**Module Overview for Farm Rover Project**

**1.** farm\_simulation.py

**Purpose:** Entry point for the rover navigation simulation.

**Key Components:**

**Health Checks:** Initializes and runs RoverHealthCheck modules before starting simulation.

**Failsafe & Logging:** Configures FailsafeModule, SafetyModule, and logging\_100mm GPS logger.

**Emlid Integration:** Uses EmlidGPSReader to stream NMEA data and inject RTCM corrections from NTRIPClient in a background thread.

**Navigation Workflow:** Creates Rover and RowNavigator, defines farm boundaries and geofence, visualizes paths using Matplotlib, and executes tasks: navigate to start, align heading, and traverse rows.

**2.** row\_navigation.py

**Purpose:** Contains core navigation and movement logic for the rover.

**Classes & Functions:**

Rover**:** Maintains state (x, y, heading), logs movement, converts between UTM and lat/lon, enforces geofence, and issues movement commands (move\_forward, visualize\_turn, etc.).

RowNavigator**:** Builds and interpolates zig‑zag row paths from CSV, calculates row-to-row transitions, and orchestrates high‑level path following (navigate\_path, move\_precisely\_to\_point).

**Helper Functions:** navigate\_to\_point, follow\_path\_precisely, update\_rover\_visualization, and visualize\_turn for fine‑grained control and plotting.

**3.** coordinate\_converter.py

**Purpose:** Bi‑directional conversion between WGS84 lat/lon and UTM (dynamic zone detection).

**Key Methods:**

latlon\_to\_utm\_coord(lat, lon): Uses the utm library to determine easting/northing and captures zone info.

utm\_to\_latlon\_coord(easting, northing, zone\_number, zone\_letter): Converts back to lat/lon using stored or default zone.

Utility functions: get\_distance\_between\_coords, get\_bearing\_between\_coords for geographical calculations.

**4.** emlid\_gps\_integration.py

**Purpose:** Serial interface to Emlid Reach GNSS receiver for real‑time position and correction injection.

**Key Features:**

**Auto‑detect Port:** Scans common device paths (COM3, /dev/ttyACM0, etc.).

**NTRIP Support:** send\_rtcm\_data(bytes) writes RTCM bytes into serial for RTK corrections.

**Data Parsing:** Supports both NMEA (\_read\_nmea) and JSON (\_read\_json) message formats.

**Callback System:** Allows registering functions (e.g. update\_rover\_from\_emlid) to handle each new position.

**5.** ntrip\_client.py

**Purpose:** Client for streaming RTCM correction data from an NTRIP caster.

**Functionality:** Connects using mountpoint credentials and yields raw correction bytes to be forwarded to the GPS reader.

**6.** rover\_health\_check.py

**Purpose:** Verifies hardware and software modules before simulation.

**Components:** Implements individual checks (battery, sensors, GPS fix) via RoverHealthCheck and raises HealthCheckFailure on critical issues.

**7.** farm\_safety.py

**Purpose:** Monitors rover path and handles obstacles, no‑go zones, and drift events.

**Mechanisms:** Defines safety callbacks and drift/no‑go recovery procedures integrated into navigation loops.

**8.** sleep\_mode.py

**Purpose:** Implements FailsafeModule to enter low‑power sleep or recovery on GPS/data/internet failures.

**Details:** Enumerates reasons (GPSFailsafeReason) and recovery actions (DriftAction).