp-simple-server

[2] Haivision. (2022, January 4). *RTMP vs. SRT: Comparing latency and maximum bandwidth*. Retrieved November 26, 2022, from https://www.haivision.com/resources/white-paper/srt-versus-rtmp/

[3] Cymontkowski , M. (2019). *Haivision/SRT: Secure, reliable, transport*. GitHub. Retrieved November 26, 2022, from https://github.com/Haivision/srt

[4] *Ubuntu (source)Á*. Ubuntu (source) - ROS 2 Documentation: Humble documentation. (n.d.). Retrieved November 26, 2022, from https://docs.ros.org/en/humble/Installation/Alternatives/Ubuntu-Development-Setup.html

**Appendix**

# -\*- coding: utf-8 -\*-

"""

Created on: 11.07.22

author: Ian Dwyer

institution: University of Colorado Denver

"""

""""FOUND 2 CONTEMPORARY FRIIS EQUATIONS USED BUT NOT SURE IF I CALCULATED dBm VS dB CORRECTLY """

import numpy as np

import matplotlib.pyplot as plt

f = 2.440\*10\*\*(9) #middle of 2.4GHz band used

wvlngth= 2.99\*10\*\*(8)/f

d = np.linspace(1,10000,10000) #check to 10km

Pt = 27 #both rockets transmit at this power

Pl = 3 #assumes a 3dB wire loss

Gt = 8 #base station antenna gain (can we use 2 8dBi antennas to get 11 to 16dbi out?)

Gr = 17 #rover antenna gain

GS = -60\*np.ones(10000) #what is considered a good signal by

RS = -90\*np.ones(10000) #receive sensitivity found through forums, no datasheet. Will note findings after characterizing equipment sensitivity (SAME AS QUANTUM!!)

BS = -30\*np.ones(10000) #best signal

# formula found from a missing resource for worstcase since it omits the Gr and adds wire loss

# does not account for attenuation outside of line of site.

Pr\_worst = (Pt-Pl+Gt)+(20\*np.log10(wvlngth/(4\*np.pi\*d)))

# formula found from wiki: contempoarary Friis transmission

Pr\_best = (Pt+Gt+Gr)+(20\*np.log10(wvlngth/(4\*np.pi\*d)))

plt.plot(d,Pr\_worst, label='worst case', linewidth='4')

plt.plot(d,Pr\_best, label='best case', linewidth='4')

plt.plot(RS, label='receive sensitivity',color='red')

plt.plot(GS, label='good signal',color='green')

plt.plot(BS, label='best signal', color='green')

plt.legend()

plt.xscale('log')

plt.grid(True,which='both', ls='-')

plt.xlabel("distance[m]")

plt.ylabel("Receve Power [dB]")

plt.title("Receive Power Over Distance from 8dBi Antenna")

#-30 dBm: This is the maximum signal strength. If you have this measurement, you are likely standing right next to the access point.

#-50 dBm: This is considered an excellent signal strength.

#-60 dBm: This is a good signal strength.

# -67 dBm: This is a reliable signal strength. This is the minimum for any online services that require a reliable connection and Wi-Fi signal strength.

# -70 dBm: This is not a strong signal strength. You may be able to check your email.

# -80 dBm: This is an unreliable signal strength. You may be able to connect to your network, but you will not support most online activity.

**Network and Hardware Setup:**

To setup and connect network, first do a full reset with hardware reset button. Then change ipv4 settings from guide. Next, use Wi-Fi tab and set to AP mode, reduce channel width to 20MHz, set modulation scheme to lower data rates using QPSK based settings as explained in the midterm report.

Graphical user interface

Description automatically generatedTable

Description automatically generated

Figure 6: Station settings for the network

Graphical user interface

Description automatically generatedGraphical user interface, application

Description automatically generated

Figure 7: Access point settings for the network

**The following commands were compiled for the initial video testing –**

To start video server at the Debian-based station, use bash command:

* “*./rtsp-simple-server”*

To start the video stream station, use:

* *“ffmpeg -fflags nobuffer  -re -stream\_loop -1 -video\_size 256x144 -an  -i /dev/video0 -filter:v fps=5 -bitrate 300k  -c:v libx264 -preset ultrafast -tune zerolatency -f rtsp rtsp://localhost:8554/mystream -rtsp\_transport udp”*

To stop stream at the station and free the video resource, use:

* “*sudo kill -9 [PID]”* (find PID using System Monitor, or kill via System Monitor)

For accessing the video stream at the controller AP, use in VLC’s “open network stream”:

* “*rtsp://192.168.42.98:8554/mystream”*