POLITECNICO DI TORINO

Facoltà di Ingegneria dell'Informazione
Corso di laurea in Ingegneria delle Telecomunicazioni

Tesi di Laurea

Progetto e realizzazione degli apparati di sviluppo, collaudo ed integrazione del satellite PiCPoT

Relatori:
prof. Claudio Sansoè
prof. Leonardo Reyneri

Candidato:
Marco Borri

Luglio 2007

… a chi mi vuole bene …

Con questo lavoro giungo alla fine della mia vita da studente, sono stati anni splendidi, indubbiamente i migliori vissuti finora. Penso quindi sia doveroso ringraziare tutti coloro abbiano in ogni modo contribuito al raggiungimento di questo traguardo. Mi scuso fin d'ora per eventuali dimenticanze.
In primo luogo ringrazio i miei genitori, Anna e Silvano, senza di loro non sarei giunto fin qui. Mi hanno sempre supportato, dandomi fiducia ogni volta stessi per perderla. Mia madre con silenziose parole, mio padre con fragorosi silenzi sono stati il faro nei momenti più bui. Ringrazio mia nonna Flora e mia zia Maria Rosa, anche loro mi hanno sempre sostenuto ed incoraggiato. Ringrazio mio cugino Marco per i preziosi consigli. Ringrazio tutti coloro che mi vogliono bene.
Ringrazio i responsabili del Collegio Universitario Crocetta: Don Aldo, Don Corrado, Don Ermete e Don Nanni, se non fossi stato accolto tra le salesiane mura non mi sarei mai laureato.
Ringrazio i compagni conosciuti tra quelle mura, in particolare Ale, Fenoz, Beltra, Bia, Dondo, Lovi, Vida, Erchy, Cass, Double, Il Napoli, Martino, Ghenghi, Luke, Peo, Luca Minelli, Fianda, Paolo Ferroni, Piero Cavallotti, Ale Buzzi. Con loro è nato qualcosa che difficilmente il tempo cancellerà, un'amicizia che ci accompagnerà per sempre anche se le nostre strade si divideranno. Ringrazio i compagni della squadra di calcio, per tre anni si sono fatti guidare da "uno di loro", consentendomi di vivere un sogno e dandomi grandi soddisfazioni.
Ringrazio Maurizio e Stefano per l'aiuto quotidiano durante questi mesi di "fatiche".
Infine, ma non per questo con minore forza, ringrazio i miei relatori, Prof. Claudio Sansoè e Prof. Leonardo Reyneri, per la fiducia dimostratami, lavorando con loro ho imparato molto.

Sommario

PiCPoT è il nome del primo satellite progettato e realizzato dal Politecnico di Torino. La sua realizzazione ha coinvolto molti professori, ricercatori e studenti di diversi dipartimenti dell'ateneo durante il periodo che va da gennaio 2004 a luglio 2007. Il primo lancio del satellite è avvenuto il 26 luglio 2006 dalla base russa di Baikonur (KAZ) su un razzo vettore Dnepr-LV, di derivazione militare. L'orbita prevista doveva essere una LEO (Low Earth Orbit) con altitudine compresa tra 600 ed 800 Km per garantire un deorbitamento autonomo a causa della resistenza aerodinamica dovuta agli strati alti dell'atmosfera. Il lancio purtroppo è fallito a causa di un problema idraulico sul razzo vettore.

Il satellite ha la forma di un cubo con lato di 13 cm, ricoperto su cinque facce da pannelli solari, fonte primaria di energia del sistema. Sulla sesta faccia sono installate due antenne per le comunicazioni con la terra: una per la trasmissione a 437 MHz, l'altra per i 2,4 GHz. All'interno sono presenti sei schede di controllo, tre fotocamere con differenti distanze focali, sei batterie multi-cella per immagazzinare l'energia da utilizzare nei periodi di eclissi ed una ruota d'inerzia comandata da un motore che consente il controllo attivo da terra del moto di rotazione attorno all'asse del satellite.

Ogni scheda è caratterizzata da una struttura ridondante al fine di garantirne il funzionamento anche in presenza di guasti.

Le funzioni del satellite sono suddivise tra le varie schede come segue:

PowerSupply
Ha il compito di caricare le batterie utilizzando i pannelli solari mentre la selezione della batteria da caricare è affidata ai due processori di bordo. Si occupa anche del condizionamento dei segnali analogici provenienti dai sensori di bordo.

PowerSwitch
Si occupa di generare le diverse tensioni di alimentazione per i sottosistemi del satellite ricevendo energia dalle batterie. Su questa scheda sono inoltre presenti due

microcontrollori che si occupano di attivare i processori di bordo e contare gli interventi dei sistemi anti-latch-up.

ProcA e ProcB
Le due schede fungono da processori di bordo e sono sostanzialmente simili nelle funzioni, ma differenti nelle soluzioni realizzative. Le principali operazioni da esse svolte sono: l'acquisizione dei dati dai sensori di bordo, la creazione dei pacchetti di telemetria, la gestione della carica delle batterie ed il controllo della scheda Payload. La scheda ProcB si occupa inoltre del controllo del motore elettrico connesso alla ruota d'inerzia; questa funzione differenzia le due schede.

PayLoad
Compito della scheda è la gestione delle fotografie che vengono scattate utilizzando le tre fotocamere. Inoltre la scheda si occupa della compressione delle immagini in formato JPEG e della loro trasmissione ai processori di bordo che provvederanno ad inviarle a terra.

TxRx
Ha il compito di far comunicare il satellite con la stazione di terra. Sono previsti due canali di comunicazione half-duplex su bande amatoriali: 437 MHz per la scheda ProcA e 2,4 GHz per la scheda ProcB.

Contributi personali

Il lavoro di tesi è iniziato con lo studio dell'architettura del satellite per capire quali fossero gli apparati esterni necessari durante le fasi di sviluppo, collaudo ed integrazione dello stesso.

In un primo momento sono risultati necessari due strumenti, un caricabatterie ed una stazione di terra portatile. In una seconda fase, quando si è assemblato il satellite si è reso necessario costruire una scheda di interfaccia per consentire la programmazione dei processori a satellite assemblato.

Le funzioni di questi tre apparati sono le seguenti:

Caricabatterie
Si occupa di caricare i sei gruppi di batterie installati a bordo di PiCPoT tramite il connettore di test. Procede alla carica rispettando appieno le specifiche degli accumulatori in modo da usurarli il meno possibile. E' stato utilizzato durante la fase di sviluppo e collaudo, ma soprattutto durante la fase di integrazione a bordo del lanciatore.

Scheda di interfaccia
Permette di effettuare la programmazione dei processori di bordo senza dover smontare le schede. La sua realizzazione si è resa necessaria a causa dell'incompatibilità dei processori utilizzati ad essere programmati in catena. Oltre alla realizzazione della scheda è stato necessario apportare alcune piccole modifiche alle schede del satellite, installando cavi di collegamento tra i processori e l'esterno del satellite. Questi collegamenti vengono rimossi a collaudi ultimati.

Stazione di terra portatile
Consente di comunicare con il satellite a breve distanza, ha funzioni ridotte rispetto alla stazione di terra vera e propria, ma si è rivelata utilissima sia in fase di sviluppo e collaudo sia durante l'integrazione sul lanciatore. Infatti la stazione di terra vera e propria non era utilizzabile mentre si procedeva, mentre era necessario effettuare alcune prove di trasmissione a breve distanza poco prima del lancio per accertarsi del corretto funzionamento di tutti i sistemi.

La realizzazione di questi apparati ha comportato la risoluzione di differenti problemi di interfaccia e lo studio di sistemi in grado di collegarsi al satellite modificando il meno possibile l'architettura dello stesso. Ciò ha ovviamente condizionato gli schemi circuitali degli stessi, rendendone, in alcuni casi, più complessa la realizzazione.

Lavorare col team PiCPoT è stata un' esperienza formativa esaltante e stimolante che mi ha permesso di apprendere come viene affrontata la realizzazione di un progetto così complesso oltre ad accrescere le mie conoscenze in questo settore della ricerca.

La struttura di questa dissertazione è la seguente:

Capitolo 1: *descrive il progetto nel suo insieme, la sua struttura ed i vincoli ambientali a cui sarà sottoposto.*

Capitolo 2: *descrive il lavoro di tesi nel suo insieme, introducendo le problematiche da affrontare.*

Capitolo 3: *descrive la struttura del caricabatterie, gli schemi circuitali e le motivazioni che hanno portato alle scelte realizzative.*

Capitolo 4: *descrive il programma di controllo del caricabatterie, gli algoritmi utilizzati, le specifiche che hanno influito sulle scelte realizzative e l'interfaccia grafica per l'utente.*

Capitolo 5: *descrive gli schemi di realizzazione e collegamento della scheda di interfaccia per la programmazione del satellite.*

Capitolo 6: *descrive gli schemi elettrici della stazione di terra portatile.*

Capitolo 7: *è dedicato alle conclusioni.*

Appendice A: *schemi elettrici.*

Appendice B: *fogli tecnici dei principali componenti utilizzati.*

Appendice C: *listato del programma di controllo del caricabatterie*

Indice

Capitolo 1

Introduzione al progetto PiCPoT

L'attenzione di molti progetti di ricerca si è rivolta, negli ultimi anni, al campo aerospaziale ed in particolare alla progettazione e realizzazione di pico-nano-satelliti. Un pico-satellite è un satellite la cui massa è inferiore ad un chilogrammo, mentre un nano-satellite ha una massa compresa tra 1 e 10 chilogrammi.

Un progetto di questo tipo ha un evidente scopo didattico, identificabile nella possibilità di far collaborare dipartimenti diversi, usufruendo così delle diverse competenze necessarie alla realizzazione, con la necessità di una grande capacità di collaborazione tra i vari soggetti interagenti.

I satelliti in questione sono sistemi mediamente complessi, questa loro particolarità li rende adatti al lavoro di tesi, poiché il loro studio permette di affrontare un problema progettuale serio ma comunque gestibile da uno studente cui sia stato affidato questo genere di ricerca.

Il punto di partenza del progetto PiCPoT è stato il nodello del Cubesat.

1.1 Il progetto Cubesat

Con il termine Cubesat si intende uno standard per pico-satelliti: una struttura cubica di 10 centimetri di lato e con una massa massima di 1 chilogrammo la cui struttura è definita in funzione dell'adattamento al lanciatore POD (Picosatellite Orbital Deployer). Lo standard è stato sviluppato nel 2001 dal Professor Robert Twiggs, docente alla Stanford University, USA, in collaborazione con lo Space System Develoment Laboratory (SSDL) della Stanford University e la California Polytechnic state University, USA, per permettere alle Università che intendono partecipare a questa

sperimentazione di realizzare un proprio satellite e di metterlo in orbita a costi contenuti.

Cubesat significa satellite di forma cubica e risponde nel contempo a una specifica filosofia progettuale. L'idea alla base di Cubesat è dunque quella di permettere agli studenti dei vari dipartimenti di ingegneria di cooperare per la realizzazione di un progetto che richiede contemporaneamente ottime conoscenze in campo ingegneristico e capacità di risolvere problemi di natura interdisciplinare.

L'idea di progettare un satellite è stata adottata da numerose università italiane e straniere, fra le quali si possono citare l'Universitat Wurzburg (D), la Norwegian University of Science and technology (NO), l'Aalborg University (DK) e l'Università "La Sapienza" di Roma (I). AAU-Cubesat, CanX-1, NCube, UWE-1 ed UNISAT sono alcuni esempi di satelliti già in orbita realizzati rispettivamente dalle Università appena ricordate.

Anche il Politecnico di Torino si è impegnato a partecipare alla progettazione ideando PiCPoT e collocandolo in un progetto più ampio, quello di una costellazione di satelliti universitari italiani, poi posticipato per i numerosi ritardi.

1.2 Il Progetto PiCPoT

Il progetto PicPoT, Piccolo Cubo del Politecnico di Torino, è nato nell'autunno 2003 con lo scopo di costruire, nell'arco di un anno, un nano-satellite universitario, avente massa inferiore a 10 chilogrammi, con finalità didattiche e di ricerca.

PiCPoT è un progetto a cui partecipano docenti, ricercatori, dottorandi e laureandi dei Dipartimenti di Ingegneria Elettronica, Ingegneria Aerospaziale, di Fisica ed Energetica del Politecnico di Torino.

La progettazione del satellite è stata così suddivisa per competenze tra vari Dipartimenti:

- la struttura, il calcolo dell'orbita ed i problemi di natura organizzativa sono stati delegati al Dipartimento di Ingegneria Aerospaziale;
- i computer di bordo, la gestione della potenza, l'acquisizione dei dati di telemetria e le comunicazione sono invece delegati al Dipartimento di Elettronica;
- i calcoli termici affidati al Dipartimento di Fisica e di Energetica

Il progetto del satellite si è basato sui seguenti requisiti:

- forma cubica con lato di 13 cm;
- massa inferiore a 5 chilogrammi;
- potenza non superiore a 1,5 Watt;
- vita di almeno 90 giorni;
- utilizzo di componenti COTS (Commercial Off The Shelf) in ambiente spaziale;

- orbita LEO (Low Earth Orbit).

Le funzioni principali del satellite sono:

- acquisire misure di temperatura ed illuminamento;
- scattare fotografie;
- trasmettere dati alla stazione di terra.

La comunicazione terra-satellite avviene su due frequenze: 437 MHz e 2.4 GHz, di prevalente uso amatoriale. L'utilizzo dei 2.4 GHz rappresenta anch'esso una sperimentazione, dato che a tale frequenza vi sono gravi problemi a causa dell'effetto di attenuazione atmosferica.

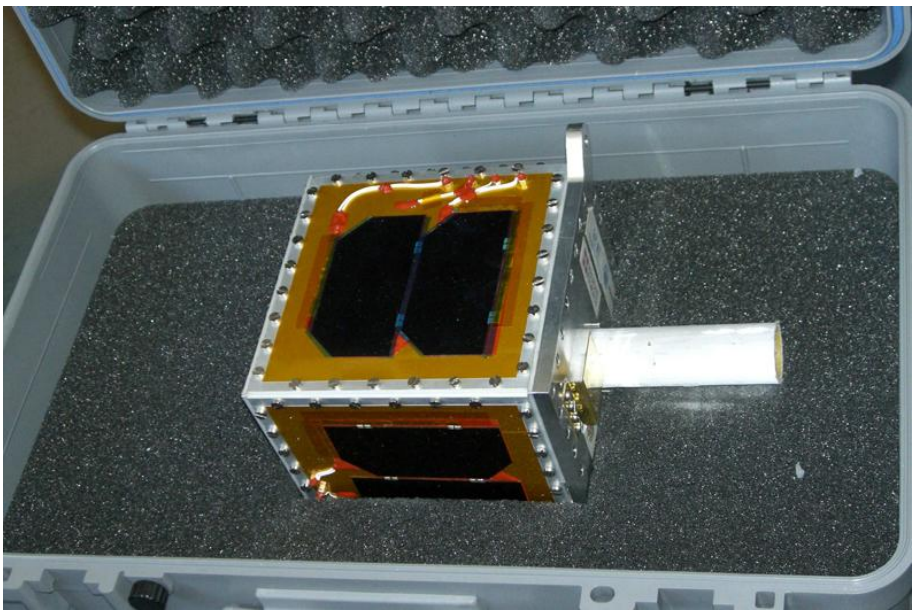


Figura 1.1 – *PiCPoT all'interno del contenitore per il trasporto*

1.2.1 L'ambiente spaziale

Lo spazio è l'ambiente più ostile in cui possa funzionare un dispositivo elettronico, in quanto i problemi legati ad esso sono numerosi, ed in particolare:

- vuoto (ridotta dissipazione termica);
- vuoto (degassamento);
- temperatura;
- radiazione (latch-up);
- radiazione (SEU – Single Event Upset);
- radiazione (total dose).

Per risolvere o limitare i problemi dovuti a questi effetti, si sono utilizzati, rispettivamente, i seguenti accorgimenti, assumendo in ogni caso che effetti del total dose siano trascurabili data la vita del satellite:

- sovradimensionamento e dissipatori volumici;
- eliminazione di componenti liquidi, pressurizzati e rotativi con spazzole;
- analisi termica (dip Energetica), selezione dei componenti e monitoraggio delle temperature;
- componenti non C-MOS, layout meccanico di schermatura e circuiti di protezione dagli effetti del latch-up;
- memorie ferro-elettriche e flash, ridondanza di programma e watch-dog timer.

Entrando nel particolare, vediamo ora come alcune di queste grandezze fisiche possono influenzare l'elettronica di bordo di un satellite.

1.2.2 I sottosistemi

Vista esterna

PiCPoT ad un primo sguardo è un cubo con lato di circa 13 centimetri, su cinque delle cui facce sono installati dei pannelli solari. E' dotato di due antenne (una per ognuna delle frequenze di trasmissione utilizzate), installate sulla faccia libera. Sulla stessa faccia delle antenne sono fissate tre fotocamere digitali, due kill-switch. Questa stessa faccia (chiamata F0) viene utilizzata per l'aggancio al lanciatore.

Le tre fotocamere (come si può vedere in figura 1.2) hanno asse ottico parallelo all'asse X e direzione di vista positiva. Sono installate all'interno del satellite con la lente a filo di F0.

Ogni fotocamera ha la medesima risoluzione (512 linee per 582 colonne), ma lunghezza focale diversa, ovvero:

- T1, lunghezza focale 3.6 mm, campo visivo totale sulla diagonale 68° di 750 km sulla terra con una risoluzione di 800 m/pixel;
- T2, lunghezza focale 6 mm, campo visivo totale sulla diagonale 42° di 450 km sulla terra con una risoluzione di 500 m/pixel;
- T3, lunghezza focale 16 mm, campo visivo totale sulla diagonale 17° di 170 km sulla terra con una risoluzione di 180 m/pixel.

Ciascuna fotocamera è indipendentemente avvitata alla faccia F0, tramite un foro filettato che ne garantisce la perpendicolarità alla faccia stessa, ma non l'orientamento assiale. Questo dovrà essere misurato a terra dopo l'assemblaggio al fine di orientare spazialmente le fotografie ricevute.

Essendo l'asse ottico delle fotocamere parallelo all'asse X, il centro del campo visivo non dipenderà dalla rotazione di PiCPoT intorno a tale asse (l'asse X dovrebbe essere l'asse di spin del satellite).

Sulle cinque facce montanti le celle fotovoltaiche vengono installate celle solari in arseniuro di gallio (GaAs), eroganti 4,9 Volt nominali, interconnesse ed assemblate in pannelli laminati, incorporanti un sensore di temperatura ciascuno. Sulla faccia F5 è montato un connettore di test (CT), si tratta di un connettore a vaschetta miniatura a 25 poli. A causa dello spazio occupato dal connettore, il pannello solare montato su F5 ha dimensioni ridotte rispetto agli altri.

Il connettore di test serve da interfaccia tra gli apparati di collaudo ed integrazione del satellite ed il satellite stesso. Agendo sui sistemi interni a satellite assemblato, dovrebbe permettere di monitorarlo e consentire di ricaricarne le batterie durante la procedura di integrazione. Questo connettore è collegato alla scheda ProcA, e su di essa, è ancora collegato con il connettore J0.

Il connettore J0 è un connettore passante a 140 poli installato su tutte le schede tranne sulla scheda PowerSupply. Agisce da bus di collegamento tra le schede, occupando meno spazio rispetto ad un cavo.

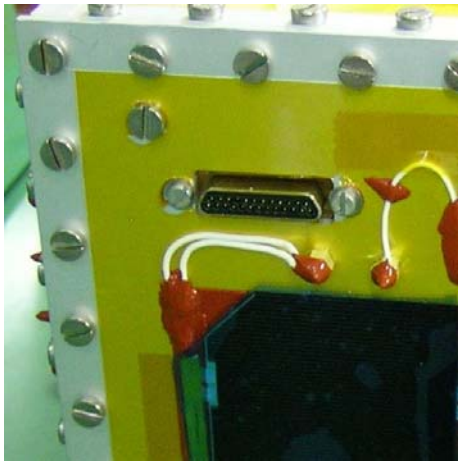


Figura 1.2 – *Particolare del connettore di test (CT)*

Vista interna

PiCPoT è alimentato da sei gruppi di batterie ricaricabili (batterie secondarie), disposte fra i pannelli solari e le schede elettroniche per svolgere anche la funzione di schermo contro le radiazioni. Sono appoggiate all'interno della faccia sul lato di superficie maggiore, circondate da uno strato di gomma elettricamente isolante, termicamente conduttivo e tale da svolgere anche effetto protettivo.

Vengono utilizzati due tipi di batterie:

- B1, B2, B4, B5 sono costituite da due celle a litio-polimeri (Li-P) collegate in serie, per un totale di 7,2 Volt nominali, capacità di 1.500 mAh, modello ULTRALIFE Batteries UBC425085.

- B3, B6 sono costituite da sei celle al nichel-cadmio (Ni-Cd) collegate in serie, per un totale di 7,2 Volt nominali, capacità di 1.000 mAh, modello SANYO Hygh Capacity size AA.

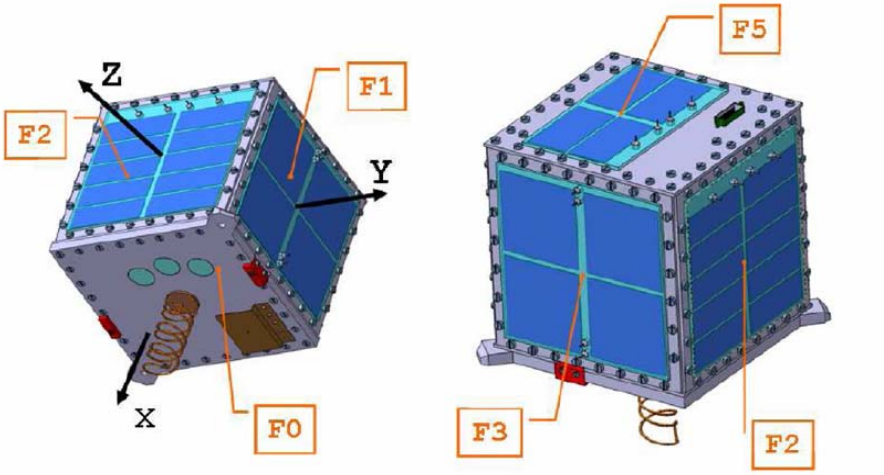


Figura 1.3 – *Vista esterna di PiCPoT*

L'utilizzo di due kill-switch è dovuto alla perentoria richiesta del lanciatore. I due interruttori verranno connessi in parallelo, per ovviare ad un eventuale guasto. I kill-switch interromperanno la massa comune delle batterie andando quindi a spegnere completamente il satellite fino al momento dello sgancio, per evitare che il sistema di bordo interferisca con l'elettronica del lanciatore.

PiCPoT non ha un controllo d'orbita. La sua orbita dipende esclusivamente dalla sua posizione e dalla velocità al momento del distacco dal lanciatore quindi non può essere controllato. L'orbita viene però tenuta sotto stretto controllo dal servizio NORAD che periodicamente rileva la posizione del satellite tramite radar e rende disponibili su internet le effemeridi per i successivi tre giorni. Tali effemeridi verranno utilizzate per il puntamento delle antenne di terra.

L'assetto del satellite invece è impostato da un magnete permanente posizionato all'interno dello stesso, parallelo all'asse X (quindi ortogonale a F0 e F5), con il sud magnetico prossimo a F0 (direzione positiva di X) ed il nord magnetico prossimo a F5. Questo dovrebbe favorire l'allineamento di F0 in direzione del nord magnetico terrestre e quindi orientare lievemente F0 verso la superficie terrestre, favorendo l'inquadramento di quest'ultima da parte delle tre fotocamere.

Ad ogni rivoluzione del satellite, il suo asse X viene ruotato di quasi 360°, causando un'oscillazione (nutazione) che verrà smorzata passivamente. A tal scopo verranno installati a bordo degli smorzatori magnetici isteretici e verranno sfruttate le correnti Eddy nelle superfici laterali del satellite. L'assetto sarà controllato attorno all'asse di spin da una ruota di inerzia con asse parallelo all'asse X. Una rotazione della ruota causerà una rotazione inversa del satellite.

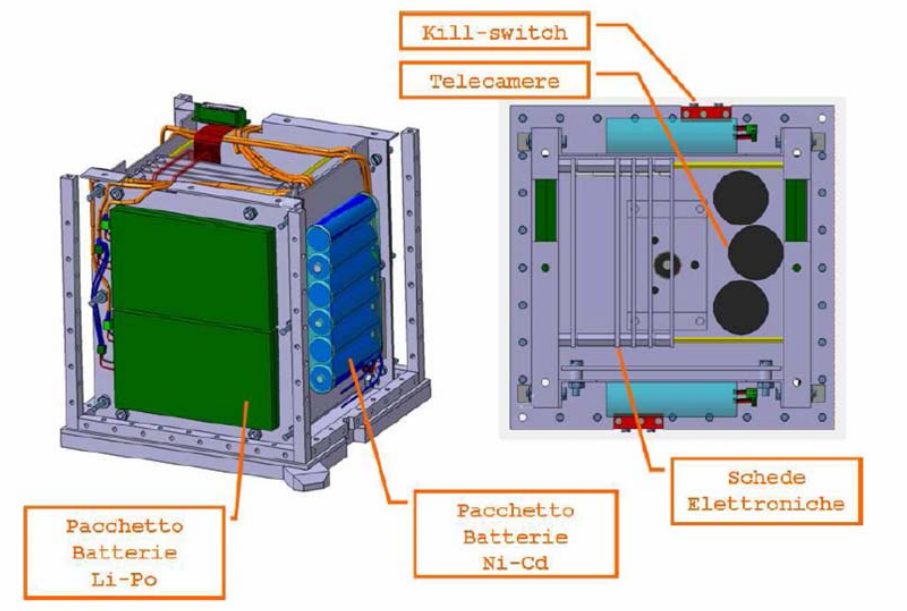


Figura 1.4 – *Vista interna di PiCPoT*

La ruota di inerzia viene mossa da un motore brushless MAXXON tipo ECflat32, con alimentazione di 9 Volt ed assorbimento di 6 Watt nominali. Il motore ha tre avvolgimenti con connessione a stella e tre sensori di Hall. Verrà controllato ad anello aperto, utilizzando i sensori di Hall e la scheda ProcB, sarà attivato dai telecomandi ricevuti da terra.

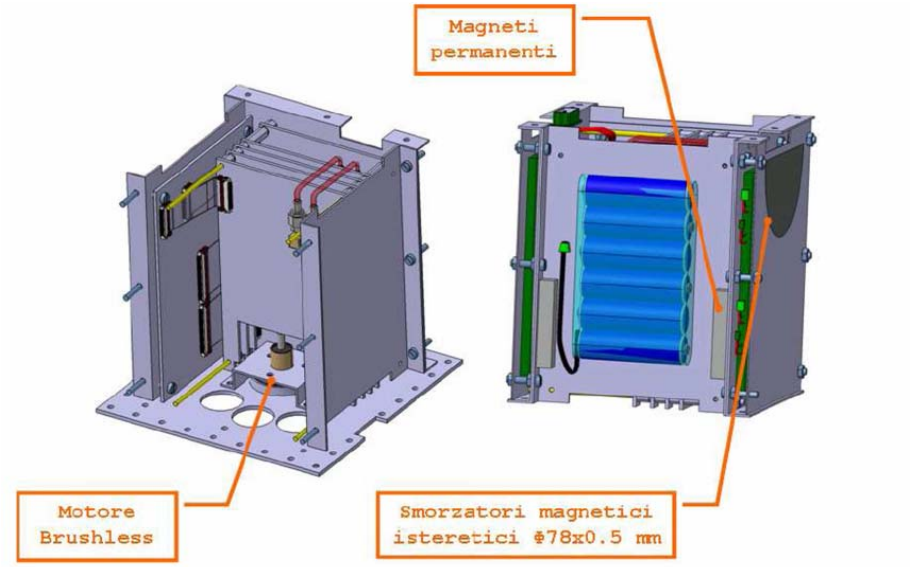


Figura 1.5 – *Vista interna di PiCPoT, particolare del motore e dei magneti permanenti*

PiCPoT contiene al suo interno tre processori:

- ProcA, utilizzato per la gestione di bordo (housekeeping), associato al canale di comunicazione a 437 MHz. E' stato utilizzato un processore Chipcon 1010 con al suo interno il modulatore/demodulatore. Frequenza di clock di 11 MHz e tensione di alimentazione di 3,3 Volt.
- ProcB, per la gestione di bordo, associato al canale di comunicazione a 2,4 GHz. E' stato utilizzato un processore Texas Instruments MSP430 a basso consumo. Frequenza di clock 4 MHz e tensione di alimentazione di 3,3 Volt.
- Payload, per la gestione del payload (fotocamere T1, T2, T3). E' stato utilizzato un processore Analog Devices BlackFIN, dotato di interfacce dedicate per le fotocamere. Frequenza di clock di 100 MHz e tensione di alimentazione 1,8 Volt.

I due processori di housekeeping (ProcA e ProcB) sono indipendenti fra di loro, con una conseguente duplicazione delle funzioni vitali del satellite da cui si ottiene una ridondanza calda. Questi due processori condividono tutti i circuiti di condizionamento dei sensori, ma entrambi dispongo di un multiplexer analogico dedicato.

I tre processori sono alimentati (e quindi avviati) periodicamente ed eseguono sempre lo stesso codice. Tramite un telecomando si possono eventualmente scaricare su di essi altrettanti programmi utente (uno per ciascun processore) ed eseguirli su richiesta di un altro telecomando.

A bordo, oltre ai sensori sono disponibili data ed ora. Quest'ultima è tenuta sincronizzata con l'ora di Greenwich dai telecomandi, mentre un orologio a bordo mantiene il tempo fra telecomandi successivi.

I sensori vengono monitorati dai processori ogni minuto ed i valori misurati vengono trasmessi automaticamente a terra come telemetria (THM). I valori massimi, minimi, medi e la deviazione standard di ciascun sensore sono salvati nel pacchetto di telemetria estesa (THME) per un tempo massimo di 250 minuti. Un comando da terra può resettare i dati memorizzati nella THME.

PiCPoT comunica a terra tramite due canali di trasmissione (anche questo per motivi di ridondanza), entrambi i canali sono in banda amatoriale. La trasmissione avviene con un protocollo APRS, rispettivamente:

- ProcA a 437,485 MHz per TC1, bit-rate di 9.600 bit/s, modulazione di frequenza 2-FSK con deviazione di 6 KHz;
- ProcB a 2440 MHz per TC2, bit-rate di 10.000 bit/s, modulazione di frequenza GFSK con deviazione di 125 KHz;

La trasmissione dei telecomandi avviene con un protocollo dedicato, utilizzando una unique-word di 40 bit, preceduta da un preambolo e seguita da telecomando di 12+27 bit (più eventualmente un programma eseguibile) codificati con un codice a parità dispari. Il protocollo non è stato pubblicizzato per evitare comandi indesiderati da parte di terzi.

1.2.3 Le schede

Le schede elettroniche all'interno del satellite sono sei: PowerSupply, PowerSwitch, ProcA, ProcB, Payload e TxRx. esse sono direttamente od indirettamente tutte in comunicazione tra loro, ad eccezione di ProcA e di ProcB, le quali, come sopra accennato, non hanno segnali in comune.

Procediamo ad una breve descrizione delle schede.

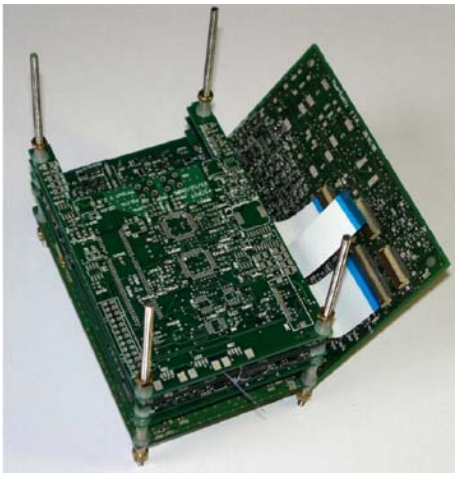


Figura 1.6 – *Pacco schede assemblato*

PowerSupply

Questa scheda svolge diversi compiti, si occupa infatti di mantenere in carica le batterie, di monitorare lo stato elettrico e termico dei pannelli solari, dei circuiti di carica delle batterie e delle batterie stesse.

E' così strutturata:

- cinque caricabatterie che ricevono potenza dai rispettivi pannelli solari;
- sensori di corrente e tensione media erogate dai cinque pannelli solari (questi sensori sono alimentati dalla scheda PowerSwitch, quando ProcA e ProcB ne fanno richiesta);
- sensori di corrente e di tensione per i cicli di carica e scarica delle batterie (alimentati e controllati da ProcA e ProcB);
- circuiti di condizionamento dei cinque sensori di temperatura (NTC) che rilevano la temperatura dei pannelli solari (alimentati e controllati da ProcA e ProcB);
- circuiti di condizionamento per sei sensori NTC che controllano la temperatura delle batterie (alimentati e controllati da ProcA e ProcB);
- circuito di condizionamento per un sensore NTC monitorante la temperatura dei caricabatterie (alimentato e controllato da ProcA e ProcB);

- due multiplexer (comandati da ciascun processore) per la lettura sequenziale delle temperature, delle tensioni e delle correnti sopra citate;
- due circuiti digitali (per ridondanza) e sei gruppi di quattro interruttori PMOS (connessi in due serie collegate in parallelo per sopperire ad eventuali guasti) per la selezione della batteria da caricare; questi circuiti sono sempre autoalimentati;
- due alimentatori (per ridondanza) non regolati per alimentare i cinque caricabatterie ed i suddetti circuiti digitali.

PowerSwitch

Questo sottosistema si occupa di distribuire l'energia agli altri sottosistemi, minimizzando i consumi e massimizzando l'efficienza. Tra le schede con microcontrollore installato è l'unica ad essere sempre alimentata poiché si occupa della sincronizzazione del satellite attraverso due orologi O1 ed O2 (master e slave rispettivamente) integrati nei rispettivi microcontrollori. Anche PowerSwitch segue il principio della ridondanza, è quindi suddivisa in due catene completamente indipendenti.

Le due catene, nominate A e B, hanno installati microcontrollori diversi, rispettivamente, un pic16F877A di Microchip ed un MSP430 di Texas Instruments. Questi due sistemi, pur essendo sempre alimentati, si attivano una volta ogni minuto con uno sfasamento di 30 secondi , per ottenere il quale viene utilizzato un timer. Durante l'attivazione di uno, l'altro entra in "low-power mode" limitando così il consumo di energia. Il processore master (A) invia un segnale di sincronismo al processore slave (B) al trentesimo secondo. Nei primi cinque secondi di funzionamento ogni controllore alimenta i circuiti di condizionamento dei sensori di temperatura, consentendo al processore di acquisirne i valori. Ogni catena si occupa di tre delle sei batterie installate sul satellite: la catena A si occupa di B1, B2, B3 e la catena B delle restanti.

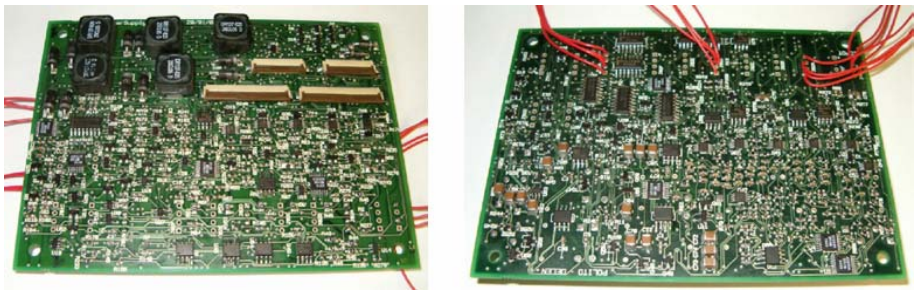


Figura 1.7 – *La scheda PowerSupply*

PowerSwitch si occupa di:

- definire quale batteria utilizzare per alimentare il satellite;
- generare l'alimentazione per tutti gli altri sottosistemi, proteggendoli da latch-up, dove richiesto;

- attivare le altre schede (fornendo o meno l'alimentazione), su base periodica;
- mantenere una coppia di orologi di sistema, sincronizzati tra loro, che periodicamente attivino i rispettivi processori;
- conteggiare il numero di interventi dei circuiti di protezione da latch-up, per ogni alimentatore;
- condizionare e multiplare i sensori di tensione e corrente all'uscita degli alimentatori.

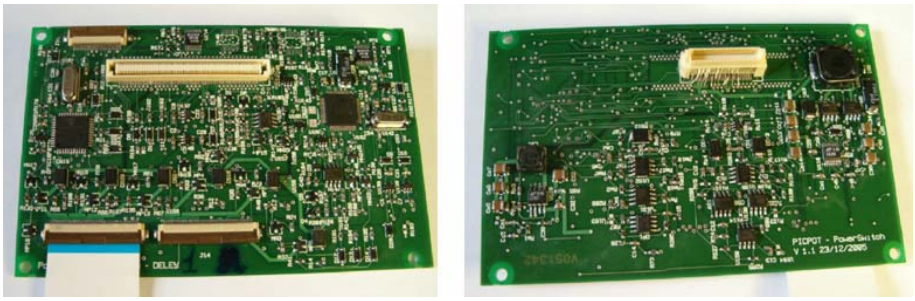


Figura 1.8 – *La scheda PowerSwitch*

ProcA e ProcB

Le due schede processore sono sostanzialmente simili. Si è scelto di duplicarle per motivi di ridondanza. Sono progettate da persone diverse, utilizzando parzialmente componenti di diverso tipo, pur mantenendo funzionalità sostanzialmente equivalenti, inoltre non comunicano mai tra di loro.

La scheda ProcA utilizza un microcontrollore Chipcon CC1010 (core 8051 + modulatore e demodulatore a 437 Mhz) ed una FPGA per svolgere le sue funzioni.

La scheda ProcB utilizza un processore Texas Instruments MPS430 e come modulatore e demodulatore un Chipcon CC2400, installato sulla scheda TxRx.

Vi sono altre differenze:

- nella versione definitiva il controllo del motore è affidato solo alla scheda ProcB;
- i connettori per le fotocamere sono presenti solo sulla scheda ProcA;
- i segnali di selezione del connettore di test sono forniti solo da ProcB alla scheda PowerSupply;
- il connettore di test è connesso su ProcA;

Poiché le schede processore vengono spente e accese ogni minuto, non è necessario un sistema "watch-dog". Infatti, un eventuale problema transitorio causerebbe solo la perdita di un ciclo di trasmissione, in quanto, spegnendo e accendendo il sistema, questo viene ripristinato ogni minuto.

Le funzioni principali svolte sono:

- acquisire i dati dai sensori per creare la telemetria;
- attuare la strategia di carica delle batterie;

- ricevere i telecomandi;
- eseguire i telecomandi ricevuti;
- trasmettere le telemetrie alla stazione di terra;
- comunicare con la scheda Payload;
- comunicare con la scheda PowerSwitch;
- attivare la ruota d'inerzia (solo ProcB)

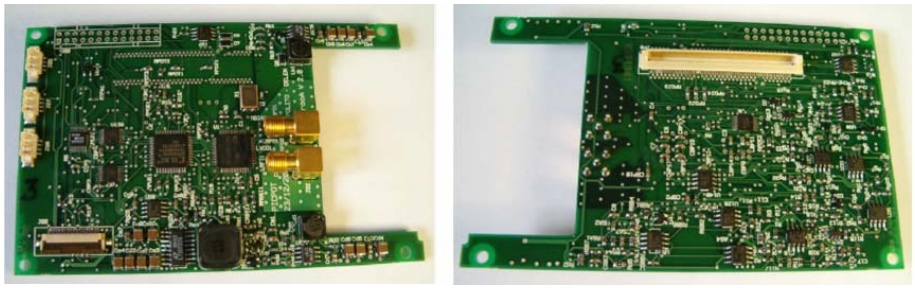


Figura 1.9 – *La scheda ProcA*

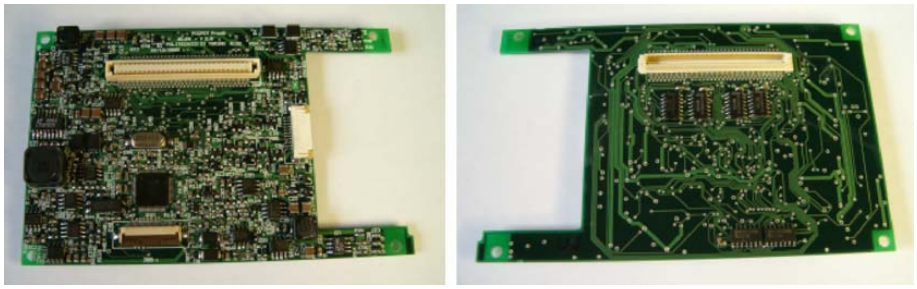


Figura 1.10 – *La scheda ProcB*

Payload

Quando viene richiesto da terra, questo sottosistema deve scattare fotografie, comprimerle in formato JPEG ed inviarle ad una delle schede ProcA o ProcB per la trasmissione a terra. La scheda si occupa quindi della gestione delle tre fotocamere.

Le fotocamere vengono alimentate individualmente per il tempo strettamente necessario (al massimo tre secondi). Il segnale d'uscita delle tre fotocamere viene inviato ad un decoder digitale, che converte il segnale PAL in uno stream digitale a 8 bit di densità più 4 + 4 bit di crominanza.

Lo stream digitale viene acquisito direttamente dal processore (BlackFINN, prodotto dalla Analog Devices) tramite una sua porta dedicata e trasferito via DMA alla memoria dati esterna al processore. Viene acquisito un solo fotogramma alla volta.

L'immagine viene poi spezzata in nove blocchi, ognuno dei quali viene compresso in formato JPEG via sofware. i nove blocchi compressi vengono poi memorizzati in una delle 5 zone di memoria dati, come richiesto dal telecomando. Si reputa che il tempo di

compressione totale sia inferiore a 15 secondi e che la dimensione dell'immagine compressa sia al massimo 45 KB.

L'ordine col quale svolgerà queste operazioni è il seguente:

- La scheda, su richiesta di una delle schede processore, viene alimentata;
- tramite interfaccia SPI il processore acquisisce il comando dalla scheda processore e lo interpreta;
- se è in grado di attuare il comando ricevuto, risponde tramite la stessa SPI con un segnale di conferma (se sta già eseguendo un comando richiesto dall'altro processore risponde negativamente);
- alimenta una delle tre fotocamere;
- scatta una foto;
- spegne la fotocamera;
- spezza l'immagine in nove blocchi;
- comprime le nove immagini e le memorizza in memoria (oppure, a seconda del comando ricevuto, invia l'immagine compressa che è stata richiesta);
- appena eseguito il comando, la scheda processore toglie l'alimentazione.

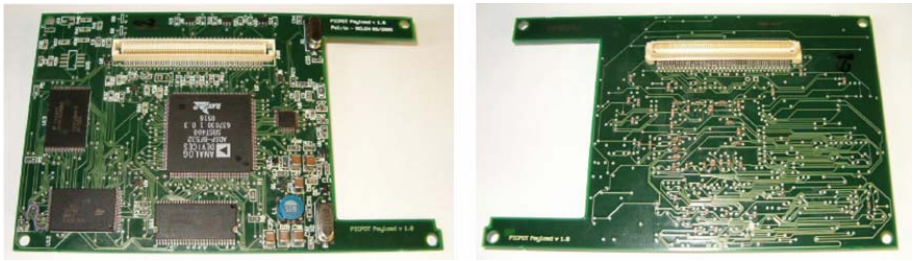


Figura 1.11 – *La scheda Payload*

TxRx

Questa scheda ha il compito di far comunicare il satellite con la stazione di terra. Sono previsti due canali di trasmissione. Ogni canale è half-duplex (trasmissione e ricezione mutuamente esclusive).

La scheda è composta da:

- un amplificatore di potenza sulla banda 437 MHz che riceve il segnale modulato in frequenza da ProcA e fornisce all'antenna un segnale con potenza di 30 dBm;
- un commutatore d'antenna per l'antenna a 437 MHz per connettere l'antenna all'amplificatore di potenza od al demodulatore presente su ProcA;
- un modulatore a 2,4 GHz (Chipcon CC2400) che riceve attraverso una linea seriale i dati da inviare alla scheda ProcB;
- un amplificatore di potenza sulla banda 2,4 GHz, che fornisce all'antenna un segnale con potenza di 30 dBm;

- un demodulatore a 2,4 GHz (Chipcon 2400) che demodula il segnale radio ed invia i dati decodificati alla scheda ProcB;
- un amplificatore a basso rumore a 2,4 GHz, che permette di avere in ricezione una sensibilità di -118 dBm;
- un commutatore d'antenna per l'antenna a 2,4 GHz che connette l'antenna all'amplificatore di potenza (PA) oppure all'amplificatore a basso rumore (LNA).

Ad ogni attivazione, e prima di ogni ricezione e trasmissione, la scheda ProcB programma i due transceiver CC2400 tramite l'interfaccia seriale SPI. Tramite la connessione SPI vengono anche forniti i comandi per selezionare la modalità di ricezione e trasmissione..

Per quanto riguarda il canale a 437 MHz avviene solo l'attivazione dell'amplificatore poiché il resto è integrato sulla scheda ProcA.

La scheda TxRx è quella che presenta la maggiore criticità in fatto di dissipazione di potenza; per questo motivo attorno ai componenti più critici, è stata depositata una speciale resina termicamente conduttiva.

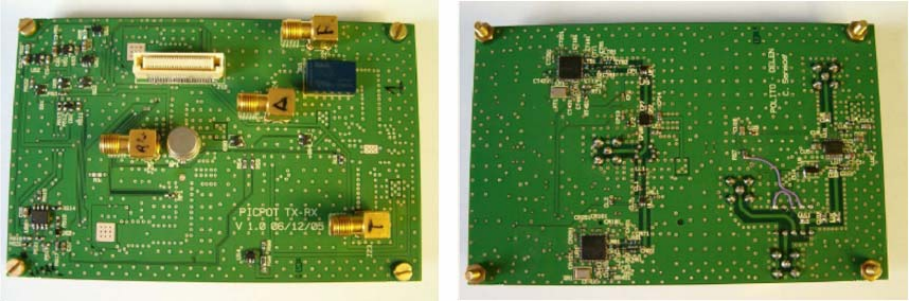


Figura 1.12 – *La scheda TxRx*

1.2.4 I numeri del progetto

In questo paragrafo vengono conclusivamente riportati una serie di dati, per rendere meglio l'idea della complessità del progetto.

Caratteristiche del satellite:

- forma cubica;
- lunghezza del lato 130 millimetri;
- peso 2,5 chilogrammi
- baricentro (x;y;z) = (36,54; 66,32; 64,81) millimetri;
- momento d'inerzia (I_{xx}; I_{yy}; I_{zz}) = (0,015; 0,022; 0,022) Kg m^2;
- temperatura di lavoro da -50°C a +150°C;
- potenza media 1,5 Watt
- tempo per lo sviluppo dell'avionica 9 anni-uomo:
  6,5 anni-uomo da studenti (72%);
  1,5 anni-uomo da professori e PhD (17%);
  1 anno-uomo per il coordinamento (11%);

di cui indicativamente:

- 3 anni-uomo per il progetto;
- 2 anni-uomo per la realizzazione del prototipo ed il collaudo;
- 2,5 anni-uomo per la messa a punto (riprogettazione);
- 1,5 anni-uomo per la realizzazione finale ed il collaudo.

Componenti necessari:

- 1576 componenti COTS;
- 6 circuiti stampati;
- 40 connettori;
- 6 processori low-power;
- 6 pacchi batterie
- 5 pannelli solari;
- 2 antenne;
- 30 metri di cavi.

Costi:

- € 20.000 per i componenti, di cui:
  (a) € 4.000 per il prototipo;
  (b) € 14.000 per 4 satelliti;
  (c) € 1.000 per attrezzatura specifica;
  (d) € 1.000 per problemi di pezzatura minima;
- $ 50.000 per il lancio;
- € 10.000 per la stazione di terra;
- € 6.000 di strumentazione;
- € 5.000 mano d'opera esterna.

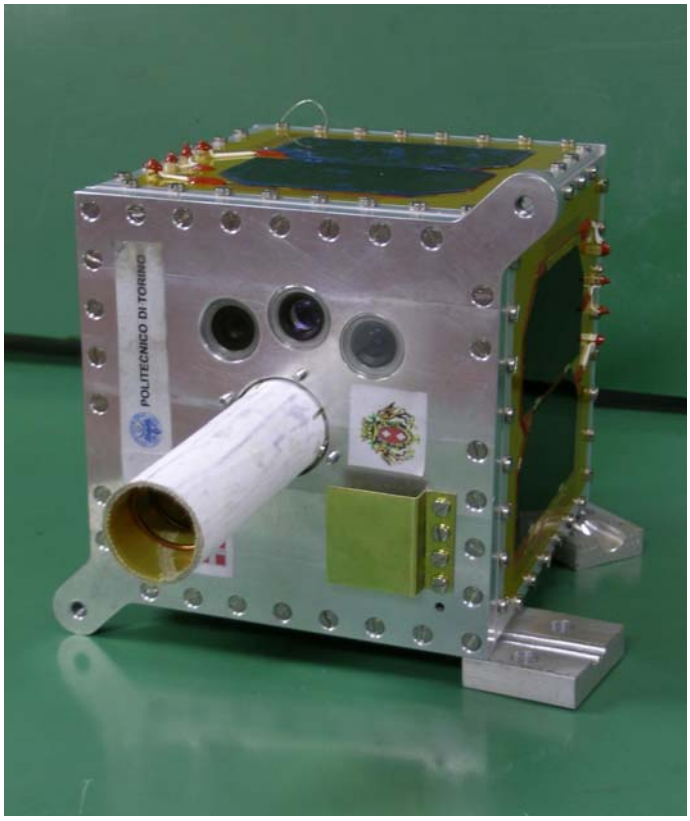


Figura 1.13 – *PiCPoT*

Capitolo 2

Introduzione agli apparati di sviluppo, collaudo ed integrazione del satellite PiCPoT

Abbiamo visto come può essere strutturato e costruito un satellite universitario, ma perché un progetto così ambizioso abbia successo, serve anche altro.

Servono infatti diversi apparati di supporto, utilizzati per costruire, sviluppare e collaudare il satellite, ma anche per renderne possibile l'installazione a bordo del vettore di lancio (questa operazione viene detta "integrazione").

L'integrazione è una procedura complessa, che prevede tempistiche ben definite e regole ferree. Chi si occupa di questa operazione è costretto a rispettarle e chi progetta il satellite deve tenerle bene in considerazione per evitare di commettere errori che possono pregiudicare il buon esito del lancio. Ad esempio, il satellite viene integrato sul lanciatore più di una settimana prima del lancio; l'ultimo contatto tra il progettista ed il satellite avviene sette giorni prima del gran giorno, dopo di che solo un rinvio del lancio stesso permetterà di lavorare nuovamente sul progetto. In fase di stesura delle specifiche si deve quindi considerare che le batterie devono rimanere cariche per almeno sette giorni, senza che nessuno possa ricaricarle.

2.1 Gli apparati

Per sviluppare, collaudare ed integrare PiCPot sono stati progettati i seguenti apparati:

- una stazione di terra fissa;
- un caricabatterie dedicato;
- una scheda di interfaccia;
- una stazione di terra portatile.

Di seguito analizzeremo brevemente questi apparati per poi illustrarli nello specifico nei capitoli successivi ad esclusione della stazione di terra fissa.

2.1.1 La stazione di terra fissa

Un problema estremamente importante legato al progetto di un satellite riguarda la progettazione e l'uso di una adeguata stazione di terra. Tale aspetto del progetto PiCPoT è tanto importante quanto lo sviluppo del satellite stesso. Dopo il lancio, infatti, il successo dell'intera missione verrà determinato dalla possibilità di ricevere ed inviare dati a PiCPoT. un progetto accurato ed affidabile della stazione di terra è quindi indispensabile per raggiungere tale obbiettivo.

Un altro punto, non meno importante, riguarda la capacità delle antenne di terra di puntare e seguire il satellite, poiché PiCPoT sarà visibile solo per pochi minuti al giorno. In figura 2.1 è rappresentato lo schema a blocchi della stazione di terra, la cui architettura può essere suddivisa nelle seguenti parti:

- antenne;
- controllo dei rotori;
- gestione della stazione di terra;
- canale TX/RX a 437 MHz;
- canale TX/RX a 2,4 GHz.

Le antenne

Le antenne hanno il compito di trasmettere verso il satellite e di riceverne le comunicazioni. Sono movimentate da due rotori.

Per la trasmissione a 437,485 MHz è stata scelta un'antenna ad elica con riflettore; la forza del vento contro il riflettore costituisce un punto critico, si è ovviato al problema realizzando il riflettore con una rete a maglie quadrate. L'antenna è costituita da un cavo in rame del diametro di 3,15 millimetri, l'elica ha un diametro di 25 centimetri ed un passo di 15 centimetri. L'ingombro dell'elica è di un metro in altezza ed il riflettore ha un diametro di un metro. Il guadagno massimo offerto dall'antenna ad elica è di circa 14 dB.

Figura 2.1 – *Le antenne ad elica ed a paraboloide*

Per il canale a 2440 MHz è stata realizzata un' antenna a paraboloide con diametro pari ad 80 centimetri e distanza focale di 25 centimetri. L'illuminatore utilizzato è un'antenna ad elica "Andrew" con polarizzazione circolare destrorsa e guadagno pari a 13 dB. Il guadagno complessivo dell'antenna risulta essere di circa 23 dB.

Controllo dei rotori

Le antenne sono fissate su una struttura meccanica composta da due rotori, uno per la rotazione azimutale e l'altro per l'elevazione, per essere sempre orientate nell'esatta direzione del satellite.

Il rotore scelto è il modello Yaesu G-5500 che consente di raggiungere 450° di rotazione azimutale e 180° di elevazione. L'orientamento delle antenne è controllato da un PC che segue l'orbita del satellite.

Gestione della stazione di terra

La gestione della stazione di terra è affidata ad un PC, dotato di tre software per le diverse funzioni.

Il primo ha il compito di comandare i rotori quando il satellite si trova all'interno della finestra di visibilità. Tale software chiama "Nova for Windows©". Questo software provvede anche a variare la frequenza del trasmettitore a 437 MHz per compensare l'effetto Doppler. Gli altri due programmi sono utilizzati per gestire le comunicazioni sui due canali di comunicazione.

Il canale TX/RX a 437 MHz

Questo canale di comunicazione è costituito da una catena di dispositivi che permettono la trasmissione dei telecomandi e la ricezione delle telemetrie.

Al fine di rendere possibile la ricezione dei dati da parte di tutti i radioamatori, oltre a comunicare su bande amatoriali si è pensato di sfruttare un protocollo già esistente ed in particolare il più diffuso, l'AX-25 (Amateur X.25).

L'apparato di trasmissione su questo canale è costituito da:

- un TNC PK-96/100 prodotto dalla Timewave;
- una radio FT-847 prodotta dalla Yaesu;

- un LNA SP-7000 prodotto dalla UKW-Berichte.

Il canale TX/RX a 2440 MHz

Il canale a 2440 MHz è stato realizzato per garantire la trasmissione e la ricezione di dati anche nel caso in cui l'altro canale abbia dei problemi di funzionamento.

La prima differenza riscontrabile rispetto all'altro canale è rappresentata dalla presenza di due catene completamente distinte per la trasmissione e la ricezione.

Il ramo di trasmissione è basato principalmente su due componenti: un transceiver CC2400 prodotto dalla Chipcon ed un amplificatore di potenza KU 2325A prodotto dalla Kuhne.

Il ramo di ricezione si basa su un LNA 2227A prodotto dalla Kuhne e su un secondo transceiver CC2400.

2.1.2 Il caricabatterie

Uno degli apparati esterni al satellite che immediatamente si è rivelato necessario è il caricabatterie dedicato.

Questo apparato è indispensabile sia durante la fase di test e collaudo di PiCPoT sia durante la procedura di integrazione. Come detto in precedenza, il satellite rimane all'interno del vettore almeno sette giorni prima di essere lanciato, durante questo lasso di tempo le batterie devono scaricarsi il meno possibile; per lo stesso motivo devono essere completamente cariche quando il satellite viene lasciato a se stesso; in più, la carica delle batterie deve avvenire nella maniera più efficiente possibile, in modo da non danneggiarle. Per questi due motivi è risultato necessario costruire un caricabatterie dedicato, che è stato poi utilizzato anche durante la fase di collaudo.

Il caricabatterie è controllato da un computer attraverso la porta USB e gestito da un programma dedicato. E' costituito da tre blocchi:

- una scheda USB-6008 prodotta da National Instruments;
- una scheda di interfaccia dedicata;
- un alimentatore switching con tensione di uscita di 17 Volt.

Tramite queste due schede il programma di controllo effettua il ciclo di carica delle sei batterie di PiCPoT (una alla volta oppure consecutivamente in modo automatico) monitorando la tensione e la corrente fornite ed effettuando una serie di controlli mirati a evitare il danneggiamento delle batterie stesse.

La connessione tra il caricabatterie ed il satellite avviene tramite il connettore di test (CT).

Figura 2.2 – *Il caricabatterie di PiCPoT*

2.1.3 La scheda di interfaccia

Questo apparato si è reso necessario a causa di un errore di progetto. Inizialmente, in verità, il progetto PiCPoT prevedeva la possibilità di monitorare l'intero sistema, ed in particolare di programmare le singole schede, attraverso il connettore di test (CT). Purtroppo i microprocessori scelti per essere installati sulle schede di PiCPoT hanno reso impossibile questa funzione.

I processori installati sono dotati di una connessione JTAG non standard, questo non consente la gestione di catene di dispositivi da programmare selezionando di volta in volta il dispositivo voluto, in altre parole essi funzionano correttamente solo se isolati da altri sistemi.

L'architettura di PicPoT prevede un pacco schede in cui i pin di programmazione sono collegati in cascata da una scheda all'altra attraverso il connettore J0; essi vengono portati attraverso la scheda ProcA al connettore di test, mediante il quale con tre opportuni pin di selezione si sarebbe potuto scegliere quale processore e su quale scheda programmare. A causa dell'incompatibilità dei processori con questa struttura non si è potuto utilizzare questo meccanismo per nessuna delle schede.

Per ovviare a questo problema vi erano tre strade.

La prima, che non avrebbe comportato modifiche al satellite, consisteva nello smontare il pacco schede ogni qualvolta fosse stata necessaria una minima modifica al software di uno dei processori, in modo da consentire la riprogrammazione della scheda "in solitudine". Questa scelta avrebbe comportato un enorme dispendio di tempo e nelle concitate fasi di collaudo finale non era assolutamente applicabile.

La seconda avrebbe comportato il rifacimento delle schede, inserendo al loro interno un modulo con una FPGA che si occupasse di standardizzare la connessione JTAG dei processori. Anche questa soluzione è stata scartata per problemi di tempo.

La terza possibilità, quella poi attuata, ha portato alla costruzione di una scheda di interfaccia. Si è provveduto a piccole modifiche sul satellite. Sono state interrotte le

connessioni in catena sul connettore J0 e si è provveduto a portare all'esterno dello stesso con l'ausilio di cavi e di connettori i pin di programmazione delle diverse schede.

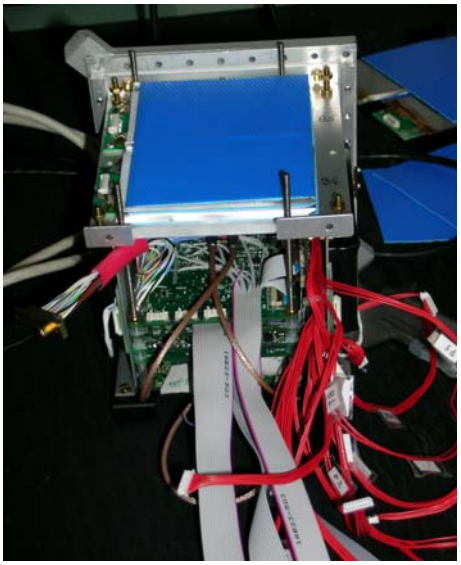


Figura 2.3 – *PiCPoT con i cavi per l'interfaccia (cavi grigi)*

La scheda di interfaccia doveva quindi occuparsi di fare da raccordo tra tutti i dispositivi di programmazione ed il satellite, fornendo anche le alimentazioni necessarie e svolgendo semplici operazioni logiche.

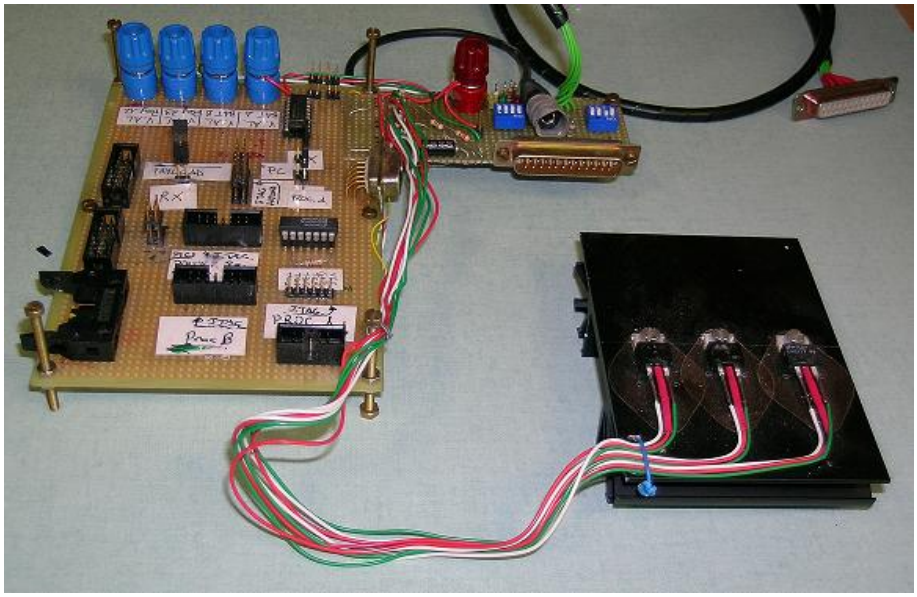


Figura 2.4 – *La scheda di interfaccia*

Vediamo ora brevemente come il connettore di test avrebbe dovuto consentire la programmazione dall'esterno. Come già accennato, il connettore di test è un connettore a vaschetta miniaturizzato a 25 poli, questa è la piedinatura:

# Pin CT	Segnale	# Pin CT	Segnale
1	Positivo B1	14	SPI/UART_MISO/RX(ProcA)
2	Positivo B2	15	SPI/UART_MOSI/TX(ProcA)
3	Positivo B3	16	SPI_SCK
4	Positivo B4	17	SPI_SEL
5	Positivo B5	18	Selettore 0
6	Positivo B6	19	Selettore 1
7	Negativo batterie	20	Selettore 2
8	JTAG TDI	21	Tensione alim. Val_Test
9	JTAG TDO	22	Comando MOS switching
10	JTAG TCK	23	Attivazione schede
11	JTAG TMS negato	24	Bit stream di TxRx
12	Segnale PAL	25	Comando MOS switching ist.
13	Massa del segnale PAL	Involucro	Massa

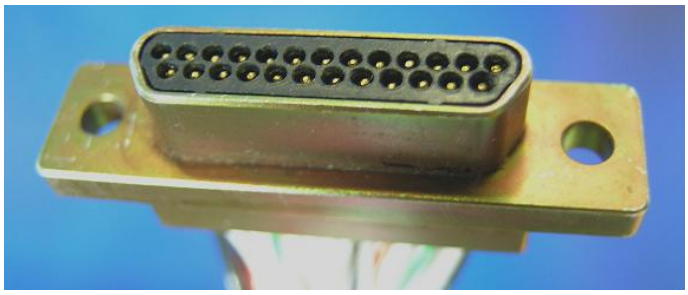


Figura 2.5 – *Il connettore di test*

Agendo sui tre pin di selezione (Selettore 0,1,2) i segnali JTAG ed SPI avrebbero dovuto essere diretti solo al dispositivo selezionato senza subire interferenze dagli altri. Questo, per i motivi spiegati in precedenza, non avveniva.

Si noti che sul connettore di test sono presenti solo i poli positivi delle singole batterie, mentre il polo negativo è comune e ciò condizionerà le scelte progettuali del caricabatterie.

2.1.4 La stazione di terra portatile

Oltre alla vera e propria stazione di terra è stato indispensabile costruire una stazione portatile. Infatti durante la fase di collaudo, ma soprattutto durante la fase di integrazione, non sarebbe stato possibile comunicare con la stazione di terra principale.

Questa stazione, di piccole dimensioni (20 x 22 x 9 centimetri), è stata portata a Baikonur (KAZ) durante l'integrazione sul lanciatore ed ha permesso di effettuare gli ultimi controlli al satellite, prima di abbandonarlo al suo destino.

La stazione di terra portatile è costituita principalmente da una scheda TxRx di PiCPoT modificata per permettere la comunicazione con un PC. Considerata la distanza ridotta a cui sarà collocata rispetto al satellite, per semplicità non si è impiegato alcun LNA od amplificatore di potenza e si è scelto anche di utilizzare un unico CC 2400.

Per semplificare ulteriormente il sistema, non è stato utilizzato alcun microcontrollore per la gestione del canale a 2,44 GHz: ciò comporta che tutte le operazioni vengano gestite tramite la porta parallela di un computer.

Questa serie di semplificazioni ha però trasferito altrove la complessità del problema, è stato infatti necessario realizzare un programma in grado di emulare il protocollo SPI per comunicare direttamente con il transceiver e ricevere o trasmettere i dati. La stesura del software si è rivelata particolarmente complessa in quanto Windows non permette una facile gestione delle applicazioni in tempo reale.

Figura 2.6 – *La stazione di terra portatile*

Capitolo 3

Il caricabatterie (hardware)

Una delle prime necessità percepite durante la fase di collaudo del satellite è stata l'aver a disposizione un dispositivo efficace di carica delle batterie.

Come vedremo in seguito, le batterie di PiCPoT, come in realtà tutte le batterie, devono essere caricate con un procedimento particolare al fine di danneggiarle il meno possibile e di prolungarne il più possibile la durata. Infatti, sul satellite non possono essere installate batterie completamente nuove, perché è necessario avere la certezza che le stesse siano esenti da difetti ed abbiano l'efficienza garantita dalle specifiche.

Sorge quindi il problema di costruire un caricabatterie da usare durante tutti i test ed i collaudi, che però sia un dispositivo autonomo, cioè che non necessiti di apparecchiature di laboratorio, ma, al più, di un personal computer oltre che di una presa di corrente. Analizziamo quindi nel dettaglio le specifiche del problema.

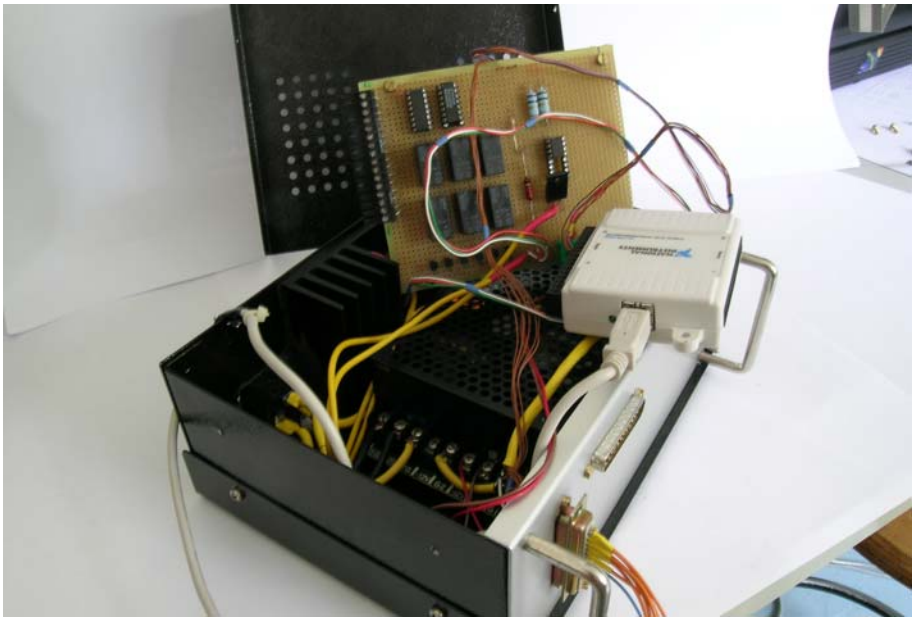


Figura 3.1 – *Hardware del caricabatterie*

3.1 Le specifiche

Le specifiche del problema sono poche, chiare, ad una prima lettura possono sembrare semplici, ma analizzandole a fondo si scopre che esse nascondono diversi problemi.

Il caricabatterie dovrà:

- interfacciarsi al satellite tramite il connettore di test; ciò significa che si avranno a disposizione i poli positivi delle sei batterie ed un polo negativo comune a tutte le batterie, coincidente con la massa del satellite;
- contenere un alimentatore al suo interno, non dovrà cioè necessitare di fonti di alimentazione esterne diverse dalla tensione di rete (220 Volt – 50 Hertz);
- caricare le batterie seguendone fedelmente le specifiche di carica dettate dal costruttore;
- avere un'interfaccia chiara e comprensibile, utilizzabile anche da persone con poche conoscenze di elettronica.

Vediamo ora quali sono le caratteristiche degli utilizzatori di questa apparecchiatura.

3.1.1 Le batterie

Le batterie sono di due tipi:

- B1, B2, B4, B5 sono costituite da due celle a litio-polimeri (Li-P) collegate in serie, per un totale di 7,2 Volt nominali, capacità di 1.500 mAh, modello ULTRALIFE Batteries UBC425085.
- B3, B6 sono costituite da sei celle al nichel-cadmio (Ni-Cd) collegate in serie, per un totale di 7,2 Volt nominali, capacità di 1.000 mAh, modello SANYO Hygh Capacity size AA.

Come spiega graficamente la figura, le batterie a litio-polimeri devono essere caricate con una procedura particolare, al fine di minimizzarne l'usura.

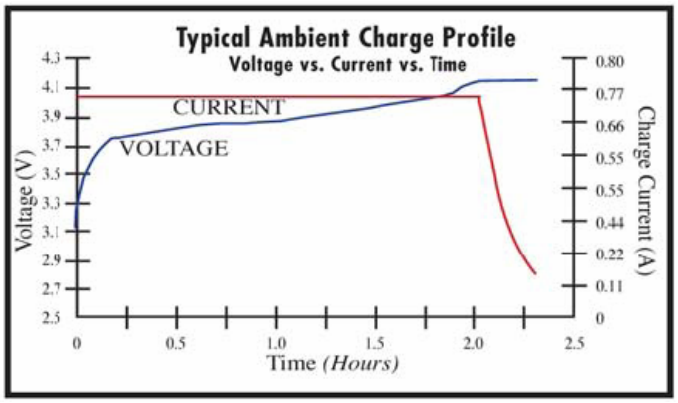


Figura 3.2 – *Diagramma della tensione e della corrente sulla batteria agli ioni di litio per un corretto ciclo di carica*

E' necessario mantenere costante la corrente fornita alla batteria (circa 800 mA) fino a quando la tensione ai capi della stessa non supera i 4,2 Volt. Avendo noi due celle in serie e procedendo alla carica della serie di due accumulatori, questa tensione di soglia risulta doppia (8,4 Volt).

Superata la soglia si deve procedere in maniera opposta, ovvero deve essere mantenuta costante la tensione fornita e deve essere libera di scendere la corrente fino ad una soglia fissata a 150mA. A questo punto la batteria può essere dichiarata carica.

La durata del ciclo supera le due ore; considerando che sul satellite vi sono 4 batterie di questo tipo vanno messe in conto quasi dieci ore per caricarle completamente (considerando di partire da batterie completamente scariche). Una carica residua parziale della batteria accorcia i tempi di carica poiché questo tipo di accumulatore non presenta "effetto memoria", quindi non si danneggia se viene caricato senza essere stato in precedenza completamente scaricato.

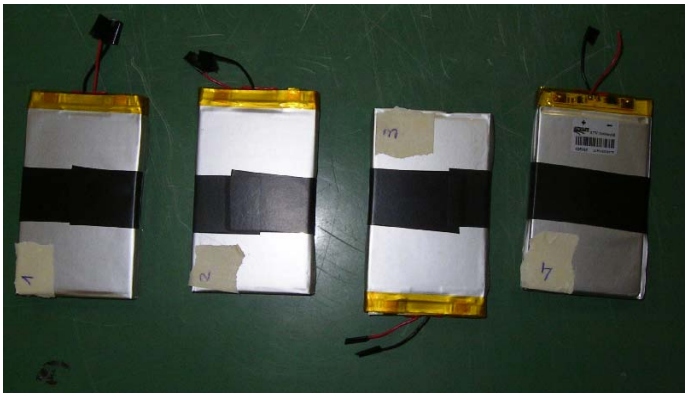


Figura 3.3 – *Batterie agli ioni di litio*

Le due batterie costituite da celle al nichel-cadmio hanno una strategia di carica differente, meno semplice da implementare perché non presenta soglie certe.

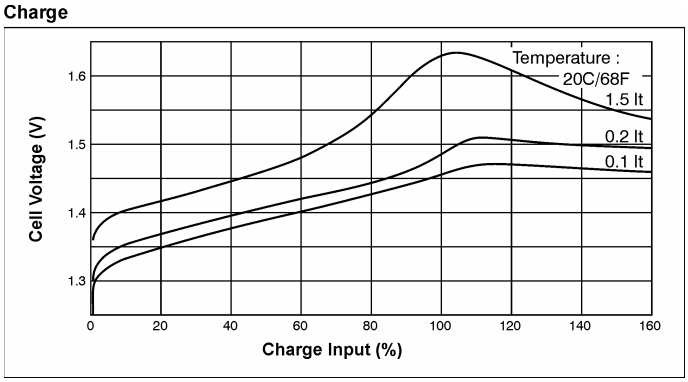


Figura 3.4 – *Diagramma della tensione sulla batteria al nichel-cadmio per un corretto ciclo di carica*

Per effettuare correttamente il ciclo di scarica è necessario prima scaricare completamente la batteria poiché questo tipo di accumulatore soffre di "effetto memoria" e perde di efficienza se ricaricato quando presenta ancora un residuo di carica. Il ciclo di scarica avverrà a corrente elevata (circa 1'100 mA pari a 10 I_t). Con I_t viene indicata la corrente tipica alla quale caricare e scaricare la batteria. Il data-sheet dell'accumulatore riporta infatti i grafici di carica e scarica per multipli della corrente tipica che risulta essere 110 mA.

La batteria viene dichiarata scarica quando la tensione di una singola cella scende al di sotto di 0,8 Volt; essendo il pacco batterie composto da 6 celle in serie, la tensione di soglia al di sotto della quale la batteria sarà considerata scarica risulta essere 4,8 Volt.

Scesi al di sotto di questa soglia (l'unica nota e certa a priori) è possibile iniziare il ciclo di carica. La strategia deve prevedere la carica a corrente costante. Si è scelto di

lavorare a 5 I_t per accelerare il processo di carica senza però ridurre l'efficienza delle batterie.

Procedendo con corrente costante si deve attendere che l'andamento della tensione, che nella prima parte del ciclo è crescente, abbia un punto di massimo; quale sia il valore di questa tensione non è noto a priori ed è diverso da esemplare ad esemplare.

Nel momento in cui si verifica l'inversione dell'andamento della tensione, si deve considerare l'energia fornita fino ad allora e continuare la carica a corrente costante fino a fornire il 40% in più di energia rispetto al valore erogato fino al raggiungimento del massimo.

Il ciclo ha una durata di circa 2 ore a cui va aggiunto il tempo di scarica, dipendente dalla condizione in cui si trova la batteria. Il tempo di scarica, con una corrente assorbita di 1.100 mA, non dovrebbe superare l'ora, poiché 1'100 mAh è appunto la capacita di queste batterie.

Poiché sul satellite sono presenti quattro batterie a litio-polimeri e due al nichel-cadmio, è possibile ipotizzare che il ciclo completo di carica possa durare da poco più di 6 ore (tutte le batterie cariche) fino a 15 ore nella condizione più sfavorevole. Occorre quindi tenere conto di questi tempi nel considerare la procedura di integrazione del satellite sul lanciatore.

Figura 3.5 – Batterie al nichel-cadmio

3.1.2 Analisi delle specifiche

Due punti delle specifiche condizionano profondamente il sistema. Le complesse strategie di carica da implementare e la necessità di fornire un'interfaccia chiara e di semplice utilizzo portano ad una scelta obbligata: diventa indispensabile avere a disposizione un computer per gestire la procedura di carica e per fare da interfaccia con l'utente. Questa scelta non rappresenta un problema, infatti un computer è comunque necessario per gestire altri apparati ausiliari al satellite e per effettuare gli ultimi test, le

ultime modifiche o, eventualmente, le correzioni di emergenza al software del satellite se in fase di integrazione dovessero esserci dei problemi di funzionamento.

Questa scelta crea un nuovo problema: quale scheda di interfaccia utilizzare per controllare il caricabatterie? La scelta è caduta su una scheda prodotta da National Instruments, controllata da computer tramite la porta USB, in particolare sulla scheda NI USB-6008.

Le altre specifiche del problema condizioneranno la realizzazione dell'interfaccia tra la scheda National ed il connettore di test.

3.2 La struttura del sistema

Analizzando il sistema sotto forma di schema a blocchi, si nota che esso è composto fondamentalmente da quattro parti (vedi figura 3.6):

- un personal computer;
- una scheda National Instruments USB-6800;
- una scheda di interfaccia tra il sistema ed il satellite;
- il connettore di test del satellite stesso.

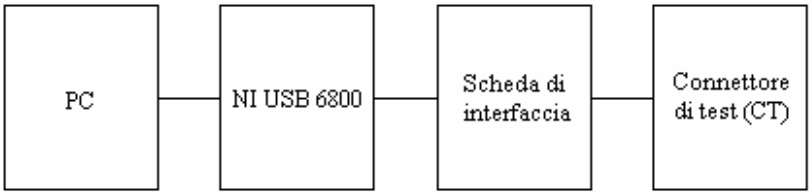


Figura 3.6 – *Schema a blocchi del sistema*

In seguito analizzeremo a fondo due blocchi di questo schema, ovvero la scheda National Instruments USB-6800 e la scheda di interfaccia.

Vediamo ora di entrare nel dettaglio della struttura del caricabatterie. Questa apparecchiatura, dovrà, sotto controllo di un computer:

- selezionare una batteria da caricare tramite una opportuna rete di selezione;
- fornire l'opportuna energia controllandola tramite la scheda NI USB-6800;
- effettuare alcune misure periodicamente (sempre attraverso la scheda NI) al fine di caricare correttamente l'accumulatore;
- decidere quando l'accumulatore sia carico;
- passare all'operazione successiva.

Da questa serie di operazioni deriva lo schema a blocchi dettagliato e definitivo del sistema.

Come possiamo vedere in figura 3.7, il personal computer agisce sulla scheda NI, la quale opera sul sistema tramite tre connessioni differenti: due di uscita (analogica e digitale) ed una analogica di ingresso.

L'uscita digitale è utilizzata per selezionare la batteria da caricare od eventualmente da scaricare. L'uscita analogica, opportunamente amplificata, fornisce la necessaria energia alla batteria scelta. L'ingresso analogico serve da retroazione al sistema al fine di renderlo stabile, infatti permette di acquisire le misure necessarie al controllo, regolando così direttamente l'energia erogate tramite l'uscita analogica.

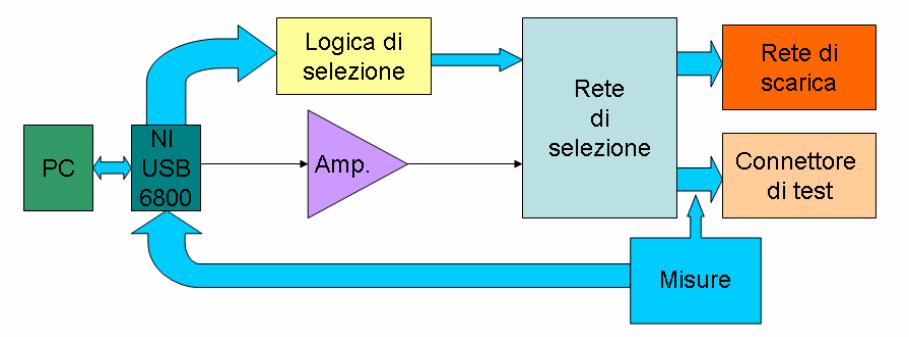


Figura 3.7 – *Schema a blocchi completo*

3.2.1 La scheda National Instruments USB-6800

Questa scheda è concepita per fornire le funzionalità base alla risoluzione del problema dell'acquisizione dati. E' adatta ad essere utilizzata come sia come data logger, sia come strumento di misura portatile, sia per esperimenti di laboratorio accademici. E' un dispositivo alla portata degli studenti, ma è sufficientemente versatile per effettuare misurazioni sofisticate.

Le peculiarità principali di questa apparecchiatura sono:

- è piccola e portatile;
- risoluzione in ingresso di 12 bit con massima velocità di acquisizione di 10 KS/s
- due connettori integrati per una maggiore facilità di interconnessione con i propri sistemi;
- due convertitori digitale/analogico che forniscono segnali di uscita accurati;
- 12 linee digitali di ingresso/uscita;
- può essere facilmente controllata col software in dotazione, oppure con la libreria dedicata per il linguaggio C;
- tramite il software LabWindows/CVI consente di avere un'interfaccia utente semplice da utilizzare.

Per ulteriori approfondimenti sulle specifiche tecniche della scheda si rimanda al foglio tecnico allegato in appendice.

Figura 3.8 – *Scheda National Instruments USB-6800*

Per realizzare il caricabatterie sono stati utilizzati due ingressi analogici differenziali (per un totale di quattro pin sul connettore della scheda NI) per effettuare le misure di controllo, una uscita analogica (opportunamente amplificata) per fornire l'energia in fase di carica e cinque uscite digitali per pilotare la rete di selezione della batteria e la rete di scarica.

3.2.2 La scheda di interfaccia

Questa scheda costituisce la parte fondamentale da progettare (unitamente al software di cui parleremo nel prossimo capitolo) al fine di ottenere un sistema performante.
E' costituita da diverse parti, più o meno interconnesse tra loro:

- amplificatore del segnale di carica;
- rete di selezione;
- rete di scarica;
- rete di misura in tempo reale.

Nel prossimo paragrafo analizzeremo separatamente questi sottosistemi, illustrando le problematiche di progetto che hanno condizionato la scelta dei componenti e degli schemi circuitali.

3.3 I sottosistemi di interfaccia

3.3.1 L'amplificatore del segnale di carica

Uno dei primi problemi incontrati è stata la necessità di amplificare il segnale analogico fornito dalla scheda National Instruments.

La massima corrente erogata dalla scheda attraverso le sue uscite analogiche è di 5 mA, mentre la corrente necessaria per caricare le batterie è dell'ordine dell'Ampere. In più, la massima tensione fornita è di 5 Volt, mentre il ciclo di carica necessita di tensioni anche superiori ai 10 Volt. L'uscita utilizzata è indicata sulla scheda NI con il codice AO0 e nello schema circuitale è chiamata $V_{controllo}$.

Si è deciso quindi di utilizzare un amplificatore operazionale (LM358) in configurazione non invertente, con guadagno circa 2,2, al quale collegare in cascata un amplificatore Darlington di potenza (Motorola JE 802), montato su dissipatore. In figura 3.9 è illustrato lo schema elettrico utilizzato. Questo circuito permetterà di fornire tensioni di carica comprese tra 0 Volt e 11 Volt circa e corrente massima di 4 Ampere col dispositivo montato su dissipatore.

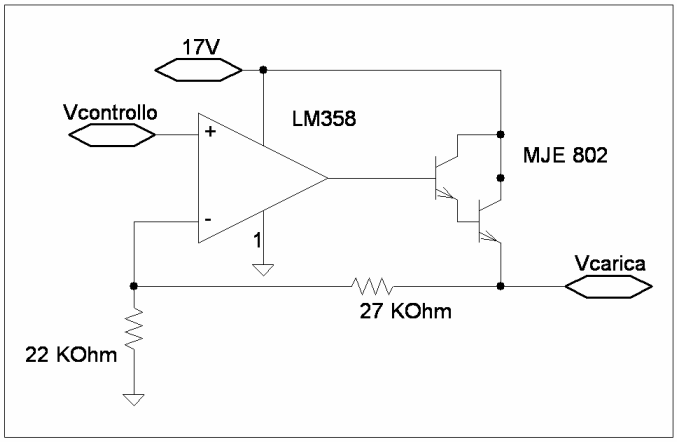


Figura 3.9 – *Schema del circuito di amplificazione del segnale di controllo*

3.3.2 La rete di selezione

La rete di selezione si deve occupare di connettere una ed una sola batteria all'uscita dell'amplificatore Darlington descritto in precedenza. Deve anche consentire di collegare una sola batteria alla rete di scarica.

Sono quindi necessari sei interruttori per collegare ogni batteria all'amplificatore. Questi sei interruttori devono essere controllati dal computer attraverso la scheda NI.

E’ necessario un ulteriore interruttore per collegare la batteria selezionata alla rete di scarica.

Per evitare di connettere tra loro due batterie, oppure di collegare la rete di scarica mentre si sta ancora caricando un accumulatore, si è deciso di utilizzare cinque bit in uscita dalla scheda NI, indirizzando quattro di questi ad un decodificatore digitale che si occupa di attivare una ed una sola delle sei batterie, mentre il quinto bit comanda direttamente la rete di scarica. Le cinque uscite digitali utilizzate sono indicate con P0.0÷P.0.4.

Il circuito di decodifica utilizzato è un doppio decoder 2 bit / 4 uscite, denominato 74HC139, scelto perché reperibile in tempi brevi.

Questo circuito viene pilotato collegando due dei segnali digitali ad entrambi i decodificatori, ovvero P0.0 ad A_0 e P0.1 ad A_1, ed utilizzando gli altri due segnali come “enable” del dispositivo, collegando cioè P0.2 ad EN1 e P0.3 ad EN2. I pin di “enable” sono negati all’interno del decodificatore.

Il quinto segnale di controllo, P0.4, è collegato direttamente al transistore PMOS che connette la rete di scarica.

In figura 3.10 sono evidenziati i collegamenti e le uscite del decodificatore che piloteranno la rete di selezione.

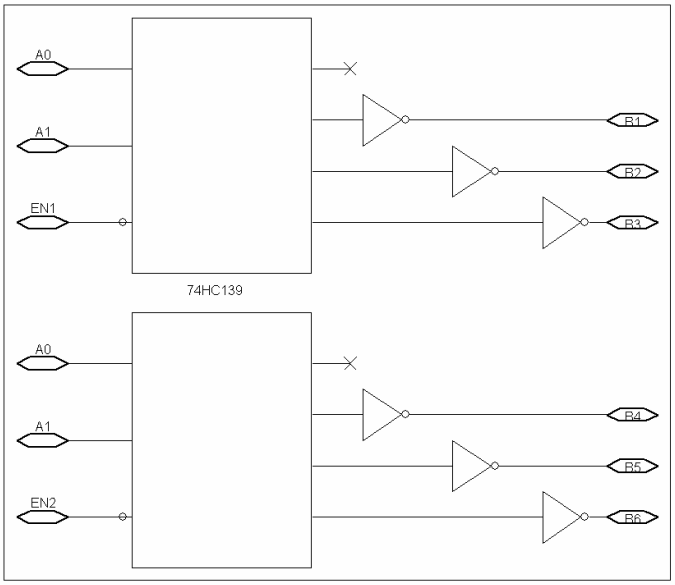


Figura 3.10 – *Schema di collegamento del decoder digitale 74HC139*

Analizziamo ora la tabella di verità del sistema, illustrando nel dettaglio ciò che accade al variare degli ingressi del sistema, ovvero dei pin P0.0÷P.0.4.

Tabella 3.1 - *Tabella di verità*

#	P	P	P	P	P	Uscite	Comportamento del

	0.4	0.3	0.2	0.1	0.0	attive	sistema
0	L	L	L	L	L	Nessuna	Azzeramento
1	L	L	L	L	H	B1-B4	Combinazione vietata
2	L	L	L	H	L	B2-B5	Combinazione vietata
3	L	L	L	H	H	B3-B6	Combinazione vietata
4	L	L	H	L	L	Nessuna	Azzeramento
5	L	L	H	L	H	B4	Carica Batteria # 4
6	L	L	H	H	L	B5	Carica Batteria # 5
7	L	L	H	H	H	B6	Carica Batteria # 6
8	L	H	L	L	L	Nessuna	Azzeramento
9	L	H	L	L	H	B1	Carica Batteria # 1
10	L	H	L	H	L	B2	Carica Batteria # 2
11	L	H	L	H	H	B3	Carica Batteria # 3
12	L	H	H	L	L	Dec.Dis.	Il decodificatore non è abilitato
13	L	H	H	L	H	Dec.Dis.	Il decodificatore non è abilitato
14	L	H	H	H	L	Dec.Dis.	Il decodificatore non è abilitato
15	L	H	H	H	H	Dec.Dis.	Il decodificatore non è abilitato
16	H	L	L	L	L	Nessuna	Azzeramento
17	H	L	L	L	H	B1-B4	Combinazione vietata
18	H	L	L	H	L	B2-B5	Combinazione vietata
19	H	L	L	H	H	B3-B6	Combinazione vietata
20	H	L	H	L	L	Nessuna	Azzeramento
21	H	L	H	L	H	B4	Scarica Batteria #4
22	H	L	H	H	L	B5	Scarica Batteria #5
23	H	L	H	H	H	B6	Scarica Batteria #6
24	H	H	L	L	L	Nessuna	Azzeramento
25	H	H	L	L	H	B1	Scarica Batteria #1
26	H	H	L	H	L	B2	Scarica Batteria #2
27	H	H	L	H	H	B3	Scarica Batteria #3
28	H	H	H	L	L	Dec.Dis.	Il decodificatore non è abilitato
29	H	H	H	L	H	Dec.Dis.	Il decodificatore non è abilitato
30	H	H	H	H	L	Dec.Dis.	Il decodificatore non è abilitato
31	H	H	H	H	H	Dec.Dis.	Il decodificatore non è abilitato

Tutte le combinazioni contrassegnate da “Azzeramento” sconnettono tutte le batterie dal circuito di carica/scarica: in pratica tutti gli interruttori sono aperti. La prima di queste combinazioni, la #0, verrà utilizzata come vero e proprio comando di azzeramento.

Le combinazioni indicate come vietate, lo sono perché connettono in contemporanea due batterie, queste vengono quindi collegate in parallelo e se hanno tensione diversa questa operazione arreca loro gravi danni.

Se P0.3 e P0.2 sono contemporaneamente sul livello logico H (high, alto), entrambi i decodificatori vengono disattivati, la rete di selezione non è controllata e potrebbe posizionarsi in uno stato casuale, per questo è utile avere un comando di azzeramento da fornire al sistema prima di iniziare ogni operazione.

Infine, le combinazioni che connettono una sola batteria lo fanno collegandola alla rete di carica oppure alla rete di scarica, evitando la possibilità di caricare e scaricare contemporaneamente una batteria.

Sarà compito del software di gestione comunicare alla scheda NI un numero compreso tra zero e trentuno da mandare in uscita, impostando quindi la rete di selezione nella configurazione desiderata.

Vediamo ora come è stata realizzata la rete di selezione.

Inizialmente si è deciso di realizzare l'interruttore con un transistore PMOS con caratteristiche di robustezza utili a sopportare la forte corrente di carica (si veda in figura 3.11 lo schema circuitale).

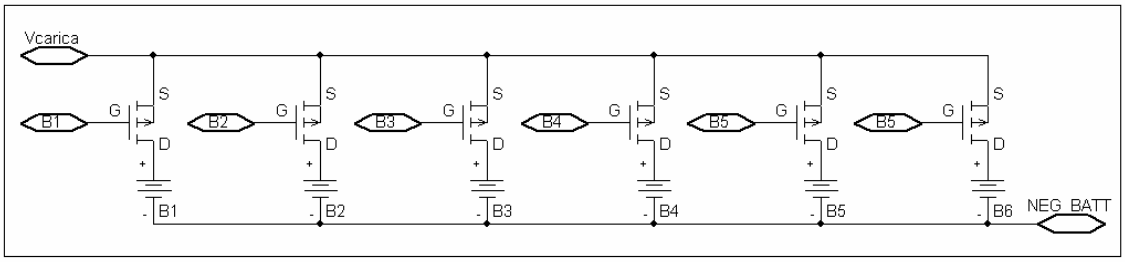


Figura 3.11 – *Schema circuitale della rete di selezione realizzata con i soli transistori P-MOS*

Un'attenta analisi dello schema di collegamento ha poi rivelato che questa soluzione non era utilizzabile. Infatti, poiché non è nota a priori la tensione di ogni accumulatore è possibile che durante la carica di una batteria, una delle altre cinque fornisca al circuito una tensione superiore a quella della batteria in carica. Questa combinazione porterebbe ad un funzionamento non voluto e potenzialmente dannoso del circuito. Facciamo un esempio.

Ipotizziamo di voler caricare B1, il PMOS1 conduce, mentre tutti gli altri sono in interdizione. Ipotizziamo ora che $V_{B2} > V_{B1}$ al punto da avere $V_{D2} > V_{S2.}$ Con queste condizioni il diodo di sub-strato parassita di PMOS2 entra in conduzione connettendo al circuito di carica anche la batteria B2. Poiché le due batterie hanno tensioni molto diverse, questa connessione non voluta può causare danni irreparabili al circuito.

Per risolvere questo problema si è deciso di ricorrere a dei relè, comandando il contatto di eccitazione con un transistore NMOS, tenuto conto che non è più necessario utilizzare transistori NMOS particolarmente robusti. In figura 3.12 è illustrato lo schema elettrico utilizzato nelle versione finale del caricabatterie.

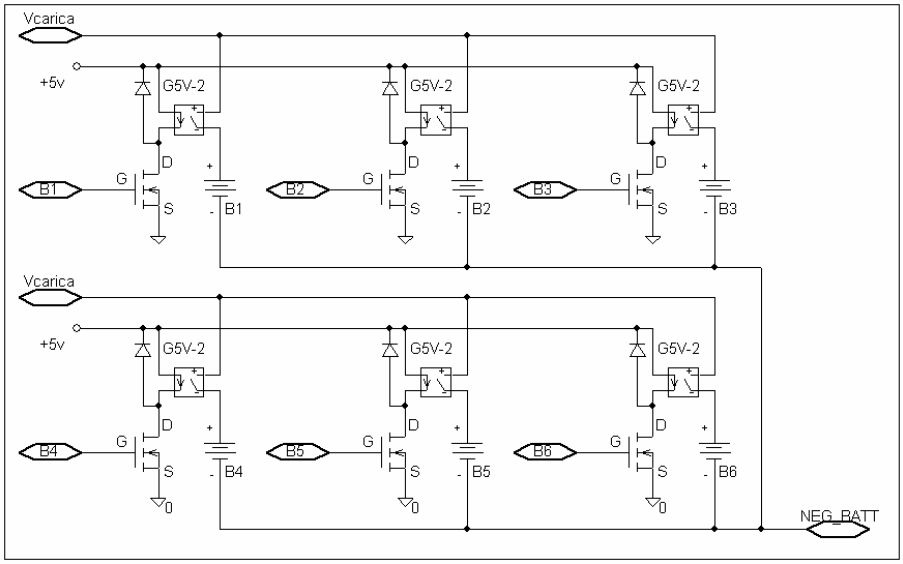


Figura 3.12 - *Schema circuitale della rete di selezione realizzata con i relè oltre che con i transistori NMOS*

Fornendo una tensione positiva al gate di uno dei NMOS, questo entra in conduzione, la bobina di eccitazione del relativo relè si carica e fa chiudere il contatto del relè. Si è deciso di collegare un diodo in parallelo alle bobine dei relè per proteggere il circuito dalle sovratensioni in fase di rilascio.

Le specifiche del sistema si sono rivelate molto restrittive al momento della scelta dei relè da utilizzare. Infatti, l'alta corrente da pilotare (circa un Ampere) costringe a scegliere relè con una resistenza di contatto molto bassa, in modo da minimizzare la caduta di tensione ai capi del contatto, poiché il contatto è in serie alla batteria da caricare e la caduta ai suoi capi comporta una tensione minore fornita alla batteria. Tenendo sempre in considerazione una delle specifiche fondamentali del progetto PiCPoT, ovvero l'uso di componentistica a basso costo, la scelta è caduta sul relè G5V2 prodotto da OMRON. I transistori NMOS utilizzati sono dei BS170 ed i diodi degli 1N4148.

Per le specifiche tecniche dettagliate dei componenti principali si rimanda ai fogli tecnici allegati in appendice.

3.3.3 La rete di scarica

La rete di scarica è in realtà molto semplice: essa è infatti costituita da un transistore NMOS che funge da interruttore di connessione tra la rete di selezione delle batterie ed un resistore di scarica.

Poiché il transistore NMOS scelto deve sopportare una corrente di più di un Ampere, si è scelto un IRFI 520 N, al cui gate è direttamente connesso l'output digitale P0.4 della scheda NI. Un segnale con valore logico alto, porta a conduzione il transistore collegando in serie alla batteria un resistore da 6,8 Ohm.

Questo resistore deve dissipare una potenza di circa 10 Watt, si è quindi deciso di utilizzare un componente adatto a sopportare una tale dissipazione. In più, avendo a disposizione il dissipatore utilizzato per l'amplificatore Darlington, si è pensato di installarlo sullo stesso al fine di facilitare lo smaltimento del calore.

Lo schema in figura mostra la rete di scarica collegata alla rete di selezione. Imponendo tramite il software di controllo che V_{carica} sia nulla, selezionando la batteria voluta ed attivando il transistore NMOS di scarica, avviene la scarica dell'accumulatore.

Il transistore NMOS utilizzato è un IRFI520N adatto a sopportare la forte corrente di scarica.

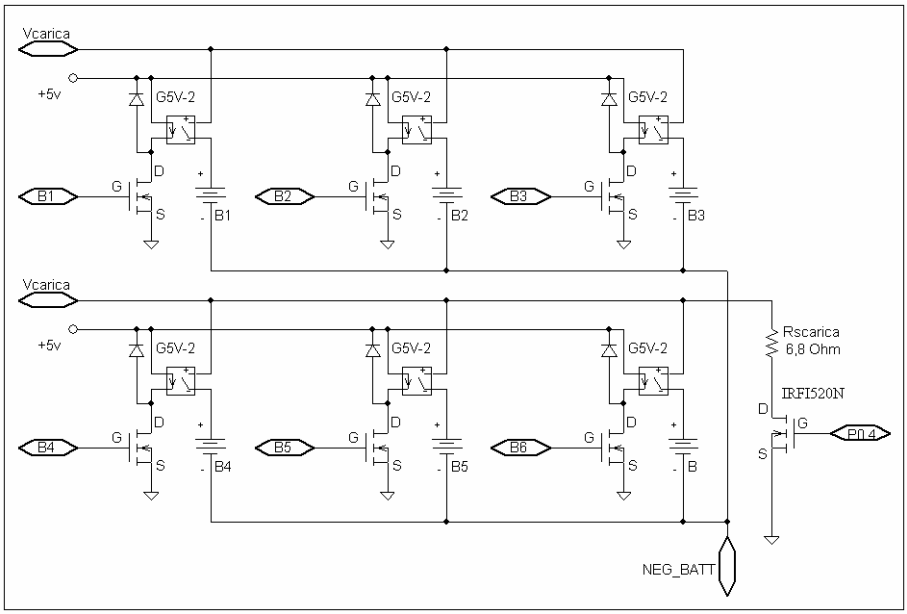


Figura 3.13 – *Circuito di selezione con annessa rete di scarica*

3.3.4 Le misure in tempo reale

Le specifiche del problema impongono un continuo controllo dello stato della batteria durante le fasi di carica e di scarica.

In particolare, durante il ciclo completo è necessario monitorare la tensione ai capi della batteria e la corrente erogata verso la stessa, mentre durante la scarica è sufficiente controllare la tensione ai capi dell'accumulatore.

La scheda National Instruments utilizzata dispone di quattro ingressi in modalità differenziale, che permettono di misurare quattro tensioni ai capi di qualunque punto del circuito. Questo non ci permette di misurare la tensione di ogni batteria - servirebbero infatti sei ingressi - dobbiamo perciò misurare la tensione tra il punto del circuito contrassegnato da V_{carica} ed il polo negativo comune a tutte le batterie.

Per misurare la corrente risulta invece necessario introdurre una resistenza di "sense", collegata tra il polo negativo comune a tutte le batterie e la massa della scheda di interfaccia. In questo resistore, molto piccolo (0,496 Ohm), scorre la stessa corrente che scorre nella batteria. Poiché la resistenza è nota, misurando la tensione ai suoi capi, siamo in grado di calcolare la corrente, effettuandone quindi una misura indiretta.

Gli ingressi della scheda NI utilizzati sono quindi due, ovvero AI0 ed AI1, ed essendo ingressi differenziali necessitano di due connessioni, chiamate AI0-, AI0+ e AI1- ed AI1+.

La misura della tensione risulta influenzata da diversi errori dovuti ai componenti in serie alla batteria. Infatti con una corrente dell'ordine dell'Ampere diventano tutt'altro che trascurabili le seguenti tensioni:

- caduta sul contatto del relè;
- cadute sui cavi che portano i poli della batteria al connettore di test;
- caduta sul cavo che connette il caricabatterie al satellite;
- cadute interne al satellite stesso.

Per questo motivo il software di gestione adotterà una procedura ad hoc per tenere conto di questo errore di misurazione intrinseco, non dovuto alla precisione dello strumento utilizzato. In figura è illustrato lo schema della rete di selezione, con la resistenza di "sense" ed indicati i punti di misurazione.

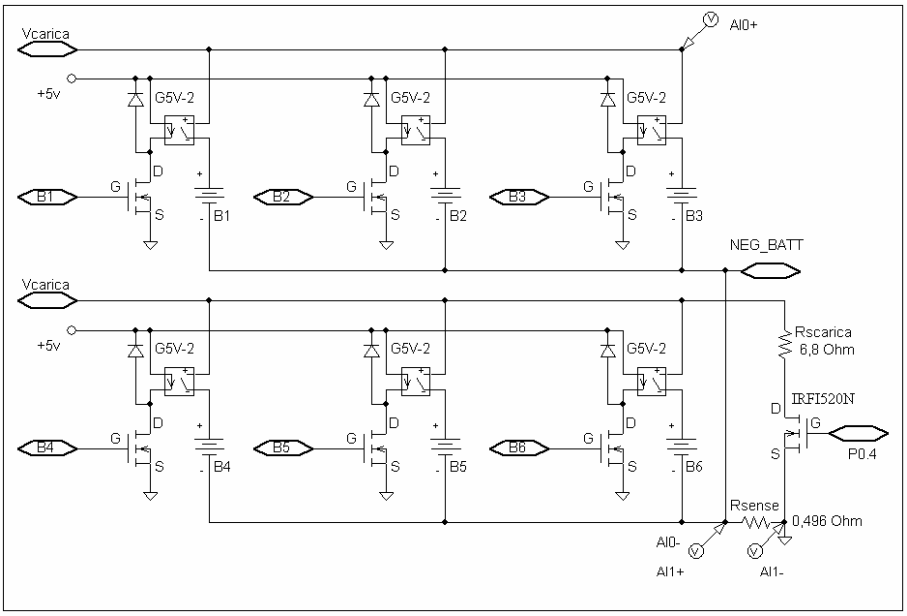


Figura 3.14 – *Rete di selezione con resistore di "sense" e punti di misura*

3.3.5 Lo schema elettrico completo

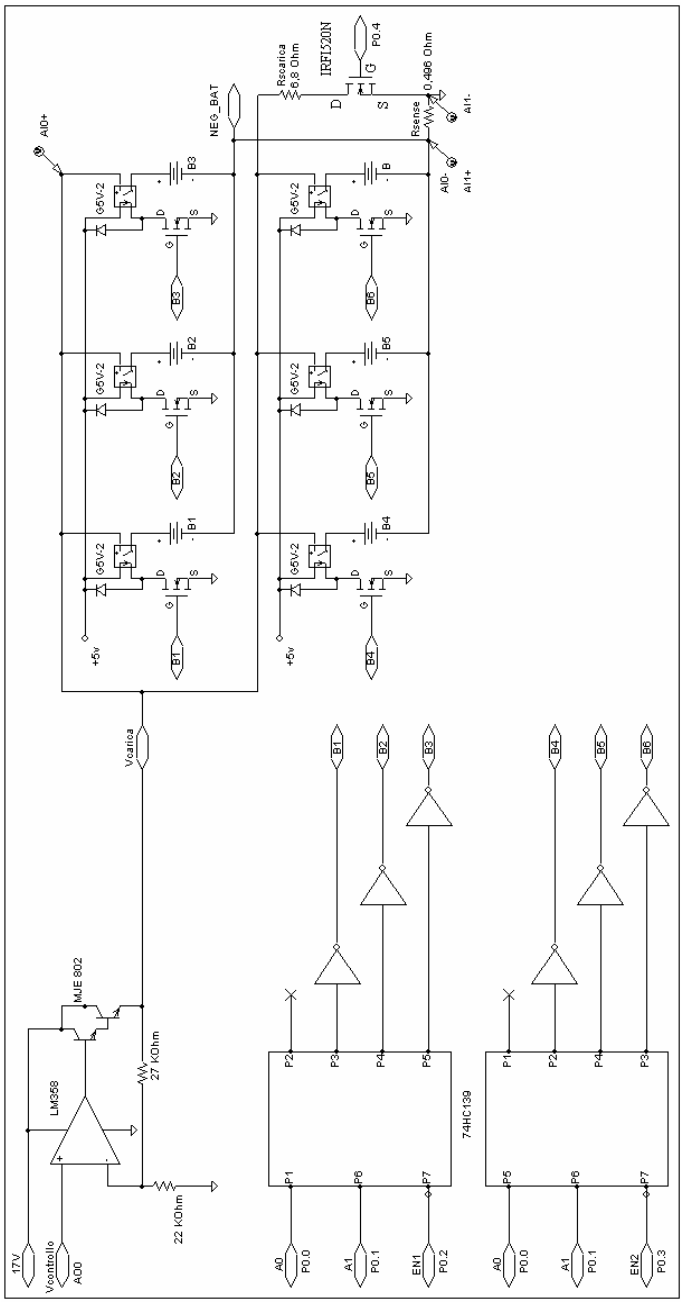


Figura 3.15 – *Schema elettrico completo*

Capitolo 4

Il caricabatterie (software)

La scelta di usare un personal computer ed una scheda di interfaccia come controllori del sistema porta a elaborare un programma di gestione che consenta di effettuare facilmente tutte le operazioni richieste anche da un operatore non esperto.

La scheda National Instruments utilizzata può essere controllata in due modi: attraverso un software dedicato (fornito con la scheda) - ma ciò limita notevolmente le potenzialità della scheda - oppure scrivendo un programma in linguaggio C ed utilizzando un'apposita libreria denominata NIDAQ.

Dopo aver optato per questa seconda possibilità, si è deciso di utilizzare anche un ambiente di programmazione che fornisse un'interfaccia grafica di facile implementazione ed uso.

La scelta è caduta su LabWindows/CVI, ambiente di programmazione del linguaggio ANSI C, creato appositamente dalla National Instruments per gestire a livello avanzato le proprie schede di acquisizione dati.

4.1 LabWindows/CVI

LabWindows/CVI di National Instruments è un ambiente di sviluppo per il linguaggio ANSI C, testato e realizzato per effettuare controlli e misure in maniera automatica anche al fine di far risparmiare molto tempo a ricercatori ed ingegneri. Questo ambiente di lavoro viene utilizzato nel settore della ricerca per sviluppare applicazioni stabili e ad

alto livello di prestazione per il settore industriale, aerospaziale, militare, delle telecomunicazioni e delle verifiche di progetto. LabWindows/CVI ottimizza lo sviluppo in queste aree attraverso procedure guidate di configurazione hardware, sistemi di debugging completi e capacità di esecuzione interattiva che aiutano ricercatori e tecnici nelle attività di competenza. Gli interessati possono rapidamente realizzare complesse applicazioni utilizzando le librerie incluse. Attraverso la flessibilità di LabWindows/CVI, i programmatori possono proteggere le loro applicazioni dall'uso senza licenza, creando pacchetti utilizzabili coi i principali sistemi operativi.

4.1.1 La struttura dell'ambiente LabWindows/CVI

Questo ambiente di programmazione permette di iniziare a lavorare al progetto partendo dal risultato finale. Infatti si parte dalla creazione del "pannello di controllo" del sistema, sistemando i pulsanti che si vorranno utilizzare, gli indicatori sui quali si vorranno visualizzare dati interessanti, eventuali grafici e tutto ciò che l'utente finale avrà a disposizione quando potrà usufruire del software finito.

Al riguardo può essere utile rappresentare un esempio pratico. Si supponga di voler misurare il valore istantaneo di una tensione che varia fra 0 Volt e 5 Volt, tracciandone l'andamento nel tempo ed avendo a disposizione un pulsante di inizio delle misure, uno di fine ed uno per uscire dall'applicazione: in figura 4.1 è rappresentato il pannello dei comandi che verrà creato.

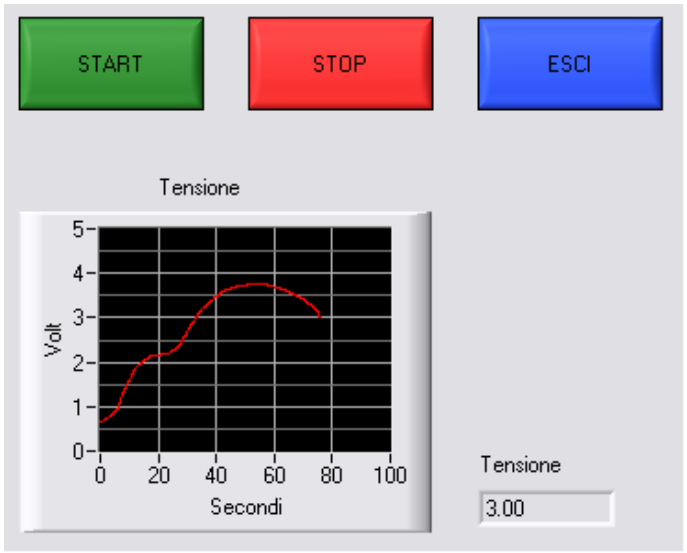


Figura 4.1 – *Esempio di pannello dei comandi realizzato con LabWindows*

Dopo aver creato questo pannello dei comandi, il cui utilizzo risulta intuitivo, all'interno dell'ambiante LabWindows è possibile "compilarlo". Questa operazione

genera un listato ANSI C contenente delle funzioni predefinite, corrispondenti ai pulsanti posizionati sul pannello comandi.

Si avrà quindi a disposizione una funzione `START`, una funzione `STOP` ed una funzione `ESCI`. Sarà a questo punto sufficiente scrivere all'interno della funzione il codice che si vuole eseguire alla pressione del pulsante.

Quali operazioni deve compiere questo semplice strumento? Deve iniziare a misurare la tensione desiderata alla pressione del pulsante START, deve interrompere la misurazione alla pressione di STOP e, mentre effettua la misurazione, deve tracciarne l'andamento ed indicarne il valore istantaneo. Alla fine delle misurazioni sarà sufficiente premere il pulsante ESCI per terminare l'applicazione.

I pulsanti START e STOP sono pulsanti normali, mentre il pulsante ESCI viene definito in partenza come pulsante di uscita. Questa operazione di definizione permette di far creare al compilatore, in automatico, le istruzioni necessarie all'uscita. La funzione `ESCI` così creata avrà al suo interno tutta la serie di comandi necessari a terminare l'applicazione.

Per effettuare queste operazioni è necessaria una funzione particolare, la funzione `Timer.` Questa funzione viene generata dal compilatore se nel pannello di comando viene inserito anche l'indicatore Timer. Eccola nel dettaglio.

4.1.2 La funzione `Timer`

Inserendo questo blocco sul pannello di comando, all'atto della compilazione verrà creata una funzione Timer. Questa funzione sarà il vero cuore del nostro apparato di misura.

La funzione `Timer` è una funzione periodica, ovvero tale che quando viene attivata viene eseguita ciclicamente fino a quando un agente esterno non la disattiva.

Il periodo viene impostato dal programmatore, compatibilmente con le specifiche del problema e con i tempi di esecuzione delle operazioni necessarie.

Per meglio illustrare il funzionamento di questa funzione, è opportuno tornare all'esempio fatto nel precedente paragrafo.

Il pulsante START si dovrà occupare di avviare la funzione `Timer` mentre il pulsante STOP dovrà interromperla. Tutte le altre operazioni si svolgeranno all'interno della funzione `Timer`, perché devono essere svolte periodicamente. Anche la funzione `ESCI` conterrà il blocco del Timer per terminare tutti i processi prima di uscire.

In figura 4.2 sono riportati i diagrammi di flusso delle funzioni.

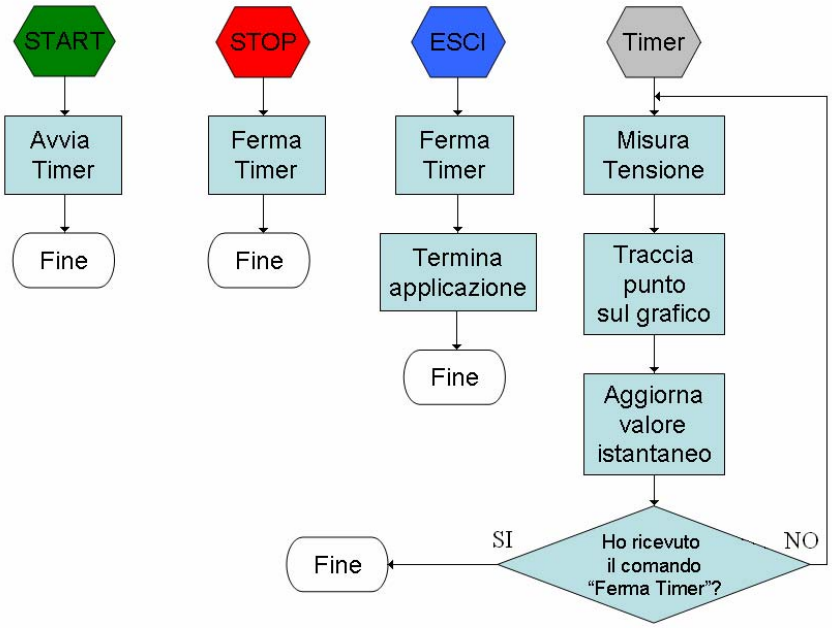


Figura 4.2 – *Diagrammi di flusso delle funzioni illustrate nell'esempio*

Come si evince dal diagramma di flusso, la funzione `Timer` si occuperà di misurare la tensione, tracciarne il valore sul grafico ed aggiornarne il valore istantaneo. A questo punto se un'altra funzione avrà provveduto nel frattempo a fermarne l'esecuzione, la funzione `Timer` si arresterà, altrimenti ripeterà le operazioni periodicamente.

Il blocco Timer è visibile solo al programmatore in fase di programmazione e di debug,ma all'utente durante il normale funzionamento. Per questo motivo in figura 4.1 non è rappresentato.

4.2 Le specifiche del software

Come tutti i problemi, anche quelli risolti mediante un programma hanno delle specifiche. In questo caso esse sono dettate sia dall'implementazione circuitale costruita, sia dalla componentistica del satellite, sia da come si vuole impostare l'interfaccia con l'utente.

Vediamo nel dettaglio le richieste:

- il programma deve caricare le batterie seguendo il più fedelmente possibile le specifiche delle stesse;
- deve essere possibile caricare una singola batteria;

- deve essere possibile caricare tutte le batterie in maniera autonoma, una dopo l'altra;
- l'interfaccia grafica deve mostrare una traccia delle operazioni compiute;
- qualunque processo deve poter essere interrotto e ripreso dal punto di interruzione;
- l'interfaccia grafica deve consentire l'uso dell'apparecchiatura anche a persone non esperte.

4.2.1 L'analisi delle specifiche

Analizzando nel dettaglio le richieste del problema si può formulare una prima ipotesi sull'aspetto del pannello di controllo dell'apparato.

Deve essere semplice ed intuitivo, quindi le azioni svolte dai vari comandi dovranno essere ben chiare. In particolare devono essere presenti comandi che permettano di caricare una singola batteria ed un comando che permetta di avviare un ciclo di carica completo di tutte le batterie del satellite.

Sono inoltre necessari due indicatori che mostrino l'andamento della tensione e della corrente fornite alle batterie nel tempo.

Fondamentale è la corretta carica degli accumulatori al fine di danneggiarli il meno possibile. E' appena il caso di aggiungere che devono essere pienamente rispettate le specifiche descritte nel precedente capitolo.

Poiché la prima fase nell'elaborazione del programma è la creazione del pannello di comando, in figura 4.3 è illustrato l'aspetto finale del quadro comandi. In seguito ne sarà illustrato l'uso.

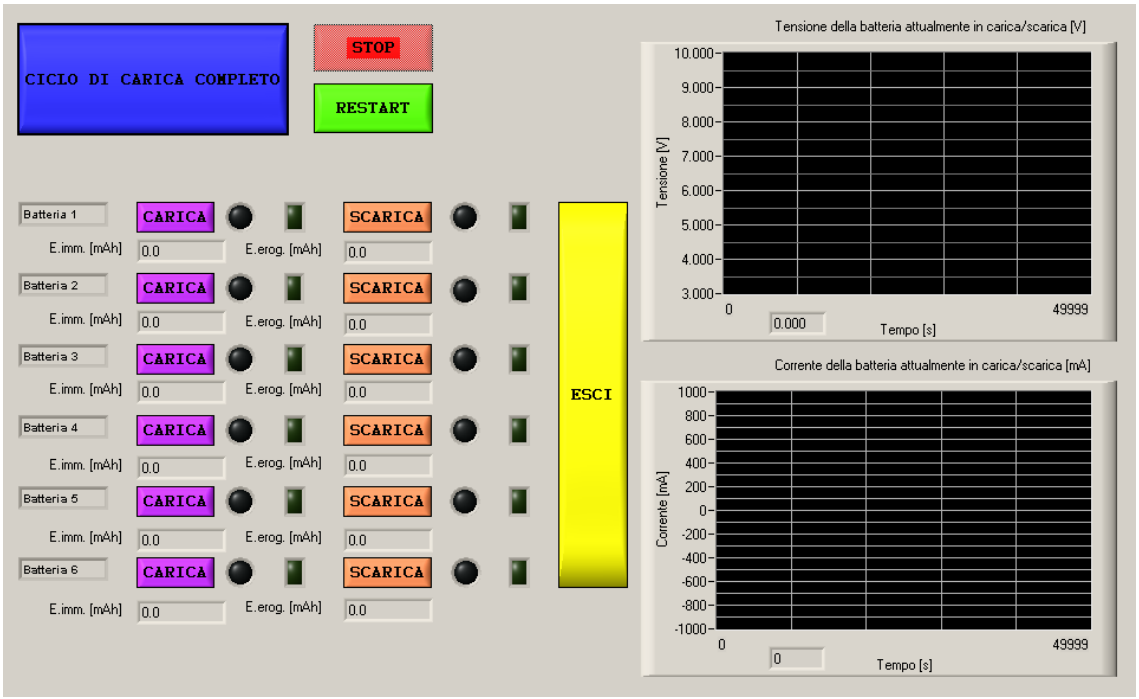


Figura 4.3 – *Pannello dei comandi*

4.3 La struttura delle funzioni implementate

Come illustrato in precedenza, ognuno dei pulsanti presenti sul pannello di interfaccia consente di svolgere una determinata funzione una volta premuto. Ecco ora nel dettaglio come sono strutturate dal punto di vista logico le principali operazioni svolte dal programma.

4.3.1 Carica a corrente costante per entrambi i tipi di batteria

Entrambe le tipologie di accumulatori da caricare prevedono, almeno in parte, un ciclo di carica a corrente costante. Ciò significa che è necessario controllare la carica attraverso la tensione $V_{controllo}$, misurare la corrente attraverso la resistenza di "sense" ed in funzione della misura ottenuta variare la tensione di controllo. Se la corrente sarà troppo alta la tensione dovrà diminuire e viceversa.

Il controllo viene effettuato ogni 100ms. Ogni volta la tensione viene incrementata o decrementata di 1,25mV, a seconda che la corrente sia minore o maggiore del valore voluto. La variazione di 1,25mV è la risoluzione massima fornita dalla scheda NI.

Prove sperimentali hanno dimostrato che questa combinazione di valori consente di giungere a buoni risultati per quanto riguarda la corretta esecuzione del ciclo di carica.

La scheda NI consentirebbe di attuare il controllo anche ogni millesimo di secondo, ma la costante di tempo del nostro sistema è tale da rendere inutile un simile livello di precisione.

In figura 4.4 è illustrato il diagramma di flusso della procedura di carica a corrente costante.

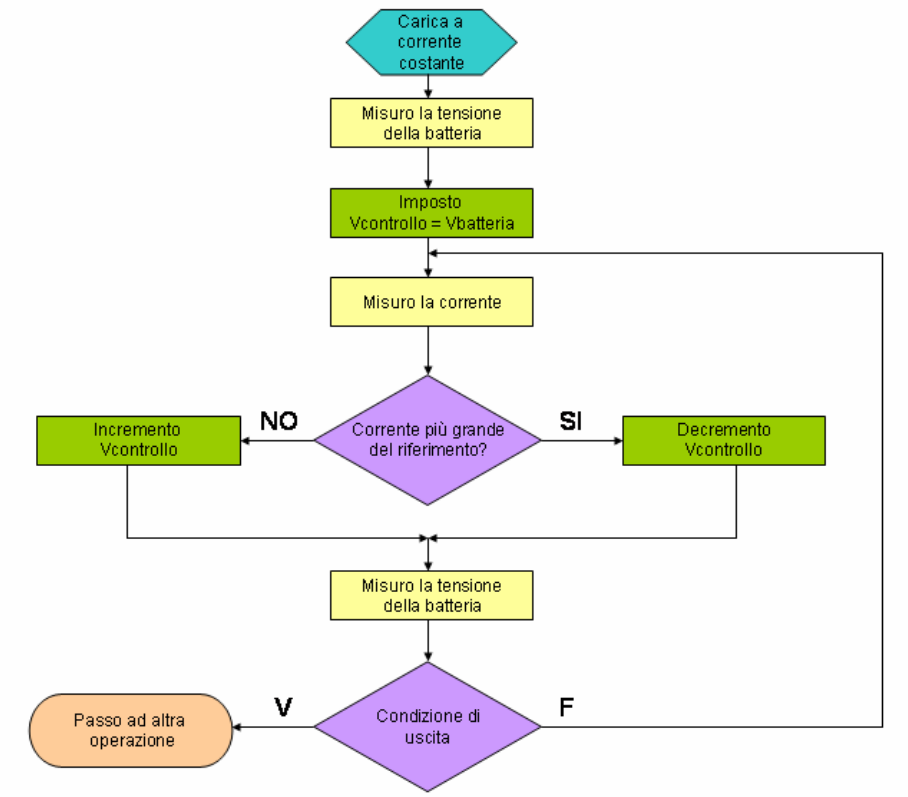


Figura 4.4 – *Diagramma di flusso del ciclo di carica a corrente costante*

Analizziamone ora il funzionamento. In primo luogo si misura la tensione della batteria, questo permette di avere una stima del punto da cui partire nel ciclo di controllo. Si imposta $V_{controllo}$ pari a $V_{batteria}$ e si inizia a monitorare il sistema.

A seconda del valore della corrente misurato, si incrementa o decrementa $V_{controllo}$. Dopo di che si controlla che non sia verificata la condizione di uscita: quando questa è verificata si esce dalla procedura, altrimenti si torna a misurare la corrente e così via.

La condizione di uscita dipende dal tipo di batteria che si sta caricando. Per le batterie agli ioni di litio il ciclo di carica a corrente costante termina quando la tensione supera una determinata soglia, per questo motivo si misura nuovamente la tensione della batteria prima del controllo. A questo punto inizia il ciclo di carica a tensione costante. Purtroppo il sistema non misura l'esatta tensione della batteria.

L'accumulatore, infatti, si trova all'interno del satellite ed è collegato al connettore di test tramite cavi e passaggi da una scheda ad un'altra. Il cavo che collega il

caricabatterie al satellite è piuttosto lungo, ma soprattutto la tensione misurata comprende anche le cadute sul contatto del relè ed eventuali altre perdite. Non è quindi possibile quindi fare riferimento ad una soglia fissa. Il problema è stato risolto effettuando una taratura in tempo reale del sistema.

Quando la tensione sulla batteria si avvicina ad 8,4 Volt, precisamente quando arriva a 8,2 Volt, il sistema interrompe momentaneamente la carica, scollega la batteria ed immediatamente dopo ne misura la tensione. Questa misura risulta influenzata dalle medesime cadute e consente di sapere quale sia la caduta di tensione tra il punto di misura e la batteria facendo una semplice sottrazione. Fatto ciò la batteria viene ricollegata e la soglia di 8,4 Volt viene aumentata in misura corrispondente alla caduta appena misurata.

Le batterie al nichel-cadmio prevedono una procedura più complessa, poiché durante il ciclo di carica è necessario quantificare l'energia fornita alla batteria e svolgere una serie di operazioni che vedremo meglio in seguito.

4.3.2 Carica a tensione costante per le batterie agli ioni di litio

Abbiamo visto che, durante il ciclo a corrente costante, quando la tensione supera una determinata soglia, dettata dalle specifiche ed aggiornata in automatico dal sistema, è necessario mantenere costante la tensione, mantenendo la batteria in carica fino a quando la corrente non scende al di sotto della soglia prestabilita (150mA).

In figura 4.5 il diagramma di flusso della procedura.

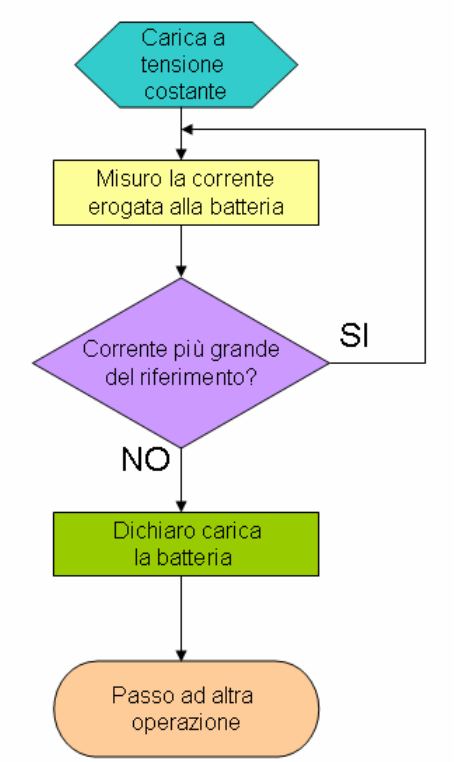


Figura 4.5 – *Carica a tensione costante*

E’ evidente che la procedura è molto più semplice: si tratta infatti solo di attendere che la corrente scenda naturalmente al di sotto della soglia voluta, controllandone periodicamente il valore.

In figura 4.6 è riportato il pannello dei comandi dopo che è stata caricata una cella agli ioni di litio. Si possono apprezzare i grafici della tensione e della corrente che seguono l’andamento desiderato. Il grafico mostra chiaramente che la tensione viene mantenuta a circa 9 Volt e non agli 8,4 Volt di specifica. La differenza è dovuta, come già spiegato, alla compensazione della caduta di tensione tra il punto di misura e la batteria.

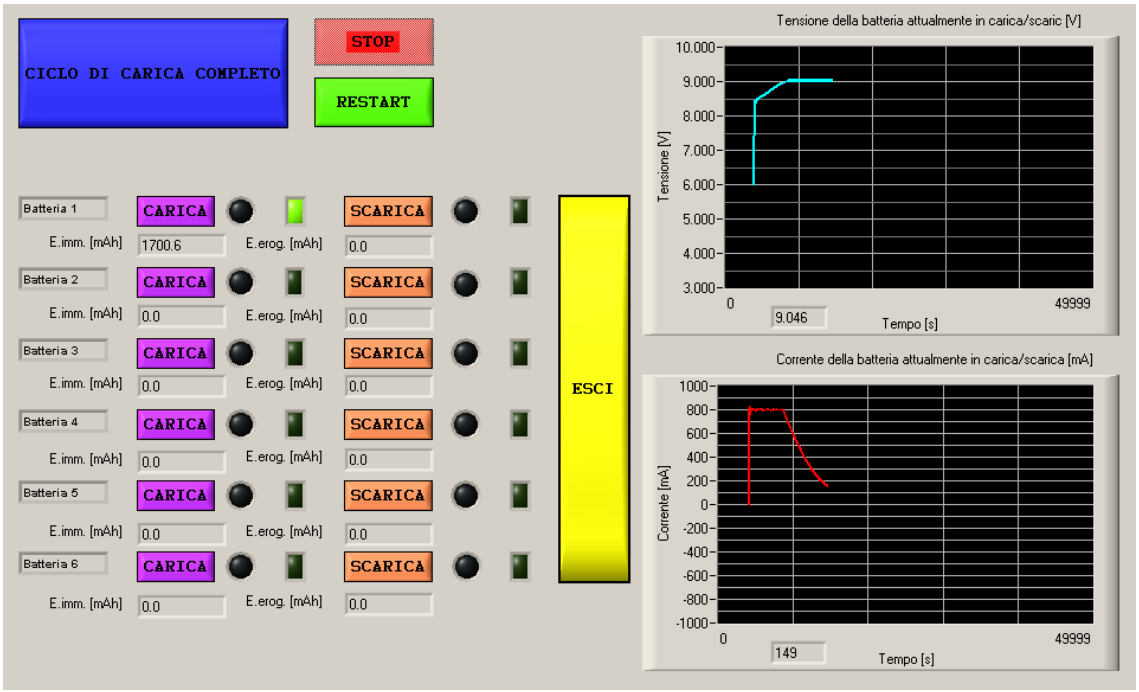


Figura 4.6 – *Pannello di controllo al termine del ciclo di carica di una batteria agli ioni di litio*

4.3.3 Il calcolo dell'energia fornita alle batterie al nichel-cadmio

Come illustrato nel precedente capitolo, quando sono state descritte le specifiche della batterie, gli accumulatori al nichel-cadmio necessitano di una procedura di carica più complessa. Essa deve avvenire a corrente costante per tutta la durata del ciclo, tenendo conto tuttavia dell'energia fornita alle celle. E' necessario fornire il 40% dell'energia erogata fino all'istante in cui l'andamento della tensione sulla batteria presenta un punto di massimo.

La procedura di carica dovrà quindi tenere conto di ciò, come illustrato in figura 4.7.

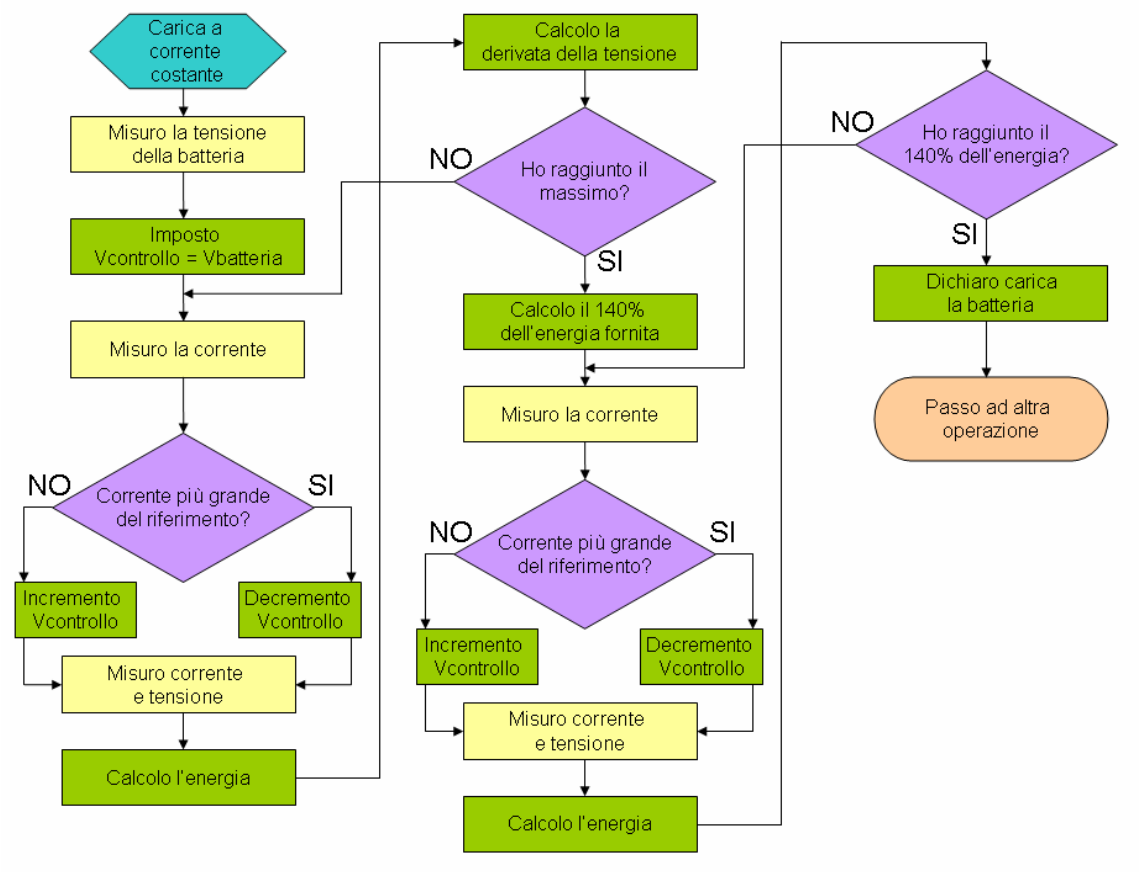


Figura 4.7 – *Carica a corrente costante per le batterie al NiCd*

Nel diagramma di flusso possiamo vedere come viene implementata la condizione di uscita dal ciclo di carica. Dopo la prima parte della procedura, già illustrata in precedenza, mi trovo a misurare tensione e corrente per poter calcolare l'energia. Osservo l'andamento della tensione nel tempo per accorgermi del raggiungimento del punto di massimo.

Se il massimo della tensione è raggiunto, significa che l'energia fornita fino a quel momento è il 100%, ne calcolo allora il 140% da utilizzare come soglia di termine del ciclo di carica. A questo punto riprendo la carica mantenendo sempre costante la corrente ed attendendo che l'energia fornita all'accumulatore raggiunga la soglia. Quando questo avviene, dichiaro carica la batteria e passo ad un'altra operazione.

Prima di svolgere tutta questa complessa procedura, nel caso degli accumulatori al nichel-cadmio, dovrò aver scaricato completamente la batteria.

In figura 4.8 è riportato il pannello dei comandi dopo aver caricato una cella al nichel-cadmio. Si possono osservare i grafici della tensione e della corrente che seguono l'andamento desiderato. E' visibile anche la fase di scarica antecedente il processo di carica. Quale sia la procedura di scarica lo vedremo in seguito

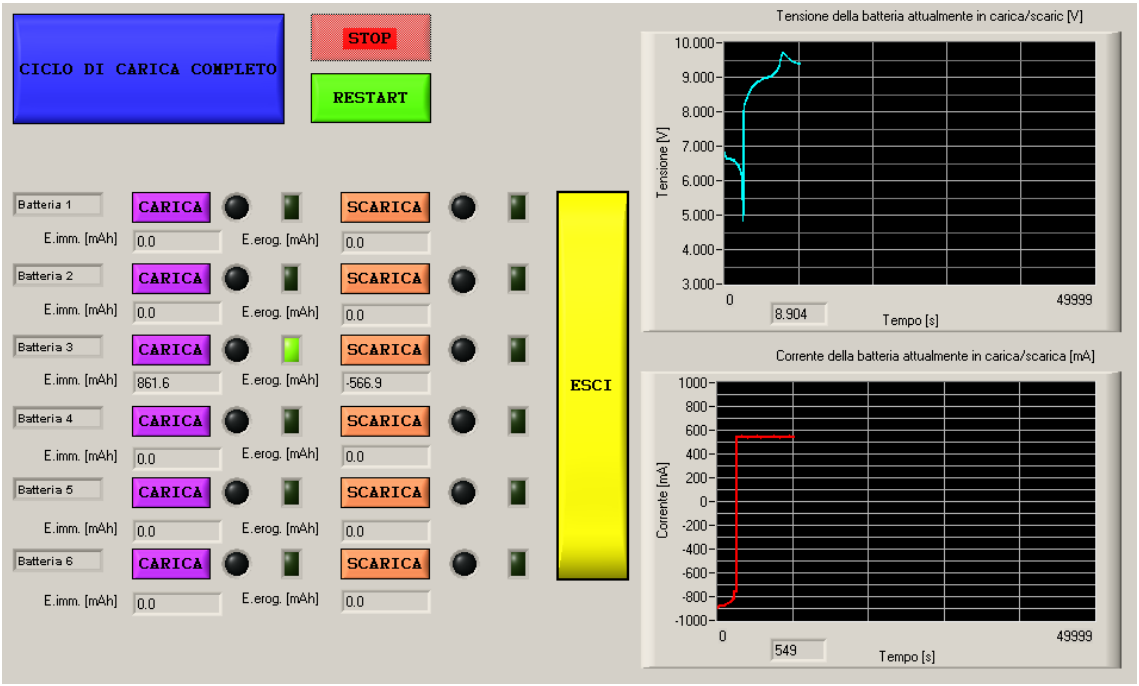


Figura 4.8 – *Pannello di controllo al termine del ciclo di scarica/carica di una batteria al nichel-cadmio*

4.3.4 La scarica

La procedura di scarica è molto semplice. Il programma si limita ad acquisire la tensione e la corrente e dichiara scarica la batteria quando la tensione scende sotto una determinata soglia, differente per i due tipi di cella.

In realtà la procedura di scarica è necessaria solo per le celle al nichel-cadmio, ma per completezza si è reso possibile attuarla anche per gli accumulatori agli ioni di litio. Infatti, questa applicazione permette di sapere quanto sia efficiente una batteria. Misurando quanta dell'energia fornita per la carica viene restituita all'utilizzatore è possibile calcolare una sorta di rendimento dell'accumulatore e ciò è stato utile in fase di scelta della batterie da utilizzare sul satellite.

In figura 4.9 è illustrato il diagramma di flusso della procedura di scarica.

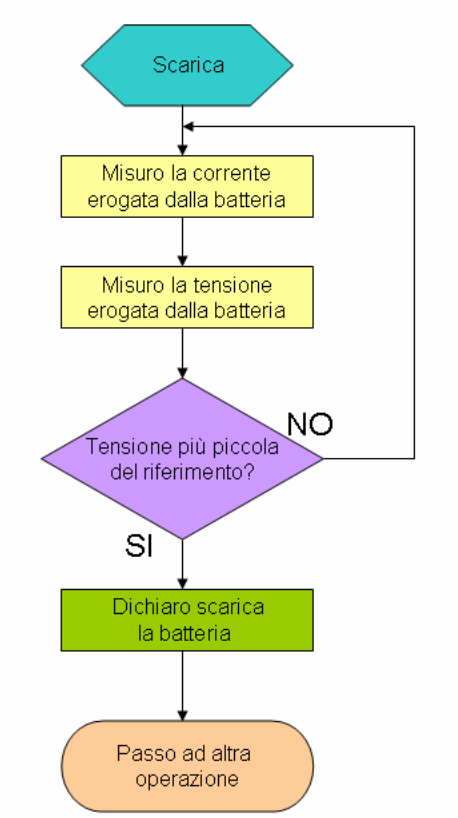


Figura 4.9 – *Procedura di scarica di una batteria*

4.4 L’interfaccia grafica del dispositivo

Analizziamo ora l’interfaccia del dispositivo, così come si presenta ad un utente qualsiasi. In figura 4.10 è riportata la parte di comando del pannello di controllo.

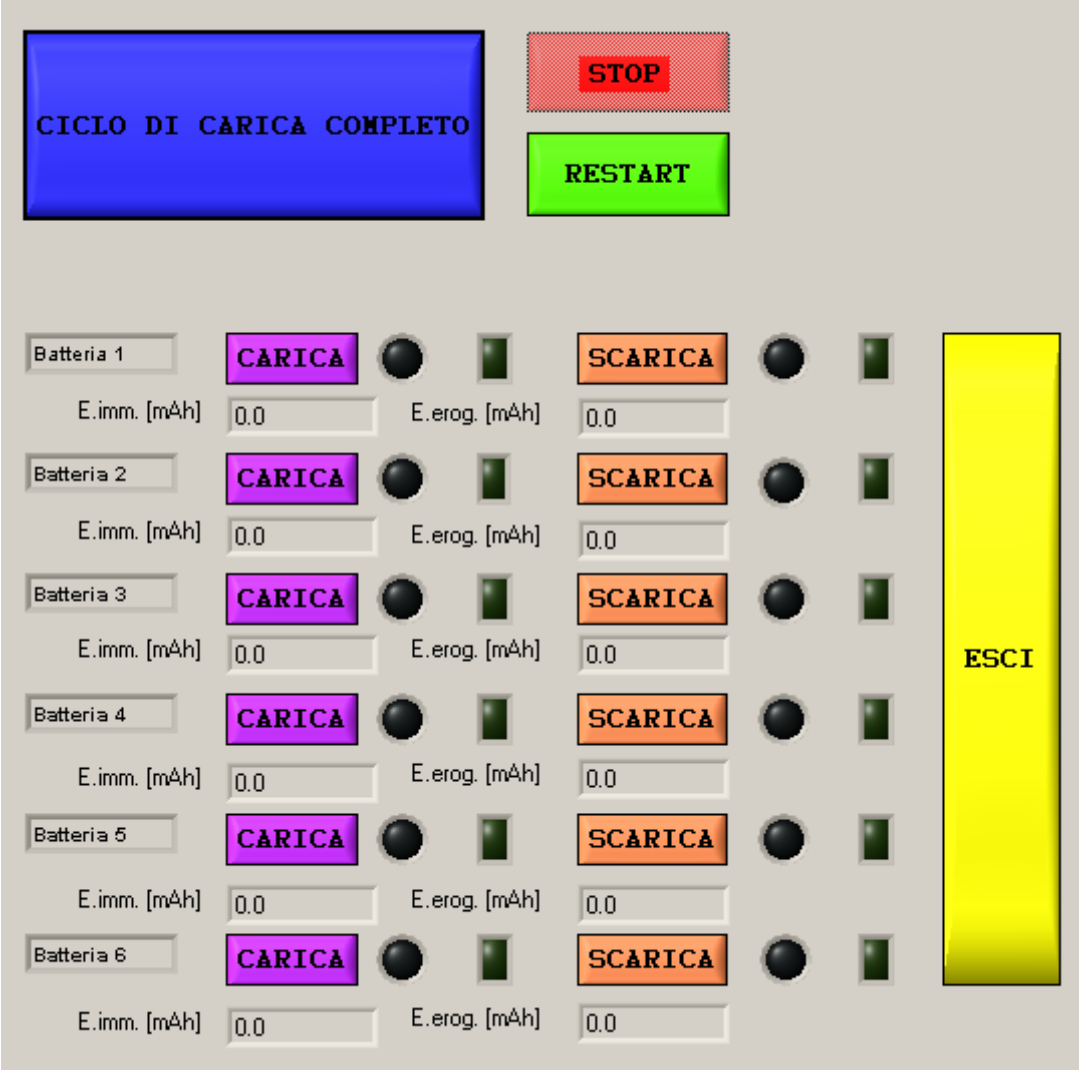


Figura 4.10 – *Settore di comando del pannello ci controllo*

I pulsanti CARICA e SCARICA servono per effettuare la singola operazione sulla specifica batteria. L’utente ha a disposizione una coppia di pulsanti per ogni accumulatore. La pressione comporta il settaggio di tutti i parametri necessari a svolgere l’operazione richiesta e avvia una delle procedure precedentemente descritte, a seconda della batteria che si intende caricare o scaricare. Ogni pulsante è corredato da due indicatori luminosi (vogliono imitare dei led). Il led rotondo, di colore rosso, indica che

il processo è in corso, si accende alla pressione del relativo pulsante e si spegne a processo terminato. Il led rettangolare verde si accende a processo terminato per indicare il corretto svolgimento dell'operazione richiesta.

Sotto ad ognuno dei pulsanti CARICA è presente un indicatore che mostra (a processo terminato correttamente) l'energia fornita alla batteria e quindi immagazzinata dalla stessa. L'indicatore collocato sotto ai pulsanti SCARICA, invece, indica l'energia restituita dalla batteria (sempre a processo terminato correttamente)

Il pulsante CICLO DI CARICA COMPLETO permette di caricare consecutivamente, una alla volta tutte le sei batterie del satellite. Premendolo vengono impostati tutti i parametri necessari a svolgere questa lunga e complessa operazione. Gli indicatori "luminosi" rossi delle singole batterie si accendono uno alla volta, indicando quale batteria è in carica o in scarica pre-carica per le celle al nichel-cadmio, l'indicatore verde si accende quando una batteria è stata caricata.

La pressione di tutti i pulsanti descritti fino ad ora attiva il pulsante STOP, che non è possibile utilizzare se non vi sono processi in corso, disabilita tutti i pulsanti ad eccezione del pulsante ESCI.

Il pulsante STOP serve per bloccare il processo in corso. Tutte le batterie vengono sconnesse dal sistema, siano esse in carica od in scarica. Alla pressione del pulsante STOP, il pulsante RESTART e tutti i pulsanti di carica e scarica vengono attivati.

Il pulsante RESTART serve per riprendere l'operazione bloccata dalla pressione di STOP dal punto in cui è stata interrotta. Più è rapida la pressione di RESTART dopo aver premuto STOP, minori sono i danni causati alle batterie. Se è necessario restare in stato di STOP per molto tempo (più di qualche minuto) è consigliabile riprendere la carica o lo scarica premendo il pulsante specifico.

Quando viene premuto il tasto STOP è quindi possibile riprendere rapidamente la funzione interrotta premendo RESTART oppure iniziare una nuova procedura premendo uno dei pulsanti precedentemente descritti.

Il pulsante ESCI è sempre attivo e permette in qualunque momento di interrompere tutte le operazioni, scollegare tutte le batterie e chiudere l'applicazione.

Vediamo ora la seconda parte del pannello di controllo. In figura 4.11 sono evidenziati i due pannelli grafici che mostrano la tensione e la corrente misurate.

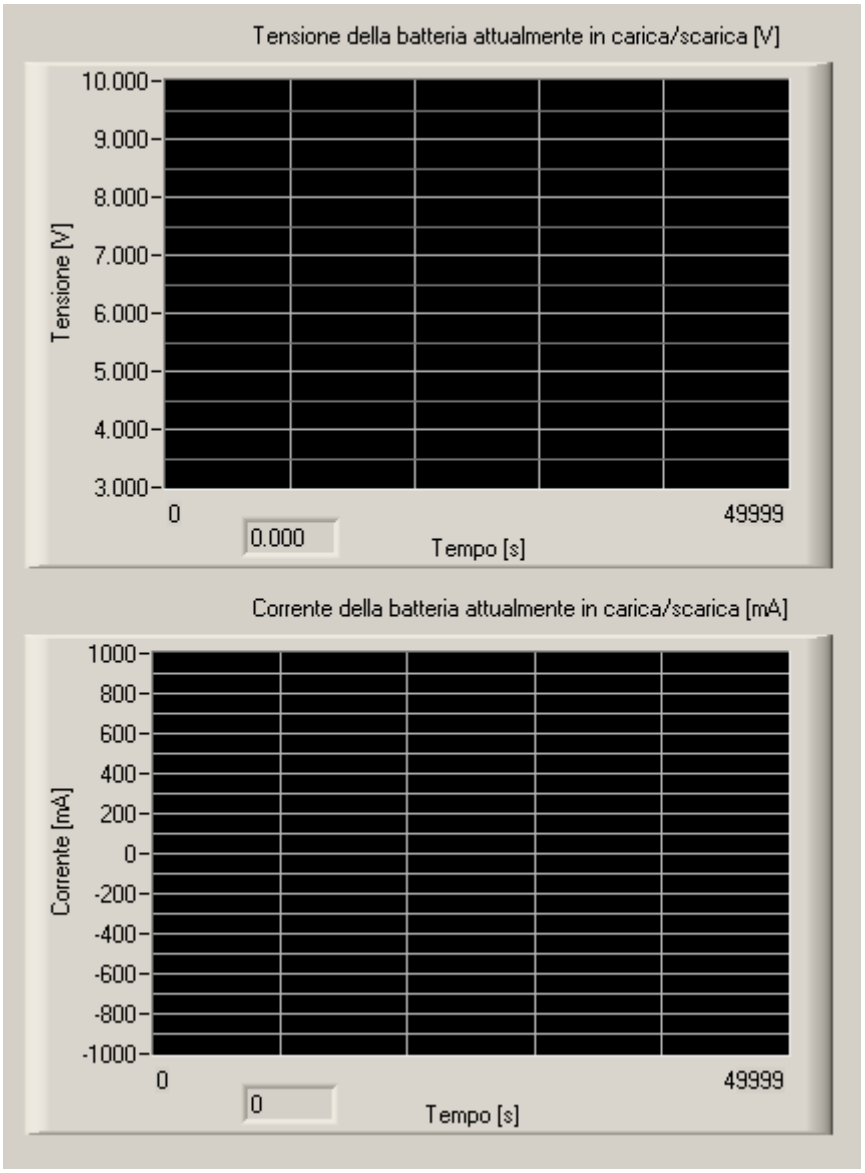


Figura 4.11 – *Indicatori grafici*

Su questi due indicatori grafici viene tracciato l'andamento della tensione e della corrente di carica o di scarica. L'andamento viene aggiornato ogni secondo ed il grafico permette di controllare procedure della durata di 50.000 secondi (circa 14 ore), Per processi più lunghi, vengono mostrati gli ultimi 50.000 secondi.

I due indicatori numerici, posti sotto ad ognuno dei grafici, indicano il valore istantaneo della tensione [V] e della corrente [mA] e vengono aggiornati dieci volte al secondo.

Capitolo 5

La scheda di interfaccia per la programmazione del satellite

La necessità di realizzare questa scheda è dovuta ad un errore in fase di progettazione dell'architettura di PiCPoT.

Il sistema elettronico originario prevedeva, attraverso il connettore J0, il collegamento in cascata di tutti i processori presenti sulle schede del satellite. Il connettore J0 è un connettore a 140 poli: su ogni scheda ne sono installati due, uno per faccia, in modo da consentire il collegamento in catena di tutte le schede.

Questa architettura consente di avere a disposizione su tutte le schede ognuno dei segnali utili al sistema, compresi quelli necessari alla programmazione dei processori.

Era infatti previsto che le connessioni JTAG, necessarie per programmare i diversi sottosistemi del satellite, fossero unificate e comuni per tutti i processori; tale struttura avrebbe consentito, con l'aggiunta di tre segnali di selezione, di programmare il processore voluto selezionandolo dall'esterno.

Attraverso la scheda ProcA le connessioni JTAG, i segnali di selezione e tutte le altre connessioni necessarie erano disponibili tramite il connettore di test per consentire il collaudo del satellite assemblato.

Purtroppo, in fase di scelta dei processori da utilizzare, questa necessità è stata trascurata e sono stati scelti ed utilizzati processori che non sono compatibili con questa architettura. Essi, infatti, possono essere programmati solo se completamente isolati da altri sistemi. In più, i vari sottosistemi di PiCPoT sono stati sviluppati, testati e messi a punto in maniera indipendente, realizzando il sistema completo solo alla fine, perciò ci si è accorti in ritardo del problema. A quel punto, per far fronte all'inconveniente, erano disponibili tre opzioni.

La prima, che non avrebbe comportato modifiche al satellite, consisteva nello smontare il pacco schede ogni qualvolta fosse stata necessaria una minima modifica al software di uno dei processori, in modo da consentire la riprogrammazione della scheda “in solitudine”. Questa scelta avrebbe comportato un enorme dispendio di tempo e nelle concitate fasi di collaudo finale non era assolutamente applicabile.

La seconda avrebbe comportato il rifacimento delle schede, inserendo al loro interno un modulo con una FPGA che si occupasse di standardizzare la connessione JTAG dei processori. Anche questa soluzione è stata scartata per motivi di tempo.

La terza possibilità, quella poi attuata, ha portato alla costruzione di una scheda di interfaccia. Sono state effettuate piccole modifiche sul satellite: sono state interrotte le connessioni in catena sul connettore J0 e si è provveduto a portare all’esterno dello stesso, con l’ausilio di cavi e di connettori, i pin di programmazione delle diverse schede. A collaudi ultimati i cavi di collegamento tra le schede di PiCPoT e la scheda di interfaccia sono stati eliminati.

In questo capitolo vengono illustrate le modifiche apportate al satellite e gli schemi elettrici della scheda di interfaccia.

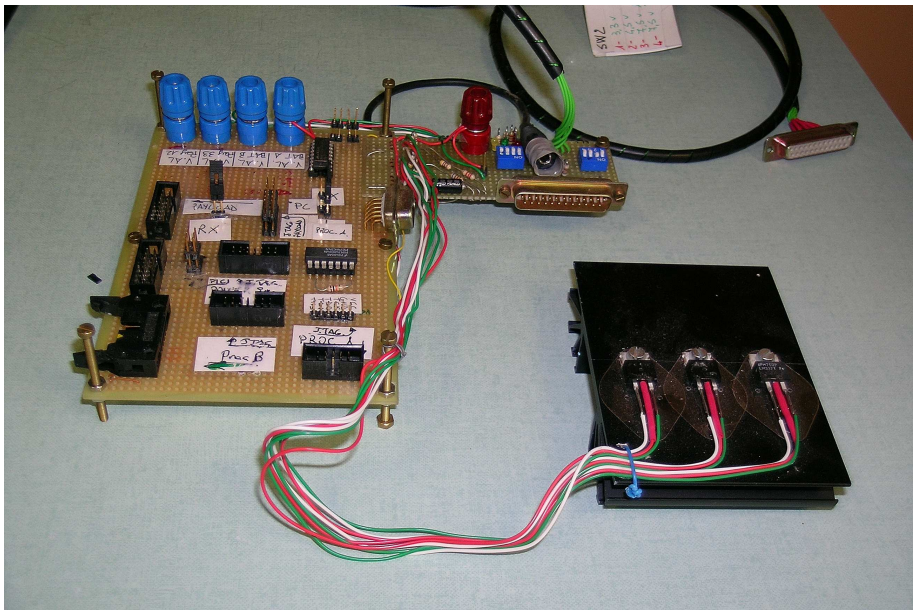


Figura 5.1 – *La scheda di interfaccia*

5.1 Le modifiche apportate al connettore J0

In primo luogo si è reso necessario isolare, dal punto di vista della programmazione, i vari microprocessori. Per far ciò si è deciso di interrompere fisicamente le connessioni su ogni singola scheda, mantenendo intatta la struttura del connettore J0.

Si è quindi provveduto ad interrompere sulle schede ProcA, ProcB, PowerSwitch e Payload le connessioni con i pin numero 21, 22, 26 e 27 del connettore J0.

Tali connessioni portavano i seguenti segnali:

# PIN	Segnale
21	TDI
22	TDO
26	TCK
27	TMS

Questi quattro segnali sono tali da condizionare le modifiche ad ognuna delle schede; la condivisione di queste connessioni costituiva la sostanza del problema, poiché si tratta dei segnali necessari a tutte le schede per essere programmate.

5.2 Le modifiche apportate alle schede

Vediamo ora nel dettaglio quali segnali è stato necessario portare all'esterno del satellite per consentire la corretta programmazione dei diversi processori.

5.2.1 ProcA

La scheda ProcA dispone di due unità programmabili, necessita quindi di due connessioni verso l'esterno per essere programmata.

Un'unità programmabile è costituita da un microcontrollore Chipcon CC1010. Tale unità non crea nessun conflitto con le altre schede, poiché usa segnali di programmazione non JTAG, si programma infatti tramite SPI via porta parallela di un personal computer. Si è quindi deciso di non apportare modifiche a questa parte di ProcA vista la possibilità di eseguirne la programmazione attraverso l'esistente connettore di test.

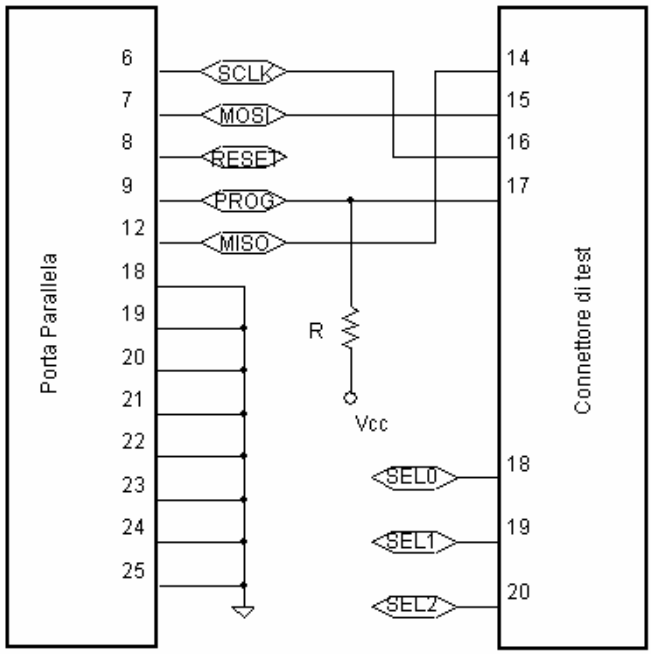


Figura 5.2 – *Connessioni utilizzate per la programmazione dell'unità Chipcon CC1010 su ProcA*

I pin numero 18, 19 e 20 del connettore di test erano i pin destinati alla selezione del processore da programmare.

La seconda unità programmabile installata su ProcA è una CPLD di Xilinx (XCR3064XL) che utilizza un programmatore JTAG tramite porta parallela di un personal computer. Si è quindi reso necessario modificare la scheda, portando all'esterno attraverso un connettore a 10 poli i segnali necessari. Attraverso lo stesso connettore si è provveduto a fornire le alimentazioni necessarie a tutti i sistemi in fase di programmazione.

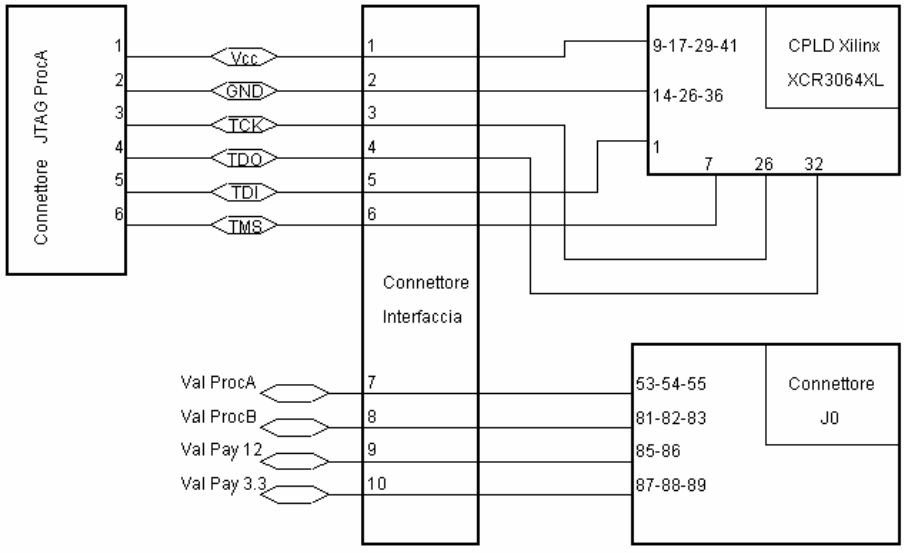


Figura 5.3 – *Schema del collegamento tra JTAG e ProcA*

5.2.2 ProcB

A bordo di ProcB è installato un solo dispositivo programmabile, denominato MSP430 prodotto da Texas Instruments.

Per poter essere programmato dall'esterno necessita di 7 segnali, generati da un personal computer attraverso la porta parallela e trasmessi alla scheda di interfaccia tramite un connettore JTAG a 14 poli. La scheda si occupa di inviarli all'interno del satellite tramite un connettore a 10 poli. Sono quindi presenti tre connessioni libere, che possono essere utilizzate in qualunque momento per eseguire operazioni aggiuntive.

Nello schema si può notare che il segnale TMS prima di essere collegato al processore viene connesso ad una porta NOT. Questa aggiunta non è dovuta ad una carenza del programmatore JTAG, bensì alla particolare architettura di PiCPoT.

Poiché il segnale TMS è normalmente settato sul livello logico "alto", mentre tutti i segnali presenti su J0 sono settati sul livello logico "basso", per evitare problemi di interfaccia tra una scheda ed un'altra si è deciso di negare questo segnale a monte di JO (sulla scheda procA) e di invertirlo nuovamente su ogni scheda interessata. Quando si è reso necessario programmare direttamente ogni scheda, sul collegamento tra il programmatore JTAG e il processore è rimasto un inverter. Si è deciso di non eliminare questa porta logica per ridurre al minimo le modifiche da apportare al satellite e di inserirne un'altra sulla scheda di interfaccia per annullarne l'effetto.

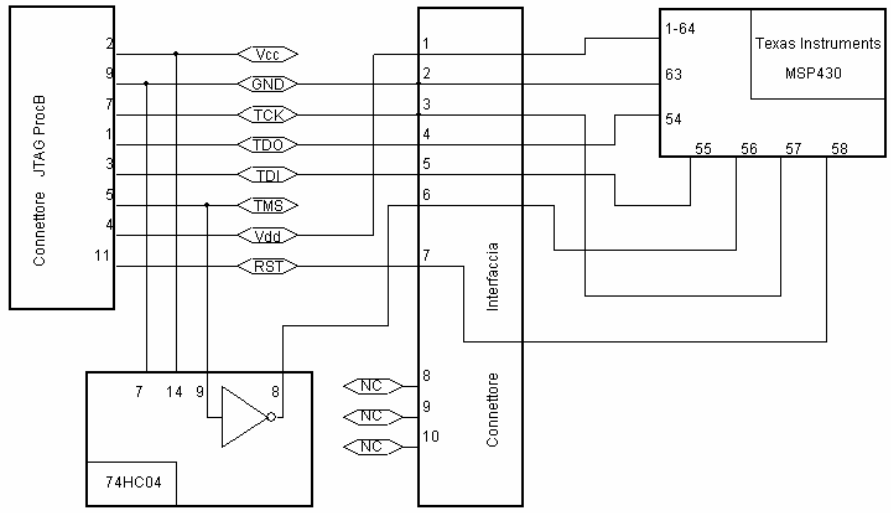


Figura 5.4 – *Collegamento tra JTAG e ProcB*

5.2.3 PowerSwitch

Questa scheda si occupa di distribuire l'energia agli altri sottosistemi, minimizzando i consumi e massimizzando l'efficienza. Per evitare che un guasto potesse compromettere l'intero sistema, su PowerSwitch sono stati installati due processori, ognuno dei quali si occupa di gestire l'alimentazione di una delle due catene di trasmissione.

I due processori sono un MSP430 di Texas Instruments ed un PIC17F877A di Microcip. Il primo, come già detto per ProcB, viene programmato tramite JTAG, il secondo viene programmato tramite un PIC dedicato.

Il JTAG necessita di 7 connessioni, il PIC di 6. La scheda di interfaccia invia quindi 13 segnali alla scheda PowerSwitch, utilizzando un connettore a 14 poli.

Il segnale TMS è connesso al processore tramite un inverter per gli stessi motivi indicati nel paragrafo 5.2.2.

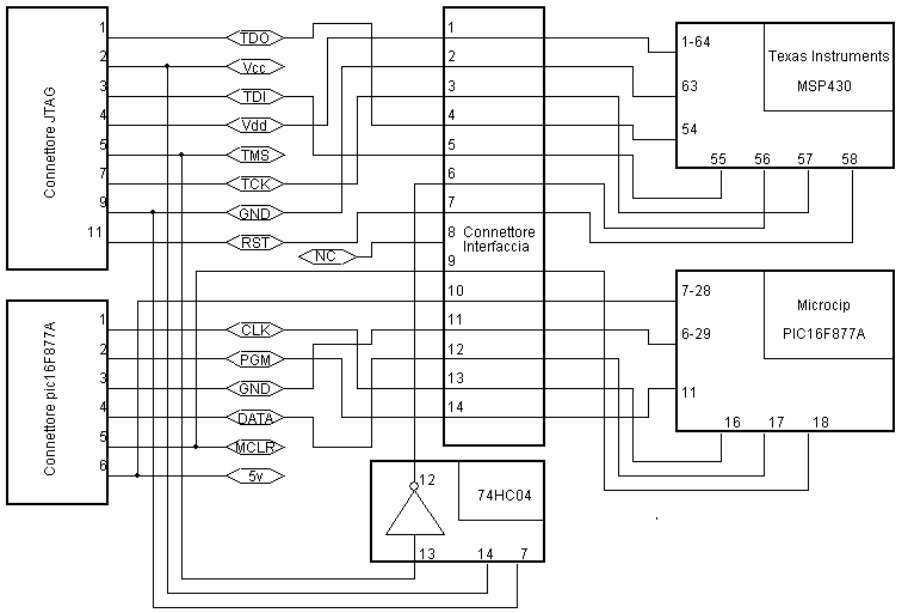


Figura 5.5 – *Schema dei collegamenti tra PowerSwitch ed i due programmatori*

5.2.4 Payload

A bordo di Payload è installato un processore prodotto da Analog Devices denominato BlackFINN. Per programmare questo processore sono necessarie due interfacce, un'interfaccia JTAG ed una porta seriale.

Poiché i livelli di tensione dei segnali forniti dalle porte seriali dei personal computer non sono standardizzati e variano da macchina a macchina, per garantire il funzionamento della scheda di interfaccia è necessario utilizzare un traslatore di livello.

In questo caso un circuito integrato (MAX3232) svolge questo adattamento. Oltre a garantire il corretto funzionamento, evita che una connessione seriale con tensioni troppo alte possa danneggiare il processore.

Sulla scheda di interfaccia sono poi presenti anche un interruttore ed un deviatore necessari per la procedura di programmazione.

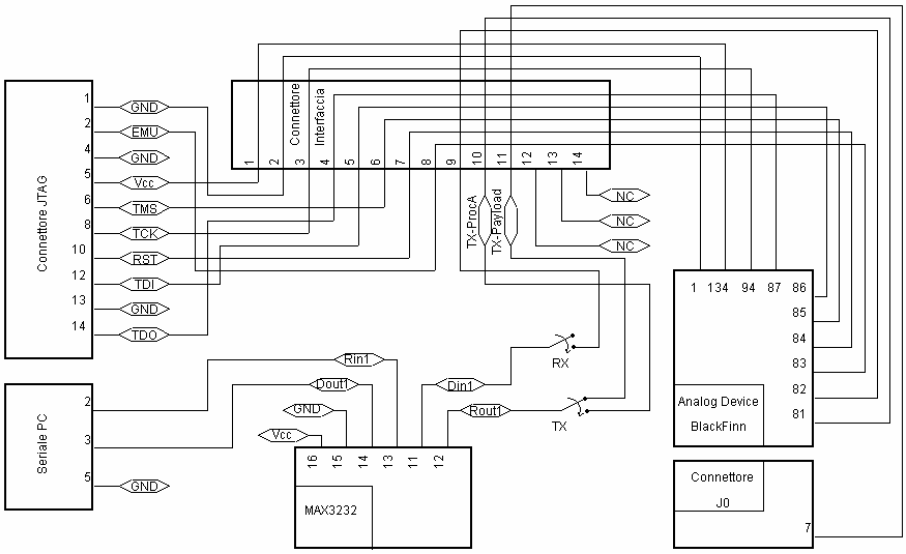


Figura 5.6 – *Schema delle connessioni necessarie per la programmazione della scheda Payload*

5.3 I circuiti a corollario della scheda di interfaccia

Per il corretto funzionamento della scheda di interfaccia sono necessari alcuni circuiti a corollario.

Servono infatti alcuni regolatori di tensione ed un circuito di regolazione per il traslatore di livello. Di seguito vedremo, brevemente gli schemi elettrici di questi apparati.

5.3.1 I regolatori di tensione

La scheda di interfaccia è alimentata per mezzo di un generatore da laboratorio con una tensione continua di 12 Volt. Per effettuare la programmazione dei processori sono necessarie tre diverse tensioni di alimentazione: 3.3 Volt, 5 Volt e 7,5 Volt.

Le due tensioni più basse servono da alimentazione ai processori, i 7,5 Volt sono invece necessari per simulare la tensione fornita dalle batterie del satellite.

Per realizzare i regolatori di tensione si è deciso di utilizzare tre LM317, componenti di agevole utilizzo che con una semplice rete di regolazione esterna permettono di ottenere la tensione desiderata. Poiché la corrente assorbita attraverso i regolatori è abbastanza elevata (circa 500 mA) si è deciso di installare gli LM317 su un dissipatore, garantendone così il funzionamento anche per lungo tempo (il dissipatore è chiaramente visibile in figura 5.1).

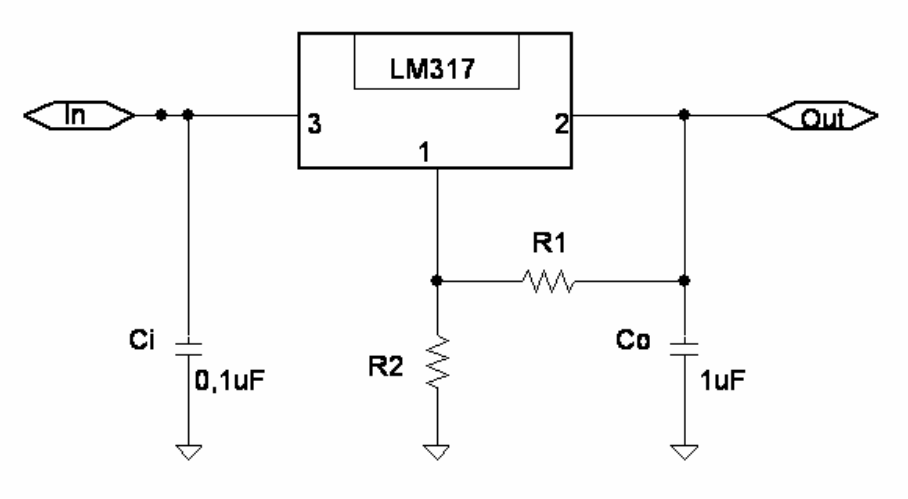


Figura 5.7 – *Schema circuitale dei regolatori di tensione*

In figura 5.7 è illustrato lo schema elettrico utilizzato per generare le tre tensioni necessarie. I tre regolatori differiscono solo per i valori dei due resistori R1 ed R2, in particolare:

Vout [V]	R1 [Ohm]	R2 [Ohm]
3,3	235	390
5,0	235	680
7,5	235	1200

La resitenza da 235 Ohm è realizzata con la connessione in parallelo di due resistori da 470 Ohm. I valori di R1 ed R2 sono ricavati dalla legge:

$$Vout \sim 1{,}25 * (1 + (R2 / R1)) \quad [Volt].$$

Questa relazione fornisce una prima indicazione per il calcolo dei valori dei resistori. La relazione corretta tiene conto anche di un altro parametro, I_{ADJ}.

I_{ADJ} è la corrente che scorre in R2 e dipende fortemente dalla temperatura. E' molto piccola (circa 50 µA), ma se la temperatura del dispositivo cresce troppo, con valori elevati di R2 la deviazione della tensione può diventare rilevante. La relazione corretta risulta quindi essere:

$$Vout = 1{,}25 * (1 + (R2 / R1)) + I_{ADJ} * R2 \quad [Volt].$$

Maggiori delucidazioni sulle caratteristiche del LM317 sono fornite dal foglio tecnico allegato in appendice.

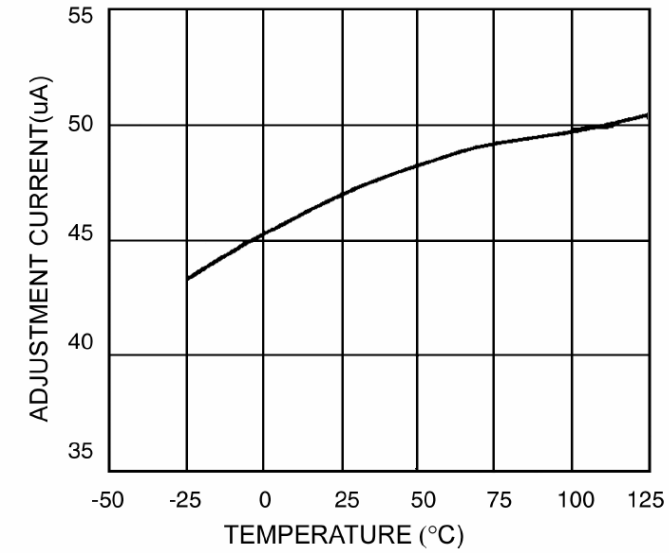


Figura 5.7 – *Andamento di* I_{ADJ} *al variare della temperatura*

5.3.2 Il traslatore di livello MAX3232

Per funzionare correttamente, il traslatore di livello realizzato con un MAX3232 necessita di alcuni componenti esterni, come è illustrato in figura 5.8.

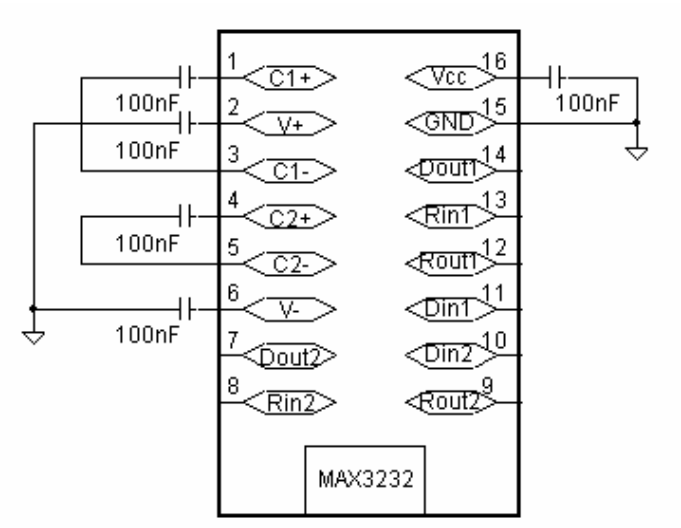


Figura 5.9 – *Schema di collegamento del MAX3232 necessario per un corretto funzionamento del dispositivo*

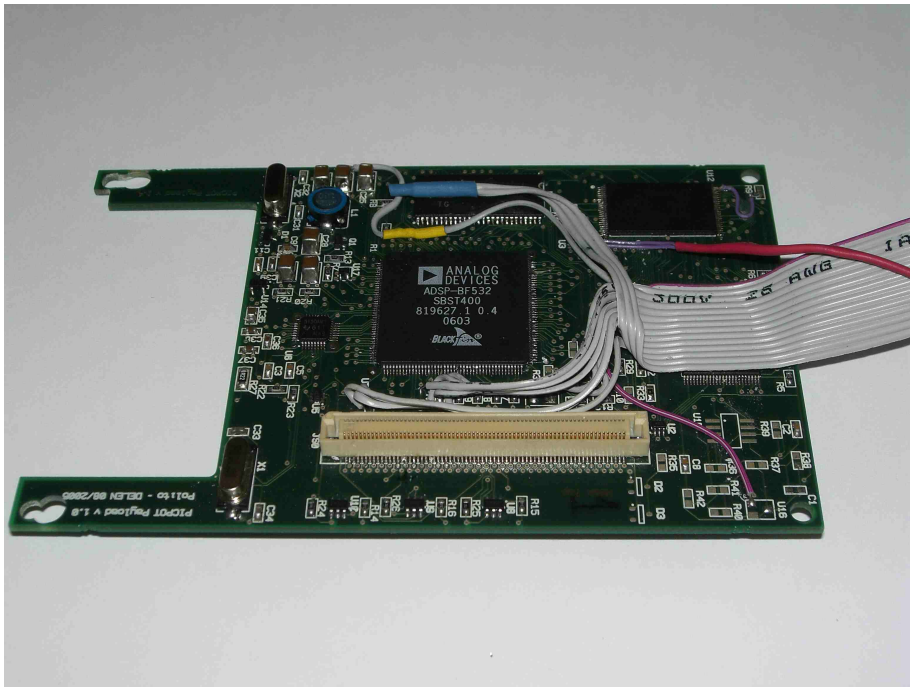


Figura 5.10 – *Scheda Payload con il cavo necessario al collegamento con la scheda di interfaccia*

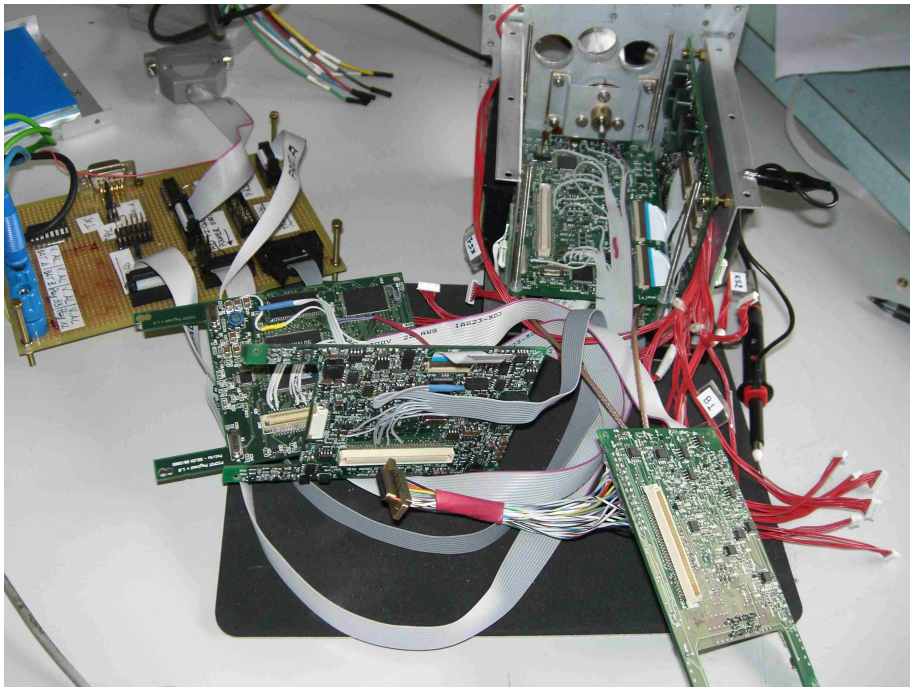


Figura 5.11 – *PiCPoT con i cavi di collegamento allacciati alla scheda di interfaccia*

Capitolo 6

La stazione di terra portatile ed i collaudi pre-lancio

Oltre alla vera e propria stazione di terra è stato indispensabile costruire una stazione portatile. Infatti durante la fase di collaudo, ma soprattutto durante la fase di integrazione, non sarebbe stato possibile comunicare con la stazione di terra principale.

Questa stazione, di piccole dimensioni (20 x 22 x 9 centimetri), è stata portata a Baikonur (KAZ) durante l'integrazione sul lanciatore ed ha permesso di effettuare gli ultimi controlli al satellite, prima di abbandonarlo al suo destino.

Per accelerare la fase di progetto e di realizzazione si è deciso di utilizzare apparati già sviluppati per altre applicazioni sempre all'interno del progetto PiCPoT.

La stazione di terra portatile è costituita principalmente da una scheda TxRx di PiCPoT modificata per permettere la comunicazione con un PC. Considerata la distanza ridotta a cui sarà collocata rispetto al satellite, per semplicità non si è impiegato alcun LNA od amplificatore di potenza e si è scelto anche di utilizzare un unico CC 2400.

Per semplificare ulteriormente il sistema, non è stato utilizzato alcun microcontrollore per la gestione del canale a 2,44 GHz: ciò comporta che tutte le operazioni vengano gestite tramite la porta parallela di un computer.

Il canale a 437 MHz è stato realizzato utilizzando un modulo dimostrativo del microcontrollore CC1010 della Chipcon, prodotto dalla stessa azienda. A questo modulo sono state affiancate due schede di interfaccia per permetterne l'interazione con un personal computer in fase di programmazione e d'uso.

Questa serie di semplificazioni ha però trasferito altrove la complessità del problema, è stato infatti necessario realizzare un programma in grado di emulare il protocollo SPI per comunicare direttamente con il transceiver e ricevere o trasmettere i dati. La stesura

del software si è rivelata particolarmente complessa in quanto Windows non permette una facile gestione delle applicazioni in tempo reale.

Per realizzare il programma di gestione ci si è affidati all'ambiente LabWindows/CVI di cui si è già parlato nel capitolo 4.

Nei prossimi paragrafi verrà illustrata la struttura delle schede di collegamento tra il personal computer e gli apparati preesistenti utilizzati per la realizzazione della stazione di terra.

6.1 – Il canale a 437 MHz

Per realizzare questo sottosistema si è deciso di utilizzare un modulo dimostrativo del microcontrollore CC1010 costruito da Chipcon. Questo modulo è stato installato su una scheda di interfaccia per portare all'esterno i segnali necessari senza intervenire direttamente sull'architettura dello stesso.

L'apparato presenta due connessioni verso il calcolatore. Una connessione parallela necessaria alla programmazione del dispositivo ed una connessione tramite porta seriale RS232 per il trasferimento delle informazioni durante il funzionamento.

La tensione di alimentazione, compresa tra 7 e 12 Volt, viene ridotta ai 3,3 Volt necessari tramite un regolatore LM317, di cui si è già dissertato nel capitolo 5.

La rete di programmazione e di interfaccia per la comunicazione è perlopiù costituita da due traslatori di livello. Per la sottorete di programmazione i livelli di tensione vengono adeguati a quelli del modulo dimostrativo per mezzo di un buffer modello 74HC244, mentre per la parte di trasferimento dei dati tramite porta seriale si è utilizzato un MAX3232.

Il circuito con il diodo 10BQ100 è necessario per avere a disposizione un power-on reset e contemporaneamente per consentire di azzerare il microcontrollore durante l'uso. In figura è illustrato lo schema elettrico dell'apparato.

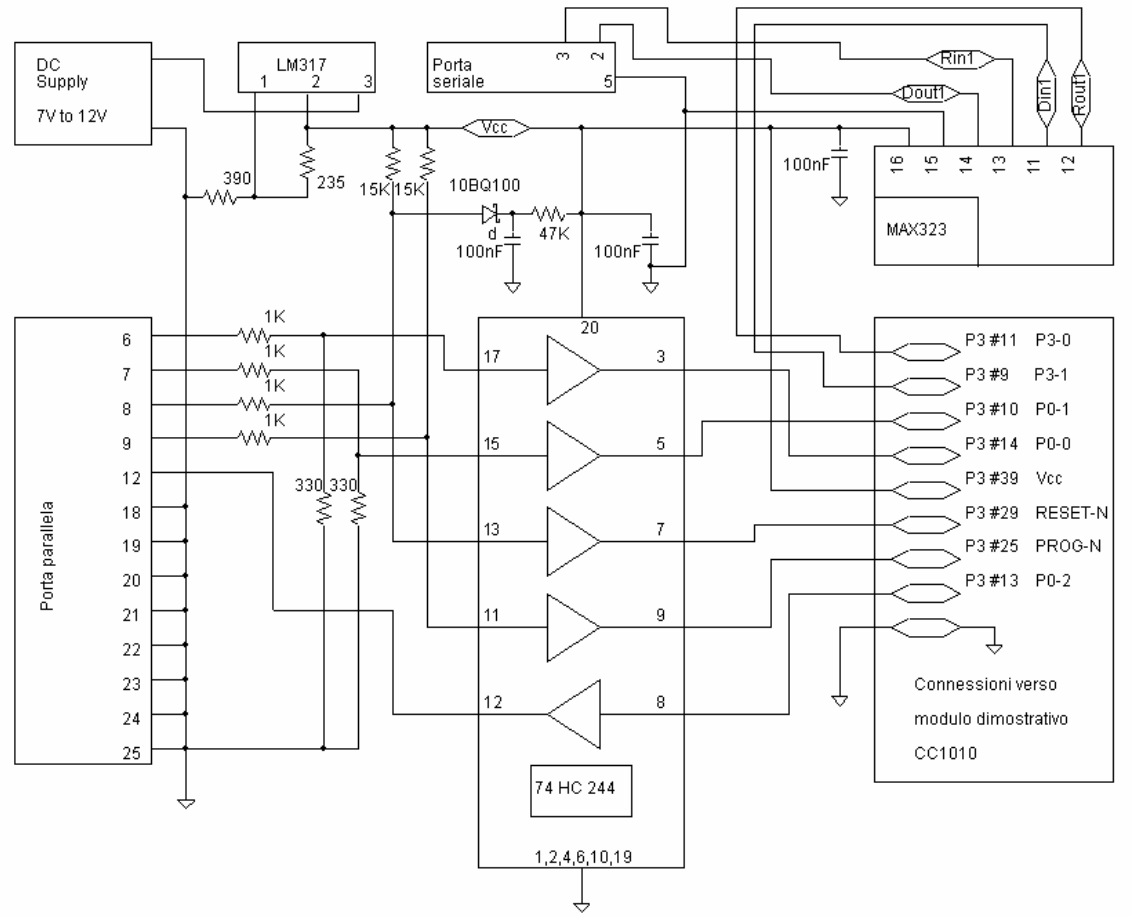


Figura 6.1 – *Schema elettrico dell'interfaccia per le comunicazioni a 437 MHz*

6.2 – Il canale a 2,44 GHz

Per consentire le comunicazioni a 2.44 GHz si è deciso di utilizzare una scheda TxRx di PiCPoT sulla quale sono stati installati solo i componenti necessari, rinunciando all'installazione dell'amplificatore.

La rete di interfaccia tra la scheda TxRx ed il personal computer è fondamentalmente composta da un traslatore di livello che consente di adattare i livelli di tensione della porta parallela a quelli più bassi utilizzati sulla scheda. Vi sono poi tre regolatori di tensione che provvedono a fornire alla scheda le alimentazioni necessarie al funzionamento.

A causa della scarsa reperibilità dell'oscillatore al quarzo installato sul satellite si è deciso di ricorrere ad un oscillatore esterno, con frequenza nominale doppia rispetto a

quella utilizzata su TxRx, per questo motivo il segnale viene fatto transitare attraverso un divisore di frequenza realizzato con un FlipFlop di tipo D.

Le figure illustrano i circuiti di collegamento.

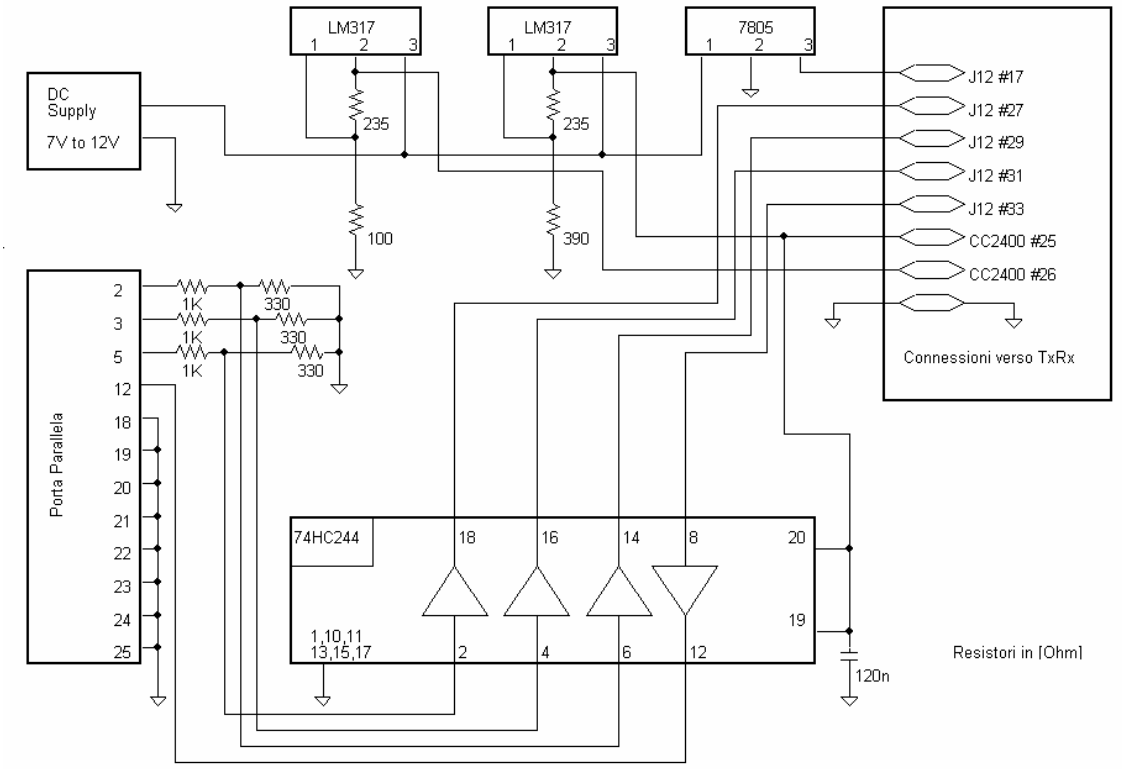


Figura 6.2 – *Schema di collegamento per il canale a 2,44 GHz*

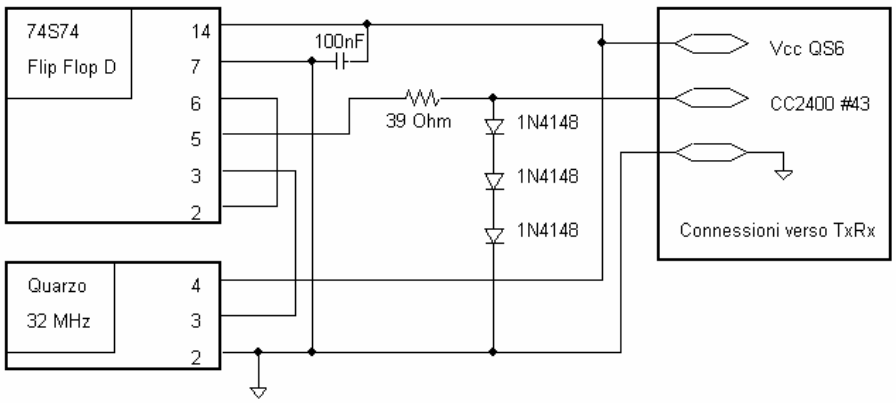


Figura 6.3 – *Schema di collegamento dell'oscillatore al quarzo*

6.3 – Il software di controllo

Il software realizzato è necessario alla gestione della stazione di terra portatile anche se con poche modifiche si potrebbe adattarlo per l'utilizzo della stazione di terra presente sul tetto del Politecnico. Il principale problema affrontato è stato la gestione in real-time della comunicazione via SPI con il CC2400 tramite la porta parallela.

L'uso della porta parallela è stato una scelta obbligata poiché si è scelto di non utilizzare un microcontrollore, come avviene invece nella stazione di terra fissa. La programmazione del software di controllo è stata affidata ad un altro studente.

Una volta realizzato il software è stato necessario testare tutte le sue funzioni ottenendo così di testare anche le funzioni del satellite. La maggior parte dei collaudi si è svolta con il satellite e la stazione di terra portatile vicini ma, nel periodo prima del lancio, si sono tenuti anche collaudi da distanze più elevate utilizzando la stazione di terra fissa. Vediamo nel dettaglio i collaudi effettuati.

6.4 – I collaudi

Durante il primo collaudo il satellite è stato lasciato acceso per 3 giorni senza alcun intervento esterno (eventuali reset del sistema) solamente con il caricabatterie collegato. Si è verificato quanti pacchetti di telemetria fossero effettivamente stati inviati dal satellite e ricevuti dalla stazione di terra e per questo collaudo le funzioni di statistica implementate nel software sono state utilissime.

Nei tre giorni trascorsi il satellite ha eseguito con precisione una trasmissione ogni minuto senza alcun problema ed ogni volta è stato possibile ricevere perfettamente il pacchetto trasmesso. Con questo test si è anche verificata la precisione del Real Time clock del satellite, andando a verificare che l'errore accumulato dall'orologio in tre giorni fosse minore di 1 secondo (non è stata effettuata una misura più precisa in quanto questo dato non era di particolare interesse).

È stato poi effettuato un test per determinare la vita del satellite nel caso in cui vi fossero problemi ai circuiti di gestione dei pannelli solari e la carica delle batterie non avvenisse quindi correttamente. Tale test ha dimostrato una vita del satellite compresa tra i due ed i tre giorni, ritenuta sufficiente per stabilire una comunicazione con esso e poter verificarne le funzionalità.

Dopo aver verificato che il sistema continuasse effettivamente a funzionare per un lungo periodo si è poi passato a testare tutte le sue altre funzioni: si è iniziato dal collaudo delle funzioni che calcolano le telemetrie, andando a verificare effettivamente che le telemetrie fossero rispondenti ai valori letti e che le telemetrie estese fossero calcolate ed inviate correttamente. È stato anche controllato se il numero di pacchetti di telemetria di housekeeping utilizzati per il calcolo della telemetria estesa fosse corretto e che il comando di reset telemetria cancellasse effettivamente i pacchetti salvati e facesse iniziare nuovamente l'acquisizione ed il calcolo dei nuovi valori.

A questo punto si è passati al collaudo funzionale del motore elettrico, verificando che esso compiesse un numero di giri proporzionale a quello programmato. Chiaramente non si è potuto andare effettivamente a contare il numero giri in quanto questi test sono stati effettuati con l'intero satellite assemblato, ma questo test era già stato svolto durante il collaudo del software della scheda ProcB.

Le ultime funzionalità verificate sono state quelle legate alla scheda Payload: come prima cosa è stato necessario provare ad effettuare lo scatto delle fotografie per poter le poi trasferire via radio. Si è testato lo scatto di una fotografia da ciascuna telecamera e si è cercato di salvare le fotografia in ogni posizione della memoria. Per maggiore sicurezza è stata provata ogni combinazione possibile tra il numero della fotocamera e la posizione in memoria della fotografia per poter escludere la presenza di errori nel software.

Dopo aver verificato da terra (tramite la coppia di pacchetti di telemetria di houekeeping inviati dal satellite durante lo scatto foto) che lo scatto fosse stato effettivamente eseguito si è passati a provare il trasferimento delle fotografie. Questa parte del collaudo è stata quella che ha occupato la porzione maggiore di tempo in quanto il trasferimento di una fotografia completa richiede alcuni minuti (a seconda del soggetto fotografato) per via della divisione in blocchi della foto. Durante questa fase del collaudo sono state anche apportate alcune modifiche ai software della scheda Payload, ProcB e della stazione di terra per ottimizzare il trasferimento dei dati.

Conclusa la fase di test in laboratorio sono stati effettuati alcuni test di comunicazione a distanze maggiori allo scopo di collaudare la stazione di terra fissa.

La prima prova è stata realizzata sul tetto del Politecnico (distanza tra il satellite e l'antenna di circa 10 metri) ed ha dato esiti positivi: il satellite è stato infatti in grado di svolgere tutte le funzioni correttamente. Si è anche provato ad introdurre un'attenuazione nella catena di trasmissione per simulare una distanza di circa 2000 km (quella massima di funzionamento) ma il test è fallito per problemi ambientali: il segnale del satellite è stato ricevuto perfettamente ma il ricevitore del satellite non era in grado di ricevere i telecomandi perché il segnale trasmesso (attenuato per simulare la maggiore distanza) era troppo debole in rapporto al rumore elettromagnetico di fondo presente in città. Questo problema non sarà assolutamente presente durante la vita operativa del satellite, perciò non ha destato preoccupazione.

Sono stati condotti poi ulteriori test da distanze maggiori (dal colle della Maddalena, sulla collina torinese) per verificare ulteriormente le funzionalità del satellite. Questi test hanno risentito in maniera più marcata del problema del rumore elettromagnetico di fondo in quanto nella zona del test sono presenti alcuni ripetitori del segnale radiotelevisivo.

Nonostante tutto il satellite è stato in grado di comunicare con la stazione di terra sul tetto del Politecnico ed eseguire i comandi trasmessi: oltre al test completo delle funzioni del satellite è stata anche scattata e trasferita via radio con successo una foto della collina torinese.

Capitolo 7

Conclusioni

Il lavoro svolto durante la tesi si è concentrato sulla realizzazione di tutti gli apparati necessari al collaudo ed all'integrazione di un sistema complesso quale è sicuramente un satellite. Le problematiche affrontate erano legate ad ambiti completamente differenti, da semplici problemi di natura elettrica a quelli di intercomunicazione tra schede, da problemi di modulazione a radio frequenza alla corretta carica delle batterie di PiCPoT, fino alla preparazione del viaggio a Baikonur per l'integrazione del satellite.

Tutto il lavoro svolto mi ha fornito una più approfondita capacità di analisi e valutazione di sistemi complessi che mi sarà estremamente utile in futuro quando potrebbe essere necessario sviluppare da zero un progetto.

Durante questi mesi ho messo in pratica molte delle nozioni apprese ed ho avuto la possibilità di lavorare in un team composto da persone con competenze in vari campi, costituito da studenti e professori del Dipartimento di Elettronica del Politecnico di Torino. Il gruppo ha lavorato sempre in armonia rendendo meno faticoso il duro lavoro svolto anche con orari inconsueti.

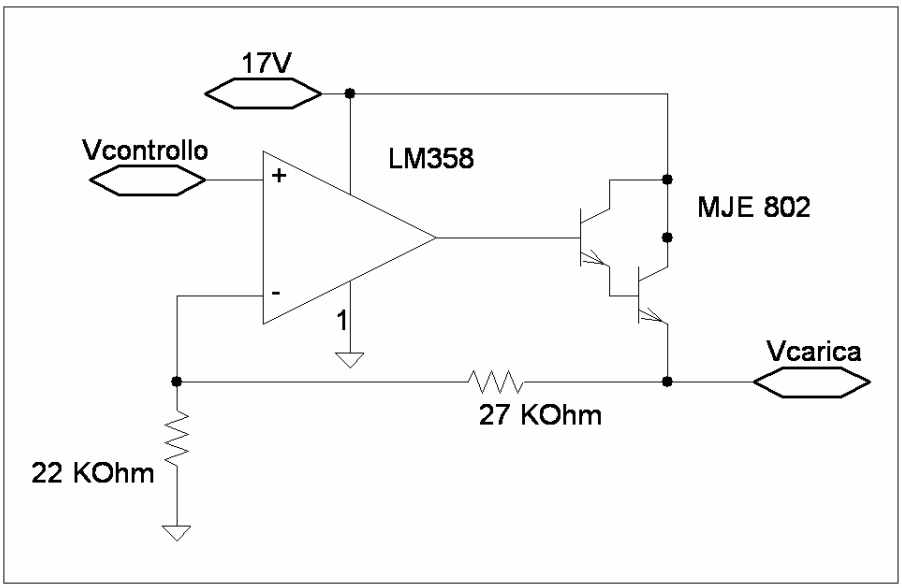
Il progetto ha una grande valenza didattica, ma ha anche una grande valenza scientifica. Tale valore è testimoniato dalla partecipazione allo Student Space Science and Tecnology Symposium 2006 e ad ESA YES3 Workshop a Patrasso in Grecia, in cui è stata tenuta una presentazione del progetto PiCPoT di fronte a studenti e ricercatori provenienti da tutto il mondo.

Purtroppo tutto il lavoro svolto non ha potuto essere collaudato effettivamente nello spazio, visti i problemi tecnici accorsi al razzo vettore il giorno del lancio, ma si è comunque fiduciosi delle possibilità di PiCPoT e si attende di poterlo finalmente portare nel suo ambiente: lo spazio.

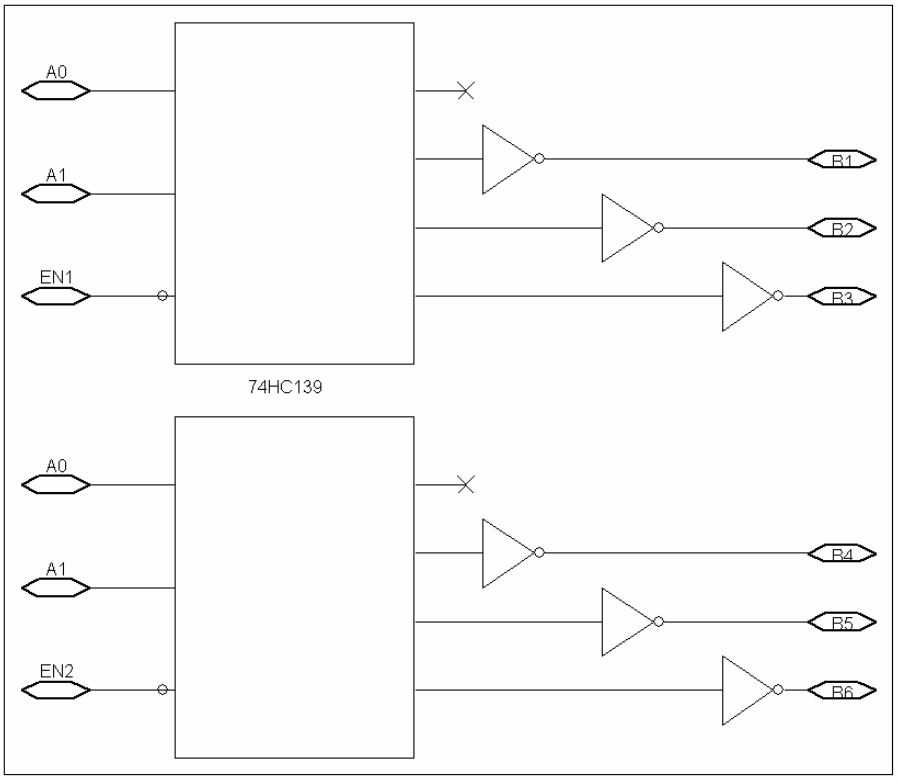
Appendice A

Gli schemi elettrici

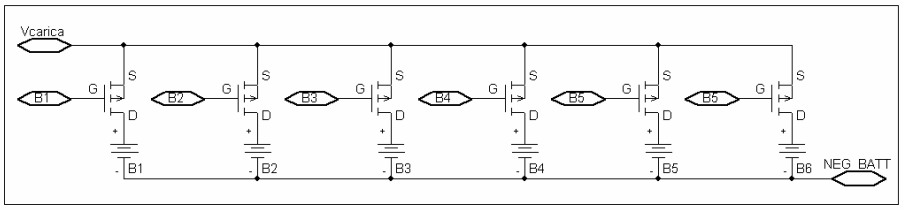
Gli schemi elettrici sono riportati in ordine di comparizione.



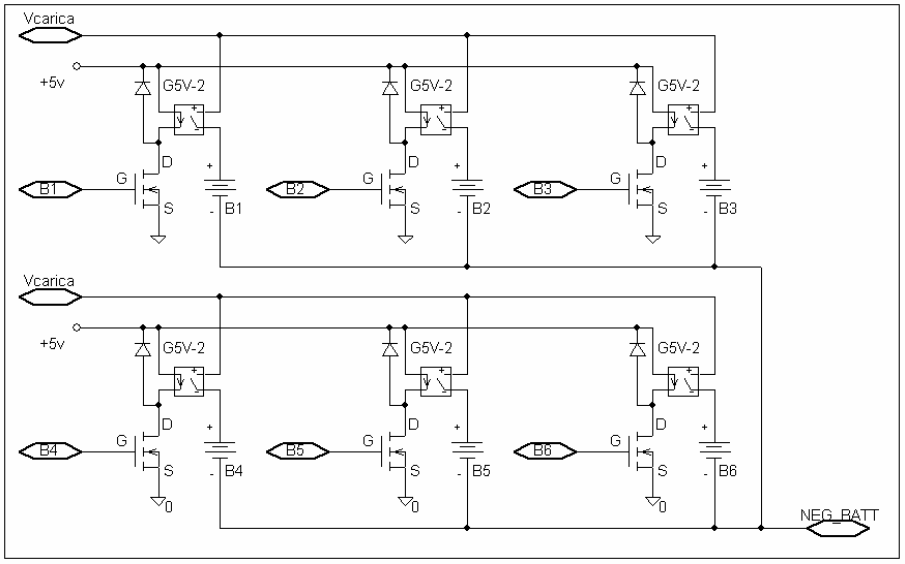
Circuito di amplificazione del segnale di controllo del caricabatterie



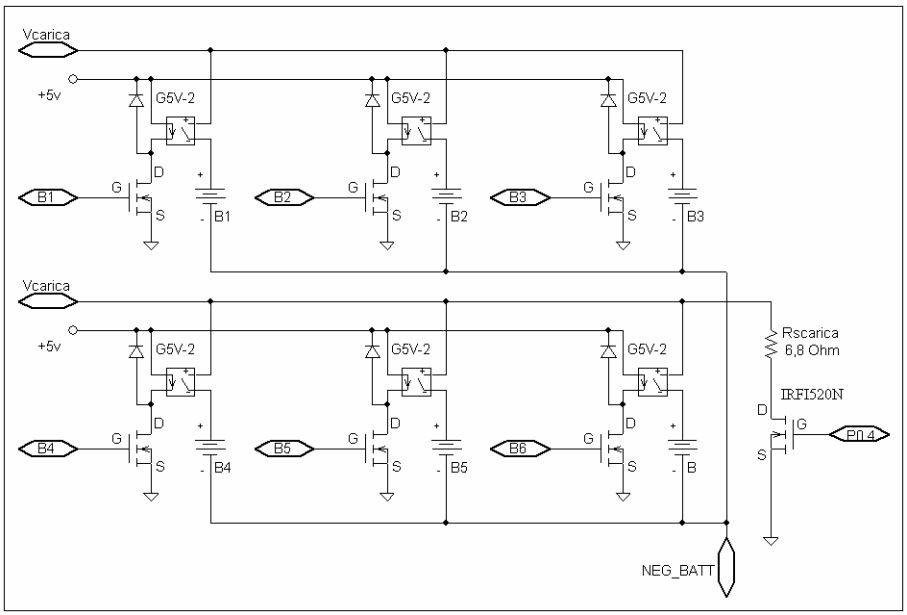
Schema di collegamento del decoder digitale 74HC139 installato sul caricabatterie



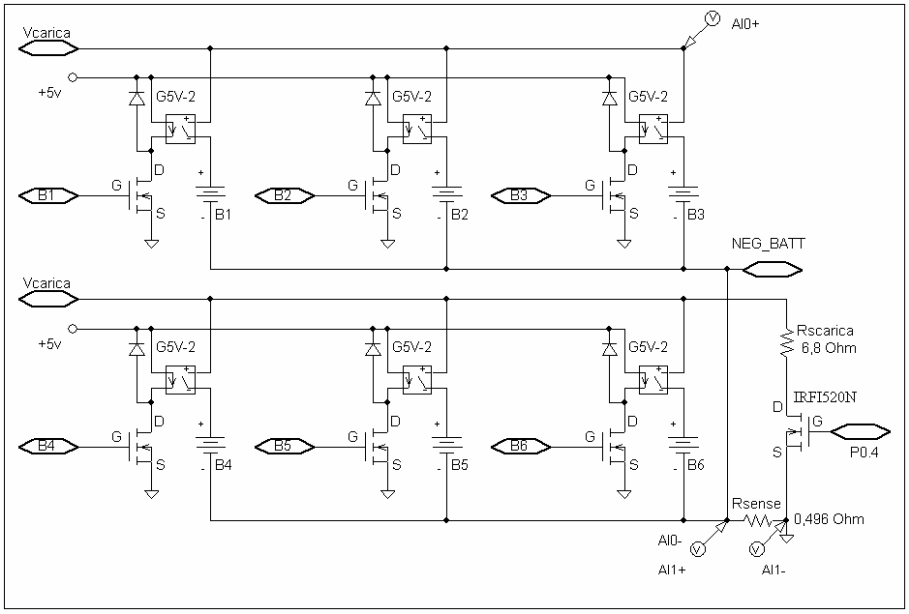
Schema circuitale della rete di selezione della batteria realizzate con i soli transistori P-MOS



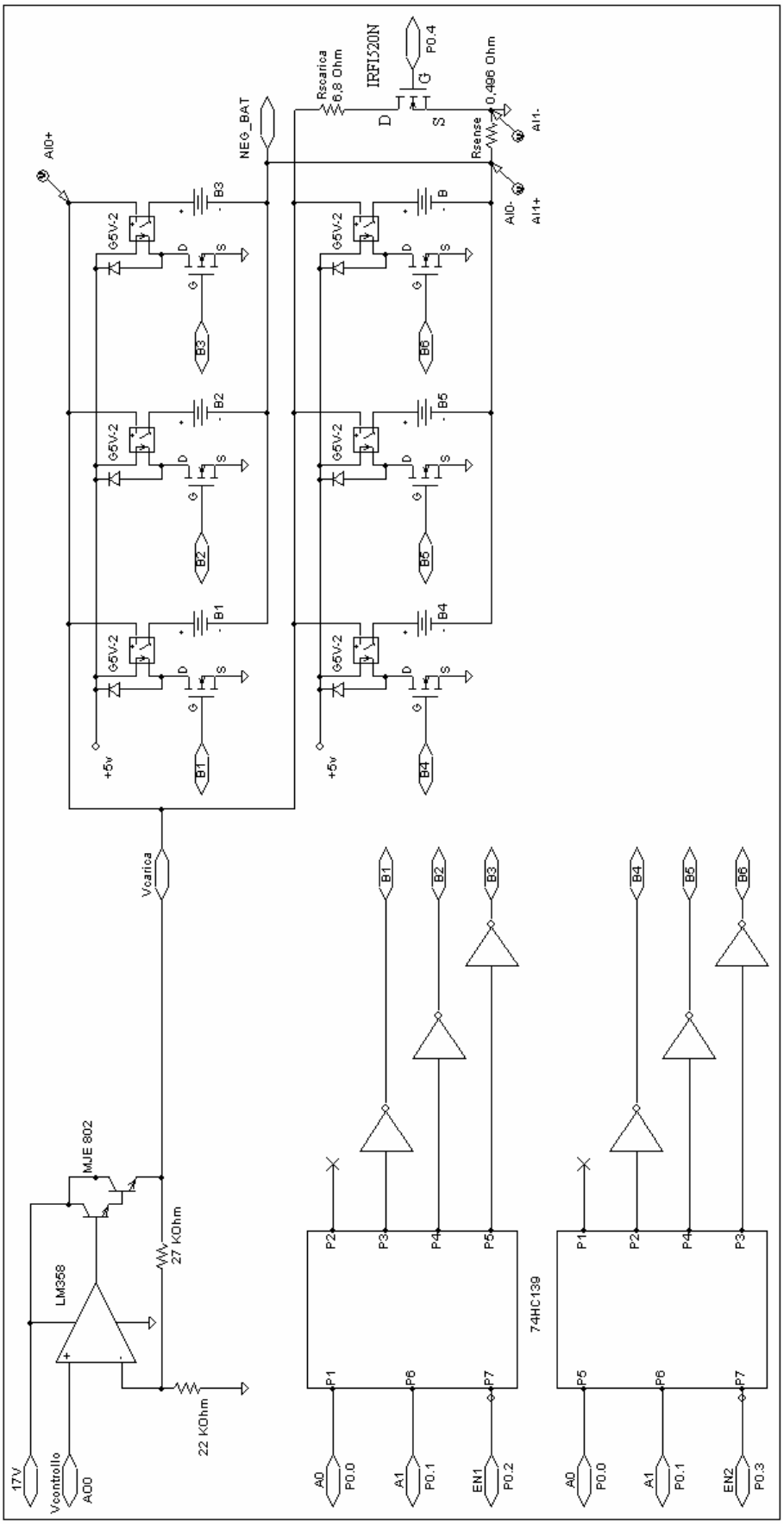
Schema circuitale della rete di selezione della batteria realizzata con i relè oltre che con i transistori N-MOS



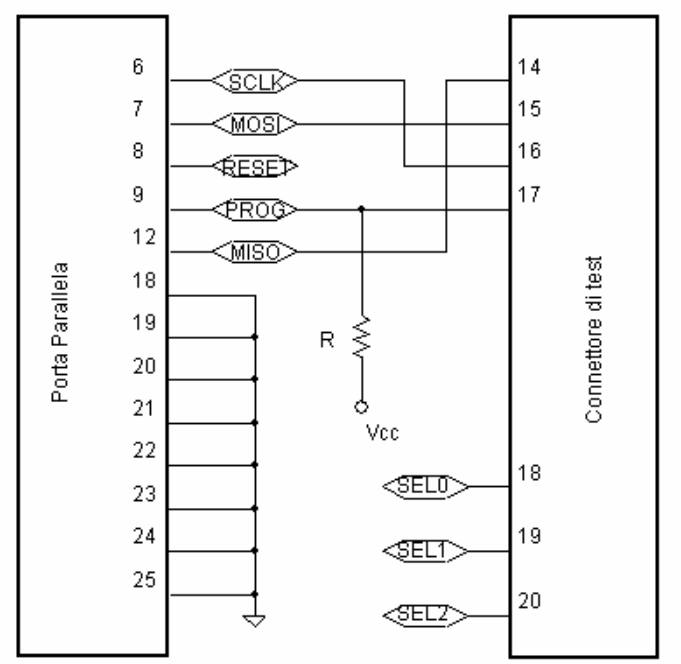
Circuito di selezione con annessa rete di scarica



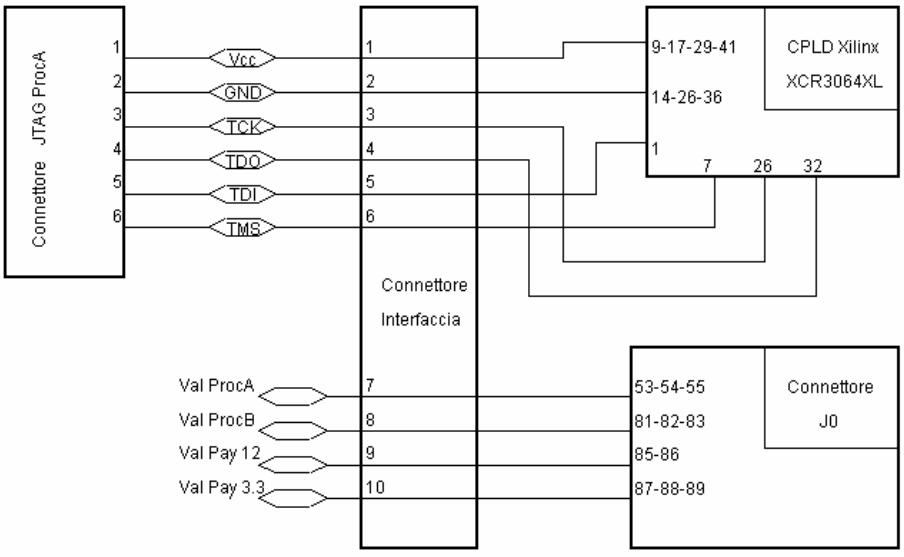
Rete di selezione con resistore di "sense" e punti di misura



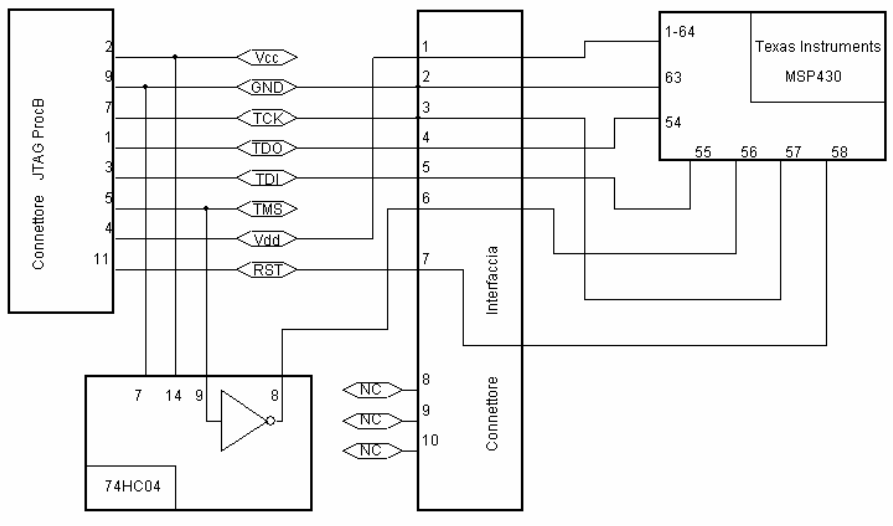
Schema elettrico completo del caricabatteria



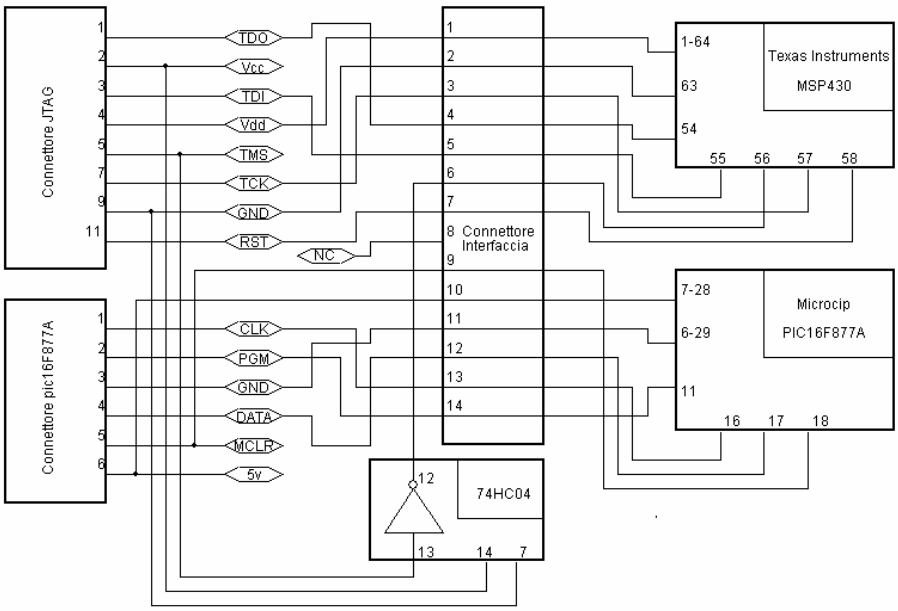
Connessioni utilizzate per la programmazione dell'unità Chipcon CC1010 su ProcA



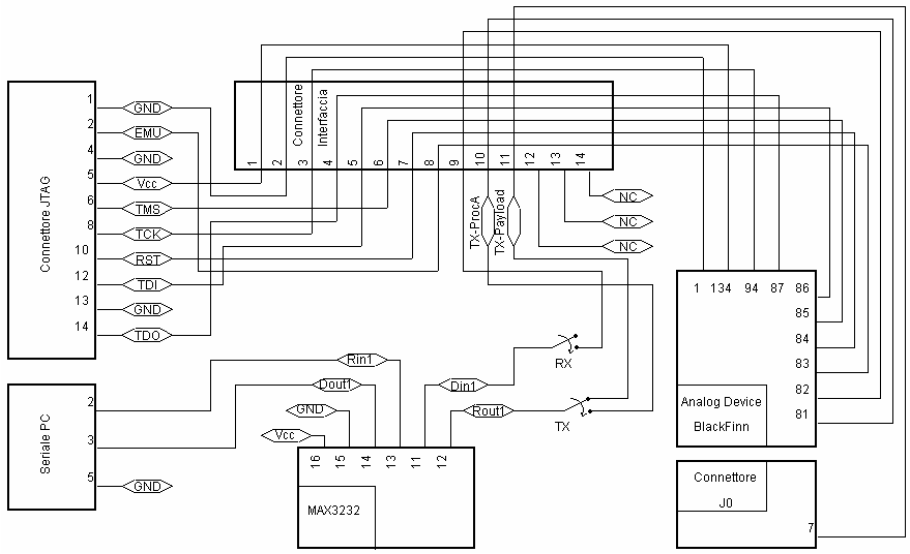
Schema di collegamento tra JTAG e ProcA



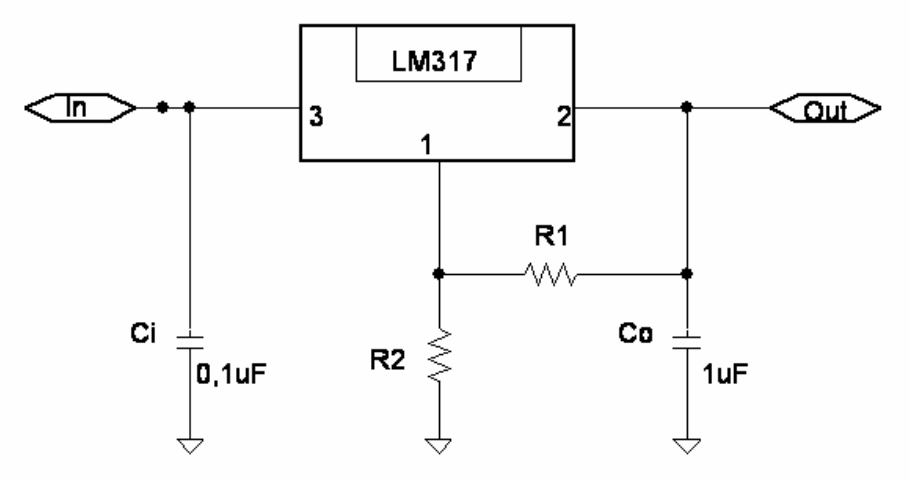
Collegamento tra JTAG e ProcB



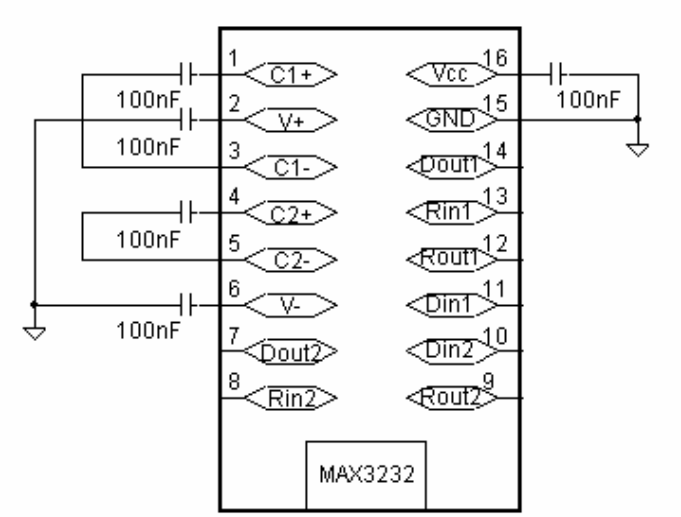
Schema di collegamento tra PowerSwitch ed i due programmatori



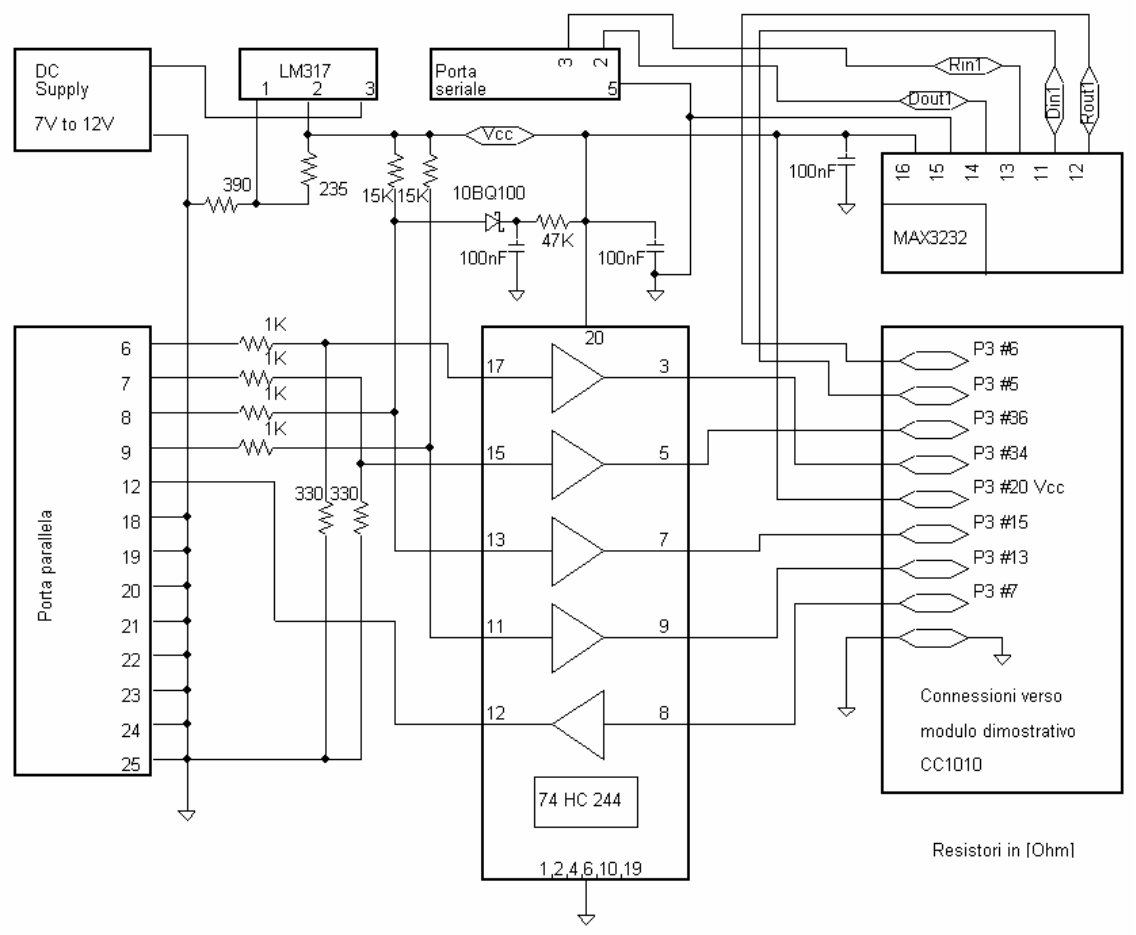
Schema delle connessioni necessarie per la programmzione della scheda Payload



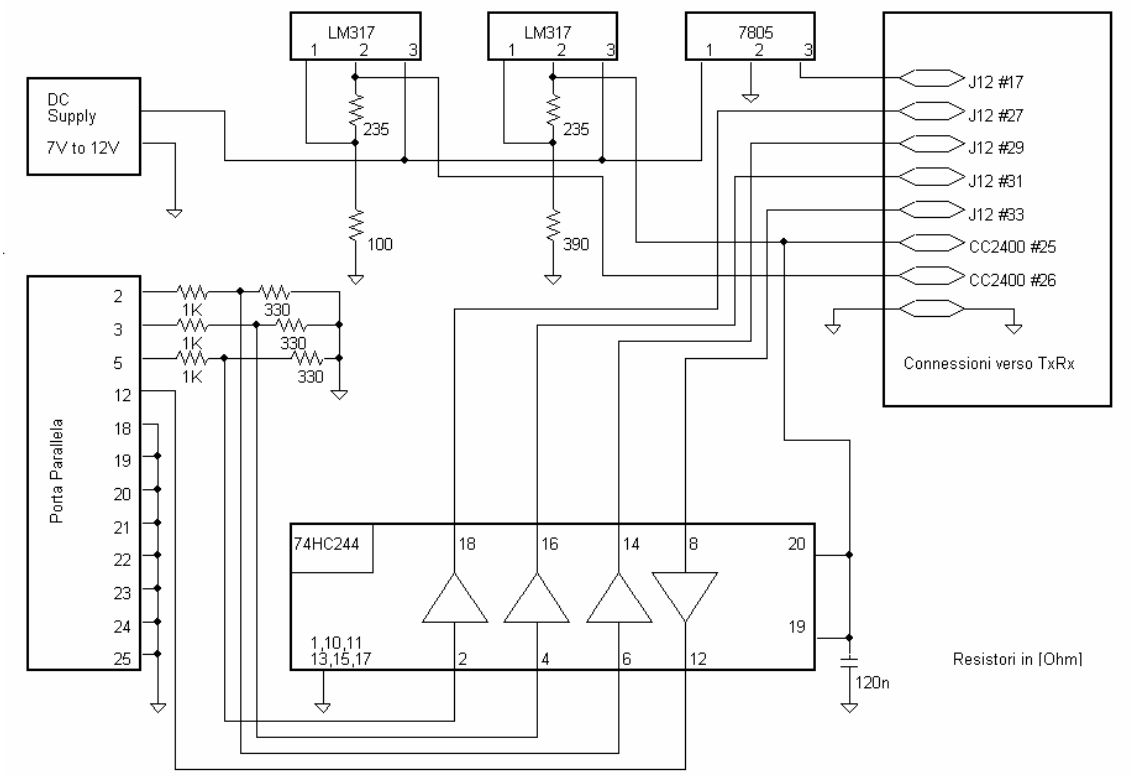
Schema circuitale dei regolatori di tensione



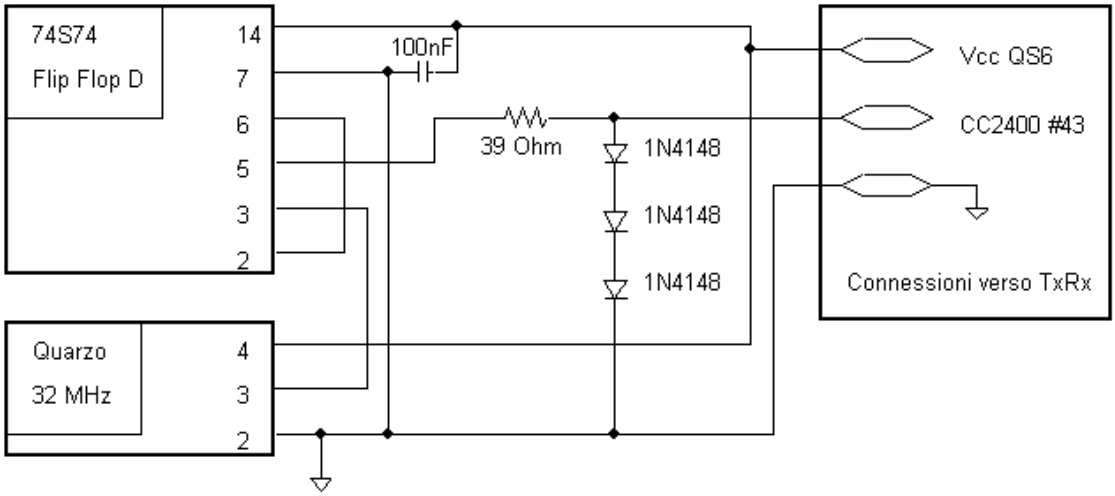
Schema di collegamento del MAX3232 necessario per un corretto funzionamento del dispositivo



Schema elettrico dell'interfaccia per le comunicazioni a 437 MHz



Schema di collegamento per il canale a 2,44 GHz



Schema di collegamento dell'oscillatore al quarzo

Appendice B

Fogli tecnici

I fogli tecnici dei principali componenti sono riportati in ordine di comparizione.

Rechargeable
Lithium Polymer

ULTRALIFE® Batteries

We. Are. Power.™

UBC425085
Technical Datasheet

The Ultralife Advantage
Better technology. Our lithium-based (lithium-manganese dioxide, lithium ion and lithium polymer) technologies enable us to design leading-edge power solutions for the world's most demanding applications.

FEATURES
- Thin
- High energy density
- Wide operating temperature range
- Lightweight
- No memory effect
- Can be assembled into packs

APPLICATIONS
- Portable Electronics
- Medical Devices
- RFID Applications
- Tracking Applications

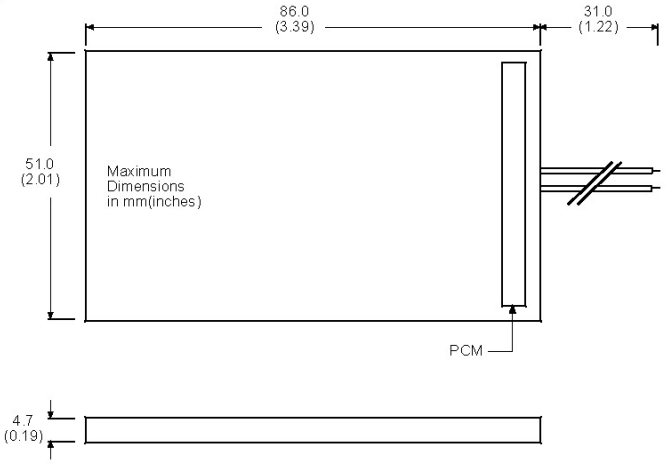
SPECIFICATIONS

Part No	UBC002
Voltage Range	3.0 to 4.2 V
Average Voltage	3.7 V
Nominal Capacity	1.6 Ah @ C/5 Rate @ 23° C
Max. Discharge	2.4 A
Energy	5.9 Wh
Energy Density	156 Wh/kg, 332 Wh/l
Weight	38 grams
Cycle Life	> 300 cycles @ C/5 to 80% of initial capacity
Memory	No Memory Effect
Operating Temp	-20° C to 60° C
Storage Temp	-20° C to 60° C
Self-Discharge	< 10% per month
Exterior/Housing	Laminated Foil
Terminals/Connector	26 AWG Wire: Red (+), Black (-)
Safety	Material Safety Datasheet – MSDS014.
Transportation	Excepted from Regulations – see note 1
Protection Circuit Module	Over Voltage Limit: 4.275 +/- 0.03 V Under Voltage Limit: 2.30 +/- 0.07 V Over Current Protection: 2.4 A Max. Quiescent Drain: 10 µA
Charging	Maximum charge rate at C/2 to 4.2 Volts in a temperature range of 0° to 45° C. Hold at 4.2 Volts until current declines to C/10. Refer also to Safety Guide UBI-5112.

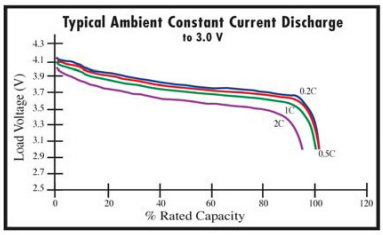
Note 1	For a complete description of transportation regulations and definitions of the transportation classifications "Excepted" and "Class 9," refer to the Ultralife web site at www.ultralifebatteries.com.

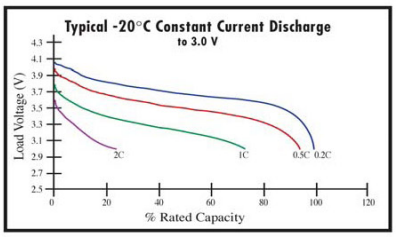
Newark, NY • 315-332-7100 • Fax 315-331-7800 / Abingdon, England • +44 (0) 1235 542600 • +44 (0) 1235 535766

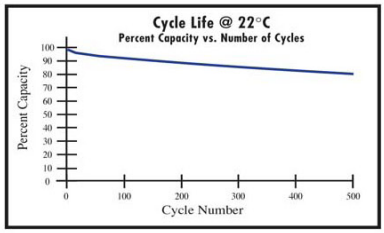
DIMENSIONS

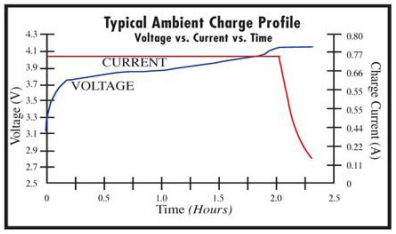


PERFORMANCE GRAPHS







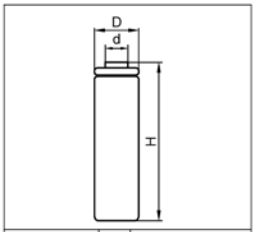


SANYO

Cadnica

Cell Type KR-1100AAU

Specifications

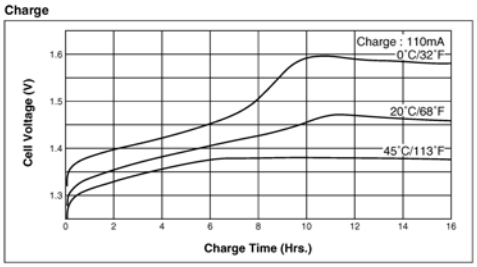


Dimensions of Bare Cell	H	49.5 ±0.3	mm
		1.949 ±0.012	inch
	D	13.8 ±0.2	mm
		0.543 ±0.008	inch
	d	7.0	mm
		0.276	inch

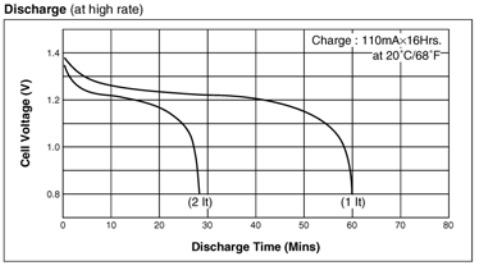
Nominal Capacity			1100mAh
Nominal Voltage			1.2V
Charging Current		Standard	110mA
		Quick	220mA
		Fast	1600mA
Charging Time		Standard	14 to 16Hrs.
		Quick	7 to 8Hrs.
		Fast	about 1Hr.
Ambient Temperature	Charge	Standard	0°C to +45°C [+32°F to 113°F]
		Quick	0°C to +45°C [+32°F to 113°F]
		Fast	0°C to +45°C [+32°F to 113°F]
	Discharge		-20°C to +60°C [-4°F to 140°F]
	Storage		-30°C to +50°C [-22°F to 122°F]
Internal Impedance (Av.) (at. 50% discharge)			19.0mΩ (at 1000Hz)
Weight			24g/0.85oz
Dimensions(D)×(H) (with tube)			14.3 $^{0}_{-0.5}$ ×50.3 $^{0}_{-1}$ mm 0.56 $^{0}_{-0.02}$ ×1.98 $^{0}_{-0.04}$ inch

Typical Characteristics

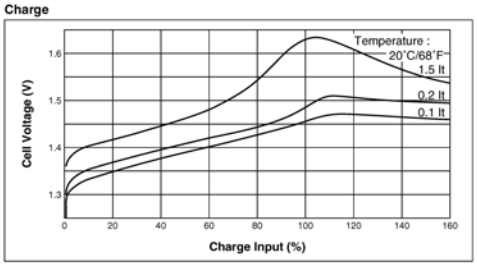
Charge



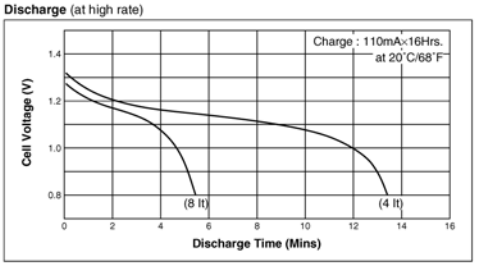
Discharge (at high rate)



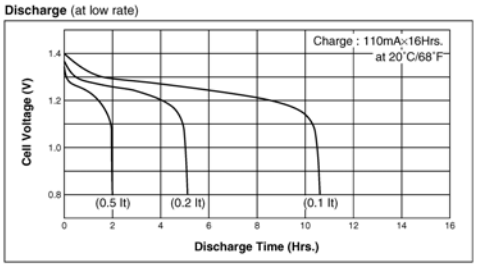
Charge



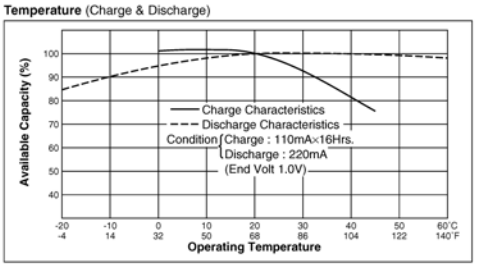
Discharge (at high rate)



Discharge (at low rate)



Temperature (Charge & Discharge)



CD010830

Low-Cost Multifunction DAQ for USB

NI USB-6008, NI USB-6009

- Small and portable
- 12 or 14-bit input resolution, at up to 48 kS/s
- Built-in, removable connectors for easier and more cost-effective connectivity
- 2 true DAC analog outputs for accurate output signals
- 12 digital I/O lines (TTL/LVTTL/CMOS)
- 32-bit event counter
- Student kits available
- OEM versions available

Operating Systems

- Windows 2000/XP
- Mac OS X[1]
- Linux®[1]
- Pocket PC
- Win CE

Recommended Software

- LabVIEW
- LabWindows/CVI

Measurement Services Software (included)

- NI-DAQmx
- Ready-to-run data logger

[1]Mac OS X and Linux users need to download NI-DAQmx Base.

Product	Bus	Analog Inputs[1]	Input Resolution (bits)	Max Sampling Rate (kS/s)	Input Range (V)	Analog Outputs	Output Resolution (bits)	Output Rate (Hz)	Output Range (V)	Digital I/O Lines	32-Bit Counter	Trigger
USB-6009	USB	8 SE/4 DI	14	48	±1 to ±20	2	12	150	0 to 5	12	1	Digital
USB-6008	USB	8 SE/4 DI	12	10	±1 to ±20	2	12	150	0 to 5	12	1	Digital

[1]SE = single ended, DI = differential

Hardware Description

The National Instruments USB-6008 and USB-6009 multifunction data acquisition (DAQ) modules provide reliable data acquisition at a low price. With plug-and-play USB connectivity, these modules are simple enough for quick measurements but versatile enough for more complex measurement applications.

Software Description

The NI USB-6008 and USB-6009 use NI-DAQmx high-performance, multithreaded driver software for interactive configuration and data acquisition on Windows OSs. All NI data acquisition devices shipped with NI-DAQmx also include VI Logger Lite, a configuration-based data-logging software package.

Mac OS X and Linux users can download NI-DAQmx Base, a multiplatform driver with a limited NI-DAQmx programming interface. You can use NI-DAQmx Base to develop customized data acquisition applications with National Instruments LabVIEW or C-based development environments. NI-DAQmx Base includes a ready-to-run data logger application that acquires and logs up to eight channels of analog data.

PDA users can download NI-DAQmx Base for Pocket PC and Win CE to develop customized handheld data acquisition applications.

Recommended Accessories

The USB-6008 and USB-6009 have removable screw terminals for easy signal connectivity. For extra flexibility when handling multiple wiring configurations, NI offers the USB-6008/09 Accessory Kit, which includes two extra sets of screw terminals, extra labels, and a screwdriver.

In addition, the USB-6008/09 Prototyping Accessory provides space for adding more circuitry to the inputs of the USB-6008 or USB-6009.

Common Applications

The USB-6008 and USB-6009 are ideal for a number of applications where economy, small size, and simplicity are essential, such as:

- Data logging – Log environmental or voltage data quickly and easily.
- Academic lab use – The low price facilitates student ownership of DAQ hardware for completely interactive lab-based courses. (Academic pricing available. Visit ni.com/academic for details.)
- Embedded OEM applications.

NATIONAL INSTRUMENTS

Low-Cost Multifunction DAQ for USB

Information for Student Ownership

To supplement simulation, measurement, and automation theory courses with practical experiments, NI has developed the USB-6008 and USB-6009 student kits, which include the LabVIEW Student Edition and a ready-to-run data logger application. These kits are exclusively for students, giving them a powerful, low-cost hands-on learning tool. Visit **ni.com/academic** for more details.

Information for OEM Customers

For information on special configurations and pricing, call (800) 813 3693 (U.S. only) or visit **ni.com/oem**. Go to the Ordering Information section for part numbers.

Ordering Information

NI USB-6008[1]	779051-01
NI USB-6009[1]	779026-01
NI USB-6008 OEM	193132-02
NI USB-6009 OEM	193132-01
NI USB-6008 Student Kit[1,2]	779320-22
NI USB-6009 Student Kit[1,2]	779321-22

[1] Includes NI-DAQmx software, NI ready-to-run data logger software, and a USB cable.

[2] Includes LabVIEW Student Edition.

BUY NOW!

For complete product specifications, pricing, and accessory information, call 800 265 9891 (U.S. only) or go to **ni.com/usb**.

BUY ONLINE at ni.com or **CALL (800) 813 3693 (U.S.)**

2

Specifications

Typical at 25 °C unless otherwise noted.

Analog Input

Absolute accuracy, single-ended

Range	Typical at 25 °C (mV)	Maximum (0 to 55 °C) (mV)
±10	14.7	138

Absolute accuracy at full scale, differential[1]

Range	Typical at 25 °C (mV)	Maximum (0 to 55 °C) (mV)
±20	14.7	138
±10	7.73	84.8
±5	4.28	58.4
±4	3.59	53.1
±2.5	2.56	45.1
±2	2.21	42.5
±1.25	1.70	38.9
±1	1.53	37.5

Number of channels ..... 8 single-ended/4 differential
Type of ADC ..... Successive approximation

ADC resolution (bits)

Module	Differential	Single-Ended
USB-6008	12	11
USB-6009	14	13

Maximum sampling rate (system dependent)

Module	Maximum Sampling Rate (kS/s)
USB-6008	10
USB-6009	48

Input range, single-ended ..... ±10 V
Input range, differential ..... ±20, ±10, ±5, ±4, ±2.5, ±2, ±1.25, ±1 V
Maximum working voltage ..... ±10 V
Overvoltage protection ..... ±35 V
FIFO buffer size ..... 512 B
Timing resolution ..... 41.67 ns (24 MHz timebase)
Timing accuracy ..... 100 ppm of actual sample rate
Input impedance ..... 144 k
Trigger source ..... Software or external digital trigger
System noise ..... 0.3 LSB_{rms} (±10 V range)

Analog Output

Absolute accuracy (no load) ..... 7 mV typical, 36.4 mV maximum at full scale
Number of channels 2
Type of DAC ..... Successive approximation
DAC resolution ..... 12 bits
Maximum update rate ..... 150 Hz, software-timed
Output range ..... 0 to +5 V
Output impedance ..... 50 Ω
Output current drive ..... 5 mA
Power-on state ..... 0 V
Slew rate ..... 1 V/µs
Short-circuit current ..... 50 mA

Digital I/O

Number of channels ..... 12 total
8 (P0.<0..7>)
4 (P1.<0..3>)
Direction control ..... Each channel individually programmable as input or output
Output driver type
USB-6008 ..... Open-drain
USB-6009 ..... Each channel individually programmable as push-pull or open-drain
Compatibility ..... CMOS, TTL, LVTTL
Internal pull-up resistor ..... 4.7 kΩ to +5 V
Power-on state ..... Input (high impedance)
Absolute maximum voltage range ..... -0.5 to +5.8 V

Digital logic levels

Level	Min	Max	Units
Input low voltage	-0.3	0.8	V
Input high voltage	2.0	5.8	V
Input leakage current	–	50	µA
Output low voltage (I = 8.5 mA)	–	0.8	V
Output high voltage (push-pull, I = -8.5 mA)	2.0	3.5	V
Output high voltage (open-drain, I = -0.6 mA, nominal)	2.0	5.0	V
Output high voltage (open-drain, I = -8.5 mA, with external pull-up resistor)	2.0	–	V

Counter

Number of counters 1
Resolution ..... 32 bits
Counter measurements ..... Edge counting (falling edge)
Pull-up resistor ..... 4.7 kΩ to 5 V
Maximum input frequency ..... 5 MHz
Minimum high pulse width ..... 100 ns
Minimum low pulse width ..... 100 ns
Input high voltage ..... 2.0 V
Input low voltage ..... 0.8 V

Power available at I/O connector

+5 V output (200 mA maximum) ..... +5 V typical
+4.85 V minimum
+2.5 V output (1 mA maximum) ..... +2.5 V typical
+2.5 V output accuracy ..... 0.25% max
Voltage reference temperature drift ... 50 ppm/°C max

[1]Input voltages may not exceed the working voltage range.

Low-Cost Multifunction DAQ for USB

Physical Characteristics

If you need to clean the module, wipe it with a dry towel.

Dimensions (without connectors)	6.35 by 8.51 by 2.31 cm (2.50 by 3.35 by 0.91 in.)
Dimensions (with connectors)	8.18 by 8.51 by 2.31 cm (3.22 by 3.35 by 0.91 in.)
Weight (without connectors)	59 g (2.1 oz)
Weight (with connectors)	84 g (3 oz)
I/O connectors	USB series B receptacle (2) 16-position (screw-terminal) plug headers
Screw-terminal wiring	16 to 28 AWG
Screw-terminal torque	0.22 to 0.25 N•m (2.0 to 2.2 lb•in.)

Power Requirement

USB (4.10 to 5.25 VDC)	80 mA typical 500 mA maximum
USB suspend	300 µA typical 500 µA maximum

Environmental

The USB-6008 and USB-6009 are intended for indoor use only.

Operating environment	
Ambient temperature range	0 to 55 °C (tested in accordance with IEC-60068-2-1 and IEC-60068-2-2)
Relative humidity range	10 to 90%, noncondensing (tested in accordance with IEC-60068-2-56)
Storage environment	
Ambient temperature range	-40 to 85 °C (tested in accordance with IEC-60068-2-1 and IEC-60068-2-2)
Relative humidity range	5 to 90%, noncondensing (tested in accordance with IEC-60068-2-56)
Maximum altitude	2,000 m (at 25 °C ambient temperature)
Pollution degree	2

Safety and Compliance

Safety

This product is designed to meet the requirements of the following standards of safety for electrical equipment for measurement, control, and laboratory use:

- IEC 61010-1, EN 61010-1
- UL 61010-1, CAN/CSA-C22.2 No. 61010-1

Note: For UL and other safety certifications, refer to the product label or visit **ni.com/certification**, search by model number or product line, and click the appropriate link in the Certification column.

Electromagnetic Compatibility

This product is designed to meet the requirements of the following standards of EMC for electrical equipment for measurement, control, and laboratory use:

- EN 61326 EMC requirements; Minimum Immunity
- EN 55011 Emissions; Group 1, Class A
- CE, C-Tick, ICES, and FCC Part 15 Emissions; Class A

Note: For EMC compliance, operate this device according to product documentation.

CE Compliance

This product meets the essential requirements of applicable European Directives, as amended for CE marking, as follows:

- 73/23/EEC; Low-Voltage Directive (safety)
- 89/336/EEC; Electromagnetic Compatibility Directive (EMC)

Note: Refer to the Declaration of Conformity (DoC) for this product for any additional regulatory compliance information. To obtain the DoC for this product, visit **ni.com/certification**, search by model number or product line, and click the appropriate link in the Certification column.

Waste Electrical and Electronic Equipment (WEEE)

EU Customers: At the end of their life cycle, all products must be sent to a WEEE recycling center. For more information about WEEE recycling centers and National Instruments WEEE initiatives, visit **ni.com/environment/weee.htm**.

NI Services and Support

NI has the services and support to meet your needs around the globe and through the application life cycle – from planning and development through deployment and ongoing maintenance. We offer services and service levels to meet customer requirements in research, design, validation, and manufacturing. Visit **ni.com/services**.

Training and Certification

NI training is the fastest, most certain route to productivity with our products. NI training can shorten your learning curve, save development time, and reduce maintenance costs over the application life cycle. We schedule instructor-led courses in cities worldwide, or we can hold a course at your facility. We also offer a professional certification program that identifies individuals who have high levels of skill and knowledge on using NI products. Visit **ni.com/training**.

Professional Services

Our Professional Services Team is comprised of NI applications engineers, NI Consulting Services, and a worldwide National Instruments Alliance Partner program of more than 600 independent consultants and integrators. Services range from start-up assistance to turnkey system integration. Visit **ni.com/alliance**.

NATIONAL INSTRUMENTS
Certified Alliance Partner

OEM Support

We offer design-in consulting and product integration assistance if you want to use our products for OEM applications. For information about special pricing and services for OEM customers, visit **ni.com/oem**.

Local Sales and Technical Support

In offices worldwide, our staff is local to the country, giving you access to engineers who speak your language. NI delivers industry-leading technical support through online knowledge bases, our applications engineers, and access to 14,000 measurement and automation professionals within NI Developer Exchange forums. Find immediate answers to your questions at **ni.com/support**.

We also offer service programs that provide automatic upgrades to your application development environment and higher levels of technical support. Visit **ni.com/ssp**.

Hardware Services

NI Factory Installation Services

NI Factory Installation Services (FIS) is the fastest and easiest way to use your PXI or PXI/SCXI combination systems right out of the box. Trained NI technicians install the software and hardware and configure the system to your specifications. NI extends the standard warranty by one year on hardware components (controllers, chassis, modules) purchased with FIS. To use FIS, simply configure your system online with **ni.com/pxiadvisor**.

Calibration Services

NI recognizes the need to maintain properly calibrated devices for high-accuracy measurements. We provide manual calibration procedures, services to recalibrate your products, and automated calibration software specifically designed for use by metrology laboratories. Visit **ni.com/calibration**.

Repair and Extended Warranty

NI provides complete repair services for our products. Express repair and advance replacement services are also available. We offer extended warranties to help you meet project life-cycle requirements. Visit **ni.com/services**.

NATIONAL INSTRUMENTS

ni.com • (800) 813 3693
National Instruments • info@ni.com

351378A-01 2006-7322-301-101-D

OMRON

PCB Relay — G5V-2

Miniature Relay for Signal Circuits

- Wide switching power of 10 μA to 2 A.
- High dielectric strength coil-contacts:1,000 VAC; open contacts: 750 VAC.
- Conforms to FCC Part 68 requirements.
- Ag + Au clad bifurcated crossbar contacts and fully sealed for high contact reliability.
- New 150-mW relays with high-sensitivity.

RC **FCC**

Ordering Information

Classification	Contact form	Contact type	Contact material	Enclosure ratings	Model
Standard	DPDT	Bifurcated crossbar	Ag + Au-clad	Fully sealed	G5V-2
High-sensitivity					G5V-2-H1

Note: When ordering, add the rated coil voltage to the model number.
Example: G5V-2 12 VDC
Rated coil voltage

Model Number Legend

G5V - □ - □ □ VDC
1 2 3

1. **Contact Form**
   2: DPDT
2. **Classification**
   H1: High-sensitivity
3. **Rated Coil Voltage**
   3, 5, 6, 9, 12, 24, 48 VDC

Specifications

■ Coil Ratings

Standard Models

Rated voltage		3 VDC	5 VDC	6 VDC	9 VDC	12 VDC	24 VDC	48 VDC
Rated current		166.7 mA	100 mA	83.3 mA	55.6 mA	41.7 mA	20.8 mA	12 mA
Coil resistance		18 Ω	50 Ω	72 Ω	162 Ω	288 Ω	1,152 Ω	4,000 Ω
Coil inductance	**Armature OFF**	0.04	0.09	0.16	0.31	0.47	1.98	7.23
(H) (ref. value)	**Armature ON**	0.05	0.11	0.19	0.49	0.74	2.63	10.00
Must operate voltage		75% max. of rated voltage						
Must release voltage		5% min. of rated voltage						
Max. voltage		120% of rated voltage at 23°C						
Power consumption		Approx. 500 mW						Approx. 580 mW

High Sensitivity Models

Rated voltage		3 VDC	5 VDC	6 VDC	9 VDC	12 VDC	24 VDC	48 VDC
Rated current		50 mA	30 mA	25 mA	16.7 mA	12.5 mA	8.33 mA	6.25 mA
Coil resistance		60 Ω	166.7 Ω	240 Ω	540 Ω	960 Ω	2,880 Ω	7,680 Ω
Coil inductance	**Armature ON**	0.18	0.46	0.70	1.67	2.90	6.72	20.1
(H) (ref. value)	**Armature OFF**	0.57	0.71	0.97	2.33	3.99	9.27	26.7
Must operate voltage		75% max. of rated voltage						
Must release voltage		5% min. of rated voltage						
Max. voltage		180% of rated voltage at 23°C						150% of rated voltage at 23°C
Power consumption		Approx. 150 mW					Approx. 200 mW	Approx. 300 mW

Note: 1. The rated current and coil resistance are measured at a coil temperature of 23°C with a tolerance of ±10%.
2. Operating characteristics are measured at a coil temperature of 23°C.

■ Contact Ratings

Item	Standard models	High sensitivity models
Load	Resistive load ($\cos\phi = 1$)	
Rated load	0.5 A at 125 VAC; 2 A at 30 VDC	0.5 A at 125 VAC; 1 A at 24 VDC
Contact material	Ag + Au-clad	
Rated carry current	2 A	
Max. switching voltage	125 VAC, 125 VDC	
Max. switching current	2 A	1 A
Max. switching power	62.5 VA, 60 W	62.5 VA, 24 W
Min. permissible load	0.01 mA at 10 mVDC	

Note: P level: $\lambda_{60} = 0.1 \times 10^{-6}$/operation

■ Characteristics

Item	Standard models	High sensitivity models
Contact resistance	50 mΩ max.	100 mΩ max.
Operate time	7 ms max.	
Release time	3 ms max.	
Bounce time	Operate: approx. 0.3 ms Release: approx. 1.5 ms	
Max. operating frequency	Mechanical: 36,000 operations/hr Electrical: 1,800 operations/hr (under rated load)	
Insulation resistance	1,000 MΩ min. (at 500 VDC)	
Dielectric strength	1,000 VAC, 50/60 Hz for 1 min between coil and contacts 1,000 VAC, 50/60 Hz for 1 min between contacts of different polarity 750 VAC, 50/60 Hz for 1 min between contacts of same polarity	1,000 VAC, 50/60 Hz for 1 min between coil and contacts 1,000 VAC, 50/60 Hz for 1 min between contacts of different polarity 500 VAC, 50/60 Hz for 1 min between contacts of same polarity
Impulse withstand voltage	1,500 V (10 x 160 μs) between coil and contacts (conforms to FCC Part 68)	
Vibration resistance	Destruction: 10 to 55 Hz, 1.5-mm double amplitude Malfunction: 10 to 55 Hz, 1.5-mm double amplitude	
Shock resistance	Destruction: 1,000 m/s^2 (approx. 100G) Malfunction: 200 m/s^2 (approx. 20G)	Destruction: 1,000 m/s^2 (approx. 100G) Malfunction: 100 m/s^2 (approx. 10G)
Life expectancy	Mechanical: 15,000,000 operations min. (at 36,000 operations/hr) Electrical: 100,000 operations min. (at 1,800 operations/hr)	
Ambient temperature	Operating: −25°C to 65°C (with no icing) Storage: −25°C to 65°C (with no icing)	Operating: −25°C to 70°C (with no icing) Storage: −40°C to 70°C (with no icing)
Ambient humidity	Operating: 35% to 85%	
Weight	Approx. 5 g	

■ Approved Standards

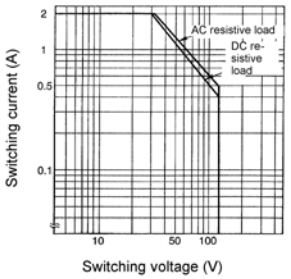
UL478, UL1950, UL508 (File No. E41515)/CSA C22.2 No.0, No.14 (File No. LR24825)

Contact form	Coil ratings	Contact ratings	
		G5V-2	G5V-2-H1
DPDT	3 to 48 VDC	0.6 A, 125 VAC (general use) 0.6 A, 110 VDC (resistive load) 2 A, 30 VDC (resistive load)	0.5 A, 125 VAC (general use) 0.2 A, 110 VDC (resistive load) 1 A, 24 VDC (resistive load)

Engineering Data

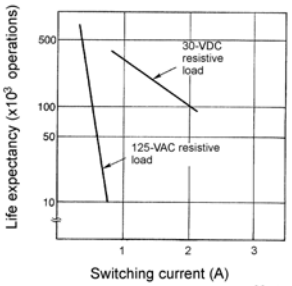
Maximum Switching Power

G5V-2



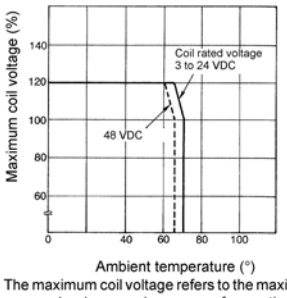
Life Expectancy

G5V-2



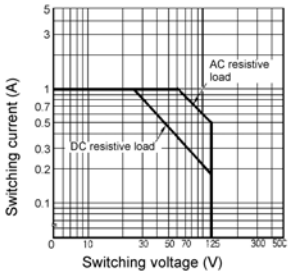
Ambient Temperature vs. Maximum Coil Voltage

G5V-2

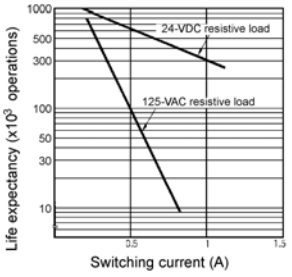


Note: The maximum coil voltage refers to the maximum value in a varying range of operating power voltage, not a continuous voltage.

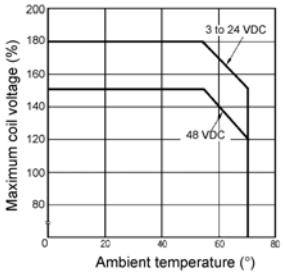
G5V-2-H1



G5V-2-H1



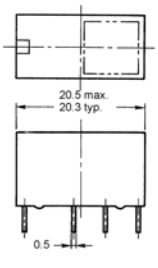
G5V-2-H1

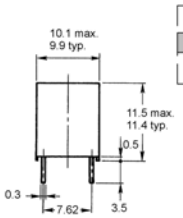


Note: The maximum coil voltage refers to the maximum value in a varying range of operating power voltage, not a continuous voltage.

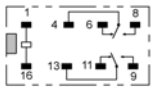
Dimensions

Note: 1. All units are in millimeters unless otherwise indicated.
2. Orientation marks are indicated as follows:



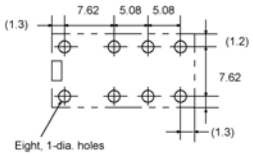


Terminal Arrangement/ Internal Connections (Bottom View)



Mounting Holes (Bottom View)

Tolerance: ±0.1



ALL DIMENSIONS SHOWN ARE IN MILLIMETERS.
To convert millimeters into inches, multiply by 0.03937. To convert grams into ounces, multiply by 0.03527.

Cat. No. K046-E1-2B

FAIRCHILD SEMICONDUCTOR®

March 2000
Revised June 2005

LM317
3-Terminal Positive Adjustable Regulator

General Description

This monolithic integrated circuit is an adjustable 3-terminal positive voltage regulator designed to supply more than 1.5A of load current with an output voltage adjustable over a 1.2 to 37V. It employs internal current limiting, thermal shut-down and safe area compensation.

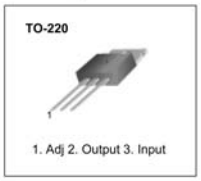
Features

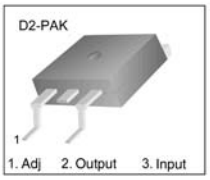
- Output Current In Excess of 1.5A
- Output Adjustable Between 1.2V and 37V
- Internal Thermal Overload Protection
- Internal Short Circuit Current Limiting
- Output Transistor Safe Operating Area Compensation
- TO-220 Package
- D2 PAK Package

Ordering Code:

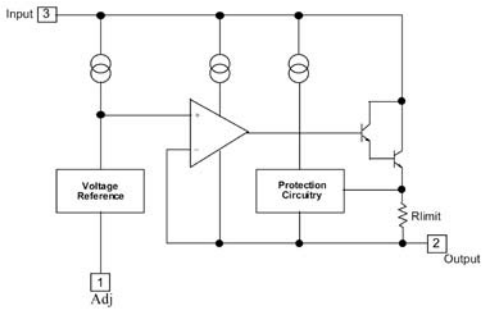
Product Number	Package	Operating Temperature
LM317T	TO-220	0°C to +125°C
LM317D2TXM	D2 PAK	0°C to +125°C

Connection Diagrams





Internal Block Diagram



 DS400281 www.fairchildsemi.com

Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Input-Output Voltage Differential	$V_I - V_O$	40	V
Lead Temperature	T_{LEAD}	230	°C
Power Dissipation	P_D	Internally limited	W
Operating Junction Temperature Range	T_J	0 ~ +125	°C
Storage Temperature Range	T_{STG}	−65 ~ +125	°C
Temperature Coefficient of Output Voltage	$\Delta V_O/\Delta T$	±0.02	% / °C

Note 1: Absolute Maximum Ratings: are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

Electrical Characteristic

($V_I - V_O$ = 5V, I_O = 0.5A, 0°C ≤ T_J ≤ + 125°C, I_{MAX} = 1.5A, P_{DMAX} = 20W, unless otherwise specified)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Line Regulation (Note 2)	Rline	T_A – +25°C 3V ≤ V_I – V_O ≤ 40V	—	0.01	0.04	% / V
		3V ≤ V_I – V_O ≤ 40V	–	0.02	0.07	% / V
Load Regulation (Note 2)	Rload	T_A – +25°c, 10mA ≤ I_O ≤ I_{MAX} V_O < 5V V_O ≥ 5V	–	18.0 0.4	25.0 0.5	mV% / V_O
		10mA ≤ I_O ≤ I_{MAX} V_O < 5V V_O ≥ 5V	–	40.0 0.8	70.0 1.5	mV% / V_O
Adjustable Pin Current	I_{ADJ}	–	–	46.0	100	µA
Adjustable Pin Current Change	ΔI_{ADJ}	3V ≤ V_I – V_O ≤ 40V 10mA ≤ I_O ≤ I_{MAX}, P_D ≤ P_{MAX}	–	2.0	5.0	µA
Reference Voltage	V_{REF}	3V ≤ V_{IN} - V_O ≤ 40V 10mA ≤ I_O ≤ I_{MAX} P_D ≤ P_{MAX}	1.20	1.25	1.30	V
Temperature Stability	ST_T	–	–	0.7	–	% / V_O
Minimum Load Current to Maintain Regulation	$I_{L(MIN)}$	V_I - V_O – 40V	–	3.5	12.0	mA
Maximum Output Current	$I_{O(MAX)}$	V_I - V_O ≤ 15V, P_D ≤ P_{MAX} V_I - V_O ≤ 40V, P_D ≤ P_{MAX} T_A – 25°C	1.0	2.2 0.3	–	A
RMS Noise,% of V_{OUT}	eN	T_A – +25°C, 10Hz ≤ f ≤ 10KHz	–	0.003	0.01	% / V_O
Ripple Rejection	RR	V_O – 10V, f – 120Hz without C_{ADJ} C_{ADJ} – 10µF (Note 3)	66.0	60.0 75.0	–	dB
Long-Term Stability, T_J – T_{HIGH}	ST	T_A – +25°C for end point measurements, 1000HR	–	0.3	1.0	%
Thermal Resistance Junction to Case	$R_{\theta JC}$	–	–	5.0	–	°C / W

Note 2: Load and line regulation are specified at constant junction temperature. Change in V_D due to heating effects must be taken into account separately. Pulse testing with low duty is used. (P_{MAX} – 20S)

Note 3: CADJ, when used, is connected between the adjustment pin and ground.

Typical Performance Characteristics

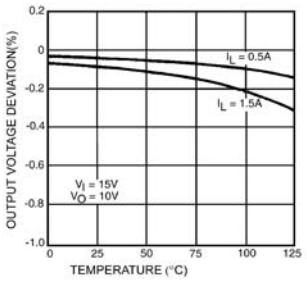


FIGURE 1. Load Regulation

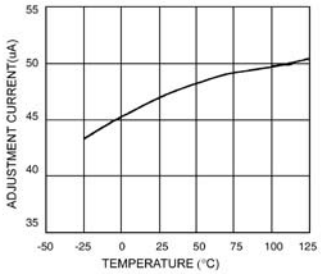


FIGURE 2. Adjustment Current

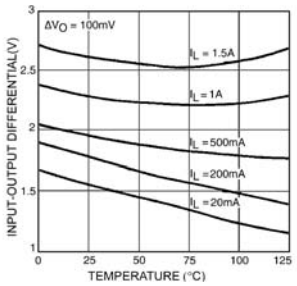


FIGURE 3. Dropout Voltage

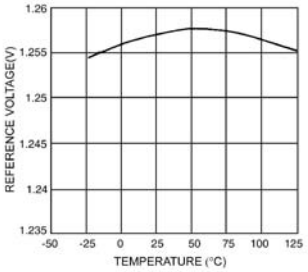
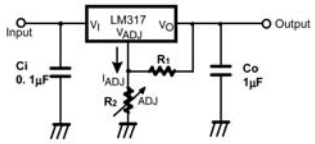


FIGURE 4. Reference Voltage

Typical Application



$$V_O = 1.25V\ (1 + R_2 / R_1) + I_{ADJ}\ R_2$$

Note: C_i is required when regulator is located an appreciable distance from power supply filter.

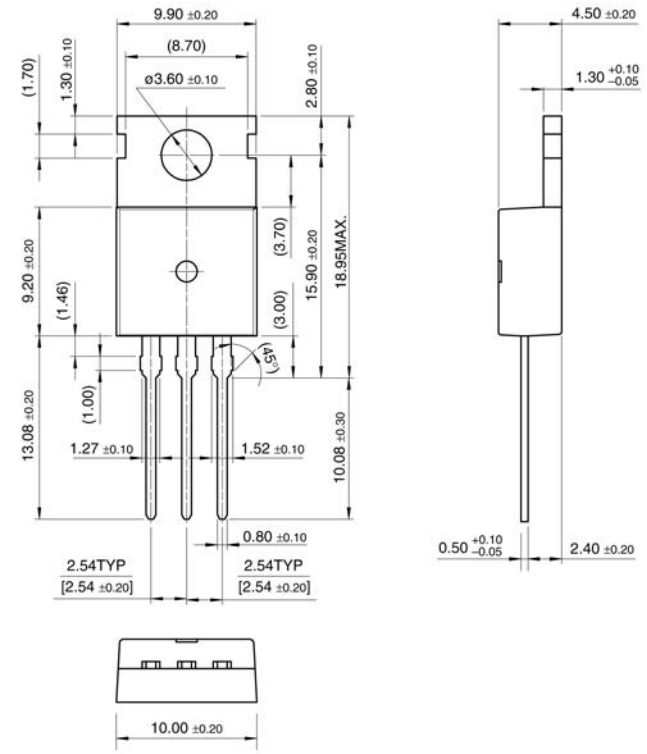
Note: C_o is not needed for stability, however, it does improve transient response.

Note: Since I_{ADJ} is controlled to less than 100µA, the error associated with this term is negligible in most applications.

FIGURE 5. Programmable Regulator

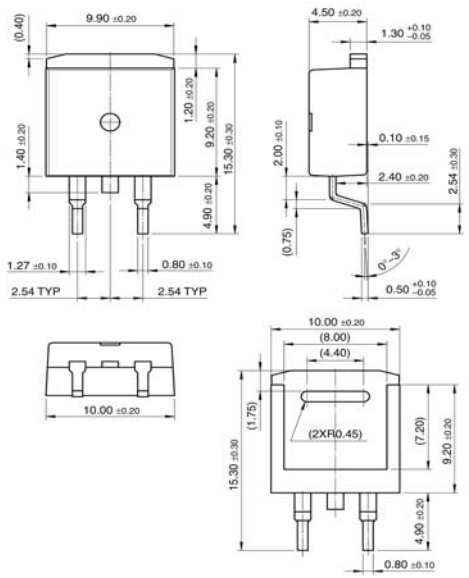
Physical Dimensions inches (millimeters) unless otherwise noted

TO-220



Physical Dimensions inches (millimeters) unless otherwise noted (Continued)

D^2-PAK



Dimensions in Millimeters

Appendice C

Il codice del programma di controllo del caricabatterie

```
#include <cvirte.h>
#include <userint.h>
#include "Caricabatterie.h"
#include <windows.h>
#include <ansi_c.h>
#include <NIDAQmx.h>

static int panelHandle;

TaskHandle vbatt, vcorr;
TaskHandle vout, p0out;
float64 vbatteria,      /*Tensione istantanea misurata*/
        vscheda,        /*Tensione di controllo*/
        vcorrente,      /*Tensione sulla resistenza di
                        sense
        ibatteria,      /* Corrente istantanea calcolata*/
        delta=0,        /* Caduta sul cavo */
        x1=0,           /* Variabile di supporto per la */
        x2=0,           /* ricerca del massimo */
        vi=0,           /*Prodotto Tensione Corrente
                        istantaneo */
        caricatot=0,    /*Energia totale fornita fino a
                        quel momento */
        v=0,            /* Somma della tensione nel tempo*/
        vminima=0,      /* Variabile di supporto */
        vi100=0,        /* Variabile di supporto */
        viSC=0,         /* Prodotto tensione corrente
                             scaricata */
        carica=0,       /* Energia fornita alla batteria*/
        scarica=0,      /* Energia fornita dalla batteria*/
        vsc=0;          /* Somma della tensione di scarica
                        nel tempo*/
int NUMBAT=0;               /* n° batteria - 1,2,4,5 litio
                                        - 3,6 cadmio */
#define RREF 0.496          /* Valore effettivo
                                    di Rsense */
#define IREF 800            /* Valore in mA della corrente
                             a cui effettuare la carica
                             a corrente costante per le
                            batterie agli ioni di litio */
#define VRIFERIMENTO 8.40   /* Valore teorico a cui
                            effettuare la carica
                             a tensione costante per le
                            batterie agli ioni di litio */
float64 VREF=VRIFERIMENTO;  /* Valore effettivo a cui
                            effettuare la carica
                             a tensione costante per le
                            batterie agli ioni di litio */
#define VMIS 8.20           /* tensione a cui esegue la
                             compensazione deve essere
                             < di VREF */
```

```
#define IMINREF 150          /* Valore in mA a cui dichiara
                             carica la batteria agli ioni di
                             litio*/
#define IREFCADMIO 550       /* Valore in mA della corrente
                              a cui effettuare la carica
                              a corrente costante per le
                             batterie al nichel cadmio */


#define NUM_MISURE 150       /*Indica il numero di campioni
                             utilizzati per la ricerca del
                             massimo della tensione durante
                             il ciclo di carica per le
                             batterie al nichel cadmio*/
float64 CARICATOTALE;        /* Vi è memorizzata l'energia
                             fornita alla batteria nel
                             punto di massimo*/
#define PASSO 0.00125        /*Passo di incremento-
                             decremento della tensione di
                             controllo*/
float64 VMINBATLITIO=6.00;   /*Tensione a cui la batteria
                             al litio è dichiarata carica*/
float64 VMINBATCADMIO=4.80; /*Tensione a cui la batteria
                             al NiCd è dichiarata carica*/
int flag11=0, flag12=0, flag13=0;    /* Flag necessari*/
int flag21=0, flag22=0, flag23=0;   /* alle verie */
int flag31=0, flag32=0, flag33=0,    /* procedure */
    flag34=0;
int flag41=0, flag42=0, flag43=0;
int flag51=0, flag52=0, flag53=0;
int flag61=0, flag62=0, flag63=0,
    flag64=0;
int flagxx=0, flagyy=0, flagtimer=0;
int32 tempo=0, tempo2=0, tempo3=0, tempo4=0;
float64 resto=0; resto2=0;          /* Variabili di */
float64 misura[NUM_MISURE];         /* supporto */
int i=0, j=0, l=0, blocco,
    blocco2, timerset, A=0, B=0;
float64 soglia;
```

```
void bloccatasti (int blocco)  /* A seconda del valore */
{                              /* ricevuto tramite     */
    int blocco2;               /* blocco, attiva o     */
    if(blocco==0) blocco2=1;   /* disattiva i pulsanti */
    if(blocco==1) blocco2=0;   /* sul pannello di cont.*/
    SetCtrlAttribute (panelHandle, MAIN_STOP,
                                   ATTR_DIMMED,blocco2);
    SetCtrlAttribute (panelHandle, MAIN_RESTART,
                                   ATTR_DIMMED, blocco);
    SetCtrlAttribute (panelHandle, MAIN_CICLOCOMPLETO,
                                   ATTR_DIMMED, blocco);
    SetCtrlAttribute (panelHandle, MAIN_CARICA1,
                                   ATTR_DIMMED, blocco);
    SetCtrlAttribute (panelHandle, MAIN_CARICA2,
                                   ATTR_DIMMED, blocco);
    SetCtrlAttribute (panelHandle, MAIN_CARICA3,
                                   ATTR_DIMMED, blocco);
    SetCtrlAttribute (panelHandle, MAIN_CARICA4,
                                   ATTR_DIMMED, blocco);
    SetCtrlAttribute (panelHandle, MAIN_CARICA5,
                                   ATTR_DIMMED, blocco);
    SetCtrlAttribute (panelHandle, MAIN_CARICA6,
                                   ATTR_DIMMED, blocco);
    SetCtrlAttribute (panelHandle, MAIN_SCARICA1,
                                   ATTR_DIMMED, blocco);
    SetCtrlAttribute (panelHandle, MAIN_SCARICA2,
                                   ATTR_DIMMED, blocco);
    SetCtrlAttribute (panelHandle, MAIN_SCARICA3,
                                   ATTR_DIMMED, blocco);
    SetCtrlAttribute (panelHandle, MAIN_SCARICA4,
                                   ATTR_DIMMED, blocco);
    SetCtrlAttribute (panelHandle, MAIN_SCARICA5,
                                   ATTR_DIMMED, blocco);
    SetCtrlAttribute (panelHandle, MAIN_SCARICA6,
                                   ATTR_DIMMED, blocco);
    return;
}
```

```
int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)
        return -1;  /* out of memory */
    if ((panelHandle = LoadPanel (0, "Caricabatterie.uir",
                                                  MAIN)) < 0)
                                                  return -1;
    DisplayPanel (panelHandle);
    bloccatasti (0);
    DAQmxCreateTask ("Tensione comando", &vout);
    DAQmxCreateAOVoltageChan (vout, "Dev1/ao0", "", 0, 5,
                                    DAQmx_Val_Volts, "");
    DAQmxWriteAnalogScalarF64 (vout, 1, 10.0, 0, 0);
    DAQmxCreateTask ("Tensione Batteria", &vbatt);
    DAQmxCreateAIVoltageChan (vbatt, "Dev1/ai0", "",
                                    DAQmx_Val_Diff,
                                    -10.0, 10.0,
                                    DAQmx_Val_Volts, "");
    DAQmxCreateTask ("Tensione corrente", &vcorr);
    DAQmxCreateAIVoltageChan (vcorr, "Dev1/ai1", "",
                                  DAQmx_Val_Diff, -1.0, 1.0,
                                  DAQmx_Val_Volts, "");
    DAQmxCreateTask ("P0OUT", &p0out);
    DAQmxCreateDOChan (p0out, "dev1/port0/line0,
                        dev1/port0/line1, dev1/port0/line2,
                        dev1/port0/line3,
                        dev1/port0/line4", "",
                        DAQmx_Val_ChanForAllLines);
    DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
    RunUserInterface ();
    DiscardPanel (panelHandle);
    return 0;
}
```

```
int CVICALLBACK stop (int panel, int control, int event,
                      void *callbackData,
                      int eventData1, int eventData2)
{                         /* Interrompe il processo */
    switch (event)        /* in corso bloccando le  */
    {                     /* funzioni timer*/
        case EVENT_COMMIT:
        DAQmxWriteAnalogScalarF64 (vout, 1, 10.0, 0, 0);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        bloccatasti (0);
        SetCtrlAttribute (panelHandle, MAIN_TIMER,
                          ATTR_ENABLED, 0);
        SetCtrlAttribute (panelHandle, MAIN_TIMER2,
                          ATTR_ENABLED, 0);
        SetCtrlAttribute (panelHandle, MAIN_STOP,
                          ATTR_DIMMED, 1);
        SetCtrlAttribute (panelHandle, MAIN_RESTART,
                          ATTR_DIMMED, 0);
        break;
    }
    return 0;
}

int CVICALLBACK ciclocompleto (int panel, int control,
                               int event,
                               void *callbackData,
                               int eventData1,
                               int eventData2)
{                         /* Imposta tutti i parametri*/
    switch (event)        /* necessari ad effettuare  */
    {                     /* il ciclo di carica delle */
        case EVENT_COMMIT:  /* sei batterie           */
        bloccatasti (1);
        NUMBAT=1;
        flagtimer=1;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        SetCtrlAttribute (panelHandle, MAIN_TIMER,
                          ATTR_ENABLED, 1);
        timerset=1;
        break;
    }
    return 0;
}
```

```
int CVICALLBACK esci (int panel, int control, int event,
                      void *callbackData, int eventData1,
                      int eventData2)
{                         /* Imposta a zero tutte le  */
    switch (event)        /* uscite, e chiude         */
    {                     /* l'applicazione           */
        case EVENT_COMMIT:
        DAQmxWriteAnalogScalarF64 (vout, 1, 10.0, 0, 0);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        QuitUserInterface (0);
        break;
    }
    return 0;
}
```

```
int CVICALLBACK carica1 (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{                   /* Esegue tutti i settagli necessari */
    switch (event)  /* ed avvia la procedura di carica   */
    {               /* della batteria 1 attivando        */
        case EVENT_COMMIT:      /* la funzione timer     */
        bloccatasti (1);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        A=13;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, A, 0);
        Sleep(10);
        B=9;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, B, 0);
        Sleep(2000);
        NUMBAT=1;
        DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
        vscheda=vbatteria/2.23; /* 2.23 = guadagno
                                   amplificatore */
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA1, 1);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA6, 0);
        SetCtrlAttribute (panelHandle, MAIN_TIMER,
                                  ATTR_ENABLED, 1);
        timerset=1;
        break;
    }
    return 0;
}
```

```
int CVICALLBACK carica2 (int panel, int control, int event,
                         void *callbackData, int eventData1,
                         int eventData2)
{                   /* Esegue tutti i settagli necessari */
    switch (event)  /* ed avvia la procedura di carica   */
    {               /* della batteria 2 attivando        */
        case EVENT_COMMIT:      /* la funzione timer     */
        bloccatasti (1);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        A=14;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, A, 0);
        Sleep(10);
        B=10;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, B, 0);
        Sleep(2000);
        NUMBAT=2;
        DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
        vscheda=vbatteria/2.23; /* 2.23 = guadagno
                                   amplificatore */
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA2, 1);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA6, 0);
        SetCtrlAttribute (panelHandle, MAIN_TIMER,
                                  ATTR_ENABLED, 1);
        timerset=1;
        break;
    }
    return 0;
}
```

```
int CVICALLBACK carica3 (int panel, int control, int event,
                         void *callbackData, int eventData1,
                         int eventData2)
{                   /* Esegue tutti i settagli necessari */
    switch (event)  /* ed avvia la procedura di carica   */
    {               /* della batteria 3 attivando        */
        case EVENT_COMMIT:      /* la funzione timer     */
        bloccatasti (1);
        flag31=0; flag32=0; flag33=0; flag34=0;
        vi=0;
        viSC=0;
        vi100=0;
        CARICATOTALE=0;
        flagxx=0;
        flagyy=1;
        x1=0;
        x2=0;
        NUMBAT=3;
        DAQmxWriteAnalogScalarF64 (vout, 1, 10.0, 0, 0);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        A=31;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, A, 0);
        Sleep(10);
        B=27;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, B, 0);
        Sleep(2000);
        vminima=VMINBATCADMIO;
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 1);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 0);
```

```
        SetCtrlAttribute (panelHandle, MAIN_TIMER,
                                       ATTR_ENABLED, 1);
        timerset=1;
        break;
    }
    return 0;
}
```

```
int CVICALLBACK carica4 (int panel, int control, int event,
                         void *callbackData, int eventData1,
                         int eventData2)
{                   /* Esegue tutti i settagli necessari */
    switch (event)  /* ed avvia la procedura di carica   */
    {               /* della batteria 4 attivando        */
        case EVENT_COMMIT:      /* la funzione timer     */
        bloccatasti (1);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        A=13;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, A, 0);
        Sleep(10);
        B=5;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, B, 0);
        Sleep(2000);
        NUMBAT=4;
        DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
        vscheda=vbatteria/2.23; /* 2.23 = guadagno
                                       amplificatore */
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA4, 1);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA6, 0);
        SetCtrlAttribute (panelHandle, MAIN_TIMER,
                                  ATTR_ENABLED, 1);
        timerset=1;
        break;
    }
    return 0;
}
```

```
int CVICALLBACK carica5 (int panel, int control, int event,
                         void *callbackData, int eventData1,
                         int eventData2)
{                    /* Esegue tutti i settagli necessari */
    switch (event)  /* ed avvia la procedura di carica   */
    {               /* della batteria 5 attivando        */
        case EVENT_COMMIT:      /* la funzione timer     */
        bloccatasti (1);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        A=14;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, A, 0);
        Sleep(10);
        B=6;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, B, 0);
        Sleep(2000);
        NUMBAT=5;
        DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
        vscheda=vbatteria/2.23; /* 2.23 = guadagno
                                  amplificatore */
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA5, 1);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA6, 0);
        SetCtrlAttribute (panelHandle, MAIN_TIMER,
                                  ATTR_ENABLED, 1);
        timerset=1;
        break;
    }
    return 0;
}
```

```
int CVICALLBACK carica6 (int panel, int control, int event,
                         void *callbackData, int eventData1,
                         int eventData2)
{                   /* Esegue tutti i settagli necessari */
    switch (event)  /* ed avvia la procedura di carica   */
    {               /* della batteria 6 attivando        */
        case EVENT_COMMIT:      /* la funzione timer     */
        bloccatasti (1);
        flag61=0; flag62=0; flag63=0; flag64=0;
        vi=0;
        viSC=0;
        vi100=0;
        CARICATOTALE=0;
        flagxx=0;
        flagyy==1;
        x1=0;
        x2=0;
        NUMBAT=6;
        DAQmxWriteAnalogScalarF64 (vout, 1, 10.0, 0, 0);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        A=31;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, A, 0);
        Sleep(10);
        B=23;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, B, 0);
        Sleep(2000);
        vminima=VMINBATCADMIO;
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 1);
```

```
        SetCtrlAttribute (panelHandle, MAIN_TIMER,
                                       ATTR_ENABLED, 1);
        timerset=1;
        break;
    }
    return 0;
}
```

```
int CVICALLBACK timer (int panel, int control, int event,
                       void *callbackData, int eventData1,
                       int eventData2)
{                   /* Viene eseguita su attivazione da  */
    switch (event)  /* parte di altre funzioni, è        */
    {               /* periodica con periodo 0.1 secondi */
        case EVENT_TIMER_TICK:
        tempo++;         /* queste due righe servono per */
        resto=tempo%50;  /* plottare un punto ogni
                            50 secondi*/
        DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
        DAQmxReadAnalogScalarF64 (vcorr, 10.0,
                                  &vcorrente, 0);
        ibatteria=(vcorrente*1000.0)/RREF;
        SetCtrlVal (panelHandle, MAIN_VALORECORRENTE,
                                  ibatteria);
        SetCtrlVal (panelHandle, MAIN_VALORETENSIONE,
                                  vbatteria);
        if (resto==0)
        {
            PlotStripChartPoint (panelHandle,
                                 MAIN_GRAFICOCORRENTE,
                                 ibatteria);
                                 /* questo if fa plottare un
                                 punto ogni 50 secondi */
            PlotStripChartPoint (panelHandle,
                                 MAIN_GRAFICOTENSIONE,
                                 vbatteria);
        }
```

```
if (NUMBAT==1)      /* Il contenuto di questo */
{                   /* if effettua la carica  */
    if (flag13==0)  /* della batteria 1       */
    {
        flag11=0; flag12=0; flag13=0;
        tempo3=0;
        v=0;
        vi100=0;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, 0, 0);
        A=13;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, A, 0);
        Sleep(10);
        B=9;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, B, 0);
        VREF=VRIFERIMENTO;
        DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
        vscheda=vbatteria/2.23; /*2.23=guad.ampl.*/
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA1, 1);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA6, 0);
        flag13=1;
        Sleep(2000);
    }
    if (vbatteria<VREF && flag11==0)
    {
        if (ibatteria<IREF)
        {
            vscheda=vscheda+PASSO;
            DAQmxWriteAnalogScalarF64 (vout, 1,
                                10.0, vscheda, 0);
        }else
        if (ibatteria>IREF)
        {
            vscheda=vscheda-PASSO;
            DAQmxWriteAnalogScalarF64 (vout, 1,
                                10.0, vscheda, 0);
        }
    }
```

```
tempo3++;
vi100=vi100+(vbatteria*ibatteria);
v=v+vbatteria;
SetCtrlVal (panelHandle, MAIN_VICARICA, vi100);
DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
DAQmxReadAnalogScalarF64 (vcorr, 10.0,
                                  &vcorrente, 0);
ibatteria=(vcorrente*1000.0)/RREF;
SetCtrlVal (panelHandle,
            MAIN_VALORECORRENTE, ibatteria);
SetCtrlVal (panelHandle,
            MAIN_VALORETENSIONE, vbatteria);
if (vbatteria>=VMIS && flag12==0)
{
    flag12=1;
    DAQmxWriteAnalogScalarF64 (vout, 1,
                                  10.0, 0, 0);
    delta=vbatteria;
    DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
    delta=delta-vbatteria;
    VREF=VREF+(delta);  /* espressione corretta
                        VREF=VREF+(delta/coeffi
                        ciente di taratura)
                        in fase di test del
                        satellite si è
                        verificato che il
                        coefficiente di
                        taratura è prossimo a
                        1, se necessario è
                        possibile modificarlo;
                        */
}
if (vbatteria>=VREF) flag11=1;
if (flag11==1)
{
    if (vbatteria<VREF)
    {
        vscheda=vscheda+PASSO;
        DAQmxWriteAnalogScalarF64 (vout, 1,
                              10.0, vscheda, 0);
    }else

    if (vbatteria>VREF)
    {
        vscheda=vscheda-PASSO;
        DAQmxWriteAnalogScalarF64 (vout, 1,
                              10.0, vscheda, 0);
    }
```

```
            if (ibatteria<IMINREF)
            {
                if (flagtimer==0)
                {
                    SetCtrlAttribute (panelHandle,
                                              MAIN_TIMER,
                                              ATTR_ENABLE
                                              D, 0);
                    bloccatasti (0);
                }
                if (flagtimer==1) NUMBAT=2;
                flag13=0;
                carica=(vi100/v)*
                        (((float64)tempo3)/36000.0);
                SetCtrlVal (panelHandle,
                                MAIN_QUANTITA1,
                                carica);
                DAQmxWriteAnalogScalarF64 (vout, 1,
                                          10.0, 0, 0);
                SetCtrlVal (panelHandle,
                                MAIN_LEDINCARICA1, 0);
                SetCtrlVal (panelHandle,
                                MAIN_LEDCARICA1, 1);
            }
        }
    }
```

```
if (NUMBAT==2)      /* Il contenuto di questo */
{                   /* if effettua la carica  */
    if (flag23==0)  /* della batteria 2       */
    {
        flag21=0; flag22=0; flag23=0;
        tempo3=0;
        v=0;
        vi100=0;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, 0, 0);
        A=14;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, A, 0);
        Sleep(10);
        B=10;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, B, 0);
        VREF=VRIFERIMENTO;
        DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
        vscheda=vbatteria/2.23; /*2.23=guad.ampl.*/
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA2, 1);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA6, 0);
        flag23=1;
        Sleep(2000);
    }
    if (vbatteria<VREF && flag21==0)
    {
        if (ibatteria<IREF)
        {
            vscheda=vscheda+PASSO;
            DAQmxWriteAnalogScalarF64 (vout, 1,
                                10.0, vscheda, 0);
        }else
        if (ibatteria>IREF)
        {
            vscheda=vscheda-PASSO;
            DAQmxWriteAnalogScalarF64 (vout, 1,
                                10.0, vscheda, 0);
        }
    }
```

```
tempo3++;
vi100=vi100+(vbatteria*ibatteria);
v=v+vbatteria;
SetCtrlVal (panelHandle, MAIN_VICARICA, vi100);
DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
DAQmxReadAnalogScalarF64 (vcorr, 10.0,
                                  &vcorrente, 0);
ibatteria=(vcorrente*1000.0)/RREF;
SetCtrlVal (panelHandle,
            MAIN_VALORECORRENTE, ibatteria);
SetCtrlVal (panelHandle,
            MAIN_VALORETENSIONE, vbatteria);

if (vbatteria>=VMIS && flag22==0)
{
    flag22=1;
    DAQmxWriteAnalogScalarF64 (vout, 1,
                                  10.0, 0, 0);
    delta=vbatteria;
    DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
    delta=delta-vbatteria;
    VREF=VREF+(delta);
}
if (vbatteria>=VREF) flag21=1;
if (flag21==1)
{
    if (vbatteria<VREF)
    {
        vscheda=vscheda+PASSO;
        DAQmxWriteAnalogScalarF64 (vout, 1,
                            10.0, vscheda, 0);
    }else
    if (vbatteria>VREF)
    {
        vscheda=vscheda-PASSO;
        DAQmxWriteAnalogScalarF64 (vout, 1,
                            10.0, vscheda, 0);
    }
```

```
            if (ibatteria<IMINREF)
            {
                if (flagtimer==0)
                {
                    SetCtrlAttribute (panelHandle,
                                      MAIN_TIMER,
                                      ATTR_ENABLED, 0);
                    bloccatasti (0);
                }
                if (flagtimer==1) NUMBAT=3;
                flag23=0;
                carica=(vi100/v)
                       *(((float64)tempo3)/36000.0);
                SetCtrlVal (panelHandle,
                            MAIN_QUANTITA2, carica);
                DAQmxWriteAnalogScalarF64 (vout, 1,
                                           10.0, 0, 0);
                SetCtrlVal (panelHandle,
                                MAIN_LEDINCARICA2, 0);
                SetCtrlVal (panelHandle,
                                MAIN_LEDCARICA2, 1);
            }
        }
    }
```

```
if (NUMBAT==3)      /* Il contenuto di questo if */
{                   /* effettua la carica della  */
    if (flagxx==0)  /* batteria 3                */
    {
        for (l=0;l<=120;l++) misura[l]=0;
        flag31=0; flag32=0; flag33=0; flag34=0;
        v=0;
        vsc=0;
        tempo3=0;
        tempo4=0;
        vi=0;
        viSC=0;
        vi100=0;
        CARICATOTALE=0;
        flagxx=0;
        flagyy=0;
        x1=0;
        x2=0;
        i=0;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, 0, 0);
        A=31;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, A, 0);
        Sleep(10);
        B=27;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, B, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINSCARICA3, 1);
        flagxx=1;
        Sleep(2000);
    }
    DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                   &vbatteria, 0);
    DAQmxReadAnalogScalarF64 (vcorr, 10.0,
                                   &vcorrente, 0);
    ibatteria=(vcorrente*1000.0)/RREF;
    if (flag34==0)
    {
        vsc=vsc+vbatteria;
        tempo4++;
        viSC=(viSC+(vbatteria*ibatteria));
        SetCtrlVal (panelHandle,
                            MAIN_VISCARICA, viSC);
    }
```

```
if (vbatteria<=VMINBATCADMIO)
{
    scarica=(viSC/vsc)
                    *(((float64)tempo4)/36000);
    SetCtrlVal (panelHandle,
                   MAIN_QUANTITASC3, scarica);
    SetCtrlVal (panelHandle,
                        MAIN_LEDINSCARICA1, 0);
    SetCtrlVal (panelHandle,
                        MAIN_LEDINSCARICA2, 0);
    SetCtrlVal (panelHandle,
                        MAIN_LEDINSCARICA3, 0);
    SetCtrlVal (panelHandle,
                        MAIN_LEDINSCARICA4, 0);
    SetCtrlVal (panelHandle,
                        MAIN_LEDINSCARICA5, 0);
    SetCtrlVal (panelHandle,
                        MAIN_LEDINSCARICA6, 0);
    SetCtrlVal (panelHandle,
                        MAIN_LEDSCARICA3, 1);
    flag34=1;
    flagyy=0;
}
if (flag34==1)
{
    if (flagyy==0)
    {
        DAQmxWriteDigitalScalarU32
                        (p0out, 1, 0.1, 0, 0);
        A=15;
        DAQmxWriteDigitalScalarU32
                        (p0out, 1, 0.1, A, 0);
        Sleep(10);
        B=11;
        DAQmxWriteDigitalScalarU32
                        (p0out, 1, 0.1, B, 0);
        Sleep(2000);
        if (flag33==0)
        {
            DAQmxReadAnalogScalarF64
                                (vbatt, 10.0,
                                &vbatteria, 0);
            vscheda=vbatteria/2.23
            flag33=1;
        }
        SetCtrlVal (panelHandle,
                        MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle,
                        MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle,
                        MAIN_LEDINCARICA3, 1);
```

```
    SetCtrlVal (panelHandle,
                    MAIN_LEDINCARICA4, 0);
    SetCtrlVal (panelHandle,
                    MAIN_LEDINCARICA5, 0);
    SetCtrlVal (panelHandle,
                    MAIN_LEDINCARICA6, 0);
    SetCtrlAttribute (panelHandle,
                        MAIN_TIMER,
                        ATTR_ENABLED, 1);
    timerset=1;
    flagyy=1;
}
misura[i]=vbatteria;
i++;
if (i==100)
{
    if (flagyy==1)
    {
        x1=0;
        for j=0;j<=100;j++)
                        x1=x1+misura[j];
        SetCtrlVal (panelHandle,
                            MAIN_X1, x1);
        if (x1>x2) x2=x1;
        SetCtrlVal (panelHandle,
                            MAIN_X2, x2);
        if (x1+2.5<x2)
        {
            carica=(vi100/v)
            *(((float64)tempo3)/36000.0);
            SetCtrlVal (panelHandle,
                            MAIN_QUANTITA3,
                            carica);
            flag31=1;
        }
    }
    i=0;
}
if (ibatteria<IREFCADMIO)
{
    vscheda=vscheda+PASSO;
    DAQmxWriteAnalogScalarF64
                        (vout, 1, 10.0,
                        vscheda, 0);
}else
if (ibatteria>IREFCADMIO)
{
    vscheda=vscheda-PASSO;
    DAQmxWriteAnalogScalarF64
            (vout, 1, 10.0, vscheda, 0);
}
```

```
            if (flag31==0)
            {
                tempo3++;
                v=v+vbatteria;
                vi100=vi100+(vbatteria*ibatteria);
                SetCtrlVal (panelHandle,
                                MAIN_VICARICA, vi100);
                CARICATOTALE=0.4*vi100;
            }
            DAQmxReadAnalogScalarF64 (vbatt,
                                10.0, &vbatteria, 0);
            DAQmxReadAnalogScalarF64 (vcorr,
                                10.0, &vcorrente, 0);
            ibatteria=(vcorrente*1000.0)/RREF;
            if (flag31==1)
            {
                vi=vi+(vbatteria*ibatteria);
                SetCtrlVal (panelHandle,
                                MAIN_VISOVRACCARICA,
                                vi);
                if (vi>CARICATOTALE)
                {
                    if (flagtimer==0)
                    {
                        SetCtrlAttribute
                                        (panelHandle,
                                        MAIN_TIMER,
                                        ATTR_ENABLED,
                                        0);
                        bloccatasti (0);
                    }
                    if (flagtimer==1) NUMBAT=4;
                    DAQmxWriteAnalogScalarF64
                                (vout, 1, 10.0, 0, 0);
                    SetCtrlVal (panelHandle,
                                MAIN_LEDINCARICA3, 0);
                    SetCtrlVal (panelHandle,
                                MAIN_LEDCARICA3, 1);
                    SetCtrlVal (panelHandle,
                                MAIN_LEDSCARICA3, 0);
                    flagxx=0;
                }
            }

            DAQmxReadAnalogScalarF64 (vbatt,
                                10.0, &vbatteria, 0);
            SetCtrlVal (panelHandle,
                                MAIN_VALORETENSIONE,
                                vbatteria);
        }
    }
```

```
if (NUMBAT==4)      /* Il contenuto di questo */
{                   /* if effettua la carica  */
    if (flag43==0)  /* della batteria 4       */
    {
        flag41=0; flag42=0; flag43=0;
        tempo3=0;
        v=0;
        vi100=0;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, 0, 0);
        A=13;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, A, 0);
        Sleep(10);
        B=5;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, B, 0);
        VREF=VRIFERIMENTO;
        DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
        vscheda=vbatteria/2.23;
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA4, 1);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA6, 0);
        flag43=1;
        Sleep(2000);
    }
    if (vbatteria<VREF && flag41==0)
    {
        if (ibatteria<IREF)
        {
            vscheda=vscheda+PASSO;
            DAQmxWriteAnalogScalarF64 (vout, 1,
                                10.0, vscheda, 0);
        }else
        if (ibatteria>IREF)
        {
            vscheda=vscheda-PASSO;
            DAQmxWriteAnalogScalarF64 (vout, 1,
                                10.0, vscheda, 0);
        }
    }
```

```
tempo3++;
vi100=vi100+(vbatteria*ibatteria);
v=v+vbatteria;
SetCtrlVal (panelHandle, MAIN_VICARICA, vi100);
DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
DAQmxReadAnalogScalarF64 (vcorr, 10.0,
                                  &vcorrente, 0);
ibatteria=(vcorrente*1000.0)/RREF;
SetCtrlVal (panelHandle,
               MAIN_VALORECORRENTE,
               ibatteria);
SetCtrlVal (panelHandle,
               MAIN_VALORETENSIONE,
               vbatteria);
if (vbatteria>=VMIS && flag42==0)
{
    flag42=1;
    DAQmxWriteAnalogScalarF64 (vout, 1,
                                  10.0, 0, 0);
    delta=vbatteria;
    DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
    delta=delta-vbatteria;
    VREF=VREF+(delta);
}
if (vbatteria>=VREF) flag41=1;
if (flag41==1)
{
    if (vbatteria<VREF)
    {
        vscheda=vscheda+PASSO;
        DAQmxWriteAnalogScalarF64 (vout, 1,
                           10.0, vscheda, 0);
    }else
    if (vbatteria>VREF)
    {
        vscheda=vscheda-PASSO;
        DAQmxWriteAnalogScalarF64 (vout, 1,
                           10.0, vscheda, 0);
    }
    if (ibatteria<IMINREF)
    {
        if (flagtimer==0)
        {
            SetCtrlAttribute (panelHandle,
                           MAIN_TIMER,
                           ATTR_ENABLED, 0);
            bloccatasti (0);
        }
```

```
                if (flagtimer==1) NUMBAT=5;
                carica=(vi100/v)*
                        (((float64)tempo3)/36000.0);
                SetCtrlVal (panelHandle,
                            MAIN_QUANTITA4, carica);
                DAQmxWriteAnalogScalarF64 (vout, 1,
                                           10.0, 0, 0);
                flag43=0;
                SetCtrlVal (panelHandle,
                                MAIN_LEDINCARICA4, 0);
                SetCtrlVal (panelHandle,
                                MAIN_LEDCARICA4, 1);
            }
        }
    }
```

```
if (NUMBAT==5)      /* Il contenuto di questo */
{                   /* if effettua la carica  */
    if (flag53==0)  /* della batteria 5       */
    {
        flag51=0; flag52=0; flag53=0;
        tempo3=0;
        v=0;
        vi100=0;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, 0, 0);
        A=14;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, A, 0);
        Sleep(10);
        B=6;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, B, 0);
        VREF=VRIFERIMENTO;
        DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
        vscheda=vbatteria/2.23
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA5, 1);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA6, 0);
        flag53=1;
        Sleep(2000);
    }
    if (vbatteria<VREF && flag51==0)
    {
        if (ibatteria<IREF)
        {
            vscheda=vscheda+PASSO;
            DAQmxWriteAnalogScalarF64 (vout, 1,
                                10.0, vscheda, 0);
        }else
        if (ibatteria>IREF)
        {
            vscheda=vscheda-PASSO;
            DAQmxWriteAnalogScalarF64 (vout, 1,
                                10.0, vscheda, 0);
        }
    }
```

```
tempo3++;
vi100=vi100+(vbatteria*ibatteria);
v=v+vbatteria;
SetCtrlVal (panelHandle, MAIN_VICARICA, vi100);
DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
DAQmxReadAnalogScalarF64 (vcorr, 10.0,
                                  &vcorrente, 0);
ibatteria=(vcorrente*1000.0)/RREF;
SetCtrlVal (panelHandle,
                MAIN_VALORECORRENTE,
                ibatteria);
SetCtrlVal (panelHandle,
                MAIN_VALORETENSIONE,
                vbatteria);
if (vbatteria>=VMIS && flag52==0)
{
    flag52=1;
    DAQmxWriteAnalogScalarF64 (vout, 1,
                                  10.0, 0, 0);
    delta=vbatteria;
    DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
    delta=delta-vbatteria;
    VREF=VREF+(delta);
}
if (vbatteria>=VREF) flag51=1;
if (flag51==1)
{
    if (vbatteria<VREF)
    {
        vscheda=vscheda+PASSO;
        DAQmxWriteAnalogScalarF64 (vout, 1,
                            10.0, vscheda, 0);
    }else
    if (vbatteria>VREF)
    {
        vscheda=vscheda-PASSO;
        DAQmxWriteAnalogScalarF64 (vout, 1,
                            10.0, vscheda, 0);
    }
    if (ibatteria<IMINREF)
    {
        if (flagtimer==0)
        {
            SetCtrlAttribute (panelHandle,
                            MAIN_TIMER,
                            ATTR_ENABLED, 0);
            bloccatasti (0);
        }
```

```
                if (flagtimer==1) NUMBAT=6;
                carica=(vi100/v)
                        *(((float64)tempo3)/36000.0);
                SetCtrlVal (panelHandle,
                            MAIN_QUANTITA5, carica);
                DAQmxWriteAnalogScalarF64 (vout, 1,
                                           10.0, 0, 0);
                flag53=0;
                SetCtrlVal (panelHandle,
                                MAIN_LEDINCARICA5, 0);
                SetCtrlVal (panelHandle,
                                MAIN_LEDCARICA5, 1);
            }
        }
    }
```

```
if (NUMBAT==6)      /* Il contenuto di questo if */
{                   /* effettua la carica della  */
    if (flagxx==0)  /* batteria 6                */
    {
        for (l=0;l<=120;l++) misura[l]=0;
        flag61=0; flag62=0; flag63=0; flag64=0;
        v=0;
        vsc=0;
        tempo3=0;
        tempo4=0;
        vi=0;
        viSC=0;
        vi100=0;
        CARICATOTALE=0;
        flagxx=0;
        flagyy=0;
        x1=0;
        x2=0;
        i=0;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, 0, 0);
        A=31;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, A, 0);
        Sleep(10);
        B=23;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, B, 0);
        SetCtrlVal (panelHandle,
                            MAIN_LEDINSCARICA6, 1);
        flagxx=1;
        Sleep(2000);
    }
    DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                  &vbatteria, 0);
    DAQmxReadAnalogScalarF64 (vcorr, 10.0,
                                  &vcorrente, 0);
    ibatteria=(vcorrente*1000.0)/RREF;
    if (flag64==0)
    {
        vsc=vsc+vbatteria;
        tempo4++;
        viSC=(viSC+(vbatteria*ibatteria));
        SetCtrlVal (panelHandle, MAIN_VISCARICA,
                                          viSC);
    }
```

```
if (vbatteria<=VMINBATCADMIO)
{
    scarica=(viSC/vsc)
            *(((float64)tempo4)/36000);
    SetCtrlVal (panelHandle, MAIN_QUANTITASC6,
                                    scarica);
    SetCtrlVal (panelHandle,
                        MAIN_LEDINSCARICA1, 0);
    SetCtrlVal (panelHandle,
                        MAIN_LEDINSCARICA2, 0);
    SetCtrlVal (panelHandle,
                        MAIN_LEDINSCARICA3, 0);
    SetCtrlVal (panelHandle,
                        MAIN_LEDINSCARICA4, 0);
    SetCtrlVal (panelHandle,
                        MAIN_LEDINSCARICA5, 0);
    SetCtrlVal (panelHandle,
                        MAIN_LEDINSCARICA6, 0);
    SetCtrlVal (panelHandle,
                        MAIN_LEDSCARICA6,1);
    flag64=1;
    flagyy=0;
}
if (flag64==1)
{
    if (flagyy==0)
    {
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, 0, 0);
        A=15;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, A, 0);
        Sleep(10);
        B=7;
        DAQmxWriteDigitalScalarU32 (p0out, 1,
                                    0.1, B, 0);
        Sleep(2000);
        if (flag63==0)
        {
            DAQmxReadAnalogScalarF64 (vbatt,
                        10.0, &vbatteria, 0);
            vscheda=vbatteria/2.23;
            flag63=1;
        }
        SetCtrlVal (panelHandle,
                        MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle,
                        MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle,
                        MAIN_LEDINCARICA3, 0);
```

```
            SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA4, 0);
            SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA5, 0);
            SetCtrlVal (panelHandle,
                            MAIN_LEDINCARICA6, 1);
            SetCtrlAttribute (panelHandle,
                    MAIN_TIMER, ATTR_ENABLED, 1);
            timerset=1;
            flagyy=1;
        }
        misura[i]=vbatteria;
        i++;
        if (i==100)
        {
        if (flagyy==1)
        {
            x1=0;
            for (j=0;j<=100;j++)
                                x1=x1+misura[j];
            SetCtrlVal (panelHandle, MAIN_X1, x1);
            if (x1>x2) x2=x1;
            SetCtrlVal (panelHandle, MAIN_X2, x2);
            if (x1+2.5<x2)
            {
                carica=(vi100/v)
                    *(((float64)tempo3)/36000.0);
                SetCtrlVal (panelHandle,
                        MAIN_QUANTITA6, carica);
                flag61=1;
            }
        }
        i=0;
    }
    if (ibatteria<IREFCADMIO)
    {
        vscheda=vscheda+PASSO;
        DAQmxWriteAnalogScalarF64 (vout, 1, 10.0,
                                   vscheda, 0);
    }else
    if (ibatteria>IREFCADMIO)
    {
        vscheda=vscheda-PASSO;
        DAQmxWriteAnalogScalarF64 (vout, 1, 10.0,
                                   vscheda, 0);
    }
```

```
            if (flag61==0)
            {
                tempo3++;
                v=v+vbatteria;
                vi100=vi100+(vbatteria*ibatteria);
                SetCtrlVal (panelHandle, MAIN_VICARICA,
                                                    vi100);
                CARICATOTALE=0.4*vi100;
            }
                DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                        &vbatteria, 0);
                DAQmxReadAnalogScalarF64 (vcorr, 10.0,
                                        &vcorrente, 0);
                ibatteria=(vcorrente*1000.0)/RREF;
                if (flag61==1)
                {
                    vi=vi+(vbatteria*ibatteria);
                    SetCtrlVal (panelHandle,
                                MAIN_VISOVRACCARICA, vi);
                    if (vi>CARICATOTALE)
                    {
                        SetCtrlAttribute (panelHandle,
                            MAIN_TIMER, ATTR_ENABLED, 0);
                        bloccatasti (0);
                        DAQmxWriteAnalogScalarF64 (vout, 1,
                                        10.0, 0, 0);
                        SetCtrlVal (panelHandle,
                                    MAIN_LEDINCARICA6, 0);
                        SetCtrlVal (panelHandle,
                                    MAIN_LEDCARICA6, 1);
                        SetCtrlVal (panelHandle,
                                    MAIN_LEDSCARICA6, 0);
                        flagxx=0;
                    }
                }
                DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                        &vbatteria, 0);
                SetCtrlVal (panelHandle,
                        MAIN_VALORETENSIONE, vbatteria);
            }
        }
        break;
    }
    return 0;
}
```

```
int CVICALLBACK scarica1 (int panel, int control,
                          int event, void *callbackData,
                          int eventData1, int eventData2)
{
    switch (event)         /* Effettua tutti I settaggi */
    {                      /* necessary e procede alla  */
        case EVENT_COMMIT: /* scarica della batteria 1  */
        tempo2=0;          /* attivando la funzione     */
        vsc=0; viSC=0;     /* timer2 */
        bloccatasti (1);
        NUMBAT=1;
        DAQmxWriteAnalogScalarF64 (vout, 1, 10.0, 0, 0);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        A=29;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, A, 0);
        Sleep(10);
        B=25;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, B, 0);
        Sleep(2000);
        vminima=VMINBATLITIO;
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 1);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA6, 0);
        SetCtrlAttribute (panelHandle, MAIN_TIMER2,
                                       ATTR_ENABLED, 1);
        timerset=2;
        break;
    }
    return 0;
}
```

```
int CVICALLBACK scarica2 (int panel, int control,
                          int event, void *callbackData,
                          int eventData1, int eventData2)
{
    switch (event)          /* Effettua tutti I settaggi */
    {                       /* necessary e procede alla  */
        case EVENT_COMMIT:  /* scarica della batteria 2  */
        tempo2=0;           /* attivando la funzione     */
        vsc=0; viSC=0;      /* timer2 */
        bloccatasti (1);
        NUMBAT=2;
        DAQmxWriteAnalogScalarF64 (vout, 1, 10.0, 0, 0);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        A=30;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, A, 0);
        Sleep(10);
        B=26;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, B, 0);
        Sleep(2000);
        vminima=VMINBATLITIO;
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 1);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA6, 0);
        SetCtrlAttribute (panelHandle, MAIN_TIMER2,
                                        ATTR_ENABLED, 1);
        timerset=2;
        break;
    }
    return 0;
}
```

```
int CVICALLBACK scarica3 (int panel, int control,
                          int event, void *callbackData,
                          int eventData1, int eventData2)
{
    switch (event)          /* Effettua tutti I settaggi */
    {                       /* necessary e procede alla  */
        case EVENT_COMMIT:  /* scarica della batteria 3  */
        tempo2=0;           /* attivando la funzione     */
        vsc=0; viSC=0;      /* timer2 */
        bloccatasti (1);
        NUMBAT=3;
        DAQmxWriteAnalogScalarF64 (vout, 1, 10.0, 0, 0);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        A=31;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, A, 0);
        Sleep(10);
        B=27;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, B, 0);
        Sleep(2000);
        vminima=VMINBATCADMIO;
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 1);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA6, 0);
        SetCtrlAttribute (panelHandle, MAIN_TIMER2,
                                       ATTR_ENABLED, 1);
        timerset=2;
        break;
    }
    return 0;
}
```

```
int CVICALLBACK scarica4 (int panel, int control,
                          int event, void *callbackData,
                          int eventData1, int eventData2)
{
    switch (event)          /* Effettua tutti I settaggi */
    {                       /* necessary e procede alla  */
        case EVENT_COMMIT:  /* scarica della batteria 4  */
        tempo2=0;           /* attivando la funzione     */
        vsc=0; viSC=0;      /* timer2 */
        bloccatasti (1);
        NUMBAT=4;
        DAQmxWriteAnalogScalarF64 (vout, 1, 10.0, 0, 0);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        A=29;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, A, 0);
        Sleep(10);
        B=21;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, B, 0);
        Sleep(2000);
        vminima=VMINBATLITIO;
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 1);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA6, 0);
        SetCtrlAttribute (panelHandle, MAIN_TIMER2,
                                        ATTR_ENABLED, 1);
        timerset=2;
        break;
    }
    return 0;
}
```

```
int CVICALLBACK scarica5 (int panel, int control,
                          int event, void *callbackData,
                          int eventData1, int eventData2)
{
    switch (event)          /* Effettua tutti I settaggi */
    {                       /* necessary e procede alla  */
        case EVENT_COMMIT:  /* scarica della batteria 5  */
        tempo2=0;           /* attivando la funzione     */
        vsc=0; viSC=0;      /* timer2 */
        bloccatasti (1);
        NUMBAT=5;
        DAQmxWriteAnalogScalarF64 (vout, 1, 10.0, 0, 0);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        A=30;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, A, 0);
        Sleep(10);
        B=22;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, B, 0);
        Sleep(2000);
        vminima=VMINBATLITIO;
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 1);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA6, 0);
        SetCtrlAttribute (panelHandle, MAIN_TIMER2,
                                       ATTR_ENABLED, 1);
        timerset=2;
        break;
    }
    return 0;
}
```

```
int CVICALLBACK scarica6 (int panel, int control,
                          int event, void *callbackData,
                          int eventData1, int eventData2)
{
    switch (event)          /* Effettua tutti I settaggi */
    {                       /* necessary e procede alla  */
        case EVENT_COMMIT:  /* scarica della batteria 6  */
        tempo2=0;           /* attivando la funzione     */
        vsc=0; viSC=0;      /* timer2 */
        bloccatasti (1);
        NUMBAT=6;
        DAQmxWriteAnalogScalarF64 (vout, 1, 10.0, 0, 0);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        A=31;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, A, 0);
        Sleep(10);
        B=23;
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, B, 0);
        Sleep(2000);
        vminima=VMINBATLITIO;
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDCARICA6, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDINSCARICA6, 1);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA1, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA2, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA3, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA4, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA5, 0);
        SetCtrlVal (panelHandle, MAIN_LEDSCARICA6, 0);
        SetCtrlAttribute (panelHandle, MAIN_TIMER2,
                                       ATTR_ENABLED, 1);
        timerset=2;
        break;
    }
    return 0;
}
```

```
int CVICALLBACK timer2 (int panel, int control, int event,
                        void *callbackData, int eventData1,
                        int eventData2)
{                   /* Viene eseguita su attivazione da  */
    switch (event)  /* parte di altre funzioni, è        */
    {               /* periodica con periodo 0.1 secondi */
        case EVENT_TIMER_TICK:
        tempo2++;        /* queste due righe servono per */
        resto2=tempo2%10;  /* plottare un punto al secondo
                              invece di 10 */
        DAQmxReadAnalogScalarF64 (vbatt, 10.0,
                                            &vbatteria, 0);
        DAQmxReadAnalogScalarF64 (vcorr, 10.0,
                                            &vcorrente, 0);
        ibatteria=(vcorrente*1000.0)/RREF;
        SetCtrlVal (panelHandle, MAIN_VALORECORRENTE,
                                                ibatteria);
        SetCtrlVal (panelHandle, MAIN_VALORETENSIONE,
                                                vbatteria);
        viSC=(viSC+(vbatteria*ibatteria));
        vsc=vsc+vbatteria;
        SetCtrlVal (panelHandle, MAIN_VISCARICA, viSC);
        if (resto2==0)
        {
            PlotStripChartPoint (panelHandle,
                        MAIN_GRAFICOCORRENTE, ibatteria);
                        /* questo if fa plottare un punto
                            al secondo invece di 10 */
            PlotStripChartPoint (panelHandle,
                        MAIN_GRAFICOTENSIONE, vbatteria);
        }
        if (vbatteria<=vminima)
        {
            scarica=(viSC/vsc)*(((float64)tempo2)/36000);
            SetCtrlVal (panelHandle,
                                    MAIN_LEDINSCARICA1, 0);
            SetCtrlVal (panelHandle,
                                    MAIN_LEDINSCARICA2, 0);
            SetCtrlVal (panelHandle,
                                    MAIN_LEDINSCARICA3, 0);
            SetCtrlVal (panelHandle,
                                    MAIN_LEDINSCARICA4, 0);
            SetCtrlVal (panelHandle,
                                    MAIN_LEDINSCARICA5, 0);
            SetCtrlVal (panelHandle,
                                    MAIN_LEDINSCARICA6, 0);
```

```
if (NUMBAT==1)
{
    SetCtrlVal (panelHandle,
                        MAIN_LEDSCARICA1, 1);
    SetCtrlVal (panelHandle, MAIN_QUANTITASC1,
                                    scarica);
}

if (NUMBAT==2)
{
    SetCtrlVal (panelHandle,
                        MAIN_LEDSCARICA2, 1);
    SetCtrlVal (panelHandle, MAIN_QUANTITASC2,
                                    scarica);
};

if (NUMBAT==3)
{
    SetCtrlVal (panelHandle,
                        MAIN_LEDSCARICA3, 1);
    SetCtrlVal (panelHandle, MAIN_QUANTITASC3,
                                    scarica);
}

if (NUMBAT==4)
{
    SetCtrlVal (panelHandle,
                        MAIN_LEDSCARICA4, 1);
    SetCtrlVal (panelHandle, MAIN_QUANTITASC4,
                                    scarica);
}

if (NUMBAT==5)
{
    SetCtrlVal (panelHandle,
                        MAIN_LEDSCARICA5, 1);
    SetCtrlVal (panelHandle, MAIN_QUANTITASC5,
                                    scarica);
}

if (NUMBAT==6)
{
    SetCtrlVal (panelHandle,
                        MAIN_LEDSCARICA6, 1);
    SetCtrlVal (panelHandle, MAIN_QUANTITASC6,
                                    scarica);
}
```

```
            DAQmxWriteDigitalScalarU32 (p0out, 1,
                                                0.1, 0, 0);
            SetCtrlAttribute (panelHandle, MAIN_TIMER2,
                                           ATTR_ENABLED, 0);
            bloccatasti (0);
        }
        break;
    }
    return 0;
}
```

```
int CVICALLBACK restart (int panel, int control, int event,
                         void *callbackData, int eventData1,
                         int eventData2)
{                   /* Può essere eseguita dopo        */
    switch (event)  /* l'esecuzione di stop e permette */
    {               /* di riprendere il processo dal   */
            case EVENT_COMMIT:  /* punto di interruzione*/
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, 0, 0);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, A, 0);
        Sleep(10);
        DAQmxWriteDigitalScalarU32 (p0out, 1, 0.1, B, 0);
        if (timerset==1) SetCtrlAttribute (panelHandle,
                                           MAIN_TIMER,
                                           ATTR_ENABLED,
                                           1);
        if (timerset==2) SetCtrlAttribute (panelHandle,
                                           MAIN_TIMER2,
                                           ATTR_ENABLED,
                                           1);
        bloccatasti (1);
        break;
    }
    return 0;
}
```

Bibliografia

[1] Reyneri L., *PiCPoT Satellite Universitario del Politecnico di Torino, Documento di specifiche dei sottosistemi elettronici, Versione 6.*
[2] polimage.polito.it/picpot
[3] Benenati G., *Progettazione del sottosistema di comunicazione a 2,4 GHz per PiCPoT 2,* Tesi di Laurea
[4] Speretta S., *Collaudo ed Integrazione del satellite universitario PiCPoT, Tesi di Laurea*.
[5] Tranchero M. *Progetto e Realizzazione del Satellite PiCPoT: Processore di Bordo,* Tesi di Laurea.
[6] www.dobson-space-telescope.com
[7] www.cubesat.auc.dk
[8] ams.astro.univie.ac.at/brite/brite-intro.html
[9] swisscube.epfl.ch
[10] microsat.sm.bmstu.ru/eng/epstruct.htm
[11] metis.umh.es/nowsat
[12] www.sstl.co.uk
[13] sourceforge.net/projects/mgsn