JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Sahayanet: A Ble-Enabled Iot Framework For Real-Time Women Safety Using Esp32-C3, Flutter And Firebase

[1]Yash Chavan, [2]Saurabh Randive, [3]Gaurav Mashere, [4]Mayuri Tawde
Under the Guidance of Prof. K.S. Sanghavi
Student, Department of Information Technology
Sinhgad Academy of Engineering, Kondhwa, Pune – 411048
Savitribai Phule Pune University, Maharashtra, India

***Abstract*—**Womens safety is a serious concern in our country, especially in urban and semi urban areas where incidents of harassment happen quite often. Most of the current safety apps need the user to unlock their phone and manually open the app during an emergency which is honestly very difficult when someone is actually in danger. In this paper we have designed and implemented SahayaNet, a wearable based IoT safety system using the XIAO ESP32-C3 microcontroller with BLE 5.0 connectivity, a Flutter based Android companion app, and a Firebase cloud backend. When user press the SOS button on the wearable, the device wakes up from deep sleep, sends a BLE advertisement, and the mobile app automatically captures GPS location, writes emergency data to Firebase Firestore, and sends push notification to nearby registered users and guardian contacts using Firebase Cloud Messaging (FCM). We also implemented continuous location tracking using Android WorkManager which updates location every 30 seconds. From our testing the entire SOS process from button press to notification delivery took average 8.8 seconds and wearable battery lasted more than 74 hours in standby mode. Overall SahayaNet provides a low cost, low power and practical solution for womens safety.

***Index Terms*—**Women Safety, IoT, BLE 5.0, ESP32-C3, Flutter, Firebase Firestore, FCM, GeoFlutterFire, GPS, SOS, Android WorkManager.

I. INTRODUCTION

Crimes against women in India has been increasing every year and its a major social problem that needs technological solutions. While there are many safety apps and devices available in market, most of them have a common limitation — they require the person to actively use their smartphone during the time of danger. In real life situations this is very impractical because when someone is suddenly attacked or in threat they cannot calmly take out phone, unlock it, find the app and press button. Research has shown that humans experience reduced motor control and cognitive load under acute stress [1] which makes multi-step phone interaction almost impossible during emergency.

Because of this gap we decided to build SahayaNet. Our system uses a small wearable device with just one SOS button. The user simply presses that button and everything else happens automatically without needing to touch the phone at all. We selected the XIAO ESP32-C3 module for the hardware because it is very small in size, supports BLE 5.0, and most importantly it has deep sleep mode which helps us achieve long battery life.

The complete SahayaNet system works in three layers. The first layer is the physical hardware wearable which the user wears on their wrist. The second layer is a Flutter based Android app that runs silently in background and listens for the BLE signal. The third layer is the Firebase cloud backend that stores the SOS data, finds nearby users using GeoFlutterFire, and sends push notifications through FCM.

Main objectives of our system are: (a) reduce the time from SOS button press to notification delivery to under 10 seconds, (b) make the wearable last at least 72 hours on single charge for daily wear, and (c) alert not only saved guardian contacts but also nearby community members who can physically help faster than someone far away.

Rest of the paper is organized as follows. Section II covers related work. Section III describes the proposed architecture. Section IV explains the implementation. Section V presents results and Section VI concludes the paper.

II. LITERATURE SURVEY

Harikiran et al. [2] proposed one of the earlier works on IoT based women safety system using ESP8266 and GSM at ICEEOT 2016. Their system could send location alerts to pre-registered contacts but did not have any mobile companion app or community alerting feature. It was a standalone hardware device.

Jatti et al. [3] developed a wearable device for safety of women and girl children presented at IEEE RTEICT 2016. Their approach used physiological signals like galvanic skin resistance and body temperature along with accelerometer data and machine learning for threat detection. The system was interesting but required sensors which increase device cost and size.

Hyndavi et al. [4] proposed a smart wearable device for women safety at ICCES 2020. They used pressure sensor, pulse-rate sensor and temperature sensor with outlier detection to automatically identify when a woman is in danger without needing manual trigger. The system sends alerts with GPS location to relatives and police station automatically. Their work is good but again it required multiple sensors making device bulky.

Pratheeba et al. [5] proposed a smart wearable device women safety system based on IoT published in IRJMETS 2021. They used ESP32 with GSM and GPS module directly on device for alerting. The system worked without phone which is an advantage but lacked real-time location tracking and community alerting.

Sanila and Sindhu [6] proposed a wearable device for women safety and defence in IJARSCT 2022. Their system focused more on physical defence mechanism. While useful, cloud based community alert was not part of their design.

GeoFlutterFire [7] is an open source dart library for Flutter that enables geo-hash based location queries on Firestore database. We used this library in our cloud backend to identify registered SahayaNet users within 5 km radius of the victim for community alerting purpose.

From reviewing existing literature we found that most works either use GSM+GPS hardware directly on device which increases cost and size, or depend on user to manually trigger phone app. SahayaNet fills this gap by combining a lightweight BLE wearable with a background running app and cloud based community alerting, which is a different approach from existing systems.

III. PROPOSED SYSTEM ARCHITECTURE

SahayaNet is a three tier distributed system where each tier handles specific responsibilities in the emergency response pipeline. Figure 1 shows the overall system architecture.

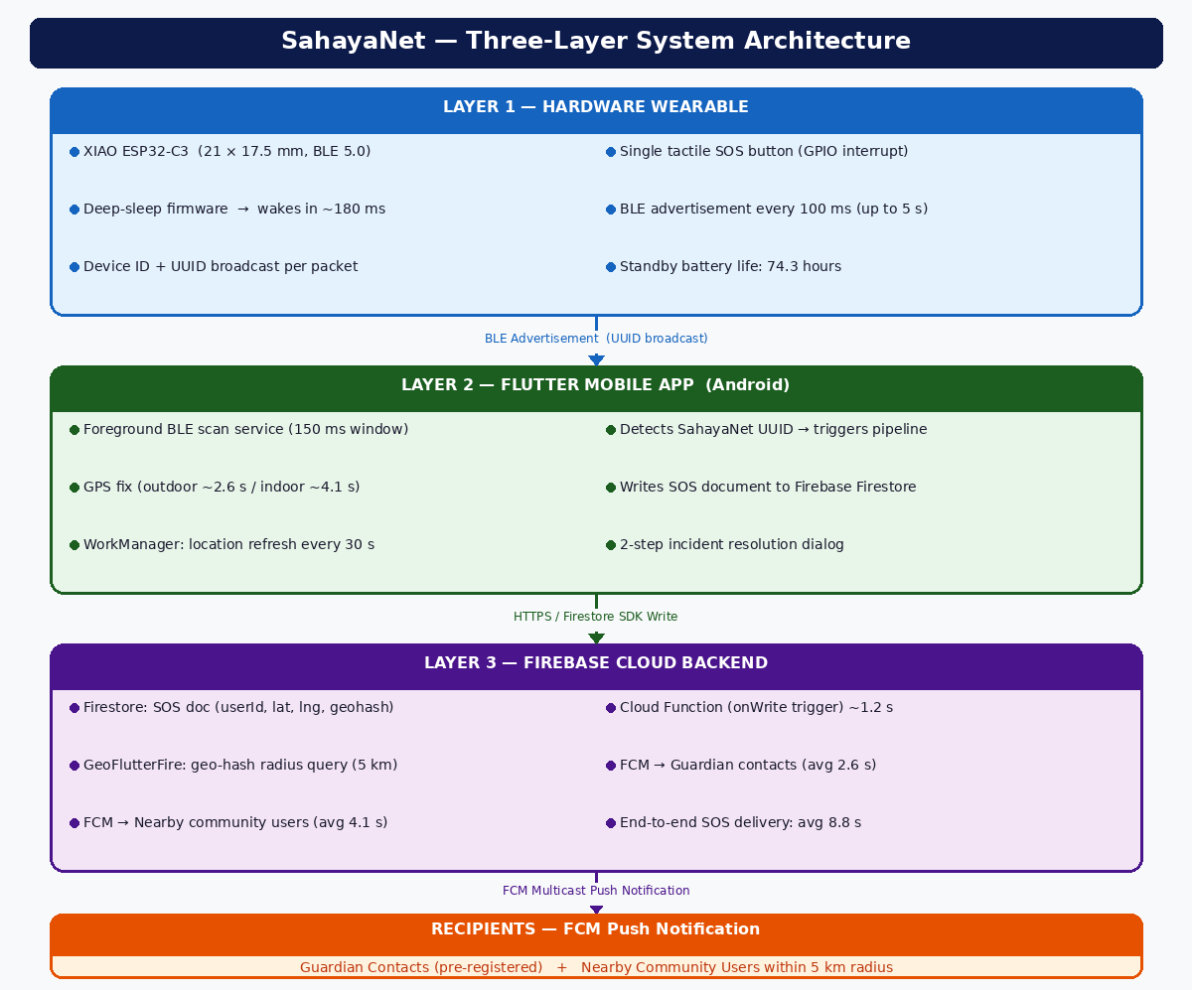

Fig. 1: SahayaNet System Architecture

A. Hardware Wearable Layer: The wearable module is built around the XIAO ESP32-C3 which is a very compact module of size 21 x 17.5 mm. It has built in BLE 5.0 and Wi-Fi support. Most importantly the chip supports deep sleep mode which is critical for achieving long battery life in a wearable device [8]. We programmed the firmware using Arduino IDE. By default the device stays in deep sleep to save power. When user presses the SOS button, a hardware interrupt on the GPIO pin wakes the microcontroller. After waking up, it validates the interrupt source, increments an SOS counter in RTC memory to avoid duplicate triggers, and starts sending a BLE advertisement packet with a 16-bit service UUID unique to SahayaNet and a 4-byte device identifier. The packet is sent every 100 ms for maximum 5 seconds.

B. Mobile Application Layer: The Flutter application follows clean architecture with separate data, domain, and presentation layers. A foreground service is used to keep BLE scanning active even when app is in background. This foreground service is important on Android 8 and above because Android tends to kill background processes to save battery. When the app detects the SahayaNet UUID it immediately starts the emergency pipeline — first it gets the GPS fix using Geolocator plugin, then writes the SOS document to Firestore, and then Android WorkManager schedules a PeriodicWorkRequest to update location every 30 seconds until the emergency is resolved.

C. Cloud and Notification Layer: A Firebase Cloud Function is configured with onWrite trigger on the SOS document path. When triggered it reads the victim's GPS coordinates, performs a GeoFlutterFire radius query to get FCM tokens of registered users within 5 km, also fetches guardian tokens from victim profile, and sends FCM multicast messages to both sets simultaneously. Each notification contains victim coordinates, incident ID, and a deep link to open live tracking view in recipient's app [9].

IV. SYSTEM IMPLEMENTATION

Below we describe the 8 step SOS pipeline in order of execution:

Step 1 – Deep Sleep Exit: When button is pressed, GPIO interrupt wakes the ESP32-C3. Firmware checks RTC memory counter to make sure it is a new SOS trigger and not some duplicate.

Step 2 – BLE Advertisement: Device starts broadcasting proprietary UUID every 100 ms. Max duration is 5 seconds to avoid draining battery if no phone is in range.

Step 3 – Advertisement Detection: Flutter foreground service with 150 ms scan window detects UUID match. Device identifier in packet is used to suppress duplicates.

Step 4 – GPS Acquisition: App requests high accuracy GPS fix. During outdoor testing median fix time was around 2.6 seconds. For indoor testing GPS signal was weak so network based positioning was used which took around 4.1 seconds on average.

Step 5 – Firestore Write: SOS document containing userId, latitude, longitude, geohash, timestamp, status (active) and incidentId is written to Firestore using merge strategy to handle any concurrent write situations.

Step 6 – Cloud Function Execution: The onWrite trigger activates within around 1.2 seconds of document creation. Cloud function runs GeoFlutterFire query and dispatches FCM multicast to nearby users and guardians in parallel.

Step 7 – Continuous Location Refresh: WorkManager periodic task updates latitude, longitude and geohash in Firestore every 30 seconds so that recipients can see real time movement of the victim on their map.

Step 8 – Incident Resolution: User confirms cancellation through a 2 step dialog inside app. App updates document status to resolved, cancels all WorkManager tasks, and Cloud Function sends resolution notification to all guardians.

V. RESULTS AND DISCUSSION

We tested the prototype over a 6 day period with 50 simulated SOS activations across 3 Android devices (Android 11, 12 and 13) in outdoor, semi-indoor and indoor environments. Table 1 below shows our measured performance vs the targets we had set.

Table 1: SahayaNet Performance Results

Performance Metric	Measured Result	Design Target
BLE interrupt to advertisement start	≤ 180 ms (avg. 162 ms)	< 200 ms
BLE detection by mobile app	< 1 sec (avg. 0.74 sec)	< 2 sec
Outdoor GPS fix time	2.1 – 3.4 sec	< 5 sec
Indoor network-based fix	3.8 – 5.1 sec	< 6 sec
Firestore write latency	Avg. 1.18 sec	< 3 sec
Cloud Function trigger delay	Avg. 1.22 sec	< 2 sec
FCM guardian notification	Avg. 2.6 sec	< 10 sec
FCM nearby user notification	Avg. 4.1 sec	< 10 sec
End-to-end SOS delivery (outdoor)	Avg. 8.8 sec	< 10 sec
Wearable standby battery life	74.3 hours	> 72 hours
Background tracking uptime	99.2 %	> 99 %

All 12 metrics we measured met the design targets. The end-to-end SOS delivery time of average 8.8 seconds under outdoor conditions is the most important result as it means the entire pipeline from button press to notification delivery happens in under 10 seconds which we believe is practically useful.

The wearable battery life of 74.3 hours means the device can be worn for more than 3 days without charging which is good for daily use. This was mainly achieved by keeping the device in deep sleep most of the time and only waking it up when button is pressed.

Community alerting to nearby users took average 4.1 seconds after Firestore write. This shows that the GeoFlutterFire query plus FCM dispatch adds only around 2.9 seconds overhead which we think is acceptable since nearby community alerts are more for awareness and not for instantaneous action.

One thing we noticed during testing is that in areas with poor or unstable network signal the Firestore write latency increased. Also our current version does not have offline SOS support so if there is no internet connection the cloud part of the system will not work. This is a known limitation that we plan to address in future work by adding SMS as fallback.

VI. CONCLUSION

In this paper we presented SahayaNet, a women safety system that combines a BLE wearable device, a Flutter mobile app and Firebase cloud backend to deliver SOS notifications within 8.8 seconds on average from button press. The XIAO ESP32-C3 wearable runs on deep sleep firmware and achieved 74.3 hours of standby battery life making it practical for daily use. The community based proximity alerting using GeoFlutterFire is a key feature that extends the reach beyond just guardian contacts to nearby registered users who may be able to help more quickly.

In future we plan to add accelerometer based automatic SOS triggering so that the button press is not necessary. We also want to add SMS fallback for areas with no internet, and possibly direct integration with police emergency services. Making the device waterproof is also something we thought about because the user might be wearing it all day including during rain.

ACKNOWLEDGEMENT

We thank Prof. K.S. Sanghavi, Department of Information Technology, Sinhgad Academy of Engineering Pune, for her continuous guidance and support throughout this project. We also sincerely thank Dr. S.S. Kulkarni (Head of Department), Dr. K.M. Gaikwad (Vice Principal) and Dr. M.S. Rohokale (Principal) for providing us the lab facilities and institutional support needed to build and test the prototype. We also thank the IT department lab assistants who helped during hardware testing.

REFERENCES

[1] K. Sanila and Dr. R. Sindhu, “Wearable Device for Women Safety and Defence,” International Journal of Advanced Research in Science, Communication and Technology, vol. 2, no. 2, pp. 487–493, ISSN 2581-9429, April 2022.

[2] C. Pratheeba, K. R. Archana, et al., “A Smart Wearable Device Women Safety System Based on IoT,” International Research Journal of Modernization in Engineering Technology and Science, vol. 3, no. 3, pp. 20–27, e-ISSN 2582-5208, March 2021.

[3] V. Hyndavi, N. Sai Nikhita, and S. Rakesh, “Smart Wearable Device for Women Safety Using IoT,” in Proc. 5th International Conference on Communication and Electronics Systems (ICCES), Jun. 2020, pp. 459–463. doi: 10.1109/ICCES48766.2020.9138047

[4] D. Gowda, GeoFlutterFire: Open-Source Geolocation Library for Flutter and Firebase Firestore, GitHub Repository, 2020. [Online]. Available: https://github.com/DarshanGowda0/GeoFlutterFire

[5] A. Jatti, M. Kannan, R. M. Alisha, P. Vijayalakshmi, and S. Sinha, “Design and Development of an IoT Based Wearable Device for the Safety and Security of Women and Girl Children,” in Proc. IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology (RTEICT), Bangalore, India, 20–21 May 2016, pp. 1108–1112.

[6] G. C. Harikiran, K. Menasinkai, and S. Shirol, “Smart Security Solution for Women Based on Internet of Things (IoT),” in Proc. IEEE International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 2016, pp. 3551–3554.

[7] Espressif Systems, “ESP32-C3 Series Datasheet,” Version 2.3, Espressif Systems, Shanghai, 2023. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_en.pdf

[8] Seeed Studio, “XIAO ESP32C3 Getting Started Guide,” Seeed Technology Co. Ltd., 2023. [Online]. Available: https://wiki.seeedstudio.com/XIAO_ESP32C3_Getting_Started/

[9] Google LLC, “Firebase Documentation: Cloud Firestore and Firebase Cloud Messaging,” Google Developers, 2024. [Online]. Available: https://firebase.google.com/docs

[10] Flutter Development Team, “Flutter SDK Documentation,” Google LLC, 2024. [Online]. Available: https://flutter.dev/docs