INNOCUBE DER ERSTE DRAHTLOSER SATELLIT

S. Montenegro, Tom Baumann, Erik Dilger, Felix Sittner, Michael Strohmeier, Thomas Walter: Universität Würzburg, Simon Gläsner: TU-Berlin

Zusammenfassung

Innocube ist ein gemeinsames Projekt der TU-Berlin und die Uni Würzburg. Ziel des Vorhabens ist ein 3U-Cubesat für die in Orbit Technologiedemonstration von zwei innovativen Technologien, die im Rahmen des InnoSpace-Master-Programms gefördert wurden: Skith und Wall#E. Im Rahmen von SKITH wurde eine kabellose Infrastruktur für Satelliten entwickelt. Durch die Kombination von kabellosen Standards, die aktuell im Bereich „Internet of Things“ und „Industrie 4.0“ entwickelt werden, und fehlertoleranter und robuster Software entsteht ein System, das mit Kabelverbindungen im Satelliten konkurrieren kann und viele Vorteile bietet. Das Entfernen der Verkabelung innerhalb des Satelliten verringert Gewicht und Komplexität des Raumfahrzeugs. Dies ermöglicht kleinere und flexiblere Systeme. Ein modularer Satellit kann aufgebaut werden, indem die Komponenten zusammengeschraubt werden, ohne auf Verkabelung für den Datenaustausch achten zu müssen.

1. ÜBERSICHT

Im Rahmen des Projektes werden programmierbare Module zur Funkübertragung entwickelt, die die typischen Schnittstellen für IO bieten. Diese Front-End Module führen entsprechend alle erforderlichen Protokollkonvertierungen durch und stellen somit für alle IO Geräte eine einheitliche, drahtlose Schnittstelle zur Verfügung. Auf diese Weise wird der Bordcomputer völlig unabhängig von den Schnittstellen der verwendeten Sensoren und Aktuatoren. Ein Gerät innerhalb des Satelliten durch ein anderes zu ersetzen, nachdem die Integration begonnen oder abgeschlossen wurde, ist normalerweise undenkbar. Durch den einheitlichen Funkstandard bietet unser System die nötige Flexibilität, dies ohne Probleme zu ermöglichen.

Ein schöner Nebeneffekt dieser Bemühungen ist die einfache Überwachung, auch nach der endgültigen Integration, ohne dass Schnittstellen für externe Geräte oder zusätzliche Software vorgesehen werden müssen. Sensordaten können einfach über einen Empfänger außerhalb des Satelliten aufgezeichnet und ausgewertet werden, zudem können auch Sensordaten von außen über einen Sender in das Avionik-Netzwerk eingespeist werden. Somit sind selbst Hardware- oder Software-In-The-Loop Tests am fertig integrierten Satelliten möglich, was auf konventionellen Satelliten nur mit großem Aufwand machbar wäre.

Im Projekt wird ein 3U-Cubesat gebaut (Siehe Abbildung 1), der alle Anforderungen des CubeSat-Standards und des momentan anwendbaren Verhaltenskodex für Satellitenbetrieb erfüllt. Der CubeSat soll alle notwendigen Tests erfolgreich bestehen, in den Orbit gebracht werden und eine Lebensdauer von mindestens 1 Jahr erreichen.

Cubesats nutzen typischerweise COTS-Prozessoren, meist aus der ARM-Familie. Sehr oft ist das Data-Management Subsystem NICHT redundant. Sensoren werden direkt an den Bordcomputer angeschlossen. Einige wenige Entwicklungen nutzen Busse wie z.B CAN und verteilte Rechner. Solche Entwicklungen haben meistens eine duale Redundanz. Bei InnoCube, basiert das gesamte Command and Data Management-Subsystem auf Skith. Skith kombiniert Rechenleistung, Speicher, IO Interfaces und Funkkommunikation als Ersatz zu den üblichen Bussystemen. Skith verwendet einen STM32F4 Prozessor, den wir bereits in Technosat, S-Net, TUBIN, eingesetzt haben. Das Data-Management System und die gesamte Avionik wird ein Netzwerk von redundanten Skith-Modulen sein. Die Software wird auf RODOS basieren, denn RODOS bietet die optimale Unterstützung für verteiltes und redundantes Rechnen.

Abbildung 1 zeigt die Struktur von Innocube.

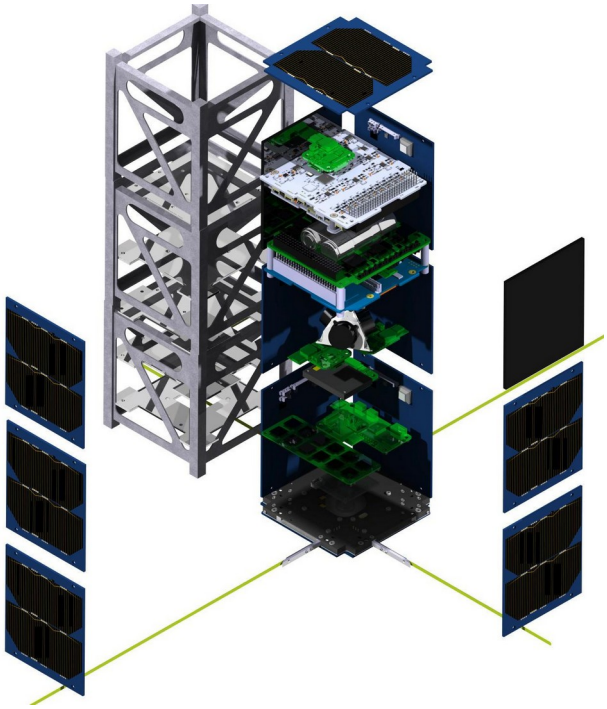

BILD 1. Mechanische Struktur von Innnocube

doi: 10.25967/570007

2. WISSENSCHAFTLICHE UND TECHNISCHE ZIELE:

Die wissenschaftliche und technische Ziele sind:

• Verwertung und Demonstration der : Skith und Wall#E Technologien im Weltraum.

• Lehre- und Studierendenausbildung

• Programmier- und konfigurierbare experimentelle Nutzlast für Forschungszwecke zur Orbitbestimmung und Propagation: EPISODE

Die technische Ziele des 4-jährigen Vorhabens ist, die InnoSpace-Sieger-Technologien Skith und Wall#E (siehe Einführung) in den Weltraum zu bringen und für künftigen Missionen zum TRL 5 zu qualifizieren. Jede Technologie, Skith und Wall#E, hat ein Alleinstellungsmerkmal. Noch nie gab einen drahtlosen Satelliten (ohne Harness für Daten) und noch nie gab es einen Satelliten, der Energie in seiner Struktur speicherte. Nebenbei haben wir als Ziel die Vermittlung von ingenieurwissenschaftlichen Fähigkeiten und eine praxisnahe nachhaltige Ausbildung von Studierenden, wichtige technische und wissenschaftliche Publikationen und die Steigerung der universitären Kompetenzen.

Im Projekt wird ein 3U-Cubesat gebaut (Siehe Abbildung 1), der alle Anforderungen des CubeSat-Standards und des momentan anwendbaren Verhaltenskodex für Satellitenbetrieb erfüllt. Der CubeSat soll alle notwendigen Tests erfolgreich bestehen, in den Orbit gebracht werden und eine Lebensdauer von mindestens 1 Jahr erreichen.

In den Betriebsjahren sollen wissenschaftliche/technologische Daten gesammelt werden und gleichzeitig wird eine konfigurierbare GPS-Nutzlast als Experimentierplattform für Studenten benutzt. Um Nachhaltigkeit zu gewährleisten, ist es das Ziel, eine Arbeitsgruppe für Kleinsatelliten zur Nachwuchsförderung zu etablieren und zu fördern, an der beide Institute beteiligt sind. Diese soll im Rahmen zukünftiger Vorhaben CubeSats vom Entwurf bis zum Flugmodell entwickeln, fertigen, qualifizieren, starten und betreiben, sowie Technologien weiterentwickeln und erproben. Abbildung 1 zeigt eine Explosionszeichnung der Struktur.

3. DRAHTLOSER SATELLITENPLATTFORM: SKITH – SKIP THE HARNESS

Im Rahmen von SKITH wurde eine kabellose Infrastruktur für Satelliten entwickelt. Durch die Kombination von kabellosen Standards, die aktuell im Bereich „Internet of Things“ und „Industrie 4.0“ entwickelt werden, und fehlertoleranter und robuster Software entsteht ein System, das mit Kabelverbindungen im Satelliten konkurrieren kann und viele Vorteile bietet. Das Entfernen der Verkabelung innerhalb des Satelliten verringert Gewicht und Komplexität des Raumfahrzeugs. Dies ermöglicht kleinere und flexiblere Systeme. Ein modularer Satellit kann aufgebaut werden, indem die Komponenten zusammengeschraubt werden, ohne auf Verkabelung für den Datenaustausch achten zu müssen.

Im Rahmen des Projektes werden programmierbare Module zur Funkübertragung entwickelt, die die typischen Schnittstellen für IO bieten. Diese Front-End Module führen entsprechend alle erforderlichen Protokollkonvertierungen durch und stellen somit für alle IO Geräte eine einheitliche, drahtlose Schnittstelle zur Verfügung. Auf diese Weise wird das Data-Management-System völlig unabhängig von den Schnittstellen der verwendeten Sensoren und Aktuatoren. Ein Gerät innerhalb des Satelliten durch ein anderes zu ersetzen, nachdem die Integration begonnen oder abgeschlossen wurde, ist normalerweise undenkbar. Durch den einheitlichen Funkstandard bietet unser System die nötige Flexibilität, dies ohne Probleme zu ermöglichen.

Ein schöner Nebeneffekt dieser Bemühungen ist die einfache Überwachung, auch nach der endgültigen Integration, ohne dass Schnittstellen für externe Geräte oder zusätzliche Software vorgesehen werden müssen. Sensordaten können einfach über einen Empfänger außerhalb des Satelliten aufgezeichnet und ausgewertet werden, zudem können auch Sensordaten von außen über einen Sender in das Avionik-Netzwerk eingespeist werden. Somit sind selbst Hardware- oder Software-In-The-Loop Tests am fertig integrierten Satelliten möglich, was auf konventionellen Satelliten nur mit großem Aufwand machbar wäre.

Im Vorhaben InnoCube-SKITH wird die SKITH-Technologie verwendet, um einen Satellitenbus (auch Satellitenplattform genannt) zu implementieren. Bei InnoCube-Skith werden wir einen „Bordcomputer“ und IO-Devices, die miteinander per Funk verbunden sind, entwickeln. In Innocube der „Bordcomputer“ ist eigentlich ein Netzwerk von vielen kleinen Computer, vernetzt per Funk. Wir werden dies das Command and Data Management System (CDMS) nennen.

Wir werden die Grundplatinen, die wir in SKITH demonstriert haben, verändern müssen, denn SKITH war für größere Satelliten geplant, denn die Vorteile der drahtlosen Kommunikation wachsen mit der Größe des Satelliten. InnoCube ist sehr klein, und daher haben wir weniger Energie für den Bordcomputer zur Verfügung als bei größeren Satelliten. Wir werden sparsamere Funk-Interfaces benutzen müssen.

3.1. Validierung von SKTIH

Der gesamte Nutzlast-Betrieb der InnoCube-Platform (Bus) basiert auf Sktih. Nach dem wir den ersten Kontakt mit InnoCube in LEOP erfolgreich durchgeführt haben, sehen wir, dass Skith im Prinzip funktioniert. Danach gilt es, die Qualität der Datenübertragung abzuschätzen. Die Plattform (Skith) wird normalen Betrieb eigenen Statistiken führen: Anzahl der Kommunikationsknoten, Anzahl der Nachrichten, verlorene Nachrichten, verfälschten Nachrichten, Feldstärken, Echtzeit Performance. In der Phase 2 (Commisioning) werden wir alle diese Statistiken als extended und History Telemetrie anfordern. Diese erste Daten werden wir dann mit unseren Erwartungen und mit unseren Messungen während des

Entwicklungsprozess vergleichen. Falls es Abweichungen gibt, werden wir sie dann genauer analysieren. In der Betriebsphase (3. Phase) werden wir periodisch diese Skith Telemetrie (aber mit niedriger Priorität) anfordern, um wie in der Phase 2 (Commisioning Phase) vorgehen.

4. INNOCUBE-HARDWARE

Die Avionik Hardware besteht aus Platinen, die ähnlich wie in einem CD Ständer in die Struktur eingeschoben werden. Diese Platinen werden im Folgenden als Modulträger bezeichnet. Die Module der Avionik, wie z.B. der OBC, benötigen zum Teil nur eine geringe Platinenfläche, sodass einige Modulträger sogar zwei Module „tragen“. Es gibt aber auch Module, die sich über zwei Modulträger verteilen, z.B. die PCDU.

Die Modulträger haben die gleichen Dimensionen und verfügen an der hinteren Kante über einen einheitlichen Stecker, der eine Verbindung zur Backplane herstellt. Die Backplane dient im Wesentlichen der Energieverteilung vom EPS zu den einzelnen Modulen. Die grundsätzliche Kommunikation mit dem Satellitenbus erfolgt per SKITH Funk.

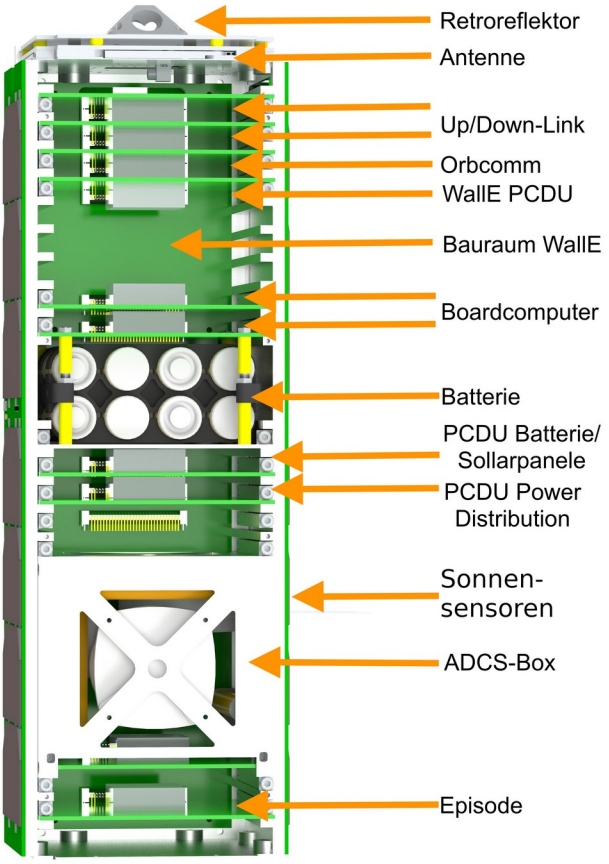

BILD 2. Innnocube-Hardware

In der Abbildung 2 wurden alle Komponenten und ihre Position im Innern des InnoCube Satelliten dargestellt. Insgesamt werden 10 Modulträger mit 12 Modulen integriert. In diesem Abschnitt sollen die wesentlichen Komponenten kurz vorgestellt werden, ohne auf Details einzugehen. Hierfür wird dann auf die entsprechenden Dokumente verwiesen. Jede Komponente wird in einem Steckbrief zusammengefasst und die Reihenfolge entspricht der Anordnung im Satelliten, beginnend beim Retroreflektor und abschließend mit der Payload Episode.

5. INNOCUBE-SOFTWARE

Für jedes Hardware-Target wird mit Hilfe von Real time Core RODOS (https://de.wikipedia.org/wiki/Rodos_(Betriebssystem)) und CORFU (Code Generator) ein boot-Image erstellt. Corfu definiert das statische Software-Design. Hierbei ist die On-Board-Software in einzelne Applikationen eingeteilt, die auf den verschiedenen Knoten laufen.

In Rodos wird das Software-Image für ein Target (Node) als Netzwerk einzelner Applikationen gesehen, die über eine publish-subscribe-Middleware miteinander kommunizieren.

Jede Node Software setzt sich zusammen aus einer Menge grundlegender Apps, die alle Nodes gemeinsam haben und einer oder mehreren speziell für den jeweiligen Node programmierten Apps. Die in einen Node zu kompilierenden Apps werden in der Corfu-Kopnfiguration des jeweiligen Nodes definiert.

Apps kommunizieren über Middleware-Topics, auf welche Topics eine App lauscht (“subscribe”) und auf welchen Topics sie Nachrichten veröffentlichen darf (“publish”), wird in ihrer Corfu-Konfiguration festgelegt. Über Topics eingehende Nachrichten können entweder direkt verarbeitet oder in einen FIFO-Buffer abgelegt und von der App dort abgerufen werden.

Die Weiterleitung der Nachrichten erfolgt über die Rodos Middleware Gateways, die Nachrichten sowohl lokal innerhalb eines Nodes als auch über diverse Protokolle, wie SKITH, UART oder auch TCP und UDP weiterleiten können. Die publisher App kann dabei vorgeben, ob die Nachricht nur lokal oder auch über den Node hinaus geroutet werden soll. Die Rodos Middleware kam bereits in mehreren Satelliten zum Einsatz.

Telecommands werden, wie in Grafik 003 dargestellt, von der Bodenstation aus über den Uplink gesendet und vom am Communications Node (@Comms) angeschlossenen Transceiver (TRX) empfangen. Der Transceiver gibt eingehende Telecommands über das Node-interne Middleware Topic (FROM RADIO) an den ebenfalls auf dem Comms Node laufenden Commander weiter.

Alle Corfu-Apps speichern ihre Standardtelemetrie, gemäß der in ihrer Corfu-Konfigurationsdatei vorgegebenen Struktur, in einer speziellen per Semaphore geschützten Variablen. Der Housekeeper eines Nodes greift periodisch auf die StandardTelemetry-Variablen aller lokalen Apps zu, fasst diese zur Node-spezifischen StandardTelemetry zusammen und veröffentlicht diese auf dem StandardTelemetryTopic.

ExtendedTelemetry veröffentlichen Apps, ausschließlich als Antwort auf ein Telekommand hin, auf dem ExtendedTelemetryTopic.

Die StandardTelemetry und ExtendedTelemetry Topics werden sowohl vom DownlinkManager auf dem Comms-Node, als auch von der TelemetryHistory-App auf dem OBC abonniert

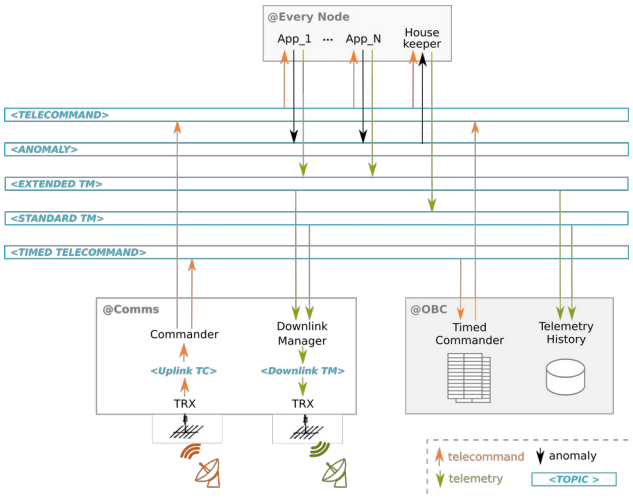

BILD 3: Als Beispiel Anomaly, Telemetry & Telecommand Weiterleitung über Topics und Apps

In der Corfu-Konfigurationsdatei jeder App sind ihre möglichen Anomalien definiert. Währen der Laufzeit veröffentlichen Apps auftretende Anomalien über das AnomalyTopic der Corfu/Rodos-Midleware.

Der Housekeeper jedes Nodes sammelt die Anomalies und schreibt die AnomalyHistory und Informationen, wie oft die jeweilige Anomalie bereits aufgetreten ist in einen Ringbuffer über dem ihm zugeteilten Bereich des SPI-Flashes (nichtflüchtig). Per Telekommando können diese Daten als ExtendedTelemetry angefordert werden

6. INTERNES KOMMUNIKATIONSSYSTEM

Das SKITH Wireless Bus System soll alle Komponenten des Satelliten, wie z.B. Kommunikation, OBC, ADCS, Payload und PCDU drahtlos miteinander verbinden. Dies soll dabei echtzeitfähig, ausfallsicher und möglichst einfach sein. Wir benutzen dazu die im OBDH-DD näher erläuterten SKITH-Module. Diese sind unser OBDH-System mit integrierter Funkschnittstelle. Zur vollen Kontrolle von Redundanz und Timing der Funkkommunikation, benutzen wir unser eigenes, in diesem Dokument vorgestellte, SKITH-Protokol.

Der verwendete „Gecko“-Microcontroller besitzt eine integrierte 2.4 GHz Funkschnittstelle. Diese ermöglicht es beliebige Daten ohne festes Protokoll im 2,4 Ghz-Band mit bis zu 19 dBm Sendeleistung, 2 Mbit/s Datenrate und verschiedenen Modulationen zu senden und zu empfangen.

Der Controller kümmert sich dabei um Präambel, Sync-Word(Frame-Anfangs-Erkennung), Frame-Länge und eine CRC-16 Prüfsumme. Um den Rest, insbesondere Kolissionsvermeidung muss sich der Benutzer selbst kümmern. Auf dieser Schnittstelle bauen wir unser SKITH-Protokoll auf.

Wir benutzen dabei eine GFSK-Modulation mit +/- 1 MHz Frequenzhub bei 1MBit/s Datenrate. Die Bandbreite ist damit mit 2 MHz doppelt so groß wie nötig für die Datenrate, da wir genug Spektrum zur Verfügung haben. Dies bringt uns mehr Stabilität bei der Datenübertragung.

Wir definieren 20 Kanäle mit 2MHz Abstand im Bereich von 2400-2440 MHz. Der Satellit wird nur einen fest eingestellten Kanal benutzen. Eine Änderung über Kommando oder Softwareupdate ist aus Sicherheitsgründen nicht vorgesehen. Die anderen Kanäle können für die Entwicklung benutzt werden. Wir haben keine weiteren 2,4 GHz Funksysteme an Bord und können den Kanal für SKITH beliebig wählen.

Das Funkprotokoll muss die Kommunikation der Module untereinander Koordinieren, so dass verhindert wird dass mehrere Module gleichzeitig senden und so die Übertragung gestört wird.

Es wird ein zentraler, fest definierter Koordinator benutzt der jedem Modul regelmäßig Zeitfenster zuteilt, in denen sie senden dürfen. Dies sorgt für eine geringe Komplexität des Systems und eine festen bekannten Systemzustand.

Damit dieser Zentrale Koordinator nicht zu einem Single-Point-of-Failure wird, ist dieser doppelt vorhanden. Ein Mechanismus unabhängig vom Funksystem sorgt dafür dass immer genau einer der beiden Koordinatoren einschaltet ist. Der Aktive Koordinator wird dabei immer auf Funktion der kompletten Funkstrecke überwacht.

6.1. Funkprotokoll

Das Funkprotokoll muss die Kommunikation der Module untereinander Koordinieren, so dass verhindert wird dass mehrere Module gleichzeitig senden und so die Übertragung gestört wird.

Es wird ein zentraler, fest definierter Koordinator benutzt der jedem Modul regelmäßig Zeitfenster zuteilt, in denen sie senden dürfen. Dies sorgt für eine geringe Komplexität des Systems und eine festen bekannten Systemzustand.

Damit dieser Zentrale Koordinator nicht zu einem Single-Point-of-Failure wird, ist dieser doppelt vorhanden. Ein Mechanismus unabhängig vom Funksystem sorgt dafür dass immer genau einer der beiden Koordinatoren

einschaltet ist. Der Aktive Koordinator wird dabei immer auf Funktion der kompletten Funkstrecke überwacht.

Die im EFR32FG12 „Gecko“ Mikrocontroller integrierte Funkhardware kümmert sich um die Kanalcodierung, das Framing und die Sicherung der übertragenen Daten mit einer Prüfsumme. Die Adressierung der Daten wird in unserem System bereits von der RODOS-Middleware übernommen, das SKITH-Funksystem schickt alle Daten immer als Broadcast an alle Module. Das Protokoll muss sich dann nur noch um die Kollisionsvermeidung kümmern.

Dazu sendet der Koordinator in periodischen Abständen sogenannte Sync-Frames aus. Diese beinhalten eine Liste die für jedes im Satelliten enthaltenes Module eine Zeitslot festlegt in dem es senden darf. Diese Zeitpunkte sind immer relativ zum Empfangszeitpunk des Sync-Frames, was eine hohe Genauigkeit der Slots ermöglicht ohne aufwändige Zeitsynchronisierung. Ein Sync-Frame mit allen dazugehörigen Datenframes der Module bezeichnen wir als SKITH-Zyklus (Siehe Abbildung 4)

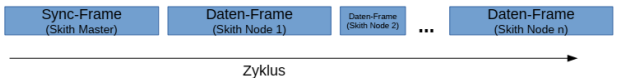

BILD 4: SKITH Protokoll Zyklus

Die Frames die vom Koordinator und den normalen Modulen benutzt werden benutzen die gleiche, in Abbildung 4 dargestellte, Struktur. Die Koordinator-Frames, die Sync-Frames sind, enthalten zusätzlich die Slot-Liste und die aktuelle UTC-Zeit des Koordinators. Die Nutzlast der Frames enthält dann ein oder mehrere RODOS-Middelware-Frames, welches jeweils ein publiziertes Topic enthält. Es werden immer so viele RODOS-Middelware-Frames in einem SKITH-Frame übertragen, wie gerade im Ausgangspuffer vorhanden sind, oder bis der Slot voll ist. Ein Slot muss also immer mindestens so groß sein, um dass größte vom Modul publizierte Topic, aufzunehmen, damit immer eine Übertragung möglich ist.

Im Normalfall weißt der Koordinator jedem eingeschaltetem Modul in jedem Zyklus einen Sendeslot zu.

In Innocube ist der Koordinator gleichzeitig die Power-unit, er weiß also immer genau welche Module eingeschaltet sind und kann so sehr einfach, nur Modulen die eingeschaltet sind, Slots zuweisen. Da wir sehr viele Redundante Module haben, wo immer nur eines eingeschaltet ist, müssen wir nur ca. der Hälfte der vorhandenen Module Slots zuordnen.

Für Module mit niedrigeren Anforderungen an die Datenrate, wie z.B. die Sonnensensoren, muss nicht in jedem Zyklus ein Slot reserviert werden. Auch kann die Slotlänge für bestimmte Module verkürzt werden, wenn diese eine geringere maximale Topicgröße haben.

Der Koordinator verteilt mit jedem Sync-Frame seine aktuelle UTC-Zeit. Weil das Sync-Frame von allen Modulen immer zum genau gleichen Zeitpunkt empfangen wird, kann so mit sehr einfachen Mitteln eine genaue Zeitsynchronisation erreicht werden.

Außerdem wird ungefähr jede Sekunde ein „One Pulse Per Second“-Flag im Sync-Frame gesetzt. Damit lassen sich Vorgänge, die regelmäßig zur genau gleichen Zeit auf mehreren Modulen laufen müssen, an-triggern (z.B. Sonnensensor Messungen). Da 1 Sekunde nicht notwendigerweise ein ganzzahliges Vielfaches der Zykluszeit ist und sich die Zykluszeit auch ändern kann, kann diese Flag nicht für Vorgänge benutzt werden die exakt jede Sekunde laufen müssen.

7. REDUNDANZMANAGEMENT

Für die zuverlässige Funktion des Gesamtsystems ist es wichtig dass immer nur genau einer der Koordinatoren eingeschaltet ist. Wenn beide Eingeschaltet wären, würden beide unkoordiniert Sync-Frames aussenden und dann ist es unklar welchem die übrigen Module folgen würden.

Ebenso muss auch jeder mögliche Ausfall des aktiven Koordinators erkannt werden, um zu dem anderen umzuschalten. Dies beinhaltet sowohl die korrekte Funktion der Software, als auch die Sende- und Empfangs-Einheiten der Funkhardware.

Da die beiden Redundanten Module der PCDU gleichzeitig die Funk-Koordinatoren sind, wird diese Umschaltung mit speziellen Power-Schaltern realisiert.

Beide Module werden über Keep-Off Schalter mit Strom versorgt. Diese Schalter bleiben im Ausgeschalteten Zustand, solange am Steuereingang Periodische Impulse im Abstand von max. 1 Sekunde anliegen. Bleiben diese Impulse aus, schalten die Schalter ein. Ein Koordinator steuert den Schalter des Anderen und umgekehrt.

Solange der aktive Koordinator feststellt, dass er funktioniert, sendet er kontinuierlich Keep-Off Impulse an den Schalter des anderen Koordinators. So ist sichergestellt dass immer nur eines der Module gleichzeitig einschaltet ist.

Diese Impulse werden nur generiert, wenn alle essenziellen Threads regelmäßig „I am alive“-Nachrichten schicken und regelmäßig Funknachrichten von anderen Modulen empfangen werden. So wird die Funktion sowohl der Sende- als auch Empfangsrichtung des Koordinators sichergestellt, weil andere Module nur antworten können, wenn sie vorher ein Sync-Frame vom Koordinator empfangen haben. Dazu muss immer mindestens ein anderes Modul eingeschaltet sein. Das Design sieht vor, dass dies bei mindestem einem der COMMS-Module immer der Fall ist.

Wenn irgendetwas nicht funktioniert bleiben die Keep-Off Impulse aus und das andere Koordinator-Modul startet. Diese generiert nun selbst Keep-Off Impulse und schaltet das erste ab.

Beim Einschalten der Stromversorgung z.B. nach Separation starten beide Koordinator-Module gleichzeitig. Der erste Keep-Off Impuls wird zufällig um ein paar Millisekunden Verzögert. So wird eines der beiden Module zufällig bestimmt und keines bevorzugt.