

Disaster-Resilient Telecommunication Infrastructure: A Systematic Approach

24-25J-049

Team



Supervisor :
Ms. Dinithi Pandithage



Co-Supervisor :
Dr. Sanika Wijayasekara



Widanage W.T.N
IT21377426



Bandara H.K.K.T
IT21176074



S.M.T.S Senaratna
IT21337512



Muhandiramge M.D.A.D
IT21383434

Introduction

- Disasters disrupt communication networks, complicating rescue efforts. To solve this, our project creates a resilient system using ad-hoc networks.
- Portable sensors are deployed in disaster zones to detect RF signals, revealing the presence of people. This data is transmitted to a central server via an ad-hoc network, analyzed, and sent to the Disaster Management Center, which informs rescue teams for faster, more efficient operations.
- This system ensures continuous information flow, improving resource allocation in disaster response.

Research Problem

How can we create a reliable system that detects and relays the presence of people in disaster areas, while also transmitting SOS alerts and instructions when traditional networks fail?

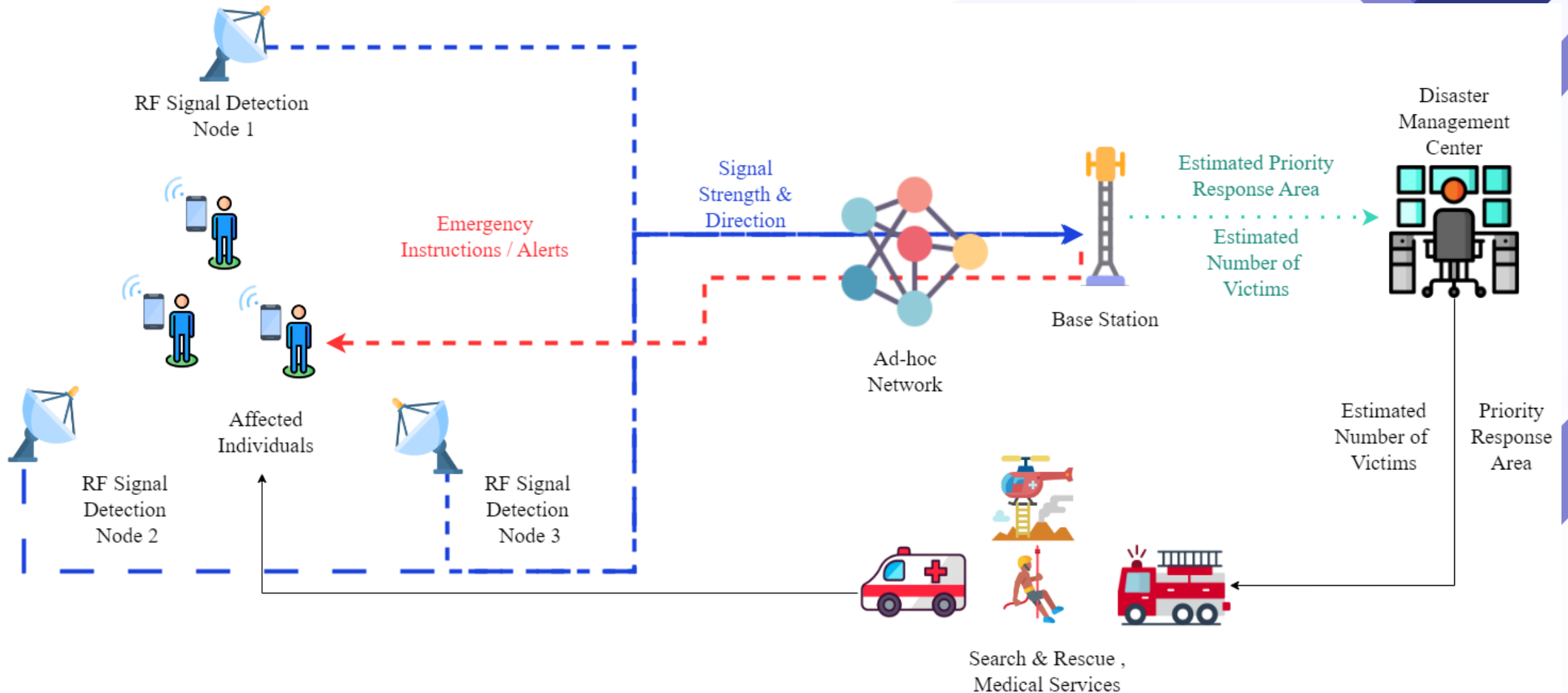
Main Objective:

To develop a robust communication system using multiple RF signal detection devices and an ad-hoc network, enhancing disaster response by accurately identifying the presence of people in disaster areas where traditional networks have failed.

Sub-Objectives:

- **To develop a reliable ad-hoc network for transmitting collected data to a central Management station.**
 - Create a robust ad-hoc network that efficiently transmits the collected RF data from multiple devices to a central base station for analysis.
- **To Design an SOS Messaging System:**
 - Create a system to facilitate efficient SOS messaging via an ad-hoc network with a focus on low latency and high reliability during emergency situations.
- **To Implement a System to Estimate the Affected Individual Count**
 - Develop mechanisms to collect data emitted from smartphones.

Overall System Diagram





IT21176074 | Bandara H. K. K. T.

Computer System and Network Engineering

INTRODUCTION

Background

- Reliable communication is critical during emergencies to coordinate rescue efforts and ensure safety.
- Ad-hoc networks enable dynamic, infrastructure-independent communication, ideal for disaster-hit areas.
- Can be quickly established and scaled using Wi-Fi-enabled devices, providing immediate, widespread coverage.
- Utilizes existing consumer devices, minimizing the need for specialized equipment.

Research Gap

Research Paper	Objectives & Tasks	
	To design and implement a dynamic ad-hoc network architecture.	To implement and test server within the network.
Quality of Sustainability Optimization Design for Mobile Ad Hoc Networks in Disaster Areas	✓	✗
Emergency Alert Networks for Disaster Management: Applications Perspective	✗	✗
A Novel Technique for Mobile Phone Localization for Search and Rescue Applications	✗	✓
Proposed System	✓	✓

Research Question

- How can a dynamic ad-hoc network architecture be designed and implemented to facilitate reliable emergency communication in disaster-affected areas without relying on existing infrastructure?

• **Main Objective**

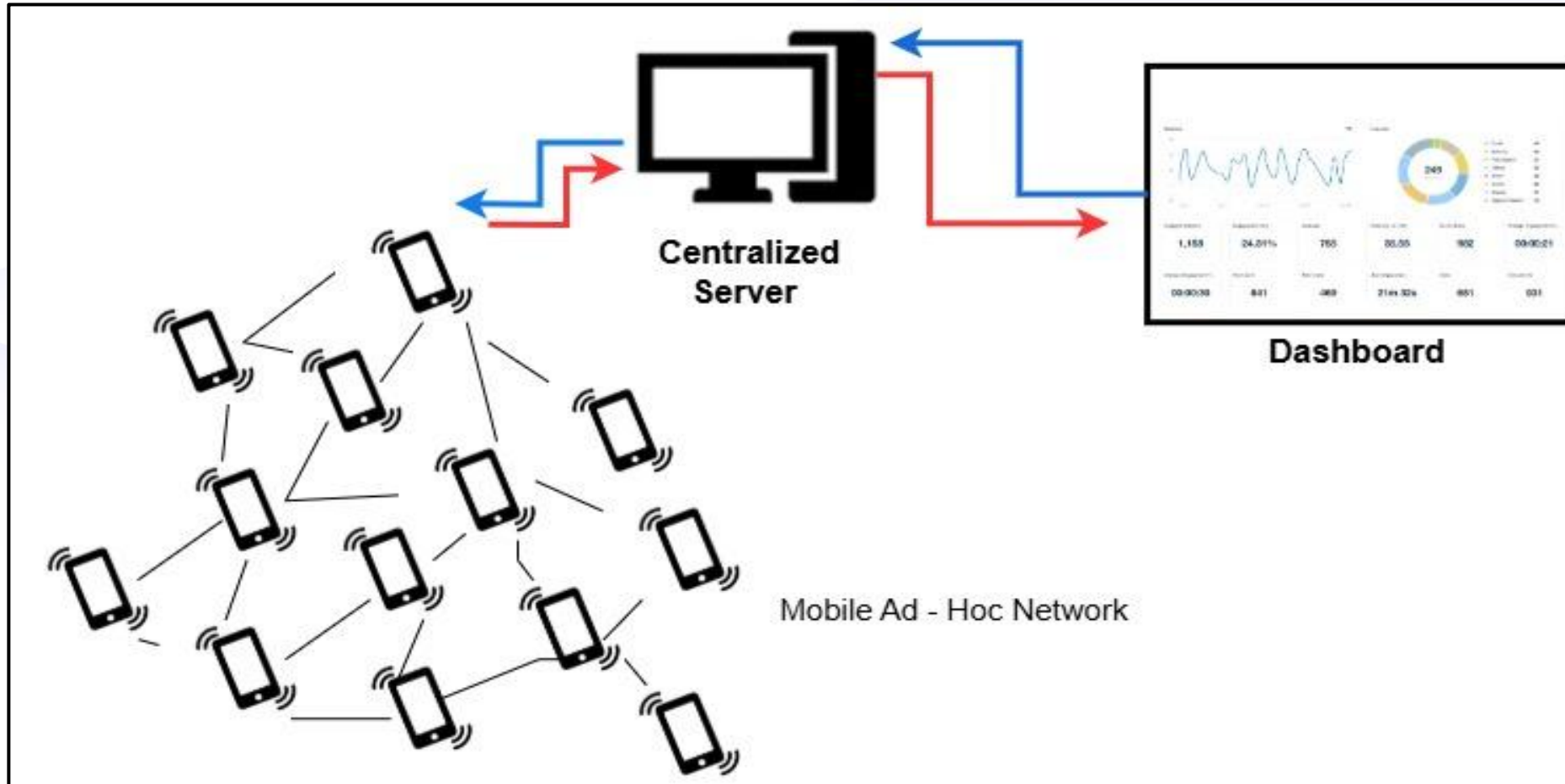
- To develop a disaster-resilient telecommunication infrastructure to ensure continuous communication during natural disasters.

• **Sub-Objectives:**

- Establish an ad-hoc network for emergency communication.
 - To design and implement a dynamic ad-hoc network architecture.
- Utilize Centralize Server to enhance the ad-hoc network.
 - To implement and test server within the network.

METHODOLOGY

System Diagram



Technologies to be used

- Ad-hoc Network Technologies:
 - Mesh networking, Mobile app (Wi-Fi Direct)
- Software Integration:
 - Middleware for seamless communication between mobile devices and the ad-hoc network.

Requirements

- Hardware
 - Wi-Fi Enabled Centralize Server.
 - Mobile devices
 - Wi-fi adapters
- Software
 - Custom Firmware

Completion of the project

- Created an Ad Hoc network using Wi-Fi Direct.
- Creating Ad-Hoc network Application

Need to be done

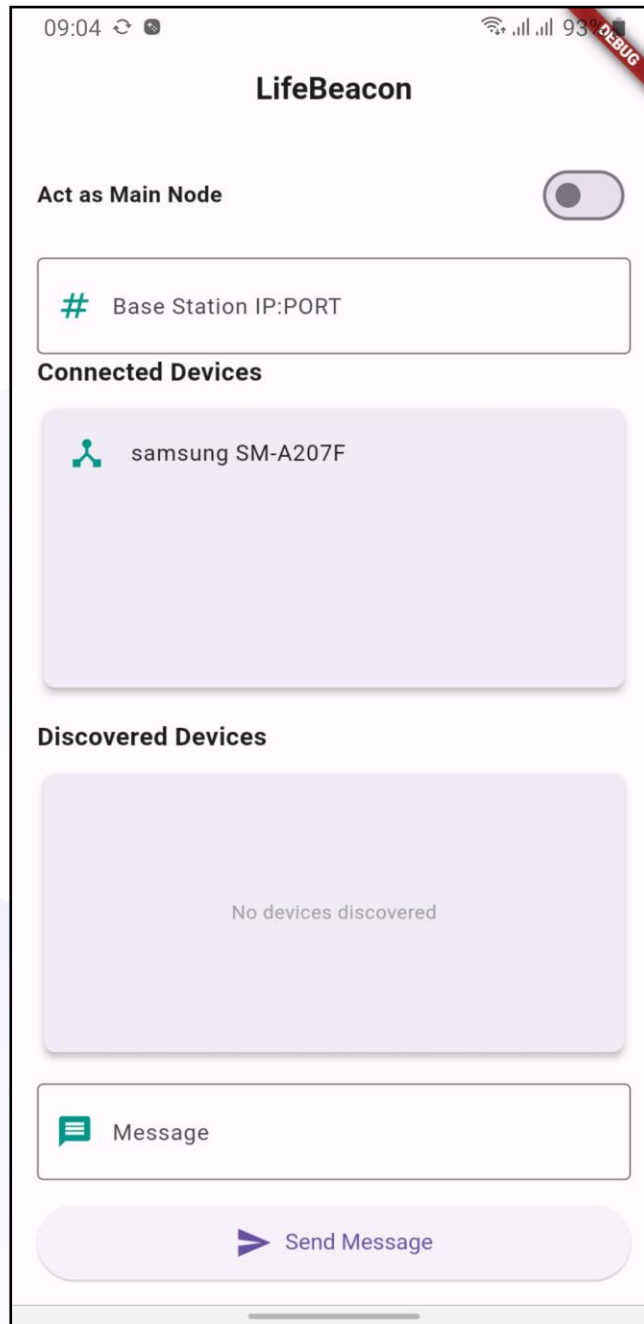
- Creating local database for all nodes.

Demonstration

```
73      @override
74      void initState() {
75          super.initState();
76          initNearbyService();
77          initNotifications();
78      }
79
80      void initNearbyService() {
81          nearbyService.init(
82              serviceType: 'mesh-network',
83              strategy: Strategy.P2P_CLUSTER,
84              callback: (isRunning) {
85                  if (isRunning) {
86                      // Start both advertising and discovering
87                      nearbyService.startAdvertisingPeer();
88                      nearbyService.startBrowsingForPeers();
89                  }
90              },
91          );
```

```
93     nearbyService.stateChangedSubscription(callback: (deviceList) {  
94         for (var element in deviceList) {  
95             switch (element.state) {  
96                 case SessionState.connected:  
97                     setState(() {  
98                         if (!connectedDevices  
99                             .any((device) => device.deviceId == element.deviceId)) {  
100                             connectedDevices.add(element);  
101                         }  
102                     });  
103                     break;
```

```
110     case SessionState.notConnected:
111         // Automatically connect to found devices
112         if (!connectedDevices.any((d) => d.deviceId == element.deviceId)) {
113             nearbyService.invitePeer(
114                 deviceId: element.deviceId, deviceName: element.deviceName);
115         }
116         setState(() {
117             if (!connectedDevices
118                 .any((device) => device.deviceId == element.deviceId)) {
119                 connectedDevices.add(element);
120             }
121         });
122         break;
123     default:
124         break;
125 }
126 }
127 });
128
```



Gantt Chart

Process	Months											
	May	June	July	August	September	October	November	December	Janurary	February	March	April
Requirement Gathering & Initial Planning												
Network Design												
Middlewre Development & Hardware Setup												
Integration & Testing												
Final Deployment												
Project Review & Documentation												
Final Presentation												



IT21383434 | MUHANDIRAMGE M.D.A.D

Computer System and Network Engineering

BACKGROUND

- The function aims to create an ad-hoc network using victims' mobile devices.
- Bluetooth Low Energy (BLE) is the one of main technology used to build the network.
- The system includes a self-healing mechanism that automatically rebuilds the network if a node disconnects.
- It ensures reliable communication in disaster areas where traditional networks are unavailable.

RESEARCH GAP

Research Paper	Objectives & Tasks		
	Implementing energy-efficient data transmission techniques.	Real-Time Data Transmission for Disaster Management	Self-healing and BLE-based communication
New Ordered Policy Routing Protocol for Active Data Transmission in Mobile Ad-hoc Networks	✗	✗	✗
Fairness Improvement and Efficient Rerouting in Mobile Ad Hoc Networks	✗	✗	✗
A Novel Technique for Mobile Phone Localization for Search and Rescue Applications	✗	✓	✗
Proposed System	✓	✓	✓

Research Question

- How can an ad-hoc network be effectively created using Bluetooth Low Energy (BLE) and mobile devices in disaster-affected areas to ensure reliable communication?
- What self-healing mechanisms can be implemented to automatically restore and maintain network connectivity in ad-hoc networks when a connected node disconnects during a disaster?

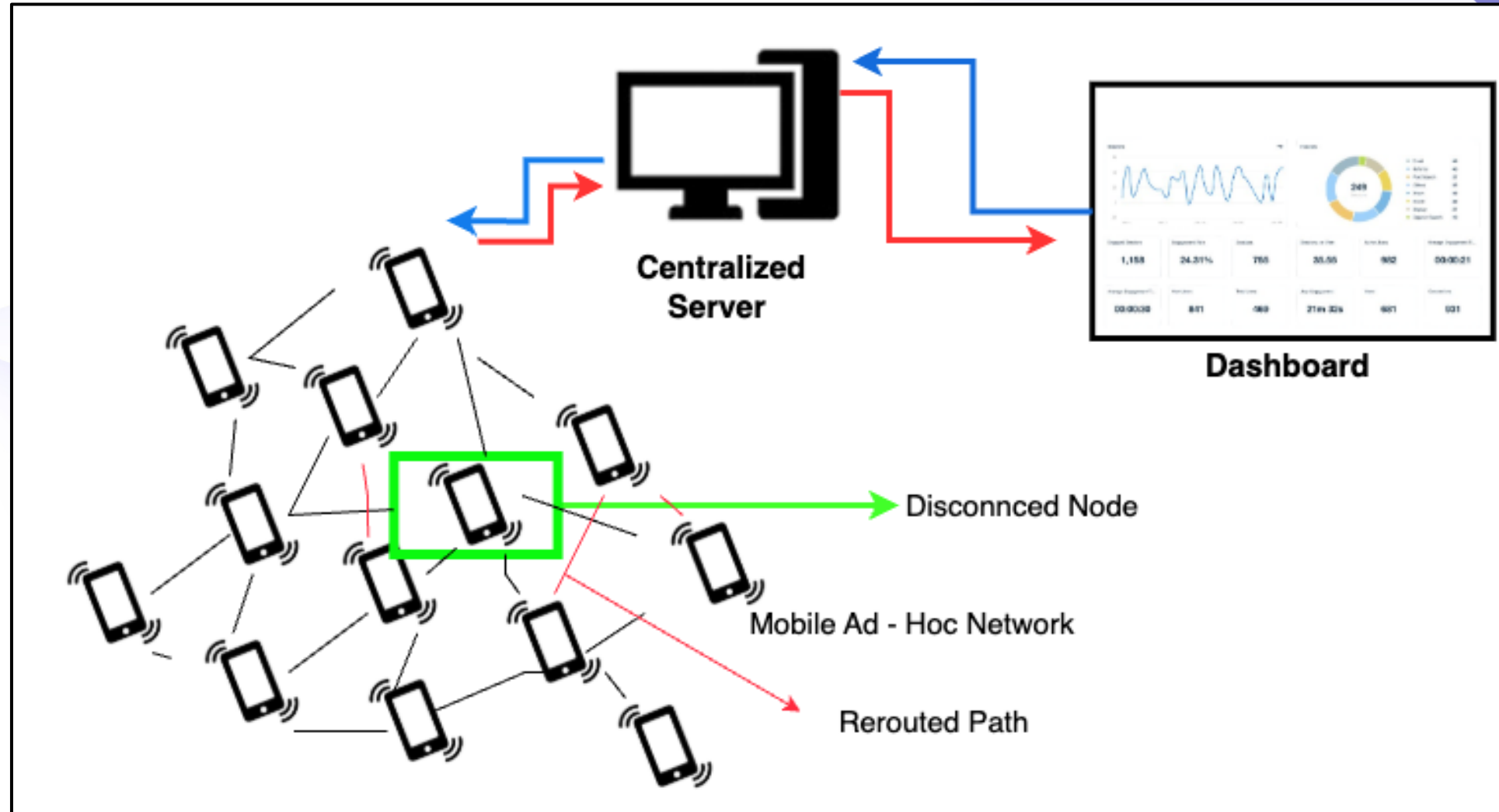
MAIN OBJECTIVES

- To create a reliable ad-hoc network using Bluetooth Low Energy (BLE) and mobile devices, ensuring continuous communication and recreating the network when a node disconnects in disaster-affected areas.

SUB OBJECTIVES

- Build an ad-hoc network using mobile devices as servers and clients with BLE technology.
- Implement a self-healing mechanism to rebuild the network when a node disconnects.
- Optimize BLE for low-power, efficient communication in the network.
- Test and validate the network's reliability and self-healing function in disaster scenarios

METHODOLOGY – SYSTEM DIAGRAM



METHODOLOGY

- Create an ad-hoc network using mobile devices in the disaster area with Bluetooth Low Energy (BLE).
- Ensure devices communicate efficiently within the network.
- Implement a self-healing mechanism to reconnect devices if a node disconnects.
- Rebuild the network automatically when a node leaves or joins. Enable dynamic device management to maintain continuous network operation.
- Test the system to ensure reliable performance in disaster scenarios.

METHODOLOGY - TECHNOLOGIES TO BE USED

1. Bluetooth Low Energy (BLE)
2. Self-Healing Algorithm

SYSTEM, PERSONAL, AND SOFTWARE REQUIREMENTS SPECIFICATION

System Requirements

- Centralized server:
- Ad-Hoc Network Components:

Personnel Requirements

- System Engineers
- Network Engineers
- Data Analysts

Software Requirement

- Communication Protocol Software
- Software development tools

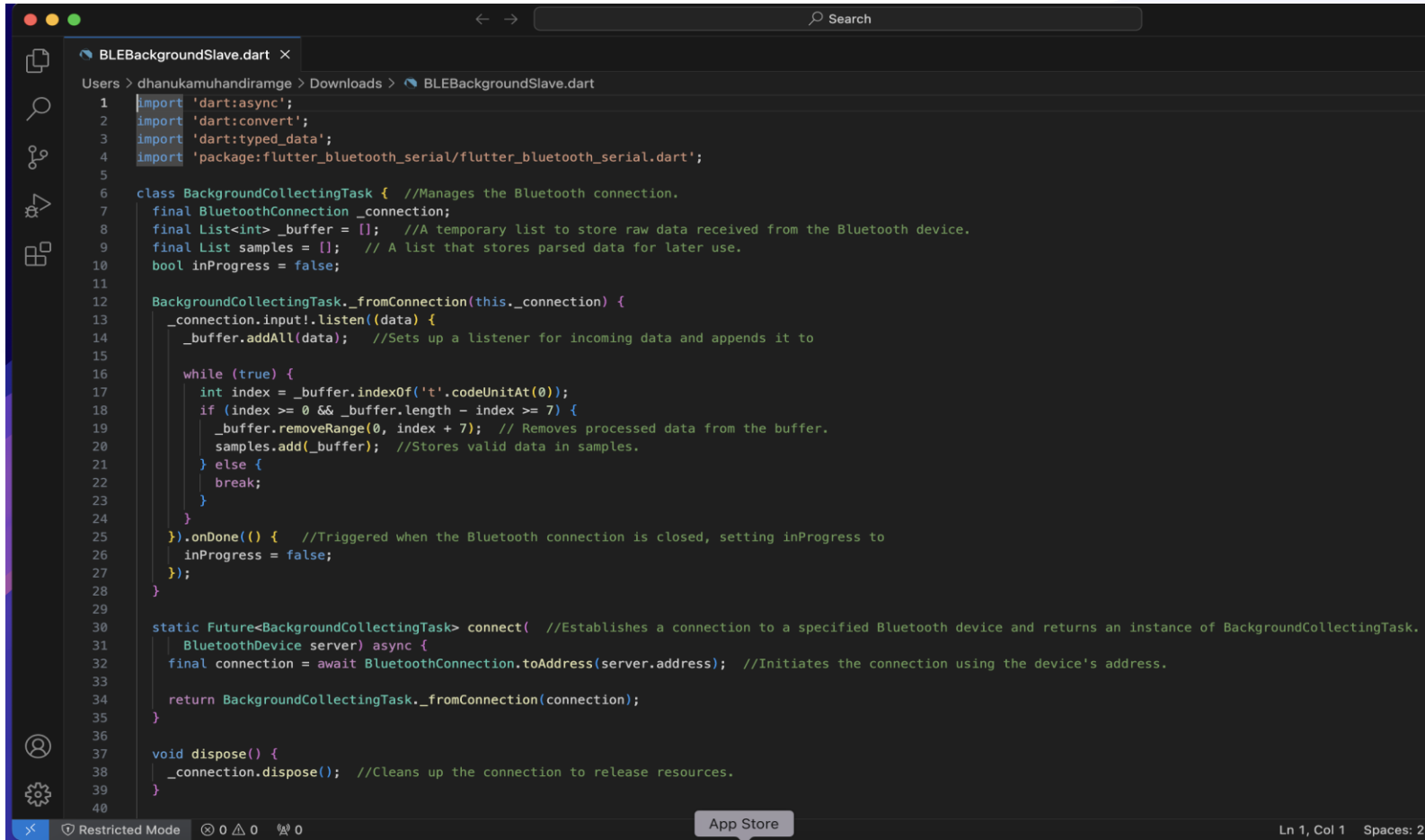
Completion of the project

- Created an Ad Hoc network using Bluetooth Low Energy .
- Creating Ad-Hoc network Application

Need to be done

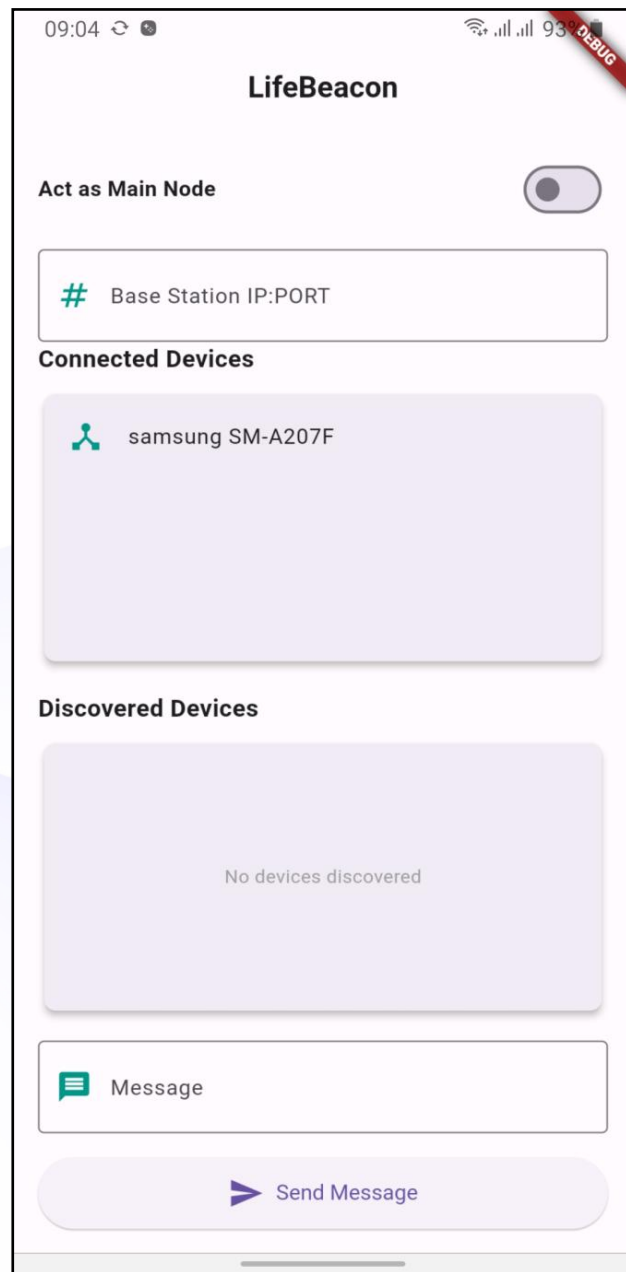
- Implementing self-healing mechanism that automatically rebuilds the network if a node disconnects.

Demonstration



```
1 import 'dart:async';
2 import 'dart:convert';
3 import 'dart:typed_data';
4 import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';
5
6
7 class BackgroundCollectingTask { //Manages the Bluetooth connection.
8   final BluetoothConnection _connection;
9   final List<int> _buffer = []; //A temporary list to store raw data received from the Bluetooth device.
10  final List samples = []; // A list that stores parsed data for later use.
11  bool inProgress = false;
12
13  BackgroundCollectingTask._fromConnection(this._connection) {
14    _connection.input!.listen((data) {
15      _buffer.addAll(data); //Sets up a listener for incoming data and appends it to
16
17      while (true) {
18        int index = _buffer.indexOf('t'.codeUnitAt(0));
19        if (index >= 0 && _buffer.length - index >= 7) {
20          _buffer.removeRange(0, index + 7); // Removes processed data from the buffer.
21          samples.add(_buffer); //Stores valid data in samples.
22        } else {
23          break;
24        }
25      }
26    }).onDone(() { //Triggered when the Bluetooth connection is closed, setting inProgress to
27      inProgress = false;
28    });
29  }
30
31  static Future<BackgroundCollectingTask> connect( //Establishes a connection to a specified Bluetooth device and returns an instance of BackgroundCollectingTask.
32    BluetoothDevice server) async {
33    final connection = await BluetoothConnection.toAddress(server.address); //Initiates the connection using the device's address.
34
35    return BackgroundCollectingTask._fromConnection(connection);
36  }
37
38  void dispose() {
39    _connection.dispose(); //Cleans up the connection to release resources.
40  }
41 }
```

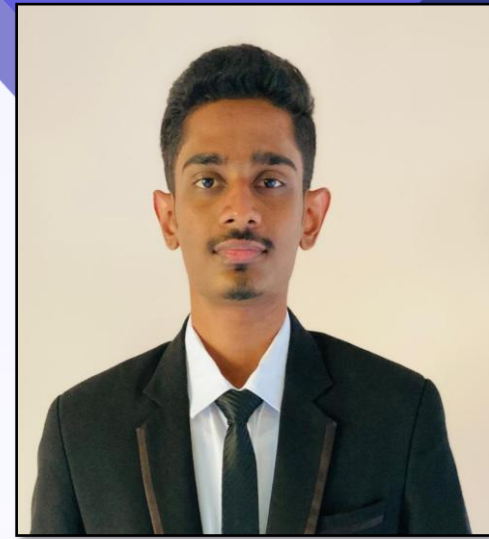
Main code is used to establish connection using BLE



Interface of the mobile application

GANTT CHART

Process	Months											
	May	June	July	August	September	October	November	December	Janurary	February	March	April
Requirement Gathering & Initial Planning												
Network Design												
Middlewre Development & Hardware Setup												
Integration & Testing												
Final Deployment												
Project Review & Documentation												
Final Presentation												



IT21377426 | Widanage W. T. N.

Computer Systems and Network Engineering

BACKGROUND

- Disasters can severely damage traditional communication infrastructure, leaving affected areas without reliable means of communication.
- Ad-hoc networks are crucial in these scenarios as they can form spontaneously and operate independently of existing infrastructure, maintaining communication links.
- In disaster situations, the ability to send and receive SOS messages is critical for coordinating emergency response and ensuring the safety of individuals.

- My role focuses on design SOS messaging system to enable the transmission of SOS messages within the ad-hoc network.
- The messaging system is designed to ensure low latency and high reliability in delivering SOS messages, tailored specifically to the dynamic nature of ad-hoc networks.^[1]

Research Gap

Research Paper	Objectives & Tasks			
	Creating SOS messaging system for Ad-Hoc Networks	Focus on Low Latency in SOS Message Delivery	Enhancing Reliability in SOS Message Delivery	Integration with Mobile Devices
Wireless Ad Hoc Networking: The Art of Networking Without a Network	✗	✗	✗	✗
Routing Mechanisms in Ad Hoc Networks	✓	✗	✗	✗
Building a Disaster Rescue Platform Utilizing Ad-Hoc Networks	✓	✓	✓	✗
Proposed System	✓	✓	✓	✓

Research Question

- How can the existing SOS messaging protocol be redesigned to better integrate with mobile devices and other technologies used in disaster response scenarios, ensuring seamless communication?^[2]

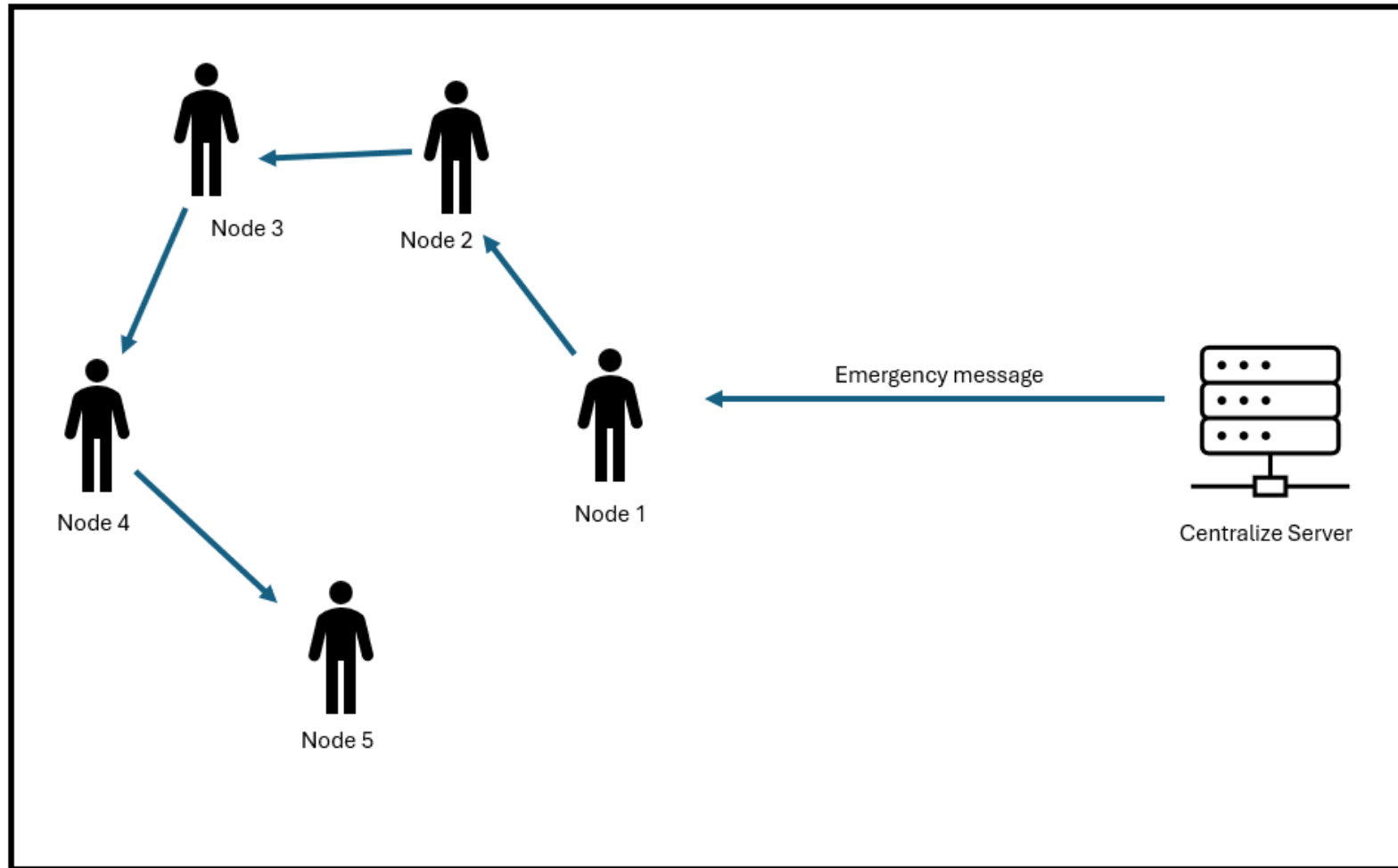
Main Objective

- To develop and enhance a dedicated SOS messaging system that ensures low latency and high reliability for communication within ad-hoc networks during disaster situations.

Sub Objectives

- To identify the limitations of the existing SOS messaging system, particularly in terms of latency and reliability within ad-hoc networks.
- To develop specific modifications to the SOS messaging system that address the identified limitations and optimize performance in dynamic, infrastructure-less environments.
- To adapt the enhanced protocol for seamless integration with mobile devices and other technologies used by rescue teams, ensuring effective communication during disaster response efforts.

System Diagram



Methodology– Technologies to be Used

- **Flask (Python):**

- Acts as the server-side framework to handle messaging and network connections.
- Enables a WebSocket-based real-time communication system via Flask-SocketIO.

- **Flutter:**

- Integrates the messaging functionality into the mobile app.
- Manages connections using **flutter_nearby_connections** for peer-to-peer messaging.

- **Flutter Nearby Connections Plugin:**

- Implements P2P communication in the mobile network using the **P2P_CLUSTER** strategy.

- **Socket.IO:**

- Facilitates WebSocket connections between the server and mobile devices for real-time message broadcasting.

- **Notification Service:**

- **Flutter Local Notifications** for alerting users about received messages.
- **Raw Resource Sounds** for high-priority alerts in critical situations.

- **Protocols Used:**

- **WebSocket Protocol:**

- Ensures real-time communication between the server and connected devices with minimal latency.

- **Bluetooth Protocol:**

- Enables direct device-to-device communication when Wi-Fi is unavailable.

- **Wi-Fi Direct:**

- Facilitates faster device discovery and data transfer in the ad-hoc network.

- **Implementation Steps:**

- **Server-Side Messaging:**

- Use **Flask-SocketIO** to establish WebSocket connections.
- Broadcast messages to all connected nodes using **emit()** functionality.

- **Mobile App Messaging:**

- Set up peer-to-peer connections with the **flutter_nearby_connections** plugin.
- Receive and display messages from other devices or servers.

- **Real-Time Communication:**

- The server and app work together to relay messages in real-time, even in an unstable or disaster-affected network.

Completion of the project

- Implement a messaging system

Need to be done

- Implement encryption for a targeted messaging system

Progress

- Sending messages to connected peers.

```
188 void sendMessage(String message) {  
189     for (Device device in connectedDevices) {  
190         nearbyService.sendMessage(device.deviceId, message);  
191     }  
192     print('Message Sent: $message');  
193 }
```


- Receiving and processing messages from peers.

```
163     nearbyService.dataReceivedSubscription(callback: (data) {  
164         _showNotification(data['message']);  
165         showDialog(  
166             context: context,  
167             builder: (context) => AlertDialog(  
168                 title: Text('Message Received'),  
169                 content: Text(data['message']),  
170                 actions: [  
171                     TextButton(  
172                         onPressed: () => Navigator.pop(context),  
173                         child: Text('OK'),  
174                     ), // TextButton  
175                 ],  
176             ), // AlertDialog  
177         );  
178     });  
179 }
```

- Real-time messaging integration with a server.

```
30 void startListeningToSocketServer() {
31     print("Listening to sock");
32     IO.Socket socket = IO.io(
33         'http://192.168.43.210:12345',
34         IO.OptionBuilder()
35             .setTransports(['websocket'])
36             .disableAutoConnect()
37             .build(),
38     );
39
40     socket.connect();
41
42     socket.onConnect(()) {
43         print('Connected to server');
44     });
45
46     socket.on('message', (data) {
47         print('Message received from server: $data');
48         sendMessage(data.toString());
49     });
50
51     socket.onDisconnect(()) {
52         print('Disconnected from server');
53     });
54 }
55
56 void stopListeningToSocketServer() {
57     print("Stopped listening to the socket server...");
58 }
```

- User alerts for incoming messages.

```
208 Future<void> _showNotification(String message) async {
209     const AndroidNotificationDetails androidNotificationDetails =
210         AndroidNotificationDetails(
211             'mesh_channel', // Channel ID
212             'Mesh Messages', // Channel Name
213             importance: Importance.high,
214             priority: Priority.high,
215             playSound: true,
216             sound: RawResourceAndroidNotificationSound('siren'),
217         ); // AndroidNotificationDetails
218     const NotificationDetails notificationDetails =
219         NotificationDetails(android: androidNotificationDetails);
220     await flutterLocalNotificationsPlugin.show(
221         0, // Notification ID
222         'Message Received', // Notification Title
223         message, // Notification Body
224         notificationDetails,
225     );
226 }
```

- Message input and send button.

```
330 // Message Input Field
331 TextField(
332   controller: messageController,
333   decoration: InputDecoration(
334     labelText: 'Message',
335     border: OutlineInputBorder(),
336     prefixIcon: Icon(Icons.message, color: Colors.teal),
337   ), // InputDecoration
338 ), // TextField
339 SizedBox(height: 16),
340 // Send Message Button
341 ElevatedButton.icon(
342   onPressed: () {
343     if (messageController.text.isNotEmpty) {
344       sendMessage(messageController.text);
345       messageController.clear();
346     }
347   },
348   icon: Icon(Icons.send),
349   label: Text('Send Message'),
350   style: ElevatedButton.styleFrom(
351     padding: EdgeInsets.symmetric(vertical: 12),
352     textStyle: TextStyle(fontSize: 16),
353   ),
354 ), // ElevatedButton.icon
```

Gantt chart

Process											
	June	July	August	September	October	November	December	January	February	March	April
Literature Review											
Requirement Analysis											
Protocol Design											
Simulation and Testing Setup											
Initial Testing and Refinement											
Testing											
Data Analysis and Optimization											
Final Report and Presentation Preparation											



IT21337512 | Senaratna S.M.T.S

Computer Systems & Network Engineering

Research Question

- How can RF technologies be utilized to collect data & estimate the number of individuals in a disaster-affected areas effectively?

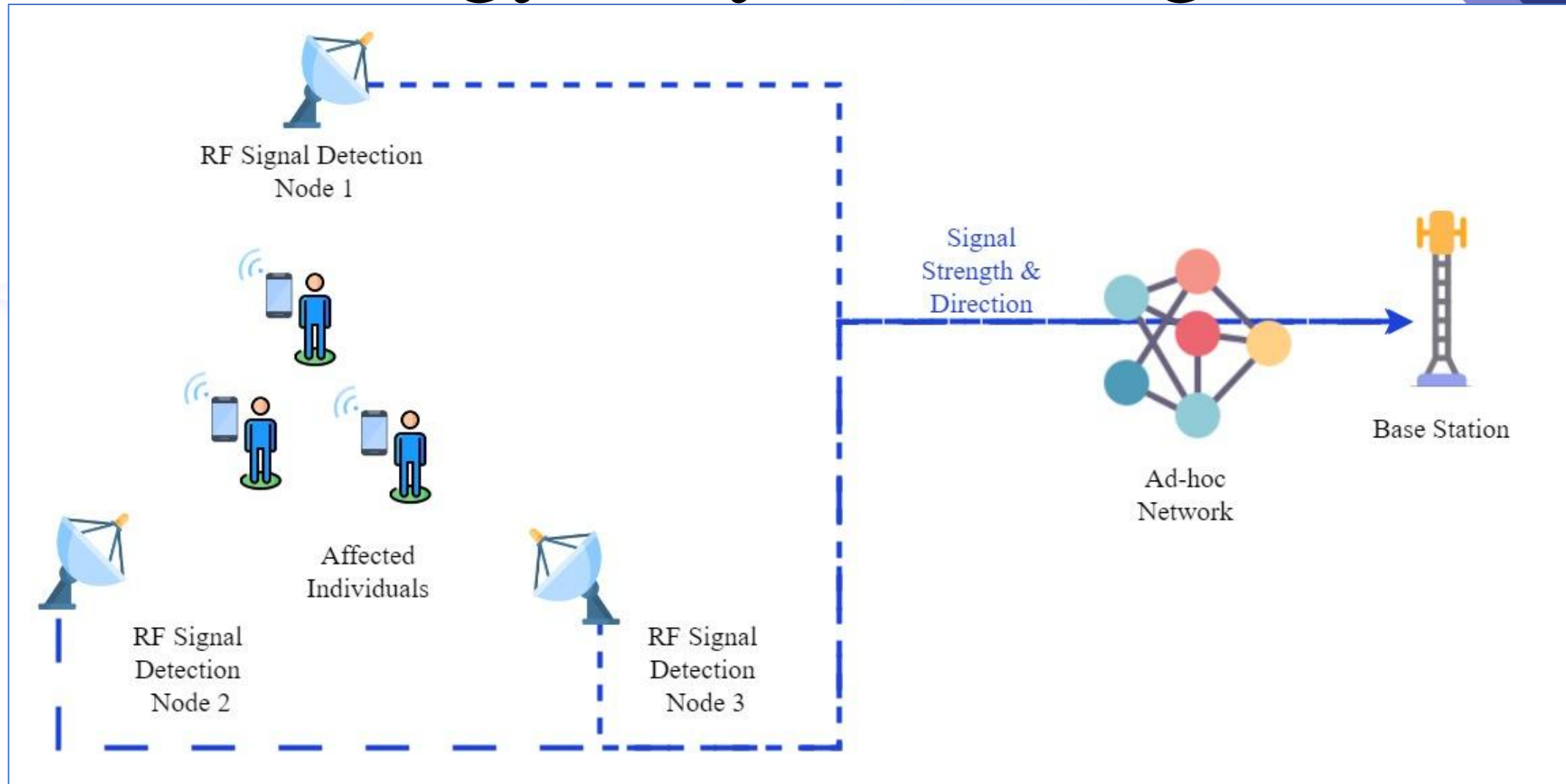
- Objective

- To develop a system that uses RF technologies to collect data and estimate the number of people in disaster-affected areas.

- Sub-objectives

- To Implement a Data Collection System:
 - Develop mechanisms to collect data using RF signals emitted from smartphones and compatible devices. Utilize Wi-Fi or Bluetooth technology to gather data from devices within proximity.
- To Filter Data for Accuracy:
 - Implement filtering techniques to ensure data accuracy and eliminate redundancies or false positives in estimating the number of affected individuals.
- To Evaluate the System and Optimize:
 - Evaluate the system's performance in real-world simulations or pilot projects. Optimize the system based on feedback and results to improve data collection accuracy.

Methodology – System Diagram



Progress as of Now

- 1. Detection Technology Comparison & Selection – Completed**
- 2. Embedded System / Microcontroller Selection, Hardware Acquisition & Implementation**
 1. Selection of Embedded System (Microcontroller) – Completed
 2. Hardware Acquisition – Completed
 3. Final Hardware Implementation – Pending
- 3. Detection Device Programming**
 1. Device Detection – Completed
 2. Data Filtering – In Progress (75% Completed)
 3. Location Approximation (Signal Triangulation Algorithm) – Pending
- 4. Enclosure Design & Implementation – Pending**

1. Detection Technology Comparison & Selection

- **GSM Signal Capturing:**

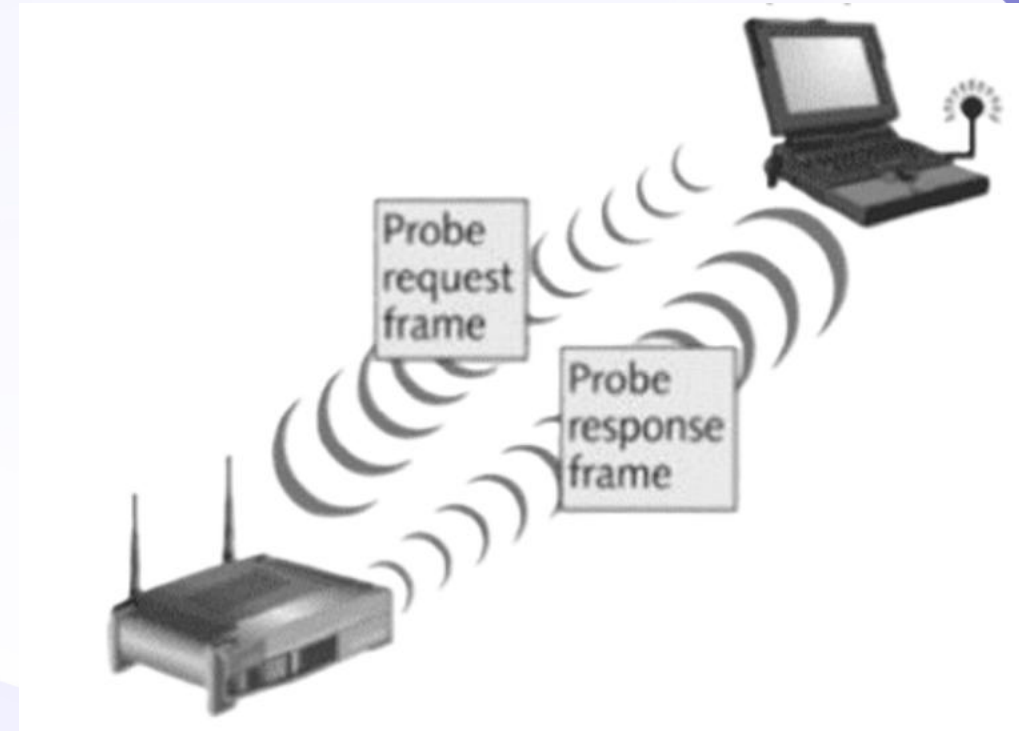
- **Advantages:** Could detect devices connected to GSM networks.
- **Challenges:**
 - Requires use of a **Jammer** to disrupt existing connections, which can be highly disruptive.
 - Complex implementation due to mimicking GSM network security features.
 - High cost and complexity of required hardware.
 - Legal challenges in obtaining and deploying GSM-capturing & jamming hardware.

- **Wi-Fi Probe Request Capturing (Selected):**

- **Advantages:**
 - No disruption to existing networks.
 - Ability to leverage promiscuous mode for non-intrusive data collection.
 - Allows capturing of Device MAC Address for unique identification.
 - Cost-effective and simpler implementation.
 - Minimal legal or regulatory restrictions.

Understanding Wi-Fi Probe Requests

- A Wi-Fi probe request is a type of management frame defined in the IEEE 802.11 standard.
- It is sent by Wi-Fi clients to actively search for available networks.
- These requests are transmitted even when no network is present, as the device is scanning for networks to join.



Wi-Fi Probe Requests

- **Key Features:**

- Broadcasted: Probe requests are sent publicly and can be captured by any device in range.
- Contains Metadata:
 - MAC Address: Unique identifier for the device.
 - SSID: Network name (if the device is searching for a specific one).

- **Relevance to the Project:**

- Enables real-time tracking of devices in disaster-affected areas.
- Provides non-disruptive data collection without affecting existing network operations.
- The MAC Address allows for unique identification of devices, crucial for rescue and recovery efforts.

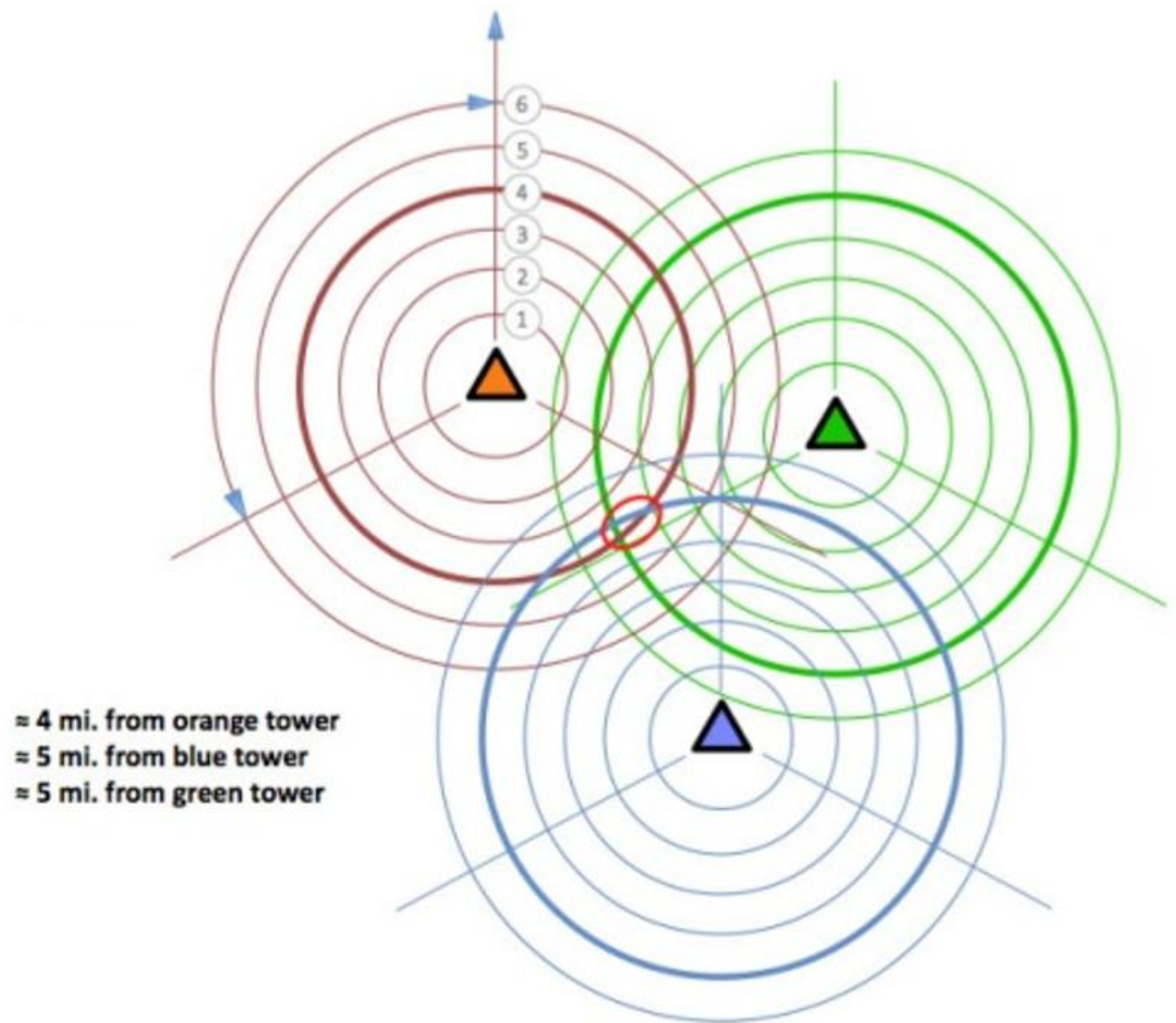
2. Embedded System / Microcontroller Selection, Hardware Acquisition & Implementation

- Selected Device : Espressif Systems 32-bit Microcontroller : (ESP32-WROOM-32U)
- **Why ESP32?**
 - Built-in Wi-Fi capabilities.
 - ESP32-WROOM-32U model allows the integration of external antennas for longer range and accuracy.
 - Supports promiscuous mode for packet sniffing.
 - Efficient power consumption allows for more portable & user-friendly device implementation.
 - Compact, cost-effective, and suitable for portable applications.

3. Detection Device Programming

- Wi-Fi Probe Request Capturing:
 - Operates in promiscuous mode to sniff packets.
 - Captures and processes metadata like:
 - RSSI (Signal Strength)
 - Channel
 - MAC Address
 - SSID (if available).
- Features:
 - Channel Hopping: Scans channels (1–13) for wider coverage.
 - Packet Processing: Filters for probe requests and extracts relevant metadata.
- Data Filtering:
 - Eliminates redundancies and false positives.
 - Filters out incomplete packets and corrupted packets to ensure accurate data.
 - Ensures accuracy for estimating individual counts in disaster scenarios.

Location Approximation (Device Triangulation Algorithm)



4. Enclosure Design & Implementation

- **Procured Hardware:**

1. ESP32 microcontrollers. (3 Nos)
2. External antennas for extended detection range. (3 Nos)
3. Voltage regulator / Li-ion battery charger for stable power supply.
4. Lithium-ion batteries for portability.
5. Enclosure
6. Display

Demonstration

```
10:30:56.026 -> RSSI: -39 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: Test 2
10:30:56.026 -> RSSI: -39 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: Test3SSID
10:30:56.069 -> RSSI: -39 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID:
10:30:56.069 -> RSSI: -38 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: SLT fiber
10:30:56.069 -> RSSI: -37 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: SLT fiber
10:30:56.069 -> RSSI: -37 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: Test 2
10:30:56.069 -> RSSI: -38 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: Test3SSID
10:30:56.214 -> RSSI: -69 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID:
10:30:56.214 -> RSSI: -78 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: SLT fiber
10:30:56.214 -> RSSI: -68 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID:
10:30:56.214 -> RSSI: -68 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID:
10:30:56.245 -> RSSI: -71 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: SLT fiber
10:30:56.245 -> RSSI: -72 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: Test3SSID
10:30:56.245 -> RSSI: -77 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID:
10:30:56.245 -> RSSI: -70 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID:
10:30:56.245 -> RSSI: -70 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: Test 2
10:30:56.279 -> RSSI: -79 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: Test3SSID
10:30:56.279 -> RSSI: -69 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: Test 2
10:30:56.279 -> RSSI: -69 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID:
10:30:56.320 -> RSSI: -68 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: SLT fiber
10:30:56.320 -> RSSI: -84 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID: Test3SSID
10:30:56.320 -> RSSI: -85 Ch: 8 Peer MAC: 7a:20:82:c7:54:d4 SSID:
10:31:01.983 -> RSSI: -69 Ch: 1 Peer MAC: 00:00:00:00:01:00 SSID:
10:31:01.983 -> RSSI: -28 Ch: 1 Peer MAC: 4e:eb:d0:8f:de:95 SSID:
10:31:02.015 -> RSSI: -64 Ch: 1 Peer MAC: a8:80:55:21:f3:c6 SSID:
10:31:02.196 -> RSSI: -67 Ch: 1 Peer MAC: 4e:eb:d0:8f:de:95 SSID:
10:31:02.230 -> RSSI: -27 Ch: 1 Peer MAC: a8:80:55:21:f3:c6 SSID:
10:31:02.680 -> RSSI: -64 Ch: 1 Peer MAC: c2:86:0f:82:a4:b4 SSID:
10:31:02.772 -> RSSI: -68 Ch: 1 Peer MAC: a8:80:55:21:e3:15 SSID:
10:31:02.805 -> RSSI: -34 Ch: 1 Peer MAC: 9e:56:ab:6c:86:bb SSID:
```

SSID

Device MAC Address

Detected Wi-Fi Channel

Signal Strength

Timestamp

Thank You !