Flexible framework for the development of versatile MAV systems for multi-disciplinary applications

M. Cordero*, M. A. Trujillo, J. Ruíz, A. Jiménez, L. Díaz, and A. Viguria
Center for Advanced Aerospace Technologies, Wilbur y Orville Wright 19, La Rinconada (Seville, Spain)

ABSTRACT

MAV (Micro Aerial Vehicle) systems have proven useful in multiple applications (e.g. aerial photogrammetry, aerial inspections,...). For each application different sensors and even different airframes are often needed. Having a flexible UAS (Unmanned Aircraft System) devices (specifically an autopilot, a payload manager and a ground control station) that can be seamlessly used with different platforms is of great interest as it reduces the development time and effort (and hence cost). This flexibility must apply to the hardware, software and design methodologies so the system can be rapidly adapted to meet the requirements of other applications. This paper introduces a flexible system developed by CATEC consisting of an autopilot, Ground Control Station (GCS) and a payload manager for MAVs. This system has already been tested in real experiments for different applications that are also presented in this paper.

1 INTRODUCTION.

Unmanned aircrafts are finding an increasing number of applications such as precision agriculture, firefighting, mining, search and rescue, surveillance,... to cite a few. Depending on the mission requirements, different aircraft systems may be needed (e.g. more autonomy may be needed for the inspection and mapping of wider areas). Furthermore, different sensors need to be used in different applications (e.g. multispectral cameras are very useful in precision agriculture applications but may not be needed in search and rescue missions). Traditionally, UAS (Unmanned Aircraft systems) have used their own ad-hoc systems (i.e. autopilot, payload manager, Ground Control Station (GCS), etc). These systems are specific for their aerial platform and cannot be easily adapted to be used in another one. Dealing with multiple different UAS systems increase operating and maintenance costs. It is of great interest to have flexible and versatile autopilot, payload manager and ground control station that can be used with different aerial platforms and for different missions. In the last decade different autopilots and GCS systems that can be easily integrated in different aerial platforms have been developed. Most of these autopilots are COTS (Commercial Off-The-Shelf) products with a very limited access to their internal operation (e.g. Piccolo from Cloud Cap Technologies [1], MP2x28 from MicroPilot [2],...). There are also open source autopilots that provide full access to the autopilot's internals. The most popular ones are probably Ardupilot [3] and Paparazzi [4]. These autopilots implement the GNC (Guidance, Navigation and Control) algorithms directly in C++ code. This can be tricky for some control engineers not used to work with programming languages. In this paper, a flexible and versatile UAS system developed by CATEC is presented. This system includes an autopilot, a GCS and a payload manager. Regarding the autopilot, the control engineers follows a model-based development (MBD) approach in which the GNC algorithms are developed in Simulink and ANSI C code is obtained using an automated code generation tool from TheMathWorks. The GCS and payload manager also use flexible architectures as will be shown. This paper is organized as follows: in section 2 a description of the system and the individual components (i.e. autopilot, GCS and payload manager) is provided; in section 3 the flexibility and versatility of this system is illustrated by showing three applications in which the system has already been used; finally in section 4 the conclusions are presented.

2 SYSTEM ARCHITECTURE.

The proposed system architecture is depicted in Figure 1. The elements of the ground and the airborne segments are shown in red and blue respectively. Additionally, these elements can be divided in those that are used for GNC tasks and those that are used for mission specific tasks. The autopilot, GCS and navigation sensors belong to the first group while the payload sensors, payload manager and payload ground software belong to the second group. The elements of the ground and the airborne segments communicate using two radiolinks operating at different frequencies (e.g. 900 MHz for the command and telemetry link and 2.4 or 5.8 GHz for the payload datalink). It is important to note that the elements of the navigation and mission payload groups are not isolated. Instead, some of them are interconnected (e.g., the autopilot is connected to the payload manager to provide it with telemetry such as the position, velocity and attitude of the aircraft that can be exploited for georeferencing sensor data).

*Email address: mcordero@catec.aero

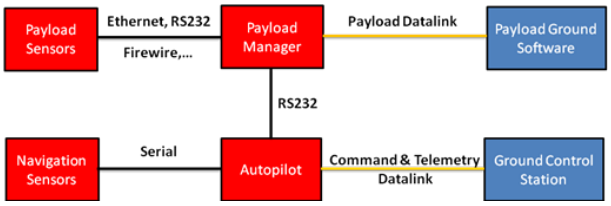

Figure 1: System architecture developed for the precision agriculture application.

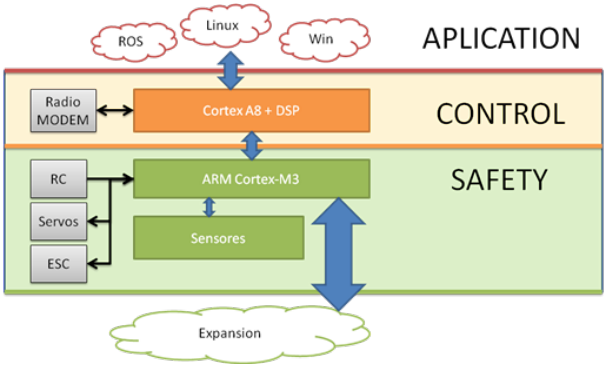

Figure 2: Autopilot functional architecture.

2.1 *Autopilot.*

The autopilot controls the throttle and control surfaces to safely fly the aircraft according to the mission configured by the operator by means of the Ground Control Station (e.g. following a given path of waypoints). For this purpose, a navigation module is used to process the information provided by the navigation sensors (i.e. inertial sensors, magnetometers, GNSS (Global Navigation Satellite System) receiver and differential pressure sensors) to estimate the aircraft state (i.e. position, velocity and attitude). From this estimation of the aircraft state and the established mission, the control signals are calculated and sent to the aircraft servos and engine throttle. CATEC's autopilot is divided in two boards: a high-level board (or control board) and a low level board (or safety board). The control board is based in an ARM Cortex A8 core running at 720 MHz with 256 MB of RAM and runs the GNC (Guidance, Navigation and Control) algorithms over QNX operating system. The safety board is in charge of collecting the data from the navigation sensors and sending the control signals to the actuators. This board is based on an ARM Cortex M4 microcontroller running at 80 MHz using FreeRTOS as the operating system. This board also implements safety features including a light navigation and control module that allows the aircraft to safely fly according to a return to home mission. The autopilot also includes the interfaces for communicating with other systems (e.g. a radiolink for telemetry, a RS232 interface for connecting with the payload manager,...) and PWM (Pulse Width Modulation) outputs that to control the aircraft control surfaces. The functional and hardware architecture of the autopilot are shown in Figure 2 and Figure **??** respectively.

The autopilot software components that implement the GNC algorithms have been developed following a model-based development approach using Simulink [5]. This approach is based on automatically generating standard ANSI C code from the Simulink models. This code can be compiled for QNX or FreeRTOS and run in an embedded PC. With this framework the development workflow can be accelerated as you can develop a model with the GNC algorithm, generate code and run it in the embedded PC and check its behaviour and performance. When an error is detected or any change needs to be applied the Simulink model can be modified and the code can be generated again to repeat the iterative process of develop-and-test until everything works as expected. For monitoring the behaviour of the system, Simulink External Mode is a extremely helpful tool. This tool provides communication between the Simulink model and the running generated code so any signal from the model can be monitored e.g. using a scope or logging the data for postprocessing.

2.2 *Ground Control Station (GCS).*

The GCS software runs in a laptop PC and is used by the aircraft operator for commanding waypoint paths, monitoring telemetry, logging flight data and managing alarms. CATEC's GCS has been developed using Qt open source frame work in C++ programming language. A Model-View-Controller (MVC) software architecture is used to decouple the code components related to the data model from the code components of the graphic part and from the internal software logistic. The functionality of the set of basic libraries that conforms the core of the GCS can be easily expanded thanks to a plug-in system. A number of plug-ins is also provided together with a simple API so third-party developers can use their own plug-ins. A plug-in manager is in charge of loading/unloading all of them. The following plug-ins are available:

- States plotter. This plug-in plots the received telemetry data for debugging and monitoring purposes.
- Logger. This plug-in logs the received telemetry data in a file in CSV format so it can be processed later.
- Primary Flight Display. This is a graphic plug-in developed in GLStudio (with a Qt/C++ wrapper) that shows the aircraft attitude, speed and flight altitude.
- Payload Remote Manager. This plug-in allows the configuration of the payload sensors from the ground. This will be analyzed in more detail in the following section.
- Waypoint Editor. This plug-in is used for the edition of paths of waypoints that are uploaded to the aircraft autopilot.

Finally, the View component is the one that shows the graphical interface for the aircraft operator. It is completely independent from the core GCS software so different view components can be easily interchanged. The layout can be easily modified, docking and undocking the different components of the view. Figure 3 shows the graphical user interface of CATEC's GCS.

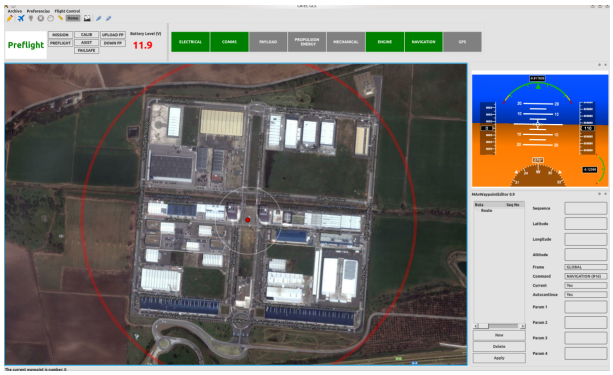

Figure 3: GCS graphical user interface.

2.3 *Payload manager.*

The payload manager is in charge of managing the different elements that are part of the aircraft payload. It performs multiple tasks including sensor data collection, sensor configuration, status monitoring and it also manages the communications with the GCS. As mentioned in the previous section, the GCS is used for controlling the sensors operation during the flight by sending commands to them via a radiolink. These commands are received by the payload manager which processes them and performs the associated actions. These commands can for example start or stop the capture of video data, change the frame rate and/or image resolution, activate or deactivate the transmission of the collected data to the GCS, etc. The flexibility of the system makes it easy to implement new configuration commands. The payload manager also monitors the status of the different elements of the payload and reports them to the GCS. The collected data can be locally logged in flash memory and/or can be sent to the GCS during the flight by means of a radiolink. Additionally, some data processing can also be performed onboard before logging or sending them to the GCS. For example, is a common practice to add a tag to the collected data with the state of the aircraft (i.e. position, velocity, attitude, etc) in the instant when that that was collected and a time stamp. With this information, collected data can be exploited e.g. for mosaicing, DEM (Digital Elevation Model) generation, etc. Depending on the mission and the application, different sensors can be needed on board the aircraft e.g. visual and infrared cameras, multispectral cameras, LIDAR (Light Detection and Ranging), sonar sensor, etc. The system has to be flexible both in hardware and software for being able to manage the different sensors and performing the different tasks of each mission.

Regarding the hardware the payload manager developed by CATEC is based on a Falcon VL-EPU-2610 embedded PC board from VersaLogic [6]. This board includes an Intel Atom E6x0T processor running at 1.6 GHz, 2 GB RAM module and multiple interfaces including four USB ports, one serial RS-232 port and one Gigabit Ethernet interface. It also includes a miniPCI slot so new interfaces can be added easily. Figure 4 shows the payload manager installed onboard an autonomous helicopter next to an infrared camera.

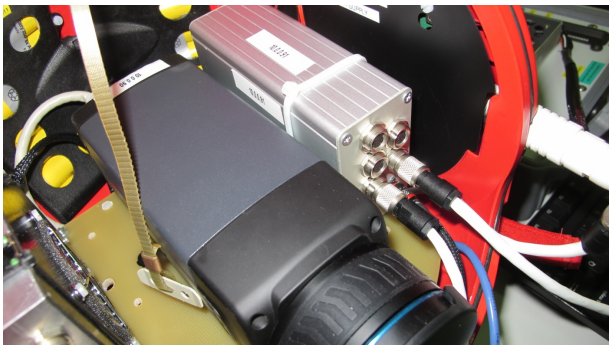

Figure 4: Payload manager system and IR camera on-board an unmanned helicopter.

Regarding the software architecture, the payload manager is entirely based in ROS (Robotic Operating System) and Qt and runs over a Linux distribution. ROS is an open source framework widely used in the robotics community and increasingly in the industry. This framework facilitates the development of modular applications so different modules (which are referred to as nodes in ROS terminology) can be developed independently and added for new functionalities. The Qt framework has been selected for its large set of multi-platform tools for handling files, threads, processes management, UDP/TCP communications, etc. Communication between ROS nodes is based in the publish-subscribe paradigm. This paradigm is based on the definition on different data topics (e.g. images, telemetry data, node status, etc). The different existing ROS nodes can publish on these topics and/or subscribe to them for sharing information. Figure 5 shows the payload manager software architecture. Red nodes are part of the payload manager and hence run onboard the aircraft while the blue node is the payload manager plug-in of the GCS. The system includes the following nodes:

- IR camera node is a mission specific node for infrared cameras. It handles the connection with the IR camera via an Ethernet connection satisfying the GenICam standard. This node publishes an IR-image topic each time the camera captures a thermo-graphic image. It also provides some services for changing some of the IR camera parameters so other nodes can change them.
- Visual camera node manages webcam devices compat-

ibles with the video4linux driver. This node publishes an RGB-image topic each time the camera captures an image. It also provides services for changing some parameters of the camera such as brightness, contrast, FPS (Frames per Second), etc.

- Streamer node subscribes to the image topics of any sensor node (e.g. the IR and/or visual cameras) in order to transmit them to the GCS software via an UDP connection and a wireless link. This node can use different video codecs for compressing the images in a video stream in order to reduce the bandwidth usage. The stream sent to the GCS can consist in one or more video streams and it can also include a metadata stream with additional information of each frame.

- Telemetry node accesses the telemetry data provided by the autopilot via a serial connection. For each data acquisition a telemetry topic is published with information related to the platform position and attitude. This node doesn't provide any service.

- Logger node doesn't publish any data topic but it subscribes to all the data topics provided by other sensor nodes in order to log them in a database file (shown in green in Figure 5). It provides basic services for starting or stopping the logging of topics. After each logging procedure, a BAG file is stored in the storage unit of the embedded system. BAG is a ROS-specific format for compressed data that can be easily read and processed with the tools provided by ROS.

- Comms node manages the communications with the GCS payload manager software. On one hand, it implements a TCP server that handles incoming connections from the GCS for listening to operational commands. These commands are then translated to the specific service requests to the target ROS nodes. In the other hand, the node is sending periodically the status of the nodes to the GCS payload management software.

- Monitor node collect the status topic published by all the other nodes. This topic indicates if they are working properly or not. The purpose of the monitor node is to launch the others, monitor their status and if something goes wrong restart them.

ROS nodes implementing the interfaces of topic publications, subscriptions and services defined in Figure 5 could be immediately replaced by other nodes with the implementations for different hardware devices providing the system with high flexibility.

3 APPLICATIONS.

CATEC's system has been tested in several applications proving its versatily. In this paper three of these applications

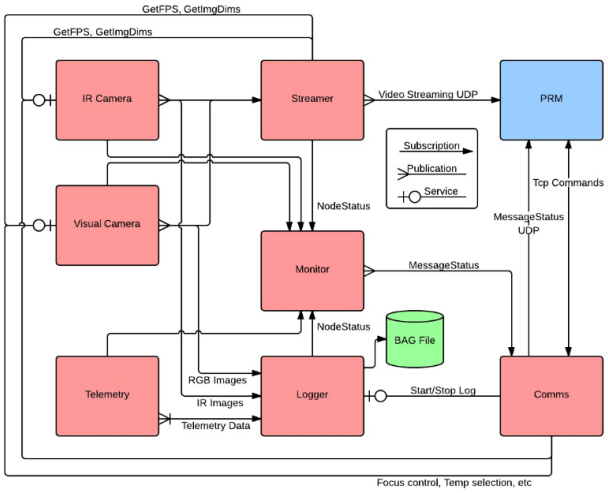

Figure 5: Application software modules. Red nodes run onboard the UAV and blue ones run in the GCS payload management software.

(aerial robotics manipulation, precision agriculture and wind blade inspections) are presented to illustrate the flexibility of the system.

3.1 Aerial robotic manipulation.

This work was developed as part of the European Commission's 7th Framework Programme (FP7) project ARCAS [7]. The objective of this project is to integrate robotic manipulators onboard of aerial robots for structure assembly tasks. In the indoor tests, a robotic arm developed by CATEC was integrated with an octocopter. CATEC's autopilot and payload manager were both integrated onboard the octocopter. During the indoor flights, GNSS signals are not available so positioning information was obtained through a motion caption system by Vicon which is depicted in Figure 6. This system is based on the real-time processing of the information provided by 20 Vicon cameras that can provide the position of an object with sub-millimetric accuracy. The system only needs that passive markers are attached to the objects to be tracked. The positions are calculated by a Vicon central computer and sent to the autopilot in real time by means of a WiFi connection. The ability to substitute the GNSS positioning by Vicon positioning for indoor flights shows the flexibility of the CATEC's autopilot. The payload manager performed the already mentioned tasks of sensor data acquisition, logging and streaming and sensor configuration. In addition, the payload manager included a ROS node that implemented a high-level controller that provided the autopilot and robotic arm controller with position references for the octocopter and the robotic arm respectively. This way, difficult tasks such as coordinated control of both the aerial vehicle and the onboard robotic arm can be achieved.

Two different experiments were carried out. In the first

Figure 6: Vicon camera (left) and CATEC indoor testbed (right).

one, the octocopter had to pick up a bar estimating its position from four visual markers that where attached to it as shown in Figure 7. A visual camera was integrated onboard together with a light source to guarantee good illumination conditions for the vision algorithm to work properly. In the payload manager, a ROS node was in charge of the acquisition of the visual data and another ROS node processed this data conveniently to provide the autopilot and the robotic arm controller with position references. With this architecture the final task of autonomously picking up the bar with information from the visual camera was achieved successfully.

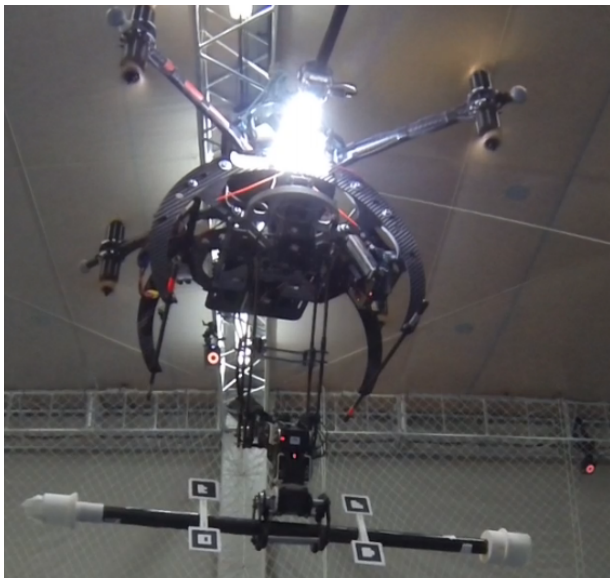

Figure 7: Grabbing a bar with visual markers.

In the second experiment the octocopter had to autonomously grab three bars with special connectors and sequentially assemble them to form a structure. In this case, Vicon markers where attached to the bars so Vicon system calculated their attitudes and positions and sent them to the ROS node implementing the high-level controller. As in the previous case this node provided the autopilot and robotic arm controller with position references. This experiment was also conducted successfully. Figure 8 shows two frames of the video that was recorded during the flight trials.

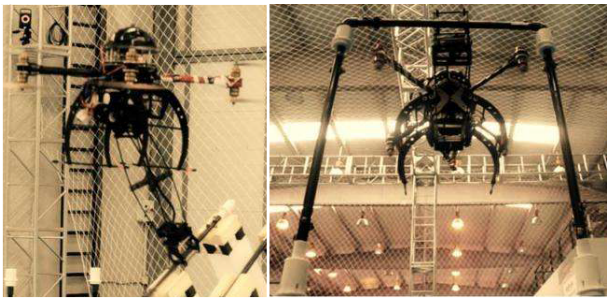

Figure 8: Octocopter grabbing a bar (left) and assemblying the structure (right).

3.2 Precision agriculture.

This work was developed as part of the European Commission's 7th Framework Programme (FP7) project Fieldcopter [8]. The objective of this project was to investigate the added value of Unmanned Aerial Systems (UAS) over satellite-based remote sensing. The principal benefits of UAS are their flexible deployment at any time in the day and their ability to acquire imagery under cloudy conditions. An infrared and visual cameras were integrated onboard an unmanned helicopter (specifically a SARAH UAV manufactured by Flying-Cam). It is a completely electrical helicopter with 7 kg payload capacity. It has around 20 minutes of autonomy and its operation is completely autonomous, including taking-off and landing. This UAV uses a commercial autopilot so only CATEC's payload manager was integrated together with visual and infrared cameras as was shown in Figure 4. Additionally, CATEC's GCS was used for payload management purposes. The experiments were conducted in vineyards in Lleida (north of Spain). In this case, the payload manager stored the collected visual and IR images in the onboard storage device. After the flights, the data were postprocessed to obtain mosaic images and segmentation of the individual vines as shown in Figure 9.

3.3 Wind blade termographic inspection.

Wind turbines are gaining importance in the last years because of their high efficiency during energy production without greenhouse gas emission. However, wind blades maintenance operations have become expensive due to operating costs (accessibility, scaffolding,...) and the cost of having them unemployed. A non-destructive technique capable of detecting most significant in-service defects in composite wind blade is Infrared Thermography (IRT). In this work, a feasibility study for defect detection during maintenance operations in wind blades using unmanned aircrafts was performed. This approach allows inspecting the blades without

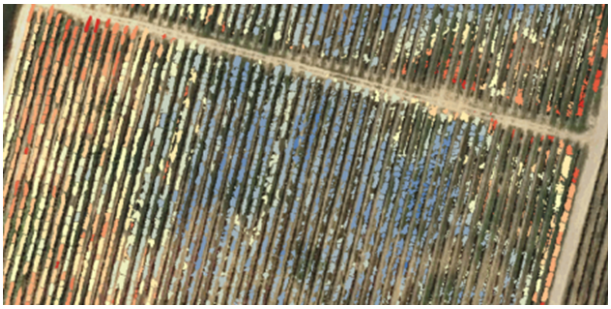

Figure 9: Canopy temperature obtained from flight experiments showing the obtained object-segmentation of the individual vines.

any scaffolding system and reduceing the time they remain unemployed.

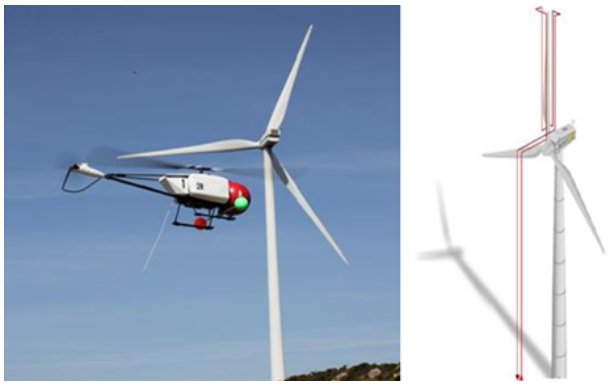

Figure 10: UAV flights close to the wind turbine to be inspected (left); Trajectory followed for the inspection (right).

In this case, a similar configuration to that used in the precision agriculture experiments was adopted. An infrared camera was integrated onboard the SARAH unmanned rotorcraft. To obtain an adequate image resolution and to keep a safe operation area for the UAS, a distance of about 5 meters from the wind turbine blade and displacement speed of 1 m/s were defined. The inspection strategy (shown in Figure 10) was designed to achieve the entire blade inspection using a single battery (whose autonomy is around 20 minutes).

The acquisition and logging of the infrared images was controlled by CATEC's payload manager and monitored with the payload manager plug-in of the ground control station. The feasibility of the proposed inspection methodology was demonstrated for detecting the most common wind blade service defects such as cracks, delaminations and impacts, by using passive infrared thermography. Figure 11 shows some of the thermographic images obtained during the flight trials performed in Tarifa (south of Spain) in which 10-meter blades were inspected.

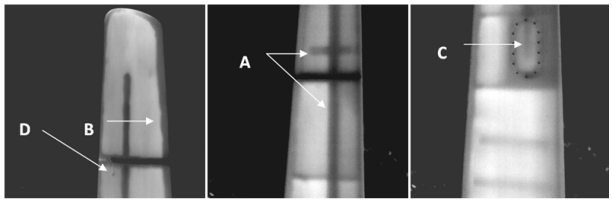

Figure 11: Thermal images showing (A) pitch brake system, (B) bonds between upper and lower shell, (C) access gate and (D) surface roughness.

4 Conclusion.

In this paper, the UAS systems developed by CATEC for unmanned aircrafts have been presented. These system have been developed following a flexible architecture both in hardware and software. This allows them to be easily adapted for their use in different missions and applications with different requirements, as well as being integrated in different aerial platforms. The systems include an autopilot, a payload manager and a ground control station with remote payload management functions. These have been used in real flight demonstrations for different applications. Three different applications have been shown, namely aerial robotic manipulation for structure assembly, precision agriculture and wind blade termographic inspection.

References

[1] CloudCapTechnologies. Piccolo II Datasheet.

[2] MicroPilot. MP2x28 Series Autopilot http://www.micropilot.com/products-mp2028-autopilots.ht.

[3] Ardupilot. http://ardupilot.com/.

[4] Paparazzi UAV Autopilot. http://wiki.paparazziuav.org/wiki/Main_Page.

[5] Daniel Santamaría, Francisco Alarcón, Antonio Jiménez, Antidio Viguria, Manuel Béjar, and Aníbal Ollero. Model-Based Design, Development and Validation for UAS Critical Software. *Journal of Intelligent & Robotic Systems*, 65(1-4):103–114, August 2011.

[6] VersaLogic. Falcon (VL-EPU-2610) http://www.versalogic.com/products/ds.asp?productid=2

[7] ARCAS. http://www.arcas-project.eu/.

[8] Fieldcopter. http://fieldcopter.eu/.