On Seamless Wireless Sensor Deployment and System Implementation

Kuan-Hsin Peng, Ting-Yu Lin, and Kuan-Hua Chen
Department of Communication Engineering
National Chiao-Tung University
a9102003@mail.ntust.edu.tw, tylin@csie.nctu.edu.tw, and pevidpevid@gmail.com

摘要

無線感測網路的覆蓋率一直都是無線感測網路中一個重要議題；在大部份的研究中，通常都假設感測器的感測範圍相同，且為不具有移動能力的靜態節點，但在實際環境中，感測器的感測範圍並不會完全相同；此外，當感測器電力不足或因外在環境造成損壞，使用者都必須付出人力對這些故障的感測器進行汰換或維修的工作。

有鑑於此，本系統便是針對感測範圍大小不同且具有移動能力的感測器作討論，透過本論文中提出的 CADT 演算法，來解決感測器失效或者整體網路感測器佈置不均勻時，所造成整體覆蓋率下降的問題；本論文中提出的 CADT 演算法中，計算出每個感測器最佳的位移量及移動方向，重新佈置整個網路，提升整個網路的覆蓋率，進而達到無縫隙的無線感測網路。

關鍵詞：CADT、sensing coverage、seamless

1. 前言

隨著微機電技術的進步與無線傳輸技術之發展，無線感測網路的應用範圍也越來越廣泛；在無線感測網路的架構下，感測器本身就像一台小型電腦，具備簡單的感測、運算、無線傳輸等功能，可以將環境中我們所感興趣的環境參數(如溫度、溼度等)傳送給主控端，讓位於主控端的使用者能確切掌握環境的狀態；而為了能達到這個目的，感測器的佈置對於整個感測網路是非常重要的。

在過去的研究中[1],[2]，都假設在環境已知下，將感測器佈置在整個監控環境中，但在大部份情況下，環境是未知且危險的區域，例如：太空、敵軍戰區、地震地帶…等等，在這種情況下，感測器的佈置就相對的困難，所以在大部份的情況下，透過飛機來將感測器灑至在監控區域中是一個解決辦法，但卻可能因為風力或者建築物的關係，而無法掌握感測器降落的位置，則網路必然無法達到我們所需的覆蓋率。

此外，在傳統感測網路中，皆為固定式的感測器，不具有移動的能力，往往因為電池的消耗或者其它外在環境因素 (颱風、人力等)的侵蝕與損毀，造成感測器失效的情況，進而導致網路感測上的死角及資訊傳遞上的阻斷，所以為了避免這種情況發生，無線感測網路會佈置大量的感測器在網路中[3]，透過排程方式來修補感測上的死角，但這種方法事實上治標卻不治本；綜合以上，所以我們便利用移動式的感測器移動到正確的位置來達到所需的覆蓋率。

本論文所提出的 CADT (Coverage Algorithm based on Delaunay Triangulation) 演算法可以針對整體覆蓋率下降的問題，透過可移動式的感測器自動重新佈置(Auto-configuration)的網路架構，來填補監控網路上的感測漏洞，來提升整體網路覆蓋率。CADT 包含三個步驟:

Step 1 將整個監控環境透過 Delaunay Triangulation 的幾何概念，切割成許多大小不一的三角形。

Step 2 CADT-HD (CADT-Hole Detection)，透過幾何的運算，找出每個三角形沒有被感測器覆蓋的區域。

Step 3 CADT-HR (CADT-Hole Removal)，針對每個三角形沒有被覆蓋的區域，找出與鄰近感測器重疊範圍較大且分布較密集區域中的感測器進行移動，來填補三角形中沒有被覆蓋的區域。

重覆以上步驟，直到整體覆蓋率達到所設定的覆蓋率門檻值(Coverage Threshold) 才終止，而根據最終模擬出來的結果，本論文所提出的演算法的確可以有效率達到覆蓋率的要求。

2. CADT 演算法

CADT 演算法包含三個步驟：

第一個步驟，我們根據感測器的佈置的位置透過 Delaunay Triangulation 將整個監控區域分割成許多大小不一的三角形。

第二個步驟，執行 CADT-HD (CADT-Hole Detection)演算法，利用幾何的運算找出每個三角形沒有被覆蓋到的區域。

第三個步驟，執行 CADT-HR (CADT-Hole Removal)演算法，針對每個三角形中沒有被覆蓋的區域，找出與鄰近感測器重疊區域最大且鄰近密集高的感測器來移動，來填補其沒有被覆蓋的區域。

2.1. 步驟 I : Divide Area

Delaunay Triangulation[4] 是在幾何運算非常重要的資料結構；它是在一種在已部署節點的空間(或平面)中，依照各節點的位置劃分出三角形之圖式方法。

如圖 1，我們根據感測器的佈置的位置透過 Delaunay Triangulation 將整個監控區域分割成許多大小不同的三角形，感測器則皆佈置在各三角形的頂點；我們擺放幾個邊界感測器來設立我們監控的區域(不具有移動能力)[S10 ~ S17]。

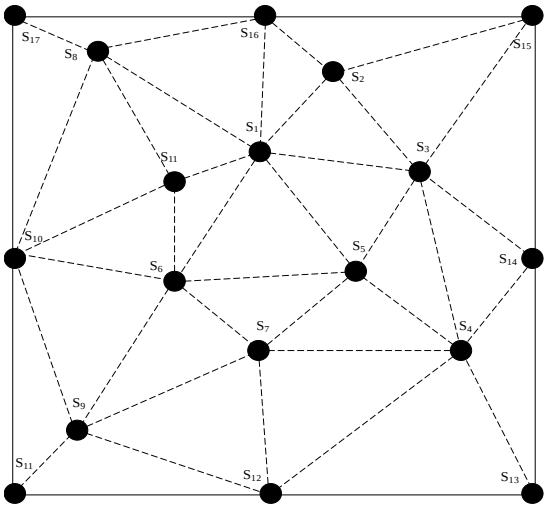


圖 1 Delaunay Triangulation 監控區域切割圖

2.2. 步驟 II : CADT-HD

由於每個感測器的感測範圍大小不盡相同，如圖 2，每個三角形中，不只有部署在三角形頂點上的感測器具有覆蓋的此三角形區域的能力；鄰近三角形的感測器也有機會會覆蓋到此三角形內的區域，所以我們便針對三角形頂點上的感測器[S1、S5、S6]，及鄰近三角形的感測器[S3、S7、S11]，探討其覆蓋的程度。當然，更遠三角形的感測器也有機會會覆蓋到此三角形內的區域[S2、S4、S8、S9、S10]，我們的演算法也可以擴充到更多的感測器，但相對的計算量也會增加；所以在此我們只討論此六個感測器的感測範圍。

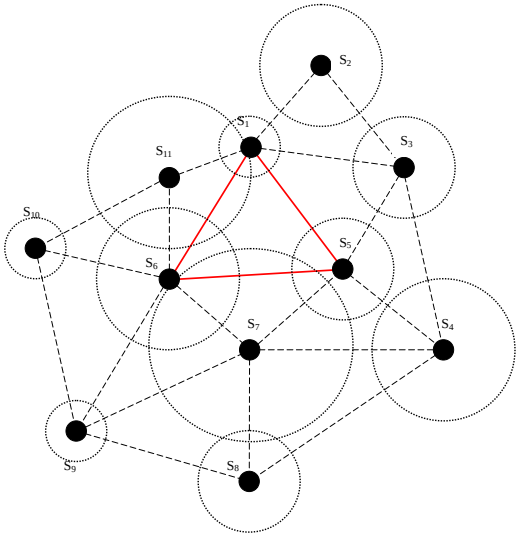


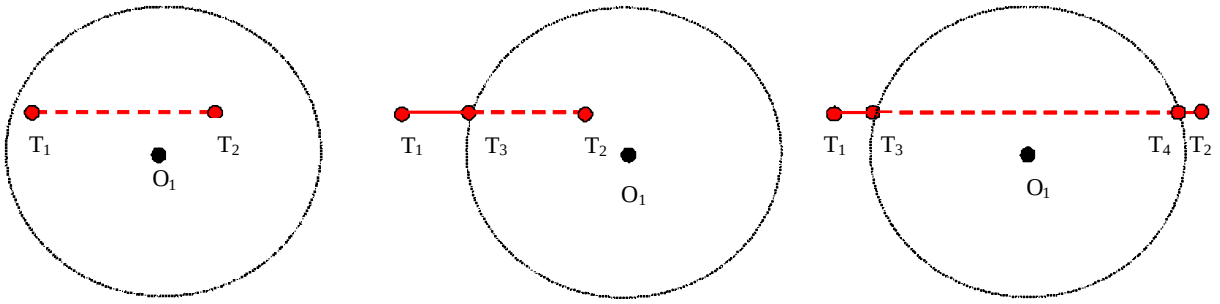
圖 2 感測器選取示意圖

2.3.1 基本概念

在幾何概念中，一個線條是由任意不同兩端點的連線所產生出來的，線條上的任一點必在兩端點之間；當任意一個圓形同時包含此兩端點，則此圓形必定將整個線條包圍住且不會和此線條產生任何交點(如圖 3 (a))；而當任意一個圓形包含任何一端點時，則圓形必包圍線條上的一部份且和此線條產生一個交點，而此交點必為新線條的端點(如圖 3 (b))；而當任意一個圓形不包含任何一端點，但卻有包含此線條上的點，則此圓形必和此線條產生兩個交點，且將此線條切分成兩線條，而此產生的兩交點各為兩新線條的端點(如圖 3 (c))。

2.3.2 CADT-HD 演算法

我們針對每一個 Delaunay Triangulation 所切割出來的三角形，一一加入感測器覆蓋範圍，然後透過上述的基本概念，來找出其三角形內尚未被覆蓋區域的交點。如圖 4 (a)初始狀態下，尚未加入任何感測器的感測範圍，其 Hole 的交點為三角形的頂點。如圖 4(b)加入 N1 的感測範圍，掃描所有 Hole 的交點(H1 ~ H3)，找出有被 N1 感測範圍包圍的交點，然後刪掉此交點(H1)。將被刪除交點的交叉線列出來(L12、L13)，如果沒同時包含交叉線的兩端點，N1 感測範圍必產生交點在交叉線上。依序加入 N2~N6，可得圖 4(c)。

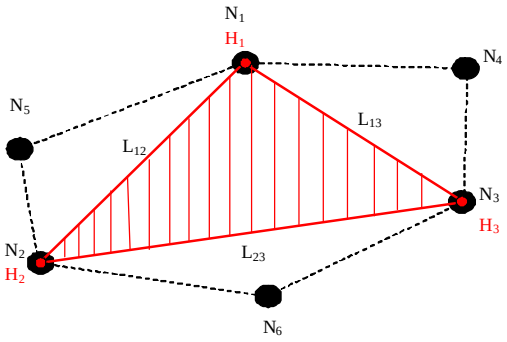


(a) 圓形(O_1)同時包含此兩端點(T_1、T_2)不會產生任何交點

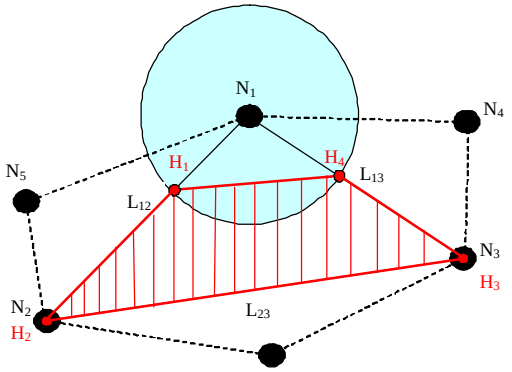
(b) 圓形(O_1)同時包含一端點(T_2)產生新的交點(T_3)

(c) 圓形(O_1) 不包含任何一端點，卻有有包含此線條上的點，則產生兩個新交點(T_3、T_4)，各為新兩線段之端點。

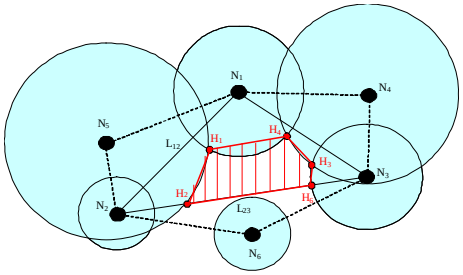
圖 3 幾何基本概念



(a) 初始狀態，尚未加入任何感測器感測範圍



(b)加入 N1 感測範圍



(c)加入 N1~N6 感測範圍

圖 4 CADT-HD 演算法表示圖

我們再來看一個例子；如圖 4(a)初始狀態下，尚未加入任何感測器的感測範圍，其 Hole 的交點為三角形的頂點。如圖 5 加入 N1 的感測範圍，掃描所有 Hole 的交點(H1 ~ H3)，找出有被 N1 感測範圍包圍的交點，然後刪掉此交點(H1)。將被刪除交點的交叉線列出來，如果沒同時包含交叉線的兩端點，N1 感測範圍必產生交點在交叉線上。

根據演算法，我們可以透過 N1 的交叉線訊息，推導出兩個新的交點(H1、H4)，但在實際的情況下，我們發現 N1 的感測範圍，也產生兩個新的交點在 L23 的線段上，並將整個尚未覆蓋的區域切分成兩塊，這個結果跟我們的所推導出來的結果有很大的誤差。

這是由於在我們的演算法中，是透過取得交點上的交叉線資料，來檢查是否包含線段或

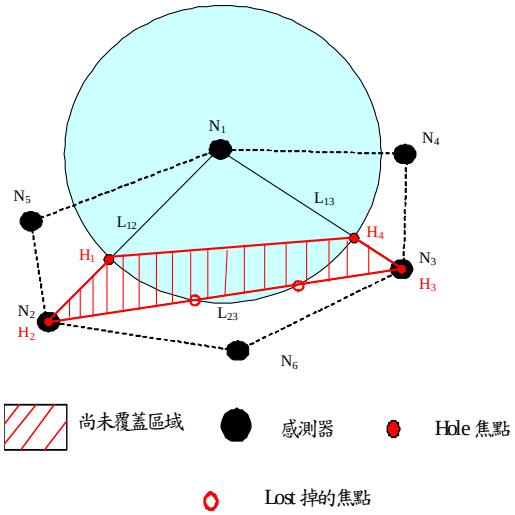


圖 5 加入 N1 感測範圍(判斷錯誤)

圓弧的兩端點，來決定是否產生交點；但 N1 的感測範圍在 L23 的線段上抓不到任何交點，所以並沒有辦法在 L23 的線段上產生新交點，而產生判斷錯誤。

2.3.3 Modified CADT-HD 演算法

由於因為沒辦法取得線段上或圓弧上的交點，而造成演算上判斷錯誤；所以我們便在線段和圓弧上加上輔助點(交點位置也需擺放兩交叉線的輔助點)，每個輔助點，記錄所在的線段及所在的位置(如圖 6)；透過輔助點的加入我們可以準確的在線段或圓弧上產生新的交點，甚至也可清楚的描述整個破洞的形狀及大小。

我們透過計算交叉線的組數，來判斷圓與線段發生交點的情形；由於每個線段與圓弧長度不同，則輔助點數量也會不大相同，所以我們將所包含到相同線段或圓弧輔助點(不論多少點)都視為兩組交叉線。

當任意一個圓形同時包含線段兩端點及線段上所有的輔助點(共有四組交叉線)，則此圓形必定將整個線條包圍住且不會和此線條產生任何交點；而當任意一個圓形包含線段任何一端點及部份輔助點(共有三組交叉線)，則圓形必包圍線條上的一部份且和此線條產生一個交點，而此交點必為新線條的端點；而當任意一個圓形不包含線段任何一端點，但卻有包含此線條上的輔助點(共有二組交叉線)，則此圓形必和此線條產生兩個交點，且將此線條切分成兩線條，而此兩交點各為兩新線條的端點。

2.4 步驟 III : CADT-HR

透過 CADT-HD 的演算法，可以清楚的描述未覆蓋區域(淡紅色區域)的形狀及大小以及位置；所以便可針對三角形沒有被覆蓋的區域(面積由大到小排列)，利用我們的 CADT-HR 演算法來找出與鄰近感測器重疊範圍較大且分布較密集區域中的感測器作移動，來填補三角形中沒有被覆蓋的區域。

2.4.1 CADT-HR 演算法

在 CADT-HR 中，為了避免感測器移動太

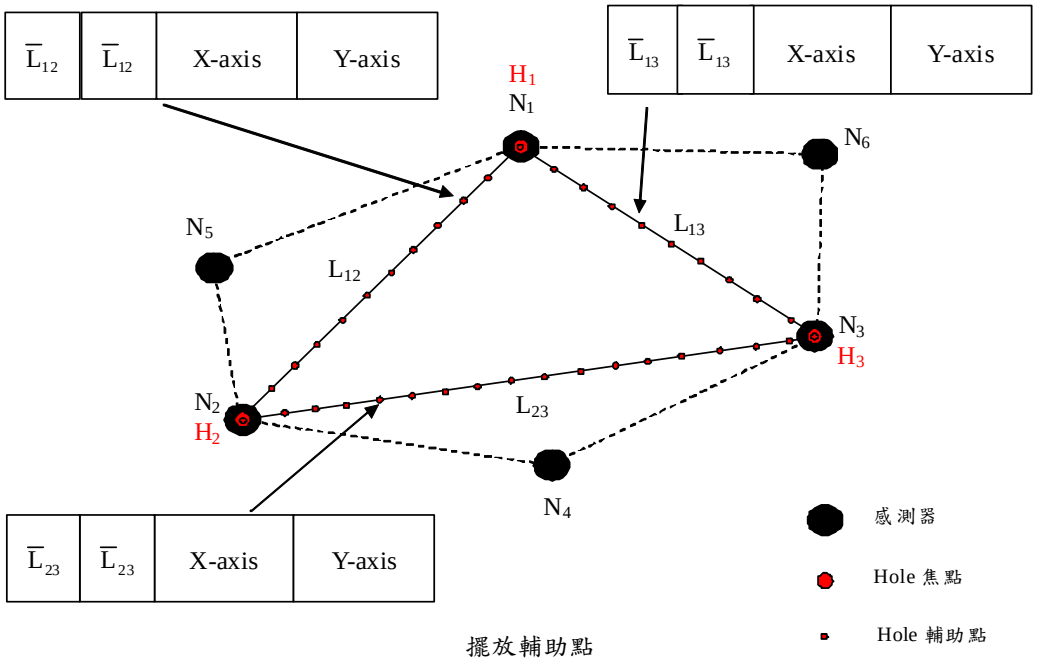


圖 6 Modified CADT-HD 演算法表示圖(加入輔助點)

遠造成電池的耗損，我們只針對造成三角形沒有覆蓋區域旁的感測器，然後再從這些感測器中挑選一個重疊範圍較大且分布較密集區域中的感測器朝向最遠的 Hole 交點作移動，來填補三角形中沒有被覆蓋的區域。

在圖 7 中，透過之前 CADT-HD 演算法找出 [H1~H6] 的 Hole 交點及交叉線；在 CADT-HR 演算法中，我們便可以透過 Hole 交點的交叉線，找出造成三角形 Hole 的感測器 [N1、N3、N4、N5、N6]及所對應最遠的 Hole 交點。我們再從五個感測器中，挑選一個重疊範圍較大(Local_Weight)且分布較密集區域中(Global_Weight)的感測器來做移動，其方程式定義如下:

$$W = (1-\beta)\times W_L + \beta \times W_G$$

透過此方程式，我們挑選最大的 W 來做移動，透過 β 值來調整其權限，當 Hole 較大時，便需要區域較密集的感測器才可以覆蓋，則 β 值需較小來增加 Global_Weight 較大的感測器做移動的機會；當 Hole 較小時，只需要鄰近感測器重疊區域較大(Local_Weight 大)的感測器就可以覆蓋，則 β 值需較大來增加 Local_Weight 較大的感測器做移動的機會；為了避免感測器移動產生來回移動振盪現象，所以在我們演算法中，感測器每個 Round 只能移動一次。

$$\beta = \alpha \times \frac{Hole\_Size}{Maximum\_Hole\_Size}$$

感測器	N1	N3	N4	N5	N6
最遠 Hole 交點	H2	H6	H2	H6	H4

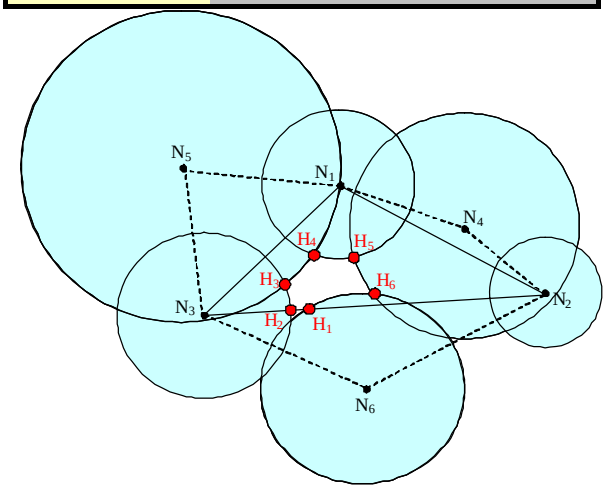


圖 7 感測器與 Hole 交點表示圖

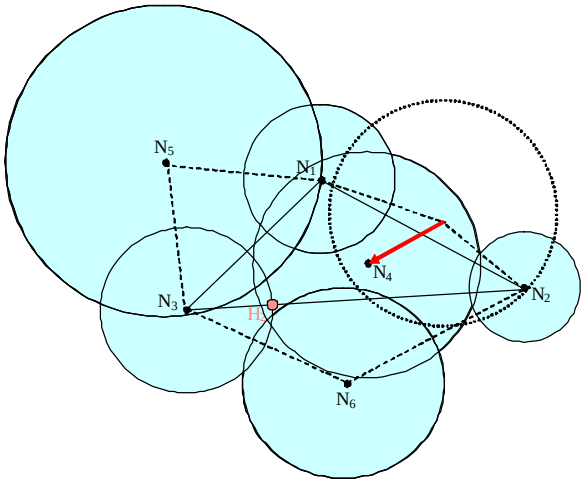


圖 8 感測器移動示意圖

圖 8 中，假設 N4 的 Total_Weight 最大，則往最遠的 Hole 交點(H2)方向做移動直到覆蓋 H2 的交點，來填補三角形未覆蓋的區域。

2.4.2 重疊區域

在過去大部份的研究中，都是假設每個感測器具有相同的感測範圍 R，所以在量測兩個感測範圍重疊的區域時，都是利用重疊區域的寬度(2R - 兩圓心距離)來區別重疊區域的大小，但在本論文中，我們針對的感測器具有不同大小的感測範圍，便無法單純的使用重疊區域的寬度來表示，只能真正計算重疊區域的面積來區別重疊區域的大小，但這計算過程是相當的複雜且煩瑣，所以我們便用一個粗糙的方式來區別重疊區域的大小。

圖 9 中，在具有不同大小感測範圍的感測器下，重疊區域的寬度 $\overline{W}_{12}(\overline{d}_{12}-\overline{r}_1-\overline{r}_2)$ 無法清楚表示重疊區域的大小，畢竟在相同重疊區域的寬度 $\overline{W}_{12}$ 下，隨著感測範圍大小不同，則重疊區域大小也隨之不同；所以我們將感測範圍的變化也表示在重疊區域大小的變化上，透過下面的表示式

$$\text{Overlap Ratio } O_{21} = \frac{\overline{W}_{12}\times\overline{r}_1\times\overline{r}_2}{\pi\times\overline{r}_1^2}$$

(For Circle C_1)

$$\text{Overlap Ratio } O_{12} = \frac{\overline{W}_{12}\times\overline{r}_1\times\overline{r}_2}{\pi\times\overline{r}_2^2}$$

(For Circle C_2)

來表示重疊區域對各個圓所占的比例大小。

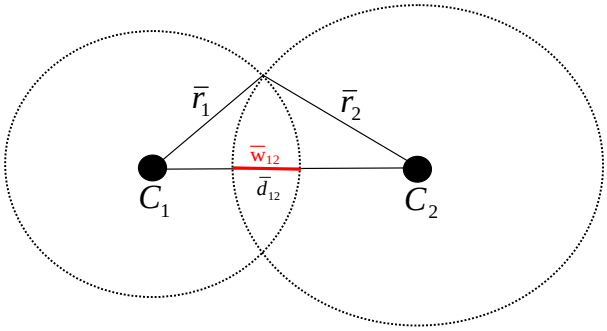


圖 9 重疊區域表示圖

2.4.3 Local_Weight 求法

Local_Weight 是探討感測器本身能支援覆蓋的能力；我們將感測器與鄰近感測器(延伸三角形的感測器) Overlap Ratio 大小全部加總在一起，來表示感測器本身填補未覆蓋區域的能力。

如圖 10 中，S1 鄰近的感測器為[S2、S3、S4、S5、S6]，由於 $\bar{W}_{13}$、$\bar{W}_{14}$、$\bar{W}_{15}$、$\bar{W}_{16}$ 大於 0，所以 O_{31}、O_{41}、O_{51}、O_{61} 也皆為正數，代表有重覆的區域；而 $\bar{W}_{12}$ 小於 0，所以 O_{21} 為負數，代表兩圓沒有重覆的區域，則 Local_Weight $O_1 = O_{21} + O_{31} + O_{41} + O_{51} + O_{61}$ 。

$$O_i = \sum_{j \in Neighbor\ i} O_{ji}$$

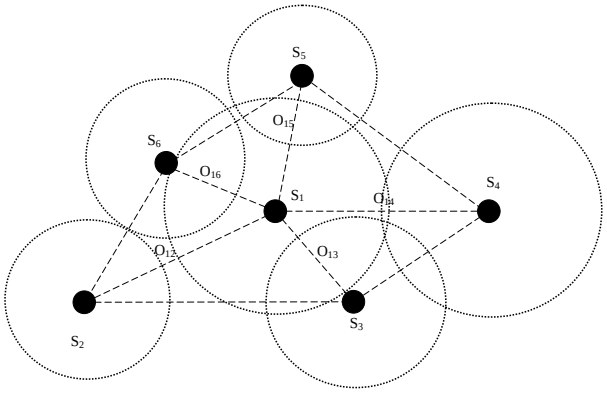


圖 10 Local_Weight 示意圖

2.4.4 Global_Weight 求法

Global_Weight 是探討感測器附近區域能支援感測器覆蓋的能力;以感測器為起點，往外擴張兩階層，將兩階層內所有感測器的 Local_Weight 大小加總在一起，除以兩階層內感測器的數量，但不考慮不具有移動能力的邊界感測器及所針對的三角形上的感測器：圖 11 中，針對[S1、S6、S11]三角形中，探討 S1 感測器的 Global_Weight，將第一階層[S2、S3、S5、S8](淡藍色)及第二階層[S4、S7](淡綠色)的 Local_Weight 大小加總在一起除以兩階層內感測器的數量就為 S1 感測器的 Global_Weight；Global_Weight 越大代表感測器

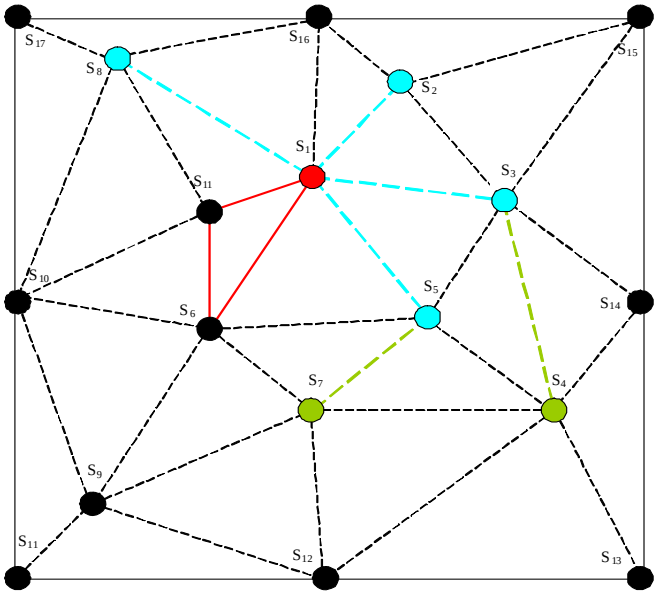


圖 11 Global_Weight 示意圖

附近重疊的區域也越大，也越有本錢可以彌補感測器移動後所造成的 Hole。

3. CADT 演算法模擬

我們針對 CADT 演算法與 VFA 演算法來做效能上的評估；我們在 3.1 節，針對 VFA 有概略的簡介；在 3.2 節我們在針對 Sensing Range 在不同機率分佈下的效能評估。

3.1 VFA (Virtual Force Algorithm)

在整個感測網路中，每個感測器都會對另一個感測器產生一個力，而這個力可能是吸引力也可能是排斥力，當兩個感測器距離太過接近時，他們則會產生排斥力；假如兩個感測器距離太過遙遠時，他們則會產生吸引力。

假設第 i 個感測器位在 (X_i, Y_j) 位置上，

則第 i 個感測器與第 j 個感測器的距離

$$d_{ij}=\sqrt{\left(X_i-X_j\right)^2+(Y_i-Y_j)^2}$$

Force $\vec{F}_{ij}$ 則代表第 i 個感測器施加在第 j 個感測器上的力，$\vec{F}_{ij}$ 在極座標定義如下：

$$\begin{cases}\vec{F}_{ij}=w_a\left(d_{ij}-d_{th}\right),\alpha_{ij} & \text{if } d_{ij}>d_{th}\\ \vec{F}_{ij}=0 & \text{if } d_{ij}=d_{th}\\ \vec{F}_{ij}=w_r\left(d_{th}-d_{ij}\right),\alpha_{ij}+\pi & \text{if } d_{ij}<d_{th}\end{cases}$$

$$d_{th}=\alpha\left(SR_i+SR_j\right)$$

SR_i 為第 i 個感測器的感測範圍

d_{th} 代表第 i 個感測器與第 j 個感測器之間距離的 threshold，為一個非常重要的參數；當 d_{th} 設定過小時，則感測器之間的覆蓋區域會變大；相反的，當 d_{th} 設定過大時，則感測器之間的會產生較大的 Hole；w_a、w_r 也是一個非常重要的參數，用來設定吸引力與排斥力的量，大部份 $w_a<<w_r$。

在圖 12 中，考慮四個感測範圍大小不同的感測器，總共對 S_1 作用力為 $\vec{F}_1=\vec{F}_{12}+\vec{F}_{13}+\vec{F}_{14}$；由於 $d_{12}>d_{th}=(r_1+r_2)$ 所以 S_2 給予 S_1 是吸引力；$d_{14}<d_{th}=(r_1+r_4)$ 所以 S_4 給予 S_1 是排斥力；$d_{13}=d_{th}=(r_1+r_3)$ 所以 S_3 不給予 S_1 作用力。

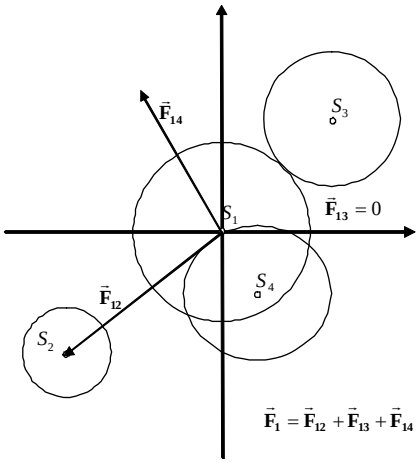


圖 12 Virtual Force 在不同感測範圍的感測器情況下 $(\alpha=1)$

假設整個監控環境，共有 k 個感測器；對第 i 個感測器，則合力定義為：

$$\vec{F}_i=\sum_{j=1,\ j\neq i}^{k}\vec{F}_{ij}$$

由於透過 VFA 將整個感測網路均勻分佈，但感測器卻很有可能移動超過我們所監控的範圍，所以我們透過設定 Boundary Force 來避免此情況發生。

為了避免感測器移動超出我們所監控的範圍，如圖 13，我們佈滿具有相同感測範圍的感測器於邊界上(邊界感測器)，用來設立監控區域的範圍，當感測器很靠近邊界時，我們會給予對邊界極大的排斥力，讓感測器往中間移動。

考慮 Boundary Force 情況下，合力定義為:

$$\begin{cases}\vec{F}_i=1,\pi & \text{if } x_i+SR_i>m-d\\ \vec{F}_i=1,0 & \text{if } x_i+SR_i<d\\ \vec{F}_i=1,\dfrac{3\pi}{2} & \text{if } y_i+SR_i>n-d\\ \vec{F}_i=1,\dfrac{\pi}{2} & \text{if } y_i+SR_i<d\\ \vec{F}_i=\sum_{j=1,j\neq i}^{k}\vec{F}_{ij} & otherwise\end{cases}$$

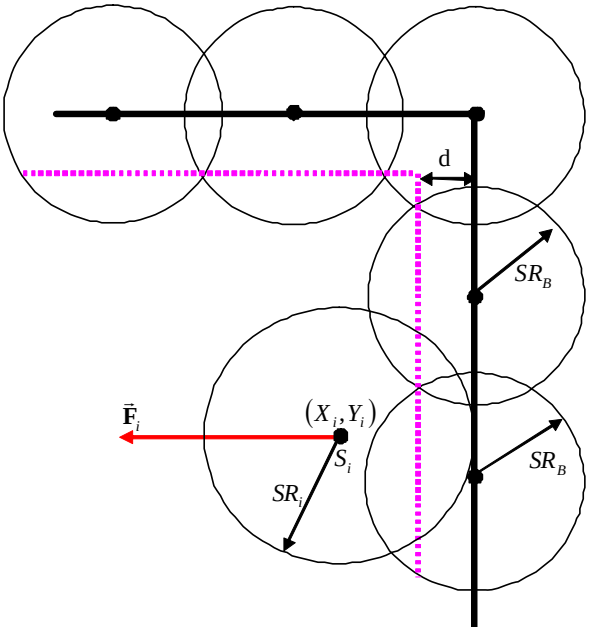


圖 13 考慮 Boundary Force 情況下

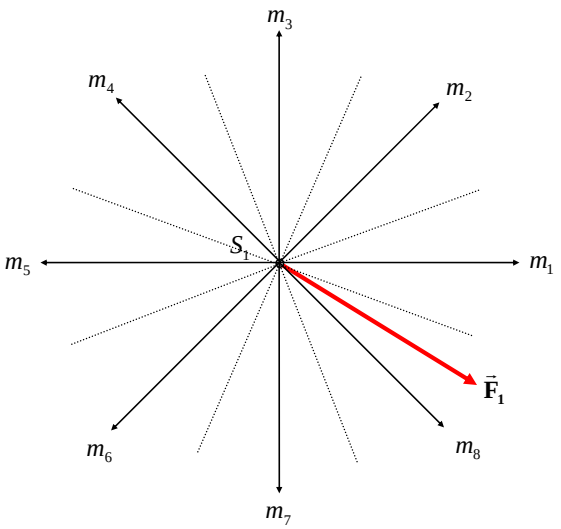


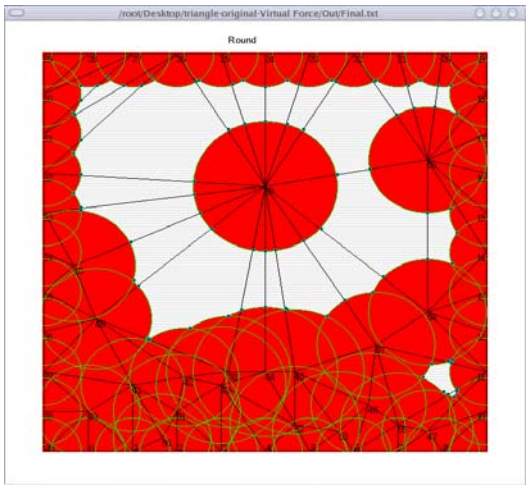
圖 14 感測器移動方向

我們將整個感測器移動區分成八個方向，根據合力所在的區域，決定移動的方向，然後朝著那個方向一次移動一個 step；圖 14 中，$\vec{F}_1$的合力落在m_8的區域中，則S_1則往m_8方向移動一個 step。

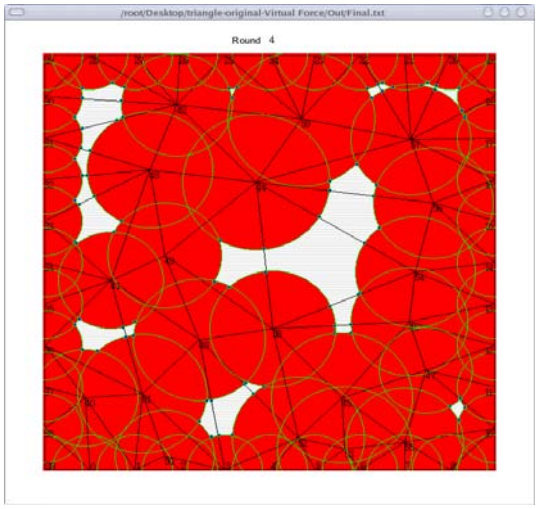
3.2 Simulation Result

表 1 模擬參數表

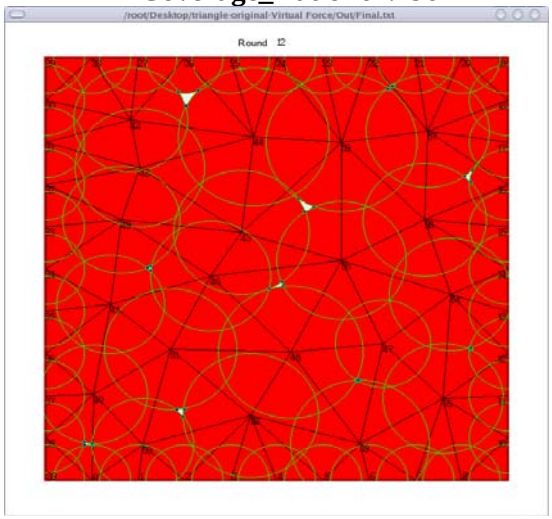
監控範圍	300*300
邊界感測器數量	40
邊界感測器感測範圍	25
移動感測器感測範圍	Normal(45,25)
移動式感測器數量	20
α	0.75
w_a	2
w_r	2000



(a) Initial Deployment
Coverage_Ratio= 51.9082



(b) Round 4
Coverage_Ratio=67.1961



(c) Round 12
Coverage_Ratio=99.5111

圖 15 CADT 演算法執行結果

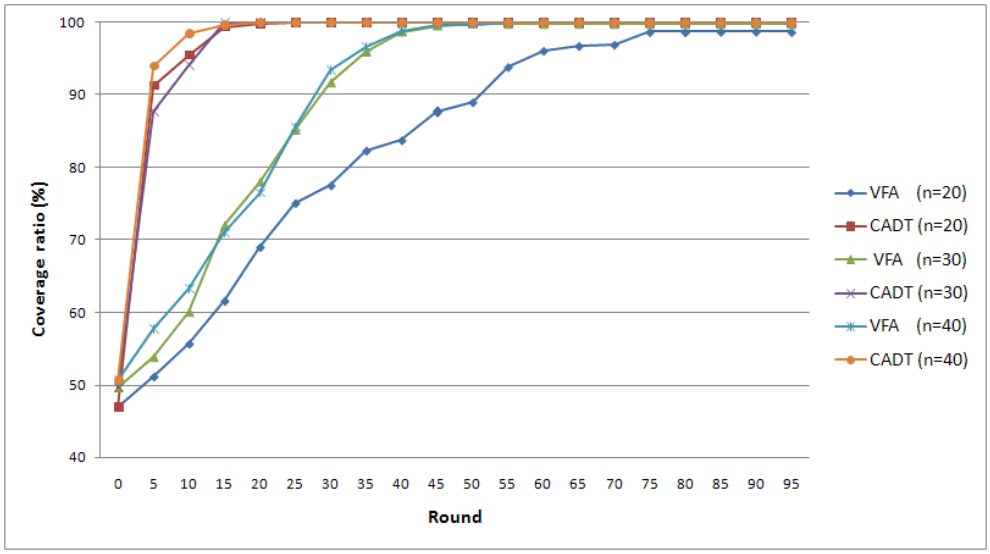


圖 16 CADT/VFA 效能比較

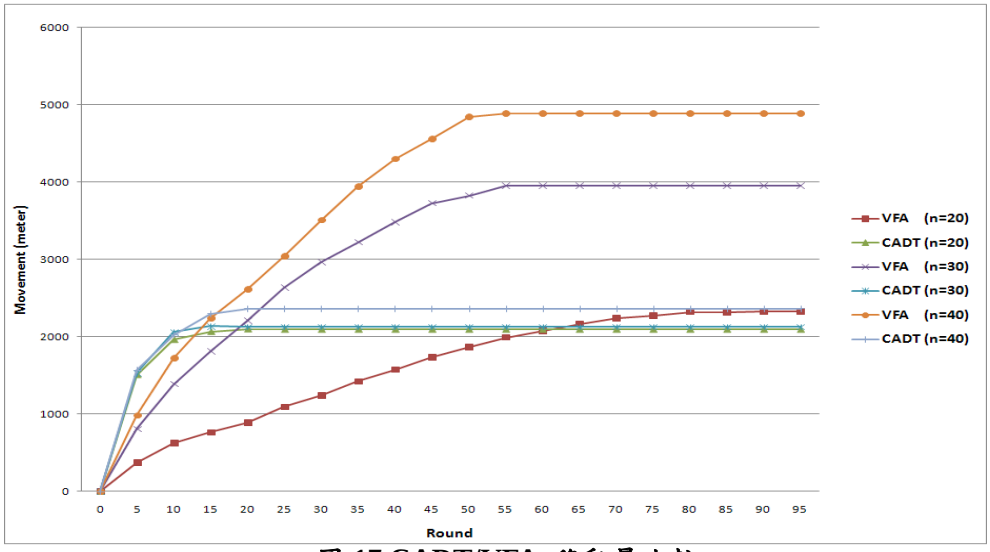


圖 17 CADT/VFA 移動量比較

圖 15 中，隨著 CADT 演算法執行次數增加，其 Coverage Ratio 也逐漸的提升；最後可以在圖 15(c)中發現在 CADT 演算法在第 12 次時，整個網路已趨近於無縫隙的感測網路；另外，圖 15 中的藍點則是透過 CADT-HD 所產生出來的焦點，證明了我們 CADT-HD 的確可以準確的抓出 Hole 焦點的所在。

圖 16 中， 我們比較 CADT 及 VFA 演算法在不同感測器數量下的效能分析；CADT 演算法，在感測器數量較少情況下，雖然 Round 次數有增加，但其覆蓋率卻仍可趨近於 Seamless Coverage；而 VFA 經過了 70 幾個 Round，到達大概 97%的水準；這是因為 CADT 是針對未被覆蓋的區域做有效率的移動，而 VFA 則是讓每個感測器能平均分散在整個區域中，而不是針對 Hole 去做填補，所以需要較多的感測器來達到 Seamless Coverage。圖 17 中，發現 VFA 演算法隨著移動感測器數量的增加，感測網路的整體移動量也隨之大幅度增加；這是由於 VFA 演算法中的感測器都是針對網路中所有感測器的所產生出來的合力做移動，即時感測器鄰近沒有 Hole，在網路還沒有達到一種平衡的狀態，它也會根據合力做移動，而且感測器數量越多越難達到網路的平衡，所需移動的量也越多；而相對的，在 CADT 演算法中，我們則是只針對 Hole 鄰近的感測器來做填補，則也是 CADT 演算法在感測器數量增加時，其移動量也不會大幅度的增加，這對於電池供電的感測器，能大大的降低電池的消耗；在 Node 數量較小時，CADT 演算移動了所有感測器來填補未覆蓋區域，所以跟 VFA 的移動量差不多。

由圖 16，圖 17 我們得知 VFA 演算法隨著感測器數量的增加，覆蓋率可以相對的提

高，但對於整體移動量也會大幅的提高；而在我們的 CADT 演算法中，透過增加 Node 數量，不但提高整體的覆蓋率，也並不會大幅增加整體的移動量。

4. 感測網路發展工具相關知識

本論文所使用的移動式感測器是採用美國 Crossbow 公司所研發的無線感測網路發展工具[6]結合由嵌入式系統所控制的載具；在感測器部分，所使用的是 Crossbow 公司所研發無線感測器 MICAz；在嵌入式系統部份，所使用的也是 Crossbow 公司所研發同時可支援感測器介面的嵌入式系統 Stargate；在載具部份，是由 Lego 公司所研發出來的 NXT 機器人[7]；以下將針對各硬體作說明。

4.1 感測器 MPR2400 (MICAz)

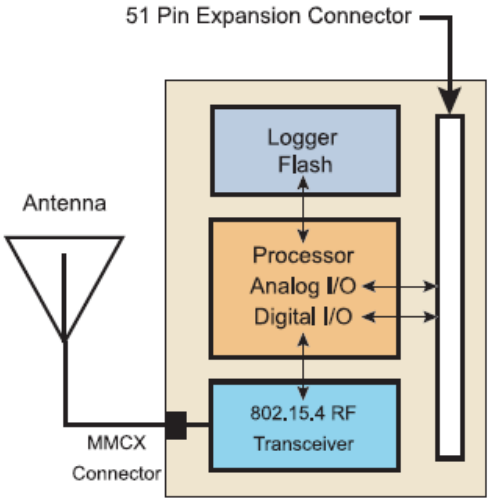


圖 18 感測器系統方塊圖

感測器硬體設計分成四部份，感測器 MPR2400—處理單元-Atmega128 微處理器、電力供應單元-3A 電池及傳輸單元-CC2420 晶片，另外 MPR2400 並外加了一個 4Mbits Serial Flash，用以存放感測單元所感測的大量資料，我們可透過無線感測網路發展工具中所提供的 MIB510 燒錄板將程式透過感測器 51 PIN Expansion Connector 燒錄至感測器的 Flash。

4.2 嵌入式平台 (Stargate)

Stargate 是由 Crossbow 公司所發展出 Linux 平台的嵌入式系統，具有強大的溝通及資料處理能力，也提供很多的硬體介面去跟外部硬體溝通，也同時支援 Crossbow 公司所發展出的無線網路感測器，可作為無線感測器與外部硬體介面溝通的橋樑。

4.3 Lego NXT Device

LEGO MINDSTORMS NXT（以下簡稱 NXT）是知名玩具廠商樂高公司在2006年推出的新一代智慧型機器人開發套件，從「玩具」的角度來看，LEGO MINDSTORMS系列產品徹底顛覆了傳統玩具的概念，它讓使用者發揮自己的想像力，從組裝各種模型或機械結構開始，配合圖形介面的開發工具撰寫簡單的應用程式，藉此控制機械模型的動作；而我們將 NXT組裝成一個可移動式的載具，透過嵌入式系統(Stargate)的USB 控制移動的方向及距離，使感測器具有移動的能力。

4.4 TinyOS 與 nesC 語言

4.4.1 介紹

一般的作業系統都是量大且複雜的程式，也並沒有內建能量管理機制；在感測網路的作業系統中，有越來越多的限制，除了必須具備能源管理機制外，對於資料處理的即時性，或是隨時回應硬體的能力等，都是必備的條件，以下簡單的列出感測網路的作業系統所具備的幾項特性：

- 緊密性：感測網路作業系統是在有限記憶體空間的小型嵌入式系統內執行，所以作業系統必須儘可能的小。
- 簡單化：雖然感測網路的作業系統必須很小，但仍然必須具備簡單的多工作(Multi-Tasking)處理及記憶體管理。
- 可靠性：由於感測器在感測網路中並沒有辦法手動將感測器重置，所以作業系統必需要求所有具有效性的設備(如看門狗計時器)都有一定的可靠度及自我保持能力(Self-Maintenance)。
- 遠端操控：在無線感測網路的作業系統中，還必須能夠被遠端的裝置重新程式化或是對遠端裝置下達重新程式化的命令。
- 省電：對電力受到限制的感測網路而言，省電是最重要的工作，而作業系

統就是要能夠儘可能的使系統的運作時間達到最少。

在本論文的感測器中，內部的作業系統是一套由美國柏克萊大學所開發，稱作 TinyOS 的作業系統，其專門用在具無線通訊能力、具自組網路能力及要求低消耗功率的小型嵌入式裝置中，而其所使用的語言也是由柏克萊大學所自行研發，一種稱作 nesC 的語言，其結合了標準 C 語言及 JAVA 兩種語言的優點。

4.4.2 TinyOS 簡介

TinyOS[8]是一套是由加州大學的伯利克分校開發出來的作業系統，具開放性原始碼，並且支援多種的無線網路協定標準，如圖 5-4 所示，由圖中可知，低傳輸率和短距離傳送之通訊協定標準的嵌入式系統硬體裝置，均適合在 TinyOS 的環境下運作。

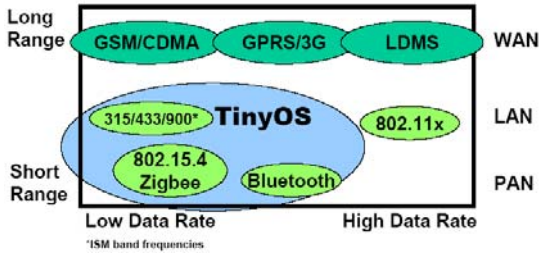


圖 19 TinyOS 支援各種低傳輸率及短距離之通訊協定

TinyOS 是由各種不同的物件程式所組成，當然使用者也可以依不同的應用需求，自行設計各種不同元件程式的功能；在 TinyOS 的應用中，每個應用都是採用模組化的設計，將各種不同功能的物件組合成使用者所要求的功能，所以它的程序核心往往都很小（一般來說核心程式大概在 400Bytes 左右），能夠突破感測器儲存資源少的限制，這能夠讓 TinyOS 很有效的運行在無線感測器網路上並去執行相應的管理工作等。

TinyOS 的應用模組中，每個物件間的關係是透過”配線(Wired)”的概念，建立物件與物件間的關係，且只有透過配線建立關係的兩個或多個物件才可以完成下命令或是回傳信號等功能；而每個物件除了建立關係外，還必須有個溝通的窗口，在 TinyOS 的物件中，是以『介面(Interface)』的概念，藉由介面與連線的概念，讓物件與物件具備溝通的管道及窗口。

4.4.3 nesC 語言簡介

nesC 是一套結合 C 語言及 JAVA 兩種程式語言之概念的新程式語言，其主要的設計概念是採用 JAVA 的純物件導向的特性，再加上 C 語言『指標』的概念，補足 JAVA 無法對記憶體存取的缺點，讓程式設計者在設計程式時更具備靈活性，TinyOS 應用中的程式碼即是用 nesC 語言撰寫。

5. 無縫隙覆蓋率系統實作

5.1 硬體設備

我們利用移動式感測器、資料搜集端及 Host 端來達到我們無縫隙的感測網路，以下是我們硬體的介紹。

5.1.1 移動式感測器

由 Micaz 感測器、Stargate 嵌入式系統及 Lego NXT 機器人所組成，定期將自己所在位置傳送給資料搜集端，也隨時根據 Host 端所下達新的位置做移動。

5.1.2 資料搜集端

由 Micaz 感測器、Stargate 嵌入式系統及無線網卡所組成，為感測網路與無線網路(802.11) 溝通的橋梁，負責搜集感測器的位址資料傳送給主控端，也負責將主控端的命令傳送給各個感測器。

5.1.2 Host 端

由 IBM 筆記型電腦及無線網卡所組成，接收所有感測器回傳的位址資訊，透過我們的 CADT 演算法來判斷是否有未覆蓋的區域；若有未覆蓋的區域，則透過 CADT 演算法計算出每個感測器重新達到無縫隙覆蓋的最佳位置，然後透過資料搜集端傳送新的位置給各個移動感測器做移動。

5.2 系統協定及佈置

5.2.1 系統通訊協定

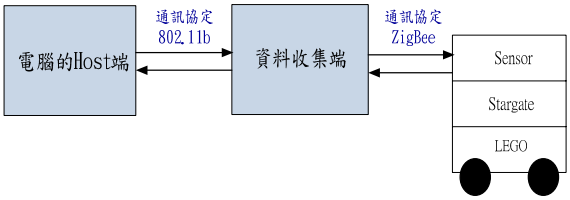


圖 20 系統通訊協定

5.2.2 系統佈置

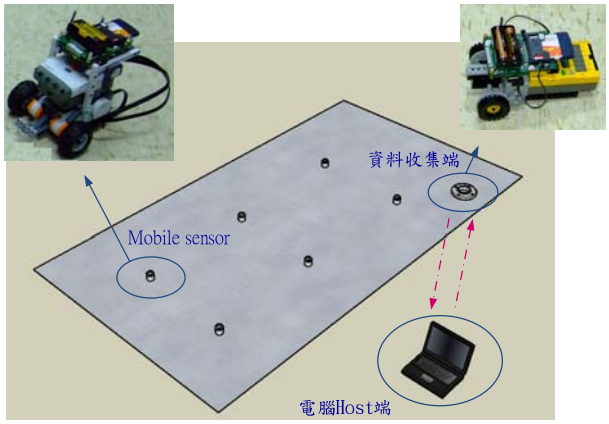


圖 21 系統佈置圖

5.3 實作流程簡述

本系統針對損毀的感測器，導致整體覆蓋率下降的問題作探討;透過本論文所提出CADT 演算法，重新佈置整個網路，來提升整體網路覆蓋率。

步驟 I

首先，先將感測器佈置於整個監控環境中，我們可以藉由 Host 端去監控感測器的分佈位置以及覆蓋範圍，我們可以由圖 23(a)得知，此時我們的Mobile sensor 完全覆蓋住整個監測環境。

步驟 II

以下模擬當監測環境中的 Mobile sensor 發生不可預測的故障時，我們所設計的無線感測網路，其 Mobile sensor 可以藉由適當的移動，自行修復無線感測網路，我們假設右下方的感測器，為即將失效的節點；而圖 23(b)表示由 Host 端可以觀察到原先的感測網路中出現漏洞。

步驟 III

透過CADT演算法，使得每個感測器可以計算出它們所需要移動的位移和方向，如圖 23(c)，如此一來，我們可以達到無縫隙的覆蓋範圍，進而使我們的無線感測網路可以正常運作。

6. 結論及未來展望

在這篇論文中，我們提出一個改善無線感測網路覆蓋率的演算法；我們將整個演算法分成三個步驟:

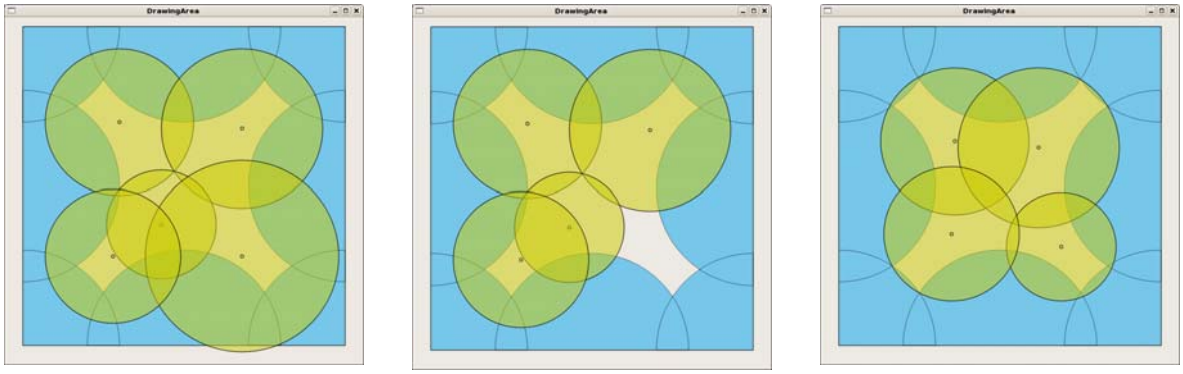
(1)將整個監控環境切割成許多大小不一的三角形。

(2)透過幾何的運算，找出每個三角形沒有被感測器覆蓋的區域(CADT-HD)。

(3)針對每個三角形沒有被覆蓋的區域，找出與鄰近感測器重疊範圍較大且分布較密集區域中的感測器進行移動，來填補三角形中沒有被覆蓋的區域(CADT-HR)。

重覆以上步驟，直到整體覆蓋率達到所設定的覆蓋率門檻值(Coverage Threshold) 才終止；我們模擬中可以明顯的看出，隨著演算法做的次數增加，其覆蓋率明顯的上升。

在未來，我們可以針對 Local_Weight、Global_Weight 及 β 做改進，讓演算法更能挑選重疊範圍較大且分布較密集區域的感測器做移動，能使整個演算法以做最少的次數來達到最佳的覆蓋效果。



(a)初始分佈圖　(b) (感測器損毀情況下)　(c) (網路修復完成)

圖 23　Host 端顯示分佈圖

參考資料

[1] Chun-Hsien Wu, and Kuo-Chuan Lee. A Delaunay Triangle Based Method for Wireless Sensor Network Deployment. In Proc. International Conference on Parallel and Distributed Systems, 2006.

[2] S. Dhillon, K. Chakrabarty, and S. Iyengar. Sensor Placement for Grid Coverage under Imprecise Detections. In Proc. International Conference on Information Fusion, 2002.

[3] T. Yan, T. He, and J. Stankovic. Differentiated Surveillance for Sensor Network. In Proc. International Conference on Embedded Networked Sensor Systems.

[4]http://www.cse.unsw.edu.au/~lambert/java/3d/delaunay.html

[5] Y. Zou, and K. Chakrabarty. Sensor Deployment and Target Localization Based on Virtual Forces. In Proc. INFOCOM, 2003.

[6]http://www.xbow.com/Home/wHomePage.aspx

[7] http://www.lego.com/en-US/default.aspx

[8] http://www.tinyos.net