Fusion Engineering Flight Controller Technology

Control technology

We use the latest research in state estimation, flight control, and fault tolerance, to make the most robust system possible.

UKF based state estimation

The State Estimator is the part that combines all available sensor data to make the best possible estimation of the current state of the drone: The position, orientation, velocity, etc. Unfortunately, no sensor is perfect. Sensors have noise, a (changing) bias, and can have quite big uncertainties. The State Estimator uses a technique called sensor fusion to take all (noisy, uncertain) information from the sensors, and combine them to make the best possible estimation of everything we want to know.

Our State Estimator is based on the Unscented Kalman Filter (UKF), as it is a generally superior method compared to the conventional Extended Kalman Filter (EKF) for nonlinear state estimation. Furthermore, the implementation of our UKF-based state estimator is extremely flexible. Additional sensors and state variables can be added with ease, to allow for optimal state estimation using all information available.

INDI based flight control

The flight controller is what controls the propellers to make the drone fly the way it should. It uses the information from the State Estimator to determine what the drone is doing, and calculates the best propeller speeds to make the drone do what it should be doing.

A drone is a highly nonlinear system with very fast dynamics. Virtually all drone flight controllers use the conventional PID (Proportional Integral Derivative) type of linear control. This type of controller results in a relatively slow response, as the linearized model is no longer valid for aggressive manoeuvres. For the application of wind disturbance rejection a faster nonlinear controller is required to keep the reaction time and the effect of disturbance on the drone minimal. A minimal displacement is crucial for executing a precision manoeuvre such as, for example, landing on a wind turbine blade even in a turbulent environment.

Alternatively, Nonlinear Dynamic Inversion (NDI) is a nonlinear control method aiming for precise and aggressive control by inverting the nonlinear dynamics and imposing linear dynamics on the system. However, NDI is notorious for the dependency on extremely accurate model parameters describing the nonlinear system. For this reason, it lacks the necessary robustness for wind disturbance rejection and unmodeled internal dynamics, which are difficult to model.

Our flight controller uses a technique based on Incremental Nonlinear Dynamic Inversion (INDI): A novel method designed at TU Delft that overcomes the robustness issues of NDI by reducing the dependency on an accurate system model while still allowing for a precise and fast response. Its combination of fast sensor update rates and a nonlinear control law allows for precise and aggressive manoeuvres. The use of acceleration measurements allows the control law to capture unmodeled dynamics, such as gust wind disturbances, and effectively cancel these without any modelling assumptions.

Fault tolerance

In order to increase the reliability and certifiability of drones even more, we are designing fault-tolerant control systems to equip multicopters with an additional layer of safety. Relying on fast detection of rotor failures, our fault-control system is able to steer a damaged multicopter to safety, even after complete loss of one of the rotors. Despite the loss in controllability of the drone, the combination of INDI with a novel in-house developed control technique allows the damaged drone to still land safely.

Software engineering

Our software platform is based on a Linux system, on top of which our own software runs. In addition, a co-processor without an operating system is used to handle real-time tasks. All our software is written in Rust, which—just like us—has reliability as its main focus, while not giving up performance.

Isolated modular software architecture

Software on our platform is split up in separate modules which run as independent isolated processes, such that a bug in one cannot take down other tasks. Even if important modules would crash mid-air, the drone will keep on flying just fine, as modules can be immediately restarted without interrupting communication with others. This way, an unexpected failure in for example the GPS driver will not take down the whole system. This flexible set-up also allows you to easily add your own software modules to the system, without risks.

Communication between the different modules happens directly through shared memory and does not go through the Linux kernel, to improve latency and decrease dependency on specific kernel behaviours. Every single page of shared memory is configured to be only writable by one process, and one process only. All other processes may read from them, and cannot possibly lock out or slow down other processes, as all communication is completely lock free. When a process crashes or is otherwise stopped or restarted, a successor will seamlessly take over without interrupting.

Open source

Fusion Engineering proudly publishes open-source software projects to contribute back to the open-source community.

At this moment, we have published several open-source projects:

- Inline Python — A Rust library that makes it possible to write Python code directly inside Rust code, to combine the best of both worlds. We use this in simulations, to be able to make use of great Python libraries such as matplotlib.
- rust-git-version — A Rust library to embed Git version and status information into your programs.
- intbits — A Rust library making it easier to work with individual bits in integers.
- bcm283x-linux-gpio — A Rust library and command line tool for working with GPIO on a BCM283x CPU.
- spicat — A command line tool for making full-duplex SPI transactions on Linux.

See Fusion-Engineering on GitHub for all our open-source projects.

Projects we contribute to

In addition to publishing our own projects, we also contribute to existing open-source projects that we make use of ourselves:

- Rust — The Rust compiler and standard library. We are actively involved in the development of Rust compiler and standard library and regularly contribute to the Rust project.
- Nalgebra — A library for working with linear algebra in Rust.
- stm32f4xx-hal — Hardware abstraction layer for the STM32 F4 family of microcontrollers.

See Fusion-Engineering-forks on GitHub for many more projects we've contributed to.