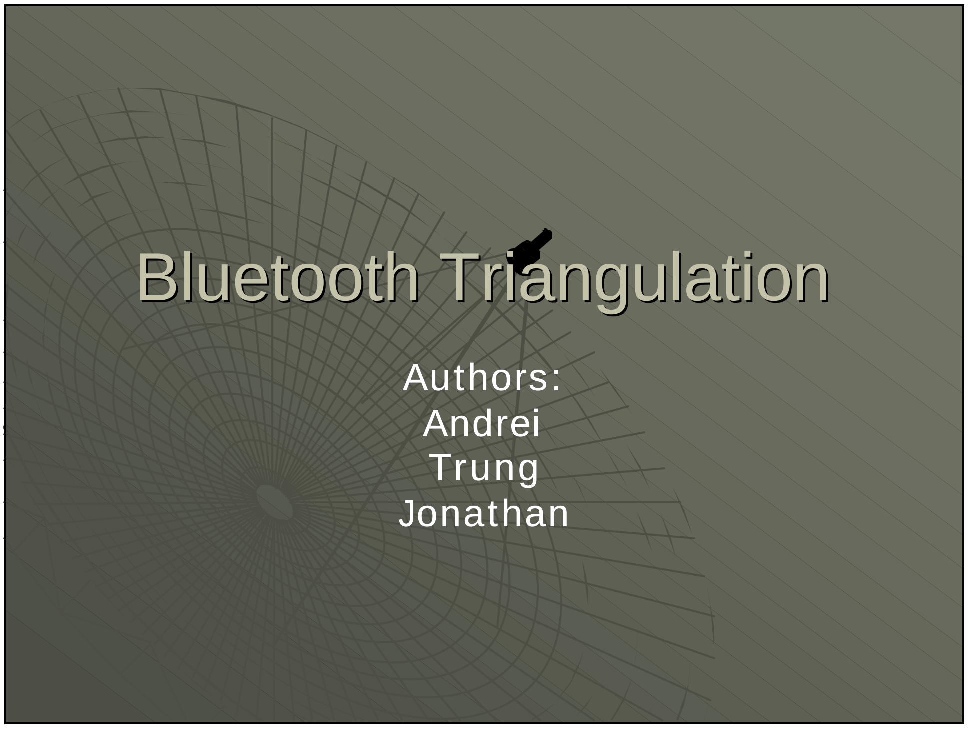
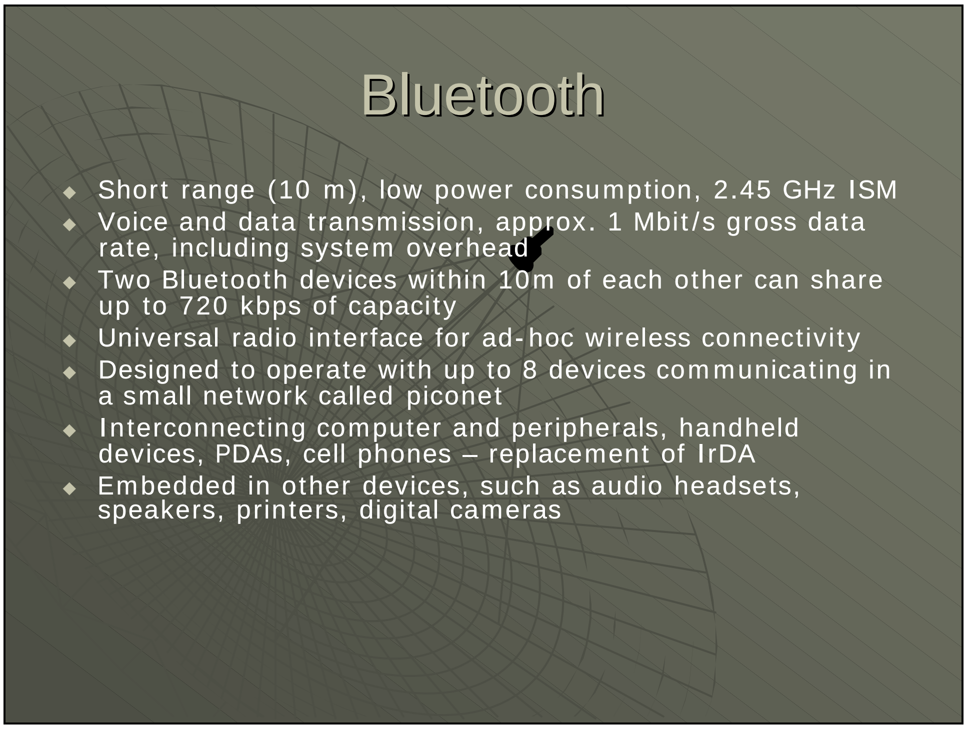


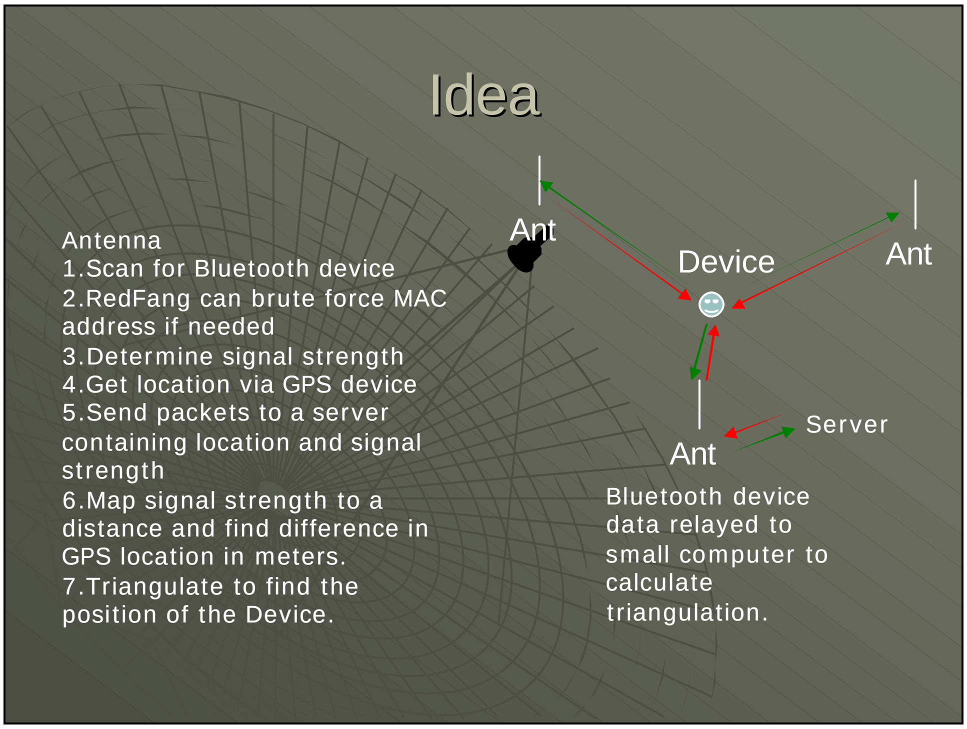
Bluetooth Triangulation

Authors:
Andrei
Trung
Jonathan



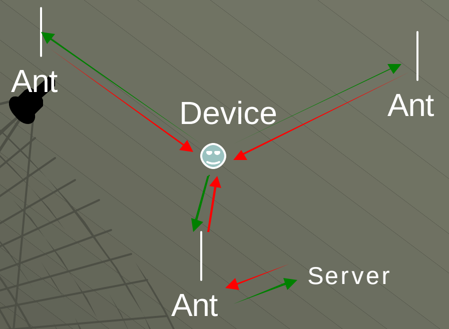
Bluetooth

- Short range (10 m), low power consumption, 2.45 GHz ISM
- Voice and data transmission, approx. 1 Mbit/s gross data rate, including system overhead
- Two Bluetooth devices within 10m of each other can share up to 720 kbps of capacity
- Universal radio interface for ad-hoc wireless connectivity
- Designed to operate with up to 8 devices communicating in a small network called piconet
- Interconnecting computer and peripherals, handheld devices, PDAs, cell phones – replacement of IrDA
- Embedded in other devices, such as audio headsets, speakers, printers, digital cameras

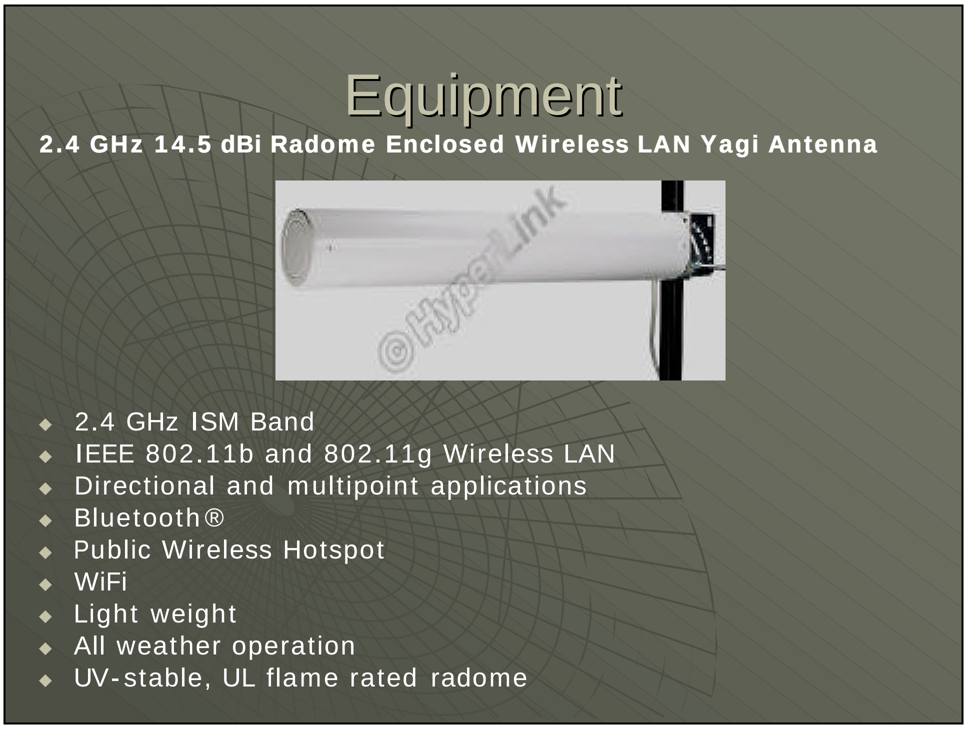


Idea

Antenna
1.Scan for Bluetooth device
2.RedFang can brute force MAC address if needed
3.Determine signal strength
4.Get location via GPS device
5.Send packets to a server containing location and signal strength
6.Map signal strength to a distance and find difference in GPS location in meters.
7.Triangulate to find the position of the Device.



Bluetooth device data relayed to small computer to calculate triangulation.



Equipment

2.4 GHz 14.5 dBi Radome Enclosed Wireless LAN Yagi Antenna

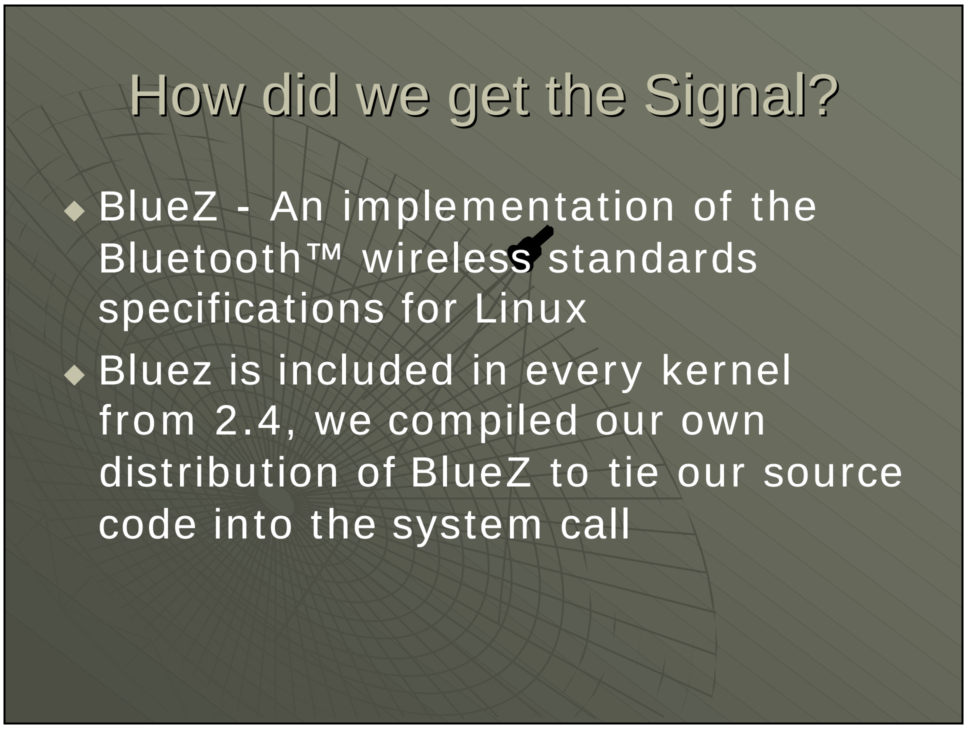
- 2.4 GHz ISM Band
- IEEE 802.11b and 802.11g Wireless LAN
- Directional and multipoint applications
- Bluetooth®
- Public Wireless Hotspot
- WiFi
- Light weight
- All weather operation
- UV-stable, UL flame rated radome

Linksys USBBT100 Bluetooth Adapter

- Add Bluetooth Class 1 connectivity to your USB-equipped notebook or desktop computer
- Wirelessly connect to local devices such as mobile phones, PDAs, printers, headsets, other computers, mice, keyboards, and more

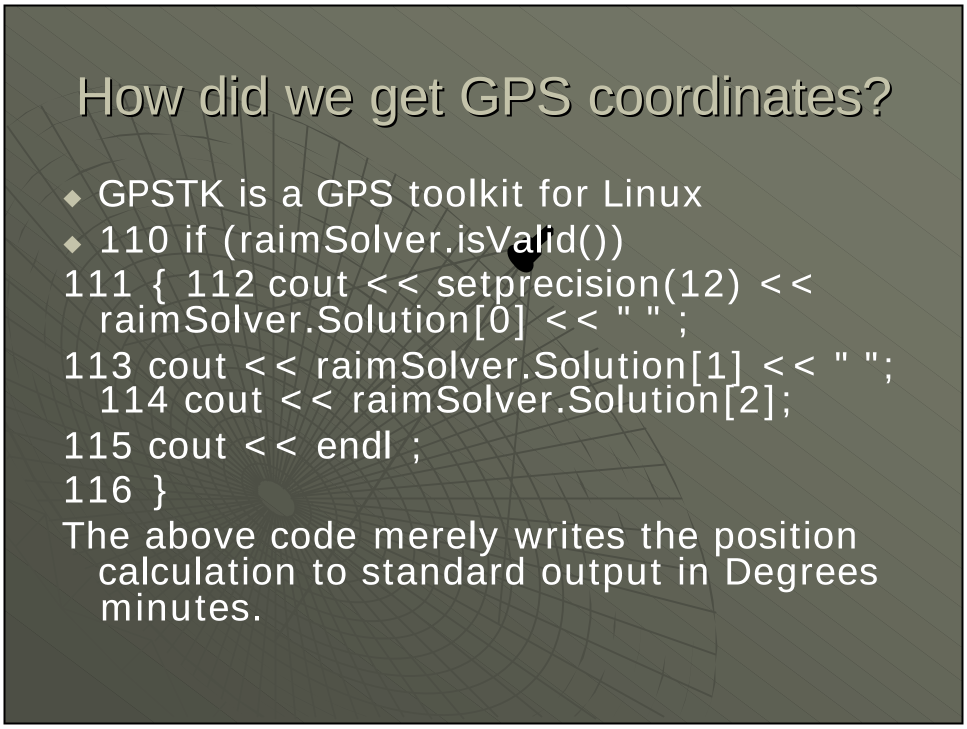
Sony Ericsson P800 phone

- Mobile phone with bluetooth support
- Networking capabilities



How did we get the Signal?

- BlueZ - An implementation of the Bluetooth™ wireless standards specifications for Linux
- Bluez is included in every kernel from 2.4, we compiled our own distribution of BlueZ to tie our source code into the system call



How did we get GPS coordinates?

- GPSTK is a GPS toolkit for Linux
- 110 if (raimSolver.isValid())

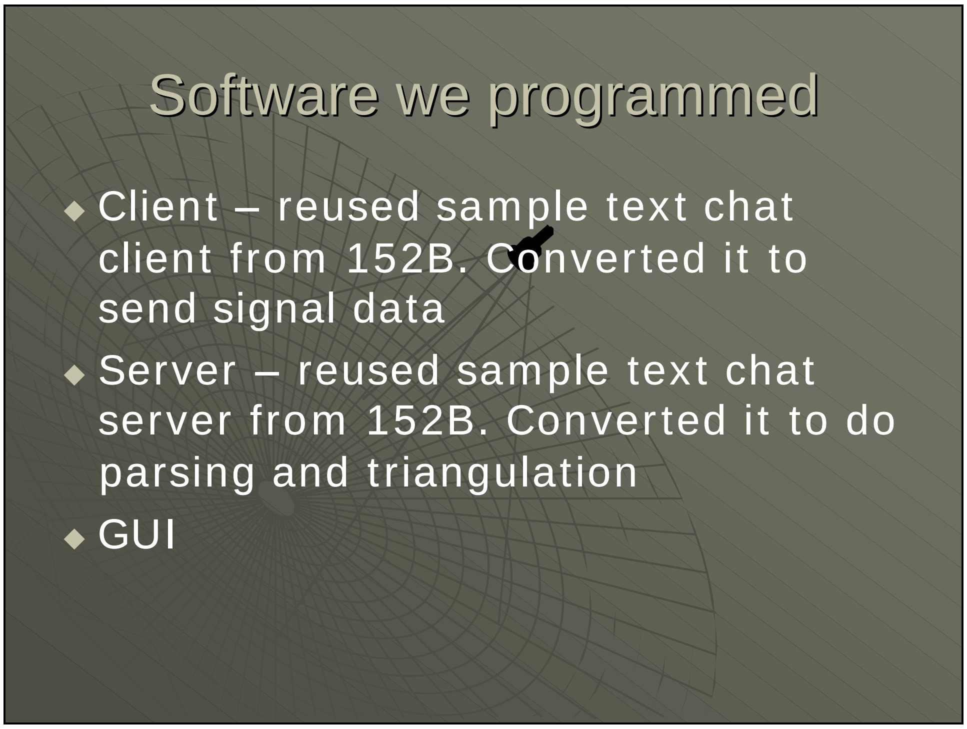
111 { 112 cout << setprecision(12) << raimSolver.Solution[0] << " " ;

113 cout << raimSolver.Solution[1] << " "; 114 cout << raimSolver.Solution[2];

115 cout << endl ;

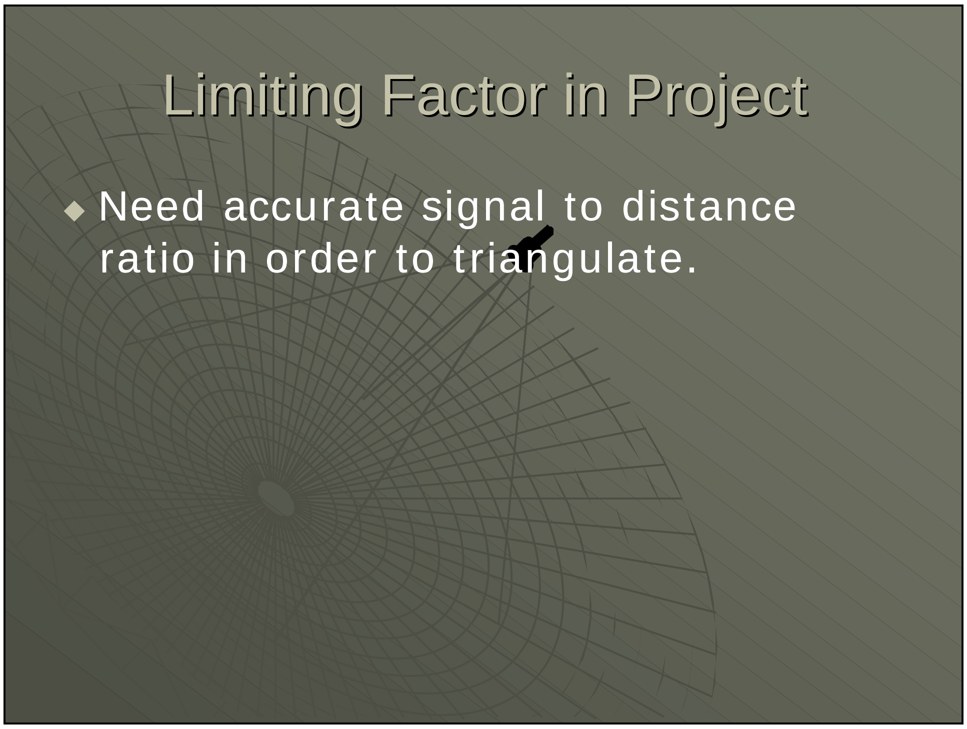
116 }

The above code merely writes the position calculation to standard output in Degrees minutes.



Software we programmed

- Client – reused sample text chat client from 152B. Converted it to send signal data
- Server – reused sample text chat server from 152B. Converted it to do parsing and triangulation
- GUI



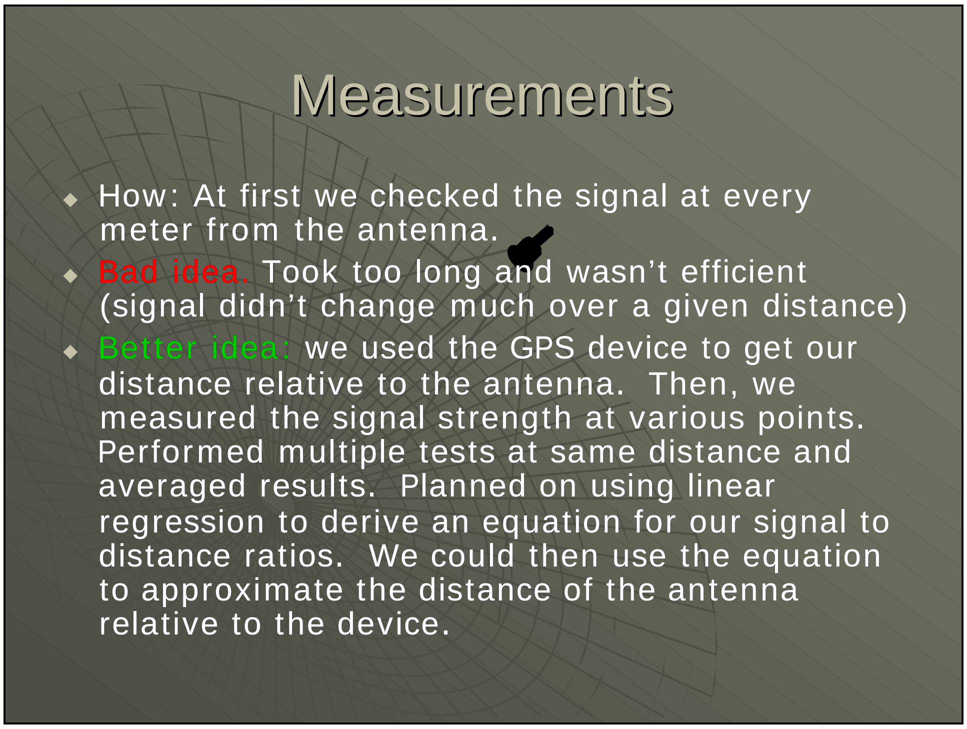
Limiting Factor in Project

- Need accurate signal to distance ratio in order to triangulate.

Measurements

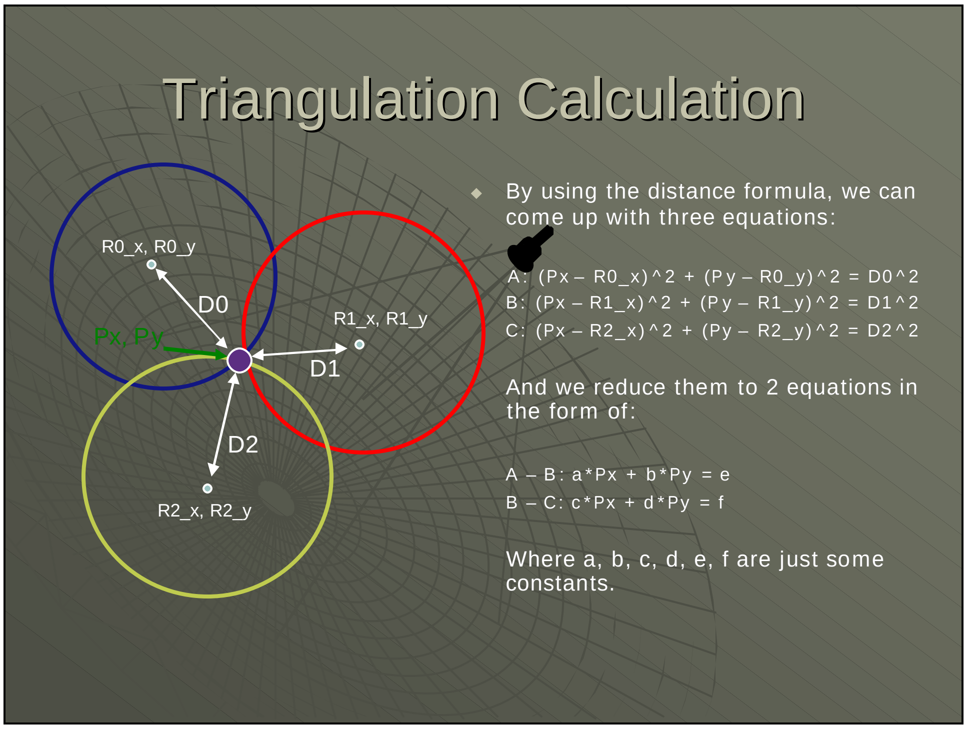
- Where: Some random country road south of Davis.

Measuring the signal

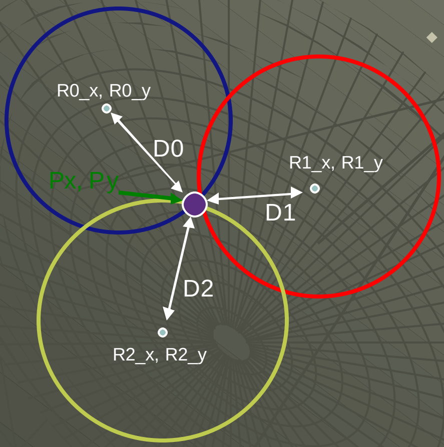


Measurements

- How: At first we checked the signal at every meter from the antenna.
- Bad idea. Took too long and wasn't efficient (signal didn't change much over a given distance)
- Better idea: we used the GPS device to get our distance relative to the antenna. Then, we measured the signal strength at various points. Performed multiple tests at same distance and averaged results. Planned on using linear regression to derive an equation for our signal to distance ratios. We could then use the equation to approximate the distance of the antenna relative to the device.



Triangulation Calculation



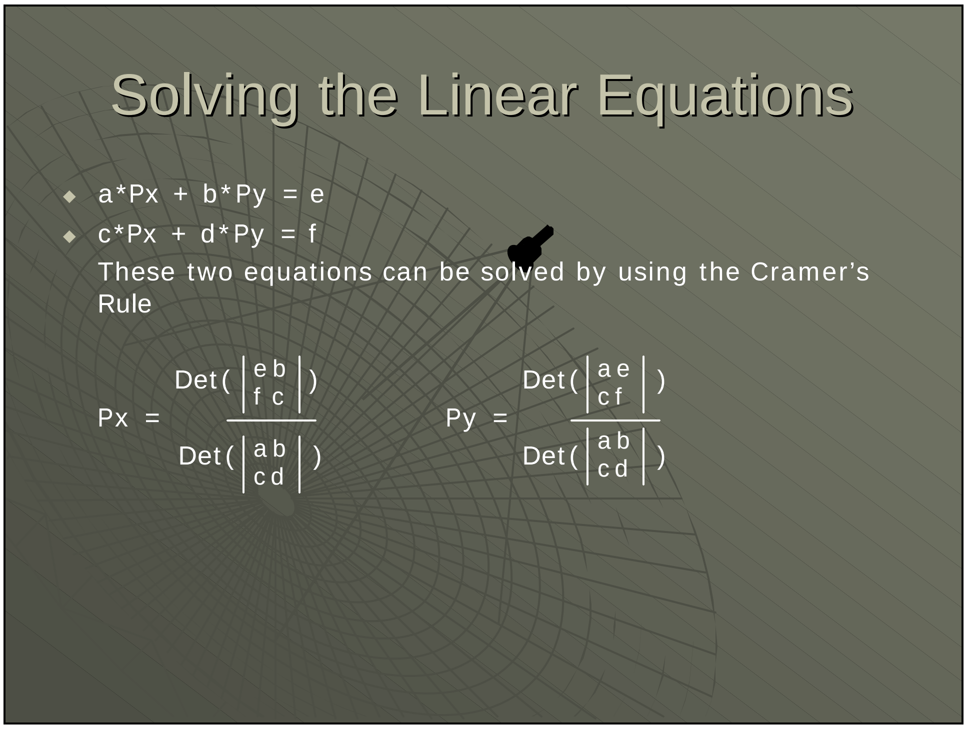
- By using the distance formula, we can come up with three equations:

$A: (Px - R0\_x)^2 + (Py - R0\_y)^2 = D0^2$
$B: (Px - R1\_x)^2 + (Py - R1\_y)^2 = D1^2$
$C: (Px - R2\_x)^2 + (Py - R2\_y)^2 = D2^2$

And we reduce them to 2 equations in the form of:

$A - B: a*Px + b*Py = e$
$B - C: c*Px + d*Py = f$

Where a, b, c, d, e, f are just some constants.

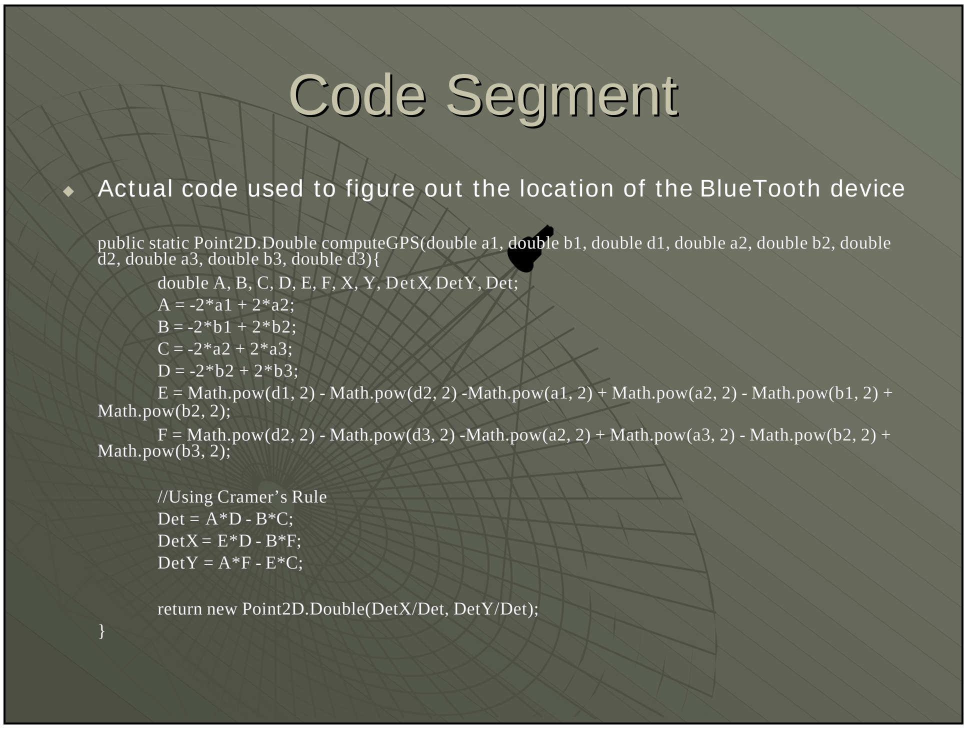


Solving the Linear Equations

- a*Px + b*Py = e
- c*Px + d*Py = f

These two equations can be solved by using the Cramer's Rule

$$Px = \frac{Det\left(\begin{vmatrix} e & b \\ f & c \end{vmatrix}\right)}{Det\left(\begin{vmatrix} a & b \\ c & d \end{vmatrix}\right)} \qquad Py = \frac{Det\left(\begin{vmatrix} a & e \\ c & f \end{vmatrix}\right)}{Det\left(\begin{vmatrix} a & b \\ c & d \end{vmatrix}\right)}$$



Code Segment

- Actual code used to figure out the location of the BlueTooth device

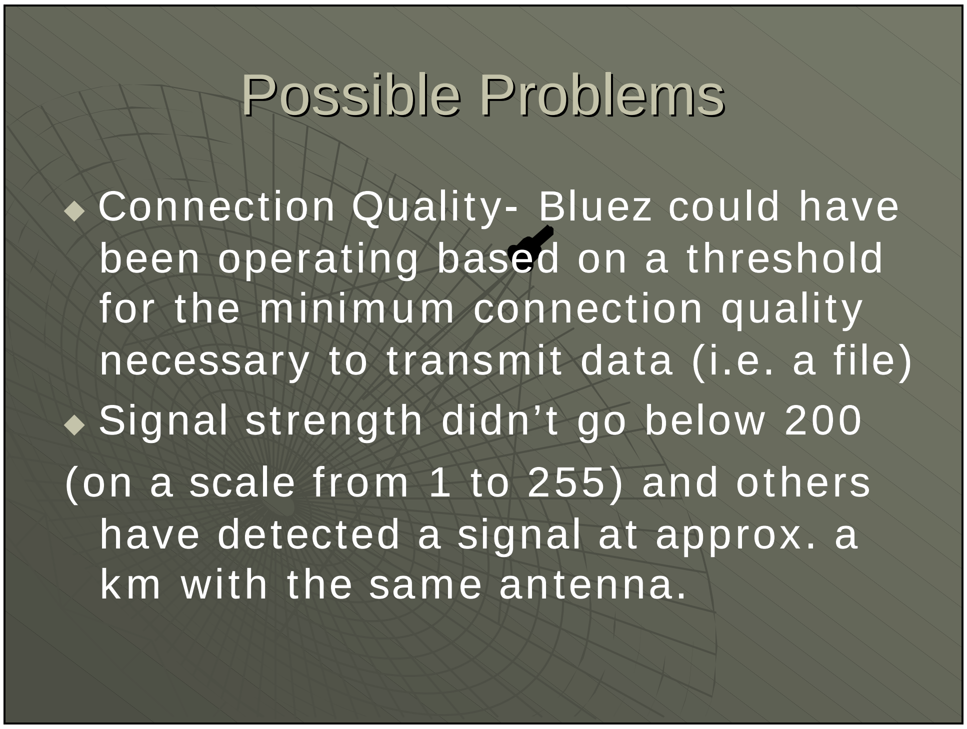
```
public static Point2D.Double computeGPS(double a1, double b1, double d1, double a2, double b2, double
d2, double a3, double b3, double d3){
        double A, B, C, D, E, F, X, Y, DetX, DetY, Det;
        A = -2*a1 + 2*a2;
        B = -2*b1 + 2*b2;
        C = -2*a2 + 2*a3;
        D = -2*b2 + 2*b3;
        E = Math.pow(d1, 2) - Math.pow(d2, 2) -Math.pow(a1, 2) + Math.pow(a2, 2) - Math.pow(b1, 2) +
Math.pow(b2, 2);
        F = Math.pow(d2, 2) - Math.pow(d3, 2) -Math.pow(a2, 2) + Math.pow(a3, 2) - Math.pow(b2, 2) +
Math.pow(b3, 2);

        //Using Cramer's Rule
        Det = A*D - B*C;
        DetX = E*D - B*F;
        DetY = A*F - E*C;

        return new Point2D.Double(DetX/Det, DetY/Det);
}
```

Data was garbage

Trung doesn't like bad data

Blame the equipment!!!

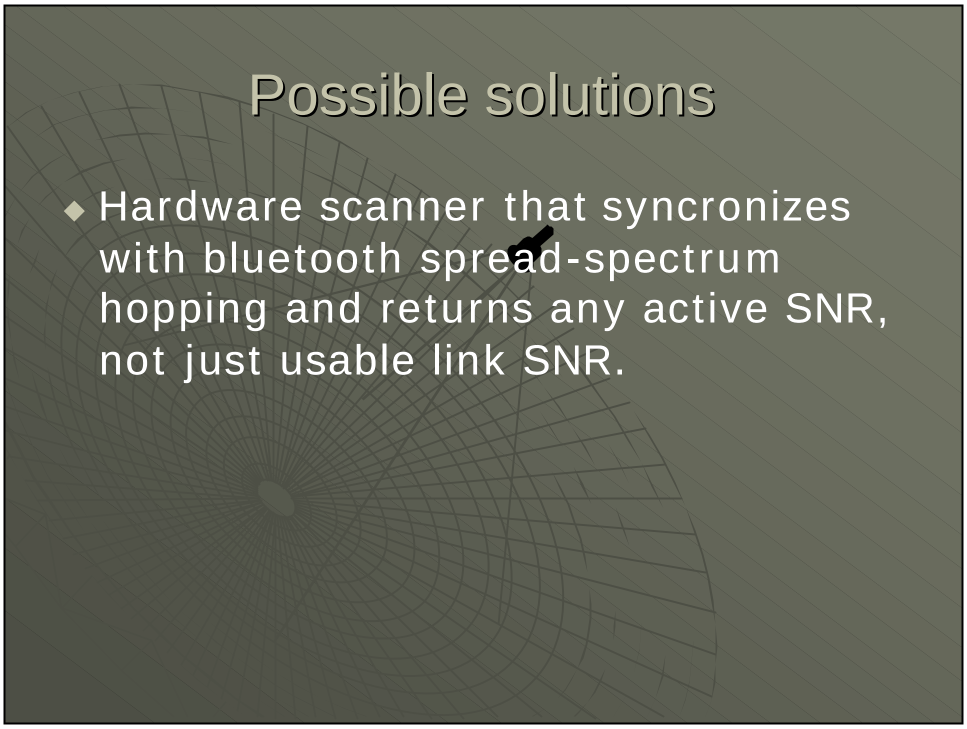
Better yet, blame the guy that suggested this lame project



Possible Problems

- Connection Quality- Bluez could have been operating based on a threshold for the minimum connection quality necessary to transmit data (i.e. a file)
- Signal strength didn’t go below 200

(on a scale from 1 to 255) and others have detected a signal at approx. a km with the same antenna.



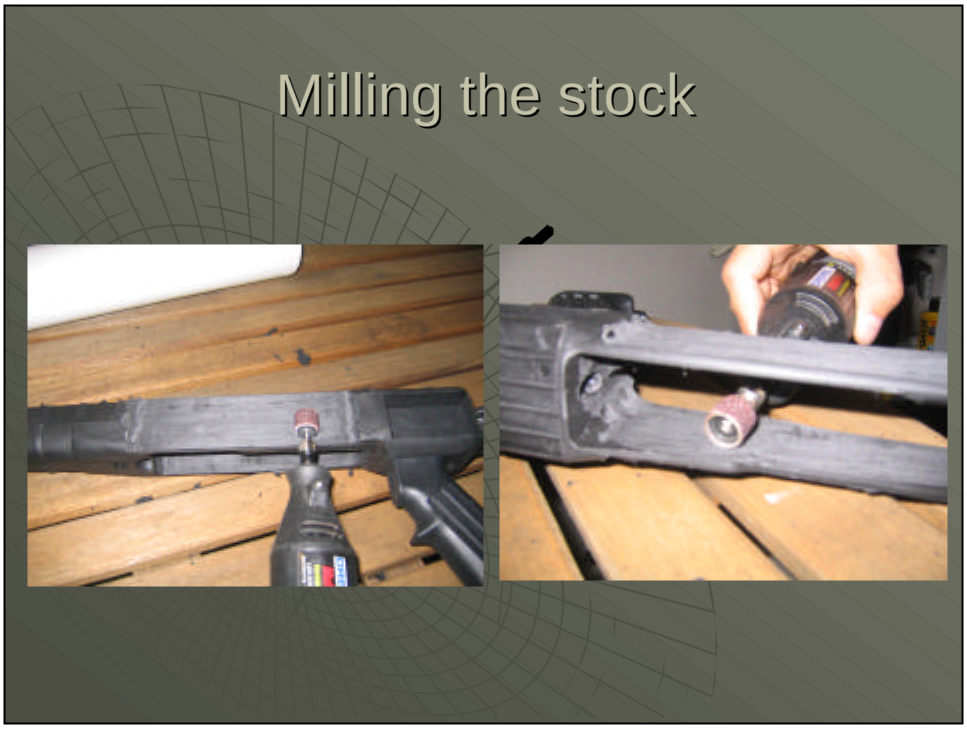
Possible solutions

- Hardware scanner that syncronizes with bluetooth spread-spectrum hopping and returns any active SNR, not just usable link SNR.

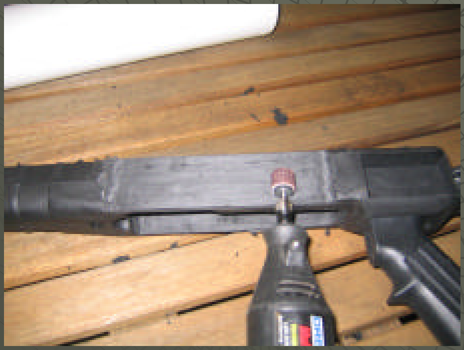
Our finished bluetooth scanner

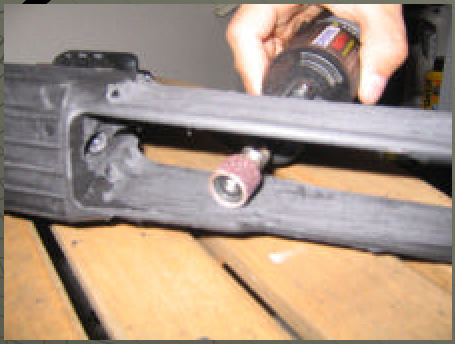
- Unveil the scanner

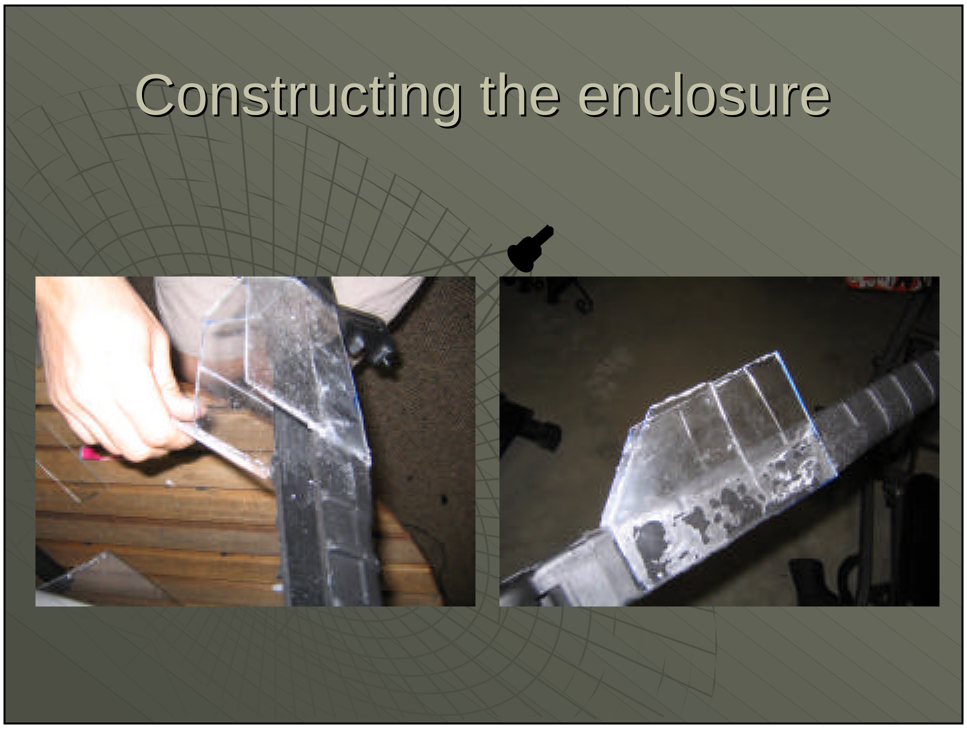
Modifying antenna to fit USB adapter



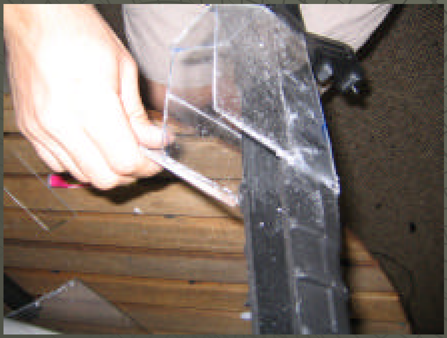
Milling the stock

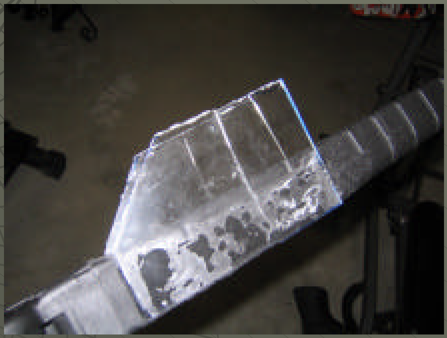






Constructing the enclosure





Painting and fitting

The end.

- Demo the scanner and GUI