NgTrDat@cv-terminal:~

NgTrDat | Wed 8 Apr - 17:05 | ~ fastfetch -c $HOME/.config/fastfetch/CV.jsonc

NgTrDat@cv-terminal

Name : Nguyễn Trọng Đạt
Age-Gender : 22 - Male
Role : Computer Engineering Intern
Study : Ho Chi Minh University of Technology
Focus : Embedded Systems • Robotics • AI • Drones
Location : Ho Chi Minh City, Vietnam
Email : trongdat4work@gmail.com
Languages : C/C++, Python, Shell script
Tools : VSCode, Git, ROS2, Gazebo
English : TOEIC L&R 970/990

NgTrDat | Wed 8 Apr - 17:06 | ~ ./script/ShowMyCV.sh

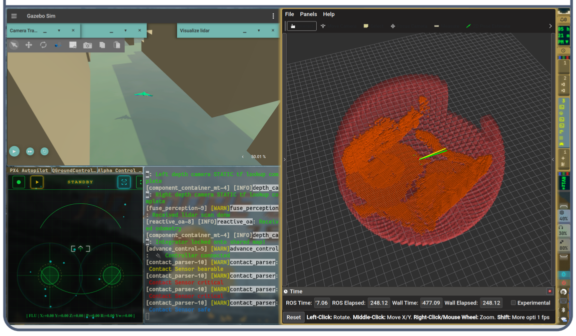
Professional_objective.txt

•Computer Engineering student focused on robotics, embedded systems, and autonomous drone software. Experienced with ROS2, PX4, Gazebo, and C++ development for real-time systems and simulation environments.

•Experienced in collaborative development environments using Git; proficient at reading, understanding, and contributing safely to shared codebases.

•Seeking to transition into a full-time engineering role after the internship.

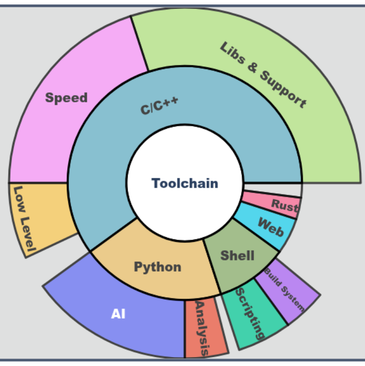
Drone_Simulation_Preview.png



Education.log

[Study] Ho Chi Minh University of Technology (BKU)
[Major] Bachelor of Engineering in Computer Engineering (Expected graduation 2027)
[GPA] 2.5/4.0 *
[Core Coursework] • Embedded System • Operating System • Machine Learning
• Microprocessors-Microcontrollers
• IoT-Oriented Logic Design Project
• AI-Oriented Multidisciplinary Project
[Passion Project] Developed a ROS2 + PX4 drone simulation pipeline in Gazebo for acrobatic flight imitation learning.**

Toolchain.png



Skills.md

Programming:
• Most proficient with C++ and Python
• Familiar with Rust, Java, Go and other system-level languages
Robotics & Middleware ecosystem:
• ROS2 (Jazzy) • Gazebo Harmonic
• PX4 Autopilot • MicroXRCE-DDS
AI, Machine Learning and Data Engineering:
• PyTorch • Robotic data processing • Computer Vision
• Latent Diffusion Learning • AI Edge Computing
Development tools: • VScode • Git • Linux OS

Contacts.txt

Phone : (+84) 399922395
Email : trongdat4work@gmail.com
Github: github.com/LemonKronos
LinkIn: www.linkedin.com/in/trong-dat

About_me.txt

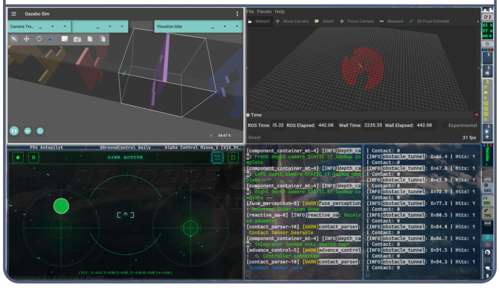
- My goal is to become a Robotics Engineer
- Open-source mindset, member of Linux community
- Strong technical communicator, ready to collaborate in international teams

Project_Acrobatic_Drone.md

Project: Full-Pipeline Acrobatic Drone based on Latent Imitation Learning

- **Latent Imitation Learning:** Trains an AI model to imitate expert pilot behavior using latent space representations and performance-based evaluation, enabling policies that can potentially surpass the original demonstrations.
- **Goal:** framework designed to act as an "AI Co-pilot," denoising erratic, suboptimal human inputs into safe, expert-level 3D trajectories.
- **Main components:** Gazebo Harmonic for simulation and physics, PX4 Autopilot for high-frequency flight control (400Hz), Micro XRCE-DDS for PX4-ROS2 communication, and ROS2 as the offboard middleware handling sensors, messaging, and AI integration (80Hz). The acrobatic AI runs as a Python ROS2 node at a lower control frequency (0.5Hz).
- **Principle:** Inputs include raw human control commands, telemetry data, and environmental perception data (vision and 3D sensing). The AI generates an intended trajectory through the environment, which is validated by a safety and obstacle-avoidance layer before being executed by PX4 and translated into motor commands.

Test_non-AI_behavior.png



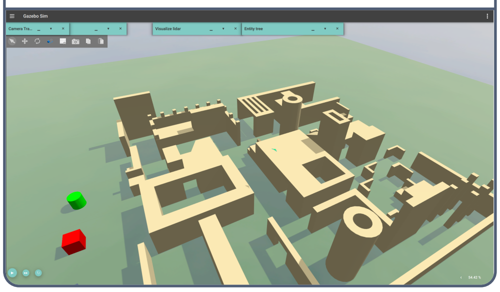
AI_train.md

Layer: 12
Total params:88.5M
Estimated size: 354MB
Dataset: 180MB

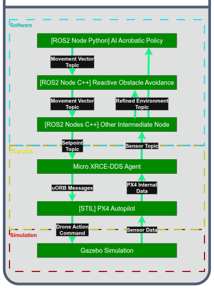
Project_Acrobatic_Drone.md

- **Execution:** Collected expert control inputs and telemetry data in Gazebo to build a training dataset. The simulation environment consisted of a procedurally generated obstacle-tunnel world. Models were trained offline and evaluated through both offline testing and real-time user-controlled flight scenarios.
- **Results:** Successfully validated the non-AI control pipeline, including low-level obstacle avoidance, achieving autonomous flights of up to 1 km without collisions. However, the AI-controlled system produced noisy outputs and occasionally conflicted with user inputs, resulting in unstable flight behavior. Simulated latency average 290us
- **Diagnosed:** Early model outputs revealed limitations in the design and data flow between high and low layers, indicated the need for an improved model and larger training datasets.
- **Future:** Currently redesigning the pipeline with a focus on low-level data processing, system integration, and control structures to support more dynamic acrobatic maneuvers.
- **Summarize:** Demonstrated the feasibility of the overall control pipeline. Further refinement of both the AI and robotics architectures is needed.

Test_Control_World.png



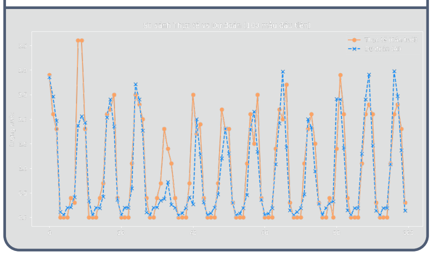
Pipeline.png



Edge_AI_Greenhouse_Project.md

Goal: Train a regression model to predict the correct irrigation amount for a greenhouse, and deploy it on an edge IoT device.
Principle: In Python, build a LightGBM regression model on the dataset, prune unimportant features. After that, using TL2cgen libabry, convert the model into C code, then compile it to binary and run directly on an ESP32. On the Embedded side, ESP32 handles sensor filtering and formating, communication, safeguards and sleep scheduling.

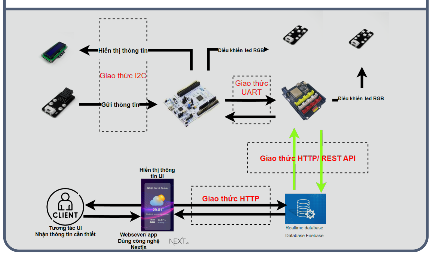
Irrigation_Comparison.png



Simple_IoT_Projects.md

Goals: Integrate multiple sensors using STM32, then evaluate and display sensor data using LCD, LEDs indicator and send data to web server via ESP32 wifi module.
Execution: Read datasheets and developed driver for sensors, LCD, LEDs. Implemented I2C communication between sensors and STM32, and UART communication between STM32 and ESP32. Set up RTOS scheduler on STM32. Developed and deployed a web server using Next.js

IoT_Diagram.png



Academic_vs_Practical_Metrics.log

- **Reality:** While my GPA is 2.5/4.0, I have focused heavily on researching, designing, and building real-world engineering systems beyond the classroom.
- **Evidence:** Classroom scores measure test retention, but engineering requires execution. The projects documented above demonstrate my ability to develop functional software and embedded systems. All source code is available on GitHub, and I am available to provide a live demonstration upon request.