International Journal of Computer Networks & Communications (IJCNC) vol 16, No 6, November 2024

OUTSTANDING FRAMEWORK FOR SIMULATING AND GENERATING ANCHOR TRAJECTORY IN WIRELESS SENSOR NETWORKS

Abdelhady Naguib [1, 2]

[1]Department of Computer Science, College of Computer and Information Sciences, Jouf University, Sakaka, Saudi Arabia
[2]Department of Systems and Computers Engineering, Faculty of Engineering, Al-Azhar University, Cairo, Egypt

ABSTRACT

This paper proposes a framework that has the ability to animate and generate different scenarios for the mobility of a movable anchor which can follow various paths in wireless sensor networks (WSNs). When the researchers use NS-2 to simulate a single anchor-assisted localization model, they face the problem of creating the movement file of the movable anchor. The proposed framework solved this problem by allowing them to create the movement scenario regarding different trajectories. The proposed framework lets the researcher set the needed parameters for simulating various static path models, which can be displayed through the graphical user interface. The researcher can also view the mobility of the movable anchor with control of its speed and communication range. The proposed framework has been validated by comparing its results to NS-2 outputs plus comparing it against existing tools. Finally, this framework has been published on the Code Project website and downloaded by many users.

KEYWORDS

Localization, Movable anchor, Simulation, Path planning, Wireless sensor networks

1. INTRODUCTION

Wireless sensor networks have benefits for different areas of applications, such as military, health care, resident applications, tracking, disaster prediction and recovery, and IoT (the Internet of Things). On the other hand, there are many challenges they face, such as restricted computing and processing, energy consumption, and storage capability [1]. WSN localization is the process of finding the location of a sensor node in the deployment field. Localization plays an important role in various WSN applications. It is useless to collect data without knowing the location of the source or event [2]. Many localization mechanisms have been proposed in recent years, but one of the most promising techniques is called static path planning algorithms, which use a single movable anchor node equipped with a GPS unit. The movable anchor node follows a predefined static path, broadcasting its current position periodically to unknown position sensor nodes. Unknown position sensor nodes can use localization algorithms such as range-free or range-based [3] to determine their location using information gathered from the movable anchor node.

Communication networks have become more enhanced and complex, which requires designing a modern and improved model to help in making an accurate analysis and in-depth understanding of the performance of these systems [4]. The simulation process is necessary to develop any modern model or algorithm before it is implemented in the real world. Therefore, a simulation

DOI: 10.5121/ijcnc.2024.16602

process is carried out to verify the suitability of the system to be implemented for the environment under implementation and its operability according to the functions desired from it. Before the virtual implementation of the system, simulation is performed in order to check for severe weaknesses or other design problems that may occur in the future [5]. On the basis of the type of application area in wireless sensor networks, simulation can be classified into continuous simulation, discrete event simulation (DES), deterministic simulation, parallel simulation, stochastic simulation, and hybrid simulation [6]. Below is a detailed description of these categories:

- **Continuous simulation:** Continuous simulations can be described by a set of constantly changing dependent variables. Continuous simulations deal with the modeling of physical events (processes, behaviors, and conditions) [7] used in differential equations that describe the physical processes.
- **Discrete event simulation:** This type of simulation is applied to systems in which events occur over discrete time periods because no change is expected between events, so every change observed in the system is equivalent to the occurrence of a specific event [8].
- **Stochastic simulation:** There are some systems in which variables change randomly as probabilities change and which have several possibilities for evolution. Applications used for this type of simulation are to simulate traffic flow in communications networks or to study climate changes [9].
- **Deterministic simulation:** A future event can be calculated accurately, without interfering with randomness, using deterministic simulation which has a previously known input and a unique set of outputs [10].
- **Hybrid simulation:** This type of simulation is a hybrid of the two types of simulations defined above, so it is considered more efficient for testing the target system because it uses more than one simulation model to evaluate the model [11].

The rest of this paper is organized as follows: Section 2 presents related works. Section 3 describes the process of framework analysis & design. Section 4 illustrates the framework interface design process. Section 5 presents the results verification and discussions. Finally, Section 6 provides the conclusion of this paper.

2. RELATED WORKS

This section presents static path panning algorithms that the proposed framework supports. The work presented in [12] proposes three different types of static path planning algorithms. The first algorithm is SCAN, where the movable anchor node starts moving vertically along with the monitoring area, and then moves horizontally by a distance equal to the trajectory resolution. The second algorithm is DOUBLE-SCAN, where the movable anchor node scans the deployment field in both directions so as to overcome the collinearity problem of the SCAN algorithm. The third algorithm is HILBERT, where the monitoring area is divided into four square regions connected by four line segments, ensuring that sensor nodes have no chance to receive collinear beacons from movable anchor node. Authors in [13] propose two static path- planning algorithms: CIRCLES and S-CURVES. According to CIRCLES, the movable anchor node follows an arrangement of concentric circles covering the monitoring area except the four corners. Hence, the second algorithm, called S-CURVES, tries to cover the four corners of the deployment field and avoid collinear beacon transmission. It allows the movable anchor node to move according to the S-shape both vertically and horizontally, like the SCAN algorithm.

Zhen Hu et al. [14] proposed a mobile anchor centroid localization method (MACL) where the movable anchor node moves in a spiral path, broadcasting its current position periodically.

Authors in [15] proposed a localization algorithm with a movable anchor node based on trilateration (LMAT). This method uses a movable anchor that moves according to the trilateration path in the sensing field while broadcasting its current position periodically. Y. C. Lin et al. [16] proposed a localization algorithm based on the triangle grid scan (TGS) method. This method aims to use Wi-Fi technology for anchor movement in an easy and simple manner.

The work in [17] proposes a new, superior method for path planning known as a Z-curve. To get around the issue of collinear beacons, the movable anchor moves along a static path that resembles a Z shape with several z-curves in the deployment region. For two-dimensional WSNs, the authors in [18] suggested a mobile anchor-assisted localization method (MAALRH) based on regular hexagons. A succession of regular hexagons that cross each other in the middle make up the path taken by the movable anchor. The entire sensing field is covered in this manner. A path-planning model called H-Curves was created by Alomari et al. [19] for mobility-assisted localization in wireless sensor networks. A movable anchor's trajectory is made up of numerous H-shaped pathways to get around the issue of collinear beacons.

K. Kannadasan et al. [20] proposed a novel path planning method for movable anchor-assisted localization called M-Curves. The authors optimized the localization system by using the Dolphin Swarm Algorithm (DSA) [21] so as to increase localization accuracy. Abdelhady Naguib and S. Ali [22] proposed a new static path-planning approach used in wireless sensor networks called SQUARE_SPIRAL, which enables a movable anchor to cover the whole deployed area to get high coverage. The trajectory of the movable anchor node follows a square spiral shape to decrease the number of collinear beacons and increase localization accuracy. Authors in [23] proposed a novel path planning method for movable anchor-based localization called Nested Hexagon Curves (NHexCurves). The trajectory of the movable anchor node based on drawing a pattern consisting of N-HexCurves so as to overcome the problem of collinear beacons using weighted centroid localization (WCL) [24] and accuracy priority trilateration (APT) [25] localization.

In this article, an outstanding framework for simulating anchor movement based on different static path-planning models is proposed. The proposed framework is an open-source tool that can be downloaded from the Code Project website, allowing researchers to change it to fit different scenario models or modify to it to make it work with other path planning mechanisms. The proposed framework has been tested with the well-known simulator NS-2, and the movable anchor's movement has been confirmed by running the same motion scenario files on the Network Animator (Nam) tool. The proposed framework has the following features: generating external text files storing the coordinates of movable anchor trajectories according to different static path planning mechanisms, animation for the movement of the movable anchor, and running of already-generated scenarios for the sake of comparison between different trajectory models.

3. FRAMEWORK ANALYSIS & DESIGN

This section is concerned with developing an object-oriented model of the proposed framework. The object class hierarchy of the proposed framework is shown in Figure 1 and consists of the following main classes: The first class is **User_Interface**, which has many attributes like *total sensors*, which represents the total number of sensors plus one anchor node. The other attribute is *simulation Time*, which represents the duration of the simulation. This class has two main operations: The first one is **network_Paint()**, which is responsible for painting objects on the specified GUI painting area. The other operation is **timerUpdate_Tick()**, which represents the elapsed real time.

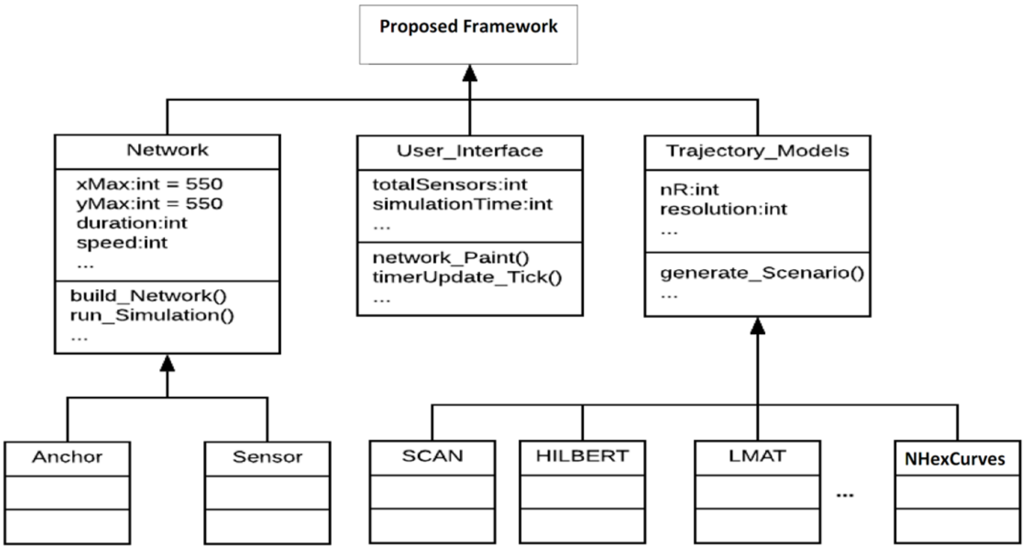


Figure 1. Framework Class Hierarchy

The second class is **Network**; this class contains all the attributes and operations needed for the wireless sensor network. The first main attribute is *xMax*, which represents the width of the deployment field in the x direction and has a default value of 550 *m*. The second one is *yMax*, which refers to the height of the deployment field in the y direction; it is also the default value, settled at 550 *m*. The other attribute is *duration*, which represents simulation time in seconds. The other attribute that controls the speed of the movable anchor is *speed* (*m/sec*). There are many operations associated with this class; The first main operation is **build_Network()**, which is responsible for setting the control data from the user interface. The second main operation is **run_Simulation()**, which controls the core processes for running the simulation.

The **Network** class has two sub-classes: **Anchor** and **Sensor**. As shown in Figure 2, the **Anchor** class defines a number of attributes that hold information about anchors, including their *ID*, (***x***, ***y***) position, *Radius*, which represents the communication range of the anchor, and *InitialEnergy*, which is a *double* variable that represents the initial energy of the anchor node measured in *joules*. Operations associated with this class are **move(x, y),** which is required for moving the movable anchor. The other operation is **send (m),** which is called periodically when the anchor broadcasts its current coordinates to neighboring sensor nodes.

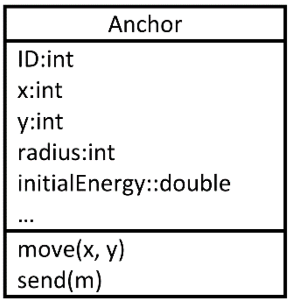


Figure 2. Anchor Class

The third main class of the proposed framework is **Trajectory_Models**; this class represents the static path planning mechanisms used in WSNs. There are many attributes for this class; the first main attribute is *nR*, which refers to the number of horizontal *R* segments needed for determining the size of the monitoring area. The second one is *resolution*, which refers to the length of the *R* segment; thus, According to operations associated with this class like **generate_Scenario()**, it is responsible for writing the coordinates of the movable anchor to an external text file. There are many trajectory models that inherit from the parent class **Trajectory_Models**. Figure 3 represents an example of a child class called **HILBERT**. The first two attributes of this class are *xInitial* and *yInitial*, which represent the initial position of the anchor in *x* and *y* coordinates, respectively. The third attribute is *curveLevel*, which refers to the level (1, 2... n) of the Hilbert trajectory model.

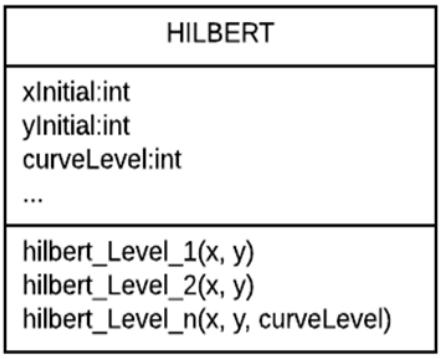


Figure 3. Hilbert Class

There are three operations needed to assign coordinate points to the movable anchor's trajectory. The first one is **hilbert_Level_1(x, y),** which takes two argument variables (*x*, *y*) that represent the initial position of the movable anchor node. The second one is **hilbert_Level_2(x, y)**, which is composed of many level-1 curves. For higher-level Hilbert curves, it is composed of many Level-2 curves that can be modeled by calling **hilbert_Level_n(x, y, curveLevel)**.

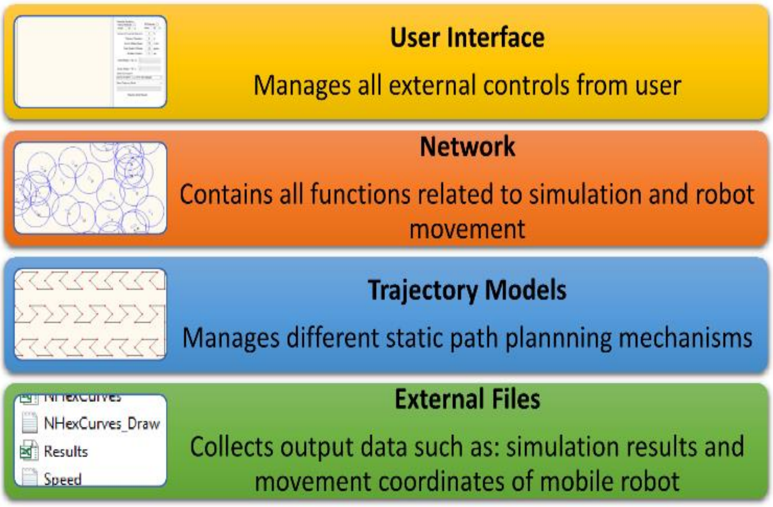

User Interface
Manages all external controls from user
Network
Contains all functions related to simulation and robot movement
Trajectory Models
Manages different static path plannning mechanisms
External Files
Collects output data such as: simulation results and movement coordinates of mobile robot
NHexCurves
NHexCurves_Draw
Results
Speed

Figure 4. Framework Architecture

The first step of the design process is based on defining the context of the proposed framework. The proposed framework supports multithreading, where the simulation process of the path planning methods runs on a separate thread from the thread controlling the process of the user interface. For better design, each class is encapsulated into a separate reference file. Hence allowing researchers to refer to their own trajectories within the proposed framework. The second step is to design the framework architecture as shown in Figure 4.

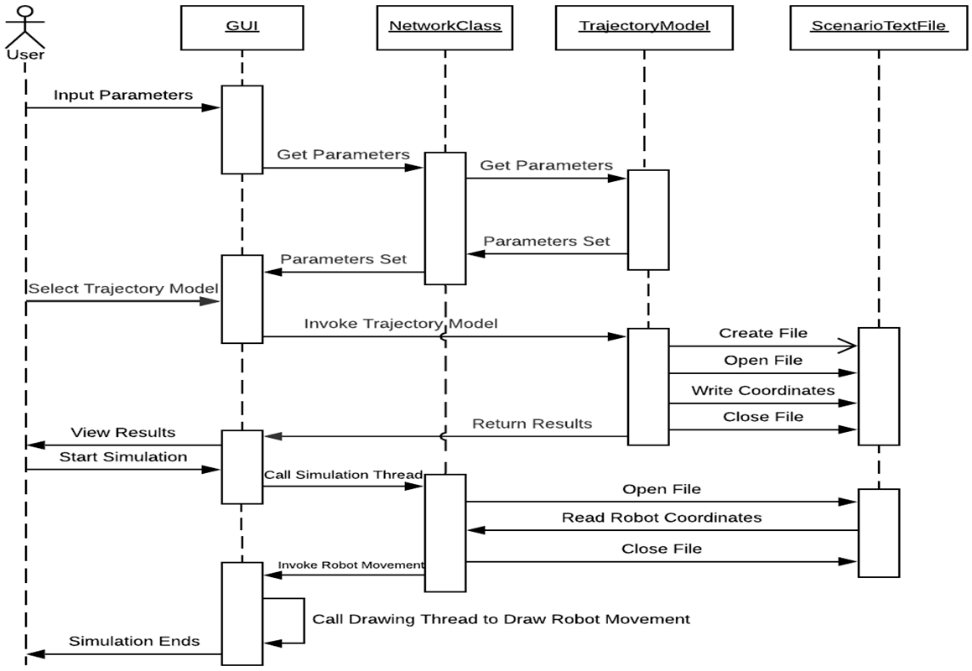


Figure 5. Framework Sequence Diagram

The proposed framework is a simple and flexible system with a layered model consisting of four main parts. The third step in architectural design is to model a system sequence diagram, as shown in Figure 5. The sequence of interactions when the user runs the simulation is as follows:

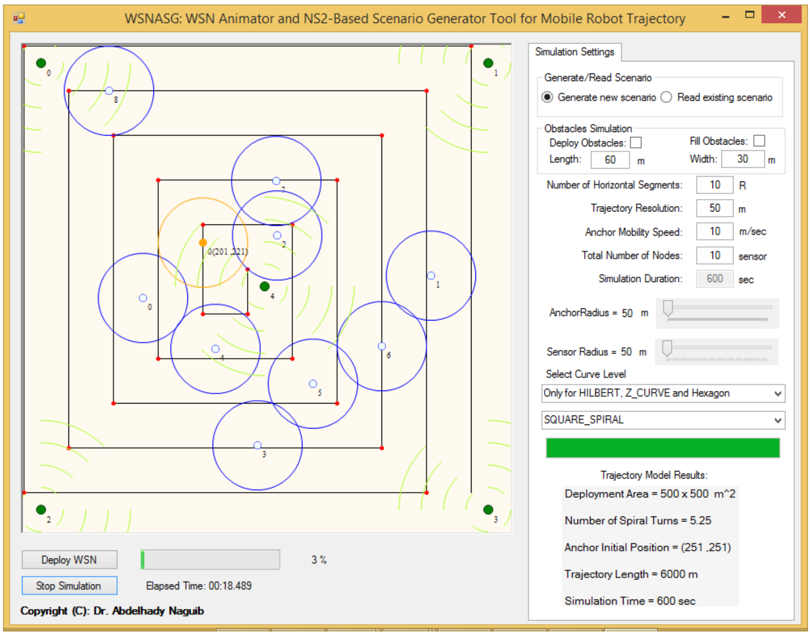
1) The user should enter the input data that is required for running the simulation.
2) After setting the required parameters, the user selects the required trajectory model, which invokes its object to calculate the path coordinates of the movable anchor node. Then, it will be written to an external scenario file.
3) The results and statistics will appear on the screen and be recorded in an external Excel file.
4) After the user presses the *start simulation* button, the coordinates of the anchor's trajectory will be read by opening the scenario file through the simulation thread.
5) The thread that draws the objects on the animation area runs over and over again while the anchor moves around the sensing field.
6) The simulation stops after the movable anchor node arrives at the end of the trajectory.

4. FRAMEWORK INTERFACE DESIGN

The design of the user interface is considered an essential part of the overall software design process. The following are the design decisions that were considered:

1) **User familiarity:** the user interface should consider some concepts in light of the experience of the users who will make the most use of the proposed framework.
2) **Consistency:** The user interface has a consistent design where comparable operations can be invoked in the same way.
3) **Recoverability:** the design of the user interface supports recoverability, so it can recover from errors.

4) **User feedback:** the interface has helpful mechanisms that guide the user when errors occur.



WSNASG: WSN Animator and NS2-Based Scenario Generator Tool for Mobile Robot Trajectory

Simulation Settings

Generate/Read Scenario

● Generate new scenario ○ Read existing scenario

Obstacles Simulation

Deploy Obstacles: ☐ Fill Obstacles: ☐

Length: 60 m Width: 30 m

Number of Horizontal Segments: 10 R

Trajectory Resolution: 50 m

Anchor Mobility Speed: 10 m/sec

Total Number of Nodes: 10 sensor

Simulation Duration: 600 sec

AnchorRadius = 50 m

Sensor Radius = 50 m

Select Curve Level

Only for HILBERT, Z_CURVE and Hexagon

SQUARE_SPIRAL

Trajectory Model Results:

Deployment Area = 500 x 500 m^2

Number of Spiral Turns = 5.25

Anchor Initial Position = (251 ,251)

Trajectory Length = 6000 m

Simulation Time = 600 sec

3 %

Elapsed Time: 00:18.489

Copyright (C): Dr. Abdelhady Naguib

Figure 6. Proposed Framework Interface

Figure 6 presents the main screen of the proposed framework. As shown, the interface is composed of two parts: the first part on the left is the drawing (animation) area where sensor devices and the movable anchor node are deployed. The second part on the right is concerned with simulation settings and a data entry and selection panel where the user can change and select simulation data. The bottom part of the simulation settings panel shows the trajectory model results. The user can select either the *Generate new scenario* or *Read existing scenario* radio buttons. By default, the *Generate new scenario* radio button is selected initially. The user can change the following parameters*: Number of horizontal segments*, *Trajectory resolution*, *Anchor mobility speed*, *Total number of sensor nodes,* and other settings. After that, the results of the selected trajectory model will be displayed in the lower right corner of the *simulation settings* panel. The displayed results differ according to the chosen trajectory model, but the common results among all trajectory models are as follows: *Deployment area size*, *Total trajectory length*, and *Simulation time*. Afterward, the user can press the *Deploy WSN* button, and the selected trajectory model will be displayed on the monitoring area. Finally, the user can press the *Start Simulation* button to begin a movement of the movable anchor across the drawn trajectory. If the user needs to run an existing scenario, he can click on the *Read existing scenario* radio button.

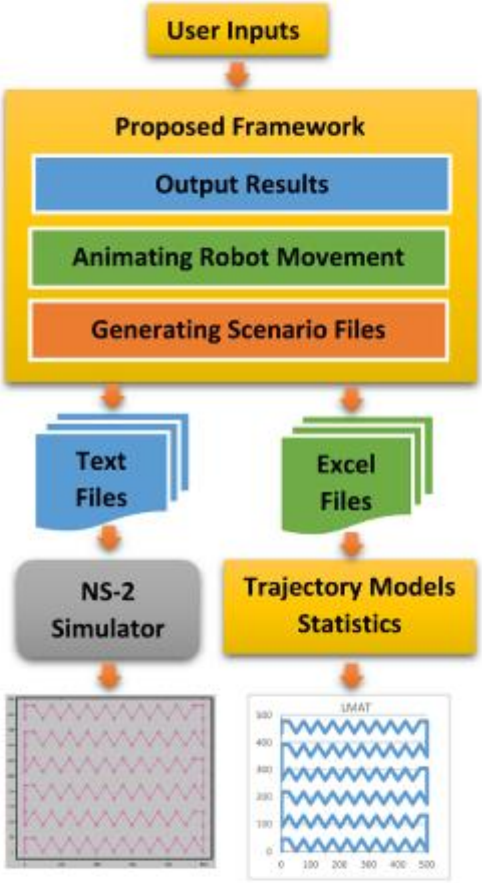
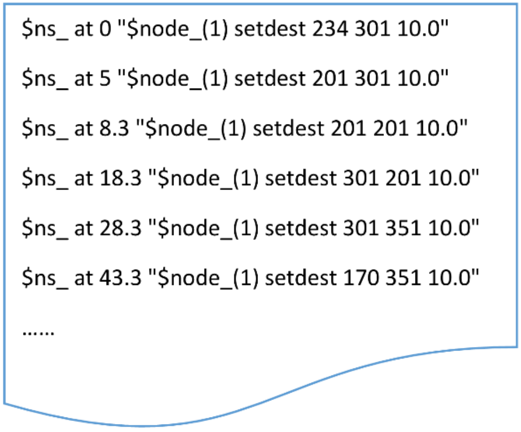


Figure 7. Block Diagram of the Proposed Framework

The block diagram of the proposed framework is shown in Figure 7. The proposed framework can generate simulation results in two ways: the first is by writing the results on the screen of the proposed framework, and the second is via two Excel files. The first Excel file can be used for calculating more statistics about the results gained, and the second Excel file can be used for drawing a directed graph of the selected trajectory model.



```
$ns_ at 0 "$node_(1) setdest 234 301 10.0"

$ns_ at 5 "$node_(1) setdest 201 301 10.0"

$ns_ at 8.3 "$node_(1) setdest 201 201 10.0"

$ns_ at 18.3 "$node_(1) setdest 301 201 10.0"

$ns_ at 28.3 "$node_(1) setdest 301 351 10.0"

$ns_ at 43.3 "$node_(1) setdest 170 351 10.0"

......
```

Figure 8. Generated Scenario File (Sample of Commands)

The proposed framework is capable of generating two scenario files (text files): the first one for storing the topology of the WSN for the purpose of comparing multiple scenarios, and the second text file for storing the coordinates of the movable anchor's trajectory that can be used as input to the NS-2 simulator. This file contains the commands (shown in Figure 8) needed by NS-2 to simply move the movable anchor node according to the given trajectory model.

5. RESULTS VERIFICATION AND DISCUSSIONS

Many researchers face difficulty in generating motion files for the movable anchor node for various types of fixed path algorithms. This section demonstrates how the proposed framework solves the problem stated by applying an experimental setup and then verifying the obtained results. With this proposed framework, researchers can make their own scenario files of how the movable anchor node might move based on different static path planning mechanisms. Afterward, the generated scenario file can be used by users in the network simulator (NS-2). In the following, an experimental setup for different static path planning mechanisms (case studies) will be conducted.

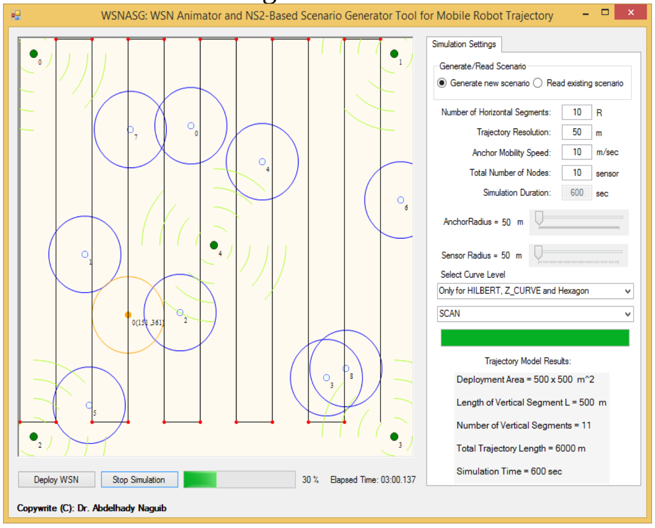
5.1. SCAN Trajectory Model

The following table shows the experimental setups used for simulating SCAN path planning mechanism.

Table 1. SCAN Experimental Setup

Input Parameters	**Value**
Number of Horizontal Segments (R)	10
Trajectory Resolution (m)	50
Anchor Mobility Speed (m/Sec)	10
Total Number of Nodes	10

The next figure shows what the proposed framework looked like after 3 minutes of simulation. As shown in Figure 9, the trajectory of the SCAN model has been drawn on the monitoring area as a series of vertical and horizontal segments in black lines, and there are nine sensor nodes as blue (hollow) nodes and the movable anchor shown by an orange (solid) node. There are five nodes: four of them lie at the four corners, and one lies in the middle of the monitoring area. These nodes are called *Base Stations* for negotiation and data dissemination.


WSNASG: WSN Animator and NS2-Based Scenario Generator Tool for Mobile Robot Trajectory
0
1
7
0
4
6
1
4
0(151,361)
2
3
8
5
2
3
Simulation Settings
Deploy WSN
Stop Simulation

Generate/Read Scenario
● Generate new scenario ○ Read existing scenario

Number of Horizontal Segments: 10 R
Trajectory Resolution: 50 m
Anchor Mobility Speed: 10 m/sec
Total Number of Nodes: 10 sensor
Simulation Duration: 600 sec
AnchorRadius = 50 m
Sensor Radius = 50 m
Select Curve Level
Only for HILBERT, Z_CURVE and Hexagon
SCAN

Trajectory Model Results:
Deployment Area = 500 x 500 m^2
Length of Vertical Segment L = 500 m
Number of Vertical Segments = 11
Total Trajectory Length = 6000 m
Simulation Time = 600 sec

30 % Elapsed Time: 03:00.137

Copywrite (C): Dr. Abdelhady Naguib

Figure 9. Simulation of SCAN Trajectory Model

The output of this SCAN mechanism is represented in the following table. There are two kinds of generated files: Excel files and text files. Figure 10 represents the chart of the SCAN trajectory model that was sketched using the generated Excel file.

Table 2. Simulation Results of the SCAN Mechanism

Results	**Value**
Area Size (m^2)	500 x 500
Length of Vertical Segment (m)	500
Number of Vertical Segments	11
Total Trajectory Length (m)	6000
Simulation Time (Sec)	600

The chart shown in Figure 10 is considered evidence that the generated SCAN trajectory in Figure 9 is correct. The other text (scenario) file that was made after running the SCAN mechanism is shown in the figure below. This scenario file is used by the NS-2 simulator to simulate how the movable anchor node moves.

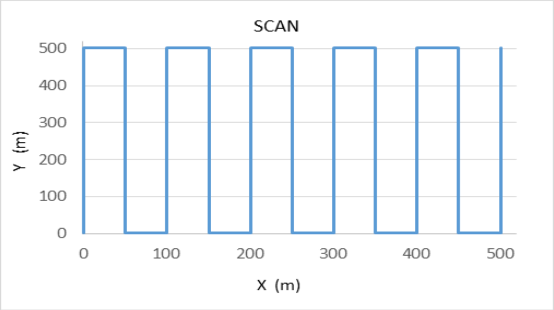
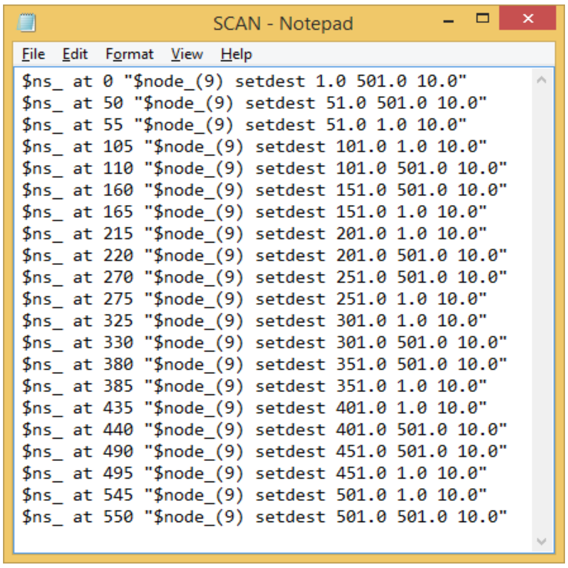


Figure 10. Chart of the SCAN Trajectory

The generated scenario file from the proposed framework in Figure 11 was applied to the NS-2 simulator to ensure that the results of the proposed framework were correct.



```
$ns_ at 0 "$node_(9) setdest 1.0 501.0 10.0"
$ns_ at 50 "$node_(9) setdest 51.0 501.0 10.0"
$ns_ at 55 "$node_(9) setdest 51.0 1.0 10.0"
$ns_ at 105 "$node_(9) setdest 101.0 1.0 10.0"
$ns_ at 110 "$node_(9) setdest 101.0 501.0 10.0"
$ns_ at 160 "$node_(9) setdest 151.0 501.0 10.0"
$ns_ at 165 "$node_(9) setdest 151.0 1.0 10.0"
$ns_ at 215 "$node_(9) setdest 201.0 1.0 10.0"
$ns_ at 220 "$node_(9) setdest 201.0 501.0 10.0"
$ns_ at 270 "$node_(9) setdest 251.0 501.0 10.0"
$ns_ at 275 "$node_(9) setdest 251.0 1.0 10.0"
$ns_ at 325 "$node_(9) setdest 301.0 1.0 10.0"
$ns_ at 330 "$node_(9) setdest 301.0 501.0 10.0"
$ns_ at 380 "$node_(9) setdest 351.0 501.0 10.0"
$ns_ at 385 "$node_(9) setdest 351.0 1.0 10.0"
$ns_ at 435 "$node_(9) setdest 401.0 1.0 10.0"
$ns_ at 440 "$node_(9) setdest 401.0 501.0 10.0"
$ns_ at 490 "$node_(9) setdest 451.0 501.0 10.0"
$ns_ at 495 "$node_(9) setdest 451.0 1.0 10.0"
$ns_ at 545 "$node_(9) setdest 501.0 1.0 10.0"
$ns_ at 550 "$node_(9) setdest 501.0 501.0 10.0"
```

Figure 11. SCAN Scenario File

Figure 12 is considered evidence of the correctness of the proposed framework results. It represents a screenshot from NS-2 running the SCAN trajectory. The area is 500 x 500 m^2, the length of a vertical segment is 500 m, and there are 11 vertical segments, as shown. These results are compatible with the results gained from the proposed framework in Table 2.

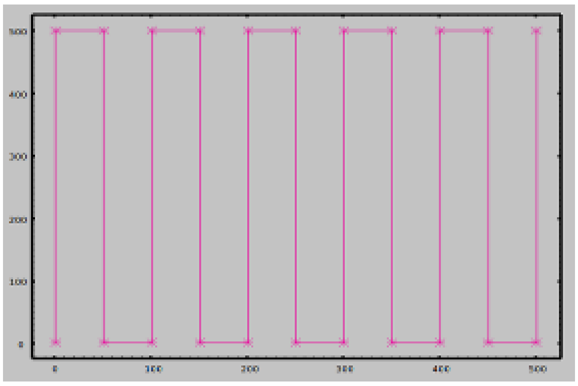


Figure 12. Screenshot of NS-2 Running SCAN Trajectory

5.2. HILBERT Trajectory Model

In this experiment, the HILBERT trajectory model has been simulated according to the parameters shown in Table 3.

Table 3. HILBERT Experimental Setup

Input Parameters	**Value**
Trajectory Resolution (m)	35
Anchor Mobility Speed (m/Sec)	10
Total Number of Nodes	5
Curve Level	4

The shape of the HILBERT path (level 4) shown in Figure 13 is generated after applying the parameters in Table 3. The anchor moves at a speed of 10 m/sec and reaches the middle of the monitoring area after 11 minutes (i.e., 71.0%) of simulation time.

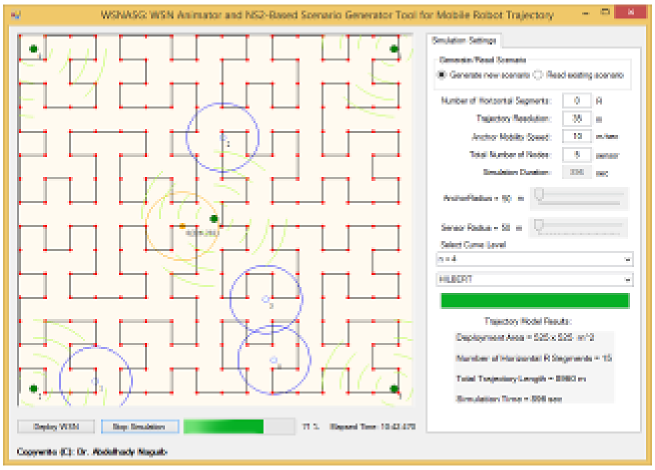


Figure 13. Simulation of the HILBERT Trajectory Model

The following table presents the results when simulating the HILBERT mechanism. As shown, the deployment field is 525 x 525 m^2, which is approximately equal to the area of the SCAN model, but the total trajectory length of the HILBERT model is 8960 m, which is longer than the path of the SCAN model due to the many turns of the HILBERT model.

Table 4. Simulation Results of the HILBERT Mechanism

Results	**Value**
Area Size (m^2)	525 x 525
Number of Horizontal Segment	15
Total Trajectory Length (m)	8960
Simulation Time (Sec)	896

From Figure 14, the generated HILBERT path from the Excel file proves that it is compatible with the trajectory present in Figure 13. The following figure represents the scenario file for the movement of the anchor node, which can be used in the NS-2.

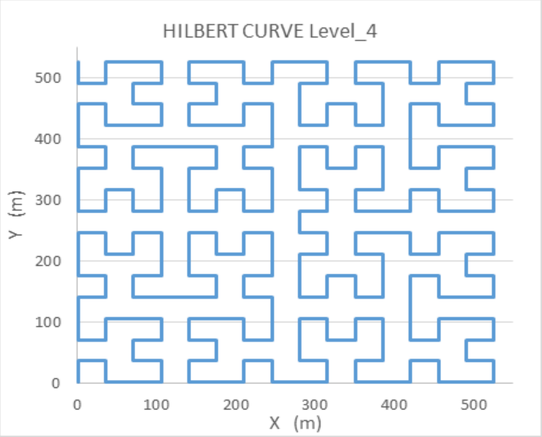
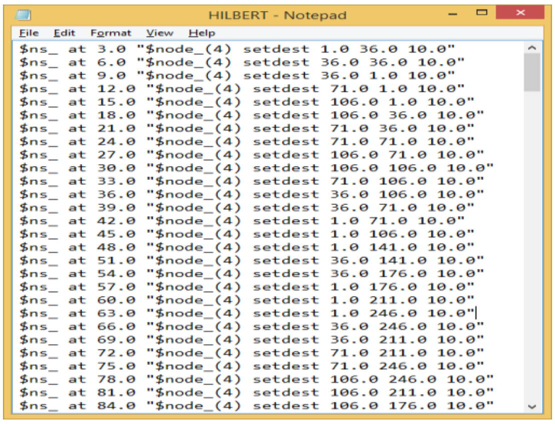


Figure 14. Chart of the HILBERT Trajectory

In order to verify the generated HILBERT scenario file (Figure 15), it has been used by the NS-2, and the resulting trajectory from this simulation is shown in Figure 16. The trajectory represented in Figure 16 is the same as the one in Figure 13.



```
$ns_ at 3.0 "$node_(4) setdest 1.0 36.0 10.0"
$ns_ at 6.0 "$node_(4) setdest 36.0 36.0 10.0"
$ns_ at 9.0 "$node_(4) setdest 36.0 1.0 10.0"
$ns_ at 12.0 "$node_(4) setdest 71.0 1.0 10.0"
$ns_ at 15.0 "$node_(4) setdest 106.0 1.0 10.0"
$ns_ at 18.0 "$node_(4) setdest 106.0 36.0 10.0"
$ns_ at 21.0 "$node_(4) setdest 71.0 36.0 10.0"
$ns_ at 24.0 "$node_(4) setdest 71.0 71.0 10.0"
$ns_ at 27.0 "$node_(4) setdest 106.0 71.0 10.0"
$ns_ at 30.0 "$node_(4) setdest 106.0 106.0 10.0"
$ns_ at 33.0 "$node_(4) setdest 71.0 106.0 10.0"
$ns_ at 36.0 "$node_(4) setdest 36.0 106.0 10.0"
$ns_ at 39.0 "$node_(4) setdest 36.0 71.0 10.0"
$ns_ at 42.0 "$node_(4) setdest 1.0 71.0 10.0"
$ns_ at 45.0 "$node_(4) setdest 1.0 106.0 10.0"
$ns_ at 48.0 "$node_(4) setdest 1.0 141.0 10.0"
$ns_ at 51.0 "$node_(4) setdest 36.0 141.0 10.0"
$ns_ at 54.0 "$node_(4) setdest 36.0 176.0 10.0"
$ns_ at 57.0 "$node_(4) setdest 1.0 176.0 10.0"
$ns_ at 60.0 "$node_(4) setdest 1.0 211.0 10.0"
$ns_ at 63.0 "$node_(4) setdest 1.0 246.0 10.0"
$ns_ at 66.0 "$node_(4) setdest 36.0 246.0 10.0"
$ns_ at 69.0 "$node_(4) setdest 36.0 211.0 10.0"
$ns_ at 72.0 "$node_(4) setdest 71.0 211.0 10.0"
$ns_ at 75.0 "$node_(4) setdest 71.0 246.0 10.0"
$ns_ at 78.0 "$node_(4) setdest 106.0 246.0 10.0"
$ns_ at 81.0 "$node_(4) setdest 106.0 211.0 10.0"
$ns_ at 84.0 "$node_(4) setdest 106.0 176.0 10.0"
```

Figure 15. HILBERT Scenario File

It can be concluded that the output of the proposed framework gives the same output as the NS-2. As shown in Figure 16, the area size is 525 x 525 m^2 and the number of horizontal segments is 15. Hence, these results are considered evidence for the correctness of the proposed framework results.

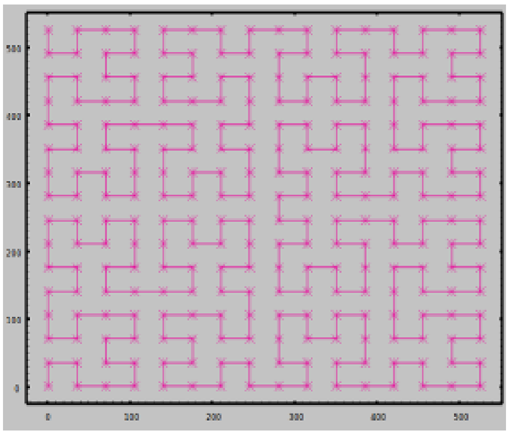


Figure 16. Screenshot of NS-2 Running the HILBERT

5.3. SPIRAL Trajectory Model

In this experiment, the SPIRAL trajectory model has been simulated according to the parameters shown in Table 5.

Table 5. SPIRAL Experimental Setup

Input Parameters	**Value**
Number of Horizontal Segments	10
Trajectory Resolution (m)	50
Anchor Mobility Speed (m/Sec)	8
Total Number of Nodes	6

As shown in Figure 17, the movable anchor node reaches coordinate of the point (376, 282) after 2 minutes of simulation at a speed of 8 m/sec.

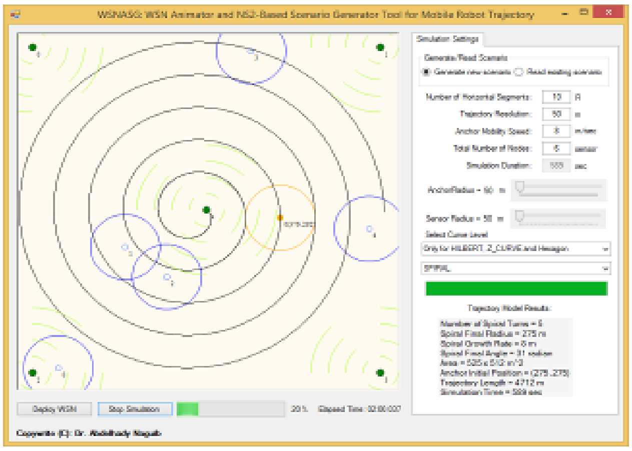


Figure 17. Simulation of the SPIRAL Trajectory Model

The results of the SPIRAL trajectory model are shown in Table 6. As shown, the movable anchor node started moving from its initial position in the middle of the monitoring area, and after 589 seconds (the end of the simulation), it traversed a distance of 4712 m. The traversed distance of the movable anchor node for the SPIRAL mechanism is smaller than SCAN and HILBERT, respectively.

Table 6. Simulation Results of the SPIRAL Mechanism

Results	**Value**
Number of Spiral Turns	5
Spiral Final Radius (m)	275
Spiral Growth Rate (m)	8
Spiral Final Angle (rad)	31
Area Size (m^2)	525 x 512
Anchor Initial Position (x, y)	(275, 275)
Total Trajectory Length (m)	4712
Simulation Time (sec)	589

The chart representing SPIRAL trajectory in Figure 18 shows that the monitoring area size equals 525 x 512 m^2, the initial position of the movable anchor is at point (275, 275), and the final radius of SPIRAL trajectory is 275 m. These observations are compatible with the results shown in Table 6.

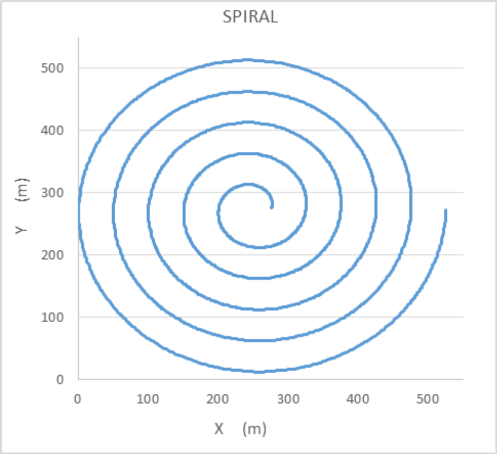
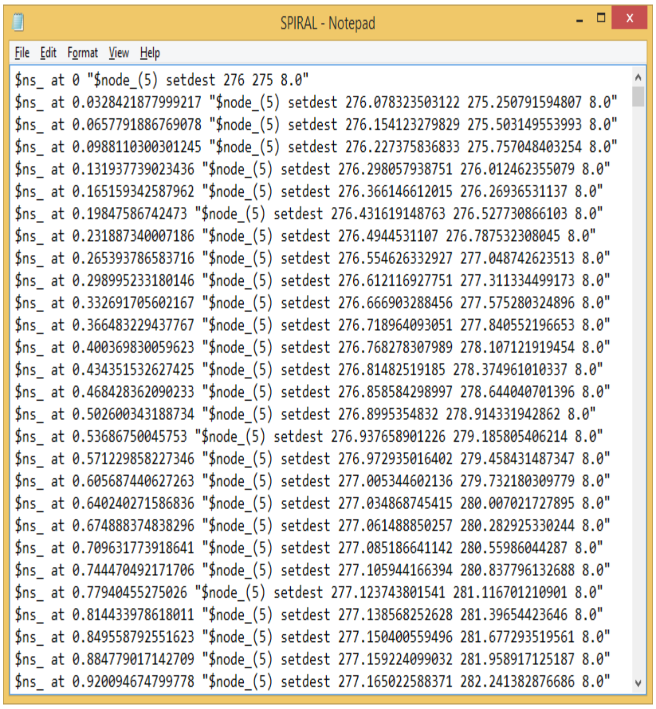


Figure 18. Chart of the SPIRAL Trajectory

The chart of SPIRAL trajectory shown in Figure 18 is the same as the trajectory drawn in the user interface of Figure 17. Note that the origin point in the coordinate system of the proposed framework lies at the upper-left corner of the monitoring area. Hence, the increment in the y-axis goes down differently than in the well-known coordinate system. Therefore, the trajectory in Figure 17 is a reflected image of the trajectory in Figure 18 This notice applies to all trajectories generated from the proposed framework but does not affect the results of the NS-2.



```
$ns_ at 0 "$node_(5) setdest 276 275 8.0"
$ns_ at 0.0328421877999217 "$node_(5) setdest 276.078323503122 275.250791594807 8.0"
$ns_ at 0.0657791886769078 "$node_(5) setdest 276.154123279829 275.503149553993 8.0"
$ns_ at 0.0988110300301245 "$node_(5) setdest 276.227375836833 275.757048403254 8.0"
$ns_ at 0.131937739023436 "$node_(5) setdest 276.298057938751 276.012462355079 8.0"
$ns_ at 0.165159342587962 "$node_(5) setdest 276.366146612015 276.26936531137 8.0"
$ns_ at 0.19847586742473 "$node_(5) setdest 276.431619148763 276.527730866103 8.0"
$ns_ at 0.231887340007186 "$node_(5) setdest 276.4944531107 276.787532308045 8.0"
$ns_ at 0.265393786583716 "$node_(5) setdest 276.554626332927 277.048742623513 8.0"
$ns_ at 0.298995233180146 "$node_(5) setdest 276.612116927751 277.311334499173 8.0"
$ns_ at 0.332691705602167 "$node_(5) setdest 276.666903288456 277.575280324896 8.0"
$ns_ at 0.366483229437767 "$node_(5) setdest 276.718964093051 277.840552196653 8.0"
$ns_ at 0.400369830059623 "$node_(5) setdest 276.768278307989 278.107121919454 8.0"
$ns_ at 0.434351532627425 "$node_(5) setdest 276.81482519185 278.374961010337 8.0"
$ns_ at 0.468428362090233 "$node_(5) setdest 276.858584298997 278.644040701396 8.0"
$ns_ at 0.502600343188734 "$node_(5) setdest 276.8995354832 278.914331942862 8.0"
$ns_ at 0.53686750045753 "$node_(5) setdest 276.937658901226 279.185805406214 8.0"
$ns_ at 0.571229858227346 "$node_(5) setdest 276.972935016402 279.458431487347 8.0"
$ns_ at 0.605687440627263 "$node_(5) setdest 277.005344602136 279.732180309779 8.0"
$ns_ at 0.640240271586836 "$node_(5) setdest 277.034868745415 280.007021727895 8.0"
$ns_ at 0.674888374838296 "$node_(5) setdest 277.061488850257 280.282925330244 8.0"
$ns_ at 0.709631773918641 "$node_(5) setdest 277.085186641142 280.5598604287 8.0"
$ns_ at 0.744470492171706 "$node_(5) setdest 277.105944166394 280.837796132688 8.0"
$ns_ at 0.77940455275026 "$node_(5) setdest 277.12374380154 1281.116701210901 8.0"
$ns_ at 0.814433978618011 "$node_(5) setdest 277.138568252628 281.39654423646 8.0"
$ns_ at 0.849558792551623 "$node_(5) setdest 277.150400559496 281.677293519561 8.0"
$ns_ at 0.884779017142709 "$node_(5) setdest 277.159224099032 281.958917125187 8.0"
$ns_ at 0.920094674799778 "$node_(5) setdest 277.165022588371 282.241382876686 8.0"
```

Figure 19. SPIRAL Scenario File

The scenario file for the SPIRALL trajectory shown in Figure 19 contains the coordinates of the movable anchor at certain times. As shown in Figure 19, the difference between two successive points on the SPIRAL trajectory is very small. Thus, the motion of the anchor node gives an exact spiral shape, which is an indication that the proposed framework gives high-accuracy results.

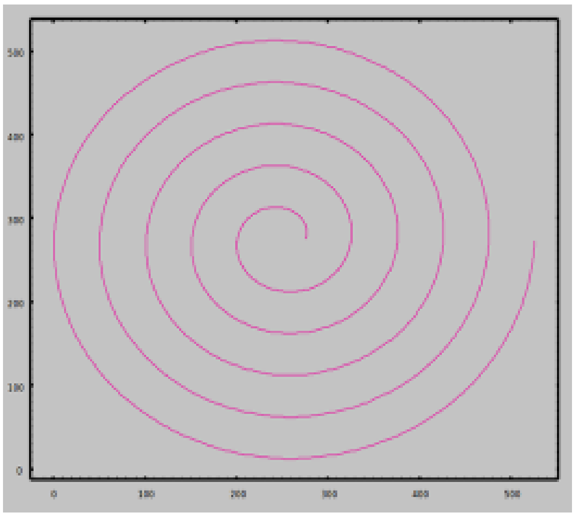


Figure 20. NS-2 Screenshot Running the SPIRAL Trajectory

The SPIRAL shape shown in Figure 20 results from the NS-2 simulator, and as shown, it is compatible with Figures 17 and 18. Hence, it is evidence that the results of the proposed framework have been verified.

5.4. Comparing the Efficiency of the Proposed Framework against other Tools

In this section, the novelty of the proposed framework has been clarified. In the following we a brief description of how this framework differs from existing WSN simulation tools. A comparison was made between the proposed framework and related systems [26 – 29]. As shown in Table 7, the proposed framework has a very good design for the user interface but the other tools fall between bad and good. According to robot movement, the proposed framework supports trace files for robot movement, which leads to outstanding performance among other tools. As for the last two metrics: generating scenarios and path models, the proposed framework supports configuration for different scenarios and can validate obtained results by running real time simulations for the movement files of many path planning models that already exist or can be added to the proposed framework.

Table 7. Comparison of the proposed framework against four related systems

Simulation Tools	**User Interface Design**	**Robot Movement**	**Generating Scenarios**	**Path Models**
Ali, Q. I. [26]	Poor	Good	No support	No support
Niewiadomska et al. [27]	Acceptable	Basic	No support	No support
Pagano et al. [28]	low	Average	No support	No support
Mozumdar et al. [29]	Good	Poor	No support	No support
The Proposed Framework	Very Good	Outstanding	Superior	Most models

As shown in Table 8, a quantitative evaluation is conducted based on eight different features this is to prove that the proposed model outperforms its other related tools. As shown, the proposed framework is an extendable system that supports adding new models to its platform as needed. The proposed framework supports interaction design which improves usability makes it easier and more intuitive to use and enhances user satisfaction. Flexibility was taken into account when designing the framework which means the system is able to adapt to changing requirements and technologies. The operability feature is supported by the proposed framework, meaning that the users can interact easily with the framework GUI and it has a good operability works. The proposed framework supports validity, ensuring that the framework meets users' expectations. Scalability feature is considered one of the most important characteristics of the framework which eliminates the burden of limiting future operations. Framework reliability is an essential feature of system quality ensuring it is free from failure. Finally, the simplicity of the proposed framework can significantly impact efficiency and performance making it easier to maintain, debug, and enhance the system over time. As shown in Table 8, the proposed framework outperforms related tools in all necessary features for any reliable system.

Table 8. Evaluation of the proposed framework based on eight features

Feature	Ali, Q. I. [26]	Niewiadomska et al. [27]	Pagano et al. [28]	Mozumdar et al. [29]	The Proposed Framework
Extendibility	✗	✗	✓	✓	✓
Interaction	✓	✓	✗	✗	✓
Flexibility	✓	✗	✓	✗	✓
Operability	✗	✗	✗	✗	✓
Validity	✓	✗	✓	✓	✓
Scalability	✗	✓	✗	✓	✓
Reliability	✓	✓	✓	✓	✓
Simplicity	✓	✗	✗	✗	✓

6. CONCLUSIONS AND FUTURE WORK

One difficulty that users face when performing simulations using NS-2 is how to create anchor motion files based on different path models. Therefore, this article proposes an outstanding framework for simulating anchor paths in wireless sensor networks and helping researchers not only generate different motion files but also modify them to simulate other static path algorithms. In this article, an introduction is made to the latest and most important static path planning mechanisms in WSNs that have been implemented in the proposed framework. There are several stages during the construction and implementation of the proposed framework, starting with analysis, design, experimental setup, and finally, verification. Broad simulations using the NS-2 have been done to verify the results, and the validation process proved the validity and accuracy of the results of the proposed framework. The proposed framework is an open source system that is available for download from the Code Project website. The proposed framework was compared with existing tools and the results proved the validity of the design and its superiority to them.

According to the future work, this work is currently underway to update the proposed framework to generate motion files recording the anchor's trajectory after avoiding obstacles of a variable size. These obstacles can be changed depending on the facilities given to the user. Also, the proposed framework is a prelude to future work, which is the development and implementation of a WSN localization simulator.

7. CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] Singh, Saurabh, et al., (2024) "Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions", *Journal of Ambient Intelligence and Humanized Computing*, 1-18.

[2] Kumar, Saureng & Sharma, Subhash. (2023) “Optimal Coverage Path Planning in a Wireless Sensor Network for Intelligent Transportation System”, *International journal of Computer Networks & Communications*, 15. 55-72. 10.5121/ijcnc.2023.15504.

[3] Güler, Samet., (2024) IEEE "Range-Based Wireless Sensor Network Localization by a Circumnavigating Mobile Anchor without Position Information.", *2024 32nd Mediterranean Conference on Control and Automation (MED).*

[4] Docquier, Théo, et al., (2023) "Performance evaluation methodologies for smart grid substation communication networks: A survey", *Computer Communications*, 198, 228-246.

[5] Razdan, Rahul, et al., (2023) "Polyverif: An open-source environment for autonomous vehicle validation and verification research acceleration", *IEEE Access*, 11, 28343-28354.

[6] Yago, Carmen María, and Francisco Javier Díez., (2023) "DESnets: A Graphical Representation for Discrete Event Simulation and Cost-Effectiveness Analysis", *Mathematics*, 11.7, 1602.

[7] Fonseca i Casas, Pau., (2023) "A continuous process for validation, verification, and accreditation of simulation models", *Mathematics*, 11.4, 845.

[8] Fathi, Sahand, Soheil Fathi, and Vahid Balali, (2023) "Time–space conflict management in construction sites using discrete event simulation (DES) and path Planning in unity", *Applied Sciences*, 13.14, 8128.

[9] Abbasi, Mohammad & Abdullah, A.. (2023) “Optimize the Network Coding Paths to Enhance the Coding Protection in Wireless Multimedia Sensor Network”, *International journal of Computer Networks & Communications*, 15. 17-36. 10.5121/ijcnc.2023.15502.

[10] Antonante, Pasquale, Heath G. Nilsen, and Luca Carlone, (2023) "Monitoring of perception systems: Deterministic, probabilistic, and learning-based fault detection and identification", *Artificial Intelligence* 325, 103998.

[11] Huang, Ziqi, et al, (2023) "Hybrid learning-based digital twin for manufacturing process: Modeling framework and implementation", *Robotics and Computer-Integrated Manufacturing* 82, 102545.

[12] Koutsonikolas, Dimitrios, Saumitra M. Das, and Y. Charlie Hu, (2007) "Path planning of mobile landmarks for localization in wireless sensor networks", *Computer Communications* 30.13, 2577-2592.

[13] Huang, Rui, and Gergely V. Zaruba, (2007) "Static path planning for mobile beacons to localize sensor networks", *Fifth annual IEEE international conference on pervasive computing and communications workshops (PerComW'07)*. IEEE, 2007.

[14] Hu, Zhen, et al, (2008) "Localization in wireless sensor networks using a mobile anchor node", *2008 IEEE/ASME international conference on advanced intelligent mechatronics*, IEEE, 2008.

[15] Han, Guangjie, et al, (2013) "Path planning using a mobile anchor node based on trilateration in wireless sensor networks", *Wireless Communications and Mobile Computing* 13.14, 1324-1336.

[16] Lin, Yueh Chen, and Jing Sin Liu, (2013) "The path planning algorithm of triangle grid scan for localization in wireless sensor network", *Applied Mechanics and Materials* 278, 1874-1877.

[17] Rezazadeh, Javad, et al, (2014) "Superior path planning mechanism for mobile beacon-assisted localization in wireless sensor networks", *IEEE Sensors Journal* 14.9, 3052-3064.

[18] Han, Guangjie, et al, (2014) "A mobile anchor assisted localization algorithm based on regular hexagon in wireless sensor networks", *The Scientific World Journal* 2014.1: 219371.

[19] Alomari, Abdullah, et al, (2018) "New path planning model for mobile anchor-assisted localization in wireless sensor networks", *Wireless Networks* 24: 2589-2607.

[20] Kannadasan, Kalidasan, et al, (2020) "M-curves path planning model for mobile anchor node and localization of sensor nodes using dolphin swarm algorithm", *Wireless Networks* 26.4: 2769-2783.

[21] Wu, Tian-qi, Min Yao, and Jian-hua Yang, (2016) "Dolphin swarm algorithm", *Frontiers of Information Technology & Electronic Engineering* 17.8: 717-729.

[22] Naguib, Abdelhady M., and Shahzad Ali, (2021) "A novel static path planning method for mobile anchor-assisted localization in wireless sensor networks", *International Journal of Sensors Wireless Communications and Control* 11.4: 482-493.

[23] Yildiz, Dogan, and Serap Karagol, (2021) "Path planning for mobile-anchor based wireless sensor networks localization: obstacle-presence schemes", *Sensors* 21.11: 3697.

[24] Blumenthal, Jan, et al, (2007) "Weighted centroid localization in zigbee-based sensor networks", *2007 IEEE international symposium on intelligent signal processing*. IEEE, 2007.

[25] Savvides, Andreas, Chih-Chieh Han, and Mani B. Strivastava, (2001) "Dynamic fine-grained localization in ad-hoc networks of sensors", *Proceedings of the 7th annual international conference on Mobile computing and networking*.

[26] Ali, Q. I. (2012) “Simulation framework of wireless sensor network (WSN) using matlab/simulink software”, *MATLAB-A Fundamental tool for scientific computing and engineering applications*, *2*(1), 263-284.

[27] Niewiadomska-Szynkiewicz, E., & Sikora, A. (2012) “A software tool for federated simulation of wireless sensor networks and mobile ad hoc networks”, In *Applied Parallel and Scientific Computing: 10th International Conference, PARA 2010, Reykjavík, Iceland, June 6-9, 2010, Revised Selected Papers, Part I 10* (pp. 303-313). Springer Berlin Heidelberg.

[28] Pagano, P., Chitnis, M., Lipari, G., Nastasi, C., & Liang, Y. (2009) “Simulating real-time aspects of wireless sensor networks”, *EURASIP Journal on Wireless Communications and Networking*, *2010*, 1-19.

[29] Mozumdar, M. M. R., Gregoretti, F., Lavagno, L., Vanzago, L., & Olivieri, S. (2008) “A framework for modeling, simulation and automatic code generation of sensor network application”, In *2008 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks* (pp. 515-522). IEEE.

AUTHOR

Abdelhady Naguib received his M.S. and Ph.D. in Systems and Computers Engineering from Al-Azhar University in 2008 and 2013 respectively. He is currently an associate professor at, Faculty of Engineering at Al-Azhar University, Cairo, Egypt. In addition, he is working as assistant professor at department of Computer Science, Jouf University, Kingdom of Saudi Arabia.

He has a research and teaching experience for many years. He is also acting as a reviewer of well-reputed journals like Springer Nature and Inderscience Online. His research area is in the field of Mobile Communications more specific in the area of mobile ad hoc and wireless sensor networks. He has contributed to this area by many papers and one book, “Extending NS-2 for Simulating a Localization Algorithm,” 1st. ed., LAP LAMBERT Academic Publishing, April 2001, ISBN-13 978-3-330-07674-7. Dr. Abdelhady Naguib is a member at Egyptian Syndicate for Engineers and has a membership of Cisco academy (as instructor).