ISSN 2039-7941
Anno 2015_Numero 293

rapporti tecnici INGV

Sistema di acquisizione dati a basso consumo basato su Linux per sensori multiparametrici: il software

293

Istituto Nazionale di Geofisica e Vulcanologia

REGISTRAZIONE AL TRIBUNALE DI ROMA N.173 | 2014, 23 LUGLIO

Sede legale: Via di Vigna Murata, 605 | Roma

ISSN 2039-7941
Anno 2015_Numero 293

rapporti tecnici

INGV

SISTEMA DI ACQUISIZIONE DATI A BASSO CONSUMO BASATO SU LINUX PER SENSORI MULTIPARAMETRICI: IL SOFTWARE

Sergio Guardato

INGV (Istituto Nazionale di Geofisica e Vulcanologia, Sezione di Napoli - Osservatorio Vesuviano)

293

Indice

Introduzione

Questo rapporto tecnico nasce con lo scopo di completare il lavoro presentato nel Numero 226 di questa stessa collana editoriale [Guardato, 2012] in cui è ampiamente descritto l'hardware di un sistema di acquisizione dati a basso consumo, basato sul sistema operativo *open-source* Linux, per l'acquisizione di segnali digitali seriali provenienti da sensori multi-parametrici. Il documento presente descrive dettagliatamente il software di gestione e funzionamento del relativo sistema con un taglio di tipo nettamente manualistico.

Il software, che gira su una CPU del tipo ARM-9 a basso consumo [AT91SAM9G20] montata su una scheda elettronica commerciale, è suddiviso in due versioni maggiori: una di essa gestisce parte della strumentazione fuori acqua installata su una meda elastica nel Golfo di Pozzuoli [Iannaccone, et al, 2009] e denominata SURFACE, mentre l'altra gestisce al completo un modulo sottomarino per misure multi-parametriche, denominata *MODULE*. Su entrambe le versioni il software consente la gestione di: otto linee digitali di I/O, due interfacce seriali USB con accensione a comando, tre porte seriali asincrone di tipo RS-232, una porta seriale asincrona *half-duplex* di tipo RS-485/RS-422, un convertitore analogico-digitale a 12-bit ad otto canali *multiplexati* (utilizzati, tra l'altro, per monitorare i consumi elettrici della strumentazione e del sistema stesso), un sensore per la misura della temperatura del sistema ed una bussola magnetica digitale per la misura dell'inclinazione e dell'orientazione della meda elastica (per la versione *SURFACE*) e del modulo sottomarino (per la versione *MODULE*). L'architettura del software si completa, per il modulo sottomarino, con la gestione dell'evento relativo all'eventuale allagamento dell'*housing* del modulo; mentre per la parte fuori acqua, con la gestione del ricevitore GPS installato sulla parte emersa della meda elastica.

Infine, una serie di *script* scritti in *Bash*, coadiuvati da una serie di applicazioni elaborate in linguaggio *C*, completano il software di gestione del microprocessore, delle periferiche e dei sensori integrati ed esterni al sistema di acquisizione.

Breve descrizione del sistema

Il sistema di acquisizione dati oggetto di questo documento è costituito da tre schede elettroniche impilabili una sull'altra mediante degli *slot* (cfr. Figura 1.1, op. cit.]). Partendo dal basso, esso è costituito da una scheda elettronica commerciale, denominata Fox Board G20, con microprocessore integrato del tipo ARM-9 a basso consumo. Innestando su questa lo *shield* denominato SERIALS, si estendono due porte seriali USB 2.0 comandabili, con un'interfaccia seriale di tipo I^2C, una di tipo SPI e tre porte seriali asincrone di tipo RS-232 con una *half-duplex* di tipo RS-485/RS-422. Innestando su questa coppia di schede un'ulteriore *shield* denominato PIGTAIL, il sistema si completa con un ADC a 12-bit, ad otto canali *multiplexati*, di cui due utilizzati per il monitoraggio di altrettanti segnali analogici in tensione provenienti da sensori esterni (la scheda offre anche la possibilità della misura dei consumi elettrici dei sensori collegati al sistema, nonché la misura della temperatura, dell'inclinazione e dell'orientazione del sistema ospitante, con la presenza di un sensore di anti-allagamento, utile per applicazioni di geofisica marina). Infine, un connettore di espansione consente di collegare al sistema un'ulteriore scheda di estensione opto-isolata, separata dal sistema, con la possibilità di rendere disponibile all'esterno sino a otto linee digitali di I/O e un'interfaccia seriale di tipo I^2C.

Il sistema siffatto si presta adeguatamente per misure di monitoraggio ambientale effettuate mediante sensori multi-parametrici, con uscita analogica o digitale, e con *rate* di acquisizione medio-basso (da 0.1 a 10 sps).

Come si può vedere dalla Figura 1, il nucleo centrale del sistema è formato da una CPU del tipo ARM-9 a basso consumo disponibile sulla scheda Fox Board G20 alla quale sono collegate gli *shield* SERAILS e PIGTAIL.

Figura 1. Schema a blocchi del sistema.

Le librerie

Prima di passare alla descrizione dettagliata dei moduli di programma, e dei relativi files di libreria non standard (residenti nella directory */usr/local/include/*), si riporta l'elenco completo dei componenti costituenti l'architettura generale del software relativo alla gestione ed al funzionamento del sistema di acquisizione dati. Esso è costituito dai files di librerie e di programma con i loro nomi, i percorsi su memoria di massa (micro-SD card) e le eventuali dipendenze software, corredati con la loro descrizione dettagliata.

L'intero codice sorgente è scritto in linguaggio ANSI C e compilato sulla stessa CPU sotto sistema operativo Linux (distribuzione Debian Squeezy con *kernel* 2.6.38) con il compilatore *gnu* gcc versione 4.3.2.

La Tabella 1 che segue riporta l'elenco completo dei files di librerie (non standard) e di programma.

Files di librerie non standard	
FILE	LIBRERIA
utils.c	
foxg20_serial.h	fox
foxg20_serial.c	fox
iopins.h	fox
foxg20_gpio.h	fox
foxg20_gpio.c	fox
rs485.c	485
gps.h	gps
gps.c	gps
i2c.h	i2c
i2c.c	i2c
ads7828.h	i2c
ads7828.c	i2c
stt75.h	i2c
stt75.c	i2c

Files di programma	
hm55b.h	
heading.c	
adc.h	
temp.h	
tilt.h	
tilt.c	
status.h	
datastruct.h	
massmem.h	
massmem.c	
module.h	*MODULE*
module.c (*main*)	
surface.h	*SURFACE*
surface.c (*main*)	

Tabella 1. Elenco dei files delle librerie non standard e di programma.

La Figura 2 riporta la struttura grafica delle dipendenze software tra i vari moduli delle librerie non standard e di programma per le versioni *SURFACE* e *MODULE*, rispettivamente.

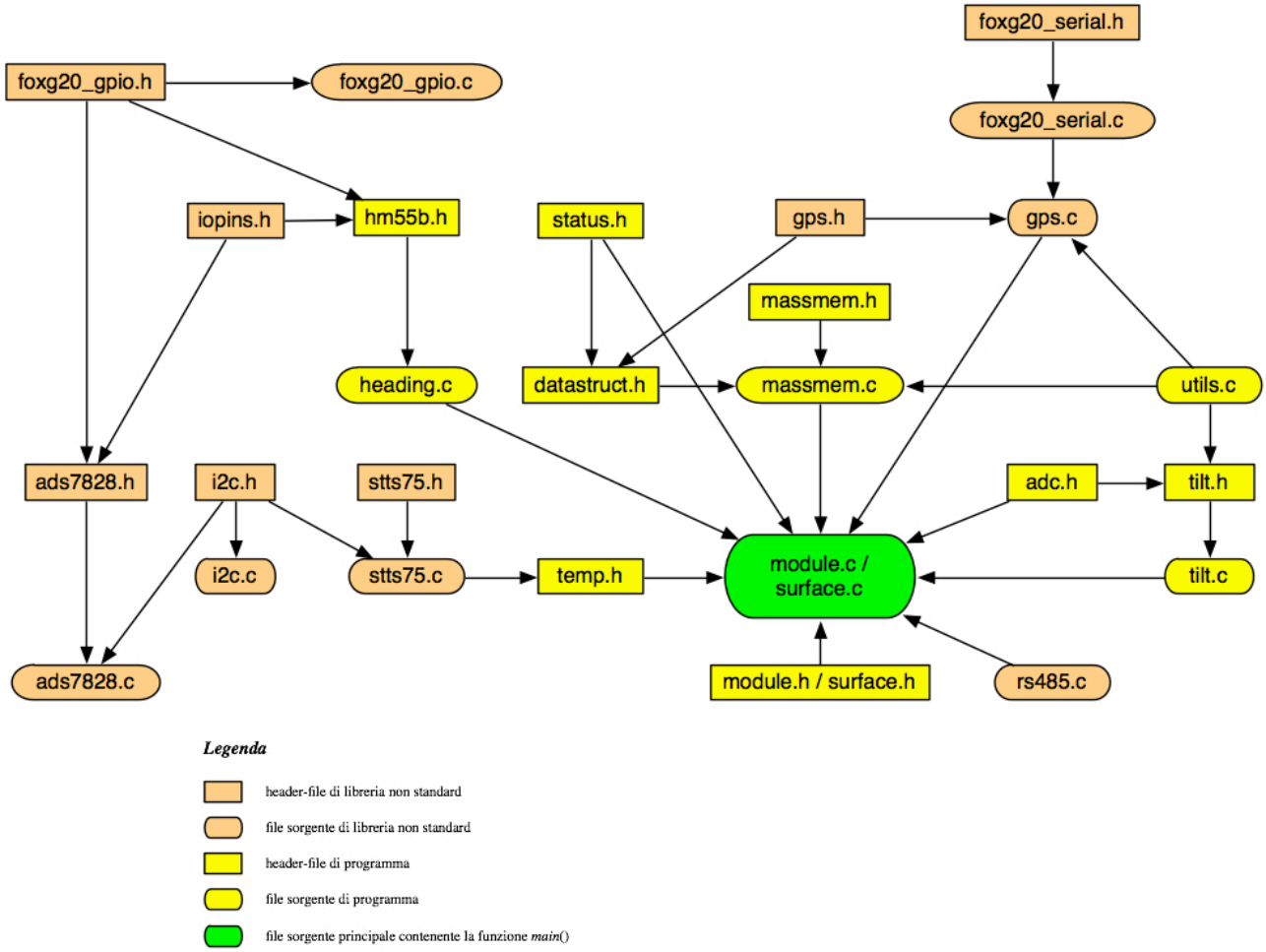


Figura 2. Dipendenze software dei moduli di libreria non standard e dei files di programma (versioni *MODULE* e *SURFACE*) così come indicati nella precedente Tabella 1. Si noti la differenza delle forme utilizzate per distinguere gli *header-file* dai sorgenti, sia per quelli della libreria che per quelli di programma.

Descrizione dei files di libreria

In questo paragrafo sono riportati e descritti tutti i files di libreria, comuni alle due versioni del software per la gestione ed il funzionamento del sistema di acquisizione dati.

File	**utils.c**
Percorso	/usr/local/include/
Dipendenze	*nessun file delle librerie non standard*

È un modulo contenente la definizione della funzione denominata *printInfo()* che stampa a video informazioni relative al programma in esecuzione, del tipo: *timestamp* dell'ultima compilazione del programma, scelta relativa ai messaggi a video da mostrare e opzioni di esecuzione con messaggi di debug a video.

File	**foxg20_serial.h**
Percorso	/usr/local/include/fox/
Dipendenze	*nessun file delle librerie non standard*

È un file di *header* contiene le macro relative alla definizione dei dispositivi di sistema (*/dev/ttySx*), utilizzabili come porte seriali, con i relativi *baud-rate*.

File	**foxg20_serial.c**
Percorso	/usr/local/include/fox/
Dipendenze	/usr/local/include/fox/foxg20_serial.h

È un modulo che contiene la definizione di cinque funzioni che interagiscono con i dispositivi seriali di sistema disponibili con il microprocessore *Netus*G20. La funzione *ttyReadByte()* serve per leggere un carattere da una delle porte seriali disponibili. Essa restituisce un intero e utilizza due argomenti: il primo è il numero intero relativo al descrittore di file del dispositivo (*/dev/ttySx*) utilizzato come porta seriale dalla quale leggere i dati, mentre l'altro è il puntatore al carattere letto dalla porta seriale. Se tutto è andato a buon fine, la funzione restituisce 0, altrimenti 1.

La funzione *ttyReadNByte()* serve per leggere una stringa da una delle porte seriali disponibili utilizzando ciclicamente la funzione appena descritta per quanti sono i caratteri da leggere. Anch'essa restituisce un intero ed ha bisogno di tre parametri: il numero intero relativo al descrittore di file del dispositivo, il puntatore alla stringa letta dalla porta seriale, ed il numero di caratteri da leggere. Essa ritorna il numero di caratteri letti dalla porta seriale considerata.

La funzione denominata *ttyDiscardByteUpToC()* ritorna un numero intero e serve a cercare un particolare carattere, passato come argomento della funzione assieme al descrittore, nel flusso di caratteri provenienti dalla porta seriale. Se lo trova, essa ritorna il codice ASCII corrispondente a quel carattere, altrimenti -1.

La funzione *ttyReadByteUpToCTimeOut()* effettua la stessa operazione della funzione di ricerca carattere appena descritta ma aggiunge un time-out, un terzo parametro espresso in millisecondi, che deve essere passato come numero intero. Vale a dire che essa se non trova il carattere di ricerca entro un certo tem-

po restituisce il codice di errore -1, altrimenti essa ritorna il codice ASCII corrispondente a quel carattere.

La funzione *lookForCoupleBB()* serve per identificare una coppia di due caratteri identici e consecutivi presenti nel flusso di caratteri del dispositivo seriale considerato. Essa è particolarmente utile per la discriminazione dei dati (*sentences*) in formato NMEA 0183 forniti dal ricevitore GPS. La funzione restituisce un intero e lavora con due argomenti: l'intero corrispondente al descrittore di file ed il carattere letto dalla porta seriale. Ritorna 1 se viene trovata la coppia, altrimenti 0.

Infine, viene definita la funzione *ttyRxFlush()* la quale effettua lo svuotamento (*flush*) necessario per la pulizia del buffer di ricezione della porta seriale individuato dal corrispondente descrittore di file passato come unico argomento della funzione. Essa ritorna l'intero associato alla chiamata di sistema *tcflush()*.

File	**iopins.h**
Percorso	/usr/local/include/fox/
Dipendenze	*nessun file delle librerie non standard*

È un file di *header* contiene tutte le macro relative alla definizione dei pin di ingresso/uscita del microprocessore *Netus*G20 utilizzabili con l'interfaccia standard di sistema *sysfs* per effettuare il *bit-banging* su di esse.

Più precisamente, esse assegnano l'identificativo del *kernel* relativo alla linea di GPIO (*General Purpose Input Output*) ad un simbolo. Per esempio, con la macro seguente:

```
#define        PB2    "66"
```

all'identificativo (id) numero 66 del *kernel*, corrispondente al pin 7 del connettore J7 del processore *Netus*G20 (J7.7), è stato assegnato il simbolo PB2. Si tratta di una linea digitale a 3.3V, utilizzata come uscita, e che serve come segnale di clock per il *bus* seriale SPI per la gestione della bussola magnetica digitale installata a bordo del sistema di acquisizione dati.

Le macro contenute in questo file assegnano i nomi ai pins di I/O seguendo la nomenclatura descritta in Figura 1.2.4 [op. cit.]. In essa sono raffigurate le nomenclature relative ai due connettori J6 e J7, costituiti di 40 pin ognuno. Le 96 linee di GPIO disponibili sono raggruppate in tre gruppi: PAx, PBx e PCx di 32 identificativi ciascuno.

File	**foxg20_gpio.h**
Percorso	/usr/local/include/fox/
Dipendenze	*nessun file delle librerie non standard*

È un file di *header* contiene le macro relative alla definizione dei simboli che indicano la direzione (DIR_IN=1, ingresso; DIR_OUT=0, uscita) ed il valore logico (LINE_LOW=0, basso; LINE_HIGH=1, alto) da assegnare ad una delle linee digitali del microprocessore utilizzabili con l'interfaccia standard di sistema *sysfs* per effettuare il *bit-banging* su di esse.

File	**foxg20_gpio.c**
Percorso	/usr/local/include/fox/
Dipendenze	/usr/local/include/fox/foxg20_gpio.h

È un modulo contenente la definizione della struttura relativa ad una linea di GPIO, e quattro funzioni correlate alla gestione delle linee di I/O digitali del microprocessore ARM. Tutte le funzioni restituiscono un intero dichiarato di tipo *extern* poiché le stesse funzioni sono utilizzate da molti moduli software del programma.

La struttura denominata *sGpioLine* è composta da una stringa di quattro caratteri, contenenti i tre caratteri corrispondenti all'identificativo del *kernel* (definiti nel file di *header* 'iopins.h') seguiti dal carattere *NULL* finale, un intero rappresentativo della direzione della linea, ed infine un intero rappresentativo del valore logico della linea (0=basso, 1=alto).

La funzione *set_gpio_direction()* imposta la direzione (ingresso o uscita) di una linea digitale, mediante l'identificativo del *kernel*, utilizzando l'interfaccia standard di sistema *sysfs*. Essa prende come argomenti un puntatore ad una struttura di tipo *sGpioLine* ed un numero intero che rappresenta la direzione della linea (Input=1, Output=0). Se l'operazione va a buon fine, viene restituito 0, altrimenti -1.

La funzione *load_gpio_direction()* esporta nel *sysfs* una linea digitale. Essa prende come argomenti un puntatore ad una struttura di tipo *sGpioLine*, un numero intero che rappresenta la direzione, ed infine la stringa di quattro caratteri relativa all'identificativo del *kernel*. Se l'operazione di scrittura sul *sysfs* va a buon fine, viene restituito 0, altrimenti -1.

La funzione *set_gpio_line()* imposta il valore (alto o basso) di una linea digitale. Essa prende come argomenti un puntatore ad una struttura di tipo *sGpioLine* ed un numero intero che rappresenta il valore della linea stessa. Se l'operazione di scrittura sul *sysfs* va a buon fine, viene restituito zero, altrimenti: -1 se la linea è impostata come ingresso (di sola lettura), e -2 se non è possibile scrivere il valore della linea utilizzando l'interfaccia standard *sysfs*.

La funzione *read_gpio_line()* legge il valore di una linea digitale utilizzando come argomento un puntatore ad una struttura di tipo *sGpioLine*. Se l'operazione di lettura dal *sysfs* va a buon fine, viene restituito zero, altrimenti: -1 se la linea digitale è stata impostata come uscita, -2 se non è stato possibile accedere alla linea tramite interfaccia *sysfs*, oppure -3 se non è stato possibile leggere il valore della linea digitale.

File	**rs485.c**
Percorso	/usr/local/include/485/
Dipendenze	*nessun file delle librerie non standard*

È un modulo contenente due funzioni che impostano ed aprono una porta seriale in modalità compatibile con lo standard RS-485. Quest'ultima, compatibilmente con l'hardware del sistema, è abilitata (per esigenze tecniche) alla sola ricezione dei dati provenienti da un ricevitore GPS esterno dedicato, e con uscita dati in formato NMEA.

La funzione denominata *setRs485()* richiede come argomento il passaggio di un descrittore di file del dispositivo seriale (numero intero assegnato dal sistema operativo) e restituisce un intero. Essa imposta gli attributi della porta (lettura, scrittura o entrambe, *baud-rate*, numero di bits, parità...) e ritorna zero se va a buon fine, -1 altrimenti.

L'altra funzione, denominata *openRs485()*, richiede il passaggio di un puntatore a carattere come argomento (il dispositivo di sistema, vale a dire */dev/ttySx*), il cui simbolo è definito mediante la macro PORT_GPS presente nel file principale 'module.c'. Essa apre il dispositivo con la chiamata di sistema *open* e, se il numero intero da quest'ultima restituito è positivo, imposta la porta seriale con la funzione precedentemente descritta restituendo il descrittore di file assegnato dal sistema operativo.

File	**gps.h**
Percorso	/usr/local/include/gps/
Dipendenze	*nessun file delle librerie non standard*

È un file *header* che definisce delle costanti per la gestione di alcuni pacchetti dati prelevati dalla porta seriale del ricevitore GPS in formato NMEA. Esso definisce anche le specifiche di formato per la stampa a video, nella sola modalità di debug, dei dati relativi al *time-stamping* ed alle informazioni sui satelliti 'in vista' dall'antenna del ricevitore GPS esterno.

In esso sono definite due strutture dati: una denominata *typeTime*, che contiene le informazioni sulla data ed il tempo in campi separati (anno, mese, …); un'altra denominata *typeGps*, contenente le informazioni sui satelliti, tipo: la latitudine, longitudine ed altri.

File	**gps.c**
Percorso	/usr/local/include/gps/
Dipendenze	/usr/local/include/fox/foxg20_serial.c /usr/local/include/utils.c /usr/local/include/gps/gps.h

Questo modulo presenta la dipendenza da due moduli della libreria non standard e dall'omonimo file *header*. In esso sono presenti quattro funzioni relative alla gestione della data e dell'ora, ed altrettante dedicate alla lettura e decodifica del particolare pacchetto NMEA inviato dal ricevitore GPS.

Per ciò che concerne la gestione del tempo, la funzione *getCpuTime()*, senza argomenti, restituisce una struttura *typeTime* (definita nel file di *header* 'gps.h') leggendo la data ed il tempo dalla CPU del sistema utilizzando le funzioni di sistema *time* e *localtime*.

La funzione *getTimeFromTimeT()* richiede il passaggio del tipo di dato *time_t* e restituisce anch'essa una struttura *typeTime* contenente la data ed il tempo letti dalla CPU, come per la funzione precedentemente descritta.

La funzione denominata *setTime()* imposta la data e l'ora nella CPU contenute nella struttura di tipo *typeTime*, passata come argomento puntatore, restituendo il tipo *void*.

Infine, la funzione *gpsTimeSyncCpu()* legge, dalla porta seriale RS-485 alla quale è collegato il ricevitore GPS (utilizzando come argomento l'intero rappresentativo del descrittore di file in uso dal sistema operativo), data e orario - in campi separati - sincronizzando la CPU con quest'ultimi. Se l'operazione va a buon fine, viene restituito zero, altrimenti un intero negativo a seconda dell'errore che si verifica.

Per ciò che è inerente alla lettura di alcuni pacchetti NMEA dal ricevitore GPS, la funzione *gpsSyncInitData()* legge tutti i caratteri dalla porta seriale in oggetto (utilizzando come argomento l'intero rappresentativo del descrittore di file in uso dal sistema operativo) sino a quando non ritrova la coppia di caratteri relativa all'inizio della stringa ZDA, entro un time-out di 1.5 ms.

La funzione denominata *gpsComReadGGA()* restituisce un intero e utilizza come argomenti un numero intero, relativo al descrittore di file in uso dal sistema, ed il puntatore ad una struttura di tipo *typeGps*. In particolare, essa legge la stringa del pacchetto NMEA relativa alla *sentence* GGA del ricevitore GPS contenente informazioni relative ai satelliti, e ne assegna i valori alla struttura. Se l'operazione va a buon fine, viene restituito zero, altrimenti un intero negativo a seconda dell'errore che si verifica.

La funzione *gpsComReadZDA()* restituisce un intero e utilizza come argomenti un numero intero, relativo al descrittore di file in uso dal sistema, ed il puntatore ad una struttura di tipo *typeTime*. Essa legge la stringa del pacchetto NMEA relativa alla *sentence* ZDA del ricevitore GPS contenente informazioni relative al tempo ed alla data, assegnandone i valori alla struttura. Se l'operazione va a buon fine, viene restituito zero, altrimenti un intero negativo a seconda dell'errore che si verifica.

Infine, la funzione denominata *gpsComGetData()* restituisce il tipo *void* e utilizza come argomenti un

numero intero, relativo al descrittore di file in uso dal sistema, ed un puntatore ad una struttura di tipo *typeGps*. Questa funzione non fa altro che utilizzare le due funzioni di lettura dei dati NMEA appena descritte e impostare, in maniera opportuna, un *flag* della struttura *typeGps*, indicativo della corretta lettura dei dati dal ricevitore GPS.

File	**i2c.h**
Percorso	/usr/local/include/i2c/
Dipendenze	*nessun file delle librerie non standard*

È un file di header utile per la gestione del sensore di temperatura [STTS75] e del convertitore analogico-digitale a 12-bit [ADS7828] installati a bordo della scheda PIGTAIL del sistema di acquisizione dati. Entrambi i dispositivi fanno uso dell'interfaccia di comunicazione seriale basata sullo standard I2C.

Nel file sono dichiarate due strutture di tipo *sGpioLine* (definite nel modulo 'foxg20_gpio.c') per la gestione delle linee di bus, denominate SDA e SCL, esplicitamente definite nel file di header 'module.h'. Inoltre in esso vi sono le quattro macro relative alle dichiarazioni che consentono di eseguire le transizioni di livello digitale sulle linee SDA e SCL del bus eseguendo il *bit-banging* mediante l'ausilio dell'interfaccia *sysfs*.

File	**i2c.c**
Percorso	/usr/local/include/i2c/
Dipendenze	/usr/local/include/i2c/i2c.h

Questo modulo è il 'cuore' della porzione di codice contenente tutte le funzioni di gestione del *bus* I^2C. In esso vi si trovano tutte le funzioni che inizializzano, leggono e scrivono nei dispositivi interessati mediante l'ausilio di questa interfaccia seriale di dati.

La funzione *i2c_sda_dir()* serve per impostare il pin digitale scelto con la funzione di linea dati (SDA), come ingresso (per le letture su *bus*) o uscita (per le scritture su *bus*). Essa prende come unico argomento il numero intero che stabilisce se impostare la linea SDA come ingresso o uscita. La funzione ritorna l'intero zero se l'impostazione della linea SDA è andata a buon fine, -1 altrimenti.

La funzione *i2c_init()* pone il *bus* I^2C nella condizione di *bus not-busy*, semplicemente portando le linee SDA e SCL (quest'ultima fissata come uscita) a livello logico alto. Essa non presenta nessun argomento e il suo valore di ritorno è zero se non vi sono errori, -1 altrimenti.

La funzione *i2c_start()* genera la condizione di *Start* sul *bus* I^2C ad opera del dispositivo Master (il microcontrollore, in ogni caso). Essa imposta la linea SDA come linea di uscita e, dopo aver portato la linea di clock SCL alta, esegue una transizione della linea SDA dal livello logico alto a quello basso. La funzione non possiede nessun argomento e ritorna il numero intero zero se va a buon fine, -1 altrimenti.

La funzione *i2c_stop()* genera la condizione di *Stop* sul *bus* I^2C ad opera del dispositivo Master. La funzione imposta la linea SDA come linea di uscita ed infine porta a livello logico alto sia la linea di clock SCL che la linea SDA, avendo cura di cambiare lo stato di quest'ultima solo quando la linea SCL si trova al livello logico alto. La funzione non possiede nessun argomento e ritorna il numero intero zero se va a buon fine, -1 altrimenti.

La funzione denominata *i2c_write()* serve a scrivere un byte seriale sul *bus* I^2C. Essa si prende cura di impostare la linea SDA come uscita e poi invia sul *bus* seriale il byte da scrivere nel relativo Slave passato come argomento, di tipo carattere senza segno. A tal fine esegue delle transizioni dal livello logico basso a quello alto della linea SCL interponendo, a ogni transizione del clock, il valore del bit da scrivere sul *bus*. La funzione ritorna il numero intero zero se va a buon fine, -1 altrimenti.

La funzione *ack_read()* serve a leggere il bit di *acknowledgement* dal *bus* I^2C inviato dal dispositivo Slave. Essa imposta la linea SDA come ingresso, manda la linea di clock SCL a livello logico basso e legge il valore digitale presente sulla linea dati SDA. La funzione non ha argomenti e restituisce il numero intero zero se il bit di *acknowledge* viene ricevuto dal dispositivo Master, altrimenti -1.

Infine, la funzione *i2c_read()* legge un byte dal *bus* I^2C. Essa imposta la linea SDA come ingresso e, durante le transizioni dal livello logico alto a quello basso della linea di clock SCL, legge il singolo bit serialmente dal *bus* e lo memorizza in una variabile di tipo carattere senza segno, che corrisponde al valore restituito dalla funzione.

Questa funzione, dopo aver letto il byte di dato, stabilisce - in base all'unico argomento passato - se il microprocessore debba inviare o meno al dispositivo con cui sta dialogando, il bit di *acknowledgement*.

Per tutto quanto detto in questo modulo si faccia riferimento al Paragrafo 3.2 [Guardato, 2012].

File	**ads7828.h**
Percorso	/usr/local/include/i2c/
Dipendenze	/usr/local/include/fox/iopins.h
	/usr/local/include/fox/foxg20_gpio.h

Questo *header* contiene tutte le definizioni delle macro per la gestione del convertitore analogico-digitale (ADC) a 12-bits installato sulla scheda PIGTAIL, e che fa uso dell'interfaccia seriale di comunicazione basata sullo standard seriale I^2C [ADS7828].

In esso vi si trovano le macro che impostano l'indirizzo dell'ADC (quello hardware è impostato con i pins dedicati del dispositivo: 0x04), e il valore da scrivere nel registro apposito per eseguire un'operazione di lettura o scrittura su di esso. Poi figurano le macro relative all'indirizzamento ed all'impostazione del registro di comando. Vi è inoltre contenuta la macro che stabilisce di impostare l'ADC con il riferimento di tensione interno e in modalità *single-ended*. Infine, compaiono le macro che definiscono la funzione degli otto canali d'ingresso dell'ADC.

File	**ads7828.c**
Percorso	/usr/local/include/i2c/
Dipendenze	/usr/local/include/i2c/i2c.h
	/usr/local/include/i2c/ads7828.h

Questo modulo contiene la definizione della funzione dedicata alla lettura dei valori delle conversioni effettuate dall'ADC installato a bordo del sistema di acquisizione dati.

La funzione denominata *read_adc()* effettua la lettura seriale, sul *bus* I^2C, dei dati provenienti dall'ADC a 12-bits sul relativo canale (fra gli otto disponibili) passato come unico argomento mediante una variabile di tipo carattere senza segno. Essa restituisce il numero intero letto (da 0 a 4095) mediato sul numero di conversioni da effettuare, ed impostato per mezzo della macro denominata NUM_CONV definita nel file denominato 'module.h'.

La funzione in questione dapprima inizializza il *bus* I^2C mediante l'utilizzo della funzione *i2c_init()* definita nel modulo 'i2c.c', poi si ripete sul numero di conversioni da effettuare. Nel ciclo essa esegue le seguenti operazioni:

- generazione della condizione di Start sul *bus*;
- scrittura sul *bus* dell'indirizzo dello Slave (in questo caso l'ADC) per mezzo della macro ADS7828_WRITE definita nel file di *header* 'ads7828.h';
- attesa del bit di *acknowledgement* dal dispositivo Slave;
- se il bit di *acknowledgement* è stato ricevuto, legge in sequenza i due byte dall'ADC (si faccia riferimento alla Figura 3.5.6 [op. cit.]);
- generazione della condizione di Stop sul *bus* ;
- calcolo della media sul numero di conversioni effettuate (al limite, una soltanto).

In caso di errore, la funzione ritorna i codici interi mostrati nella tabella seguente:

Codice di errore	Significato
-1	Errore di scrittura dell'Address Byte (W) nell'ADC
-2	Bit di *acknowledgement* in scrittura dell'Address Byte non ricevuto dall'ADC
-3	Errore di scrittura del Command Byte (W) nell'ADC
-4	Bit di *acknowledgement* in scrittura del Command Byte non ricevuto dall'ADC
-5	Errore di scrittura dell'Address Byte (R) nell'ADC
-6	Bit di *acknowledgement* in lettura del MSB e del LSB non ricevuto dall'ADC *(dopo la condizione si Start)*

Tabella 2. Codici di errore per la funzione 'ads7828.c'.

File	**stts75.h**
Percorso	/usr/local/include/i2c/
Dipendenze	*nessun file delle librerie non standard*

Il file di *header* contiene le macro relative alla definizione dei registri del sensore di temperatura digitale [STTS75] installato a bordo della scheda PIGTAIL del sistema di acquisizione dati; questo viene utilizzato per il monitoraggio e la misura della temperatura interna al cilindro dell'elettronica del modulo sottomarino. Anche questo sensore fa uso dell'interfaccia seriale di comunicazione rispondente allo standard I^2C. Nel file vi si trovano anche le macro relative alla risoluzione con cui debbono essere letti i valori di temperatura dal dispositivo (da 9 a 12-bit).

File	**stts75.c**
Percorso	/usr/local/include/i2c/
Dipendenze	/usr/local/include/i2c/stts75.h
	/usr/local/include/i2c/i2c.h

Il modulo software contiene quattro funzioni che consentono di effettuare la lettura della temperatura rilevata dal sensore digitale STTS75, e la gestione dei suoi registri interni di configurazione.

La funzione *read_conf()* legge il contenuto del registro di configurazione (CONF) del dispositivo. Essa fa uso delle funzioni definite nel modulo 'i2c.c' e ritorna il byte letto se non vi sono errori, altrimenti -1. La funzione non ha nessun argomento.

La funzione *write_conf()*, invece, scrive il byte di configurazione nel registro CONF, necessario per impostare la risoluzione delle misure di temperatura lette dal sensore. Quest'operazione avviene per mezzo dell'unico argomento che gli viene passato: un tipo carattere senza segno. La funzione compie le operazioni rispettando il protocollo visibile in Figura 3.3.5 [op. cit.]. Essa ritorna zero se va a buon fine, altrimenti un numero intero negativo.

La funzione *decode_sign()* viene utilizzata per decodificare il segno del valore di temperatura letto dal sensore e contenuto nel bit più significativo del byte MSB (bit 15, SB) rilevato dalla lettura del registro di temperatura TEMP (si faccia riferimento alla Tabella 3.3.7 [*ibid.*]). L'argomento passato, di tipo carattere senza segno, rappresenta il byte più significativo sul quale operare. Il numero intero restituito rappresenta il bit di segno.

Infine, la funzione *read_temp()* è quella dedicata esclusivamente alla decodifica della lettura del valore di temperatura rilevata dal sensore digitale. Essa richiede come argomento un numero intero che indica la risoluzione, in termini di numero di bits, con cui il dispositivo dovrà effettuare le conversioni (da 9 a 12-bit). Nel richiamare questa funzione verrà passata una delle costanti presenti nel file di *header* 'stts75.h' e definite attraverso delle macro. La funzione ritorna la temperatura interna del cilindro metallico - espressa sotto forma di numero in doppia precisione - nel quale è installata tutta l'elettronica, compreso il sistema di acquisizione dati costituente il modulo sottomarino. In caso di errore viene restituito il numero -999.99.

Descrizione dei files di programma

Il seguente paragrafo contiene la descrizione dettagliata di tutti i files di programma con i loro nomi, i percorsi e le dipendenze dai files di libreria non standard descritti nel paragrafo precedente. Si noti che il percorso su memoria di massa cambia a seconda che si tratti della versione *MODULE* o *SURFACE*, ma il contenuto del file è perfettamente identico.

File	**hm55b.h**
Percorso	/root/programs/module[surface]/
Dipendenze	/usr/local/include/fox/iopins.h
	/usr/local/include/fox/foxg20_gpio.h

Questo file di *header* contiene tutte le macro e le funzioni relative alla gestione del sensore costituente la bussola magnetica digitale [HM55B]. Le macro definiscono i pin del microprocessore utilizzati come I/O per i dati, per il clock e l'abilitazione del sensore rispettando lo schema visibile nella tabella 3.4.3 [op. cit.].

La funzione *spi_init()* imposta le linee anzidette ed inizializza il dispositivo mettendolo in *sleep*. Essa non ha nessun argomento e non ritorna alcun valore.

La funzione *comp_reset()* abilita il dispositivo, poi lo resetta, ed infine lo disabilita. Essa non ha nessun argomento e non ritorna alcun valore.

La funzione *check_status()*, senza argomento, restituisce l'intero relativo alla lettura dello stato di funzionamento del sensore secondo quanto indicato nella tabella 3.4.2 [*ibid.*].

La funzione *comp_start()* abilita il sensore ad effettuare le letture delle componenti della forza del campo magnetico terrestre parallele ai due assi di sensibilità del dispositivo, e poi introduce un'attesa di 40 ms, necessari a quest'ultimo per completare la fase di preparazione alle letture. La funzione non ha nessun argomento e non ritorna alcun valore.

La funzione *read_sign()* legge il primo dei 22-bit forniti in risposta dal sensore e lo memorizza nel numero intero che essa restituisce. Non ha nessun argomento.

La funzione *read_angle()* effettua tutto il necessario per leggere in successione i restanti bit in uscita dal sensore e rappresentativi della lettura dell'angolo formato dal vettore forza del campo magnetico terrestre ed uno dei assi di sensibilità del dispositivo. L'angolo letto, opportunamente manipolato, viene fornito come un numero intero ritornato dalla funzione. Quest'ultima utilizza come argomento l'intero rappresentativo del segno della misura restituito dalla funzione *read_sign()*.

File	**heading.c**
Percorso	/root/programs/module[surface]/
Dipendenze	/root/programs/module/hm55b.h

Questo modulo contiene la funzione che restituisce l'angolo di lettura dell'*heading* dal sensore di bussola magnetica digitale.

La funzione, denominata *readHeading()*, fa uso di tutte le funzioni definite nel rispettivo file di *header* 'hm55b.h' ed effettua tutti i calcoli necessari per fornire il numero rappresentativo dell'angolo di direzione del modulo sottomarino rispetto al Nord magnetico terrestre, espresso in doppia precisione. Essa non ha nessun argomento e, in caso di errore, restituisce il numero 999.99.

File	**adc.h**
Percorso	/root/programs/module[surface]/
Dipendenze	/usr/local/include/i2c/ads7828.c

Questo file di *header* contiene le macro relative alla definizione dei simboli degli otto canali *single-ended* dell'ADC disponibili ai quali sono connessi: i segnali di *tilt* (*pitch* e *roll*) provenienti dall'uscita del sensore accelerometrico biassiale di tipo MEMS [ADXL203], una coppia di tensioni-correnti di monitoraggio ed una coppia di tensioni ausiliarie.

Qui sono anche ridefiniti i simboli dei pin usati come linea dati (SDA) e di clock (SCL) per l'interfaccia I^2C da utilizzare per la gestione dell'ADC. Le stesse definizioni sono ripetute nell'*header* 'temp.h', questo perché si potrebbe pensare di utilizzare soltanto l'ADC, oppure solo il sensore di temperatura.

File	**temp.h**
Percorso	/root/programs/module[surface]/
Dipendenze	/usr/local/include/i2c/stts75.c

L'*header* contiene quattro macro necessarie alla gestione del sensore di temperatura digitale STTS75 mediante interfaccia seriale I^2C. Due di esse definiscono le connessioni fisiche realizzate sui pin di indirizzo hardware A0-A2 del sensore. Quest'ultimo è montato sulla scheda PIGTAIL e presenta tutti e tre i pin di indirizzo collegati alla tensione di alimentazione V_{cc} di +3.3V. Avendo operato in tal modo, l'indirizzo software del sensore sul *bus* I^2C è, pertanto, 1001.1111_b, per operazioni di lettura, e 1001.1110_b per operazioni di scrittura nei suoi registri (si faccia riferimento alla Tabella 3.3.9 [op. cit.]).

Sono ridefiniti i simboli dei due pin usati come linea dati (SDA) e di clock (SCL) per l'interfaccia I^2C da utilizzare per la gestione del sensore di temperatura (la ridefinizione delle linee compare anche nel file *header* denominato 'adc.h').

File	**tilt.h**
Percorso	/root/programs/module[surface]/
Dipendenze	/root/programs/module[surface]/adc.h

Nell'*header* è contenuta la macro, denominata CONVS, con la quale si definisce il numero di letture da effettuare dai due canali dell'ADC a cui sono connessi i segnali analogici in uscita dall'accelerometro MEMS utilizzato per la misura di *tilt*. Si è potuto osservare che eseguendo quattro letture di queste due grandezze si impiega un tempo pari a circa 250 ms.

Le altre due macro definiscono la costante pi-greca (π) ed il fattore di conversione da radianti a gradi.

File	**tilt.c**
Percorso	/root/programs/module[surface]/
Dipendenze	/root/programs/module[surface]/tilt.h

Questo modulo contiene il codice relativo alla dichiarazione della sola funzione *read_tilt()* la quale legge dall'ADC un certo numero di conversioni (definite nell'*header* 'tilt.h'). Essa ritorna il numero in doppia precisione rappresentante l'angolo medio di *tilt*, in termini sia di *pitch* che di *roll*, in letture successive (La lettura può riguardare sia il modo in cui il modulo sottomarino è adagiato sul fondale, versione software *MODULE*, sia come si muove nello spazio la parte emersa della meda elastica, versione *SURFACE*). La funzione fa uso di un unico argomento, espresso come numero intero, e che serve a discriminare tra le letture di *pitch* e quelle di *roll*. Essa utilizza le chiamate alla funzione *read_adc()*, definita nel modulo 'ads7828.c', manipolando opportunamente le letture. In caso di errore, la funzione ritorna il numero -100.00.

File	**status.h**
Percorso	/root/programs/module[surface]/
Dipendenze	*nessun file delle librerie non standard*

Questo file di *header* contiene la definizione di una struttura e le macro relative al formato di stampa a video, in fase di *debug*, delle informazioni relative allo stato di funzionamento dei sensori di stato (sia per quelli installati nell'*housing* del modulo sottomarino, sia per quelli installati in un opportuno box da strumentazione sulla parte emersa della meda elastica).

Le macro consentono di dare un'informazione immediata del sistema visualizzando i valori di:

- temperatura interna del modulo (o del box strumentazione);
- *heading* magnetico del modulo (o della parte emersa della meda elastica);
- *pitch* e il *roll* (tilt lungo la coordinata X e Y del modulo sul fondale marino, o della parte emersa della meda elastica);

- tensione di alimentazione del modulo (o di tutto il sistema modulo/meda);
- corrente totale assorbita dal modulo (o da tutto il sistema modulo/meda);
- tensione e corrente ausiliarie;
- due valori di tensione analogica;
- *flag* di intrusione acqua nel modulo (o di apertura dello sportello del box strumentazione);

tutti corredati anche dell'informazione relativa al *timestamp*.

La struttura denominata *typeStatus* contiene una variabile di tipo carattere rappresentante il *flag* di stato del sistema (un numero intero che normalmente vale zero e stabilisce se il cilindro metallico, in cui è montata tutta l'elettronica del modulo sottomarino, è minacciato dall'intrusione dell'acqua di mare; oppure, se lo sportello del box strumentazione, installato sulla parte emersa della meda elastica, è stato aperto). Se, per la versione *MODULE*, esso vale '1', allora questo è indice che l'*housing* del sistema sottomarino si sta allagando; a questo punto, dopo un opportuno *polling* che dura qualche secondo, si attiva un allarme software che informa l'elettronica intelligente presente in superficie, sulla parte emersa della meda elastica, di spegnere immediatamente il modulo sottomarino ed invia, a chi di dovere, e-mail di allerta. Nella struttura sono presenti le dichiarazioni dei tipi, rappresentati in doppia precisione, relativi a tutte le variabili di stato sopra elencate.

File	**datastruct.h**
Percorso	/root/programs/module[surface]/
Dipendenze	/usr/local/include/gps/gps.h
	/root/programs/module[surface]/status.h

Il file *header* in questione contiene solo la definizione della struttura rappresentativa del pacchetto completo dei dati della sensoristica del modulo sottomarino (o della parte emersa della meda elastica) connessa alle schede PIGTAIL e SERIALS.

La struttura denominata *typeSPack* contiene, essenzialmente, le strutture *typeStatus* e *typeTime*, prima definite nei file di *header* 'status.h' e 'gps.h', rispettivamente.

File	**massmem.h**
Percorso	/root/programs/module[surface]/
Dipendenze	*nessun file delle librerie non standard*

Contiene le macro relative: alla definizione del percorso con la cartella in cui salvare localmente i file dei dati (/root/DATA/) prima di inviarli a terra, l'estensione del nome del file (.sta), nonché le macro inerenti al salvataggio su file dei dati relativi al *timestamp* (solitamente prelevato dal ricevitore GPS), le informazioni sui satelliti, e tutte quelle del modulo sottomarino (o della parte emersa della meda elastica) acquisite ed elaborate localmente dal relativo sistema di acquisizione dati.

File	**massmem.c**
Percorso	/root/programs/module[surface]/
Dipendenze	/root/programs/module[surface]/datastruct.h

	/root/programs/module[surface]/massmem.h
	/usr/local/include/utils.c

Questo modulo contiene le funzioni di gestione dei file orari su memoria di massa relativi ai dati, ed una funzione, denominata *videoPrint()*, che stampa a video, nella modalità di debug, tutte le informazioni acquisite ed elaborate localmente dal sistema di acquisizione dati. La funzione possiede per argomenti le strutture di tipo *typeGps* e *typeStatus* e non ritorna alcun valore.

La funzione *getFileNames()* stabilisce, in base al *timestamp* prelevato dal ricevitore GPS, se disponibile, o dal sistema operativo, il nome del file dei dati nel formato YYMMDDhh.sta. La nomenclatura del nome del file è così definita:

- YY : le ultime due cifre dell'anno;
- MM : le cifre rappresentanti il mese dell'anno (01 per Gennaio, 12 per Dicembre);
- DD : il giorno del mese considerato;
- hh : l'ora UTC di apertura del file (formato 24 ore).

La funzione *createFiles()* crea il file dei dati in modalità *append* di scrittura prelevando le informazioni sulla data e sull'ora dall'argomento di tipo *typeTime*; l'altro argomento, di tipo puntatore a stringa, contiene le informazioni relative al percorso di creazione dei files dati. La funzione non restituisce alcun valore.

Infine, la funzione *writeFile()* scrive su file i dati acquisiti. Essa ha tre argomenti: un puntatore al file dei dati creato dalla chiamata di funzione *createFiles()*, una struttura dati di tipo *typeGps* relativi alla data e all'orario, ed il puntatore alla struttura dati di tipo *typeSPack* contenente tutto il resto.

I file orari (*.sta) generati dal sistema di acquisizione sono tutti in formato testo con i campi di dati ordinati in colonne separate dal carattere di tabulazione orizzontale (H-TAB, avente codice ASCII HEX 09). La formattazione è sempre mantenuta, anche in caso di errori di acquisizione (evidenziati in colonne dedicate) in modo da permettere all'utente una facile visualizzazione dei dati con semplici strumenti software.

I dati sono acquisiti con una frequenza di un campione ogni 10 secondi, e pertanto un file *.sta completo contiene al massimo 360 righe di dati.

Il formato dei files creati e salvati nella memoria micro-SD della scheda FoxBoardG20 è quello mostrato nella Tabella 3 della pagina successiva.

Colonna	Nome del Campo	Esempio	Note
1	Data (DD/MM/YY)	21/01/12	21 Gennaio 2012
2	Tempo (hh:mm:ss)	16:27:00	
3	Latitudine WGS84 (ddpp.pppp)	4049.1877	
4	Emisfero (N/S)	N	
5	Longitudine WGS84 (dddpp.pppp)	1410.9589	
6	Posizione riferita a Greenwich (E/W)	E	
7	Indicatore della Qualità	01	Tipicamente pari a 01
8	Numero dei satelliti in vista	06	
9	Flag dati GPS	OK	Flag = ER (Error) se il ricevitore GPS è fuori uso; in questo caso le colonne da 3 a 8 sono vuote (contengono il carattere NULL=0x00). OK altrimenti
10	Tensione di alimentazione per la sensoristica esterna al modulo di acquisizione dati	12.04	Espressa in Volt

11	Corrente assorbita dai sensori esterni al modulo di acquisizione dati	0104	Espressa in mA
12	Temperatura interna del contenitore metallico dell'elettronica del modulo sottomarino (o del box strumentazione)	31.04	Espressa in Gradi centigradi
13	*n.d.*	9999	Campo fisso a 9999 (per compatibilità con vecchio formato files, precedentemente assegnato alla pressione interna)
14	Allarme intrusione acqua (o apertura sportello box strumentazione)	0	Se dovesse entrare acqua nel contenitore metallico dell'elettronica del modulo sottomarino (o, se lo sportello del box contenente la strumentazione sulla parte emersa della meda elastica dovesse essere aperto) questo flag si setta ad '1'. Normalmente è pari a '0'.
15	*n.d.*	OK	Campo fisso a OK (per compatibilità con vecchio formato files)
16	Heading (hhh.hh)	183.58	Bussola magnetica. Il numero rappresenta l'orientazione del modulo sottomarino (o della parte emersa della meda elastica) in gradi rispetto al Nord magnetico terrestre.
17	Tilt X (-xx.xx)	-4.16	Pitch del modulo sottomarino (o della parte emersa della meda elastica)
18	Tilt Y (-yy.yy)	3.73	Roll del modulo sottomarino (o della parte emersa della meda elastica)
19	Temperatura interna del contenitore metallico dell'elettronica del modulo sottomarino (o del box strumentazione)	31.04	Uguale alla colonna 12; ridondanza (per compatibilità con vecchio formato files, precedentemente assegnato alla temperatura interna registrata da un sensore Compass)
20	Flag dati relativi allo status	OK	Campo fisso a OK (per compatibilità con vecchio formato files)

Tabella 3. Formato dei files dei dati.

Nel malaugurato caso di intrusione acqua nel contenitore metallico dell'elettronica del modulo sottomarino, la CPU del modulo, oltre a segnalare tale situazione impostando ad uno il *flag* relativo (colonna 14) nel file dei dati, invia una e-mail di allarme alla CPU fuori-acqua presente sulla parte emersa della meda elastica, la quale oltre ad inviare una e-mail al personale di terra preposto al controllo della strumentazione del sistema denominato *miniCUMAS*, provvede automaticamente anche a spegnere il sistema sottomarino.

File	**module[surface].h**
Percorso	/root/programs/module[surface]/
Dipendenze	*nessun file delle librerie non standard e di programma*

Questo file di *header* dichiara una struttura del tipo *sgpio*, definita nell'*header* "hm55b.h", necessaria per poter utilizzare la linea digitale PC2 come linea di ingresso per l'attivazione dell'allarme di intrusione acqua (WD_ALARM) nel modulo sottomarino (oppure, per leggere lo stato di apertura dello sportello del box contenente la strumentazione sulla parte emersa della meda elastica).

Per impostare la linea digitale come ingresso e poterne leggere il valore (0/1) si è fatto uso dell'interfaccia GPIO. Questo valore (WD, Water Detector) viene scritto nel file di stato.

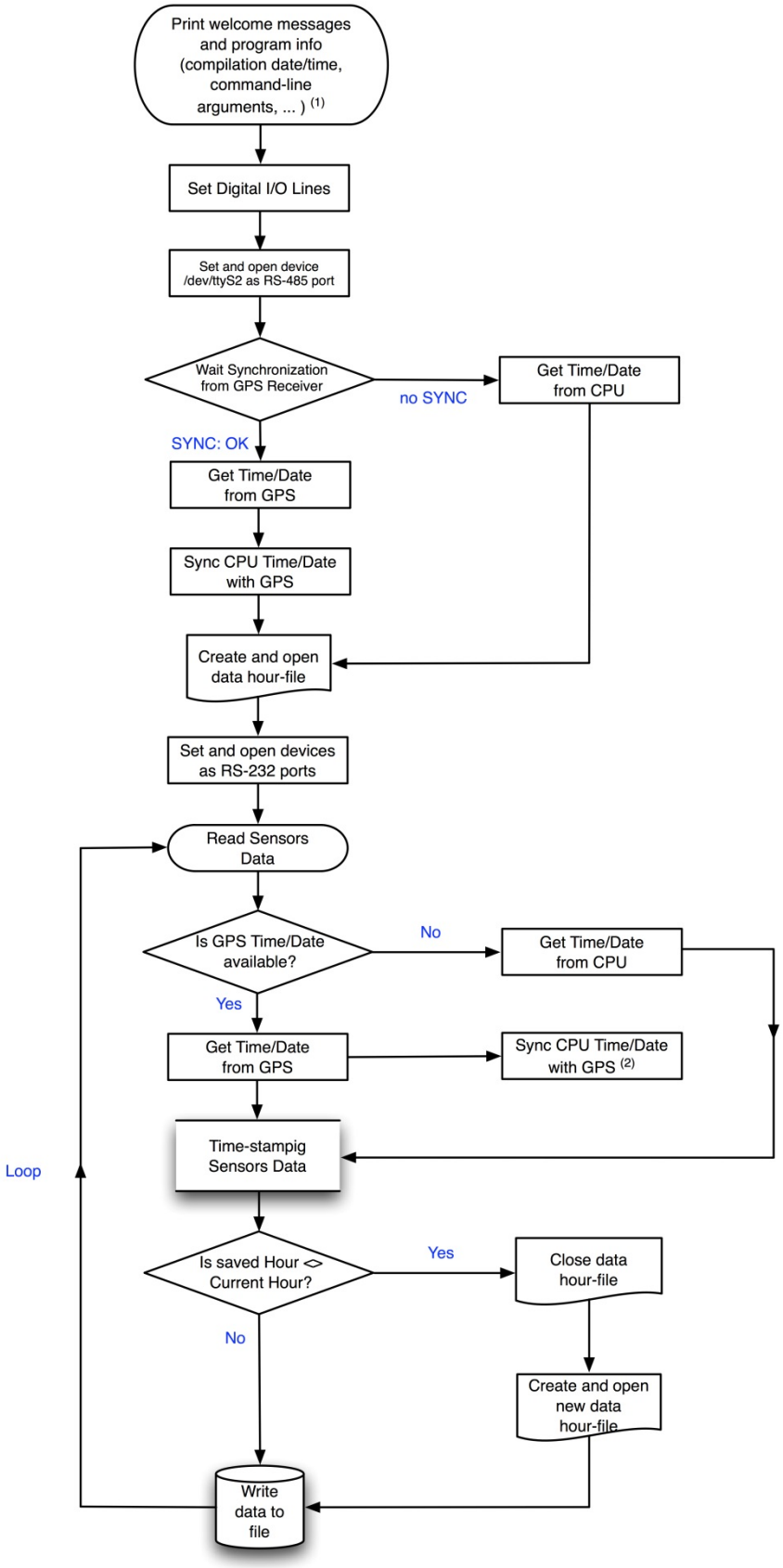
File	**module[surface].c**
Percorso	/root/programs/module[surface]/
Dipendenze	/root/programs/module[surface]/module[surface].h
	/root/programs/module[surface]/status.h
	/usr/local/include/gps/gps.c
	/usr/local/include/485/rs485.c
	/root/programs/module[surface]/massmem.c
	/root/programs/module[surface]/heading.c
	/root/programs/module[surface]/adc.h
	/root/programs/module[surface]/temp.h
	/root/programs/module[surface]/tilt.c

Questo è il modulo principale contenente la funzione *main()* dell'applicativo denominato *MODULE* (o, *SURFACE*) e che gira all'avvio dei servizi del sistema operativo Debian installato sulla CPU. Per la descrizione che segue si faccia riferimento alla Figura 4 della pagina successiva, la quale mostra il diagramma di flusso del programma.

All'avvio del programma sono stampate sul terminale a video dei messaggi di benvenuto seguiti dalle informazioni sul programma, quali: la data e l'ora di compilazione, gli argomenti passati dalla linea di comando ed altro. Ovviamente queste informazioni sono visibili a video solo se il programma è lanciato dalla *shell* di utente, altrimenti esse sono salvate nel file di log *module.log* (oppure, *surface.log*) presente nella directory */usr/local/log/*.

Il modulo utilizza le direttive *#define* per definire: il dispositivo seriale */dev/ttyS2* di sistema da utilizzare come porta seriale di tipo RS-485 per acquisire le informazioni, in formato NMEA, ricavate dal ricevitore GPS esterno; il numero di conversioni al secondo che l'ADC esegue su tutti gli otto canali di ingresso disponibili, attraverso la macro NUM_CONV.

La prima operazione compiuta dal programma è l'impostazione delle linee digitali relative sia alla scheda PIGTAIL sia alla scheda SERIALS. Una di queste linee digitali è usata come ingresso, le altre sono tutte linee di uscita che pilotano dei diodi led per il monitoraggio di alcune delle attività compiute dal sistema di acquisizione dati. Altre linee sono utilizzate per l'accensione e il relativo controllo dei dispositivi seriali di tipo RS-232, SPI e USB. Infine, ulteriori linee digitali di ingresso-uscita sono state utilizzate per implementare le interfacce seriali di tipo I^2C (2 linee) ed SPI (3 linee).



(1) visible on the video-terminal only if the program is launched within the user-shell;
(2) this feature is user-programmable in step of 12-hours.

Figura 4. Diagramma di flusso dell'eseguibile principale *MODULE* (*SURFACE*).

Successivamente vengono impostate ed aperte tre porte seriali, in modalità RS-232, attraverso i dispositivi di sistema opportunamente dedicati ad esse ed utilizzati per la gestione dei sensori del modulo sottomarino direttamente a contatto con l'acqua (quest'ultima funzionalità è valida solo per la versione software denominata *MODULE*)[1]. Per il sistema sottomarino *miniCUMAS*, si tratta di un correntometro acustico, di un sensore di pressione assoluta con elevata risoluzione e di un dispositivo seriale ausiliario definibile dall'utente [Iannaccone, G. 2010; 2009]. Segue poi l'impostazione e la conseguente apertura della porta seriale assegnata per la ricezione dei dati provenienti dal ricevitore GPS installato sulla parte emersa della meda elastica. A questo punto, il microprocessore si mette in attesa del messaggio seriale rappresentante la sincronizzazione del tempo fornito dal ricevitore GPS. Se questo viene fornito entro cinque secondi circa, allora le informazioni relative alla data ed al tempo sono prelevate dal ricevitore GPS e questi dati vanno a sincronizzare la CPU del sistema di acquisizione dati, altrimenti si farà uso delle informazioni disponibili dalla CPU.

L'informazione temporale relativa all'ora viene salvata in una variabile intera (*hour*) ed essa viene usata per la creazione ed apertura del file orario dei dati, come precedentemente descritto. A questo punto viene eseguito il grosso del lavoro della CPU leggendo tutti i sensori del sistema di acquisizione dati (cfr. Figura 4, Read Sensors Data), sia per ciò che concerne la sensoristica a bordo della scheda PIGTAIL che per quelli geofisici ed acustici esterni direttamente a contatto con l'acqua (quest'ultima funzionalità è valida solo per la versione software denominata *MODULE*).

Se le informazioni temporali ricavabili dal GPS sono disponibili, queste vengono usate per marcare temporalmente i dati, altrimenti si fa uso di quelle disponibili con la CPU. I dati vengono quindi scritti nell'apposito file di dati ed il ciclo software riprende indefinitamente con la lettura di tutti i sensori.

Si tenga presente che in quest'ultimo caso, qualora l'informazione temporale del GPS sia disponibile, ad un tempo prefissato e programmabile dall'utente (in passi di dodici ore), avviene la sincronizzazione della data e del tempo fornito dal GPS con quelli della CPU.

Se l'informazione sull'ora attuale dovesse differire da quella in precedenza salvata nella variabile dedicata, allora il file orario dei dati viene chiuso e ne viene creato ed aperto uno nuovo con l'ora corrente.

Software aggiuntivo

Il presente lavoro si chiude con la descrizione dell'ulteriore software a corredo del sistema e consistente in alcuni *script* scritti in *Bash* e fondamentali per il corretto funzionamento del sistema.

L'applicativo principale che gira in *background* sulla CPU (MODULE, per la CPU del modulo sottomarino; SURFACE, per la CPU di superficie della meda) è l'unico che parte, alla fine del processo di *boot* di sistema, con *Runlevel* di default uguale a N 2. Esso risiede nella directory di sistema */etc/init.d/* come se fosse un 'demone' e viene abilitato editando il file *crontab* (opportunamente abilitato) ed aggiungendo la seguente linea:

```
@reboot  /etc/init.d/MODULE &
```

ed analogamente per l'applicativo SURFACE.

L'invio dei files orari contenenti i dati acquisiti dal sistema è gestito da uno script in *Bash*. Questo è attivato dal *crontab* di sistema che gira sulla CPU (sia per il sistema sottomarino, sia per la parte emersa della meda elastica) ed invia, con il comando *scp* di sistema, i relativi files (*.sta) ad un computer dedicato presso la sede dell'Osservatorio Vesuviano.

Inoltre, degli opportuni script *Bash* attivati dal *crontab* di sistema che gira sulla CPU, inviano, a degli opportuni indirizzi e ad intervalli orari di otto ore, tre e-mail che contengono il riepilogo dello stato del si-

[1] Rispetto a quanto descritto nel Rapporto n. 226 dallo stesso autore, nuove e motivate esigenze progettuali hanno reso necessario dotare il modulo sottomarino di ulteriori sensori geofisici. Ciò ha avuto come inevitabile conseguenza la scrittura di ulteriori moduli software dedicati e indipendenti che, tuttavia, per concordanza con i contenuti del precedente lavoro, in questo documento non vengono descritti affinché il lettore possa rimanere in un contesto organico e concorde con la descrizione hardware del sistema.

stema (sia per la CPU del sistema sottomarino sia per quella installata sulla parte emersa della meda elastica).

Una serie di applicativi di test per i vari sensori e dispositivi presenti sul sistema di acquisizione dati risiedono nella cartella */root/programs/TEST/*. Con ognuno di essi, fermando la relativa applicazione principale che qualora dovesse essere attiva, si possono testare singolarmente sensori, adc, linee digitali particolari. Uno strumento molto utile per il debug che facilita l'accesso alle singole periferiche, soprattutto quando qualcosa dovesse andare non per il verso giusto.

Nota

Tutti i files sorgenti descritti nel presente Rapporto Tecnico sono distribuiti e rilasciati liberamente sotto licenza GNU GPL[2] e scaricabili dalla rete, in un unico pacchetto in formato compresso .zip, con il servizio gratuito Google Docs© all'indirizzo seguente:

https://docs.google.com/uc?authuser=0&id=0BwHxFHwcUfGceE9YZUw5RGx0eVk&export=download

Se tutto è andato a buon fine, il download procederà con il salvataggio locale del file denominato *pigtail_fox-DD.MM.AAAA.zip*, dove i numeri DD.MM.AAAA si riferiscono alla data dell'ultima versione del software rilasciata. Una volta decompresso il file appena scaricato, si avrà sulla propria macchina una directory denominata *pigtail_fox-DD.MM.AAAA*. Il file *readme.txt* in esso contenuto ne descrive l'ultima versione disponibile e la struttura delle directory e subdirectories in essa contenute, con l'indicazione della posizione in cui queste vanno copiate.

Bibliografia

Guardato, S. (2012). *Sistema di acquisizione dati a basso consumo basato su Linux per sensori multiparametrici: l'hardware*. Rapporti Tecnici INGV, N. 226 – ISSN 2039-7941.

Iannaccone, G., Vassallo, M., Elia, L., Guardato, S., Stabile, T.A., Satriano, C., and Beranzoli L. (2010). *Long-term Seafloor Experiment with CUMAS Module: Performance, Noise Analysis of Geophysical Signals, and Suggestions about the Design of a Permanent Network*. Seismological Research Letters, Vol. 81, N. 6, pp. 916-927.

Iannaccone, G., Guardato, S., Vassallo, M., Elia, L., and Beranzoli L. (2009). *A new multidisciplinary Marine Monitoring system for the surveillance of Volcanic and Seismic areas*. Seismological Research Letters, Vol. 80, N. 2, pp. 203-213.

AT91SAM9G20 (2009). *AT91 ARM Thumb Microcontroller*. ATMEL.

STTS75 (2007). *Digital temperature sensor and thermal watchdog*. ST Microelectronics, Rev. 7.

HM55B (2005). *Compass Module*. Parallax, Hitachi.

ADS7828 (2005). 12-Bit, *8-Channel Sampling Analog-to-Digital Converter with I^2C Interface*. Burr-Brown from Texas Instruments, SBAS181C.

ADXL203, *Precision ±1.7g Dual-Axis MEMS Accelerometer*, Rev. A. Analog Device.

[2] http://www.gnu.org/licenses/gpl.html

ISSN 1590-2595

Quaderni di Geofisica

http://istituto.ingv.it/l-ingv/produzione-scientifica/quaderni-di-geofisica/

I Quaderni di Geofisica coprono tutti i campi disciplinari sviluppati all'interno dell'INGV, dando particolare risalto alla pubblicazione di dati, misure, osservazioni e loro elaborazioni anche preliminari, che per tipologia e dettaglio necessitano di una rapida diffusione nella comunità scientifica nazionale ed internazionale. La pubblicazione on-line fornisce accesso immediato a tutti i possibili utenti. L'Editorial Board multidisciplinare garantisce i requisiti di qualità per la pubblicazione dei contributi.

ISSN 2039-7941

rapporti tecnici INGV

http://istituto.ingv.it/l-ingv/produzione-scientifica/rapporti-tecnici-ingv/

I Rapporti Tecnici INGV pubblicano contributi, sia in italiano che in inglese, di tipo tecnologico e di rilevante interesse tecnico-scientifico per gli ambiti disciplinari propri dell'INGV. La collana Rapporti Tecnici INGV pubblica esclusivamente on-line per garantire agli autori rapidità di diffusione e agli utenti accesso immediato ai dati pubblicati. L'Editorial Board multidisciplinare garantisce i requisiti di qualità per la pubblicazione dei contributi.

ISSN 2039-6651

Miscellanea INGV

http://istituto.ingv.it/l-ingv/produzione-scientifica/miscellanea-ingv/

La collana Miscellanea INGV nasce con l'intento di favorire la pubblicazione di contributi scientifici riguardanti le attività svolte dall'INGV (sismologia, vulcanologia, geologia, geomagnetismo, geochimica, aeronomia e innovazione tecnologica). In particolare, la collana Miscellanea INGV raccoglie reports di progetti scientifici, proceedings di convegni, manuali, monografie di rilevante interesse, raccolte di articoli ecc..

Coordinamento editoriale e impaginazione
Centro Editoriale Nazionale | INGV

Progetto grafico e redazionale
Daniela Riposati | Laboratorio Grafica e Immagini | INGV

Via di Vigna Murata, 605
00143 Roma
Tel. +39 06518601 Fax +39 065041181

http://www.ingv.it

Istituto Nazionale di Geofisica e Vulcanologia