SOFTWAREARCHITEKTUR FÜR EINEN UNBEMANNTEN LUFTFRACHTTRANSPORTDEMONSTRATOR

S. Benders, L. Goormann, S. Lorenz, J. C. Dauer,
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), Institut für Flugsystemtechnik,
Lilienthalplatz 7, 38108 Braunschweig, Deutschland

Zusammenfassung

Mit voranschreitender Technologiereife rückt der automatisierte Frachttransport mithilfe unbemannter Luftfahrzeuge in greifbare Nähe. Dennoch sind noch einige Fragestellungen unbeantwortet und durch die Erprobung zu beantworten. Zu diesem Zweck wird am Deutschen Zentrum für Luft- und Raumfahrt der Automated Low Altitude Air Delivery Demonstrator aufgebaut und betrieben. Dieser Beitrag beschreibt einleitend das Gesamtsystem und geht im Speziellen auf die bordseitige Softwarearchitektur ein. Besonderheiten der Softwarearchitektur bei der Entwicklung des Prototyps werden beschrieben. Unter anderem werden die Auswirkungen einer risikobasierten Nachweisführung des Systems auf die Softwarearchitektur und sich daraus ergebende Möglichkeiten erläutert.

1. EINLEITUNG

Unbemannte Luftfahrzeuge werden für den automatisierten Frachttransport immer attraktiver. Technologien im Zusammenhang mit tiefliegendem automatischem Lufttransport, werden am Deutschen Zentrum für Luft- und Raumfahrt (DLR) in den Projekten Automated Low Altitude Air Delivery (ALAADy) und ALAADy-Demonstrator gebündelt. Ziel ist eine Technologiedemonstration für ein innovatives, kostengünstiges Konzept zum Frachttransport mit bis zu einer Tonne Nutzlast. Die Entwicklung des ALAADy-Demonstrators orientiert sich an den zukünftigen Regeln für die Zertifizierung von unbemannten Luftfahrzeugsystemen (Unmanned Aircraft Systems, UAS) für den zivilen Luftraum. Die Sicherheitsbewertung basiert dabei auf einem missionszentrierten Ansatz, der die aus dem Betriebsszenario resultierenden Risiken berücksichtigt, um das erforderliche Sicherheitsniveau zu bestimmen. Je niedriger das Risiko ist, dass ein Kontrollverlust zu Todesfällen oder erheblichen Sachschäden führt, desto höher ist das erreichte Sicherheitsniveau.

Dieser Beitrag beginnt mit einem Überblick über das Projekt und den Technologiedemonstrator. Nach einer kurzen Beschreibung der Avionik wird die bordseitige Softwarearchitektur vorgestellt. Der Beitrag schließt mit einer Zusammenfassung und einem Ausblick auf die nächsten Schritte des Projektes ALAADy-Demonstrator.

2. ALAADY-DEMONSTRATOR

Im Jahr 2016 startete am DLR das Forschungsprojekt ALAADy, welches die Erforschung von Technologien und der Anwendung neuartiger Regularien für den unbemannten, automatisierten Luftfrachttransport in geringer Flughöhe zum Ziel hat. Im Fokus des Projektes stehen insbesondere Sicherheitsaspekte bei der Nutzung unbemannter Luftfahrzeuge für den Luftfrachttransport [1].

BILD 1 Unbemannter ALAADy-Demonstrator

Das Projekt ALAADy untersucht dazu mögliche Luftfahrzeugkonfigurationen, Einsatzbedingungen und -grenzen, sowie Systeme, einschließlich der Software und Hardware, um den sicheren Betrieb eines automatisierten unbemannten Frachtflugzeugs zu gewährleisten.

Um die gesammelten Erkenntnisse im realen Flug zu demonstrieren und zu validieren, wird am DLR parallel zum Projekt ALAADy ein entsprechendes unbemanntes Luftfahrzeug aufgebaut, der so genannte ALAADy-Demonstrator (siehe BILD 1). Als Basisluftfahrzeug für den ALAADy-Demonstrator wurde ein Tragschrauber vom Typ AutoGyro MTOfree gewählt. Die Sitze und das Cockpit wurden entfernt, um die Nutzlast, die Akkumulatoren, die Bordcomputer und die Navigationsausrüstung sowie ein Flugabbruchsystem und Aktuatoren aufzunehmen.

Nach den beschriebenen Modifikationen kann dieses Luftfahrzeug eine Nutzlast von bis zu 200 kg tragen. Weitere technische Kenndaten zeigt TAB 1.

doi:10.25967/480238

TAB 1 Technische Daten des ALAADy-Demonstrators

Länge:	5,08 m
Breite:	1,88 m
Höhe:	2,71 m
Rotor Durchmesser:	8,4 m
max. Nutzlast:	200 kg
MTOW:	500 kg
max. Flugzeit	3 h

Basierend auf dem Luftfahrzeug und dem Einsatzkonzept wird die spezifische Risikobewertung (Specific Operations Risk Assessment, SORA) [2] demonstriert und angewendet, um das Luftfahrzeug zu betreiben. Die Anwendung von SORA wird durch die Veröffentlichung eines Konzepts für den Einsatz von Drohnen durch die Europäische Agentur für Flugsicherheit (EASA) im Jahr 2015 ermöglicht [3]. Der wesentliche Paradigmenwechsel bei diesem Ansatz besteht darin, dass der Absturz oder Unfall des unbemannten Flugzeugs nicht länger als eine Gefahr betrachtet wird, die verhindert werden muss. Stattdessen sind nur drei Sicherheitsrisiken zu berücksichtigen: Kollision in der Luft mit bemannten Flugzeugen, sowie Personen- und Sachschäden am Boden. Das Risiko einer Kollision in der Luft mit bemannten Luftfahrzeugen wird durch den Betrieb des Luftfahrzeugs in sehr niedrigen Höhen unterhalb des kontrollierten Luftraums erheblich verringert. Schäden an Personen und Eigentum werden durch die Installation eines Flugabbruchsystems gemindert, welches alle Steuereingaben im Falle eines Kontrollverlustes mit vorab definierten Werten überschreibt und das Luftfahrzeug kontrolliert und räumlich begrenzt zu Boden bringt. Durch die Wahl von Flugrouten über unbesiedeltes Gebiet werden die Sicherheitsrisiken ausreichend gering gehalten. Die Nachweisführung kann sich demnach auf ein zuverlässiges Flugabbruchsystem beschränken, welches im Falle des Verlassens der geplanten Flugroute ausgelöst wird. Ein zuverlässiges und robustes System zur Flugdurchführung erhöht die Wirtschaftlichkeit, kann in der Nachweisführung jedoch vernachlässigt werden.

3. AVIONIKSYSTEM

Nach einem Überblick über das Luftfahrzeuges und das Betriebskonzept, soll in diesem Abschnitt speziell auf das Avioniksystem eingegangen werden.

Obwohl das Luftfahrzeug autonom durch bordseitige Systeme betrieben werden soll, ist es bei einem Experimentalsystem sinnvoll alternative Steuermöglichkeiten vorzusehen. Aus diesem Grund gibt es zum einen die Möglichkeit das Luftfahrzeug vollständig manuell über eine Funkfernsteuerung zu steuern und zum anderen die Möglichkeit einen kontrollierten Flugabbruch herbeizuführen.

Die Systeme zur automatischen und manuellen Steuerung werden im Folgenden beschrieben. BILD 2 zeigt schematisch das Avioniksystem. Die Avionik wird von zwei zentralen Bordcomputern verwaltet. Die Bordcomputer sind baugleich, erfüllen jedoch unterschiedliche Aufgaben unterschiedlicher Kritikalität. Der Core Interface Computer (CIC) übernimmt die Ansteuerung der Subsysteme des Luftfahrzeugs, während der Flight Control Computer (FCC) der automatischen Flugsteuerung, der Ausführung von experimenteller Software und zur Verbindung mit der Bodenkontrollstation (Ground Control Station, GCS) über den Datenlink (Command & Control, C2) dient.

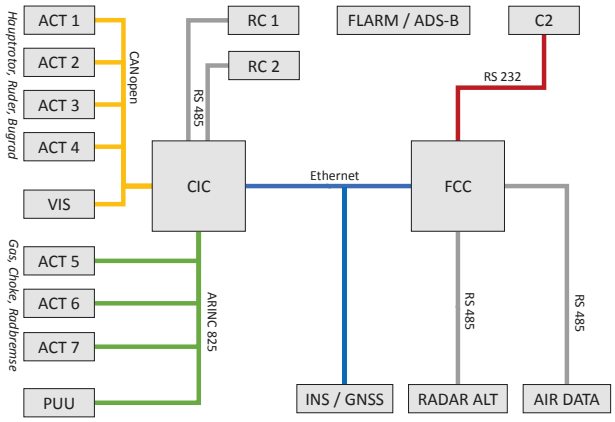

BILD 2 Avioniksystem (vereinfachte Darstellung)

Die Entscheidung zur verteilten Rechnerarchitektur wurde primär aus Gründen der Modularität und Partitionierung gefällt. Der ALAADy-Demonstrator ist ein Prototyp und wird daher in Teilen mit experimenteller Software betrieben. Die verteilte Rechnerarchitektur unterstützt an dieser Stelle die Partitionierung der Software [4]. Fehlfunktionen in der experimentellen Software auf dem FCC beeinträchtigen so beispielsweise nicht die Möglichkeit zur manuellen Steuerung des Luftfahrzeuges auf dem CIC. Auch die Software auf den Rechnern ist modular ausgelegt. Diese Modularität der Rechner- und Softwarearchitektur erlaubt neben einer guten Testbarkeit, eine sequentielle und unabhängige Entwicklung der Software auf den beiden Bordcomputern. Entwicklungen an der automatischen Flugsteuerung auf dem FCC erfordern beispielsweise keine Änderungen an bereits erprobter Software auf dem CIC. Sowohl Software als auch Hardware können so ohne großen Aufwand ausgetauscht werden. Zusätzlich wird so die Möglichkeit geschaffen einzelne für die Betriebserlaubnis nach SORA erforderlichen Funktionen, etwa ein automatisches Geofencing, auf einen separaten Rechner auszulagern, um so den Umfang der Nachweisführung zu begrenzen. Die beiden Computer und deren Funktionen werden in Abschnitt 3.1 näher beschrieben.

Zusätzlich zu den Bordcomputern sind, wie in BILD 2 gezeigt, Aktuatoren (ACT 1-4) zur Ansteuerung der Hauptsteuerflächen (Seitenruder, Rotor und Bugrad), sowie Aktuatoren (ACT 5-7) zur Ansteuerung des Gaszuges, des Chokes und der Radbremse verbaut [4]. Der Instrumentenpilz des bemannten Tragschraubers wird durch das Vehicle Interface System (VIS) ersetzt. Das VIS steuert beispielsweise die Vorrotation, die Magnete und den Anlasser und interpretiert die Daten diverser Sensoren des Basisluftfahrzeuges, wie beispielsweise die Motordrehzahl- oder die Öltemperatursensoren. Die Power Utility Unit (PUU) verwaltet die Stromversorgung und überwacht die Akkumulatoren. Zwei redundante Funkfernsteuerungsempfänger (Remonte Control, RC) bieten die Möglichkeit zur manuellen Steuerung des Luftfahrzeuges über eine RC-Fernsteuerung. Darüber hinaus sind eine Trägheitsnavigationsplattform mit Satellitennavigation (INS / GNSS), ein Radarhöhenmesser (RADAR ALT) und ein Luftdatensystem (AIR DATA) in Form eines Nasenmastes verbaut. Der Kommunikation zur Bodenstation dient der C2 Datenlink. Darüber hinaus ist ein kombiniertes FLARM / ADS-B System verbaut. Ein Flugabbruchssystem dient als verlässliche Einrichtung, um im Notfall einen kontrollierten Flugabbruch herbeizuführen (siehe Abschnitt 3.2).

BILD 3 Bedienpanel am ALAADy-Demonstrator mit Sicherheitssteckern

Um einen sicheren Laborbetrieb des Luftfahrzeuges zu gewährleisten, wurden vier Sicherheitsstecker (Flight, Red, Yellow, Green) definiert, die zum Betrieb an das System gesteckt werden müssen (siehe BILD 3). Je nach gewähltem Sicherheitsstecker werden bestimmte Funktionen (z.B. der Anlasser) des Systems gesperrt, sodass die Verletzungsgefahr durch Fehlbedienung im Labor gemindert wird.

3.1. Bordcomputer

Die baugleichen Bordcomputer FCC und CIC sind als PC104 Systeme ausgeführt. Die Computer verfügen über 2 x 2,2 GHz Prozessoren, 4 GB RAM, eine SSD Festplatte. Die Bordcomputer sind lüfterlos und besitzen somit keine beweglichen Teile. Für die Kommunikation mit den Avioniksystemen sind RS232, RS485, RS422, Ethernet-LAN und CAN-Schnittstellen vorgesehen. Auf beiden Bordcomputer kommt das Echtzeitbetriebssystem QNX6.6.0 [5] zum Einsatz.

Durch die baugleichen Rechner wird die Möglichkeit gewahrt, den Ort der Ausführung der verschiedenen Softwarekomponenten auch noch nachträglich zu ändern. Diese Möglichkeit ist bei dem Experimentalsystem ausdrücklich gewünscht und die verteilte Rechnerarchitektur unterscheidet sich zu anderen Systemarchitekturen [6].

Ein Ausfall des CIC hat den Verlust der Steuerautorität zur Folge. Daher werden an die Softwarekomponenten hohe Anforderungen in Bezug auf Stabilität, Robustheit und Ausfallsicherheit gestellt. Die Anforderungen an den FCC sind geringer, da auf diesem Computer experimentelle Software ausgeführt wird.

3.2. Flugabbruchsystem

Das Flugabbruchsystem ermöglicht es zu jedem Zeitpunkt das Luftfahrzeug am Verlassen des zugewiesenen, räumlichen Flugbereiches zu hindern. Im Falle eines Flugabbruchs wird der Motor abgeschaltet und die Aktuatoren des Rotors, des Ruders und des Bugrads bewegen sich in eine vordefinierte Position. Durch diese Konfiguration gleitet das Luftfahrzeug in einer räumlich begrenzten, spiralförmigen Bewegung in Richtung Boden. Der Flugabbruch ist nicht reversibel (siehe BILD 4). Bei einem Flugabbruch wird das Luftfahrzeug sehr wahrscheinlich beschädigt.

Drei Möglichkeiten sind zum Auslösen des Flugabbruchs vorgesehen:

1. C2 Datenlink / Software
2. RC-Funkfernsteuerung
3. Flugabbruchdatenlink

Die Möglichkeiten 2 und 3 sind derart realisiert, dass unabhängig von einer Fehlfunktion oder einem Ausfall der Bordcomputer ein Flugabbruch ausgelöst werden kann. Der Flugabbruch wird ebenfalls durch einen Stromausfall an Bord ausgelöst. Allerdings wird in diesem Fall ausschließlich der Motor abgestellt.

4. SOFTWAREARCHITEKTUR

In diesem Abschnitt wird auf die bordseitige Software eingegangen, welche auf den beiden Bordcomputern CIC und FCC ausgeführt wird. Ziel der Architektur ist es, modulare bordseitige Software mit möglichst geringer Komplexität und minimalen Schnittstellen pro Modul zu entwickeln, die jeweils unterschiedlichen Anforderungen an die Kritikalität der bereitgestellten Funktion entsprechen können. Das modulare Konzept unterstützt die Entwicklung im experimentellen System, schafft wiederverwendbare Implementierungen und ermöglicht die saubere Auftrennung in stabile, robuste und experimentelle Software. Nachteile sind die erhöhte Komplexität in der Systemsicht, oder die möglicherweise erschwerte Einhaltung von Echtzeitanforderungen.

Die in diesem Bericht beschriebene Software ist nicht relevant für die Erlangung einer Betriebserlaubnis nach den risikobasierten Prinzipen der SORA, denn das Flugabbruchsystem verhindert, dass das Luftfahrzeug eine Gefahr für Dritte darstellt. Jedoch ist im weiteren Laufe des Projektes die Möglichkeit vorgesehen die Entscheidung zum Flugabbruch von den Bordcomputern treffen zu lassen. In diesem Fall werden Teile der Software relevant für die Betriebserlaubnis und die verteilte und modulare Architektur hilft dann bei der Auslagerung von Teilen der Hard- und Software in nachweispflichtige Subsysteme.

Die Software ist objektorientiert in der Programmiersprache C++ geschrieben. Funktionalitäten werden modular gekapselt, und erfüllen die Ansprüche an Testbarkeit und Wiederverwendung. Eine ausreichende Codequalität wird durch Coding-Guidelines [7], Reviews und ein abgestimmtes Testkonzept mit unterschiedlicher Granularität und Komplexität gewährleistet. Auf die Entwicklungsinfrastruktur und die Testprozeduren wird im Abschnitt 4.4 näher eingegangen.

Die bordseitige Software wird ohne weitere Nutzerinteraktion nach dem Start der Bordcomputer automatisch gestartet.

Die funktionale Trennung der robusten Software auf dem CIC und der experimentellen Software auf dem FCC gliedert die folgenden Abschnitte. Eine Middleware (Abschnitt 4.3) übernimmt die Kommunikation unter den Softwaremodulen.

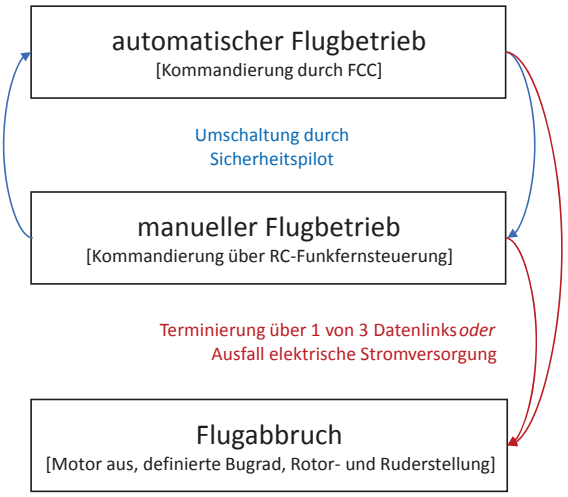

BILD 4 Übersicht über die unterschiedlichen Kommandierungsmodi

4.1. Core Interface Computer

Der CIC übernimmt die Ansteuerung der Aktuatoren, Schalter und sonstiger Steuerelemente. Zusätzlich werden diverse Systemstatusdaten auf dem CIC empfangen. Durch die zentrale Rolle und die als Single Point of Failure ausgelegte Funktion zur Ansteuerung der Subsysteme, ergeben sich besondere Anforderungen an die Robustheit und Ausfallsicherheit der Software auf dem CIC. Da jedoch ein Ausfall der CIC Funktionalität nach dem Betriebskonzept nach SORA nicht nachweispflichtig ist, ist es möglich die Software nicht ausdrücklich nach Standards wie etwa der DO-178C [8] zu entwickeln.

Dennoch ist die Software auf dem CIC hochverfügbar und robust gestaltet: Zum einen ist das Hauptprogramm für die Kernfunktionalitäten zustandsfrei gehalten, was die Komplexität auf ein Minimum reduziert und eine gute Testbarkeit ermöglicht. Zum anderen werden nur ausgewählte Bibliotheken dritter Parteien eingesetzt, für die aufgrund langjähriger Einsatzerfahrung bzw. Herstellergarantien eine hohe Zuverlässigkeit evident ist. Die Zustandsfreiheit erlaubt es darüber hinaus den Prozess durch einen Software-Watchdog zu überwachen, der im Falle eines Programmabsturzes einen unmittelbaren Neustart herbeiführt.

Im Folgenden wird die Architektur und Implementierung genauer beleuchtet.

Eine Aufteilung der Funktionen in unterschiedliche Prozesse, soll die CIC Software modular und die einzelnen Prozesse einfach gestalten. Auf dem CIC laufen neben dem Betriebssystem QNX folgende Programme:

- Initialisierungsprogramm
- Hauptprogramm
- Kommunikation & Logging Programm (K&L)

Während das Initialisierungs- und Hauptprogramm nach BILD 5 sequentiell aufgerufen werden und ablaufen, läuft das K&L-Programm parallel und schreibt die Systemzustände laufend mit.

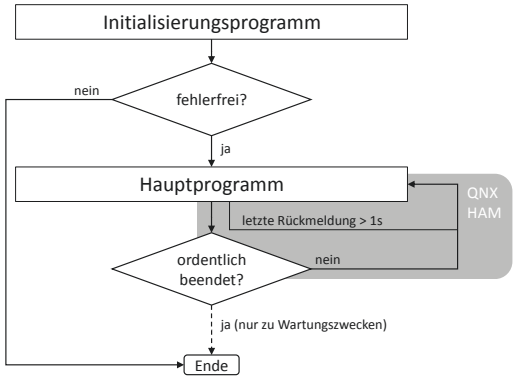

BILD 5 Initialisierung und Hauptprogramm

Initialisierungs- und Hauptprogramm

Während das Prinzip einer zustandslosen Software im laufenden Betrieb bis auf wenige Ausnahmen, etwa Nachrichtenparser, eingehalten werden kann, erlaubt die Initialisierung der verschiedenen Aktuatoren und Subsysteme in weiten Teilen keine zustandsfreie Software. Aus diesem Grund werden alle Systeme in einem eigenen, vorgelagerten Prozess initialisiert. In Abhängigkeit des Sicherheitssteckers werden die Subsysteme des Luftfahrzeuges initialisiert. Während beim Sicherheitsstecker „Flight“ sämtliche Subsysteme initialisiert werden, werden bei Sicherheitssteckern für den Laborbetrieb, beispielsweise die Aktuatoren nicht initialisiert oder der Anlasser deaktiviert, um Verletzungen zu vermeiden. Während der Initialisierung wird geprüft, ob die Kommunikation mit allen Systemen fehlerfrei stattfindet. Die Aktuatoren werden, je nach Sicherheitsstecker, mit verminderter Verfahrgeschwindigkeit in Nullposition bewegt. Sollte die Initialisierung an einer Stelle fehlschlagen, wird das Gesamtsystem automatisch und gefahrlos wieder heruntergefahren.

Nach der erfolgreichen Initialisierung des Systems wird, wie in BILD 5 gezeigt, das Hauptprogramm automatisch gestartet. Das Hauptprogramm nimmt die Steuerkommandos entgegen und leitet diese an die entsprechenden Subsysteme weiter (siehe BILD 6). Manuelle Steuerkommandos kommen von den zwei verbauten RC-Empfängern. Durch ein Voting, welches u.a. auf der Empfangsqualität des Signals beruht, werden ungültige RC-Kommandos verworfen. Die Kommandos der automatischen Flugsteuerung kommen vom FCC. Durch einen Schalter auf der RC-Fernsteuerung kann der Sicherheitspilot zu jedem Zeitpunkt die Steuerung zwischen automatisch und manuell wechseln. Nach der Wahl der Quelle der Steuerkommandos, werden die Wertebereiche der Kommandos überprüft, limitiert und an die jeweiligen Treiber der Subsysteme weitergeleitet.

Kommandierte Werte, die Rückmeldungen von den Aktuatoren sowie Sensorwerte aus dem VIS und der PUU werden aus dem Hauptprogramm an das K&L-Programm weitergeleitet.

Fehlerzustände im Hauptprogramm dürfen das Gesamtsystem nicht gefährden. Aus diesem Grund wird zwischen fehlerhaften Informationen, die durch Subsysteme eingespeist werden und internen Fehlern unterschieden. Da der Ausfall eines Subsystems nicht zwingend zum Gesamtsystemverlust führen muss, ist das Hauptprogramm robust gegen solche Fehler von außen ausgelegt. Das Ausbleiben einer Statusnachricht von einem Aktuator, darf bei-

spielsweise nicht die Kommunikation zu anderen Aktuatoren behindern. Darüber hinaus sind in Abhängigkeit des gewählten Sicherheitssteckers beispielsweise Aktuatoren nicht mit Strom versorgt. Die Funktionalität des Hauptprogramms wird davon nicht beeinflusst.

Sollte jedoch im Hauptprogramm ein unerwarteter, internen Laufzeitfehler auftreten, wird es durch den Software-Watchdog High Availability Monitor (HAM) [9] des QNX Betriebssystems instantan neu gestartet. Dazu wird das Hauptprogramm zu Beginn beim HAM angemeldet und sendet regelmäßig einen Heartbeat (siehe auch BILD 5). Ein Neustart wird beim Ausbleiben des Heartbeats oder bei einem nicht fehlerfreien Beenden des Programms ausgelöst.

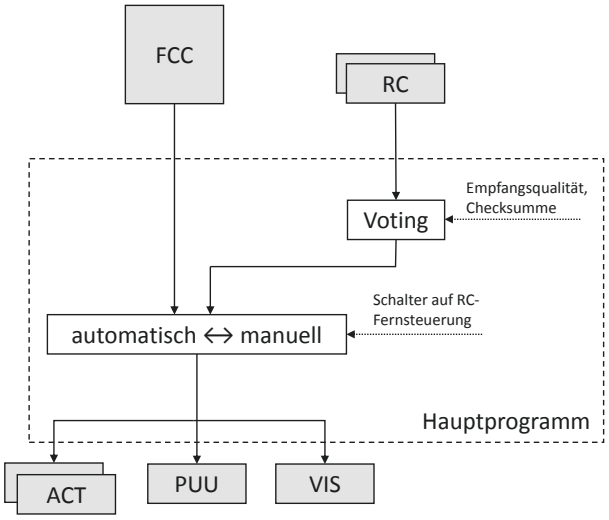

BILD 6 Signalfluss der Steuerkommandos im CIC Hauptprogramm für den rein automatischen und manuellen Flugbetrieb

Das CIC Hauptprogramm läuft mit einer konstanten Frequenz von 100 Hz. Dies reicht aus um alle Kommandos der RC-Empfänger, welche mit der gleichen Frequenz empfangen werden, ohne Undersampling an die Aktuatoren und das VIS weiterzuleiten. Die Aktuatoren und das VIS werden mit einer Frequenz von 50 Hz und 100 Hz kommandiert und senden mit der gleichen Frequenz Statusrückmeldungen. Die Totzeit einer Steuereingabe auf der RC-Fernsteuerung bis zur Aktuatorbewegung wurde mit $\Delta T \leq 30$ ms gemessen, wobei dies der minimalen Auflösung des gewählten Messverfahrens entsprach.

Kommunikation und Logging

Das Programm zur Kommunikation und Logging ist von der Funktionalität des Initialisierungs- und Hauptprogramms getrennt. Dieses Programm loggt Daten aus dem Initialisierungs- und Hauptprogramm und sorgt für die Kommunikation ankommender und ausgehender Datenströme der beiden Programme über die Middleware an weitere Programme, z.B. an die automatische Flugsteuerung (FCC) oder die Bodenstationssoftware.

Die Interprozesskommunikation zwischen Hauptprogramm und K&L-Programm geschieht per User Datagram Protocol (UDP) und Shared Memory (siehe BILD 8). Das K&L-Programm läuft mit einer niedrigeren Priorität, damit die Ausführung des Hauptprogramms nicht beeinflusst wird.

4.2. Flight Control Computer

Der FCC dient als Plattform für experimentelle Software und die Flugsteuerung. Darüber hinaus werden neben der Verwaltung des C2 Datenlinks Flugzustandsdaten der INS, des Radarhöhenmessers und des Luftdatensystems durch Software des FCC entgegengenommen und weiterverarbeitet. Ebenso wie die Software auf dem CIC ist die FCC Funktionalität nicht redundant ausgeführt.

Auf dem FCC werden die Flugzustandsdaten zunächst in einer Sensorfusion verarbeitet und anschließend an weitere Programme und die Bodenkontrollstation verteilt. Da die Bandbreite über den C2 Datenlink begrenzt ist, sorgt ein eigener Prozess dafür, dass Daten nur mit einer definierten Senderate und nicht etwa doppelt versendet werden.

Zum Zeitpunkt der Veröffentlichung dieses Beitrags sind bereits die Sensorfusion und die Verwaltung des C2 Datenlinks in den FCC integriert. Die folgenden Funktionen werden zukünftig integriert.

Die automatische Flugsteuerung wird in Form des Missionsmanagements implementiert. Das Missionsmanagement wird als hierarchischer Automat modelliert. Zustände definieren unterschiedliche Betriebsmodi, Flugphasen und Flugmaneuver. Je nach Zustand werden auch Flugregelungsmodi geschaltet. Darüber hinaus können im Missionsmanagement Flugpläne und -bahnen automatisch geplant werden oder von der Bodenstation empfangen werden. Die Datenflüsse auf dem FCC sind in BILD **7** dargestellt.

Weitere experimentelle Systeme sind beispielsweise ein automatisches Health-Monitoring oder Geofencing. Letzteres soll im Rahmen des Betriebskonzeptes in Zukunft die Einhaltung des zugewiesenen Flugbereiches garantieren. Nach der experimentellen Phase kann eine solche Software dann beispielsweise auf dem CIC oder einem weiteren dedizierten Rechner als robustes sicherheitsrelevantes System eingesetzt werden.

Auf dem FCC werden neben dem Betriebssystem QNX die folgenden Programme laufen:

- Sensorfusion
- C2 Datenlink
- Missionsmanagement und Flugregelung
- experimentelle Software (z.B. Health-Monitor)

4.3. Middleware

Durch die modulare und verteilte Softwarearchitektur kommt der Kommunikation einzelner Softwaresegmente über eine Middleware eine zentrale Bedeutung zu.

Die Kommunikation zwischen Haupt- und K&L-Programm auf dem CIC stellt pro Datagramm stets eine direkte gerichtete Verbindung dar, bei der aufgrund der Ausführung der beiden Prozesse auf derselben Hardware davon ausgegangen werden kann, dass Verbindungsausfälle nicht vorkommen. Darüber hinaus ist durch die festgelegte Ausführungsreihenfolge der einzelnen Prozesse auf dem CIC sichergestellt, dass beide Kommunikationsteilnehmer auch vorhanden und empfangsbereit sind. Bei der Experimentalsoftware auf dem FCC sowie der Verbindung zur Bodenstation sind diese Voraussetzungen so nicht gegeben.

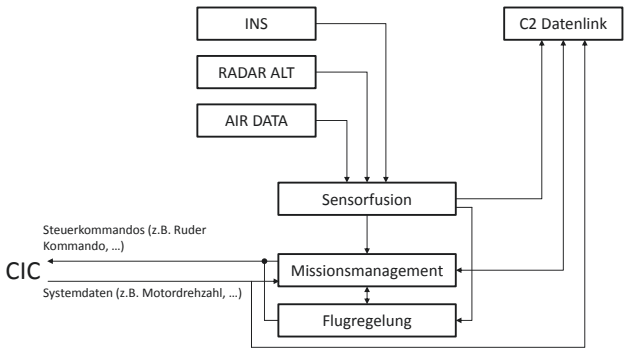

BILD 7 Datenfluss auf dem FCC ohne experimentelle Software

Je nach Experiment bzw. Situation und Flugzustand können unterschiedliche Prozesse zu- und abgeschaltet werden. Auch Daten die jeweils für Berechnungen oder das Situationsbewusstsein des Bodenpersonals benötigt werden variieren. Andererseits ist die Bandbreite der Datenlinks insbesondere zum Bodensegment begrenzt, sodass nicht jederzeit alle verfügbaren Daten zwischen allen potentiellen Kommunikationsteilnehmern mit der höchstmöglichen Datenrate ausgetauscht werden können.

Daher kommt für alle Prozesse die nicht auf dem CIC ausgeführt werden die DLR interne nachrichtenorientierte Middleware BBOO (Bulletin Board Object Oriented) zum Einsatz, um die Kommunikation zwischen den verteilten Komponenten zu gewährleisten. Die jeweils benötigten Daten können dabei flexibel angefordert und neue Verbindungen bei Bedarf ad Hoc aufgebaut werden. Der Datenaustausch und Verbindungsaufbau erfolgt dabei direkt zwischen den beteiligten Prozessen, sodass der Ausfall einzelner Prozesse nicht die restliche Kommunikation beeinträchtigt. Ein Routing findet nur statt, wenn eine direkte Verbindung im Netzwerk nicht möglich ist. Die Programme auf dem CIC sind über das K&L-Programm ebenfalls an die Middleware angebunden (siehe BILD 8).

Für Daten, die regelmäßig aktualisiert werden und bei denen neue Datensätze die alten überschreiben, werden die Daten meist mit begrenzter Datenrate über ein Publish-Subscribe (Veröffentlicher und Abonnent) System an die Empfänger verteilt. Die Middleware übernimmt dabei die Verwaltung der Abonnenten und die möglichst effiziente Verteilung der Daten.

Kommandos und dringliche einmalige Nachrichten an dedizierte Empfänger wie Warnungen oder Fehlerzustände werden asynchron über Verbindungen mit Empfangsbestätigung versandt. Dabei überwacht die Middleware den Versand und stellt Rückmeldungen zum Transferstatus bereit.

Große Datensätze wie Konfigurationsdateien, Umgebungsinformationen oder Beschreibungen ganzer Missionen werden von der Middleware in übertragbare Pakete aufgeteilt, versandt und erst bei Empfang des vollständigen Datensatzes an den Empfänger weitergegeben, sodass nur konsistente Datensätze ausgewertet werden.

Über einen Registrierungsservice, der die Informationen über die aktuell verfügbaren Datenquellen unter allen Programmen, die mit Hilfe der Middleware kommunizieren können verteilt, kann der Verbindungsaufbau dynamisch zur Laufzeit realisiert werden. Somit erlaubt der Einsatz der Middleware auch eine nachträgliche Aus- bzw. Verlagerung einzelner Algorithmen auf weitere Prozesse bzw. Recheneinheiten. Weitere Informationen, auch zur Zeitsynchronisation der Daten finden sich in [10].

4.4. Entwicklungsinfrastruktur

In diesem Abschnitt wird die Entwicklungsinfrastruktur beschrieben werden.

Der Quellcode sämtlicher Software wird in Code Repositories verwaltet und ist in einen Continuous Integration (CI) Prozess einbezogen. Dieser CI Prozess bietet neben dem automatisierten Kompilieren und Ausführen von Unittests die Möglichkeit die Codequalität durch entsprechende Code-Analyse Werkzeuge fortlaufend zu überprüfen. Darüber hinaus sind Software-Reviews von wesentlicher Bedeutung für die Codequalität.

Für die Verifikation der Software mit Hilfe von kontinuierlich durchgeführten Tests wurden parallel zur Entwicklung Unittests und Integrationstests definiert und implementiert.

Nach BILD 9 unterstützen Unit-Tests, Subsystemtests, Prüfstandtests, Software in the Loop (SITL) und Hardware in the Loop (HITL) Tests und Gesamtsystemtests die Gesamtsystemintegration. Vor einem Flugversuch wird das Gesamtsystem in Motorläufen und Rollversuchen betrieben, welche als zusätzliche Gesamtsystemtests unter realistischer Beanspruchung der Hardware, z.B. im Hinblick auf Vibrationen dienen.

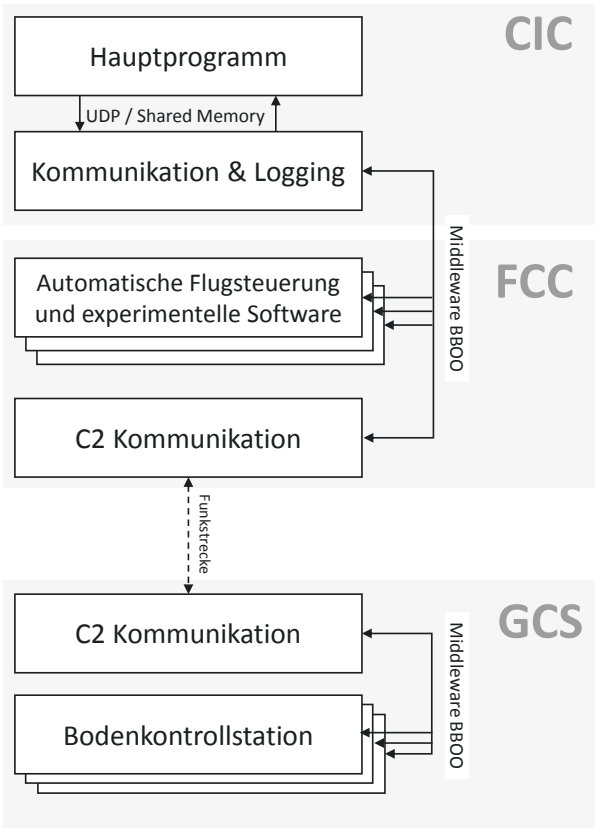

BILD 8 Kommunikation über die Middleware BBOO

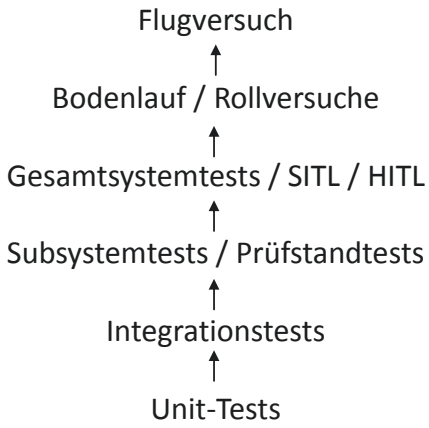

BILD 9 Vorgehen bei der Gesamtsystemintegration

Neben der Entwicklungs- und Testinfrastruktur werden zu Betrieb, Wartung und Fehleranalyse auch Werkzeuge zum Aufbereiten und zur Visualisierung aufgezeichneter Daten benötigt. Diese werden in Abstimmung mit dem Logging entwickelt bzw. ausgewählt und sind integraler Bestandteil in den komplexeren Testszenarien. Ein komfortables Analysieren der Logdaten ermöglicht eine effiziente Fehlersuche während der Integrationsphasen und eine verständliche Analyse von Flugversuchen. Neben der klassischen Visualisierung kommen Werkzeuge zur automatischen Auswertung der Logdaten zum Einsatz, beispielsweise auf der Basis der Spezifikationssprache „LOLA“ [11].

5. ERPROBUNG

Neben dem ausführlichen Testen unter Laborbedingungen wurde das System in Motorläufen und Rollversuchen erprobt. Während der Erprobung wurde das Luftfahrzeug von einem Sicherheitspiloten manuell gesteuert. Die Systeme wurden durch Bodenstationspersonal überwacht.

Die Analyse der Logdaten ergab, dass die Daten vollständig und mit einer konstanten Rate aufgezeichnet wurden. Das Logging des Initialisierungs- und Hauptprogrammes auf dem CIC generierte 10,5 MB Logdaten pro Minute. Die Sensorfusion auf dem FCC erzeugte 2,5 MB Logdaten pro Minute. Die Datenübertragung zur Bodenstation war robust und zuverlässig. Es traten keine Datenverluste auf.

BILD 10 Umbenannter ALAADy-Demonstrator bei Rollversuchen mit drehendem Rotor

6. ZUSAMMENFASSUNG UND AUSBLICK

In diesem Beitrag wurden die Softwarearchitektur und -funktionen für einen unbemannten Luftfrachttransportdemonstrator vorgestellt. Die vorgestellten Softwarefunktionen reichen von der Ansteuerung der Aktuatoren bis zur automatischen Flugsteuerung. Vor dem Hintergrund einer risikobasierten Nachweisführung wurden besondere Merkmale, wie die Aufteilung der Software auf mehrere Rechner und in robuste und experimentelle Softwareteile beschrieben und diskutiert. Mit der Verwendung eines Software-Watchdogs wurde die Anwendung einer technischen Methode zur Bereitstellung der Hochverfügbarkeit von Softwareteilen vorgestellt. Der Beitrag schließt mit der Vorstellung der Entwicklungsinfrastruktur und einem kurzen Bericht der bisherigen Erprobung.

Nach einem manuellen Erstflug soll der ALAADy-Demonstrator schrittweise automatisiert werden, um letztendlich einen voll automatisierten Frachttransport zu demonstrieren. Neben den technischen Herausforderungen werden in diesem Zusammenhang auch Fragestellungen im Rahmen der Nachweisführung beantwortet werden müssen, etwa wenn das Flugabbruchssystem durch Software automatisch ausgelöst werden soll. Die modulare Software- und Avionikarchitektur wird sowohl bei den technischen Fragestellungen, als auch für die Herausforderungen bei der Nachweisführung hilfreich sein wird. Die Auftrennung der Bordcomputer in mehrere Systeme wird es in Zukunft ermöglichen, Teile der bordseitigen Software in abgekapselten zuverlässigen Subsystemen zu verwenden. Ein automatisches Geofencing kann so beispielsweise das Verlassen des vorab definierten Flugbereiches bei Flügen außerhalb des Sichtbereiches (Beyond Vision Line of Sight, BVLOS) verhindern, während experimentelle Software weiterhin auf einer separaten Hardware ausgeführt werden kann.

7. DANKSAGUNG

Neben den Autoren dieses Artikels haben diverse Kollegen an dem Aufbau des ALAADy-Demonstrators mitgewirkt. Jörg Dittrich sei unter anderem für die Ideen und organisatorische Unterstützung bei der Realisierung des Projekts gedankt. Holger Duda und Falk Sachs ermöglichen mit ihrem detaillierten Wissen über die Flugmechanik von Tragschraubern die Modifikation des bemannten Systems. Michael Kislat und Jörg Seewald unterstützen als Sicherheitspilot und Co-Pilot mit ihren Fertigkeiten den Betrieb des unbemannten Systems. Besondere Anteile an der Konzeptionierung und Implementierung der Software haben (alphabetisch): Thorben Bornscheuer, Danilo Galisteu, Christian Hoffmann, Martin Laubner, Sebastian Schirmer, Simon Schopferer und Susanne Schulz.

8. LITERATUR

[1] J. C. Dauer, S. Lorenz und J. Dittrich, „Automated Low Altitude Air Delivery,“ in *Deutscher Luft- und Raumfahrtkongress (DLRK)*, Braunschweig, Germany, 2016.

[2] Joint Authorities for Rulemaking of Unmanned Systems, „JARUS guidelines on Specific Operations Risk Assessment (SORA),“ JARUS, 2017.

[3] EASA, „Concept of operation for Drones: A risk based approach to regulation of unmanned aircraft,“ EASA, 2015.

[4] SAE International, „Guidelines for Development of Civil Aircraft and Systems EUROCAE ED-79A / SAE ARP 4754A,“ Eurocae, 2010.

[5] A. Bierig, S. Lorenz, M. Rahm und P. Gallun, „Design Considerations and Test of the Flight Control Actuators for a Demonstrator for an Unmanned Freight Transportations Aircraft,“ in *Recent Advantages in Aerospace Actuations Systems and Components*, Toulouse, France, 2018.

[6] Blackberry QNX, „Blackberry QNX,“ Blackbarry, [Online]. Available: https://blackberry.qnx.com/en. [Zugriff am 1. August 2018].

[7] A. Schönhoff, M. Friedrich, K. Harth, G. Jarasch, M. Momberg, S. Schärer und D. Wetteborn, „System Design of the Barracuda Flight Control System,“ in *IFAC Proceedings Volumes*, Pretoria, South Africa, 2007.

[8] W. A. Halang und R. M. Konakovsky, Sicherheitsgerichtete Softwaresysteme, Springer, 2013.

[9] RTCA, „DO-178C Software Considerations in Airborne Systems and Equipment Certification,“ RTCA SC-167 / EUROCAE WG-12, 2011.

[10] Blackberry QNX, „QNX Software Development Platform 6.6 High Availability Monitor,“ BlackBerry, [Online]. Available: http://www.qnx.com/developers/docs/6.6.0_anm11_wf10/#com.qnx.doc.ham/topic/hamover.html. [Zugriff am 1. August 2018].

[11] J. C. Dauer, L. Goormann und C. Torens, „Steps Towards Scalable and Modularized Flight Software for Unmanned Aircraft Systems,“ International Journal of Advanced Robotic Systems, 2014.

[12] F. M. Adolf, P. Faymonville, B. Finkenbeiner, S. Schirmer und C. Torens, „Stream Runtime Monitoring on UAS,“ *International Conference on Runtime Verification,* pp. 33-49, 13.-16. September 2017.