CONCEPTION OF AN UAV GENERIC MISSION SYSTEM

Damien Poinsot, Jacques Lamaison, Alain Hostallier, Caroline Bérard
ONERA-DCSD and SUPAERO 10, avenue E. Belin, 31055 Toulouse, France

Keywords: *UAV, simulator, flight software, guidance, navigation, control*

Abstract

An UAV is a flying vehicle equipped with an onboard mission system which grants it full autonomy by operating guidance, navigation and control tasks. The process involves three steps : guidance equipment and software first compute the aircraft trajectory required to satisfy the mission objectives, navigation then tracks the vehicle's actual position and attitude, and flight control then transports the aircraft along the required flight path in order to accomplish the mission. Functions of the mission system also include communication with the operator and the ground station before, during and after flight. Theses communications consist in receiving commands from the operator or sending back status information and mission data. Supaero is currently handling in parallel various UAV projects of different concepts (helicopter, aircraft, unidentified...) but whatever the flying platform is, the requirements in terms of mission system remain similar ; only the specific command laws have to be changed from one vehicle to another. The aim of this study is to define and present the development of such a generic mission system.

1 Introduction

Guidance, Navigation and Control (GNC) consists in the ability to compute onboard an optimal flight path according to the mission objectives and the aircraft estimated position, and then to follow it with maximum accuracy and in-flight stability so as to carry out the UAV observation mission. To meet these requirements, the mission system must provide first of all an accurate and complete way of measurement, efficient means of calculation with a powerful central processing unit (CPU) and flexible ways of communication with the ground station system (which means both uplink and downlink).

The hardware components must be light, performant, and moreover easy to implement and easily transferable to different types of UAVs. These constraints have to be linked with the high targeted modularity. For instance the sensor system should be able to provide its measurement values to the CPU whatever the configuration and attitude of the the UAV is. It should also work properly under strong electromagnetic and vibration perturbation, and provide its data fast enough to follow the fast movement of an aircraft with the high precision required to stabilize an helicopter in stationary flight.

2 The workstation

The mission system is integrated in the common environment for all UAVs developed at Supaero, whose workstation is described on figure 1. The mission system definition also includes the definition of the user interfaces before, during and after flight :

2.1 Flight Software upload

The way to implement the flight software from the pocket PC to the main memory. This connection can be made wireless or not, but this has to allow the possibility to implement separately the device drivers and the command laws - which depend on the vehicle configuration -, and then the laws' coefficients on a separate table for instance.

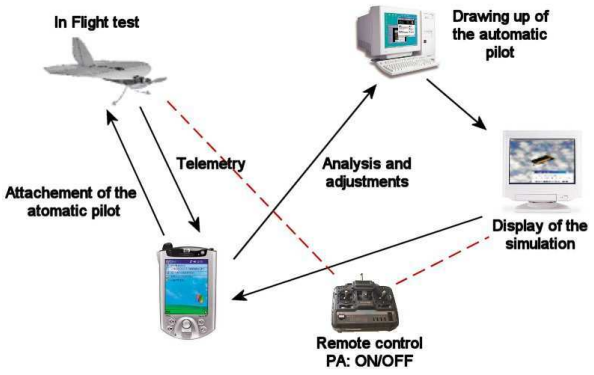

Fig. 1 Workstation for the mission system

This is to enable on-the-field simple adjustments between two test flights. Currently this link is operated through a serial link directly situated on the CPU chip.

2.2 Telemetry and user interface

The mission system features a wireless link through a modem in order to send test data to the ground station (a pocket PC) all along the flight. The UAV emits at a defined sampling period a table of information concerning the system (3-axis velocities, attitude, airspeed, altitude, GPS position, distance to next waypoint...) to allow a post-treatment of the test. Telemetry is displayed on the pocket PC in real-time, so that the system operator can have instantaneous access to the aircraft status.

An incoming upgrade of the user interface will be the replacement of the pocket PC by a tablet PC. This device features a wide screen, a powerful CPU and video chip, which will enable to display in real-time a virtual cockpit of the UAV and video transmission from the onboard camera, once installed. It has also a very intuitive user interface with a touch screen and a stylus, which will be easier to use on the field.

2.3 Remote control

The mission system consists of two operational modes: auto and manual. The avionics software have been designed to have an autonomous flight capability in the automatic mode ; however for development purposes or whenever safety is involved a ground-based operator may control the UAV using a standard radiomodelism RF emitter. The emitter works on 4 channels in PCM mode at 41.000 kHz : 1 for thrust control, 2 for the left and right elevon servos and 1 for auto/manual commutation. In automatic mode the onboard CPU calculates and applies commands to the 3 actuators ; however the CPU has access to the operator commands which can be used for development purposes to study step response or perturbation tolerance of the automatic pilot. In manual mode the operator commands out of the onboard receiver are directly forwarded to the actuators. The CPU is then put out of the control loop and for the operator the UAV becomes a standard radio-controlled aircraft.

At the current stage of development, take-off and landing manoeuvres are still performed in manual mode. When switched to autopilot the UAV performs loops around its predefined waypoints at its initial altitude until switched back to manual mode for landing. The CPU directly reads the commands out of the receiver unit and commutes from auto to manual mode by itself, but for safety reasons the auto/manual command is operated by a redundant remote-controlled switch, separated from the CPU chip. We made the choice to bypass the software with a highly reliable hardware device for operating safety reasons, as the manual mode is considered as a failsafe mode which has to be available every time during the flight.

Even in automatic mode the radio-controller commands are read by the CPU (PWM inputs). This is allows to define semi-autopilot modes, for development purposes. This aspect will be a major issue for the validation of the UAV Autopilot through flight tests. It allows the operator while in autopilot mode to make different kinds of step (velocity step, angular step or height step) to study the real response of the UAV to a precise perturbation and to compare it with simulation results.

2.4 Simulation

The UAV physical model is entirely simulated in a Matlab/Simulink environment. First of all this model is used to define and set up the different control loops of the autopilot. In a second time it is used for flight software validation using S-fonctions in the same simulink model (see part 4). Finally this simulation tool is used for flight test data analysis, which eventually leads to improvements of both the physical model and the autopilot.

3 Onboard electronics

The system hardware architecture is presented on the block diagram of Fig. 2 for the Drenalyn UAV, which is an airplane of a flying wing design, featuring one engine and two elevons that replace both elevators and ailerons. It is possible to adapt this hardware architecture to any of the other aircraft designs by changing the number of actuators - servos and variators -. Its different subsystems are described below.

3.1 Central Processing Unit (CPU)

The core of the system architecture, as shown on Fig. 2 is a Motorola MPC 555 Central Power Unit. This is a high speed, 32 bits device which features a floating point unit designed to accelerate the advanced algorithms necessary to support complex applications. It is powerful enough to perform calculations to the benefit of the IMU and ensure a 10 Hz control loop with a comfortable safety margin. The flight software coded in C language is directly implemented on the 4MB flash memory of the the CPU chip.

3.2 Inertial Measurement Unit (IMU)

In order to control its attitude and to determine its position in a terrestrial reference, the UAV needs to embark an Inertial measurement Unit (IMU). The IMU relies on an MT9 motion tracker coupled to 2 pressure captors and a GPS receiver to compute the estimated aircraft position and attitude.

The MT9 is an highly integrated microdevice that features 3 accelerometers and 3 gyrometers to measure the aircraft linear and angular accelerations, and 3 magnetometers to determine the norm and the direction of the local magnetic field vector. It weighs less than 35 g ; miniaturization to that extent has been made possible by the use of MEMS technology. These sensors work however in a disturbed environment due to strong vibrations and electromagnetic field induced by the electric engine and onboard RF antennas. The sensors' very small sizes make them likely to be very sensitive to external perturbation ; their measurements require proper noise filtering.

One pressure captor measures the static pressure to determine the aircraft altitude (altimeter) while the other captor measures the dynamic pressure to infer the aircraft airspeed (pitot tube). Derivation of the static pressure gives the vertical climb rate (variometer). The GPS receiver provides 3D positioning and absolute velocimetry at 4Hz in a terrestrial reference. GPS data are essential for navigation purposes but of little interest as far as aircraft stability is concerned. The pitot tube remains the only way to determine the UAV's airspeed, which is the relevant parameter for airplane control as the local wind vertical profile is unknown. Static pressure measurement provides once calibrated a very accurate estimation of the altitude (it can detect altitude shifts of a few centimeters only). Attitude angles, airspeed, altitude and climb rate (enables an estimation of the angle of climb) are the parameters used for command computation in the 10 Hz control loop.

The GPS receiver and the MT9 motion tracker are directly connected to the Central Processing Unit by typical RS232 serial links ; the MPC555 processor offers enough serial communication ports for that. Theses sensors can be defined as "intelligent sensors" because they provide a communication protocol. The software must establish communication to configure sensors and to get back data measurement. It gives possibility to adjust some parameters like measures periods, communication baudrates ...

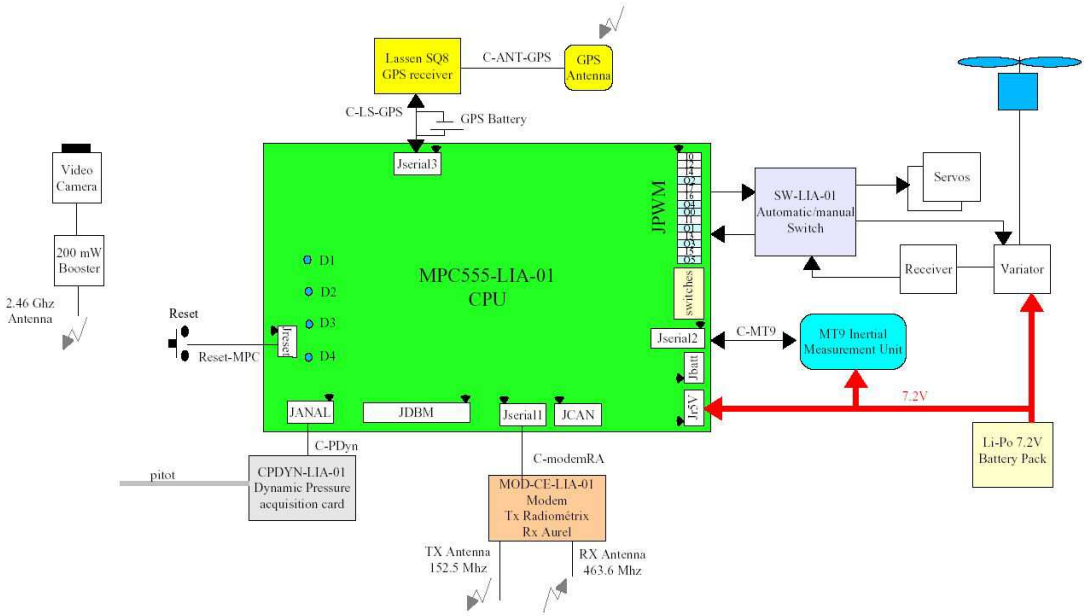

Fig. 2 Onboard system architecture for the Drenalyn UAV

The IMU is a complex device which is itself the subject of another ongoing research project at Supaero (see [2]). However its principle will be more detailed in the next part. Once fully developed, this unit will equip various unmanned vehicles of Supaéro's LIA (Laboratoire d'Informatique et Automatique), terrestrial and aerial. This is one of the aspects of the high level of community in terms of software and hardware architecture between the UAV projects at Supaero .

3.3 Communication

The UAV operates 4 RF links at different wavelengths (plus the GPS receiver) to receive its instructions from the ground station and send back telemetry and mission data :

4-channels receiver : this is a standard model-aircraft component ; the receiver frequency has been set to 41.000 MHz. It features a pulse-code demodulator which provides high noise tolerance and emitter authentification in case there are more than one remote control working at the same frequency. When the radio-controller signal is lost, it is possible to program a failsafe mode with predetermined commands to prevent the actuators from receiving aberrant commands. It is likely to be the last commands received, in case the signal loss is temporary, but the engine could also be switched off to prevent the UAV from flying away uncontrolled.

Bi-directionnal modem : the downlink at 152.5 MHz is used to send real-time mission data down to the ground station, and the optional uplink at 436.6 MHz can be used to receive mission updates (new waypoints or obstacles). This device will shortly be replaced by a network card based on wi-fi technology at 2.4 GHz.

Video transmission (optional feature) : This broadband link will also rely on wi-fi components at 2.4 GHz.

3.4 Actuators

Aircraft control is achieved by standard model-aircraft components. They consist in 2 servos (one for each elevon), and a variable voltage regulator for throttle control. The latter is an advanced component : when the input voltage (battery level) is detected critically low, the engine is turned off so as to save power for the CPU and servos, in order to keep the aircraft in control during its glide down to the ground. However in

such a situation the pilot would certainly prefer to switch back to manual control mode.

4 Software architecture

Now that the system hardware architecture has been defined, we will focus on the software architecture that can manage, coordinate and control the different subsystems and interfaces. The software is developed in C language and is designed on a real-time concept. Real-time operations are managed in the synchronous approach, that is to say calculations for different tasks are made as soon as they appear in chronological order without preemption, as their execution time is very short. It is only when 2 interruptions are raised at the same time that we can set a priority order between them. The synchronous approach has been chosen because the MPC-555 CPU offers high performance and the time constraints can be respected. The measurements need to be refreshed every 5 ms and the period of the calculation of the autopilot is 10 Hz. Moreover, the code is designed in order to maintain a high level of modularity. As for the hardware structure, if we want to change a component, or the drivers for a component, it is easily possible.

The software architecture is presented on Fig. 3. We will now explain what the structure of each subsystem is and finally how theses blocks are combined together.

4.1 Scheduler

This system is very important because it is the system that handles the others. In a real-time approach, it can be divided into two main parts. The first part is the handling of the main loop, and the second one is the handling of the interruptions raised by all the other functions.

The main loop: The goal on the main loop is to coordinate all the systems together in order to go from the measures to the commands. This main loop is supposed to call the measures every 25 milliseconds and to compute the command laws every 100 milliseconds.

We will now explain the sequence of operations for the flight software. At the beginning, the program has to proceed with the initialisation of all the subsystems. (GPS, TM/TC, Autopilot, other sensors ...). Then the infinite loop manages the events (Scheduler). First, the software receives the measures, then, the inputs of the remote control are read, the Inertial guidance platform loop is called, in order to filter the measures. Then, if it is enabled, the autopilot loop is called, and the commands are computed.

The commands are sent to the actuators. At the end, the telemetry structure is build and sent to the modem.

This loop is a very important part of the software but it doesn't handle communications because of time constraints. Indeed the data must be sent at a very precise time. That is the reason why the communication methods are handled with interruptions.

Handling interruptions: As said previously, data broadcast and reception are handled with interruptions. That allows the data to be sent at a constant rate to the ground station. The main loop is a background task and when an interruption is raised, the main loop is stopped immediately, and the interruption software is called.

The problem here is that there are several kinds of interruption possible. For instance, an interruption from the modem could be raised at the same time as an interruption from the measures. In order to handle this, the interruption handling processes the interruptions in a predetermined order.

More precisely, the interruption that the CPU can receive are those that are from the modem, and from the sensors. The choice is to process the measurements before the modem.

4.2 TM/TC drivers

Modem downlink : An onboard modem on the UAV allows to send data to the ground during the flight. This data contains attitude and position of the UAV, and other guidance information (targeted waypoint, distance to next waypoint...). Pictures and/or video from the onboard camera will be transmitted via a separated broad-

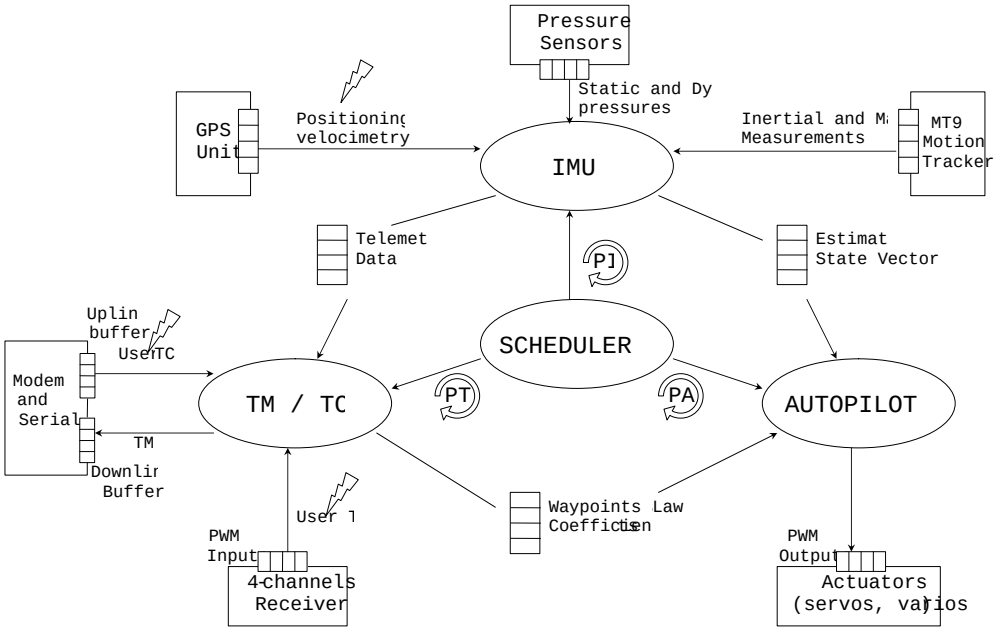

Fig. 3 Software architecture

band modem.

The organisation of the telemetry is based on interruptions. At the end of each loop of the software, a data structure is filled.

The transfer is made at a rate of 9600 bauds. That means that a structure is sent each millisecond. When it is time to send the data of the loop, the software raises an interruption and the system immediately send this structure. On the ground, the data is received in real time. An interface is developed in Delphi in order to save the data into text files, and to plot some curve about the parameters.

Modem Uplink: As the onboard modem also features an uplink, it is possible to send data to the UAV through a wireless link. The Delphi interface used for telemetry is able to upload some parameters to the UAV. The parameters are the mission waypoints, the command laws' coefficients and some coefficients about the sensors and the actuators (gains and offsets). Therefore, it is already possible to update the mission objectives during the flight even though this feature has not been used yet.

When the modem receives data transmitted from the ground, it raises an interruption and the software immediately process the data.

Remote control: The CPU is able to read PWM signals that are sent by the remote control directly out of the 4-channels receiver on the UAV. Every time a new information is received, it raises an interruption and the main software process the incoming data. This feature is useful to conduct the development of the UAV autopilot.

The remote control allows a lot of possibilities. With this system, we can :

- switch between auto/manual modes, although this commutation is done by a redundant external switch
- control the longitudinal stability while the lateral stability is controlled by the autopilot.
- control the lateral stability while the longitudinal stability is controlled by the autopilot.
- control entirely the UAV as a standard remote-controlled model aircraft, but through the CPU (transparent mode).

4.3 GPS Data

The GPS Data processing is similar to modem data processing. Every 250 milliseconds, the GPS receiver raises an interruption. The main software has to handle this in order to provide the inertial measurement unit with the measured position and velocity of the UAV in a terrestrial reference.

4.4 Inertial Measurement Unit

The Inertial guidance software may use a predefined time segment of the main calculation cycle (inertial calculation, autopilot, telemetry ...). Using a synchronous approach, the software is divided in three loops which allow sensors to provide measures for the estimator (prediction and correction). The estimator is described at the next point. Without available measures from the sensors, prediction part of the estimator calculates system state vector. A high speed loop (typically 100Hz) permits to apply a correction to state vector with pressure sensor and MT9 motion tracker measures. This information is very precise on the short term. However measurements derive with time because of acceleration and speed integrations. To fix derives, a low speed loop uses GPS data to readjust UAV position state vector.

The IMU software segment can be divided in two parts: sensors drivers and the hybrid estimator.

4.4.1 Sensors drivers

In order to respect the synchronous approach, two main state machines are used by sensors drivers. The first one retrieves frames data byte per byte from the UART (universal asynchronous receiver-transmitter) buffer. At each scheduler's activation, it checks this buffer to know if a new byte is available. Then, it builds the frames to store it in a current frame buffer. The second state machine reads the frame to process it (configuration communication or measure retrievement). With the start of the UAV, drivers go automatically into configuration mode. Afterwards, sensors send periodically their measures. If no problem is detected, drivers only get back data for the hybrid estimator described below. MT9 motion trackers include special calibration data. These data are unique for each sensor and they are implanted in a ROM into the sensor. The MT9 driver is able to retrieve a misalignment and a gain matrix to correct its measurements. Moreover for each frame, a MT9 integrated temperature sensor delivers temperature information in order to apply other corrections.

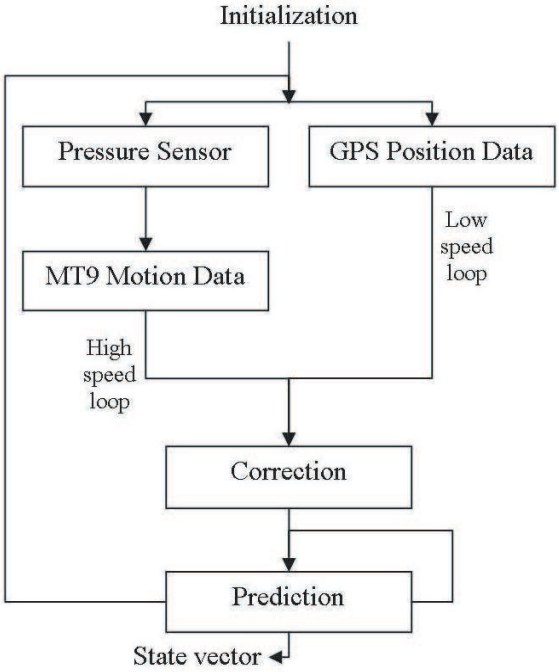

Fig. 4 Software architecture of the IMU

4.4.2 The hybrid estimator

The hybrid estimator must fusion data which are provided at different frequencies from different sensors. It has to provide states vectors with these following UAV parameters:

- $[\phi, \theta, \psi]^T$: Attitude (rolling, pitching, courses).
- $[x, y, z]^T$: Position in UAV NED (North East Down) referential.

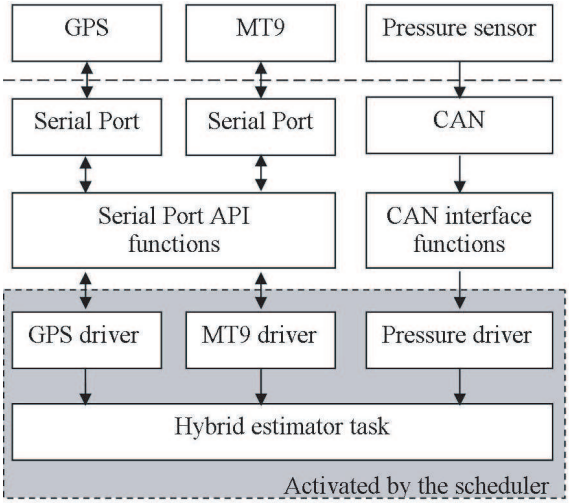

Fig. 5 Scheme of the IMU architecture

- $[u,v,w]^T$: Mobile referential UAV velocity vector.

In a stochastic noisy environment, Kalman filters are used to observe a system in order to predict or to estimate system states vector. Results are optimal with white gaussian noises. It balances information to give them more or less priority during estimation computation.

In this case, Kalman filtering is employed to estimate UAV states vectors.

To determine previously described states vectors, an estimator is used and based on two Kalman filters build in cascade.

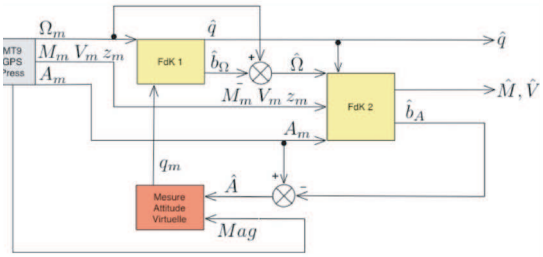

Fig. 6 Kalman filters of the IMU

This method allows reducing processors calculation charge by decoupling attitude state and velocity/position states. Each of them is estimated by a Kalman filter (FdK).

Where: $q = [q_0, q_1, q_3]^T$ is attitude expressed with a quaternion. $M = [x,y,z]^T$ is NED (North East Down) referential position. $V = [u,v,w]^T$ is velocity vector in mobile referential, and $\Omega = [p,q,r]^T$ is instantaneous rotation velocity vector in mobile referential (p, q, r for rolling, pitching and course velocity in mobile referential)

4.5 Autopilot

To design the autopilot, the linear lateral model and longitudinal model are decoupled. This assumption is common in aeronautics.

The auto-pilot gets some information from the sensors, it integrates these inputs and creates the appropriate commands to stabilize the UAV. In order to achieve the mission, the autopilot has to stabilize the height, the velocity and the course.

As we have two linear models, a longitudinal and a lateral one, the structure of the autopilot is divided into two separate problems, as described on figure (7).

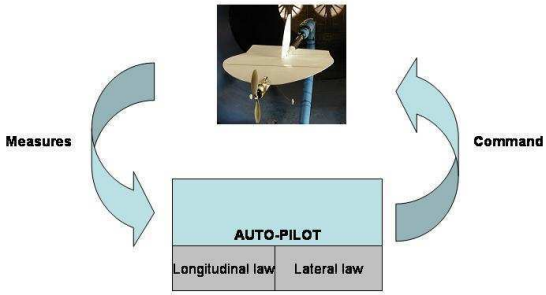

Fig. 7 Structure of the auto-pilot

Classical technique are used to design longitudinal and lateral stabilizer.
Every 100 ms, a procedure is called in order to compute the commands that will be applied for guidance and flight stability of the UAV. The mission system is generic, but the autopilot is specific to the chosen aerial platform. A generic autopilot has been created for model airplanes and we have to provide an environment that can support this peculiarity.

The command laws for an UAV family can be summarized in some constants. A special environment is developed in Delphi that is able to upload these constants into the UAV. Indeed, it is possible to save the data of an autopilot in a simple text file and upload it just before the mission. In the future, it is possible that the same drone could be given different parameters according to the flight characteristics imposed by a specific mission. Moreover, the guidance algorithm is separated from the autopilot algorithm and the trajectory can be uploaded before the mission. The type of the trajectory is a tabular of waypoints (longitude, latitude, height, time).

5 Validation

Complete validation is carried out first by checking the behavior of each subsystem separately before validation of the system as a whole through synthesis tests.

5.1 Real-time software

Now the autopilot software has been finalized, the computation time for the control loop is measured so as to confirm the hypothesis of low CPU usage for the synchronous approach, at the chosen frequency. It appears on figure 8that the required time for command computation is a of approximately 15 ms but always below 20 ms. The hypothesis that lead to the choice of the synchronous approach is easily confirmed for a working frequency of 10 Hz for the Drenalyn (100 ms foreclosure duration).

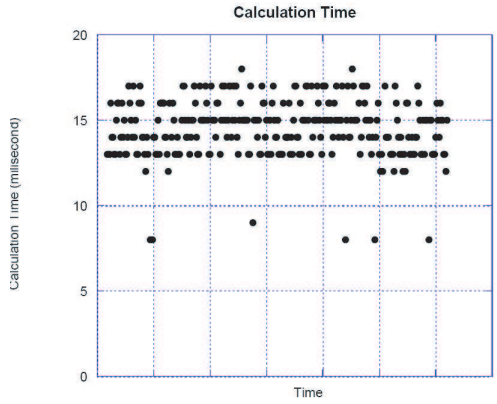

Fig. 8 Computation time for the control loop

5.2 Inertial Measurement Unit

The IMU attitude measurements are tested by simple +/- 90° roll (fig 9), pitch (fig 10) and yaw (fig 11) manoeuvres. The measurements are very precise and moreover almost noiseless.

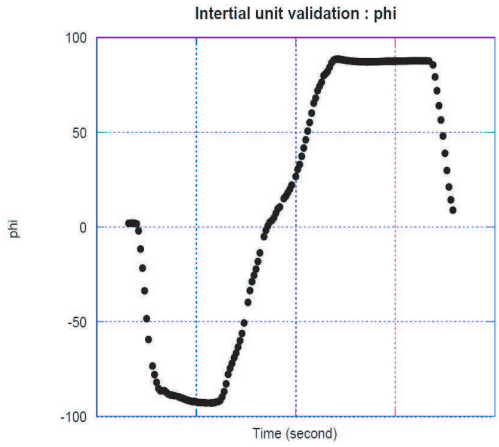

Fig. 9 Inertial Measurement Unit output values for +/- 90° roll manoeuvers

5.3 Guidance and control

The autopilot, and especially the guidance and control loops have been developed in a Matlab/Simulink environment but should be imple-

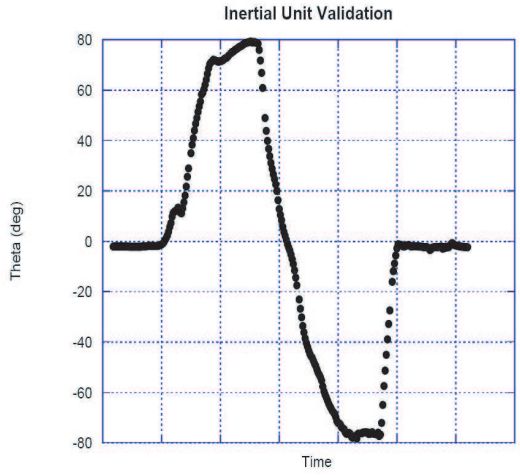

Fig. 10 Inertial Measurement Unit output values for +/- 90° pitch manoeuvers

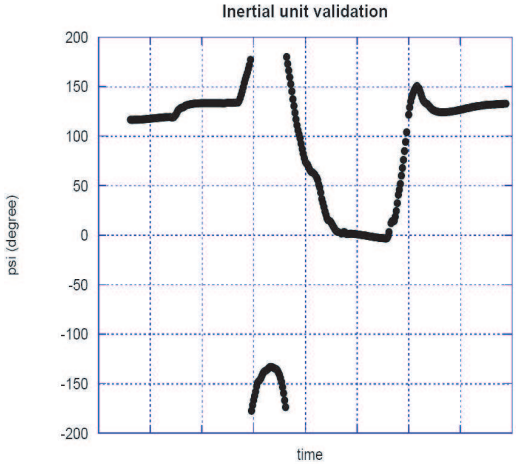

Fig. 11 Inertial Measurement Unit output values for +/- 90° yaw manoeuvers

mented in C-language on the onboard CPU. Transcription is relatively easy as M-files syntax is not that different from C and Simulink objects are nothing but matrixes or relatively simple functions. However, for troubleshooting and validation purposes the new code requires extensive testing, using the C compiler and S-functions.

The first step after having transcripted a simulink block diagram into a function coded in C-language is first to have it accepted by the C-compiler and then to validate it in the initial Matlab/Simulink development environment. The block diagram is replaced in the non-linear simulator with an S-function working with the new C-code. System outputs and response are then observed to check that the new function accomplish the same task as before. Theoretically, the autopilot response should be exactly the same in both cases. The results for the Drenalyn UAV (cf. [1]) are presented below.

Control loop: The autopilot response to an altitude and airspeed step in the non-linear simulator is presented on fig. 12, for the Simulink controller and the flight software implemented in Matlab as an S-Function. Besides the controller performance, which is more precisely described in [1], what is remarkable on these plots is the extreme closeness of fit between the Simulink model and the flight software coded in C-language.

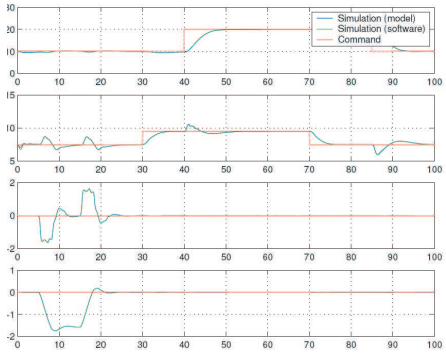

Fig. 12 Simulated autopilot response to longitudinal steps (altitude and airspeed)

Guidance loop: The guidance performance of the Simulink controller is compared with the

performance autopilot coded in C on fig. 13, for a simple trajectory around 4 waypoints. Here again both followed paths are extremely close to one another.

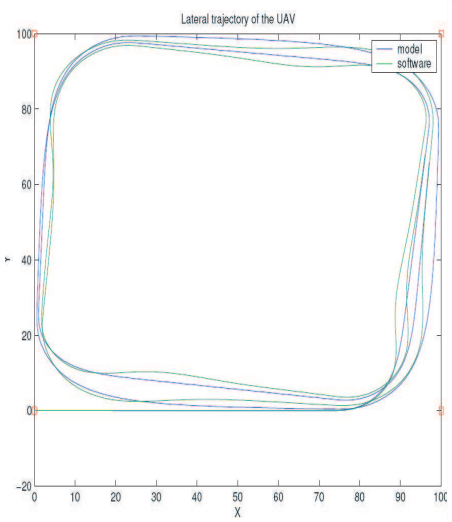

Fig. 13 Simulated autopilot response to longitudinal steps (altitude and airspeed)

Considering the previous results for the guidance and control loops, we can see that the autopilot transcription from its Matlab development environment to the flight software is extremely accurate, so now on we will consider that the autopilot defined on Matlab is exactly the same as the one implemented in the UAV flight software. This feature will be very useful for flight test data analysis.

5.4 Positioning

The GPS receiver should be able to give an estimated position of the aircraft whatever the aircraft attitude and velocity. This ability is tested with the experimental apparatus described on figure 14. The GPS sensor is placed at the end of a 3.8 meter pole which can rotate around a vertical axis. It can also tilt around an horizontal axis. Such a long pole has been chosen so as to have a movement amplitude that can be easily detected by the standard GPS system (precision around 1 m). The sensor attitude can vary around its two rotational degrees of freedom. The horizontal velocity may vary with the main rotation angular rate.

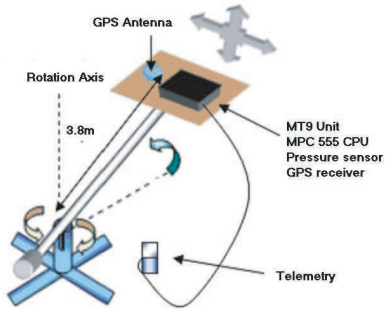

Fig. 14 Bench test for the GPS receiver

The following measurements (fig. 15) have been made for various attitudes and velocities. We see that the GPS positioning remain extremely precise whatever the attitude ; the decrease of the spatial resolution of the trajectory with horizontal velocity is due to the relatively low working frequency (4 Hz). However, considering the relatively low airspeed of our UAVs (generally under 10 m/s), this GPS sensor will provide excellent performances in positioning when used in combination with the Inertial Measurement Unit.

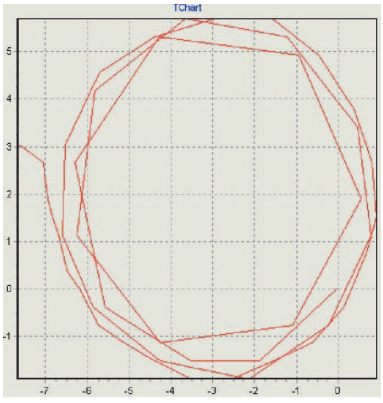

Fig. 15 GPS measured position for increasing horizontal velocity and list angle

5.5 Actuators

The checklist for the actuators is done in 3 points in laboratory :

- Manual commands: We first check in manual mode, engine off, that the proper commands are executed by the aircraft servos upon operator request, and that their directions are correct for flare and roll manoeuvres. Then we check throttle control while holding firmly the aircraft. These tests also control the proper operation of the remote control.

- Autopilot commands: We check in auto mode, engine disconnected, that the autopilot commands in response to induced perturbation (roll, heading, airspeed and height) are executed and that their directions are correct for the targeted recovering manoeuvres. Airspeed step is simulated by simply blocking the pitot with a finger. The same response are checked engine on while firmly holding the aircraft. Theses tests also control the proper operation of the auto/manual switch, and the correct level of the command outputs from the CPU.

- PWM inputs: We check that the CPU correctly reads the command outputs out of the receiver and that it can correctly interprate them.

6 Conclusion

Up to now we have developed and qualified a generic mission system designed for small aerial vehicles. It includes a powerful CPU, an accurate Inertial Measurement Unit, and a complete TM/TC subsystem as well as a ground station. Thanks to its light weight, its robustness and its high level of modularity, it can equip the wide range of UAV of various concepts developed at Supaero.

Acknowledgements

We want to thank all the student at Supaero who worked on this project previously or simultaneously, and particularly Matthieu Bouttier, Alexis Fouesneau, Lionel Rosellini, Jean-Baptiste Vergniaud, and the PEI 2005 team [2].

Fig. 16 The Drenalyn UAV

References

[1] Mathieu Bouttier, Lionel Rosellini, Jean-Baptiste Vergniaud *Conception of a generic autopilot of an UAV family*, PIR Report, 2005.

[2] S. Depraz, A. Fouesneau, V. Louvet, T. Nussmann, D. Thomas *Centrale hybride inertielle/GPS*, PEI Report, 2005.

[3] Arej-Saad, Scacchi, Siddiqui, Tillier, Zadrozynski *La Drenalyne*, PIE Report, 2004.

[4] Baisez, Bretegnie, Michel, Pollet *Automatisation des drones: Drenalyne et MaxiKiool*, PIR Report, 2004.

[5] C. Chiappa, J-M Biannic. *Trois techniques de synthèse multivarable appliquées au réglage d'un pilote automatique d'avion civil*, Editions de Supaéro, 2003.

[6] F.L. Lewis *Applied Optimal Control and Estimation* Prentice-Hall, 1992