Wild F.I.R.E. Sentry: A Connected Wildfire Intelligence and Risk Evaluation System

Timothy Seibert, Adam Le, Pranav Madan, Ryan Butnariu, Mikhail Gromov, Jafar Saniie

Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago, IL, U.S.A.

***Abstract*—In recent years, wildfires have grown in both frequency and severity, fueled by rising global temperatures, prolonged droughts, and extreme weather conditions. These fires have caused widespread destruction, claiming lives, displacing communities, and resulting in significant environmental and economic losses. This escalating threat highlights the need for faster, more intelligent wildfire detection methods that can operate continuously and scale across vast landscapes. In this paper, we present the system architecture and design flow of the *Wild F.I.R.E. Sentry*, a connected wildfire intelligence and risk evaluation system. This scalable and smart system will incorporate a combination of sensors and cameras to act as "noses" and "eyes" in the forest. Replacing traditional fire watchtowers, our project will incorporate sensor technologies, advanced IoT techniques, and AI algorithms to generate alerts of rapidly spreading wildfires. These alerts will be viewable on a real-time dashboard along with easy integration into automated messaging systems. This project is designed for early fire detection to reduce the response time of wildfire teams to emerging threats and, in doing so, help protect lives and minimize damage to infrastructure and the environment.**

Keywords— Wildfire detection, IoT, environmental monitoring, LoRa communication, AI object detection, real-time systems, fire risk evaluation, embedded systems

I. Introduction

In recent years, the devastating impact of wildfires has escalated dramatically. Major events, such as the 2023 Maui wildfire, which caused over $5.5 billion in damages and resulted in 102 fatalities [1], and the 2025 Palisades wildfire, responsible for an estimated $275 billion in economic losses and widespread displacement [2], emphasize an urgent need for proactive wildfire detection systems. According to the Environmental Protection Agency (EPA), both the frequency and severity of wildfires have notably increased over the last four decades, driven by climate change and prolonged drought conditions [3].

Recognizing these threats, the U.S. Department of Homeland Security (DHS) has prioritized developing advanced wildfire detection methods, supporting innovations combining sensor technologies and artificial intelligence (AI) [4]. For instance, DHS collaboration with *N5 Sensors* produced systems integrating environmental monitoring and AI-based analysis to deliver early warnings with sensitivity surpassing conventional smoke alarms by orders of magnitude [4].

Commercial solutions, such as Dryad Networks' *Silvanet* platform, deploy solar-powered gas sensors to detect the smoldering phases of wildfires early, often hours before traditional visual detection [5]. Another system, *SmokeD*, relies predominantly on camera networks using AI to visually identify smoke plumes and flames but lacks early gas or particulate detection capabilities, potentially missing critical early warnings [6].

Academic research has further explored integrated sensor networks coupled with computer vision algorithms. Chan et al. proposed an IoT-based wildfire monitoring system that integrates infrared, ultraviolet, and visible-light imaging sensors alongside environmental sensing to achieve robust detection capabilities [7]. Moreover, Haroun demonstrated the effectiveness of YOLO-based deep learning models, such as YOLOv8, for real-time detection of wildfires from image data, emphasizing the importance of accurate and reliable visual detection algorithms [8]. Recent improvements with YOLOv10 provided even higher detection accuracy and efficiency, reducing inference latency and increasing the system's responsiveness, thus further advancing AI-driven detection methods [9].

Our efforts are guided by fire weather criteria from the National Weather Service, such as high-risk definitions based on humidity and wind [10]. Sensors are integrated using industrial communication standards like RS485, Modbus [11], and LoRa UART-based transmission [12][13]. These technologies enable robust data transmission in field-deployed wildfire sensing networks. Wild F.I.R.E. Sentry synthesizes these leading-edge research efforts by integrating the strengths of each into a cohesive, scalable platform. Inspired by DHS-supported frameworks [4] and academic advancements in environmental sensing and AI-driven detection [7][8][9], our system leverages a distributed network of sensor nodes. Each node integrates sensors for temperature and humidity (DHT22), gas detection (MQ135), and wind measurement (UWD01-SD ultrasonic anemometer), following similar sensor synthesis principles as highlighted by Chan et al. [7]. These nodes communicate via a LoRa-based network, providing extended coverage and efficient power management inspired by IoT best practices [5][7][13].

Moreover, incorporating a YOLOv10 object detection model, enhanced with techniques such as focal loss [14] and label smoothing [15], reflects the latest developments in optimizing detector robustness and addressing class imbalance in complex environments. This visual analysis capability provides essential confirmation of potential wildfire events, addressing limitations observed in purely sensor-based or visual-only solutions. Through the integration of environmental sensor data and AI-validated visual inputs, Wild F.I.R.E. Sentry aims to dramatically reduce detection-to-response times, facilitate earlier interventions, and mitigate risks posed by wildfires to communities and ecosystems.

II. System Overview

The system consists of remote sensor nodes equipped with microcontrollers and a central Raspberry Pi hub. Each node collects environmental data and transmits it via LoRaWAN to the hub, which aggregates, evaluates, and responds to detected threats. When fire risk thresholds are met, a camera module captures visual evidence, processed by an onboard YOLOv10 model. Results are displayed in a live dashboard for response teams.

979-8-3315-6026-3/26/$31.00 ©2026 IEEE

A. *Sensors System*

The Wild F.I.R.E. Sentry system leverages multiple environmental sensors connected through Arduino Nano microcontrollers to gather real-time data essential for wildfire detection. Each sensor node includes several key components, as listed below and depicted in Figure 1:

- **Wind Sensor (Ultrasonic Anemometer):** Captures wind speed and direction without moving parts, making it suitable for long-term outdoor deployment.
- **Smoke and Gas Sensor (MQ135):** Acts as the primary fire detection sensor by measuring gas concentrations like carbon monoxide (CO) and carbon dioxide (CO_2) released during combustion.
- **LoRa Module (RYLR998):** Communicates sensor data wirelessly to the central node using low-power, long-range transmission.
- **Humidity and Temperature Sensor (DHT22):** Monitors atmospheric conditions that influence wildfire behavior.
- **Soil Moisture Sensor (HW-101):** Provides data on moisture levels in the ground, critical for evaluating the potential for fire spread.

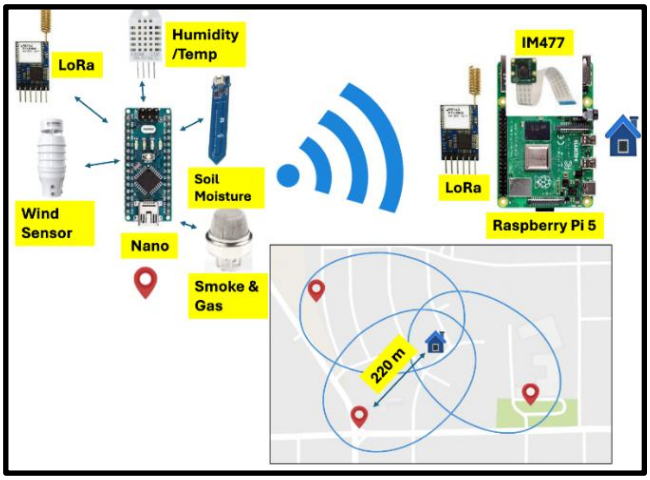

Figure 1. Sensor Overview

B. *Network System*

The Wild F.I.R.E. Sentry deploys a distributed network architecture, depicted in Figure 2, placing multiple sensor nodes across a targeted landscape to ensure comprehensive area coverage. Each node operates independently, powered by solar energy to maintain continuous monitoring without reliance on external power sources. Nodes transmit data over LoRaWAN, a scalable, low-power wide-area network protocol optimized for IoT applications. The central hub, located strategically to maximize network range and reliability, collects data from multiple remote sensor nodes. This central station is equipped with a Raspberry Pi that aggregates and processes sensor data, and an AI-driven camera system intended for wide-angle. It is powered by solar energy solutions.

C. *Dataflow*

Sensor nodes periodically transmit environmental data packets—including temperature, humidity, wind speed, soil moisture, smoke levels, alert levels, and node locations—to a central Raspberry Pi via LoRaWAN [13], where preliminary analysis identifies potential fire risks or active wildfires. This processed data is then managed by a backend service on the Raspberry Pi that converts the information into JSON format to be dynamically served to an externally hosted web dashboard built with HTML, CSS, and JavaScript. Following standard IoT practices [7][13], the dashboard provides real-time visualizations, geographic risk mapping, and fire alerts, while also archiving historical data in CSV format for offline analysis. As depicted in Figure 3, this streamlined data pathway ensures that critical wildfire detection information remains easily accessible for timely response by first responders, local authorities, and community members..

D. *AI Model Overview*

Figure 4 exemplifies the system overview of the AI model, which takes in an untrained image along with a .pt file and computer vision packages such as YOLO, cvzone, and cv2. Each box in each result is iterated on, and based on the assigned confidence score, the appropriate bounding box will be applied based on the class label. Upon detection of fire or smoke, a confidence score is applied to the image, and a message is sent to the main script that a fire has been detected.

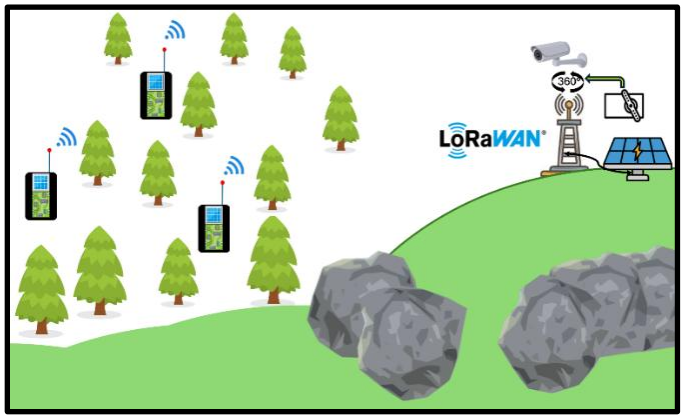

Figure 2. Network Overview

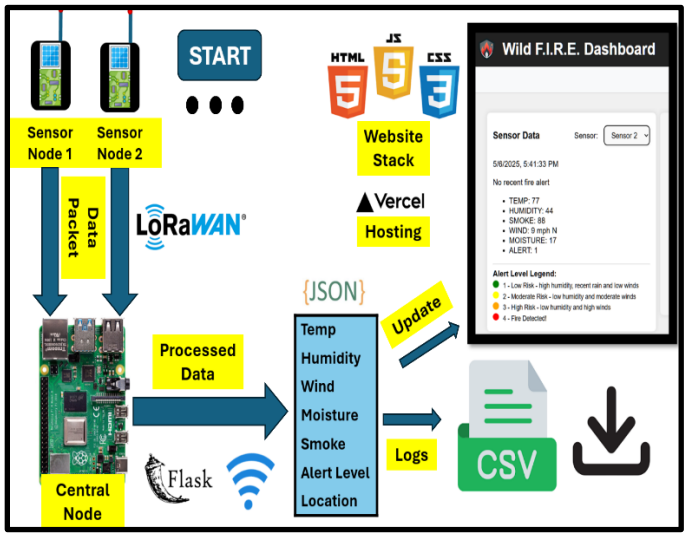

Figure 3. Dataflow Overview

III. Hardware Implementation

A. *Microcontrollers*

The Arduino Nano microcontroller forms the backbone of each remote sensor node. It was selected for its compact footprint, low cost, ease of use, and low power consumption. The Nano reads data from various analog and digital sensors using protocols like I2C, SPI, and UART. It processes this data locally and formats it into transmission-

ready packets. These packets are then sent to the Raspberry Pi via a LoRa module. The Nano's compatibility with numerous sensor libraries and communication modules makes it an ideal choice for scalable IoT applications like ours.

The Raspberry Pi 5 functions as the central node in the system, performing both data aggregation and high-level processing. receives environmental data from the LoRa transmission, processes it, and sends it to the real-time web interface via Flask. The Pi also supports camera integration through the IMX477 module, used for optical fire verification using YOLO [8][9]. It supports LoRa communication [12] for our current transmission system, while future upgrades can incorporate LoRaWAN for scalable networking [13]. Given its built-in internet capabilities, the Raspberry Pi can upload data to cloud dashboards or enable remote access, making it ideal for real-world deployment. In the future, LTE or Starlink connectivity may replace Wi-Fi for off-grid operation.

B. Sensors – UWD01-SD Anemometer

The UWD01-SD Anemometer is our ideal anemometer. This ultrasonic wind sensor accurately measures wind speed and direction. Having 2 pairs of transducers located in the upper housing, ultrasonic pulses are sent to the other transducers, seen below. Depending on wind speed and direction, the pulses received can differ from the transmitted pulses. The differences in the time it takes these pulses to be received are influenced by the speed and direction of the wind. By measuring these differences, calculations of both wind speed (0-40 m/s) and direction (0-360 degrees) are made. Speed is obtained in meters per second, but our system adjusts this to miles per hour with the constant 2.23694. Wind direction data is most effectively interpreted by converting numerical degrees into cardinal and intermediate directions. However, sensor placement is critical; accuracy may be compromised by obstructions such as trees, particularly as wind patterns shift at higher altitudes.

Since the anemometer uses the RS485 protocol to communicate and the Arduino cannot directly read this, a module converts RS485 to TTL for the Arduino. The RS-485 standard is useful for long-distance communication (even up to 1 km) and can reach speeds up to 10 Mbps. 2 wires are used to provide half-duplex (bidirectional communication one device at a time). With 1 master and multiple slaves, requesting data can only be made by the master. Logic '1' is considered 2-6 V between A and B, while '0' is –2-6 V. Modbus is a protocol used to communicate with RS485 for the Arduino to communicate with a slave device [11]. This works by specifying what format the data is to be transmitted. An inquiry frame is sent from the master to the slave, comprising what function is being used, the address of registers to read (speed & direction), the number of points, and an error check for low and high.

C. Sensors – DHT22 Temp. & Humidity Sensor

The DHT22 sensor can measure both humidity and temperature, both of which are vital to determine if a fire is present. The humidity sensor works by having a substrate that can absorb humidity in the air. Depending on the moisture, the conductivity of the substrate changes, affecting the resistance between the two electrodes that can be analyzed. The temperature sensor works with a thermistor, a variable resistor whose resistance decreases when the temperature increases. Both sensor data are fed from one pin on the sensor to the Arduino using a protocol that can be utilized from the DHT library. Our project uses both to determine if a fire's heat is present and if it is likely to spread (depending on humidity levels).

D. Sensors – MQ135 Gas Sensor

The MQ135 gas sensor, communicating through analog voltage signals, works through an electrochemical reaction that occurs when certain gases interact with oxygen on the surface of the sensor. Once a gas, like carbon dioxide (CO2), encounters the surface of the sensor, a signal is generated. As more gas meets the surface, it creates a lower resistance that is represented by an integer value. This will be used as the main fire detection sensor. If the threshold set on the sensor is reached, it is an automatic alert for a fire, unlike the other sensors, which are just signs of a fire risk.

E. Sensors – HW-101 Soil Moisture Sensor

The HW-101 Soil Moisture Sensor contains VCC, GND, *Aout*, and *Dout*. The VCC provides the power, while the GND is the ground. *Aout* gives information about moisture levels in the soil, while *Dout* gives information about soil moisture status in binary form. The sensor probe in the moisture includes two conductive pads which investigate the moisture in the soil, while the electronic module processes the data readable by humans. The soil moisture has 2 bits that measure the electricity flow. The amount of water in the soil is proportional to the conductivity of electricity in the soil.

This sensor is an optional component of the package, depending on the environment where the system is deployed. In areas where fire is less likely to spread through ground foliage—such as a dry forest—this sensor may be omitted; in these cases, soil moisture is less critical because ground-level fire spread is not a primary concern.

F. Arducam IMX477

The Arducam was integrated with a Raspberry Pi for setup. While the camera lacks an autofocus feature and requires manual calibration, it is programmed to capture photos automatically upon detecting sufficient indicators of fire.

G. RYLR998 LoRa Module

LoRa modules are very powerful radio transmission modules. For our system, they will be used to connect the sensor packages to the central Raspberry Pi node. The module that we chose uses the Universal Asynchronous Receiver-Transmitter (UART) protocol, which is considered a bit outdated, but the data rate limitations will not affect our project [12]. For the United States, LoRa uses the 916 MHz band for communications, which is reserved for medical and research purposes. This means that we don't need a license to operate across the band, limiting costs and expanding scalability.

The current system uses a point-to-point (P2P) LoRa setup to simplify prototyping. Each sensor node directly communicates with the Raspberry Pi without intermediate relays or gateways. However, the architecture is designed with future scalability in mind. Transitioning to LoRaWAN, a more structured protocol that enables star-topology networks with multiple nodes, gateways, and backend servers, would significantly increase range and networking capabilities. LoRaWAN also adds features like built-in encryption, automatic retries, and device provisioning [13].

Our current LoRa module, RYLR998, works under the 916 MHz band and, when a basic wire antenna was added, achieved a range of ~220 meters across an urban landscape. The range had several obstructions, including concrete buildings, trees, and foliage. While the testing was not exhaustive, it provides a baseline range for our unoptimized system. Other models of LoRa modules boast up to 1 km of range. Improvements to signal strength and hardware could amplify our signal to cover a larger area if needed,

H. Power System

The power solution involves a large-capacity battery with a built-in solar panel. This solution was necessary for several reasons: convenience, long battery life, and stable power. Connecting a small panel to a small-capacity battery is more work than necessary for our system. To avoid short battery life, this solar battery can charge itself during the day while supplying power to the system. For periods of the day when solar energy is unavailable, the capacitor can supply power to itself.

IV. Software Implementation

A. Arduino Nano Code

The sensor nodes utilize custom C++ firmware running on Arduino Nano microcontrollers. The code integrates multiple environmental sensors, including smoke detection (MQ135), temperature and humidity sensing (DHT-22), soil moisture measurement, and wind speed/direction via an ultrasonic anemometer (Modbus-based RS485 protocol). Sensor readings are gathered sequentially, processed into readable units, and then packaged into a structured LoRa radio message.

The Arduino uses the RH_RF95 LoRa library to transmit sensor data securely and wirelessly at 915 MHz. The data is structured into null-terminated character arrays to ensure memory safety during packet assembly and transmission. Sensor readings are converted from numeric sensor values into fixed-size character buffers using standard functions like itoa() for integers and dtostrf() for floating-point numbers. These buffers are carefully concatenated into a single, well-defined transmission packet, explicitly bounded to prevent buffer overflows or memory corruption. This methodology ensures that each LoRa data packet maintains integrity and predictable memory usage, facilitating reliable parsing and error-free processing on the receiving Raspberry Pi.

B. AI Model & Training

The AI-based object detection model relies on several key packages, including the YOLO framework [9], which provides functionality for bounding box generation, image preprocessing, and confidence scoring. The model utilizes the PyTorch file YOLOv10-FireSmoke-M.pt [16], processes input images by resizing them, and then performs inference to generate detection results. For each detected object, the model iterates through the bounding boxes, extracting and adjusting the associated confidence scores while also retrieving the class indices. Class labels are obtained via the .cls attribute. When the confidence score exceeds a threshold of 0.25, a mapping function is applied to determine the bounding box coordinates. The bounding box representation is conditioned on the detected class, specifically distinguishing between smoke and fire. Further refinement of confidence thresholds and scoring mechanisms remains an area for future work.

C. Backend

The backend of our system leverages Flask, a lightweight Python web framework, providing efficient routing, HTTP request handling, and dynamic content serving. Flask bridges the gap between the hardware layer and the user interface by processing incoming JSON-formatted sensor data received from the Raspberry Pi processing scripts. Through RESTful APIs, Flask ensures:

- Real-time updating of sensor data, including temperature, humidity, wind speed, soil moisture, smoke concentration, and calculated alert levels.
- Automated logging of historical sensor data into CSV format, supporting offline analysis and long-term record-keeping.
- Image processing workflows, where Flask manages the uploading and displaying of AI-annotated images captured by the camera system upon fire detection.
- Robust support for multi-device access through local IP addressing or cloud-based hosting solutions for scalability and accessibility.

D. Frontend

The *Wild F.I.R.E. Sentry* dashboard provides a user-friendly frontend developed using HTML, CSS, and JavaScript. HTML structures the interface components such as interactive maps, data display panels, real-time charts, and multiple pages. CSS ensures a responsive, visually coherent, and intuitive design. JavaScript enhances functionality, providing asynchronous data fetching from the backend APIs, dynamically updating visual elements in real-time, including:

- A real-time geographical map displaying the positions and status of sensor nodes.
- Interactive plots for alert levels, wind speed, and humidity.
- Real-time display of sensor data enabling immediate assessment of fire risk (Figure 4).
- Automatic triggering of visual alerts and image displays whenever AI-confirmed fire conditions are detected.
- Multi-page design, including a home page to add more networks and a history page for accessing data logging.

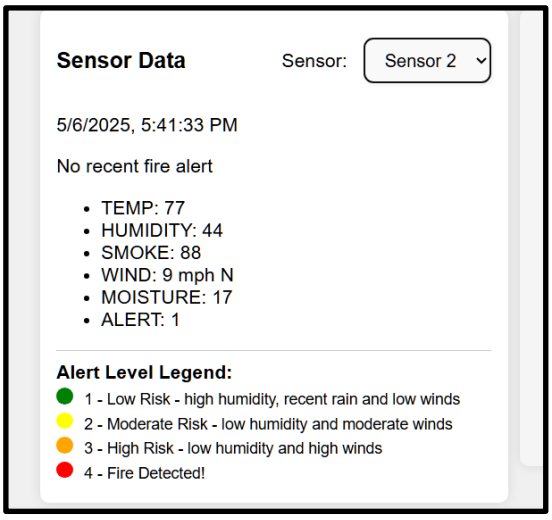

Figure 4. Dashboard Sensor Data

V. Results and Analysis

A. Sensor Calibration

The first part of our system to test was the fire risk levels. This meant calibrating and testing how wind speeds and humidity/moisture levels affected our system's fire risk values. Our key environmental datapoints were wind speed and humidity. These parameters were set based on standards used by the National Fire Weather Service [10]. The red flag warning, high risk in our case, is when the system detects less than 15% humidity and greater than 25 mph wind speed. The moderate risk was set when wind speeds were greater than 18 mph, and the humidity was less than 30%. Low risk was any other set of values. We found that these two data points, wind and humidity, were most critical to setting alert values, while other data points, like temperature and soil moisture, were mostly overlooked. While they can provide insight into the region, such as more accurate temperatures and an understanding of rainfall patterns, they don't provide something that isn't already available using advanced weather models.

B. Fire Detection

Sensor-based fire detection was validated through three distinct testing methodologies: targeted CO_2 exposure via human breath, controlled match ignition, and a small-scale open bonfire. The match tests provided critical baseline data for the MQ135 gas sensor's response to smoke concentrations. For testing the AI model and web dashboard during inclement weather, targeted CO_2 breath exposure served as a reliable proxy for air quality fluctuations. The gas sensor exhibited high sensitivity to combustion gases, particularly during the bonfire test,

where a steady increase in analog values successfully triggered the system's fire alert. Furthermore, the sensor demonstrated reliability in detecting sustained smoldering, a capability essential for early-stage wildfire intervention. Computer vision testing further validated the system using images of both fire and smoke; as illustrated in Figures 5 and 6, the model correctly identified these elements under varying conditions.

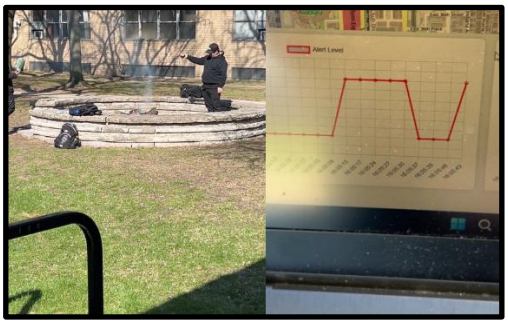

Figure 5. Testing of the System with Bonfire

Figure 6. Fire Detection AI Model

C. *Website*

The real-time web interface successfully displayed sensor readings, issued alerts, and updated the fire risk indicator. During live tests, sensor data flowed into the dashboard at 2-second intervals, and AI-verified fire alerts triggered automatic image uploads. The CSV export functionality worked as intended, achieving some level of data logging for review. During live testing, the system demonstrated a data propagation latency of under five seconds from the Arduino node to the dashboard. While this serves as a successful proof-of-concept for real-time responsiveness, future iterations will focus on rigorous quantitative benchmarking against industry-standard latency metrics to validate performance in high-density network environments. Overall, we created a solid user interface shell and sufficient backend features ready for scaling and deployment.

VI. Conclusion

The *Wild F.I.R.E. Sentry* presents a practical and scalable solution for early wildfire detection. Through the integration of temperature, humidity, gas, wind, and moisture sensors, the system provides layered insights into fire risk and presence conditions. A Raspberry Pi central node consolidates this data, verifies threats using computer vision, and communicates alerts through a real-time web dashboard. While the current experimental results demonstrate the fundamental feasibility of the sensor-to-dashboard pipeline, subsequent research will focus on large-scale field calibration and the collection of quantitative accuracy metrics (including precision, recall, and false-positive rates) to validate the system's reliability in complex, real-world forest environments.

Field testing demonstrated that the system can effectively detect smoke and fire-related signatures through a combination of chemical gas sensing and AI-based visual analysis [8][9]. The incorporation of LoRa communication [13,14] enabled reliable deployment of distributed sensor nodes at distances exceeding 200 meters, even in obstructed environments. By minimizing the latency between wildfire ignition and emergency response, the Wild F.I.R.E. Sentry presents a promising approach for enhancing early detection capabilities and mitigating risks to communities, natural resources, and ecosystems.

This platform could be refined by integrating servo motors to enable panoramic camera coverage, adopting LoRaWAN to support scalable multi-node networking, improving YOLO model performance through non-max suppression and quantization, tuning parameters like confidence, and implementing full 360-degree camera coverage. Additional enhancements include transitioning to a cloud-based dashboard and developing ruggedized enclosures for both sensor and central nodes to ensure durability across diverse environmental conditions.

References

[1] C. of Hope, "Convoy continues to serve Maui residents after wildfire," Convoy of Hope, Feb. 26, 2024. [Online]. Available: https://convoyofhope.org/disaster-services/2023-maui-wildfire

[2] V. Babrauskas, "The Palisades Fire of Los Angeles: Lessons to Be Learned," Fire, vol. 8, no. 8, p. 303, Jul. 2025. [Online]. Available: https://doi.org/10.3390/fire8080303

[3] United States Environmental Protection Agency, "Climate change indicators: Wildfires," US EPA, Jul. 2016. [Online]. Available: https://www.epa.gov/climate-indicators/climate-change-indicators-wildfires

[4] "Technology to reduce the impacts of wildfires," Homeland Security, Sep. 19, 2024. [Online]. Available: https://www.dhs.gov/science-and-technology/technology-reduce-impacts-wildfires

[5] "Silvanet wildfire detection," Dryad Networks. [Online]. Available: https://www.dryad.net

[6] "Wildfire detection systems - forest fire & smoke automatic detection," SmokeD. [Online]. Available: https://smokedsystem.com

[7] C. C. Chan et al., "IoT-based wildfire detection systems," IEEE Sensors J., vol. 23, no. 6, pp. 2843-2852, 2023.

[8] A. Haroun, "Wildfire detection using YOLOv8," IEEE Access, vol. 12, pp. 40567-40575, 2024.

[9] Ultralytics, "YOLOv10," Ultralytics.com, 2024. [Online]. Available: https://docs.ultralytics.com/models/yolo10

[10] US Weather, "Fire weather criteria," Weather.gov, 2022. [Online]. Available: https://www.weather.gov/gjt/firewxcriteria

[11] "What is Modbus and how does it work?" Schneider Electric USA, Mar. 19, 2013. [Online]. Available: https://www.se.com/us/en/faqs/FA168406/

[12] "UART interface 868/915 MHz LoRa® antenna transceiver module datasheet," Reyax, July 5, 2023. [Online]. Available: https://reyax.com//upload/products_download/download_file/RYLR998_EN.pdf

[13] The Things Network, "LoRaWAN architecture," The Things Network. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/architecture/

[14] P. Yun et al., "Focal Loss in 3D Object Detection," IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 1263-1270, Apr. 2019. [Online]. Available: https://doi.org/10.1109/LRA.2019.2894858

[15] C.-B. Zhang et al., "Delving Deep Into Label Smoothing," IEEE Trans. Image Process., vol. 30, pp. 5984-5996, 2021. [Online]. Available: https://doi.org/10.1109/TIP.2021.3089942

[16] T. Ng, "YOLOv10-Fire-and-Smoke-Detection," Hugging Face, 2024. [Online]. Available: https://huggingface.co/TommyNgx/YOLOv10-Fire-and-Smoke-Detection