Localization in WSN

Software Recommended: NetSim Standard v11.0, Microsoft Visual Studio 2015/2017

Project Download Link:

https://github.com/NetSim-TETCOS/Localization_in_WSN_v11.0/archive/master.zip

Localization is the process of finding the physical or relative location of a sensor node as data and information are useless if the nodes have no idea of their geographical positions. GPS (global positioning system) is the simplest method for localization of nodes, but it becomes very expensive if large number of nodes exists in a given network.

Anchor Nodes:

Sensor nodes with known location information are called "Anchor nodes". Typically, anchor nodes obtain their location information by using a global positioning system (GPS), or by manually being placed at defined coordinates.

Unknown Nodes:

Sensor nodes with unknown location information are called "Non-Anchor nodes" or "Unknown nodes". Localization is estimated through communication between localized node and unknown node for determining their geometrical placement or position. Location is determined by means of distance and angle between nodes.

Trilateration:

Location of node is estimated through distance measurement from three nodes. In this concept, intersection of three circles is calculated, which gives a single point which is a position of unknown node.

Use the distance equation. If your unknown point is (x, y) and known points are (x_i, y_i) which are at distances r_i from unknown point, then you get three equations:

$$(x - x1)^2 + (y - y1)^2 = r1^2$$

$$(x - x2)^2 + (y - y2)^2 = r2^2$$

$$(x - x3)^2 + (y - y3)^2 = r3^2$$

To calculate the distance between to sensors we have used NetSim API

```
DEVICE_DISTANCE(d1,d2)
```

Expand out the squares and subtract the second equation from the first and third equation from second, we get

$$2(x2 - x1)x + 2(y2 - y1)y = r1^2 - r2^2 + x2^2 - x1^2 + y2^2 - y1^2$$

$$2(x3 - x2)x + 2(y3 - y2)y = r2^2 - r3^2 + x3^2 - x2^2 + y3^2 - y2^2$$

This is a system of two equations with two unknowns:

$$Ax + By = C$$

$$Dx + Ey = F$$

The values of x and y is obtained from the below equations:

$$x = (CE - FB) / (EA - BD)$$

$$y = (CD - AF) / (BD - AE)$$

Localisation in NetSim:

1. To implement Localisation, we have added **Localisation.c** file in Zigbee project. The file contains the following functions:

   - `int fn_NetSim_localisation()`

     This function is used to find the anchor nodes based on the highest received powers received at unknown sensors from anchor nodes.

   - `int fn_NetSim_trilateration_method()`

     This function is used to implement the trilateration method to calculate the position / location of the unknown sensor.

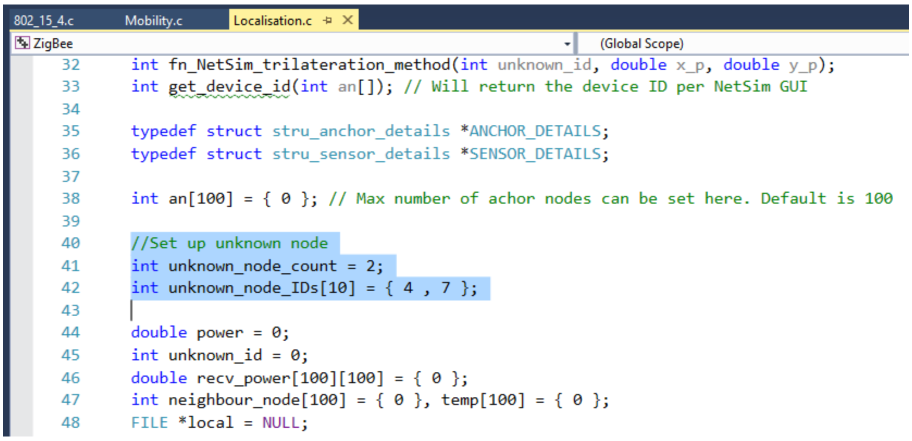
   - `bool IsUnknownNode()`

     This function is used to check whether the given node is unknown node or not.

   - `bool determine_anchor_node()`

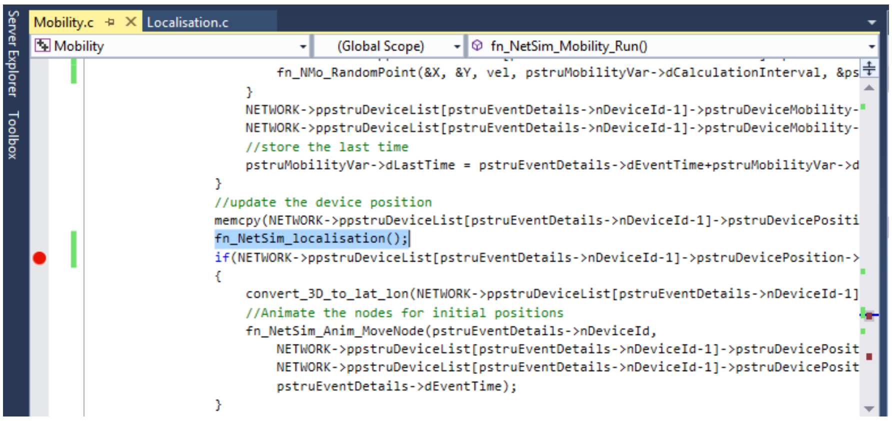
     This function is used to check whether the given node is anchor node or not.

2. Users can give their own unknown node IDs and unknown node count in **Localisation.c** file. NetSim knows all the positions of sensor nodes. Localisation is used to find the position of unknown nodes and then comparing this position with NetSim sensor positions.



```c
int fn_NetSim_trilateration_method(int unknown_id, double x_p, double y_p);
int get_device_id(int an[]); // Will return the device ID per NetSim GUI

typedef struct stru_anchor_details *ANCHOR_DETAILS;
typedef struct stru_sensor_details *SENSOR_DETAILS;

int an[100] = { 0 }; // Max number of achor nodes can be set here. Default is 100

//Set up unknown node
int unknown_node_count = 2;
int unknown_node_IDs[10] = { 4 , 7 };

double power = 0;
int unknown_id = 0;
double recv_power[100][100] = { 0 };
int neighbour_node[100] = { 0 }, temp[100] = { 0 };
FILE *local = NULL;
```

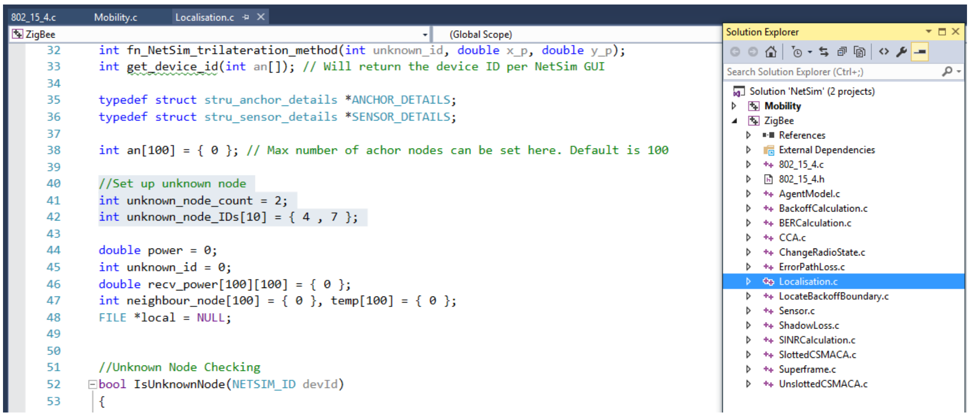
3. Since the unknown nodes are mobile, we have added a call to localisation in **fn_NetSim_Mobility_Run()** function present in **mobility.c** file inside Mobility project to calculate the new positions of the unknown node whenever a node moves.



```c
            fn_NMo_RandomPoint(&X, &Y, vel, pstruMobilityVar->dCalculationInterval, &ps
        }
        NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId-1]->pstruDeviceMobility-
        NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId-1]->pstruDeviceMobility-
        //store the last time
        pstruMobilityVar->dLastTime = pstruEventDetails->dEventTime+pstruMobilityVar->d
    }
    //update the device position
    memcpy(NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId-1]->pstruDevicePositi
    fn_NetSim_localisation();
    if(NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId-1]->pstruDevicePosition->
    {
        convert_3D_to_lat_lon(NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId-1]
        //Animate the nodes for initial positions
        fn_NetSim_Anim_MoveNode(pstruEventDetails->nDeviceId,
            NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId-1]->pstruDevicePosit
            NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId-1]->pstruDevicePosit
            pstruEventDetails->dEventTime);
    }
```

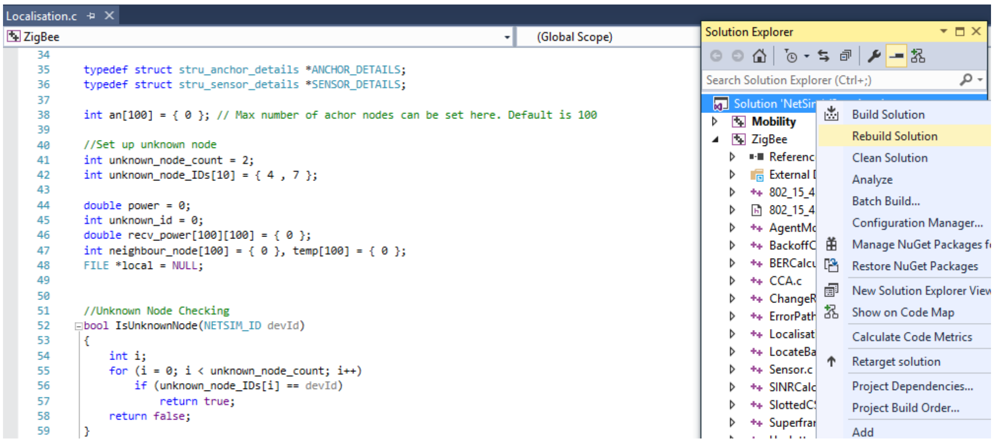
Steps:

- After you unzip the file open Code folder and double click on the NetSim.sln file present to open the project in Visual Studio 2015.



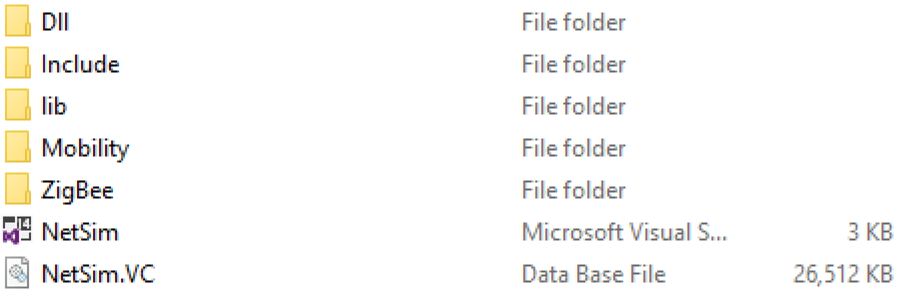
```c
int fn_NetSim_trilateration_method(int unknown_id, double x_p, double y_p);
int get_device_id(int an[]); // Will return the device ID per NetSim GUI

typedef struct stru_anchor_details *ANCHOR_DETAILS;
typedef struct stru_sensor_details *SENSOR_DETAILS;

int an[100] = { 0 }; // Max number of achor nodes can be set here. Default is 100

//Set up unknown node
int unknown_node_count = 2;
int unknown_node_IDs[10] = { 4 , 7 };

double power = 0;
int unknown_id = 0;
double recv_power[100][100] = { 0 };
int neighbour_node[100] = { 0 }, temp[100] = { 0 };
FILE *local = NULL;


//Unknown Node Checking
bool IsUnknownNode(NETSIM_ID devId)
{
```

- Right click on Solution in Solution Explorer and select rebuild solution.



```c

typedef struct stru_anchor_details *ANCHOR_DETAILS;
typedef struct stru_sensor_details *SENSOR_DETAILS;

int an[100] = { 0 }; // Max number of achor nodes can be set here. Default is 100

//Set up unknown node
int unknown_node_count = 2;
int unknown_node_IDs[10] = { 4 , 7 };

double power = 0;
int unknown_id = 0;
double recv_power[100][100] = { 0 };
int neighbour_node[100] = { 0 }, temp[100] = { 0 };
FILE *local = NULL;


//Unknown Node Checking
bool IsUnknownNode(NETSIM_ID devId)
{
    int i;
    for (i = 0; i < unknown_node_count; i++)
        if (unknown_node_IDs[i] == devId)
            return true;
    return false;
}
```

- Upon rebuilding, **libZigbee.dll** and **libMobility.dll** will get created in the DLL folder.



Dll	File folder	
Include	File folder	
lib	File folder	
Mobility	File folder	
ZigBee	File folder	
NetSim	Microsoft Visual S...	3 KB
NetSim.VC	Data Base File	26,512 KB

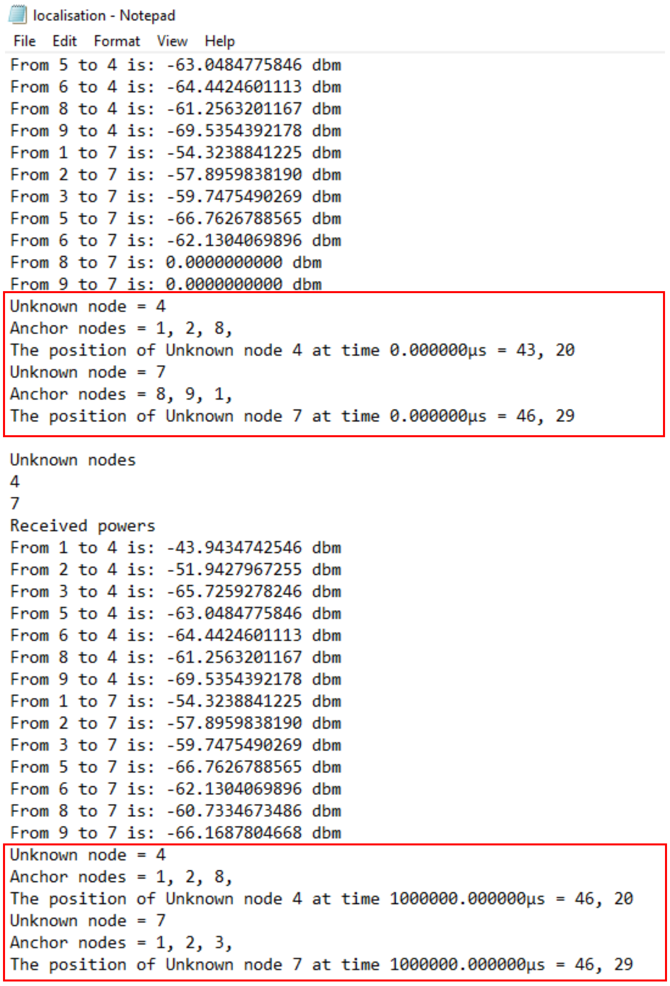
- Now copy the **libZigbee.dll** and **libMobility.dll** from this DLL folder and paste it in NetSim bin folder present in the NetSim installation directory. The NetSim install directory would look something like **"C:\Program Files\NetSim Standard\bin"**.

- Note that there exists **libZigbee.dll** and **libMobility.dll** in this bin folder. This is the default file being shipped with NetSim. The user is replacing this file with the newly built file.

- Therefore, take care to rename the original **libZigbee.dll** and **libMobility.dll** files, so that it isn't lost. For example, you may rename it as **libZigbee_default.dll**, **libMobility_default.dll**.

- Run NetSim and open **Configuration.netsim** file present inside the Config_File folder and run the simulation.

- After simulation, localisation.txt file will get created in the bin folder of NetSim which logs the unknown node IDs, received powers from all anchor nodes to unknown nodes, anchor node IDs based on highest received powers and the position or coordinates of the unknown nodes with variation in time as shown below.



```
From 5 to 4 is: -63.0484775846 dbm
From 6 to 4 is: -64.4424601113 dbm
From 8 to 4 is: -61.2563201167 dbm
From 9 to 4 is: -69.5354392178 dbm
From 1 to 7 is: -54.3238841225 dbm
From 2 to 7 is: -57.8959838190 dbm
From 3 to 7 is: -59.7475490269 dbm
From 5 to 7 is: -66.7626788565 dbm
From 6 to 7 is: -62.1304069896 dbm
From 8 to 7 is: 0.0000000000 dbm
From 9 to 7 is: 0.0000000000 dbm
Unknown node = 4
Anchor nodes = 1, 2, 8,
The position of Unknown node 4 at time 0.000000µs = 43, 20
Unknown node = 7
Anchor nodes = 8, 9, 1,
The position of Unknown node 7 at time 0.000000µs = 46, 29

Unknown nodes
4
7
Received powers
From 1 to 4 is: -43.9434742546 dbm
From 2 to 4 is: -51.9427967255 dbm
From 3 to 4 is: -65.7259278246 dbm
From 5 to 4 is: -63.0484775846 dbm
From 6 to 4 is: -64.4424601113 dbm
From 8 to 4 is: -61.2563201167 dbm
From 9 to 4 is: -69.5354392178 dbm
From 1 to 7 is: -54.3238841225 dbm
From 2 to 7 is: -57.8959838190 dbm
From 3 to 7 is: -59.7475490269 dbm
From 5 to 7 is: -66.7626788565 dbm
From 6 to 7 is: -62.1304069896 dbm
From 8 to 7 is: -60.7334673486 dbm
From 9 to 7 is: -66.1687804668 dbm
Unknown node = 4
Anchor nodes = 1, 2, 8,
The position of Unknown node 4 at time 1000000.000000µs = 46, 20
Unknown node = 7
Anchor nodes = 1, 2, 3,
The position of Unknown node 7 at time 1000000.000000µs = 46, 29
```

Open the sample configuration file present in the Config_File folder. Users can also create their own network scenarios in WSN and run simulation.

Settings that were done to create the network scenario for localisation:

- Set grid length 100*100 m.
- First drop sensors and then drop sink node and agent.
- Set the unknown node ids in the **Localisation.c** file as explained above (default unknown nodes are 4, 7).
- The unknown node IDs which are setting in the code should match with the sensor IDs in the GUI.
- Set mobility for unknown nodes (5m/s or 10m/s).
- Set channel characteristics as Path loss only, Path loss model as LOG DISTANCE and Path loss exponent as 3.5.
- Run simulation and check for localisation.txt file gets created in bin folder.
- Open the text file and compare the positions of unknown nodes with variation in time.