SecureScape – Smart Deployable Security System

Jaxon Topel, Dylan Myers, Colin Kirby, Phillip Murano

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — SecureScape is a smart, deployable security system tailored for remote or off-grid environments without internet access. It integrates infrared motion detection, image capture, and onboard person detection to enable autonomous threat identification. All data is transmitted locally to a central gadget and mobile application, allowing users to receive real-time alerts and manage system controls without external connectivity. By combining efficient hardware design with edge AI, SecureScape delivers a practical, low-power solution for dependable security in disconnected environments.**

***Index Terms* — Edge computing, embedded systems image processing, passive infrared sensors, person detection, mobile application, wireless communication.**

I. Introduction

SecureScape, is a smart, deployable security system designed to address the growing need for increased security in rural areas with limited or no access to the internet. To achieve the goal of building a comprehensive security solution, we have developed two primary hardware components and divided the software into three main segments: machine learning/software architecture, embedded code, and front-end development.

Our demonstration will feature one node, but it can be scalable up to three nodes—our first hardware component—each consisting of one ESP32-CAM microcontroller and a passive IR sensor. The passive IR sensor continuously monitors the environment for motion. Upon detecting motion, the corresponding ESP32 captures an image, and our image processing algorithm runs person detection on the recorded image. If a person is detected, the node triggers the alarm and sends a message over a local network, hosted by our router, to the mobile app and to the Gadget—our second hardware component.

The Gadget is a local interface device used to interact with the system. It receives alerts for potential threats and provides the user with control over various system features. Image processing occurs on the ESP32-CAM microcontroller embedded in the Node, which runs an object detection algorithm to determine whether a person is present in the captured image. If a threat is identified, the system activates an onboard alarm located within each node and sends notifications to both the iOS application and the Gadget.

In addition to threat detection and alerts, we are developing interactive features accessible via the Gadget and the mobile app. These include controls to enable or disable the alarm, manage lighting, and—exclusively through the iOS app—request images from the nodes.

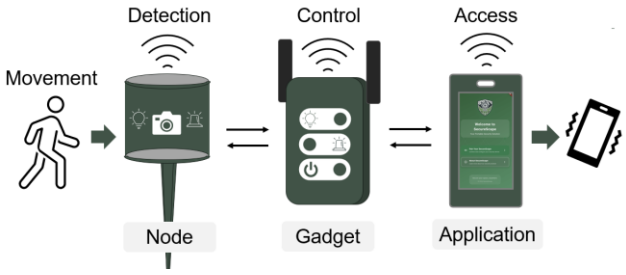

Fig. 1. Graphical Explanation of the Threat Detection Process: The words above encapsulate the process in a single term, while the words below associate the descriptors with the corresponding image.

II. System Components

A. Microcontroller

The main microcontroller used is the ESP32-S3-WROOM-1. This module was selected for its extensive peripheral interfaces and flexibility for a wide array of applications. It offers 2.4 GHz Wi-Fi (802.11 b/g/n) and Bluetooth 5 (LE) connectivity. The S3 also has native USB support as well as SPI, I2C, UART, ADC, DAC, and USB OTG. With embedded flash memory and dual-core Xtensa® 32-bit LX7 microprocessor with AI acceleration support, this module is ideal for the application. Finally the low power modes make it useful for the mobility of the device.

B. Camera

The camera used is the ESP32-CAM. It too supports 2.4 GHz Wi-Fi and Bluetooth connectivity. As well as the same interfaces as the regular ESP32, i.e. UART, SPI, and I2C. the camera embedded is a 2 Megapixel 65° FOV picture and video camera. With built in deep sleep modes it uses very little power while on standby.

C. Wi-Fi Router

The router utilized is the GL.iNet GL-AR300M16-Ext. This module is a compact and portable wireless travel router for safe internet browsing and flexible networking.

We found GL.iNet to be the only company offering affordable low-power routers. This module is powered by a Qualcomm QCA9531 SoC and 16MB of flash storage and is shipped with OpenWrt/LEDE-based firmware offering complex functionality such as VPN support (OpenVPN and WireGuard) and firewall handling. The router supports 2.4GHz Wi-Fi (up to 300Mbps) and comes with two Ethernet ports for WAN/LAN connectivity. It uses 2 amps of power and is the most power intensive component of our system.

D. External Memory

MicroSD cards are tiny, portable flash memory cards widely used to hold data in portable electronic devices. MicroSD cards are just 15mm x 11mm—microSD cards offer extremely high storage capacities of up to 1 TB. Their plug-and-play functionality, low power requirements, and high reliability make microSD cards a great and indispensable component to hold data in our system.

E. Passive Infrared

The AM312 (also called HC-SR312) is a low power small size passive infra-red (PIR) motion sensor module, which detects human body motion as heat through infrared radiation. The module can be powered between 2.7V to 12V and draws very low current, making it ideal for battery operated and low power applications. It can sense movement 3 to 5 meters away with a wide field of view (~100°) and produces a simple digital high/low-level output signal when motion is detected.

F. Alarm Hardware

Named after the piezoelectric materials that comprise them, piezo buzzers provide auditory cues to the end user. Their materials offer a unique ability to convert energy between mechanical and electrical forces. The piezoceramic disk converts electric current into sound producing mechanical movement through vibration. The SFM-27-W is rated for DC 12 volts at 15mA, with the capability to operate from 3 to 24 volts. At the operating voltages in SecureScape, the buzzer will still produce sound above 85dB.

G. DC-DC Converter

Voltage regulation is consistent for both the Node and Gadget, powered by a dedicated PCB. The PCB features the XL1509-5.0E1 and AMS-1117-3.3 as the primary components for voltage conversion. Starting from a 7-volt input, the voltage is reduced to 5 volts for distribution to devices requiring it. A second converter further steps the voltage down to 3.3 volts for additional distribution.

III. BOARD DESIGN

SecureScape hardware development involved the designing of PCBs so that the functionality and scalability of the system were possible. There were two main versions of PCB created during the project. Both replicated improvements in integration, efficiency, and modularity.

Hardware Version 1 consisted of two 4-layer PCBs with power delivery on board. One of the boards, now referred to as the Gadget PCB, was dedicated entirely to user interaction and system control. It had subsystems for image data reception, alert processing, and user interface functionality. The Node PCB was the second board, intended to handle environmental sensing and image acquisition. This board accommodated the passive IR sensors and was also in charge of communication with the Gadget via the local network. Version 2 had the PCB architecture redesigned into a more modular and efficient three-board system, each with a 2-layer design.

The new design features a dedicated Power PCB that is dedicated solely to power regulation and management; an optimized Gadget PCB that removes noncritical subsystems for simplicity and smaller footprint; and an enhanced Node PCB that also removes unused features to focus solely on image processing and sensor data acquisition. Modular design enhanced the readability of the design, facilitated easier troubleshooting, and allowed more targeted testing and revisions within individual subsystems. The move from Version 1 to Version 2 is a purposeful step toward modularity, reduced system complexity, and improved maintainability, all of which facilitate better deployment and scalability down the line.

IV. PHYSICAL DESIGN & SYSTEM ASSEMBLY

A. Node Design & Assembly

Each SecureScape Node was designed with both durability and practicality in mind, ensuring reliable operation in a range of environmental conditions. The enclosure is built from rigid materials and weather-resistant components, providing protection against dust, light rain, and physical disturbances. A transparent plastic film covers the camera opening, allowing clear image capture while safeguarding the lens from debris and moisture. This window is flush-mounted to the case for a streamlined profile.

The internal layout emphasizes accessibility and maintenance efficiency. Key components—including the ESP32-CAM, PIR sensor, and LED lighting—are securely mounted within the enclosure using snap-fit mounts and

custom slots to reduce vibration and wiring clutter. The LED strip follows a routed channel around the upper perimeter of the case, enhancing visibility while protecting the circuitry. A dedicated corridor allows for clean internal wiring and quick access to all interfaces during debugging or upgrades. The entire node is optimized for both standalone use and multi-node scalability, enabling easy deployment in diverse field environments.

B. Gadget Design & Assembly

The SecureScape Gadget is housed in a compact, two-tier enclosure that balances space efficiency with functional accessibility. Its block-style housing is purpose-built to accommodate the system's core electronics while providing intuitive user interaction. The top lid is detachable, granting direct access to internal components such as the ESP32-S3 Gadget controller, support PCBs, and auxiliary circuitry. This design significantly reduces downtime during troubleshooting and allows seamless hardware upgrades.

The interior is vertically segmented into two primary sections. The lower tier securely houses the power bank and Wi-Fi router, while the upper tier contains the control PCBs responsible for orchestrating system-wide communication and device coordination. A side cutout near the router provides clearance for antenna extension or wired connectivity, ensuring uninterrupted network communication. The outer surface of the Gadget features flush-mounted switches, each labeled for specific control actions such as triggering the alarm or resetting system peripherals. These tactile controls offer a secondary, hands-on alternative to mobile app interaction, making the Gadget a robust command center for both on-site and off-grid deployments.

V. SOFTWARE ARCHITECTURE

A. Embedded Node Logic (ESP32-CAM)

The embedded node logic is written in Arduino C++ and is deployed to each ESP32-CAM device, which serves as the primary sensing unit for SecureScape. The main purpose of this software is to manage sensor monitoring, camera control, Wi-Fi communication, and system event response such as person detection, alarm activation, and status reporting.

Each node continuously monitors its environment using a passive infrared (PIR) sensor while maintaining a low-power state through ESP32's sleep capabilities. When motion is detected, the PIR triggers the onboard camera to capture an image. This image is processed locally using a lightweight AI model powered by Edge Impulse to determine if a person is present. If a person is detected, the node sends an HTTP POST request to the Gadget (central controller) with its image URL and metadata. The node also activates its alarm and lighting systems for visual and audio deterrence.

Additionally, the node runs a local HTTP server, exposing endpoints for external requests such as triggering the alarm, toggling lights, capturing images manually, and heartbeat monitoring. A heartbeat mechanism ensures the gadget can verify each node's operational status every 30 seconds, which improves system resilience and tamper detection. Network reconnection logic is implemented to ensure continuous operation in unreliable Wi-FI environments.

B. Gadget Logic (ESP32-S3)

The Gadget acts as the central controller for SecureScape, managing communication with all active nodes on the network. Its embedded code is designed to process alerts from each node, manage peripheral interactions, and maintain real-time responsiveness to both the hardware switches and mobile app commands.

The gadget runs on an ESP32-S3 and uses its built-in Wi-Fi to maintain local network communication. When a node detects motion and sends an image URL via a POST request, the gadget logs the alert, stores node information, and sets an internal flag to notify the app and user interface. From there, the gadget enables interaction with the connected nodes through various endpoints to trigger alarms or control lights remotely.

In addition to responding to alerts, the gadget polls for switch inputs and includes heartbeat monitoring to ensure all devices stay connected. It also serves as a local server, exposing HTTP endpoints to the app for testing and real-time system feedback. Its architecture is built to scale with additional nodes, ensuring smooth coordination between components within the SecureScape network and seamless integration of new nodes.

C. Communication Protocols

SecureScape uses HTTP over a local Wi-Fi network to facilitate communication between nodes and the gadget. Nodes send alerts and image URLs via POST requests in JSON format, which the gadget parses and logs for user notifications. The gadget also exposes various HTTP GET endpoints that allow commands to be sent back to nodes, such as triggering alarms or activating lights. This lightweight, stateless communication structure allows for scalability and real-time responsiveness without relying on external connectivity. Each endpoint is designed with simplicity in mind, allowing for quick integration,

efficient debugging, and reliable performance even on resource-constrained devices.

D. Mobile Application Integration

The SecureScape mobile application, built using Flutter, acts as the user's bridge to the system's backend. It communicates exclusively with the central Gadget over a local Wi-Fi network via structured HTTP GET and POST requests. Through these endpoints, users can retrieve images, control system functions, and monitor real-time status.

Instead of interfacing with individual nodes, the app centralizes all interactions through the Gadget, enabling unified command routing and system logic. This centralized architecture also simplifies scalability and enhances security by reducing direct exposure of node interfaces.

To support offline reliability, the app features automatic reconnection logic and periodic background syncing of detection alerts and system health indicators. Together, these design choices ensure responsive, resilient performance in local-only environments.

E. Image Processing (Person Detection)

The image processing (IP) algorithm embedded within each SecureScape ESP32-CAM node is written in Arduino C++ and initialized through a header file included at the top of the main.ino source. Once motion is detected, the system allocates memory for the snapshot buffer. Immediately after, it performs error checking to ensure the memory was successfully allocated. If successful, the algorithm moves on to declare all necessary variables for processing. At the heart of this setup is the creation of a signal object, which acts as the input for the machine learning model. This object is configured with a length and a pointer to the raw image data captured by the camera—ensuring that when an image is taken, the pixel data is available for inference.

Once the signal is set up, the ESP32-CAM captures an image and performs another round of error handling to verify a successful capture. The image is then passed into the classifier, which was generated using Edge Impulse and trained on a dataset of locally collected images. The classifier outputs a prediction score, with the key metric being the confidence that a person is present in the frame. If the prediction exceeds the confidence threshold, a detection flag is raised. From there, the algorithm enters a decision block—if a person is detected, the ESP32-CAM takes a series of actions: it notifies the central gadget, turns on the lights, and activates the alarm. This sequence forms the core of SecureScape's reactive detection logic, enabling real-time responses to verified intrusions.

VI. MOBILE APPLICATION DESIGN & FUNCTIONALITY

The SecureScape mobile application provides users with a streamlined and responsive interface for monitoring and controlling their security system in real time. Developed using Flutter, the app offers seamless cross-platform support for both Android and iOS devices. The application enables connectivity to the Gadget, through which users can access captured images, receive person detection alerts, control alarms and lighting, and review system activity logs.

Upon launching the SecureScape mobile application, users are greeted with a clean and inviting Welcome Screen that introduces the purpose of the app while offering a streamlined experience to begin system interaction. This screen features branding elements such as the SecureScape logo, a short tagline, and a prominent call-to-action button labeled "About SecureScape." When tapped, this button transitions the user to the About Screen, which provides a brief but informative overview of the system's goals, functionality and extra details about the team. The About Screen helps orient users by explaining SecureScape's mission: to deliver smart, off-grid security using image processing detections and remote alerts.

From the Welcome Screen, users can proceed to the Connecting Screen, which handles discovery and pairing with the central gadget (ESP32-S3 device). This screen guides users through locating the device on the local network, typically via IP input or auto-discovery. Once connected, users are routed to the Dashboard—the central control hub. This flow from Welcome to About, to Connecting, and into the Dashboard creates a natural user journey that blends onboarding with technical setup. Each step ensures users are ready before engaging with key features like camera previews, detection alerts, test utilities, and system logs.

The Dashboard Screen is the operational center of the app. It offers a real-time view of the system's status, including detection alerts, the latest image captured, and key metadata such as detection time and source. Users can trigger or stop the alarm, toggle lights, or initiate an emergency response—actions that are sent directly to the gadget. Visual feedback such as snackbars and loading indicators confirms each action or flags connection issues.

From the bottom navigation bar, users can access core system features that provide enhanced control and visibility. The Cameras Screen displays all connected nodes in either a grid or list view, depending on user preference, with each camera represented by a preview card. These cards include periodic live image updates,

timestamps, and the option to manually capture a fresh snapshot for immediate review. This feature is especially useful for quickly verifying the current status of monitored areas.

The Logs Screen offers a scrollable, chronological feed of past detections, showcasing associated images, camera identifiers, time of detection, and whether alarms were triggered. Designed for intuitive navigation, it includes refresh functionality and infinite scroll to allow seamless access to both recent and historical records. This enables users to track activity patterns or investigate potential security events over time.

Lastly, the Gadget Utility Screen serves as a dedicated command interface for advanced system control. From this screen, users can remotely trigger core actions such as sounding alarms, toggling lighting, or restarting system components—all via secure, local network commands. Its real-time responsiveness and clean layout make it a vital tool for both routine monitoring and high-stakes moments requiring immediate action.

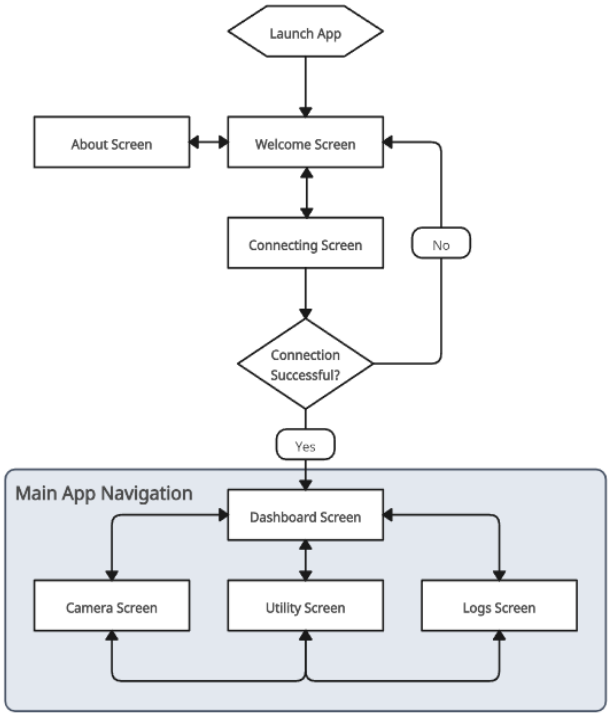

Fig. 2. Graphical Explanation of the SecureScape App Navigation Flow.

VII. TESTING

Testing for the SecureScape system was divided into three key categories, each targeting a critical aspect of overall functionality. These included: (1) Person Detection and Response, (2) Mobile App Controls, and (3) Gadget Command Execution. Each testing phase served a distinct purpose, ensuring that both user-facing features and backend logic performed reliably under real-world conditions. Detailed results and procedures for each test are outlined in the following subsections.

A. Person Detection & Response

The Person Detection & Response test was designed to validate the full end-to-end functionality of the SecureScape system. Using our custom PCBs, the test begins when the Passive Infrared (PIR) sensor detects motion. At that moment, the ESP32-CAM (Node) captures an image and invokes the onboard Edge Impulse person detection algorithm. If a person is detected, the Node automatically triggers the alarm, activates the lights, and sends real-time notifications to both the Gadget and the mobile application. This test was the final step in our validation process, ensuring that all integrated components operate as expected and that the system was fully prepared for the Final Demo. All testing was conducted under normal operating conditions to simulate realistic field deployment and ensure dependable results.

B. Mobile App Control

The Mobile Application Control test was designed to validate the functionality and reliability of the SecureScape mobile interface as a primary user control mechanism. This test was essential to confirm proper interaction between the user, mobile application, and hardware nodes across the local network. The general procedure involved the user launching the SecureScape mobile app, establishing a connection to the SecureScape's Wi-Fi network and then the SecureScape gadget, and initiating various control commands such as activating the alarm or toggling the lighting system. Upon command issuance, the corresponding ESP32 node is expected to receive and execute the appropriate action in real time. This test demonstrates the viability of the mobile application as a fully integrated control hub, offering an alternative to the physical gadget and its commands for managing system responses and thus providing more flexibility for the user's operations.

The Mobile Application Control test was divided into four distinct sub-tests, each targeting a critical control function. The test conditions and expected outcomes are outlined below:

1. Trigger Alarm

In this test, the user opens the SecureScape mobile application and navigates to the utility control screen. Upon tapping the "Trigger Alarm" button, a command is sent over the local Wi-Fi network to the corresponding

node, instructing it to activate the buzzer. A successful test is confirmed if the buzzer sounds immediately, indicating proper signal transmission, quick command execution, and a functional connection between the app and the hardware.

2. Turn Off Alarm

For this test, the user accesses the utility screen and selects the "Turn Off Alarm" control. The mobile app sends a shutdown signal to the node through a GET request, prompting the buzzer to stop. The test is considered successful if the alarm ceases operation without delay, confirming the node's ability to receive and respond to deactivation commands reliably.

3. Turn On Lights

To initiate this test, the user selects the "Turn On Lights" button from the app's utility screen. The command is transmitted to the node, instructing it to activate its connected lighting components. The test is deemed successful if the lights power on within seconds, validating both the responsiveness of the system and the accuracy of command execution via the mobile interface.

4. Turn Off Lights

Finally, the user presses the "Turn Off Lights" button within the utility interface. The command is sent to the node to deactivate its lighting elements, returning the system to a neutral state. If the lights shut off as expected, the test result is confirmed as successful, demonstrating dependable performance in toggling peripheral hardware.

C. *Gadget Control*

The Gadget Control test evaluates the functionality of the ESP32-S3-based central controller, specifically its ability to manage peripheral actions via physical hardware switches. The purpose of this test was to ensure that direct user input through the gadget's built-in switches reliably triggered system responses on connected nodes. This was a critical step in validating the system's ability to operate independently of the mobile application, especially in scenarios where users prefer tactile, on-device interaction. The testing process was divided into three sub-tests, described below.

1. Trigger Alarm

For this test, the user manually toggles the "Trigger Alarm" switch located on the physical Gadget. Upon activation, the system is expected to promptly send a command to the designated node, which then sounds the onboard buzzer as an audible alert. This immediate response serves as a confirmation that the Gadget is correctly interfacing with the node. The user may then deactivate the alarm by flipping the same switch again, using the mobile application, or toggling the "Turn Off Alarm and Lights" switch. A successful test confirms proper signal handling between the Gadget and the node.

2. Turn On Lights

In this test, the user flips the "Turn On Lights" switch on the Gadget. This action should result in the immediate activation of the node's connected lighting components. Similar to the previous test, lights can be turned off using the same switch, the mobile application, or by utilizing the "Turn Off Alarm and Lights" switch. This test confirms the Gadget's ability to control lighting peripherals independently of mobile input.

3. Turn Off Alarm & Lights

This combined test validates the system's ability to reset all active peripherals from a single hardware command. After triggering both the alarm and lights using the previous tests, the user activates the "Turn Off Alarm and Lights" switch. A successful result is indicated by the deactivation of all node-based outputs, effectively restoring the system to a neutral state.

VIII. RESULTS

This section presents the findings from our Engineering Specifications Demonstration, where we evaluated the system's performance across three critical metrics: Detection Accuracy, Response Time, and Detection Range. Each test was carefully designed to assess how well our system meets its intended functional requirements in real-world conditions.

1. Detection Accuracy

To evaluate detection accuracy, we conducted a series of 10 trials under standard operating conditions. With the full system powered on, a test subject entered the monitored area, triggering the passive IR sensor. We then recorded whether the person detection algorithm correctly identified the subject, which was verified through an audible buzzer alert.

Across the 10 trials, our system achieved an 80% success rate. Detection was consistent at close range, but accuracy declined slightly as the subject moved farther from the camera. Notably, while the detection algorithm maintained integrity, delays in data transmission impacted the timeliness of mobile notifications.

These results establish a reliable performance baseline while also highlighting areas for future improvement, particularly in enhancing detection range and optimizing data transfer for faster alert delivery.

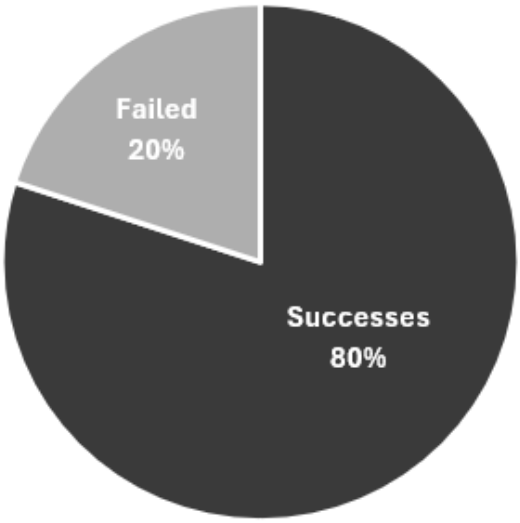

Fig. 3. Person detection success rates across 10 test trials under standard operating conditions.

2. Response Time

To evaluate system responsiveness, we conducted a timed test using a stopwatch and the mobile app. The procedure involved resetting the system, triggering motion detection by waving in front of the PIR sensor, and recording the time until the app received a person detection notification.

Response times ranged from 1.41 to 7.43 seconds, with an average of 3.56 seconds, indicating generally prompt performance. However, some variability was observed, with a standard deviation of 1.90 seconds and a variance of 3.59—mainly due to network latency and fluctuations in image processing under different ambient conditions.

Overall, the system was reliable, but improvements to processing speed could enhance real-time performance.

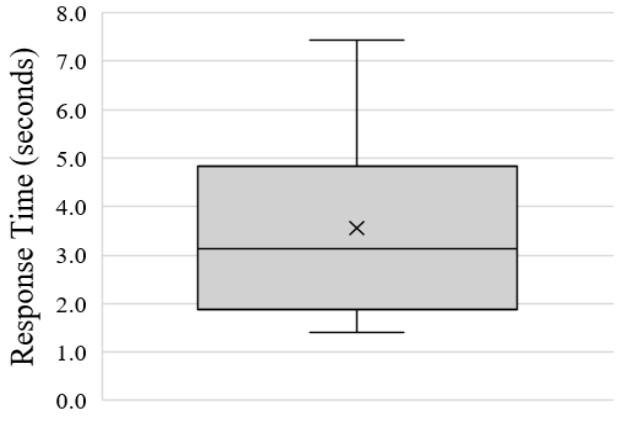

Fig. 4. Response Time Results Over 10 Trials.

3. Detection Range Results

To evaluate the system's detection range, we conducted tests at randomized intervals between 2 and 15 feet from the sensor node. In each trial, the subject approached from a fixed distance while the system attempted to detect motion, identify the person, and send a threat notification. These tests were performed under consistent lighting and environmental conditions to ensure measurement reliability.

Figure 5 summarizes baseline results at 2 ft, 5 ft, 10 ft, and 15 ft. While detections occurred at all distances, noticeable delays were recorded at the two farthest ranges, confirming processing and communication bottlenecks.

Test #	Distance (ft)	Detected	Delay Noted
1	2	Yes	No
2	5	Yes	No
3	10	Yes	>20 sec
4	15	Yes	>20 sec

Fig. 5. Baseline Detection Range Results

To expand on this, a series of 10 randomized range tests was performed to assess detection reliability across more granular distances. These covered ranges from ~4 ft to nearly 13 ft, with one failure occurring at 12.83 ft.

Figure 6 illustrates the results. The chart highlights the single failure at the farthest range (dark bar) compared to consistent detections at closer distances. The system achieved a 90% success rate, with the 5–7 ft range showing the highest consistency. The mean detection distance was 6 feet 11 inches, with a standard deviation of 2 feet 6 inches, indicating a moderate spread and a practical upper limit near 10 feet.

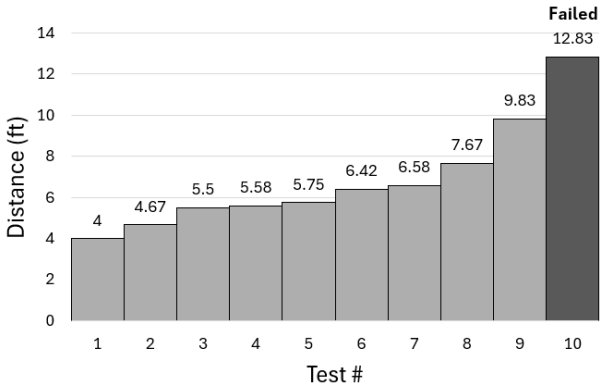

Fig. 6. Detection results across 10 trials. The system successfully detected a subject at all distances except for the 12.83-foot test, where the detection failed (indicated by the dark bar).

Performance at longer ranges is currently constrained by memory and processing limitations, in addition to HTTP latency between the ESP32 node and the mobile app. These constraints limit how quickly and reliably the system can analyze and transmit image data captured at farther distances, particularly under higher ambient noise or lower lighting conditions. Future improvements could include optimizing the detection model, intelligently reducing image resolution, and upgrading to higher-performance microcontrollers.

In summary, the detection range tests demonstrated that the system performs reliably within moderate distances, particularly between 5 and 10 feet. While detection is technically possible at ranges up to 15 feet, performance degradation at longer distances confirms the need for system optimizations. Incorporating more efficient communication protocols or integrating dedicated AI accelerators could also enhance responsiveness in future iterations. These results validate the effectiveness of our current design and provide clear directions for future improvements.

IX. CONCLUSION

SecureScape is a smart, deployable security system designed to detect and respond to potential threats in areas without reliable internet access. Through the integration of PIR sensors, onboard image capture, and person detection using lightweight AI models, the system provides a local, real-time response to motion events. By using a central Gadget controller and a mobile app, users are able to receive alerts and control system functions without external connectivity.

One of the key goals of SecureScape was to ensure that users could monitor and manage their security system easily and reliably. With the mobile app and physical gadget both serving as control hubs, the system offers flexibility and redundancy in how users interact with it. Our testing confirmed that the system responds within an average of 3.5 seconds and detects motion most effectively within a range of 5 to 10 feet.

The modular design of both the hardware and software also allows SecureScape to be expanded or adapted for different scenarios. Nodes can be added, system settings can be tuned, and updates can be made to support more advanced detection or communication features in the future.

Overall, this project demonstrates a complete, functioning security solution for remote or off-grid areas, built with a focus on usability, reliability, and real-world practicality.

THE ENGINEERS

Jaxon Topel is a 22-year-old Computer Engineering student. Jaxon plans to pursue his master's in electrical engineering at Purdue University or Georgia Tech. He is also actively searching for defense positions for a career in software engineering, autonomous systems, or GNC related fields to work at while he pursues his masters.

Dylan Myers is a 31-year-old Electrical Engineering student who has specialized in RF systems. He plans on taking his hands on experience from being an Avionics Specialist in the Air Force and an Engineering Technician at Northrop Gruman and blending them with traditional engineering practices. He hopes to create systems and processes for advanced RF development in the future.

Colin Kirby is a 21-year-old Computer Engineering student with a strong interest in application development & machine learning. He aspires to build full-fledged applications that create meaningful impact and serve the broader public. Colin is particularly passionate about the design and development process, and is focused on turning innovative ideas into practical, user-friendly solutions.

Phillip Murano is a 28-year-old Electrical Engineering student with a strong foundation in leadership and problem-solving, shaped by his time as a sergeant in the United States Marine Corps. He specializes in circuit design, power systems, and embedded systems, with a focus on developing efficient & practical solutions. Phillip is passionate about contributing to impactful engineering projects and continuous learning.

ACKNOWLEDGEMENT

The authors wish to acknowledge the assistance and support of Dr. Lei Wei. We are deeply grateful for his guidance and the time he has devoted to us.

We extend our sincere thanks to Mike Borowczak and Chinwendu Enyioha for graciously agreeing to serve on our group's review committee.

We also wish to recognize the invaluable support of Dr. Arthur Weeks. We are deeply thankful for his constant presence and the time he devoted to assisting us whenever we needed him.