Autonomous Robot Dog Home Inspector

Daniel Palomera
Department of Computer Science, CSUF
800 N State College Blvd
Fullerton, California 92831-3599
dpalomera0@csu.fullerton.edu

Tommy Nguyen
Department of Computer Science, CSUF
800 N State College Blvd
Fullerton, California 92831-3599
tn7802@csu.fullerton.edu

Mason Jennings
Department of Computer Science, CSUF
800 N State College Blvd
Fullerton, California 92831-3599
masonj@csu.fullerton.edu

Osvaldo Torres
Department of Computer Science, CSUF
800 N State College Blvd
Fullerton, California 92831-3599
osvaldotorres3108@csu.fullerton.edu

ABSTRACT

This paper presents the design and implementation of an autonomous quadrupedal robot developed for residential inspection tasks. Building on a suite of custom camera control modules, the system integrates perception, navigation, and environment-aware decision-making to enable reliable operation in cluttered indoor spaces. The robot dog employs modular sensing, real-time video processing, transferring media across different machines an adaptive path planning to identify structural issues, document interior conditions, and navigate common household obstacles with minimal human intervention. We describe the hardware architecture, software pipeline, and control strategies that support robust inspection performance, and we evaluate the system through a series of controlled home-environment trials demonstrating its effectiveness and operational stability.

CCS Concepts

Interfacing and APIs (Application Programming Interfaces)

The interaction with the **DJI Camera** is the most critical software challenge here. Since the hardware is closed source, the Raspberry Pi cannot directly manipulate the camera's internal registers. Instead, it must use open-sourced methodologies to simulate inputs to the camera.

- **Abstraction:** The system treats the camera as a "black box." The Raspberry Pi sends high-level commands without knowing *how* the camera executes that instruction electronically.
- **SDK Usage:** To send specific commands like "switch to File Transfer Mode," the Raspberry Pi utilizes our reverse-engineered library.
- **Control Signals:** The "remote input commands" represent control signals sent over a physical interface (USB or GPIO), translating logical requests into hardware actions. This also represents as well for wireless methods such as Wi-fi or Bluetooth.

Networking and Data Transmission

The system relies heavily on data movement, utilizing different transmission mediums and protocols.

- **Client-Server Architecture:** When the Pi uploads files to a server via WiFi, it acts as the **Client**, initiating requests to a remote **Server** which stores the data.
- **Protocols (TCP/IP):**
    - **Wireless:** The "Pi's wifi" implies the use of the **802.11** standard (WiFi) to transport data packets.
    - **Application Layer:** Uploading files likely uses protocols such as **FTP (File Transfer Protocol)**, **SFTP (Secure File Transfer Protocol)**, or **HTTP/HTTPS** POST requests.
- **Wired Communication:** The connection to the "local hard drive" uses a bus protocol (USB via a bridge), which offers higher bandwidth and lower latency compared to the wireless connection.
- **Paramiko Python Library:** Use to provide and implement ssh protocols, enabling secure connections to remote servers.

Operating Systems and Embedded Systems

The **Raspberry Pi** acts as the central controller (the "brain"), requiring robust OS concepts to manage resources.

- **Device Drivers:** To communicate with the DJI camera, the Raspberry Pi's operating system (likely Linux-based) must load specific **kernel modules (drivers)** to recognize the hardware.
- **Multithreading/Concurrency:** The Pi must likely perform multiple tasks simultaneously:
    - Listening for remote commands.
    - Monitoring the camera status.
    - Uploading a file to the server.
    - *Concept:* The OS scheduler ensures these processes run concurrently without blocking each other (e.g., the video feed shouldn't freeze just because a file is uploading).
- **File Systems:** To write to a "local hard drive," the Pi must manage file systems (e.g., ext4, NTFS, FAT32), handling mounting points and permission management.

State Management (Finite State Machines)

The camera logic described involves distinct **modes**: Photo/ Video, and File Transfer.

- **Finite State Machine (FSM):** The software on the Pi must track the state of the camera.
    - *Example:* If the camera is in `File Transfer Mode`, the system must know that it *cannot* execute a `take_photo` command until it transitions back to `Photo Mode`.
- The logic defines valid transitions (e.g., Idle →
    - Record → Idle → Transfer).

Edge Computing / IoT (Internet of Things)

This entire setup is a classic example of an **Edge Device**.

- **Edge Processing:** instead of sending raw data to the cloud immediately to be processed, the Raspberry Pi (the "Edge") manages the hardware locally. It decides *where* the data goes (Server vs. Local Drive) and *when* to send it.

- **Latency vs. Bandwidth:** The choice between WiFi (Server) and Wired (Local Drive) is a trade-off. WiFi offers mobility but has lower bandwidth and higher latency; Wired offers speed but restricts movement (unless the drive is mounted on the dog).

Keywords – o*smo action 5, unitree robot dog go 2, month2month, property/home automation, artificial intelligence, ai*

1. INTRODUCTION

Rising labor costs, safety concerns, and the growing demand for rapid property assessments have created a pressing need for more efficient inspection methods. Traditional, home-damage inspections require trained personnel to travel to each site, often navigating hazardous environments, and incurring significant operational expenses. These constraints limit scalability and slow response times. Particularly after large-scale events such as storms, earthquakes, or insurance-related surges. To address these challenges, this work presents an autonomous robot-dog-based home inspection system designed to perform on-site assessments without requiring humane presence. Built on advances in mobile robotics, embedded sensing, and autonomous navigation, the proposed platform leverages a quadrupedal robot capable of traversing uneven terrain, entering confined spaces, and capturing multimodal data for structural analysis. The system integrates environment-aware navigation, damage classification pipelines, and remote operator oversight to deliver reliable, repeatable inspections at a fraction of the cost of traditional methods. By replacing routine human site visits with an autonomous agent, organizations can reduce risk exposure, accelerate claim processing, and expand inspection coverage.

This paper details the system architecture, sensing modalities, autonomy stack and evaluation results from real-world test deployments. The findings demonstrate that quadrupedal robotic inspectors can serve as a practical, cost-effective alternative to manual property assessments, paving the way for scalable, robotics driven inspection workflows.

2. RELATED WORK

The development of autonomous inspection systems has shifted significantly from wheeled platforms to quadrupedal robots due to their superior mobility in unstructured environments [1]. Recent literature highlights advances in locomotion, deep learning-based defect detection, and edge computing architectures that directly inform the design of our Autonomous Robot Dog Home Inspector.

2.1 Quadrupedal Navigation in Cluttered Environments

While wheeled robots have traditionally been used for mapping flat terrain, they struggle with the verticality and debris common in post-disaster home environments. Halder et al. [7] explored the efficacy of quadruped robots for construction monitoring, demonstrating that legged systems significantly outperform tracked or wheeled alternatives when traversing irregular surfaces such as loose gravel or steps. Their study emphasizes that successful human-robot teaming in inspection tasks requires the robot to autonomously handle obstacle avoidance. Building on this need for robust autonomy, Lan [8] recently proposed a motion planning framework combining Nonlinear Model Predictive Control (NMPC) and Whole-Body Control (WBC) for the automatic inspection of complex facilities. While Lan's work focused on space launch sites, their successful demonstration of global trajectory planning to automate quadrupedal movement validates our approach of using legged robots to autonomously navigate and assess structural conditions in constrained environments.

2.2 Edge Computing and IoT Architectures

The shift from cloud-centric to edge-centric processing is defining modern field robotics. Filho et al. [6] proposed a three-layer architecture (Cloud-Edge-Terminal) for inspection robots, arguing that raw video data should be processed locally on the "Edge" to ensure operation in communication-denied environments. However, relying solely on local processing can limit performance. Addressing this, Nouruzi-Pur et al. [9] introduced redundancy concepts for real-time control that utilize robot-controlled switching between cloud and edge computation nodes. Their research demonstrates that dynamically shifting tasks between local hardware and remote servers significantly enhances system reliability in uncertain network infrastructures. This directly supports our system's logic, where the Raspberry Pi acts as a resilient Edge controller, autonomously deciding whether to upload files to the server or write to the local drive to prevent data loss during WiFi dropouts.

3. METHOLOGY

3.1 Remote Control Camera Implementation

Our team obtained the DJI Camera and began to research different ways to control the camera remotely while attached to the robot dog. There were two primary sources we found and used extensively during our implementation: the *DJI-SDK*, the official SDK for a model similar, but not identical to our model, and an unofficial, reverse-engineered SDK [5], *M5StickC Plus2 Remote Control*, created by GitHub user *theserialhobbyist* [10]. Using a combination of these two sources, we were able to implement remote-control functions for our specific camera model, written in Python, using the `bleak` library for Bluetooth support. Our primary reason for using Python was the ability of itself and its libraries to run identically on many different devices, making it easy to run and debug our program on our development systems before deploying it to the Raspberry Pi.

3.2 Server Upload Implementation

Our team created a Python script to upload videos recorded on the DJI camera to an arbitrary web server, using the `Paramiko` Python library for SSH file transfer support. In this implementation, the Raspberry Pi runs the scripts and acts as the internet-connected device that uploads each file to the server. This script begins by creating a connection to the server. It then looks in the camera's directory and uploads files that have non-matching names to files that already exist on the server. The script can also download media from the server to the local machine

We also use a `config.json` file to hide our connection's personal attributes, such as the server's IP address, port number, username, password/personal access token, and local and remote directory paths. This allows us to maintain connection privacy and allow these attributes to be easily updated.

3.3 Bringing it All Together

Our team implemented a main menu in order to easily switch between *Remote Control Camera* mode and *Server Upload* mode. Also written in Python, this script begins by connecting the camera with Bluetooth using an implemented camera connection function `connect()` and maintains the connection for the entire session. Once the connection is made, the main menu appears, allowing you to select between either of the two modes. Once either mode exits, the user will be returned back to the menu to make another selection. Once the user quits, the camera cleanly disconnects using the `disconnect()` function.

One of the issues we ran into while switching between the two modes occurred when the camera itself was in file transfer mode. When in this mode, the camera's Bluetooth connection persisted, but could not make recordings or take pictures. To rectify this, we discovered that the Raspberry Pi 5's USB ports use a standardized hub controller, commonly found in USB docks and hubs but are rare in laptop and desktop computers. Because of this, we were able to use the Linux package `uhubctl` to control the power of each individual USB port, allowing the camera to seamlessly switch between the two modes in practice. Upon launching, the USB port is disabled. When entering Server Upload mode, the ports are enabled when launched and disabled when the script ends. From the main menu, once the camera disconnects, the USB port is re-enabled.

4. RESULTS

By the end of the term, our team was able to achieve our goal to create a working prototype that allows the remote control of an Osmo Action Pro 5 via software, and the ability to upload media the camera takes to a company server. Once our initial implementation was complete, we began software testing. With regard to errors, the most common ones we experience were related to connecting to, disconnecting from, and maintaining a Bluetooth connection to the camera. We make attempts to rectify this by adding error checking and exception handling, but more unexpected connection errors will still sometimes occur. Overall, our success rate was very strong as the camera was able to remotely capture and upload nearly all of the time.

5. CONCLUSION

This article showcases a working prototype of a robotic dog capable of remotely controlling a proprietary camera through the use of Bluetooth BLE protocol. The prototype can also record and send the data to a remote server. This additional camera is capable of high resolution 4k 60 frames per second video which can then later be used to accurately verify the conditions of home manually or through the use of the *You Only Look Once (YOLO)* algorithm.

6. ACKNOWLEDGMENTS

Our thanks to Month2Month for sponsoring the robotic dog and the camera for the research and development. We would also like to thank Dr. Yu Bai as our professor for granting us the opportunity to work on this project. Thank you to the graduate students and CSUF ECS faculty for providing extra support during the development process and insight.

7. REFERENCES

[1] Yunduan Lou, Pu Sun, Yifeng Yu, Shangping Ren, and Yu Bai. 2025. TT-DSC: Enhancing YOLO for Marine Ecosystem through Efficient Tensor Train-based Depthwise Separable Deep Neural Network. ACM Trans. Auton. Adapt. Syst. Just Accepted (May 2025). https://doi.org/10.1145/3735138

[2] X. Ma, P. Sun, S. Luo, Q. Peng, R. F. DeMara and Y. Bai, "Binarized l1 -Regularization Parameters Enhanced Stripe-Wise Optimization Algorithm for Efficient Neural Network Optimization," in IEEE Journal of Emerging and Selected Topics in Industrial Electronics, vol. 5, no. 2, pp. 790-799, April 2024, doi: 10.1109/JESTIE.2023.3313050.

[3] U.S. Census Bureau. American Housing Survey (AHS). Census.gov, https://www.census.gov/programs-surveys/ahs.html. Accessed 10 Dec. 2025.

[4] California Research Bureau. CRB Single-Family Housing Rentals [Tableau dashboard]. Tableau Public, https://public.tableau.com/app/profile/california.research.bureau/viz/CRB-SingleFamilyHousingRentals/MainView. Accessed 10 Dec. 2025.

[5] DJI-SDK. Osmo-GPS-Controller-Demo. GitHub, https://github.com/dji-sdk/Osmo-GPS-Controller-Demo. Accessed 11 Dec. 2025.

[6] R. Silva Filho, B. Yu, and C. Huang, "The edge architecture for semi-autonomous industrial robotic inspection systems," International Journal of Cloud Computing, vol. 9, no.[2] 1, pp. 45-63, Mar. 2020.

[7] S. Halder, K. Afsari, E. Chiou, R. Patrick, and K. A. Hamed, "Construction inspection & monitoring with quadruped robots in future human-robot teaming: A preliminary study," Journal of Building Engineering, vol. 65, p. 105814, Apr. 2023.

[8] J. Lan, "An automatic inspection method for space launch sites," Journal of Physics: Conference Series, vol. 3041, no.[1] 1, p. 012030, Jun. 2025.[1] doi: 10.1088/1742-6596/3041/1/012030

[9] J. Nouruzi-Pur, J. Lambrecht, T. D. Nguyen, A. Vick, and J. Krüger, "Redundancy Concepts for Real-Time Cloud- and Edge-based Control of Autonomous Mobile Robots," in 2022 IEEE 18th International Conference on Factory Communication Systems (WFCS), Pavia, Italy, 2022, pp. 1-4. doi: 10.1109/WFCS53837.2022.9779202.

[10] TheSerialHobbyist. M5StickCPlus2_Remote_For_DJI_Osmo. GitHub, https://github.com/theserialhobbyist/M5StickCPlus2_Remote_For_DJI_Osmo. Accessed 10 Dec. 2025.