ECT-104 Project Report

Trilateration using ESP32 using IITR Wi-Fi

Kunal Virwal (23116049)[1] **and Dakshveer Chahal (23116030)**[2]

[1,2]*Department of Electronics and Communication Engineering, IIT Roorkee*

Abstract—This project explores indoor device trilateration using IITR WiFi. All test and experiments have been conducted in the ground floor of ECE department of IITR. Initial tests were done using ESP32 based access points; and were later scaled to IITR's Wi-Fi routers, identified by their MAC addresses. An ESP32 receiver captured RSSI values, which were processed in MATLAB for generating path loss functions and to estimate position via trilateration. The system demonstrated location accuracy within 2-4 meters under optimal conditions.

Keywords— *Trilateration, ESP32, Indoor GPS, Path Loss, MATLAB*

1. Introduction

This project aims to determine the location of a device within a building using Wi-Fi trilateration. IITR Wi-Fi routers were used as access points identified by their MAC addresses. An ESP32 receiver scans for RSSI values from these routers, and MATLAB processes this data to estimate the receiver's location.

This process involves first passing the received RSSI values through path loss functions which were created using actual RSSI values noted at a regular interval of 5 meters from each router.

The distances derived from the path loss functions were used to find a common region which carried the highest probability of presence of the user.

2. Objective

The objective of this project is to design and implement a real-time indoor location tracking system that leverages the signal strength (RSSI) from known Wi-Fi routers. By utilizing this data, the system aims to demonstrate the practical integration of communication theory, signal processing techniques, and MATLAB-based computation to accurately estimate device positions in an indoor environment.

3. Technical Workflow

3.1. Programming ESP32 to behave as a Wi-Fi receiver

ESP32 is a crucial component of this project. It is used to detect the RSSI values at any point in the target region. It was also used to find out the MAC addresses of the routers being used for the experiment.

It is programmed to simulate a user that will be connected to the IITR Wi-Fi network and will be receiving multiple signal strengths from individual routers. It will detect received signal strength and log this data to MATLAB where further processing will be done.

For security reasons the MAC addresses have been replaced by respective placeholders.

```
#include "WiFi.h"

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.disconnect(true);
  delay(100);
}

void loop() {
  int n = WiFi.scanNetworks();
  if (n == 0) {
    Serial.println("No networks found");
  } else {
    for (int i = 0; i < n; ++i) {
      String ssid = WiFi.SSID(i);
      String bssid = WiFi.BSSIDstr(i);  // MAC address of the scanned AP
      int rssi = WiFi.RSSI(i);

      if (ssid == "IITR_WIFI") {
        String apLabel = "";

        if (bssid == "00:XX:XX:XX:AP:01") {
          apLabel = "AP1";
        }

        else if (bssid == "00:XX:XX:XX:AP:02") {
          apLabel = "AP2";
        }

         else if (bssid == "00:XX:XX:XX:AP:03") {
          apLabel = "AP3";
        }

         else if (bssid == "00:XX:XX:XX:AP:04") {
          apLabel = "AP4";
        }


        else if (bssid == "00:XX:XX:XX:AP:05") {
          apLabel = "AP5";
         }


        if (apLabel != "") {
          Serial.printf("SSID: %s\tLabel: %s\tMAC: %s\tRSSI: %.1f dBm\n",
              ssid.c_str(), apLabel.c_str(), bssid.c_str(), (float)rssi);

        }
      }
    }
  }

  delay(2000); // Scan every 2 seconds
}
```

Code 1. ESP32 receiver code

3.2. Rendering a model of ECE Dept. South Block Ground Floor

To perform a visual analysis of the most favorable area we have to generate a 3D plot of the ECE department. We can simulate this using the `surf()` function in MATLAB.

These are the is a floor map of ground floor of South block ECE department. The scale is 1 unit in simulation equals to 10 centimeters in actual scale. The five peaks visible in the simulation correspond to five routers present on the ground floor of ECE department.

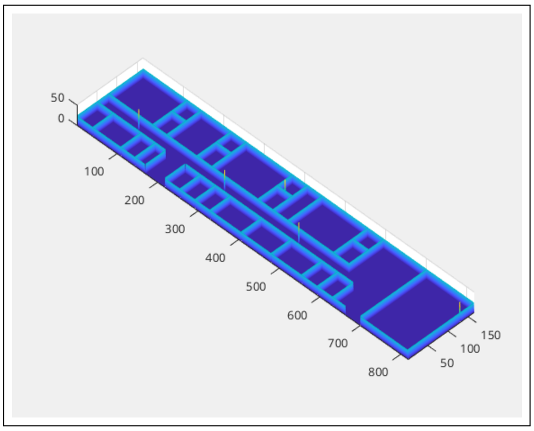

Figure 1. Simulation of ground floor of ECE Dept.

The MATLAB code snippet that generates this simulation is given below.

```
close all;
side =817; % dm floor length
width = 165; % dm floor width
matri = zeros(width,side);

towers = [ 70 , 85;
           70 ,297;
           128, 388;
           70 , 480;
           130 , side;];

% Right wall = x = 1-52
% Left wall = x = 165-80

% Making Walls
matri = Hline(1,1,width,matri); % bottom wall
matri = Hline(700,1,width,matri); % audi wall
matri = Hline(side,1,width,matri); % upper side
matri = Vline(1,1,side,matri); % right side
matri = Vline(width,1,side,matri); % left side

lwalls = [109,144,218,255,369,406,443,555,594];
lwalls = round(lwalls);
for i=1:length(lwalls)
    matri=Hline(lwalls(i),80,165,matri);
end

matri = Vline(80,1,594,matri);

rwalls = [40,109,144,170,218,255,292,329,406,480,555,594,
     632];
rwalls = round(rwalls);
for i=1:length(rwalls)
    matri=Hline(rwalls(i),1,52,matri);
end

matri = Vline(52,1,632,matri);
matri = Vline(128,lwalls(1),lwalls(2),matri);
matri = Vline(128,lwalls(3),lwalls(4),matri);
matri = Vline(128,lwalls(5),lwalls(6),matri);
matri = Vline(128,lwalls(8),lwalls(9),matri);

% Doors and alleys
matri = Rmv(1,665,700,matri);
matri = Rmv(1,170,218,matri);
matri = Rmv(52,170,218,matri);

% Routers
th = 50;
for i=1:length(towers)
    matri(towers(i,1), towers(i,2)) = th;
    matri(towers(i,1)+1, towers(i,2)) = th;
    matri(towers(i,1)-1, towers(i,2)) = th;
    matri(towers(i,1), towers(i,2)+1) = th;
    matri(towers(i,1), towers(i,2)-1) = th;
end

surf(matri);
shading interp;
colormap(parula);  % parula
axis equal tight;
view(45, 45);
hold on;

% Necessary functions
function dis = Edist(x1,y1,x2,y2)
    dis = sqrt((x1-x2)^2+(y1-y2)^2);
end

function matri = Hline(y,x1,x2,matri)
    wh = 24;
    for i = x1:x2
        matri(i,y)=wh;
    end
end

function matri = Vline(x,y1,y2,matri)
    wh = 24;
    for i = y1:y2
        matri(x,i)=wh;
    end
end

function matri = Rmh(y,x1,x2,matri)
    for i = x1:x2
        matri(i,y)=0;
    end
end

function matri = Rmv(x,y1,y2,matri)
    for i = y1:y2
        matri(x,i)=0;
    end
end
```

Code 2. Floor map simulation code

3.3. Computing average RSSI values using ESP32 data in MATLAB

The next step involves accepting data coming from ESP32 in MATLAB using wired media and computing average RSSI values with respect to the five available routers over 20 seconds to get the most accurate results as variations in RSSI can happen at any instant and due to various environmental factors. The following code is followed by Code 1.

```
serialportlist("available")
s = serialport("/dev/cu.usbserial-10", 115200);
configureTerminator(s, "LF");
flush(s);
aps = ["AP1","AP2","AP3","AP4","AP5"];
rssi_logs = containers.Map();

% Initialize each AP with an empty array
for i = 1:length(aps)
    rssi_logs(aps(i)) = [];
end

disp("Starting serial reading for 20 seconds...");
startTime = datetime('now');
durationSeconds = 20;

while seconds(datetime('now') - startTime) <
     durationSeconds
    if s.NumBytesAvailable > 0
        line = readline(s);
        disp(line);

        if contains(line, "Label:") && contains(line, "
     MAC:") && contains(line, "RSSI:")
            parts = split(line, char(9));
            if length(parts) >= 4
                label_part = strrep(parts{2}, "Label: ",
      "");
                rssi_part = strrep(parts{4}, "RSSI: ",
     "");
                rssi_val = str2double(strrep(rssi_part,
     " dBm", ""));

                if ismember(label_part, aps)
                    % Append RSSI values
                    rssi_logs(label_part) = [rssi_logs(
     label_part), rssi_val];
                end
            end
        end
    end
end

% Compute averages
disp("=== Average RSSI Readings Over 20 Seconds ===");
avg_rssi_values = containers.Map();
val=[-120,-120,-120,-120,-120];
for i = 1:length(aps)
```

```
    values = rssi_logs(aps(i));

    if isempty(values)
        avg = -120;  % If no reading is received
    else
        avg = mean(values);
    end
    avg_rssi_values(aps(i)) = avg;
    val(i)=avg;
    fprintf("%s: %.2f dBm\n", aps(i), avg);
end

ap1=val(1);
ap2=val(2);
ap3=val(3);
ap4=val(4);
ap5=val(5);
```

Code 3. RSSI data collection and averaging code

Note: The serial port selection part is written for Mac OS and will vary with device. The variables ap*n* carry the RSSI value corresponding to the nth router.

3.4. Generating path loss functions

To get an approximation of distance from the router using the received we need to get a function that will behave as a path loss function for that particular router. As the difference in position brings about many changes in the path loss, thus we need separate path loss functions for all five access points (routers).

To emulate the path loss functions we collected RSSI data with respect to each router at regular intervals of 5 meters in the central corridor to get discrete points in the path loss functions which we interpolated to get exponential representations of actual path loss functions.

The readings were taken in central corridor so as to ensure line of sight signal strength for most of the routers.

At the end the best fit path loss functions are as follows:

$$AP1 : d_{AP1} = 10^{\frac{(-40.81-r)}{10\times 2.16}}$$

$$AP2 : d_{AP2} = 10^{\frac{(-61.65-r)}{10\times 1.86}}$$

$$AP3 : d_{AP3} = 10^{\frac{(-30.53-r)}{10\times 3.69}}$$

$$AP4 : d_{AP4} = 10^{\frac{(-39.59-r)}{10\times 1.80}}$$

$$AP5 : d_{AP5} = 10^{\frac{(-50.06-r)}{10\times 2.70}}$$

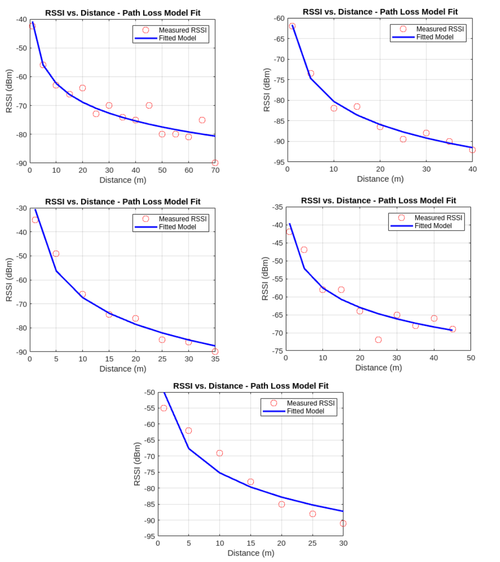

Figure 2. Comparison of path loss model fits for AP1 through AP5.

```
% Data
d = [1,5,10,15,20,25,30,35,40,45,50,55,60,65,70];
r = [-42.5,-56,-63,-66,-64,-73,-70,-74,-75,-70,-80,-81,-75,-90];

% Transform for linear regression
X = log10(d)’;
Y = r’;

% Fit linear model: Y = a + b*log10(d)
coeffs = polyfit(X, Y, 1);
b = coeffs(1);  % Slope
a = coeffs(2);  % Intercept

% Calculate path loss exponent n
n = -b / 10;

fprintf("Estimated RSSI_0 (at 1m): %.2f dBm\n", a);
fprintf("Estimated path loss exponent (n): %.2f\n", n);

% Plot
fitted_rssi = polyval(coeffs, X);
plot(d, r, ’ro’, ’MarkerSize’, 8, ’DisplayName’, ’Measured RSSI’);
hold on;
plot(d, fitted_rssi, ’b-’, ’LineWidth’, 2, ’DisplayName’, ’Fitted Model’);
xlabel(’Distance (m)’);
ylabel(’RSSI (dBm)’);
legend;
title(’RSSI vs. Distance - Path Loss Model Fit’);
grid on;hold off
out=@(r) 10.^((-40.81-r)/(10*2.16))
rssi = -90;  % RSSI reading from ESP32
distance_est = out(rssi);
fprintf("Estimated distance for RSSI %.1f dBm: %.2f meters\n", rssi, distance_est);
% Data
d = [1,5,10,15,20,25,30,35,40];
r = [-62,-73.5,-82,-81.5,-86.5,-89.5,-88,-90,-92,];


% Transform for linear regression
X = log10(d)’;
Y = r’;

% Fit linear model: Y = a + b*log10(d)
coeffs = polyfit(X, Y, 1);
b = coeffs(1);  % Slope
a = coeffs(2);  % Intercept

% Calculate path loss exponent n
n = -b / 10;

fprintf("Estimated RSSI_0 (at 1m): %.2f dBm\n", a);
fprintf("Estimated path loss exponent (n): %.2f\n", n);

% Plot
fitted_rssi = polyval(coeffs, X);
plot(d, r, ’ro’, ’MarkerSize’, 8, ’DisplayName’, ’Measured RSSI’);
hold on;
plot(d, fitted_rssi, ’b-’, ’LineWidth’, 2, ’DisplayName’, ’Fitted Model’);
xlabel(’Distance (m)’);
ylabel(’RSSI (dBm)’);
```

```
legend;
title('RSSI vs. Distance - Path Loss Model Fit');
grid on;
hold off
out=@(r) 10.^((-61.65-r)/(10*1.86))
rssi = -85;  % RSSI reading from ESP32
distance_est = out(rssi);
fprintf("Estimated distance for RSSI %.1f dBm: %.2f
    ↪ meters\n", rssi, distance_est);
% Data
d = [1,5,10,15,20,25,30,35];
r = [-35,-49,-66,-74.5,-76,-85,-86,-90];

% Transform for linear regression
X = log10(d)';
Y = r';

% Fit linear model: Y = a + b*log10(d)
coeffs = polyfit(X, Y, 1);
b = coeffs(1);  % Slope
a = coeffs(2);  % Intercept

% Calculate path loss exponent n
n = -b / 10;

fprintf("Estimated RSSI_0 (at 1m): %.2f dBm\n", a);
fprintf("Estimated path loss exponent (n): %.2f\n", n);

% Plot
fitted_rssi = polyval(coeffs, X);
plot(d, r, 'ro', 'MarkerSize', 8, 'DisplayName', '
    ↪ Measured RSSI');
hold on;
plot(d, fitted_rssi, 'b-', 'LineWidth', 2, 'DisplayName',
    ↪ 'Fitted Model');
xlabel('Distance (m)');
ylabel('RSSI (dBm)');
legend;
title('RSSI vs. Distance - Path Loss Model Fit');
grid on;hold off
out=@(r) 10.^((-30.53-r)/(10*3.69))
rssi = -120;  % RSSI reading from ESP32
distance_est = out(rssi);
fprintf("Estimated distance for RSSI %.1f dBm: %.2f
    ↪ meters\n", rssi, distance_est);
% Data
d = [1,5,10,15,20,25,30,35,40,45];
r = [-42,-47,-58,-58,-64,-72,-65,-68,-66,-69];

% Transform for linear regression
X = log10(d)';
Y = r';

% Fit linear model: Y = a + b*log10(d)
coeffs = polyfit(X, Y, 1);
b = coeffs(1);  % Slope
a = coeffs(2);  % Intercept

% Calculate path loss exponent n
n = -b / 10;

fprintf("Estimated RSSI_0 (at 1m): %.2f dBm\n", a);
fprintf("Estimated path loss exponent (n): %.2f\n", n);

% Plot
fitted_rssi = polyval(coeffs, X);
plot(d, r, 'ro', 'MarkerSize', 8, 'DisplayName', '
    ↪ Measured RSSI');
hold on;
plot(d, fitted_rssi, 'b-', 'LineWidth', 2, 'DisplayName',
    ↪ 'Fitted Model');
xlabel('Distance (m)');
ylabel('RSSI (dBm)');
legend;
title('RSSI vs. Distance - Path Loss Model Fit');
grid on;hold off
out=@(r) 10.^((-39.59-r)/(10*1.80))
rssi = -64;  % RSSI reading from ESP32
distance_est = out(rssi);
fprintf("Estimated distance for RSSI %.1f dBm: %.2f
    ↪ meters\n", rssi, distance_est);
% Data
d = [1,5,10,15,20,25,30];
r = [-55,-62,-69,-78,-85,-88,-91,];

% Transform for linear regression
X = log10(d)';
Y = r';

% Fit linear model: Y = a + b*log10(d)
coeffs = polyfit(X, Y, 1);
b = coeffs(1);  % Slope
a = coeffs(2);  % Intercept

% Calculate path loss exponent n
n = -b / 10;

fprintf("Estimated RSSI_0 (at 1m): %.2f dBm\n", a);
fprintf("Estimated path loss exponent (n): %.2f\n", n);

% Plot
fitted_rssi = polyval(coeffs, X);
plot(d, r, 'ro', 'MarkerSize', 8, 'DisplayName', '
    ↪ Measured RSSI');
hold on;
plot(d, fitted_rssi, 'b-', 'LineWidth', 2, 'DisplayName',
    ↪ 'Fitted Model');
xlabel('Distance (m)');
ylabel('RSSI (dBm)');
legend;
title('RSSI vs. Distance - Path Loss Model Fit');
grid on;hold off
out=@(r) 10.^((-50.06-r)/(10*2.7))
rssi = -85;  % RSSI reading from ESP32
distance_est = out(rssi);
fprintf("Estimated distance for RSSI %.1f dBm: %.2f
    ↪ meters\n", rssi, distance_est);
```

Code 4. Interpolation of observed RSSI values to generate path loss functions

3.5. Plotting the Estimated Area of User Presence

Now we can use the calculated path loss functions from Code 4 to estimate the distance of the user from routers AP1 to AP5. This distance can be used to plot circles centered at the respective routers. The points of interaction and the centroid of the common region represent points of highest probability of being the user's actual location.

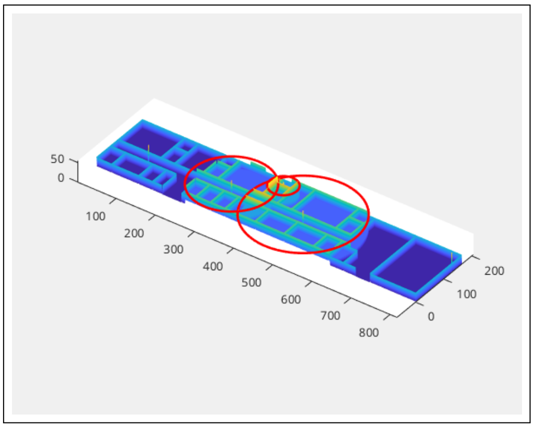

Figure 3. Region of highest probability.

We can also plot the centroid of the region of highest probability as it is also an important point used to estimate the location of the user inside the region. The highest peak visible in Figure 4 represents the centroid of the common region.

Note: So as to calculate centroid, the image processing toolbox in MATLAB must be installed.

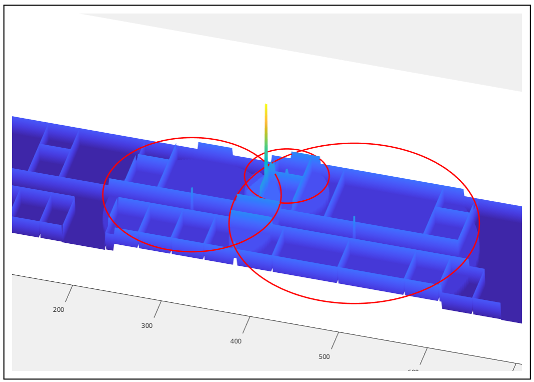

Figure 4. Centroid of the common region.

The following code is used to convert RSSI values to distances using the calculated path loss functions and make the best probability region plot. This code directly follows Code 3:

```
% Getting expected value of radius from trained function
%  AP1
out=@(r) 10.^((-40.81-r)/(10*2.16));
rssi = ap1;  % RSSI reading from ESP32
distance_est_AP1 = out(rssi);
fprintf("Estimated distance for RSSI %.1f dBm: %.2f meters\n", rssi, distance_est_AP1);

% AP2
out=@(r) 10.^((-61.65-r)/(10*1.86));
rssi = ap2;  % RSSI reading from ESP32
distance_est_AP2 = out(rssi);
fprintf("Estimated distance for RSSI %.1f dBm: %.2f meters\n", rssi, distance_est_AP2);

% AP3
out=@(r) 10.^((-30.53-r)/(10*3.69));
rssi = ap3;  % RSSI reading from ESP32
distance_est_AP3 = out(rssi);
fprintf("Estimated distance for RSSI %.1f dBm: %.2f meters\n", rssi, distance_est_AP3);

% AP4
out=@(r) 10.^((-39.59-r)/(10*1.80));
rssi = ap4;  % RSSI reading from ESP32
distance_est_AP4 = out(rssi);
fprintf("Estimated distance for RSSI %.1f dBm: %.2f meters\n", rssi, distance_est_AP4);

% AP5
out=@(r) 10.^((-50.06-r)/(10*2.52));
rssi = ap5;  % RSSI reading from ESP32
distance_est_AP5 = out(rssi);
fprintf("Estimated distance for RSSI %.1f dBm: %.2f meters\n", rssi, distance_est_AP5);

mdis=60;
h=10;
for i=1:width
    for j=1:side
        if (Edist(i,j,towers(1,1),towers(1,2))<distance_est_AP1*10 && distance_est_AP1<mdis)
            matri(i,j)=matri(i,j)+h;
        end

        if (Edist(i,j,towers(2,1),towers(2,2))<distance_est_AP2*10 && distance_est_AP2<mdis)
            matri(i,j)=matri(i,j)+h;
        end

        if (Edist(i,j,towers(3,1),towers(3,2))<distance_est_AP3*10 && distance_est_AP3<mdis)
            matri(i,j)=matri(i,j)+h;
        end

        if (Edist(i,j,towers(4,1),towers(4,2))<distance_est_AP4*10 && distance_est_AP4<mdis)
            matri(i,j)=matri(i,j)+h;
        end

        if (Edist(i,j,towers(5,1),towers(5,2))<distance_est_AP5*10 && distance_est_AP5<mdis)
            matri(i,j)=matri(i,j)+h;
        end
    end
end

hmax=0;
for i = 1:width
    for j = 1:side
        if matri(i,j)>hmax
            hmax = matri(i,j);
        end
    end
end

bin_matri=zeros(width,side);
for i = 1:width
    for j = 1:side
        if matri(i,j)~=hmax
            bin_matri(i,j)=0;
        else
            bin_matri(i,j)=1;
        end
    end
end

% Routers

th = 50;

for i=1:length(towers)
    matri(towers(i,1), towers(i,2)) = th;
    matri(towers(i,1)+1, towers(i,2)) = th;
    matri(towers(i,1)-1, towers(i,2)) = th;
    matri(towers(i,1), towers(i,2)+1) = th;
    matri(towers(i,1), towers(i,2)-1) = th;
end

stats = regionprops(bin_matri, 'Centroid');
centroid_coords = stats.Centroid;

centroid_X=round(centroid_coords(2));
centroid_y=round(centroid_coords(1));

expected_centroid=150;

disp(num2str(centroid_X)+"  "+num2str(centroid_y))

matri(centroid_X,centroid_y) = expected_centroid;
matri(centroid_X+1,centroid_y-1) = expected_centroid;
matri(centroid_X+1,centroid_y+1) = expected_centroid;
matri(centroid_X-1,centroid_y-1) = expected_centroid;
matri(centroid_X-1,centroid_y+1) = expected_centroid;
matri(centroid_X,centroid_y-1) = expected_centroid;
matri(centroid_X,centroid_y+1) = expected_centroid;
matri(centroid_X-1,centroid_y) = expected_centroid;
matri(centroid_X-1,centroid_y) = expected_centroid;

% Plotting probable location

dist=[distance_est_AP1,distance_est_AP2,distance_est_AP3,distance_est_AP4,distance_est_AP5];
figure;
for i=1:5
    xr=towers(i,1);
    yr=towers(i,2);
    r=dist(i)*10; % dm
    if dist(i)>mdis
        continue
    end
    theta = linspace(0, 2*pi, 200);
    xc = yr + r*cos(theta);
    yc = xr + r*sin(theta);
    plot3(xc, yc, repmat(40, size(xc)), 'r-', 'LineWidth', 2);
    hold on;
end
```

```matlab

surf(matri);
shading interp;
colormap(parula);  % parula
axis equal tight;
view(45, 45);
hold on;
```

Code 5. Best probability region and Centroid code

4. Findings and Observations

The system was tested on the ground floor of South block of ECE Dept using known positions of AP1–AP5. RSSI readings were successfully captured and averaged over 20 seconds. Trilateration produced estimated locations that were generally within 2–5 meters of the actual position, depending on signal quality and line-of-sight to the routers.

Some noticeable observations are:

- RSSI values follow a trend but have significant variations between reading amount of which depends on position and time of the observation.
- RSSI readings significantly improved during the night time which may be traced back to lowering of temperature or the lowering of load on routers during the night time.
- The accuracy dropped in areas with significant obstacles (e.g., walls, pillars), as expected with RSSI-based localization. Specifically areas with metallic objects like metal cupboards and fire extinguishers showed significant deviations from the expected trends of path loss functions.
- The predictions made along Y axis (y coordinates) were significantly better than those in X axis (x coordinates) due to presence of multiple levels of routers along the Y axis.

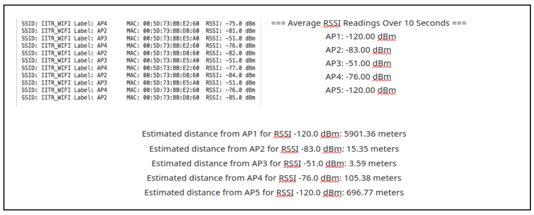

Figure 5. RSSI readings averaged over 20 seconds

5. Testing and Error Analysis

So as to test the viability and effectiveness of this system, we ran it through a series of observations. We picked up six locations at random and conducted the experiment three times for each locations. Here is the testing code and observed outputs:

```matlab
close all;
% Actual positions of the user
a=[
 13.5 , 62.4; % In South Block Lobby
 6.72, 53.1; % In front of second office in the corridor
 6.72, 32.05; % At the mid point of the corridor
 9.5  , 17.0; % In the corner of the new lab in front of
    ↪ corridor stairs
 9.6  , 52.8; % Communication Systems lab
 15.5 , 35.9 % Signal Processing lab
];

% Predicted positions
pred1=[8.5,47.1;10.8,68.5;9.9,67.4];
pred2=[7.1,48.3;9.5,23.7;10.2,25.9];
pred3=[10.3,35.8;10.5,35.2;8.1,35.6];
pred4=[8.2,12.8;8.2,12.3;8.2,12.5];
pred5=[12.0,45.2;8.9,41.4;12.6,47.7];
pred6=[12.8,38.8;12.8,38.9;12.8,38.9];

% Making a 3D array
preds = cat(3,pred1,pred2,pred3,pred4,pred5,pred6);

colors = [
    0.0000, 0.4470, 0.7410;  % Blue
    0.8500, 0.3250, 0.0980;  % Red-Orange
    0.9290, 0.6940, 0.1250;  % Yellow
    0.4940, 0.1840, 0.5560;  % Purple
    0.4660, 0.6740, 0.1880;  % Green
    0.3010, 0.7450, 0.9330;  % Cyan
];

%% X axis analysis

ax= a(:,1);
fig=figure;
fig.Position = [0 0 1000 500];
% Predictions vs actual
subplot(1,2,1)
for i=1:length(a)
    px=squeeze(preds(:,1,i));
    scatter(ax(i), px, 80, 'filled', 'MarkerFaceColor',
     ↪ colors(i,:));
    hold on;
    avg=sum(px)/3;
    scatter(ax(i), avg, 80, 'x', 'MarkerEdgeColor',
     ↪ colors(i,:));
    hold on;
end
lims = [min(ax,[],'all')  max(ax,[],'all')];
plot(lims, lims, 'r--', 'LineWidth', 1.5);
xlabel('Actual X Coordinates (m) ');
ylabel('Predicted X Coordinate (m)');
title('Actual vs. Predicted X');
grid on;

% Errors vs Location
subplot(1,2,2);
errors=[0,0,0,0,0,0];
for i=1:length(a)
    px=squeeze(preds(:,1,i));
    avg=sum(px)/3;
    errors(i)=avg-ax(i);
end
location={'A','B','C','D','E','F'};
b=bar(location,errors);
for k = 1:length(errors)
    b.FaceColor = 'flat';
    b.CData(k,:) = colors(k,:);
end
xlabel('Location');
ylabel('Error (m)');
title('Error vs Location');
grid on;

%% Y axis analysis

ay= a(:,2);
fig=figure;
fig.Position = [0 0 1000 500];
% Predictions vs actual
subplot(1,2,1)
for i=1:length(a)
    py=squeeze(preds(:,2,i));
    scatter(ay(i), py, 80, 'filled', 'MarkerFaceColor',
     ↪ colors(i,:));
    hold on;
    avg=sum(py)/3;
    scatter(ay(i), avg, 80, 'x', 'MarkerEdgeColor',
     ↪ colors(i,:));
    hold on;
end
lims = [min(ay,[],'all')  max(ay,[],'all')];
plot(lims, lims, 'r--', 'LineWidth', 1.5);
xlabel('Actual Y Coordinates (m) ');
ylabel('Predicted Y Coordinate (m)');
title('Actual vs. Predicted Y');
grid on;

% Errors vs Location
subplot(1,2,2);
```

```
errors=[0,0,0,0,0,0];
for i=1:length(a)
    py=squeeze(preds(:,2,i));
    avg=sum(py)/3;
    errors(i)=avg-ay(i);
end
location={'A','B','C','D','E','F'};
b=bar(location,errors);
for k = 1:length(errors)
    b.FaceColor = 'flat';
    b.CData(k,:) = colors(k,:);
end
xlabel('Location');
ylabel('Error (m)');
title('Error vs Location');
grid on;

%% Cluster plot

figure;
for i=1:length(a)
    px=squeeze(preds(:,1,i));
    py=squeeze(preds(:,2,i));
    scatter(px, py, 80, 'filled', 'MarkerFaceColor', colors(i,:));
    hold on;
    avgx=sum(px)/3;
    avgy=sum(py)/3;
    scatter(avgx, avgy, 80, 'x', 'MarkerEdgeColor', colors(i,:));
    hold on;
    scatter(a(i,1), a(i,2), 80, '^', 'MarkerFaceColor', colors(i,:),'MarkerEdgeColor', colors(i,:));
    hold on;
end

plot([1, 16.5], [1, 1], 'k-', 'LineWidth', 1.5);
plot([1, 16.5], [81.7, 81.7], 'k-', 'LineWidth', 1.5);
plot([1, 16.5], [70, 70], 'k-', 'LineWidth', 1.5);
plot([1, 1], [1, 81.7], 'k-', 'LineWidth', 1.5);
plot([16.5, 16.5], [1, 81.7], 'k-', 'LineWidth', 1.5);
plot([8, 8], [1, 59.4], 'k-', 'LineWidth', 1.5);
plot([5.2, 5.2], [1, 63.2], 'k-', 'LineWidth', 1.5);
plot([8, 16.5], [59.4, 59.4], 'k-', 'LineWidth', 1.5);
plot([1, 5.2], [63.2, 63.2], 'k-', 'LineWidth', 1.5);

axis equal;
set(gca, 'XDir', 'reverse')

% Add legend
h1=plot(nan, nan, 'o','Color', 'k', 'MarkerFaceColor', 'k');
h2=plot(nan, nan, 'x', 'Color', 'k');
h3=plot(nan, nan, '^', 'Color', 'k', 'MarkerFaceColor', 'k');
hLegend=legend([h1,h2,h3], {'Predicted positions', 'Avg. Prediction', 'Actual position'});
xlabel('X axis (16.5m');
ylabel('Y axis (81.7m)');
title('South Block ECE Department');
grid on;
```

Code 6. Error analysis and testing code

- The first set of readings were taken in the South block lobby in which 2/3 results proved to be quite accurate.
- The second set of readings were taken in the corridor of South block from in front of second office in the corridor. These readings were highly inaccurate due to reasons like presence of many metallic bodies.
- The third set readings are taken at the mid point of the ECE corridor where 2/3 readings were accurate and in the 3rd reading even though there was no single best region of probability, the centroid lied pretty close to actual position.
- The forth set of readings were taken from inside the lab room in front of the corridor stairs. Due to the absence of multiple routers in this region, the region of best probability is quite large but is quite accurate in terms of positioning.
- The fifth set of readings were taken from inside the communication systems lab and all the observed readings were quite accurate
- The sixth set of readings were taken from inside the signal processing lab and the results where highly accurate.

Now to analyze this data we can plot all actual positions and predicted positions along with there respective means.

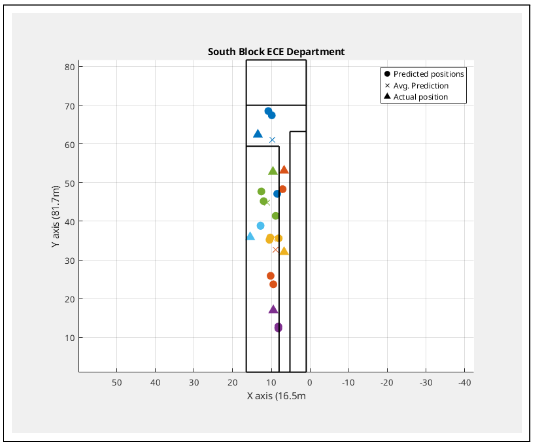

Figure 6. 2D Plot of Actual and Predicted positions

Some more graphs tell us that what is the extent of error present in our system. We can plot the scatter plots of Actual vs Predicted X and Y. In the following graphs color represents a particular position where reading was taken.

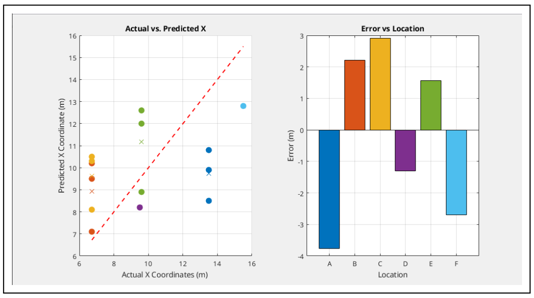

Figure 7. Actual X vs Predicted X and Error vs Location in X

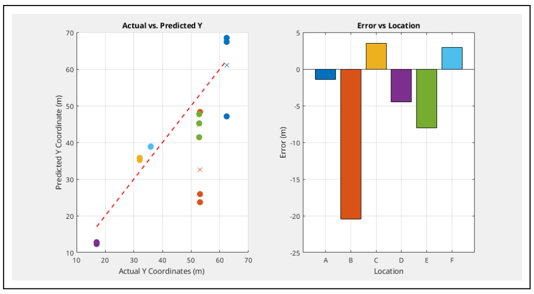

Figure 8. Actual Y vs Predicted Y and Error vs Location in Y

From these graphs we can see that the average error in position lies between 3-4 meters except ignoring the the outlier error in Y at position B.

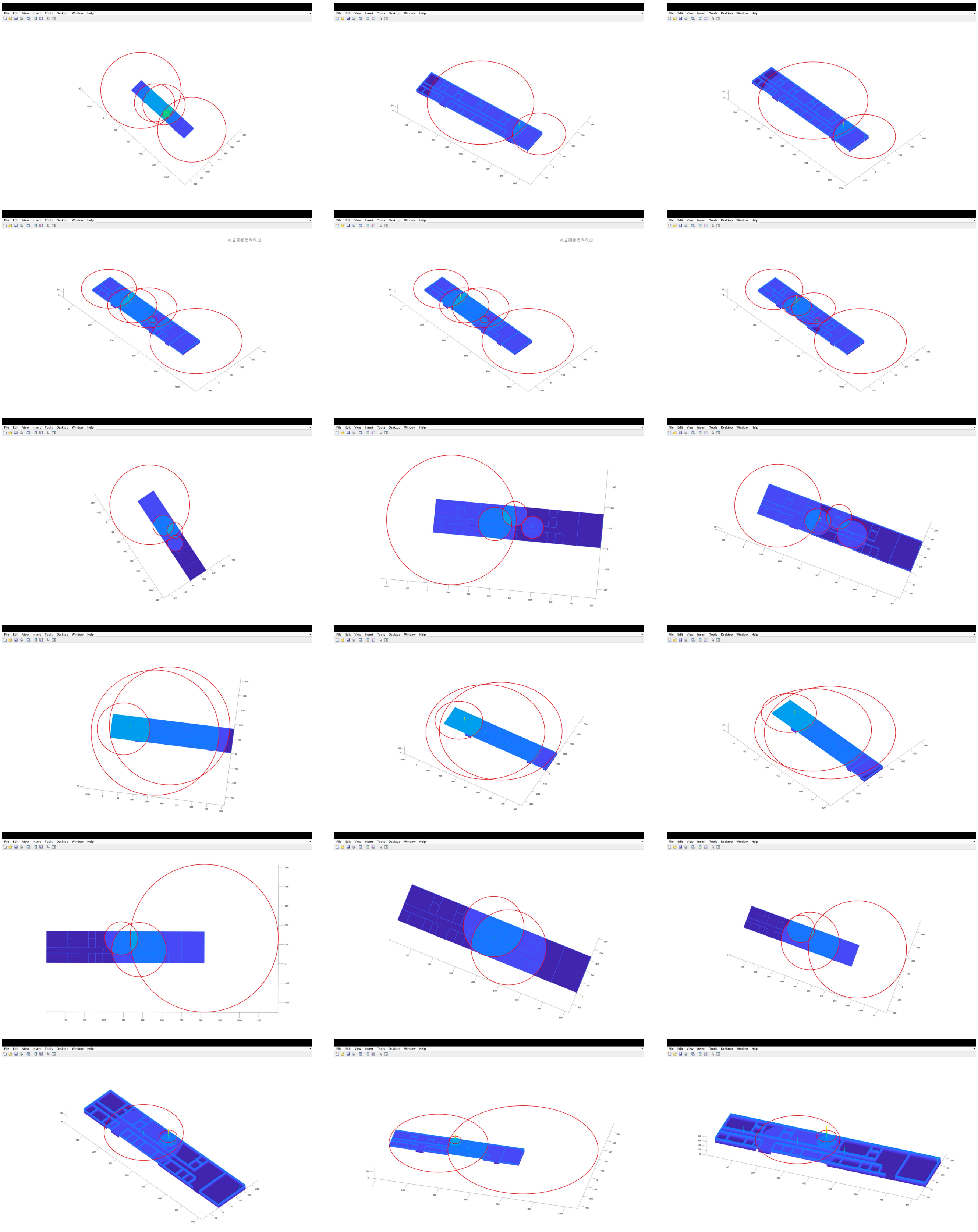

Figure 9. All 18 orientations of the object captured at the 6 positions with 3 observations each.

6. Conclusions and Applications

This project demonstrates that RSSI-based location trilateration using any centralized Wi-Fi routers is feasible and reasonably accurate in indoor environments. It also highlights the importance of network modeling, noise mitigation, and mathematical processing in practical communication networks.

This system has many practical applications that can be used to resolve many real life logistical issues such as:

- **Indoor Navigation in Retail Stores (e.g., Walmart, Target):** Wi-Fi-based location trilateration can be used to determine the exact location of a shopper inside a large store like Walmart. Customers can be guided to product shelves using a mobile app. The system can show nearby offers, personalized recommendations and navigation assistance. This enhances customer experience and operational efficiency.
- **Asset Tracking in Warehouses:** Equipment and critical medical devices in hospitals (or inventory in warehouses) can be tagged and tracked in real-time to prevent loss and speed up access.
- **Smart Campus or Office Automation:** The system can adjust lighting, HVAC, or resource allocation based on the real-time presence of users in specific zones, enhancing energy efficiency.

7. Acknowledgment

We would like to express our sincere gratitude to Prof. Abhay Kumar Sah and Prof. Ekant Sharma for giving us the opportunity to work on this course project. Their guidance, encouragement, and insightful feedback throughout the project were invaluable and greatly contributed to our learning experience. We are truly grateful for their support and mentorship.

8. References

The following are some resources which were quite helpful in making this project:

- https://arxiv.org/pdf/1912.07801
- https://ieeexplore.ieee.org/document/10496456
- https://forum.arduino.cc/t/esp32-wifi-rssi/1150959