A Self-Sustained CPS Design for Reliable Wildfire Monitoring

YIGIT TUNCEL and TOYGUN BASAKLAR, University of Wisconsin - Madison, USA
DINA CARPENTER-GRAFFY, Pacific Northwest National Laboratory, USA
UMIT OGRAS, University of Wisconsin - Madison, USA

Continuous monitoring of areas nearby the electric grid is critical for preventing and early detection of devastating wildfires. Existing wildfire monitoring systems are intermittent and oblivious to local ambient risk factors, resulting in poor wildfire awareness. Ambient sensor suites deployed near the gridlines can increase the monitoring granularity and detection accuracy. However, these sensors must address two challenging and competing objectives at the same time. First, they must remain powered for years without manual maintenance due to their remote locations. Second, they must provide and transmit reliable information if and when a wildfire starts. The first objective requires aggressive energy savings and ambient energy harvesting, while the second requires continuous operation of a range of sensors. To the best of our knowledge, this paper presents the first self-sustained cyber-physical system that dynamically co-optimizes the wildfire detection accuracy and active time of sensors. The proposed approach employs reinforcement learning to train a policy that controls the sensor operations as a function of the environment (i.e., current sensor readings), harvested energy, and battery level. The proposed cyber-physical system is evaluated extensively using real-life temperature, wind, and solar energy harvesting datasets and an open-source wildfire simulator. In long-term (5 years) evaluations, the proposed framework achieves 89% uptime, which is 46% higher than a carefully tuned heuristic approach. At the same time, it averages a 2-minute initial response time, which is at least 2.5× faster than the same heuristic approach. Furthermore, the policy network consumes 0.6 mJ per day on the TI CC2652R microcontroller using TensorFlow Lite for Micro, which is negligible compared to the daily sensor suite energy consumption.

CCS Concepts: • **Computing methodologies → Reinforcement learning**; • **Computer systems organization → Embedded and cyber-physical systems;**

Additional Key Words and Phrases: Wildfire monitoring, self-sustainable, energy harvesting, edge device, IoT, resource management, decision making, sensing

ACM Reference format:
Yigit Tuncel, Toygun Basaklar, Dina Carpenter-Graffy, and Umit Ogras. 2023. A Self-Sustained CPS Design for Reliable Wildfire Monitoring. *ACM Trans. Embedd. Comput. Syst.* 22, 5s, Article 135 (September 2023), 23 pages.
https://doi.org/10.1145/3608100

This article appears as part of the ESWEEK-TECS special issue and was presented in the International Conference on Hardware/Software Codesign and System Synthesis, (CODES+ISSS), 2023.
This work was supported in part by the NSF Career-2114499 and NSF Ascent-2132904 awards.
Authors' addresses: Y. Tuncel, T. Basaklar, and U. Ogras, University of Wisconsin - Madison, 1415 Engineering Dr, Madison, Wisconsin, USA, 53706; emails: {tuncel, basaklar, uogras}@wisc.edu; D. Carpenter-Graffy, Pacific Northwest National Laboratory, 902 Battelle Blvd, Richland, Washington, USA, 99354; email: dina.carpenter-graffy@pnnl.gov.

1539-9087/2023/09-ART135 $15.00
https://doi.org/10.1145/3608100

1 INTRODUCTION

Wildfires occur naturally in many ecosystems worldwide, with increasing frequency and severity in recent years, as depicted in Figure 1. According to data from the European Forest Fire Information System, wildfires in 2022 have burnt 3× more area than the historical average, as shown in Figure 1(a) [43]. Similarly, The United States National Interagency Fire Center reports that the total burned area has doubled in the last 20 years, as shown in Figure 1(b) [32]. The devastating wildfires have far-reaching ecological, social, and economic impacts, such as the loss of human lives, homes, property, and infrastructure, as well as adverse effects on animal health and air quality.

The increasing trend in wildfire occurrences is due to various human-caused factors, including logging, agriculture, and urbanization, which alter the landscape, and external ignition sources, such as cigarettes, campfires, and electric power infrastructure. Among these, electric power infrastructure poses a particular risk because it can cause highly deadly wildfires and be disabled by them. For example, the 2018 Camp Fire in California caused 85 fatalities, over $16 billion in damage, and led the power utility to bankruptcy [45]. Thus, monitoring areas near power grids is essential to developing effective strategies for preventing or promptly detecting grid-caused wildfires.

Current preventive techniques are based on various risk indexes that represent a rough likelihood of fire occurrences, such as the Wildland Fire Potential Index (WFPI), the National Fire Danger Rating System (NFDRS), and others [16, 17, 23]. These indexes use meteorologic data from weather stations hundreds of miles away from the actual location of interest (i.e., the electric grid in this case). Hence, the temporal and spatial resolutions of these index maps are poor. Similarly, the existing satellite and airborne wildfire detection systems collect intermittent data. Moreover, they are unaware of local ambient factors such as high wind speeds or low humidity, which results in poor awareness and long detection times [31]. Motivated by these shortcomings, recent research has focused on a new class of data-informed, energy-harvesting cyber-physical systems (CPS), i.e., sensor suites, to improve wildfire prevention and detection [35]. The deployed sensor suites must comprise multiple sensors for accuracy and be self-sustainable due to low maintenance requirements. However, accommodating many sensors with varying levels of power consumption as well as the dynamic nature of the environment make achieving self-sustainability difficult. The current practice to alleviate this problem is over-designing the system (i.e., using larger than necessary batteries and energy harvesters) and employing heuristics to control the total energy consumption by duty-cycling the sensors. However, these solutions increase the cost of the system and are far from optimal. Therefore, there is a strong need for theoretically grounded and practical cyber-physical systems that maximize monitoring accuracy while ensuring self-sustainability.

This paper presents a CPS framework that achieves self-sustained operation throughout the system lifetime (years). We consider sensor suites that integrate multiple sensors, such as temperature and particle, with different accuracy and power consumption. The sensor suites replenish the battery by harvesting solar energy to avoid manual maintenance. On the one hand, the sensors must stay on with a high sampling rate to maximize the detection accuracy and response time when a wildfire approaches. On the other hand, the sensor suite must preserve energy for arbitrarily long durations (five years in our evaluations) since wildfires can happen anytime, and the battery cannot be replaced easily due to remote locations. One can formulate this challenge as a constrained optimization problem and solve it with modern solvers like CPLEX [10] and Gurobi [21]. However, these approaches are impractical for online deployment since they require long execution times [9] and significant computational power beyond what simple sensor suites can provide. As a powerful and practical alternative, we propose a novel RL framework to co-optimize the sensor energy consumption and wildfire monitoring accuracy. The primary advantages of the proposed CPS framework are:

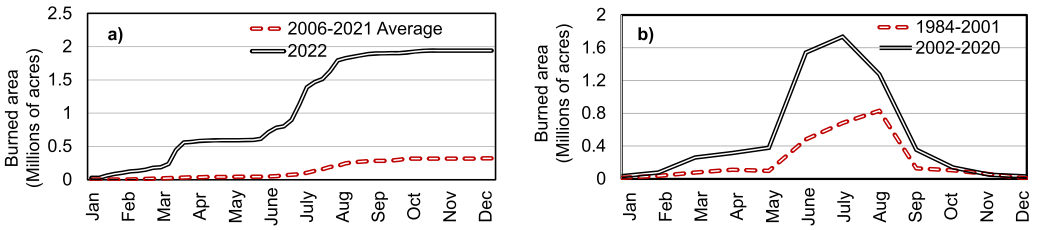

Fig. 1. (a) Cumulative burnt area in Europe in 2022 compared to the 2006-2021 average. (b) Average monthly burnt area in the US between 2002-2020 compared to between 1984-2001.

(1) *Efficient online deployment* – After the training stage is completed, the execution time of the policy network is negligible, facilitating online deployment.
(2) *Model-free* – RL implicitly learns the energy harvesting and consumption patterns, which makes it a *prediction-free* approach, i.e., it does not rely on forecasts of the future.

The proposed framework trains an RL agent that controls the sensor sampling rates such that the accuracy of sensor readings is maximized without depleting the battery and forcing the sensor suite to enter a sleep state. As a result, it achieves *reliable wildfire monitoring with a self-sustained operation* for as long as five years, as demonstrated in Section 5. To train the agent, we develop a detailed environment for wildfire monitoring CPS that models the energy consumption due to sampling, data processing, communication, and standby power, as well as the sensing accuracy. Evaluations under different wildfire scenarios demonstrate that the trained RL agent successfully controls the sensors under dynamic, uncertain conditions. Specifically, it achieves 89% system uptime in 5-year long simulations, 46% higher than a carefully tuned heuristic approach. Similarly, the RL agent averages a 2-minute initial response time, which is at least 2.5× faster than the heuristic approach. In addition, when deployed on the TI CC2652R microcontroller, the agent consumes only 0.6 mJ in a day, which is negligible compared to the energy consumption of the sensors. *In summary, our major contributions are:*

— A novel RL based CPS design for self-sustained, reliable and continuous monitoring of wildfires,
— An open-sourced, detailed RL environment that can simulate wildfires and model the energy consumption of a sensor suite, and the behavior of the sensor readings, which can be used *beyond* training RL agents,
— Detailed experimental evaluation of the proposed CPS in terms of long-term performance and energy consumption on the TI CC2652R microcontroller.

The rest of this paper is organized as follows: Section 2 reviews the prior work. Section 3 overviews the problem and introduces the background concepts and mathematical formulations. Section 4 presents the proposed environment and the RL framework. Then, Section 5 presents the experimental evaluations of the trained RL agent. Finally, Section 6 concludes the paper.

2 RELATED WORK

There are three current mechanisms for wildfire monitoring: Unmanned aerial vehicles (UAVs), satellites, and sensor networks. UAVs commonly refer to vehicles or systems that are remotely operated and travel by flight. They have become an increasingly popular tool for detecting wildfires due to their ability to cover large areas quickly and provide real-time data to ground crews [37, 46]. Despite their advantages, UAVs are unsuitable for continuous, uninterrupted monitoring in

wildfire-prone areas owing to their limited flight times [31]. Geostationary satellites address the lack of continuous availability of UAVs, as they can monitor a specific area continuously for fires [31]. With more accessible access to satellites through projects such as the CubeSat [6, 20], recent research has focused on wildfire detection algorithms using the low-resolution images taken by the satellites. Furthermore, since sending images to the ground stations may take hours, system designs for processing the images at the edge gained traction [3]. Despite these efforts, the limited spatial and temporal nature of the data makes timely detection challenging. For example, detecting young fires smaller than a pixel is extremely difficult [31]. In addition, dense clouds, smoke, and other atmospheric effects may lead to false alarms or missed detections.

Wireless sensor networks (WSNs) address the limited spatial and temporal coverage of airborne approaches. They consist of many distributed sensor suites that accommodate various sensors like humidity, temperature, and gas particles to monitor the covered area for wildfire. As a result of their promising outlook, recent research efforts focused either on improving the detection probability through the use of machine learning [11, 13], improving the bandwidth of the link between suites [47] or network-level optimizations that minimize network-wide communication energy overhead [1, 24, 38]. However, the benefits of WSNs quickly diminish if the sensor suites run out of battery, as maintenance and replacement require significant manual labor and logistic effort. In addition, many wilderness zones are protected from e-waste [31]. Therefore, *guaranteeing self-sustained operation* to achieve maximum device lifetime (at the suite level) is extremely critical for this technology. Yet, most existing literature does not discuss the performance of the suites for extended periods of time, like 5+ years. Most importantly, the existing literature targets perpetual monitoring rather than stochastic events like wildfire occurrence. In contrast, our formulation aims to address the stochasticity of wildfire occurrence and improve performance at the node level. We provide 1-year and 5-year simulations that depict how the performance of the RL agent is affected over time.

Energy harvesting and management approaches for self-sustained operation gained attention thanks to decreasing energy consumption requirements of sensing, processing, and communication hardware. One of the early studies in this field [25] proposes a linear programming approach to determine the duty cycle of the IoT device for self-sustained operation. This approach relies on the predictions of harvested energy to determine the future on-time of the device. A more recent approach proposes a rollout-based optimization to determine the energy consumption budget for a wearable IoT device [44]. Similarly, this approach first finds initial solutions based on the predicted values for future harvested energy. It then corrects the initial solutions in runtime by considering variations between predicted and actual harvested energy to achieve self-sustainability. *The performance of such approaches depends critically on prediction accuracies.* RL-based approaches eliminate this dependency and enable prediction-free techniques. Various studies have used RL to manage sampling rates and energy consumption in WSNs [2, 5, 15, 18, 28]. For example, RLMan [2] utilizes RL to optimize the packet generation rate in a point-to-point communication system. Similarly, tinyMAN [5] utilizes RL to allocate energy for a wearable IoT device for self-sustainability. However, the environment dynamics, the RL algorithm, and the performance metrics they optimize do not translate to wildfire monitoring. For example, they focus on optimizing either the data processing energy or the communication energy only (i.e., they do not consider the sensor sampling energy). In addition, these approaches lack thorough energy consumption and harvesting models. They do not model the sensor behavior, which is crucial for wildfire monitoring. Moreover, the implementations (codebase) of these studies are often not shared, which renders such RL-based approaches impossible to reproduce. In contrast, the current work explains the underlying models in detail and provides an open-source codebase.

To the best of our knowledge, there are no prior sensor accuracy-energy co-optimization techniques for wildfire monitoring. The self-sustainable design choices have been either over-designing the energy harvester and the battery [41] or using very low-power, low-cost sensors [35]. Over-designing increases the cost and makes deployment difficult, and it still does not guarantee self-sustainability. Conversely, using low-cost sensors may compromise the reliability of a long-term solution in terms of hardware lifetime and measurement accuracy. Therefore, a self-sustained wildfire monitoring CPS design is an essential contribution to the wildfire monitoring literature. To this end, we propose an RL-based framework to co-optimize the sensor energy consumption and wildfire monitoring accuracy for the first time in the literature.

3 PROBLEM OVERVIEW AND PRELIMINARIES

3.1 Objective of the Proposed RL-based CPS Design Framework

The proposed CPS design considers an energy-harvesting sensor suite with sensing, processing, and communication capabilities, an energy-harvesting source, and a rechargeable battery, as depicted in Figure 2. The sensor suites are attached to existing electric towers in rural areas to monitor the environment using the various sensors. We assume the battery energy is complemented by harvested energy since the power utility companies do not allow any direct physical interface to the wires due to the safety, liability concerns, and cost of power conversion equipment. This work employs solar energy harvesting because it is a robust and mature technology [44], but our formulation does not exclude any other energy harvesting modality, such as electromagnetic coupling with the power distribution lines [36]. As there are no trees near the towers, shade from trees is not a concern. When the sensor suites detect a potential wildfire, they communicate the preprocessed data to a central node where decision-making (e.g., risk index calculation and intervention) occurs. This work focuses on producing accurate sensor readings at the edge regardless of the onset time of a wildfire. Therefore, the suite must obtain accurate sensor readings for robust decision-making while avoiding running out of battery. This is challenging because the response time and accuracy of the sensor readings improve by using higher sampling rates. In turn, energy consumption increases with sampling rates due to higher sensing, data processing, and communication energy. Consequently, the energy in the battery drains faster, and achieving self-sustainability becomes a challenging objective. Our framework poses this objective as an optimization problem and solves it using RL:

Objective: Maximize the accuracy of sensor readings by increasing the sampling rates while keeping the device operational at all times (i.e., the battery is not depleted).

The rest of this section presents the mathematical background that allows us to describe the dynamics of the sensor node as an RL environment. Then, it formulates the optimization problem and provides background about the Twin Delayed Deterministic Deep Policy Gradient (TD3) algorithm used in this work.

3.2 Sensor and Energy Models

3.2.1 Sensor Noise and Accuracy. Each of the sensor suites in Figure 2 has multiple sensors. The sensors in different sensor suites will read different values due to their relative locations with respect to the wildfire. For example, in this case, the temperature sensors on suite_1 and suite_3 will read higher temperatures than the sensor on suite_2. Similarly, the sensor readings on suite_1 and suite_3 will be more noisy (higher variance) due to a nearby heat source. There are no existing studies in the literature that model how sensor readings behave during a wildfire. Therefore, we used the Fire Dynamics Simulator (FDS) [29], a sophisticated computational fluid dynamics model of fire-driven fluid flow, to simulate the smoke particles and heat during the progression of a

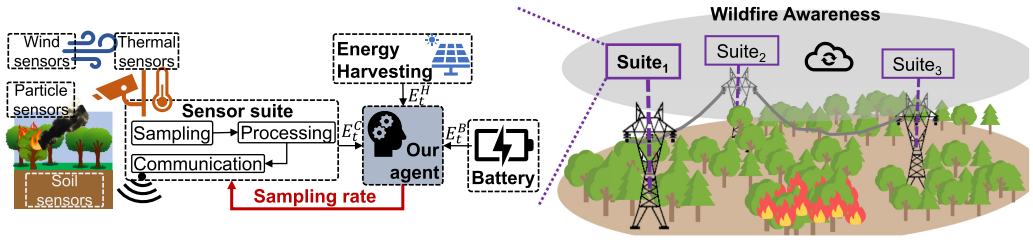

Fig. 2. Overview of the proposed CPS design: An energy harvesting sensor suite with sensing, processing and communication capabilities. Our RL-based framework trains an agent to control the sampling rates of the sensors on the sensor suite.

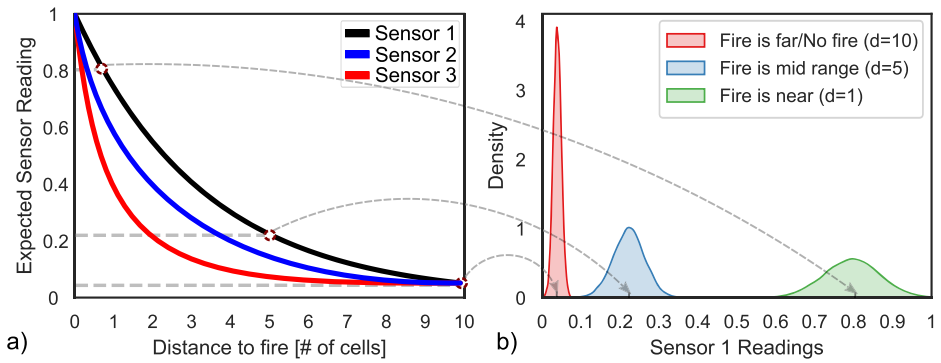

Fig. 3. An overview of the sensor model. (a) The expected sensor readings are modeled with exponentially decaying readings as a function of the distance to fire. (b) The actual sensor readings are drawn from a distribution that is centered around the expected sensor reading.

wildfire. Our experiments with representative wildfire scenarios reveal that the sensor readings are exponentially correlated with the distance from the fire. Using the measurements from detailed FDS simulations, we use a normalized model for the expected sensor readings:

$$\mathbb{E}[S(d)] = Me^{-kd} + N \tag{1}$$

where M is the maximum of the sensor reading range, k is a constant that controls the decay rate of sensor readings with distance (i.e., the sensor's far-sightedness), N is the nominal reading value for the sensor, and d is the distance between the sensor and the firefront. For example, Figure 3(a) shows three sensors with different k values. Sensor 1 starts to deviate from the nominal at greater distances to fire than sensors 2 and 3, i.e., sensor 1 is more sensitive to fire than sensors 2 and 3.

Given the expected sensor reading model in Equation (1), the actual sensor readings are drawn from a normal distribution with $\mu = \mathbb{E}[S(d)]$ and $\sigma = 1/d$.

$$S_i(d) \sim \mathcal{N}\left(\mathbb{E}[S_i(d)], \frac{1}{d}\right), \quad S_i \in \mathbb{R}^{a_i} \tag{2}$$

Here, the standard deviation of the distribution is inversely proportional to the distance to fire. This relationship models the increase in noisy readings when the fire is nearer. For example, sensor 1 is expected to read 0.22 at a distance of 5 units from the fire, as shown in Figure 3(a). This constitutes the mean of the distribution of the sensor readings in Figure 3(b). The larger the distance to fire, the smaller the deviation and the narrower the distribution gets, and vice versa as

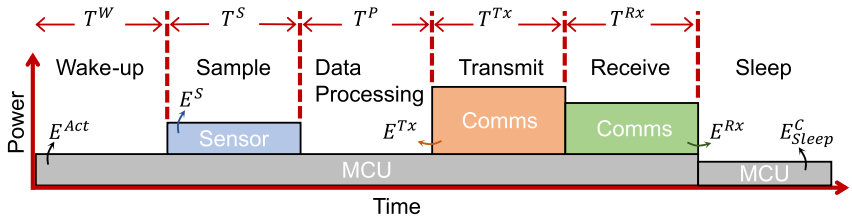

Fig. 4. The energy consumption model. Each sampling event repeats these steps. *Comms: Communications.

shown in Figure 3(b). Finally, a_i samples are drawn from the distribution (Equation (2)), and their sample mean constitutes the actual sensor reading $\hat{S}$:

$$\hat{S}_i(a_i, d) = \frac{1}{a_i} \sum_{n=0}^{a_i} S_i(d) \tag{3}$$

where i denotes the sensor index and a_i is the sampling rate per hour for sensor i. Thus, higher sampling rate improves sensing accuracy by sampling more samples from the distribution. Using higher sampling rates is more critical in the presence of a wildfire to maximize sensing accuracy.

3.2.2 Energy Consumption of the Suite. We model the energy consumption of the sensor suite as shown in Figure 4. This figure shows six main stages for the operation of the sensor suite: (i) The suite wakes up from sleep (T^W), (ii) the sensor i measures a sample (T^S), (iii) the microcontroller (MCU) processes the collected data (T^P), (iv) data is transmitted to neighboring sensor suites (T^{Tx}), (v) an acknowledgment is expected before going back to sleep (T^{Rx}), (vi) the suite goes to sleep mode until the next wake up. Since each sensor on the suite can be assigned different sampling rates, *this cycle is executed on a per-sensor basis.*

The total active time for a sensor i is

$$T_i^{Act} = T^W + T_i^S + T_i^P + T_i^{Tx} + T^{Rx} \tag{4}$$

Since different sensors use different techniques for measurements, the sampling time T^S is a function of sensor (i). For example, a temperature sensor typically has a shorter T^S than an image sensor. Similarly, different sensors use different number of bits to represent a sample. For example, a temperature sensor uses 4 bytes per sample, whereas an image sensor may use $640 \times 480 \times 3$ bytes per sample. As a result, T^P and T^{Tx} are also functions of individual sensors.

The energy consumption for the node due to sensor i has the following four main components:

$$\begin{aligned} E_i^{Act} &= T_i^{Act} P_{active}^{MCU} & E_i^S &= T_i^S P_i^S \\ E_i^{Tx} &= T_i^{Tx} P^{Tx} & E^{Rx} &= T^{Rx} P^{Rx} \end{aligned} \tag{5}$$

where P_{active}^{MCU} is the power consumption of the microcontroller, P_i^S is the power consumption of the sensor i, and P^{Tx} and P^{Rx} are the power consumption owing to the transmit and receive phases of the communication protocol, as listed in Table 1. Using the above, *the hourly energy consumption* of the node due to a sensor i is:

$$E^C(a_i) = \left(E_i^{Act} + E_i^S + E_i^{Tx} + E^{Rx}\right) a_i \tag{6}$$

Consequently, the *the hourly energy consumption* of the node due to sleep is:

$$E_{Sleep}^C = \left(3600 - \sum_i T_i^{Act} a_i\right) P_{idle}^{MCU} \tag{7}$$

Table 1. Symbols used in Section 3.2 and their Description

Symbol	Description	Symbol	Description
d	Distance from the fire	T^W	Wake-up time for the node
M	Maximum sensor reading	T_i^S	Sensing time for sensor i
k	Sensor distance constant	T_i^P	MCU data processing time for sensor i
N	Nominal sensor reading	T_i^{Tx}	Comms transmit time for sensor i
n	Number of sensors on the node	T^{Rx}	Comms ack receive time
i	Sensor index $i \in [1, 2, \ldots n]$	T_i^{Act}	Total active time for sensor i
a_i	Samples/hour for sensor i	η	Efficiency of the energy harvester
$\mathbf{a}$	Vector of a_is: $[a_1, a_2, \ldots a_n]$	t	Time step (interval length)
E_t^B	Battery level at the start of interval t	T	Episode length
E_t^H	Harvested energy in interval t	P_{active}^{MCU}	Power consumption of the MCU
E_{min}^B	Minimum battery level	P_{idle}^{MCU}	Idle power consumption of the MCU
E_{max}^B	Maximum battery level	P_i^S	Power consumption of the sensor i
E_T^B	Battery level at the end of episode	P^{Tx}	Comms transmit power consumption
E_i^S	Energy consumption per sample of sensor i	P^{Rx}	Comms receive power consumption
E_i^{Tx}	Energy consumption per sample for data transmit due to sensor i		
E^{Rx}	Energy consumption per sample for ack receive		
$E^C(a_i)$	Hourly energy consumption of the node due to sensor i		
E_{Sleep}^C	Hourly energy consumption of the node due to sleeping		
$E^C(\mathbf{a})$	Total hourly energy consumption of the node		
E_i^{Act}	Energy consumption per sample of the microcontroller due to sensor i		
$\mathbb{E}[S_i(d)]$	Expected sensor reading for sensor i at distance d from the fire		
$S_i(d)$	Sensor readings sampled from $\mathcal{N}(\mathbb{E}[S_i(d)], \frac{1}{d})$		
$\hat{S}_i(a_i, d)$	Actual sensor reading for Sensor i set to a_i samples/hr at distance d from the fire		

where P_{idle}^{MCU} is the power consumption of the microcontroller during sleep mode. Finally, *the hourly energy consumption of the node* is given by:

$$E^C(\mathbf{a}) = \sum_i E^C(a_i) + E_{Sleep}^C \tag{8}$$

where $\mathbf{a} \in \mathbb{N}^n$ is the vector of sampling rates a_i assigned to sensors $i \in [1, 2, \ldots n]$.

3.2.3 Energy Harvesting and Battery Dynamics. The energy harvesting source converts the energy in the surrounding environment into usable electrical energy, which is often used to extend the battery lifetime. There are many energy harvesting modalities, including light, motion, heat, and electromagnetic coupling based harvesters. The proposed framework is oblivious to the type (e.g., light) of energy harvester, and it does not rely on any predictions of the harvested energy. Our evaluations use solar energy harvesting data, as detailed in Section 4.1.2.

Our framework uses 150-hour long episodes (T = 150 hours) divided into one-hour time steps (t). We denote the battery energy level at the start of time step t as E_t^B, the harvested and the consumed energy in time step t as E_t^H, E_t^C respectively. Using the above, the following equation governs the evolution of the battery energy:

$$E_{t+1}^B = E_t^B + \eta E_t^H - E_t^C(\mathbf{a}),\ t \in T \tag{9}$$

where η accounts for the inefficiencies at the energy harvesting and battery charging interfaces. Furthermore, there are two physical constraints on the battery energy level, such that the battery level is bounded between empty (i.e., $E^B_{min} = 0\%$) and full (i.e. $E^B_{max} = 100\%$):

$$E^B_{min} \leq E^B_t \leq E^B_{max}, \quad t \in T \tag{10}$$

3.2.4 Problem Formulation. Using equations 1, 3, 8–10, we formulate the optimization problem as follows:

$$\begin{aligned} &\textit{minimize} \quad \sum_i |\hat{S}_i(a_i, d) - \mathbb{E}[S_i(d)]| \\ &\textit{subject to} \quad E^B_{t+1} = E^B_t + \eta E^H_t - E^C_t(\mathbf{a}), \quad t \in T \\ &\qquad\qquad E^B_t \geq E^B_{min} \quad E^B_t \leq E^B_{max} \quad E^B_T \geq E^B_0 \end{aligned} \tag{11}$$

where i denotes the different sensors. Thus, the optimal solution minimizes the absolute error in sensor readings (i.e., the objective) while satisfying battery constraints by optimizing the sampling rates of the sensors.

3.3 Twin Delayed Deep Deterministic Policy Gradient (TD3) Algorithm

The objective of RL is to train an agent to make optimal decisions within an environment by acquiring a policy that maximizes the overall reward over a period of time. Since deterministic policy gradient approaches show superior performance compared to SOTA RL approaches and are less susceptible to hyperparameter tuning, we employ Twin Delayed Deep Deterministic Policy Gradient [19] (TD3) algorithm in our work, which is an extension of the deep deterministic policy gradient (DDPG) [27] approach. DDPG comprises an actor, a policy network that leverages the policy gradient method to obtain the optimal policy, and a critic, a deep Q network (DQN) responsible for evaluating the action produced by the actor. The actor is trained to learn the correlation between state and action, while the critic is trained to learn the connection between state-action pairs and anticipated cumulative returns (Q-values). However, one drawback of DDPG is that the critic tends to overestimate the target Q-value, which can result in issues with policy stability and convergence to local optima due to approximation errors in the Q-value utilized to enhance the policy [19]. To mitigate the overestimation error, TD3 [19] is proposed with three significant improvements on DDPG: (i) clipped double Q-learning, (ii) target policy smoothing, and (iii) delayed policy updates.

Clipped double Q-learning: TD3 uses two separate critic networks instead of one. It uses the smaller of the two Q-values to form the targets in Bellman's optimality equation and updates both critics using the following loss:

$$L_{critic}(\theta_i) = \mathbb{E}_{(\mathcal{S}_t, \mathbf{a}_t, r_t, \mathcal{S}_{t+1}) \sim \mathcal{D}}[(y - Q(\mathcal{S}_t, \mathbf{a}_t; \theta_i))^2] \tag{12}$$

$$y = r + \gamma \arg_Q \min_{i=1,2} Q(\mathcal{S}_{t+1}, \tilde{\mathbf{a}}; \theta'_i) \tag{13}$$

where $\mathcal{D}$ is the experience replay buffer that stores transitions $(\mathcal{S}_t, \mathbf{a}_t, r_t, \mathcal{S}_{t+1})$ for every time step in the environment and y is the target value.

Target policy smoothing: The action $\tilde{\mathbf{a}}$ while computing target values is generated by the target actor-network. However, the DDPG method is prone to generating target values with high variance, even for similar actions. This is due to the deterministic policies tend to overfit to the sharp peaks in the value estimate. To mitigate this issue, instead of using the action given directly by the target actor-network, we add some noise ϵ to the action as follows:

$$\tilde{\mathbf{a}} = \pi(\mathcal{S}_{t+1}, \phi') + \epsilon : \epsilon \sim clip(\mathcal{N}(0, \sigma), -c, c) \tag{14}$$

The target actor-network policy for state $\mathcal{S}_{t+1}$ is represented by $\pi(\mathcal{S}_{t+1}, \phi')$, and the range from $-c$ to $+c$ indicates that any noise added is clipped to ensure that target action remains in proximity to the actual action. This regularization technique helps to decrease the variance in the target values.

Delayed policy updates: In deterministic policy gradient approaches, the parameters of the actor-network are updated by maximizing the Q-values obtained using the actions generated by the actor-network as follows:

$$L_{actor}(\phi) = \mathbb{E}_{(\mathcal{S}_t, \mathbf{a}_t, r_t, \mathcal{S}_{t+1}) \sim \mathcal{D}}[Q(\mathcal{S}_t, \mathbf{a}_t; \theta_1)|_{\mathbf{a}_t = \pi(\mathcal{S}_t; \phi)}] \tag{15}$$

Adopting this update rule can lead to divergence during the training of the agent, particularly when a suboptimal policy is overestimated. Consequently, the agent may update on states with high error, leading to instability and a shift towards inferior policies. TD3 updates the actor-network less frequently than the value network to address this challenge. This approach of infrequent policy updates produces value estimates with lower variance and therefore promotes the generation of better policies.

4 PROPOSED RL-BASED CPS DESIGN FRAMEWORK

RL methods have led to remarkable advancements in diverse fields, such as autonomous driving, robotics, and gaming. The remarkable strides in RL's application in these domains are largely attributed to the widespread adoption of open-source ML frameworks. For example, Google DeepMind's StreetLearn [30] and Microsoft Research's AirSim [39] environments have facilitated several innovative approaches to autonomous driving. Thus, designing a representative environment for wildfire monitoring is of utmost importance as it enables the application of RL techniques for wildfire detection. As such, a significant contribution of this work is an RL environment for wildfire propagation and sensor suite dynamics. This environment enables us to apply the proposed RL-based framework for reliable wildfire monitoring. We also plan to release it to the public to catalyze research in this area and enable the broader RL community to use our environment as a benchmark. This section first presents the design of this RL environment. Then, it describes the proposed TD3 framework that leverages this environment.

4.1 RL Environment Design for Wildfire Monitoring CPS

4.1.1 Wildfire Simulation. Detailed fire simulators, such as FDS, are extremely slow due to the underlying differential equations. For example, executing the small-scale test fires in Section 3.2.1 takes several hours. Therefore, we employ FDS only for characterizing the sensor readings and revert to Cell2fire [34], another open-source wildfire simulator, for large-scale wildfire simulations. Cell2fire employs cellular-automata networks to model wildfires at a higher level than FDS. It uses real-world temperatures, wind measurements, and terrains, as shown in Figure 5. The Cell2fire simulator has been validated using several real-world fires. Among these, we use the Dogrib Creek instance for the underlying elevation and vegetation map. This instance consists of a rectangular grid of 357 × 223 cells, where each cell is a 100 m × 100 m square (the total area is 35.7 km × 22.3 km). Each cell has an elevation and vegetation value that determines the fire's rate of spread within that cell. The simulator combines this information with ambient temperature and wind data to simulate the direction and the rate of spread of the fire across the grid given an ignition point. Therefore, to generate a burn trace, we choose a random 150-hour segment from an hourly temperature and wind dataset from the BANFF CS Canadian weather station. In addition, we randomly pick an ignition point from the set of 6 points shown in Figure 5(a). By doing this, we model wildfires approaching from different directions with upwind and downwind conditions. In total,

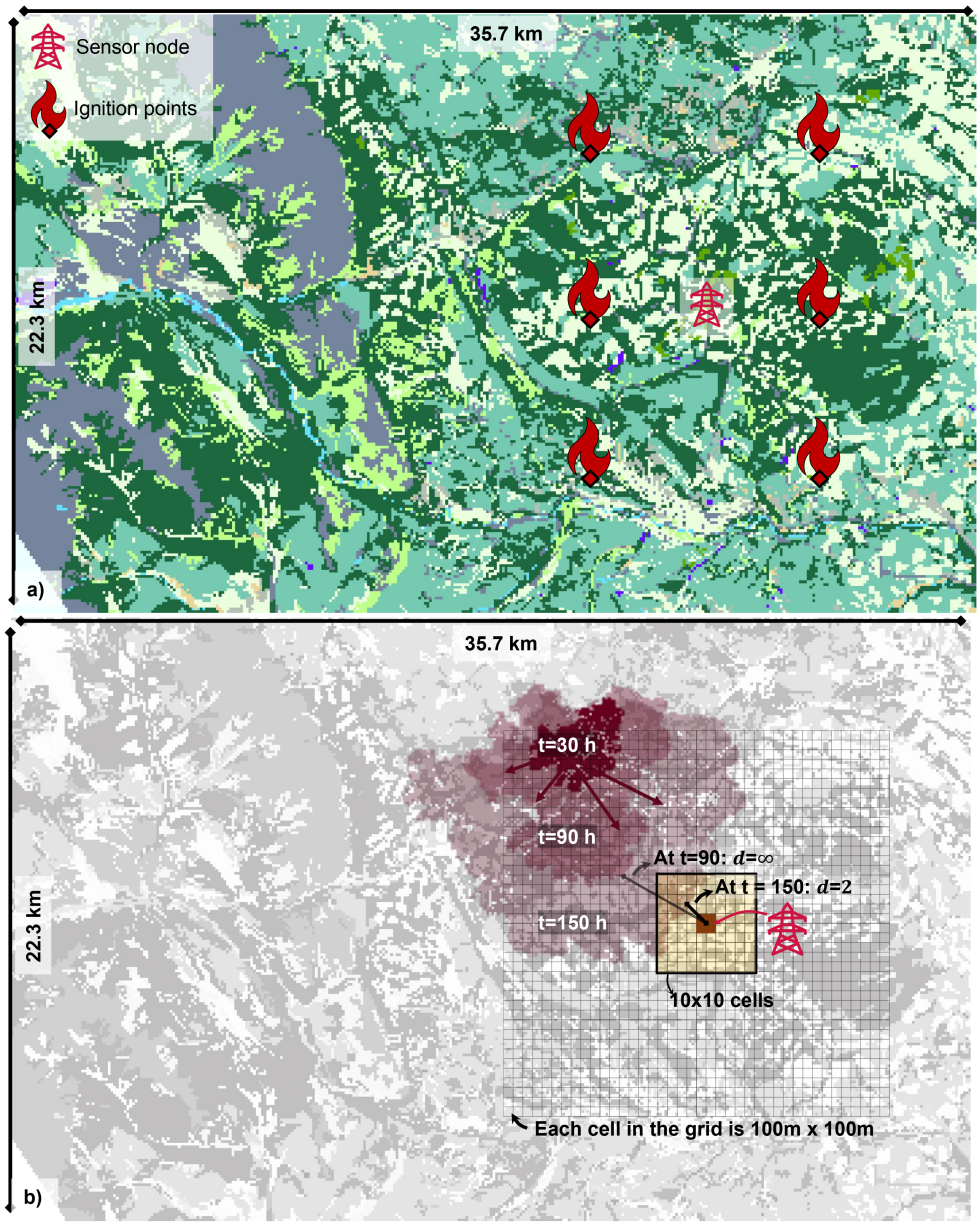

Fig. 5. The Dogrib Creek, CA map illustrated using the Cell2fire simulator. (a) Our sensor suite placement and six different ignition points. (b) An example wildfire trace that shows the spread of the fire in three time instants. The wildfire gets within 10 cells of the sensor suite after around 100 hours.

we generate 6000 different 150-hour-long wildfire burn traces. An example burn trace that shows the wildfire at three time instants during its evolution is shown in Figure 5(b). Finally, we stress that our framework is not limited to the Dogrib Creek map or Cell2fire.

Despite the relatively faster speed of Cell2fire, embedding a wildfire simulator inside an RL environment is not feasible for training since RL algorithms require thousands of episodes to converge,

and running a new simulation at each episode slows down the training process significantly. To address this challenge, we decouple the fire simulations from the RL environment. Thus, we use Cell2fire to generate various wildfire scenarios in a batch and store them as trace files for use during training, as outlined above.

4.1.2 Energy Harvesting Dataset. Our framework can use any available energy harvesting data. In this work, we assume the sensor suite has a solar energy harvester. To generate a dataset for solar energy harvesting, we use pvlib, a well established tool for simulating the performance of photovoltaic (PV) energy systems developed at Sandia labs [22]. Using this tool, we combine real hourly solar irradiance data from 2015 with a realistic electrical model [26] of a 0.1m^2 PV-Cell (Uni-Solar US-5). As a result, we obtain a year-long hourly energy harvesting data with seasonal changes (8760 hours in total).

Table 2. Parameter values

	M	k	N	T^W	T^S	T^P	T^{Tx}	T^{Rx}	P^S
S_1	1	2	0	0	5 s	0	6.5 ms	0	4 mW
S_2	1	0.7	0	0	4 s	0	6.5 ms	0	250 mW
S_3	1	0.3	0	0	2s	0	13.0 ms	0	11 mW

	P^{Tx}	P^{Rx}	P^{MCU}_{active}	P^{MCU}_{idle}
MCU/LoRa	92.4 mW	0	12.4 mW	1 μW

	T	α	η	T^{Steps}	$^*\lvert\mathcal{D}\rvert$	$^*\lvert\mathcal{B}\rvert$	lr
Others	150 hrs	1	1	10^6	10^6	128	10^{-4}

$^*\lvert.\rvert$ denotes the size. lr denotes the learning rate. S_1: Temperature sensor, S_2: Particle sensor, S_3: Wind sensor. All power values are in mW. P^S and T^S are the power consumption and response time of the sensors that are provided in their datasheets. We ignore the effects of T^W since we assume MCU wakes up instantly from idle state (i.e., we do not use the stand-by mode). We assume transmission is one way and no acknowledgment is expected (i.e., $T^{Rx} = 0$). We assume 4 bytes per sample for S_1 and S_2, and 8 bytes per sample for S_3. Using these, T^{Tx} is calculated (1.63 ms/byte [7]). Finally, we assume processing time T^P for these sensors is negligible owing to their small data sizes.

4.1.3 Choice of Sensors, Microcontroller and Battery. Numerous sensors can be incorporated into a sensor suite, such as temperature, humidity, pressure, wind, particle sensors, thermal camera, and others. In this work, we use three widely adopted meteorological sensors: Temperature sensor [4], particle sensor [12], and wind speed/direction sensor [33]. We choose these three as they represent different levels of energy consumption and different levels of sensitivity to fire (e.g., particle sensor's range is higher than temperature sensor). However, we emphasize that the proposed framework works with any arbitrary type and number of sensors. We use LoRa as the communication protocol [7]. For the microcontroller, we use the STM32WLE5 series, a system-on-chip that accommodates an ARM Cortex M4 CPU and a LoRa radio on the same package [40]. Finally, we use a 10 Ah @ 3.3 V LiPo battery with a 5% annual capacity degradation [44]. The values of the parameters are listed in Table 2.

4.1.4 State Space, Action Space, and Reward Function. The RL environment for wildfire monitoring CPS is designed with generality in mind to enable any reinforcement learning technique.

State Space: The state space is a 14-tuple $\mathcal{S} \subseteq \mathbb{R}^{14}$ that consists of:

— *Current battery energy* ($\frac{E^B_t}{E^B_{max}} \in [0,1]$)*:* The energy level of the battery at the beginning of the current step t divided by the battery capacity.
— *Harvested energy in the previous time step* ($\frac{E^H_{t-1}}{E^B_{max}} \in [0,1]$)*:* Harvested energy during the previous step $t-1$ divided by the battery capacity.
— *Cumulative EH* ($\sum_{\tau=0}^{t-1} E^H_\tau \in \mathbb{R}$)*:* Cumulative harvested energy in the previous time steps.
— *Initial battery energy level* ($\frac{E^B_0}{E^B_{max}} \in [0,1]$)*:* The energy level of the battery at the beginning of the episode ($t = 0$) divided by the battery capacity.

— *Target battery energy level* ($\frac{E_T^B}{E_{max}^B} \in [0,1]$)*:* The desired energy level of the battery at the end of the episode ($t = T$) divided by the battery capacity.
— *Actions in the previous step* ($\mathbf{a}_{t-1} \in [-1,1]^3$)*:* The sampling rates assigned to the three sensors in the previous step $t-1$.
— *Sensor readings in the previous step* ($\hat{\mathbf{S}}_{t-1} \in [0,1]^3$)*:* The sensor readings of the three sensors in the previous step $t-1$.
— *Moving average of sensor readings* ($\frac{1}{5}\sum_{\tau=t-5}^{t-1} \hat{\mathbf{S}}_\tau \in [0,1]^3$)*:* The moving average of sensor readings in the previous 5 time steps ($t-5$ to $t-1$).

Action Space: The actions are the assigned *hourly sensor sampling rates* at every time step ($\mathbf{a}_t \in [-1,1]^3$). We limit the actions in the $[-1,1]$ interval, and map this interval to $[0,60]$ when calculating the energy consumption of the suite (i.e., -1 maps to 0 samples/hr and 1 maps to 60 samples/hr.) The training performance takes a significant hit if we use the full $[1,60]$ range instead of $[-1,1]$.

Reward function: Our objective is to minimize the difference between the expected sensor readings (i.e., golden values) and the actual sensor readings under battery energy constraints. In an RL setting, the objective and the constraints are imposed by the reward function. In this case, two constraints can be imposed on the reward function: *(i)* minimum battery level constraint and *(ii)* target battery level constraint. We do not include the maximum battery level constraint because under-utilizing the energy implicitly decreases the objective. Considering the objective and the constraints on the battery, the reward function becomes:

$$r_t = \begin{cases} -100 & E_t^B \leq E_{min}^B \\ \alpha\left(\frac{E_t^B - E_T^B}{E_{max}^B}\right) - |\hat{S}(\mathbf{a}, d) - \mathbb{E}[S(d)]| & otherwise \end{cases} \tag{16}$$

Here, we impose the minimum battery energy level by punishing the training heavily (−100 reward) if this constraint is violated. Otherwise, we impose the target battery level constraint using the first term ($\frac{E_t^B - E_T^B}{E_{max}^B}$). The objective is considered through the second term $|\hat{S}(\mathbf{a}, d) - \mathbb{E}[S(d)]|$. α is a coefficient to scale the weight of the target battery level constraint, i.e., higher α will push the agent to be more conservative and linger around the target battery level.

4.1.5 Implementation. We develop our environment in Python and register it as an OpenAI Gym [8] environment using the components explained in this section. When starting training after a reset, the environment randomly chooses a 150-hour slice from the energy harvesting dataset, one wildfire trace, and an initial battery energy level, as summarized in Algorithm 1. During testing, the environment takes the energy harvesting data, wildfire trace, and the initial battery level as inputs for repeatable results. Then, the state vector $\mathcal{S}$ is initialized and the done signal, which shows whether the episode ended, is initialized as False.

At each step, the environment gets the data rates $\mathbf{a}$ and the state vector $\mathcal{S}_{t-1}$ from the previous time step as input. Then, it reads the harvested energy for the current time step E_t^H from the energy harvesting data. In addition, it also reads the status of the wildfire from the wildfire trace. The environment uses this trace to calculate the distance d between the sensor suite and the fire, as shown in Figure 5(b). When calculating d, sensors have a maximum range beyond which they cannot sense environmental changes. For example, Figure 5(b) shows a case where the sensors are affected only by a wildfire that is within 10 cells of the sensor suite. If the wildfire is outside of this range, we assign the distance to a very high value (e.g., ∞), so that the sensors read nominal values. If the wildfire is inside the range, we calculate the distance from the closest point of the wildfire to the sensor suite. In the example presented in Figure 5(b), at t = 30h and t = 90h, the wildfire

ALGORITHM 1: Pseudocode for the wildfire RL environment

Reset()
- **Training**: Randomly choose energy harvesting and wildfire traces, and an initial battery level E_0^B
- **Test**: Take the energy harvesting and wildfire traces, and the initial battery energy as inputs
- Reset the state $\mathcal{S}$ and done ← False
- **Output:** $\mathcal{S}$, done

Step()
- **Input: Data rate a (i.e., actions),** $\mathcal{S}_{t-1}$
- Get E_t^H from the energy harvesting trace
- Get distance to fire d from the wildfire trace
- Use d to get expected sensor readings (Equation (1))
- Use **a** and d to get actual sensor readings (Equation (3))
- Use **a** to calculate E_t^C (Equation (8))
- Calculate E_t^B (Equation (9))
- Calculate r_t (Equation (16))
- Calculate $\mathcal{S}_t$
- **if** $E_t^B < E_{min}^B$ ***or*** $d = 0$ ***or*** $t = 150$ **then**
  - done ← True # End of episode
- **else**
  - done ← False
- **Output:** r_t, $\mathcal{S}_t$, done

has not reached within 10 cells of the sensor suite and thus the distance is set to ∞. In contrast, at t = 150h, the closest point of the wildfire to the sensor suite is within 10 cells, so the distance is now calculated as $d = 2$. Next, the environment uses d, **a**, and $\mathcal{S}_{t-1}$ to calculate the expected and actual sensor readings, the energy consumption of the sensor suite, the battery level at the end of the step, and the reward for this step. Finally, it updates the state vector and outputs the reward, the state vector and the done signal. An episode terminates (i.e., $done \leftarrow True$) if the battery is depleted or if the wildfire reaches the sensor suite (i.e., $d = 0$) or if time step reaches 150.

4.2 Proposed TD3 Framework

The proposed framework trains an RL agent by interacting with the wildfire CPS environment. The trained agent is a policy that observes the state vector and yields actions (i.e., data rates **a**) that maximize the Q value. Algorithm 2 summarizes the proposed TD3 framework. First, we initialize an empty replay buffer $\mathcal{D}$ and critic and actor networks with random weights. We use a multi-layer perceptron with three hidden layers and 64 neurons in each layer within these networks. We also initialize the target critic and actor networks with the same parameters. Then, the framework starts training. At each episode, we reset the environment to choose new energy harvesting and wildfire traces, and an initial battery energy level. The agent then interacts with the environment using its actor-network (π_ϕ). The transitions collected ($\mathcal{S}_t, \mathbf{a}_t, r_t, \mathcal{S}_{t+1}, done$) are then stored in the experience replay buffer $\mathcal{D}$. Then, the algorithm samples a minibatch ($\mathcal{B}$) of transitions from $\mathcal{D}$. Using this minibatch of samples, the smoothed target action values are computed according to the noise ϵ and the clipping factor c. The target value y is computed using the target critic-networks and target action values. The actor and critic networks are then updated using the loss functions explained in Section 3.3. The training terminates when number of time steps reaches T^{Steps}. The hyperparameters for our approach are presented in Table 2.

5 EXPERIMENTAL EVALUATIONS

This section presents the experimental evaluation of the proposed framework. It first describes the evaluation scenarios. Then, it introduces a class of heuristic algorithms as a baseline since there are no alternative techniques in the literature. Finally, it presents the evaluation results and execution time, energy overhead and memory footprint measurements of the proposed framework when deployed on the TI CC2652R MCU [42].

5.1 Experimental Setup

5.1.1 Evaluation Scenarios. We evaluate the proposed CPS with five wildfire traces that were generated as part of the 6000 traces as explained in Section 4.1.1. *These five traces are excluded from the training process* explained in Section 4.2. During testing with each of the five traces, we use six different ignition times for long term simulations: (6 months, 1 year, 2 years, 3 years, 4 years and 5 years), leading to a total of 30 combinations. For example, one of the combinations is to use the first trace with the first ignition time, which runs the simulation with *no-fire* conditions for 24 weeks, and the ignition happens in the 25th week (after 6 months). We use a fixed 150-hour long energy harvesting data for each week, such that each approach is evaluated fairly.

ALGORITHM 2: TD3 framework for RL training

1 **Initialize:** Replay buffer $\mathcal{D}$, Critic networks $\mathbf{Q}_{\theta_1}$,$\mathbf{Q}_{\theta_2}$
2 and actor network π_ϕ with parameters θ_1, θ_2, ϕ
3 Target networks $\mathbf{Q}_{\theta_1'} \leftarrow \mathbf{Q}_{\theta_1}$, $\mathbf{Q}_{\theta_2'} \leftarrow \mathbf{Q}_{\theta_2}$, $\pi_{\phi'} \leftarrow \pi_\phi$
4 *done* $\leftarrow$ True
5 **for** $t = 0$: T^{Steps} **do**
6 **if** *done* = *True* **then**
7 # End of an episode, start a new one
8 $\mathcal{S}_t$, done $\leftarrow$ **Reset()**
9 Observe state $\mathcal{S}_t$ and take actions: $\mathbf{a}_t \leftarrow \pi_\phi(\mathcal{S}_t)$
10 r_t, $\mathcal{S}_{t+1}$, done $\leftarrow$ **Step()**
11 Store the transition ($\mathcal{S}_t$,$\mathbf{a}_t$,r_t,$\mathcal{S}_{t+1}$,done) in $\mathcal{D}$.
12 Sample random B transitions from $\mathcal{D}$;
13 $\tilde{\mathbf{a}}_t \leftarrow \pi(\mathcal{S}_{t+1}, \phi') + \epsilon : \epsilon \sim clip(\mathcal{N}(0, \sigma), -c, c)$
14 $y \leftarrow r + \gamma \arg_Q \min_{i=1,2} Q_{\theta_i'}(\mathcal{S}_{t+1}, \tilde{\mathbf{a}}_t)$
15 $Loss_{critic}(\theta_i) = \mathbb{E}\left[\left(y - Q_{\theta_i}(\mathcal{S}_t, \mathbf{a}_t)\right)^2\right]$
16 $Loss_{actor}(\phi) = \mathbb{E}\left[Q_{\theta_1}(\mathcal{S}_t, \mathbf{a}_t)|_{\mathbf{a}_t = \pi(\mathcal{S}_t;\phi)} \right]$
17 Update critic, actor, and target network parameters.

Evaluation metrics: Since a wildfire can occur at any time, the proposed system *should* preserve energy under uncertainty and leverage this energy to minimize sensor reading error when a wildfire danger arises. Using this observation, we extract three quantities from these simulations: (i) The cumulative number of inactive hours where the battery was depleted, (ii) the cumulative sampling error, and (iii) the initial response time to a wildfire, which is defined as the time delay between the emergence of the wildfire and the first measurement after that. If the initial response time is too long, the fire is detected late, leading to *catastrophic* consequences.

Ideal Case: The ideal case should have no inactive hours (i.e., battery is never depleted), the cumulative sampling error is minimized and the initial response time is minimized. However, we emphasize that achieving this is challenging since the CPS does not know when the fire takes place and the stored and harvested energies are limited and varying.

5.1.2 Baseline Heuristic Approach. Since no similar technique exists in the literature, we developed a hierarchical heuristic as the baseline, as summarized in Algorithm 3. This heuristic first sorts the sensors according to their energy consumption in ascending order (line 5). Then, it uses an harvested energy predictor for predicting the battery energy in the next interval (lines 6-7). The energy predictor is obtained by averaging the energy harvested at each hour in a day over the 365 days in the dataset. As a result, we obtain an estimator for each hour in a day. Using this predictor, the heuristic calculates the battery level for the next interval.

If the predicted battery level exceeds 75%, the heuristic allocates 30 samples/hr to the sensor with the lowest energy consumption and 10 samples/hr to the remaining sensors (lines 9-11). Conversely, if the predicted battery level falls below 25%, the sensor with the lowest energy consumption is assigned 15 samples/hr while the other sensors remain in a sleep state (lines 12-14). If the predicted battery level lies between 25% and 75%, the sensors are assigned a constant sampling rate, denoted as c_1 (lines 15-16). Finally, the heuristic checks the sensor readings for a potential wildfire occurrence (lines 18-26). If the reading of the first sensor exceeds the threshold γ_1, the heuristic wakes up the next sensor in line and sets the sampling rates of both active sensors to c_2. Then, if the reading of the second sensor exceeds γ_2, the next sensor wakes up, all three are set to c_3, and so on. This way, the heuristic overrides the low rates set by the low-battery level condition.

We perform an extensive sweep of 285 different combinations of c_1,c_2,c_3 values for this heuristic, with the complete list provided in Table A1 of the Appendix. From these sets, we selected two representative variants to present here, showing distinct levels of "aggressiveness" as depicted in Table 3. The first variant, the balanced heuristic, employs a sampling rate of 15 samples/hr for c_1, ensuring that it maintains a moderate energy consumption level most of the time. It slightly increases the sampling rates for potential fire situations to 21 samples/hr for $c2$ and 27 samples/hr for $c3$. The second variant, referred to as the aggressive heuristic, adopts sampling rates of 33 samples/hr or higher for all scenarios. This approach prioritizes maximum data collection, disregarding potential energy constraints. Furthermore, we implemented a conservative heuristic that consistently employs the minimum sampling rate of 1 sample/hr for all sensors (i.e., c_1, c_2, c_3).

ALGORITHM 3: Pseudocode for the heuristic.

```
1  Input: List of sensors: sensors = [S_1, S_2, ..., S_n],
2  threshold values γ ∈ ℝ^n, sensor readings Ŝ,
3  constant sampling rates c, time of day t and
4  battery level E_t^B
5  sensors ← Sort(sensors)
6  Ê_t^H ← predictEnergy(t)
7  Ê_{t+1}^B ← E_t^B + Ê_t^H
8  # Assign according to predicted battery level
9  if Ê_{t+1}^B > 75% then
10     sensors[1].setSamplingRate(30)
11     sensors[2...n].setSamplingRate(10)
12 else if Ê_{t+1}^B < 25% then
13     sensors[1].setSamplingRate(15)
14     sensors[2...n].setSamplingRate(0)
15 else
16     sensors[1...n].setSamplingRate(c_1)
17 # Then check for wildfire occurrence
18 if Ŝ_1 > γ_1 then
19     sensors[1].setSamplingRate(c_2)
20     sensors[2].setSamplingRate(c_2)
21     if Ŝ_2 > γ_2 then
22         sensors[1].setSamplingRate(c_3)
23         sensors[2].setSamplingRate(c_3)
24         sensors[3].setSamplingRate(c_3)
25         if Ŝ_3 > γ_3 then
26             # Goes on like this ...
```

Table 3. Three Heuristics with Different Energy Consumption aggressiveness

	Conservative	Balanced	Aggressive
γ_1	0.001	0.001	0.001
γ_2	0.3	0.3	0.3
γ_3	2	2	2
c_1	1 S/hr	15 S/hr	33 S/hr
c_2	1 S/hr	21 S/hr	39 S/hr
c_3	1 S/hr	27 S/hr	45 S/hr

Conservative uses Very Little Energy all the Time. Balanced uses less energy most of the time. Aggressive uses high energy most of the time.

5.2 Performance Evaluation

5.2.1 Short-term Analysis: Weekly Behavior. This section evaluates the proposed CPS design based on one-week simulations, focusing on its behavior during a wildfire that approaches the

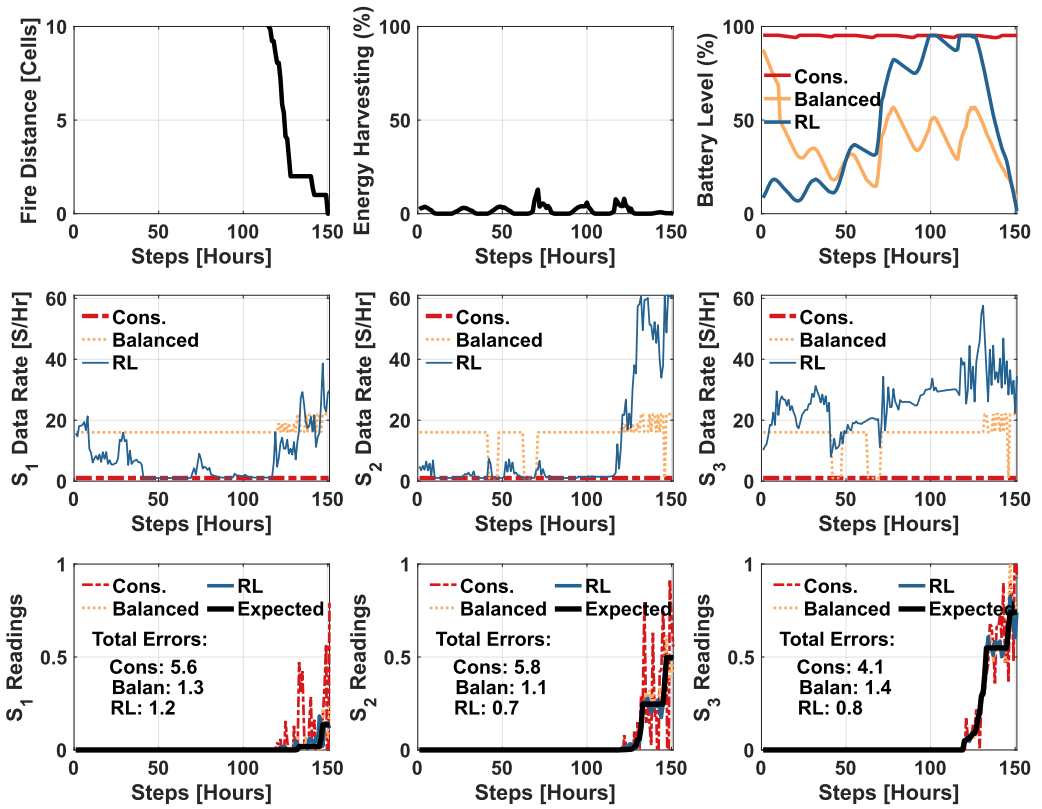

Fig. 6. Weekly behavior of conservative and balanced heuristics and the RL agent. The RL agent achieves 83% and 29% less error than the conservative and balanced heuristics, respectively.

sensor suite on the final day of the week, as depicted in the upper left plot of Figure 6. All algorithms utilize the same wildfire traces and harvested energy, but the RL agent starts with a considerably lower battery energy to showcase its superiority even under more challenging conditions. The sensor data rate plots in the second row of Figure 6 show that the conservative heuristic consistently employs low sampling rates throughout the week. This choice helps preserve the battery energy, as shown in the top right plot. However, it fails to accurately track the environmental conditions after the wildfire arrives, as demonstrated by the plots in the bottom row. The carefully tuned balanced heuristic utilizes the sensors more effectively by activating sensors 2 and 3 only when the battery level exceeds 25%. Additionally, it increases the sampling rate for all sensors after the wildfire arrives (as evident in the second row of plots). Nevertheless, it operates the sensors more than needed, wasting energy. As a result, it depletes the battery towards the end of the week and eventually shuts down. This behavior highlights the inability of static heuristics to guarantee robust operation in dynamic and uncertain conditions. Similarly, the aggressive heuristic quickly drains the battery even before the wildfire reaches the sensor suite. In strong contrast, our RL agent dynamically controls the sensor data rates, co-optimizing monitoring accuracy and battery energy. It actively uses all sensors to guard against environmental changes (the second row), while simultaneously replenishing the battery (top right plot). *Despite starting with a significant disadvantage in terms of initial battery level, the RL agent outperforms the heuristics in terms of average sampling rate and total accumulated errors, as illustrated in the second and third rows. Specifically, the average sampling rate is 1.9× higher, and the total error is 29% smaller than the balanced heuristic.*

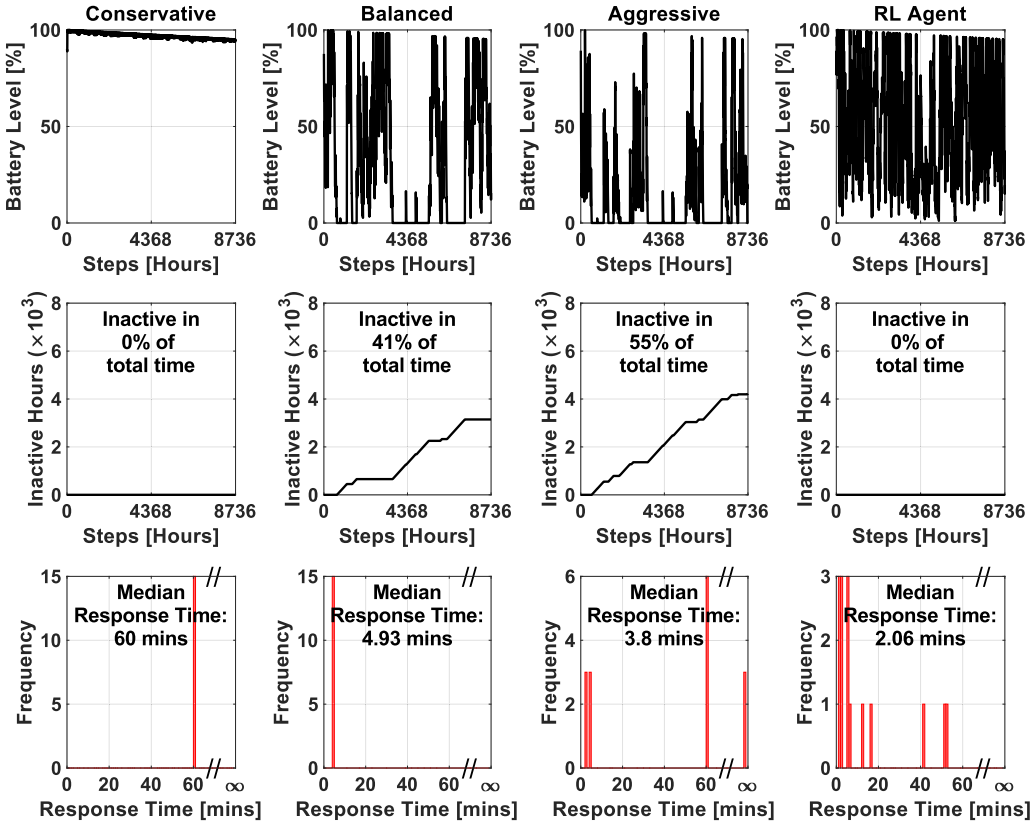

Fig. 7. Comparison of 1-year long simulation results for the conservative, balanced, aggressive heuristics and the trained RL agent.

5.2.2 Long-term Analysis: 1-year and 5-year Long Behavior. In this section, we conduct evaluations of the proposed CPS design over durations of 1 year and 5 years. Similar to the previous section, each algorithm employs identical wildfire traces and harvested energy. However, in this case, all approaches (including the three heuristics and the RL agent) start with an initial battery level of 90%, as we argue that this represents a typical scenario for initial deployment. Figure 7 compares the conservative, balanced, and aggressive heuristics against the RL agent over a 1-year simulation. The first row illustrates the evolution of the battery level for each approach throughout the simulation, based on one wildfire trace. The conservative heuristic maintains the battery level near-maximum capacity, as it consistently employs low sampling rates. Conversely, the balanced and aggressive heuristics often experience battery depletion, as indicated by the fluctuations in their battery levels. This behavior is illustrated in the second row, where each approach's cumulative number of inactive hours (i.e., depleted battery) is plotted. Specifically, the balanced and aggressive heuristics deplete the battery 41% and 55% of the time, respectively. *In strong contrast*, the RL agent *never* depletes the battery due to its capability to recover from low-battery conditions, as demonstrated in the previous section. Finally, the third row illustrates the distribution of the initial response times for each approach across the five traces. The conservative heuristic has a fixed response time of 60 mins. The balanced heuristic has a median response time of 4.93 mins, whereas the aggressive heuristic has 3.8 mins. However, the aggressive heuristic often dies out in

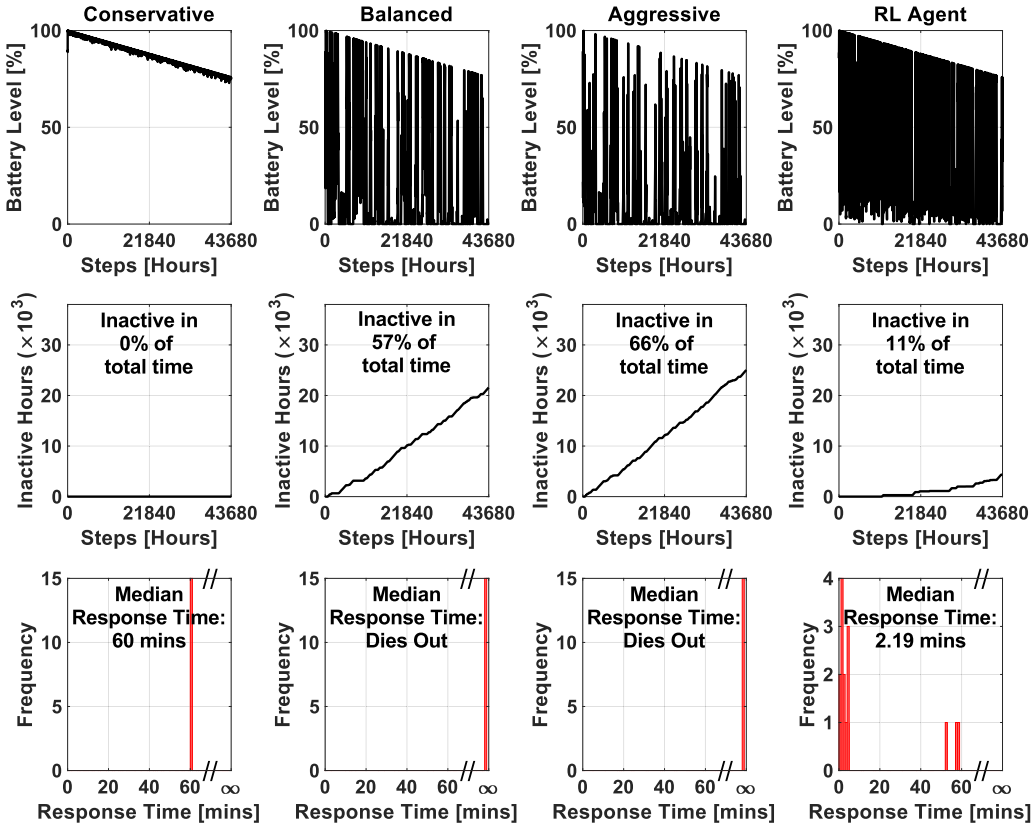

Fig. 8. Comparison of 5-years long simulation results for the conservative, balanced, aggressive heuristics and the trained RL agent.

the final week (after the fire onset), yielding ∞ response time when this happens, as the device shuts down and the sampling rate is set to 0 c2. *In contrast,* the RL agent has a shorter response time compared to heuristic approaches, with a median response time of 2.06 minutes, which is 2.5× faster than the balanced heuristic. Notably, the RL agent achieves this faster response time while *never* depleting the battery.

Figure 8 compares the conservative, balanced, and aggressive heuristics against the RL agent over a 5-year period. As expected, the conservative heuristic exhibits similar behavior to the 1-year results. The performance of the balanced heuristic deteriorates, evidenced by the increased inactive time of 57% (from 41%). Furthermore, it *consistently* shuts down in the final week, resulting in ∞ response time. The aggressive heuristic is inactive for 66% of the time and always ceases to operate before the final week. Consequently, both heuristics fail to collect any data during the wildfire period. In contrast, the RL agent significantly outperforms the heuristics, with only 11% of the time being inactive. The increase in inactive hours compared to the 1-year case can be attributed to the 5% annual battery degradation, which becomes more pronounced over the 5-year duration. As the battery capacity diminishes, the RL agent can no longer maintain its flawless record from the previous case. Notably, the inactive hours begin to increase after approximately 21,000 hours (around 30 months). Despite this, the RL agent achieves a median response time of 2.19 minutes and avoids battery depletion in the final week.

5.3 Ablation Study on the State Space

We conduct an ablation study on the state space to assess the impact of removing three key components: (i) the target battery level, (ii) the cumulative harvested energy (EH) through the current step, and (iii) the moving average of sensor readings. This study aims to unravel the effects of these components on the agent's performance and decision-making process.

(i) Excluding the target battery level from the state space resulted in the agent's inability to incorporate the crucial information regarding the desired battery level from the reward function. As a result, the energy in the battery is significantly underutilized, and *the battery level lingers around the maximum capacity.*

(ii) Removing the cumulative EH information from the state space causes the agent to allocate energy based solely on the instantaneous harvested energy. *Consequently, the assigned sampling rates exhibit high variance, oscillating between 0 and 60 samples/hr.* In contrast, including the cumulative EH information in the state space reduces the fluctuation in sampling rates.

(iii) When the moving average of sensor readings is removed from the state space, the agent fails to learn the relationship between sampling rate and variance in sampling readings. Consequently, it does not increase the sampling rates when the readings deviate from the nominal values. In contrast, the agent with the original state space increases the sampling rates when the fire is near.

These findings underscore the importance of including the target battery level, cumulative EH information, and moving average of sensor readings in the state space. These components enable the agent to make informed decisions and effectively allocate energy and sampling rates to optimize the system's performance.

5.4 Energy Consumption Evaluation on Real Hardware

We deployed the trained RL agent using TensorFlow Lite for Micro (TFLM) flow [14] on the TI CC2652R MCU. The MCU has an ARM Cortex M4F running at 48 MHz and has 352 KB of flash memory and 80 KB of SRAM. The execution time and energy consumption overhead of a single policy network call are measured as 2 ms and 25 μJ, respectively. There are 24 network inference calls daily (i.e., once every hour). Therefore, the daily energy consumption is 0.6 mJ, which is negligible compared to the daily energy consumption of the sensor suite. In addition, the policy network has 13635 parameters. Considering these parameters, the TFLM operators, inputs, outputs, and intermediate values, the total memory footprint of the proposed design is less than 200 KB, which easily fits into the onboard memory. Therefore, deploying and executing the RL agent on the target MCU does not compromise achieving self-sustainable operation.

6 CONCLUSION AND FUTURE WORK

Wildfires continue to be a significant threat, with devastating consequences lasting for years. Early detection and prevention of wildfires are crucial to mitigate their impacts. Sensor suites deployed near the gridlines can enable continuous monitoring and improve detection accuracy. These sensor suites *must* achieve self-sustained operation. To this end, this paper presents a novel self-sustained CPS that co-optimizes the accuracy of wildfire detection and device lifetime by dynamically controlling sensor sampling rates. The proposed approach employs reinforcement learning and is evaluated extensively using real-life datasets and an open-source wildfire simulator. The simulation results show that the proposed framework achieves 89% uptime and a 2-minute response time, and is at least 2.5× faster than a carefully tuned heuristic approach. Moreover, the trained policy network consumes 0.6 mJ daily energy, which is negligible compared to the daily energy consumption of the sensor suite. As a result, the proposed approach enables *reliable, continuous, and granular*

wildfire monitoring. It has significant potential for deployment in remote areas and could enhance the efficiency of wildfire detection and prevention. In addition, the proposed framework allows adaptation to unseen conditions; the model can be fine-tuned using collected data after deployment, providing consistent performance under varying conditions. As the next step, we plan to use our framework to design a multi-agent, neighbor-aware self-sustained sensor-suite network.

A APPENDIX

A.1 Parameter Sweep for the Heuristic

Table A1. All 285 Configurations for c_1, c_2, c_3

c_3	c_2	c_1	c_3	c_2	c_1	c_3	c_2	c_1	c_3	c_2	c_1	c_3	c_2	c_1	c_3	c_2	c_1	c_3	c_2	c_1	c_3	c_2	c_1	c_3	c_2	c_1
15	15	15	30	15	15	33	27	27	36	33	15	39	30	30	42	27	15	42	42	15	45	33	27	45	45	30
18	15	15	30	18	15	33	30	15	36	33	18	39	33	15	42	27	18	42	42	18	45	33	30	45	45	33
18	18	15	30	18	18	33	30	18	36	33	21	39	33	18	42	27	21	42	42	21	45	33	33	45	45	36
18	18	18	30	21	15	33	30	21	36	33	24	39	33	21	42	27	24	42	42	24	45	36	15	45	45	39
21	15	15	30	21	18	33	30	24	36	33	27	39	33	24	42	27	27	42	42	27	45	36	18	45	45	42
21	18	15	30	21	21	33	30	27	36	33	30	39	33	27	42	30	15	42	42	30	45	36	21	45	45	45
21	18	18	30	24	15	33	30	30	36	33	33	39	33	30	42	30	18	42	42	33	45	36	24			
21	21	15	30	24	18	33	33	15	36	36	15	39	33	33	42	30	21	42	42	36	45	36	27			
21	21	18	30	24	21	33	33	18	36	36	18	39	36	15	42	30	24	42	42	39	45	36	30			
21	21	21	30	24	24	33	33	21	36	36	21	39	36	18	42	30	27	42	42	42	45	36	33			
24	15	15	30	27	15	33	33	24	36	36	24	39	36	21	42	30	30	45	15	15	45	36	36			
24	18	15	30	27	18	33	33	27	36	36	27	39	36	24	42	33	15	45	18	15	45	39	15			
24	18	18	30	27	21	33	33	30	36	36	30	39	36	27	42	33	18	45	18	18	45	39	18			
24	21	15	30	27	24	33	33	33	36	36	33	39	36	30	42	33	21	45	21	15	45	39	21			
24	21	18	30	27	27	36	15	15	36	36	36	39	36	33	42	33	24	45	21	18	45	39	24			
24	21	21	30	30	15	36	18	15	39	15	15	39	36	36	42	33	27	45	21	21	45	39	27			
24	24	15	30	30	18	36	18	18	39	18	15	39	39	15	42	33	30	45	24	15	45	39	30			
24	24	18	30	30	21	36	21	15	39	18	18	39	39	18	42	33	33	45	24	18	45	39	33			
24	24	21	30	30	24	36	21	18	39	21	15	39	39	21	42	36	15	45	24	21	45	39	36			
24	24	24	30	30	27	36	21	21	39	21	18	39	39	24	42	36	18	45	24	24	45	39	39			
27	15	15	30	30	30	36	24	15	39	21	21	39	39	27	42	36	21	45	27	15	45	42	15			
27	18	15	33	15	15	36	24	18	39	24	15	39	39	30	42	36	24	45	27	18	45	42	18			
27	18	18	33	18	15	36	24	21	39	24	18	39	39	33	42	36	27	45	27	21	45	42	21			
27	21	15	33	18	18	36	24	24	39	24	21	39	39	36	42	36	30	45	27	24	45	42	24			
27	21	18	33	21	15	36	27	15	39	24	24	39	39	39	42	36	33	45	27	27	45	42	27			
27	21	21	33	21	18	36	27	18	39	27	15	42	15	15	42	36	36	45	30	15	45	42	30			
27	24	15	33	21	21	36	27	21	39	27	18	42	18	15	42	39	15	45	30	18	45	42	33			
27	24	18	33	24	15	36	27	24	39	27	21	42	18	18	42	39	18	45	30	21	45	42	36			
27	24	21	33	24	18	36	27	27	39	27	24	42	21	15	42	39	21	45	30	24	45	42	39			
27	24	24	33	24	21	36	30	15	39	27	27	42	21	18	42	39	24	45	30	27	45	42	42			
27	27	15	33	24	24	36	30	18	39	30	15	42	21	21	42	39	27	45	30	30	45	45	15			
27	27	18	33	27	15	36	30	21	39	30	18	42	24	15	42	39	30	45	33	15	45	45	18			
27	27	21	33	27	18	36	30	24	39	30	21	42	24	18	42	39	33	45	33	18	45	45	21			
27	27	24	33	27	21	36	30	27	39	30	24	42	24	21	42	39	36	45	33	21	45	45	24			
27	27	27	33	27	24	36	30	30	39	30	27	42	24	24	42	39	39	45	33	24	45	45	27			

REFERENCES

[1] Abdulaziz Alarifi and Amr Tolba. 2019. Optimizing the network energy of cloud assisted internet of things by using the adaptive neural learning approach in wireless sensor networks. *Computers in Industry* 106 (2019), 133–141.

[2] Fayçal Ait Aoudia, Matthieu Gautier, and Olivier Berder. 2018. RLMan: An energy manager based on reinforcement learning for energy harvesting wireless sensor networks. *IEEE Transactions on Green Communications and Networking* 2, 2 (2018), 408–417.

[3] Muhammad Hasif bin Azami, Necmi Cihan Orger, Victor Hugo Schulz, Takashi Oshiro, and Mengu Cho. 2022. Earth observation mission of a 6U CubeSat with a 5-meter resolution for wildfire image classification using convolution neural network approach. *Remote Sensing* 14, 8 (2022), 1874.

[4] Barani Design. MeteoTemp RH+T. (n.d.). https://static1.squarespace.com/static/597dc443914e6bed5fd30dcc/t/618a9376bd08d7243005ce49/1636471677532/MeteoTemp-DataSheet.pdf, accessed Mar 2023.
[5] Toygun Basaklar, Yigit Tuncel, and Umit Y. Ogras. 2022. tinyMAN: Lightweight energy manager using reinforcement learning for energy harvesting wearable IoT devices. *arXiv:2202.09297* (2022).
[6] Muhammad Hasif bin Azami et al. 2021. Demonstration of wildfire detection using image classification onboard CubeSat. In *IEEE International Geoscience and Remote Sensing Symposium IGARSS*. 5413–5416.
[7] Taoufik Bouguera, Jean-François Diouris, Jean-Jacques Chaillout, Randa Jaouadi, and Guillaume Andrieux. 2018. Energy consumption model for sensor nodes based on LoRa and LoRaWAN. *Sensors* 18, 7 (2018), 2104.
[8] Greg Brockman et al. 2016. OpenAI Gym. (2016). arXiv:arXiv:1606.01540
[9] Quentin Cappart, Thierry Moisan, Louis-Martin Rousseau, Isabeau Prémont-Schwarz, and Andre A. Cire. 2021. Combining reinforcement learning and constraint programming for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 3677–3687.
[10] IBM ILOG Cplex. 2009. V12. 1: User's manual for CPLEX. *International Business Machines Corporation* 46, 53 (2009), 157.
[11] Miguel G. Cruz and Martin E. Alexander. 2019. The 10% wind speed rule of thumb for estimating a wildfire's forward rate of spread in forests and shrublands. *Annals of Forest Science* 76, 2 (2019), 1–11.
[12] Cubic. Low Power Particle Sensor PM2105L. (n.d.). https://en.gassensor.com.cn/IndoorPMSensor/info_itemid_1830.html, accessed Mar 2023.
[13] Udaya Dampage, Lumini Bandaranayake, Ridma Wanasinghe, Kishanga Kottahachchi, and Bathiya Jayasanka. 2022. Forest fire detection system using wireless sensor networks and machine learning. *Scientific Reports* 12, 1 (2022), 46.
[14] Robert David et al. 2020. Tensorflow lite micro: Embedded machine learning on tinyml systems. *arXiv:2010.08678* (2020).
[15] Gabriel Martins Dias, Maddalena Nurchis, and Boris Bellalta. 2016. Adapting sampling interval of sensor networks using on-line reinforcement learning. In *IEEE 3rd World Forum on Internet of Things (WF-IoT)*. 460–465.
[16] Fire Danger Forecast. Wildland Fire Potential Index (WFPI). (n.d.). https://www.usgs.gov/fire-danger-forecast/wildland-fire-potential-index-wfpi, accessed Mar 2023.
[17] Forest Service. National Fire Danger Rating System. (n.d.). https://www.fs.usda.gov/detail/scnfs/home/?cid=fseprd634957, accessed Mar 2023.
[18] Francesco Fraternali, Bharathan Balaji, and Rajesh Gupta. 2018. Scaling configuration of energy harvesting sensors with reinforcement learning. In *Proceedings of the Workshop on Energy Harvesting & Energy-neutral Sensing Systems*. 7–13.
[19] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*. PMLR, 1587–1596.
[20] Joseph W. Gangestad, Darren W. Rowen, and Brian S. Hardy. 2014. Forest fires, sunglint, and a solar eclipse: Responsive remote sensing with AeroCube-4. In *IEEE Geoscience and Remote Sensing Symposium*. 3622–3625.
[21] Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual. (2023). https://www.gurobi.com, accessed Mar 2023.
[22] William F. Holmgren, Clifford W. Hansen, and Mark A. Mikofski. 2018. pvlib python: A python package for modeling solar energy systems. *Journal of Open Source Software* 3, 29 (2018), 884.
[23] W. Matt Jolly, Patrick H. Freeborn, Wesley G. Page, and Bret W. Butler. 2019. Severe fire danger index: A forecastable metric to inform firefighter and community wildfire risk management. *Fire* 2, 3 (2019), 47.
[24] Jaewoong Kang, Jongmo Kim, Minhwan Kim, and Mye Sohn. 2020. Machine learning-based energy-saving framework for environmental states-adaptive wireless sensor network. *IEEE Access* 8 (2020), 69359–69367.
[25] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B. Srivastava. 2007. Power management in energy harvesting sensor networks. *ACM Trans. Embedd. Comput. Syst.* 6, 4 (2007), 32–es.
[26] David L. King, Jay A. Kratochvil, and William Earl Boyson. 2004. *Photovoltaic Array Performance Model*. Vol. 8. Citeseer.
[27] Timothy P. Lillicrap et al. 2015. Continuous control with deep reinforcement learning. *arXiv:1509.02971* (2015).
[28] Mohamed Maalej, Sofiane Cherif, and Hichem Besbes. 2013. QoS and energy aware cooperative routing protocol for wildfire monitoring wireless sensor networks. *The Scientific World Journal* (2013).
[29] Kevin McGrattan, Randall McDermott, Craig Weinschenk, and Glenn Forney. 2013. Fire Dynamics Simulator, Technical Reference Guide, Sixth Edition. (2013).
[30] Piotr Mirowski et al. 2018. Learning to navigate in cities without a map. *Advances in Neural Information Processing Systems* 31 (2018).
[31] Ankita Mohapatra and Timothy Trinh. 2022. Early wildfire detection technologies in practice-a review. *Sustainability* 14, 19 (2022), 12270.
[32] National Interagency Fire Center. Wildfires and Acres. (n.d.). https://www.nifc.gov/fire-information/statistics/wildfires, accessed Mar 2023.

[33] NRG Systems. NRG S1 ANEMOMETER. (n.d.). https://www.nrgsystems.com/blog/anemometer-wind-speed-threshold/, accessed Mar 2023.
[34] Cristobal Pais, Jaime Carrasco, David L. Martell, Andres Weintraub, and David L. Woodruff. 2021. Cell2Fire: A cell-based forest fire growth model to support strategic landscape management planning. *Frontiers in Forests and Global Change* 4 (2021), 692706.
[35] Yaokun Pang et al. 2020. Multilayered cylindrical triboelectric nanogenerator to harvest kinetic energy of tree branches for monitoring environment condition and forest fire. *Advanced Functional Materials* 30, 32 (2020), 2003598.
[36] Sarbajit Paul and Junghwan Chang. 2019. Design of novel electromagnetic energy harvester to power a deicing robot and monitoring sensors for transmission lines. *Energy Conversion and Management* 197 (2019), 111868.
[37] Huy Xuan Pham, Hung Manh La, David Feil-Seifer, and Matthew C. Deans. 2018. A distributed control framework of multiple unmanned aerial vehicles for dynamic wildfire tracking. *IEEE Transactions on Systems, Man, and Cybernetics* 50, 4 (2018), 1537–1548.
[38] Kunal Shah and Mohan Kumar. 2007. Distributed independent reinforcement learning (DIRL) approach to resource management in wireless sensor networks. In *IEEE International Conference on Mobile Adhoc and Sensor Systems*. 1–9.
[39] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. 2017. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*. arXiv:arXiv:1705.05065 https://arxiv.org/abs/1705.05065
[40] STMicro. Multiprotocol LPWAN 32-bit Arm® Cortex®-M4 MCUs. (n.d.). https://www.st.com/resource/en/datasheet/stm32wle5jb.pdf, accessed Mar 2023.
[41] Yen Kheng Tan and Sanjib Kumar Panda. 2011. Self-autonomous wireless sensor nodes with wind energy harvesting for remote sensing of wind-driven wildfire spread. *IEEE Transactions on Instrumentation and Measurement* 60, 4 (2011), 1367–1377.
[42] Texas Instruments Inc. CC2652R . [Online] https://www.ti.com/product/CC2652R, accessed March 2023. (n.d.).
[43] The European Commission. Data from the EFFIS database. (n.d.). https://effis.jrc.ec.europa.eu/applications/data-and-services, accessed Mar 2023.
[44] Yigit Tuncel, Ganapati Bhat, Jaehyun Park, and Umit Ogras. 2021. ECO: Enabling energy-neutral IoT devices through runtime allocation of harvested energy. *IEEE Internet of Things Journal* (2021). https://doi.org/10.1109/JIOT.2021.3106283
[45] Vanessa Romo. PG&E Pleads Guilty On 2018 California Camp Fire: "Our Equipment Started That Fire". (n.d.). https://www.npr.org/2020/06/16/879008760/pg-e-pleads-guilty-on-2018-california-camp-fire-our-equipment-started-that-fire, accessed Mar 2023.
[46] Chi Yuan, Zhixiang Liu, and Youmin Zhang. 2015. UAV-based forest fire detection and tracking using image processing techniques. In *IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*. 639–643.
[47] Rachid Zagrouba and Amine Kardi. 2021. Comparative study of energy efficient routing techniques in wireless sensor networks. *Information* 12, 1 (2021), 42.

Received 23 March 2023; revised 2 June 2023; accepted 13 July 2023