Interner Bericht

DLR-IB-FT-BS-2021-263

Analysis of the UAVCAN standard with regard to the SORA process for UAS

Hochschulschrift

Thorben Bornscheuer

Deutsches Zentrum für Luft- und Raumfahrt

Institut für Flugsystemtechnik
Braunschweig

Deutsches Zentrum
DLR für Luft- und Raumfahrt

Institutsbericht
DLR-IB-FT-BS-2021-263

Analysis of the UAVCAN standard with regard to the SORA process for UAS

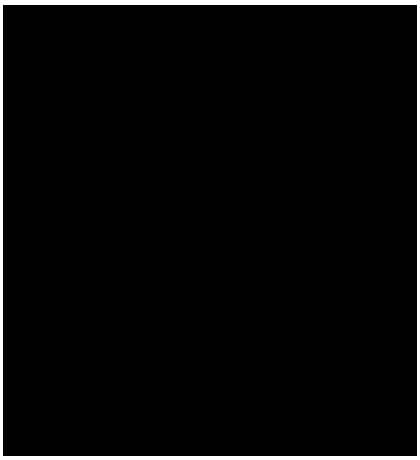
Thorben Bornscheuer

Institut für Flugsystemtechnik
Braunschweig

122 Seiten
17 Abbildungen
27 Tabellen
28 Referenzen

Deutsches Zentrum für Luft- und Raumfahrt e.V.
Institut für Flugsystemtechnik
Abteilung Unbemannte Luftfahrzeuge

Braunschweig, den 22.02.2024



Institutsleitung:	Prof. Dr.-Ing. S. Levedag
Abteilungsleitung:	Johann Dauer
Betreuer:in:	Dr. Sven Lorenz
Verfasser:in:	Thorben Bornscheuer

Technische Universität Braunschweig
INSTITUTE OF COMPUTER AND NETWORK ENGINEERING
DLR Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

Master's thesis

Analysis of the UAVCAN standard with regard to the SORA process for UAS

Thorben Bornscheuer

First examiner	:	Prof. Dr.-Ing. H. Michalik
Second examiner	:	Prof. Dr.-Ing. S. Levedag
Submitted on	:	19th July 2021

Statement of Authorship

Hiermit versichere ich, die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt zu haben. Die verwendete Literatur und sonstige Hilfsmittel sind vollständig angegeben.

English:
Hereby, I declare that I have composed the presented thesis independently and without without any help from others. The used literature and other aids are completely indicated.

[redacted]

Braunschweig, 19th July 2021

Abstract

This work analyses the UAVCAN standard in context of the SORA. The *certified* category for Unmanned Aircraft System (UAS) operation requires operators to perform a SORA. The risk of an operation defines the technical and operational requirements for the UAS. The increasing complexity of operations and avionic systems require for distributed avionic systems. This demand can be satisfied by communication standards based on shared mediums like a data bus. UAVCAN meets this low-level requirement and additionally defines high-level functions that improve the system design, system monitoring and configuration processes.

In this thesis the capabilities of the UAVCAN standard are matched to the requirements of a SORA for a flight test demonstrator operated by the DLR. The deduction of supporting standards for system safety assessment is described and system safety assessments are conducted to identify weak points within the current avionic system. In addition this work presents new concepts that benefit an operators system design process. These concepts include configuration management, system monitoring and redundancy concepts supported by the UAVCAN standard. Finally this work summarises the effects of an avionic architecture based on the UAVCAN standard on the SORA process.

- UAVCAN
- SORA
- UAS
- System Design

Contents

List of Figures

List of Tables

Acronyms

AGL height above ground level. 70, 72

AP Autopilot. 59, 61–64

APUS APUS. 88

ARC Air Risk Class. 46

ASTM American Society for Testing and Materials. 55, 66

AW Drones AW Drones. 2, 54–56, 77, 92

BMS Battery Management System. 86, 91

BVLOS Beyond visual line of sight. 2, 46, 47, 58, 61, 62, 66, 77, 88

CAN Controlled Area Network. 2, 12

City-ATM City Air Traffic Management. 58, 59, 62, 63, 67, 75, 78–83, 92

CMN Configuration Management Node. 85, 86

ConOps Concept of Operations. 5, 40, 43, 49, 53, 66, 71, 76

COTS Commercial of-the-shelf. 77, 79, 90

DAA Detect and Avoid. 46, 47, 61–63, 68–70, 75

DJI Da-Jiang Innovations Science and Technology Co., Ltd. 58, 63, 65, 66

DLR German Aerospace Center. 2, 57, 59, 84, 88

DRS Drone Rescue System. 59, 65, 67, 69, 72–74, 81

DSDL Data Structure Description Language. 18–20, 26, 30, 36, 37, 68, 71, 75, 85, 88, 90

DUT Device Under Test. 70, 71, 76

EASA European Union Aviation Safety Agency. 5, 54

ERP Emergency Response Plan. 44, 45

ESC Electronic Speed Controller. 58, 65, 73, 86

FC Flight Controller. 1, 65, 66, 68–70, 73, 75, 77, 84, 87–91

FHA Functional Hazard Analysis. 56, 67, 77, 79

FTA Failure Tree Analysis. 56, 57, 67–69, 79

GCS Ground Control Station. 9, 62, 64, 77

GCSO Ground Control Station Operator. 64

GNSS Global Navigation Satellite System. 66, 69, 70, 89

GRC Ground Risk Class. 43, 44, 46

GUI Graphical User Interface. 36, 37, 63

HAL Hardware Abstraction Layer. 9, 91

HITL Hardware in the loop. 70, 71, 74, 78, 82, 93

IMU Intertial Measurement Unit. 66, 69, 72

JARUS Joint Authorities for Rulemaking on Unmanned Systems. 40, 54

LAN Local Area Network. 34, 35

LOC Loss of Control. 39, 43, 45, 53, 84, 87

M600 Pro Matrice 600 Pro. XI, 58, 59, 61, 63–67, 69, 72, 77

MTOW Maximum Takeoff Weight. 65

MTU Maximum Transmission Unit. 21, 28

NAT Network Address Translation. 35

OSI model Open Systems Interconnection model. 20

OSO Operational Safety Objective. 40, 43, 48–55, 67, 68, 70, 76–79, 81, 82, 90, 92

PDB Power Distribution Board. 58, 59

PIC Pilot in Command. 63

PnP plug-and-play. 34

PWM Pulse Width Modulation. 65, 66

RC Remote Control. 63

SAA See and Avoid. 47

SAE Society of Automotive Engineers. 56

SAIL Specific Assurance and Integrity Level. 3, 38, 43, 47–49, 66, 71–73, 75–79, 81, 92

SI International System of Units. 27

SORA Specific Operational Risk Assessment. V, 1–3, 6, 38–41, 43, 46, 48, 53–57, 69, 79, 80, 83, 85–87, 89, 92, 93

SPF Single Point of Failure. 68, 69, 72, 80, 89–93

TAI International Atomic Time (temps atomique international). 32

TMPR Tactical Mitigation Performance Requirement. 46, 47, 66, 81, 93

UART Universal Asynchronous Receiver Transmitter. 62

UAS Unmanned Aircraft System. V, 1–3, 5, 6, 8, 9, 12, 31, 38–41, 43–46, 48–58, 61, 64, 65, 68–72, 74–80, 82, 84–87, 89–93

UAVCAN Uncomplicated Application-level Vehicular Computing And Networking. V, 1–3, 12–27, 29–37, 45, 47, 68–93

UTM Unmanned Traffic Management. 46, 50–52, 75, 81, 93

VLOS Visual line of sight. 46, 47

VPN Virtual Private Network. 62

1 Introduction

Manned aviation avionic architectures made the transition from star architectures to distributed systems over the last decades. Unmanned aircraft systems nowadays mostly use a star architecture for connecting the Flight Controller (FC) with sensors and actuators. The growing complexity of UAS and increasing safety requirements promote distributed architectures in context of UAS. As in manned aviation the key factor for distributed systems is a robust and reliable communication between components.

The Uncomplicated Application-level Vehicular Computing And Networking (UAVCAN) standard closes the gap of intra-vehicular communication solutions. The standard defines a whole ecosystem ranging from low-level message transfer up to high-level functions that simplify the system design, integration and monitoring during operation. Independent of the concrete message transport the standard provides durable interfaces for message definition and communication. The bandwidth of UAVCAN can be compared to the well-known *AUTOSAR* standard for automobiles. Originating from a hobbiest sector UAVCAN turns into a upcoming open-source industry standard.

New operational concepts for UAS reflect the growing range of use cases for UAS. In regulatory context UAS are classified in an open, specific and certified category. The open category contains hobbiest drones and low risk operations. The certified categories hold UAS designed, manufactured and operated according to standards. A UAS may exceed the limitations of the open categories but an expensive full type certification of a UAS may exaggerated the requirements for this specific operation. The specific category contains these *specific* UAS operations. To perform an operation within the specific categories, operators must conduct a Specific Operational Risk Assessment (SORA).

This thesis brings together the regulative background by the SORA and the intra-vehicular communication standard UAVCAN.

1.1 Problem definition

The SORA defines the boundaries for operations of UAS in the specific category. The risk-based assessment defines hard contstraints concerning the general - *intrinsic* - risk of the UAS operation. The overall risk of an operations lead to qualitative requirements for the operational procedures, UAS systems and UAS performance. As the requirements for higher risk operations increase, the costs, development effort and system complexity increase.

The UAVCAN standard enables complex systems by high-level and low-level functionalities. E.g. the low-level transport on a Controlled Area Network (CAN) bus allows for a deterministic message transport and interconnection of components with a limited number of interfaces and processing power (e.g. flight controllers).

As the UAVCAN standard is a relatively new standard, less information on compatible components reliability are availbale in comparison to well-known long-existing standards (used in manned aviation). Therefore a qualitative analysis is performed and deducted changes and located within the real world by a market research.

1.2 Thesis design

The presented thesis is seperated in a background chapter, an analysis on the previously *described problems*. The background chapter gives an overview about the UAVCAN standard and the SORA. Low-level details and high-level functions of UAVCAN are described in required detail for argumentation within the *anlysis*. The SORA chapter describes the existing regulations for operations within the specific category and explains terms and requirements which are the starting points for the analysis.

In a two-folded approach the UAVCAN standard is located within the SORA requirements. A top-down approach will present the chances and limitations following the requirements given by the SORA for a multicopter operated by the German Aerospace Center (DLR) under *Beyond visual line of sight (BVLOS)* conditions. A following bottom-up approach covers further UAVCAN features that affect the SORA.

The **top-down approach** starts by analysing the requirements defined for the assurance and integrity levels by the SORA. An overview about the SORA shows possible touching points with functions defined by the UAVCAN standard. The AW Drones project is used to transfer abstract requirements of the SORA to existing standards for system safety assessments. The

following figure 1.1 describes the logical flow and exemplary outlines the connection between safety assessment methods and the SORA. The critical points are determined via the safety assessment. Benefits of UAVCAN usage for critical points are outlined and may lead to a reduction of the Specific Assurance and Integrity Level (SAIL). On the other hand the analysis shows limitations and chances for an extension of the range of operations of an UAS above the SAIL of the referenced demonstrators operation.

The **bottom-up approach** lists further functionalities of the UAVCAN standard with varying consequences for the SORA that were not considered by the *top-down approach*. The approach presents new and creative solutions for meeting the requirements by the SORA and estimates their impact. As part of the loosely structured *bottom-up* approach, a market analysis locates all presented concepts within the set of available hobbiest grade and professional UAVCAN components.

The concluding chapter summarises the results of the previous approaches and states open questions and future work.

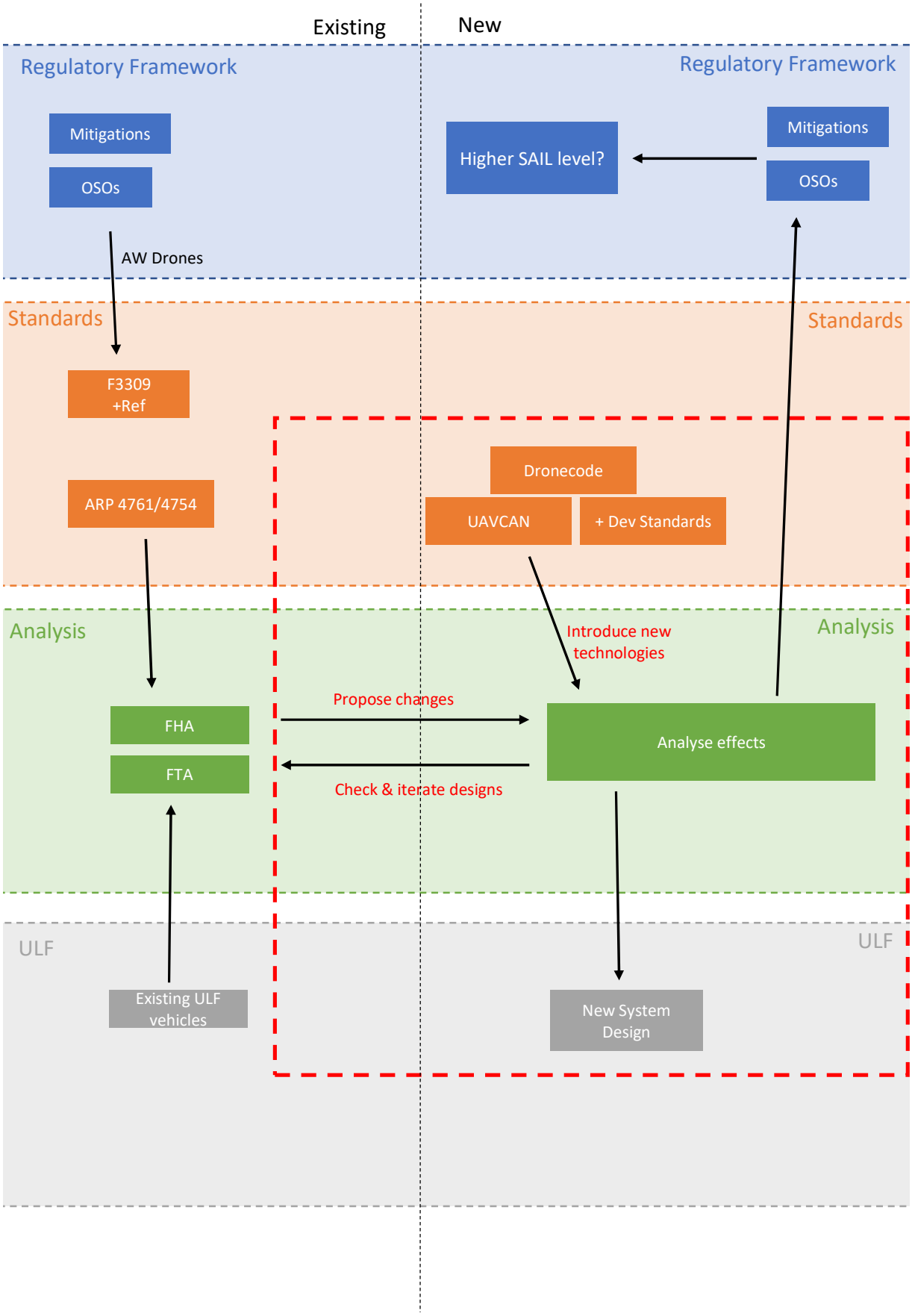


Figure 1.1: Thesis Design *top-down approach*

2 Unmanned Aerial System

UAS describe the group of fixed or rotary wing remotely or autonomous controlled aerial vehicles. Despite the obvious difference between fixed and rotary wing UAS, UAS can be grouped in further classes.

2.1 UAS classes

UAS exist in various forms and sizes and are used for different operations. A rough classification can be done by analysing the weight of an UAS. Additionally the planned operation of the UAS must be considered.

The European Union Aviation Safety Agency (EASA) performs another classification. Depending on their planned operation and worst-case outcome UAS are classified into three major categories.

The open category mainly targets at hobbiest drone pilots with a non-commercial background. Nevertheless a commercial operation within these limits is allowed if the drone meets the requirements formulated in table 2.1. On the other end of the scale the certified category is applied to all UAS that perform high risk missions or missions with a potential great damage to people and/ or infrastructure. For certifying an UAS defined regulations and standards need to be met which require in general more expensive devolopment processes, manufacturing and operational procedures. The certified category is the counterpart of the certified category in manned aviation and permits the operator the general usage of the UAS.

In between the open and the certified category the specific categorie is placed. In contrast to the former described categories the specific category permits an operator of an UAS to conduct a specific mission that is defined in the Concept of Operations (ConOps). The UAS is only allowed to perform missions that are explicetly described within the ConOps. The exact definition of the planned usage of the UAS allows a closer examination of the risks

Class	Requirements	Examples
Open	• AOW: < 25 kg • Safe distance from people and not over assemblies of people • VLOS • < 120 m AGL • No hazardous material	hobbiest drones
Specific	• AOW: > 25 kg • > 120 m AGL or in special airspace • BVLOS	CityATM demonstrator (5.1.1)
Certified	• Above assemblies of people • Transport of hazardous material • Transport of passengers	military drones

Table 2.1: UAS: Classification [18]

and available mitigations. Regulations within the specific category are further defined in the following chapter on the SORA (4).

2.2 Avionic architectures

UAS avionics follow and adopt architectures of manned aviation. Avionic architectures take the required functions and reliability requirements of the planned operation into account. The architectures can be categorized into

- **centralized architectures** - All systems are connected to one central component that performs the majority of the functions. The subsystems are not interconnected.
- **federated architectures** - Functions are hierarchical grouped in tree-like structures.
- **distributed architectures** - All functions are distributed across multiple systems. A communication bus is often used to interconnect components.

The described system architectures ([26]) focus on the information flow and hierarchy between functions. The physical architecture may nevertheless consist of another architecture design. Functional centralized architectures may interconnect distributed components performing small tasks via a data bus.

All architectures for a avionic systems have advantages and disadvantages. The *federated architecture* requires many independent subsystems that are dedicated to a single (set of) function(s). The hierarchical structure composes an inflexible system design that profits of well-defined interfaces and subsystems close to the data source/ sink. *Centralized architectures* consists of a few central components a central locations of an aircraft focus the data flow. The compact system design allows for many optimization concerning the weight and volume reduction, reliability designs and fast inter-function communication. It comes at the disadvantages of higher integration costs and far away data sources/ sinks. The distributed architecture is a combination of the former as remote data acquisiton and processing systems are connected to multiple data sources and sinks. The remote nodes may perform indepent functions locally, but can be hierarchically assigned to a higher-level component, too. The distributed system architecture combines the advantages of the centralized and federated architecture and is used in modern aircraft ([26]).

2.3 Redundancy

"Anything that can go wrong will go wrong" states *Murphy's law.*

Therefore safety-critical systems try to reduce the consequences of a failure. The failure rate of a function is reduced either by the quality of involved components and/ or the redundant setup of the involved components. The effort put into development and manufacturing of components can only be increased until a certain level. Above this limit the manufactoring process, the human developer or the installation process of components cannot lower the failure probability of a component. Alternatively the expense for a component may exceed the economic efficiency in contradiction to redundancy concepts consisting of multiple components. Redundant components may come at a higher effort for integration and a greater weight of the overall avionics, but can compensate the failure of another component.

In case of a failure the system falls back to the redundant component. Based on system design the fall back component may only provide a reduced set of functionality or the guaranteed reliability requirements are no longer met. The system therefore switches into one of the following states in case of the failure of a function:

- **Fail-safe** - The system switches into a safe state and is no longer operational.
- **Fail-secure** - The system maintains maximum security and is no longer operational.
- **Fail-operational** - The systems remains operational.

- **Fail-passive** - The systems continuous operation even though an error occures.

- **Fault-tolerant** - The system avoids function failure and stays operational.

Redundance concepts are categorized in cold, warm and hot redundancy ([14]). Within cold redundance concepts the redundant device is kept in stand-by. The component cannot fail as it is not operated but the activation of the component may leave a gap in the availability of the function. Hot redundant components are continuously active and can immediately provide the requested functionality on request. Warm redundant systems guarantee a response time for the component in stand-by.

A *Triple Modular Redundancy* system describes a 2-out-of-3 redundancy that arranges three redundant components between the multiplied system input and a single voter. The single voter can identify and exclude one malfunctioning component which preserves a *fail-secure* state. Depending on the requirements the state may be escalated to a *fail-safe* state requiring immediate attention as another failure can only be detected and indicated.

2.4 Open Source Community

The availability of affordable motors, sensors and microcontrollers lead to the development of small UAS quadcopters by enthusiast and university projects. Nowadays multiple open-source projects and communities coexist that target at different hardware, operations and target audience. Projects like *BetaFlight* ([6]) and *iNav* ([11]) target at small multicopters for manual and first-person flights. *ArduPilot* ([17]) and *PX4* ([16]) provide more complete autopilot systems supporting multiple aircraft configurations (multicopter, helicopter, planes), flight control, navigation algorithms and on-board mission planning.

The *PX4* project is part of the **Dronecode foundation** ([23]) which sets the development, maintenance and publication of multiple open-source standards and software projects around UAS as its goal. The *Dronecode foundation* facilitates the following projects:

PX4	Autopilot software-ecosystem with guidance, control and navigation algorithm. Integration of multiple other communication standards and simulation environments.
Pixhawk	An open standard that defines hardware specifications and guidelines for hardware development. The hardware and *PX4* software go hand in hand in development, operation and support. The *Pixhawk* standard defines the interfaces, connectors and internal sensors of the autopilot. Third parties, including companies, can manufacture own hardware based on the standard and can expect the *PX4* software to be compatible.
MAVLink	light-weight micro air vehicle communication protocol following a publish-subscribe, point-to-point pattern.
QGroundControl	GCS that provides mission design, flight state control and configuration of an UAS. It communicates via the *MAVLink* protocol with the UAS.

Table 2.2: Dronecode foundation

The *PX4* projects enables support of multiple platforms by a Hardware Abstraction Layer (HAL) that serves as a common interface for higher level software components. Together with an embedded real-time operating system the HAL provides a general hardware independent basis. The intra-autopilot communication standard *uORB* is used to exchange standardized messages between different software components. The software components run as standalone processes e.g. sensor fusion, flight control, navigation and mission planning. The physical interfaces of the autopilot are controlled by software components that forward messages between the *uORB* intra-autopilot communication and connected external devices.

In addition to the regular flight related functions, the *PX4* autopilot implements fail-safe procedures. In case of missing inputs on required interfaces (RC, GNSS receiver, ...) or detected failures of components, fail-safe actions can be automatically triggered. This includes the definition of a geofence that defines the allowed operational area of the UAS and mitigations based on the autonomous flight capabilities of the autopilot software.

The *Pixhawk* hardware and *PX4* software operate an UAS with a **centralised architecture**. Only small tasks are outsourced to external components (GNSS receiver, compass, electrical current measurement) while all flight control and system monitoring tasks are performed by the autopilot.

2.5 CAN Bus

The *Controlled Area Network Standard* was first introduced by *Robert Bosch GmbH* in 1986. Its origin lays in the need of car manufactors for a reliable, deterministic, low-cost inter-car communication. As distributed avionics have the same requirements, the CAN bus is adopted for airborne hardware and architectures, too. By the years the original standard was renamed to CAN 1.0 and increasing requirements by the users lead to *CAN 2.0 A/B*. The latest release is *CAN FD* published in 2012. The *CAN in automation* organization develops and supports the different CAN standards.

The physical layer uses a symmetric, differential pair of cables that transport a *dominant 0* or *recessive 1* ([8]). The recessive state allows the CAN standard to perform a CSMA/CR procedures to allow multiple connected devices while still managing collision on the bus. As all devices on the bus must receive the dominant or recessive bit for arbitration, the bus length is dependent on the data rate. *Low speed* CAN bus with data rates up to 125 kBit/s support bus length of up to 500 m. *High speed* buses may communicate at a data rate of up to 1 MBit/s while the maximum bus length is limited to 40 m.

A CAN frame consists of an message start bit and an object identifier. The object identifier is 11-bit in length for the *CAN 2.0A* standard and 29-bit in length for the *CAN 2.0B* standard. The object identifier allows message receivers to categorise and correctly interpret the received message payload. The DLC flag indicates the length of the data field that can hold up to 8 Bytes (CAN 2.0A/B). The CRC and ACK flags are used for error detection and allow CAN to detect single bit errors and multi-bit errors with a high probability. The *end of frame* bits and *inter frame space* are used to synchronize the clocks of the different CAN bus devices.

The *CAN FD* standard extends the *CAN 2.0B* standard by a concept for dynamic data rates[7]. Devices on the CAN bus can negotiate a data rate that is higher than the nominal bus data rate for the payload transport (up to 5 MBit/s). Additionally the payload size may be increased up to 64 bytes.

The bus load of a *CAN 2.0A/B* and *CAN FD* can be calculated ([2]). Due to the deterministic arbritation process the worst-case response time for a frame can be calculated ([10]) based on the object identifier and considering periodically send data.

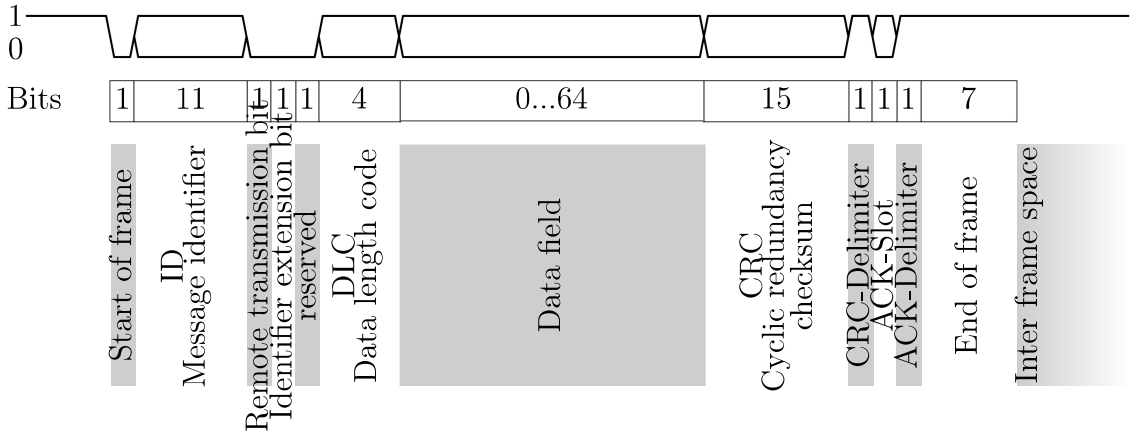


Figure 2.1: CAN 2.0A frame [9]

3 UAVCAN

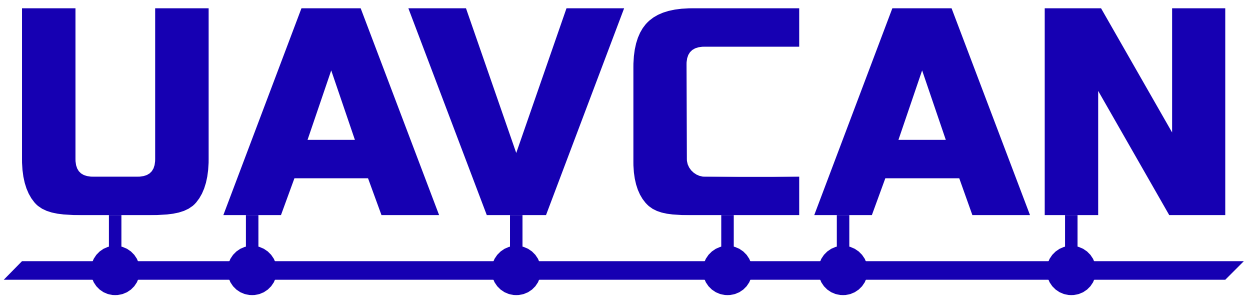


Figure 3.1: UAVCAN Logo [25]

The Uncomplicated Application-level Vehicular Communication And Networking (UAVCAN) protocol was first mentioned in 2014 by by Pavel Kirienko [24]. Back then the acronym UAVCAN stood for Unmanned Aerial Vehicular Controlled Area Network which reveals the origins of the protocol. The common architectur 2.2 for small to medium sized UAS with its 1:1 connections between the flight controller and auxillary devices cannot satisfy the requirements of the ongoing development and constant increase in complexity of UAS. The usage of a data bus is indispensable to transport the information generated by the numerous devices. The I^2C bus is an existing and popular solution for hobbiest projects but the CAN bus fitted better into the following protocol requirements and gave the protocol its name. UAVCAN.

UAVCAN targets to be a

1. reliable
2. intravehicular
3. robust
4. deterministic
5. lightweight

communication protocol in aerospace and robotics. In the following sections the requirements are explained in further detail within the technical details of the UAVCAN protocol.

3.1 Design Principles and Capabilities

UAVCAN is an open standard that aims at providing a reliable communication following a publish-subscribe pattern. Based on robust vehicle bus networks a deterministic on-board data exchange allows to communicate messages and data types defined within an interface description language. Automatic serialization methods ensure reliable design and implementation of interfaces while multiple (redundant) data links ensure a reliable operation. The main design principles are outlined in 3.2.

Figure 3.2: UAVCAN design principles

3.1.1 Version history

UAVCAN is a protocol standard still under development. The first release is called *v0* and is used as the first working prototype functioning as a test bed and basis for discussions. *v0* was under constant development and therefore unattractive for industrial players as support

times for features and compatibililty of software and hardware was not clear. *v1* targets this problem and shall be the long-time supported standard.

As *v0* found already its place in hobby-grade projects and software frameworks like *ardupilot* and *PX4*, experiences and feedback from users are included within the *v1* standard definition. As industrial player are attracted by the success of *v0* multiple industrial partners added their knowledge and comments during the definition of *v1*.

At the time of writing this thesis *UAVCAN v1* definition is still under development. Constant reviews lead to the release of a version *v1 - alpha* in January 2020 and *v1 - beta* in September 2020. *UAVCAN v1 - beta* version serves as the base for this thesis.

3.2 Basic concepts

Within an UAVCAN network multiple peers are combined to a decentralized network. The single peers are called nodes and are identified by an identifier (ID) that is unique within the network. UAVCAN guarantees at least 127 usable node IDs. The nodes communicate via two major methods.

- **Message publication** - one-to-many publish/ subscribe pattern
- **Service invocation** - one-to-one request/ response pattern

The communication methods are grouped into three sectors: *Required*, *Standard* and *custom* functions. *Required* functions are minimum requirements by the UAVCAN standard. *Standard* functions cover all functions included within the published UAVCAN standard. *Custom* function are vendor specific and optional published functionalities. Closed-source systems could rely on them.

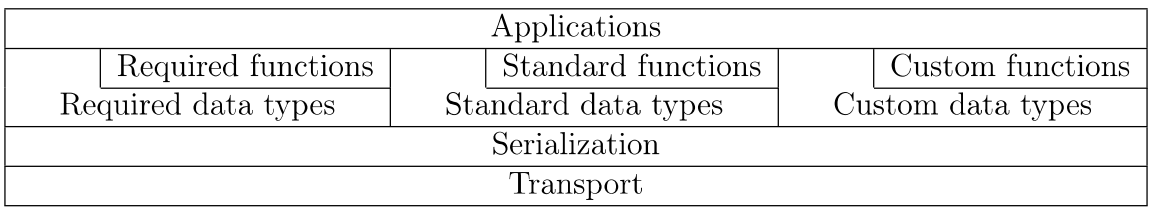


Figure 3.3: UAVCAN architectural diagram

3.3 sorts the functions into the UAVCAN architecture. The functions serve as interfaces to an application. On the other end standardized serialization methods guarantee a correct data exchange using the transport layer.

3.2.1 Communication

Messages and Request/ Response communication between nodes are grouped into subjects and services. Subjects and services indicate the semantic meaning and allows the correct interpretation of the received data. Subjects and services allow functions and processes to identify data exchanged between nodes.

Message subjects are identified by a unique natural number - the *subject-ID*. Messages transmit a serialized data type as the payload over the UAVCAN network to other nodes. UAVCAN is optimized for this type of communication and ensures minimal overhead, communication integrity and large payloads transmission (multiple transport frames). A message is composed of the following properties:

Property	**Description**
Payload	Serialized data structure
Subject-ID	Identifier for semantic background
Source node-ID	The node-ID of the transmitting node
Transfer-ID	Distinction of messages of the same subject

Table 3.1: UACAN message properties

A message containing a payload concerning a specific subject is transmitted by one source node to many other nodes that subscribed for the messages of the subject. To identify messages sent by the same node on the same subject, a transfer-ID is assigned to every single message transfer by a node. The transfer-ID allows to monitor message sequences (and losses), multi-frame transfers, deduplication and automatic management of redundant transports. A special case is the *anonymous* message transmission which allows nodes without an uniqued node-ID to publish data within the UAVCAN network. Nodes without an (unique) node-ID can transmit messages without setting the source node-ID within a message transfer.

Request/ Response services are identified by a unique *service-ID*. The request/ response pattern requires a client node that sends a request. UAVCAN defines that this request has to address exactly one server node. The server handles the request and sends a response to the client.

Property	Description
Payload	Serialized data structure
Service-ID	Identifier for semantic background
Source node-ID	The node-ID of the request/ response trasmitting node
Destination node-ID	The node-ID of the request/ response receiving node
Transfer-ID	Distinction of messages of the same subject

Table 3.2: UACAN service properties

3.2 shows the properties of a service request or response. As with the message transfer a service request/ response can be identified by the combination of the *service-ID* and the *transfer-ID*. The *transfer-ID* serves the same purpose as described for the message transfer. In addition a *destination node-ID* is added to allow a unique identification of the requested server. During the a request the source is the client while the destination of a request is the server. During the response the source and destination node IDs are inverted.

The terms *Subject-ID* and *service-ID* can be condensed into an abstract *port-ID*. The port-ID is assigned to functions, processes or data streams that communicate specific data types. The port-ID is set during the integration process (non-fixed) of a node into an UAVCAN network or is fixed to a specific value (e.g. minumum required functions by UAVCAN).

3.2.2 Data Types

Messages and services communicate defined data types. The definition specifies name, major version, minor version, attributes and an (optional) fixed port-ID. The data types can be defined by UAVCAN maintainers, adapted by UAVCAN or by (closed-source) third parties. Within an UAVCAN communication the data types are identified by the port-ID. 3.3 presents the acceptable combinations of the data types and, their state of publication and unique port identification.

	Regulated	Unregulated
Public	Standard and contributed definitions	Definitions distributed seperately from the UAVCAN specification
Private	Nonexistant category	Definitions only available to authors

Table 3.3: UAVCAN data types

Regulated data types are considered as part of the UAVCAN standard. They are maintained and published by the UAVCAN maintainers within a public repository. Necessary UAVCAN messages and services (high-level functions) are defined within this space. UAVCAN maintainers can accept data types contributed by vendors. Data types within the regulated space are allowed *fixed port-IDs*. Regulated data types need to be public for everyone and so no private definitions exist.

Unregelutated data types are divided between public and private definitions. Private unregulated data types are (closed) source data types. The data types are defined by vendors and can be part of closed-source systems. Fixed-port identifiers for these types are permitted even though fixed-port IDs for closed source data types can interfere with other systems during the integration process. Public unregulated data types are definitions contributed, maintained and published seperately from the UAVCAN standard (e.g. vendors and their servers). The unlimited amount of public unregulated definitions contradicts the limited range of available port-IDs. Therefore fixed port-IDs are not allowed for unregulated data types. System integrators need to set these port-IDs during a node integration into an (existing) UAVCAN network.

The data types are defined within the Data Structure Description Language (DSDL). Common grammar, version compatibility and data serialization methods are defined within the DSDL. The DSDL simplifies the development and integration of UAVCAN capable devices.

3.3 Data Structure Description Language

The DSDL allows the abstract - system and programming language independent - definition of data structures used for communication. Data structure defintions are compiled into system dependent code that can used within code. Automatic (de-) serialization functions can be generated for the defined data structures.

Within a UAVCAN network nodes can be programmed by different programming languages and run on different processor architectures. Nevertheless an inter-node communication must be possible.

A DSDL data type needs to follow certain rules and grammar that ensures correct interpretation of the definition in the following steps.

3.3.1 Message compilation

A python tool can compile DSDL definitions into programming language dependent represantions. These representations can be included as message definitions (C, C++: Headers) into the node code and are compiled into system dependent firmware.

At this stage the memory layout of the message can differ from system to system depending on the endian and/ or processor type, OS, ... The compiled DSDL message can be used as normal data structure definition in the targeted programming language. Inter-system exchange of this data requests for a generic representation of the data structure that can be interpreted by different systems.

3.3.2 (De-) Serialization of messages

Inter-system compatibility of messages is ensured by (de-) serialization of a message in a system-independent binary format. This format follows system-indepent rules defined by the UAVCAN standard. A serialized message can be transmitted by a node while another can deserialize the message and store it in its system-dependent memory layout of the data structure.

DSDL guarantees the correct exchange of data structures between systems of any type by providing aumatically generated (de-)serialization methods.

3.3.3 Extendability and Compatibility

Long-time standards and up-today development don't follow the same principles. Up-today developments request for a high innovation rate, regulary introducing new ways to solve given problems and especially rapid prototyping designing results in in-compatible designs and many diverges in the released versions. Long-time standards on the other side focus on the professionel end user who requests for a broad market and guaranteed support periods for components integrated into its product.

The DSDL tries to close this gap with a naming convention and grammar concept that allows extension and modification (to a certain level) of message while guaranteeing compatibility to other (older and newer) versions of the message definition.

3.4 Message transport

The Open Systems Interconnection model (OSI model) defines seven different layers in a network communication stack from layer 1 (Physical) to layer 7 (Application). The UAVCAN standard covers all of these layers by high-level functions and application interfaces to low-level transport mechanisms. UAVCAN tries to be transport layer independent (OSI-Model Layer 2) and therefore defines multiple abstraction layers.

UAVCAN general concepts including the main principles (3.1), the DSDL (3.3) and high-level functions are independent of the transport layer. Therefore focussing on a concrete transport mechanism (CAN, UDP, ...) is considered as a *concrete transport*.

Table 3.4 gives an more detailed overview of the necessary properties of a message independent of the concrete transport layer.

<table>
<tr><th colspan="3">Taxonomy</th><th>Message transfers</th><th>Service transfers</th><th>Description</th></tr>
<tr><td colspan="3">Transfer payload</td><td colspan="2">Serialized object</td><td>The serialized instance of a specific DSDL data type.</td></tr>
<tr><td rowspan="5">Transfer metadata</td><td colspan="2"></td><td colspan="2">Transfer priority</td><td>Defines the urgency (time sensitivity) of the transferred object.</td></tr>
<tr><td colspan="2"></td><td colspan="2">Transfer-ID</td><td>An integer that uniquely identifies a transfer within its session.</td></tr>
<tr><td rowspan="3">Session specifier</td><td rowspan="2">Route specifier</td><td colspan="2">Source node-ID</td><td>Source node-ID is not specified for anonymous transfers.</td></tr>
<tr><td>Destination node-ID</td><td>Destination node-ID is not specified for broadcast transfers.</td></tr>
<tr><td>Data specifier</td><td rowspan="2">Subject-ID</td><td>Service-ID</td><td>Port-ID specifies how the serialized object should be processed.</td></tr>
<tr><td colspan="2"></td><td>Request Response</td><td>Request/response specifier applies to services only.</td></tr>
<tr><td colspan="3"></td><td colspan="2">Transfer kind</td><td>Message (subject) or service transfer.</td></tr>
</table>

Figure 3.4: UAVCAN transport layer model

Concrete transports may define additional properties to fullfill requirements set within the main principles. E.g. checksums and additional properties for multi-frame transfers are added to the transport layer model.

Every single data transmission between UAVCAN node(s) over the UAVCAN network is called a **transfer**. A *transfer* carries an serialized data type object as its payload together with the associated transfer metadata (3.4). The *data specifier* defines if a message or service with port-ID identifying the transferred data type. For messages the *route specifier* only consists of the source node-ID (n.a. for anonymous transfers). As services consist of a request/ response and client/ server the route specifier additionally holds the destination

node-ID for a response/ request. In summary message transfers are *broadcast* transfers and services are *unicast* transfers.

As the data specifier together with the *route specifier* identify a specific data type transmission between UAVCAN nodes, the *transfer-ID* allows to distinguish between multiple transmissions. The *transfer-ID* is an unsigned integer that is unique during operation. The *transfer-ID* allows

- Message sequence monitoring - Dropped messages can be detected
- Service response matching - The client can recognize the response as it uses the same transfer-ID as the request
- Transfer deduplication - As the transfer-ID is unique duplicated frames with the same *transfer-ID* can be sorted out
- Multi-frame transfer reassembly - All transport frames share the same transfer-ID
- Automatic management of redundant interfaces - Detection of interface failures and automatic switchover

A **transport frame** is the atomic entity a *transfer* consists of. As transfer sizes may exceed the Maximum Transmission Unit (MTU) of a concrete transport, the *transfer* is transported within multiple *transport frames*. This *multi-frame transfer* is handled by UAVCAN and defined within the concrete transports. Concrete transports may add additional properties to the transport layer model.

3.4.1 Transfer Priority

Distributed real-time applications request for priorization of different data. UAVCAN defines a *transfer priority* parameter which range differs for the concrete transports but allows for at least eight distinct priority levels using any concrete transport.

The *transfer priority* shall guarantee and reduce response times of high priorized services. Equally the priorization allows shorter transmission times for messages. Transmission of transfers should be ordered by the priority of the transfer. Concrete transports may offer automatic functions for ordering (e.g. CAN) based on external hardware support.

The UAVCAN standard presents a guideline for the eight guaranteed priority levels from the highest to the lowest priority:

Exceptional	Single messages (e.g. forced reset, system failures)
Immediate	The highest nominal priority.
Fast	This level should acknowledge timing requirements in addition to the requirements of the *high* priority level
High	Messages that are more important than nominal messages with looser timing requirements than fast messages (e.g. important system state controlling sporadic messages)
Nominal	The default message priority used e.g. for the hearbeat
Low	Messages are sent on the bus under all conditions and latency should be constrained by the bus designer
Slow	Only transfer is guaranteed
Optional	Messages that may never be sent. The system shall not rely on correct transmission

Table 3.4: UAVCAN transfer priorities

3.4.2 Redundancy

UAVCAN supports redundant concrete transport interfaces that interconnect (a subset of) an UAVCAN network via more than one interface. A set of nodes redundantly connected are named a *redundant transport group*. A *redundant transport group* can support the fullfillment of requirements for safety-critical applications by creating redundancy in hardware.

UAVCAN is capable of *non-uniform* redundant transport configurations. A redundant transport group could contain nodes with higher reliability requirements while nodes with lower reliability requirements are connected to only a subset of the redundant transports.

3.4.3 UAVCAN/CAN

UAVCAN/CAN defines a concrete transfer via the *CAN 2.0* and *CAN FD* protocol. *CAN 2.0* defines a 27-bit message identifier with a fixed 8 Byte payload while *CAN FD* lifts the communication speed and payload size (2.5).

The CAN standards allow UAVCAN the following capabilities:

Parameter	Value
Maximum node-ID value	127 (7 bits wide)
Transfer-ID mode	Cyclic, modulo 32
# of transfer priority levels	8 (no additional levels)
Largest single-frame transfer payload	Classic CAN - 7 bytes, CAN FD - up to 63 bytes
Anonymous transfers	Supported with non-deterministic collision resolution policy

Table 3.5: UAVCAN CAN capabilities

The capabilities overview in 3.5 shows that they match the minimum requirements defined in previous sections. UAVCAN originated from the CAN protocol and its characteristics. The deterministic communication based on the priorization within the message identifier matches the requirements for concrete transports by UAVCAN. UAVCAN reinterprets the message identifier by placing meta-information about a transfer within the message identifier. The comfort and functionality that come with the meta-information outweight the reduction of available priorization levels. In addition the full payload of a CAN frame can be used for the serialised data object with single-frame transfers optimising the data throughput.

CAN FD offers a larger payload size in comparison to CAN 2.0. As CAN 2.0 and CAN FD share the same message identifier structure most of the implementations apply for both CAN standards. Implementation details and differences are described in the following sections.

CAN ID

UAVCAN/CAN implements message and service transfers by utilizing two different layouts for the CAN identifier. Figure 3.5 displays the meaning of each bit.

Message	Service, not message				Anonymous																							
	Priority					R	R	R	Subject-ID													R	Source node-ID						
Values	$[0,7]$			0	$\mathbb{B}$	0	1	1	$[0,8191]$													0	$[0,127]$						
CAN ID bit	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAN ID byte	3				2									1								0							

Service	Service, not message				Request, not response																							
	Priority					R	Service-ID									Destination node-ID							Source node-ID						
Values	$[0,7]$			1	$\mathbb{B}$	0	$[0,511]$									$[0,127]$							$[0,127]$						
CAN ID bit	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAN ID byte	3				2									1								0							

Figure 3.5: CAN ID bit layout [25]

Messages and services share the same priority range of $[0,7]$. An application is allowed to use any value within this range (3.4). The bit-wise representation of the priority follows the

priorization principle of the CAN bus (2.5). 0 defines the highest priority and 7 the lowest priority.

Following the concept of the dominant 0 messages are prioritised over services as the CAN ID bit 4 is bind to 0 for messages. Both CAN ID bit structures end with a 7 bit field for the guaranteed range of 127 unique node-IDs. As this field is set at the end of the CAN ID it has the least influence on the priorization of the CAN bus.

The bits inbetween the *Service, not message* bit and the source node-ID differ for messages and services. **Messages** indicate an anonymous transmission by recessive 1 at bit position 24. Following three reservered (not yet used) bits the 13 bit-width subject-ID is placed within the CAN ID. This allows 8192 different subject-IDs to be transmitted. Another reserved bit not used within the version of the standards completes the CAN ID with the node-ID for the message transfer.

The more complex logic of **Services** require another CAN ID layout following the *service identification bit*. Requests and responses are identified by bit 24. Only one reserved bit is followed by the 9 bit width service-ID allowing 512 different service identificators. As stated within (3.4) the unicast service transfers request for a destination node-ID. The destination node-ID uses the the same range as the source node-ID and is located between the service-ID and source node-ID within the CAN ID.

CAN data field

The *CAN ID* field is used by UAVCAN to transport meta-information about the transfer. The definitions do not include transfer checksums, multi-frame transfers, etc.. Therefore UAVCAN reserves the last byte of the standard CAN and CAN FD payload for further meta-information. The last byte is called **tail byte** and is defined by the following structure:

Bit	Field	Single-frame transfers	Multi-frame transfers
7	**Start of transfer**	Always 1	First frame: 1, otherwise 0.
6	**End of transfer**	Always 1	Last frame: 1, otherwise 0.
5	**Toggle bit**	Always 1	First frame: 1, then alternates.
4 3 2 1 0	**Transfer-ID**	Modulo 32 (range [0, 31]) (least significant bit)	

Table 3.6: UAVCAN tail byte

Multi-frame transfers indicate the transmission of the first and last frame by setting the corresponding bit in the *tail byte*. The *toggle bit* at third position alternates its state within every frame of a transfer. The *toggle bit* is used for deduplication of frames during the transfer reassambly process. The *transfer-ID* is stored within the last 5 bits.

The CAN standards already defines a checksum for every single CAN frame. In addition *UAVCAN/CAN* transmits a checksum for multi-frame transfers. The transfer checksum is computed over the whole transfer payload and is placed right after the end of the payload within the last frame of a multi-frame transfer. The CRC can be used for validating the correctness of reassambly and discarding incomplete multi-frame transmissions.

UAVCAN/CAN sets a fixed MTU of 8 bytes for classic CAN and 64 bytes for CAN FD. Using classic CAN potential unused bytes between the payload and the *tail byte* shall be filled by zero padding bytes. Multi-frame transfers in addition place the calculated checksum between the padding bytes and right before the *tail byte*. CAN FD allows different defined sizes for the CAN data field. UAVCAN recognizes these constraints and inserts padding bytes. The padding bytes must guarantee independent of the data field size that the *tail byte* and in case of the multi-frame transfers the checksum are placed at the end of the received CAN data field.

In summary 7 bytes of an serialized object can be transmitted within a single frame using classic CAN. CAN FD allows for up to 63 bytes. Therefore *UAVCAN/CAN* guarantees up to 7 bytes to be transmitted within a single - atomic - transfer.

3.5 Application Layer

This section targets the interface to applications and the requirements for applications using UAVCAN. The previous chapters described low-level foundations for transfering data. This section describes how an application must handle data transmitted or received via UAVCAN. The UAVCAN standard additionally defines common high-level functions. Some of the high-level functions are mandatory to support by an UAVCAN node 3.5.1 and some at least simplify the integrations and network monitoring process.

3.5.1 Application Requirements

Before using the high-level functions supported by UAVCAN some general high-level conventions should be followed by an application. The communicated data types and the broad range for port-IDs must follow a specific concept to limit the chaos by the open standard and potential extions from various players.

The broad range of subject-IDs and service-IDs is seperated into three categories:

- *standard regulated identifiers* are reserved for integral data types defined within the UAVCAN standard
- *Non-standard fixed regulated identifiers* can be reserverd by UAVCAN maintainers for publicly released (third party) data types (e.g. published vendor-specific services)
- Unregulated identifiers are freely set by system integrators can be used for any fixed and/ or non-fixed data type subject-ID.

The categories are distributed as shown in table 3.7.

Subject-ID	**Service-ID**	**Purpose**
$[0, 6143]$	$[0, 255]$	Unregulated identifiers
$[6144, 7167]$	$[256, 383]$	Non-standard fixed regulated identifiers
$[7168, 8191]$	$[384, 511]$	Standard fixed regulated identifiers

Table 3.7: UAVCAN port identification distribution

The data types communicated within the regulated identifiers are maintained by the UAVCAN maintainers. A system integrator can assume that these data types can be interpreted by any node offering and potential version differences are handled by the DSDL. Outside the regulated subject-IDs a system integrator must ensure the data types transmitted on a subject-ID are sufficiently congruent so that a receiving node can correctly interpret the data. UAVCAN publishes the standard regulated DSDL data types 3.3 within the *uavcan* namespace.

3.5.2 Application Conventions

In additions to the requirements applications should follow some generic conventions.

UAVCAN application developers encouraged to communicate data using these definitions to increase the compatibility between devices and simplify the integration process. UAVCAN strongly recommends to use International System of Units (SI) units as the basis for data types. UAVCAN defines common SI units that can be used in custom DSDL or raw message definitions. This enhances the type safety and minimizes the risk of human error during the development process. Furthermore UAVCAN recommends specific definitions for coordinate frames and matrix representations. The right-handed coordinate frame definitions are adopted from UAVCAN original purpose - the unmanned *aircraft*.

Frame	**X**	**Y**	**Z**
World	northward	eastward	downward
Body	forward	rightward	downward
Optical	rightward	downward	optical axis

Table 3.8: UAVCAN coordinate systems

Within the node-ID range from 0 to the conrete transport upper limit (guaranteed) 127 the two uppermost IDs are reserved. The IDs linked to diagnostic and debugging tools and should not be used in production systems. System integrators can use these IDs for entry points into the UAVCAN network and manipulate and monitor node functions via high-level functions described in the following section.

3.5.3 Node Status

The UAVCAN standard defines multiple messages and services to supervise the status of nodes within the network. The only mandatory message is the *heartbeat* message.

Every non-anonymous node must report its status periodically within a **heartbeat** message. The *hearbeat* message belongs to the standard regulated fixed subject-ID messages and must published at fixed rate. Following the message definition a *heartbeat* should be published at least every second by a node. Considering the concrete requirement for a node this rate can be increased to allow other nodes a faster response in case a heartbeat is missed. The *heartbeat* message transports the monotic uptime, health state and mode of a node.

Property	Context
Uptime	Monotic uptime of a node (reset on restart)
Health	Abstract health status of the node
Mode	Operating mode of the node
Vendor	Free to use for e.g. vendor-specific status information

Table 3.9: UAVCAN hearbeat message

The message is designed for an atomic transfer within concrete transports with a MTU of at least 8 byte. Classic CAN (and CAN FD) meets this requirement.

The abstract health states of a node are described in table 3.10

Nominal	Component functioning properly
Advisory	Minor failure detected that does not affect the performance of real-time functions
Caution	Major failure detected and performing in degraded mode
Warning	Fatal malfunction unable to perform the intended function

Table 3.10: UAVCAN Health states

The available abstract modes are showed in table 3.11.

Operational	Nominal operation mode (all functions available)
Initialization	Start-up of node in progress
Maintenance	During calibration, setting parameters, self-test, ...
Software Update	Set during the firmware update process of a node

Table 3.11: UAVCAN Node modes

The **generic node information** service allows nodes to obtain detailed information about a node. In contradiction to the *hearbeat* message the information is transmitted only on request but contains more detailed information about a node. The focus of the message lays on the current configuration of the node including the software and hardware revisions. Table 3.12 describes further information sent by the *generic node information* service.

HW & SW version	Revision the HW and SW
...	
Unique ID	This ID should be globally unique allowing the unique identification of devices (e.g. accross multiple UAVCAN networks)
Name	A human-readable ASCII node name simplifying the integration and/ or monitoring
SW image CRC	The value of an arbitrary hash function applied to an image. This allows the validation of the running system image.
Certificate	The certificate of authenticity can be set for reporting optional digital signatures of the node (e.g. read-only non-modifiable vendor certificates)

Table 3.12: UAVCAN Generic node information

The *generic node information* service is considered semi-mandatory as many standard high-level functionalities rely on it. The nework discovery, firmware update and various maintance functions rely on this service and the information responded.

Another standard service is recommended to support to control a node. The **generic node commands** service defines multiple standard commands while it reserves space for vendor-specific commands. As a command targets a specific node it must be service request containing the concrete destination node-ID. The service uses a fixed node-ID and defines the following standard commands:

- Restart - Reboot the node
- Power off - Shut down the node (Restarted at next power on)
- Factory reset - Reset node configuration to factory defaults (hard coded within the software)
- Emergency stop - Immediately enters a safe state
- Store persistant states - Requests a node to store the current configuration in non-volatile storage (faulty configurations can be resetted by restart of the node without storing the configuration)
- Begin software update - The software update process is triggered (3.5.6)

The service defines a 16 bit field for command identification. After substraction of the standard commands 32767 command IDs are free to use for third parties.

The receiption and accomplishment of the requested command is acknowledged by the commanded node. The nodes response contains the status of the command and/ or result of

the command execution. The node can respond the successfull completion or failures during execution of the command. In particular status responses for

- denied permanent storage
- unknown command
- lack of authorization
- node in wrong state
- general internal error

are defined.

3.5.4 Monitoring & Diagnostics

In addition to the services concerning single nodes UAVCAN defines standard services for monitoring and debugging interactions between nodes and the network itself.

The **uavcan.node.port** namespace contains DSDL definitions for messages that transfer the published/ subscribed subject-IDs and requested/ responded service-IDs of a node. A system integrator can activte the publication of these information by e.g. setting a parameter. Nevertheless UAVCAN recommends to permanently the in- and outgoing ports on a regular basis (max. 10 sec) and after each modification. A low message priority guarantees that the optional monitoring information do not conflict with real-time functions.

By obtaining the port information from all nodes within a UAVCAN network the data flow can be monitored. Data flow graphs can simplify debugging and optimizing the communication between nodes.

Within the **uavcan.diagnostics** namespace messages are defined that serve the real-time monitoring and/ or logging of events. The *uavcan.diagnostics.Record* message transfers a human-readable string containing the diagnostic information, a severity of the published information and an optional timestamp. Transfer the diagnostic information as a (inefficient) human-readable string allows the real-time presentation and interpretation of the information as the decomposition of an abstract status code is not needed. Diagnostic information can is classified into the following severity categories:

- **Trace** - System trace information during development
- **Debug** - Aid in troubleshooting

- **Info** - General iformation of low importance
- **Notice** - General iformation of high importance
- **Warning** - Abnormalaties and
- **Error** - Problems and error conditions
- **Critical** - Serious problems and critical conditions
- **Alert** - Dangerous circumstances the require immediate attention

Based on the severity nodes can filter the messages to publish or to process (as a receiver). The treshhold should be modifiable as a parameter which is set to >Warning by default. The timestamp send with diagnostic information should use the network-synchronized time. The synchronized time allows the arrangement of diagnostic message of multiple nodes in post-processing. Potential failure chains of incidents can be identified.

3.5.5 Time synchronization

Within a system of components in general time synchronization enhances the quality of functions at multiple starting points. In context of UAS the best-known use case is sensor value timestamping. Simple system estimate the delay between the measurement and processing of a sensor value as a constant fixed offset. Taking the offset into account allows a more precise sensor fusion and therefore system state estimation. As the system complexity increases the *transfer delay* gets inaccurate and cannot be set as a constant value. Synchronized timestamps between different components of a system eliminate the need for a fixed offset. Sensor fusion algorithms can process measurements linked to a timestamp as long as the timestamp are synchronized between all inputs.

The UAVCAN standard defines a concept for interfaces and processes for time synchronization between nodes. Concrete implementations of the time synchronization depend on the concrete transport. Even though UAVCAN is a stateless and democratic network, time synchronization requires a **master** node as the reference for other nodes within the network. Up to three hierachical sorted time synchronization **masters** set the reference time for the synchronization process. The master nodes should be selected by the accuracy of their internal clock and optional external sources (e.g. GNSS systems often offer global timestamps and accurate interrupts for precise clock counting). The master nodes publish a synchronization message and store the timestamp of the transfer. The synchronization message transports the stored timestamp (of the previous message) or 0 (first message).

The time synchronization **slaves** subscribe for synchronization messages and store the receive timestamp of the last message. The difference between the send and receive timestamp is termed *phase error*. It is the delay introduced by the transmission of a transfer over the CAN bus and the clock diverge between the master and slaves (including the initial difference and lags by the component real-time clocks). Finally the calculated phase error is used to adjust the local clock.

Master nodes publish the synchronization message on all available **interfaces** (in case of redundant interfaces). Slaves can decide on which interface to use for phase error calculation. The sent and receive timestamps of messages must be monitored for every interface.

Multiple masters within a UAVCAN network permantly sent their synchronization messages. The hierarchical order is defined by the node-ID of the masters (lowest node-ID equals highest priority). Slaves receive the messages of all time synchronization masters and calculate the phase error by the timestamps received from the highest priority time synchronization master. In case a slave detects a timeout in the message flow of a master, it switches to the next master within the hierarchie. UAVCAN recommends a timeout treshold of 3 seconds and automatic (re-)acceptance of a higher prioritized master. During nominal operation subordinary time synchronization masters equally run the synchronization logic as the slaves.

UAVCAN supports any **time system** which must stay constant during operation and different synchronizations masters. Time systems referenced to the International Atomic Time (TAI, temps atomique international) are defined by UAVCAN. Information about the currently used time system and the *error variance* can be obtained by the **GetSynchronizationMasterInfo** service from a master.

3.5.6 File System & SW Updates

UAVCAN defines a concept for file transfers. Software updates are defined in another process and use a file transfer for transmitting new firmware.

The **file transfer** concept defines multiple services for requesting information and modifying files and directories. The services are provided by *file servers* and can be requested by any node.

Service	Purpose
GetInfo	Requests information (size, is file, access rights) about a file/ directory
List	Obtains informations about the files within a directory
Modify	Creates, moves or deletes a file on a file server
Read	Copies a file from a file server. Verification via an additional checksum is recommended due to possible race conditions
Write	Copies a local file to a file server

Table 3.13: UAVCAN File transfer services

Every request by the services listed in table 3.13 contains at least the path to the file or path to check or modify. Additional parameters are filled based on the purpose (write, modify). The *file server* can deny service request and set an appropiate error code within the response.

The **software update** process is initiated by sending the corresponding command 3.5.3 to the node. The requested command contains the path to the software image for a node and smaller devices may receive their whole firmware via the software update process. Next the affected node reads the new software image from the update requesting node. During the the update process nodes need to indicate their reduced functionality by setting the *Node Software Update* status. After the update process the downloaded software image should be verified by a checksum. Devices limited in their available memory (e.g. microcontrollers) may need to activate a UAVCAN capable bootloader that overwrites the existing firmware.

3.5.7 Support of integration

UAVCAN allows node configuration during runtime by defining **registers** as interface for key-value pairs that can store node specific information. The registeres are addressed by a human-readable string as the key and can hold data types defined within the *uavcan.register.Value* namespace (e.g. integers, floats, strings). The registers are accessed via services with the node to configure acting the *server*. The server can deny requests as the passed register name does not exist, the value to set exceeds the allowed range or is marked as read-only. The use cases of the registers used as an interface to internal states of a node can be classified in the following categories:

- Configuration parameter management - The node-ID and port-IDs could be defined via registers (non-fixed port-IDs)

- Diagnostics and monitoring - Internal states could be published via registers and severity tresholds for diagnostic messages dynamically set
- Advanced node information - Node information in addition to the UAVCAN standard. E.g. vendor specific configuration and calibration settings
- Special functions and debugging - Non-nominal functions could be triggered and additional internal states for debugging purposes requested

In combination with the *permanent storage* command (3.5.3) the register values can be stored beyond a restart of the node. The mentioned services enable **automatic configuration management** as remote nodes can request the register values and display (3.9) or store them.

UAVCAN defines an highly optional **plug-and-play (PnP)** process that allows nodes to obtain a node-ID without prior manual configuration. The process uses messages and services defined within the *uavcan.pnp* namespace. Nodes that performed the plug-and-play (PnP) are non-anonymous nodes and can regularly send non-anonymous messages and request/ respond to services.

Node-IDs accross the UAVCAN network must be unique. Therefore "PnP masters" - called *allocators* - control the allocation of node-IDs. Redundant allocators are allowed and can increase the availibility of the PnP process. Every allocator keeps a list of allocated node-IDs and synchronizes it redundant allocators. The allocated node-IDs are mapped to the unique-ID of a PnP node. This allows to differ between multiple PnP nodes requesting node-IDs at the same time (e.g. power-on) or granting PnP nodes the same node-ID after a restart. *Static nodes* (with fixed configured node-ID) must be known to allocators before the allocation process starts. There is no auto-discovery service required for static nodes by the standard as the PnP process is an optional functionality. The process is described in more detail within the [25] but as this small introduction already suggests: The PnP process is highly indeterminstic concerning the latency and success.

System designers should avoid mixture of PnP components and high-reliability components. The process contradicts the aim of a stateless and deterministic communication.

3.5.8 Meta-Transport

UAVCAN supports tunneling of packet based transport protocols. Explicitely tunneling of Local Area Network (LAN) packages is defined allowing nodes to connect to a LAN via a modem (node).

Modem nodes are connected to the UAVCAN and a LAN. The LAN can be connected to the internet via Wifi, mobile networks, ... and is physically (as LAN connector) or virtually (node directed connected to an network interface) present. LAN packages are transferred as the payload of a message. The modem forwards the LAN packages and acts as a Network Address Translation (NAT) to redirect possible replies from the LAN package adfressee.

Exemplary uses cases for the transport of LAN packages are:

- Telemetry transmission by a UAVCAN node to a remote server
- Receiption of real-time correction data streams (e.g. for GNSS)
- Automatic updates or query for available software updates

3.6 Reference implementations

The UAVCAN ecosystems develops and maintains multiple reference implementations for multiple programming languages.

Language	Name	Version	Comment
C++	Libuavcan	v0	features all services
C	libcanard	v0/v1	under development for v1
Python	Pyuavcan	v1	currently used for v1 development
Rust	Uavcan.rs	v0	maintenance state unknown

Table 3.14: UAVCAN Reference implementations

The *libuavcan* library should implement the full messages and services feature set. It is considered as the most complete implementation of the presented high-level functions. Currently it is under development for the v1 standard. The *libcanard* - programmed in C - is the complete opposite and implements only the bare minimum required functionality. Developers could base their work on the minimalistic libcanard and implement furhter required high-level functions based on the *libuavcan* reference.

Pyuavcan is the reference implementation for the Python programming language. It is mainly used for *uavcan* development and therefore up-to-date with the *v1* version. In contradiction to *Uavcan.rs* it is not targeted at embedded systems due to the Python programming language.

3.7 PX4 integration

The basic design of the *PX4* software stack is already described in section 2.4 as part of the open-source community. As UAVCAN is supported by the *Dronecode* foundation the *PX4* developers - including companies providing reference hardware - are actively developing on the *PX4* uavcan integration.

The *PX4* UAVCAN integration is based on the *libcanard*. It translates messages send on the UAVCAN bus to the intra-autopilot *uORB* communication and vice versa. The Dronecode foundation defined an own set of DSDL message definition specifically targeted at *PX4* and drones. The *dsdl* data type are part of the set of public regulated DSDL data types published by the UAVCAN organisation.

3.8 Hardware standards

The UAVCAN standard defines multiple connector type as a reference. All connectors have in common that a VCC and ground line connected in addition to the required *CAN High* and *CAN Low* lines.

Depending on the connector type varying maximum currents can be transported. In case of daisy-chained CAN components a system integrator must take the maximum current flow of all components into account. The power supply in cojunction with the data wires allow to power small devices only from the installed UAVCAN network.

3.9 GUI for integration & monitoring

The UAVCAN ecosystem includes multiple interfaces for system integrators to the UAVCAN network. These are either command-line based or a Graphical User Interface (GUI).

Name	Version	Comment
Yakut	v1	Command-line support used during v1 development
Yukon	v1	Graphical interface for high-level functions. This will the future system integrator perspective.
GUI Tool	v0	Old GUI that supports node configuration and commanding. Enable bus monitoring and diagnostic message display.

Table 3.15: UAVCAN Public GUIs

The last named *GUI Tool* is the *GUI* for the *v0* standard and implemented the *libuavcan*. The GUI directly communicated via a *SLCAN interface* and implements bus traffic monitoring, diagnostic message presentation and node listings with node configuration options.

The *Yukon* GUI does not integrate the interface into the GUI application and interacts with the *Yakut* command line interface. The seperation of representation and processing of information simplifies the development.

3.10 v0/ v1 Compatibility

UAVCAN version v0 and v1 have varying DSDL definitions for messages and services. Both versions relay on the same *CAN 2.0B* standard and object identifier. Nevertheless the version can be distinguished by low-level specific differences.

Even though this distinction is possible a component would need to operate two UAVCAN network stacks or perform a message tranlation at low-level. These procedures require a higher processing power and higher memory consumption that may not be supported by low resource components.

4 Specific Operational Risk Assessment

The Specific Operational Risk Assessment (SORA) defines a risk assessment of UAS operations. The SORA focusses on operations within the specific category and precises requirements for operation of UAS that exceed the simplified categories of the open category. An operation appraisal based on the SORA considers the operation as a whole and not only the reliability of the UAS. This reduces the approval on *specific operations* but prevents the potential disproportional costs for *certification* of an UAS (depends on the planned operation bandwidth).

SORA follows a risk-based assessment that initially determines the "static" risk originating fron the UAS during planned operation. This includes a ground risk for people at ground and an air risk considering other airspace users. The risk assessment is based on the potential of fatalities and the *number of people at risk*. An operator may claim to perform mitigations to lower the risk of fatatlies on ground and in air. The result of the qualitative determines a SAIL that defines the requirements for the UAS, operational procedures and external systems.

The upcoming sections describe the SORA in necessary resolution of details for this thesis and follow the released version 2.0 [12].

4.1 Semantic model

The SORA defines a semantic model for defining and describing new terms. A UAS may operate within an operational volume. The operational volume consists of the *flight geography* - that covers the nominal flight - and a contingency volume for that covers the UAS position in case of a failure. The border of the contingency define a *loss of control* over operation

and require for immediate emergency plans. This includes information of manned aviation and grounding the UAS which is taken into account by a *ground risk buffer*.

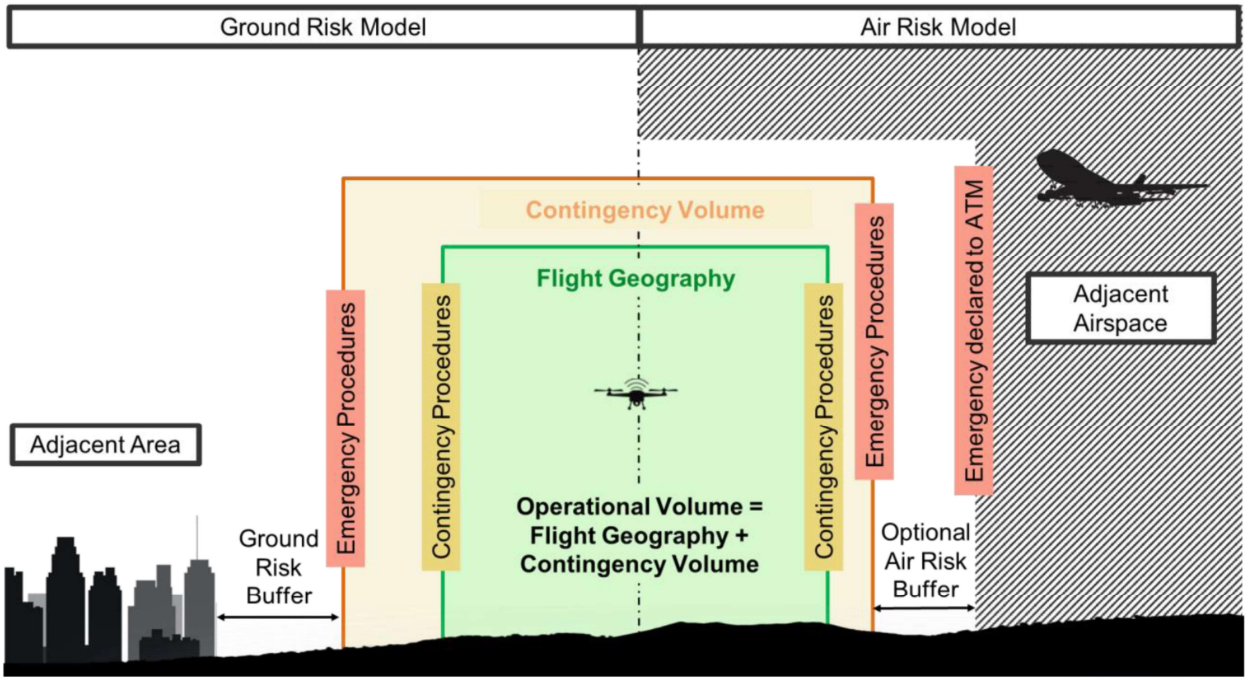


Figure 4.1: SORA Semantic model [12]

The semenantic is seperated into a ground and air risk model. The adjacent areas and airspace of an operation are taken into account by the SORA, too.

4.2 Risk of Loss of Control

Loss of Control (LOC) in context of the SORA means that the

- outcome of situation highly relies on providence
- situation could not be handled by a contingency procedure and leaves the contingency volume
- there is grave and imminent danger of fatalities within any volume

This shows that SORA not only addresses the flight envelope of the UAS, but the supporting conditions for UAS operation.

The SORA references the risk definition of the *SAE ARP 4761* standard:
"The combination of the frequency (probability) and its associated level of severity". The

severity levels are simplified by the SORA as it focusses on all occurance that may lead to fatal injuries on ground or air and damage to critical intfrastructure.

4.3 Robustness in context of SORA

The SORA has an own definition of the term robustness. Robustness is defined by a combination of the level of assurance and the level of integrity and can be used for rating mitigations and OSOs.

Robustness	Low Assurance	Medium Assurance	High Assurance
Low Integrity	Low	Low	Low
Medium Integrity	Low	Medium	Medium
High Integrity	Low	Medium	High

Table 4.1: SORA: Robustness level

Integrity describes the effect of the entity (mitigation/ OSO) under test and its possible impact on the mission. Assurance on the other hand describes the level of confidence that the proposed effects of the entity under test can be achieved.

At this point integrity and assurance are defined in an abstract manner. The description of miitgations and OSOs set these terms and the resulting robustness level in a more specific context.

As the Joint Authorities for Rulemaking on Unmanned Systems (JARUS) SORA is a guideline for national competent authorities and formulates requirements in an abstract manner, the national authorities ask experts for defining and referencing standards that match the abstract definition for a robustness proposed by the international organization JARUS.

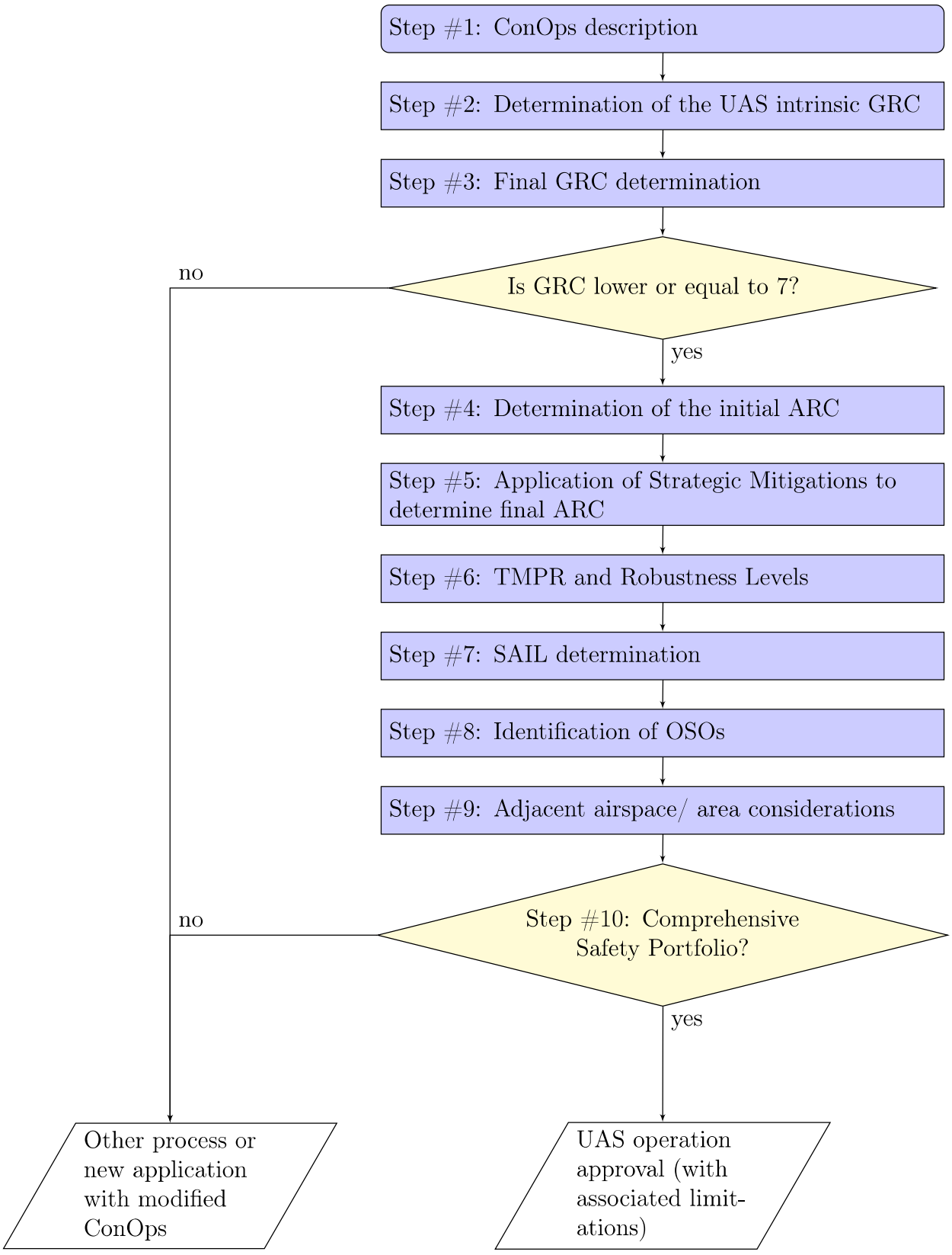
4.4 Workflow

The SORA defines a process (4.2) to assess the risk of an operation and determine requirements for the UAS operation.

The whole SORA is based on the Concept of Operations (ConOps) that defines the *intrinsic risk* for fatalities on ground by the UAS operation. An applicant may use mitigations to

lower the *intrinsic risk* to the *final risk* (Step #3). If the risk of an operation exceeds the limits covered by the *specific category* for UAS, the SORA cannot be used for the approval of the UAS operation.

Figure 4.2: SORA: Process flow [12]



Otherwise the SORA continues by determining the risk for other airspace users and defines requirements for systems that mitigate the risk of a *mid-air collision*.

Step #7 identifies the SAIL of an operation and required robustness levels for OSOs are derived. Finally adjacent areas and airspaces are considered.

4.5 Concept of Operations

An operator must "collect and provide all relavant technical, operational and system information needed to assess the risk associated with the intended operation" [12]. The more precise a ConOps is formulated, the more risks can be eliminated or mitigated.

4.6 Ground Risk

The *ground risk* estimation is seperated into two categories. First the *intrinsic risk* by the UAS in the air and second the *final ground risk* after mitigations reducing the former have been applied. The risk is measured in the qualitative unit *Ground Risk Class (GRC)*. A higher GRC means a higher risk for people at the ground.

The **intrinsic GRC** is determined by estimating the people at risk and potential fatalities. The estimation follows the risk definition (4.2) and seperates the probability and effect. The propability is defined by the environment of the operation while the effects (of a potential LOC) are estimated by the UAS characteristics. The UAS dimensions take the surface affected into account while the expected kinetic energy allows a prediction of the effects on ground. Table 4.2 combines the probability and effects and defines the corresponding GRC.

Max UAS dimension		1 m	3 m	8 m	> 8 m
Kinetical energy		< 700 J	< 34 kJ	< 1048 kJ	> 1084 kJ
Environment	**Flight**				
controlled ground area	VLOS	1	2	3	4
	BVLOS	1	2	3	4
sparsely populated	VLOS	2	3	4	5
	BVLOS	3	4	5	6
populated	VLOS	4	5	6	8
	BVLOS	5	6	8	10
gathering of people	VLOS	7			
	BVLOS	8			

Table 4.2: SORA Intrisic ground risk

Inside a controlled ground area are only active particapants of the operation. Applicants must state in which environment they operate the UAS and how the operation is kept within the ground risk buffer (4.1). An applicant has to underline the operational volume of the UAS considering the 4D position keeping capabilities of the UAS including uncertainties of the sensors, the flight path error and path definition errors.

Specific **mitigations for the ground risk** are defined and arranged in a sequence (4.3). A mitigation may reduce the risk of a person struck by the UAS. The effects and reliability of a mitigation are outlined by a robustness level which determines the quantitative reduction of the intrinsic GRC by a mitigation.

			Robustness	
Mitigation sequence	**Mitigations for ground risk**	Low/ None	Medium	High
1	M1 - Strategic mitigations for ground risk	0: None -1: Low	-2	-4
2	M2 - Effects on ground impact are reduced	0	-1	-2
3	M3 - An ERP is in place, operator validated and effective	1	0	-1

Table 4.3: SORA Ground risk mitigations

Strategic mitigations for the ground risk summarise counter-measurer that reduce the number of people at risk on the ground. Two criterions are used for robustness classification.

1. Definition of ground risk buffer - The ground risk buffer is designed in conservative way and takes improbable single malfunctions (e.g. projection of rotor-parts), metereological conditions, activation of other mitigations into account. Supporting evidence like testing, simulation, design review, etc. assure the defined mitigation functionality as with a potential validation by third parties for a level of hight assurance.
2. Evaluation of people at risk - The number of people at risk can be reduced by operational volume design, time of flight or exactly specified by authoritative density data. Live data (e.g. mobile network clients) can increase the accuracy of the density data and allow for a high level of assurance.

The M2 mitigation *Effects on the ground impact are reduced* imply three criterions:

1. Technical design - The mitigation significantly reduces the effects of a ground impact and is still applicable in case of malfunctions and failures. The UAS operation must safe even in case of a malfunction of the mitigation. A high level of integrity can be claimed by automatic activation of the mitigation and proving that a fatality on ground will not occur when the mitigation is correctly functioning.

2. Procedures - The technical equipment of the mitigation are correctly installed and maintained.

3. Training - The installation and maintenance personell and operator are trained for installing/ maintaining/ activating the mitigation.

The technical design criterion allows a medium level of assurance in case that the applicant documents supporting evidence about testing, inspections, simulations, design review, etc.. A high level can only be claimed by validation by a competent third party. The procedures defined for the mitigation are either self-declared or validated against standards. Additional flight tests and/ or simulation are required for medium level of assurance. Mitigations that are tested within flight tests that cover the whole flight envelope and are validated by a competent third party can claim a high level of assurance. Correct reaction on activation of the mitigation is assured by documention of the training and a training syllabus.

Last but not least an *Emergency Response Plan* shall reduce the effects in the event of a LOC. The ERP defines immediate procedures and escalation steps and does not include any requirements targeting at the UAS and thus UAVCAN

4.7 Air Risk

The the term *air risk* summarizes the probability and effect of a collision or obstruction between the UAS and another airspace user (manned and unmanned). The risk is initially determined based on the UAS and its operting envrionment (airspace class, density). The operations are sorted in four qualitative *Air Risk Class (ARC)* (ARC-a: low to ARC-d: high). Strategic mitigations can lower the *initial ARC* to a *residual ARC*. Based on the *residual ARC* UAS tactical mitigations for mid-air collisions must fullfill specific performance requirements.

The **initial ARC** is the qulatitative representation of the encounter rate with an manned aircraft. The **sora** abstracts the encounter rate by using the airspace class as a reference for the airspace usage and distinguishes between the following chacteristics: altitude, (un-)controlled airspace, air-/ heliport, urban vs. rural and (a-)typical airspace. The SORA uses the airspace characteristics to determine the *initial Air Risk Class (ARC)* following a flow chart ([12], 2.4.2). Competent authorities may extend the characteristics or specific airspace classes to *initial ARC*.

The **residual ARC** combines the *initial ARC* with applicable strategic mitigations. A applicant can claim a lower aircraft encounter rate by following *common flight rules* and *common airspace structures*. The former set of mitigations include airspace surveillance task and traffic avoidance manoeuvres. The *common airspace structure* mitigations include precise collaboration with *ATM* and *Unmanned Traffic Management (UTM)* providers to create safe UAS path with an lower aircraft encounter rate.

The **tactical mitigation** cannot lower the *residual ARC* in contradiction to the GRC as mid-air collisions request for an avoidance manoeuvre. The "See and Avoid" concept is applied to Visual line of sight (VLOS) and BVLOS operations. Especially in BVLOS operations equipment for Detect and Avoid (DAA) must fullfill specific **Tactical Mitigation Performance Requirement (TMPR)**. For **BVLOS** operations detection system are mandatory in combination with (automatic) deconfliction procedures. Table 4.4 depicts the correlation of the *residual ARC* and *TMPR (robustness)*.

Residual ARC	TMPR	TMPR Level of robustness
ARC-d	High	High
ARC-c	Medium	Medium
ARC-b	Low	Low
ARC-a	No requirement	No requirement

Table 4.4: SORA Air Risk: Robustness of TMPR

During ARC-a operations the aircraft encounter rate is extremely low and no further tactical mitigations need to be applied. ARC-b to ARC-d continuosly increase the requirements for detection and avoidance. ARC-b operations are expected to take place below 500 ft resulting in rapic descent traffic avoidance manoeuvre. A detection system should aid the pilot in detecting other traffic and by operating in ARC-b the detection system should be chosen in regard to (active) systems currently in aviation (or specifically in the operational volume). Traffic avoidance manoeuvers should be more advanced. The encounter rate of other traffic is considered that high in ARC-a airspace that the DAA systems must match the performance of manned aviation. An operator must show compliance to DAA system standards accepted by the competent authority.

The TMPR must be met by mitigations that reduce the *initial ARC*. For *VLOS* operations this included See and Avoid (SAA) methods and for BVLOS operations DAA systems. Quantifiable *system risk ratios* are translated into qualitive requirements for the different *residual ARC*.

DAA systems for ARC-c and d *residual ARC* must proove that the *loss of the daa function*-rate is less than < 1 per 10^4 flight hours. The qualitative perspective of this requirement implies that a "single propable condition" may occure. UAVCAN supports the availability requirement by providing necessary low and high level functionalities for redundance concepts that mitigate a probable failure of a single function (5.3).

The TMPR are further classified into detect, decide, command, execute and *feedback loop*.

4.8 Specific Assurance Safety Level (SAIL)

The risk assessment from the previous steps is summarized in an abstract quantified scale, the SAIL. The SAIL combines the air and ground risks and can be seen as an abstract risk estimation of the overall mission.

SAIL Determination				
	Residual ARC			
Final GRC	a	b	c	d
≤2	I	II	IV	VI
3	II	II	IV	VI
4	III	III	IV	VI
5	IV	IV	IV	VI
6	V	V	V	VI
7	VI	VI	VI	VII
>7	Certified category operation			

Table 4.5: SORA: SAIL determination

As the SAIL displays the risk of specific mission, it can be used to identify (abstract) requirements for the UAS, the operating team, third parties and further entities envolved in the performing the mission.

4.9 Operational Safety Objectives

OSOs mirror the identified SAIL to requirements for specific parts of a mission. These parts are the supporting services, the operating crew, environmental conditions and of course the hardware and software of the UAS itself. The SORA defines overall 24 objectives. These are grouped in four categories, namely

- Technical Issue with the UAS
- Deterioration of external systems supporting UAS operation
- Human Error
- Adverse operating conditions

In the following brief presentation of the OSOs the romanic numerals represent the SAIL and the alphabetic character maps to the level of robustness.

4.9.1 Technical issue with the UAS

#	Technical issues with the UAS	I	II	III	IV	V	VI
01	Ensure the operator is compotent and/ or proven	O	L	M	H	H	H

For this OSO an operator must proove his ability to perfom the mission. If necessary by external review.

#	Technical issues with the UAS	I	II	III	IV	V	VI
02	UAS manufactured by competent and/ or proven entity	O	O	L	M	H	H

This OSO targets at the UAS manufacturing process and requires inäprocess and final inspection & testing for a medium level of assurance.

#	Technical issues with the UAS	I	II	III	IV	V	VI
03	UAS manufactured by competent and/ or proven entity	L	L	M	M	H	H

Within this OSO the operator proove the integrity of its maintenance program by available maintenance documents and schedules. The assurance criterion checks on the quality of the maintenance programme and the qualification of the maintenance personell.

#	Technical issues with the UAS	I	II	III	IV	V	VI
04	UAS developed to authority recognized design standards	O	O	O	L	M	H

The OSO determines the robustness level based on the standards applied to the development process. The competent authority may define or refer to concrete standards for each robustness level.

#	Technical issues with the UAS	I	II	III	IV	V	VI
05	UAS is designed considering system safety and reliability	O	O	L	M	H	H

This OSO defines requirements for keeping the UAS within the boundaries of the ConOps. All SAIL must show by analysis and design & installation appraisals that the hazards of probable malfunctions are minimized. For a medium robustness level an operator must prove

that failure and malfuntions can be detected and managed by performing a safety analysis and defining procedures for pre-flight checks. The highest robustness levels defines concrete failure condition frequencies depending on the failure classification (catastrophic - major). The competent authority must validate if the previous requirements are met and additionally specifies industry development, manufactuing and operation standards that must be applied to functions that may lead to hazardous and catastropic failure conditions.

#	**Technical issues with the UAS**	**I**	**II**	**III**	**IV**	**V**	**VI**
06	C3 Link performance is appropriate for the operation	O	L	L	M	H	H

The *C3* provides the command & control functionality and any further communication channel needed for the safety of the operation (e.g. connection to UTM service providers). An operator must show the availability, continuity and integrity of the C3 is mainted during the whole operation (for example by using exclusive communication mediums).

#	**Technical issues with the UAS**	**I**	**II**	**III**	**IV**	**V**	**VI**
07	Inspection of the UAS (product inspection) to ensure consistency to the ConOps	L	L	M	M	H	H

This OSO describes the quality of the UAS inspection. The assurance of the inspection includes documentation, checklist and even validated procedures.

4.9.2 OSOs related to Operational procedures

#	**Operational procedures**	**I**	**II**	**III**	**IV**	**V**	**VI**
08	Operational procedures are defined, validated and adhered to	L	M	H	H	H	H

#	**Deterioation of external systems**	**I**	**II**	**III**	**IV**	**V**	**VI**
11	Procedures are in-place to handle the deterioration of external systems supporting the UAS operation	L	M	H	H	H	H

#	**Human Error**	**I**	**II**	**III**	**IV**	**V**	**VI**
14	Operational procedures are defined, validated and adhered to	L	M	H	H	H	H

#	Adverse operating conditions	I	II	III	IV	V	VI
21	Operational procedures are defined, validated and adhered to	L	M	H	H	H	H

These OSOs target the procedures and checklists before, during and after an UAS operation. Operators must perform pre- and post-flight inspections of the UAS and permantly monitor the environmental conditions during the flight tests. Further, limitations of supporting external system (e.g. GNSS, mobile data provider, UTM service provider) must be addressed in operational procedures. These must include methods for detection of the deterioration and process to on malfunction detection.

A low level of assurance only requires the decleration of adequate operational procedures checked by the competent authority. A medium level of assurance already requires procedure development according to standards and dedicated flight tests (in real-world and/ or validated simulation).

4.9.3 OSOs related to Remote crew training

#	Technical issues with the UAS	I	II	III	IV	V	VI
09	Remote crew trained and current and able to control the abnormal situation	L	L	M	M	H	H

#	Human Error	I	II	III	IV	V	VI
15	Remote crew trained and current and able to control the abnormal situation	L	L	M	M	H	H

#	Adverse operating conditions	I	II	III	IV	V	VI
22	The remote crew is trained to identify critical environmental conditions and to avoid them	L	L	M	M	M	H

This OSO requires from the applicant to show the qualification and experience of the UAS crew.

4.9.4 OSOs related to Safe design

#	Technical issues with the UAS	I	II	III	IV	V	VI
10	Safe recovery from technical issue	L	L	M	M	H	H

#	Deterioation of external systems	I	II	III	IV	V	VI
12	The UAS is designed to manage the deterioration of external systems supporting the UAS operation	L	L	M	M	H	H

This OSO addresses the risk of fatalities when operating above populous areas. An operator must show that no probable failure leads to fatalities Functions whose failure conditions directly lead to a reasonably expected fatality must be developed according to development standard considered adaquate by the competent authority. As assurance criteria an external review is required for a high level of assurance. Medium and low level of assurance can be claimed by analyses and design and installation appraisals.

4.9.5 Deterioation of external systems supporting UAS operation

#	Deterioation of external systems	I	II	III	IV	V	VI
13	External services supporting the UAS operations are adequate to the operation	L	L	M	H	H	H

This OSO focuses on external systems like (GNSS, UTM service providers, ...) **off-board** the UAS. An operator must monitor their availability and performance.

4.9.6 Human Error

#	Human Error	I	II	III	IV	V	VI
16	Multi crew coordination	L	L	M	M	H	H

#	Human Error	I	II	III	IV	V	VI
17	Remote crew is fit to operate	L	L	M	M	H	H

#	Human Error	I	II	III	IV	V	VI
18	Automatic protection of the flight envelope from Human Error	O	O	L	M	H	H

#	Human Error	I	II	III	IV	V	VI
19	Safe recovery from Human Error	O	O	L	M	H	H

#	Human Error	I	II	III	IV	V	VI
20	A Human Factors evaluation has been performed and the HMI found appropriate for the mission	O	L	L	M	M	H

Finally the OSO considering human errors define procedure and simple safety requirements to mitigate the risk of human errors.

4.9.7 Adverse operating conditions

#	Adverse operating conditions	I	II	III	IV	V	VI
23	Environmental conditions for safe operations defined, measurable and adhered to	L	L	M	M	H	H

#	Adverse operating conditions	I	II	III	IV	V	VI
24	UAS designed and qualified for adverse environmental conditions	O	O	M	H	H	H

An operator must continuously monitor the environmental conditions as these are part of the ConOps. Therefore on-board monitoring system may be necessary.

4.10 Adjacent area/ airspace considerations

The last step of the SORA analyses the risk of a LOC during operation for adjacent area and adjacent airspace. The adjacent environment of an operation adds additional safety requirements that target the worst-case scenario: A LOC leading to a UAS position outside the operational volume. The restriction of the operational volume of an UAS is one of the main ideas behind the specific UAS category and therefore bound to strict requirements.

All operations must guarantee that “no probable failure of the UAS or any external system supporting the operation shall lead to operation outside of the operational volume” ([12]). Thus at least a diesign and installation appraisal should be conducted that examines the independence, seperation of function that are crucial for keeping the operation control.

Further requirements are defined for the special operations where adjacent

- areas involves gatherings of people or
- airspace is classified ARC-d

or operations in populated environment

- applied M1 mitigations (strategic mitigations) or
- operate in a controlled ground area.

The requirements define the propability of leaving the operational volume must be $< 10^{-4}$ per flight hour. In addition no single failure of any on-board component or any external system may lead to an operation outide of the ground risk buffer. Systems contradicting the previous requirement must be developed according to indstry standards recognized by the competent authority.

4.11 AW Drones Project

In manned aviation Certification Specification define requirements that need to be met by an aircraft. The unmanned aviation lacks of standards accepted by the competent authorities [27] yet. In context of the SORA this complicates the implementation of the abstract requirement defined within the assurance and integrity levels for mitigations and OSOs.

The EASA targets this gap in the appraisal process for drones (operations) by uniting UAS experts from research, industry and standardization organizations within the AW Drones project. The project aim is to identify existing standards that match the abstract JARUS requirements and can be used as a reference for complying to a specific robustness level.

The project states that its aim is to:

> "supports the rulemaking process for the definition of rules, technical standards and procedures for civilian drones to enable safe and reliable operations in the European Union." [27]

In a top-down approach all existing and available standards on UAS are collected and sorted into suitable categories within the UAS development, manufacturing and operation processes. A bottom-up approach analyzed the potential of the concerning the concrete legislative background. As a part of the bottom-up approach the SORA is used for sorting the standards. The abstract requirements defined within the SORA were mapped to existing standards. An example is given in the excerpt below for the 4.9.4.

Table 230 OSO 10, 12 Standards' effectiveness in fulfilling the requirement (in order of ranking)

Standard Title	SDO	Doc. Reference	Robustness *Criteria 1*			Global Score
			L	M	H	
Integrity/Assurance						
System Design and Analysis	FAA	AC 25.1309-1A	F	F		N.A.
System Safety Analysis and Assessment for Part 23 Airplanes	FAA	AC 25.1309-1E	F	F		N.A.
Standard Specification for Design, Construction, and Verification of Fixed-Wing Unmanned Aircraft Systems (UAS)	JARUS	CS-LUAS	P	P		N.A.
Prognostics and Health Management Guidelines for Electro-Mechanical Actuators	SAE	AIR8012	P	P		N.A.
System Safety Assessment Objectives and Criteria Inputs to AMC 1309	EUROCAE	ER-019	F	F		N.A.
Air traffic management guidelines for Global Hawk in European airspace	EUROCONTROL		P	P		N.A.
Standard Guide for Electrical Load and Power Source Capacity Analysis	ASTM	F2490-05	P	P		11
Specification for Electrical Systems in Small Aircraft	ASTM	F3231	F	F		11
Specification for Flight Controls in Small Aircraft	ASTM	F3232	P	P		4
Standard Practice for Methods to Safely Bound Flight Behaviour of Unmanned Aircraft Systems Containing Complex Functions	ASTM	F3269	P	P		4

Figure 4.3: Excerpt from AW Drones report [27]

After collecting and sorting the standards the standards were rated on their suitability for assigned mitigation or standard. Some standards cover exactly one robustness evaluation while other standards (partially) cover multiple mitigation or OSO requirements.

The referenced standards define necessary quantifiable requirements that component manufactures and system engineers can comply to. Complying to the mentioned standards guarantess the acceptance of a robustness level for the

4.12 ASTM F3309

American Society for Testing and Materials (ASTM International) publishes the *F3309* standard practice that composes methods for c̈onducting a simplified safety assessment of aircraft systems and equipment¨ [3]. The standard is referenced by the AW Drones project (4.11) as (partially) applicable for the OSOs focussing on safe design. Therfore the F3309 can be used as a starting point for safery assessment of **uas** (components).

The *F3309* standard defines a safety assessment process. The process starts by *identification and classification failure conditions* for various UAS functions and connects functions to severity levels. Following the severity classification further analysis must be conducted growing in complexity by the severity levels of failure conditions. Minor severity failure conditions only require a design and installation appraisal for a function. This is a qualitive analysis of the integrity and safety of the design and force a system designer to respect Functional Hazard Analysis (FHA) and safe installation procedures. Major, hazardous and catastrophic failure conditions require a more extensive qualitative analysis focussing on common mode failures and function failure probabilities. The qualitative analysis must proove that e.g. a catastrophic failure condition "at least two indepent failures". The independence of failures must be proven by a common mode analysis including e.g. a *Failure Mode and Effects* analysis.

The previous standard does not define concrete assessment methodologies. Instead the standard references compatible standards. For the *failure condition identification and classification* an FHA in accordance to the *SAE ARP 4761* standard (4.13) is recommended. A Failure Tree Analysis (FTA) performed following the ARP standard can proove the independence claims.

4.13 SAE ARP4761

The *ARP4761 - Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment* standard was published by the Society of Automotive Engineers (SAE) [22] in 1996. The standard presents processes and analysis techniques mainly targeted at aerospace application. In conjunction with the ARP4754 (Guidelines for development) the standard is well-known for development of (critical) manned aviation avionics. Following the findings of the AW Drones project and the *F3309* standard the *ARP4761* can be applied in specific situation within a SORA, too.

The guidelines and methods for the safety assessment define a safety life cycle covering all steps of the aircraft development process. Multiple analysis methods are therefore defined and recommended to conduct within specific steps of the development process.

1. Preliminary System Safety Assessment
2. Failure Mode and Effects Analysis
3. Common Cause Analysis (CCA)
    - Zonal Safety Analysis

- Common Mode Analysis

4. Functional Hazard Analysis
5. Failure Tree Analysis

The Functional Hazard Analysis (FHA) and Failure Tree Analysis (FTA) was performed for a UAS operated by DLR. The analysis is used to identify common failure sources and classify all failure conditions that originate from each function used and/ or offered by an UAS. Specific requirements defined within the SORA require for a more detailed insight into possible interdepdencies of failure conditions. The insight is presented using a FTA defined by the *ARP4761* standard.

5 SORA > UAVCAN (following M600 UAS)

5.1 Description City-ATM Demonstrator

The UAS demonstrator of the project City Air Traffic Management (City-ATM) is used as the reference for the top-down approach. The City-ATM project investigates on the integration of UAS in urban airspace. As part of the project a BVLOS mission is planned, approved and conducted in the *specific category* A hexacopter specialized for BVLOS operations was designed and build and is depicted in more detail in the following sections.

5.1.1 Overview

The Da-Jiang Innovations Science and Technology Co., Ltd (DJI) M600 Pro hexacopter is used as the base platform for the BVLOS demonstrator. The platform is aimed at professional aerial photography and can be equipped with various payloads.

5.1.2 Airframe

The M600 Pro airframe bases on a symmetric hexacopter configuration. 6 brushless motors with integrated Electronic Speed Controllers (ESCs) are mounted to carbon pipes that form the arms connected to the body of the airframe. The arms can be folded down ease transport and are hold in place by brackets in flight.

The (central) body of the airframe takes the loads of the arms and motors and covers most of the equipped high voltage (>20 V) electronics. The 6 batteries are inserted into the body between the mounts for the arms and make contact to the M600 Pro original Power Distribution Board (PDB). The PDB is mounted on the base plate of the airframe.

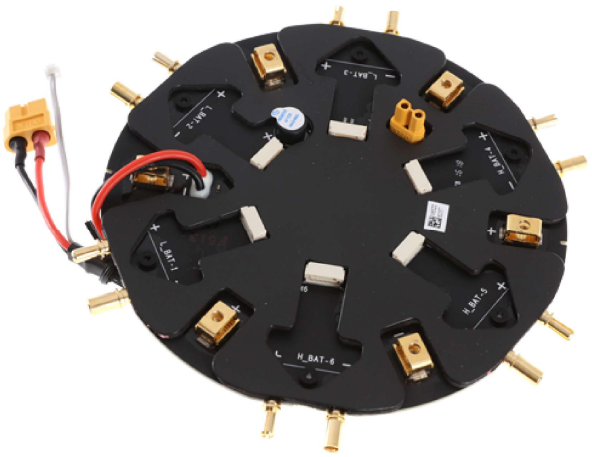


Figure 5.1: M600 Pro PDB [13]

The space above the PDB is reserved for the flight relevant electronics while a payload hinge underneath the body can be used for mounting various payloads. Originally designed for (video-) cameras the payload area is used for experimental payloads by DLR. Even though the payload area is equipped with (experimental) mission supporting hardware, the servo powered retraction of the landing gear is currently not used by the City-ATM project.

Inside the central compartment above the PDB the Autopilot (AP) is located. The top plate of the body offers multiple mounting points for additional sensors (e.g. GNSS receivers) and safety related equipment (e.g. the Drone Rescue System (DRS) and status lights).

5.1.3 Autopilot

The demonstrator is equipped with a Pixhawk series autopilot (2.4). Within the City-ATM project a Pixhawk FMUv5 (Flight Management Unit version 5) manufactured by Holybro ©

is used.

FMU Processor	STM32F765 • 32 Bit Arm © Cortex ©-M7 • 216 MHz • 2 MB memory • 512 KB RAM
IO Processor	STM32F100 • 32 Bit Arm © Cortex ©-M3 • 24 MHz • 8 KB SRAM
On-Board sensors	• Accel/Gyro: ICM-20689 • Accel/Gyro: BMI055 • Magnetometer: IST8310 • Barometer: MS5611
Interfaces	• 8-16 PWM outputs (8 from IO, 8 from FMU) • 3 dedicated PWM/Capture inputs on FMU • Dedicated R/C input for CPPM • Dedicated R/C input for Spektrum / DSM and S.Bus with analog / PWM RSSI • Dedicated S.Bus servo output • 5 general purpose serial ports • SPI bus • Up to 2 CANBuses for dual CAN • Analog inputs for voltage / current of 2 batteries
Power System	• Power module output: 4.9 5.5V • USB Power Input: 4.75 5.25V • Servo Rail Input: 0 36V
Weight	15.8 g
Dimensions	44x84x12 mm
Operating temperature	-40°C +85°C

Table 5.1: M600 Autopilot Specication [19]

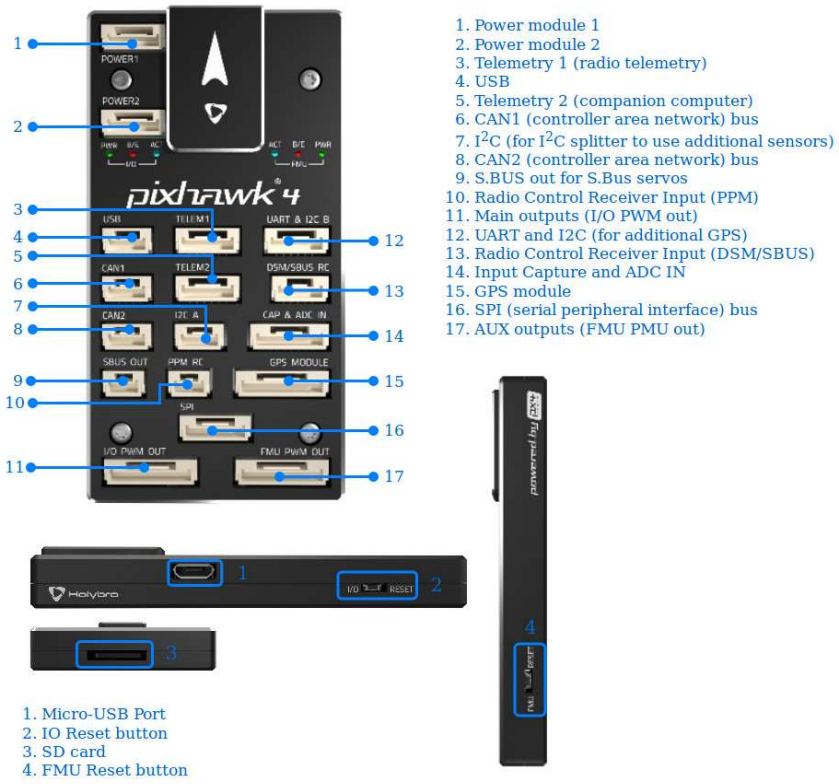


Figure 5.2: M600 Pro Autopilot connectors [19]

The Pixhawk board is running the open source PX4 flight software. Within this demonstrator the PX4 software provides not only the basic flight control, but also the navigation, guidance and DAA tasks and is therefore called an AP. The AP enables the demonstrator to be flown in various manual and automatic flight modes. For operations within line of sight the demonstrator can be manually controlled via a remote control. Even in manual basic controllers are enable to ease the control of the demonstrator at larger distances. The BVLOS operations require automatic flight modes. These cover automatic takeoff/ landing, position hold and a mission (waypoint) mode. As an addition to the mission mode an intelligent *Return Mode* allows automatic return to the closest (as part of the mission) defined *rally point*. These *rally points* act as safe to land in case of emergencies and/ or uncertenties with the UAS.

To fullfill the tasks of the *position hold* and automatic flight modes external sensors are needed that are integrated within the Pixhawk FMUv5. The most important external sensor system is the GNSS receiver with integrated magnetomers. The installed uBlox Neo-M8N GNSS receivers fuses signals of the most known GNSS systems (GPS, Galileo, GLONASS,

BeiDou) and transmits a 4D position via Universal Asynchronous Receiver Transmitter (UART) to the AP. The IST8310 magnemoter communicates via I^2C information about the orientation of the demonstrator in the horizontal pane. As the global position estimation is essential for the operation (nominal and including emergency procedures) the demonstrator is equipped with a second combined GNSS/ compass system. Both systems are actively used for position estimation as their sensor data are fused continiously within the AP.

Within the PX4 software a state machine checks on the activated and valid states. As part of it (flight) modes including physical movements of the demonstrator need to be granted by pressing a safety button beforehand. Potential major errors and system states can be displayed by an RGB LED in an abstract fashion. In case of the City-ATM demonstrator the safety button the the status LED are physically integrated into the GNSS/ compass system. As the GNSS receiver and the compass the safety button and the status LED are connected via seperate electrical lines to the AP.

The combined ADS-B In/ FLARM DAA systems is directly connected to the AP. As seen in figure ?? the AP forwards information about the sourrounding airspace and the demonstrator positions within it via the telemetry link to the GCS.

5.1.4 C2 Link

The *C2 Link* for BVLOS operations is established by the multipath communication system *SKIDER® Link Pro*. The *C2 Link* transmits data received on a serial port via a short range, long range and mobile network connection. The different communication channels can be monitored by a GCS operator.

The *SKIDER® Link Pro* consists of an air and a ground unit that differ in function but are identical in construction. The units are equipped with two serial connections (RJ11 & RJ45) and 5 antennas are connected in total. Two short range (2.4 GHz), one long range (868 MHz) and two mobile network (LTE). The SKIDER® air unit is mounted within the payload are of the demonstrator while the antennas are attached to the arms facing downwards.

Serial data streams received by the air and ground unit are packaged and sent via all three wireless links. While the short and long range connections are encrypted direct links, the LTE connection send the data packages via the (public) mobile data network. The connection between the air and ground unit is established by a Virtual Private Network (VPN) with the access point hosted by the manufacturer. The receiving side ensures the correct reception of the redundantly transmitted packages and outputs the payload on the serial line.

Both units are equipped with status LEDs that indicate the connection status of the three possible data links. As these systems are mounted at remote place for good signal reception, the ground unit also provides a GUI that can be accessed via the ethernet link and a webbrowser.

The AP is equipped with a failsafe configuration that activates the *Return Mode* in case the *C2 Link* connection is lost for defined period of time and no connection to the Remote Control (RC) of the Pilot in Command (PIC) exists.

5.1.5 Detect & Avoid Systems

The City-ATM project aims at the integration of drones into urban environment. Urban environments include manned aerial traffic (airport approach and landing sectors, rescue helicopters, ...) that need to be detected in first place. Secondly the sourrounding air traffic needs to be informed about the demonstrators existence and procedures for avoiding the traffic need to executed. The City-ATM demonstrator uses an *active* and *passive* traffic avoidance system.

The *passive* approach enhances the visibility of the comparable small demonstrator to other airspace users. The stock DJI M600 Pro systems comes with RGB LED lights installed within every motor nacell. The landing skid is modified and eqipped with single LED controllable LED stripes. The LED stripes are controlled by *ARDUINO* microcontrollers that can activate individiual coloring patterns. The coloring of the motor nacell and landing skid LEDs follow the common definition *red on the left* and *green on the right* side. An additional white blinking pattern shall ensure the visibilty on larger distances. The lighting is sufficient for slow flying air traffic. Other airspace users request for long range DAA systems.

A *LX Navigation PowerFLARM Eagle* system is installed. *FLARM* is one implementation of the *ADS-B* protocol and complient systems are optimized to alert pilots on potential collisions with sourrounding aircrafts. The warning is generated based on its own position estimation and sending/ receiving the estimation to/ from other airspace users equipped with a *FLARM* receiver. The *PowerFLARM Eagle* acts as independent system and estimates the position based on an own GPS receiver, barometer and 4D position estimation algorithms. An *FLARM* antenna receives and sends position data and an optional *ADS-B* antenna can receive *ADS-B* signals. The *PowerFLARM* system enables other airspace users to detect

the demonstrator while the received position information of other aircrafts are forwarded to the AP via serial line.

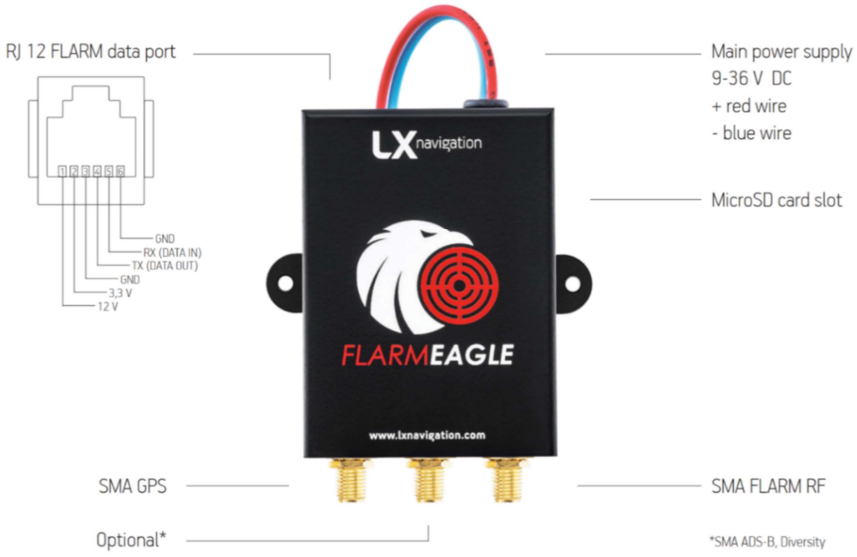


Figure 5.3: M600 Pro PowerFLARM Eagle [20]

A software driver for the *PowerFLARM* system is integrated into the running PX4 firmware so that the AP can interpret the recived information. The AP forwards the received positioning information of the sourrounding aircrafts to the GCS and allows the Ground Control Station Operator (GCSO) an overview of the airspace. As the *FLARM* system is installed on-board the displayed information are always based on the UAS position. The position estimations of the AP and the *FLARM* need to be fused into one visualization.

As part of the fail-safe procedures the AP can activate the *Return Mode* in case another aircraft is crossing a safety distance treshhold (independent of a potential collision path). This can be seen as an emergency procedure as in nominal operation the GCSO decides on the mission continuation as soon as an aircraft is displayed in the GCS.

5.1.6 Power Supply

The demonstrators electronic systems are powered by 6 DJI TB47S smart batteries. The batteries are connected to the ex factory *DJI PDB* and are interconnected via an proprietary communication protocol based on the CAN bus. The interconnected smart batteries perform a self-check before providing power at 22 V to the PDB. If a single battery reports an error or

if the batteries are not equally balanced, no battery provides power to the central PDB and the UAS (components) wont power on. The *DJI PDB* provides power to the UAS powertrain at 44 V and external components at 18 V. The *DJI PDB* implements a redundancy that allows up to 4 batteries to fail while still applying the required voltage to the motors.

The drivetrain and UAS components are connected to the *DJI PDB* as depicted in figure A.1. The Flight Controller (FC) is powered by a another *Pixhawk PDB* enabling the FC to provide power to its internal and external sensors and remote control receivers. Both *GNSS* required for navigation are supplied with power via the *Pixhawk PDB*.

Another distribution board connects to further UAS components. As all channels are protected by fuses the board is called **fuse box**. The *fuse box* is input the external components power suplly from the *DJI PDB* and provides power to the *FLARM*, the *C2 Link* and the optional *Hook-on device* and *FPV system*. The mentioned devices in turn power their sensor systems (e.g. GNSS receiver in case of the *FLARM*).

The **Drone Rescue System (DRS) PDB** is placed in between the *DJI PDB* and the drivetrain (propulsion system). The PDB is part of the drone rescue systems and powers its processing unit and (external) sensors. The *DRS PDB* can cut off the power supply of the drivetrain as part of the mitigation described in detail in section 5.1.8. In addition to the drivetrain the external lighting is powered by the *DRS PDB*.

5.1.7 Propulsion

The propulsion system (also noted as drivetrain) consists of the six independent subcomponents. An ESC, motor and propeller form one component. Every propulsion component is connected to the *DRS PDB* and the FC. The *DRS PDB* provides power at 44.4 V while the FC sends throttle commands to each component via Pulse Width Modulation (PWM).

The propulsion system allows for a Maximum Takeoff Weight (MTOW) of up to 15,5 kg and up to 30 minutes of flight time depending on the payload according to manufacturer.

5.1.8 Drone Rescue System

The demonstrator is equipped with automatic parachute rescue system by *Drone Rescue Systems*. The *DRS* is designed for and tested on the DJI M600 Pro fullfilling the requirements

by the ASTM F3322-18 standard. It consists of the *DRS PDB*, the DRS controller, a sound generator and the parachute (ejection system).

The *DRS controller* provides the logic for the flight envelope protection and geofencing capabilities. An internal Intertial Measurement Unit (IMU) in combination with a non-redundant external Global Navigation Satellite System (GNSS) receiver are used for determining the system status. The GNSS receiver inputs position data into the 4D position estimation, too. Flight envelope and geofence violations lead to the activation of the DRS. Another PWM input allow activation of the DRS by the FC 5.1.3.

The *parachute* is connected to defined fix points on the airframe and is ejected by elastic rubber bands. On activation of the DRS, the *DRS PDB* cut-offs the power to the propulsion system and ejects the the parachute after a defined delay. The delay reduces the risk of parachute lines tangled in the propellers but increases the minimum altitude for activation of the DRS. The minimum recommended altitude for parachute deployment is 10 m according to the manufacturer.

A simple state machine expresses possible states of the DRS via a LED. Displayed system states are e.g. boot, no PWM input, ready for takeoff, error, geofence violation, ... The limits of the flight envelope and geofence can be programmed via a SD Card.

The *DRS* is approved for all M600 Pro system configurations within specifications provided by DJI and summarized in previous sections.

5.2 City-ATM SORA

The City-ATM demonstrator is operated within sparsely populated environment under BVLOS conditions. It has an **intrinsic GRC 2**. The demonstrator claims a low level of robustness for *Strategic mitigations* and the *Emergency Respone Plan.* As it is equipped with a certified M2 mitigation the **final GRC is 3**.

The operation defined within the ConOps uses **ARC-b** airspace. As no strategic mitigation are applied the **residual ARC stays at level b**. Therfore **TMPR must meet the low criterion**.

In conclusion the operation is classified as **SAIL II**.

SAIL Determination				
	Residual ARC			
Final GRC	a	b	c	d
≤2	I	II	IV	VI
3	II	II	IV	VI
4	III	III	IV	VI
5	IV	IV	IV	VI
6	V	V	V	VI
7	VI	VI	VI	VII
>7	Certified category operation			

Table 5.2: M600 demonstrator SAIL

5.3 Shared functions & services

Distributed systems imply the shared usage of functions & services provided by components. Different components can perform high-level functions based on the functionality of single another component. Furthermore redundancies can be implemented without the need for additional components (in comparison hierarchical systems where each components fully integrates dependent components).

In context of the OSOs #10 & #12 (safe design) the F3309 standards recommends performing a FHA and/ or FTA. A Functional Hazard Analysis (FHA) is performed for the M600 Pro demonstrator and presented in appendix B while appendix C contains Failure Tree Analysis (FTA) for exemplary selected functions with a high impact on single or multiple requirements for mitigation or OSO robustness levels.

The FHA for the City-ATM demonstrator shows that multiple functions rely on the same subfunctions. The subfunctionality *position estimation* is requested by the high-level functionalities (physical devices in brackets)

- DAA (FLARM/ ADS-B)
- M2 mitigation (Flight termination/ DRS)
- Flight control (FC)
- Mission control (FC)
- System monitoring (FC/ DRS)

The FTA exemplary performed for the *FC position estimation* (C.3), *M2 mitigation* (C.2) and *DAA functionality* (C.1) describe the effect of failures on the top-level functionality. Example:
The FTA for the listed functions shows that the correct top-level function execution is based on multiple GNSS receivers. Despite the number of four GNSS receivers on-board a redundancy is only implemented for the position estimation by the FC. The other functions rely on a single GNSS receiver resulting in a Single Point of Failure (SPF).

Sharing an existing components (sub-)functionality can reduce the number of required components and can aid avoiding SPF. The concrete transport *UAVCAN/CAN* (3.4.3) enables and guarantees the data exchange between components in distributed system designs under real-time conditions. The UAVCAN concept for the DSDL supports and simplifies the interface (API) definitions for components. The (open-access) standardization based on the DSDL allows any vendor to implement specific functions based on standard message and services (3.2.1). Components of different vendors providing the same function reduce the risk of common cause errors introduced by mistakes during development (HW and SW) or manufacturing of a single component.

One advantage of hierarchical system designs is that components and their provided functions are in full control of subordinated components. The direct connection between dedicated interfaces ensure a deterministic communication. UAVCAN guarantees these existing requirements by node status reports (3.5.3). The challenge of potential introduced delays for sensor values caused via the communication on a shared medium (e.g. CAN bus) is addressed by timestamping sensor values based on a UAVCAN network synchronized time (3.5.5).

A subtype of the shared functionality are *shared sensors*. *Shared sensors* can be used by multiple components and single components can join sensor values of multiple, independent components. Last-named case is used within the position estimation of an UAS.

5.4 4D position estimation

The *4D position keeping capabilities* of a drone are mentioned and referenced in multiple requirements for robustness levels of mitigations and OSOs. Therefore the effects and chances of UAVCAN as part of the position estimation functionality is described in this early section.

The position estimation profits by 2 factors:

- precise sensor information
- complementary and independent sensors

Sensor fusion algorithms take the accuracy of sensors into account. The algorithms can filter potential peaks in sensor values and weight sensor values based on the information about sensor precision (e.g. standard deviation). Complementary and independent (redundant) sensors can simplify the filtering process. Considering the limited number of interfaces of a FC and the required precisely timed sensor information, the previously presented concept of *shared functions* enhance the position estimation.

The *4D position estimation* includes the factor time as the fourth dimension. In context of the SORA this does not only imply the calculation of speed and acceleration, but the continuous and reliable availability of position estimation information. The *shared functions* section already states the avoidance of SPF and common mode errors as advantages of UAVCAN. The FTA shows that the DRS, the FLARM system and of course the FC rely on a precise position estimation.

The position estimation for the flight control and navigation tasks of the M600 Pro demonstrator is performed by the FC. The hierarchical architecture requests for an own set of sensors connected to the FC. The FC is connected to internal (on-chip) IMU, compass and height sensors. The height is calculated by an integrated precise barometric sensor. External sensors are connected via serial line including two GNSS receivers and I^2C including two compasses. The sensor fusioning ans position estimation algorithm require correct functioning of IMU, compass, GNSS and height sensors to calculate one exact position. These functionalities are SPF and partially installed with redundancies. Internally two IMUs are installed. The single internal compass is extended with two external compasses. The compasses are installed at a remote position on the UAS aquiring more precise measurements about the UAS horizontal orientation. All compasses are redundantly used in hardware and software. The GNSS receivers are placed at a remote location on-top of the UAS allowing clear view of the sky that improves the receiption of satellite signals and the determined positional information. The UAS height information Even though the height of the UAS is measured by a barometric sensor and two GNSS receivers, the height estimation must be carefully checked. GNSS receivers can not provide as accurate information about the z-axis as barometric sensors do. Considering this an additional barometric sensor is recommended to prevent a SPF.

In contradiction to the position estimation of the FC, the sensor sets of the DRS and DAA system is not as well equipped. FTA (C.2) shows that the DRS position information used for

flight envelope protection and geofencing rely on a single GNSS receiver. The same applies to the DAA (FLARM) subsystem. Both system could input their position data into the FC and/ or benefit from the fusioned position estimation by the FC. UAVCAN encourages this distributed system setup as described in section (5.3).

Once in the air the UAS and its sensors are exposed to the evironmental conditions. During flight GNSS e.g. the offsets can change, the barometric pressure in the operational volume can change or local anomalies of the magnetic field of the earth are passed. Several concepts exists to address the stated anomalies. GNSS offsets can be corrected by Differential GPS systems, barometric pressure can be corrected by reference stations on ground and magnetic field data can be accessed via open or closed source web-services. Without the need equip every involved component with its own data link for receiving correction data, UAVCAN allows the transport of non-internal (no DSDL definition) messages via the meta-transport concept (3.5.8). Components can request correctional data for the current UAS position via modem nodes from e.g. webservers and therefore improve the accuracy of measured sensor values. The system complexity does not proportianal increase by the number of correctional data receivers as only one (or a defined number of redundant) modems establish the connection to external services.

Low altitude flights require for a precise height above ground level (agl). On the one hand further sensors could be installed (e.g. radar altimeter, LIDARs) following the shared functionalities concept. On the other hand additional height information of the ground level could requested like previously described correctional data. Sensor fusion algorithms or "intelligent" height sensors (based on barometric sensors) could publish more precise AGL information without additional physical sensors.

5.5 HITL simulation

Multiple mitigations and OSOs require for flight test and/ or tests within a (validated) simulation to assure a specific robustness level. The existing solutions for star architectures require an additional abstraction layer or physical modifications to the Device Under Test (DUT). An abstraction layer discards input an interfaces and feds simulated data into the following data processing. This procedures requires changes to the software running on a compotent in comparison to the software version and configuration during the operation. Alternatively a Hardware in the loop (HITL) simulation is connected to all physical interfaces

of the DUT. This procedure corrupts the installation appraisal process and requires extensive checks before the next operation.

Distributed systems and distributed functions enable a HITL simulation without any physical and software modifications to the DUT. A system integrator can easily connect a HITL simulation via a diagnostic port to access the UAVCAN network. Specific nodes may be remotely deactivated by node commands (3.5.3) and replaced by the HITL. The DSDL definitions of messages can be compiled for varying processing architectures and therefore adapted to existing (validated) simulations. Via the diagnostic port the HITL feds emulated data of deactivated components into the UAVCAN network. Malfunctions like

- (Single) GNSS drift
- Compass errors
- Mitigation failures (status reports)
- ...

can be emulated and used for tests of system monitors (6.2) or automatic activation of defined procedures (e.g. mitigation activation, ignoring malfunctioning sensors).

5.6 Intrinsic GRC

The *intrinsic GRC* is determined by operational scenarios and the maximum characteristics of the UAS.

The operational scenarios are classified by the operational volume defined within in the ConOps. Enhanced position keeping and navigation capabilities of the UAS can reduce the required flight geography by the UAS for performing a specific operation. Methods and suggestions for reduction of the flight geopgraphy are defined in section 5.4. The required contingency area is taken into account in the following mitigation sections. Obviously the effect of enhanced navigation capabilities depend on the ConOps and the adjacent area aroung the flight path.

The description of the demonstrator and the power flow diagram A.1 show that the demonstrator is equipped with at least 4 GNSS receivers (2x FC, FLARM, DRS). The current configuration shows the each subcomponent carries its own set of sensors. Operations performed at higher SAIL levels require for additional (redundant) components. These components may introduce further similar sensors that are only available for the specific component

even though they measure the same values. *Shared sensors* (5.3) could be used by multiple components and reduce the weight of the UAS. The lower mass of the UAS leads to a lower typical energy and lower *intrinsic GRC* 4.6. Following the example configuration of the demonstrator three GNSS receivers could establish *triple modular redundancy* providing positioning information to the FC, FLARM, DRS and other components while still saving the weight of one GNSS receiver. Similar concepts can be used for other sensors (e.g. IMU, barometer, compass).

5.7 Final GRC - Mitigations

The *final GRC* used for SAIL determination is determined by applying mitigations to the *intrinsic GRC*. The mitigations are categorized in M1 (), M2 () and M3 (Emergency Response Plan). The M1 and M2 can benefit from UAVCAN and are described in the following sections. The M3 mitigation focusses on procedures in case of an emergency without taking technical features of an UAS into account and is therefore not considered any further.

5.7.1 M1

The M1 mitigation rates the *ground buffer* calculation of the UAS. A higher level of robustness (and therefore reduction of the *intrinsic GRC*) is applicable the more precisely a ground buffer is defined and the more reliable a ground buffer will cover the operational volume under all conditions.

A low level of integrity can be claimed by calculating the ground risk buffer based on the 1:1 rule. Operations at higher altitudes and/ or environmental conditions that affect a mitigation activation (e.g. wind) may extend the required ground risk buffer to an unacceptable size. In case of the M600 Pro demonstrator the activation of the DRS at greater heights lead to an enlargement of the ground risk buffer. When the DRS detects an violation of the geofence or flight envelope, it immediately activates the parachute deploymet process. Considering a precise and reliable *4D position estimation* including AGL (5.3) information the DRS could deploy the parachute at lower heights resulting in a smaller required ground risk buffer. In case of malfunctions and failures this procedure could lead to operations outside the ground risk buffer. Therefore this function must be considered as a **SPF** for the operation and must be handled during the safe design process (4.9.4).

The second criterion for this mitigation takes the people at risk into account. “Static” population density data (aquired prior to takeoff) assure up to a medium level of assurance. A high level of assurance requires near-real time monitoring of the density on ground. UAVCAN can support acquiring these information by the *meta transport* (3.5.8) functionality.

The minimum operation height is limited by the DRS activation procedure. As the motors are controlled only via the FC and send no feedback to the FC and DRS, the DRS can only intervene in motor operation by cutting the power off the motors. The DRS system reserves a delay before parachute deployment to allow for a complete motor stop avoiding parachute lines to get tangled in the propellers. The delay leads to two disadvantages. On the one hand a potential rotational momentum continues and possibly brings the aircraft into a state even further outside the flight envelope. On the other hand the delay increases the free fall distance before parachute deployment and therefore the minimum required height that allows the parachute to fully expand. Following the 1:1 rule for ground risk buffer the increased operation height leads to a larger ground risk buffer. This increases the potential people at risk and may lead to a higher SAIL level.

The described concepts for *shared functionalities* (5.3) and *system monitoring* (6.2) imply that an UAVCAN capable ESC can report its current status and react to inputs from multiple sources. The status can include the current consumption, the voltage and real-time RPM of the motor. The DRS system can publish an emergeny report message (3.5.4) at a high message priority (3.4) that ensures the real-time transfer. The UAVCAN capable ESC receive speed commands from the FC but additionally react to emergency-stop commands. As soon as the ESC reports that the motors are stopped, the DRS can deploy the parachute. This procedure eliminates the general delay before parachute deployment and leads to a lower operation flight height. The lower operation height allows for a smaller ground risk buffer simplifying the operation planning and avoiding high(er) risk areas leading to a lower SAIL.

5.7.2 M2

The M2 mitigations states that all elements required for a mitigation must be on-board. In addition in must be proven that “any failure or malfunction of the proposed mitigation does itself [...] does not adversely affect the safety of the operation”([12]). In context of distributed architectures and UAVCAN this can be translated to close monitoring of the mitigations status and dependent components. The section about *shared functions* (5.3) and *system monitoring* (6.2) analyse chances of UAVCAN in further detail. A *configuration*

management node (6.3.1) may even check the systems state and correct installation as part of a self-test before takeoff and wont clear the mitigation status before the systems configuration is confirmed.

In tandem with the monitoring the reliability of a mitigation must be sufficiently high enough to perform the mission. As the reliability of monitoring and failure detection concept lowers the risk of an operation, it does aid in increasing the operation performance. Multiple sensor inputs from *shared sensors* between components and status message from all UAS components increase the reliability and availability of the mitigation and therefore of the operation. Automated activation of the mitigation (e.g. by geofence violation or detected failures of UAS components) is required for claiming a high level of integrity. UAVCAN implements all requirements for performing such an automated mitigation activation.

The mitigations assurance levels based on certification of components (as the DRS), extensive flight tests and reviews by third parties (only for high level). In the future a specific redundant sensor input configuration could be required by recommended standards and implemented via UAVCAN. Analysis of the (automated) mitigation processes are supported by the chances of a validated HITL(5.5).

5.8 Initial ARC

The *initial ARC* is only determined by operational volume and the encounter rate of manned aircraft. UAVCAN cannot support the reduction of the initial ARC directly as no technical requirements can be derived from this qualitative classification. Nevertheless the enhanced *4D position estimatation* (5.4) can lead to a reduction of the required operational volume. In areas with dense airspace already small modification of the operational volume can help, avoiding an airspace with higher encounter rates (e.g. air-/ heliports) leading to a lower *initial ARC*.

5.9 Residual ARC (Strategic Mitigations)

Lowering the *initial ARC* to a *residual ARC* can be done by high-level strategic processes and concepts. The mitigation does not address requirements for single components but defines functions that must be available to lower the ARC. The reduction can be claimed in case

common flight rules and/ or *common airspace structure* are respected. The requirements for the functions are defined within the following *TMPR* analysis.

Common flight rules implies electronic cooperative systems and anti-collision lightings. The *DAA* system of the City-ATM demonstrator allows other airspace users to detect the demonstrator and enables the demonstrator to check on a high airspace usage before takeoff and during flight. Continuous monitoring of the airspace usage allows an operator (together with further requirements) to lower the initial ARC resulting in a lower SAIL level. The demonstrator is equipped with multiple lighting systems (5.1.5).

Common airspace structures set the boundaries for UAS flight operations. Upcoming UTM service providers could define define specific transit corridors for UAS through airspaces with a high encounter rate. The transit corridors allow for a reduction of the *initial ARC* but must be coordinated with the UTM service provider and strictly followed. If the communication with an UTM service provider will be performed on-time in-flight the *meta-transport* (3.5.8) function of UAVCAN enables a reliable connection to the service provider. The benefits for following UAVCAN concepts for precise navigation are already stated in 5.4.

5.10 TMPRs

The *detect* function defines the percentage of aircraft within the operational volumen that must be detected (low: 50%, med: 90%). UAVCAN and its well-defined interfaces (DSDL) allow for the simple and robust integration of multiple detection systems. Installation of independent components allows to support differenct DAA standards (e.g. in addition to the FLARM in case of the City-ATM demonstrator) that hide their complexity behind a common DSDL service for DAA systems. Dependent components must only support the single DAA DSDL service definition and can process the input from multiple components. The *decide* function sets requirements for the decision process in case of a collision path detection. The collision path detection can be performed by an operator, the FC or another independent on-board system. This system could be a system monitor or the DAA system itself. Collision path detections can be reported as a high priority message (guarantees maximum delays) and automatically activate appropriate mitigations.

5.11 Operational Safety Objectives

The following section contain specific analysis of the OSOs more or less affected by the UAVCAN standard.

5.11.1 OSO #2 / #3 - manufactured/ maintained by competent and/ or proven entity

UAVCAN can support the final inspection & testing process. As part of the testing process correct function of the component and calibration of e.g. sensors is performed. UAVCAN addresses this requirement by registers 3.5.7 that allow modification and permanent storage of parameters not known at software development. An automatic testing rig can be set up that is connected to the DUT and calibration tools. The testing rig can trigger the DUT function, measure the effect (e.g. sensor offset) and automtically set and save the corresponding register on the DUT. The documentation can be organized by the recommended globally unique device-ID (3.5.3). An automated testing rig could adapt some of the concepts of the *HITL simulation* (5.5).

Even though the OSO is target at the UAS as a whole system, UAVCAN can support the maintenance of single components. Entry points into the UAVCAN network allows maintenance personell to trigger test on single components and read-out logged diagnostics management 3.5.4.

The required *maintenance programme* for a medium level of integrity could include checking an resetting and operating hours counter of the UAS or components. If the maintenance is not performed a component could report a "maintenance required" message and set a corresponding node state 3.5.3. As the node status is periodically published dependent device could block processing of received messages and stop further system operation.

As the OSOs are mainly target at a maintenance programme complying to standards, UAVCAN can only support the processes accross all robustness levels and SAIL.

5.11.2 OSO #5 - designed considering system safety and reliability

This OSO requires the UAS to stay within the defined ConOps (4.9.1). In regard of the operational volume section *position estimation* (5.4) presents concepts that enhance the

position keeping capabilities of the UAS. The AW Drones project considers methods defined by the *ASTM F3309* standard as suitable for the required FHA. The Annex B presents such an analyse that is referenced in detail in sections 5.7 and 5.11.6.

The medium robustness level requires for failure condidition detection and management concepts. The previously presented *shared sensors* concept (5.3) presents redundancy and independence concepts realised by UAVCAN usage the prove the management of failures by an UAS. The sections 6.2 describes system monitoring with UAVCAN in more detail while the previous section 5.11.1 already presents UAVCAN functions that support an installation appraisal. A *configuration management node* (6.3) supports the pre-flight check before takeoff that is required for medium level of assurance.

A high level of robustness must be proven by concrete failure condition frequencies. Redundancy and independecy claims presented with the *shared functions* section support a system designer to meet the requirements using lower quality components (e.g. available Commercial of-the-shelf (COTS) components D).

5.11.3 OSO #6 - C3 link characteristics

OSO #6 describes the general performance characteristics of a data link. UAVCAN is not target at acting as C3 link for BVLOS operations but supports monitoring of the *air side* of a single or multiple C3 links.

SAIL *II and higher* operations must continuously monitor the C3 link perfomance. The M600 Pro demonstrator uses MavLink hearbeat messages exchanged between the GCS and FC to monitor the C3 link performance.

Apart from the serial line connection to the FC the C3 link could be connected to a UAVCAN network. Thus the C3 link could join the common status and diagnostic messages network allowing independent monitors to check on the performance of the C3 link. Section 6.2 describes potential system monitoring in detail.

The M600 Pro demonstrators simple layout already allows to claim a low level of robustness. Nevertheless UAVCAN can support more complex systems to claim at least the low level of robustness.

5.11.4 OSO #7 - Inspection of the UAS to ensure consistency with ConOps

As the City-ATM demonstrator operates within SAIL *II* only a trained and documented UAS inspection is required. For a medium level of robustness the product inspection must be documented using checklist. UAVCAN allows connected components to easily report diagnostic information. This report can act as a check mark for the remote crew.

In combination with the proposed *Configuration Management Node* (6.3) inspection of the UAS could be easily documented fullfilling the requirements for a medium level of robustness (SAIL *III* and *IV*).

5.11.5 OSOs related to operational procedures

UAVCAN aids an operator in detection of deterioration of external systems. Before takeoff a *configuration management node* (6.3) can perform a pre-flight check (self-test) of all on-board devices and report errors and diagnostic information. *System monitors* (6.2) can closely and automated monitor the correct functionality and identify components that are affected by an external system malfunction. Procedures for mitigation of the deterioration can be automated or are even integrated into the UAVCAN standard. Redundancies in communication and GNSS acquisition can be supported by *shared functions* (5.3). A system monitor and/ or operator can send commands to components 3.5.3 to regain the correct operating mode (e.g. forced restart).

Dedicated flight tests and/ or validated simulations must test the defined procedures to assure a medium level of robustness. The HITL capabilities described in section 5.5 may support the testing process as component states can be modified during runtime and e.g. sensor inaccuracies, drifts and malfunctions can be emulated.

5.11.6 OSOs related to safe design

OSOs #10 & #12 are summarized by the term *safe design*. A low level of robustness is claimed by the City-ATM demonstrator by not operating over populous areas or gatherings of people.

For operations in previously stated environments all SAIL levels must at least fullfill the requirements for a low level of robustness. The level can be assured by respecting design and

installation features like independence, separation and redundancies for components that lead to a fatality in case of an failure. UAVCAN supports independence of components by enabling all components connected to the UAVCAN network to access published information. Functional separation is supported by the concept of *shared sensors* 5.3 and redundancy can be introduced redundant components and a redundant UAVCAN networks 6.1.

The SORA states that for SAIL *III* and *IV* operations over populous areas no single failure of the UAS or external systems will lead to fatalities. Components that are a SPF leading to fatalities must be developed according to standards considered adequate by the competent authority. Software and hardware developed in accordance to standards often come at higher costs and are less available on the market (D). Therefore UAVCAN can support the avoidance of SPFs by redundancy. Components that may directly lead to fatalities can be installed in a redundant manner avoiding a SPF. Costly components developed in accordance to standards can be replaced by COTS equipment while claiming a medium level of integrity. Redundant setups may support the system design process claiming a medium level of robustness as certified UAVCAN components are not yet available.

In addition to a design and installation appraisal the claimed system design integrity must be substantiated by analysis and/ or test. For example the *shared sensors* and *4D position* sections are based on an FHA and FTA analysis.

5.11.7 OSO #18 - Automatic protection of flight envelope from human errors

The City-ATM demonstrator is not affected by this OSO as it operates without a human in the control loop and at SAIL *II*.

Operations performed with a human in the loop and above SAIL *II* must claim at least a low level of robustness for the flight envelope protection. A flight envelope protection is a subcategory of a system monitor. UAVCAN can support the sensor input into the flight envelope protection (5.4) and integration of a system monitor 6.2.

5.11.8 Adverse operating conditions

The SORA requests for procedures to evaluate environmental conditions before and during the mission. The real-time evaluation of sensor values for determining the environmental conditions can be supported by UAVCAN. A UAVCAN network allows for standardized

interconnection between sensors. The UAVCAN standard can therefore be part of the verification process for claiming a specific level of robustness.

As a high level of robustness includes tests that cover all environmental conditions within the flight envelope, UAVCAN can support execution and documentation of the test by node commands, registers and the recommended unique device-ID.

A high level of assurance can be claimed if devices are continuously maintained. UAVCAN supports this requirement as stated in 5.11.1 and allows easy remote access to highly integrated sensors. E.g. air speed sensors integrated into the aircraft hull can be checked remotely without disassembling the aircraft hull. In combination with fixed installation appraisals the remote access can simplify the maintenance procedures for deeply integrated components.

5.12 Adjacent area/ airspace

The last step of the SORA formulating technical safety requirements defines that no singular probable failure may lead to operations outside the operational volume. The *shared functions* (5.3) concepts supports this requirement by offering a system integrator redundancy concepts with indepently developed components.

The second - more stringent - set of requirements takes the position accuracy into account and benefits of the strategies presented in the *position accuracy* section (5.4). SPF that lead to an operation outside the ground risk buffer must be costly developed according to industry standard. UAVCAN offers concepts for SPF mitigation by redundancy and independence claims (5.3).

5.13 Preliminary summary

The previous sections focused on multiple aspects of the SORA and allow for an intermediate summary of the chances of UAVCAN presented within the **top-down approach**.

A weight reduction due to *shared functions* allow for a weight reduction even though the consequences for the City-ATM demonstrator are hard to determine without available components (*market research* (6.7)). The weight reduction may lower the kinetical energy of an UAS and so the *intrinsic GCR*. Assuming a demonstrator with the same functions but

equipped with an UAVCAN network, an operator could claim a medium level of robustness for the *M1 mitigation*. This would allow lowering the *residual GRC* by another point. The *M2 mitigation* is already rated medium due to the certificated DRS. Nevertheless the *shared functions* concept may increase the reliability and availability of the system above the minimum requirements which increases the operation efficiency.

As the *initial ARC* depends on the operations airspace, UAVCAN can "only" support by the increased *4D position accuracy* to define operations around higher ARC airspaces. UAVCAN supports all required functionalities for later operation within future UTM airspace. Future UTM concepts may allow an operator to lower the *initial ARC* by applying tactical mitigation. This does not apply to the City-ATM demonstrator, yet.

SAIL Determination				
	Residual ARC			
Final GRC	a	b	c	d
≤2	I	II	IV	VI
3	II	II	IV	VI
4	III	III	IV	VI
5	IV	IV	IV	VI
6	V	V	V	VI
7	VI	VI	VI	VII
>7	Certified category operation			

Table 5.3: SAIL reduction

Figure 5.3 shows that lowering the GRC does not affect the current SAIL of the City-ATM demonstrator (gray cell, red arrow). Nevertheless the current system functions - based on UAVCAN - allow for a higher intrinsic GRC while keeping the same SAIL of *II* (blue).

UAVCAN supports the interaction of components to claim a medium robustness level for the *TMPR*. This may allow in operator to operate within ARC-b airspace.

UAVCAN can only be partially applied to most OSOs. The OSOs with the largest impact of UAVCAN are

- **#02/#03** - UAVCAN supports the testing, calibration and maintenance process by high-level functions (register, node commands, ...)
- **Safe design (#05, #10, #12)** - The *shared functions* allow for safe system design that implements redundancies with independently developed components

- #**07** - The inspection and continuous monitoring of UAS components is supported by feedback of components, a self-test of the UAVCAN network and configuration checks before every flight

Many other OSOs are indirectly affected by UAVCAN as their assurance level require for flight tests or dedicated simulation tests within validated simulation. UAVCAN offers the chance for HITL simulation tests of component to validate correct functioning within a highly realistic scenario and interactions between multiple systems.

Even though the consequences for City-ATM demonstrator are not fundamental, operators of other UAS may benefit of a reduction of the *final GRC*. The presented ideas for requirement fullfillment of different OSOs may encourage a system operator to select UAVCAN as the intra-vehicular communication standard.

6 UAVCAN > SORA

The *bottom-up approach* summarises high-level functions of UAVCAN and derived functionalities that benefit the SORA, but are not yet covered by the City-ATM demonstrator as an example.

6.1 Redundant interfaces

UAVCAN enables redundant interfaces for interconnection of components. Redundant interfaces allow for higher system reliability as mechanical and electrical failures can be mitigated.

Damaged cables and defect connectors can cause mechanical failures. Especially vibrations and moving sections (e.g. tiltwing drones) can lead to mechanical damages of a bus system. Despite precautions and continuous maintenance such a single failure can occure. Electrical failures can be caused by potential electromagnetic interferences or a malfunction of CAN transceivers of connected components. The electrical malfunction can be temporary or permanent but is generally considered as a failure by the SORA.

The SORA states that single failures are not allowed to lead to operations outside the operational volume. Higher level of robustness even require the assessment of combinations of multiple failures. As *shared sensors*, monitors and components for mitigations are connected to the UAVCAN network, the network is part of the safety-critical failures that could lead to a failure outside the operational volume.

Operating a UAVCAN network with redundant interfaces (3.4.2) can mitigate the stated malfunctions. The network must be still considered as an operation-critical system but does not act as a single point of failure for the operation.

6.2 System monitoring

System monitoring and state estimations are essential for safely conducting an UAS operation. Flight controllers are depending on the proclaimed accuracy of sensor values and are essential for keeping control over an operation. The central placement of the FC in classical star architectures predestine the FC as the (sub-)system monitoring entity on an UAS. The FC itself already contains most of the UAS critical states.

Monitoring of modern, complex systems requires much more processing power as probably safe component states may interfere resulting in reduced range of functions. External, independent or distributed system monitors present a modern solution.

6.2.1 Failure detection

Several studies ([4], [28], [21]) show the enormous effort put into detection of anomalies of a UAS if the error source is unknown. Servos for control panes on fixed-wing, motors on multicopter and control rods can fail requiring immediate counter-measures to prevent a LOC or potential mitigation activation (if successfull depending on the flight state). KI and model predictive controllers are used to identify malfunctioning actuators and adapt the control algorithms. Sensor fusion algorithm can eliminate single event faulty measurements, but are distracted by continiuous faulty sensor inputs. All these problems are addressed by the recommendation and capabilities of UAVCAN to publish status information.

Sensors can continuously publish the estimated standard deviation of measurements. Actuators can report their current position and/ or power consumption. A reported servo position can helb the control algorithm applying correctly quantified counter-measures while inconsistency in the power consumption can indicate upcoming or existing loss of thrust.

All these status information not only free processing power required failure detection but enhance the counter-measures precision and response time.

6.2.2 (RT)LOLA

RTLola is real-time capable implementation of the stream-based specification language co-developed and part of the research objectives at DLR ([1]) *LOLA* can be used for runtime verfication and monitoring of systems based on data streams from sensors, etc.. *LOLA* allows abstract system descriptions beeing defined by a specification language. The formal system

description can contain information about the planned flight path, expected frequency of sensor values and flight envelope limitations. In context of the SORA *LOLA* supports the development of reliable system and flight envelope monitoring.

The idea behind *RTLola* is the execution of *LOLA* description on real-time capable (embedded) system. This hardware monitor should be "placed in a central position where as much sensor data as possible can be collected" [5]. UAS systems based on a bus system support this integration of a hardware monitor as all components are interconnected - including a (passive) monitor. The DSDL 3.3 simplifies the interface implementation for a monitor while the published sensor values, diagnostic and status message send by nodes are perfect inputs for the hardware monitor.

The stream-based monitor can publish its results on the UAVCAN network allowing other components to subsribe for. Mitigations, mission planning algorithms, etc. can rely on the status of the hardware monitor and react on detected incidents.

6.3 Configuration Management

Updating the software (or firmware for microcontrollers) is part of the essential configuration management requested by the SORA. OSO #3(5.11.1) and #7 request for a continous configuration management and checking the system before every flight.

6.3.1 Configuration Management Node

A part of the configuration management is ensuring the correct software configuration is running on the installed components. This is ensured by a configuration management process and continuosly documented during preparation for an operation. UAVCAN allows for another check as part of the pre-flight self-test. The *Software Update* 3.5.6 and *integration* 3.5.7 high-level functions defined the UAVCAN protocol allow for an immediate configuration check during the self-test of an UAS at power-up. UAVCAN nodes can display their configuration (registers) and can report the running software configuration. These information can be displayed in a GUI and reviewed by an operator. The human factor can be eliminated by introducing a **Configuration Management Node (CMN)** into the network. The *CMN* automatically request the aforesaid information of every node and compares the information with a permantly stored backup. Upon on completion of the integration test the system integrator once commands the *CMN* to permantly store the current

configuration. On power-on the *CMN* checks on connected nodes and publishes it node status and operational mode. Dependent functionalities (like further arming of the aircraft) can subscribe for the status of the **CMN** and wait for the confirmation of the configuration. Within a static configuration the *CMN* could check for the general existence of nodes and report nodes that were not expected (as far as they are visible and addresseable for CMN requests 3.5.3).

The *software update process* in cojunction with the *meta-transport 3.5.8* high-level function enables components to automatically check for available updates either on the vendors update server or propriatary servers specified by the system integrator. As the software update process can prevent the node from normal operation and the transmission of a software update requires free bus capacity, the concrete update should not be performed automatically. Nodes can inform and report available system updates but leave the decision to the operator/ integrator.

The previous paragraph describes the limiting conditions for a *software update* of a node. Even though the concrete update process is not explicitly mentioned by SORA it is part of the installation and maintenance tasks. Configuration management processes are described within documents but the software update process is enhanced by UAVCAN. Once installed and appraised components can stay at their mounting position while the software update is transmitted via the UAVCAN network. Maintenance personell can easily connect and check remote components by connecting to a diagnostic entry point to the UAVCAN network (e.g. a SLCAN adapter for UAVCAN/CAN).

6.3.2 Intelligent Power-On

The UAVCAN standard recommends a 4 wire connector between components that supplies a ground and VCC (e.g. 5 V) to connected components in addition to the *CAN high* and *CAN low* connections. After performing its self-test the Battery Management System (BMS) applies the VCC to its CAN connector. Further connected components power up and report the results of their self-test.

Next the **CMN** presented in the previous section determines the system configuration and report an error in case of not desired configurations or not defined system states. In this situation the BMS wont apply the full battery voltage to the high-voltage lines of the ESC or a starter motor. This prevents potential physical damage to the UAS and its environment.

6.4 Redundant flight controllers

The FC are essential for the planned operation success as they perform flight control and navigation tasks. Even though the SORA focusses on the worst-case scenario and allows mitigations that prevent a LOC, the planned mission must be suspended in case of a single FC. Redundant FC can increase the mission accomplishment rate and lower the robustness requirements for mitigations.

6.4.1 Cold redundancy

The easiest solution for redundant FC is the cold redundancy. Redundant FC are connected to the UAVCAN network and listen for heartbeats 3.5.3 by the primary FC.

The redundant FC continuously receives sensor information. As soon as a heartbeat timeout is detected that redundant FC starts its flight control algorithms, sends commands and stabilizes the UAS. The back-up FC then continuous the navigation and operation.

The cold redundancy has the disadvantage that a malfunctioning FC may continue to (irregular) send commands and a mode confusion is probable. Hand over the voting logic to the components addresses this uncertainity.

6.4.2 Nodes vote

Instead of setting the backup FC as the voter, the voting logic can be transferred to every single node. Similar to the time synchronization 3.5.5 function a primary and secondary FC (or even more) can be specified and classified by their unique node-ID (priority increases with node-ID). All FC are continuously listening to messages on the bus and the components respond to requests by all FC. Concrete commands (e.g. ESC commands) are filtered by the components and only accepted from the primary FC. The components therefore independently run a voting logic and Of course the voting logic is essential and possible mode confusion can occure in case of components (temporary) accepting different FC.

The described logic can be used in a hot and cold redundancy configuration (2.3). In case of a hot redundancy the secondary FC control and navigation algorithm must be resilient to permanent errors.

6.4.3 Single voter (APUS UAS)

Another system architecture with redundant FC is installed in the APUS flight test carier. The *APUS* is a fixed-wing aircraft designed for BVLOS missions and operated by the institute for flight systems at DLR, too. The aircraft is equipped with two FC that are the centers in two independent star architectures of sensors. The FC are connected to independent *C2 links* and enact as two fully independent flight control and navigation systems. A single voter receives (motor and servo) commands by both FC and forwards the signals of the primary FC to the motor controller and servos.

The advantage of two fully seperated systems comes at the cost of additional weight, installation and configuration complexity. Even though the command flow over the singular voter is system-dependent and unchangeable due to the ex factory base system, UAVCAN functions can reduce the stated disadvantages. The statelessness of simple components (like sensors) allow the publication of sensor value to multiple FC. *Shared sensors* between both FC can be easily implemented and integrated. Components could even respond to simple requests by the FCs while specific command modifying the node functionality should only be allowed to the primary FC.

The proposed design changes do not target the system dependent voter as a SPF, but supports the availability, reliability and *initial GRC* determination. It comes without the complexity of the previous *device voting* logic 6.4.2.

6.4.4 Independence

Common cause failures are often mitigated by independent development of redundant hardware and software. The tactic is known from high-reliability aircraft functions and can be adopted. Common interfaces are defined by standards and vendors can claim support of the standardized interface reaching out for new markets. In case of UAVCAN as a middleware e.g. Dronecode published standard message and service defininitions using the DSDL 3.3. Vendors are adopting the standard and label their indepently developed and manufactured devices as compatible to the Dronecode standard.

An integrator can compose a system with (redundant) components from multiple vendors claiming that independence reduces the risk of single point of failures. Unfortunatelly common open source abstraction layers could circumvent the independence argument and must be addressed by a system designer.

6.5 Virtual functions

Virtual functions are specialized form. They are called virtual as they are generally not bound to specific component. For the purpose of virtual functions combinations of subfunctions are suitable. On large UAS the position accuracy of two GNSS with correctional data may allow for a compass emulation. Therefore the devices must be classified into a primary and secondary receiver. During system configuration the primary GNSS receiver must be informed about the installation position of the secondary receiver and itself. In operation the primary GNSS receiver component can additionally calculate the attitude of the UAS and publish these information as *virtual sensor* data.

Another form of a *virtual sensor* may be the output of a sensor fusion algorithm. A highly rated sensor fusion algorithm can publish its fused sensor data (e.g. position estimation by the FC 5.1.3) and provide it to other components connected to the bus.

6.6 Mixed-criticality system

In context of the SORA and UAVCAN mixed-criticality can simplify the system design process. A subset of components running functions required for fail-safe operation are connected to the redundant UAVCAN interface. This subset of components connected to the *non-uniform* uavcan network 3.4.2 must match the requirements set by the SORA. This potentially involves higher development efforts and costs. The redundant connection of these components allows an operator the avoidance of SPF. In case of an failure of the secondary interface (connected to the fail-safe components) a system monitor must report the fail-safe status on the primary bus. On proclamation of the fail-safe status all non-fail-safe components must stop there communication to free the bus from non safety related traffic and reduce the risk of further failures.

Components that are not related to the subset of fail-safe functions are only connected via a single interface and can be developed to a lower assurance level. The set of non-fail-safe components may include payload computers, cameras or ranging sensors.

6.7 Market Research

As part of this work a market researach was conducted. The research should show if the proposed concepts and changes to the UAS avionics are technically feasible out of a system designer/ integrator perspective. As UAVCAN is a comparable new standard, the availibity of components can not be considered guaranteed.

Table D lists companies and their publicly stated UAVCAN compatible components. The first column contains the company name while the following two columns rate the companies targetted customer and the product level. Companies either directly stated their background and planned customer or it could be concluded from the ordering process, etc.. The columns *Product* contains one product per row. In case of products integrating multiple functionalities they are described within the same cell. The *status* columns presents the availability of listed components while the *version* column indicates the supported UAVCAN version (3.1.1). The *comment* columns contains not previously assigned information that nevertheless could effect the system design process.

The research shows that the number of **commercially available** components at a professional level roughly equals the number of hobbiest grade product. The majority of components at professional level focus on FC and GNSS/ compass components. Further sensors (barometers, range sensors, status lights) are not yet available as COTS. These sensors are partially available as commercially distributed but hobbiest grade products. The (continuous) support of these devices cannot be guaranteed and marks these products as rather irrelevant for professional UAS manufactures that must rely on reliable operation of and support for products.

Even though the different versions of UAVCAN can operate on the same concrete transport, the parallel operation comes at cost. Two network stacks must be maintained which can be critical for low-memory devices and wrappers must facilitate the differing message and service DSDL defintions. As the used UAVCAN version is defined by the software and not every vendor published this information, the information can be deduced based on claimed support and integration with other components. Even though the compatibility is defined by the software, updates on especially for hobbiest grade projects can not guaranteed. v1 supporting components are outnumbered by v0 compatible (hobbiest grade) components. In particular r̈areṗroducts (status lights, voltage monitors) do only support v0 yet.

The proposed concepts within this chapter, the *shared functionalities* (5.3) reference and requirements for assuring robustness levels of mitigations and OSOs state that SPF must

be avoided. Components and functions that are a SPF must be developed according to industrial avionics standards which are currently supported by none of the listed products. For lack of multiple products supporting the same functionalities, independence claims from a *common mode analysis* for redundantly installed components are hard to support. As the UAVCAN network can be part of the SPFs on a UAS, too, components should support multiple UAVCAN interfaces. The *comment* columns presents that only a minority of the listed products are equipped with a redundant *CAN* interface.

As the market analysis shows up blank spaces in commercially available products and functionalities, hobbiest projects and open-source communities have adapted the UAVCAN standard. Small hobbiest projects like (UC4H, [15]) lower the initial hurdle for UAVCAN development while open-source communities like ArduPilot and PX4 provide full *hardware abstraction layers* that simplify the development process. The abstraction layers unite the communication interfaces of e.g. a FC and provide the access to concrete UAVCAN implementation (e.g. one of the reference implementations 3.6). The *abstraction layers* are designed for specific hardware defines by a company, consortium or (open-source) community. As far as the protecting license of the software includes the desired use case, a developer may base the new function on an existing HAL. Standalone reference applications like the *libcanard* (3.6) may be better suited for resource-constrained and/ or high criticality applications that require for a detailed review and testing of the code.

In addition to reference designs for FC published by the open-source communities, hardware manufactures start publishing reference designs. Companies like NXP provide hardware manufactured at industry level and encourages software developers to participate in the software development process and provide feedback on the hardware design. NXP provides reference designs for BMS, FC and development kits. In particular the BMS closes a gap of available functionalities as shown in the market research.

7 Concluding remarks and outlook

The following sections summarise the results of this work and phrases remaining questions. The outlook presents future applicability of presented concepts and chances of the UAVCAN protocol.

7.1 Conclusion

The preliminary summary for the *top-down approach* already stated the advantages and effects of UAVCAN on the SORA for the City-ATM demonstrator. Even though the effects on the SAIL determination of the demonstrator are small, it shows the influence of UAVCAN on the SORA. In particular the OSOs aiming at safe design and configuration of the UAS are affected by UAVCAN.

For the *safe design* OSOs referenced standards for detailed system analysis were deduced from the AW Drones project. The system analyses located weak points of the current demonstrator configuration assuming requirements of higher SAIL operations. As the available reliability information corresponds to the small number of available devices, a qualitative analysis has been conducted that is focusing on SPF. This work describes multiple concepts for avoiding SPF by physical and functional redundancy. Physical redundancy can be accomplished by redundant UAVCAN networks and multiple, indepent components. *Shared functions* and *virtual functions* concepts presented solution for implementing functional redundancies within an UAVCAN network.

The *bottom-up approach* shows that further advantages of the UAVCAN high-level functions only arise in more complex systems. Nevertheless the *shared functions* concept can increase the system availability immediately and may enhance the operation by remaining in a fail-operational instead in case of sensor failures. The *mixed-criticality* section describes the chances for integration of mixed-criticality components within one (redundant) UAVCAN network.

The configuration and inspection requirements of the SORA are addressed by the new *configuration management node* concept and an intelligent power-on of devices connected to the bus and high-voltage circuit. In combination with presented *system monitoring* concepts the UAS can be checked and monitored during all flight phases.

The conducted analyses and proposed concepts show that UAVCAN is not only applicable on small UAS. UAVCAN has the most effect when it comes to scalable, complex system designs that require for safe design and system monitoring tasks.

7.2 Outlook

As UTM services are not available yet the analysis for TMPR requirements must be conducted in the future. As the concrete technique is determined, a review of the mitigation shows the effects of a connecting the UTM interface to the UAVCAN network. UAVCAN will support the reliability of UTM service status reports.

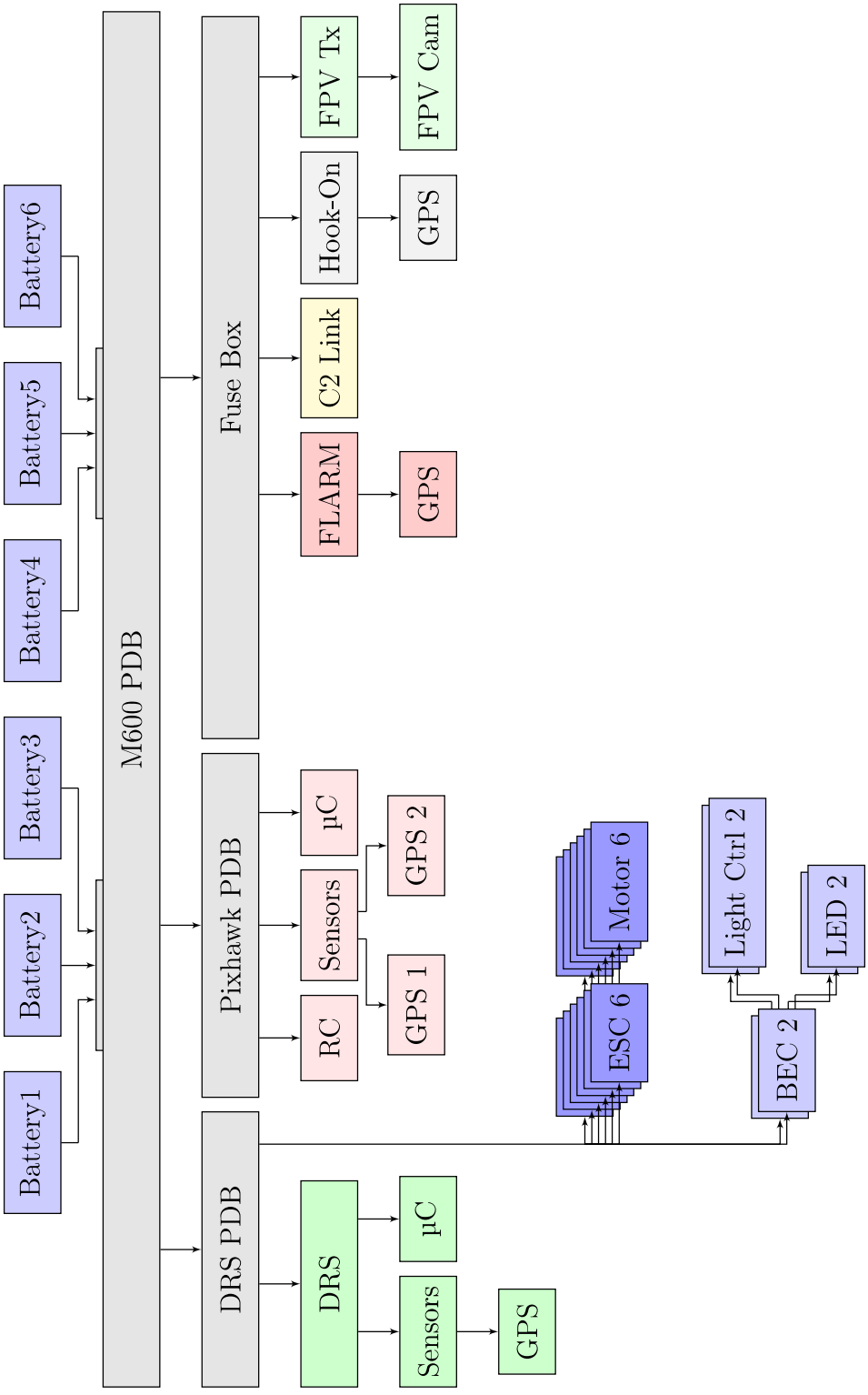
In the future more UAVCAN capable devices will be available. It must be noted that devices should be equipped with redundant CAN interfaces to support the independence claims and mitigations of SPF. More independent components with the same functionality will support independence claims and port assumptions about redundancy concepts into the real world.

Within the *bottom-up approach* this work presents multiple concepts that affect the SORA but are not yet realised for existing UAS. *Configuration management nodes*, *virtual sensor* services and *HITL simulations* are still open tasks that require for UAVCAN service definitions and hardware and software development. The *HITL simulation* support development and integration tasks. Therefore the integration of an UAVCAN interface into existing - validated - simulations must be examined and implemented

Currently the UAVCAN standard is limited by the available systems complexity. With more UAVCAN capable components available the UAVCAN standard will simplify the integration of new components and support system designers matching the requirements stated by the SORA for operations of specific category UAS.

A Power Flow

Figure A.1: Power Flow: M600



B Functional Hazard Analysis

Table B.1: Failure Hazard Analysis: Multicopter

No.	Function	Failure Condition	Flight Phase	Effect
1	Detect and Avoid			
1.1		FLARM GPS malfunction	any	No/ Misleading traffic detection
1.2		FLARM GPS inaccurate	any	Traffic information cannot be used within PixhawkAutomatic Traffic Avoidance not possible
1.3		FLARM device malfunction	any	No/ Misleading traffic detection
1.4		FLARM electrical failure (e.g. shortcut)	in-flight	No traffic detectionMissing status detected - mission abortOther systems protected by fuse
1.5		Pixhawk navigation solution inaccurate	in-flight	Pixhawk traffic avoidance not functioningSynchronization with FLARM data impossible
1.6		Pixhawk Traffic Avoidance fails	in-flight	Mid-Air crash possible
1.7		Pixhawk Telemetry fails	in-flight	Traffic not displayed for operatorFail-Safe triggered - mission abort
1.8		SKIDER Telemetry fails	in-flight	Traffic not displayed for operatorFail-Safe triggered - mission abort
2	Anti-Collision Lighting			
2.1		LED Strip malfunction	in-flight	Visibility of drone reduced
2.2		LED controller malfunction	in-flight	Visibility of drone reduced
2.3		Power failure	in-flight	Visibility of drone reduced
3	Flight Termination/ Rescue System			
3.1		Pixhawk NavSol inaccurate (SW)	any	false-posive geofence (violation)
3.2		DRS malfunction	in-flight	Parachute opening without stopped motors
3.3		DRS Navigation Solution inaccurate	in-flight	Treshhold height for parachute
3.4		DRS Navigation Solution inaccurate	in-flight	false-positive geofence violationimmediate DRS activation/ Crash
3.5		DRS Navigation Solution inaccurate	in-flight	geofence violation not detected
3.6		DRS GPS malfunction		DRS navigation solution inaccurate
4	System Power			
4.1		DJI PDB fails (VR 44.4V)	in-flight	Pixhawk and ESCs not powered-dActivation of DRSCrash
4.2		DJI PDB fails (VR 18V)	in-flight	DAA failure (FLARM)C2 link/ mission control failure (SKIDER Link)Anti-Collision Lighting failure-FPV system failure
4.3		DRS PDB fails	in-flight	Propulsion failureActivation of DRS
4.4		Pixhawk PDB fails	in-flight	Activation of DRSLoss of flight control
4.5		Fuse Box failure	in-flight	Payload devices not powered (FLARM, SKIDER, ...)
4.6		LED BEC failure	in-flight	Visibility of drone reduced
4.7		Power Supply FLARM fails	in-flight	DAA failsmission abort
5	Propulsion			
5.1		Motor failure	in-flight	Loss of flight control1(2) out 6 redundancy

Continued on next page

Table B.1 Continued from previous page

5.2		ESC failure	in-flight	Loss of flight control1(2) out 6 redundancy
5.3		DRS PDB Failure	in-flight	Loss of flight controlDRS activation
5.4		DJI PDB Failure	in-flight	DRS activation
6	Flight Control			
6.1		Pixhawk NavSol inaccurate (SW)	any	Loss of ControlDRS activated (failsafe)
6.2		Pixhawk GNSS Failure	any	Pixhawk NavSol inaccurate1 out 2 hot redundancy
6.3		Pixhawk IMU failure	any	Pixhawk NavSol inaccurateLoss of flight control
6.4		Pixhawk Barometer failure	any	Pixhawk NavSol inaccurate in z-AxisFallback to inaccurate GNSS height
6.5		Pixhawk Compass failure	any	Loss of Control1 out of 3 redundancy
7	Mission Control			
7.1		Pixhawk NavSol inaccurate (SW)	any	Flight path outside geofenceDRS activated (failsafe)
7.2		Pixhawk GNSS Failure	any	Pixhawk NavSol inaccurate1 out 2 hot redundancy
7.3		SKIDER Link failure	pre-flight	mission upload impossible
7.4		SKIDER Link failure	in-flight	mission status control impossible-failsafe triggered - immediate land
8	System Status Monitoring			
8.1		Pixhawk malfunction: irresponsive	in-flight	DRS activated
8.2		SKIDER malfunction	any	Loss of communicationSensors monitoring impossibleMission surveillance impossibleDAA impossiblefailsafe triggered
8.3		Smart Battery/ Batteries malfunction	pre-flight	System not powering up
8.4		Smart Battery/ Batteries malfunction	pre-flight	Intransparent loss of energy

C Failure Tree Analysis

Figure C.1: FTA: M600 DAA

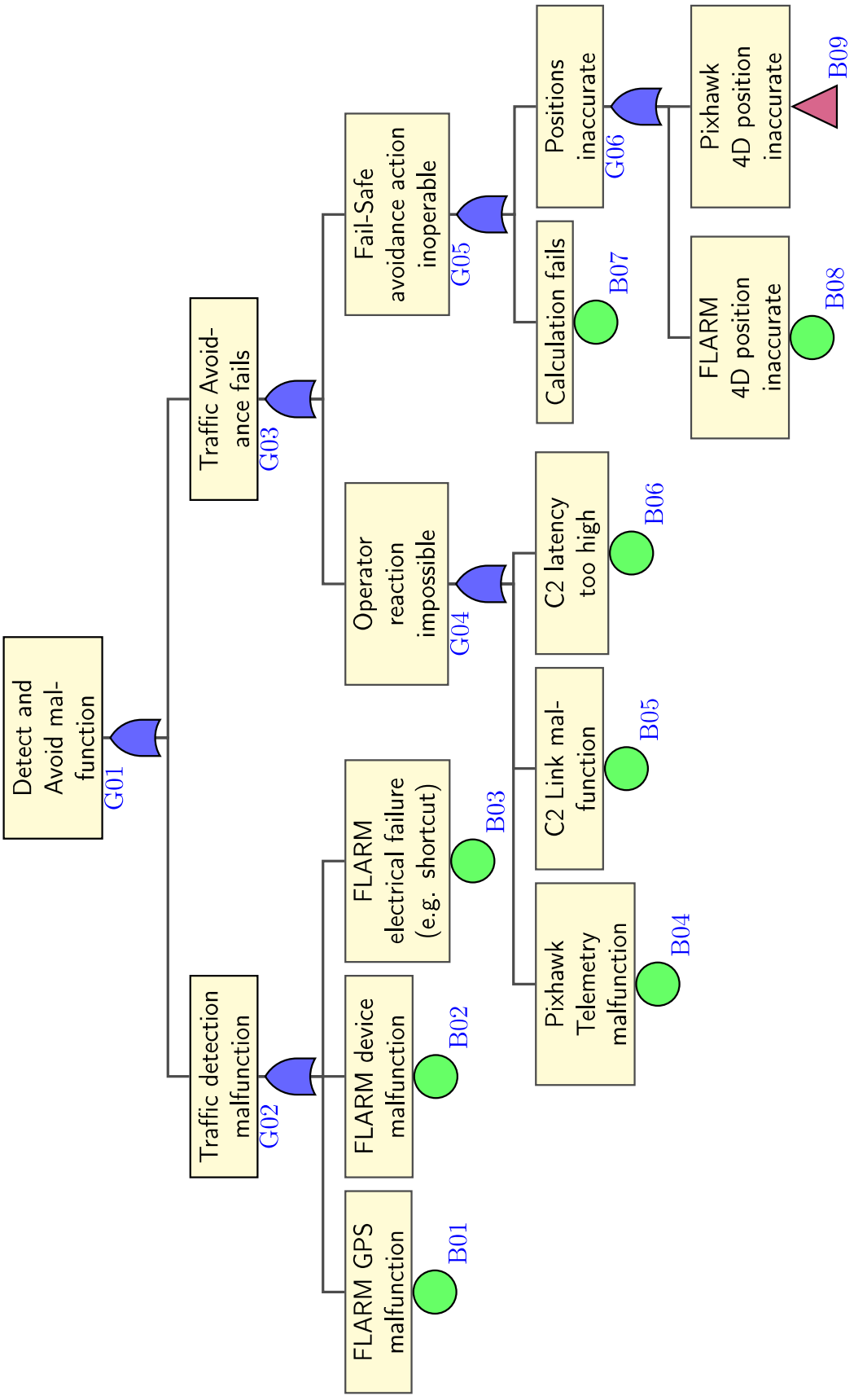


Figure C.2: FTA: M600 M2 mitigation

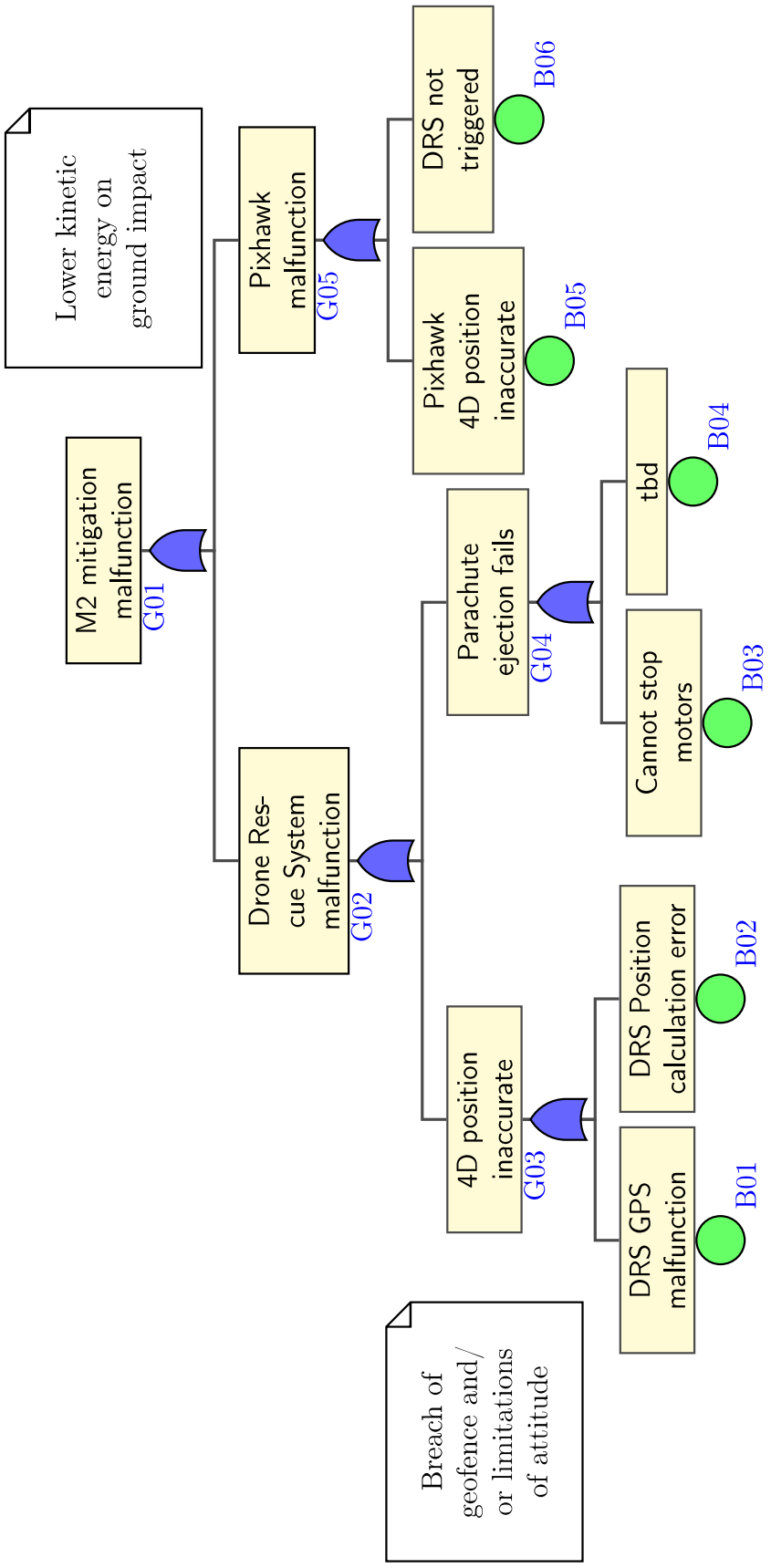
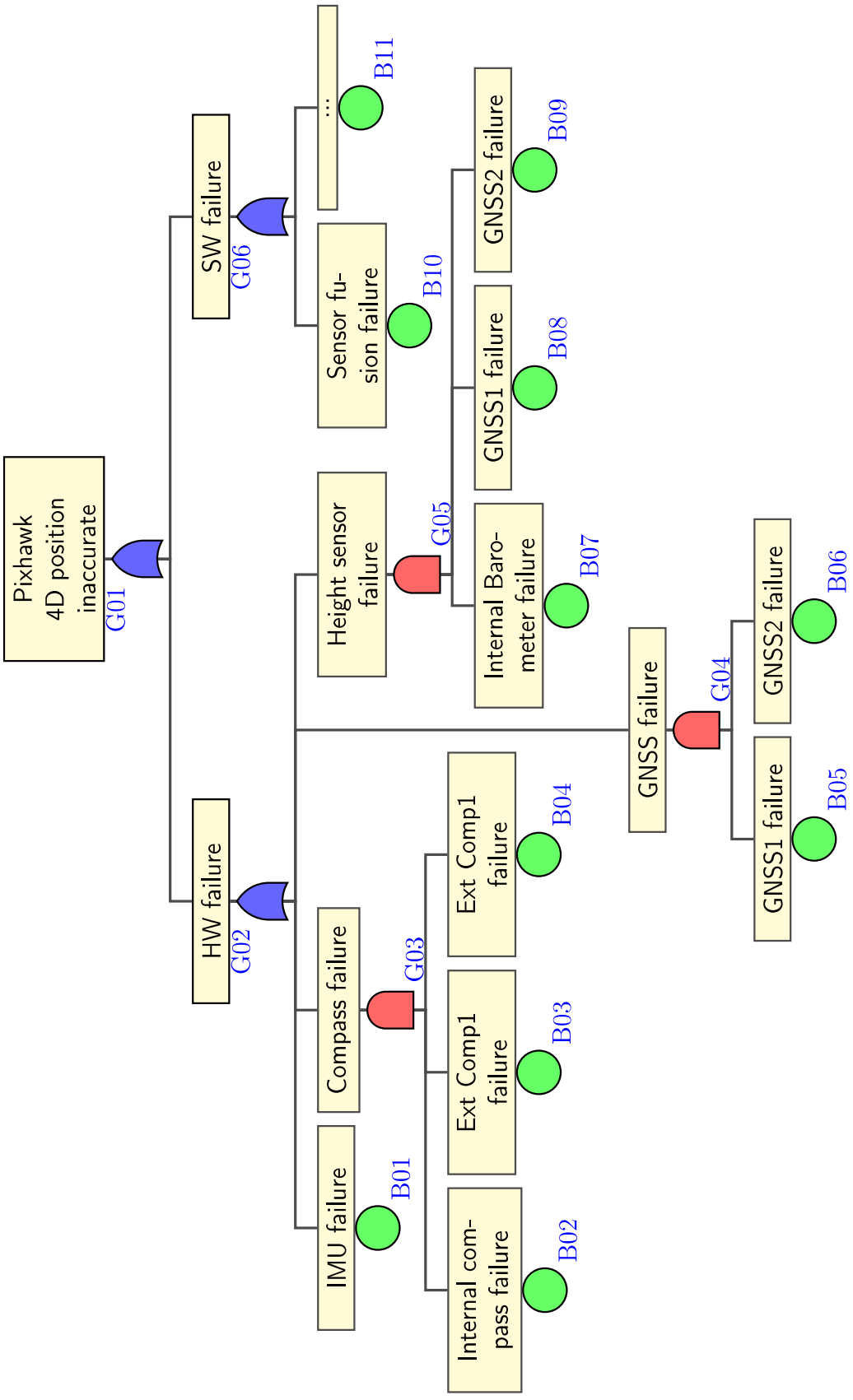


Figure C.3: FTA: M600 Pixhawk 4D positon



D Market Research

Table D.1: UAVCAN: Market analysis

Company name	Sector	Grade	Product	Status	Version	Comments
Avionics Anonymous	consumer	hobby	Light Controller	available	v0	Displays PX4 controller statesSingle CAN-connection (Daisy-Chain possible)
Avionics Anonymous	consumer	hobby	Engine Monitor (Adapter)	coming soon	v0	Single CAN-connection (Daisy-chain possible)External sensors connectedAP Integration work necessary
Avionics Anonymous	consumer	hobby	Voltage Monitor	coming soon	v0	Single CAN-connection (Daisy-chain possible)Monitors up to 8 analog voltage inputs sharing comon groundPublishes CircuitStatus message
Avionics Anonymous	consumer	hobby	Micro ADC (Air Data Computer)	available	v0	Single CAN-conncetion (Daisy chain possible)Published fused and raw dataPX4 works better with raw data (own sensor fusion)
Avionics Anonymous	consumer	hobby	GNSSMagnetometer	available	v0	Single CAN-conncetion (Daisy chain possible)high accuracy magnetometer (0.5 degree)GPS Pulse-Per-Second Output
Avionics Anonymous	consumer	hobby	Magnetometer (high precision)	available	v0	Single CAN-connection (daisy chain possible)mini and maxi versions
Avionics Anonymous	consumer	hobby	Laser InterfaceRange	available	v0	Single CAN-connection (daisy chain possible)Interface for Laser Range sensors
CUAV	commercial	pro	GPS, Status Light, Safety Switch	available	n.a.	
CUAV	commercial	pro	Power Module	available	n.a.	
CUAV	commercial	pro	Pixhawk FMUv5 controller	available	n.a.	
CUAV	commercial	hobby	Power Module	available	n.a.	
CUAV	commercial	pro	Power ModuleVoltage Regulator	available	n.a.	Single CAN-interface
Currawong	commercialmilitary	pro	ESCVelocity ESC	on request	n.a.	Has CAN interfaceprobably single CAN only
Currawong	commercialmilitary	pro	Servo	on request	n.a.	
HiTec	industrial		Servo	available	v0	Probably v0 (due to claimed ArduPilot support)
HiTex	industrial	pro	ESC	on request	v0	on request UAVCAN capable ESC
Holybro	consumer		ESC	available	v0 (v1 open)	single CAN, daisy chain possibleKotleta ESCdepends on Sapog FW
Holybro	pro	pro	Flight Controller	available	v0/ v1	STM32 H7 processorCAN FD transceiver to be questioned
MatekSys	commercial	hobby	GPSCompass	available	v0	Single CAN-connection
MatekSys	commercial	hobby	BarometerAirSpeed	available	v0	Single CAN-connection
MatekSys	commercial	hobby	Flight ControllerArduPilot StackPixhawk FMUv5 controller	available		only Single CAN-connection
mRobotics	commercialconsumer	hobby	Flight Controller APM and PX4	available	n.a.	redundant CANH743 (only APM) and F765 (PX4) available
mRobotics	consumer	exp	UAVCAN Adapter Board	out of stock	n.a.	redundant CANAirSpeed, GPS, Compass, I2CArdupilot HAL

Continued on next page

Table D.1 Continued from previous page

mRobotics	consumer	consumer	GPSCompassBarometerSafety SwitchStatus Light	available	n.a.	Single CAN-interface
mRobotics	commercial	hobby	Adapter Board	available	n.a.	Single CAN-interface
NXP	commercial	exp	Battery management system (reference design)	available	v0/ v1	Single CAN-interface
NXP	commercial	exp	development/ adapter board	available	v0/ v1	Redundant CAN-interfaceCAN-FD support
NXP	commercial	exp	Flight controller	available	v0/ v1	
Pomegranate Systems	consumer	hobby	Power Module	n.a.	v0	single can (daisy chain possible)
Pomegranate Systems	consumer	hobby	landing gear controller	n.a.	v0	single can (daisy chain possible)no docs and code found
Zubax	commercialconsumer	pro	ESC	available	v0/ v1	Redundant CAN
Zubax	commercialconsumer	pro	GNSS	available	n.a.	redundant CAN
Zubax	commercialconsumer	pro	Integrated Drive (Motor + ESC)	n.a.	n.a.	

Bibliography

[1] Adolf, F., Faymonville, P., Finkbeiner, B., Schirmer, S. & Torens, C. *Stream Runtime Monitoring on UAS*. 6th Sept. 2017. 33 pp.

[2] Andrade, R. D., Hodel, K. N., Justo, J. F., Laganá, A. M., Santos, M. M. & Gu, Z. 'Analytical and Experimental Performance Evaluations of CAN-FD Bus'. In: *IEEE Access* 6 (2018). Pp. 21287–21295.

[3] ASTM. *F3309 Standard Practice for Simplified Safety Assessment of Systems and Equipment in Small Aircraft*.

[4] Baskaya, E., Bronz, M. & Delahaye, D. 'Fault detection diagnosis for small UAVs via machine learning'. In: *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*. 2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC). Sept. 2017, pp. 1–6.

[5] Baumeister, J., Finkbeiner, B., Schirmer, S., Schwenger, M. & Torens, C. 'RTLola Cleared for Take-Off: Monitoring Autonomous Aircraft'. In: 14th July 2020, pp. 28–39.

[6] betaflight. *betaflight.com - Home*. URL: `https://betaflight.com/`.

[7] *CAN in Automation (CiA): CAN FD - The basic idea*. URL: `https://www.can-cia.org/can-knowledge/can/can-fd/`.

[8] *CAN in Automation (CiA): CAN knowledge*. URL: `https://www.can-cia.org/can-knowledge/`.

[9] *Controller Area Network*. In: *Wikipedia*. 1st May 2021. URL: `https://de.wikipedia.org/w/index.php?title=Controller_Area_Network&oldid=211489451`.

[10] Davis, R. I., Burns, A., Bril, R. J. & Lukkien, J. J. 'Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised'. In: *Real-Time Systems* 35.3 (1st Apr. 2007). Pp. 239–272. URL: `https://link.springer.com/article/10.1007/s11241-007-9012-7`.

[11] *iNavFlight/inav*. 18th July 2021. URL: `https://github.com/iNavFlight/inav`.

[12] JARUS. *JARUS guidelines on Specific Operations Risk Assessment v2.0*. 30th Jan. 2019.

[13] *Matrice 600 Serie Stromverteilerplatte - DJI Mobile Online Store (Deutschland)*. URL: https://m.dji.com/de/product/matrice-600-series-power-distribution-board.

[14] Michalik, H. *Vorlesungsskript - Entwurf fehlertoleranter Rechnersysteme*. 22nd Oct. 2007.

[15] *OlliW's Bastelseiten » UC4H: UAVCAN for Hobbyists*. URL: http://www.olliw.eu/2017/uavcan-for-hobbyists/.

[16] *Open Source Autopilot for Drones*. PX4 Autopilot. URL: https://px4.io/.

[17] *Open Source Drone Software. Versatile, Trusted, Open. ArduPilot*. URL: https://ardupilot.org/.

[18] OpenUAVAdmin. *UAS-Klassifizierungen – LBA – OpenUAV*. URL: https://lba-openuav.de/onlinekurs/lehrmaterial/luftrecht-und-sicherheit/uas-klassifizierungen

[19] *Pixhawk 4 | PX4 User Guide*. URL: https://docs.px4.io/master/en/flight_controller/pixhawk4.html.

[20] *PowerFLARM Eagle | flarm-eagle*. LX navigation. URL: https://www.lxnavigation.com/powerflarm-eagle/flarm-eagle.

[21] Rot, A., Hasan, A. & Manoonpong, P. *Robust Actuator Fault Diagnosis Algorithm for Autonomous Hexacopter UAVs*. 16th June 2020.

[22] SAE. *ARP4761 - GUIDELINES AND METHODS FOR CONDUCTING THE SAFETY ASSESSMENT PROCESS ON CIVIL AIRBORNE SYSTEMS AND EQUIPMENT*. 1st Dec. 1996.

[23] *The Dronecode Foundation - We are setting the standards in the drone industry with open-source - Join the Community!* Dronecode Foundation. URL: https://www.dronecode.org/.

[24] *UAVCAN - CAN bus for UAV*. diydrones. 23rd Jan. 2014. URL: https://diydrones.com/profiles/blogs/uavcan-can-bus-for-uav.

[25] *UAVCAN v1 beta*. 11th Nov. 2020.

[26] Viebahn, H. von. *Vorlesungsskript - Avioniksysteme*. 2017.

[27] wp_2206893. *Home page*. AW-Drones. URL: https://www.aw-drones.eu/.

[28] Zhang, X., Zhang, Y., Su, C.-Y. & Feng, Y. *Fault-Tolerant Control for Quadrotor UAV via Backstepping Approach*. 4th Jan. 2010.

We cannot solve our problems with the same thinking we used when we created them.

Albert Einstein

I've also prepared a safety briefing for you to entirely ignore.

J.A.R.V.I.S (AI system from the movie *Iron Man*)