DEVELOPMENT AND ANALYSIS OF A MOBILE NODE TRACKING ANTENNA CONTROL SYSTEM

Phillip H. Hensley

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

August 2017

APPROVED:

Shengli Fu, Major Professor and Chair of the Department of Electrical Engineering
Yan Wan, Committee Member
Xinrong Li, Committee Member
Costas Tsatsoulis, Dean of the College of Engineering
Victor Prybutok, Dean of the Toulouse Graduate School
A wireless communication system allows two parties to exchange information over long distances. The antenna is the component of a wireless communication system that allows information to be converted into electromagnetic radiation that propagates through the air. A system using an antenna with a highly directional beam pattern allows for high power transmission and reception of data. For a directional antenna to serve its purpose, it must be accurately pointed at the object it is communicating with. To communicate with a mobile node, knowledge of the mobile node’s position must be gained so the directional antenna can be regularly pointed toward the moving target. The Global Positioning System (GPS) provides an accurate source of three-dimensional position information for the mobile node. This thesis develops an antenna control station that uses GPS information to track a mobile node and point a directional antenna toward the mobile node. Analysis of the subsystems used and integrated system test results are provided to assess the viability of the antenna control station.
ACKNOWLEDGEMENTS

Thank you, Dr. Fu, for your guidance and patience in bearing with me through the production of this thesis. Thank you for always having time and advice for me when I poke my head into your office discouraged by another setback.

Thank you, Dr. Li, and Dr. Wan for your willingness to serve as committee members in the approval of this thesis.

Thank you to my loving family, for raising me and instilling in me a desire to pursue higher education. The joy gained from random phone calls and text messages has been invaluable.

Thank you to my roommates, Josh and Thomas, for bearing with me. I hope my many late nights and early mornings were too frustrating to deal with. Thank you both for your daily encouragement and friendship.

Thank you to my friends and colleagues in the Department of Electrical Engineering for good times and mutual bonding over the arduous process of thesis writing!

Thank you to my Providence Chapel church family. Your love, prayers, and fellowship have been so dear to my heart. Thank you for wisdom and guidance from the scriptures. Praise God for His mercy and grace!
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ACKNOWLEDGEMENTS</th>
<th>iii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td>CHAPTER 1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>Wireless Communication Systems</td>
<td>1</td>
</tr>
<tr>
<td>Proposed Solution</td>
<td>2</td>
</tr>
<tr>
<td>CHAPTER 2. LITERATURE REVIEW</td>
<td>4</td>
</tr>
<tr>
<td>CHAPTER 3. ANALYSIS OF SUBSYSTEMS</td>
<td>6</td>
</tr>
<tr>
<td>Microcontroller: ATmega328P</td>
<td>6</td>
</tr>
<tr>
<td>Communication Link: RFM95W Long Range (LoRa™) Radio Transceiver</td>
<td>10</td>
</tr>
<tr>
<td>Theoretical Wireless Communication System Analysis</td>
<td>13</td>
</tr>
<tr>
<td>Position Tracking: Global Positioning System (GPS)</td>
<td>14</td>
</tr>
<tr>
<td>Antenna Pointing and Closed-Loop Motor Control</td>
<td>20</td>
</tr>
<tr>
<td>CHAPTER 4. SYSTEM INTEGRATION AND TEST PROCEDURE</td>
<td>29</td>
</tr>
<tr>
<td>Mobile Node System Overview</td>
<td>29</td>
</tr>
<tr>
<td>Ground Station System Overview</td>
<td>31</td>
</tr>
<tr>
<td>Final Test Procedure</td>
<td>33</td>
</tr>
<tr>
<td>CHAPTER 5. TEST RESULTS AND CONCLUDING REMARKS</td>
<td>35</td>
</tr>
<tr>
<td>Results</td>
<td>35</td>
</tr>
<tr>
<td>Future Work and Improvements</td>
<td>40</td>
</tr>
<tr>
<td>Conclusion</td>
<td>41</td>
</tr>
<tr>
<td>APPENDIX A. MOBILE NODE CODE</td>
<td>43</td>
</tr>
<tr>
<td>APPENDIX B. GROUND STATION CODE</td>
<td>48</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>60</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>WGS84 Defining Parameters [2]</td>
<td>15</td>
</tr>
<tr>
<td>Table 2</td>
<td>Example GPS Sentence Breakdown [6]</td>
<td>17</td>
</tr>
<tr>
<td>Table 3</td>
<td>DC Motor Specifications [13]</td>
<td>21</td>
</tr>
<tr>
<td>Table 4</td>
<td>L293DNE Pin Descriptions [16]</td>
<td>22</td>
</tr>
<tr>
<td>Table 5</td>
<td>Effect of PID Gain on System Response [4]</td>
<td>23</td>
</tr>
<tr>
<td>Table 6</td>
<td>Ziegler-Nichols Results for Azimuth Control Loop</td>
<td>26</td>
</tr>
<tr>
<td>Table 7</td>
<td>Voltage vs Angle for Potentiometer</td>
<td>28</td>
</tr>
<tr>
<td>Table 8</td>
<td>Ziegler-Nichols Results for Elevation Control Loop</td>
<td>28</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Directional Antenna Pointing at Mobile Node</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Proposed System Block Diagram</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>ATmega328P and the Arduino UNO Board</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>UART Block Diagram</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>SPI Block Diagram</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>I²C Block Diagram</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>RFM95W Radio Transceiver</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>HG902PU Omni-Directional Antenna</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>Omni Beam Pattern (Red: Vertical Plane, Black: Horizontal Plane) [10]</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>HG903YE Yagi Antenna</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>Yagi Horizontal Beam Pattern [9]</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>Wireless Communications Diagram</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>MTK3339 GPS Module</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>Received GPS Sentences Shown on PuTTY Software</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>Relationship of GPS coordinates to WGS84</td>
<td>18</td>
</tr>
<tr>
<td>17</td>
<td>ENU vs ECEF Coordinates</td>
<td>19</td>
</tr>
<tr>
<td>18</td>
<td>AER as Defined in a Local ENU Coordinate System</td>
<td>20</td>
</tr>
<tr>
<td>19</td>
<td>HMC5883L Compass</td>
<td>25</td>
</tr>
<tr>
<td>20</td>
<td>Compass Heading Variation</td>
<td>25</td>
</tr>
<tr>
<td>21</td>
<td>EVW-AE4001B14 Potentiometer</td>
<td>26</td>
</tr>
</tbody>
</table>
Figure 22: Potentiometer Internal Mechanism .............................................................. 27
Figure 23: Node Message Format ................................................................................ 29
Figure 24: Mobile Node Flowchart ........................................................................... 29
Figure 25: Mobile Node Wiring Diagram ..................................................................... 30
Figure 26: Mobile Node Picture .................................................................................. 30
Figure 27: Ground Station Flowchart ........................................................................ 32
Figure 28: Ground Station Wiring Diagram .............................................................. 31
Figure 29: Ground Station Picture ............................................................................. 33
Figure 30: Tracking Test Layout and Path ................................................................. 34
Figure 31: RSSI at Ground Station and Mobile Node .................................................. 35
Figure 32: RSSI Comparison – Tracking vs Stationary Ground Station ...................... 36
Figure 33: RSSI vs Range .......................................................................................... 37
Figure 34: Azimuth Tracking Results ......................................................................... 38
Figure 35: Elevation Tracking Results ...................................................................... 38
Figure 36: Commanded Position .............................................................................. 39
Figure 37: Effect of Median Filtering on Compass Heading ...................................... 40
Figure 38: Motor Controller Fire Damage .................................................................. 41
CHAPTER 1

INTRODUCTION

Wireless Communication Systems

A communication system is a system through which two parties exchange information. These systems require a medium through which the information can be transmit. In the context of wireless communications, message signals are transmitted over the air using antennas. The antennas convert a modulated message signal into electromagnetic radiation which propagates from the source to its destination over wireless channel. Upon reception, the antenna converts the electromagnetic wave back into an electrical signal that can be demodulated and interpreted by its recipient.

An important parameter in evaluating the quality of a wireless communication link is the received signal strength indicator (RSSI). RSSI is the measured power of the received signal, often expressed in units of decibel-milliwatts (dBm). RSSI is a quantity that deteriorates with increasing communication range, and in the presence of obstacles that attenuate electromagnetic radiation. Antenna choice strongly influences transmission and reception power. The antennas amplify the message signal when it is transmitted, and when it is received.

The radiation pattern of antenna characterize how signal strength is effected by incoming or outgoing wave direction. An omnidirectional antenna ideally transmits equally in all directions, at the cost of decreased signal gain. Directional antennas have higher signal gain in specific directions, but must be accurately pointed to ensure that desired signals are transmit and received in this main lobe. In a fixed-infrastructure communication system, the antennas only need to be pointed once during installation.
When communicating with a mobile node, the relative angles between antennas are constantly changing, and a mechanically pointing control system is necessary to ensure a directional antenna is pointed at its target.

Proposed Solution

To solve the mobile node tracking problem, a GPS-aided solution is proposed. Implementing a GPS module on both the mobile node and the ground station will provide knowledge of the physical location of both objects. Using the latitudes, longitudes, and altitudes, the relative angles will be calculated. Instead of using a spherical Earth approximation as is commonplace, the proper Earth model that the GPS system uses is implemented to ensure accuracy over long distances.

A directional antenna is placed on the ground station to provide high signal gain. Using an azimuth and elevation motor control system, the antenna will be mechanically pointed toward mobile node (Figure 1).

![Directional Antenna Pointing at Mobile Node](image)

*Figure 1: Directional Antenna Pointing at Mobile Node*

The control system allows the ground station to point at the mobile node as it moves around. A block diagram is given in Figure 2 to provide a brief overview of this process.
The 902-928 MHz frequency band is used to increase both the range and penetration of communication signals. A tradeoff for this decision is that the ground station antenna is larger than that of a 2.4-2.5 GHz antenna and will require stronger motors to move. The system is tested to verify communication system performance and antenna pointing accuracy.
CHAPTER 2
LITERATURE REVIEW

Ganti and Kim have developed an auto-tracking antenna system similar to the one proposed in this paper. The authors tackled the inverse problem in which the mobile node, attached to an unmanned aerial system (UAS), tracks the ground station. The UAS system uses an inertial measurement unit (IMU) to calculate yaw, pitch, and roll angles as the UAS flies. These angles are used in conjunction with GPS data to keep the antenna pointed at the fixed ground station. The authors use this system to prove that placing a directional antenna on the mobile node yields better signal strength than placing the directional antenna on the ground station, and that the best solution would be to place directional antennas on both ends of the communication system.

Soler and Eisemann analyze the effect of using an approximate spherical shape for the Earth rather than its proper ellipsoidal model. Their analysis is performed in the satellite communications field, where pointing inaccuracies as small as thousandths of degrees can drastically reduce communication quality. The authors calculate look angles for several satellites at various geographic locations using both the spherical and ellipsoidal models. They conclude that using a spherical approximation can cause pointing angles to differ by a few hundredths of a degree, more than enough to impact a satellite communication system. The authors also assert that because the Earth is an ellipsoid of revolution, it is more factually correct to use the ellipsoidal model.

Shahrokhi and Zomorrodi conduct a study of PID controller tuning methods. The authors simulate several common tuning methods and compare error values in a set point tracking test. The closed-loop methods tested include Ziegler-Nichols, modified
Ziegler-Nichols, Tyreus-Luyben, and the damped oscillation method. The test results conclude that for set point tracking, the Ziegler-Nichols closed loop tuning method yields minimal error and can be “confidently used for a majority of systems” in which the process model is not known.

Ekstrom proposes a median filtering algorithm as a method of minimizing the effect of a random large data value on a running average calculation. A median filter involves numerically sorting the data from least to greatest and picking the middle value. Because of the nature of this average calculation, an unexpected outlier will be either at the lowest or highest end of the sorted list, and will be filtered out. The author acknowledges that a classical mean average is more effective in a scenario where a proper statistical distribution can be used to model these outlying samples, but in scenarios where the data may behave unexpectedly, the median filter is a preferred choice.

Bakibillah, Rahman, and Zaman assess the viability of using PWM for a closed loop DC motor control system. Their study involves monitoring DC motor speed as output load is varied. The authors verify that a closed loop control system will allow DC motor speed to remain constant under a dynamic load. As the load is increased, the duty cycle of the PWM waveform is increased and motor output speed is kept constant.
CHAPTER 3
ANALYSIS OF SUBSYSTEMS
Microcontroller: ATmega328P

Features and Relevant Specifications

The ATmega328P (Figure 3) is used as the central microcontroller that receives peripheral data, performs calculations, and issues system commands. The system clock rate is 16 MHz, allowing complicated instructions to be performed rapidly [1]. The microcontroller is equipped with 14 digital input/output pins and 6 analog input pins that can be configured to interface with other devices. Communication protocols such as Serial Peripheral Interface (SPI), Universal Asynchronous Receive/Transmit (UART), and Inter-Integrated Circuit (I²C) are configured using input/output pins specified in the product datasheet. An internal voltage regulator allows the microcontroller to serve as a 5V power supply for attached components.

Programs for the microcontroller are written in the Arduino language which contains elements of both the C and C++ programming languages. Programs are
written, compiled, and uploaded to the microcontroller using the Arduino Integrated Development Environment (IDE). The open-source nature of the Arduino platform makes it an attractive choice for rapidly prototyping ideas. Commands and functions are written at a high level, simplifying programming, but limiting lower level configuration that can be achieved using other microcontrollers such as an ARM Cortex-M4F based microcontroller.

Universal Synchronous and Asynchronous Serial Receiver/Transmitter (USART)

The Atmega328P is capable of both synchronous and asynchronous serial communication [1]. When using only the asynchronous functionality the name is shortened to UART. UART communication lacks a shared clock connection between devices, making it an asynchronous form of communication. UART requires data to be broken into 8 bit frames (often two 4-bit ASCII characters) that are then sent one bit at a time from source to destination. The bit rate of the source and destination must be match or data will not be received accurately. The bit rates are determined in the main program before serial communication begins. UART is the method of communication used in this project for transmission of GPS coordinates from the GPS module to the Atmega328P microcontroller at a rate of 9600 bits per second as illustrated in Figure 2.
Serial Peripheral Interface (SPI)

SPI is the protocol used to interface between the microcontroller and the wireless radio module (RFM95W) for data transmission and reception. SPI is a synchronous protocol used to establish communicate between a microcontroller (Master) and an external peripheral (Slave) [1]. SPI communication is configured using three pins: Master Out – Slave In (MOSI), Master In – Slave Out (MISO), and Serial Clock (SCK) as illustrated in Figure 5. Communication over this protocol is full duplex, in that both devices may receive and transmit at the same time over two different physical lines. The master generates a clock signal that is shared with the slave over the SCK pin to ensure that both the MOSI and MISO link are always in sync.
Inter-Integrated Circuit Bus

Inter-Integrated Circuit (I\(^2\)C) is a bi-directional two-wire communication bus [11]. I\(^2\)C follows a master-slave relationship, where the master is responsible for generating the serial clock line (SCL) and sending to the slave over a wire. In addition, a serial data line (SDA) provides for the bi-directional communication of information. I\(^2\)C is the bus used for communication between the ATmega238P and the compass module (HMC5883L) as illustrated in Figure 6.
Communication Link: RFM95W Long Range (LoRa™) Radio Transceiver

LoRa™ Overview

LoRa™ is a modulation scheme owned by HopeRF that “uses spread spectrum modulation and forward error correction techniques to increase the range and robustness of radio communication links compared to traditional FSK or OOK based modulation” [8]. Spread spectrum describes the process of “representing each bit of information by multiple chips of information” to help resist signal noise interference. Forward error correction involves adding redundant bits to the message signal, again to eliminate the impact of noise and interference. The LoRa™ modulation scheme is implemented on the RFM95W radio transceiver (Figure 7) used in this project.

![RFM95W Radio Transceiver](image)

Figure 7: RFM95W Radio Transceiver

The RFM95W module operates at 915MHz with data rates of up to 300kbps. This module boasts a transmission power that can be varied between 5dBm and 23dBm, a receiver sensitivity of -148dBm, and the ability to connect an external antenna, making it a suitable choice for long range communications at a moderate data.
Antenna Selection

*Omni-Directional Antenna: HyperLink HG902PU*

An omni-directional antenna was chosen for the mobile node so that it ideally transmits the same power in all directions regardless of the mobile node’s orientation. An embedded PCB antenna (Figure 8) was chosen because of its small form factor making it easily portable. The HG902PU has a gain of 2dBi for signals in the 860-960MHz range and a beam pattern shown in Figure 9.

*Figure 8: HG902PU Omni-Directional Antenna*

*Figure 9. Omni Beam Pattern (Red: Vertical Plane, Black: Horizontal Plane) [10]*
Yagi Antenna: HyperGain HG903YE-NF

A Yagi antenna (Figure 10) was selected for the ground station because of its high gain (3dBi) and directional beam. The goal of the ground station will be to keep this antenna pointed toward the mobile node as accurately as possible. The large main lobe on the radiation pattern (Figures 11 and 12) helps compensate for any pointing or tracking inaccuracies.

Figure 10: HG903YE Yagi Antenna

Figure 11: Yagi Horizontal Beam Pattern [9]
Theoretical Wireless Communication System Analysis

The block diagram for the wireless communication system is shown in Figure 13.

Based on this configuration, a calculation of theoretical may be formed as follows:

\[ P_{t} = +17\, dBm = 50\, mW \quad (\text{Maximum Transmitter Power}) \]

\[ P_{r} = -148\, dBm = 1.58 \times 10^{-15}\, mW \quad (\text{Receiver Sensitivity}) \]

\[ G_t = 2\, dBi \sim \text{Gain Magnitude of 1.58 (Omni Antenna Gain)} \]

\[ G_r = 3\, dBi \sim \text{Gain Magnitude of 2 (Yagi Antenna Gain)} \]

\[ f = 915\, MHz \quad (\text{Communication Frequency}) \]

\[ c = 3 \times 10^{8}\, m/s \quad (\text{Speed of Light}) \]
The Friis transmission equation (assuming L=1 and n=2 for free space) provides an estimation for received power [12]:

\[ P_{r \min} = \frac{P_{t \max} G_t G_r c^2}{(4\pi)^2 f^2 d_{\max}^2} \]  

(3.1)

Perform algebraic manipulation to solve for maximum distance.

\[ d_{\max} = \sqrt{\frac{P_{t \max} G_t G_r c^2}{(4\pi)^2 f^2 P_{r \min}}} \]  

(3.2)

Plug in specified values to find maximum theoretical transmission distance.

\[ d_{\max} = \sqrt{\frac{(50 \times 10^{-3} W)(1.58)(2)(3 \times 10^8 m/s)^2}{(4\pi)^2 (915 \times 10^6 Hz)^2 (1.58 \times 10^{-15} W)}} \]

\[ d_{\max} = 260909.743 \text{ m} = 260.910 \text{ km} = 162.12 \text{ mi} \]

A similar calculation can be performed for maximum transmission distance at minimum transmitter power.

\[ P_{t \min} = -18 dBm = 1.585 \times 10^{-5} W \]

\[ d_{\min} = \sqrt{\frac{P_{t \min} G_t G_r c^2}{(4\pi)^2 f^2 P_{r \min}}} \]  

(3.3)

\[ d_{\min} = \sqrt{\frac{(1.585 \times 10^{-5} W)(1.58)(2)(3 \times 10^8 m/s)^2}{(4\pi)^2 (915 \times 10^6 Hz)^2 (1.58 \times 10^{-15} W)}} \]

\[ d_{\min} = 2071.840 \text{ m} = 2.072 \text{ km} = 1.288 \text{ mi} \]

Position Tracking: Global Positioning System (GPS)

GPS and WGS84 Overview

The Global Positioning System uses satellites in medium Earth orbit to cover the globe and provide fixed reference positions for receivers on the ground. The GPS
receiver and satellite both generate identical codes that the satellite then sends to the receiver [3]. Upon reception, the receiver measures the time delay between the two signals and can then calculate the distance between itself and the satellite. By repeating this process for multiple overhead satellites, the GPS receiver can gain a more accurate knowledge of its latitudinal and longitudinal position on the globe. To get an accurate reading of latitude and longitude, a minimum of three satellites must be visible overhead. To add the extra altitude dimension (resulting in a knowledge of relative position in all three dimensions), a fourth satellite is necessary.

GPS uses the World Geodetic System 1984 (WGS84), to establish an accurate ellipsoidal shape for the Earth rather than using a spherical approximation. The WGS84 system has its origin at the center of mass of the Earth, and has several parameters (Table 1) to describe the ellipsoid.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Notation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semi-major Axis</td>
<td>a</td>
<td>6378137.0 meters</td>
</tr>
<tr>
<td>Semi-minor Axis</td>
<td>b</td>
<td>6356752.3142 m</td>
</tr>
<tr>
<td>Reciprocal of Flattening</td>
<td>1/f</td>
<td>298.257223563</td>
</tr>
<tr>
<td>First Eccentricity</td>
<td>e</td>
<td>8.1819190842622*10^{-2}</td>
</tr>
<tr>
<td>First Eccentricity Squared</td>
<td>e^2</td>
<td>6.69437999014*10^{-3}</td>
</tr>
<tr>
<td>Angular Velocity of the Earth</td>
<td>v</td>
<td>7292115.0*10^{-11} rad/s</td>
</tr>
<tr>
<td>Earth’s Gravitational Constant</td>
<td>GM</td>
<td>3986004.418*10^8 m^3/s^2</td>
</tr>
</tbody>
</table>

*Table 1: WGS84 Defining Parameters [2]*
MTK3339 Chipset

The MTK3339 GPS module (Figure 14) is a highly sensitive (-165dBm) GPS receiver with a variable update rate (1Hz-10Hz).

Upon tracking satellites, the GPS module sends NMEA strings (Figure 15) to the microcontroller using the UART protocol. These strings contain information about time, position, and number of satellites tracked (Table 2). Once extracted from the sentences, the latitude, longitude, and altitude are values are used to provide the physical location of both the mobile node and ground station. Using these two sets of position information, an azimuth and elevation angle between the objects is calculated and used to issue motor control commands.

![MTK3339 GPS Module](image)

*Figure 14: MTK3339 GPS Module*

![Received GPS Sentences Shown on PuTTY Software](image)

*Figure 15: Received GPS Sentences Shown on PuTTY Software*
Received NMEA sentence:
$GPGGA,183056.400,3313.5277,N,09709.0535,W,2,05,2.47,179.7,M,24.0,M,0000,0000*56

<table>
<thead>
<tr>
<th>Name</th>
<th>Example</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message ID</td>
<td>$GPGGA</td>
<td></td>
<td>GGA protocol header</td>
</tr>
<tr>
<td>UTC Time</td>
<td>183056.400</td>
<td></td>
<td>hhmmss.sss</td>
</tr>
<tr>
<td>Latitude</td>
<td>3313.5277</td>
<td>dddmm.mmmm</td>
<td></td>
</tr>
<tr>
<td>N/S Indicator</td>
<td>N</td>
<td></td>
<td>N=north or S=south</td>
</tr>
<tr>
<td>Longitude</td>
<td>09709.0535</td>
<td>dddmm.mmmm</td>
<td></td>
</tr>
<tr>
<td>E/W Indicator</td>
<td>W</td>
<td></td>
<td>E=east or W=west</td>
</tr>
</tbody>
</table>
| Position Fix Indicator | 2     |                | 0 Fix not available  
1 GPS Fix  
2 Differential GPS Fix |
| Satellites Used    | 5       | Range 0 to 14   | Range 0 to 14                                    |
| HDOP               | 2.47    | Horizontal Dilution of Precision                  |
| MSL Altitude       | 179.7   | meters         | Antenna Altitude above/below mean-sea-level      |
| Units              | M       | meters         | Units of antenna altitude                        |
| Geoidal Separation | 24.0    | meters         | Units of geoids separation                       |
| Age of Diff. Corr. |         | second         | Null fields when DGPS is not used                 |
| Checksum           | *56     |                | End of message termination                       |

Table 2: Example GPS Sentence Breakdown [6]

Look Angle Calculations

Knowledge of the ground station and mobile node’s GPS coordinates can be used to calculate relative angles, but first, multiple coordinate system conversions must take place to guarantee accurate calculations [15]. Using the ground station’s longitude and latitude in combination with the WGS84 model of the Earth, the ground station’s position may be converted into an earth-centered, earth-fixed (ECEF), rectangular coordinate system (Figure 16).
Figure 16: Relationship of GPS coordinates to WGS84

Applying equations 3.4, 3.5, and 3.6 with parameters defined in Table 1 and Figure 14 yields the desired ECEF coordinates.

\[
\begin{bmatrix}
X \\
Y \\
Z \\
\end{bmatrix} = \\
\begin{bmatrix}
\frac{a}{\sqrt{1 - e^2 \sin^2(\phi)}} + h \cdot \cos(\phi) \cdot \cos(\lambda) \\
\frac{a}{\sqrt{1 - e^2 \sin^2(\phi)}} + h \cdot \cos(\phi) \cdot \sin(\lambda) \\
\left(\frac{a \cdot (1 - e^2)}{\sqrt{1 - e^2 \sin^2(\phi)}} + h\right) \cdot \sin(\phi) \\
\end{bmatrix}
\]

(3.4)

where \(h\) is the height above the Earth's surface

By applying the ECEF transformation to both the mobile node and ground station, a relative displacement vector between the two objects can be calculated by subtracting their coordinates.

\[
\begin{bmatrix}
X_{\text{relative}} \\
Y_{\text{relative}} \\
Z_{\text{relative}} \\
\end{bmatrix} = \\
\begin{bmatrix}
X_{\text{node}} \\
Y_{\text{node}} \\
Z_{\text{node}} \\
\end{bmatrix} - \\
\begin{bmatrix}
X_{\text{ground}} \\
Y_{\text{ground}} \\
Z_{\text{ground}} \\
\end{bmatrix}
\]

(3.5)

Using the displacement vector, the ECEF coordinates can be transformed into a local East-North-Up (ENU) coordinate system centered at the ground station, as pictured in Figure 17.
This coordinate transformation is performed by applying a rotational matrix to the displacement vector.

$$\begin{bmatrix} -\sin(\lambda_{\text{ground}}) & \cos(\lambda_{\text{ground}}) & 0 \\ -\sin(\phi_{\text{ground}} \cos(\lambda_{\text{ground}})) & -\sin(\phi_{\text{ground}} \sin(\lambda_{\text{ground}})) & \cos(\phi_{\text{ground}}) \\ \cos(\phi_{\text{ground}} \cos(\lambda_{\text{ground}})) & \cos(\phi_{\text{ground}} \sin(\lambda_{\text{ground}})) & \sin(\phi_{\text{ground}}) \end{bmatrix}$$

$$\begin{bmatrix} E \\ N \\ U \end{bmatrix} = [R] \begin{bmatrix} X_{\text{relative}} \\ Y_{\text{relative}} \\ Z_{\text{relative}} \end{bmatrix}$$

Using the local ENU coordinate system, the relative azimuth, elevation, and range (AER) parameters (Figure 18) can be calculated using basic trigonometry.

$$\begin{bmatrix} \text{Azimuth} \\ \text{Elevation} \\ \text{Range} \end{bmatrix} = \begin{bmatrix} \tan^{-1} \left( \frac{E}{N} \right) \\ \sin^{-1} \left( \frac{U}{\sqrt{E^2 + N^2 + U^2}} \right) \\ \sqrt{E^2 + N^2 + U^2} \end{bmatrix}$$
The calculated azimuth and elevation angles are used by the ground station’s microcontroller to issue movement commands to the motor control system.

**Antenna Pointing and Closed-Loop Motor Control**

**Direct Current Motors**

A Direct Current (DC) motor is a device that converts electrical to mechanical energy. Applying a DC voltage across the input terminals, a magnetic field is produced around the armature. Stationary magnets interact with this magnetic field and produce a rotational force. This rotational force is delivered to an external load through an output shaft. Varying the voltage across the input terminals results in a change in output speed and torque. The speed and torque can be further altered using a gearbox to produce the desired mechanical output. DC motors lack a position feedback mechanism and an external source of feedback must be used to gain knowledge of motor position.
Two planetary gear brushed DC motors (RobotZone #638242) are used to point the ground station antenna toward targeted angles. One motor is dedicated to azimuth angle control, and the other is dedicated to elevation angle control. Relevant motor specifications are listed in Table 3.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Voltage Range</td>
<td>6-12</td>
<td>VDC</td>
</tr>
<tr>
<td>Rated Voltage</td>
<td>12</td>
<td>VDC</td>
</tr>
<tr>
<td>Rated Load</td>
<td>6</td>
<td>kgf-cm</td>
</tr>
<tr>
<td>Max. No-load current</td>
<td>0.21</td>
<td>A</td>
</tr>
<tr>
<td>No-load Speed</td>
<td>26±3</td>
<td>Rpm</td>
</tr>
<tr>
<td>Rated-load Current</td>
<td>0.48</td>
<td>A</td>
</tr>
<tr>
<td>Rated-load Speed</td>
<td>24±2</td>
<td>rpm</td>
</tr>
<tr>
<td>Min. Stall Torque</td>
<td>42</td>
<td>kgf-cm</td>
</tr>
<tr>
<td>Max. Stall Current</td>
<td>4.9</td>
<td>A</td>
</tr>
<tr>
<td>Gear Type</td>
<td>Planetary</td>
<td></td>
</tr>
<tr>
<td>Gear Ratio</td>
<td>1/455</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: DC Motor Specifications [13]

Pulse-Width Modulation (PWM)

DC motor speed can also be controlled using Pulse-Width Modulation (PWM). Rather than using a variable voltage supply to change the motor speed, PWM cycles between a high pulse (12VDC) and low (0VDC) pulse. The relationship between high pulse width and total pulse time is referred to as duty cycle.

\[
Duty\ Cycle = \frac{High\ Pulse\ Width}{High\ Pulse\ Width + Low\ Pulse\ Width} = \frac{High\ Pulse\ Width}{On/Off\ Cycle\ Period} \quad (3.9)
\]

Increasing the duty cycle increases the DC motor output speed. The inverse relationship also holds true. The ATmega328P has output pins that can be configured for PWM [1]. The PWM-capable pins are used to control elevation and azimuth motor speeds. The duty cycle of the output may be varied at any time by the microcontroller by
changing the value of a stored variable.

H-Bridge Motor Controller: L293DNE

The low power from the ATmega328P pins is insufficient for driving a motor system under load. In addition, without the use of additional hardware it is not possible to reverse motor direction. As a result, a dedicated motor control module with an external power source is necessary. The L293DNE is a quadruple half-H driver circuit capable of supplying 600mA of current to both the elevation (channel 1) and azimuth (channel 2) motors at the same time [16]. The motor controller is also responsible for reversing the polarity across the motor terminals to switch the direction of the output shaft. The description of each pin of the motor controller is given in Table 4. The logical input pins that toggle motor direction are controlled by GPIO pins from the microcontroller.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2 EN</td>
<td>Channel 1 enable pin. Handles PWM input to determine motor output speed</td>
</tr>
<tr>
<td>1A, 2A</td>
<td>Channel 1 logical inputs. Set one high and the other low to determine motor direction.</td>
</tr>
<tr>
<td>1Y, 2Y</td>
<td>Channel 1 output pins. Connect to the two terminals of a motor</td>
</tr>
<tr>
<td>3, 4 EN</td>
<td>Channel 2 enable pin. Handles PWM input to determine motor output speed</td>
</tr>
<tr>
<td>3A, 4A</td>
<td>Channel 2 logical inputs. Set one high and the other low to determine motor direction.</td>
</tr>
<tr>
<td>3Y, 4Y</td>
<td>Channel 2 output pins. Connect to the two terminals of a motor</td>
</tr>
<tr>
<td>GND</td>
<td>Common ground and heatsink</td>
</tr>
<tr>
<td>VCC1</td>
<td>5V supply for internal logic</td>
</tr>
<tr>
<td>VCC2</td>
<td>12V supply for driving channel 1 and channel 2 motors</td>
</tr>
</tbody>
</table>

*Table 4: L293DNE Pin Descriptions [16]*
Proportional-Integral-Derivative Control

Proportional-Integral-Derivative (PID) is a method of control for linear, time-invariant control systems. PID control requires knowledge of the differential equations governing system dynamics and a feedback mechanism that measures system output. The output is then subtracted from the desired input to generate an error signal. The error signal is operated on by a proportional, integral, and derivative controller before new system output is generated. The advantage of PID control is that it allows control of overshoot, settling time, and steady-state error. PID control is mathematically described in the time domain by equation 3.10 [4].

\[
Output = K_P e(t) + K_I \int e(t) \, dt + K_D \frac{de(t)}{dt}
\]

where \(K_P, K_I, \text{and } K_D\) are the proportional, integral, and derivative gains and \(e(t)\) is the error signal

Changing the gains in (3.10) produces effects that can be approximately described by Table 5.

<table>
<thead>
<tr>
<th>Increasing (K_P)</th>
<th>Percent Overshoot</th>
<th>Settling Time</th>
<th>Steady-State Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increasing (K_I)</td>
<td>Increases</td>
<td>Minimal Impact</td>
<td>Decreases</td>
</tr>
<tr>
<td>Increasing (K_D)</td>
<td>Decreases</td>
<td>Decreases</td>
<td>No Impact</td>
</tr>
</tbody>
</table>

\textit{Table 5: Effect of PID Gain on System Response [4]}

Ziegler-Nichols Tuning Method

For a system which cannot easily be mathematically described, an iterative method for obtaining the PID gains must be used. The Ziegler-Nichols method (paper for citing this as a viable choice?) is viable. The Closed-Loop Ziegler-Nichols process may be described as follows [4]:
1. Set $K_I$ and $K_D$ to 0

2. Increase $K_P$ until the system reaches the boundary of instability (indefinite oscillations of constant amplitude)

3. Define $K_U$, the ultimate gain, to be the same as this $K_P$ value

4. Measure the period of the oscillations (time between peaks)

5. Define $T_U$, the ultimate period, to be the same as the measured period

6. Calculate $K_P$, $K_i$, and $K_D$ using the ultimate gain and period

   a. $K_P = 0.6 \times K_U$  \hspace{1cm} (3.11)

   b. $K_i = \frac{1.2 \times K_U}{T_U}$  \hspace{1cm} (3.12)

   c. $K_D = \frac{0.6 \times K_U \times T_U}{0.8}$  \hspace{1cm} (3.13)

7. Apply calculated gains to motor control loop

8. Repeat process for both elevation and azimuth control loops

Azimuth Control Loop

*Compass: HMC5883L*

The azimuth angle as defined by the local ENU coordinate system is equivalent to a magnetic compass heading. Using a compass module (HMC5883L) to track ground station heading provides a source of feedback for azimuth motor control. The commanded angles are given as deviations north in the same way that a compass provides deviation from magnetic north. The azimuth motor operates until targeted angle matches current heading.
The HMC5883L (Figure 19) is a triple-axis magnetometer that communicates with the ATmega328P through I\(^2\)C. The module senses the magnetic field in three perpendicular directions and uses these measurements to calculate a compass heading [7]. The compass heading is then sent to the microcontroller for usage in the azimuth control loop.

![HMC5883L Compass](image)

*Figure 19: HMC5883L Compass*

The sampled compass heading (Figure 20) varies rapidly with time. It can be seen from the plot of sampled values that the range of compass values is about 2.5 degrees. As such, azimuth accuracy cannot be guaranteed without specifying an allowable error threshold. The range of values was taken, doubled, and used to define an allowable error threshold of 5 degrees.

![Compass Heading Variation](image)

*Figure 20: Compass Heading Variation*
**Ziegler-Nichols Tuning Results**

The Ziegler-Nichols tuning method was followed for the azimuth control loop and produced the results in Table 6. The tuning method provided a helpful starting point for determining final tweaked values for gains.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Ziegler Nichols Values</th>
<th>Final Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ultimate Gain $K_U$</td>
<td>3</td>
<td>N/A</td>
</tr>
<tr>
<td>Ultimate Period $T_U$</td>
<td>0.9 seconds</td>
<td>N/A</td>
</tr>
<tr>
<td>$K_p$</td>
<td>1.8</td>
<td>3</td>
</tr>
<tr>
<td>$K_i$</td>
<td>4</td>
<td>0.9</td>
</tr>
<tr>
<td>$K_D$</td>
<td>2.025</td>
<td>1</td>
</tr>
</tbody>
</table>

*Table 6: Ziegler-Nichols Results for Azimuth Control Loop*

**Elevation Control Loop**

*Feedback Mechanism: EVW-AE4001B14 Potentiometer:*

The method used for providing elevation feedback is a potentiometer (EVW-AE4001B14) pictured in Figure 21. A potentiometer behaves like a voltage divider with a

*Figure 21: EVW-AE4001B14 Potentiometer*
variable resistor. As the shaft of the potentiometer is rotated, an internal wiper rotates, changing the resistance between the middle and outer fixed terminals (Figure 22).

![Potentiometer Internal Mechanism](image)

*Figure 22: Potentiometer Internal Mechanism*

This particular potentiometer was chosen for its unique property of having a through-hole actuator instead of an output shaft. The hole of the potentiometer slides over the output shaft of the elevation motor, and rotates as the motor turns. The potentiometer also has the advantage of a ±2.5% linearity error allowing for a reliable input/output relationship to be developed [17].

By keeping a constant 5V DC across the outer terminals and measuring the potential at the middle terminal with respect to ground, a relationship between angular rotation and voltage has been empirically constructed. A summary of the measurements and derived linear equation are presented in Table 7.

<table>
<thead>
<tr>
<th>Measured Angle (Degrees)</th>
<th>Measured Voltage at Middle Terminal (Volts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.3887 V</td>
</tr>
<tr>
<td>90</td>
<td>4.7217 V</td>
</tr>
</tbody>
</table>
Table 7: Voltage vs Angle for Potentiometer

The potentiometer voltage is measured by the ATmega328P, then equation 3.14 is applied, and the resulting angle is passed into the elevation control loop. To account for potentiometer resolution and a small amount of mechanical instability in the physical elevation axis, an allowable error threshold of 3 degrees is defined for the elevation control loop.

Ziegler-Nichols Tuning Results

The Ziegler-Nichols tuning method was followed for the elevation control loop and produced the results in Table 8. The tuning method was not as helpful on the elevation control, and the final values were heavily modified to produce desirable performance.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Ziegler Nichols Values</th>
<th>Final Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ultimate Gain $K_U$</td>
<td>12</td>
<td>N/A</td>
</tr>
<tr>
<td>Ultimate Period $T_U$</td>
<td>0.24 seconds</td>
<td>N/A</td>
</tr>
<tr>
<td>$K_p$</td>
<td>7.2</td>
<td>3.6</td>
</tr>
<tr>
<td>$K_I$</td>
<td>60</td>
<td>0.2</td>
</tr>
<tr>
<td>$K_D$</td>
<td>2.16</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 8: Ziegler-Nichols Results for Elevation Control Loop
CHAPTER 4
SYSTEM INTEGRATION AND TEST PROCEDURE

Mobile Node System Overview

The mobile node functions as a beacon that regularly broadcasts its GPS coordinates. When the node receives updated coordinates from the GPS module, it organizes its latitude, longitude, altitude, and last received signal strength into a message (Figure 23) that is sent to the LoRa™ module for modulation and transmission. A flowchart of the node’s logic is found in Figure 24 and a wiring diagram of components in Figure 25. The node is externally powered by a 4xAA battery pack connected to the onboard DC barrel jack. The physical system is pictured in Figure 26.

![Node Message Format](image1.png)

*Figure 23: Node Message Format*

![Mobile Node Flowchart](image2.png)

*Figure 24: Mobile Node Flowchart*
Figure 25: Mobile Node Wiring Diagram

Figure 26: Mobile Node Picture
Ground Station System Overview

The ground station receives GPS coordinates from the mobile node through the wireless communication system. Using knowledge of its own GPS coordinates and the node’s coordinates, the ground station calculates relative look angles. The calculated look angles are fed into the azimuth and elevation control loops to move the antenna to the calculated position within a previously defined error tolerance. A wiring diagram is given in Figure 27. The flowchart of this process is illustrated in Figure 28, and picture of the physical system in Figure 29.

Figure 28: Ground Station Wiring Diagram
Figure 28: Ground Station Flowchart
Final Test Procedure

Two tests were performed to evaluate final system performance. Both tests were conducted on the same day with minimal cloud coverage to ensure GPS accuracy. The purpose of the tests is to assess pointing accuracy, and its effect on communication system performance. A test procedure is followed once using the tracking ground station and then repeated using a non-tracking omnidirectional antenna. The test setup is as follows:

1. Lay flags in a field approximately 50 meters apart to designate test points
   a. Google maps was used to provide a rough estimate of an expected heading value between the ground station and each test point
2. Place ground station in a fixed location, and allow ample time to acquire a strong GPS signal
3. Power on mobile node and allow ample time to acquire a strong GPS signal

4. Enable automatic tracking functionality of ground station

5. Walk a predetermined path (Figure 30) between test points while carrying the mobile node

6. Record the following data at each test point: Mobile node latitude, mobile node longitude, mobile node altitude, target azimuth angle, target elevation angle, actual azimuth angle, actual elevation angle, received signal strength at ground station, received signal strength at mobile node, and range

7. Repeat test procedure with a stationary omnidirectional antenna

Figure 30: Tracking Test Layout and Path
CHAPTER 5
TEST RESULTS AND CONCLUDING REMARKS

Results

After the tests were performed, the PuTTY log files were exported to Microsoft Excel for processing. The received signal strength values at the tracking ground station and mobile node (Figure 31) were plotted against each other to illustrate the strength of the wireless communication link between the two devices.

![Received Signal Strength](image)

Figure 31: RSSI at Ground Station and Mobile Node

The received signal strength values from the omnidirectional station were also plotted to demonstrate the enhanced performance gained by the tracking ground station (Figure...
Using the automated tracking antenna yields up to a 22dBm improvement in received signal strength, a much stronger connection at the same range.

![Received Signal Strength Graph](image)

**Figure 32: RSSI Comparison – Tracking vs Stationary Ground Station**

The received signal strength for each system was plotted against range (Figure 33) to illustrate how the communication link is changed as distance increases.
Figure 33: RSSI vs Range

The azimuth angle data was grouped together to evaluate tracking in the horizontal plane (Figure 34). The expected azimuth line sits directly underneath the target azimuth line, showing that the calculated azimuth angles between the ground station and mobile node closely match reality. The ground station heading is defined to be the compass heading after the control loop has finished moving the station to the target position. The heading values stay within the five-degree error tolerance as defined in the azimuth control loop. The elevation angle data was also grouped together to yield the results in Figure 35. The ground station angle stays within the three-degree error tolerance as defined in the elevation control loop.
A plot of error between target angles and actual ground station angles is included in Figure 36 for discussion about sources of error and how to improve accuracy. Error in this context is defined as the difference between target angle and ground station angle.
Much of the error seen is due to the generous error thresholds defined in the motor control loops. These large error thresholds were included to limit the amount of oscillation when pointing at the target. The ground station can reach a final position faster than it would with smaller error thresholds. The error thresholds also compensate for noisy feedback data. The compass and sampled potentiometer voltage are subject to frequent variation. By defining an error threshold that is outside the range of variation, the effect of outlier samples is limited. Investing in sensors with higher resolution and applying filtering techniques would allow these error thresholds to be decreased. Some of the variation can also be attributed to mechanical instability. The ground station has a bit of play in both the azimuth and elevation axes that can restrict how accurately the antenna can be pointed.
Future Work and Improvements

Several changes and improvements can be made for a second revision of this system. A complete overhaul of the mechanical structure would allow for a sturdier system to be created. Instead of using off-the-shelf aluminum components, precise measurements and more exact structural analysis would create a system with better dynamics.

Adding digital filtering techniques to the compass and potentiometer readings would provide a steadier tracking signal. For example, applying a median filter to the compass heading helps reject outlying compass headings that disturb smooth tracking. A median filter was applied in early revisions of the system, but the increased calculation time resulted in a control system that would never reach steady-state. The smoothing effect of the applied median filter is shown in Figure 37. A microcontroller with a faster clock would allow this filtering technique to be fully implemented.

![Compass Heading Variation](image)

*Figure 37: Effect of Median Filtering on Compass Heading*
Using a motor controller with a higher power limit, or two separate motor controllers for azimuth and elevation control would allow both control loops to be executed simultaneously instead of sequentially. Both loops were run simultaneously with the current hardware, but the increased power draw caused the motor controller to catch fire (Figure 38).

![Motor Controller Fire Damage](image)

*Figure 38: Motor Controller Fire Damage*

Improving elevation control requires a more accurate determination of altitude. The GPS altitude varied by as much as twenty meters at times, heavily skewing target elevation angle calculations. A barometric pressure sensor could be used to get a more accurate altitude reading, but the system would need to adapt to different atmospheric and weather conditions. Future verification of the elevation pointing accuracy would involve attaching the mobile node to a drone and following a similar test procedure but with altitudinal variation.

**Conclusion**

A mobile node tracking ground station has been proposed, designed, implemented, and tested. It has been verified that keeping a directional antenna consistently pointed at a mobile node provides a better communication system than one based purely on omnidirectional antennas. The increased received signal strength at
both the mobile node and ground station allows for an increase in communication range and a more reliable connection.

The calculation of target angles using GPS modules has also been assessed. With a strong GPS signal from at least four satellites and accounting for the WGS84 ellipsoidal earth model, the position of two objects relative to each other can be accurately calculated. The limiting factor in these calculations is an inconsistent altitude reading that needs to be addressed in future revisions.

The motor control aspects of tracking can be implemented using PID control loops for both axes. Following the Ziegler-Nichols tuning method provides good values for the P, I, and D gains used in the control loops. These values can be further tweaked to improve system dynamics. Using compass heading is effective for azimuth angle feedback, but the signal needs to be filtered in future implementations to provide more steady performance. The through-shaft potentiometer used for elevation feedback proved to be consistently accurate, but studies would need to be performed to assess its mechanical longevity as the antenna cycles up and down repeatedly. A motor controller with a higher power output threshold is also desirable so that both control loops may be executed simultaneously.
APPENDIX A

MOBILE NODE CODE
// Mobile Node that Broadcasts GPS Coordinates
#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>
#include <SPI.h>
#include <RH_RF95.h>

#define RFM95_CS 10
#define RFM95_RST 9
#define RFM95_INT 2
#define RF95_FREQ 915.0
#define GPSECHO false

RH_RF95 rf95(RFM95_CS, RFM95_INT);
SoftwareSerial mySerial(3, 4);
Adafruit_GPS GPS(&mySerial);

boolean usingInterrupt = false;
void useInterrupt(boolean);

byte * message = new byte[16];
float rssi;

void setup()
{
    // Configure Transceiver
    pinMode(RFM95_RST, OUTPUT);
digitalWrite(RFM95_RST, HIGH);

    while (!Serial);
    Serial.begin(9600);
delay(100);

digitalWrite(RFM95_RST, LOW);
delay(10);
digitalWrite(RFM95_RST, HIGH);
delay(10);

    while (!rf95.init())
    {
        Serial.println("LoRa radio initialization failed");
        while (1);
    }

    if (!rf95.setFrequency(RF95_FREQ))
    {
        Serial.println("Failed to set LoRa frequency");
        while (1);
    }
rf95.setTxPower(23, false);
GPS.begin(9600);
GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
GPS.sendCommand(PMTK_SET_NMEA_UPDATE_10HZ); // 10 Hz GPS update rate
useInterrupt(true);
delay(1000);

// Interrupt is called once a millisecond, looks for any new GPS data, and stores it
SIGNAL(TIMER0_COMPA_vect) {
    char c = GPS.read();
#ifdef UDR0
    if (GPSECHO)
        if (c) UDR0 = c;
#endif
}

void useInterrupt(boolean v) {
    if (v)
        {
            OCR0A = 0xAF;
            TIMSK0 |= _BV(OCIE0A);
            usingInterrupt = true;
        }
    else
        {
            TIMSK0 &= ~_BV(OCIE0A);
            usingInterrupt = false;
        }
}

uint32_t timer = millis();
int16_t packetnum = 0;

void loop()
{
    if (GPS.newNMEAreceived())
    {
        if (!GPS.parse(GPS.lastNMEA()))
            return;

        if (timer > millis()) timer = millis();


if (millis() - timer > 1000)
{
    timer = millis();
    if (GPS.fix)
    {
        Serial.print("\nLatitude: ");
        Serial.print(GPS.latitudeDegrees, 20);
        Serial.print("\nLongitude: ");
        Serial.print(GPS.longitudeDegrees, 20);
        Serial.print("\nAltitude: ");
        Serial.print(GPS.altitude, 20);
        Serial.print("\nSatellites: ");
        Serial.print("\nFix: ");
        Serial.println((int)GPS.fix);
        Serial.println((int)GPS.satellites);
    }
}
Serial.println("Sending to rf95_server");
Serial.println("Sending...");

// Prepare Data for Transmission
byte * latdata = (byte *) &GPS.latitudeDegrees;
byte * londata = (byte *) &GPS.longitudeDegrees;
byte * altdata = (byte *) &GPS.altitude;
rssi = rf95.lastRssi();
Serial.println(rssi);
byte * rssidata = (byte *) &rssi;
for (int i = 0; i < 4; i++)
{
    message[i] = latdata[i];
    message[i+4] = londata[i];
    message[i+8] = altdata[i];
    message[i+12] = rssidata[i];
}
rf95.send((uint8_t *)message, 16);
Serial.println("Waiting for packet to complete..."); delay(10); rf95.waitPacketSent();
// Wait for a reply
uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
uint8_t len = sizeof(buf);
Serial.println("Waiting for reply..."); delay(10);
if (rf95.available())
{
    if (rf95.recv(buf, &len))
    {

Serial.print("Got reply: ");
Serial.println((char*)buf);
Serial.print("RSSI: ");
Serial.println(rf95.lastRssi(), DEC);
} else {
    Serial.println("Receive failed");
}
else {
    Serial.println("No reply, is there a listener around?");
}
APPENDIX B

GROUND STATION CODE
// GPS Aided Antenna Control Station
#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>
#include <SPI.h>
#include <RH_RF95.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_HMC5883_U.h>

#define RFM95_CS 10
#define RFM95_RST 9
#define RFM95_INT 2
#define RF95_FREQ 915.0
#define GPSECHO false
#define LED 13
#define EARTH_A 6378.137 // Earth major axis in meters
#define EARTH_flnv 298.257223563 // Reciproval of flattening

RH_RF95 rf95(RFM95_CS, RFM95_INT);

SoftwareSerial mySerial(5, 4);
Adafruit_GPS GPS(&mySerial);

boolean usingInterrupt = false;
void useInterrupt(boolean);

Adafruit_HMC5883_Unified mag = Adafruit_HMC5883_Unified(12345);

uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
uint8_t len = sizeof(buf);
float x = *(float *)&buf;
uint8_t latbuf[4];
uint8_t lonbuf[4];
uint8_t altbuf[4];
uint8_t rssibuf[4];
float droneLatDegrees = *(float *)&latbuf;
float droneLonDegrees = *(float *)&lonbuf;
float droneAltitude = *(float *)&altbuf;
float droneRssi = *(float *)&rssibuf;
float myHeading;
float EARTH_f; // Flattening
float EARTH_B = EARTH_A * (1 - EARTH_f); // Earth minor axis in meters
float EARTH_eccSq = 1 - (EARTH_B*EARTH_B) / (EARTH_A*EARTH_A);
float dsq;
float d;
float RN;
float X; float Y; float Z;
float N; float E; float U;
float targetAz; float targetEl; float targetRange;
void LatLonToAzElR(float dronelat, float dronelon, float dronealt, float gslat, float gslon, float gsalt);
float TempEl;

int AzenablePin = 6;
int Azin1Pin = 7;
int Azin2Pin = 8;
int Azspeed = 0;
int KpAz = 3;
int KiAz = 0.9;
int KdAz = 1;
int Azerror;
int AzAccumulator = 0;
int Azerrorlast = 0;
boolean AzComplete;
int ElenablePin = 3;
int Elin3Pin = A1;
int Elin4Pin = A2;
int Elspeed = 0;
double KpEl = 3.6;
double KiEl = 0.2;
double KdEl = 3;
int Elerror;
int ElAccumulator = 0;
int Elerrorlast = 0;
boolean ElComplete;

void setup()
{
    // Configure Motor Control
    pinMode(Azin1Pin, OUTPUT);
    pinMode(Azin2Pin, OUTPUT);
    pinMode(AzenablePin, OUTPUT);
    pinMode(Elin3Pin, OUTPUT);
    pinMode(Elin4Pin, OUTPUT);
    pinMode(ElenablePin, OUTPUT);

    // Configure Transceiver
    pinMode(LED, OUTPUT);
    pinMode(RFM95_RST, OUTPUT);
digitalWrite(RFM95_RST, HIGH);

    while (!Serial);
Serial.begin(9600);
delay(100);

// Make sure motors are not initially active
analogWrite(ElenablePin, 0);
analogWrite(AzenablePin, 0);

digitalWrite(RFM95_RST, LOW);
delay(10);
digitalWrite(RFM95_RST, HIGH);
delay(10);

while (!rf95.init()) {
    Serial.println(F("LoRa radio initialization failed"));
    while (1);
}

if (!rf95.setFrequency(RF95_FREQ)) {
    Serial.println(F("Failed to set LoRa frequency"));
    while (1);
}

// TX Power range 5 to 23 dBm (default 13dBm):
rf95.setTxPower(23, false);
GPS.begin(9600);
GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); // 1 Hz GPS update rate
useInterrupt(true);

// Initialize Compass
if (!mag.begin()) {
    /* There was a problem detecting the HMC5883 ... check your connections */
    Serial.println(F("Compass initialization failed"));
    while (1);
}

// Interrupt is called once a millisecond, looks for any new GPS data, and stores it
SIGNAL(TIMER0_COMPA_vect) {
    char c = GPS.read();
#ifdef UDR0
    if (GPSECHO)
        if (c) UDR0 = c;
#endif
}
void useInterrupt(boolean v) {
    if (v)
    {
        OCR0A = 0xAF;
        TIMSK0 |= _BV(OCIE0A);
        usingInterrupt = true;
    }
    else
    {
        TIMSK0 &= ~_BV(OCIE0A);
        usingInterrupt = false;
    }
}

uint32_t timer = millis();

void loop()
{
    if (GPS.newNMEAreceived())
    {
        if (!GPS.parse(GPS.lastNMEA()))
            return;
    }

    if (timer > millis())  timer = millis();

    if (rf95.available())
    {
        // Print Recorded Data
        Serial.print(rf95.lastRssi(),DEC);
        Serial.print(", ");
        Serial.print(targetAz, 4);
        Serial.print(", ");
        Serial.print(myHeading, 4);
        Serial.print(", ");
        TempEl=(targetEl-694)*90/(967-694);
        Serial.print(TempEl,4);
        Serial.print(", ");
        TempEl=90*(analogRead(3)-694)/(967-694);
        Serial.print(TempEl, 4);
        Serial.print(", ");
        Serial.print(targetRange, 8);
        Serial.print(", ");
        Serial.print(GPS.latitudeDegrees, 8);
        Serial.print(", ");
        Serial.print(GPS.longitudeDegrees, 8);
    }
Serial.print("");
Serial.print(GPS.altitude, 1);
Serial.print(";");
Serial.print(droneLatDegrees, 8);
Serial.print(";");
Serial.print(droneLonDegrees, 8);
Serial.print(";");
Serial.print(droneAltitude, 1);
Serial.print(";");
Serial.print(droneRssi, 0);
Serial.print(";");
Serial.print(N, 2);
Serial.print(";");
Serial.print(E, 2);
Serial.print(";");
Serial.print(U, 2);
Serial.print(";");
Serial.print("\n\r");
if (rf95.recv(buf, &len))
{
  digitalWrite(LED, HIGH);

    // Send a reply
  uint8_t data[] = "Received";
  rf95.send(data, sizeof(data));
  rf95.waitPacketSent();

digitalWrite(LED, LOW);

    // Parse Message into Lat, Lon, Alt, Last RSSI
for (int i = 0; i <= 3; i++)
{
  latbuf[i] = buf[i];
  lonbuf[i] = buf[i + 4];
  altbuf[i] = buf[i + 8];
  rssibuf[i] = buf[i+12];
}
droneLatDegrees = *(float *)&latbuf;
droneLonDegrees = *(float *)&lonbuf;
droneAltitude = *(float *)&altbuf;
droneRssi = *(float *)&rssibuf;
}
else
{
  Serial.println(F("Receive failed"));
}
if(GPS.fix) // Wait for GPS signal
{
    // Function for calculating desired Azimuth and Elevation angles
    LatLonToAzElR(droneLatDegrees, droneLonDegrees, droneAltitude,
    GPS.latitudeDegrees, GPS.longitudeDegrees, GPS.altitude);

    // Get current compass heading
    getCompassHeading();
    Azerror = targetAz - myHeading;

    // Convert from 0-90 degree elevation value to match analogread scale
    targetEl = (targetEl/90)*(967-694) + 694; // 967 and 694 are the analogread values
    // corresponding to 0 and 90 degrees elevation respectively (approximated from
    // leveling tool)

    // Read Elevation feedback potentiometer
    Elerror = targetEl - analogRead(3);

    // Set Control Loop Completion Flags
    AzComplete=false;
    ElComplete=false;
    while((AzComplete)!=true)
    {
        setAzMotor();
    }
    while((ElComplete)!=true)
    {
        setElMotor();
    }
}

void displayCompassSensorDetails(void)
{
    sensor_t sensor;
    mag.getSensor(&sensor);
    Serial.println("------------------------------------");
    Serial.print("Sensor: "); Serial.println(sensor.name);
    Serial.print("Driver Ver: "); Serial.println(sensor.version);
    Serial.print("Unique ID: "); Serial.println(sensor.sensor_id);
    Serial.print("Max Value: "); Serial.print(sensor.max_value); Serial.println(" uT");
    Serial.print("Min Value: "); Serial.print(sensor.min_value); Serial.println(" uT");
    Serial.print("Resolution: "); Serial.println(sensor.resolution); Serial.println(" uT");
    Serial.println("------------------------------------");
    Serial.println("");
}
```c
void getCompassHeading(void) // (void)
{
    /* Get a new sensor event */
    sensors_event_t event;
    mag.getEvent(&event);

    /* Display the results (magnetic vector values are in micro-Tesla (uT)) */
    Serial.print("X: "); Serial.print(event.magnetic.x); Serial.print("  ");
    Serial.print("Y: "); Serial.print(event.magnetic.y); Serial.print("  ");
    Serial.print("Z: "); Serial.print(event.magnetic.z); Serial.print("  ");Serial.println("uT");

    // Hold the module so that Z is pointing 'up' and you can measure the heading with x&y
    // Calculate heading when the magnetometer is level, then correct for signs of axis.
    float heading = atan2(event.magnetic.y, event.magnetic.x);

    // Once you have your heading, you must then add your 'Declination Angle', which is
    // the 'Error' of the magnetic field in your location.
    // Find yours here: https://www.ngdc.noaa.gov/geomag-web/#declination
    // 3.59 degrees E or 0.06266 radians for 33.2543N 97.1525W
    float declinationAngle = 0.06266;
    heading += declinationAngle;

    // Correct for when signs are reversed.
    if (heading < 0)
        heading += 2 * PI;

    // Check for wrap due to addition of declination.
    if (heading > 2 * PI)
        heading -= 2 * PI;

    // Convert radians to degrees
    float headingDegrees = heading * 180 / M_PI;
    myHeading = headingDegrees;
}

void LatLonToAzElR(float dronelat, float dronelon, float dronealt, float gslat, float gslon, float gsalt)
{
    gsalt=(gsalt)/100; // Convert from cm to m
    dronealt=dronealt/100;
    dronelat = dronelat * M_PI / 180;
    dronelon = dronelon * M_PI / 180;
```
gslat = gslat * M_PI / 180;
gslon = gslon * M_PI / 180;

dsq = 1.0 - EARTH_eccSq * sin(dronelat) * sin(dronelat);
d = sqrt(dsq);
RN = EARTH_A / d;

// Get XYZ of ground station and subtract from that of drone
X = (RN + dronealt) * cos(dronelat) * cos(gslon);
Y = (RN + dronealt) * cos(dronelat) * sin(gslon);
Z = ((1 - EARTH_eccSq) * RN + dronealt) * sin(dronelat);

dsq = 1.0 - EARTH_eccSq * sin(gslat) * sin(gslat);
d = sqrt(dsq);
RN = EARTH_A / d;
X = X-(RN + gsalt) * cos(gslat) * cos(gslon));
Y = Y-(RN + gsalt) * cos(gslat) * sin(gslon));
Z = Z-((1 - EARTH_eccSq) * RN + gsalt) * sin(gslat));

// Convert to NED coordinates
N=-(-sin(gslat)*cos(gslon))*X+(-sin(gslat)*sin(gslon))*Y+(cos(gslat))*Z; //y
E=(-sin(gslon))*X+(cos(gslon))*Y; // x
U=-1*((-cos(gslat)*cos(gslon))*X+(-cos(gslat)*sin(gslon))*Y+(sin(gslat)); //z

// Calculate target angles
targetRange=sqrt(N*N+U*U+E*E);
targetAz=atan(fabs(E)/fabs(N));
targetAz=targetAz*180/M_PI;

// Conditional statements for Heading Calculation
if (N>=0 & E>=0)
{
   targetAz=targetAz;
}
else if (N<0 & E>=0)
{
   targetAz=180-targetAz;
}
else if (N<0 & E<0)
{
   targetAz=180+targetAz;
}
else if (N>=0 & E<0)
{
   targetAz=360-targetAz;
}
targetEl=asin(U/targetRange);
targetEl=targetEl*180/M_PI;

// Establish phyiscal angle limits
if (targetEl>90)
{  
  targetEl=90;
  }
if (targetEl<0){
  targetEl=0;
}
if (targetAz>360)
{
  targetAz=0;
}
if (targetAz<0)
{
  targetAz=0;
}
}

void setAzMotor()
{
  getCompassHeading();
  Azerror = targetAz - myHeading;
  AzAccumulator+=Azerror;

  // PID Calculation
  Azspeed=KpAz*abs(Azerror)+KiAz*AzAccumulator+KdAz*(Azerror-Azerrorlast);
  Azerrorlast=Azerror;

  // Establish minimum and maximum output parameters
  if (Azspeed<80){
    Azspeed=80;
  }
  if (Azspeed>255){
    Azspeed=255;
  }

  // Establish azimuth error threshold of 5 degrees
  if (Azerror>5)
  {
    analogWrite(AzenablePin, Azspeed);
    digitalWrite(Azin1Pin, true); // Increase heading angle
    delayMicroseconds(1);
    digitalWrite(Azin2Pin, false);
  }  
  else if (Azerror<-5)
  {
    analogWrite(AzenablePin, Azspeed);
    digitalWrite(Azin1Pin, false); // Decrease heading angle
  }
delayMicroseconds(1);
digitalWrite(Azin2Pin, true);
}
else
{
    AzAccumulator=0;
analogWrite(AzenablePin, 0); // Stop azimuth motor
    AzComplete=true; // Set loop completion flag
}
}

void setElMotor()
{
    Elerror = targetEl - analogRead(3);
    ElAccumulator+=Elerror;

    // PID Calculation
    Elspeed=KpEl*abs(Elerror)+KiEl*ElAccumulator+KdEl*(Elerror-Elerrorlast);
    Elerrorlast=Elerror;

    // Establish minimum and maximum output parameters
    if (Elspeed<55){
        Elspeed=55;
    }
    if (Elspeed>255){
        Elspeed=255;
    }

    // Establish elevation error threshold of approximately 3 degrees
    // 10*0.3=3
    if (Elerror>10)
    {
        analogWrite(ElenablePin, Elspeed);
        digitalWrite(Elin3Pin, false); // Increase elevation angle
        delayMicroseconds(1);
        digitalWrite(Elin4Pin, true);
    }
    else if (Elerror<-10)
    {
        analogWrite(ElenablePin, Elspeed);
        digitalWrite(Elin3Pin, true); // Decrease elevation angle
        delayMicroseconds(1);
        digitalWrite(Elin4Pin, false);
    }
    else
    {
        ElAccumulator=0;
    }
analogWrite(ElenablePin, 0); // Stop elevation motor
ElComplete=true; // Set loop completion flag
}
REFERENCES


