**CAPSTONE PROJECT REPORT**

**LocalRobots2**

****

Submitted by:

[Logan Huber](mailto:lhuber@mail.bradley.edu), Rose Marie Godoy, Alec Pizziferro, Steven Watanabe, Josephine Qualls, Pranaav Jamigumpala

Under the guidance of/Submitted to:

Dr. Tony Grichnik

Bradley University, Department of Computer Science

**Table of Contents**

[**1. Abstraction 2**](#_ad4n2itnmtv)

[**2. Introduction 3**](#_me670e6xl6ho)

[2.1 Project Overview 3](#_anydv8wzm9ql)

[2.2 Objectives 4](#_2yzivttku07k)

[2.21 Capstone Objectives 4](#_4lceahjs8pyg)

[2.22 Robot Objectives 5](#_ug6er38ior2z)

[2.23 Virtual Simulation Objectives 5](#_ubgy6blc44vm)

[**3. Challenges and Problems 5**](#_mpneggh80xzp)

[**4. Robot Specifications 7**](#_2nvh5hfk15n8)

[4.1 Connecting to Robot 10](#_36feztgqouyh)

[**5. Tools and Resources 10**](#_ota5xy6nk7vw)

[**6. Finding and Results 11**](#_436db3bn6sw4)

[**7. Next Steps (for Spring 2025) 13**](#_sqhmncdgrdc)

# **Abstraction**

This Capstone project focuses on developing an autonomous quadruped robotic system capable of navigating dynamic environments and performing specific tasks such as obstacle detection, path planning, and object recognition. The project integrates robotics, machine learning, sensor technology, and simulation to create a versatile platform for experimentation and educational use.

**High-Level Abstraction**

Conceptually, the project aims to enable the robot to autonomously explore its surroundings, avoid obstacles, and interact with objects. The system combines real-time sensor data with machine learning algorithms to adapt to unstructured environments. The virtual simulation mirrors this functionality, providing a safe and controlled environment to test features before physical implementation, reducing risks and enhancing reliability.

**Low-Level Abstraction**

On a technical level, the physical robot relies on a ROS2 framework, which integrates sensors such as LiDAR, ultrasonic modules, and a camera for environmental data collection. Machine learning algorithms developed in Python, using TensorFlow, perform object detection and classification. These processes are controlled by a C++-based system that ensures precise motor control and real-time adaptability.

In parallel, the virtual team developed an accurate simulation of the robot using Webots, focusing on realistic modeling of quadruped movement through inverse kinematics. This simulation allows testing of core functionalities such as navigation and object detection in a virtual space, with results informing the implementation of similar capabilities on the physical robot.

By integrating these efforts, the project balances high-level system goals with detailed technical solutions, creating a robust robotic platform for both practical application and academic exploration.

# **Introduction**

## **2.1 Project Overview**

This Capstone project aims to develop an autonomous robotic system capable of navigating dynamic environments and performing tasks such as object recognition and obstacle avoidance. Using Raspberry Pi-based quadruped robots (DOGZILLA), the physical team focuses on building a hands-on platform for AI and machine learning courses.

To minimize risks and streamline testing, the virtual team has created a Webots simulation of the robot, allowing for experimentation in a controlled environment. This integrated approach ensures robust functionality while providing valuable educational tools and advancing autonomous robotics capabilities.

Our group consists of two teams:

1. **Physical**

* Built and maintained robot chassis
* Executed, handled, and recorded the factory scripts in the Raspberry Pi’s
* Check for deviations in the virtual robot's behavior executing physically
* Documented results of execution of factory settings for navigation and mapping
* Audited and logged data input streams from sensors

1. **Virtual**

* Created Robot Operating System (ROS) scripts
* Simulate robot behaviors in a virtual environment (WeBots)
* Executed & created challenges in WeBots
* Created scripts for navigation and mapping

## **2.2 Objectives**

The primary objectives of this Capstone project are to develop, test, and refine an autonomous robotic system that combines physical and virtual functionalities for education and advanced robotics applications. By integrating efforts across both teams, the project seeks to achieve the following:

### **2.21 Capstone Objectives**

1. Replace locally produced robots with two Raspberry Pi-based quadruped robots (DOGZILLA) equipped with LIDAR.
2. Design new challenges for AI and machine learning classes, expanding beyond the existing DOGZILLA repository.
3. Implement these challenges both in physical robots and within virtual simulations.
4. Demonstrate the robot’s capabilities through an engaging final presentation, such as maze traversal or location pinpointing and navigation.

### **2.22 Robot Objectives**

1. Develop an Autonomous Navigation System: Create a reliable system for independent exploration and movement using tools like ROS, Python, and C++.
2. Implement Obstacle Detection and Avoidance: Integrate sensor systems for safe navigation in dynamic environments.
3. Incorporate Machine Learning for Object Recognition: Train and deploy models to recognize and respond to objects in real-time.
4. Create a Robust Control System: Ensure smooth integration of control mechanisms with sensors and machine learning, allowing precise and adaptive robot movement.
5. Test and Optimize Performance: Conduct iterative testing in both physical and virtual environments to refine functionality and improve reliability.

### **2.23 Virtual Simulation Objectives**

Simulate all functionalities of the physical DOGZILLA robot, enabling controlled, physically accurate testing. This includes careful modeling of inverse kinematics to achieve realistic leg movement for effective spatial navigation.

By aligning these objectives, the project aims to deliver a versatile autonomous robotic system capable of functioning in real-world scenarios while also serving as a valuable educational tool.

# **3. Challenges and Problems**

During the course of this Capstone project, our team encountered various challenges related to both the physical and virtual aspects of developing and testing the robotic system. These challenges required iterative problem-solving and coordination between both teams to ensure progress.

**Integration of Physical and Virtual Efforts**

One of the overarching challenges was combining the physical team's work with the virtual team's efforts. This included synchronizing the physical team’s work on Jupyter Notebooks and the robot’s hardware with the virtual team’s work on ROS2. Ensuring seamless collaboration and functional integration required significant effort, particularly given the differences in platforms and resources.

**Physical Team Challenges**

* **Hardware and Sensor Issues:** Troubleshooting issues such as the camera display not working required resetting the capture widget and using VNC. Additionally, retrieving full LiDAR data proved challenging due to difficulties in saving RViz2 images and outputting data effectively.
* **Balancing Advanced Features and Stability:** Testing advanced features often required force-stopping critical startup functions, necessitating frequent robot resets. Furthermore, balancing Raspberry Pi’s processing power across base functionalities, experiments, and data gathering was complex.
* **File Management and Organization:** The robot's Raspberry Pi files were poorly organized, complicating file integrity checks and the retrieval of necessary data.
* **Mobility and Navigation Limitations:** Physical constraints such as the robot's inability to climb stairs higher than 2 cm limited its adaptability. Creating a map of the first floor of Bradley University was time-intensive due to the building’s large dimensions and tight deadlines.

**Virtual Team Challenges**

* **Robot Operating System (ROS) Issues:** ROS presented significant challenges in both setup and compatibility. Running ROS2 required Linux-based systems, with limited functionality on Windows. While a more comprehensive guide was located to aid in setup, issues like configuring OpenSSL and setting environmental variables persisted. Debugging and behavior testing on simulated robots added further complexity.
* **Simulation and Gait Solvers:** Developing realistic movement for the virtual quadruped robot involved overcoming challenges with inverse kinematics (IK) and forward kinematics (FK). While IK was identified as a suitable method for driving motion, creating an accurate gait required substantial experimentation. Attempts to leverage open-source controllers were limited by imprecise movement and compatibility issues with DOGZILLA.
* **Communication Between Robots:** Designing a system where robots could communicate path deviations within a maze was particularly complex, requiring a robust server for real-time calculations and messaging.

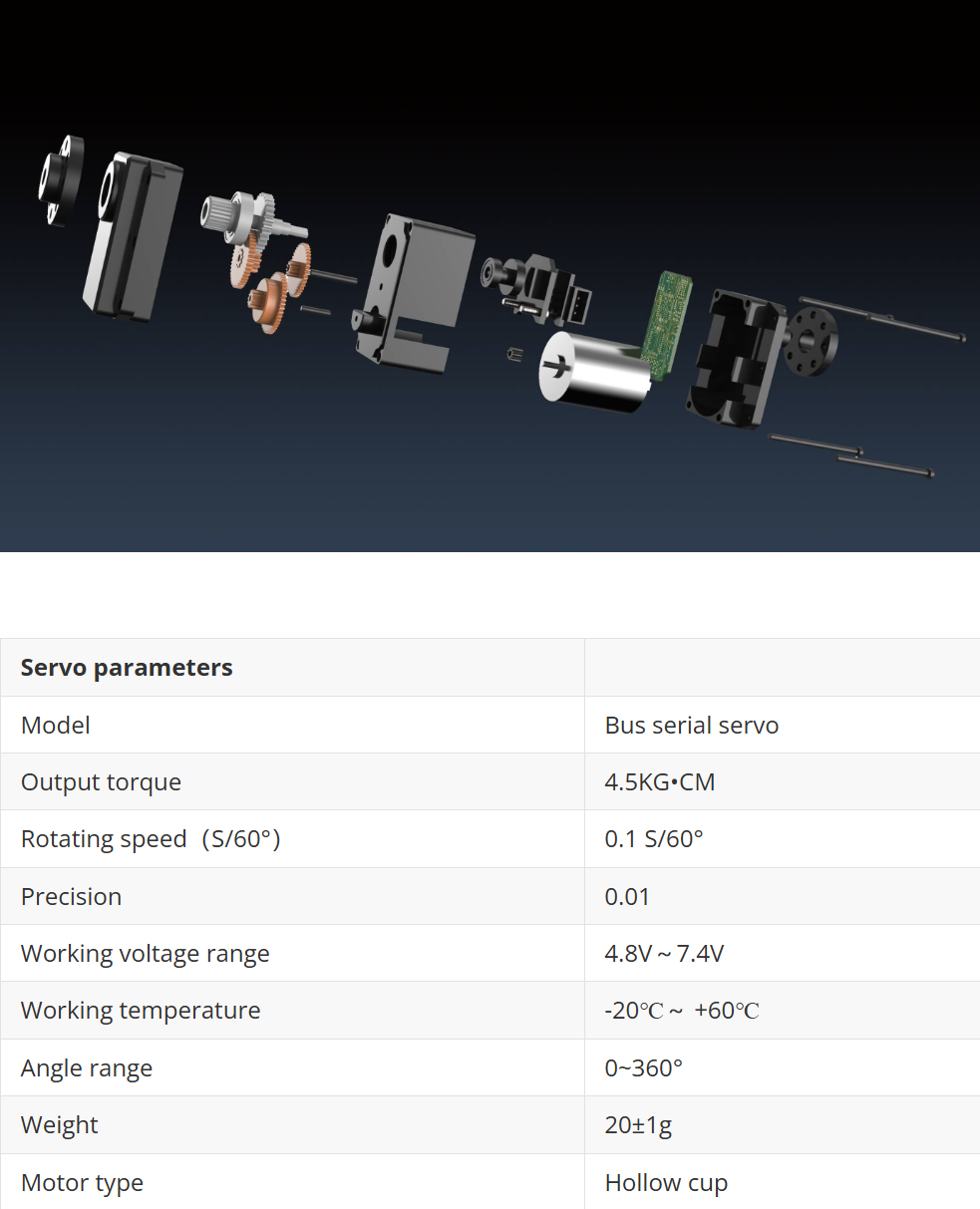
**Project Timeline and Demonstration Goals**

A critical constraint was ensuring the robot's readiness for a functioning demo by mid-March 2025. This required careful prioritization of tasks to meet deadlines while maintaining functionality. Additionally, both physical and virtual teams had to ensure that DOGZILLA robots were project-ready within four weeks for integration with an AI and ML learning class.

# **4. Robot Specifications**

DOGZILLA is an advanced visual AI robot dog designed for versatility and movement featuring:

* 12 degrees of freedom with 6 high-precision servos and an aluminum alloy body
* Realistic movement capabilities, including walking and twisting like a real dog
* Omni-directional movement and six-dimensional attitude control
* Equipped with IMU and servo angle sensors for real-time posture and joint angle feedback
* Utilizes inverse kinematics algorithms for various motion gaits
* Powered by Raspberry PI 4 as the main controller, with Ubuntu 20.04 ROS2 for programming and RVIZ simulation
* Supports AI visual recognition, lidar mapping navigation, and voice control through Python programming
* Compatible with multiple remote control methods, including APP, handle, web pages, and computer keyboards
* Features advanced functions like label recognition, face detection, target tracking, and visual line patrol
* The S2 model adds lidar and intelligent voice modules for enhanced mapping navigation, obstacle avoidance, and voice commands



## 4.1 Connecting to Robot

Every time that you want to connect to the robot, it is **mandatory** to be connected to the robot's Wifi signal. This is true whether you connect with the phone app or wireless handle controller, using Virtual Network Computing (VNC), SSH, SCP, rdp, or jypternotebook. The different ways to connect are noted in [DOGZILLA’s repository](http://www.yahboom.net/study/DOGZILLA) in sections 3 & 5, but we will go over the pros and cons of it.

The following are the Login Credentials (IMPORTANT):

Login user and password-related content:

Username: **pi** Password: **yahboom**

Vnc remote login password: **yahboom**

Jupyterlab login password: **yahboom**

Factory hotspot signal: **DOGZILLA\_WIFI** Password: **12345678**

# **5. Tools and Resources**

LocalRobots Google and GitHub access codes and passwords

1. GitHub Login - **LocalRobots**

2. GitHub Password - **Webots123()**

3. Google Account Login - **localrobots0@gmail.com**

4. Google Account Password – **Webots123()!**

NOTE: Base DOGZILLA Factory File (Zipped – **RPi07\_DOGZILLA**) is in the Google Drive

Dogzillas Repository (GitHub)

<https://github.com/YahboomTechnology/DOGZILLA/tree/main>

Dogzillas Repository (Primary with downloads)

<http://www.yahboom.net/study/DOGZILLA>

Backing up your Raspberry Pi

<https://forums.raspberrypi.com/viewtopic.php?t=26463>

Open CV and Understanding Image Processing In Python

<https://docs.opencv.org/4.x/d4/da8/group__imgcodecs.html>

**Virtual Resources:**

<https://github.com/RRL-ALeRT/spot-webots-cloud/tree/main>

<https://github.com/MASKOR/webots_ros2_spot>

<https://www.researchgate.net/publication/322594373_Inverse_Kinematic_Analysis_of_a_Quadruped_Robot>

<https://cyberbotics.com/doc/guide/index>

<https://github.com/dpilger26/NumCpp>

<https://github.com/LucaVits/LearningROSWithPythonTurtle/blob/main/ROS2/WindowsInstallation/ROS2IronInstallationWindows.ipynb>

# **Finding and Results**

Throughout this Capstone project, significant progress was made in both the physical and virtual realms of the robotic system. By addressing challenges in robot control, movement optimization, and system integration, we achieved milestones that advance the project's goals.

**Robot Performance and Control**

Testing on the DOGZILLA robots, Charlotte and Charlie, highlighted key differences in movement speed and precision across various modes. Charlotte consistently outperformed Charlie, with lower average times in normal, low, and high-speed modes. Both robots exhibited slight drifts during motion, with adjustments required to improve directional stability. Movement control relied on position-based adjustments (translation and rotation) and inverse kinematics for coordinated leg movements, resulting in stable navigation. Key motor functions and auto-stabilization via the IMU provided reliable performance in diverse scenarios.

**LiDAR Integration**

LiDAR sensor testing demonstrated its utility for creating sanitized visualizations and environmental mapping. However, setting up LiDAR for short-term use in the AI and Machine Learning course posed challenges due to the complexity of environment configuration and scripting. LiDAR data handling scripts were centralized within the project repository for streamlined access.

**Maze Design**

The team developed a cost-effective, modular maze system using 3D-printed polyresin molds and cardboard components. This approach allows for dynamic maze creation and easy reconfiguration, enabling consistent testing environments for navigation challenges. The design efficiently balances resource constraints with functional flexibility.

**Simulation and Virtual API Development**

In the virtual environment, we adapted and refined the Webots-based quadruped simulation, incorporating a modified API to align with DOGZILLA’s capabilities. The simulation now includes an inverse kinematics solver and a gait trajectory solver, enabling realistic and smooth movements. The virtual robot, Spot, can navigate in multiple directions and has preliminary functionality for vertical traversal, such as climbing stairs. A comprehensive command API has been developed, mirroring DOGZILLA’s functionality, with additional capabilities for testing advanced movement scenarios.

**Unified Vision**

By merging efforts from both physical and virtual teams, we established a robust framework for testing and deploying robotic capabilities. The physical systems benefitted from insights gained through simulation, while the virtual systems provided a safe testing ground to refine algorithms before real-world implementation. Together, these results create a cohesive platform for future exploration and educational use.

# **Next Steps (for Spring 2025)**

The next phase of this project, scheduled for Spring 2025, focuses on advancing both physical and virtual systems toward a cohesive demonstration in March. Key tasks include refining navigation, enhancing interoperability, and integrating capabilities between the physical DOGZILLA robot and its virtual counterpart.

**Maze Navigation and QR Code Integration**

We will finalize the design and construction of additional maze components and implement scripts for autonomous navigation. Using Python scripts in Jupyter Notebooks or ROS2 with Webots, we aim to enable DOGZILLA to identify QR codes within the maze, interpret their content, and map the environment. Specific tasks include:

* **QR Code Recognition and Tracking:** The robot will read QR codes, store their content, and log distances between nodes, creating a mental map of the maze.
* **Action Execution:** QR codes will trigger specific behaviors, such as navigation to designated nodes.
* **Synchronization:** Ensure physical and virtual QR codes are identical for consistency across platforms.

**Refining Robotic Control and Features**

The physical robot will undergo further testing to enhance its movement and recognition abilities. Goals include:

* Linking sections of code to achieve seamless object tracking and distance recording.
* Enabling the robot to recognize QR codes and adjust actions accordingly.
* Improving basic and advanced movements, such as navigating tight corners and adjusting to dynamic obstacles.

**Virtual Simulation Enhancements**

Efforts will continue to enhance the Webots simulation, with a focus on creating an accurate model of DOGZILLA that matches its physical dimensions and functionality. Key improvements include:

* Developing a virtual model aligned with DOGZILLA’s scale and joint mechanics.
* Adjusting controller parameters, such as joint limits and leg lengths, to optimize motion simulation.
* Enhancing the API to unify code execution across physical and virtual platforms, including potential development of a C++ driver for improved interoperability.

**Testing and Interoperability**

Rigorous testing of code designed for Spot on DOGZILLA, and vice versa, will ensure seamless functionality across platforms. This includes:

* Validating code compatibility between Webots and DOGZILLA.
* Using tools like calipers and photogrammetry software to improve the precision of virtual models.
* Developing workflows, such as bash scripts, to streamline code transitions between simulation and physical environments.

By aligning these tasks across physical and virtual domains, we aim to present a unified demonstration that highlights DOGZILLA’s capabilities and showcases the integration of advanced robotics and simulation.