20XTO5 – WIRELESS NETWORKS

ASSIGNMENT PRESENTATION

DONE BY

21PT28 - SHERIN. J. A

21PT32 - THIRULOASHANA. A

21PT34 - V. SHAMYUKTHA

ON

BLE PROXIMITY MONITORING APPLICATION

Project report submitted in partial fulfilment of the requirements for the degree of

FIVE YEAR INTEGRATED

M.Sc. THEORETICAL COMPUTER SCIENCE

of Anna University

MARCH - 2025

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)
COIMBATORE – 641 004

CONTENTS

INTRODUCTION 2

BLE PROXIMITY MONITORING 3

  2.1 PROPERTIES 3

  2.2 DETAILED METHODOLOGY 3

    2.2.1 BLE Device Scanning 3

    2.2.2 Distance Estimation 3

    2.2.3 Data Logging 4

    2.2.4 Visualization 4

  2.3 ADVANTAGES 4

  2.4 DISADVANTAGES 4

  2.5 APPLICATIONS 4

PYTHON IMPLEMENTATION 5

OUTPUT - DEMONSTRATION 8

CONCLUSION 10

REFERENCES 11

CHAPTER 1

INTRODUCTION

Bluetooth Low Energy (BLE) technology is widely used for short-range wireless communication, enabling applications such as device tracking, proximity detection, and IoT connectivity. The BLE Proximity Monitoring Application leverages BLE scanning to detect nearby devices, estimate their distances based on Received Signal Strength Indicator (RSSI), and visualize their positions relative to the scanning device. This project is particularly relevant in scenarios requiring real-time spatial awareness, such as asset tracking and indoor navigation.

In this project, we implement a BLE proximity monitoring system using Python, integrated with a Streamlit-based graphical interface for user interaction and visualization. The system scans for BLE devices, logs their details (MAC address, name, and RSSI), estimates distances using a path-loss model, and presents a radial proximity map using Plotly. The application is designed to be user-friendly, allowing customization of scan duration and providing intuitive outputs for analysis.

This report details the methodology, implementation, and results of the BLE proximity monitoring system, supported by demo images of the executed code and analysis of detected devices, demonstrating its effectiveness in real-world proximity detection scenarios.

CHAPTER 2

BLE PROXIMITY MONITORING

2.1 PROPERTIES

- **Low Power Consumption:** BLE enables efficient scanning with minimal energy usage.
- **Real-Time Detection:** Provides instantaneous discovery of nearby devices.
- **Distance Estimation:** Uses RSSI to approximate device proximity.
- **Visualization:** Supports graphical representation of device locations.

2.2 DETAILED METHODOLOGY

2.2.1 BLE Device Scanning

- The process begins with an asynchronous BLE scan using the BleakScanner library, which detects nearby devices broadcasting BLE signals.
- The scan duration is user-configurable, typically set to 10 seconds, to balance thoroughness and responsiveness.
- Detected devices are characterized by their MAC address, name (if available), and RSSI.

2.2.2 Distance Estimation

- The RSSI value is used to estimate the distance to each device using a logarithmic path-loss model:

$$Distance = 10^{(TxPower - RSSI)/(10.n)}$$

where $TxPower = -59\ dBm\ (assumed)\ and\ n = 2$.

- The calculated distance is rounded to two decimal places for clarity.

2.2.3 Data Logging

- Device details (name, MAC address, RSSI) are logged into a text file with a timestamp for record-keeping and analysis.
- Logs include cases where no devices are found to ensure comprehensive tracking.

2.2.4 Visualization

- A radial proximity map is generated using Plotly, with the scanning device (laptop) at the origin and detected devices plotted at coordinates derived from their estimated distances and angular distribution.
- Each device is represented by a marker, connected to the origin by a dashed line, with its name displayed for identification.

2.3 ADVANTAGES

- Non-intrusive detection of BLE devices
- User-friendly graphical interface
- Accurate proximity estimation for short ranges
- Persistent logging for analysis

2.4 DISADVANTAGES

- RSSI-based distance estimation varies with environmental factors
- Limited to BLE-enabled devices
- Requires compatible hardware for scanning
- Assumed TxPower may reduce accuracy

2.5 APPLICATIONS

- Asset tracking
- Indoor navigation systems
- Proximity-based alerts
- IoT device management

CHAPTER-3

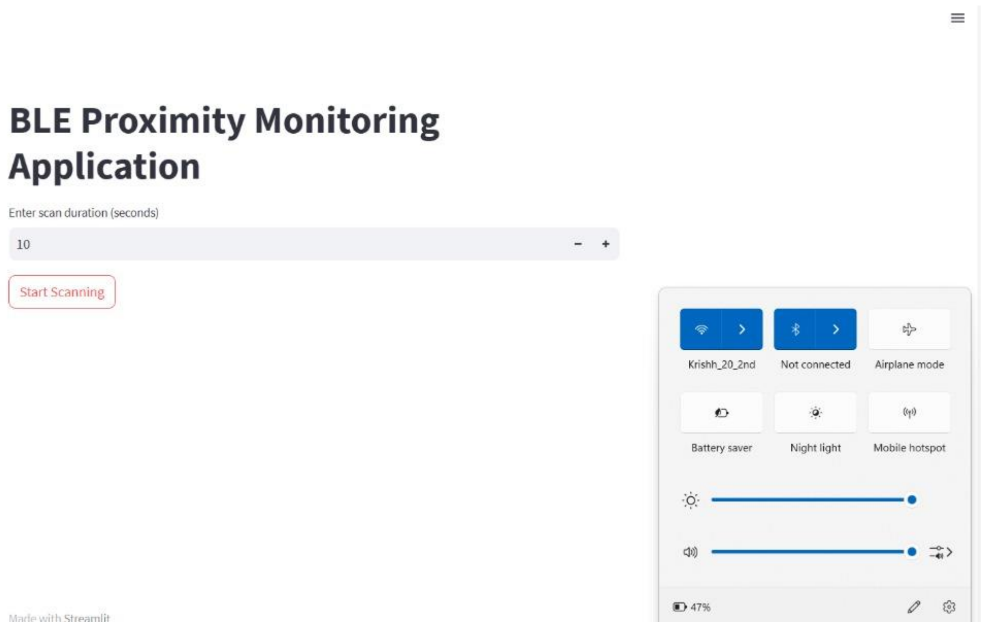
PYTHON IMPLEMENTATION

```
import asyncio
import streamlit as st
from bleak import BleakScanner
from datetime import datetime
import math
import plotly.graph_objects as go
from mac_vendor_lookup import MacLookup

# Initialize MAC lookup
mac_lookup = MacLookup()

UNKNOWN_ICON = "https://cdn-icons-png.flaticon.com/512/0/186.png"

async def scan_ble_devices(duration=10):
    print(f"Scanning for BLE devices for {duration} seconds...")
    scanner = BleakScanner()
    await scanner.start()
    await asyncio.sleep(duration)
    devices = scanner.discovered_devices
    await scanner.stop()
    return [(device.address, device.name or "Unknown", device.rssi) for device in devices]

def estimate_distance(rssi):
    tx_power = -59  # Assumed Tx power
    n = 2  # Path loss exponent
    distance = 10 ** ((tx_power - rssi) / (10 * n))
    return round(distance, 2)

def log_devices(devices, filename="ble_log.txt"):
    with open(filename, "a") as log_file:
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        log_file.write(f"\nScan at {timestamp}:\n")
        if devices:
            for addr, name, rssi in devices:
                log_file.write(f"Device: {name}, MAC: {addr}, RSSI: {rssi} dBm\n")
        else:
            log_file.write("No devices found.\n")

def create_radial_plot(devices):
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=[0], y=[0], mode="markers+text", marker=dict(size=30,
symbol="pentagon", color="blue"), text=["Laptop"], textposition="middle center"))
    for i, (addr, name, rssi) in enumerate(devices):
        distance = estimate_distance(rssi)
        angle = (i / len(devices)) * 2 * math.pi
        x = distance * math.cos(angle)
        y = distance * math.sin(angle)
        fig.add_trace(go.Scatter(x=[x], y=[y], mode="markers+text", marker=dict(size=20,
color="red"), text=[name], textposition="top center"))
        fig.add_trace(go.Scatter(x=[0, x], y=[0, y], mode="lines", line=dict(color="gray",
dash="dash")))
    fig.update_layout(title="BLE Device Proximity Map", showlegend=False,
xaxis_title="Distance (m)", yaxis_title="Distance (m)", xaxis=dict(range=[-30, 30]),
yaxis=dict(range=[-30, 30]))
    return fig

def main():
    st.title("BLE Proximity Monitoring Application")
    scan_duration = st.number_input("Enter scan duration (seconds)", min_value=1,
value=10)

    if st.button("Start Scanning"):
        with st.spinner("Scanning for BLE devices..."):
            devices = asyncio.run(scan_ble_devices(scan_duration))
            log_devices(devices)

            if devices:
                st.success(f"Found {len(devices)} devices!")
                for addr, name, rssi in devices:
                    icon = UNKNOWN_ICON
                    distance = estimate_distance(rssi)
                    col1, col2 = st.columns([1, 3])
                    with col1:
                        st.image(icon, width=50)
                    with col2:
```

```
                st.write(f"**Name**: {name}")
                st.write(f"**MAC**: {addr}")
                st.write(f"**Distance**: ~{distance} meters (RSSI: {rssi} dBm)")
            st.write("---")
        st.plotly_chart(create_radial_plot(devices))
    else:
        st.warning("No devices found.")

if __name__ == "__main__":
    main()
```

CHAPTER-4

OUTPUT - DEMONSTRATION

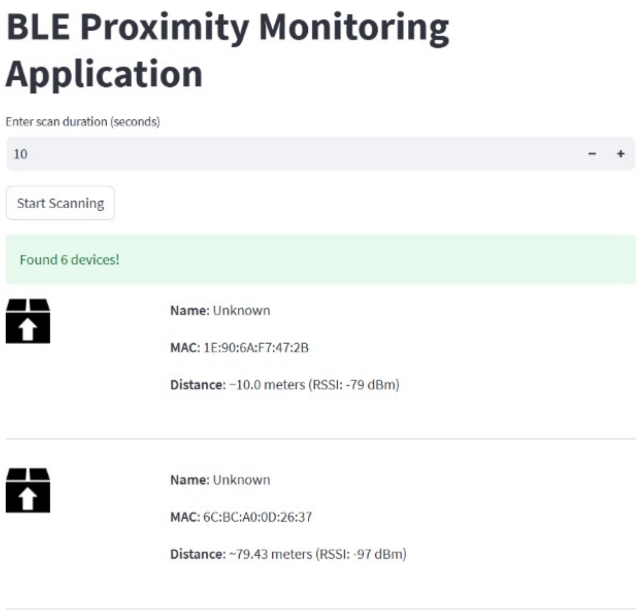


BLE Proximity Monitoring Application

Enter scan duration (seconds)

10

Start Scanning



BLE Proximity Monitoring Application

Enter scan duration (seconds)

10

Start Scanning

Found 6 devices!

Name: Unknown

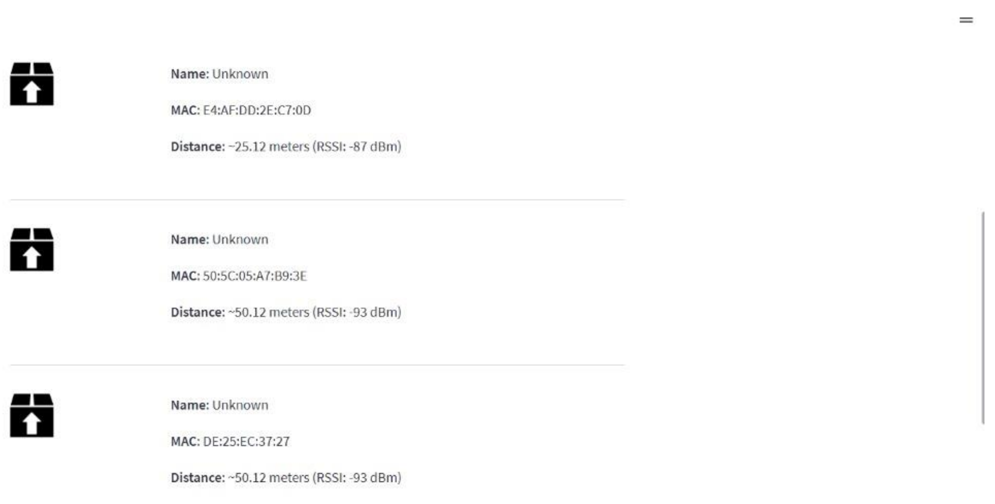
MAC: 1E:90:6A:F7:47:2B

Distance: ~10.0 meters (RSSI: -79 dBm)

Name: Unknown

MAC: 6C:BC:A0:0D:26:37

Distance: ~79.43 meters (RSSI: -97 dBm)



Name: Unknown

MAC: E4:AF:DD:2E:C7:0D

Distance: ~25.12 meters (RSSI: -87 dBm)

Name: Unknown

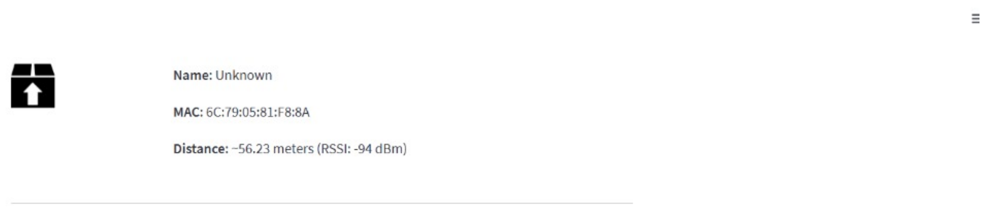
MAC: 50:5C:05:A7:B9:3E

Distance: ~50.12 meters (RSSI: -93 dBm)

Name: Unknown

MAC: DE:25:EC:37:27

Distance: ~50.12 meters (RSSI: -93 dBm)

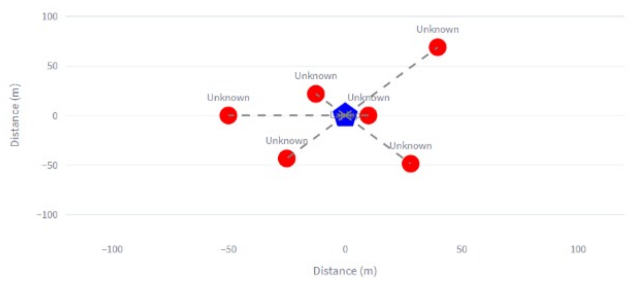


Name: Unknown

MAC: 6C:79:05:81:F8:8A

Distance: ~56.23 meters (RSSI: -94 dBm)

BLE Device Proximity Map



CHAPTER-5

CONCLUSION

The BLE Proximity Monitoring Application demonstrates the potential of BLE technology for real-time device detection and proximity estimation. By integrating asynchronous scanning, RSSI-based distance calculations, and interactive visualization, the system provides a practical tool for applications requiring spatial awareness. While limitations such as environmental interference and assumed TxPower values impact precision, the application’s simplicity and visual output make it valuable for asset tracking, IoT, and proximity-based services. Future enhancements could include adaptive TxPower calibration and advanced filtering to improve accuracy.

REFERENCES

1. Bluetooth SIG. (2024). Bluetooth Low Energy Specification. Retrieved from https://www.bluetooth.com/.
2. Bleak Documentation. (2024). Retrieved from https://bleak.readthedocs.io/.
3. Streamlit Documentation. (2024). Retrieved from https://docs.streamlit.io/.
4. Plotly Python Documentation. (2024). Retrieved from https://plotly.com/python/.
5. Gomez, C., Oller, J., & Paradells, J. (2012). Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. Sensors, 12(9), 11734-11753.