Documentation of the QGroundControl WiMA-Extension

Valentin Platzgummer

September 11, 2019

1 Introduction

WiMA is a abbreviation for **Wi**reless **M**easurement **A**pplication. Dieses Dokument soll einerseits die Funktionen der WiMA-Erweiterung dokumentieren und andererseits den Leser dazu anregen Fehler im Programm zu finden. Das Dokument ist in zwei abschnitte unterteilt. Die eigentliche Dokumentation und Vorschläge welche Funktionen getestet werden sollen. Da die WiMA-Erweiterung noch weiterentwickelt wird, können die Funktionen und das Aussehen des Programms von der in dieser Dokumentation dargestellten Inhalten abweichen.

Der Ordner deploy enthält eine unter Linux ausführbare Datei des Programms, mit dem nahmen "QGroundControl.AppImage".

2 Documentation

2.1 Structure of QGroundControl with WiMA-Extension

Figure 1 shows a detail view of the QGC main window. Relevant for this documentation are the flight view, the plan view and the WiMA main window. Their function will now be briefly summarized.

The **plan view window** is used to create flight plans. They can be stored as .plan files or directly be uploaded to a vehicle (either a real one or a simulated one). Flight plans consist of waypoints. Each waypoint stores, amongst others, a coordinate (latitude, longitude, altitude) and a command (take off, land, wait for x seconds, etc.). To define a sensible flight plan a sequence of waypoints should begin with a take off command and end with a land command. Besides waypoints more complex patterns can be defined, which is done by inserting a Survey, a Circular Survey, a Structure Scan or a Corridor Scan. Just try them out! Using the simulator can be very helpful.

The **flight view window** comes in necessary as soon as you want a vehicle to get in action. Once the vehicle established a connection (serial, TCP, UDP, etc.) QGroundControl starts to communicate automatically with it automatically in most cases. If not, check the settings menu of QGroundControl. A red arrow will appear on the map showing the vehicles position and orientation. On the top indicator strip telemetry data will be published. At the left edge a tool strip will be activated, which can be used to command the vehicle.

The **WiMA main window** is used to automatically generate flight paths on the base of minimal user input. The user can define a Measurement Area, a Service Area (for take off, land, supply tasks, etc.) and a Corridor, which connects the previos two areas. Below the WiMA main window will be described in more detail.

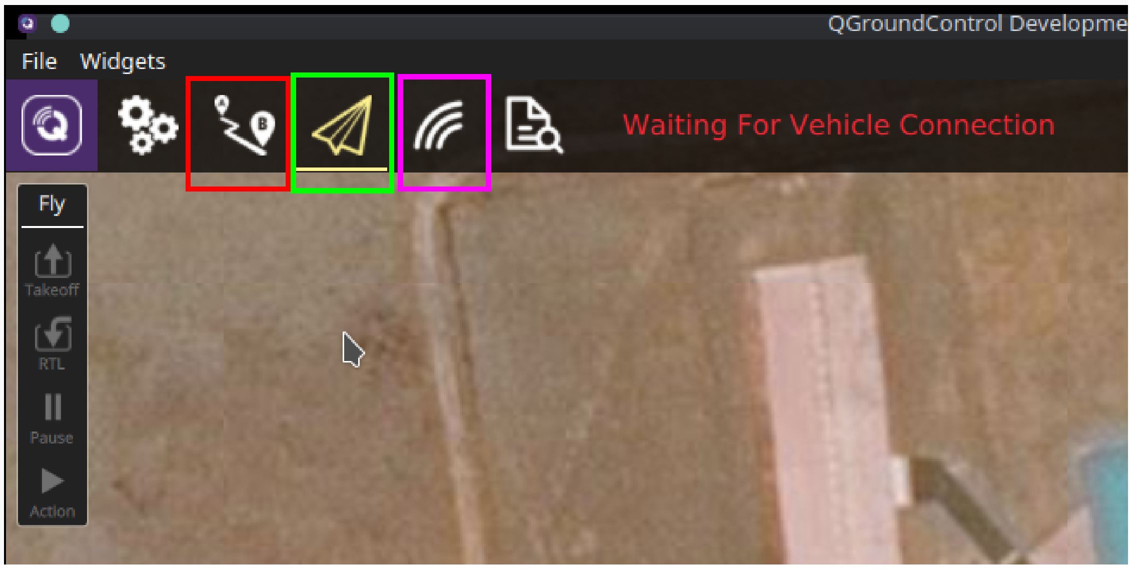

Figure 1: Detail view of the QGC window, which appears after start-up. Marked in red is the button for switching to the plan view window, green indicates the flight view button (current window) and marked in magenta is the button for switching to the WiMA main window.

3 WiMA Main Window

4 ArduPilot Simulator

For tasks like debugging, program verification or flight plan testing a simulated vehicle can be very useful. It can save time, money and prevent you from any excessive sunburns, if you forgot that you are actually outside, starring on your screen, exposed to the hot summer sun.

For this task the ArduPilot simulator can be used. It simulates a vehicle runnig the ArduPilot flight stack (firmware) on your local machine. Data is beeing published by the simulator via UDP and should ideally connect to QGroundControl without any further tweaks.

The simulator is part of the ArduPilot project, which can be downloaded from Github: https://github.com/ArduPilot/ardupilot. It is recommanded to fork the repository. After cloning the repository, the submodules must be initialized and updated. Execute the following code to do this.

```
git clone https://github.com/ArduPilot/ardupilot
cd ardupilot
git submodule init
git submodule update
```

The simulator is launched by a Python script, thus Python must be installed on your machine. The simulator is now ready to run, it can be launched from your ArduPilot root directory with the following command.

```
./Tools/autotest/sim_vehicle.py -v ArduCopter
```

Thereby the `-v` option specifies the vehicle type. The `-l` option can be used to define a custom start location. See the `sim_vehicle.py --help` option for further information. After launching, the vehicle should appear inside QGC.