Forest Fire Forecast IOT
Final report

Jinyan Yi, yijinyan, alexjy.yi@mail.utoronto.ca

Zihao Gong, gongziha, zihao.gong@mail.utoronto.ca

Chris Huang, huan1940, chrim.huang@mail.utoronto.ca

Problem statement:

In recent years, global warming and extreme weather have significantly increased the frequency of forest fires, and the economic and property losses caused by forest fires are becoming larger. For example, the recent 2025 Los Angeles fire has caused more than 16,000 structures to be damaged, more than 30 people to die, resulting in $250 billion loss as well.

Traditional fire monitoring methods such as manual patrols or satellite monitoring suffer from high cost, large time delay, and low accuracy. They also may not effectively predict the risk of fires and cannot meet the growing needs for rapid detection and response to fire alarms. To achieve a balance between cost, response time and energy consumption, we proposed and designed this IoT-based intelligent forest fire monitoring system. The system can not only provide immediate feedback on the spread of fire but also predict the probability of a fire in the future through 8 different data. By sending and storing data on cloud servers, we ensure resilience and availability of the system to make sure proper warnings are delivered accurately and timely.

Description of the system:

The overall architecture of this system is shown in Figure 1, which includes sensor data acquisition modules, an edge visual recognition module, an edge data processing and communication module, a data aggregation and relay node, a cloud server and database, and a data display module. In the cloud we are hosting a Flask server that is connected to a firebase database to store the most recent data readings from sensors, and record the timestamp they are collected.

Sensor data acquisition modules include wind speed sensor, gas sensor, temperature and humidity sensor module. We have selectively chosen sensors affordable and accessible to us, without compromising in their functionality and adhering to our power consumption limits. In short, total cost of the system is $51 including import taxes. See Table 1 for detail.

We use DHT11 as a temperature and humidity sensor, which has been included in the professor's lab kit, so using it can reduce the cost of our project. DHT11 can be used to detect changes in ambient temperature and humidity. Temperature and humidity are important parameters for predicting the probability of fire. An important condition for natural forest fires is that the air is dry enough and the temperature is high enough, in which case flammable materials have a higher chance to be ignited.

We used four gas detectors: MQ-2, MQ-7, MQ-9, and MQ-135. These gas sensors' conductivity will change when exposed to specific gases. By comparing the change in output voltage, we can roughly estimate the concentration of a specific gas. MQ series gas detectors often cannot provide very accurate results. As the gas concentration increases, its results become more inaccurate because each MQ sensor is designed to detect the concentration of multiple gases in combination. For example, MQ135 can not only detect carbon dioxide, but its output voltage is also affected by ammonia gas. The main reason we chose the MQ series is that they are cheap enough, each sensor only costs about 1.5 Canadian dollars. In addition, the amount of gas changes is very large during a fire, and sacrificing some accuracy in exchange for a low price is our best solution. We use MQ-2 to detect smoke, MQ-7 to detect carbon monoxide, MQ-9 to detect flammable gas, and MQ-135 to detect carbon dioxide. These four parameters are usually very low in the normal air, but once a fire occurs, their values will rise rapidly, such as carbon monoxide. We use these gas sensors to determine whether a fire is happening.

We deployed a wind speed sensor because strong winds can increase the probability of forest fire generation and accelerate the spread of fires. Detecting wind speed can help us better decide the probability, the severity and the development trend of fires. The wind speed sensor can only generate

a voltage of 1.37V at maximum wind speed. Compared with the 3.3V requirement of ESP32, its output will not damage the microprocessor.

We use ESP32 CAM as the edge visual recognition module. We use the camera built into ESP32 CAM to take a picture and scale it to a resolution of 96*96 to reduce the computing power required for subsequent model calculations. Then, the localized CNN model deployed on esp32 CAM predicts the probability of a fire from the picture. We build and train the model on a computer with GPU. When training the model, we use more than 5,000 different fire scenarios, including daytime fires, nighttime fires, far away fires, and close fires, to ensure that the model can accurately judge the probability of a fire in a complex field environment. After the model is trained, we use pruning and quantization techniques to further reduce the model's parameter size and finally deploy it to ESP32CAM for localized operation.

We use an ESP32 as an edge data processing and communication module, It uses hard connections to gather data from various sensors, obtains the fire prediction probability of the edge visual recognition module through Bluetooth, performs preliminary processing, and transmits the data to the remote Raspberry Pi node through the LoRa module. We use Bluetooth to obtain the fire prediction probability because in actual situations, our camera module needs to be deployed in a place with a wide field of view, and we do not want to use the hard connection, which may limit the deployment location of esp32CAM. At the same time, we use the LoRa module to achieve long-distance and low-power communication because we hope to deploy our detection system as sparsely as possible in the forest. We did not choose the esp8266 of the Lab kit because esp32 has stronger performance and can support Bluetooth.

We use Raspberry Pi as data aggregation and relay node, and implement a data upload method similar to the "IoT with the Google Cloud" lab. First, the Raspberry Pi is also connected to a LoRa module to obtain the data sent by the edge data processing module through LoRa.

After receiving the data, the Raspberry Pi will forward data to our cloud Flask server. It then uploads the data to Firebase and then displays it on the web page. The web page will have detailed data of eight parameters (4 gases, temperature, humidity, wind speed, image detection probability), and use red colour to clearly indicate which parameter currently exceeds the threshold. However, flask server is having slightly different set of APIs for the purpose of simplicity and readability:

1. /api/sensors
   GET: Retrieves all sensors
   POST: Creates a new sensor (requires description and sensor_name)
2. /api/sensors/<id>
   GET: Retrieves specific sensor data
   PUT: Updates sensor data
   DELETE: Removes a sensor
3. /api/debug
   GET: Returns debug information including latest data, history, and warning counts
   Web Interface Endpoints. This is also the endpoint where the count of warnings will be produced in order for the LCD display to show it.
4. /dashboard
   GET: Main monitoring dashboard with real-time sensor data and history. Figure 2.

We use ESP8266 as a data display module from the lab kit, which is connected to an LCD monitor. ESP8266 will continuously obtain the latest data from the cloud and display the number of parameters exceeding the threshold and their corresponding station ID on the LCD monitor. This allows the deployment and simultaneous monitoring from multiple nodes, facilitating coverage of large forest areas in reality.

Challenges throughout the process:

1. **ESP32's pins are not enough:** we have four gas sensors, a temperature and humidity sensor, a wind sensor, a LoRa module, and an ESP32 CAM that need to connect to ESP32. The pins of the ESP32 development board are limited, and only a part of them supports analog-to-digital conversion, so we need to reduce the pins used as much as possible. We originally planned to use UART for data transmission between ESP32 and ESP32 CAM, but later, we found that Bluetooth can also achieve efficient and stable transmission, and using wireless Bluetooth communication saves us two pins. Connecting four gas sensors directly to ESP32 requires four ADC pins, and the wind speed sensor also requires an ADC pin, so we chose to connect four gas sensors to an ADC converter ADS1115 and then ADS1115 communicates with ESP32 through I2C, which only takes two pins of ESP32 (SDA, SCL), it saves us two pins compared to directly using four ADC pins for gas sensors. Moreover, ADS1115 has a higher 16-bit ADC resolution, which can provide higher data reading accuracy compared to the 12-bit ADC resolution of ESP32. Totally we use different strategies to save 4 pins of ESP32.

2. **CNN model is too large:** The model we trained at the beginning was very large and not suitable for deployment on edge devices. So first, we studied MobileNet V3 released by Google and implemented many of its architectures to reduce the amount of computation, such as using HardSwish Activation instead of Relu Activation, using reverse residual to replace residual network, adding SE module, etc. At the same time, we also used many of our own unique designs, such as replacing the second last fully connected layer with a Global average pooling layer to reduce model parameters. We hard pruned the model to delete some unimportant channels of the model, which reduced the size of the model from 184kB to 96kB and the accuracy of the model from 99.1% to 95%. Finally, we used Edge Impulse to complete the quantization of the model from int64 to int8, and the accuracy became 88.4%, which enabled the model to run on ESP32 CAM with an inference time of less than 1s.

3. **Bluetooth communication:** ESP devices communicate asynchronously. To prevent incomplete data reception, we choose ESP32 CAM to run a loop every second to send the predicted probability and ESP32 to run a loop every five seconds to read from the Bluetooth buffer. When reading, ESP32 will record data from the first "\n" (the line break at the end of the first data) until the second "\n" (the line break at the end of the second data). In this way, we can ensure that the data obtained is complete in most cases. Finally, after reading the data, we will clear the Bluetooth buffer to prevent reading older historical records the next time.

4. **Abnormal readings of gas sensor and ESP32 CAM:** At the beginning, we used ESP32 to power the gas sensor, but the data of the gas sensor was always abnormal. After checking many replies in the MQ system forum, we found that many people suggested to power the MQ sensor independently, because there is a heating wire inside the MQ sensor that will drop the current in the circuit, especially since we have 4 MQ sensors, which will cause each MQ sensor to not get the required power. So in the end, we chose to use the Type C adapter board to get a stable current directly from the power supply to ensure the normal operation of the MQ sensor. ESP32 CAM was unable to return data at first, and later we found that the inference of the model required higher power, and our data line could not provide it, so we changed to a more efficient line to ensure the work of ESP32 CAM.

Evaluation of our system

1. **Functional Evaluation:**
   In evaluating the functional performance of our forest fire detection system, we focus on four key capabilities: accurate sensing, reliable image-based prediction, stable communication, and clear system integration. First, the system successfully collects accurate and consistent sensor data using four different gas sensors (MQ-2, MQ-7, MQ-9, MQ-135), along with temperature, humidity, and wind measurements, ensuring comprehensive environmental monitoring. We regret to admit that each gas sensor has an approximate ±20% of error due to limited budget requirements. They all capture the reading related to the fire and will give a warning if it passes the threshold, which is kind of decreasing the probability of missing the sign of fire. However the cost is these gas sensors are the most power consumers, they require about 3000 mW power which is over our expectation. Maybe in the future we are going to set up periodic switch gas sensors and have a longer time of sleep. Second, the ESP32-CAM performs on-device fire detection using a quantized CNN model, maintaining high accuracy (88.4%) while operating within tight memory and time constraints—typically under 1 second per inference. This is a very strong function that can give high accuracy predictions and even if the fire distance is far away we can still detect the fire. Third, the communication between nodes is stable and reliable, utilizing Bluetooth Classic for short-range transmission between ESP32 devices and LoRa for long-range, low-power data forwarding to the Raspberry Pi. However, here we used normal Bluetooth which will consume more power than BLE, and may improve in future. Finally, the system demonstrates smooth integration between all components—sensor input, local processing, wireless communication, and cloud synchronization via the Raspberry Pi and Firebase. Since the system is designed for early warning rather than real-time emergency response, ultra-low latency is not a primary requirement. A detection and response window of a few seconds is sufficient to fulfill its purpose, allowing the system to prioritize reliability, power efficiency, and scalability in real-world deployments.

2. **Distance Evaluation:**
   For our system, distance is also a critical point. We tried to do the experiment with 1km which works in a relatively clear environment. We expect it to be able to transmit signals around 5 km in the forest, yet we have not done it between the woods yet.

3. **System Integration & Scalability:**
   In evaluating the system's integration and scalability, we found that all components—from sensors to communication modules—operated cohesively and reliably under typical field conditions. The seamless integration of gas sensors (MQ-series), environmental sensors (temperature, humidity, wind), and visual fire detection via the ESP32-CAM running a CNN model allowed for effective sensor fusion, which significantly enhanced detection accuracy and reduced false positives. Furthermore, the use of LoRa with duplex communication capability opens up strong scalability potential. In scenarios where multiple ESP32 nodes are deployed across a wider area, each node can act as both a data source and a relay. This enables nodes to receive data from farther sensors and aggregate them into a combined message before forwarding it to the central system. While this introduces some communication overhead, it allows for the creation of a more distributed and collaborative sensing network, capable of covering larger forest regions with minimal infrastructure and enhanced data reliability.

4. **Cost & Hardware Efficiency:**
   Our system demonstrates high hardware efficiency by achieving an effective balance between cost, performance, power consumption, and on-device intelligence. We strategically selected low-cost yet reliable components to ensure affordability without compromising functionality. For instance, we utilized commonly available gas sensors (MQ-2, MQ-7, MQ-9, MQ-135) to detect various fire-related gases, along with a temperature and humidity sensor (DHT11) to provide essential environmental context. To enhance fire risk assessment, we also included a relatively rare but valuable wind speed sensor, which plays a critical role in determining the potential spread and severity of a detected fire—something often overlooked in other low-cost systems.
   In addition to sensor selection, we optimized computational resources by leveraging the full potential of the ESP32-CAM module. Our CNN model, built and pruned specifically for edge deployment, uses less than 100 KB of memory and achieves inference in under one second. This allows for fast and efficient on-device fire detection without relying on cloud computing or high-bandwidth communication. The main ESP32 also handles multi-sensor integration and communication tasks with minimal CPU load, supporting Bluetooth and LoRa connectivity in parallel. Overall, our hardware design achieves a high level of integration and performance while remaining lightweight, low-power, and cost-effective—making it highly suitable for remote field deployments where both efficiency and autonomy are critical.

Related work:

1. **Related system:**
   We briefly introduce two established fire detection and prevention systems here. The first one is Dryad Silvanet, which costs hundreds for each node and it only detects gas. The AI is also on cloud. However, it has higher scalability since it's using LoRaWan which gives better node-to-node communication. Another product is the FireWatch Tower Sensor. They used cameras mounted on towers, which is similar to our system but has a better location. They also use central AI not the edge AI. The cost for them may be over 10k which is way higher than our system. Our system is a more integrated system, each node can handle all information by itself and just forward the result to the RPI and later to the central server.
2. **Related publications:**
   In the field of forest fire detection and prevention, machine learning based approaches were being widely deployed starting from the early dates of machine learning. Due to the nature of fire detection by camera, this problem is best solved by CNN (Convolutional Neural Networks) since it can learn and extract image features automatically and effectively. Many scholars employed CNNs into their fire detection system. Li and Zhao ([1]) examined many state-of-the-art CNN based models in their survey in 2020, and concluded that YOLO v3 achieved 83.7% accuracy and demonstrated strong robustness. Lee et al. ([2]) proposed a wildfire detection system using Unmanned Aerial Vehicle (UAV)) with deep CNN models and showed that GoogLeNet excel in their system with a 99.0% accuracy. More recently with the emergence of the attention mechanism [3], Almasoud ([4]) furthered the optimization of hyperparameter tuning for an attention-based CNN model using bacterial foraging optimization (BFO) algorithm and resulting in improved performance. Machine learning algorithms, especially CNN-based models, were proven to be both accurate and reliable in numerous works and surveys.

Appendix:

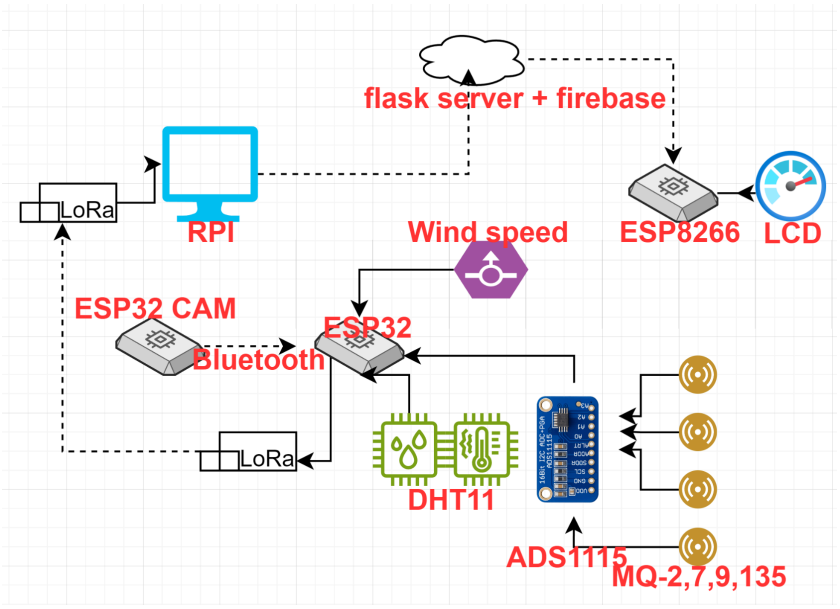

Figure 1. project system deployment

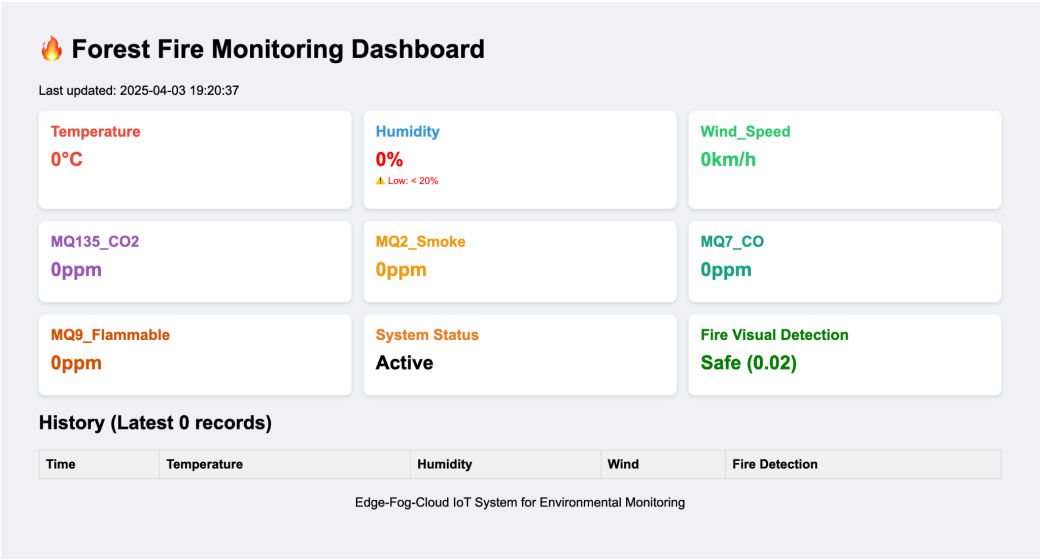

Figure 2. UI Dashboard showing all readings.

ESP32	10$
ESP32 CAM	10$
Type C adaptor	3$
ADS1115	3$
MQ 2,7,9,135	5$ for 4
Wind speed sensor	14$
DHT11	Lab Kit
LCD monitor	Lab kit
LoRa (SX1276-915M)	6$ for 2
Total	51$

Table 1. Total cost

References:

1. P. Li and W. Zhao, "Image fire detection algorithms based on convolutional neural networks," Case Studies in Thermal Engineering, vol. 19, p. 100625, 2020. [Online]. Available: https://doi.org/10.1016/j.csite.2020.100625]
2. W. Lee, S. Kim, Y.-T. Lee, H.-W. Lee, and M. Choi, "Deep neural networks for wild fire detection with unmanned aerial vehicle," in 2017 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, 2017, pp. 252–253. doi: 10.1109/ICCE.2017.7889305.]
3. A. S. Almasoud, "Intelligent deep learning enabled wild forest fire detection system," Comput. Syst. Sci. Eng., vol. 44, no. 2, pp. 1485–1498, 2023. [Online]. Available: https://www.scopus.com/inward/record.url?eid=2-s2.0-85133182094&partnerID=10&rel=R3.0.0]
4. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, \L. Kaiser, and I. Polosukhin, "Attention is all you need," in Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), Long Beach, CA, USA, 2017, pp. 6000–6010. [Online]. Available: https://dl.acm.org/doi/10.5555/3295222.3295349]