

# Localization in WSN

---

**Software Recommended:** NetSim Standard v12.1/v12.2 (32 bit/ 64 bit), Microsoft Visual Studio 2019

Follow the instructions specified in the following link to clone/download the project folder from GitHub using Visual Studio:

<https://tetcos.freshdesk.com/support/solutions/articles/14000099351-how-to-clone-netsim-file-exchange-project-repositories-from-github->

Other tools such as GitHub Desktop, SVN Client, Sourcetree, Git from the command line, or any client you like to clone the Git repository.

**Note:** It is recommended not to download the project as an archive (compressed zip) to avoid incompatibility while importing workspaces into NetSim.

**Secure URL for the GitHub repository:**

**v12.1:** [https://github.com/NetSim-TETCOS/Localisation\\_in\\_WSN\\_v12.1.git](https://github.com/NetSim-TETCOS/Localisation_in_WSN_v12.1.git)

**v12.2:** [https://github.com/NetSim-TETCOS/Localisation\\_in\\_WSN\\_v12.2.git](https://github.com/NetSim-TETCOS/Localisation_in_WSN_v12.2.git)

**Note:** The cloned project directory will contain the documentation specific to the NetSim version (v12.1/v12.2).

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:

$$(x - x_1)^2 + (y - y_1)^2 = r_1^2$$

$$(x - x_2)^2 + (y - y_2)^2 = r_2^2$$

$$(x - x_3)^2 + (y - y_3)^2 = r_3^2$$

To calculate the distance between two 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

$$2(x_2 - x_1) + 2(y_2 - y_1)y = r_1^2 - r_2^2 + x_2^2 - x_1^2 + y_2^2 - y_1^2$$

$$2(x^3 - x^2) + 2(y^3 - y^2)y = r^2^2 - r^3^2 + x^3^2 - x^2^2 + y^3^2 - y^2^2$$

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.

```

32 int fn_NetSim_trilateration_method(int unknown_id, double x_p, double y_p);
33 int get_device_id(int an[]); // Will return the device ID per NetSim GUI
34
35 typedef struct stru_anchor_details *ANCHOR_DETAILS;
36 typedef struct stru_sensor_details *SENSOR_DETAILS;
37
38 int an[100] = { 0 }; // Max number of anchor nodes can be set here. Default is 100
39
40 //Set up unknown node
41 int unknown_node_count = 2;
42 int unknown_node_IDS[10] = { 4 , 7 };
43
44 double power = 0;
45 int unknown_id = 0;
46 double recv_power[100][100] = { 0 };
47 int neighbour_node[100] = { 0 }, temp[100] = { 0 };
48 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.

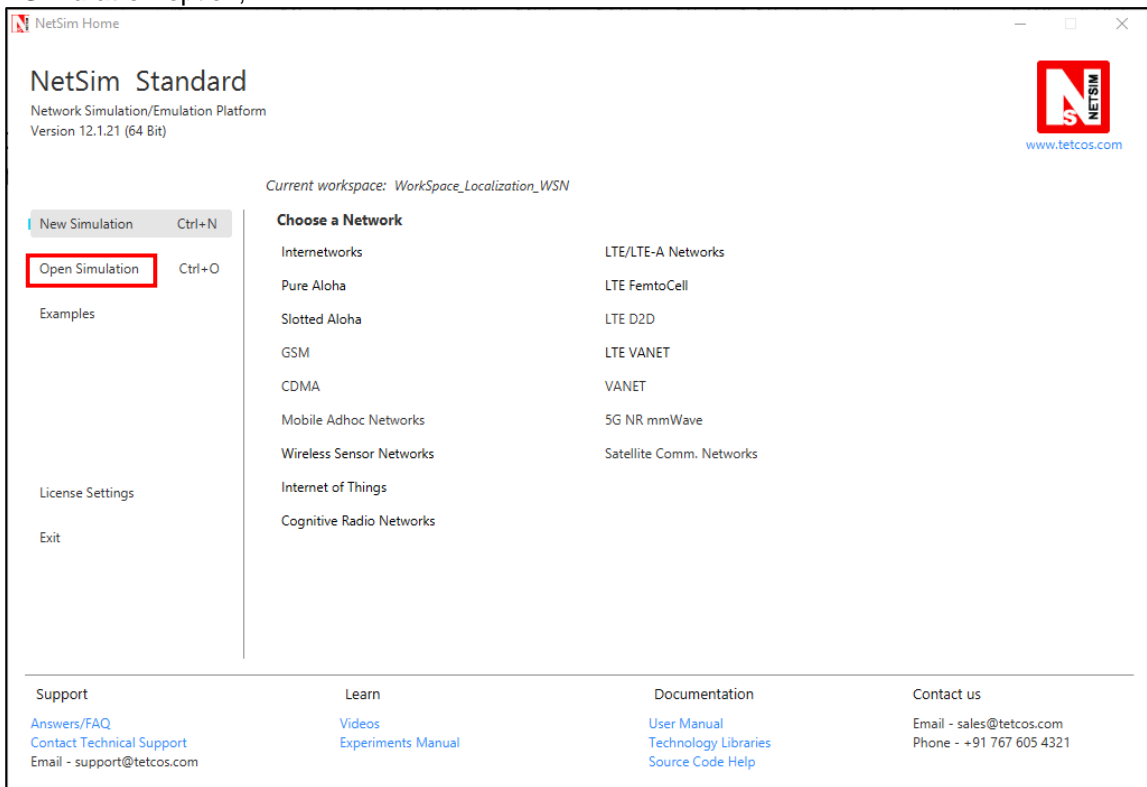
```

Mobility.c  Localisation.c
(Global Scope)  fn_NetSim_Mobility_Run()
466 memcpy(cor, ncor, sizeof* cor);
467
468 if(pstruMobilityVar->dLastTime+pstruMobilityVar->dPauseTime*SECOND<pstruEventDetails->dEventTime+pstruMobilityVar->dCalculati
469 {
470     fn_NMo_RandomPoint(&X, &Y, ve1, pstruMobilityVar->dCalculationInterval, &pstruMobilityVar->ulSeed1, &pstruMobilityVar->ul
471     while (cor->corrType == CORRTYPE_CARTESIAN &&
472           (X > dSimulationArea_X || X < 0 || Y < 0 || Y > dSimulationArea_Y))
473     {
474         X = cor->X;
475         Y = cor->Y;
476         fn_NMo_RandomPoint(&X, &Y, ve1, pstruMobilityVar->dCalculationInterval, &pstruMobilityVar->ulSeed1, &pstruMobilityVar
477     }
478     ncor->X = X;
479     ncor->Y = Y;
480     //store the last time
481     pstruMobilityVar->dLastTime = pstruEventDetails->dEventTime+pstruMobilityVar->dCalculationInterval;
482 }
483 //update the device position
484 fn_NetSim_localisation();
485 memcpy(pos, cor, sizeof* pos);
486
487 mobility_pass_position_to_animation(pstruEventDetails->nDeviceId,
488                                   pstruEventDetails->dEventTime,
489                                   pos);
490
491 //Add event for next point
492 pstruEventDetails->dEventTime+=pstruMobilityVar->dCalculationInterval;
493 fnAddEvent(pstruEventDetails);
494 pstruEventDetails->dEventTime+=pstruMobilityVar->dCalculationInterval;
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

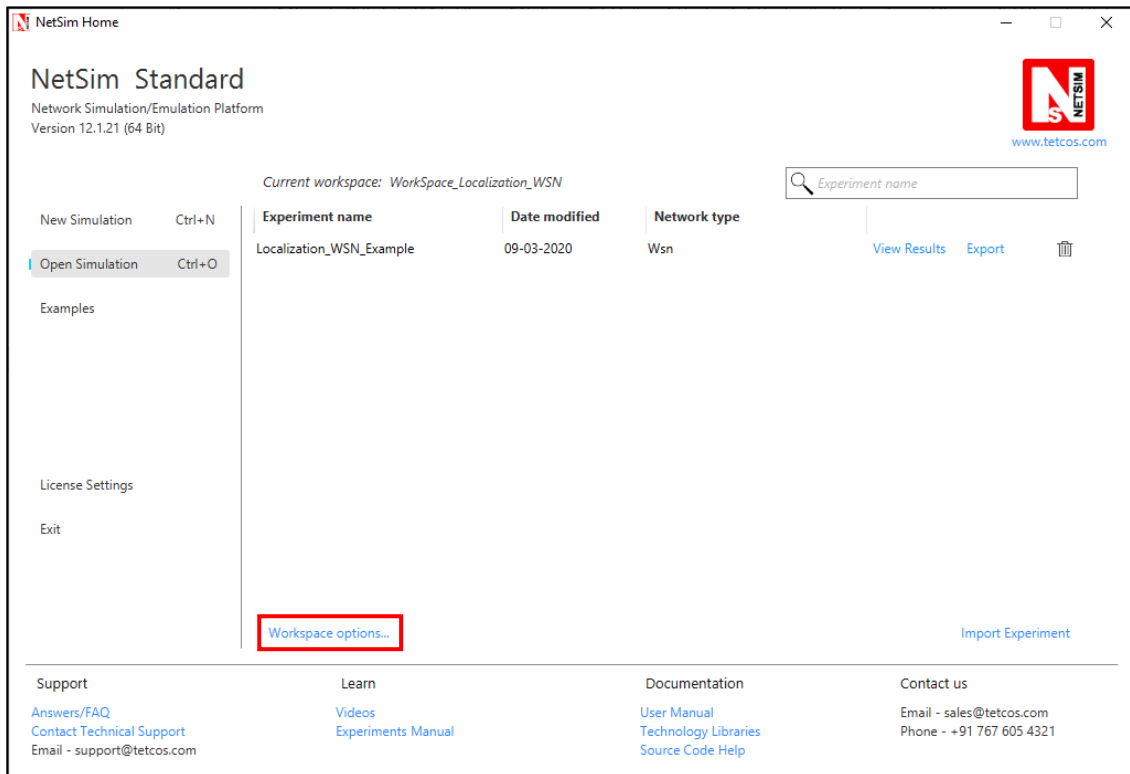
```

**Steps:**

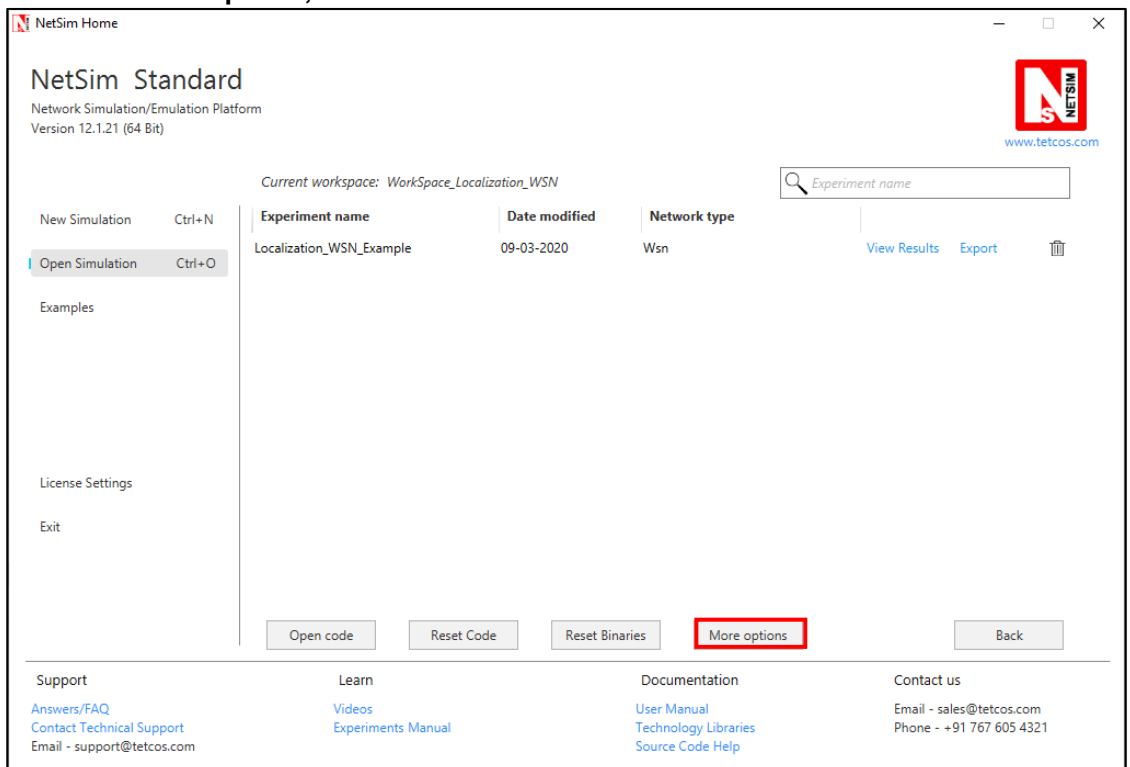
- After you unzip the downloaded project directory, Open NetSim Home Page click on **Open Simulation** option,



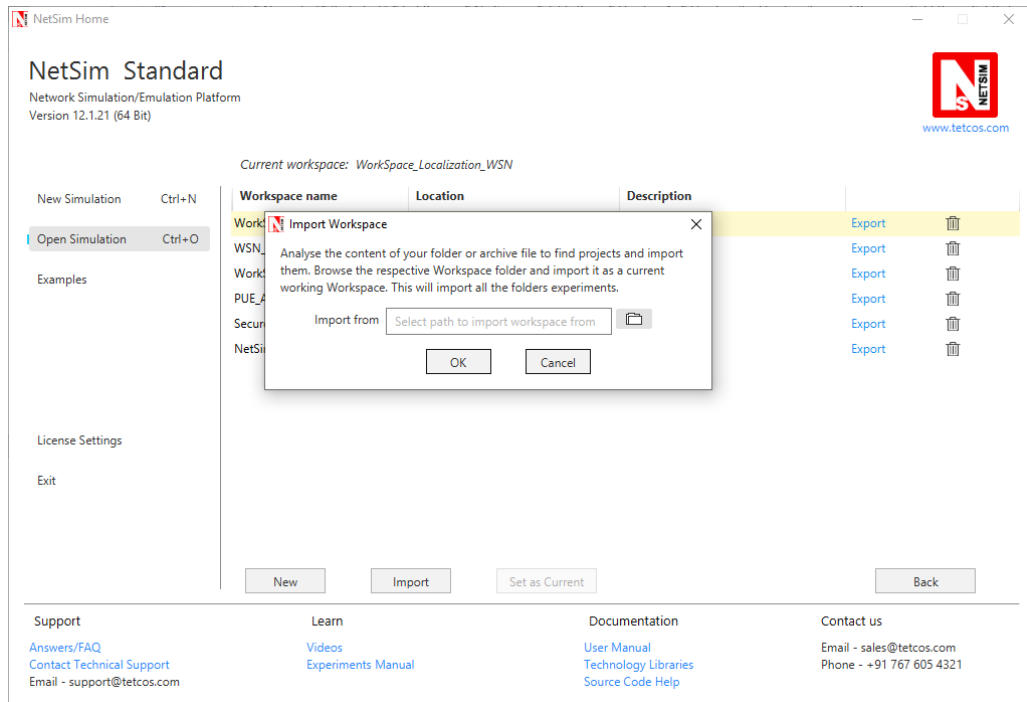
- Click on **Workspace options**



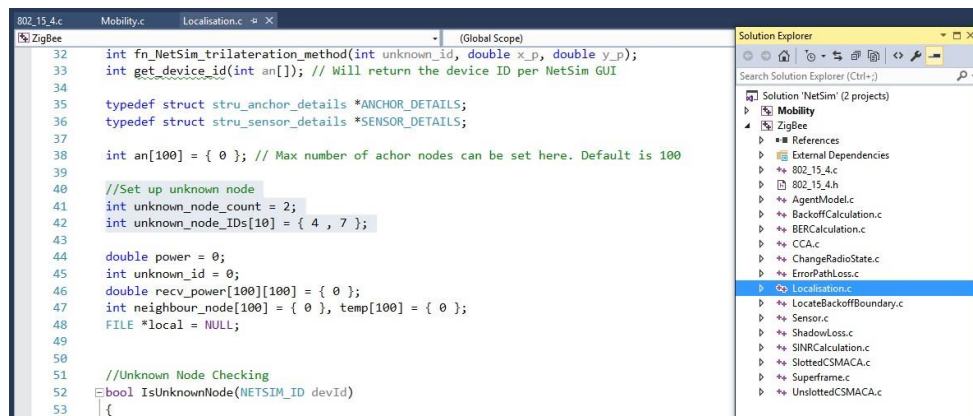
- Click on **More Options**,



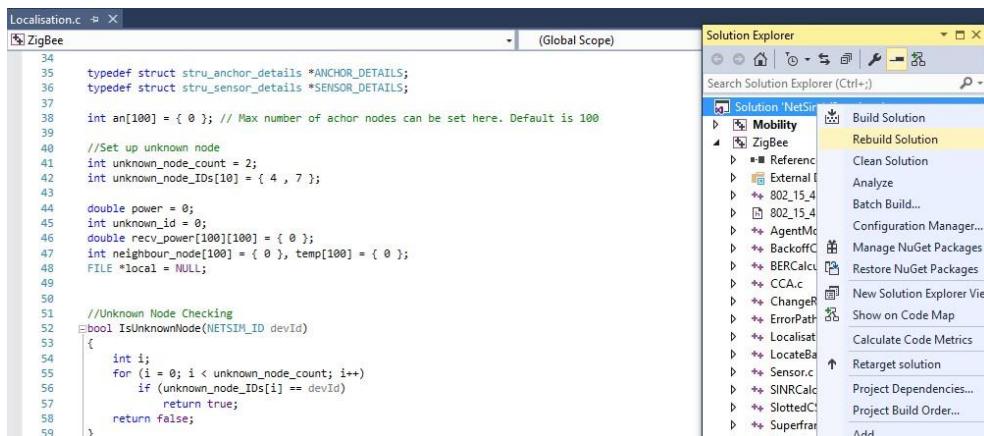
- Click on **Import**, browse the extracted folder path and go into the `WorkSpace_Localization_WSN` directory. Click on the **Select folder** button and then on **OK**.



- Go to home page, Click on **Open Simulation** → **Workspace options** → **Open code**

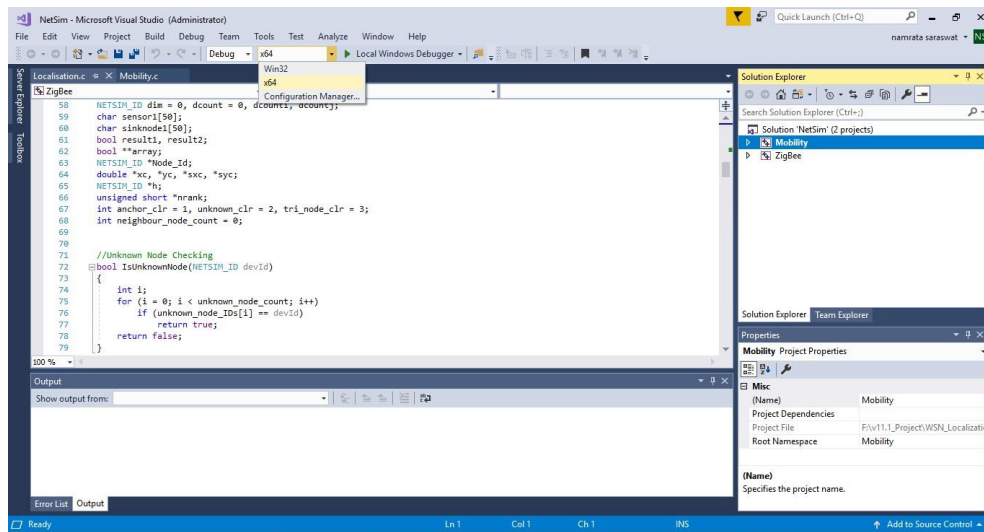


- Right click on Solution in Solution Explorer and select rebuild solution.

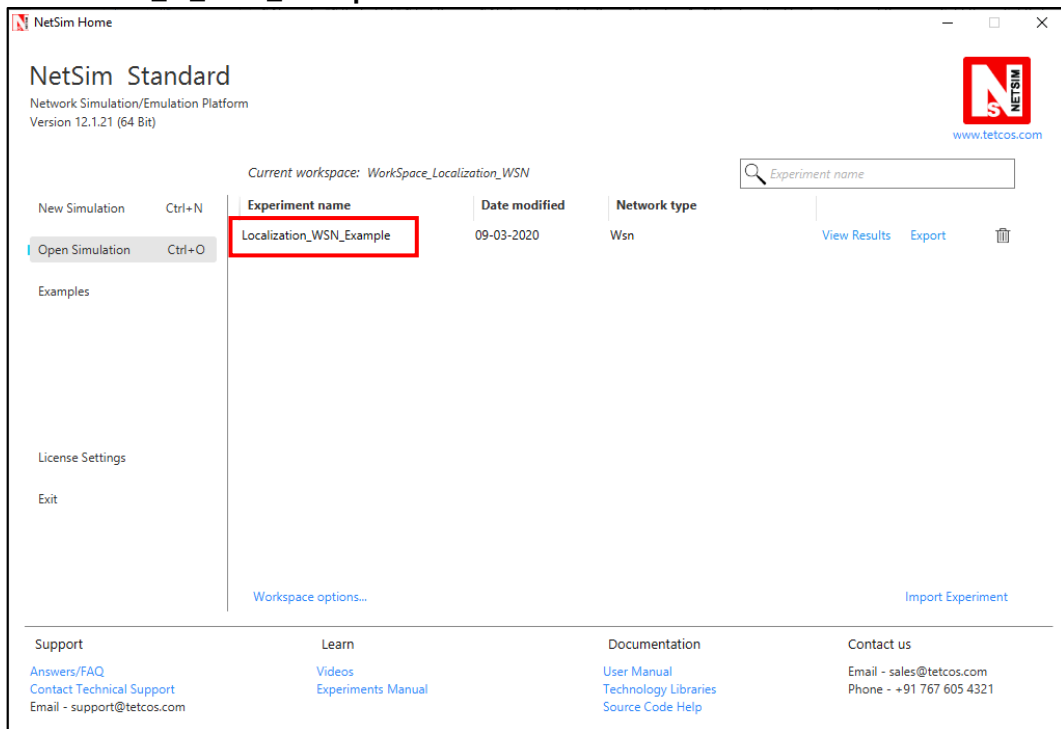


- Upon rebuilding, **libZigbee.dll** and **libMobility.dll** will automatically get updated in NetSim binary folder.

**Note:** Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit DLL files respectively as shown below:



- Go to NetSim home page, click on **Open Simulation**, Click on **Localization in WSN Example**.



- Run simulation after the network scenario gets loaded.
- After simulation, localisation.txt file will get created in the **bin** folder of NetSim. Right click on the NetSim shortcut icon in your desktop and select Open file location to go to NetSim bin folder. The localisation.txt file 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.

localisation - Notepad

File Edit Format View Help

From 3 to 7 is: -84.7827469788 dbm  
From 5 to 7 is: -85.1956026853 dbm  
From 6 to 7 is: -87.7656517820 dbm  
From 8 to 7 is: -89.4904372973 dbm  
From 9 to 7 is: -86.0518999457 dbm

Unknown node = 4  
Anchor nodes = 1, 2, 9,  
The position of Unknown node 4 at time 24000000.000000 $\mu$ s = 7, 12  
Unknown node = 7  
Anchor nodes = 1, 2, 3,  
The position of Unknown node 7 at time 24000000.000000 $\mu$ s = 62, 9

Unknown nodes

4

7

Received powers

From 1 to 4 is: -80.3528052692 dbm  
From 2 to 4 is: -83.1987039075 dbm  
From 3 to 4 is: -86.4787581036 dbm  
From 5 to 4 is: -87.2647955930 dbm  
From 6 to 4 is: -89.2354282518 dbm  
From 8 to 4 is: -90.6437199782 dbm  
From 9 to 4 is: -86.4472286436 dbm  
From 1 to 7 is: -77.8310409707 dbm  
From 2 to 7 is: -79.5792536503 dbm  
From 3 to 7 is: -84.7827469788 dbm  
From 5 to 7 is: -85.1956026853 dbm  
From 6 to 7 is: -87.7656517820 dbm  
From 8 to 7 is: -89.4904372973 dbm  
From 9 to 7 is: -86.0518999457 dbm

Unknown node = 4  
Anchor nodes = 1, 2, 9,  
The position of Unknown node 4 at time 24000000.000000 $\mu$ s = 7, 12  
Unknown node = 7  
Anchor nodes = 1, 2, 3,  
The position of Unknown node 7 at time 24000000.000000 $\mu$ s = 62, 9