A HARDWARE/SOFTWARE ARCHITECTURE FOR UAV PAYLOAD AND MISSION CONTROL

Enric Pastor, Juan Lopez and Pablo Royo, Department of Computer Architecture, Technical University of Catalonia, Castelldefels (Barcelona), Spain

Abstract

This paper presents an embedded hardware/software architecture specially designed to be applied on mini/micro Unmanned Aerial Vehicles (UAV). An UAV is low-cost non-piloted airplane designed to operate in D-cube (Dangerous-Dirty-Dull) situations [8]. Many types of UAVs exist today; however with the advent of UAV's civil applications, the class of mini/micro UAVs is emerging as a valid option in a commercial scenario. This type of UAV shares limitations with most computer embedded systems: limited space, limited power resources, increasing computation requirements, complexity of the applications, time to market requirements, etc. UAVs are automatically piloted by an embedded system named "*Flight Control System*". Many of those systems are commercially available today, however no commercial system exists nowadays that provides support to the actual mission that the UAV should perform.

This paper introduces a hardware/software architecture specially designed to operate as a flexible payload and mission controller in a mini/micro UAV. Given that the missions UAVs can carry on justify their existence, we believe that specific payload and mission controller s for UAV should be developed. Our architectonic proposal for them orbits around four key elements: a LAN based distributed and scalable hardware architecture, a service/subscription based software architecture and an abstraction communication layer.

1. Introduction

An Unmanned Aerial Vehicle (UAV) is an expression that identifies an aircraft that can fly without pilot; that is, an airframe and a computer system which combines sensors, GPS, servos and CPUs. All these elements combined have to pilot the plane without any human intervention. Another usual definition is that of an aircraft which is capable to fly in an autonomous way and operates in a wide range of missions and emergencies that can be controlled from a ground base station. The UAV's size, type and configuration could be different and depends on the actual application.

There is no doubt today that a huge market is currently emerging from the potential applications and services that will be offered by unmanned aircrafts. More precisely, UAVs can be applied in so called "*D-cube*" missions [8], *i.e.* missions identified as Dangerous, Dirty or Dull. If we pay attention to civil applications a wide range of scenarios appears. For instance; remote environmental research, pollution assessment and monitoring, fire-fighting management, security e.g. border monitoring, agricultural and fishery applications, oceanography, communication relays for wide-band applications. In general we can divide all of these applications into four large groups: environmental applications, emergency-security applications, communication applications and monitoring applications.

Nowadays and after many years of development, UAVs are reaching the critical point in which they could be applied in a civil/commercial scenario. However, we believe that there is a lack of hardware and software support to effectively develop such potentialities. Basically an UAV is automatically piloted by an embedded computer called the *Flight Control System* (FCS)[6]. This is a system that reads information from a wide variety of sensors (accelerometers, gyros, GPS, pressure sensors) and drives the UAV mission along a predetermined flight-plan.

1-4244-0378-2/06/$20.00 ©2006 IEEE.

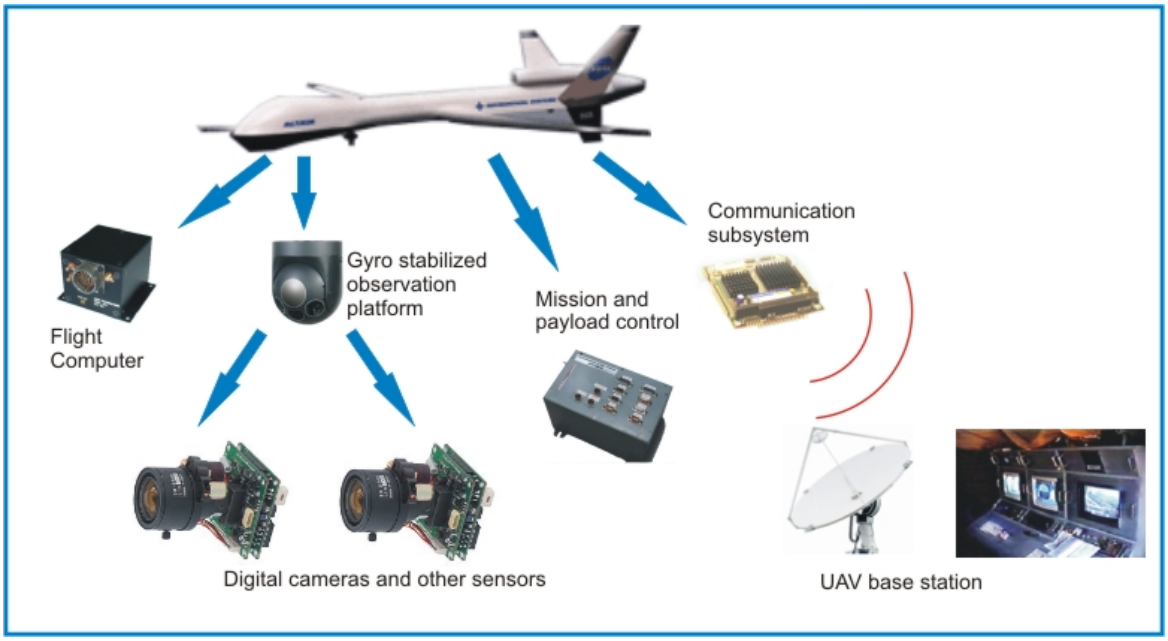

Figure 1. Main components of a UAV system.

Even though reliable autopilots exist, the main purpose of the UAV is the actual mission that should execute with its required payload (sensors, etc.) The thesis of this research is that mission and payload control are the main bottlenecks that may prevent the actual development of UAV in the civil sector. Military UAV use specific control designs specially tailored to the particular surveillance mission that they will implement. However a civil UAV should be able to implement a large variety of missions with little reconfiguration time and overhead if it must be economically viable.

This paper presents a novel hardware/software architecture [1] specially designed to operate as mission and payload controller in a mini/micro UAV. We will call this system the Mission Control Computer . Our architectonic proposal orbits around four main innovative elements: (1) a LAN based distributed and therefore easily scalable hardware architecture, (2) a service/subscription based software architecture, (3) an abstraction communication layer, and (4) a workflow-based mission planning.

The paper is organized as follows. Section 2 will detail the structure and components inside a UAV system and motivate the reason why a mission/payload is necessary. Section 3 introduces the proposed hardware architecture, Section 4 the selected software application architecture and Section 5 the architecture of the communication mechanisms inside the UAV, and between UAV and base station. As an example, our Mission Control Computer prototype is detailed in Section 6. Section 7 concludes this paper and identifies some of our future developments.

2. Background and Motivation

An UAV is a complex system composed of six main sub modules that work coordinately to obtain a highly valuable observation platform [5]. Figure 1 depicts a schematic view of each sub module.

The UAV airframe. A simple, lightweight, aerodynamically efficient and stable platform with limited space for avionics, and obviously no space for a pilot.

The flight computer. The heart of the UAV. A computer system designed to collect aerodynamic information through a set of sensors (accelerometers, gyros, magnetometers,pressure sensors, GPS, etc.), in order to automatically direct the flight of an airplane along its flight-plan via several control surfaces present in the airframe.

The payload. A set of sensors composed of TV cameras, infrared sensors, thermal sensors, etc. to gather information that can be partially processed

on-board or transmitted to a base station for further analysis.

The mission/payload controller. A computer system onboard the UAV that has to control the operation of the sensors included in the payload. This operation should be performed according to the development of the flight-plan as well as the actual mission assigned to the UAV.

The base station. A computer system on the ground designed to monitor the mission development and eventually operate the UAV and its payload.

The communication infrastructure. A mixture of communication mechanisms (radio modems, satcomm, microwave links, etc.) that should guarantee the continuous link between the UAV and the base station.

Current UAV technology offers feasible technical solutions for airframes, flight control, communications, and base stations. However, if civil/commercial applications should be tackled two elements limit the flexibility of the system: human intervention and mission flexibility.

Too much human control from the ground station is still required. Flight control computers do not provide additional support beyond basic flight plan definition and operation. Additionally, payload is most times remotely operated with very little automation support.

Economical efficiency requires the same UAV to be able to operate in different application domains. This necessity translates into stronger requirements onto the mission/payload management subsystems, with increased levels of flexibility and automation.

3. System Architecture

The hardware architecture of the proposed Mission Control Computer is built as a set of embedded microprocessors connected by a local area network (LAN), *i.e.* it is a purely distributed and therefore scalable architecture (see Figure 2). Even though this is a simple scheme it offers a number of benefits that motivates its selection in our application domain.

The high level of modularity of a LAN architecture offers extreme flexibility to select the actual type of processor to be used in each sub module. Different processors can be used according to functional requirements, and they can be scaled according to computational needs of the application. System modules can be waken up on-line when required at specific points of the mission development. Modules can be added (even hot plugged) if new requirements appear.

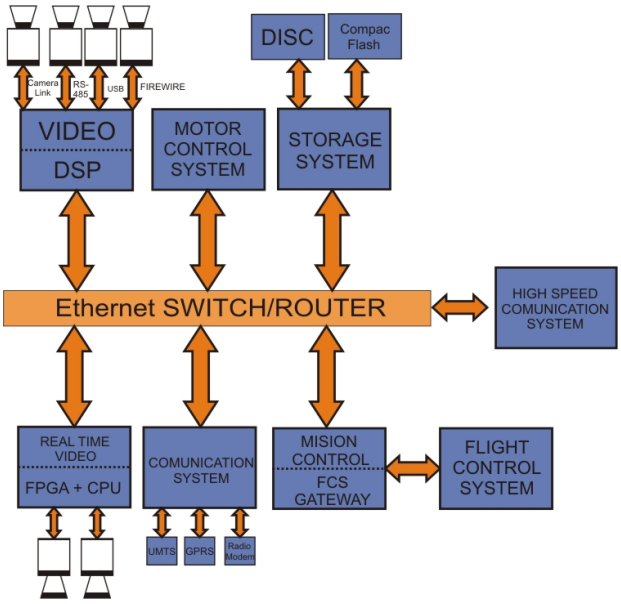

Figure 2. General view of the architecture of a Mission Control Computer.

Module interconnection is an additional extra benefit because the complex interconnection schemes needed by parallel buses do not fit properly with the space and weight limitations in a mini/micro UAV.

Finally, development simplicity is the main advantage of this architecture. By using techniques inspired by Internet communication protocols, computational requirements can be organized as services that are offered to all possible clients connected to the network. These communication schemes will be further described in Section 4.

A number of specific computational modules have been identified as "*a must*" in any real life application of UAVs. These modules are depicted in Figure 2. On top of these modules several

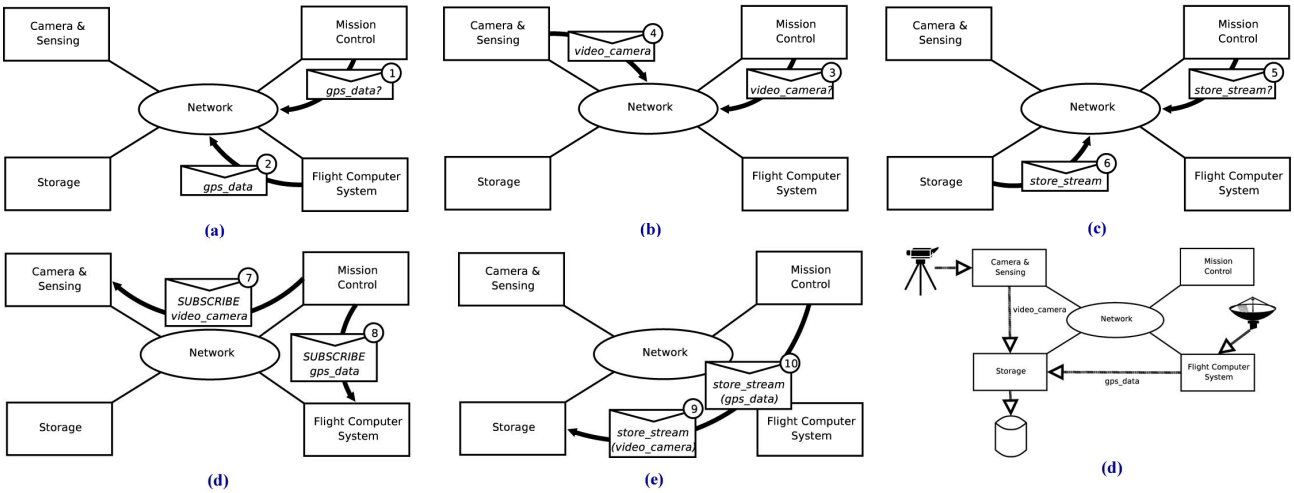

Figure 3. Operational Scenario: Mission Control starts a geo-referenced video recording.

applications will be executed providing specific services to other applications.

Even providing the computational support to these applications will be scaled according to requirements, but without requiring major modifications in the communication schemes. Critical services to be offered include the following elements: an interface with the Flight Computer System, the Mission Control service, a communication service with a selection of communication infrastructures, video and photo management as a data gathering system or even with real time processing, a data storage service, and a motor control service in case mobile components should be controlled.

4. Service-Based Software Architecture

Over this LAN infrastructure we implement a software layer that allows each computation module to support multiple applications by providing services to the network. Each application could create and subscribe to available services. The services could be discovered and consumed in a dynamic way like web services in the Internet domain. Applications could interchange information transparently from network topology, application implementation and actual data payload. This approach together with the hot-pluggable LAN offers interoperability of different modules and applications.

Service oriented architectures (SOA) are getting common in several domains, for example *Web Services* [1] in the Internet world and *UPnP* [2] in the home automation area. SOA is an architectural style whose goal is to achieve loose coupling among interacting components or services. A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by software agents on behalf of their owners. The results of a service are usually the change of state for the consumer but can also be a change of state for the provider or for both.

The idea of these architectures is to increment the interoperability, flexibility and extensibility of the designed system and their individual components. In the implementation of any system, we want to be able to reuse components of existing system, however doing that we usually introduce additional dependencies between the components. Service oriented architectures tries to minimize these dependencies by using loose coupled components.

SOA achieves loose coupling among interacting components by employing two architectural constraints. First, a small set of simple and ubiquitous interfaces to all participant components with only generic semantics encoded in

them. Second, each interface can send on request descriptive messages explaining its functioning and its capabilities. These messages define the structure and semantics of the services provided. These constraints depart significantly from that of object oriented programming, which strongly suggests that you should bind data and its processing together.

When some component needs a functionality not provided by itself, it asks the system for the required service. If other component of the system has this capability, their location will be provided and finally the client component can consume the service using the common interface in the provider component. The interface of a SOA component must be simple and clear enough to be easily implemented in different platforms both hardware and software.

4.1. Proposed architecture

In our SOA-based system, services will be provided by different modules, each one composed by an embedded microprocessor and the needed hardware to accomplish their assigned task. The designed architecture should fulfill several functional and non-functional requirements.

Dynamic discover. New modules can be attached to the LAN while the system is working and the modules already present on the system will be informed of its presence [4]. In a similar way when a module needs a service it can ask the system if any other module is offering this functionality.

Remote execution. Modules will be able to consume services exposed by other modules in the net. A service will offer several functions that can be invoked remotely. The consumer module will send the function and its parameters and will wait for the results returned by the provider module.

Module self-description. Each module will provide on request a description of the services it offers. This will help on the development of complex functionalities and the possibility that one module can use services that didn't exist when it was built/programmed.

Get/set data from modules. Modules will be able to expose their inner state using variables. Other modules can ask for this information or send changes to it in a synchronous way. These variables contain information from sensors or inputs to the different actuators in the UAV.

Data streaming. Some variables will change at high rates. It will not be efficient for a module to ask constantly for new values of such a variable. For handling this sort of data, our architecture will provide data streams that will efficiently send that information to multiple modules using the multicast capabilities of the network layer.

Two naming policies. The users of our system will want to use clear and sound names for the variables, streams and services like "*adf compass*", "*fuel flow*" or "*lights on*". However at the network layer it will be more efficient to use numeric identifiers occupying fewer bytes. One of the responsibilities of the software layer we provide is to discover and cache the numeric identifiers used internally by the network while the modules can be developed thinking in terms of external human-understandable identifiers.

Subsystem grouping. Information transmitted in our network could be related to different subsystems and our protocol is able to keep this relation. This allows us to mix and group data from different subsystems. For instance, this will be usefull in a ground base station to select data from different UAVs, or inside the UAV itself for grouping information from several engines, etc.

Distributed architecture. Our system should avoid centralized nodes to guarantee its correct global operation. In our vision, each module is responsible for announcing its capabilities and for providing its services without the help of any other module.

Lightweight protocol. There exist some protocols common in other areas that could be extended and modified to be used for our intention, *i.e.* RTP[3] for the transmission of streamlined information. However we decided to implement a very lightweight brand-new protocol, to be able to obtain real-time behavior in microcontrollers with limited computational resources.

4.2. Operational Scenario

Imagine the following scenario: the mission control decides to take a georeferencied video. For this task it will need the services provided from

storage, flight computer system and the camera & sensing modules. In Figure 3 we show how these different modules interact and interchange messages with the system to accomplish this complex task.

1-2. The Mission Control asks the system for the module generating the variable or *stream gps* data and the Flight Computer System responds with its location.

3-4. In a similar way, the Mission Control asks the system for the module generating the variable or stream *video camera* and the Camera & Sensing module responds with its location.

5-6. The Mission Control asks the system for the module providing the service *store stream* and the Storage module responds with its location. Now, the Mission Control knows the location of all the services it needs and it will subscribe itself to the streams that we want to store.

7. The Mission Control subscribes itself to the stream *gps data* that it is generated by the Flight Computer System.

8. The Mission Control subscribes itself to the stream *video camera* that it is generated by the Camera &Sensing module.

9-10. The Mission Control commands the Storage module to store both the *gps data* and *video camera* data streams. From this point on, the Storage Module will be storing the data in the streams without Mission Control intervention.

The software layer we have designed allows the development of complex and collaborative services that can be easily reutilized among several UAV applications. The available modules in our UAV provide an extensive set of services, covering an important part of the generic functionalities present in many missions; therefore, to adapt our aircraft for a new mission it will be enough to reconfigure the mission control to use the adapted services without adding any new software or hardware. Furthermore this abstraction layer provides some desirable capabilities to our system: redundancy, parallelism, fault tolerance and scalability. When a module distributes a data stream, multiple modules can be receiving it. The processing of this information could be done then in a parallel or redundant way, for example by processing alternate video frames in two modules or for storing two copies of the sensor data in two different storage modules.

In an analogous way, different modules can offer the same service, variable or stream. When a module asks the system for a particular action, it doesn't know exactly (and it doesn't need to know it) what node is going to answer it. This permits that in case of module failure, another module with equivalent functionalities could attend its responsibilities transparently. This arquitecture also allows an important degree of scalability and flexibility because it's not needed to know in advance in which hardware nodes are the services mapped. A hardware node can initially offer several services, and if a low performance is detected we can distribute their services among several nodes. The applications implementing the mission will be unaware of this sort of changes.

5. Communication Gateway

In a UAV several communication links may be available, for instance RF-links, SATCOM links, or wireless links. However, not all links may be available at the same time, and moreover the cost of using each link could be completely different. Depending on the flight stage and application some of the links may be more appropriate than others. Therefore, in a flexible architecture it should be possible to dynamically choose the most convenient or reliable network link.

Our system includes a communication manager (or gateway) that monitors all communication links and routes the traffic between the UAV and the ground base station through one or more communication links. Network capabilities, their link quality (bandwidth and latency), the required throughput and the usage cost (both economical and power requirements) should be taken into account. The gateway should have enough intelligence to select the appropriate routing decision in a real-time and autonomous way.

One of the key elements of this communication gateway is the fact that it provides a homogenization mechanism to hide the actual infrastructure used at any time. A data router at the entry point of the base station and another at the Mission Computer will redirect all traffic between the air and the ground segments through the best

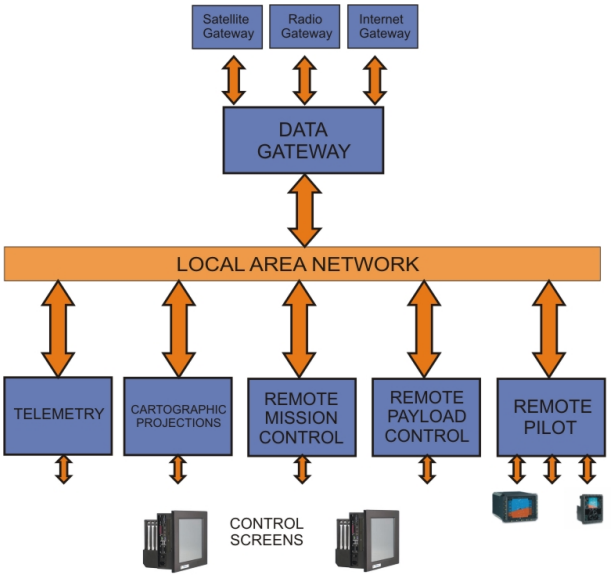

Figure 4. General architecture of the communication gateway between UAVs and the ground station.

available link. Figure 4 depicts a possible architecture of the ground station and the gateway that will provide connectivity to the UAV in flight. The gateway will concentrate all traffic from the available links and re-inject it into the LAN at the ground station.

6. Prototype Development

We are currently developing a prototype of our architecture using ARM9 hardware modules involving the following components: an autopilot module, a camera and sensing module, a communication manager with two network interfaces: a RF modem and a Wi-Fi device and finally a storage module.

The autopilot module is built from UAVNavigation's AP04 autopilot [7]. AP04 is a full integrated autopilot with manual override, capable of fully automatic take-off, waypoint-based flight plan following and landing. This device has integrated a GPS/INS navigation system that will provide positioning data for other subsystems. It also provides a RF modem for long range communications operating in the 900 Mhz band.

We could sense the environment using several sensors and cameras that provides the most important physical magnitudes. A camera and sensing module is in charge of interfacing with this devices and providing their data to other modules in the system. For this first prototype our sensor device will be a standard digital USB camera. In addition the core modules will provide temperature and voltage sensors to monitor their state.

For the communication links we will use an RF modem operating in the 2.4 Ghz band and a commercial off-the-shelf (or COTS) Wi-Fi network card. The first one will be used for the long range telemetry and the later will be used for close range configuration and data downloading without connecting cables to the UAV. We are also studying the possibility of using Line-of-sight directional Wi-Fi antennas for increasing the coverage.

Some modules will have higher bandwidth requirements and we will not be able to transmit the information to ground base station nor process it directly in the UAV. Therefore, we will provide a storage module able to save permanent information in a Compact Flash media for later processing on ground.

This prototype has the essential modules to demonstrate the viability and capabilities of our proposed architecture.

7. Conclusions

This paper has introduced a hardware/software architecture designed to be used as avionics for mission and payload control in the area of Unmanned Aerial Vehicles. The design tackles a number of elements that are critical for the operation of these systems. The architecture is a LAN-based pure distributed system, therefore being highly modular and scalable according to the requirements of the applications. A small connectivity infrastructure is required among the modules, but yet enough connectivity bandwidth could be obtained. The applications architecture is service based, following WEB-based/Internet paradigms. Therefore it conforms to the underlying hardware, offering low developing complexity through a number of standardized protocols, as well as an inherent fault-tolerance.

An initial prototype is currently being completed, the Mission Control Computer. This system will cover basic services that should exists to provide support to any UAV civil application

like: an homogeneous communication mechanism through an heterogeneous infrastructure, a massive storage service, an image/video recording service,a Flight Computer interface service and the mission control service itself.

References

[1] W3C Note: Web Services Architecture, www.w3.org/TR/ws-arch/.

[2] UPnP Forum, www.upnp.org.

[3] Schulzrinne, H., et al., RFC 3550, RTP: A Transport Protocol for Real-Time Applications www.ietf.org/rfc/rfc3550.txt.

[4] Rosenberg, J., et al., RFC 3261, SIP: Session Initiation Protocol www.ietf.org/rfc/rfc3261.txt.

[5] Spitzer, C., Digital Avionics Systems: Principles and Practice 2nd Edition, Blackburn Press, 2001.

[6] Pratt, R., Flight Control Systems: Practical Issues in Design and Implementation, Institution of Electrical Engineers (IEE), 2000.

[7] AP04 autopilot, www.uavnavigation.com.

[8] EU Civil UAV Roadmap, www.uavnet.com.

[9] Vahid, F., T. Givardis, Embedded System Design: A Unified Hardware/Software Introduction, Morgan Kaufmann, 2005.

Disclaimer

This work has been partially supported by the Spanish Ministry of Education with grant TIN2004-07739-C02-01.