Corso di TECNOLOGIE DEI SISTEMI DI CONTROLLO
A.A. 2021/2022, Docente: Ing. Marcello Bonfè

SPUNTI PER PROGETTI DI APPROFONDIMENTO (TESINE FACOLTATIVE)

NOTE GENERALI:

1. **Ogni progetto deve essere PERSONALE, cioè sviluppato e presentato da ciascuno studente in autonomia.**

2. **I progetti qui descritti sono dei suggerimenti, realizzabili con materiale in dotazione presso il Dipartimento. Altri progetti possono essere proposti direttamente dallo studente su propria iniziativa e inventiva. Qualsiasi progetto di tesina dovrà però contenere almeno:**
   - **Uno o più sensori per la misura di grandezze fisiche OPPURE uno o più attuatori (es. motore DC, passo passo, termoriscaldatore, ventola ecc.) OPPURE entrambi!**
   - **Un microcontrollore/DSP/DSC, il cui programma sia sviluppato in un modo che permetta di approfondirne le caratteristiche hardware (preferibilmente tramite configurazione diretta degli SFR).**

3. **Il docente è disponibile per fornire suggerimenti più dettagliati (esempi di codice C, schemi elettrici e datasheet dei componenti specifici), su richiesta specifica dello studente interessato ad uno degli esempi qui proposti.**

4. **PRIMA DI INIZIARE LO SVILUPPO di una tesina, lo studente DOVRA' descrivere al docente il proprio proposito di progetto, sia estratto tra quelli qui elencati sia di propria inventiva, sia al fine di valutarne l'applicabilità come tesina per il corso (Vedi punto 1 sui contenuti richiesti!) sia al fine di permettere al docente di fornire ulteriori suggerimenti utili (vedi punto 3).**

5. **IN SEDE D'ESAME NON VERRANNO VALUTATE TESINE NON "CONCORDATE" PRIMA DELL'INIZIO DEL PROGETTO STESSO. Inoltre, lo studente che si presenti all'esame con un progetto dovrà segnalarlo anche in fase di iscrizione all'appello (ammessa solo via web!) tra le note.**

6. **Il progetto dovrà essere accompagnato da una breve relazione: poche pagine con schema a blocchi o schematico dettagliato, listato programma e commenti personali.**

1. **Implementazione del controllo PID fixed-point per il motore DC (per dsPIC33 o altri microcontrollori/DSP/DSC)**

Strumenti utilizzabili (per dsPIC33):

- MPLAB X con compilatore XC16
- Combinazione PICDEM Mechatronics + Microstick II disponibile in Laboratorio
- Code Example CE019 (per dsPIC30F, adattabile a dsPIC33): http://ww1.microchip.com/downloads/en/DeviceDoc/CE019_PID.zip
- Documentazione ulteriore nella cartella "docs\dsp_lib" del percorso di installazione di XC16 (es. C:\Program files\Microchip\xc16\v\.50)

Il compilatore XC16 fornisce una libreria specifica per sfruttare pienamente il DSP engine dei dsPIC30/33, con funzioni implementate in assembler ma con prototipo (i.e. header file) per richiamare tali funzioni in linguaggio C. In particolare le funzioni per il controllo PID permettono di eseguire l'algoritmo di controllo in soli 33 cicli di istruzione, con uscita codificata in formato Q15 (a 16 bit), quindi **scalata tra -1.0 e 0.9999 (considerati anche come valori di saturazione per una logica anti-windup "implicita")**.

L'utilizzo pratico del PID "fractional" richiede quindi di scalare opportunamente l'errore di controllo (es. differenza tra velocità desiderata e misurata, in RPM), in modo che tale che prima della chiamata del PID sia compreso tra -1.0 e 0.9999 (con ipotesi ragionevoli sui valori massimi/minimi delle velocità misurabili..), poi scalare l'uscita in modo tale che il valore -1.0 corrisponda al minimo della variabile di comando (es. 0% di duty cycle del comando PWM per il convertitore di potenza del motore) e 0.9999 al massimo (es. 100% del duty cycle).

2. **Implementazione di un'applicazione (es. uno degli esercizi mostrati nel corso o una relativa variante) tramite "Embedded target" Matlab/Simulink (per dsPIC33 o altri microcontrollori/DSP/DSC)**

Strumenti utilizzabili (per dsPIC33):

- MPLAB X con compilatore XC16
- Combinazione PICDEM Mechatronics + Microstick II disponibile in Laboratorio
- Matlab/Simulink con Matlab Coder, Simulink Coder e Embedded Coder
- MPLAB Device Blocks per Simulink, scaricabile tramite https://microchipdeveloper.com/simulink:start oppure https://github.com/LubinKerhuel/MPLAB-Device-Blocks-for-Simulink

Tramite i blocchi dell'MPLAB Device Blocks è possibile creare l'applicazione completa per un dsPIC30/33 con uno schema Simulink. Tale schema deve includere alcuni blocchi di preparazione del codice, cioè blocchi senza ingressi o uscite ma con un menu contestuale dedicato alla configurazione del processore (es. clock, configuration bits, ecc.), i blocchi di configurazione delle periferiche (es. ADC, McPWM ecc.), blocchi direzionabili DAGLI ingressi analogici/digitali e VERSO le uscite fisiche digitali/PWM del dsPIC e qualunque altro blocco di elaborazione matematica supportato da Simulink.

Ad esempio, sarebbe possibile realizzare un filtraggio di segnali analogici acquisiti tramite ADC del dsPIC oppure modulare i segnali PWM in modo da ricreare dei pattern sinusoidali (vedi https://www.eeeguide.com/sinusoidal-pulse-modulation/) oppure ancora ricreare il controllo closed-loop già descritto al punto 1, ma in modo interamente grafico.

Avendo a disposizione una scheda target supportata come Programmer/Debugger (es. Microstick II o un programmatore della famiglia PicKit) sarebbe possibile anche trasferire direttamente l'applicazione compilata usando solamente comandi di Matlab/Simulink. Nel caso di utilizzo di Proteus VSM bisognerebbe invece generare SOLO il codice sorgente (da Simulink: Apps → Embedded Coder → Generate Code invece di Build) per poi importarlo in un progetto MPLAB X creato ad-hoc.

NOTA: allargando il contesto ad altri microcontrollori/DSP/DSC si trovano molti altri "blockset" o toolbox specifici per la generazione automatica di codice C/C++ ottimizzato per questi target. Si potrebbe quindi sviluppare anche una tesina relativa all'uso di tali strumenti, per chi abbia già o voglia procurarsi una propria scheda con hardware differente da dsPIC33. Si veda https://it.mathworks.com/discovery/simulink-embedded-hardware.html per una panoramica completa dell'hardware supportato.

3. Sviluppo di un'applicazione multi-tasking con FreeRTOS (per dsPIC33 o altri microcontrollori/DSP/DSC)

Strumenti utilizzabili (per dsPIC33):

- MPLAB X con compilatore XC16
- Combinazione PICDEM Mechatronics + Microstick II disponibile in Laboratorio
- Porting del sistema operativo per applicazioni embedded FreeRTOS: https://www.freertos.org/portpic24_dspic.html
- MPLAB X Add-on FreeRTOS Viewer (per visualizzare le caratteristiche dei task programmati)

FreeRTOS è un insieme di librerie con funzioni di sistema operativo compatibili con le limitate capacità di memoria e calcolo dei sistemi embedded, per i quali però fornisce primitive relativamente semplici per la definizione di:

- Task periodici
- Sincronizzazione fra i Task (es. semafori/mutex)
- Gestione di comunicazioni inter-task (code, stream buffers, ecc.)

Un primo obiettivo potrebbe essere quello di testare esempi forniti con il codice sorgente del Porting per dsPIC/PIC24 adattandoli alle caratteristiche dei dsPIC33 utilizzato in laboratorio durante le esercitazioni del corso. Un obiettivo più ambizioso, che richiede però anche l'uso di hardware reperito autonomamente dallo studente, potrebbe essere quello di sviluppare una piccola applicazione IoT grazie all'integrazione di servizi forniti da Amazon con FreeRTOS: https://github.com/MicrochipTech/amazon-freertos

4. Controllo "microstepping" di un motore passo-passo o *stepper* (per dsPIC33 o altri microcontrollori/DSP/DSC)

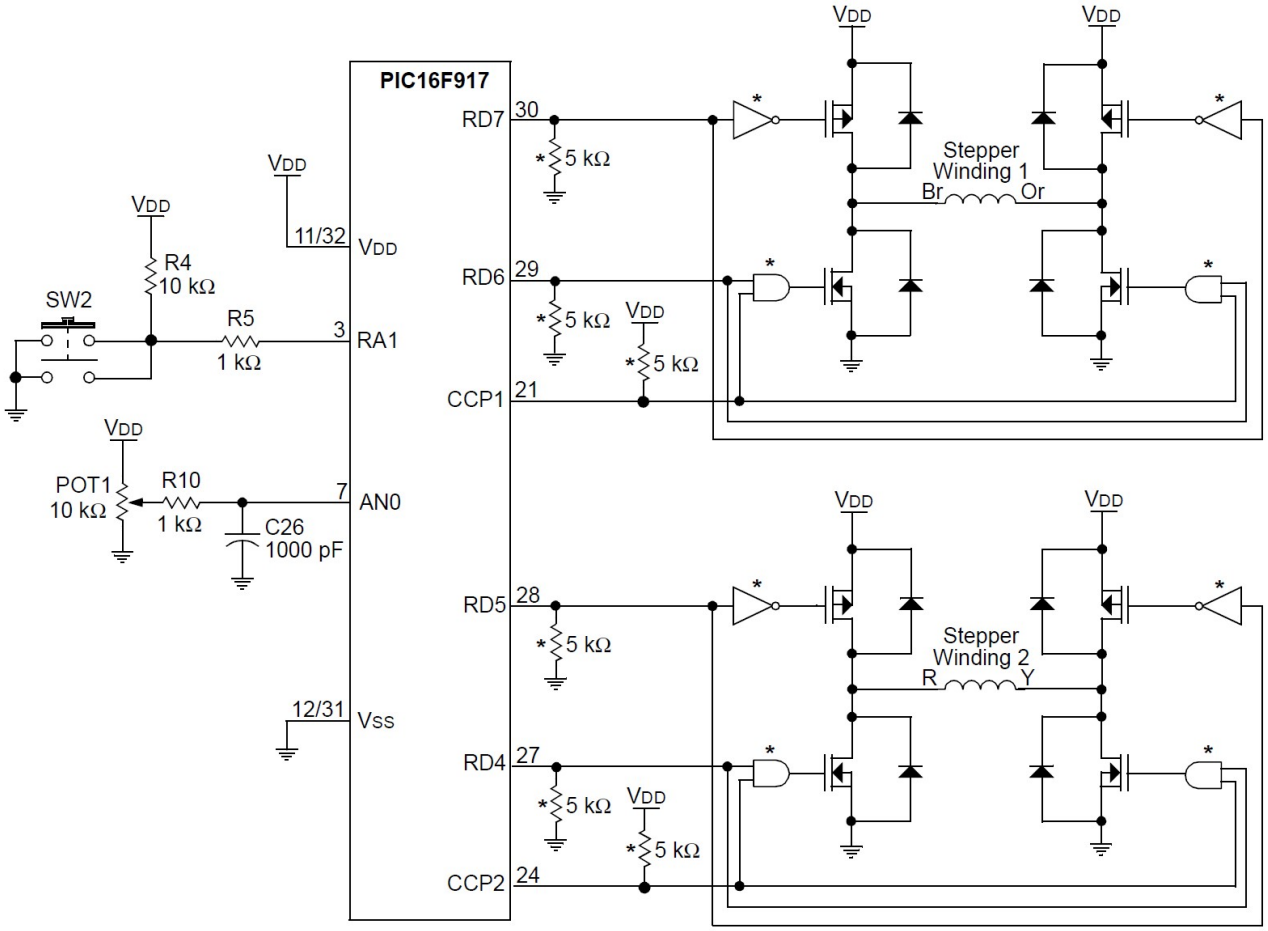
Strumenti utilizzabili (per dsPIC33):

- MPLAB X con compilatore XC16
- Combinazione PICDEM Mechatronics + Microstick II disponibile in Laboratorio

Il programma del microcontrollore deve permettere il movimento orario/antiorario del motore stepper in dotazione, connettendo opportunamente il dsPIC ai seguenti dispositivi installati sulla PICDEM Mechatronics:

- POT1 (J4), riferimento velocità motore
- SW2 (J4), pulsante per richiesta di rotazione oraria
- SW3 (J4), pulsante per richiesta di rotazione antioraria.
- P1 (J1), comando transistor del convertitore di potenza
- P2 (J1), comando transistor del convertitore di potenza
- P3 (J1), comando transistor del convertitore di potenza
- P4 (J1), comando transistor del convertitore di potenza
- PWM1 (J1), modulazione di una coppia di transistor del convertitore di potenza
- PWM2 (J1), modulazione di una coppia di transistor del convertitore di potenza

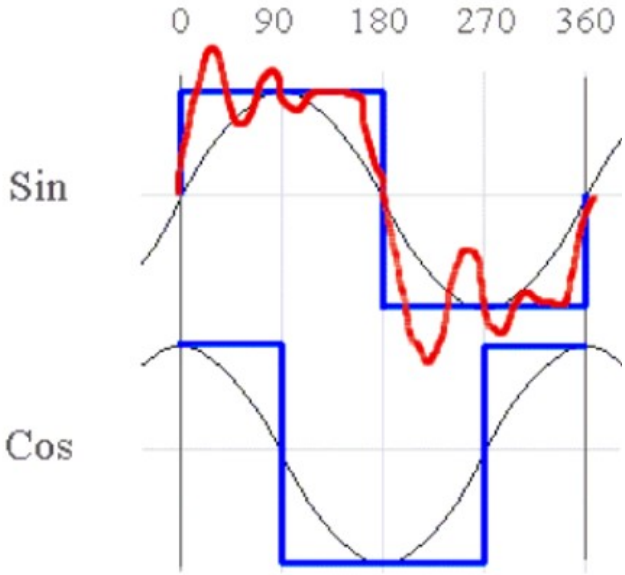
Per effettuare la rotazione del motore, occorre anzitutto attivare/disattivare i comandi P1-2-3-4 secondo una determinata sequenza, in base allo schema proposto dalla User Guide originale per la PICDEM Mechatronics e riportato per riferimento nel seguito:



NOTA BENE: per l'adattamento dello schema alla combinazione con Microstick II è necessario chiedere suggerimenti specifici al docente.

Grazie alla modulazione PWM, è possibile regolare l'andamento della corrente nelle fasi del motore in modo sinusoidale, anzichè ad onda quadra come normalmente effettuato per il pilotaggio in modalità mezzo-passo o passo intero. In questo modo, si può ottenere un movimento molto fluido del motore, poco rumoroso e stabile, in quanto vengono evitate le problematiche di risonanza elettromeccanica tipiche del movimento "standard" dei motori stepper, come illustrato dalla seguente figura (in blu, sequenza di attivazione fase passo

intero, in rosso, corrente reale di fase, in nero, corrente ottenibile con la modulazione PWM sinusoidale):



Per ottenere la corrente sinusoidale, è necessario introdurre, in aggiunta all'esecuzione della sequenza di attivazione dei pin di comando P1 – P2 – P3 – P4, una sequenza che faccia variare il duty cycle del PWM, ovviamente in modo sincrono alla sequenza delle fasi. Tale sequenza può essere ottenuta definendo un vettore di valori che descrivano un quarto di periodo di una sinusoide, ad esempio con 8 bit di risoluzione del PWM: {0, 57, 111, 159, 199, 230, 249, 255} nella fase crescente, o {255, 249, 230, 199, 159, 111, 57, 0} nella fase decrescente. Per una descrizione completa della logica di pilotaggio, si faccia sempre riferimento alle tabelle disponibili sul web: http://www.vincenzov.net/tutorial/passopasso/elettronicabipolare.htm, **oltre che ai suggerimenti specifici del docente per la corretta configurazione d'uso della scheda PICDEM Mechatronics con l'adattatore per Microchip Microstick II**.

5. Migrazione di un'applicazione Arduino per un target dsPIC33

Strumenti utilizzabili:

- MPLAB X con compilatore XC16
- Combinazione PICDEM Mechatronics + Microstick II disponibile in Laboratorio
- Esempi di librerie "Arduino-like" per altri microcontrollori/DSC Microchip:
  https://circuitcellar.com/cc-blog/execute-open-source-arduino-code-in-a-pic-microcontroller-using-the-mplab-ide/
  https://web.archive.org/web/20191020103950/http://kibacorp.com/free-downloads
  (Link in fondo alla pagina, "Circuit Cellar Article – Arduino Librari Code..")
  https://chipkit.net/wiki/index.php?title=MPLABX_Importer

Gli esempi citati forniscono strumenti per convertire "sketch" per Arduino in programmi compilabile in ambiente MPLAB X. Purtroppo, si riferiscono entrambi all'utilizzo di PIC32. Tuttavia, la conversione di un insieme base di tali librerie (es. funzionalità elementari per configurazione pin, lettura/scrittura di I/O digitali, lettura di segnali analogici) in modo che siano compatibili ALMENO con il dsPIC33FJ128MC802, visto nelle esercitazioni del corso, sarebbe un interessante approfondimento di vari contesti della programmazione embedded.

6. Programmazione di schede Arduino tramite MPLAB X

Strumenti utilizzabili:

- MPLAB X con compilatore XC8 o XC32 (a seconda dell'Arduino usato)
- Scheda Arduino reperita autonomamente dallo studente
- Esempi di procedura proposta per altri target (i.e. schede Arduino-like ma con processori Microchip PIC32): https://chipkit.net/programming-chipkit-boards-in-mplabx/
- Esempi di procedura per Arduino ma che richiedono hardware aggiuntivo: https://microchipdeveloper.com/mplabx:avr-debugwire

Da quando Microchip ha acquisito Atmel, quest'ultima produttrice dei microcontrollori AVR/ARM su cui sono basate le schede Arduino, è possibile programmare tali microcontrollori con MPLAB X, i relativi compilatori ed alcuni Programmer/Debugger prodotti dopo la fusione tra le due aziende (es. PicKit4). Tuttavia, com'è noto le schede Arduino sono pre-programmate con un bootloader che permette un più semplice ed economico (NO hardware aggiuntivo) trasferimento del codice da Arduino IDE.

Sviluppare e testare una procedura per programmare schede Arduino tramite il bootloader nativo, ma utilizzando MPLAB X come ambiente di sviluppo è quindi un interessante approfondimento su varie tecnologie per sistemi embedded.

7. Cenni generali su come impostare un proprio progetto con attinenza al corso

Proseguendo le considerazioni su schede Arduino, essendo nota la loro diffusione tra hobbisti e, presumibilmente, anche tra studenti di Ingegneria, si forniscono nel seguito alcune linee guida sull'uso di tali schede in modo che una qualunque relativa applicazione risulti compatibile con il corso in oggetto:

- **Configurazione tramite SFR**: uno degli obiettivi del corso è l'apprendimento delle metodologie di configurazione ed uso di periferiche integrate in microcontrollori/DSC, tramite la comprensione delle impostazioni associate ai relativi SFR. Qualunque sia l'applicazione sviluppata con una scheda Arduino, se questa è programmata configurando l'hardware direttamente tramite SFR anziché sfruttando le librerie di Arduino IDE, risulterà certamente istruttiva e professionalizzante tanto quanto gli esempi proposti nell'ambito delle lezioni di Tecnologie dei Sistemi di Controllo. A puro titolo di esempio, sono forniti nel seguito alcuni link a pagine che trattano l'argomento della programmazione *avanzata* di schede Arduino:
  https://www.arduino.cc/en/Reference/PortManipulation
  http://www.gammon.com.au/adc
  http://www.gammon.com.au/interrupts
  http://nicecircuits.com/playing-with-analog-to-digital-converter-on-arduino-due/
  http://www.atwillys.de/content/cc/using-custom-ide-and-system-library-on-arduino-due-sam3x8e/?lang=en
  http://forum.arduino.cc/index.php?topic=140205.0
- **Uso di strumenti di debug**: Arduino IDE non permette l'esecuzione di codice in modalità debug, sia per la filosofia stessa di trasferimento del codice sulla scheda target tramite bootloader che per la inerente semplicità (o anche, povertà funzionale..) dell'IDE stesso. Esistono però dei modi, sebbene perlopiù basati su

hardware aggiuntivo, per effettuare un debug approfondito di un'applicazione Arduino, operazioni che costituiscono anch'esse un importante competenza nel settore dei sistemi embedded. A titolo di esempio:
https://sites.google.com/site/wayneholder/debugwire
https://starter-kit.nettigo.eu/2015/debug-sketch-on-arduino-zero-pro-with-gdb-and-openocd/
https://devblogs.microsoft.com/iotdev/debug-your-arduino-code-with-visual-studio-code/
https://www.codeproject.com/Articles/5150391/Creating-and-Debugging-Arduino-Programs-in-Visual
https://www.codeproject.com/Articles/5160447/Creating-and-Debugging-Arduino-Programs-in-Visua-2
https://hackaday.io/project/162302-debugging-arduino-uno-in-vscode

Infine, oltre alle schede Arduino anche i Single-Board Computer come **Raspberry Pi** hanno conquistato uno spazio rilevante nel panorama dei sistemi embedded per l'hobbistica. **Raspberry Pi è però un target molto differente da quello di interesse principale per il corso di Tecnologie dei Sistemi di Controllo per i seguenti motivi**:

- Non integra nativamente periferiche specifiche per il controllo (es. ADC o PWM ad altra frequenza per controllo motori o convertitori di potenza).
- L'eventuale acquisizione di sensori o il comando di motori richiedono normalmente hardware aggiuntivo la cui interfaccia avviene tramite bus di comunicazione I2C/SPI, che rappresentano tecnologie significativamente diverse da quelle presentate nel corso citato.
- Essendo a tutti gli effetti dei Computer (Single-Board) con sistema operativo Linux-based, la programmazione è di fatto analoga a quella dei contesti non embedded, pertanto anch'essa significativamente diversa da quella presentata nel corso.

Tuttavia, un argomento interessante e potenzialmente attinente alle Tecnologie per Sistemi di Controllo è l'installazione su Raspberry di un sistema operativo con capacità Real-Time. A tale proposito, si forniscono nel seguito alcuni link a puro titolo di esempio:

https://www.socallinuxexpo.org/sites/default/files/presentations/Steven_Doran_SCALE_13x.pdf

https://www.stevebate.net/chibios-rpi/GettingStarted.html

https://easychair.org/publications/open/VPzR

https://www.get-edi.io/Real-Time-Linux-on-the-Raspberry-Pi/

https://lemariva.com/blog/2019/09/raspberry-pi-4b-preempt-rt-kernel-419y-performance-test