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	Robust Multi-Sensor Fusion for Localization in Hazardous Environments Using Thermal, LiDAR, and GNSS Data	3
2	Testing Foundational Sensors: VO, GNSS, and Visual-Based Positioning	1
3	Debugging Visual Odometry and GNSS Integration	3
4	Reinstalling Gazebo and Integrating an External GPS Publisher	3
5	Creating a Simulated GNSS Bridge Node from Gazebo Model States	3
6	GNSS Integration Pipeline for ROS2-Based Mobile Robots	3
7	Decision-Making with GNSS, Visual Odometry, and YOLO	3
8	Report writing	1
	Total hours for the week:	20

Verification Documentation:

Action Item 1: Robust Multi-Sensor Fusion for Localization in Hazardous Environments Using Thermal, LiDAR, and GNSS Data – 3 hour(s).

Research

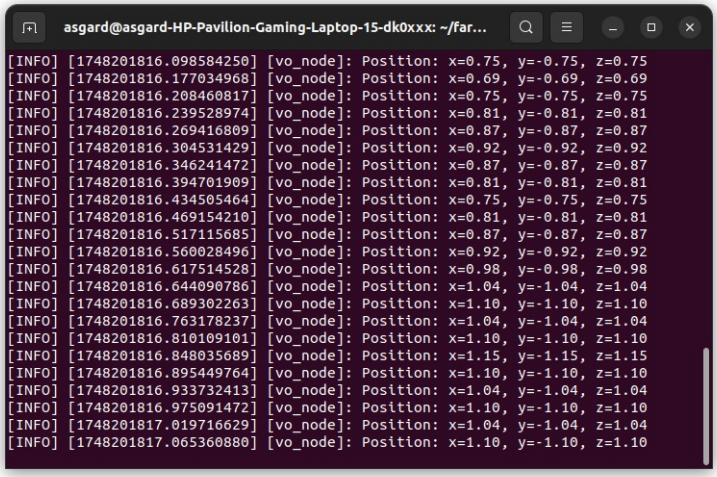
- https://doi.org/10.3390/s25072032
- Robust Multi-Sensor Fusion for Localization in Hazardous Environments Using Thermal, LiDAR, and GNSS Data
- Summary of Report
  - This paper presents a robust multi-sensor localization strategy for autonomous vehicles operating in unpredictable and harsh environments by fusing data from GNSS, LiDAR, and thermal cameras.
  - The proposed framework centers around an Extended Kalman Filter (EKF) to integrate diverse sensory data with varying update rates, improving overall positioning reliability.
  - The authors place particular emphasis on sensor failure handling — dynamically weighting sensors based on availability and trustworthiness, which is critical when one or more inputs (e.g., GNSS or visual) become unreliable.
  - They provide detailed insights into the sensor calibration process, the importance of maintaining consistent coordinate frames, and the delay compensation strategies required to synchronize incoming data streams.
  - Real-world validation in urban and semi-structured environments demonstrates substantial reduction in position error and drift, especially in GNSS-degraded zones such as alleyways and underpasses.
- Relation to Project
  - My autonomous farming rover project involves integrating GNSS and visual odometry, and this paper's robust fusion strategy gives me a clear blueprint for making my localization pipeline resilient to partial sensor failure.
  - The emphasis on EKF design and real-time adaptability will be directly useful as I implement fusion in ROS2 using robot_localization, particularly for switching or reweighting VO and GNSS inputs based on trust scores or drift.
  - Their approach to delay compensation and asynchronous sensor update handling offers practical insights for tuning my node configurations and launching transforms at appropriate frequencies.
  - Even though I don't use thermal or LiDAR, their modular design shows how different sensing modalities can plug into a generalized fusion pipeline. I can treat my YOLO-based distance estimates as a "soft observation" layer in future iterations.
- Motivation for Research
  - I'm trying to build a cost-effective, accurate, and robust localization system for small-to-medium scale farming robots where GNSS can occasionally glitch and VO can drift and this paper addresses both pain points directly.
  - My existing VO + GNSS setup hasn't been fused yet, and this study provides both a validation of EKF-based integration and implementation-level insights for aligning timestamps and reference frames, the key issues I've been debugging.
  - I want to extend this to a real-world deployable system later, so the focus on handling partial observability, reliability scoring, and dynamic environment challenges prepares me to future-proof the design.
  - This paper complements earlier theoretical studies by grounding the fusion approach in real sensor imperfections and real-time recovery — exactly what I need to push my project forward.

Action Item 2: Testing Foundational Sensors: VO, GNSS, and Visual-Based Positioning – 1 hour(s).

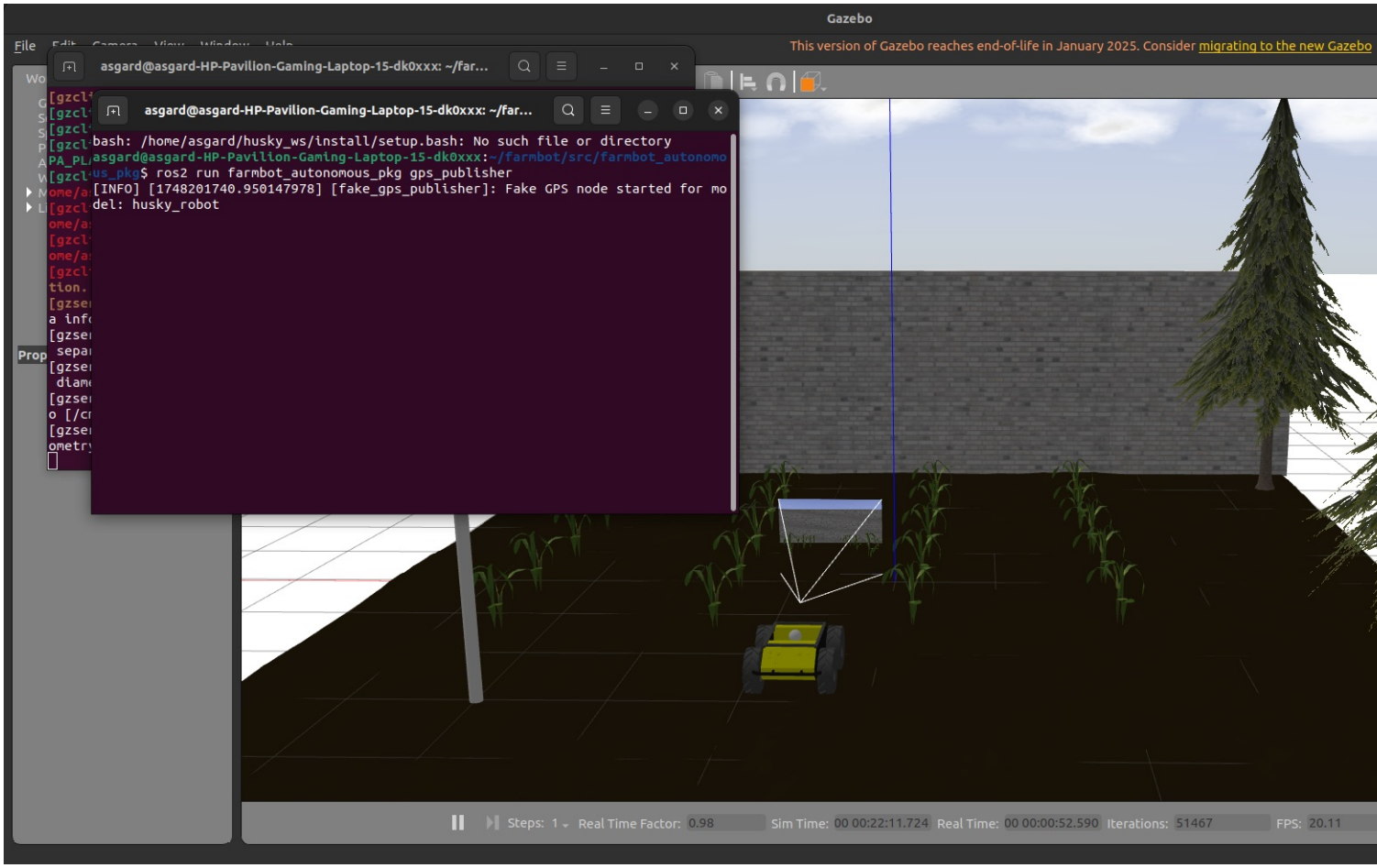
Project Work Summary

- After rebuilding several core components of the FarmBot simulation pipeline, I wanted to validate that the foundational perception systems were all operational before moving into debugging or fusion. The primary goal of this task was to ensure that visual odometry, GNSS simulation, and any visual-based positioning layers were correctly publishing their respective outputs, and to get a sense of how they drift or behave when isolated.
- To start, I launched my custom world and Husky robot model inside the farmland simulation using the standard farmland_launch.py. This launch brings up the robot along with the VO node, my GNSS publisher, and the camera feed. I visually confirmed that all nodes came up cleanly in the terminal without error.
- Once the system was running, I opened up three terminal sessions to monitor the key output topics in real time /gps/fix for GNSS position, /vo/pose or /vo/odom for visual odometry and Any camera-based estimated position (like feature-matched pose or fused localization if active).

- With these active, I first let the robot sit idle in the field to evaluate baseline drift. This was particularly useful for identifying if visual odometry falsely perceives motion when the camera feed is static — which, in my case, it did. GNSS readings remained mostly steady, as expected, but the VO stream showed positional drift even when the robot wasn't moving.



-



-
-

Action Item 3: Debugging Visual Odometry and GNSS Integration – 3 hour(s).

Project Work Summary

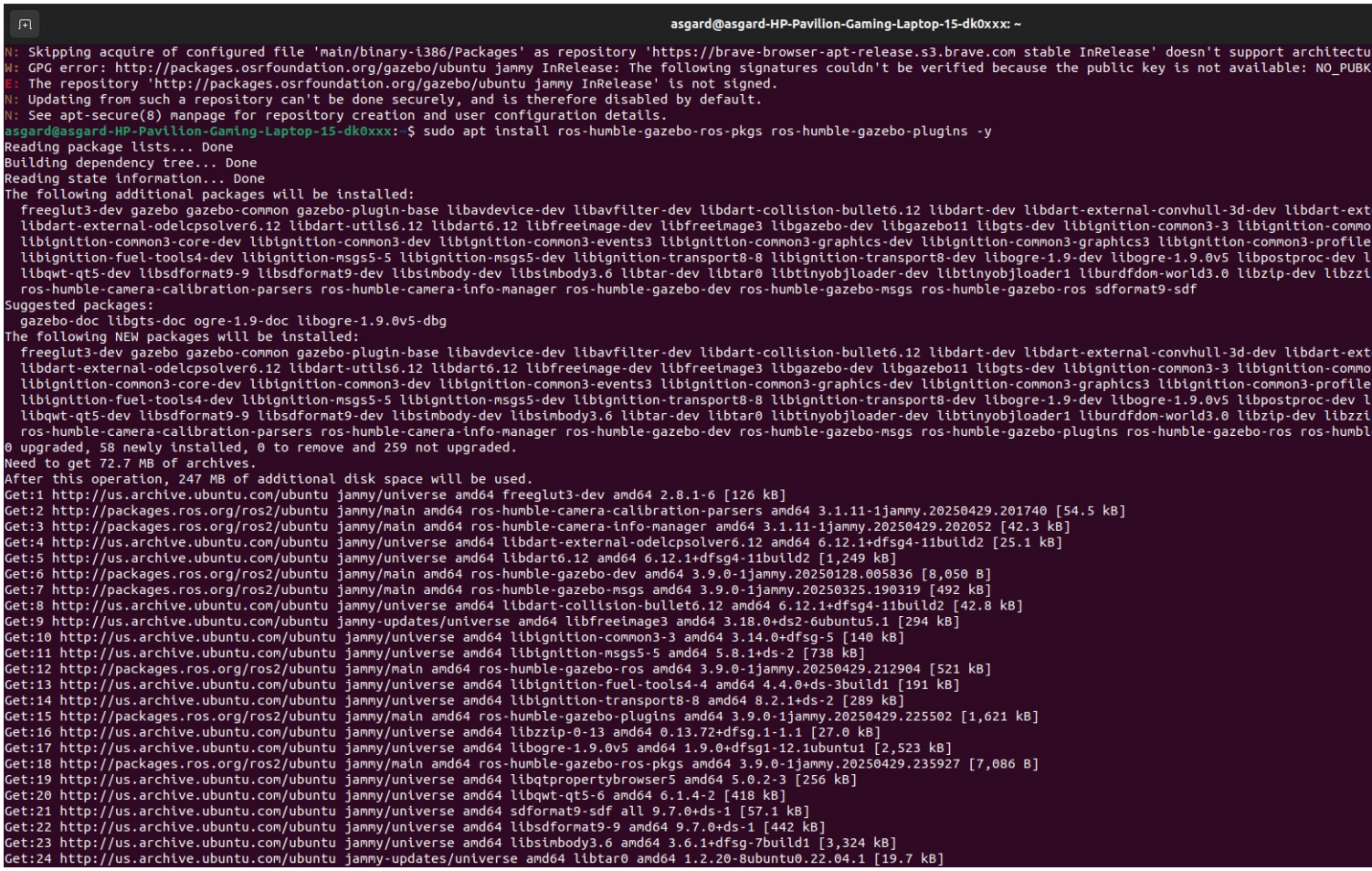
- After verifying that all foundational sensor streams were publishing, I turned my focus toward debugging the two components that showed the most estimation error in static and dynamic tests: visual odometry (VO) and the simulated GNSS feed. The aim of this task was to isolate inaccuracies in both, understand their behavior in idle and motion scenarios, and apply structured debugging methods to improve reliability.
- Began with visual odometry, since that's where I observed the most surprising behavior — namely, the robot was stationary, but the VO pose estimate kept drifting. The first thing I checked was whether the image callback in the VO node was being triggered properly. I added a simple log statement (self.get_logger().info("Entered callback")) to confirm consistent message reception from the camera stream.
- Examined the ORB features being extracted and matched between frames. I visualized the keypoints using cv2.drawKeypoints() and noted that in some frames, especially when the robot was idle, the feature matcher was still producing matches — even though they were based on noise or lighting shifts rather than actual movement. To mitigate this, I considered adding a match quality filter or enforcing a threshold on match count before proceeding with pose estimation.
- I also double-checked the intrinsic matrix used in the VO code. Although I had hardcoded fx, fy, cx, and cy, I validated these against the camera parameters reported by Gazebo to make sure they were accurate. Mismatched intrinsics can throw off essential matrix decomposition, which in turn skews relative pose.
- To reduce aggressive translation updates, I revised the step_size scaling logic in the VO node. When the estimated t vector was too large, I added a normalization step to prevent explosive motion in idle frames. This helped suppress overconfident translation estimates during visual jitter.
- Once VO debugging was in place, I moved on to the GNSS pipeline. The GPS node wasn't initially publishing, and after checking /gps/fix I confirmed that nothing was coming through. I began by ensuring /gazebo/model_states was publishing — which it wasn't. This led to a deeper inspection where I confirmed that the Gazebo plugins hadn't been loaded correctly because I was using gzserver.launch.py instead of the combined gazebo.launch.py.
- I also added a logging line inside the GNSS callback to print msg.name from /gazebo/model_states, which helped verify whether my robot (husky_robot) was even recognized. In earlier runs,

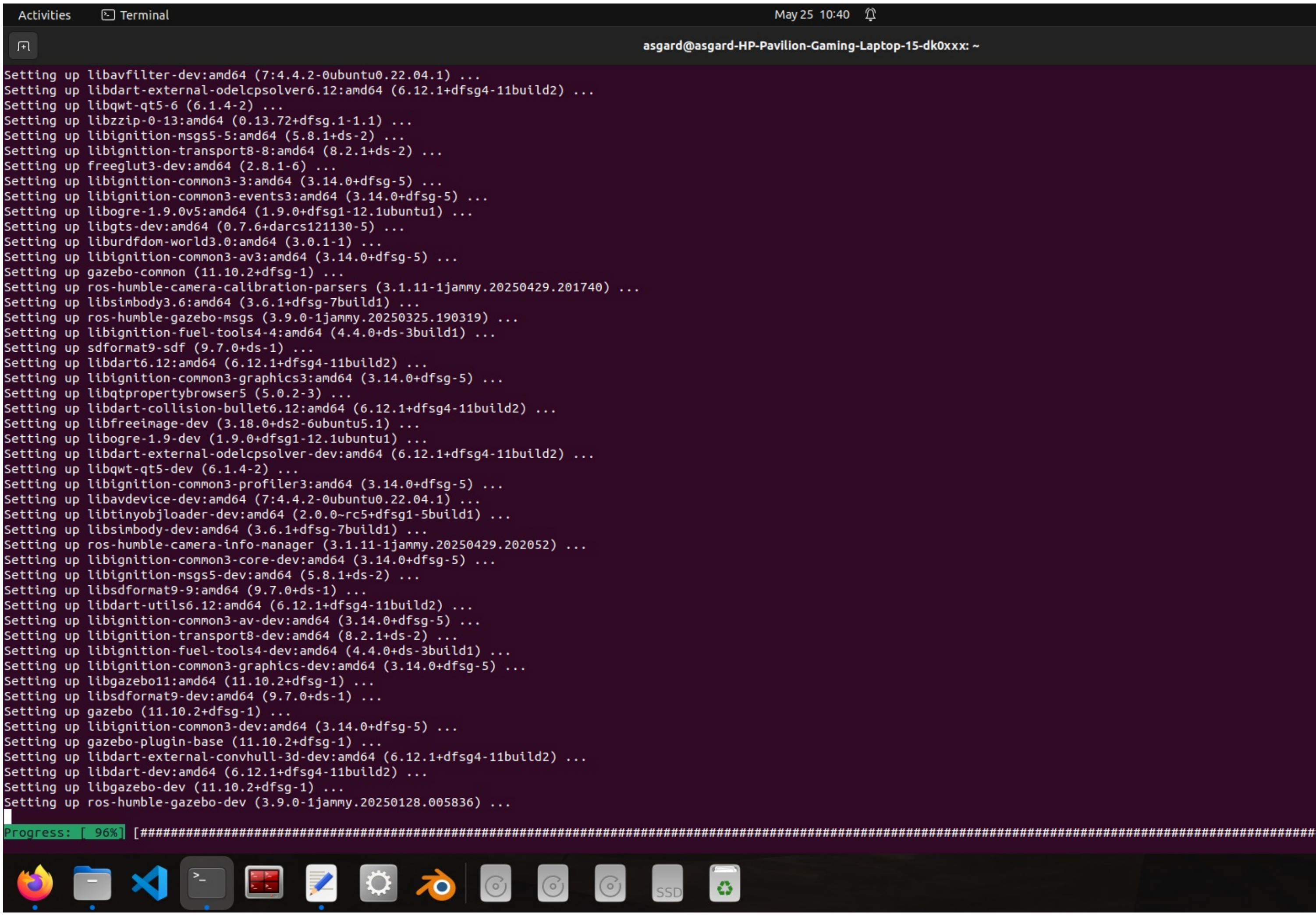
this was blank — a clear sign that the model either hadn't spawned or Gazebo wasn't reporting states.
- Once I saw ['ground_plane', 'husky_robot'] in the model list, I confirmed the robot name in the node was correctly set and that the GNSS publisher was now active. I then echoed /gps/fix and verified that latitude, longitude, and altitude were being published.

Action Item 4: Reinstalling Gazebo and Integrating an External GPS Publisher – 3 hour(s).

Project Work Summary

- After uncovering persistent issues with GNSS data not being published — even after correcting model names, launch structure, and topic subscriptions — I suspected a deeper system-level problem with the Gazebo–ROS 2 integration. Specifically, the /gazebo/model_states topic wasn't being published at all, which effectively blocked my GNSS node from functioning. This led me to the decision to fully uninstall and cleanly reinstall all Gazebo-related components to eliminate any broken dependencies, missing plugins, or conflicting versions.
- I started by purging every Gazebo- and sdformat-related package from the system using apt remove --purge 'gazebo*' 'libgazebo*' 'sdformat*' ros-humble-gazebo* -y. I followed up with an autoremove and clean to wipe any lingering cache. After confirming that gazebo --version no longer worked (which meant the uninstallation was successful), I proceeded with reinstalling the official versions supported by ROS 2 Humble on Ubuntu 22.04 — namely Gazebo Classic 11.x and its corresponding ROS bridge packages.
- Once the reinstall was complete, I verified plugin availability using a find command to check for critical .so files like libgazebo_ros_factory.so and libgazebo_ros_gps.so. I also made sure to export the GAZEBO_PLUGIN_PATH correctly to include /usr/lib/x86_64-linux-gnu and sourced my ROS 2 setup file to ensure everything was correctly linked.
- To avoid relying entirely on Gazebo's /model_states for GNSS simulation, I explored a more modular approach by integrating an external GPS publisher. I cloned the repository cavadibrahimli1/gps_publisher_ros2humble into my workspace. Instead of using their launch and world files — which were built for a standalone setup — I integrated only the GPS publishing node into my existing launch structure.
- I added the node manually inside my farmland_launch.py using a Node() block with custom parameters for origin_latitude, origin_longitude, and frame_id. This allowed me to control how GPS data was published without relying on Gazebo's internal state. After building the package, I launched the simulation and confirmed that /gps/fix was now being published independently of Gazebo's model state reporting.
- While this decoupled GPS approach is promising, full system validation hasn't been completed yet. I still need to assess how this external GPS node interacts with other localization modules, and whether it can be reliably fused with VO inside the EKF pipeline. Additionally, I haven't yet validated if the reinstall fully restored /gazebo/model_states for other dependent tools. So while the infrastructure has been reset and modularized, the debugging process, particularly for GNSS consistency and end-to-end localization





.

Action Item 5: Creating a Simulated GNSS Bridge Node from Gazebo Model States – 3 hour(s).

Project Work Summary

- After realizing that third-party GNSS packages lacked flexibility for integration with my custom world and launch setup, I opted to create a clean bridge node that would simulate GPS data directly from Gazebo's /model_states topic. The goal was to extract the robot's current pose, convert it to latitude and longitude using a fake UTM-like origin, and publish it as a standard NavSatFix message on /gps/fix, effectively simulating a GNSS sensor tied to the robot's body frame.
- I placed this new Python node (gps_pose_bridge.py) inside my main farmbot_autonomous_pkg under the correct internal folder where other nodes reside. The executable was registered via setup.py using a console script, and the node itself declared a robot_name parameter so it could dynamically identify and extract the robot's position from the list of models published by Gazebo.
- To run this node alongside my simulation, I edited world_gazebo.launch.py to include it using a Node() block with matching parameters. I also ensured the node had access to simulated time by verifying that use_sim_time was enabled across both the Gazebo launch and the node's configuration. After building and sourcing the workspace, the launch structure initialized correctly without errors.
- The debugging process revealed that the node was not receiving messages on /gazebo/model_states, or at least not entering the callback. To isolate the issue, I added logger outputs to print all available model names and confirm that the target robot (husky_robot) existed. However, the "hi" message placed inside the callback never triggered, suggesting either a name mismatch or that the topic wasn't publishing at runtime.
- I ran topic introspection commands like ros2 topic echo /gazebo/model_states and ros2 topic hz to confirm the state of the topic stream. Although /clock was active and Gazebo was launching successfully, /model_states appeared inactive, possibly due to the simulation being paused, the plugin being misconfigured, or Gazebo itself not properly initializing the robot model.
- While the bridge node itself is structurally correct and now lives inside the correct package namespace, it has not yet entered functional testing due to the absence of incoming model state messages. I plan to validate the topic publisher next, and ensure the model is being spawned with the correct name so the GNSS callback logic can begin publishing usable /gps/fix data.

Action Item 6: GNSS Integration Pipeline for ROS2-Based Mobile Robots – 3 hour(s).

Research

- https://arxiv.org/abs/2209.14649
- GNSS Integration Pipeline for ROS2-Based Mobile Robots
- Summary of Report
  - The paper introduces a factor graph-based approach that tightly fuses raw GNSS measurements with IMU and LiDAR data. This method enhances localization accuracy by considering the correlations between different sensor modalities, leading to more robust state estimation.
  - Unlike traditional GNSS systems that rely on base stations for differential corrections, this approach achieves high-precision localization without such infrastructure. This is particularly beneficial for mobile robots operating in remote or infrastructure-less environments.
  - The proposed system demonstrates resilience in environments with degraded GNSS signals, such as urban canyons or forests. By integrating LiDAR and IMU data, the system maintains localization accuracy even when GNSS data is unreliable.
  - Extensive experiments conducted on urban driving datasets and forest environments validate the system's performance. The results showcase smooth and accurate trajectories, confirming the effectiveness of the proposed fusion strategy.
- Relation to Project
  - Integrating this fusion approach can significantly improve the localization accuracy of the FarmBot, especially in agricultural fields where GNSS signals may be intermittent or obstructed.
  - The elimination of base station requirements aligns with the project's goal of deploying autonomous systems in rural areas without relying on external infrastructure.
  - The system's ability to handle GNSS-degraded scenarios ensures that the FarmBot can operate reliably in diverse agricultural terrains, including areas with dense vegetation or structures.
  - The factor graph-based method can be integrated into the existing ROS2 framework of the FarmBot, leveraging packages like robot_localization for seamless sensor fusion.
- Motivation for Research
  - The paper addresses the Limitations of using GNSS in Agriculture environment. Agricultural environments often present challenges for GNSS-based localization due to signal blockages. This research provides a viable solution to overcome such limitations.
  - The paper simplifies Deployment, by removing the need for base stations, the deployment process becomes more straightforward, reducing setup time and complexity in the field.

    - Improving Operational Reliability, through fusion strategy enhances the system's reliability, ensuring consistent performance even in the face of sensor anomalies or environmental challenges.
    - Insights from this research can inform future developments, such as incorporating additional sensors or refining the fusion algorithms to further boost localization accuracy.

Action Item 7: Decision-Making with GNSS, Visual Odometry, and YOLO – 3 hour(s).

Research

- GVINS: Tightly Coupled GNSS-Visual-Inertial Fusion for Smooth and Consistent State Estimation
- Title of the Article (each research article must be its own action item)
- Summary of Report
    - The paper presents GVINS, a system that tightly integrates GNSS, visual odometry, and inertial measurements to achieve accurate and drift-free state estimation, crucial for autonomous navigation.
    - GVINS maintains localization accuracy even when GNSS signals are unavailable or unreliable by relying on visual and inertial data, ensuring continuous operation in challenging conditions.
    - Performance of the system is designed for real-time applications, making it suitable for dynamic environments where timely decision-making is essential and could be used for our decision making algoritgms.
    - GVINS has been tested in various environments, including urban areas and indoor settings, demonstrating its versatility and adaptability to different operational contexts.
- Relation to Project
    - Enhanced Decision-Making for FarmBot by integrating GVINS can improve the FarmBot's ability to make informed decisions based on accurate localization, even in environments with limited GNSS availability.
    - Seamless Integration with YOLO and the system's real-time capabilities complement the YOLO object detection framework, allowing the FarmBot to respond promptly to visual cues in its environment.
    - By combining GNSS, visual, and inertial data, the FarmBot can achieve more precise navigation, essential for tasks like row following or obstacle avoidance in agricultural fields.
    - GVINS's robustness ensures that the FarmBot can adapt to varying environmental conditions, maintaining operational efficiency across different terrains and weather scenarios.
- Motivation for Research
    - Ensuring Continuous Operation through the ability to maintain accurate localization without sole reliance on GNSS is critical for the FarmBot's uninterrupted operation in diverse agricultural settings.
    - Enhancing Autonomous Capabilities and incorporating GVINS can elevate the FarmBot's autonomy, enabling it to make complex decisions based on a comprehensive understanding of its surroundings.
    - Accurate and reliable localization is foundational for executing complex agricultural tasks, such as targeted spraying or precise planting, which GVINS supports.
    - The insights from this research can guide the design and development of future iterations of the FarmBot, focusing on resilience, adaptability, and precision.

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.