ArduPilot Methodic Configurator

WHY, WHAT, HOW?

DR.-ING. AMILCAR DO CARMO LUCAS

2025-03-13

ArduPilot Methodic Configurator

- Why ArduPilot Methodic Configurator exists?
- What does it do?
- How does it do it?
- Is it reliable?
- Is it compliant?

Why ArduPilot Methodic Configurator exists?

- ArduPilot supports:
  - Over 244 different STM32 flight controller boards and then some ESP32 and Linux boards
  - Hundreds of different additional sensors
  - Many different propulsion systems

- It must be configured and tuned to operate correctly
  - Configuration is complex requiring many interdependent steps
  - Order and interdependency is unclear for many users

- Extensive documentation exists, but …
  - It is complex, over 1600 unique webpages
  - Requires the users to learn a lot and manually use different tools
  - Many users incorrectly configure their vehicles

ArduPilot Methodic Configurator was created to address these issues

Common pitfalls addressed by AMC

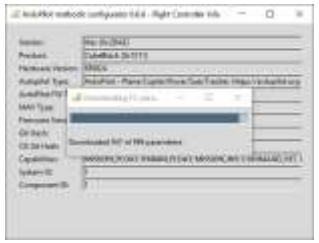
- Missed mandatory configuration steps
- Wrong sequence of steps, causing previous steps to be invalid. For example:
  - Autotune before notch filter
  - MagFit before current sensor calibration
  - No GNSS when doing compass calibration, etc, etc
- A lot of trial and error, with an unnecessary number of test flights
- Not all relevant components parameter are found under one prefix. For example:
  - ESC settings are found under: ESC_, SERIAL_, MOT_, SERVO_, CAN_,
  - GNSS settings are found under: GPS_, SERIAL_, CAN_, NTF_
- When doing a configuration step, where is the relevant documentation for that step?
- Where is the relevant documentation for the flight controller?
- When reviewing a configuration there is no way to find out why a parameter change was made

What does it do?

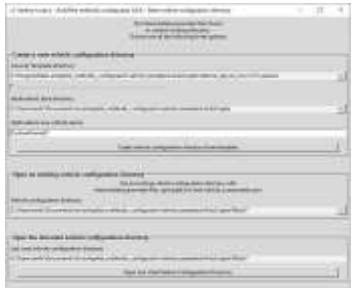
Feature	Mission Planner, QGroundControl, ... etc	ArduPilot Methodic Configurator
full automatic configuration	No	No
configuration type	manual [1]	semi-automated [2]
explains what to do	No	Yes
explains when to do something	No	Yes, explains the path
explains why do something	No	Yes
configuration method	a different menu for each task, some tasks have no menu, so you need to dig into the 1200 parameters	each task only presents you a relevant subset of parameters
parameter documentation	Yes, only on the full-parameter tree view	Yes
displays relevant documentation	No	Yes
makes sure you do not forget a step	No	Yes
checks that parameters get correctly uploaded	No (MP), unsure (QGCS), yes (MAVProxy)	Yes
reuse params in other vehicles	No, unless you hand edit files	Yes, out-of-the-box
documents why you changed each parameter	No	Yes
tutorials and learning resources	No, scattered and not integrated	Yes, context-aware help integrated
auto. install lua scripts on the FC	No	Yes
auto. backup of parameters before changing them	No	Yes

How it does it?

It uses a Wizard-like interface to:



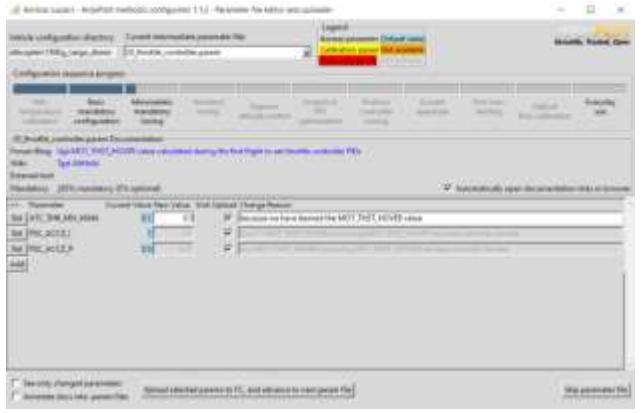
or



1. Check for software updates

2. Connect to the flight controller

3. Create a new project or open an existing one



4. Enter vehicle components information and how they are connected to the flight controller

5. Change the parameters – semi-automated

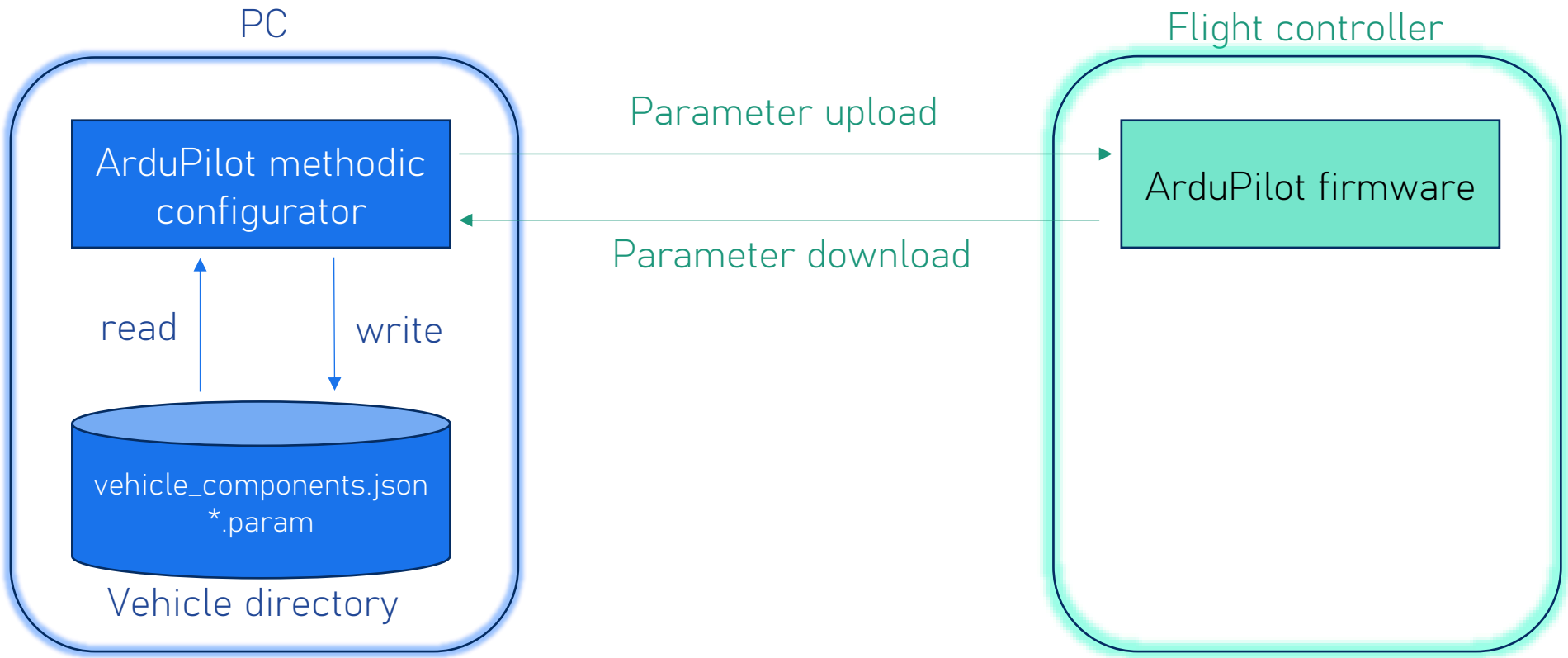
Parameter configuration steps categories

- Interdependencies mandate that they are done in a fixed sequence
- Each category contains multiple steps
- Each step is a *.param* file

Local files and remote flight controller

To avoid confusion, the software is consistent:

- The verbs ***read*** and ***write*** are used for local files
- The verbs ***upload*** and ***download*** are used for flight controller operations



Parameter configuration steps

Steps that require **no** experimental data gathering:

Set the parameter values directly in a single *.param file and upload them to the flight controller

Steps that require experimental data gathering:

or

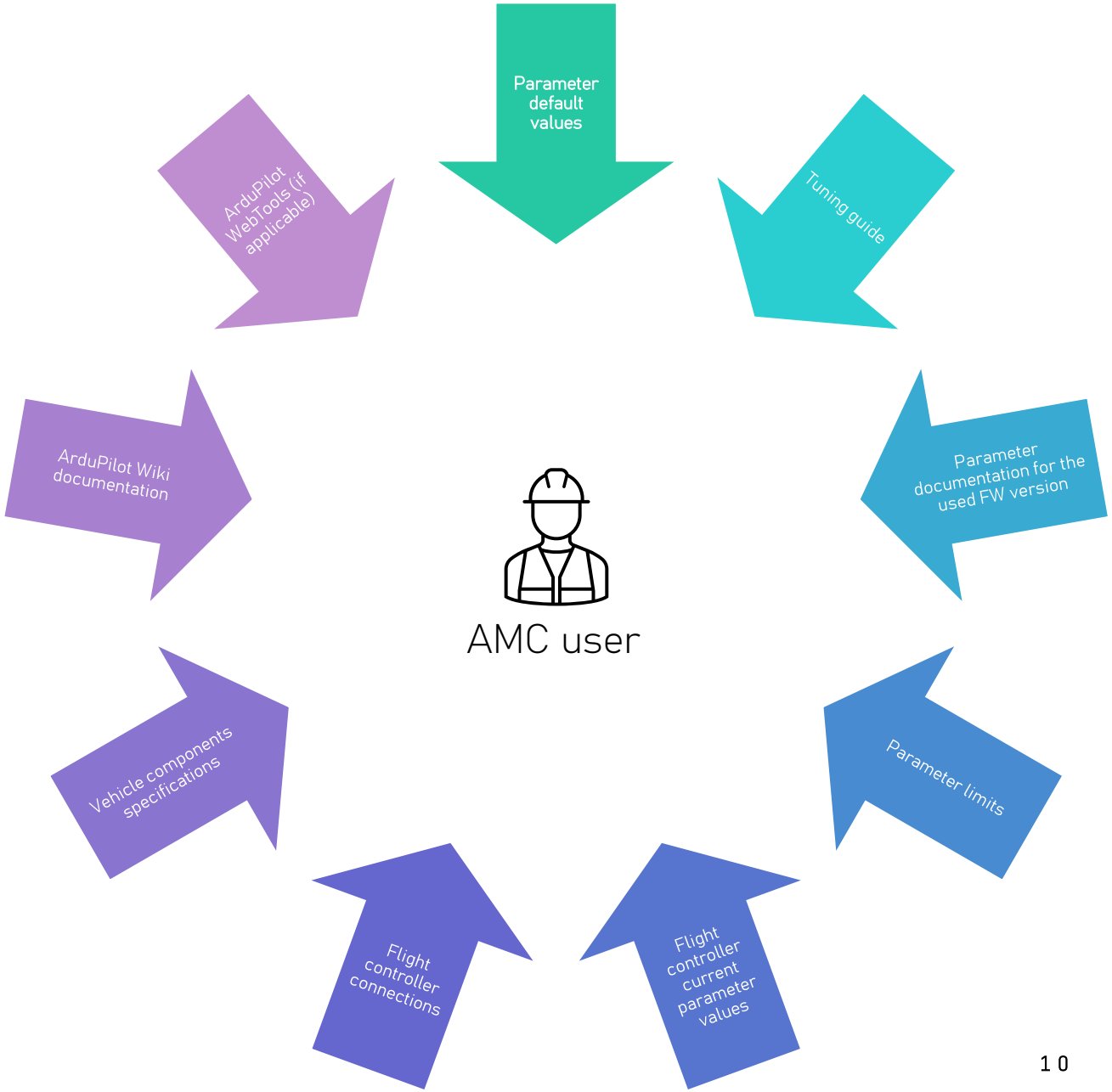
1. Setup the *_setup.param parameters to configure the data gathering experiment, and upload them to the flight controller
2. Conduct the data gathering experiment as explained in the linked tuning guide
3. Analyze the collected data as explained in the tunning linked guide
4. Store the resulting parameters in a *_results.param file, most times by automatically downloading them from the flight controller

Integrated decision aids

The software automatically correctly sets some parameters based on user provided information but asks user permission to upload them to the FC.

When setting the parameter values the user has many sources of information available in the software GUI.

Some tools/webpages open automatically in a browser window, the user needs less clicks and less *googling*



Is it reliable?

600g

900g

2900g

- the software has been used by hundreds of ArduPilot developers and users.
  - From beginners to advanced.
  - On big and small vehicles.
- It contains over 484 automated regression tests
- It has no known bugs

7Kg

50Kg

350Kg

Is it compliant? 1/2

Usability

•Uses *What you see is what gets changed* paradigm. No parameters are changed without the users's knowledge

•Translated into multiple languages

•No visible menus, no hidden menus.

Code Quality

• PEP 8 Python code style guidelines, PEP 484 type hints, PEP 621 project metadata standards

•Follows object-oriented design principles and clean code practices

•Implements comprehensive error handling and logging, with 5 verbosity levels

•Automated changelog in Keep a Changelog format

•Complies with Python Packaging Authority (pypi) guidelines

Software Development

•Maintains comprehensive assertion-based test coverage through pytest

•Uses semantic versioning for releases

•Follows git-flow branching model

•Implements automated security scanning and vulnerability checks

•Implements git pre-commit hooks to ensure code quality and compliance on every commit

•Implements reproducible builds with pinned dependencies

•Uses containerized CI/CD environments for consistency

•Implements automated dependency updates and security patches using renovate and dependabot

Is it compliant? 2/2

Open Source

•Complies with OpenSSF Best Practices for open-source projects

•Uses REUSE specification for license compliance

- Uses CI job to ensure compliance
- Uses SPDX license identifiers

•Maintains comprehensive (more than 5000 lines) documentation

•Implements inclusive community guidelines

•Provides clear contribution procedures

Security

•Regular security audits through Snyk, codacy, black duck and other tools

•Follows OpenSSF Security Scorecard best practices

•Uses gitleaks pre-commit hook to ensure no secrets are leaked

•Implements secure coding practices, runs anti-virus in CI

•Maintains security policy and vulnerability reporting process

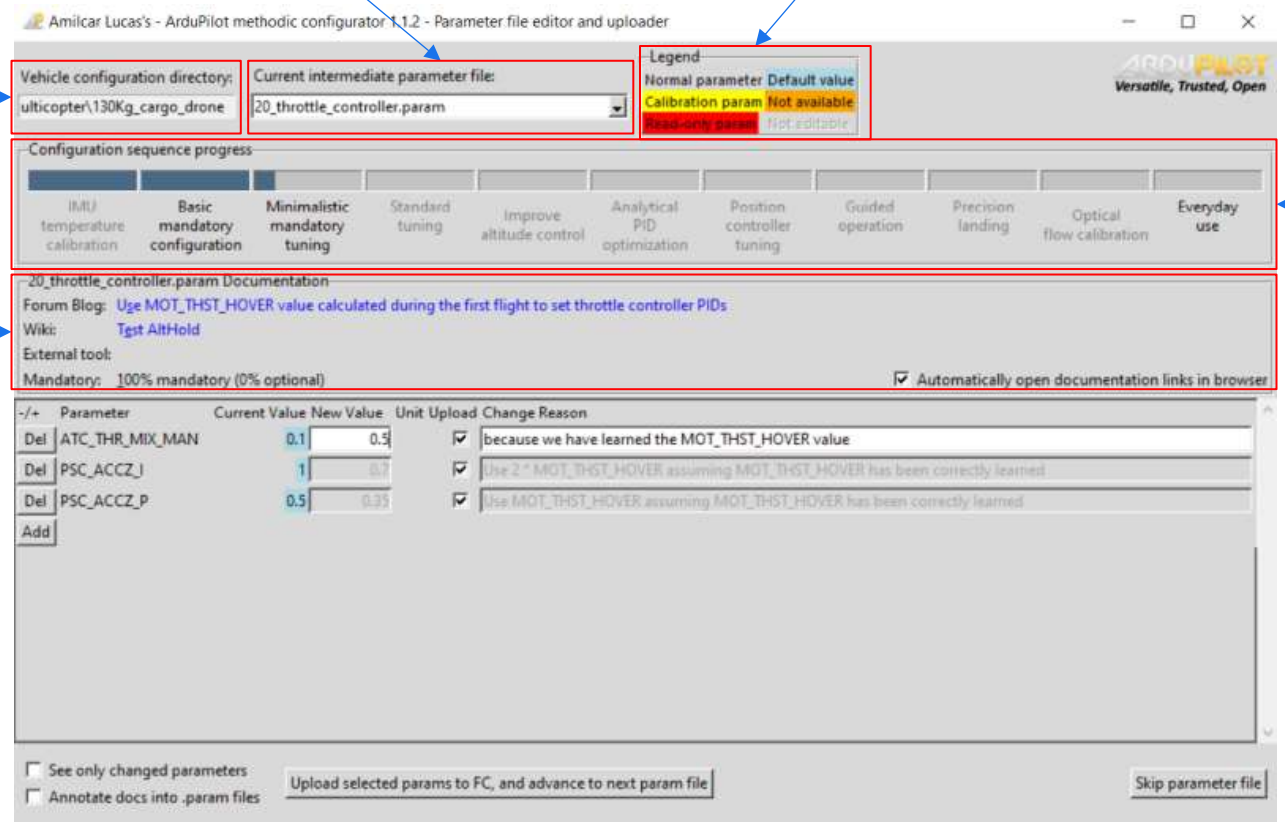
Usage 1/2

Current configuration step
Usually, you do not manually change this

Parameter attributes legend
Detailed information on mouse-over

All this project's files are stored here

Documentation for this configuration step
Click on the blue texts to open in browser



Shows your progress trough the configuration phases

- Each phase contains configuration steps that must be completed in sequence
- Greyed-out phases are optional and can be skipped

Usage 2/2

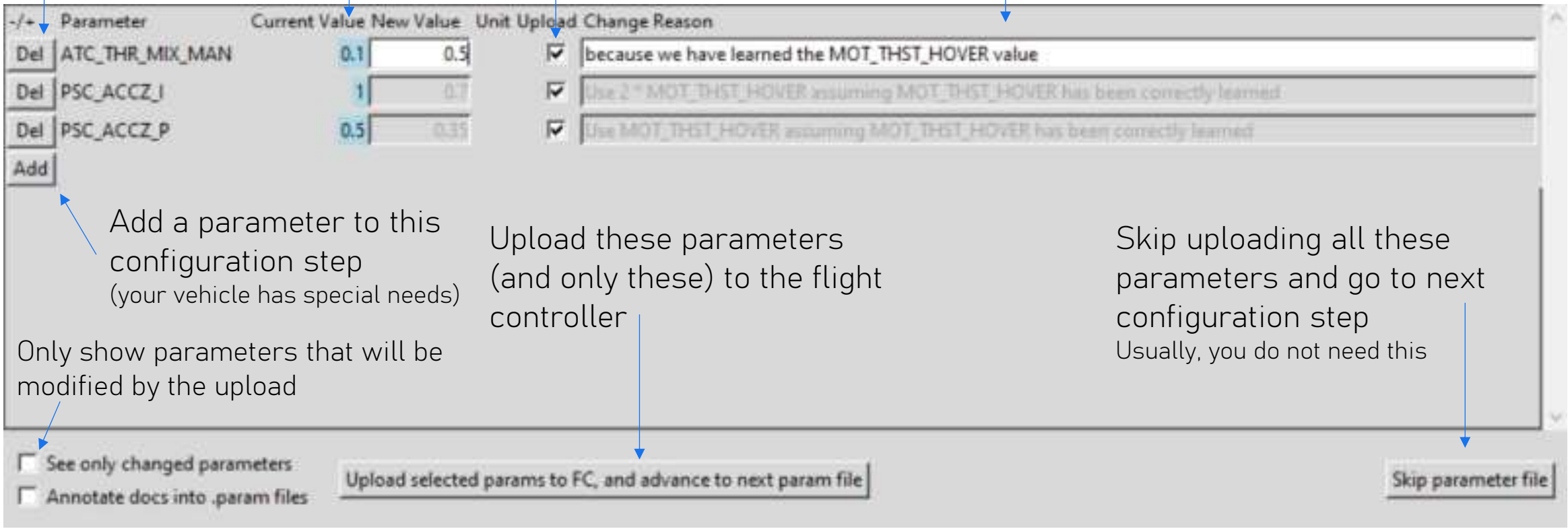
Delete this parameter from this step
(Do it if it does not apply to your vehicle)
It will remain on the FC

Current flight controller value

When selected upload the new value to the flight controller
Usually, you do not manually change this

Document the reason you changed this parameter to the new value at this configuration step

- Allows you and others to understand why this change was made
- Forces you to think
- Provides traceability



Add a parameter to this configuration step
(your vehicle has special needs)

Only show parameters that will be modified by the upload

Upload these parameters (and only these) to the flight controller

Skip uploading all these parameters and go to next configuration step
Usually, you do not need this

Customization without changing source code

The **vehicle components are customizable** without changing the python source code.

- Fully defined in the vehicle_components.json file.
- Follows vehicle_components_schema.json.
- Copy the vehicle_components.json file to your vehicle directory, edit it and you'll have all your new components in the GUI.

The **configuration sequence is customizable** without changing the python source code.

- Fully defined in the configuration_steps_ArduCopter.json file.
- Follows configuration_steps_schema.json.
- Copy the configuration_steps_ArduCopter.json file to your vehicle directory, edit it and you'll have changed the configuration sequence on the GUI. You can change, add, remove or reorder any of the steps.

Additional information sources

Configuration guide

User manual

Use cases

Software architecture

All these resources include an **ArduPilot-trained AI chatbot** on the bottom right corner

Summary

- Reduces human error probability
- Increases configuration speed
- Increases consistency and reproduce-ability
- Improves documentation and traceability

Thank you for your attention

let's make better ArduCopters

AMILCAR.LUCAS@IAV.DE