Developing Embedded Control Systems with XtratuM
Application to Control the Attitude of a Mini-helicopter

P. García[1], P. Albertos[1], A. Crespo[1] and F. González[2]
[1]*Institute of Automática e Informática Industrial, Universitat Politecnica de Valencia, Valencia, Spain*
[2]*Universidad Central de las Villas, Santa Clara, Cuba*

Keywords: Embedded Control Systems, Development Platform, UAV Control.

Abstract: Developing embedded control systems requires to have the possibility of analyzing and evaluating new control algorithms before their implementation in the final system as well as their robust operation once implemented. For that purpose, a generic platform composed by the hypervisor *XtratuM* and RTOS *PaRTiKLe* is introduced in this paper. The interaction between the user and the process is generated by using Linux, while the real-time execution of the process is ensured by PaRTiKLe. Moreover, hypervisor XtratuM provides the facilities to execute several partitions with different guest OSs as well as the mechanisms to communicate PaRTiKLe partition and the control environment. By means of this platform the performance of future partitioned embedded systems are analyzed, evaluated and improved. Experimental tests are carried out in order to prove the effectiveness of the system. The reported results show the good performance of the designed system and the robustness of the platform.

1 INTRODUCTION

One of the latent problems when working on embedded control systems (ECS) is the lack of a platform to evaluate the control algorithm before being embedded. This pre-validation of the control algorithm would save substantial time, which can be normally tedious, to adjust the controller gains. Moreover, potential system failures can be detected before the control is put into operation and, in this way, avoid economic losses.

Recent advances in software technology, have produced a tendency to design, develop and integrate more software and hardware components for real-time embedded systems. Example can be found in aeronautics, automobile equipment, in mobile telecommunication, and so on. The capacity to support real time activities, device drivers access, fault tolerance and minimal distribution of software and hardware (Kopetz, 2011) are the main aspects to conceive an ECS.

It was in the avionics field were the partitioned systems were proposed and developed in order to increase the security and predictability of the control systems. The foundations of this approach are presented in (Rushby, 1999). Today, partitioned systems are being evaluated and promoted in several fields offering the services and mechanisms to build up safe and secure embedded systems (AUTOSAR,), (M. H. Deredempt, A. Crespo et al., 2012).

The use of partitioned systems permits to develop mixed criticality applications where some partitions may have different levels of temporal constraints or security properties (Commision, 2012). In control systems, the partitioned systems can involved one or several partitions with real-time operating systems executing control activities and other partitions with general purpose OSs than can provide the graphical and end-users interfaces and other remote services. The system is executed on top of a hipervisor that provides virtual machines to the partitions.

This approach allows a feasible and optimized software implementation with a better resource usage, and this in turn results in a cost reduction. Therefore, the main qualities of such a well-engineered software should lie in its capacity to support fast, easy-to-use, low-cost solutions, while guaranteeing, at the same time, a predefined quality of service. Moreover, evaluate the performance of the software and hardware is also necessary and very important before the control system is embedded.

Currently, there are evaluating methods for embedded systems but they are only valid for specific applications or specific programming languages. Like-

García P., Albertos P., Crespo A. and González F..
Developing Embedded Control Systems with XtratuM - Application to Control the Attitude of a Mini-helicopter.
DOI: 10.5220/0004484504340441
In *Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2013), pages 434-441
ISBN: 978-989-8565-70-9

wise, the use of these systems leads to high development costs and maintenance, besides requiring a deep knowledge of them. On the other hand, the implementation of control laws in these evaluation systems must obey certain pre-established criteria for the system. This can result in a problem, since the control algorithm would have to be adjusted or reconfigured, producing a change in its structure and in the system behavior when finally applied.

To our knowledge, there is not yet a generic platform, in process control, being able to implement different types of operating systems and/or processes. The use of an open-source hypervisor and RTOS (PaRTiKLe), could improve the development process in many ways, such as a low price in the development and maintenance and good robustness when other applications are running. Specifically, the model of development conceived in the open source community could be successfully adapted to the design and implementation of real-time critical application, improving the reliability and the extensibility of the software components.

In this paper we introduce a new partitioned platform composed by *XtratuM* and *PaRTiKLe* systems, allowing a robust design of embedded control systems. A Linux partition provides the advantage of the interaction between the user and the process, while PaRTiKLe partition ensures the real-time execution of the control tasks even when the system is running other tasks. XtratuM allows for strong safety of the application, always required in ECS. The major applications of the platform are to analyze, evaluate and improve the performance of future embedded systems. Additionally, the proposed platform can be used in critical embedded systems, i.e. where the dynamics of the system is very fast and the processing time must be executed under strict constraints. A more detailed description of the proposed platform is presented in section 2.

The validation of the platform and its application to the design of embedded control systems for fast dynamic plants (UAV) are introduced in section 3. A comparison of the proposed platform with respect to a platform under only Linux RT environment is also provided. Finally, some comments and conclusions are included in the last section.

2 SYSTEM DESCRIPTION

Embedded control systems are increasingly being used in safety critical applications and infrastructures such as, aircraft flight control and electric grids (Crespo and Alonso, 2006) (Parkinson and Kinnan, 2003) (J. Loyall and Fernandez.,). It is well known that, the integrity of these critical systems will depend on the good performance of the installed software components. And, a bad performance or losing data in the framework, may cause massive material and even human life losses.

In many critical systems, and the proposed platform is not an exception, it is required that numerous software functions with varying levels of exigency (for example, hard, soft and non real time tasks) are running at the same time. This kind of integration creates a special challenge because it is required to guarantee the execution of every task and the non interference with each other. In (Masmano et al., 2009), the authors present a software architecture based on a virtualization layer offering virtual machines to execute partitions. This approach offers the capability to define several partitions guaranteing their temporal and spatial isolation and achieving the deterministic behavior required by safety-critical systems.

To ensure the previous requirements, the designed platform uses an open source based hypervisor (XtratuM) which executes real-time constrained partitions based on PaRTiKLe RTOS. Moreover, this fact will facilitate the reuse of the code and will enable the development of more complex software without sacrificing real-time performance. In the following, the main characteristics of XtratuM and ParTiKle systems are described.

2.1 XtratuM

XtratuM is an hypervisor for embedded devices, it provides a framework to run multiple concurrent operating systems in a robust partitioned environment (Masmano et al., 2005). XtratuM can be also defined as an open source nanokernel (www.opensource.org/docs/definition.php,), just implementing two simple device drivers: interrupt and timer. Likewise, XtratuM can be considered as a small subset of the lowest operating system layers that can meet the hard real-time system requirements: fast, compact, portable, simple and predictable.

The main characteristics of the XtratuM system are (see Figure 1):

- Bare metal hypervisor designed for real-time embedded systems.
- Several partitions can be simultaneously executed
- Every partition contains the most appropriated OS according to the application needs.
- Every partition is executed in separated memory areas or regions. It prevents for errors or attacks

outside of the partition creating a new and higher level of security.

- Every partition may have different level of security.

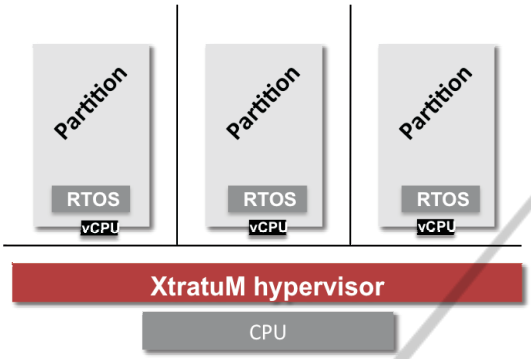

Figure 1: XtratuM structure.

For critical control processes, these features are very important, because the execution of two different systems at the same time, should give quick access to some devices or files and an easy interaction between the user and the process, and at the same time guarantee the implementation and execution of the tasks in a RTOS domain. Additionally, the last two feature are essential to fulfill security requirements in security-critical real-time systems. For example, working in a system composed by Linux and RTOS domains, if a computer attack happens to the Linux domain, it should not affect the control law execution embedded in the RTOS domain.

An example of this can be an engine control program with a graphic interface. The application can be split into two parts: the control algorithm which interacts with the process to be controlled, with hard real-time and fault-tolerance requirements (where the use of a hard RTOS is compulsory) and the graphic interface, without real-time requirements, displaying interesting data or accepting some input commands.

Figure 2 introduces GNU/Linux and PaRTiKle environments implemented in XtratuM (see details in section 3.2). The main characteristic of this configuration is the effectiveness to separate the priority tasks being executed by ParTiKle from the other ones performed by GNU/Linux. This fact yields the system's behavior stable even if an error appears in the general system. The PaRTiKle partition forward data to the Linux one, to be independently treated, and receives from that one updating of the control algorithms or tuning parameters.

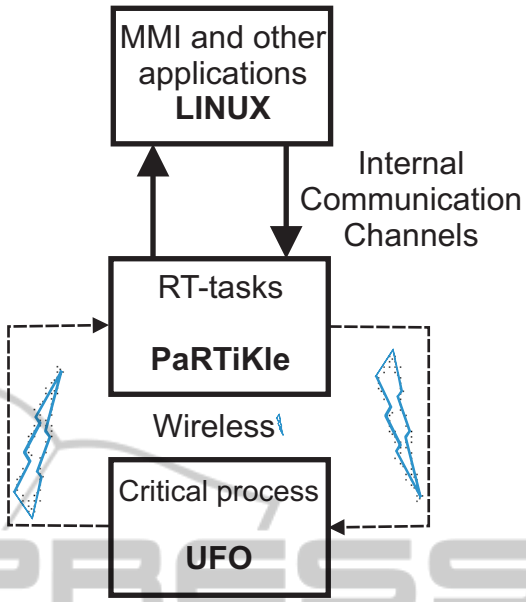

Figure 2: Critical process and software platform.

2.2 PaRTiKle Partition

PaRTiKle is an embedded real-time operating system, distributed under the terms of the GNU Public License (Peiro et al., 2007). It provides features such as full preemptability, minimal interrupt latencies, synchronization primitives, scheduling policies, and interrupt handling mechanisms, needed for critical applications with hard real time constrains.

Additionally, PaRTiKle brings support, in a bare machine, for multiple execution environments such as for example, a Linux regular process and a hypervisor domain, see Figure 3. Although, PaRTiKle was designed to be POSIX compatible, it can also provide support for *C++*, *Ada* and *Java* languages, and it furnishes a standard *C* library in the application context.

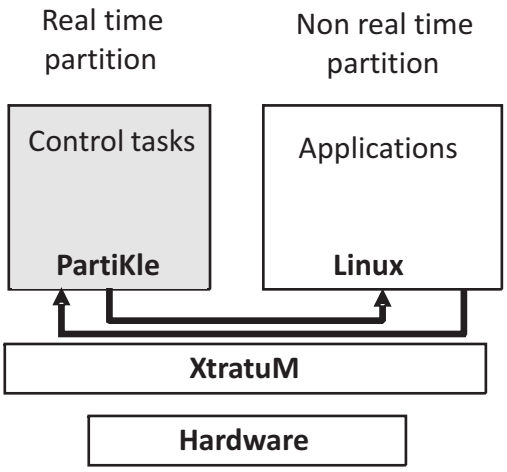

Figure 3: PaRTiKle as XtratuM domain.

These characteristics help to design different levels of abstraction in the process and control domains. This is very useful to reuse the code and to make shorter the development cycle.

Likewise, the PaRTiKle's kernel provides additional characteristics like thread scheduling, synchronization, timer and communication primitives. On the other hand, it also handles hardware resources, such as, interrupts, exceptions, memory, timers, etc.

A brief summary of the main characteristics of PaRTiKle system as it is implemented nowadays is as follows:

Scheduling: The scheduler only implements a Rate-Monotonic (RM) scheduling policy.

Timing Management: It provides multiple virtual timers to the kernel and the application.

Physical Memory Management: It is able to manage the free available physical memory.

Kernel Library: There is only a minimal C library used by the kernel. This library cannot be accessed by the application.

System Call: All kernel services are provided via a single entry point.

2.3 The Designed Platform

The physic platform is composed by a processor (up or equal to pentium II), the software described previously, and the input/ouput cards. The platform can be thus connected to the process to control or evaluate. The items added to the platform will depend to the process to study.

The designed platform presents some advantages with respect to others in the literature, such as

- Some tests, developed by using several Linux tools, can be done using PaRTiKle, like a Linux process.
- The platform can be executed using PaRTiKle like being a XtratuM domain and, at the same time, take advance of the Linux domain (persisting data, monitoring some value, etc).
- The final program can be embedded using PaRTiKle as a bare machine.
- The platform is portable (different hardware under the same software can be implemented), configurable and maintainable (Peiro et al., 2007).

In addition, all the main components in the platform, GNU/Linux, PaRTikle and XtratuM, are based on the well known open standard languages (POSIX, ARINC)[1]. Likewise, future works in the proposed platform will use some important advantages of these languages, for example, the temporal nature of Ada or the facilities to create strong interfaces in C++ and Java.

To prove the robustness of the proposed platform and to verify the behavior of some important parameters, a real-time analysis of the experiment can be done using the jitter performance evaluation tool (as reported in (Berna et al., 2011)). In fact, it is well known that certain values of the jitter can degrade the control performance and in extreme cases even cause instability of the closed-loop system, see (Stothert and MacLeod, 1998) (Shin and Cui, 1995).

In the experiments here performed, heavy computation load tasks have been concurrently run with the RT control tasks. If the load is too high, there are not enough resources. The control tasks performance is shown not being affected by the system degradation.

3 PLATFORM VALIDATION

To validated the proposed platform two kinds of experiments are presented in this section. First, a Quanser Helicopter with four rotors is used ((*Quanser*, 2007)). The performance of the platform is tested in order to stabilize the roll angle in a desired position. This is a 'subcritic' process because the helicopter is placed on a base platform in such a way that the attitude is the only possible variable to be stabilized. This base platform avoids the helicopter crashing in case that the controller is not able to stabilize it. The second experiment is also devoted to stabilize the attitude of the helicopter but flying in 3D. In this helicopter any wrong measurement from the sensors and/or malfunction of the microcontroller will result in a helicopter crash.

The two processes are tested using the same platform and they can be selected by using a switch, see Figure 4.

Both processes are helicopters and the goal is to stabilize the attitude dynamics for each one. Therefore, the classical attitude mathematical model is given by (Castillo et al., 2005)

$$\ddot{\eta} = \tau_\eta \quad \forall\, \eta = \psi, \theta, \phi \tag{1}$$

where η represents the Euler angles and τ_η the control input.

[1]Under the POSIX standard design guides there are a lot of good products, several Unix versions are among the most successful ones. The ARINC specification was designed to be used by high-reliable products and is used in avionics applications.

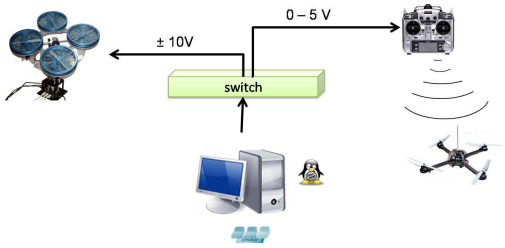

Figure 4: Platform & process schema.

3.1 Application to the Quanser Helicopter

In this section, the proposed platform is tested stabilizing the roll angle (ϕ) of a mini helicopter with four rotors mounted in a vertical base (see more characteristic of the helicopter in (*Quanser*, 2007)). The helicopter is a modified Quanser helicopter connected with the processor using the input/output cards. Moreover, the helicopter can be only controlled in attitude. An IMU (MicroStrain, 2007)) is used to measure the orientation (ψ, θ, ϕ) and the angular rate $(\dot{\psi}, \dot{\theta}, \dot{\phi})$ of the vehicle. The IMU is connected via an USB port. In addition, the system receives commands from a small keyboard and will send, periodically, the system status to a host computer in order to display and analyze the system variables and status.

The following controller, based on saturation functions, is used to stabilize the roll angle of the helicopter

$$\tau_\phi = -\sigma_{\phi_1}\left(k_d\dot{\phi}\right) - \sigma_{\phi_2}\left(k_p(\phi - \phi_d)\right) \qquad (2)$$

The stability analysis of the closed-loop system is given in (Sanahuja et al., 2010).

The behavior of the platform described in section 2 is compared with respect to a platform only based on the *Linux RT* system. The control objective is that the helicopter should follow a step trajectory. Once this task is completed, more tasks are added to the processor in order to saturate its capacity and verify the system's robustness. Figure 5 shows the ϕ response when applying the previous control law in Linux RT system. Solid line represents the almost perfect system response without others tasks executing, while the dotted line represents the same response when others tasks need to be executed (stress conditions). Note from this figure that, when executing the control algorithm at the same time with others tasks, and increasing the load, the system becomes unstable for some given load. The control inputs generated in these experiments are shown in Figure 6.

The ϕ behavior when the proposed platform (XtratuM + PartiKle system) is used by applying the control law (2) to stabilize the roll angle, is shown in Figure 7. This experiment is also done concurrently with (or without) others tasks. The solid line represents the system response without others tasks while the dotted line represents the same response with others tasks executing at the same time. Note from this figure that the system remains stable in both cases and, therefore, these experiments illustrate the robustness of the platform.

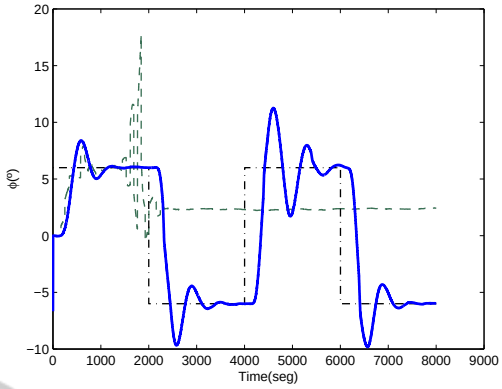

Figure 5: ϕ Angle response.

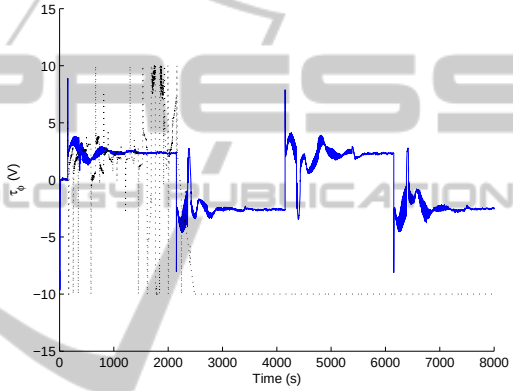

Figure 6: τ_ϕ Control input.

3.2 Application to the XUFO

The XUFO platform is more complete and it is composed by a processor, an IMU, a radio, a helicopter and a joystick, see Figure 8. The employed helicopter is a mini-helicopter with four rotors from X-UFO company (*Xufo*, 2002)). It is a well known helicopter used in some control aeronautical applications. The main characteristics are that the front and the rear motors rotate clockwise while the other two rotate counter clockwise, the gyroscopic phenomena are relatively small. This helicopter does not have a swatch plate. In fact it does not need any servomechanism. The main thrust is the sum of the

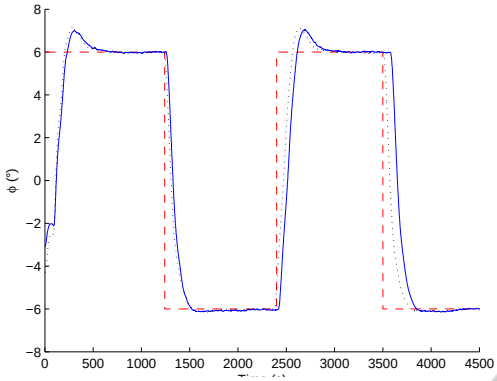

Figure 7: ϕ Angle response.

Figure 8: Control platform schema.

thrusts of each motor. Pitch movement is obtained by increasing/reducing the speed of the rear motor while reducing/increasing the speed of the front motor. The roll movement is obtained similarly using the lateral motors. The yaw movement is obtained by increasing/decreasing the speed of the front and rear motors while decreasing/increasing the speed of the lateral motors. This should be done while keeping the total thrust constant. The helicopter evolves freely in a 3D space without any flying stand.

The radio is a *Futaba Skysport* 6. The radio and the PC are connected using data acquisition cards. The connection in the radio is directly made to the joystick potentiometers for the trust, yaw, pitch and roll controls. To measure the orientation and the angular rate of the vehicle, a wireless IMU (Microstrain, (MicroStrain, 2007)) is used. The IMU sends this information via wireless to a receiver located in the ground station (processor). Moreover, the computed control inputs are sent to the rotors via a digital/analogic converter. Additionally, the system can receive commands from the keyboard and send, periodically, the system status to a host computer in order to display and analyze the system variables.

Experimental Results

Several experiments have been done. To illustrate the results, the following control is applied in order to stabilize the yaw angle under aggressive disturbances. The following nonlinear control law based on saturation functions has been used to control the system (1),

$$\tau_\eta = -\sigma_{\eta_a}(k_{\eta_1}\dot{\eta}) - \sigma_{\eta_b}(k_{\eta_2}(\eta - \eta_d)) \quad \forall\, \eta = \psi, \theta, \phi \tag{3}$$

where $|\sigma_{\eta_i}(\circ)| \leq \eta_i$, $\forall\ i = a, b$, is a saturation function, $a, b > 0$ are constant and $k_{\eta_j} > 0$, $\forall\ j = 1, 2$, are constant. Introducing (3) into (1), we obtain

$$\ddot{\eta} = -\sigma_{\eta_a}(k_{\eta_1}\dot{\eta}) - \sigma_{\eta_b}(k_{\eta_2}(\eta - \eta_d)) \tag{4}$$

The stability analysis of the system (4) was also presented in (Sanahuja et al., 2009). The main characteristic of this control strategy is that when the system is working in the linear part of the saturation function it can be seen as a linear controller.

Yaw Stabilization. The idea here, is to apply the control law in open-loop and closed-loop in order to analyze the performance of the platform. Additionally, some manual aggressive perturbations and a desired trajectory are added on-line to validate the robustness of the system.

The performance of the controller used to stabilize the yaw angle of a helicopter is shown in Figure 9. Some forces have been manually (or using the joystick) added to perturb the system. Note that, the closed-loop system remains stable in the presence of aggressive perturbations. To improve the performance of the experiments the controller parameters have been tuned on line. On the other hand, a stress condition is given to the processor, executing a video at the same time, in order to perturb the process. In pure Linux environments the use or execution of other applications could “stop” the process, and notice that it is not the case for the developed platform. Figure 10 shows the control input applied to the system.

4 PARTITIONED APPROACH ANALYSIS

When comparing a partitioned approach with respect to a more classical approach, the following advantages based on the temporal and spatial isolation to partitions provided by the hypervisor can be mentioned:

- Separation of concerns: Each partition can use the most approriated operating system. In this case,

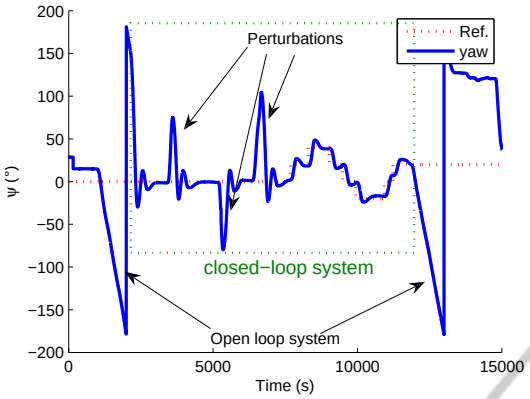

Figure 9: ψ Angle response.

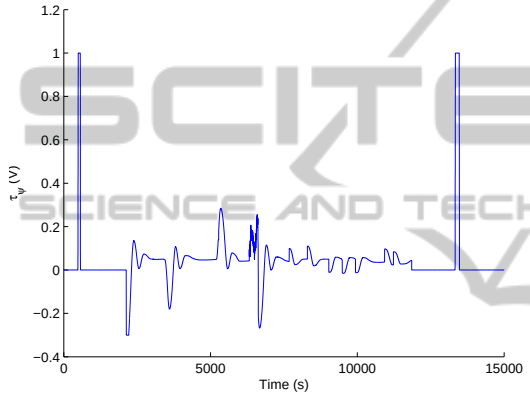

Figure 10: τ_ψ Control input.

real-time tasks require a small real-time system. On the other hand, the set of activities related with graphical interface, network communications, remote access, etc., may require a full operating system as Linux. It allows the integration of real-time with non real-time activities in the same platform.

- Several levels of criticality: each partition may have different real-time constraints and, consequently, different level of validation, security, etc.
- Independent development: partitions can be independently developed and validated. Very well defined mechanisms to communicate partitions are provided by the hypervisor.
- Faults are limited to the faulting partition: partitioning approach allows to have independent partitions that are not affected by the faults in other partitions. The hypervisor grants that a fault in a partition is not propagated to other partitions. In this case, the real-time partition defines a limited set of tasks that can be fully validated jointly with the real-time operating system using the appropriated techniques to achieve a secure partition. On the Linux partition, it is more complex (and costly) to fully validate the interfaces, communications, etc. If a fault occurs in the Linux partition, the hypervisor confines the fault to that partition without impacting the execution of other partitions.
- External attacks: if the Linux partitions permit the external access to the application services, some external attacks may happen. Even if a hacker gets the Linux partition control, the spatial partition isolation grants that no access to other partitions is allowed.

However, some drawbacks can be also considerated:

- More complex software architecture: the software architecture is more complex due to the existence of a hypervisor and several guest OS for the partitions.
- Scheduling: a cyclic scheduling has to be designed to fulfil the temporal requirements of the real-time partitions.
- System deployment: the final system to be deployed in the embedded hardware platform has to include the hypervisor and the partitions.

5 CONCLUSIONS

A new test-bed platform has been presented in order to validate and improve the design of embedded control systems. To run different tasks in an independent way, providing maximum operation safety, a hipervisor (XtratuM) has been used at the lower level.

Linux RT and Partikle like O.S. have been used in order to implement the control strategies and to guarantee priority tasks.

The proposed platform has been used for validating the ECS designed for a quadrotor and a lab helicopter. It has been proved to be robust under different load conditions.

The platform is versatile and allows the use of different OSs as well as control algorithms, with the option to experiment with changes in the control laws, control parameters and operating conditions.

Once the control structure is validated, it can be deployed and implemented in the final system.

REFERENCES

AUTOSAR. Internet *Automotive open System Arquitecture*.

Berna, A., Castillo, P., Sanahuja, G., González, F., Garcıa, P., and Albertos, P. (2011). Development of a test-bed

to implement and validate real-time control strategies for aerial vehicles. In *Proceedings of the 18th IFAC World Congress*.

Castillo, P., Lozano, R., and Dzul, A. E. (2005). *Modelling and control of mini-flying machines*. Springer.

Commision, E. (2012). Workshop on mixed criticality systems. http://cordis.europa.eu/fp7/ict/computing/home en.html.

Crespo, A. and Alonso, A. (2006). Una panorámica de los sistemas de tiempo real. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)*, 3(2):7–18.

J. Loyall, R. Schantz, D. C. and Fernandez., S. A distributed real-time embedded application for surveillance, detection, and tracking of time critical targets. *11th IEEE Real-Time Embedded Technology and Applications Symposium,11th IEEE Real-Time Embedded Technology and Applications Symposium, 2005*.

Kopetz, H. (2011). *Real-time systems: design principles for distributed embedded applications*, volume 25. Springer.

M. H. Deredempt, A. Crespo et al. (2012). Integrated modular avionics for spacecraft software architecture and requirements. In *DASIA 2012. DAta Systems In Aerospace.*

Masmano, M., Ripoll, I., and Crespo, A. (2005). An overview of the xtratum nanokernel. In *Proceedings of the Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT)*.

Masmano, M., Ripoll, I., Crespo, A., Metge, J., and Arberet, P. (2009). Xtratum: An open source hypervisor for TSP embedded systems in aerospace. In *DASIA 2009. DAta Systems In Aerospace.*

MicroStrain (2007). *3DM-GX2TM Data Communication Protocol*, technical report version 1.13 edition.

Parkinson, P. and Kinnan, L. (2003). Safety-critical software development for integrated modular avionics. *Embedded System Engineering*, 11(7):40–41.

Peiro, S., Masmano, M., Ripoll, I., and Crespo, A. (2007). Partikle os, a replacement for the core of rtlinux-gpl. In *Proceedings of the 9th Real-Time Linux Workshop, Linz, Austria*, page 6.

Rushby, J. (1999). Partitioning in avionics architectures: Requirements, mechanisms, and assurance.

Sanahuja, G., Castillo, P., and Sanchez, A. (2009). Stabilization of *n* integrators in cascade with bounded input with experimental application to a VTOL laboratory system. *International Journal of Robust and Nonlinear Control*. DOI: 10.1002/rnc.1494.

Sanahuja, G., Castillo, P., and Sanchez, A. (2010). Stabilization of n integrators in cascade with bounded input with experimental application to a vtol laboratory system. *International Journal of Robust and Nonlinear Control*, 20(10):1129–1139.

Shin, K. G. and Cui, X. (1995). Computing time delay and its effects on real-time control systems. *Control Systems Technology, IEEE Transactions on*, 3(2):218–224.

Stothert, A. and MacLeod, I. (1998). Effect of timing jitter on distributed computer control system performance. In *Proceedings of 15th IFAC Workshop DCCS*, pages 25–30.

Quanser, I. (2007). “http://www.quanser.com/”.

Xufo, I. (2002). “http://www.xufo-shop.de”.

www.opensource.org/docs/definition.php. Open source initiative, open source definition. published online in open source initiative’s official web site. *December 20095*.