Definition of industrial use-cases

Prepared by Paolo Giuseppetti (Cobra)

Issued by COBRA

Document Number/Rev. CONTREX/COBRA/R/D1.2.1/1.5

Classification CONTREX Public

Submission Date 2013-09-30

Due Date 2013-03-31

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)
## History of Changes

<table>
<thead>
<tr>
<th>ED.</th>
<th>RE/ V.</th>
<th>DATE</th>
<th>PAGES</th>
<th>REASON FOR CHANGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR</td>
<td>0.1</td>
<td>2013-11-04</td>
<td>26</td>
<td>OFFIS use case 1 contribution</td>
</tr>
<tr>
<td>CB</td>
<td>0.2</td>
<td>2014-03-13</td>
<td>31</td>
<td>Initial description of use case 2</td>
</tr>
<tr>
<td>PG</td>
<td>0.3</td>
<td>2014-03-20</td>
<td>39</td>
<td>Cobra contribution to use case 2</td>
</tr>
<tr>
<td>JF</td>
<td>0.4</td>
<td>2014-03-25</td>
<td>50</td>
<td>Intecs contribution to use case 3</td>
</tr>
<tr>
<td>SB</td>
<td>0.5</td>
<td>2014-03-28</td>
<td>50</td>
<td>Comments</td>
</tr>
<tr>
<td>CL+SR</td>
<td>0.6</td>
<td>2014-03-30</td>
<td>65</td>
<td>GMV contribution + use-case 1 update</td>
</tr>
<tr>
<td>SB+CB</td>
<td>0.7</td>
<td>2014-03-31</td>
<td>65</td>
<td>Comments</td>
</tr>
<tr>
<td>PG+CB</td>
<td>1.1</td>
<td>2014-04-10</td>
<td>73</td>
<td>Use case 2 update</td>
</tr>
<tr>
<td>PG</td>
<td>1.2</td>
<td>2014-04-22</td>
<td>73</td>
<td>Final version</td>
</tr>
<tr>
<td>SSch</td>
<td>1.3</td>
<td>2014-09-03</td>
<td>74</td>
<td>Use case 1 multicopter part refinement with respect to reviewer comments</td>
</tr>
<tr>
<td>CL</td>
<td>1.4</td>
<td>2014-09-17</td>
<td>76</td>
<td>Use case 1 Remotely Piloted Aircraft part refinement with respect to reviewer comments</td>
</tr>
<tr>
<td>FR</td>
<td>1.5</td>
<td>2014-09-17</td>
<td>84</td>
<td>Use case 3 refinement with respect to reviewer comments</td>
</tr>
</tbody>
</table>
Contents

1 Introduction.................................................................................................................. 5
2 Use-Case 1: Unmanned Aerial Vehicle (UAV)............................................................ 7
   2.1 Use-Case 1a: Overall multi-rotor system .......................................................... 7
      2.1.1 Introduction...................................................................................................... 7
      2.1.2 State of the art ............................................................................................... 8
      2.1.3 Fundamental Work ......................................................................................... 9
      2.1.4 Requirements on multi-rotor systems ..........................................................17
      2.1.5 Discussion on extensibility............................................................................ 20
      2.1.6 Proposed multi-core MPSoC based avionics ................................................ 22
      2.1.7 Requirements regarding non-functional properties ........................................ 27
      2.1.8 Power consumption estimation of demonstrator ........................................... 28
      2.1.9 Conclusion and future work ........................................................................... 29
2.2 Use-Case 1b: Remotely Piloted Aircraft – FCC’s I/O module ................................ 30
      2.2.1 General ATLANTE program overview ......................................................... 30
      2.2.2 ATLANTE Flight Control Computer (FCC) ................................................... 30
      2.2.3 Demonstrator based on the ATLANTE FCC I/O module .................. 34
3 Use-Case 2: Automotive Telematics ............................................................................ 41
   3.1 State of the art ....................................................................................................... 41
   3.2 Scenarios ................................................................................................................ 42
      3.2.1 Scenario 1: Device installation ..................................................................... 43
      3.2.2 Scenario 2: Crash management .................................................................... 47
      3.2.3 Scenario 3: Key-off services ........................................................................ 51
      3.2.4 Scenario 4: B2B services ............................................................................. 52
   3.3 Use-case at a glance ............................................................................................... 60
   3.4 Requirements ........................................................................................................ 60
   3.5 Mixed criticality issues ........................................................................................... 61
4 Use-Case 3: Ethernet Over Radio System ................................................................. 63
   4.1 Telecom Use Case: Overview ............................................................................... 63
      4.1.1 Extra-functional and Green computing challenges ......................................... 63
      4.1.2 International standards .................................................................................. 64
      4.1.3 Mixed-criticality issues .................................................................................. 64
   4.2 State of the Art ....................................................................................................... 65
      4.2.1 Ethernet over Radio System main features .................................................. 67
      4.2.2 Hardware architecture ................................................................................... 67
      4.2.3 Software architecture ..................................................................................... 70
      4.2.4 Ethernet over Radio System: Thermal and Power Analysis ......................... 71
   4.3 Intecs CONTREX expected goals and Definition of New possible Scenarios ...... 73
      4.3.1 Intecs CONTREX expected goals ................................................................. 73
      4.3.2 Analysis of possible new Scenarios and Test Bench ....................................... 74
   4.4 Requirements on Ethernet over Radio System .................................................... 77
      4.4.1 Requirements for Ethernet Over Radio System Software porting ................ 77
      4.4.2 Requirements for the introduction of a System Modeling Layer .................... 78
      4.4.3 Requirements for a power analysis environment ............................................ 79
5 References .................................................................................................................. 81
   A. STM32F103REY6 ................................................................................................. 83
Definition of industrial use-cases
1 Introduction

Scope of this deliverable is to describe the three use-cases in detail and define the specification and constraints for the demonstrators that will be developed within the project.

The document is structured in three sections, namely:

Section 2 – Avionics: In the avionic domain two demonstrators will be developed. The first one is a demonstrable flying system has a single overall UAV controller executing several tasks of heterogeneous criticality levels. The demonstrator bases on a pre-existing multi-rotor system that is used as an aerial platform which will be extended by a Multi-Processor System on Chip (MPSoC) to implement the safety-, mission- and non-critical functions of an autonomously civilian UAV.

The second demonstrator is based on the Flight Control Computer (FCC) developed by GMV in charge of the guidance, navigation and control of the Remotely Piloted Aircraft (RPA). Safety concerns are strongly associated to this type of systems as the failure of a given component may have catastrophic consequences, as for instance making the RPA fall on populated areas or crash with another aircraft. Even when a failure does not incur direct human losses (as in the case of flying in segregated airspace), the loss of the vehicle supposes a great economic loss.

Section 3 – Automotive telematics: This demonstrator provides private and/or fleet vehicle drivers with a support service in case of accident. The architecture is based on three main components: a sensing unit for acceleration measurements, a localization unit for GPS reading and a data processing and communication system for identification of accidents and communication of position data either to public authorities (hospital, police) or private support providers. Cobra Telematics is extending this scenario by moving part of the processing from the in-vehicle ECU to a remote data center. The acceleration sensing unit and the GPS localization unit are still installed on the customer vehicle but perform only limited processing while extensive processing is in charge of the data center. This already allows providing improved services to end users. The overall architecture will be constituted by a cloud of semantic services within a framework, sharing a semantic data model. A single node will consist of a multi-core HW platform that integrates one or more sensor and performs pre-processing and elaboration of sensor data.

Section 4 – Telecommunications: The telecom demonstrator is based on the Ethernet Over Radio System, it is specifically designed and engineered for such situations where E1 signals transportation is required. It allows a smooth transition with the past generation of transport (PDH) networks, encapsulating the E1 signal into an Ethernet frame. It is, furthermore, particularly suited to cover mobile broadband infrastructure data growth from GSM to WDCMA to LTE and many other needs of high data transport. Thus it is a naturally mixed critical scenario. Guarantee of timing requirements under optimization of power consumption and temperature maps of the hosting equipment, as well as installation weight and space footprint, is essential. The new CONTREX techniques for global optimization over the entire installation will greatly enhance cost/performance characteristics. More information can be found in Annex B3: Use-case 3: Ethernet Flexible Access System.

Each section is organized into:
• An overview of the use-case and its context
• A discussion of the state of the art
• A detailed description of the goals
• A description of the technologies that will be used for implementation
• A set of functional requirements and the non-functional constraints
• Some concluding remarks concerning mixed-criticality issues
2 Use-Case 1: Unmanned Aerial Vehicle (UAV)

This use-case 1 is driven by GMV and OFFIS. Mixed-criticality is considered at different levels of granularity in this use case. Firstly, the developed demonstrable flying system has an overall UAV controller that executes several tasks of heterogeneous criticality levels. This part bases on a pre-existing multi-rotor system that is used as an aerial platform which will be extended by a Multi-Processor System on Chip (MPSoC) to implement the safety-, mission- and non-critical functions of an autonomously civilian UAV. Secondly, a subset of the existing software of the Flight Control Computer of the Remotely Piloted Aircraft (RPA) ATLANTE system will be adapted into a demonstrator and deployed on a Multi-Processor System on Chip (MPSoC) using tasks with different criticality levels. This part, driven by GMV, has a special focus on exploitation of methodologies and tools. In the following sections both parts will be explained in detail.

2.1 Use-Case 1a: Overall multi-rotor system

2.1.1 Introduction

Unmanned Aerial Vehicles (UAVs) are omnipresent nowadays but nearly every time UAVs are associated with a military use. Since a few years small and affordable systems are on the market which are also eligible for personal and sovereign use cases, e.g. flying infrared cameras for firefighters or aerial photography for archeological excavations.

![Figure 2.1 Multi-rotor system (Quadcopter) used in [1]](image)

This part of the deliverable focuses especially on multi-rotor systems with four or more symmetric mounted rotors for civilian use cases, like shown in Figure 2.1 Multi-rotor system (Quadcopter) used in [1]. Octocopters (with eight rotors) are able to carry up to 5.5 kilograms of payload [2]. Such a heavy payload is needed by several use cases that require infrared cameras, reflex cameras or other heavy equipment. The flight time of the systems depends on the weight of the payload and environmental conditions and varies mostly between 15 and 25 minutes.
In the early days of multi-rotor system development the key aspect was on the stability and the navigation control to enable safe flight maneuvers and autonomous acting UAVs. Since many open source and commercial projects now provide stable control algorithms and navigation methods the focus changes step by step to invent new operational scenarios with increasing payload functionality. Thereby, quite often complex on-board processing is required, e.g. in terms of video processing or processing of big sensor data. As a result of the limited interfaces and performance of the microcontrollers used for the safety and real-time critical flight and navigation algorithms, additional microcontrollers are needed for the non-safety-critical payload processing. Unfortunately this segregation of mission- and safety-critical parts results in a higher communication effort between the microcontrollers and a higher power consumption of the overall system resulting in a decreased flight time.

With the invention of Multi-Processor System on Chips (MPSoCs), smaller high-performance chips have been introduced which are able to process many more tasks than the long-established microcontrollers used in most of present multi-rotor avionics systems. Most recent MPSoCs, such as in the Xilinx ZYNQ MPSoC family, are easily capable to compute all tasks of a UAV system on a single die resulting in a power efficient and cost-effective solution. Nevertheless the sharing of resources introduces dependencies between tasks of different criticalities and problems regarding a compositional certification of applications’ correctness, run-time properties and reliability.

This use-case proposes a mixed-critical implementation of a UAV’s avionics using the Xilinx ZYNQ platform and presents the resulting challenges of this integrated solution in terms of predictability along extra-functional properties such as real-time behavior, power consumption, and temperature. Further it gives an outlook to future work that is necessary to develop a generic predictable computing platform and mechanisms for segregation between applications of different criticalities sharing computing resources.

The remaining part of this work is structured as follows: Section 2.1.2 gives an overview on the state of the art of available UAV systems followed by a detailed description of a present system implementation in Section 2.1.3. Section 2.1.4 defines requirements on a multi-rotor system and Section 2.1.5 outlines limitations of existing solutions e.g. in terms of extensibility. Section 2.1.6 proposes our new multi-core SoC based avionic that overcomes the limitations following the idea of integrating all control functions of different criticalities to one high performance MPSoC. This section further describes new challenges and how they can be targeted. The use-case description is concluded and an outlook on future work is given in Section 2.1.9.

2.1.2 State of the art

In literature many approaches for avionics hardware architectures for multi-rotor systems can be found. One of these avionics is presented in [3]. It uses an approach for its architecture which is also used in many other projects. The processing hardware consists of two 32Bit Infineon TriCore® TC1796 microcontrollers running at 150MHz and follows a strict separation of concerns. The first controller operates the sensor and remote control processing and processes the flight controller tasks. The second controller is responsible for the navigation controller tasks and the communication with the ground control station. At this avionics a very simple payload processing is possible such as servo control or switching outputs. This task sharing can also be found in other projects like mentioned before.
The partly commercial MikroKopter platform from HiSystems GmbH [4] also uses two microcontrollers. An 8Bit Atmel ATmega1284P for the sensor and remote control processing as well as the flight algorithms and a 32Bit ARM9 chip from STMicroelectronics of type STR911FAM44X6 for navigation controller and communication tasks. The performance of these controllers is many times smaller than the Infineon TriCores. As a consequence many of the used algorithms need to be optimized by using integers instead of floating point numbers, e.g. the sensor processing as well as the stability and height controllers. In addition, high-performance payload processing is not possible. A detailed description of this platform can be found in section 2.1.3.

The open source project Universal Aerial Video Platform - Next Generation (UAVP-NG) also relies on the two controller approach for its avionics [5]. The hardware revision HW-0.24-mini-r1 uses an ARM7 controller of type NXP LPC2148 and an Atmel ATmega644P whereas the task separation is different to the other avionics. The ARM7 controller does sensor processing and also computes the flight and navigation controller tasks while the ATmega is responsible for the remote control processing and the communication tasks. In the newest developed prototype HW-0.30-r0 two STM32F407IGT6 ARM M4 controllers from STMicroelectronics are used whereas the main controller operates the overall sensor and flight control and the second controller is exclusively reserved for user tasks like payload processing.

Ascending Technologies GmbH developed a full commercial avionics architecture with the same approach of task separation [6]. Again two microcontrollers (NXP LPC214x ARM7) are assembled on the avionics. The first one is again responsible for sensor processing and flight algorithms and the second one operates the navigation and communication tasks. In addition, the platform has the option to carry one of two available on-board PCs for high-performance on-board processing. The first one contains an Intel Atom processor, the second one an Intel i7 core. With these extensions it is possible to realize high-performance payload processing, e.g. high-resolution video processing with the drawback of a high energy consumption and increased weight.

Beside the presented platforms and architectures many other approaches exists but all recent developments underline that payload processing and processing of special user tasks are getting more and more into the focus of the developers. The approach in this use-case also follows this demand but with the difference to use a single MPSoC for critical and non-critical tasks.

2.1.3 Fundamental Work

In the past huge effort has been spent in the development of multi-rotor systems at OFFIS and University of Oldenburg. We have selected three of our previously designed avionics architectures for deriving requirements for the mixed-critical MPSoC implementation. This section will present the three selected avionics, including the used avionics architectures. The presentation is followed by general requirements of our multi-rotor systems which can be deduced from the previous work and a discussion about the extensibility of the actually used avionics for high-performance payload processing.

2.1.3.1 Model-driven aviation software design

In [1] a new development approach for the aviation software of multi-rotor system is described. The development bases on a model-driven development approach using Esterel
Technologies SCADE Suite® [7] that is specialized for the development of safety-critical software for embedded systems. It uses a graphical design language based on Lustre. Models created in SCADE are compiled to C or Ada by a verified compiler. The created model includes five main software parts which are necessary for a flyable system (see simplified Figure 2.2). The functions of the individual parts are:

- The set point processing is responsible for the computation of the control signals coming from the remote control. The main challenges of this part are to limit and to scale the values to defined and useful ranges. Another job is the decoding of remote control events like positions of the switches or motors on and off signals. Outputs are the set points and events which are used in the subsequent blocks.

- The sensor processing contains algorithms to filter and to fuse the sensor values of the gyroscopes, accelerometers, magnetometers, and the pressure sensor. Also a scaling of the filtered sensor values takes place to get the right units, e.g. degree for angles. The results are the actual values of the angles Φ (roll), Θ (pitch) and Ψ (yaw) (see Figure 2.3) and the present height mainly measured by the pressure sensor.

- To stabilize the multi-rotor system during a flight the stability controller uses three extended Proportional-Integral-Derivative (PID) controllers for all three angles Θ, Φ and Ψ. The extension of the PID controller is the usage of the angle velocities to counteract the changes of the angles from environmental influences.

- To hold the height or to perform controlled changes, the height controller also uses an extended PID controller, based on the velocity in direction of the \( z_n \) axis (index \( n \) for the world frame coordinate system NED (North, East, Down), which is used by navigation tasks). There are several other small extension which can directly be found in [1], e.g. fixing the integral part by changing the set point to prohibit overshooting.
The last part consists of calculating the control values. It uses the results of the stability and height controller to calculate the final set points which will be transmitted to the motor drivers of the multi-rotor system.

With these five controller parts the multi-rotor system can be operated manually by remote control with the option to use the height controller as assistance system. A more detailed description of the algorithms can be found in [1].

The designed model was exported to C code by the SCADE Suite® compiler and has then been embedded in the handwritten configuration code of the microcontroller. The used avionics is a self-designed single controller board with a 32Bit ARM9 STR911FAM44X6 controller from STMicroelectronics operating at 96MHz. The developed architecture is shown in Figure 2.4. In the center of the figure the ARM9 controller is connected to the peripheral devices, e.g. sensors, remote control receiver, motor drivers and a radio module. In the controller itself the compiled SCADE model which receives its necessary data from the interrupt service routines of the used interfaces and the internal analog digital converter is shown. The SCADE model returns the calculated control values for each motor driver and debug data which is communicated via a radio module to a ground control station. The interface for a GPS module was arranged but not implemented.
Figure 2.4 also shows the communication rates between the used components and the loop frequency of the SCADE model. These frequencies are the basis for arranging timing that will be done later in this section. The model is executed by a periodic timer event every 2ms (500Hz) and transmits the calculated control values directly to the motor drivers. This high update rate with real-time constraints guarantees a stable flight behavior. Indeed the high frequency of execution only makes sense if the model gets also new sensor data with such a high frequency. Especially new values of the gyroscopes and the accelerometers are needed every control loop because they are highly relevant for the stability controller. Therefore, the values of these sensors are read by the analog digital converter every millisecond (1kHz) to stabilize the three degrees of freedom shown in Figure 3. The update frequencies of the values of the external measured pressure sensor (5ms periodic cycle time → 200Hz) and the external compass (8ms periodic cycle time → 125Hz) are much lower. They are long time stable during a flight and are used as references in the calculations of the sensor processing for the height and the yaw angle. As a result they are not needed that often. The frequency of the remote control receiver is fixed by manufacturer.

The SCADE model was proofed against special properties, e.g. division by zero, correct ranges of values, motor start and stop conditions. The final multi-rotor system shows straightaway a stable behavior with the designed model and reacts well on the set points given by the remote control.

2.1.3.2 Autonomous take-off and landing

In addition to the control software that is necessary for a manual operation, an autonomous take-off and landing control function has been developed in [8]. It bases on the MK Hexa XL multi-rotor system that is developed and distributed by HiSystems GmbH and whose functionality ranges from manually controlled flights by remote control to autonomous waypoint navigation and executing planned tasks like taking photos of a point of interest.
Figure 2.5 Overview of the existing avionics architecture from HiSystems GmbH annotated with interface types and communication frequencies

An overview of the architecture of this avionics is shown in Figure 2.5. It illustrates the two used controllers with their peripheral devices and the different types of connections. The Flight-Ctrl PCB contains an 8Bit Atmel ATmega1284P controller which runs at 20MHz. This controller has direct connections to the on-board sensors, the remote control receiver, and the motor drivers. For payloads like cameras, servo outputs are available to realize active camera mounts with pitch and roll correction. The main tasks of the 8Bit controller are the sensor and remote control processing, automatic control technique (stability and height controller) and actuating the motor drivers. With this PCB it is possible to get a flyable multi-rotor system. A 32Bit ARM9 STR911FAM44X6 from STMicroelectronics is placed on the Navi-Ctrl PCB and operates at 48MHz. Its main functions are navigation tasks, data logging on an SD-card and provision of an extension port for new on-board devices. For the navigation tasks the Navi-Ctrl PCB contains a magnetometer and is connected to an external LEA 6S GPS chip from uBlox. With these sensors the ARM9 controller is able to compute the actual heading with reference to the geodetic north and the present position of the multi-rotor system on earth.

Like Figure 2.4, Figure 2.5 also illustrates the communication frequencies between the components of the avionics. Especially the update rates of the sensors at the Flight-Ctrl PCB are higher than those of the avionics used in [1]. This enables better filtering of the sensor values. But the control loop, including the stability and height controller, works also with a frequency of 500Hz (2ms periodic cycle time).

To enable the multi-rotor system to manage autonomous take-off and landing maneuvers a new hardware module has been developed. It extends the existing sensor capabilities because with the available sensors of the described avionics it is not possible to determine the actual altitude above ground level of the multi-rotor system. This value has to be known in order to
guarantee a safe take-off and a smooth landing process. The developed module is stacked on top of the existent avionics and applies a high power ultrasonic rangefinder which uses 40kHz signals to get the altitude above ground level. Figure 2.6 shows the developed hardware extension which could be considered as mission-critical function when the planned mission requires safe autonomous take-off and landing maneuvers.

![Figure 2.6 Developed prototype of the high power ultrasonic rangefinder](image)

It consists of three main parts. An 8Bit Atmel ATmega168P microcontroller at 16MHz, a high power amplifier for sending ultrasonic impulses and a high sensitive amplifier for receiving the echoes of the transmitted impulses. The overall module is connected to the Navi-Ctrl PCB via the extension port (see box on the right in Figure 2.5) and uses the I²C interface for in-flight communication with the ARM9 controller. The UART interface is used to transmit telemetry data to the ground control station. The high power amplifier sends ultrasonic impulses with maximum amplitude of 40V. The receiving amplifier gains the incoming ultrasonic echo signals 1300-times magnified. With this implementation it is possible to get the altitude above ground level up to 600cm over hard surfaces and 350cm over grass with a resolution of one millimeter. The altitude above ground level is measured by the ultrasonic rangefinder with a frequency of 10Hz. The Navi-Ctrl asks for the result with 20Hz update rate to get every measurement (Nyquist–Shannon sampling theorem). The Navi-Ctrl passes the result via SPI to the Flight-Ctrl. It uses valid results in the sensor fusion to correct the altitude value for the PID height controller.

The evaluation of the height controller has shown that the multi-rotor system holds its altitude set point of 200cm with an accuracy of ±10cm at 3Bft wind speed. Without correction of the height value through the ultrasonic rangefinder the height near ground is held with an accuracy of ±30cm while the accuracy of the pressure sensor as altimeter varies strongly due to environmental influences. Another advantage of the ultrasonic rangefinder is that it is not influenced by the ground effect which builds a kind of air buffer and additionally influences the pressure sensor.

With the improved accuracy it is possible to realize autonomous take-off and landing maneuvers. For the take-off maneuver the avionics runs through the following states:

1. Check for valid values of ultrasonic rangefinder
2. Activate height and navigation controller
3. Start motors
4. Increment set point of the altitude up to defined height with constant rate of ascend

The autonomous **landing maneuver** has the following states:

1. Check if ultrasonic rangefinder is available and active
2. Choose sink rate in dependence on the actual set point
3. Decrement set point until the ultrasonic rangefinder returns valid measurements
4. Hold actual height at a defined value and validate measurements over a specified time slot
5. Choose sink rate in dependence on the actual height above ground level (lower height ! lower sink rate)
6. If the multi-rotor system hovers 5cm above ground the set point is reduced very fast to secure a safe touchdown
7. Stop motors

These methods ensure a safe take-off as well as a safe and smooth autonomous landing of the system on hard surfaces as well as on grass.

**2.1.3.3 3D navigation**

In [3] methods for the locality determination and navigation control algorithms for a multi-rotor system have been developed. This kind of 3D navigation enables the multi-rotor systems to fly defined trajectories between the planned navigation points with the accuracy of the used sensors for the localization, e.g. GPS and pressure sensors.

The used avionics was already mentioned in section 2.1.2 and can be seen in Figure 2.7. The two used controllers of type TriCore® TC1796 from Infineon are connected via a MLI (Micro Link serial bus Interface) which enables each controller to get read- and write-access to the memory of the other controller. The flight control board includes all sensors for providing necessary data to the stability and height controller. For derive the actual heading with reference to the geodetic north an external inertial measurement unit (IMU) with nine degrees of freedom is used. The other data of this external IMU is used as reference in the sensor processing. The flight management board contains all necessary components for the navigation controller, data logging, and communication with a ground control station as well as with a wireless sensor network which is explained later.

Figure 2.7 also shows the communication frequencies between the components. Especially the update rates of the gyroscopes and the accelerometers are much higher compared to the previously presented architectures. The availability of more sensor data enables a better filter and pre-processing before feeding the data into the safety-critical stability controller. Also the communication frequency between the two controllers is doubled with respect to the avionics
shown in Figure 2.5. This has been done to enable the flight control board to get a higher update rate of the values from the navigation controller which is running at the flight management board at a frequency of 100Hz (10ms periodic cycle time).

The main challenge of the work was an autonomous landing maneuver on a moving platform. To handle this challenge two main components have been developed. A wireless sensor network for accurate localization of the multi-rotor system near the moving landing place and algorithms which enables 3D navigation control.

The wireless sensor network consists of five radio nodes with an update rate of 14Hz. One mounted on the multi-rotor system itself and four at the edges of the moving landing platform. The radio nodes contain two localization techniques, transit time measurements of radio messages and ultrasonic impulses. For a landing maneuver the multi-rotor system first flies by navigating with GPS data. If the system gets in range of the radio nodes the localization is done by logging the transit time of radio messages. When the multi-rotor system is between the radio nodes of the landing place, the ranging switches from radio messages to ultrasonic impulses. The multi-rotor system follows the movements of the landing platform and is able to land at the center of it. The ground contact is detected by the accelerometer which registers a negative acceleration at touchdown. The main difference between this landing maneuver and the one realized in [8] is, that the focus here is on an accurate positioning in x and y direction and not in a very smooth landing.
To enable the multi-rotor system to do an accurate positioning a 3D navigation controller was developed. It controls the position, the velocity and the acceleration of the multi-rotor system at a frequency of 100Hz. In every control loop a new target state is calculated which represents the actual deviation to the planned trajectory from a point A to a point B and not the direct deviation to the destination point B. So it is possible to choose the way how to come from A to B and to minimize the deviations to it in x, y and z direction. The controller is realized with modified PI controllers which can be found in [3]. This controller allows the multi-rotor system to fly safe to user defined points at defined trajectories (e.g. line, circle, etc.) as well as to follow its moving landing place and to land in the center of this place.

2.1.4 Requirements on multi-rotor systems

The three above presented avionics in combination enable a fully autonomous multi-rotor system which is able to execute planned tasks by itself. To realize the mentioned functionality many requirements are essential for a safe flying system. Our proposed approach has also to
meet these requirements. The derived requirements from our previous works are clustered and listed in the following categories.

**2.1.4.1 Requirements on the environment**

The operator of the system is responsible to meet the following requirements:

a) The environment must not disturb the radio signals of the remote control and other communication tasks.

b) The environment must not disturb the sensors of the multi-rotor system.

c) The environment has to allow a safe and collision-free flight without any obstacles.

d) The environment has to have eligible weather conditions (No rain, wind speed max. 3Bft. for autonomous take-off and landing).

e) The take-off and landing place has to have at least 15m x 15m free space.

f) The take-off and landing place has to be a planar surface without irregularities.

g) The surface of the take-off and landing place has to be plane and reflects ultrasonic, e.g. grass (up to 5cm), cobblestone, broken rock, asphalt, wooden panel or nonmagnetic metal (otherwise the compass will be disturbed).

**2.1.4.2 Requirements on the multi-rotor system**

In the following general requirements on the whole multi-rotor system are described.

a) The multi-rotor system has to have a stable behavior.

b) The multi-rotor system has its own on-board battery to power itself.

c) The multi-rotor system has to be within sight of the pilot at all time.

d) The multi-rotor system has to fly in range of the remote control and the radio module used for communication with a ground control station.

e) The Pilot is able to interrupt every action done by the multi-rotor system with the remote control.

f) The multi-rotor system has to have motors with enough power to produce double thrust of the weight of the multi-rotor system to enable agile flight characteristics.

g) Requirements on the avionics

The requirements on the avionics can be arranged in two categories, requirements on the software and requirements on the hardware. Most of the following requirements are annotated with time constraints in terms of frequencies of their execution because they require a real-time behavior. These frequencies are derived from the three presented avionics architectures.
They are representing the minimum experimentally identified frequencies needed for an agile and safe flying multi-rotor system.

**Software:** The avionics

- a) has to communicate its actual status and errors at a minimum frequency of 10Hz (100ms periodic cycle time).
- b) has to contain a signal processing for the remote control signals at a minimum frequency of 500Hz.
- c) has to contain a sensor processing at a minimum frequency of 500Hz (2ms periodic cycle time).
- d) has to contain a stability controller at a minimum frequency of 500Hz (2ms periodic cycle time).
- e) has to contain a computation for the final control values of the motor drivers at a minimum frequency of 500Hz (2ms periodic cycle time).
- f) should contain a height controller at a minimum frequency of 500Hz (2ms periodic cycle time).
- g) should contain a navigation controller at a minimum frequency of 100Hz (10ms periodic cycle time).

**Hardware:** The avionics

- a) has to have gyroscopes which allow measurements at a minimum frequency of 1kHz (1ms periodic cycle time).
- b) has to have accelerometers which allow measurements at a minimum frequency of 1kHz (1ms periodic cycle time).
- c) has to have magnetometers which allow measurements at a minimum frequency of 125Hz (8ms periodic cycle time).
- d) has to have a remote control receiver which communicates the transmitted control values at a minimum frequency of 45Hz (~22.2ms periodic cycle time).
- e) has to transmit the calculated control values at a minimum frequency of 500Hz (2ms periodic cycle time).
- f) has to have a radio module which allows a communication with a ground control station at a minimum frequency of 10Hz (100ms periodic cycle time).
- g) should have a pressure sensor to get information about the height of the multi-rotor system at a minimum frequency of 100Hz (10ms periodic cycle time).
h) should have a GPS receiver which transmits the positioning data at a minimum frequency of 5Hz (200ms periodic cycle time).

i) should have an ultrasonic rangefinder to measure its height above ground level (350cm in minimum) at a minimum frequency of 10Hz (100ms periodic cycle time).

j) has to be galvanically isolated from the motors.

k) has to have at least one controller which has enough performance to meet all mentioned real-time constraints.

![Diagram showing software execution and hardware communication frequencies together with data dependencies derived from the mentioned requirements.]

Figure 2.8 illustrates the data dependencies between the identified software components and the direction of data flow. In addition, the software components are annotated with the minimum execution frequencies mentioned in the requirements above. The communication paths between hardware components and the interfaces of the controller are also annotated with the corresponding frequencies.

These are the main requirements that need to be considered to get a safe flying system with the minimum functionality including a manual remote controlled flight and optional height and navigation control as well as autonomous take-off and landing maneuvers.

2.1.5 Discussion on extensibility

The extensibility of the avionics used in [1], [8] is strongly limited. Both interfaces and performance of the used microcontrollers are very restricted. Many of the available interfaces...
are used by the connected peripheral devices, e.g. all sensors, GPS receiver, remote control receiver, motor drivers, etc. Thus it is difficult to connect new devices, like microcontrollers for every payload processing task to the existent avionics. But to handle high-performance payload processing more performance and accordingly new processing elements are needed. The avionics used in [8] has the availability to connect new extensions to the Navi-Ctrl PCB via an I2C bus, but the bandwidth of this interface is very limited and shared with all devices connected to this bus. This is also applies for other interfaces like UART or SPI. To conclude, these types of interfaces are not qualified to transfer heavy data like images or video streams.

The extensibility of the avionics used in [3] is better with respect to the available interfaces and performance. But also with this avionics it is not possible to handle compute intensive payload tasks like video or other heavy data stream processing. Also the other approaches for avionics architectures presented in section 2.1.2, with the exception of the one developed by Ascending Technologies, are not able to handle high-performance payload processing on-board of the multi-rotor systems. Ascending Technologies’ approach shows that high-performance payload processing is a necessary and interesting function. It enables the multi-rotor system to process complex data of special sensors or cameras on-board and to immediately use the results in other tasks, e.g. landmark tracking or collision prevention. By using off-board processing, latencies are coming into existence through the transmission paths which causes new problems.

Together with the need for high-performance payload new requirements arise. More performance means also powerful controllers with higher power consumption. While the multi-rotor systems are battery powered the power consumption represents an influential role. The main challenge is to balance between the needed performance and the maximum power consumption which influences the flight time of the multi-rotor system within the range of minutes.

### 2.1.5.1 Requirements for high-performance payload processing

a) The payload processor has to have enough performance to process the individual payload tasks.

b) The power consumption of the payload processor has to be smaller than 15W.

c) The size and weight of the new hardware which contains the payload processor has to be chosen by the available space and thrust of the used multi-rotor system.

d) The payload processor has to provide interfaces for connection to the existent avionics.

e) The payload processor has to provide interfaces for connection to the payload.

If a payload processor will be added to the existing avionics, payload processing becomes possible. But the communication paths between the processing elements of the avionics represent an abiding bottle neck. The approach of Ascending Technologies uses an Intel Atom or i7 processor for onboard processing. These processors are clearly violating the requirement of a maximum power consumption of 15W. Our approach, which enables high-performance payload processing while using a single MPSoC for the whole avionics will be presented in the next section.
2.1.6 Proposed multi-core MPSoC based avionics

Since there are MPSoCs available which have high performance but also small power consumption, our approach tries to introduce mixed-criticality at avionics for multi-rotor systems which enables also the possibility of high-performance payload processing. Mixed-Criticality in our context means: running safety and real-time critical tasks together with uncritical tasks on the same MPSoC. By running mixed-critical tasks on one MPSoC new challenges are arising which will be presented in the following.

2.1.6.1 Platform description

Our decision was to use an MPSoC which combines multicore processors and programmable logic. The Xilinx ZYNQ 7020 [9] includes both technologies in one housing. The multicore processor is represented by a dual ARM Cortex -A9 MPCore at 866MHz and an Artix-7 FPGA with 85k logic cells. The development with this MPSoC is fully supported by the Xilinx Vivado Toolchain. The structure of this MPSoC is shown in Figure 2.9.

The ARM dual core is connected to the peripherals by the AMBA® Interconnect. On the left side the available interfaces are shown which can be connected to the pinout of the MPSoC by the Processor I/O Mux. The AMBA Interconnect represents also the interface to Multiport DRAM Controller and the Flash Controller as well as the connection to the Programmable Logic (FPGA) part of the MPSoC.

With this flexible and heterogeneous MPSoC it becomes possible to process the presented safety and real-time critical flight algorithms together with the mission-critical payload tasks on a single MPSoC. With the Artix-7 FPGA it is possible to define and build further interfaces, processing elements (e.g. MicroBlaze Softcore) or also specialized hardware for the payload processing tasks. While the development and production of an own board is very time and cost intensive we decided to use an industry board which is available at Trenz
Electronic GmbH [10]. The TE0720-01-2IF board, shown in Figure 2.10, has a dimension of 50mm x 40mm, a weight of 20g and a power consumption of 10W. It is connected to an individual breakout board via the assembled industry connectors on the bottom of the board.

The individual breakout board will contain all needed interfaces for the connection to the motor drivers, remote control receiver, sensors, radio module, network, USB, SD-card, etc. With a SD-card it is possible to run a Linux operating system on the dual ARM core as also to log telemetry or payload processing data on it. Figure 2.11 shows our proposed avionics architecture with the Xilinx ZYNQ 7020.

Most of the needed sensors will be connected to the MPSoC via an $I^2$C bus, including the gyroscopes, accelerometers, magnetometers, pressure sensor and an ultrasonic rangefinder. The motor drivers will get a dedicated $I^2$C bus, because the communication bandwidth of one bus will be too small to connect both sensors and motor drivers. The GPS module, the radio module and the remote control receiver, will be connected to a dedicated UART interface each. The hardware needed for the uncritical payload task will be connected to the ZYNQ 7020 with an adequate interface. While the ZYNQ has by standard only two UART interfaces an extra UART interface has to be implemented in the Programmable Logic part. The communication rates between the components will be chosen so that they are meeting the named requirements with their timing constraints.
2.1.6.2 Software mapping possibilities

The Xilinx ZYNQ 7020 offers many possibilities for mapping and implementing the functionality on the provided processing elements. In the following, two possible implementations will be presented. Since safety and real-time critical task are running on the ZYNQ 7020 real-time capable operating systems will be omnipresent in the shown approaches. The tasks running on the MPSoC can be assigned different levels of criticality. We have allocated the following three different criticality levels:

- Safety-critical: Tasks belonging to this class have to meet hard real-time constraints. They are needed for safe flying and an error will cause a crash-landing and loss of the multi-rotor system. To this class belong the flight algorithms, e.g. sensor and set point processing, stability, height and navigation controller and calculation of control values.

- Mission-critical: Tasks which are mission-critical are not needed for a safe flight behavior of the multi-rotor system, but they may have real-time constraints. If a mission-critical task generates an error or fails, the multi-rotor system will not make a crash-landing, but the planned mission cannot be correctly executed. Tasks which are arranged in this category are typical payload processing tasks (e.g. video processing, software-defined radio of communication transceivers, etc.).

- Uncritical: In this last category tasks will be arranged which are neither needed for a safe flight behavior of the multi-rotor system nor for the correct execution of a planned mission. In this class the status display via on-board LEDs can be found for example.

Figure 2.11 Proposed avionics architecture with the Xilinx ZYNQ 7020
Figure 2.12 presents our first software mapping approach. Both, the safety-critical tasks, e.g. the flight algorithms, and the mission-critical as well as the uncritical tasks will be processed on the ARM dual core. To guarantee a firm execution of the safety and real-time critical tasks, e.g. the flight algorithms, as well as of the mission-critical tasks, e.g. the payload processing, an Asymmetric Multi Processing (AMP) real-time solution will be used. Together with the Linux kernel, which processes the uncritical tasks, a Xenomai kernel [11] executes the safety and mission-critical tasks using priorities. The Xenomai kernel has the capability to block the Linux kernel from execution and to generate a complete context switch of the ARM dual core.

![Xilinx ZYNQ 7020 - MPSoC](image)

**Figure 2.12 First mapping approach**

Figure 2.13 Second mapping approach shows our second software mapping approach. The ARM dual core will only be used by the mission-critical and uncritical tasks, while the real-time and safety-critical flight algorithms are running on a dedicated MicroBlaze softcore processor which uses the FreeRTOS™ [12] embedded real-time operating system. FreeRTOS™ will guarantee that the safety-critical tasks meet their real-time constraints. On the ARM dual core again a combination of Linux and Xenomai kernels will be used. This will be done to guarantee a firm execution of the mission-critical tasks with higher priority than the uncritical tasks. Here, the FPGA will be used for the implementation of additional interfaces and the MicroBlaze softcore processor. Since the mission-critical payload tasks might require most performance, they can run on the ARM dual core. The main challenge of this approach will be the communication between the different processing elements, if they have to share data with each other, and the sharing of the memory interface of the MPSoC
Another possibility for the implementation of custom payload processing is the implementation as custom hardware in the FPGA part of the ZYNQ 7020 MPSoC, e.g. using the Xilinx Vivado High-Level Synthes (HLS) Tool. Since the payload processing task is not further specified in this use-case description yet, this mapping option is not further detailed.

### 2.1.6.3 Mixed-criticality issues

The main problem to be solved with our proposed approach is the ability to process payload tasks on-board of the multi-rotor system. This will result in savings of the power consumption of the overall avionics because only a single MPSoC will be used. There will also be savings in size and weight of the avionics, in particular less boards are needed. But while all tasks will be processed on a single MPSoC new problems arise. First of all the scheduling via an RTOS has to guarantee a frictionless computing of all tasks on all processing elements of the MPSoC. The presence of shared caches, memories and peripherals has to be analyzed as well as the communication between safety-critical, mission-critical and uncritical tasks.

Especially in Figure 2.12 a main problem will be a crash of the Linux operating system which will also cause a crash-landing of the multi-rotor system. This problem may be solved by calculating the safety-critical flight algorithms e.g. on three or more processing elements and check the plausibility of the control values of each processing element. This leads to other possible software mappings which are not discussed in this use-case description. In addition to these real-time constraints, memory management/access protection and error robustness properties also problems of physical nature are able to occur. In particular power consumption and temperature problems. Especially in battery powered systems, like the multi-rotor systems, power consumption has an important role. If the error-free execution of safety-critical tasks cannot be guaranteed, processing elements with mission-critical or uncritical tasks have to be powered off down for example. And even more for temperature issues. Here the main question is: What happens if the execution of a mission-critical task overheats the MPSoC and a power management decides to reduce the clock frequency (DFVS) to protect the chip? Even under these conditions, the correct and timely execution of safety-critical tasks has to be guaranteed.
2.1.7 Requirements regarding non-functional properties

Several scenarios such as autonomous take-off and landing, manual flight as well as 3D navigation are addressed in the UAV use case as they have been described in the above sections. In the following the requirements are summarized regarding the none-functional properties that are of importance for the use-case evaluation.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Importance</th>
<th>Impacts on</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>The flight control algorithms have to meet their deadlines.</td>
<td>M</td>
<td>Safety, performance of processor, implementation</td>
<td>The non-safety critical tasks must not disturb the execution of the flight algorithms.</td>
</tr>
<tr>
<td>The communication with the remote control has to meet its deadlines.</td>
<td>M</td>
<td>Safety, performance of processor, implementation</td>
<td>During manual flight it is mandatory to have a stable connection to the remote control.</td>
</tr>
<tr>
<td>The communication with the ground control station should be stable.</td>
<td>A</td>
<td>Quality of Service, debugging capability</td>
<td>The multi-rotor system can transfer telemetry data to a ground control station to determine its present state.</td>
</tr>
<tr>
<td>The power supply of the avionics should last longer than the one which will drive the motors.</td>
<td>M</td>
<td>Power consumption of avionics</td>
<td>The avionics and the motors will have separate power supplies.</td>
</tr>
<tr>
<td>The overall weight of the multi-rotor system should be as low as possible</td>
<td>NTH</td>
<td>Selection of computing platform for avionics and payload processing</td>
<td>The multi-rotor system should carry as much payload as possible while maintain a safe flyable system.</td>
</tr>
</tbody>
</table>

M= Mandatory, H= High, A=Average, NTH= nice to have

From the requirements identified above, quantitative non-functional requirements have been derived and summarized in the table below.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>NF-measure</th>
<th>Requirement</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor update rates (all on-board sensors, incl. GPS)</td>
<td>Time</td>
<td>Meet the needed update rate for each sensor</td>
<td>See Section 2.1.4.2 for the detailed update rates.</td>
</tr>
<tr>
<td>Remote control update rate</td>
<td>Time</td>
<td>Meet the needed update rate for the remote control values.</td>
<td>See Section 2.1.4.2 for the detailed update rate.</td>
</tr>
<tr>
<td>Power consumption of the avionics including payload processing</td>
<td>Power</td>
<td>TBD, see Section 2.1.5.1 for an approximation</td>
<td>While payload processing has a lower level of criticality it can be switch off or the QoS can be reduced to secure the safe execution of flight algorithms.</td>
</tr>
<tr>
<td>Dense integration of mixed critical tasks for avionic and payload processing on one computing platform</td>
<td>Weight</td>
<td>&lt; 150g</td>
<td>Including all processing infrastructure and necessary cooling mechanisms.</td>
</tr>
<tr>
<td>Temperature constraints of processing elements</td>
<td>Temperature</td>
<td>&lt; as specified by used components, e.g. 85°C for ZYNQ (industry version)</td>
<td>The temperature of each processing element of the avionics has to be within its specified operating range.</td>
</tr>
</tbody>
</table>


### 2.1.8 Power consumption estimation of demonstrator

With the knowledge of the used parts we are able to make a first worst-case estimation of the demonstrator’s power consumption. Especially the power consumption of the avionics is of importance, because it will get its own battery which is separated from the motor circuit’s battery. The separation of the power supplies will be done to avoid electrical interferences between motors and avionics. With an own battery the power supply of the avionics has to last longer than the one of the motors, which was mentioned in the requirements regarding non-functional properties. In the following we will present the worst-case power consumptions for the motor circuit and the avionics circuit to estimate their envisioned operation times.

The chassis will be presented by a Quadro XL multi-rotor-system by MikroKopter (HiSystems GmbH). We will use the frame, motors and motor drivers from this company. In addition to these components the 3-axes gimbal-mount for the camera will also be powered by the motor circuit. These parts will be powered by a 6600mAh 4S (14.8V) lithium polymer battery (98Wh).

<table>
<thead>
<tr>
<th>Component</th>
<th>Power consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Motors</td>
<td>~1440W (maximum motor power)</td>
</tr>
<tr>
<td></td>
<td>~320W (average for 2.5kg system)</td>
</tr>
<tr>
<td>4 Motordriver electronics</td>
<td>~10W</td>
</tr>
<tr>
<td>Gimbal</td>
<td>~20W</td>
</tr>
<tr>
<td>Overall</td>
<td>~1470W (maximum motor power)</td>
</tr>
<tr>
<td></td>
<td>~350W (average for 2.5kg system)</td>
</tr>
</tbody>
</table>

Under the maximum power the motors will produce an ascending force of ~86.3N (~8.8kg). This value will be reached by hectic maneuvers, but not in stationary flights. The average power consumption of a multi-rotor-system with a weight of 2.5kg is 320W which has been measured empirical. This leads to an estimated flight time of ~16.8 minutes. With 10% tolerance we end up by ~15 minutes.

We decided that the battery of the avionics has to last for at least 30 minutes. Thus, the size of the avionics battery will be chosen after estimating the power consumption of its components. The main parts of the avionics are listed in the following table.

<table>
<thead>
<tr>
<th>Component</th>
<th>Power consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZYNQ board including external memory, flash, IO, dc/dc converter</td>
<td>~10W</td>
</tr>
<tr>
<td>Remote receiver</td>
<td>~0.5W</td>
</tr>
<tr>
<td>Sensors (gyros, acceleration, hall, pressure, gps)</td>
<td>~1.0W</td>
</tr>
<tr>
<td>Component</td>
<td>Power Consumption</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Camera</td>
<td>~2.5W</td>
</tr>
<tr>
<td>WiFi-stick</td>
<td>~2W</td>
</tr>
<tr>
<td>SD-card</td>
<td>~1W</td>
</tr>
<tr>
<td>Other (level-shifters, LEDs, etc.)</td>
<td>~5W</td>
</tr>
<tr>
<td>Overall</td>
<td>~22W</td>
</tr>
</tbody>
</table>

With a little overestimation we will set an overall power consumption of the avionics and the mentioned components at 25W. This leads to a required battery with ~12.5Wh for the avionics. We choose a 1300mAh 3s (11.1V) lithium polymer battery which has ~14.4Wh. This will fulfil our requirements for the up times of the multi-rotor-system.

### 2.1.9 Conclusion and future work

In this use-case description we have presented three different avionics architectures ranging from simple flight control to advanced 3D navigation for a multi-rotor system. We have proposed the combination of these avionics for building a fully autonomous multi-rotor system, capable to integrate different user-defined payload units (e.g. camera, antenna or any other advanced sensor system). The main challenge is represented by the need of flexible and high-performance on-board payload processing. Since the performance and interfaces of existing and used avionics architectures are insufficient, we propose a new approach based on a single chip solution. Our decision is to use a heterogeneous MPSoC which combines an ARM dual core and a FPGA in one chip. The Xilinx ZYNQ 7020 MPSoC has enough performance to process payload tasks like inflight video processing. In combination with the Xilinx MPSoC we have presented possible software mappings capable to run mixed-critical tasks on the MPSoC. Appropriate temporal and spatial segregation techniques supported by hardware and software will enable isolated execution of applications with mixed-criticalities.

In future work we are going to use a power aware virtual platform of the Xilinx ZYNQ MPSoC for assessing power and timing properties of the avionics under different flight profiles, mission scenarios and custom payload configurations (based on the flow in [13], [14]). The power model shall be capable of generating workload-specific power over time traces for each processing element of the MPSoC. In combination with the floor plan and a package temperature model of the ZYNQ (based on the flow in [15]), a temperature model will be updated during simulation to gain information about the thermal influence and coupling between the used processing element. This information can be used for the development and testing of a power and temperature management system, capable to guarantee undisturbed operation of all safety-critical applications under heavy payload processing and extreme environmental temperatures.
2.2 Use-Case 1b: Remotely Piloted Aircraft – FCC’s I/O module

2.2.1 General ATLANTE program overview

The objective of the ATLANTE Program is the development of a Tactical Remotely Piloted Aircraft (RPA) to be used for operations like adjustment of a fire mission, damage assessment and ISTAR (Intelligence, Surveillance, Target Acquisition and Reconnaissance).

The basic configuration for the ATLANTE system includes 4 RPAs, a Ground Control Station, a Data Link Station, a Launching and Recovery Transport unit and a Maintenance unit.

The ATLANTE system is foreseen to operate in all-weather conditions in a 24 hour operation basis. The RPA it is capable of using a parachute landing system, to be launched from a catapult, or to use a normal landing gear when taking off and landing on a runway.

![ATLANTE RPA](image1)

Figure 2.14 ATLANTE RPA

2.2.2 ATLANTE Flight Control Computer (FCC)

As part of this program, GMV developed a Flight Control Computer (FCC) for the Air Vehicle.

![ATLANTE FCC chassis](image2)

Figure 2.15 ATLANTE FCC chassis
This Flight Control Computer (FCC) provides, among others, the following functionalities:

- Navigation
- Guidance and Control Logic Moding
- Stability and Control
- Actuators Control and Monitoring
- Failure Detection and Management
- Flight Termination Trigger (using a recovery system consisting of a parachute)

In order to provide these functionalities, the Flight Control Computer (FCC) includes sensor devices as well as processing units for the logics of the system. It also implements Automatic Take Off and Landing (ATOL) modules.

The FCC interfaces with the Mission Computer to provide the actual state of the Remotely Piloted Aircraft (RPA) and to obtain its desired state and with the Flight Termination Unit to trigger the flight termination when needed. The FCC directly manages the control surfaces and the engine control unit, being a critical system within the aircraft.

Furthermore, GMV has developed an ATOL module for the Ground Segment and a FCC Test Computer, which is an AGE for FCC maintenance, FCC software loading and actuators calibration purposes.
2.2.2.1 Flight Control Computer (FCC) Hardware architecture

The following figure shows the FCC hardware elements and interfaces:

![Diagram of FCC hardware](image)

Figure 2.16 Hardware avionics and interfaces of the ATLANTE FCC

The different HW elements depicted in Figure 2.16 will be detailed herein:

**Sensors:**
- GNSS receiver.
- IMU (Inertial measurement unit).
- Radar-altimeter.
- Absolute and differential pressure transducers.
- Magnetometer.
Actuators:

- Surface actuators. Each surface actuator is controlled through a PWM line connected to the I/O card. Each surface actuator provides feedback through differential analogue lines.
- Engine actuator. The engine actuator is controlled through a PWM line connected to the I/O card.
- Steering wheel actuator. The steering wheel actuator is similar to the surface actuators, being controlled through a PWM line connected to the I/O card. The steering wheel actuator provides feedback through a differential analogue line that needs to be connected to an A/D converter at the I/O card.

External sensors and devices:

- Angle of Attack (AoA) and Total Air Temperature (TAT).
- Heater.
- Safety pin. An input digital line shall be used for enabling or disabling the surface actuators and steering wheel actuator.
- Hook. Input digital lines shall be used for getting Weight-on-Wheels (WoW) data.

Other ATLANTE subsystems:

- Mission computer.
- Test Port. A serial port bus for maintenance and debugging purposes.
- Engine Control Unit. A serial port bus for acquiring data from the Engine for monitoring purposes.
- Flight Termination Unit (FTU). One output digital line shall be connected from the FCC I/O card to the FTU in order to activate the termination procedure.

2.2.2.2 Flight Control Computer (FCC) Software architecture

The main SW components of the FCC application include:

- I/O module

  It collects all the data from the I/O devices (RS232/422 based devices, analogue lines and digital lines) and sends the different commands/data to them.

  It checks the validity of the I/O data and formats it into a suitable form for the Navigation module, informing the FDIR module of any detected malfunction: corrupted data, data out of bounds, delayed or undetected data received, etc.

- Navigation

  This module includes all the operations required to provide any sensible data that the stability and guidance algorithms might need.

  It performs the navigation algorithms required to obtain an estimation of the state of the aircraft. This includes processing and consolidation algorithms on data obtained from the sensor devices, such as GNSS algorithms (including normal navigation mode and ATOL algorithms), as well as IMU, magnetometer and air data processing algorithms.
• Flight Control Laws

This module processes mission-related commands/data coming from the Mission Computer (list of waypoints, demanded flight phase...), taking into account the navigation information, to provide the mission data that is required using guidance algorithms. This module also performs all the Stability and Control algorithms.

• Fault Detection, Isolation and Recovery (FDIR)

This module gathers all the internal built-in test results performed by all the modules within the FCC in order to assess the FCC health status (PBIT, IBIT and CBIT results) in such a way that can raise different alerts to the different FCC modules and the DUO.

This module would also decide whether the detected malfunctions within the system would make necessary to terminate the normal flight mode and activate the FTU.

• Real-Time Operating System (RTOS)

The different SW modules previously mentioned are integrated in a VxWorks 653 architecture taking advantage of the ARINC 653 partitions isolation concept.

2.2.3 Demonstrator based on the ATLANTE FCC I/O module

In order to demonstrate the benefits of CONTREX, a subset of the FCC software developed by GMV for a medium sized Remotely Piloted Aircraft (RPA), applicable for surveillance missions such as damage assessment and intelligence, will be reused. The demonstrator will implement a subset of the FCC components, including most elements of the I/O module.

As described in previous sections, the FCC developed by GMV is in charge of the guidance, navigation and control of the Remotely Piloted Aircraft (RPA): it implements the flight control laws, which govern the aircraft’s handling characteristics. Timing and reliability requirements are critical in this system, but also heat dissipation, power consumption, temperature, and installation space and weight (SWaP). Moreover, safety concerns are strongly associated to this type of systems as the failure of a given component may have catastrophic consequences, as for instance making the RPA fall on populated areas or crash with another aircraft. Even when a failure does not incur direct human losses (as in the case of flying in segregated airspace), the loss of the vehicle supposes a great economic loss.

2.2.3.1 Extra-functional requirements on the demonstrator

The demonstrator shall fulfil the following high-level extra-functional requirements:

1. The demonstrator shall interact with the simulated sensor devices using the appropriate interface. The demonstrator shall configure each device to generate the required data with a minimum frequency specified in Table 2-1: Minimum device data frequencies.
<table>
<thead>
<tr>
<th>Device</th>
<th>Minimum Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNSS receiver</td>
<td>2Hz</td>
</tr>
<tr>
<td>IMU</td>
<td>50Hz</td>
</tr>
<tr>
<td>Magnetometer</td>
<td>10Hz</td>
</tr>
<tr>
<td>Pressure transducers</td>
<td>10Hz</td>
</tr>
<tr>
<td>Radio Altimeter</td>
<td>25Hz</td>
</tr>
<tr>
<td>ADC Signals (Angle of Attack,</td>
<td>50Hz</td>
</tr>
<tr>
<td>Total Air Temperature) TBC</td>
<td></td>
</tr>
</tbody>
</table>

Table 2-1: Minimum device data frequencies

2. The demonstrator shall decode and time-stamp the data received from every sensor device at the minimum frequency specified in Table 2-1: Minimum device data frequencies.

3. The demonstrator shall log telemetry data to the SD-Card (2Hz) or send them through one of the 10/100/1000 tri-mode Ethernet peripherals.

4. The power consumption of the demonstrator (including all the components of the selected platform, such as external memory, flash or IO devices) has to be smaller than 5 W.

2.2.3.2 SW architecture of the demonstrator

The SW architecture of the demonstrator consists of a series of tasks in charge of managing the I/O devices. These tasks can be grouped according to their different levels of criticality, as shown in Figure 2.19.
Three levels of criticality have been considered: Safety-Critical, Mission-Critical and non-critical tasks. They have been established according to the importance of every I/O device data for the navigation, guidance and flight control algorithms.

Safety-Critical tasks are those processing tasks required for a safe flight. If any failure, data loss or delay were to happen, the navigation, guidance and control modules will fail to operate resulting in a system failure. Therefore, these tasks must meet hard-real time constraints. Tasks that fall into this category are the sensor processing for the IMU, GPS, Differential and Absolute pressure transducers and the Angle of Attack.

Tasks of the following criticality level, Mission-Critical, may not cause a system failure in the presence of an error but will be unable the aircraft to complete its mission parameters. An example of mission-critical task is the component of the I/O partition of the FCC that cooperates with a dedicated mission computer for mission data acquisition. This component will not be implemented in the demonstrator, however. In its stead, less critical sensor processing tasks have been included in this category. They are responsible for communication with data sensors such as the Radio Altimeter, Magnetometer and Total Air Temperature.

The last category encompasses the so called non-critical tasks. These tasks are not critical to flight safety or mission goals. Within the demonstrator, data logging tasks fall into this category.
2.2.3.3 Proposed multi-core MPSoC platform

The multiprocessor System-on-Chip (MPSoC) is a system-on-a-chip (SoC) which implements a variety of processing elements. MPSoCs are usually targeted for embedded applications given their high processing capabilities while limiting the power consumption through the use of specialised processing elements and architecture.

The proposed platform for the present demonstrator (the I/O module of the ATLANTE FCC) is a MPSoC based on the Xilinx All Programmable SoC Architecture. The Xilinx Z-7020 [9] is a product of Xilinx Zynq-7000 family that integrates a dual-core ARM® Cortex™-A9 based processing system (PS) and 28nm Xilinx programmable logic (PL) in a single device.

The key advantages provided by this platform are performance and flexibility. With this flexible and heterogeneous MPSoC it becomes possible to (1) extend the platform with new I/O interfaces or new specialized processing elements while (2) deploying a multi-threaded application on a multi-core system under real-time and mixed-criticality constraints.

The TE0720-01-2IF industry board shown in Figure 2.10 (available at Trenz Electronic GmbH [10]), which includes the previous Xilinx Z-7020 MPSoC, will be used.

2.2.3.4 Hardware mapping possibilities

Given the flexibility of the proposed platform, all necessary hardware components can be allocated on the Xilinx Z-7020 board either using existing dedicated elements or by implementing new elements within the Programmable Logic part.

The Figure 2.18 illustrates the hardware mapping with all interfaces necessary to connect the different data sensors and signals.
All current data sensors GPS, IMU, magnetometer, 2 pressure transducers and radio altimeter use RS232/RS422 serial buses to send and receive data. AoA and TaT sensors use analog signal (the feasibility of the analogue interface is actually under consideration). The only two dedicated UARTs provided by the Xilinx Z-7020 could be assigned to the IMU and GPS data sensors with baud-rates sufficient to handle the high data traffic from those devices. This means that four UARTs shall be implemented in the Artix-7 FPGA in order to send/receive data from the Radio Altimeter, pressure transducers or magnetometer.

Telemetry data can be logged into the SD-Card or sent through one 10/100/1000 tri-mode Ethernet peripherals.

### 2.2.3.5 Software mapping possibilities

The Xilinx ZYNQ 7020 is a multi-core board and so it offers already an advantage over the single core processor used in the FCC. Actual parallel thread execution allows a significant increase in performance for the overall system and higher executing rates for tasks that otherwise may be scheduled away.

Additionally, the concept of mixed criticality must be addressed. The term mixed criticality system (MCS) describes those systems in which applications of different levels of criticality run and interact on the same hardware platform. For the purpose of this demonstrator, the software should be adapted into a model with different levels of criticality that will be deployed on a Xilinx Z7020 board.

The usage model chosen for software mapping is based in the asymmetric-multiprocessor (AMP) mode. In this mode, the processor cores in the device are largely unaware of each other. Separate OS images exist in main memory, though there may be a shared location for inter-processor communications.
Among the many configurations for AMPs shown at [19] the Xenomai dual-kernel approach seems to be the most interesting for the goals of this use case.

The first configuration is as follows:

**Xenomai dual-kernel approach.**

In this first approach all different tasks are assigned to the ARM dual core CPUs. Xenomai is a dual-kernel, open source real time solution for Linux. It uses a separate kernel (and separate API) for real-time tasks, in our case safety and mission critical tasks. A different Linux kernel is used for non-critical tasks. Dual-kernel system actually means that two kernels run concurrently without any synchronization between them. To server real-time event however, one kernel must run with higher priority than the non-real-time kernel, so the first may preempt the latter with no delay. In this approach, the Programming Logic part will be utilized only to define the necessary additional interfaces (UARTs, etc).

Figure 2.19 illustrates this configuration:

![Figure 2.19 Approach to software mapping on the Xilinx ZYNQ](image)

**Xenomai dual-kernel approach extension: MicroBlaze softprocessor.**

The Xenomai approach could be later extended takes advantage of the possibility of implementing a Microblaze soft processor in the programmable logic (PL). This configuration will allow to balance the use of the ARM core running the Xenomai kernel by moving some tasks to the Microblaze soft processor. An interesting solution would be to move safety-critical tasks such IMU and GPS sensor processing to this softcore.

Building upon the AMP mechanism, an independent operating system -with respect to that running on the ARM cores- can be run on the Microblaze processor or tasks could just be implemented bare metal.
Figure 2.20 illustrates this configuration:

![Diagram of software mapping on the Xilinx ZYNQ]

Figure 2.20 Second approach to software mapping on the Xilinx ZYNQ
3 Use-Case 2: Automotive Telematics

At present, several non-automotive companies provide private and/or fleet vehicle drivers with a support service in case of accident. The architecture is based on three main components: a sensing unit for acceleration measurements, a localization unit for GPS reading and a data processing and communication for identification of accidents and communication of position data either to public authorities (hospital, police) or private support providers.

Cobra Telematics and a few other companies are extending this scenario by moving part of the processing from the in-vehicle ECU to a remote data center. The acceleration sensing unit and the GPS localization unit are still installed on the customer vehicle but perform only limited processing while extensive processing is in charge of the data center. This already allows providing improved services to end users.

This approach, shown in Figure 3.1, is completely automated, from data collection (on the car) to data processing for accident recognition (on the data center) and operator call to the driver.

![Figure 3.1 Automotive use case overall scenario](image)

Furthermore, operators are trained to collect additional information when contacting the driver. This is aimed at refining the automated procedure of accident classification.

3.1 State of the art

The entire system chain currently implemented and deployed by Cobra is structured into three subsystem:
The **sensor node**. A tiny node integrating a tri-axial accelerometer, a Cortex-M0 core and a flash memory to store crash data and all configuration parameters. The node is responsible for sampling the accelerations at 80Hz, filtering them, running the crash detection algorithm and saving a snapshot (from 4s before to 3s after the instant identified as crash) of the acceleration in case of crash. The sensor node also implement a simple serial protocol supporting command for configuration, diagnostics, synchronization with the Cobra ECU and for retrieving the crash snapshot.

- The **Cobra ECU** integrates a Cortex-M3 core running native code and a Java application, a GPS receiver and a GSM/GPRS transceiver to communicate with the server-side application in the control room. The main goal of the ECU is that of retrieving the crash snapshot, combining acceleration information with speed and position data from the GPS and sending it over the wireless link to the control room servers.

- The **Control Room** servers are responsible for receiving the snapshots of the crash, storing it in a database, processing the crash data (acceleration, position and speed) and generating a textual report of the accident. Algorithms running on the server discriminated between moderate entity crashes and severe ones automatically, supporting the call center personnel to decide the suited intervention, such as contacting the driver, calling road support, ambulance, police, etc.

Being after-market devices, neither the sensor node nor the Cobra ECU can interact with the main vehicle ECU. The only connection between the devices is the power supply. This is shown in Figure 3.2.

![Figure 3.2 Overall BlackBox architecture and connections to the vehicle](image)

The evolution of the present system is mandatory for Cobra innovation and strategic plans. Based on the experience on the field, with more than 150,000 black-boxes installed, Cobra has identified four “scenarios” that highlight current limitations of the system and the corresponding improvements. The following sections describe such scenarios.

### 3.2 Scenarios

The Figure 3.3 shows the identified applicative scenarios that will drive the process of functional specification, non-functional requirements identification and product development.
Each scenario, discussed in the following sections, identifies some limitation of the current implementation on the system, a set of high-level requirements classified by importance and describes the proposed approach to evolve the system.

### 3.2.1 Scenario 1: Device installation

The sensor node installation procedure is currently in charge of non-specialized dealers and for this reason should be kept as simple (to avoid installation problems) and as quick as possible (to reduce installation costs).

One of the key aspects of the installation procedure consists in the positioning and orientation of the device. Given the current implementation of the system the sensor node must be installed in such a way that the X-axis of the accelerometer is oriented exactly as the back-to-front axis of the vehicle and the Y-axis must be perpendicular to gravity. The Z-axis, finally, need not to have a specific orientation. This allows compensating the inclination of the device with respect to the ground (or, equivalently, to gravity) with a simple rotation around the Y-axis and can be done by detecting the components of the gravity acceleration on the X and Z axes of the accelerometer. The diagram in Figure 3.4 shows this situation.

![Figure 3.4 Current orientation constraints](image)
The main problems that have been emerged from a analysis performed by Cobra are:

- Precise installation procedure requires long time and constitutes a significant cost.
- Personnel (third-party) is not always reliable or fully controllable.

This leads to two main drawbacks:

- **Wrong orientation** of the device leads to a very high number of false-positive events to be communicated to the Cobra call center. This, in turns, increases personnel costs, communication costs and computational requirements on the server-side.

- **Unreliable installation** produces perturbations on signals (offsets and vibrations) making hard or even impossible the server-side crash reconstruction procedure. Need of a maintenance of the installation is an extra cost to be possibly avoided.

The main goal in this scenario is to allow an unconstrained installation, as shown in

![Diagram showing generic orientation of the sensor node w.r.t. the vehicle reference axes](image)

Figure 3.5 Generic orientation of the sensor node w.r.t. the vehicle reference axes

To this purpose it is necessary to develop an algorithm for automatic detection of the sensor node orientation (self-calibration) with respect to the absolute (vehicle) axes. To this problem two approaches are possible:

- **Exact approach.** To determine the exact orientation of the device accelerations and angular rotations are necessary. This requires a 6-DOF sensing unit combined with suitable filters for sensor fusion. The advantage of such solution is that it is well-known in literature and is potentially very accurate. The cost of a 6-DOF device, though, might be unaffordable with respect to competitors.

- **Approximate approach.** An alternative solution might be based on accelerations only by statistically analysing a sequence of estimates performed in an interval of time of 200-300ms, immediately after the vehicle start moving. The advantage of such a solution is the simpler implementation of the algorithm (less computational power required) and the lower cost of the sensor node, which only requires a tri-axial
accelerometer. The limitation, on the other hand, lay in the reduced accuracy of the estimation of the orientation.

In a wider view, the high-level requisites associated with the installation problem of this scenario are summarized in the table below.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Importance</th>
<th>Impacts on</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>No ad-hoc equipment shall be needed for installation</td>
<td>M</td>
<td>Installation cost</td>
<td>Short installation time. Any non-specialized installation facility can be used.</td>
</tr>
<tr>
<td>No specific driving maneuvers shall be required</td>
<td>H</td>
<td>Installation cost</td>
<td>Short installation time</td>
</tr>
<tr>
<td>Good accuracy in angles estimation shall be achieved</td>
<td>M</td>
<td>Quality of the crash reconstruction on server application. Reduction of the number of false positives</td>
<td>Should be comparable with ad-hoc manual tuning. For the approximate approach, little degradation is acceptable</td>
</tr>
<tr>
<td>Low cost</td>
<td>H</td>
<td>Market competitiveness</td>
<td>Possibly without compass and gyroscopes (iNemo used as functionally optimal reference implementation)</td>
</tr>
<tr>
<td>Automatic detection of orientation angles shall be rapidly converging</td>
<td>H</td>
<td>Early insurance coverage of the driver</td>
<td>For the approximate solution, a statistical analysis is necessary. To be compared solutions w/w-out other sensors</td>
</tr>
<tr>
<td>Reusability</td>
<td>NTH</td>
<td>Cost, customizability</td>
<td>Enable diversification of product lines</td>
</tr>
</tbody>
</table>

M= Mandatory,  H= High,  A=Average,  NTH= nice to have

The solution that will be experimented and implemented as prototypes within CONTREX will be initially based on the iNEMO platform. Such a solution will either be the final one or the reference golden-model for an optimized solution implemented on a smaller, low-cost ad-hoc platform.

This sensor node, in fact, is intended to be a very low-cost, small-sized and low power device performing self-calibration and basic crash analysis functionality. The evolution of this platform that will be developed during the project timeframe will:

- Implement self-calibration services, to simplify the installation procedure.
- Implement on-device axis rotation according to the actual node orientation.
- Improve crash detection algorithms.

Such improvements will significantly increase the power consumption of the device. The first consequence is that the Vehicle Main ECU might consider the current absorption out of the admissible ranges and thus probably due to a faulty subsystem (especially at key-off). This will cause the vehicle main ECU (see Figure 3.2) reacting by disconnecting the power supply
of the Cobra ECU. To cope with such limitations on the power consumption of the sensor node, a suitable layer for performance/energy monitoring and a simple yet effective run-time management will also be developed. Figure 3.2 shows a in more detail a scheme of the node and its interfacing with the main Cobra ECU. In the following we will refer to this sensor node as the **Low-cost Sensor Node**, in contrast to the high-end one, describe in the following scenario.

![Figure 3.6 Architecture of the low-cost sensor node](image)

The low-cost sensing unit will be implemented on the iNEMO-M1 platform, provided by ST.

![Figure 3.7 The iNEMO M1 System-on-Board](image)

The iNEMO-M1 is the first 9-DOF motion sensing System-on-Board (SoB) of the iNEMO module family. It integrates multiple MEMS sensors from ST and a computational core:

- **The LSM303DLHC e-compass module.** The e-compass module LSM303DLHC is a system-in-package featuring a 3D digital linear acceleration sensor and a 3D digital magnetic sensor. The accelerometer has full scales of ±2g/±4g/±8g/±16g and the magnetometer has full scales of ±1.3/±1.9/±2.5/±4.0/±4.7/±5.6/±8.1 gauss. All full scales available are selectable by the user.

- **The L3GD20 digital gyroscope.** The L3GD20 is a low-power digital gyroscope able to sense the angular rate on the three axes. It has a full scale of ±250 / ±500 / ±2000 dps and is capable of measuring rates with several bandwidths, selectable by the user.

- **The STM32F103REY6 ARM® Cortex™-M3 32-bit microcontroller.**

This 9-DoF inertial system represents a fully integrated solution that can be used in a broad variety of applications such as robotics, personal navigation, gaming and wearable sensors for
healthcare, sports and fitness. A complete set of communication interfaces and motion-sensing capabilities in a small size form factor (13x13x2 mm) and the possibility to embed ST’s sensor fusion software makes the iNEMO-M1 system-on-board a flexible solution for high-performance, effortless orientation estimation and motion-tracking applications. The STM32F103REY6 high-density performance line microcontroller is the computational core of the iNEMO-M1 module. It operates as the system coordinator for the on-board sensors and the several communication interfaces. Exploiting the features of the MCU, the iNEMO-M1 offers a wide set of peripherals and functions such as 12-bit ADCs, DAC, general-purpose 16-bit timers plus PWM timers, I2C, SPI, I2S, USART, USB and CAN, that enable different operative conditions and several communication options.

The prototypical version of the sensor node (the one integrating the exact orientation estimation algorithm) will be developed based on the iNEMO platform and its non-functional aspects estimated and optimized. Trade-offs between accuracy and performance on one side, and power consumption and cost on the other, will possibly lead to the identification of a custom architecture capable of meeting all the specified requirements for the low-cost node. If not all the constraints will be met (both in terms of costs and performance/power trade-offs), a simplified firmware will be derived (approximate self-calibration). This might entail using a subset of the available sensors.

From the software point of view, the application will be structured as shown by the simplified software architecture shown in Figure 3.8 and to be refined during the project timeframe.

![Figure 3.8 Low-cost sensor node software architecture overview](image)

### 3.2.2 Scenario 2: Crash management

The crash management procedure involves the entire Cobra infrastructure, from the sensing node(s) to the control room serve-side applications and services. Improving the quality, the reliability and the effectiveness of the entire process is a key factor for Cobra competitiveness. An analysis of the current situation has led to the identification of the following problems:

- Quality of the acceleration signals is not sufficient for reliable crash analysis
- Cost of communication (proportional to data traffic) has significant impact
- Long and error prone compilation of insurance crash reports
- Cost of management of false-positives (control room) has significant impact
- Assistance is not tailored based on the number and condition of vehicle occupants.

A fifth problem is related to black-box portability. According to emerging European regulation in the field of insurance policies, it is desired that in the future there will be portability of black-boxes across different insurance companies and automotive telematics service providers. Though the matter is rather complex – mainly due to privacy of data and standardization at different levels – Cobra is starting to consider the problem related to crash management. Though this is related to crash management, more details are given in the discussion of Scenario 4, concerning B2B services.

From a top-level point of view, thus, the following requirements shall be satisfied.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Importance</th>
<th>Impacts on</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis range (duration of the crash picture)</td>
<td>H</td>
<td>Market competitiveness, Portability</td>
<td>Compatibility with customers adopting competitors. See more detail in Scenario 4.</td>
</tr>
<tr>
<td>The sampling rate and signal bandwidth shall be</td>
<td>A</td>
<td>Quality of the crash reconstruction</td>
<td>Good accuracy in estimation crash severity</td>
</tr>
<tr>
<td>False-positive shall be avoided</td>
<td>H</td>
<td>Management and personnel cost of the control room</td>
<td>Combine accurate crash detection algorithm on the device with post-processing and filtering on the server-side application</td>
</tr>
<tr>
<td>Transmitted data size shall be small</td>
<td>M</td>
<td>Management cost</td>
<td>Fares based on traffic. Storage space on servers</td>
</tr>
<tr>
<td>BOM cost shall be minimum</td>
<td>A</td>
<td>Cost</td>
<td>To be compared with dumb solution just sending data</td>
</tr>
<tr>
<td>Power consumption</td>
<td>NTH</td>
<td>Freedom of installation</td>
<td>To prevent detach by the on board electronics (low-cost node only)</td>
</tr>
</tbody>
</table>

M= Mandatory, H= High, A=Average, NTH= nice to have

To cope with the four main shortcomings identified and try to meet the requirements outlined, the following improvements are needed:

- Increase the **quality of acceleration data** to support improves crash reconstruction algorithms on the servers of the control room. This requires increasing the sampling frequency (currently 80Hz) and the measurement rage (currently ±6g). As a consequence, more processing power is needed on the low-cost sensor node. Since higher sampling frequency means more data to be transmitted to the server-side applications, a trade-off between data quality and communication cost is necessary.

- Improvement of the **crash detection algorithm** to avoid false-positive. Again, this requires more sophisticated algorithms on the low-cost sensor node. Current algorithm only consider acceleration profiles in the horizontal, ignoring vertical accelerations.
This lead to the identification of false-positive events cause, for examples by road holes or bumpers.

- Improvement of assistance. To this purpose it is necessary to identify the number of occupants of the vehicle and, as a long-term goal their physical conditions. While identification of the health status of the occupants is far too complex to be afforded in the project timeframe, experimenting solutions for the identification of the number of occupants is a first step that is considered both crucial and feasible within CONTREX. It is worth noting that identifying the number of occupants also supports insurances to avoid frauds. To implement a simple, prototypical system to solve this problem a smart-camera system is necessary.

Considering the three goals, thus an augmented solution is necessary, combining the low-cost sensor node with a High-end Sensor Node, as shown in Figure 3.9.

![Figure 3.9 High-end sensor node and overall architecture](image)

In the overall system the Cobra ECU will collect information from the two nodes independently and will operate as bridge to allow the two node to exchange data, mostly configuration options, parameters, alarms and diagnostic information.

The prototypical version of the high-end node that will be developed within CONTREX is based on a system-on-chip specifically intended for image processing captured by a low-power CMOS camera. The architecture of the SeCSoC (shown in ) extends the typical structure of a high-end microcontroller with specific modules for image processing, ultra-low power analog modules, and power island and clock gating capabilities. The SoC is based on the multi-core R4MP processor.

The board that will be provided has a size similar to a credit card, and it includes two VGA sensors, optional sensors connectors, USB, SPI, UART, JTAG host connectivity, digital stereo microphones and pressure and temperature sensors. The adoption of multi-core technology also in some of the sensing units will allow for an improved management of extra-functional aspects such as power consumption, execution time, quality of service, security and reliability. The proprietary cores can be programmed through a development toolchain (gcc+GNU binutils based) released by ST. Moreover peripheral library APIs are available.
The board that will be provided has a size similar to a credit card, and it includes two VGA sensors, optional sensors connectors, USB, SPI, UART, JTAG host connectivity, digital stereo microphones and pressure and temperature sensors. The adoption of multi-core technology also in some of the sensing units will allow for an improved management of extra-functional aspects such as power consumption, execution time, quality of service, security and...
reliability. The proprietary cores can be programmed through a development toolchain (gcc+GNU binutils based) released by ST. Moreover peripheral library APIs are available.

Due to the complexity of the image processing algorithm, within the project timeframe only a prototypical implementation will be realized, namely the identification of the number of occupants, provided that a crash has happened and has been recognized by the low-cost sensor node. This entails the following sequence of operation:

1. The low-cost sensor node identifies a crash event.
2. The low-cost sensor node communicates to the main Cobra ECU that a crash has been detected. This communication needs to be performed when the crash event has been completely managed by the sensor node.
3. The main Cobra ECU wakes-up the high-end node.
4. The high-end node takes one or more pictures of the inner of the vehicle and executes the image processing algorithms, estimating the number of occupants.
5. The high-end node communicates the number of occupants to the main Cobra ECU

The algorithms – yet to be studied and defined in detail – will exploit background/foreground separation based on a reference image of the empty vehicle. It is worth noting that the background (i.e.) the image of the empty car, can only be taken when the car is recognized as being at rest and empty. This may require additional algorithms to be executed on the low-cost sensor node and resource run-time management on the high-end node. In the prototype solution – mainly devoted to experiment a proof-of-concept – that will be implemented the assumption of having the background image available will be done.

### 3.2.3 Scenario 3: Key-off services

Insurance companies suffer from high costs related to accidents and acts of vandalism that happen while the car is parked and unattended. A major requirement of insurance companies to Cobra and other competitors concerns the ability of detecting – and possibly classifying – the widest possible range of such events. This will reduce management costs of real damages and will contribute to prevent frauds.

At present the system deployed by Cobra suffers from the following problems and limitations:

- No or incomplete data is available if a **minor crash** occurs when the car is key-off. Currently, in fact, the sensor node is in a deep sleep mode of operation with the microcontroller waiting for an interrupt to be awakened. The sensor is configured with a simple, threshold-based criterion to detect shocks and awaken the system. When the shock is modest, it might not be sufficient to awaken the sensor node, but, on the other hand, if the threshold is configured to a lower value such to detect low-energy events, the simple processing algorithm on the node will flood the server with a large amount of crash snapshots, most of them are likely to be false positives. Even in the best case, no data can be available in the 4s before the crash, as expected by the servers.

- Other **acts of vandalism** are not detectable by measuring accelerations only. This is mainly caused by the low energies associated to such events. To cope with this problem, additional sensors are required, such as magnetic field sensors or microphone arrays. Though critical for Cobra, the introduction of new sensor on the low-cost
sensing node and the development of suitable algorithms is out of the scope of this project.

- **Long-time parking** (e.g. two weeks or more) would lead to either the car battery be discharged or the main vehicle ECU intervention to disconnect the Cobra black-box system. This requires very strict power consumption constraints and a run-time mechanism that autonomously enables/disables the different functionality of the sensor node.

The following requirements emerge from the analysis of the problems:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Importance</th>
<th>Impacts on</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low power consumption</td>
<td>H</td>
<td>Battery duration.</td>
<td>For long-term parking</td>
</tr>
<tr>
<td>Incomplete acceleration and position profiles shall be managed properly</td>
<td>M</td>
<td>Quality of the crash reconstruction</td>
<td>Accepted degradation of performance with loss of initial part of the crash</td>
</tr>
<tr>
<td>Specific set of key off services at device level shall be available</td>
<td>M</td>
<td>Cost, power consumption</td>
<td>Depends on agreements with customers. Some specific for this scenario, some capability will be lost</td>
</tr>
</tbody>
</table>

The (partial) solution that will be implemented within CONTREX will only support the detection and, possibly, the classification of low-energy events at key-off, such as the vehicle being touched by another car while parking.

To this purpose, ad-hoc algorithms will be developed, their impact on the power consumption evaluated and their implementation optimized to meet the requirements. Alternation between high-energy events and self-calibration on one hand and low-energy algorithms on the other will be managed autonomously by a dedicated software layer.

### 3.2.4 Scenario 4: B2B services

A valuable asset of Cobra is the extremely large amount of data that is collected from hundreds of thousand vehicles circulating in very different environmental conditions. This constitutes an interesting business opportunity to create value-added services constructed on the analysis of such a large data base.

The main potential services that emerged from an analysis of this scenario are the following:

- **Data mining.** The key idea is to exploit the large amount of data collected by the Cobra control room to extract aggregated information of interest to different organizations or companies.

- **Value-added services.** Provided exploiting M2M interoperability and third-party source of data, such as traffic information, weather conditions, etc.

Beside being an opportunity, this large amount of data and processing that is required on the Cobra data centre is more and more becoming critical, strongly pushing towards scalable solutions. The current implementation of the data centre is based on a cluster of servers that provide several different services, namely:
- **Data collection.** Several telecommunication interfaces are available to collect data from the devices on the field. This is the entry point of data in the processing and storage chain.

- **Data storage.** All the personal information about the customers and their data collected from the field is stored in suitable non-relational databases both for activity logging and for further processing.

- **Data processing.** Whenever a crash snapshot is received, the accelerations, speeds and positions recorded by the black-box are passed to a set of algorithms for analysis. The current implementation is basically a set of Java applications executed when suitable triggers on the database are activated.

At present, no B2B services are implemented, as Cobra has only recently started to concretely consider this opportunity.

As anticipated in the discussion of Scenario 2, a main issue that will impact on the server-side infrastructure of all automotive telematics providers concerns black-box portability. This problem must clearly be afforded considering the entire process and infrastructure, from sensor node to the server-side processing. A first immediate consequence is that crash snapshot processing must be decoupled from the retrieval and storage infrastructure and provided to third parties as a service, exploiting cloud technologies. At the same time, moving crash processing from the Cobra servers to the cloud will guarantee seamless scalability to support the rapidly increasing number of devices installed. Summarizing, the high-level requirements collected in the table below have emerged.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Importance</th>
<th>Impacts on</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data security and privacy</td>
<td>A</td>
<td>On-device memory, transmission protocol, server (database)</td>
<td>Data mining on data should not disclose private data.</td>
</tr>
<tr>
<td>Long term storage</td>
<td>M</td>
<td>Server infrastructure</td>
<td>Anonymous storage of data for statistics and for the customers</td>
</tr>
<tr>
<td>Data set, format and protocol standardization</td>
<td>NTH</td>
<td>Interoperability among black boxes</td>
<td>Should be the output of an association of telematics</td>
</tr>
<tr>
<td>Scalability</td>
<td>NTH</td>
<td>Management of server</td>
<td>Opens new B2B opportunities</td>
</tr>
</tbody>
</table>

The implementation of such a complex scenario goes well beyond the scope of the CONTREX project, nevertheless a first step in this direction will be done in this use-case, namely:

- Adoption of a **standard protocol** and a cloud framework for crash data collection.

- Implementation of crash processing as **service on a cloud infrastructure**.

The current implementation of the overall infrastructure is shown in Figure 3.12.
To migrate this architecture to a first experimental cloud-based implementation the following changes need to be performed:

- On the main Cobra ECU the communication layer will be replaced with the embedded portion of the Kura pervasive framework. Note that memory constraints on the main Cobra ECU may not allow integrating the Kura framework. In this case, and for demonstration purposes only, an embedded gateway provided by Eurotech will be used.

- The proprietary protocol will be replaced by the standard MQTT protocol.

- The server-side crash algorithms will be ported to the cloud infrastructure and delivered to the Cobra control room personnel as a service.

This leads to the new cloud-based application architecture shown in Figure 3.13.
This experimental set-up, though, cannot be considered a final industrial solution, since, as mentioned before, private and sensitive data must be treated extremely carefully in respect with the current national and international regulations. A potential evolution, though, is possible by decoupling the storage of personal data on the cobra servers, associated with suitable keys and the storage of crash data on the cloud, also linked to the aforementioned keys. This would allow performing anonymous processing on the cloud, being Cobra the only player that can associate customers to data. Such an architecture is though far more complex and is out of the scope of the project.

The development of this scenario is based on the cloud infrastructure and services provided by Eurotech and described in the following. It is worth noting that the embedded platform provided by Eurotech as in-field gateway has not been designed for the automotive field and does not satisfy the many requirements typical of this context.

On the other hand, from a more general point of view this scenario follows the philosophy and the technological approach of the "Internet of Things" (IoT). In the vision of IoT, the embedded systems are smart objects that communicate with each other and, at the same time, with business applications hosted in data centers or in the cloud. In this scenario, we plan to develop the elements needed to build a distributed system, in which devices and sensors are interconnected with the traditional IT infrastructure. From this point of view, the cloud infrastructure can be considered as a M2M Integration platform, because it covers the role of integration between the M2M devices and the IT.

The M2M Integration platform is based on two main components: an embedded pervasive framework (Kura) and a Cloud platform.

The Kura pervasive framework is an open source initiative aimed to provide a standard solution for deploying and configuring hundreds or thousands of embedded devices. Kura has been conceived as a software framework for the gateway of the Internet of Things. The framework offers a hardware abstraction layer and OSGi-based application development environment that simplifies the deployment of the embedded devices installed in the vehicle and the development of the business logic specific of this use case.

The cloud platform is a software-as-a-service specifically designed to target M2M applications: it is a horizontal technology that we apply in this scenario to the automotive and driving assistance contexts. The platform provides the services needed to collect data from the field and to integrate them into downstream applications, business processes, dashboards and reports. It also provides all the services required for the management of the M2M devices on the field including configuration management, application life-cycle management and remote access.

The **Kura pervasive framework** offers a platform that can live at the boundary between the private device network and the local network, public Internet or cellular network. In this context it provides a manageable and intelligent gateway capable of running applications that can harvest locally gathered information and deliver it reliably to the cloud.

Kura is an open source platform that is contributed to Eclipse by Eurotech. Eurotech developed the original technology to run on a large set of devices: from general purpose devices, rugged mobile computers, wearable devices, service gateways and vehicle consoles, all the way down to the Raspberry Pi. It is implemented as a Java-based platform, can be installed on Linux based devices and provides:
- A remotely manageable system, complete with all the core services applications need
- An abstraction layer for accessing the device’s own hardware.

Kura introduces a standard framework based on OSGi for handling events, packaging code and a range of standard services. An application in Kura is delivered as an OSGi module and it runs within the container along with the other components of Kura (see Figure 3.14). The framework provides a store-and-forward repository service conceived to take the information gathered from the locally attached devices and send that data onwards to one or more brokers and other cloud services. The communications are based on MQTT protocol.

![Kura Application](image)

**Figure 3.14 Software stack of the pervasive framework**

Applications, that encapsulate the pervasive part of the business logic of the automotive scenario, can be remotely deployed as OSGi bundles and their configuration imported (or exported) through a snapshot service. The framework will offer some useful service (in the form of OSGi bundles) like:

- A GPS location service that helps to geo-locate the device
- A time service to ensure good time synchronization
- A database service for local storage using a embedded SQL database
- A process and watchdog services to keep things running smoothly
- Communication services
- A set of services to collect raw data, abstract event and information from the sensing unit installed in the vehicle

Nevertheless, the framework (see Figure 3.15) is designed to simplify the development activities and is fully integrated with Eclipse IDE. The HW abstraction layer hides the complexity of the hardware, of the firmware and of the operating system. Furthermore, abstracting this complexity through the OSGi framework provides to developers a simple and unified solution that allows an optimal exploitation of the hardware capabilities without losing the focus on the business logic. With this approach Kura will offer OSGi services for Serial, USB, Bluetooth and a portable access to a wide range of common devices, which can still be used programming with Java’s own APIs. Finally, a specific API for devices attached via
GPIO, I2C, PWM or SPI will allow a system integrator to incorporate custom hardware as part of their gateway.

In the present use-case we will evaluate the possibility to integrate only the strictly necessary portions of this architecture (see Figure 3.16) to implement communication towards the cloud and thus allow providing data collection and crash processing as a service.

![Figure 3.15 Pervasive framework functional overview](image)

![Figure 3.16 Subset of the pervasive framework used in the B2B scenario](image)
The cloud technology is the key to provide the services needed to collect data from the vehicles and to integrate them into downstream applications, business processes, dashboards and reports, including Cobra’s existing IT infrastructure. In potential evolution of the present scenario the cloud platform will also support providing all the services required for the management of the devices installed on the vehicles including configuration management, application life-cycle management and remote access.

Figure 3.17 shows the architecture of the cloud platform and its main functional components.

Figure 3.17 Cloud platform architecture

The cloud platform is composed by seven main functional units:

- A device connectivity unit
- A security layer
- A device management unit (not needed for the demonstration)
- A data management unit
- An account management layer (not needed for the demonstration)
- A unit for device configuration administration (not needed for the demonstration)
- A layer for application integration support

These components will be developed in a Linux environment and will rely on Amazon EC2 infrastructure as a service. The choice of Amazon Web Services is justified by the practical requirements of this use case. Amazon Web Services provides a scalable cloud computing infrastructure and its data centers are certified with industry-recognized safety certifications such as PCI DSS Level 1, ISO 27001, FISMA Moderate, HIPAA, and SSAE 16. In addition,
data centers have different physical and operational measures to ensure the protection of the data. Amazon is currently present with its data centers in almost every continent.

The device connectivity unit will be responsible for the connectivity of the devices installed in the vehicle. The connectivity is maintained by a Message Broker, which follows the MQ Telemetry Transport (MQTT) protocol. MQTT is a protocol designed from the ground up for M2M applications. It is supported by several messaging systems including IBM MQ Series. It is currently in the process of standardization by the OASIS standards body under the supervision of Eurotech and IBM.

The MQTT protocol has been designed as a lightweight "publish/subscribe" messaging system for M2M applications. Its main benefits can be summarized as:

- **Optimized for M2M applications.** Only 2 bytes of overhead per packet and integrated management of "quality of service". Through a session-oriented connection to the broker, the communication latency is only limited by the available bandwidth.

- **Firewall friendly.** The installation of devices within corporate intranets doesn’t require opening additional incoming network ports since the connection is initiated by the device.

- **Publish/Subscribe messaging.** Message pattern to provide one-to-many message distribution and decoupling of message producers (Devices) from consumers (Applications).

- **Session awareness.** The system automatically generates events when a device disconnects abnormally and provides the ability to fully re-establish the session upon reconnection.

- **Security.** The connection is protected by SSL and authenticated with username and password.

The device management unit will offer a complete set of features to simplify the management of the devices in the vehicle. To avoid additional complexity and requirements on the device deployment and configuration, this unit will be layered as an application over the MQTT protocol.

Basic functionality such as sensor device configuration and diagnostic retrieval will only be implemented in the use case. In addition that specific functionality, a security layer provides to all the units of the cloud platform a centralized security foundation layer following Role Based Access Control (RBAC) model. User's identities can be defined and associated with one or more permission guaranteeing the principle of "least privilege". The devices installed in the vehicles and connected to the platform are identified using credentials of one of these user identities.

Finally and most importantly for the use case at hand, the application integration layer provides support for the integration with existing applications and, more generally, existing IT infrastructure. This layer will offers modern Web Services API based on Representational State Transfer (REST). The REST API expose all the platform functionality described above, including data management and device management. The REST API also offers a "bridge" to the MQTT broker allowing to route commands from applications to the devices without a specific connection to the Message Broker. Technologies such as REST/Comet are included.
allowing to display real-time data published by the devices in web pages and mobile dashboards.

3.3 Use-case at a glance

This picture in Figure 3.18 shows all the major improvements that will implemented within this use case. As it can be seen, the propose developments impact on the entire end-to-end application, from the sensor node(s) up to the server/cloud infrastructure.

![Figure 3.18 Use case 2 at a glance](image)

3.4 Requirements

From a preliminary analysis of the non-functional aspects involved in the development of the different functionality over the different subsystem, the following quantitative requisites have been identified. The table below indicates the subsystem involved and the scenario which it refers to or where it is derived from.

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Scenario</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low-cost sensor node</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Self-calibration algorithm run-time &lt; 1ms</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Improved crash recognition algorithm run-time &lt; 1ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average power consumption at key-on &lt; 80mW</td>
</tr>
<tr>
<td></td>
<td>1, 2</td>
<td>Sampling frequency &gt;= 250Hz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Acceleration range -16g/+16g</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Acceleration quantization &gt;= 12 bit</td>
</tr>
</tbody>
</table>
### 3.5 Mixed criticality issues

The overall system shows several different criticalities at all the architectural layers. In particular the following aspects have been identified.

- **Power consumption.** At all levels, but especially for in-field devices, power consumption is a critical issue. Sensor nodes and the main Cobra ECU are in fact expected to be permanently operating, that is both when the vehicle is used and when it is parked (key off). In addition to power optimization techniques applied to the processing elements of the subsystems, sophisticated power management techniques shall be adopted. Energy requirements of the cloud infrastructure, though less relevant from the functional point of view, they still need to be considered for economic and environmental reasons.

- **Size.** Sensor nodes, especially the low-cost one, should be as small as possible for a twofold reason: minimize the disturbance to the driver and simplify installation procedures.

- **Cost.** The overall cost can be split in three different contributions: cost of the services offered by the cloud infrastructure, cost of devices (nodes and main Cobra ECU) and the cost communication. This last contribution is particularly relevant and collects both the costs of “functional” data transmission (e.g. accelerations, images, GPS coordinates, etc.) and non-functional information transmission (e.g. control information, device configuration data, firmware/software updates, etc.). This cost is a direct consequence of the maintainability that should be guaranteed by the system.

- **Customizability.** As the main goal of the application considered in this use case is that of offering new services both to final users and to business stakeholders, customizability and extendibility are of great value. Though a certain degree of customization is also expected for sensor nodes, the focus is on the services provided both to end-users and other businesses. The cloud infrastructure and a software as a service approach is the key technology to enable customizability and extendibility.

- **Scalability.** As the potential customer for Cobra is expected to grow significantly thanks to the new and improved services offered, scalability on the server-side is of utmost importance. A shift from a dedicated server approach to a cloud infrastructure is the key choice to ease scaling of the processing and storage capability.
- **Performance.** The concept of performance can be regarded as assuming different meanings depending on the specific level of the architecture. At the lower levels, it can be measured by traditional metrics such as throughput and latency, while at the top levels of the application the concept of performance gradually fades into the more general idea of quality of service.

- **Installation.** The installation procedure of the in-vehicle devices (sensors and main Cobra ECU) is performed by non-specialized personnel. In order to reduce costs and minimize field malfunctioning, a simple installation procedure must be devised. This requires the device itself to provide self-calibration and self-configuration capabilities.

- **Maintainability.** This is especially critical for the sensor nodes and the main Cobra ECU, as they are deployed in several hundred thousand of units all over one or more countries. Configuration of the devices and firmware/software updates are managed by resorting to the GPRS/3G networks. Besides the technical aspects, the main problem related to maintainability is the economic cost of such operations.

- **Security.** Some of the information transmitted between the different subsystems are sensible and should be protected against unauthorized access. This poses particularly critical problems for the sensor nodes and the main ECU, as critical trade-offs must be identified to balance security with cost and power consumption.
4 Use-Case 3: Ethernet Over Radio System

4.1 Telecom Use Case: Overview

4.1.1 Extra-functional and Green computing challenges

Several extra-functional property requirements converge within modern telecom systems. In large Central Office installations, the overall footprint of equipment in terms of installation space is a critical factor, the overall weight of the installed equipment influences the eventual cost and the mission-critical public telecom service provision applications are nearly universally characterized by extremely high availability and reliability requirements. The Telecom system must take into account functional specifications and HW specific constraints while integrating the following extra-functionalities in the system:

- **Time:**
  - The system must dynamically react to the radio channel condition

- **Reliability:**
  - Considering that the system will be installed in areas characterized by limited accessibility and difficulties to provide spare parts, the system must have very high constraints in term of Mean Time Between Failures (MTBF);
  - QoS (e.g. IEEE 802.1q): the TLC system provides QoS capabilities to support TDM traffic and different classes of priority among the different kinds of non-TDM traffic

- **Temperature:**
  - The system must be able to work under some temperature constraints (-35°C ÷ +60°C);
  - BER, channel quality, and power transmission depend on the run-time environment temperature changes.

- **Power, Safety and EMC:**
  - The system must be compliant with CE directives concerning radiated emission and Electromagnetic Compatibility (CE Certification Test according to 1999/05/EC R&TTE Directive, EN 60950-1 (2006-04), EN 60215 (1989-01), EN 60825-1 (1994-03), EN 50385 (2002-08))

The assurance of thermal and power properties is challenging in several ways; from the environment point of view, because components are often characterized by outdoor placement, or by the electronic circuits themselves due to the fact that the clock rate is heavily influenced by the technology used for CPU, busses, and FPGA components.
Due to its physical scale, the telecommunications use case also exhibits challenges in terms of overheating the external environment, due to the potential scale of the components (especially in large central installations).

In this sense, this use case fully represents the problem of “green computing” from two perspectives:

- minimizing internal power consumption;
- dealing with thermal dissipation into the external environment.

The provision of facilities for heat dissipation (e.g. sinks, fans) is associated with high extra costs to installations.

### 4.1.2 International standards

Public-oriented telecommunications facilities are regularly classified as critical infrastructures (defined in European Directive 114/08 EC as those assets, systems or parts thereof located in the EU Member States which are essential for the maintenance of vital societal functions).

As such, they are also subject to emerging certification procedures such as those being elaborated within the **European Programme for Critical Infrastructure Protection** (Figure 4-1).

![Figure 4-1: The European Programme for Critical Infrastructure Protection (EPCIP)](image)

### 4.1.3 Mixed-criticality issues

The facility is mixed-criticality by its nature, with business data applications sharing resources with applications such as voice. The business data applications are considered mission-critical, with security and confidentiality issues as well as requirements on high integrity, with high associated economic losses in cases of malfunction. The voice applications are generally non-critical in the sense that performance degradation and malfunction is generally only associated with temporary decrease in customer satisfaction without high economic impact. (Note that voice applications with high integrity and security requirements also exist, but they are not considered in this specific use case.) Figure 4-2 illustrates the major components of the use case in their relation to mixed criticality. The system exhibits high SW ⇔ HW ⇔
FPGA interoperability and must execute several tasks while guaranteeing their differing requirements for real-time operations. Whereas in embedded systems, the provision of security usually has a safety-related dimension, here there is primarily a dimension of confidentiality, as noted above in the discussion of business data applications.

4.2 State of the Art

Radio relay systems have unique competitive features, such as quick deployment and fast network roll-out with simple civil works, that strongly justify a modern telecommunications network scenario in which radio systems and fiber optic systems will complement and support each other in a very effective mixed media approach.

The wireless family is made of a variety of systems for different applications: i.e. base stations for mobile networks (GSM, UMTS, GPRS), terminals for nomadic usage (Wi-Fi and Wi-Max), terminals for unidirectional broadcasting (TV and DVB), bidirectional Point-to-Point and Point-to-Multipoint systems for fixed networks.

Within the CONTREX project Intecs Telecom focuses its attention on the Point-to-Point (P2P) Ethernet over Radio Microwave Wireless System (Figure 4-3).
The Ethernet over Radio System is specifically designed and engineered for such situations where it is required to transport E1 signals. It allows a smooth transition from the previous generation of transport (PDH) networks, encapsulating the E1 signal into an Ethernet frame.

The Ethernet over Radio System is particularly suited to cover mobile broadband infrastructure data growth from GSM to WDCMA to LTE and many other needs of high data transport.

In other words, it provides smooth migration path from legacy to modern systems, an essential capability in the telecom area (Figure 4-4).

Figure 4-4: Ethernet over Radio System migration path

The Ethernet over Radio System mainly consists of an OutDoor Unit (Figure 4-5), the ODU-IP-LC (also abbreviated as ODU or ODU-IP). The ODU Unit encapsulates Ethernet packets in a GFP frame, modulates, and sends them on the radio channel. Management packets are redirected to a Controller Unit.

Figure 4-5: The OutDoor Unit (ODU)
The ODU implements all functionalities required by the system, in particular:

- Signal base band processing
- Modem stage
- RF interface
- User Ethernet interface

### 4.2.1 Ethernet over Radio System main features

The main features of the Ethernet over Radio System are listed below:

- ODU board partitioning (digital section with modem, analog section with RF circuitry)
- User interface: Ethernet with proprietary Power over Ethernet (PoE)
- Traffic Capacity: from 11.5 to 354.7 Mbit/s
- Element management by Local Craft Terminal (Web based) and by remote OS (SNMP)
- Single antenna. Direct connection to the antenna (by means a dedicated socket on the board)
- Modulation QAM - Quadrature Amplitude Modulation

### 4.2.2 Hardware architecture

The ODU Card houses a Freescale communication processor based on the PowerPC core (MPC880) with plenty of RAM (64MB) and FLASH (32MB), two Ethernet interfaces and I2C/SPI bus support. It is capable of running a pre-emptive real time operating system like Linux (xenomai) and a full featured network stack that allows the developer to access a pool of ready to run software applications. In particular the board can run an SNMP agent, a WEB server and different processes to control the devices forming the system. The operating system is capable of managing hard real time events.

The full version of the ODU Hardware architecture is depicted in Figure 4-6.
In Figure 4-7 is shown a simplified version of the ODU HW architecture, with the main interfaces between the logical blocks.

The ODU board Hardware details are listed below:
- **Freescale MPC880 PowerQUICC**
  The control section allows controlling the ODU card by the functionalities shown in Figure 4-8:

Figure 4-8: ODU Control Architecture block diagram

The main functional control blocks are:
- Microprocessor (uP), it manages the ODU card, handles alarms, handles the protection switch protocol and communicates with a remote Element Manager, a Local Craft Terminal and the ODU partners (local and remote);
- Memory and Peripherals, e.g. SDRAM, Flash EPROM (serial and parallel), FPGA and other devices used for management of RF channel.

- **LATTICESEMI LFE2M35E-5F256C FPGA (Ethernet Layer 2 Switch + MODEM functionalities)**
  The Switch performs the following functions:
  - Routing of LCT controller messages based on MAC address and optionally on VLAN tag.
  - Routing of CT traffic from/to Cable Interface to/from INT A of Modem, based on proprietary VLAN tag. No buffering (unless the one required for the store&forward mechanism) and flow control is applied to this traffic between Switch and Modem. When the radio channel is not available the traffic is dropped.
  - Routing of DUT (Data User Traffic) from/to Cable Interface to/from INT B of Modem, based on MAC address and optionally on VLAN tag. This traffic to the radio link can be served by four priority queues to implement QoS, based on IEEE 802.1p, IP TOS&DS, VLAN ID and MAC address. Because of the limited capacity available on the radio link, on INT B a PAUSE Flow Control is applied to stop forwarding frames from Switch to Modem when the Modem Tx buffer is full.
The incoming messages from the Cable interface are buffered in the Switch priority queues according to VLAN tag. The queues are emptied according to their priority. When the queues are full the following messages are discarded.

- Routing of Management information from/to System controller to/from INT B of Modem (radio channel), based on MAC address and optionally on VLAN tag.

- **Memories:**
  - 32-bit-wide 64 Mbyte DDR3 SDRAM for data handling and program execution
  - 32 MByte NOR Flash memory for SW code and FPGA netlist
  - 1MB Serial Flash EPROM for station data

- **Parallel and serial busses to/from memory and peripheral devices:**
  - Microprocessor local bus at 66 MHz to demultiplex address, data and control lines required to access SDRAM, Flash EPROM, FPGA and other devices
  - SPI bus required to access the external serial Flash EPROM for station inventory data and some RF section’s devices.
  - I2C bus required to access some devices used into the RF section.

- **The Serial Management Controller** is used as a Debug Serial Interface for HW and SW debugging purposes.

### 4.2.3 Software architecture

The software running on the ODU card can be partitioned into:

- **Basic software services to the Application software including a layer to access hardware devices.** This task is accomplished by the Linux operating system and a suite of software drivers for the different busses and devices housed on the board.

- **Communication services via a full blown TCP/IP stack.** The TCP-IP stack is provided by the Linux operating system.

- **SNMP stack to handle SET and GET requests coming from an OS,** this task is accomplished by the NET-SNMP agent. Furthermore, the agent is capable of forwarding requests to an application process via an AgentX connection.

- **Web server and an application gateway to manage HTTP connections coming from an external LCT and translate HTTP requests into SNMP GET and SET to be handled by the SNMP agent.**

- **Management and configuration of the FPGA and the RF devices housed on the card.** In particular, the driver to manage the FPGA should be able to manage an interrupt request from the FPGA.

The software running on the ODU will be stored in two different portions of the flash in order to allow a safe upgrade of the software. The software will also contain the netlist of the FPGA and the system will be able to download the netlist without any user intervention. The ODU board Software details are listed below:
- Linux-2.6.19.2-xenomai
- U-boot (bootloader) and ELDK toolchain from Denx (available at http://www.denx.de/wiki/DULG/ELDK)
- Linux Device Drivers for the different busses and devices housed on the board (parallel Bus, SPI, I2C, MII)
- Kernel Threads through Linux work queues mechanism to manage critical tasks:
  - Automatic Transmitted Power Control (ATPC) management
  - Received Signal Strength Indication control
  - Power monitoring
  - Temperature updating

The ATPC function allows controlling the output level of the microwaves amplifier, which can be selected between a maximum and a minimum value according to the knowledge about power at the receiver side (obtained through feedback from the mate ODU). In this way the amplifier works for a high percentage of time with a low output power values and reaches the higher levels only in unfavourable conditions of propagation. The ODU Software architecture is depicted in Figure 4-9.

![Figure 4-9: ODU Software Architecture](image)

### 4.2.4 Ethernet over Radio System: Thermal and Power Analysis

As described earlier, the Ethernet over Radio system must be able to work under some temperature constraints (-35°C – +60°C) because BER, channel quality, and power transmission depend on the run-time environment temperature changes.

That is why during the design phase some thermal analyses are performed in order to keep the temperature of the components under control while the system is working.
Currently, the FloTHERM tool is used in the Intecs laboratories in order to perform thermal analysis. As shown in Figure 4-10, FloTHERM needs some inputs including the geometry of the system, the number of components, and the power dissipation of each component, normally retrieved from the datasheet and estimated at the maximum level in order to obtain conservative measurements.

Concerning power analysis, currently Intecs laboratories do not have any tools or methodologies. Rather, the maximum power consumption of components is retrieved from datasheets and used as input for the FloTHERM tool.

The current implementation of the Ethernet Over Radio System is based on a Linux environment running on a PowerPC microprocessor. Currently the different tasks are running on the same chip and therefore the usual system tasks are disturbing, in terms of CPU load and scheduling, the real time tasks for the management of the radio link.

The Power and Temperature original constraints defined during the design phase are described below (please refer to Section 4.2.2 for details).

**Power and Temperature original constraints:**

- **ODU System Controller Block Diagram**

Ethernet Over Radio houses a Freescale communication processor based on the PowerPC core (MPC880) with plenty of RAM (16 to 32MB) and FLASH (8 to 32MB), two Ethernet interfaces and I2C/SPI bus support.

The original power constraints for the System Controller are shown in Table 4-1:

<table>
<thead>
<tr>
<th>V [ Volt ]</th>
<th>3.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Controller</td>
<td>Current [ Ampere ]</td>
</tr>
<tr>
<td></td>
<td>&lt;=0,5</td>
</tr>
</tbody>
</table>

Table 4-1: System Controller original power constraints
Unfortunately, during the test phase, the values measured didn’t respect the original assumptions, as shown in Table 4-2:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODU - Temperature range for measurements</td>
<td>[-35°C ; +60°C]</td>
<td>Ambient temperature - 35°C to +55°C (still air). +5°C have been accounted for irradiation extra temperature.</td>
</tr>
</tbody>
</table>

Table 4-3: System Controller original temperature constraints

4.3 Intecs CONTREX expected goals and Definition of New possible Scenarios

4.3.1 Intecs CONTREX expected goals

The Intecs motivations behind the exploitation of CONTREX tools and methodologies and the expected goals are listed below:

1. *More services integration, increasing the quality of services by exploiting the Zynq dual core*

   Currently different tasks (with different criticality level) of the Telecom application are executed on the same processor with the risk that less critical tasks affect the performance of higher critical ones. The adoption of a multicore platform such as Zynq will allow the partition of tasks over different cores with a higher service integration and QoS as a consequence.

2. *Intecs Design Process innovation*

   A System Modeling Layer should be introduced as a new step in the Intecs design flow to formally model task execution and interaction; this step is essential to drive allocation of tasks among different cores as described in the previous item.
3. **New Microcontroller (Zynq) Modelling and Virtualization + FPGA Power Estimation**

Intecs expects to have a reliable simulation platform in order to:

- simulate the HW platform
- test SW execution
- perform test on power consumption and temperature range of work

The simulation platform should reproduce also a full and realistic network scenario since the behaviour of most tasks is triggered by network behaviour.

4. **Introduction of a power and thermal analysis environment to improve the power and thermal estimation accuracy and validation of the original terms of assumption**

As described in Section 4.2.4, during the design phase we had some power and thermal assumptions. Some of them are respected by the current implementation, some others are out of range. In any case, we expect to introduce a new power and thermal analysis environment in our flow in order to have an earlier validation of the system with respect to these constraints and, at the same time, we expect that the CONTREX tools and methodologies, applied to the new microcontroller (Zynq), will validate the original terms of assumptions.

5. **Reduce the Power Microcontroller dissipation (~20%) in order to achieve the original specifications**

As described in Section 4.2.4, for the System Controller we had some original power constraints that have not been respected, so we would like to exploit the new microcontroller (Zynq) and the new power analysis environment in order to reduce the system controller power consumption by about 20% by accomplishing the original specifications.

### 4.3.2 Analysis of possible new Scenarios and Test Bench

On the basis of considerations made in previous Sections, Intecs would like to exploit CONTREX tools and methodologies in order to implement the following new scenarios.

1. **FIRST SCENARIO: Porting of current SW to ARM and Power measurements on System Controller & FPGA Power estimation**

   The existing design (refer to Figure 4-7) will be ported to the Xilinx Zynq platform (the SW part is ported to the ARM dual core system running Linux in SMP mode). This first scenario is shown in Figure 4-11.
As you can see, the first scenario consists of the following actions:

- The current SW application (currently running on a Motorola MPC880) will be ported to ARM, running on Linux in Symmetric multiprocessing (SMP) mode.

- The Modem and the Ethernet Switch Layer 2 functionalities, currently implemented in an LATTICESEMI LFE2M35E-5F256C FPGA will not be ported to the FPGA inside the Xilinx Zynq platform. Here we are interested on the power consumption estimation only, so, the idea is to synthesize the available VHDL parts of the current FPGA for the Xilinx Zynq and use the resulting...
LUT, BRAM, DSP count to estimate the power consumption (using Xilinx Power Estimation Excel sheet).

- The test bench of this first scenario would be the evaluation of the power consumption of the ported software measured on the Zynq platform and its comparison against the power consumption of the existing system (see Table 4-1 and Table 4-2), taking into account that the FPGA part of the current system will not be implemented in Zynq but will be evaluated just in terms of power consumption.

Since the first scenario might not beat the existing system in terms of total power consumption (because the Xilinx Zynq platform might not be fully utilized) a second conceptual case study will evaluated in next section.

2. SECOND (CONCEPTUAL) SCENARIO: Integration of multiple Ethernet over Radio channels in a single SoC (Xilinx Zynq)

A second conceptual case study (depicted in Figure 4-12) could be the integration of multiple Ethernet over Radio channels in a single SoC (Xilinx Zynq).

![Diagram](image.png)

Figure 4-12: Second conceptual case study: Integration of multiple Ethernet over Radio channels in a single SoC (Xilinx Zynq)

This second scenario shall fully utilize the Xilinx Zynq platform resources, showing that the possible increased power consumption is more than compensated by the enhanced transmission functionality (support of multiple modems).

The total power consumption of the 2nd conceptual multi modem on chip design can be evaluated based on the estimates obtained from the 1st scenario.
In the next sections the Telecom domain-specific requirements are described, on the basis of the considerations of this chapter.

4.4 Requirements on Ethernet over Radio System

4.4.1 Requirements for Ethernet Over Radio System Software porting

To test the functional and extra-functional behaviour of the system under development, a virtual prototype offering a simulation environment on the host processor should be used so that the need of a real hardware prototype can be delayed to later design phases. This allows an early and low cost evaluation of the system’s behaviour and the exploration of different hardware architecture configurations.

To maximize synergies with other EDALab activities, the Intecs application should be ported onto the Open Virtual Platform (OVP), which is a simulation model of the hardware design providing libraries of processor and behavioural models and APIs for building your own processors, peripherals and platforms.

The OVP tool should be configured at microprocessor level to simulate as accurately as possible the targeted hardware platform. Furthermore, the FPGA present in the Intecs system should be modelled in the virtual platform. This will be done by using the HIFSuite abstraction tool developed in COMPLEX. This tool is able to manipulate descriptions written in different languages (including VHDL) and to generate a SystemC abstraction of the FPGA itself.

Since the Intecs system contains adaptive algorithms that regulate the transmission according to the importance of transmitted data and to channel condition, it should be simulated in a full and realistic network scenario to test the behaviour of transmission control tasks as a function of time-varying transmission condition. It must be possible to perform sufficiently realistic simulation of critical tasks, at several levels: first, at packet level, then, optionally, at signal level, and finally, with a mapping of signal level results to packet level.

The main requirements related to the Ethernet Over Radio System Software porting on a new environment/platform are summarized in next table.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Applicability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexible HW platform (we would prefer and support the Zynq 7020 as it is used in UC1)</td>
<td>HW development</td>
</tr>
<tr>
<td>It must be possible to abstract the current Intecs UC3 application platform into SystemC/TLM in order to apply the EDALab tools</td>
<td>modelling methodology/tools</td>
</tr>
<tr>
<td>The Intecs application must be ported onto the Open Virtual Platform to maximize synergies with other EDALab initiatives</td>
<td>modelling methodology/tools</td>
</tr>
</tbody>
</table>
It must be possible to perform sufficiently realistic simulation of critical tasks, at several levels: first, at packet level, using SCNSL; second (optional), at signal level using SystemC-AMS; finally, with a mapping of signal level results to packet level.

CONTREX shall provide to Intecs lab the Zynq environment (tool chain, kernel,…)

CONTREX shall provide to Intecs lab the Zynq platform abstraction within OVP simulation environment (tool chain, kernel,…)

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Applicability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A subset of the application functionalities must be representable by a concurrent process network diagram, with no shared state or side effects.</td>
<td>modelling methodology/tools</td>
</tr>
</tbody>
</table>
For design space exploration purposes, design constraints and performance parameters must be available, such as WCET, maximum memory size, and the like

An investment of Intecs in learning to use the ForSyDe tool and methodology is required

CONTREX shall provide to Intecs lab a System Design methodology with the objective to move Telecom system design (i.e. System on Chip, Hardware and Software telecom systems) to a higher level of abstraction

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Applicability</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTREX shall provide to Intecs lab a power analysis environment to organize and trace power related data to ensure getting consistent power estimates</td>
<td>Simulation/Analysis tools.</td>
</tr>
<tr>
<td>CONTREX shall provide to Intecs lab a power analysis environment to make it easier to collect, reuse and share power related data</td>
<td>Simulation/Analysis tools.</td>
</tr>
<tr>
<td>CONTREX shall provide to Intecs lab a power and thermal analysis environment to develop a holistic (covering all component types) and scalable (hierarchical) approach</td>
<td>Simulation/Analysis tools.</td>
</tr>
<tr>
<td>CONTREX shall provide to Intecs lab a power and thermal analysis environment to make power estimation dependent on additional parameters/metrics (eg activity), i.e. not systematically rely on maximum power</td>
<td>Simulation/Analysis tools.</td>
</tr>
<tr>
<td>CONTREX shall provide to Intecs lab a power and thermal analysis environment to improve power and thermal estimation accuracy</td>
<td>Simulation/Analysis tools.</td>
</tr>
</tbody>
</table>

Table 4-5: Requirements for the introduction of a System Modelling Layer

4.4.3 Requirements for a power analysis environment

Due to the fact that power and thermal analysis are key issues for the Telecom system reliability, in CONTREX we would like to investigate some new tools and methodologies in order to improve the current Intecs methodology and power estimation accuracy.

The main requirements related to the introduction of a power analysis environment in the current Intecs Telecom design flow are listed in the next table.
CONTREX shall reduce the Power Microcontroller dissipation (~20%) in order to achieve the original specifications

CONTREX shall validate the power and thermal original assumptions

CONTREX shall provide the Xilinx Power Estimation Excel sheet in order to make the FPGA power consumption estimation

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTREX shall reduce the Power Microcontroller dissipation (~20%) in order to achieve the original specifications</td>
<td>Simulation/Analysis tools.</td>
</tr>
<tr>
<td>CONTREX shall validate the power and thermal original assumptions</td>
<td>Simulation/Analysis tools.</td>
</tr>
<tr>
<td>CONTREX shall provide the Xilinx Power Estimation Excel sheet in order to make the FPGA power consumption estimation</td>
<td>Simulation/Analysis tools.</td>
</tr>
</tbody>
</table>

Table 4-6: Requirements for the introduction of a power and thermal analysis environment
5 References


Definition of industrial use-cases


A. STM32F103REY6

The INEMO-M1 STM32F103REY6 microcontroller has an ARM Cortex™-M3 processor core, the latest generation of ARM processors for embedded systems. With its 72 MHz maximum frequency, a 512-Kbyte embedded Flash and a 64-Kbyte SRAM accessed (read/write) at CPU clock speed with 0 wait states, it is suitable for storing programs and data.

Timers

The high-density STM32F103REY6 performance line microcontroller includes two advanced control timers (TIM1 and TIM8), four general-purpose timers (TIM2, TIM3, TIM4 and TIM5), two basic timers (TIM6 and TIM7), two watchdog timers and a SysTick timer. These timers are based on a 16-bit auto-reload up/down counter, a 16-bit prescaler, and can be configured for input capture/output compare functionalities, PWM or onepulse mode output.

SPI

The STM32F103REY6 microcontroller is equipped with an SPI interface able to communicate up to 18 Mbits/s in slave and master modes in full-duplex and simplex communication modes. The 3-bit pre-scaler gives 8 master mode frequencies and the frame is configurable to 8 bits or 16 bits. All SPIs can be served by the DMA controller. There are 2 SPI peripherals available on the INEMO-M1: SPI1 has been assigned for enabling an external communication channel with remote devices using the same peripheral, while SPI2 has been set for enabling the internal communication with the onboard digital gyroscope L3GD20.

I2C

Two I2C bus interfaces can operate in master and slave modes on the STM32F103REY6 microcontroller. They can support standard and fast modes. Both can be served by DMA and they support SMBus 2.0/PMBus. The INEMO-M1 exploits both I2C peripherals provided by the STM32F103REY6: I2C1 has been chosen for enabling the external communication and can be configured in two different sets of pins. On the other hand, I2C2 has been selected for enabling the internal communication with the onboard digital e-compass module.

USART

The STM32F103REY6 performance line family embeds three universal synchronous/asynchronous receiver transmitters (USART1, USART2 and USART3) and two universal asynchronous receiver transmitters (UART4 and UART5). The USART1 interface is able to communicate at speeds of up to 4.5 Mbit/s. The other available interfaces communicate at up to 2.25 Mbit/s. USART1, USART2 and USART3 also provide hardware management of the CTS and RTS signals. All interfaces can be served by the DMA controller except for UART5.

The INEMO-M1 module exploits the USART1 and the USART2 peripherals as in-application programming (IAP), application controller interfaces (ACI) or as simple communication standards.

CAN

The INEMO-M1 STM32F103REY6 is equipped with a CAN peripheral, compliant with specifications 2.0 A and B (active) with a bit rate up to 1 Mbit/s. It can receive and transmit
standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers and has three transmit mailboxes, two receive FIFOs with 3 stages and 14 scalable filter banks.

**USB**

The STM32F103xE performance line family embeds a USB device peripheral compatible with the USB full-speed 12Mbit/s.

**Analog to digital converter (ADC)**

The INEMO-M1 provides three 12-bit analog-to-digital converters embedded in the STM32F103xE performance line devices for a total of 16 available external channels, performing conversions in single-shot or scan modes. In scan mode, automatic conversion is performed on a selected group of analog inputs. The ADCs can be served by the DMA controller.

**Digital-to-analog converter (DAC)**

The STM32F103REY6 available in the INEMO-M1 embeds two 12-bit digital-to-analog output (DAC) converters. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data may be left- or right-aligned. The DAC has two output channels, each with its own converter. In dual DAC channel mode, conversions may be done independently or simultaneously when both channels are grouped together for synchronous update operation.

**DMA**

The two direct memory access (DMA) controllers have 12 channels in total (7 for DMA1 and 5 for DMA2), each dedicated to managing memory access requests from one or more peripherals. The DMA feature can be used with the main peripherals previously described: SPI, I2C, USART, general-purpose, basic and advanced-control timers TIMx, ADC and DAC.