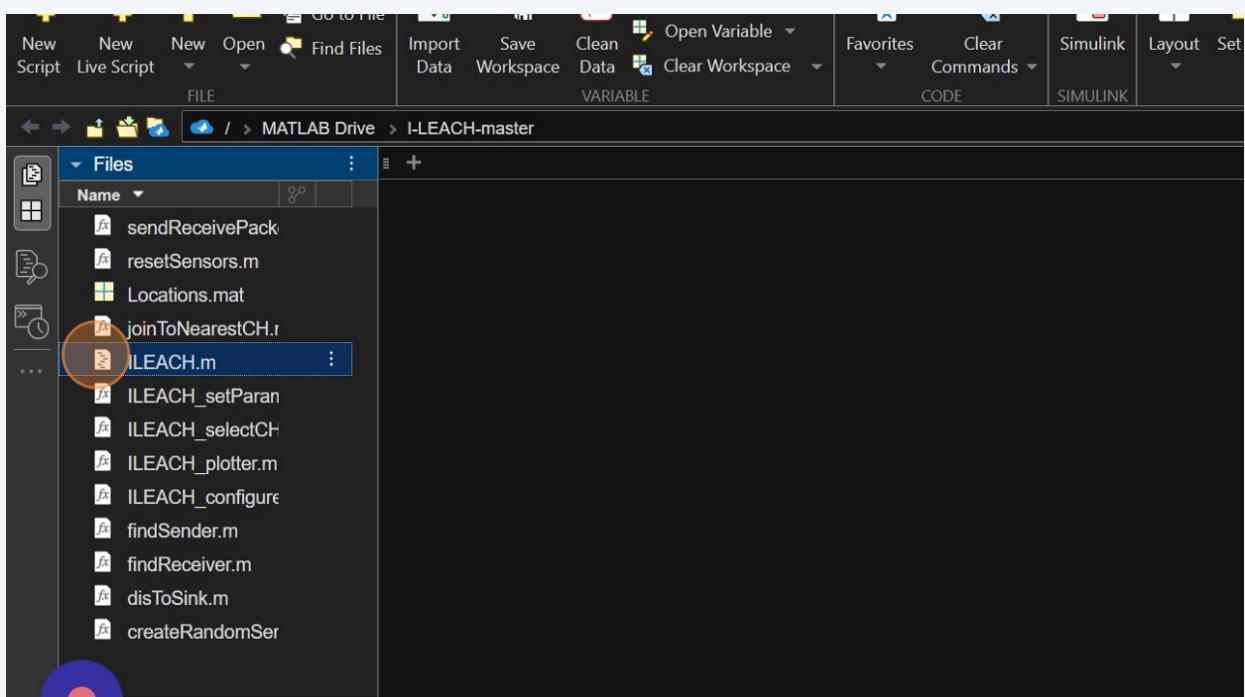


Step-by-Step Explanation of Implementing LEACH PROTOCOL in WSNs in MATLAB

Scribe

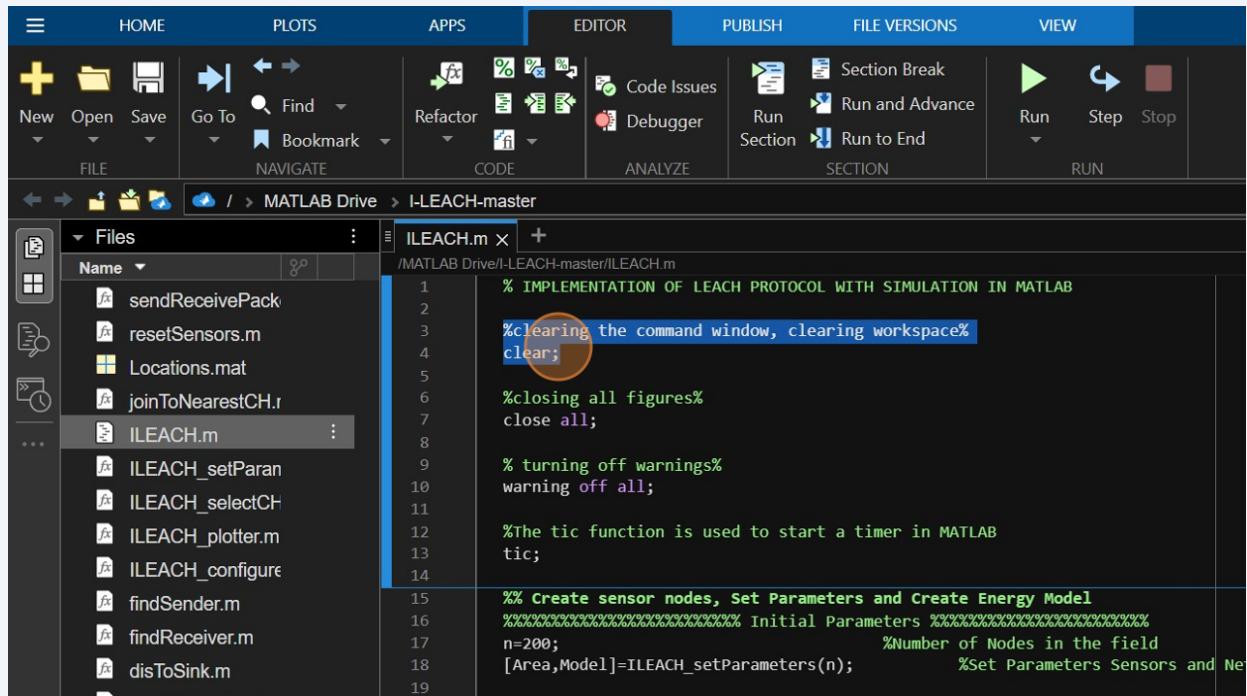
- 1 Navigate to <https://matlab.mathworks.com/>

- 2 Click "I-LEACH.m" This is the Main file to Run the Leach Protocol



LEACH (Low-Energy Adaptive Clustering Hierarchy) is a widely used hierarchical clustering protocol designed for wireless sensor networks to prolong network lifetime by efficiently managing energy consumption. In LEACH, nodes self-organize into clusters, and each cluster has a rotating cluster head (CH) selected periodically to distribute the energy consumption across the network. Nodes take turns serving as CHs to reduce the load on individual nodes and mitigate energy depletion. LEACH employs a probabilistic approach for CH selection, where nodes decide to become CHs based on a pre-defined probability. Cluster heads collect data from their member nodes and transmit aggregated data to a sink, facilitating energy-efficient communication. The periodic rotation of CHs and distributed data aggregation contribute to balancing energy usage and extending the overall network lifespan in resource-constrained sensor environments.

3 Firstly, the clear command written here is to clear the workspace in MATLAB



The screenshot shows the MATLAB IDE interface. The top menu bar includes HOME, PLOTS, APPS, EDITOR (selected), PUBLISH, FILE VERSIONS, and VIEW. Below the menu is a toolbar with icons for New, Open, Save, Go To, Find, Refactor, Debugger, Run Section, Run and Advance, Run to End, Run, Step, and Stop. The current workspace path is MATLAB Drive > I-LEACH-master. The left sidebar shows a file tree with files like sendReceivePack, resetSensors.m, Locations.mat, joinToNearestCH.r, ILEACH.m, ILEACH_setParan, ILEACH_selectCH, ILEACH_plotter.m, ILEACH_configure, findSender.m, findReceiver.m, disToSink.m. The main editor window displays the ILEACH.m script:

```
% IMPLEMENTATION OF LEACH PROTOCOL WITH SIMULATION IN MATLAB
%clearing the command window, clearing workspace%
clear;

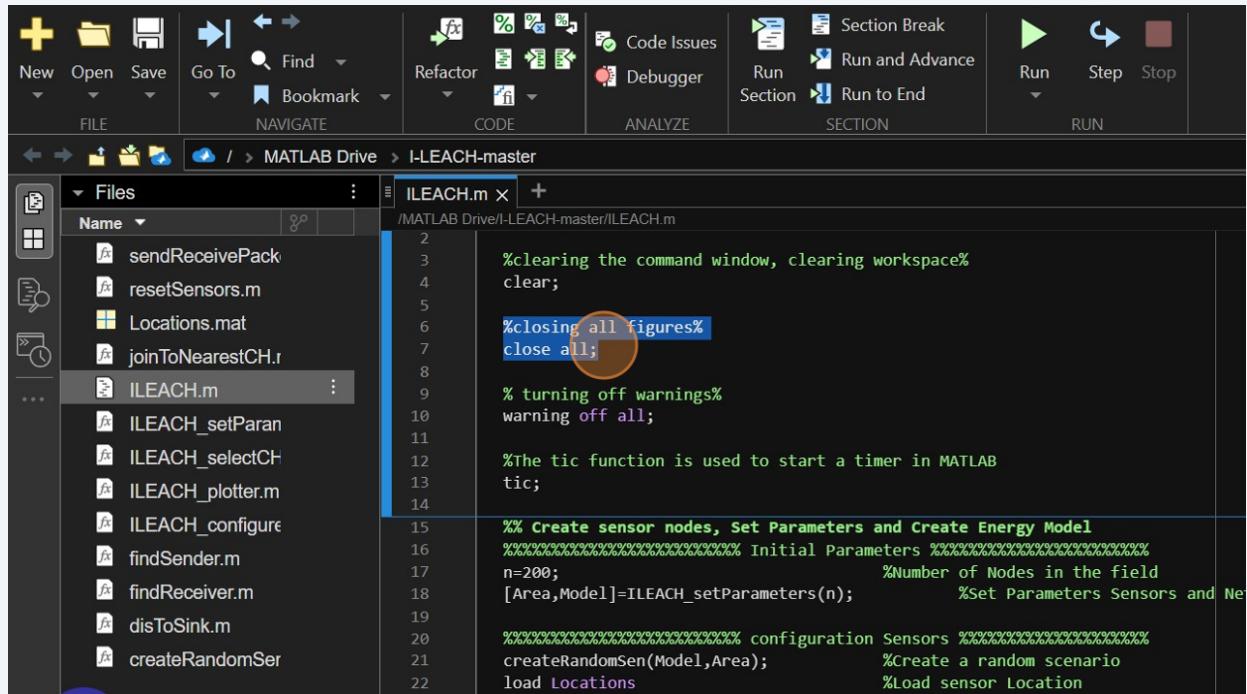
%closing all figures%
close all;

% turning off warnings%
warning off all;

%The tic function is used to start a timer in MATLAB
tic;

%% Create sensor nodes, Set Parameters and Create Energy Model
%%%%%%%%%%%%% Initial Parameters %%%%%%
n=200; %Number of Nodes in the field
[Area,Model]=ILEACH_setParameters(n); %Set Parameters Sensors and Ne
```

4 close all command here is used to close the figures of output simulation if previously opened.



The screenshot shows the MATLAB IDE interface, identical to the previous one but with a different line of code highlighted. The top menu bar, toolbar, and file tree are the same. The main editor window displays the ILEACH.m script:

```
%clearing the command window, clearing workspace%
clear;

%closing all figures%
close all;

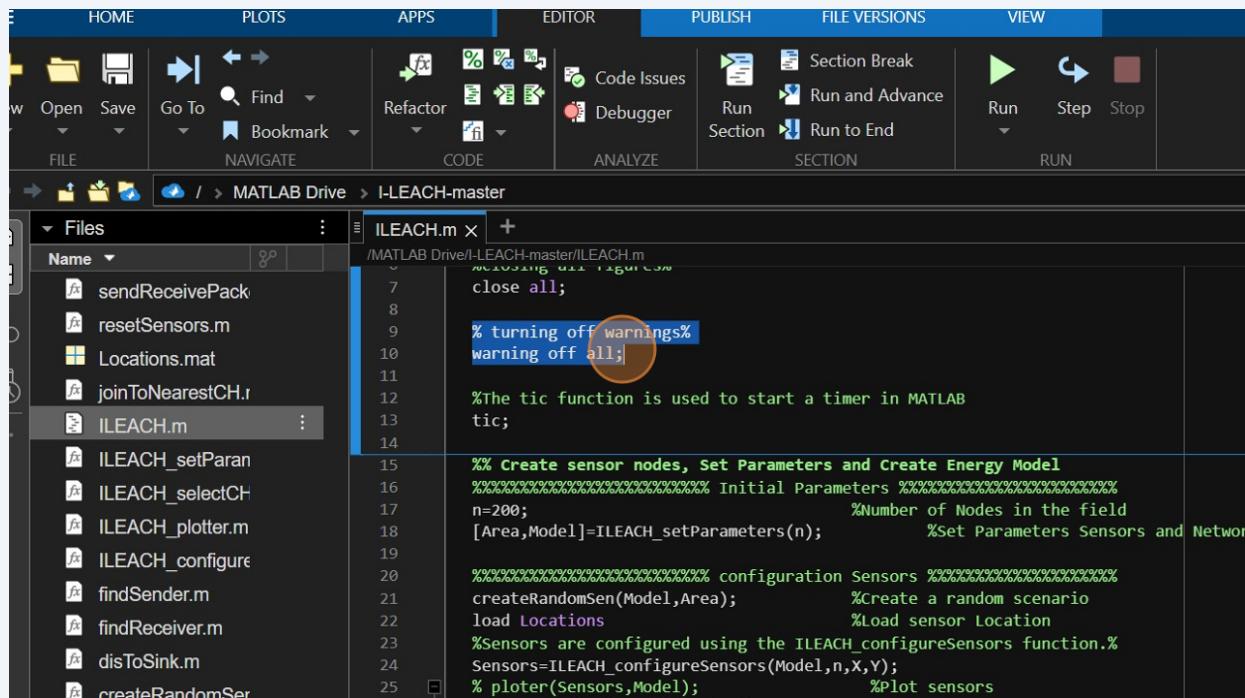
% turning off warnings%
warning off all;

%The tic function is used to start a timer in MATLAB
tic;

%% Create sensor nodes, Set Parameters and Create Energy Model
%%%%%%%%%%%%% Initial Parameters %%%%%%
n=200; %Number of Nodes in the field
[Area,Model]=ILEACH_setParameters(n); %Set Parameters Sensors and Ne
```

5

By using warning off all; the code is instructing MATLAB to turn off all warning messages, meaning that any warning that would typically be displayed during the execution of the script will be suppressed.



The screenshot shows the MATLAB IDE interface with the 'CODE' tab selected. The current file is 'ILEACH.m'. The code editor displays the following script:

```

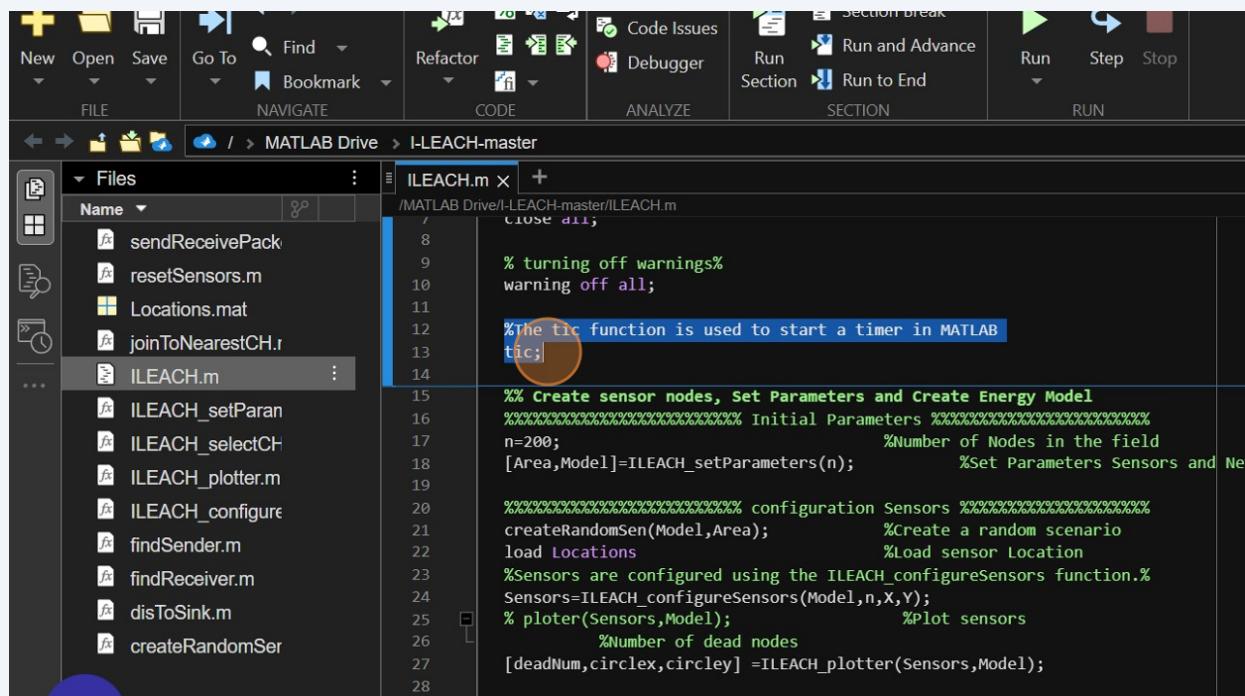
1 close all;
2 % turning off warnings%
3 warning off all;
4 %The tic function is used to start a timer in MATLAB
5 tic;
6
7 %% Create sensor nodes, Set Parameters and Create Energy Model
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Initial Parameters %%%%%%%%%%%%%%
9 n=200; %Number of Nodes in the field
10 [Area,Model]=ILEACH_setParameters(n); %Set Parameters Sensors and Network
11
12 %%%%%%%%%%%%% configuration Sensors %%%%%%%%%%%%%%
13 createRandomSen(Model,Area); %Create a random scenario
14 load Locations %Load sensor Location
15 %Sensors are configured using the ILEACH_configureSensors function.%
16 Sensors=ILEACH_configureSensors(Model,n,X,Y);
17 % ploter(Sensors,Model); %Plot sensors
18
19 %Number of dead nodes
20 [deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model);
21
22
23
24
25
26
27
28

```

The line 'warning off all;' is highlighted with a red circle.

6

In MATLAB, the tic function is used to start a timer, marking the beginning of a time interval



The screenshot shows the MATLAB IDE interface with the 'CODE' tab selected. The current file is 'ILEACH.m'. The code editor displays the following script:

```

1 close all;
2 % turning off warnings%
3 warning off all;
4 %The tic function is used to start a timer in MATLAB
5 tic;
6
7 %% Create sensor nodes, Set Parameters and Create Energy Model
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Initial Parameters %%%%%%%%%%%%%%
9 n=200; %Number of Nodes in the field
10 [Area,Model]=ILEACH_setParameters(n); %Set Parameters Sensors and Network
11
12 %%%%%%%%%%%%% configuration Sensors %%%%%%%%%%%%%%
13 createRandomSen(Model,Area); %Create a random scenario
14 load Locations %Load sensor Location
15 %Sensors are configured using the ILEACH_configureSensors function.%
16 Sensors=ILEACH_configureSensors(Model,n,X,Y);
17 % ploter(Sensors,Model); %Plot sensors
18
19 %Number of dead nodes
20 [deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model);
21
22
23
24
25
26
27
28

```

The line 'tic;' is highlighted with a red circle.

7 Here's a breakdown of the code:

1. `n=200;`: This line sets the number of nodes (n) in the field to 200.
2. `[Area,Model]=ILEACH_setParameters(n);`: This line calls the ILEACH_setParameters function with the argument n to set up initial parameters for the LEACH (Low-Energy Adaptive Clustering Hierarchy) protocol simulation. The function returns two outputs, Area and Model.
 - Area is a structure containing the field dimensions (x and y).
 - Model is a structure containing various parameters and settings for the simulation, including sink position, energy model parameters, maximum number of rounds, packet sizes, radio range, and I-LEACH specific parameters like the number of radio regions (numRx) and region width (dr).

```
%The tic function is used to start a timer in MATLAB
tic;

%% Create sensor nodes, Set Parameters and Create Energy Model
%%%%% Initial Parameters %%%%%%
n=200; %Number of Nodes in the field
[Area,Model]=ILEACH_setParameters(n); %Set Parameters Sensors and Network

%%%%% configuration Sensors %%%%%%
createRandomSen(Model,Area); %Create a random scenario
load Locations %Load sensor Location
%Sensors are configured using the ILEACH_configureSensors function.%
Sensors=ILEACH_configureSensors(Model,n,X,Y);
% ploter(Sensors,Model); %Plot sensors
%Number of dead nodes
[deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model);

%%%%% Parameters initialization %%%%%%
countCHs=0; %counter for Cluster Heads
flag_first_dead=0; %flag_first_dead
```

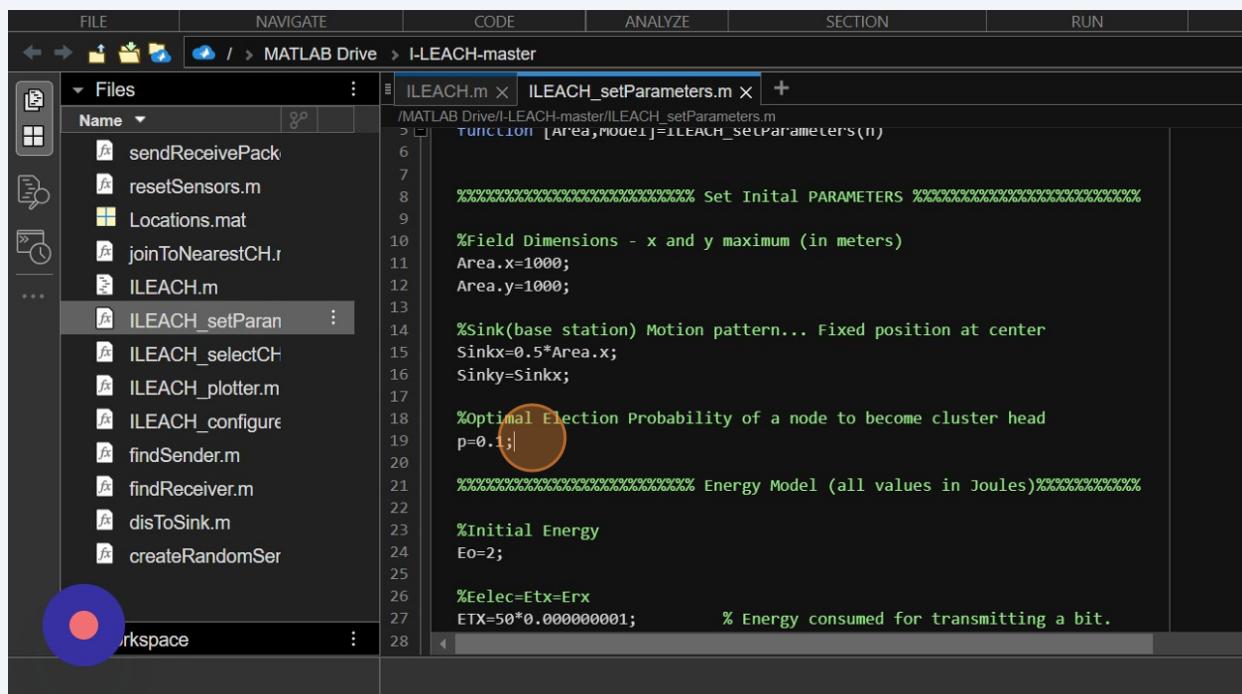
8

Now navigate to ILEACH_setParameters.m and Lets see what is the process of setting paremeters here:

This part of the code is defining the initial parameters for the LEACH (Low-Energy Adaptive Clustering Hierarchy) simulation. Let's break down the key parameters:

1. Area.x=1000; and Area.y=1000;: These lines set the field dimensions. The field is a square with a maximum size of 1000x1000 meters.
2. Sinkx=0.5*Area.x; and Sinky=Sinkx;: These lines set the fixed position of the sink (base station) at the center of the field. The sink's coordinates are determined as half of the maximum field dimensions.
3. p=0.1;: This line sets the optimal election probability of a node to become a cluster head. In LEACH, nodes probabilistically become cluster heads in each round, and p represents the probability.

These parameters are crucial for the initial setup of the simulation and define the physical characteristics of the sensor field, the sink's location, and the probability for cluster head election



```

FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master
Files : ILEACH.m × ILEACH_setParameters.m × +
Name
sendReceivePack
resetSensors.m
Locations.mat
joinToNearestCH.r
ILEACH.m
ILEACH_SetParam ...
ILEACH_SelectCH
ILEACH_Plotter.m
ILEACH_Configure
findSender.m
findReceiver.m
disToSink.m
createRandomSer
Workspace : ...

/MATLAB Drive/I-LEACH-master/ILEACH_setParameters.m
FUNCTION [Area,model]=ILEACH_SetParameters(n)
% Set Initial PARAMETERS
%
% Field Dimensions - x and y maximum (in meters)
Area.x=1000;
Area.y=1000;

% Sink(base station) Motion pattern... Fixed position at center
Sinkx=0.5*Area.x;
Sinky=Sinkx;

% Optimal Election Probability of a node to become cluster head
p=0.1; (Red circle)

% Energy Model (all values in Joules)
%
% Initial Energy
E0=2;

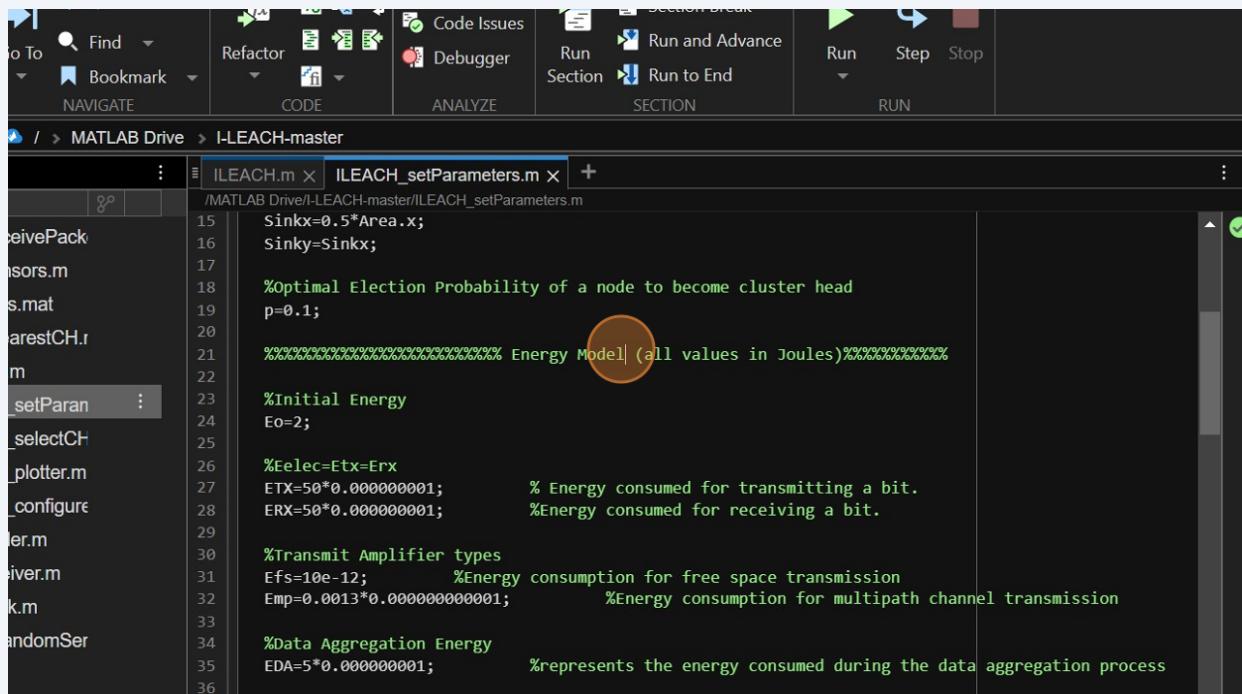
% Eelec=Etx=Erx
ETX=50*0.000000001; % Energy consumed for transmitting a bit.

```

9

Now Lets Discuss the Energy Model:

This part of the code defines the energy model parameters for the sensor nodes in the LEACH simulation. Let's go through each parameter:



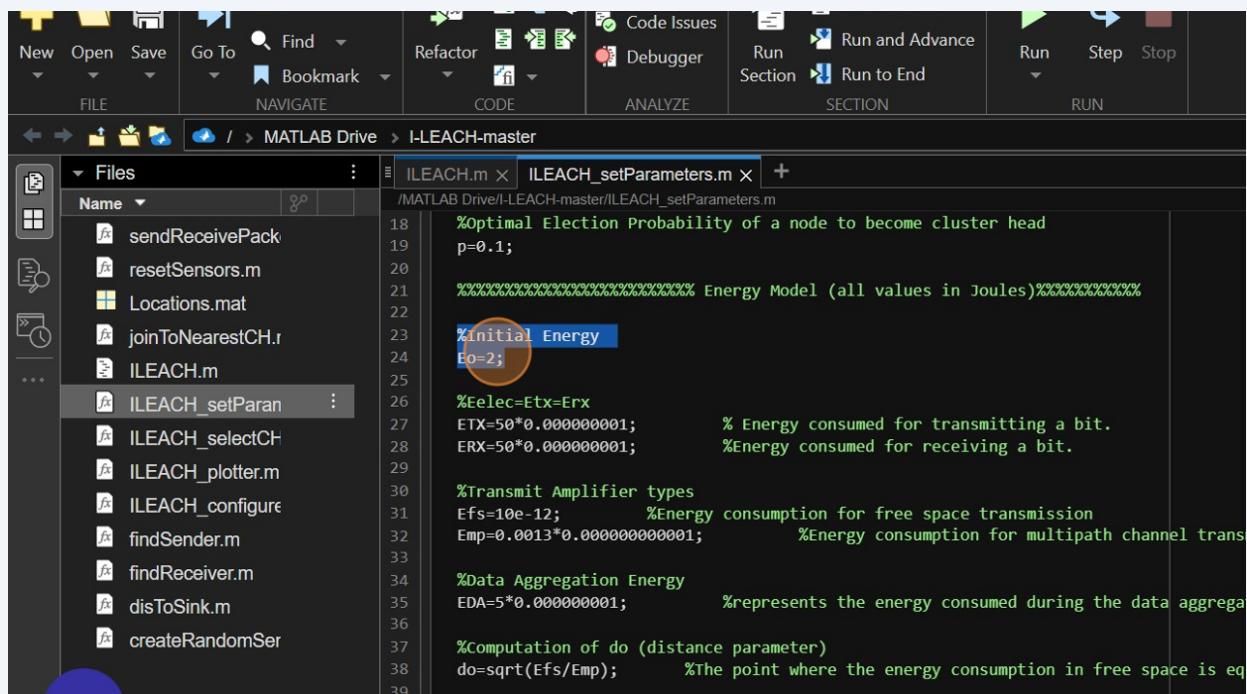
```

ILEACH.m x ILEACH_setParameters.m x +
/MATLAB Drive/I-LEACH-master/ILEACH_setParameters.m
15 | Sinkx=0.5*Area.x;
16 | Sinky=Sinkx;
17 |
18 %Optimal Election Probability of a node to become cluster head
19 p=0.1;
20
21 %%%%%%%%%%%%%% Energy Model (all values in Joules)%%%%%%%%%%%%%
22
23 %Initial Energy
24 Eo=2;
25
26 %Elec=Etx=Erx
27 ETX=50*0.000000001; % Energy consumed for transmitting a bit.
28 ERX=50*0.000000001; %Energy consumed for receiving a bit.
29
30 %Transmit Amplifier types
31 Efs=10e-12; %Energy consumption for free space transmission
32 Emp=0.0013*0.00000000001; %Energy consumption for multipath channel transmission
33
34 %Data Aggregation Energy
35 EDA=5*0.000000001; %represents the energy consumed during the data aggregation process
36
37
38
39

```

10

1. $Eo=2;$: This line sets the initial energy level of the sensor nodes to 2 Joules.



```

FILE NAVIGATE CODE ANALYZE SECTION RUN
I-LEACH-master
ILEACH.m x ILEACH_setParameters.m x +
/MATLAB Drive/I-LEACH-master/ILEACH_setParameters.m
18 | %Optimal Election Probability of a node to become cluster head
19 | p=0.1;
20 |
21 %%%%%%%%%%%%%% Energy Model (all values in Joules)%%%%%%%%%%%%%
22
23 %Initial Energy
24 Eo=2;
25
26 %Elec=Etx=Erx
27 ETX=50*0.000000001; % Energy consumed for transmitting a bit.
28 ERX=50*0.000000001; %Energy consumed for receiving a bit.
29
30 %Transmit Amplifier types
31 Efs=10e-12; %Energy consumption for free space transmission
32 Emp=0.0013*0.00000000001; %Energy consumption for multipath channel trans
33
34 %Data Aggregation Energy
35 EDA=5*0.000000001; %represents the energy consumed during the data aggrega
36
37 %Computation of do (distance parameter)
38 do=sqrt(Efs/Emp); %The point where the energy consumption in free space is eq
39

```

11

1. ETX=50*0.000000001; and ERX=50*0.000000001;: These lines set the energy consumed for transmitting (ETX) and receiving (ERX) a single bit, respectively. Both values are set to 50 nanojoules/bit.
2. Efs=10e-12;: This line sets the energy consumption for free space transmission. It represents the energy consumed per bit when transmitting over a free space channel. The value is set to **1×10-12** Joules/bit.
3. Emp=0.0013*0.00000000001;: This line sets the energy consumption for multipath channel transmission. It represents the energy consumed per bit when transmitting over a multipath channel. The value is calculated based on a specific formula and set to **1.3×10-15** Joules/bit.
4. EDA=5*0.000000001;: This line sets the energy consumed during the data aggregation process. The value is set to **5×10-9** Joules.
5. do=sqrt(Efs/Emp);: This line computes the distance parameter do, which represents the distance at which the energy consumption in free space is equal to that in a multipath channel. It is calculated based on the defined values of Efs and Emp.

These parameters collectively define the energy consumption characteristics of the sensor nodes during communication and data aggregation in the simulation.

The screenshot shows the MATLAB Drive interface with the 'I-LEACH-master' folder selected. The left sidebar displays a file tree with files like sendReceivePack, resetSensors.m, Locations.mat, joinToNearestCH.r, ILEACH.m, and ILEACH_setParan. The main workspace shows the code for ILEACH_setParameters.m:

```
%%%%%% Energy Model (all values in Joules)%%%%%
%Initial Energy
Eo=2;

%Elec=Etx=Erx
ETX=50*0.00000001; % Energy consumed for transmitting a bit.
ERX=50*0.00000001; %Energy consumed for receiving a bit.

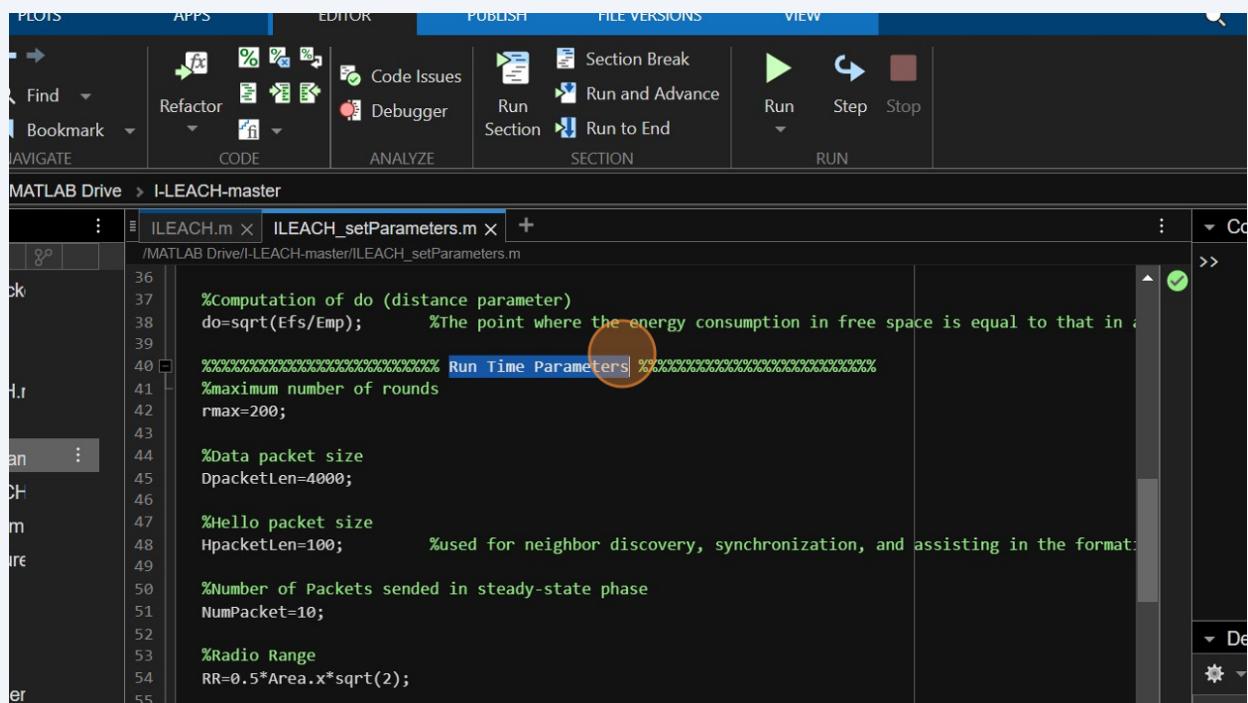
%Transmit Amplifier types
Efs=10e-12; %Energy consumption for free space transmission
Emp=0.0013*0.0000000001; %Energy consumption for multipath channel transm

%Data Aggregation Energy
EDA=5*0.00000001; %represents the energy consumed during the data aggregat

%Computation of do (distance parameter)
do=sqrt(Efs/Emp); %The point where the energy consumption in free space is equ

%%%%% Run Time Parameters %%%%%%
%maximum number of rounds
rmax=200;
```

12 Now here are the Run Time Parameters lets discuss about these



The screenshot shows the MATLAB IDE interface. The menu bar includes PLOTS, APPS, EDITOR (selected), PUBLISH, FILE VERSIONS, and VIEW. The toolbars include NAVIGATE, CODE, ANALYZE, SECTION, and RUN. The code editor displays two files: ILEACH.m and ILEACH_setParameters.m. The ILEACH_setParameters.m file contains MATLAB code for setting parameters. A red circle highlights the text '%Run Time Parameters' in line 40 of the code.

```
%Computation of do (distance parameter)
do=sqrt(Efs/Emp); %The point where the energy consumption in free space is equal to that in a
%maximum number of rounds
rmax=200;

%Dpacket size
DpacketLen=4000;

%Hello packet size
HpacketLen=100; %used for neighbor discovery, synchronization, and assisting in the format

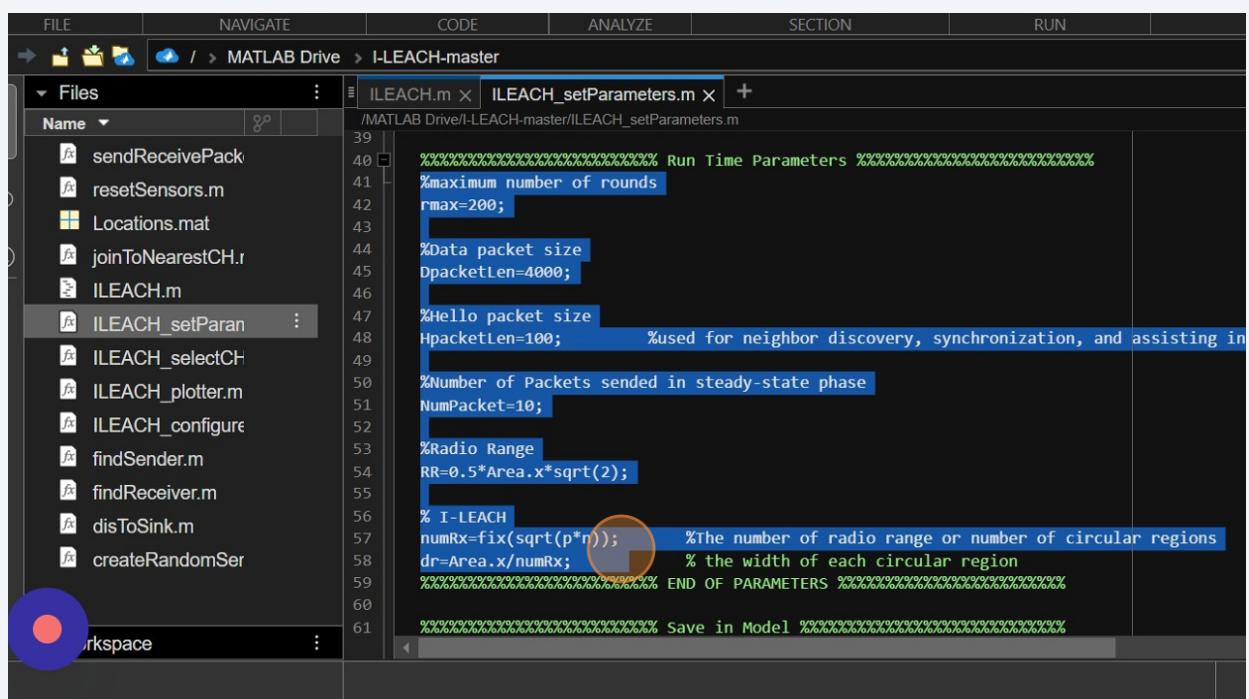
%Number of Packets sended in steady-state phase
NumPacket=10;

%Radio Range
RR=0.5*Area.x*sqrt(2);
```

13 Let's go through each parameter:

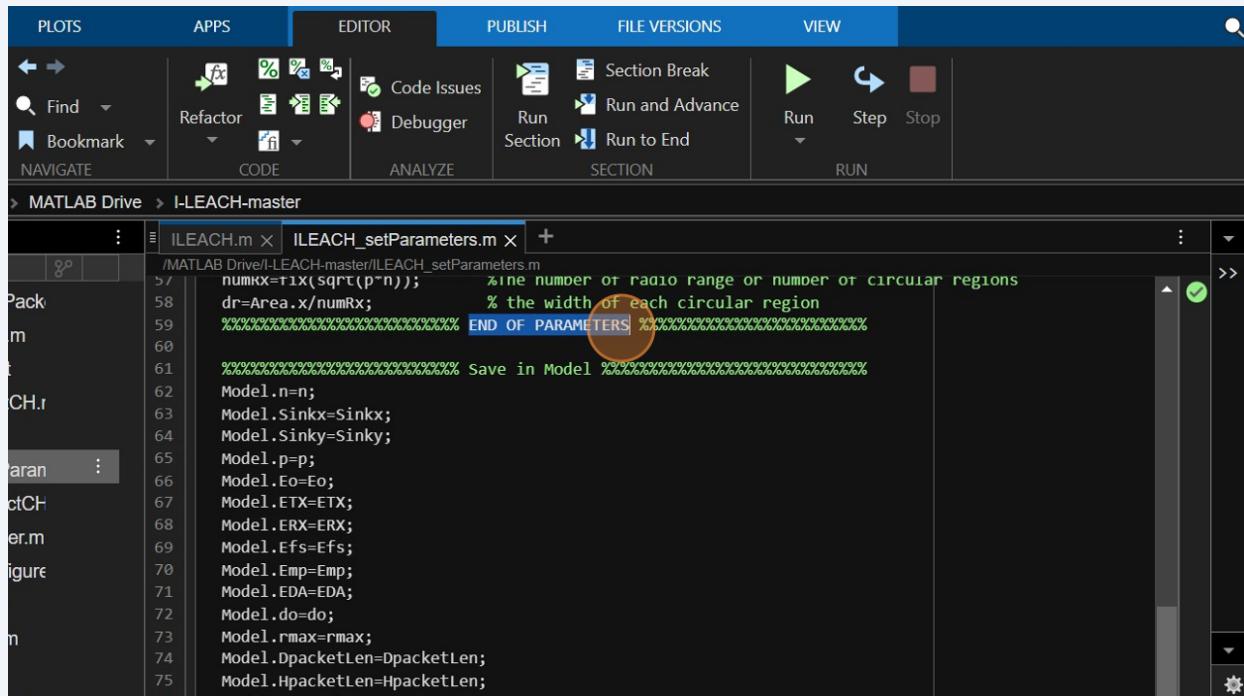
1. rmax=200;: This line sets the maximum number of rounds for the simulation to 200.
2. DpacketLen=4000;: This line sets the data packet size to 4000 bits.
3. HpacketLen=100;: This line sets the hello packet size to 100 bits. Hello packets are used for neighbor discovery, synchronization, and assisting in the formation and maintenance of clusters.
4. NumPacket=10;: This line sets the number of packets sent in the steady-state phase to 10.
5. RR=0.5*Area.x*sqrt(2);: This line calculates the radio range (RR) based on the field dimensions (Area.x and Area.y). The radio range is set to half of the diagonal of the field.
6. numRx=fix(sqrt(p*n));: This line calculates the number of radio ranges or the number of circular regions (numRx) based on the optimal election probability (p) and the number of nodes (n). The fix function rounds down the result to the nearest integer.
7. dr=Area.x/numRx;: This line calculates the width of each circular region (dr) based on the field dimensions (Area.x) and the number of circular regions (numRx).

These parameters control aspects such as the duration of the simulation, packet sizes, radio range, and the configuration of circular regions in the I-LEACH protocol.



```
FILE NAVIGATE CODE ANALYZE SECTION RUN
/MATLAB Drive/I-LEACH-master/ILEACH_setParameters.m
39 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Run Time Parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
40 rmax=200;
41 %Data packet size
42 DpacketLen=4000;
43 %Hello packet size
44 HpacketLen=100; %used for neighbor discovery, synchronization, and assisting in
45 %Number of Packets sended in steady-state phase
46 NumPacket=10;
47 %Radio Range
48 RR=0.5*Area.x*sqrt(2);
49 % I-LEACH
50 numRx=fix(sqrt(p*n)); %The number of radio range or number of circular regions
51 dr=Area.x/numRx; % the width of each circular region
52 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% END OF PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Save in Model %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

14 Now Parameters in this function are ended



The screenshot shows the MATLAB Editor interface with two files open: ILEACH.m and ILEACH_setParameters.m. The ILEACH_setParameters.m file contains the following code:

```
/MATLAB Drive/I-LEACH-master/ILEACH_setParameters.m
57
58 numRx=fix(sqrt(p*n)); %the number of radio range or number of circular regions
59 dr=Area.x/numRx; % the width of each circular region
60 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% END OF PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Save in Model %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
62 Model.n=n;
63 Model.Sinkx=Sinkx;
64 Model.Sinky=Sinky;
65 Model.p=p;
66 Model.Eo=Eo;
67 Model.ETX=ETX;
68 Model.ERX=ERX;
69 Model.Efs=Efs;
70 Model.Emp=Emp;
71 Model.EDA=EDA;
72 Model.do=do;
73 Model.rmax=rmax;
74 Model.DpacketLen=DpacketLen;
75 Model.HpacketLen=HpacketLen;
```

15

In this section, the values of the parameters and calculated variables are assigned to the Model structure. Here's a breakdown of what each line does:

1. Model.n=n;: Assigns the number of nodes (n) to the n field in the Model structure.
2. Model.Sinkx=Sinkx;: Assigns the x-coordinate of the sink (Sinkx) to the Sinkx field in the Model structure.
3. Model.Sinky=Sinky;: Assigns the y-coordinate of the sink (Sinky) to the Sinky field in the Model structure.
4. Model.p=p;: Assigns the optimal election probability (p) to the p field in the Model structure.
5. Model.Eo=Eo;: Assigns the initial energy (Eo) to the Eo field in the Model structure.
6. Model.ETX=ETX;: Assigns the energy consumed for transmitting a bit (ETX) to the ETX field in the Model structure.
7. Model.ERX=ERX;: Assigns the energy consumed for receiving a bit (ERX) to the ERX field in the Model structure.
8. Model.Efs=Efs;: Assigns the energy consumption for free space transmission (Efs) to the Efs field in the Model structure.
9. Model.Emp=Emp;: Assigns the energy consumption for multipath channel transmission (Emp) to the Emp field in the Model structure.
10. Model.EDA=EDA;: Assigns the energy consumed during the data aggregation process (EDA) to the EDA field in the Model structure.
11. Model.do=do;: Assigns the distance parameter (do) to the do field in the Model structure.
12. Model.rmax=rmax;: Assigns the maximum number of rounds (rmax) to the rmax field in the Model structure.
13. Model.DpacketLen=DpacketLen;: Assigns the data packet size (DpacketLen) to the DpacketLen field in the Model structure.
14. Model.HpacketLen=HpacketLen;: Assigns the hello packet size (HpacketLen) to the HpacketLen field in the Model structure.
15. Model.NumPacket=NumPacket;: Assigns the number of packets sent in the steady-state phase (NumPacket) to the NumPacket field in the Model structure.
16. Model.RR=RR;: Assigns the radio range (RR) to the RR field in the Model structure.

All these values are crucial for defining the simulation parameters and the I-LEACH protocol's behavior.

```

FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive / > I-LEACH-master
Files ILEACH.m x ILEACH_setParameters.m x +
Name
sendReceivePack
resetSensors.m
Locations.mat
joinToNearestCH.r
ILEACH.m
ILEACH_SetParam ...
ILEACH_SelectCH
ILEACH_Plotter.m
ILEACH_Configure
findSender.m
findReceiver.m
disToSink.m
createRandomSer
Workspace

```

```

60 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Save in Model %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61 Model1.n=n;
62 Model1.Sinkx=Sinkx;
63 Model1.Sinky=Sinky;
64 Model1.p=p;
65 Model1.Eo=Eo;
66 Model1.ETX=ETX;
67 Model1.ERX=ERX;
68 Model1.Efs=Efs;
69 Model1.Emp=Emp;
70 Model1.EDA=EDA;
71 Model1.do=do;
72 Model1.rmax=rmax;
73 Model1.DpacketLen=DpacketLen;
74 Model1.HpacketLen=HpacketLen;
75 Model1.NumPacket=NumPacket;
76 Model1.RR=RR;
77
78
79
80 % I-LEACH
81 Model1.numRx=numRx;
82 Model1.dr=dr;

```

16

In this last part of this function,
the last two lines are additional assignments for the I-LEACH extension:

1. Model.numRx=numRx;: Assigns the number of radio range or the number of circular regions (numRx) to the numRx field in the Model structure.
2. Model.dr=dr;: Assigns the width of each circular region (dr) to the dr field in the Model structure.

```

FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive / > I-LEACH-master
Files ILEACH.m x ILEACH_setParameters.m x +
Name
sendReceivePack
resetSensors.m
Locations.mat
joinToNearestCH.r
ILEACH.m
ILEACH_SetParam ...
ILEACH_SelectCH
ILEACH_Plotter.m
ILEACH_Configure
findSender.m
findReceiver.m
disToSink.m
createRandomSer
Workspace

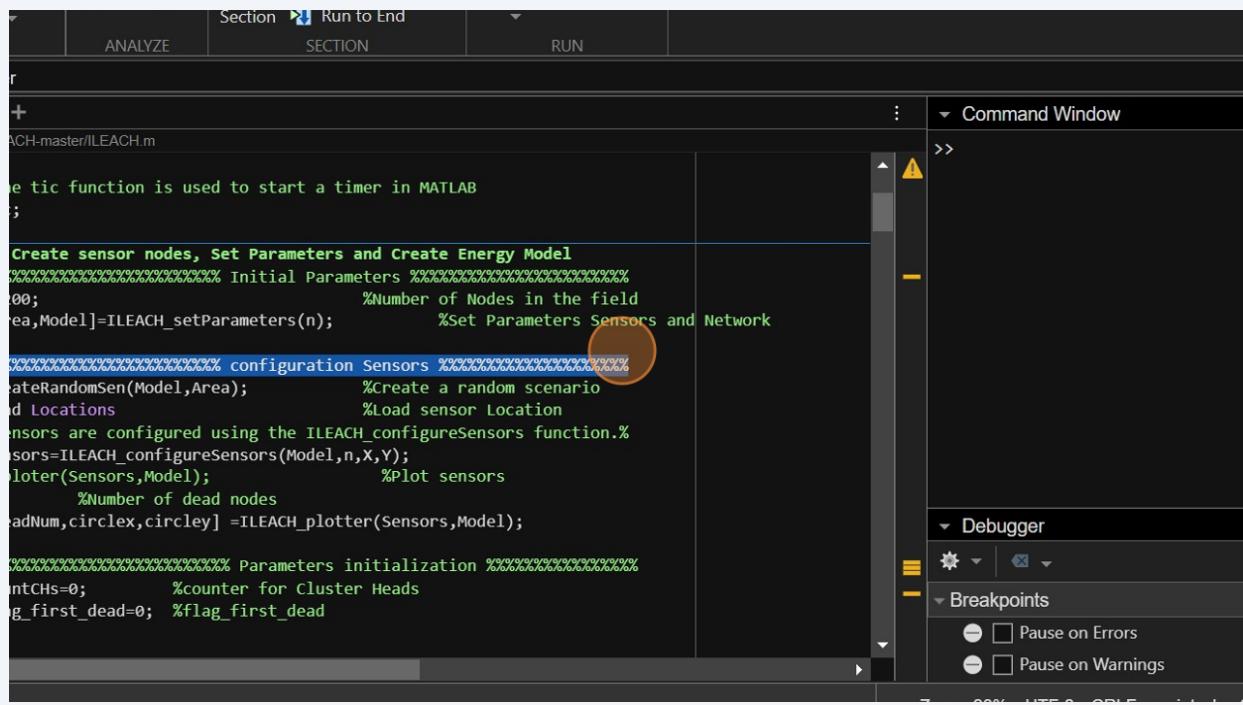
```

```

62 %%%%%%
63 function [Area,Model]=ILEACH_SetParameters(n)
64 Model1.Sinky=Sinky;
65 Model1.p=p;
66 Model1.Eo=Eo;
67 Model1.ETX=ETX;
68 Model1.ERX=ERX;
69 Model1.Efs=Efs;
70 Model1.Emp=Emp;
71 Model1.EDA=EDA;
72 Model1.do=do;
73 Model1.rmax=rmax;
74 Model1.DpacketLen=DpacketLen;
75 Model1.HpacketLen=HpacketLen;
76 Model1.NumPacket=NumPacket;
77 Model1.RR=RR;
78
79
80 % I-LEACH
81 Model1.numRx=numRx;
82 Model1.dr=dr;
83
84 end

```

17 After setting perimeters Sensors are configured



The screenshot shows the MATLAB IDE interface. The top menu bar includes ANALYZE, Section, Run to End, SECTION, and RUN. The left pane displays the code for ILEACH.m. A circled line highlights the line `createRandomSen(Model,Area);`. The right pane shows the Command Window and the Debugger panel, which includes a Breakpoints section with options for Pause on Errors and Pause on Warnings.

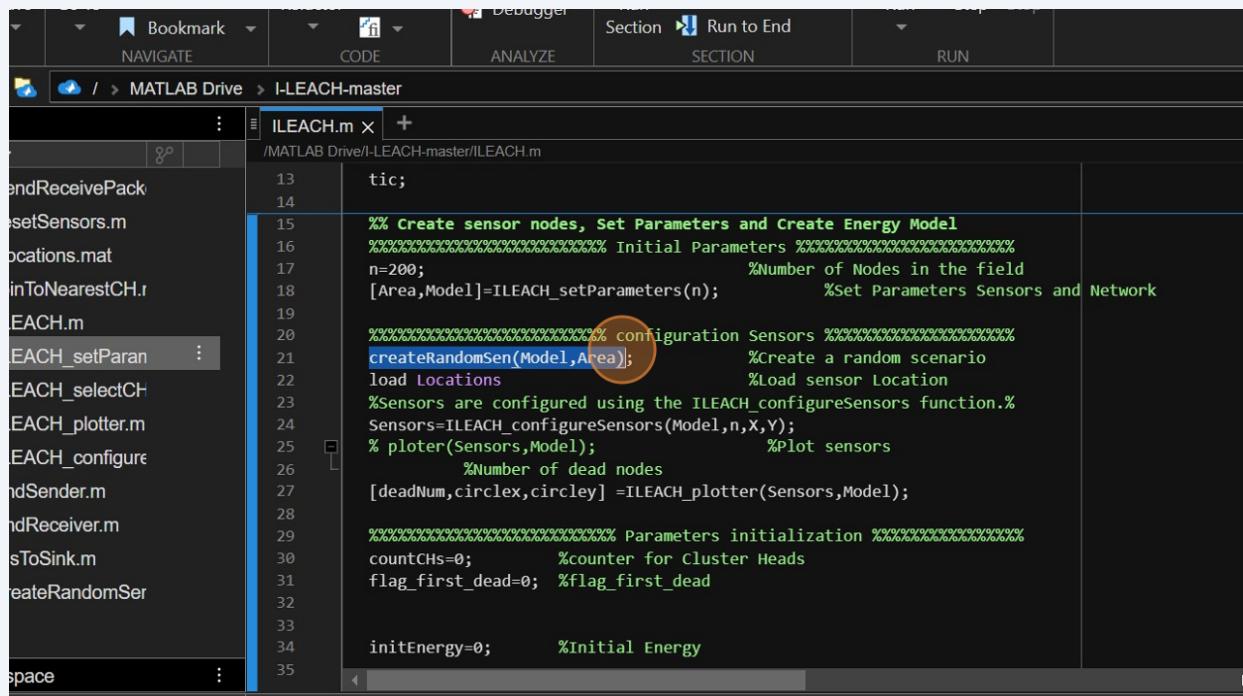
```
%tic function is used to start a timer in MATLAB
;

Create sensor nodes, Set Parameters and Create Energy Model
%%%%%%%%%%%%% Initial Parameters %%%%%%
n=200; %Number of Nodes in the field
[Area,Model]=ILEACH_setParameters(n); %Set Parameters Sensors and Network

%%%%%%%%%%%%% configuration Sensors %%%%%%
createRandomSen(Model,Area); %Create a random scenario
load Locations %Load sensor Location
%Sensors are configured using the ILEACH_configureSensors function.%
Sensors=ILEACH_configureSensors(Model,n,X,Y);
ploter(Sensors,Model); %Plot sensors
%Number of dead nodes
[deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model);

%%%%%%%%%%%%% Parameters initialization %%%%%%
countCHs=0; %counter for Cluster Heads
flag_first_dead=0; %flag_first_dead
```

18 In first there is a function createRandomSen that takes two parameters and now lets observe the functionality of this funxtion from the file of the function name createRandomSen

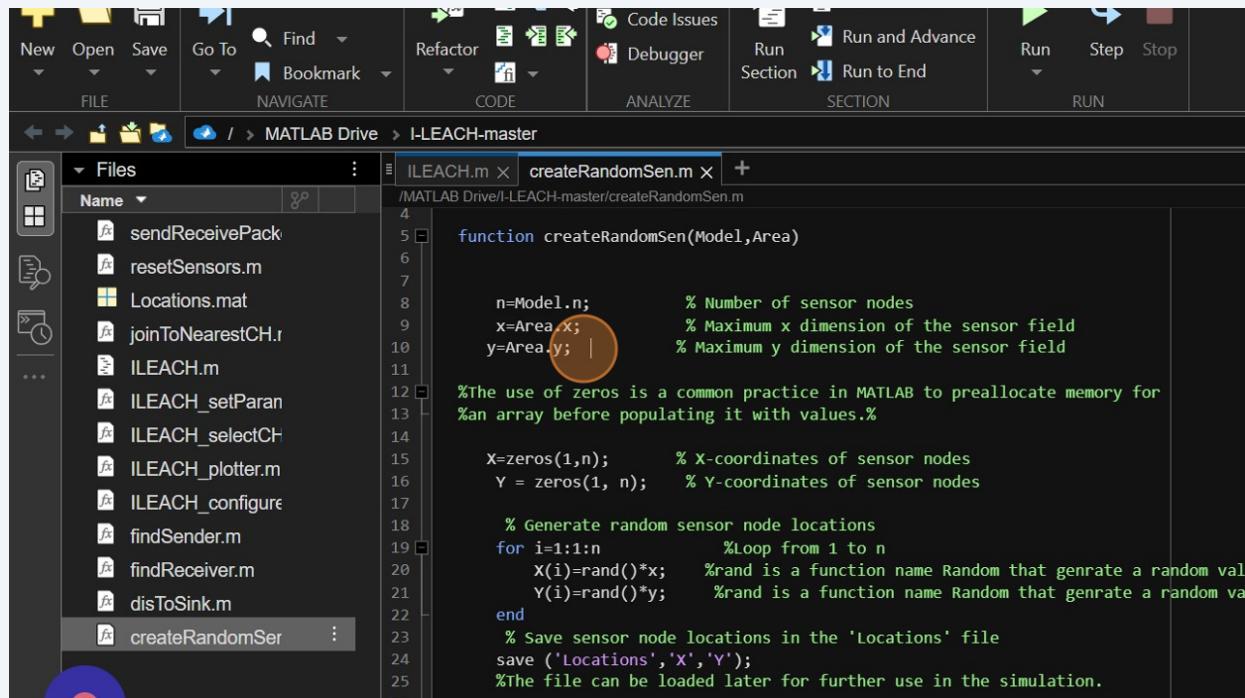


The screenshot shows the MATLAB IDE interface. The top menu bar includes NAVIGATE, CODE, Debugger, Section, Run to End, SECTION, and RUN. The left pane shows a list of files in the MATLAB Drive, with ILEACH.m selected. The right pane displays the code for ILEACH.m. A circled line highlights the line `createRandomSen(Model,Area);`.

```
endReceivePack
setSensors.m
locations.mat
inToNearestCH.r
LEACH.m
ILEACH_setParan :
ILEACH_selectCH
ILEACH_plotter.m
ILEACH_configure
endSender.m
endReceiver.m
sToSink.m
createRandomSer
space
```

```
13 | tic;
14 |
15 | % Create sensor nodes, Set Parameters and Create Energy Model
16 | %%%%%% Initial Parameters %%%%%%
17 | n=200; %Number of Nodes in the field
18 | [Area,Model]=ILEACH_setParameters(n); %Set Parameters Sensors and Network
19 |
20 | %%%%%% configuration Sensors %%%%%%
21 | createRandomSen(Model,Area); %Create a random scenario
22 | load Locations %Load sensor Location
23 | %Sensors are configured using the ILEACH_configureSensors function.%
24 | Sensors=ILEACH_configureSensors(Model,n,X,Y);
25 | % ploter(Sensors,Model); %Plot sensors
26 | %Number of dead nodes
27 | [deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model);
28 |
29 | %%%%%% Parameters initialization %%%%%%
30 | countCHs=0; %counter for Cluster Heads
31 | flag_first_dead=0; %flag_first_dead
32 |
33 |
34 | initEnergy=0; %Initial Energy
35 |
```

19 Click "y=Area.y;"



```
function createRandomSen(Model,Area)

n=Model.n; % Number of sensor nodes
x=Area.x; % Maximum x dimension of the sensor field
y=Area.y; | % Maximum y dimension of the sensor field

%The use of zeros is a common practice in MATLAB to preallocate memory for
%an array before populating it with values.

X=zeros(1,n); % X-coordinates of sensor nodes
Y = zeros(1, n); % Y-coordinates of sensor nodes

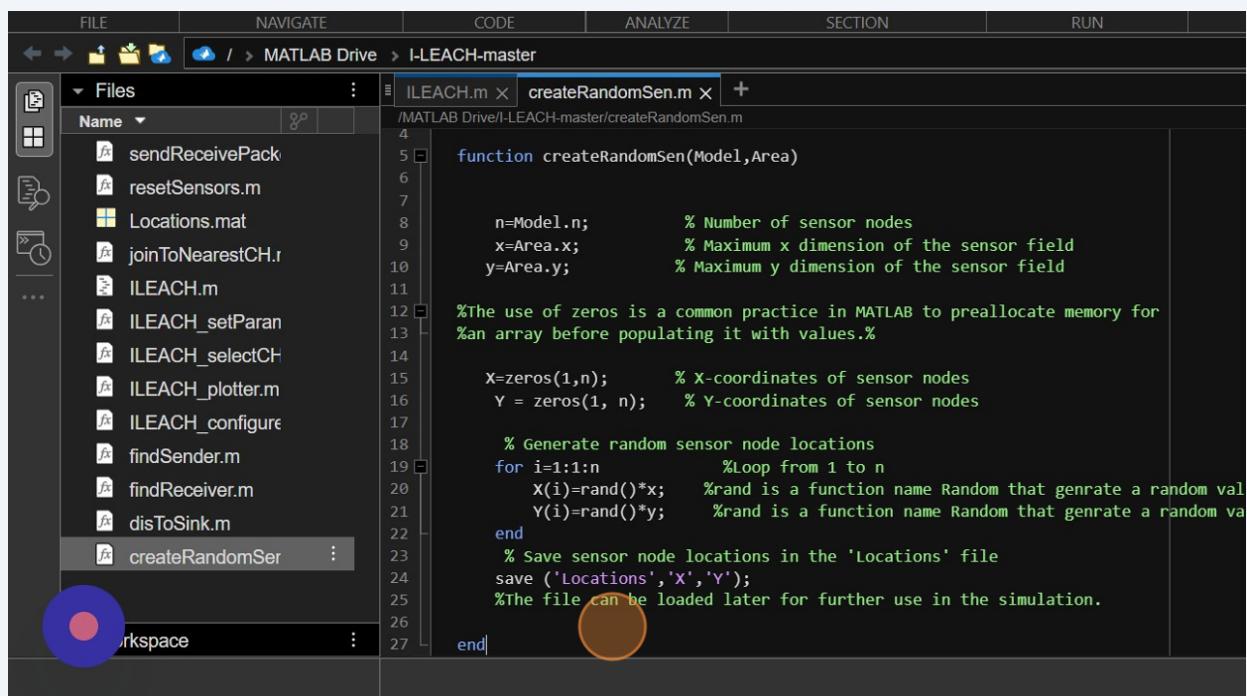
% Generate random sensor node locations
for i=1:n %Loop from 1 to n
    X(i)=rand()*x; %rand is a function name Random that generate a random val
    Y(i)=rand()*y; %rand is a function name Random that generate a random va
end
% Save sensor node locations in the 'Locations' file
save ('Locations','X','Y');
%The file can be loaded later for further use in the simulation.
```

20

The remaining part of the this function code generates random sensor node locations within the specified field dimensions and saves them in a file named 'Locations'. Here's a breakdown of the code:

- $X=zeros(1,n);$: Preallocates an array X of size 1 x n to store x-coordinates of sensor nodes.
- $Y=zeros(1,n);$: Preallocates an array Y of size 1 x n to store y-coordinates of sensor nodes.
- for $i=1:1:n$: Iterates from 1 to n.
 - $X(i)=rand()*x;$: Generates a random x-coordinate for the i-th sensor node and assigns it to X(i).
 - $Y(i)=rand()*y;$: Generates a random y-coordinate for the i-th sensor node and assigns it to Y(i).
- $save ('Locations', 'X', 'Y');$: Saves the generated sensor node locations in the 'Locations' file. The file will contain the X and Y coordinates of each sensor node.

This file can be used later in the simulation for various purposes, such as initializing the sensor nodes with specific locations.



The screenshot shows the MATLAB IDE interface with the following details:

- Toolbar:** FILE, NAVIGATE, CODE, ANALYZE, SECTION, RUN.
- Current Path:** MATLAB Drive > I-LEACH-master
- File Explorer:** Shows files in the 'I-LEACH-master' folder, including 'ILEACH.m', 'createRandomSen.m', 'sendReceivePack.m', 'resetSensors.m', 'Locations.mat', 'joinToNearestCH.r', 'ILEACH.m', 'ILEACH_setParam.m', 'ILEACH_selectCH.m', 'ILEACH_plotter.m', 'ILEACH_configure.m', 'findSender.m', 'findReceiver.m', 'disToSink.m', and 'createRandomSen.m' (selected).
- Editor:** Displays the content of the 'createRandomSen.m' script:

```
function createRandomSen(Model,Area)

n=Model.n;           % Number of sensor nodes
x=Area.x;           % Maximum x dimension of the sensor field
y=Area.y;           % Maximum y dimension of the sensor field

%The use of zeros is a common practice in MATLAB to preallocate memory for
%an array before populating it with values.

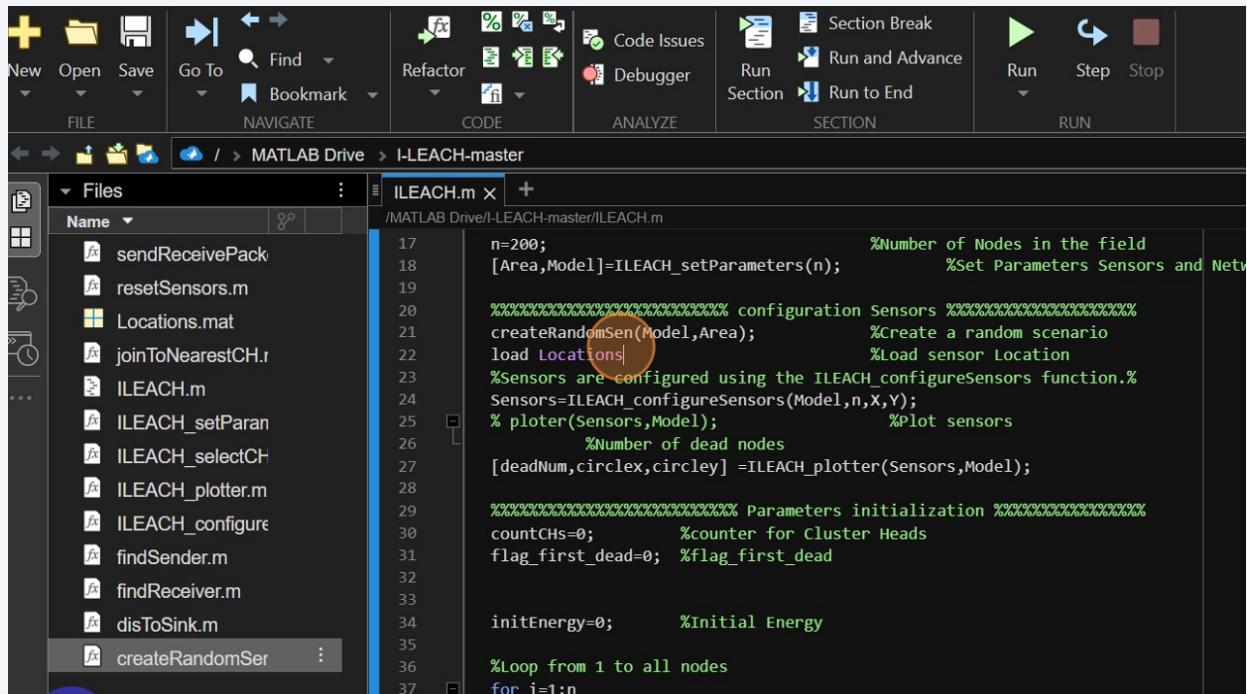
x=zeros(1,n);       % X-coordinates of sensor nodes
Y = zeros(1, n);    % Y-coordinates of sensor nodes

% Generate random sensor node locations
for i=1:1:n          %Loop from 1 to n
    X(i)=rand()*x;   %rand is a function name Random that generate a random val
    Y(i)=rand()*y;   %rand is a function name Random that generate a random va
end
% Save sensor node locations in the 'Locations' file
save ('Locations', 'X', 'Y');
%The file can be loaded later for further use in the simulation.

end
```

21

Now moving back to Main ILEACH file Here load Locations command is to load the Location of sensor



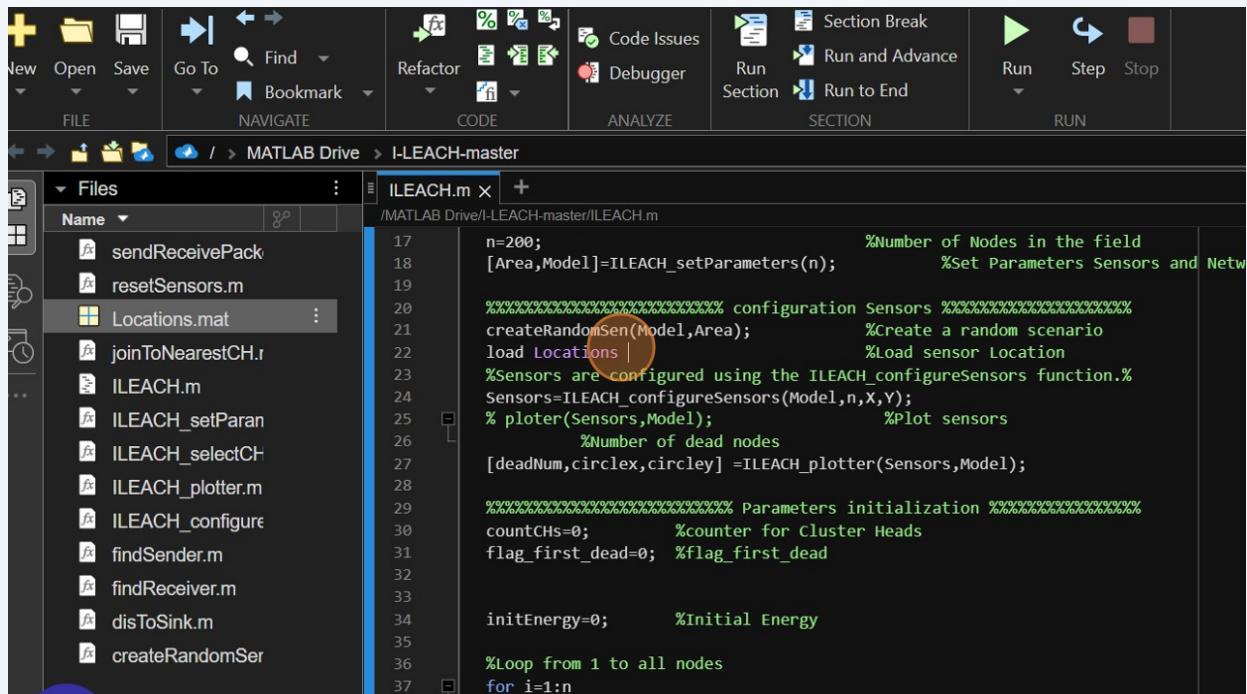
```

17 n=200; %Number of Nodes in the field
18 [Area,Model]=ILEACH_setParameters(n); %Set Parameters Sensors and Network
19
20 %%%%%%%%%%%%%% configuration Sensors %%%%%%%%%%%%%%
21 createRandomSen(Model,Area); %Create a random scenario
22 load Locations; %Load sensor Location
23 %Sensors are configured using the ILEACH_configureSensors function.%
24 Sensors=ILEACH_configureSensors(Model,n,X,Y);
25 % ploter(Sensors,Model); %Plot sensors
26 %Number of dead nodes
27 [deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model);
28
29 %%%%%%%%%%%%%% Parameters initialization %%%%%%%%%%%%%%
30 countCHs=0; %counter for Cluster Heads
31 flag_first_dead=0; %flag_first_dead
32
33
34 initEnergy=0; %Initial Energy
35
36 %Loop from 1 to all nodes
37 for i=1:n

```

22

This command also has a file Locations.mat as you can see in Files , when it run it just loads the locations



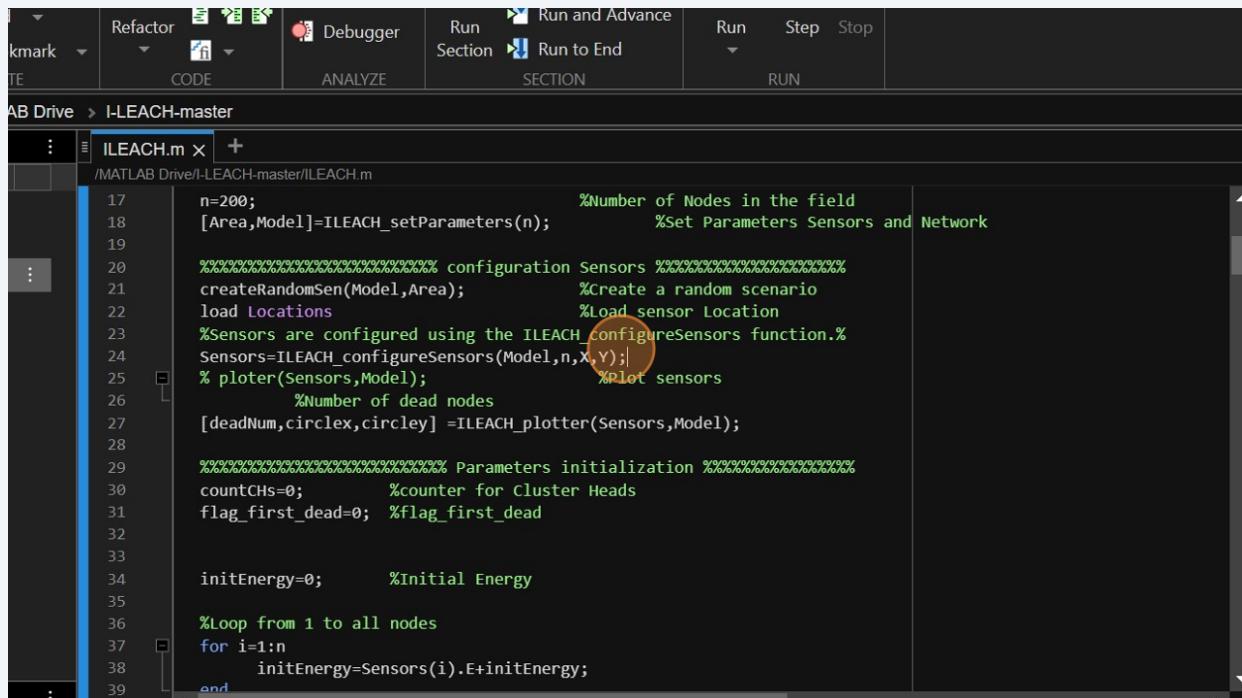
```

17 n=200; %Number of Nodes in the field
18 [Area,Model]=ILEACH_setParameters(n); %Set Parameters Sensors and Network
19
20 %%%%%%%%%%%%%% configuration Sensors %%%%%%%%%%%%%%
21 createRandomSen(Model,Area); %Create a random scenario
22 load Locations; %Load sensor Location
23 %Sensors are configured using the ILEACH_configureSensors function.%
24 Sensors=ILEACH_configureSensors(Model,n,X,Y);
25 % ploter(Sensors,Model); %Plot sensors
26 %Number of dead nodes
27 [deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model);
28
29 %%%%%%%%%%%%%% Parameters initialization %%%%%%%%%%%%%%
30 countCHs=0; %counter for Cluster Heads
31 flag_first_dead=0; %flag_first_dead
32
33
34 initEnergy=0; %Initial Energy
35
36 %Loop from 1 to all nodes
37 for i=1:n

```

23

Now in the next step ILEACH_configureSensors function is used that is also in another file with the same name as function



```
n=200; %Number of Nodes in the field
[Area,Model]=ILEACH_setParameters(n); %Set Parameters Sensors and Network

%%%%%%%%%%%%% configuration Sensors %%%%%%
createRandomSen(Model,Area); %Create a random scenario
load locations %Load sensor Location
%Sensors are configured using the ILEACH_configureSensors function.%
Sensors=ILEACH_configureSensors(Model,n,X,Y);
% ploter(Sensors,Model); %Plot sensors
%Number of dead nodes
[deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model);

%%%%%%%%%%%%% Parameters initialization %%%%%%
countCHs=0; %counter for Cluster Heads
flag_first_dead=0; %flag_first_dead

initEnergy=0; %Initial Energy

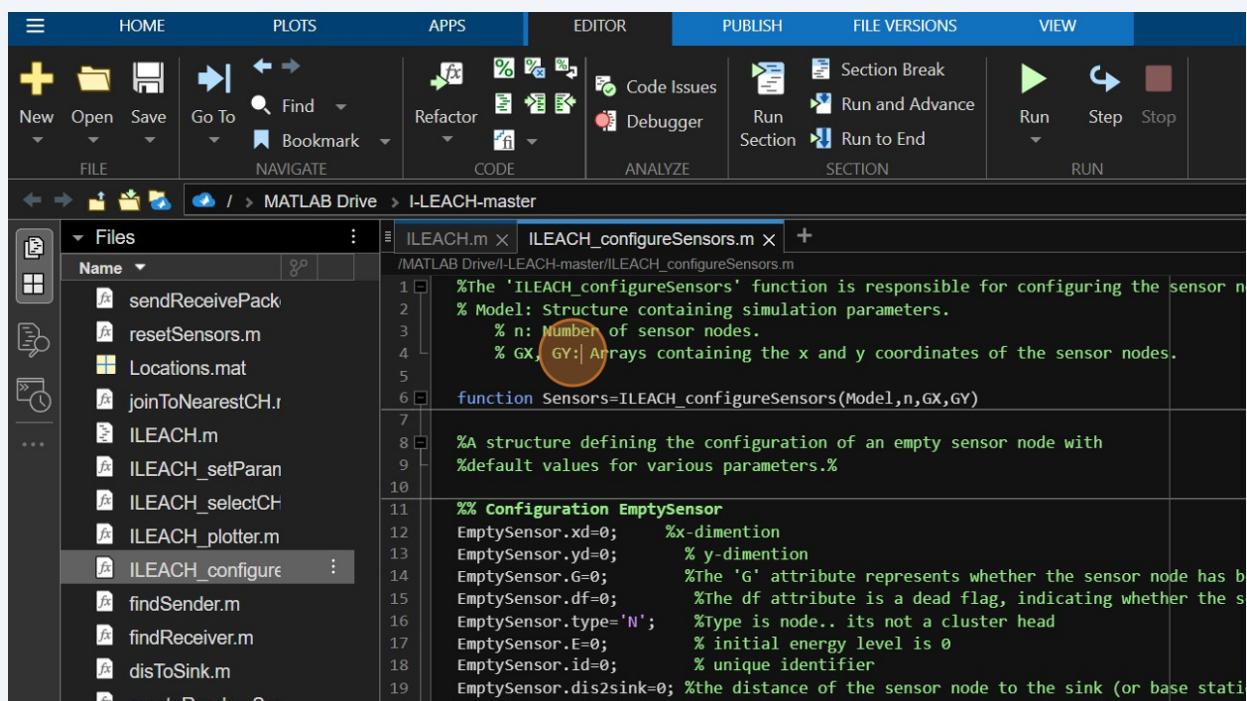
%Loop from 1 to all nodes
for i=1:n
    initEnergy=Sensors(i).E+initEnergy;
end
```

24

Now let's observe this function file code

This function, ILEACH_configureSensors, configures the sensor nodes in the LEACH protocol based on the specified model parameters. Here's a breakdown of the code:

- EmptySensor: Defines a structure representing the configuration of an empty sensor node with default values for various parameters.
 - EmptySensor.xd=0;: Initializes the x-dimension of the sensor node to 0.
 - EmptySensor.yd=0;: Initializes the y-dimension of the sensor node to 0.
 - EmptySensor.G=0;: Initializes the 'G' attribute, representing whether the sensor node has been a cluster head in previous periods, to 0.
 - EmptySensor.df=0;: Initializes the 'df' attribute, representing the dead flag (0 for alive, 1 for dead), to 0.
 - EmptySensor.type='N';: Initializes the type of the sensor node to 'N', indicating it's a normal node.
 - EmptySensor.E=0;: Initializes the initial energy level of the sensor node to 0.
 - EmptySensor.id=0;: Initializes the unique identifier of the sensor node to 0.
 - EmptySensor.dis2sink=0;: Initializes the distance of the sensor node to the sink (or base station) to 0.
 - EmptySensor.dis2ch=0;: Initializes the distance of the sensor node to its cluster head to 0.
 - EmptySensor.MCH=n+1;: Initializes the member of CH (Cluster Head) to n+1, indicating that the node is not a member of any cluster initially.



The screenshot shows the MATLAB IDE interface with the following details:

- Toolbar:** HOME, PLOTS, APPS, EDITOR (highlighted in blue), PUBLISH, FILE VERSIONS, VIEW.
- File Explorer:** Shows files in the 'ILEACH-master' folder, including ILEACH.m, ILEACH_configureSensors.m, sendReceivePack.m, resetSensors.m, Locations.mat, joinToNearestCH.r, ILEACH.m, ILEACH_setParam.m, ILEACH_selectCH.m, ILEACH_plotter.m, and ILEACH_configure.m.
- Editor Tab:** Displays the code for ILEACH_configureSensors.m.
- Code Content:**

```
%The 'ILEACH_configureSensors' function is responsible for configuring the sensor n
% Model: Structure containing simulation parameters.
% n: Number of sensor nodes.
% GX, GY: Arrays containing the x and y coordinates of the sensor nodes.

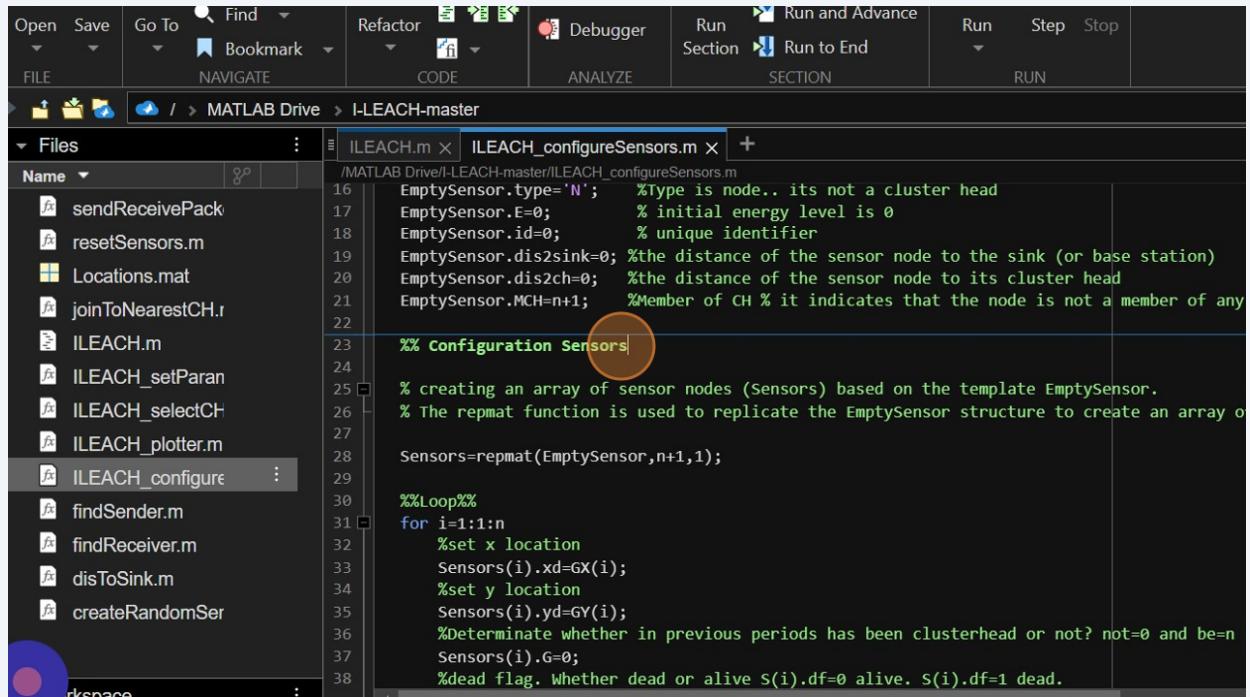
function Sensors=ILEACH_configureSensors(Model,n,GX,GY)

%A structure defining the configuration of an empty sensor node with
%default values for various parameters.

%% Configuration EmptySensor
EmptySensor.xd=0; %x-dimention
EmptySensor.yd=0; % y-dimention
EmptySensor.G=0; %The 'G' attribute represents whether the sensor node has b
EmptySensor.df=0; %The df attribute is a dead flag, indicating whether the s
EmptySensor.type='N'; %type is node.. its not a cluster head
EmptySensor.E=0; % initial energy level is 0
EmptySensor.id=0; % unique identifier
EmptySensor.dis2sink=0; %the distance of the sensor node to the sink (or base stati
```

25

- Sensors=repmat(EmptySensor,n+1,1);: Creates an array of sensor nodes (Sensors) based on the template EmptySensor. The repmat function replicates the EmptySensor structure to create an array of size (n+1) x 1.



The screenshot shows the MATLAB IDE interface with the following details:

- Toolbar:** Open, Save, Go To, Find, Refactor, Debugger, Run Section, Run and Advance, Run to End, Run, Step, Stop, RUN.
- File Explorer:** Shows files in the 'ILEACH-master' folder, including ILEACH.m, ILEACH_configureSensors.m, and several .m and .mat files.
- Code Editor:** Displays the content of ILEACH_configureSensors.m. A red circle highlights the line of code:

```
Sensors=repmat(EmptySensor,n+1,1);
```
- Code Content:**

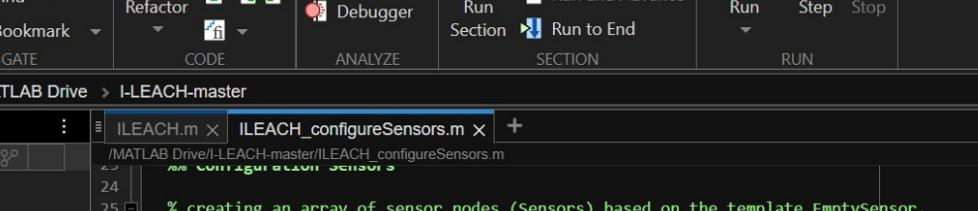
```

16 EmptySensor.type='N'; %type is node.. its not a cluster head
17 EmptySensor.E=0; % initial energy level is 0
18 EmptySensor.id=0; % unique identifier
19 EmptySensor.dis2sink=0; %the distance of the sensor node to the sink (or base station)
20 EmptySensor.dis2ch=0; %the distance of the sensor node to its cluster head
21 EmptySensor.MCH=n+1; %Member of CH % it indicates that the node is not a member of any
22
23 %% Configuration Sensors
24
25 % creating an array of sensor nodes (Sensors) based on the template EmptySensor.
26 % The repmat function is used to replicate the EmptySensor structure to create an array o
27
28 Sensors=repmat(EmptySensor,n+1,1);
29
30 %%Loop%%
31 for i=1:1:n
32     %set x location
33     Sensors(i).xd=GX(i);
34     %set y location
35     Sensors(i).yd=GY(i);
36     %Determinate whether in previous periods has been clusterhead or not? not=0 and be=n
37     Sensors(i).G=0;
38     %dead flag. Whether dead or alive S(i).df=0 alive. S(i).df=1 dead.

```

26

- for i=1:n: Iterates over all nodes.
 - Sensors(i).xd=GX(i);: Sets the x-location of the i-th sensor node.
 - Sensors(i).yd=GY(i);: Sets the y-location of the i-th sensor node.
 - Sensors(i).G=0;: Initializes whether the sensor node has been a cluster head in previous periods to 0.
 - Sensors(i).df=0;: Initializes the dead flag to 0, indicating the node is alive.
 - Sensors(i).type='N';: Initializes the type of the sensor node to 'N', indicating it's a normal node.
 - Sensors(i).E=Model.Eo;: Initializes the energy of the sensor node with the initial energy specified in the model parameters.
 - Sensors(i).id=i;: Sets the unique identifier of the sensor node.
 - Sensors(i).RR=Model.RR;: Sets the radio range of the sensor node based on the model parameters.



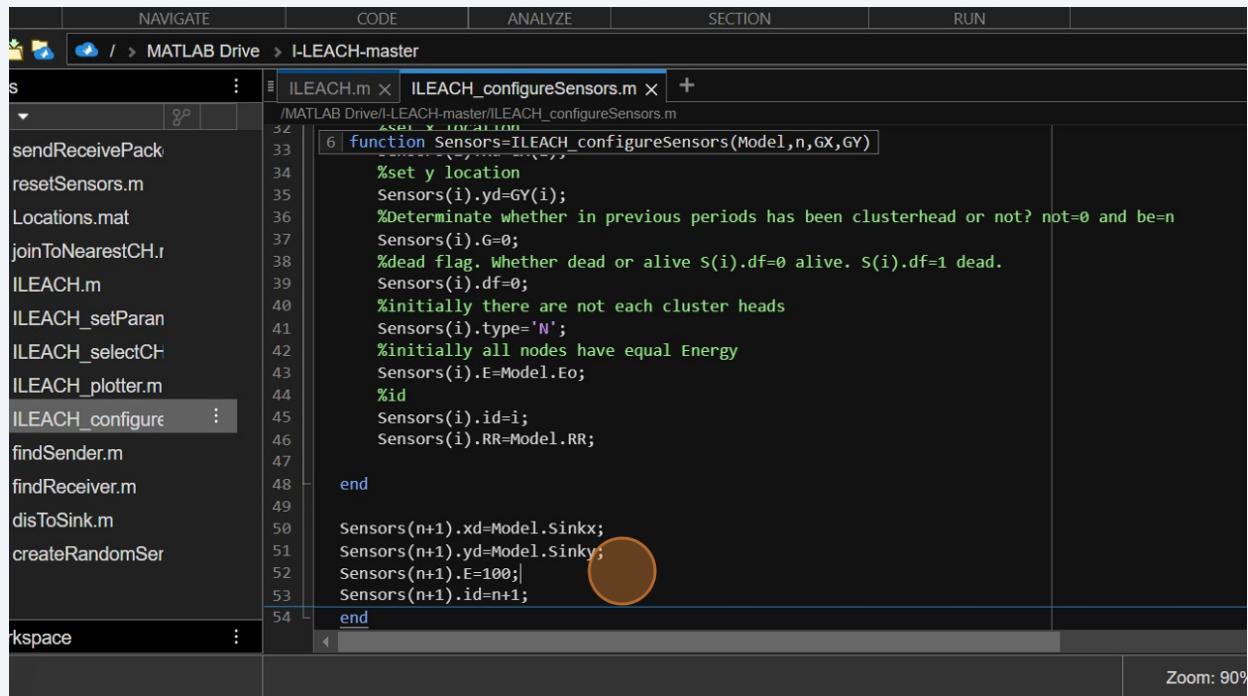
The screenshot shows the MATLAB IDE interface. The top menu bar includes 'File', 'Edit', 'Home', 'File Explorer', 'Variables', 'Run', 'Help', and 'Help Center'. The toolbar has icons for Go To, Find, Refactor, Code Issues, Debugger, Run Section, Run and Advance, Run to End, Run to Section, Run, Step, Stop, and RUN. The left sidebar shows a navigation tree for 'MATLAB Drive > I-LEACH-master' and lists files like 'IReceivePack.m', 'tSensors.m', 'tions.mat', 'toNearestCH.r', 'CH.m', 'CH_setParan.m', 'CH_selectCH.m', 'CH_plotter.m', 'CH_configure.m', 'Sender.m', 'Receiver.m', 'pSink.m', and 'teRandomSer'. The main workspace shows two open files: 'ILEACH.m' and 'ILEACH_configureSensors.m'. The code in 'ILEACH_configureSensors.m' is as follows:

```
% creating an array of sensor nodes (Sensors) based on the template EmptySensor.  
% The repmat function is used to replicate the EmptySensor structure to create an array of size (n+1)  
  
Sensors=repmat(EmptySensor,n+1,1);  
  
%%Loop%%  
for i=1:n  
    %set x location  
    Sensors(i).xd=GX(i);  
    %set y location  
    Sensors(i).yd=GY(i);  
    %Determinate whether in previous periods has been clusterhead or not? not=0 and be=n  
    Sensors(i).G=0;  
    %dead flag. Whether dead or alive S(i).df=0 alive. S(i).df=1 dead.  
    Sensors(i).df=0;  
    %Initially there are not each cluster heads  
    Sensors(i).type='N';  
    %Initially all nodes have equal Energy  
    Sensors(i).E=Model.Eo;
```

27

- Sensors(n+1).xd=Model.Sinkx;; Sets the x-location of the sink (base station).
- Sensors(n+1).yd=Model.Sinky;; Sets the y-location of the sink (base station).
- Sensors(n+1).E=100;; Initializes the energy of the sink (base station) to 100.
- Sensors(n+1).id=n+1;; Sets the unique identifier of the sink (base station).

The function returns the configured array of sensor nodes (Sensors).



The screenshot shows the MATLAB IDE interface with the following details:

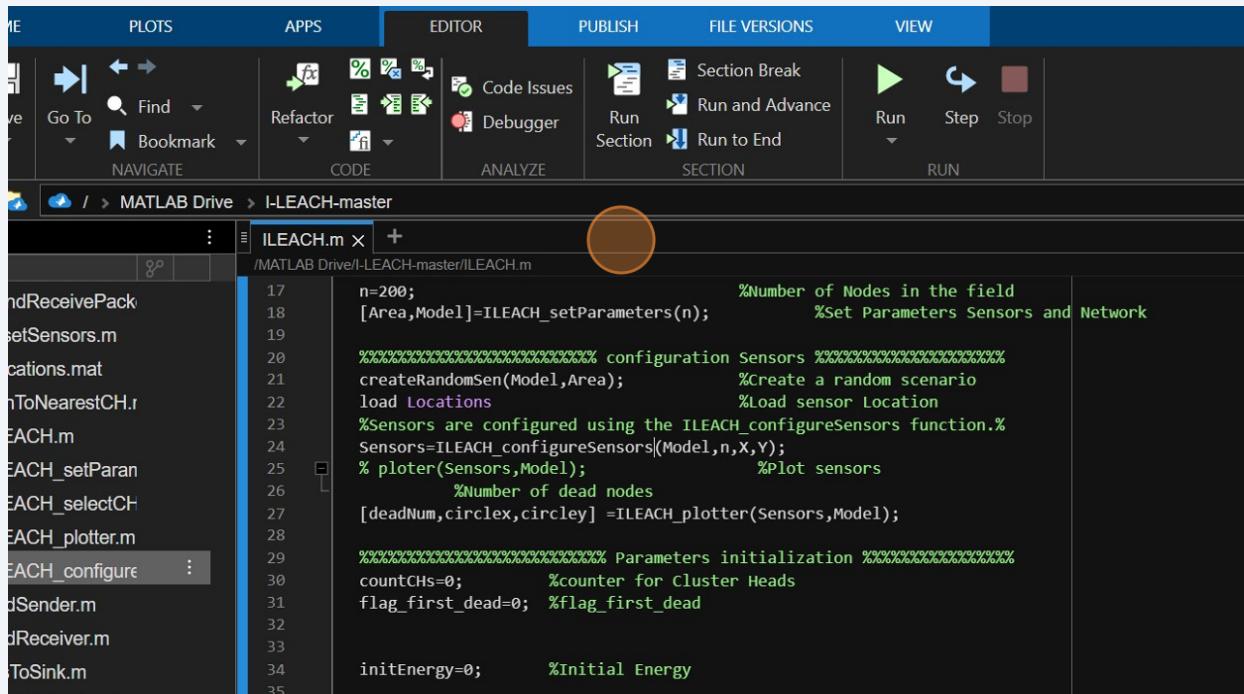
- Toolbar:** NAVIGATE, CODE, ANALYZE, SECTION, RUN.
- Path:** / > MATLAB Drive > I-LEACH-master
- Code Editor:** The file ILEACH_configureSensors.m is open. The code defines a function Sensors = ILEACH_configureSensors(Model, n, GX, GY). It initializes sensor parameters like y location, clusterhead status, dead/alive flag, type, energy, and ID, and sets the sink's location, energy, and ID.
- Code Content:**

```
function Sensors=ILEACH_configureSensors(Model,n,GX,GY)
    %set y location
    Sensors(i).yd=GY(i);
    %Determinate whether in previous periods has been clusterhead or not? not=0 and be=1
    Sensors(i).G=0;
    %dead flag. Whether dead or alive s(i).df=0 alive. s(i).df=1 dead.
    Sensors(i).df=0;
    %initially there are not each cluster heads
    Sensors(i).type='N';
    %initially all nodes have equal Energy
    Sensors(i).E=Model.Eo;
    %id
    Sensors(i).id=i;
    Sensors(i).RR=Model.RR;

end

Sensors(n+1).xd=Model.Sinkx;
Sensors(n+1).yd=Model.Sinky;
Sensors(n+1).E=100;
Sensors(n+1).id=n+1;
```
- Toolbars:** Standard MATLAB toolbars for file operations, search, and help.
- Status Bar:** Zoom: 90%

28 Now lets move back to Main file to observe next step



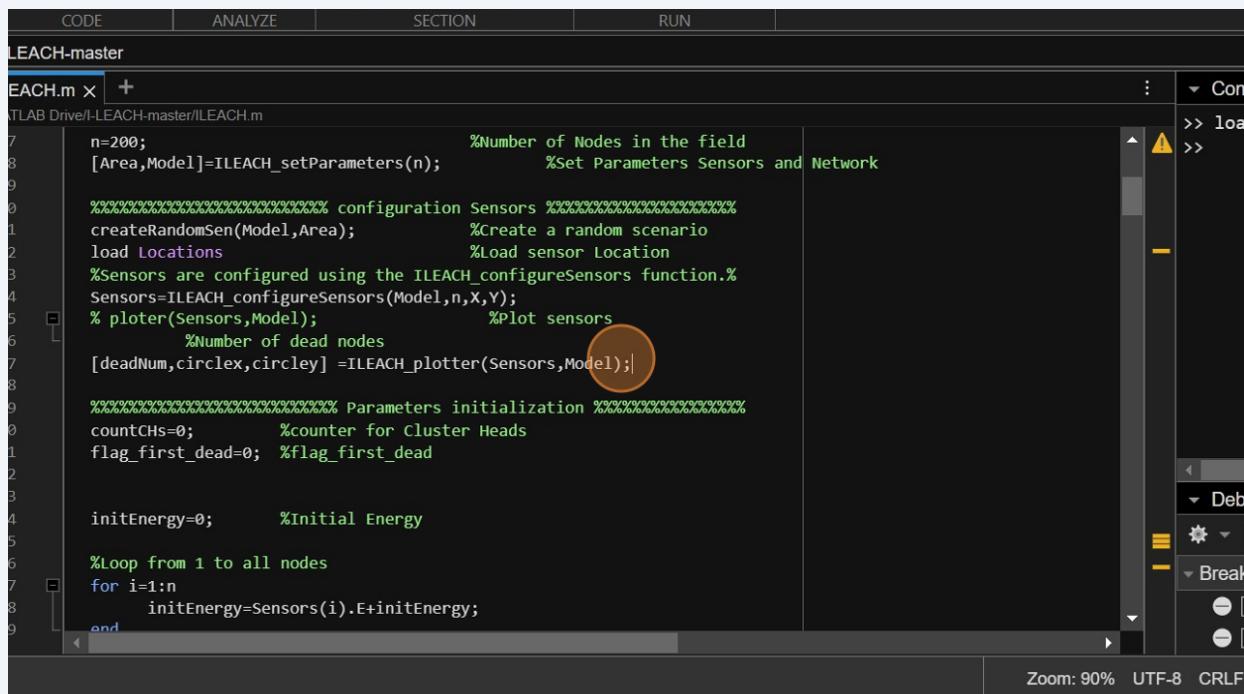
```
n=200; %Number of Nodes in the field
[Area,Model]=ILEACH_setParameters(n); %Set Parameters Sensors and Network

%%%%%%%%%%%%% configuration Sensors %%%%%%
createRandomSen(Model,Area); %Create a random scenario
load Locations %Load sensor Location
%Sensors are configured using the ILEACH_configureSensors function.%
Sensors=ILEACH_configureSensors(Model,n,X,Y);
% ploter(Sensors,Model); %plot sensors
%Number of dead nodes
[deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model);

%%%%%%%%%%%%% Parameters initialization %%%%%%
countCHs=0; %counter for Cluster Heads
flag_first_dead=0; %flag_first_dead

initEnergy=0; %Initial Energy
```

29 After the configureSensors plotter function is used that takes two parameters



```
n=200; %Number of Nodes in the field
[Area,Model]=ILEACH_setParameters(n); %Set Parameters Sensors and Network

%%%%%%%%%%%%% configuration Sensors %%%%%%
createRandomSen(Model,Area); %Create a random scenario
load Locations %Load sensor Location
%Sensors are configured using the ILEACH_configureSensors function.%
Sensors=ILEACH_configureSensors(Model,n,X,Y);
% ploter(Sensors,Model); %plot sensors
%Number of dead nodes
[deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model); %Red circle here

%%%%%%%%%%%%% Parameters initialization %%%%%%
countCHs=0; %counter for Cluster Heads
flag_first_dead=0; %flag_first_dead

initEnergy=0; %Initial Energy

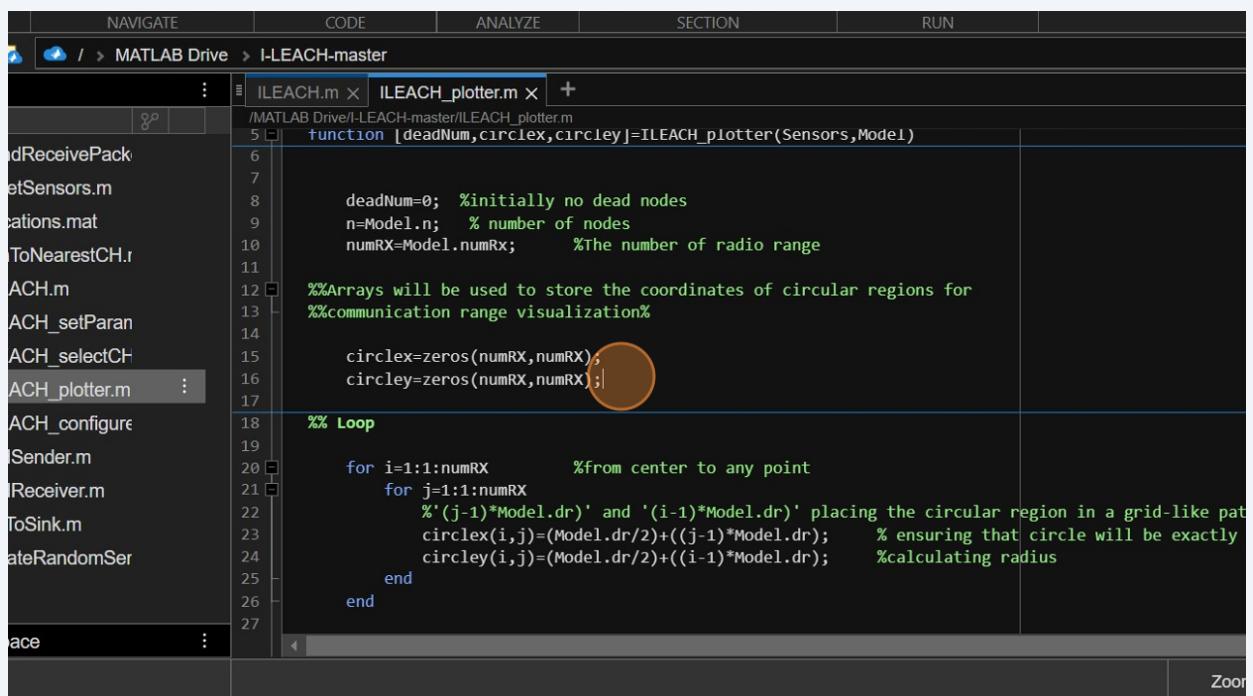
%Loop from 1 to all nodes
for i=1:n
    initEnergy=Sensors(i).E+initEnergy;
end
```

30

Now lets observe this function code and working

This function, ILEACH_plotter, is responsible for visualizing the state of the sensor network in the LEACH simulation. Let's break down the code:

- deadNum=0; Initializes a variable deadNum to 0, which will be used to count the number of dead nodes.
- n=Model.n; Retrieves the total number of nodes from the model parameters.
- numRX=Model.numRx; Retrieves the number of circular regions (radio range) from the model parameters.
- circlex=zeros(numRX,numRX); initializes a 2D array circlex with zeros to store the x-coordinates of circular regions for communication range visualization.
- circley=zeros(numRX,numRX); initializes a 2D array circley with zeros to store the y-coordinates of circular regions for communication range visualization



```

NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master
ILEACH.m x ILEACH_plotter.m +
/MATLAB Drive/I-LEACH-master/ILEACH_plotter.m
5 function [deadNum,circlex,circley]=ILEACH_plotter(Sensors,Model)
6
7
8     deadNum=0; %initially no dead nodes
9     n=Model.n; % number of nodes
10    numRX=Model.numRx; %The number of radio range
11
12 %%Arrays will be used to store the coordinates of circular regions for
13 %%communication range visualization%
14
15     circlex=zeros(numRX,numRX);
16     circley=zeros(numRX,numRX);
17
18 %% Loop
19
20 for i=1:1:numRX %from center to any point
21     for j=1:1:numRX
22         %(j-1)*Model.dr)' and '(i-1)*Model.dr)' placing the circular region in a grid-like pat
23         circlex(i,j)=(Model.dr/2)+((j-1)*Model.dr); % ensuring that circle will be exactly
24         circley(i,j)=(Model.dr/2)+((i-1)*Model.dr); %calculating radius
25     end
26 end
27

```

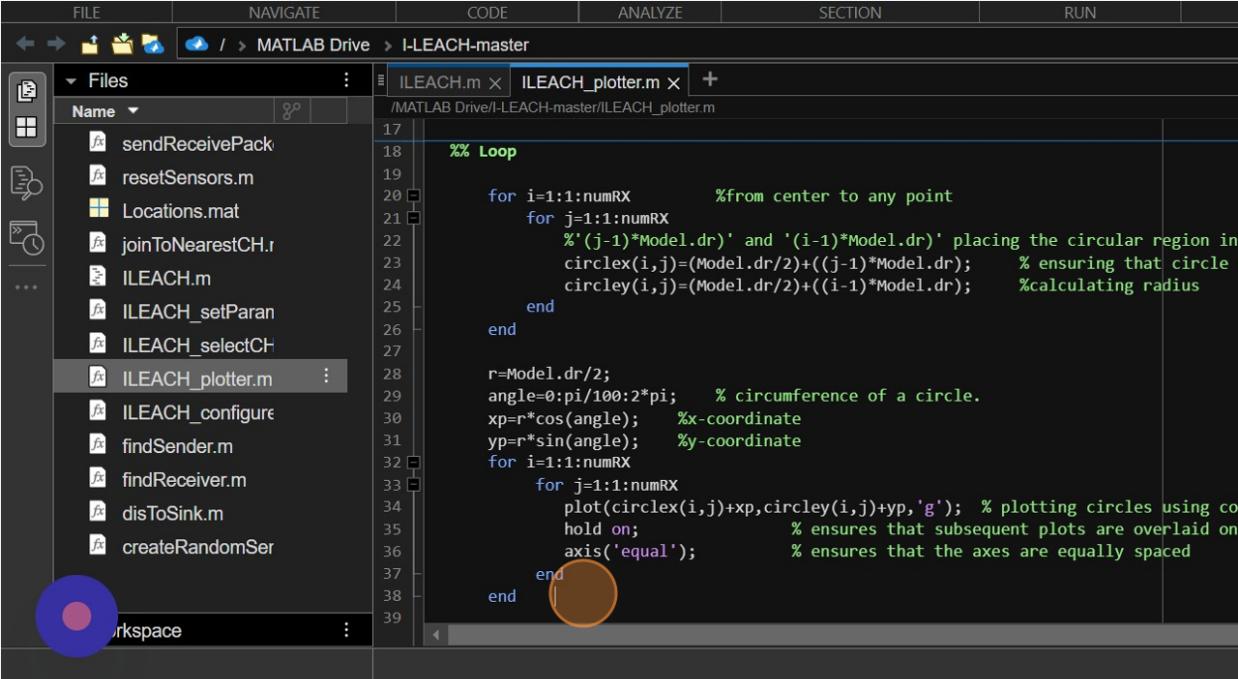
31

Inside the %Loop

This section of code is responsible for generating and plotting circular regions (communication ranges) in the sensor network simulation. Let's break it down:

- for $i=1:1:numRX$: This loop iterates through the rows of the circular regions.
- for $j=1:1:numRX$: This nested loop iterates through the columns of the circular regions.
- $\text{circlex}(i,j)=(\text{Model.dr}/2)+((j-1)*\text{Model.dr});$: Calculates the x-coordinate of the center of each circular region based on the grid-like pattern.
- $\text{circley}(i,j)=(\text{Model.dr}/2)+((i-1)*\text{Model.dr});$: Calculates the y-coordinate of the center of each circular region based on the grid-like pattern.

After executing these loops, the code generates a grid of circular regions based on the specified communication range (Model.dr) and the number of circular regions (numRX).



```

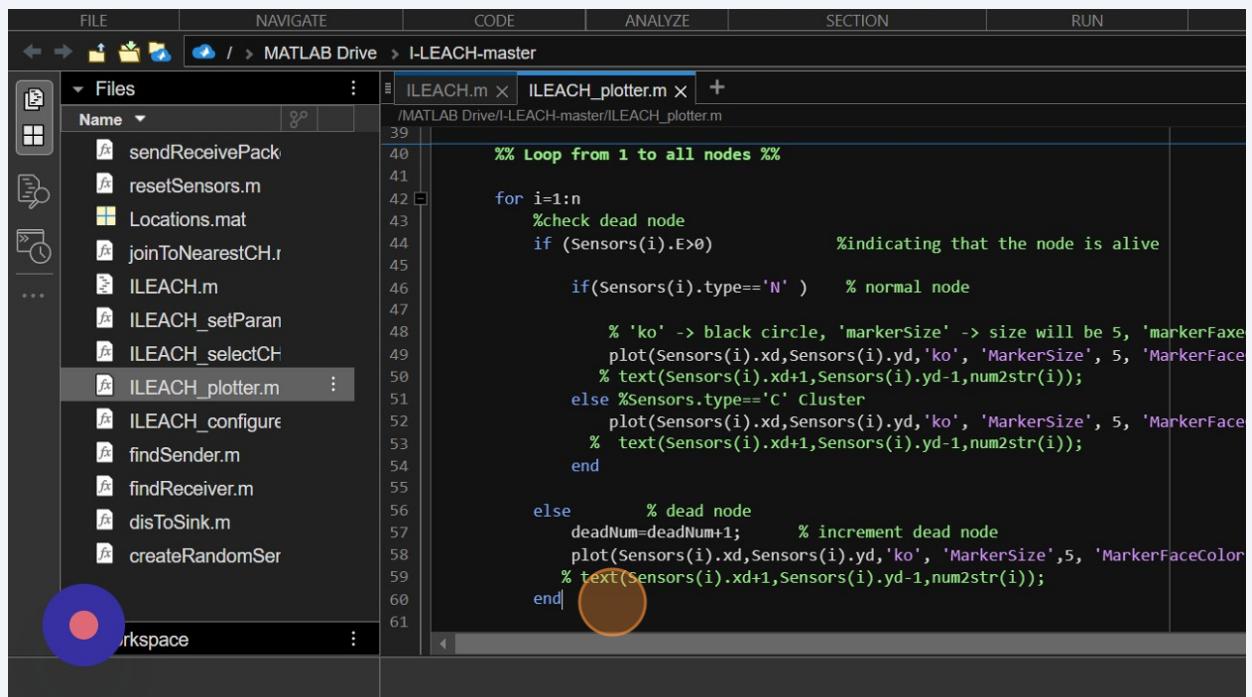
FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master
Files ILEACH.m × ILEACH_plotter.m × +
Name
sendReceivePack
resetSensors.m
Locations.mat
joinToNearestCH.r
ILEACH.m
ILEACH_setParam
ILEACH_selectCH
ILEACH_plotter.m : ILEACH_configure
findSender.m
findReceiver.m
disToSink.m
createRandomSer
Workspace :
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
%% Loop
for i=1:1:numRX %from center to any point
    for j=1:1:numRX %'(j-1)*Model.dr)' and '(i-1)*Model.dr)' placing the circular region in
        circlex(i,j)=(Model.dr/2)+((j-1)*Model.dr); % ensuring that circle
        circley(i,j)=(Model.dr/2)+((i-1)*Model.dr); %calculating radius
    end
end
r=Model.dr/2;
angle=0:pi/100:2*pi; % circumference of a circle.
xp=r*cos(angle); %x-coordinate
yp=r*sin(angle); %y-coordinate
for i=1:1:numRX
    for j=1:1:numRX
        plot(circlex(i,j)+xp,circley(i,j)+yp,'g'); % plotting circles using co
        hold on; % ensures that subsequent plots are overlaid on
        axis('equal'); % ensures that the axes are equally spaced
    end
end

```

32

- The loop iterates through all nodes (for i=1:n).
- It checks if the node is alive (if (Sensors(i).E > 0)). If yes, it further checks whether the node is a normal node or a cluster head.
 - If it's a normal node (if (Sensors(i).type == 'N')), it plots a black circle ('ko') with a size of 5 and a black filled circle ('MarkerFaceColor', 'k') with a size of 5.
 - If it's a cluster head (else), it plots a black circle ('ko') with a size of 5 and a red filled circle ('MarkerFaceColor', 'r') with a size of 5.
- If the node is dead (else), it increments the deadNum counter and plots a black circle ('ko') with a size of 5 and a white filled circle ('MarkerFaceColor', 'w') with a size of 5.

This section visually represents the state of each node in the simulation based on its status and type. The color and style of the circles indicate whether the node is alive, dead, a normal node, or a cluster head.



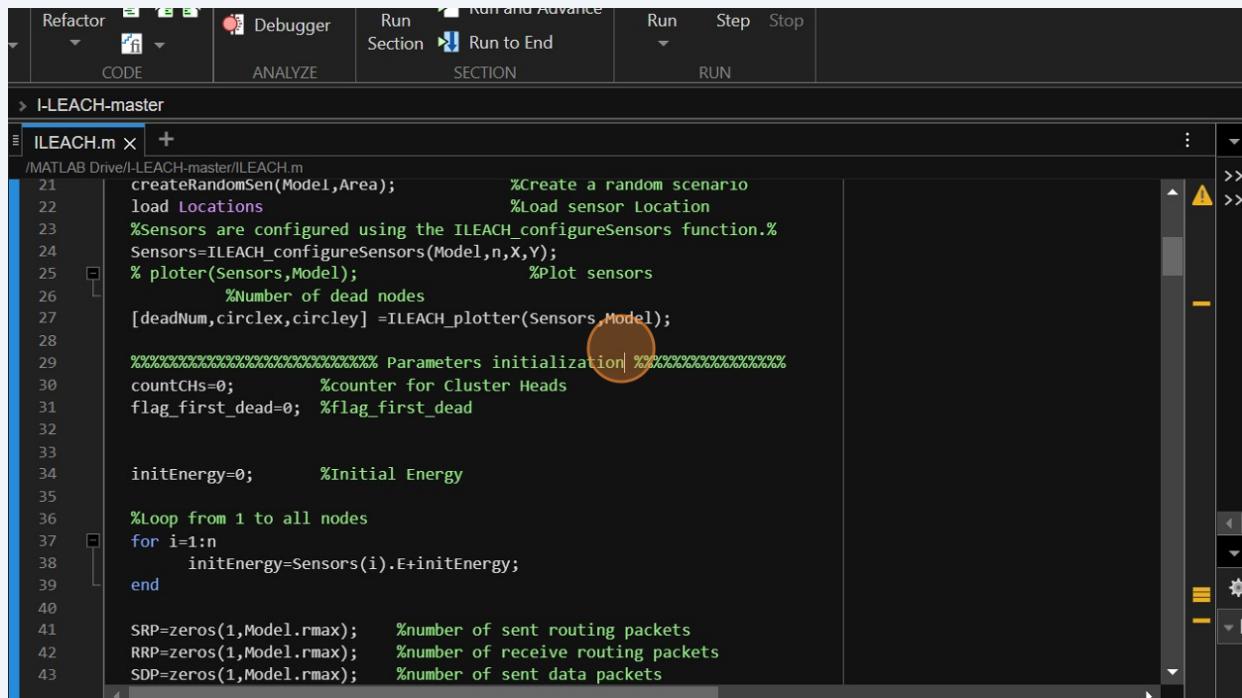
The screenshot shows the MATLAB IDE interface with the following details:

- Toolbar:** FILE, NAVIGATE, CODE, ANALYZE, SECTION, RUN.
- File Explorer:** Shows files in the 'I-LEACH-master' folder: sendReceivePack, resetSensors.m, Locations.mat, joinToNearestCH.r, ILEACH.m, ILEACH_setParam, ILEACH_selectCH, ILEACH_plotter.m (selected), ILEACH_configure, findSender.m, findReceiver.m, disToSink.m, createRandomSer.
- Editor:** Displays the code for ILEACH_plotter.m. The code is as follows:

```
%>>> %% Loop from 1 to all nodes %%
for i=1:n
    %check dead node
    if (Sensors(i).E>0) %indicating that the node is alive
        if(Sensors(i).type=='N') % normal node
            % 'ko' -> black circle, 'markerSize' -> size will be 5, 'MarkerFaceColor' -> 'k'
            plot(Sensors(i).xd,Sensors(i).yd,'ko', 'MarkerSize', 5, 'MarkerFaceColor', 'k');
            % text(Sensors(i).xd+1,Sensors(i).yd-1,num2str(i));
        else %Sensors.type=='C' Cluster
            % 'ko' -> black circle, 'markerSize' -> size will be 5, 'MarkerFaceColor' -> 'r'
            plot(Sensors(i).xd,Sensors(i).yd,'ko', 'MarkerSize', 5, 'MarkerFaceColor', 'r');
            % text(Sensors(i).xd+1,Sensors(i).yd-1,num2str(i));
        end
    else % dead node
        deadNum=deadNum+1; % increment dead node
        plot(Sensors(i).xd,Sensors(i).yd,'ko', 'MarkerSize',5, 'MarkerFaceColor', 'w');
        % text(Sensors(i).xd+1,Sensors(i).yd-1,num2str(i));
    end
end
```

33

Now back to the Main ILEACH file code
We have a Parameters Initialization part



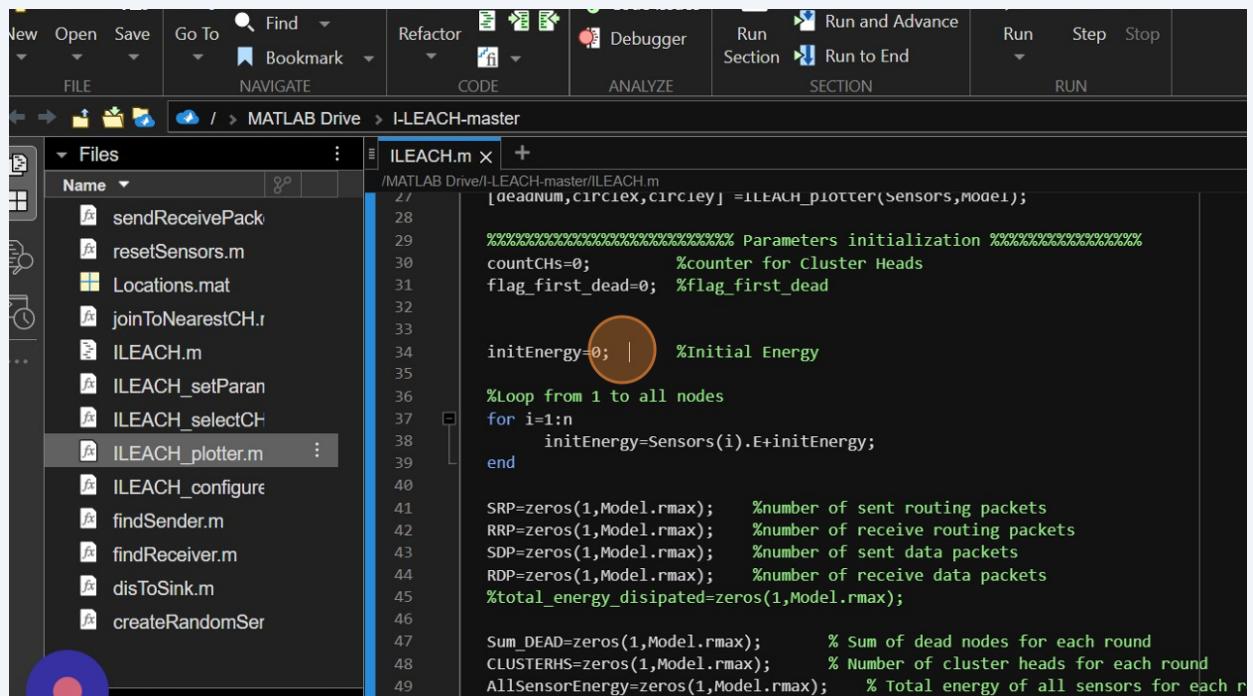
```
Refactor Debugger Run and Advance
CODE ANALYZE Run Step Stop
SECTION RUN

> I-LEACH-master
ILEACH.m x +
/MATLAB Drive/I-LEACH-master/ILEACH.m
21 createRandomSen(Model,Area); %Create a random scenario
22 load Locations %Load sensor Location
23 %Sensors are configured using the ILEACH_configureSensors function.%
24 Sensors=ILEACH_configureSensors(Model,n,X,Y);
25 % ploter(Sensors,Model); %Plot sensors
26 %Number of dead nodes
27 [deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model);
28
29 %%%%%%%%%%%%%% Parameters initialization %%%%%%%%%%%%%%
30 countCHs=0; %counter for Cluster Heads
31 flag_first_dead=0; %flag_first_dead
32
33
34 initEnergy=0; %Initial Energy
35
36 %Loop from 1 to all nodes
37 for i=1:n
38     initEnergy=Sensors(i).E+initEnergy;
39 end
40
41 SRP=zeros(1,Model.rmax); %number of sent routing packets
42 RRP=zeros(1,Model.rmax); %number of receive routing packets
43 SDP=zeros(1,Model.rmax); %number of sent data packets
```

34

- countCHs: It is a counter that keeps track of the number of cluster heads in the network.
- flag_first_dead: This is a flag used to indicate whether the first node in the network has died. It's initialized to 0.
- initEnergy: This variable is set to 0 and is labeled as "Initial Energy." It might be used for some calculations or comparisons later in the code.

These variables are initialized at the beginning of the simulation and are likely to be updated and used throughout the execution of the code. The exact purpose of these variables will become clearer as we analyze more parts of the code.

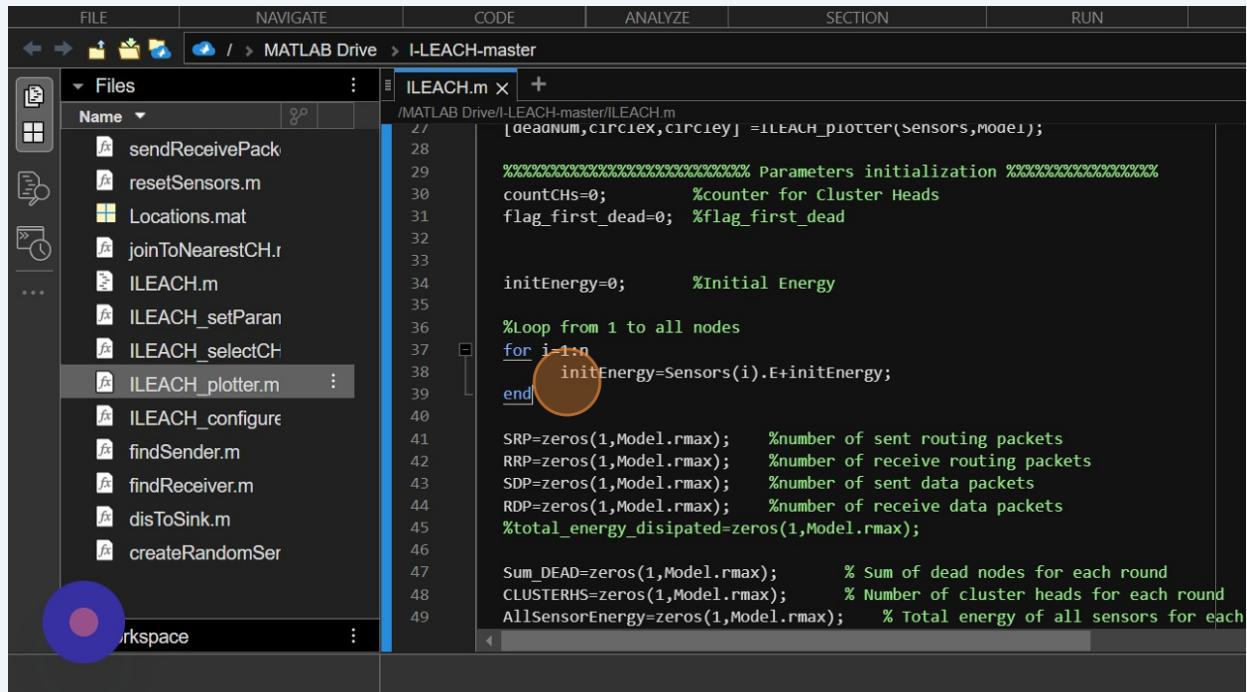


```
ILEACH.m x +  
/MATLAB Drive/I-LEACH-master/ILEACH.m  
27  
28  
29  
30 countCHs=0; %counter for Cluster Heads  
31 flag_first_dead=0; %flag_first_dead  
32  
33  
34 initEnergy=0; | %Initial Energy  
35  
36 %Loop from 1 to all nodes  
37 for i=1:n  
38     initEnergy=Sensors(i).E+initEnergy;  
39 end  
40  
41 SRP=zeros(1,Model.rmax); %number of sent routing packets  
42 RRP=zeros(1,Model.rmax); %number of receive routing packets  
43 SDP=zeros(1,Model.rmax); %number of sent data packets  
44 RDP=zeros(1,Model.rmax); %number of receive data packets  
45 %total_energy_dissipated=zeros(1,Model.rmax);  
46  
47 Sum_DEAD=zeros(1,Model.rmax); % Sum of dead nodes for each round  
48 CLUSTERHS=zeros(1,Model.rmax); % Number of cluster heads for each round  
49 AllSensorEnergy=zeros(1,Model.rmax); % Total energy of all sensors for each r
```

35

- It uses a loop to iterate over all sensor nodes in the network (for i=1:n).
- For each node, it adds its initial energy level (Sensors(i).E) to the variable initEnergy.
- The result is the sum of the initial energy levels of all sensor nodes in the network.

This sum can be used for various purposes in the simulation, such as calculating the total initial energy of the network or analyzing energy distribution among nodes.



The screenshot shows the MATLAB IDE interface with the following details:

- File Explorer:** Shows files in the "ILEACH-master" folder, including ILEACH.m, ILEACH_plotter.m, and several .m and .mat files.
- Current File:** ILEACH.m is open in the editor.
- Code Editor Content:**

```
% MATLAB Drive/I-LEACH-master/ILEACH.m
27
28 [deadnum,circlex,circley] =ILEACH_plotter(Sensors,Model);
29 %%%%%%%%%%%%%% Parameters initialization %%%%%%
30 countCHs=0; %counter for Cluster Heads
31 flag_first_dead=0; %flag_first_dead
32
33
34 initEnergy=0; %Initial Energy
35
36 %Loop from 1 to all nodes
37 for i=1:n
38     initEnergy=initEnergy+Sensors(i).E;
39 end
```

A red circle highlights the line "initEnergy=initEnergy+Sensors(i).E;" which is part of the loop.
- Toolbars and Menus:** FILE, NAVIGATE, CODE, ANALYZE, SECTION, RUN.

36

This section initializes several arrays to keep track of various metrics over the rounds of the simulation:

- SRP: Number of sent routing packets per round.
- RRP: Number of received routing packets per round.
- SDP: Number of sent data packets per round.
- RDP: Number of received data packets per round.

These arrays are preallocated with zeros and will be updated during the simulation to record the corresponding metrics for each round (Model.rmax rounds). These metrics are essential for evaluating the performance of the LEACH protocol and understanding the communication dynamics in the sensor network. The commented line at the end (%total_energy_dissipated=zeros(1,Model.rmax);) suggests that there might be an additional metric related to total energy dissipation that is currently commented out.

```
TE CODE ANALYZE SECTION RUN
AB Drive > I-LEACH-master
ILEACH.m X +
/MATLAB Drive/I-LEACH-master/ILEACH.m
30    countCHs=0; %counter for cluster heads
31    flag_first_dead=0; %flag_first_dead
32
33
34    initEnergy=0; %Initial Energy
35
36    %Loop from 1 to all nodes
37    for i=1:n
38        initEnergy=Sensors(i).E+initEnergy;
39    end
40
41    SRP=zeros(1,Model.rmax); %number of sent routing packets
42    RRP=zeros(1,Model.rmax); %number of receive routing packets
43    SDP=zeros(1,Model.rmax); %number of sent data packets
44    RDP=zeros(1,Model.rmax); %number of receive data packets
45    %total_energy_dissipated=zeros(1,Model.rmax); | A red circle is placed here.
46
47    Sum_DEAD=zeros(1,Model.rmax); % Sum of dead nodes for each round
48    CLUSTERHS=zeros(1,Model.rmax); % Number of cluster heads for each round
49    AllSensorEnergy=zeros(1,Model.rmax); % Total energy of all sensors for each round
50
51    %%%%%%%%%%%%%% Start Simulation %%%%%%%%%%%%%%
52    global srp rrp sdp rdp %declearing variables
53
```

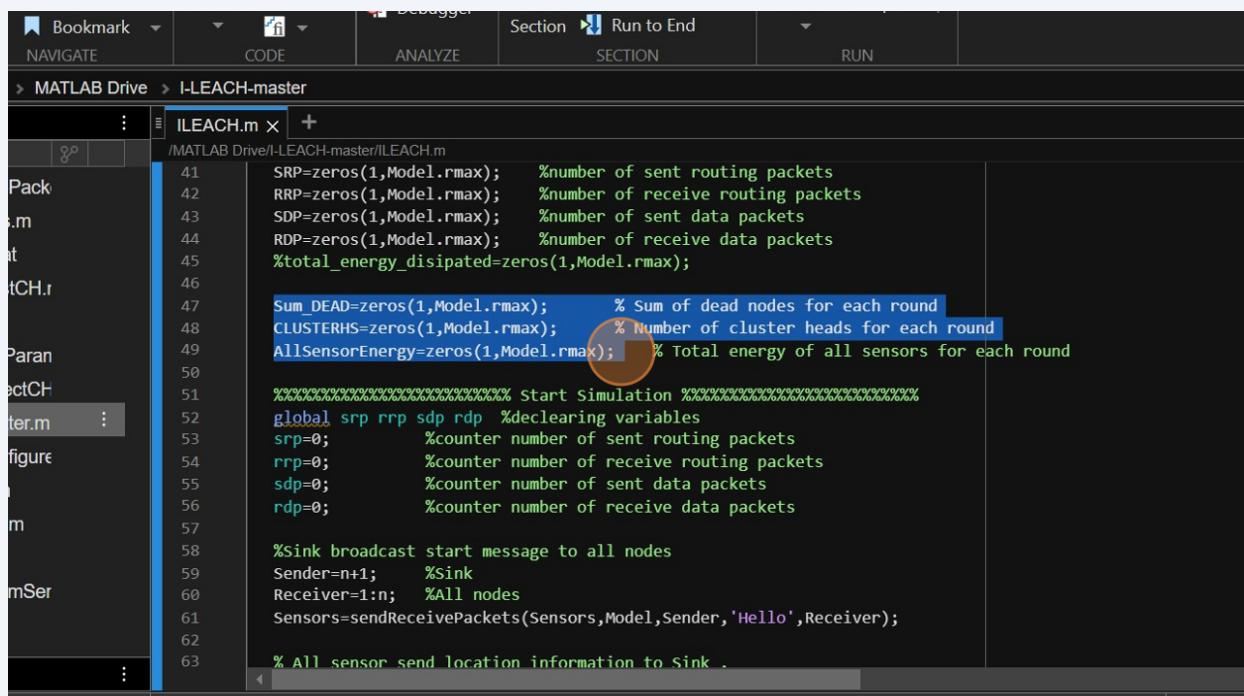
Zoom: 90%

37

This section initializes additional arrays to keep track of certain statistics over the rounds of the simulation:

- Sum_DEAD: Sum of dead nodes for each round.
- CLUSTERHS: Number of cluster heads for each round.
- AllSensorEnergy: Total energy of all sensors for each round.

Similar to the previous arrays, these are preallocated with zeros and will be updated during the simulation to record the corresponding statistics for each round (Model.rmax rounds). These metrics provide insights into the network's overall status, such as the number of dead nodes, the number of cluster heads, and the total energy of all sensors at each round.



```

Bookmarks | NAVIGATE | CODE | ANALYZE | Debugger | Section | Run to End | RUN
MATLAB Drive > I-LEACH-master
ILEACH.m
/MATLAB Drive/I-LEACH-master/ILEACH.m
41 SRP=zeros(1,Model.rmax); %number of sent routing packets
42 RRP=zeros(1,Model.rmax); %number of receive routing packets
43 SDP=zeros(1,Model.rmax); %number of sent data packets
44 RDP=zeros(1,Model.rmax); %number of receive data packets
45 %total_energy_disipated=zeros(1,Model.rmax);
46
47 Sum_DEAD=zeros(1,Model.rmax); % Sum of dead nodes for each round
48 CLUSTERHS=zeros(1,Model.rmax); % Number of cluster heads for each round
49 AllSensorEnergy=zeros(1,Model.rmax); % Total energy of all sensors for each round
50
51 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Start Simulation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
52 global srp rrp sdp rdp %declearing variables
53 srp=0; %counter number of sent routing packets
54 rrp=0; %counter number of receive routing packets
55 sdp=0; %counter number of sent data packets
56 rdp=0; %counter number of receive data packets
57
58 %Sink broadcast start message to all nodes
59 Sender=n+1; %Sink
60 Receiver=1:n; %All nodes
61 Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver);
62
63 % All sensor send location information to Sink .

```

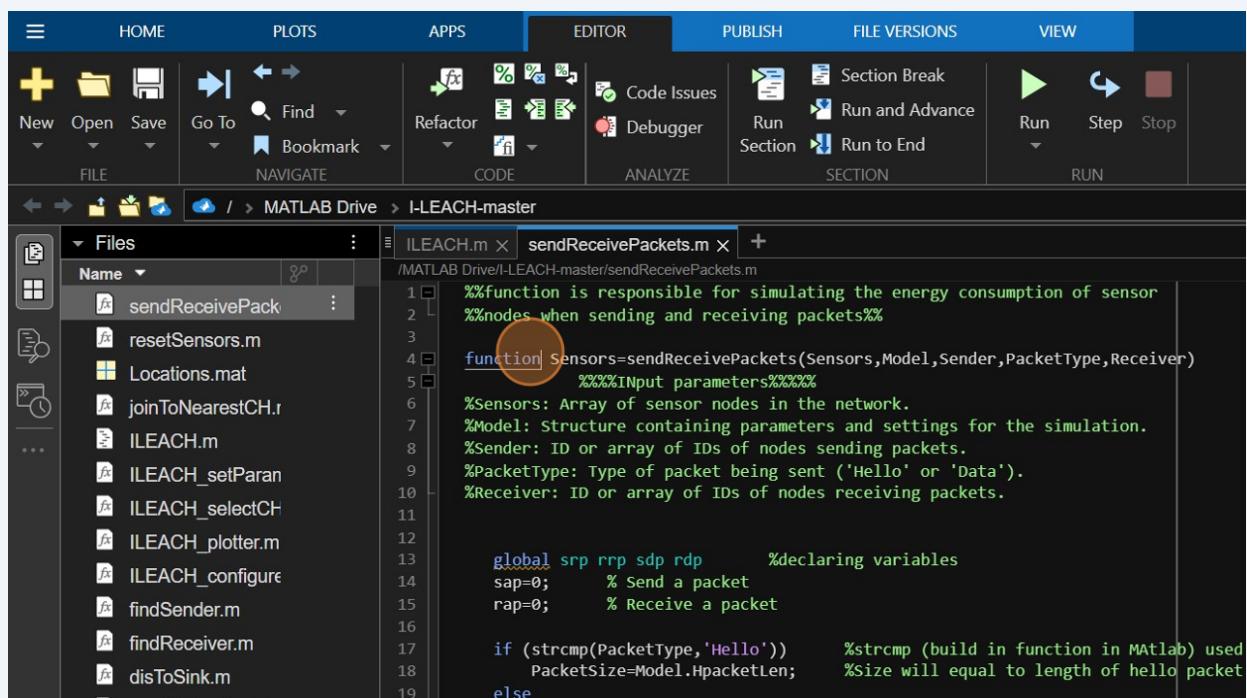
38

Now when sink broadcast start message to all nodes sendReceivePackets function is called and lets observe that function code file named sendReceivePackets

The sendReceivePackets function simulates the energy consumption of sensor nodes when sending and receiving packets. It takes the following input parameters:

- Sensors: Array of sensor nodes in the network.
- Model: Structure containing parameters and settings for the simulation.
- Sender: ID or array of IDs of nodes sending packets.
- PacketType: Type of packet being sent ('Hello' or 'Data').
- Receiver: ID or array of IDs of nodes receiving packets.

This function is crucial for modeling the energy consumption behavior of nodes during communication. Depending on the type of packet being sent ('Hello' or 'Data'), it adjusts the energy levels of sender and receiver nodes accordingly. The energy consumption is based on the transmission and reception costs specified in the energy model parameters.



The screenshot shows the MATLAB Editor interface with the 'EDITOR' tab selected. The current file is 'sendReceivePackets.m'. The code is as follows:

```
%function is responsible for simulating the energy consumption of sensor
%nodes when sending and receiving packets%
function Sensors=sendReceivePackets(Sensors,Model,Sender,PacketType,Receiver)
    %%Input parameters%%%%%
    %Sensors: Array of sensor nodes in the network.
    %Model: Structure containing parameters and settings for the simulation.
    %Sender: ID or array of IDs of nodes sending packets.
    %PacketType: Type of packet being sent ('Hello' or 'Data').
    %Receiver: ID or array of IDs of nodes receiving packets.

    global srp rrp sdp rdp      %declaring variables
    sap=0;          % Send a packet
    rap=0;          % Receive a packet

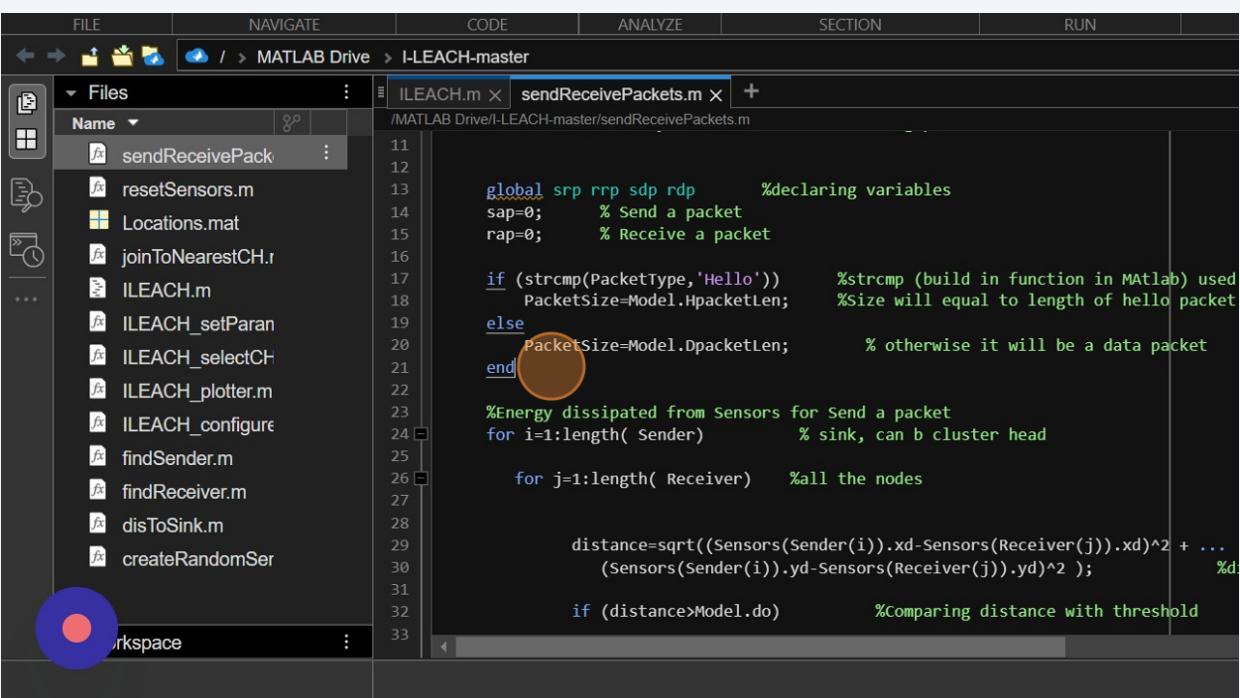
    if (strcmp(PacketType,'Hello'))      %strcmp (build in function in Matlab) used
        PacketSize=Model.HpacketLen;      %Size will equal to length of hello packet
    else

```

39

In this part of the code, the function is setting up some variables and conditions before proceeding with the simulation of sending and receiving packets.

- `global srp rrp sdp rdp`: This line declares global variables srp, rrp, sdp, and rdp, which are likely used to keep track of the number of sent routing packets, received routing packets, sent data packets, and received data packets, respectively. The use of global variables allows these counts to be accessed and modified across different functions or parts of the code.
- `sap=0;`: Initializes the variable sap (send a packet) to 0.
- `rap=0;`: Initializes the variable rap (receive a packet) to 0.
- The conditional statement `if (strcmp(PacketType,'Hello'))` checks whether the PacketType is equal to 'Hello'. strcmp is a built-in MATLAB function that compares two strings. If the condition is true, it sets PacketSize to Model.HpacketLen, which represents the size of a hello packet. Otherwise, it sets PacketSize to Model.DpacketLen, representing the size of a data packet



```

FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master
Files / MATLAB Drive/I-LEACH-master/sendReceivePackets.m
Name
sendReceivePack ...
resetSensors.m
Locations.mat
joinToNearestCH.r
I-LEACH.m
I-LEACH_setParam
I-LEACH_selectCH
I-LEACH_plotter.m
I-LEACH_configure
findSender.m
findReceiver.m
disToSink.m
createRandomSer
...
Workspace
I-LEACH.m x sendReceivePackets.m x + /MATLAB Drive/I-LEACH-master/sendReceivePackets.m
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
global srp rrp sdp rdp %declaring variables
sap=0; % Send a packet
rap=0; % Receive a packet

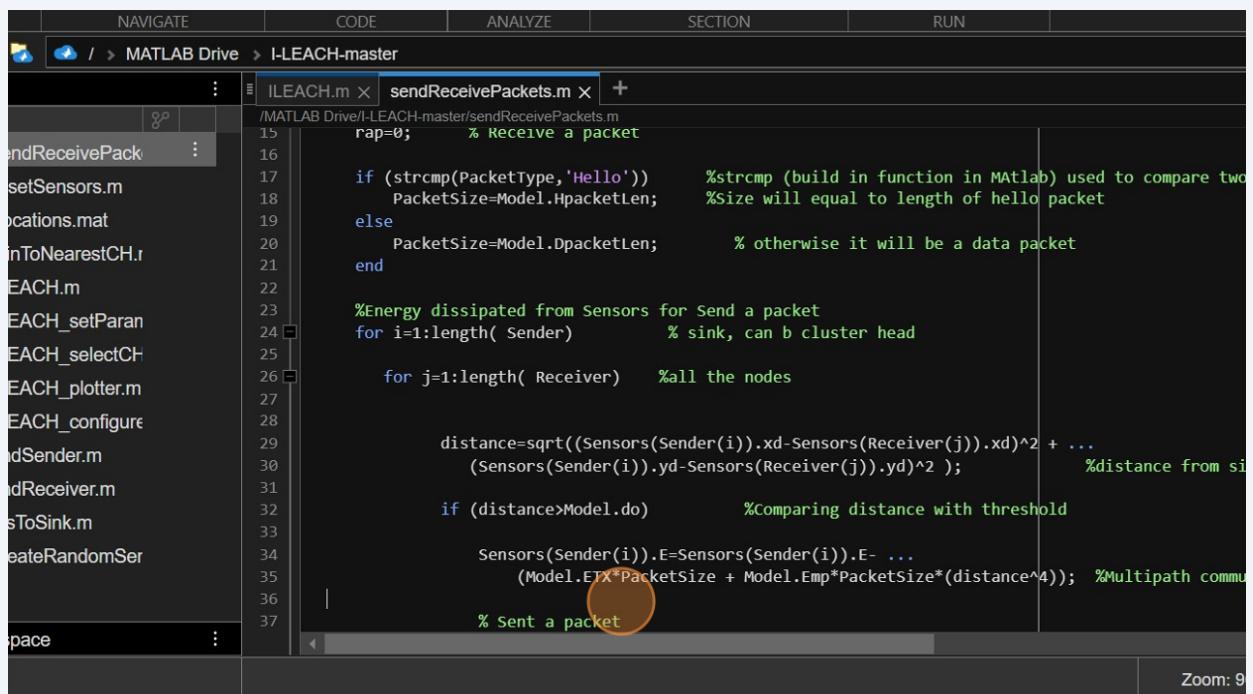
if (strcmp(PacketType, 'Hello')) %strcmp (build in function in Matlab) used
    PacketSize=Model.HpacketLen; %Size will equal to length of hello packet
else
    PacketSize=Model.DpacketLen; % otherwise it will be a data packet
end %Energy dissipated from Sensors for Send a packet
for i=1:length(Sender) % sink, can b cluster head
    for j=1:length(Receiver) %all the nodes
        distance=sqrt((Sensors(Sender(i)).xd-Sensors(Receiver(j)).xd)^2 + ... %d
                      (Sensors(Sender(i)).yd-Sensors(Receiver(j)).yd)^2 );
        if (distance>Model.do) %Comparing distance with threshold

```

40

This part of the code calculates the energy dissipated from sensors when sending a packet. Let's break down the key components:

- The code is using a nested loop where the outer loop (for $i=1:length(Sender)$) iterates over all elements in the Sender array, and the inner loop (for $j=1:length(Receiver)$) iterates over all elements in the Receiver array.
- For each combination of $Sender(i)$ and $Receiver(j)$, it calculates the distance (distance) between the sender and receiver using the Euclidean distance formula.
- The conditional statement if ($distance > Model.do$) checks whether the calculated distance is greater than a threshold value $Model.do$. If the distance is greater, it means that the communication is subject to multipath conditions.
- Inside the conditional block, it calculates the energy dissipated from the sender node ($Sensors(Sender(i)).E$) based on the energy model. The energy dissipation is calculated as a combination of energy consumed for transmitting a bit ($Model.ETX$), energy consumed for multipath channel transmission ($Model.Emp$), and the distance between the sender and receiver. This energy dissipation model reflects the impact of distance and channel conditions on energy consumption during communication.



```

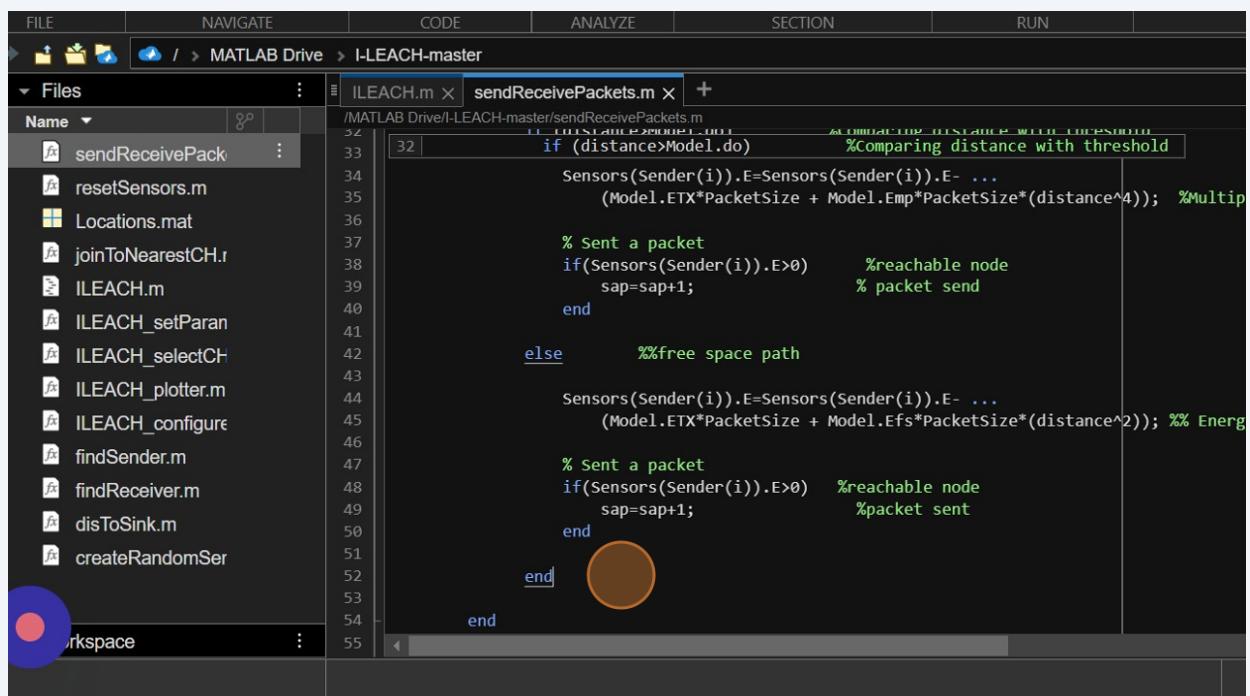
NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive / > I-LEACH-master
ILEACH.m x sendReceivePackets.m +
/MATLAB Drive/I-LEACH-master/sendReceivePackets.m
15 rap=0; % Receive a packet
16
17 if (strcmp(PacketType,'Hello')) %strcmp (build in function in Matlab) used to compare two
18 PacketSize=Model.HpacketLen; %Size will equal to length of hello packet
19 else
20 PacketSize=Model.DpacketLen; % otherwise it will be a data packet
21 end
22
23 %Energy dissipated from Sensors for Send a packet
24 for i=1:length(Sender) % sink, can b cluster head
25
26 for j=1:length(Receiver) %all the nodes
27
28
29 distance=sqrt((Sensors(Sender(i)).xd-Sensors(Receiver(j)).xd)^2 + ...
30 (Sensors(Sender(i)).yd-Sensors(Receiver(j)).yd)^2); %distance from si
31
32 if (distance>Model.do) %Comparing distance with threshold
33
34 Sensors(Sender(i)).E=Sensors(Sender(i)).E- ...
35 (Model.ETX*PacketSize + Model.Emp*PacketSize*(distance^4)); %Multipath communication
36
37 % Sent a packet

```

41

In this part of the code, it handles the scenario where the distance (distance) is less than or equal to the threshold (Model.do). This corresponds to the free-space path scenario.

- Inside the else block, it calculates the energy dissipated from the sender node (Sensors(Sender(i)).E) based on the energy model. This calculation considers the energy consumed for transmitting a bit (Model.ETX) and the energy consumption for free-space transmission (Model.Efs). The energy dissipation is proportional to the square of the distance between the sender and receiver.
- After calculating the energy dissipation, it checks if the sender node's energy level (Sensors(Sender(i)).E) is still greater than zero. If the energy level is greater than zero, it increments the sap counter, indicating that a packet has been successfully sent.



```

FILE NAVIGATE CODE ANALYZE SECTION RUN
> MATLAB Drive / > MATLAB Drive / I-LEACH-master
Files ILEACH.m x sendReceivePackets.m +
Name
sendReceivePackets.m
resetSensors.m
Locations.mat
joinToNearestCH.r
ILEACH.m
ILEACH_setParan
ILEACH_selectCH
ILEACH_plotter.m
ILEACH_configure
findSender.m
findReceiver.m
disToSink.m
createRandomSer
Workspace

/MATLAB Drive/I-LEACH-master/sendReceivePackets.m
32 | 32 | if (distance>Model.do) %Comparing distance with threshold
33 | Sensors(Sender(i)).E=Sensors(Sender(i)).E- ...
34 | (Model.ETX*PacketSize + Model.Emp*PacketSize*(distance^4)); %Multip
35 |
36 |
37 |
38 | % Sent a packet
39 | if(Sensors(Sender(i)).E>0) %reachable node
40 | sap=sap+1; % packet send
41 | end
42 |
43 |
44 | else %%free space path
45 | Sensors(Sender(i)).E=Sensors(Sender(i)).E- ...
46 | (Model.ETX*PacketSize + Model.Efs*PacketSize*(distance^2)); %% Energ
47 |
48 | % Sent a packet
49 | if(Sensors(Sender(i)).E>0) %reachable node
50 | sap=sap+1; %packet sent
51 | end
52 |
53 |
54 | end
55 |

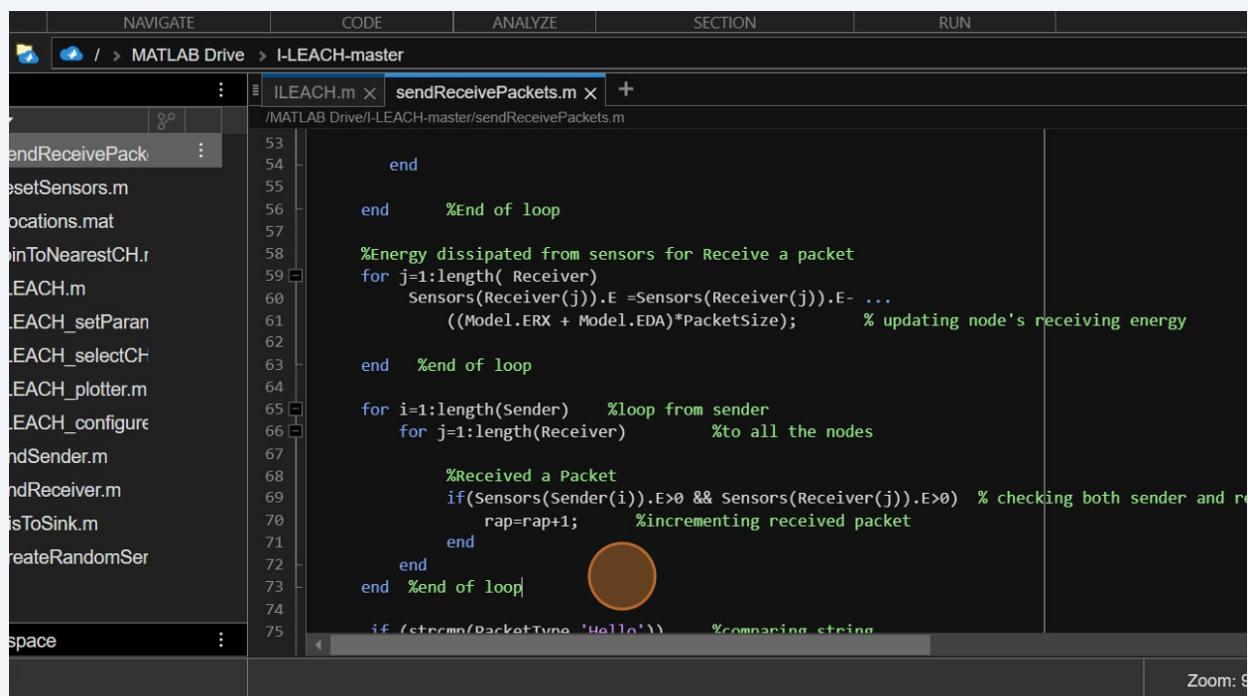
```

42

In this part of the code, it models the energy consumption when receiving a packet.

- Inside the first loop (for $j=1:length(Receiver)$), it iterates through all the receivers (Receiver) to update the energy levels of the receiver nodes. The energy dissipation is based on the energy consumed for receiving a bit (Model.ERX) and the data aggregation energy (Model.EDA). This energy dissipation is applied to each receiver in the loop.
- In the second loop (for $i=1:length(Sender)$ nested inside another loop for $j=1:length(Receiver)$), it iterates through each pair of sender (Sender(i)) and receiver (Receiver(j)) nodes. If both the sender and receiver have remaining energy (Sensors(Sender(i)).E and Sensors(Receiver(j)).E are greater than zero), it increments the rap counter, indicating that a packet has been successfully received.

These loops collectively model the energy consumption associated with receiving a packet by updating the energy levels of the receiver nodes and tracking the number of received packets (rap).



The screenshot shows the MATLAB Drive interface with the 'I-LEACH-master' folder selected. The 'sendReceivePackets.m' file is open in the editor. The code is as follows:

```

NAVIGATE | CODE | ANALYZE | SECTION | RUN
MATLAB Drive > I-LEACH-master
ILEACH.m X sendReceivePackets.m X +
/MATLAB Drive/I-LEACH-master/sendReceivePackets.m
53
54
55
56
57
58 %Energy dissipated from sensors for Receive a packet
59 for j=1:length( Receiver)
60 Sensors(Receiver(j)).E =Sensors(Receiver(j)).E- ...
61 ((Model.ERX + Model.EDA)*PacketSize); % updating node's receiving energy
62
63 end %end of loop
64
65 for i=1:length(Sender) %loop from sender
66 for j=1:length(Receiver) %to all the nodes
67
68 %Received a Packet
69 if(Sensors(Sender(i)).E>0 && Sensors(Receiver(j)).E>0) % checking both sender and receiver
70 rap=rap+1; %incrementing received packet
71
72 end
73 end %end of loop
74 if(strcmp(PacketType, 'Hello')) %comparing string

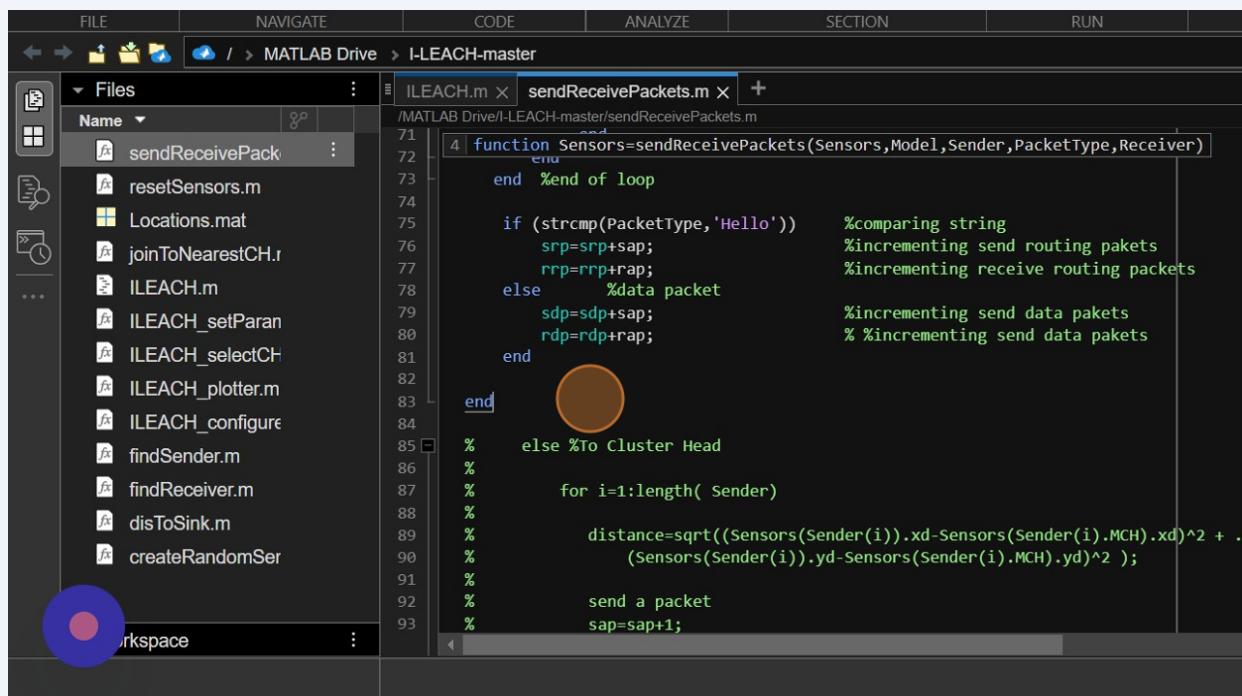
```

43

This part of the code distinguishes between 'Hello' packets and 'Data' packets. Depending on the type of packet (PacketType), it updates the counters for sent and received packets accordingly.

- If the packet type is 'Hello', it increments the counters for sent routing packets (srp) by adding the value of sap (send a packet) and increments the counters for received routing packets (rrp) by adding the value of rap (receive a packet).
- If the packet type is not 'Hello' (i.e., it's a 'Data' packet), it increments the counters for sent data packets (sdp) by adding the value of sap and increments the counters for received data packets (rdp) by adding the value of rap.

This way, it keeps track of the total number of sent and received packets for both routing and data packets in the simulation.



```

FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master
Files : ILEACH.m x sendReceivePackets.m x
/MATLAB Drive/I-LEACH-master/sendReceivePackets.m
71 | 4 function Sensors=sendReceivePackets(Sensors,Model,Sender,PacketType,Receiver)
72 |     %end of loop
73 |
74 |     if (strcmp(PacketType,'Hello'))      %comparing string
75 |         srp=srp+sap;                   %incrementing send routing pakets
76 |         rrp=rrp+rap;                   %incrementing receive routing packets
77 |     else                                %data packet
78 |         sdp=sdp+sap;                   %incrementing send data pakets
79 |         rdp=rdp+rap;                   % %incrementing send data pakets
80 |     end
81 |
82 | end
83 |
84 | % else %To Cluster Head
85 | %
86 | %
87 | % for i=1:length( Sender)
88 | %
89 | %     distance=sqrt((Sensors(Sender(i)).xd-Sensors(Sender(i).MCH).xd)^2 +
90 | %                         (Sensors(Sender(i)).yd-Sensors(Sender(i).MCH).yd)^2 );
91 | %
92 | %     send a packet
93 | %     sap=sap+1;

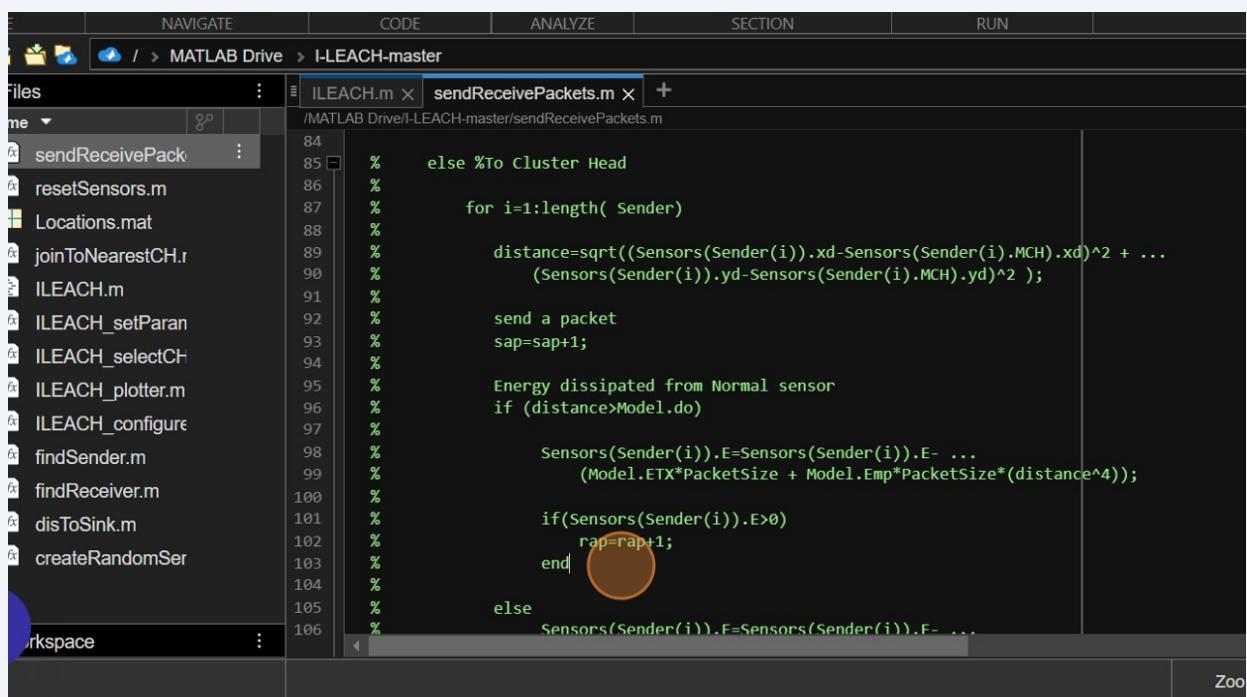
```

44

This commented-out code is an alternative implementation or an additional scenario for sending and receiving packets. The code here is to handle the case when a packet is sent to a cluster head (CH). However, it is currently commented out, indicating that it is not being executed in the current version of the code.

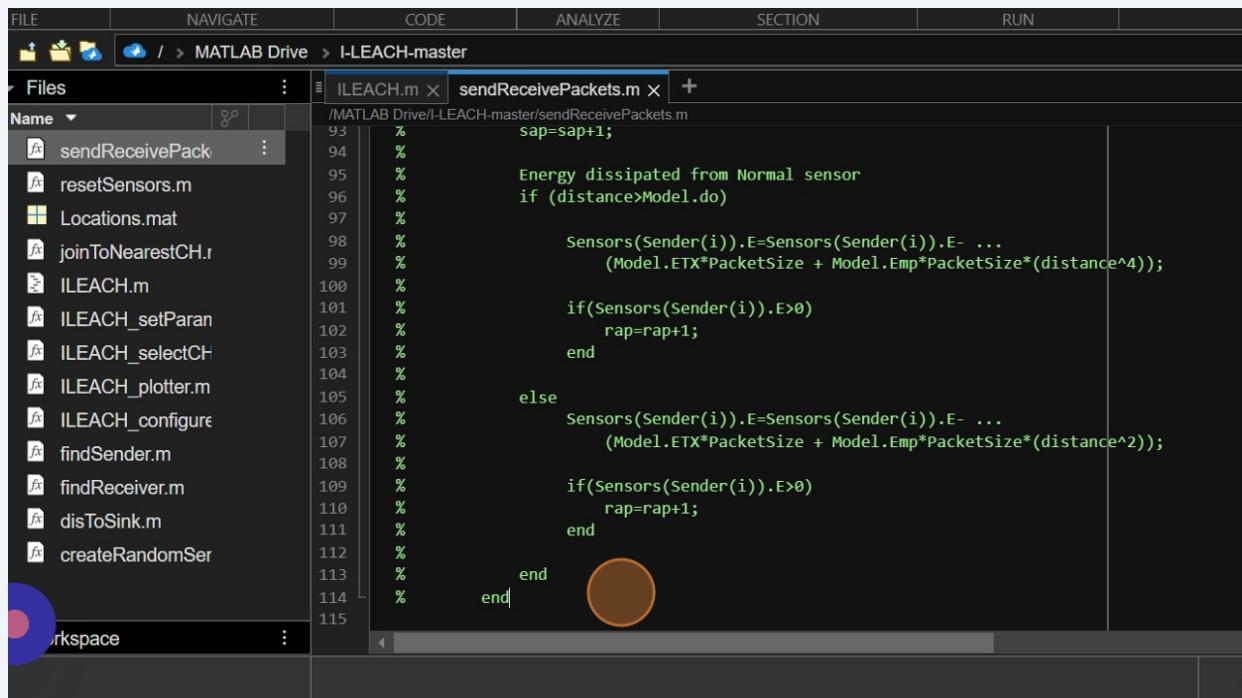
Here is a breakdown of the code inside the commented block:

1. **For each sender** (Sender(i)), calculate the distance to its associated cluster head (Sender(i).MCH).
2. **Send a packet** (sap=sap+1).
3. **Calculate energy dissipated from the normal sensor based on the distance to the cluster head.**
 - If the distance is greater than Model.do, it uses multipath communication energy dissipation.
 - If the distance is less than or equal to Model.do, it uses free space path communication energy dissipation.
4. **If the sender's energy is still positive after sending the packet, increment the received packet counter (rap=rap+1).**



```
NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master
Files : ILEACH.m x sendReceivePackets.m x +
/MATLAB Drive/I-LEACH-master/sendReceivePackets.m
84 % else %To Cluster Head
85 %
86 %
87 % for i=1:length( Sender)
88 %
89 % distance=sqrt((Sensors(Sender(i)).xd-Sensors(Sender(i).MCH).xd)^2 + ...
90 % (Sensors(Sender(i)).yd-Sensors(Sender(i).MCH).yd)^2 );
91 %
92 % send a packet
93 % sap=sap+1;
94 %
95 % Energy dissipated from Normal sensor
96 % if (distance>Model.do)
97 %
98 % Sensors(Sender(i)).E=Sensors(Sender(i)).E- ...
99 % (Model.ETX*PacketSize + Model.Emp*PacketSize*(distance^4));
100 %
101 % if(Sensors(Sender(i)).E>0)
102 % rap=rap+1;
103 % end
104 %
105 % else
106 % Sensors(Sender(i)).E=Sensors(Sender(i)).E- ...
```

45 Continued



The screenshot shows the MATLAB IDE interface with the following details:

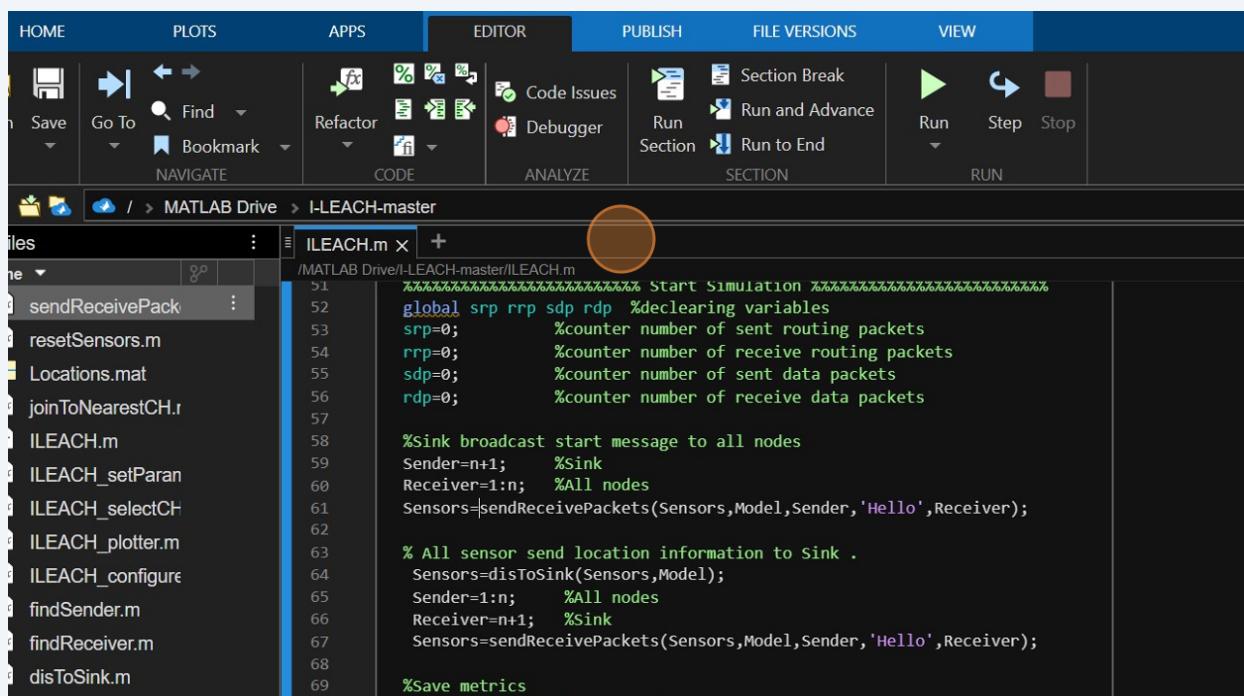
- File Path:** /MATLAB Drive / I-LEACH-master
- Open File:** sendReceivePackets.m
- Code Content:** The code is a script for the I-LEACH protocol. It includes logic for energy dissipation and packet transmission. A specific line of code is highlighted with a brown circle: "end" at line 114.
- Code Lines:** The code spans from line 93 to line 115. Key sections include:
 - Line 93: `sap=sap+1;`
 - Line 95: `% Energy dissipated from Normal sensor`
 - Line 96: `if (distance>Model.do)`
 - Line 98: `Sensors(Sender(i)).E=Sensors(Sender(i)).E- ...`
 - Line 99: `(Model.ETX*PacketSize + Model.Emp*PacketSize*(distance^4));`
 - Line 101: `if(Sensors(Sender(i)).E>0)`
 - Line 102: `rap=rap+1;`
 - Line 103: `end`
 - Line 105: `else`
 - Line 106: `Sensors(Sender(i)).E=Sensors(Sender(i)).E- ...`
 - Line 107: `(Model.ETX*PacketSize + Model.Emp*PacketSize*(distance^2));`
 - Line 109: `if(Sensors(Sender(i)).E>0)`
 - Line 110: `rap=rap+1;`
 - Line 111: `end`
 - Line 113: `end`
 - Line 114: `end` (highlighted with a brown circle)
 - Line 115: `%`

46

Now back to Main File ILEACH where simulation code starts
 This section is setting the initial values for the counters that will keep track of the number of sent and received packets during the simulation. The counters are initialized to zero:

- srp: Counter for the number of sent routing packets.
- rrp: Counter for the number of received routing packets.
- sdp: Counter for the number of sent data packets.
- rdp: Counter for the number of received data packets.

These counters will be updated during the simulation as packets are sent and received. They provide a way to monitor and analyze the network's behavior and performance.



```

 51 % Start Simulation
 52 global srp rrp sdp rdp %declearing variables
 53 srp=0; %counter number of sent routing packets
 54 rrp=0; %counter number of receive routing packets
 55 sdp=0; %counter number of sent data packets
 56 rdp=0; %counter number of receive data packets
 57
 58 %sink broadcast start message to all nodes
 59 Sender=n+1; %Sink
 60 Receiver=1:n; %All nodes
 61 Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver);
 62
 63 % All sensor send location information to Sink .
 64 Sensors=disToSink(Sensors,Model);
 65 Sender=1:n; %All nodes
 66 Receiver=n+1; %Sink
 67 Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver);
 68
 69 %Save metrics

```

47

This part of the code simulates the initial broadcast of "Hello" messages from the sink to all nodes in the network, and then each node sends its location information back to the sink. Additionally, the code updates the counters for the number of sent and received routing packets (SRP and RRP) and the number of sent and received data packets (SDP and RDP) for the first round.

Here's a breakdown of the code:

1. Sink broadcast "Hello" messages to all nodes:

- Sender: The sink node.
- Receiver: An array containing all nodes in the network.
- Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver);: This function simulates the transmission of "Hello" packets from the sink to all nodes. It updates the energy levels of the sink and nodes accordingly.

2. All nodes send their location information to the Sink:

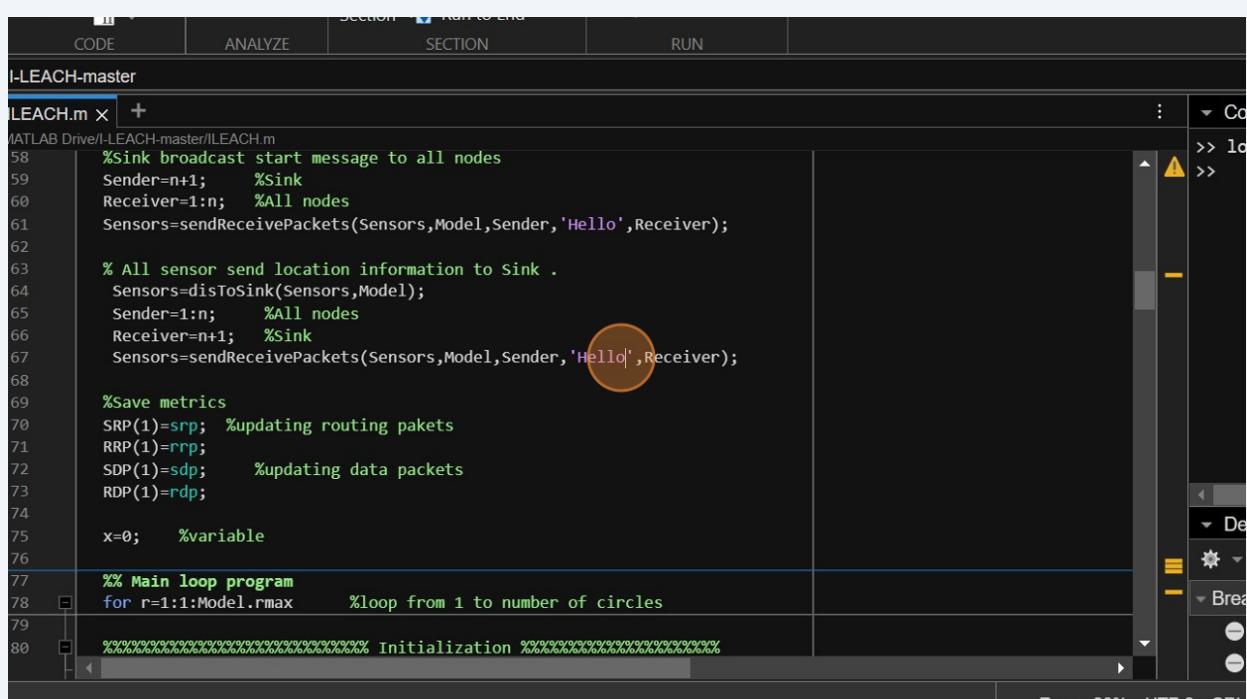
- Sender: An array containing all nodes in the network.
- Receiver: The sink node.
- Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver);: This function simulates each node sending its location information to the sink. It updates the energy levels of the nodes and the sink.

3. Save metrics for the first round:

- SRP(1), RRP(1), SDP(1), RDP(1): Update the counters with the number of sent routing packets, received routing packets, sent data packets, and received data packets for the first round.

4. Reset the variable x to 0:

- x=0;: This variable is declared but not used in this part of the code



```

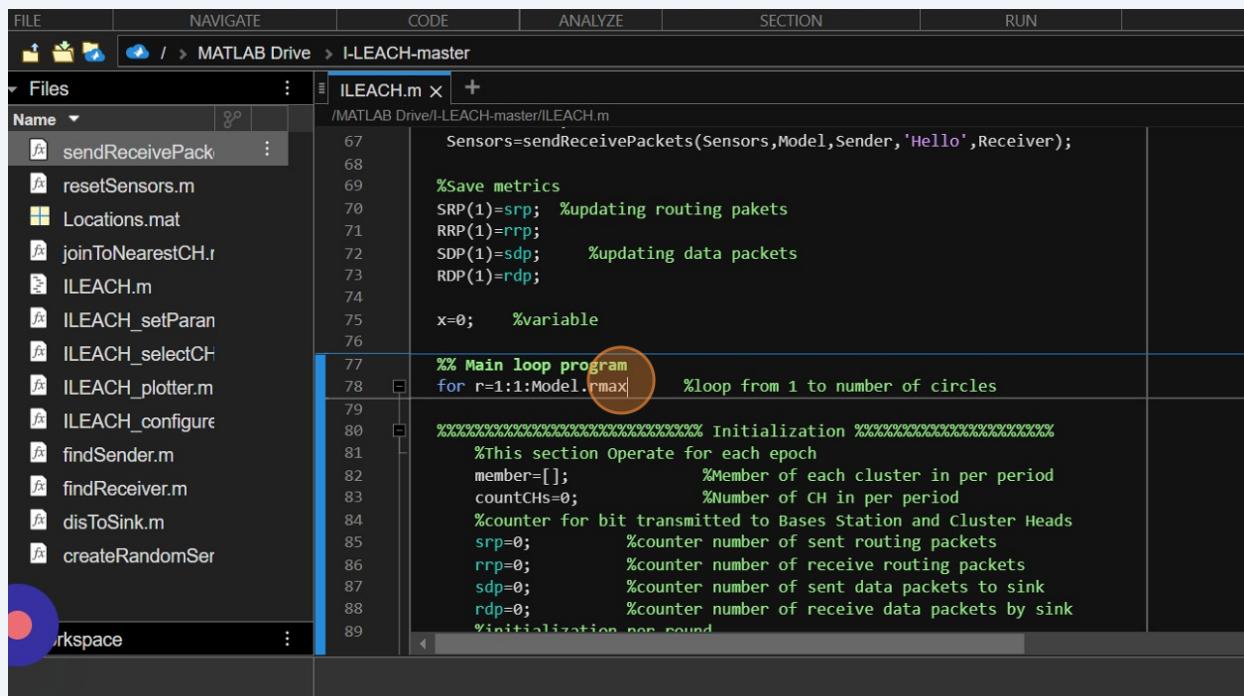
CODE ANALYZE SECTION RUN
I-LEACH-master
I-LEACH.m x +
MATLAB Drive/I-LEACH-master/I-LEACH.m
58 %sink broadcast start message to all nodes
59 Sender=n+1; %Sink
60 Receiver=1:n; %All nodes
61 Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver);
62
63 % All sensor send location information to Sink .
64 Sensors=disToSink(Sensors,Model);
65 Sender=1:n; %All nodes
66 Receiver=n+1; %Sink
67 Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver);
68
69 %Save metrics
70 SRP(1)=srp; %updating routing pakets
71 RRP(1)=rrp;
72 SDP(1)=sdp; %updating data packets
73 RDP(1)=rdp;
74
75 x=0; %variable
76
77 %% Main loop program
78 for r=1:1:Model.rmax %loop from 1 to number of circles
79 %%%%%%%%%%%%%% Initialization %%%%%%%%%%%%%%
80

```

48

- for r=1:1:Model.rmax: This line initiates a for loop that runs from r=1 to r=Model.rmax with a step size of 1. The loop variable r represents the current round of the simulation.

The code inside this loop will be responsible for executing the simulation logic for each round.



FILE NAVIGATE CODE ANALYZE SECTION RUN

/ > MATLAB Drive > I-LEACH-master

ILEACH.m X +

Name : ILEACH.m

sendReceivePack : /MATLAB Drive/I-LEACH-master/ILEACH.m

resetSensors.m
Locations.mat
joinToNearestCH.r
ILEACH.m
ILEACH_setParan
ILEACH_selectCH
ILEACH_plotter.m
ILEACH_configure
findSender.m
findReceiver.m
disToSink.m
createRandomSer

Workspace

```
67 Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver);
68
69 %Save metrics
70 SRP(1)=srp; %updating routing pakets
71 RRP(1)=rrp;
72 SDP(1)=sdp; %updating data packets
73 RDP(1)=rdp;
74
75 x=0; %variable
76
77 %% Main loop program
78 for r=1:1:Model.rmax %loop from 1 to number of circles
79
80 %%%%%%%%%%%%%% Initialization %%%%%%%%%%%%%%
81 %This section Operate for each epoch
82 member=[]; %Member of each cluster in per period
83 countCHs=0; %Number of CH in per period
84 %counter for bit transmitted to Bases Station and Cluster Heads
85 srp=0; %counter number of sent routing packets
86 rrp=0; %counter number of receive routing packets
87 sdp=0; %counter number of sent data packets to sink
88 rdp=0; %counter number of receive data packets by sink
89 %initialization per round
```

49

This section of the code appears to be related to the initialization and setup for each epoch (simulation round).

```
% This section Operates for each epoch (round)
member=[]; % Member of each cluster in each period
countCHs=0; % Number of Cluster Heads in each period

% Counter for bits transmitted to Base Station and Cluster Heads
srp=0; % Counter for the number of sent routing packets
rrp=0; % Counter for the number of received routing packets
sdp=0; % Counter for the number of sent data packets to the sink
rdp=0; % Counter for the number of received data packets by the sink

% Initialization per round
SRP(r+1)=srp; % Update the counters for sent routing packets
RRP(r+1)=rrp; % Update the counters for received routing packets
SDP(r+1)=sdp; % Update the counters for sent data packets
RDP(r+1)=rdp; % Update the counters for received data packets

pause(0.001) % Pause the simulation for a short duration
hold off; % Clear the current figure to prepare for the next visualization
packets_TO_BS=0; % Variable for counting packets sent to the Base Station
```

The screenshot shows the MATLAB IDE interface with the following details:

- Toolbar:** FILE, NAVIGATE, CODE, ANALYZE, SECTION, RUN.
- File Explorer:** Shows files in the 'I-LEACH-master' folder, including sendReceivePack.m, resetSensors.m, Locations.mat, joinToNearestCH.r, ILEACH.m, ILEACH_setParan.m, ILEACH_selectCH.m, ILEACH_plotter.m, ILEACH_configure.m, findSender.m, findReceiver.m, disToSink.m, and createRandomSer.m.
- Editor:** The 'ILEACH.m' file is open. The code is annotated with comments explaining its purpose. A red oval highlights the line 'SRP(r+1)=srp;'. The code is as follows:

```
%% Main loop program
for r=1:1:Model.rmax %loop from 1 to number of circles
    %% Initialization
    %This section Operate for each epoch
    member=[]; %Member of each cluster in per period
    countCHs=0; %Number of CH in per period
    %counter for bit transmitted to Bases Station and Cluster Heads
    srp=0; %counter number of sent routing packets
    rrp=0; %counter number of receive routing packets
    sdp=0; %counter number of sent data packets to sink
    rdp=0; %counter number of receive data packets by sink
    %initialization per round
    SRP(r+1)=srp;
    RRP(r+1)=rrp;
    SDP(r+1)=sdp;
    RDP(r+1)=rdp;
    pause(0.001) %pause simulation
    hold off; %clear current figure to prepare for the next visualization
    packets_TO_BS=0; % the number of packets sent to the Base Station
    Sensors=resetSensors(Sensors,Model);
```

50

- Sensors=resetSensors(Sensors,Model);: This function resets the sensors to their initial state. It's likely that the initial state includes information about whether a node was a cluster head in the previous round (G attribute).

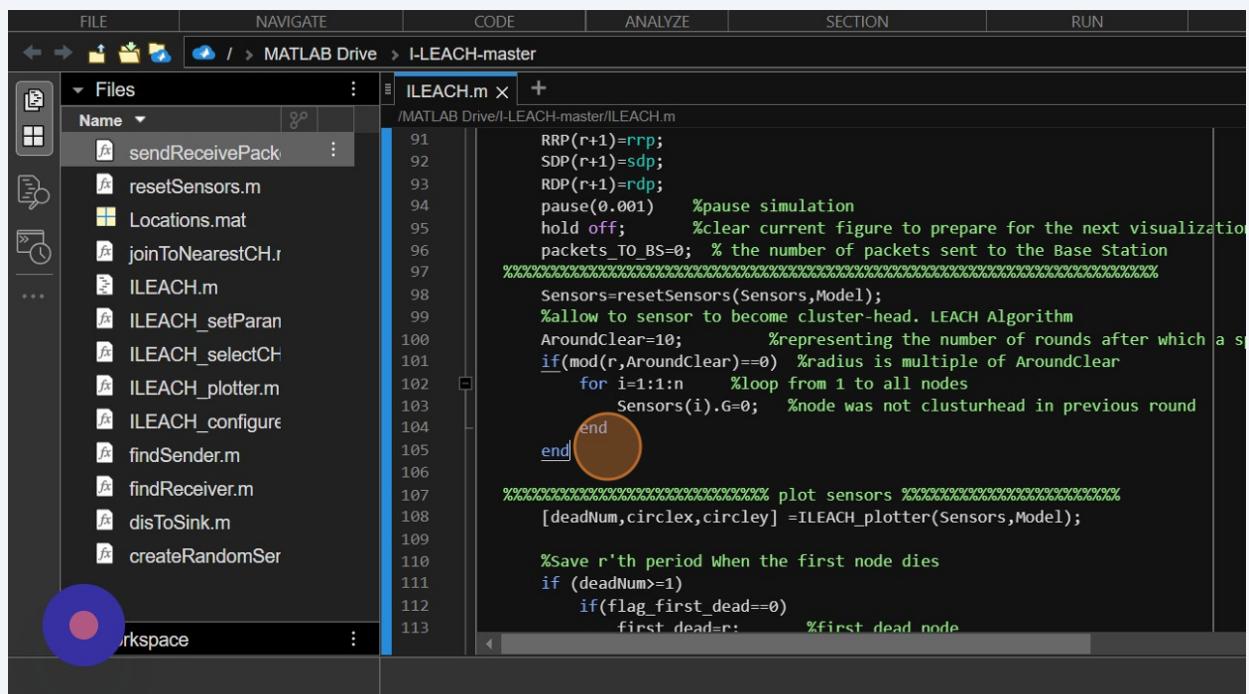
- AroundClear=10;: This variable represents the number of rounds after which a specific action will be taken. This action, as we can see in the next block, is resetting the 'G' attribute for all nodes.

- if(mod(r,AroundClear)==0): Checks if the current round r is a multiple of AroundClear. If it is, the following block of code is executed.

- for i=1:1:n: Loops through all nodes.

- Sensors(i).G=0;: Resets the 'G' attribute for each node, indicating that the node was not a cluster head in the previous round.

This process seems to be part of the LEACH protocol where cluster heads are selected periodically based on the defined parameter AroundClear.

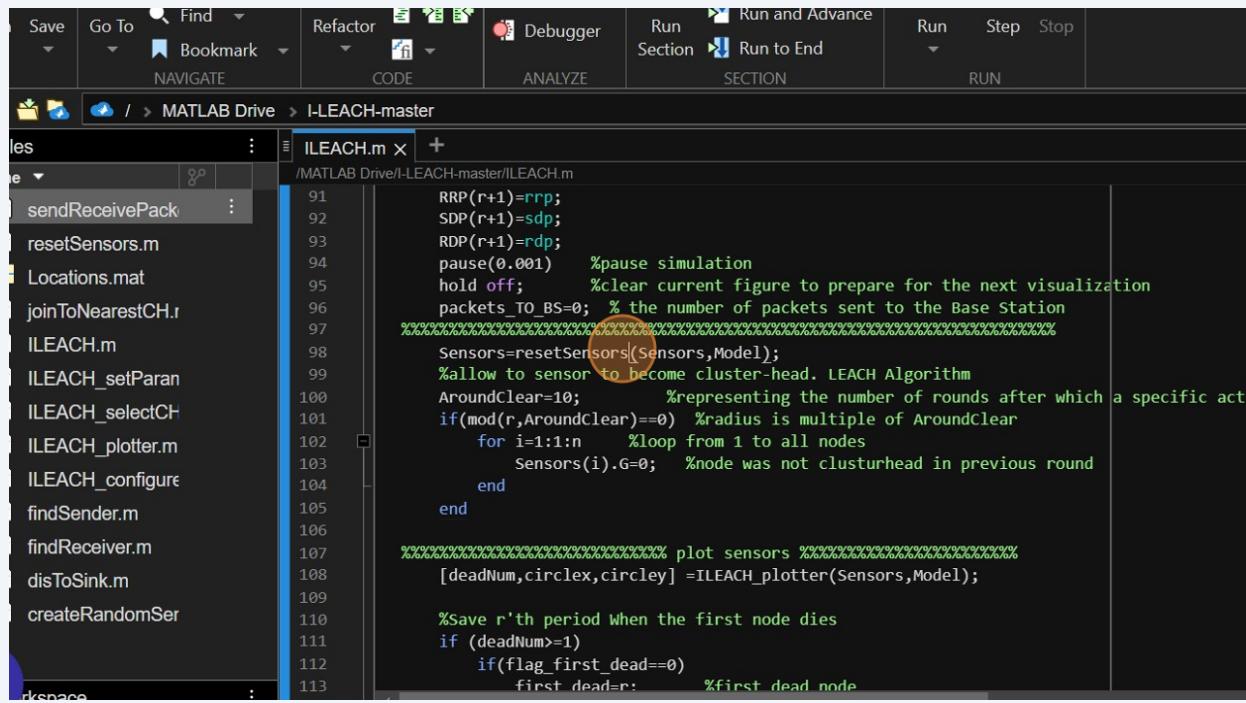


```

FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master
Files : ILEACH.m +
/MATLAB Drive/I-LEACH-master/ILEACH.m
91 RRP(r+1)=rrp;
92 SDP(r+1)=sdp;
93 RDP(r+1)=rdp;
94 pause(0.001) %pause simulation
95 hold off; %clear current figure to prepare for the next visualization
96 packets_TO_BS=0; % the number of packets sent to the Base Station
97 %%%%%%%%%%%%%%
98 Sensors=resetSensors(Sensors,Model);
99 %allow to sensor to become cluster-head. LEACH Algorithm
100 AroundClear=10; %representing the number of rounds after which a s
101 if(mod(r,AroundClear)==0) %radius is multiple of AroundClear
102     for i=1:1:n %loop from 1 to all nodes
103         Sensors(i).G=0; %node was not clusturhead in previous round
104     end
105 end %%%%%%%%%%%%%%
106 % plot sensors %%%%%%%%%%%%%%
107 [deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model);
108
109 %Save r'th period When the first node dies
110 if (deadNum>=1)
111     if(flag_first_dead==0)
112         first_dead=r; %first dead node
113

```

51 Now lets observe the resetSensors function



The screenshot shows the MATLAB IDE interface with the following details:

- Toolbar:** Save, Go To, Find, Refactor, Debugger, Run, Run and Advance, Run to End, Run, Step, Stop, RUN.
- File Explorer:** Shows files in the MATLAB Drive / I-LEACH-master folder, including ILEACH.m, sendReceivePack.m, resetSensors.m, Locations.mat, joinToNearestCH.r, ILEACH.m, ILEACH_setParam.m, ILEACH_selectCH.m, ILEACH_plotter.m, ILEACH_configure.m, findSender.m, findReceiver.m, disToSink.m, createRandomSer...
- I-LEACH.m Editor:** The code is as follows:

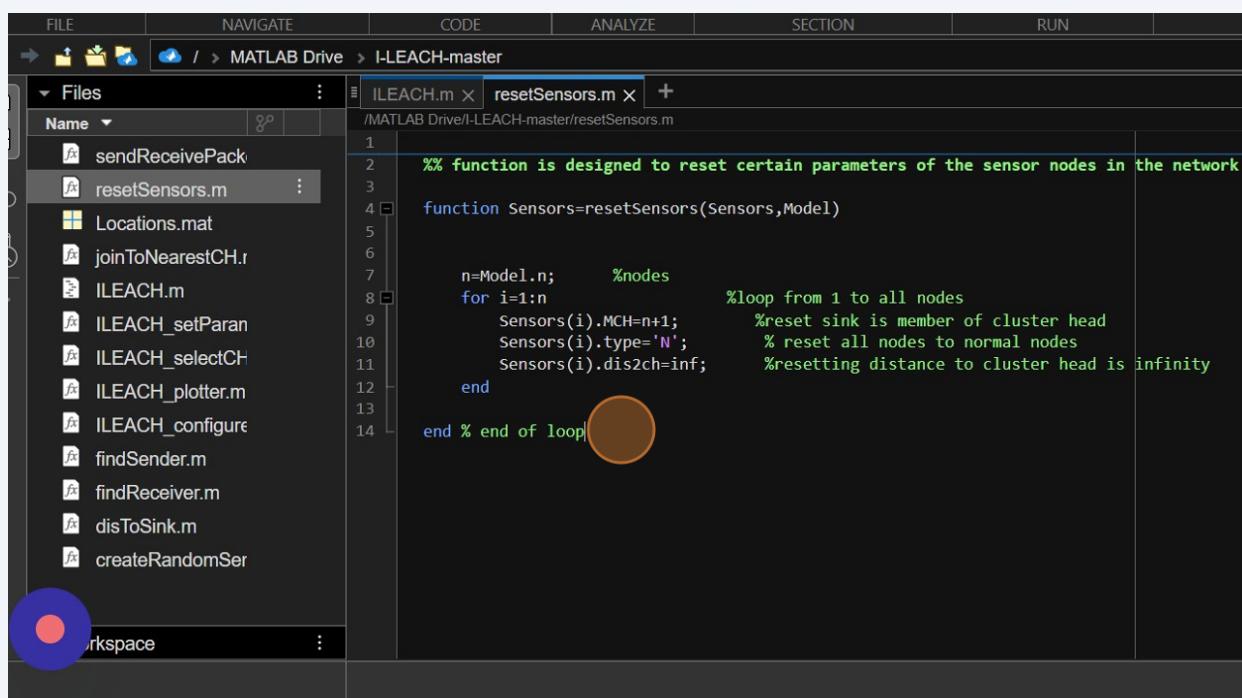
```
RRP(r+1)=rrp;
SDP(r+1)=sdp;
RDP(r+1)=rdp;
pause(0.001) %pause simulation
hold off; %clear current figure to prepare for the next visualization
packets_TO_BS=0; % the number of packets sent to the Base Station
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Sensors=resetSensors([Sensors,Model]);
%allow to sensor to become cluster-head. LEACH Algorithm
AroundClear=10; %representing the number of rounds after which a specific act
if(mod(r,AroundClear)==0) %radius is multiple of AroundClear
    for i=1:n %loop from 1 to all nodes
        Sensors(i).G=0; %node was not clusterhead in previous round
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% plot sensors %%%%%%
[deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model);

%Save r'th period When the first node dies
if (deadNum>=1)
    if(flag_first_dead==0)
        first_dead=r; %first dead node
```

52 Here's what each part does:

- function Sensors=resetSensors(Sensors,Model): This function takes as input the Sensors array and the Model structure and returns the modified Sensors array.
- n=Model.n;: This line assigns the number of nodes (n) from the Model structure.
- for i=1:n: The loop iterates from 1 to the total number of nodes (n).
- Sensors(i).MCH=n+1;: It resets the MCH (Member of CH) attribute of each node to be equal to n+1, indicating that the node is a member of the sink or base station. This is typical in LEACH-like protocols where the sink plays a special role.
- Sensors(i).type='N';: It resets the type attribute of each node to be 'N', indicating that the node is a normal node, not a cluster head.
- Sensors(i).dis2ch=inf;: It resets the dis2ch attribute (distance to the cluster head) to infinity. This indicates that the distance to the cluster head is not known or undefined.

The purpose of this function is to prepare the sensor nodes for the next round of the simulation by resetting certain parameters



FILE NAVIGATE CODE ANALYZE SECTION RUN

/ MATLAB Drive / I-LEACH-master

Files

Name

- sendReceivePack
- resetSensors.m
- Locations.mat
- joinToNearestCH.r
- I-LEACH.m
- I-LEACH_setParan
- I-LEACH_selectCH
- I-LEACH_plotter.m
- I-LEACH_configure
- findSender.m
- findReceiver.m
- disToSink.m
- createRandomSer

Workspace

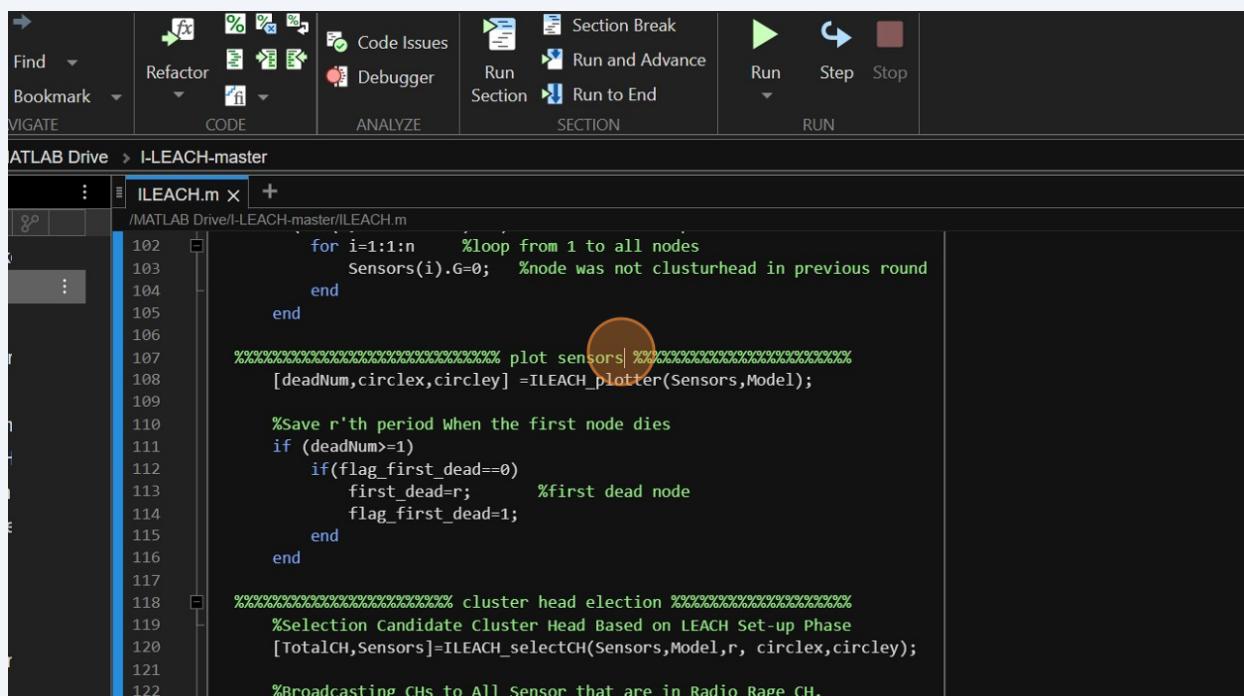
```
%>>> %% function is designed to reset certain parameters of the sensor nodes in the network
function Sensors=resetSensors(Sensors,Model)
n=Model.n; %nodes
for i=1:n %loop from 1 to all nodes
    Sensors(i).MCH=n+1; %reset sink is member of cluster head
    Sensors(i).type='N'; % reset all nodes to normal nodes
    Sensors(i).dis2ch=inf; %resetting distance to cluster head is infinity
end % end of loop
```

53

- ILEACH_plotter: This function is responsible for visualizing the state of the sensor network in the LEACH simulation. It takes the Sensors array and the Model structure as input and returns three outputs: deadNum (the number of dead nodes), circlex, and circley (coordinates of circular regions for communication range visualization).

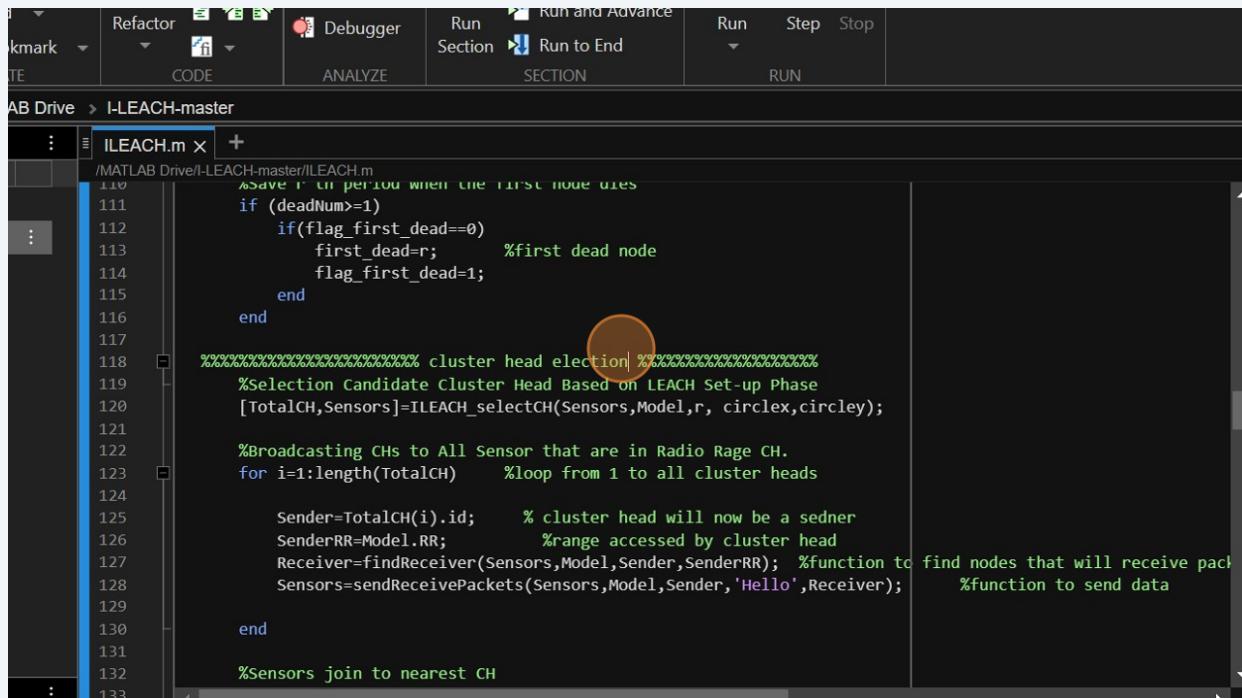
• If there is at least one dead node ($\text{deadNum} \geq 1$), it checks whether the flag_first_dead is 0. If it is, it means this is the first time a node has died in the simulation. It records the current round r as the first_dead round and sets flag_first_dead to 1 to indicate that the first dead node is recorded.

This part of the code seems to be monitoring the rounds during which nodes die in the simulation. The first_dead variable is used to keep track of the round when the first node dies



```
ATLAB Drive > I-LEACH-master
ILEACH.m X +
/MATLAB Drive/I-LEACH-master/ILEACH.m
102     for i=1:1:n    %loop from 1 to all nodes
103         Sensors(i).G=0;    %node was not clusterhead in previous round
104     end
105
106
107     %%%%%%%%%%%%%% plot sensors %%%%%%%%%%%%%%
108     [deadNum,circlex,circley] =ILEACH_plotter(Sensors,Model);
109
110     %Save r'th period When the first node dies
111     if (deadNum>=1)
112         if(flag_first_dead==0)
113             first_dead=r;      %first dead node
114             flag_first_dead=1;
115         end
116     end
117
118     %%%%%%%%%%%%%% cluster head election %%%%%%%%%%%%%%
119     %Selection Candidate Cluster Head Based on LEACH Set-up Phase
120     [TotalCH,Sensors]=ILEACH_selectCH(Sensors,Model,r, circlex,circley);
121
122     %Broadcasting CHs to All Sensor that are in Radio Range CH.
```

54 Now comes the Cluster Head Election for each cluster



The screenshot shows the MATLAB IDE interface with the 'ILEACH.m' script open. The code implements a cluster head election process. It starts by saving a variable 'r' if the first node dies. Then, it checks if 'deadNum' is greater than or equal to 1. If so, it sets 'flag_first_dead' to 1 and updates 'first_dead' to 'r'. After this, it enters a loop for all cluster heads. Inside the loop, it selects a cluster head based on the LEACH setup phase. It then broadcasts CHs to all sensors within the radio range of the cluster head. Finally, it sends data to the nearest cluster head.

```
%Save r in period when the first node dies
if (deadNum>=1)
    if(flag_first_dead==0)
        first_dead=r; %first dead node
        flag_first_dead=1;
    end
end

%%%%%%%%%%%%% cluster head election%%%%%%%%%%%%%
%Selection Candidate Cluster Head Based on LEACH Set-up Phase
[TotalCH,Sensors]=ILEACH_selectCH(Sensors,Model,r, circlex,circley);

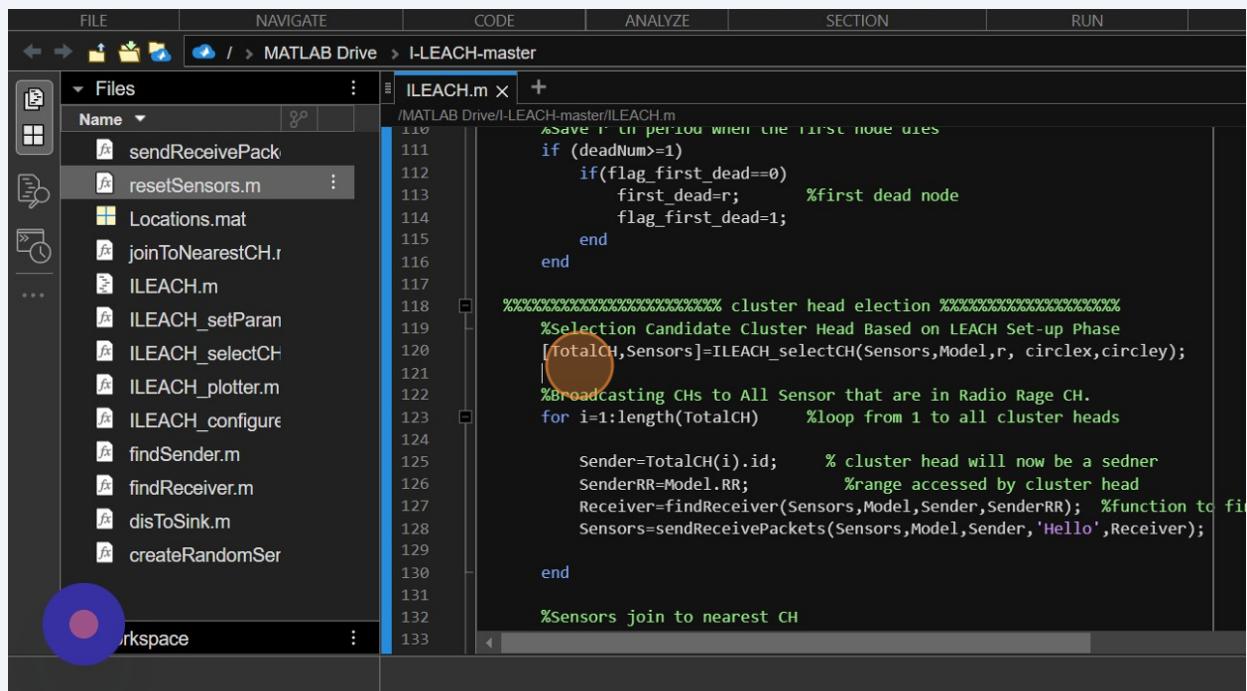
%Broadcasting CHs to All Sensor that are in Radio Range CH.
for i=1:length(TotalCH) %loop from 1 to all cluster heads

    Sender=TotalCH(i).id; % cluster head will now be a sender
    SenderRR=Model.RR; %range accessed by cluster head
    Receiver=findReceiver(Sensors,Model,Sender,SenderRR); %function to find nodes that will receive pack
    Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver); %function to send data

end

%Sensors join to nearest CH
```

55 A function named ILEACH_selectCH is called and passed some parameters



The screenshot shows the MATLAB IDE interface with the 'ILEACH.m' script open. The 'ILEACH_selectCH' function call is highlighted with a red circle. This function is responsible for selecting candidate cluster heads based on the LEACH setup phase. It takes several parameters: 'Sensors', 'Model', 'r', 'circlex', and 'circley'.

```
%Save r in period when the first node dies
if (deadNum>=1)
    if(flag_first_dead==0)
        first_dead=r; %first dead node
        flag_first_dead=1;
    end
end

%%%%%%%%%%%%% cluster head election%%%%%%%%%%%%%
%Selection Candidate Cluster Head Based on LEACH Set-up Phase
[TotalCH,Sensors]=ILEACH_selectCH(Sensors,Model,r, circlex,circley);

%Broadcasting CHs to All Sensor that are in Radio Range CH.
for i=1:length(TotalCH) %loop from 1 to all cluster heads

    Sender=TotalCH(i).id; % cluster head will now be a sender
    SenderRR=Model.RR; %range accessed by cluster head
    Receiver=findReceiver(Sensors,Model,Sender,SenderRR); %function to find nodes that will receive pack
    Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver);

end

%Sensors join to nearest CH
```

56 Now lets observe this function

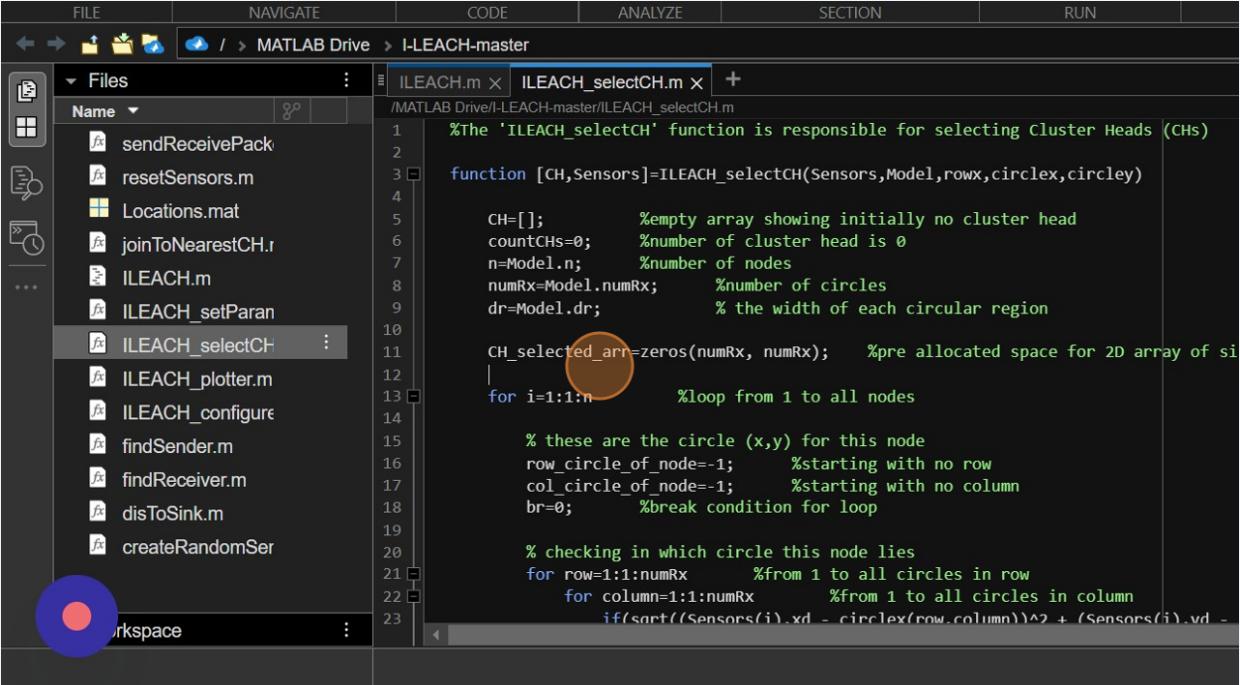
```
NAVIGATE CODE ANALYZE SECTION RUN
> MATLAB Drive > I-LEACH-master
ILEACH.m x +
/MATLAB Drive/I-LEACH-master/ILEACH.m
110 %save r in period when the first node dies
111 if (deadNum>=1)
112     if(flag_first_dead==0)
113         first_dead=r;      %first dead node
114         flag_first_dead=1;
115     end
116 end
117
118 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% cluster head election %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
119 %Selection Candidate Cluster Head Based on LEACH Set-up Phase
120 [TotalCH,Sensors]=ILEACH_selectCH(Sensors,Model,r, circlex,circley);
121
122 %Broadcasting CHs to All Sensor that are in Radio Range CH.
123 for i=1:length(TotalCH)    %loop from 1 to all cluster heads
124
125     Sender=TotalCH(i).id;    % cluster head will now be a sender
126     SenderRR=Model.RR;        %range accessed by cluster head
127     Receiver=findReceiver(Sensors,Model,Sender,SenderRR); %function to find nodes that will receive
128     Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver);   %function to send data
129
130 end
131
132 %Sensors join to nearest CH
133
```

57

This function ILEACH_selectCH seems to be responsible for selecting Cluster Heads (CHs) in the LEACH simulation. Let's break down the code:

CH: It is an empty array that will be used to store the IDs of selected Cluster Heads.

- Sensors: This is the array of sensor nodes.
- Model: The structure containing simulation parameters.
- rowx: It seems to be the current row index in the circular grid pattern.
- circlex and circley: These are arrays storing the coordinates of circular regions for communication range visualization.
- CH: Initialized as an empty array to store the selected Cluster Heads.
- countCHs: Initialized to 0, representing the count of selected Cluster Heads.
- n: Number of sensor nodes.
- numRx: The number of circular regions.
- dr: The width of each circular region.
- CH_selected_arr: A 2D array of zeros with a size equal to the number of circles. It seems to be used for tracking the selected CHs in each circular region.



```

FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive / > MATLAB Drive > I-LEACH-master
Files ILEACH.m x ILEACH_selectCH.m x +
Name
sendReceivePack
resetSensors.m
Locations.mat
joinToNearestCH.r
ILEACH.m
ILEACH_setParam
ILEACH_selectCH ...
ILEACH_plotter.m
ILEACH_configure
findSender.m
findReceiver.m
disToSink.m
createRandomSer
Workspace :
/MATLAB Drive/I-LEACH-master/ILEACH_selectCH.m
%The 'ILEACH_selectCH' function is responsible for selecting Cluster Heads (CHs)
function [CH,Sensors]=ILEACH_selectCH(Sensors,Model,rowx,circlex,circley)
    CH=[]; %empty array showing initially no cluster head
    countCHs=0; %number of cluster head is 0
    n=Model.n; %number of nodes
    numRx=Model.numRx; %number of circles
    dr=Model.dr; % the width of each circular region
    CH_selected_arr=zeros(numRx, numRx); %pre allocated space for 2D array of si
    for i=1:n %loop from 1 to all nodes
        % these are the circle (x,y) for this node
        row_circle_of_node=-1; %starting with no row
        col_circle_of_node=-1; %starting with no column
        br=0; %break condition for loop
        % checking in which circle this node lies
        for row=1:1:numRx %from 1 to all circles in row
            for column=1:1:numRx %from 1 to all circles in column
                if(sqrt((Sensors(i).xd - circlex(row,column))^2 + (Sensors(i).yd - circley(row,column))^2) < dr)
                    br=1;
                    break;
                end
            if(br==1)
                break;
            end
        end
    end
end

```

58

Here's what's happening:

- The outer loop (for $i=1:1:n$) iterates over all sensor nodes in the network.
- $\text{row_circle_of_node}$ and $\text{col_circle_of_node}$ are initialized as -1, indicating that the node has not been assigned to any circular region.
- br is used as a flag to break out of the loop once the circular region is found for the current node.

Now, let's look at the inner loop:

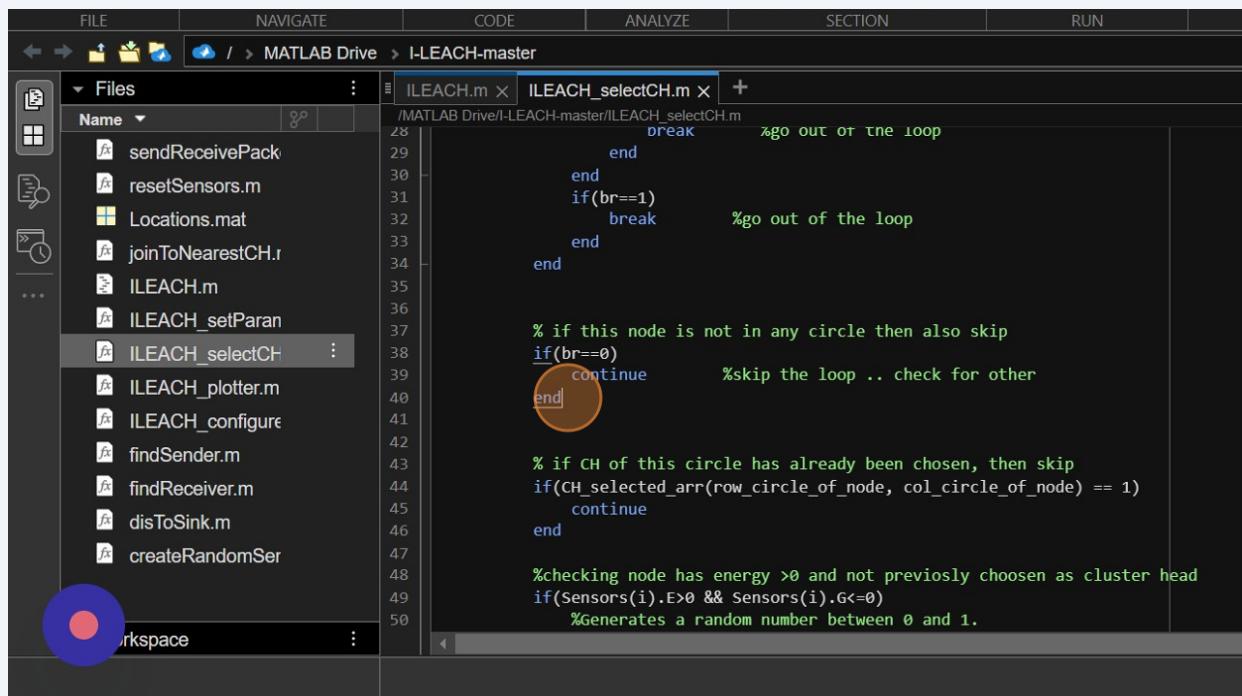
- The inner loop (for $\text{row}=1:1:\text{numRx}$) iterates over the rows of the circular regions.
- The nested loop (for $\text{column}=1:1:\text{numRx}$) iterates over the columns of the circular regions.
- The condition checks whether the Euclidean distance between the sensor node's coordinates and the center of the circular region is less than or equal to half of the circular region's width ($\text{dr}/2$).
 - If the condition is true, it means the sensor node lies within the circular region, and $\text{row_circle_of_node}$ and $\text{col_circle_of_node}$ are set to the corresponding row and column indices.
- The flags br and break are used to break out of both loops once the circular region is found.

This logic is useful for determining the position of a sensor node within the grid of circular regions.

```
FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master
Files / MATLAB Drive / I-LEACH-master / ILEACH_selectCH.m
Name
sendReceivePack
resetSensors.m
Locations.mat
joinToNearestCH.r
ILEACH.m
ILEACH_setParam
ILEACH_selectCH : ILEACH.m x ILEACH_selectCH.m x
13 for i=1:1:n %loop from 1 to all nodes
14
15 % these are the circle (x,y) for this node
16 row_circle_of_node=-1; %starting with no row
17 col_circle_of_node=-1; %starting with no column
18 br=0; %break condition for loop
19
20 % checking in which circle this node lies
21 for row=1:1:numRx %from 1 to all circles in row
22 for column=1:1:numRx %from 1 to all circles in column
23 if(sqrt((Sensors(i).xd - circlex(row,column))^2 + (Sensors(i).yd - circley(row,column))^2) <= dr/2)
24 row_circle_of_node=row; %node in circle that is horizontally
25 col_circle_of_node=column; %node in circle that is vertically
26
27 br=1; %break and check for next node
28 break %go out of the loop
29 end
30 end
31 if(br==1)
32 break %go out of the loop
33 end
34 end
```

59

If br is equal to 0 (indicating that the sensor node was not found within any circular region), the `continue` statement is executed, causing the code to skip the remaining part of the loop for the current sensor node and move on to the next iteration of the loop, where the next sensor node is considered



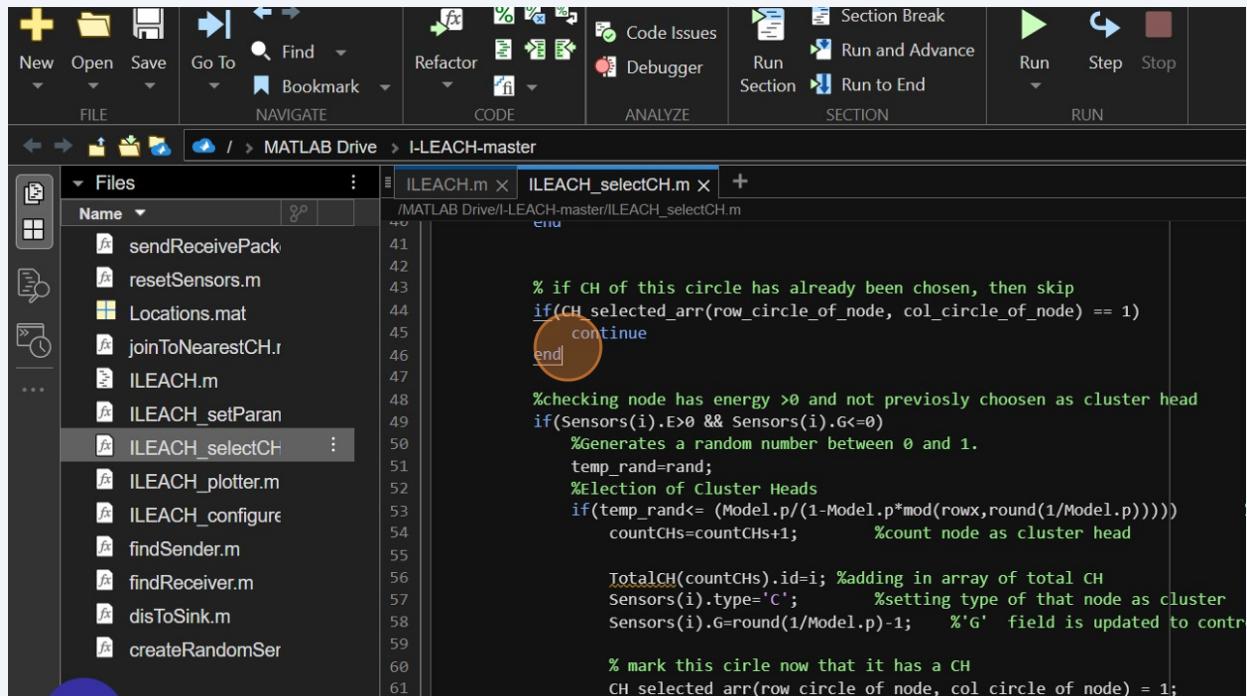
The screenshot shows the MATLAB IDE interface with the following details:

- File Explorer:** Shows files in the "ILEACH-master" folder, including `ILEACH.m`, `ILEACH_selectCH.m`, `sendReceivePack.m`, `resetSensors.m`, `Locations.mat`, `joinToNearestCH.r`, `ILEACH.m`, `ILEACH_setParam.m`, `ILEACH_selectCH.m`, `ILEACH_plotter.m`, `ILEACH_configure.m`, `findSender.m`, `findReceiver.m`, `disToSink.m`, and `createRandomSer.m`.
- Editor:** Displays the `ILEACH_selectCH.m` file content. The code implements a loop to select a Cluster Head (CH). It includes logic to skip nodes if they are not within a circle or if they have already been chosen. The highlighted line 38 is the closing brace of a nested loop, indicating the end of a iteration where br was 0.

```
28         break %go out of the loop
29     end
30     if(br==1)
31         break %go out of the loop
32     end
33 end
34
35 % if this node is not in any circle then also skip
36 if(br==0)
37     continue %skip the loop .. check for other
38 end
39
40 % if CH of this circle has already been chosen, then skip
41 if(CH_selected_arr(row_circle_of_node, col_circle_of_node) == 1)
42     continue
43 end
44
45 %checking node has energy >0 and not previously chosen as cluster head
46 if(Sensors(i).E>0 && Sensors(i).G<=0)
47     %Generates a random number between 0 and 1.
```

60

This part of the code checks whether a Cluster Head (CH) has already been chosen for the circular region where the current sensor node is located. If a CH has already been selected for that specific circle (as indicated by `CH_selected_arr(row_circle_of_node, col_circle_of_node) == 1`), the continue statement is executed. This means that the code skips the remaining part of the loop for the current sensor node and moves on to the next iteration of the loop.



```

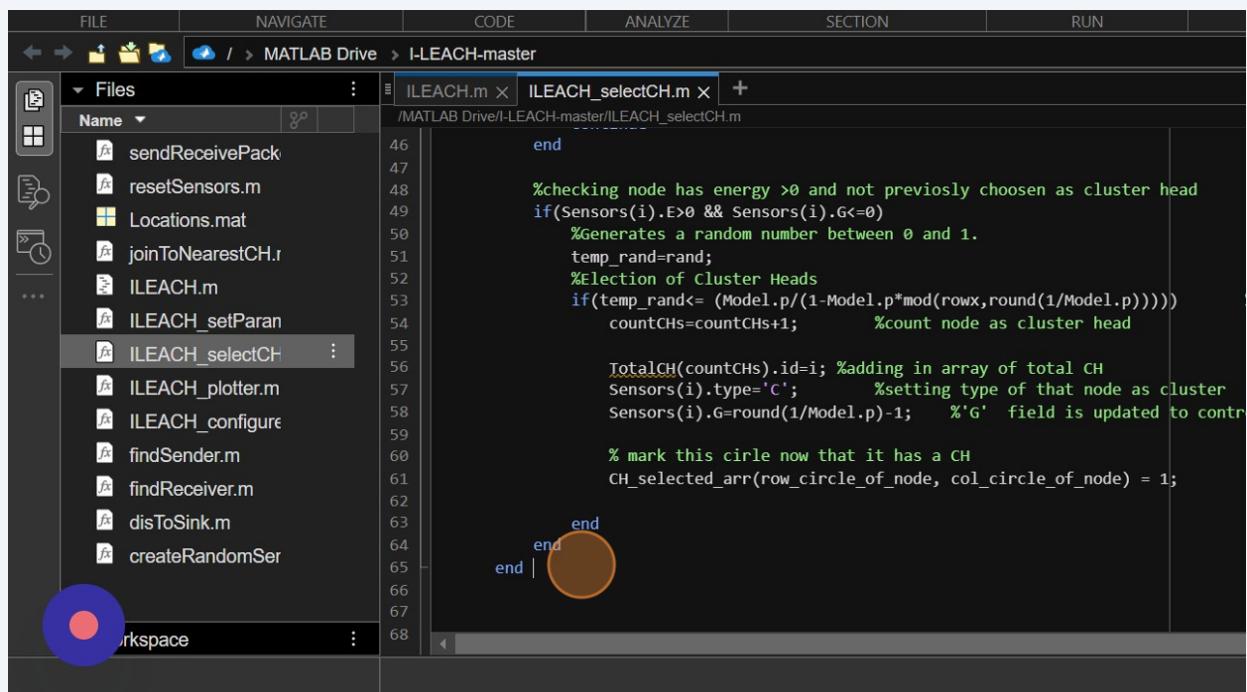
New Open Save Go To Find Refactor Code Issues Debugger Run Section Break Run and Advance Run to End Run Step Stop
FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master > ILEACH.m > ILEACH_selectCH.m
Files
Name
sendReceivePack
resetSensors.m
Locations.mat
joinToNearestCH.r
ILEACH.m
ILEACH_setParam
ILEACH_selectCH
ILEACH_plotter.m
ILEACH_configure
findSender.m
findReceiver.m
distoSink.m
createRandomSer
ILEACH.m
ILEACH_selectCH.m
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
% if CH of this circle has already been chosen, then skip
if(CH_selected_arr(row_circle_of_node, col_circle_of_node) == 1)
    continue
end
%checking node has energy >0 and not previosly choosen as cluster head
if(Sensors(i).E>0 && Sensors(i).G<=0)
    %Generates a random number between 0 and 1.
    temp_rand=rand;
    %Election of Cluster Heads
    if(temp_rand<= (Model.p/(1-Model.p*mod(rowx,round(1/Model.p)))))
        countCHs=countCHs+1;           %count node as cluster head
        TotalCH(countCHs).id=i; %adding in array of total CH
        Sensors(i).type='C';      %setting type of that node as cluster
        Sensors(i).G=round(1/Model.p)-1;    '%G' field is updated to contr
        % mark this cirle now that it has a CH
        CH_selected_arr(row_circle_of_node, col_circle_of_node) = 1;
    end
end

```

61

This part of the code checks whether a sensor node has energy greater than zero (`Sensors(i).E > 0`) and has not been previously chosen as a cluster head (`Sensors(i).G <= 0`). If these conditions are met, a random number (`temp_rand`) is generated between 0 and 1. If this random number is less than or equal to a certain probability, the node is elected as a cluster head. The probability of being elected as a cluster head is determined by the LEACH algorithm.

The `countCHs` variable keeps track of the total number of cluster heads. The information about the chosen cluster heads is stored in the `TotalCH` array. Additionally, the type of the selected node is set to 'C' to indicate that it is a cluster head. The `Sensors(i).G` field is updated to control the periodicity of cluster head selection. Finally, the circular region where a cluster head has been chosen is marked in the `CH_selected_arr` array to ensure that only one cluster head is selected per circular region.

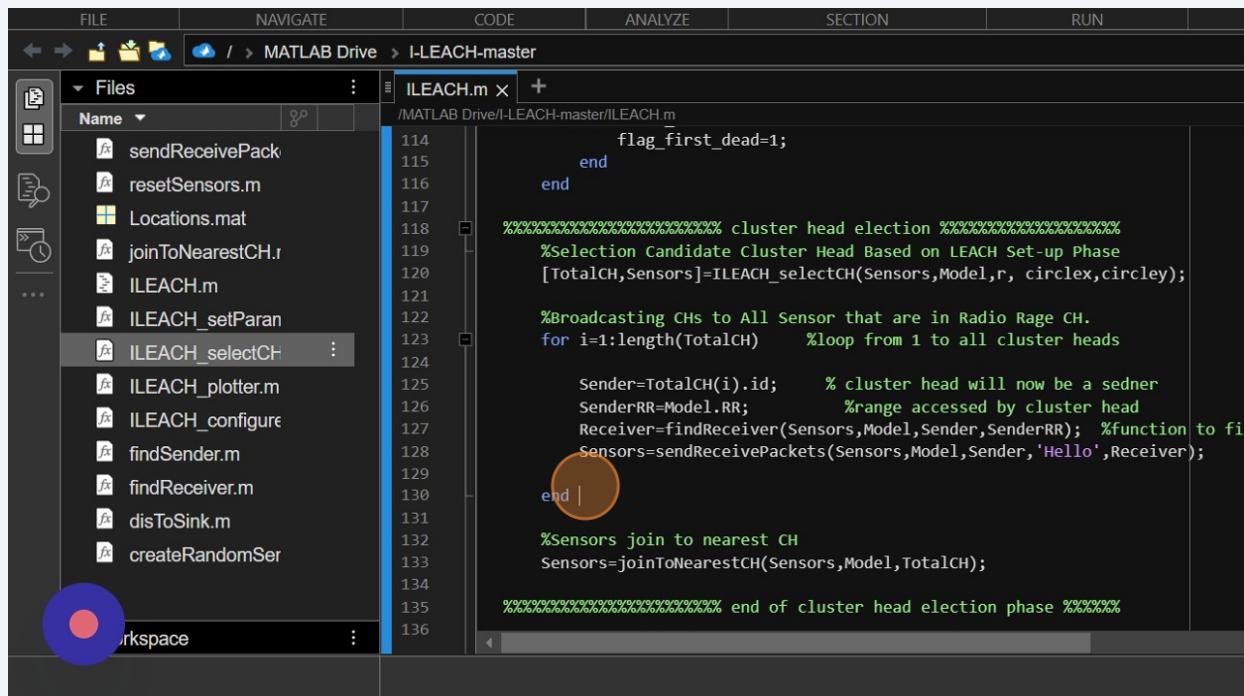


```

FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master
Files : ILEACH.m x ILEACH_selectCH.m +
Name /MATLAB Drive/I-LEACH-master/ILEACH_selectCH.m
sendReceivePack 46
resetSensors.m 47
Locations.mat 48
joinToNearestCH.r 49
ILEACH.m 50
ILEACH_setParam 51
ILEACH_selectCH 52
ILEACH_plotter.m 53
ILEACH_configure 54
findSender.m 55
findReceiver.m 56
disToSink.m 57
createRandomSer 58
TotalCH(countCHs).id=i; %adding in array of total CH
Sensors(i).type='C'; %setting type of that node as cluster
Sensors(i).G=round(1/Model.p)-1; %'G' field is updated to contr
% mark this circle now that it has a CH
CH_selected_arr(row_circle_of_node, col_circle_of_node) = 1;
end | end
end

```

62 Click here.



The screenshot shows the MATLAB IDE interface with the following details:

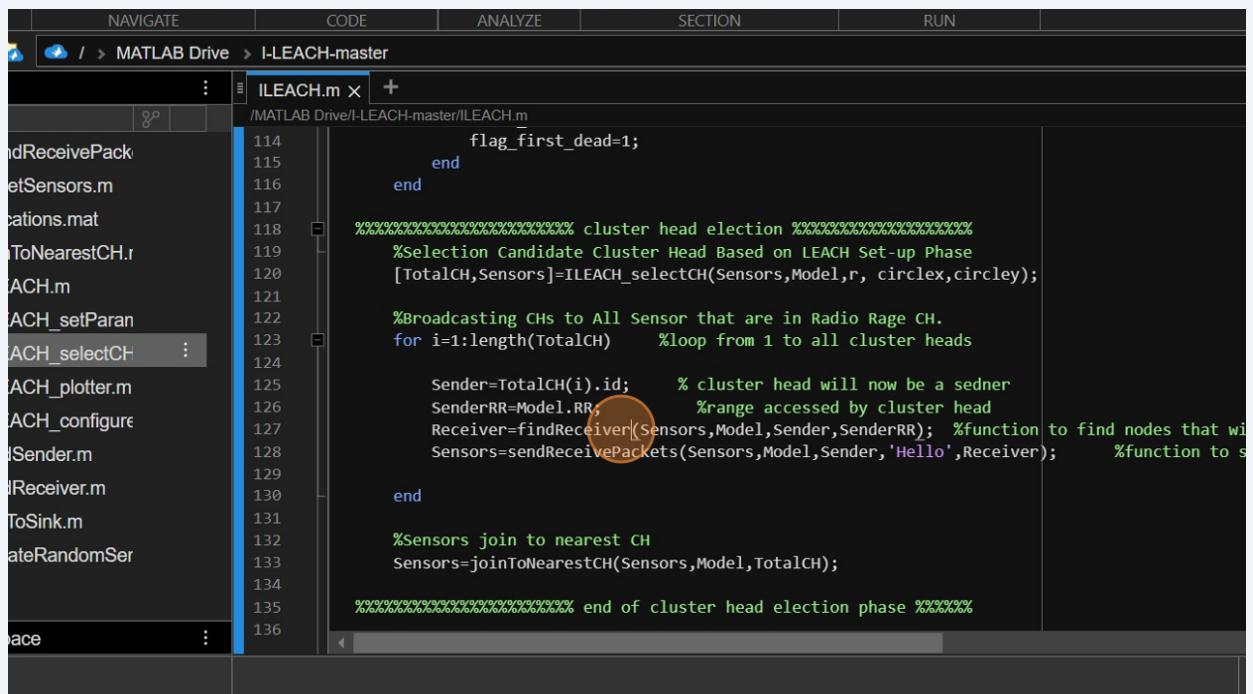
- File Explorer:** Shows files in the "Files" folder: sendReceivePack, resetSensors.m, Locations.mat, joinToNearestCH.r, ILEACH.m, ILEACH_setParam, ILEACH_selectCH, ILEACH_plotter.m, ILEACH_configure, findSender.m, findReceiver.m, disToSink.m, createRandomSer.
- Current File:** ILEACH.m (selected)
- Code Editor:** Displays the MATLAB script ILEACH.m. The code is related to cluster head election and sensor joining. A yellow circle highlights the line "end |" at the end of a loop structure.

```
114 flag_first_dead=1;
115
116 end
117
118 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% cluster head election %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
119 %Selection Candidate Cluster Head Based on LEACH Set-up Phase
120 [TotalCH,Sensors]=ILEACH_selectCH(Sensors,Model,r, circlex,circley);
121
122 %Broadcasting CHs to All Sensor that are in Radio Range CH.
123 for i=1:length(TotalCH) %loop from 1 to all cluster heads
124
125 Sender=TotalCH(i).id; % cluster head will now be a sender
126 SenderRR=Model.RR; %range accessed by cluster head
127 Receiver=findReceiver(Sensors,Model,Sender,SenderRR); %function to find
128 Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver);
129
130
131
132
133
134
135 %Sensors join to nearest CH
136 Sensors=joinToNearestCH(Sensors,Model,TotalCH);
137
138 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% end of cluster head election phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

63 Here's what each line does:

- for i = 1:length(TotalCH): Loop through all cluster heads stored in the TotalCH array.
- Sender = TotalCH(i).id;: Set the current cluster head as the sender.
- SenderRR = Model.RR;: Set the radio range of the sender (cluster head).
- Receiver = findReceiver(Sensors, Model, Sender, SenderRR);: Find nodes that are in the radio range of the current cluster head.
- Sensors = sendReceivePackets(Sensors, Model, Sender, 'Hello', Receiver);: Use the sendReceivePackets function to simulate the sending of "Hello" packets from the cluster head to the identified receivers.

This part of the code ensures that each cluster head broadcasts "Hello" packets to all sensor nodes within its radio range.

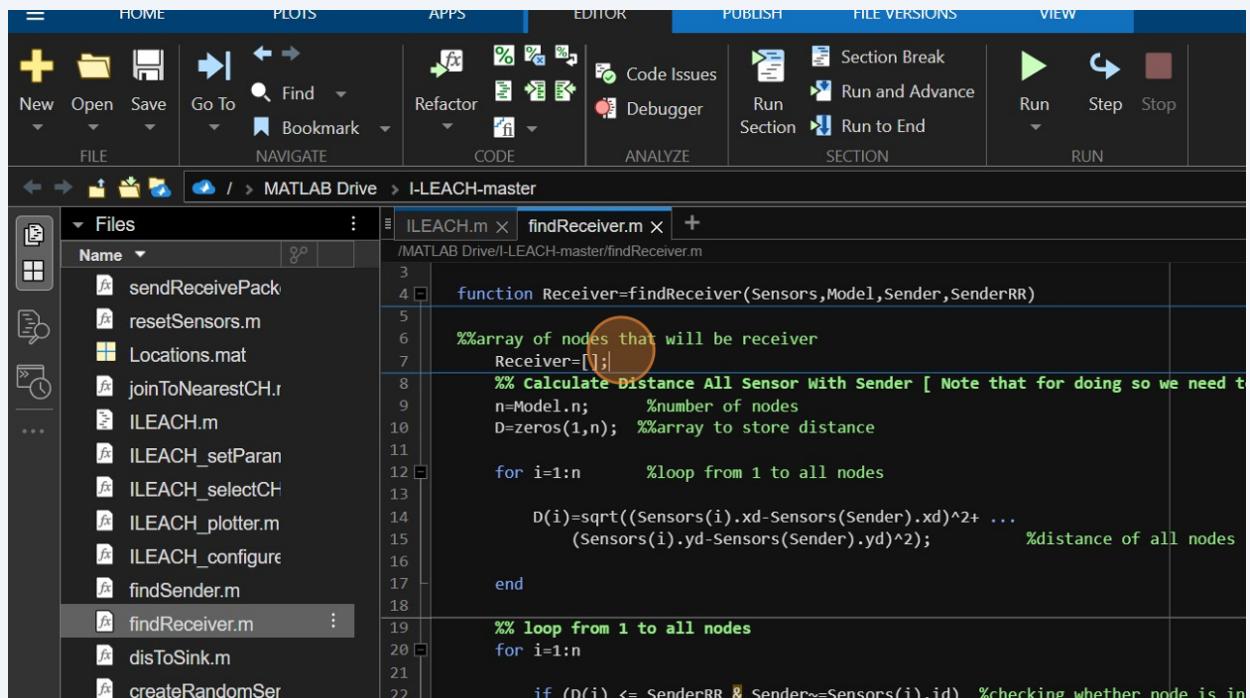


```
NAVIGATE CODE ANALYZE SECTION RUN
I / > MATLAB Drive > I-LEACH-master
ILEACH.m x +
/MATLAB Drive/I-LEACH-master/ILEACH.m
114     flag_first_dead=1;
115 end
116 end
117
118 %%%%%%%%%%%%%% cluster head election %%%%%%%%%%%%%%
119 %Selection Candidate Cluster Head Based on LEACH Set-up Phase
120 [TotalCH,Sensors]=ILEACH_SelectCH(Sensors,Model,r, circlex,circley);
121
122 %Broadcasting CHs to All Sensor that are in Radio Range CH.
123 for i=1:length(TotalCH)    %loop from 1 to all cluster heads
124
125     Sender=TotalCH(i).id;      % cluster head will now be a sender
126     SenderRR=Model.RR;        %range accessed by cluster head
127     Receiver=findReceiver(Sensors,Model,Sender,SenderRR); %function to find nodes that wi
128     Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver);    %function to s
129
130 end
131
132 %Sensors join to nearest CH
133 Sensors=joinToNearestCH(Sensors,Model,TotalCH);
134
135
136 %%%%%%%%%%%%%% end of cluster head election phase %%%%%%%%%%
```

64

The function findReceiver has a file of code lets observe that here:
 Receiver: An array that will store the IDs of potential receiver nodes.

- Calculate Distance of All Sensors with Sender: It calculates the Euclidean distance between the specified sender and all other sensor nodes.



```

/MATLAB Drive/I-LEACH-master/findReceiver.m
3
4 function Receiver=findReceiver(Sensors,Model,Sender,SenderRR)
5
6 %array of nodes that will be receiver
7 Receiver=[];
8 %% Calculate Distance All Sensor With Sender [ Note that for doing so we need to
9 n=Model.n; %number of nodes
10 D=zeros(1,n); %array to store distance
11
12 for i=1:n %loop from 1 to all nodes
13
14     D(i)=sqrt((Sensors(i).xd-Sensors(Sender).xd)^2+ ...
15                 (Sensors(i).yd-Sensors(Sender).yd)^2); %distance of all nodes
16
17 end
18
19 %% loop from 1 to all nodes
20 for i=1:n
21
22     if (D(i) <= SenderRR & Sender~=Sensors(i).id) %checking whether node is in

```

65

- for $i = 1:n$: Loop through all nodes to calculate distances.

- if ($D(i) \leq SenderRR$ & $Sender \neq Sensors(i).id$): Check if the distance between the sender and a node is within the communication range ($SenderRR$) and the node is not the sender itself.

The screenshot shows the MATLAB IDE interface with the following details:

- File Explorer:** Shows files in the 'I-LEACH-master' folder, including 'ILEACH.m', 'findReceiver.m', 'sendReceivePack.m', 'resetSensors.m', 'Locations.mat', 'joinToNearestCH.m', 'ILEACH.m', 'ILEACH_setParam.m', 'ILEACH_selectCH.m', 'ILEACH_plotter.m', 'ILEACH_configure.m', 'findSender.m', 'findReceiver.m', 'disToSink.m', and 'createRandomSer.m'. The 'findReceiver.m' file is currently selected.
- Editor:** Displays the 'findReceiver.m' code. A portion of the code is highlighted with a yellow oval, specifically the section from line 19 to line 24. The highlighted code is:

```
% loop from 1 to all nodes
for i=1:n
    if (D(i) <= SenderRR & Sender~=Sensors(i).id) %checking whether node is in
        Receiver=[Receiver,Sensors(i).id]; %adding to receiver array
    end
end
```

66

- Receiver = [Receiver, Sensors(i).id];: If the conditions are met, add the node to the receiver array.

The function essentially finds and returns the set of nodes that are potential receivers within the communication range of the specified sender

```
FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive / > MATLAB Drive > I-LEACH-master
Files : ILEACH.m x findReceiver.m x +
/MATLAB Drive/I-LEACH-master/findReceiver.m
6 %%%%%% OF NODES THAT WILL BE RECEIVER
7
8 function Receiver=findReceiver(Sensors,Model,Sender,SenderRR)
9
10 % Calculate Distance All Sensor With Sender [ Note that for doing so we need to a
11 n=Model.n; %number of nodes
12 D=zeros(1,n); %array to store distance
13
14 for i=1:n %loop from 1 to all nodes
15
16 D(i)=sqrt((Sensors(i).xd-Sensors(Sender).xd)^2+ ...
17 (Sensors(i).yd-Sensors(Sender).yd)^2); %distance of all nodes
18
19 %% loop from 1 to all nodes
20 for i=1:n
21
22 if (D(i) <= SenderRR & Sender~=Sensors(i).id) %checking whether node is in ran
23 Receiver=[Receiver,Sensors(i).id]; %adding to receiver array
24 end
25
26 end %%end of loop
27
28 end
```

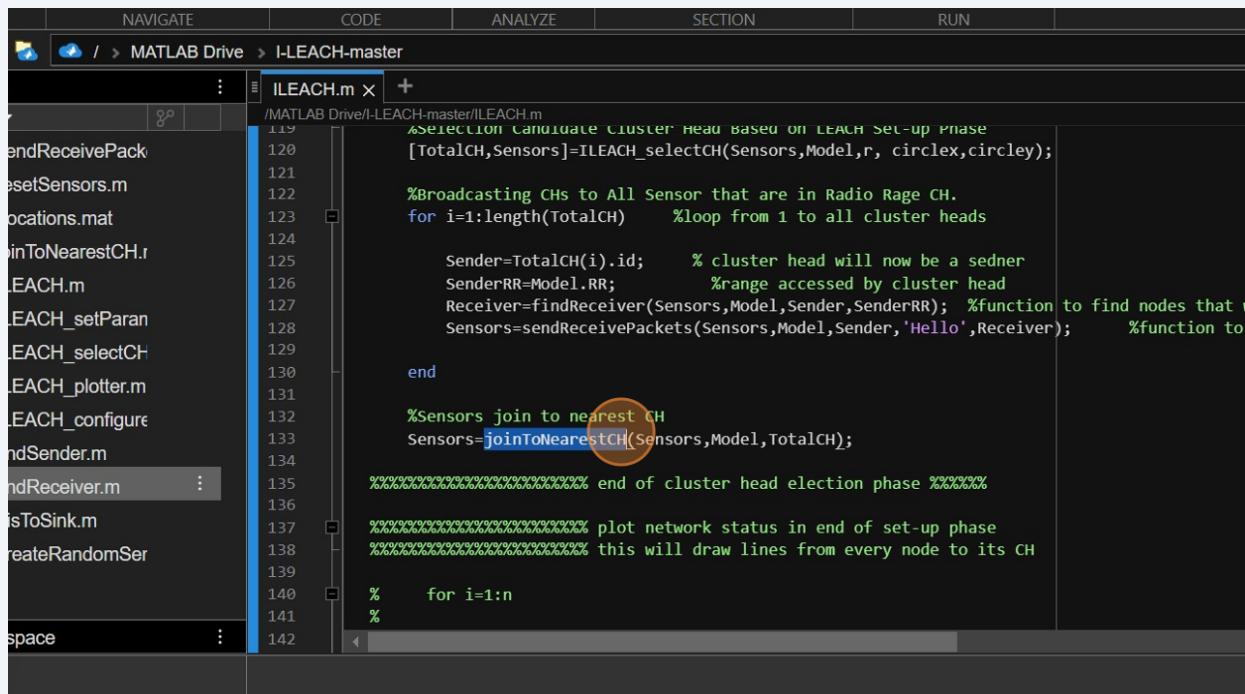
67

Now lets move back to main File ILEACH

```
HOME PLOTS APPS EDITOR PUBLISH FILE VERSIONS VIEW
FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive / > MATLAB Drive > I-LEACH-master
Files : ILEACH.m x +
/MATLAB Drive/I-LEACH-master/ILEACH.m
113 first_dead=r; %first dead node
114
115 end
116 end
117
118 %%%%%%%%%%%%%%% cluster head election %%%%%%%%%%%%%%
119 %Selection Candidate Cluster Head Based on LEACH Set-up Phase
120 [TotalCH,Sensors]=ILEACH_selectCH(Sensors,Model,r, circlex,circley);
121
122 %Broadcasting CHs to All Sensor that are in Radio Range CH.
123 for i=1:length(TotalCH) %loop from 1 to all cluster heads
124
125 Sender=TotalCH(i).id; % cluster head will now be a sender
126 SenderRR=Model.RR; %range accessed by cluster head
127 Receiver=findReceiver(Sensors,Model,Sender,SenderRR); %function to find
128 Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver); %
129
130
131 end
```

68

Now lets dicuss the next function used here for the sensors join to nearest Cluster Head



```

NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive / > MATLAB Drive > I-LEACH-master
ILEACH.m x +
/MATLAB Drive/I-LEACH-master/ILEACH.m
%Selection Candidate Cluster Head Based On LEACH Set-up Phase
[TotalCH,Sensors]=ILEACH_selectCH(Sensors,Model,r, circlex,circley);

%Broadcasting CHs to All Sensor that are in Radio Range CH.
for i=1:length(TotalCH) %loop from 1 to all cluster heads

    Sender=TotalCH(i).id; % cluster head will now be a sender
    SenderRR=Model.RR; %range accessed by cluster head
    Receiver=findReceiver(Sensors,Model,Sender,SenderRR); %function to find nodes that v
    Sensors=sendReceivePackets(Sensors,Model,Sender,'Hello',Receiver); %function to

end

%Sensors join to nearest CH
Sensors=joinToNearestCH(Sensors,Model,TotalCH);

%%%%%%%%% end of cluster head election phase %%%%%%
%%%%%%%%% plot network status in end of set-up phase
%%%%%%%%% this will draw lines from every node to its CH

%     for i=1:n
%
```

69

Here's a step-by-step explanation:

- $n=Model.n$: Assign the total number of nodes in the network to the variable n.
- $m=length(TotalCH)$: Calculate the total number of clusters by getting the length of the TotalCH array, which contains information about cluster heads.
- $\text{if}(m>1)$: Check if there is more than one cluster head (if there is at least one cluster).
- $D=zeros(m,n)$: Initialize a matrix D with dimensions m (number of clusters) by n (number of nodes) to store the distances.
- The nested loops (for $i=1:n$ and for $j=1:m$) iterate over all nodes and cluster heads, respectively.
- Inside the nested loops, it calculates the Euclidean distance between each node ($Sensors(i)$) and each cluster head ($Sensors(TotalCH(j).id)$), then stores the result in the matrix D.

The resulting matrix D contains distances between each non-cluster head node and each cluster head. Each row corresponds to a cluster head, and each column corresponds to a non-cluster head node.

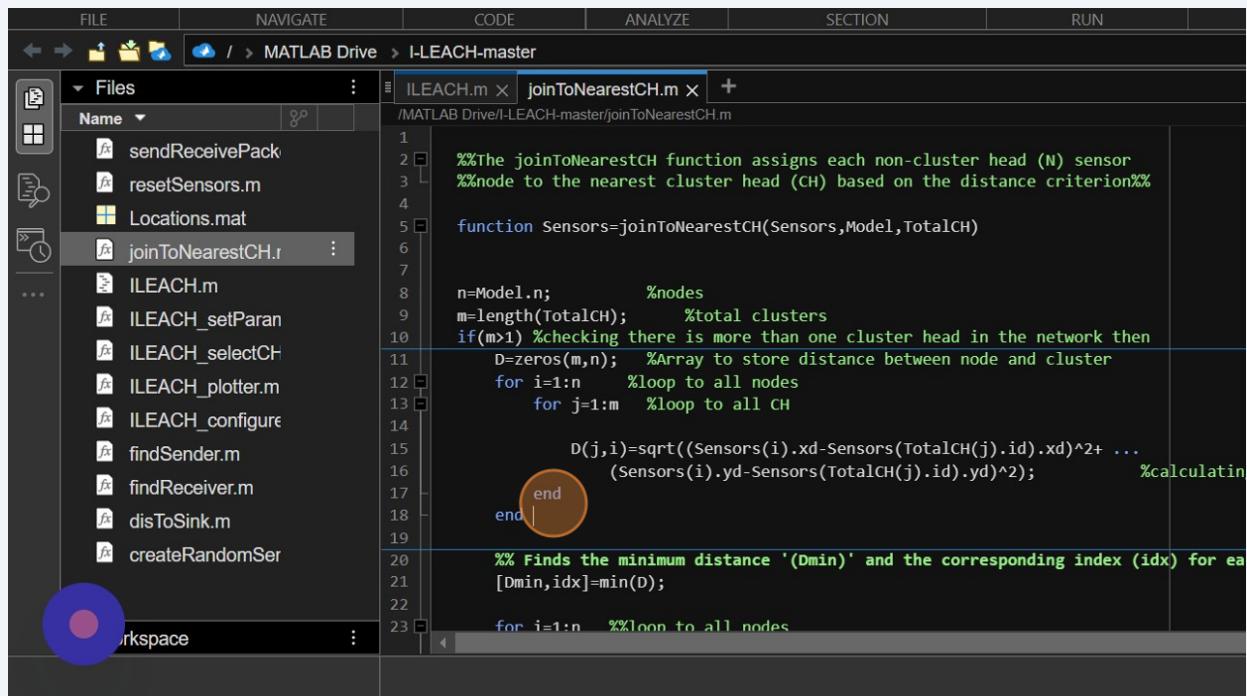
The screenshot shows the MATLAB IDE interface with the following details:

- Toolbar:** Includes Bookmarks, Refactor, Code, Debugger, Run, Run to End, SECTION, Run, Step, Stop, and RUN buttons.
- Current Path:** / > MATLAB Drive > I-LEACH-master
- Code Editor:** Displays the `joinToNearestCH.m` file content. The code is as follows:

```
%>>> joinToNearestCH.m
%The joinToNearestCH function assigns each non-cluster head (N) sensor
%node to the nearest cluster head (CH) based on the distance criterion%%
function Sensors=joinToNearestCH(Sensors,Model,TotalCH)
n=Model.n; %nodes
m=length(TotalCH); %total clusters
if(m>1) %checking there is more than one cluster head in the network then
    D=zeros(m,n); %Array to store distance between node and cluster
    for i=1:n %loop to all nodes
        for j=1:m %loop to all CH
            D(j,i)=sqrt((Sensors(i).xd-Sensors(TotalCH(j).id).xd)^2+ ...
                (Sensors(i).yd-Sensors(TotalCH(j).id).yd)^2); %calculating distance and storing
        end
    end
    % Finds the minimum distance '(Dmin)' and the corresponding index (idx) for each non-cluster head no
    [Dmin,idx]=min(D);
    %Loop to all nodes
    for i=1:n
        Sensors(i).CH_idx=idx;
    end
end
```

A red circle highlights the line `if(m>1)`.

70 Continued



The screenshot shows the MATLAB IDE interface with the following details:

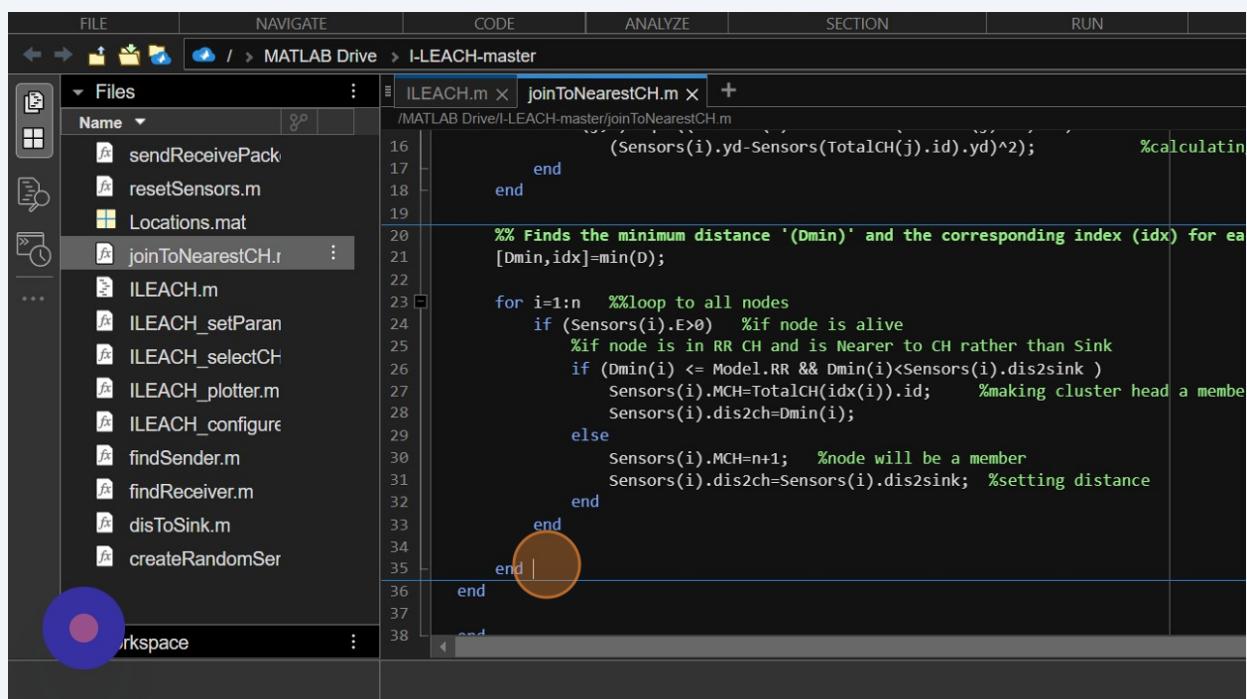
- File Menu:** FILE, NAVIGATE, CODE, ANALYZE, SECTION, RUN.
- Current Path:** MATLAB Drive > I-LEACH-master.
- Files:** sendReceivePack, resetSensors.m, Locations.mat, joinToNearestCH.r, ILEACH.m, ILEACH_setParam, ILEACH_selectCH, ILEACH_plotter.m, ILEACH_configure, findSender.m, findReceiver.m, disToSink.m, createRandomSer.
- Code Editor:** The code for `joinToNearestCH.m` is displayed. The highlighted line is:

```
%>%The joinToNearestCH function assigns each non-cluster head (N) sensor
%>%node to the nearest cluster head (CH) based on the distance criterion%
function Sensors=joinToNearestCH(Sensors,Model,TotalCH)
n=Model.n; %nodes
m=length(TotalCH); %total clusters
if(m>1) %checking there is more than one cluster head in the network then
    D=zeros(m,n); %Array to store distance between node and cluster
    for i=1:n %loop to all nodes
        for j=1:m %loop to all CH
            D(j,i)=sqrt((Sensors(i).xd-Sensors(TotalCH(j).id).xd)^2+ ...
                (Sensors(i).yd-Sensors(TotalCH(j).id).yd)^2); %calculatin
        end %>%for loop to all CH
    end %>%for loop to all nodes
    [Dmin, idx]=min(D);
    for i=1:n %loop to all nodes
        Sensors(i).CH_idx=idx(i);
    end
end
```

71

- $[D_{min}, idx] = \min(D)$: This line finds the minimum distance (D_{min}) and the corresponding index (idx) for each column (non-cluster head node) in the distance matrix D . D_{min} is an array containing the minimum distance for each node, and idx is an array containing the index of the cluster head associated with the minimum distance.

- The subsequent loop (for $i = 1:n$) iterates through all nodes.
 - Inside the loop, it checks if the node is alive ($Sensors(i).E > 0$).
 - If the node is in the radio range of a cluster head and is closer to that cluster head than the sink, it assigns the cluster head as the member ($Sensors(i).MCH$) and updates the distance ($Sensors(i).dis2ch$).
 - Otherwise, it assigns the sink as the member and sets the distance to the sink.



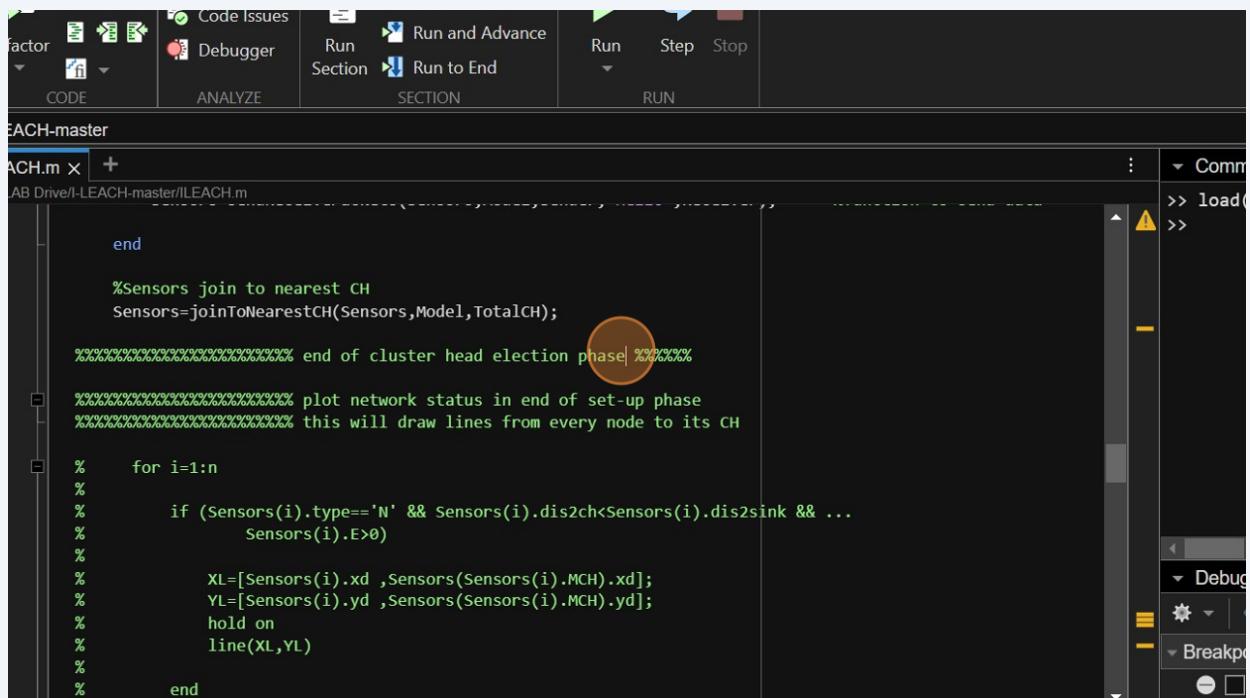
```

FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master
Files : ILEACH.m x joinToNearestCH.m x +
Name
sendReceivePack
resetSensors.m
Locations.mat
joinToNearestCH.r :
ILEACH.m
ILEACH_setParan
ILEACH_selectCH
ILEACH_plotter.m
ILEACH_configure
findSender.m
findReceiver.m
disToSink.m
createRandomSer
Workspace :
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
/MATLAB Drive/I-LEACH-master/joinToNearestCH.m
(Sensors(i).yd-Sensors(TotalCH(j).id).yd)^2);
end
end
%% Finds the minimum distance '(Dmin)' and the corresponding index (idx) for ea
[Dmin,idx]=min(D);
for i=1:n %%loop to all nodes
if (Sensors(i).E>0) %if node is alive
%if node is in RR CH and is Nearer to CH rather than Sink
if (Dmin(i) <= Model.RR && Dmin(i)<Sensors(i).dis2sink )
Sensors(i).MCH=TotalCH(idx(i)).id; %making cluster head a member
Sensors(i).dis2ch=Dmin(i);
else
Sensors(i).MCH=n+1; %node will be a member
Sensors(i).dis2ch=Sensors(i).dis2sink; %setting distance
end
end
end
end
end

```

72

Now moving back to main file ILEACH and now Here cluster head election phase ends



The screenshot shows the MATLAB IDE interface with the code editor open to the file `ILEACH.m`. The code is written in MATLAB and describes the end of the cluster head election phase. A yellow warning icon is visible in the top right corner of the code editor. The code includes comments indicating the joining of sensors to nearest CH and the plotting of network status.

```
%Sensors join to nearest CH
Sensors=joinToNearestCH(Sensors,Model,TotalCH);

%%%%%%%%%%%%% end of cluster head election phase %%%%%%%

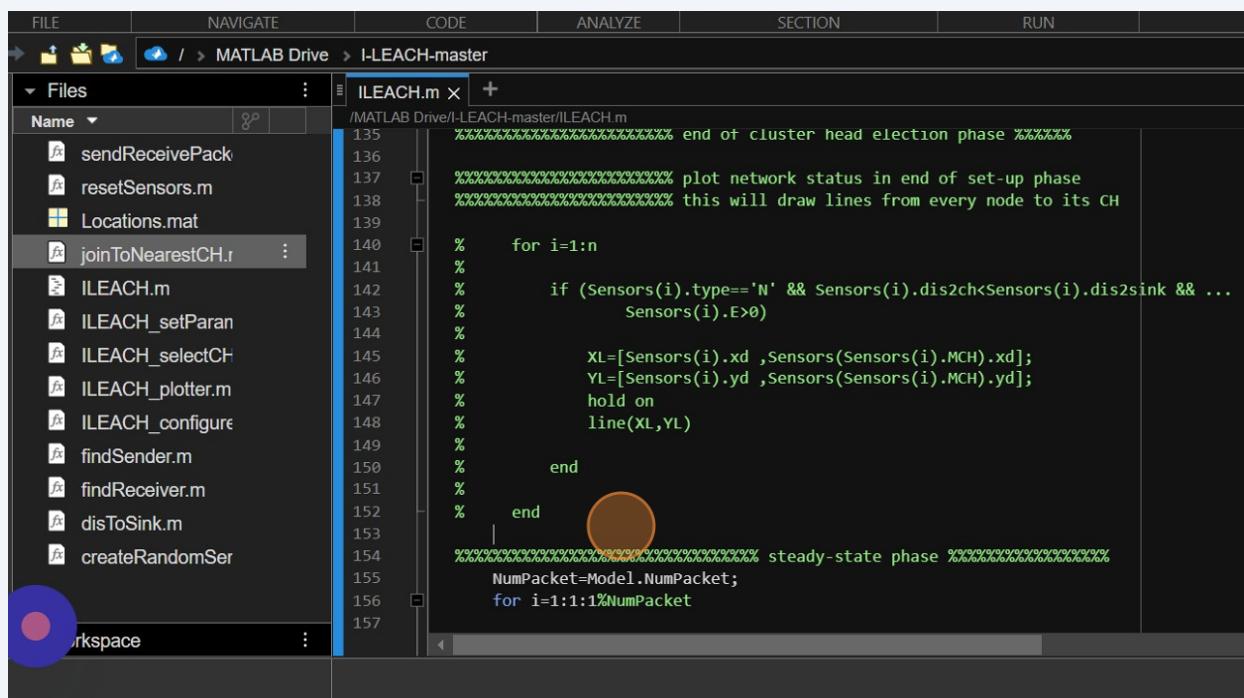
%%%%%%%%%%%%% plot network status in end of set-up phase
%%%%%%%%%%%%% this will draw lines from every node to its CH

% for i=1:n
%
% if (Sensors(i).type=='N' & Sensors(i).dis2ch<Sensors(i).dis2sink && ...
% Sensors(i).E>0)
%
%     XL=[Sensors(i).xd ,Sensors(Sensors(i).MCH).xd];
%     YL=[Sensors(i).yd ,Sensors(Sensors(i).MCH).yd];
%     hold on
%     line(XL,YL)
%
% end
```

73

- The loop iterates through all nodes.
- It checks if the node is a normal node (`Sensors(i).type == 'N'`), closer to its cluster head (`Sensors(i).dis2ch < Sensors(i).dis2sink`), and still has energy (`Sensors(i).E > 0`).
- If the conditions are met, it defines the X and Y coordinates (XL and YL) for the line connecting the node to its cluster head.
- `hold on` is used to ensure that subsequent plots are overlaid on the current figure.
- Finally, `line(XL, YL)` draws a line from the node to its cluster head.

This visualization helps illustrate the connectivity between normal nodes and their respective cluster heads in the network.



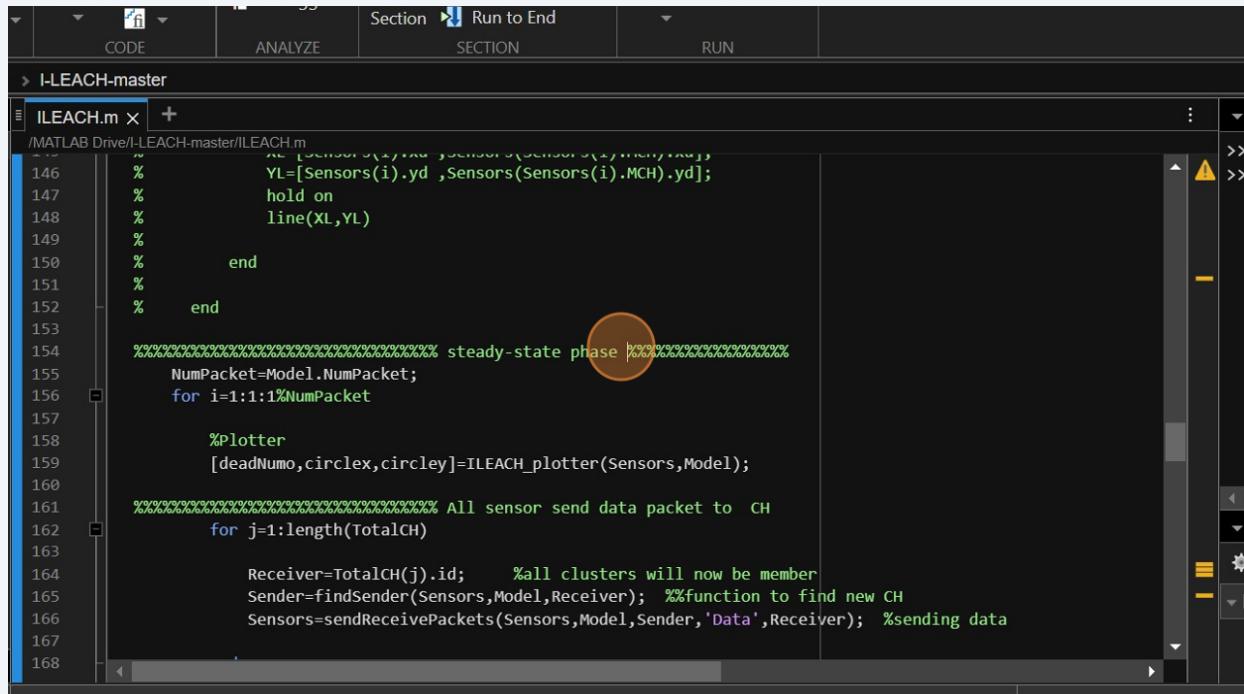
The screenshot shows the MATLAB IDE interface with the following details:

- File Menu:** FILE, NAVIGATE, CODE, ANALYZE, SECTION, RUN.
- Current Path:** MATLAB Drive > I-LEACH-master.
- Code Editor:** The file ILEACH.m is open. The code is as follows:

```
%%%%%%%% end of cluster head election phase %%%%%%
%%%%%%%% plot network status in end of set-up phase
%%%%%%%% this will draw lines from every node to its CH
%
% for i=1:n
%
% if (Sensors(i).type=='N' && Sensors(i).dis2ch<Sensors(i).dis2sink && ...
% Sensors(i).E>0)
%
%     XL=[Sensors(i).xd ,Sensors(Sensors(i).MCH).xd];
%     YL=[Sensors(i).yd ,Sensors(Sensors(i).MCH).yd];
%     hold on
%     line(XL,YL)
%
% end
%
% end
%
%%%%%%%% steady-state phase %%%%%%
NumPacket=Model.NumPacket;
for i=1:1:1%NumPacket
```

74

Now here steady state phase is implemented Lets have an explanation of how it works



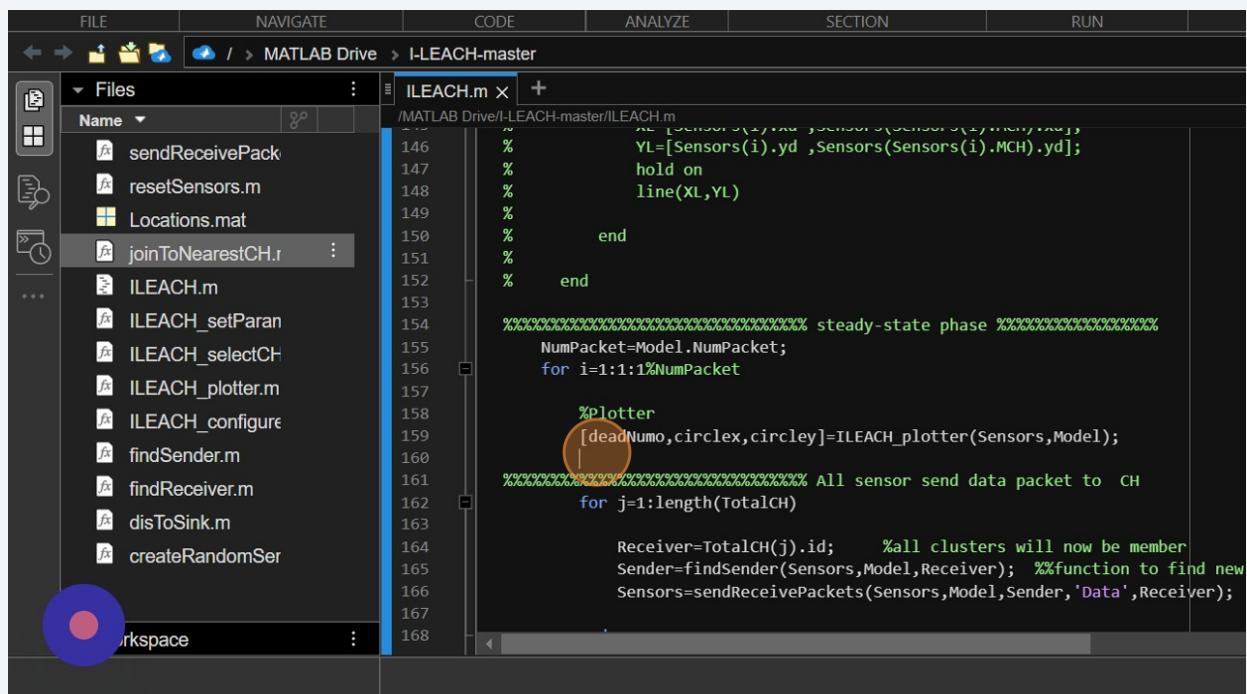
```
/MATLAB Drive/I-LEACH-master/I-LEACH.m
146 %           XL=[Sensors(i).x ,Sensors(i).Sensor(i).MCN.x];
147 %           YL=[Sensors(i).y ,Sensors(i).Sensor(i).MCN.y];
148 %           hold on
149 %           line(XL,YL)
150 %
151 %
152 %       end
153 %
154 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% steady-state phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
155 NumPacket=Model.NumPacket;
156 for i=1:1:NumPacket
157
158     %Plotter
159     [deadNumo,circlex,circley]=I-LEACH_plotter(Sensors,Model);
160
161 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% All sensor send data packet to CH
162 for j=1:length(TotalCH)
163
164     Receiver=TotalCH(j).id;      %all clusters will now be member
165     Sender=findSender(Sensors,Model,Receiver);  %%function to find new CH
166     Sensors=sendReceivePackets(Sensors,Model,Sender,'Data',Receiver);  %sending data
167
168
```

75

Here we're starting a loop to simulate packet transmissions (presumably data packets).

`NumPacket = Model.NumPacket;` retrieves the number of packets (NumPacket) from the simulation model (Model).

- The loop for `i = 1:1:1` seems to iterate only once, indicating that this block of code is specifically for the first packet.
- Inside the loop, the `I-LEACH_plotter` function is called to visualize the network status. The function returns the number of dead nodes (deadNumo) and circular regions (circlex and circley).



```

FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master
Files : ILEACH.m +
IMATLAB Drive/I-LEACH-master/ILEACH.m
146 % XL=[Sensors(i).x ,Sensors(i).y ,Sensors(i).x ,Sensors(i).y ];
147 %
148 % YL=[Sensors(i).yd ,sensors(Sensors(i).MCH).yd];
149 %
150 % hold on
151 %
152 % line(XL,YL)
153 %
154 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% steady-state phase %%%%%%
155 NumPacket=Model.NumPacket;
156 for i=1:1:1%NumPacket
157 %
158 %plotter
159 [deadNumo,circlex,circley]=I-LEACH_plotter(Sensors,Model);
160 %
161 %%%%% All sensor send data packet to CH
162 for j=1:length(TotalCH)
163 %
164 Receiver=TotalCH(j).id; %all clusters will now be member
165 Sender=findSender(Sensors,Model,Receiver); %function to find new
166 Sensors=sendReceivePackets(Sensors,Model,Sender,'Data',Receiver);
167 %
168

```

76

- The loop iterates through each cluster head (`TotalCH(j).id`).
- Receiver is set to the ID of the cluster head, as each cluster head is intended to receive data packets.
- Sender is determined using the `findSender` function. This function likely identifies the normal nodes in the vicinity of the cluster head that will send data packets.
- The `sendReceivePackets` function is then called to simulate the process of sending data packets from normal nodes to their respective cluster heads.

```
<!--> Debugger | Run | Step | Stop | <!-->
ANALYZE | Run Section | Run to End | SECTION | RUN | <!-->

er
+ : <!--> Command Wi...
ILEACH-master/ILEACH.m
%%%%% steady-state phase %%%%%%
NumPacket=Model.NumPacket;
for i=1:1:1%NumPacket

    %Plotter
    [deadNumo,circlex,circley]=ILEACH_plotter(Sensors,Model);

%%%%% All sensor send data packet to CH| <--> circled
for j=1:length(TotalCH)

    Receiver=TotalCH(j).id;      %all clusters will now be member
    Sender=findSender(Sensors,Model,Receiver);  %function to find new CH
    Sensors=sendReceivePackets(Sensors,Model,Sender,'Data',Receiver);  %sending data

end
end

%%%%% send Data packet from CH to Sink after Data aggregation
for i=1:length(TotalCH)
```

>> load "/MATLAB"

>>

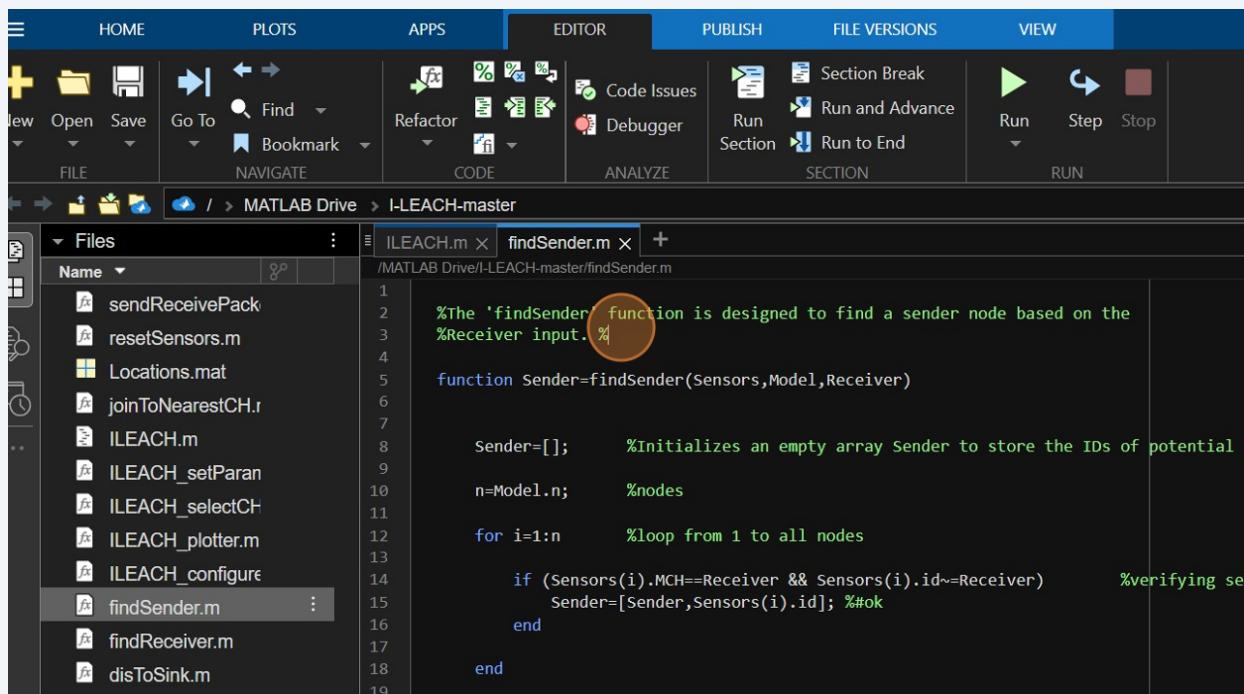
Debugger

Breakpoints

77

- The function loops through all nodes (Sensors) in the network.
- For each node, it checks if the node is a potential sender to the specified Receiver based on the conditions `Sensors(i).MCH == Receiver` and `Sensors(i).id ~= Receiver`.
- If a node satisfies the conditions, its ID is added to the Sender array.
- The final Sender array contains the IDs of potential sender nodes.

This function is used to identify nodes that should send data packets to a specific receiver, such as a cluster head.



The screenshot shows the MATLAB IDE interface with the following details:

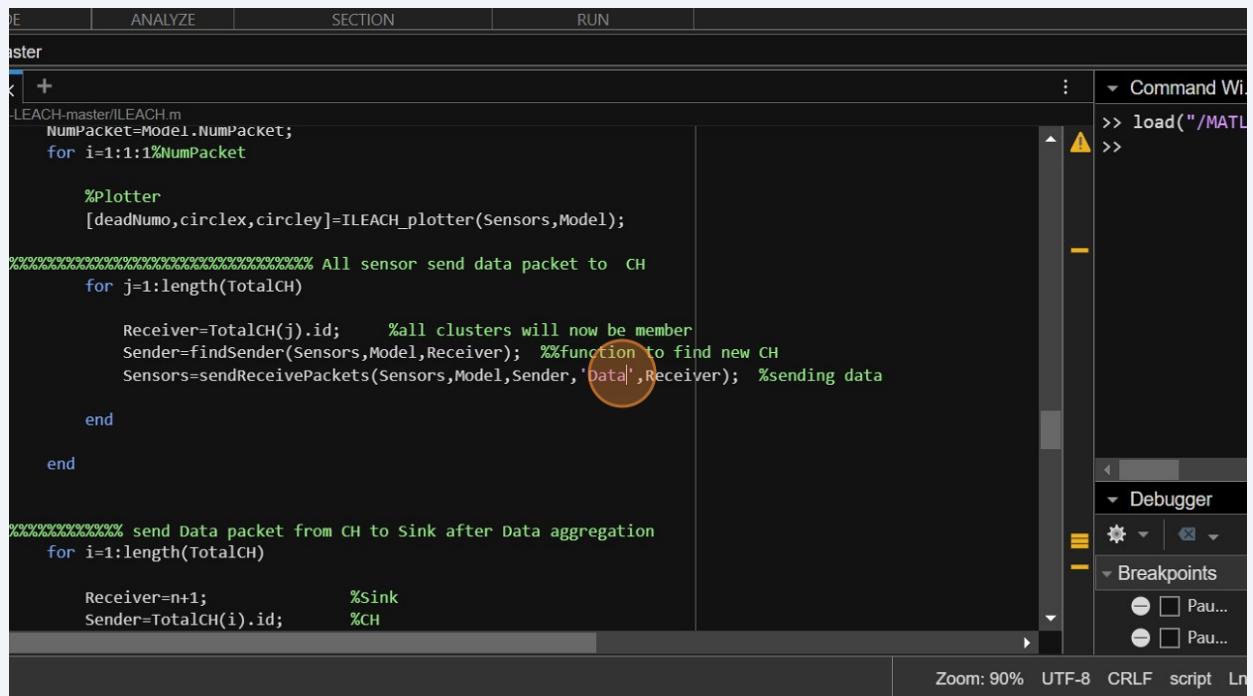
- Toolbar:** HOME, PLOTS, APPS, EDITOR (selected), PUBLISH, FILE VERSIONS, VIEW.
- File Explorer:** Shows files in the 'I-LEACH-master' folder, including ILEACH.m, ILEACH_setParan.m, ILEACH_selectCH.m, ILEACH_plotter.m, ILEACH_configure.m, findSender.m (selected), findReceiver.m, and disToSink.m.
- Editor:** Displays the content of findSender.m.

```

1 %The 'findSender' function is designed to find a sender node based on the
2 %Receiver input.
3
4 function Sender=findSender(Sensors,Model,Receiver)
5
6     Sender=[]; %Initializes an empty array Sender to store the IDs of potential
7     %nodes
8
9     n=Model.n;
10
11    for i=1:n %loop from 1 to all nodes
12
13        if (Sensors(i).MCH==Receiver && Sensors(i).id~=Receiver) %verifying se
14            Sender=[Sender,Sensors(i).id]; %#ok
15        end
16
17    end
18
19

```

78 Now here Data is send instead of Hello Messages



```
DE | ANALYZE | SECTION | RUN | master
+ -LEACH-master/ILEACH.m
NumPacket=Model.NumPacket;
for i=1:1:1%NumPacket

    %Plotter
    [deadNumo,circlex,circley]=ILEACH_plotter(Sensors,Model);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% All sensor send data packet to CH
for j=1:length(TotalCH)

    Receiver=TotalCH(j).id;      %all clusters will now be member
    Sender=findSender(Sensors,Model,Receiver);  %%function to find new CH
    Sensors=sendReceivePackets(Sensors,Model,Sender,'Data',Receiver);  %sending data
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% send Data packet from CH to Sink after Data aggregation
for i=1:length(TotalCH)

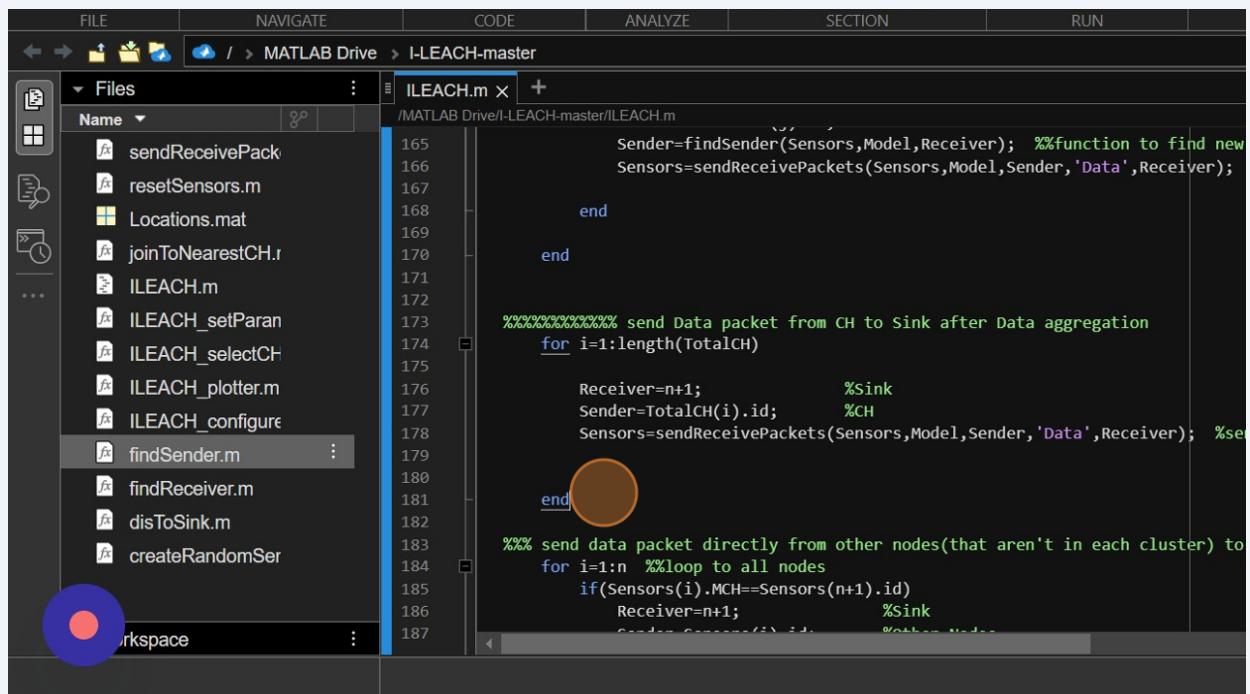
    Receiver=n+1;          %Sink
    Sender=TotalCH(i).id;  %CH

```

79

- The loop iterates over each Cluster Head (TotalCH is an array containing information about all cluster heads).
- For each Cluster Head, it sets the Receiver variable to the Sink ($n + 1$).
- It sets the Sender variable to the ID of the current Cluster Head (TotalCH(i).id).
- Calls the sendReceivePackets function to simulate the transmission of a data packet from the CH to the Sink. The function updates the energy levels of the involved nodes.

This part of the code represents the data aggregation phase where each Cluster Head forwards aggregated data to the Sink.



FILE NAVIGATE CODE ANALYZE SECTION RUN

/ MATLAB Drive > I-LEACH-master

Files

Name

- sendReceivePack
- resetSensors.m
- Locations.mat
- joinToNearestCH.r
- I-LEACH.m
- I-LEACH_setParam
- I-LEACH_selectCH
- I-LEACH_plotter.m
- I-LEACH_configure
- findSender.m
- findReceiver.m
- disToSink.m
- createRandomSer

ILEACH.m

```

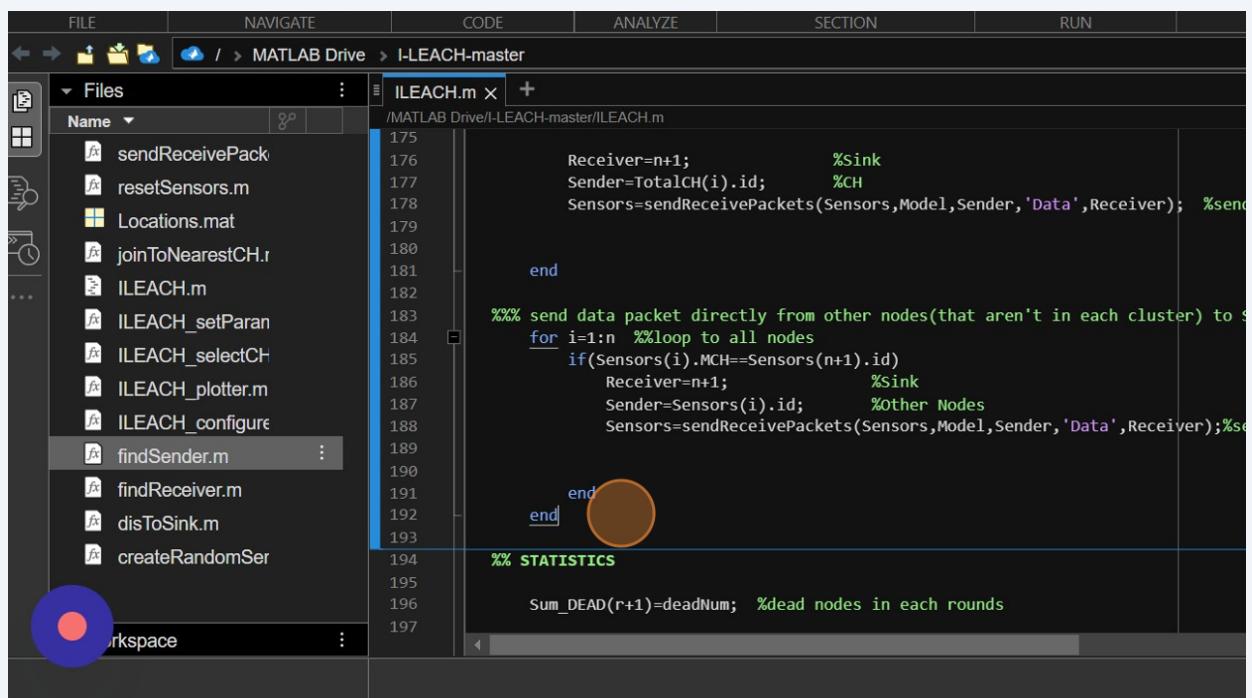
165
166
167
168
169
170
171
172
173 %%%% send Data packet from CH to Sink after Data aggregation
174 for i=1:length(TotalCH)
175
176     Receiver=n+1; %Sink
177     Sender=TotalCH(i).id; %CH
178     Sensors=sendReceivePackets(Sensors,Model,Sender,'Data',Receiver); %se
179
180
181
182
183 %%% send data packet directly from other nodes(that aren't in each cluster) to
184 for i=1:n %loop to all nodes
185     if(Sensors(i).MCH==Sensors(n+1).id)
186         Receiver=n+1; %sink
187

```

80

- The loop iterates over all nodes (n represents the total number of nodes).
 - It checks whether the current node (`Sensors(i)`) is a member of any cluster by verifying if its MCH (Member Cluster Head) ID is equal to the Sink ID (`Sensors(n + 1).id`).
 - If the node is a member of a cluster, it sets the Receiver variable to the Sink (`n + 1`).
 - It sets the Sender variable to the ID of the current node (`Sensors(i).id`).
 - Calls the `sendReceivePackets` function to simulate the transmission of a data packet from the non-cluster head node to the Sink. The function updates the energy levels of the involved nodes.

This part of the code represents the direct transmission of data packets from nodes that are not cluster heads to the Sink.

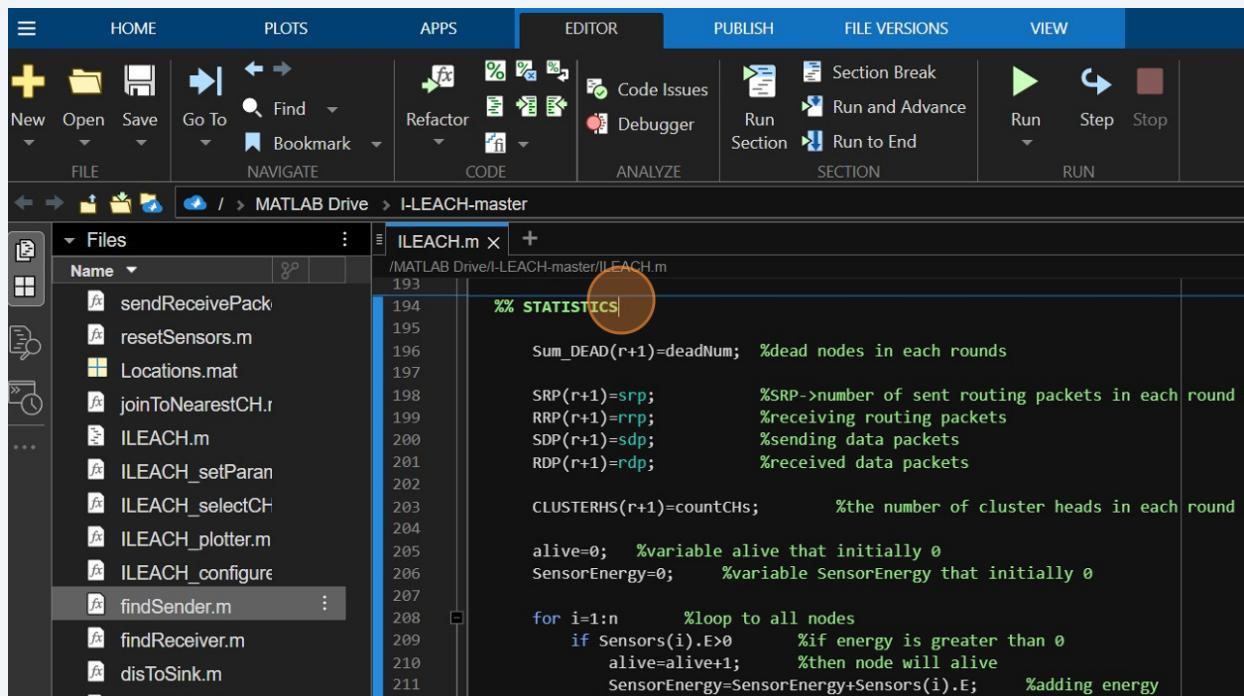


The screenshot shows the MATLAB IDE interface with the following details:

- File Explorer:** Shows files in the 'ILEACH-master' folder, including `sendReceivePack.m`, `resetSensors.m`, `Locations.mat`, `joinToNearestCH.r`, `ILEACH.m`, `ILEACH_setParan`, `ILEACH_selectCH`, `ILEACH_plotter.m`, `ILEACH_configure`, `findSender.m` (selected), `findReceiver.m`, `disToSink.m`, and `createRandomSer`.
- Editor:** Displays the `ILEACH.m` script. The code handles data transmission between nodes. A specific section of the code is highlighted with a brown circle around the `end` keyword at line 193. The code snippet is as follows:

```
175 Receiver=n+1; %Sink
176 Sender=TotalCH(i).id; %CH
177 Sensors=sendReceivePackets(Sensors,Model,Sender,'Data',Receiver); %send
178
179
180
181
182
183 %% send data packet directly from other nodes(that aren't in each cluster) to Sink
184 for i=1:n %loop to all nodes
185     if(Sensors(i).MCH==Sensors(n+1).id)
186         Receiver=n+1; %sink
187         Sender=Sensors(i).id; %Other Nodes
188         Sensors=sendReceivePackets(Sensors,Model,Sender,'Data',Receiver);%send
189
190
191
192
193 end %end
194
195
196
197 %% STATISTICS
198 Sum_DEAD(r+1)=deadNum; %dead nodes in each rounds
```

81 Now comes the statistics



The screenshot shows the MATLAB IDE interface. The top menu bar includes HOME, PLOTS, APPS, EDITOR (selected), PUBLISH, FILE VERSIONS, and VIEW. Below the menu is a toolbar with icons for New, Open, Save, Go To, Find, Refactor, Code Issues, Debugger, Run Section, Section Break, Run and Advance, Run to End, Run, Step, and Stop. The current workspace shows a folder structure under MATLAB Drive / I-LEACH-master, containing files like sendReceivePack, resetSensors.m, Locations.mat, joinToNearestCH.r, ILEACH.m, ILEACH_setParan, ILEACH_selectCH, ILEACH_plotter.m, ILEACH_configure, findSender.m (highlighted in blue), findReceiver.m, and disToSink.m. The main editor window displays the ILEACH.m script. A specific section of the code is highlighted with a red circle:

```
%% STATISTICS
Sum_DEAD(r+1)=deadNum; %dead nodes in each rounds
SRP(r+1)=srp; %SRP->number of sent routing packets in each round
RRP(r+1)=rrp; %receiving routing packets
SDP(r+1)=sdp; %sending data packets
RDP(r+1)=rdp; %received data packets
CLUSTERHS(r+1)=countCHs; %the number of cluster heads in each round
alive=0; %variable alive that initially 0
SensorEnergy=0; %variable SensorEnergy that initially 0
for i=1:n %loop to all nodes
    if Sensors(i).E>0 %if energy is greater than 0
        alive=alive+1; %then node will alive
        SensorEnergy=SensorEnergy+Sensors(i).E; %adding energy
```

82

- Sum_DEAD(r + 1): Records the number of dead nodes in each round.
- SRP(r + 1): Records the number of sent routing packets in each round.
- RRP(r + 1): Records the number of received routing packets in each round.
- SDP(r + 1): Records the number of sent data packets in each round.
- RDP(r + 1): Records the number of received data packets in each round.
- CLUSTERHS(r + 1): Records the number of cluster heads in each round.
- alive: A variable initialized to 0. It appears to be used to track the number of alive nodes, but the specific implementation details are not provided in this segment.
- SensorEnergy: A variable initialized to 0. It seems to be related to tracking the total energy of the sensor nodes, but the specific details are not provided in this segment.

These variables are likely used for monitoring and analyzing the performance of the simulation over multiple rounds

The screenshot shows the MATLAB IDE interface with the following details:

- ODE**, **ANALYZE**, **SECTION**, and **RUN** tabs are visible at the top.
- The current file is `I-L-LEACH-master/I-LEACH.m`.
- The code in the editor is:

```
%% STATISTICS
Sum_DEAD(r+1)=deadNum; %dead nodes in each rounds
SRP(r+1)=srp;           %SRP->number of sent routing packets in each round %%srp->number of sent routing
RRP(r+1)=rrp;           %receiving routing packets
SDP(r+1)=sdp;           %sending data packets
RDP(r+1)=rdp;           %received data packets

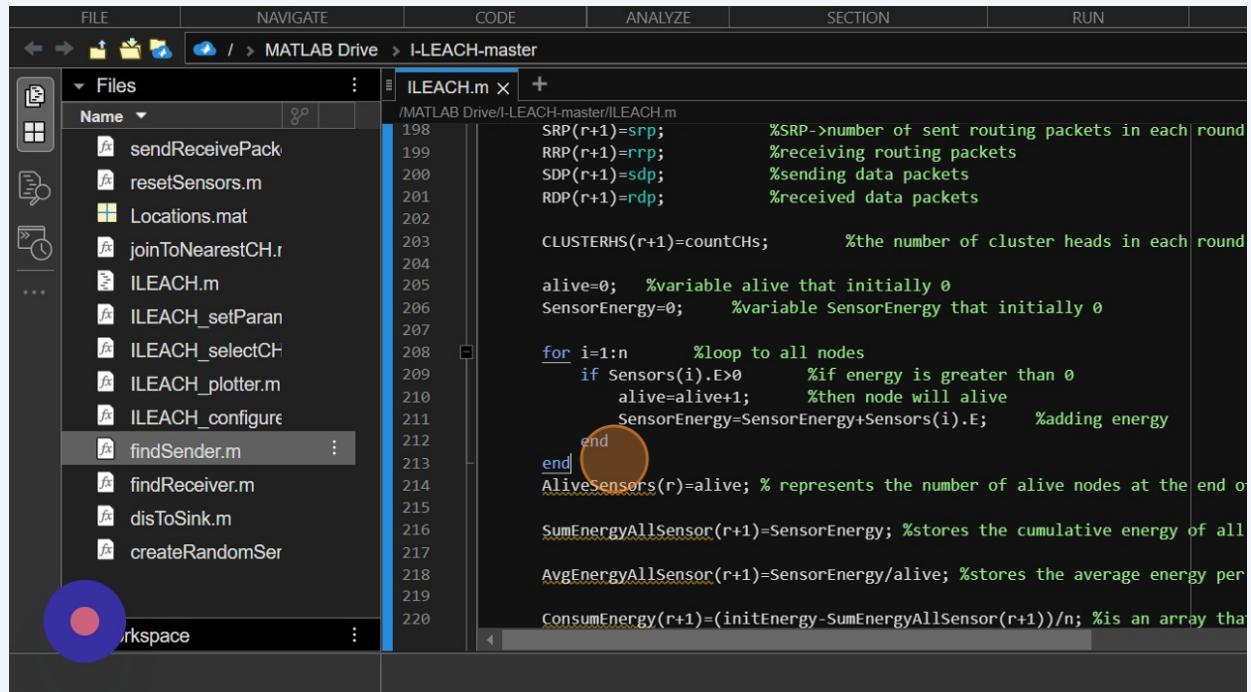
CLUSTERHS(r+1)=countCHs; %the number of cluster heads in each round

alive=0;    %variable alive that initially 0
SensorEnergy=0; %variable SensorEnergy that initially 0 | (highlighted with a yellow circle)

for i=1:n      %loop to all nodes
    if Sensors(i).E>0      %if energy is greater than 0
        alive=alive+1;      %then node will alive
        SensorEnergy=SensorEnergy+Sensors(i).E;    %adding energy
    end
end
AliveSensors(r)=alive; % represents the number of alive nodes at the end of the current round.
```
- The right pane shows the **Command Window** with the command `>> load("MA")` and the **Debugger** and **Breakpoints** toolbars.
- The bottom status bar shows **Zoom: 90%**, **UTF-8**, **CRLF**, and **script**.

83

- alive: This counter is incremented for each node with energy greater than 0, representing the number of alive nodes.
- SensorEnergy: This variable accumulates the total energy of all nodes with energy greater than 0. It provides a measure of the total energy in the network.



The screenshot shows the MATLAB IDE interface with the following details:

- File Explorer:** On the left, under the "Files" tab, there is a list of files including sendReceivePack, resetSensors.m, Locations.mat, joinToNearestCH.r, ILEACH.m, ILEACH_setParam, ILEACH_selectCH, ILEACH_plotter.m, ILEACH_configure, findSender.m, findReceiver.m, disToSink.m, and createRandomSer.
- Current File:** The main editor window displays the code for **ILEACH.m**. The code is a MATLAB script for the I-LÉACH algorithm. A red circle highlights the word "end" at line 212, which marks the end of a loop that iterates over all nodes.
- Code Content:** The script starts by defining variables SRP, RRP, SDP, RDP, CLUSTERHS, alive, and SensorEnergy. It then enters a loop for each node (i=1:n). Inside the loop, it checks if the sensor's energy (Sensors(i).E) is greater than 0. If true, it increments the alive counter and adds the node's energy to SensorEnergy. After the loop, it calculates the number of alive nodes (AliveSensors(r)) and the average energy per alive node (AvgEnergyAllSensor(r+1)). Finally, it calculates the cumulative energy of all alive nodes (SumEnergyAllSensor(r+1)).

```
SRP(r+1)=srp; %SRP->number of sent routing packets in each round
RRP(r+1)=rrp; %receiving routing packets
SDP(r+1)=sdp; %sending data packets
RDP(r+1)=rdp; %received data packets

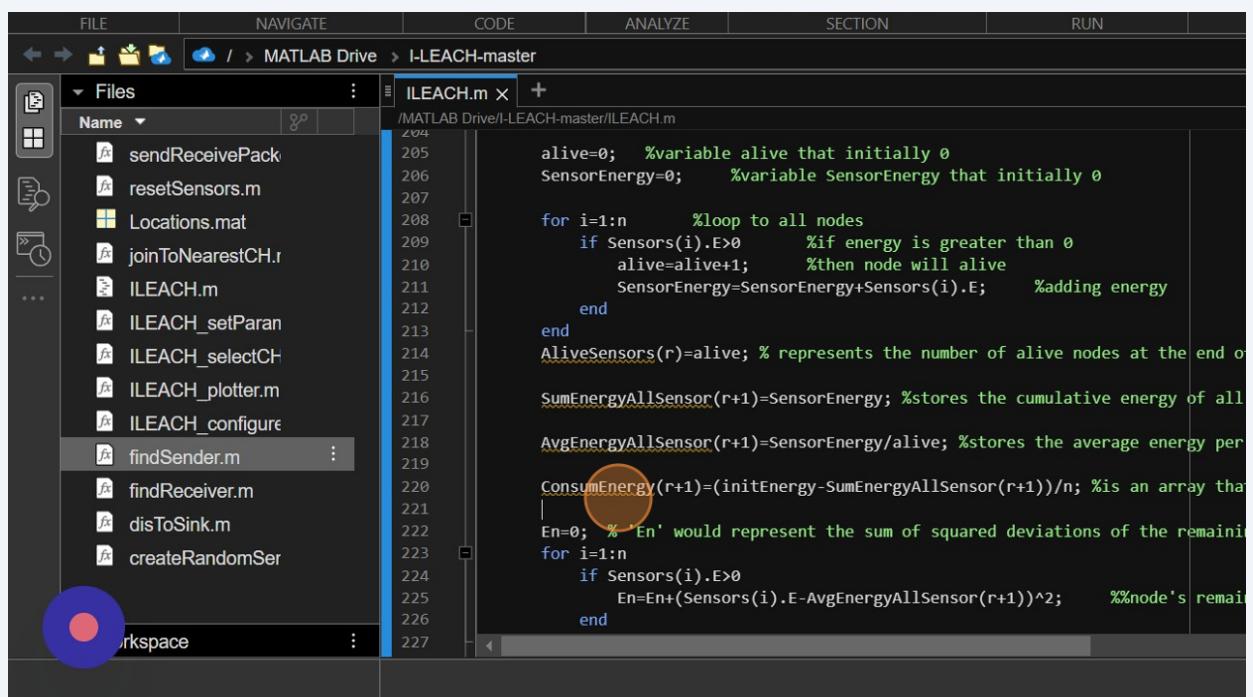
CLUSTERHS(r+1)=countCHs; %the number of cluster heads in each round

alive=0; %variable alive that initially 0
SensorEnergy=0; %variable SensorEnergy that initially 0

for i=1:n %loop to all nodes
    if Sensors(i).E>0 %if energy is greater than 0
        alive=alive+1; %then node will alive
        SensorEnergy=SensorEnergy+Sensors(i).E; %adding energy
    end
end
AliveSensors(r)=alive; % represents the number of alive nodes at the end of the round

SumEnergyAllSensor(r+1)=SensorEnergy; %stores the cumulative energy of all alive nodes
AvgEnergyAllSensor(r+1)=SensorEnergy/alive; %stores the average energy per alive node
ConsumEnergy(r+1)=(initEnergy-SumEnergyAllSensor(r+1))/n; %is an array that stores the consumed energy of each round
```

- AliveSensors(r): This array stores the number of alive nodes at the end of each round (denoted by r). The variable alive from the previous loop is used here.
- SumEnergyAllSensor(r+1): This array stores the cumulative energy of all nodes up to the current round (r+1). The SensorEnergy variable, which accumulates the energy of alive nodes, is used here.
- AvgEnergyAllSensor(r+1): This array stores the average energy per node up to the current round (r+1). It is calculated by dividing the cumulative energy (SensorEnergy) by the number of alive nodes (alive).
- ConsumEnergy(r+1): This array tracks the average energy consumption per node in each round. It is calculated as the difference between the initial energy (initEnergy) and the cumulative energy at the end of the round (SumEnergyAllSensor(r+1)), divided by the number of nodes (n).
- En: This variable is initialized to 0. However, it seems to be unused in the provided code segment.



The screenshot shows the MATLAB IDE interface with the following details:

- File Path:** MATLAB Drive\I-LEACH-master\ILEACH.m
- Code Editor:** The script file ILEACH.m is open, displaying MATLAB code. A red circle highlights the line: `ConsumEnergy(r+1)=(initEnergy-SumEnergyAllSensor(r+1))/n;`
- Code Content:**

```

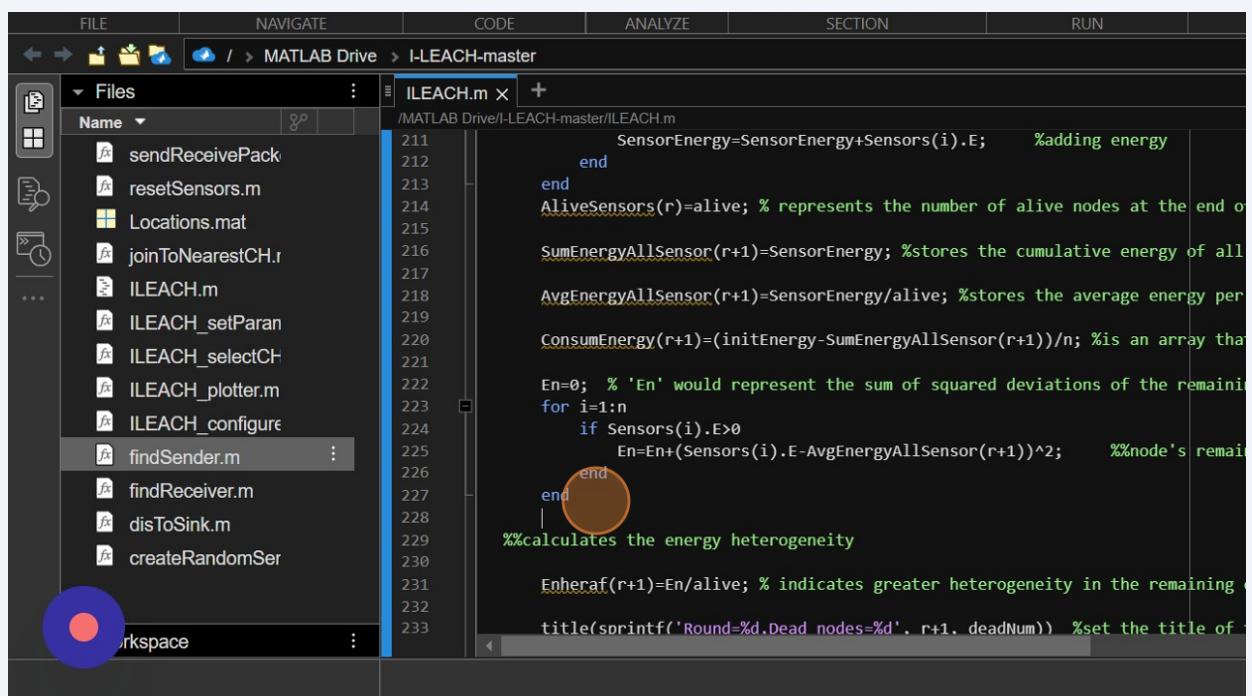
1 alive=0; %variable alive that initially 0
2 SensorEnergy=0; %variable SensorEnergy that initially 0
3
4 for i=1:n %loop to all nodes
5     if Sensors(i).E>0 %if energy is greater than 0
6         alive=alive+1; %then node will alive
7         SensorEnergy=SensorEnergy+Sensors(i).E; %adding energy
8     end
9 end
10 AliveSensors(r)=alive; % represents the number of alive nodes at the end of
11
12 SumEnergyAllSensor(r+1)=SensorEnergy; %stores the cumulative energy of all
13
14 AvgEnergyAllSensor(r+1)=SensorEnergy/alive; %stores the average energy per
15
16 ConsumEnergy(r+1)=(initEnergy-SumEnergyAllSensor(r+1))/n; %is an array that
17
18 En=0; % 'En' would represent the sum of squared deviations of the remainin
19 for i=1:n
20     if Sensors(i).E>0
21         En=En+(Sensors(i).E-AvgEnergyAllSensor(r+1))^2; %%node's remainin
22     end
23 end

```

85

- This loop iterates through all nodes (1 to n).
- The if Sensors(i).E > 0 condition checks if the energy of the current node (i) is greater than 0, indicating that the node is alive.
- The line En = En + (Sensors(i).E - AvgEnergyAllSensor(r+1))^2 accumulates the squared difference between the remaining energy of the current node (Sensors(i).E) and the average energy per node (AvgEnergyAllSensor(r+1)). This is done for all alive nodes in the network.

This operation is part of a calculation related to the variance of the remaining energy among nodes.



The screenshot shows the MATLAB IDE interface with the following details:

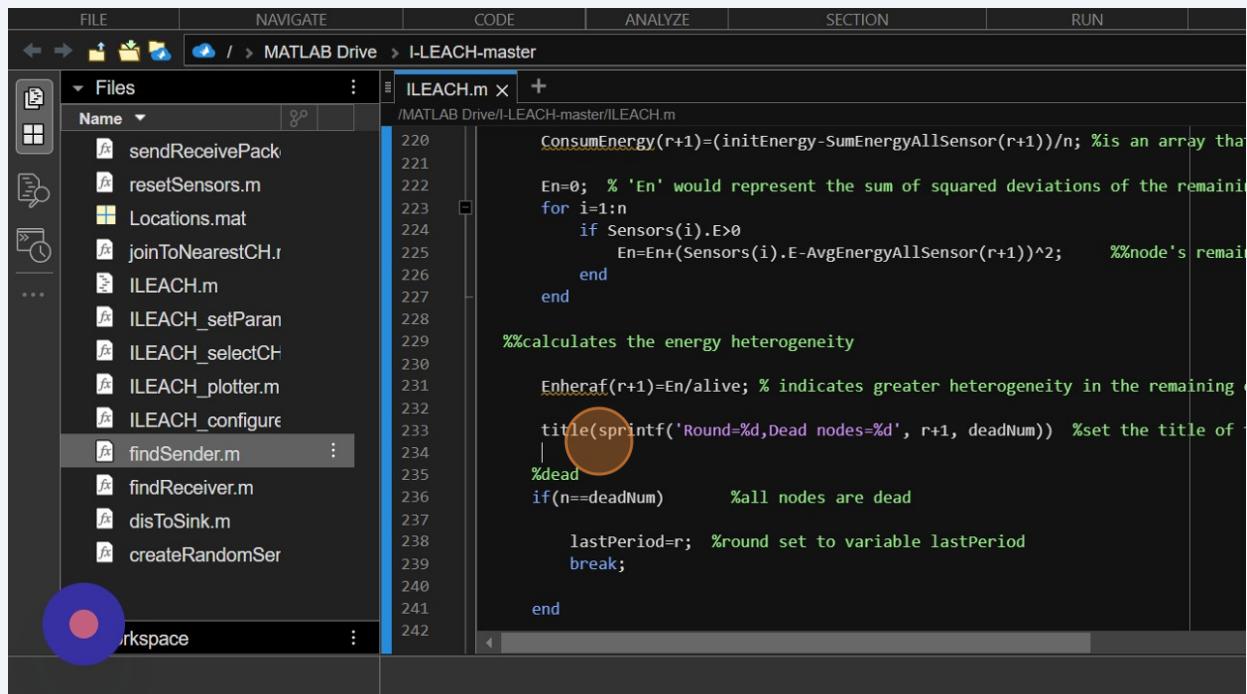
- File Explorer:** Shows a list of files in the 'I-LEACH-master' folder, including sendReceivePack, resetSensors.m, Locations.mat, joinToNearestCH.r, ILEACH.m, ILEACH_setParam, ILEACH_selectCH, ILEACH_plotter.m, ILEACH_configure, findSender.m, findReceiver.m, disToSink.m, and createRandomSer.
- Code Editor:** Displays the content of the ILEACH.m script. The highlighted line is: `En=En+(Sensors(i).E-AvgEnergyAllSensor(r+1))^2;`. This line is annotated with a red circle.
- Toolbars and Menus:** Standard MATLAB toolbars and menus like FILE, NAVIGATE, CODE, ANALYZE, SECTION, and RUN are visible at the top.

```
211 SensorEnergy=SensorEnergy+Sensors(i).E; %adding energy
212
213 end
214
215 AliveSensors(r)=alive; % represents the number of alive nodes at the end of
216 SumEnergyAllSensor(r+1)=SensorEnergy; %stores the cumulative energy of all
217 AvgEnergyAllSensor(r+1)=SensorEnergy/alive; %stores the average energy per
218 ConsumEnergy(r+1)=(initEnergy-SumEnergyAllSensor(r+1))/n; %is an array that
219
220 En=0; % 'En' would represent the sum of squared deviations of the remaining
221 for i=1:n
222     if Sensors(i).E>0
223         En=En+(Sensors(i).E-AvgEnergyAllSensor(r+1))^2; %%node's remaining
224     end
225 end
226 %%calculates the energy heterogeneity
227 Eheteraf(r+1)=En/alive; % indicates greater heterogeneity in the remaining
228
229 title(sprintf('Round=%d.Dead nodes=%d'. r+1, deadNum)) %set the title of
230
231
232
233
```

86

The line `Enheteraf(r+1) = En/alive;` calculates the energy heterogeneity at the end of the current round ($r+1$). Energy heterogeneity is a measure of how non-uniformly energy is distributed among the nodes in the network. It is calculated by dividing the sum of squared differences between each node's remaining energy and the average energy by the number of alive nodes (alive).

The `title(sprintf('Round=%d,Dead nodes=%d', r+1, deadNum))` sets the title of the plot. It displays information about the current round number ($r+1$) and the number of dead nodes (deadNum). The sprintf function is used to format the text that will be displayed in the title.



The screenshot shows the MATLAB IDE interface with the following details:

- File Explorer:** Shows a list of files in the 'ILEACH-master' folder, including `sendReceivePack.m`, `resetSensors.m`, `Locations.mat`, `joinToNearestCH.r`, `ILEACH.m`, `ILEACH_setParam.m`, `ILEACH_selectCH.m`, `ILEACH_plotter.m`, `ILEACH_configure.m`, `findSender.m`, `findReceiver.m`, `disToSink.m`, and `createRandomSer.m`.
- Current File:** The file `ILEACH.m` is open in the editor.
- Code View:**

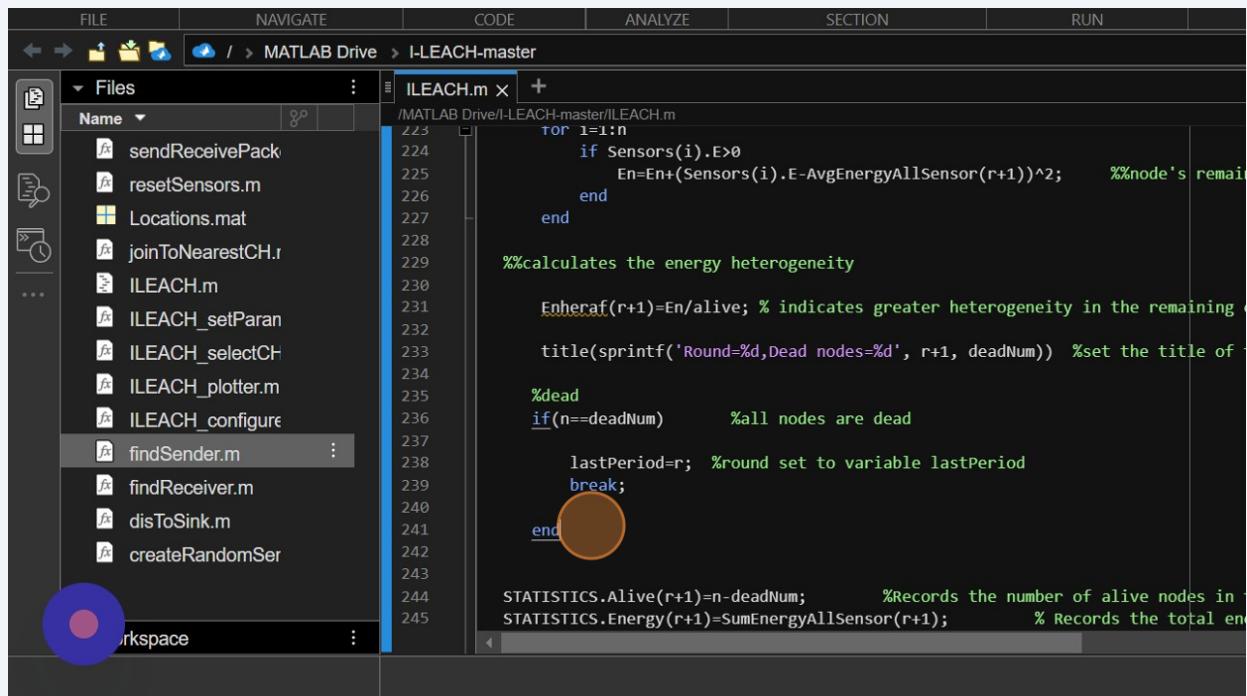
```

220 ConsumEnergy(r+1)=(initEnergy-SumEnergyAllSensor(r+1))/n; %is an array that
221
222 En=0; % 'En' would represent the sum of squared deviations of the remaining
223 for i=1:n
224     if Sensors(i).E>0
225         En=En+(Sensors(i).E-AvgEnergyAllSensor(r+1))^2;      %%node's remain
226     end
227 end
228
229 %%calculates the energy heterogeneity
230
231 Enheteraf(r+1)=En/alive; % indicates greater heterogeneity in the remaining e
232
233 title(sprintf('Round=%d,Dead nodes=%d', r+1, deadNum)) %set the title of t
234
235 %dead
236 if(n==deadNum)      %all nodes are dead
237
238     lastPeriod=r; %round set to variable lastPeriod
239     break;
240
241 end

```
- Annotations:** A red circle highlights the line `Enheteraf(r+1)=En/alive;`

87

The condition if ($n == \text{deadNum}$) checks whether the number of dead nodes is equal to the total number of nodes (n). If this condition is true, it implies that all nodes in the network are dead, and the simulation is considered completed. The variable lastPeriod is then set to the current round (r), and the break statement is used to exit the loop, ending the simulation.

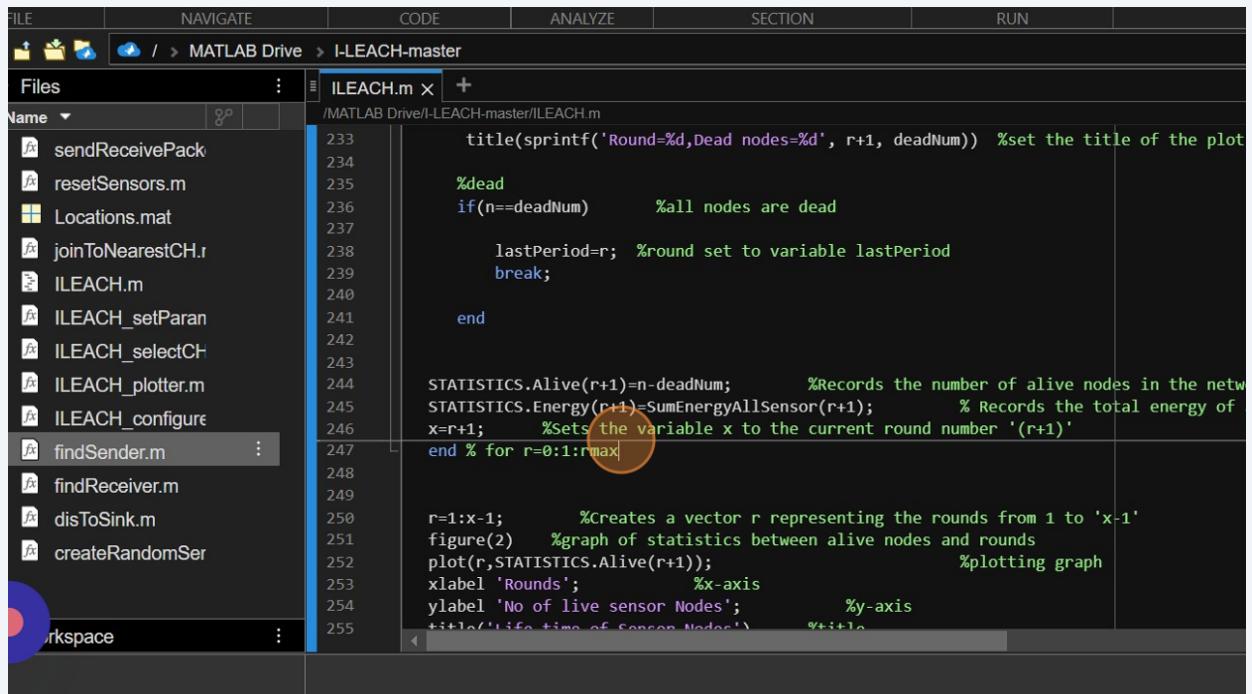


```
FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master
ILEACH.m x +
/MATLAB Drive/I-LEACH-master/ILEACH.m
223 for i=1:n
224     if Sensors(i).E>0
225         En=En+(Sensors(i).E-AvgEnergyAllSensor(r+1))^2;    %%node's remain
226     end
227 end
228
229 %%calculates the energy heterogeneity
230
231 Etheraf(r+1)=En/alive; % indicates greater heterogeneity in the remaining
232
233 title(sprintf('Round=%d,Dead nodes=%d', r+1, deadNum)) %set the title of
234
235 %dead
236 if(n==deadNum)      %all nodes are dead
237
238     lastPeriod=r; %round set to variable lastPeriod
239     break;
240
241 end
242
243
244 STATISTICS.Alive(r+1)=n-deadNum;           %Records the number of alive nodes in
245 STATISTICS.Energy(r+1)=SumEnergyAllSensor(r+1);   % Records the total en
```

- STATISTICS.Alive(r+1) records the number of alive nodes in the network at the end of the current round (r+1). It subtracts the number of dead nodes (deadNum) from the total number of nodes (n).

- STATISTICS.Energy(r+1) records the total energy of all sensors at the end of the current round (r+1). It uses the cumulative energy recorded in SumEnergyAllSensor(r+1).

These statistics provide insights into the network's performance and energy consumption throughout the simulation.



```

FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive > I-LEACH-master
ILEACH.m X +
/MATLAB Drive/I-LEACH-master/ILEACH.m
Name
sendReceivePack
resetSensors.m
Locations.mat
joinToNearestCH.r
ILEACH.m
ILEACH_setParan
ILEACH_selectCH
ILEACH_plotter.m
ILEACH_configure
findSender.m ...
findReceiver.m
disToSink.m
createRandomSer
Workspace ...

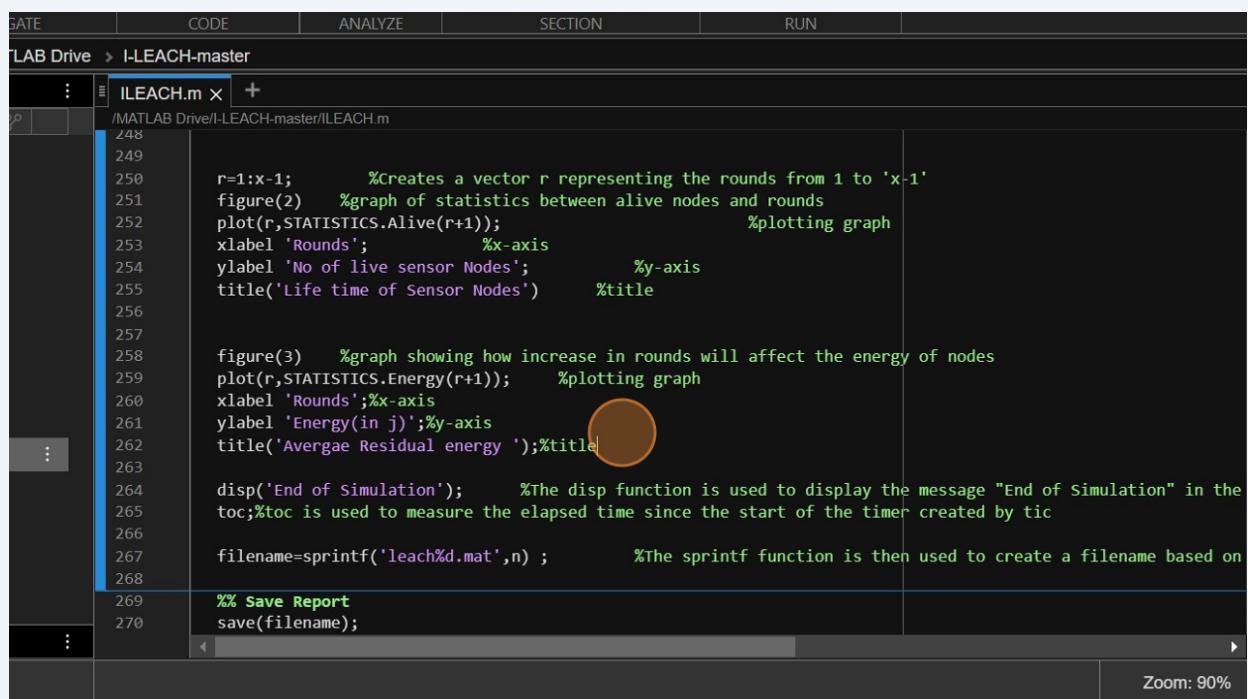
233 title(sprintf('Round=%d,Dead nodes=%d', r+1, deadNum)) %set the title of the plot
234
235 %dead
236 if(n==deadNum)      %all nodes are dead
237
238     lastPeriod=r; %round set to variable lastPeriod
239     break;
240
241 end
242
243
244 STATISTICS.Alive(r+1)=n-deadNum;           %Records the number of alive nodes in the netw
245 STATISTICS.Energy(r+1)=SumEnergyAllSensor(r+1);    % Records the total energy of
246 x=r+1;          %Sets the variable x to the current round number '(r+1)'
247 end % for r=0:1:rmax
248
249
250 r=1:x-1;        %Creates a vector r representing the rounds from 1 to 'x-1'
251 figure(2)       %graph of statistics between alive nodes and rounds
252 plot(r,STATISTICS.Alive(r+1));               %plotting graph
253 xlabel 'Rounds';                          %x-axis
254 ylabel 'No of live sensor Nodes';          %y-axis
255 +i+1+'life time of Sensor Nodes')        %+i+1

```

89

The last part of the code generates a plot and saves simulation results to a MAT file. Here's a breakdown:

- `figure(3)`: Creates a new figure with a handle of 3. This is necessary if you want to plot multiple graphs in the same script.
- `plot(r, STATISTICS.Energy(r+1))`: Plots the average residual energy against the number of rounds (`r`). The `STATISTICS.Energy(r+1)` array holds the total energy of all sensors at the end of each round. The x-axis is labeled "Rounds," the y-axis is labeled "Energy (in J)," and the title of the plot is set to "Average Residual Energy."
- `xlabel 'Rounds'`: Sets the label for the x-axis.
- `ylabel 'Energy(in J)'`: Sets the label for the y-axis.
- `title('Average Residual Energy')`: Sets the title of the plot.



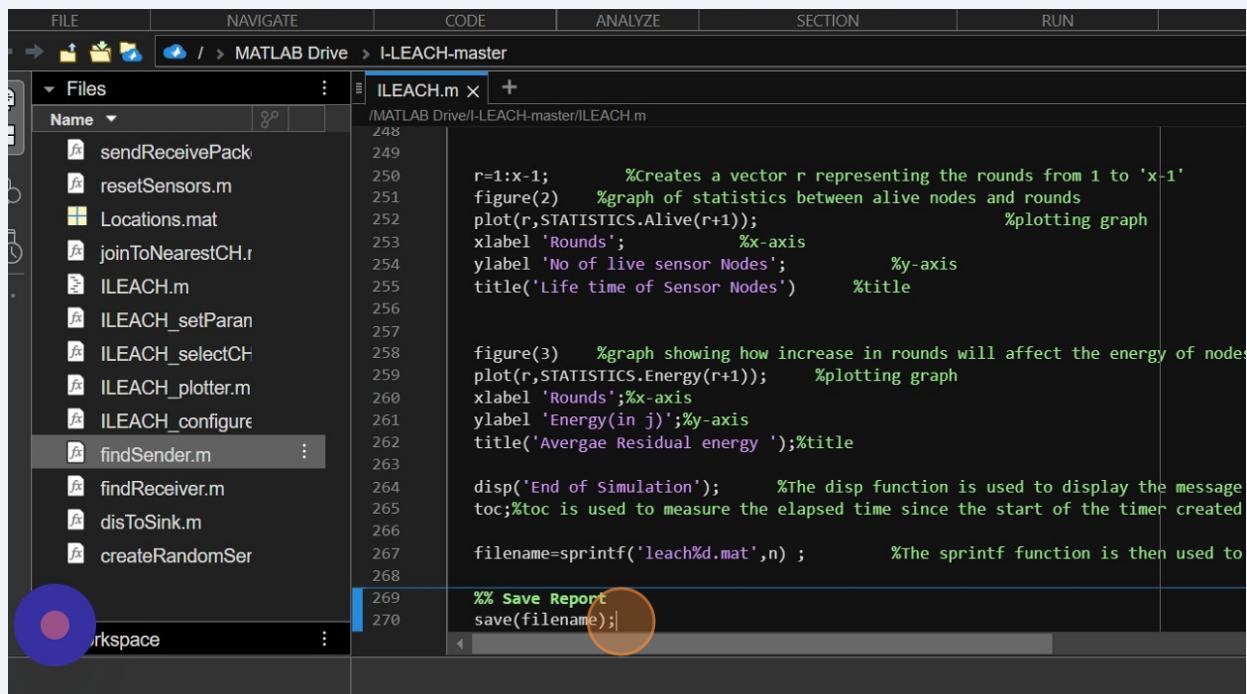
```
GATE      CODE      ANALYZE      SECTION      RUN
LAB Drive > I-LEACH-master
:   ILEACH.m X +
/MATLAB Drive/I-LEACH-master/ILEACH.m
248
249
250 r=1:x-1;           %Creates a vector r representing the rounds from 1 to 'x-1'
251 figure(2)    %graph of statistics between alive nodes and rounds
252 plot(r,STATISTICS.Alive(r+1));          %plotting graph
253 xlabel 'Rounds';           %x-axis
254 ylabel 'No of live sensor Nodes';        %y-axis
255 title('Life time of Sensor Nodes')      %title
256
257
258 figure(3)    %graph showing how increase in rounds will affect the energy of nodes
259 plot(r,STATISTICS.Energy(r+1));          %plotting graph
260 xlabel 'Rounds';           %x-axis
261 ylabel 'Energy(in j)';        %y-axis
262 title('Avergae Residual energy ')%;title
263
264 disp('End of Simulation');           %The disp function is used to display the message "End of Simulation" in the
265 toc;%toc is used to measure the elapsed time since the start of the timer created by tic
266
267 filename=sprintf('leach%d.mat',n);       %The sprintf function is then used to create a filename based on
268
269 %% Save Report
270 save(filename);

```

Zoom: 90%

90

- `disp('End of Simulation');`: Displays the message "End of Simulation" in the command window.
- `toc`: Outputs the elapsed time since the timer was started by `tic`.
- `filename = sprintf('leach%d.mat', n)`: Generates a filename for saving the simulation results as a MAT file. The `sprintf` function is used to create a filename based on the number of nodes (`n`).
- `save(filename)`: Saves the simulation results in a MAT file with the generated filename.



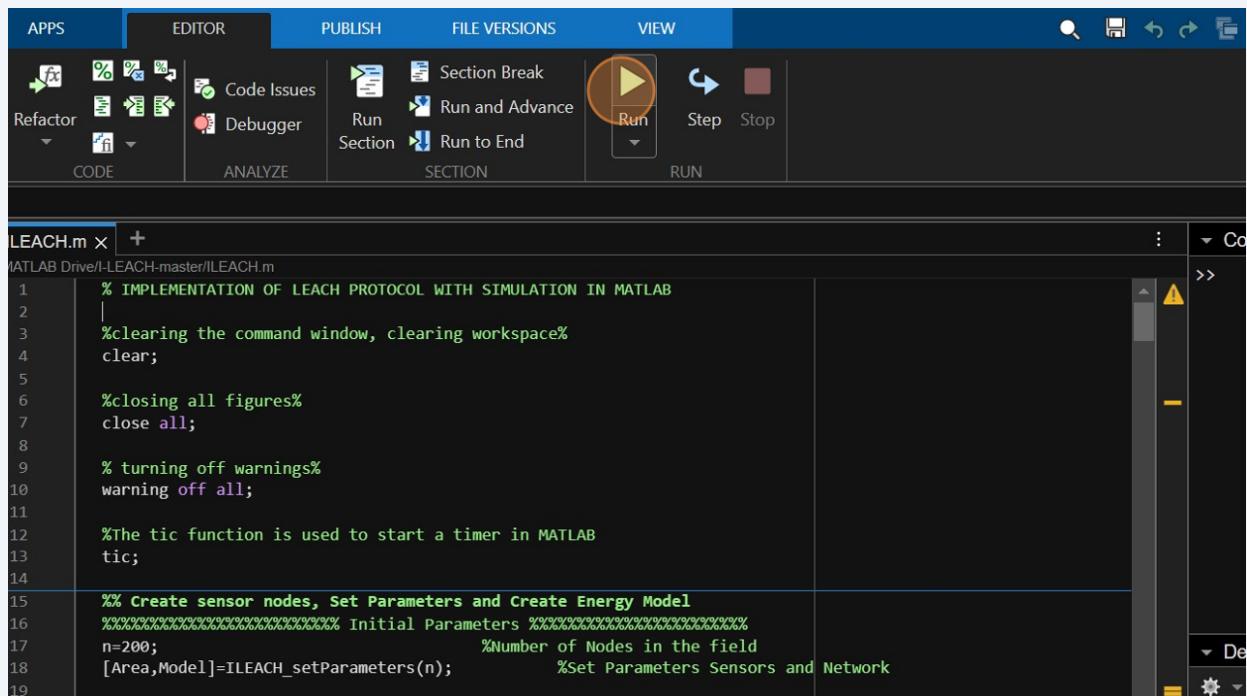
The screenshot shows the MATLAB IDE interface with the following details:

- Toolbar:** FILE, NAVIGATE, CODE, ANALYZE, SECTION, RUN.
- Path:** / MATLAB Drive > I-LEACH-master
- File Explorer:** Shows files in the I-LEACH-master folder, including sendReceivePack, resetSensors.m, Locations.mat, joinToNearestCH.r, ILEACH.m, ILEACH_setParan, ILEACH_selectCH, ILEACH_plotter.m, ILEACH_configure, findSender.m (highlighted), findReceiver.m, disToSink.m, createRandomSer.
- Workspace:** Shows variables like r, STATISTICS, Energy, and alive.
- I-LEACH.m Editor:** The code is as follows:

```
% MATLAB Drive/I-LEACH-master/ILEACH.m
248
249
250 r=1:x-1; %Creates a vector r representing the rounds from 1 to 'x-1'
251 figure(2) %graph of statistics between alive nodes and rounds
252 plot(r,STATISTICS.Alive(r+1)); %plotting graph
253 xlabel 'Rounds'; %x-axis
254 ylabel 'No of live sensor Nodes'; %y-axis
255 title('Life time of Sensor Nodes') %title
256
257
258 figure(3) %graph showing how increase in rounds will affect the energy of nodes
259 plot(r,STATISTICS.Energy(r+1)); %plotting graph
260 xlabel 'Rounds';%x-axis
261 ylabel 'Energy(in j)';%y-axis
262 title('Avergae Residual energy');%title
263
264 disp('End of Simulation'); %The disp function is used to display the message
265 toc;%toc is used to measure the elapsed time since the start of the timer created
266
267 filename=sprintf('leach%d.mat',n); %The sprintf function is then used to
268
269 %% Save Report
270 save(filename);
```

LEACH PROTOCOL OUTPUT IN MATLAB

- 1 Click this icon. to run the simulation



The screenshot shows the MATLAB Editor window with the file 'LEACH.m' open. The code implements the LEACH protocol with simulation in MATLAB. The toolbar above the editor has a 'RUN' section with a 'Run' button highlighted by a red circle. The code itself includes comments for clearing the command window, workspace, figures, and warnings, and initializes parameters for sensor nodes and energy model.

```
% IMPLEMENTATION OF LEACH PROTOCOL WITH SIMULATION IN MATLAB
%
%clearing the command window, clearing workspace%
clear;

%closing all figures%
close all;

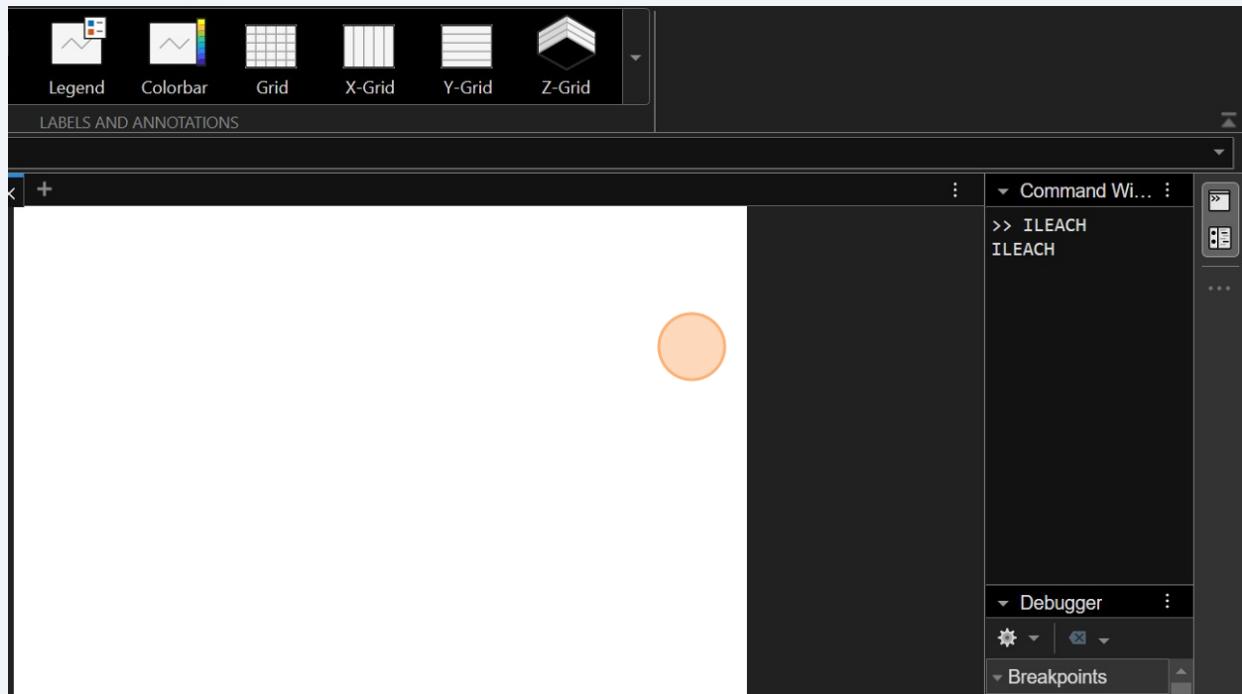
% turning off warnings%
warning off all;

%The tic function is used to start a timer in MATLAB
tic;

%% Create sensor nodes, Set Parameters and Create Energy Model
%%%%%%%%%%%%% Initial Parameters %%%%%%
n=200; %Number of Nodes in the field
[Area,Model]=ILEACH_setParameters(n); %Set Parameters Sensors and Network
```

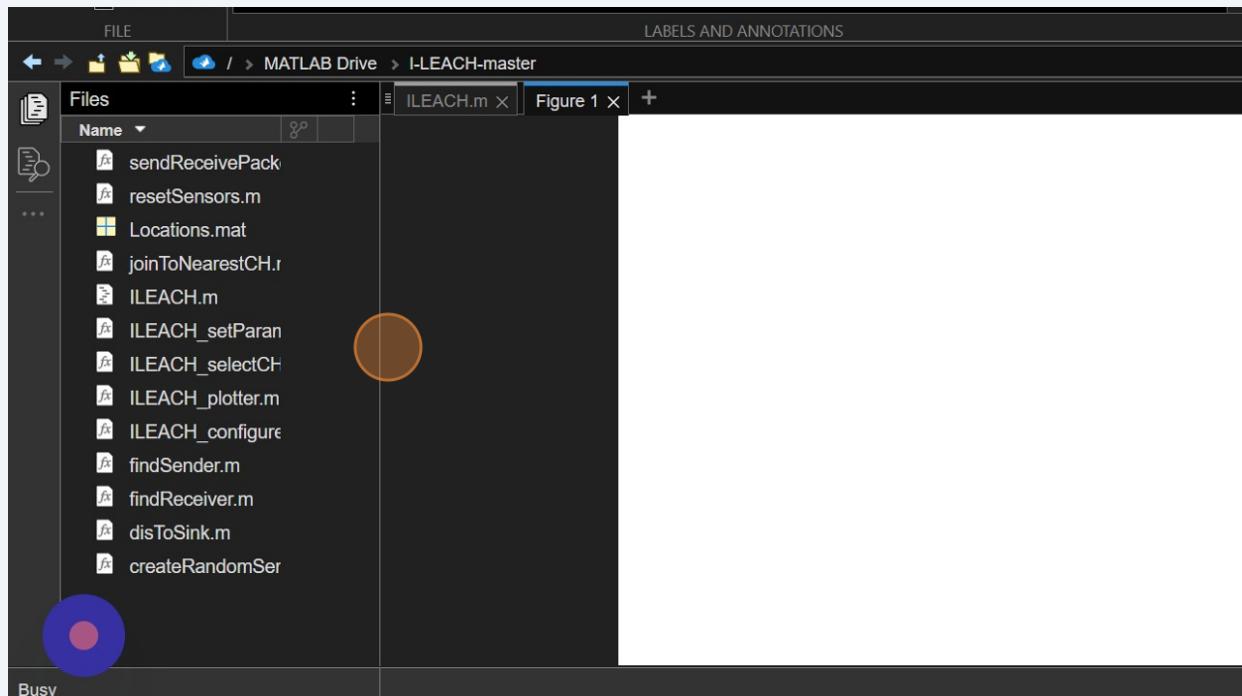
2

The protocol simulation starts It can also be run from command prompt by typing name of main file ILEACH



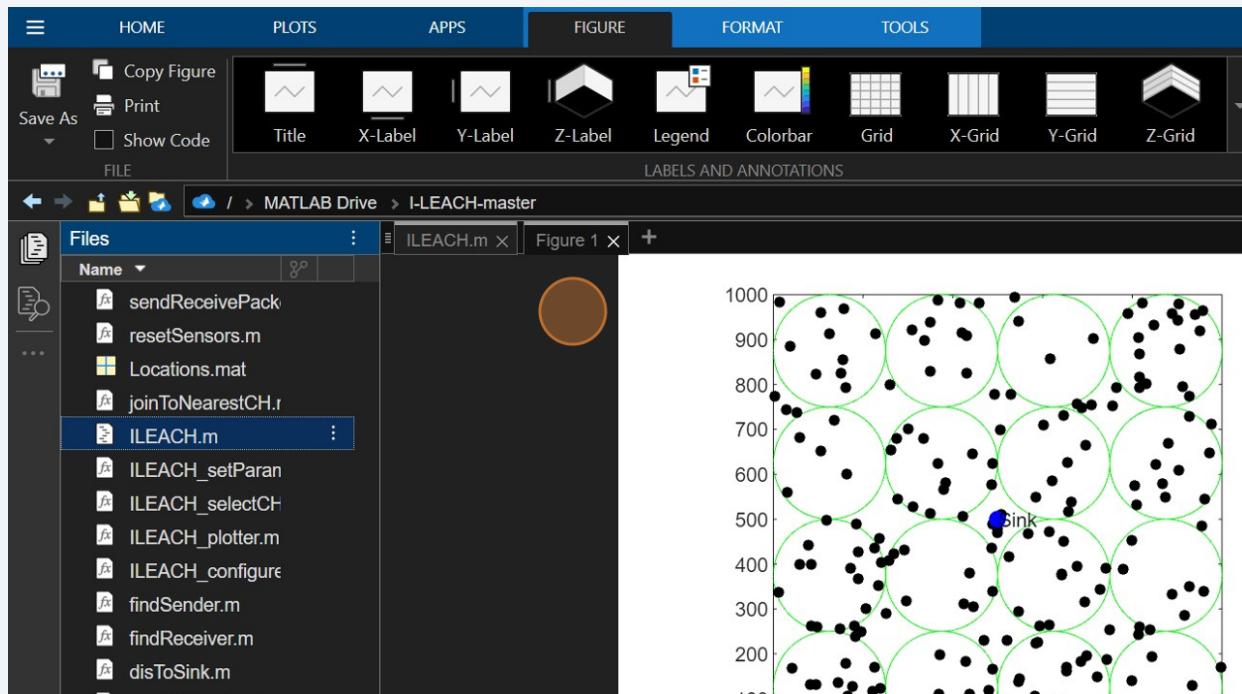
3

A Figure will be opened



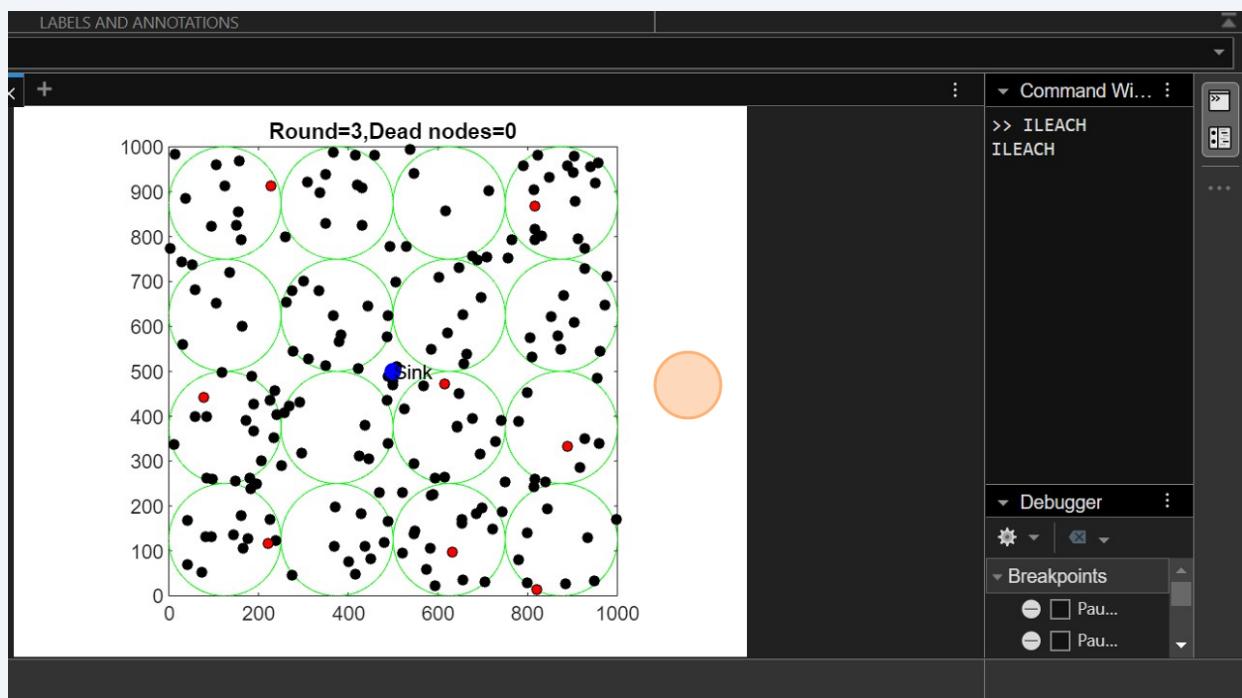
4

And Simulation is started, The green nodes represents the clusters black nodes are member nodes and cluster heads are represented by red color and the sink is displayed with blue color



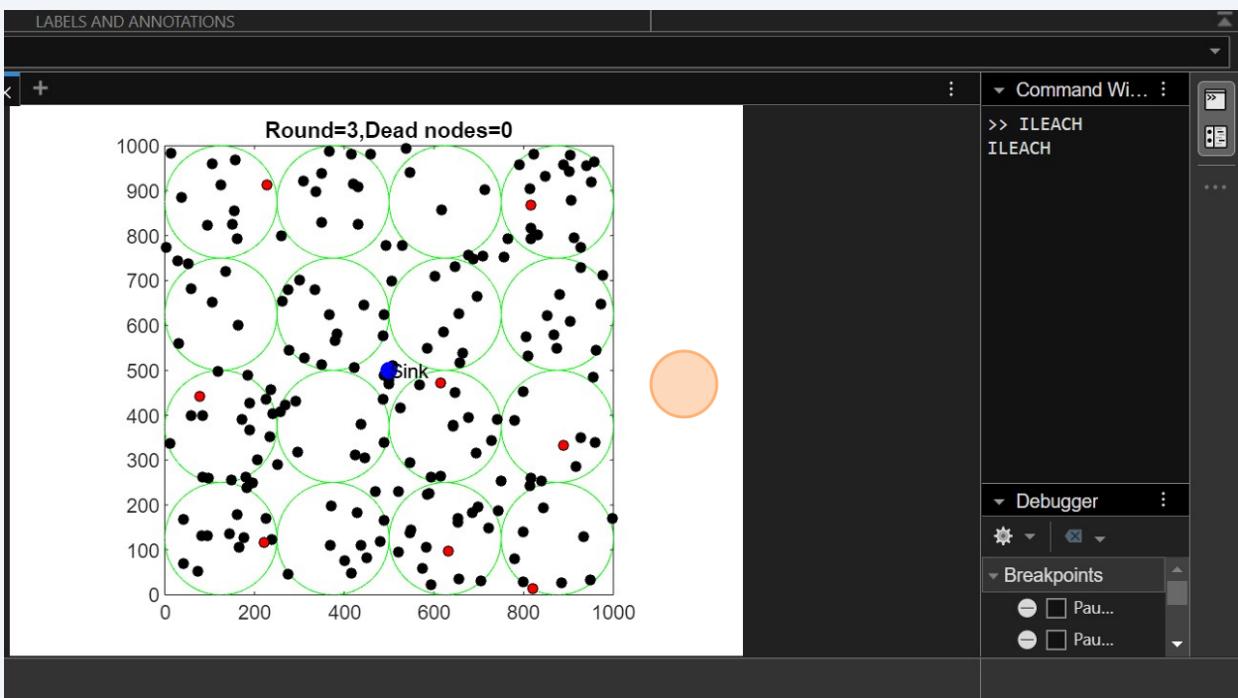
5

As LEACH works in Rounds , so in each round each cluster make its new cluster head to conserve the energy for longer time



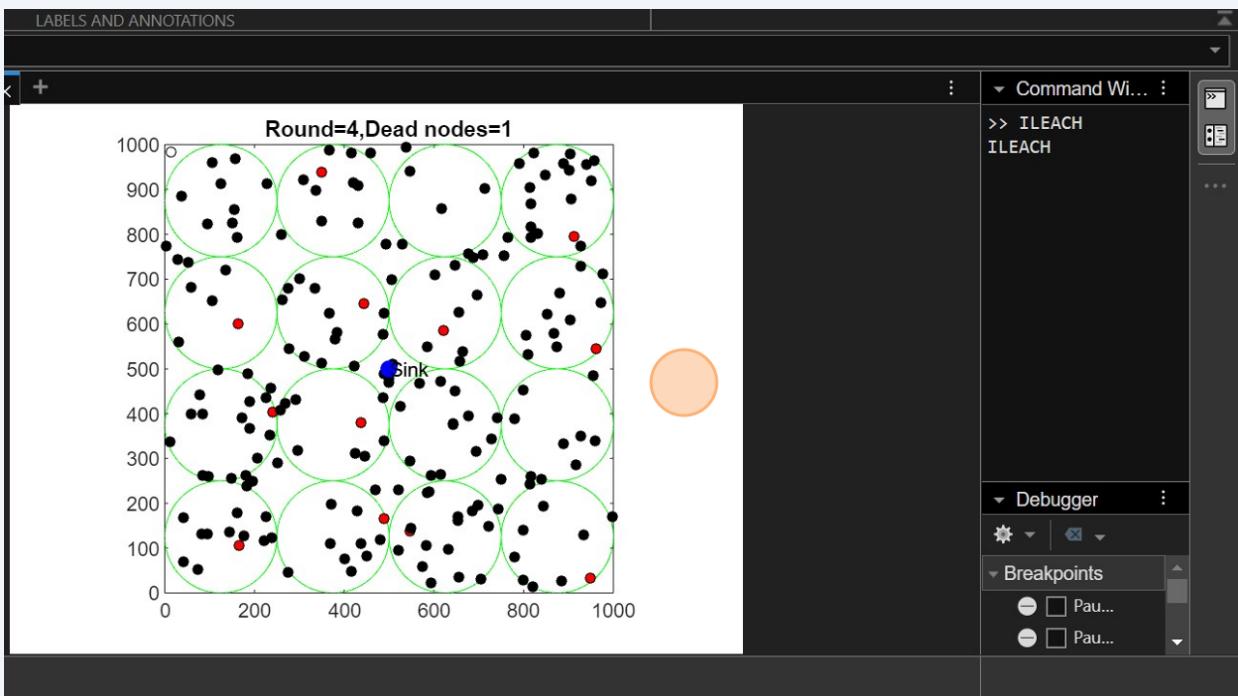
6

Nodes cannot send the data to one another only the members nodes can send data packets to cluster head and then cluster head send the packets to sink node



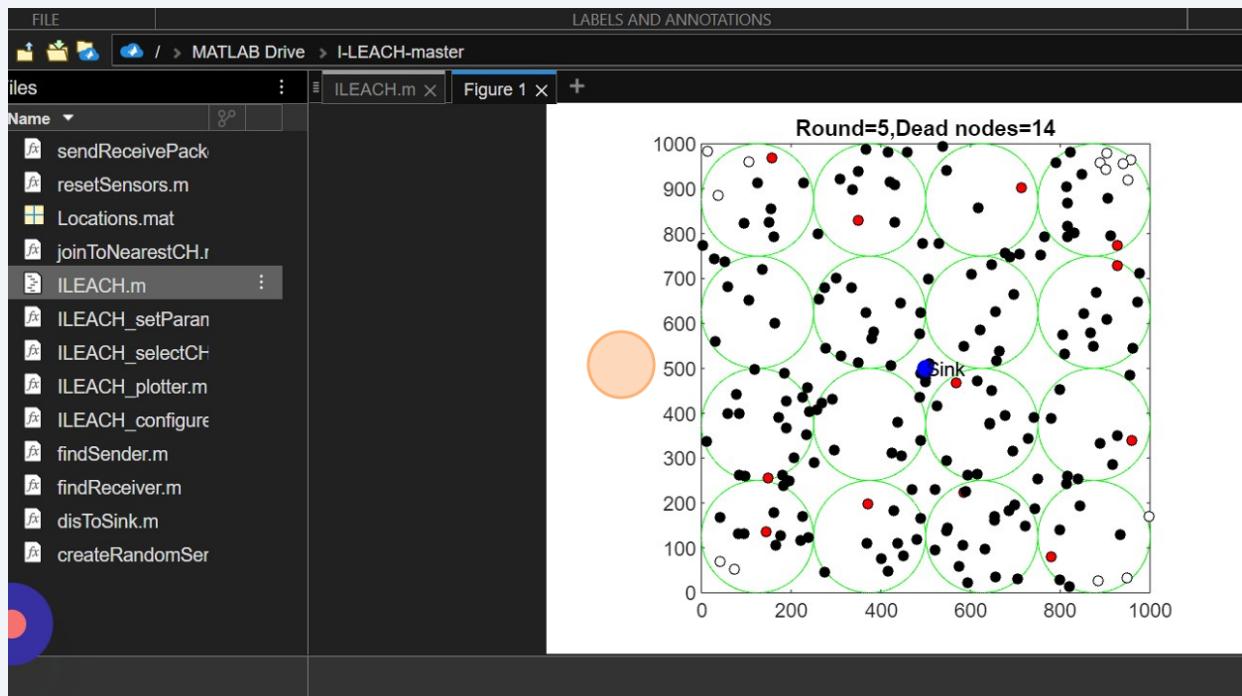
7

Each node has time slot to send data packets to Cluster Head



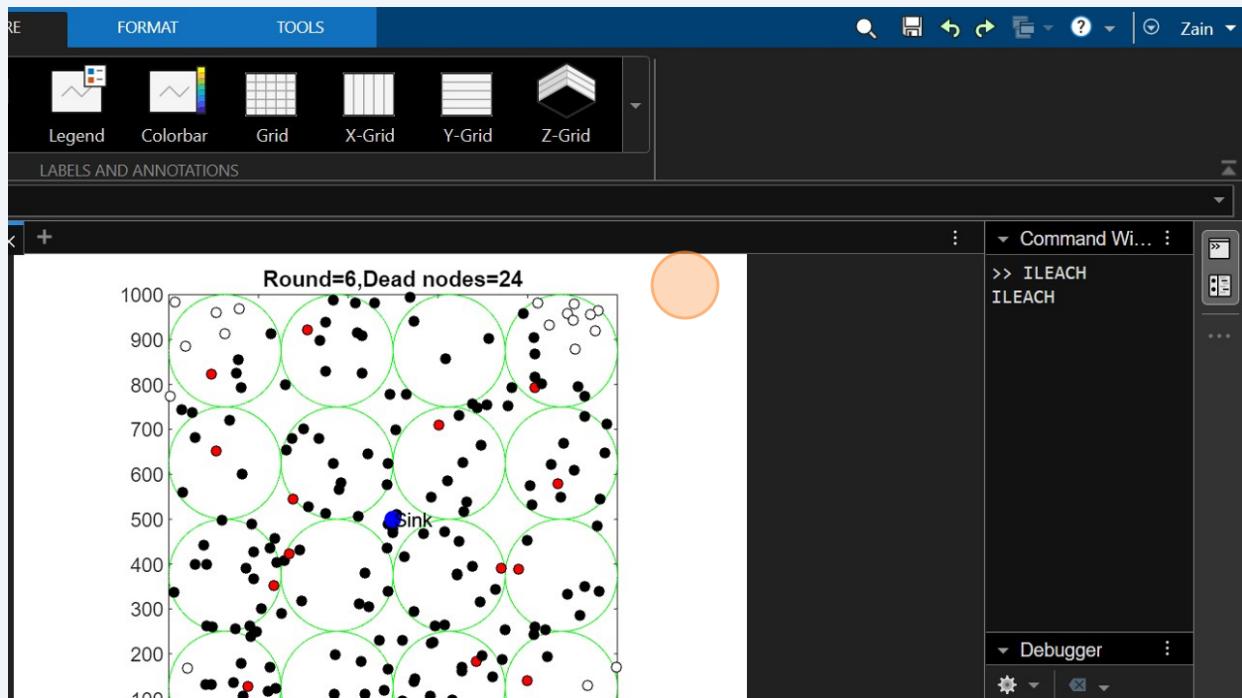
8

As soon as round changes the energy of member nodes become less and a time will come when the nodes will be dead

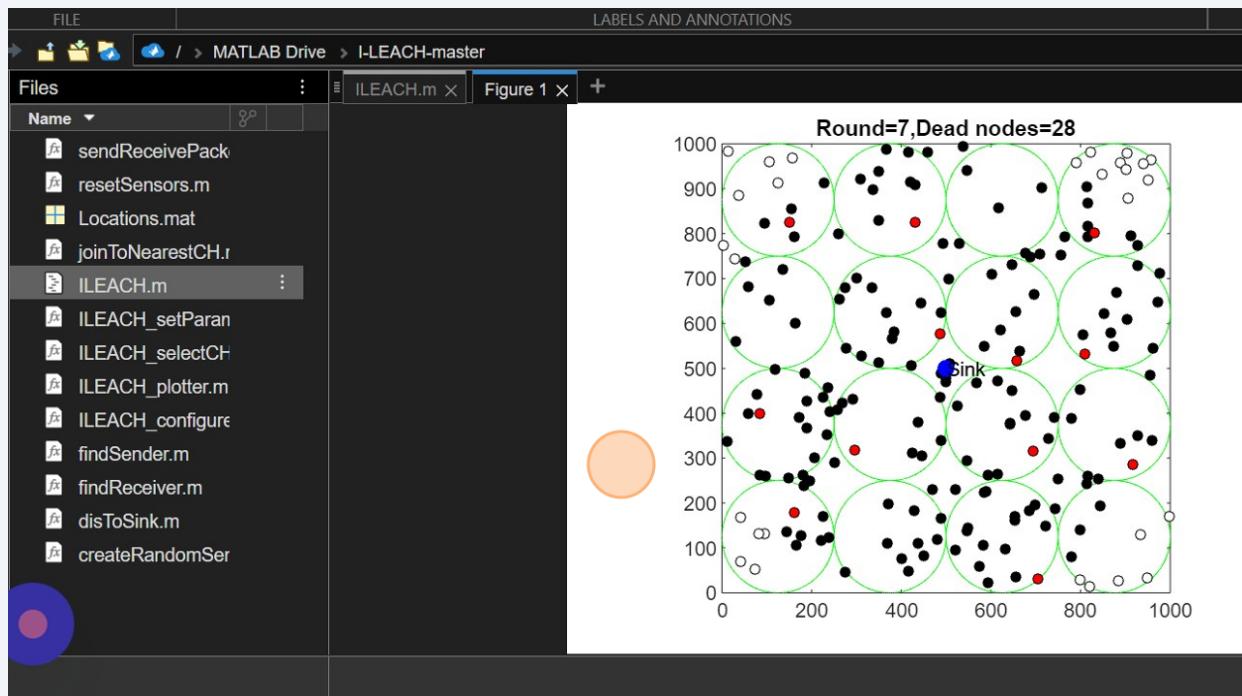


9

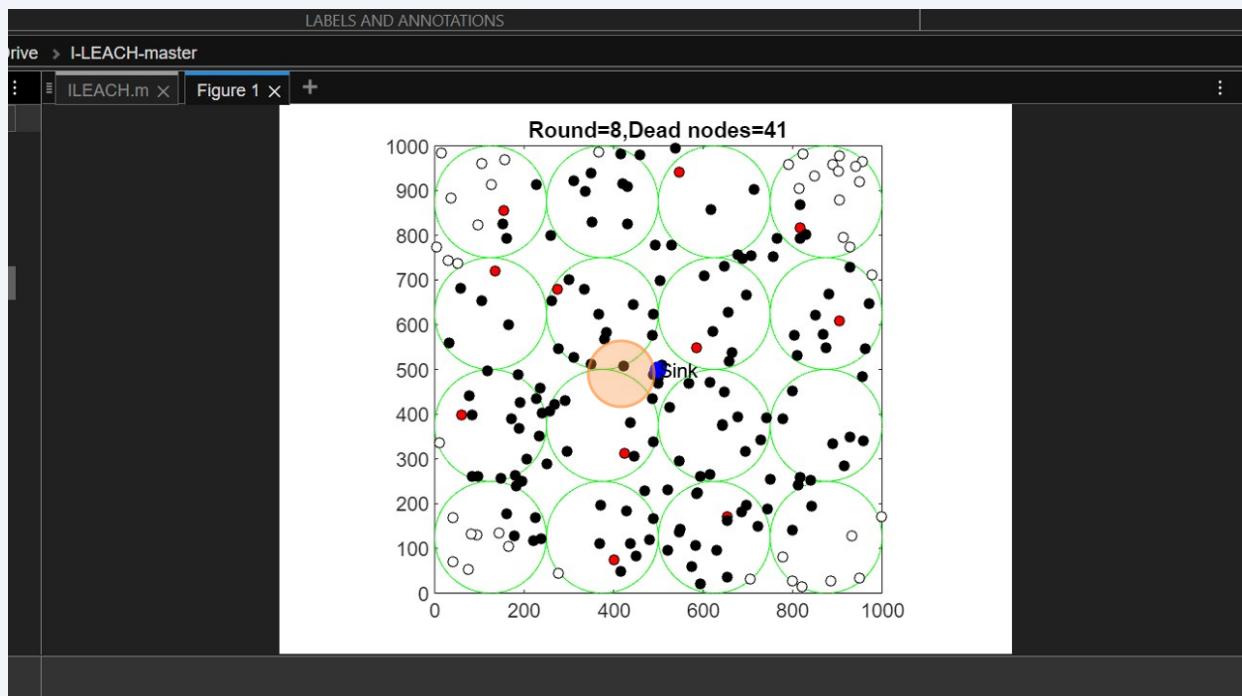
Number of dead nodes and Rounds can be observed from graph



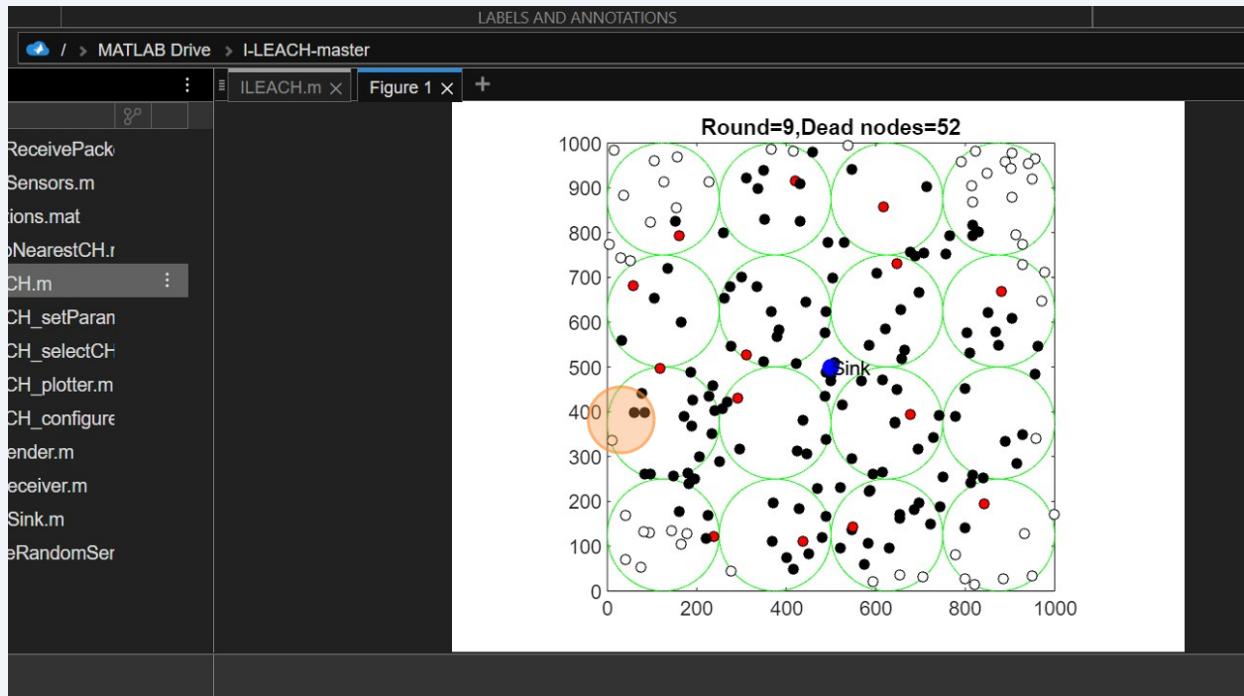
10 Dead nodes will be empty circle of white filled circles



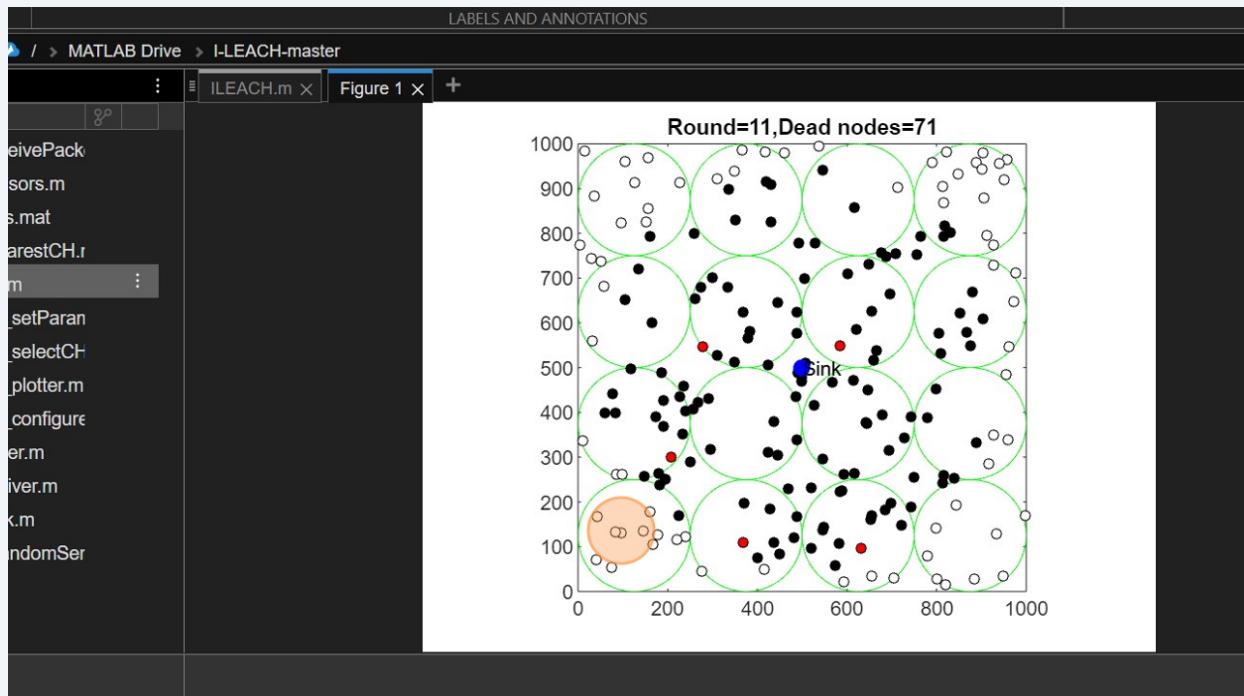
11 This data transmission will keep on going for the number of rounds that was fixed in the code



12 Cluster heads are selected on the bases of their energy level

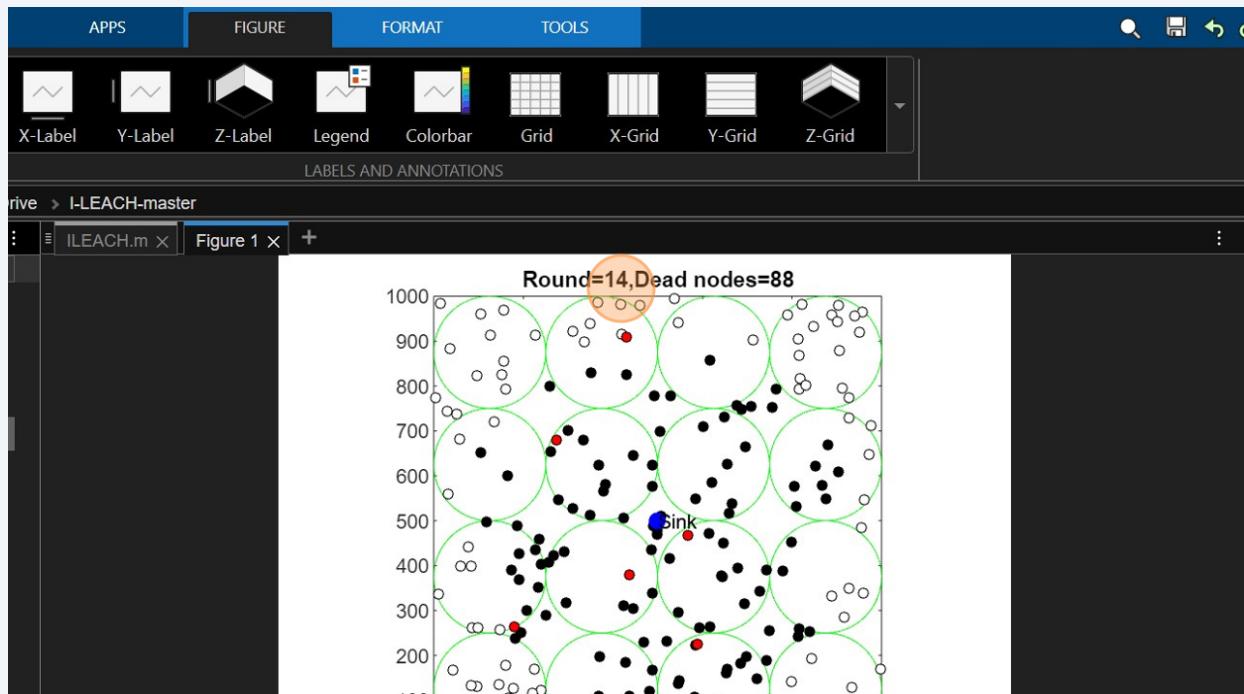


13 In Each round there is a new cluster head in order to conserve energy



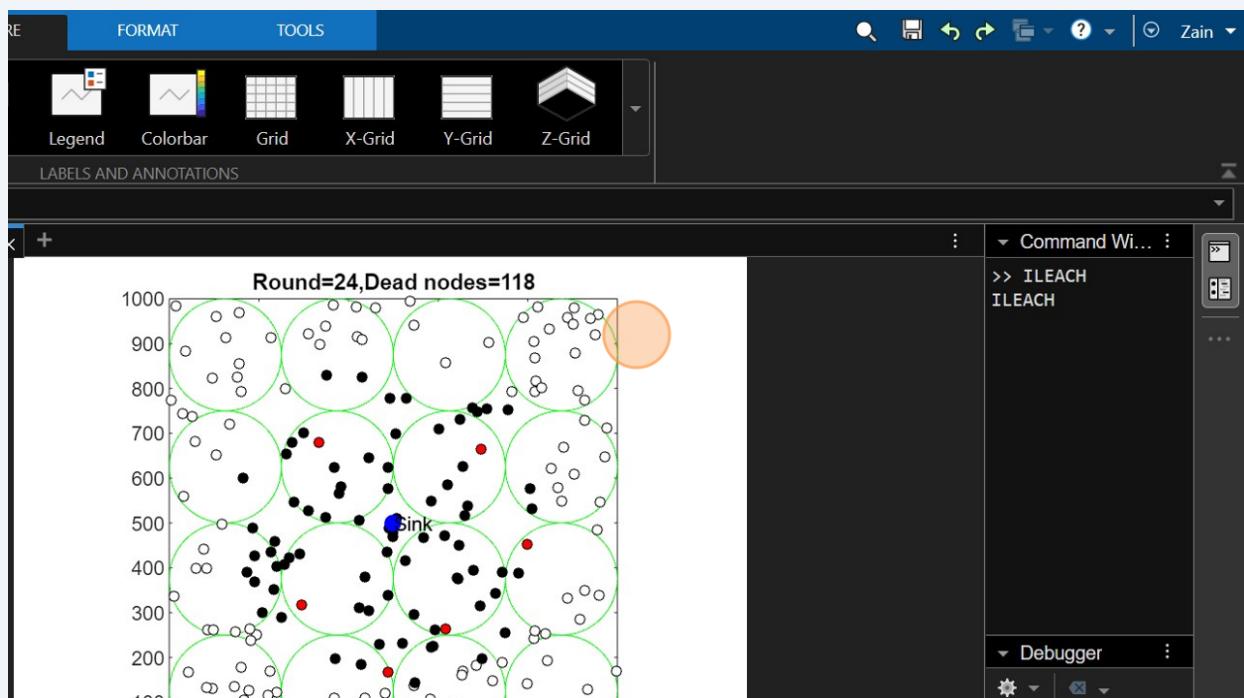
14

Similarly the member node itself do no send any data packet to sink as if each of the nodes sends on its own then sooo much energy will be required and lose so in order to conserve energy the member nodes sends tha data packet to CH and all CHs sends it to sink

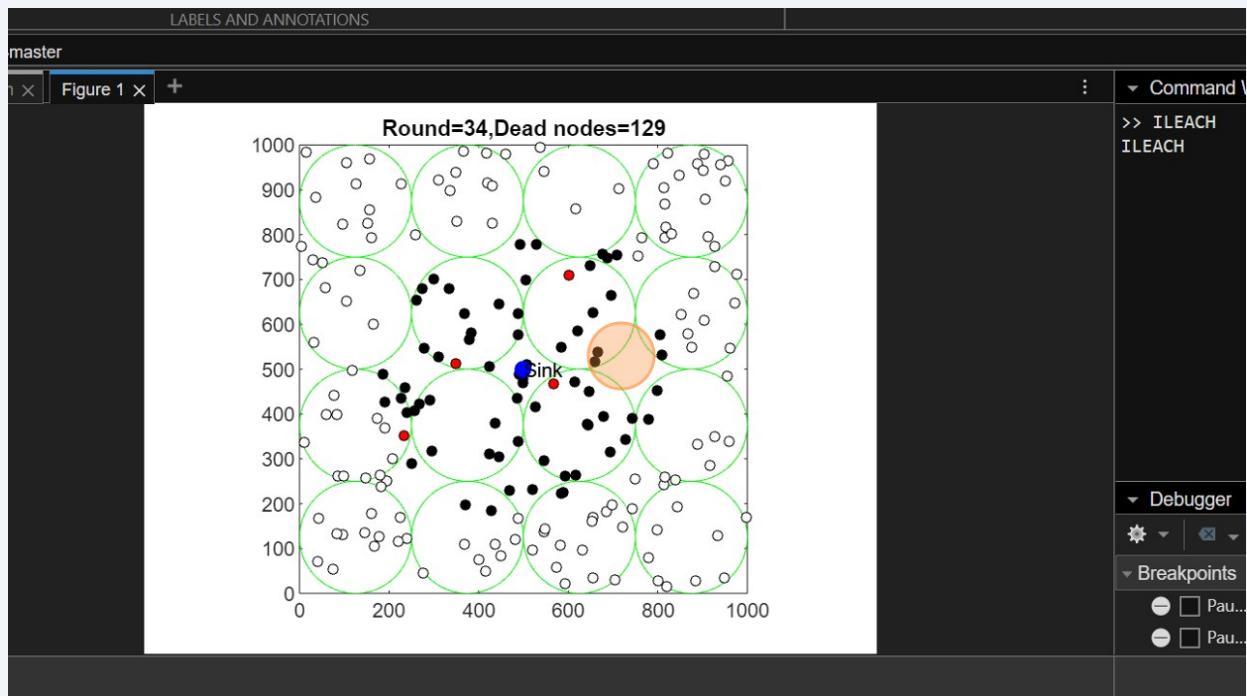


15

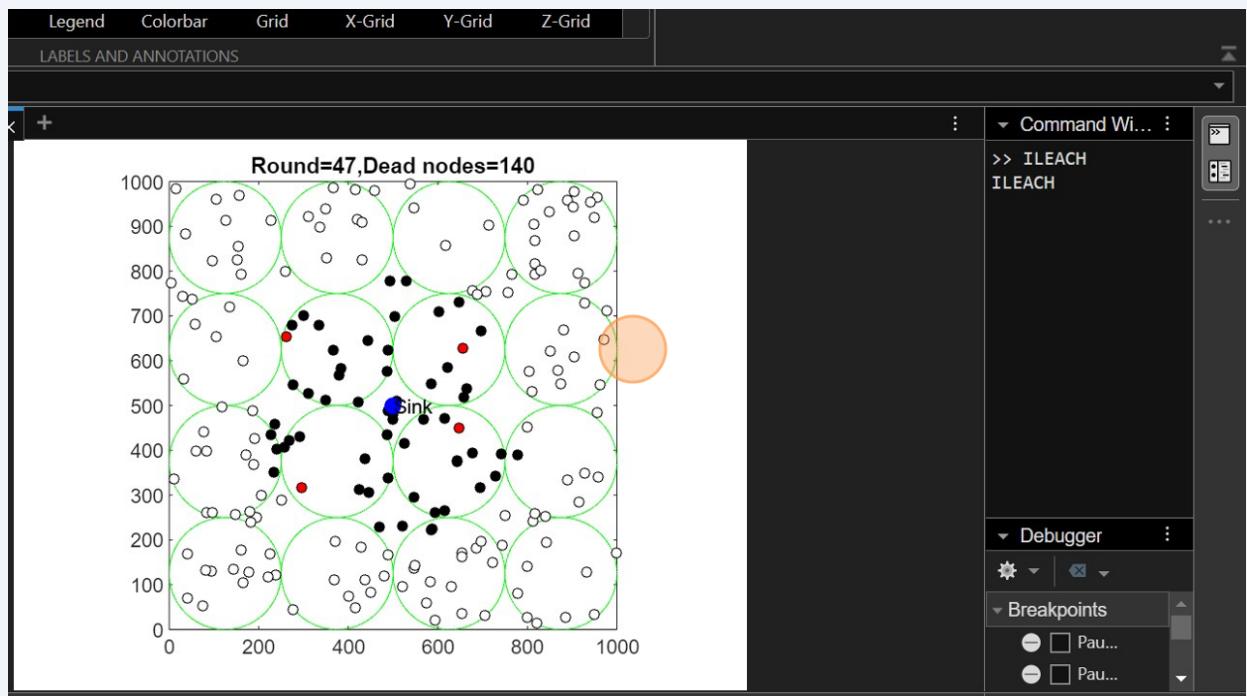
Here are some sample screenshots of different rounds



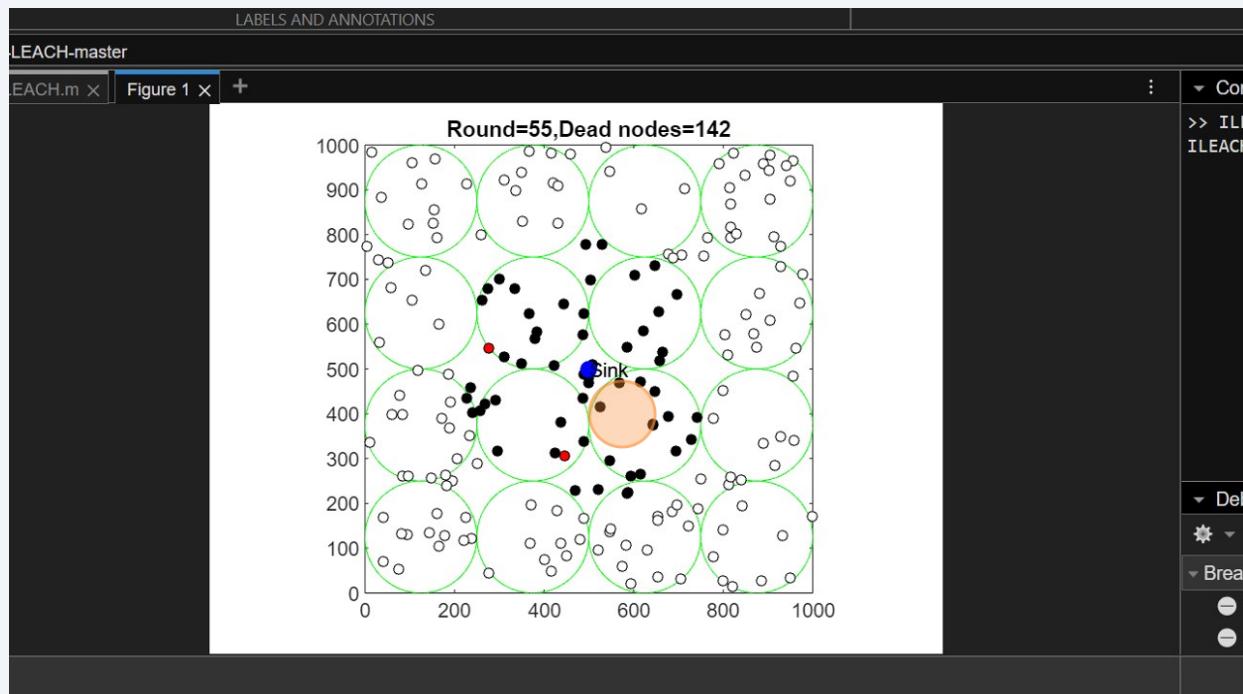
16 Sample 1



17 Sample 2



18 Sample 3



19 Click here.

