Article

Not peer-reviewed version

# No Pain Device: Empowering Personal Safety with An Artificial Intelligence-Based Nonviolence Embedded System

Agostino Giorgio *

Posted Date: 29 March 2024

doi: 10.20944/preprints202403.1763.v1

Keywords: Personal Safety; Health; Microcontrollers; Arduino; Internet of Things; Artificial Intelligence; Tiny Machine Learning

*Article*

# No Pain Device: Empowering Personal Safety with An Artificial Intelligence-Based Nonviolence Embedded System

**Agostino Giorgio**

Politecnico di Bari, Bari, Italy; agostino.giorgio@poliba.it

**Abstract:** This paper presents the development of a novel anti-violence device titled "no pAIn" (an acronym for Never Oppressed Protected by Artificial Intelligence Nonviolence system), which harnesses the power of artificial intelligence (AI). Primarily designed to combat violence against women, the device offers personal safety benefits for individuals across diverse demographics. Operating autonomously, it necessitates no user interaction post-activation. The AI engine conducts real-time speech recognition and effectively discerns genuine instances of aggression from non-violent disputes or conversations. Facilitated by its Internet connectivity, in the event of detected aggression, the device promptly issues assistance requests with real-time precise geolocation tracking to predetermined recipients, for immediate assistance. Its compact size enables discreet concealment within commonplace items like candy wrappers, or within purpose-built casings, or as wearable accessories. The device is battery-operated. The prototype was developed using a microcontroller board (Arduino Nano RP2040 Connect), incorporating an omnidirectional microphone and Wi-Fi module, all at a remarkably low cost. Subsequent functionality testing, performed in debug mode using the Arduino IDE serial monitor, yielded successful results. The AI engine exhibited exceptional accuracy in word recognition, complemented by a robust logic implementation, rendering the device highly reliable in discerning genuine instances of aggression from non-violent scenarios.

**Keywords:** personal safety; health; microcontrollers; Arduino; Internet of Things; artificial intelligence; tiny machine learning

## 1. Introduction

Gender-based violence refers to acts of violence directed at individuals due to their gender, with women and girls comprising the predominant victims.

This issue remains a significant challenge within our societies and is deeply entrenched in gender disparities.

It serves both as a cause and a consequence of gender inequality, representing a violation of human rights and the most severe form of discrimination based on gender.

According to a survey carried out by the EU Fundamental Rights Agency in 2014 [1], one in three women in the EU has been a victim of violence in her lifetime, and one in 20 women has been raped.

The phenomenon of femicide is a grave social issue that afflicts the entire world.

The statistics provide chilling data into femicides in Italy and globally over the last four years [1–11].

In 2020, there were 116 female victims of intentional homicide in Italy, accounting for 0.38 per 100,000 women.

In 2021, 104 femicides were reported, with 70% of the victims being women killed by their partners or ex-partners.

In 2022, the number of female victims increased to 196 compared to 2021.

In 2023, 120 female victims of intentional homicide were recorded in Italy.

 doi:10.20944/preprints202403.1763.v1



Globally, in 2020, more than five women and girls were killed every hour, totaling 45,000 victims. Shockingly, 56% of the 81,100 homicides worldwide that year involved women and girls killed by their husband, partner, or another relative. The actual number of femicides is likely higher, as at least four out of 10 deaths in 2020 were not officially classified as femicides due to insufficient data.

In 2022, almost 89,000 women and girls were intentionally killed worldwide, marking the highest annual total in the last two decades. This is an increase from 81,100 victims in 2021. Of these, 55% (48,800) were committed by family members or partners.

The impact of violence on victims, even if not killed, is profound, significantly hindering their ability to fully engage in societal activities.

While the human costs are immense, the economic implications of gender-based violence are also staggering. According to estimates by the European Institute for Gender Equality (EIGE), the annual cost of gender-based violence in the European Union amounts to €366 billion. Of this total, violence against women constitutes 79%, equivalent to €289 billion.

These statistics underscore the severity of the gender-based violence issue in Italy and globally. It is crucial to continue raising awareness and taking action to prevent these violent crimes.

One of the most effective ways to mitigate the severity of this phenomenon is undoubtedly prevention.

To this aim, a universally recognized hand gesture was established to signal a request for help from a woman (or a person, in general) in danger.

Unfortunately, there may be instances where the gesture goes unnoticed, or the victim may be unable to mimic it or there are no individuals nearby who can observe the plea for help. Often gender violence takes place within the home, in the absence of other people.

Furthermore, there is a risk of too much precious time elapsing between the moment when a person in danger signals for help through a gesture and the point at which someone noticing the gesture can effectively alert emergency services.

Hence, the author of this paper has conceptualized, designed, prototyped, and tested a fully automated, miniaturized, electronic device powered by artificial intelligence (AI). The device is a sort of electronic gesture of a plea for help in a totally automated mode and with the further advantage of notifying the plea even to people who are physically far from the place of violence and, therefore, would not be able to see the gesture if done by hand.

It is named “no pAIn” (which stands for: Never Oppressed Protected by Artificial Intelligence Nonviolence system).

The “no pAIn” device is an Embedded System designed using the most advanced digital, geolocation and communication technologies available but unique for its behavior and operating mode.

It is miniaturized and can be easily incorporated into a bracelet or concealed within a bag or other accessory used by the persons at risk of violence who wish to carry it.

It promptly discerns situations of serious danger in real-time and, when such situations arise, automatically dispatches requests for help along with geolocation data.

This fully automated device obviates the necessity for any explicit gesture or plea for assistance from the woman, who may find herself incapacitated to do so.

Moreover, it transmits requests for help autonomously regardless of the presence or awareness of other individuals in proximity to the victim, dispatching real-time alerts for assistance in the form of messages, notifications, and emails to multiple remote recipients equipped to offer or coordinate immediate aid.

At present, there is not such a device in commerce. There exists an electronic bracelet mandated for potentially dangerous individuals, which prevents them from approaching a woman. In contrast, the “no pAIn” device is intended for use by potential victims, rather than potential assailants. The existing bracelet, mandated for dangerous individuals, can be likened to a “leash” that keeps them away from the person in danger. Conversely, the new device presented in this article, used by the person in danger, can be likened to a shield or bodyguard, offering protection against aggression.

 doi:10.20944/preprints202403.1763.v1



Furthermore, the existing bracelet requires a complaint from the woman and a judicial sentence (in Italy) for a dangerous individual to be obligated to wear it. However, this legal process is not always effective, as women may not always report aggressive partners, and serious threats can arise suddenly from unfamiliar or completely unknown individuals.

Hence, the novelty and unique benefits of the "no pAIn" device are that it offers greater safety and reliability, functioning without the necessity of obtaining any authorization but instead operating in a completely free and voluntary manner. It is a personal device, managed privately, as the woman utilizes it directly, akin to an invisible bodyguard — a perpetually vigilant yet unseen sentinel safeguarding her health and life. It is fully automated: the AI detects dangerous situations and remotely sends requests for help with geolocation coordinates. No user intervention is needed. Finally, very importantly, an essential benefit of the "no pAIn" device is that it allows a situation of gender violence to be identified at the very first signs and therefore to effectively prevent harmful consequences.

Of course, the existing bracelet and the new "no pAIn" device can be used in conjunction with each other, respectively from the assailant and from the victim; one does not preclude the other.

The aim of this paper is to describe the "no pAIn" device and its project and operation.

Hence, Section 2 provides an overview of widely recognized devices useful for surveillance purposes, demonstrating the originality of the new one; Section 3 delves into the device's architecture and design; Section 4 demonstrates the device in action, and finally, Section 5 presents conclusions and potential avenues for further project development.

## 2. Literature Survey

In scientific literature, there are several works discussing projects and devices utilizing artificial intelligence (AI) for home automation and surveillance purposes.

The review in [12] explores intelligent audio surveillance for public safety, emphasizing the role of artificial intelligence in enhancing security through sound analysis.

The authors in [13] investigate wearable devices for personal safety, providing insights into the integration of AI for enhancing safety features in wearable technologies.

A focus on voice-based emotion recognition using deep learning, offering potential applications in systems emphasizing personal safety can be found in [14].

The work in [15] provides a survey of IoT-based personal safety systems, with a focus on AI integration for effective monitoring and response.

Cloud-based emergency response systems are examined in [16], highlighting the use of AI for efficient communication and coordination during emergencies.

In [17] there is a review of wireless sensor networks-based health monitoring systems, offering insights into the potential integration of AI for personalized safety monitoring.

Paper [18] investigates machine learning applications in public safety and emergency management, presenting a broad view of AI implementations in safety-related contexts.

A survey about mobile-based solutions for personal safety, including AI-enhanced features in mobile applications dedicated to individual security, is in [19].

A comprehensive review of voice emotion recognition, providing potential applications in systems focused on personal safety can be found in [20].

A discussion about wearable technology in personal safety applications, examining the role of AI in enhancing the capabilities of wearable safety devices, is in [21].

The design of a smart home system utilizing voice control and deep learning for enhanced automation is in [22].

The article [23] focuses on a home automation system employing speech recognition with a deep learning approach.

The paper [24] discusses a home automation system integrating speech-based controls with AI.

The paper [25] presents a smart home system incorporating speech recognition and IoT technology for efficient control and communication.

 doi:10.20944/preprints202403.1763.v1



The authors in [26] detail an intelligent home automation system utilizing voice control and Raspberry Pi technology.

The article [27] discusses a real-time home automation system employing voice control with a focus on deep learning techniques.

The paper [28] introduces an IoT-based home automation system integrating intelligent voice control through Amazon Alexa.

The authors in [29] present a smart home automation system utilizing voice recognition and deep learning techniques for improved functionality.

The paper [30] discusses a smart home system incorporating voice recognition and deep learning technologies for advanced automation.

The authors in [31] introduce a speech-based intelligent personal assistant designed for real-time control of home automation tasks.

Despite the extensive existing bibliography regarding intelligent devices, currently, to the best of the author's knowledge, none resemble the "no pAIn" device because there seems to be none with its distinct characteristics, which will be described upon in detail below.

## 3. Device Operation, Architecture, and Project

The “no pAIn” device is an Embedded System, having a microcontroller (mC) as the core of the hardware part, with its firmware part.

It behaves as a virtual "sentinel" employing AI to interpret scenarios in real-time with exceptionally high reliability. It can discern situations with a high risk of violence from more commonplace quarrels and discussions, with non-violent outcomes.

The architecture of the “no pAIn” device is as in Figure 1, which also delineates the dataflow and, consequently, the overall behavior of the device.

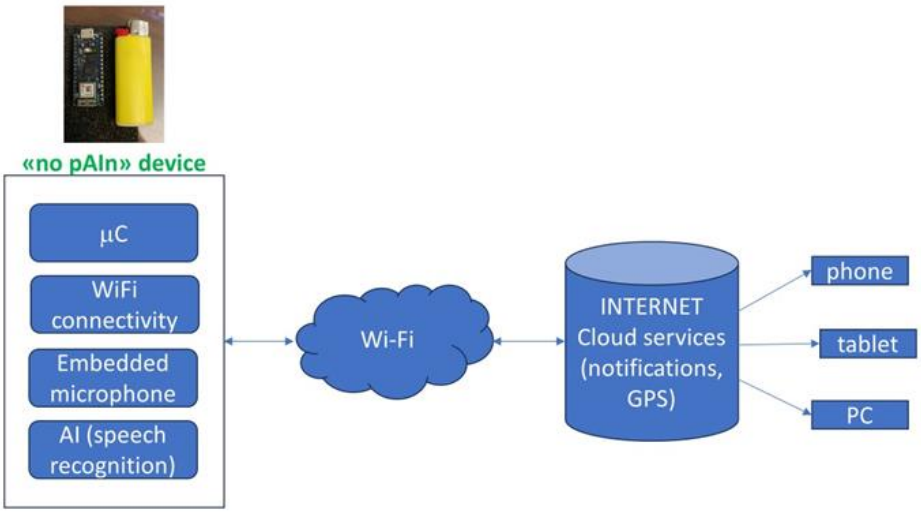


**Figure 1.** System architecture.

Its operation is governed by a firmware program developed in accordance with the flowchart illustrated in Figure 2.

 doi:10.20944/preprints202403.1763.v1



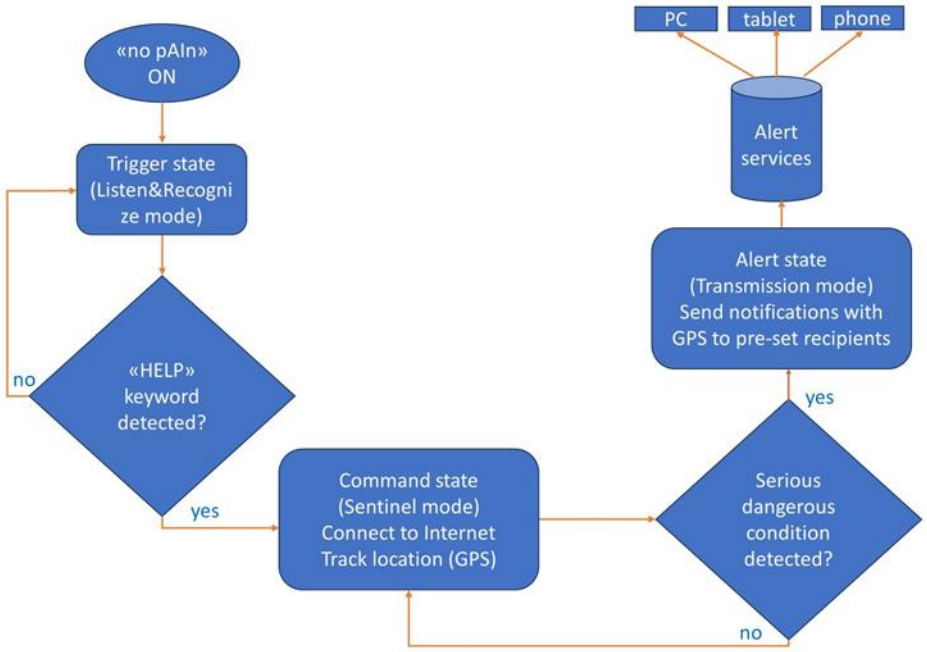


**Figure 2.** Flow chart of the firmware of the "no pAIn" embedded device behavior.

When the device detects a scenario with a high risk of violence, it activates a virtual but effective "shield" to protect the user. This shield involves sending requests for help with geolocation automatically, in real-time, without requiring human intervention, and discreetly and silently (the sending of these requests for help remains unnoticed except by those who receive them).

Requests for help are transmitted in the form of messages and emails directed to the smartphones, tablets, or PCs of pre-defined recipients.

In detail, upon powering on, the device enters the "Trigger" state, initiating the activation of AI which utilizes the device built-in microphone to listen and recognize the words heard ("Listen&Recognize" mode).

All audio data are processed without storage, ensuring compliance with privacy laws. However, incorporating data storage into the firmware code would be straightforward, utilizing either a microSD card or a flash memory integrated into the device board. This would enable the "no pAIn" device to store voice recordings of several minutes on board.

Upon detection of the keyword "help" (or an alternative keyword, customizable based on user preference), the device transitions from the "Trigger" state to the "Command" state, thereby activating the "Sentinel" mode, prepared to detect and respond appropriately to serious danger conditions.

In fact, in the "Command" state ("Sentinel" mode), the device establishes an Internet connection and retrieves GPS coordinates.

Upon identification of a situation deemed as serious danger, based on the subsequent words heard and recognized, the device transitions from the "Command" state ("Sentinel" mode) to the "Alert" state ("Notification transmission" mode).

During the "Notification transmission" mode, the device autonomously sends a sequence of notifications to pre-defined recipients equipped with smartphones, tablets, or PCs. These notifications may consist of emails, Telegram messages, Discord messages, and in-app alerts, containing messages such as "Request for help" or "Likely need for rescue" or any other customizable help request message, accompanied by the geolocation coordinates of the victim.

The "no pAIn" device operates with utmost respect for privacy, abstaining from storing or disseminating any sensitive data, as already stated, except for geolocation information. Nonetheless, the sharing of geolocation data for personal safety with preferred individuals (the recipients of help request notifications) is governed by the implicit consent of users who choose to utilize the device.

 doi:10.20944/preprints202403.1763.v1



The device requires a "one-off" configuration to set, for example, the recipients of emergency notifications and to perform other possible customizations.

The hardware component of the "no pAIn" device primarily comprises a mC and a built-in microphone.

The mC possesses the necessary specifications to store an AI model, trained for speech recognition, in its memory. Additionally, it features Internet connectivity (via Wi-Fi and/or a suitable data SIM card) and geolocation capabilities.

Each of the modules composing the architecture of the "no pAIn" device (refer to figure 1) will be thoroughly described in subsequent sections.

### *3.1. Microcontroller, Hardware and Firmware*

The chosen microcontroller is the 32-bit ARM Cortex M0+ owing to its low power consumption, rendering it suitable for portable, battery-powered applications, alongside its commendable performance aligned with the device's intended functionalities. Furthermore, its capabilities are apt for the integration of speech recognition artificial intelligence.

For prototyping purposes, the Nano RP2040 Connect board, powered by Arduino, and depicted in Figure 1, emerges as the optimal selection. It is noticeable that the dimensions of the prototyping board are very small, about 20x45 mmxmm, smaller than a mini lighter.

The Arduino Nano RP2040 Connect board represents a feature-rich development board in "nano" format, harnessing the capabilities of the RP2040 microcontroller. Noteworthy features of this board are outlined below [32–34]:

- Microcontroller: it uses Raspberry Pi's new silicon, the RP2040, which is a dual-core 32-bit Arm® Cortex®-M0+ running @ 133MHz.
- Memory: it has 264KB of SRAM, and the 16MB of flash memory is off-chip to give the designer an extra storage.
- Connectivity: it supports Internet of Things projects with Bluetooth® and Wi-Fi connectivity, thanks to the U-blox® Nina W102 module.
- Sensors: it comes with an onboard accelerometer, gyroscope (6-axis IMU), temperature sensor, and an omnidirectional microphone.
- LED: it has a built-in RGB LED.
- Machine Learning: it supports machine learning using TinyML, TensorFlow Lite or Edgeimpulse, thanks to the high-performance energy-efficient microprocessor.
- Python® Support: This board can be programmed using MicroPython and Arduino IDE.

This board is suitable for developing robust embedded AI and IoT solutions with minimal effort.

It is noticeable that the Nano RP2040 Connect board is equipped with the MP34DT06JTR embedded, omnidirectional microphone, enabling real-time capture and analysis of sound for a wide range of applications.

These features make the Nano RP2040 Connect board particularly suitable for the "no pAIn" project prototyping:

### *3.2. Internet Connectivity*

The "no pAIn" device can connect to the Internet in two ways.

Indeed, it can be outfitted with either its own SIM card and corresponding host module or it can establish Wi-Fi connection to Internet with a hotspot via a Wi-Fi module.

The Arduino Nano RP2040 Connect board used for prototyping is equipped with the U-blox® Nina W102 module. Certainly, both modules - the SIM and the Wi-Fi - may be present concurrently.

The mC manages the connectivity modules (either SIM or Wi-Fi), activating the Internet connection as needed with simple dedicated instructions.

Specifically, the Internet connection is established only when the device switches from the "Listen&Recognize" mode to the "Sentinel" mode, to optimize the power consumption.

In the Wi-Fi connectivity scenario, implemented in the device prototype, there are two further options.

 doi:10.20944/preprints202403.1763.v1



One is that the device is programmed to connect to the user's router (if at home) or to the user's smartphone hotspot, both having known SSID name and password.

The second option is for the mC to scan the surrounding Wi-Fi networks (many public places currently offer free Wi-Fi connections without a password) and connect to the nearest open hotspot.

Both options have been successfully assessed in the "no pAIn" device prototype.

The mC initially checks for the presence of an encrypted connection with the router or with the user's smartphone hotspot. If such a connection is not detected, it proceeds to search for the closest non-encrypted Wi-Fi network to establish a connection with.

The credentials (SSID and password) required for connecting to the user's encrypted network are preconfigured during the device setup phase.

From a programming standpoint, the Wi-Fi management libraries provided for Arduino streamline the procedure of scanning for available networks, identifying the SSID, assessing signal strength, determining encryption type, and allowing automatic connection.

Functions of the Wi-Fi library (WiFiNINA for the U-blox® Nina W102 module on the RP2040 Connect board) as:

```
WiFi.scanNetworks();   //scanning available networks
WiFi.RSSI();   // measure the strength
WiFi.SSID();   //determine the SSID
WiFi.encryptionType();   // determine encryption type
WiFi.begin(<NetworkSSID>);   //connect
```

are suitable for implementing the appropriate logic and corresponding code, useful for this purpose.

### *3.3. Artificial Intelligence Engine*

The use of AI by mC -based Embedded Systems can occur in two modes: cloud-based and on-board. The strategy that entails the deployment and utilization of AI directly on the device is known as Tiny Machine Learning (TinyML) for edge computing.

Both approaches offer distinct advantages and disadvantages.

Specifically, the benefits of the on-board implementation are as follows.

Data are processed directly on the device, reducing the need to transfer data over the network.

As AI runs locally, response times are faster compared to sending data to a remote server for processing and wait for the outcome of the processing in response.

Sensitive data need not be shared over the network, reducing the risk of privacy breaches and unauthorized access.

Constant Internet connection is not required, thereby reducing the power consumption of the device.

On the contrary, the disadvantages are as follows.

Microcontroller devices may have limited resources such as memory and computing power, limiting the complexity and size of AI models that can be run.

AI model updates need to be manually distributed to devices, making software management and updates more complex.

Integrating on-board AI models may require investments in more advanced hardware, which can be more expensive compared to devices relying solely on cloud services.

Cloud-based AI services offer numerous benefits. They are capable of processing vast amounts of data and provide scalable computing resources for handling AI processing requests.

Furthermore, AI model updates can be centrally distributed to devices, streamlining software management, and guaranteeing consistent access to the latest model versions.

Cloud data processing can provide access to external resources such as large datasets and pre-trained models, improving AI performance.

 doi:10.20944/preprints202403.1763.v1



Finally, there is no need to invest in advanced hardware to run AI, as processing occurs in the cloud.

On the contrary, the disadvantages of the cloud-based AI services are as follows.

Device operation relies on Internet connectivity, which can pose issues in case of unstable or absent connectivity.

Sending data to a remote server may raise concerns about security and privacy, especially for sensitive data.

The time required to transmit data to the cloud server and receive a response can cause delays in AI system responses, especially in situations where speed is critical.

In summary, on-board AI usage offers advantages such as increased speed, privacy, and reliability, but may be constrained by available hardware resources. On the other hand, cloud service usage offers scalability, ease of update, and access to external resources, but may be affected by network connectivity and raise concerns about security and privacy.

The choice between the two approaches will depend on the specific project requirements and technical limitations.

Given the features and objectives of the "no pAIn" project, an on-board approach has been selected, primarily due to its distinct advantages in terms of privacy and speed. Furthermore, the implemented on-board AI engine, which will now be described, meets the exceptionally high accuracy requirements, rendering the utilization of a cloud-based AI service unnecessary and potentially detrimental to the device's operational speed.

Currently, there are various methods for creating TinyML algorithms, including using the Google Colab platform to develop TensorFlow Lite models, utilizing Python libraries [35], employing MATLAB environment [36], and utilizing the Edgeimpulse platform [37], which is highly efficient.

Developing a bespoke AI system with exceptional classification accuracy necessitates the initial creation of a training dataset of significant size, followed by AI training durations proportional to the dataset's magnitude. Therefore, where feasible, it is advisable to utilize pre-existing libraries, meticulously developed and readily accessible to designers for expedited implementation. In the specific context of the "no pAIn" project, a library was searched that precisely aligned with the device's specifications, ensuring both accuracy and functionality.

Therefore, since the “no pAIn” device is prototyped on the Arduino Nano RP2040 Connect board, it was deemed that the most efficient way to implement the AI on-board was using the Arduino Speech Recognition Engine library [38], which boasts exceptionally high accuracy and other very interesting characteristics, as described in the following.

Speech Recognition Engine is an extensive software library that allows anyone to interact with devices and machines quickly and easily by talking. It was developed by Cyberon to be part of Arduino Pro’s growing ecosystem of advanced professional solutions. It can be applied to different industrial and professional fields, from speech dictation to command-voice controllers, health monitoring, robotics, accessibility, among many others.

The Speech Recognition Engine is compatible with multiple Arduino boards, including the RP2040 Connect, and integrates seamlessly with the Arduino IDE, requiring no additional hardware, software, or internet connectivity. It operates entirely on-board.

This engine comprehends commands specified through text input in over 40 languages, irrespective of the speaker's voice, intonation, or accent. Consequently, multiple wake-up words and sequences can be swiftly configured without the necessity for retraining for different users. Therefore, in addition to its high accuracy, the primary advantage lies in its inherent lack of requirement for training. Furthermore, it possesses the capability to discern the speaker's voice amidst background noise, rendering it suitable for deployment in busy or crowded environments.

In summary, key benefits include [38]:

- Recognition of multiple words and sequences of commands by a phoneme-based modeling.
- No vocal training required; words to be recognized configurable through text input (text-to-speech, TTS, technology).
- Support for over 40 languages, independent from accent variations.

 doi:10.20944/preprints202403.1763.v1



- One configuration for one language, multiple speakers, without retraining.
- Recognition on the edge, no need of additional HW/SW or connectivity.
- Suitable for noisy environments.
- Compatible with multiple Arduino Nano and Portenta products.
- Compatible with Arduino IDE.

The Speech Recognition Engine is offered in both free and paid versions. While the paid version incurs a nominal cost (approximately $9 as of today), the free version includes limitations that do not compromise the performance criteria specified for the design of the “no pAIn” device described herein. Therefore, the utilization of the free version is deemed acceptable.

Following the integration of the Speech Recognition engine into the device, it becomes feasible to develop the Arduino sketch to execute customized tasks in response to recognized speech.

The process of embedding the Speech Recognition Engine into the "no pAIn" prototype to enable the recognition of custom words/commands is outlined as follows.

First, it is necessary to complete a series of steps to register the board and activate the trial and free-of-charge license, as detailed in [38]. Subsequently, custom voice commands can be created.

To accomplish this, it is essential to utilize the Cyberon Model Configuration webpage [39] to establish a new project with custom voice commands, following the provided procedure. After completing the required fields, a new project must be created, and the desired language for speech recognition selected.

Next, the command list needs to be defined.

Upon completion of the model configuration process, the model header file (model_ID.h) and the license file (CybLicense_ID.h) created must be incorporated into the Arduino sketch using the #include directive.

So, the AI engine is embedded in the device.

### *3.4. Geolocation Methods*

Several geolocation methods can be employed to track a device equipped with an Internet-connected μC.

Employing a dedicated GPS module managed by the mC is the most conventional and accurate approach, enabling the reception of signals from GPS satellites to determine accurately latitude, longitude, and altitude.

However, alternative geolocation methods exist beyond the direct utilization of a GPS hardware module.

In selecting the optimal method for implementing device tracking for the "no pAIn" project, various solutions have been implemented and evaluated.

The direct use of a hardware GPS module connected to the mC of the "no pAIn" device has been implemented and tested using the Adafruit Mini GPS PA1010D module [40], as depicted in Figure 3.


**Figure 3.** GPS module.

This module is particularly beneficial for the "no pAIn" device project due to its specific features. The Adafruit Mini GPS PA1010D is a compact module (~25mm x 25mm) equipped with both I2C and UART interfaces, providing flexibility for the designer to select the interface that best aligns with their requirements.

 doi:10.20944/preprints202403.1763.v1



The module supports GPS, GLONASS, GALILEO, QZSS. It has a sensitivity of -165 dBm and can provide up to 10 location updates per second. It can handle up to 210 PRN channels with 99 search channels and 33 simultaneous tracking channels. The design is 5V friendly and only draws 30mA of current. It's breadboard-able, with 4 mounting holes.

The module has a RTC battery-compatible, PPS output on fix ±20ns jitter, an internal patch antenna, and a low-power and standby mode with WAKE pin.

In the "no pAIn" project, the I2C interface has been used. The library "TinyGPSPlus" makes very simple the Arduino code to obtain all satellite signals and information.

The test confirms that the device's performance aligns perfectly with the specifications outlined by the manufacturer. It swiftly establishes connections with satellites and commences receiving data within a matter of seconds or less. It is the best solution in terms of accuracy of the geolocation, for outdoors applications.

However, there are some drawbacks.

The primary limitation persists due to the requirement for unobstructed visibility of satellites, which poses challenges in reliably acquiring data within indoor environments. Furthermore, although the miniaturization of the Adafruit Mini GPS PA1010D module eliminates the need for an external antenna, its inclusion still enlarges the device, counteracting the necessity for compactness to ensure discreet concealment. Finally, the addition of a hardware module increases the cost of the "no pAIn" device.

Hence, alternative approaches for geolocation have been evaluated, leveraging solely the Wi-Fi module, to circumvent the need for supplementary hardware like a physical GPS module. This strategy aims particularly at achieving optimal performance in indoor environments while concurrently reducing both the cost and dimensions of the "no pAIn" device.

These solutions are based on the usage of cellular networks, Wi-Fi networks, hotspot, and access points (APs), and Bluetooth Low Energy (BLE) beacons information [41–43].

Cellular Network-Based Positioning (GSM, LTE, 5G) utilizes cellular networks for location estimation. It is measured the signal strength from nearby cellular towers. This method is reliable for both indoor and outdoor scenarios, but the device to be tracked must support cellular connectivity.

The BLE and Wi-Fi positioning method is efficient for environments with multiple Wi-Fi APs and BLE beacons. It is suitable for indoor and outdoor applications. Location accuracy improves if the device to be tracked also supports cellular connectivity.

The approach entails scanning nearby Wi-Fi networks and BLE devices to collect information such as network names, MAC addresses of routers and BLE devices, and signal strength (which is inversely related to distance). If the device also has cellular connectivity capabilities, data regarding nearby cellular towers are included. Then, this data is transmitted in JSON format to a geolocation service, such as Google Geolocation, via an HTTPS call to the service's API.

The geolocation service maintains a database containing MAC addresses, router names, and beacon names along with their corresponding locations. Consequently, it provides the position (geolocation) corresponding to the data collected by the device being tracked.

All these operations can be executed rapidly, nearly in real-time, by a mC connected to the Internet and outfitted with a Wi-Fi module and BLE module, such as the Arduino Nano RP2040 Connect board utilized in this project.

The precision of geolocation using this method is highly variable, contingent upon numerous factors, primarily the quantity of collected data. It consistently provides an approximate estimation of the position, with accuracy ranging from a few meters to tens of kilometers. The accuracy of this method is generally deemed acceptable particularly in indoor settings, where it performs well. In outdoor scenarios, it emerges as the preferred method only when a GPS hardware module is unavailable on the tracked device.

An alternative geolocation technique, which circumvents direct dependence on a GPS module, utilizes IP address data.

Several services cater to this purpose [44–48]. It relies on HTTP or HTTPS calls to two distinct services: the initial service retrieves the public IP address of the target device, which must be

 doi:10.20944/preprints202403.1763.v1



connected to the Internet via a Wi-Fi hotspot [47,48]; the subsequent service is invoked by supplying the obtained public IP address and returns the corresponding geolocation [44–46]. However, this method has been evaluated in the "no pAIn" project and has shown limited accuracy, often resulting in geolocations within a radius of several hundred kilometers.

The final alternative approach to geolocation, explored within the scope of the "no pAIn" project, involves leveraging the geolocation capabilities of smartphones.

Smartphones commonly integrate GPS modules along with highly precise geolocation algorithms designed for accurate tracking in both indoor and outdoor environments. Thus, the concept involves the "no pAIn" device utilizing geolocation data from a smartphone.

For this to occur, it is essential for both the "no pAIn" device and the smartphone to be connected to the same cloud service. Periodically (the frequency is customizable by the designer), the smartphone uploads the acquired geolocation coordinates to the cloud, which can then be retrieved by the device as necessary.

This solution is notably swift and accurate, capitalizing on the precise geolocation capabilities of modern smartphones indoors and outdoors. Moreover, it reduces hardware requirements since the "no pAIn" device does not require a physical GPS module.

Hence, for the purpose of prototyping the "no pAIn" device, a geolocation technique based on the collaborative use of smartphones and cloud technology has been chosen.

An effective tool for implementing this design solution may be the OwnTracks service [49], which offers a smartphone application and a cloud server. The smartphone application automatically launches upon smartphone startup, operates in the background, and records GPS coordinates on the OwnTracks server. These coordinates are then accessible to the device, in cloud, when needed.

No user intervention is required for either recording or retrieving the geolocation coordinates.

The same concept can be easily implemented in the “no pAIn” device using OwnTracks or any other smartphone application that is automatically launched upon smartphone startup. Operating in the background, this app periodically transmits GPS coordinates to a cloud service from the smartphone. The μC embedded within the "no pAIn" device can retrieve the location data by connecting to the designated cloud service and reading them. This process is fully automated, highly accurate, and fast, requiring no user intervention. It involves minimal instructions in the Arduino sketch to retrieve geolocation coordinates from the cloud. This results in an optimization and reduction of the memory and time resources utilized by the “no pAIn” device to achieve precise geolocation in any situation.

For the smartphone application, the author has chosen a proprietary solution by developing a minimalistic app using the Simulink Android support package. Consequently, the corresponding choice for the cloud service is ThingSpeak [50], powered by The MathWorks Inc., which aligns with Simulink.

The developed application, named "GPSsender," is designed to be minimalistic, ensuring its invisibility during operation on the smartphone while also minimizing the consumption of smartphone resources, particularly the battery. Its sole function is to periodically record GPS coordinates from the smartphone and transmit them to the ThingSpeak cloud.

To accomplish this, the initial step entails creating a channel within the ThingSpeak cloud where GPS coordinates can be stored and retrieved. Within the channel, three fields are designated: one for latitude, another for longitude, and a third for altitude, respectively.

The architecture of the Simulink model, which is then automatically translated into an app installable on smartphones via MATLAB environment, is very simple, as shown in Figure 4.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 March 2024 doi:10.20944/preprints202403.1763.v1



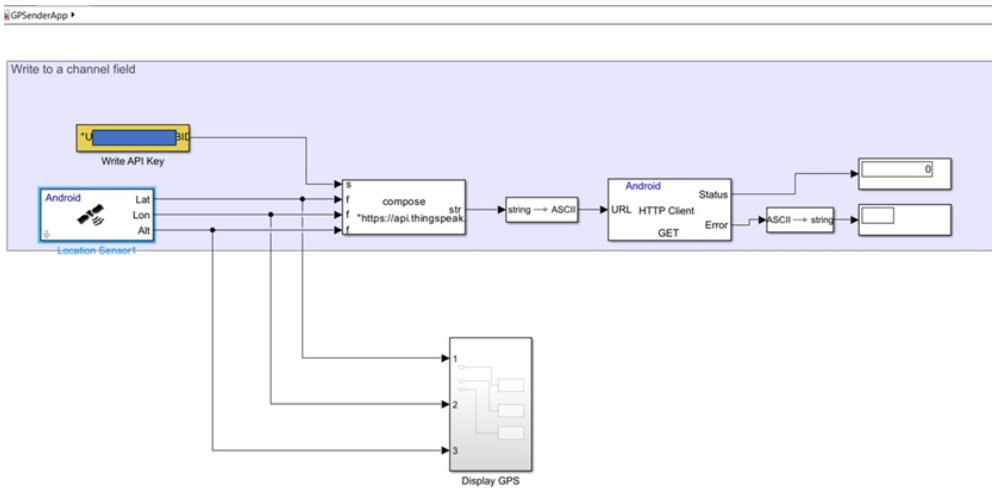


**Figure 4.** Simulink model of the GPSsender app for android devices.

The GPS signal is acquired via the "Location Sensor1" block integrated within smartphones. Subsequently, the "compose" block effectively creates a URL string that includes the acquired GPS coordinates and the "write API key", enabling writing operations in the specifically created ThingSpeak channel.

The format of the URL string is: https://api.thingspeak.com/update?api_key=%s&field1=%f&field2=%f&field3=%f

The HTTPS call's URL explicitly conveys the values of the three GPS coordinates, which the GPSsender app writes into their corresponding fields through the HTTPS call.

The "HTTP Client" block initiates an HTTPS call using the GET method, which records acquired GPS coordinates into the ThingSpeak channel. The "GPS Display" block, as depicted in Figure 4, is not essential for the proper operation of the app but serves solely for debugging purposes. Its function is to validate the accurate acquisition of GPS coordinates. On the smartphone where the app runs, no visible indication is present. The user of the "no pAIn" device (which is also the owner of the smartphone) is the only person to be aware of its operation having expressly agreed to its execution.

When needed, the mC within the "no pAIn" device, connected to Internet, retrieves the GPS coordinates from the corresponding channel fields by issuing an HTTPS GET request to the ThingSpeak channel, utilizing the appropriate "read API keys".

The format of the URL is: https://api.thngspeak.com/channels/<channelID>/feeds.json?api_key=<XXXXXXX>&results=2

Location data are retrieved by the "no pAIn" device through an HTTPS GET call in JSON format.

```
void GPSread() { //function to retrieve location data
  if (!client.connect(host, httpsPort)) {
    Serial.println("connection failed");
    return;
  }
  String url = "/channels/YYYYY/feeds.json?api_key=XXXXXX&results=2";
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
// Reading data from ThingSpeak
String response = "";
// Wait for response
while (client.connected()) {
  if (client.available()) {
    String response = client.readString();
```

 doi:10.20944/preprints202403.1763.v1



```
    }
  }
    client.stop();
  }
```

Subsequently, these data are unpacked by the µC and then attached to the alert notifications.

Indeed, when the AI recognizes the wake-up keyword ("help" in our project) that transitions the "no pAIn" device from the “Trigger” state to the “Command” state, the Wi-Fi Internet connection is suddenly established and the GPS coordinates are promptly retrieved from the cloud (or from the physical module, if available). This ensures that the device is prepared to transmit real-time location data along with help notifications upon detecting a danger. By default, the coordinates are acquired every 5 seconds, although the time intervals can be customized as desired.

It is noteworthy that although there exists a ThingSpeak library for Arduino, it currently does not function on the ARM M0+ microcontroller, which is the microcontroller utilized in the Arduino Nano RP2040 Connect board employed for creating the "no pAIn" prototype. Consequently, the Arduino code for reading GPS coordinates from the ThingSpeak channel was specifically developed without reliance on that library.

The "no pAIn" device can benefit from employing both the direct acquisition of GPS coordinates from a physical module connected to the device and retrieving them from a cloud updated by a smartphone app. Simultaneous utilization of these two methods provides redundancy, thereby enhancing the device's reliability. However, it's important to remark that while the direct use of a physical GPS module increases the device's size, power consumption, and cost, leveraging cloud technologies in conjunction with a smartphone app in the "no pAIn" prototype mitigates these limitations.

### *3.5. Alert Delivery Methods*

While the "no pAIn" device remains in its “Trigger” state, it operates by actively listening and utilizing its built-in AI engine to classify spoken words.

When the "no pAIn" device detects the keyword "help" and transitions from the “Trigger” state to the “Command” state, that is the “Sentinel” operating mode, as previously explained and depicted in figure 2, it automatically establishes an Internet connection and acquires geolocation coordinates as a precautionary measure.

Subsequently, while in “Sentinel” mode, if the device identifies speech indicating a state of danger based on recognized words, it transitions to “Alert” state (or “Notification transmission” mode) and initiates requests for assistance to predefined recipients.

To streamline this feature, several approaches for sending alert notifications were evaluated to identify the most suitable options for integration into the "no pAIn" prototype.

These methods are categorized into two groups, as illustrated in Figure 5: direct calls to alert services from the device (via multiple calls from the mC) and alert service calls routed through cloud services triggered by a single mC call.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 March 2024 doi:10.20944/preprints202403.1763.v1



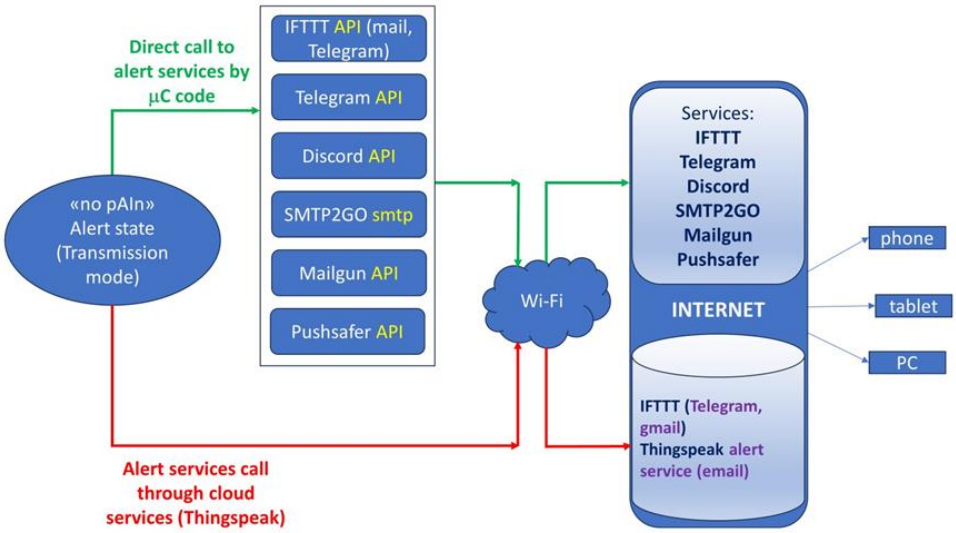


**Figure 5.** Alert methods have been implemented and tested to determine the optimal choice for prototyping purposes.

In the former scenario, direct calls were executed by the "no pAIn" device mC as follows: for email dispatch, an API call was made to "Mailgun" [51], and SMTP protocol was utilized with the "SMTP2GO" [52] service. Moreover, for message dissemination, HTTPS calls were placed to the APIs of "Pushsafer" [53], "Telegram" [54], and "Discord" [55], respectively. Furthermore, a single HTTPS call initiated by the mC to the API of IFTTT [56] enabled the transmission of email and Telegram messages. Specifically, the configured IFTTT service sends an email and Telegram message using webhooks and associated applets, as will be elaborated upon later.

Conversely, in the latter scenario of alert service calls routed through cloud services triggered by a single mC call, the "no pAIn" device simply transmits an alert code to a cloud service, ThingSpeak in the current project, where have been configured an alert channel (to store the alert code) and two associated ThingSpeak apps: "reaction" and "MATLAB Analysis".

As will be detailed later, upon receiving the alert code sent from the “no pAIn” device, ThingSpeak initiates a "reaction" process, which involves the automatic execution of a MATLAB code written in the "MATLAB Analysis" application, disseminating notifications through various services, as IFTTT and ThingSpeak alert service.

Hence, the utilization of the IFTTT service has been evaluated from the "no pAIn" device, both via direct API calls from the mC and through MATLAB code integrated within the "MATLAB Analysis" application. The triggering mechanism involves a ThingSpeak "reaction" in response to the alert code transmitted by the mC.

In every alert notification, both messages and emails include attached location coordinates along with a text indicating a request for assistance.

Further specifics regarding each alert delivery solution are outlined in the subsequent subsections.

### 3.5.1. IFTTT, Webhooks and Applets

IFTTT is a platform that enables the creation of conditional command sequences, referred to as "applets," connecting various services and devices.

Through the utilization of webhooks, users can conveniently configure IFTTT to issue notifications, emails, and Telegram messages in response to specific events. The operational mechanism is illustrated in Figure 6.

 doi:10.20944/preprints202403.1763.v1



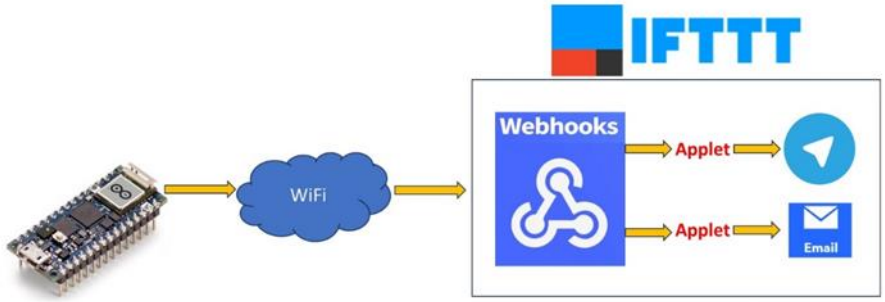


**Figure 6.** Flow chart illustrating the functioning of IFTTT (If This Then That).

A webhook, commonly utilized to trigger various cloud services, is literally termed as a "web hook." It enables the real-time transmission of information from one application to another through an HTTP request triggered by a specific event. In essence, it serves as a means for web applications to communicate autonomously, exchanging data automatically in response to predefined events without requiring human intervention.

Therefore, webhooks are customizable mechanisms activated by an event to transmit data to a preconfigured URL [57,58]. They enable the automation of tasks across different web applications. For instance, upon updating a database, a webhook can automatically trigger a notification or update another application.

Webhooks enhance software efficiency by facilitating immediate responses to specific events, such as order updates or message receipt. Given their simple structure, are easy to access even for less experienced programmers.

Due to their features, webhooks find widespread utility in integrating and automating interactions between various online services, including IFTTT, content management systems, e-commerce platforms, and more.

To integrate an IFTTT service into an Embedded System-based project, a few configurations need to be performed.

Within the personal IFTTT account, users are instructed to search for and activate the webhooks service. Upon activation, a unique URL is provided for triggering events. Subsequently, users must create one or more applets by selecting "Create" and then "This" to specify the trigger. "Webhooks" is chosen as the service, with "Receive a web request" designated as the trigger type. Users then input the event name that will activate the applet.

Once the trigger is configured, the user proceeds to select the action by clicking on "That." To send an email, the "Email" service must be selected and the requisite details input. For sending a Telegram message, the "Telegram" service is chosen, and the recipient's Telegram account is connected. The action is then configured with the message details.

To execute the applet, users must send an HTTP (or HTTPS) POST request to the IFTTT webhook URL with the configured event name. This can be accomplished via a browser, script, or any application supporting HTTP(S) requests.

The format of the URL is: https://maker.ifttt.com/trigger/NAME/with/key/API-KEY

For the "no pAIn" device, the HTTPS POST request can be directly sent through proper Arduino code or delegated to a MATLAB code executed as part of a "reaction" when an alert code is transmitted by “no pAIn” to the ThingSpeak alert channel in cloud.

Once the event is triggered, IFTTT executes the configured action, such as sending an email, Telegram message, and so. In the "no pAIn" project, one webhook service and two applets are configured. Upon receipt of the webhook, one applet dispatches an email via Gmail provider, while the other sends a Telegram message. In both instances, the device's previously acquired geolocation coordinates are included in the notifications.

There are also a couple of Arduino libraries available to facilitate the writing of code for making an IFTTT POST call (IFTTTMaker library and IFTTTWebhook library), but neither library currently

 doi:10.20944/preprints202403.1763.v1



works well for the ARM M0+ microcontroller of the RP2040 Connect board. Therefore, the code for the HTTPS call has been written from scratch and functions perfectly.

### 3.5.2. E-Mail Direct Sending

The transmission of email messages has been evaluated using both cloud-based methods and direct transmission by the "no pAIn" device's μC.

Email sending via the IFTTT applet has been previously explained.

The email sending using the alert services of ThingSpeak will be further detailed in the next subsection.

In this subsection, the methods for directly sending emails by the mC of the "no pAIn" device are detailed.

The first method utilizes the Simple Mail Transfer Protocol (SMTP) and relies on the SMTP2GO service [52]. This service facilitates email sending, including through Arduino projects. The process entails configuring the Arduino to connect to the SMTP2GO server using SMTP protocol. Once connected, the μC of the Arduino board can transmit emails by supplying crucial details including sender address, recipient address, subject, and message content. SMTP2GO serves as an intermediary between the Arduino and the recipient's email server, ensuring reliable delivery. This simplifies the integration of communication and notifications into various applications.

The second method assessed is based on the "Mailgun" service [51]. This service is an efficient email management tool, particularly suitable for Arduino use as it allows sending email messages to multiple recipients simultaneously via a straightforward HTTPS call using the API. The service call is executed with a simple Arduino code, primarily comprising instructions for performing an HTTPS call.

The procedure for utilizing this service is remarkably straightforward.

Upon creating an account on the service's website, a domain is assigned for use as the sender of the emails. Following this, users select the preferred technology, whether SMTP protocol or the HTTPS call to the API (chosen in this project) and generate the "Sending API key." This key serves as the "token" to be incorporated into the Arduino code.

```
String request = "/v3/" + String(mailgun_domain) + "/messages";
String authorization = "Basic " + base64Encode("api:" + String(mailgun_token));
String text = "Probabile bisogno di soccorso in http://maps.google.com/maps/place/" + String(latitude, 6) + "," + String(longitude, 6);
String body = "from=" + urlEncode(from) + "&to=" + urlEncode(to) + "&subject=" + urlEncode(subject) + "&text=" + urlEncode(text);
httpClient.beginRequest();
httpClient.post(request);
httpClient.sendHeader("Authorization", authorization);
httpClient.sendHeader("Content-Type", "application/x-www-form-urlencoded");
httpClient.sendHeader("Content-Length", body.length());
httpClient.beginBody();
httpClient.print(body);
httpClient.endRequest();
```

### 3.5.3. ThingSpeak Cloud Service

Sending notifications via ThingSpeak operates according to the flowchart depicted in Figure 7.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 March 2024 doi:10.20944/preprints202403.1763.v1



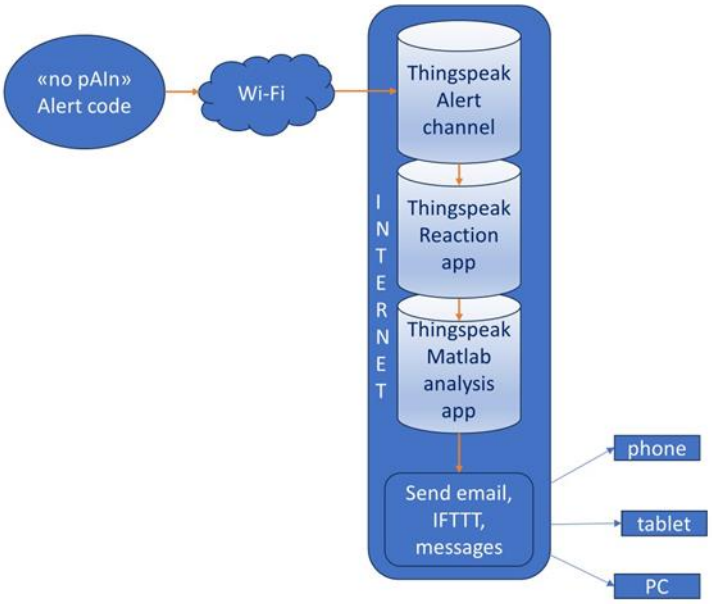


**Figure 7.** Flowchart illustrating the process of sending notifications via cloud service using ThingSpeak.

Initially, an alert channel is created in the user's own ThingSpeak account, where the “no pAIn” device inputs the alert code. The device transmits the alert code via an HTTP GET call to ThingSpeak, employing the appropriate “write API key”.

Subsequently, the creation of a "React" type app, or a “reaction,” is necessary. This reaction is triggered when the device transmits the alert code to the alert channel by a simple function.

```
void thingspeakReac() {
if (client.connect(host, 80)) {
        String url = "/update?api_key=";
        url += writeApiKey;
        url += "&field1=";
        url += alertCode;
        client.print(String("GET ") + url + " HTTP/1.1\r\n" + "Host: " + host + "\r\n" +
"Connection: close\r\n\r\n");
        delay(100);
        client.stop();
      }
}
```

The reaction involves executing a "MATLAB analysis," constituting the third element to be established in ThingSpeak account. Within the "MATLAB analysis," MATLAB code is authored to execute upon detection of the alert code in the designated channel.

It is important to note that in this scenario, the mC of the “no pAIn” device operates swiftly, solely focusing on writing the alert code to the dedicated ThingSpeak channel. Consequently, all tasks related to sending notifications are handled by the reaction within ThingSpeak cloud. As a result, the "no pAIn" device can promptly resume its “Sentinel” function, as it is not occupied with sending notifications.

Upon execution, the code of the "MATLAB analysis" app entails:

- Sending an email via the ThingSpeak alerts service, through an HTTPS call. The URL utilized is:
  https://api.thingspeak.com/alerts/send
  and the corresponding MATLAB command is:

 doi:10.20944/preprints202403.1763.v1



webwrite(alertUrl , "body", alertBody, "subject", alertSubject, options);

- Making the HTTPS call to the IFTTT service, which subsequently dispatches notifications via Gmail and Telegram (configured as actions within the IFTTT applets). The URL for this operation is: https://maker.ifttt.com/trigger/PainAI/with/key/<yourkey> and the corresponding MATLAB command is:
  webwrite(alertUrl,'value1', queryString);

The GPS coordinates, retrived by the “no pAIn” mC as previously explained, are always included in the “queryString” or “alertBody” of the alert notifications.

If the physical GPS module is present as a part of the “no pAIn” device, the mC transmits the geolocation coordinates to the appropriate ThingSpeak channel, along with the alert code. However, if the GPS coordinates are written into ThingSpeak location channel by the smartphone, their retrieval is performed using the appropriate MATLAB code included in the "MATLAB Analysis" app and included in the body of the notifications to be sent.

### 3.5.4. Telegram Messages

Telegram [54] is a free-of-charge instant messaging and voice-over-IP service accessible via web clients and native applications for various operating systems.

It allows users to exchange messages, photos, videos, stickers, and files, as well as create groups and channels for broadcasting to unlimited audiences.

Telegram has significance advantages over similar platforms.

Indeed, Telegram places a strong emphasis on privacy and security by providing end-to-end encryption for secret chats, optional two-factor authentication, and self-destructing messages. Additionally, Telegram offers unlimited cloud storage for messages and media, enabling users to seamlessly access their data from multiple devices. It supports the transmission of large files up to 2 GB, making it well-suited for sharing high-quality media files without compression.

Moreover, Telegram's Bot API empowers developers to create robust and interactive Bots for a variety of purposes, ranging from customer service to automation and entertainment. Bots are complete programs that mainly deal with user interaction, receiving commands and consequently carrying out operations, according to the instructions given to them in the development phase. Bots can also integrate the use of webhooks to access additional features.

Given Telegram's focus on privacy and encryption, as well as the availability of the Bot API, it emerges as a compelling solution for the "no pAIn" project.

As for sending messages via Telegram using Arduino RP2040 Connect, it involves using the Telegram Bot API.

Developers can create a Telegram Bot and obtain an API token, which they can then use to send messages programmatically. The recipients of the messages are identified by means of their own “chatID”.

Creating a Telegram Bot entails some steps, as follows.

Initiate a conversation with the BotFather (@BotFather) in Telegram. Use the "/newbot" command to create a new Bot and follow the prompts to set a name and username. After Bot creation, BotFather provides an API token essential for interaction with the Telegram Bot API.

It is also possible to set up a webhook for the Bot to receive updates from Telegram whenever there's new activity or interaction.

To execute HTTPS calls and automatically dispatch messages to predefined recipients via the Telegram Bot API, it is necessary to obtain the chat IDs of the recipients intended to receive messages by capturing them in the Bot logic through user interactions.

Therefore, it is possible to implement HTTPS call to make a POST request to the Telegram Bot API, including essential parameters such as the API token, chat ID, and message text in the request body.

It is also possible to trigger message sending activating the HTTPS call from the Bot logic whenever necessary to send messages to the predefined recipients, based on specific conditions or events.

 doi:10.20944/preprints202403.1763.v1



The Arduino RP2040 Connect communicates with Telegram servers and dispatches messages using HTTPS requests or dedicated libraries. One prominent library for this purpose is the “Universal Telegram Bot” library.

In the present project, the alert Telegram message has been tested both directly (via HTTPS call to the Telegram API) and utilizing the "Universal Telegram Bot" library, as well as employing the IFTTT webhook and relevant applet.

Between using the library and directly implementing the HTTPS call to Telegram's API, the author has opted for the latter approach. Although libraries offer convenience, they are susceptible to updates, necessitating constant maintenance by their own author to prevent significant malfunctions. Conversely, by developing proprietary code, these issues are circumvented, affording the developer complete control over their project.

Anyway, the solution adopted in the prototype of the “no pAIn” device, owing to its speed, involves sending Telegram messages utilizing an IFTTT webhook activated by the programmed reaction in ThingSpeak upon receiving the alert code.

### 3.5.5. Discord Messages

The platform Discord allows to create, manage, and participate in communication channels organized in well-defined structures [55].

In fact, Discord has a very simple structure that encloses the channels within the program's secondary servers. Users can independently manage one or more servers and, for each server, one or more channels, allowing or not allowing other users to take part.

The primary tools provided to programmers include Bots and the webhook service, to execute an action in response to a trigger event.

To seamlessly incorporate Discord notifications into the "no pAIn" project, a webhook service activated by the μC of the "no pAIn" device is utilized through an HTTPS POST call, initiating the notification dispatch process.

To achieve this, the initial step involves creating a server within a Discord account. Subsequently, within the server, a text-type channel is established, inviting the intended recipients of alert notifications to subscribe. Finally, within the created channel, the "Create Webhook" item is generated via the "Integrations" option.

Upon creation of the webhook, Discord provides a webhook URL. This URL serves as the address through which users can interact with the service.

At this juncture, the remaining task involves configuring the code responsible for making the HTTPS POST call to the created webhook service.

```
#include <ArduinoHttpClient.h>
const String discord_tts = "true";
const char server[] = "discordapp.com";
const int port = 443;
const String discord_webhook = "WEBHOOK_URL";
int status = WL_IDLE_STATUS;
WiFiSSLClient client;
HttpClient http_client = HttpClient(client, server, port);
```

The content to be transmitted is delineated within the parameters of the POST call. The primary code snippet for dispatching messages is as follows:

```
void discord_send(String content) {
  http_client.post(discord_webhook, "application/json", "{\"content\":\"" + content + "\", \"tts\":" + discord_tts + "}");
}
```

 doi:10.20944/preprints202403.1763.v1



3.5.6. Pushsafer Notifications

Pushsafer [53] is a service that allows a mC to send alerts and notifications to devices where the Pushsafer app is installed, through an HTTPS API call.

It enables users to customize notifications with various parameters such as message content, title, icon, sound, and vibration. Additionally, Pushsafer supports various platforms including Android, iOS, and desktop browsers, making it versatile for different device types.

Users can send notifications either through direct HTTPS calls or by utilizing their provided libraries and plugins for popular programming languages and platforms.

Overall, Pushsafer offers a flexible and customizable solution for sending notifications across different devices and platforms.

A notable feature of Pushsafer is its inherent geolocation service.

The Pushsafer service is accessed via its API using an HTTP call to the following URL: https://www.pushsafer.com/api?k=XXXXXXXXX&d=a&t=Alert! Serious danger!&m=Message from nopAIn&s=8&v=1&i=1&c=#FFCCCC&u=https://www.pushsafer.com&ut=Open Pushsafer.com

The method and the corresponding Arduino code remain consistent with any other HTTPS call.

The summary of this section (refer to Figure 5) outlines the implementation of Telegram alert message transmission through both the μC and the IFTTT applet. Email services have been utilized via the μC (employing “Mailgun” and SMTP2GO), the IFTTT applet, and ThingSpeak (in two ways: utilizing ThingSpeak's alert services and through a call from ThingSpeak to IFTTT applets). Discord alert messages have been transmitted via the μC. Additionally, Pushsafer notifications have been sent through the μC, although implementing the HTTPS call via ThingSpeak with a few lines of MATLAB code in the “MATLAB Analysis” app is straightforward.

The decision for the prototype of the "no pAIn" device is to adopt a cloud-based approach for transmitting alert notifications, as depicted in Figure 7.

It is important to note that in the prototype implementation of the "no pAIn" device, ThingSpeak serves as a cloud resource not only for transmitting alert notifications but also for storing and retrieving geolocation data. Therefore, when the “no pAIn” device switches into the "Alert" state, the mC transmits the alert code to the ThingSpeak alert channel. Upon receiving the alert code, the chain of events, namely "reaction – MATLAB Analysis," is initiated in the cloud. Consequently, two tasks are executed in the cloud: sending an email through ThingSpeak alert services and triggering a call to the IFTTT service. Subsequently, the IFTTT applets dispatch an email (via the Gmail provider) and a Telegram message.

The advantage of this scenario is evident: the μC only needs to transmit the alert code to a ThingSpeak alert channel. This enhances the device's reliability, reduces power consumption, minimizes device resource requirements, and improves operational speed. Moreover, thanks to the cloud solution, the "no pAIn" device does not necessitate a real-time operating system. Instead, it delegates all notification-sending operations to the cloud, merely transmitting an alert code and promptly reverting to a “Sentinel” mode, to detect any further critical or dangerous situations. Utilizing real-time operating systems would exacerbate the device's resource demands in terms of memory and complicate its management. By simplifying this aspect, the device has significantly reduced requirements and operates reliably.

Certainly, if desired, alongside the cloud approach, one or more of the other methods of directly transmitting alert notifications by the μC, which have been previously described and successfully tested, can be implemented. This redundancy would undoubtedly enhance the device's reliability, although this advantage must be carefully weighed against the drawback of increased power consumption and greater hardware resource requirements.

## 4. Working of the System

The prototype of the "no pAIn" device underwent testing by connecting it to the Arduino IDE serial monitor for debug purposes.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 March 2024 doi:10.20944/preprints202403.1763.v1



During the testing phase, screens of both the PC where the device was connected (with the Arduino IDE serial monitor activated) and a smartphone receiving alert notifications were simultaneously recorded.

A test scenario was initiated (refer to figure 8) wherein a woman initially engaged in a calm conversation with her interlocutor, during which the "no pAIn" device operated in the "Listen&Recognize" mode. The green LED onboard illuminates during this interaction, indicating the activation of the “Trigger” state of the device.

As the conversation progressed, the speech became more agitated and becomes suspicious, prompting the woman to request assistance (the keyword “help” has been detected) and causing the "no pAIn" device to transition to the "Sentinel" mode in which it immediately establishes and Internet connection and acquires GPS coordinates for alert transmission preparation. The red LED onboard illuminates always for debugging purposes, indicating the device's transition to the "Sentinel" mode (protection activated).

Subsequently, the speech escalates to a level indicating serious danger, the device switches from the "Sentinel" mode to the "Transmission" mode, initiating the transmission of relevant notifications, containing geolocation data and seeking assistance.

The prototype was developed using the Italian language; therefore, all messages in figure 8 are in Italian. However, the figure caption provide English translations of the text displayed in the images captured from the recorded screen video.


**Figure 8.** Real functioning of the “no pAIn” device. Please, translate the speech as follows:.

Ciao, come stai? >>> Hi, how are you?
ma no, no, cosa fai? >>> but no, no, what are you doing?
Aiuto >>> Help
Lasciami stare, lasciami andare >>> Leave me alone, let me go
Vuoi farmi male? >>> Do you want to hurt me?
Aiutatemi >>> Help me
Chiedo aiuto >>> I ask for help
Chiamate la polizia >>> Call the police
Chiamate i carabinieri >>> Call the police
Traslate the “no pAIn” actions as follows:
no pAIn attivo >>> no pAIn activated
Tutto ok >>> that’s ok
Situazione sospetta >>> Suspicious situation
Coordinate GPS acquisite >>> location tracked
Situazione critica. Protezione attivata >>> Critical situation. Protection activated

 doi:10.20944/preprints202403.1763.v1



Pericolo imminente. Alert trasmesso >>> Sudden danger. Alert sent
Richiesta di aiuto inviata! >>> Request for help sent.

Figure 9 displays the sequence of alerts received on a smartphone: a Telegram message via IFTTT, an email by Gmail via IFTTT, and an email via ThingSpeak alert service, all containing requests for assistance along with location coordinates.

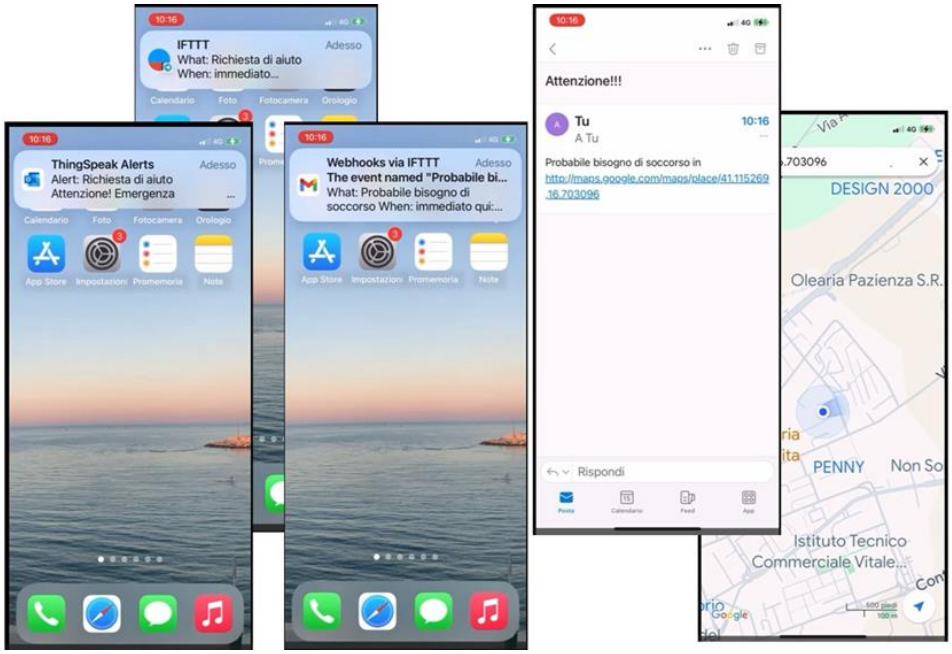


**Figure 9.** Flowchart illustrating the process of sending notifications via cloud service using ThingSpeak.

Upon observing the smartphone screens depicted in both Figure 8 and Figure 9, it becomes apparent that within a minute's time (or seconds in reality, as evidenced by the consistent time reading of 10:16 from the onset of speech to the receipt of notifications), all expected notifications promptly arrive, accompanied by the victim's location.

Speech recognition operates seamlessly in real-time, without any noticeable delays. Notifications are promptly delivered to the smartphone in real-time as soon as the "no pAIn" device transitions from the “Sentinel” to “Transmission” mode.

Within mere fractions of seconds from the request for help, the victim's location is pinpointed, and potential helpers are immediately alerted. Consequently, the device demonstrates exceptional efficiency in preventing gender-based crimes.

## 5. Conclusions

This paper outlines a new device powered by artificial intelligence, designed to prevent gender-based violence.

The device currently stands out as unique and original due to its operational characteristics, leveraging cutting-edge Artificial Intelligence, Internet, geolocation, and Embedded Systems project technologies.

The outcomes derived from the prototype implementation and testing reveal that despite the device boasting intricate and technologically advanced functionalities, its cost appears to be remarkably low (few tens of dollars). Moreover, owing to the utilization of highly advanced technologies in its construction, the degree of automation is exceptionally high, eliminating the necessity for users to possess any specialized skills; they simply need to activate the device, and all operations proceed seamlessly on their own.

 doi:10.20944/preprints202403.1763.v1



Therefore, the most significant attribute of the “no pAIn” device lies in its ability to be marketed at a reasonable low cost, rendering it accessible to a wide demographic. Additionally, it is suitable for both indoor and outdoor applications. Its discreet form factor enables it to be easily concealed, ensuring that potential attackers remain unaware of their victim's protection. This feature facilitates the early identification of domestic and gender violence risks, thereby enabling the prevention of such tragic events with a high degree of probability. Furthermore, the device operates in fully automatic mode, requiring no user intervention beyond the initial switch-on.

The selected technical solutions, along with additional strategies aimed at enhancing the device and optimizing its performance, have been extensively discussed.

The present prototype, utilizing the Arduino Nano RP2040 Connect board equipped with a built-in omnidirectional microphone and Wi-Fi connectivity, demonstrates highly promising results regarding the reliability of speech classification and the prompt transmission of alert notifications.

The prototype is a minimal version that requires connection to a hotspot and interaction with a smartphone to function, as explained in the article. However, all the required implementation solutions have also been successfully tested so that the device can be produced in a fully optional version, equipped with a physical GPS module and its own SIM card, thereby becoming completely autonomous in performing all its functions. In this case, there is an increase in cost and size, although it remains a miniaturized device.

A further design development pertains to some third-party services used by the hardware device, which could be specifically developed.

There is also the possibility of transforming the entire project into a smartphone app, although the design and functional aspects of such an app are still under evaluation.

Indeed, in principle, the functionality of the device could also be replicated through a smartphone application. However, due to the prevalent nature of gender-based violence where attackers often deactivate or discard the victim's smartphone, such an approach would pose safety and reliability concerns. Conversely, the standalone version of "no pAIn", particularly when equipped with its own SIM card and discreetly concealed, offers enhanced reliability as the attacker is less likely to detect its presence and operation.

Another potential advancement, perhaps more pertinent than a smartphone app, could be a smartwatch app. Nevertheless, like smartphone apps, this option presents comparable vulnerabilities; the attacker might seize the victim's smartwatch. Even if such an eventuality is avoided, the associated costs would be considerably higher, given that smartwatches are substantially more expensive than the standalone "no pAIn" device. This elevated cost could impede widespread adoption, which is a desirable outcome.

The commercial prospects are also noteworthy and may involve not only the marketing of the device but, also, certain services. This includes software technologies underlying its operation (which could be specifically developed and provided by a service provider) and the alert and rescue organization service. For instance, a dedicated company could handle the marketing or rental of the device, receive, and manage the requests for help it sends 24 hours a day, and consequently coordinate requests for assistance (such as contacting the police). There is also consideration for a type of "insurance" against attacks based on the use of the device and related services.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. https://fra.europa.eu/en/publication/2014/violence-against-women-eu-wide-survey-main-results-report
2. https://www.istat.it/it/violenza-sulle-donne/il-fenomeno/omicidi-di-donne
3. https://www.rainews.it/articoli/2022/11/istat-violenza-donne-femminicidi-famiglia-42c5e092-8d4e-44a7-85a2-b0f52801ff98.html
4. https://www.internazionale.it/bloc-notes/giulia-testa/2019/11/25/dati-grafici-violenza-genere

 doi:10.20944/preprints202403.1763.v1




5. https://www.ansa.it/sito/notizie/mondo/europa/2023/11/25/record-femminicidi-nel-mondo-nel-2022-quasi-89.000-donne-uccise_b3349caf-4aed-40d8-9438-01ae34fe53a0.html
6. https://tg24.sky.it/stories/cronaca/femminicidi-italia/index.html
7. https://www.elle.com/it/magazine/women-in-society/a46517044/dati-femminicidi-2023-ufficiali/
8. https://www.onuitalia.com/2023/11/24/donne-14/
9. https://www.tag24.it/946219-femminicidi-dati-istat-2023-120-donne-uccise-meta-dai-partner/
10. https://www.today.it/attualita/femminicidi-dati-italia-quanti-2023.html
11. https://www.money.it/quanti-femminicidi-italia-2023-numeri-europa-mondo
12. Chen, Y., & Zhang, J. (2019). Intelligent audio surveillance for public safety: A review. Journal of Ambient Intelligence and Humanized Computing, 10(12), 4985-5001.
13. Smith, A., Jones, B., & Wang, C. (2018). Wearable devices for personal safety: A comprehensive review. IEEE Transactions on Mobile Computing, 17(12), 2891-2909.
14. Gupta, S., Verma, A., & Sharma, S. (2017). Voice-based emotion recognition using deep learning: A review and future directions. Journal of King Saud University - Computer and Information Sciences.
15. Kim, D., & Yoon, J. (2019). IoT-based personal safety systems: A survey. Journal of Network and Computer Applications, 141, 92-104.
16. Wang, L., Liu, X., & Jin, H. (2020). Cloud-based emergency response systems: A review. Journal of Cloud Computing: Advances, Systems and Applications, 9(1), 1-16.
17. Rahman, M. A., Hossain, M. S., & Alhamid, M. F. (2018). A review on wireless sensor networks-based health monitoring systems. Journal of King Saud University - Computer and Information Sciences.
18. Li, W., & Li, Y. (2019). Machine learning applications in public safety and emergency management: A survey. IEEE Access, 7, 46560-46573.
19. Albuquerque, P., Rocha, Á., & Bernardino, J. (2017). Mobile-based solutions for personal safety: A survey. Journal of Ambient Intelligence and Humanized Computing, 8(4), 497-513.
20. Bakshi, S., Mittal, N., & Jindal, P. (2016). A survey on voice emotion recognition: Features and classification methods. Journal of Ambient Intelligence and Humanized Computing, 7(4), 519-540.
21. Liu, Y., Zhang, W., & Wang, Z. (2017). An overview of wearable technology in personal safety applications. Journal of Sensors, 2017
22. Chen, Y., & Zhang, J. (2019). Intelligent audio surveillance for public safety: A review. Journal of Ambient Intelligence and Humanized Computing, 10(12), 4985-5001.
23. Smith, A., Jones, B., & Wang, C. (2018). Wearable devices for personal safety: A comprehensive review. IEEE Transactions on Mobile Computing, 17(12), 2891-2909.
24. Gupta, S., Verma, A., & Sharma, S. (2017). Voice-based emotion recognition using deep learning: A review and future directions. Journal of King Saud University - Computer and Information Sciences.
25. Kim, D., & Yoon, J. (2019). IoT-based personal safety systems: A survey. Journal of Network and Computer Applications, 141, 92-104.
26. Wang, L., Liu, X., & Jin, H. (2020). Cloud-based emergency response systems: A review. Journal of Cloud Computing: Advances, Systems and Applications, 9(1), 1-16.
27. Rahman, M. A., Hossain, M. S., & Alhamid, M. F. (2018). A review on wireless sensor networks-based health monitoring systems. Journal of King Saud University - Computer and Information Sciences.
28. Li, W., & Li, Y. (2019). Machine learning applications in public safety and emergency management: A survey. IEEE Access, 7, 46560-46573.
29. Albuquerque, P., Rocha, Á., & Bernardino, J. (2017). Mobile-based solutions for personal safety: A survey. Journal of Ambient Intelligence and Humanized Computing, 8(4), 497-513.
30. Bakshi, S., Mittal, N., & Jindal, P. (2016). A survey on voice emotion recognition: Features and classification methods. Journal of Ambient Intelligence and Humanized Computing, 7(4), 519-540.
31. Liu, Y., Zhang, W., & Wang, Z. (2017). An overview of wearable technology in personal safety applications. Journal of Sensors, 2017
32. Nano RP2040 Connect - Arduino Docs. https://docs.arduino.cc/hardware/nano-rp2040-connect/
33. Nano RP2040 Connect Cheat Sheet - Arduino Docs. https://docs.arduino.cc/tutorials/nano-rp2040-connect/rp2040-01-technical-reference/
34. Arduino Nano RP2040 Connect — Arduino Official Store. https://store.arduino.cc/products/arduino-nano-rp2040-connect
35. Tensorflow library: https://www.tensorflow.org

 doi:10.20944/preprints202403.1763.v1




36. MATLAB&Tools environment: www.mathworks.com
37. www.edgeimpulse.com
38. https://store.arduino.cc/products/speech-recognition-engine?queryID=undefined
39. https://tool.cyberon.com.tw/ArduinoDSpotterAuth/CTMain.php
40. https://www.adafruit.com/product/4415
41. https://www.espressif.com/sites/default/files/documentation/geolocating_with_esp8266_en.pdf
42. https://www.wispot.it/blog/tecnologia-wifi-localizzazione-dispositivi/
43. Spachos, P. and Plataniotis, K., "BLE Beacons in the Smart City: Applications, Challenges, and Research Opportunities," in IEEE Internet of Things Magazine, vol. 3, no. 1, pp. 14-18, March 2020, doi: 10.1109/IOTM.0001.1900073. online available on: https://arxiv.org/abs/2102.08751
44. https://developers.google.com/maps/documentation/geolocation/overview?hl=it
45. https://ipstack.com
46. https://www.geolocation.com
47. https://ipinfo.io
48. https://www.whatismypublicip.com/
49. https://owntracks.org
50. https://thingspeak.com
51. https://www.mailgun.com
52. https://www.smtp2go.com
53. https://www.pushsafer.com
54. https://web.telegram.org
55. https://discord.com
56. https://ifttt.com
57. https://planbproject.it/blog/webhook-perche-usarlo/.
58. https://www.mdirector.com/it/blog/webhook/..