COMP4DRONES

DELIVERABLE

D3.1 – Specification of Integrated and Modular Architecture for Drones

Project Title	COMP4DRONES
Grant Agreement number	826610
Call and topic identifier	H2020-ECSEL-2018
Funding Scheme	Research & Innovation Action (RIA)
Project duration	36 Months [1 October 2019 – 30 September 2022]
Coordinator	Mr. Rodrigo Castiñeira (INDRA)
Website	www.comp4drones.eu

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826610

Document fiche	
Authors:	See Page 4
Internal reviewers:	Muhammad Tufail [SHERPA] Jean-Patrick Mascomere [TOTAL]
Work Package:	WP3
Task:	T3.1
Nature:	R
Dissemination:	PU

Document History

Version	Date	Contributor(s)	Description
V0.3	25/05/2020	Mahmoud Hussein, Reda Nouacer	Initial draft of the deliverable with its contributors' identification
V1.2	10/07/2020	See Page 4	Complete draft of the deliverable
V1.3	15/07/2020	Muhammad Tufail, Jean-patrick Mascomere	Internal review process
V1.5	31/07/2020	Mahmoud Hussein, Reda Nouacer	Final version of the deliverable

Keywords:	Architecture, Autopilot, Avionics, Automotive, Robotics, System Functions, Payloads, Tools, and Hardware Platform
Abstract (few lines):	This deliverable provides our **vision** for the **specification** of the COMP4DRONES drone's **architecture.** First, the **state of the art architectures** for embedded software are presented. It includes architectures in avionics, automotive, and robotics domains. Second, based on the state of the art architectures and return of experience of drone providers in the project, an **initial architecture** that will be adopted in the project is described. Finally, the different **generic components** that will be developed and integrated to the architecture are presented. These components are divided into four categories: system functions, payloads, hardware platforms, and tools.

DISCLAIMER

ACKNOWLEDGEMENT

This document is a deliverable of COMP4DRONES project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Nº 826610

D3.1 Authors:

Partner Name	Contributors
INDRA	Rodrigo Castiñeira, Adrian Irala
IMEC-BG	Hiep Luong, Michiel Vlaminck, Murali Jayapala
BUT	Svetozar Nosko
UWB	Miroslav Flídr, Ondřej Severa
ENAC	Fabien Bonneval, Gautier Hattenberger
SIEMENS	Federico Cappuzzo
SCALIAN	Raphaël Lallement
ENSMA	Yassine Ouhammou, Matheus Ladeira, Emmanuel Grolleau, Patrick Coirault
UNIMORE	Andrea Marongiu, Alessandro Capotondi
UNISANNIO	Giuseppe Silano, Luigi Iannelli, Valerio Mariani
UNISS	Francesca Palumbo, Tiziana Fanni
UNIVAQ	Stefano Digennaro
TOPVIEW	Alberto Mennella
EDI	Rihards Novickis, Daniels Jānis Justs
SkyA	Philipp Knopf
TUE	Kees Goossens, Dip Goswami, Saeid Dehnavi
ACORDE	Fernando Herrera, David Abia
HIB	Diego Fuentes
IKERLAN	Leire Rubio
CEA	Mahmoud Hussein, Reda Nouacer, Ansgar Radermacher, Fabio Arnez, Huascar Espinoza, Chokri Mraidha
UNICAN	Eugenio Villar
AI	Leonardo Napoletani
SM	Jiri Bartak, Petr Bartak
ATE	Quentin Godbert
UDANET	Bianchi Domenico, Preziuso Maurizio
MODIS	Daniela Parletta
ROT	Niccolò Cometto, Diego Grimani, Maria Luisa Scalise
IFAT	Dominic Pirker, Rainer Matischek

Table of Contents

Table of Figures

Table of Tables

Definitions, Acronyms and Abbreviations

Acronym	Title
AMC	Aerial Mission Controller
APEX	Application/Executive interface
ARA	AUTOSAR Runtime for Adaptive Applications
AUTOSAR	AUTomotive Open System ARchitecture
COTS	commercial-off-the-shelf
FTTI	Fault Tolerant Time Interval
FPGAs	Field Programmable Gate Arrays
GCS	Ground Control Station
GNSS	Global Navigation Satellite System
HSI	Hyperspectral imaging
IMA	Integrated Modular Avionics
IMU	Inertial measurement unit
MAS	Multi-Agent System
ROS	Robot Operating System
RTE	Runtime Environment
RTOS	Real-Time Operating System
SLAM	Simultaneous Localization and Mapping
SoC	Systems-on-chip
TLS	Transport Layer Security
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle
UWB	Ultra-Wideband
YARP	Yet Another Robot Platform

Executive Summary

Drones/Unmanned Aerial Vehicles (UAVs) can perform air operations that manned aircrafts struggle with. Their use brings significant economic savings and environmental benefits whilst reducing the risk to human life. However, current drone embedded architectures are organized in loosely coupled monolithic boards, each composed of processor, memory and communication resources (e.g. flight control, planning and vision boards), which in turn does not support the continuous development of drone applications such as the ever-increasing demand on autonomy. Drone embedded platforms must meet this demand, while still providing safety, robustness and a small footprint in physical size and power consumption.

In this deliverable, we provide our **vision** for the **COMP4DRONES architecture** that enables the **easy integration** and **customization** of qualified systems embedded in drones. To do so, we first review existing well-advanced **architectures for embedded software**. The reviewed architectures are from avionics, automotive, and robotics domain. We also review the different existing autopilots' architectures.

Second, based on existing architectures best practices, and needs/requirements for designing a drone architecture, an **initial architecture** that will adopted in the project is introduced (this architecture will be specified in detail in the deliverable D3.2).

Finally, a number of **generic components** will be developed and integrated with the proposed drone architecture. Thus, we describe these components, their current state of the art, planned improvements, and metrics to measure their improvements.

1 Introduction

Current drone embedded architectures are organized in loosely coupled monolithic boards, each composed of processor, memory and communication resources (e.g. flight control, planning, and vision boards). This separation ensures that the subsystems operate almost independently from each other and avoids interference coming from the other parts. However, this approach does not support the continuous development of drone applications such as the ever-increasing demand on autonomy. Drone embedded platforms must meet this demand, while still providing safety, robustness and a small footprint in physical size and power consumption. Another drawback is that a huge number of different resources are needed to perform integration, customization, and maintenance in terms of component provisioning and handling.

In existing open autopilots, even if a modular object approach is employed, to the best of our knowledge, there is no component-based autopilot approach. However, this would be a very important contribution to raise the diversity of sensors and actuators that can be used for piloting a drone, and especially in the scalability of drones in their different versions and improvements.

One of the main objectives of COMP4DRONES project is to **ease the integration and customization** of qualified systems embedded in drones. This objective is going to be achieved through providing reference architecture of a flexible embedded platform inspired from the avionic practice. This reference architecture will be based on modularity and separation of concerns. The availability of such an integrated modular architecture will be a step forward with respect to the current state-of-the-art.

To specify the architecture that is going to be adopted in the COMP4DRONES project, in Section 2, we review the existing well-advanced **architectures for embedded software**. The reviewed architectures are from the avionics, automotive, and robotics domains. In addition, software architectures from the project partners in such domains are described. Furthermore, different existing autopilots' architectures are presented.

In Section 3, the existing architectures are analyzed to get the best practices and the needs/requirements for designing drone architecture are described. Then, based on the current state of the art and the best practices, an **initial architecture** that will adopted in the project is introduced (this architecture will be specified in detail in the deliverable D3.2).

Finally, in Section 4, a number of **generic components** that will be developed and integrated with the proposed drone architecture are described. These components are divided into four categories: system functions, payloads, hardware platform, and tools. For each generic component, we give its description, the current state of the art, the improvements that are going to be performed during the project, and how these improvements going to be measured.

2 State of the Art Embedded Software Architectures

In this section, we review the architectures that have been proposed in a number of embedded systems' domains. These domains are avionics, automotive, and robotics. We also present the state of the art of the existing autopilots' architectures.

2.1 Avionics Domain

In this section, we start by describing the Integrated Modular Avionics Architecture followed by the ARINC 653 standard, and an architecture proposed by Indra.

2.1.1 IMA: Integrated Modular Avionics Architecture

Growth in aerospace industry raised the need for utilizing increased processing power, communication bandwidth and hosting multiple federated (decentralized) applications into a single integrated platform. This has been realized with Integrated Modular Avionics (IMA) Architecture. The central idea of IMA is multiple applications share the same resources like processing unit, communication bus etc. (see Figure 1).

Integrated Modular Avionics Architecture also encourages each module to host different applications which might have separate criticality level. Each module is isolated from other by robust partitioning mechanisms. This makes breaking the module into multiple virtual computers easier.

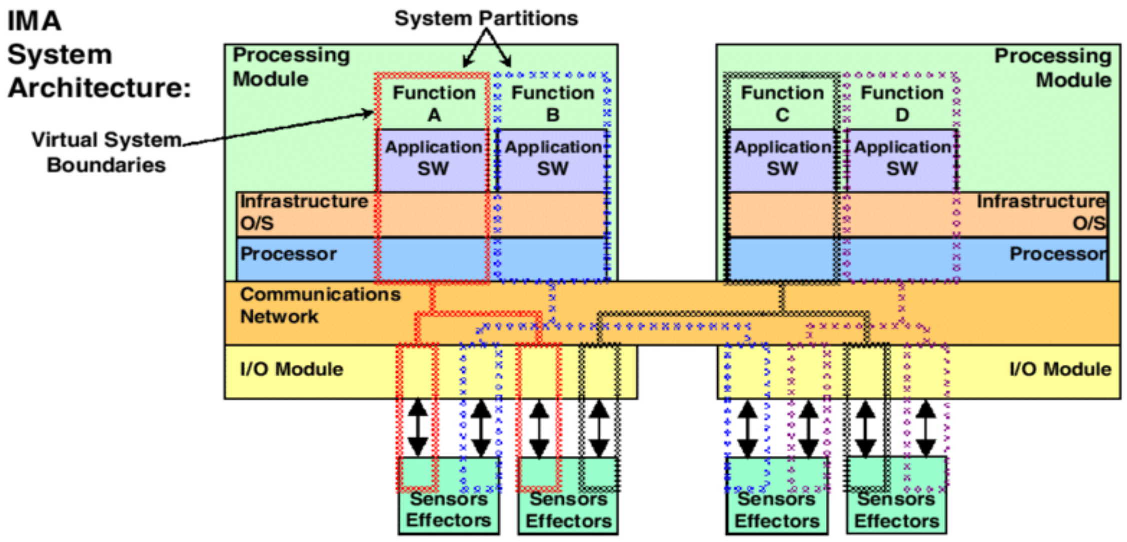


Figure 1: Integrated Modular Architecture

IMA characteristics are:

- **Reduced Size, Weight, and Power**: Resources are shared between multiple applications instead of having their separate but duplicated resources, costs are lower.
- **Competitiveness**: IMA separated applications from the hardware. Before IMA, vendors supplied both hardware and software together. With IMA, hardware can be bought alone and software can be developed according to the requirements.
- **Portability and Reuse**: Standardized modules can be reused thanks to flexibility provided by IMA.
- **Easier Modification**: Since applications are moved from separate modules to central shared processing unit, it is easier to modify the application compared to asking modifications from the vendors.

Although there are clear advantages of Integrated Modular Architecture, integration of different modules are harder compared to Federated Architecture.

2.1.2 ARINC 653

ARINC 653 assumes a set of time-critical and safety-critical real time applications that may be executed on a single processor. ARINC 653 defines the general structure of the underlying RTOS (see Figure 2).

ARINC 653 also provides Application/Executive interface (APEX). APEX is defined as a set of software interfaces that an ARINC 653 compliant operating system must provide to avionics application developers. APEX specifies how to create platform independent software that fulfills ARINC 653 requirements. Avionic applications communicate with their environment through APEX interface. Main components of this interface are:

Partition Management Module: Provides means for modifying the operation mode of partitions.

Process Management Module: Each partition can have multiple processes which can be periodic or aperiodic. ARINC 653 partitions are analogous to Windows/Unix processes and ARINC 653 processes are analogous to Windows/Unix threads. Process Management Module provides process scheduling. Each process can be in Waiting, Ready or Running State.

Time Manager Module: Time is unique and independent of partition execution. This module provides services like reading time, wait/timeout services for processes, increasing time budget of a process with a hard real time deadline. Together with process management module, they guarantee timely execution of processes in a partition.

Partitioning Mechanisms: A partition creates a kind of container for application that provides spatial and temporal partitioning.

- *Spatial Partitioning*: Memory of a partition is always protected. Also, no partition can access memory zones out of their scope. At any given time, only one of the partitions can access to system resources, there is no competition between them.
- *Temporal Partitioning*: This is provided by major and minor time frame mechanisms. Major time frame is time windows where each partition is executed at least once. Minor time frames are time windows allocated to each partition. Minor time frames are combined into major time frames.

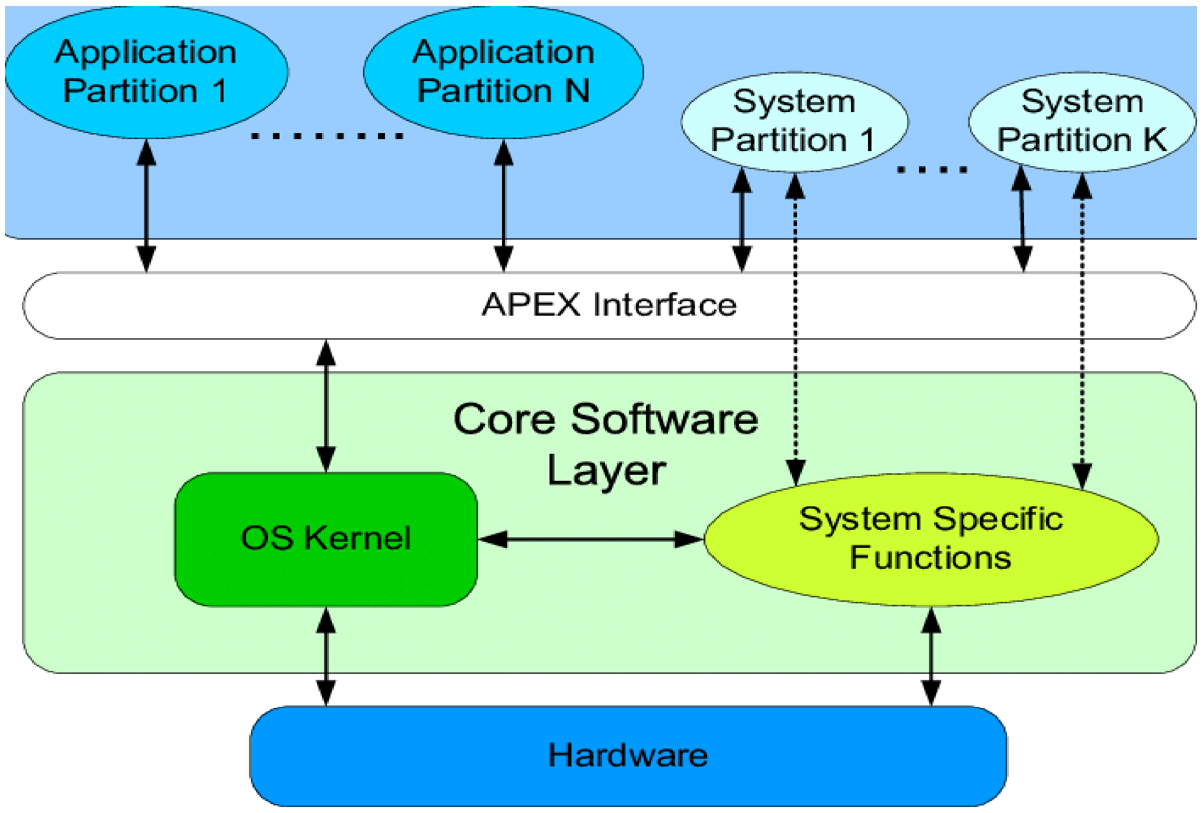


Figure 2: Standard ARINC 653 Architecture1

A partition can be constructed as set of periodic and aperiodic processes. These processes must be executed on a single processor (they cannot be split between processors). But partitions as a whole must be portable between different processors.

All processes have fixed time for execution and priority. Periodic processes of partitions are executed each time a minor time frame starts for a partition. Logically, their time capacity should be shorter than

1 https://medium.com/@mehmetcagrikose/a-simple-introduction-to-arinc-653-775b1c5888c0

minor time frame of the partition they belong to. Aperiodic processes are used to handle rare events and may span over multiple minor time frames.

When the execution window of a partition (minor time frame) terminates, partition is preempted (paused) and next partition in the major time frame starts to execute. Preempted partitions continue their execution in the next minor time frame. For this reason, periodic processes should be shorter than whole partition's minor time frame.

Communication Mechanisms: Communications between partitions are handled through ports and channels. Ports are gates at the boundaries of partitions and channels virtually connect ports to each other. In a sense, partitions do not know other partitions, they only know about ports (see Figure 3).

Ports can be in sampling or queuing mode. Sampling mode implies aperiodic message transfer like altitude data sent by navigation related partition. Queuing mode is used for aperiodic message transfer, like button pushes.

Communication between processes inside a partition is realized through buffers, blackboards. Synchronization between processes inside a partition on the other hand is realized through events and semaphores.

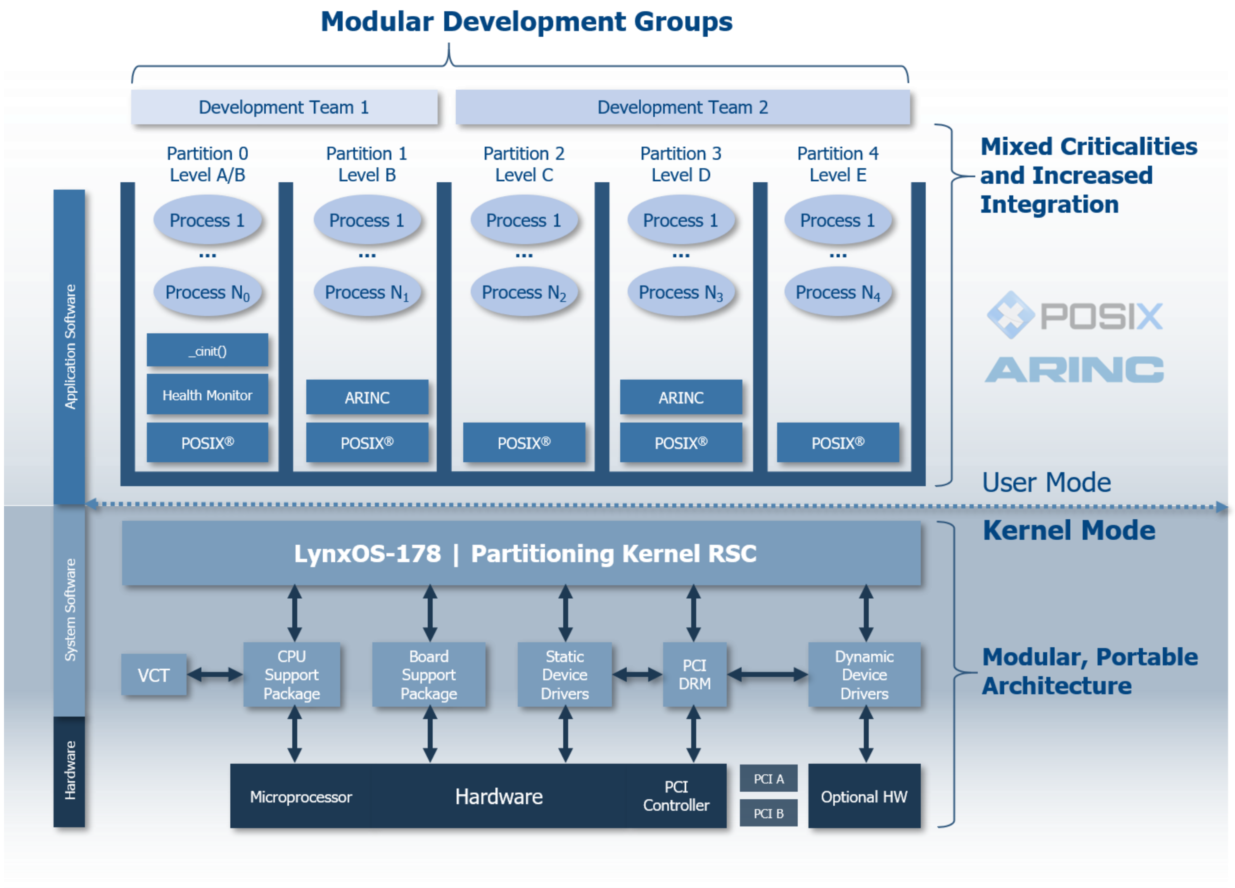


Figure 3: Detailed View of ARINC 653

2.1.3 Mantis Avionics

The avionics architecture of the MANTIS system is designed to be modular and open. To do this, Indra designed a motherboard that includes the autopilot and the power supply:

- The autopilot was designed to have as many communication ports as possible with as many communications protocols used in the UAV market.

- The power supply was designed to have as many different voltage and amperage outputs as possible within the values most used by COTS teams, from the mini-UAVs sector, available on the market.

This modularity and variety in the types of communications and power, provides two great advantages:

- It provides great versatility when adapting to the needs of the customer in relation to the frequency of use of the radio modems and sensors necessary to perform the mission.
- It allows to adapt quickly both to the evolutions of the sensors installed by obsolescence, and to the new sensors that may arise on the market.

Figure 4 shows the current avionics architecture of the MANTIS system:

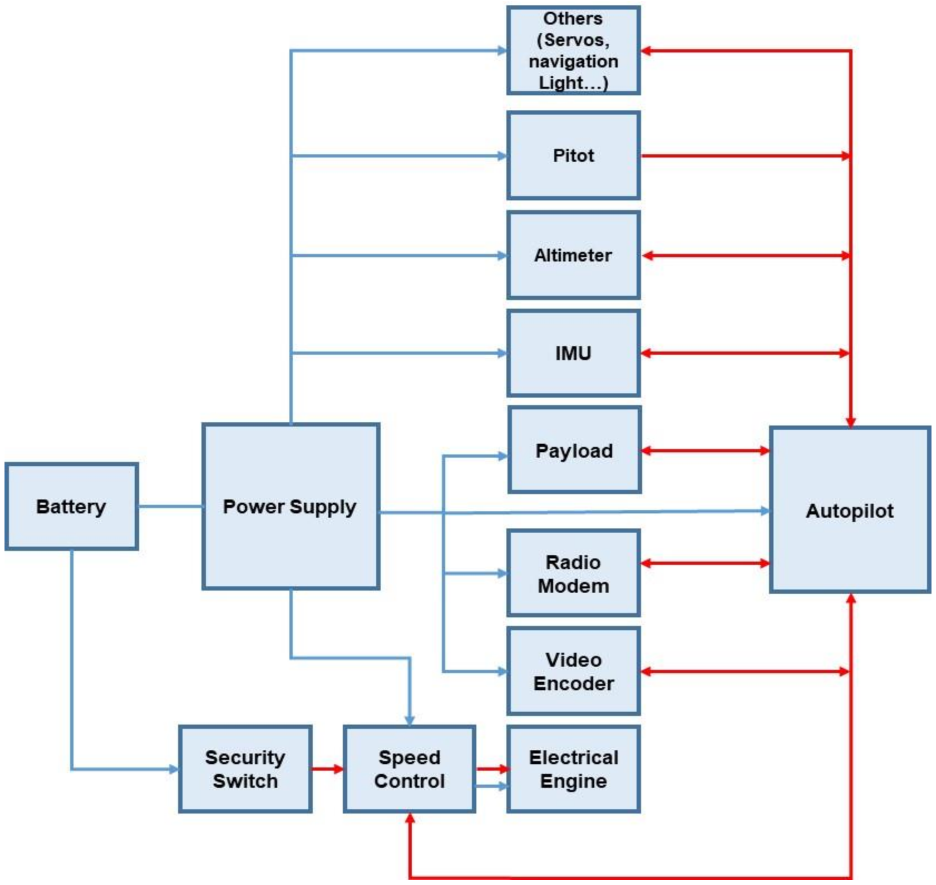


Figure 4: Avionics architecture of the MANTIS system

The Mantis avionics pack is a collection of the following electronic parts:

- Autopilot.
- Inertial measurement unit (IMU) with GPS, inertial, magnetic, and barometric navigation sensors.
- Video encoder: Video processing unit for stabilization, compression and video-tracking.
- Radio Modem: Electro-Optics (EO) and InfraRed (IR) payload video-link transmitter and telemetry/control transceiver.
- Speed Control and Electrical Engine: Engine controller for three phase control of the electrical motor.
- Security Switch with removable security pin-flag for disabling motor running when inserted.
- Power Supply for selective powering all the electronics including payload, external sensors (pitot and altimeter) and other passive electronic components (servos and navigation light).

2.2 Automotive Domain

AUTomotive Open System ARchitecture (AUTOSAR) is a development partnership founded in 2003 by German car manufacturers (OEM) and suppliers. Today, the consortium has grown considerably to 9 core partners and a large number of premium and associate partners, including world-wide OEMs, suppliers and tool vendors. AUTOSAR has become the predominant standard for architectures in the automotive domain.

Its objective is the creation of standardized integrated E/E architecture by fostering reuse of SW modules between OEMs and suppliers. Figure 5 shows the envisioned exchangeability between modules.

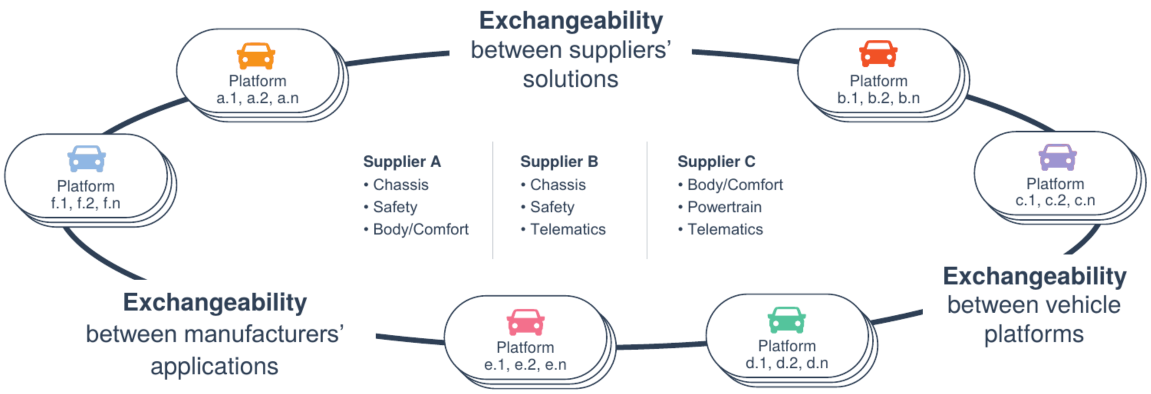


Figure 5: AUTOSAR reuse / exchangeability [source: AUTOSAR introduction, 2018]

The AUTOSAR standard defines an architecture that separates application software from infrastructure related basic software, as depicted in Figure 6. The functional contents of the application software are different and related to the brand identity and the desired characteristics of the car manufacturer, or its system suppliers, whereas the functionality of the so-called basic software is not visible to the customer and is standardized by AUTOSAR.

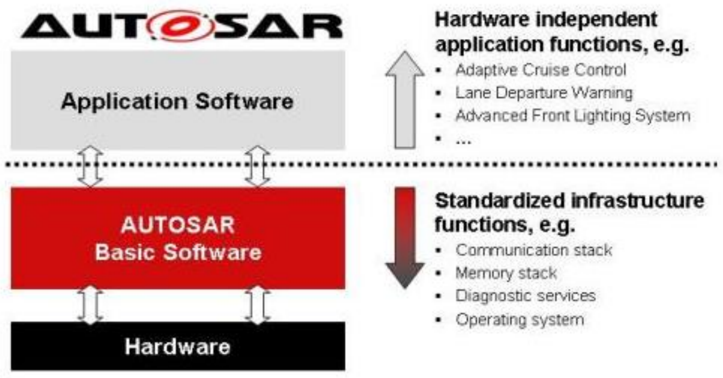


Figure 6: Application software separated from infrastructure functions

The AUTOSAR standard defines exchange formats and description templates to enable a seamless configuration process of the basic software stack and the integration of application software in engine control units (ECUs). The interfaces of typical automotive applications from different domains are defined in terms of syntax and semantics, which should serve as a standard for application software. These formats (often in XML) are read by tools coming with AUTOSAR. These typically include a methodology how to use this framework.

AUTOSAR comes in two variants (that will eventually merge in the future), introduced in the following sections.

2.2.1 AUTOSAR Classic Platform

The classic platform is intended for systems with strict resource and timing constraints. The platform distinguishes three main software layers which run on a Microcontroller: Application Software, Runtime Environment (RTE), and Basic Software (BSW), as shown in Figure 7. The Basic Software layers are further divided into functional groups, for example System, Memory and Communication Services. The micro-controller abstraction is typically provided by the OSEK real-time operating system.

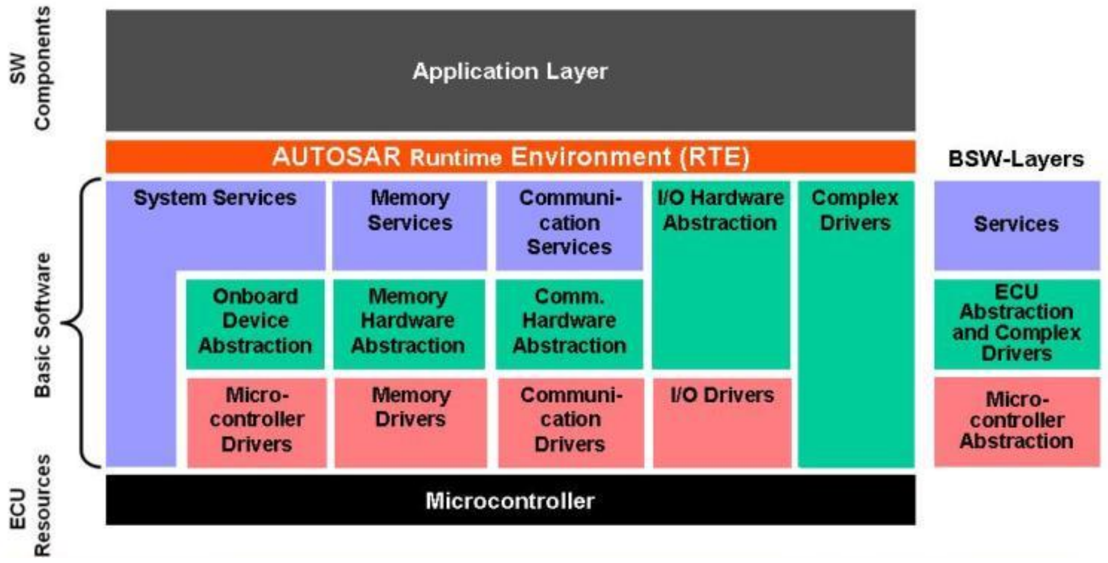


Figure 7: AUTOSAR Layered Software Architecture (Classic)

2.2.2 AUTOSAR Adaptive Platform

The adaptive platform supports (sub-) systems with less strict resource requirements, but the need for more flexibility. This platform runs on operating systems supporting the PSE-51 subset of the POSIX standard. This opens AUTOSAR to a wider choice of (free) operating systems compared to the classic platform.

The Adaptive Platform implements the AUTOSAR Runtime for Adaptive Applications (ARA). Two types of interfaces are available, services and APIs. The platform consists of functional clusters which are grouped in services and the Adaptive AUTOSAR Basis. Figure 8 shows the architecture of the adaptive platform. The platform has range of platform services include a wide range of functionality, such as persistency, health management, logging and tracing and synchronization. State-management enables state-specific behavior. Support for automated driving is available via standardized interfaces.

Communication in the adaptive platform is based on ara::com, it can include SOME-IP enabling a link with the classic platform or Data Distribution Service (DDS) which is an Object Management Group (OMG) standardized communication middleware also used by ROS2 as underlying communication mechanism. The architecture of the adaptive platform is quite generic, it´s rather a tool-box with various services than a concrete software architecture.

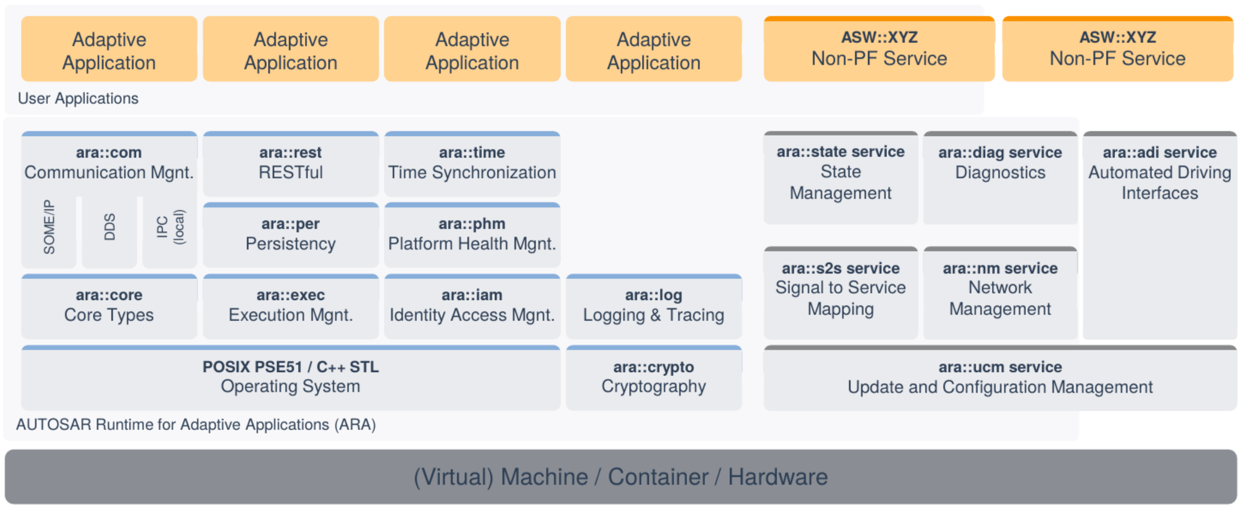
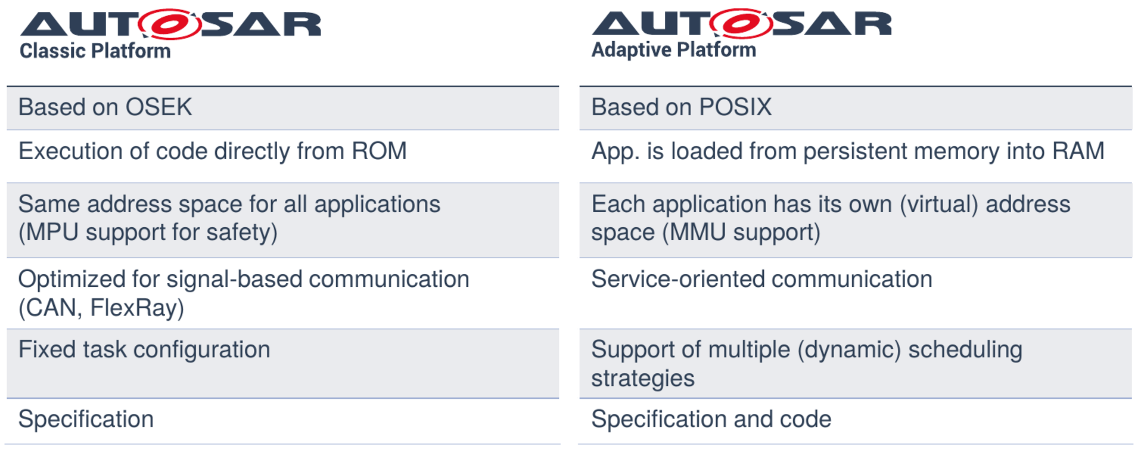

Adaptive Application
Adaptive Application
Adaptive Application
Adaptive Application
ASW::XYZ Non-PF Service
ASW::XYZ Non-PF Service
User Applications
ara::com Communication Mgnt.
SOME/IP
DDS
IPC (local)
ara::rest RESTful
ara::time Time Synchronization
ara::per Persistency
ara::phm Platform Health Mgnt.
ara::state service State Management
ara::diag service Diagnostics
ara::adi service Automated Driving Interfaces
ara::s2s service Signal to Service Mapping
ara::nm service Network Management
ara::core Core Types
ara::exec Execution Mgnt.
ara::iam Identity Access Mgnt.
ara::log Logging & Tracing
POSIX PSE51 / C++ STL Operating System
ara::crypto Cryptography
ara::ucm service Update and Configuration Management
AUTOSAR Runtime for Adaptive Applications (ARA)
(Virtual) Machine / Container / Hardware

Figure 8: Adaptive platform architecture

Figure 9 compares the classic and adaptive platform. The main difference is that the adaptive platform is more open and flexible. It also comes with a reference implementation of some services. The price for this flexibility is increased resource requirements, caused for instance by the need to parse manifests at runtime or using the C++ standard template library.



AUTOSAR Classic Platform	AUTOSAR Adaptive Platform
Based on OSEK	Based on POSIX
Execution of code directly from ROM	App. is loaded from persistent memory into RAM
Same address space for all applications (MPU support for safety)	Each application has its own (virtual) address space (MMU support)
Optimized for signal-based communication (CAN, FlexRay)	Service-oriented communication
Fixed task configuration	Support of multiple (dynamic) scheduling strategies
Specification	Specification and code

Figure 9: Comparison between classic and adaptive platform

2.2.3 Infineon automotive platform for functional safety (FuSa)

This section summarizes the state of the art with respect to a widely used automotive µController platform by Infineon with special features for functional safety and security. Particularly in combination with additional power driver hardware, high level of functional safety can be achieved as described in the following paragraphs[2].

Given the critical importance of safety in the automotive world, it should come as no surprise that the processes and steps to be taken are very carefully defined in internationally-recognized standards. The 'master' safety standard is IEC61508 that deals with safety in many industrial applications. ISO26262 is a sector-specific extension of this standard that deals with the functional safety of electrical, electronic and electromechanical systems within vehicles.

[2]Whitepaper "Safety design for modern vehicles – including EV/HEV", see https://www.infineon.com/dgdl/Infineon-Safety%20Design%20for%20Modern%20Vehicles%20Whitepaper-Whitepaper-v01_00-EN.pdf

Functional safety (FuSa) is defined by the standard as “the absence of an unreasonable risk”. These risks could be caused by a malfunction or failure of all, or part, of an electronic system within the vehicle. FuSa is a ‘whole-life’ approach and deals with random hardware issues from concept through development, production, repair and ultimately decommissioning of the vehicle. FuSa is not a measure of reliability; systems can fail, provided that they do so safely.

The Infineon AURIX™ microcontroller family is specifically designed for use in such safety-critical applications, and mostly in the automotive domain is often used as hardware basis for AUTOSAR standard applications. The block diagram shown in Figure 10 gives a high-level overview of the included hardware blocks. The AURIX safety concept has the following three principal cornerstones:

a) The AURIX hardware is designed for FuSa, with ISO26262 forming an integral part of Infineon’s standardized development process. Key elements include single point fault detection with a lockstep CPU, Error-Correcting Code (ECC) / Error-Detecting Code (EDC) on the memory components and buses, and redundant peripheral devices. Also, latent fault detection with both hardware built-in self-tests (BISTs) and software based self-tests is included. Common cause failure mitigation includes clock and voltage monitors, layout diversity, functional diversity and multiple watchdogs.

b) One of the key metrics for a FuSa system is the time taken to return to a safe state after a fault occurs and is detected. This period, known as the fault tolerant time interval (FTTI) is the sum of three elements: the fault detection time, the fault reaction time, and the time for a safe state to be established. The worst case for the fault detection time is application specific and defined by the software diagnostic time interval. The hardware safety mechanisms within AURIX hardware provide for a very fast fault detection time which is significantly less than 1µs when operating at 100 MHz.

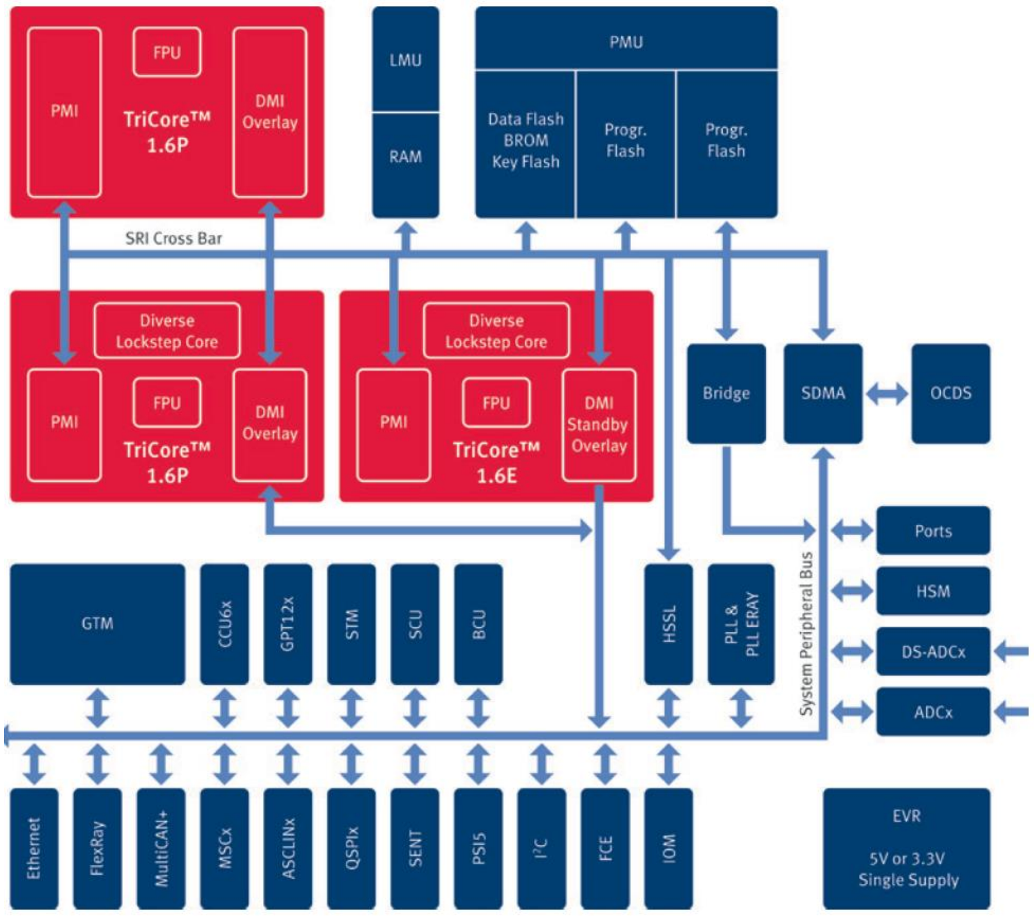


Figure 10: Block diagram of AURIX architecture

The high-power gate driver is another crucial element of the overall safety concept and contains a number of important safety features. A well-known device that fulfils FuSa considerations for the inverter application is the 1EDI2002AS EiceDRIVER™ from Infineon. This insulated-gate bipolar transistor (IGBT) driver provides on-chip basic galvanic isolation and ensures faithful driving of the power module due to its low propagation delay and negligible Pulse-width modulation (PWM) distortion. The AEC-Q100 qualified device contains several safety relevant features including gate signal monitoring which checks the IGBT gate voltage and signal waveform during the turn-on or turn-off sequence, current sense protection and support for active short circuit detection and strategies implementation such as output stage disabling to prevent cross current.

c) Perhaps one of the most critical components in terms of safety is the TLE35584 safety power supply as supplement to the AURIX microcontroller – as depicted in Figure 11. One of the primary roles of this device is to monitor the voltage supplies of the system, internally generated by the device or from other on-board regulators, and, if necessary, disconnect the microcontroller from the power supply to avoid a violation of the safety goals. It is also capable of detecting dependent failures that affect both the function as well as the diagnostic (such as a watchdog error). Should this happen, then the safety power supply can initiate a return to a safe state by driving output pins to disconnect the power feed to the actuators and / or triggering a reset of the microcontroller.

In summary, the example reference design describes, how systems with functional safety requirements are built today in automotive applications. Such concepts also should be considered for future drone architectures, since generally functional safety is also highly important in the aviation domain.

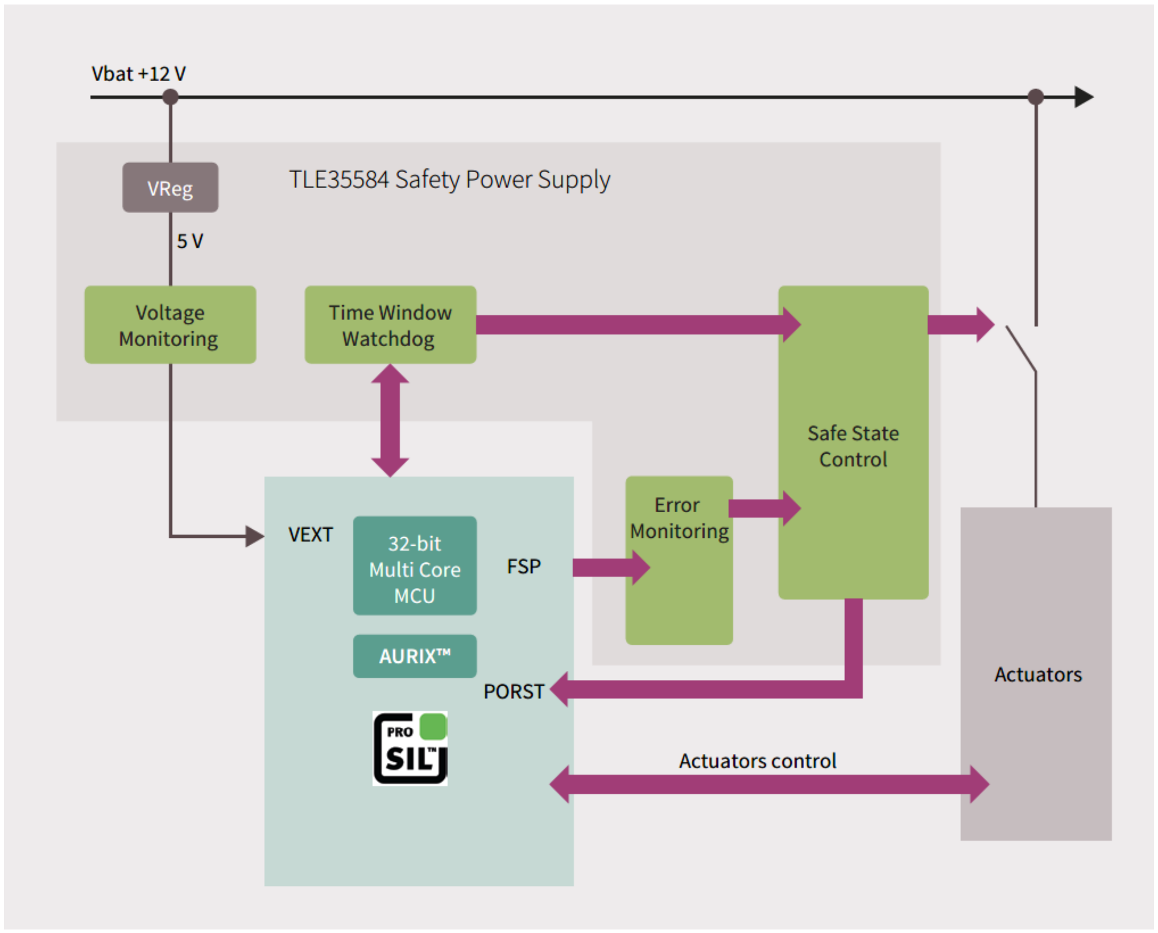


Figure 11: Functional safety system based on AURIX in combination with the safety power supply

2.2.4 Conclusion

The AUTOSAR specifications comes in two flavors targeting on the one hand resource constraint systems, on the other more powerful systems for more flexible functions, including automated driving. The two platforms can co-exist within a vehicle, using the classic platform for the rather static car-body functions (such head-light control) and more flexible ones in infotainment and advanced driver assistance. Some of its functionality can not only be found in automotive applications, but would make sense in other applications as well. For instance, state-management and platform health management are suitable building blocks for a drone architecture and can also be found in other platforms, notably ROS2.

However, it is not a specific architecture but rather a toolbox and infrastructure. Support for creating a complete architecture is out of the AUTOSAR´s scope – unlike as in modeling languages such as EAST-ADL. This means that some of the AUTOSAR services might be interesting for COMP4DRONES without necessarily using the concrete reference implementation. But as AUTOSAR lacks higher level modeling concepts, it has little to offer for COMP4DRONES with respect to modeling.

2.3 Robotics Domain

Autonomous robots are complex systems that require the interaction between many heterogeneous components (software and hardware) in order to properly operate. Due to the increasing complexity of modern robotics applications in terms of tasks and number of hardware components, writing software for robots might be challenging. Also, different types of robots can have widely varying hardware, making code reuse nontrivial. On top of this, the size of the required code can be complicated, as it must contain a deep stack starting from driver-level software up to perception, abstract reasoning, and beyond. Since the required breadth of expertise can be well beyond the capabilities of any single researcher, the development workflow and corresponding environments for robotic applications should also support large-scale projects distributed among several teams along with the corresponding integration activities[3].

To meet these challenges, a wide variety of software frameworks have been developed through years (e.g., GenoM[4], YARP[5], ROS[6], Player[7]). They allow to manage the complexity and heterogeneity of the hardware and applications. For example, YARP (Yet Another Robot Platform) allows to control humanoid robots, ROS and Player are suitable for the development of shared libraries for robotic applications, GenoM allows to design real-time software components. Indeed, the middleware hides low-level communication details to the developers being an abstraction layer that resides between the operating system and the application layer (see in Figure 12)[8].

When developing software for robotic applications, one main focus is the design and the implementation of systems that have to be reusable, scalable and maintainable. In this way software modules can be adapted to new platforms with different requirements, not only reducing the cost and time-to-market of a complete robotic system but also improving its overall performance. In this context, the tasks carried out by robots are arranged as distributed processes, which communicate through the middleware.

Most of these robotics' middleware are based on the publish/subscribe model. In this model, data providers publish data and consumers subscribe to the information they need, receiving updates from

[3] T. Sobh and A. Elkady, "Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography," Journal of Robotics, pp. 1-15, 2012
[4]S. Fleury, M. Herrb and R. Chatila, "Genom: A tool for the specification and the implementation of operating modules in a distributed robot architecture," in IEEE International Conference on Intelligent Robots and Systems, 1997
[5]G. M. P. Fitzpatrick and L. Natale:, "YARP: Yet another robot platform," International Journal on Advanced Robotics Systems, vol. 3, no. 1, pp. 43-48, 2006.
[6]M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, "ROS: an open-source Robot Operating System," in Proceedings ICRA Workshop Open Source Software, 2009
[7]T. H. Collett, B. A. MacDonald and B. P. Gerkey, "Player 2.0: toward a practical robot programming framework," in Australasian Conference on Robotics and Automation, 2005
[8] G. Bradski, "Development environments for autonomous mobile robots: A survey," Dr. Dobbs Journal, 2000.

the providers as soon as information has been published. This model is based on the notion of event. Components interested in some class of events "subscribe" expressing their interest. Components providing information "publish" it to the rest of the system as event notification. Then, producers and consumers are decoupled since publishers and subscribers do not know each other: publish/subscribe operations, and in particular the delivery of an event notification to all the subscribers, are mediated by a component, the event dispatcher, whose structure can be either centralized or distributed (see Figure 13)[9].

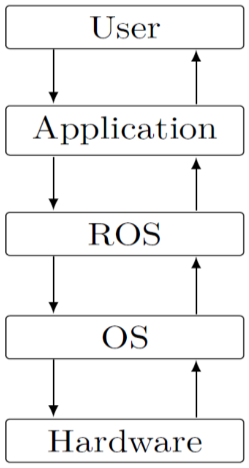


Figure 12: Software abstraction levels from the user to the hardware layers. Here ROS has been used as an example of robotic middleware.

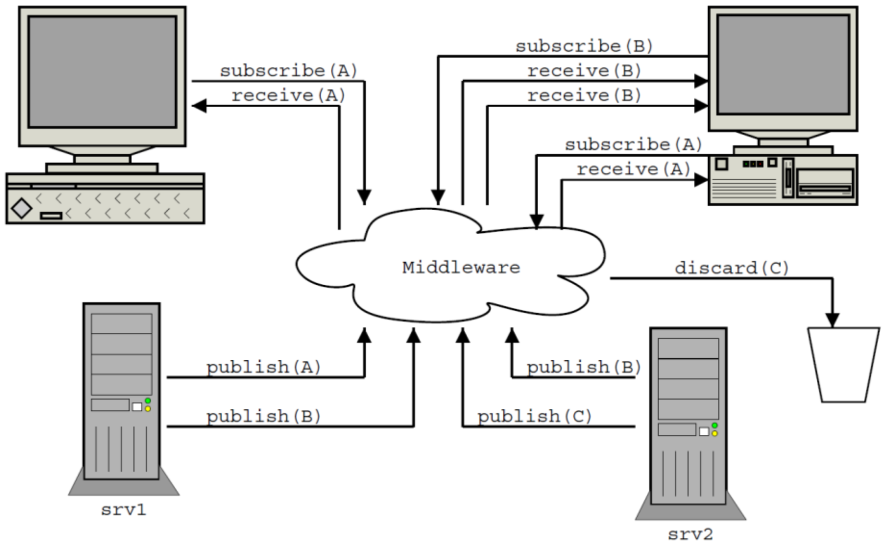


Figure 13: Publish/Subscribe model.

The publish/subscribe model is convenient whenever a data transfer can be triggered only in case of updates. Alternatively, there would be a stream of constant data which conflicts with the typical requirements of robotic applications.

A very popular robotic middleware is ROS[10], the Robot Operating System, which is widely used for space robot challenges, autonomous driving, industrial assembly, and surgery. ROS fully supports the

[9] M. Matteucci, "Publish/Subscribe Middleware for Robotics: Requirements and State of the Art," http://home.deib.polimi.it/matteucc/Download/MOMRob.pdf, 2013
[10] L. Zhang, R. Merrifield, A. Deguet and G. Z. Yang, "Powering the world's robots-10 years of ROS," Science Robotics, vol. 2, no. 11, 2017

publish/subscribe paradigm, offering a great flexibility also promoted by code and contributions sharing from worldwide researchers.

2.3.1 Robotics Middleware

In this section, some among the most used middleware are presented. Since a complete survey is impractical due to the large number of solutions either already existing or brand new, we restrict the range to ROS, YARP and Player.

ROS, Robot Operating System: ROS is an open source robotics middleware that was initially developed 13 years ago on the basis of some work carried out at the Stanford Artificial Intelligence Laboratory and additional efforts by the Willow Garage[11].

ROS includes drivers for a wide range of sensors, simulators, and algorithms for navigation tasks. Support for different hardware manipulators has also been pursued in parallel, with recent efforts on standardizing the application program interfaces via ROS Industrial (ROS-I) and motion-planning tools such as MoveIt.

ROS also implements a distributed paradigm, where many processes can be run in parallel across multiple machines. In addition, accurate physical simulation (achieved by using external tools interfaced to ROS, e.g., Gazebo[12], V-REP[13]) can be used to test different algorithms, including deep learning, without any further hardware investment. ROS is compatible with different robot simulators, which is essential for prototyping and validating in different scenarios the specific robotic application under development and also training them when based on neural networks. A largely adopted robot simulator, both academia and industry is Gazebo. It is standalone, i.e., it does not require a robotic middleware for being executed. However, Gazebo can be easily integrated with ROS by simplifying its use.

The impact of ROS on academic research is evident from the steady increase of publications that benefited from the software platform. These studies include but are not limited to space robotics, service robots, medical robots, humanoid robots, robotic manipulators and grippers, unmanned aerial vehicles (UAV) and autonomous underwater vehicles (AUV). ROS-I provides interfaces for common industrial robots and sensory devices along with software libraries specific to the manufacturing automation. A growing number of industrial hardware, such as robots produced by ABB, Fanuc, and Yaskawa, has been supported by ROS-I. The recent ROS-II release will address one of the major limitations that has slowed down ROS adoption in the industry, i.e., it will rely on data distribution service for inter-process communication, which brings much better reliability with quality-of-service protocols and security with encryption.

YARP: YARP has been designed in order to cope with computational burden and complex tasks. Indeed, achieving visual, auditory, and tactile perception while performing elaborate motor control in real-time requires a lot of processor cycles.

YARP uses a communication mechanism which tries to make writing and running code as easy as possible. The code is maintained and reused through its simple organization in small processes, each one performing a simple task. With YARP it is easy to write processes that are location independent and that can run on different machines without code changes. This allows moving processes across the cluster at runtime to redistribute the computational load on the CPUs or to recover from a hardware failure. YARP does not contain any means of automatically allocating processes. The task assignment is deliberately left to the developer. In this way, a sort of "soft real-time" parallel computation cluster without the more demanding requirements of a real-time operating system can be obtained.

[11] Willow Garage is a robotics research lab and technology incubator devoted to developing hardware and open source software for personal robotics applications.
[12] C. E. Agüero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J. L. Rivero, J. Manzo and E. Krotkov, "Inside the virtual robotics challenge: Simulating real-time robotic disaster response," IEEE Transactions on Automation Science and Engineering, vol. 12, pp. 494-506, 2015
[13]E. Rohmer, S. P. N. Singh and M. Freese, "V-REP: a Versatile and Scalable Robot Simulation Framework," in IEEE International Conference on Intelligent Robots and Systems, 2013

As for the communication level, YARP minimizes dependencies between processes, so that communication channels between processes can be set-up or can drop without any issue: a process that is killed or dies unexpectedly does not affect other process that was communicating with that. This also simplifies cooperation between people, as it minimizes the need for synchronization development on different parts of the system.

A frequent problem encountered during development in robotics is that it is very hard to reuse code on different platforms. For example, two mechanically similar platforms may have different electronics. Another problem occurs when two identical boards are used on setups that are mechanically different. In YARP, the whole of these routines is grouped in an adapter class. This class is, in general, responsible for implementing methods to correctly initialize and shut down the device, but it can implement other functionalities as well. As such it collects all and the only routines specific to each hardware device[14].

Today YARP is a platform for long-term software development for applications that are real-time, computationally intensive, and involve interfacing with diverse and changing hardware. It is successfully used on several platforms being highly interoperable, and works in multiple operating systems including Linux, Windows and OSX[15]. Nowadays, YARP enables the design of pure Peer-to-Peer (P2P) frameworks[16] with a partially implemented discovery service. YARP offers a good naming-service mechanism that allows the definition of namespaces (pushing), and provides good programming support, allowing C++, Java, Python, and Matlab.

Player: The Player[17] project is designed to provide an infrastructure, drivers and a collection of dynamically loaded device-shared libraries for robotic applications. It is one of the first middleware that emerged for robotic systems and is wrapped by others. It does not consider a robot as a unity, but it instead treats each device separately, being a repository server for actuators and sensors.

Player is based on a three-layer architecture in which the top layer is represented by clients that are specialized software components. The middle layer provides common interfaces for different robot devices and services. The last layer concerns the hardware, i.e., the robots, sensors, and actuators. Player refers to the device and server interface. The devices are made of a driver and an interface which are independent of each other. They can subscribe to a Player server repository to become accessible to clients. Clients can connect to the repository to request data from the sensors, send commands to the actuators, or perform configuration changes to an existing device.

The connections between the clients and the devices are set-up into separate connection-less sockets[18], making the data transfer available for multiple concurrent clients and leaving the control architecture for the client to deal with. The components of the device that allow the client to retrieve information and send control commands to the devices are the device interfaces. These interfaces communicate with the device drivers that process the information. The socket communication implies that the clients' software can be written in any programming language that has socket support. C, C++, Java, Common Lisp, TCL and Python are supported client programming languages. Other programming languages can access the interface provided by Player using various client-side libraries.

[14]G. Metta, P. Fitzpatrick and L. Natale, "YARP: Yet Another Robot Platform," International Journal of Advanced Robotic Systems, vol. 3, no. 1, pp. 43-48, 2006
[15]L. Natale, "Linking action to perception in a humanoid robot: a developmental approach to grasping," DIST. Genova, 2004
[16]P. Iñigo-Blasco, F. Diaz-del-Rio, M. C. Romero-Ternero, D. Cagigas-Muñiz and S. Vicente-Diaz, "Robotics software frameworks for multi-agent robotic systems development," Robotics and Autonomous Systems, vol. 60, pp. 803-821, 2012
[17]M. Kranz, R. B. Rusu, A. Maldonado, M. Beetz and A. Schmidt, "A Player/Stage System for Context-Aware Intelligent Environments," in Annual Conference on Ubiquitous Computing, 2006
[18] According to this model, communication processes do not necessarily have to establish a connection before exchanging messages. Instead, the sender specifies a destination address in each message. There is no guarantee that the recipient will be ready to receive the message and no error will be returned if the message cannot be delivered.

2.3.1.1 Main Features of a Robotics Middleware

As described above, the development of software for robotic systems is challenging because they are required to solve problems with several hardware constraints. Moreover, the need of autonomy and intelligence entailed complex software systems and the increase of computational needs.

Software modules, scalability, reusability, efficiency and fault tolerance play a fundamental role. They have to be designed in a distributed and modular way. The main features to which the software modules must satisfy are listed below[19]:

Concurrent and distributed. It is necessary to be able to take advantage of all the processing units available in a concurrent way (processors, multi-processors and microcontrollers) in order to cover all the computational needs that a complex robotic system presents. Due to the consequent system complexity, a powerful remote inspection (also known as introspection) mechanism is needed.

Modularity. The software is formed of several components of high cohesion and low coupling. The components interact with each other. However, the dependences must be kept at a minimum in order to obtain maintainable, scalable, and reusable software, which is adaptable to changes and improvements.

Robustness and fault tolerance. The malfunction of a component must not completely block the whole system. On the other hand, the rest of the system must be capable of continuing to work as best it can. To this end, the rest of the components (still in working order) must be capable of acting on their own initiative and autonomously make decisions to overcome these situations.

Real-time and Efficiency. The majority of robotic systems have some type of real-time constraints. These restrictions are problematic when the software is distributed. Efficiency is also a common requirement, especially when a robotic system has limited communication and computation capacities. Hence the design of the system must consider the use of software, hardware, communication mechanisms and protocols that guarantee compliance with these restrictions.

The use of an operating system is not always necessary and lightweight solutions are typically employed (RTKernel[20], FreeRTOS[21], QNX[22], etc.) that are specifically adapted to the hardware and to the devices and peripherals used. Some of the most important characteristics supporting these systems are multi-tasking and real-time restrictions; highly interesting aspects in robotics. They also possess controllers for a group of specific devices that allow them to access typical peripherals such as Ethernet and CAN. However, in order to maintain reusability with complex software libraries (they might not work on lightweight systems), they also support high level Operating Systems and a large quantity and diversity of off-the-shelf libraries and drivers for devices and peripherals. The general-purpose operating systems which are the most highly developed in these aspects are found in the world of PCs: Linux, Windows and OSX are some examples. As a disadvantage, these systems can be considered as heavyweight since they require greater hardware resources. Nevertheless, in many cases, this does not prevent them from being able to support tasks with real-time restrictions.

Finally, the programming language used for development is also an important aspect. The greater the number of supported languages, the more flexibility is required to the development, which in turn results in more libraries and projects that can be reused. The language employed brings major consequences on performance, the real-time capabilities, the transport mechanism, and some agent capacities, such as mobility. The most widely accepted language in robotic application frameworks is C/C++ since it offers a good balance between the access to devices, sensors and actuators at low level, while still offering a sufficient level of abstraction to create complex distributed software systems. This means that it presents an adequate language for both low-level control tasks and high-level programming. On the

[19]A. Farinelli, G. Grisetti and L. Iocchi, "Design and implementation of modular software for programming mobile robots," International Journal of Advanced Robotic, vol. 3, pp. 037-042, 200
[20]http://www.on-time.com/rtkernel-dos.htm
[21]https://www.freertos.org/
[22]https://blackberry.qnx.com/en/products/neutrino-rtos/index

other hand, many algorithms in robotics deal with problems with a high level of abstraction which is why it is also recommended that the system support high-level languages such as Java, Python or MATLAB. Nevertheless, most platforms tend to offer a wider range of languages[23], to prevent programming language restrictions to be an application barrier when deploying a robotics application.

2.3.1.2 Communication Mechanisms

The software for robotic systems is in most of the case peer-to-peer, P2P. In a P2P system, the nodes interact among themselves without the need for a coordinating fixed central node or authority. This model displays a high tolerance to failures since no special node is indispensable. With this approach, the communication performance can be also improved since bottlenecks are avoided. Nevertheless, a pure P2P framework has some drawbacks, such as the increase of internal complexity, and presents certain difficulties when different nodes are put into contact, and when coordinating these nodes. Hence, some extra services are needed, such as a distributed naming service, and a discovery service (see Figure 14).

The nodes interact among themselves by means of message passing. Simple message passing is the basis that allows other types of more advanced mechanisms to be constructed, such as ports, topics, events, services, and properties. These mechanisms allow the attempted communication to stand out and they offer some advantage in the design or implementation: low coupling, performance, ease of use. A common characteristic of these mechanisms with respect to simple message passing is that they are addressable communication channels, for example a service or port of a node needs a name in order for it to be located and referenced. In the case of Hybrid P2P systems, all these communication channels must be registered in the main node that hosts the naming service and lookup service.

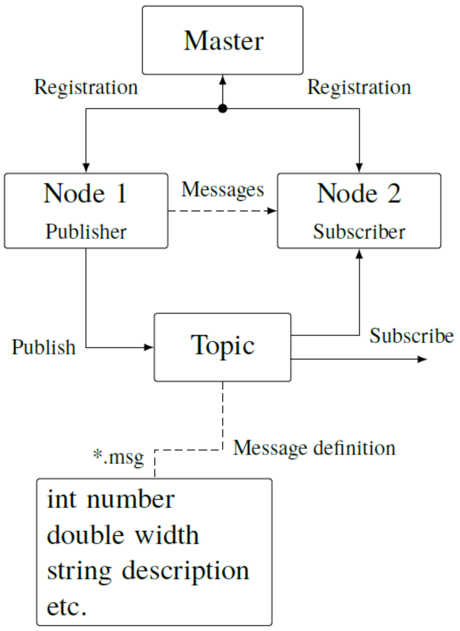


Figure 14: Example of the peer-to-peer communication.

Messages. This is the simplest mechanism. It is a one-to-one communication where a node sends an asynchronous message and another node receives it. The messages also tend to include metadata about the message, such as the intention, content format, and content ontology. It is the most basic communication mechanism, from which all the other mechanisms are derived.

[23] D. Vallejo, J. Albusac, J. Mateos, C. Glez-Morcillo and L. Jimenez, "A modern approach to multiagent development," Journal of Systems and Software, vol. 83, no. 3, pp. 467-484, 2010

Topics. This is an asynchronous communication mechanism that follows the publish/subscribe model and allows a many-to-many communication to be made whilst maintaining low coupling. A topic represents a centralized channel where all the nodes connected to it receive any message that is sent by a node. This channel represents logic centralization; that is to say, it does not imply a bus at the transport level.

Ports. These are mechanisms that allow nodes to communicate with a low degree of coupling. The nodes are made up of two types of ports: in-ports and out-ports. These ports can be interconnected with another port through one or several connections. They can be created and destroyed dynamically, which contributes great flexibility. Each node is designed to read and to write to its ports independently of which node is connected in execution. The messages are usually stored in buffers in the in-ports and out-ports.

Services. This is a communication mechanism that allows the remote execution of a procedure. Two messages come into play: request and response. The message request is sent by the client node and indicates what procedure is desired to be executed and its arguments. The response message is sent by the server node with the result of the operation. It is a typically synchronous procedure where the client remains blocked while the response is awaited. This mechanism is typically used for Robot Hardware Interfaces (RHI).

2.3.1.3 Other Aspects of Robotics Middleware

In addition to previously described aspects, a robotics middleware presents other aspects significant for robotic systems development, some of which are analysed in this section, i.e., developments and deployments tools, simulation capabilities, robotic algorithms and robot hardware interfaces.

Some frameworks provide a set of tools that facilitate robotics software development. Commands for the construction of the system tend to be of a regular type, and can solve problems of dependences on other modules of the framework. On the other hand, some frameworks allow the graphical creation and interconnection of system modules.

Frameworks usually offer infrastructure to facilitate deployment tasks. This infrastructure typically appears in the form of tools and configuration files. Deployment tools are crucial in maintaining the scalability of the distributed and complex robotic framework. The framework of the system is defined and the participating nodes are specified in the configuration files. The configuration files can also define boot and connection parameters and mechanisms of communication between the nodes. The tools can be visual or command-lines and allow the start, stop and administration of system nodes, as well as the establishment of the initial parameters through configuration files, or the conducting of operations related to the naming service, such as renaming or name pushing.

The simulators offer a visual representation of the problem and execution in virtual worlds of rigid solids. The frameworks studied are not focused on simulation capacities; nevertheless, some frameworks have implemented modules that allow a rapid integration with simulators such as Stage, Gazebo, V-REP or AirSim[24]. Simulators play a major role by allowing the system to be modeled in the design stage, by finding errors in the early stages of development, and by saving time and money. They are also useful for testing ideas.

Many frameworks offer reusable generic algorithms in the sphere of robotics: kinematics, robot perception, Bayesian estimation, SLAM, motion planning and machine learning, to name but a few. These algorithms present one of the most important aspects for evaluation, since to some extent it measures the good design of a robotics middleware and means that it has successfully overcome the chronic obstacles to the reuse and generalization of robotics technologies.

[24]S. Shah, D. Dey, A. Lovett and C. Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles," in Field and Service Robotics, 2018

Robotics middleware usually support a set of hardware devices and robots. This is one of the lower-level aspects of the robotics software development. Most existing frameworks tackle this aspect, and commonly use one of the two following approaches for the implementation of the interface:

- The API isolation approach. An application programming interface (API) is designed in a language, such as C++, and Java. The specific code then implements such interface, usually through inheritance. Programmers are encouraged to make high-level algorithms over these interfaces instead of specific hardware code.
- The node isolation approach. This approach is peculiar to P2P frameworks. The device driver code is isolated in a component that can be instantiated as a node at run-time. This node uses the communication mechanisms, i.e., ports, topics, services, etc. The programmer incorporates these mechanisms explicitly in the node code, and hence the coupling point is located in the message data structure.

As in the API isolation approach, the main drawback of this approach is that the use of distributed communication mechanisms may increase the latency of operations and therefore real-time capabilities may be lost.

2.3.2 An Architectural Approach for Robotics

The traditional approach to developing robotics software is fundamentally code-centric and fragmented into numerous, often incompatible, frameworks. Consequently, software engineering of robotics applications is still in an early “craft” stage when compared to other domains such as automotive or avionics. AUTOSAR, for instance, combines the model-driven and component-based approaches to enable agile value chains involving car manufacturers, part suppliers, and IT developers. This is definitely not the case in any of the emerging robotics application domains – at least not yet. There is no standard for software development that would enable the combination and interaction of components independently of any specific software infrastructure. There is also no general characterization of common robotics functionalities, such as “planning”, “sensing” and “control”(in the broader sense of these terms),nor is there a general support framework that addresses the quality aspects of these systems, such as safety or reliability.

During last years the European Commission has funded many basic and industrial research projects to leverage to Europe a framework of key enabling technologies in Robotics domain. One of these projects is RobMoSys[25] that has developed a System Engineering Approach to enable design of safe and efficient robots based on reuse and composability of qualified components. In the context of the RobMoSys project, two complementary development environments (i.e. Papyrus4Robotics and SmartMDSD, described in sub-section 2.3.2.2) have been produced. These environments follow the proposed engineering approach and the standard frameworks in the robotics domain such as RTC and SmartSoft (see sub-sections 2.3.2.3 and 2.3.2.4)

2.3.2.1 RobMoSys Approach

RobMoSys[26] envisions an integration approach built on top of current code-centric platforms by means of systematic development and application of modern model-driven methods and tools. This enables system-of-systems type of integration, which occurs at all levels of abstraction and interaction, rather than merely at the code level. RobMoSys defines modeling constructs (the RobMoSys Composition Structures or Metamodel) that are specific to the robotics domain (from low-level concepts such as communication patterns used in robotics middleware to high-level concepts such as Mission, Tasks, and World Models). At present, there is no modelling language with such a rich set of features specific to robotics that are, nevertheless, generic enough to be applicable to different domains (healthcare robotics, industrial robots, service robots, etc.).

[25] Deliverable D2.6 of RobMoSys (H2020-EU.2.1.1. - INDUSTRIAL LEADERSHIP project under grant agreement number 732410)
[26]https://robmosys.eu/

The RobMoSys approach is organized into three tiers. Tier 1 defines all the robotics meta-structures that are required to consistently model robotic systems throughout several development phases and thereby supporting different developer roles. Tier 2 conforms to these foundations, structuring the particular domains within robotics and is shaped by the experts of these domains, for example, object recognition, manipulation, or SLAM. Tier 3 supplies and use model content. Here are the main "users" of the ecosystem, for example, component suppliers and system builders. The RobMoSys composition structures is a bottom abstraction layer on Tier 1 (see Figure 15).

The meta-structures follow a general composition-oriented approach where systems can be constructed out of reusable building blocks with explicated properties. In other words, RobMoSys enables the composition of robotics applications with managed, assured and maintained system-level properties via model-driven techniques. Figure 15 shows an exemplary list of possible composition structures (highlighted with the yellow background color), which can be clustered into (a) specializations of blocks and (b) specializations of relations. One of the central structures defined by RobMoSys is a consistent and rich enough component model that considers the interaction with related structures around the component model (such as e.g. the definition of communication services and the binding to different middleware).

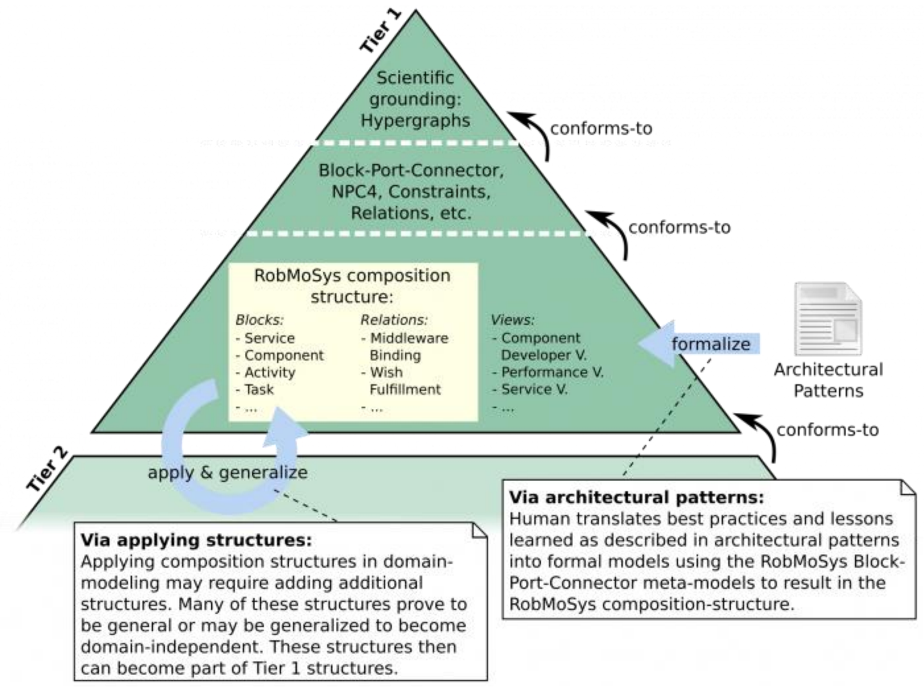

Tier 1
Scientific grounding: Hypergraphs
conforms-to
Block-Port-Connector, NPC4, Constraints, Relations, etc.
conforms-to
RobMoSys composition structure:
Blocks:
- Service
- Component
- Activity
- Task
- ...
Relations:
- Middleware Binding
- Wish Fulfillment
- ...
Views:
- Component Developer V.
- Performance V.
- Service V.
- ...
formalize
Architectural Patterns
conforms-to
Tier 2
apply & generalize
Via applying structures:
Applying composition structures in domain-modeling may require adding additional structures. Many of these structures prove to be general or may be generalized to become domain-independent. These structures then can become part of Tier 1 structures.
Via architectural patterns:
Human translates best practices and lessons learned as described in architectural patterns into formal models using the RobMoSys Block-Port-Connector meta-models to result in the RobMoSys composition-structure.

Figure 15: Tier 1 in RobMoSys (meta-structures)

Figure 16 provides an overview of the RobMoSys composition structures (i.e. the RobMoSys Meta-models). Each block in the figure represents a separate Meta-model. There are high-level relations between the meta-models that are depicted with the uses keyword. Further information about the RobMoSys composition structures can be found in the RobMoSys wiki[27].

In addition to the RobMoSys composition structures (or meta-models), another essential concept is the Architectural Patterns[28]. RobMoSys defines patterns describing recurring design problems. For instance, an architectural pattern for Communication relates to different common robotics software layers: Service structures communication, Execution container as it provides resources for communication, Operating System / Middleware as it realizes communication, Hardware as it physically execute communication. Different middlewares allow for different middleware abstraction levels. For instance, message-based middlewares require a protocol-based abstraction, while DDS allows for a

[27]https://robmosys.eu/wiki/modeling:composition-structures:start
[28]https://robmosys.eu/wiki/general_principles:architectural_patterns:start

higher level of abstraction (i.e. directly using the publish/subscribe communication with accordingly preselected Quality of Service (QoS) attributes). In any case, middleware details should be hidden from both, the component's internal communication access and the communication semantics between components. Not every semantic detail needs to be made explicit on model level (some may come from "de-facto standard" implementations). The focus in models helps to be on a consistent representation and systematic management of different communication schemes.

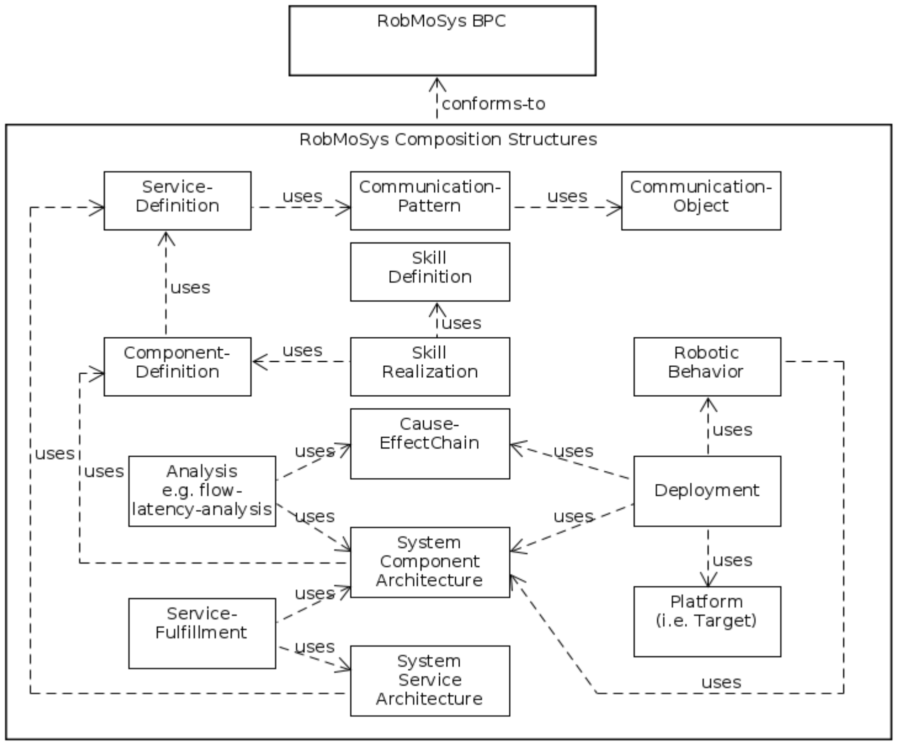


Figure 16: RobMoSys Composition Structures.

2.3.2.2 RobMoSys Development Environments

During RobMoSys project, two complementary development environments have been produced: SmartMDSD and Papyrus4Robotics.

Papyrus4Robotics[29] (developed by CEA) is an open source tool based on Eclipse, to deploy an agile MBSE process due to its usability and wide industrial adoption. Papyrus4Robotics provides a support for UML and SysML modeling. It offers advanced customization capabilities based on profiles and specific tooling artefacts including model explorer, versioning, source control, model concurrent access, code and documentation generation, libraries, user type management and change request. The other benefits of the platform come from the availability of detailed guidelines for plugin construction, built-in support for storing and linking heterogeneous development artifacts and its flexible integration with others tools, e.g. with DOORS for requirement management, OpenCert[30] for system assurance and certification management.

The SmartMDSD Toolchain is an Integrated Software Development Environment (IDE) for system composition in robotics software business ecosystem. It supports the different roles that act around the development of robotics systems to offer building blocks and/or use building blocks to build systems.

Robotics solution providers can use the SmartMDSD Toolchain to develop software components. Robotics System Builders can use the SmartMDSD Toolchain to compose their systems from previously

[29]https://robmosys.eu/wiki/baseline:environment_tools:papyrus4robotics
[30]https://www.eclipse.org/opencert/

developed software components. For both of them, the SmartMDSD Toolchain provides guidance through tooling and enables users to gain benefit from 20 years of best-practices of robotics component and system development. The SmartMDSD Toolchain is based on the Eclipse IDE and model-driven software development techniques (using Xtext, Xtend and Sirius).

2.3.2.3 RT Component Framework[31]

To modularize the elements which are constructing a robot system, there are many types of granularity of the modularization. For example, a single function device like a motor/sensor, a compound function device like a moving cart robot, robot arm and a set of algorithms which handle various processes are considerable. A system would be built up by hierarchical integration of them. In RT middleware, the software part of the fundamental functional elements is called as "Core logic". RT component framework is a mechanism for covering the Core Logic by common interface and makes the user be able to use the modules uniformly (see Figure 17).

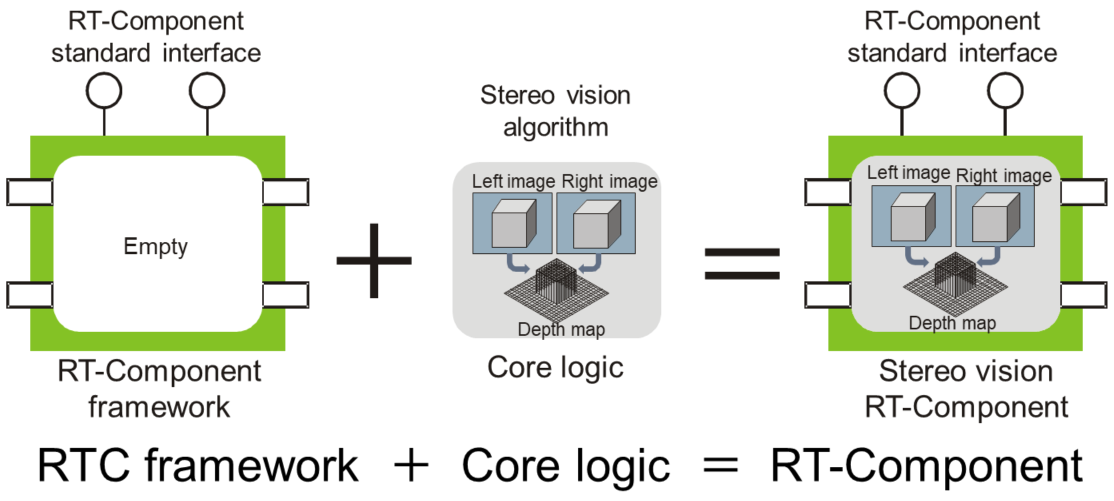

RT-Component standard interface
Empty
RT-Component framework
Stereo vision algorithm
Left image
Right image
Depth map
Core logic
RT-Component standard interface
Left image
Right image
Depth map
Stereo vision RT-Component
RTC framework + Core logic = RT-Component

Figure 17: RT Component Framework and Core Logic

Consider an example of componentized stereo vision algorithm as shown in Figure 17. The program itself which implements the algorithm corresponds to the Core logic. Stereo vision component can be made by preparing RT component framework with appropriate Port's and implementing a stereo vision algorithm into the framework. The module in which Core logic is implemented based on RT component framework is called as "RT component". RT component framework hides the implementation of the common interface from the component developer and the integrator who make a system by combining the components. As a result, the component developer can focus on the implementation of the main logic and the integrator can focus on the whole system design without worrying about the implementation details of the component.

RT Component architecture: In RT system, various levels of processing should be handled with considering the cooperation of them from low-level sensor process and actuator control to high-level Recognition, Judgment, Controlling the behavior, etc. In low-level controlling program might require the speed and the real-time capability. However, the high-level program might require computer language which has high abstraction capability and high descriptive power. Also, the use of multiple CPU system is currently increasing in RT system world, and it is necessary to have parallel control and cooperation function via the network.

To modularize these functional elements, the RT component provides a framework based on a distributed component technology that can be modularized with various granularity and operate on

[31] https://openrtm.org/openrtm/en/doc/aboutopenrtm/rtc_architecture

various languages and OS. Figure 18 shows the basic architecture of RTC. The main functions of RTC are as follows.

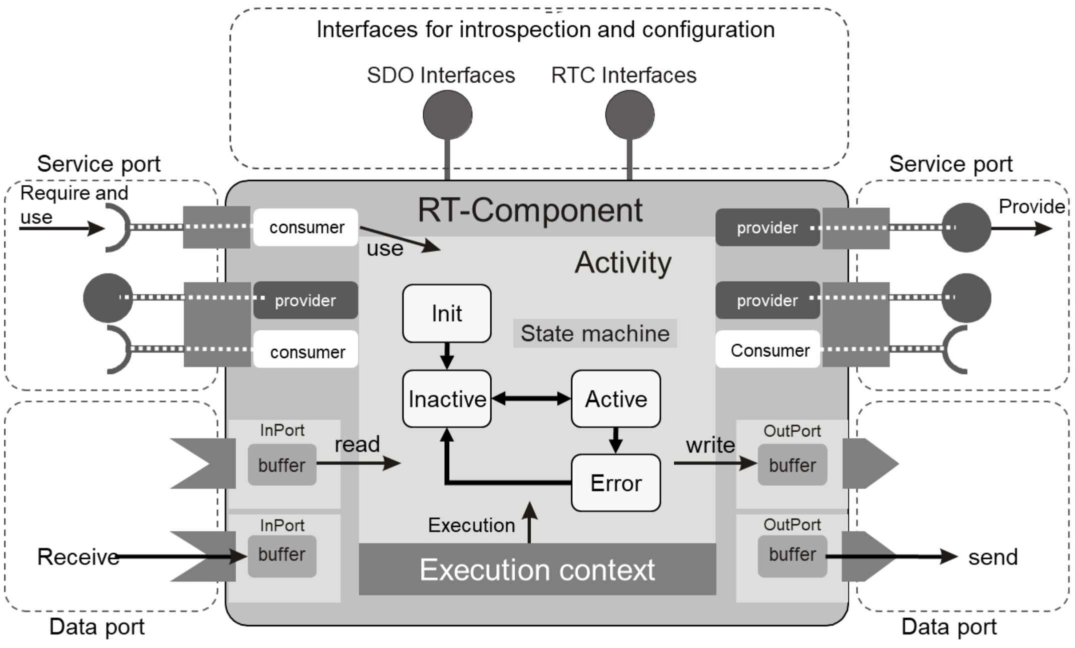
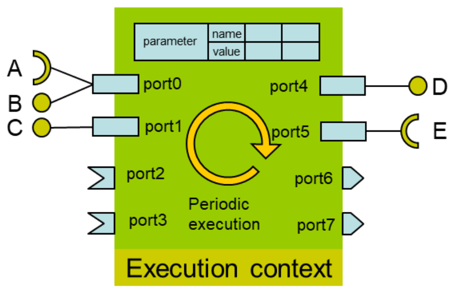
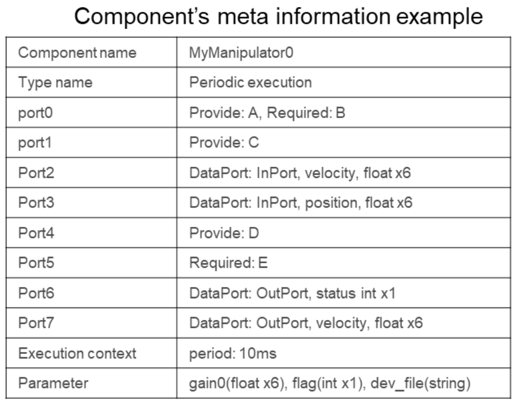


Figure 18: RT Component architecture

Get meta information: RTC has interfaces (introspection function) for obtaining meta information (RTC profile). RTC profile is a set of information that describes the characteristics of a component, such as the name of the component and the profile of Port's it owns. This function is necessary for dynamic system configuration at run time.

Introspection: A mechanism to acquire meta information of objects and components. The definitions are not commonly fixed, but the function is similar to Reflection in Java. OMG RTC specification defines the function as "Introspection".




Component's meta information example

Component name	MyManipulator0
Type name	Periodic execution
port0	Provide: A, Required: B
port1	Provide: C
Port2	DataPort: InPort, velocity, float x6
Port3	DataPort: InPort, position, float x6
Port4	Provide: D
Port5	Required: E
Port6	DataPort: OutPort, status int x1
Port7	DataPort: OutPort, velocity, float x6
Execution context	period: 10ms
Parameter	gain0(float x6), flag(int x1), dev_file(string)

Figure 19: Get meta information

Activity: Execution itself to implement major functions into a component. Common states such as Inactive (OFF state), Active (ON state), Error (Error state), etc. are defined for unified management of RTC. An RTC developer creates an RTC by mainly implementing the desired feature into the function (callback function) assigned to each state and each state transition event.

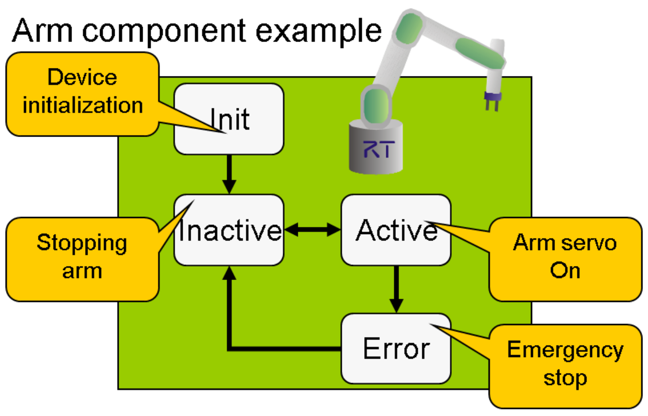


Figure 20: Activity and execution context

Execution Context: The functions which construct "Activity" are executed by a thread called Execution Context (EC). EC can be attached and detached dynamically to RTC. It is possible to attach one EC to multiple RTCs to execute them synchronously in series. Also, it is possible to make RTC execution real-time by changing the EC to real-time capable EC,

Data port: A data directional port for transmitting and receiving continuous data. There are two types of input port (InPort) and output port (OutPort). Even if the same data types are used, even if the language and OS are different, you can connect and communicate via the network.

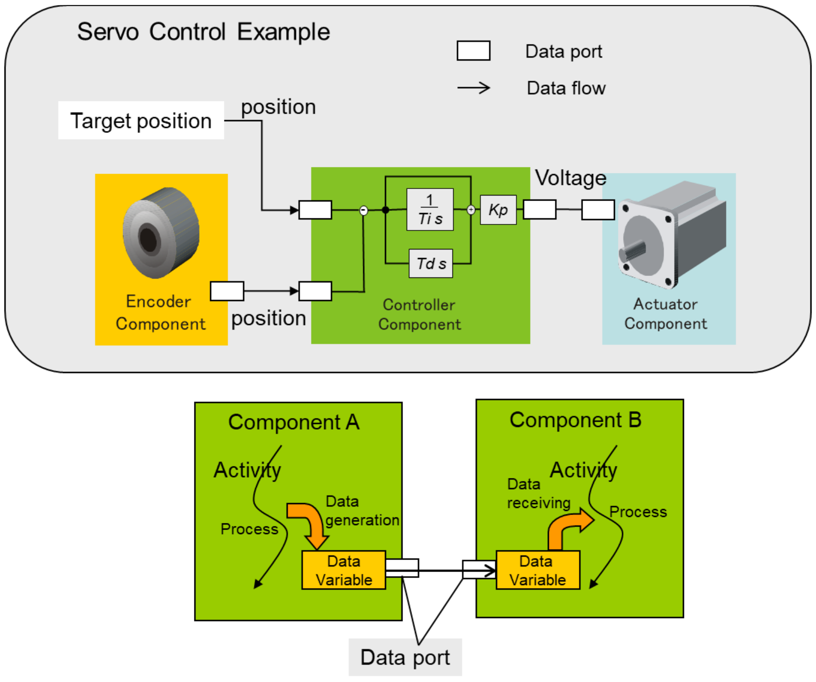


Figure 21: Data port

Service port: Ports which is used for providing functions and using external functions in command-base. It is used-defined. There are two types of interface - Provider (Provided interface) and Consumer (Required interface). The Provided interface provides functions to external units. Required interface is for requesting/using the function on external units. Like Data port, even if the language and OS are different, you can connect and call the function if the interface type is the same.

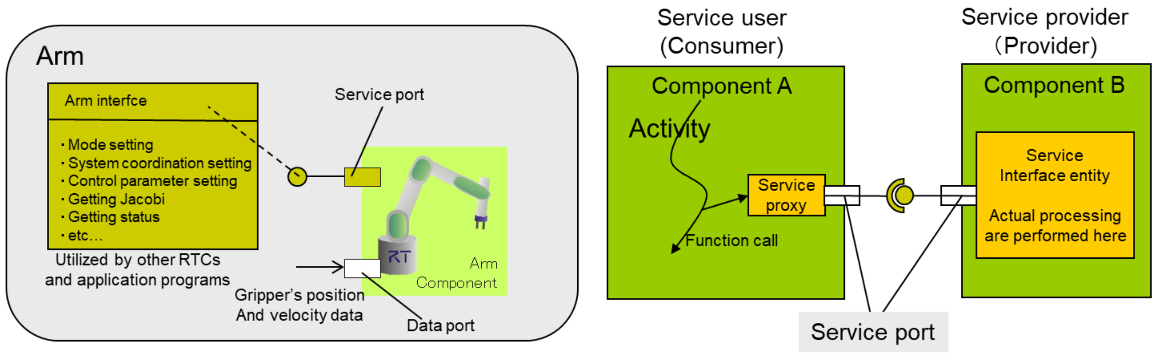
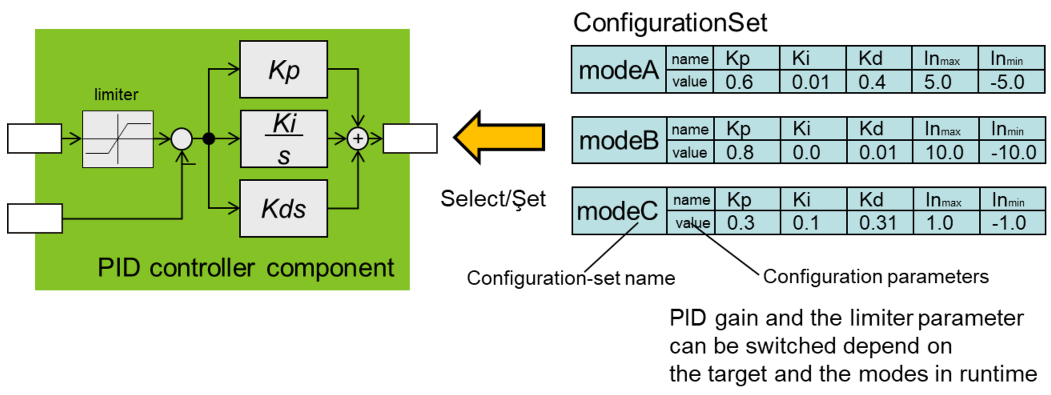


Figure 22: Service port

Configuration: A function to change the user-defined parameters externally at runtime. It has multiple sets of the parameters and you can swap them all at once. By making the parameters changeable, RTC can be reused in various systems.



		Kp	Ki	Kd	In$_{max}$	In$_{min}$
modeA	value	0.6	0.01	0.4	5.0	-5.0
modeB	value	0.8	0.0	0.01	10.0	-10.0
modeC	value	0.3	0.1	0.31	1.0	-1.0

Figure 23: Configuration

Generally, in the low-level part of the RT system like the servo controller, etc. are mainly composed by data-oriented tightly coupled subsystems. In high-level part like subsystems which judges and decides the behavior would be mainly coarse-grained and service-oriented. Since RTC realizes such modularization with various granularity with a common framework, the coupling between hierarchies which becomes a problem in the hierarchical other frameworks is not a problem.

The transparent linkage between RTCs in different languages is realized by using CORBA (Common Object Request Broker Architecture) which is the standard specification of distributed object middleware.

2.3.2.4 SmartSoft Framework[32]

The SmartSoft software aims at assisting the researcher, the robotics professional, the system integrator, the application domain expert and the end user, in realizing their ideas. The package offers model-driven tools for component developers, system integrators and professional end users. The designer can make use of component selection guides, and automatic generation of component hulls for wrapping algorithms and ensuring system-level composability. Further, the kit provides component repositories for navigation, vision, human-machine interface, task coordination, manipulation and a lot more that go into the simplest of robotic designs.

The factor that serves as a differentiator for SmartSoft is its approach towards robotics. Adopting a component-based view, SmartSoft looks at each component model in terms of communication patterns, which include clearly-defined semantics for interfacing these components (see Figure 24). While the component-based approach gives the robotic engineer flexibility in achieving software complexity, the catch is in having a system that can assist putting together a complete setup and providing software

[32] http://smart-robotics.sourceforge.net/

architecture without enforcing a specific robot architecture. It is precisely this fact that SmartSoft stands for.

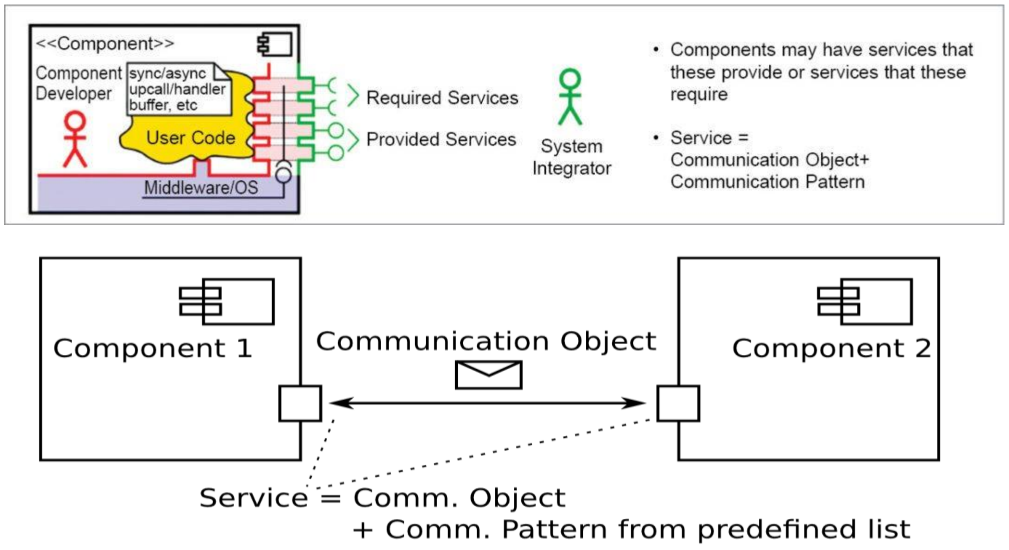


Figure 24: Structure of SmartSoft and what it means to you

The basic idea is to provide a small set of communication patterns, which can transmit objects between components and then squeeze every component interaction into those predefined patterns. Components interact solely via those patterns (see Figure 25).

The entire process is segregated into components, communication patterns and objects, and services. Components are technically implemented as processes containing multiple threads and interacting via predefined communication patterns. Patterns define the communication mode, doing away with all communication and synchronization issues, and acting like the master-slave protocol. Content of the packet is the communication object that is always transmitted by value, to avoid intermittent inter-component communication while using other methods. Every time a packet is instantiated, a service is provided.

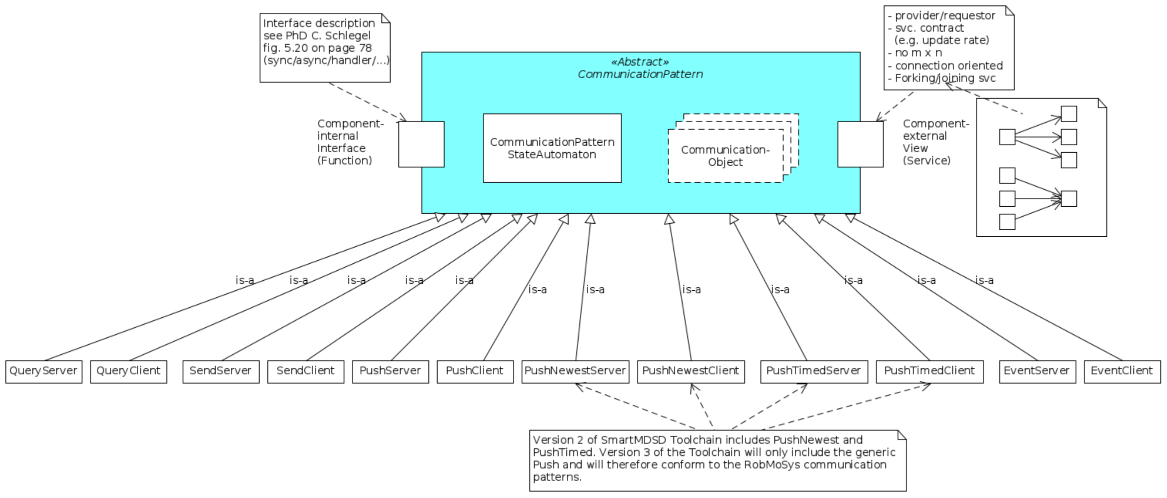

Interface description see PhD C. Schlegel fig. 5.20 on page 78 (sync/async/handler/...)
«Abstract» CommunicationPattern
Component-internal Interface (Function)
CommunicationPattern StateAutomaton
Communication-Object
- provider/requestor
- svc. contract (e.g. update rate)
- no m x n
- connection oriented
- Forking/joining svc
Component-external View (Service)
is-a
QueryServer
QueryClient
SendServer
SendClient
PushServer
PushClient
PushNewestServer
PushNewestClient
PushTimedServer
PushTimedClient
EventServer
EventClient
Version 2 of SmartMDSD Toolchain includes PushNewest and PushTimed. Version 3 of the Toolchain will only include the generic Push and will therefore conform to the RobMoSys communication patterns.

Figure 25: SmartSoft communication pattern

The main problem behind looking at things from a component perspective is the integration to form a whole. Many components might be manufactured concurrently and independently, and sometimes at different sites. How does one account for communication between these? This is where the component interfaces come into the picture. These need to be powerful enough to decouple external behavior from internal implementations, and this is what SmartSoft sets out to do. It allows the focus on a single component, go into depth and perfect it, regardless of the internals of other components. Then, a system is simply composed out of these approved components.

The same is extended to components that are being reused, avoiding the need for designing or analyzing from scratch. Internal state automation has at least three states representing neutral, alive and fatal states. Be it software or hardware, with lack of implementation standards at the former level, the bottom-up method turns advantageous.

The approach allows for dynamic wiring of components during run-time. Thus, loosely-coupled parts can easily be put together and adjusted according to the current context and environment, paving the way for designing distributed systems with ease. Control and data flows are configurable from outside a component. In addition, asynchronous communication decouples components to avoid passing on tight time dependencies and exploit concurrency to reduce latencies. It also allows a robotics engineer to work on his or her application without being dependent on the software counterpart. SmartSoft takes this into consideration, addressing challenging topics like location transparency of components, synchronization parameters or thread safety.

2.4 Drone Autopilot

UAV autopilot systems allow an unmanned aerial vehicle, such as a drone, to perform entire missions autonomously without the need for manual remote control. Operators use ground control stations to set the parameters of the mission and the UAV autopilot directs the drone or other unmanned craft to complete the task.

Among several open source autopilots researched, four of them stood out: Ardupilot, PX4, Paparazzi and LibrePilot. These were the ones still in development and that had a focus on autonomy. They provide several versions depending on the type of underlying drone structure: rover, helicopter, quadcopter, hexacopter, plane, vertical take-off plane, etc. They are depicted in Figure 26, together with inactive projects (i.e. with no updates since February 2017) and software focused on first person view (FPV). Inactive projects are in black and white, while active ones are colored in green. Also, when a project forks from another, it appears connected to the origin at its right.

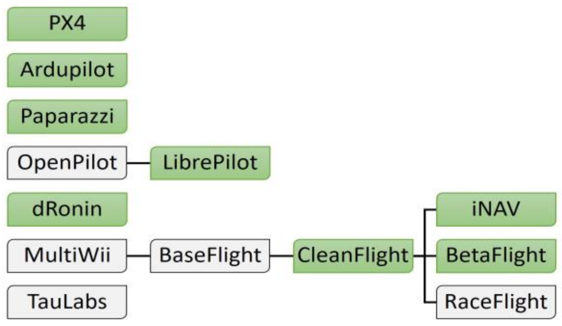


Figure 26: Autopilot projects evaluated during research

2.4.1 Ardupilot

From a task execution point of view, Ardupilot is based on a main loop that distinguishes critical functions from non-critical ones. The first piece of code to be executed, called the fast loop, regards the critical functions (critical motion control). Later, the time left for the non-critical tasks is distributed by an application level scheduler. The scheduler executes the other tasks in a best-effort manner, according to priorities statically defined by the coders. This structure is represented in Figure 27.

In Pixhawk boards, the main loop is executed at 400 Hz, and at 100 Hz in APM 2.x boards. These values are defined statically. For each board it is deployed to, Ardupilot has a different structure for its tasks and a different set of priorities, in order to consider the specific implementation characteristics of each hardware. Nevertheless, the distinction in priority levels between critical tasks and complementary ones remains the same.

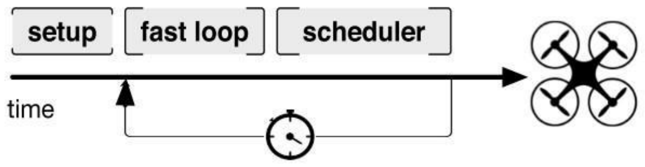


Figure 27: Ardupilot's control loop

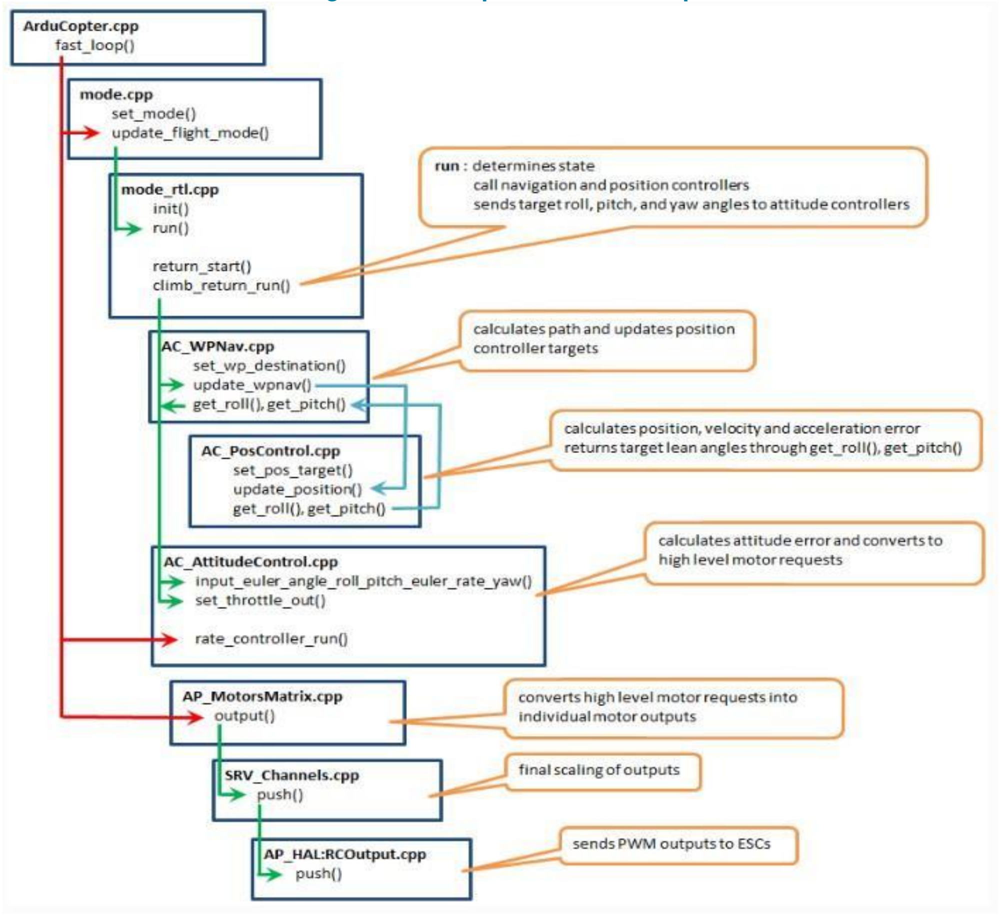

ArduCopter.cpp
fast_loop()
mode.cpp
set_mode()
update_flight_mode()
mode_rtl.cpp
init()
run()
return_start()
climb_return_run()
run : determines state
call navigation and position controllers
sends target roll, pitch, and yaw angles to attitude controllers
AC_WPNav.cpp
set_wp_destination()
update_wpnav()
get_roll(), get_pitch()
calculates path and updates position controller targets
AC_PosControl.cpp
set_pos_target()
update_position()
get_roll(), get_pitch()
calculates position, velocity and acceleration error
returns target lean angles through get_roll(), get_pitch()
AC_AttitudeControl.cpp
input_euler_angle_roll_pitch_euler_rate_yaw()
set_throttle_out()
rate_controller_run()
calculates attitude error and converts to high level motor requests
AP_MotorsMatrix.cpp
output()
converts high level motor requests into individual motor outputs
SRV_Channels.cpp
push()
final scaling of outputs
AP_HAL:RCOutput.cpp
push()
sends PWM outputs to ESCs

Figure 28: Ardupilot's autonomous control sequence

From a functionality point of view, the code has different modes, each one containing a slightly different structure. Yet, every mode has three main stages:

- Sensor data interpretation: At first, sensor data is filtered and translated to meaningful values from an engineer point of view (i.e. position, velocity and accelerations, both linear and angular, and also remote commands).
- Controllers: The values interpreted from the sensors are compared to reference values or used as new references in control loops.
- Actuators translation: The control outputs are translated to individual actuator (motor) commands.

An autonomous mode will have a simple attitude controller, that receives the targeted attitude (row, pitch and yaw), compares to the actual attitude, and then commands a certain change in the inclination of the vehicle, but also some control loops happening before, as seen on Figure 28: at first, the measured position is taken into account for updating the trajectory in the navigation control loop; then, this trajectory will determine a target position, which will be compared to the measured position in a position control loop in order to determine a target attitude; finally, once it is determined, the target attitude is used as an input for the same attitude control loop that exists in the manual mode.

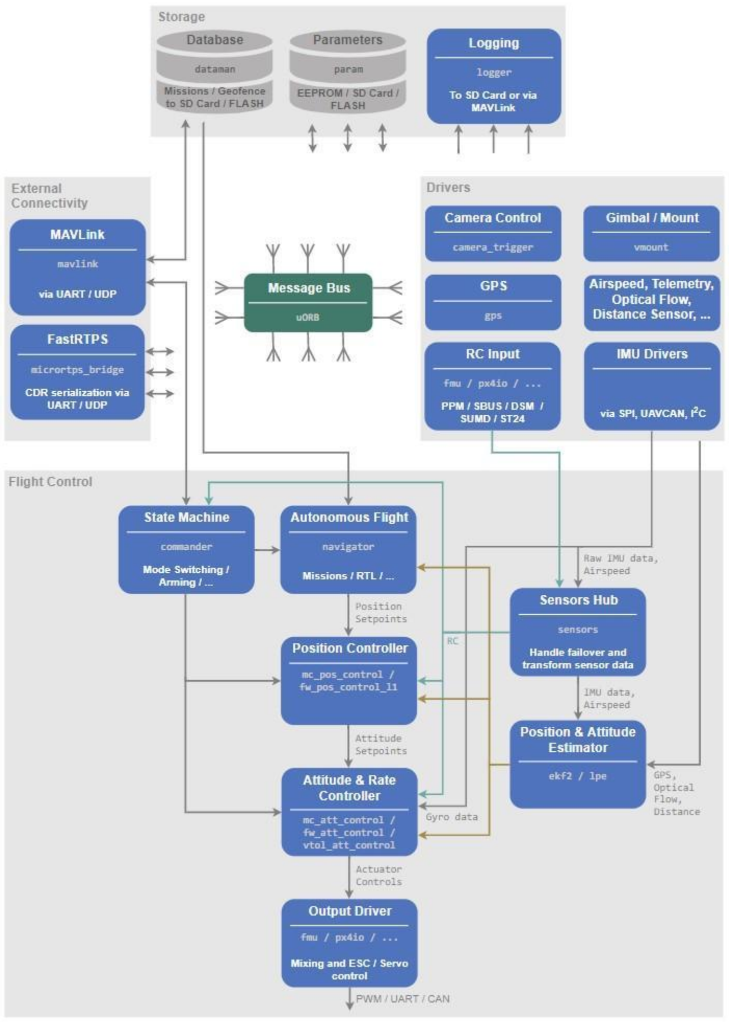


Figure 29: PX4 modules

2.4.2 PX4

PX4 is composed of several modules as seen in Figure 29, with roughly each module representing a different parallel task. This makes the system's operations and communication fully parallelized, and its components can consume data from anywhere in a thread-safe fashion. This data is made available in the form of messages, which are transmitted between modules using the uORB middleware. The drivers and operating system are modeled after the POSIX interface standards, and the onboard networking is following the UAVCAN standard proposal. PX4 also offers a FastRTPS Bridge to enable the exchange of uORB messages between PX4 components and off-board Fast RTPS applications, including those built using the ROS2/ROS frameworks. All of this gives PX4 a great modularity, which means, for developing a new module, only the topics it must interact with will matter to the developer. On the other hand, from a functionality point of view, it resembles that of Ardupilot. A diagram representing its behavior is presented in Figure 30. Also, the development team considered the time constraints involved in each control loop, as seen in Figure 31[33].

[33] H. Lim, J. Park, D. Lee, and H. J. Kim, "Build your own quadrotor: Open-source projects on unmanned aerial vehicles," Robotics &Automation Magazine, IEEE, vol. 19, no. 3, pp. 33–45, 2012.

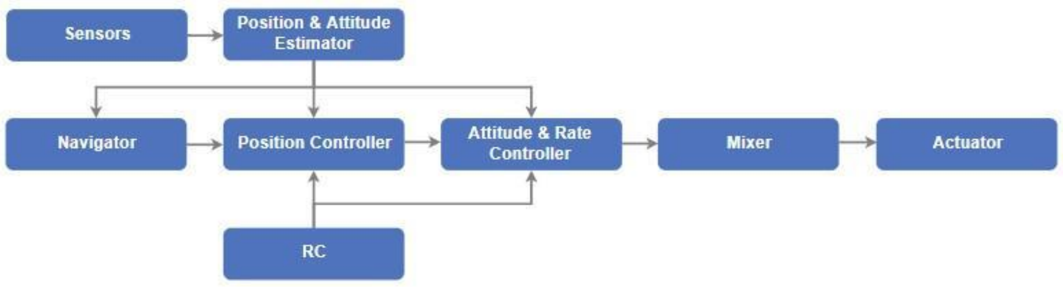


Figure 30: PX4 functionality structure

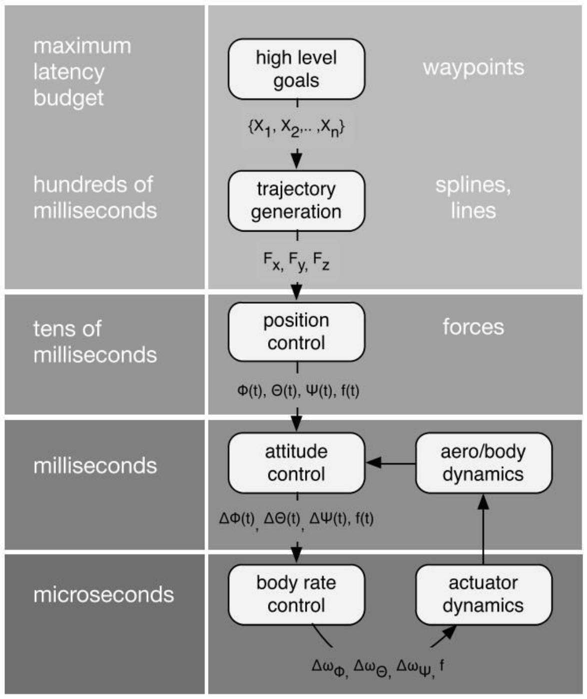


Figure 31: PX4 time constraints

2.4.3 LibrePilot

It is implemented using the FreeRTOS embedded real time operating system and a hardware abstraction layer named “PiOS” on top of the FreeRTOS base, as well as a custom scheduler for delayed callback functions with real-time scheduling guarantees.

The control loops, on the other hand, are different from the preceding ones. In Manual Control mode, the flight controls directly steer the actuators in an open loop. It is the pilot actuating the control transmitter that closes the loop using hand eye coordination or relying on aerodynamic flight stability. In Stabilized Control mode, the command from the receiver is fed into one of the two loops: in Rate mode, a single PID control loop is used for control; in Attitude mode, two cascaded PID loops are used where the outer loop controls the angular velocity setpoint based on attitude error, and the inner loop controls the actuator based on angular velocity errors. Finally, in control modes that involve the Autopilot, the craft is flown by the Flight Controller based on trajectory instructions.

The code is made of modules that can be callback functions or specific threads. The modules can only communicate using UAVObjects[34]. This gives the code a lot of modularity, but the modules are not self-contained – one still must know the code of the interacting modules to develop a function to interact with them.

[34] https://librepilot.atlassian.net/wiki/spaces/LPDOC/pages/8552484/UAVObjects

2.4.4 Paparazzi

The proposed autopilots have a list of built-in modes, covering most of the usual needs. A certain control stack is called for each mode, and although it is possible to choose the control loop being used at the moment. However, an experimental mode allows the implementation of a custom autopilot state machine. It is then possible to change or extend the number of modes, or even run several parallel control loops. See http://wiki.paparazziuav.org/wiki/Autopilot_generation for more details.

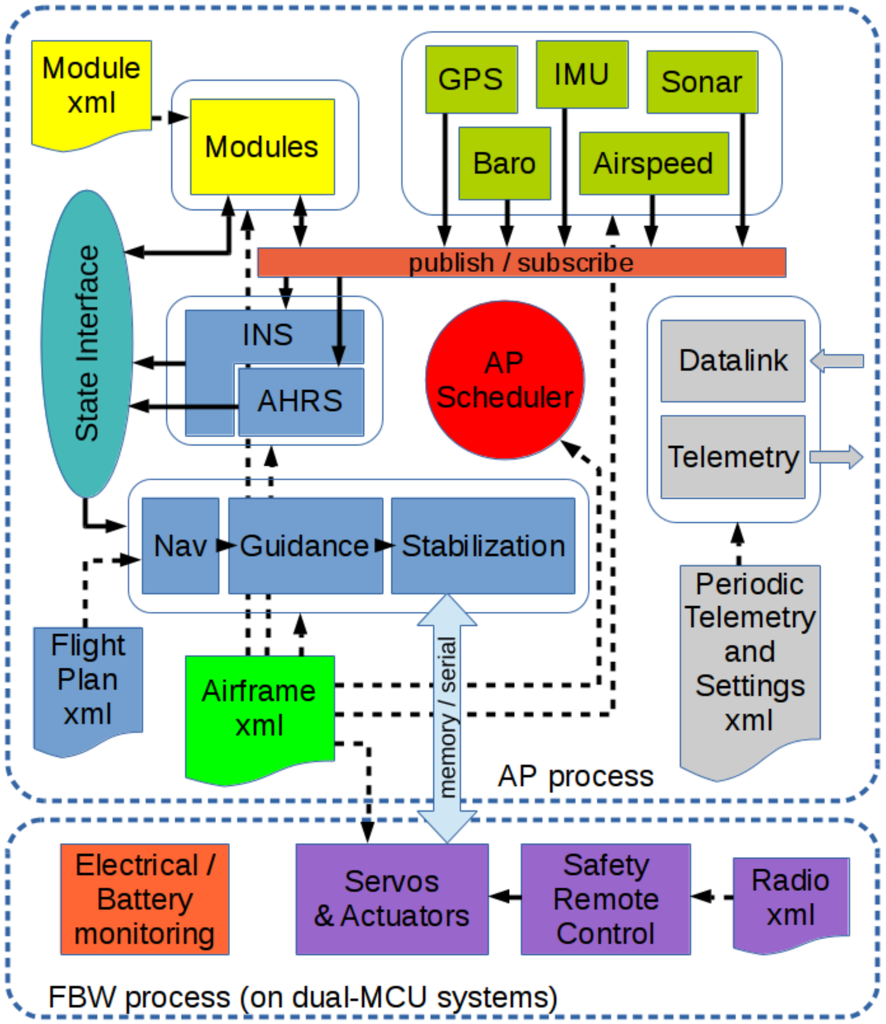


Figure 32: Paparazzi on-board architecture

These components are separated between critical (such as AHRS, IMU, INS and GPS) and non-critical, and the act of calling them is done in the same way as Ardupilot, with statically defined periods and in a sequential order. Although, efforts are being made to transition to a real-time operating system (RTOS) based solution. The latest hardware is based on ChibiOS (an efficient and light RTOS for microcontrollers). The main part of the autopilot is handled in a single thread, but most of the low-level drivers have their own threads, with either high priority (communication with internal sensors) or low priority (Secure Digital (SD) logging) compared to the autopilot task. It is also possible to create threads for certain heavy tasks related to payload management or parameter estimation in dedicated modules. This is ensuring a proper timing for the core autopilot control stack.

In addition, the data between the sensors and the estimation filters are managed by a publisher/subscriber mechanism, where the estimated state is published in a blackboard type interface for the control and navigation loops. Conversion between the different attitude or position representations (euler, DCM, quat, lat/lon, ecef, etc.) are automatically handled by this state interface.

The control loops, just like PX4, resemble that of ArduPilot, with a core attitude control loop, and the possibility of other control loops to execute on top of this one, with slower rates. However, the navigation layer is based on a specific flight plan language (based on XML files generating C code) allowing complex mission description (http://wiki.paparazziuav.org/wiki/Flight_Plans). It is also possible to use a dynamic task-based approach (http://wiki.paparazziuav.org/wiki/Mission). Both approaches are based on built-in flight patterns like goto point, fly segment or around circle, but can be extended with custom navigation patterns.

Paparazzi provides a simplified way to add modules without having to know the functioning of other modules, by using standardized XML file descriptions. The developer still needs to know how functions are called in other modules so that it can interact with them. It is also possible to create interaction between modules by using the internal publish/subscribe software bus.

3 C4D Reference Architecture

In this section, we introduce our vision of the drone system architecture that is going to be adopted in the COMP4DRONES project (this architecture will be specified in detail in the deliverable D3.2). To do so, we first describe the requirements/needs that need to be considered in designing the drone architecture. Then, based on these needs and the existing practices, we first start by describing the general architecture of embedded systems and then detail it for the drone system. We also describe a component template to ease the integration of the generic components (described in Section 4) with the architecture.

3.1 Requirements for Drone System Architecture

The primary stakeholders for the drone architecture are the engineers and developers that will implement the architecture in their application. Thus, a majority of the effort in designing the architecture should be to meet the developers' needs for the drone system. These needs are divided into two groups: user stories, and specific architecture elements' requirements. In this section, we also introduce the KPIs that are going to be used evaluated the architecture and its generic components.

3.1.1 Overall Architecture Requirements

The following are a set of user stories that have been collected from researchers and industrials to help define exactly what this architecture needs to specify[35]:

1. As a leader of a development team, I would like the developers to be able to **create independent software modules** in different programming languages so that development is faster and easier for the developers.

2. As a developer, I would like a **well-defined communication protocol** for the data that needs to be communicated between software modules so that modules can be developed independently.

3. As a system engineer, I would like the **software to be processor and computer architecture independent**, so that the system can be ported to a new computer without additional development.

4. As a system engineer, I would like the **software modules to maintain the lowest coupling** possible so that modules can be developed or modified independently.

[35]https://commons.erau.edu/edt/264/

5. As a system engineer, I would like the software modules to run seamlessly on a distributed computing platform with **minimal configuration**, so that the end user does not have to spend time configuring each module when process distribution changes.

6. As a leader of a development team, I would like the **common features of a software module to be defined and abstracted**, so that developers do not spend time re-implementing already implemented functionality.

7. As a leader of a development team, I would like the **communication between software modules to be lowly coupled** so that modules do not need to be aware of other modules within the system.

8. As a system engineer, I would like to be able to **seamlessly pick and replace** navigation algorithms like sensor filtering, path planning, and control systems so that different algorithms can be tested quickly and efficiently.

9. As a system engineer, I would like to add a new sensor or actuator to the system without having to change my state estimation, world modeling, or control algorithms and code base so that the **development time in adding a new device is reduced**.

3.1.2 Specific Architecture Elements' Requirements

From drone providers' perspective, there are a number of requirements for the drone autopilot and for integrating and customizing the generic components. In the following, we describe the requirements from the two perspectives.

3.1.2.1 Drone Autopilot

The requirements for the drone autopilot are divided into two groups: autopilot software, and autopilot hardware.

Autopilot Software Requirements:

- The autopilot shall provide a **modular approach to integrate new functionalities** or low-level drivers for external components.
- Autopilot software **shall be implemented by functional layers** (communications, data, devices, flight management, etc.). Every layer shall be implemented with independent functional modules.
- The autopilot software **must be separated** from the applicative domain and be easily extensible.
- The autopilot shall be able to use a Real-Time Operating System (**RTOS**).
- The autopilot shall provide **event-based mechanism to exchange data** between its internal modules, especially for the sensor data.
- The autopilot shall perform **sensor fusion**, unless an external component is already providing the information over a safe interface
- The autopilot shall **monitor** and report to the ground station and external onboard components(e.g. companion computer)the sensor data, the status of the sensor fusion algorithms, the power supply of all critical components, and the CPU load
- The autopilot shall **log onboard the flight data** for post-flight or failure analyses
- The autopilot shall implement a **communication protocol** (data encapsulation and messages definitions) for communication with the ground station and the external onboard components
- The autopilot shall provide the necessary **stabilization and guidance** capabilities to comply with the mission and the regulatory requirements in case of fail-safe procedures.
- Each **function** provided to the Ground Station shall be **accessible** through to the communication protocol to the external onboard Components (to allow a companion flight mission computer).
- The autopilot shall have a **strict version control and version release procedure**, in order to ease integration in complex systems.

- The autopilot should provide a mean to **measure its integrity** and well-functioning (either through the communication protocol or through a dedicated interface, e.g. heartbeat port).
- The autopilot shall allow **flying a path** that is dynamically defined/updated by a ground station or an external onboard component.
- The autopilot shall provide **low level navigation capabilities** such as “move to relative”, “move to global”, “set 3D-velocity”, etc.
- The autopilot shall **provide a geocaging feature** that guarantees the vehicle will not breach a defined flight area.

Autopilot Hardware Requirements:

- Autopilot hardware must have **different power sources** (voltage and amperage)
- The autopilot shall have **hardware interfaces** (communication ports) used by low level sensors such as UART, I2C, SPI, GPIO, ICU, ADC.
- Autopilot shall **cover a single component failure** “slave” controller takes over control in case of a component fail of “master” controller.
- The autopilot should be **cooled or allow cooling** to ensure proper functioning for ambient temperature.

3.1.2.2 Requirements for Component Integration

- The hardware component to be integrated in the drone:
  - Shall be sufficiently small and light.
  - Low power consumption.
  - Shall support voltage ranges that are typically available in drones.
- The external component (software or hardware) shall support a minimum set of configuration capabilities.
- Hardware and software components should support at least one standard communication protocol and will defined interfaces.

3.2 KPIs for the Architecture and its Generic Components

The overall success criteria seek to demonstrate measurable gains for the COMP4DRONES users, in terms of efficiency, risks, reuse, and sustainability, by comparing estimated levels at the end of the project with the levels at the beginning. The numbers used in the success criteria are derived from internal discussions among the project partners and estimations coming from practical experience in the field and from the state of the art. The following are the two project objectives and their success criteria that are related to the architecture specification and implementation.

Objective 1: Easing the integration and customization of drone embedded system

- SC1.1. Demonstrate a potential gain for design efficiency of drone embedded platforms by reducing their integration, customization and maintenance efforts by 35%.
- SC1.2. Demonstrate a potential reuse of pre-qualified drones application components, leading to 35% of effort reduction for system-level assurance activities.

Table 1: KPIs for the architecture and its generic component linked to project objectives

KPI Source	Metrics	WP KPI	Criteria	Objective
BUT, IMEC-BG, ACORDE, CEA, HIB, ENSMA	Time spent for configuration	Ease of HW/SW **configuration**	SC1.1	O1
ACORDE	Number of user interactions			

SCALAIN, HIB	Number of lines of code			
BUT, IMEC-BG	Number of parameters			
EDI	Number of files changed			
BUT, ACORDE	Number of parameters	Ease of HW/SW **integration**	SC1.1	O1
ACORDE	Number of mechanical elements			
BUT, ACORDE SCALAIN	Number of lines of code			
EDI	Number of files changed			
MODIS, IFAT	Integration effort			
BUT, SCALAIN, ENSMA	Number of lines of code	Ease of **incorporating** new hardware/software components	SC1.1	O1
INDRA	Is same internal HW architecture is maintained?	Maximizing **reuse** of components	SC1.1	O1
IKERLAN	Obstacle and trajectory tracking can be reused?			
UNISS, UNIMORE	Camera communication protocols can be re-used?			
IFAT	Is the provided API designed to be reusable?			
INDRA	Lines of code Number of parameters	Improve **interoperability**	SC1.1	O1
SCALAIN	Number of lines of code	Easy to **configure new missions** for a fleet	SC4.1	O4
UNISS, UNIMORE	Number of lines of code	Easy to **design, develop, and deploy** applications	SC4.1	O4
UDANET				
UWB & SM				
TUE	Verification and certification time and cost	Improving the **verification, validation and certification** process	SC4.1	O4
ENAC	Time for test case simulation Code coverage percentage			
SIEMENS	Modeling and simulation effort	Improving **modeling and simulation** accuracy	SC4.1	O4
UINVAQ	Simulation execution time			
UDANET				

Objective 4: Minimizing the design and verification efforts for complex drone applications

- SC4.1. Demonstrate a potential gain for design and assurance efficiency of drones embedded platforms by reducing specification, verification & validation and implementation efforts by 30%.
- SC4.2. Demonstrate a potential gain for certification of drones embedded systems by reducing the certification effort by 25%.

The KPIs for the work that is going to be performed for specifying and implementing the reference architecture and its generic components are shown in Table 1. These KPIs are also linked with the project success criteria and objectives.

3.3 COMP4DRONES Architecture Specification

The goal of the architecture is to describe, in sufficient detail, all the components, their responsibilities and behavior, and their interfaces to allow a more detailed design to proceed forward. A goal of the architecture is to support those responsible for integrating systems that consist of parts coming from multiple providers. A modular architecture should help in customizing the drones easily to support new features and missions.

In this section, we start by describing the general system of systems view of the UAS (sub-section 3.3.1), followed by reference model of embedded systems (sub-section 3.3.2). Then, we describe our vision of the COMP4DRONES architecture (sub-section 3.3.3), and its generic components' template (sub-section 3.3.4).

3.3.1 Drone System-of-Systems View

A UAS can be divided into five distinct elements as shown in Figure 33[36]. First, the air vehicle element consists of the airframe, propulsion and the avionics required for air vehicle and flight management. Second, the payload element is comprised of payload packages. These can be sensor systems and associated recording devices that are installed on the UAV, or they can consist of stores, e.g., weapon systems, and associated control/feedback mechanisms, or both. Third, as illustrated, the data link element consists of the Vehicle Data Terminal (VDT) in the air vehicle and the Control Data Terminal (CDT) which may be located on the surface, sub-surface or air platforms. Control of the UAS is achieved through the UAV control system (UCS) and data link elements. Although shown as part of the UAS surface component, the UCS and the associated data link terminal can be located in any platform, (e.g., another air platform).

Forth, the UCS element incorporates the functionality to generate, load and execute the UAV mission and to disseminate useable information data products to various C4I (Command, Control, Communication, Computer and Intelligence) systems. It should be noted that Figure 33 shows a common path for UAV command and control, payload command and control, and products. These functions may be accomplished on separate, independent data links. Finally, the launch and recovery element incorporate the functionality required to launch and recover the air vehicle(s).

In this deliverable, the focus is on the architecture for the air component of the system. Therefore, the next sections will describe the reference model of embedded systems followed by our vision for the COMP4DRONES architecture and its components template.

[36] STANAG 4586 Ed.3 Nov 2012, Standard Interfaces of UAV Control System (UCS) for NATO UAV Interoperability, NATO Standardization Agency (NSA), 2012

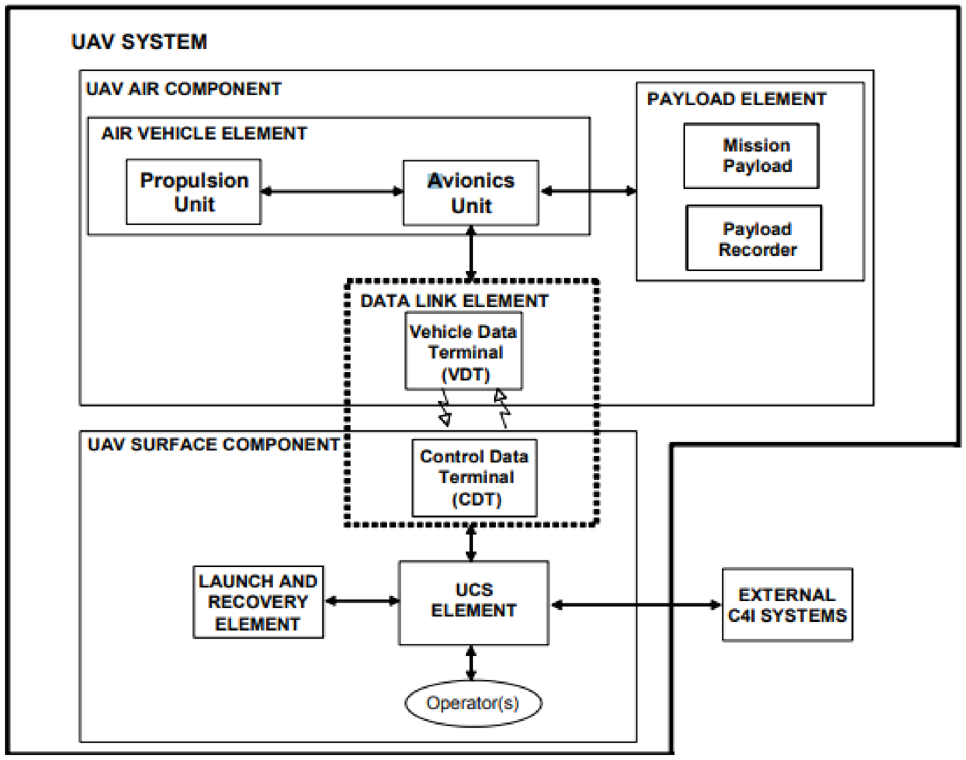


Figure 33: General model of standard UAV System

3.3.2 Reference Model of Embedded Systems

The reference model of embedded systems is illustrated in Figure 34[37]. It shows the main components of an embedded system. What the Embedded Systems Model implies is that all embedded systems share one similarity at the highest level; that is, they all have at least one layer (hardware) or all layers (hardware, system software and application software) into which all components fall. The hardware layer contains all the major physical components located on an embedded board, whereas the system and application software layers contain all of the software located on and being processed by the embedded system.

This reference model is essentially a layered (modular) representation of an embedded systems architecture from which a modular architectural structure can be derived. It is possible to understand the architecture of all of the embedded systems by visualizing and grouping the components within these devices as layers. The concept of layering is a useful tool in visualizing the possible combinations of hundreds, if not thousands, of hardware and software components that can be used in designing an embedded system. In general, this modular representation of embedded systems architecture is selected for two main reasons:

- The layered approach allows readers to visualize the various components of an embedded system and their inter-relationship.
- In modular architectural representations, the various modules (elements) within this type of structure are usually functionally independent. These elements also have a higher degree of interaction, thus separating these types of elements into layers improves the structural organization of the system without the risk of oversimplifying complex interactions or overlooking required functionality.

[37] Tammy Noergaard, Embedded Systems Architecture (Second Edition), Newnes, 2013

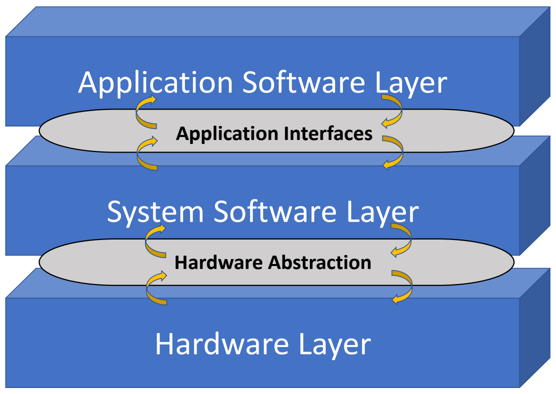


Figure 34: Embedded Systems Reference Model

By analyzing the architectures described in Section 2, we found all of them are following this reference model. Therefore, in the next section, we are going to describe our vision for the architecture that will be adopted in the project following this reference model.

3.3.3 Initial Specification of COMP4DRONES Architecture

Inspired by the modules present in PX4, Ardupilot and Paparazzi, as well as some other autonomous vehicles architecture, a draft of a general architecture is presented in Figure 35. The architecture contains (a) Flight Management block that controls allocation on respective actuators/servos and a sensor network installed onboard of the drone platform; (b) Guidance refers to the definition of a desired path of travel from starting point to the destination, along with associated changes in velocity, acceleration, and drone attitude; (c) Navigation refers to drone state determination (location, attitude, velocity, acceleration) at any given time, (d) the Control itself refers to the manipulation of forces and moments by steering, to execute guidance commands while preserving drone's stability; (e) Communication block to enable drone-to-drone and drone to ground control station communication; and (f) Coordination block to coordinate the drone with the others in the fleet. Information arriving from cooperating drones and control stations are processed in the flight management block.

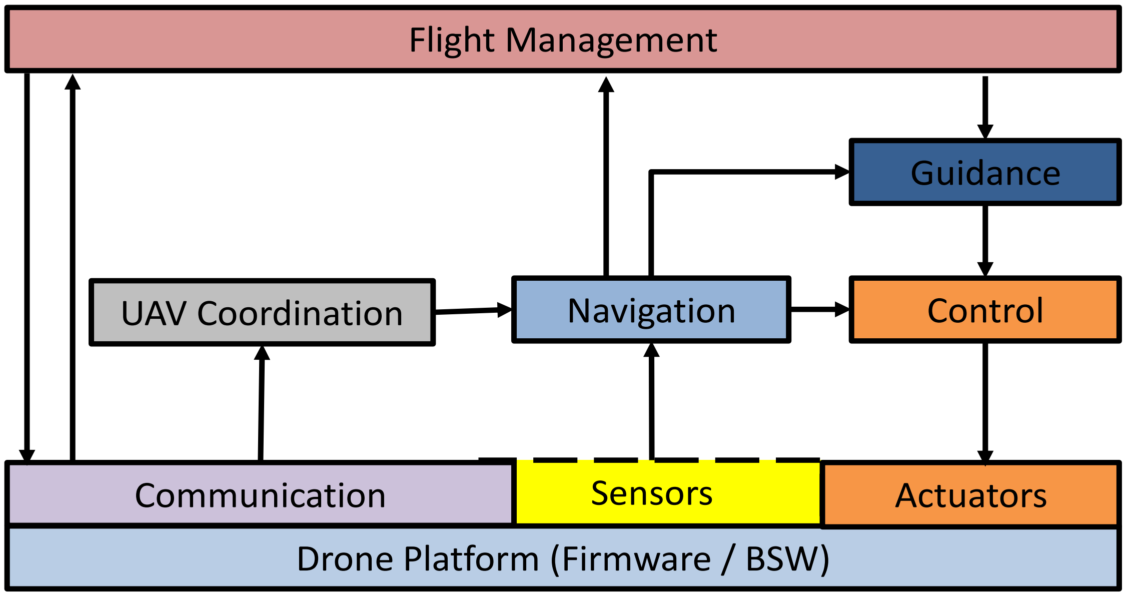


Figure 35: General architecture of drone systems

In relation to the reference model in Figure 34, the drone platform together with the sensors, actuators, and communication are corresponding to the hardware and hardware abstraction layers. On top of them, the system software layer that is composed of the navigation, control, coordination, and guidance modules. The most abstract layer present nowadays is the flight management that provides functions so that the drone can follow simple orders such as “Fly to” or “Circle point”.

Each module mentioned can be composed of sub-modules, and every change in the requirements corresponds to a change in a subset of these modules. For example, changing the physical characteristics of the drone might only require a change in the Control module. Therefore, technologies that increase modularity, such as ROS and Service Oriented Architecture (SOA), can have a positive impact on designing iterative embedded software.

The following list shows some of the architecture interfaces (presented in Figure 36):

- **Measured quantities:** Instant values obtained from sensors.
- **State vectors**: Vectors of Position, Attitude, Velocity, Angular Velocity, Acceleration, Angular Acceleration, and Jerk.
- **World model:** Interpreted information about surrounding world (obstacles, other entities, etc.).
- **Targets:** Desired trajectory (may include attitude targeting).
- **Functions:** Basic functions, as “FlyTo”, “Circle”, etc.
- **Mission:** Set of activities and priorities to be considered during execution.
- **Change parameters:** Possibility to change gains in order to adapt to new situations (such as the change in dynamics resulting of a package drop).
- **Change target values:** Change the reference value in position and attitude control loops.
- **Applied force and torque:** 3D vectors commanding force and torque to be executed by actuators.
- **Commands:** Individual commands to individual actuators.

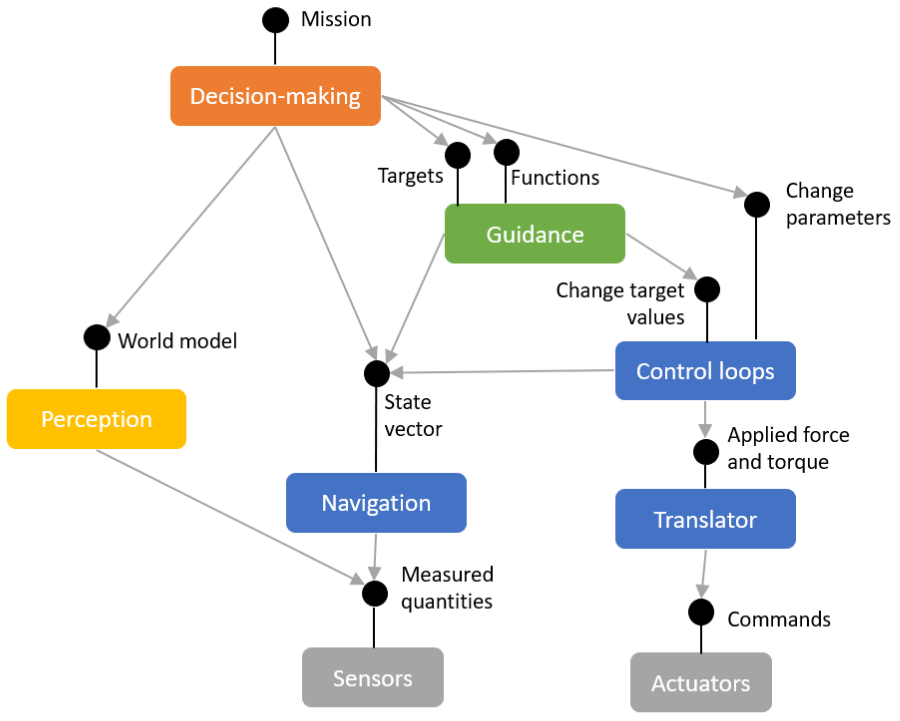


Figure 36: Initial proposition of interface between drone software modules

The communication module shall interact with most of the interfaces presented in Figure 36. Also, payload-management modules might only be able to access State Vector and World Model interfaces, and maybe some specific sensors, minimizing interactions with the autopilot. The latter, if seen as a black box, will need a bridge to connect with the rest of the system, and the bridge shall allow access to as many of the mentioned data as possible.

From an end-user perspective, a drone can be seen as a tool that has to fulfill certain requirements, in order for it to complete a designated mission. These requirements are the starting point for a drone configuration process, as they will define the missions that can be executed. Each mission can be seen as an ordered set of elementary activities, or operations. The latter will be elements of a set which is defined by the components present in the drone.

Therefore, starting from a set of models of available components, one could integrate the components of a subset of one's choice to compose the model of a desired structure, which completes a designated mission. The resulting model could then be validated according to constraints chosen by him.

The proposed general architecture shall comply with security constraints currently imposed by international organizations[38], as well as be flexible enough to adapt to changes in these constraints. Its modularity and pre-defined components shall allow a faster adaptation to multiple kinds of missions.

3.3.4 Template for the Architecture Generic Components

Inspired from openRTM (see sub-section 2.3.2.3) and SmartSoft (see sub-section 2.3.2.4) component architectures, we have proposed a template for the generic architecture components as show in Figure 37.

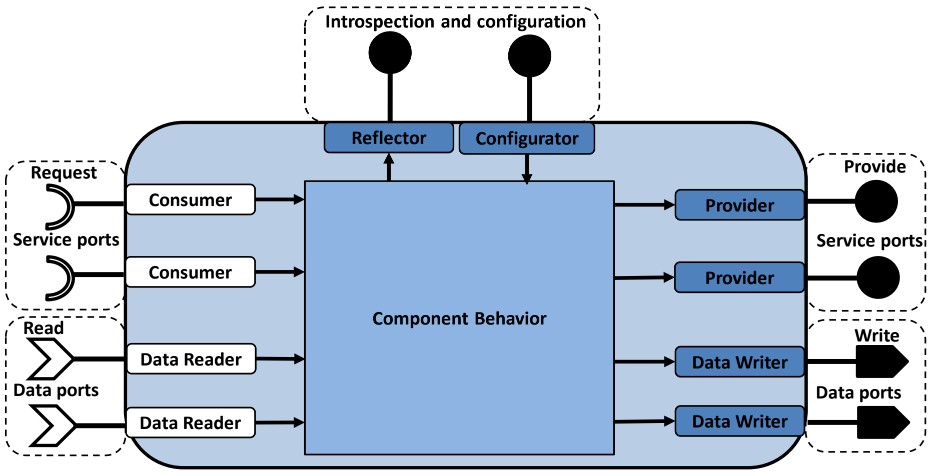


Figure 37: A template for the architecture generic components

The component template contains the following elements:

- **Introspection and Configuration**: Introspection is the mechanism to acquire meta-information of the generic component, while the configuration function is used for changing the user-defined parameters externally at runtime.
- **Service Ports**: Service ports are used for providing functions and using external functions in command-base. There are two types of interfaces - Provider (Provided interface) and Consumer (Required interface). The Provided interface provides functions to external units. Required interface is for requesting/using the function on external units.
- **Data Ports**: A data port is used for transmitting or receiving data. There are two types: input ports (for reading data trough readers) and output ports (for writing data through writers).
- **Component Behavior**: The behavior is corresponding to the implementation of functions into a component. It includes common component states such as inactive, active, error, etc.

[38] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7206421/

4 Generic Architecture Components

To ease the customization and integration of drone systems, a number of generic components will be developed together with the architecture. These components are divided into four groups and described briefly in the following sections (a more detailed description of these components is available in Appendix A). These groups are: system functions, payloads, hardware platform, and tools.

4.1 System Functions

The drone system functions are the core functions required for the drone to perform its flying stages in safe and efficient manner. The generic components in this category includes: data processing algorithms, computer vision components, indoor-positioning, SLAM algorithms, etc.

4.1.1 Sensor Data Processing Algorithms - BUT

In this component BUT will implement/improve sensor data processing algorithms which will include software and firmware for FPGA. This will involve video processing algorithms (for example HDR algorithms). HDR multi-exposure fusion algorithm to be implemented in the drone, possibly implementing tone mapping and/or ghost removal (with most probably somewhat limited capabilities) in order to "feed" further image and video processing subsystems in the drone by image information with high dynamic range.

The HDR pipeline is implemented in FPGA and HDR image acquisition pipeline is composed from three main blocks: image capturing, HDR merging, and tone mapping. The image capturing part is grabbing the images from the sensor. HDR merging processes multiple images into the HDR frame. The tone-mapping block is compressing the high dynamic range into the standard, 8-bit image while preserving the details from HDR. This output can be fed into following computer vision algorithms. In the COMP4DRONES project BUT will improve HDR fusion algorithm to be suitable for drones and also improve latency and throughput.

4.1.2 Computer Vision Component for Drones - HIB

The computer vision component for drones is an AI system will be built upon deep learning techniques in order to improve the way of interpreting surroundings and detecting scenarios from data captured with drones. The component will carry out a post-processing of the captured images with the drone cameras.

Concretely, the computer vision component is envisaged as a system for the post-processing of images captured by drones that will be used in the UC2-Demo 1: Digitalization of the state of the constructive process of a Civil Infrastructure, with the aim of understanding the captured scenario by drones in civil infrastructure. The AI system will be built upon automatic algorithms for the auto-detection/geo-referencing of road elements and taking cloud of points generated by a LIDAR sensor.

The component will go beyond the state of the art by providing automatic algorithms for the auto-detection/geo-referencing of road elements. The process will be done thanks to the combination of a computer vision system that will be applied upon a cloud of points generated for a specific scenario captured from a LIDAR camera integrated in a drone.

The innovation expected for the component provided is to analyze scenarios captured during drone campaigns with an in-device camera, for the detection of objects in the images and provide an inventory of elements and their position in the terrain with the objective of accelerating the Constructive Process of a Civil Infrastructure.

4.1.3 Generic API for Trusted Communication - IFAT

Within the scope of WP3, IFAT will develop modular and generic software components which are closely aligned with the corresponding hardware-security component which IFAT will develop within WP5.

The main application purpose of IFAT's modular software components (WP3) and the corresponding security-module parts (WP5) in the COMP4DRONES context is using "IFAT hardware-security component". In addition to the main (general purpose) microcontroller, in order to increase the level of security of future drone-identification and to establish secured communication between drone-2-basestation as well as potential drone-2-drone.

The targeted improvements proposed by IFAT are:

- Using additional separated hardware-based security component – often denoted as Secure Element (SE) or Hardware Security Module (HSM) – to partly assist the main microcontroller for protecting the higher security-critical parts, such as key generation and storage and according cryptographic primitives.
- On the software side (main microcontroller), for increased flexibility the proposed IFAT software components and libraries will be designed in a modular and partly generic way. On the one hand this concept should allow easier integration into different drone hardware/software-platforms and on the other hand partly allow certain flexibility in the case of future protocol updates.

4.1.4 Control Components that implement Potential Barriers - ENSMA

A control loop that implements a "potential field" is used for preventing the drone to access certain areas by evaluating its position, geofences and potential obstacles in the environment.

It is assumed that drone is equipped with proximity sensors and can detect any relative position of both non-cooperative and cooperative entities within a sensing range. A cooperative entity is another drone which has the same capability. In contrary, a non-cooperative entity is an entity without collision avoidance system.

Sensing capability is required to sense the presence of any other entities in its close vicinity which may lead to a collision. These sensors only give the relative position of any entity in its range in the local frame and do not provide position information in the global frame. It is to note that this assumption is used for the purpose of collision avoidance only.

The autonomous flight control will include a mechanism to avoid collision between the drones, obstacles and/or virtual barriers. An Artificial Potential Field (APF) method will be used for collision avoidance. In APF, a drone is considered as a point in a potential field. This drone experiences a repulsion force from the obstacles and therefore, instead of colliding with them, it steers around them. Typically, potential functions are based on the relative distance between drone and the obstacle and do not require any global information.

4.1.5 Multi-agent Swarm Control- ENSMA

In many practical scenarios, it is required that the agents of Multi-Agent System (MAS) create and maintain a desired geometric shape. The required shape could either be fixed or time-varying. In some cases, it is further required that the agents follow a trajectory while maintaining the shape. The trajectory is produced by a virtual or real leader. This is known as formation tracking.

The aim of this task is to develop distributed consensus and formation of a swarm of drones. From the state of art, it is clear that the available cooperative control schemes do not take various limitations[39] into account. Motivated by this, the current task focused on the design and implementation of distributed cooperative control laws for a swarm of drones with communication and sensor constraints.

4.1.6 Ultra-Wideband-based Indoor Positioning - ACORDE

The Ultra-Wideband (UWB) Indoor positioning system (IPS) will enable real-time, accurate position to a tag in an indoor scenario. The integration of the tag on a drone enables its safe navigation on the indoor scenario, where other accurate location sources are not available. It is based on the estimation of

[39] https://arxiv.org/pdf/1908.02789.pdf

accurate ranges from a mobile tag to statically deployed anchors. The mobile tag can either provide the ranges to the anchors for being externally fused with other data sources (default use) or its auto-calculated position. Ranges estimation will be based in this case on ultra wideband technology and two-Way ranging based location.

The UWB IPS to be provided by ACORDE will be focused for providing an affordable solution specifically designed for real-time location on long indoor infrastructures (e.g. tunnels, mines, and long corridors).

The solution is being designed for being specifically oriented to 3D location (essential for drone navigation) in the tunnel scenario, but with the aim to offer geo-location (to let the drone navigate through a geo-located predefined path).

4.1.7 Position and Attitude Estimation - ACORDE

The position and attitude estimation system of ACORDE provides a trustable positioning and attitude solution. It is based on a multi-antenna/multi-receiver approach and the implementation of sensor fusion with different kind of sensors like accelerometers, gyroscopes and barometer. The component supports an extensive set of parameters whose values can be adapted to the characteristics of an UAV scenario and its expected dynamics.

The integrity and accuracy of position and attitude estimates output enable to feed reliable and accurate navigation data to a drone autopilot, for its safer operation. In addition, an offline trace of this position and attitude data synchronized with the GNSS absolute time base can be exploited by a digitization application.

Regarding the GNSS sensing capabilities, the main ones are possibility to exploit multi-constellation (and specifically Galileo) and additional security features, to avoid both unintended and malicious attacks. This is a specific aspect that has been already started to be tackled in COMP4DRONES, though a redesign covering a new type of GNSS receiver supporting also Galileo constellation.

Even more interest and focus will be put for improving the output interface of GLAD (ACORDE HW/SW platform). ACORDE has a specific interest in leveraging, and working on a more standardized interface able to retrieve the output data (and thus added value) provided by GLAD system, and enable, ideally, a Plug and Play approach (where the SDK support become less or no relevant). Another relevant aspect that ACORDE has already started to analyze is the possibility for the SW platform to rely on a license free, real-time operative system.

ACORDE and UNICAN will also develop key models of position and attitude estimation. ACORDE in SystemC and UNICAN with S3D (UML/MARTE) will feed the design frameworks developed and reported in WP6. The S3D model will be integrated in a component's library for further reuse in future projects. The S3D modeling methodology encapsulates a functional component with its functional code (i.e. in C++) and the verification test-bench. The system model is automatically generated taking into account the underlying execution platform where the components will be executed.

Both the SystemC and the S3D models will be used and assessed in UC2. Moreover, ACORDE and UNICAN will collaborate to align the models so that comparatives of interest on alternative design technologies, e.g., for performance estimation, can be assessed, and this way gains key information on the best design technology choices.

4.1.8 Hyperspectral Imaging (HSI) Processing Pipeline - IMEC-BG

The goal of this component is to design a processing pipeline based on hyperspectral imagery to inspect (off-shore) critical infrastructure, such as off-shore wind turbines. IMEC-BG will further develop its current pipeline and rely on its expertise to efficiently deploy algorithms on the NVidia Jetson boards.

The pre-processing consists of four different sub-modules. First, a demosaicking algorithm is developed to estimate a multispectral image with full spatial-spectral definition. Second, a hyperspectral image cube is created. Third, several corrections are carried out: radiometric, non-uniformity, reflectance, spectral (varying lighting conditions) and corrections coping with degradations due to vibrations, fading, etc. Finally, the hyperspectral images are stitched together and a 3D model is constructed.

4.1.9 Simultaneous Localization and Mapping Algorithms - Modis

The Simultaneous Localization and Mapping (SLAM) capabilities will be based on mixed sensors measurements and will not rely necessary on the GPS signal. In fact, the system will be equipped with a watchdog alert for failure detection of the GPS.

Magnetic-field positioning has been successfully exploited in classical indoor robotics and it is appreciated for its low implementation costs. We aim to investigate its suitability as a support to the GPS signal for outdoor positioning. Our algorithms will also provide state of the art mechanisms for fail-safe mission and increased Performance Based Navigation (PBN) operations. As a byproduct, this system should also allow for real-time as well as post-mission analytics aimed at identify critical regions in the map where GPS signal is particularly weak and/or detect potential attacks.

4.1.10 High Accuracy EGNSS Navigation Component -TopView

The Global Navigation Satellite System (GNSS) component that will be integrated by TopView is based on a dual frequency, multi-constellation European GNSS receiver to be installed on-board with two typologies of drone platforms capable to embark it, considering both fixed wing and multi-copter drones. In both cases, the electrical and power interfaces of the GNSS component will be connected with the drone’s main communication bus and with the drone power bus. This component will improve drone navigation performance and will introduce a new functionality tailored for U-space tracking services.

In particular for smart farming applications (Use Case 5), this component is intended to address specific challenges in survey application:

- Facilitation of AI algorithms analysis: Thermal and Multispectral images with the current relative low resolutions available are in general not suitable for accurate orthmosaic processes or to build up heatmap layers in software suites, unless the drone flights are very close to the terrain surface (with the drawback of too many images to handle). The proposed improvement is to collect a temporal time-series of nadiral Thermic / Multispectral images collected always in the same sampling spatial points with high accuracy in the positioning. This process will improve feature based on AI algorithms to monitor the plants during all growing phases, but it requires high accurate positioning performance in order to minimize false positive of the AI/ Machine learning techniques.
- Terrain Following: High end productions as grapes for fine wines are typically located on hills. When planning a mission aimed at allowing AI analysis in post processing or spraying fertilizers, it is important to reach each waypoint with a high positioning accuracy not only on horizontal plane, but also on vertical axis. The GNSS based component (with the utilization of European GNSS signal) will improve navigation and position performance also on the vertical axis without additional proximity sensors (ultrasonic, optical flow, LIDAR, etc.) that often present issues above vegetation.

4.1.11 Complex System for Autonomous Drone Battery Management - UWB & SM

The UWB and SM will be jointly developing complex system for autonomous drone battery management. The main component of this system will be device called Droneport (DP). The DP will present an autonomous station with defined communication interface realized via MAVLink messages. The main tasks of the DP will be guidance of the drones for the landing, change or charging of drone batteries and proper resumption of the original drone mission. The DP will communicate wirelessly with the drones and will be able to monitor the request for recharging of the drones. The DP will also assist the drones in landing by providing local positioning system based on visual servoing.

The Droneport HW and SW platform aims are to provide open all-round battery management system that could provide the battery exchange/recharge capability to various drone systems. The battery mounting unit of the DP should make it possible to extend wide variety of drones with system capable of exchange of batteries and use the DP landing assistant. Also, it will provide unified interface for the DP base station battery storage and charging unit. The SW platform will employ standard MAVLink communication technology for the purposes of exchanging the status information between the drones and DP thus easing integration to current flight control systems.

4.1.12 Smart and Predictive Energy Management System - UDANET

An energy management system is vital to optimize the energy life and the purpose of the system. It will continuously monitor important system parameters, while dealing with the varying power demands of the many aspects, the objectives of the mission and optimizing the usage of the energy. The designed energy management system will be verified and tested via Software in The Loop using software Matlab-Simulink. Given the initial and final positions, the objective of the component is to compute the control inputs that rule the motion and vehicle trajectory to optimize energy consumption and the computational burden of the algorithm.

The aim of this activity is to use a model-based approach to control UAV flight to minimize energy consumption. It is therefore necessary to consider the flight dynamics, battery, and energy flow and actuator models. The strategy that led to good results in some research[40] is the optimal control and therefore this type is considered in the activity. A first objective is to improve control by making it robust with respect to model approximation errors or disturbance (for example under wind conditions).

One of the problems encountered in electronic devices in all sectors, and even more so in drones for obvious reasons, is that related to the computational complexity that weighs on the overall processing capacity. The idea to be applied in this case is to use the choice of optimal control to identify common paths in preferences with the aim of obtaining a sub-optimal rule-based strategy which has a reduced computational load.

The idea is imported from the automotive sector, where there is undoubtedly greater calculation capacity on board and there are fewer problems due to weight, and has been successful for controlling the energy flow in hybrid vehicles to minimize consumption and divide the pairs among the possible actuators. Another aspect that can improve control is to study it with respect to external disturbances, for example in windy situations.

4.1.13 AI Drone System Modules - UDANET

UDANET will design and implement an artificial intelligent method to classify leaf diseases. The methods inputs are images from cameras, and the output is the plant health status classification. Artificial intelligence algorithms with different characteristics will be designed and implemented with the aim of taking into account the computational cost of each, as well as the over-fitting problem and the dataset that is not always adequate, and the diagnostic performance is drastically decreased when used on test datasets from new environments (i.e. generalization problem).

A large amount of data increases the performance of machine learning algorithms and avoids over-fitting problems. Creating a large amount of data in the agricultural sector for the design of models for the diagnosis and detection of plant diseases is an open and challenging task that takes a lot of time and resources. One way to increase the dataset will be to use data augmentation techniques. It increases the diversity of training data for machine learning algorithms without collecting new data.

The improving objective of this component is to develop and test the algorithms, increase the reference dataset using basic image manipulation and deep learning based image augmentation techniques such as image flipping, cropping, rotation, color transformation, PCA color augmentation, noise injection, Generative Adversarial Networks (GANs) and Neural Style Transfer (NST) techniques. Performance of

[40] Mirzaeinia A, Hassnalian M. and Lee K. Drones for Borders Surveillance: Autonomous Battery Maintenance Station and Replacement for Multirotor Drones, AIAA SciTech Forum, DOI 10.2514/6.2020-0062, 2020
Liu Z.-N. et. al. An Autonomous Dock and Battery Swapping System for Multirotor UAV, DOI 10.13140/RG.2.2.19437.90085, 2018

the data augmentation techniques was studied using state of the art transfer learning techniques both in terms of accuracy and computational cost.

4.1.14 Generic Mission Controller - SCALIAN

Scalian has worked on developing a generic architecture to allow fleet of UAVs to perform a variety of mission. The architecture allows to use heterogeneous UAVs and to allocate them different missions. For instance, a fleet of 10 UAVs performs logistic operations by dropping sensors over a huge area collaboratively, or a fleet of 4 UAVs swipes scanned areas to build an orthophoto.

Inside the architecture there are several components (from low level like control, up to high-level like task planning). The Aerial Mission Controller (AMC) or Flight Mission Controller (not to be confused with Flight Controller), is the center point and coordinates all the other components. Scalian aims at expanding the Aerial Mission Controller (AMC) into a Generic Mission Controller (GMC or xMC). The goal is to allow our EZ_Chains architecture to be able to operate UAVs, rovers, and other types of autonomous system, for instance intelligent sensors and actuators (e.g. weather station, operation-area access control).

In addition to accommodating more types of autonomous system, it needs to increase the number of agents it can control during a mission to make operations faster but also to account for the new variety of agents. The AMC has been demonstrated in simulations, software-in-the-loop tests, hardware-in-the-loop tests and real operations. The development of the GMC must also use this proof process to ensure a safe system. This constraint is taken into account during the design phase.

In order to increase the safety of operations, the GMC must also integrate new components to work with Air-Traffic Management. When a foreign aircraft enters the airspace, the human operators must add a geofence around it in the geofence or trigger an appropriate mode for the fleet (return to base or landing on alternative zones). It would be better if the GMC could take the ATM into account: it could act sooner and more efficiently. For instance placing a smaller geofence, sending orders only some UAVs in the flight plan, and so on.

4.1.15 Fleet Knowledge Base - SCALIAN

Scalian has developed a component called Knowledge Base, and it is part of the EZ_Chains architecture. It is the component responsible for maintaining a shared mission status inside the system. Agents connect to it and are then able to get and update that status, but also share their own status.

As of now, the system supports two types of agents: UAVs and GCS. The UAVs use the mission status to plan their mission, and report their status. The GCS uses the KB to display the mission progress to the operators and allow them to send orders to the UAVs (e.g. return to base).

The first and main improvement is to allow for more types of agents. The goal is to offer two main classes of agents: mobile agents and static agents. The static ones correspond to hardware and software components that are meant to provide information to the system (e.g. weather station) or actuate with low or no movement (e.g. access-control barrier). The mobile agents represent all the robotic systems, UAVs and UGVs in the Use Cases.

This requires to extend the current definition of an agent in the system and to correlate it to the rights it has on the database (read and write updates and send orders). If the number of agents increases a lot, it is also necessary to setup messages priority in order to sort the most important messages to ensure they are shared before the least important ones.

The other improvement is to connect the KB to an Air-Traffic Management system (ATM) and Unmanned-aircraft Traffic Management system (UTM). This would allow the fleet to take into account aircraft foreign to the system. Furthermore, the fleet could provide its flight paths ensuring a better integration in the airspace removing the need for a segregated airspace for the operations.

4.1.16 Video and Data Analysis Algorithms - AI

Aitek will define and implement a software component during COMP4DRONES. It will consist of video and data analysis algorithms to be used in the scope of the smart agriculture use case. It will be based on deep learning approaches and will be used to process RGB (mainly) and infrared (eventually) images. The results of this processing will be combined with some further heterogeneous data collected by the sensors installed on the drone.

Drones properly equipped with a camera and a computational unit can be used to collect and process images to detect relevant information in real time. Starting from a consolidated experience in video processing, Aitek contribution will be focused on the application of these technologies in the context of smart agriculture to monitor crop and plant status.

Deep Learning algorithms can detect several different information, after a proper training phase during which they learn what they must be able to do. Two main functionalities currently under definition are: Enable precise tree crowns definition for water management and harvest forecast, and Monitoring and processing to detect exact location of un-healthy crops/plants. Both functionalities can be done either on-board in real-time either in post processing modalities on the ground station.

Aitek has a large experience in video processing in particular for what concerns the use of state-of-the-art approaches based on deep learning. During this project, particularly in the scope of the smart agriculture use case, we are working to improve currently available solutions, targeting the following objectives:

1. Algorithm optimization in order to find a best compromise between performance and computational resource needed.
2. Integration of heterogeneous information (GPS position, altitude from the ground and other data) collected by the onboard sensors
3. Chance to enlarge specific datasets to rely on for future applications and reuse of data
4. Field testing to identify which solution is best-fitted for this market segment as for HW (i.e. IR-NIR cameras, hyperspectral sensors)

4.2 Payload Technologies

The drone system includes a set of payload technologies to support the drone mission specific operations. The payload technologies that will be the project focus includes: hyperspectral camera, camera, and avionics encoder.

4.2.1 Hyperspectral (HSI) Cameras - IMEC-BG

Hyperspectral cameras as a payload can improve detection of material imperfections. The hyperspectral cameras will be based on IMEC-BGUAV platform: (dual) mosaic sensors/cameras with Ximea breakout board and Jetson TX2 board. Regarding the software blocks, we will reuse Airobot's server-based interface for ground controller with IMEC's camera commands.

IMEC-BG has developed a unique integrated hyperspectral filter/imager technology, where the spectral filters are monolithically deposited (integrated) on top of CMOS image sensors at wafer level. The materials of the filters are chosen such that they are compatible with the production flows available in most CMOS foundries. This is achieved using a set of CMOS compatible production steps, like deposition, patterning and etching, which allows pixel level accuracies in filter alignment. The result is a compact and fast hyperspectral imager made with low-cost CMOS process technology.

4.2.2 Single Visible, Infrared and Dual HD Camera - INDRA

The payload of the Mantis system has been designed by Indra and consists of 2 main subsystems: the camera and the target system. In this case, three different payloads and lenses will be installed in the camera subsystem: visible optics, infrared and dual optics with HD resolution.

The improvements to be made is to have single visible, infrared and dual payload HD will be the replacement of the current camera with analog resolution with a COTS camera with HD resolution with weight and dimensions similar to analog. To ensure its feasibility, once the candidate cameras have been analyzed, the camera's fixing system must be checked and all the internal wiring redesigned to allow the digital video to be obtained at its output. Finally, it must be ensured that the weight and the center of gravity of the total payload is within the established parameters. It will avoid, as far as possible, having to redesign the fairing of the payment charge.

4.2.3 Avionics Encoder - INDRA

Encoder for the reception, treatment and parameterization of the video received from the four HD-optics and tracking features will be developed. The MANTIS System video encoder is a next-generation COTS element designed for miniUAVs due to its reduced weight and size versus its high performance

The encoder will be enabled for the reception and treatment of HD video from the four new HD-optics for which it will be necessary to parameterize them taking into account the characteristics of each of them. In addition, tracking, settings, configuration and tests will be performed to improve the current tracking features available for SD video by applying them to the new HD video.

4.3 Hardware Platform

As an execution environment, the project will provide improvements including onboard programmable and reconfigurable compute platform, system on module (SoM) module, heterogeneous systems-on-chip (SoC)-based reference platform, etc.

4.3.1 Onboard Programmable and Reconfigurable Compute Platform Design Methodology - UNIMORE and UNISS

The demand for onboard computational power of modern drones is exponentially growing due to the ever-increasing request for autonomous operation. To satisfy this request, more powerful (in terms of operational throughput) compute platforms must be embedded on the drones, while at the same time maintaining operational constraints related to the power envelope, the form factor, and the interoperability and connectivity with standard drone software stacks.

Modern heterogeneous SoC tackle this challenge by exploring options that combine traditional CPU scalar processing with VLIW/SIMD processing (DSP, GPU) or FPGA logic. FPGA-based, commercial-off-the-shelf (COTS) SoC such as Xilinx’s Zynq, Ultrascale+ or Versal, combine the flexibility of a general-purpose, multi-core ARM CPU, to the efficiency of custom-designed hardware IPs. The peak performance, verifiability and certifiability of FPGA design flows make these devices a very appealing candidate for an onboard drone compute platform.

However, the more heterogeneous components are integrated into any compute platform, the more complex the integration process becomes in terms of hardware-software interactions, access to shared resources, and diminished regularity of the design. The critical challenges are in the integration with the legacy software components, the programming interfaces, and the management of many heterogeneous components more than in the design of any individual components.

The proposed solution for a low power, high performance onboard drone compute platform consists of a FPGA-based design methodology rooted on three key components:

- An Open-Source FPGA overlay based on a compute cluster integrating RISC-V cores and low-latency, high-bandwidth shared memory as a standard interface to application-specific hardware processing elements (HWPE). This enables plug-and-play deployment of HWPEs in typical drone workloads [UNIMORE].

- A methodology and tool (the Multi-Dataflow Composer, or MDC) for the design of Coarse-Grained Reconfigurable Co-processing Units, which constitutes the basis for the definition and integration of HWPEs in the overlay compute clusters [UNISS].
- A high-level programming model, with associated runtime and tools for the offloading and local orchestration of the HWPEs. This last component is developed as part of WP6 and detailed in D6.1 [UNIMORE-UNISS].

Joint efforts from UNISS, UNIMORE and UNIVAQ will contribute to the development of the proposed compute platform for autonomous drones, targeting ease of deployment of FPGA accelerators and programmability of the whole platform. Within the context of WP3, the proposed technology is defining an acceleration architecture template and infrastructure compliant with the overlay concepts and easily adoptable by HW accelerator developers, capable to answer to the need for embedding ever-increasing computational capabilities on SoC. Extensions/adjustments of the described component are needed to integrate the CGR accelerators within an overlay infrastructure.

For the acceleration architecture infrastructure, we propose solutions for drone design that leverage commercial off-the-shelf, FPGA-based heterogeneous SoCs (e.g., Xilinx Ultrascale+) for the deployment of a specialized (according to the needs of the service being deployed) instance of a cluster-based, many-core accelerator architecture template. Putting together UNISS and UNIMORE already available components, the building block of such a soft IP is meant to be a compute cluster, where several simple processing elements share local memory, to which dedicated HW accelerators can also be attached (e.g., for deep learning). HW accelerators are intended to be, where needed, CGR reconfigurable ones, implementing different working points (i.e. trade-offs among Quality of Service and Energy consumption) or functionalities (i.e. different convolutional kernels) on the same configurable substrate as discussed above.

Note that, while in the WP3 the contributions focus on the design methodology of onboard computing platforms, based on FPGA SoC, the evaluation of the KPI can be measured only within the complete UC definition, and thus taking in account also the complementary contributions coming from WP4 and WP6. First, cooperation in WP4 will allow us to assess the proposed accelerators and the integration in computational hungry scenarios by developing an accelerator for AI/ML applications.

Second, cooperation with WP6 will tackle mainly the second SoA open issue discussed above by investing in automated strategies for the generation of these kinds of co-processing units and for task offloading.

4.3.2 HW/SW System on Module - IKERLAN

IKERLAN will develop a HW/SW SoM based on a heterogeneous SoC such as XILINX ZU+MPSoC. That will enable to the HW/SW co-design of the system and accelerate functionalities in drones to improve its performance.

IKERLAN will develop both HW (SoM board IK-SCC-SOM) as well as the SW and FPGA code to implement accelerators of a variety of functionalities such as objet detection and positioning. The SoM module will take into account arising standards in the drone industry for the safety certification and will produce a safety concept to prove the certifiablity of the SoM module of a selected flight safety function.

No implementations on HW/SW have been done yet of components that follow those safety regulations. Therefore, IKERLAN will analyze the HW and SW requirements for a drone module to match those arising standards and an architecture proposal and actual implementation will be done with the techniques to be implemented in order to achieve desired robustness level. Also, an implementation of those safety functions will be done in one ZU+ MPSoC processor that has a higher level of capabilities to run faster and more efficiently that traditional companion computers.

4.3.3 Modular SoC-based embedded reference architecture - EDI

EDI will develop an embedded heterogeneous SoC-based (FPGA + Memory Protection Unit (MPU)) reference platform, which will be used to deploy components developed by EDI and other partners. The main purpose of this platform is to enable software and hardware (FPGA) co-design and deployment on a functional drone. Furthermore, the architecture will include the linux-based board support package, software component intercommunication framework, methodology for algorithm partitioning between sequential and parallel processing paradigms and examples on setting up DMA-based communication with the FPGA accelerators.

To accelerate the electronics hardware development, an embedded SoC module will be utilized and a special carrier board will be designed and manufactured to deploy heterogeneous platform on the drone. EDI will enable:

a) Distribution of algorithms across heterogeneous processing paradigms.
b) Novel approaches to sensing and processing pipelines.

4.3.4 CompSOCHardware Platform - TUE

In a multi-processor System On Chip (SoC), applications with different requirements are executed on a heterogeneous set of resources and communicate through an interconnect such as Network On Chip (NoC). Proposing a SoC whose temporal behavior is easily predictable has been under research for a long time.

As TUE’s planned contribution in COMP4DRONES project, an improved version of the CompSOC[41] platform implemented on FPGA will be utilized in the reference architecture of the drone systems. This improved version uses a commercial implementation of the basic CompSOC hardware platform by Verintec, together with a predictable ROS stack made by the TUE.

The overall hardware architecture improvements planned in this project is to support predictable and composable ROS2 software stack and can be summarized as follows:

- Migration from MicroBlaze to RISC-V processor due to more complex instruction set and real use cases of RISC-V in industry compared to limited domain for MicroBlaze.
- Improving memory controller in terms of predictability and performance by integrating reliable and secure communication protocols such as c-heap.
- Integrating I/O ports like UART to the platform to be utilized by sensor devices in Robotic domain.
- Reliable and transparent communication channel with the world outside based on an agent-client implementation of Data Distribution Service (DDS) standard. This will help drone to drone communication.
- Improving monitoring tools such as channel monitoring for system debugging and troubleshooting.

4.3.5 Highly Embedded Customizable Platform for SLAM technique - Modis

SLAM (Simultaneous Localization and Mapping) techniques are considered to be a mature field, and nowadays there are a many open-source systems that are able to deliver fast and accurate estimation in typical real-world scenarios. The actual problem is the lack of a modular architecture (involving hardware and firmware configurations) that can deal with heterogeneous sensors’ configurations and that can provide an easy-to-use and easy-to-adapt, extensible and reusable base configuration package.

The system will be equipped with the following sensing capabilities: magnetometers, compasses, GPS (optional), a gyroscope and accelerometers. Additionally the board will be provided with a serial

[41]Goossens, et al. "NoC-Based Multiprocessor Architecture for Mixed-Time-Criticality Applications." Springer (2017): 491-530.

communication bus (or more than one), e.g. USART, SPI, MavLink and a 32-bit MCU. In addition, the modularity will be improved with respect to ad hoc drone HW supporting only mission specific tasks.

4.3.6 Efficient Digital Implementation of Controller on FPGAs - UNIVAQ

In order to operate safely and autonomously, the controller shall enable new functionalities, such as collision avoidance, navigation, situation awareness, etc., while ensuring the basic control actions. To ensure shorter execution times of the control algorithm, FPGAs can be used. To further reduce the execution time in FPGAs, some techniques (retiming/pipelining, folding/unfolding, interleaving, etc.) can be used that allow transforming the circuit structure, in order to reduce this time, and possibly the power consumption, maintaining the desired functionalities.

The expected improvements are the increased performance of the controller, with reduced response time, and optimal balance between performance, power consumption and weight, and increased resiliency of the drone during operations.

4.4 Tools

To support the development of drone systems, a number of tools need to be developed. In this section, we describe the tools that will support the architecture implementation.

4.4.1 Mixed-Criticality Design Space Exploration - UNIVAQ

The main goal is the integration of an existing design space exploration approach that takes into consideration mixed-criticality system constraints, introducing and considering the hypervisor scenario, into the COMP4DRONES framework. UNIVAQ will exploit a specific software component able to (semi)automatically generate application partitioning, mapping and architectural definition by means of search methods and simulations in the design space exploration activities.

Improvements inside the project will be related to a SystemC timing simulator that checks F/NF requirements fulfillment, and to the hierarchical scheduling that allows to simulate the use of hypervisor technologies. The main goal is to reduce simulation time and design space exploration execution time, while keeping bounded the accuracy and the errors associated with the estimation values. The work is directly linked to the WP6 tool called HEPSYCODE.

4.4.2 Reference Architecture Modeling - CEA

The current embedded architectures of drones are organized in loosely coupled monolithic boards. Each board composed of processor, memory and communication resources (e.g. flight control, planning boards). This separation ensures that the subsystems operate almost independently from each other and avoids interference coming from the other parts. However, this approach does not support the continuous development of drone applications such as the ever-increasing demand on autonomy. The COMP4DRONES project will face this challenge by developing a compositional and integrated drone embedded reference architecture following the IMA principles, adapted to, and still considering, the drone resource constraints. With the support of WP6 tools, CEA will enable the modeling of the compositional and integrated drone embedded reference architecture.

4.4.3 Paparazzi Autopilot - ENAC

Paparazzi is a complete open source system for Unmanned Aircraft System (UAS) including both airborne autopilot as well as complete ground station, mission planning and monitoring. The ground segment uses a software bus (the Ivy bus) to exchange information, making it easy to add custom software or replace software.

The airborne segment is made for multiple vehicles, like fixed wings, rotorcrafts, but also rovers. It is highly configurable and one can choose between multiple alternatives for control algorithms, navigation algorithms, sensors, target platforms. Custom modules can be added to extend the capabilities of the system. The airborne code is generated by the Paparazzi build system according to the chosen configuration. This architecture allows the generation of efficient platform-specific code from a platform-agnostic configuration.

The objective for this component is to improve the validation of code modifications, in particular by applying the outcome of the tools and methods from WP6.The internal validation will focus on improving the code coverage with automated compilation and execution of static analyses. In addition, a specific effort will be made to deploy a simulation framework that can be used from other partners to simulate multi-UAV operations with a realistic flight model in order to validate external packages. Potentially hardware-in-the-loop simulations are also possible. The objective is to reduce the time required to prepare a test case simulation for validation.

4.4.4 Integrated Systems Simulation - Siemens

Siemens' SimcenterAmesim is a software tool dedicated to modeling and simulation of dynamic and multi-physics systems. In the tool environment, systems are modeled connecting components available in the libraries, which cover several physical (fluids, mechanical, electric) and application (aerospace, automotive, gas turbine) domains. The libraries components' interface is realized through ports that allow the flow of physical variables. The contributions the tool can bring to this work package are listed here below:

a) Physical behavior modeling of drones' platforms for dynamic performance simulations. These components can be modeled and validated separately.
b) Integration of the aforementioned models to build an integrated model of a drone platform for performance simulations. Such a model can predict the performance of the drone before it is manufactured.
c) The physical plant model created with Simcenter Amesim can be used to calibrate and validate controllers and autopilots with Software-in-the-loop or Hardware-in-the-loop.
d) By means of dedicated interfaces with third tools, it is possible to couple the model obtained (plant and controllers) with software which provides sensors and environment modeling capabilities for environmental awareness simulations. This would give the ability to validate detect and avoid algorithms.
e) The model consisting of the plant and the controllers (autopilots) can then be used to evaluate the performance of the drone and the controllers' robustness in case of failure events

5 Conclusions

In this deliverable, we have provided an initial architecture that we going to be adopted in the project. To do so, first, we have reviewed existing autopilot architectures as well as embedded software architectures in the domains of avionics, automotive, and robotics. This state of the art provides useful insights in specifying the COMP4DRONES architecture.

Second, based on the current state of the art architectures, best practices, and needs/requirements for drone architecture design, an initial architecture that will adopted in the project is proposed.

Finally, to ease the customization and integration of drone systems, a number of generic components will be developed. These components are divided into four groups: system functions (e.g. data processing algorithms, computer vision components, indoor-positioning, and SLAM algorithms), payloads (e.g. hyperspectral cameras and avionics encoders), hardware platform (e.g. onboard programmable and reconfigurable compute platform, SoM module, and heterogeneous SoC-based reference platform), and tools (e.g. system simulation, reference architecture modeler, and design space exploration).

Appendix A: Details of the Architecture Generic Components

To ease the customization and integration of drone systems, a number of generic components will be developed together with the architecture. These components are divided into four groups. These groups are: system functions, payloads, hardware platform, and tools. In the following, we describe them in detail by giving each component description, state of the art, and expected improvement during the project.

6 System Functions

The drone system functions are the core functions required for the drone to perform its flying stages in safe and efficient manner. The generic components in this category includes: data processing algorithms, computer vision components, indoor-positioning, SLAM algorithms, etc.

6.1 Sensor Data Processing Algorithms - BUT

6.1.1 Component Description

Description: In this component BUT will implement/improve sensor data processing algorithms which will include software and firmware for FPGA. This will involve video processing algorithms (for example HDR algorithms). HDR multi-exposure fusion algorithm to be implemented in the drone, possibly implementing also tone mapping and/or ghost removal (with most probably somewhat limited capabilities) in order to "feed" further image and video processing subsystems in the drone by image information with high dynamic range. As a "demonstrator", we can also provide e.g. object detection in HDR.

Keywords: HDR, FPGA, HDR Deghosting, HDR Tone Mapping methods

6.1.2 State of the Art

HDR is a technique used to reproduce a greater dynamic range than is typically possible with standard image capturing methods. The goal is to capture the range of brightness as perceived by the human eye. HDR video acquisition is a very important feature of modern surveillance, traffic monitoring, and other applications. Typically, for both economical and technological reasons, the HDR image video is acquired using multi-exposure using sensors with limited dynamic range. Algorithms for single HDR image processing are known quite well, but these algorithms are currently feasible on PC platforms and specialized accelerators. Real-time acquisition and processing of HDR video is therefore still quite opened and challenging topic. The integration of a high-quality algorithm for HDR image acquisition directly in the UAV will enable the application of advanced computer vision methods also in difficult light conditions.

Multi exposure approach (multi exposure bracketing) produces high quality HDR images only for static scenes. Any change in the scene between each captured image (or any camera motion) results in ghost artifacts in the resulting HDR image. Therefore, to pro-vide HDR acquisition without adverse effects it is necessary to use a suitable deghosting method. While deghosting is being researched for a long time, the state of the art methods that have good visual results are very computationally demanding; therefore, they are impossible to be implemented in embedded systems attached to UAVs.

6.1.3 Improvements

Our HDR pipeline is implemented in FPGA and HDR image acquisition pipeline is composed from three main blocks: image capturing, HDR merging and tone mapping. The image capturing part is grabbing the images from the sensor. HDR merging processes multiple images (in our architecture three) into the HDR frame. The tone-mapping block is compressing the high dynamic range into the standard, 8-bit image while preserving the details from HDR. This output can be fed into following computer vision algorithms. In the COMP4DRONES project we will improve HDR fusion algorithm to be suitable for this application and also improve latency and throughput.

6.1.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Ease of integration (software)	• Number of lines of code • Time spent	SC1.2, SC4.1	O1, O4
Ease of configuration	• Number of user parameters • Time spent for the initial setup • Configuration time spent per flight	SC1.2, SC4.1	O1
Ease of incorporating new hardware (FPGA) components	• Number of lines of code necessary to communicate with the accelerator	SC1.1, SC1.2	O1

6.2 Computer Vision Components for Drones - HIB

6.2.1 Component Description

Description: The computer vision component for drones is an AI system built upon deep learning techniques in order to improve the way of interpreting surroundings and detecting scenarios from data captured with drones. The component will carry out a post-processing of the captured images with the drone cameras.

Concretely, the computer vision component is envisaged as a system for the post-processing of images captured by drones that will be used in the UC2-Demo 1 Digitalization of the state of the constructive process of a Civil Infrastructure, with the aim of understanding the captured scenario by drones in civil infrastructure. The AI system will be built upon automatic algorithms for the auto-detection/geo-referencing of road elements and taking cloud of points generated by a LIDAR sensor.

More in detail, the component is composed of two convolutional neural networks that will be trained with a cloud of points generated through:

- Simulated data aggregator, developed in WP4 where the cloud of points will be generated from the scenario and drone parameters pre-defined.
- Real data from some initial drone data collection campaigns.

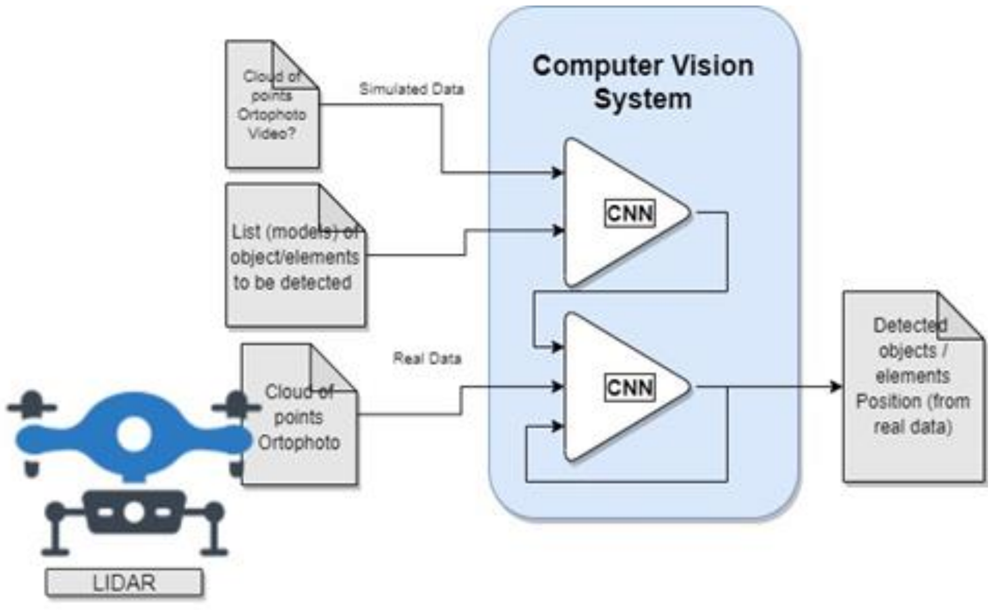

Cloud of points Ortophoto Video?
Simulated Data
Computer Vision System
List (models) of object/elements to be detected
CNN
CNN
Real Data
Cloud of points Ortophoto
Detected objects / elements Position (from real data)
LIDAR

Figure 38: Base architecture of the component

The trained CNN will be cloned and applied to the captured drone images in order to recognize the scenario based on the trained dataset and so provide the expected output for the recorded scenario (detected objects, position…). Following these details, the base architecture of the component is available in Figure 38:

As it can be observed in the figure, the input data for the module will be the cloud of points provided in ".las" format generated with the simulated and real data, and the list of elements to be detected in CAD format. After the processing of this data, the component will provide as output the set of elements detected and their position in the captured terrain to serve as input for accelerating the construction of the civil infrastructure.

Keywords: image processing, computer vision,

6.2.2 State of the Art

Besides conventional post-data processing system, innovative ways are used in extracting information. Machine learning and deep learning are an arising approach in dealing with large amount of data gained from drones[42]. For infrastructure planning and design, typical data acquired through drones are images. 3D geometrical models can be generated from these images through manually method or semi-automated algorithms. For construction monitoring, either real time videos or 3D models are needed. As for infrastructure inspection, machine learning and deep learning algorithms are employed. Gopalakrishnan et al. proposed pre-trained deep learning models for crack damage detection in UAV images[43]. Kang and Cha successfully detected concrete cracks with 97.7% specificity and 91.9% sensitivity from UAV video[44]. Additionally, for object detection, Arifin et al[45] present a method of detecting and tracking objects with specific shapes and color in drone flight. This method applies some algorithms such as Hough transformation to the images taken by drones. In addition, Maria et al[46] propose a method for vehicle detection from videos captured by drones in an urban environment.

Focusing on object recognition and detection in aerial images captured by drones, a major challenge with the integration of artificial intelligence and machine learning with autonomous drones' operations is that these tasks are not executable in real-time or near-real-time due to the complexities of these tasks and their computational costs. The most accurate modern neural networks do not operate in real time and require large number of GPUs for training with a large mini-batch-size. One of the proposed solutions is the implementation of a deep learning-based software which uses a convolutional neural network algorithm to track, detect, and classify objects from raw data. In the last few years, deep convolutional neural networks have shown to be a reliable approach for image object detection and classification due to their relatively high accuracy and speed[47]. Furthermore, a CNN algorithm enables UAVs to convert object information from the immediate environment into abstract information that can be interpreted by machines without human interference. The main advantage of CNN algorithms is that they can detect and classify objects while being computationally less expensive and superior in performance when compared with other machine-learning methods.

Focusing now on the specific role of this component in the UC2-Demo 1 Digitalization of the state of the constructive process of a Civil Infrastructure, the road elements detection process comprises an

[42]Shakhatreh, H., Sawalmeh, A., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N. S., Khreishah, A., and Guizani, M., "Unmanned aerial vehicles: A survey on civil applications and key research challenges." arXiv preprint arXiv:1805.00881, submitted for publication
[43]Gopalakrishnan, K., Gholami, H., Vidyadharan, A., Choudhary, A., and Agrawal, A., "Crack damage detection in unmanned aerial vehicle images of civil infrastructure using pretrained deep learning model." International Journal for Traffic and Transport Engineering, vol.8, pp.1-14, Nov, 2017
[44]Kang, D., and Cha, Y. J., "Autonomous UAVs for structural health monitoring using deep learning and an ultrasonic beacon system with geo-tagging." Computer-Aided Civil and Infrastructure Engineering, vol.33, pp.885-902, May. 2018
[45]Arifin F, Daniel RA, Widiyanto D. Autonomous detection and tracking of an object autonomously usingar.drone quadcopter. J IlmuKomputerdanInformasi. 2014;7(1):11-17
[46]Maria G, Baccaglini E, Brevi D, Gavelli M, Scopigno R. A drone-based image processing system for cardetection in a smart transport infrastructure. In: Proceedings of 2016 18th Mediterranean Electrotechnical Conference (MELECON); 2016; Lemesos, Cyprus
[47]Sherrah, J. Fully Convolutional Networks for Dense Semantic Labelling of High-Resolution Aerial Imagery. Available online: https://arxiv.org/pdf/1606.02585.pdf (accessed on 8 June 2017).

identification of objects that can be found either on the road or in the road proximity, and many companies therefore are trying to process at least part of this task automatically in order to save time.

Because of the LiDAR point clouds properties, these approaches are widely used to detect sign positions in three-dimensional space. Virtually all methods using point clouds to detect road signs use reflexivity of the sign as a key feature for the detection[48]. However, as González-Jorge et al.[49] points out, road signs can suffer degradation (from dirt, wind, vandalism etc.) or can be concealed (by trees, building, trucks...). It is also possible to gain accurate information about the sign location, shape (including its potential bumpiness), direction and the position of the sign base. Moreover, there are many objects that can be classified as pole-shaped. This group comprises electricity poles, public lights that are placed along the road, in some cases even trees (particularly tree trunks). Detection of these objects is in most cases based on point clouds. The basic idea of pole-like object identification is the detection of points distributed in the horizontal direction as a cylinder. A method that classifies objects into three categories: utility poles, lamp posts and traffic signs is proposed[50]. LiDAR data can be used to extract most of the information needed. The most widely used information is a position of the object. It can be either a position of the centroid or position of the base of the trunk. However, even information such as height of the pole and trunk diameter can be easily extracted from the point cloud that represents the object.

6.2.3 Improvements

The component will go beyond the state of the art by providing automatic algorithms for the autodetection/geo-referencing of road elements. The process will be done thanks to the combination of a computer vision system that will be applied upon a cloud of points generated for a specific scenario captured from a LIDAR camera integrated in a drone.

The innovation expected for the component provided is to analyses scenarios captured during drone campaigns with an in-device camera, for the detection of objects in the images and provide an inventory of elements and their position in the terrain with the objective of accelerating the Constructive Process of a Civil Infrastructure.

6.2.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
High accuracy in the detection of work elements through point clouds	Percentage of work elements correctly detected with regards to the total number of elements	SC2.1	O2
High differentiation in the detection of different kinds of work elements	Number of different work elements detected with regards to the total number of trained elements	SC2.1	O2
Easy integration to any domain	Number of lines of code necessary to adapt the component. Time spent.	SC1.2	O1

6.3 Generic API for Trusted Communication - IFAT

6.3.1 Component Description

Introduction:

[48]Chen, X., Kohlmeyer, B., Stroila, M., Alwar, N., Wang, R., Bach, J., 2009. Next Generation Map Making: Geo-Referenced Ground-Level LIDAR Point Clouds for Automatic Retro-Reflective Road Feature Extraction. GIS '09 Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. 488–491
[49]González-Jorge, H., Riveiro, B., Julia Armesto, P Arias et al., 2011. Geometric Evaluation of Road Signs Using Radiometric Information From Laser Scanning Data. Proceedings of the 28th ISARC. 1007–1012
[50]Yokoyama, H., Date, H., Kanai, S., Takeda, H., 2013. Detection and Classification of Pole-like Objects from Mobile Laser Scanning Data of Urban Environments. International Journal of CAD/CAM Vol. 13, No. 2, 31–40

Within the scope of WP3, IFAT will develop modular and generic software components which are closely aligned with the corresponding hardware-security component which IFAT will develop within WP5. The basic architecture specifications of the IFAT components are briefly summarized in this WP3 deliverable.

The main application purpose of IFAT's modular software components (WP3) and the corresponding security-module parts (WP5) in the Comp4Drones context is using the "IFAT hardware-security component" in addition to the main (general purpose) microcontroller in order to increase the level of security of future drone-identification and to establish secured communication between drone-2-basestation as well as potential drone-2-drone.

Targeted modular security architecture:

The proposed basic concept of the modular security architecture is depicted in Figure 39 and briefly summarized in the following paragraphs:

The most essential underlying part of the concept is the integration of a physically separated hardware-based security component. Either the main microcontroller of the drone or the main microcontroller of the communication system (in our concept denoted as "Host Controller") is connected and interacting with that hardware-based security component – typically denoted as "Secure Element" (SE) or "Hardware Security Module" (HSM). In typical embedded systems, proven inter-chip communication busses such as SPI or I2C are used as hardware interface. Furthermore, corresponding low-level software libraries are developed for the basic low-level communication between Host Controller and Secure Element.

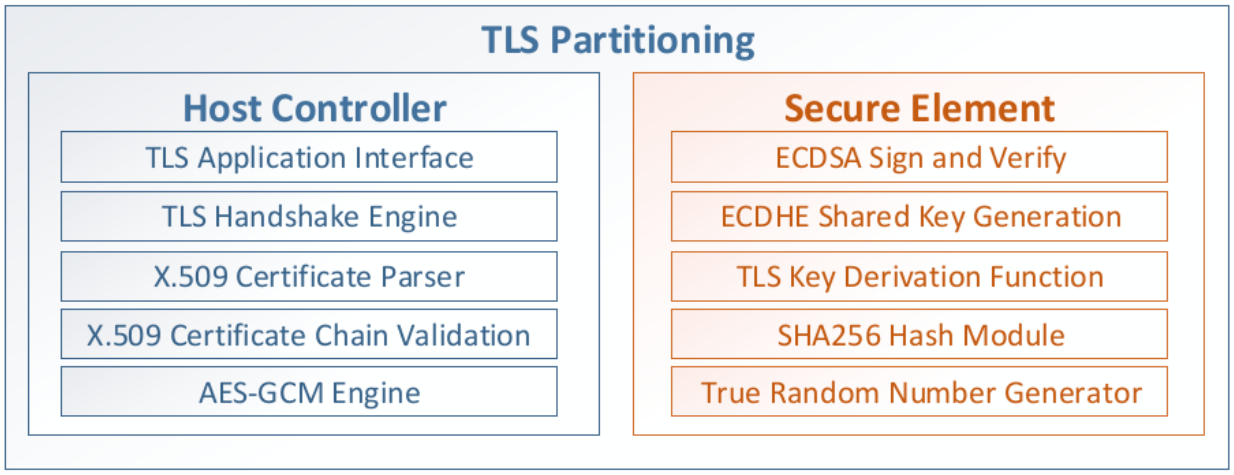


Figure 39: Proposed basic concept of "TLS Partitioning" between Host Controller and SE/HSM

Keywords: Secure Communications

The second part of the proposed concept is the targeted modular architecture of the higher-level software components/libraries for interaction with the Secure Element. These software components should be designed to allow for efficient and flexible functional integration of the Secure Element into the overlying security- and communication architecture of the drone. Therefore, a functional partitioning concept is proposed, which partly splits certain higher security-critical functionalities into the protected "Secure Element" part and keeps less security-relevant parts in the "Host Controller", so that these parts can be updated for reasons of protocol compatibility. In Figure 39, the basic concept of this modular architecture is sketched for the TLS protocol (since this will be the most important protocol for the drone communication).

6.3.2 State of the Art

Standard TLS Handshake (purely software-based):

The Transport Layer Security (TLS) protocol is very widespread and used to protect communication links between two endpoints. TLS is a protocol standardized by the Internet Engineering Taskforce (IETF) capable of providing confidentiality, authenticity and message integrity. For this purpose TLS supports a large variety of cipher suits. A cipher suite is a set, consisting of a key exchange method, an authentication method, an encryption algorithm and a Message Authentication Code (MAC). Cipher suites dedicated for embedded systems with constraint resources are part of the Standard. These cipher suits include message authentication codes that require only low bandwidth. State-of-the-art authentication methods include the Elliptic Curve Digital Signature Algorithm (ECDSA). It outperforms other algorithms such as RSA in terms of execution time. For key exchange, the Elliptic Curve Diffie-Hellman protocol is available. By utilizing this protocol over a public channel both parties generate a shared secret key. The mathematical security of both methods is based on the Elliptic Curve Discrete Logarithm Problem (ECDLP). ECDLP is considered a computational hard problem for that no client solution is known to exist.

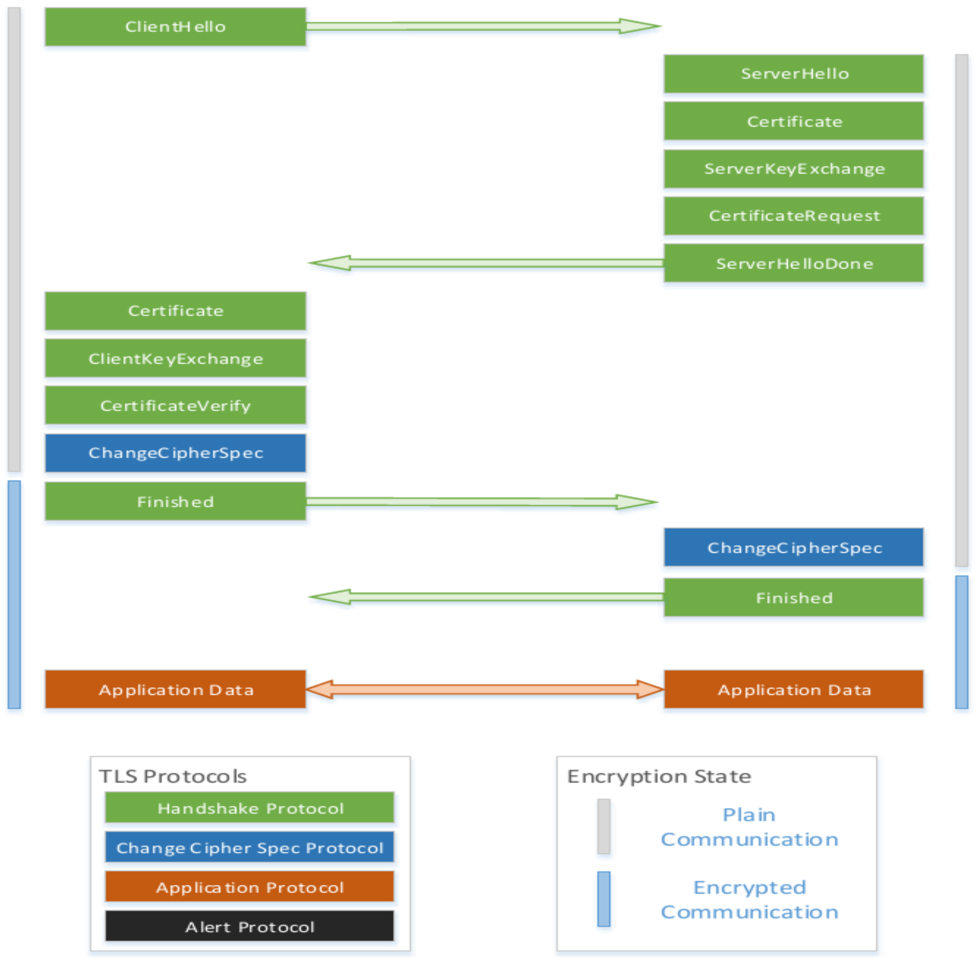


Figure 40: State-of-the-art TLS1.2 Handshake

The TLS protocol works in two phases. In the first phase a handshake is conducted. During the handshake the peers are authenticated towards each other and a set of shared keys is generated with a key exchange mechanism. This state-of-the-art TLS handshake is depicted in Figure 40. In most today's conventional embedded systems, it is typically performed by the main microcontroller purely in software, with the disadvantage that the overall security is still prone to software attacks and physical tempering attacks.

6.3.3 Improvements

The targeted improvements of the architecture proposed by IFAT are:

- Using additional separated hardware-based security component – often denoted as Secure Element (SE) or Hardware Security Module (HSM) – to partly assist the main microcontroller for protecting the higher security-critical parts, such as key generation and storage and according cryptographic primitives.
- On the software side (main microcontroller), for increased flexibility the proposed IFAT software components and libraries will be designed in a modular and partly generic way. On the one hand this concept should allow easier integration into different drone hardware/software-platforms and on the other hand partly allow certain flexibility in the case of future protocol updates.

The corresponding most important metrics to verify if the proposed architecture achieves the goals of improving the overall security of the drone's communication system:

- Is the finally developed drone communication concept less prone to potential software- and physical attacks?
- Are the developed hardware and software components designed in a way that they can be used in a compatible or supplementary way with existing pure software libraries?
- Are the developed software components and libraries designed in a modular way, so that parts of the cryptographic methods can be used either in hardware or software to be compatible with future protocols?

6.3.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Ease of software-integration (of the hardware-security-component)	Are there useful API/C-functions provided (for the main/communication-microcontroller),in the sense that they can be easily used to enhance the Software libraries(e.g. TLS) for partly using the hardware-security-component for security-critical operations	SC1.1	O1
Ease of reuse (for different hardware-security component versions)	Is the provided API designed to be reusable / transparent for different versions of the hardware-security component?	SC1.2	O1

6.4 Control Components that implement Potential Barriers - ENSMA

6.4.1 Component Description

Description: A control loop that implements a "potential field" that prevents the drone to access certain areas by evaluating its position, geofences and potential obstacles in the environment.

It is assumed that drone is equipped with proximity sensors and can detect any relative position of both non-cooperative and cooperative entities within a sensing range. A cooperative entity is another drone which has the same capability. In contrary, a non-cooperative entity is an entity without collision avoidance system.

Sensing capability is required to sense the presence of any other entities in its close vicinity which may lead to a collision. These sensors only give the relative position of any entity in its range in the local frame and do not provide position information in the global frame. It is to note that this assumption is used for the purpose of collision avoidance only.

Keywords: Artificial Potential Field, collision avoidance system

6.4.2 State of the Art

Nowadays drone autopilots, like Ardupilot, PX4, LibrePilot and Paparazzi, mostly use linear control loops that do not take geofences into account, focusing only on stabilization and following a given target.

6.4.3 Improvements

The autonomous flight control includes a mechanism to avoid collision between the drones, obstacles and/or virtual barriers.

An Artificial Potential Field (APF) method is used for collision avoidance. In APF, a drone is considered as a point in a potential field. This drone experiences a repulsion force from the obstacles and therefore, instead of colliding with them, it steers around them. Typically, potential functions are based on the relative distance between drone and the obstacle and do not require any global information.
Based on the practical aspects, an ideal potential function must have the following properties:

- The range of the potential field must be bounded. Usually, it depends on the range of obstacle sensors mounted on the agent
- The value of the potential field and the corresponding repulsion must be infinity at the boundary of the obstacle and must decrease with the increase in the distance
- First and second derivatives of the potential function must exist in order to have a smooth repulsion force

APF based repulsion mechanism is combined with the control algorithm.
Inputs:

- Distance to the obstacle from sensor proximity detection
- Cartesian coordinates and Euler angles of the drone

Outputs:

- Four rotor speed

6.4.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Control loop with collision avoidance	Trajectory tracking despite obstacles	SC2.1, SC2.2	O2
Easy to configure new missions for a fleet	Number of lines of code necessary to support the new mission	SC1.1, SC1.2, SC2.1	O1, O2

6.5 Multi-agent Swarm Control- ENSMA

6.5.1 Component Description

Description: In many practical scenarios, it is required that the agents of Multi-Agent System (MAS) create and maintain a desired geometric shape. The required shape could either be fixed or time-varying. In some cases, it is further required that the agents follow a trajectory while maintaining the shape. The trajectory is produced by a virtual or real leader. This is known as formation tracking.

Keywords: Multi-Agent System (MAS), Control loop, collision avoidance, formation tracking, trajectory tracking

6.5.2 State of the Art

Swarm control is often centralized, heavily dependent on constant communication with a ground control.

6.5.3 Improvements and Metrics

The aim of this task is to develop distributed consensus and formation of a swarm of drones. From the state of art, it is clear that the available cooperative control schemes do not take various limitations into account. Motivated by this, the current task focused on the design and implementation of distributed cooperative control laws for a swarm of drones with communication and sensor constraints. These considered constraints are given below:

- Each drone can only measure its position state
- Drones are not equipped with sensors to measure their velocity
- Drones do not have access to the input (control) of their neighbors
- The measured state is transmitted to the neighbors at irregular and non-uniform time intervals
- The transmission among the drones is asynchronous and totally independent of other drones in the network
- The communication topology among the drones may be directed or undirected
- The connectivity of the communication network is maintained
- Both time-varying and fixed formation shape cases are investigated
- Since inter-agent collision is another important issue in formation tracking control, a potential function-based collision avoidance algorithm is incorporated with the proposed formation tracking controller. The collision avoidance algorithm ensures that the drones converge to produce the desired geometric shape without colliding with each other

Inputs:

- From communication module, Cartesian coordinates of the neighbors
- From internal sensor of the drone, Cartesian coordinates and Euler angles
- Distance to the obstacle from sensor proximity detection

Outputs:

- Four rotor speed

6.5.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Formation tracking	Tracking a planned trajectory while maintain a time-varying formation	SC2.1,	O2
Aperiodic and asynchronous inter-agent communications	Stability of the MAS	SC2.1,	O2
Ease of configuration	• Number of user parameters • Time spent for the initial setup • Configuration time spent per flight	SC1.2, SC4.1	O1, O4
Easy to integrate a new agent to a system	Number of lines of code necessary to add a new UAV	SC1.1, SC1.2	O1

6.6 UWB-based Indoor Positioning - ACORDE

6.6.1 Description

Component Description: The Ultra-Wideband (UWB) Indoor positioning system will enable real-time, accurate position to a tag in an indoor scenario. The integration of the tag on a drone enables its safe navigation on indoor scenario, where other accurate location sources are not available.

It is based on the estimation of accurate ranges from a mobile tag to statically deployed anchors. The mobile tag can either provide the ranges to the anchors for being externally fused with other data sources (default use) or its auto-calculated position. Ranges estimation will be based in this case on ultra wideband technology and Two-Way ranging based location.

Keywords: (Indoor) Positioning, Geo-Location, UWB

6.6.2 State of the Art

UWB location is a technology of growing interest. While the technology is not new, its relevance is starting to be apparent for the consumer market with the recent announcement of the U1 chip on the iPhone 11[51]. In this context, the UWB technology provides "spatial awareness", which is explained as "the ability for your phone to recognize its surrounding and the objects in it.

As mentioned, the technology was known, at least for a more academic point of view. UWV[52] consists in the sending of ultra-short pulses (in the range <1ns) which are sent over large bandwidth (>500M), in the range from 3.1 to 10.6GHz, using a very low duty cycle. This results in lower power consumption. The technology was used in origin for short range communications. A specific interest of this technology for indoor localization was due to its resilience to interferences. The short pulse durations provides also resilience vs. multi-path (once the main path can be better identified). Among the several existing location techniques (Fingerprinting, TWR, TDoA, AoA, UWB technology, Phase of Arrival), time measurement-based techniques, e.g. TDoA and TWR, or some derivative are typically employed.

There are challenges to be faced with lighter or higher intensity depending on the scenario The UWB technology is that it is very "line-of-sight", i.e., not good to traverse walls of obstacles in the larger frequencies.

Competing technologies based on images (visible or near-visible) and SLAM techniques let provide an interesting alternative. A main benefit argued is their independence from anchor deployment. However, the scenarios as the one of indoor construction demo of C4D (UC2. Demo 2) is also challenging for these technologies (bad illumination, varying temperature conditions, homogeneous surfaces and lack of references) to make them fully independent from specific "landmarks" suited on that technology. In any case, they are interesting technologies in their own, and indeed, this combination will be also tested in UC2, demo2 (ACORDE collaboration with FADA-CATEC). For this, is specifically relevant the enabling of specific interfaces which allow effectively and efficiently merge both technologies.

It is also mentioned that the low progress on UWB standard development has limited the use of UWB on consumer products. However, its undoubtable benefits and the availability of very affordable chipsets, e.g. the Decawave DW1000 and the DWM1000 module[53](shown in Figure 41), and their encompassing evaluation kits and documentation, has enabled the blooming of newer solutions and companies[54], and boost applications from 2D location on 2 D location "conventional" applications (e.g. museums), to other domains (e.g. sports[55], industry, etc.).

[51]https://www.computerworld.com/article/3490037/ultra-wideband-explained-and-why-its-in-the-iphone-11.html
[52]F. Zafari. A. Gkelias, K.K. Leung. « A Survey of Indoor Localization Systems and Technologies » .Jan, 2019.
[53]https://www.decawave.com/product/dwm1000-module/
[54]Pozyz website. https://www.pozyx.io/?ppc_keyword=%2Buwb&gclid=EAIaIQobChMIj6fP2cbX6QIVAofVCh3gCwOJEAAYASAAEgL6x_D_BwE
[55]Eliko website. https://www.eliko.ee/products/kio-rtls/?gclid=EAIaIQobChMIhfXDj9DX6QIVzrLVCh0HYww4EAAYAiAAEgLoA_D_BwE

Figure 41: Cheap UWB modules are allowing the blooming of specific developments of UWB location solutions for new domains and applications.

In any case, the richness of different indoor scenarios (flat, a factory, an industrial ship, a tunnel, buildings, parking, supermarkets, etc.) and application needs (several tags, accuracies, required sizes, wiring and powering possibilities, configuration needs, etc.) leads to conclude that there is not an optimal specific IPS solution, nor specifically for a UWB IPS. The deep knowledge of each scenario and, within the UWB presumption, also the knowledge and analysis of the available location techniques, anchor and tag platform alternatives, communication protocols, and other applicative aspects, requires a richness of knowledge and developers able to customize this technology to the specific needs.

6.6.3 Improvements

The UWB IPS to be provided by ACORDE will be focused for providing an affordable solution specifically designed for real-time location on long indoor infrastructures (e.g. tunnels, mines, long corridors).

The solution is being designed for being specifically oriented to 3D location (essential for drone navigation) in the tunnel scenario, but with the aim to offer geo-location (to let the drone navigate through a geo-located predefined path).

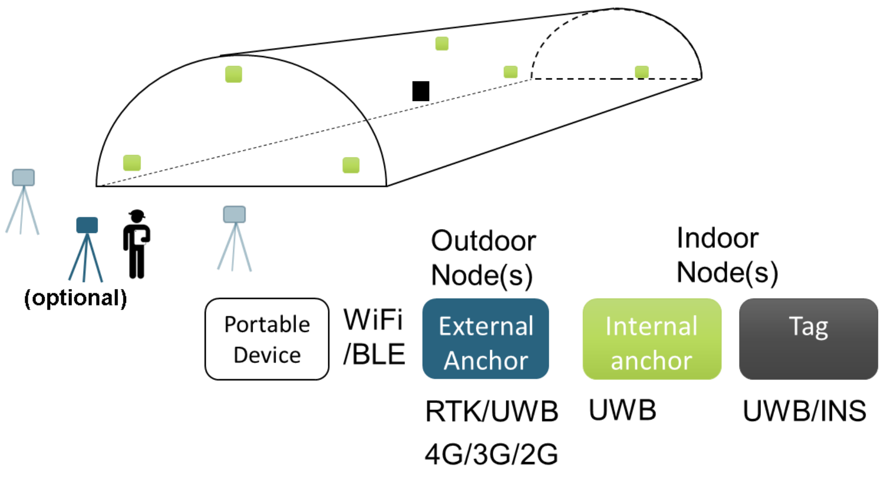


Figure 42: Basic approach to the UWB-based IPS developed by ACORDE.

The solution is being also designed to be versatile enough to allow that the tag, as well as processed geo-location can also provide raw data, i.e. ranges to beacons in range, so that a third party location solution can fusion these data with other sources (e.g. image, thermography, etc.), as it will be done in the construction (use case 2), demo indoor (2). Therefore, this is a specific feature to ease the integration with other technologies. To this respect, the specific output protocol to send this raw data is to be defined. In any case, this design will also consider the developments and interface considerations.

The UWB IPS will also enable the auto-positioning of the beacon deployment, supported by the possibility to communicate with RTK-based anchors in the entries to the indoor infrastructures, and by eventual geolocation of some inner fixed anchors.

While the aforementioned features will depend to a great extent on the algorithms being specifically developed in C4D (in WP4), and on the efficient and robust protocols developed in WP4 (T4.2), a determinant factor will be the anchor and tag platforms. The solution proposed and sketched in Figure 42, shows the need to consider not only the tag-anchor specialization, but also the distinction among internal and external anchors. ACORDE will aim a specific custom design for them on WP3, with the main objective to achieve an optimum solution considering typically contradictory sub-objectives:

- Bounded cost for the solution (taking as a reference tunnels 500m-1000m long)
- Minimization of tag and anchors power consumption for long battery lifes.
- Minimization (or elimination) or wiring of anchor deployment

- Configuration interfaces for easy user-based configuration and calibration.
- Versatility for easily support remote firmware upload and update.

This versatility is another aspect of interest to be dealt with. The specific drone location (for digitization mission) will take a short time slot on the overall weekly work plan of tunnel construction. In other time slots, anchor deployments can be exploited for other locations. However, this type of location has a different type of requirements and challenges which justify a very different profile and that can justify the reconfiguration of anchor functionalities.

6.6.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Ease of integration (hardware)	<u>Tag:</u> • Number of mechanical pieces required for the integration • Need of custom mechanical design • Time spent for integration • Need of connector adapters • Need of voltage adapters • Need of additional power sourcing <u>Anchor network:</u> • Number of anchors deployed (per km) • Number of mechanical elements (per anchor) • Time spent for integration (per km)	SC1.1, SC1.2	O1
Cost of integration (hardware)	<u>Anchor network:</u> • Number of anchors deployed (per km) • Cost of mechanical elements (per anchor)	SC1.1, SC1.2	O1
Ease of integration (software)	• Number of lines of code Time spent for the integration	SC1.2, SC4.1	O1, O4
Ease of configuration	• Number of user interactions required for initial anchor deployment • Configuration time to add/removing/re-configure a fixed and a removable anchor • Configuration time for initial and successive deployments (flights)	SC1.2, SC4.1	O1, O4
Indoor positioning performance	• Horizontal & vertical positioning error (compared against a known good reference)	SC2.1 SC2.2	O2

6.7 Position and Attitude Estimation - ACORDE

6.7.1 Component Description

Description: The position and attitude estimation system of ACORDE provides a trustable positioning and attitude solution. It is based on a multi-antenna/multi-receiver approach and the implementation of sensor fusion with different kind of sensors like accelerometers, gyroscopes and barometer.

The component supports an extensive set of parameters whose values can be adapted to the characteristics of an UAV scenario and its expected dynamics.

The integrity and accuracy of position and attitude estimates output enable to feed reliable and accurate navigation data to a drone autopilot, for its safer operation.

In addition, an offline trace of this position and attitude data synchronized with the GNSS absolute time base can be exploited by a digitization application.

Keywords: (Outdoor) Positioning, Geo-location, GNSS, Sensor-fusion

6.7.2 State of the Art

ACORDE has in portfolio a family of geo-position and attitude estimation systems. They allow the customer to select the most suited performance/cost configuration, i.e. accuracy on position and attitude vs. cost of the solution. For this ACORDE has developed a modular solution for the HW/SW platform.

HW Integration

The appearance of a GLAD Dragonfly+ unit is shown in Figure 43, where the stacked boards that compose the device are distinguished. The Dragonfly+ unit is the best performance configuration of the GLAD family, which in deed demands all platform modules. There is a basic GLAD system (GNSS and sensors board plus a central computing board), plus an RTK add-on (a communications board with an RTK module board) plugged into its bottom side.

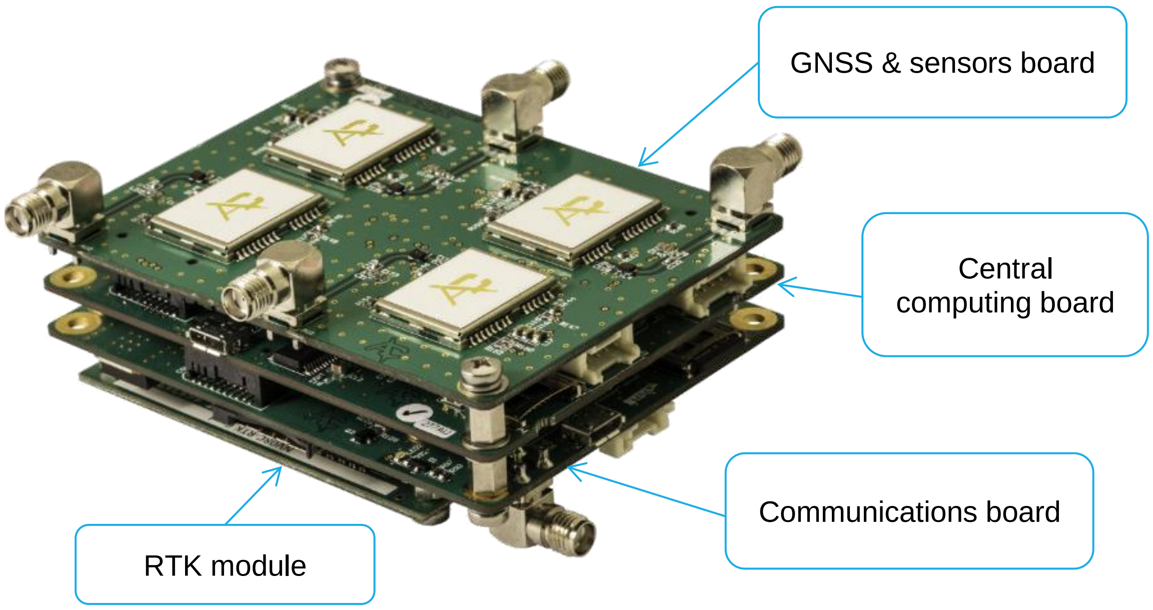


Figure 43: GLAD Dragonfly+ unit.

Physical integration:

For its integration in the drone platform, it has the following main specification:

- Dimensions: 90 x 70 mm x 40 mm
- Weight: 190 g
- Physical integration: 4 fixing points (M3)
- 32 VDC with consumption <5 W

As well as the size and weight, factors, the integrator needs to consider the connection of up to 4 GNSS antennas at least. When the RTK module is used, a 1 GNSS antenna and 1 GSM antenna are also required. Some basic instructions have also to be followed on the GNSS antenna set-up (e.g., using ground plain, keep them in the same plane with clear sky visibility) and recommendations on the location of the GLAD platform.

HW interfaces:

For the integration, the existing HW interfaces are shown in Figure 44.

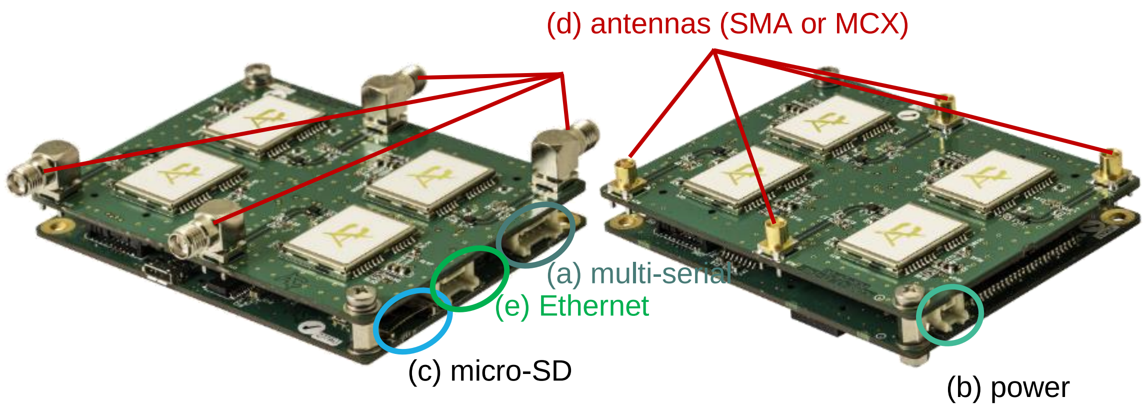


Figure 44: GLAD board interfaces (basic version).

These interfaces correspond to:

- Multi serial port connector with:
- User serial port (3-wire RS232): monitoring & configuration
- Console serial port (3-write RS232): for debug & diagnostics
- Power connector: 6-32 VDC
- Micro-SD card connector for the micro SD card with the GLAD firmware
- RF antenna connectors for the GNSS antennas, labeled from 1 to 4 on the serigraphy of the board for easy identification. By default, SMA connectors are mounted.
- Ethernet connector: for debugging purposes

Figure 45 shows the extra interfaces of the “plus” versions (corresponding to the RTK add-on, i.e. related to the communication and RTK module boards):

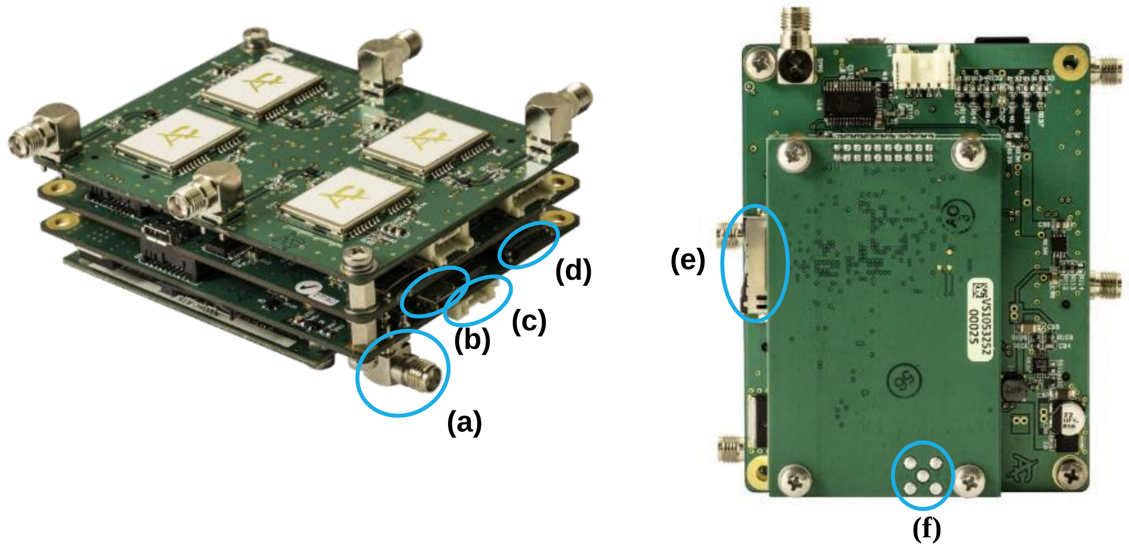


Figure 45: GLAD board interfaces (at RTK and communication modules).

The additional interfaces of the communications board are:

- The GSM antenna connector
- The Micro-USB interface for monitoring & configuration purposes
- The multi-serial interface for alternative retrieval of remote corrections and debug
- The Micro SD card holder, in order to boot the communications board firmware
- The SIM connector for the SIM card used by the integrated GPRS module

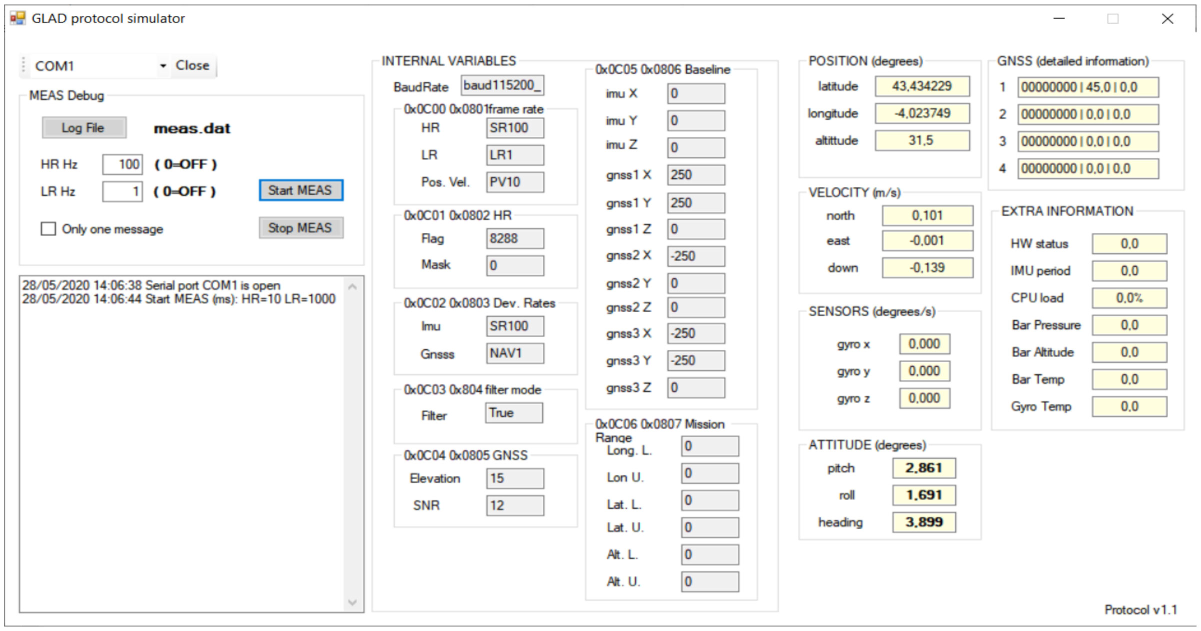
- And the ones contributed by the integrated RTK module:
- MCX antenna connector for the additional GNSS antenna required by the RTK engine.

SW integration

GLAD output: Communication protocol, Integration API, and simulation tool for integration validation.

GLAD output can be retrieved through a proprietary binary protocol (through serial port). GLAD product encompasses an SDK, the GLAD SDK, which facilitates an API for enabling a high-level programmatic integration of the device. This is a light-weight API (few header and source C files, with the parsing functions and data structures), which allows a smooth; software integration of GLAD in the system is serving to.

In addition to that, the GLAD SDK provides a simulation tool, in support for the integrator. This tool enables the generation of output frames as they are produced by GLAD. As shown in Figure 46, this enables the production of the binary frames, sent to a selected COM port. The simulation application also parses the frames and shows the navigation and other data provided by GLAD at its output to let the integrator validate its own usage of the API.


GLAD protocol simulator

COM1 Close

MEAS Debug
Log File meas.dat
HR Hz 100 (0=OFF)
LR Hz 1 (0=OFF) Start MEAS
Only one message Stop MEAS

```
28/05/2020 14:06:38 Serial port COM1 is open
28/05/2020 14:06:44 Start MEAS (ms): HR=10 LR=1000
```

INTERNAL VARIABLES
BaudRate baud115200_
0x0C00 0x0801frame rate
HR SR100
LR LR1
Pos. Vel. PV10
0x0C01 0x0802 HR
Flag 8288
Mask 0
0x0C02 0x0803 Dev. Rates
Imu SR100
Gnsss NAV1
0x0C03 0x804 filter mode
Filter True
0x0C04 0x0805 GNSS
Elevation 15
SNR 12

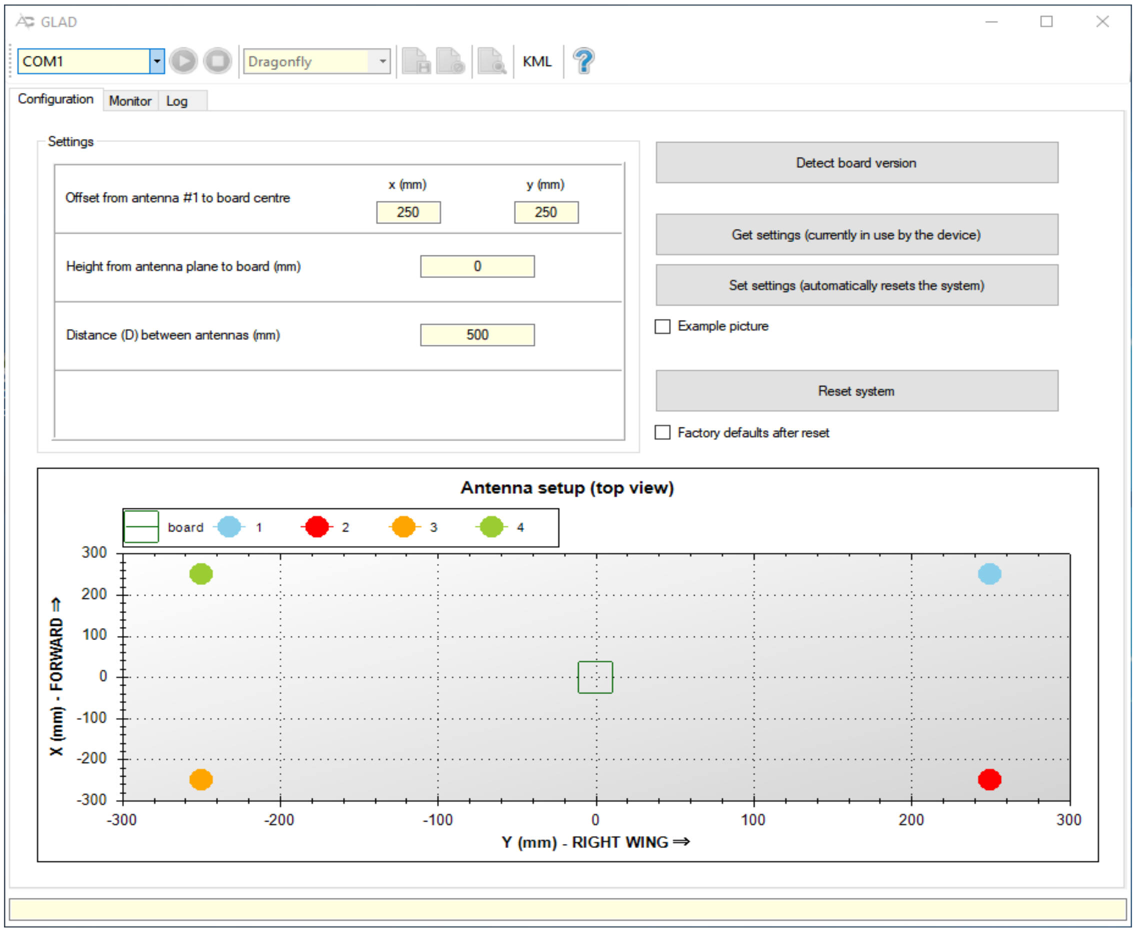
0x0C05 0x0806 Baseline
imu X 0
imu Y 0
imu Z 0
gnss1 X 250
gnss1 Y 250
gnss1 Z 0
gnss2 X -250
gnss2 Y 0
gnss2 Z 0
gnss3 X -250
gnss3 Y -250
gnss3 Z 0
0x0C06 0x0807 Mission Range
Long. L. 0
Lon U. 0
Lat. L. 0
Lat. U. 0
Alt. L. 0
Alt. U. 0

POSITION (degrees)
latitude 43,434229
longitude -4,023749
altittude 31,5
VELOCITY (m/s)
north 0,101
east -0,001
down -0,139
SENSORS (degrees/s)
gyro x 0,000
gyro y 0,000
gyro z 0,000
ATTITUDE (degrees)
pitch 2,861
roll 1,691
heading 3,899

GNSS (detailed information)
1 00000000 | 45,0 | 0,0
2 00000000 | 0,0 | 0,0
3 00000000 | 0,0 | 0,0
4 00000000 | 0,0 | 0,0
EXTRA INFORMATION
HW status 0,0
IMU period 0,0
CPU load 0,0%
Bar Pressure 0,0
Bar Altitude 0,0
Bar Temp 0,0
Gyro Temp 0,0

Protocol v1.1

Figure 46: GLAD SDK simulation tool for supporting SW integration.



Settings

	x (mm)	y (mm)
Offset from antenna #1 to board centre	250	250
Height from antenna plane to board (mm)	0	
Distance (D) between antennas (mm)	500	

☐ Example picture

☐ Factory defaults after reset

Figure 47: GLAD GUI Configuration tab

Configuration

At the developer design perspective GLAD internal algorithms admit a wide and rich configuration. However, for easier end-user usage and easier integration, GLAD enables a user level, simplified configuration (Moreover, much of the internally fixed configuration is the result of previous research and part of the added value).

The configuration can be done in two ways:

- Init set up file (application.ini) on the SD card (read and cold start up
- Via serial interface and a configuration GUI (configuration tab of the GLAD GUI application).

Figure 47 shows configuration tab, which provides a one shot view a quick view of the possible configuration. Most of that configuration is required for the algorithms to consider the integration set up (e.g., distance among antennas, height from antenna plane to board), and the dynamic condition.

HW/SW Platform internals

The base GLAD platform comprises the sensing capabilities (i.e. GNSS and sensors board) and computational capabilities (Central Computing board) for added-value functionality and performance. The board integrates up to 4 low-cost GNSS receivers. It also integrates a low-cost IMUs and a barometer. An ARM-base microcontroller is used for the computation of the ACORDE fusion and position and attitude estimation algorithms. This fusion considers raw data messages from the GNSS receivers, as well as other raw data from the low-cost sensors. To this respect, it is worth to mention that any functionality computed in this computing resource has to provide an added value (i.e. the processing already embedded on the GNSS receivers). Specifically, GLAD fusion produces a tight integration of data from several GNSS receivers, with IMU and barometer data, to produce higher accuracy and integrity on position and high accuracy on attitude resilient to magnetic interference. For the sake of real-time, robust implementation of the firmware, GLAD SW platform relies on a proprietary real-time operative system, with an associated tool-set which has allowed debugging, and also analyzing and inspecting the performance in order to ensure it on the validation phase.

6.7.3 Improvements and Metrics

Improvements:

Among the main limitations foreseen so far, some of them refer to some aspects of the HW platform.

Regarding to the GNSS sensing capabilities, the main ones are possibility to exploit multi-constellation (and specifically Galileo) and additional security features, to avoid both unintended and malicious attacks. This is a specific aspect that has been already started to be tackled in C4D, though a redesign covering a new type of GNSS receiver supporting also Galileo constellation. This receiver will bring also anti-jamming and anti-spoofing capabilities (support and integration addressed in WP5). At the same time, this also is requiring the adaptation of different decoding protocols, as the interface with GNSS receivers is not fully homogeneous. Despite the existence of receiver independent standards (RINEX, NMEA, and RTCM), GNSS receiver integration fully exploiting relevant capacities typically requires to consider also manufactured related protocols (e.g., BINR, UBX, SBF, etc.). To this sense, ACORDE is also making an effort to get, at least, a common, effective API for handling different types of receivers on its internal developments. ACORDE will be monitoring and promoting advances on that sense to get a common format to exploit raw and other relevant data from different GNSS receiver types.

Even more interest and focus will be put for improving the output interface of GLAD. The binary protocol plus the integration SDK provided by ACORDE is a pragmatic, convenient solution to enable and facilitate any integration at programmatic level. However, ACORDE has a specific interest in leveraging, and working on a more standardized interface able to retrieve the output data (and thus added value) provided by GLAD system, and enable, ideally, a Plug and Play approach (where the SDK support become less or no relevant). A number of approaches can be foreseen for that:

- Extension/Selection or some of the existing standards
- Combination of standards
- Protocol designed from scratch accounting the C4D

Here, the more pragmatic thing seems starting from an existing protocol considering other standard solutions and C4D requirements. In fact, ACORDE has started to experiment with its own extension of NMEA 0183. However, ACORDE is fully open to discuss, agree and develop with partners the best option for a fully P&P integrateable components ecosystem.

Another relevant aspect that ACORDE has already started to analyze is the possibility for the SW platform to rely on a license free, real-time operative system. This can drastically reduce the cost and augment the margins. While several options are on a scope (RT-Linux, FreeRTOS, etc.), not only of them will be addressed in the former phase of this activity. Even if, relying on the available information, all of them are decided good candidates, the first phase is to perform the port of GLAD firmware on the studied RTOS, and of this RTOS on the GLAD HW platform. After this, ACORDE is analyzing if the proper functioning and performance vs. the existing implementation.

Last but not least, development conditions need to be taken into account (e.g., tooling, methods for functional or performance bug fixing, testing methods) for their future suitability for certification. Notice that this activity is then very much related to methods and tools (proper of WP6), where these aspects will be touched. In WP3, the activity is related to the selection of candidate RTOS, and their porting.

6.7.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective

Ease of integration (hardware)	• Number and cost of mechanical pieces required for the integration • Time spent for the integration • Need of custom mechanical design • Need of connector adapters • Need of voltage adapters • Need of additional power sourcing	SC1.1, SC1.2	O1
Ease of integration (software)	• Number of lines of code • Time spent	SC1.2, SC4.1	O1, O4
Ease of configuration	• Number of user parameters • Time spent for the initial setup • Configuration time spent per flight	SC1.2, SC4.1	O1
Positioning & Attitude performance	• Positioning error in semi-urban canyon and urban canyon scenarios (compared against a known good reference) • Heading and pitch/roll error in static & dynamic scenarios (compared against a known good reference)	SC2.1 SC2.2	O2

6.7.5 Models for Position and Attitude estimation (ACORDE and UNICAN)

The Position and Altitude System will be encapsulated as a HW/SW component ready to be integrated in the product in which they have to operate. The description of the component will include the components themselves with all the additional information. That additional information comprises the models of the whole component and of key parts of it, which are able to feed the modeling, simulation, performance analysis and synthesis tools. A complete model is useful in the design process of the system where the component is integrated. Models associated to key part of the position and attitude system, and of its current and future alternative architectures are greatly relevant for the optimization at the two planes: algorithm and architectures. A current main focus on the position and attitude system is on the attitude functionality, although assessment could reveal other parts of interest.

ACORDE and UNICAN will develop those key models. ACORDE in SystemC and UNICAN with S3D (UML/MARTE) will feed the design frameworks developed and reported in WP6.

The S3D model will be integrated in a component's library for further reuse in future projects. The S3D modeling methodology encapsulates a functional component with its functional code (i.e. in C++) and the verification test-bench. The system model is automatically generated taking into account the underlying execution platform where the components will be executed.

Both the SystemC and the S3D models will be used and assessed in UC2. Moreover, ACORDE and UNICAN will collaborate to align the models so that comparatives of interest on alternative design technologies, e.g., for performance estimation, can be assessed, and this way gains key information on the best design technology choices.

6.8 Hyperspectral Imaging (HSI) Processing Pipeline - IMEC-BG

6.8.1 Component Description

Description: Hyperspectral imaging (HSI) processing pipeline

The goal of this component is to design a processing pipeline based on hyperspectral imagery to inspect (off-shore) critical infrastructure, such as off-shore wind turbines, as is depicted in Figure 48. Imec will further develop its current pipeline and rely on its expertise to efficiently deploy algorithms on the NVidia Jetson boards.

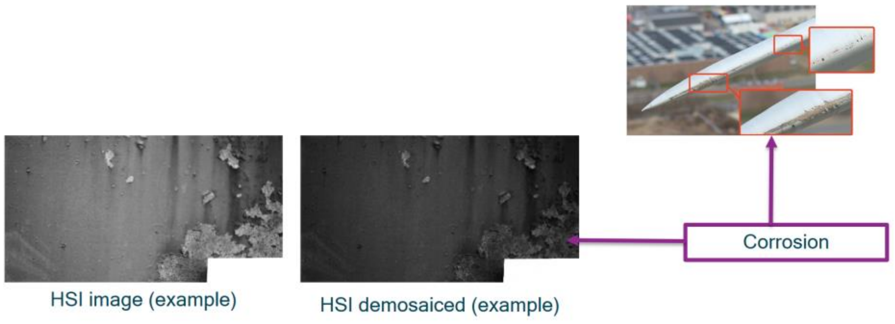


Figure 48: The goal of this component is to detect corrosion on critical infrastructure using hyperspectral imaging.

Keywords: hyperspectral imaging, AI, corrosion detection

6.8.2 State of the Art

Currently, there exist many hyperspectral image processing algorithms (e.g. demosaicking for mosaicked sensor layouts or deep learning-based detection, segmentation or classification). However, they are developed and designed for (off-board) PC platforms and are totally not optimized for the imec's hyperspectral dual camera payload, integrated nor run on embedded hardware platforms such as the Jetson TX2 board.

Classic deep learning frameworks rely on massive amount of annotated data, over which we will not dispose (and are not able to collect ourselves). Therefore, we rely on recently developed few-shot learning techniques, which are trained with only a limited number of annotated samples. However, the robustness under various noise conditions and few-shot learning performance needs further research. In the case of hyperspectral imaging, this will also impact the acquisition: e.g. the varying incident sun light will create different appearances of the same physical material. Proper normalization procedures are needed to be developed.

6.8.3 Improvements

The entire pipeline is made up of three main modules: pre-processing, on-board analysis and post-processing, as can be seen in Figure 49.

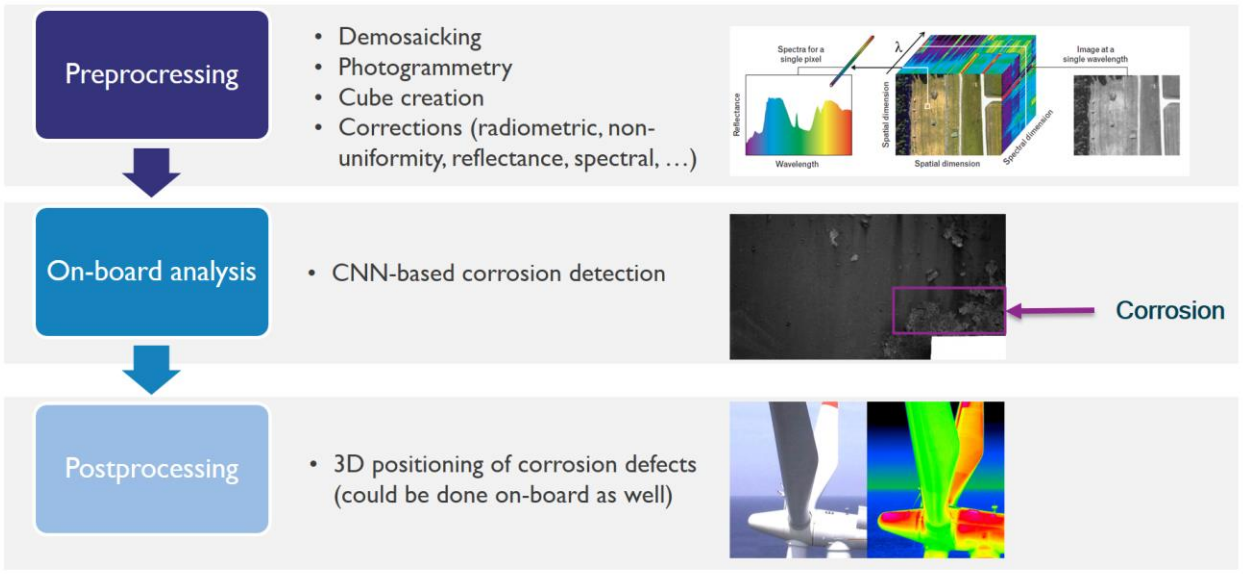

Preprocrocessing
• Demosaicking
• Photogrammetry
• Cube creation
• Corrections (radiometric, non-uniformity, reflectance, spectral, …)
Spectra for a single pixel
Image at a single wavelength
Reflectance
Wavelength
Spatial dimension
Spectral dimension
λ
On-board analysis
• CNN-based corrosion detection
Corrosion
Postprocessing
• 3D positioning of corrosion defects (could be done on-board as well)

Figure 49: The hyperspectral imaging processing pipeline.

The pre-processing consists of four different sub-modules. First, a demosaicking algorithm is developed to estimate a multispectral image with full spatial-spectral definition. Second, a hyperspectral image cube is created. Third, several corrections are carried out: radiometric, non-uniformity, reflectance, spectral (varying lighting conditions) and corrections coping with degradations due to vibrations, fading, etc. Finally, the hyperspectral images are stitched together and a 3D model is constructed.

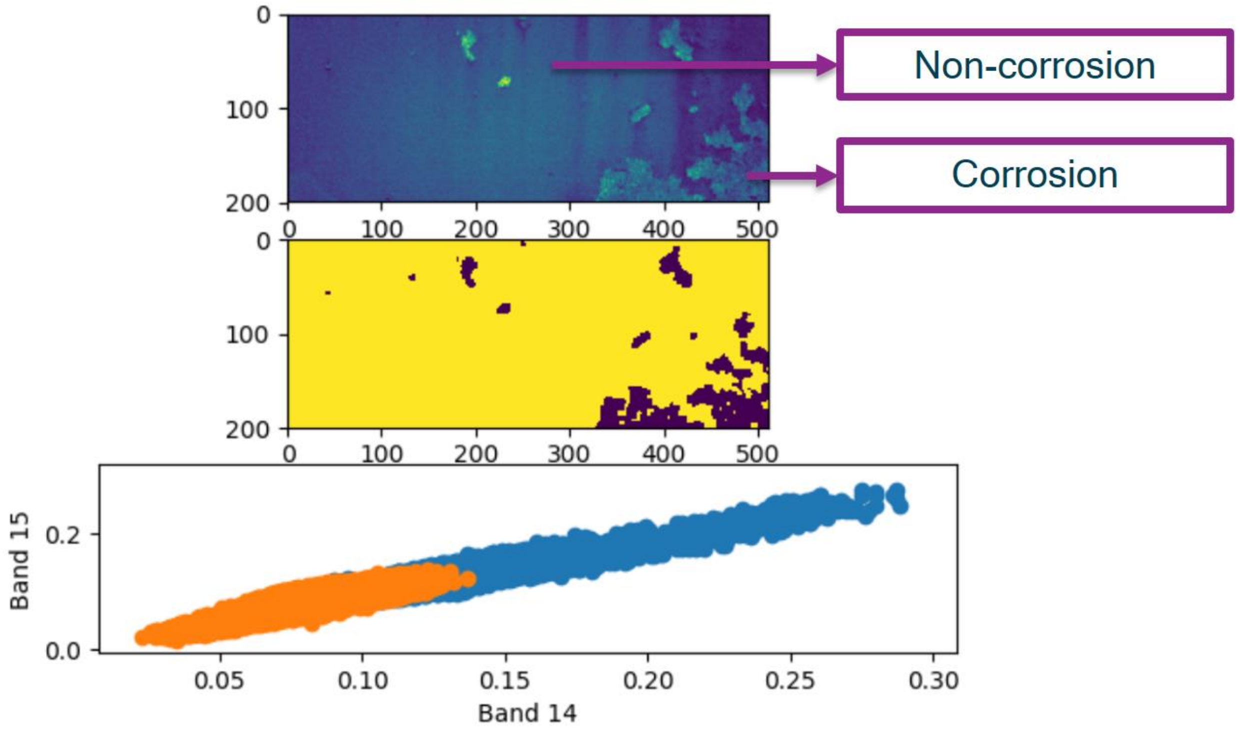


Figure 50: Result of the corrosion detection algorithm: yellow denotes back ground, purple denotes corrosion.

The on-board analysis is dealing with the detection of degradations, i.e. corrosion, of the infrastructure. To this end, AI-algorithms (CNN-based) are developed. A result of the current corrosion detection algorithm is depicted in Figure 50. The purpose is to implement these algorithms on the Jetson TX2 board in order for them to be executed in real-time. This way the corrosion parts can be identified and located online and the drone can be instructed to fly towards the most degraded areas in order to limit fly-time and avoid that relevant areas remain uncaptured. The final goal is to achieve automatic HS-image-based detection and quantification of corrosion using AI technology with an accuracy of 80% compared to human inspections.

Finally, the post-processing module will position the corrosion defects on the 3D-model acquired from the infrastructure. This sub-module can be conducted online as well in case the final algorithm is fast enough. This way, the drone operator can have a real-time view of the corrosion defects overlaid on the 3D-model.

6.8.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
On-board hyperspectral image processing	- Time to configure the HSI pipeline to work fully automated. - Number of parameters to set	SC1.1 SC1.2	O1

Automatic HS-image-based detection and localization of corrosion using AI technology	Time to configure sub-modules for the HSI processing pipeline, e.g. the application of corrosion detection using HSI	SC2.1	O1, O2

6.9 SLAM Algorithms - Modis

6.9.1 Component Description

The SLAM capabilities will be based on mixed sensors measurements and will not rely necessary on the GPS signal. In fact, the system will be equipped with a watchdog alert for failure detection of the GPS.

Keywords: (Outdoor) Positioning, SLAM, Particle Filters

6.9.2 SoA comparison

SLAM (Simultaneous Localization and Mapping) techniques are considered to be a mature field. In particular, SLAM based on IMU has been successfully leveraged to solve indoor positioning problems in 2D classical robotics applications.

As for our concern we will mostly look at the localization problem using IMU technologies and possibly other information (e.g., lasers, ultrasounds, cameras). The purpose of this algorithm is not to replace GPS signal, but rather to enhance drone autonomy by providing a backup mechanism triggered by GPS failures. Below we provide a high-level description of some of the state-of-art components we will be using in our SLAM algorithms, namely particle filters.

We will adopt a probabilistic approach based on Bayesian inference and Gaussian modeling. Informally, we formulate the localization problem as the problem of determining the location of an agent at a given time t within a map m. Due to errors in the sensor readings and in the controls provided to due robot, it is not possible to determine (or attempt to do so) the exact position of a robot in a given time. Rather, it is common to model such position via a conditional probability distribution called belief:

$$bel(x_t) = p(x_t|z_t, u_t)$$

This is the probability distribution of the location of the robot at time t given the sensor measurement z_t and the control u_t. The goal of robot localization then can be stated as the problem of finding a good approximation to $bel(x_t)$ at each timestamp and particle filters are algorithms that approximate the belif distribution.

Models. Notice that both the sensor readings z_t and the controls u_t are treated as random variables due to potential hardware failures and noise. Typically, in robotics, the design also provides a motion model and a measurement model. Those models take into account the dynamic of the robot and specify how the robot move from one state to another, and how the measurements are affected by the current position. Specifically, the motion model provide a formula for the conditional distribution

$$p(x\_t|u_t, x_{t-1})$$

and the measurement model provide a formula for the conditional distribution

$$p(z_t|x_t).$$

For example, in the case of a robot moving in a two dimensional plane we could model the robot position at a given time t as a vector (x_t, y_t, θ_t) where x_t is the x-coordinate, y_t is the y-coordinate and θ_t is the robot heading orientation measured as an angle with respect to the x-axis. Similarly, we could model a control u_t as a rigid translation and a post (pre) rotation with the vector (x_t', y_t', θ_t'). Notice that the components of ut are relative to the current position rather than absolute (i.e. relative to the origin). The system dynamics is given by

$$x_t = a\, x_{t-1} + b x_t' + e_x(u_t)$$

$$y_t = a\, y_{t-1} + by + e_y(u_t)$$

$$\theta_t = a\, \theta_{t-1} + b\, \theta_t' + e_\theta(u_t)$$

Where the as and bs are given constant depending on design and the es are random functions of the control modeling the noise. Typically, noise is assumed to be Gaussian and therefor, the motion model is Gaussian as well.

For the measurement model a map-based model might be employed. We define a map as a function $m: R^n \to R^d$ assigning measurements to a given position in the space. Here, n denotes the dimensionality of the position space (e.g. 2 for planes and 3 for spaces) and m denotes the number of measurements collected at a given time (this may depend on the number of sensors available). One can think of m(x) as an estimated measure at position x. concretely, given a certain position x_t we could model the measurement z_t as a multivariate Gaussian distribution with mean and covariance related to the map as below

$$p(z_t|x_t) = c\mathrm{e}^{-\frac{d(z_t, m(x_t))}{2\sigma^2}}$$

Where d is a distance function between the actual measurement z and the estimated measurement at x based on the map and c is normalization constant. Such a function might be given by a generic quadratic form of the type.

Particle Filters. Particle filters approximates the belief function with a set of points $X_t = \left(x(i)\right)_{i=1}^{P}$and a set of weights$W_t = \left(w(i)\right)_{i=1}^{P}$. Points are referred as particles. Each point x is a possible position of the agent at time t and its weight w represents the probability of such a position. The algorithm computes the pair (X' , W') from the previous approximation (X , W), the current control and the current measurement. Specifically, the algorithm starts looping through the set of previous particles X and use the motion model to propagate such particles on the new position. To do so, for each particle the motion model is instantiated with the current position x(i) and the current control u, i.e. a sample x'(i) from p(x|x(i)) is drawn. After, each x'(i) is weighted proportionally to its measurement likelihood p(z|x'(i)) instantiated on it. The factor w(i) accounts for the probability of the sample x(i) originating the propagation. Finally, the normalization step ensures that the new set of weights is an actual probability distribution over the new set of particles. One issue with this algorithm is that over time, particles with small weights tend to get even smaller weights, until the belief becomes essentially a Dirac distribution settled on a single particle. This phenomenon is known as particle depletion. In order to prevent such phenomenon, a variant is often implemented. The algorithm is the following. First, it form the set of particle and weights using the standard particle filter, then the algorithm fixes the sample set by resampling according to the new weights W' . This step allows for selecting sample with higher probability. Finally, in order to avoid depletion after resampling, the practical the weights are set to the uniform distribution

Importance Sampling. Particle filters require to sample from the conditional distributions associated to the motion and measurement models. Those distributions may be arbitrary complicated and algorithms for exact sampling might be not available. For these cases, it is possible to resort to a technique called importance sampling. In importance sampling there is a distribution (given via a pdf, for example) f given in closed and form which sampling is hard, this is called target distribution. The technique is based on the choice of a proposal distribution g from which sampling is easy and such that f(x) > 0 implies g(x) > 0. The importance factor of x is given by w(x) = f(x) g(x) and represent the mismatch between the distributions f and g on x. After choosing such a g, it is possible to approximate statistics of f starting from P samples from g. For example, if A is an event in the σ-algebra of f then

$$\mu_f(A) \approx \sum_{i=1}^{P} w(x(i)) I(x(i) \in A)$$

Where$\mu_f(A)$ is the probability of A under f and I is the indicator function.

Estimation. Once the belief has been computed, it is often the case that a single guess for x needs to be provided. One possible choice for this is to estimate the current position with the average of the particles. This estimator is optimal with respect with the Mean Square Error (MSE). An alternative to the optimal MSE estimator, is the Maximum A Posteriori (MAP) estimator, which output the particles x with maximum weight. If a single point estimate is not enough, one can provide an estimate distribution, by fitting a Gaussian to the particle or training an histogram.

6.9.3 Improvements

Magnetic-field positioning has been successfully exploited in classical indoor robotics and it is appreciated for its low implementation costs. We aim to investigate its suitability as a support to the GPS signal for outdoor positioning. Our algorithms will also provide SoA mechanisms for fail-safe mission and increased PBN operations. As a byproduct, this system should also allow for real-time as well as post-mission analytics aimed at identify critical regions in the map where GPS signal is particularly weak and/or detect potential attacks.

6.9.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
SLAM drone localization / build open API to easily interface with control software	Correct interaction through open API and standard communication Buses (e.g. MavLink) with other drone components (IMU, GPS sensor, Control Unit). Code length minimizing	SC2.2	O2
Increase of drone availability	Reduction of Mean time between failures with faster inter-component communication through reuse of commercial components with redundancy and fault prediction	SC2.2	O2

6.10 High Accuracy EGNSS Navigation Component- TopView

6.10.1 Component Description

Description: The GNSS component that will be integrated by TopView is based on a dual frequency, multi constellation EGNSS receiver to be installed on-board two typologies of drone platforms capable to embark it, considering both fixed wing and Multicopter drones. In both cases the electrical and power interfaces of the GNSS component will be connected with the drone's main communication bus and with the drone power bus.

The first GNSS component solution prototype (TRL5) will not have an industrial form factor, and may reach also a quite considerable weight for a small drone (0,5 -Kg), however it will exhibit all the technical features needed to achieve the following features:

- interface the U-space tracking service envisaged for position reporting, with particular reference to the Italian U-space service provider “D-Flight”
- interface to Open source Autopilot PixHawk / NAvio2 / Mavlink enabled through UART / I2C bus
- interface to Proprietary DJI A3 / N3 autopilots though Can BUS interface and DJI SDK SW components for navigation purposes

- Decametric accuracy (15- 20 cm) in combination with a Local base station or trough IP based network.
- Telemetry reading of main parameters on the main autopilot bus for feeding the respective U-space tracking service

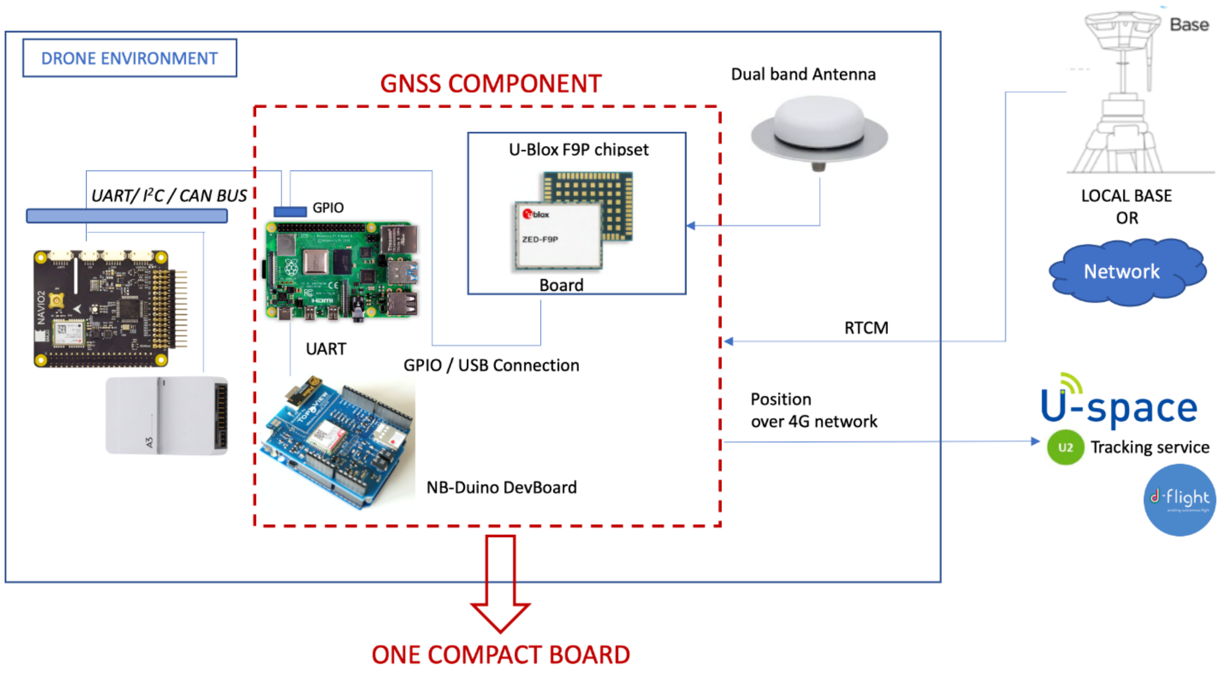


Figure 51: TopView component

The GNSS chipset ZED-F9P positioning module features the new u-blox F9 receiver platform, which provides multi-band GNSS to high volume industrial applications in a compact form factor. ZED-F9P is a multi-band GNSS module with integrated u-blox multi-band RTK technology for centimeter-level accuracy. The module enables precise navigation for the envisioned drones' platforms. Moreover, ZED-F9P ensures the security of positioning and navigation information by using secure interfaces and advanced jamming and spoofing detection technologies. ZED-F9P offers support for a range of correction services allowing each application to optimize performance according to the application's individual need. ZED-F9P comes with built-in support for standard RTCM GNSS observables, supporting centimeter-level navigation from local base stations or from virtual reference stations (VRS) in a Network RTK setup.

The Communication Module NB-Duino has been developed by TopView as Development Board to test the 4G Narrow-Band-IOT network and will be used in the envisaged GNSS C4D component, to offer Communication services towards the U-space and local GNSS base station RTCM communications.

Finally, a local Microcomputer (e.g. Cortex STM32) will be used to interface the target autopilot.

The objective is to design one unique compact board (Plug and Play) for different categories of drones that can implement both high accurate Navigation performance and position reporting to U-space tracking services.

Keywords: European GNSS, U-space Tracking service, 4G NB-IOT, Repeatability, Accuracy, Integrity, Continuity, Multi Frequency, Multi constellation

6.10.2 State of the Art

UTM Box are being proposed by some manufacturer for implementing the U-space tracking service; reporting to U-space drones positioning to unlock some airspaces (i.e. Zu, Za with respect to CORUS project Conops) where tactical separation is offered.

These boards have an alternative GNSS receiver (generally a low-cost component) used only for reporting positions, but not used for navigation purposes (see Figure 52).

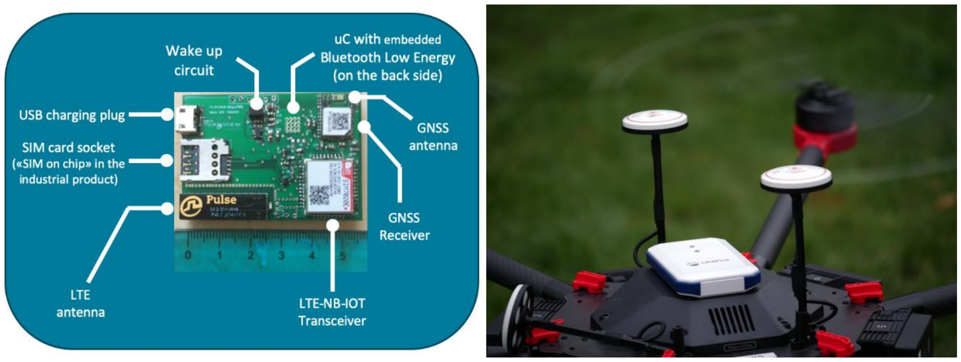

Wake up circuit
uC with embedded Bluetooth Low Energy (on the back side)
USB charging plug
SIM card socket («SIM on chip» in the industrial product)
GNSS antenna
GNSS Receiver
LTE antenna
LTE-NB-IOT Transceiver

Figure 52: Examples of GNSS UTM boxes for position reporting

On the other hand, there are many high end commercial systems implementing RTK functionalities for high end applications, however none of them impalements both functionalities. Moreover, no GNSS component on the market is actually available to be used with Commercial drones (such as DJI Matrice series) as navigation sensor instead of the one built in.

6.10.3 Improvements

This component will improve drone Navigation performance (also in the vertical component) and will introduce a new functionality tailored for U-space tracking services.

In particular for smart farming applications (Use Case 5), this component is intended to address specific challenges in survey application:

Facilitation of A.I. algorithms analysis: Thermal and Multispectral images with the actual relative low resolutions available are in general not suitable for accurate orthmosaic processes or to build up heat map layers in software suites, unless the drone flights are very close to the terrain surface (with the drawback of too many images to handle). The proposed improvement is to collect a temporal time-series of nadiral Thermic / Multispectral images collected always in the same sampling spatial points with high accuracy in the positioning. This process will improve feature based A.I. algorithms to monitor the plants during the all growing phases, but it requires high accurate positioning performance in order to minimize false positive of the AI/ Machine learning techniques.

Terrain Following: High end productions as grapes for fine wines are typically located on hills. When planning a mission aimed at allowing A.I. analysis in post processing or sprays fertilizers, it is important to reach each waypoint with a high positioning accuracy not only on the horizontal plane, but also on the vertical axis. The GNSS based component (with the utilization of European GNSS signal) will improve navigation and position performance also on the vertical axis without additional proximity sensors (ultrasonic, optical flow, lidar, etc.) that often-present issues above vegetation.

Galileo Added value: The utilization of the European GNSS differentiators (Galileo Navigation Message Authentication service OS-NMA) on GNSS COTS receivers (when available) may represent a first layer for the certification of position feature, opening up a new market for agricultural use (Insurance companies payback, traceability of organic food,)

Enhance Link efficiency for Ground Sensor data collection: Ground Sensors placed over the farmyard can be awaken with more reliability by a drone transponder, to upload data collected [TOPVIEW PATENT USPTO, 62895514 filed on Sept. 2019:" A System for Data Collection through Unmanned Aerial Systems (UAS) "].

Interface the U-space tracking service envisaged for position reporting interface to Autopilot for accurate navigation purposes.

6.10.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Improve navigation precision (accuracy)	Navigation accuracy	SC2.2	O2
Reduce the navigation error	Error ratio	SC2.2	O2

6.11 Complex System for Autonomous Drone Battery Management - UWB & SM

6.11.1 Component Description

Description: The UWB and SM will be jointly developing complex system for autonomous drone battery management. The main component of this system will be device called Droneport (DP). The DP will present an autonomous station with defined communication interface realized via MAVLink messages. The main tasks of the DP will be guidance of the drones for the landing, change or charging of drone batteries and proper resumption of the original drone mission. The DP will communicate wirelessly with the drones and will be able to monitor the request for recharging of the drones. The DP will also assist the drones in landing by providing local positioning system based on visual servoing.

The DP HW will comprise of two basic components a base station and a battery mounting and connection unit. The base station will not only provide a landing space for the drones, but it will be equipped by battery charging and storage unit and exchanging manipulator. The battery mounting and connecting unit will provide unified charging interface for the drones making it possible to incorporate support for the DP to UAVs of various providers. This unit will offer easy locking and contact system similar to standard battery based on contact sliders and spring lockers.

The SW architecture of the DP will be responsible for drone to DP communication protocol specification and will provide drone landing assistant, battery exchange system control and battery management for charge control of the stored batteries. The SW architecture will provide open API for possible interoperability with various drone flight controllers.

Keywords: autonomous battery management, landing assistant, visual servoing

6.11.2 State of the Art

There is ongoing interest in extension of the drone flight time in all sorts of mission. The surveillance and inspection mission are prime example of drone applications where shot flight time constrained by limited battery capacity is a major problem. There are already proprietary solutions like ASYLON DRONEHOME and Skydio 2 Dock which are however targeted on specific drones and have closed architecture. There is also some recent research[56] that provides description of autonomous battery swapping systems prototypes. They are however mainly focused only on the exchange system HW and do not attempt to solve all the aspects of the battery management system.

6.11.3 Improvements and Metrics

The Droneport HW and SW platform aims are to provide open all-round battery management system that could provide the battery exchange/recharge capability to various drone systems. The battery mounting unit of the DP should make it possible to extend wide variety of drones with system capable of exchange of batteries and use the DP landing assistant. Also, it will provide unified interface for the DP base station battery storage and charging unit. The SW platform will employ standard MAVLink

[56]Mirzaeinia A, Hassnalian M. and Lee K. Drones for Borders Surveillance: Autonomous Battery Maintenance Station and Replacement for Multirotor Drones, AIAA SciTech Forum, DOI 10.2514/6.2020-0062, 2020
Liu Z.-N. et. al. An Autonomous Dock and Battery Swapping System for Multirotor UAV, DOI 10.13140/RG.2.2.19437.90085, 2018

communication technology for the purposes of exchanging the status information between the drones and DP thus easing integration to current flight control systems.

6.11.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Ease of battery power management	Increase of effective mission time due to the early detection of low energy situations and timely return of the drone to the mission after battery replacement	SC2.2	O2
Ease of battery system design	Increase efficiency of drone power delivery design by offering ready to use design of smart battery system. The smart battery system provides specification of compartment suitable for quick automatic replacement using robotic manipulator with standartized interface that makes it possible to monitor the battery overall state.	SC1.1	O1

6.12 Smart and Predictive Energy Management System - UDANET

6.12.1 Component Description

An energy management system is vital to optimize the energy life and the purpose of the system: it will continuously monitor important system parameters, while dealing with the varying power demands of the many aspects, the objectives of the mission and optimizing the usage of the energy. The designed energy management system will be verified and tested via Software in The Loop using software Matlab-Simulink. Given the initial and final positions, the objective of the component is to compute the control inputs that rule the motion and vehicle trajectory to optimize energy consumption and the computational burden of the algorithm.

Keywords: Miscellaneous (Battery management), Model-based control, Optimal control, Rule-based strategy, Consumption minimization, Energy efficiency, UAV dynamic modeling, Computational cost, Robust control.

6.12.2 State of the Art

In recent years, various research[57] has been done to solve problems related to flight control for drones. Various control techniques have been applied: PID controller, sliding mode, model predictive control, and optimal control. The applied techniques have almost always used an approach on the model and the results obtained have shown that it is possible to save energy with the application of the control theory. The results obtained are dependent on the flight dynamics models of the drone used, on the battery model or on the energy model, which are necessary to formulate and then solve the control problem.

6.12.3 Improvements

The aim of this activity is to use a model-based approach to control UAV flight to minimize energy consumption: it is therefore necessary to consider the flight dynamics, battery, and energy flow and actuator models. The strategy that led to good results in some research is the optimal control and therefore this type is considered in the activity. A first objective is to improve control by making it robust with respect to model approximation errors or disturbance (for example under wind conditions).

[57] F. Yacef,, N. Rizoug, O. Bouhali, and M. Hamerlain*Optimization of Energy Consumption for Quadrotor UAV*, International Micro Air Vehicle Conference and Flight Competition (IMAV), pp. 215-222, 2017.

One of the problems encountered in electronic devices in all sectors, and even more so in drones for obvious reasons, is that related to the computational complexity that weighs on the overall processing capacity.

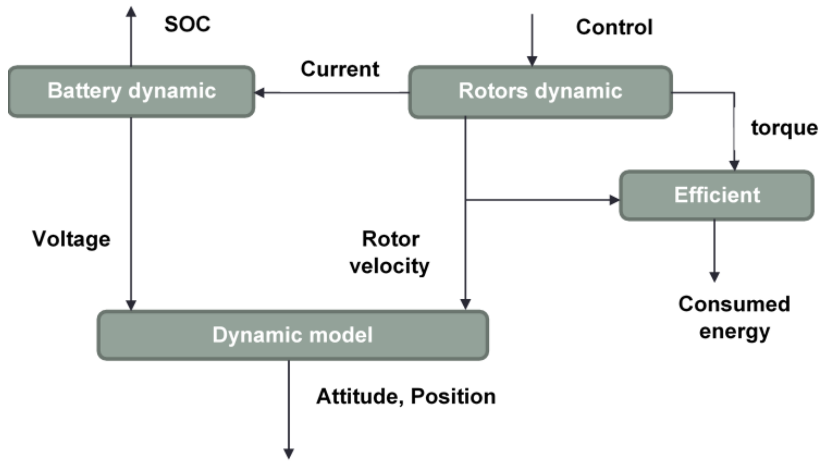


Figure 53: Power Lose Model

The idea to be applied in this case is to use the choice of optimal control to identify common paths in preferences with the aim of obtaining a sub-optimal rule-based strategy which has a reduced computational load.

The idea is imported from the automotive sector, where there is undoubtedly greater calculation capacity on board and there are fewer problems due to weight, and has been successful for controlling the energy flow in hybrid vehicles to minimize consumption and divide the pairs among the possible actuators. Another aspect that can improve control is to study it with respect to external disturbances, for example in windy situations.

6.12.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Improvements in performance and resource consumption	Application execution time and memory requirements Percentage of improvement in terms of energy saved Robustness with respect to changes in conditions	SC4.1	O4

6.13 AI Drone System Modules - UDANET

6.13.1 Component Description

The design and implementation of Artificial Intelligent methods to classify leaf diseases will be provided. Analysis and development of the training, verification and testing phases of the algorithms characterize the operational phases of component formation. The component inputs are images from cameras, the output is the plant health status classification.

Keywords: Image processing, Artificial Intelligence, Deep Learning, Convolutional Neural Network, Data augmentation, Classification methods, Accuracy, Cost.

6.13.2 State of the Art

Artificial intelligence algorithms applied in image and video processing has been in full development for some years. In literature many studies have been proposed58, each with different network architectures and learning algorithms. Depending on the applications, the reference dataset that can be accessed varies and this influences the performance of the algorithms themselves. For example, it is not easy to find complete and well-made datasets for smart agriculture as in other areas of application. Furthermore, an aspect often underestimated at the theoretical level in the literature is the computational one, which in fact has a great influence on real-time performance. The results obtained both in terms of detection and recognition are remarkable and open the future to this type of algorithm for problems such as those in question.

6.13.3 Improvements

Artificial intelligence algorithms with different characteristics will be designed and implemented with the aim of taking into account the computational cost of each, as well as the over-fitting problem and the dataset that is not always adequate, and the diagnostic performance is drastically decreased when used on test datasets from new environments (generalization problem).

A large amount of data increases the performance of machine learning algorithms and avoids over-fitting problems. Creating a large amount of data in the agricultural sector for the design of models for the diagnosis and detection of plant diseases is an open and challenging task that takes a lot of time and resources.

6.13.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Improvements in performance and resource consumption	- Application execution time and memory requirements - Accuracy	SC4.1	O4
Easy to design, develop, and deploy applications	- Simulation execution time - Number of lines of code (for SW design and application design)	SC4.1	O4

One way to increase the dataset will be to use data augmentation techniques: it increases the diversity of training data for machine learning algorithms without collecting new data.

The improving objective of this component is to develop and test the algorithms, increase the reference dataset using basic image manipulation and deep learning-based image augmentation techniques such as Image flipping, cropping, rotation, color transformation, PCA color augmentation, noise injection, Generative Adversarial Networks (GANs) and Neural Style Transfer (NST) techniques. Performance of the data augmentation techniques was studied using state of the art transfer learning techniques both in terms of accuracy and computational cost.

[58]PragatiPukkela and SurekhaBorra, *Machine Learning Based Plant Leaf Disease Detection and Severity Assessment Techniques: State-of-the-Art*, Lecture Notes in Computational Vision and Biomechanics, pp. 199-226, 2018.

6.14 Component Generic Mission Controller - SCALIAN

6.14.1 Component Description

Description: Scalian has worked on developing a generic architecture to allow fleet of UAVs to perform a variety of mission. The generality allow to use heterogeneous UAVs and to allocate them different missions. For instance a fleet of 10 UAVs has performed logistic operations by dropping sensors over a huge area collaboratively, or a fleet of 4 UAVs has swipe scanned areas to build an orthophoto.

Inside the architecture there are several components (from low level like control, up to high-level like task planning). The Aerial Mission Controller (AMC) or Flight Mission Controller (not to be confused with Flight Controller), is the center point and coordinates all the other components.

The AMC uses a set of finite state machines (FSM) to control its behavior. The main FSM holds the different main mission steps the UAV has to execute to complete the operations. Before taking off, the AMC ask the Task Planning component for the list of next actions. The Task Planner computes this list from the current status of the mission and of the other UAVs as reported in the Knowledge Base (KB, see section 4.1.15). It takes into account the UAV type to determine the type of actions to carry out. Changing the Task Planner allows to change the type of mission the fleet will accomplish.

Once the AMC receives the list of actions, it will start the execution: it carries out the action in the list, one by one while monitoring their completion and the activities of the rest of the fleet. Throughout the flight cycle, the AMC will report to the Knowledge Base the status of the UAV (position, attitude, battery, and so on) in addition to mission progress (current action, completed actions, and so on).

For each action in the list, the AMC will ask a Path Planner to find the trajectory to use taking into account the trajectories of all the other UAVs in the fleet. Once a path has been found, the AM books the corresponding airspace to prevent another UAV from using it.

Task planning and task execution and monitoring are the main activities of the AMC, but it must also monitor the internal status of the UAV and its components (both software and hardware). A set of watchdogs monitors those components, when one deviates from its expected behavior then the AMC tries to solve the issue. It starts by reporting the issue to the fleet (through the KB), then follows a corrective procedure. Depending on the criticality of the component and the failure, it can range from holding the position (e.g. when losing connection to the KB), to aborting mission (e.g. when the payload no longer operates), to finally performing an emergency landing (e.g. main hardware failure).

Another responsibility of the AMC is to computes a new list of actions when the mission parameters or the action cannot be performed. Several occurrences can trigger this re-plan: a new geofence appears and the current UAV must change its trajectory (intruder detected on the ground), another UAV had to change its trajectory and will use an airspace booked by the current UAV, an intruder has been detected at the location where the UAV was supposed to drop a sensor, to name only a few. When the AMC needs to re-plan, it tells the fleet about it asks the Task Planner for a new list of actions and restarts the process of monitoring its execution.

Finally, a Ground Control Station (GCS) is connected to the KB, it can then display the mission progress and UAVs status. Additionally, the GCS is allowed to send updates on the mission parameters: the GCS operator can create a geofence when an intruder has been spotted (either via one the UAV or via a security operator). The GCS also has the possibility to send high-level orders to a single UAV or several of them, for instance to force their return to base. This guarantees that event that cannot be controlled (and understand by an autonomous system) are accounted for by the system. For instance an issue in one the infrastructure, not monitored by the system, could make the operations too dangerous and require the fleet to stop its operations.

Keywords: Operations management, Intelligent Mission Management, Planning and Scheduling, Fail-safe Mission, Drone and Rover, Swarm formation and cooperation.

6.14.2 State of the Art

The EZ_Chains architecture, for which the Aerial Mission Controller has been developed, is described in D2.1.

The approach of using Finite State Machine to control and monitor the execution is a common practice. The addition of watchdogs allows monitoring as many elements as needed. Most of their outputs are the same events that trigger the changes in the FSM. This allows adding and removing most of the watchdog with low impact on the AMC. The watchdogs whose output is not "common" events are in fact watchdogs required to ensure proper functioning, for instance: critical battery level should trigger a behavior required only to handle this case.

6.14.3 Improvements

Scalian aims at expanding the Aerial Mission Controller (AMC) into a Generic Mission Controller (GMC or xMC). The goal is to allow our EZ_Chains architecture to be able to operate UAVs, rovers, and other types of autonomous system, for instance intelligent sensors and actuators (e.g. weather station, operation-area access control).

In addition to accommodating more types of autonomous system, it needs to increase the number of agents it can control during a mission to make operations faster but also to account for the new variety of agents.

The AMC has been demonstrated in simulations, soft-in-the-loop tests, hardware-in-the-loop tests and real operations. The development of the GMC must also use this proof process to ensure a safe system. This constraint is taken into account during the design phase.

In order to increase the safety of operations, the GMC must also integrate new components to work with Air-Traffic Management. When a foreign aircraft enters the airspace, the human operators must add a geofence around it in the geofence or trigger an appropriate mode for the fleet (return to base or landing on alternative zones). It would be better if the GMC could take the ATM into account: it could act sooner and more efficiently. For instance placing a smaller geofence, sending orders only some UAVs in the flight plan, and so on.

Finally, in the scope of Comp4Drones, many components will be developed, by Scalian and other partners, so the GMC must be made more composable to benefit from the rich ecosystem.

6.14.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Easy to configure new missions for a fleet	Number of lines of code necessary to support the new mission	SC1.1, SC1.2, SC2.1	O1, O2
Easy to integrate a new agent to a system	Number of lines of code necessary to add an agent (mobile or static)	SC1.1, SC1.2	O1
Easy demonstration of safety of operations for a given fleet and mission	Simulations and code safety	SC2.2, SC4.1, SC4.2	O2, O4
Easy integration of new components/features in the architecture	Number of lines of code to integrate a partners component	SC1.1, SC1.2	O1

6.15 Component Fleet Knowledge Base - SCALIAN

6.15.1 Component Description

Description: Scalian has developed a component called Knowledge Base, and it is part of the EZ_Chains architecture. It is the component responsible for maintaining a shared mission status inside the system. Agents connect to it and are then able to get and update that status, but also share their own status.

As of now, the system supports two types of agents: UAVs and GCS. The UAVs use the mission status to plan their mission, and report their status. The GCS uses the KB to display the mission progress to the operators and allow them to send orders to the UAVs (e.g. return to base …).

The UAVs use the current mission status to choose the next actions to carry out, and then tell the other UAVs (through the KB) their list of actions. During the execution of the actions they will report the progress. Additionally, when the UAVs plan their trajectories, they get the position and flight plan of the other UAVs, and once they have found a trajectory they book the corresponding airspace (up to a certain distance) to prevent the other to use it.

When the UAVs carry out their actions, they will periodically report other information or instance their position, level of battery, and so on. Furthermore, some actions require reporting information such as intruder detection: the UAV records the detection, its position it provides the image and the detection bounding boxes.

The UAVs can rely on the KB to obtain mission data in addition to its status. The data can be: number of other UAVs, the flight area (geocaging), the exclusion zones (geofencing) and so on. We can update the data as the mission is carried out, for instance we use dynamic geofences to prevent UAVs from approaching intruders that have been detected.

Technically speaking, the KB is a PostgreSQL database synchronized inside the fleet thanks to a communication layer that can retrieve missing message when a UAV was not able to communicate for a bit. In order to ensure consistency of the database content across the fleet, a centralized approach has been used. All the update and messages are sent to a central server (located on the GCS but could be anywhere as long as it is always reachable through the network) which validates each one. Once validated a message is broadcasted inside the fleet.

Keywords: Operations management, Vehicle to Vehicle communication, Vehicle to Infrastructure communication, Detect and Avoid, Planning and Scheduling, Geofencing, Swarm formation and cooperation, Information sharing, Fleet coordination.

6.15.2 State of the Art

The centralized-database approach has been used in robotics architecture for years. More recent architecture proposes a decentralized approach. However, the centralized approach we chose allows for simpler hence more convenient and robust implementation. For instance, the consistency algorithm in centralized approach is straightforward while it requires complex computation in distributed approaches.

As stated, the centralized approach is more convenient to implement, indeed there are more off-the-shelf solutions since we can rely on many solutions that work with a server client method. All the agents can be seen as clients and the central point as a server. It is important since it allows to benefit from the extensive list of (centralized/server) database implementations and modules. For instance, on top of the PostgreSQL, the KB is using a Geographic Information System to tackle the manipulation of GPS data and their projection in different coordinate systems (e.g. WGS84, …) simplifying the integration of different technologies using different representations.

6.15.3 Improvements

The first and main improvement is to allow for more types of agents. The goal is to offer two main classes of agents: mobile agents and static agents. The static ones correspond to hardware and software components that are meant to provide information to the system (e.g. weather station) or actuate with low or no movement (e.g. access-control barrier). The mobile agents represent all the robotic systems, UAVs and UGVs in the Use Cases.

This requires to extend the current definition of an agent in the system and to correlate it to the rights it has on the database (read and write updates, send orders …). If the number of agents increases a lot, it is also necessary to setup messages priority in order to sort the most important messages to ensure they are shared before the least important ones.

The other improvement is to connect the KB to an Air-Traffic Management system (ATM) and Unmanned-aircraft Traffic Management system (UTM). This would allow the fleet to take into account aircraft foreign to the system. Furthermore, the fleet could provide its flight paths ensuring a better integration in the airspace removing the need for a segregated airspace for the operations.

6.15.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Easy to integrate a new agent to a system	Number of lines of code necessary to add an agent (mobile or static)	SC1.1, SC1.2	O1
Reduce risk by more efficient and trusted information sharing in the fleet	No data loss and smaller latency on important messages (compared to less important ones)	SC2.2	O2
Easy integration of new components providing data to share through the database	Number of lines of code necessary to integrate the data sharing of the new component	SC1.1, SC1.2, SC4.1	O1, O4

6.16 Video and Data Analysis Algorithms - AI

6.16.1 Component Description

Aitek will define and implement a SW component during Comp4Drones. It will consist of video and data analysis algorithms to be used in the scope of the Smart Agriculture use case. It will be based on Deep Learning approaches and will be used to process RGB (mainly) and infrared (eventually) images. The results of this processing will be combined with some further heterogeneous data collected by the sensors installed on the drone.

Drones properly equipped with a camera and a computational unit can be used to collect and process images to detect relevant information in real time. Starting from a consolidated experience in video processing, Aitek contribution will be focused on the application of these technologies in the context of smart agriculture to monitor crop and plant status.

Deep Learning algorithms can detect several different information, after a proper training phase during which they learn what they must be able to do. Two main functionalities currently under definition are:

Enable precise tree crowns definition for water management and harvest forecast.

Monitoring and processing to detect exact location of un-healthy crops/plants

Both functionalities can be done either on-board in real-time either in post processing modalities on the ground station.

As explained in the next section, the (large) computational resources needed to run such algorithms may represent a limitation to their widespread application, particularly in scenario with severe constraints as in this case. Algorithms optimization is needed to find a good compromise between the needed resources and detection performance. This point represents a crucial and an innovative contribution with respect to the state-of-the-art approaches.

Moreover, in order to execute the task listed above, information detected by the drone should be associated with other heterogeneous data (at least the position where the information has been collected). The software module should be able to acquire also data from different on-board sensors and to add to the metadata produced by the video processing

Crown definition

As for the calculation of the volume of the plant, we are studying and defining a solution that consists in the three different steps. It is worth noticing that such methods consist of a video processing approach and the correlation of other data collected by the drones. It will be more robust and precise with respect to a solution based only on video processing. Here below are reported and briefly described the different steps of the proposed solution.

A convolutional neural network properly trained in order to do image segmentation. In this way it will be possible to process the images collected by the drone and recognizes pixel by pixel plants and background.

Once the image has been segmented, it will be possible to compute dimension of the area occupied by the plant.

Of course, this measure will be done in "pixels". Therefore, it is needed to “convert” this measure in meters, in order to provide as a final output an estimation of crown volume. To perform this conversion the drones should know roughly its distance from the plant, I can get an estimate of its volume.

Some further data must be taken into consideration, in particular regarding the camera installed on the drone to collect image. Moreover, the drone should fly slightly higher with respect to the plant during the image acquisition campaign. Drone altitude from the ground and the angle of camera tilt should be known as well; having such data it will be possible to have a more precise calibration and consequently a more robust crown measure. Instead, if the drone flies parallel to the plant, it is difficult to compute such estimation.

Starting from this general methodology proposed, our algorithms will need a specific characterization based on the type of plant we will monitor.

Detect exact location of un-healthy crops/plants

In order to detect location of un-healthy plants we are currently studying and defining a video analysis solution based on Deep Learning. Different alternatives are under evaluation and so far, Convolutional Neural Network (CNN) represents the best one also in this case. There are different types of neural network architecture, the most suitable one will be chosen during the project.

According to the type of diseases to be recognized there will be different possibilities:

In case the system has to recognize certain surface features in the plant (e.g. spots, dots, discolorations) it will be possible: i) to detect the individual features (bounding box style) or ii) to apply the segmentation, in order to have a pixel-by-pixel detection of possible disease.

Instead, if the disease determines a different conformation of the plant (e.g. shriveled leaves) it is possible to use a neural network properly trained to do a simple classification. In this case the output of the system will be directly a categorical data of the plant like “sick / not sick" with an accuracy level associated to such classification.

Moreover, both the aforementioned methodologies can be improved by adding another network in charge of a prior segmentation that distinguishes the plant from the rest of the background.

Keywords: Video analysis, Artificial Intelligence, Deep Learning, Convolutional Neural Network, Data fusion and processing.

6.16.2 State of the Art

Together with the latest data fusion techniques, the computer vision models have proven to be a key component within the field of industry 4.0, among which various applications can be found in smart agriculture scenarios. Whether these technologies are used to support existing use cases, such as precision agriculture, taxonomic identification, food quality and plants health monitoring, or for more innovative purposes in the case of data-driven productivity and genetic forecasting, the latest AI networks based on Machine Learning (ML) algorithms provide new and useful ways to gather information, enabling also scalability and reuse of data for the reference scenarios.

The current State Of The Art (SOTA) for such systems relies on heterogeneous data acquisition and remote sensing techniques, empowered image analysis methodologies and neural network models based on Deep Learning (a branch within the ML ontology), which provides higher performances compared to the traditional video analysis approaches.

Typically, these ML algorithms are applied into 4 steps in order to come out with a trained network:

- **Pre-processing**: a preliminary set of image analysis (i.e. image cropping, contrast enhancement, grey scale conversion) and data augmentation (i.e., rotation, image scaling and noise addition) techniques are used to ease the upcoming phases and to properly organize data in order to achieve more reliable outcomes.
- **Segmentation**: in order to simplify the image analysis process, digital images are divided in sub-sets of pixels and then labeled to identify common features shared by the various segments which are then used to locate objects and boundaries. Semantic segmentation (or dense prediction) allows classifying all the pixels into “human-readable” classes of objects, while Instance segmentation identifies all the image instances of every object. The traditional image segmentation techniques (i.e. K-means clustering, thresholding, Edge detection) have been superseded by the modern models based on deep learning technologies, such as Convolutional Neural Networks (CNN), Fully Convolutional Neural Networks (FCNs), DeepLab, Ensemble learning or SegNet.
- **Feature extraction**: this procedure aims to reduce the number of features within a dataset and creates a new set of derived values meant to be non-redundant and more informative.
- **Classification**: all the outcomes deriving from the previous steps are used to build classifiers by applying ML models (linear and nonlinear), such as CNN, Kernel Support Vector Machines (SVM) or K-nearest Neighbors (KNN).

One of the problems related to this kind of learning is that a good Neural Network heavily relies on a valid dataset so to be able to apply all the above procedures, which in the past was seen as an obstacle due to the lack of labeled resources, but thanks to the raise of the applications of the ML techniques there are now many new reserves to rely on (such as PlantCV for green use cases).

The Hardware evolution also improved the outcomes of the computer vision techniques which now use RGB cameras, Near Infra-Red (NIR) and Infra-Red (IR) Cameras, multispectral and hyperspectral sensors, LIDAR and SAR to acquire images and data from the field.

6.16.3 Improvements

Aitek has a large experience in video processing in particular for what concerns the use of state-of-the-art approaches based on deep learning. During this project, particularly in the scope of the Smart Agriculture use case, we are working to improve currently available solutions, targeting the following objectives:

- Algorithm optimization in order to find a best compromise between performance and computational resource needed.
- Integration of heterogeneous information (GPS position, altitude from the ground and other data) collected by the onboard sensors
- Chance to enlarge specific Datasets to rely on for future applications and reuse of data
- Field testing to identify which solution is best-fitted for this market segment as for HW (i.e. IR-NIR cameras, hyperspectral sensors)

6.16.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Improvements in detection performance using an AI-based computer-video approach	Detection Accuracy	SC4.1	O4
Computational resources optimization	Algorithm execution time Computational load Maximum supported FPS	SC1.2	O1

7 Payload Technologies

The drone system includes a set of payload technologies to support the drone mission specific operations. The payload technologies that will be focus in the project includes: hyperspectral camera and avionics encoders.

7.1 Hyperspectral (HSI) UAV Payload - IMEC-BG

7.1.1 Component Description

Description: Hyperspectral hardware payload

Hyperspectral cameras can improve detection of material imperfections. The hyperspectral payload will be based on imec's UAV platform: (dual) mosaic sensors/cameras with Ximea breakout board and Jetson TX2 board. Regarding the software blocks, we will reuse Airobot's server-based interface for ground controller with imec's camera commands.

Keywords: Hyperspectral Imaging, Multispectral Imaging, Camera, Payload,

7.1.2 State of the Art

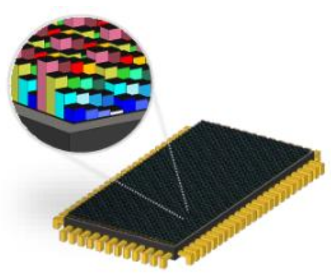
Currently, there are no real lightweight hyperspectral UAV cameras which have more than 4/5 bands. Such a camera would be a real breakthrough in the domain of UAV precision agriculture. Parrot's sequoia multispectral camera with about 4-5 spectral bands is the leading state of the art in this domain[59]. However, with 4-5 spectral bands only simple agriculture indices like NDVI can be extracted. Tetracam's 3-filter camera or multi-camera systems supporting up to 12 bands are other alternatives[60]. However, multi camera systems lead to much more bulkier systems with additional complexity of software to register images from different cameras to obtain the same spatial field of view, which could potentially lead to loss in image quality. For our target applications more spectral information would be required (>10 bands in VISNIR) to provide accurate diagnostic and actionable information. Our proposed camera can enable such applications. Headwall's micro-hyperspec[61] is another camera intended for UAV platforms, which uses conventional grating-based solutions for the spectral unit. This leads to a bulkier camera than our proposed solution, making this unsuitable for lightweight drones. Micro-hyperspec cameras can weigh up to 1kg or more, making this perhaps more suitable for larger drones/UAVs.

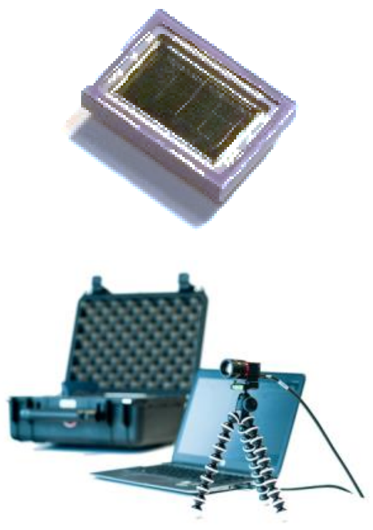
7.1.3 Improvements

IMEC-BG has developed a unique integrated hyperspectral filter/imager technology, where the spectral filters are monolithically deposited/integrated on top of CMOS image sensors at wafer level. The materials of the filters are chosen such that they are compatible with the production flows available in most CMOS foundries. This is achieved using a set of CMOS compatible production steps, like

[59] Parrot's sequoia camera: https://www.parrot.com/us/Business-solutions/parrot-sequoia
[60] Tetracam micro-mca: http://www.tetracam.com/Products-Micro_MCA.htm
[61] Headwall micro-hyperspec: http://www.headwallphotonics.com/blog/bid/336623/Hyperspectral-Sensors-for-UAV-Applications

deposition, patterning and etching, which allows pixel level accuracies in filter alignment. The result is a compact & fast hyperspectral imager made with low-cost CMOS process technology. This technology has been demonstrated on multiple instances, few of them are shown below and an overview is shown in the link: https://www.imec-int.com/en/hyperspectral-imaging


(a)


(b)

(c)

Spectral range	470-620nm/630-1000nm
# Spectral /bands	16/25
FWHM	< 10nm, using collimated light
Filter transmission efficiency	~ 85%
Imager	CMOSIS CMOS CMV 2000 imager
Imager resolution	2Mpixel
Resolution per tile/band	512 x 272 pixels 410 x 218 pixels
Frame rate in # hypercubes/sec	340

Figure 54: (a) concept of mosaic layout (filter heights exaggerated for illustration), (b) a packaged mosaic sensor and a USB3 hyperspectral camera (c) on-market XIMEA camera integrated with IMEC sensors; Table with key specifications of tile based spectral imager

This hyperspectral UAV camera will be integrated with other compute enabled features of the AiroCore platform to provide a complete UAV hyperspectral payload that is light weight and spans visible and NIR spectral ranges (450-970nm) with about 32 spectral bands. An overview of the architecture is shown in the figure below.

The hardware hyperspectral payload can communicate with the AiroCore platform. Making a rugged box: hyperspectral measurements for detecting and quantifying corrosion damage on offshore structures in realistic conditions (wind up to 5 Beaufort, wave heights of 1,2-1,5m, minimum temperature of -10°C).

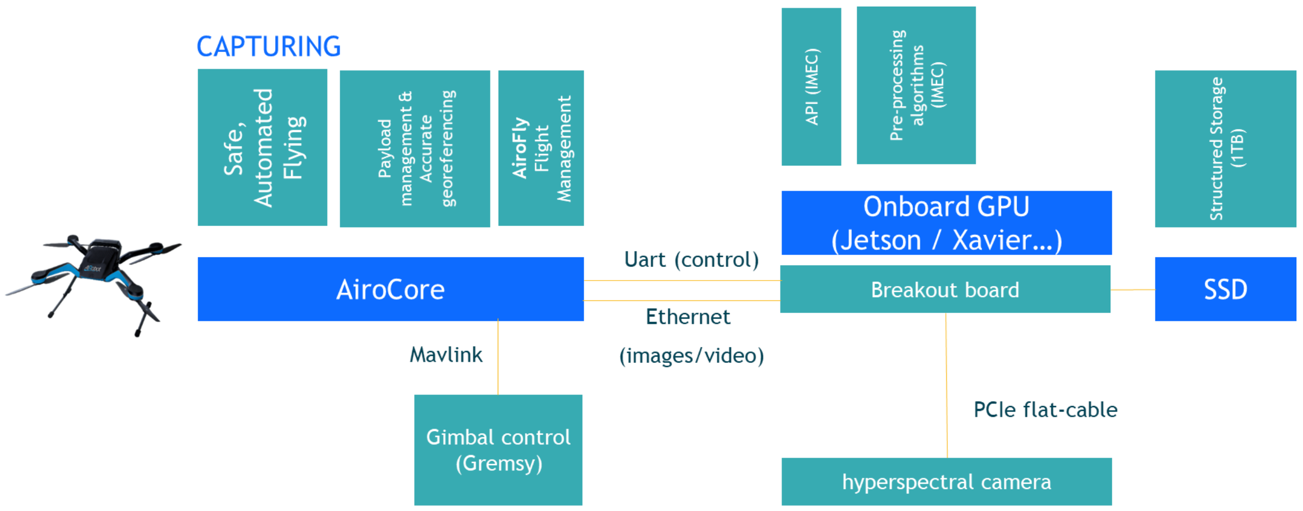


Figure 55: System architecture of UAV payload with compute enabled system

7.1.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Hyperspectral data acquisition	Total number of spectral bands acquired in VISNIR range >= 16 Spatial resolution of individual frame per spectral band >= 512 x 272pixels	SC1.1, SC1.2	O1
Onboard processing & storage capability	Computing platform capable to store data of >= 500GB And platform powerful to provide real-time computation capability >= Jetson TX2 like	SC2.1,	O2
Compact HSI camera module	Overall dimensions and weight of the hyperspectral payload to be as small as possible (< 10cm^3 &< 1Kg including optics)	SC3.1	O3

7.2 Camera - Payload (Single Visible, Infrared and Dual HD) - INDRA

7.2.1 Component Description

The payload of the Mantis system has been designed by Indra and consists of 2 main subsystems: the camera and the target system. In this case, three different payloads and lenses will be installed in the camera subsystem: visible optics, infrared and dual optics with HD resolution.

Keywords: Image Processing

7.2.2 State of the Art

The target system consists of a 3-axis gyro gimbal on which the camera is mounted. Each of the 3 axes is driven by a separate brushless motor (see Figure 56). The motors are controlled by a COTS controller card.

The controller measures the angles that the mobile camera forms relative to the fixed frame through 2 IMUS of 6 degrees of freedom (Inertial Measurement Unit): one is included within the controller's own PCB and the other is mounted on the camera's mobile housing and moved next to it. Each IMU is equipped with MEMS accelerometers and gyroscopes that measure the angle that forms the vector of gravity relative to them and the rotational speed. From this data, three-dimensional euler angles are calculated.

The horizontal angle of azimuth or "yaw" cannot be calculated through the gravity vector because it is perpendicular to this plane. Therefore, the camera is equipped with a magnetic encoder to calculate the yaw angle relative to the fixed frame.

The controller uses these angles to perform PID control loop calculations and moves brushless motors using a three-phase current. This gimbal control method allows for much faster and more continuous control over PWM-controlled servo motors.

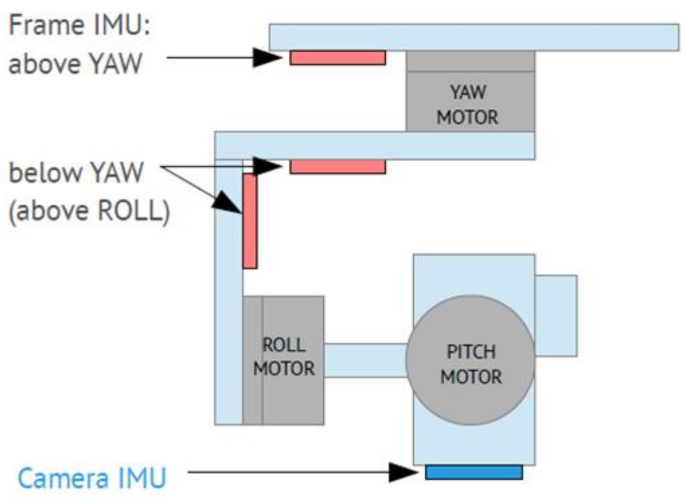


Figure 56: Payload of the Mantis system

7.2.3 Improvements and Metrics

The improvements to be made to have the single visible, infrared and dual payload HD will be the replacement of the current camera with analog resolution with a COTS camera with HD resolution with weight and dimensions similar to analog. To ensure its feasibility, once the candidate cameras have been analyzed, the camera's fixing system must be checked and all the internal wiring redesigned to allow the digital video to be obtained at its output. Finally, it must be ensured that the weight and the center of gravity of the total payload are within the established parameters. It will avoid, as far as possible, having to redesign the fairing of the payment charge.

7.2.3.1 Requirements

- Payload must be gyro-stabilized in 3 axes.
- Optics and gyrostabilization system must be non-dependent modules.
- Airplane's center of gravity should not vary with new payloads.
- Current payload fairing should be respected for new payloads.

7.2.3.2 Metrics

KPI	Metrics	Success Criteria	Project Objective
Interoperability between different camera types and image resolutions	Lines of code (minimum) Parameterization	SC1.1, SC1.2	O1

7.3 Avionics Encoder - INDRA

7.3.1 Component Description

Encoder for the reception, treatment and parameterization of the video received from the 4 HD optics and tracking features.

Keywords: Image Processing

7.3.2 State of the Art

The MANTIS System video encoder is a next-generation COTS element designed for mini-UAVs due to its reduced weight and size versus its high performance

7.3.3 Improvements and Metrics

Allow the encoder to be enabled for the reception and treatment of HD video from the 4 new HD-optics for which it will be necessary to parameterize them taking into account the characteristics of each of them.

* Tracking: settings, configuration and tests that allow improving the current tracking features available for SD video by applying them to the new HD video.

7.3.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Maximize reuse of analog payload architecture	Maintain same internal HW architecture with gyrostabilizer (minimum changes) and external (reduce exterior redesign – fairing)	SC1.1.	O1
Reuse OEM-related application components	Reuse camera communication protocols (SW), minimize code lines	SC1.2.	O1

8 Hardware Platform

As an execution environment, the project will provide improvements including onboard programmable and reconfigurable compute platform, SoM module, heterogeneous SoC-based reference platform, etc.

8.1 Onboard Programmable and Reconfigurable Compute Platform - UNIMORE and UNISS

The demand for onboard computational power of modern drones is exponentially growing due to the ever-increasing request for autonomous operation. To satisfy this request, more powerful (in terms of operational throughput) compute platforms must be embedded on the drones, while at the same time maintaining operational constraints related to the power envelope, the form factor, and the interoperability and connectivity with standard drone software stacks.

Modern heterogeneous systems-on-chip (SoC) tackle this challenge by exploring options that combine traditional CPU scalar processing with VLIW/SIMD processing (DSP, GPU) or FPGA logic. FPGA-based, commercial-off-the-shelf (COTS) SoC such as Xilinx's Zynq, Ultrascale+ or Versal, combine the flexibility of a general-purpose, multi-core ARM CPU, to the efficiency of custom-designed hardware IPs. The peak performance, verifiability and certifiability of FPGA design flows make these devices a very appealing candidate for an onboard drone compute platform.

However, the more heterogeneous components are integrated into any compute platform, the more complex the integration process becomes in terms of hardware-software interactions, access to shared resources, and diminished regularity of the design. The critical challenges are in the integration with the legacy software components, the programming interfaces, and the management of many heterogeneous components more than in the design of any individual components.

The proposed solution for a low power, high performance onboard drone compute platform consists of a FPGA-based design methodology rooted on three key components:

- An Open-Source FPGA overlay based on a compute cluster integrating RISC-V cores and low-latency, high-bandwidth shared memory as a standard interface to application-specific hardware processing elements (HWPE). This enables plug-and-play deployment of HWPEs in typical drone workloads [UNIMORE];
- A methodology and tool (the Multi-Dataflow Composer, or MDC) for the design of Coarse-Grained Reconfigurable Co-processing Units, which constitutes the basis for the definition and integration of HWPEs in the overlay compute clusters [UNISS];
- A high-level programming model, with associated runtime and tools for the offloading and local orchestration of the HWPEs. This last component is developed as part of WP6 and detailed in D6.1 [UNIMORE-UNISS].

8.1.1 Open-Source FPGA overlay with RISC-V compute clusters [UNIMORE]

As shown in Figure 57, the proposed overlay targets COTS FPGA-based SoC. The host processor, an industry-standard, multi-core ARM Cortex-A CPU, is connected by a set low-latency, high-bandwidth AXI interfaces to a large Programmable Logic block, implemented as FPGA. The two subsystems communicate via shared main LPDDR memory. ARM Cortex R5 cores, typically available in this type of SoC, could be used for timing-critical operations.

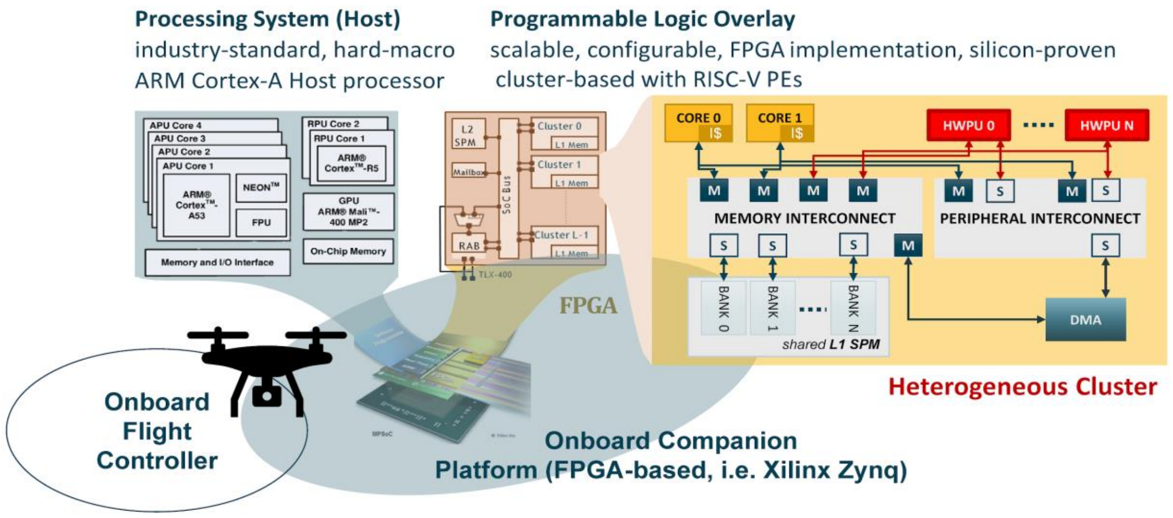


Figure 57: The proposed overlay targets COTS FPGA-based SoC

The hard-macro ARM Cortex-A CPU is capable of running full-fledged Linux OS, and support out-of-the-box execution of the whole communication software stack for integration with the drone flight controller system (i.e. MAVLink, or ROS).

On the FPGA subsystem, the overlay soft IP is designed as one (or more) compute cluster, where a small (configurable) number of RISC-V soft-cores share a local, fast memory, to which a set of HWPEs can be attached. This solution enables: (i) to tightly control the accelerators, avoiding the high overheads and the high-latency operations associated with CPU-to-PL (i.e., FPGA-accelerators) communication (i.e., Linux Driver Pass through, and/or Syscall); (ii) to replace complex FSM control logic for HWPE operation with software routines, executed on the RISC-V cores; (iii) to allow for easy offloading of such control code (the aforementioned software routines) from a standard programming model such as OpenMP. RISC-V cores can also flexibly execute any glue-code and data marshaling operation for connecting the accelerated functionality and the high-level flight controller devices and onboard sensors, without the need for changing the platform design.

To enable plug-and-play integration of custom HWPEs in the proposed overlay, a generic wrapper IP will be defined for interoperation with the MDC design flow (see below)[62]. The wrapper enables the seamless integration of the HWPEs such that designers can execute the original software application as-is.

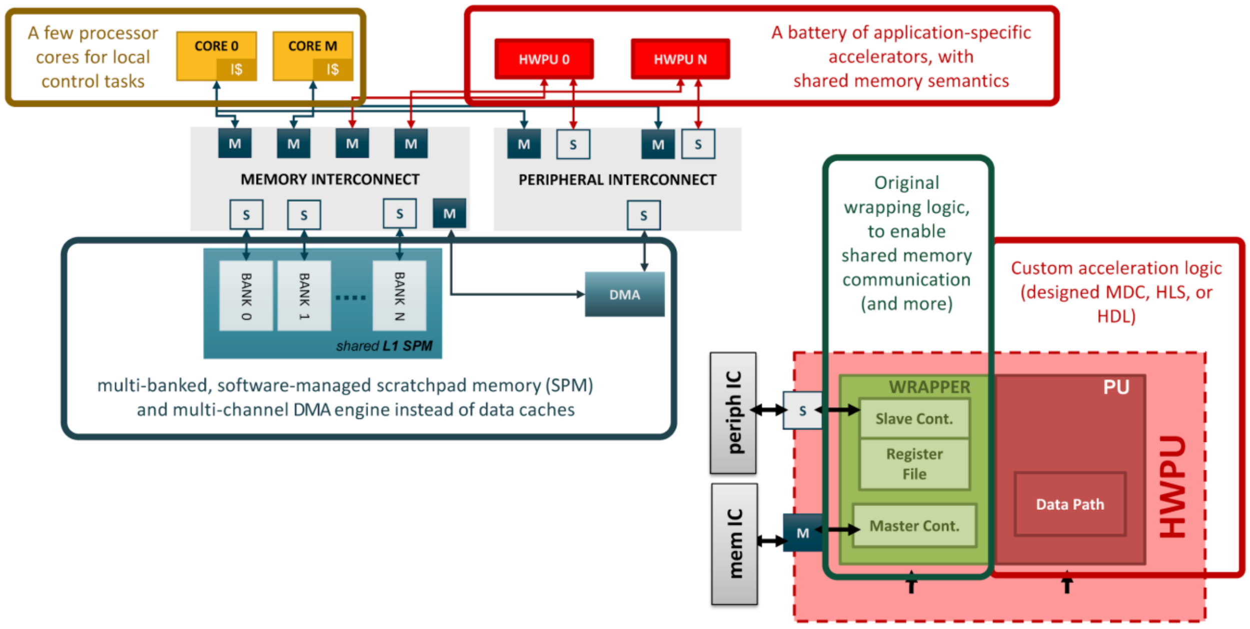

A few processor cores for local control tasks
CORE 0
I$
CORE M
I$
HWPU 0
HWPU N
A battery of application-specific accelerators, with shared memory semantics
M
S
MEMORY INTERCONNECT
PERIPHERAL INTERCONNECT
BANK 0
BANK 1
BANK N
shared L1 SPM
DMA
multi-banked, software-managed scratchpad memory (SPM) and multi-channel DMA engine instead of data caches
Original wrapping logic, to enable shared memory communication (and more)
Custom acceleration logic (designed MDC, HLS, or HDL)
periph IC
mem IC
WRAPPER
Slave Cont.
Register File
Master Cont.
PU
Data Path
HWPU

Figure 58: The internals of the overlay cluster

Figure 58 shows in detail the internals of the overlay cluster, as well as a high-level block diagram of the wrapper for HWPU integration. The logic related to the accelerated functionality - generated via MDC - is isolated inside the data path (PU), making the design of such block agnostic to the surrounding system. The wrapper implements the connectivity and the control interface that is necessary to offload computation to the HWPU.

The proposed overlay will be highly configurable. Number and type of RISC-V core (with/w/o FPU, DSP extensions, SIMD, etc.)[63], memory size and banking factor, number of HWPEs and number of clusters can be defined via parameters (based on the resources available in the target FPGA SoC).

Programmability of the resulting compute platform will be built on top of the OpenMP4.0 accelerator model[64] to effectively offload computation from the ARM CPU host the RISC-V cores. Locally, the RISC-V code relies on a dedicated API to control and orchestrate HWPU execution and access to local and global memory.

Previous work has highlighted the potential of this type of shared-memory accelerator paradigm[65], confirming the potential of the approach also in the context of autonomous drones. The starting point for designing the overlay component is the HERO platform[66]. HERO is an emulation platform of anSoC featuring traditional host CPU plus programmable many-core accelerators based on the PULP platform (pulp-platform.org). HERO relies on FPGA SoCs to provide physical host CPU and a multi-cluster,

[62] The *wrapper* and the MDC tool could allow seamless integration with accelerators developed with any methodology (HLS, or HDL), or with existing accelerators, by simply choosing among a set of bus-standard adapters to interface it with the *overlaycluster*.
[63]Rossi, D., Conti, F., Marongiu, A., Pullini, A., Loi, I., Gautschi, M., Tagliavini, G., Capotondi, A., Flatresse, P. and Benini, L., 2015, August. PULP: A parallel ultra low power platform for next generation IoT applications. In 2015 IEEE Hot Chips 27 Symposium.
[64]Capotondi, Alessandro, and Andrea Marongiu. "Enabling zero-copy OpenMP offloading on the PULP many-core accelerator." Proceedings of the 20th International Workshop on Software and Compilers for Embedded Systems. 2017.
[65]F. Conti, C. Pilkington, A. Marongiu, L. Benini, He-P2012: Architectural Heterogeneity Exploration on a Scalable Many-Core Platform, 2014 IEEE 25th International Conference on Application-specific Systems, Architectures and Processors (ASAP) – Best paper award
[66]Kurth, A., Capotondi, A., Vogel, P., Benini, L., & Marongiu, A. (2018, November). HERO: an open-source research platform for HW/SW exploration of heterogeneous manycore systems. In Proceedings of the 2nd Workshop on AutotuniNg and aDaptivityAppRoaches for Energy efficient HPC Systems (pp. 1-6).

many-core RISC-V accelerator deployed on the PL. In COMPDRONES, HERO clusters will be re-designed to implement the proposed overlay and accelerator wrappers.

8.1.2 Coarse-Grained Reconfigurable Co-processing Units [UNISS]

UNISS has planned to release reconfigurable co-processing units, suitable to execute the different functionalities over a virtual reconfigurable platform. Currently, these kinds of accelerators are derivable with the Multi-Dataflow Composer (MDC) tool, which is able to define such a virtual reconfigurable substrate starting from a set of dataflow descriptions of the applications/tasks to be accelerated. Moreover, the tool offers support for the deployment of Xilinx compliant IPs, ready to be used in a processor-coprocessor system.

The Coarse-Grain Reconfiguration (CGR) offered by MDC is a virtual in the sense that resources are always available in the accelerator, and they are multiplexed in time according to the identifier (ID) of the selected operation, to be properly driven by the user. Reconfiguration is as fast as the transmission of a new ID to the co-processing unit, and within the reconfigurable computing, core takes place in a single clock cycle. Techniques for optimizing the co-processing unit deployment, in terms of resource usage minimization, are put in place to allow optimal resource efficiency, and in turn, minimizing both area occupation and power consumption.

CGR co-processing units supports two types of reconfiguration/ operations:

- Functional-oriented (Figure 59) – the accelerator embeds different functionalities (e.g. different image processing algorithms) that are based on a common set of actors.

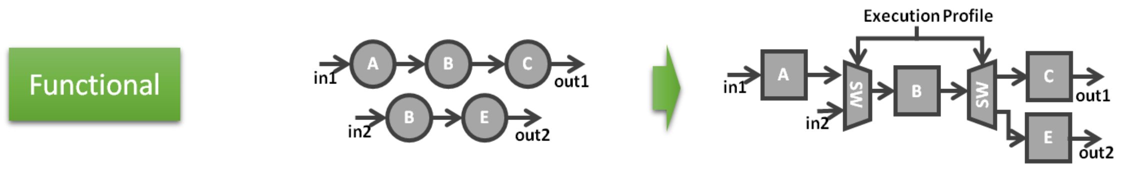


Figure 59: Functional-oriented reconfiguration/ operations

- Working point-oriented (Figure 60) – the accelerator is able to execute the same functionality, but with different trade-offs in terms of non-functional metrics (e.g. different image quality vs. power consumption profiles in encoding/decoding algorithms).

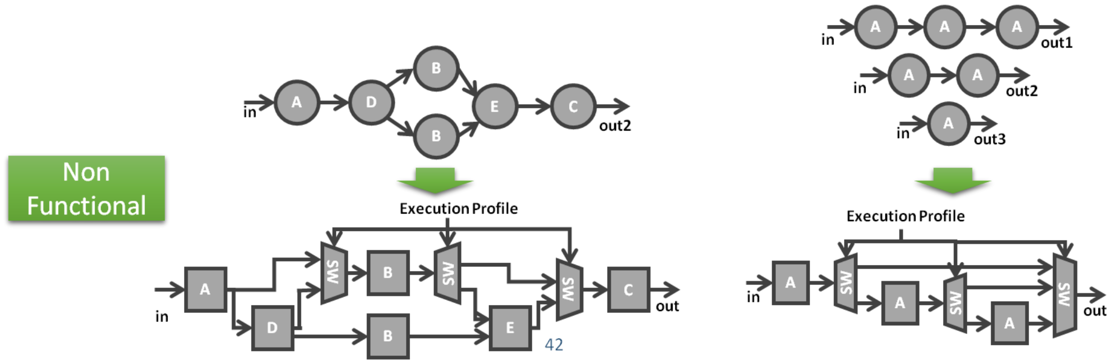


Figure 60: Working point-oriented reconfiguration/ operations

The flexibility of the CGR allows for the implementation of different applications and for the exploration and optimization of different metrics and achievements of different trade-off. Table 2 depicts an overview of different trade-offs on different applications implemented as CGR accelerators, using MDC[67].

Table 2: Overview of possible trade-off that can be achieved with coarse-grained reconfiguration on three different applications

[67] Palumbo F., Sau C., Fanni T., Raffo L. (2019) Challenging CPS Trade-off Adaptivity with Coarse-Grained Reconfiguration. In: De Gloria A. (eds) Applications in Electronics Pervading Industry, Environment and Society. ApplePies 2017. Lecture Notes in Electrical Engineering, vol 512. Springer, Cham

Design	Working points	Trade-off	Target Technology
HEVC	Number of taps involved in the computation	**Energy vs. Quality** High quality - 8 taps luma and 4 taps chroma. Low Energy - 3 taps luma and 2 taps chroma.	Artix-7 FPGA
JPEG decoder	Parallelism degree of critical computational steps	**Quality and Compression Efficiency vs Dynamic Power Consumption and Execution Time**	Virtex-5 FPGA
CGR AES-128	Number of round "r" involved in the computation.	**Energy vs Throughput** r = 2: more energy efficient cypher but slower execution. r = 4: less energy efficient cypher and maximum throughput.	Artix-7 FPGA

Such a reconfigurable computing core, to be easily used in a heterogeneous computing infrastructure, has to be embedded in a co-processing unit. MDC tool is currently capable of automatically composing, synthesizing and deploying runtime reconfigurable coprocessors compliant with the Xilinx Vivado design suite. The co-processing unit infrastructure is available (see Figure 61).

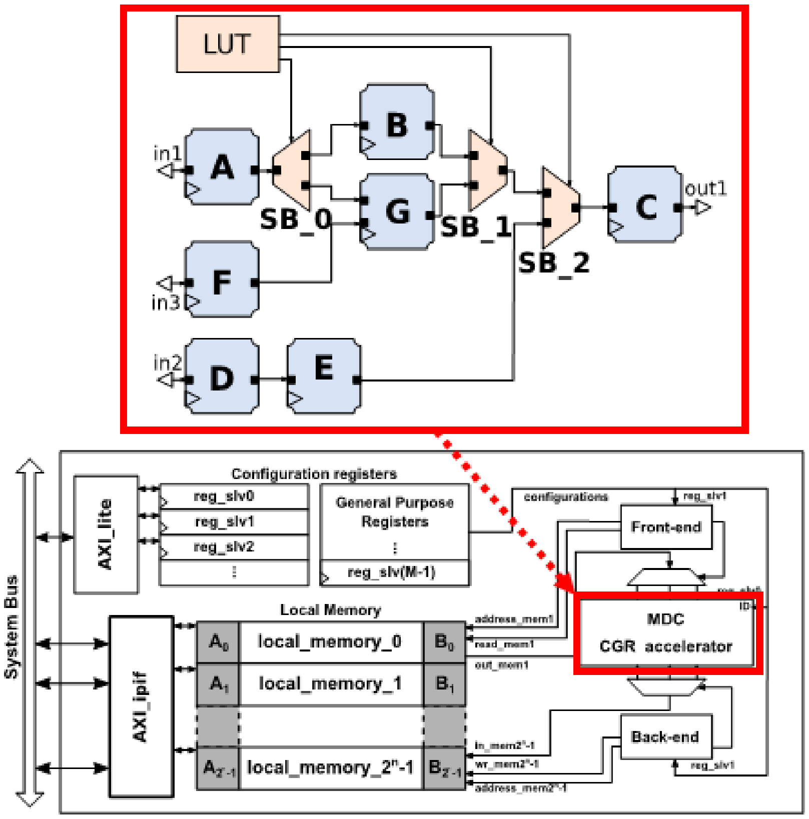


Figure 61: Co-processing unit infrastructure

Users can customize both the:

- CGR computing core (indicated as MDC CGR in the figure): giving as input the handshake communication protocol among the actors involved in the computation within the reconfigurable computing core.
- IP logic, choosing the
    - Processor: soft-core (MicroBlaze) or hard-core (ARM)

- Processor-coprocessor coupling: memory-mapped or stream-based
- Direct memory access: enabling its insertion.

This co-processing unit comes along with a set of APIs to facilitate delegation and retrieval of computed results.

Keywords: System and Environment State, Data fusion and processing, SoC, FPGA, Virtual Reconfiguration, Coarse Grained Reconfiguration, Co-processing Units, Dataflow MoC, Data-path Merging

8.1.3 State-of-the-art

Modern embedded applications require huge computing power, especially when AI workloads have to be served. These days, to satisfy such requirements while fitting tight power envelopes, standard general-purpose processors have progressively evolved into more efficient heterogeneous systems-on-a-chip (SoC), where they are coupled to Massively Parallel Processor Arrays (MPPAs), Graphic Processing Units (GPUs) or Field Programmable Gate Arrays (FPGAs). The same revolution is about to happen in the context of unmanned aerial vehicles (UAV), where state-of-the-art devices are being tasked with increasingly complex activities. Similar to other embedded systems, also in this context Application-Specific Integrated Circuits (ASICs) represent the optimal solution to deliver the required computing and power efficiency. However, full-custom designs imply deep hardware design knowledge, long time-to-market to go from design to chip deployment, and high costs due to low production volumes. Commercial off-the-shelf (COTS), FPGA-based heterogeneous SoCs might represent an optimal alternative in this scenario, as i) FPGA designs can be pushed to provide efficiency close to the ASICs, while at the same time offering flexibility and programmability[68]; ii) FPGA development in the context of COTS HW and associated SW stacks offer much lower design complexity and production cost; iii) FPGAs allow an efficient digital implementation of controllers.

FPGA platforms fall within the reconfigurable computing domain of digital electronic system architectures, which combine the flexibility typical of software programmed systems to the high performance of the hardware implementations[69]. Reconfigurable systems are often called adaptive, meaning that the logic units and interconnects of the system can be modeled to fit a specific functionality by programming it at the hardware level[70]. However, the more these components are able to fit the application requirements, the slower they are with respect to less flexible components, which can easily turn out to be also smaller in area and less power consuming[71]. CGR systems provide word-level reconfigurability and, despite being customizable over a smaller number of scenarios, they reconfigure faster than fine-grain ones. Moreover, CGR is less power-hungry than dynamic partial reconfiguration typical of FPGA substrate and is also prospectively suitable also for ASIC implementation even if in the context of Comp4Drones COTS FPGA-based heterogeneous SoCs platforms will be used. Usability of these kinds of platforms implies to address the different state of the art open issues:

- Hardware design, customization, and integration (in the COTS SoC) efforts should be minimized.
- From the application developer perspective offloading computations to hardware accelerators should be no more complex than programming a GPU with OpenCL or OpenMP.
- Concerning the first point, it is fundamental to
- Define an architectural template with clear interfaces for the HW accelerators to seamlessly communicate with the rest of the system,

[68] S.M. Trimberger, “Three ages of FPGA: a retrospective on the first thirty years of FPGA technology”, IEEE, 2015
[69] K. Compton, S. Hauck, “Reconfigurable computing: A survey of systems and software”, ACM Computing Surveys 34 (2) (2002) 171–210. doi:10. 1145/508352.508353
[70] R. Tessier, W. Burleson, “Reconfigurable computing for digital signal processing: A survey”, Journal of Signal Processing Systems 28 (1-2) (2001) 7–27. doi:10.1023/A:1008155020711
[71] T. Todman, G. Constantinides, S. Wilton, O. Mencer, W. Luk, P. Cheung, “Reconfigurable computing: architectures and design methods”, IEE Proceedings-Computers and Digital Techniques 152 (2) (2005) 193–207. doi:10.1049/ip-cdt:20045086
K Desnos, F Palumbo, “Dataflow modeling for reconfigurable signal processing systems”, Handbook of Signal Processing Systems, 787-824, 2019

- Develop methodologies and tools to simplify the design of the HW accelerators themselves.

In Comp4Drones to tackle these issues, we intend to leverage FPGA overlays and High-Level Synthesis (HLS). FPGA overlays are programmable hardware abstraction layers on top of FPGAs obtained by means of pre-implemented programmable components mapped on the available FPGA resources and serving both computing and routing functionalities[72]. The acceleration infrastructure can be easily coupled to an operating system running on general-purpose processors, which will dispatch tasks and housekeeping. In such template-oriented architectures, the operating system could manage hardware tasks in the same way it deals with software ones, solving the hardware-software interface issue. Then, HLS can be used to relieve designers from the burden of specifying the accelerators bitstream by introducing the possibility of programming FPGAs directly from high-level languages, such as C or C++[73].

Concerning the second point, the shared memory paradigm can be adopted for the design of the HW accelerator interfaces, which allows us to transparently wrap their operation within an abstract notion of a hardware thread. This further enables us to rely on high-level programming models and associated offloading mechanisms, like the already mentioned OpenCL or OpenMP, for the development of the target applications. Still, it is necessary to invest in the overlay infrastructure as well as in the compilation and customization tool chain to lead these approaches from being a promising solution to a concrete one.

8.1.4 Improvements

Joint efforts from UNISS, UNIMORE and UNIVAQ will contribute to the development of the proposed compute platform for autonomous drones, targeting ease of deployment of FPGA accelerators and programmability of the whole platform. Within the context of WP3, the proposed technology is defining an acceleration architecture template and infrastructure compliant with the overlay concepts and easily adoptable by HW accelerator developers, capable to answer to the need for embedding ever-increasing computational capabilities on SoC. Extensions/adjustments of the described component are needed to integrate the CGR accelerators within an overlay infrastructure.

For the acceleration architecture infrastructure, we propose solutions for drone design that leverage commercial off-the-shelf, FPGA-based heterogeneous SoCs (e.g., Xilinx Ultrascale+) for the deployment of a specialized (according to the needs of the service being deployed) instance of a cluster-based, many-core accelerator architecture template. Putting together UNISS and UNIMORE already available components, the building block of such a soft IP is meant to be a compute cluster, where several simple processing elements share local memory, to which dedicated HW accelerators can also be attached (e.g., for deep learning). HW accelerators are intended to be, where needed, CGR reconfigurable ones, implementing different working points (i.e. trade-offs among Quality of Service and Energy consumption) or functionalities (i.e. different convolutional kernels) on the same configurable substrate as discussed above.

8.1.5 Component Metrics

As we can see in the following table, the possible improvements that can be obtained with this kind of platforms in relation to the implemented applications and to the KPI relevant in the specific application

[72] X. Fang, S. Ioannidis, M. Leeser, “Secure Function Evaluation Using an FPGA Overlay Architecture”, ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2017
Fanni, A Rodríguez, C Sau, L Suriano, F Palumbo, L Raffo, E de la Torre, “Multi-grain reconfiguration for advanced adaptivity in cyber-physical systems” International Conference on ReConFigurable Computing and FPGAs, 2018
A. Brant, G.G.F. Lemieux, “ZUMA: An Open FPGA Overlay Architecture”, IEEE International Symposium on Field-Programmable Custom Computing Machines, 2012
D. Wilson, G. Stitt, “Seiba: An FPGA Overlay-Based Approach to Rapid Application Development”, International Conference on ReConFigurable Computing and FPGAs, 2018
[73] R. Nane, V. M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi, J. Anderson, and K. Bertels, “A Survey and Evaluation of FPGA High-Level Synthesis Tools,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, iss. 10, pp. 1591-1604, 2016

field. A set of KPI and metrics to measure the impact of the proposed technologies in terms of programmability and flexibility, as defined in the D1.1 (see UC5 Demo 1) are the following:

KPI	Metrics	Success Criteria	Project Objective
Easy to design and deploy	Number of lines of code (for HW design and system configuration)	SC1.1, SC1.2	O1
Easy to develop applications for heterogeneous platforms (FPGA SoCs)	Number of lines of code (for SW design and application design)	SC1.2, SC4.1	O1, O4
Improvements in performance and resource consumption	Application execution time and clock cycles and speedup achieved Latency for Offloading and Accelerator Reconfiguration time	SC4.1	O4

Note that, while in the WP3 the contributions focus on the design methodology of onboard computing platforms, based on FPGA SoC, the evaluation of the KPI can be measured only within the complete UC definition, and thus taking in account also the complementary contributions coming from WP4 and WP6.

Cooperation in WP4 will allow us to assess the proposed accelerators and the integration in computational hungry scenarios by developing an accelerator for AI/ML applications.

Cooperation with WP6 will tackle mainly the second SoA open issue discussed above by investing in automated strategies for the generation of these kinds of co-processing units and for task offloading.

8.2 HW/SW SoM Module - IKERLAN

8.2.1 Component Description

IKERLAN will develop a HW+SW SoM module based on a heterogeneous SoC such as XILINX ZU+MPSoC. That will enable to the HW/SW co-design of the system and accelerate functionalities in drones to improve its performance. The initial architecture is the one shown is Figure 62.

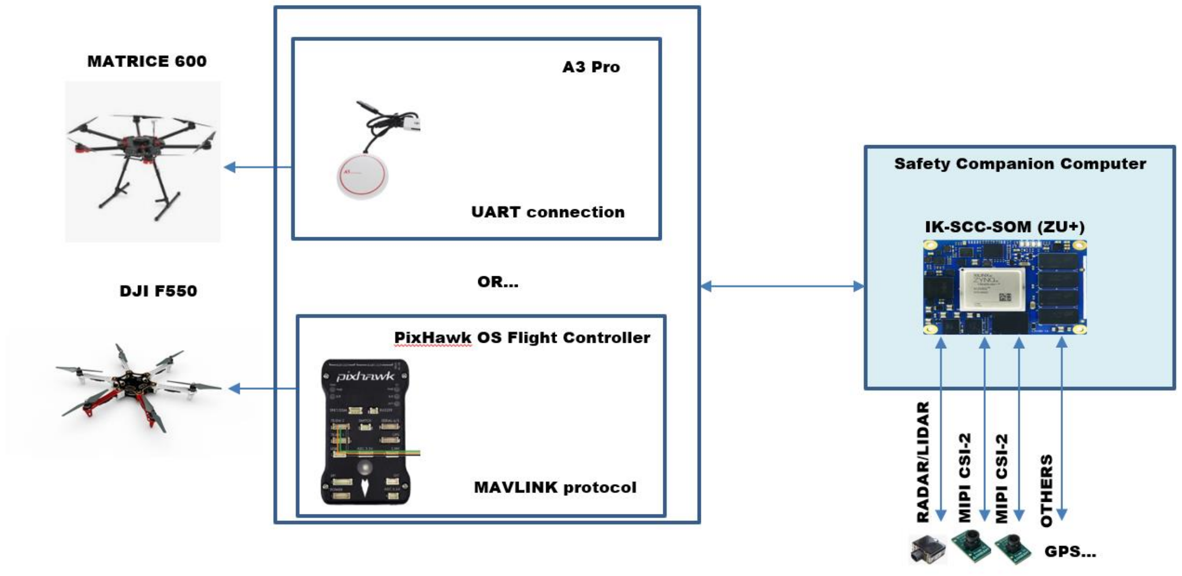


Figure 62: Initial composition of the SOM setup to be developed by IKERLAN

IKERLAN will develop both HW (SoM board IK-SCC-SOM) as well as the Sw and FPGA code to implement accelerators of a variety of functionalities such as objet detection and positioning.

IKERLAN will develop SoM module taking into account arising standards in the drone industry for the safety certification and will produce a Safety Concept to prove the certifiablity of the SoM module of a selected flight safety function.

Keywords: SoC, FPGA, safety, certification, image processing, positioning

8.2.2 State of the Art

From the certification point of view, Safety Certification Standards for drone industry have started to arise lately. The “AMC&GM to Commission Implementing Regulation” is a guide on which the operation of UAS is regulated according to the objective for which it has been conceived and one of the requirements of this guide is the SORA system (see the process in Figure 63). No implementations on HW/SW have been done yet of components that follow those regulations.

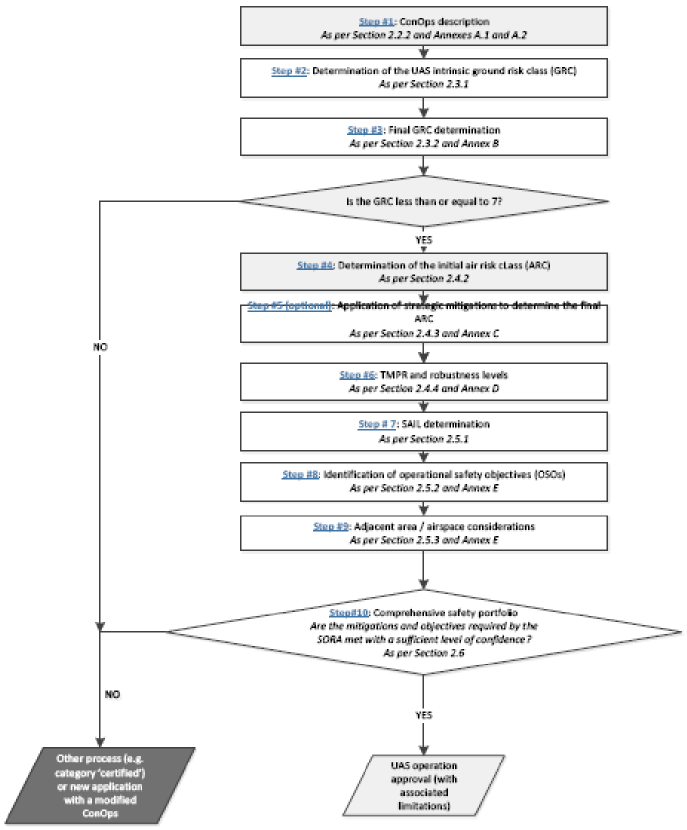

Step #1: ConOps description
As per Section 2.2.2 and Annexes A.1 and A.2
Step #2: Determination of the UAS intrinsic ground risk class (GRC)
As per Section 2.3.1
Step #3: Final GRC determination
As per Section 2.3.2 and Annex B
Is the GRC less than or equal to 7?
YES
NO
Step #4: Determination of the initial air risk cLass (ARC)
As per Section 2.4.2
Step #5 (optional): Application of strategic mitigations to determine the final ARC
As per Section 2.4.3 and Annex C
Step #6: TMPR and robustness levels
As per Section 2.4.4 and Annex D
Step # 7: SAIL determination
As per Section 2.5.1
Step #8: Identification of operational safety objectives (OSOs)
As per Section 2.5.2 and Annex E
Step #9: Adjacent area / airspace considerations
As per Section 2.5.3 and Annex E
Step#10: Comprehensive safety portfolio
Are the mitigations and objectives required by the SORA met with a sufficient level of confidence?
As per Section 2.6
NO
YES
Other process (e.g. category ‘certified’) or new application with a modified ConOps
UAS operation approval (with associated limitations)

Figure 63: SORA process

From the HW point of view several companion computers exist on the market, such as Raspberry Pi, etc. that makes an easy path to add sensor fusion and processing capabilities but do not have enough performance and are not designed with a safety standard point of view and therefore would not be prone to a safety certification process.

8.2.3 Improvements

No implementations on HW/SW have been done yet of components that follow those safety regulations. IKERLAN will analyses the HW and SW requirements for a drone module to match those arising standards and an architecture proposal and actual implementation will be done with the techniques to be implemented in order to achieve desired robustness level.

Also an implementation of those Safety Functions will be done in one ZU+ MPSoC processor that has a higher level of capabilities to run faster and more efficiently those traditional companion computers.

8.2.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Object detection and avoidance in SoM	Reuse of obstacle and trajectory tracking IP components in SoC based SoM, ease drone customization	SC1.2 and SC2.1	O1 and O2
Availability of pre-certified SoM module	Easier certification process, reduction in V&V	SC4.1	O4

8.3 Modular SoC-based embedded reference architecture - EDI

8.3.1 Component Description

EDI will develop an embedded heterogeneous SoC-based (FPGA + MPU) reference platform, which will be used to deploy components developed by EDI and other partners. The main purpose of this platform is to enable software and hardware (FPGA) co-design and deployment on a functional drone. Furthermore, the architecture will include the Linux-based board support package, software component intercommunication framework, methodology for algorithm partitioning between sequential and parallel processing paradigms and examples on setting up DMA-based communication with the FPGA accelerators.

To accelerate the electronics hardware development, an embedded SoC module will be utilized and a special carrier board will be designed and manufactured to deploy heterogeneous platform on the drone. The initial anticipated hardware setup is shown in Figure 64. This figure has an informative value and the hardware composition is still being improved. An additional advantage of utilizing the SoC module is that modules have standardized interfacing and represent a range in the price-performance relationship, which allows decreasing overall drone costs to the level of required performance. To comply with the requirements of indoor use case demonstrators, the drone itself is being built as well by aggregating different components complying with the size, energy and carrying requirements. The built drone is shown in Figure 65.

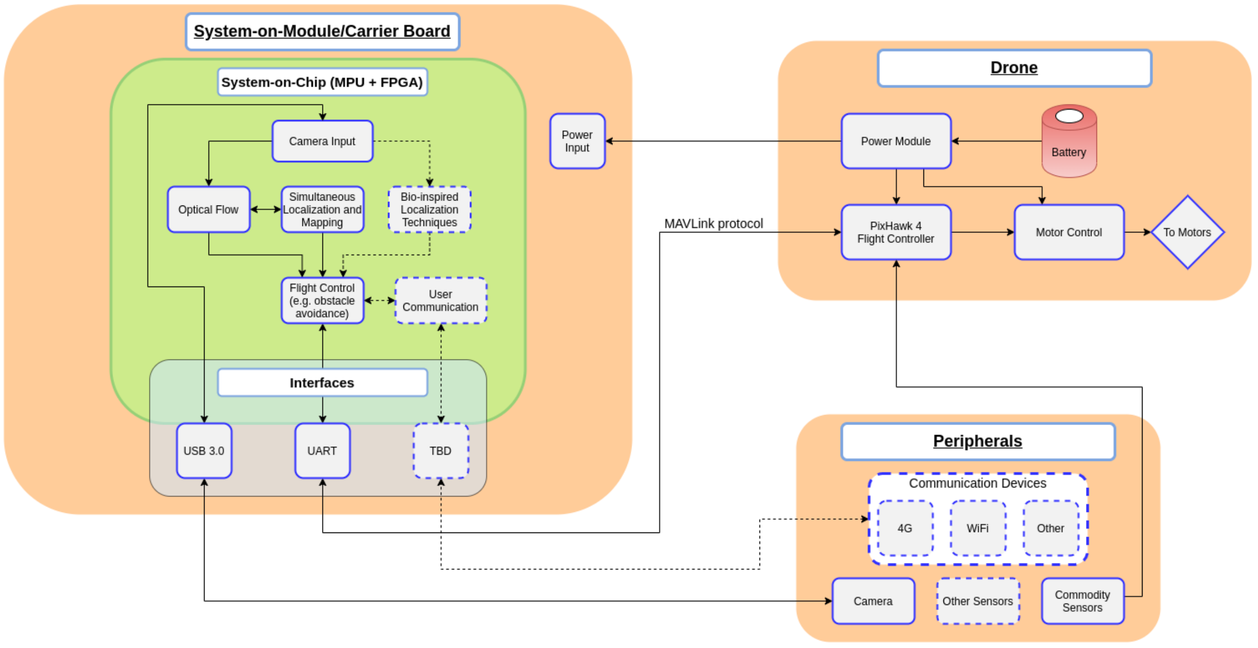


Figure 64: Initial composition of the SoC reference architecture’s hardware, carrier board and sensor setup.

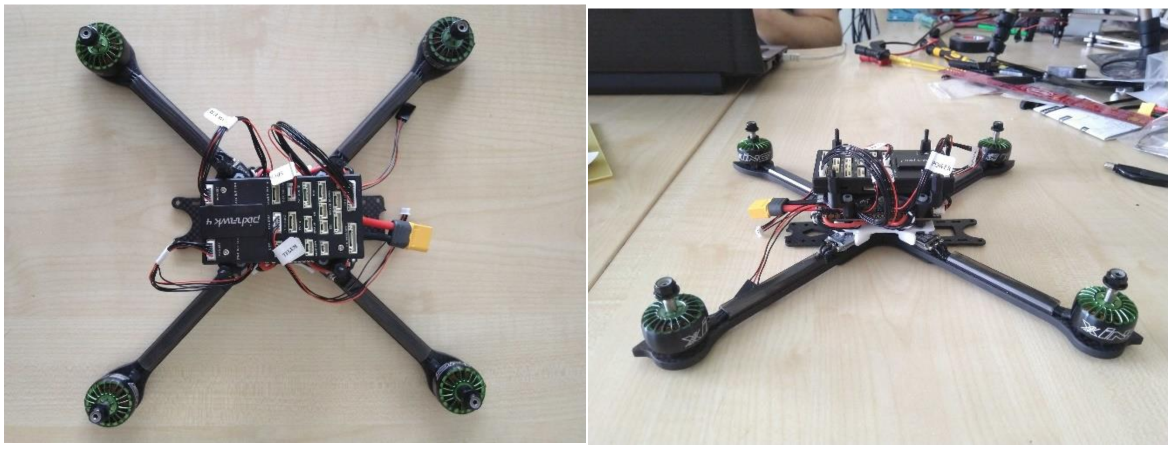


Figure 65: Built drone for the deployment of the SoC-based reference platform.

Additionally, this reference architecture will include a methodology on the algorithm separation between concurrent (digital logic) and sequential processing paradigms, a framework for managing and reusing different software components of the application and component intercommunication framework, including zero data copy communication.

Keywords: SoC, FPGA, Reference Architecture, Heterogeneous processing, Co-processing, Acceleration

8.3.2 State of the Art

The currently available embedded drone flight controllers and computational platforms are based on a sequential processing paradigm, which is a limitation to the drone's onboard computational capacity. Modern heterogeneous systems that incorporate different computational paradigms (MPU, FPGA, GPU) promise advantages of great computational capabilities and even better power efficiency with a drawback of greater system complexity. In the COMP4DRONES project, we will utilize our experience in heterogeneous SoC systems to develop anSoC-based embedded reference architecture, which would be the core of the autonomous flight controller.

8.3.3 Improvements and Metrics

EDI will enable:

c) Distribution of algorithms across heterogeneous processing paradigms.
d) Novel approaches to sensing and processing pipelines.

8.3.3.1 Requirements

- The embedded reference architecture shall be able to execute applications by utilizing concurrent and sequential processing paradigms.
- The embedded reference platform will be based on exchangeable SoC modules and a developed carrier board.
- The embedded reference platform’s carrier board shall incorporate USB3.0 interfacing capability.
- The embedded reference platform’s carrier board shall incorporate UART interfacing capability.
- The embedded reference platform’s carrier board shall incorporate SPI interfacing capability.
- The embedded reference platform’s carrier board shall incorporate I2C interfacing capability.
- The embedded reference architecture shall incorporate Linux-based board support package.
- The embedded reference architecture shall incorporate component management framework.
- The component management framework shall enable the reuse of software components.
- The component management framework shall enable a generic configuration of software components.

- The component management framework shall enable means of monitoring software components.
- The component management framework shall enable means of stopping/starting individual software components.
- The component intercommunication framework shall support push-pull communication paradigm.
- The component intercommunication framework shall support publish-subscribe communication paradigm.
- The component intercommunication framework shall support zero data copy communication.

8.3.3.2 Metrics

KPI	Metrics	Success Criteria	Project Objective
Ease of incorporating new software components	Number of files changed to incorporate new component	SC1.1, SC1.2	O1
Ease of configuring software components	Number of lines of code changed to incorporate new software component	SC1.1, SC1.2	O1
Ease of incorporating new hardware (FPGA) components	Number of lines of code necessary to communicate with the accelerator	SC1.1, SC1.2	O1
Efficiency of inter-component communications (software)	Latency and effective throughput of the communications framework	SC4.1	O4
Flexibility of the inter-component communications' framework (software)	Number of lines of code necessary to implement communication mechanism	SC1.1, SC1.2	O1

8.4 CompSOCHardware Platform - TUE

8.4.1 Component Description

Description: In a multi-processor System On Chip (SOC), applications with different requirements are executed on a heterogeneous set of resources and communicate through an interconnect such as Network On Chip (NOC). Proposing a SOC whose temporal behavior is easily predictable has been under research for a long time.

In a general view, there are two fundamental requirements for a hardware platform to be utilized in the mixed criticality environments in which various applications with different criticality levels run on the same resource. 1) Predictability: The temporal behavior of every instruction is bounded by the Worst Case Execution Time (WCET). 2) Composability: There is no interference between applications, meaning that each application is executed in an isolated virtual execution platform (VEP). In today's computers, the sources of unpredictability fall into two levels: Some design strategies like caching and speculation prevent predictable behavior and have to be corrected at the hardware level. Additionally, scheduling and resource management must be predictable at the software level. Regarding the composability, there should be a virtualization technique to make a virtual execution platform per each application. Although the spatial virtualization is reachable by using a Memory Mapped Unit (MMU), the temporal virtualization should be guaranteed at the clock cycle level of the hardware of the processor, memories, interconnect, and all their arbiters. Additionally, resource management should be composable too, e.g. it's handling of starting & stopping of applications.

As TUE's planned contribution in Comp4Drones project, an improved version of the CompSOC[74] platform implemented on FPGA will be utilized in the reference architecture of the drone systems. This improved version uses a commercial implementation of the basic CompSOC hardware platform by Verintec, together with a predictable ROS stack made by the TUE.

[74]Goossens, et al. "NoC-Based Multiprocessor Architecture for Mixed-Time-Criticality Applications." Springer (2017): 491-530.

Keywords: System on Chip (SOC), FPGA, Embedded Real-time hardware, Predictability, Composability

8.4.2 State of the Art

The State of the Art version of the CompSOC platform, implemented on FPGA, can be seen in Figure 66. In this version, we have implemented a Multi-Processor System on Chip (MPSoC) with three independent MicroBlaze based tiles. However, considering the modularity of the design, the platform is easily salable to more tiles.

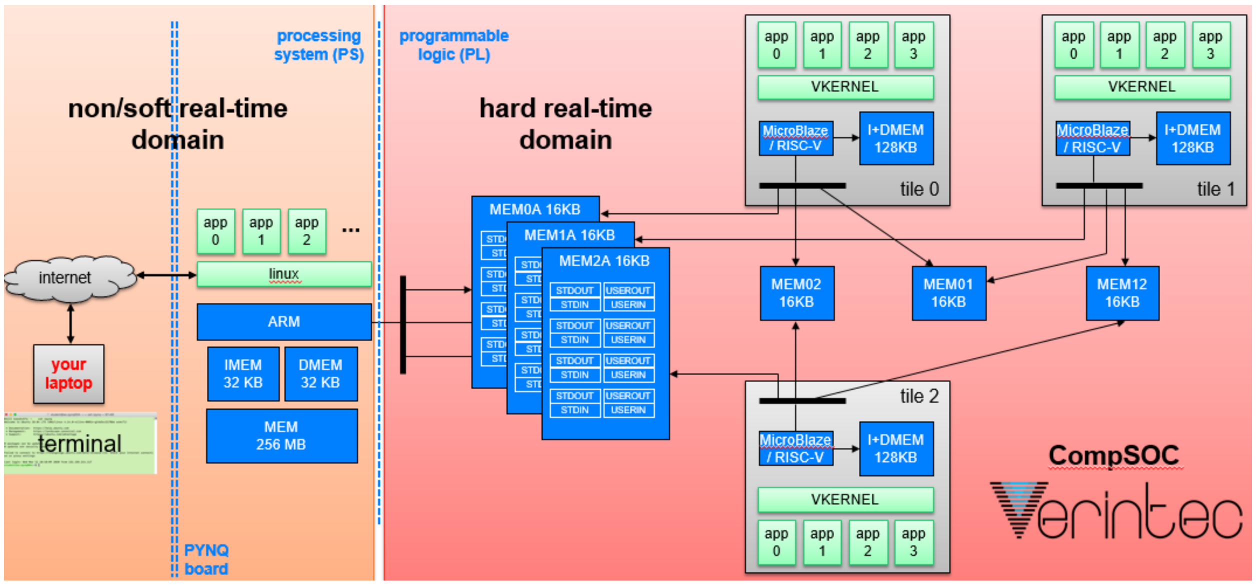


Figure 66: The overall view of CompSOC platform

Processor Tiles: As the computation core, a 32-bit MicroBlaze processor at the frequency of 100 MHz is embedded inside each tile. Virtualizing the tile to the applications is done by a virtualization micro kernel (VKERNEL) that multiplexes multiple tasks on the same hardware using preemptive scheduling mechanism. From the scheduling perspective, the processor period is divided into fixed slots and each VEP is assigned to a specific slot with predefined size. It is worth mentioning that the context switch time between slots is fixed and the first slot of each period is occupied by the system application (SysVEP). Regardless of timing isolation, an Instruction - Data Memory (I+DMEM) is spatially divided by VEKRNEL between different VEPs of the tile. In this version we have assigned 32 KB to each VEP, but there is no limitation to customize this configuration. Finally, there is a MMU component which is responsible to convert virtual memory address passed by each VEP to the physical address in I+DMEM. It should be mentioned that the proposed architecture is a globally asynchronous multiprocessor platform.

Inter/Intra Tile Communication: In the initial version of CompSOC, the inter-tile communication was implemented by a hard real-time (Æthereal) NOC on which each VEP has its own virtual port. However, considering the simplicity and maintainability of the platform, we have decided to use shared memory for this purpose. As it can be seen in Fig. 1, there is a real-time double channel shared memory with the capacity of 16KB between each two tiles of the platform. It should be mentioned that the write/read instruction on both local and shared memories are hard real-time. Since the mentioned memory is accessible by all the VEPs of each tile, it is also used for intra-tile communication.

Off-Chip Communication: Due to the need for system troubleshooting as well as communication with the outside world, there should be a channel with the embedded system platform. For this purpose, we have defined a shared memory between each tile of the embedded platform and the ARM processor in the Processing System (PS) side. The shared memory of each tile (specified with MEMxA) is also a double channel 16KB real-time memory implemented in the Programmable Logic (FPGA) part.

8.4.3 Improvements

The overall hardware architecture improvements planned in this project can be summarized by the following bullet points:

- (Requirements for) improved CompSOC hardware to support predictable &composable ROS2 software stack, including, for example:
- Migration from MicroBlaze to RISC-V processor due to more complex instruction set and real usecases of RISC-V in industry compared to limited domain for MicroBlaze.
- Improving memory controller in terms of predictability and performance by integrating reliable and secure communication protocols such as c-heap.
- Integrating I/O ports like UART to the platform to be utilized by sensor devices in Robotic domain.
- Reliable and transparent communication channel with the world outside based on an agent-client implementation of Data Distribution Service (DDS) standard. This will help drone to drone communication in the context of Comp4Drones project.
- Improving monitoring tools such as channel monitoring for system debugging and troubleshooting.

8.4.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Predictable and composable application development on FPGA SoCs	Jitter-free execution of application code; Interference-free execution of multiple application codes;	SC1.1, SC4.2	O1, O4
Improving the verification, validation and certification process	Reducing the verification and certification time and cost	SC4.1, SC4.2	O4

8.5 Highly Embedded Customizable Platform for SLAM technique - Modis

8.5.1 Component Description

The system will be equipped with the following sensing capabilities: magnetometers, compasses, GPS (optional), a gyroscope and accelerometers. Additionally, the board will be provided with a serial communication bus (or more than one), e.g. USART, SPI, MavLink and a 32-bit MCU.

Keywords: Sensors, SLAM

8.5.2 SoA comparison

SLAM (Simultaneous Localization and Mapping) techniques are considered to be a mature field, and nowadays there are a many open-source systems that are able to deliver fast and accurate estimation in typical real-world scenarios. The actual problem is the lack of a modular architecture (involving hardware and firmware configurations) that can deal with heterogeneous sensors' configurations and that can provide an easy-to-use and easy-to-adapt, extensible and reusable base configuration package.

First of all, the best way to go is the multi-modal SLAM, which involves the use of several heterogeneous sensors that can complement each-other and enhance the precision of the computed cues. But, as the goal of this task is to provide a modular and easy-to-adapt architecture, the SLAM system has to be capable of providing a sufficient degree of precision when some of the predefined set of sensors is not available and it has to rely on a constrained array of detecting devices.

Multi-modal SLAM is already a well-known concept[75], based on the use of a main sensor and a supplementary one. The first one is used to provide constant detection, while the supplementary one is generally used to provide specific cues, such as the scale or inertial measurements. In this project, the SLAM technique will not be based on visual recognition or camera sensors, but on a combination of radio, ultrasonic and magnetic signals, this in order to be a lightweight and low-cost solution to deploy on different types of UAV and UGV.

Recently, a modular, flexible and fully extensible SLAM architecture, named MOLA1 is proposed[76]. This system is specifically aimed to support visual sensors like LIDAR, monocular and stereo cameras, visual odometry and others. The concept idea is to use several different independent modules, each of which covers a specific role.

This idea can be applied to a non-visual SLAM technique, developing a modular architecture, with sub-modules for magnetometers, compasses, GPS, gyroscope and accelerometers, paired with a specific data-fusion algorithm and a data-exchange system that can provide raw data to the computing unit. The data-fusion algorithm will also provide a modular architecture, that will provide the adaptation layer to different sensor-sets' configurations.

The hardware base of this work will be a 32-bit MCU with ARM Cortex architecture, with an embedded software layer capable of real-time execution (e.g. RTOS and Embedded Linux).

The system will make a full use of the USART protocol to ensure communication with the sensors' array, while a supervisor software module will act as a bare-metal watchdog, managing in-system communication and the communication with the UAV/UGV control unit through MAVLink bus.

8.5.3 Improvements and Metrics

Improved modularity with respect to ad hoc drone HW supporting only mission specific tasks.

KPI	Metrics	Success Criteria	Project Objective
Easy integration of additional components	Integration effort	SC1.1	O1

8.6 Efficient Digital Implementation of Controller on FPGAs - UNIVAQ

8.6.1 Component Description

Description: Component name: Efficient digital implementation of controllers, designed on the basis of a discrete–time models, on FPGAs.

In order to operate safely and autonomously, the controller shall enable new functionalities, such as collision avoidance, navigation, situation awareness, etc., while ensuring the basic control actions. To ensure shorter execution times of the control algorithm, FPGAs can be used. To further reduce the execution time in FPGAs, some techniques (retiming/pipelining, folding/unfolding, interleaving, etc.) can be used that allow transforming the circuit structure, in order to reduce this time, and possibly the power consumption, maintaining the desired functionalities.

Keywords: Supporting Tool, SoC, FPGA

8.6.2 State of the Art

Control law implementation on FPGAs is a relatively unexplored field due to various factors: A specialized expertise is needed, new design methodologies have still to be developed, the nature of most control algorithms themselves[77]. There are several works suggesting the use of FPGAs in various

[75]Colosi M., Aloise I., Guadagnino T., Schlegel D., Della Corte B., Arras K. O., Crisetti G. Plug-and-Play SLAM: A Unified SLAM Architecture for Modularity and Ease of Use. https://arxiv.org/abs/2003.00754
[76]Blanco-Claraco J. L. A Modular Optimization Framework for Localization and Mapping. Proc. of Robotics: Science and Systems, 2019
[77]E. Monmasson and M. N. Cirstea, FPGA Design Methodology for Industrial Control Systems—A Review, in IEEE Transactions on Industrial Electronics, vol. 54, no. 4, pp. 1824-1842, Aug. 2007, doi: 10.1109/TIE.2007.898281

fields, however there are no standard recognized methodologies known to the contributors, which in turn are experienced designers and concentrate their efforts on this subject to provide their expertise in both digital control and electronics[78]. Using FPGAs as "accelerators" such as co-processors for heavy computing tasks is a widespread practice in image processing or neural networks fields, as an example[79]. This is partly what is being done on the proposed controller. However, one is not limited to the most common practices, since there are a lot of good ones to be taken in account, such as hybrid System-on-Chip architectures that are perfect candidates in other contexts, such as for autonomous vehicle applications, due to their unique combination of peculiarities such as low power consumption and weight, high performances, great versatility[80]. Combining state of the art software and hardware design, advanced controllers are expected to be feasible.

8.6.3 Improvements

The expected improvements are the increased performance of the controller, with reduced response time, and optimal balance between performance, power consumption and weight, and increased resiliency of the drone during operations.

8.6.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Improvements in performance and resource consumption	Control algorithm latency reduction, and controller throughput improvement	SC4.1	O4

9 Tools

To support the development of drone systems, a number of tools need to be developed. In this section, we describe the tools that support the architecture implementation.

9.1 Mixed-Criticality Design Space Exploration - UNIVAQ

9.1.1 Component Description

Description: Component Name: Mixed-Criticality design space exploration for HW/SW partitioning, mapping and architectural definition.

Objectives: The main goal is the integration of an existing design space exploration approach that takes into consideration mixed-criticality system constraints, introducing and considering the hypervisor scenario, into the C4D framework. UNIVAQ will exploit a specific software component able to (semi)automatically generate application partitioning, mapping and architectural definition by means of search methods and simulations in the design space exploration activities.

Keywords: Supporting Tool, Design Space Exploration, Mixed-Criticality, Simulation, Timing Performance, Hierarchical Scheduling

[78]C. Acosta-Lúa, S. Di Gennaro, A. Navarrete-Guzman, S. Ortega-Cisneros and J. R. Domínguez, Digital Implementation via FPGA of Controllers for Active Control of Ground Vehicles, in IEEE Transactions on Industrial Informatics, vol. 15, no. 4, pp. 2253-2264, April 2019, doi: 10.1109/TII.2019.2890839
[79]S. Che, J. Li, J. W. Sheaffer, K. Skadron and J. Lach, Accelerating Compute-Intensive Applications with GPUs and FPGAs, 2008 Symposium on Application Specific Processors, Anaheim, CA, 2008, pp. 101-107, doi: 10.1109/SASP.2008.4570793
[80]Ying-Shieh Kung and Gua-Shieh Shu, Development of a FPGA-based motion control IC for robot arm, 2005 IEEE International Conference on Industrial Technology, Hong Kong, 2005, pp. 1397-1402, doi: 10.1109/ICIT.2005.1600854

9.1.2 State of the Art

A growing trend in the embedded system domain (both in industry and academy) is the switch from single-processor/core to (heterogeneous) multi-processor/cores (i.e., parallel) HW/SW platforms used to execute embedded applications with different levels of criticality (i.e., Mixed-Criticality Embedded Systems, MCESs). The main problem in the management of a MCES is to ensure that low criticality applications do not interfere with high criticality ones. This type of systems can be found in many domains such as aerospace[81] and automotive industry[82]. Critical and non-critical applications can be further divided by identifying different criticality classes. The goal is always to allow these applications to interact and coexist on the same platform, but a proper management of such Mixed Criticality (MC) systems becomes a very complex task that poses several challenges also from the implementation point of view[83]. At the system level of abstraction, there are very few works that introduce mixed-criticality issues directly into a HW/SW co-design flow, and a proper methodology that tries to fully consider all these kinds of requirements doesn't exist up to now. All these aspects will be analyzed inside the C4D project.

9.1.3 Improvements

Improvements inside the C4D project will be related to a SystemC timing simulator that checks F/NF requirements fulfillment, and to the hierarchical scheduling that allows to simulate the use of hypervisor technologies. The main goal is to reduce simulation time and design space exploration execution time, while keeping bounded the accuracy and the errors associated with the estimation values. The work is directly linked to the WP6 tool called HEPSYCODE.

9.1.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Easy to design and deploy	Simulation execution time and hierarchical scheduling overheads	SC4.1	O4
Improvements in performance and resource consumption	Application simulated time and speedup achieved	SC4.1	O4

9.2 Reference Architecture Modelling - CEA

9.2.1 Component Description

Description: The current embedded architectures of drones are organized in loosely coupled monolithic boards. Each board composed of processor, memory and communication resources (e.g. flight control, planning boards). This separation ensures that the subsystems operate almost independently from each other and avoids interference coming from the other parts. However, this approach does not support the continuous development of drone applications such as the ever-increasing demand on autonomy. The COMP4DRONES project will face this challenge by developing a compositional and integrated drone embedded reference architecture following the IMA principles, adapted to, and still considering, the drone resource constraints.

Keywords: Supporting Tools

9.2.2 State of the Art

Current open drone embedded platforms (such as ArduPilot, Paparazzi or Dronecode) embrace communities of hundreds of stakeholders, which need to be preserved and strengthened. However,

[81]R. I. Davis A. Burns. Mixed criticality systems - A review. In Research report, University of York
[82]P. J. Prisaznuk. Integrated modular avionics. In Proceedings of the IEEE 1992 National Aerospace and Electronics Conference, NAECON 1992, pages 39–45 vol.1, May 1992
[83]S. Baruah, H. Li, and L. Stougie. Towards the design of certifiable mixed-criticality systems. In 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium, pages 13–22, April 2010

these communities are still rather fragmented, by representing specific technologies. To strengthen these platforms, to enable interconnections between them, and definitively also between new ones (e.g. AI-based technologies), the COMP4DRONES project envisions a model-driven approach built on top of, or rather “around, the current code-centric platforms. The main principle behind this approach will be to agreeing on common modeling structures that share these platforms, formalizing the specification of the different platform patterns by using the proposed modeling structures, and integrating these modeling structures in the COMP4DRONES tools ecosystem. The modeling structures will be realized by an agreed meta-model. We are aware that agreeing on common modeling structures is a long process, but the project will build on existing modeling languages, such as RobMoSys (e.g., to enable composition and separation of concerns in the robotics domain) and MARTE/AADL (e.g. for timing and performance concerns).

Papyrus for Robotics is an open-source, Eclipse-based modeling tool for robotic system. It is a customization of the Papyrus UML modeling tool. Papyrus for Robotics conforms to the RobMoSys modeling approach. This implies that the tool supports different views for different roles/stakeholders, for instance a service designer, a component developer and a safety as well as system architect.

A user of Papyrus for Robotics can specify a robotic (or drone) system as an assembly of component instances. The components in turn provide or require services via ports. These services are ideally not project specific but standardized by domain experts in order to facilitate the reuse of component definitions. In case of Papyrus for Robotics, the service definitions are part of a set of model libraries that have been obtained from ROS message and service definitions via reverse engineering. This means that a Papyrus for Robotics user has access to a wide range of service definitions that have been established by the community of several years, for instance geometry-related messages or messages needed for navigation. A component instance can be configured, i.e. the configuration parameters of the component definition can be specified for a particular instance.

9.2.3 Improvements and Metrics

With the support of WP6 tools, CEA will enable:

- The modeling of the compositional and integrated drone embedded reference architecture.

9.2.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Simplifying integration of the drone system components	Effort for customization of drones	SC1.2	O1

9.3 Paparazzi Autopilot - ENAC

9.3.1 Component Description

Description: Paparazzi is a complete open source system for UAS including both airborne autopilot as well as complete ground station, mission planning and monitoring. The ground segment uses a software bus (the Ivy bus) to exchange information, making it easy to add custom software or replace software.

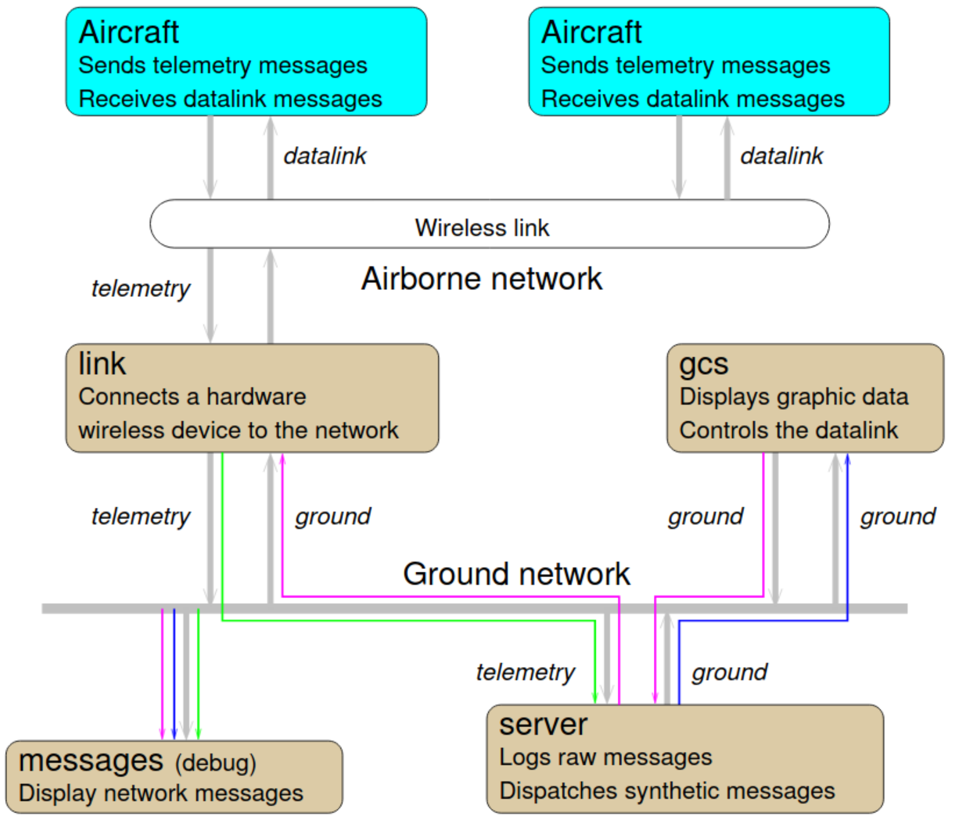


Figure 67: Paparazzi system architecture

The airborne segment is made for multiple vehicles, like fixed wings, rotorcrafts, but also rovers. It is highly configurable and one can choose between multiple alternatives for control algorithms, navigation algorithms, sensors, target platforms. Custom modules can be added to extend the capabilities of the system. The airborne code is generated by the Paparazzi build system according to the chosen configuration. This architecture allows generating efficient platform-specific code from a platform-agnostic configuration.

For more details, please visit http://wiki.paparazziuav.org/wiki/Overview

Keywords: Autopilot

9.3.2 State of the Art

Many autopilots systems exist, some of them being open source. We can cite as examples Ardupilot, PX4, LibrePilot.

9.3.3 Improvements

The objective for this component is to improve the validation of code modifications, in particular by applying the outcome of the tools and methods from WP6.

The internal validation will focus on improving the code coverage with automated compilation and execution of static analyses. The current coverage is about 20 to 30 % (specific compilation) and we expect to reach at least 60 to 70 % of coverage to match expectation of SC4.1.

A specific effort will be made to deploy a simulation framework that can be used from other partners to simulate multi-UAV operations with a realistic flight model in order to validate external packages.

Potentially hardware in the loop simulations is also possible. The objective is to reduce by two the time required to prepare a test case simulation for validation to match expectation of SC1.1.

9.3.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Improving the code coverage	Code coverage percentage	SC4.1	O4
Reduce the time required to prepare a test case simulation	Time for test case simulation	SC4.1	O4

9.4 Integrated Systems Simulation - Siemens

9.4.1 Component Description

Description: The goal of this work package is to provide reference architecture to ease the integration and customization of embedded drone systems. SimcenterAmesim, a system simulation software tool developed by Siemens, can help achieve this objective thanks to its systems integration capabilities.

Siemens' SimcenterAmesim is a software tool dedicated to modeling and simulation of dynamic and multi-physics systems. In the tool's environment, systems are modeled connecting components available in the libraries, which cover several physical (fluids, mechanical, electrical…) and application (aerospace, automotive, gas turbines…) domains. The libraries components' interface is realized through ports that allow the flow of physical variables. The definition of the interfaces is based on the bond graph theory, which allows a common representation of dynamic systems regardless their physical domain. This approach ensures consistency in the conservation of energy, conservation of mass and units of measurements when connecting different components. SimcenterAmesim is by nature an integration platform, with dedicated interfaces to other software tools. Siemens is part of the consortium defining and developing the Functional Mock-Up Interface (FMI) standard. This is a protocol that defines a container and an interface to exchange dynamic models using a combination of XML files, binaries and C code zipped into a single file. FMI can be considered as an enabler for scalable and tool neutral integration of simulation models from different technical disciplines, developed by different actors. Amesim openness is an asset for this research project, and it will be used to couple drone's system simulation model to other tools modelling complementary aspects of drones' design, analysis and optimization. For example, it can be coupled with SimcenterPrescan, another software tool of Simcenter portfolio, which provides sensors and environment modeling capabilities for autonomous vehicle simulation.

The contributions the tool can bring to this work package are listed here below:

f) Physical behavior modeling of drones' platforms (electric motors, propellers, batteries, flight dynamic…) for dynamic performance simulations. These components can be modeled and validated separately.
g) Integration of the aforementioned models to build an integrated model of a drone platform for performance simulations. Such a model can predict the performance of the drone before it is manufactured. Thanks to the innate modular approach of SimcenterAmesim, it is possible to model and evaluate several variants of the drone platform (e.g. quadcopter vs. octocopter, impact of larger propellers on manoeuvrability vs. endurance…) and make design trade-offs. This largely supports the simplification of integration and customization of drones' platform as sought by this work package.
h) The physical plant model created with SimcenterAmesim can be used to calibrate and validate controllers and autopilots with Software-in-the-loop or Hardware-in-the-loop thanks to the tool real-time capabilities and its interfaces available for this kind of analysis. Using these standard

interfaces (FMU) it is then possible to easily change the controllers to be tested, supporting again the modularity approach invoked by this work package.

i) By means of dedicated interfaces with third tools, it is possible to couple the model obtained (plant and controllers) with software which provides sensors and environment modeling capabilities for environmental awareness simulations. This would give the ability to validate D&A algorithms.

j) The model consisting of the plant and the controllers (autopilots) can then be used to evaluate the performance of the drone and the controllers' robustness in case of failure events

Keywords: Supporting tool, System Simulation, Performance Analysis

9.4.2 State of the Art

There is no evidence in public scientific literature of the systematic use of system simulation to support and improve the design of drones, in particular regarding the ease of integration and customization.

9.4.3 Improvements and Metrics

- Development of eventual missing interfaces to run Software-in-the-loop or Hardware-in-the-loop analysis with components provided by partners
- Improve (with dedicated developments or through methodologies) the integration with other tools for environment simulation, sensors simulation and flight simulators. The goal is to provide a comprehensive simulation framework to address autonomous drones' simulation.

9.4.4 Component Metrics

KPI	Metrics	Success Criteria	Project Objective
Tools integration	Successful integration with at least another software tool useful to partners	SC4.1	O4
Modeling and simulation accuracy	Introduce in the tool developments dedicated to improving drones modelling and simulation results	SC4.1	O4