NASA Operational Simulator for Small Satellites (NOS3)

1) NOS3

NASA Açık Kaynak Anlaşması aracılığıyla erişilebilen, küçük uydular için açık kaynaklı, yalnızca yazılımdan oluşan bir test ortamıdır. Linux yürütülebilir dosyaları, kütüphaneler ve kaynak kodlarından oluşan bir koleksiyondur. Mevcut simülasyonlar, çeşitli CubeSat'larda kullanılan ticari hazır donanımlara (COTS) dayanmaktadır. NASA Core Flight System (cFS) kullanılarak geliştirilen uçuş yazılımıyla kolayca arayüz oluşturması amaçlanmıştır.

NOS3, uydu görevleri için temel bir mimari görevi görür ve belirli bir göreve uyarlanabilen sağlam ve esnek bir çerçeve sunar. Bileşen tabanlı yapısı, yazılım modüllerinin entegrasyonunu kolaylaştırarak geliştiricilerin uzay aracı davranışını simüle etmelerine, enstrüman arayüzlerini test etmelerine ve yazılım/donanım entegrasyonunu gerçekleştirmelerine olanak tanır. Açık kaynaklı bir platform olan NOS3, ekiplere göre görev gereksinimlerine uyum sağlamak için esnek bir temel sağlayan değerli bir referans görevi görür.

2) NOS Engine

NOS Engine, simülasyonlarda kullanılmak üzere özel olarak tasarlanmış bir mesaj aktarım ara yazılımıdır. Modüler tasarımıyla, kütüphane belirli iletişim protokollerini simüle etmek için genişletilebilen güçlü bir çekirdek katman sağlar. Zaman senkronizasyonu, veri işleme ve hata enjeksiyonu gibi gelişmiş özellikleriyle NOS Engine, simülasyon parçalarını bağlamak ve test etmek için hızlı, esnek ve yeniden kullanılabilir bir sistem sunar.

NOS Engine bir kütüphane olduğu için, birincil kullanıcı arayüzü API'dir. Geliştiriciler, NOS Engine'i bir simülasyon içindeki diğer yazılım bileşenlerine entegre etmek için API'yi kullanacaktır. Desteklenen birincil programlama dilleri C ve C++'dır. İkincil hedef, API'yi daha geniş bir kitleye açarak diğer dillere(Python, Java gibi) bağlamalar sağlamaktır.

3) cFS (Core Flight System)

NASA'nın geliştirdiği açık kaynaklı bir uçuş yazılım çatısıdır. cFS aslında bir işletim sistemi gibi değil işletim sistemi üzerinde çalışan bir yazılım mimarisidir.

4) COSMOS

Ball Aerospace tarafından geliştirilen açık kaynaklı bir yer istasyonu (ground station) yazılımıdır. Haberleşmede kullanılır. Bizim sistemimizde COSMOS; komut gönderme ve telemetri alma amacıyla kullanılmaktadır. Bunun için cFS altyapısı ile birlikte çalışır. COSMOS, cFS uygulamalarından çıkan telemetrileri UDP/TCP soketleri üzerinden alır ve kullanıcıya grafiksel arayüzde gösterir. Aynı şekilde, operatörün arayüz üzerinden gönderdiği komutlar da yine UDP/TCP aracılığıyla cFS'ye iletilir.Böylece yer istasyonunda operatör – uçuş bilgisayarı arasında çift yönlü (bidirectional) bir haberleşme sağlanmış olur. COSMOS ayrıca telemetri paketlerini kaydedebilir, script desteği ile otomatik testler çalıştırabilir ve hata ayıklama süreçlerinde kolaylık sağlar.

5) 42 Dynamics Simulator

NASA tarafından geliştirilmiş açık kaynaklı bir yazılım olup uzay araçlarının yörünge ve tutum dinamiklerinin simülasyonu için kullanılmaktadır. Yazılım, çevresel etkiler (yerçekimi, manyetik alan, güneş ışınımı basıncı vb.) ile sensör ve aktüatör modellerini dikkate alarak gerçekçi bir benzetim ortamı sunmaktadır. Tekli veya çoklu uydu senaryoları desteklenmekte, gerçek zamanlı ya da hızlandırılmış çalışma seçenekleri sağlanmaktadır.

cFS (core Flight System) ile TCP/UDP üzerinden entegre edilebilmesi sayesinde uçuş yazılımlarının yerde test edilmesine olanak tanır. COSMOS gibi yer istasyonu yazılımlarıyla birlikte kullanıldığında telemetri alma ve komut gönderme işlemleri simülasyon ortamında gerçekleştirilebilmektedir.

6) cFE (Core Flight Executive)

NASA tarafından geliştirilen cFS mimarisinin çekirdek bileşenidir. Gerçek zamanlı görev yazılımlarının çalıştırılmasını sağlayan framework olarak görev yapar. Uçuş bilgisayarında farklı uygulamaların yönetilmesi, sistem kaynaklarının paylaşılması ve görevler arası iletişimin sağlanması cFE'nin temel işlevlerindendir.Görev zamanlayıcı, event servisi, Software Bus ve telemetri/komut servislerini içermektedir. Böylece farklı uygulamaların birbirinden bağımsız fakat ortak bir altyapı üzerinde çalışması sağlanır.

- **ES (Executive Services):**

  Uygulama ve görev (task) yönetimi: Uygulamaları başlatır/durdurur/yeniden başlatır, öncelik ve yığın ayarı yapar.

  Reset kipleri: Processor Reset (RAM korunur) ve **Power-On Reset** (temiz başlangıç).

  Performans ölçümü, bellek/heap takibi.

- **SB (Software Bus):**

  Publish/subscribe modeli: Publish/Subscribe mesaj omurgası ile mesaj taşır. (CAN/Ethernet'teki mesaj yolu gibi, ama yazılım içi.) App'ler Message ID (MID)'lere abone olur.

  API örnekleri: `CFE_SB_TransmitMsg()` (yayınla), `CFE_SB_ReceiveBuffer()` (pipe'tan al).

  Kuyruk/pipe yönetimi, MID abonelikleri, CCSDS başlıklı paket akışı.

- **EVS (Event Services):**

  Loglama ve hata mesajları: INFO/WARNING/ERROR/CRITICAL olayları üretir, oran sınırlama ve kimliklendirme (Event ID) ile log/telemetri gönderimi.

- **TIME:**

  Zaman eşitleme: Zaman kaynağı (GPS/1PPS/tone) ile senkronizasyon, paketlere doğru timestamp basımı, zaman kalitesi bayrakları

- **TBL:**

  Konfigürasyon tabloları: Eşikler (LC), planlar (SCH/SC), haritalar (CAN ID↔MID), parametreler (payload) gibi tabloların yönetimini sağlar.

  Uçuş sırasında tablo güncelleme/doğrulama yaparak tablo yönetimi sağlar. (yapılandırmalar uçuş sırasında güncellenebilir)

Bunu bir “Windows işletim sistemi” gibi düşünürsek; ES = Görev Yöneticisi, SB = Uygulamalar arası mesajlaşma kanalı, EVS = Hata log’u, TIME = Saat, TBL = Ayarlar klasörü.

6.1 cFE İçinde Görev (Task) Haberleşmesi

cFE'de görevler (tasks) ve uygulamalar (applications), birbirleriyle doğrudan fonksiyon çağrısı yapmak veya global değişken paylaşmak yerine, mesajlaşma (message passing) yöntemiyle haberleşir. Bu haberleşmenin merkezinde, bir posta kutusu veya veri yolu görevi gören Software Bus (SB) ve her mesajı benzersiz şekilde tanımlayan Message ID (MID) bulunur. Bu yapı, modülerlik, güvenilirlik ve bağımsız geliştirmeyi sağlar.

Temel Bileşenler

a) SB (Software Bus):

cFE'nin servislerinden biridir. Görevlerin mesaj yayınlamasını (publish) ve belirli mesajlara abone olmasını (subscribe) yöneten bir dağıtım mekanizmasıdır. Bir görev bir mesajı SB'ye "yayınladığında", o mesaja abone olan tüm görevlere SB o mesajın bir kopyasını iletir.

b) MID (Message ID):

Her mesajın benzersiz kimliğidir. MID, mesajın içeriğini, amacını ve hangi görevin onu işleyeceğini belirlemek için kullanılır. Genellikle `#define` direktifleri ile header dosyalarında tanımlanırlar.

Cmd MID (Komut MID): Bir görevin yapması gereken bir işi belirten komut mesajlarının kimliğidir. Örn: "Vana A'yı aç", "Telemetri paketini gönder".

Tlm MID (Telemetri MID): Bir görevin durum bilgisini veya ölçüm verisini içeren telemetri (durum) mesajlarının kimliğidir. Örn: "Sıcaklık verisi", "Vana A'nın mevcut durumu".

c) Messages:

cFE'de iletilen veri paketleridir. İki ana türü vardır:

Komut Paketleri (Command Packets): Belirli bir yapıya (*CCSDS_CommandPacket_t)* sahiptirler. Cmd MID'leri bu paketleri tanımlar.

Telemetri Paketleri (Telemetry Packets): Belirli bir yapıya (*CCSDS_TelemetryPacket_t*) sahiptirler. Tlm MID'leri bu paketleri tanımlar.

6.2 Haberleşme Süreci: Publish/Subscribe Modeli

1. Subscription:

Bir görev, ilgilendiği bir mesajı alabilmek için SB'ye kendini abone olarak kaydeder. Bu işlem sırasında hangi MID'e abone olacağını belirtir.

Örn: *HK_App* (Housekeeping Uygulaması), *SENSOR_APP_TLM_MID*``'e abone olur. Artık SB, bu MID ile yayınlanan her mesajın bir kopyasını *HK_App*'e iletecektir.

2. Publishing

Bir görev bir mesaj oluşturduğunda, onu SB'ye gönderir. Gönderirken mesajın MID'ini de belirtir.

Örn: *Sensor_App* yeni bir sıcaklık okuması yaptı. Bu veriyi içeren bir telemetri paketi hazırlar ve bu paketi *SENSOR_APP_TLM_MID* kimliği ile SB'ye yayınlar.

3. Mesajın Yönlendirilmesi (Routing)

SB, gelen mesajın MID'ine bakar. Bu MID'e abone olan tüm görevlerin listesini içeren kendi abonelik subscription table kontrol eder.

4. Mesajın Teslimi (Delivery)

SB, abone olan her görev için, mesajın bir kopyasını o görevin mesaj kuyruğuna koyar. Her görevin kendisine ait bir mesaj kuyruğu vardır.

5. Mesajın İşlenmesi (Processing)

Abone olan görevler, kendi mesaj kuyruklarını sürekli kontrol eder. Kuyruğa yeni bir mesaj geldiğinde, görev mesajı alır, MID'ine bakar ve içeriğini işlemek üzere ilgili fonksiyonuna yönlendirir.

Örn: *HK_App*, kuyruğundaki *SENSOR_APP_TLM_MID* mesajını alır. MID'i tanır ve *ProcessSensorData* fonksiyonunu çağırarak içindeki sıcaklık verisini işler.

6.3 OSAL (Operating System Abstraction Layer)

cFE'nin üzerinde çalıştığı işletim sisteminden bağımsız olabilmesi için OSAL (Operating System Abstraction Layer) kullanılır. OSAL, gerçek zamanlı işletim sistemi (VxWorks, RTEMS, Linux gibi) ile cFE arasındaki soyutlama katmanıdır. Bu sayede cFE uygulamaları, farklı işletim sistemlerinde herhangi bir değişiklik yapılmadan çalıştırılabilir.

6.4 PNP (Plug and Play)

cFS içerisinde PnP (Plug and Play) yaklaşımı, yeni bir uygulamanın veya görev modülünün sisteme kolayca eklenebilmesini ifade etmektedir. Yani bir modül eklendiğinde, gerekli servislerle otomatik olarak entegre olur ve yazılım yeniden derlenmeden sisteme dahil edilebilir. Bu durum görev esnekliği ve modülerlik sağlar.

6.5 Görev Uygulamaları (Apps)

HK, SC, SCH, LC, DS, FM, CF/CFDP, HS vb. cFE üzerinde çalışan görev yazılımlarıdır. Bu yazılımların bazıları standart bazıları ise özel görev yazılımlarıdır.

- **HK (Housekeeping):** Sağlık verilerini toplar (sıcaklık, voltaj, akım).
- **SC (Stored Command):** Önceden yüklenen komutları belirli zamanda çalıştırır.
- **SCH (Scheduler):** Düzenli iş tetikleme (örn. her 1 saniyede bir sıcaklık oku).
- **LC (Limit Checker):** Eşik kontrolü (örn. sıcaklık > 50 °C olursa uyar).
- **DS (Data Storage):** Verileri kaydetme.
- **FM (File Manager):** Dosya sisteminde işlem yapma.
- **CF (File Transfer):** Dosya indirme/gönderme (CCSDS CFDP protokolü ile).
- **HS (Health & Safety):** Uygulama durumlarını kontrol etme.

Bunlar, uydu yazılımının temel uygulamalarıdır.

Her uyduya özel:

- **CI/TO (Command Ingest / Telemetry Output):** Yer istasyonu ↔ uydu arası komut ve telemetri yönetimini sağlar.
- **1553, SpaceWire, CAN sürücüleri:** Donanım veri yolu üzerinden iletişimi sağlar.
- **EDAC (Error Detection and Correction):** Bellek hatalarını düzeltir.
- **Recorder Manager:** Verileri geçici bellekte kaydetme.

Bunlar, uydu yazılımının görev-özel uygulamalarıdır.

6.6 Telemetry (TM) ve Telecommand (TC) (Frame/Packet Yapısı)

Uzay aracı ile yer istasyonu arasındaki haberleşme, TM ve TC verileri üzerinden gerçekleşmektedir.

- **TM (Telemetry):** Uzay aracından yere gönderilen ölçüm, durum ve sağlık verileridir.
- **TC (Telecommand):** Yer istasyonundan uzay aracına gönderilen komutlardır.

Veriler, önce paket seviyesinde oluşturulur. cFE uygulamaları tarafından üretilen TM paketleri veya alınan TC paketleri daha sonra CCSDS (Consultative Committee for Space Data Systems) standardına uygun şekilde frame yapısına yerleştirilir. Frame, iletişim linkinde kullanılan fiziksel aktarım birimidir.

6.7 CCSDS (Consultative Committee for Space Data Systems)

Uluslararası standartlar belirleyen bir komitedir. Dünya'daki yer istasyonları ile uzaydaki uyduların ortak dilde konuşabilmesi için paketleme ve iletişim kuralları sunar.

6.8 UDP Tabanlı Haberleşme

Yer istasyonu yazılımı cFS ile haberleşmek için genellikle UDP (User Datagram Protocol) tabanlı bağlantı kullanır. UDP, düşük gecikme ve basit yapı sağladığından gerçek zamanlı komut/telemetri iletişimi için uygundur. Böylece yer istasyonundan cFE'ye TC paketleri gönderilir, cFE tarafından üretilen TM paketleri ise yer istasyonuna iletilir. Bu yapı sayesinde sistem; gerçek donanım, simülasyon ortamı veya hibrit test düzeneklerinde aynı iletişim altyapısı üzerinden çalıştırılabilmektedir.