Università degli Studi di Padova

Dispensa sulle comunicazioni seriali asincrone

per il Corso di Sistemi Operativi

prof. Claudio Enrico Palazzi

(a cura di Oreste Venier)

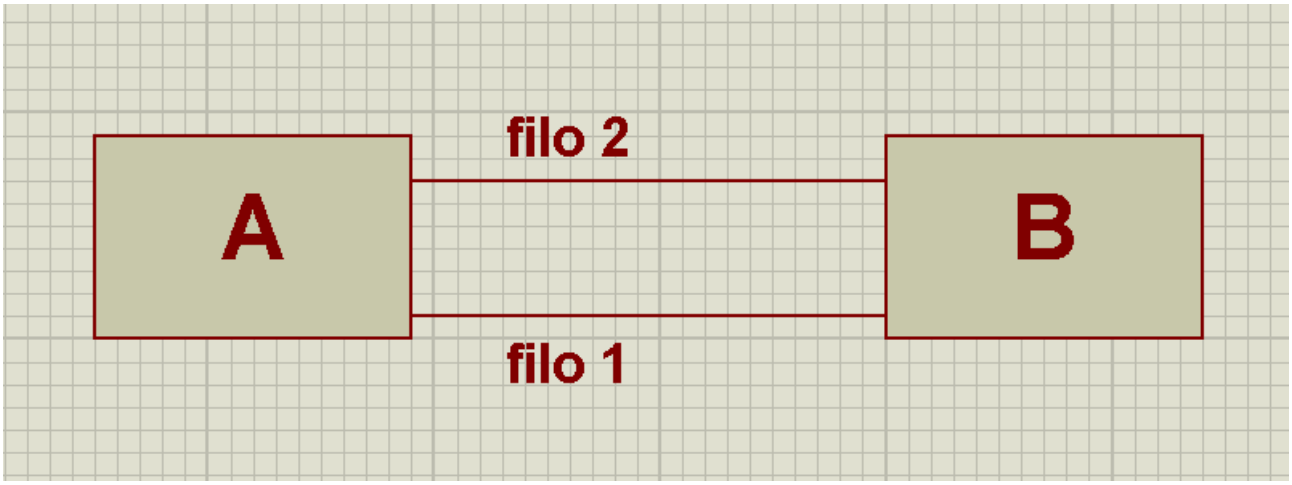
La presente breve dispensa descrive le nozioni di base delle comunicazioni seriali UART.

UART è acronimo di Universal Asynchronous Receiver-Transmitter e si riferisce ad un modulo elettronico in grado di comunicare in modo asincrono con un altro modulo sia trasmettendo che ricevendo dei dati.

Nel 2010, dove le comunicazioni fra dispositivi utilizzano sistemi molto più complessi quali i vari Ethernet, Sata, USB, ecc..., è quanto mai necessario conoscere le basi di questo argomento.

Per "asincrono", come vedremo, si intende la caratteristica di non avere alcuna sincronizzazione fra i due moduli, ovvero nessun segnale particolare che tenga i moduli sincronizzati fra loro. Questo come si vedrà verso la fine della dispensa ha delle implicazioni sia positive (semplicità) che negative (possibilità di deriva in frequenza).

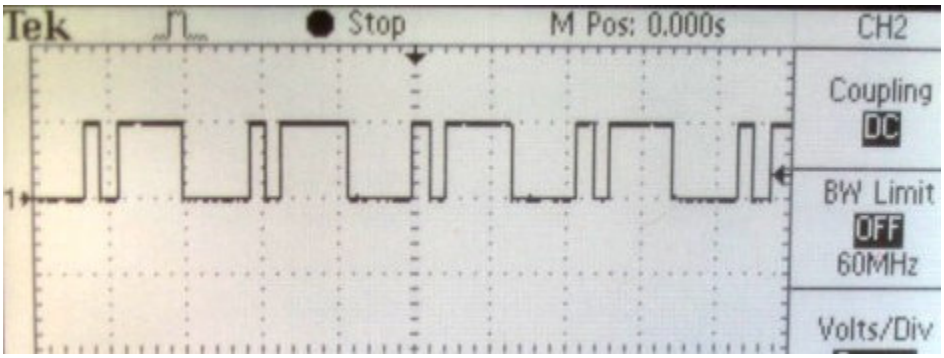
Sia definito un dispositivo A e un dispositivo B collegati fra loro semplicemente da due fili elettrici come in figura 1.



Ipotizziamo inizialmente, sempre per semplicità, che il dispositivo A abbia il ruolo di "trasmettitore" (ovvero abbia dei dati da trasmettere) verso B che farà invece da "ricevitore". Questi dati sono per esempio il frutto di un'elaborazione di A, oppure la raccolta dall'esterno di informazioni da sensori, ecc... in pratica qualunque informazione digitale (si veda l'approfondimento sui convertitori analogico-digitali che è stato posto a fine dispensa per non interrompere il discorso principale).

Uno di questi fili (filo 1) fa da riferimento per l'altro (normalmente viene chiamato "massa"), mentre l'altro (filo 2) viene posto (dal dispositivo A) ad una tensione, ovvero ad un voltaggio, che può avere due valori: 0 oppure una certa tensione predeterminata, che dipende da come è fatto il circuito stesso. Per esempio, 3.3 Volt.

Pertanto idealmente i valori di tensione che possiamo trovare sul filo 2 (rispetto al filo 1 che fa da riferimento) sono o 0 o (p.es.) 3.3V, e questi due valori sono assimilabili, per quanto segue, a due valori logici 0 e 1, come in foto:



La comunicazione seriale, come è facile immaginare, consiste nell'alternare i valori di tensione sul filo nr. 2 da parte del dispositivo A (trasmettitore), in modo che il dispositivo B (ricevitore) li percepisca, e seguendo i valori e le loro alternanze sia in grado di decodificare le informazioni trasmesse.

Ovviamente questa alternanza viene effettuata secondo una tempistica precisa, che dipende da un cosiddetto clock o orologio del circuito: sia A che B integrano un clock che *scandisce il tempo* alla comunicazione, ed in pratica la durata di ogni singolo "bit" è pari al reciproco di questo clock.

Questo clock nelle UART si esprime come "baudrate". Ogni "tick" o "scatto" del clock permette al circuito trasmittente di spedire sul filo nr. 2 il bit successivo; per fare un esempio, nel caso di un baudrate di 100.000 baud, ogni "scatto" avviene ogni 1 / 100.000 secondi = 10^{-5} secondi = 10 microsecondi (milionesimi di secondo).

I valori standard di baudrate sono 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 76800, 115200, 230400, 460800 e 921600; essi sono in parte multipli e sottomultipli in quanto ricavati originariamente da un quarzo tagliato ad una frequenza particolare, 14.745.600 cicli/secondo (Hz).

Si faccia attenzione, come sarà evidente poco oltre, al fatto che i bit che si intendono trasmettere sono accompagnati da altri bit di protocollo, pertanto il "rate" di trasmissione reale delle informazioni (in bit/s) nelle UART è inferiore al "baudrate", in quanto va considerato anche l' *overhead* per la comunicazione: dal momento infatti che i due dispositivi non hanno alcun altro modo di dialogare fra loro, è necessario che i dati veri e propri (che sono facilmente identificabili come stringhe di bit) siano "incastonati" attraverso un protocollo.

Il protocollo della UART è fra i più semplici ed è costituito dalle seguenti fasi:
1) si definisce una velocità di trasmissione / ricezione dei dati (baudrate);
2) si spezza la stringa di bit da trasmettere in stringhe molto corte, di lunghezza p.es. 8 bit (sono possibili altre lunghezze) e si creano vari pacchetti;
3) si fascia ogni pacchetto con un ulteriore bit iniziale, detto bit di start, e con uno finale, detto bit di stop (è possibile averne più di uno, si veda in seguito perché);
4) opzionalmente, si aggiunge (dopo i dati) all'interno di ogni pacchetto un "bit di parità" che ha lo scopo di fornire un controllo di integrità del pacchetto pervenuto.

Le scelte di cui sopra: velocità, lunghezza della stringa di bit, numero di bit di stop, presenza o assenza del bit di parità sono informazioni che A e B devono aver condiviso preliminarmente (in fase di progettazione o di configurazione), in quanto la decodifica si basa su questi parametri (in alcuni casi è possibile la deduzione automatica del baudrate da parte del ricevitore, campionando ad una velocità maggiore in una fase preliminare i dati ricevuti; in questo caso di solito il trasmettitore invia per un certo numero di volte un byte ricco di alternanze 0-1). Questi parametri saranno pertanto presenti anche negli esercizi (oppure può essere chiesto di dedurli da altri dati forniti) in quanto necessari alla risoluzione del problema.

Iniziamo a vedere in dettaglio ciascuno di questi punti: per semplicità utilizziamo i seguenti parametri di comunicazione:
1) 1 bit di start (che c'è sempre ed è uno solo);
2) 8 "bit di dati" (ovvero vengono inviati per ogni pacchetto 8 bit di dati "utili", più ovviamente quelli del protocollo);
3) un solo bit di stop (nella stragrande maggioranza si usa 1 bit di stop, ma è utile sapere che in alcuni casi si sceglie 2, per esempio per dare più tempo al ricevente di elaborare/registrare i dati ricevuti);
4) nessuna parità, ovvero il bit di parità è assente.

Ogni "pacchetto" sarà lungo pertanto: 1 + 8 + 1 = 10 bit, e porta 8 bit di informazione.

Supponendo che il baudrate sia di 9600 baud, ciò significa che:

- ogni "bit" ha lunghezza 1/9600 = 104,2 microsecondi (attenzione a non fare errori con gli ordini di grandezza!!);
- ogni pacchetto (di 10 bit) ha lunghezza 1,042 millisecondi;
- supponendo che ogni pacchetto sia immediatamente consecutivo al precedente, ovvero senza pause fra pacchetti, le informazioni trasmesse ogni secondo in termini di bit utili (=i bit di informazione che vogliamo trasmettere da A a B) sono: 1 / (1.042 millisecondi) = 960 byte, e non: 9600 / 8 = 1200 byte: infatti il protocollo determina un'overhead che "appesantisce" la comunicazione. Si faccia attenzione a questo fatto per evitare di stimare

erroneamente il rate di scambio delle informazioni.

Nell'analisi della comunicazione partiamo dal bit di start. Il bit di start permette al ricevitore (B) di "capire" quando iniziano ad arrivare dei dati, rispetto ad una situazione "di riposo" (*idle*) della linea. Nella semplicità della linea che stiamo considerando, questa "situazione di riposo" si attua mantenendo la linea ad un livello costante. Per default questo livello è "1" ovvero la linea viene mantenuta al livello costante di xx Volt (p.es. 3.3 Volt, dipende dalla tecnologia).
Il bit di start non è altro che un primo bit a 0: è un bit che fa da "contrasto" al valore costante "1" della linea.

In pratica, per iniziare la trasmissione A pone a 0 la linea per un tempo pari al reciproco del baudrate, mentre B si accorge di ciò in quanto sta continuamente osservando la linea proprio per vedere quando il voltaggio sulla stessa passa da 3.3 Volt a "0".

A questo punto, il trasmettitore A può inviare, uno ad uno, ad una cadenza dettata dal baudrate, gli 8 bit della parola; questo significa che a cadenza regolare imposta sul filo nr. 2 il valore del bit corrispondente, e, mantenendo tale valore sulla linea, attende un tempo come sopra calcolato di 104,2 microsecondi.

Terminati gli 8 bit, il trasmettitore imposta il "bit di stop": è semplicemente un bit fisso, aggiuntivo, e rovesciato rispetto al bit di start, ovvero nel nostro caso posto a "1".

Perchè deve essere rovesciato rispetto al bit di start? Questo sarà evidente fra poco dalla descrizione del ricevitore, ma si può già immaginare che la spiegazione risiede nel semplice fatto che nell'ipotesi di trasmissione continua, ovvero di un pacchetto nuovo subito dopo il precedente, il bit di start fa da "contrasto" con lo stato precedente (pertanto la linea a "1", dato che il bit di start è uno "0"), in modo da assolvere alla sua funzione.

Vediamo ora il funzionamento del ricevitore B.

B vede solo la linea costituita dal filo 1 e filo 2, e misura continuamente il valore della tensione del filo 2 rispetto al filo 1 (che fa da riferimento).

Per quanto detto sopra, in condizioni di attesa continua a misurare una tensione presente, pertanto un livello logico "1".

Ad un certo punto B vede cambiare lo stato della linea da 1 a 0: cosa significa? Significa che stanno arrivando dei dati, e il cambiamento è dovuto proprio al bit di start.

Va detto che la misurazione dello stato della linea avviene a velocità molto più elevata del periodo di ogni bit, pertanto in pratica B si accorge quasi immediatamente (rispetto alla durata di un bit) del cambio di stato. Che cosa fa a questo punto B? B è interessato a capire lo stato della linea nei vari punti "centrali" dei bit trasmessi, pertanto:
1) B attende "mezzo bit" (siamo ancora all'interno del bit di start);
2) per otto volte, B attende la durata nominale di un bit e rileva lo stato della linea (pertanto al "centro" di ogni bit: viene scelto il centro per evitare errori di misura proprio nei "salti" di stato fra un bit e l'altro: ovvero si fa la misura quando la linea è più stabile);
3) ogni rilevazione corrisponde ad uno dei bit di dati utili trasmessi;
4) terminato l'ottavo bit ... il ricevitore B attende un altro bit e si rimette in attesa di un cambio di stato: infatti siamo al centro del bit di stop, e quindi in condizioni similari alla "linea in attesa";
5) in alcuni casi si preferisce mettere un secondo bit di stop, pertanto ritardando l'invio del pacchetto successivo, in modo da dare un po' di "respiro" al ricevitore per effettuare alcune operazioni sui dati ricevuti; p.es. per registrarli.

Il ciclo di cui sopra continua senza sosta.

Vediamo ora quali possono essere alcuni problemi legati a questa implementazione.

Innanzitutto, come dice la parola stessa, la modalità di ricetrasmissione dei dati è "asincrona" in quanto i due dispositivi non condividono un'unica fonte di riferimento per le temporizzazioni, ma 1) devono mettersi d'accordo prima su che clock (baudrate) usare; 2) l'intero processo si basa di fatto su due clock che possono risultare all'atto pratico leggermente diversi.

Cosa succede nel caso in cui i clock del ricevitore sia diverso da quello del trasmettitore?

Nel caso di differenze notevoli (p.es. errata scelta del baudrate) non è possibile alcuna comunicazione, in quanto le letture del ricevitore sono completamente scoordinate (su un terminale compaiono solo caratteri “spazzatura”), ma nel caso di differenze limitate ci possono essere dei problemi soprattutto quando i dati vengono inviati senza interruzione fra i pacchetti.

Perchè i clock, pur nominalmente uguali, potrebbero essere diversi? Per motivi fisici, legati alla generazione e alla stabilità dei clock stessi. I clock sono quasi sempre derivati da oscillatori al quarzo (anche quello all'interno dei vostri orologi da polso), ovvero sottili lamine di quarzo che sono posti in determinate condizioni elettriche per oscillare ad una frequenza che dipende dalle loro caratteristiche fisiche.

La soluzione più semplice consiste nel mettere due quarzi identici sia nel trasmettitore che nel ricevitore, ma in molte situazioni ciò non è sufficiente.

Per prima cosa c'è una tolleranza nella produzione dei quarzi, pertanto quarzi nominalmente "uguali" oscillano con frequenza leggermente diversa fra loro. Ma questa differenza, di poche parti per milione, è ininfluente nell'ambito delle UART; la differenza maggiore dipende da altri fattori: la frequenza del clock generato da un quarzo dipende, fra le altre cose, anche dalla temperatura alla quale viene mantenuto. Se A e B hanno due quarzi nominalmente uguali, ma A e B vengono mantenuti a temperature diverse, i rispettivi clock avranno delle differenze.

Questo è un caso molto comune per gli apparecchi esposti all'esterno: si consideri il seguente caso applicativo.

Un pannello informativo a LED mostra ai passanti il numero di posti liberi in un garage a pagamento. Il pannello è posto all'esterno sotto il sole cocente d'estate e ad una temperatura molto rigida d'inverno. Il pannello (dotato di elettronica interna) è collegato tramite un collegamento seriale asincrono ad una centralina elettronica posta all'interno del garage, che è a temperatura controllata (temperatura ambiente) in quanto all'interno di un ufficio. La centralina è il trasmettitore A ed invia le informazioni (posti liberi) al ricevitore B posto sul pannello.

Anche se dispongono dello stesso quarzo nominale, il clock del pannello sarà leggermente diverso da quello della centralina, tanto maggiore sarà la differenza di temperatura.
Ritornando ad uno dei primi paragrafi della dispensa, si nota così che il vantaggio principale della comunicazione UART è la semplicità (sia costruttiva che implementativa), mentre uno degli svantaggi è l'assenza di una sincronizzazione che porta a non tollerare gli effetti della possibile deriva in frequenza e la conseguente limitazione di velocità, in quanto al crescere della velocità si accorciano i tempi di ciascun bit e si acuiscono pertanto gli effetti dei clock differenti o poco stabili.

Questi problemi nelle comunicazioni sono oggi brillantemente risolte da varie tecniche elettroniche, nondimeno riteniamo che costituiscano la base di quella conoscenza tecnico-fisica che un informatico nel 2010 deve avere per affrontare con più serenità la comprensione delle tecnologie più moderne.

Segue ora un approfondimento sui convertitori analogico-digitali e due quesiti risolti, che saranno esposti anche nel corso delle lezioni.

Approfondimento: dati digitali dal mondo reale: i convertitori analogico-digitali

Il mondo reale offre molte grandezze fisiche che si prestano ad interessanti elaborazioni digitali. Esse sono in origine *analogiche* e pertanto per riferirsi ad esse si utilizza una rappresentazione numerica, normalmente sotto forma di un numero reale (p.es. la temperatura, la velocità del vento ecc...).

Per gestire queste grandezze all'interno di un elaboratore elettronico, in grado di trattare (solo) con dati digitali, è necessario convertirle dal mondo analogico, rappresentato (per lo più) da numeri reali e continui nel tempo, al mondo digitale, rappresentato invece dai bit e in modo non continuo nel tempo.

Ciò può essere fatto da speciali dispositivi elettronici che prendono il nome di **convertitori analogico-digitali**.

Per prima cosa è necessario “presentare” al dispositivo la grandezza fisica (che si intende acquisire) sotto forma di una tensione elettrica; ciò si effettua tramite un cosiddetto trasduttore, che è un termine più generale per indicare un sensore o misuratore: per esempio, la temperatura di un corpo può venire convertita in una tensione elettrica da parte di un termometro, la velocità del vento tramite una dinamo collegata alle pale di un anemometro, ecc...

Questa tensione ovviamente spazierà all'interno di un range prevedibile a seconda della grandezza da misurare e del trasduttore utilizzato. Il convertitore analogico digitale deve perciò poter gestire l'intero range di interesse, e viene calibrato fra un minimo e un massimo valore di tensione ammissibile, tensioni queste ultime che prendono il nome di “tensioni di riferimento” (in quanto vengono materialmente fornite al convertitore come riferimento). Normalmente per semplicità la tensione inferiore è il riferimento zero o di massa.

Il convertitore analogico-digitale provvederà pertanto su base regolare, ogni intervallo di tempo definito, a convertire la tensione in ingresso in un numero intero in modo lineare fra le due tensioni di riferimento. La tensione in ingresso viene convertita in un numero ad n bit dove “n” è la precisione del dispositivo convertitore. I più semplici dispositivi esprimono 8 o 10 bit (pertanto un valore numerico da 0 a 2^8=256 oppure 2^10 = 1024), i più complessi anche 24 bit.

Facciamo un esempio pratico di calcolo ed applicazione di quando sopra.

Ipotizziamo di avere un convertitore analogico-digitale a 10 bit che lavora fra 0 e 5 Volt.

Ipotizziamo di avere un anemometro per misurare la velocità del vento, collegato ad una dinamo che fornisce le seguenti tensioni:
0 Volt, quando non c'è vento e le pale dell'anemometro sono ferme;
3 Volt, con un vento a 50 Km/h;
linearmente nel range [0-3] Volt. Oltre i 50 Km/h, … è meglio non provare :)

Collegando la dinamo al convertitore analogico-digitale possiamo convertire il valore della velocità del vento in un numero atto ad essere ricevuto e gestito da un elaboratore elettronico.

In particolare, il convertitore può essere collegato per mezzo di un'interfaccia seriale ad un PC e trasferire così i dati (valori di velocità acquisiti) ad un nostro programma per visualizzarli, registrarli ecc...

La velocità con cui il convertitore effettua il proprio lavoro prende il nome di frequenza di campionamento: essa dipende dal dispositivo e può andare da poche conversioni al secondo (per i dispositivi più precisi, per esempio all'interno delle bilance elettroniche) fino a molte migliaia di conversioni al secondo (anche milioni).

Ovviamente, maggiore è la velocità di campionamento e maggiormente è necessario disporre a valle di un'interfaccia veloce con cui trasferire i dati all'esterno.

Tornando al nostro esempio dell'anemometro dobbiamo calcolare i valori in uscita dal convertitore.

In caso di assenza di vento, la dinamo presenta 0 Volt e il numero in uscita è 0 (o meglio, sarà senz'altro presente una piccola fluttuazione di qualche unità, a causa del “rumore” elettrico).

In caso di vento a “v” km/h, la tensione (Vout) generata dalla dinamo sarà (calcolo lineare):

Vout = 3 * (v/50); per 0<=v<=50;

Il nostro convertitore converte una tensione di 0 Volt nel numero 0, ed una tensione di 5 Volt nel numero 2^10-1 = 1023 (non 1024 perché 1024 è il totale degli intervalli digitali), pertanto per una tensione Vout qualsiasi:

n = (Vout / 5) * 1023;

sostituendo l'espressione di cui sopra troviamo le relazioni fra n e v:

n (uscita del convertitore) = (Vout / 5) * 1023 = (3 * v / (50 * 5) * 1023 = 12,276 * v (in km/h).

e viceversa: velocità del vento (in km/h) = n / 12,276.
Per esempio, nel caso di un vento a 10 km/h, il convertitore presenterà (idealmente) in uscita il numero 122; a causa delle piccole inevitabili fluttuazioni del segnale elettrico corrispondente ad ogni grandezza analogica ci sarà una “nuvola” di valori di qualche unità attorno a tale valore.

Ma ogni quanto tempo è possibile / necessario effettuare la conversione? Dipende dal dispositivo e dal problema. Nel nostro caso, per apprezzare brusche variazioni di velocità delle pale è necessario effettuare la conversione anche molte volte al secondo, p.es. 15 o 20. Nel caso di misure di temperatura ambientale può essere invece sufficiente un tempo più lungo fra una misura e l'altra; dipende caso per caso. E' anche possibile effettuare, ogni tot tempo, più conversioni ravvicinate, per recuperare maggiori informazioni / più pulite / dalla conversione. Ovviamente, più precisione ha il convertitore e più alta è la sua frequenza di campionamento, più veloce dovrà essere la linea di comunicazione che trasferisce i dati dal convertitore al PC o al circuito che si occupa dell'elaborazione.

Quesito.

Un sensore elettronico per il controllo di celle frigorifere per alimenti rileva continuamente la temperatura dell'oggetto a cui è applicato e grazie ad un circuito elettronico interno invia tale valore, sotto forma di un byte di dati, ogni secondo ad un altro dispositivo.

Nelle ipotesi che:
1) il valore prodotto possa assumere tutto il range di un byte senza segno;
2) il range di funzionamento del sensore sia [-10 °C, +15.6 °C[
3) all'interno di tale range la risposta del sensore sia lineare;
calcolare la risoluzione in temperatura del sensore.
Risposta: 1 byte = 256 valori possibili, (15.6-(-10.0))/256 = 0.1 °C (notare il verso delle parentesi quadre: il valore 15.6 non è compreso nell'intervallo).

Considerando che in una farm di frigoriferi siano presenti 50 apparecchi, ciascuno dei quali è dotato di 12 sensori per monitorare la temperatura in varie parti, e considerando che tutti questi sensori siano collegati ad un dispositivo che bufferizza i dati e che li invia per l'elaborazione ad un computer esterno sotto forma di un unico flusso seriale UART, calcolare qual è la velocità minima della linea seriale necessaria a questo trasferimento, scelta fra quelle standard 1200, 2400, 4800, 9600, 19200 baud. Si consideri una linea seriale con protocollo 8N1 (8 bit di dati, nessuna parità ed un solo bit di stop).

Risposta:
numero totale di sensori: 50 x 12 = 600.
ogni secondo producono 600 byte di dati
ogni byte di dati per essere trasmesso ha bisogno di 10 bit: un bit di start, 8 bit di dati e un bit di stop. Attenzione, il bit di start c'è sempre! Non fatevi trarre in inganno dalla descrizione “8N1”: la presenza del bit di start è sottintesa).
Nel protocollo seriale asincrono UART grazie alla presenza dei bit di start e di stop i dati possono essere accodati con continuità (anche se ciò non è consigliato per lunghi stream. Perché?).
Pertanto in linea teorica i sensori producono 6000 bit da trasmettere e la linea più lenta fra quelle standard è pertanto la 9600 baud.

Considerando la scelta che il candidato ha fatto al punto precedente, calcolare il tempo di un singolo bit trasmesso e considerando pertanto che ogni secondo il sistema invierà al PC un burst contenente i dati di tutti i sensori, calcolare qual è il "gap" o intervallo fra la fine di un burst e l'inizio dell'altro.
Risposta: a 9600 baud ogni bit occupa un tempo pari a 1 / 9600 = 1.000.000 microsecondi / 9600

= 104,2 microsecondi. 6000 bit occupano 625.000 microsecondi ovvero 0.625 secondi. Il gap presente prima della trasmissione successiva è pertanto 1-0.625 = 0.375 secondi.

Quesito.

Una telecamera miniaturizzata utilizzata per applicazioni meteo in alta montagna è collegata ad un dispositivo (PC portatile) che funge da gateway per trasmettere a valle le immagini raccolte.
Tale telecamera funziona in due modalità: JPEG e RAW, e possiede un sensore con risoluzione 160x120 pixel RGB, 24 bit per pixel (ogni pixel è descritto da 3 byte).
Nella modalità RAW la telecamera trasmette i valori di ogni singolo pixel dell'immagine, codificati come sopra descritto (=3 byte necessari per ogni pixel).
Nella modalità JPEG la telecamera comprime i dati (v. sotto) e trasmette pertanto una quantità inferiore di informazioni per ogni immagine.
La telecamera funziona nel seguente modo: scatta una foto e la memorizza internamente; tale tempo è trascurabile; la trasmette al PC tramite una linea seriale a 230.400 baud in modalità 8E1; tale sequenza di operazioni avviene a ciclo infinito senza interruzioni. Si trascuri anche il tempo di compressione dell'immagine JPEG (tale tempo è ingente ma l'operazione viene svolta in modo parallelo alla trasmissione da un'unità dedicata interna alla telecamera).
Ai fini dei seguenti calcoli pertanto il candidato dovrà considerare solamente le operazioni di trasmissione seriale dei dati dalla telecamera al PC.

Calcolare il numero di immagini complete che la telecamera è in grado di trasferire al PC ogni minuto:
1) nella modalità RAW;
2) nella modalità JPEG, considerando che nel caso specifico dato il soggetto da inquadrare e la luce/contrasto presenti il rapporto di compressione fra l'immagine originale e l'immagine finale sia di 15:1;

Risposte:
per ogni immagine nella modalità RAW il numero totale di byte da trasferire lungo la linea seriale è 160 x 120 x 3 = 57600. Nella modalità 8E1 è presente anche un bit di parità (E = even, pari) e 1 bit di stop, oltre ovviamente al sempre presente bit di start. Pertanto per ogni byte i bit reali da trasferire sono 1+8+1+1 = 11 e per l'immagine 57600x11 = 633600. La linea è a 230400 baud pertanto ogni immagine in questa modalità impiega 2,75 secondi e in un minuto si riescono a trasferire 21 immagini complete.
per ogni immagine nella modalità JPEG di giorno il numero totale di bit è 57600 / 15 x 11 = 42240. In un minuto corrisponde a 327 immagini intere. Attenzione, non a 21 x 15 = 315! Attenzione a non sbagliare gli arrotondamenti!