



COVER SHEET

This is the author version of article published as:

Ellen, Robert A. and Roberts, Peter J. and Greer, Duncan G. (2005) An investigation into the next generation avionics architecture for the QUT UAV project. In Goh, Roland and Ward, Nick, Eds. Proceedings Smart Systems 2005 Postgraduate Research Conference, Brisbane.

Copyright 2005 (please consult author)

Accessed from <http://eprints.qut.edu.au>

An investigation into the next generation avionics architecture for the QUT UAV project

ABSTRACT: This paper investigates the next generation Unmanned Airborne Vehicle (UAV) Avionics Architecture for QUT's unmanned research aircraft. Firstly, the generalised UAV avionics requirements for small to medium civilian UAVs are developed, taking into account areas including flight control, guidance, navigation, interoperability, sensing, data buses, communications data links, airspace integration, airworthiness and system safety. Significant consideration is also given to the extensive experience with past avionics architectures. An exploration is undertaken as to the current and predicted advances in technology, such as the use of embedded computational clusters and alternate computing architectures that would provide benefits to the next-generation avionics architecture if they were to be incorporated. This is put in context with a comparison against the existing avionics architecture as well as the avionics currently being developed for the QUT UAV project.

The results of this paper discuss the various options for avionics architectures with respect to the design requirements, and the experiences with their physical implementations. Conclusions are drawn which make recommendations on the optimal standardised architecture for UAV research at QUT.

KEYWORDS: UAV, Avionics, RTOS, QNX, Airworthiness.

Introduction

The Unmanned Airborne Vehicle (UAV) industry has undergone growth of rapid proportions in recent years due primarily to the military market. However the potential benefits that can be obtained through the use of UAVs for civil applications are constantly increased, and is the topic of research in many institutions. The successful and safe operation of a UAV largely depends on the avionics package contained within it, after all this is what is replacing the pilot.

UAV research at QUT has been under way for over eight years with the use of a well developed and stable platform. In the past few years, however, a far greater focus has been placed on UAV research and development both in Australia and around the world. Consequently, the time has come to investigate issues relating to the avionics architecture at QUT. This investigation is necessary for the development of a next-generation avionics package to meet the immediate and long term future requirements of the UAV project.

1MEng Candidate, SES, BEE, S1115, 2 George St, Brisbane, QLD 4001

2PhD Candidate, CRCSS, SES, BEE, S1107, 2 George St, Brisbane, QLD 4001

3PhD Candidate, CRCSS, SES, BEE, S1107, 2 George St, Brisbane, QLD 4001

Along with the increase in the prevalence of UAVs come tighter regulatory control and the need for the correct procedures to be followed and certification to be gained. As such, a robust and comprehensive design process must be employed during all stages of UAV avionics development.

The architecture for UAV avionics vary significantly based on the operational role that the UAV will be required to carryout. However, in general the core architecture is dictated by two basic requirements, navigation and control. These two requirements are crucial for all UAVs to be functional. To satisfy the two broad requirements a situational and self awareness must be maintained by the UAV at all times along with the ability to modify its state. This requires varied sensor technologies, computing power, software, data link and actuators.

New technology and methodology available for the UAV sector include miniaturisation of hardware, innovative sensor technologies and increased autonomy. Emerging technologies that provide increased promise for the UAV sector include embedded computational clusters and alternate computing architectures such as reconfigurable logic and dedicated parallel pipelined processors.

The paper begins with a brief overview of a basic UAV avionics architecture followed by an overview of the system design and certification process applicable to the UAV avionics architecture development. This is important as any architecture design must ultimately be certifiable under the civilian UAV certification framework. Following this a description of the high-level UAV functional requirements is provided to give an insight into the amount and complexity of computational processing to be undertaken by the UAV's systems. Next, the generic functional components and the preferred software environment are introduced.

To address the computational performance needed to fulfill the UAV functional requirements, current and future technologies are discussed, with emphasis placed on non-conventional solutions. The paper concludes with results of an investigation into the use of low-cost embedded Linux platforms, their potential benefits and disadvantages over the current system and a proposal for a next-generation UAV architecture.

Avionics System Overview

The avionics of a UAV has to fulfill the role of the avionics on a traditional aircraft as well as accommodate the level of autonomy that the UAV is desired to have. There is a trend to make the UAV increasingly autonomous as to reduce the skill level necessary for operators as well as the operator workload. Generally UAVs have limited autonomy particularly in the takeoff and landing phases of flight. This results in a need for a data-link and ground station of some description for navigation and control in at least these flight phases. The rest of the avionics architecture consists of the flight computer, sensors and actuators.

The basic UAV avionics architecture follows the flow as depicted in Figure 1. The functionality and requirements of each of the components of this architecture will be addressed within the following sections.

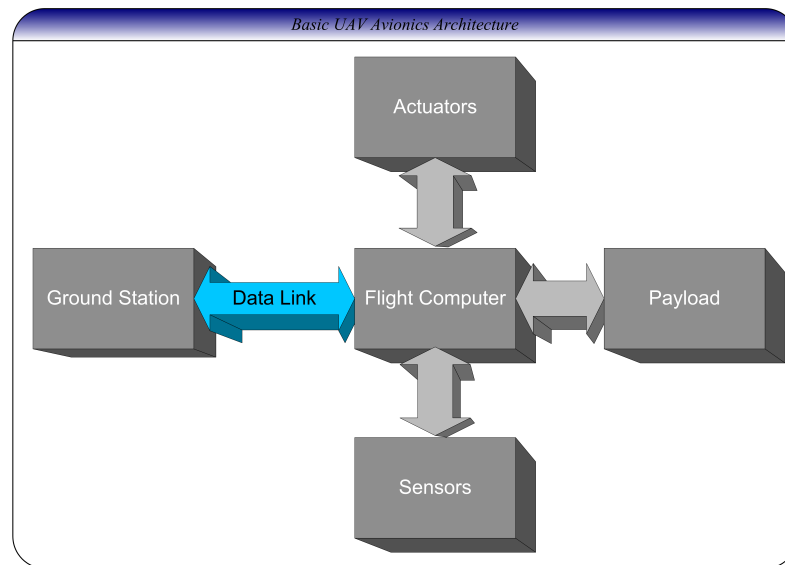


FIG. 1— *Basic UAV Avionics Architecture*

System Design Process

In this section, a concise overview of the systems development process being undertaken is provided. The process consists firstly of establishing the design context, developing system requirements, performing system design, and finally design validation and certification.

Establishing the Context

One of the most fundamental steps is establishing the context of the system development. This involves identifying the operational concept, functional requirements and regulatory requirements. These three aspects set the stage for the development of system requirements and system design.

In the case of UAV development, the Australian Civil Aviation Safety Authority (CASA) has established national UAV regulations. The regulations primarily pertain to the operation of UAVs, however guidance material for the regulations contains information relevant to the system design.

System Requirements Development

Various levels of requirements are developed at all phases of design. In the preliminary design phase, the functional and performance requirements dominate.

System Development

The context of the development will provide a specification for the minimum functionality of the UAV. The system development processes takes the functional, regulatory and operational requirements, and develops a system specification which will fulfill the requirements.

Design Tradeoffs

The primary design tradeoffs encountered are:

- Weight
- Redundancy
- Robustness and Environmental Protection
- Power

Design Validation and Certification

The system design will lead to the development of a design safety case. The design safety case is provided to the certification authority as evidence that the UAV will be safe when employed in its intended operation.

The design safety case includes a detailed Failure Modes Effects and Criticality Analysis (FMECA) of the UAV design. During the early phases of architecture development, it is critical to consider the inherent safety in a design, rather than attempting to design in safety at a later stage. A qualitative FMECA analysis should be undertaken on the proposed avionics architecture to identify the critical reliability points of the design.

Once the certification authority is satisfied that the UAV and its operation are airworthy, it will issue certifications to both the air vehicle system, and its operators.

With a proposed design process in mind, the requirements of the UAV architecture can be addressed.

Functional Requirements

The functional requirements of UAV avionics architecture under consideration are related to the research objectives of QUT, namely autonomy of the UAV and human operator workload. The core components of autonomy are flight control, navigation and guidance. Higher levels of autonomy, which in ~~term~~turn reduce operator workload, include (in increasing order) sense-and-avoid, fault-monitoring, intelligent flight planning and reconfiguration (See Figure 2).

Navigation

The UAV must have a means of finding its position at any point on the earth at any time. This results in the requirement for a robust, high accuracy, highly available and high integrity navigation system. The obvious choice for such a navigation system is GPS, however even GPS must be augmented with additional sensors to ensure the robustness and integrity of the navigation solution.

Guidance and Flight Control

The UAV must generate the steering commands and subsequent control surface deflections to adequately find its way along the chosen flight path. These calculations are relatively simple compared with the flight planning and navigation algorithms, however ideally require the use of floating-point calculations.

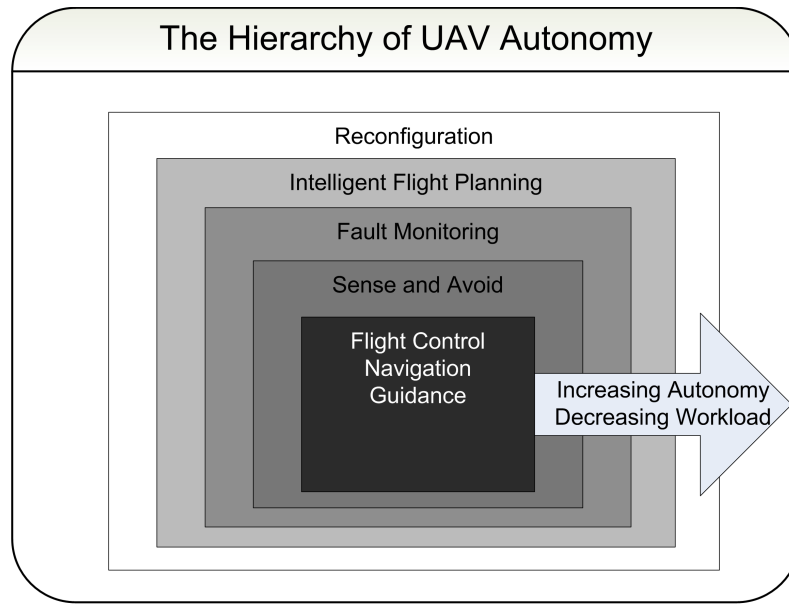


FIG. 2— *The Hierarchy of UAV Autonomy*

Sense-and-Avoid

One of the biggest limitations to the widespread use of unmanned vehicles in civilian airspace has been the Sense-and-Avoid problem [1]. In manned civilian aviation, See-and-Avoid is the primary mechanism by which piloted aircraft avoid collisions with each other. Obviously this is impractical for widespread use of unmanned vehicles, so they must achieve an equivalent level of safety to that of manned aircraft operations.

There is currently a large amount of research being conducted on the UAV Sense-and-Avoid problem. Active solutions include the use of radar or TCAS to detect collision threats, however this requires high amounts of electrical power, and are quite heavy (in the order of 20kg or more). Passive solutions include the use of machine vision, which reduces the power requirement to a degree but, however, has a high computational requirement.

Fault Monitoring

To ensure the integrity of the UAV's systems, fault monitoring must be continually conducted on flight and mission critical systems. Fault monitoring ensures that undetected system faults will not lead to a catastrophic failure of the aircraft's systems which could lead to human casualties on the ground.

Intelligent Flight Planning

The UAV system must have the capability to plan and re-plan its own flight path. This results in the requirement for a high level computing environment where flight planning algorithms can be run. The flight planning operation requires knowledge of the UAV's surroundings; including airspace, terrain, other traffic, weather, restricted areas and obstacles.

The UAV must plan the optimal route for its mission, considering the local environment, flight time and fuel usage. In the event of system faults, the UAV must have the capability to reconfigure itself and re-plan its flight path in a fail-safe manner. The flight planning requirements result in a significant requirement for memory and floating point operation performance.

Payload

A civilian UAV is designed to perform a particular mission at a lower cost or impact than a manned aircraft equivalent. The payload is the equipment installed by the customer that performs a specific task.

The payload will require at least a space, weight and power allocation. However, certain payloads may also require access to UAV system data, such as position, airspeed, or altitude. Thus, a mechanism must exist to provide UAV data to the payload in a manner such that the failure of the payload cannot impact the safety of the UAV's own systems (for example denying access to the data-bus by grounding signal lines)

Operating System and Software Considerations

The Operating System (OS) Application Programming Interface (API) is a very important consideration, not only from the point of view of execution, but also in the ease of system development. Due to the time critical nature of flight control, high reliability and real-time execution is mandatory.

The Portable Operating System Interface (POSIX) IEEE 1003.1 is the preferred operating system interface standard since it is widely supported, and allows easy porting of applications between the various flavours of Unix, Linux and QNX.

QNX is currently used widely in the QUAV group for desktop and embedded computing requirements as it provides an excellent feature set and performance. The advantages of QNX are fully evident with the process of porting applications from Linux to QNX being very straight forward, in many cases simply requiring a re-compile under the new OS.

Analysis of Evolution Options

The continual development of new algorithms for aerospace automation places an ever-increasing demand upon the avionics platforms of UAVs. Computational performance of UAV avionics must match this demand in dimensions such as processing speed, input/output throughput, memory size while at the same time providing robustness, reliability and a small footprint in both physical size and mass and power consumption. This section explores some of the options available in specifying a next-generation UAV avionics architecture in the immediate, short and medium terms. These options include the up-scaling of the current platform; out-scaling using proven and familiar components, and; investigation of leading-edge architectures not historically applied in QUAV avionics.

Immediate Term: Upscale of current platform

The most immediately applicable option in the development of a next-generation avionics architecture is to continue with the current largely monolithic platform. Typically this is a single high-level computational unit such as a PC-104 running QNX attached to dedicated micro-controllers (or other ASIC devices) that interface with sensors and actuators. For the current architecture to match rising demand the specifications of the computational unit and the peripherals must be improved - that is: the PC-104 (or equivalent) must run at a faster clock-speed and have more RAM and secondary storage.

The amount of speed-up that can be achieved by upgrading the current platform is limited by four interrelated constraints that are introduced by the realities of UAV operations. Cost, power consumption, volume and mass of avionics are strictly bounded quantities dictated by the UAV airframe and the allotted budget. Ideally downward pressure should be applied to avionics for all of these constraints but the interaction between them must be considered. For instance, reducing the mass of the avionics may mean the use of lighter but more expensive materials, pushing up the cost.

Up-scaling is also subject to the law of diminishing returns. As applies to many high-tech products there exists a 'sweet-spot' or minima in the price-performance function where the highest 'bang-for-your-buck' is achieved. On either side multiple factors, such as marketing and technological, limit the value of increasing the performance of the avionics. This leads to small increments of performance requirements but at disproportionately greater cost.

Once extra performance gains in the avionics hardware are exhausted through up-scaling the natural immediate-term option available is the minimisation of the avionics software. Smaller, faster avionics code can be achieved in two ways: more efficient algorithms can be developed or the existing code can be otherwise cut down or 'optimised'. The first method has merit and can, in the case of reducing the order of complexity of a problem, provide extra breathing space for the avionics platform. The second method is not so desirable. Compromises made in the code can lead to reduced modularity, readability, interoperability, and so on which in the long run is detrimental to the success of the avionics platform.

Clearly, up-scaling the current platform will help provide computational performance to run more demanding avionics software. However, the sustainability of up-scaling is very poor. Alternatives to simply using more powerful CPUs must be sought to continually keep up with computational demand.

Short-term: Out-scale Leveraging Current Components

Out-scaling involves the application of proven and familiar components (which may be less or as powerful as are used in the current platform) in parallel to distribute the computational load. Considering the currently used components this implies the use of several PC/104 or XScale based platform nodes networked together on a single airframe. By spreading the load among multiple CPUs slightly slower, but more cost effective hardware from a diminishing returns point of view, can be used.

The decision as to the number and configuration of units would depend on factors such as power, size and weight constraints, input/output requirements and computational demand from the avionics software. A simple example may be a single PC/104 as a master and one or more XScale based processors managing groups of I/O. The software on board may include a diagnostic system to determine the reliability of sensors and actuators. The XScale based processors could handle some of the sensor filtering duties while the PC/104 takes filtered data and feeds it to decision support modules.

The partitioning of the avionics software for the UAV among the nodes is dependent upon the distributed software architecture employed. A basic, static, system would have its software partitioned at design time. A more advanced system would allow the system to be reconfigured but only through the static assignment of application modules to nodes at start-up. This type of system can be called a statically reconfigurable system. Modularity, robustness, reliability and fault-tolerance in static and statically reconfigurable systems is improved over a monolithic architecture in that the different software modules are loosely coupled and one node failing does not necessarily cause a critical failure.

Static reconfiguration, however, does not provide a high level of robustness or fault-tolerance in that the code assigned to a failed node cannot be migrated to a different node at run-time. A dynamically reconfigurable system architecture in which software modules are transient and provide location-transparent services further improves robustness, reliability and fault-tolerance. Code running on a failing node can be moved to one that is operational. Dynamically reconfigurable systems can also take advantage of the varying load presented by the avionics software by searching for more efficient arrangements of code modules.

Distributed avionics architecture does not require all of the previously developed software to be thrown out. The building blocks of distributed systems are the same modules as are currently used. What is required is a framework that allows the code modules to become mobile and provide their services in a location-transparent way.

Distributed Embedded Framework

Research involving distributed embedded systems with application to UAV avionics as well as other related domains (such as robotics, manufacturing and mining) is currently under way [2]. The research aims to develop a framework that will allow the intelligent control and automation software applied in these areas to be parallelised and distributed over a network of heterogeneous, embedded processing nodes. Software developed with the framework will be able to produce the dynamically reconfigurable systems as mentioned previously.

The research objectives to be pursued include:

- Performance improvement and fault-tolerance through reconfigurability
- Platform-neutrality and location-transparency through component technology
- Ease-of-use for engineers and users

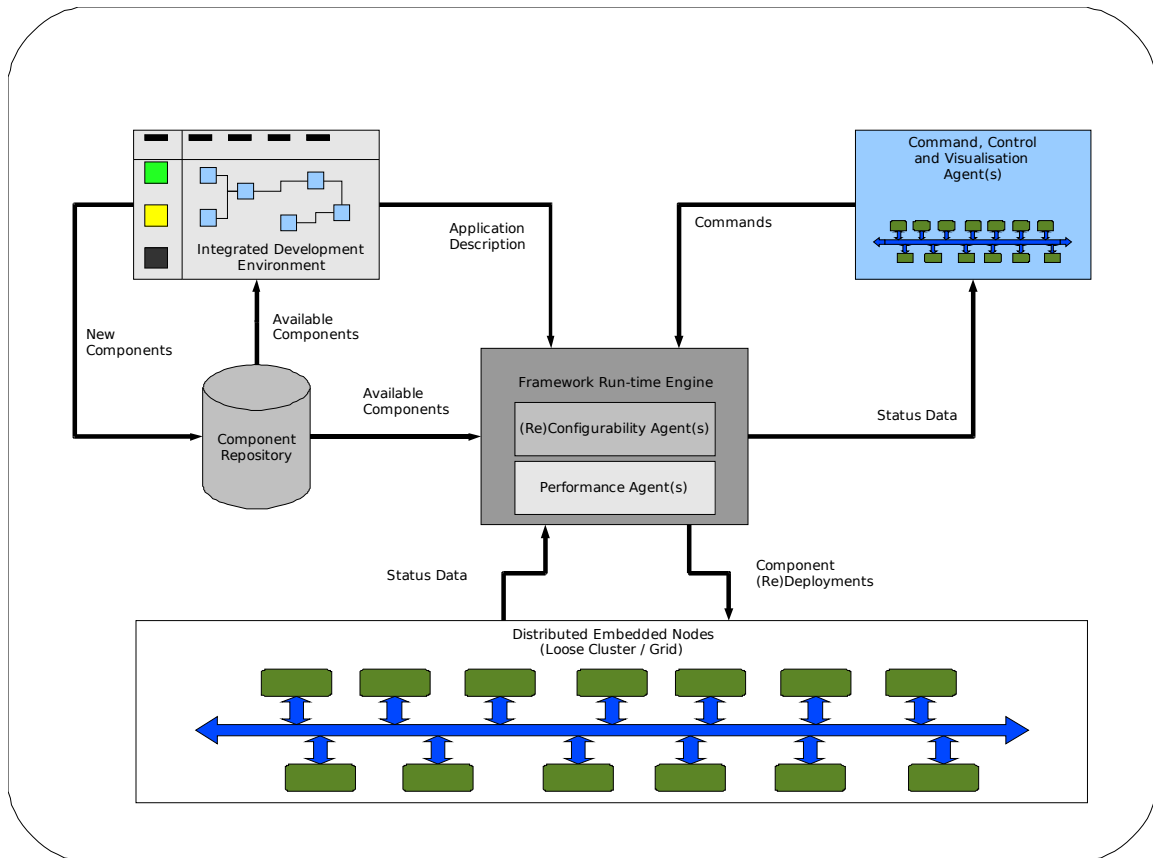


FIG. 3—*Proposed Framework Architecture*

An architecture for the framework has been proposed to meet these objectives. The main artifacts of the proposed framework include (see Figure 3):

- An Integrated Development Environment (IDE)
- A repository of platform-neutral component definitions and their (potential multiple) platform-specific, location-transient realisations
- Run-time engine agents for reconfigurability and performance improvement
- User agents for command, control and visualisation of the distributed applications
- The underlying distributed embedded system arranged as a heterogeneous, loosely coupled cluster or grid of embedded nodes

The IDE will be used to assemble (as block diagrams) distributed embedded applications from a repository of platform-neutral components (which could include for example signal filters, I/O interfaces, artificial neurons and so on). The range of applications that could be developed in this fashion appears to be quite broad but an example in UAVs could be a diagnostic system reasoning about the reliability of the aircraft's sensors and actuators. Such a system could be represented as a block diagram with the blocks (components) including sensor signal filters and decision support tools such as fuzzy inference systems or neural nets.

At compile-time a description of a complete application will be sent to the framework's run-time which would be operating within the distributed system itself. The run-time, through its constituent software agents, will initiate and monitor the application. The currently proposed run-time agent types include a reconfigurability agent and a performance agent.

Reconfigurability agents will coordinate the initial application deployment by instructing the application components to move to initial nodes. Under node failure situations the reconfigurability agents will re-deploy the affected component to new locations. Reconfigurability itself will be achieved through the application of code mobility which allows light-weight software fragments to be moved automatically from one execution location to another in the network at run-time with minimal interruption.

Performance agents will reason about the throughput, memory, footprint and power consumption of the distributed system and request reconfiguration to improve these attributes. The Theory of Constraints (TOC) has been proposed as a technique for throughput improvement in the frameworks distributed applications. TOC presents a simple, intuitive algorithm for identifying and alleviating throughput bottlenecks in parallel pipelined systems [3].

User agents will provide feedback and command and control capabilities to users. Multiple delivery methods are available to present user agents including native graphical user interfaces, dynamic web pages or web-based applets. Together with the IDE, user agents will be tested qualitatively for ease-of-use.

The underlying network of embedded systems is the substrate upon which the framework will be built. Target platforms include the PC/104 and XScale based platforms as well as Motorola Coldfire microprocessors which are available to the school. The common networking facility for these platforms is Internet Protocol (IP) and Ethernet. Deterministic buses used in avionics, such as Controller Area Network (CAN), may also be investigated to provide higher inter-node robustness.

Through the use of a distributed framework such as this it is hoped that high-level avionics will be able to run faster and more reliably and be better value-for-money than the current platform.

Medium Term Adoption of Alternate Technologies

To leverage the current skill and knowledge base of the QUAV researchers, the immediate and short-term options considered rely upon embedded PC-style platform components. However, looking ahead, future-generation avionics architectures need not be so limited. There is a broad range of technology that could be applied in UAV avionics to provide here computational performance. Even if the search is constrained to proven, but not necessarily familiar, technologies there are several options. If computing devices are considered in terms of the specificity of their use, PC-style CPUs, such as x86, Sparc, ARM or even Harvard occupy the middle ground between general and totally specific. The instruction sets of these CPUs allow for a reasonable range of operations that can be performed in a sequential fashion. However, if each end of the specialisation spectrum is explored (as shown in Figure 4), two technologies for which there is a moderate level of research activity at QUT can be identified.

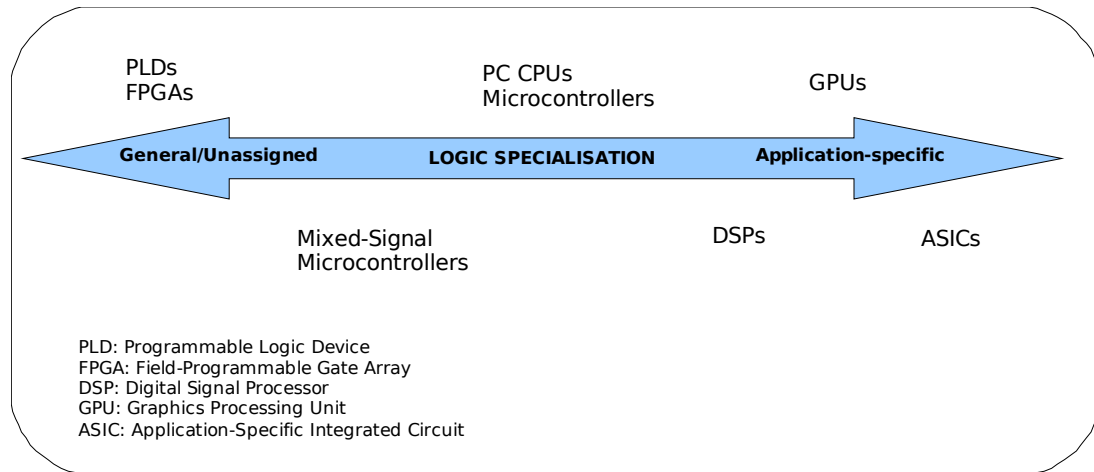


FIG. 4— *Logic Specialisation of Selected Hardware Architectures*

At the general end of the spectrum reconfigurable hardware devices, such as programmable Logic Devices (PLDs) and Field Programmable Gate Arrays (FPGAs), have been applied to many domains, including aerospace, for flexible and high-performance parallel hardware implementations. These devices, commonly referred to as reconfigurable hardware or reconfigurable computing technology, are arrays of unassigned logic blocks that can be programmed to do any logical function. They provide both the flexibility of programming software with the speed of logic hardware. Application of FPGA technology to aerospace researched in Queensland recently includes adaptive filtering for space[4] and avionics life-cycle management[5]. Given the growing body of research into its application in aerospace and the availability of low-cost reconfigurable platforms such as Egret[6] the adoption of reconfigurable hardware to a next-generation UAV avionics architecture is an attractive medium-term option.

If the application-specific end of the specialisation spectrum is considered, the less than intuitive suggestion of Graphics Processing Units (GPUs) can be made. General Purpose computation on Graphics Processing Units (GPGPU) is a technology that allows a computer graphics processor to do a range of general computations in a parallel, pipelined manner. Modern GPUs are floating-point vector processors meaning they can perform a single instruction on multiple data points in parallel. For problems that can be efficiently solved using the stream approach GPUs offer massive performance gains over CPUs. Many matrix-based, signal processing and equation solving tasks have been shown to benefit greatly from GPU implementation[7].

Developing software for GPUs involves manipulating the problem domain until it fits the graphics processing paradigm of vertices and textures. Once this is done input data can be passed to the GPU as if it were a set of vertices and textures and processed as such. The output is stored in a memory-mapped frame buffer[8].

GPU implementations in Virtual Reality and other image processing and machine vision applications are becoming more common. Typically these systems are implemented upon full desktop or workstation computers.

There are, however, several limitations attached to GPUs that make their adoption in avionics architectures unlikely in the short-to-medium term. The primary constraint is power consumption. GPUs have comparable transistor counts to high-end desktop PC CPUs. The power required to attain the high performance rules current GPUs out for most UAVs. Another point against GPUs is the cost and time involved in developing custom PCBs and software to utilise them.

In the light of these limitations it is unlikely GPUs in their current form will be applicable to a next-generation UAV avionics architecture. DSP systems are a more suitable technology that provides similar levels of parallelism to GPUs but at far lower power consumption.

Next Generation Avionics Architecture

Given the functional requirements, and design constraints, there is an obvious emphasis on increasing computational power, whilst reducing component footprints in terms of weight, size and power consumption.

With this in mind, a trade-study in to the use of low-cost embedded processors has been initiated. A Gumstix Connex 400XM and a Stargate SBP-400 have been purchased for evaluation. Both of these computers are based on the Intel PXA255 embedded processor, however the Stargate incorporates the StrongARM SA-1111 chipset. This class of embedded processors is commonly found in handheld devices such as Personal Digital Assistants (PDA).

The following table highlights some of the key parameters of the avionics architecture that are significantly different between the PC/104 legacy platform and the new XScale based platform.

TABLE 1—Avionics Architecture Platform Parameters[9,10]

| | PC/104 Platform | XScale Based Platform - Stargate |
|---------------------|-----------------|----------------------------------|
| Power | 18 Watts | < 2.5 Watts |
| Weight | 1.2 Kg | < 200 grams |
| Size | 20 x 10 x 10 cm | 10 x 6.5 x 5 cm |
| CPU Clock Speed | 66 MHz | 400 MHz |
| Memory | 32 MB EDO RAM | 64 MB SDRAM |
| Floating Point Unit | Hardware | Software Emulated |

It is evident from the table above that the next-generation platform is superior to the legacy PC/104 platform, however there are some things that must be taken into consideration.

Hardware-introduced Constraints

The embedded XScale CPUs do not have hardware Floating Point Coprocessor Units. This means that floating point operations are performed in software, which significantly burdens the CPU. The result of this is that the X-Scale based computers may not be suitable for the high level functional requirements such as Sense-and-Avoid, or Intelligent Flight Planning.

Software thus must be designed to minimize the use of Floating Point Operations, which can significantly complicate the software design. For example sensor readings should be kept in integer format for operations, which can limit the resolution if algorithms are not designed correctly. Similarly, map operations requiring trigonometric calculations (used extensively in the navigation algorithms) will be much slower, and thus cannot be performed as often. This can reduce the navigation system performance.

Operating System

As sold, the XScale based platforms (Gumstix and Stargate) ship with a Linux distribution based on uCLinux and uClibc. The Linux operating system installed on these devices are suitable for most applications however the specific application of flight control is very demanding and requires a high level of robustness with respect to timing. That being the case, a real time operating system is necessary and the clear leading contender for this is the QNX RTOS. The installation of the QNX operating system on these devices is underway and is expected to be completed shortly.

Proposed Avionics Architecture

Given the objectives and constraints outlined, the proposed next-generation UAV avionics architecture is as shown in Figure 5.

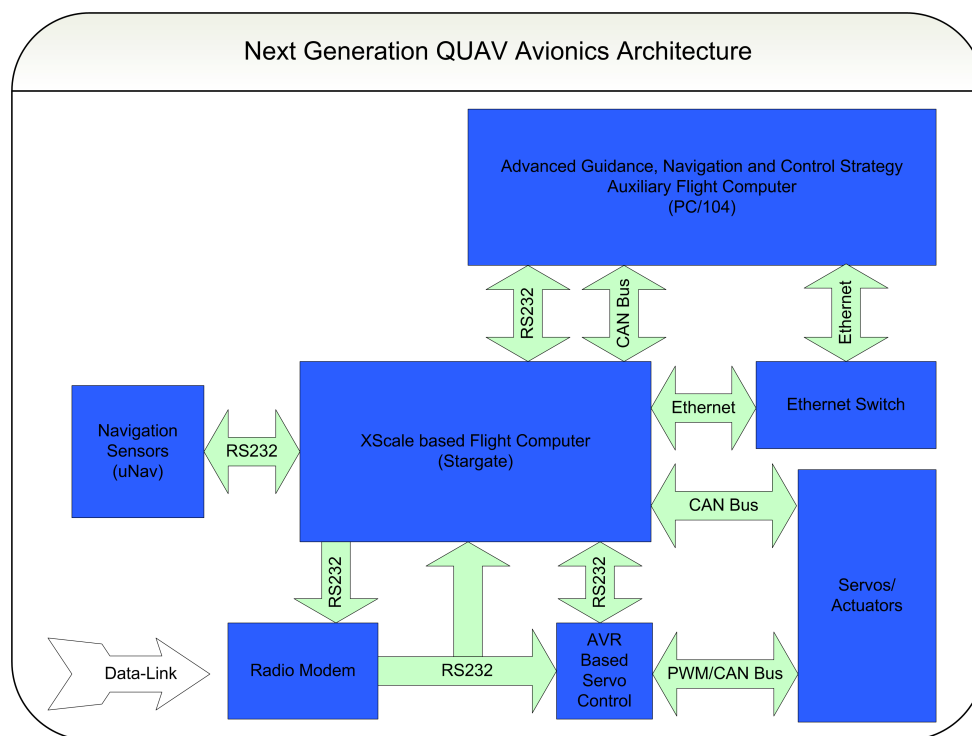


FIG. 5— *Proposed Next-Generation UAV Avionics Architecture*

Relating the proposed architecture back to the basic architecture in Figure 1, the flight computer function will be provided by the XScale-based Stargate. For higher-level payloads, PC/104 computers will be used. The UAV sensors will be handled by a uNav module while the servos and actuators by either a remote controller AVR based system or by the flight computer. These components will communicate via several bus schemes dependant upon the bandwidth and determinism as outlined in Table 2.

TABLE 2—Avionics Architecture Buses

| Bus | Bandwidth | Deterministic | Use |
|----------|--------------|---------------|---|
| RS232 | 115.2 kbit/s | YES | Short-range inter-module communications |
| CAN | 1 Mbit/s | YES | Medium-range high-speed critical control and status |
| Ethernet | 100 Mbit/s | NO | Non-critical status and bulk data |

Conclusion

The market for UAVs and their research and development have mutually expanded in recent years. While QUT has had a relatively mature involvement in UAV research, the need to upgrade the avionics architecture of its aircraft has been identified. The major considerations are the definition of avionics architecture, the process for designing a new one and what is required.

Given these considerations this paper has discussed a next-generation avionics architecture of the QUT UAV project. After outlining the development process and functional requirements, options for immediate, short-term and medium-term adoption for UAV architecture have been discussed. For the immediate future the up-scaling of the currently utilized PC/104 and XScale platforms has been proposed. This involves the purchasing of newer, faster and more fully featured hardware components. In the short-term, out-scaling on the current platform has been identified as a desirable option. Out-scaling involves again using the current platform, but running multiple nodes to process the avionics software in parallel. To this end, a proposed software framework has been outlined that could provide the infrastructure to achieve the out-scaling of the avionics architecture. Mid-term options considered included FPGAs and GPUs, which have the potential to provide flexible and high-performance processing capabilities. Since FPGAs have a history in UAV and aerospace they are a recommended option. GPUs, however, have several limitations which bar them from practical use in UAV avionics.

Taking the functional requirements and the evolution strategy outlined, the next-generation UAV avionics architecture has been proposed. It is a compromise between an up-scale and out-scale of the current platform, utilizing a low-cost XScale for core autonomous functions and a PC/104 for higher-level modules. It is believed that this configuration provides a high-level of scalability, robustness and performance and will meet the immediate and short-term requirements of the QUT UAV project.

Acknowledgments

This work was carried out in the Cooperative Research Centre for Satellite Systems with financial support from the Commonwealth of Australia through the CRC Program.

This research was supported in part by a grant of computer software from QNX Software Systems Ltd.

References

- [1] Walker, M. (2001). The Regulatory Environment. UAV Australia Conference and Exhibition, Melbourne, Australia.
- [2] R. A. Ellen and D. A. Campbell, "A Framework for Executing Computational Intelligence Upon Distributed Embedded Nodes," presented at IASTED International Conference on Computational Intelligence, Calgary, Canada, 2005.
- [3] E. M. Goldratt, What is this thing called Theory of Constraints and how should it be implemented? Great Barrington: MA: North River Press, 1990.
- [4] S. Visser, A. Dawood, and J. A. Williams, "FPGA Based Real-time Adaptive Filtering for Space Applications," presented at IEEE International Conference on Field Programmable Technology, Hong Kong, 2002.
- [5] N. W. Bergmann and J. A. Williams, "Avionics Upgrade Management using Reconfigurable Logic," presented at Australian International Aerospace Congress, Brisbane, 2003.
- [6] N. Bergmann, J. Williams, and P. Waldeck, "Egret: A Flexible Platform for Real-Time Reconfigurable System-on-Chip," presented at The International Conference on Engineering of Reconfigurable Systems and Algorithms, Las Vegas, Nevada, 2003.
- [7] N. Goodnight, R. Wang, and G. Humphreys, "Computation on programmable graphics hardware Computer Graphics and Applications," IEEE Computer Graphics and Applications, vol. 25, pp. 12-15, 2005.
- [8] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A Survey of General-Purpose Computation on Graphics Hardware," presented at Eurographics 2005, State of the Art Reports, Dublin, Ireland, 2005.
- [9] "Stargate Developers Guide Rev. A," Crossbow 7430-0317-13, June 2005.
- [10] "MNAV100CA User's Manual Rev. B," Crossbow 7430-0198-01, October 2005.