ALBATROZ ENGENHARIA
NEWS

Introduction to UAV Autopilot Architecture

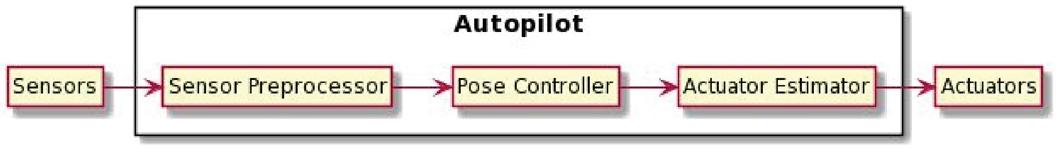
In recent times, there has been a spike in the popularity of Unmanned Aircraft Vehicles (UAVs). These systems inherit a new limitation that's not present in Manned Aircrafts: The communication between pilot and aircraft is indirect, requiring a wireless communication system, bringing its own drawbacks, such as latency and maximum range. One way of surpassing this limitation is to reduce or even eliminate the pilot's intervention using sophisticated flight autopilots.
But how are these autopilot's structured? Before answering that question, we must understand the domain challenges that these autopilots are designed to overcome:

1. They are less stable than their manned counterparts. This is most true on multicopter vehicles but even in fixed-wing aircrafts, the lower weight will make them more susceptible to the effects of wind. As result, a high throughput and low latency are required.
2. UAVs have a very limited maximum payload, most of which is reserved for different sensors, leaving little for the actual autopilot processor.
3. Culture of robustness in the aerospace domain. A side-effect of this is the slowness of adopting new hardware, preferring to use a solution based on its flight heritage.

While the first point indicates the minimum level for the computing power of the autopilot, the next two points forbid us to simply throw more or better hardware at the problem, we cannot simply use a newer CPU or add a GPU on top to solve the problem. As such, we have no choice but to design our architecture as such to use our processing units to their fullest potential.

Bare-bones Architecture

First of all, a preprocessor block is required to handle the multiple sensors, which typically output data at very high rates (well above 1KHz). This is more than the autopilot is able to handle so the measurements are sub-sampled to a more manageable rate, at the cost of some latency. When possible this should be done by the sensor itself rather than by the autopilot. After this, data from different sensors are fused together to get measurements in a usable form: a pose, ie. attitude and position.
Having acquired a pose we can run the controller which will estimate controls that would generate the desired pose given the current one based on a number of gains. This block is very optimized, only requiring very simple mathematical operations. When possible, the control of the attitude and position are separated into different blocks to further reduce the dimensionality of the problem.
Finally there's the actuator block, which will receive the controls and convert them into an actuator input. This usually only requires scaling and offsetting the controls.
This concludes the most essential blocks of an autopilot, resulting in the following diagram:



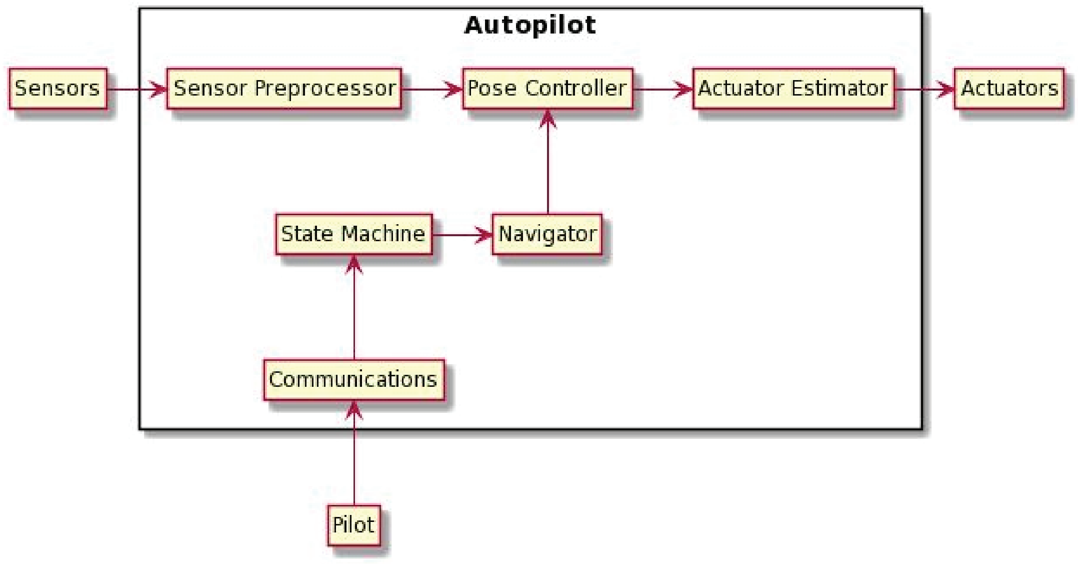
However, this barebones architecture is only able to control our aircraft to a fixed pose and only if already in flight and in similar conditions to the desired pose.

Complete Architecture

A real autopilot will present additional blocks which while not essential, add a great deal of value to the system. These blocks are the following:

- A state machine block, which handles the different steps of flight, from takeoff to the mission and finally to the landing.
- A navigation block, which will control the set-points and gains of the controller based on the current state.
- Finally, a communication block is defined to enable the pilot to set new tasks or abort the current flight.

All these blocks are run at a substantially lower frequency than the bare-bones architecture, therefore having minimal performance implications.



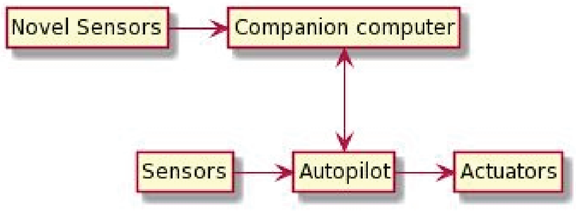
Extensibility

In the previous section we were able to create extra value with little drawbacks by running others tasks at a lower rate. This begs the question: “Can the same be done again?” The answer is yes, but will most likely require the autopilot source to compile the new blocks.
An interesting alternative to this is to enable the connection of a companion computer to the autopilot. Obviously the aircraft must be able to carry the extra weight, but there are many advantages to this approach:

- Add more and/or novel sensors, such a LiDAR or a multispectral camera.
- Setting new tasks can be automated using the new sensors.

- New safety measures may be implemented using these new sensors, eg. the world can be mapped to avoid colliding against an obstacle.

A companion computer architecture will be of the form:



Since the autopilot is unchanged, both the flight robustness and performance are maintained.

For further information contact our commercial department: comercial@albatroz-eng.com
or visit our website: www.albatroz-eng.com