What is Aerostack?

It is a software framework that helps developers design and build the control architecture where we integrate multiple heterogeneous computational solutions such as motion controllers, computer vision algorithms, self-localization and mapping methods.

It is a useful research tool for aerial robotics to test new algorithms and architectures. It is currently an active open-source project. Actually, Aerostack 2.0 is scheduled to release in two weeks.

The software running on the computers onboard UAS can be classified in three main software components:

Firmware: the time-critical management of the hard ware.

Middleware: the time-critical control of the system.

Operative System: the computer-intensive non time critical control and management of the system.

Other architecture frameworks for UAS

“astec\_mav\_framework”: It is developed by the Autonomous Systems Lab in ETH Zurich. It is basically a framework for data acquisition and position control to be used with the high level processors of Ascending Technologies, a Germany-based startup which is now owned by Intel.

“hector\_quadrotor” framework: It is developed by Team Hector from TU Darmstadt. Hector stands for Heterogeneous Cooperating Team of Robots. This framework is mainly used to develop heterogeneous search and rescue robots cooperating with each other.

“Telekyb” framework: It is developed by HRI-MPI (Max Plank Institute). Telekyb stands for Tele-Operation platform of the MPI for Biological Cybernetics. It allows fully autonomous multi-UAS navigation. The main drawback of this powerful framework is that there is no provision for the user to change the architecture design to incorporate new features.

“Paparazzi” project: It is developed and used by ENAC and MAVLAB – TU Delft. This project includes not only the software framework but also the hardware autopilot and sensors and it is not compatible with any other commercial hardware.

“Twirre” architecture: It is developed by NHL Computer Vision proposes a hardware and software design. It is focused mainly on hardware and it does not report a high level of autonomy.

PX4 Flight Stack and APM Flight Stack: These two are definitely the well-known open-source commercial projects. There has always been a debate of sorts between which to choose. As far as I learnt, PX4 seems to be the better option.

Why Aerostack?

Aerostack provides two main features:

1. A complete multi-layered architectural organization to support fully autonomous flights.
2. A versatile software framework for multiple uses.

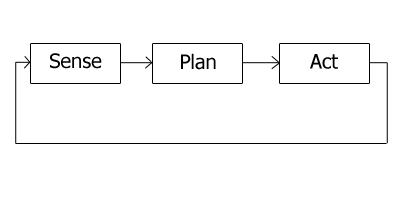
The multi-layered architecture includes both low-level layers for reactive behavior and high-level layers for intelligent behaviors. At the low-level, Aerostack provides a number of specialized reusable components for visual perception, motion controllers, etc. At the high-level, Aerostack includes a number of components to provide a high degree of autonomy and self-adaptation in complex and dynamic environments with fault management procedures to increase the degree of safety.

The versatility of Aerostack is based on the following two main features. On the one hand, Aerostack is flexible enough for a wide range of applications from tele-operated flights of single UAS to highly autonomous missions of multi-robot UAS platforms. On the other hand, Aerostack is hardware independent. Aerostack is focused on software development that is designed to work as part of the operative system, which means that it requires the appropriate hardware design as well as its proper firmware and middle-ware software components.

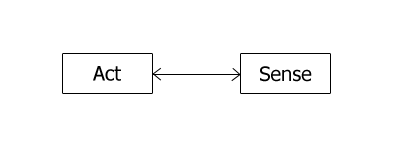
Aerostack System Architecture

It follows the Hybrid deliberative/reactive paradigm. The three different paradigms are as follows:

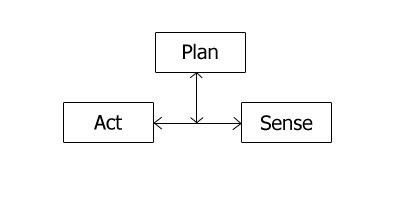
1. Hierarchical/deliberative paradigm: The robot operates in a top-down fashion, heavy on planning. The robot senses the world, plans the next action, act; at each step the robot explicitly plans the next move. All the sensing data tends to be gathered into one global world model.



1. The reactive paradigm: Sense-Act type of organization. The robot has multiple instances of sense act couplings. These couplings are concurrent processes, called behaviors, which take the local sensing data and compute the best action to take independently of what the other processes are doing. The robot will end up doing a combination of behaviors.



1. Hybrid Deliberative/reactive paradigm: The robot first plans how to best decompose a task into subtasks (mission planning) and then what are the suitable behaviors to accomplish each task. Then the behaviors start executing as per the reactive paradigm. Sensing organization is also a mixture of hierarchical and reactive styles; sensor data gets routed to each behavior that needs the sensor, but is also available to the planner for construction of a task-oriented global world model.



Components of Aerostack

The architecture is formed by N heterogeneous robotics agents and the human operators. Every robotic agent shares the same layered architecture, although it can have different component implementations as well as different hardware. The first three layers correspond to the popular hybrid design known as the three layer architecture.

Reactive layer: Low-level control with sensor-action loops. The reactive layer is a sensor-action loop that includes feature extractors (in the feature extraction system) and motion controllers (in the motor system). Feature extractors may read simple states of sensors or may implement complex vision and pattern recognition algorithms (signal processing, recognition of objects and basic relationships). Examples of feature extractors are: read bumper, extract color, compute centroid of an image, recognize visual marker, detect power line tower, etc. A feature extractor can integrate a set of perception procedures using different combination methods (e.g., fusion, sequence, etc.). Motion controllers typically implement combinations of Proportional-Integral-Derivative (PID) controllers (e.g., cascade controllers). For example, these type of controllers can accept orders about a desired value for a variable (position, speed, altitude, and yaw) in form of single commands or simultaneous commands that are translated into low level commands to be sent to actuators.

Executive layer: It is also called as sequencing layer that accepts symbolic actions from the deliberative layer and generates detailed behavior sequences for the reactive layer; this layer also integrates the sensor information into an internal state representation.

Deliberative layer: It generates global solutions to complex tasks using planning (e.g., planning optimal trajectories). The reactive layer functions in the present while the deliberative layer uses information from the past and projection to the future.

Reflective layer: To simulate certain self-awareness able to supervise the other layers. The reflective layer helps to see if the robot is actually making progress to its goal and to react in the presence of problems (unexpected obstacles, faults, etc.) with recovery actions.

Social layer: It has communication abilities, as it is proposed in multi-agent systems and other architectures with social coordination. In this level is important to establish an adequate communication with human operators and other robots.

Ontology of Aerial Robotics

In order to facilitate the semantic interoperability of the different components, we use an ontology for aerial robotics that has been defined specifically for Aerostack following common terminology found in the research literature about robotics and aerial systems. The ontology defines the formal and explicit specification of shared concepts.

For example, the notion of skill is useful as an intuitive concept to help operators express more easily what complex abilities should be active for a particular robot. A skill is automatically translated to a set of running processes. Skills differ from actions in that an action (e.g., take-off, go to a point) finishes by itself when it reaches the goal, and a skill (e.g., interpret visual commands, avoid obstacles), once it is activated, it is permanently active without any limit of time, until it is deactivated by an external influence (e.g., the mission planner, the operator, etc.).

Autonomy and self-adaptation in complex environments

Aerostack includes components for a reactive behavior that provides certain level of autonomy. In addition, Aerostack also includes components to increase the degree of autonomy to operate in complex and dynamic environments.

The different dimensions of autonomy:

1. Human independence. The robot can be operated with simple commands from general human operators and does not require highly specialized operators and technical jargon.
2. Dynamic environment. The robot can operate in dynamic and complex environments with unexpected situations where it is required abilities such as: self-adaptation, threat avoidance, and self-diagnosis, fault tolerance, etc.
3. Complex missions. The robot can perform complex missions (such as search and rescue missions) where situation awareness and complex planning is required.

Aerostack provides the following components related to these levels of autonomy:

1. The planning system automatically generates the goals in order to accomplish a particular mission. Aerostack includes a task-oriented mission planner and other more specific planners (trajectory planner and yaw planner). This planning system helps to increase autonomy in terms of human independence, complex missions and self-adaptation in dynamic environments. Using Aerostack, the operator can specify in advance a mission using an xml-based language and the mission planner interprets such a specification to generate step by step actions to be done. Event handlers can be defined using condition-action rules to react in the presence of certain exceptional situations. Exceptions can change the execution of a mission by requesting additional actions or skills and by using particular commands (abort mission, abort task, etc.). Aerostack’s mission planner follows the approach called reactive planning that differs from classical planning in the way that it computes just one next action in every instant, based on the internal state (about goals, tasks, etc.) and the external world state. This simplification is useful to cope with highly dynamic and unpredictable environments.
2. The executive system includes a behavior manager that accepts directives from the deliberative layer and sequences them to be performed by the reactive layer. The behavior manager translates requested actions and behaviors expressed as symbolic descriptions (e.g., takeoff, move to a point, etc.) into specific orders for the motion controllers and the activation of certain running processes. The executive layer helps to increase autonomy in terms of human independence and complex missions.
3. The supervision system is a key functional package that ensures a correct behavior of the robot. The supervision system helps to provide self-adaptation and fault-tolerance. This typically consists of three steps: failure detection, notification, and recovery. The supervision system includes the following processes: the action monitor and the process monitor for failure detection and notification, and the problem manager for fault recovery. The supervision system helps to increase autonomy in dynamic environments providing self-adaptation and fault tolerance. Within the supervision system, the action monitor supervises the execution of a requested action and informs when the action has been completed or failed. This is important to simulate self-awareness of the degree of completion of goals to be able to notify it to the adequate destination (to the operator, to the mission planner, etc.). The process monitor supervises the execution of different processes and is responsible for acquiring and informing about the errors produced by such processes. The process monitor verifies that each process is alive using a watchdog technique and to get the states of processes. The process monitor collects errors produced by processes in order to notify in a uniform way these errors to other components.

Applications of UAV

* UAV Inspection and Monitoring
* UAV Surveying and Mapping
* UAV Condition Survey and Civil Engineering
* UAV Precision Agriculture
* UAV Aerial Imaging
* UAV Computer Vision
* UAV Flight Dynamics
* UAV Swarming
* UAV SLAM