CS CAPSTONE TECHNOLOGY REVIEW

NOVEMBER 10, 2017

30K ROCKET SPACEPORT AMERICA

PREPARED BY

LEVI WILLMETH

Signature *Date*

Abstract

This technology review document compares the technical features of several models of telemetry transmitter, methods of decoding APRS packets, and database solutions to organize and store flight data.

REVISION HISTORY

Name	Date	Reason For Changes	Version
Levi Willmeth	11/3/17	Initial document draft	0.1

CONTENTS

1 PROJECT OVERVIEW

1.1 Introduction

Our project is to design, build, and test software that will fly on board the Oregon State University's entry to the Spaceport America Cup's 30k Challenge. The Spaceport America Cup is an international engineering competition to design, build, and fly a student-made rocket to 30,000 feet. The competition is scored on several criteria including software components like flight avionics, recording and displaying telemetry, and using a scientific research payload.

1.2 My personal role in the team

Our team has many members, including at least 12 mechanical engineering, three electrical engineering, and three computer science students, as well as several faculty and industry mentors. My primary role in the team is to work with my computer science subteam to design, build, and test the ground station software, as well as the flight software that will control avionics for the rocket, and a scientific payload that will be ejected from the rocket at apogee.

We have divided the programming aspect of the project into 4 groups of software: rocket avionics, payload avionics, parsing data, and displaying data. Our goal is to work together so that all three of us contribute to each piece, even though one of us may be called on to take the lead on different pieces at different times. Personally, my primary interests lie in the rocket and payload avionics components, but I also have previous experience in parsing APRS packets and graphing data.

1.3 My expected contributions towards our team goals

This project requires writing avionics software to control events on the rocket, another avionics software to control events on the payload, a ground station software to process incoming experiment or telemetry data and store the results in a database, and software to display information from the database in a browser. Additionally, each program will include a suite of unit tests.

These four goals provide a way of dividing the project into individual portions that can be developed simultaneously. Each of us intends to contribute to all of the portions, but we may also choose to each manage a portion in order to gain experience both leading, and following other team members.

I would like to manage one of the avionics programs, and possibly the parser program. For the rocket avionics, I would be excited to write a kalman filter to exclude errant sensor data and prevent accidental early parachute deployment. For the payload avionics, I would like to write a closed loop pid controller to adjust the motor speed to provide optimum thrust to achieve micro gravity. The parser will also be interesting to work on because it may require parsing a wide variety of sensor data from csv files.

2 TRANSMITTING TELEMETRY DATA

2.1 Overview of telemetry

In this context, telemetry is the data transmitted down from the rocket during flight. This data will be used to calculate the rocket's maximum altitude, as well as locate the rocket after the flight. If something goes wrong during the flight and the rocket is not recovered, telemetry can be used to determine what went wrong so that future flights may avoid the same problem.

2.2 Criteria for telemetry

This project will use a commercial, off the shelf telemetry module to transmit data from the rocket to the ground. There are several commercial products that offer this capability. The important features of a telemetry module are:

- Radio frequency
- Sensor types and transmission rates
- Physical characteristics and costs

2.3 Bigredbee BeeLine GPS

The BeeLine GPS unit is a self contained GPS and telemetry transmitter that operates on the 70cm radio band. It is 1.25″ x 3″, weighs about 2 ounces, and commonly uses a 35cm long antenna. The BeeLine GPS can also record up to 2 hours of GPS data internally. The BeeLine GPS can log and transmit limited positional telemetry fields at 1Hz intervals at 100mW of power, with an advertised range of 40 miles.

The BeeLine GPS transmits an audio signal that needs to be decoded on the ground, using a terminal node controller (TNC). There are several devices that can accomplish this, including using the sound card on a computer. The price varies by vendor, but most hardware TNC's cost over $80. Software TNC's are limited to one signal per sound card, which makes exact pricing difficult to determine. Using a dedicated raspberry pi zero with USB sound card, the cost to process APRS packets with a software TNC should be around $20 per signal.

It is unknown how many signals can be processed by a single computer, but it seems reasonable to assume that processing additional signals will eventually begin to reduce the rate of successful packet captures. In other words, processing each signal with a dedicated computer may add redundancy and performance.

The Bigredbee Beeline GPS costs $259 and requires a handheld radio, which costs around $40. It will also likely require a dedicated computer per signal, which could be a raspberry pi zero with usb sound card for an additional $25 per signal. This brings the total cost to around $324 per signal.

2.4 Altus Metrum TeleMega

The TeleMega is a self contained GPS and telemetry transmitter on the 70cm radio band, that can also function as a flight computer. It measures 1.25″ x 3.25″ and often uses a 35cm long antenna. The data is sent as an audio packet that is decoded by the receiver. The TeleMega can store several hours of data on board, and transmits many types of telemetry at custom intervals, using 40mW of power and an advertised range of 40 miles.

TeleMega data packets are formatted in 32 bit packets to reduce errors, and are designed to be received and decoded by a TeleDongle ground station. The TeleDongle is an additional cost but also provides a convenient serial output over

USB. This is important because it means a single computer can simultaneously receive multiple signals from different transmitters.

The TeleMega costs around $280 and offers a 10% student discount. The TeleDongle is another $100, bringing the total to around $342 per signal.

2.5 RFD900

The RFD900 is a 900 MHz radio designed to work as a pair of serial modems. We could generate a signal on the flight computer, send that signal to a TX RFD900 on the rocket, pick up the signal on a RX RFD900 on the ground, and get whatever telemetry fields we want on the ground. The advantage is that the system would be very customizable. We could send any size packets at any rate we want. The disadvantage is reliability. The RFD900 transmits at a much higher frequency, which reduces rates and has more trouble with interference from atmospheric conditions such as humidity.[1]

The RFD900 costs about $300 for a transmitter/receiver pair, and requires additional processing on the ground, for a total of around $350 per signal. One significant downside to the RFD900 is that it is designed to work as a transmitter/receiver pair, which means we could not easily use redundant receivers. There is an open broadcast mode available that would offer that feature, but there are reasons to believe the RFD900 is not the most reliable choice.

2.6 Comparisons

Model	Radio Frequency	Range	Sensor fields	Transmission rate	Size	Cost
BeeLine GPS	100mW 70 cm	40 miles	Latitude Longitude Altitude	1 Hz	1.25″ x 3″, 2 ounces	$324
TeleMega	40mW 70 cm	40 miles	Latitude Longitude Altitude Ground speed Ascent rate Bearing Gps confidence Time	1 Hz	1.25″ x 3″	$342
			Device ID Flight number Configuration version number Apogee deploy delay Main parachute deploy Radio operator identifier Flight state Battery voltage Pyro battery voltage Calculated height	5 Hz		
			Angle from vertical 100g accelerometer on Z axis 3 axis accelerometer 3 axis gyroscope 3 axis magnetometer Barometer Temperature	10 Hz		
RFD900	1W 900MHz	26 miles	Customizable, any	Any	1.25″ x 2.5″	$300

2.7 Conclusions

Based on all factors, I believe the TeleMega is the best choice for the telemetry module. It provides far more fields of telemetry, offers similar range and reliability, could act as a redundant flight computer if needed, is competitively priced, and is designed and supported by one of our project mentors.

3 Receiving and parsing the telemetry packets

3.1 Overview of telemetry signal formatting

This section depends entirely on the exact model of telemetry transmitter used. Based on several discussions with the ECE subteam, we assume they will select the Bigredbee Beeline GPS telemetry unit, which is limited to sending a tone-based signal formatted as an APRS packet. That means we will need to use a HAM radio to receive the signal and convert it to an audio tone, which then needs to be parsed into a string using a process called Terminal Node Controller (TNC).

3.2 Criteria

The goal of this component is to parse an audio signal into text. As such, the main priority is accuracy and reliability. The parser should be able to correctly and reliably interpret the highest number of packets, even under sub optimal conditions. Multiple options will be judged based on performance (compare identical sample input packets across multiple parsers), reliability (what can go wrong), license or comprehension (open source vs black box), and cost.

3.3 Using a Hardware TNC

For a long time, this problem was solved by electronics hardware. Several HAM radio companies exist and sell units known as hardware TNC's, which process the audio signal into serial text. These have the advantage of being plug-and-play. They convert input to output and the user doesn't really need to know much about what happens inside.

There is an argument to be made that hardware TNC's are not all made using the same algorithm, and that modern software TNC's may be able to decode a greater proportion of 'messy' signals with some amount of noise or interference. (Will address this in the software TNC portion.)

Another downside to a hardware TNC is that it is essentially a black box to us. We will not be able to understand the circuit well enough to make any changes, fixes, or improvements.[2]

There are several hardware TNC's available for sale, listed from most to lease common.

- Kantronics KPC-3+ Packet Communicator, MSRP $199.
- Kantronics KAM, MSRP unknown, no longer commonly sold.
- Kenwood TM-D710A, MSRP $549.
- PacComm Tiny-2 MK-II, MSRP $99.

Another problem with hardware TNC's is that they are not as popular as they once were. It was difficult to find any units for sale to make this list, even using sites like ebay or amazon. Finding several identical and new hardware TNC units for our project may prove difficult or impractical.

3.4 Using a Software TNC

Software TNC's convert audio tones to text, which allow them to perform the same task as a hardware TNC without a physical circuit. Some software TNC's also include features to verify APRS packets, and even decode packets into individual fields. This is ideal for a situation like ours where individual fields could then be inserted into a database.

The downside to using a software TNC is that it requires one sound card per audio signal. Even if an audio splitter were used, multiple simultaneous audio sources would likely increase the number of errors even if additional software were used to attempt to separate the sources. Therefor it seems likely that one computer would be needed per audio source. Using several Raspberry Pi computers may be a good choice because they are small, cheap, and can run headless without the need for a keyboard or monitor.

3.4.1 Direwolf

The most popular software TNC is an open source project called Direwolf. Direwolf is a software "soundcard" AX.25 packet modem/TNC and APRS encoder/decoder. Because Direwolf is open source, it does not need to be a black box to us. We can look inside to understand how it works, make changes as needed, and use only the pieces we need for our project.[3]

Additionally, the developer of Direwolf proposed an experiment to compare it's new algorithm against a hardware TNC, using a pre-recorded 2 hour sample of APRS traffic from the city of Los Angeles. This allows others to run the same experiment and compare their results against other hardware TNC's, or their specific computer hardware including sound card and processor. The results of his experiments showed that the Kantronics KPC-3 Plus successfully decoded 30% fewer packets, and the Kenwood TM-D710A decoded 33% fewer packets than the software TNC.[4]

3.4.2 AGWPE

AGWPE is another software TNC solution. Although it is free, AGWPE is not open source. Instead, it is distributed as an exe file and operates as a black box. It is not clear if it can run in Linux under wine or a virtual machine running Windows, but it is very likely that the additional complexity would present problems for using cheap, distributed redundant receivers using raspberry pi computers. There are no easily available statistics comparing the performance of AGWPE to similar technologies.

3.5 Comparisons

All of these options also require purchasing a portable HAM radio unit and appropriate audio cable for each signal.

Product	Software/Hardware	Performance	Availability	Cost
Kantronics KPC-3+	Hardware	0.7	Limited, used	$199
Kenwood TM-D710A	Hardware	0.67	Limited, used	$549
PacComm Tiny-2	Hardware	?	Limited, used	$99
Direwolf+Pi Zero	Software	1	Open source	$20
AGWPE(Windows laptop)	Software	?	Closed source	$0

3.6 Conclusions

Using this information, I believe that using the Direwolf software TNC is the right choice for our project. Furthermore, I believe that using one Raspberry Pi per audio source is a good way to distribute the signal processing. If using a Pi Zero and USB soundcard, the cost would be around $20 per audio signal. Each Pi would process one telemetry signal and be networked to a central computer, which would collect and store data from each Pi.

4 Storing telemetry and flight data

4.1 Overview of the data to be stored

This project will generate telemetry data from the rocket, telemetry data from the payload, sensor data from the rocket, and sensor data from the payload. All of this data is useful and may be related. For example, using acceleration data from the rocket may clarify or improve results obtained from the payload.

Additionally, the data will need to be used by another program. We intend to view telemetry data and generate dynamic graphs while the rocket is still in the air, and after recovering the rocket we will import sensors and generate a more complete set of graphs using that data.

With these conditions, we believe a database is a good solution for storing all of this data.

4.2 Comparisons

After researching several of the popular relational databases, the following comparison table was made[5][6][7]:

Database	Pros	Cons	Conclusion
SQLite	Easy to set up Stored as a single file	No user/access management Cannot easily function as server	Good for single-user applications
MySQL	Popular, lots of tutorials Fast writes	Not fully SQL compliant Queries using TIME lack precision	Not a bad choice
PostgreSQL	Fully SQL compliant Fast reads	More complex than others	More database than we really need
MariaDB	Active userbase Claims to be faster than MySQL		I like it!

4.3 Conclusions

An active user base and being open source are nice features for any piece of software. It would also be smart to use a popular database so that the skills we learn during this project are more likely to be applicable on future projects.

SQL lite lacks the server functionality we need for this project. PostgreSQL may complicate setup and usage by offering more features than we really need. I don't think it will matter for our scale of application, but PostgreSQL also seems to be optimized for read time, while we will probably want to optimize for write times due to the time-sensitive nature of our incoming data.

After comparing several types of databases, I believe MariaDB fits our needs the best. I like that it's open source and has modern, easy to use guides and support forums. It is SQL compliant so the skills we learn can also be applied to other, similar databases on future projects. MariaDB also supports precise queries for TIME or TIMEDATE fields, which may prove very useful for our type of data.

5 CITATIONS

1) RFD 900+ Modem, http://store.rfdesign.com.au/rfd-900p-modem/
2) http://www.aprs.net/vm/picktnc/picktnc.htm#kpc3
3) https://github.com/wb2osz/direwolf
4) https://github.com/wb2osz/direwolf/blob/master/doc/A-Better-APRS-Packet-Demodulator-Part-1-1200-baud.pdf
5) https://db-engines.com/en/system/MariaDB;PostgreSQL;MySQL;SQLite
6) https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems
7) http://blog.panoply.io/a-comparative-vmariadb-vs-mysql