Progetto sistemi embedded anno accademico 2021-2022

Si vuole progettare un sottosistema di acquisizione dati facente parte di un sistema software di controllo di volo per un drone ad ala rotante. Il drone integra la scheda Nucleo F767ZI e lo shield di espansione X-NUCLEO-IKS01A2, ed il sistema di controllo di volo esegue sull'MCU della scheda Nucleo. Il sottosistema di acquisizione dati riceve gli input da alcuni dei sensori sullo shield IKS01A2, li converte in grandezze fisiche, e li produce in output in maniera che possano essere consumati da altri tre sottosistemi del sistema di controllo di volo, che si suppongono già sviluppati e di cui si riporta una breve descrizione:

- Sottosistema di controllo e stabilizzazione motori: controlla i motori per stabilizzare il volo del drone in condizioni stazionarie e per far reagire il drone ai comandi del pilota in maniera che il drone si sposti nella direzione che il pilota ha indicato; l'azione di controllo viene calcolata passando gli input attraverso una sequenza di filtri passa-basso e elimina-banda, e passando gli input filtrati ad opportuni controllori PID.
- Sottosistema di calcolo dell'assetto: calcola l'orientamento assoluto del drone rispetto al riferimento north-east-down (NED) utilizzando lo stimatore di Mahony.
- Sottosistema di calcolo dell'altitudine: calcola l'altitudine del drone rispetto al livello del mare.

Il sottosistema di acquisizione dati deve funzionare in tempo reale, in maniera da soddisfare una serie di vincoli temporali, di sottosistema e di sistema, che saranno successivamente specificati.
Il sottosistema di acquisizione dati interagisce con i seguenti sensori integrati nello shield IKS01A2:

- LSM6DSL: accelerometro e giroscopio;
- LSM303AGR: accelerometro e magnetometro;
- LPS22HB: pressione barometrica.

Il dato tridimensionale del giroscopio deve essere acquisito con frequenza ≥5 KHz, convertito nel vettore tridimensionale delle velocità angolari (rad s^{-1}), e prodotto in output. Questo viene consumato in input dal sottosistema di controllo e stabilizzazione motori e dal sottosistema di calcolo dell'assetto.
I dati tridimensionali di entrambi gli accelerometri devono essere acquisiti con frequenza ≥1 KHz, convertiti ciascuno nel vettore tridimensionale delle accelerazioni (m s^{-2}), e prodotti entrambi in output. Inoltre, deve essere prodotto in output anche il vettore ottenuto dalla media dei due vettori prodotti dagli accelerometri. Tali dati di output vengono consumati in input dal sottosistema di calcolo dell'assetto.
Il dato tridimensionale del magnetometro deve essere acquisito con frequenza ≥10 Hz, convertito nel vettore tridimensionale delle densità di flusso magnetico (μT), e prodotto in output. Questo viene consumato in input dal sottosistema di calcolo dell'assetto.
Il dato (scalare) del sensore di pressione barometrica deve essere acquisito con frequenza ≥20 Hz, convertito in pressione (hPa), e prodotto in output. Questo viene consumato in input dal sottosistema di calcolo dell'altitudine.
Il sottosistema di acquisizione dati deve garantire che:

- la frequenza di ciascun output del sottosistema di acquisizione dati sia sufficientemente elevata da far si che, ad ogni ciclo di attivazione periodica, ogni sottosistema che consuma

tali output riceva un valore “fresco”, ossia ottenuto da un campione di dato sensore diverso dal precedente che aveva consumato;
- il sistema di controllo di volo nel suo complesso sia in grado di effettuare i calcoli periodici di controllo e stabilizzazione motori, assetto, ed altitudine entro le rispettive deadline, le quali sono uguali al 95% dei loro rispettivi periodi.

I sottosistemi già implementati sono composti ciascuno da un task periodico. I task operano alle seguenti frequenze, ed hanno i seguenti WCET:

- Task controllo e stabilizzazione motori: frequenza 5 KHz, WCET 90 µsec;
- Task calcolo assetto: frequenza 100 Hz, WCET 120 µsec;
- Task calcolo altitudine: frequenza 40 Hz, WCET 60 µsec.

Il software deve essere sviluppato utilizzando le API ST e il sistema operativo real-time FreeRTOS con scheduling rate monotonic, dando priorità diverse anche ai task con uguale frequenza (non devono esservi due task con la stessa priorità).
Infine, a scopo di debugging si vuole che le grandezze fisiche prodotte in output dal sottosistema di acquisizione dati siano visualizzate sull’SWV ITM Data Console dell’STM32CubeIDE. I dati di output del sottosistema devono essere stampati con una frequenza di circa 0,5 Hz (la stampa di debugging non deve avvenire in modalità real-time), secondo questo formato:

- Prima riga: le tre componenti del vettore tridimensionale delle velocità angolari del giroscopio; le velocità angolari vanno visualizzate con due cifre decimali;
- Riga successiva: le tre componenti del vettore tridimensionale delle accelerazioni del primo accelerometro (dato prodotto dal sensore LSM6DSL); le accelerazioni vanno visualizzate con due cifre decimali;
- Riga successiva: le tre componenti del vettore tridimensionale delle accelerazioni del secondo accelerometro (dato prodotto dal sensore LSM303AGR); le accelerazioni vanno visualizzate con due cifre decimali;
- Riga successiva: le tre componenti del vettore tridimensionale delle accelerazioni (media dei vettori prodotti dai due accelerometri); le accelerazioni vanno visualizzate con due cifre decimali;
- Riga successiva: le tre componenti del vettore tridimensionale delle densità di flusso magnetico del magnetometro; le densità di flusso magnetico vanno visualizzate con due cifre decimali;
- Riga successiva: la pressione del sensore di pressione barometrica; la pressione va visualizzate con una cifra decimale;
- Riga successiva: riga bianca (per separare la successiva stampa).

Richieste:

1. Si progetti e si documenti opportunamente la macchina a stati che illustra il funzionamento dettagliato del sottosistema di acquisizione dati, inclusa la componente di stampa di debugging.
2. Si produca un progetto dei task del sottosistema di acquisizione dati (inclusi il o i task che si occupano della stampa di debugging), e del loro scheduling temporale, e si produca un argomento rigoroso che dimostri che il progetto dello scheduling temporale permette di rispettare tutti i requisiti real-time. A tale scopo, per semplificare il calcolo del WCET del

vostro codice, si faccia l'ipotesi che, nella catena comunicazione microcontrollore / shield X-NUCLEO-IKS01A2 + conversione dei dati letti dai sensori in grandezze fisiche, il tempo dominante sia quello della comunicazione I2C e che tutti gli altri tempi siano trascurabili. Nel caso usiate primitive di sincronizzazione tra task, tenete conto dell'impatto che queste hanno sulle proprietà real-time della soluzione progettata (potete fare eventuali semplificazioni ed assunzioni, sempre ben giustificate). L'argomento che dimostra che il sistema soddisfa i requisiti real-time deve tener conto dell'impatto del/dei task di stampa di debug, oppure va prodotto un argomento che spieghi perché questo/questi task ha/hanno un impatto trascurabile sulle proprietà real-time del sistema.

3. Si produca l'implementazione C, completa e funzionante, del progetto per ambiente STM32CubeIDE. Il codice deve essere "pulito", ossia adeguatamente modularizzato, facente uso dell'information hiding, delle costanti simboliche, dei commenti… Deve essere inoltre messa chiaramente in evidenza la correlazione del codice con la macchina a stati progettata al punto 1. Agli scopi della valutazione della correttezza del codice, se due task / routine di interrupt condividono dati sarà considerato un errore affidarsi alle priorità dei task per garantire la mutua esclusione: dovete invece sempre utilizzare le primitive di sincronizzazione / comunicazione offerte da FreeRTOS, stando attenti ad evitare la priority inversion (e deadlock, starvation…). Notare in particolare che le funzioni dell'HAL ST per la comunicazione I2C non sono thread safe, e quindi se decidete di ripartire la comunicazione con lo shield su più task, questi devono opportunamente coordinarsi per accedere in mutua esclusione alla periferica I2C.

Estensione facoltativa:

Per migliorare il debugging si vuole che, anziché sull'SWV ITM Data Console dell'STM32CubeIDE, le grandezze fisiche prodotte in output dal sottosistema di acquisizione dati siano visualizzate sul display Olimex. Notare quindi che, in questa estensione, non bisogna implementare la stampa sull'SWV ITM Data Console.
L'interazione con il display Olimex deve funzionare in questo modo. Il pulsante freccia destra deve permettere di passare dalla visualizzazione del dato del giroscopio, che è il primo ad essere visualizzato quando il sistema viene avviato (occorre visualizzare le tre componenti della velocità angolare con due cifre decimali), a quella dell'accelerometro (occorre visualizzare le tre componenti dell'accelerazione con due cifre decimali), del magnetometro (occorre visualizzare le tre componenti della densità del flusso magnetico con due cifre decimali), e del sensore di pressione (occorre visualizzare la pressione con una cifra decimale), e poi di nuovo a quella del giroscopio in maniera ciclica. Il pulsante freccia sinistra, analogamente, deve permettere lo scorrimento ciclico nella direzione opposta. I pulsanti freccia in alto e freccia in basso devono permettere di effettuare lo scrolling (sempre in maniera ciclica) tra le tre componenti delle grandezze di tipo vettoriale quando queste sono selezionate, altrimenti sono inattivi. Il pulsante rimanente deve permettere di selezionare ciclicamente il dato dell'accelerometro del sensore LSM6DSL, dell'accelerometro del sensore LSM303AGR, o la media dei dati dei due accelerometri: questo quando il dato dell'accelerometro viene visualizzato, mentre quando vengono visualizzati gli altri dati il pulsante è inattivo. La frequenza di aggiornamento dei dati visualizzati sul display deve essere di circa 0,5 Hz (la visualizzazione periodica dei dati sul display non deve avvenire in modalità real-time). Per evitare interferenze con la comunicazione con i sensori, e per permettere che questa possa avvenire ad una velocità più elevata da quella consentita dalla massima velocità I2C del display (che è piuttosto bassa), si richiede che la comunicazione con il display avvenga via UART.

Chiarimenti sulle richieste per il progetto esteso:

1. La macchina a stati di progetto del sottosistema di acquisizione dati deve includere anche la gestione del display.
2. Il progetto dei task del sistema deve includere anche il/i task di gestione del display; l'argomento che dimostra che il sistema soddisfa i requisiti real-time deve tener conto dell'impatto del/dei task di gestione del display, oppure va prodotto un argomento che spieghi perché questo/questi task ha/hanno un impatto trascurabile sulle proprietà real-time del sistema.
3. L'implementazione C dei task di gestione del display deve rispettare gli stessi requisiti del resto del codice, ossia, deve essere "pulita", bisogna correlare il codice C con la macchina a stati del punto 1, e deve rispettare tutti i vincoli sulla mutua esclusione precedentemente esposti.