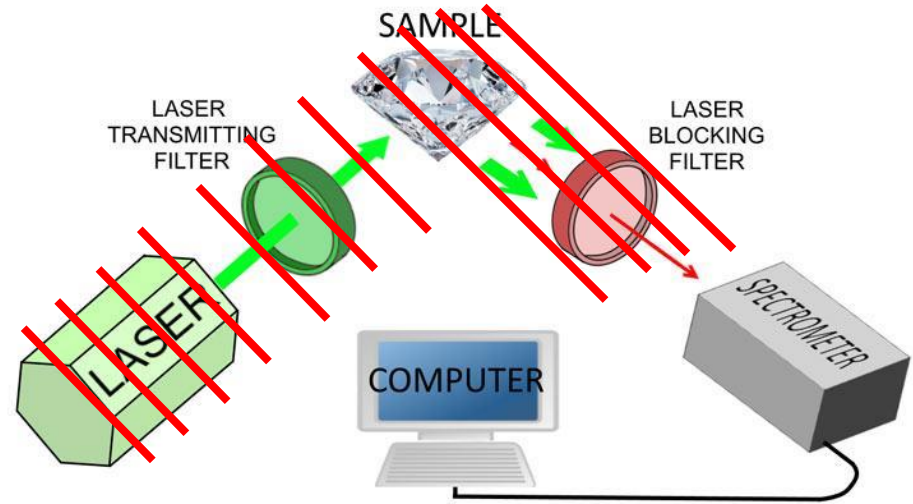


# Progetto tecniche digitali di acquisizione dati - Colombini William

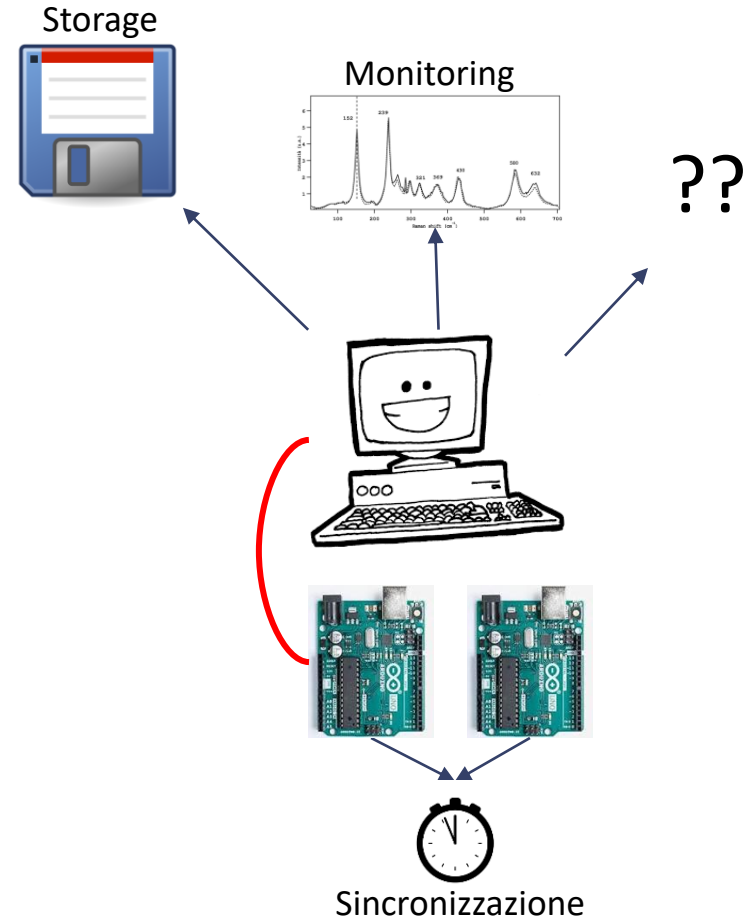
**Obiettivo:** Simulazione di acquisizione dati da un CCD-array mediante l'uso di due Arduino. La simulazione riguarda presunti dati provenienti da uno spettrometro Raman, si dovrà quindi tenere conto del *range dello spettro* da analizzare, della presenza di *picchi caratteristici* degli elementi e del *background* dato dal rumore di fondo della CCD.



## Parti del progetto:

Fondamentalmente il progetto si divide in tre macroblocchi:

1. La parte riguardante la simulazione dati da parte dei due Arduino;
2. la parte riguardante la raccolta dati da parte del computer;
3. la parte riguardante l'analisi delle performance.



**Inizio progetto: TO DO LIST** ci sono alcune cose da **stabilire** prima di dare il via al lavoro di programmazione

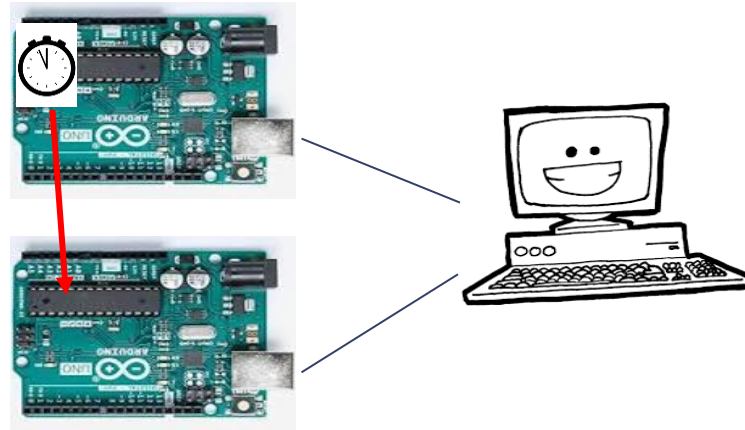
**0) Metodo di generazione e presa dati:** Il dato sarà raccolto in push o in pull?

Come sincronizzo gli arduini?

L'idea é di lavorare in *push*, gli arduini inviano i dati con una certa frequenza che vengono poi raccolti dal pc e poi elaborati per il salvataggio ed il monitoring.

La sincronizzazione avviene sfruttando come “trigger” per uno il clock dell'altro (il Master).

L'idea é (usando la libreria Wire.h) di far si che il Master ogni volta che conclude l'invio dica allo Slave di procedere.



# 1) Definire l'evento: Il dato prodotto dagli arduini come sarà?

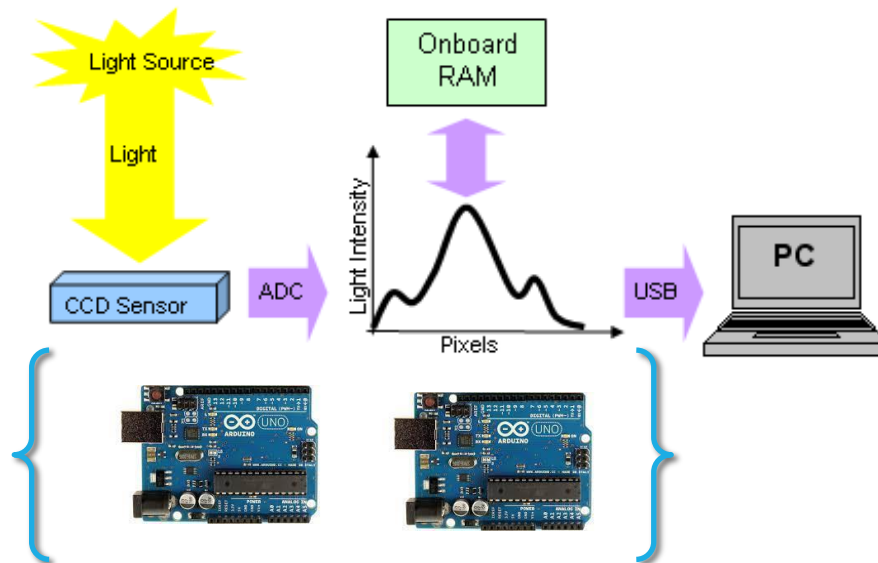
Il dato dell'arduino corrisponde a un dato già elaborato e digitalizzato da un'ADC.

L'idea é di avere un CCD array monodimensionale di 1024 pixels in cui ogni pixel ha una certa capacità di storage (es. 256 livelli di intensità) e che "pusha" fuori il dato ogni  $\Delta t$  (es. 200ms).



## Stabilito per la prima versione:

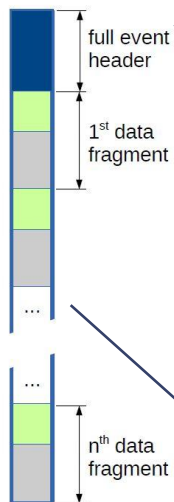
- CCD da 1024 segmenti (10bit);
- 2 arduini, ciascuno che elabora 512 segmenti (9 bit);
- 256 livelli di intensità per segmento (8bit).



- 2) **Definire il dataformat:** In che formato verrà mandato l'evento ? Occorre stabilire qual é il formato dell'evento considerando espandibilità, possibilità di fare controlli etcetc.

## Full event header

- **Checksum:** begin of frag (e.g.: 0xAA1234AA)
- **Fragment size:** where actual data ends
- **Header size:** where actual data starts
- **Time/bunchID:** timestamp
- **Run number**
- **Event classification**
- **Error words**
- **Array of offset (one for each fragment)**
  - Implemented only if random access is required
  - Otherwise, just navigate from fragment to fragment



CHECKWORD - DA7AEC00 (32bit)			
TIME COUNT (32 bit)			
Num_of_tot Detector (4bit)	Num_of_tot channels (12bit)	Controllo, versione, lunghezza header, estensioni varie (16 bit)	
Det-ID (4bit)	Num_of_Ch in Detector (12 bit)	Errori e controllo, estensioni varie (16 bit)	
Det-ID (4 bit)	Ch-ID (12 bit)	Ch Data (8 bit)	controllo ed estensioni (16 bit)
... Dati da tutti i canali del detector			
ENDWORD detector- EOFDA7A+Det-ID			
...			

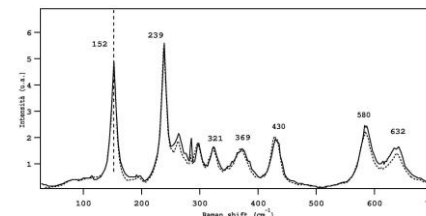
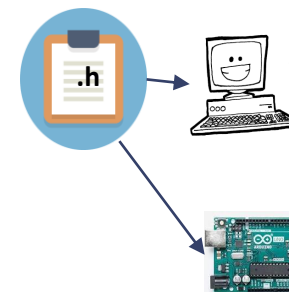
Parte prima del progetto: ignoriamo la fisica e iniziamo ad elaborare un sistema di invio e ricezione dati basato sul dataformat definito ma con dati semplici.

## TO DO LIST:

✓ 1) **Creare un file \*.h** contenente le informazioni necessarie per pc e Arduino come alcuni `#define` per evitare durante le riscritture di dover cambiare tutti i valori singolarmente.

✓ 2) **Scrivere codici base** che saranno implementati ed elaborati, sia lato arduino che lato pc. In particolare creare un sistema di invio e lettura dati dagli arduino verso il pc che li legge e trascrive in un file.

✓ 3) **Elaborare lato pc** aggiungendo gli strumenti necessari per il monitoring e per i controlli.





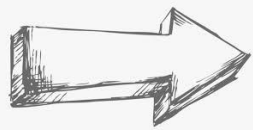
**Problema** di memoria emerso durante l'implementazione su singolo arduino. Sembrerebbe che per ogni arduino il massimo di memoria utilizzabile sia circa 2048 bytes per le variabili globali dinamiche.



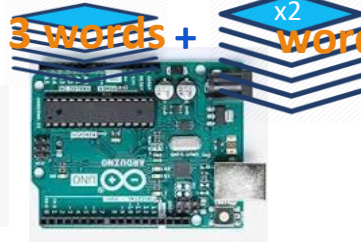
**E se si rimpicciolisce il buffer dati?** Invece che inizializzare un buffer dati da 512+header word e inviare tutto contemporaneamente ne inizializzo 2: uno per gli header e uno che invia 2 volte 256 canali. L'header deve essere inviato rapidamente quindi si é scelto un buffer a parte. Per i dati si é scelto il modo "intuitivamente" più efficiente creando due buffer il più grande possibile.



Header + Data  
~510 words



Header  
~ 3 words + Data ~256  
x2 words



**Problema** col sistema di trigger. Prima lo Slave preparava il buffer da inviare e quando il master lo chiamava inviava i dati. Dovendo riscrivere adesso il buffer dati questa cosa é fattibile solo parzialmente. Occorre trovare la soluzione più ottimale per la creazione e l'invio dati dello slave.

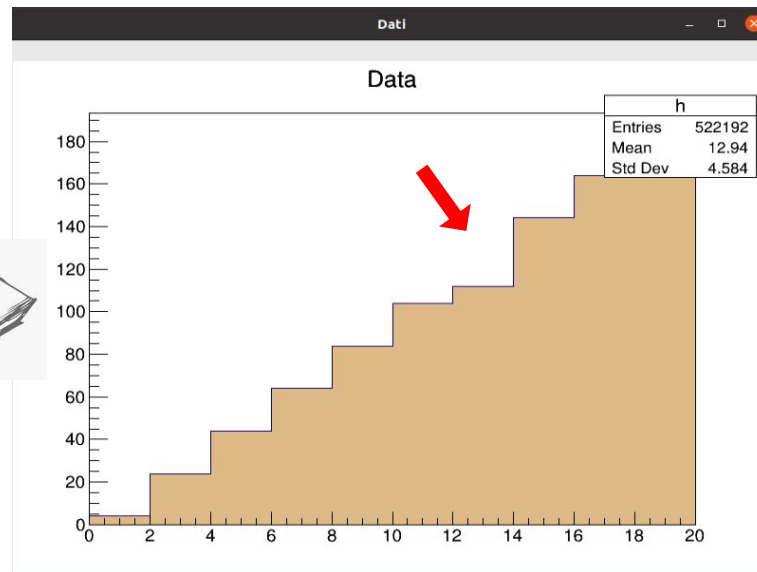
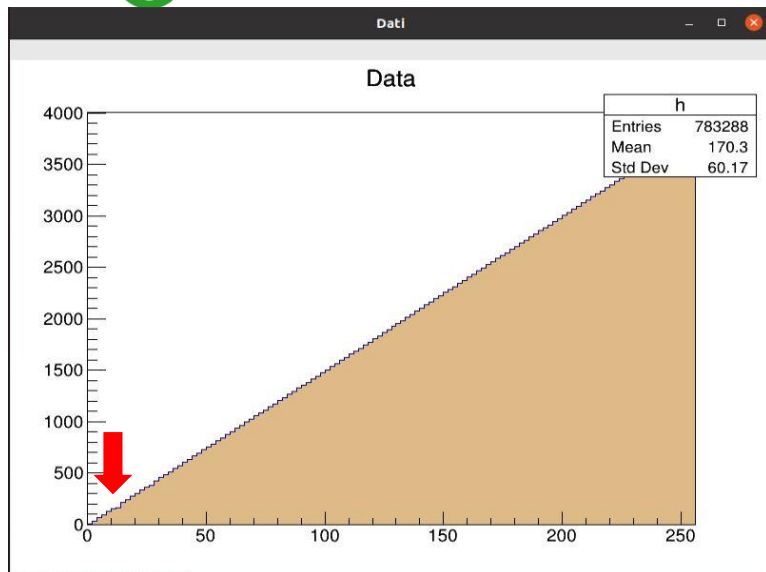


**Riguardo alla parte 3)** É stato sviluppato un codice SINGLE THREAD che:

- raccoglie i dati dai due arduini;
- effettua i controlli sulle checkword e sull'Event count;
- salva in un file i dati;
- fa il monitoring ogni volta che finisce la lettura dell'evento.



**Problema** Per testare il sistema gli arduini inviano un segnale che dovrebbe dare luogo a un istogramma a scalini, si osservano però un paio di problemi







“Fortunatamente” in contemporanea durante il controllo dell’event counter veniva evidenziato un errore nell’evento 13 (0x0d) che veniva letto come 10 (0x0a). Essendo un problema emerso durante le lezioni é stato risolto in modo rapido.

## termios

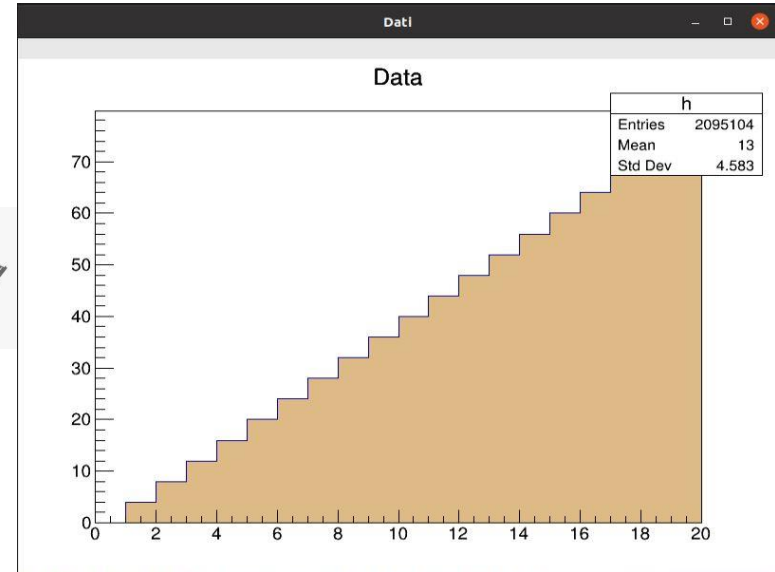
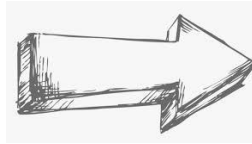
- Settare raw mode in C (via termios) non è banale
  - Per dettagli vedi “man termios” o questo [link](#)
  - Nella configurazione che usavo mancava la disabilitazione del parametro **ICRNL**
    - Translate carriage return to newline on input

```
struct termios raw;  
tcgetattr(fd, &raw);  
raw.c_iflag &= ~(BRKINT | ICRNL  
                | INPCK | ISTRIP | IXON);  
raw.c_oflag &= ~(OPOST);  
raw.c_cflag |= (CS8);  
raw.c_lflag &= ~(ECHO | ICANON | IEXTEN | ISIG);  
tcsetattr(fd, TCSAFLUSH, &raw);
```

Michele & Andrea

Debugging

19




**Parte seconda del progetto:** una volta scritto un codice funzionante vediamo come ottimizzarlo e come gestire eventuali imprevisti durante la presa dati

**! Problema 1)** Durante l'analisi dei grafici è emerso un problema relativo ai tempi di acquisizione dell'arduino. In particolare lo slave impiega molto più tempo del master per inviare i dati.

```
bil@bil-VirtualBox:~/Documents/Project/I0part/Mont
bil@bil-VirtualBox:~/Documents/Project/I0part/Mont
Ready to receive the words of /dev/ttyACM1
Ready to receive the words of /dev/ttyACM1
File opened
Time to read : 3693
Time to read : 2150
Time to read : 2150
Time to read : 2151
Time to read : 2148
Time to read : 2154
Time to read : 2150
Time to read : 2150
Time to read : 2151
Time to read : 2148
Time to read : 2151
Time to read : 2154
Time to read : 2150
Time to read : 2150
Time to read : 2148
Time to read : 2150
Time to read : 2151
Time to read : 2153
Time to read : 2157
Time to read : 2144
Time to read : 2148
Time to read : 2150
Time to read : 2154
Time to read : 2150
Time to read : 2151
Time to read : 2148
Time to read : 2150
Time to read : 2155
Time to read : 2149
Time to read : 2149
Time to read : 2150
```

```
bil@bil-VirtualBox:~/Documents/Project/I0part/Mont
bil@bil-VirtualBox:~/Documents/Project/I0part/Mont
Ready to receive the words of /dev/ttyACM1
Ready to receive the words of /dev/ttyACM1
File opened
Time initialize : 16
Time head master : 0
Time data master : 2138
Time header slave : 0
Time data slave : 2185
Time head master : 0
Time data master : 0
Time header slave : 3
Time data slave : 2178
Time head master : 0
Time data master : 0
Time header slave : 3
Time data slave : 2182
Time head master : 0
Time data master : 0
Time header slave : 3
Time data slave : 2181
Time head master : 0
Time data master : 0
Time header slave : 3
Time data slave : 2177
```

Tempo in ms

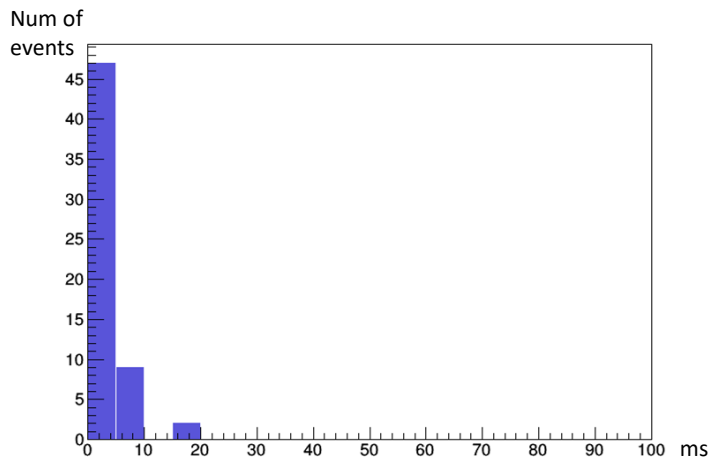
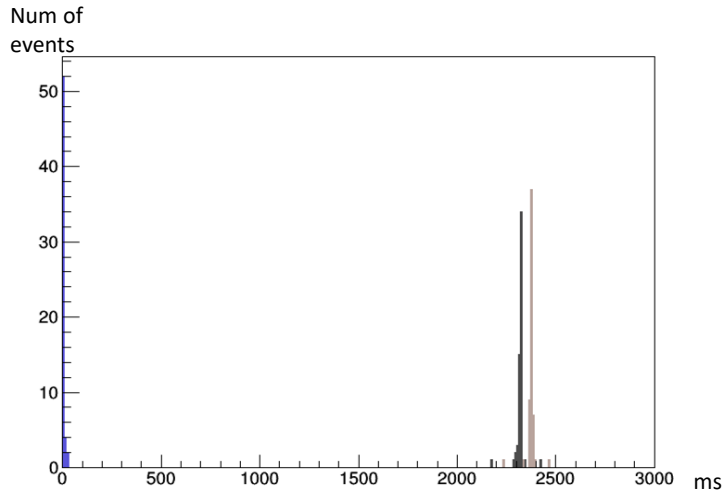


**! Problema 2)** Durante l'acquisizione dati ogni tanto ci sono errori: viene "perso" un byte tra la lettura e la scrittura su file portando a una desincronizzazione tra arduini e computer.

```
Got 4 bytes, 0Xe0fda7a2
Time slave sending : 2155
Time total sending : 2194
Got 4 bytes, 0Xda7aec00
Got 4 bytes, 0X 220
Got 4 bytes, 0X24000040
Got 4 bytes, 0Xe0fda7a1
Time master sending : 1
Got 4 bytes, 0Xe0fda7a2
Time slave sending : 2154
Time total sending : 2186
Got 4 bytes, 0Xda7aec00
Got 4 bytes, 0X 221
Got 4 bytes, 0X24000040
Got 4 bytes, 0Xe0fda7a1
Time master sending : 2
Got 4 bytes, 0Xe0fda7a2
Time slave sending : 2158
Time total sending : 2197
Got 4 bytes, 0Xda7aec00
Got 4 bytes, 0X 222
Got 4 bytes, 0X24000040
Got 4 bytes, 0Xe0fda7a1
Time master sending : 2
Got 4 bytes, 0X 4920b1
Time slave sending : 6643
Time total sending : 8215
Got 4 bytes, 0Xda7aec00
Got 4 bytes, 0X 223
Got 4 bytes, 0X24000040
Got 4 bytes, 0X 4d10ab
Time master sending : 3
S: Wrong detector reading
```



## Partiamo dal problema 1)

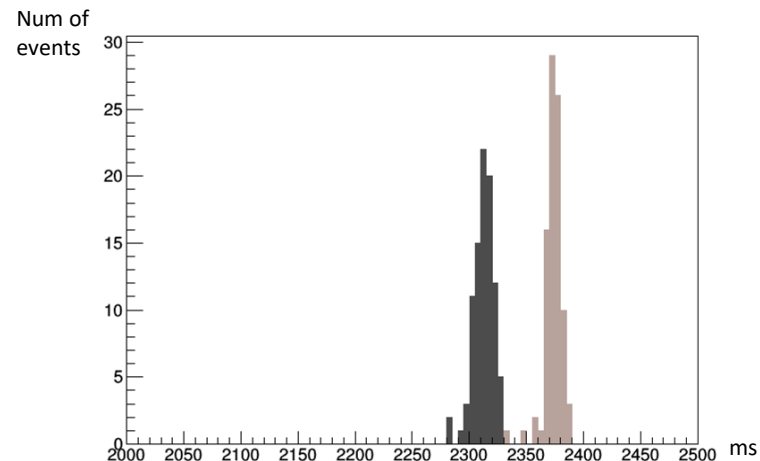


Legenda:

Blu: tempo di invio e lettura del master;

Nero: tempo di invio e lettura dello slave;

Grigio: tempo totale (invio, lettura e draw grafico).





**L'arduino é difettoso?** Essendo solo uno dei due arduini a dare problemi magari é un problema dell'arduino. Tuttavia invertendo i due arduini o cambiandoli con un terzo arduino il problema persiste.



**Magari é un problema software, se togliessi la libreria Wire.h<sup>\*</sup>?** Provando un nuovo metodo di comunicazione, in effetti qualcosa é cambiato. Tuttavia il tempo totale rimane praticamente invariato.

Legenda:

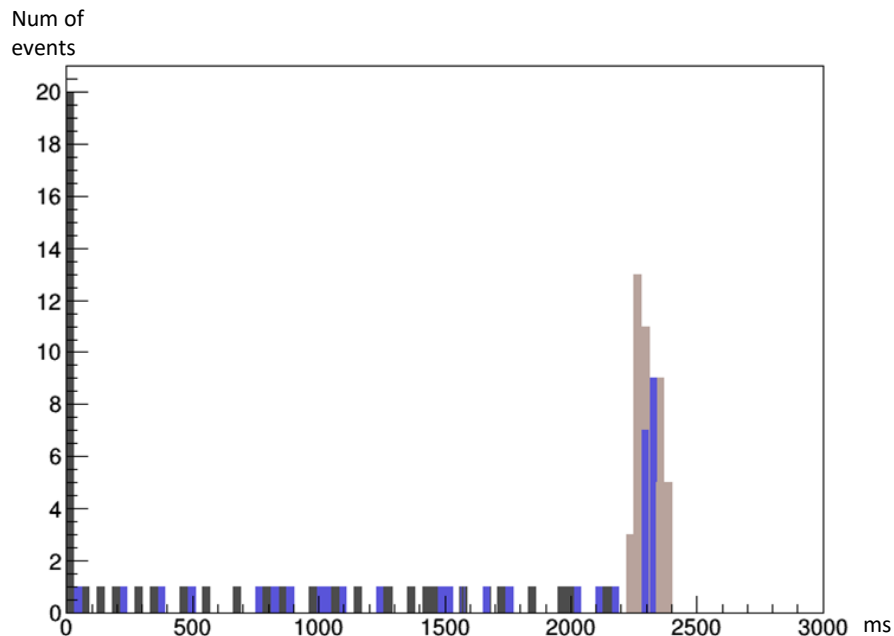
Blu: tempo di invio e lettura del primo;

Nero: tempo di invio e lettura del secondo;

Grigio: tempo totale (invio, lettura e draw grafico).

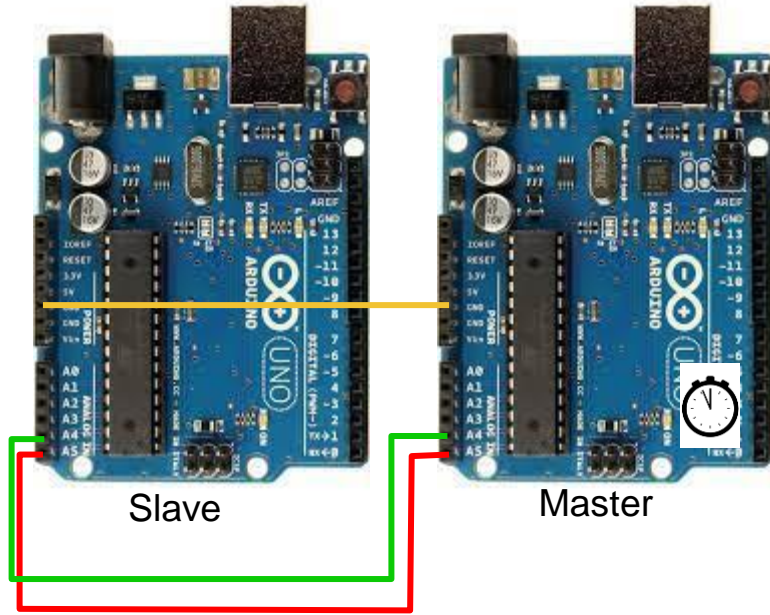
<sup>\*</sup>

La libreria che gestisce lo scambio di dati tra Master e Slave.



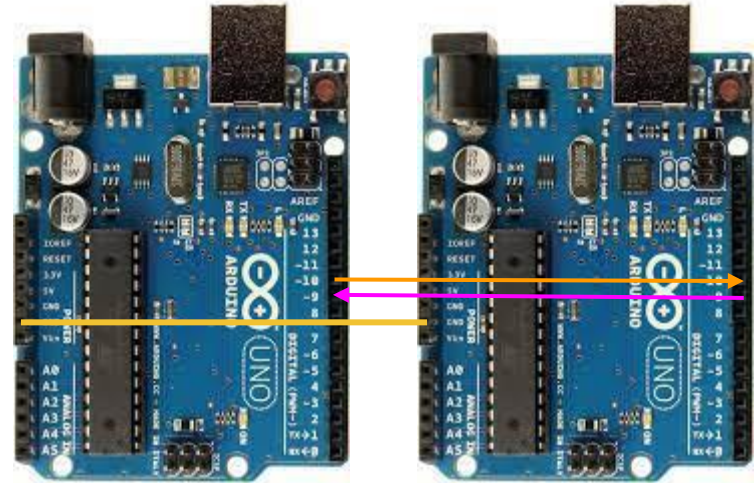
## Differenze tra i due metodi di comunicazione:

*Con libreria wire.h*



1. Il master invia i dati alla seriale;
2. Il master invia un segnale da 1 bit allo slave (0 é tutto ok, 1 ci sono stati problemi) seguito da un event counter;
3. Lo slave confronta il suo event counter con quello arrivato ed invia i dati alla seriale;

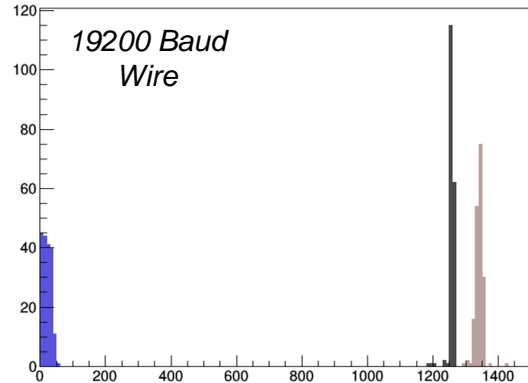
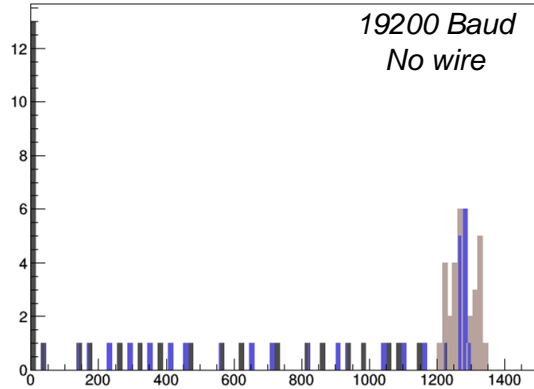
*Senza libreria wire.h*



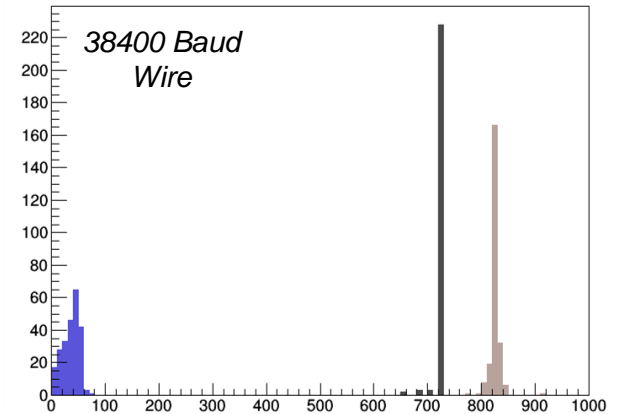
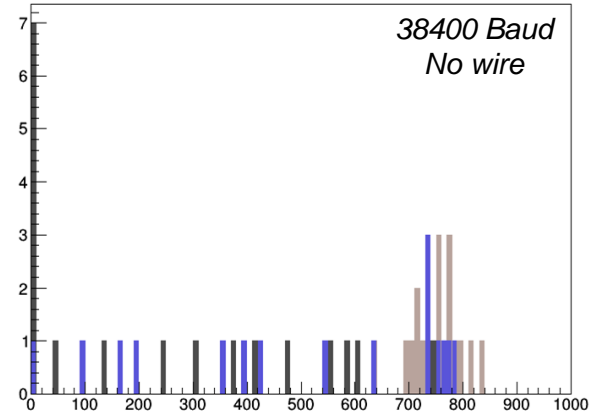
1. Il primo arduino setta il pin 10 come HIGH;
2. Il primo arduino invia i dati;
3. Il primo arduino setta il pin 10 come LOW;
4. Il primo arduino aspetta che il pin 9 sia LOW;
5. Il secondo arduino setta il pin 9 come HIGH;
6. Il secondo arduino invia i dati;
7. Il secondo arduino setta il pin 9 come LOW;
8. Il secondo arduino aspetta che il pin 10 sia LOW.



**Se alzassi il baud?** Alzando il baud aumenta il numero di bit inviati al secondo quindi, pur non risolvendo il problema legato alla differenza in tempi tra master e slave si riduce il tempo di ricezione dati dallo slave.



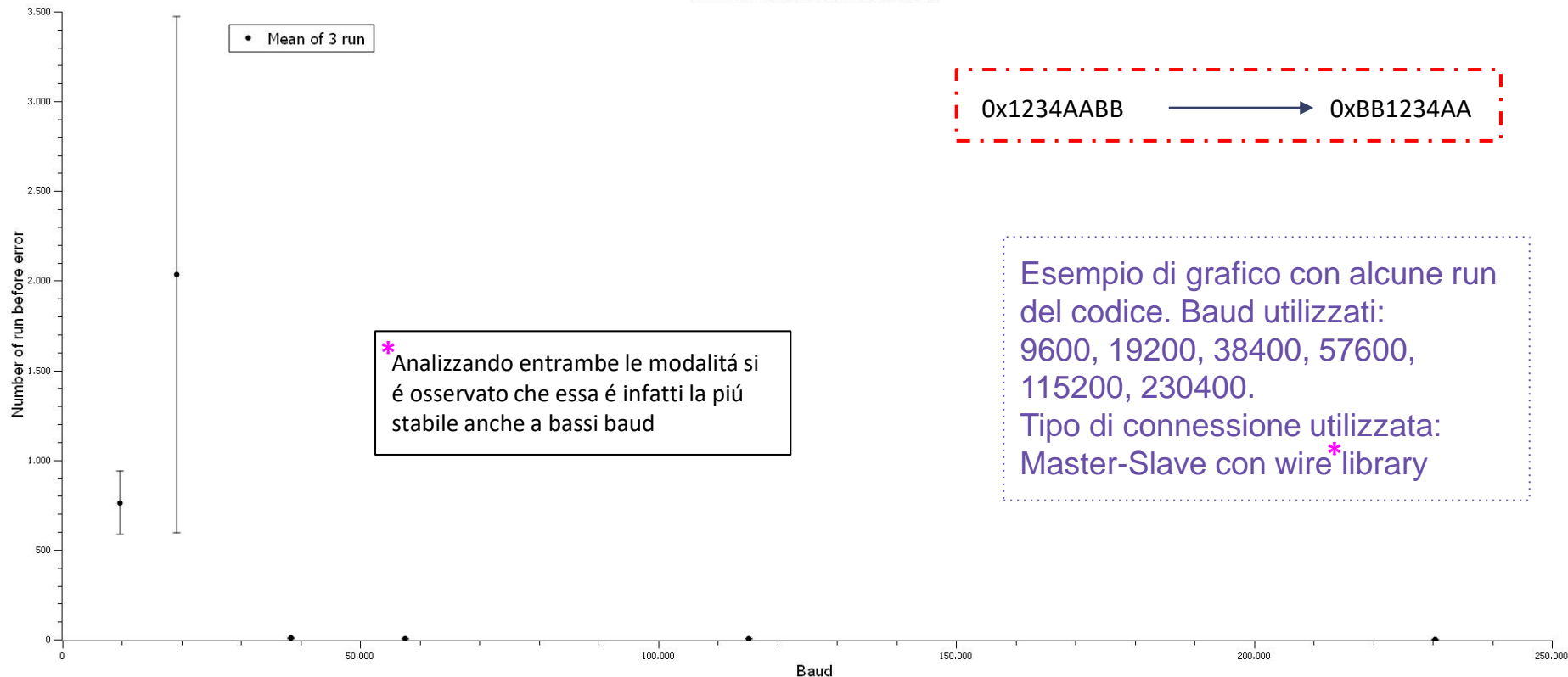
Esempi grafici a 19200 baud ed a 38400 baud con i due metodi di comunicazione





**Problema 2.2)** Aumentando il baud si risolve in parte la lentezza della presa dati ma cresce la frequenza con cui avviene lo sfasamento, quindi diventa particolarmente inefficiente la lettura ad alti baud. Inoltre il fenomeno é abbastanza imprevedibile.

code error and baud correlation



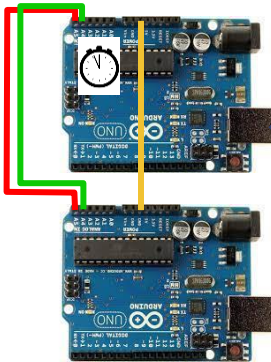


## Test effettuati per inquadrare meglio il problema:

Dopo aver semplificato al minimo indispensabile il codice dei due arduini facendo sì che il master inviasse solo la stringa 0xAABBCCDD e lo slave solo la stringa 0xC1A0C1A0 sono stati effettuati vari test:

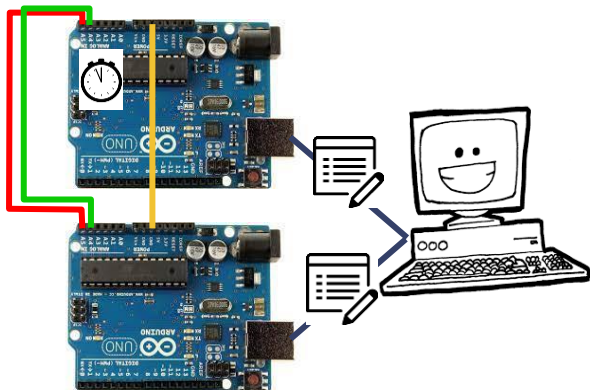


Testando il singolo arduino a vari baud non sussiste nessun problema.

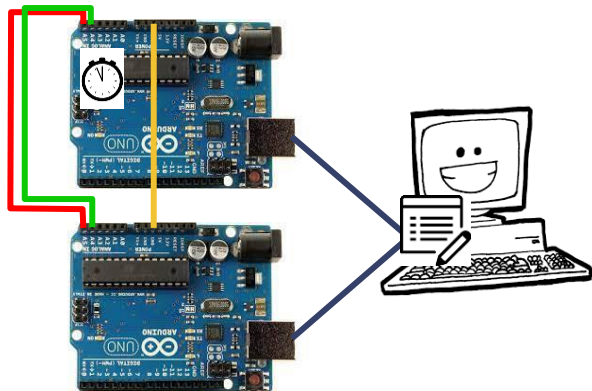


Similmente facendo runnare normalmente i due arduini ma leggendo i dati da uno solo di essi non si incontrano errori di desincronizzazione.





Leggendo contemporaneamente i due arduini ma facendo partire separatamente un programma di lettura e scrittura dati per ciascuno non ci sono problemi di alcun tipo per i due arduini.



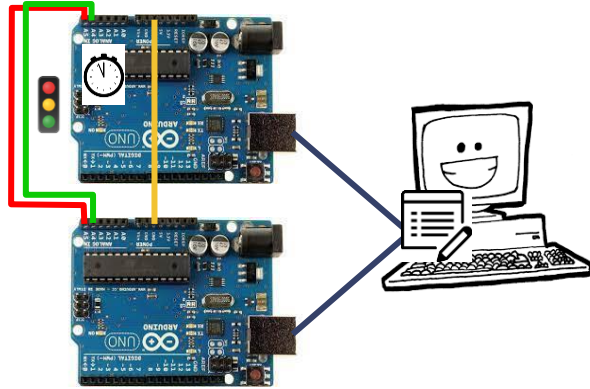
Leggendo con un solo programma due arduini ma definendo una variabile differente per ogni dato inviato dagli arduino non ci sono problemi nella presa dati, ma aggiungendo una checkword per entrambi gli arduini talvolta sorgono problemi.



Definendo poi una nuova variabile per leggere la checkword essa talvolta appare in mezzo ai dati.



In tutti i test é emerso che a baud particolarmente alti (115200, 230400) il programma a volte dava errori direttamente al primo ciclo di lettura.



***Se si provasse a mettere un delay di qualche secondo all'apertura della porta seriale del master?***

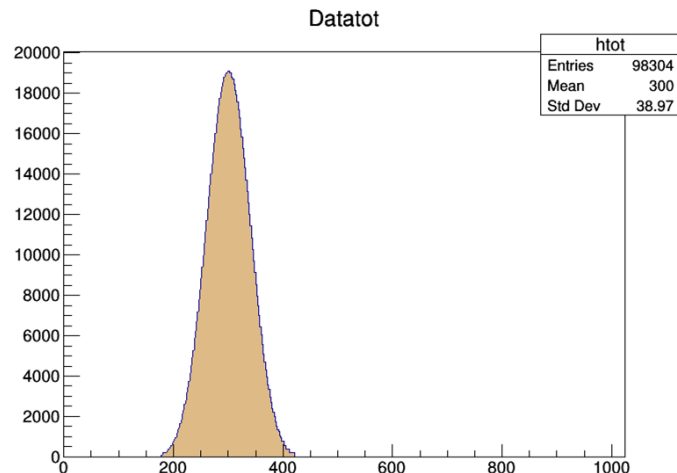
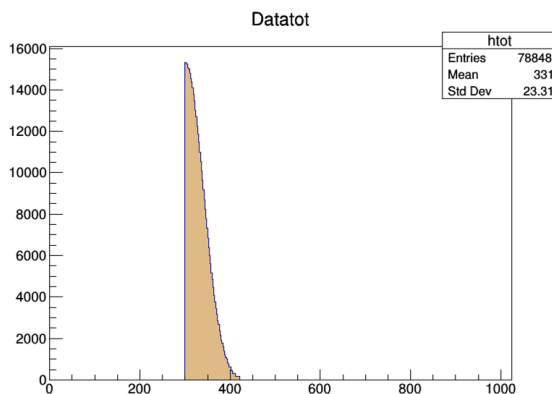
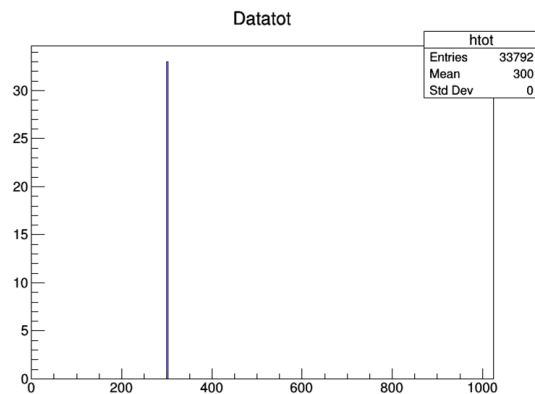
Si osserva che questa soluzione risolvere i problemi di acquisizione al primo ciclo.

Da una seconda analisi però é emerso che questo, unito all'utilizzo di qualche variabile in più per l'acquisizione, ha risolto quasi totalmente gli errori di sincronizzazione.

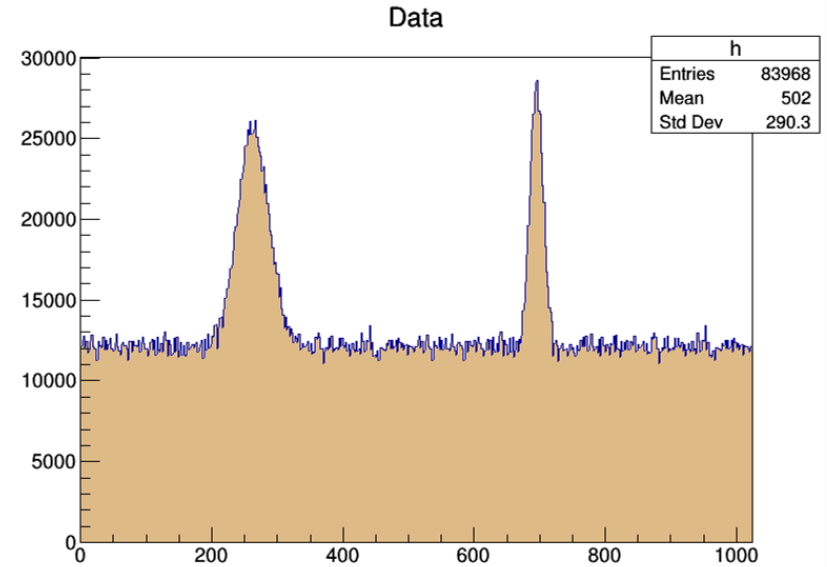
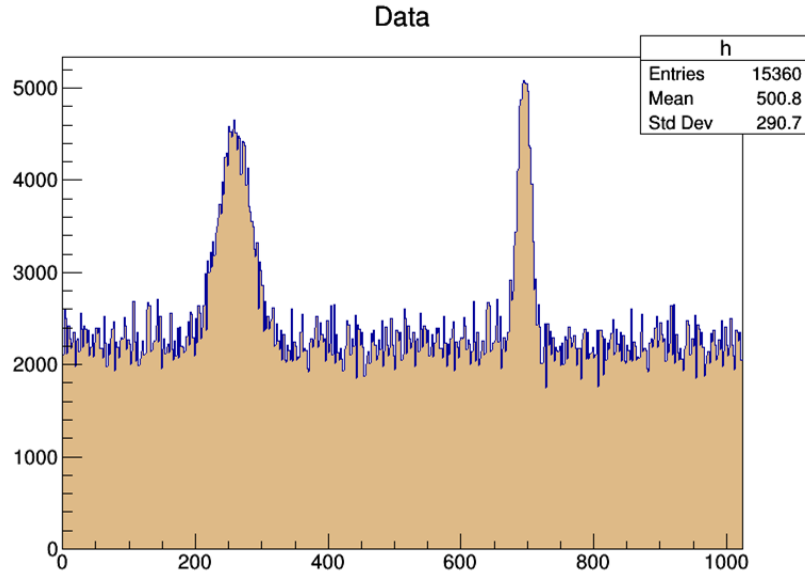
**Parte terza del progetto:** Sviluppare meglio il codice dell'arduino. Lo scopo é quello di far si che l'arduino invii un segnale di “background” uniforme e casuale a cui si sommano dei picchi che per semplicitá saranno di forma gaussiana. Anche in questa fase si é andati per grado:

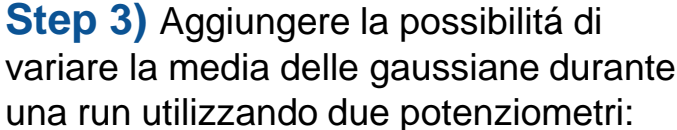
**Step 1)** Inviare una gaussiana comune ai due arduini con media e deviazione standard fissate;

Dopo qualche tentativo fallito a causa della definizione errata di alcune variabili utilizzate per la costruzione della gaussiana, si é ottenuta una forma accettabile.

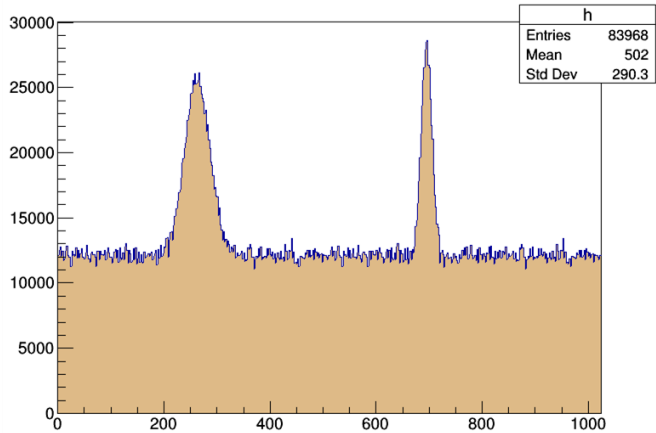


**Step 2)** Aggiungere una seconda gaussiana con media e deviazione standard diversa ma sempre fissa ed aggiungere il background;

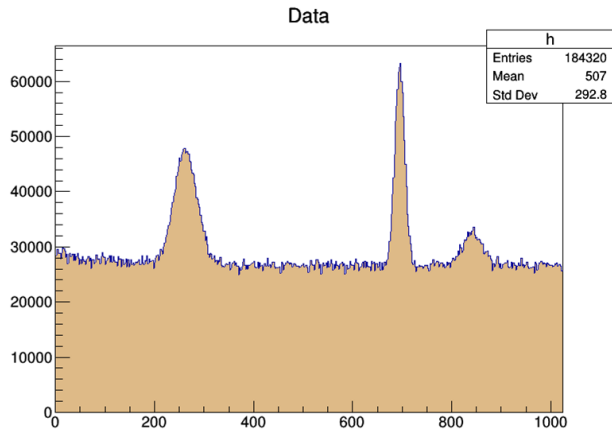




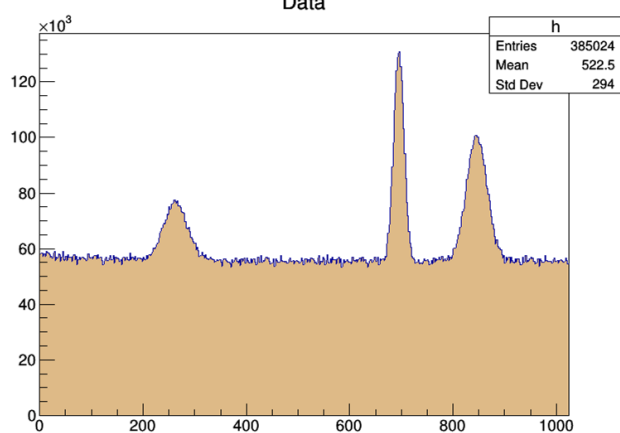
Data



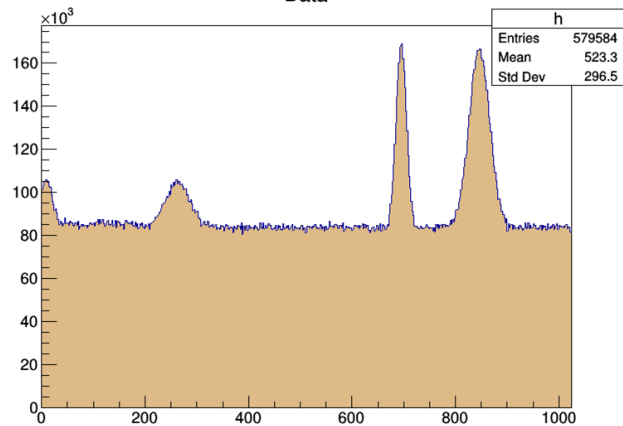
Data



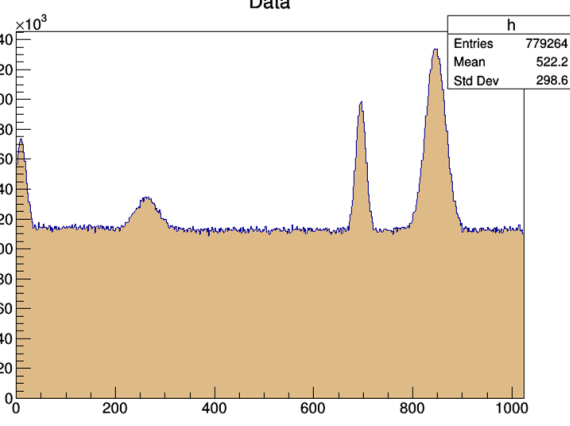
Data



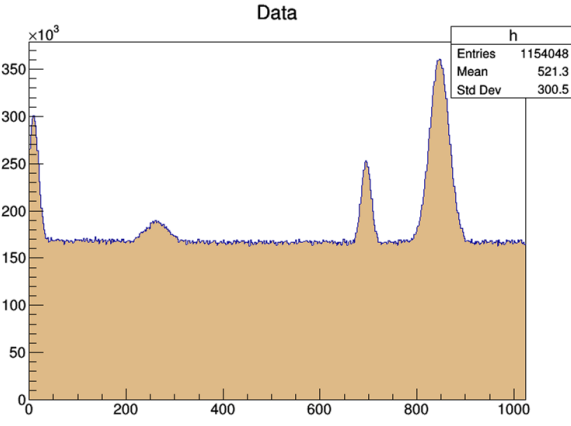
Data



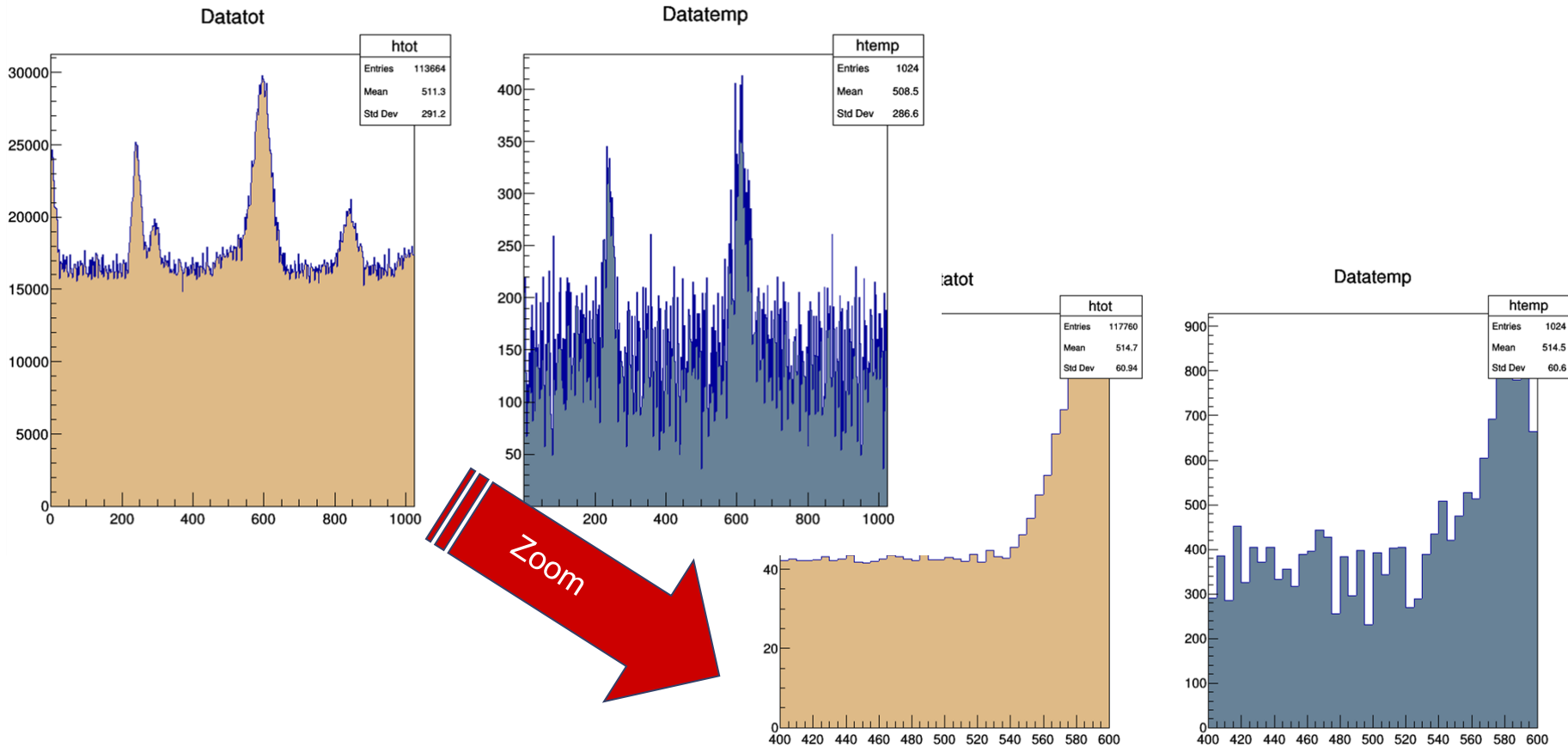
Data



Data



**Step 4)** Aggiungere un grafico che, accanto al cumulativo dei dati, mostra i dati dell'ultimo ciclo di lettura:



**Parte quarta del progetto:** Valutare le prestazioni raggiunte e fare un'analisi dei tempi di ricezione dati variando i baud sotto varie condizioni:

**1) Dati semplici:** il codice caricato sugli arduini fa modo che gli arduini inviino dati su cui non viene svolta alcuna operazione matematica. I dati cioè inviati da ogni canale sono fissi;

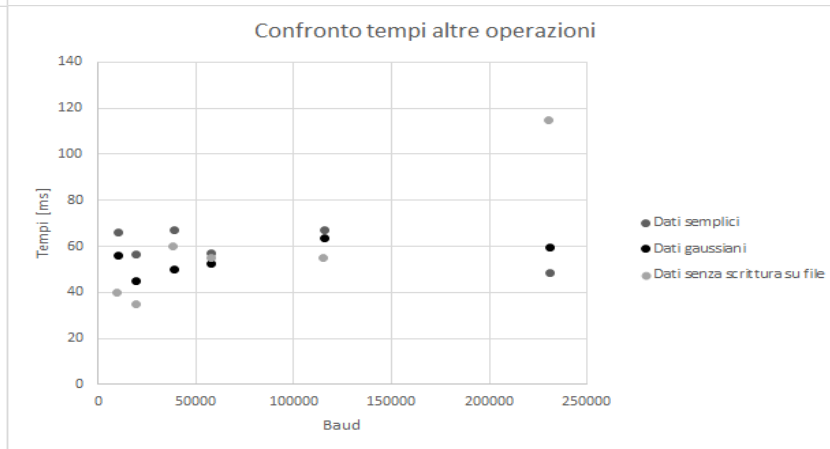
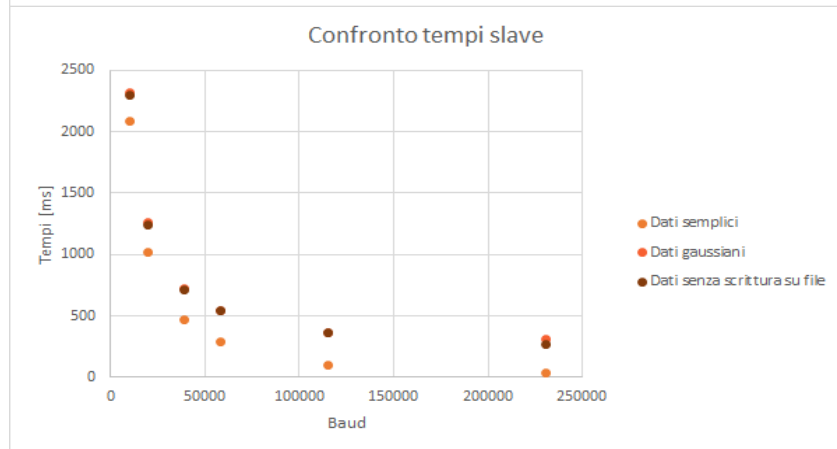
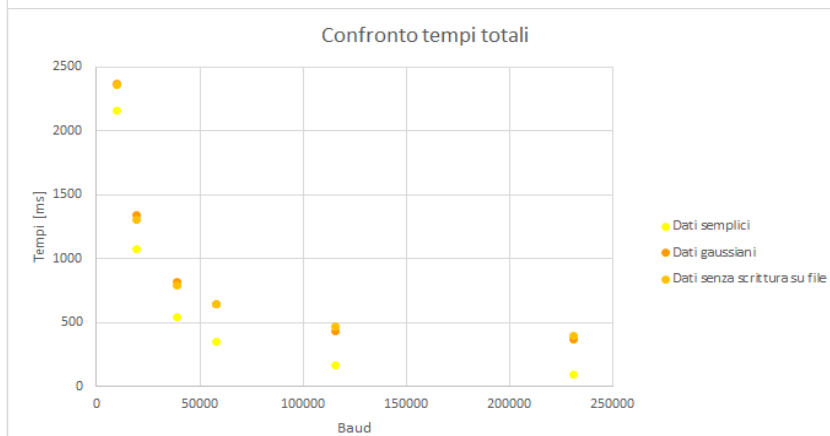
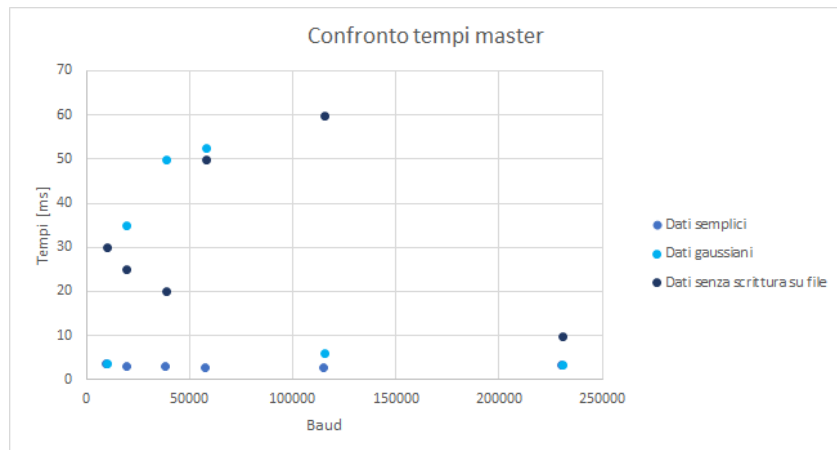
**2) Dati gaussiani:** il codice caricato sugli arduini fa modo che gli arduini inviino i dati osservati negli istogrammi precedenti: due gaussiane variabili con un background costante.

**3) Senza scrittura su file:** il codice caricato sugli arduini é quello del caso precedente. Questa volta si varia il codice lato pc eliminando un passaggio dalla raccolta dati: la scrittura su file.

Il tempo misurato é quello  
inerente a un singolo ciclo  
di raccolta, scrittura e draw  
dei dati.



# Confronto rapido dei risultati

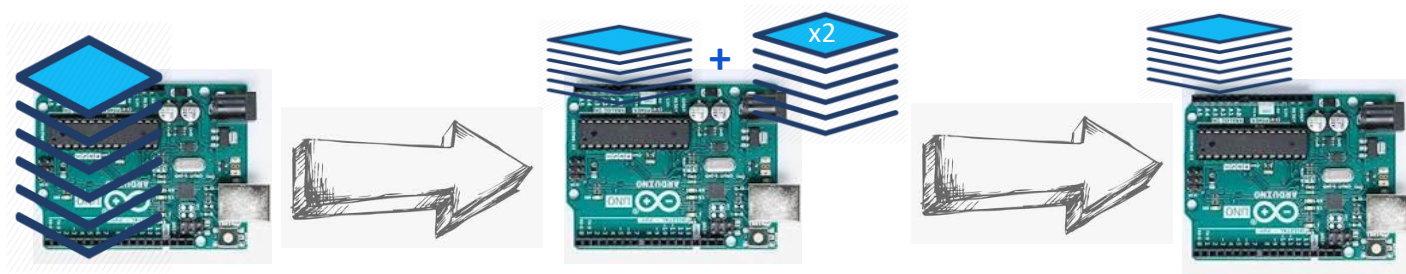




## Perché i dati semplici danno tempistiche così differenti rispetto agli altri?



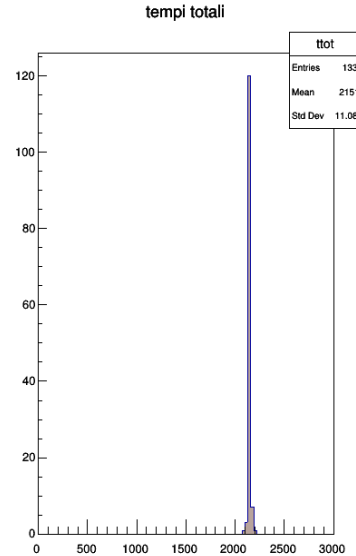
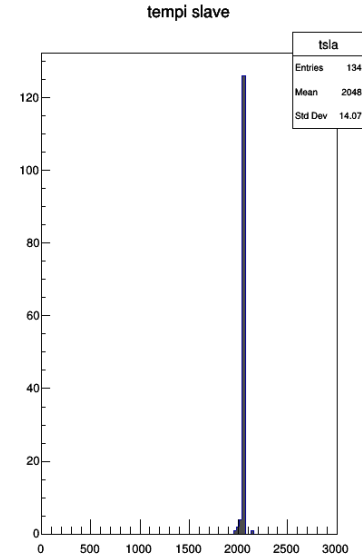
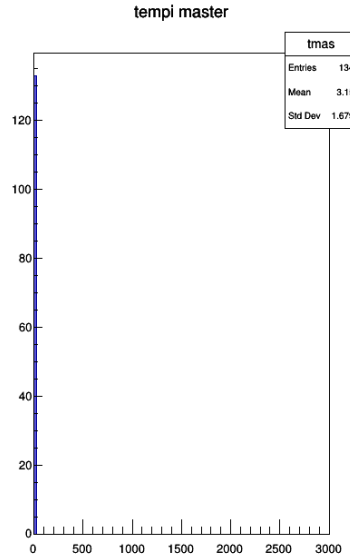
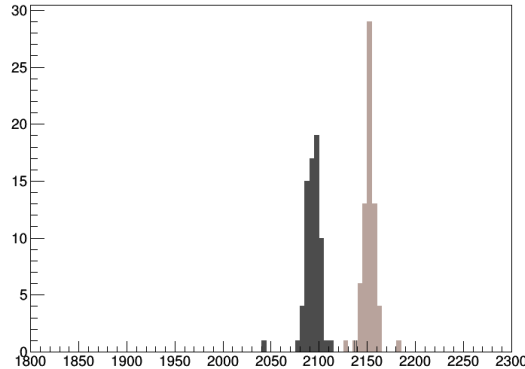
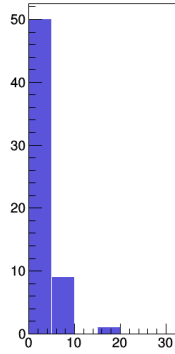
La differenza nel codice tra i dati gaussiani e quelli semplici in realtà é anche un'altra: per l'invio dei dati semplici é stato rimosso il buffer di dati da 256 words ed é stata inviata sempre lo stesso `uint32_t`, la scrittura veniva dunque fatta word per word e non una volta riempito il buffer





## Se inviassi dati gaussiani ma togliendo il buffer dati e alternando la generazione-scrittura ogni 32 bit?

Si osserva che questa soluzione porta a tempistiche simili a quelle ottenute per i dati semplice, l'aumento delle performance dunque non risiede nella complessità dei dati ma nella gestione dei cicli di creazione-scrittura.



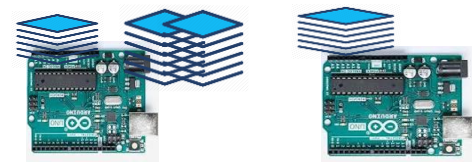
Ultima parte del progetto: Formalizzare meglio l'analisi dei tempi di ricezione dati variando i baud.



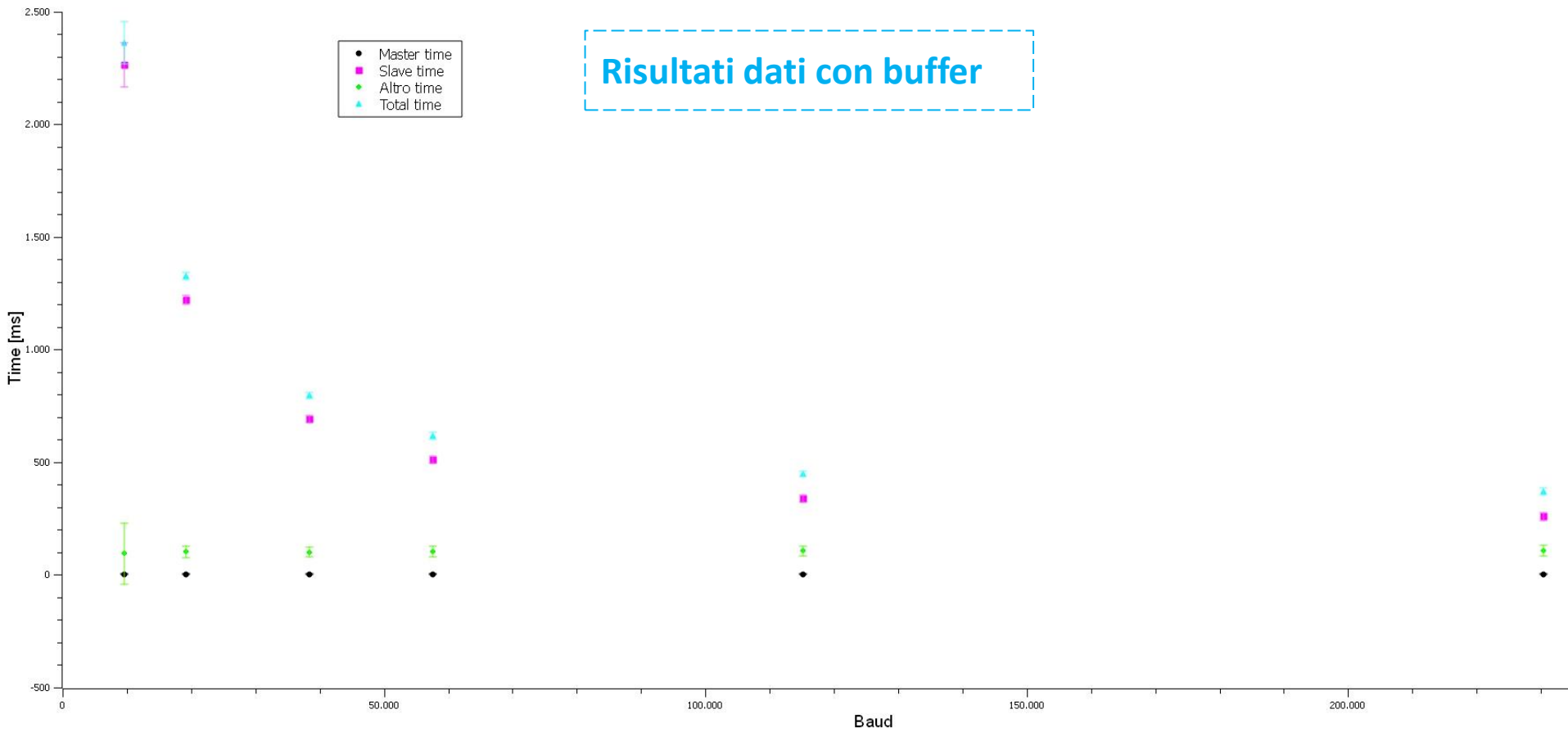
Il codice per la misura dei tempi delle varie operazioni é stato modificato in modo da fornire correttamente la media e la deviazione standard delle misure. Per far ciò gli istogrammi sono stati divisi in tre grafici separate

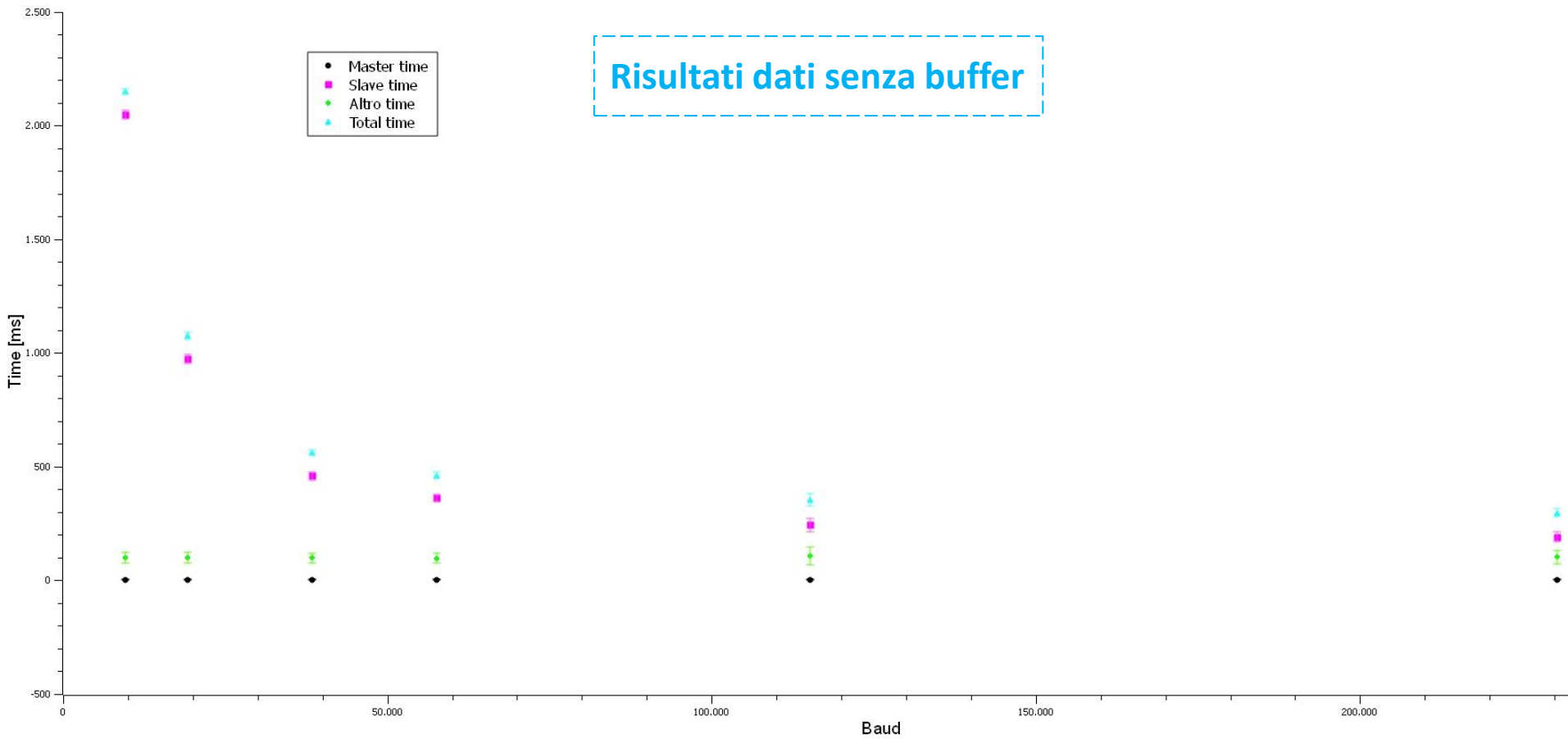


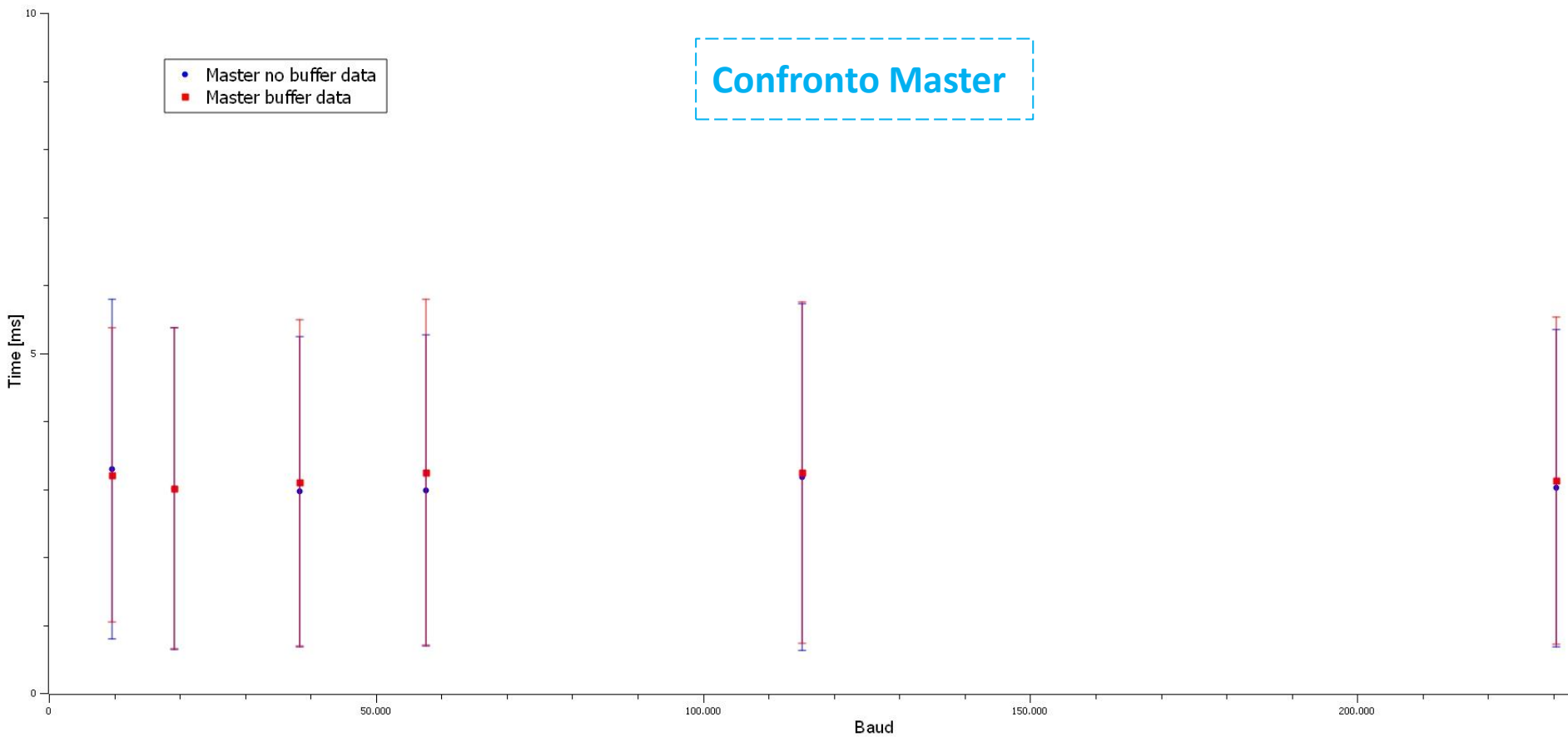
Sono state effettuate misurazioni su quattro run di codice per ogni baud: due caricando sugli arduini il codice con presente il buffer dati e due caricando il codice senza buffer dati. Non é stata però cambiata la natura dei dati

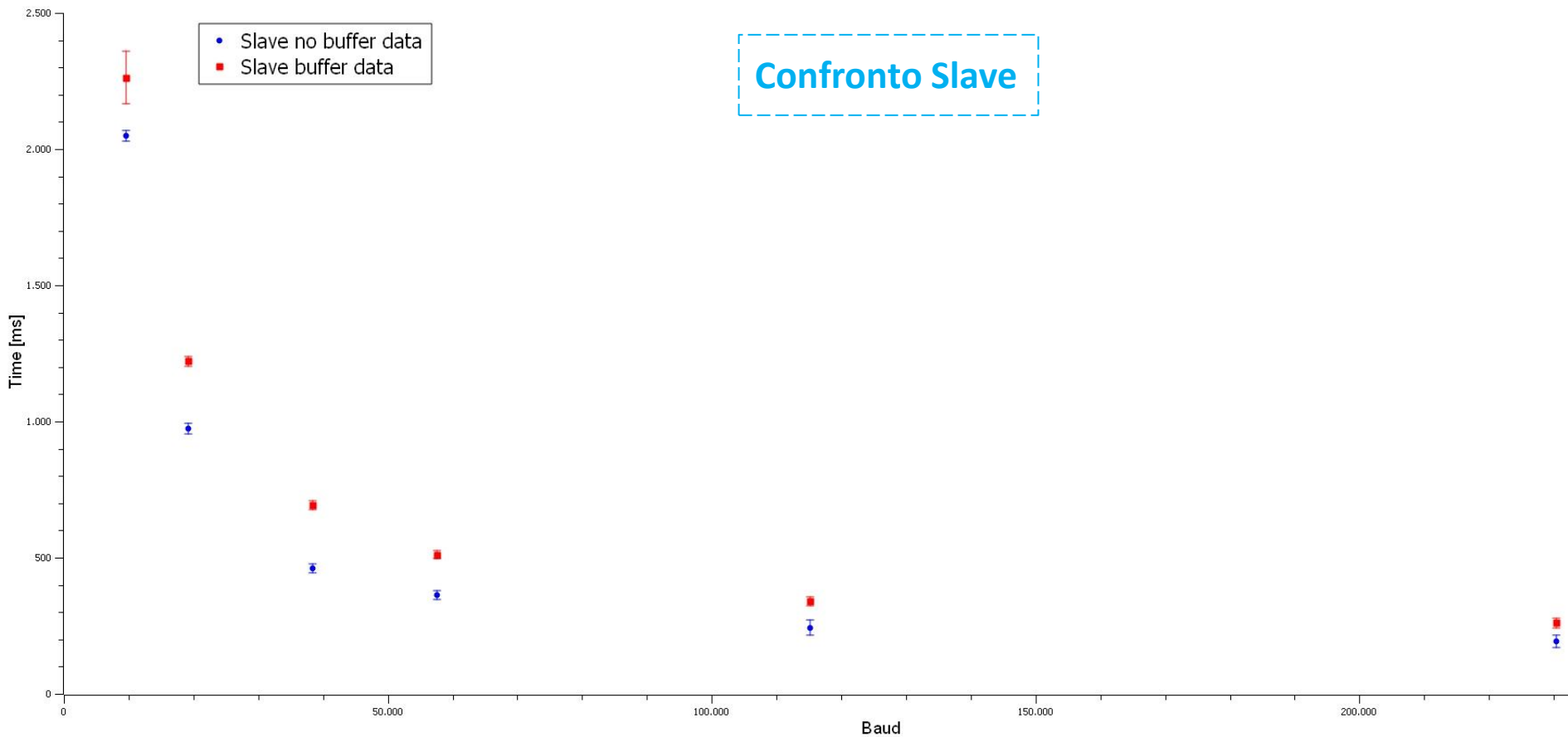


Dall'analisi rapida sembrava che togliere o inserire la scrittura su file nel programma su pc non cambiasse le tempistiche in modo rilevante o notevole, quindi in questa ultima analisi é omessa questa casistica



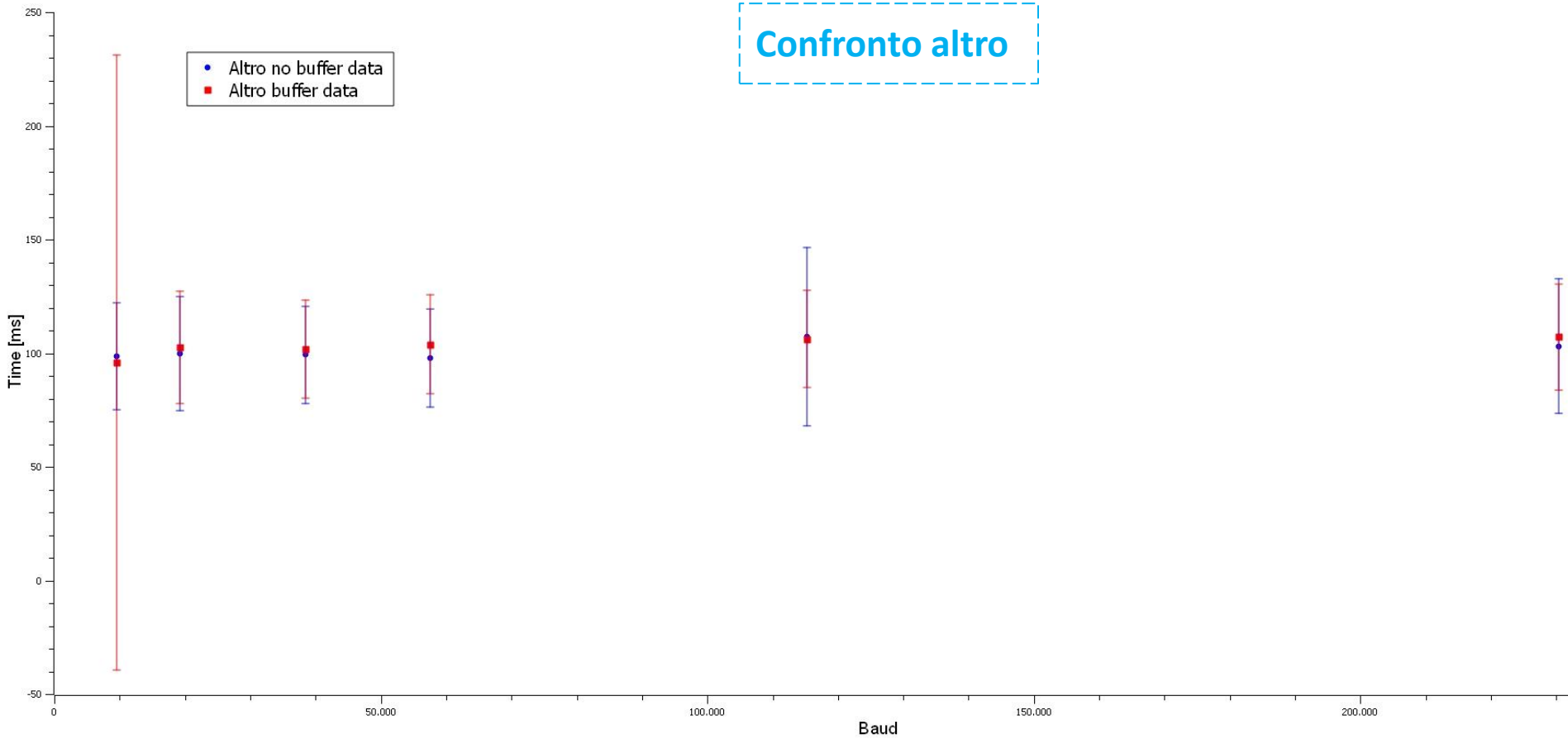


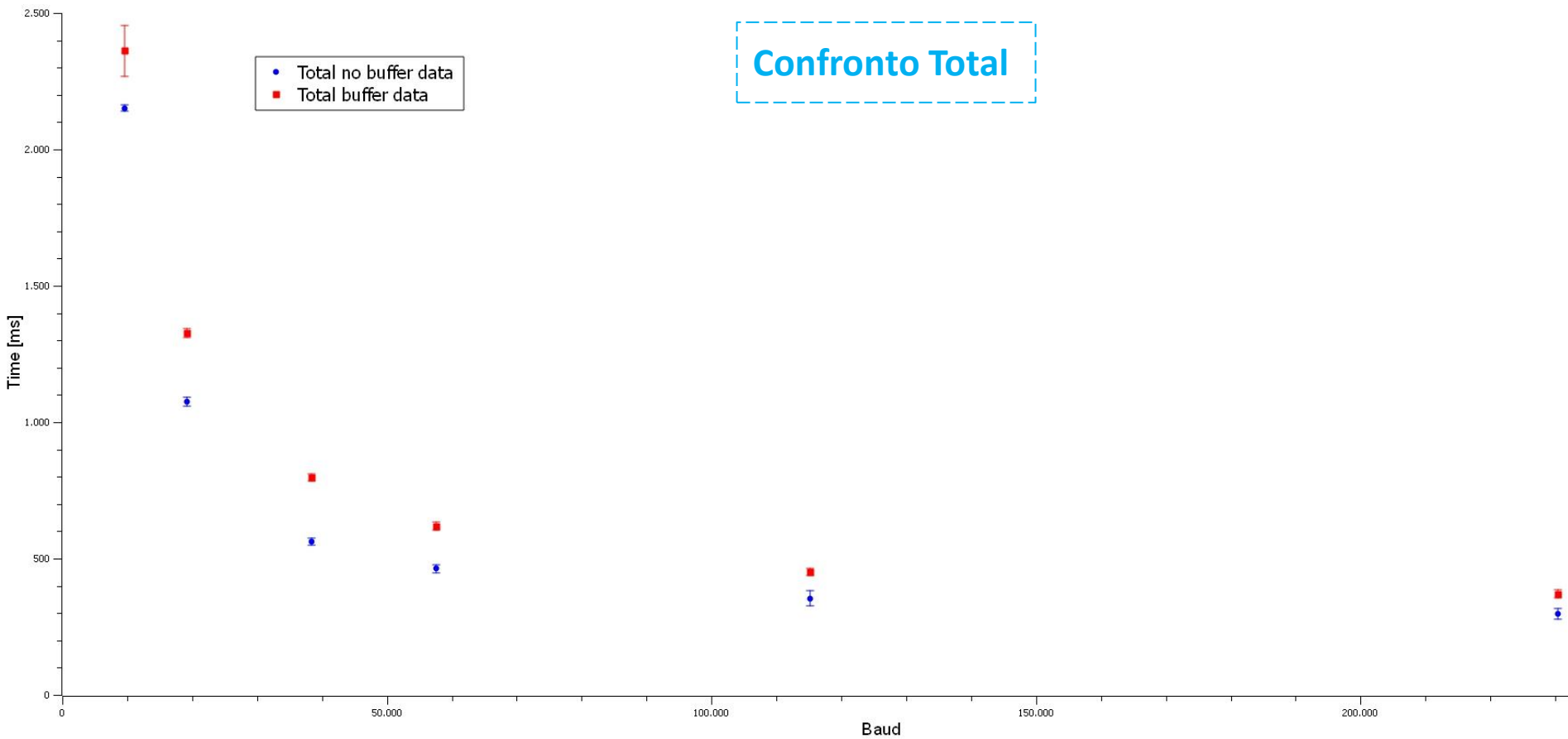






## Confronto altro





### Obiettivi futuri:

- Capire come, se e dove possibile, rendere multithread il programma;
- Introdurre un buffer di lettura di dimensione diversa da una word per volta;
- Testare con numeri diversi di canali l'istogramma e vedere se emergono ulteriori problematiche
- Trovare una soluzione efficiente al problema di trigger degli arduini
- Miglioramento dei grafici rimuovendo in tempo reale il background (?)