InternPro Weekly Progress Update

Name	Email	Project Name	NDA/ Non-NDA	InternPro Start Date	OPT
Adharsh Prasad Natesan	anatesan@asu.edu	IT-Core Foundation Suriname	Non-NDA	2024-08-05	Yes

Progress

Include an itemized list of the tasks you completed this week.

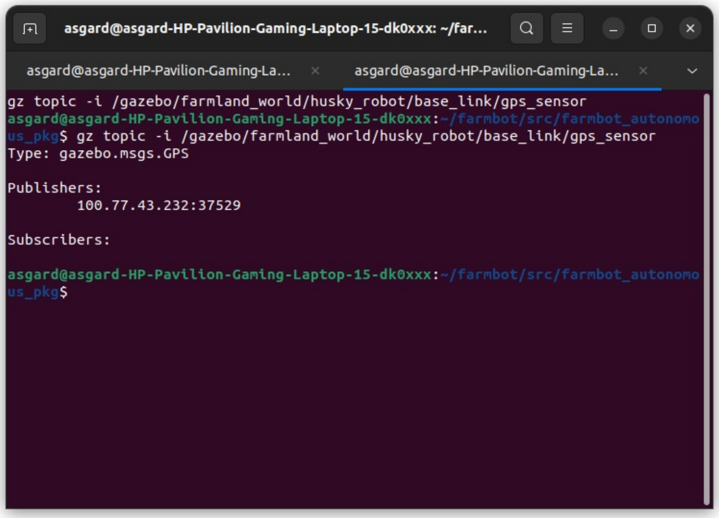
#	Action Item/ Explanation	Total Time This Week (hours)
1	Adding GNSS Sensor to Husky in Gazebo	3
2	Visual Representation of GNSS Receiver on Husky	3
3	Simulated GPS Integration Using Custom ROS2 Publisher	3
4	GNSS Base Station Antenna Integration in Simulation Environment	3
5	Integrating Reinforcement Learning and Large Language Models for Crop Management Decision Support Systems	3
6	Experimental Evaluation of a Hierarchical Operating Framework for Ground Robots in Agriculture	3
7	Control, decision-making, and preparing the system for autonomy	1
8	Report Writing	1
	Total hours for the week:	20

Verification Documentation:

Action Item 1: Adding GNSS Sensor to Husky in Gazebo – 3 hour(s).

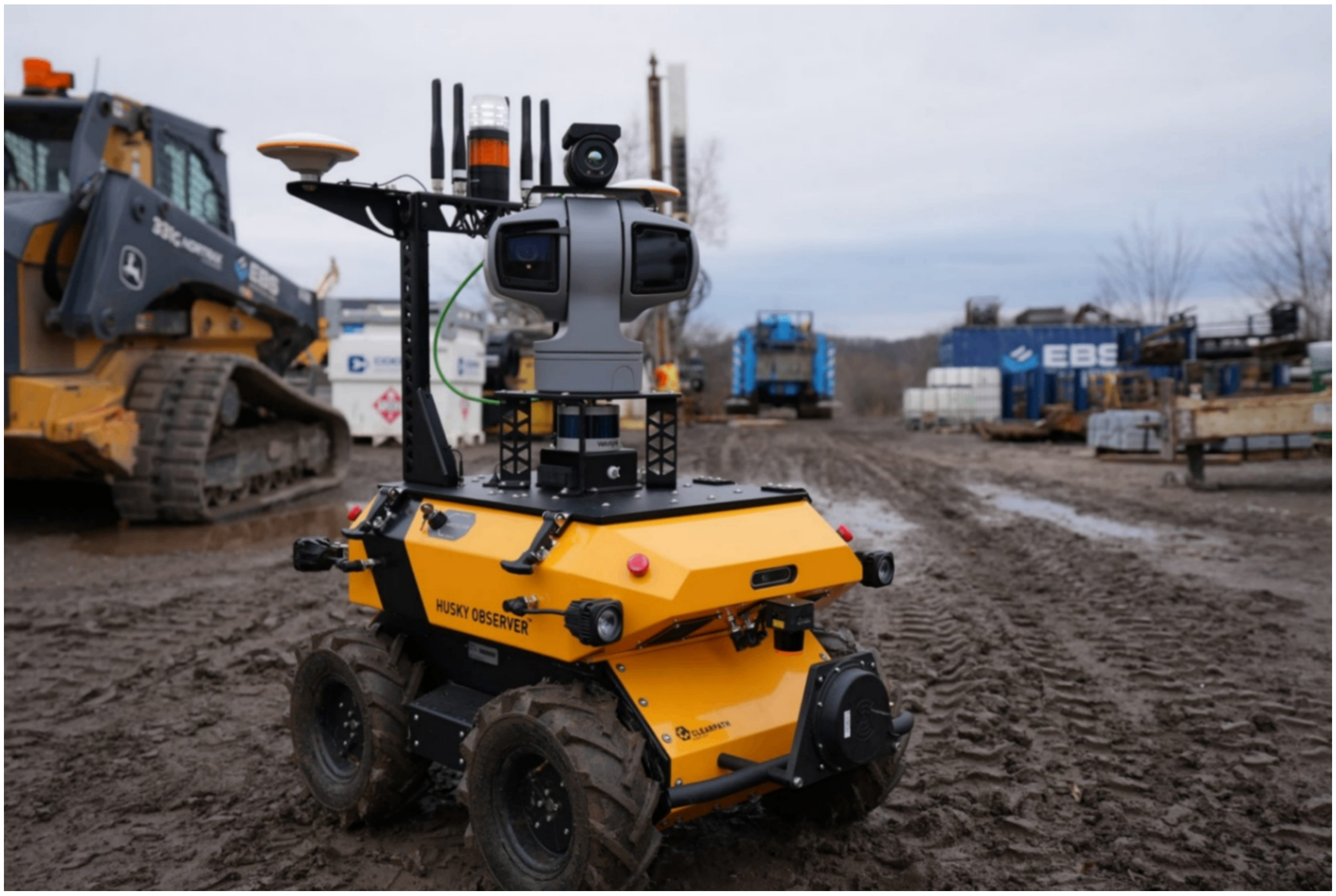
Project Work Summary

- Implemented a GNSS receiver in the Husky simulation to simulate satellite-based positioning as part of the state estimation pipeline for precision farming. The GPS sensor was introduced into the base model to test standalone GNSS output before sensor fusion with other modalities like visual odometry.
- Defined a <sensor type="gps"> block within the Husky's base_link, configuring it with realistic Gaussian noise in both horizontal and vertical directions. The sensor had a fixed update rate of 5 Hz and a mounting height of 1 meter above the base link.
- Since gnss was worked with an SDF-based Husky model, the sensor was embedded directly into the existing .sdf file. Verified that Gazebo was publishing GNSS readings under its internal topic structure (gazebo.msgs.GPS) by inspecting /gazebo/<world>/<robot>/base_link/gps_sensor.
- Although the sensor did not natively publish to ROS 2's /gps/fix, it was confirmed to function within Gazebo via gz topic -l. The implementation served as a working baseline for GNSS data generation ahead of ROS2-compatible translation.
- After defining the GNSS sensor in the Husky's .sdf, I validated its functionality using native Gazebo Transport tools. Used gz topic -l to identify the GPS publishing topic (/gazebo/farmland_world/husky_robot/base_link/gps_sensor) and confirmed the message type with gz topic -i, which revealed gazebo.msgs.GPS.
- Echoed the topic using gz topic -e to inspect real-time GNSS output including latitude, longitude, and velocity. This ensured that the sensor was actively publishing spatial data in Gazebo, even though it was not natively available to ROS2.
- This task was part of a broader setup to simulate GNSS data in the context of sensor fusion and was intended to be followed by bridging or emulation to match ROS 2's expected NavSatFix message format.


asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx: ~/far...
asgard@asgard-HP-Pavilion-Gaming-La...
asgard@asgard-HP-Pavilion-Gaming-La...

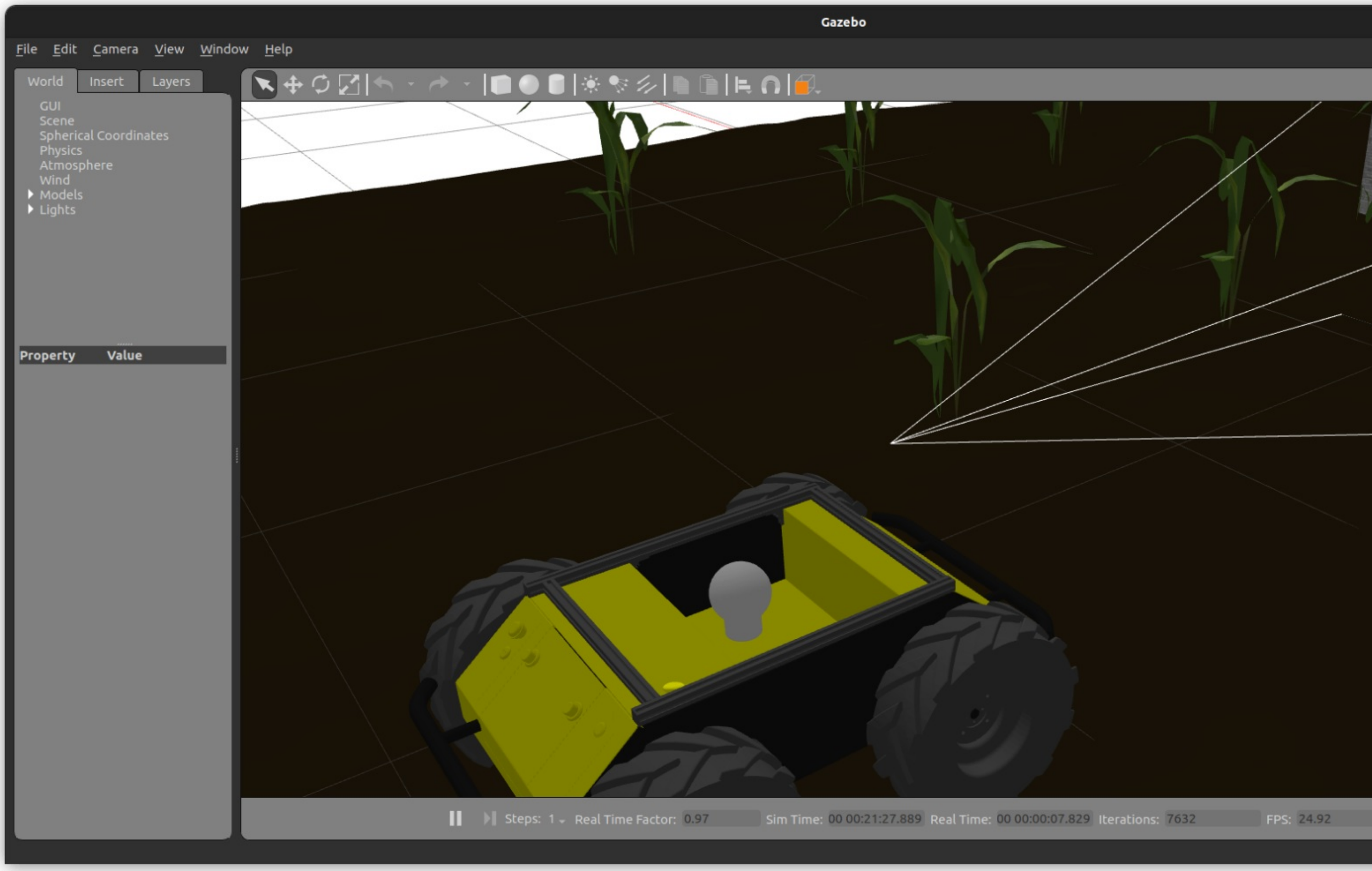
```
gz topic -i /gazebo/farmland_world/husky_robot/base_link/gps_sensor
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/farmbot/src/farmbot_autonomo
us_pkg$ gz topic -i /gazebo/farmland_world/husky_robot/base_link/gps_sensor
Type: gazebo.msgs.GPS

Publishers:
        100.77.43.232:37529

Subscribers:

asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/farmbot/src/farmbot_autonomo
us_pkg$
```

Action Item 2: Visual Representation of GNSS Receiver on Husky – 3 hour(s).



Project Work Summary

- I wanted to create a physical representation of the GNSS receiver mounted on the Husky, mainly for visual clarity during simulation and documentation. Although the GPS sensor in Gazebo works functionally without any visual geometry, it felt incomplete not having a visible antenna on the robot—especially when taking screenshots or analyzing sensor layouts.
- To guide the design, I spent some time searching online for real-world examples of GNSS receivers mounted on mobile robots—particularly Husky UGV setups. Most setups featured a compact dome-like antenna mounted on a short vertical base, typically centered above the robot's chassis.
- Using that reference, I added a visual structure directly to the base_link of the robot. I started with a small white cylinder as the mounting base, then stacked a white sphere above it to represent the GNSS antenna dome. The two shapes combined to closely match the compact, survey-grade GPS modules seen in field robots.
- I adjusted the vertical pose values so the visuals sat neatly above the robot, aligned with the simulated GPS sensor. For materials, I used Gazebo/White to match the real-world modules and ensure clear visibility against the Husky's darker frame. I kept the geometry simple and lightweight to avoid affecting rendering performance.
- While this addition was purely cosmetic, it made the sensor configuration much easier to understand at a glance. It helped during simulation reviews, added realism for documentation, and contributed to a clearer visual story around the robot's perception system.

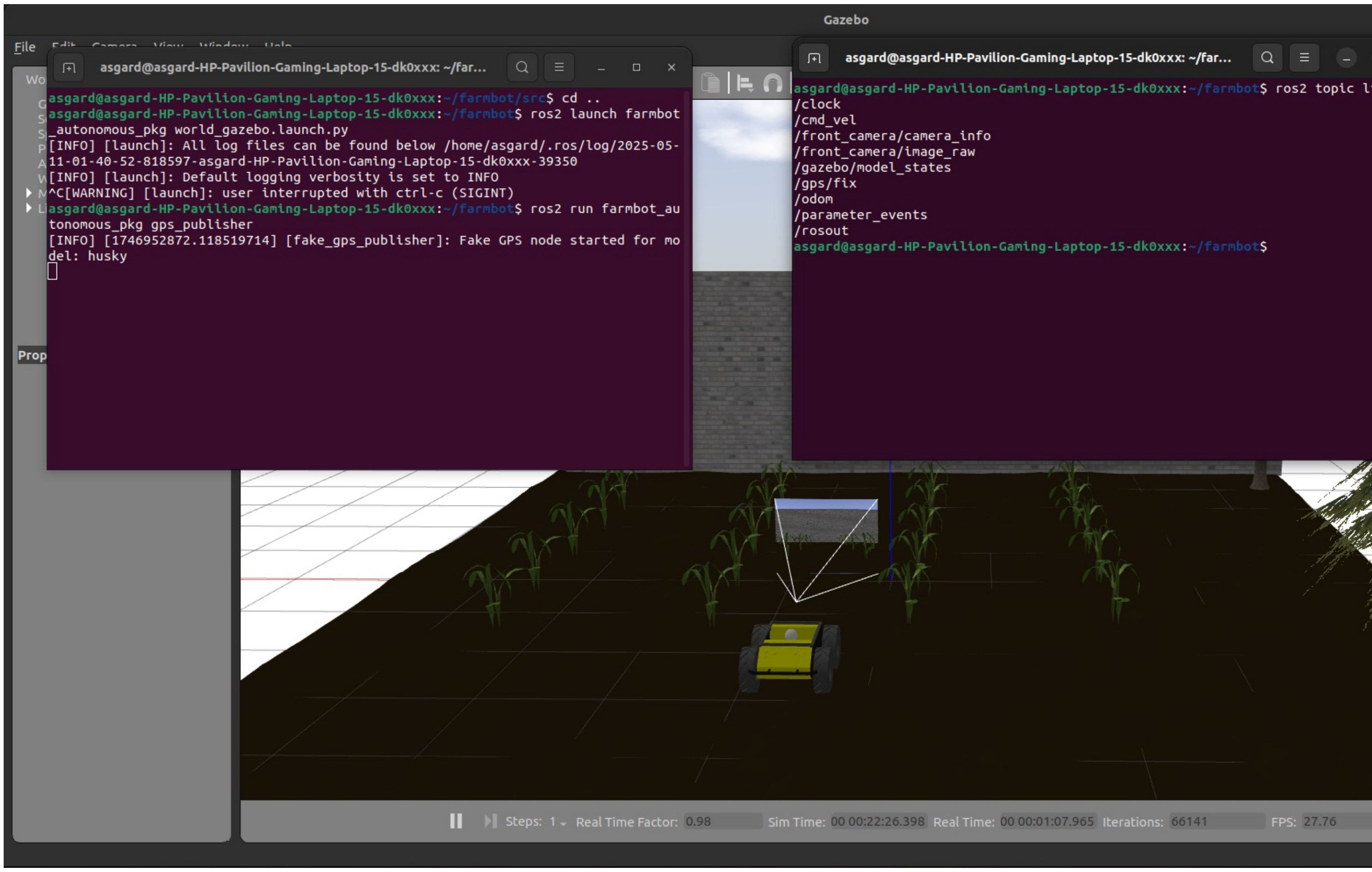


-
-

Action Item 3: Simulated GPS Integration Using Custom ROS2 Publisher – 3 hour(s).

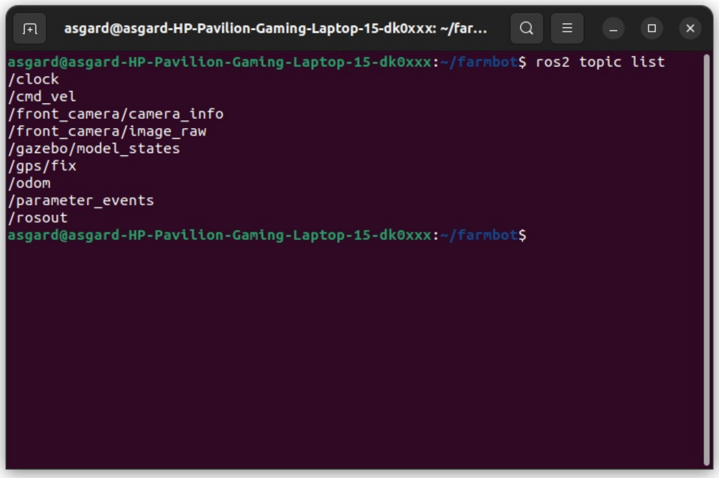
Project Work Summary

- Developed a ROS2 node (gps_publisher.py) to simulate GPS data for the Husky robot in Gazebo, since the built-in GPS sensor doesn't publish sensor_msgs/NavSatFix without the libgazebo_ros_gps.so plugin.
- The node listens to /gazebo/model_states, extracts the Husky's current pose, and transforms it into latitude and longitude using a scaled origin point, adding Gaussian noise to simulate real-world sensor drift.
- Manually published to the /gps/fix topic, defining position covariance and frame information to match NavSatFix message requirements, allowing it to integrate seamlessly with localization pipelines.
- Initially, the GPS plugin approach was explored via <plugin filename="libgazebo_ros_gps.so">, but it failed with a shared library error — the .so file wasn't present in the Humble binary and couldn't be installed via apt.
- Attempted to build gazebo_ros_pkgs from source to obtain the missing plugin, but the dependency tree was large and the build failed due to environment mismatches and missing symbols.
- The workaround was to fully drop the GPS plugin idea and rely solely on our custom publisher node as a temporary but stable solution for simulating GNSS.
- During testing, found that although /gazebo/model_states appeared in the topic list after launching the node, it never published any data — which meant the callback in the GPS node was never triggered.
- After verifying the .world file and plugin declarations were correct, the issue was finally traced to the missing <static>false</static> flag in the Husky model's root tag, which caused Gazebo to simulate it as a non-dynamic object.
- Confirmed the robot was inserted correctly, teleop commands worked, but movement alone wasn't enough — the model needed to be flagged as dynamic to emit state updates.
- We're now refocusing efforts on getting dynamic publishing to work properly, so that /gazebo/model_states reliably feeds into the GPS node — this is essential before layering any EKF, VO, or navigation logic on top.


Gazebo
File
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx: ~/far...
Prop
Steps: 1
Real Time Factor: 0.98
Sim Time: 00 00:22:26.398
Real Time: 00 00:01:07.965
Iterations: 66141
FPS: 27.76

```
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/farmbot/src$ cd ..
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/farmbot$ ros2 launch farmbot
_autonomous_pkg world_gazebo.launch.py
[INFO] [launch]: All log files can be found below /home/asgard/.ros/log/2025-05-
11-01-40-52-818597-asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx-39350
[INFO] [launch]: Default logging verbosity is set to INFO
^C[WARNING] [launch]: user interrupted with ctrl-c (SIGINT)
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/farmbot$ ros2 run farmbot_au
tonomous_pkg gps_publisher
[INFO] [1746952872.118519714] [fake_gps_publisher]: Fake GPS node started for mo
del: husky
```

```
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/farmbot$ ros2 topic l
/clock
/cmd_vel
/front_camera/camera_info
/front_camera/image_raw
/gazebo/model_states
/gps/fix
/odom
/parameter_events
/rosout
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/farmbot$
```

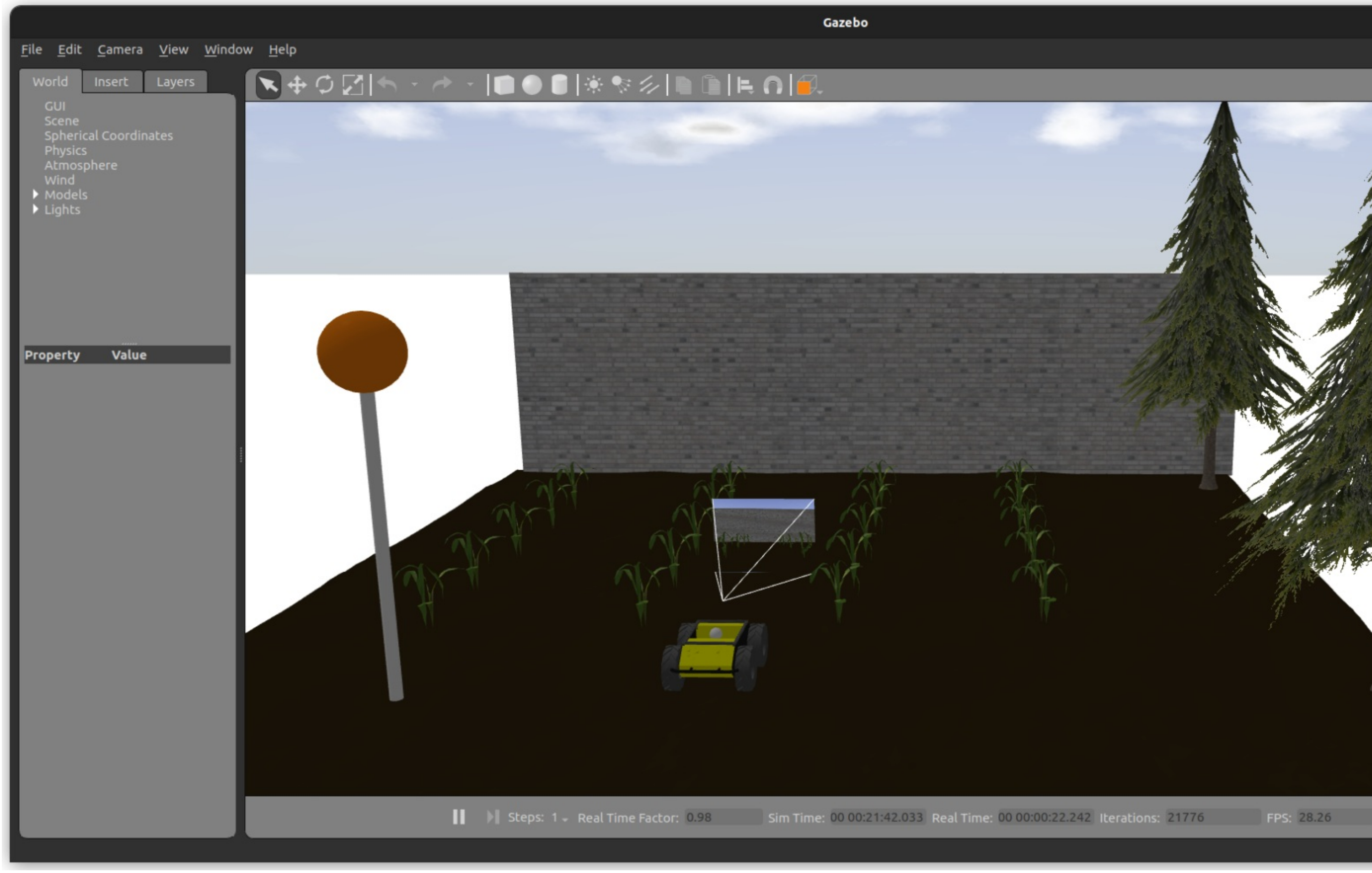


```
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/farmbot$ ros2 topic list
/clock
/cmd_vel
/front_camera/camera_info
/front_camera/image_raw
/gazebo/model_states
/gps/fix
/odom
/parameter_events
/rosout
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/farmbot$
```

Action Item 4: GNSS Base Station Antenna Integration in Simulation Environment – 3 hour(s).

Project Work Summary

- I added a static GNSS antenna model directly to the Gazebo world to visually represent a fixed base station, often used in RTK-enabled localization setups. The purpose was to include a physical reference marker within the environment, even though the antenna itself wouldn't emit any correction data or perform sensing.
- This model acts purely as a structural element to help with visualization, documentation, and to support future simulation features such as base-rover localization tests.
- The antenna was created using two simple geometric primitives: a narrow white cylinder as the support pole and a hemisphere-shaped white sphere on top to mimic the dome of a real GNSS antenna. I adjusted the <pose> to lift the components slightly above ground and placed the entire model at a fixed position inside the world coordinate frame using the <model> and <static> tags.
- To keep it non-interactive and light, I didn't attach any plugin or sensor to this model. The antenna does not publish or subscribe to any topics — it's intended purely for structural realism and as a visual landmark for use in overhead maps and renders.
- Later, I could optionally add a minimal publisher node (e.g., broadcasting a static PoseStamped as a known RTK anchor), but for now, the task was focused solely on the visual integration.
- This antenna model gives us a cleaner way to demonstrate GNSS-aware rover behavior in the field layout and helps reinforce a ground-truth concept when integrating visual odometry or sensor fusion pipelines later on.



Steps: 1 Real Time Factor: 0.98 Sim Time: 00 00:21:42.033 Real Time: 00 00:00:22.242 Iterations: 21776 FPS: 28.26

-
-

Action Item 5: Integrating Reinforcement Learning and Large Language Models for Crop Management Decision Support Systems – 3 hour(s).

Research

- https://arxiv.org/pdf/2410.09680
- Integrating Reinforcement Learning and Large Language Models for Crop Management Decision Support Systems
- Summary of Report
  - This paper explores the synergistic integration of Reinforcement Learning (RL) and Large Language Models (LLMs) to enhance decision-making in crop management.
  - The authors propose a framework where RL algorithms learn optimal strategies through interactions with the environment, while LLMs provide contextual understanding and reasoning capabilities.
  - The combination aims to address the complexities and uncertainties inherent in agricultural decision-making, such as varying weather conditions, pest infestations, and soil health.
  - The paper proposes a novel architecture that combines RL's adaptability with LLMs' contextual reasoning to form a comprehensive decision support system (DSS) for agriculture.
  - Development of a simulated agricultural environment to train and evaluate the integrated system, allowing for safe and efficient testing of various scenarios.
  - The author also implemented a framework for scenarios like irrigation scheduling and pest control, demonstrating improved decision-making compared to traditional methods.
  - Introduction of performance metrics to assess the effectiveness of the integrated system, including yield improvement, resource utilization, and adaptability to changing conditions.
- Relation to Project
  - Enhanced Decision-Making: The integration of RL and LLMs can improve our system's ability to make informed decisions in real-time, adapting to new data and unforeseen circumstances.
  - Resource Optimization: By learning optimal strategies, the system can better manage resources like water and fertilizers, aligning with our goals of medium scale farming.
  - Scalability: The modular nature of the proposed framework allows for scalability, enabling us to extend the system to various crops and farming conditions.
  - Risk Mitigation: The simulation environment provides a platform to test and refine strategies before real-world deployment, reducing the risk of crop failure due to suboptimal decisions.
- Motivation for Research
  - Our current system is good in data acquisition through GNSS and camera sensors but lacks advanced decision-making capabilities. This paper offers insights into integrating adaptive learning and contextual reasoning.
  - Autonomous Operations: Enabling the system to operate independently with minimal human intervention.
  - Dynamic Adaptation: Allowing the system to adjust strategies based on real-time data, improving resilience to environmental changes.
  - Intelligent Control: Facilitating the transition from data collection to actionable insights, leading to more effective control mechanisms.

Action Item 6: Experimental Evaluation of a Hierarchical Operating Framework for Ground Robots in Agriculture – 3 hour(s).

Research

- https://arxiv.org/abs/2105.10845
- Experimental Evaluation of a Hierarchical Operating Framework for Ground Robots in Agriculture
- Summary of Report
  - This study presents a hierarchical operating framework designed to enhance the autonomy of ground robots in agricultural settings. The framework is structured into three layers:
    - Strategic Layer: Handles high-level planning, including task allocation and route planning based on global objectives.
    - Tactical Layer: Focuses on decision-making in response to dynamic environmental factors, such as obstacle avoidance and task prioritization.
    - Operational Layer: Manages low-level control, ensuring precise execution of tasks like navigation and manipulation.

    - The authors implemented this framework on the SwagBot platform and conducted field trials to assess its performance in real-world agricultural tasks, including weeding and crop monitoring.
    - The paper proposes three distinct ways in which it excels in the automation of the agricuture. Improved Autonomy: The hierarchical approach enabled the robot to perform complex tasks with minimal human intervention. Adaptability: The system demonstrated the ability to adapt to changing field conditions and unexpected obstacles. Efficiency: Task execution was more efficient compared to traditional control systems, with reduced time and resource consumption.
- Relation to Project
    - The paper structured a control strategy for the agriculture and by implementing a similar layered control system can enhance our robot's ability to handle complex tasks systematically.
    - The main advantage of the paper is the real world applicability and the successful field trials provide a validated approach that we can adapt and apply to our specific agricultural context.
    - Integration with Existing Systems: The framework can be integrated with our current GNSS and camera systems, providing a comprehensive solution from data acquisition to action execution.
- Motivation for Research
    - Our project requires a control system that can manage multiple tasks and respond to dynamic environments. This paper offers 3 practical solutions and integrate it to the real world.
    - Task Management is the strategic layer that would help in planning and allocating tasks efficiently.
    - Responsive Control layer is tactical and operational layers ensure the system can react appropriately to real-time changes, maintaining operational integrity.
    - Scalability and Flexibility through the modular nature of the framework allows for easy adaptation to different tasks and environments, which is essential for diverse farming operations.

Action Item 7: Control, decision-making, and preparing the system for autonomy – 1 hour(s).

Project Work Summary

- Finalize the integration of the visual GNSS base station antenna in the farmland world and commit it as a reusable static model for all future map tests. Capture a clean screenshot showing its position for future documentation and EKF ground truth alignment references.
- Refactor the working `gps_publisher.py` into a modular ROS2 node with clear parameters for GNSS noise level, publishing rate, and origin, and create a corresponding launch file for consistency across test runs.
- Begin prototyping the **task control layer** for the autonomous rover using a decision tree or FSM framework. Start with a basic mode switcher between idle, move-to-goal, and stop-on-obstacle, and implement it as a ROS2 node that listens to GNSS and mock task goals.
- Begin wiring up a reward or success-monitoring mechanism to simulate the feedback loop needed for RL or rule-based decision-making. For now, use position error thresholding (i.e., goal reached if within 1.0m of target) and log results to CSV for later comparison.
- Prepare a placeholder LLM integration node (even if not yet connected to a real model) that receives task context and returns mock decisions. This can be as simple as mapping text input like “weed row 3” into a waypoint goal or state switch in the FSM.

Action Item 8: Report Writing – 1 hour(s).

Project Work Summary

- Created word document layout to write contents of the weekly progress.
- Created relevant subsections in the epicspro website and documented 20 hours of weekly progress.
- Collected relevant documents research papers, relevant links and company's objective from their portal.