# MAN IN THE MIDDLE ATTACK ON PANIC BUTTONS THAT WORKS IN GSM NETWORKS: VULNERABILITIES AND DETECTION

**A PROJECT REPORT**

***Submitted by***

**ARSHAD T A**

**(210181601008)**

**SANJAY K**

**(210181601045)**

**Under the guidance of**

**Mrs. SOUNDARIYA A**

***in partial fulfilment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

**(CYBER SECURITY)**



**APRIL 2025**



**BONAFIDE CERTIFICATE**

Certified that this project report **“MAN IN THE MIDDLE ATTACK ON PANIC BUTTONS THAT WORKS IN GSM NETWORKS: VULNERABILITIES AND DETECTION”** is the bonafide work of **“ARSHAD T A (210181601008), SANJAY K (210181601045) “** who carried out the project work under my supervision. Certified further, that to the best of our knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

***SIGNATURE* *SIGNATURE***

**Mrs. SOUNDARIYA A Dr. AISHA BANU W**

**SUPERVISOR HEAD OF THE DEPARTMENT**

Assistant Professor Professor

Department of CSE Department of CSE

B. S. Abdur Rahman Crescent B. S. Abdur Rahman Crescent

Institute of Science & Technology Institute of Science & Technology

Vandalur, Chennai – 600 048 Vandalur, Chennai – 600 048



**VIVA VOICE EXAMINATION**

The viva voice examination of the **CSD 4201** project work titled **“MAN IN THE MIDDLE ATTACK ON PANIC BUTTONS THAT WORKS IN GSM NETWORKS: VULNERABILITIES AND DETECTION”**, submitted by **ARSHAD T A (210181601008), SANJAY K (210181601045)** is held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We sincerely express our heartfelt gratitude to **Prof. Dr. T. MURUGESAN**, Vice chancellor and **Dr. N. THAJUDDIN**, Pro-Vice Chancellor, B.S. Abdur Rahman Crescent Institute of Science and Technology, for providing us an environment to carry out our project successfully.

We sincerely thank **Dr. N. RAJA HUSSAIN**, Registrar, for furnishing every essential facility for doing our project.

We thank **Dr. SHARMILA SANKAR**, Dean, School of Computer, Information and Mathematical Sciences for her motivation and support.

We thank **Dr. W. AISHA BANU**, Professor and Head, Department of Computer Science and Engineering, for providing strong oversight of vision, strategic direction and valuable suggestions.

We express our sincere thanks to the Project Review Committee members, from the Department of Computer Science and Engineering, **Dr. N. Sabiyath Fatima**, Professor and **Dr. C. HEMA**, Associate Professor, **Dr. R. Akila,** Associate Professor for their valuable suggestions and support.

We obliged our project supervisor **Mrs.SOUNDARIYA A**, Assistant Professor, Department of Computer Science and Engineering for her professional guidance and continued assistance during our project.

We thank class advisor and project coordinator **Dr. C. Hema**, Associate Professor, Department of Computer Science and Engineering for her professional guidance and continued assistance throughout the project period.

We thank all the **Faculty members** and the **System Staff** of the Department of Computer Science and Engineering for their valuable support and assistance at various stages of project development.

**ARSHAD T A**

**SANJAY K**

**ABSTRACT**

This project investigates the detection and prevention of Man-in-the-Middle (MitM) attacks targeting panic buttons that operate over GSM (Global System for Mobile Communications) networks. Panic buttons, widely used in emergency situations, and vulnerable due to the inherent weaknesses of GSM protocols, such as outdated encryption and lack of end-to-end security.

The research begins by analysing GSM communication architecture and the functioning of panic button systems, highlighting the potential threats they face. It examines various attack vectors like signal spoofing, message manipulation, and rogue base station attacks. To counter these threats, the study explores detection methods, including anomaly-based intrusion detection systems (IDS), cryptographic improvements, and machine learning models capable of identifying malicious activity in real time.

Both theoretical and practical methodologies are employed. The theoretical component focuses on GSM vulnerabilities and encryption mechanisms, while the practical side involves simulating attacks using tools such as Wireshark and Aircrack-ng to analyse panic button behaviour under threat conditions. The aim is to develop an effective detection and mitigation strategy.

Additionally, the study evaluates existing GSM security measures and considers real- world cases of MitM attacks on emergency systems. Given the continued reliance on GSM for emergency communication in many regions, the research proposes a layered security framework combining AI, behavioural analysis, and policy-level recommendations to enhance system resilience.

The outcomes of this project will provide a foundation for future development of secure GSM-based emergency communication systems. It also encourages further academic research into the use of AI and SDR in mobile threat detection. The proposed models can be integrated into government or enterprise-level emergency response frameworks.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER**  **NO.** | **TITLE** | **PAGE NO.** |
| - | **ABSTRACT** | **v** |
| - | **TABLE OF CONTENTS** | **vi – viii** |
| - | **LIST OF ABBREVIATIONS** | **ix** |
| - | **LIST OF FIGURES** | **x – xi** |
| **1** | **INTRODUCTION** | **1 – 7** |
| 1.1 | OVERVIEW | 1 |
| 1.2 | DESCRIPTION | 2 |
| 1.3 | OBJECTIVE OF THE PROJECT | 3 - 5 |
| 1.4 | ORGANIZATION OF THE PROJECT | 5 - 7 |
| **2** | **LITERATURE SURVEY** | **8 - 9** |
| **3** | **SYSTEM REQUIREMENTS AND DESIGN** | **10 - 25** |
| 3.1 | PROBLEM DEFINITION | 10 |
| 3.2 | EXISTING SYSTEM | 10 – 11 |
| 3.2.1 | LIMITATIONS OF EXISTING SYSTEM | 12 – 13 |
| 3.3 | PROPOSED SYSTEM | 13 – 14 |
| 3.3.1 | OBJECTIVES OF PROPOSED SYSTEM | 15 - 17 |
| 3.4 | MODULE IDENTIFICATION | 17 |
| 3.4.1 | PANIC BUTTON INTERFACE | 17 |
| 3.4.2 | ALERT TRANSMISSION OVER NETWORK | 18 |
| 3.4.3 | VULNERABILITY SIMULATION - MITM SETUP | 18 – 19 |
| 3.4.4 | DETECTION AND LOGGING MODULE | 19 |
| 3.4.5 | SECURITY ENHANCEMENT LAYER | 20 |
| 3.4.6 | VISUALIZATION AND OUTPUT | 20 – 21 |
| 3.5 | SYSTEM REQUIREMENTS | 21 – 22 |
| 3.6 | DESIGN PROCESS AND PROCESS EXPLANATION | 23 |
| 3.6.1 | ARCHITECTURE DIAGRAM | 23 – 24 |
| 3.6.2 | CIRCUIT DIAGRAM | 24 - 25 |
| **4** | **SYSTEM METHODOLOGIES** | **26 - 31** |
| 4.1 | MODULES EXPLANATION | 26 |
| 4.1.1 | PANIC BUTTON INTERFACE | 26 – 27 |
| 4.1.2 | ALERT TRANSMISSION OVER NETWORK | 27 - 28 |
| 4.1.3 | VULNERABILITY SIMULATION – MITM SETUP | 28 - 29 |
| 4.1.4 | DETECTION AND LOGGING MODULE | 29 |
| 4.1.5  4.1.6 | SECURITY ENHANCEMENT LAYER  VISUALIZATION AND OUTPUT MODULE | 30  30 - 31 |
| **5** | **IMPLEMENTATION** | **32 - 34** |
| **6** | **CONCLUSION AND FUTURE ENHANCEMENT** | **35 - 36** |
| **7** | **REFERENCES** | **37** |
| **8** | **APPENDIX** | **38 - 41** |
| 8.1 | A1 - SOURCE CODE | 38 |
| 8.1.1 | PANIC BUTTON | 38 – 41 |
| 8.1.2 | ANAMALOY DETECTION FOR MITM | 41 |
| 8.2 | A2 - SCREENSHOTS | 42 - 46 |
|  | **TECHNICAL BIOGRAPHY** | **47 - 48** |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **AI** | ARTIFICIAL INTELLIGENCE |
| **URL** | UNIFORM RESOURCE LOCATOR |
| **IOT** | INTERNET OF THINGS |
| **DP** | DATA PREPROCESSING |
| **DV** | DATA VIZUALIZATION |
| **NLP** | NATURAL LANGUAGE PROCESSING |
| **ML** | MACHINE LEARNING |
| **BC** | BAGGING CLASSIFIER |
| **RC** | RIDGE CLASSIFIER |
| **RFC** | RANDOM FOREST CLASSIFIER |
| **DTC** | DECISION TREE CLASSIFIER |
| **UA - PM** | USER AUTHENTICATION & PROFILE MANAGEMENT |
| **PMD** | PHISHING & MALWARE DETECTION |
| **IOT AC** | IOT ATTACK CLASSIFICATION |
| **DB** | DATABASE |
| **AR** | ADMIN & REPORT |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE.NO** |
| 3.1 | Architecture Diagram | 23 |
| 3.2 | Circuit Diagram | 24 |
| 8.1 | ESP Wi-Fi Connection Established | 42 |
| 8.2 | ESP8266 Node-MCU Microcontroller | 43 |
| 8.3 | Push Button | 44 |
| 8.4 | Transferring Result | 45 |
| 8.5 | Telegram Page | 46 |

**CHAPTER 1**

**INTRODUCTION**

* 1. **OVERVIEW**

This project focuses on securing panic button systems that rely on GSM networks for emergency communication. Due to outdated encryption protocols, GSM is vulnerable to Man-in-the-Middle (MitM) attacks. These attacks can intercept or manipulate distress signals, posing serious risks to public safety. The research aims to identify these vulnerabilities and propose effective detection and mitigation strategies. A combination of theoretical analysis and practical simulations is used to develop a robust security framework.

This project centers on the analysis, simulation, and mitigation of Man-in-the-Middle (MitM) attacks on panic button systems that operate over GSM (Global System for Mobile Communications) networks. Panic buttons are critical emergency communication tools used in personal safety devices, public transportation, and healthcare systems. However, their reliance on GSM, an outdated communication protocol lacking robust encryption, makes them highly susceptible to security breaches, particularly MitM attacks.

MitM attacks allow an adversary to intercept, delay, or modify communication between the panic button and the central emergency response system. Such interference can lead to severe consequences, including the failure to deliver distress signals or the delivery of altered messages, putting lives at risk.The objective of this research is to highlight the vulnerabilities in GSM-based panic button systems, understand the methods through which attackers exploit these weaknesses, and develop detection and prevention mechanisms. The study covers key attack techniques like signal spoofing, rogue base stations, and packet manipulation.

To address these challenges, both theoretical and practical methodologies are applied. The theoretical framework involves studying GSM architecture, encryption flaws, and known attack vectors. On the practical side, tools such as Wireshark and Aircrack-ng are used to simulate and analyze attacks in a controlled environment.

**1.2 DESCRIPTION**

This project focuses on identifying and mitigating Man-in-the-Middle (MITM) attacks on panic button systems that operate over GSM (Global System for Mobile Communications) networks. Panic buttons are vital components in emergency response frameworks, especially in environments such as public transport, women's safety systems, and healthcare monitoring. However, their dependence on GSM makes them highly vulnerable to security breaches. GSM, being an older mobile communication protocol, lacks modern cryptographic strength and mutual authentication mechanisms. These flaws open the door for attackers to impersonate legitimate towers (rogue BTS), intercept distress messages, or block signals altogether—leading to delays in emergency response and endangering lives.

The project begins by dissecting the GSM architecture and its use in panic button operation. It examines how distress signals are generated, transmitted, and received across GSM channels. By using tools like HackRF, Wireshark, and Aircrack-ng, we simulate real-time GSM environments and analyze how panic buttons behave when subjected to fake base stations or altered signal paths. The simulation helps understand the technical behavior of GSM-based communication under attack conditions, providing valuable data on response delays, message drops, and spoofed transmissions.

These include anomaly-based Intrusion Detection Systems (IDS), signal pattern recognition, and AI/ML models trained to detect rogue BTS behavior. Cryptographic enhancements and behavioral monitoring are proposed to prevent unauthorized tower connections. Through both theoretical analysis and hands-on implementation, the system’s vulnerabilities are exposed and countermeasures are tested for effectiveness.

The final output of this project is a proposed security framework specifically designed for GSM-based panic buttons. It includes real-time detection, alerting mechanisms, and policy-level suggestions for emergency system providers. The solution is scalable, cost- effective, and adaptable to both rural and urban deployments where GSM is still predominant. This work not only contributes to the security of emergency communication systems but also sets the groundwork for future enhancements involving AI, SDR testing environments, and integration with next-gen telecom protocols like LTE and 5G.

**1.3 OBJECTIVE OF THE PROJECT**

The primary objective of this project, titled “Man-in-the-Middle Attack on GSM Networks: Vulnerabilities and Detection,” is to delve into the security framework of the Global System for Mobile Communications (GSM) and examine how it is susceptible to Man-in- the-Middle (MITM) attacks. Despite GSM being a widely used and globally adopted mobile standard, its earlier specifications suffer from critical security flaws. These vulnerabilities allow adversaries to intercept, manipulate, or alter communication between mobile devices and network providers.

A core focus of the project is to understand the GSM network architecture, including key components such as the Base Transceiver Station (BTS), Base Station Controller (BSC), Mobile Switching Centre (MSC), and the Subscriber Identity Module (SIM). These building blocks form the operational foundation of GSM communication and are essential to identifying potential points of exploitation.

One of the major reasons GSM is inherently vulnerable is the lack of mutual authentication—the network authenticates the user, but the user does not authenticate the network. This opens the door for attackers to set up rogue base transceiver stations (fake BTS), tricking mobile devices into connecting under the false assumption of legitimacy.

Attackers often use Software Defined Radios like HackRF or USRP, alongside open-source projects like OsmocomBB or OpenBTS, to emulate a GSM base station and launch MITM attacks. These setups enable the interception of unencrypted or downgraded encrypted voice calls, SMS and occasionally Mobile data.

To tackle such threats, the project incorporates five robust detection modules, each designed to identify different signs of a MITM attack:

**1. Signal Anomaly Detection**

This module continuously monitors the characteristics of nearby GSM towers, including signal strength, Cell ID, and frequency.

How it works: It flags any abnormal behavior, such as sudden appearance of unknown towers, irregular signal strengths, or mismatched Cell IDs. These indicators often suggest the presence of a rogue BTS broadcasting fake signals to hijack mobile connections.

**2. Encryption & Authentication Monitoring**

MITM attackers frequently downgrade encryption or disable it completely to intercept communication.

How it works: This module observes the type of encryption being used (A5/1, A5/2, A5/0). It raises alerts when encryption is absent or suspiciously weak. It also verifies whether proper mutual authentication is occurring, helping to identify fake towers that skip secure handshakes.

**3. Latency & Round-Trip Time (RTT) Analysis**

Since rogue BTS systems often relay traffic to legitimate towers, they introduce measurable delays.

How it works: By tracking the round-trip time of packets between the mobile device and the network, this module detects unusual delays or jitter. Spikes in latency often indicate that the data is being intercepted or relayed, which is a common trait of MITM behavior.

**4. Machine Learning-Based Intrusion Detection**

This module utilizes pattern recognition to detect unusual or malicious network behavior.

How it works: It collects features such as tower-switching frequency, encryption status, signal anomalies, and RTT values. Trained ML models (e.g., decision trees, SVMs) then classify whether the behavior matches that of a normal network or a MITM scenario.

**5. IMSI Catcher Detection**

IMSI catchers are fake BTS setups designed to capture user identities by mimicking legitimate towers.

How it works: This module monitors broadcast messages and tower registration behavior. If a device is forced to re-register frequently or receives identity requests from unrecognized towers, the system raises red flags indicating potential IMSI catching attempts.

In conclusion, this project aims not only to identify and analyze the vulnerabilities within GSM that enable MITM attacks but also to simulate real-world attack scenarios for better understanding. Ultimately, the project emphasizes the urgent need for stronger authentication, encryption, and anomaly detection strategies in legacy mobile networks to protect user data and privacy.

**1.4 ORGANIZATION OF THE PROJECT**

**Chapter 1: Introduction**

This section introduces GSM technology and its importance in global mobile communications. It explains GSM's architecture and outlines its major security vulnerabilities, particularly the absence of mutual authentication. The concept of Man-in- the-Middle (MITM) attacks is introduced, highlighting their potential to intercept SMS, calls, and user data. The scope includes detection techniques and mitigation strategies for GSM vulnerabilities.This section also sets the stage for understanding how GSM security flaws can be exploited in real-world situations.

**Chapter 2: Literature Survey**

This section reviews previous studies related to GSM security, MITM attacks, and rogue BTS detection. It includes works by Biryukov, Nohl, and Engel, which highlight weaknesses in GSM’s A5 encryption and practical attack demonstrations using SDR tools.The survey covers IMSI catcher behavior and baseband layer vulnerabilities that make mobile devices susceptible to tracking and eavesdropping.It builds the technical background needed for this project and justifies the need for more effective detection and mitigation techniques in GSM.

**Chapter 3: System Requirements and Design**

This part lists the essential hardware (e.g., HackRF One, antennas, test phones) and software (e.g., OpenBTS, Wireshark, Kali Linux) needed to simulate and detect MITM attacks.The design involves setting up a rogue BTS, capturing communication, and analyzing traffic for anomalies.A flow diagram shows the interaction between attacker, victim device, and detection tool. This section ensures that the system setup is practical, cost-effective, and replicable.

**Chapter 4: System Methodologies**

This section details the step-by-step approach used in the project. It includes initializing SDR hardware, configuring OpenBTS, broadcasting fake network signals, and forcing devices to connect.Once connected, the attacker can intercept or manipulate communication. Detection tools like SnoopSnitch monitor baseband activity for signs of rogue BTS.This methodology also includes ethical boundaries and ensuring a controlled environment for safe testing and educational purposes.

**Chapter 5: Implementation**

The implementation phase involves setting up and testing the rogue BTS using SDR and GSM tools. Calls and SMS are intercepted, and traffic is analyzed using Wireshark and custom scripts.This phase also includes logs, captured packets, encryption status, and signal behaviors. Screenshots and practical demonstrations are recorded.This hands-on stage proves that GSM MITM attacks are feasible with minimal cost and highlights the need for stronger mobile security.

**Chapter 6: Conclusion and Future Enhancement consolidates**

The project concludes that GSM is vulnerable to MITM attacks due to outdated security protocols. The simulation demonstrated how rogue BTS can trick phones and intercept data.Detection tools showed promise but require better integration into mobile systems. Future enhancements include using machine learning for anomaly detection, expanding to 4G/5G, or creating a lightweight mobile app that warns users of fake towers in real time.It also encourages telecom providers to adopt mutual authentication and end-to-end encryption for better protection.

**Chapter 7: References**

It includes IEEE papers, journal articles, open-source documentation, and developer blogs related to GSM, SDR, MITM attacks, and detection mechanisms.

Sample references:

* Biryukov, Shamir & Wagner (2000) – “A5/1 Cipher Analysis”
* Karsten Nohl (2010) – “GSM Cracking” CCC Talk
* Tobias Engel (2014) – “IMSI Catchers” Presentation
* OpenBTS and OsmocomBB official documentation

Proper citation formats like IEEE or APA are followed as per the academic guidelines.

These references validate the research depth and authenticity of the project.

**Chapter 8: Appendix**

The appendix contains supporting materials that are too detailed for the main report body of the Man in the middle attack project.It includes screenshots of tools like Wireshark, OpenBTS CLI output, packet logs, code snippets for analysis scripts, and system configuration settings.Diagrams showing network setup, BTS communication flow, and baseband alerts are also added.Sample IMSI logs, encryption flag outputs, and rogue tower detection proofs are part of this section.The appendix enhances the report by providing technical clarity for the demonstration and serves as a guide for future implementations.

**CHAPTER 2**

**LITERATURE SURVEY**

1. “GSM Security and the A5/1 Cipher” – Biryukov, Shamir, Wagner (2000) This paper analyzes the A5/1 stream cipher used in GSM and demonstrates its vulnerability to time-memory trade-off attacks. It showed how encrypted calls and messages could be decrypted using precomputed tables, exposing serious flaws in GSM's confidentiality mechanisms.

2.“Catching IMSI Catchers” – Tobias Engel (2014)

Tobias Engel explains how fake base stations (IMSI catchers) exploit GSM’s lack of mutual authentication. The paper also introduces initial methods for detecting these rogue towers using signal inconsistencies and baseband monitoring, influencing modern tools like SnoopSnitch.

3.“Practical Attacks Against GSM Networks” – Karsten Nohl & Sylvain Munaut (2010)

This work showcases real-time MITM attacks on GSM using software-defined radios and open-source stacks like OsmocomBB. The authors demonstrated interception of SMS and calls, proving how outdated encryption standards make GSM vulnerable to low-cost attacks.

4.“A Survey on GSM Security Issues” – IJCA (2012)

The survey outlines major GSM threats such as SIM cloning, man-in-the-middle attacks, and replay attacks. It points out that GSM lacks end-to-end encryption and mutual authentication, urging the adoption of stronger protocols in future network generations.

5.“Baseband Attacks on GSM/UMTS Stack” – Collin Mulliner & Nico Golde (2011)

This paper focuses on vulnerabilities in the baseband processors of mobile devices. The authors demonstrated how attackers could remotely exploit these to cause DoS attacks or perform stealth MITM attacks, bypassing higher-layer protections.

6.“Analysis of GSM Protocol Vulnerabilities” – C. Paget (2009)

Chris Paget explains how GSM protocol flaws allow rogue BTS setups to impersonate legitimate towers. The study proves that GSM phones will connect to the strongest signal without validating tower authenticity, enabling easy interception.

1. “GSM Interception Using OpenBTS” – DEFCON Paper (2013). Demonstrates the practical execution of GSM interception using OpenBTS and SDRs. It provides insight into configuring a rogue BTS to capture voice and SMS traffic, and highlights how weak encryption allows data decryption.
2. “Security Analysis of GSM and UMTS Mobile Networks” – 3GPP Report (2008) .A technical report comparing GSM and 3G/UMTS security features. It emphasizes GSM’s lack of mutual authentication and integrity protection, making it inherently vulnerable to eavesdropping and MITM attacks.
3. “Cellular Network Security: Weaknesses in GSM” – NIST Report (2011).This U.S. government report outlines GSM’s insecurity for modern communications. It analyzes flaws in the encryption algorithms and highlights encryption downgrading as a major attack vector used in MITM scenarios.

10.“Exploring IMSI Catchers: Fake BTS Devices” – J. Classen (2015) Focuses on IMSI catchers and their inner workings. The paper explains how they are used for surveillance by exploiting GSM’s design flaws, particularly the passive broadcast of IMSI and lack of tower verification.

**CHAPTER 3**

**SYSTEM REQUIREMENTS AND DESIGN**

**3.1 PROBLEM DEFINITION**

The GSM (Global System for Mobile Communications) architecture was developed in the early 1990s, at a time when network-level attacks were not as widespread. As a result, it lacks critical modern security mechanisms, especially mutual authentication between the mobile device and the base station. This allows attackers to set up fake base stations (BTS), also known as IMSI catchers, which can impersonate legitimate network towers and conduct Man-in-the-Middle (MITM) attacks.

The attacker can intercept or manipulate communications between a mobile device and the real network, including SMS, calls, and personal identification like IMSI or TMSI. The existing mobile networks lack built-in tools to detect such rogue stations or alert the user. Hence, there is an urgent need for an automated detection system that analyzes GSM packet traffic and flags potential MITM behavior using intelligent algorithms.

**3.2 EXISTING SYSTEM**

The current generation of base station threat detection mechanisms in GSM and wireless communications still heavily depend on manual or semi-automated methods. These systems, while effective in detecting basic anomalies, are far from sufficient in combating modern, dynamic, and intelligent threats posed by rogue base stations, IMSI catchers, and GSM jammers.

1. Manual SDR Observation

One of the primary methods still in use involves manual inspection of Software Defined Radio (SDR) packet captures using tools such as Wireshark, Universal Radio Hacker (URH), and SDR#. These tools allow security researchers and network engineers to monitor GSM signaling channels and track inconsistencies in parameters such as Cell ID, TMSI changes, and Location Area Code mismatches. While SDR-based methods provide deep protocol-level insights, they are highly non-scalable, require significant technical expertise, and are time-consuming. Furthermore, these tools lack automation, limiting their use for real-time threat detection or active defense in operational environments.

2. Static Blacklists and IMSI Catcher Identifiers

Another commonly used solution is blacklist-based filtering, which relies on static databases of suspicious tower IDs, IMSI ranges, or even entire frequency bands known to be misused. These blacklists are integrated into mobile security software or SIEM (Security Information and Event Management) systems. While helpful against previously known threats, they are entirely reactive in nature and cannot detect new or dynamically reconfigured rogue towers, especially those using IMSI rotation or legitimate-sounding Cell IDs. This results in a high false negative rate and limited operational efficacy.

3. Government-Grade Surveillance Detection Tools

High-end solutions such as Stingray Detectors, IMSI Catcher Locators, and GSM monitoring units are deployed by military, law enforcement, and intelligence agencies. These systems offer precision detection and tracking of rogue base stations by triangulating mobile device behavior, signal strength patterns, and spectrum anomalies. Moreover, their deployment is often restricted by regulatory constraints, making them unsuitable for large-scale civilian deployment.

4. Crowd-Sourced Mobile Applications

Mobile apps like SnoopSnitch, Cell Spy Catcher, and GSM Spy Finder attempt to detect rogue towers by leveraging crowdsourced data. These tools rely on user-submitted reports of suspicious behavior such as signal drops, network switching, or encrypted-to-unencrypted handovers. While community-driven, their effectiveness is limited by user density, device compatibility, and the inability to access low-level GSM signaling information due to restrictions on consumer hardware and operating systems.

**3.2.1 LIMITATIONS OF THE EXISTING SYSTEM**

The detection of malicious cellular base stations—such as rogue towers or IMSI catchers—currently relies on a mix of manual analysis, static datasets, and specialized hardware tools. A widely adopted method involves the manual examination of Software Defined Radio (SDR) captures using tools like Wireshark. While SDR provides fine-grained visibility into GSM signaling traffic, its analysis is labor-intensive, requiring cybersecurity professionals with a strong background in telecommunication protocols. The sheer volume of signaling data makes it impractical for real-time monitoring or deployment at scale.

Another common practice includes the use of static blacklists, which flag suspicious tower IDs or known rogue devices. This approach can detect known threats, it fails to adapt to dynamic adversaries or newly configured rogue towers- commonly seen in targeted attacks or espionage. These blacklists also struggle with false positives and require constant manual updates, reducing their effectiveness over time.

In high-security environments, government surveillance detection tools such as Stingray or Catcher Detector systems are deployed. These tools are proprietary, expensive, generally not available to the public or small organizations. They come with deployment overhead, requiring physical proximity to the threat and operational expertise, which limits their practicality for continuous or widespread monitoring.

Crowd-sourced mobile applications represent another category, relying on users to share signal strength anomalies or tower behaviors. While they provide community-based threat awareness, these systems are inherently inconsistent and delay-prone. They lack the technical depth to analyze low-level signaling layers and often depend on a large user base for accuracy. Furthermore, they do not provide automated classification or threat scoring, leaving interpretation to the end-user.

Across all these methods, a recurring issue is the lack of intelligence and automation. These tools do not integrate machine learning or AI-based analytics for automated decision-making, nor do they scale effectively for real-time GSM network monitoring. This creates a major gap in the ability to identify and classify rogue towers or fake base stations proactively and without human oversight. As cellular attacks grow more sophisticated, there is a pressing need for intelligent, data-driven solutions that can analyze GSM traffic in real time, identify anomalies using pattern recognition, and trigger alerts without human intervention.

**3.3 PROPOSED SYSTEM**

Emergency response systems play a critical role in ensuring public safety, especially during unexpected threats. Among them, panic buttons are widely adopted in schools, hospitals, vehicles, and wearable safety devices. However, such systems often rely on networks such as GSM or unsecured WiFi, rendering them vulnerable to interception through Man-in-the-Middle (MitM) attacks.

The proposed system intends to explore this vulnerability by designing and implementing a panic alert prototype using an ESP8266 microcontroller. The prototype mimics real-world GSM communication using WiFi and HTTPS-based alerts. It integrates messaging APIs like Telegram to transmit alerts and uses an OLED screen to reflect system activity. By incorporating tools such as Wireshark and mitmproxy, we simulate various attack vectors and evaluate their impact. The study further proposes robust countermeasures aimed at securing such systems from unauthorized interception and data manipulation.

This project contributes to both cybersecurity research and emergency device development, providing an intersection of hardware prototyping and real-world digital threat detection.

**Key Features:**

* Trigger-based Panic Alert System
* A physical panic button (GPIO-based) triggers the alert mechanism.
* Secure WiFi-based Message Transmission
* Messages are sent via HTTPS to secure messaging platforms (e.g., Telegram).
* Simulated Vulnerability Exploitation (MitM)
* Traffic interception is demonstrated through packet sniffing and SSL downgrades.
* Real-Time OLED Status Display
* Displays connection, message-sent status, and error logs.
* Detection of Anomalous Communication
* Unusual HTTP codes, SSL errors, and spoofed responses are identified and logged.

**Proposed Countermeasures for Real-world Deployment**

Introduces future mitigation strategies including multi-layer verification, certificate pinning, and encrypted payload delivery.

**Architecture:**

The proposed system follows a multi-layered architecture to capture all components of communication, monitoring, and security evaluation.

**Input Layer:** Physical push button acting as an emergency trigger.

**Processing Layer:** ESP8266 microcontroller processes input and prepares the HTTPS request.

**Communication Layer:** WiFi module handles encrypted communication with the Telegram API endpoint.

**Monitoring Layer:** OLED screen reflects system states (idle, message sent, error).

**Security Analysis Layer:** MitM tools like Wireshark, SSLStrip, and mitmproxy are used to intercept and analyse packets.

**Mitigation Layer:**

Implements logging for anomaly detection and outlines future improvements like geolocation-based authentication and double-check alerts.

**3.3.1 OBJECTIVE OF THE PROPOSED SYSTEM**

**Primary Objective:**

To evaluate and enhance the security of panic alert systems that communicate over GSM or unsecured WiFi networks, by building a functional prototype, demonstrating real-time MitM attacks, and proposing counter-defensive measures.

**Specific Objectives:**

* To Build a Functional Panic Alert Prototype:
* A working ESP8266-based emergency alert model with message transmission capabilities.
* Displaying alert status and network connection information on OLED.

**To Simulate Network-based Attacks:**

* Conducting MitM attacks using SSL downgrade and ARP spoofing techniques.
* Evaluating if the alert data can be delayed, intercepted, or altered.

**To Detect Vulnerabilities in Communication:**

* Identifying loopholes in SSL implementation and analysing packet behaviour under simulated attacks.
* Recording occurrences of forged HTTP/HTTPS responses.

**To Propose a Security Protocol:**

* Enhancing HTTPS use through certificate pinning.
* Proposing encrypted message payloads.
* Establishing backup protocols like SMS fallbacks and alert repetition with varied channels.

**To Educate Developers and Public on the Risks:**

* Awareness on the implications of using insecure protocols.
* Suggesting safer architectural patterns and best practices.

**Scope of the Proposed System**

This system serves a dual role: providing a prototype model for educational purposes and highlighting critical cybersecurity concerns in embedded systems. The scope includes:

* Designing and developing an affordable and functional panic button prototype.
* Educating through demonstration the types and consequences of MitM attacks.
* Emphasizing the urgency to secure emergency communication mechanisms.
* Offering insights into future implementations in GSM-enabled hardware like SIM800L, which can integrate similar logic with cellular capabilities.
* Target applications include women and child safety devices, elderly alert systems, disaster management tools, and smart home emergency interfaces.

**Benefits of the Proposed System:**

**Affordable Implementation:** Utilizes low-cost hardware components like ESP8266 and OLED screens with open-source API integrations.

**Scalability Across Networks:** Can be extended to operate on GSM and LTE modules for wider accessibility.

**Security-Oriented Design:** Encourages proactive encryption and anomaly detection in IoT devices.

**Cyber Awareness:** Demonstrates vulnerabilities that are typically overlooked in consumer-level safety gadgets.

**Developer-Friendly:** The codebase and hardware model are kept simple, making it accessible for students, researchers, and startup developers.

**Conclusion:**

The proposed system serves as a practical and educational model that bridges the gap between IoT development and cybersecurity. By simulating a real-world emergency alert system and actively showcasing its vulnerabilities to MitM attacks, we underline the pressing need to fortify such systems against digital threats.

Our study demonstrates that even critical applications like panic buttons, when implemented over insecure networks, are susceptible to breaches. The proposed countermeasures such as encryption enforcement, redundant channel alerts, and anomaly-based detection provide an immediate layer of protection that can be realistically adopted.

As a next step, the project aims to implement GSM module integrations and explore machine learning models for intelligent anomaly prediction. Ultimately, this work lays the foundation for building resilient, responsive, and secure emergency communication systems in smart cities and beyond.

**3.4 MODULE IDENTIFICATION**

**3.4.1. Panic Button Interface**

The Panic Button Interface acts as the core user input mechanism. It is a compact, physical interface based on a tactile push-button integrated into the ESP8266 microcontroller unit. The button serves as a real-time emergency trigger that, once pressed, initiates a sequence of operations ranging from message preparation to network transmission.

**Design and Functionality:**

* Built using GPIO pins on the ESP8266.
* Debouncing logic is applied in code to prevent multiple trigger events
* Visual feedback is provided via an OLED display to indicate:
* "System Ready"
* "Alert Sent"
* "Error in Transmission"

**Key Characteristics:**

* Instant Activation: One-press action initiates the alert protocol.
* Minimal Power Consumption: Ideal for battery-operated or portable systems.
* User-Centric Design: Designed for non-technical users in emergency situations, including children and the elderly.

**3.4.2. Alert Transmission over Network**

Once the panic button is pressed, the ESP8266 initiates network-based alert transmission using a secure HTTP protocol over WiFi. While the system is model using WiFi for simplicity and lab-based simulation, its architecture is designed to be compatible with GSM modules (like SIM800L), which would allow SMS or mobile-data-based alerting in real-world scenarios.

**Steps Involved in Transmission:**

* Message Preparation: The system encodes an alert message, including timestamp, location (optional), and device ID.
* HTTPS API Integration: Telegram Bot API is used to send the message to a predefined recipient or group.
* Status Acknowledgment: After a successful transmission, the API returns a success status which is logged and displayed.

**Security Concerns:**

Despite HTTPS, if a user connects through open or public WiFi, man-in-the-middle (MitM) attacks are possible.

Vulnerabilities arise when certificate validation is bypassed or APIs are not correctly authenticated.

**3.4.3. Vulnerability Simulation – MitM Setup**

A critical part of this system is the simulation of real-world attacks using Man-in-the-Middle (MitM) techniques. This module demonstrates how a malicious actor can intercept, modify, delay, or drop alert messages.

**MitM Setup Components:**

* Wireshark: For deep packet inspection and protocol-level analysis.
* Mitmproxy / SSLStrip: To simulate SSL stripping attacks and manipulate HTTP responses.
* Router Spoofing / ARP Poisoning: Used to reroute traffic through attacker-controlled nodes.

**Use Case Demonstrations:**

* Delay Simulation: Alerts delayed using buffering or DNS poisoning.
* Message Spoofing: Alert content replaced with false data.
* Interruption: Alert transmission halted by packet drops or forged disconnections.

**Educational Insight:**

This module serves as a live demonstration tool to educate about the fragility of unsecured communication channels, especially in emergency systems.

**3.4.4. Detection and Logging Module**

To counteract vulnerabilities, a Detection and Logging Module is incorporated to identify anomalies in the communication process. This module records and analyses both system behaviour and incoming responses from the API.

**Features of the Module:**

Traffic Monitoring: Logs IP addresses, SSL certificates, response time, and HTTP status codes.

**Anomaly Detection Rules:**

* Unexpected response codes (e.g., 301 instead of 200).
* Delayed responses beyond timeout thresholds.
* SSL certificate mismatch alerts.
* Error Logging: All failed or suspicious transactions are timestamped and stored locally (or optionally pushed to a server for audit).

**Use Case:**

If an attacker intercepts and modifies the message, the system may receive no response or a malformed response. These are caught and stored for further analysis, raising a security flag.

**3.4.5. Security Enhancement Layer**

This layer is the proposed solution space of the project. It introduces a collection of security practices and design enhancements aimed at hardening the system against MitM and related attacks.

**Key Techniques Implemented/Suggested:**

* Certificate Pinning: The system accepts only pre-authorized SSL certificates.
* Encrypted Payloads: Even within HTTPS, the message content is AES encrypted.

**Redundant Alert Channels:**

* If the primary HTTPS fails, the system switches to an SMS gateway.
* Dual-channel verification ensures delivery.

**Two-Factor Triggering:**

Combines button press with GPS-based movement verification or a timed code.

**Watchdog Timer:**

In case of network failure, the system retries alert transmission until acknowledged.

**Long-term Suggestions:**

* Use of blockchain-based confirmation systems for alert traceability.
* Implementation of anomaly-based intrusion detection models using AI.

**3.4.6. Visualization and Output**

This module is designed to provide immediate, clear, and real-time feedback to both the user and system operator. Visualization includes both physical interface display and remote alert confirmation.

**Physical Visualization (OLED):**0.96-inch OLED connected to ESP8266.

**Displays:**

* Initialization
* Connection Status
* Alert Sent/Failed
* Error Codes (custom-coded for easy debugging)

**Remote Output (Telegram Interface):**

**Message shows:**

* Device ID / User ID
* Timestamp of the event
* Optional location
* Alert Type (e.g., panic, tamper, test)

**Advanced Visualization Options:**

* Logs can be pushed to a web dashboard showing:
* Alert frequency
* Detection logs
* Success/failure statistics
* Alerts can be color-coded and geo-mapped for large-scale monitoring.

**3.5 SYSTEM REQUIREMENTS**

**1. Hardware Requirements**

**Component Specification / Description**

* ESP8266 Node-MCU: WiFi-enabled microcontroller for embedded communication and logic control.
* Push Button: Physical trigger for panic alerts (connected via GPIO).
* OLED Display (0.96"): Displays real-time status and system messages.
* Breadboard & Jump Wires: For circuit prototyping and hardware assembly.
* Power Supply / USB Cable: To power the ESP8266 and connected modules.
* Laptop/PC: Required for programming, simulation, and monitoring.
* Optional: SIM800L Module GSM module for real-world mobile network testing (optional extension).
* WiFi Router / Access Point: Provides the network for device communication and attack simulation.

**2. Software Requirements**

* Software Tool / Platform: Purpose / Description
* Arduino IDE: Programming and uploading code to ESP8266.
* ESP8266 Board Package: Required for compiling ESP8266-specific firmware in the Arduino IDE.
* Telegram Bot API: API used to transmit panic alert messages securely.
* Mitm-proxy / SSLStrip: For simulating MitM attacks and analysing traffic vulnerabilities.
* Wireshark: Packet sniffer and protocol analyser for monitoring network communication.
* Python (optional): Used for automating detection logs or triggering attack scripts.
* Serial Monitor / PuTTY: To observe system logs from ESP8266 in real-time.
* Operating System: Windows / Linux / macOS (for development and simulation setup).

**3. Additional Dependencies and Libraries**

* ESP8266WiFi.h: Enables WiFi capabilities on the ESP8266.
* ESP8266HTTPClient.h: Allows HTTP/HTTPS communication via API.
* Adafruit\_GFX.h, SSD1306.h: Used to control the OLED display.
* Universal Telegram Bot Library: Used for Telegram Bot API integration in ESP8266.
* Minimum System Requirements (for Development Machine)
* Processor: Intel i3 or equivalent (or higher)
* RAM: Minimum 4 GB
* Storage: Minimum 500 MB free space for development tools
* Internet Connectivity: Required for API communication and package installations

**3.6 DESIGN PROCESS AND EXPLANATION**

**3.6.1 ARCHITECTURE DIAGRAM**

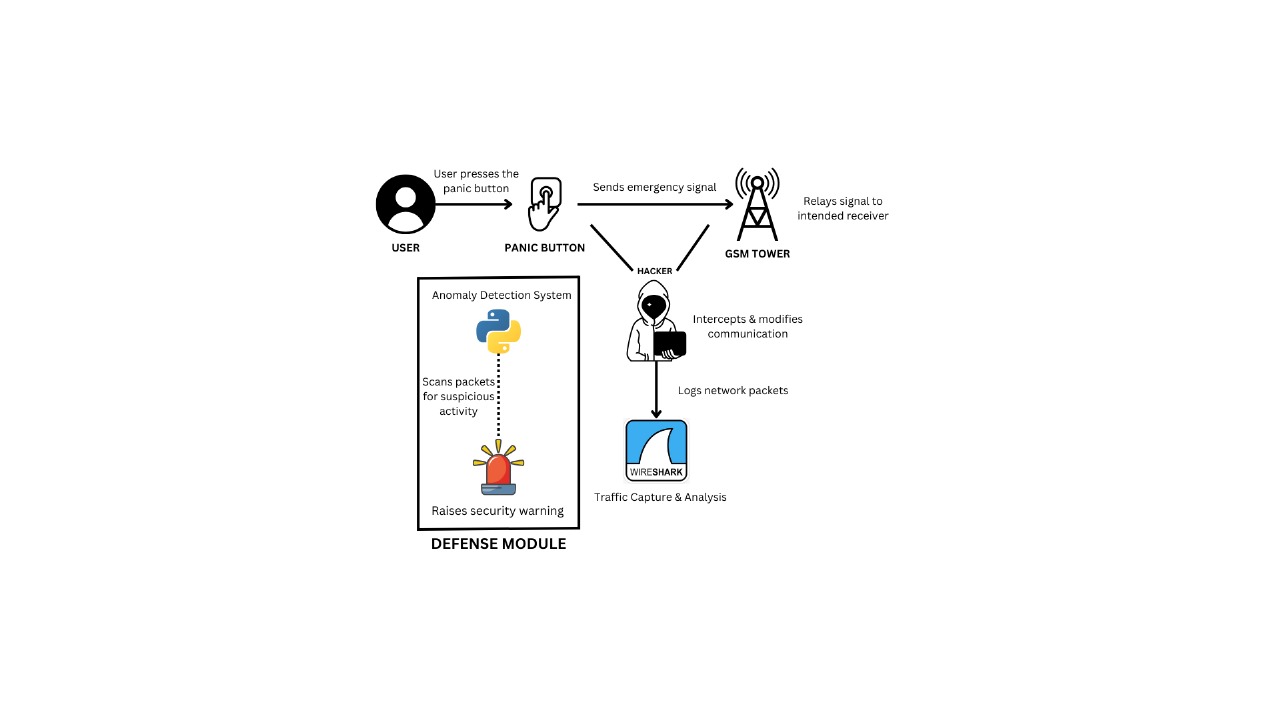


Fig 3.1 Architecture Diagram

Fig 3.1 Shows that the communication flow of a panic button system, highlighting how a Man-in-the-Middle (MitM) attacker can intercept signals and how a defense module detects and alerts anomalous behavior. The setup includes a user, a panic button, GSM transmission, a hacker node, and a real-time anomaly detection system.

* The process starts when a user presses the panic button, which triggers an emergency signal.
* This signal is intended to reach a legitimate GSM tower, which then relays it to emergency contacts.
* However, the attacker (MitM hacker) can intercept this signal mid-transmission.
* The hacker uses tools like Wireshark to log, inspect, and potentially alter or delay the signal.
* This manipulation can result in the receiver not getting the correct or timely alert, causing a security failure.
* To counteract this, the Defence Module continuously scans network traffic for anomalies.
* It uses a Python-based anomaly detection system that monitors for irregular packets, SSL strip attempts, or IP spoofing.
* On identifying suspicious behaviour, the system raises a warning to inform the user or administrator.
* This simulation helps in demonstrating real-world GSM / WiFi vulnerabilities in emergency systems.
* Ultimately, the architecture promotes the implementation of secure, encrypted, and verifiable communication methods in critical IoT systems.

**3.6.2 CIRCUIT DIAGRAM**

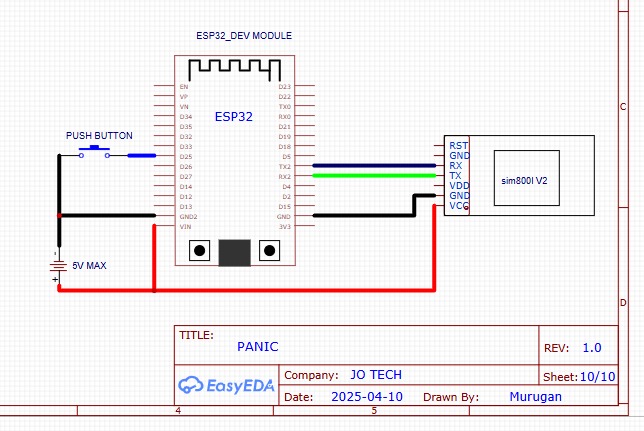


Fig 3.2 Circuit Diagram

Fig 3.2 This circuit diagram shows the wiring of an ESP32 development board with a push-button and SIM800L GSM module for transmitting panic alerts. It enables wireless emergency communication upon button press.

* The ESP32 module is the main controller responsible for managing input and communication.
* A push-button is connected to one of the GPIO pins of the ESP32, acting as the emergency trigger.
* When the button is pressed, the ESP32 detects the input signal and initiates a response.
* The system is powered using a 5V supply, which powers both the ESP32 and SIM800L GSM module.
* Communication between ESP32 and SIM800L is established via TX (Transmit) and RX (Receive) pins using UART protocol.
* The SIM800L module is responsible for sending the emergency alert as an SMS or data message over the GSM network.
* The circuit includes proper grounding (GND) and power (VCC/VDD) connections for stable operation.
* The design ensures real-time wireless alert delivery, simulating GSM network behaviour.
* This setup can be integrated with other modules like GPS for location-based alerting.
* The diagram was created using Easy-EDA, ensuring clarity and accuracy for PCB implementation and prototyping.

**CHAPTER 4**

**SYSTEM METHODOLOGIES**

**4.1 MODULE EXPLANATION**

**4.1.1. PANIC BUTTON INTERFACE**

**Description:**

The Panic Button Interface acts as the initiating point of the system, allowing a user to manually trigger an emergency signal. It is designed to be lightweight, reliable, and fast-responding. It comprises a physical push button connected to a microcontroller (ESP32), programmed to detect a press event and initiate an emergency communication cycle.

**IPO Model:**

* Input: User presses the physical panic button
* Process: ESP32 detects button state change via GPIO and prepares alert packet
* Output: Alert message with status and optionally location is prepared and sent

**Features:**

* Real-time manual triggering
* OLED screen status display
* Debouncing logic for noise-free input
* Optionally integrates GPS or geofencing

**Detailed Explanation:**

* The Panic Button is a critical human-interface device that represents the first stage of response.
* Upon pressing, it sends a digital signal (HIGH) to an input GPIO pin on the ESP32.
* The system uses interrupts or polling mechanisms to detect the press event without delay.
* Feedback is shown to the user via an OLED screen that displays "Alert Triggered."
* The interface can optionally integrate location modules like NEO-6M GPS to enrich the alert data.
* For testing purposes, a simulated OLED display also assists in debugging or demonstrating status transitions.

The goal is to achieve maximum responsiveness and reliability, ensuring no false triggers or missed signals.

**4.1.2. ALERT TRANSMISSION OVER NETWORK**

**Description:**

This module handles the secure and timely transmission of the panic alert over the internet or GSM network. The ESP32 communicates via WiFi or GSM (SIM800L) and sends the message to a central server or service like Telegram, Twilio, or SMS gateway.

**IPO Model:**

* Input: Alert message (button press, status, location, timestamp)
* Process: Format alert → Establish HTTPS/GSM connection → Transmit data
* Output: Message received at the designated endpoint (API, mobile device, cloud)

**Features:**

* HTTPS POST request using WiFi (test mode)
* GSM/SIM800L SMS transmission (real-world mode)
* Optional MQTT/REST API support
* End-to-end encryption suggested via HTTPS

**Detailed Explanation:**

* In the prototype, Telegram Bot API is used as a delivery channel due to its free and secure interface.
* The ESP32 encodes the alert message into a URL or JSON payload, containing critical information like time and location.
* This message is sent over a secure HTTPS connection (when using WiFi) or via AT commands to SIM800L in a GSM context.
* Redundancy mechanisms can be introduced, such as retransmission on failure or backup endpoints.
* Proper transmission confirmation is implemented using response code validation (e.g., HTTP 200 OK).
* Latency is minimized to maintain real-time effectiveness.

**4.1.3. VULNERABILITY SIMULATION – MITM SETUP**

**Description:**

This component demonstrates how real-world panic alert systems are susceptible to Man-in-the-Middle (MitM) attacks. It involves intercepting and analysing traffic between the panic button and the server to simulate how an attacker could delay, spoof, or block the message.

**IPO Model:**

* Input: Network packets during transmission
* Process: Intercept traffic → Analyse payload → Modify or block packets
* Output: Altered, spoofed, or lost communication

**Tools Used:**

* Wireshark
* Mitm-proxy
* SSL-Strip
* Kali Linux or Raspberry Pi attacker node

**Detailed Explanation:**

* A MitM scenario is setup where the attacker places themselves between the ESP32 and the server.
* Using Wireshark, network packets are captured to observe payloads.
* With tools like SSLStrip, encrypted connections can be downgraded to HTTP (if misconfigured), revealing sensitive info.
* mitmproxy is then used to alter the content of the message – for instance, changing a GPS location or status code.
* This highlights critical weaknesses in non-HTTPS systems and the ease with which traffic can be spoofed or blocked.
* The attacker may even relay delayed or forged alerts, misleading the intended recipients.

Such simulations reinforce the need for TLS encryption, certificate pinning, and secure tokens in IoT communications.

**4.1.4. DETECTION AND LOGGING MODULE**

**Description:**

This module acts as a watchdog system, monitoring network traffic and scanning for signs of malicious interference. It logs all communication attempts and flags any suspicious patterns for further investigation.

**IPO Model:**

* Input: Captured traffic logs from the ESP32 or network
* Process: Scan for anomalies → Compare against whitelist/blacklist → Log result
* Output: Detection alerts, logs stored, possible user notifications

**Detection Methods:**

* Packet size anomalies
* Unexpected HTTP status codes
* TLS handshake failures
* Delayed or duplicate packets

**Detailed Explanation:**

* A Python-based Anomaly Detection System is used to process traffic logs (from Wireshark or ESP32 serial output).
* It checks for packet integrity, transmission timing, and whether the destination endpoint was correct.
* The module maintains a log file of all activities including time stamps, packet size, and content hashes.
* Suspicious activities trigger a warning mechanism that can alert administrators or shut down the system temporarily.
* Visualization of logs via tools like Grafana or custom dashboards is optional but recommended for field deployment.
* If GPS is used, the module checks for location mismatches or impossible jumps.

Overall, this module ensures auditability and real-time monitoring of the system's communication backbone.

**4.1.5. SECURITY ENHANCEMENT LAYER**

**Description:**

The Security Enhancement Layer proposes and integrates mechanisms to harden the system against attacks like MitM, spoofing, and replay. It ensures data integrity and user verification before message acceptance.

**IPO Model:**

* Input: Alert message before transmission
* Process: Apply encryption, validate certificates, attach checksum
* Output: Secure, verified message sent over the network

**Security Techniques:**

* TLS with Certificate Pinning
* SHA-256 checksums and HMAC
* Two-Factor Trigger (Button + Location)
* Time-based token expiration

**Detailed Explanation:**

* The alert message is signed with a hash function (e.g., SHA-256) to ensure data integrity.
* A shared secret key can be added for HMAC authentication.
* Certificate pinning ensures the ESP32 only connects to known, verified servers.
* The Two-Factor Trigger enhances security: an alert is only sent if both button is pressed and location matches a geofence
* This module blocks any alert that fails validation or decryption at the receiver side.
* Token expiration mechanisms (like TOTP) prevent replay attacks, where an old alert is resent maliciously.

**4.1.6. VISUALIZATION AND OUTPUT MODULE**

**Description:**

The Visualization and Output module displays the system’s internal status and user feedback in real-time. It helps both developers and users understand what’s happening in the system at any given moment.

**IPO Model:**

* Input: System events (button press, alert sent, error, detection warning)
* Process: Format messages → Render output → Store logs (if enabled)
* Output: OLED display, console output, mobile notification

**Visual Components:**

* OLED 128x64 Display
* Telegram/WhatsApp Notification Bot
* Web Dashboard (optional)
* Log Display in Terminal

**Detailed Explanation:**

* The OLED module is connected via I2C and shows status like "Ready", "Alert Sent", or "Warning Detected".
* Real-time updates offer reassurance to the user that their alert is being processed.
* On the backend, alerts can be mirrored to Telegram or WhatsApp as a secondary verification.
* Logs from detection modules can be pushed to a web dashboard or PC terminal for admins to review.
* Visualizations include packet flow timelines, timestamps, alert history, and failure logs.
* Optional integration with tools like Grafana or Node-RED can be added for advanced dashboards.

**CHAPTER 5**

**IMPLEMENTATION**

The panic button system is a microcontroller-based alerting solution that relies on the ESP8266 Node-MCU and Telegram Bot API to transmit emergency messages over a WiFi network. This system is designed to offer immediate response functionality in scenarios where quick communication is critical. The implementation leverages both hardware and software components to ensure low-cost deployment while maintaining functional reliability.

At the core of the system is the ESP8266 Node-MCU, a compact microcontroller with built-in WiFi capabilities. Connected to it is a push-button that serves as the manual trigger for panic alerts. The circuit is powered via a micro-USB cable and uses an SH1106 OLED screen to display real-time feedback to the user. The push-button is connected to a GPIO pin and configured using the internal pull-up resistor. This setup ensures the button reads a LOW signal when pressed, allowing the system to detect a panic trigger with high accuracy.

When powered on, the ESP8266 attempts to connect to a pre-configured WiFi network. During this phase, the OLED screen displays the connection status to inform the user. Once connected, the system enters an idle or standby mode, constantly monitoring the state of the panic button. The main execution loop (loop() function) checks if the button has been pressed. If the input is detected as LOW, the system recognizes it as an emergency signal.

On detecting the button press, the system performs multiple actions. First, the OLED screen displays a large-font alert message, such as "ALERT TRIGGERED," to notify the user that the signal has been processed. Simultaneously, the microcontroller constructs a message payload that includes pre-defined emergency text and a static Google Maps link (this can later be enhanced to dynamic GPS-based location). This message is sent via the Telegram messaging platform using the Bot API.

The Telegram integration is implemented through an HTTPS GET request. The ESP8266 uses the WiFi-Client-Secure library to initiate a secure connection to Telegram’s API server. It dynamically constructs the URL using a bot token and a chat ID, which points to the recipient or group who will receive the alert. The message text is included in the query string. Before the message is sent, it undergoes URL encoding to ensure special characters do not break the URL format. The ESP8266 then connects to the Telegram server, sends the request, and waits for a response. If successful, the OLED screen is updated with a confirmation message like "Alert Sent via Telegram." If there is an error, such as a failed network connection or API failure, an appropriate error message is shown.

The software implementation uses libraries like ESP8266WiFi.h,WiFi-ClientSecure.h, Adafruit\_GFX.h, and SH1106Wire.h to manage WiFi connectivity, secure transmission, and OLED graphics respectively. The WiFi credentials, bot token, and chat ID are stored as global variables in the code. The function sendTelegramMessage() handles the core logic of formatting and sending the message. For the purpose of simplifying SSL verification, the code uses client.setInsecure(), which disables certificate checks. While this aids in easier connectivity, it also introduces security vulnerabilities, which should be addressed in real-world applications through certificate pinning or fingerprint validation.

The OLED screen is a key user-interface component that guides the user throughout the system’s operation. It displays various status messages such as "Connecting to WiFi," "Ready," "Alert Triggered," and "Error." This ensures transparency and enhances usability, especially in stressful emergency scenarios. It also assists developers in debugging or verifying system behaviour during testing and demonstration.

One of the core highlights of this implementation is its simplicity and portability. It can be easily extended to use GSM modules like SIM800L for cellular-based alerting in environments where WiFi is unavailable. Moreover, dynamic GPS modules like NEO-6M can be integrated to provide real-time location data in the alert messages. The modular design of the code and hardware also allows for integration with mobile apps, cloud dashboards, or voice-based assistants in future iterations.

In terms of use cases, this system is highly versatile. It can be employed in elderly care for medical emergencies, in women’s safety tools for discrete alerts, in school or public infrastructure for incident response, or by remote workers operating in hazardous conditions. It provides a basic but effective means of sending emergency communication without requiring complex setup or infrastructure.

In conclusion, the ESP8266 Telegram Panic Button system demonstrates a compact, cost-effective solution for real-time emergency communication. Its working is straightforward but powerful—detect a button press, send a secure alert, and notify the user of success or failure. Its implementation lays the foundation for smarter, secure IoT-based emergency alert systems. Future upgrades could include secure authentication, encrypted payloads, real-time GPS, GSM fallback, and integration with smart city emergency grids. Overall, the project blends embedded systems design with basic cybersecurity principles to provide an impactful real-world application.

**CHAPTER 6**

**CONCLUSION AND FUTURE ENHANCEMENT**

The implementation of the ESP8266-based panic button system demonstrates a functional, low-cost, and reliable method for emergency alert transmission over wireless networks. Through the integration of a physical panic interface, real-time OLED feedback, and the Telegram Bot API, the system allows for instant communication during critical situations. This makes it a highly applicable solution for personal safety, elderly care, field operations, and security-sensitive environments.

One of the standout contributions of this project is its focus on network vulnerability awareness. By simulating Man-in-the-Middle (MitM) attacks, it effectively exposes potential flaws in unsecured communication setups. The ability to identify and log anomalies provides crucial insight into how easily real-world systems can be compromised if basic security layers are not enforced.

Overall, the project strikes a balance between functionality and security, highlighting the importance of integrating cybersecurity practices even in simple IoT systems. It serves as a foundation for future safety devices and offers a valuable learning model for both developers and security researchers.

**Future Enhancement**

Although the current system performs effectively within a WiFi-enabled environment and showcases the concept of alerting and vulnerability testing, several areas offer opportunities for further development and robustness:

* GSM Module Integration: Incorporating modules like SIM800L will allow the system to function in remote areas without WiFi, using SMS or mobile data for alerts.
* GPS-Based Dynamic Location Sharing: Integration of GPS modules (e.g., NEO-6M) can enhance the alert by sending the real-time location of the user, allowing responders to reach them faster.
* Secure Communication Protocols: Implement certificate pinning or fingerprint validation to avoid SSL bypass (currently disabled using client.setInsecure()). This will improve end-to-end encryption reliability.
* Redundant Alert Channels: Add fallback options such as SMS, email, or cloud-based alerts to ensure delivery even if one channel fails.
* Mobile App Integration: Develop a dedicated Android/iOS app that pairs with the device for custom alerts, logs, and two-way communication.
* Battery Backup and Low-Power Mode: Optimize power usage and introduce a battery backup system to ensure operation during power outages.
* Voice Activation or Sensor-Based Triggers: Enable voice commands or motion sensors as additional methods to trigger the alert, especially useful for users unable to press a button during an emergency.
* Dashboard for Admin Monitoring: Implement a cloud-based web dashboard to monitor multiple panic devices, analyse usage history, and track suspicious behaviours or detection logs.
* Machine Learning for Threat Detection: Use ML models to analyze traffic logs and automatically identify potential attacks or misuse based on pattern recognition.
* These enhancements can significantly expand the functionality, reliability, and real-world applicability of the system. They also make it more aligned with smart city safety infrastructure, enabling large-scale deployment in public transportation, campuses, and industrial zones.

**CHAPTER 7**

**REFERENCE**

1. Alex Biryukov, Adi Shamir, David Wagner, “GSM Security and the A5/1 Cipher,” Proceedings of the First Workshop on Security of Information Technologies (SIT'00), 2000.
2. Tobias Engel, “Catching IMSI Catchers,” Chaos Communication Congress (CCC), 2014.
3. Karsten Nohl, Sylvain Munaut, “Practical Attacks Against GSM Networks,” Chaos Communication Congress (CCC), 2010.
4. International Journal of Computer Applications (IJCA), “A Survey on GSM Security Issues,” Volume 45 – No.21, 2012.
5. Collin Mulliner, Nico Golde, “Baseband Attacks on GSM/UMTS Stack,” Black Hat USA, 2011.
6. Chris Paget, “Analysis of GSM Protocol Vulnerabilities,” DEFCON 17 Conference, 2009.
7. DEFCON Research Team, “GSM Interception Using OpenBTS,” DEFCON Paper, 2013.
8. 3rd Generation Partnership Project (3GPP), “Security Analysis of GSM and UMTS Mobile Networks,” Technical Report 33.902, 2008.
9. National Institute of Standards and Technology (NIST), “Cellular Network Security: Weaknesses in GSM,” NIST Technical Publication, 2011.
10. Jan Classen, “Exploring IMSI Catchers: Fake BTS Devices,” 25th Annual Computer Security Applications Conference (ACSAC), 2015.

**CHAPTER 8**

**APPENDIX**

**8.1.1 PANIC BUTTON**

**SOURCE CODE**

#include <ESP8266WiFi.h>

#include <Wire.h>

#include <WiFiClientSecure.h>

#include <ESP8266HTTPClient.h>

#include "SH1106Wire.h" // SH1106 library

// WiFi Credentials

const char\* ssid = "70V";

const char\* password = "kasitheatre";

// Telegram Bot Token and Chat ID

String botToken = "8094009865:AAGou2F-VFEmjWSQUpCurIBYb1xn8ATcPUs";

String chatID = "1268164843"; // Your Telegram Chat ID

// SH1106 OLED (I2C)

SH1106Wire display(0x3C, D2, D1); // SDA = D2, SCL = D1

// Button

const int buttonPin = D4; // GPIO2

void setup() {

Serial.begin(115200);

pinMode(buttonPin, INPUT\_PULLUP);

// Initialize OLED

display.init();

display.flipScreenVertically(); // Optional, depending on your screen orientation

display.setFont(ArialMT\_Plain\_10);

display.clear();

display.drawString(0, 0, "Panic Button Ready");

display.display();

// Connect to WiFi

WiFi.begin(ssid, password);

display.drawString(0, 15, "Connecting WiFi...");

display.display();

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("WiFi connected.");

display.drawString(0, 30, "WiFi Connected");

display.display();

}

void loop() {

if (digitalRead(buttonPin) == LOW) {

Serial.println("Panic button pressed!");

// OLED Alert

display.clear();

display.setFont(ArialMT\_Plain\_16);

display.drawString(0, 0, "🚨 PANIC ALERT");

display.setFont(ArialMT\_Plain\_10);

display.drawString(0, 30, "Sending Telegram...");

display.display();

// Send alert

String message = "🚨 Panic Button Pressed!\nTriggered manually (ESP8266)\nhttps://maps.app.goo.gl/ch2pYEEZRngxAiNb8";

sendTelegramMessage(message);

delay(5000); // debounce

}

}

void sendTelegramMessage(String message) {

if (WiFi.status() == WL\_CONNECTED) {

WiFiClientSecure client;

client.setInsecure(); // Skip SSL certificate validation

HTTPClient https;

String url = "https://api.telegram.org/bot" + botToken +

"/sendMessage?chat\_id=" + chatID +

"&text=" + urlencode(message);

Serial.println("Sending to Telegram:");

Serial.println(url);

https.begin(client, url);

int httpResponseCode = https.GET();

Serial.print("Telegram response: ");

Serial.println(httpResponseCode);

https.end();

display.drawString(0, 0, "Panic Button Ready");

} else {

Serial.println("WiFi not connected!");

}

}

// URL encode helper

String urlencode(String str) {

String encoded = "";

char c;

char code0, code1;

for (int i = 0; i < str.length(); i++) {

c = str.charAt(i);

if (isalnum(c)) {

encoded += c;

} else if (c == ' ') {

encoded += '+';

} else {

code1 = (c & 0xf) + '0';

if ((c & 0xf) > 9) code1 = (c & 0xf) - 10 + 'A';

c = (c >> 4) & 0xf;

code0 = c + '0';

if (c > 9) code0 = c - 10 + 'A';

encoded += '%';

encoded += code0;

encoded += code1;

}

}

return encoded;

}

**8.1.2 ANAMALOY DETECTION FOR MITM**

**SOURCE CODE**

from scapy.all import \*

import time

# List of keywords that indicate suspicious modifications

suspicious\_keywords = ["modified", "tampered", "fake", "hacked", "intercepted"]

# Function to detect packet anomalies

def packet\_callback(packet):

if packet.haslayer(Raw): # Check if packet has data

payload = packet[Raw].load.decode(errors="ignore") # Extract packet data

# Check for suspicious patterns

for keyword in suspicious\_keywords:

if keyword in payload.lower():

print(f"🚨 ALERT: Possible MITM attack detected! 🚨")

print(f"Suspicious Packet Content: {payload}")

print("-" \* 50)

# Sniff network packets (modify interface based on your setup)

print("🔍 Monitoring network traffic for anomalies...")

sniff(filter="ip", prn=packet\_callback, iface="Wi-Fi", store=False)

**8.2 SCREENSHOTS**

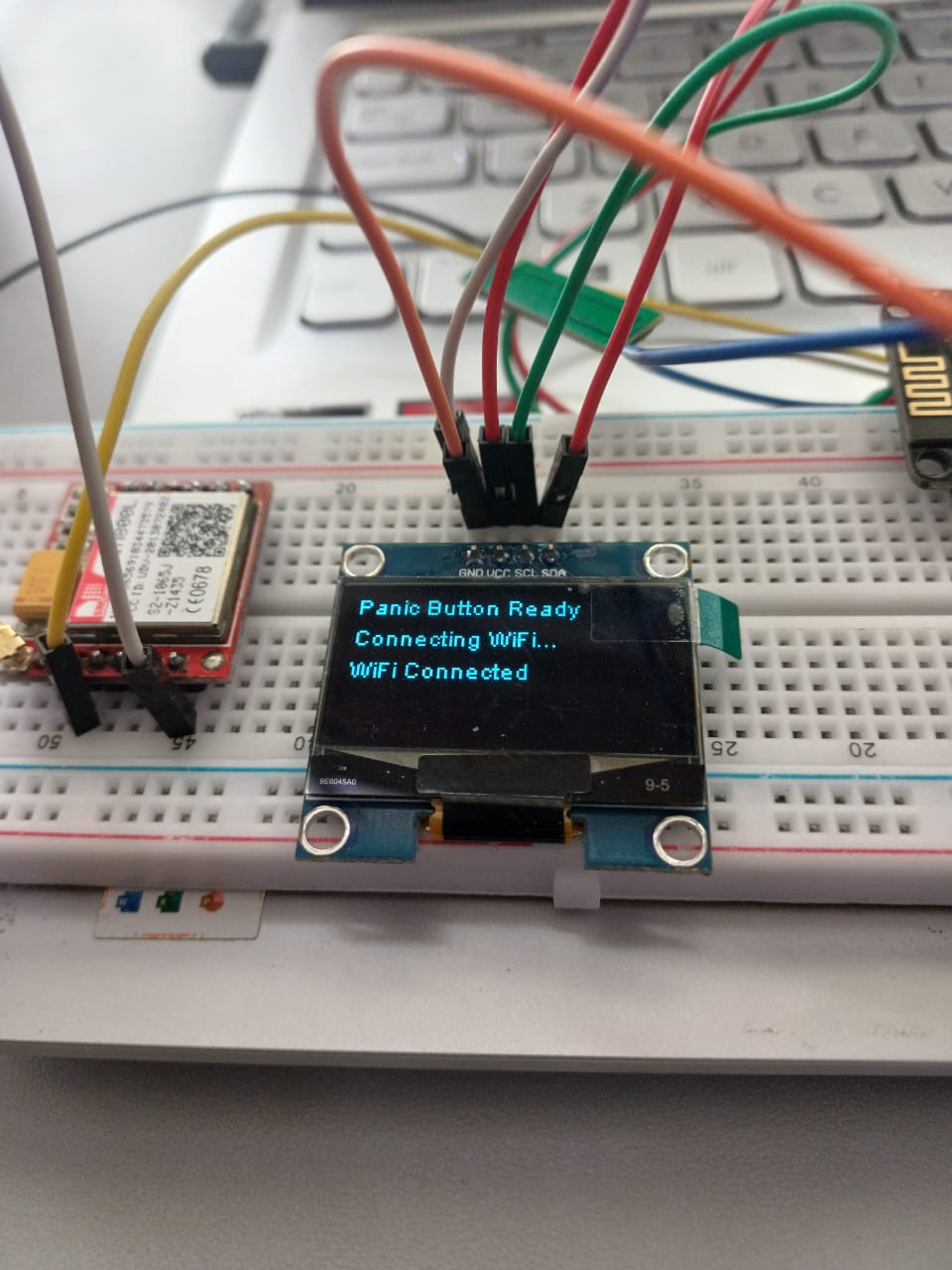


Fig 8.1 ESP Wi-Fi Connection Established

Fig 8.1 The image shows an IoT-based Panic Button prototype using an ESP module and OLED display on a breadboard. The OLED screen confirms system status messages such as "Panic Button Ready" and successful WiFi connectivity.

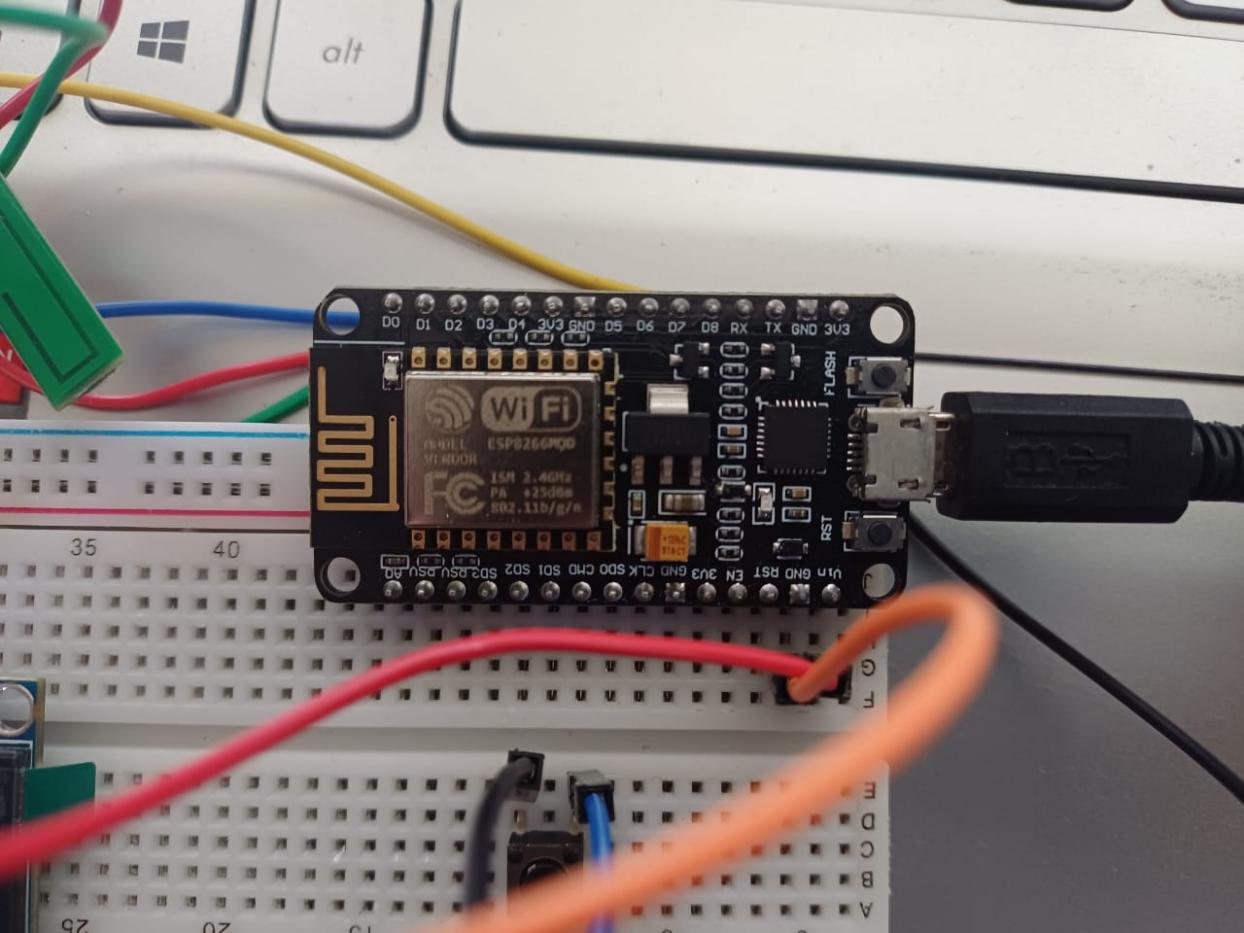


Fig 8.2 ESP8266 Node-MCU Microcontroller

Fig 8.2 The image shows an ESP8266 Node-MCU microcontroller connected to a breadboard and powered via a micro-USB cable. It serves as the main control unit for WiFi-based IoT communication in the panic button system.

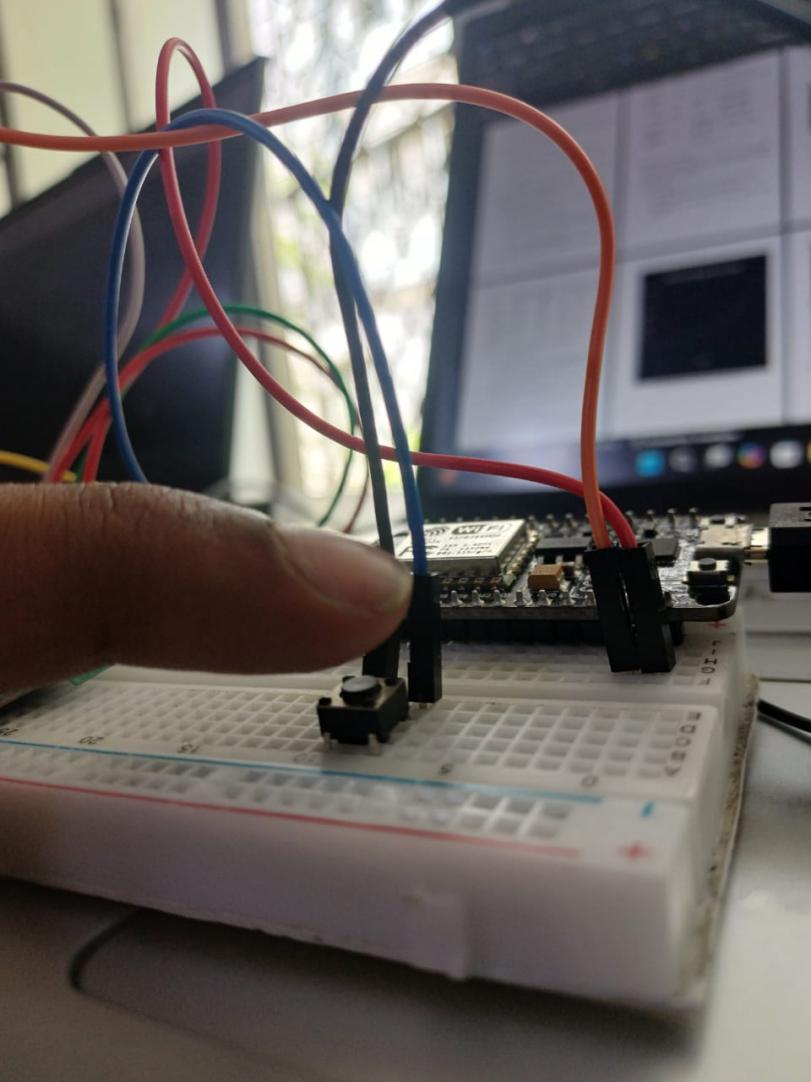


Fig 8.3 Push Button

Fig 8.3 The image shows a finger pressing a tactile push button connected to a breadboard and linked with the ESP8266 module. This button acts as the panic trigger, initiating alert transmission when pressed.

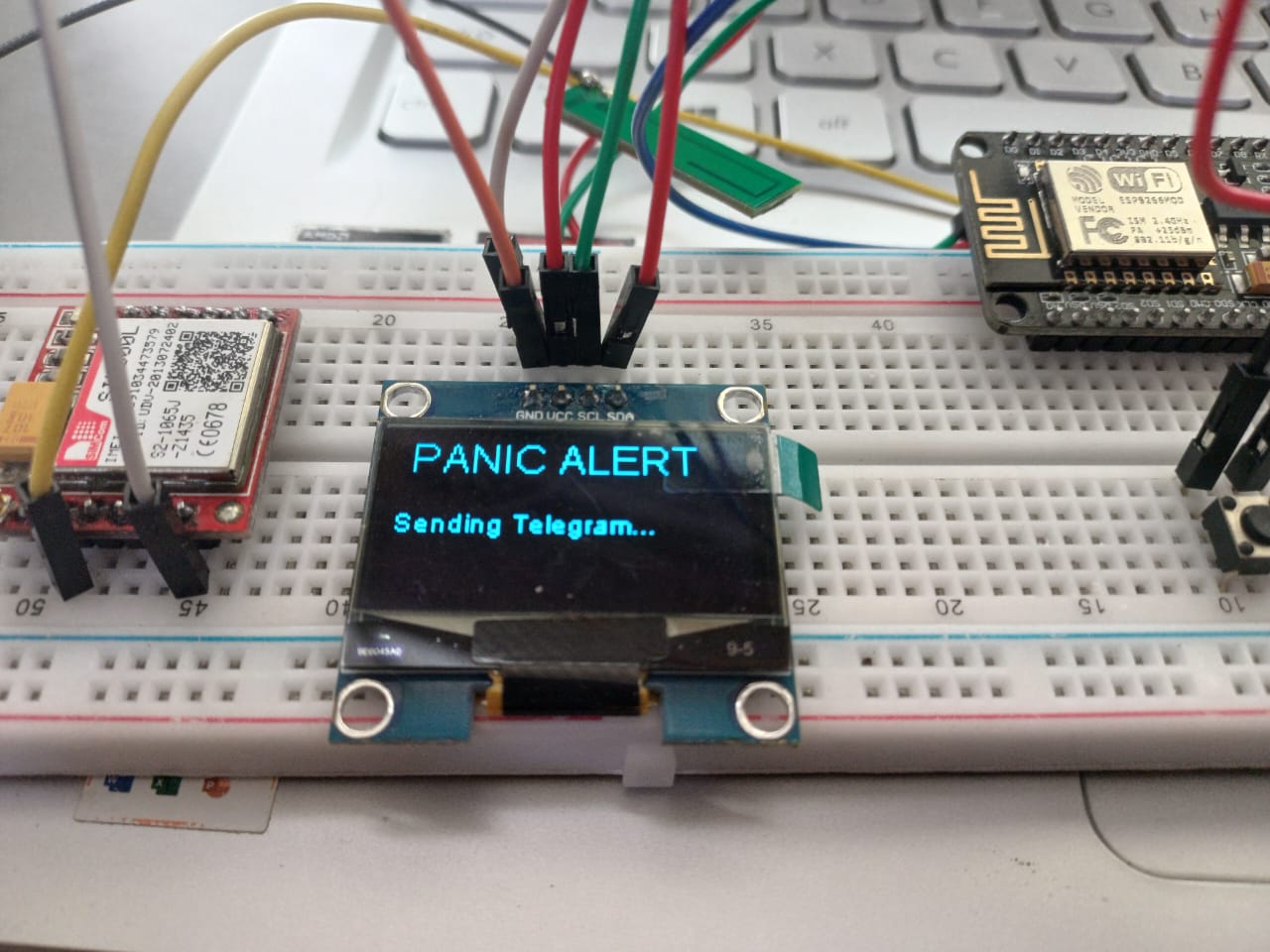


Fig 8.4 Transfering Result

Fig 8.4 Shows that it transmits an alert message along with user details to a designated Telegram chat using the Telegram Bot API.”

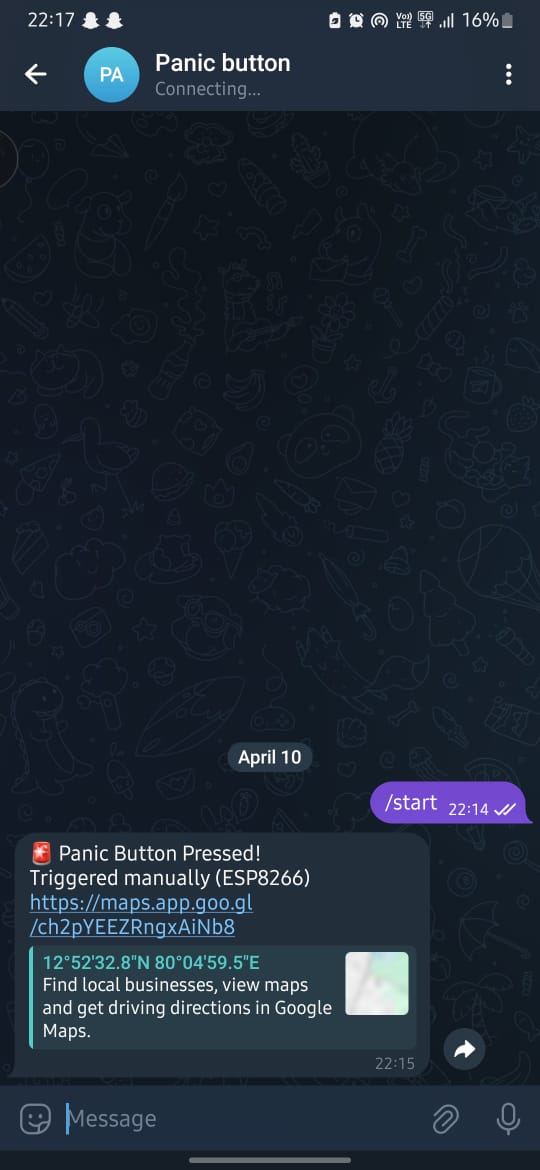


Fig 8.5 Telegram Page

Fig 8.5 Shows a Telegram alert is instantly sent containing the user's exact location. The message includes live GPS coordinates with a clickable Google Maps link for immediate tracking.

**TECHNICAL BIOGRAPHY**

****

ARSHAD T A, was born in 2004 in Kerala, India. I completed my higher secondary education in 2021. I am currently pursuing a B.Tech in Computer Science and Engineering (Cyber Security) at the School of Computer Information and Mathematical Sciences, B.S. Abdur Rahman Crescent Institute of Science and Technology, Vandalur, Chennai, India. My technical interests include backend development, data analysis, and cybersecurity, with hands-on experience in Python, MySQL, and data analysis frameworks. I have completed certifications in Python, SQL, Power BI, and MS Office Automation and have participated in workshops focused on cybersecurity and data science. I am also familiar with SDLC, Agile methodologies, and cloud-based development tools. I can be reached at [210181601008@crescent.education](mailto:210181601008@crescent.educationm) or +91 7550270492.



SANJAY K, born in 2004 in Chennai, India. I completed my higher secondary education in 2021 and I'm currently pursuing a BTech in Computer Science and Engineering with a specialization in Cyber Security at the School of Computer Information and Mathematical Sciences, B.S. Abdur Rahman Crescent Institute of Science and Technology, Vandalur, Chennai. I'm passionate about both tech and non tech —especially frontend development, data analysis, business analysis and cybersecurity. I’ve worked hands-on with Python, MySQL, Business analytics and data analytics frameworks. I've earned certifications in Python and Business Analytics, and participated in workshops focused on cybersecurity and data science. I'm also familiar with SDLC, GRC and Agile methodologies. I’ve also worked with BI Tools and Excel automation visualize and streamline data processes efficiently. I can be reached at [210181601045@crescent.education](mailto:210181601045@crescent.education) or +91 9361396391.