ADIS16475 IMU Sensörü ile FreeRTOS Tabanlı Telemetri Sistemi

Bu projenin amacı, ADIS16475 isimli yüksek hassasiyetli IMU (Inertial Measurement Unit) sensöründen elde edilen ivme ve açısal hız (gyro) verilerini, STM32F4 serisi bir mikrodenetleyici kullanarak SPI protokolü ile okumak ve bu verileri FreeRTOS tabanlı bir sistemde UART ve CAN arayüzleri üzerinden dış dünyaya iletmektir.

Projede sıcaklık verisi dahili ADC üzerinden ölçülmekte ve bu veri ile birlikte tüm sensör verileri bir telemetri paketi formatında toplanarak düzenli aralıklarla gönderilmektedir. Aynı zamanda sistemde, hata durumlarını (örneğin SPI haberleşme hatası, sıcaklık limiti aşımı gibi) tespit eden ve bu hataları telemetri paketine yansıtan bir yapı da mevcuttur.

Bu yapı ile gömülü sistemlerde çok görevli (multitasking), eşzamanlı (concurrent), hata toleranslı ve modüler bir haberleşme sistemi kurulması hedeflenmiştir.

Donanım Bileşenleri :

- STM32F407 Discovery Board
- ADIS16475 IMU
- Internal Temperature ADC Channel
- CAN Transceiver
- USB-TTL (Uart to PC)
- Jumper Wires

Bu projede STM32F4 serisi mikrodenetleyici ile çeşitli çevresel birimlerle haberleşme kurulmuştur. Aşağıda kullanılan her bir haberleşme protokolü için pin bağlantıları verilmiştir.

UART (MCU → USB to TTL Dönüştürücü)

İŞLEV	MCU Pin	Açıklama
UART TX	PA2	MCU'dan veri çıkışı
UART RX	PA3	MCU'ya veri girişi

SPI (MCU ↔ ADIS16475 IMU Sensörü)

İŞLEV	MCU Pin	Açıklama
CS	PA4	Chip Select
SCK	PA5	Saat Sinyali
MISO	PA6	Sensörden veri alımı
MOSI	PA7	Sensöre veri gönderimi

CAN (MCU ↔ CAN Transceiver)

İŞLEV	MCU Pin	Açıklama
CAN RX	PD0	CAN hattından veri alımı
CAN TX	PD1	CAN hattına veri gönderimi

- **Transceiver:** MCP2551 veya benzeri bir CAN transceiver entegresi gereklidir.
- **Not:** Doğru çalışabilmesi için CANH-CANL hatlarına 120Ω sonlandırma direnci bağlanmalıdır.

Yazılım Mimarisi :

Proje, **FreeRTOS tabanlı çok görevli (multitasking) bir mimari**ye sahiptir. Tüm görevler (task) bağımsız çalışmakta, sistem kaynaklarına senkronize erişim sağlanmakta ve tüm çevresel birimler modüler fonksiyonlarla yönetilmektedir.

Kullanılan Yapılar:

RTOS Görevleri :

- sensor_task: IMU verilerini SPI üzerinden okur
- temp_task: MCU iç sıcaklık sensörünü ADC ile okur
- comms_task: UART ve CAN haberleşmesini yürütür
- error_monitor_task: sistem hatalarını gözlemler ve flag'leri günceller

Semaphore (Mutex):

- xTelemetryMutex isimli mutex, telemetry_packet veri yapısına erişimi güvenli hale getirir

- Aynı anda birden fazla görevin bu yapıya müdahale etmesi engellenir (race condition önlenir)

Veri Akışı (Basit Özet):

- sensor_task IMU'dan veri okur ve telemetry_packet içine yazar
- temp_task sıcaklık ölçümünü ekler
- comms_task tüm veriyi UART ve CAN üzerinden gönderir
- Herhangi bir hata oluşursa ilgili flag setlenir, error_monitor_task bu hatayı işler

Haberleşme Protokolleri

Bu projede dış dünyaya veri iletimi için UART ve CAN protokolleri birlikte kullanılmıştır. Her iki haberleşme kanalı da düzenli aralıklarla telemetri verilerini iletmek amacıyla comms_task görevinde çalıştırılmaktadır.

UART (Universal Asynchronous Receiver Transmitter) :

- Kullanım Amacı: Telemetri verilerinin bilgisayara veya bir terminal cihazına iletilmesi
- **Baudrate:** 115200
- **Kullanılan UART:** USART2
- **Veri Formatı:** TelemetryPacket_t yapısı, ham olarak HAL_UART_Transmit() ile gönderilir
- **Veri Periyodu:** 1 saniyede 1 paket

CAN (Controller Area Network) :

- Kullanım Amacı: Aynı verilerin CAN hattındaki diğer cihazlara iletilmesi
- CAN ID: 0x123
- Bölünmüş Gönderim: Telemetri paketi 8 byte'lık parçalara bölünerek gönderilir (çünkü CAN DLC max. 8 byte)
- Zamanlama: UART ile aynı anda, her saniyede 1 kez

Port ID ile Ayrım :

Gönderilen telemetri verisinin hangi haberleşme protokolüyle yollandığı `port_id` alanı ile belirlenir:

- **0x01 → UART**
- **0x02 → CAN**

Bu sayede alıcı taraf, gelen veriyi hangi kanaldan aldığını anlayabilir.

Hata Senaryoları ve Alınan Önlemler :

1. SPI Haberleşme Zaman Aşımı (Timeout) :

ADIS16475 sensörüyle SPI haberleşmesi sırasında veri alınamazsa veya işlem beklenenden uzun sürerse sistem kilitlenebilir.

Çözüm :

- ADIS16475_ReadRegister() içinde HAL_SPI_Transmit() ve HAL_SPI_TransmitReceive() işlemleri kontrol edilir.
- Hata durumunda spi_timeout_flag setlenir.
- error_monitor_task, bu flag'i telemetri paketine hata olarak işler.

2. Sıcaklık Limit Aşım

MCU iç sıcaklığı 70°C üzerine çıkarsa veya 0°C altına düşerse sistem zarar görebilir.

Çözüm :

- temp_task içinde sıcaklık değeri ADC üzerinden okunur ve kontrol edilir.
- Limit dışı sıcaklık algılanırsa temp_overunder_flag setlenir.
- Bu bilgi error_monitor_task ile işlenir ve telemetri paketine yansıtılır.

3. Görevler Arası Veri Yarışı (Race Condition)

Birden fazla görev aynı anda telemetry_packet yapısını okur/yazarsa veri tutarsızlığı oluşabilir.

Çözüm:

- telemetry_packet, volatile olarak tanımlanmıştır.
- Görevler arası erişim xTelemetryMutex mutex'i ile kontrol altına alınmıştır.
- Bu sayede her an sadece bir görev pakete erişebilir.

Geliştirme Ortamı :

Proje aşağıdaki yazılım ve donanım ortamında geliştirilmiştir:

IDE: STM32CubeIDE (v1.xx veya üzeri önerilir)

Programlama Dili: C

RTOS: FreeRTOS

Toolchain: GCC

Donanım Soyutlama Katmanı : HAL (Hardware Abstraction Layer)

Debug Aracı: ST-Link