Documentation of the QGroundControl WiMA-Extension

Valentin Platzgummer

September 14, 2019

1 Introduction

WiMA is a abbreviation for **Wi**reless **M**easurement **A**pplication.

This document was created to explain the functionality of the WiMA-Extension at one hand and to encourage the reader to find bugs inside the program. Hence the document is split in two parts. The first part contains instructions on how to use WiMA and the second part gives some suggestions for testing. As the extension is still being developed the contents demonstrated inside this document may differ from those ones in the program.

The folder ”deploy” in the QGroundControl root directory (can be cloned from Gitlab) contains a AppImage of the program. QGroundControl can be launched by double-clicking the AppImage. Currently only a Linux version is available.

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, among 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 previous two areas. Below the WiMA main window will be described in more detail.

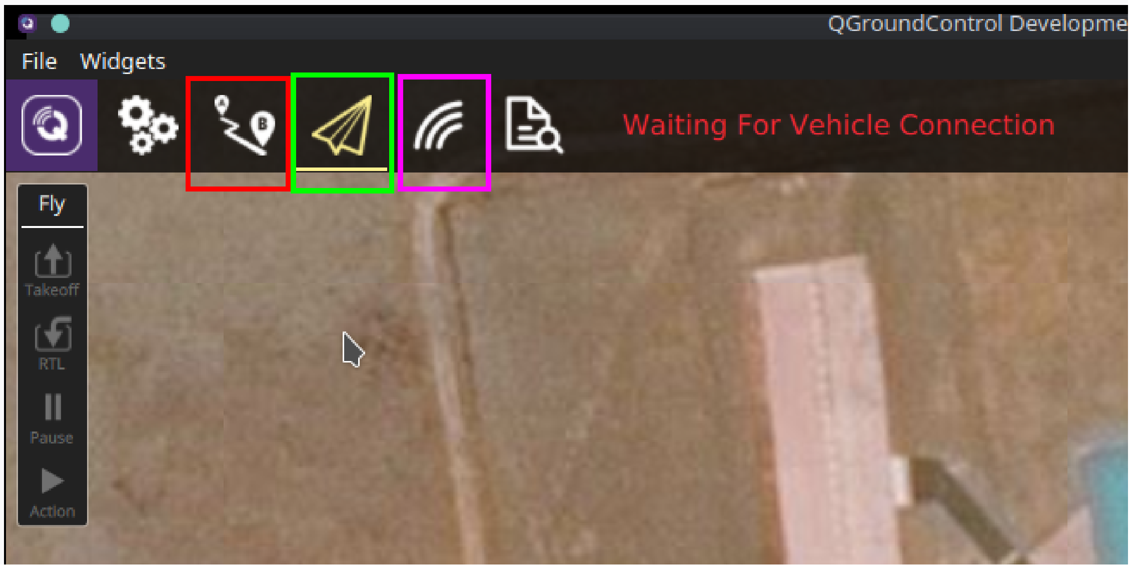


Figure 1: Detail view of the QGC window, which appears after start-up. Marked in red is the button for switching to the plan view window, green indicates the flight view button (current window) and marked in magenta is the button for switching to the WiMA main window.

2.2 WiMA Main Window

2.2.1 WiMA Tool Strip

By clicking the wave symbol (see fig. 1; magenta square) the WiMA main window appears. After entering, at the left edge, the WiMA tool strip will appear (depicted in fig. 2).

Pressing the **File** button opens a menu which offers saving and loading operations. At one hand all WiMA areas and mission items can be saved using the `.wima` file extension, at the other hand the mission items only can be stored using the `.plan` file extension. The Open button can be used to load previously stored files. Pressing the New button deletes all contents within the WiMA Main Window. Additionally with the Upload, Download and Clear Vehicle Mission button the mission items (if present) can be uploaded, downloaded or be deleted from the vehicle respectively.

The **Measure**, **Service** and **Corridor** buttons are used to insert a measurement, service and corridor area, respectively. At the time only one of each areas can be inserted. To automatically generate a flight plan at least a measurement and a service area must be defined. Both must be overlapping.

To auto generate a flight plan the **Calculate** button must be pressed. Further information about how to display the generated flight plan will follow below. In the future this button might be removed and be replaced by a routine which automatically triggers recalculation after any modification. However as the flight plan generation can be time consuming on older devices, this button will remain in the near future, up to the point, underlying routines get optimized.

As flight generation is manually triggered the flight plans might look wrong (for e.g. after user modification), pressing the **Calculate** button often will remove errors. If not, please report any

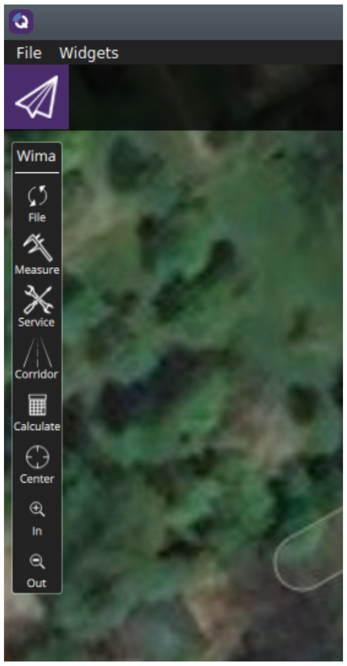


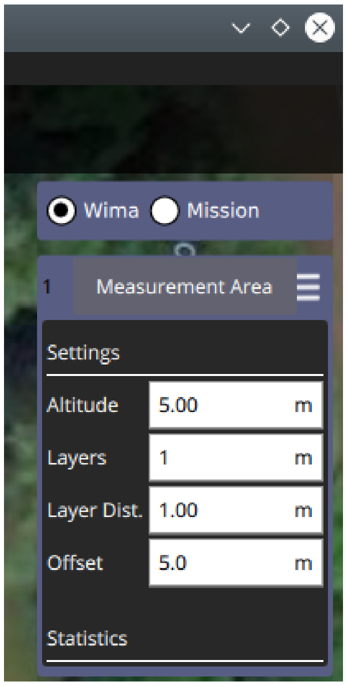
Figure 2: WiMA Toolstrip inside WiMA Main Window.

bugs.

The last three buttons, **Center**, **In** and **Out** are used to center the view and for zooming respectively.

2.2.2 WiMA Areas

As all ready mention the three areas, namely the Measurement Area, the Service Area and the Corridor can be defined using the corresponding buttons of the WiMA Tool Strip (see 2.2.1).



Wima [✓] Mission

1 Measurement Area

Settings

Altitude	5.00 m
Layers	1 m
Layer Dist.	1.00 m
Offset	5.0 m

Statistics

Figure 3: WiMA Item Editor at the right edge of the WiMA Main Window.

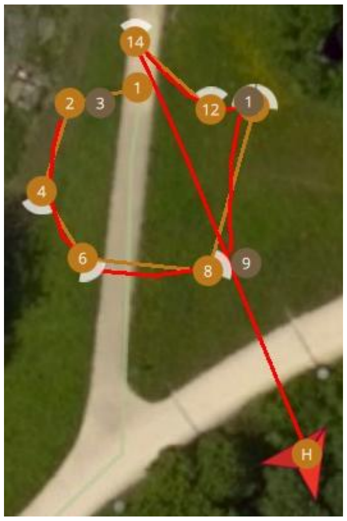


Figure 4: Flight path of a real world vehicle (the red line).

The **WiMA Measurement Area** defines, as the name indicates, the area of interest within which any measurements should be performed. The area will be displayed as a green shaded rectangle surrounded by a white line, after pressing the Measure button. The area will be listed at the right side within the WiMA Item Editor, after creation (see fig. 3). The WiMA Item Editor can be used to modify area parameters.

The Offset parameter (see fig. 3) changes the distance between the measurement area and its

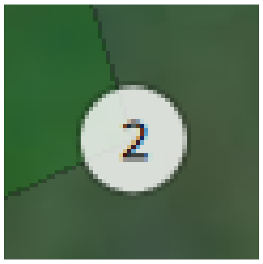


Figure 5: Adjust the vertex by dragging it's handle.

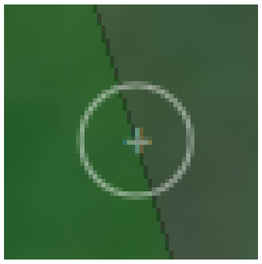


Figure 6: Add vertices by hitting the plus sign.

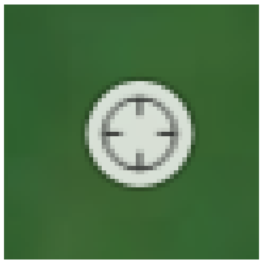


Figure 7: Move any area by dragging its drag handle.

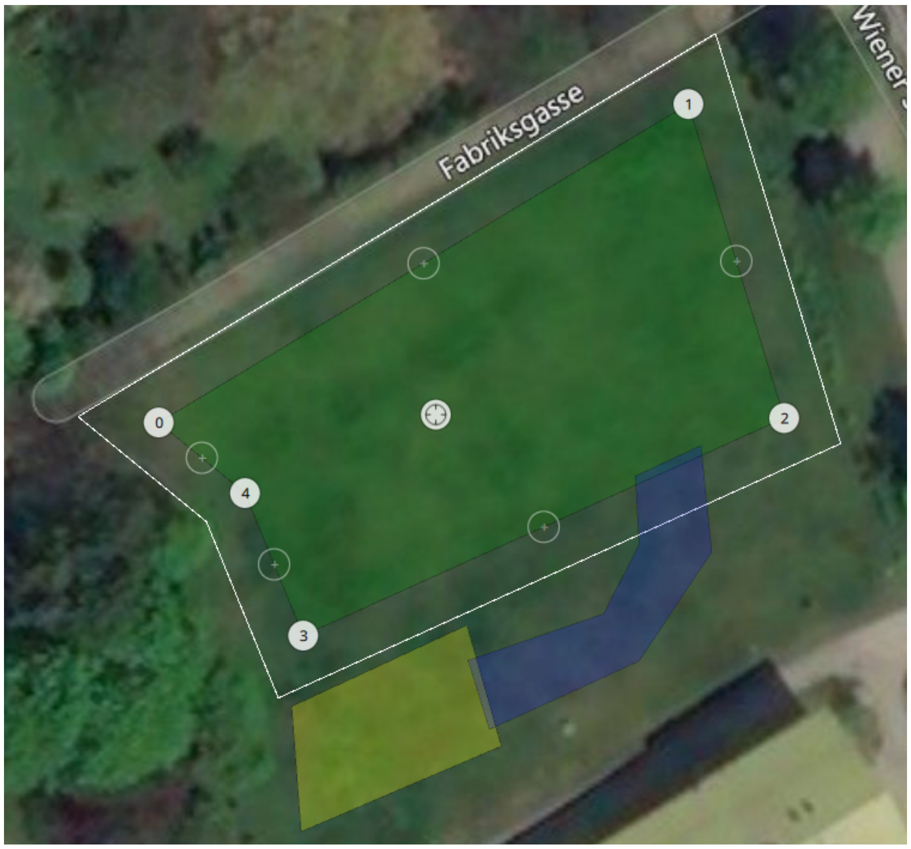


Figure 8: A valid configuration of Measurement, Service Area and Corridor.

surrounding polygon. The surrounding polygon is a helper to provide sufficient clearance between the measurement area and surrounding obstacles like trees or buildings.

All other parameters shown in fig. 3 don't yet have any impact (intended for future use).

The **WiMA Service Area** is meant to be the area were takeoff and landing happens as well as battery exchange or (hopefully no) repair work is done, hence the name.

The **WiMA Corridor** connects the two previous areas and defines a corridor which the vehicles uses to travel between Service and Measurement Area.

The flight plan will be generated such that all waypoints are within or at least at the edge of the above mentioned areas. However it should be taken in account, that the vehicle could still leave the save area, even under fully functional operation. Depending on software implementation of the vehicles flight stack (firmware) the flight controller could decide to perform any kind of path optimization. Additionally drifts caused by wind gusts could happen. An example can be seen in fig. 4. The vehicles flight controller indeed has no information about any WiMA Areas, they exist exclusively within the QGroundControl application.

All WiMA areas can be shaped by dragging the vertex handles (see fig. 5). New vertices can be created by hitting the plus signs at the edges (see fig. 6). The whole area can be moved by dragging its drag handle (see fig. 7).

A valid configuration including Measurement, Service Area and Corridor could look like shown in fig.

2.2.3 The generated Flight Path

2.2.4 Circular Survey

2.3 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.