PROJECT Autonomous Drone Navigation

END USER MANUAL

Objective

This project aims to develop a sophisticated Autonomous Drone Navigation System. It involves the creation of a high-tech flight computer that can interpret depth images from lidar and a Zed stereoscopic camera. This technology will enable the drone to autonomously navigate its environment and determine flight paths without relying on GPS data.

Hardware Requirements

- **Drone**
- **Zed 2 Camera**
- **2D Lidar**
- **NVIDIA Jetson Nano**
- **Pixhawk Flight Controller**

Software Requirements

- **Visual Studio Code**
- **Zed SDK**
- **ROS (Robot Operating System)**
- **Unreal Engine**
- **QGroundControl**

Installation Guidelines

ZED SDK Installation

Prerequisites:

- A compatible ZED stereo camera.
- System meeting Stereolabs' minimum hardware requirements.

Installation Steps:

1. **Download the SDK**: Visit the [ZED SDK Download Page] and download the latest version for your Windows system.
2. **Run the Installer**: Execute the downloaded .exe file and follow the on-screen instructions to install drivers and the SDK.
3. **Restart Your Computer**: Reboot to ensure all drivers are properly loaded.

Unreal Engine Setup

Prerequisites:

- A PC with Windows 10/11, macOS, or Linux.
- Hardware meeting Unreal Engine's system requirements.
- A free Epic Games account.

Installation Steps:

1. **Download Epic Games Launcher**: Visit the Epic Games Download Page and download the launcher.
2. **Install the Launcher**: Execute the installer and follow the on-screen instructions.
3. **Launch Epic Games Launcher**: Sign in with your Epic Games account.
4. **Install Unreal Engine**: Go to the 'Unreal Engine' tab, select 'Library', and click the '+' button to add a new engine version. Choose the desired version and click 'Install'.

QGroundControl Installation

Prerequisites:

- A compatible computer.
- A UAV using MAVLink protocol.
- A stable internet connection.

Installation Steps:

1. **Download QGroundControl**: Visit the QGroundControl Download Page and select the version for your OS.
2. **Install the Application**:
   - Windows: Run the .exe installer and follow the prompts.
   - macOS: Open the .dmg file and move QGroundControl to your Applications folder.

Post-Installation:

- **Connect to Your UAV**: Use USB, Telemetry Radio, or Wi-Fi to connect your UAV to QGroundControl.
- **Firmware Setup**: Update your UAV's firmware via QGroundControl if necessary.
- **Calibrate Sensors**: Follow QGroundControl's instructions for sensor calibration.

ROS (Robot Operating System) with RPLidar: Installation and Setup Guide

Prerequisites

- **Operating System**: Ubuntu 20.04 (Focal) or Windows 10/11.
- **Ubuntu Setup**:
  - Configure Ubuntu repositories.
  - Set up the necessary environment.

Installation Steps for ROS on Ubuntu

1. Follow the comprehensive installation guide for ROS Noetic at [ROS Noetic Installation on Ubuntu].

Building the RPLidar ROS Package

1. **Create a Catkin Workspace**: Set up a new workspace for Catkin.
2. **Clone the RPLidar Project**:
    - Navigate to the src folder of your Catkin workspace.
    - Clone the RPLidar ROS package from GitHub: [RPLidar ROS GitHub Repository].
3. **Build the Package**:
    - Run catkin_make to build the rplidarNode and rplidarNodeClient.

Running the RPLidar ROS Package

- **Option 1: Using RViz**:
    - Execute roslaunch rplidar_ros view_rplidar.launch.
    - View RPLidar's scan results in RViz.
- **Option 2: Using Test Application**:
    - Start the node with roslaunch rplidar_ros rplidar.launch.
    - Run rosrun rplidar_ros rplidarNodeClient.
    - Check the RPLidar's scan results in the console.

RPLidar SDK Installation on Windows

1. **Download the SDK**:
    - Get the RPLidar A1M8 SDK from [Slamtec], which includes user manuals, the RPLidar kit, datasheets, and more.
2. **Install the Driver**:
    - Use CP210xVCPInstaller_x64 from the SDK package, located in rplidar_sdk\rplidar_sdk-master\tools\cp2102_driver\CP210x_Windows_Drivers.
3. **Connect and Verify**:
    - Connect the RPLidar to your system via USB.
    - Ensure the correct COM port is visible in the Device Manager.
    - A UI application representing the Lidar should appear once the setup is successful.

Running the Autonomous Drone Navigation System Project

Step 1: Setting Up the Hardware

1. **Assemble the Drone**: Ensure the drone is properly assembled, including the attachment of the Zed 2 camera, 2D lidar, and the Pixhawk flight controller.
2. **Connect to Jetson Nano**: Securely connect the Zed 2 camera and 2D lidar to the NVIDIA Jetson Nano, ensuring all connections are stable and secure.

Step 2: Initial Software Configuration

1. **Configure ROS**: Launch ROS on the Jetson Nano and ensure it recognizes the Zed 2 camera and 2D lidar.

2. **Unreal Engine Setup**: If simulation is part of the project, set up the environment in Unreal Engine. Ensure that the Unreal Engine is ready to simulate the environment for the drone.

Step 3: Integrating the Components

1. **Sync Devices**: Ensure that all devices (camera, lidar, Jetson Nano, Pixhawk) are communicating correctly with each other.
2. **Calibrate Sensors**: Using QGroundControl, calibrate the drone's sensors, including the camera and lidar, for accurate data collection and navigation.

Step 4: Test Run in a Controlled Environment

1. **Simulation Test**: If applicable, run a simulation test in Unreal Engine to verify the drone's navigation and obstacle avoidance algorithms.
2. **Physical Test**: Conduct a controlled physical test of the drone in a safe, open area. Monitor the drone’s response to environmental data and its autonomous decision-making process.

Step 5: Monitoring and Debugging

1. **Live Monitoring**: Use Visual Studio Code and QGroundControl for live monitoring of the drone's performance. Check for real-time data transmission and processing.
2. **Debugging**: Identify any issues in the navigation system. Utilize logs and real-time data to debug and refine the system.

Step 6: Iterative Testing and Refinement

1. **Iterative Approach**: Conduct multiple test flights, making adjustments to the software and calibration as necessary based on test outcomes.
2. **Environment Variation**: Test in various environments and conditions to ensure robustness and reliability of the navigation system.

Step 7: Documentation and Analysis

1. **Record Findings**: Document the outcomes of each test, including any anomalies or successful navigation instances.
2. **Analyze Data**: Analyze the collected data to understand the drone's performance and areas for improvement.

Step 8: Final Integration and Testing

1. **Integration**: Once satisfied with the test results, integrate all components for the final setup.
2. **Comprehensive Testing**: Perform comprehensive testing in diverse environments to ensure the system is fully functional and reliable.

Troubleshooting

Hardware Issues

1. **Camera/Lidar Not Detected**:
   - **Check Connections**: Ensure that the Zed 2 camera and 2D lidar are properly connected .
   - **Reboot System**: Sometimes a simple reboot can resolve detection issues.
   - **Driver Verification**: Verify that the correct drivers are installed for the camera and lidar.
2. **Pixhawk Connection Issues**:
   - **Firmware Check**: Ensure the Pixhawk has the latest firmware.
   - **Wiring**: Check all cables and connections to the Pixhawk for any loose or damaged wires.
   - **QGroundControl**: Use QGroundControl to diagnose connection or recognition issues.

Software Issues

1. **ROS Integration Problems**:
   - **Dependency Check**: Ensure all ROS dependencies are properly installed.
   - **Package Conflicts**: Look for any conflicting packages that might cause issues and resolve them.
2. **Unreal Engine Simulation Errors**:
   - **System Requirements**: Verify that your system meets the hardware requirements for running Unreal Engine.
   - **Update Engine**: Ensure Unreal Engine is up-to-date.
3. **SDK Integration Issues :**
   - **SDK Version:** Ensure you are using the correct version of the ZED SDK that is compatible with your system and the ROS version you are using.
   - **Dependency Conflicts**: Check for any conflicts or missing dependencies that the ZED SDK might have with other software components on your system

General Troubleshooting

1. **Systematic Approach**: Tackle one issue at a time, starting from the most basic checks to more complex diagnostics.
2. **Documentation**: Keep a record of any errors encountered and how they were resolved for future reference.
3. **Community and Forums**: Utilize online forums and communities related to ROS, Unreal Engine, and drone hardware for additional support and insights.
4. **Software Updates**: Regularly update all software components to ensure compatibility and performance.