Acknowledgements

Dr. Neal Patwari
Piyush Agrawal
University of Utah Faculty
L–3 Communications

Motivation and Overview for Using Angle of Arrival in Wireless Sensor Networks

Shawn Solberg
3/27/08
Existing Networks

- GPS
  - Accurate up to 5 feet
  - 2 Dimensional
  - Chip cost upwards of $100
  - Only works outdoors

Existing Networks

- Wireless Sensor Network
  - Works on any plane (3D)
  - Chip cost as low as $1
  - Low Range of Operation
  - Works Indoors
  - Prone to Errors
Wireless Sensor Network

Applications
- Localization
  - People
    - Firefighters/Servicemen/S.W.A.T.
  - Inventory
  - Tools/Equipment

MICA2
- Features include
  - Accelerometer
  - Microphone
  - Motion/Light Detector
  - Speaker
  - LEDs
  - etc
How it Works

- MICA2 Motes use Received Signal Strength to determine Distance between Motes.
  - Signal Strength Decays Proportional to distance \( d \) and Path-Loss Exponent \( n_p \) Typically Between 2 and 4.[1]

  \[
P(d) = P_0 - 10n_p \log(d/d_0)
\]

  Where \( P_0 \) is the Transmitted Power in dBm at a Reference Distance \( d_0 \).

For Example:

Consider a predefined area:
The Locations of Nodes A, B, C, and D are known.
For Example:

Consider a predefined area:
The Locations of Nodes A, B, C, and D are known.

We can then locate nodes that are randomly placed within this area (nodes 1, 2,
For Example:

Consider a predefined area: The Locations of Nodes A, B, C, and D are known.

We can then locate nodes that are randomly placed within this area (nodes 1, 2, and 3).

The nodes do two things:
First, they transmit a signal at a constant power (or dB) level.
The nodes do two things:
First, they transmit a signal at a constant power (or dB) level.
Second, they use the RSS value to calculate distance.
In this way, ‘A’ knows how far away ‘1’ is,
The nodes do two things:
First, they transmit a signal at a constant power (or dB) level.
Second, they use the RSS value to calculate distance.
In this way, ‘A’ knows how far away ‘1’ is, ‘B’ knows how far away ‘1’ and ‘3’ are, ‘C’ knows how far ‘2’ and ‘3’ are, and ‘D’ knows how far ‘1’ and ‘2’ are.
The nodes do two things:

First, they transmit a signal at a constant power (or dB) level.

Second, they use the RSS value to calculate distance.

In this way, ‘A’ knows how far away ‘1’ is, ‘B’ knows how far away ‘1’ and ‘3’ are, ‘C’ knows how far ‘2’ and ‘3’ are, and ‘D’ knows how far ‘1’ and ‘2’ are.

Nodes 1, 2 and 3 also know their distance.

How it Works

- This builds a network and ideally each node uses simple trigonometry to find its location.
Does it Work?

- The RSS-based distance values differ from one half of the actual distance to double the actual distance.
- These errors are caused by multi-path fading and still exist after implementing Frequency Hopping (900-916MHz).

Solution

- Find the Signals Angle of Arrival
  - Phased-Array Antennas
  - Directional Antennas
Getting Started

- Break-Up Responsibilities
  1. Research Directional Antennas
     - Develop a Function that calculates AoA
  2. Improve RSS Algorithm to Include AoA
     - Learn TinyOS and NesC
       • Tutorials on MICA2

Next Up

- Paul Pryor
  - Discussion of Antenna Design and Characterization for a 3D Array
- Joseph Peterson
  - Development of Self-Localization Algorithm Based on Angle of Arrival
- Jonathan Gorzitze
  - Angle of Arrival Sensor Network An Extension to a Received Signal Strength Sensor Network
- Luan Nguyen
  - MICA2/TinyOS: Implementation and Applications
References

“Locating the Nodes” Neal Patwari, Joshua N. Ash, Spyros Kyparountas, Alfred O. Hero III, Randolph L. Moses, and Neiyer S. Correal

DISCUSSION OF ANTENNA DESIGN AND CHARACTERIZATION FOR 3D-SENSOR ARRAY

Paul Pryor
3/27/08
Table of Contents

I. Antenna and Module Design
II. Experimental Results
III. Future Design Considerations for the Antenna and Module

Antenna Types Considered

- Patch
- Yagi
- Log-Periodic (Special Yagi)
Patch Antenna

- Advantages
  - Small
  - Less Expensive

- Disadvantages
  - Not Off the shelf
Yagi Antenna

- Advantages
  - Directional
  - Inexpensive
  - Small
  - Available
- Disadvantages
  - Gain Low

Yagi Antenna

<table>
<thead>
<tr>
<th>Antenna Type</th>
<th>Radiation Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>YAGI</td>
<td><img src="image" alt="Radiation Pattern" /></td>
</tr>
</tbody>
</table>

![Diagram of Yagi Antenna](image)
Actual Pattern

-40db

-35db

Log-Periodic

- Advantages
  - Directional
  - High Gain
- Disadvantages
  - Larger
  - Expensive
Log-Periodic

Actual Pattern

-35 db
PCB Antennas

900 MHz to 2.4 GHz
400 MHz – 1000 MHz

915 MHz

Module Designs

Vertical
Horizontal
Actual Module

Experiment Results

Setup Results
Radiation Pattern

Received Path-Loss For 90°

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>3</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>25</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>24</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>24</td>
<td>26</td>
<td></td>
</tr>
</tbody>
</table>

Values are in db
30° Test

Received Path-loss For 30°

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>3</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>25</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>24</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>25</td>
<td>29</td>
<td></td>
</tr>
</tbody>
</table>

Values are in db
Final Test Setup

Received Path-Loss

<table>
<thead>
<tr>
<th>Mote #</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

Path-Loss (db) | 28 | 27 | 41 | 27 | 21 | 20 | 31 | 33 | 44
Future Considerations

- 2.4 GHz
- One IC
- All in one Package
- Patch Antenna

Antennas Too Large
One Chip

- Small
- Less Expensive

Patch Antenna

- Advantages
  - Small
  - Less Expensive

- Disadvantages
  - Not Off the shelf
The Future Module

Patch Antenna

Trapezoidal Patch:

- 916.5 MHz
- 25 mm
- 22 mm
- Via to ground
- Circuit area
Patch Antenna Pattern

Trigonal Patch over a Large Ground Plane
(914.5 MHz)

Conclusion

- Project Review
- Antenna and Module
- Future Design
References

Design of Self-Localization Algorithm & MATLAB Platform for Testing

Joseph Peterson
3/27/08

Table of Contents

I. Overview of Self-Localization
II. 1st Try: Angles from Triplets
III. 2nd Try: Angles from Pairs
IV. MATLAB Demonstration
Self Localization
Phase 1: Data Reception

Each Node obtains
- Reported Coordinates
- Distance/Angle Measurements
For Other Nodes in Group

Self Localization
Phase 2: Optimization

Algorithm Compares
- Reported Locations with Predicted Locations
  - Based on Measurements
  - & Node’s Assumed Position
- Node Re-Estimates Own Coordinates
Self Localization
Phase 3: Transmission

Node Broadcasts
New, Better Estimate
of Its Own Coordinates

Self Localization
Phase 4: Convergence

All Nodes Simultaneously
Revise Own Coordinates
& Broadcast.

After Many Iterations,
All Nodes in Network Located.
Self Localization Assumptions

- 2-Dimensional
- Stationary
- Some Nodes Pre-Localized
  (Programmed w/ Own Coordinates)

Strategy

Cost Function = $\Sigma (\text{Reported-Predicted})^2$

Solution: Find Coordinates to Minimize Cost Function
Angle Between Two Nodes Must Assume a Reference “North”

Each Node has its Own Reference “North”

Disadvantage: Conflicting Assumptions = Conflicting Estimates
Triplets:
Angle Between 3 Nodes do Not Need Reference “North”

Advantages:
1. Simpler Hardware
2. No Orientation Error

Disadvantages:
1. Extra Hardware
2. New Source of Error

All Nodes Standardize to the Same Reference “North”
Triplets:
First Disadvantage-
Combinatorial Complexity

\[ N = \text{# nodes in group} \]

Pairs: \[ N\times(N-1) \]

Triplets: \[ N\times(N-1)\times(N-2) \]

Triplets:
Second Disadvantage-
Intricate Computation

Algorithm Developed

But Dead End

Not Implemented in MATLAB
First Try: Triplets

Local Cost Function:

\[
S_q = \sum_{t=1}^{T} \left( \sum_{i \neq q}^{N+M-1} \sum_{k \neq q}^{N+M} w_{iqk}^t \left( \cos \phi_{iqk}^t - \cos \theta_{iqk}^t \right)^2 \right) + \sum_{j=N+1}^{N+M} \sum_{k=q+1}^{N+M} w_{jqk}^t \left( \cos \phi_{jqk}^t - \cos \theta_{jqk}^t \right)^2
\]

First Try: Triplets

Introduce Some Short-hand:

\[
\cos \theta_{ijk} = \frac{(\bar{x}_i - \bar{x}_j) \cdot (\bar{x}_k - \bar{x}_j)}{\delta_{ij} \cdot \delta_{kj}} = \frac{L_{ijk}}{\delta_{ij} \cdot \delta_{kj}}
\]

\[
G_{ijk} = \beta_{ijk}^2 = \sum_{t=1}^{T} \left( \frac{w_{ijk}^t}{\delta_{ij} \cdot \delta_{kj}} \right)^2
\]

\[
H_{ijk} = \alpha_{jk}^t \beta_{ijk} = \sum_{t=1}^{T} \frac{w_{iqk}^t \cos \phi_{iqk}^t}{\delta_{ij} \cdot \delta_{kj}}
\]
First Try: Triplets

Differentiate to Minimize:

\[
\frac{\partial S_q}{\partial x_q} = 2 \sum_{j=N+1}^{N+M} \sum_{k=q+1}^{N+M} \left( -H_{qjk} [x_k - x_j] + G_{qjk} \left[ (x_k - x_j)^2 x_q - x_j (x_k - x_j)^2 \right] \right) + 2 \sum_{i=1}^{N+M-1} \sum_{k=q+1}^{N+M} \left( -H_{iqk} \left[ x_i^2 - (x_i + x_k) x_q + x_i x_k \right] + G_{iqk} \left[ 2x_i^3 - 3(x_i + x_k) x_i^2 + (x_i^2 + 4x_i x_k + x_k^2) x_q - x_i x_k (x_i + x_k) \right] \right)
\]

First Try: Triplets

Derivative is a Cubic Polynomial:

\[
\frac{1}{2} \frac{\partial S_q}{\partial x_q} = Ax_q^3 + Bx_q^2 + Cx_q + D
\]

\[
A = 2 \ r_1 \\
B = -3 \ r_2 \\
C = r_3 - 2 \ r_4 + r_5 \\
D = -r_6 - r_7 + r_8 - r_9
\]
First Try: Triplets

Solve Using Lagrange Resolvents:

Discriminant of Polynomial:
\[ \Delta = 4B^3D - B2C2 + 4AC3 - 18ABCD + 27A2D2 \]

Let \[ z_1, z_2 = \frac{9 BC}{2 A^2} - \left( \frac{B}{A} \right)^3 - \frac{27 D}{2 A} \pm \frac{3}{2} \sqrt{3 \Delta} \]

Root:
\[ p_1 = \frac{1}{3} \left( -\frac{B}{A} + z_1^{1/3} + z_2^{1/3} \right) \]

Back to Square One… Solution:

Each Node Guesses
1. Its Own Coordinates
2. Its Own Orientation
Second Try: Pairs

Clockwise Rotation of Axes

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix} \begin{bmatrix}
x \\
y
\end{bmatrix} = O \begin{bmatrix}
x \\
y
\end{bmatrix}
\]

Second Try: Pairs

\(\varphi_i\) Measured Angle Between “i” and “j”

\(\delta_i\) Measured Distance Between “i” and “j”

\(z_j\) Reported Coordinates for Node “j”

\(z_i\) Last Estimate for Node “i” Itself

\(Z=[z_1, z_2 \ldots z_n]\) All Reported Coordinates (z, Not Included)

\(\theta_i\) Orientation Bias for Node “i”

\(r_j\) Location of Node “j” Relative to Node “i”

\(R=[r_1, r_2 \ldots r_n]\) All Locations Relative to Node “i”

\(\zeta_i\) Predicted Coordinate for Node “i”

\(Q\) Rotation Matrix for Clockwise \(\theta_i\)

\(h\) Scaling Factor (Accounts for Distortion)

\(S_i\) Cost Function for Node “i”
Second Try: Pairs

Node “i” Takes Measurements On Node “j”

\[ r_j = \begin{bmatrix} \delta_{ij} \cos \phi_{ij} \\ \delta_{ij} \sin \phi_{ij} \end{bmatrix} \]

\[ \zeta_j = z_i + h \cdot Q \cdot r_j \]

Second Try: Pairs

Cost Function: \[ S_i = \sum_j |z_j - \zeta_j|^2 \]

Matrix Form:

\[ S_i = \|Z - h \cdot Q \cdot R - z_i \cdot [1 1 1 \ldots 1 1]\|^2_F \]

Differentiate & Use Averaging to Extract:

\[ z_i \cdot [1 1 1 \ldots 1 1] = (Z - z_j) - h \cdot Q \cdot (R - r_j) \]
Second Try: Pairs

Singular Value Decomposition:

\[ U \cdot K \cdot V' = (R - r_j) \cdot (Z - z_j) \]

Find Rotation Matrix & Scaling Factor:

\[ Q = V \cdot U' \quad h = \frac{\text{trace}(K)}{\text{trace}(R \cdot R')} \]

Can Now Compute New zi

MATLAB Platform

Reasons to Use Matlab
• No Bugs from TinyOS Code
• No Errors from Antennas or Other Hardware
• Verify Algorithm Converges Correctly
• Automatically Store Data
• Investigate Modifications to Algorithm
• Perform Controlled Statistical Testing
actual coordinates

Old Algorithm

1000 iterations

New Algorithm

21 iterations

actual coordinates

Old Algorithm

1000 iterations

New Algorithm

17 iterations
Further Research

- Fixed Patterns for Pre-Localized Nodes
- How to Choose Neighbors
- Weighting Coefficients
- Interface w/ Nodes that Use Old Algorithm
- How to Isolate a Malfunctioning Node
Further Research: Reinvestigate Triplets

Why?

- More Angles Available = More Flexibility
- Use Redundant Triplets To Correct Error
- Isolate Malfunctioning Node

Further Research: Reinvestigate Triplets

Use Redundant Triplets To Correct Error
Further Research: Reinvestigate Triplets

Use Redundant Triplets To Correct Error

\[
\begin{align*}
\Phi_{513} + \Phi_{136} + \Phi_{351} &= 180 \\
\Phi_{312} + \Phi_{123} + \Phi_{231} &= 180 \\
\Phi_{534} + \Phi_{345} + \Phi_{453} &= 180
\end{align*}
\]

References

ANGLE OF ARRIVAL ANALYSIS
AND EXTENSION TO RECEIVED
SIGNAL STRENGTH SENSOR
NETWORK

Jonathan Gorzitze
3/27/08

Outline:

• Angle of Arrival Analysis
• nesC Implementation for Angle Calculation
• Simulations
  Distance Only Measurements
  Distance and Angle Measurements
• Conclusion
Get Angle Code Analysis

Get Data
Create Gain Pattern
Find Angle

Get Data

Graph showing gain pattern with degrees on the x-axis and dB on the y-axis.
Get Data

for(i_getAngle=0;i_getAngle<TOTAL_NODES;i_getAngle++) {
    if((i_getAngle < NODES_DIRECTIONAL_ANTENNA_START) || (i_getAngle > (NODES_DIRECTIONAL_ANTENNA_START + MAXNUM_DIRECTIONAL_ANTENNA -1)))
        //if((i_getAngle < 5) || (i_getAngle > 7)){
            Received1 = ((float) plintWithinGrp[0][i_getAngle])* .255;
            Received2 = ((float) plintWithinGrp[1][i_getAngle])* .255;
            Received3 = ((float) plintWithinGrp[2][i_getAngle])* .255;

Find Gain of Antenna

task void getAngle()
    uint16_t N = 36; // number of times to iterate between points in gain arrays
    uint16_t inc = 2; // degree incrementation size
    float sqr_error, least_sqr_error = 260000.0; //largest possible error
    float Gain1_meas[] = {-66, -59.135, -49.465, -44.145, -40.985, -39.95, -36.945, -35.8175, -35.13, -34.88, ...
    float Gain2_meas[] = {-36.945, -39.95, -40.985, -44.145, -49.465, -59.135, -66, ...
    float Gain3_meas[] = {-53, -52, -51, -50, -51, -52, -53, -54, -55, ...


Find Gain of Antenna

//create array for each slope point
for (k_getAngle = 0; k_getAngle < N; k_getAngle++) {
    slope1[k_getAngle] = (Gain1_meas[k_getAngle+1] - Gain1_meas[k_getAngle])/(360/N);
    slope2[k_getAngle] = (Gain2_meas[k_getAngle+1] - Gain2_meas[k_getAngle])/(360/N);
    slope3[k_getAngle] = (Gain3_meas[k_getAngle+1] - Gain3_meas[k_getAngle])/(360/N);
}

for (z_getAngle = 0; z_getAngle <= 358; z_getAngle+=2) {
    angle_by_ten = z_getAngle / 10;
    rem_by_ten = z_getAngle % 10; //remainder term

    //Calculate the Gain value at every two degrees
    Gain1 = Gain1_meas[angle_by_ten] + slope1[angle_by_ten]*rem_by_ten;
    Gain2 = Gain2_meas[angle_by_ten] + slope2[angle_by_ten]*rem_by_ten;
    Gain3 = Gain3_meas[angle_by_ten] + slope3[angle_by_ten]*rem_by_ten;

    Received Signal Strength = Gain Pattern ( ) – Received Signal ( )
    Difference = Received Signal Strength(1) – Received Signal Strength(2) ~ 0
    Squared Error = Difference(12)^2 + Difference(23)^2 ~ 0
Find Angle

\[ \text{RSS}_1 = -\text{Gain}_1 + \text{Received}_1; \]
\[ \text{RSS}_2 = -\text{Gain}_2 + \text{Received}_2; \]
\[ \text{RSS}_3 = -\text{Gain}_3 + \text{Received}_3; \]
\[ \text{diff}_{12} = \text{RSS}_1 - \text{RSS}_2; \]
\[ \text{diff}_{23} = \text{RSS}_2 - \text{RSS}_3; \]
\[ \text{diff}_{31} = \text{RSS}_3 - \text{RSS}_1; \]
\[ \text{sqr}_\text{error} = (\text{diff}_{12} \times \text{diff}_{12}) + (\text{diff}_{23} \times \text{diff}_{23}); \]
\[ \text{if}(\text{sqr}_\text{error} < \text{least}_\text{sqr}_\text{error}) \}
\[ \quad \text{least}_\text{sqr}_\text{error} = \text{sqr}_\text{error}; \]
\[ \quad \text{low}_\text{index} = \text{z}_\text{getAngle}; \]
\[ \}\]
\[ \text{AngleToEachNode}[\text{i}_\text{getAngle}] = (\text{float}) \text{low}_\text{index} \times \text{inc}; \]

Simulations

Setup
Distance Only Algorithm
Angle and Distance Algorithm
Simulation Setup:

Implemented testing for:
- Accuracy of self-localization
- Time to self-localization

Introduced three error terms:
- Angle Error
- Distance Error
- Noise term (Jitter)

Root Mean Squared = $\sqrt{\frac{\sum_{n} (x_{\text{actual}} - x_{\text{est}})^2 + (y_{\text{actual}} - y_{\text{est}})^2}{\text{total motes}}}$
Self-Localization

No Angle Error, No Distance Error, No Noise error

Accuracy

Iterations Until RMS <1m

RMS Error in dB

Distance Algorithm

Angle and Distance Algorithm

RMS Error in dB

Distance Algorithm

Angle and Distance Algorithm

Iterations Until RMS <1m

99% Succeed

100% Succeed

Self-Localization

5% Distance Error, No Angle Error, No Noise error

Accuracy

Iterations Until RMS <1m

RMS Error in dB

Distance Algorithm

Angle and Distance Algorithm

RMS Error in dB

Distance Algorithm

Angle and Distance Algorithm

Iterations Until RMS <1m

90% Succeed

99% Succeed
Self-Localization

No Angle Error, No Distance Error, 5% Noise error

Accuracy

Iterations Until RMS <1m

Self-Localization

5° Angle Error, 5% Distance Error, 5% Noise error

Accuracy

Iterations Until RMS <1m
Self-Localization

How much angle error can be tolerated until angle measurements no longer reflect higher accuracy over distance only measurements?

8° Angle Error

Conclusion

• In the absence of noise terms, the Angle and Distance algorithm outperforms the Distance Only algorithm by converging approximately five times quicker and approximately one order of magnitude higher accuracy (after 500 iterations).

• When Distance Errors and noise errors are introduced, the Angle and Distance algorithm outperforms the Distance Only algorithm by converging approximately four times quicker and 3-4 times more accurately.

• The Angle and Distance algorithm outperforms the Distance Only algorithm until an angle error of 8° or greater is measured.
Questions:

MICA2 AND TinyOS

IMPLEMENTATION AND APPLICATION

Luan Nguyen
3/27/08
Table of Contents

I. Mica2
   a. Overview
   b. Design
   c. MIB510

II. TinyOS
   I. Overview
   II. Cygwin

III. Application

Mica2
   a. Overview
   b. Components
   c. MIB510
Mica2

Overview
- Third generation MPR (Mote Processor Radio)

Usage

Mica2

Design
- Data Flash Logger
- Microcontroller
- Radio
- Expansion Connector
- Antenna
Mica2

- Advantages
  - Small, compact
  - Power efficiency
  - Economical
- Disadvantages
  - Lack of User Interface for diagnosing and debugging
  - Interaction between programming software and motes

MIB510

- Mote Interface Board
  - On board ISP (In System Processor)
  - Port RS 232 to USB port
  - Baud rate of 115.2 kbaud
  - 5 leds: power, isp and 3 MPR leds
  - Programming interface for the motes
TinyOS

Overview

- Embedded Operating System
- Written in NesC
- Created by UC Berkley and Intel
NesC
- Embedded Operating System
- Written in NesC
- Overall similar with C
- More specified declaration of variables
  - NesC
  - C
    - Uint16_t
    - integer
    - Uint8_t
- Limit the memory allocated for each variable

CygWin
- Provide Linux Inside Windows
- NesC is compiled inside Linux environment
- Similar to DOS environment
CREATE A WIRELESS SENSOR NETWORK
Common Problems when working with Mica2 and TinyOS
Memory Issue

- Create an array for look-up table with estimated angle

- Use array to calculate and confirm angle

- Program crash because there is not enough memory

- Solution: Use for loop to calculate each element of the array
Not Responding Motes

Compiled but Where is the data?
Applications

- Controlled Environment Agriculture Monitor
  - Sense Temperature and Moisture
  - Minimize care and labor
  - Minimize water required (reduced by 50%)
  - Sensor network ties to automation system make CEA systems Economical
Applications

- Controlled Environment Agriculture Monitor

Applications

- Military – Snipers Detection
  - PinPtr Application
  - Multi acoustic sensors network measures the muzzle blasts and shock wave
  - Determine the location of the shooters base on speed of sound and sensors location
Applications

- Military – Snipers Detection

Applications

- Firefighters
  - Sensors with hear rate monitor
  - Determine the exact location of firefighters even in high rise buildings
Applications

- Firefighters
  - Monitor the condition of firefighters
  - Determine the locations of firefighters not only from a bird view (GPS) but also in 3D environment

Conclusion

- Lessons Learned
  - Experiences from working with new and different technologies
  - Planning and Executing Project effectively
  - Communications in Teamwork
References


Final Questions