**Project Overview**

**This project implements a high-precision GPS-based navigation system for agricultural rovers, capable of centimeter-level accuracy when used with RTK corrections. The system enables autonomous navigation through predefined waypoints in farm fields, with support for zigzag patterns commonly used in precision agriculture.**

Key Features

* Centimeter-level Precision: Achieves 100mm accuracy with RTK corrections
* Waypoint Navigation: Follows predefined paths with minimal deviation
* Real-time Visualization: Shows rover position, path, and farm boundaries
* Safety Systems: Includes geofencing, obstacle detection, and failsafe modes
* Health Monitoring: Continuously checks GPS quality and system status
* Data Logging: Records position, heading, and quality metrics at 10Hz
* Emlid GPS Integration: Works with Emlid M2/M+ GNSS receivers
* NTRIP Support: Connects to RTK correction services for enhanced precision

System Architecture

The system consists of 11 integrated modules:

* farm\_simulation.py: Main orchestration module that ties everything together
* row\_navigation.py: Handles path planning and rover movement
* emlid\_gps\_integration.py: Interfaces with Emlid GNSS receivers
* ntrip\_client.py: Manages RTK correction data streams
* gps\_system\_monitor.py: Monitors GPS quality and connection status
* coordinate\_converter.py: Transforms between coordinate systems
* logging\_100mm.py: Records high-frequency position data
* sleep\_mode.py: Implements failsafe mechanisms
* rover\_health\_check.py: Performs system diagnostics
* farm\_safety.py: Enforces boundaries and safety constraints
* waypoints\_100mm.csv: Contains predefined navigation paths

Hardware Requirements

* Computer: Windows/Linux with Python 3.7+
* GNSS Receiver: Emlid Reach M2/M+ (or compatible)
* NTRIP Subscription: For RTK corrections (optional but recommended)
* Serial Connection: USB to connect to Emlid receiver
* Software Dependencies
* Python 3.7+
* matplotlib
* numpy
* pyserial
* utm
* requests
* threading

The system will:

Perform health checks

Connect to the Emlid receiver

Attempt NTRIP connection if configured

Display a visualization window

Begin navigation through waypoints

Using Custom Waypoints

Create a CSV file with waypoints in the format:

id,left,top,right,bottom,row\_index,col\_index,id\_2,distance,angle,x,y,start\_point

"1","380000.0000","2044900.0000","380100.0000","2044900.0000","0","0","1","0","90.000000000000","380000.000000000","2044900.000",1

"1","380000.0000","2044900.0000","380100.0000","2044900.0000","0","0","1","10","90.000000000000","380010.000000000","2044900.000",0

"1","380000.0000","2044900.0000","380100.0000","2044900.0000","0","0","1","20","90.000000000000","380020.000000000","2044900.000",0

Position data is logged to rover\_log.csv with the following fields:

* timestamp
* run\_id
* x\_utm, y\_utm (UTM coordinates)
* latitude, longitude (WGS84 coordinates)
* heading, bearing, compass\_heading
* fix\_quality, satellite\_count
* deviation (from planned path in cm)
* data\_age, status

Safety Features

Geofencing

Failsafe Modes

Automatic detection and response to:

GPS signal loss

* RTK fix degradation
* Position jumps
* Communication failures
* Health Monitoring

Continuous monitoring of:

* GPS quality metrics
* Satellite count and geometry
* System communication
* Position accuracy
* Troubleshooting
* GPS Connection Issues
* Check USB connection to Emlid
* Verify COM port settings (default: COM12)
* Ensure Emlid has clear sky view
* NTRIP Connection Issues
* Verify internet connectivity
* Check NTRIP credentials
* Confirm mountpoint is appropriate for your location
* Visualization Problems
* Ensure matplotlib is properly installed
* Check for conflicting GUI frameworks
* Reduce visualization frequency if performance is poor
* The system visualizes and enforces no-go zones through the SafetyModule in farm\_safety.py. When you're moving with the Emlid, the system will:
* Visualize No-Go Zones: The farm boundaries and any defined no-go zones are displayed on the matplotlib plot.
* Detect Boundary Violations: The code checks if your position would cross into a no-go zone