Power-Aware Data Collection for IoT Cattle Tracking Using Reinforcement Learning

First Author[†,*], Second Author[†], Third Author[◇]
[†] Affiliation One
[‡] Affiliation Two
[◇] Affiliation Three

Abstract

IoT-based cattle trackers enable real-time livestock monitoring, but their energy constraints limit long-term deployment in remote environments. Efficient power management is critical to ensure continuous data collection while maximizing battery life. This study presents a Q-learning-enabled IoT device that dynamically adjusts data collection and transmission schedules to optimize power efficiency. The reinforcement learning agent adapts its strategy based on battery state, time of day, and message queue size. The reward function is tuned using Bayesian optimization to maximize data continuity and power levels. Simulation results show an increase in 4.3% in message count at a substantially higher average power level compared to a fixed scheduling approach of alternating collection and transmission. The agent was deployed on prototype hardware over 9 days and demonstrated the ability to adaptively collect and transmit data to the cloud. These findings highlight the potential for power-adaptive reinforcement learning in IoT systems, contributing to energy-efficient smart agriculture. Future work will focus on hardware hardening and the expansion of the agent to account for message age.

Keywords: Internet of Things (IoT), Reinforcement Learning, Cattle Monitoring.

1. Introduction

Timely tracking of free-range beef cattle is essential to assess herd health and monitor rest, grazing, and rumination behaviours. Understanding grazing patterns on pasture enables cattle producers to optimize feed efficiency, reducing production cost, and lowering environmental impact. Internet of Things (IoT)-based solutions have been developed to replace labor-intensive manual observation with automated, real-time monitoring [1] [2]. The devices take the form of collars, leg bands, or ear tags and are small, low power, and equipped with sensors and communication modules. While their communication abilities are limited, they provide valuable insights into the environment that they are deployed in [1] [3]. However, their installation is a time consuming process, making long-term autonomous operation (months), without human intervention, a key requirement. Power limitations remain another significant challenge, as frequent data transmission depletes battery, necessitating the design of energy-efficient solutions. Solar panels can extend battery life but their effectiveness relies heavily on weather conditions and the cattle's orientation and location.

Reinforcement learning (RL) is a branch of machine learning that allows an agent to learn strategies to maximize a reward. The agent learns by interacting with the environment and updating its behaviour based on the received rewards for those actions[4]. The combination of reinforcement learning and IoT devices, also called the Artificial intelligence of Things (AIoT), offers a promising alternative by dynamically optimizing energy consumption through adaptive decision-making. The TinyML paradigm, a subset of AIoT, focuses on machine learning on devices with highly constrained resources and is often limited to inference only[5]. Overall, the addition of intelligence to the device allows it to make better

[*]corresponding_author@example.ca

decisions, however, on the other hand the addition of intelligence to the device increases the power consumption of the device.

In this paper, we present a reinforcement learning-enabled cattle tracker prototype that optimizes power consumption by deciding when to transmit, store, or delay data collection. The physical device incorporates a solar panel to extend operational longevity, while the Q-learning-based agent adapts to varying power availability to maintain continuous data collection. Our approach leverages RL to outperform static scheduling approaches, ensuring uninterrupted monitoring while strategically checking in and transmitting data, making it a viable solution for long-term autonomous cattle tracking.

2. Related Work

IoT sensors are faced with (i) limited battery life, (ii) limited processing power, and (iii) limited storage[6]. Model training is computationally expensive so devices that fall under the TinyML paradigm are often limited to inference only[5]. The addition of cloud communication opens the possibility of training the model in the cloud and deploying a new model to the device that adapts to changes in the environment.

Existing IoT-based cattle monitoring solutions have primarily focused on health tracking, behavior classification, and disease detection using machine learning and sensor networks. Studies such as [7] have proposed multi-sensory IoT devices for cattle activity monitoring using XGBoost and Random Forest classifiers. [2] introduced an ML-based livestock management system that classifies cattle behavior but does not optimize energy consumption. Additionally, [8] and [9] have applied LoRa-based IoT solutions and federated learning for disease detection, emphasizing data accuracy over power efficiency.

In contrast, our work integrates RL to dynamically optimize power consumption in cattle trackers, ensuring long-term device operation while maintaining data quality. Prior studies, such as [10], demonstrated that deep Q-learning could extend IoT device battery life, but their work relies on a network of devices to determine whether correlated sensors should transmit data or save power. Our system leverages RL-driven decision-making for a single device to balance data transmission, storage, and power conservation, addressing a key limitation in prior research.

3. System Design and Architecture

AIoT device design is an iterative process involving simulations, hardware development, measurement, and redeployment Automation is essential to facilitate rapid development with reduced human intervention. RL helps this process by allowing the agent to adapt to new device properties without the need for manual update of collection strategies.

3.1. Q-learning

Central to Q-learning, a widely used RL approach, is the action-value function $Q(S_t, A_t)$, which is the expected cumulative reward of taking action A_t in state S_t. The agent approximates the optimal policy by updating the action-value function based on the reward that it receives during training[4]:

$$Q^{new}(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \tag{3.1}$$

where α is the learning rate, γ is the discount factor, and R_{t+1} and S_{t+1} are the reward and state after taking action A_t in state S_t, respectively.

The agent's state consists of time of day T, battery power level p, and queued message count m. T (discretized into 48 bins) informs the power generation likelihood, p was discretized to 100 levels, and m was discretized to 5 levels to enforce a transmission check every 2.5 hours to prevent data loss. This resulted in $T \times p \times m = 24,000$ possible states. The agent action space contains three discrete actions: $\mathcal{A} = \{a_0, a_1, a_2\}$, where a_0 =delay data collection, a_1 =collect and store data, and a_2 =collect and transmit data, turning the action-value function into a 24000×3 table.

A tabular representation for $Q(S_t, A_t)$ was used as it turns the model inference into a table lookup with $O(1)$ complexity. Deep Q-learning alternatives, which use a neural network to approximate $Q(S_t, A_t)$ require additional on-device computation and were not considered. A tabular representation does require larger memory or storage, but can be partially updated from the cloud to reflect new conditions.

The goal of the agent is to maximize the number of messages sent to the cloud and the power level of the device. It is not trivial to determine the reward values that achieve this goal. We consider the reward values to be hyperparameters and used a Bayesian optimization framework from the python `scikit-optimize` package to tune them. The hyperparameters (Table 1) show that the most negative reward is for delaying collection. The reward for transmitting data also collects the reward per message, which only becomes attractive if there are messages in the queue.

Table 1. Reward function parameters extracted during hyperparameter tuning.

Parameter	**Description**	**Value**
reward_power_loss	reward for full discharge of battery	0.00
reward_power_multiplier	reward per power level at every time step	0.0001
reward_action_0	reward for delaying collection	-1.716
reward_action_1	reward for collecting and storing data	-0.600
reward_action_2	reward for collecting and transmitting data	-1.485
reward_message_count	reward for each message at transmission	0.841

The agent was trained in a simulated environment with the power consumption and generation numbers collected from the prototype hardware. At each step, the environment updates the battery level, time of day, and queued messages based on the agent's action. The battery (discretized into 100 levels) drains 5.0 mAh per 30 minutes in sleep mode, while transmission and data collection consume 1.6 mAh and 0.1 mAh, respectively.

Solar gain data from 17 June days in Edmonton was averaged into sunny and cloudy profiles, with an 80% probability of a cloudy day. This constraint limited power availability, forcing the agent to optimize message transmission. The training process was repeated four times with different seeds for each hyperparameter search iteration. Once the optimal hyperparameters were found, the action-value function was exported as an h-file for firmware deployment. This automated approach reduced manual effort and effectively optimized reward values.

3.2. **Hardware**

The hardware prototype is shown in Figure 1 and consists of an Arduino microcontroller, a GPS device, a solar array, a power distribution system, and a battery. The detailed specifications are shown in Table 2. Prototype testing revealed high sleep power consumption (9.9 mA) caused by always-on LEDs and integrated components. A custom PCB with full component isolation is planned to reduce sleep current to μA levels. An unexpected benefit of high power draw was a faster development cycle, as battery depletion occurred in days rather than weeks.

Table 2. Hardware list for the cattle tracker prototype

Hardware	Description	Serial Number
Arduino MKR WiFi 1010	WiFi-enabled microcontroller board	ABX00023
Arduino MKR GPS Shield	Integrated GPS module	ASX00017
DFRobot Solar Power Manager	Power distribution board	DFR0559
Solar Panel	75x100x2.9 mm panel, 1W output	TPX00181
Sparkfun Fuel Gage	Power measurement sensor	MAX17043
LiPo Battery	3.7V, 2100mAh rated, 1300mAh eff. capacity	2528

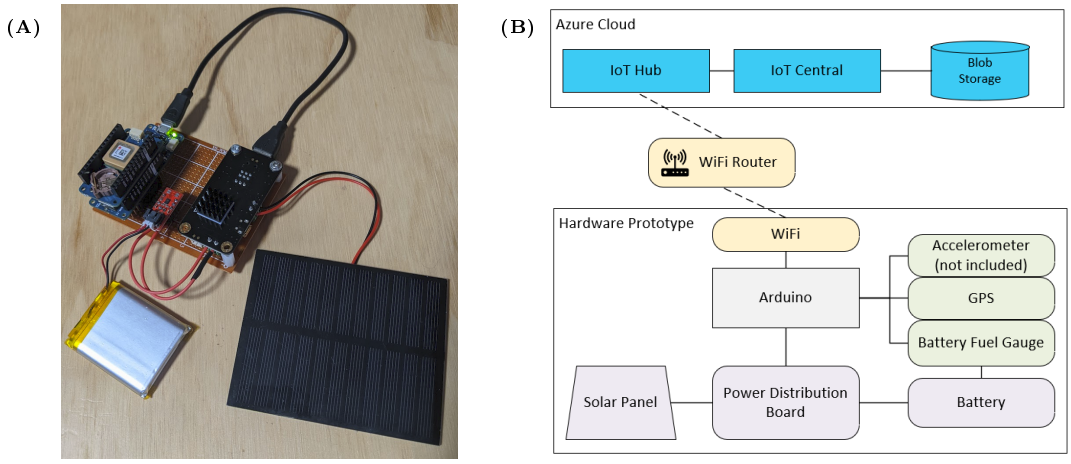

Figure 1. (A) Physical prototype with hardware components. (B) System architecture diagram illustrating data processing and power management.

3.3. Firmware

The prototype firmware, written in Arduino C++, handles sensor data collection, decision-making, and cloud transmission (see Figure 1 B). The main loop runs every 30 minutes, alternating between sleep mode and decision-making. At each decision point, the device reads the time, battery level, and queued messages, retrieves a recommended action from the policy table, and executes it. Due to memory constraints, only the highest-value action per state was stored, reducing the policy table size. The policy was bit-shifted into an array of 32-bit integers for efficient storage in memory.

3.4. Cloud

The prototype transmits data to Azure IoT Hub, which integrates with IoT Central for dashboarding and Blob Storage for data retention (see Figure 1 B). IoT Central supports device registration, dashboard creation, and digital twins, enabling remote monitoring and state updates. The cloud facilitates two-way communication, allowing the device to receive policy updates for potential seasonal changes or location-specific solar gain variations.

4. Results

4.1. Simulation Training Results

To evaluate the effectiveness of the reinforcement learning agent, we compare it to a baseline collection strategy that alternates between data collection and transmission, sending data once per hour. This represents a simple, fixed scheduling approach. Table 3 shows that

the RL agent reduces transmissions, conserving power while increasing collection actions, leading to a higher message count (+4.3%) and a more sustainable average power level (76 vs. 34). The trained agent optimized collection-to-transmission behavior, performing 2.12 collection actions per transmission, compared to the 1:1 fixed schedule of the baseline. This suggests the agent learned an adaptive strategy, balancing energy efficiency with message throughput.

Table 3. Comparison of baseline agent and RL agent

	Baseline Agent	RL Agent
Transmission Actions	480	308
Collection Actions	480	652
Sleep Actions	0	0
Avg. Message Count	756.5	789.5
Avg. Power Level	34	76

4.2. **Deployment Results**

The trained agent was deployed on the prototype hardware in an indoor environment without power generation for a 9-day evaluation period, with occasional manual charging to ensure continuous operation. This test assessed the agent's long-term decision-making performance in real hardware and validated its ability to operate under realistic power constraints. The results are shown in Figure 2. During the test, the hardware experienced several outages, the longest lasting 3.6 hours on 2024-10-07. The agent maintained an average collection-to-transmission ratio of 2.18, which closely matches the simulation results.

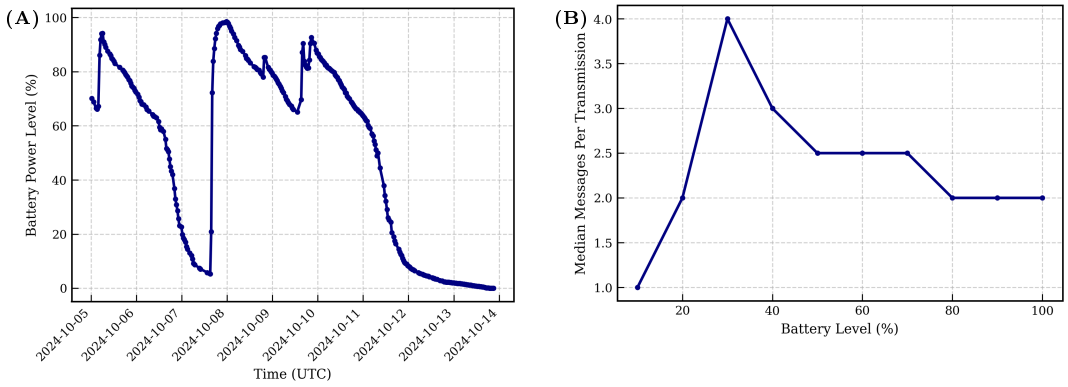

Figure 2. (A) Nonlinear power level discharge over a 9-day period. (B) Learned agent strategy adapting to varying power levels.

Figure 2A illustrates the nonlinear battery discharge curve observed over the 9-day period. The discharge rate during sleeping is relatively constant and the nonlinearity is a result of the mapping function of voltage to remaining capacity. While sleep mode consumption remains constant, the voltage-to-capacity mapping introduces nonlinearity, making it difficult for the agent to accurately predict future battery levels at low charge states.

Figure 2B shows the agent's adaptive strategy under varying power levels. The agent alternates between collection and transmission at full charge, prioritizes storage with a 4:1 collection-to-transmission ratio at 30% power, and transmits every collected message at low power which accelerates battery drain. Messages are stored in volatile memory so the queue

is lost when the battery is fully discharged. The agent learned to transmit every time to prevent data loss at low power levels. This revealed a short-coming of the prototype and future iterations will store the queue in non-volatile memory to prevent data loss.

5. Conclusion

In this study, we developed a Q-learning-enabled cattle tracker that optimizes power consumption while ensuring efficient data collection and transmission. The device dynamically adjusts its strategy based on energy availability, time of day, and message queue length, improving battery life and efficiency. By integrating solar power with adaptive decision-making, the prototype offers a viable solution for long-term autonomous tracking. Future work will focus on hardware optimizations, including low-power components and a custom PCB to enhance reliability and longevity. Additionally, we aim to increase the message queue length, improve the reward function to account for the age of the messages, and incorporate non-volatile memory storage.

Acknowledgements

References

[1] O. Unold, M. Nikodem, M. Piasecki, K. Szyc, H. Maciejewski, M. Bawiec, P. Dobrowolski, and M. Zdunek. "IoT-Based Cow Health Monitoring System". In: *Computational Science – ICCS 2020*. Ed. by V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, and J. Teixeira. Vol. 12141. Cham: Springer International Publishing, 2020, pp. 344–356.

[2] N. Yamsani, K. Muthukumaran, B. S. Kumar, A. V, N. Singh, and J. Dhanraj. "IoT-Based Livestock Monitoring and Management System Using Machine Learning Algorithms". In: *2024 International Conference on Science Technology Engineering and Management (ICSTEM)*. Coimbatore, India: IEEE, Apr. 2024, pp. 1–6.

[3] K. E. M. Moutaouakil, H. Jdi, B. Jabir, and N. Falih. "Digital Farming: A Survey on IoT-based Cattle Monitoring Systems and Dashboards". In: *Agris on-line Papers in Economics and Informatics* 15.2 (June 2023), pp. 31–39.

[4] R. S. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. Second edition. Adaptive Computation and Machine Learning. Cambridge, Massachusetts London, England: The MIT Press, 2020. ISBN: 978-0-262-03924-6.

[5] P. P. Ray. "A Review on TinyML: State-of-the-art and Prospects". In: *Journal of King Saud University - Computer and Information Sciences* 34.4 (Apr. 2022), pp. 1595–1623.

[6] W. Chen, X. Qiu, T. Cai, H.-N. Dai, Z. Zheng, and Y. Zhang. "Deep Reinforcement Learning for Internet of Things: A Comprehensive Survey". In: *IEEE Communications Surveys & Tutorials* 23.3 (2021), pp. 1659–1692.

[7] D. Dutta, D. Natta, S. Mandal, and N. Ghosh. "MOOnitor: An IoT Based Multi-Sensory Intelligent Device for Cattle Activity Monitoring". In: *Sensors and Actuators A: Physical* 333 (Jan. 2022), p. 113271.

[8] J. Arshad, A. Irtisam, T. Arif, M. S. Rasheed, S. T. Chauhdary, M. K. I. Rahmani, and R. Almajalid. "A Federated Learning Model for Intelligent Cattle Health Monitoring System Using Body Area Sensors and IoT". In: *Egyptian Informatics Journal* 27 (Sept. 2024), p. 100488.

[9] S. A. I, C. Priya, R. Premkumar, and H. N. S.M. "Real Time Cattle Health Monitoring and Early Disease Detection Using IoT and Machine Learning". In: *2024 5th International Conference on Electronics and Sustainable Communication Systems (ICESC)*. Coimbatore, India: IEEE, Aug. 2024, pp. 450–455.

[10] J. Hribar, A. Marinescu, G. A. Ropokis, and L. A. DaSilva. "Using Deep Q-Learning to Prolong the Lifetime of Correlated Internet of Things Devices". In: *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. Shanghai, China: IEEE, May 2019, pp. 1–6.