ISSN 2039-7941
Anno 2017_Numero 388

rapporti tecnici INGV

Gas Gauge System – Sistema di controllo delle batterie al piombo per l'Osservatorio geomagnetico di Lampedusa

388

Istituto Nazionale di Geofisica e Vulcanologia

Segreteria di Redazione
Francesca Di Stefano - Referente
Rossella Celi
Tel. +39 06 51860068
redazionecen@ingv.it

in collaborazione con:
Barbara Angioni (RM1)

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

Rappresentante legale: Carlo DOGLIONI
Sede: Via di Vigna Murata, 605 | Roma

ISSN 2039-7941
Anno 2017_Numero 388

rapporti tecnici

INGV

GAS GAUGE SYSTEM – SISTEMA DI CONTROLLO DELLE BATTERIE AL PIOMBO PER L'OSSERVATORIO GEOMAGNETICO DI LAMPEDUSA

Giovanni Benedetti, Achille Emanuele Zirizzotti

INGV (Istituto Nazionale di Geofisica e Vulcanologia, Sezione Geomagnetismo, Aeronomia e Geofisica Ambientale)

388

Come citare: Benedetti G., Zirizzotti A.E., (2017). Gas Gauge System - Sistema di controllo delle batterie al piombo per l'Osservatorio geomagnetico di Lampedusa. Rapp. Tec. INGV, 388: 1-34.

Indice

Introduzione

In molti sistemi di monitoraggio, come gli osservatori remoti e le stazioni di misura di parametri geofisici, geochimici o altro, si utilizzano batterie, gruppi di continuità (UPS) e pannelli solari, sia per fornire una adeguata alimentazione, là dove non è disponibile la rete elettrica, che per garantire una continuità di funzionamento della strumentazione. In particolare, nei siti in cui vi sono esclusivamente sistemi fotovoltaici, le batterie vengono sfruttate con cicli continui di carica e scarica e quindi sono soggette ad un inevitabile e progressivo deterioramento. Conoscere lo stato delle batterie è quindi fondamentale per un corretto funzionamento di tutta la strumentazione.

Lo Stato Di Carica (SOC) misurato in percentuale dalla sua capacità, è un parametro fondamentale per capire lo stato di una batteria. Infatti, al termine di un ciclo di carica, il SOC indica esattamente il livello di efficienza di una batteria. Il metodo più semplice per misurarlo è controllare il voltaggio della batteria, ma è una soluzione poco affidabile. Infatti, nel caso in cui la componente chimica di una batteria sia in via di deterioramento e l'accumulatore è posto sotto carica, la tensione può comunque portarsi ad un buon livello, mentre lo stato di carica potrebbe non essere al 100%. Inoltre, all'aumentare della temperatura, aumenta il voltaggio a circuito aperto, mentre a basse temperature la tensione della batteria diminuisce. In queste condizioni quindi, la misura del voltaggio può indicare uno stato di carica errato e la batteria non renderà la sua vera capacità.

Un metodo migliore per la misura del reale stato di carica di una batteria è la misura continua della sua corrente, integrata nel tempo. In questo modo, attraverso cicli di carica e scarica completa, è possibile verificare lo stato di una batteria misurandone la carica accumulata e rilasciata. Questa operazione può essere eseguita anche durante il suo normale funzionamento, ma potrebbe alla lunga condurre a risultati falsati per le perdite di potenza per effetto Joule.

Un altro metodo è l'"impedance track" o "fuel gauge" della Texas Instruments (*gas gauge* o *fuel gauge* è infatti lo strumento che indica il livello del carburante in un serbatoio). Il metodo, implementato in un algoritmo, utilizza l'impedenza dell'accumulatore misurata dalle sue celle come chiave fondamentale per calcolare la capacità residua. L'impedenza viene misurata e memorizzata in tempo reale come una funzione dello stato di carica. Il profilo di impedenza in tempo reale, con le misure in tensione a circuito aperto (OCV), permettono il calcolo della curva di scarica dell'accumulatore. In fase di carica e scarica, l'algoritmo utilizza il metodo della misura di corrente integrata nel tempo (columb counting), mentre in modalità passiva (senza correnti di carica e scarica) l'algoritmo sfrutta le misure in tensione a circuito aperto (OCV) per calcolare lo stato di carica in qualsiasi condizione di funzionamento.

1. Descrizione generale

Il sistema qui descritto, che chiameremo "Battery Gas Gauge" (BGG), è stato realizzato per monitorare le grandi batterie di accumulatori utilizzate negli osservatori geomagnetici e tenere sotto controllo ogni singolo accumulatore utilizzato. Essendo montato su ogni accumulatore, il sistema permette di evidenziare ogni malfunzionamento, evitando di effettuare dei controlli manuali attraverso delle misure specifiche. È stato progettato per lavorare anche con accumulatori di grande capacità, oltre i 120Ah. Il circuito di forma rettangolare (visibile in Figura 10) è stato realizzato per essere montato direttamente sul polo negativo del singolo accumulatore attraverso un collegamento sulla pista elettrica del circuito (*pad*) di dimensioni adeguate per essere avvitato direttamente sul polo della batteria e nel contempo per consentire il passaggio di grandi correnti. Un secondo *pad* permette di collegare il carico o essere utilizzato come riferimento comune per il collegamento in parallelo con altre batterie (fig. 1). Infine un connettore permette il collegamento del circuito al polo positivo. Nella Figura 1 sono schematizzati i collegamenti per l'installazione. Avendo pacchi di batterie con più accumulatori in parallelo, sono necessari dei diodi di separazione (*input ed output diodes*) opportunamente dimensionati, affinché ogni batteria sia separata dall'altra ed ogni *Gas Gauge* possa rilevare i parametri relativi alla batteria su cui è montato, senza essere influenzato da quelle in parallelo.

Sulle schede sono inoltre presenti 5 diodi led per l'indicazione locale dello stato di carica di ogni singola batteria, semplificando le operazioni di verifica da parte di un operatore che, controllando i led accesi, può valutare in modo semplice e sul posto, l'efficienza delle batterie. Tutti i dispositivi BGG sono inoltre dotati di un'interfaccia I^2C per la comunicazione: con un semplice collegamento a 3 fili si possono quindi ricevere

tutte le informazioni acquisite dai BGG. Nel nostro caso, questo viene fatto con una scheda digitale programmabile *Rabbit 3710* http://ftp1.digi.com/support/documentation/019-0136_L.pdf) che, opportunamente programmata, interroga continuamente ogni singolo BGG. Inoltre la scheda *Rabbit* è anche dotata di un server *http* ed è quindi possibile creare pagine *web* sulle quali mostrare i parametri più importanti di ogni accumulatore e verificarne lo stato sia da remoto sia localmente con un computer dedicato.

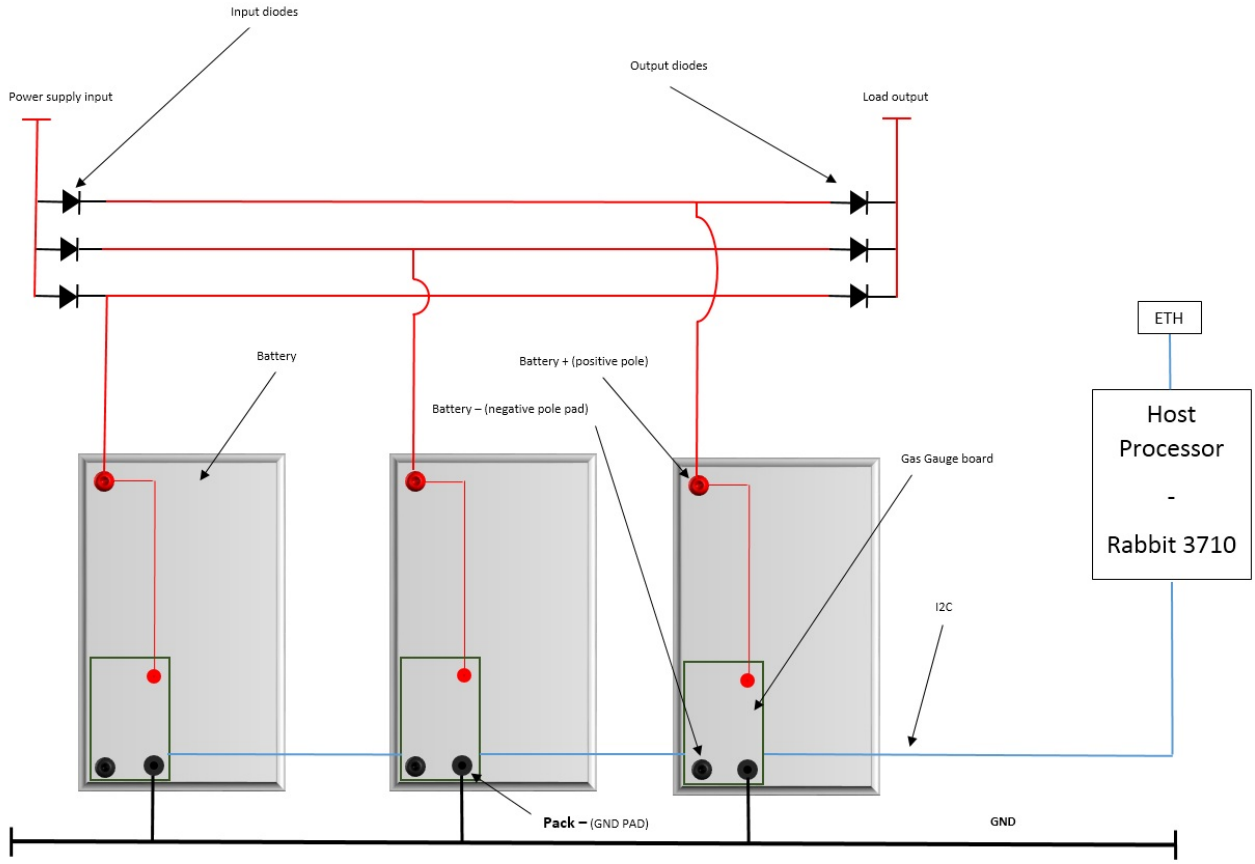


Figura 1. Schema illustrativo di collegamento.

2. Il circuito integrato di controllo - bq34z110

Il BGG è stato realizzato mediante l'utilizzo dell'integrato **bq34z110** della Texas Instruments. L'integrato può calcolare accuratamente la capacità reale della batteria, nonché il suo stato di carica. Il voltaggio, la corrente, la temperatura e diversi altri parametri sono raccolti in una memoria flash interna; come detto, è anche integrata un'interfaccia I^2C per la comunicazione. L'integrato è studiato per l'utilizzo con varie tecnologie di batterie, come quella al litio, ma è particolarmente indicato per la chimica delle batterie al piombo.

Il bq34z110 lavora con un algoritmo ideato dalla Texas Instruments chiamato "Impedance Track™" (www.ti.com/lit/an/slua450/slua450.pdf) che misurando l'impedenza delle celle, permette di conoscere istantaneamente lo stato di carica della batteria con buona approssimazione. Il procedimento si basa sul *Depht Of Dishcarge* (DOD) che è un metodo alternativo e complementare al SOC. La capacità totale della batteria è ricavata attraverso il confronto tra lo stato di carica prima e dopo l'applicazione del carico. Quando si applica un carico l'impedenza delle celle è misurata comparando il voltaggio a circuito aperto (OCV) con il voltaggio misurato sotto carico. La misura dell'OCV e le misure di corrente integrate nel tempo (Columb counting) determinano la capacità totale della batteria. La capacità totale iniziale, presa dalle caratteristiche tecniche della batteria (*datasheet*), deve essere impostata (*Qmax*). Effettuando diversi cicli di carica e scarica (columb counting), la precisione di calcolo aumenta. Nella Figura 2 è possibile vedere lo schema base per i

collegamenti tratto dal *datasheet* dell'integrato della Texas Instruments. Il dispositivo misura la corrente monitorando l'attività di carica e scarica attraverso una piccola resistenza di precisione (*Sense Resistor* - da 5mΩ a 20mΩ a seconda della capacità della batteria e quindi delle correnti in gioco) collegata tra i pin SRP e SRN (fig. 2) ossia in serie tra il negativo della batteria ed il negativo della tensione di carica. Il circuito in Figura 2 mostra una configurazione base per accumulatori a singola cella (2V), quindi la tensione può essere applicata direttamente sui pin REGIN e CE. Per accumulatori a 12V, aventi 6 celle da 2V in serie, la tensione viene ripartita attraverso una rete elettrica visibile nello schema del BGG in Figura 8. Per batterie di elevata capacità si può attivare la modalità X10 (tramite una configurazione di registro), che permette di raggiungere capacità di lettura per batterie fino a 327Ah.

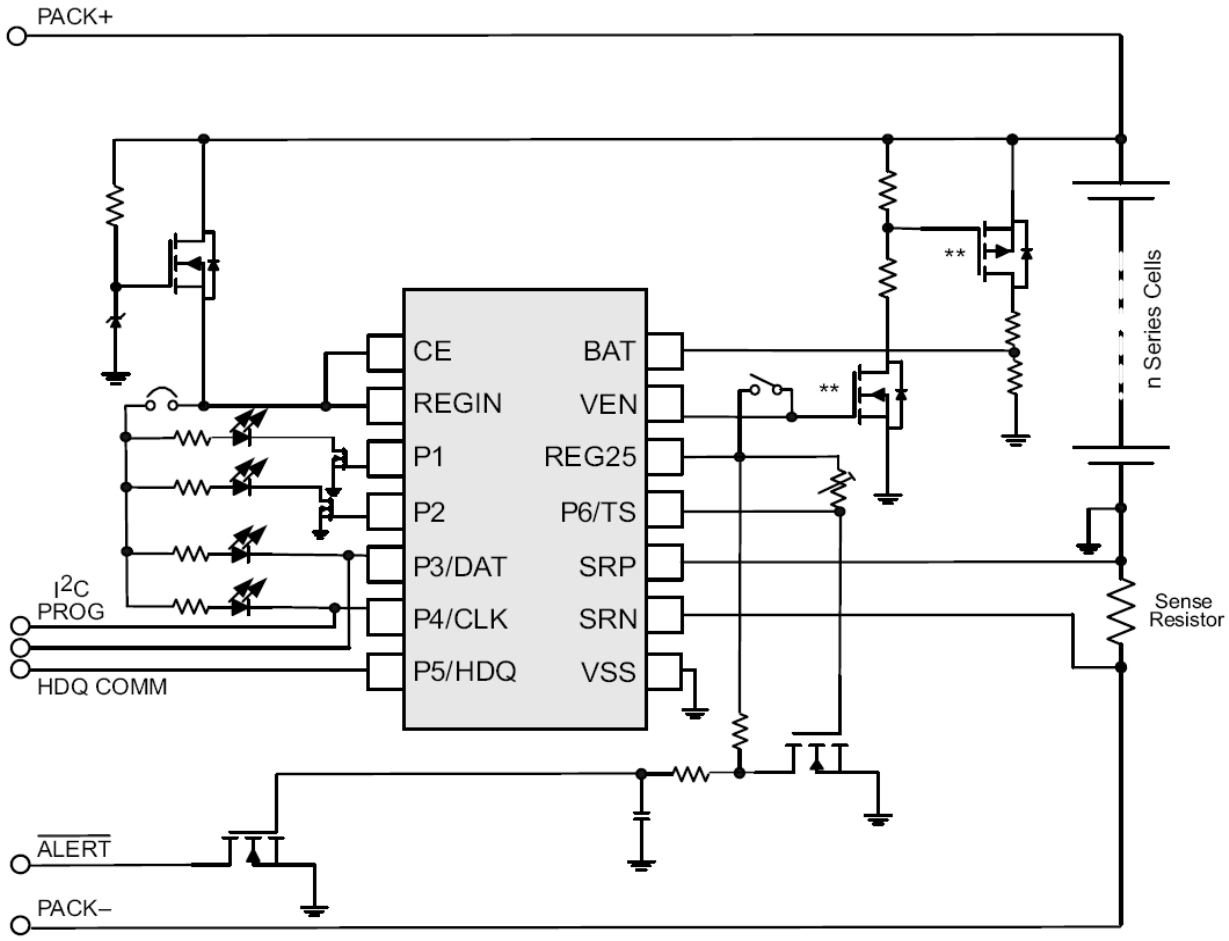


Figura 2. bq34z110 – Schema di base.

3. Configurazione e calibrazione

Una volta montato, il bq34z110 può essere facilmente calibrato e configurato attraverso l'*evaluation* software della Texas Instruments (fig. 3). Le impostazioni di fabbrica sono già adatte per lavorare con le batterie al piombo quindi, affinché l'*Impedance Track*™ funzioni regolarmente, basta modificare pochi parametri nel tab DataFlash:

- il *Qmax (capacità chimica totale)* nelle impostazioni di *gas gauge* rappresenta la quantità di carica in mAh della batteria, parametro che può essere preso dai *datasheet* delle batterie;

- il *Design Cell Capacity* e il *Design Cell Energy* che rappresentano la quantità di carica in mAh della singola cella e la quantità di energia in mWh rispettivamente. Qual ora si lavorasse con batterie maggiori di 32Ah, va attivata la modalità X10, impostandola nel *Pack Configuration* e nel *Design Energy Scale*;
- il registro *Led Comm Configuration* permettere di scegliere quale modalità di indicazione a led utilizzare o se attivarli tramite il pulsante montato sulla scheda;
- il *Led Hold Time* con il quale si può anche decidere per quanti secondi far restare attivi i led dopo aver premuto il pulsante di visualizzazione.

In Figura 3 è possibile vedere la schermata del *data ram* del dispositivo: sono evidenziati i bit *VOK* e *QEN* in *Control Status – SCANNING* che indicano che *l'Impedance Track* è attivo e i bit *X10* e *VOLSEL* nel *Pack Configuration - SCANNING* che indicano l'utilizzo della modalità ad alto amperaggio e la presenza della rete elettrica divisoria per batterie a 12V, quindi con 6 celle in serie. Dopo aver configurato i registri più rilevanti, si può calibrare il dispositivo attraverso la pagina di calibrazione dell'*Evaluation* software mostrata nella Figura 4.

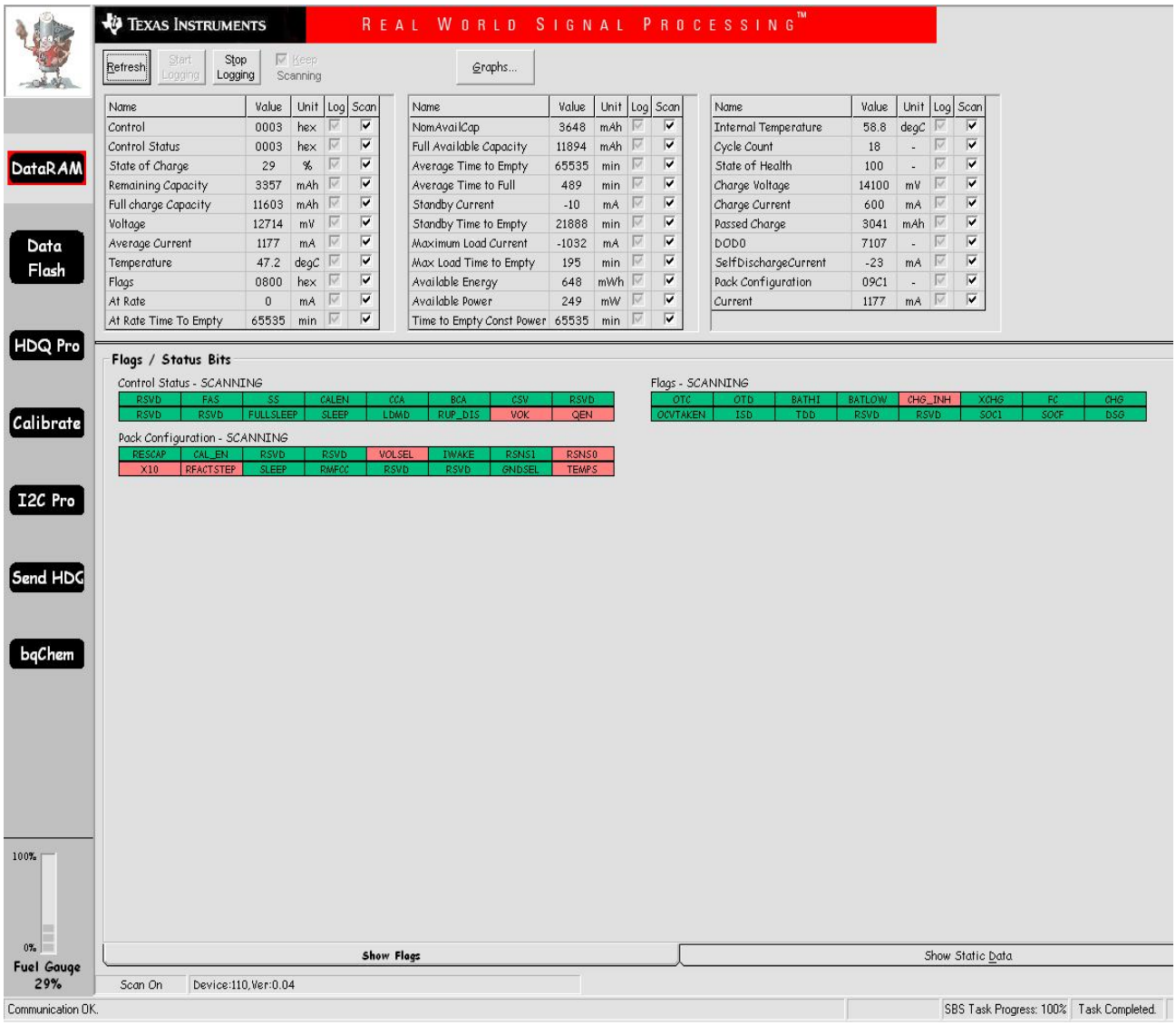


Figura 3. Schermata del Data Ram – *Evaluation* software.

Collegando il dispositivo a una batteria, sulla pagina di calibrazione si può leggere la tensione che, verificata con un voltometro, può essere calibrata attraverso il pulsante "*calibrate voltage*" se non è esattamente corrispondente. Nello stesso modo, utilizzando un amperometro e un carico sulla batteria, verrà generata una corrente negativa di scarica e si potrà calibrare la corrente misurata dal dispositivo con "*calibrate pack current*". Per una calibrazione più accurata del dispositivo si possono eseguire tutti i passaggi step-by-step (fig. 4). A questo punto il *gas gauge* è in grado di calcolare lo stato della batteria utilizzando la misura di impedenza. Effettuando vari cicli di lavoro in carica e scarica, le misure saranno sempre più precise.

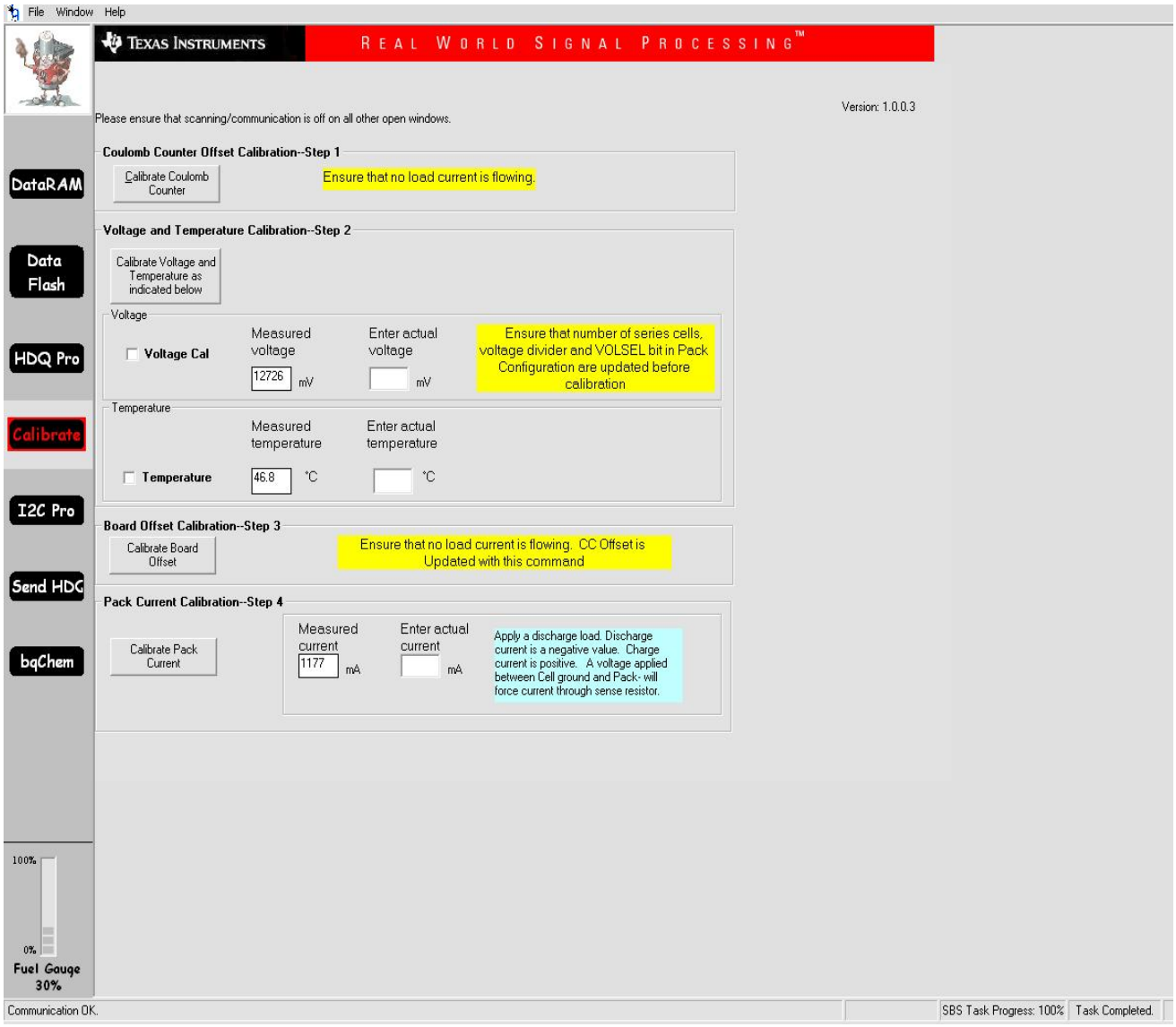


Figura 4. Schermata di calibrazione – *Evaluation software*.

4. Data logging

Il BGG è stato progettato per l'osservatorio geomagnetico di Lampedusa, per monitorare i pacchi batteria dell'osservatorio, ma può anche essere anche utilizzato come strumento per esaminare una o più batterie in laboratorio.

I grafici sottostanti di Figura 5 rappresentano un *data logging* registrato con il *gas gauge* su un ciclo di lavoro in scarica e carica di due batterie. In essi è possibile vedere il ciclo di lavoro di una batteria da 5Ah in scarica e carica veloce a circa 2A. Nel grafico il voltaggio (mV) e lo stato di carica (%) diminuiscono costantemente mentre la corrente di scarica è fissa a circa -2A. Quando la corrente sale a circa +2A il voltaggio e lo stato di carica iniziano a salire simultaneamente per poi stabilizzarsi. Dallo stato di carica si vede che la batteria raggiunge il 100%, quindi si deduce che sia in buone condizioni.

In Figura 6 si vede il ciclo di lavoro di una batteria da 120Ah in scarica a corrente costante di -10A. Il voltaggio e lo stato di carica diminuiscono simultaneamente. Applicando una corrente di carica di circa +12A, voltaggio e stato di carica crescono per poi stabilizzarsi. Il voltaggio raggiunge il livello di carica di circa 13,5V, mentre lo stato di carica si stabilizza a circa l'80%. Questa informazione ci dice che la batteria è leggermente deteriorata e non fornirà la sua totale capacità di carica.

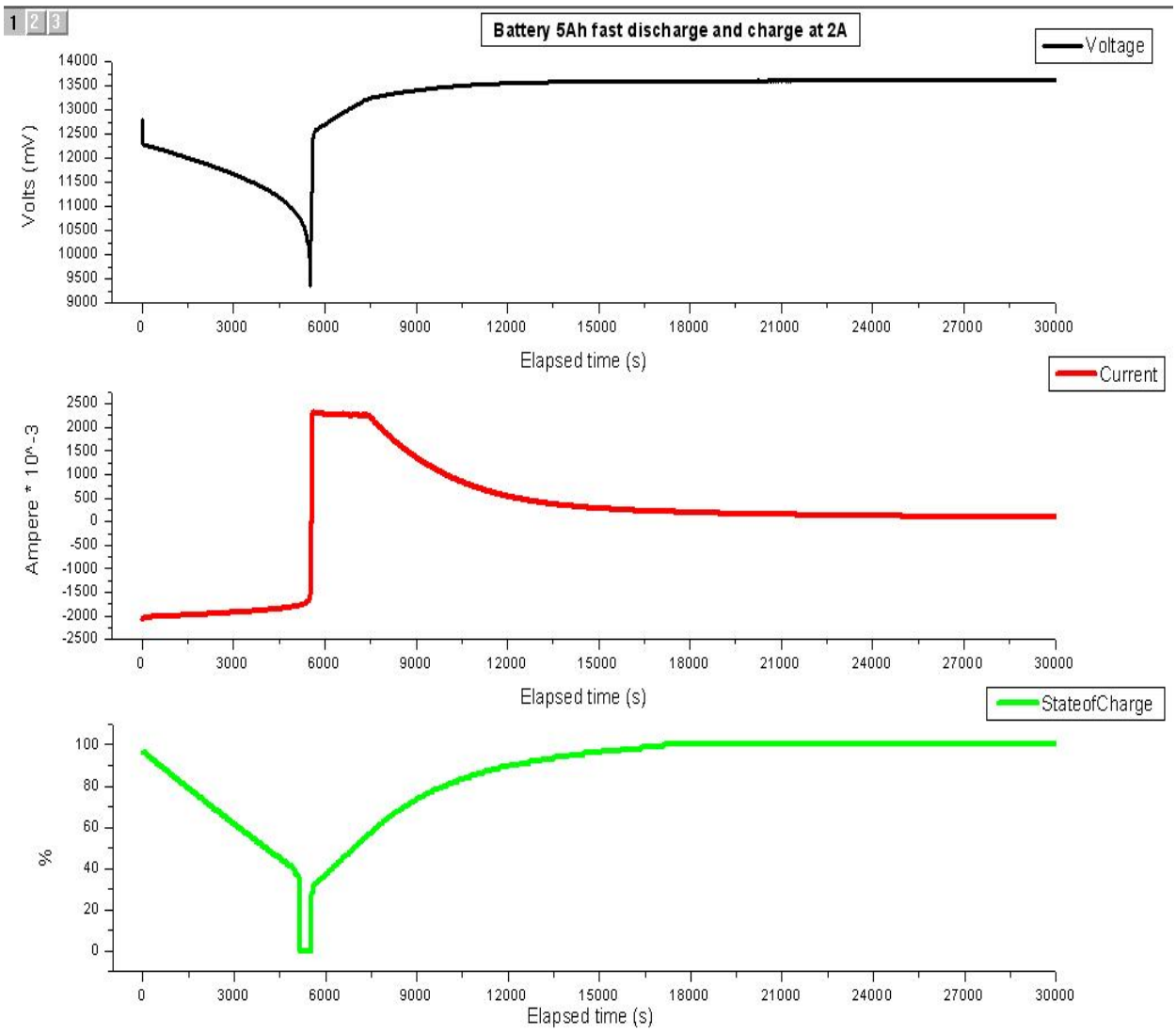


Figura 5. Ciclo di lavoro di una batteria da 120Ah leggermente deteriorata.

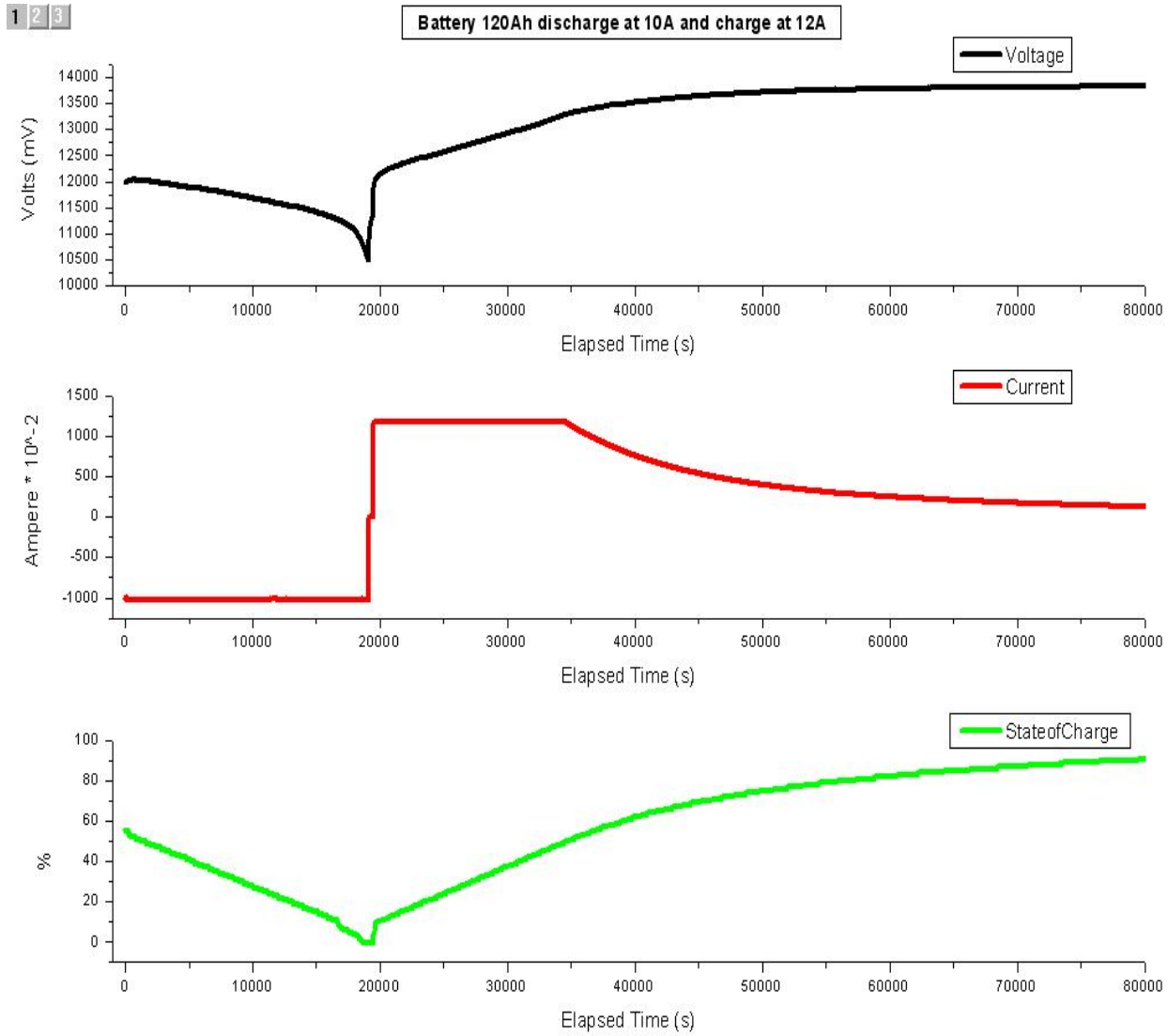


Figura 6. Ciclo di lavoro di una batteria da 120Ah.

5. L'interfaccia I^2C

L'interfaccia I^2C di cui è dotato l'integrato bq34z110 è un sistema seriale bifilare (con massa), che permette la comunicazione tra un dispositivo *master* e vari dispositivi *slave* collegati in cascata. Il protocollo richiede due sole linee di comunicazione chiamate SDA per i dati e SCL per il clock. Una terza linea va aggiunta come riferimento di massa comune.

Per leggere o scrivere i dati sui dispositivi *slave* tramite l'I^2C, occorrono 4 parametri, cioè lo *slave address* che rappresenta l'indirizzo generale per raggiungere un determinato dispositivo, *l'EEPROM address* che rappresenta l'indirizzo del registro del dispositivo *slave* su cui scrivere o leggere, il *data buffer* che contiene l'informazione e il *length of data* che indica la lunghezza dell'informazione trasmessa.

Il bq34z110 presenta però una seria limitazione per quanto riguarda la comunicazione tra più dispositivi. Ha, infatti, un indirizzo generale *slave address* fisso a 0xAA, che non è possibile modificare. La comunicazione tra più *gas gauge* non è quindi possibile. Per risolvere il problema è stato utilizzato l'integrato **LTC4302** della Linear Technology (*addressable I^2C*) che permette, come un'interfaccia aggiunta al circuito, di configurare un indirizzo differente per lo *slave* ed avviare la comunicazione sulle linee SDA e SCL tra più dispositivi.

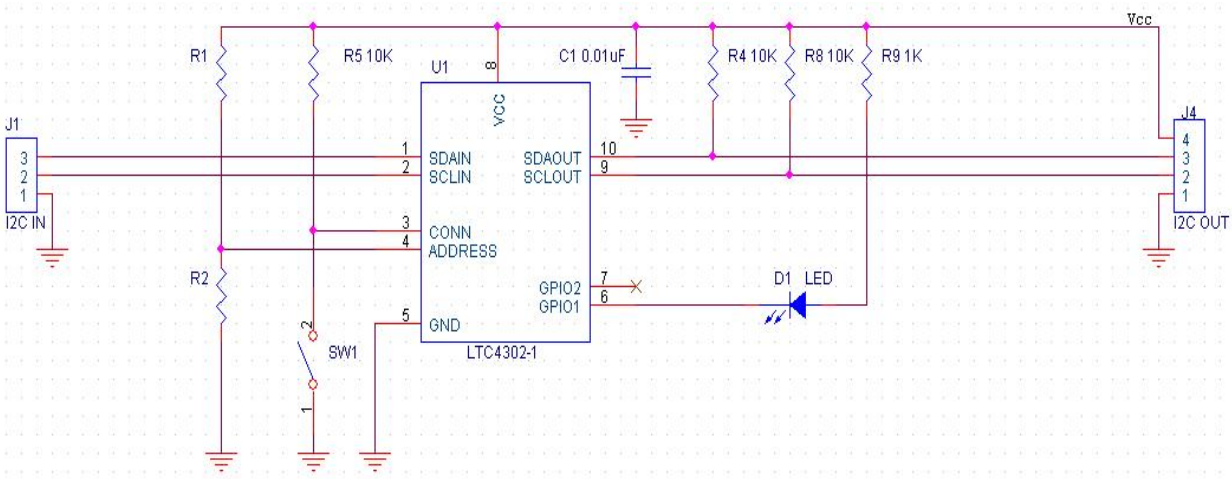


Figura 7. Schema elettrico dell'interfaccia I^2C dal *datasheet* della Linear Technology.

Come si vede nello schema elettrico di Figura 7, l'indirizzo è configurabile attraverso il partitore resistivo R1 e R2 collegato al piedino 4 (*pin address)* dell'integrato LTC4302. Il voltaggio sul *pin address* è convertito in 5 bit dall'ADC interno al dispositivo. Usando resistenze con 1% di tolleranza è possibile scegliere fra 32 valori di indirizzi diversi, visibili nella Tabella 1 (dal *datasheet* della Linear Technology). Sulla porta GPIO1 è stato inoltre inserito un led che viene acceso da programma dal *master*, in modo da evidenziare quale *slave* si trova in comunicazione. Lo schema elettrico in Figura 8 è stato integrato nel progetto finale.

ADDRESS CODE	$R_{1(TOP)}$ RESISTOR	$R_{2(BOTTOM)}$ RESISTOR	5V IDEAL VOLTAGE	ALLOWED ADDRESS VOLTAGE RANGE	3.3V IDEAL VOLTAGE	ALLOWED ADDRESS VOLTAGE RANGE
00	8660	137	0.078125	0.076 to 0.079	0.051563	0.050 to 0.052
01	2800	137	0.234375	0.229 to 0.238	0.154688	0.151 to 0.157
02	1180	100	0.390625	0.383 to 0.398	0.257813	0.253 to 0.263
03	1370	169	0.546875	0.539 to 0.559	0.360938	0.356 to 0.369
04	1070	174	0.703125	0.687 to 0.711	0.464063	0.454 to 0.470
05	1070	221	0.859375	0.842 to 0.870	0.567188	0.556 to 0.574
06	4120	1050	1.015625	0.999 to 1.032	0.670313	0.660 to 0.681
07	3320	1020	1.171875	1.157 to 1.193	0.773438	0.764 to 0.788
08	3160	1150	1.328125	1.315 to 1.354	0.876563	0.868 to 0.893
09	6490	2740	1.484375	1.464 to 1.505	0.979688	0.966 to 0.993
10	2150	1050	1.640625	1.619 to 1.663	1.082813	1.068 to 1.097
11	2050	1150	1.796875	1.774 to 1.820	1.185938	1.171 to 1.201
12	2150	1370	1.953125	1.922 to 1.970	1.289063	1.269 to 1.300
13	1960	1430	2.109375	2.085 to 2.134	1.392188	1.376 to 1.408
14	2100	1740	2.265625	2.241 to 2.290	1.495313	1.479 to 1.512
15	2000	1870	2.421875	2.391 to 2.441	1.598438	1.578 to 1.611
16	1870	2000	2.578125	2.559 to 2.609	1.701563	1.689 to 1.722
17	1740	2100	2.734375	2.710 to 2.759	1.804688	1.788 to 1.821
18	1430	1960	2.890625	2.866 to 2.915	1.907813	1.892 to 1.924
19	1370	2150	3.046875	3.030 to 3.078	2.010938	2.000 to 2.031
20	1150	2050	3.203125	3.180 to 3.226	2.114063	2.099 to 2.129
21	1050	2150	3.359375	3.337 to 3.381	2.217188	2.203 to 2.232
22	2740	6490	3.515625	3.495 to 3.537	2.320313	2.307 to 2.334
23	1150	3160	3.671875	3.646 to 3.685	2.423438	2.407 to 2.432
24	1020	3320	3.838125	3.807 to 3.843	2.526563	2.512 to 2.536
25	1050	4120	3.984375	3.968 to 4.001	2.629688	2.619 to 2.640
26	221	1070	4.140625	4.130 to 4.158	2.732813	2.726 to 2.744
27	174	1070	4.296875	4.289 to 4.313	2.835938	2.830 to 2.846
28	169	1370	4.453125	4.441 to 4.461	2.939063	2.931 to 2.944
29	100	1180	4.609375	4.602 to 4.617	3.042188	3.037 to 3.047
30	137	2800	4.765625	4.762 to 4.771	3.145313	3.143 to 3.149
31	137	8660	4.921875	4.921 to 4.924	3.248438	3.248 to 3.250

Tabella 1. Lista degli indirizzi.

6. Schema elettrico e layout

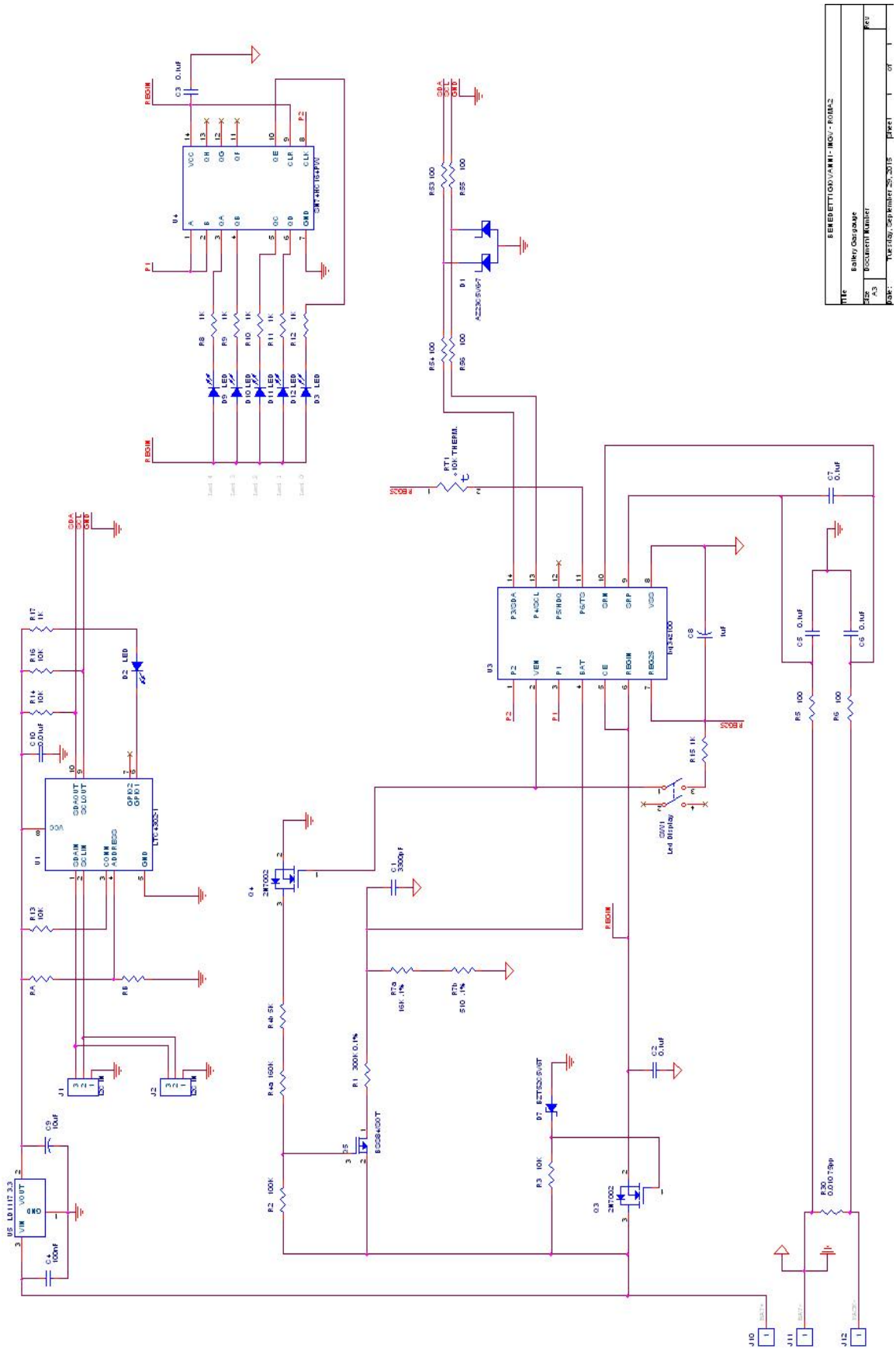


Figura 8. Schema elettrico del *gas gauge*.

Lo schema elettrico mostrato in Figura 8 rappresenta lo schema di una configurazione del bq34z110 in modalità multi cella, per lavorare con batterie al piombo da 12V. La tensione dal connettore J10 (BAT+) viene ripartita attraverso la rete con elementi di precisione per l'alimentazione del circuito e per creare le tensioni di riferimento. La resistenza per il monitoraggio della corrente R30 si trova tra i contatti BAT- e PACK- (J11 e J12 rispettivamente). Sono presenti i due connettori per la comunicazione I^2C (J1, J2) in modo da collegare facilmente più dispositivi in cascata. Attraverso il registro a scorrimento SN74Hc164PW della Texas Instruments si pilotano 5 led per la visualizzazione dello stato di carica SOC. Ogni led rappresenta il 20% della carica. Il dispositivo prevede un pulsante (SW1) per visualizzare lo stato dei led attivandoli per un controllo locale e per tenerli normalmente spenti al fine di limitarne il consumo di corrente.

Il layout, mostrato nelle Figure 9-11, è stato progettato per montare il circuito sulla parte superiore della batteria, con un contatto di massa a vite sul polo negativo e dimensionato per la maggior parte dei connettori usati nelle batterie al piombo di elevata capacità. Tutti i componenti sono SMD, tranne i connettori J1 e J2, il led di identificazione I^2C ed il sensore di temperatura. Sulle estremità ci sono dei fori di fissaggio per un'eventuale scatola di protezione. Il disegno dello schema elettrico e il layout sono stati realizzati con *Orcad,* mentre il prototipo del PCB (circuito stampato in Figura 9) è stato creato utilizzando la fresa meccanica *Colimbus* utilizzata nei nostri laboratori INGV di geomagnetismo. Successivamente le schede del *gas gauge* sono state prodotte da una fabbrica specializzata, per avere dei PCB industriali con materiali migliori e più resistenti (fig. 10).

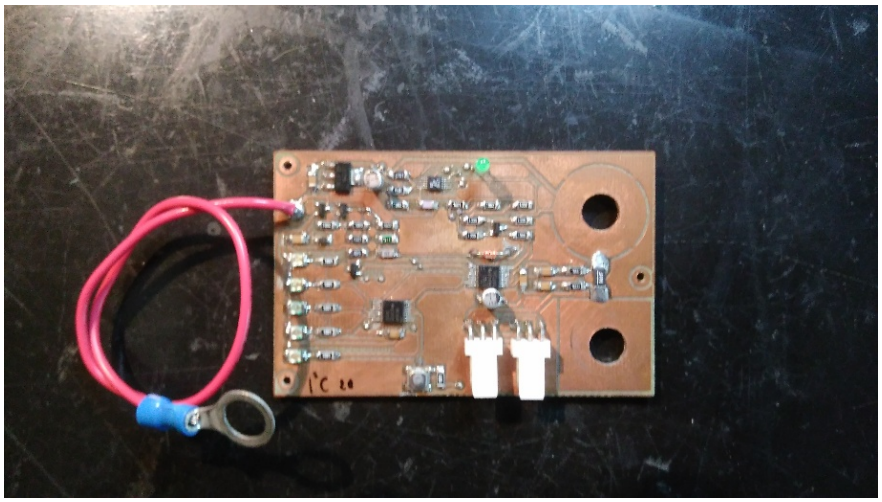


Figura 9. Il prototipo realizzato con la fresa meccanica per PCB, nel laboratorio INGV di geomagnetismo.

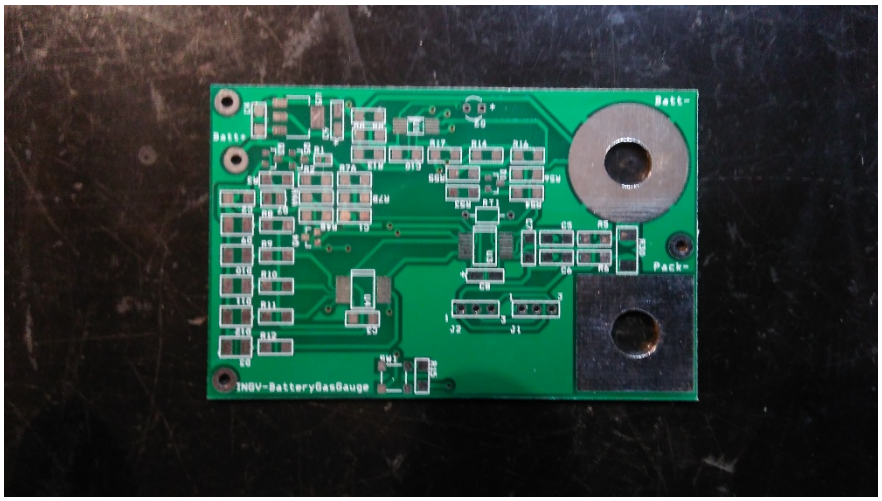
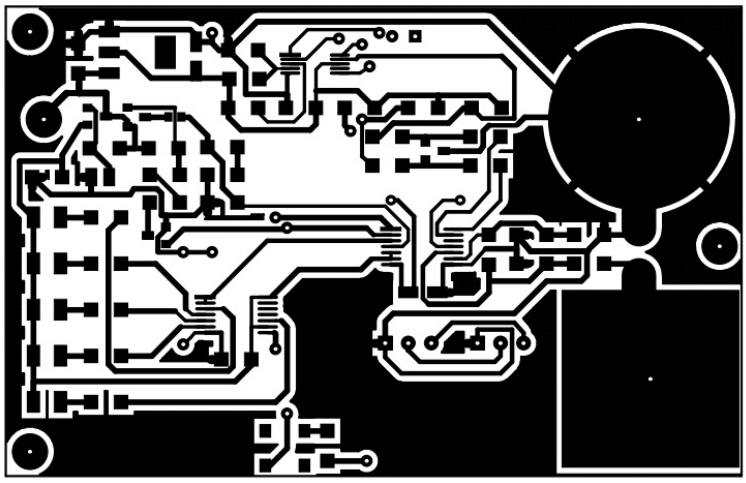
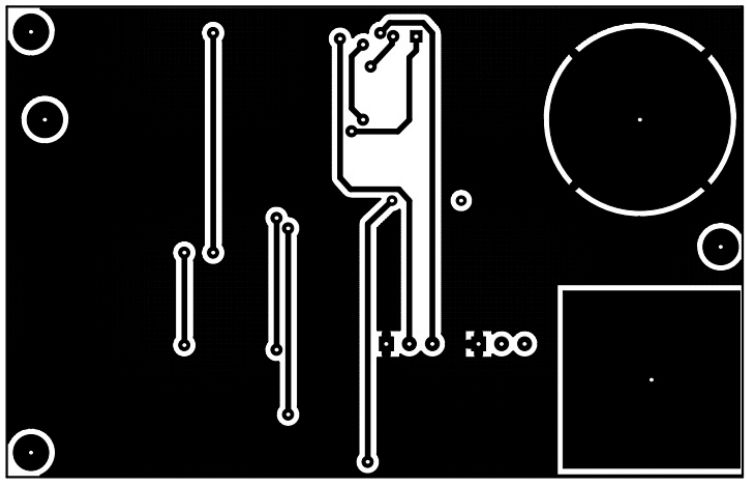


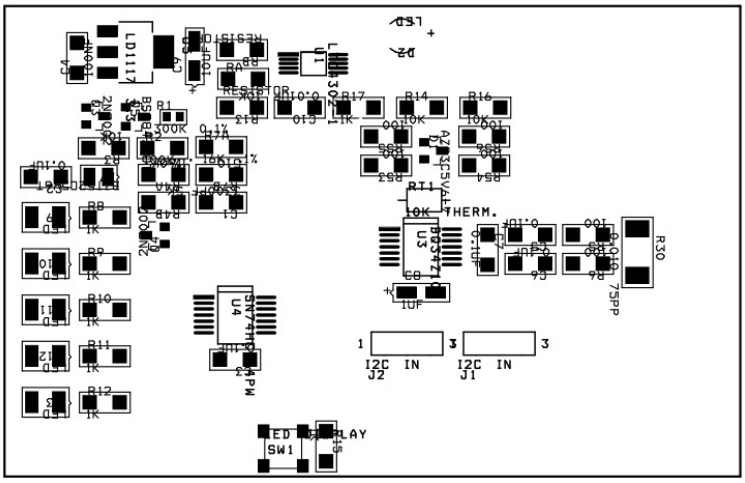
Figura 10. Il PCB del *gas gauge* realizzato in fabbrica.



Top



Bottom



Assembly top

Figura 11. Layout del *gas gauge*.

7. L'*host processor* per il controllo remoto e il server http

Attraverso l'unità programmabile *Rabbit 3710,* che funge da dispositivo *master*, è possibile interrogare da remoto tutti i *gas gauge* collegati ad esso e leggere le informazioni sullo stato delle batterie.

Figura 12. L'host processor.

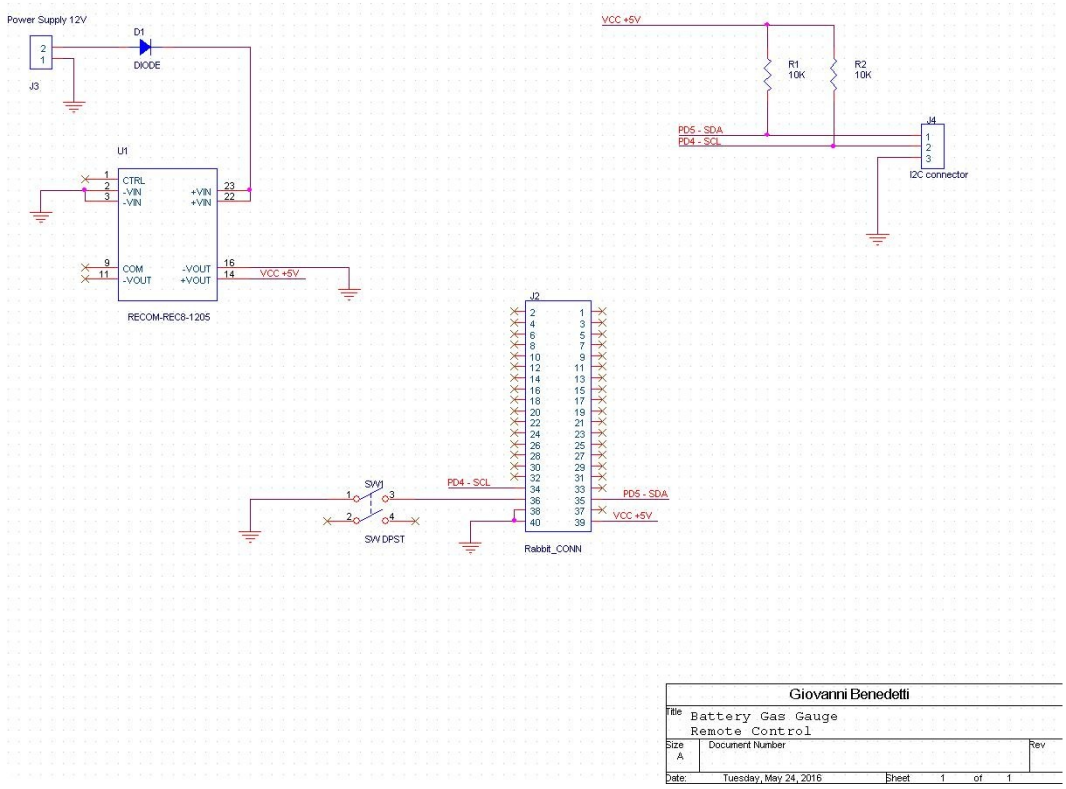
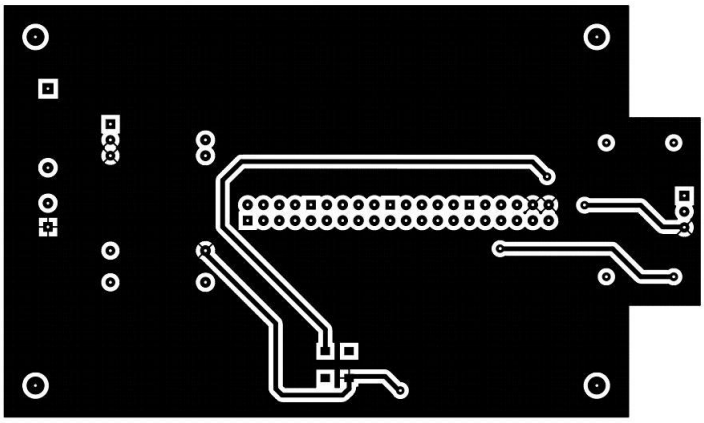
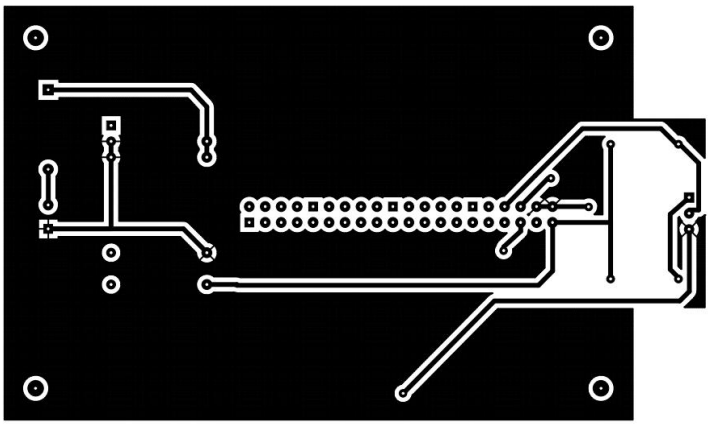


Figura 13. Schema elettrico della scheda dell'host processor.

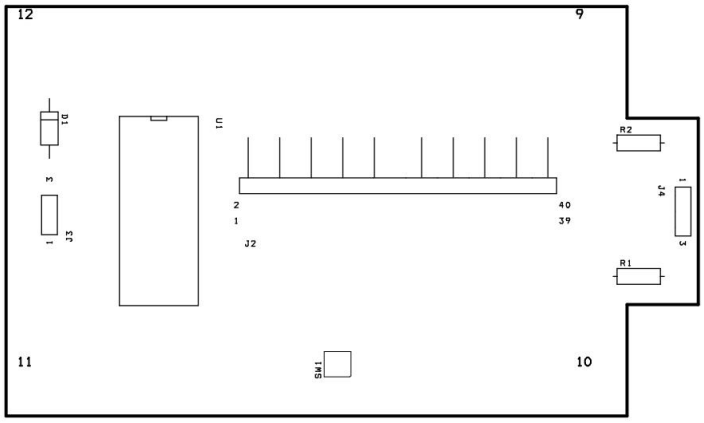
L'unità *Rabbit 3710* è stata montata su una scheda progettata e realizzata nei laboratori di geomagnetismo dell'INGV. Come si vede dallo schema elettrico in Figura 13, sulla scheda sono presenti un DC-DC *converter* per regolare l'alimentazione del dispositivo ed un connettore per le linee I^2C connesse ai resistori di *pull-up* come previsto dallo standard. L'unità programmabile *Rabbit* monta un connettore ethernet per il collegamento in rete. In Figura 14 è mostrato il layout della scheda di alimentazione progettato con *Orcad*. La forma della scheda permette ai connettori di rete ed I^2C una giusta posizione dei collegamenti posti sul contenitore esterno. La scheda è stata realizzata mediante la fresa meccanica per PCB *Colimbus*.



Top



Bottom



Assembly top

Figura 14. Layout della scheda host processor.

Sul dispositivo *Rabbit* è implementato un server *http* raggiungibile attraverso un indirizzo impostato coerentemente agli indirizzi di rete dell'osservatorio di Lampedusa. In Figura 15 viene mostrata la pagina *web* del *Battery Gas Gauge*, dove, in una struttura a tabella, vengono mostrati diversi parametri per ogni batteria in funzione. Monitorando la tensione, le correnti di carica e scarica, lo stato di carica e la temperatura di funzionamento, si può tenere sotto controllo lo stato degli accumulatori e stabilire quando è necessario svolgere un intervento di manutenzione sui pacchi batteria della stazione.

BATTERY GAS GAUGE - OSSERVATORIO GEOMAGNETICO DI LAMPEDUSA

Name	Value	Description
V Battery1-System1	13464	mV
A Battery1-System1	50	A * 10^-2
SOC Battery1-System1	100	%
T Battery1-System1	30.50	°C
V Battery2-System1	13446	mV
A Battery2-System1	33	A * 10^-2
SOC Battery2-System1	97	%
T Battery2-System1	29.90	°C
V Battery3-System1	13440	mV
A Battery3-System1	34	A * 10^-2
SOC Battery3-System1	64	%
T Battery3-System1	29.90	°C
V Battery1-System2	13404	mV
A Battery1-System2	34	A * 10^-2
SOC Battery1-System2	100	%
T Battery1-System2	29.60	°C
V Battery2-System2	13398	mV
A Battery2-System2	29	A * 10^-2
SOC Battery2-System2	99	%
T Battery2-System2	29.20	°C
V Battery3-System2	13392	mV
A Battery3-System2	29	A * 10^-2
SOC Battery3-System2	100	%
T Battery3-System2	29.10	°C

Figura 15. La pagina *web* per il monitoraggio delle batterie dell'osservatorio geomagnetico di Lampedusa.

8. L'installazione presso l'Osservatorio Geomagnetico di Lampedusa

L'installazione del BGG nell'Osservatorio Geomagnetico di Lampedusa è avvenuta nel mese di Aprile 2017. L'osservatorio è stato realizzato presso un'area remota dell'isola ed è privo di energia elettrica. Tutti i dispositivi sono mantenuti in funzione da pannelli fotovoltaici e batterie al piombo. In questo osservatorio è cruciale conoscere da remoto lo stato di ogni singolo accumulatore, in modo da monitorare il deterioramento delle batterie ed intervenire prima che la stazione resti priva di alimentazione. Nelle Figure 16 e 17 è possibile vedere la struttura di tutto il sistema di monitoraggio delle batterie. La stazione geomagnetica è dotata di due sistemi paralleli di misura e acquisizione dati, ciascuno aventi 3 batterie da 12V al piombo da 100Ah. Sono stati realizzati due box in legno per contenere la circuiteria del *gas gauge*. Come è visibile nelle figure, sul pavimento sono montati dei binari in teflon per permettere ai pesanti box delle batterie di essere spostati avanti e indietro per effettuare i controlli e gli interventi necessari. Come diodi di separazione per ogni batteria, sono stati utilizzati diodi di potenza a vite *Vishay*, montati su adeguati dissipatori. I diodi di ingresso utilizzati sono da 15A ciascuno con caratteristica di catodo a vite, mentre in uscita sono stati utilizzati diodi da 12A ciascuno con caratteristica di anodo a vite. Questo per garantire un adeguato passaggio delle correnti di carica e scarica delle batterie. Per la realizzazione del sistema sono stati utilizzati

esclusivamente materiali amagnetici per non introdurre interferenze sui dati misurati dalla strumentazione di monitoraggio.

Il circuito del BGG realizzato, ha un consumo di circa 10mA anche con i 5 LED di controllo accesi. Il BGG è quindi un circuito *low power* che non influisce sulla vita delle batterie su cui è montato, anche quando non ci sono correnti di carica e scarica attive sugli accumulatori.

Il sistema si è rivelato molto utile al fine del monitoraggio delle batterie nell'osservatorio geomagnetico di Lampedusa. Infatti, negli anni passati, è capitato che alcuni accumulatori fossero deteriorati e che alcuni apparati smettessero di funzionare improvvisamente. In questo modo è possibile prevedere con largo anticipo un eventuale intervento presso l'isola di Lampedusa, senza compromettere la continuità dei dati magnetici ed il funzionamento dell'intero osservatorio.

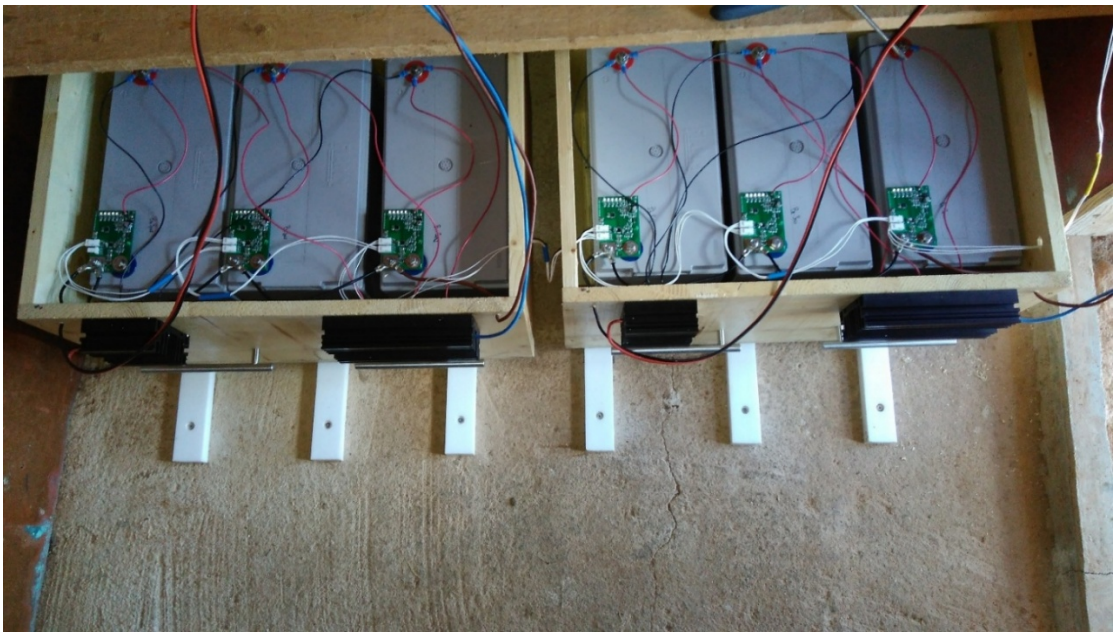


Figura 16. Foto dell'installazione del BGG presso l'Osservatorio geomagnetico di Lampedusa.

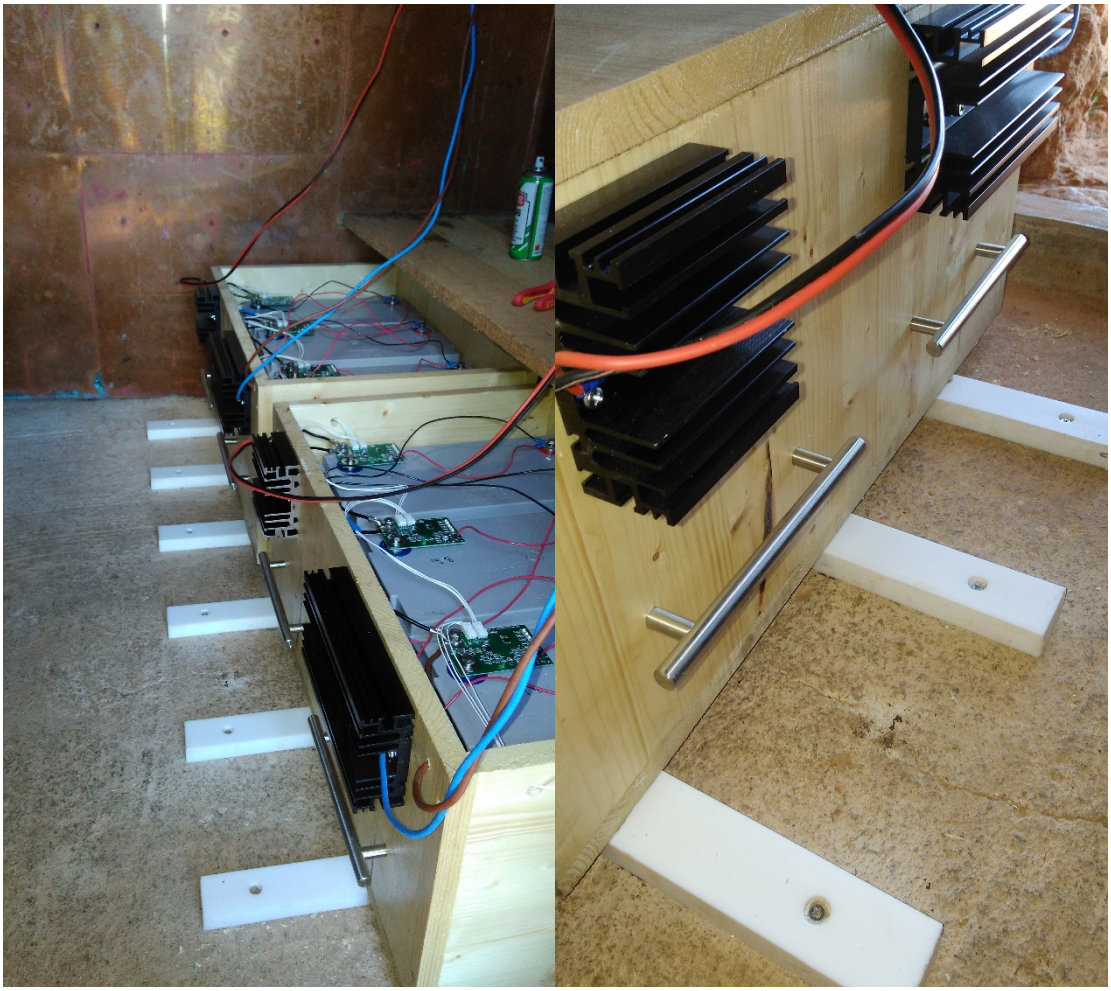


Figura 17. Foto dell'installazione del BGG presso l'Osservatorio geomagnetico di Lampedusa.

ISSN 2039-7941
Anno 2017_Numero 388

Appendice A

Il modulo programmabile *Rabbit 3710* permette di implementare un server *http* e la comunicazione I^2C attraverso le sue librerie, in cui sono state fatte delle modifiche al fine di ottimizzare sia la comunicazione con i dispositivi, che il funzionamento della pagina *web*.

Il codice inizia con alcune *define* e dichiarazioni di librerie utili al funzionamento del controller remoto.

Viene eseguita una prima inizializzazione della *mimetable,* che permette di creare una tabella dinamica con i parametri letti dai vari *gas gauge* e si dichiarano poi le relative variabili. Viene riservato uno spazio in memoria per una tabella dinamica di 24 posizioni e si imposta il titolo della *web page*.

A questo punto inizia il ciclo di inizializzazione del sistema. Si invia un codice di inizializzazione dell'*Impedace Track* a tutti i dispositivi, per forzare ogni *gas gauge* ad iniziare la misura di impedenza e a mostrare una prima stima dello SOC di ogni batteria. Il processo è visibile dall'operatore grazie all'accendersi ed allo spegnersi, progressivamente, dei led di comunicazione I^2C presenti su ogni dispositivo.

Inizia poi un ciclo continuo *while (1) { }* all'interno del quale vengono interrogati ciclicamente tutti i *gas gauge* e vengono letti i parametri momentanei di voltaggio, corrente, SOC e temperatura per ogni singola batteria. I parametri letti vanno poi a riempire le relative posizioni della tabella dinamica (fig. 15). All'interno del ciclo *while* ci sono anche le istruzioni che mantengono attivo l'*http server*, per cui il sito web con la tabella dei valori delle batterie è on line. Il buon funzionamento dell'*host processor* può essere osservato dall'operatore dall'accendersi e dallo spegnersi, simultaneamente, dei led relativi alla comunicazione I^2C.

Codice sorgente

```
#class auto
////////////////////////////////////Zserver//////////////////////
#define TCPCONFIG 1
#define HTTP_MAXSERVERS 1
#define MAX_TCP_SOCKET_BUFFERS 1

#define FORM_ERROR_BUF 4096

#define HTTP_NO_FLASHSPEC
#define DISABLE_DNS

#memmap xmem
#use "dcrtcp.lib"
#use "http.lib"

//////////////////////////I2C gasgauge/////////////
#use "I2C_devices.lib"

#define WRITE_TIME 5

#define GASGAUGE_ADDRESS 0xAA
//////////////////////////////////////////////////

/* the default mime type for '/' must be first */
SSPEC_MIMETABLE_START
        SSPEC_MIME(".html", "text/html")
SSPEC_MIMETABLE_END

//Variabili parametri batterie, batterie sistema 1 e 2
int Voltage11, Voltage21, Voltage31;
int Voltage12, Voltage22, Voltage32;
int Current11, Current21, Current31;
int Current12, Current22, Current32;
int StateofCharge11, StateofCharge21, StateofCharge31;
int StateofCharge12, StateofCharge22, StateofCharge32;
float Temperature11, Temperature21, Temperature31;
float Temperature12, Temperature22, Temperature32;

 int return_code, reg1, reg2;
 unsigned long t;
 long lError; //As Long
 char yData[2]; //As Byte

 // Declare the FormVar array to hold form variable information
  static FormVar myform[24];
   int var;
   int form;

   int i;

void main(void)
{


        // Add the form (array of variables)
        form = sspec_addform("myform.html", myform, 24,
SERVER_HTTP);

        // Set the title of the form
        sspec_setformtitle(form, "BATTERY GAS GAUGE -
OSSERVATORIO GEOMAGNETICO DI LAMPEDUSA");


I2C_init();
t = MS_TIMER;
while((long)(MS_TIMER - t) < WRITE_TIME);


//////////////////////////////////////////////////Inizializzazione Impedance Track
enebled per tutte le board////////////////////////////////

////////////////imposto registri interfaccia LT su ON_ledOn//////////////////////
  reg1 = 0x80;
  reg2 = 0x03;
//////////////////////////////////////////

//////////////////////////////accendo ed imposto IT per B1-
Sys1////////////////////////////////
 return_code = I2CWrite2(0xE8, reg1 ,reg2, 2);
if(return_code)
printf("Return %d\n",return_code);
while((long)(MS_TIMER - t) < WRITE_TIME);

//invio comando inizio Impedance Track
  yData[0] = 0x21;
  yData[1] = 0x00;
 return_code = I2CWrite(GASGAUGE_ADDRESS, 0x00, yData, 2);
    if(return_code)
        printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);
```

```
for(i=0; i<=20000; i++);
//////////////////////////////////////////////////////////////


//////////////////////////accendo ed imposto IT per B2-
Sys1////////////////////////////////////
 return_code = I2CWrite2(0xD6, reg1 ,reg2, 2);
   if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);

 //invio comando inizio Impedance Track
 yData[0] = 0x21;
 yData[1] = 0x00;
 return_code = I2CWrite(GASGAUGE_ADDRESS, 0x00, yData, 2);
    if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);


for(i=0; i<=20000; i++);
//////////////////////////////////////////////////////////////


 ///////////////////////accendo ed imposto IT per B3-
Sys1/////////////////////////////////////
 return_code = I2CWrite2(0xEC, reg1 ,reg2, 2);

   if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);
 //invio comando inizio Impedance Track
 yData[0] = 0x21;
 yData[1] = 0x00;
 return_code = I2CWrite(GASGAUGE_ADDRESS, 0x00, yData, 2);
    if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);

for(i=0; i<=20000; i++);
 //////////////////////////////////////////////////////////////


 //////////////////////////accendo ed imposto IT per B1-
Sys2////////////////////////////////////
 return_code = I2CWrite2(0xD4, reg1 ,reg2, 2);

   if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);

 //invio comando inizio Impedance Track
 yData[0] = 0x21;
 yData[1] = 0x00;
 return_code = I2CWrite(GASGAUGE_ADDRESS, 0x00, yData, 2);
    if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);

for(i=0; i<=20000; i++);
//////////////////////////////////////////////////////////////


//////////////////////////accendo ed imposto IT per B2-
Sys2////////////////////////////////////
 return_code = I2CWrite2(0xCE, reg1 ,reg2, 2);

   if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);

 //invio comando inizio Impedance Track
 yData[0] = 0x21;
 yData[1] = 0x00;
 return_code = I2CWrite(GASGAUGE_ADDRESS, 0x00, yData, 2);
    if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);

for(i=0; i<=20000; i++);
 //////////////////////////////////////////////////////////////


 //////////////////////////accendo ed imposto IT per B2-
Sys2////////////////////////////////////
 return_code = I2CWrite2(0xF0, reg1 ,reg2, 2);

   if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);

 //invio comando inizio Impedance Track
 yData[0] = 0x21;
 yData[1] = 0x00;
 return_code = I2CWrite(GASGAUGE_ADDRESS, 0x00, yData, 2);
    if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);

for(i=0; i<=20000; i++);
 //////////////////////////////////////////////////////////////


 ////////////////////spengo le board...al contrario////////////////
   reg1 = 0x20;
   reg2 = 0x03;
//////////////////////////////////////////////

 return_code = I2CWrite2(0xF0, reg1 ,reg2, 2);

   if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);


for(i=0; i<=20000; i++);


 return_code = I2CWrite2(0xCE, reg1 ,reg2, 2);

   if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);

for(i=0; i<=20000; i++);


 return_code = I2CWrite2(0xD4, reg1 ,reg2, 2);

   if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);

for(i=0; i<=20000; i++);

 return_code = I2CWrite2(0xEC, reg1 ,reg2, 2);

   if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);

for(i=0; i<=20000; i++);

 return_code = I2CWrite2(0xD6, reg1 ,reg2, 2);

   if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);

for(i=0; i<=20000; i++);

 return_code = I2CWrite2(0xE8, reg1 ,reg2, 2);

   if(return_code)
            printf("Return %d\n",return_code);
 while((long)(MS_TIMER - t) < WRITE_TIME);

for(i=0; i<=20000; i++);
```

```
//////////////////////////////////////////////////////fine
inizializzazione//////////////////////////////////////////


   sock_init();
   http_init();
   tcp_reserveport(80);


   while (1) {
     i = 0;


//////////////////////////////////////////////////////Battery1-
System1//////////////////////////////////////////////////////
   reg1 = 0x80;
   reg2 = 0x03;
// indirizzo 20
 return_code = I2CWrite2(0xE8, reg1 ,reg2, 2);
               while((long)(MS_TIMER - t) < WRITE_TIME);

   //Read current voltage
   yData[0]=0;
   yData[1]=0;
   return_code = I2CRead(GASGAUGE_ADDRESS, 0x8, yData, 2);
   if(return_code){
               printf("Return %d\n",return_code);
   }else {
   Voltage11 = 256 * yData[1] + yData[0] ;
   }
   var = sspec_addvariable("V Battery1-System1", &Voltage11, INT16,
"%d", SERVER_HTTP);
               var = sspec_addfv(form, var);
               sspec_setfvname(form, var, "V Battery1-System1");
               sspec_setfvdesc(form, var, "mV");
               sspec_setfvlen(form, var, 5);
   sspec_setfvreadonly(form, var, 1);  //READ ONLY

   //Read average current
   yData[0]=0;
   yData[1]=0;
   return_code = I2CRead(GASGAUGE_ADDRESS, 0xA, yData, 2);
   if(return_code){
               printf("Return %d\n",return_code);
   }else {
   Current11 = 256 * yData[1] + yData[0] ;
   }
   var = sspec_addvariable("A Battery1-System1", &Current11, INT16,
"%d", SERVER_HTTP);
               var = sspec_addfv(form, var);
               sspec_setfvname(form, var, "A Battery1-System1");
               sspec_setfvdesc(form, var, "A * 10^-2");
               sspec_setfvlen(form, var, 5);
   sspec_setfvreadonly(form, var, 1);  //READ ONLY

   //Read state of charge %
   yData[0]=0;
   yData[1]=0;
   return_code = I2CRead(GASGAUGE_ADDRESS, 0x2, yData, 2);
   if(return_code){
               printf("Return %d\n",return_code);
   }else {
   StateofCharge11 = 256 * yData[1] + yData[0] ;
   }
               // Add the first variable, and set it up with the form
               var = sspec_addvariable("SOC Battery1-System1",
&StateofCharge11, INT16, "%d", SERVER_HTTP);
               var = sspec_addfv(form, var);
               sspec_setfvname(form, var, "SOC Battery1-System1");
               sspec_setfvdesc(form, var, "%");
               sspec_setfvlen(form, var, 5);
               sspec_setfvreadonly(form, var, 1);  //READ ONLY


   //Temperature °C
   yData[0]=0;
   yData[1]=0;
   return_code = I2CRead(GASGAUGE_ADDRESS, 0xc, yData, 2);
   if(return_code){
               printf("Return %d\n",return_code);
   }else {
   Temperature11 = 256 * yData[1] + yData[0] ;
   Temperature11 = (Temperature11-2730.0)/10.0;
   }
               // Add the first variable, and set it up with the form
               var = sspec_addvariable("T Battery1-System1",
&Temperature11, FLOAT32, "%.2f", SERVER_HTTP);
               var = sspec_addfv(form, var);
               sspec_setfvname(form, var, "T Battery1-System1");
               sspec_setfvdesc(form, var, "°C");
               sspec_setfvlen(form, var, 5);
               sspec_setfvreadonly(form, var, 1);  //READ ONLY


   reg1 = 0x20;
   reg2 = 0x03;
return_code = I2CWrite2(0xE8, reg1 ,reg2, 2);
 while((long)(MS_TIMER - t) < WRITE_TIME);
//////////////////////////////////////////////////////END BATT
1//////////////////////////////////////////////////////


//////////////////////////////////////////////Battery2-
System1////////////////////////////////////////////////////////////////////
   reg1 = 0x80;
   reg2 = 0x03;
// indirizzo 20
 return_code = I2CWrite2(0xD6, reg1 ,reg2, 2);
               while((long)(MS_TIMER - t) < WRITE_TIME);

   //Read current voltage
   yData[0]=0;
   yData[1]=0;
   return_code = I2CRead(GASGAUGE_ADDRESS, 0x8, yData, 2);
   if(return_code){
               printf("Return %d\n",return_code);
   }else {
   Voltage21 = 256 * yData[1] + yData[0] ;
   }
   var = sspec_addvariable("V Battery2-System1", &Voltage21, INT16,
"%d", SERVER_HTTP);
               var = sspec_addfv(form, var);
               sspec_setfvname(form, var, "V Battery2-System1");
               sspec_setfvdesc(form, var, "mV");
               sspec_setfvlen(form, var, 5);
   sspec_setfvreadonly(form, var, 1);  //READ ONLY

   //Read average current
   yData[0]=0;
   yData[1]=0;
   return_code = I2CRead(GASGAUGE_ADDRESS, 0xA, yData, 2);
   if(return_code){
               printf("Return %d\n",return_code);
   }else {
   Current21 = 256 * yData[1] + yData[0] ;
   }
   var = sspec_addvariable("A Battery2-System1", &Current21, INT16,
"%d", SERVER_HTTP);
               var = sspec_addfv(form, var);
               sspec_setfvname(form, var, "A Battery2-System1");
               sspec_setfvdesc(form, var, "A * 10^-2");
               sspec_setfvlen(form, var, 5);
   sspec_setfvreadonly(form, var, 1);  //READ ONLY

   //Read state of charge %
   yData[0]=0;
   yData[1]=0;
   return_code = I2CRead(GASGAUGE_ADDRESS, 0x2, yData, 2);
   if(return_code){
               printf("Return %d\n",return_code);
   }else {
   StateofCharge21 = 256 * yData[1] + yData[0] ;
   }
               // Add the first variable, and set it up with the form
               var = sspec_addvariable("SOC Battery2-System1",
&StateofCharge21, INT16, "%d", SERVER_HTTP);
               var = sspec_addfv(form, var);
               sspec_setfvname(form, var, "SOC Battery2-System1");
               sspec_setfvdesc(form, var, "%");
               sspec_setfvlen(form, var, 5);
```

```
            sspec_setfvreadonly(form, var, 1);  //READ ONLY


  //Temperature °C
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0xc, yData, 2);
  if(return_code){
            printf("Return %d\n",return_code);
  }else {
  Temperature21 = 256 * yData[1] + yData[0] ;
  Temperature21 = (Temperature21-2730.0)/10.0;
  }
            // Add the first variable, and set it up with the form
            var = sspec_addvariable("T Battery2-System1",
&Temperature21, FLOAT32, "%.2f", SERVER_HTTP);
            var = sspec_addfv(form, var);
            sspec_setfvname(form, var, "T Battery2-System1");
            sspec_setfvdesc(form, var, "°C");
            sspec_setfvlen(form, var, 5);
            sspec_setfvreadonly(form, var, 1);  //READ ONLY

  reg1 = 0x20;
  reg2 = 0x03;
return_code = I2CWrite2(0xD6, reg1 ,reg2, 2);
 while((long)(MS_TIMER - t) < WRITE_TIME);
//////////////////////////////////////////END BATT
2//////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////Battery3-
System1//////////////////////////////////////////////////////////
  reg1 = 0x80;
  reg2 = 0x03;
// indirizzo 20
 return_code = I2CWrite2(0xEC, reg1 ,reg2, 2);
            while((long)(MS_TIMER - t) < WRITE_TIME);

  //Read current voltage
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0x8, yData, 2);
  if(return_code){
            printf("Return %d\n",return_code);
  }else {
  Voltage31 = 256 * yData[1] + yData[0] ;
  }
  var = sspec_addvariable("V Battery3-System1", &Voltage31, INT16,
"%d", SERVER_HTTP);
            var = sspec_addfv(form, var);
            sspec_setfvname(form, var, "V Battery3-System1");
            sspec_setfvdesc(form, var, "mV");
            sspec_setfvlen(form, var, 5);
  sspec_setfvreadonly(form, var, 1);  //READ ONLY

  //Read average current
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0xA, yData, 2);
  if(return_code){
            printf("Return %d\n",return_code);
  }else {
  Current31 = 256 * yData[1] + yData[0] ;
  }
  var = sspec_addvariable("A Battery3-System1", &Current31, INT16,
"%d", SERVER_HTTP);
            var = sspec_addfv(form, var);
            sspec_setfvname(form, var, "A Battery3-System1");
            sspec_setfvdesc(form, var, "A * 10^-2");
            sspec_setfvlen(form, var, 5);
  sspec_setfvreadonly(form, var, 1);  //READ ONLY

  //Read state of charge %
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0x2, yData, 2);
  if(return_code){
            printf("Return %d\n",return_code);
  }else {
  StateofCharge31 = 256 * yData[1] + yData[0] ;
  }
            // Add the first variable, and set it up with the form
            var = sspec_addvariable("SOC Battery3-System1",
&StateofCharge31, INT16, "%d", SERVER_HTTP);
            var = sspec_addfv(form, var);
            sspec_setfvname(form, var, "SOC Battery3-System1");
            sspec_setfvdesc(form, var, "%");
            sspec_setfvlen(form, var, 5);
            sspec_setfvreadonly(form, var, 1);  //READ ONLY


  //Temperature °C
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0xc, yData, 2);
  if(return_code){
            printf("Return %d\n",return_code);
  }else {
  Temperature31 = 256 * yData[1] + yData[0] ;
  Temperature31 = (Temperature31-2730.0)/10.0;
  }
            // Add the first variable, and set it up with the form
            var = sspec_addvariable("T Battery3-System1",
&Temperature31, FLOAT32, "%.2f", SERVER_HTTP);
            var = sspec_addfv(form, var);
            sspec_setfvname(form, var, "T Battery3-System1");
            sspec_setfvdesc(form, var, "°C");
            sspec_setfvlen(form, var, 5);
            sspec_setfvreadonly(form, var, 1);  //READ ONLY

  reg1 = 0x20;
  reg2 = 0x03;
return_code = I2CWrite2(0xEC, reg1 ,reg2, 2);
 while((long)(MS_TIMER - t) < WRITE_TIME);
//////////////////////////////////////////////////////////END BATT
3//////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////Battery1-
System2//////////////////////////////////////////////////////////
  reg1 = 0x80;
  reg2 = 0x03;
// indirizzo 20
 return_code = I2CWrite2(0xD4, reg1 ,reg2, 2);
            while((long)(MS_TIMER - t) < WRITE_TIME);

  //Read current voltage
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0x8, yData, 2);
  if(return_code){
            printf("Return %d\n",return_code);
  }else {
  Voltage12 = 256 * yData[1] + yData[0] ;
  }
  var = sspec_addvariable("V Battery1-System2", &Voltage12, INT16,
"%d", SERVER_HTTP);
            var = sspec_addfv(form, var);
            sspec_setfvname(form, var, "V Battery1-System2");
            sspec_setfvdesc(form, var, "mV");
            sspec_setfvlen(form, var, 5);
  sspec_setfvreadonly(form, var, 1);  //READ ONLY

  //Read average current
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0xA, yData, 2);
  if(return_code){
            printf("Return %d\n",return_code);
  }else {
  Current12 = 256 * yData[1] + yData[0] ;
  }
  var = sspec_addvariable("A Battery1-System2", &Current12, INT16,
"%d", SERVER_HTTP);
            var = sspec_addfv(form, var);
            sspec_setfvname(form, var, "A Battery1-System2");
            sspec_setfvdesc(form, var, "A * 10^-2");
```

```
                sspec_setfvlen(form, var, 5);
  sspec_setfvreadonly(form, var, 1);  //READ ONLY

  //Read state of charge %
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0x2, yData, 2);
  if(return_code){
                printf("Return %d\n",return_code);
  }else {
  StateofCharge12 = 256 * yData[1] + yData[0] ;
  }
                // Add the first variable, and set it up with the form
                var = sspec_addvariable("SOC Battery1-System2",
&StateofCharge12, INT16, "%d", SERVER_HTTP);
                var = sspec_addfv(form, var);
                sspec_setfvname(form, var, "SOC Battery1-System2");
                sspec_setfvdesc(form, var, "%");
                sspec_setfvlen(form, var, 5);
                sspec_setfvreadonly(form, var, 1);  //READ ONLY


  //Temperature °C
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0xc, yData, 2);
  if(return_code){
                printf("Return %d\n",return_code);
  }else {
  Temperature12 = 256 * yData[1] + yData[0] ;
  Temperature12 = (Temperature12-2730.0)/10.0;
  }
                // Add the first variable, and set it up with the form
                var = sspec_addvariable("T Battery1-System2",
&Temperature12, FLOAT32, "%.2f", SERVER_HTTP);
                var = sspec_addfv(form, var);
                sspec_setfvname(form, var, "T Battery1-System2");
                sspec_setfvdesc(form, var, "°C");
                sspec_setfvlen(form, var, 5);
                sspec_setfvreadonly(form, var, 1);  //READ ONLY

  reg1 = 0x20;
  reg2 = 0x03;
return_code = I2CWrite2(0xD4, reg1 ,reg2, 2);
 while((long)(MS_TIMER - t) < WRITE_TIME);
////////////////////////////////////////////////////////////END BATT 1 - STSYEM
2////////////////////////////////////////////////////////


//////////////////////////////////////////////Battery2-
System2////////////////////////////////////////////////////////////////////////
  reg1 = 0x80;
  reg2 = 0x03;
// indirizzo 20
 return_code = I2CWrite2(0xCE, reg1 ,reg2, 2);
                while((long)(MS_TIMER - t) < WRITE_TIME);

  //Read current voltage
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0x8, yData, 2);
  if(return_code){
                printf("Return %d\n",return_code);
  }else {
  Voltage22 = 256 * yData[1] + yData[0] ;
  }
  var = sspec_addvariable("V Battery2-System2", &Voltage22, INT16,
"%d", SERVER_HTTP);
                var = sspec_addfv(form, var);
                sspec_setfvname(form, var, "V Battery2-System2");
                sspec_setfvdesc(form, var, "mV");
                sspec_setfvlen(form, var, 5);
  sspec_setfvreadonly(form, var, 1);  //READ ONLY

  //Read average current
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0xA, yData, 2);
  if(return_code){
                printf("Return %d\n",return_code);
  }else {
  Current22 = 256 * yData[1] + yData[0] ;
  }
  var = sspec_addvariable("A Battery2-System2", &Current22, INT16,
"%d", SERVER_HTTP);
                var = sspec_addfv(form, var);
                sspec_setfvname(form, var, "A Battery2-System2");
                sspec_setfvdesc(form, var, "A * 10^-2");
                sspec_setfvlen(form, var, 5);
  sspec_setfvreadonly(form, var, 1);  //READ ONLY

  //Read state of charge %
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0x2, yData, 2);
  if(return_code){
                printf("Return %d\n",return_code);
  }else {
  StateofCharge22 = 256 * yData[1] + yData[0] ;
  }
                // Add the first variable, and set it up with the form
                var = sspec_addvariable("SOC Battery2-System2",
&StateofCharge22, INT16, "%d", SERVER_HTTP);
                var = sspec_addfv(form, var);
                sspec_setfvname(form, var, "SOC Battery2-System2");
                sspec_setfvdesc(form, var, "%");
                sspec_setfvlen(form, var, 5);
                sspec_setfvreadonly(form, var, 1);  //READ ONLY


  //Temperature °C
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0xc, yData, 2);
  if(return_code){
                printf("Return %d\n",return_code);
  }else {
  Temperature22 = 256 * yData[1] + yData[0] ;
  Temperature22 = (Temperature22-2730.0)/10.0;
  }
                // Add the first variable, and set it up with the form
                var = sspec_addvariable("T Battery2-System2",
&Temperature22, FLOAT32, "%.2f", SERVER_HTTP);
                var = sspec_addfv(form, var);
                sspec_setfvname(form, var, "T Battery2-System2");
                sspec_setfvdesc(form, var, "°C");
                sspec_setfvlen(form, var, 5);
                sspec_setfvreadonly(form, var, 1);  //READ ONLY

  reg1 = 0x20;
  reg2 = 0x03;
return_code = I2CWrite2(0xCE, reg1 ,reg2, 2);
 while((long)(MS_TIMER - t) < WRITE_TIME);
////////////////////////////////////////////////END BATT 2 -
SYSTEM2////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////Battery3-
System2////////////////////////////////////////////////////////
  reg1 = 0x80;
  reg2 = 0x03;
return_code = I2CWrite2(0xF0, reg1 ,reg2, 2);
                while((long)(MS_TIMER - t) < WRITE_TIME);

  //Read current voltage
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0x8, yData, 2);
  if(return_code){
                printf("Return %d\n",return_code);
  }else {
  Voltage32 = 256 * yData[1] + yData[0] ;
  }
  var = sspec_addvariable("V Battery3-System2", &Voltage32, INT16,
"%d", SERVER_HTTP);
                var = sspec_addfv(form, var);
                sspec_setfvname(form, var, "V Battery3-System2");
                sspec_setfvdesc(form, var, "mV");
                sspec_setfvlen(form, var, 5);
  sspec_setfvreadonly(form, var, 1);  //READ ONLY
```

```
  //Read average current
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0xA, yData, 2);
  if(return_code){
            printf("Return %d\n",return_code);
  }else {
  Current32 = 256 * yData[1] + yData[0] ;
  }
  var = sspec_addvariable("A Battery3-System2", &Current32, INT16,
"%d", SERVER_HTTP);
            var = sspec_addfv(form, var);
            sspec_setfvname(form, var, "A Battery3-System2");
            sspec_setfvdesc(form, var, "A * 10^-2");
            sspec_setfvlen(form, var, 5);
  sspec_setfvreadonly(form, var, 1);  //READ ONLY

  //Read state of charge %
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0x2, yData, 2);
  if(return_code){
            printf("Return %d\n",return_code);
  }else {
  StateofCharge32 = 256 * yData[1] + yData[0] ;
  }
            // Add the first variable, and set it up with the form
            var = sspec_addvariable("SOC Battery3-System2",
&StateofCharge32, INT16, "%d", SERVER_HTTP);
            var = sspec_addfv(form, var);
            sspec_setfvname(form, var, "SOC Battery3-System2");
            sspec_setfvdesc(form, var, "%");
            sspec_setfvlen(form, var, 5);
            sspec_setfvreadonly(form, var, 1);  //READ ONLY


  //Temperature °C
  yData[0]=0;
  yData[1]=0;
  return_code = I2CRead(GASGAUGE_ADDRESS, 0xc, yData, 2);
  if(return_code){
            printf("Return %d\n",return_code);
  }else {
  Temperature32 = 256 * yData[1] + yData[0] ;
  Temperature32 = (Temperature32-2730.0)/10.0;
  }
            // Add the first variable, and set it up with the form
            var = sspec_addvariable("T Battery3-System2",
&Temperature32, FLOAT32, "%.2f", SERVER_HTTP);
            var = sspec_addfv(form, var);
            sspec_setfvname(form, var, "T Battery3-System2");
            sspec_setfvdesc(form, var, "°C");
            sspec_setfvlen(form, var, 5);
            sspec_setfvreadonly(form, var, 1);  //READ ONLY

  reg1 = 0x20;
  reg2 = 0x03;
return_code = I2CWrite2(0xF0, reg1 ,reg2, 2);
 while((long)(MS_TIMER - t) < WRITE_TIME);
//////////////////////////////////////////////////////////END BATT 3 - SYSTEM
2//////////////////////////////////////////////////////////




            sspec_aliasspec(form, "index.html");
            sspec_aliasspec(form, "/");



  while( i < 20000){
    http_handler();
  i=i+1;
  }

  }
}
```

ISSN 1590-2595

Quaderni di Geofisica

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

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

ISSN 2039-7941

rapporti tecnici INGV

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

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

ISSN 2039-6651

Miscellanea INGV

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

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

Coordinamento editoriale e impaginazione
Centro Editoriale Nazionale | INGV

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

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

http://www.ingv.it

Istituto Nazionale di Geofisica e Vulcanologia