**SecureGuard-Pro: A Web-Enabled ESP32-Based System for Integrated Security and Environmental Monitoring**

|  |  |
| --- | --- |
| **Author:** | Olara Emmanuel |
| **Email:** | [eolara68@gmail.com](mailto:eolara68@gmail.com) |
| **Student Number:** | 2401003714 |
| **Registration Number:** | 24/U/3714/GCS/PS |
| **Affiliation:** | Department of Computer Science, Faculty of Science |
|  | Gulu University, Gulu, Uganda |
| **Course:** | Embedded Systems (GAT1301) |

**Abstract**

The rapid advancement of the Internet of Things (IoT) presents unprecedented opportunities for creating intelligent, connected, and autonomous systems. This article details the design, implementation, and application of "SecureGuard-Pro," a multifunctional embedded system built on the ESP32 platform. The system addresses the growing need for affordable, integrated, and customizable security and environmental monitoring solutions. By integrating a sound sensor for intrusion detection, a temperature sensor, and a Light Dependent Resistor (LDR) for environmental analysis, SecureGuard-Pro provides a holistic view of a given space. It features a robust alert mechanism using a local buzzer, visual feedback via an RGB LED, and real-time data display on a 7-segment module. The system's core innovation lies in its self-hosted, asynchronous web server, which provides a password-protected, dynamic web dashboard for remote monitoring and control. This article explores the system's architecture, dissects the key problem-solving techniques employed during its development, and evaluates its significant potential for industrial applications, particularly in smart automation and low-cost security sectors.

1. **Introduction**

### 1.1. Background and Motivation

We are currently in the midst of the Fourth Industrial Revolution, or **Industry 4.0**, an era defined by the profound integration of the physical, digital, and biological worlds. A cornerstone of this transformation is the **Internet of Things (IoT)**, which facilitates the creation of vast **cyber-physical systems**. These systems bridge the gap between machinery and digital networks by embedding sensors, processors, and communication hardware into everyday objects. Consequently, traditional embedded systems—once isolated, single-function units like those in a simple appliance—have evolved. They are no longer standalone devices but are now intelligent, networked nodes capable of collecting and exchanging data, enabling real-time monitoring, remote control, and data-driven decision-making on an unprecedented scale.

This paradigm shift has fueled an explosive demand for smart devices across a spectrum of applications. In the consumer space, **smart home automation** has become mainstream, with devices for intelligent climate control, security, and lighting gaining popularity. In the industrial sector, the **Industrial Internet of Things (IIoT)** is revolutionizing manufacturing and logistics through predictive maintenance sensors, automated supply chains, and enhanced operational efficiency. This technological wave holds particular promise for developing regions. In the context of Gulu, Uganda, for example, IoT principles can be applied to create **smart farming** solutions for crop monitoring and resource management, directly addressing local agricultural challenges.

However, the path to adopting this technology is fraught with significant obstacles, primarily due to the nature of existing commercial solutions. Many products on the market are built within **proprietary ecosystems** (e.g., Apple HomeKit, Google Home, Amazon Alexa). This practice leads to **vendor lock-in**, where consumers are confined to a single brand's compatible devices and software, stifling interoperability and choice. Furthermore, the cost of these systems is often prohibitive. The financial barrier includes not only the high initial price of the hardware but also mandatory **recurring subscription fees** for cloud storage and premium features, making them inaccessible for individual users, students, and small-to-medium-sized enterprises (SMEs).

Perhaps the most critical limitation is the inherent **lack of flexibility and customizability**. Commercial systems are closed boxes, offering little to no room for modification. A user cannot easily integrate a third-party sensor or adapt the software to a unique, specific need—a critical requirement for local innovation. This rigidity creates a significant barrier to entry, preventing local problem-solvers from building solutions tailored to their unique environmental and economic contexts. It is precisely this gap—the need for an affordable, open, and adaptable platform—that motivates the development of the **SecureGuard-Pro** system.

**1.2. Problem Statement**

The primary problem addressed by this project is the lack of accessible, low-cost, and integrated systems for security and environmental monitoring. Homeowners, small business operators, and small-scale farmers often face a choice between expensive, inflexible commercial security systems or a collection of disparate, non-communicating gadgets. There is a clear need for a unified platform that can:

* Provide reliable intrusion detection.
* Monitor key environmental variables like temperature and light.
* Offer both local and remote alerts and controls.
* Be easily customizable and scalable without vendor lock-in.

The "SecureGuard-Pro" system is engineered as a comprehensive solution to this problem, leveraging open-source hardware and software to deliver a powerful, cost-effective, and user-centric platform.

**2. System Design and Methodology**

The architecture of SecureGuard-Pro is a carefully integrated assembly of hardware and software components chosen for their performance, cost-effectiveness, and compatibility.

**2.1. Hardware Architecture**

* **Microcontroller (MCU): ESP32** - The ESP32 was selected as the system's brain due to its powerful dual-core processor, extensive GPIO pins, and, most importantly, its built-in Wi-Fi and Bluetooth capabilities. This eliminates the need for external communication modules and provides the foundation for the web server.
* **Sensors (Input Layer):**
  + **Sound Sensor:** A digital microphone module acts as the primary trigger for the security alarm. It provides a simple HIGH/LOW output, making it easy to integrate for detecting noise events that cross a preset threshold.
  + **Temperature Sensor (LM36DZ):** This analog sensor was chosen for its precision and linear output. It provides reliable temperature readings, which are converted using the formula: Temperature (°C) =ADCreading​×0.322265625. This data is crucial for environmental monitoring applications.
  + **Light Dependent Resistor (LDR):** An LDR provides a simple, effective method for measuring ambient light levels. This data is used to create an adaptive nightlight with the RGB LED and is also displayed on the user dashboard.
* **Actuators (Output Layer):**
  + **Buzzer:** An active buzzer provides an immediate, high-decibel audible alarm upon intrusion detection.
  + **RGB LED:** Serves as a multi-purpose visual indicator. Its brightness is dynamically controlled by the LDR reading, making it a functional nightlight that adapts to its surroundings.
  + **Dual-Digit 7-Segment Display (5161AS):** This provides at-a-glance information locally, displaying temperature and light levels without requiring the user to connect to the web interface.

**2.2. Software Architecture**

The system is programmed using the Arduino framework, known for its simplicity and extensive library support.

* **Core Libraries:**
  + WiFi.h: Manages the creation of the ESP32's Wi-Fi Access Point (softAP), making the system a self-contained network hub.
  + ESPAsyncWebServer.h: This library is critical for creating a non-blocking web server. It can handle multiple simultaneous HTTP requests without halting the main program loop, ensuring that sensor monitoring remains responsive.
  + TimeLib.h: Used for timekeeping functions, essential for logging events and scheduling tasks.
* **Program Flow:**
  1. **setup ():** This function runs once on startup. It initializes serial communication, configures all GPIO pins as inputs or outputs, establishes the Wi-Fi access point, and sets up the web server routes (e.g., /login, /dashboard, /mute).
  2. **loop():** This is the heart of the system, running continuously. It performs three main tasks: reading sensor data, processing this data to make decisions (e.g., trigger the alarm), and updating all outputs (buzzer, LED, and 7-segment display).
  3. **processor() Function:** This template processor is a key software pattern. When a web page is requested, this function intercepts it and replaces placeholder strings (e.g., "%TEMPC%", "%BUZZER\_STATE%") with real-time data before sending the HTML to the user's browser. This allows for dynamic, data-rich web pages without complex server-side scripting.

**3. Challenges and Problem-Solving**

Developing an integrated system like SecureGuard-Pro presented several technical challenges. The following section details these hurdles and the solutions implemented.

**3.1. Challenge: Real-time Responsiveness and Concurrency**

* **Problem:** A naive implementation using delay() functions to time events would block the entire processor. This would make the web server unresponsive and could lead to missed sensor readings, rendering the security system unreliable.
* **Solution:** A non-blocking architecture was adopted. Instead of delay(), the system uses the millis() function to track time intervals for updating the display and other periodic tasks. The most critical decision was using the **ESPAsyncWebServer library**, which operates on an event-driven model. It handles incoming web requests in the background without halting the primary loop(), ensuring the system can serve web pages while simultaneously monitoring sensors in real time.

**3.2. Challenge: Efficient Hardware Data Visualization**

* **Problem:** Driving a dual-digit 7-segment display requires rapidly alternating which digit is lit—a technique called **multiplexing**. If not implemented efficiently, this constant switching can consume significant CPU resources and cause flickering.
* **Solution:** The logic was encapsulated in displayDigit() and displayNumber() functions. displayNumber() breaks the number into two digits and calls displayDigit() for each one in quick succession. A minimal delay(5) within the display cycle ensures persistence of vision for the human eye while being short enough not to disrupt other processes. The main loop() calls the display update function periodically rather than on every single pass, striking a balance between a stable display and system performance.

**3.3. Challenge: Sensor Accuracy and Data Interpretation**

* **Problem:** Raw analog values from sensors are meaningless to an end-user. The LM36DZ sensor's output voltage and the LDR's resistance change needed to be translated into intuitive, human-readable units.
* **Solution:**
  + **Temperature:** A precise conversion factor (0.322265625) was calculated based on the LM36DZ datasheet and the ESP32's 3.3V ADC reference. This ensures the conversion from the raw 12-bit ADC value (0-4095) to degrees Celsius is accurate.
  + **Light:** The raw LDR value, which can be unintuitive, is mapped to a simple 0-100 percentage scale using the map() function. This provides a universally understandable measure of brightness on the web dashboard and for the 7-segment display.

**3.4. Challenge: Enhancing User Experience (UX) and Accessibility**

* **Problem:** A purely functional interface is not enough. To be truly useful, the system must be accessible and easy to use for non-technical individuals.
* **Solution:** The system was built with UX at its core. The web interface is not static; it is **customizable**. The implementation of selectable themes (changing PRIMARY\_COLOR, SECONDARY\_COLOR, etc.) and a framework for multiple languages (EN\_SELECTED, ES\_SELECTED, etc.) makes the system far more approachable. The logical separation of pages (Dashboard, Settings, Help) further improves usability.

**4. Industrial Relevance and Technological Advancement**

SecureGuard-Pro is not merely an academic exercise; it is a prototype with significant relevance for various industries and a direct solution to existing technological gaps.

* **Smart Homes and Building Automation:** The system serves as a low-cost, open-source alternative to expensive commercial smart home hubs from companies like Google, Amazon, or Apple. Its integrated nature allows it to perform roles of a security system, a smart thermostat, and an automated lighting controller simultaneously. Homeowners can deploy it for security and also leverage its environmental data to optimize energy consumption.
* **Small-Scale Smart Agriculture:** In contexts like Gulu, Uganda, this technology can be transformative for local farmers. The temperature and light sensors are ideal for monitoring conditions in a small greenhouse or a storage facility for perishable goods. The sound sensor could be repurposed to detect the operation of irrigation pumps or even pests. Its low cost makes advanced monitoring accessible.
* **Education and Rapid Prototyping:** As an open-source project, its code and design serve as an invaluable educational tool for students of computer science and engineering. It is a practical case study in embedded systems, IoT networking, and web development. Industries can also use this platform as a base for rapidly prototyping more complex and specialized IoT products without investing heavily in initial R&D.

**4.1. Solution to Industry-Wide Challenges**

* **Overcoming the Cost Barrier:** By using ubiquitous, low-cost components like the ESP32, the system drastically reduces the cost of deploying a smart security and monitoring solution, democratizing the technology.
* **Eliminating Vendor Lock-in:** The open-source nature of the software and hardware design means users are not tied to a single company's ecosystem. They can modify, expand, and repair the system freely, a significant advantage over closed, proprietary products.
* **Bridging the Integration Gap:** Many IoT devices are single-purpose. SecureGuard-Pro demonstrates how to build a single, cohesive unit that integrates security, environmental sensing, and remote control, providing more value and a more streamlined user experience than a collection of separate devices.

**5. Conclusion**

The "SecureGuard-Pro" system successfully demonstrates the immense potential of modern microcontrollers to create sophisticated, highly functional, and user-centric IoT solutions. By addressing key challenges in real-time processing, hardware control, and user interface design, the project culminates in a reliable and robust prototype. It effectively solves the real-world problem of needing affordable and integrated monitoring systems. More than a proof-of-concept, SecureGuard-Pro stands as a flexible, scalable, and powerful platform with direct applications in home automation, smart agriculture, and education, showcasing a path forward for accessible technological advancement.