➢ CODE Introduction:

The provided MATLAB code implements a simulation of a wireless sensor network using the LEACH (Low-Energy Adaptive Clustering Hierarchy) protocol. LEACH is a hierarchical clustering protocol designed to prolong the network lifetime by reducing energy consumption, particularly in large-scale wireless sensor networks. The simulation model considers a two-dimensional area where sensor nodes are randomly deployed. The nodes operate in rounds, and each round involves the selection of cluster heads and data transmission.

Key Elements of the Code:

- ***Initialization:***

The simulation area is defined with specified dimensions (xm and ym).
Parameters such as the number of nodes (N), probability of selecting a node as a cluster head (p), initial energy of nodes (Eo), and various energy consumption constants are set.

- ***Node Deployment:***

Nodes are initialized with random positions, and collision avoidance is implemented to ensure that nodes are deployed without overlapping.
Nodes are categorized as either normal nodes or cluster heads based on a probabilistic approach.

- ***Simulation Loop:***

The main simulation loop iterates over rounds (rmax), updating the state of nodes and clusters.
Nodes may become cluster heads with a probability determined by the LEACH protocol.
Energy consumption is modeled based on the distance between nodes and the sink, considering direct and multi-hop communication.

- ***Visualization:***

The code includes visualization elements using MATLAB's plotting functions to show the deployment of nodes, cluster heads, and the sink node.
The remaining energy of nodes is plotted over time.

➢ MATLAB code:

```
clc;
clear all;
close all;

% Parameters
xm = 3500; % Length of the area
ym = 3500; % Width of the area
sink.x = 0.5 * xm; % Sink coordinates
sink.y = 0.5 * ym;
N = 100; % Number of nodes
p = 0.1; % Probability of selecting a node as a cluster head
Eo = 70; % Initial energy of nodes
```

```
ETX = 1e-11; % Energy consumption for sending a bit by a node and cluster head
ERX = 1e-11; % Energy consumption for receiving a bit by a node and cluster head
Efs = 5e-14; % Energy consumption for amplifying a bit by a node or cluster head for
the direct link
Emp =  0.0005e-12; % Energy consumption for amplifying a bit by a node or cluster
head for the multi-hop link
EDA = 2e-9; % Energy consumption for aggregation of a bit by cluster heads
m = 0.1; % Percentage of nodes that can become cluster heads in each round
a = 1;
rmax = 200; % Maximum number of rounds
do = sqrt(Efs / Emp); % Threshold distance for direct transmission
first_dead = 0; % Record the first death
h1 = figure('name', ['Number of Clusters:' 'Number of Nodes:']);
pause_time = 0.1; % Adjust the pause time for smoother visualization

% Collision avoidance distance
collision_distance = 50;

% Initialization of nodes with collision avoidance
for i = 1:N
    collision_flag = true;
    while collision_flag
        S(i).xd = rand(1, 1) * xm;
        S(i).yd = rand(1, 1) * ym;

        % Check for collisions with existing nodes
        collision_flag = false;
        for j = 1:i-1
            distance_ij = sqrt((S(i).xd - S(j).xd)^2 + (S(i).yd - S(j).yd)^2);
            if distance_ij < collision_distance
                collision_flag = true;
                break;
            end
        end
    end

    S(i).G = 0;
    S(i).type = 'N';
    S(i).id = i; % Assign each node a unique ID
    temp_rnd0 = rand;

    if (temp_rnd0 >= m)
        S(i).E = Eo;
        S(i).ENERGY = 0;
        plot(S(i).xd, S(i).yd, 'bo');
        text(S(i).xd + 30, S(i).yd, num2str(S(i).id), 'FontSize', 8); % Display
smaller node number
        hold on;
    else
        S(i).E = Eo * (1 + a);
        S(i).ENERGY = 1;
        plot(S(i).xd, S(i).yd, 'b+');
        text(S(i).xd + 30, S(i).yd, num2str(S(i).id), 'FontSize', 8); % Display
smaller node number
        hold on;
```

```
    end
end

% Sink node
S(N+1).xd = sink.x;
S(N+1).yd = sink.y;
plot(S(N+1).xd, S(N+1).yd, 'rx');
Cluster = 1;
flag_first_dead = 0;

% Time measurement
tic;

% Remaining energy vs. time plot setup
h2 = figure;
plot_handle = plot(0, 0, 'o-'); % Initialize with a dummy point
title('Remaining Energy vs. Time');
xlabel('Time (seconds)');
ylabel('Remaining Energy');
grid on;

% Main loop
for r = 0:rmax
    pause(pause_time);
    if (mod(r, round(1/p)) == 0)
        for i = 1:N
            S(i).G = 0;
        end
    end

    % Update the main figure
    figure(h1);
    hold off;
    dead = 0;
    set(h1, 'name', ['Number of Clusters:' num2str(sum([S.type] == 'C')) ' Number of
Nodes:' num2str(sum([S.type] == 'N'))]);

    % Nodes visualization
    for i = 1:N
        if (S(i).E <= 0)
            plot(S(i).xd, S(i).yd, 'red.');
            dead = dead + 1;
            text(S(i).xd + 30, S(i).yd, num2str(S(i).id), 'FontSize', 8); % Display
smaller node number
            hold on;
        else
            S(i).type = 'N';
            if (S(i).ENERGY == 0)
                plot(S(i).xd, S(i).yd, 'bo');
                text(S(i).xd + 30, S(i).yd, num2str(S(i).id), 'FontSize', 8); %
Display smaller node number
                title(['Number of dead nodes: ' num2str(dead) ' Number of alive
nodes: ' num2str(N-dead) ' First death: ' num2str(first_dead) ' Round: ' num2str(r) '
Remaining energy: ' num2str(sum([S.E]))]);
                hold on;
```

```
            elseif (S(i).ENERGY == 1)
                plot(S(i).xd, S(i).yd, 'b+');
                text(S(i).xd + 30, S(i).yd, num2str(S(i).id), 'FontSize', 8); %
Display smaller node number
            end
            hold on;
        end
    end
    % Sink node visualization
    plot(S(N+1).xd, S(N+1).yd, 'rx');

    % Check for the first death
    if (dead >= 1)
        if (flag_first_dead == 0)
            first_dead = r;
            flag_first_dead = 1;
        end
    end

    cluster = 1;
    for i = 1:N
        if (S(i).E > 0)
            temp_rand = rand;
            if ((S(i).G) <= 0)
                if (temp_rand <= (p / (1 - p * mod(r, round(1/p)))))
                    S(i).type = 'C';
                    S(i).G = round(1/p) - 1;
                    plot(S(i).xd, S(i).yd, 'k*');
                    C(cluster).xd = S(i).xd;
                    C(cluster).yd = S(i).yd;
                    distance = sqrt((S(i).xd - (S(N+1).xd))^2 + (S(i).yd -
(S(N+1).yd))^2);
                    C(cluster).distance = distance;
                    C(cluster).id = i;
                    cluster = cluster + 1;
                    if (distance > do)
                        S(i).E = S(i).E - ((ETX + EDA) * (4000) + Emp * 4000 *
(distance^4));
                    end
                    if (distance <= do)
                        S(i).E = S(i).E - ((ETX + EDA) * (4000) + Efs * 4000 *
(distance^2));
                    end
                end
            end
        end
    end

    for i = 1:N
        if (S(i).type == 'N' && S(i).E > 0)
            if (cluster - 1 >= 1)
                min_dis = sqrt((S(i).xd - S(N+1).xd)^2 + (S(i).yd - S(N+1).yd)^2);
                sss = min_dis;
                min_dis_cluster = 1;
                for c = 1:cluster - 1
```

```
                    temp = min(sss, sqrt((S(i).xd - C(c).xd)^2 + (S(i).yd -
C(c).yd)^2));
                    if (temp < sss)
                        sss = temp;
                        min_dis_cluster = c;
                    end
                end
                if (sss > do)
                    S(i).E = S(i).E - (ETX * (4000) + Emp * 4000 * (sss^4));
                end
                if (sss <= do)
                    S(i).E = S(i).E - (ETX * (4000) + Efs * 4000 * (sss^2));
                end
                if (sss > 0)
                    S(C(min_dis_cluster).id).E = S(C(min_dis_cluster).id).E - ((ERX +
EDA) * 4000);
                end
            end
        end
    end
    hold on;

    % Record the number of dead nodes for each round
    sta(r+1).death = dead;
    % Update the remaining energy vs. time plot
    figure(h2);
    time_elapsed = toc;
    remaining_energy = sum([S.E]);
    plot_handle.XData = [plot_handle.XData time_elapsed];
    plot_handle.YData = [plot_handle.YData remaining_energy];
    title('Remaining Energy vs. Time');
    xlabel('Time (seconds)');
    ylabel('Remaining Energy');
    hold on;
end
% Plot the results
sta = sta(1:end-1);
figure;
subplot(1, 10, [1, 2, 6, 7]);
plot(1:rmax, N - [sta.death], '-');
title('Lifetime');
xlabel('Round');
ylabel('Number of Alive Nodes');
disp(['Simulation time: ' num2str(time_elapsed) ' seconds']);
```

- **Conclusion:**

The simulation code provides insights into the dynamic behavior of a wireless sensor network using the LEACH protocol. It illustrates the impact of clustering on energy consumption, node death, and the overall network lifetime. The code allows for the analysis of key performance metrics, such as the number of alive nodes over rounds and the remaining energy versus time. Researchers and practitioners can use this simulation as a tool for studying and optimizing the behavior of wireless sensor networks in resource-constrained environments.