PEGASIS in WSN

Software Recommended: NetSim Standard v11.0 64 bit, Visual Studio 2015/2017, MATLAB 64 bit

Project Download Link:

https://github.com/NetSim-TETCOS/PEGASIS_Routing_in_WSN_v11.0/archive/master.zip

Reference:

https://in.mathworks.com/matlabcentral/fileexchange/67504-pegasis-power-efficient-gathering-in-sensor-information-systems

In Wireless Sensor Networks energy efficiency plays a crucial role as the sensors are generally battery powered. Hierarchical routing protocols can be used to overcome this constraint.

PEGASIS – Power Efficient Gathering in Sensor Information Systems is one such hierarchical routing protocol which follows a chain based approach and a greedy algorithm. The sensor nodes organize themselves to form a chain. If any node dies in between then the chain is reconstructed to bypass the dead node. A leader or a cluster head node is assigned and it takes care of transmitting data to the base station/ sink node. The main goal of PEGASIS is to receive and transmit data to and from the neighbour and take turns being the cluster head for transmission to the Sink Node.
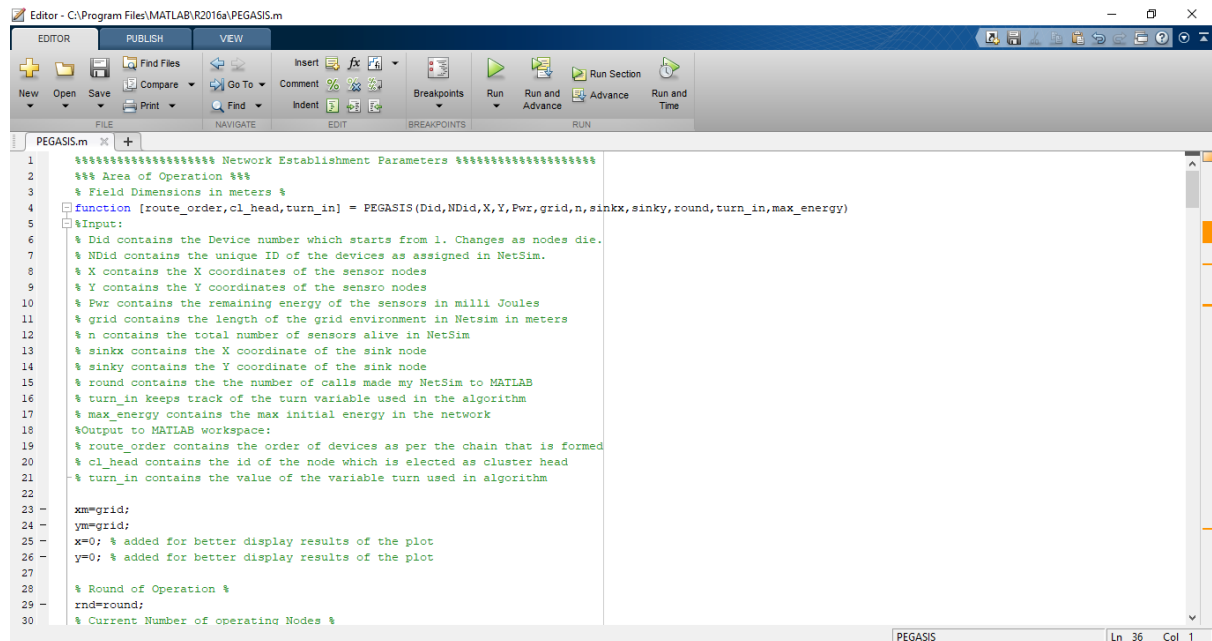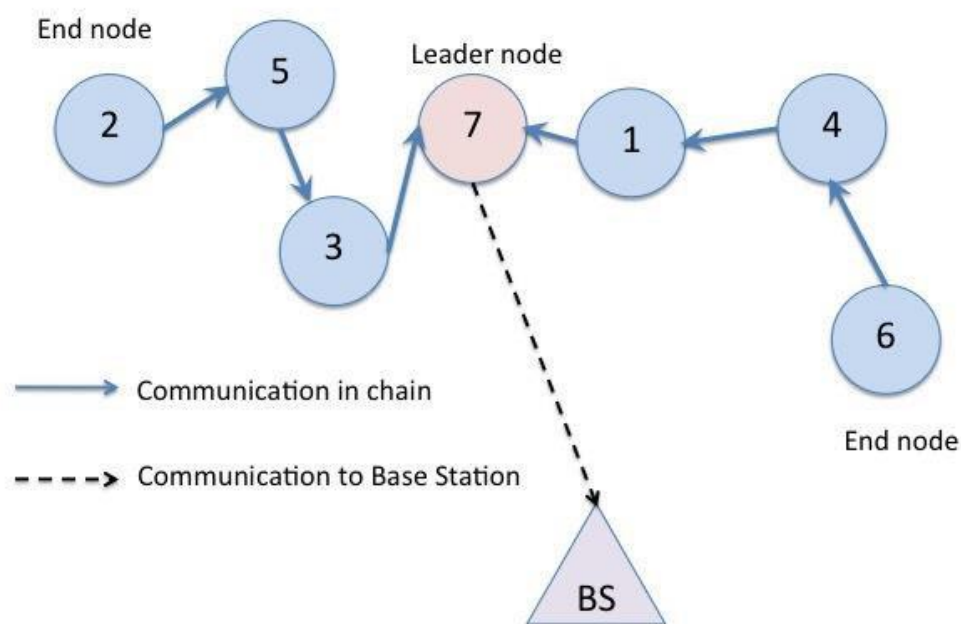

PEGASIS in NetSim with MATLAB Interfacing:

PEGASIS algorithm is implemented in NetSim by Interfacing with MATLAB for the purpose of mathematical calculation. The sensor coordinates are fed as input to MATLAB and PEGASIS algorithm that is implemented in MATLAB is used to dynamically form a chain between the nodes and to elect one of them as a head node.

From MATLAB the order of devices in the chain and the head node id is retrieved to perform routing in NetSim.

All the above steps are performed during each transmission and can be defined as per the implementation. Each time a node dies, the chain will be reconstructed. Also nodes take turn to become the head node in each iteration.
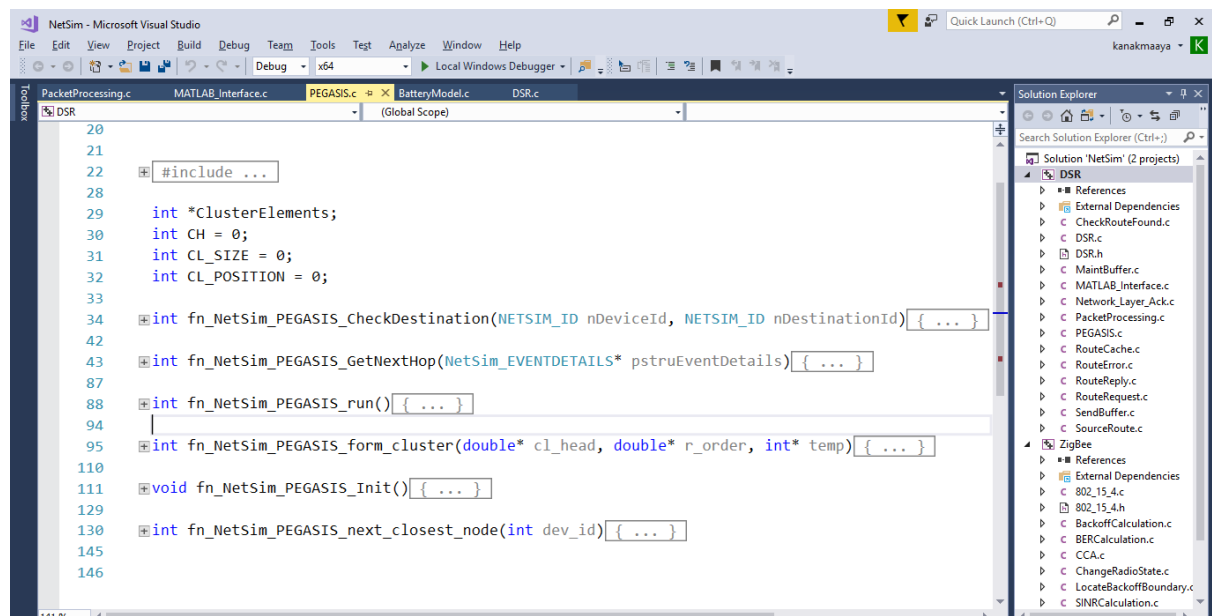
The codes required for the mathematical calculations done in MATLAB are written to a **PEGASIS.m** file as shown below:

A **PEGASIS.c** file is added to the DSR project which contains the following functions:

fn_NetSim_PEGASIS_CheckDestination()

This function is used to determine whether the current device is the destination of a packet or an intermediate node.

fn_NetSim_PEGASIS_GetNextHop()

This function handles routing in the sensor network by determining the next hop device based on the chain that is formed as part of PEGASIS protocol.

fn_NetSim_PEGASIS_run()

This function makes a call to MATLAB interfacing function and passes the inputs from NetSim to MATLAB and also retrieves the computed parameters from MATLAB workspace for further calculations in NetSim.

fn_NetSim_PEGASIS_form_clusters()

This function updates the information obtained from MATLAB to identify the head node and the neighbouring nodes in the PEGASIS chain.

fn_NetSim_PEGASIS_next_closest_node()

This function is used to identify the neighbouring node and next hop for routing packets.

fn_NetSim_PEGASIS_Init()

This function initializes the parameters specific to PEGASIS algorithm and identifies the number of sensors that are alive throughout the simulation.

Static Routing:

Static Routing is defined in such a way that the sensors send packets to neighbouring node in the PEASIS chain which is closest to the head node. Once packet arrives at the head node it is forwarded to the destination or the sinknode.
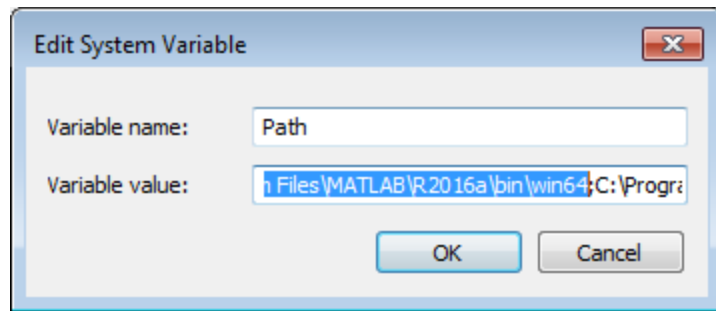
NOTE:

To run this code 64- bit version of MATLAB must be installed in your system.

Steps:

1. Open the Code folder and double click on the **NetSim.sln** file to open the project in Visual Studio.

2. Copy **PEGASIS.m** file present from the Code folder and place it in the root directory of MATLAB. For Eg: **"C:\Program Files\MATLAB\R2016a"**.

3. Create a user variable with the name of MATLAB_PATH and provide the path of the installation directory of user's respective MATLAB version.
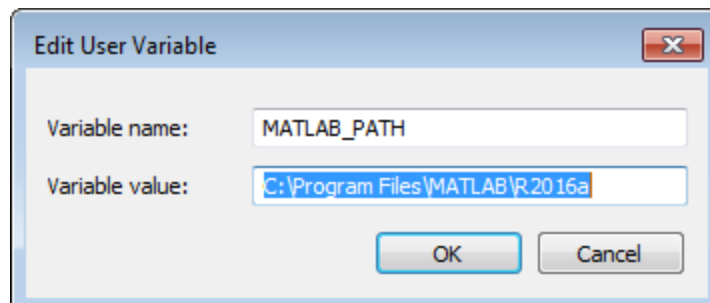

4. Make sure that the following directory is in the PATH(Environment variable)
   **<Path where MATLAB is installed>\bin\win64**

Note: If the machine has more than one MATLAB installed, the directory for the target platform must be ahead of any other MATLAB directory (for instance, when compiling a 64-bit application, the directory in the MATLAB 64-bit installation must be the first one on the PATH).

5. To register MATLAB as an automation server for NetSim to interact, start command prompt as an administrator and execute the command **matlab –regserver** as shown below:
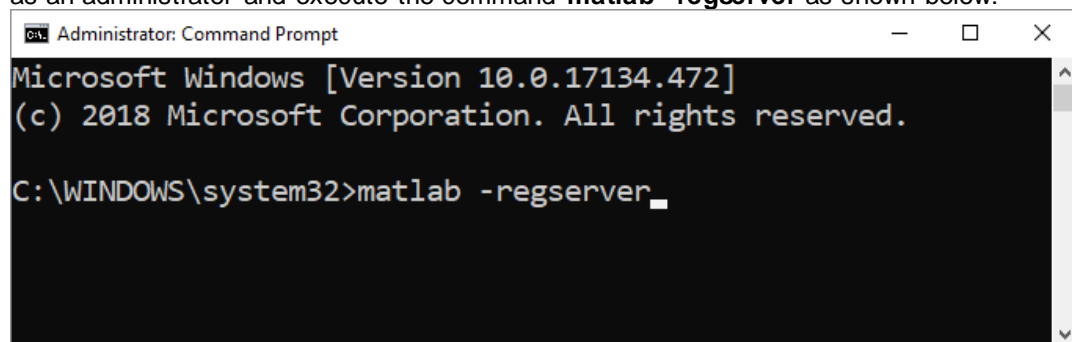

6. Right click on the DSR project in the solution explorer and select Rebuild.
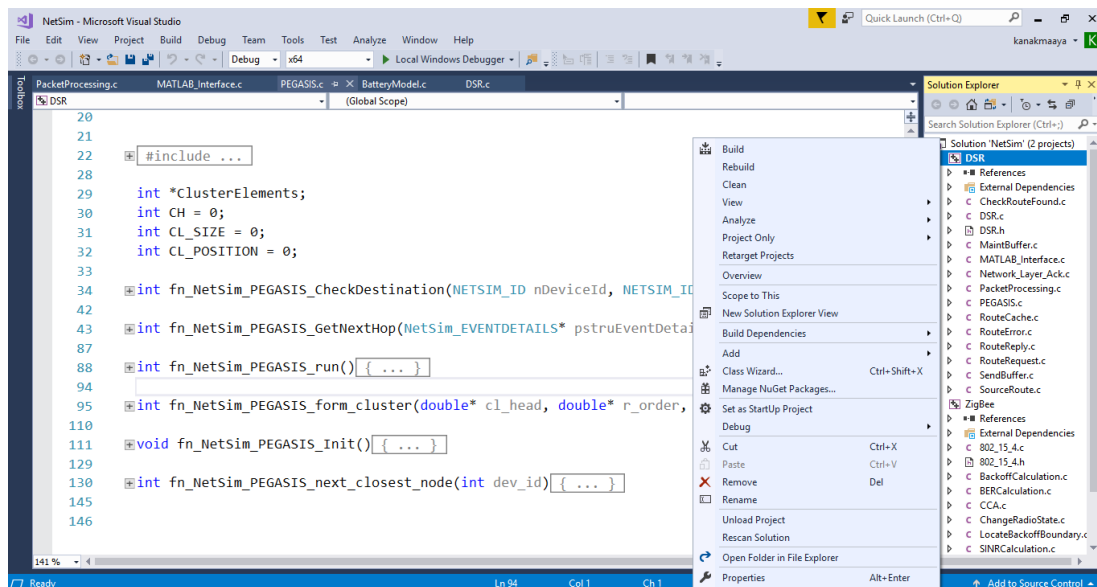

7. Similarly build the ZigBee project.

8. Copy the newly built **libDSR.dll** and **libZigBee.dll** from the DLL folder inside the Code Directory.

9. Replace the DLL's in the bin folder inside NetSim Installation Directory, after renaming the original libDSR.dll and libZigBee.dll suitably as a backup.

10. Run NetSim in administrator mode. Right click on NetSim shortcut in your desktop and select Run as Administrator.

11. Open the sample **Configuration.netsim** file present in the **Config_File** folder of this project
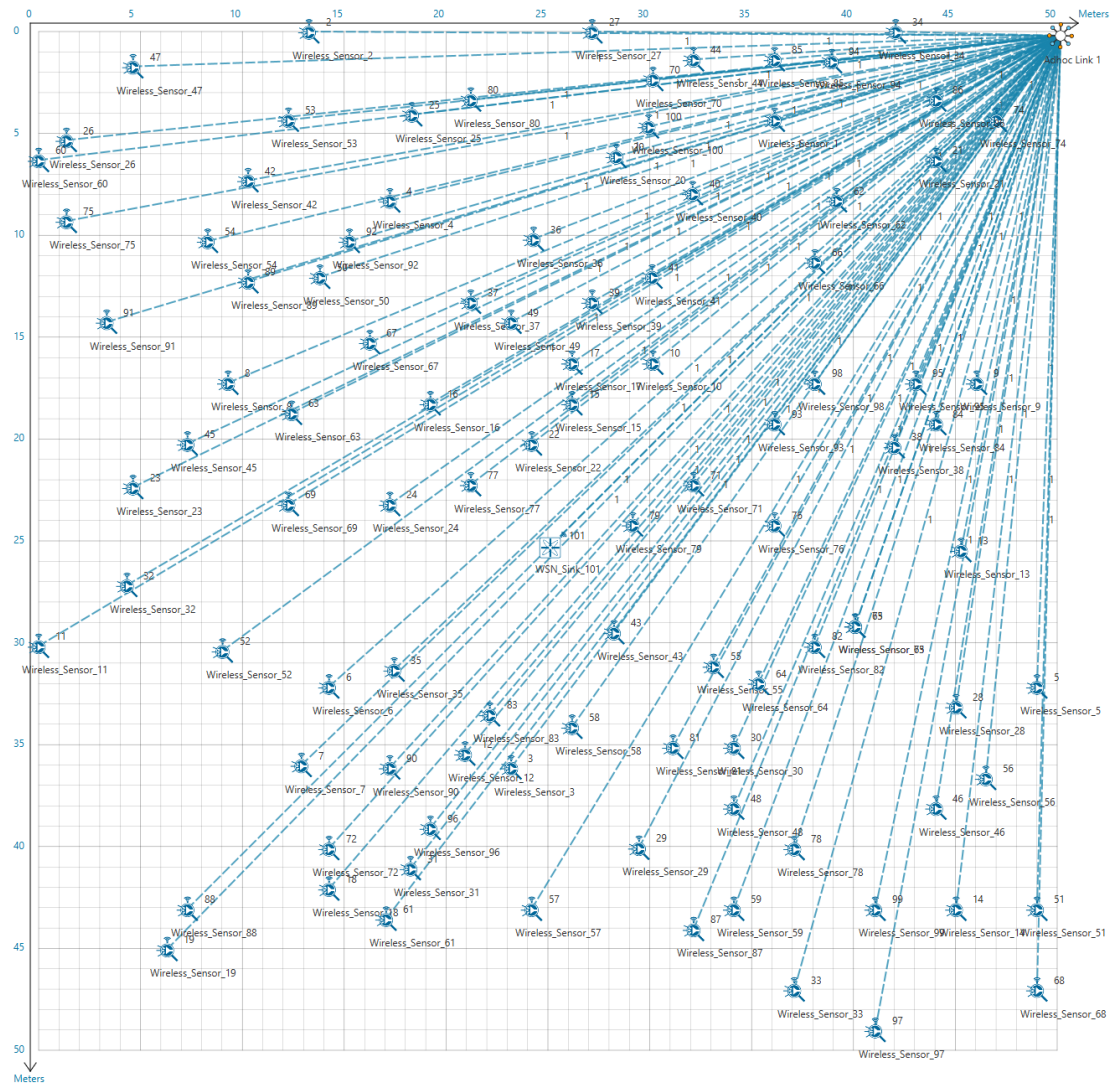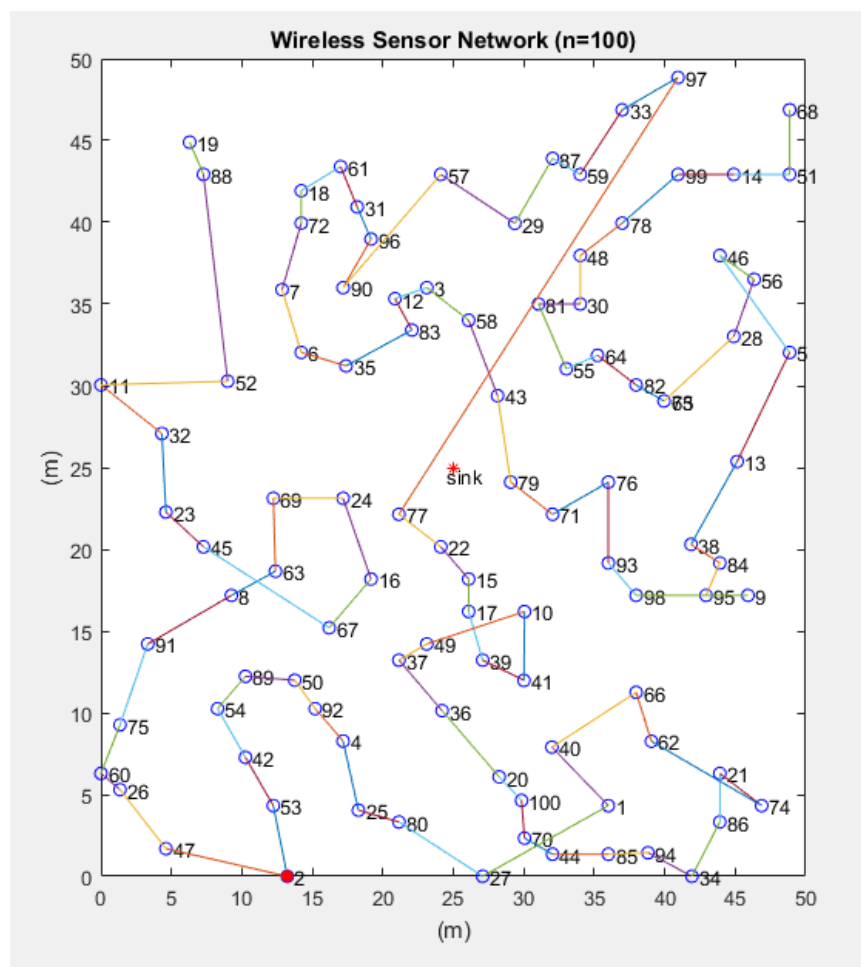

The above scenario consists of 100 sensors deployed randomly in a 50x50m grid environment with a sink node placed in the centre.

12. Run the Scenario. You will observe that as the simulation starts, NetSim automatically initializes MATLAB and the plots will be displayed as shown below:
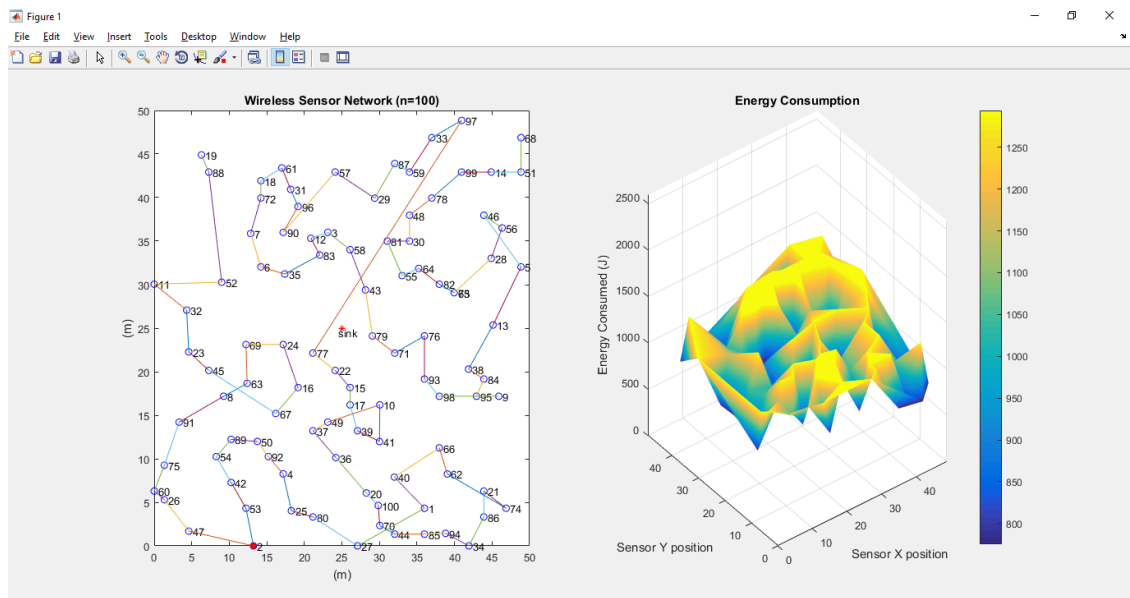
Explanation:

Plot 1:

The above plot contains the nodes that are part of the network in NetSim. All sensor nodes are initially alive. N in the plot indicates the total sensor nodes that are currently alive. Sink node is

highlighted in the centre and the node marked in red acts as the head node for the current transmission.

All nodes are connected to form a chain which is used to forward packets to the destination or the sink node. This chain may get reconstructed whenever one or more node runs out of energy and dies. The value of 'n' keeps getting updated during the simulation.

Plot 2:
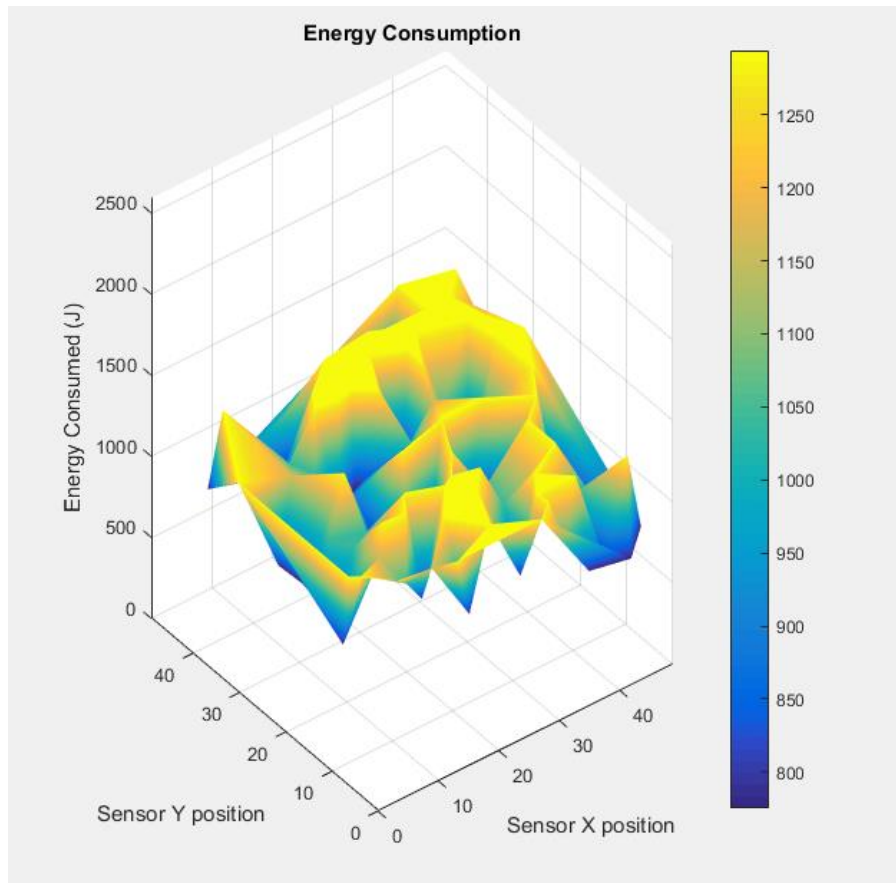
The above plot denotes the energy level in the sensor network. Sensors have different levels of energy remaining in the battery right from the simulation start. The z axis represents the power consumed while the sensors are placed on the x, y plane. As simulation progresses this plot gets updated dynamically based on energy consumed by the sensors in the network.