Antenna Test Range
First Semester Report
Fall Semester 2008

by
Kenneth Mitchell
Tejas Tank
Steven Turner

Prepared to partially fulfill the requirements for
ECE401

Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, Colorado 80523

Report Approved: ________________________________

Dr. Branislav Notaros

__________________________

Olivera Notaros
Abstract

The primary purpose of this interdisciplinary senior design project is to design and build an affordable and working antenna test range to be used as a classroom tool and in many different applications. We will integrate numerous advanced research tools and knowledge from electromagnetic theory, antennas, RF/microwaves, electronics, communications, computer hardware, control, power, mathematics, computational algorithms, and programming. In addition to research the work will be developmental, design, analysis, modeling, simulation, programming, building, budgeting, purchasing, installation, assessment, and testing at the system, device, component, and processor levels. Upon completion of this senior design project, it is hoped to have a fully functional antenna test range set up, and operating in the Antenna Test Lab room on the CSU campus. This system will consist of the various components described in this document.
# Table of Contents

Abstract .................................................................................................................................................. 2
Chapter 1: Introduction .......................................................................................................................... 5
Chapter 2: System Overview ................................................................................................................. 6
  2.1: General Overview ....................................................................................................................... 6
  2.2: System Research .......................................................................................................................... 7
Chapter 3: Hardware ............................................................................................................................. 8
  3.1: Positioning Hardware .................................................................................................................. 8
    Axes Configuration .......................................................................................................................... 9
    Cardboard Model .......................................................................................................................... 9
    Rotary Stages ............................................................................................................................... 11
  3.2: Controller Hardware .................................................................................................................... 12
    Breadboard Circuit ....................................................................................................................... 12
    Breadboard With Rotary Stage .................................................................................................... 13
    Printed Circuit Board Design ...................................................................................................... 13
Chapter 4: Stepper Controller Software .............................................................................................. 14
  4.1: Micro Controller ......................................................................................................................... 14
    Communications .......................................................................................................................... 15
    Timer Compare Match .................................................................................................................. 15
  4.2: Program flow chart ..................................................................................................................... 15
  4.3: Linear Speed Control Algorithm ............................................................................................... 17
Chapter 5: Computer Software ........................................................................................................... 25
  5.1 Stepper Controller Communication ............................................................................................ 26
  5.2 Network Analyzer Communications ........................................................................................... 26
  5.3 Data plotting .............................................................................................................................. 26
  5.4 Future Plans ................................................................................................................................ 29
Chapter 6: Conclusions and Future Work ......................................................................................... 29
  Positioner Mounting Hardware ....................................................................................................... 30
  Positioning Control Software ......................................................................................................... 30
  PC Application Development ......................................................................................................... 30
  System Test and Calibration ............................................................................................................. 31
Appendix A: Budget ............................................................................................................................. 31
Appendix B: Abbreviations .................................................................................................................. 31
References .............................................................................................................................................. 31
Acknowledgments .............................................................................................................................. 32
List of Figures

Illustration 1: Regions of Radiation.................................................................6
Illustration 2: Far Field Approximation..........................................................6
Illustration 3: $28,000 positioner.................................................................7
Illustration 4: $18,000 positioner.................................................................7
Illustration 5: 2-D configuration design.......................................................9
Illustration 6: Cardboard model.................................................................11
Illustration 7: 8" and 5" Rotary Stages.........................................................11
Illustration 8: Breadboard Circuit...............................................................12
Illustration 9: Breadboard with 5" motor..................................................13
Illustration 10: PCB Design...............................................................14
Illustration 11: Micro Controller Programmer...........................................15
Illustration 12: Stepper Controller Flow Chart...........................................16
Illustration 13: Stepper Plots.................................................................19
Illustration 14: Step Output.................................................................19
Illustration 15: Small Move Speed............................................................23
Illustration 16: PC Application.................................................................26
Illustration 17: Cartesian representation of Gain in dB scale........................27
Illustration 18: Radiation pattern with a linear scale.................................27
Illustration 19: Far field pattern...............................................................29

List of Tables

Table 1: Controller Test Output.................................................................17
Table 2: Budget.........................................................................................31
Chapter 1: Introduction

A great deal of research can be done using an antenna test range for there exists many applications. They include the research of new electromagnetic materials and their RF properties. New electromagnetic materials can be placed within an antenna test range and analysis can be done to determine characteristics about it including its radio frequency properties. Next using an antenna test range one can validate theoretical or computational models for an antenna to predict or obtain a better understanding as to their behavior and applications. From this once a new concept is designed it too can be tested and put under analysis to run simulations and experiment with. Other applications include electromagnetic scattering and small-scale models of such things like cars and aircraft. For further development on stealth technology radar profiling can be done on a small-scale aircraft before it is realized on a large scale level.

There are a number of various types of test ranges. These seem to be mostly classified as far field, near field, and compact ranges. The far field test ranges were the first developed, and were used to study the antennas at large distances which simulated real world situations. The near field test ranges were developed because of the large space required for far field test ranges. Techniques were developed that allowed antennas to be tested at distances of about 3 to 10 wavelengths. These results could then be transformed to the far field. The Compact Antenna Test Range (CATR) uses a reflector to create a field that approximates the plane wave of the far field. The Illustration one shows different regions of radiation. The fields from a source can be approximated by a multi-pole expansion. These terms can be written as a product of spherical Bessel functions that vary with distance r. For a large value of r the Bessel function will decay as 1/r which represents the far field. Other components must be factored in as the r value approaches the near field. The next term is called the induction term and varies as 1/r^2. For a very small value of r, the electrostatic term should be taken into account. This term varies as 1/r^3. By the principle of reciprocity the same pattern should be observed whether the antenna is transmitting or receiving. In the far field the relationships between the field patterns and current distributions usually involve trigonometric functions or Fourier transforms.

Illustration 1: Regions of Radiation
The far field distance is typically approximated to be greater than $\frac{2D^2}{\lambda}$, where $\lambda$ is the wavelength and $D$ is the antenna diameter. This ensures the phase variation is small enough to obtain satisfactory results. See IEEE-Std-149-1979 for suggested far field measurement standards.

**Chapter 2: System Overview**

**2.1: General Overview**

An antenna test range consists of two antennas sending and receiving signals in different positions while gathering and receiving data. One antenna is known as the antenna under test which will be a three axes system and is where all the primary controls will be done for our project. The other antenna is known as the transmitting or probe antenna which will be equipped with one polarization axis. A PC program will communicate back and forth from a stepper driver board commanding different stages connected to various axes on the positoner. As of right now a move command is sent by the user with a desired number of steps for a given stage to move. Once finished the stage stops and the stepper driver board displays the position of the stage. For further developing the antenna test range system a vector network analyzer must also be incorporated into the system. From here the PC program will communicate back and forth with the vector network analyzer sending and gathering measurements at various positions while the system is in operation. Some parameters that could be measured include input impedance, polarization, directivity, gain, radiation pattern, and beam width.

**2.2: System Research**

The primary goal of our project is to design and implement an affordable, working, multi-axis
system to be used for the antenna laboratory. In order to do this we began our research looking at complete commercial systems for sale. We quickly noticed that for just the positioner the cost was exceptionally high. One system we found similar to the configuration we are hoping to implement had a price tag of $28,000. Another one we considered was $18,000 but it was for small load antennas. Both of the antennas positioners we found were not only expensive but they did not include any controllers or other accessories that could be used to the overall system.

We decided it was best if we attempt to build our own positioner, because we would be able to keep costs down and be able to add or take away as many axes as we needed. We began looking at rotary tables/stages that would be used as an axis. Many different models were found with various specifications. We calculated that we were going to have approximately a 20lb antenna that would be connected three feet above the azimuth rotary stage. We ran calculations to figure out what the moments on the rotary stages would be and the approximate loads on the stages. Since only one of our three stages placed on the antenna under test positioner will have a load mounted directly on top of it, radial load calculations had to be completed for the other two stages. This led us to requiring a high torque and motor moment capabilities within the stage. After much deliberation and collaboration we agreed on two different stages to be used to construct both positioners for the antenna under test and the transmitting antenna. A 5 inch rotary stage would be used for both polarization axes on both antennas and for the azimuth stage on the antenna under test has a load limit of 275lbs and a moment of 40ft-lbs. The other stage is an 8 inch stage that will be used for the elevation axis which has a max load limit of 700lbs and a moment of 200ft-lbs. The reason for the increase in size for this stage is that all of the load of both the positioner and antenna will be connected in a radial configuration to this stage and therefore needed to be considerably larger than the others.

**Chapter 3: Hardware**

Hardware design for our senior design we are trying to implement the whole system design
that includes transmitting antenna design and receiving antenna design (under test). The system includes two types of hardware design, positioning and controller. Both hardware types are similar in terms of interaction between each other, but use different components to function as a whole.

3.1: Positioning Hardware

For the positioning hardware we are using stepper motors, the way stepper motor works is we apply certain voltage in sequence to move a step. We are using four stepper motors of sizes of 5 inch diameter (three motors) and 8 inch diameter (one motor). Using four stepper motors in the system implementation will give us four axes for the system. The four axes include Polarization (two), Azimuth (one), and Elevation (one). For the transmitting antenna, we will use one of the polarization axes that include one 5 inch diameter stepper motor. For the receiving antenna, we are going to combine the remaining three axes (polarization, azimuth and elevation) in such configuration that looks like Illustration 6: 2-D configuration design. Stepper motor will be used in the places of all the blue boxes in the image.

Axes Configuration

![Illustration 5: 2-D configuration design](image)
Cardboard Model

Illustration 6: Cardboard model

Rotary Stages

The main reason to use stepper motor for our senior design is the accuracy in movement of each stage. Illustration 8 shows the image of 8 inch and 5 inch rotary stages.

Illustration 7: 8" and 5" Rotary Stages
• The 5 inch motors have a gearing ratio of 72:1 that gives us 14,400 Steps / Revolution.
• The 8 inch motor has a gearing ratio of 80:1 that gives us 16,000 Steps / Revolution.

Having so many steps per revolutions, we can be very accurate with our stage movement. Next step in this part of project will be to carry out the hardware design with actual mounting.

### 3.2: Controller Hardware

For the controller hardware in this senior design, the main goal was to design a Printed Circuit Board (PCB) that controls the positioning of each axis. We started off with building our own breadboard circuit design as shown on Illustration 9.

**Breadboard Circuit**

![Breadboard Circuit](image)

Illustration 8 shows an AVR Atmega168 micro controller along with a breadboard power supply, and a serial port programmer.

This system is running at 20MHz using a ceramic resonator and was programmed in C using the free WinAVR package that includes the GCC compiler for use with the AVR microprocessors. This system operating at 20MHz requires 5 volts, but if the clock is slowed...
down to below 10MHz it can be operated at down to 2.7 volts.

This system has built in timers, and counters that can be used for reading encoders. Also included are PWM modules than can be used to drive the DC motors at variable speeds through use of a circuit such as an H-bridge. We could use these features to create a board that would control the position of one axis of the positioning system that would be interfaced to a computer through the serial port.

Also shown is a FT232 breakout board that is used to convert TTL serial to USB. This maps to a virtual serial port on a computer. This would make it possible to communicate with the system through the USB port.

**Breadboard With Rotary Stage**

Illustration 10 shows how our breadboard circuit connects and operates with one of the 5 inch stepper motor.

![Illustration 9: Breadboard with 5" motor](image)

**Printed Circuit Board Design**

We took our breadboard circuit design and used it with CadSoft Eagle (freeware software) to implement our PCB design. Actual size of the PCB came out to be 4.3 inch x 3.9 inch. Illustration 11 shows the PCB design with some of the components highlighted. We are using four stepper IC chips so each axis (motor) can connect with one stepper IC. Our breadboard circuit worked perfectly so the PCB design for it was ordered on first week of December and we are expecting it to be here within couple of weeks.
Chapter 4: Stepper Controller Software

The code running on the micro controller in our stepper board handles several features. These include those listed below.

- Moving a rotation stage to a specific location
- Controlling the speed, acceleration, and deceleration of the motor
- Keeping track of the position of each stage
- Accepting commands from a computer through the USB connection
- Reporting the stage positions back to the computer

4.1: Micro Controller

At the heart of our stepper controller board is an Atmel ATMEGA168 microcontroller running at 20 MHz. This controller was chosen for the reasons listed below.

- Free C compiler and other development tools
• Team member with personal experience
• High performance
• Easy to use

The program was written in C using tools provided by the WinAVR package. The Eclipse IDE was used to aid in the software development. Once the program was compiled it is downloaded using the AVRISP mkII In-System Programmer. This programmer plugs into a PC USB port, and a 6 pin header on the embedded system.

Communications

Communications to the computer were handled using the UART built into the micro controller configured to run at 19,200 baud. The TTL serial signals are converted to USB using the FT232RL IC chip.

Timer Compare Match

Our algorithm uses the timer module in the micro controller to have precise control of the speed, acceleration, and deceleration of the motor. Our main clock runs at 20 MHz, but we have configured the timer to run at the clock rate divided by 64 which is 312.5 KHz.

A 16 bit output compare register is loaded with a number. This number is then compared to the value stored in the timer register. When these numbers match, the timer compare interrupt is triggered. The movement of the motor is handled entirely by this interrupt service routine once the initial calculations are performed.

The output compare register is loaded with the next value required for the movement, and the timer register is reset. If the motor was accelerating then this value would be smaller than the current value for a faster interrupt time.

4.2: Program flow chart

A block diagram of the program flow is shown below.
When the micro controller enters the reset state the first thing to happen is setting up the hardware. This involves the following tasks.

- Initializing the UART for 19,200 baud, 8 data bits, 1 stop bit, and no parity
- Set the home switch pins to inputs with internal pull ups
- The motor direction and step pins as outputs

After the hardware is set up the program waits until a line of text is received. This text is placed in a circular buffer until a new line character is received. When this happens the line of text is parsed and the received data buffer is cleared. If the command is not valid then a help menu is displayed and the program waits for another command.

If a valid command is received then the appropriate action is performed. One type of command will set the acceleration, deceleration, or speed of the movement. A move can also be commanded. Using this command the number of steps to move is specified. If the number is positive the stage will move in one direction, and a negative number will specify a movement in the opposite direction.
If a move command is received then most of the required variables are calculated before the move is started. The timer module and output compare interrupt are then enabled. The main loop of the code then waits until the move complete flag is set by the interrupt service routine. The rest of the movement is then handled by the output compare match interrupt routine. The algorithm will be discussed in a later section.

Shown below is an example run of the program. The program was tested using a terminal emulation program.

<table>
<thead>
<tr>
<th>Controller test through a terminal program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colorado State University</td>
</tr>
<tr>
<td>Antenna Positioner Controller</td>
</tr>
<tr>
<td>?            - Show help</td>
</tr>
<tr>
<td>a [data]    - Set acceleration (range: 71 - 32000)</td>
</tr>
<tr>
<td>d [data]    - Set deceleration (range: 71 - 32000)</td>
</tr>
<tr>
<td>s [data]    - Set speed (range: 12 - motor limit)</td>
</tr>
<tr>
<td>m [data]    - Move [data] steps (range: -64000 - 64000)</td>
</tr>
</tbody>
</table>

acc/dec data given in 0.01*rad/sec^2 (100 = 1 rad/sec^2)
speed data given in 0.01*rad/sec (100 = 1 rad/sec)

Motor pos: 0 a:100 d:100 s:800 m:1000
> m 1000
Done

Motor pos: 1000 a:100 d:100 s:800 m:1000
> m -400
Done

Motor pos: 600 a:100 d:100 s:800 m:-400

Table 1: Controller Test Output

In the program run above the motor was run with an acceleration and deceleration of 1 radian per second squared and a speed of 10 radians per second. After each move the position has been updated.

The program currently does not use the home switch information to zero the axis. This functionality will be added next semester.

**4.3: Linear Speed Control Algorithm**

The program uses linear acceleration and deceleration. The parameters that can be set are the acceleration in radians per second squared, deceleration in radians per second squared,
and speed in radians per second. The move command specifies a specific number of steps to move in a direction. The position in steps is stored in a program variable and displayed after each move is completed.

The plots below were made in Matlab and show the operation of our stepper driver.

![Illustration 13: Stepper Plots](image)

The top plot shows the motor position change with time. The middle plot shows the motor speed. The lower graph shown acceleration which can be seen to be constant.

The stepper IC used has a step input pin. The motor advances one step on each rising edge of this pin. The frequency of these rising edges determines the speed of the motor. The below image of the oscilloscope shows the step output wave form of the micro controller.
The parameters shown below are discussed in the following section.

- $SPR =$ Motor steps per revolution
- $\alpha =$ angle of motor per step (rad)
- $t_i =$ timer period (sec)
- $\omega =$ speed (rad/sec)
- $\omega_{max} =$ maximum speed (rad/sec)
- $\omega' =$ acceleration or deceleration (rad/sec$^2$)
- $\omega_{accel} ' =$ acceleration (rad/sec$^2$)
- $\omega_{decel} ' =$ deceleration (rad/sec$^2$)
- $c_{min} =$ minimum counter delay for max speed
- $c_0 =$ counter delay for first step
- $c_n =$ counter delay for $n^{th}$ step
- $n =$ the current step
- $n_{total} =$ total number of steps to move
- $n_{accel\ limit} =$ number of steps before unbounded acceleration must stop
- $n_{accel} =$ number of steps for acceleration
- $n_{decel} =$ number of steps for deceleration
- $n_{decel\ start} =$ number of steps before start of deceleration

$\alpha$ can be calculated as shown below.

$$\alpha = \frac{2 \pi}{SPR}$$

We are using 200 step per revolution motors.
\[ \alpha = \frac{\pi}{100} \]

The time delay between stepper pulses control the speed of the motor by using the timer module. This means that the next time delay value must be calculated and loaded into the output compare register after each interrupt.

This time delay depends on the frequency of the counter, and the value stored into the output compare register. The frequency of the counter can be configured, but is not changed during the program. The counter module in our design is set at a rate of the main clock divided by 64. This is 312.5 Khz.

The speed of the motor is given by the equation below where \( \omega' \) is the acceleration.

\[ \omega(t) = \int_0^t \omega' \, d\tau = \omega' t \quad (1) \]

The motor position \( \theta(t) \) is given below which is also equal to \( n\alpha \).

\[ \theta(t) = \int_0^t \omega \, d\tau \]
\[ \theta(t) = \int_0^t \omega' \tau \, d\tau \]
\[ \theta(t) = \frac{1}{2} \omega' t^2 = n\alpha \quad (2) \]

Solving for \( t \) gives the time at which the step should occur for the specified acceleration.

\[ \frac{1}{2} \omega' t^2 = n\alpha \]
\[ t^2 = \frac{2n\alpha}{\omega'} \]
\[ t = \sqrt{\frac{2n\alpha}{\omega'}} \quad (3) \]

This equation is the time to the \( n^{th} \) step pulse. Now we need to solve for the time between two pulses.
\[ t_{n+1} - t_n = \sqrt{\frac{2(n+1)\alpha}{\omega'}} - \sqrt{\frac{2n\alpha}{\omega'}} \]
\[ t_{n+1} - t_n = \sqrt{\frac{2\alpha}{\omega'}} \left( \sqrt{n+1} - \sqrt{n} \right) = c_n t \] (4)

Where \( c_n \) is the timer count and \( t \) is the timer period.

Solving this equation for the timer count leads to the equation below. The first count delay is factored out in the lower equations.

\[ c_n = \frac{1}{t} \sqrt{\frac{2\alpha}{\omega'}} \left( \sqrt{n+1} - \sqrt{n} \right) \]
\[ c_0 = \frac{1}{t} \sqrt{\frac{2\alpha}{\omega'}} \]
\[ c_n = c_0 \left( \sqrt{n+1} - \sqrt{n} \right) \] (5)

This would require the calculation of two square roots for each timer compare interrupt. A faster algorithm is desired, and we wish to express the current time delay in relation to the previous delay.

\[ \frac{c_n}{c_{n-1}} = \frac{c_0 \left( \sqrt{n+1} - \sqrt{n} \right)}{c_0 \left( \sqrt{n} - \sqrt{n-1} \right)} \]
\[ \frac{c_n}{c_{n-1}} = \frac{\sqrt{n+1} - \sqrt{n}}{\sqrt{n} - \sqrt{n-1}} \]
\[ c_n = \sqrt{\frac{1 + \frac{1}{n} - 1}{1 - \sqrt{1 - \frac{1}{n}}} \left( \sqrt{n+1} - \sqrt{n} \right)} \] (6)

Using a Taylor series approximation lead to the following.
\[
\sqrt{1 + \frac{1}{n}} = 1 + \frac{1}{2n} - \frac{1}{8n^2} + O\left(\frac{1}{n^2}\right)
\]
\[
c_n = 4n - 1
\]
\[
c_{n-1} = 4n + 1
\]
\[
c_n = c_{n-1} - \frac{2c_{n-1}}{4n + 1}
\]

This approximation has an error of 44% at n=1, but very good accuracy for higher step values. This is corrected by multiplying the first step delay \(c_0\) by 0.676. The equation used to calculate the first step delay is then shown below.

\[
c_0 = \frac{0.676}{t_c} \sqrt{\frac{2\alpha}{\omega'}}
\]

We now know the step delay required between steps for the specified acceleration, timer period, and steps per revolution of the stepper motor. Now by using equations (1) and (2) we can derive an equation that will show us the number of steps required to reach a specified speed.

\[
n_{\text{accel}} = \frac{\omega^2}{2 \omega' \alpha}
\]

We now have to figure out where we need to start decelerating the motor to stop at the correct number of steps. First we determine the acceleration limit for the requested number of steps to move.

\[
n_{\text{accel limit}} = \frac{n_{\text{total}} \omega_{\text{decel}}'}{\omega_{\text{accel}}' + \omega_{\text{decel}}'}
\]

When the motor is moving enough steps to reach the maximum speed then \(n_a < n_{\text{accel limit}}\) the step count to start deceleration is found by the following equation.

\[
n_{\text{decel}} = n_{\text{accel}} \frac{\omega_{\text{accel}}}{\omega_{\text{decel}}}
\]

The plot below shows the plot for a small move command. The motor does not have enough steps to accelerate to the desired speed before deceleration is required.
The following listing shows a summary of all the equations used in the stepper controller program.

• The minimum counter compare value for the maximum speed is shown below.

\[ c_{\text{min}} = \frac{\alpha}{t_t \omega_{\text{max}}} \]  

(12)

• The initial counter value for the first step is shown below.

\[ c_0 = \frac{0.676}{t_t} \sqrt{\frac{2\alpha}{\omega'}} \]  

(13)

• To calculate the nth counter value for the timer compare match interrupt the equation below is used. This same equation is used for both acceleration and deceleration.

\[ c_n = c_{n-1} - \frac{2c_{n-1}}{4n+1} \]  

(14)

• This will calculate the number of steps required before the desired speed is reached

\[ n_{\text{accel}} = \frac{\omega^2}{2 \omega' \alpha} \]  

(15)
• Calculate the step value for the maximum speed that can be achieved.

\[ n_{\text{accel limit}} = \frac{n_{\text{total}} \omega_{\text{decel}}'}{\omega_{\text{accel}}' + \omega_{\text{decel}}'} \] (16)

• If this is less than the number of steps required to reach the specified speed then the motor will not reach the maximum speed. The deceleration step count is calculated as below.

\[ n_{\text{decel}} = n_{\text{accel limit}} - n_{\text{total}} \] (17)

• If this is larger then we will have a constant speed portion of the movement. The step value to start decelerating is shown below.

\[ n_{\text{decel}} = -n_{\text{accel}} \frac{\omega_{\text{accel}}'}{\omega_{\text{decel}}'} \] (18)

• The deceleration start value is then calculated.

\[ n_{\text{decel start}} = n_{\text{total}} + n_{\text{decel}} \] (19)

**Chapter 5: Computer Software**

The second part to our software is the application running on a desktop computer. This will be the major focus for our next semester plans. This application is responsible for several tasks.

- Send commands to the stepper controller board to move the positioners
- Receive the position data from the controller board
- Send commands to the vector network analyzer
- Receive measurements from the network analyzer
- Display the data in a meaningful manner
- Provide data export options

A screen shot of our first attempt at the PC application is shown below. This application was written in Java using the NetBeans IDE. This application was a very basic shell that did not have much functionality built in. It was able to send and receive data through the serial port. There were also working controls that could be set to show the antenna position.
5.1 Stepper Controller Communication

The PC application communicates with the stepper driver board through the USB port. The USB driver used maps to a virtual serial port. ASCII commands are then sent to and received from the micro controller running on the stepper driver board.

5.2 Network Analyzer Communications

Communications with the network analyzer has not been studied much at this point. We are planning on hooking the network analyzer to the network using an Ethernet connection. We would then use the system IO libraries provided by Agilent for use with the .Net framework.

5.3 Data plotting

Once the data is acquired, it should be presented in a meaningful way. There are a number of possibilities for this. Some of these include polar or Cartesian plots for the radiation pattern, gain, or the beam width which is the angle at the -3dB points around the maximum gain. Logarithmic or standard scales should also be considered. Some examples of these types of graphs are shown below.
Illustration 17: Cartesian representation of Gain in dB scale

Illustration 18: Radiation pattern with a linear scale
We used the JfreeChart Java library in our application pictured above. Built in functionality included saving the chart as an image, zooming, changing the labels, and graph colors. These types of features would be great to have in our final application.

5.4 Future Plans

Since this application was written we have decided to change directions. It was discovered that Agilent has a free IO library for the .Net languages written to communicate with their instruments. They also provide code examples.

After some investigation it was also discovered that programming using Microsoft Visual Studio makes developing a graphical user interface very easy.

Because of these reasons it was decided to switch to the C# programming language and to use Visual Studio. We have not yet started developing this part of the project since we have been focused on other parts of the project this semester.

Chapter 6: Conclusions and Future Work

We have completed a lot of research and work on the positioners, and the positioner controller this semester. There is still much work left to do on our project. A listing of our next semester goals is given below.
Build the positioner mounting hardware
Completion of positioning control software for four-axis rotational RF scanning system
Communication with vector network analyzer
RF data acquisition and processing software
RF cabling throughout the system
Power cabling and power supplies
Calibration of source antennas

**Positioner Mounting Hardware**

One goal for next semester is to find a way of getting our positioner mounting hardware manufactured by someone. This will give us our complete polarization over azimuth over elevation positioner.

Our stepper controller board is currently being manufactured. Time will be dedicated to assembly and testing of the board.

**Positioning Control Software**

We have a working micro controller program running on our stepper controller board, but it will require some modifications.

The home switches are not currently read, and a home function will need to be created to zero the position of the stage.

Also on the agenda is a function to specify movements, and report position in a more coherent format.

We also need to modify the program to be able to control multiple axes. It is currently set up to only handle one stage.

**PC Application Development**

This will be the major focus for our next semester plans. This application is responsible for several tasks.

Send commands to the stepper controller board to move the positioners
Receive the position data from the controller board
Send commands to the vector network analyzer
Receive measurements from the network analyzer
Display the data in a meaningful manner
Provide data export options

We have not done much work in this area of our project yet. The work that we have done will not be entirely applicable since we have changed programming languages.
The first task we take on will probably be developing the part of the application responsible for sending and receiving commands from the stepper controller board. After this task is completed we will move on to communicating with the vector network analyzer.

We will have a functional system at that point, and then the focus will be on improving functionality, and usability.

We will add a better graphical user interface, and data plotting capabilities in order to better visualize the data. We would also like to add data export capabilities.

**System Test and Calibration**

Once we have a functional system we will need to conduct verification and testing. It will be necessary to design tests to verify the accuracy and repeatability of our system.

User documentation will also need to be created on the system. It is important for users to be able to use the system safely and to know the limitations and proper operations.

**Appendix A: Budget**

<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>8&quot; Rotational stage</td>
<td>$4,475.00</td>
<td>1</td>
</tr>
<tr>
<td>5&quot; Rotational stage</td>
<td>$2,050.00</td>
<td>3</td>
</tr>
<tr>
<td>Motor and home switch cables</td>
<td>$45.00</td>
<td>4</td>
</tr>
<tr>
<td>Stepper controller PCB</td>
<td>$59.96</td>
<td>1</td>
</tr>
<tr>
<td>Circuit comp</td>
<td>$50.00</td>
<td>Many</td>
</tr>
<tr>
<td>Vector network analyzer</td>
<td>Owned by antenna lab</td>
<td>1</td>
</tr>
</tbody>
</table>

*Table 2: Budget*

**Appendix B: Abbreviations**

ASCII - American Standard Code for Information Interchange  
PCB – Printed Circuit Board  
IDE – Integrated Development Environment  
IO – Input / Output  
USB – Universal Serial Bus

**References**


**Acknowledgments**

We would like to thank the following people for their assistance through the semester.

- Dr. Branislav Notaros
- Oliver Notaros
- Eve Klopf
- Carlos Flores