INTERNAL REPORT

Una libreria di classi per il controllo dei sensori lineari

Raimondo Concu, Franco Buffa,
Tonino Pisanu, Claudio Pernechele

Report N. 18, released: 23/07/2012

Reviewer: Sergio Poppi

OAC Osservatorio Astronomico di Cagliari

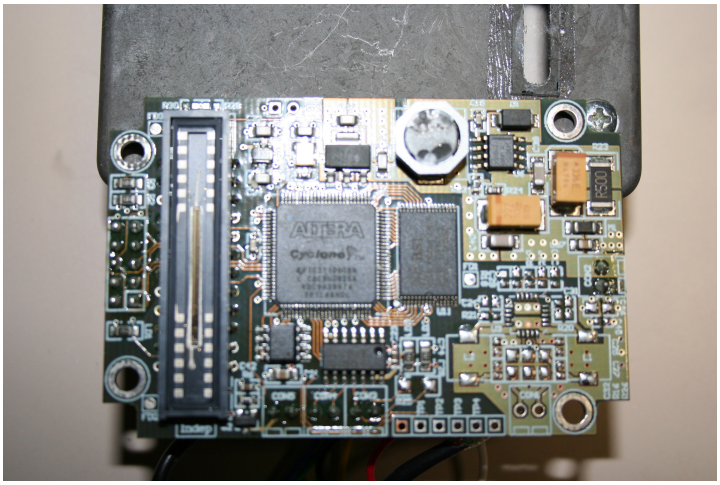
Indice generale

Introduzione

Il presente rapporto interno, descrive il software che è stato progettato e scritto per il controllo e l'acquisizione dei dati dei sensori lineari, sviluppati dal GAI02 Metrologia, per il controllo dello specchio primario dell'SRT. Il sistema di misura dei sensori lineari, nasce in seguito alla necessità di misurare le deformazioni dello specchio primario di un radiotelescopio con superficie attiva. La possibilità infatti di modificare la posizione dei pannelli, in quanto montati su attuatori lineari, permette di posizionare la superficie per avere sempre la forma migliore anche quando su di essa agiscono le deformazioni gravitazionali o i gradienti termici. Queste misure e correzioni sono particolarmente importanti ad alta frequenza, come ad esempio a 100 GHz a cui corrisponde una lughezza d'onda di 3 mm e quindi è necessario che la misura della superficie abbia una accuratezza di circa cento micron. Una volta che il riflettore primario sia stato allineato staticamente in modo "assoluto" è sufficiente operare una misura "relativa" delle deformazioni rispetto a questo "punto di zero" per mantenere la superficie nella sua forma ottimale in maniera dinamica. I sensori lineari vengono utilizzati per la misura delle deformazioni di un riflettore di grandi dimensioni (maggiore di 30 metri in diametro) a livelli di accuratezza di qualche centinaio di μm e con una frequenza di lettura di qualche Hz. I sensori utilizzano un CMOS lineare (modello S10077 della Hamamatsu), formato 1024 pixels delle dimensioni di 14 μm x 50 μm, per una lunghezza totale di 14.3 mm. I pixels hanno una profondità di 10 bit (1024 livello di grigio). La scheda di controllo viene alimentata da una tensione di 13Volts ed ha on board una FPGA, in grado di eseguire l'impostazione e la lettura del sensore. La scheda contiene inoltre un controllo di potenza per un diodo laser ed un sensore di temperatura. Una immagine di un elemento del sistema e' mostrata in figura 1, si noti che il diodo laser viene accoppiato ad ogni scatola del sensore. In figura 2 e' mostrata la scheda elettronica in cui si notano la FPGA e il sensore CMOS.

Figura 1: Il sensore lineare.

Figura 2: La scheda di controllo (FPGA) ed il sensore CMOS.



La comunicazione tra il sensore (slave) ed il computer host (master) avviene tramite RS485. I sensori vengono collegati in cascata, tramite una linea cablata composta dalla linea RS485 e dall'alimentazione (tramite un cavo a 6 poli, contenente la linea di comunicazione RS485 e l'alimentazione). Ogni sensore viene univocamente identificato da un numero che và da 1 a 255, possono essere indirizzati al massimo 255 sensori. L'i-esima scheda sensore, ha due connettori, uno per il cavo proveniente dal sensore i-1, uno per il cavo che và al sensore i+1 assieme all'alimentazione del diodo laser. L'ultimo sensore di una linea ha una resistenza di terminazione a 120 Ohm. La parte finale di ogni linea giunge ad un HUB, e da lì vengono fatti i collegamenti al PC di controllo, tramite un convertitore USB-RS485.

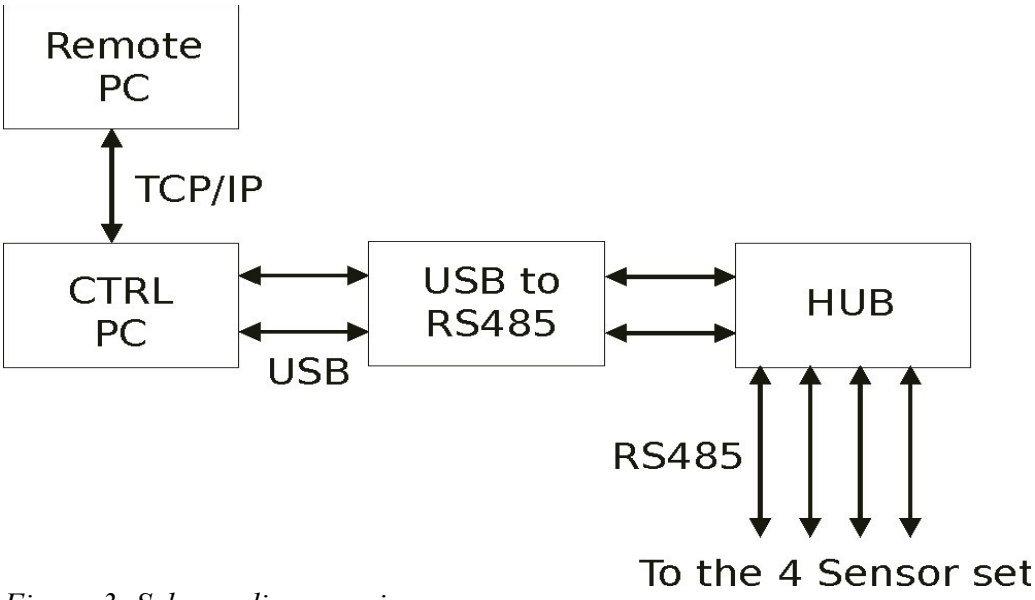


Figura 3: Schema di connessione.

Figura 4: Un sistema costituito da 24 sensori lineari.

Figura 5: Un HUB utilizzato per il collegamento dei senssori lineari al PC master.

L'applicazione LinearSensorTest

L'applicazione LinearSensorTest, è scritta in C#.NET. Essa utilizza la libreria LinearSensorLib per la comunicazione con i sensori lineari .

Il Bussiness Object Model

L'applicazione è composta da un form principale detto MainForm e da un form di opzioni detto FormOptions.

Figura 6: Diagramma di classe dell'applicazione LinearSensorTest.

La Human-Machine Interface

L'HMI dell'applicazione è composta da un menu principale, da una listview per mostrare l'elenco dei sensori rilevati e di un pannello dei comandi da inviare al sensore selezionato nella listview.

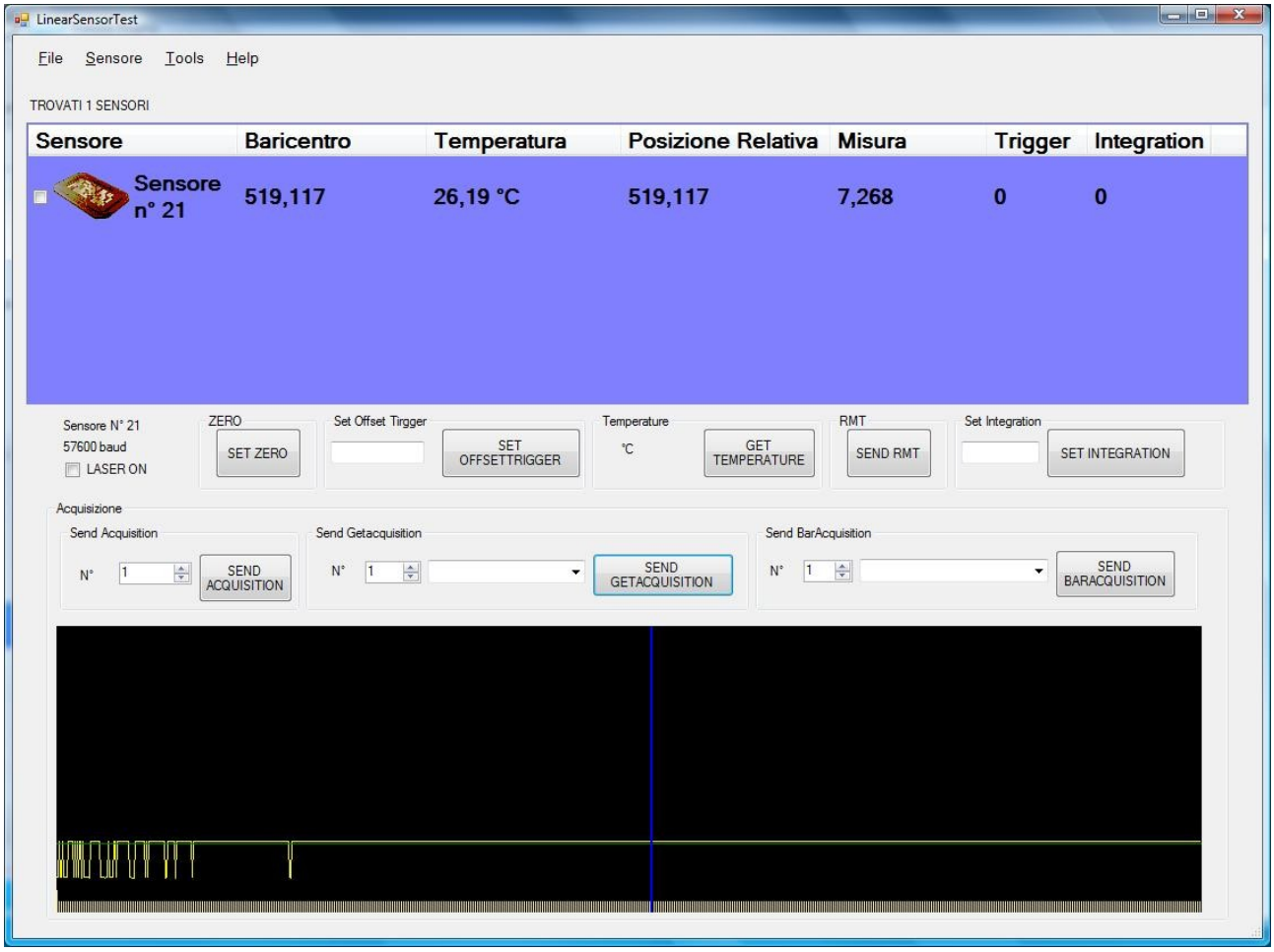


Figura 7: La GUI dell'applicazione LinearSensorTest.

Di seguito la descrizione di alcune funzionalità fornite dall'interfaccia.

Ricerca Sensori

Dal menu ***File->Ricerca Sensori*** è possibile avviare la ricerca dei sensori collegati al PC master. La libreria LinearSensorLib, istanziando l'oggetto Reader, avvia un giro di ACKNOLEDGE, alla ricerca di 24 sensori per ogni porta COM. Ossia per ciascuna porta COM viene eseguito un ciclo su i da 1 a 24 per ricercare i sensori inviando un messaggio di ACK, in attesa di ricevere un messaggio di BACKACKNOLEDGE, se il sensore i risulta presente.

La Lista Sensori

Al centro dell'interfaccia grafica è presente quindi una listView con l'elenco di tutti i sensori trovati.

Difatti dopo aver istanziato il Reader, è possibile recuperare, tramite la funzione Reader.GetSensorList(), l'elenco dei sensori trovati. Con questo elenco è quindi possibile inserire un listViewItem per ogni sensore trovato e creare quindi la lista dei sensori, dove ogni elemento della lista è associato univocamente al sensore lineare presente nel sistema. In questo elenco è possibile leggere le seguenti informazioni:

- **Sensore**: il numero del Sensore.
- **Baricentro**: il valore del baricentro in pixels.
- **Temperatura**: il valore dell'ultima temperatura rilevata.
- **Posizione Relativa**: la differenza, in pixels, tra il baricentro rilevato e quello scelto come zero alla pressione sul pulsante SET ZERO.
- **Misura**: il valore precedente espresso in millimetri, ossia il valore precedente moltiplicato per 0,014.
- **Trigger**: il trigger, ossia il livello di soglia impostato.
- **Integration**: il tempo di integrazione impostato.

I comandi

Selezionando un sensore presente nella lista sensori appare un pannello con i pulsanti che consentono l'invio dei comandi al sensore selezionato. Possiamo notare il comando di SETOFFSETTRIGGER, il comando di SETINTEGRATION, il comando di LASERON, e tutti i comandi utili per richiedere al sensore fino a 128 acquisizioni in sequenza (ACQUISITION) e le relative immagini (GETAQUISITION) e baricentri (GETBARACQUISITION).

I comandi in broadcast, come il LASERON ed il passaggio alla velocità di 115200 baud per tutti i sensori, possono essere inviati accedendo dal menu ***Sensore->All Laser ON*** e ***Sensore->Passa a 115200*** rispettivamente. E' possibile settare il baricentro di zero per tutti i sensori selezionando ***Sensore->Set Zero***.

Il grafico

In basso sotto il pannello comandi è presente un grafico nel quale è possibile vedere l'iesima immagine o baricentro acquisite dal sensore. Viene anche mostrato il trigger o soglia otteniti dall'immagine.

Lettura continua

Dopo aver selezionato un sensore, dal menu ***Sensore->Lettura Continua*** è possibile avviare la lettura in continuo dei sensori. Affianco a ciascuna sensore è presente una checkbox, che se selezionata, attiva la lettura continua su quel sensore. La lettura continua avviene in un thread separato, che utilizza le funzioni Sensore.LeggiImmagine() e Sensore.LeggiBaricentro(). In alto a destra, sopra la lista sensori è possibile vedere il tempo di round trip, ossia il tempo impiagato dal thread a leggere le immagini o i baricentri dei sensori selezionati. Il thread di lettura continua, quando avviato, crea un file di log denominato yyyyMMdd-hhmmsstmp.log. Dove yyyy è l'anno, MM è il mese, dd è il giorno, hh è l'ora, mm sono i minuti, ss i secondi dell'orologio di sistema. All'interno di questo file, in formato csv, son memorizzati, nell'ordine, i seguenti dati: data\ora di sistema, numero sensore, il baricentro scelto come zero in pixels, il baricentro rilevato in pixels, la temperatura rilevata.

Il Menu Opzioni

Dal menu ***TOOLS->Options*** è possibile accedere ad un form di opzioni. In questo form è possibile scegliere se, durante la lettura continua, richiedere al sensore tutta l'immagine o solo il baricentro.E' inoltre possibile settare l'intervallo di lettura continua, ossia il tempo di aggiornamento della GUI per la visualizzazione dei dati. Si noti che, invece, il thread di lettura continua viene eseguito alla sua massima velocità, che dipende dal tipo di dati richiesti al sensore.

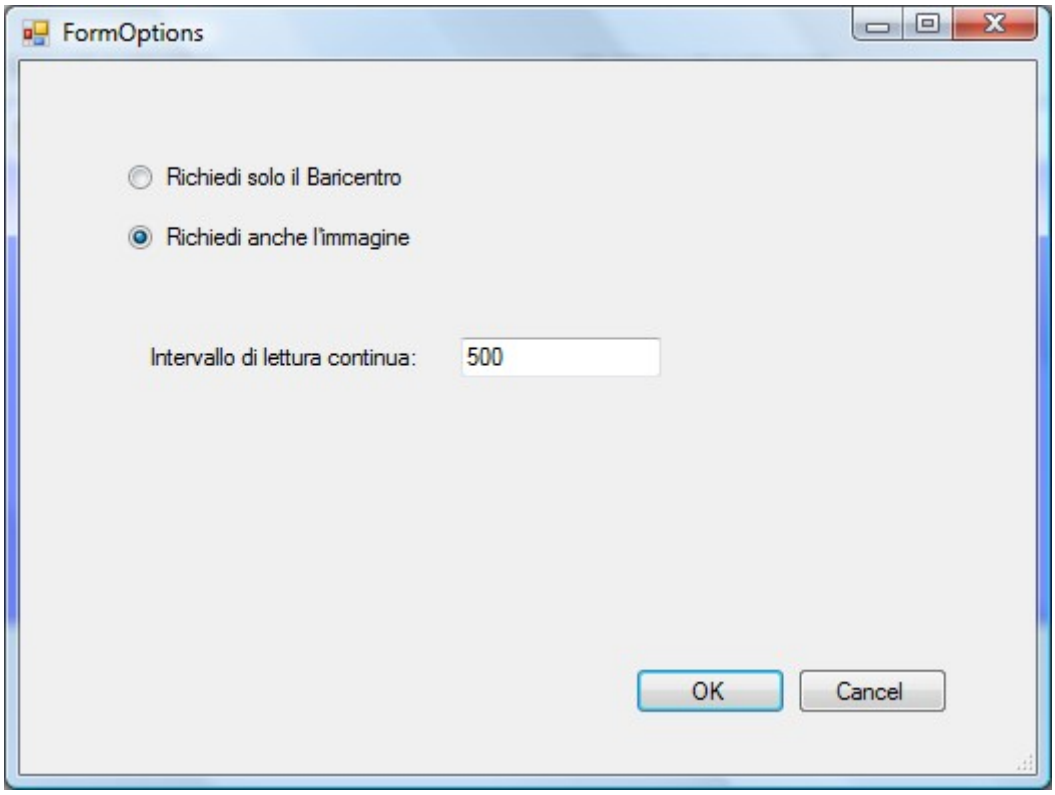


Figura 8: La finestra di dialogo delle Opzioni.

Salvataggio dati

Dal menu ***File->Save*** è possibile salvare in un file di testo, separato da ';', nell'ordine, le seguenti informazioni: baricentro calcolato, temperatura, trigger calcolato, valore dei pixels da 1 a 1024. Queste informazioni vengono ripetute tante volte quante sono state le acquisizioni richieste.

Dal menu ***File->Save Image*** è invece possibile salvare in formato bmp, il grafico dell'ultima immagine rilevata.

Figura 9: L'applicazione utilizzata per il monitoraggio di una trave metallica.

La libreria LinearSensorLib

La libreria LinearSensorLib, è scritta in C#.NET. Essa contiene la logica di comunicazione tra il computer master ed i sensori slave. La libreria implementa in un'architettura ad oggetti tutti quei comandi che sono stati esposti nel capitolo precedente e che consentono l'interazione con i sensori lineari.

Il Business Object Model

La libreria è composta da tre classi: la classe Reader, la classe PortaCOM e la classe Sensore.

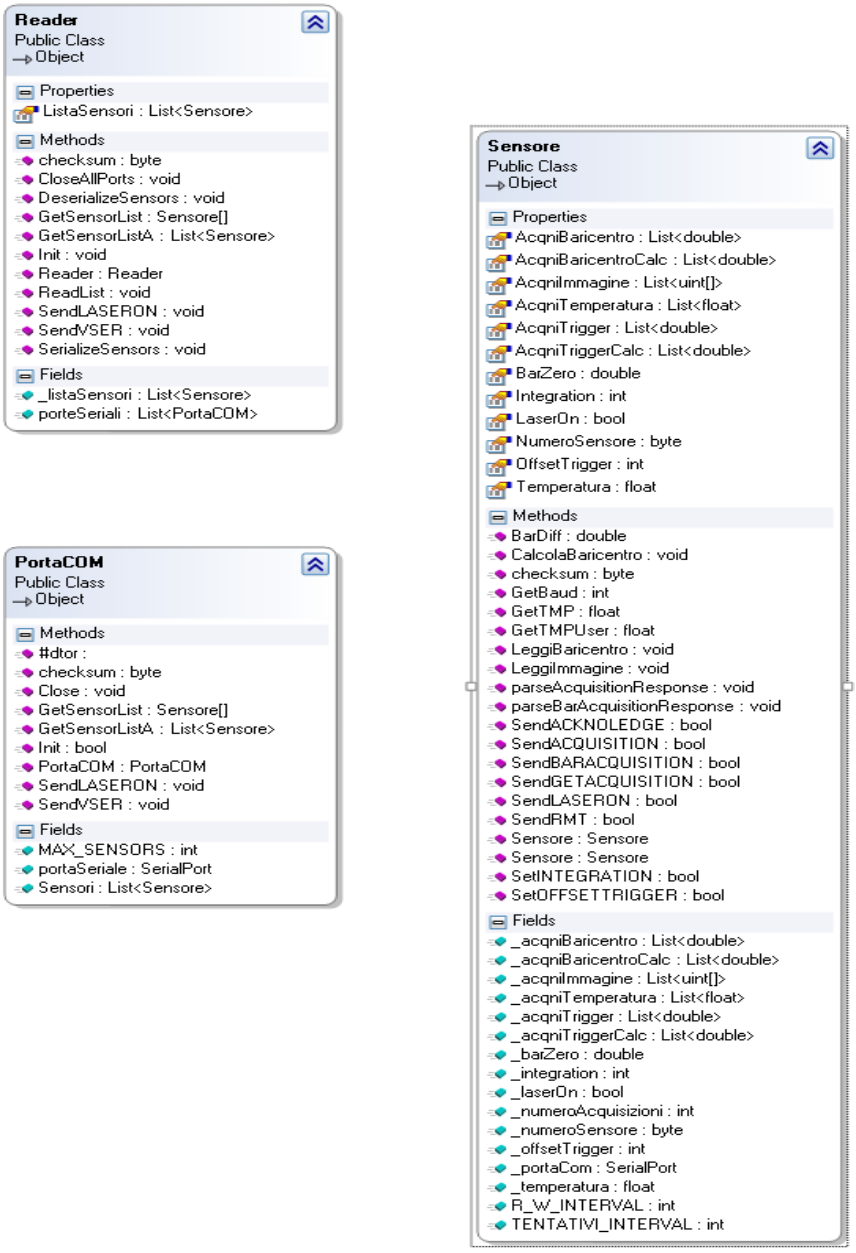


Figura 10: Diagramma delle classi della libreria LinearSensorLib.

La classe Reader possiede una lista di oggetti di tipo PortaCOM. Ogni oggetto PortaCOM rappresenta una delle porte COM installate nel sistema ed attraverso le quali è possibile comunicare con i sensori collegati al computer. La classe PortaCOM possiede una lista di oggetti di tipo Sensore, ossia la lista dei sensori collegati fisicamente a quella porta COM. La classe Sensore, possiede quasi tutta la logica di comunicazione con il sensore cui fa riferimento. Difatti ogni istanza di tale classe è un oggetto che rappresenta fisicamente il singolo sensore collegato al sistema.

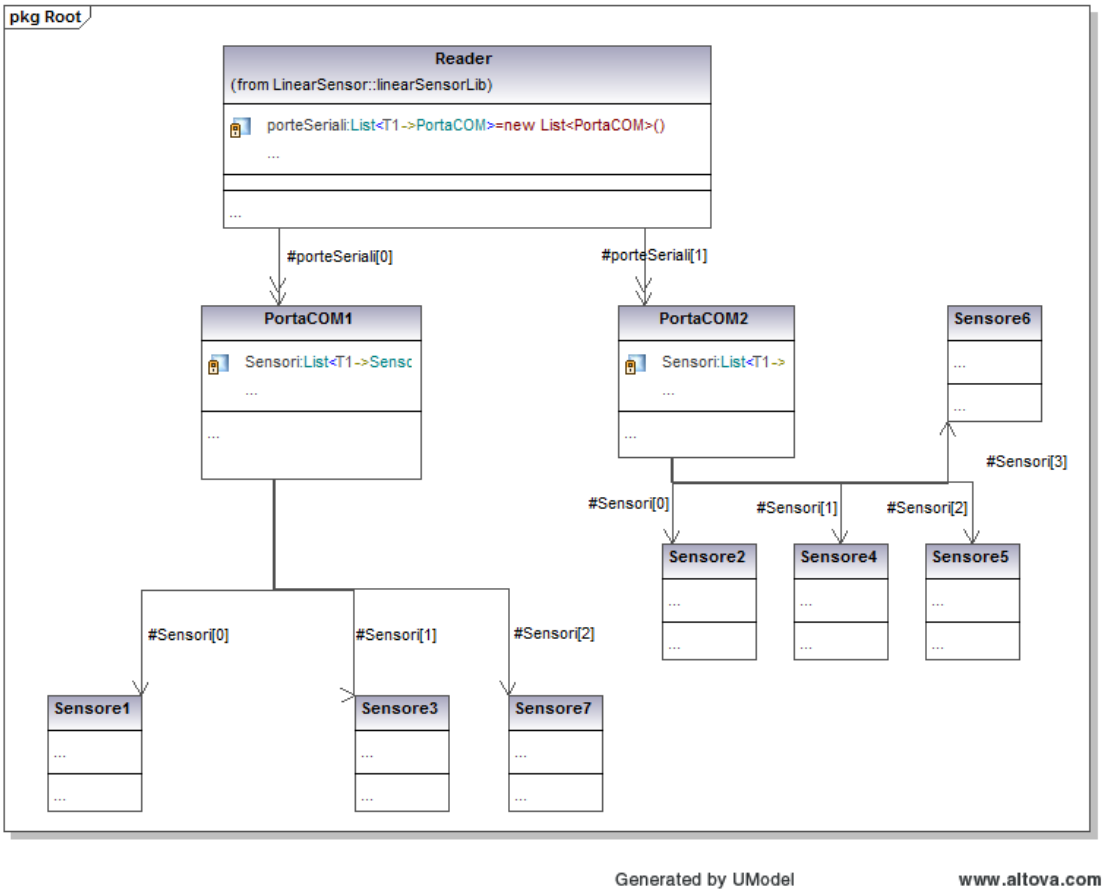


Figura 11: Una possibile configurazione, quando il sistema è composto da 7 sensori collegati a 2 porte COM.

La classe Reader

La classe Reader è il contenitore che possiede la lista delle porte COM installate nel sistema, attraverso cui accede ai sensori ad esse collegati. Il costruttore, attraverso il metodo Init(), consente l'inizializzazione della rete di sensori. Tramite il metodo GetSensorList() e GetSensorListA() è possibile ottenere la lista di tutti i sensori collegati alla rete. La lista privata porteSeriali contiene l'elenco delle porte seriali cui è collegato almeno un sensore lineare.

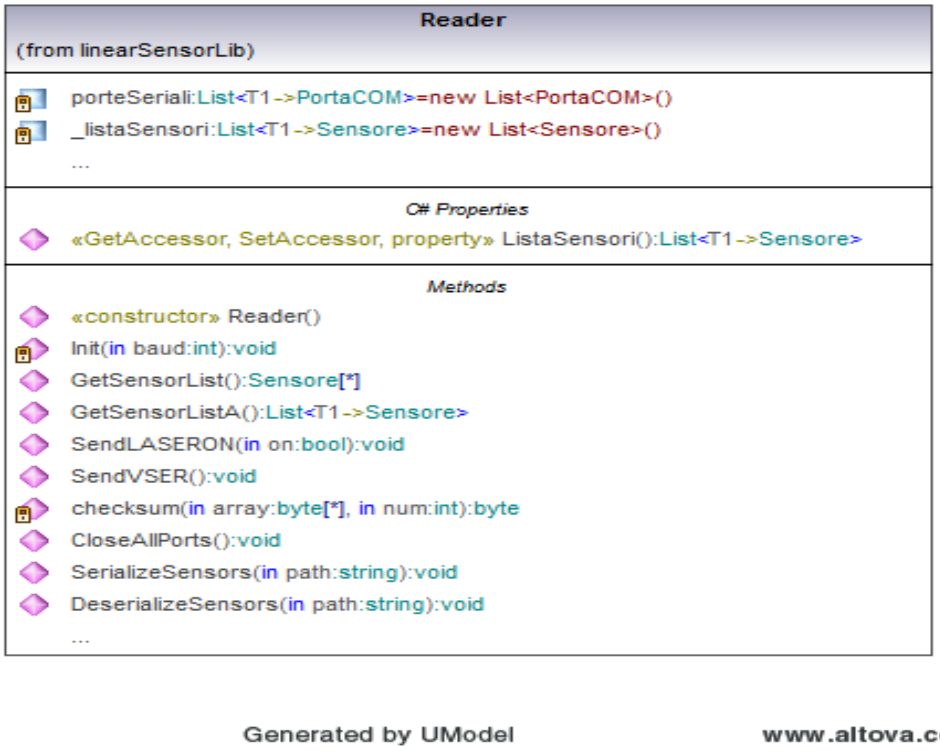


Figura 12: Rappresentazione della classe Reader e dei suoi metodi.

Di seguito le funzionalità dei metodi pubblici.

Reader.GetSensorList()

Restituisce in un array di Sensori[] la lista dei sensori installati nel sistema.

Reader.GetSensorListA()

Restituisce in un List<T> di oggetti Sensori (List<Sensore>) la lista dei sensori installati nel sistema.

Reader.Init(int baudRate)

Il metodo privato Reader.Init() viene richiamato dal costruttore per ricercare tutte le porte seriali

installate nel sistema, per ognuna di queste porte rilevate viene istanziato l'oggetto PortaCOM, di cui viene a sua volta richiamato il metodo PortaCOM.Init(). Tale metodo restituisce true se rileva almeno un sensore lineare collegato alla porta. Se ciò avviene questa istanza di PortaCOM viene aggiunta alla lista delle porteSeriali. Può capitare difatti che ad una porta COM di sistema non sia collegato alcun sensore lineare oppure vi sia collegato un altro apparato, in questo caso questa porta COM viene scartata e non viene aggiunta alla lista delle porteSeriali.

Reader.SendLASERON(bool on)

Manda in broadcast, attraverso tutte le porte della lista porteSeriali, il comando di accensione (on==true) o di spegnimento (on==false) dei laser di TUTTI i sensori del sistema.

Reader.SendVSER()

Manda in broadcast, attraverso tutte le porte della lista porteSeriali, il comando di passaggio alla velocità di 115200bps a TUTTI i sensori del sistema.

Reader.CloseAllPorts()

Questa funzione chiude tutte le porte aperte col metodo Reader.Init() ed elimina i corrispondenti handle.

Reader.SerializeSensors()

Serializza in formato XML la lista dei sensori installati nel sistema.

Reader.DeserializeSensors()

Deserializza in formato XML la lista dei sensori installati nel sistema.

La classe PortaCOM

La classe PortaCOM è il contenitore che possiede il riferimento ad una porta COM di sistema. Questa classe possiede la lista dei sensori lineari collegati fisicamente a quella data porta COM di sistema. Il metodo Init() consente l'inizializzazione della rete di sensori collegati. Tramite il metodo GetSensorList() è possibile ottenere la lista di tutti i sensori collegati alla porta COM. La lista privata Sensori contiene l'elenco dei sensori lineari collegati fisicamente a quella porta.

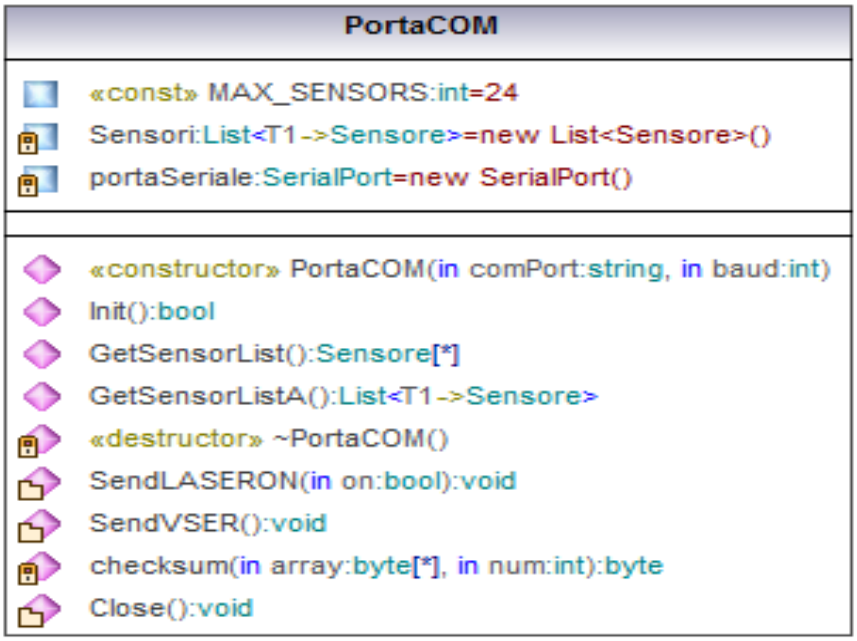


Figura 13: Rappresentazione della classe PortaCOM e dei suoi metodi.

Di seguito le funzionalità dei metodi.

PortaCOM.GetSensorList()

Restituisce in un array di Sensori[] la lista dei sensori collegati a questa porta COM.

PortaCOM.Init(int baudRate)

Il metodo privato PortaCOM.Init() viene richiamato dal costruttore della classe Reader per ricercare tutti i sensori collegati a quella porta COM. Viene effettuato un ciclo da con i che và da 1 a MAX_SENSORS, ad ogni ciclo viene istanziato un sensore con ID pari a i e poi viene inviato un comando di ACK a quel sensore i-esimo in quella data porta. Se il sensore i-esimo esiste e risponde con un messaggio di BACKACKNOLEDGE allora l'oggetto sensore viene aggiunto alla lista Sensori di questa porta. Tale metodo restituisce true se rileva almeno un sensore lineare collegato alla porta.

PortaCOM.SendLASERON(bool on)

Manda in broadcast, attraverso quella porta, il comando di accensione (on==true) o di spegnimento (on==false) dei laser di TUTTI i sensori del sistema. Ossia invia il messaggio LASERON nella porta COM, senza specificare alcun ID di sensore (ID=0).

PortaCOM.SendVSER()

Manda in broadcast, attraverso quella porta, il comando di passaggio alla velocità di 115200bps a TUTTI i sensori del sistema.

PortaCOM.Close()

Questa funzione chiude la porta COM di riferimento.

La classe Sensore

Questa classe rappresenta il sensore lineare connesso al sistema. Ogni istanza di questa classe identifica un reale sensore con un dato ID. Difatti la property NumeroSensore identifica un sensore fisico con quello specifico ID, unico in tutta la rete. Ci sarà quindi una sola istanza di un sensore con un dato valore per NumeroSensore. Questa classe incapsula, sotto forma di metodi, tutti i comandi che possono essere inviati al sensore lineare secondo le specifiche descritte al precedente capitolo. Possiede inoltre un riferimento interno alla porta COM cui è collegato.

La classe sensore memorizza i dati provenienti dal sensore in array di max 128 elementi (numero massimo di nmagini memorizza l ITION. L'array Acq essaggio SENDACQI i 128 possibili ba li array AcqniBarice i con un messaggio E

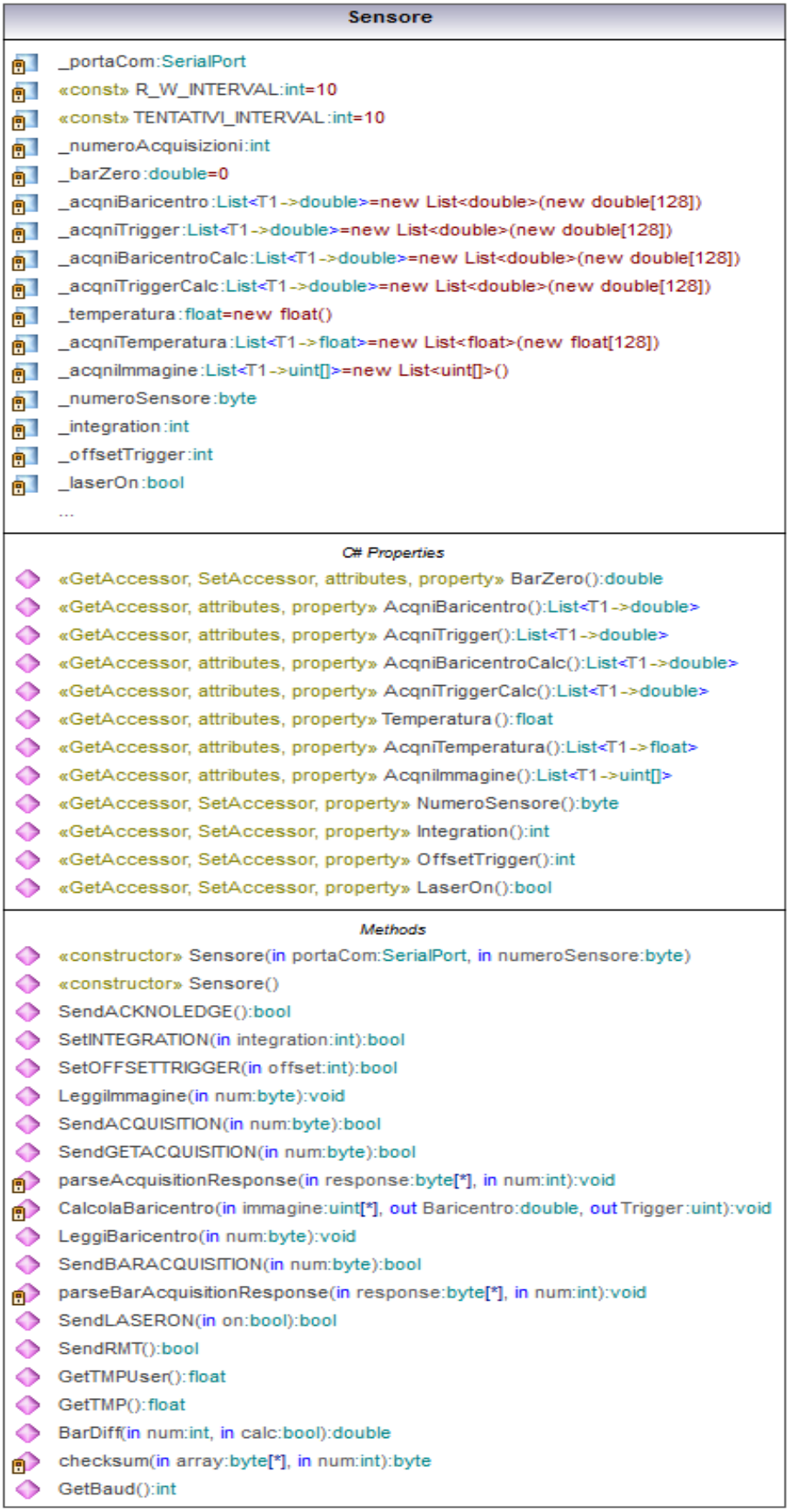


Figura 14: Rappresentazione della classe Sensore e dei suoi metodi.

Sensore.SendACKNOLEDGE():bool

Invia al sensore il comando di ACKNOLEDGE, se quel sensore restituisce BACKACNOLEDGE, la funzione restituisce *true*, restituisce *false* altrimenti.

Sensore.SetINTEGRATION(int integration):bool

Invia al sensore il comando di SETINTEGRATION, per definire il tempo di integrzione. Se quel sensore risponde con un BACKACKNOLEDGE, la funzione restituisce *true*, restituisce *false* altrimenti.

Sensore.SetOFFSETTRIGGER(int offset):bool

Invia al sensore il comando di SETOFFSETTRIGGER, specificando il valore di offset. Se quel sensore risponde con un BACKACKNOLEDGE, la funzione restituisce *true*, restituisce *false* altrimenti.

Sensore.SendLASERON(bool on):bool

Invia al sensore il comando di LASERON, specificando se accendere (on = true) o spegnere (on = false) il laser. Se quel sensore risponde con un BACKACKNOLEDGE, la funzione restituisce *true*, restituisce *false* altrimenti.

Sensore.SendRMT():bool

Invia al sensore il comando di RMT. Se quel sensore risponde con un BACKACKNOLEDGE, la funzione restituisce *true*, restituisce *false* altrimenti.

Sensore.GetTMP():bool

Invia al sensore il comando di TMP, restituisce la temperatura restituita dal messaggio BACKTMP. Se quel sensore risponde con un BACKTMP, la funzione restituisce *true*, restituisce *false* altrimenti. La temperatura restituita dal messaggio BACKTMP, viene anche memorizzata nella property Temperatura.

Sensore.SendACQUISITION(byte num):bool

Invia al sensore il comando di ACQUISITION specificando il numero di acquisizioni che il sensore deve effettuare. Se quel sensore risponde con un BACKACKNOLEDGE, la funzione restituisce *true*, restituisce *false* altrimenti.

Sensore.SendGETACQUISITION(byte num):bool

Invia al sensore il comando di GETACQUISITION specificando il numero dell'immagine che si vuole richiedere al sensore. Se la GETACQUISITION è coerente, quel sensore risponde con un

SENDACQUISITION contenente l'immagine rilevata dal sensore e la temperatura, la funzione quindi restituisce *true*, restituisce *false* altrimenti. La funzione, richiama il metodo privato parseAcquisitionResponse(), il quale memorizza le immagini restituite nella property AcqniImmagine e la temperatura nella property AcqniTemperatura, nell'indice corrispondente a num. La funzione, dopo aver richiamato la funzione Sensore.CalcolaBaricentro(), memorizza il baricentro ed il trigger calcolati dall'immagine nelle property AcqniBaricentroCalc e AcqniTriggerCalc rispettivamente, in corrispondenza dell'indice num.

Sensore.parseAcquisitionResponse(byte[] response, byte num)

Effettua il parsing del messaggio SENDACQUISITION. Il messaggio SENDACQUISITION è un array di 2053 bytes contenente la temperatura in una word e le 1024 word dei valori dei pixel che compongono l'immagine. Dopo aver effettuato il parsing memorizza l' immagine nella property AcqniImmagine e la temperatura nella property AcqniTemperatura, in corrispondenza dell'indice num.

Sensore.CalcolaBaricentro(byte[] immagine, out double baricentro, out double Trigger)

Calcola il baricentro ed il trigger di un'immagine. Il trigger T è la media dei valori dei pixels costituenti l'immagine. Il baricentro è dato dalla formula:

$$\frac{\sum_i X_i}{\sum_i i}$$

solo per le x_i tali che $x_i \geq T$ e dove i và da 1 a 1024.

Sensore.SendBARACQUISITION(byte num):bool

Invia al sensore il comando di BARACQUISITION specificando il numero del baricentro che si vuole richiedere al sensore. Se la BARACQUISITION è coerente, quel sensore risponde con un SENDBARACQUISITION contenente il baricentro ed il trigger rilevati dal sensore, la funzione quindi restituisce *true*, restituisce *false* altrimenti. La funzione, richiama il metodo privato parseBarAcquisitionResponse(), il quale memorizza i baricentri ed i trigger restituiti nella property AcqniBaricentro e la temperatura nella property AcqniTrigger, nell'indice corrispondente a num.

Sensore.parseBarAcquisitionResponse(byte[] response, byte num)

Effettua il parsing del messaggio SENDBARACQUISITION. Il messaggio SENDACQUISITION è un array di 12 bytes contenente il numeratore ed il denominatore del baricentro, nonché il valore di trigger utilizzato. Dopo aver effettuato il parsing memorizza il baricentro nella property AcqniBaricentro ed il trigger nella property AcqniTrigger, in corrispondenza dell'indice num.

Sensore.LeggiImmagine(byte num)

Questa funzione effettua la chiamata in sequenza delle funzioni Sensore.SendACQUISITION(byte num) e Sensore.SendGETACQUISITION(byte num). Tra la chiamata di una funzione e l'altra viene inserito un delay di 20 millisec. Questa funzione è utile se richiamata in un thread.

Sensore.LeggiBaricentro(byte num)

Questa funzione effettua la chiamata in sequenza delle funzioni Sensore.SendACQUISITION(byte num) e Sensore.SendGETBARACQUISITION(byte num). Tra la chiamata di una funzione e l'altra viene inserito un delay di 20 millisec. Questa funzione è utile se richiamata in un thread.

Persistenza

Lo stato delle impostazioni dei sensori installati nel sistema, come offset e tempo di integrazione, possono essere memorizzati in un file tramite la funzione Reader.SerializeSensors(string path). Possono essrere recuperati tramite la funzione Reader.DeserializeSensors(string path). Di seguito un esempio di file xml:

```
<?xml version="1.0" encoding="us-ascii"?>
<Sensore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore>
<NumeroSensore>3</NumeroSensore>
<Integration>13000</Integration>
<OffsetTrigger>0</OffsetTrigger>
<LaserOn>false</LaserOn>
</Sensore>
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore>
<NumeroSensore>10</NumeroSensore>
<Integration>0</Integration>
<OffsetTrigger>0</OffsetTrigger>
```

```
<LaserOn>false</LaserOn>
</Sensore>
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
<Sensore xsi:nil="true" />
</Sensore>
```

LISTA COMANDI versione 2.1D

1. Comando di ACKNOLEDGE (Master -> Slave):

ACK | ADR | XXX | XXX | CHK

ACK = 0x01

ADR = Indirizzo da 1 a 255

XXX = Non Usati

CHK = CheckSum

Richiede al dispositivo, il cui indirizzo è indicato da ADR, di rispondere con un comando di BACKACKNOLEDGE (BCK).

2. Comando di BACKACKNOLEDGE (Slave -> Master):

BCK | ADR | CHK

BCK = 0x02

ADR = Indirizzo da 1 a 255

CHK = CheckSum

Comando di risposta di un dispositivo slave, il cui indirizzo è indicato da ADR, a una richiesta di ACK.

3. Comando di SETINTEGRATION (Master -> Slave):

STI | ADR | INT[0] | INT[1] | CHK

STI = 0x10

ADR = Indirizzo da 0 a 255

INT[0] = Parte bassa della Word di dati

INT[1] = Parte alta della Word di dati

CHK = CheckSum

Comando che impone al dispositivo, il cui indirizzo è indicato in ADR, di impostare il tempo di integrazione in funzione del valore della Word (INT[1]:INT[0]) che va da 0 a 13200 (con 0 massimo periodo di integrazione). All'accensione il valore di default è 12900. Da notare che INT[0] è LSB mentre INT[1] è MSB. Il dispositivo slave risponderà con un comando BACKACKNOLEDGE. Se l'indirizzo è 0 il comando è riferito a tutti i dispositivi della linea (Broadcast), in questo caso però il dispositivo non da alcuna risposta.

4. Comando LASERON (Master -> Slave):

LAS | ADR | PWR | XXX | CHK

LAS = 0x30

ADR = Indirizzo da 0 a 255

PWR = Accensione Laser

XXX = Non Usati

CHK = CheckSum

Comando che impone accensione o spegnimento del Laser al sensore il cui indirizzo è indicato in ADR. Se PWR è 0 il Laser viene spento altrimenti qualsiasi valore non 0 fa accendere il Laser. Il dispositivo slave risponderà con un comando BACKACKNOLEDGE. Se l'indirizzo è 0 il comando è riferito a tutti i dispositivi della linea (Broadcast), in questo caso però il dispositivo non da alcuna risposta.

5. Comando TMP (Master -> Slave):

TMP | ADR | XXX | XXX | CHK

TMP = 0x82

ADR = Indirizzo da 1 a 255

XXX = Non Usati

CHK = CheckSum

Comando che chiede alla scheda di indirizzo ADR il valore della temperatura. Tale valore viene restituito col comando BACKTMP.

6. Comando di BACKTMP (Slave -> Master):

BTP | ADR | TMP[0] | TMP[1] | CHK

BTP = 0xC2

ADR = Indirizzo da 1 a 255

TMP[0] = Sono gli 8bit LSB di temperatura

TMP[1] = Sono i restanti 3bit MSB di temperatura

CHK = CheckSum

Comando di risposta di un dispositivo slave, il cui indirizzo è indicato da ADR, a una richiesta di lettura della temperatura. La temperatura è data tramite 12bit più un bit di segno (totale 13bit). Il dato è in complemento a due. La risoluzione è pari a 0.0625°C. L'intervallo di temperatura va da +150°C a -55°C.

7. Comando RMT (Master -> Slave):

RMT | ADR | XXX | XXX | CHK

RMT = 0x86

ADR = Indirizzo da 1 a 255

XXX = Non Usati

CHK = CheckSum

Comando che ordina al dispositivo ADR di fare un test della SRam. Se il test ha successo viene dato un BCK. Altrimenti non vi è risposta.

8. Comando di ACQUISITION (Master -> Slave):

ACQ | ADR | NUM | XXX | CHK

ACQ = 0x90

ADR = Indirizzo da 0 a 255

NUM = Numero acquisizione da fare da 1 a 128

XXX = Non Usati

CHK = CheckSum

Comando che impone al dispositivo, il cui indirizzo è indicato in ADR, di eseguire una serie di misure pari al numero indicato da NUM. Se NUM non rientra nell'intervallo da 1 a 128 il comando non viene eseguito. Se l'indirizzo è 0 il comando è riferito a tutti i dispositivi della linea (Broadcast). Il Master, dopo un certo tempo, deve inviare un comando di GETACQUISITION o di BARACQUISITION per farsi spedire le misure dal singolo slave.

9. Comando di GETACQUISITION (Master -> Slave):

GAQ | ADR | DAT | XXX | CHK

GAQ = 0x91

ADR = Indirizzo da 1 a 255

DAT = Selezione dati

XXX = Non Usati

CHK = CheckSum

Comando che impone al dispositivo, il cui indirizzo è indicato in ADR, di spedire l'ultima serie di misure eseguite tramite il comando SENDACQUISITION (SAQ). Se il campo DAT contiene il valore 0 lo slave invierà tutte le misure effettuate. Qualora invece il campo DAT specifichi un valore compreso tra 1 e 128 il dispositivo invierà solo la misura indicata da DAT. Se la richiesta non è consistente (esempio non sono state fatte misure o DAT supera il valore 128) viene restituito un BCK.

10. Comando SENDACQUISITION (Slave -> Master):

SAQ | ADR | TMP[0] TMP[1] | PIX[0]...PIX[1023] | CHK

SAQ = 0x99

ADR = Indirizzo da 1 a 255

TMP[0…1] = Temperatura del Sensore

PIX[n][0…1023] = Pixel del Sensore della n-esima Misura (Rappresenta una Word)

CHK = CheckSum

Comando (Pacchetto) che il sensore, il cui indirizzo è indicato in ADR, invia al master per comunicare i dati della misura. Questo pacchetto viene inviato in risposta al comando GAQ. Il dato si compone di 2 Byte in complemento a 2 che indicano la temperatura e poi la misura da 1024 Word (ogni Word 2 Byte) provenienti dal sensore lineare. Tale pacchetto può essere ripetuto in numero da 1 a 128 a seconda di quella che è stata la richiesta di misura effettuata col comando ACQ. Notare che ogni blocco PIX contiene 1024 Pixel e ogni pixel è rappresentato da una Word ove il primo Byte rappresenta gli 8 bit MSB mentre il secondo i restanti 2 bit LSB.

Il Singolo pixel essendo a 10bit viene rappresentato con 2 bytes. In Particolare il generico dato di pixel è dato da PIX[i] = BYTE_H[i] : BYTE_L[i]. Dove in BYTE _H[i] sono contenuti i bit da B9 … B2 e in BYTE_L[i] sono contenuti i bit da B1 … B0. Il Valore di ampiezza del generico pixel PIX[i] viene quindi ottenuto con la seguente operazione 4 * BYTE _H[i] + BYTE _L[i].

11. Comando di BARACQUISITION (Master -> Slave):

BAQ | ADR | DAT | XXX | CHK

BAQ = 0x93

ADR = Indirizzo da 1 a 255

DAT = Selezione dati

XXX = Non Usati

CHK = CheckSum

Comando che impone al dispositivo, il cui indirizzo è indicato in ADR, di spedire l'ultima serie di misure Baricentro eseguite tramite il comando SENDBARACQUISITION (ZAQ). Se il campo DAT contiene il valore 0 lo slave invierà tutte le misure effettuate. Qualora invece il campo DAT specifichi un valore compreso tra 1 e 128 il dispositivo invierà solo la misura indicata da DAT. Se la richiesta non è consistente (esempio non sono state fatte misure o DAT supera il valore 128) viene restituito un BCK

12. Comando SENDBARACQUISITION (Slave -> Master):

ZAQ | ADR | TRIG[0] TRIG[1] | BARNUM[0]...[3] | BARDEN[0]...[2] | CHK

SAQ = 0x9A

ADR = Indirizzo da 1 a 255

TRIG[0…1] = Livello trigger usato per calcolo Baricentro

BARNUM [0…3] = Numeratore Baricentro

BARDEN [0…2] = Denominatore Baricentro

CHK = CheckSum

Comando (Pacchetto) che il sensore, il cui indirizzo è indicato in ADR, invia al master per comunicare i dati della misura. Questo pacchetto viene inviato in risposta al comando BAQ. Il dato si compone di 2 Byte (Lsb:Msb) denominati TRIG[0…1] che contengono il livello di trigger usato per il calcolo del baricentro. Tutti i punti del frame aventi ordinata maggiore di tale livello partecipano al calcolo del baricentro. Si noti che il livello di Trigger = Offset + Media.

L'Offset è un valore impostabile dall'utente tramite il comando SETOFFSETTRIGGER (OFT) mentre la media è il malore medio di ordinata calcolato su tutto il frame. La seconda serie di 4 bytes (Lsb:Msb) denominati BARNUM [0…3] rappresentano il numero intero che compone il numeratore del baricentro. Similmente i successivi 3 bytes (Lsb:Msb) denominati BARNUM [0…2] rappresentano il denominatore intero che compone il denominatore del baricentro. Alla fine il valore del Baricentro (ascissa) si ottiene eseguendo il rapporto: BARNUM/BARDEN.

13. Comando di SETOFFSETTRIGGER (Master -> Slave):

OFT | ADR | OFF[0] OFF[1] | CHK

OFT = 0x94

ADR = Indirizzo da 1 a 255

OFF[0…1] = Livello offset

CHK = CheckSum

Comando che impone al dispositivo, il cui indirizzo è indicato in ADR, di impostare il livello di offset del trigger in funzione del valore della Word (OFF[1]:OFF[0]) che va da 0 a 1023.

All'accensione il valore di default è 0. Da notare che OFF[0] è LSB mentre OFF[1] è MSB. Il dispositivo slave risponderà con un comando BACKACKNOLEDGE.

Se l'indirizzo è 0 il comando è riferito a tutti i dispositivi della linea (Broadcast), in questo caso però il dispositivo non da alcuna risposta.

Si ricorda che per il calcolo del baricentro vengono presi tutti i pixel aventi ordinata maggiore del Trigger. Il Trigget è dato dalla somma del valore di Offset con il valore della media di ordinata di tutti i pixel.

14. Comando di VSER (Master -> Slave):

VSR | ADR | XXX | XXX | CHK

ACQ = 0x92

ADR = 0

XXX = Non Usati

CHK = CheckSum

Il Comando è solo in Broadcast e impone a tutti i dispositivi di passare da una comunicazione seriale iniziale di default di 57600bps a una comunicazione di 115200bps.

La comunicazione Seriale:

La velocità di trasmissione dati è impostata all'accensione a 57600bps (salvo commutare in seguito a 115200bps). Il singolo dato (parola) del pacchetto si compone di un bit di start, 8 bit di dati e un bit di stop. I pacchetti dal Master al Sensore sono di lunghezza fissa pari a 5 parole.

La durata di un pacchetto quindi si aggira attorno ai 0.868mS (a 57600bps). Il sensore ha un timeout di lunghezza pacchetto pari a 3.39mS.

Se si desidera far andare il sistema a 115200bps si consigli sempre e comunque di effettuare alla velocità iniziale di 57600bps un giro di ACK in modo da sincronizzare tutti i dispositivi della catena di misura. E solo in seguito commutare a 115200bps. Anche in questo caso per sicurezza è bene rifare un giro di ACK.

Un tipico Esempio:

Una tipica sequenza di comandi è la seguente:

ACK: Per verificare che il dispositivo risponda con un BCK

LAS: Comando di accensione del Laser

STI: Impostazione del Tempo di Integrazione

OFT: Impostazione dell'Offset che contribuisce a determinare il livello di Trigger

ACQ: Comando di acquisizione Frames

GAQ o BAQ: Lettura dei Frames (Per intero o solo il Baricentro