Acquisitori – modulo CAN Bus

Yanez Diego Parolin, Electronics Division

L'intero codice e le varue versioni software sono presenti sia nella sezione apposita sul DRIVE della MotorSport UNIBS che sulla mia pagina github

In caso di problemi futuri, file corrotti/danneggiati o chiarimenti, potete contattarmi via email a y.parolin@studenti.unibs.it specificando il file responsabile della discussione.

INTRODUZIONE ALL'ACQUISITORE

Come acquisitore si è scelto di utilizzare, in una prima versione, un Arduino Mega, successivamente rimpiazzato con un microcontrollore "custom" programmabile sempre attraverso Arduino IDE e che utilizza la CPU dell'Arduino Mega. Maggiori info a riguardo nella documentazione tecnica relativa a quest'ultimo.

COMUNICAZIONE E ACQUISIZIONE

Il nostro microcontrollore si interfaccia con una centralina, dalla quale e verso la quale, invia messaggi sfruttando il protocollo CAN Bus, nello specifico, nel nostro caso, sfruttiamo un unsigned char array di 8 elementi (dunque ogni elemento essendo un unsigned char può variare tra 0-255), suddiviso in quattro slot, costituiti rispettivamente da 2 elementi dell'array, questo, come vedremo, risulta necessario perchè la centralina richiede il dato in centesimi di Volt, scritti in HEX.

COME FUNZIONA IL CODICE

Infatti noi acquisiamo il segnale in un range 0-1024, successivamente lo convertiamo in un segnale 0-500 (perdendo ovviamente precisione) e quest'ultimo lo suddividiamo in una in due parti da 0-255, che la centralina sommerà tra di loro per ottenere il suo messaggio in 0-500,
La suddivisione segue una semplice logica, se il segnale risulta inferiore o uguale a 255, allora tale valore verrà scritto all'interno del primo slot libero del nostro array e il successivo slot viene posto a 0x00, viceversa se è maggiore di 255, nel primo slot viene scritto il corrispondente valore HEX di 255, cioè 0xFF, mentre nel secondo slot il valore HEX corrispondente alla differenza tra il nostro dato e 255.

In questo modo riusciamo a scrivere in ogni CAN Bus Message quattro valori provenienti da quattro sensori, evitando dunque di sovraccaricare la rete.
Ciò comporta chiaramente di porre attenzione allo slot dell'array su cui scrivere il dato, ciò può essere risolto con un puntatore count.

Questa operazione di conversione e trascrizione viene svolta dalla funzione "canBusValue".

Il nostro CAN Bus Message richiede anche un ID univoco, per permettere ovviamente alla centralina di sapere a che sensori corrisponde il dato messaggio. Visto che il CAN Bus Protocol in caso di conflitto da priorità al messaggio con ID più basso, la scelta dell'ID da assegnare al messaggio non può essere casuale.

L’invio e la stampa del messaggio (per il Debug) viene affidata alle funzioni printCanMSG e sendMessage.

In seguito a ogni invio, azzeriamo il nostro Can Bus Value sfruttando la funzione memset.

I sensori acquisiti possono essere molteplici, dalle sospensioni, all’accelerometro, passando per giroscopio.

CALIBRAZIONE E CONFLITTI

Questo richiede una particolare attenzione alla configurazione dello zero e alla calibrazione dei sensori.

Al fine di evitare di dover ogni volta modificare il codice per calibrare l’acquisizione del sensore, si è deciso di utilizzare un Arduino indipendente, il quale, interfacciandosi dall’esterno, permette di calibrare i sensori.

A questo arduino attualmente ci possiamo interfacciare sfruttando la porta seriale, con relativo software scritto sfruttando Processing da PC (vedi paragrafo PROGRAMMA PROCESSING E GUI) e in futuro sarà possibile controllarlo attraverso una semplice app Android, che tramite Bluetooth si interfaccerà sempre tramite la Seriale, sfruttando i moduli HC-05/06.

Da ora in avanti questo microcontrollore verrà denominato “Conf”, mentre i nostri acquisitori “Acq”.

il Conf comunica con gli Acq sfruttando la medesima rete CAN Bus.

Questa implementazione ha portato numerose implicazioni, dovuti alla modifica della nostra rete, l’aggiunta di un nodo trasmettitore e ricevitore di configurazione comporta la modifica del codice degli Acq. che dovranno anche ricevere messaggi e non solo inviare.

La principale problematica, che ha portato alla scrittura di sei versioni software, è il possibile conflitto fra i vari messaggi.

Mentre l’arduino invia un messaggio alla centralina, potrebbe ricevere un messaggio di configurazione dal Conf, creando uno stallo all’interno del nostro BUS, che darà priorità al messaggio con ID più basso, ma purtroppo non è detto che l’Acq al momento dell’invio, si trovi in modalità “ascolto” e il messaggio molto probabilmente verrà scartato.

Il nostro modulo CAN Bus prevede un Interrupt, settabile con maschere e filtri, ma funziona solamente a determinate velocità di invio, noi utilizzando 1Mbps, ci troviamo teoricamente fuori da questo range e rischiamo di far bloccare l’Acq.

La documentazione del modulo CAN Bus consiglia in questi casi di non usare l’interrupt e di inserire l’intero codice nel loop, ma ciò non ci garantisce la lettura del messaggio.

Ci sono molteplici soluzioni, io ne ho sviluppate sei, utilizzando anche l'interrupt.
L'una opzione che non ho sviluppato, ma che potrebbe essere interessante, sarebbe quella di introdurre un interrupt hardware manuale.

Elementi in comune tra le varie versioni:

- printCanMSG(unsigned long *id, unsigned char *canMsg)
- sendMessage(unsigned long *id, unsigned char canMsg[])
- canBusValue(unsigned int *sensorValue, unsigned char canMsg[], unsigned char *count)
- readCanMsg()
- attesa di 100 ms prima di inviare un nuovo messaggio (modificabile cambiando il valore di period)
- salvataggio nella EEPROM dello "zero" della calibrazione in un array

Versione ACQ_SOLOLOOP:
E' la più semplice, funziona solamente con un loop (oltre alle funzioni citate e ovviamente il setup), per evitare conflitti, prima di inviare il messaggio controlla se ci sono messaggio in arrivo, non garantisce assenza di conflitti.

Versione ACQ_SOLOINT:
Presenta l'aggiunta di un interrupt che interrompe l'esecuzione del loop quando arrivano i messaggi (post mask e filtro) provenienti dal Conf.
Anche questo non garantisce l'assenza di conflitti.

Versione ACQ_READTXRX:
In questa variante, prima di inviare un nuovo messaggio, l'Acq controlla lo stato del bus TX e RX per essere sicuro che non ci siano messaggi in coda o in procinto di arrivare.
Ciò permette di ridurre drasticamente la possibilità di conflitti, ma non di escluderla completamente.

Versione ACQ_LOOPLOCK:
Segue la logica dei semafori lock, gli Acq possono inviare solamente dopo aver ricevuto un token proveniente dall'altro/i Acq, che gli dà il via libera, una volto terminato l'invio dei messaggi, invia tramite CAN bus il token per il successivo Acq e si mette in standby.

Versione ACQ_CONFINT e ACQ_CONFLOOP:
Queste due revisioni, probabilmente risultano le più congeniali nonché quelle che garantiscono la minor probabilità di conflitti.
Gli Acq lavorano indipendentemente, finchè non ricevano un determinato messaggio CAN (che avrà un ID basso, così da poter essere ricevuto quasi sicuramente) proveniente dal Conf che manderà in una modalità "configurazione" gli Acq, che da questo momento non invieranno più dati verso la centralina, ma solamente al Conf in caso di richiesta dati.
Successivamente usciranno da questa modalità in seguito a un nuovo messaggio di configurazione del Conf.

Le rispettive versioni Arduino_ConfigvX risultano essere i programmi da caricare sul Conf in base alla scelta della versione ACQ_XXX

PROGRAMMA PROCESSING E GUI

Inoltre è stato sviluppato un piccolo programma sfruttando Processing, che si occupa di interagire con il Conf. tramite seriale.
Ha una semplice GUI grafica, per la selezione dei sensori sui relativi Acq, richiede l'attuale valore "segnalato" dal rispettivo sensore e setta "lo zero", con un meccanismo di richiesta-invio messaggi su Seriale, che a sua volta determina dei messaggi CAN Bus tra Conf e Acq

DEBUG

Infine è presente un programma "letturacan" che serve per Debuggare i messaggi sul CAN Bus.

NOTA: Il programma scritto con Processing (GUI) e le varie versione software risultano attualmente UNTESTED. L'unica versione testate e FUNZIONANTE è quella relativa al semplice invio dagli Acq alla centralina dei messaggi CAN, questo programma è attualmente caricato sugli Acq.