## 3. Trilateration

### 3.1 Introduction to Trilateration

To locate an object that is in the field a system was created that uses radio frequency signals. The strength of the radio frequency signal is measured between the tagged object and the readers that are stationed in the field. Once the signal strength is gathered, it can be converted into a distance using the distance formula which will be explained in further detail later in this report. When the distances have been calculated they are plugged into a system on quadratic equations called trilateration. Using trilateration makes it is possible to find the tagged object on the XY plane and it will also allow us to find the z axis of the object as well. There are two types of trilateration, the first one is 2-D trilateration, which only allows you to find a tagged object on the XY plane and then there is 3D trilateration. 3D trilateration allows you to find a tagged object on the X, Y, and Z coordinate system.

### 3.2 Trilateration in 2D

For the indoor location system 2D trilateration is used to find a tagged object that is located on a surface, which will be on an XY plane. The location of three readers has to be known along with the distances between the readers and the unknown tagged object for 2D trilateration to work correctly. One can visualize this by looking at figure (x1), where the red dot in the center has an unknown location. The red dot will represent the location of the object that is being searched for. The reference nodes, also known as the readers, are labeled $A_1$, $A_2$, and $A_3$, the distances between the reference nodes and the tagged node are labeled $d_1$, $d_2$ and $d_3$. The intersection between all three nodes is the location of our unknown tag.

Each tag and reader will consist of a transceiver and an antenna. The tag transmits a signal and the signal strength, also known as RSSI (Received Signal Strength Indication) is measured between the tag and the stationed readers. The signal strength can be converted into distance, which gives us the three known distances that are needed for trilateration. The downside for using signal strength is that the calculated distance will not always have the exact distance between the reader and the tagged object. The reason for this is because the signal strength coming from the tag could be interfered with other signal frequencies, room temperature, humidity, construction within a building and metal interference. The equation converting signal strength to distance can be tuned for some of the parameters listed below, but there are some things that cannot be controlled, therefore with every distance that is calculated there will be a small error.

Once the distance is calculated, trilateration will be used to find the position of the tagged object. The way 2D trilateration works is shown below. In the following equations Xi and Yi represent the position of Ai (readers), where $i = 1,2,3$.

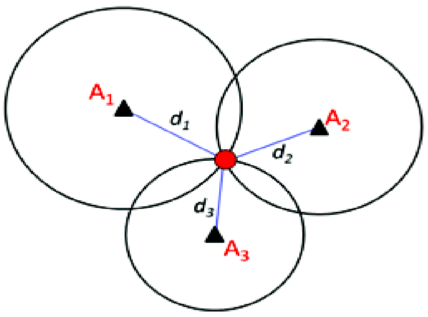


Figure 8: Trilateration in 2D

$$(x - x_1)^2 + (y - y_1)^2 = d_1^2 \qquad (Equation\ 1)$$

$$(x - x_2)^2 + (y - y_2)^2 = d_2^2 \qquad (Equation\ 2)$$

$$(x - x_3)^2 + (y - y_3)^2 = d_3^2 \qquad (Equation\ 3)$$

To simplify this system of quadratic equations, equation 3 will be substituted into equations 1 and 2, which will leave two linear equations.

$$2(x_2 - x_1)x + 2(y_2 - y_1)y = (d_1^2 - d_2^2) - (x_1^2 - x_2^2) - (y_1^2 - y_2^2) \qquad (Equation\ 4)$$

$$2(x_3 - x_1)x + 2(y_3 - y_1)y = (d_1^2 - d_3^2) - (x_1^2 - x_3^2) - (y_1^2 - y_3^2) \qquad (Equation\ 5)$$

The X and Y coordinates are found by solving equation 4 and equation 5 using Cramer's rule.

$$X = \frac{\begin{vmatrix} (d_1^2 - d_2^2) - (x_1^2 - x_2^2) - (y_1^2 - y_2^2) & 2(y_2 - y_1) \\ (d_1^2 - d_3^2) - (x_1^2 - x_3^2) - (y_1^2 - y_3^2) & 2(y_3 - y_1) \end{vmatrix}}{\begin{vmatrix} 2(x_2 - x_1) & 2(y_2 - y_1) \\ 2(x_3 - x_1) & 2(y_3 - y_1) \end{vmatrix}} \qquad (Equation\ 6)$$

$$Y = \frac{\begin{vmatrix} 2(x_2 - x_1) & (d_1^2 - d_2^2) - (x_1^2 - x_2^2) - (y_1^2 - y_2^2) \\ 2(x_3 - x_1) & (d_1^2 - d_3^2) - (x_1^2 - x_3^2) - (y_1^2 - y_3^2) \end{vmatrix}}{\begin{vmatrix} 2(x_2 - x_1) & 2(y_2 - y_1) \\ 2(x_3 - x_1) & 2(y_3 - y_1) \end{vmatrix}} \qquad (Equation\ 7)$$

Equations 6 and 7 will be solved using Matlab. The simulation will be shown in the section called Trilateration Simulation Results (Section 3.7).

## 3.3 Trilateration in 3D

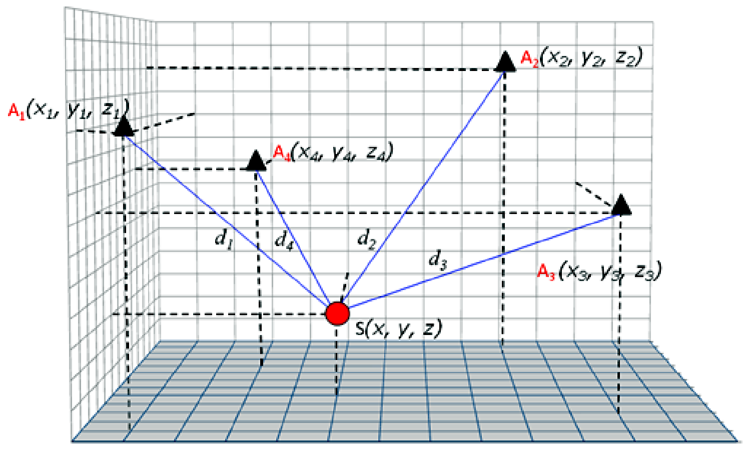


Figure 9: Trilateration in 3D

Originally, the indoor positioning system was supposed to find an object within a warehouse. With this in mind, it was taken into consideration that there are items elevated in high placed that can't be seen or reached. To do this, a 3D system is needed that will have a Z component along with the X and Y components. As you can see in figure 9, there is a fourth reader. This fourth reader gives us an extra component; therefore linear algebra can be used to find the height of the tagged object. The 3D trilateration quadratic equations are similar to the 2D trilateration quadratic equations. Also, because of the quadratic equations every reader can be at a different height. The equations for 3D trilateration are as follows:

$$(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2 = d_1^2 \qquad (Equation\ 8)$$

$$(x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2 = d_2^2 \qquad (Equation\ 9)$$

$$(x-x_3)^2 + (y-y_3)^2 + (z-z_3)^2 = d_3^2 \qquad (Equation\ 10)$$

$$(x-x_4)^2 + (y-y_4)^2 + (z-z_4)^2 = d_4^2 \qquad (Equation\ 11)$$

The above equations can be simplified into 3 linear equations.

$$2(x_2-x_1)x + 2(y_2-y_1)y + 2(z_2-z_1)z = (d_1^2-d_2^2) - (x_1^2-x_2^2) - (y_1^2-y_2^2) - (z_1^2-z_2^2) \qquad (Eq\ 12)$$

$$2(x_3-x_1)x + 2(y_3-y_1)y + 2(z_3-z_1)z = (d_1^2-d_3^2) - (x_1^2-x_3^2) - (y_1^2-y_3^2) - (z_1^2-z_3^2) \qquad (Eq\ 13)$$

$$2(x_4-x_1)x + 2(y_4-y_1)y + 2(z_4-z_1)z = (d_1^2-d_4^2) - (x_1^2-x_4^2) - (y_1^2-y_4^2) - (z_1^2-z_4^2) \qquad (Eq\ 14)$$

Now, the X,Y and Z components are found by solving equations 11,12, and 13 using Cramer's rule.

$$X = \frac{\begin{matrix} (d_1^2 - d_2^2) - (x_1^2 - x_2^2) - (y_1^2 - y_2^2) - (z_1^2 - z_2^2) & 2(y_2 - y_1) & 2(z_2 - z_1) \\ (d_1^2 - d_3^2) - (x_1^2 - x_3^2) - (y_1^2 - y_3^2) - (z_1^2 - z_3^2) & 2(y_3 - y_1) & 2(z_3 - z_1) \\ (d_1^2 - d_4^2) - (x_1^2 - x_4^2) - (y_1^2 - y_4^2) - (z_1^2 - z_4^2) & 2(y_4 - y_1) & 2(z_4 - z_1) \end{matrix}}{\begin{matrix} 2(x_2 - x_1) & 2(y_2 - y_1) & 2(z_2 - z_1) \\ 2(x_3 - x_1) & 2(y_3 - y_1) & 2(z_3 - z_1) \\ 2(x_4 - x_1) & 2(y_4 - y_1) & 2(z_4 - z_1) \end{matrix}} \qquad (Equation\ 15)$$

$$Y = \frac{\begin{matrix} 2(x_2 - x_1) & (d_1^2 - d_2^2) - (x_1^2 - x_2^2) - (y_1^2 - y_2^2) - (z_1^2 - z_2^2) & 2(z_2 - z_1) \\ 2(x_3 - x_1) & (d_1^2 - d_3^2) - (x_1^2 - x_3^2) - (y_1^2 - y_3^2) - (z_1^2 - z_3^2) & 2(z_3 - z_1) \\ 2(x_4 - x_1) & (d_1^2 - d_4^2) - (x_1^2 - x_4^2) - (y_1^2 - y_4^2) - (z_1^2 - z_4^2) & 2(z_4 - z_1) \end{matrix}}{\begin{matrix} 2(x_2 - x_1) & 2(y_2 - y_1) & 2(z_2 - z_1) \\ 2(x_3 - x_1) & 2(y_3 - y_1) & 2(z_3 - z_1) \\ 2(x_4 - x_1) & 2(y_4 - y_1) & 2(z_4 - z_1) \end{matrix}} \qquad (Equation\ 16)$$

$$Z = \frac{\begin{matrix} 2(x_2 - x_1) & 2(y_2 - y_1) & (d_1^2 - d_2^2) - (x_1^2 - x_2^2) - (y_1^2 - y_2^2) - (z_1^2 - z_2^2) \\ 2(x_3 - x_1) & 2(y_3 - y_1) & (d_1^2 - d_3^2) - (x_1^2 - x_3^2) - (y_1^2 - y_3^2) - (z_1^2 - z_3^2) \\ 2(x_4 - x_1) & 2(y_4 - y_1) & (d_1^2 - d_4^2) - (x_1^2 - x_4^2) - (y_1^2 - y_4^2) - (z_1^2 - z_4^2) \end{matrix}}{\begin{matrix} 2(x_2 - x_1) & 2(y_2 - y_1) & 2(z_2 - z_1) \\ 2(x_3 - x_1) & 2(y_3 - y_1) & 2(z_3 - z_1) \\ 2(x_4 - x_1) & 2(y_4 - y_1) & 2(z_4 - z_1) \end{matrix}} \qquad (Equation\ 17)$$

Both 2D and 3D trilateration have its pros and cons. Adding an extra reader to the system will increase the accuracy of the location of the tagged object in 2D. However in 3D trilataration, the fourth reader is needed for the extra unknown z-component. 3D trilateration will also be less accurate and more complex than 2D trilateration. The more complex the system is; the longer the computational time will be, which is another advantage that 2D trilateration has over 3D trilateration. But there is a way to execute a 3D system by expanding 2D trilateration and this system is called COLA, which will be explained in further detail in a section called COLA (Complexity Reduced 3D Trilateration Localization Approach) (Section 3.5).

## 3.4 Matlab code for Trilateration in 2D and 3D

The 2D trilateration Matlab code has to have two different Matlab files for it to work correctly. One file is the actual trilateration code, which is called two_tri.m and is located in Appendix B, the other Matlab file stores the distances between the node we are searching for and the known readers locations. The file that stores the reader coordinates and distances will be called Test2D.m and is located in Appendix B. The Test2D file has to have the known locations of the three readers and the node with the unknown location. The location of the reader will be manually put into the host computer by the user when the system is set up. However, the three distances will be fed to Matlab by the Arduino connected to the host computer. The Arduino is converting the RSSI signal into distance and sending it to Matlab through the serial port. Both Matlab files share information with each other, so, therefore once all the X and Y components of the readers as well as the distances are known, the X and Y component of the unknown node can be found using the two_tri.m file. When the program is done executing a screen will be displayed with three circles, one around each reader node, intersecting at the location of the unknown node.

The Matlab code for 3D Trilateration is very similar to the MATAB code for 2D trilateration. You can see the 3D trilateration code in Appendix B, and see how much larger and complex it is compared to the 2D trilateration code. This is why the 3D trilateration computational time is exponentially larger than the 2D trilateration computational time. Also you will notice in the simulations that there are circles around each reader node in 2D trilateration and where those circles intersect is where the unknown object is located, but for 3D trilateration there are no circles. That's because for 3D trilateration there has to be spheres instead of circles and since we were under time constraints we decided to put a dot where the unknown object is located. Other than the computational time and the extra component, the basic idea of the 3D trilateration code and the 2D trilateration code is the same. For 3D trilateration there are two Matlab files, one Matlab file stores the distances between the node that is being searched for along with the known reader nodes, and this Matlab file can be seen in Appendix B, the other Matlab file was mentioned before and it has the actual 3D trilateration code. The simulation for both the 2D and 3D trilateration are located in the section called Trilateration Simulation Results (Section 3.7.1 and 3.7.2).

## 3.5 COLA (Complexity Reduced 3D Trilateration Localization Approach):

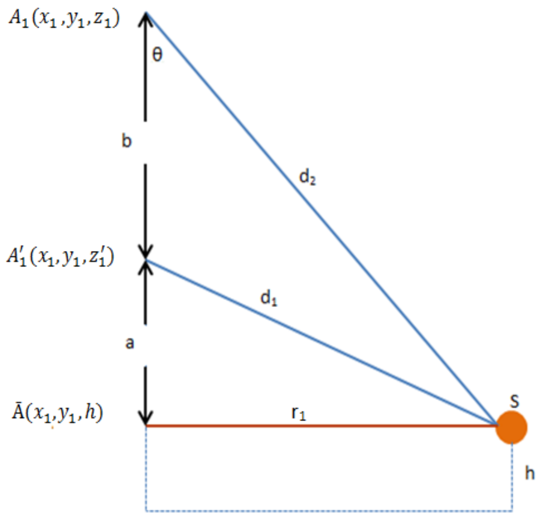


Figure 10: COLA 2D diagram

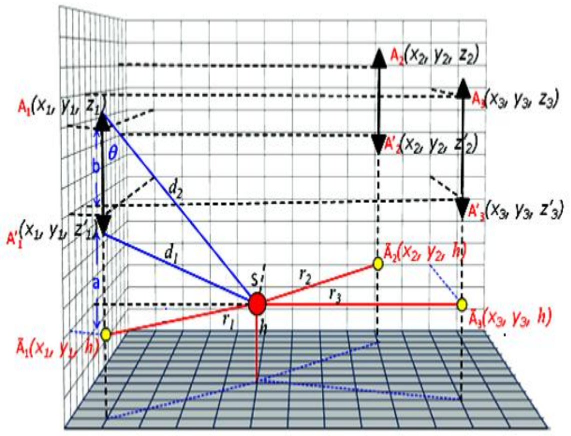


Figure 11: COLA 3D diagram

As mentioned before in the section 3.3**,** COLA can be used for 3D applications. COLA's computation time is much shorter than the traditional 3D trilateration, but COLA is much more expensive. The reason why it is more expensive is because three additional readers have to exist at the same XY axis as the original 3 readers, but have to be at an elevated location. This is the ideal method for tracking a tagged object indoors due to the fact that it is 60 percent more accurate than trilateration in 3D, the more readers, the more accurate the system will be. To find the height using COLA, complex algebra will be used and is shown below. To make this process easier to understand, take figure 10 and dissect it into 3 parts. To show how to find the height, the readers at $\bar{A}_i$ will be considered.

$$\cos\theta = \frac{a+b}{d_2} \qquad (Equation\ 18)$$

$$a = d_2\cos\theta - (z - z_i') \qquad (Equation\ 19)$$

$$d_2^2 = r_1^2 + (a + (z - z_i'))^2 \qquad (Equation\ 20)$$

$$d_2^2 = d_1^2 + 2(z_i - z_i')d_2\cos\theta - (z_i - z_i')^2 \qquad (Equation\ 21)$$

Using equation 20, cosθ can be solved for and gives the following equation:

$$\cos\theta = \frac{d_2^2 - d_1^2 + (z_i - z_i')^2}{2(z_i - z_i')d_2} \qquad (Equation\ 22)$$

The height equation is the following:

$$height = z_i - \left(a + (z_i - z_i')\right) \qquad (Equation\ 23)$$

where

$$\left(a + (z_i - z_i')\right) = (d_2\cos\theta - (z_i - z_i')) + (z_i - z_i') \qquad (Equation\ 24)$$

$$\left(a + (z_i - z_i')\right) = d_2\left[\frac{d_2^2 - d_1^2 + (z_i - z_i')^2}{2(z_i - z_i')d_2}\right] \qquad (Equation\ 25)$$

Substituting equation 23 into 22 gives us the height of the tagged object:

$$height = z_i - \left[\frac{d_2^2 - d_1^2 + (z_i - z_i')^2}{2(z_i - z_i')}\right] \qquad (Equation\ 26)$$

To find the distance($r_i$), which is the distance between the tagged object and the XY coordinate of the readers is as follows,

$$r_i = \sqrt{d_2^2(1 - (\cos\theta)^2)} \qquad (Equation\ 27)$$

Substituting equation 20 in for $d_2^2$ and equation 21 in for $\cos\theta$ the distance can be calculated with the formula below:

$$r_i = \sqrt{\frac{-d_1^4 - d_2^4 + 2(z_i - z_i')^2 d_1^2 + 2(z_i - z_i')^2 d_1^2 + 2d_1^2 d_2^2 - (z_i - z_i')^4}{4(z_i - z_i')^4}} \qquad (Equation\ 28)$$

The last step of COLA is performing 2D trilateration using the distances ($r$) between the tagged object and the reference readers.

## 3.6 Matlab Code for COLA

The COLA Matlab code has two Matlab files. One Matlab file is used for finding distance between the known reader and the unknown tag, this Matlab file is called COLA.m and can be found in Appendix B**.** The other Matlab file is used for finding height of the unknown tag, this Matlab file is called Cola_Height.m and can be found in Appendix B. You can see that this code is less complex than the traditional 3D trilateration code and this means that the COLA computation time is much faster. The location of the unknown tag will be known so we can find the distances between the readers and the tagged object. But when the whole Indoor positioning system is set up the distances will be known from the RSSI signal strength conversion coming from the Arduino. Once the distances between the readers and the tags are known, the distance between the XY coordinate of the reader and the unknown tag will be known by using COLA.m. Once the distances between the readers XY coordinate and the unknown tags are known then the two_tri.m file will be run to find the XY coordinate of the unknown tag. The height of the object will be found using the Cola_Height.m file and it will take the data from the COLA.m file to solve for height. Once the height is found it will be stored as a variable. After the program has been completed, the location of the object will be displayed to the screen. The simulation results of COLA will be shown in the section called Trilateration Simulation Results, Section 3.7.3.

## 3.7 Trilateration Simulation Results

### 3.7.1 Simulation for 2D Trilateration

Location of Reader 1: [-2, 2]
Location of Reader 2: [2, 1]
Location of Reader 3: [-1, -2]
Location of Unknown Tag: [1, 1]

In figure 12 each red dot symbolized one reader and the location of the unknown tag is located where all three circles intersect.

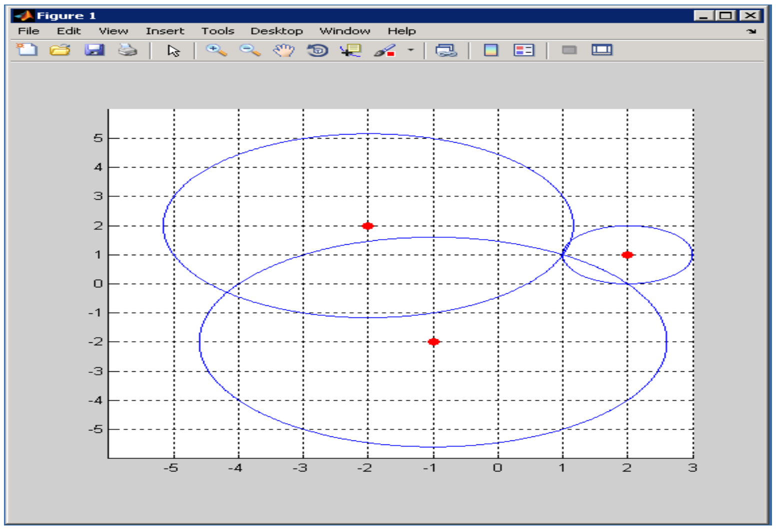


Figure 12: Simulation for 2D Trilateration

### 3.7.2 Simulation for 3D Trilateration

Location of Reader 1:[-2, 2, 2]
Location of Reader 2: [2, 1, -1]
Location of Reader 3: [-1, -2, 2]
Location of Reader 4: [3, 3, 3]
Location of Unknown Tag: [0, 0, 0]

For 3D trilateration, refer to figure 13. Each outer dot symbolized a reader. The location of the object is not shown with the intersection of spheres, but it is shown by a dot within the outer dots. Also, note that the readers are at different heights and the system still works correctly.

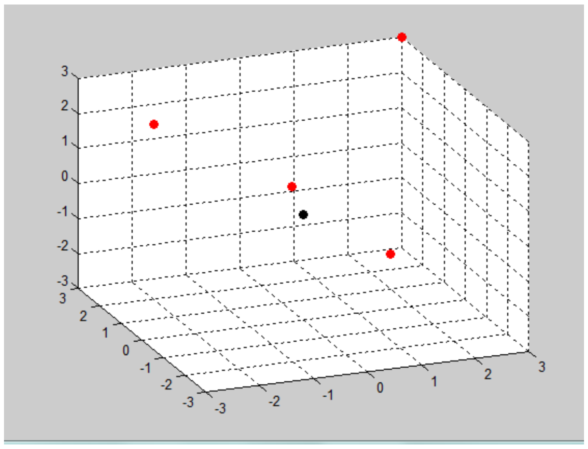


Figure 13: Simulation for 3D Trilateration

### 3.7.3 Simulation for COLA

Location of Lower Reader 1: [-2, 2, 2]
Location of Lower Reader 2: [1, 1, 2]
Location of Lower Reader 3: [-3, 0, 2]
Location of High Reader 1: [-2, 2, 4]
Location of High Reader 2: [1, 1, 4]
Location of High Reader 3: [-3, 0, 4]
Location of Unknown Tag: [4, 5, 1]

For the COLA simulation, refer to figure 14. Each red dot symbolizes a reader. Again, we are not able to use spheres to show the location of the unknown tag because this is in 3D; therefore, we show the location of the unknown tag with a black dot.

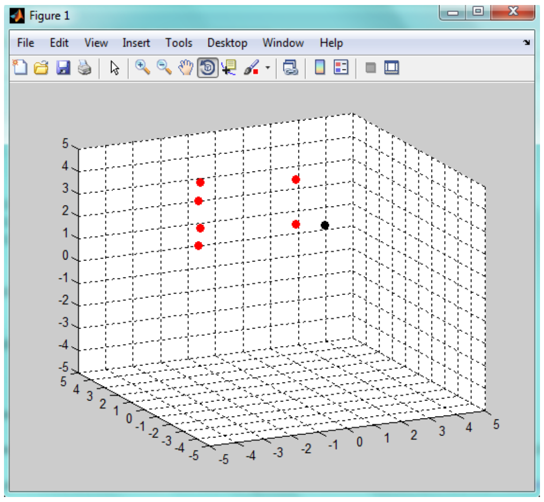


Figure 14: Simulation for COLA

## 3.8 Reader Stands for Trilateration

The reader stands were built originally for COLA, as you can see in figure 15, there is available space for a lower reader and a higher reader on each of the three stands. The original design was only about 2ft high. The reason for this design is because for COLA trilateration, as long as there are two different readers at two different elevations and have the same XY coordinate, height of the unknown tag can be found. With that in mind, the more compact the system is the more convenient it would be while we were demonstrating the indoor positioning system. Although it would be more convenient to have smaller stands, every time we tested the signal strength with the readers close to the ground the signal strength was inconsistent and weak. After doing some further research we realized that the ground is a good reflector of electromagnetic radiation. With the understanding that the earth could have possibly been reflecting our signal, it was decided to build the stands at approximately 5ft high. The height of these stands is where the lower readers will be. If COLA trilateration was to be tested, the stands would have to be extended to approximately 9 to 10 ft high for it to work properly.

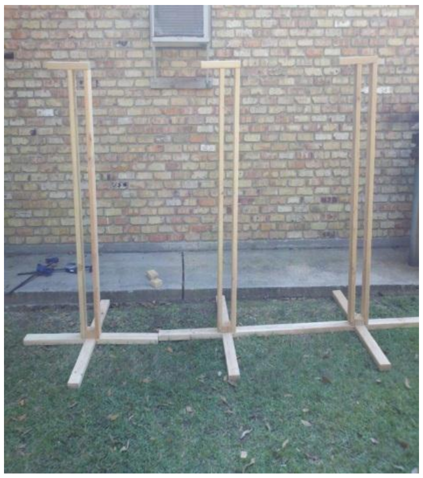


Figure 15: All Three Trilateration Stands

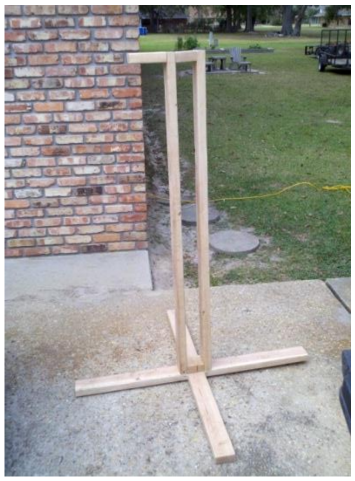


Figure 16: Trilateration Stand

## 3.9 Trilateration Conclusion

Using trilateration was a success in this project and the actual testing and proofs will be shown in detail later in this report. Although trilateration does not give us the exact location of the object in the actual field, it does get the user in very close proximity. Throughout this semester we were not able to get the COLA working in the indoor positioning system due to time constraints, but the 2D trilateration is working better than expected. We hope another group can compound on our studies and hard work to make COLA a success for the indoor positioning system. As I mentioned before, COLA is the ideal way to locate an object. It's just a system that uses 2D trilateration, and finds the height using trigonometry. It also used six readers, which makes it much more accurate than the traditional 2D and 3D trilateration.

## 4 RFID Detection Device

## 4.1 Introduction to the RFID Detection Device

The detection device was created to make the Indoor positioning system much more user friendly. The project had to keep in mind that not all people using our system will have computer skills and may not have any type of technical background. Instead of a user typing in the object that they want to find, one could use an RFID reader with RFID cards to bring up the location of a tagged object. The basic idea behind the detection device is, for every item that is tagged in the field, it will have a RF card that is linked to it. Each RFID card has a unique 12 digit hex number. For example, if RFID card #1 = 4500B8E95541 we can link Box #1 to RFID card #1 by referencing Box #1 with the same unique 12 digit ID in our database. For this system to work correctly, it will need RFID cards, an ID-12 detection chip, and an Arduino Uno. The datasheet for the ID-12 chip is listed in Appendix A. There were several issues that came up while building the device. The first one that that will be discussed is figuring out the unique ID for each RF card, the second challenge to overcome was sending the unique ID to Matlab so the host computer would know which object the user is looking for. The project's final challenge was making the system where multiple unique ID's are stored in the host computer, so the user can just scan each card and find each item in the system.

## 4.2 RF Card 12 Digit Unique ID

Finding out the 12 digit unique ID was an unexpected challenge for this semester. It was assumed that the manufacturer would give us the ID for each card, but this didn't happen. To solve this problem, the ID-12 chip was connected to an Arduino as shown in figure 17. The Arduino Uno can transmit serial data using the TX pin and receive serial data using the RX pin, due to the built in serial communication library. Pin 9, also known as the data 0 pin of the ID-12 chip, will output at 9600 baud serial every time it reads a card being scanned. The data 0 pin is connected to the RX pin of the Arduino and the RF card data will be stored into the Arduino's 128 byte serial receive buffer. Once the 12 digit hex number is in the buffer, the next goal is to display it to the screen.

To program the Arduino to display to the screen, we used the serial library that is already built into the Arduino. There are some key functions that are used in the program and they are listed below.

- Serial.begin(), opens the serial port and sets the data rate at some value of bits per second, for the serial data transmission.

- Serial.read(), tells the Arduino to read the incoming serial data.
- Serial.avaliable(), checks to see if there is any data in the 128 byte serial buffer
- Serial.print(), prints the data to the screen as a human readable ASCII character

The code that was used to find the 12 digit unique ID for the RFID cards is listed in Appendix C. The code is written to open the serial port of the Arduino Uno and set the output rate at 9600 bps. Next, the code will check to see if there is data in the serial receive buffer. If nothing is returned at this point the RFID card was not detected and it will need to be rescanned. Otherwise, if it returns the number of bytes available, the unique ID number will be read and printed out to the screen.

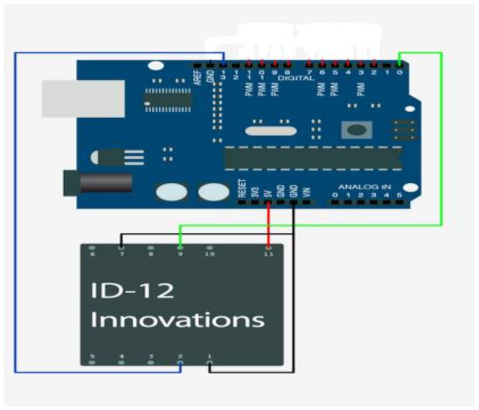
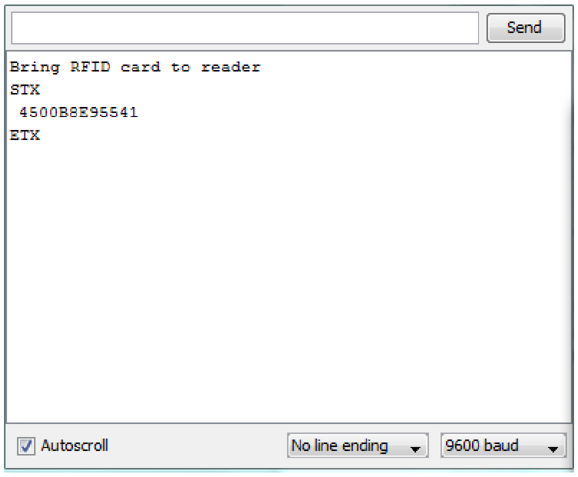


Figure 17: ID-12/Arduino circuit connection for Detection Device

## 4.3 Testing/Results for Finding 12 Digit Unique ID

The initial test with the circuit configuration in figure X10 and the original code, failed. At first, troubleshooting started with the ID-12 chip. To test the chip we connected a resistor and a LED light to PIN 5 of the Arduino Uno. A simple code was written to set pin 5 to a high voltage when a RFID card is detected. The experiment was tested and when the RFID card was scanned the LED would come on 50 percent of the time and stay off 50 percent of the time. This led to the conclusion of a bad connection within the breakout board or breadboard. With that in mind, the ID-12 chip was taken off the breakout board and soldered the PINS of the ID-12 chip directly to the wires that are connected to the Arduino Uno. Then we executed the test again and the LED light came on every time the RFID card was scanned. The conclusion from this test was that the ID-12 chip had good connections now; therefore, the original code was then uploaded to the Arduino. The original experiment was tested once again and when the RFID card was scanned the 12 digit unique ID was displayed on the screen as shown in figure 18. This allowed the group to move forward with the project knowing the ID number that will link the RFID card to the tag that is out in the field.



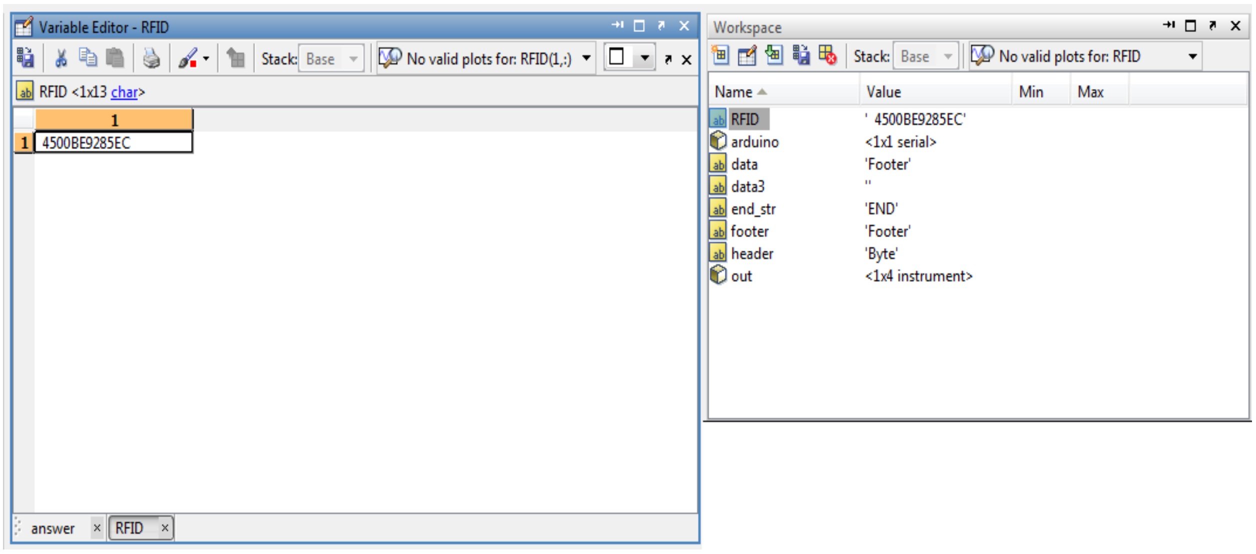
```
Bring RFID card to reader
STX
 4500B8E95541
ETX
```

Figure 18: Display of Unique ID

## 4.5 Sending 12 Digit Unique ID to Host Computer

Since the ID for each RF card is known, a database is made within the Matlab program and this will allow us to run a comparison to know which tagged object the user is looking for. A 12 digit unique ID must be sent to Matlab from the Arduino Uno through the serial port for comparison of tags in the database. The newer versions of Matlab are capable of receiving and sending serial data using the communications port on a computer. For the 12 digit unique ID to be sent serially, two separate codes were written; an Arduino code and a Matlab code. Both codes are shown in Appendix C and B respectively.

For the Arduino code, additional code was added to the code that was written to display the ID number on to the screen. Instead of displaying it on to the screen, it is sent to the Matlab program. This is done by taking the ID number that is stored in the buffer and sent it through the communications port one byte at a time. The reason why it's sent one byte at a time is because that's what the Arduino Uno is programmed to do internally. For the Matlab code, Arduino COM port is opened as a file which allows the Matlab program to receive the data coming from the communications port. The Matlab code is written to receive the 12 digit ID one byte at a time since the Arduino is sending it at that rate. If both the Matlab program and the Arduino program are not sending and receiving the data at the same rate, it will not be effective. The proof that these two codes work together is shown below. In figure 18¸located in the previous section, one can see the unique ID (4500BE9285EC) and the 12 digit unique ID in Figure 19 are the same. The unique ID is sent to Matlab from the Arduino as a string of characters. Therefore when the unique ID's are programmed in to the Matlab database, they are stored as strings. This allows the user to write a string comparison code to find the tagged object that the user is looking for. Now that the unique ID's are sent to Matlab, the system is able to compare them and know which object the user is looking for.


Variable Editor - RFID

RFID <1x13 char>

|  | 1 |
|---|---|
| 1 | 4500BE9285EC |

Workspace

| Name | Value | Min | Max |
|---|---|---|---|
| RFID | ' 4500BE9285EC' | | |
| arduino | <1x1 serial> | | |
| data | 'Footer' | | |
| data3 | '' | | |
| end_str | 'END' | | |
| footer | 'Footer' | | |
| header | 'Byte' | | |
| out | <1x4 instrument> | | |

Figure 19: Display of Unique stored in Matlab

## 4.6 Detection Device Conclusion

The detection device is very simple to use. Once the user knows which object he/she wants to locate all they have to do is select the RFID card that is linked to that specific object. The card will be scanned by the ID-12 chip which is on the host computer box. After the card is scanned a graphical user interface will be displayed with the location of the object. As you can see in figure 20, the ID-12 chip is mounted on the outside of the host computer's box for easy access. The host computer box includes one of the readers, the host computer's Arduino (the Arduino MEGA) and the ID-12 detection device. Although, there were some difficulties with finding the unique ID numbers and sending that data to Matlab, the challenges were overcome and the system was successfully completed.

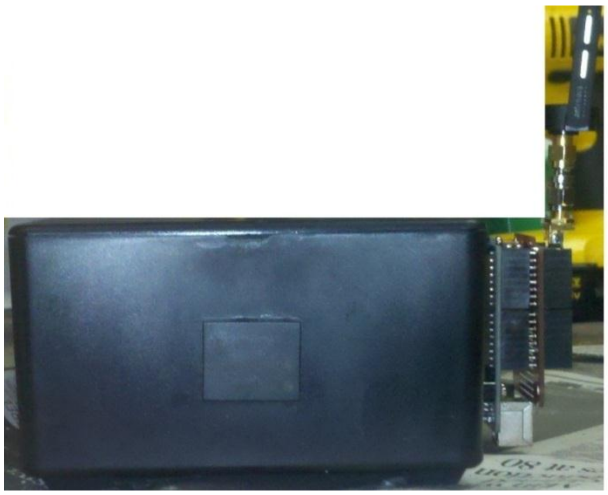


Figure 20: Host Computer Box