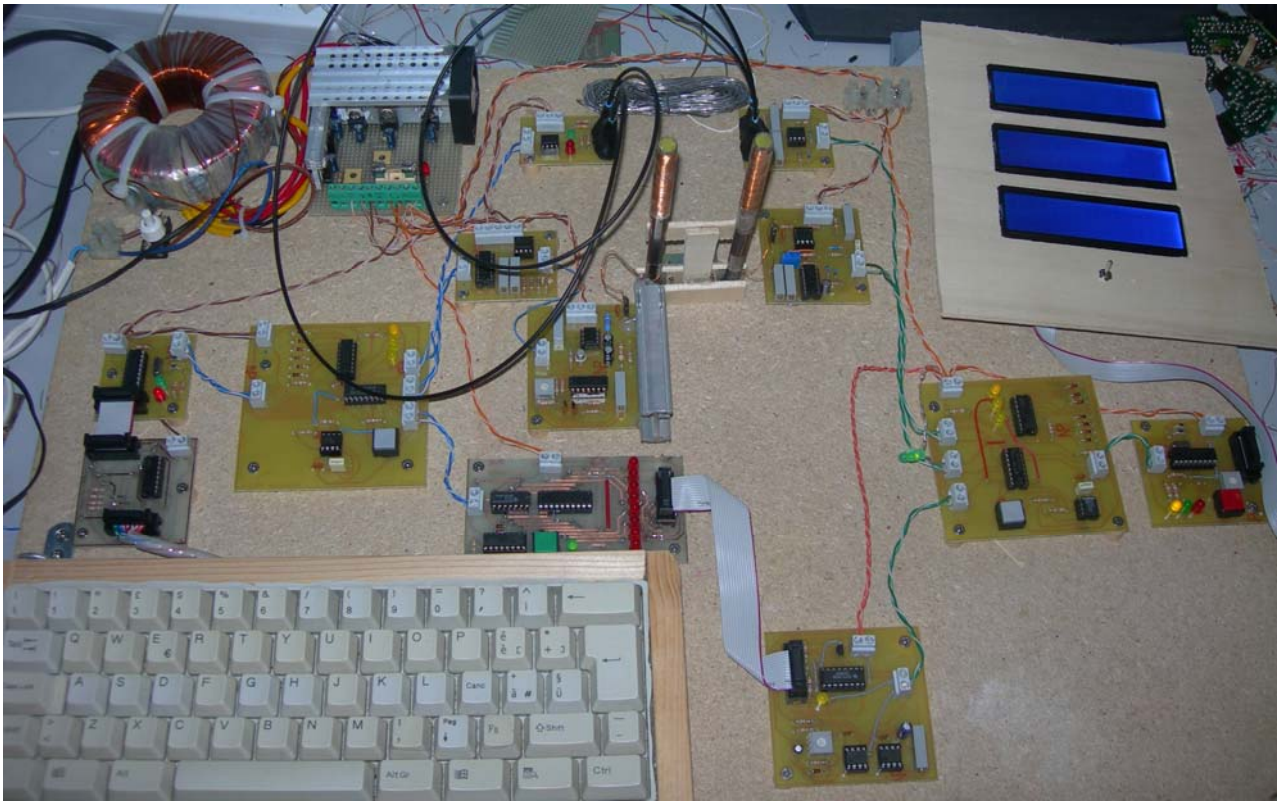
STUDIO DEI MEZZI TRASMISSIVI



Francesco Mazzetti

Tesina Esame di stato AS 2007-2008
Classe 5ªB$_4$

ITIS "Odone Belluzzi" (Bologna)

INDICE

Prefazione

1a Introduzione

Come da titolo, questo progetto si prefigge come obiettivo lo studio dei più famosi mezzi di trasmissione dati: via cavo, via radio e via fibra ottica.
Con il termine studio, viene inteso, la progettazione e realizzazione (naturalmente tramite una conoscenza a monte) di strutture capaci di rendere semplice e intuibile le tecniche di utilizzo dei suddetti mezzi trasmissivi.
Per fare ciò, come verrà approfondito successivamente, ho deciso di creare progetti completamente capaci di adattarsi l'un l'altro, quindi utilizzando standard e vincoli da me prefissati.
Più semplicemente ho cercato di mantenere un'unica e semplice codifica dati in invio e ricezione. Quello che cambia è appunto ciò che sta in mezzo: come parlare in un microfono registrando la proprio voce, che potrà essere incisa in un vinile o in una cassetta, o ancora in un cd. Quello che conta è che la propria voce torni tale e diventi riascoltabile da qualsiasi altra parte.
Nel mio caso specifico, la fonte non è altro che un comune carattere digitato da una tastiera qwerty.
Il risultato sarà invece la visualizzazione di tale carattere su vari display LCD 16*2. Questa ultima parte ha come unico scopo la verifica della fedeltà della trasmissione (tornando all'esempio di prima, è come incidere un cd ed ascoltarlo per verificare la riuscita della masterizzazione).
Di seguito verrà analizzato un po' tutto il progetto in modo semplice e schematico, senza scendere troppo nei dettagli che renderebbero lunga e di difficile lettura, una tesina già complessa di suo.

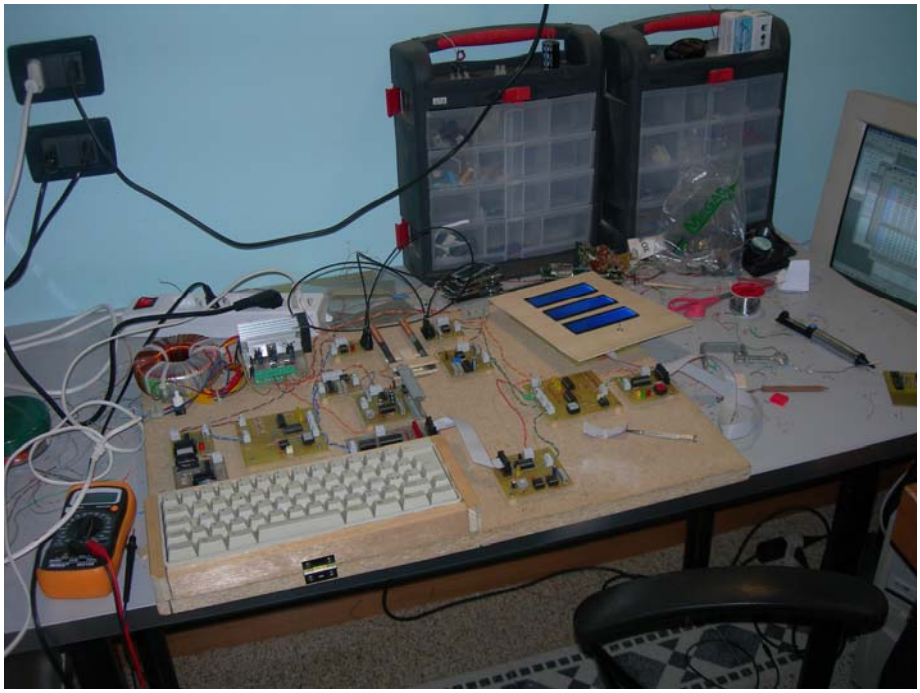
1b L'idea

All'inizio circa della 5^{a} ho cominciato a pensare ad un originale progetto d'esame, che potesse essere veramente innovativo o geniale.
Non riuscendovi, ho quindi iniziato ad ideare una semplice simulazione di trasmissione di numeri tramite fibra ottica. Questo tipo di progetto necessitava la conoscenza di argomenti non ancora del tutto trattati.
Per allargare il progetto, che mi pareva abbastanza banale o per lo meno di semplice realizzazione, ho deciso di fissare uno standard di comunicazione (standard unicamente per il mio progetto) e da questo ho esteso il trasferimento del segnale con altri due mezzi (che successivamente saranno ancora modificati e perfezionati): via radio in modulazione d'ampiezza (AM) e via cavo, in trasmissione dati parallela.
Verso febbraio ho quindi migliorato i vari blocchi, il loro compito e il loro metodo di lavoro.
La realizzazione è quindi iniziata verso marzo e, pian piano che il progetto prendeva forma si aggiungevano idee e quindi nuovi circuiti o soluzioni perfezionate.

1c Metodo di lavoro

Il metodo di lavoro da me attuato non è stato certamente il più corretto. Le cause possono essere varie. La principale credo sia dovuta alla poca esperienza che in generale uno studente possiede. In secondo luogo, lo stesso corso di studio mi suggeriva nuove idee o metodi che semplificassero il lavoro, così da convincermi spesso a cambiare circuiti. Inoltre, le poche ore disponibili in laboratorio mi costringevano a lavorare a casa, dove gran parte della strumentazione o dei componenti per il collaudo, non erano disponibili.
Inoltre la stessa evoluzione delle mie idee mi portava a cambiarle spesso, anche completamente. In assoluto ho deciso, come detto prima, di dedicarmi all'idea: inviare un dato o un treno di dati da una fonte all'altra. Successivamente, diviso il progetto in vari blocchi, mi sono dedicato ad ognuno di questi singolarmente, sempre tenendo conto di quello che precedeva e seguiva il blocco stesso. Per mia fortuna tutti i circuiti (a parte varie tarature previste in partenza) hanno subito funzionato.

Unicamente il caso della trasmissione dati via radio ha portato a qualche problema costringendomi a qualche modifica on-board.
In generale la progettazione dei circuiti stampati mi ha impegnato in tre fasi differenti, ognuna di queste è stata seguita dalla relativa realizzazione, test e implementazione nel circuito finale. (un lavoro a parte è stato fatto per i due programmi dei microcontrollori utilizzati)
Nonostante le varie difficoltà, credo di essere riuscito ad avere un buon rendimento nel rapporto risulti/tempo, permettendomi di perfezionare anche graficamente ed ampliare parti comunque futili al risultato (ad esempio l'utilizzo di tre display LCD in tri-state piuttosto che uno).
La cosa che più mi ha meravigliato è che tutto ha funzionato correttamente.

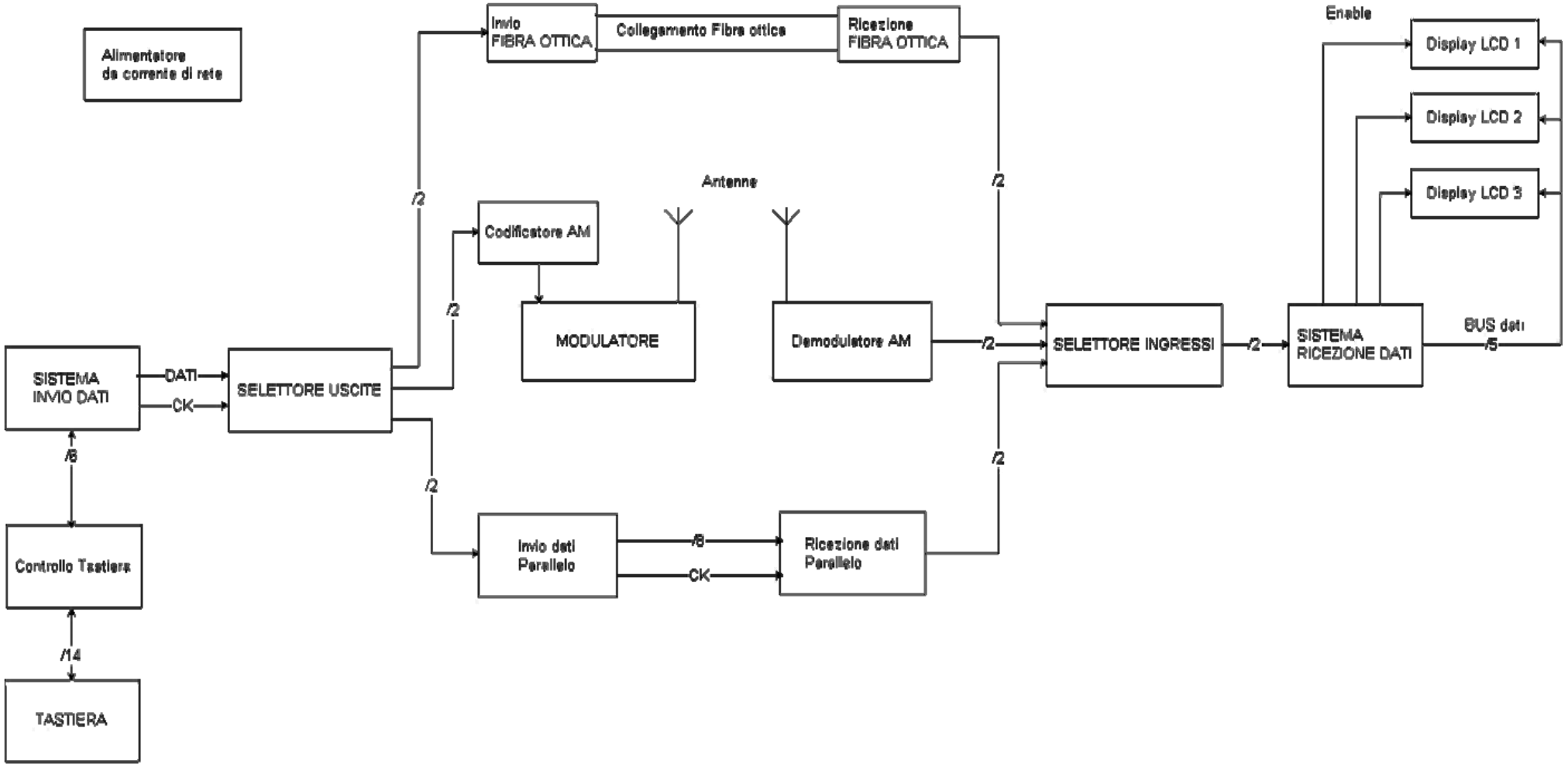


Progetto

2a Scopo

Il compito di questo progetto è quindi una trasmissione dati da una "stazione" di invio e una di ricezione. Come detto sopra, i mezzi di comunicazione utilizzati sono tre:
-Fibra ottica
-Radio AM (in modulazione d'ampiezza)
-Via cavo, in comunicazione parallela
Queste tre scelte sono state prese anche per la loro importanza nella storia delle telecomunicazioni. In tutti e tre i casi, ho deciso di utilizzare e adattare i componenti utili alla trasmissione in modo semplificato e più didattico, senza seguire le vere tecniche di applicazione, anche perché non sarebbero stati da progettare, ma solo da comprare, eliminando così tutta la parte di studio e applicazione che è stata necessaria prima del collaudo.
La verifica della trasmissione avverrà istantaneamente, controllando il carattere premuto e il carattere visualizzato sul display. Così si avrà, altre alla verifica, anche un risultato di simbolica utilità (immaginando di avere la trasmissione e la ricezione in luoghi differenti, ad esempio in due stanze separate). Difatti i blocchi di ricezione e i blocchi di trasmissione sono completamente separate (nel mio caso, ai fini pratici l'alimentazione è in comune, ma pure questa potrebbe essere separata) cosicché da poter allontanare le due parti ed avere i medesimi risultati, previe le dovute tarature.
Ora verranno visualizzati e spiegati i vari blocchi in modo molto schematico, per non appesantire troppo la tesina e complicare la lettura.

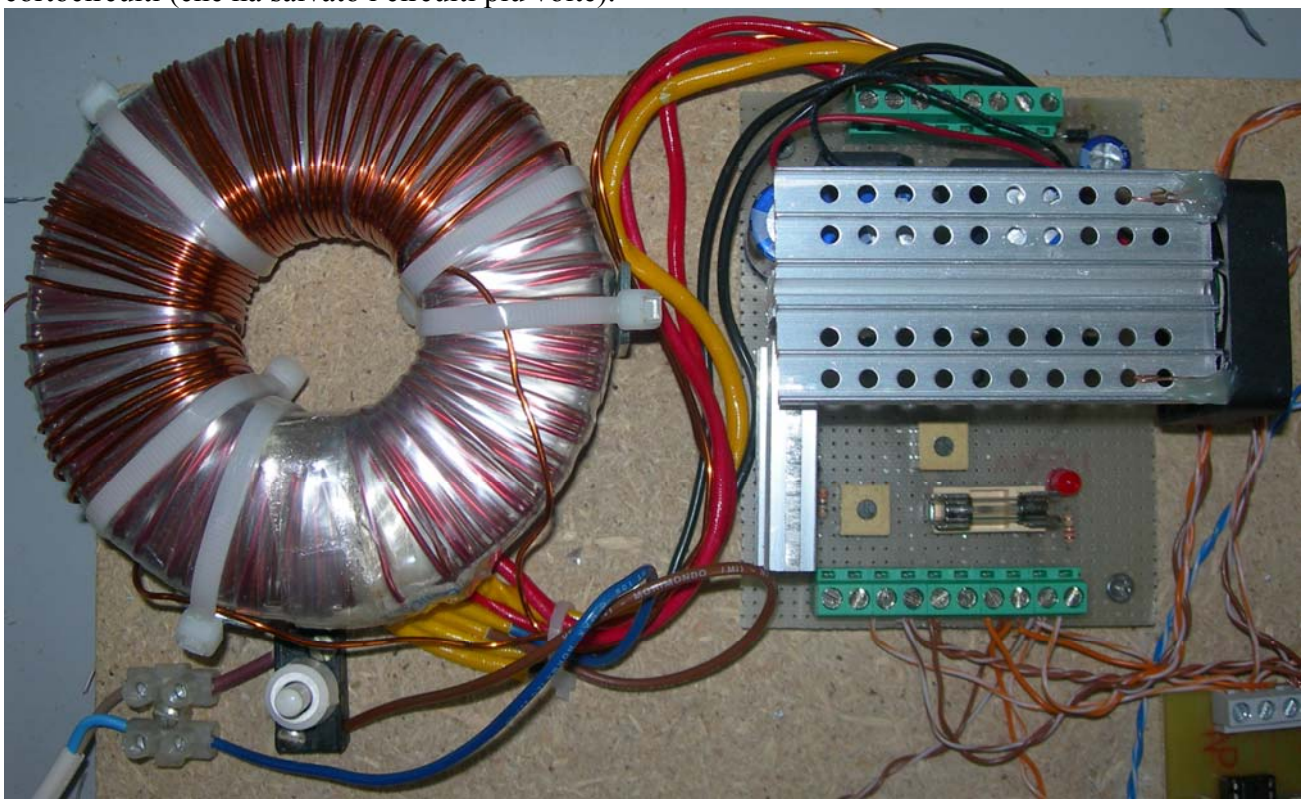
2b Schema a blocchi


Alimentatore
da corrente di rete
Invio
FIBRA OTTICA
Collegamento Fibra ottica
Ricezione
FIBRA OTTICA
Enable
Display LCD 1
Display LCD 2
Display LCD 3
Antenne
/2
Codificatore AM
MODULATORE
Demodulatore AM
SELETTORE INGRESSI
SISTEMA
RICEZIONE DATI
BUS dati
/5
SISTEMA
INVIO DATI
DATI
CK
SELETTORE USCITE
/8
Controllo Tastiera
/14
TASTIERA
Invio dati
Parallelo
Ricezione dati
Parallelo

2c Funzionamento blocchi

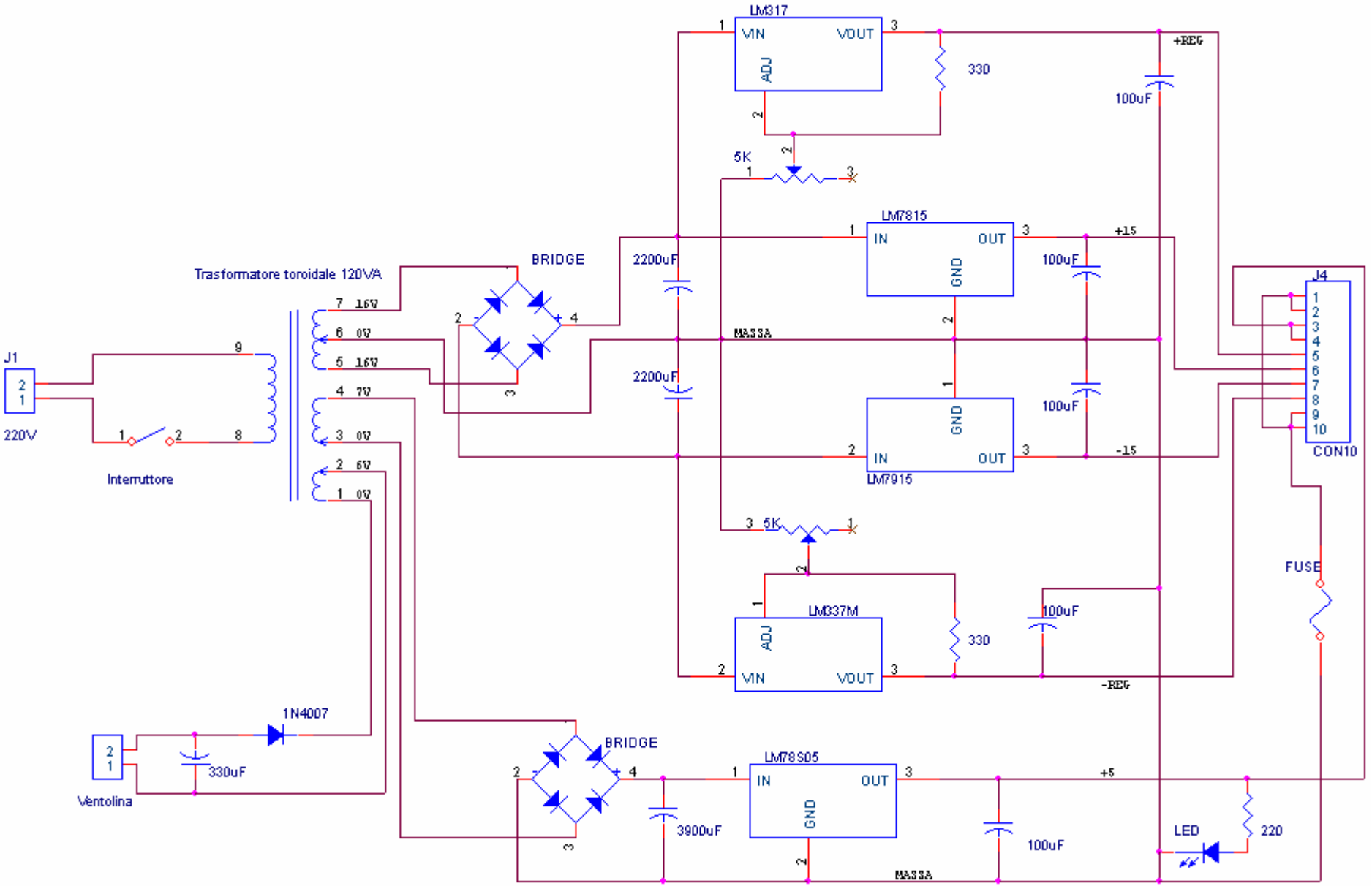
Alimentatore:

L'alimentatore è stato completamente progettato e realizzato da me. La progettazione è partita valutando le tensioni e le correnti necessarie. Sono risultate indispensabili una tensione duale a ±15V e una tensione TTL a 5V. In più ho pensato potesse diventare utile, se non necessario, per l'evoluzione del progetto e del collaudo due tensioni regolabili, una positiva e una negativa (da ±3V a ±15V). Inizialmente ho immaginato l'utilizzo di correnti molto alte (non assorbite) e comunque è sempre meglio prevedere un buon margine di sicurezza. Così ho disposto che le correnti massime siano di 1A per ogni tensione, meno la 5V in cui ho previsto 2A. Di conseguenza sono stati montati due dissipatori sopra gli integrati a tre terminali (uno per le tensioni positive e uno per le negative) e una ventolina da 4 pollici. Infine un fusibile veloce sulla massa che permette la protezione contro i cortocircuiti (che ha salvato i circuiti più volte).



Dalla foto è possibile vedere il trasformatore toroidale utilizzato. Esso è stato comprato ed aveva caratteristiche ottimali in quanto possedeva due secondari: uno a 16-0-16 (ottimo per la tensione duale e per quelle regolabili, e il secondo a 0-6. E' stato verificato però che quest'ultimo non riusciva ad erogare corrente e quindi mantenersi stabile, sopra i 400mA.
Ho dovuto quindi aggiungere un secondario esterno, con un filo di rame di diametro di 1mm, più che sufficiente per 2A. il numero di spire è stato calcolato per 7V (valore efficace). In modo da avere il massimo rendimento e la minor potenza dissipata, quindi meno calore sui dissipatori.
La ventolina nera, è posizionata in modo da riuscire a ventilare entrambi i dissipatori, tenuti separati, per comodità di posizionamento e perché il comune collegato ai vari integrati (PIN2) era differente e quindi sarebbe stato possibile un cortocircuito.
Tramite i due trimmer (i componenti di colore beige) è possibile regolare le due tensioni variabili. Una è stata poi utilizzata, e quindi fissata a -5V, utili per codificare il segnale per il modulatore AM (di cui parlerò in seguito). Il LED rosso, funge da spia di accensione.

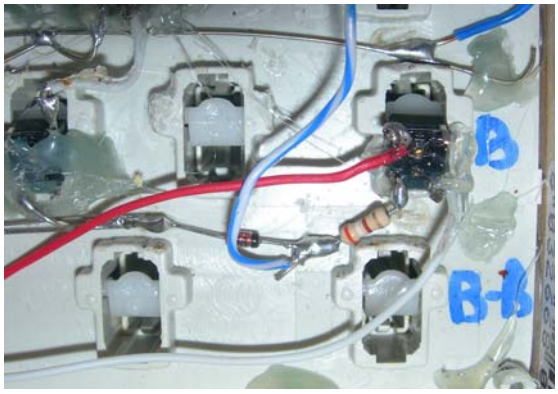
Schema elettrico:


LM317
VIN
VOUT
ADJ
330
100uF
+REG
5K
LM7815
IN
OUT
GND
+15
100uF
BRIDGE
2200uF
Trasformatore toroidale 120VA
7 16V
6 0V
5 16V
4 7V
3 0V
2 6V
1 0V
MASSA
J1
220V
Interruttore
LM7915
-15
J4
CON10
FUSE
LM337M
-REG
1N4007
330uF
Ventolina
LM78S05
+5
3900uF
LED
220

Cuore della stabilizzazione sono i due ponti di diodi (ad alta potenza), i tre grandi condensatori elettrolitici che filtrano la tensione alternata per i vari stabilizzatori a tre terminali. I condensatori posti dopo gli integrati servono a migliorare ancora di più la stabilità (da datasheet).
Questo circuito, insieme a quello di collegamento dei display LCD, è stato realizzato su una scheda millefiori, per permettere una miglior passaggio di alte correnti, difatti, i collegamenti che avrebbero dovuto far passare molti mA, sono stati "ingrossati" con stagno e fili più spessi, cosa molto difficile con il sottile strato di rame degli stampati.

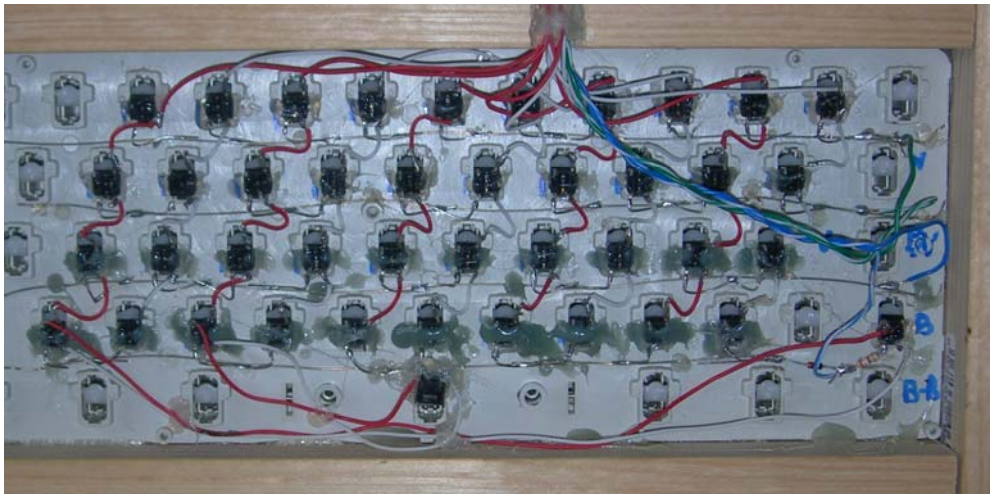
Tastiera

La tastiera è una semplice tastiera qwerty (quella normalmente utilizzata con il computer) tagliata in modo da diminuirne le dimensioni rendendo disponibili unicamente i tasti interessati.
Il problema di quali tasti utilizzare è stato un serio problema, risolto grazie al blocco *controllo tastiera*, di cui si parlerà in seguito.
Così, senza troppi problemi si è riuscito a controllare ben 40 tasti disposti su dieci colonne e quattro righe.(la classica disposizione a matrice)
Sono compresi tutti i numeri e tutte le 26 lettere, i quattro tasti rimanenti sono stati utilizzati come tasti funzione:



1) maiuscolo
2) backspace (canc)
3) spazio
4) invio (freccia giù)

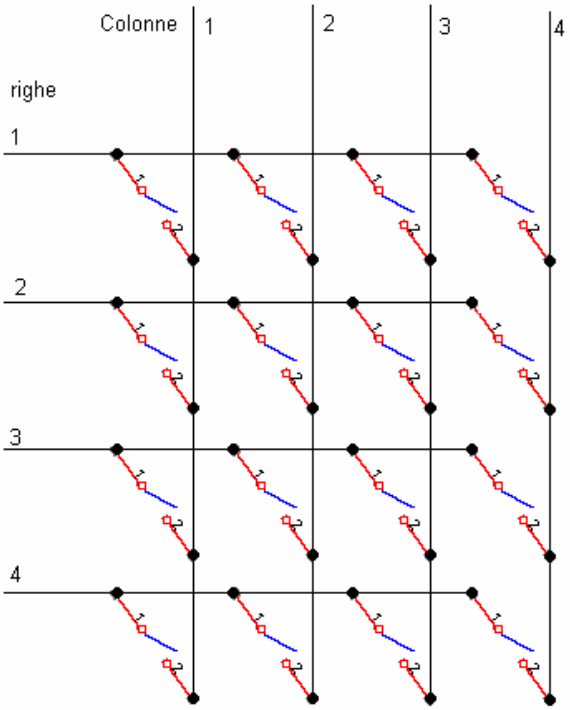
Difatti tramite un sistema di diodo-resistenza è stato possibile controllare in modo separato il tasto maiuscolo in modo che non interferisse sulla pressione degli altri tasti della stessa riga.



Dietro ogni tasto è stato posizionato un microinterruttore, incollato con colla calda e saldato agli altri interruttori con fili in disposizione a matrice (vedi immagine sotto)

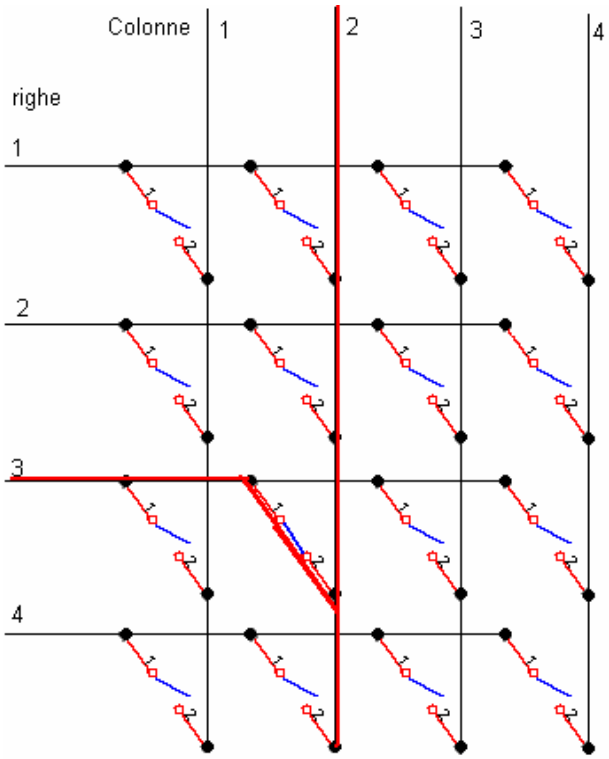
Controllo Tastiera:

Il controllo tastiera sarebbe stato inutile se non avessi necessitato di tanti tasti, o se avessi avuto tante porte input/output nel microcontrollore utilizzato. Ad esempio, se avessi avuto solamente tre tasti, avrei potuto collegarli direttamente a PIC (microcontrollore), se invece ne avessi avuto 16, con un sistema a matrice avrei potuto controllarli, cioè disponendo i tasti in righe e colonne è possibile con soli 8 pin controllarli tutti. Tra l'altro il PIC possiede ben 13 pin IN/OUT, quindi, considerando che 4 venivano utilizzati per funzioni esterne, ne rimaneva pure uno libero.



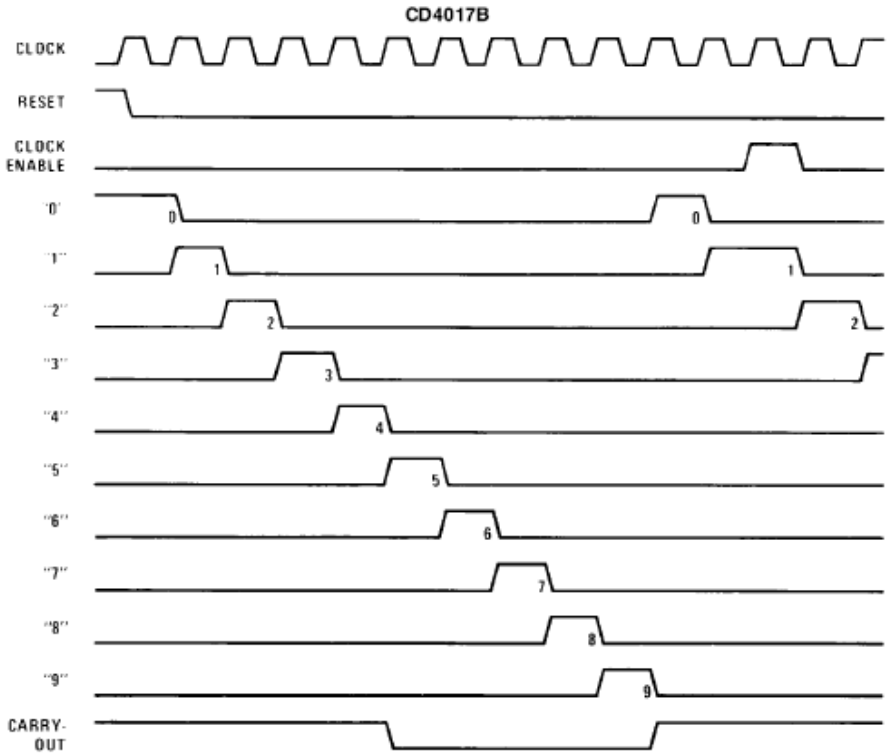
Gli otto pin vengono collegati rispettivamente alle quatto righe e quattro colonne.
Il PIC, abilita (porta a livello logico alto) una colonna alla volta e controlla se le quattro righe sono a livello logico basso (il tasto non è stato premuto); se invece il livello logico è alto, controlla quale riga e quale colonna, in modo da poter risalire al rispettivo carattere.

Se ad esempio viene premuto il terzo tasto della seconda colonna, solo la rispettiva riga, nel momento di pressione e di abilitazione, subirà il passaggio di corrente.



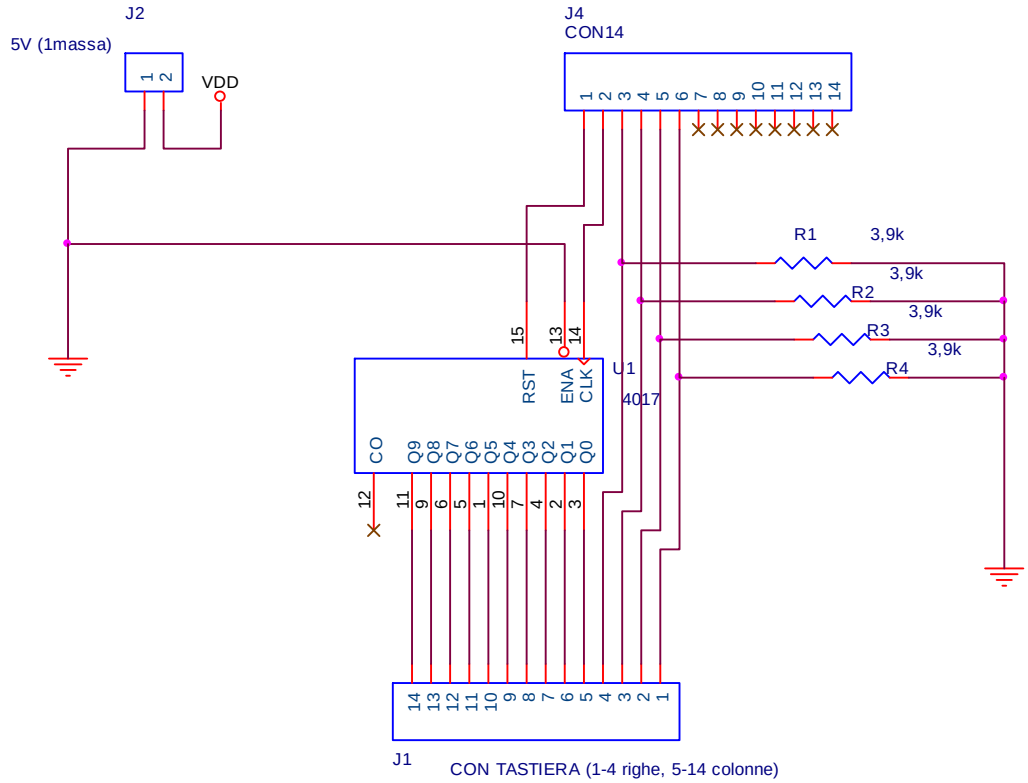
Con lo stesso principio si sarebbe potuto fare per la mia tastiera che possedeva dieci colonne e quattro righe. Quindi sarebbero stati necessari soltanto 14 pin, che però sono ancora troppi per le possibilità del mio PIC, anche perché altri quattro PIN sono necessari ad altre funzioni.
Ho così ideato un sistema di abilitazione automatico e sequenziale, in modo che il PIC, mandando un solo impulso, potesse controllare tutte e 10 le colonne, e intendere il tasto premuto semplicemente contando gli impulsi inviati.

Un sistema del genere è possibile farlo con un contatore Johnson (4017), una serie ad anello di flip-flop D, in cui al primo viene posto livello logico alto (di cui sotto la tabella di verità).
Necessario è pero l'utilizzo di un altro PIN, il reset, utile in quanto all'accensione non è sicuro che la colonna abilitata sia proprio la prima, se ciò non fosse, difatti, scombinerebbe tutto il controllo.
Proprio per questo dal PIC, sono utilizzate solamente 6 porte per il controllo, 4 per il controllo delle righe, uno per il reset, e l'ultimo per l'abilitazione delle colonne (ck).

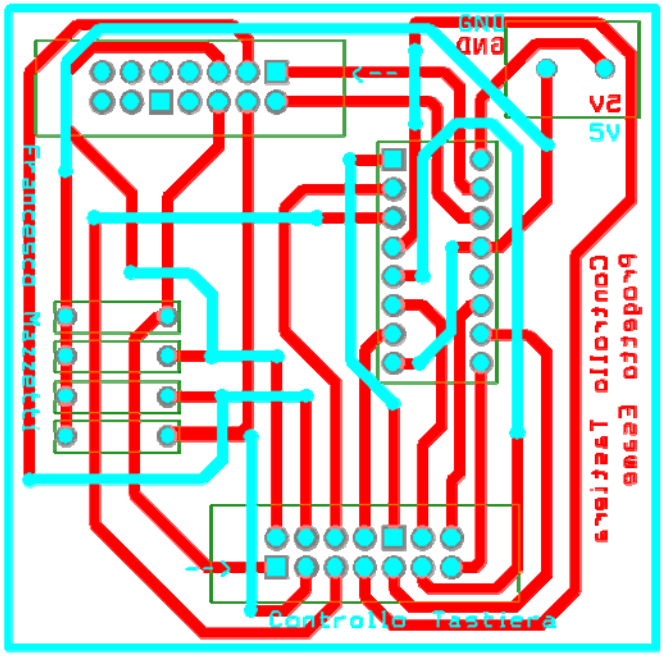


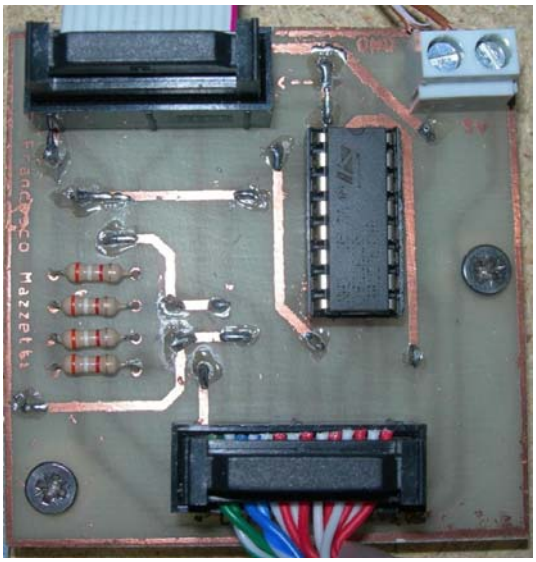
Il rapporto pin/tasti, che normalmente è a 1, raggiunge l'invidiabile risultato di 0,15.
Un altro notevole risultato è il numero di controllo di ogni singolo tasto, difatti ogni singolo carattere viene controllato circa ogni 5ms, tra l'altro questo controllo è stato molto rallentato dal codice sorgente del PIC in protezione dei possibili errori di selezione colonna.

Il circuiti che semplifica il lavoro al PIC è il seguente:



Le resistenze da 3,9KΩ vengono utilizzate come pull-down, cioè, nel momento in cui il tasto non viene premuto, non è vero che il filo è come se fosse collegato a massa, proprio per questo molti integrati potrebbero confondere questo stato, e rilevare un valore logico alto.
Il pull-down permette quindi a qualsiasi integrato di rilevare il livello logico corretto.
Questo circuito, come quasi tutti gli altri è stato realizzato su uno stampato in vetronite. Nonostante la sua semplicità è stato necessario realizzarlo in doppia faccia, in quanto molti incroci impedivano un collegamento diretto.
Ecco i due lati rame
(in azzurro il lato rame superiore, mentre in rosso, il lato inferiore)





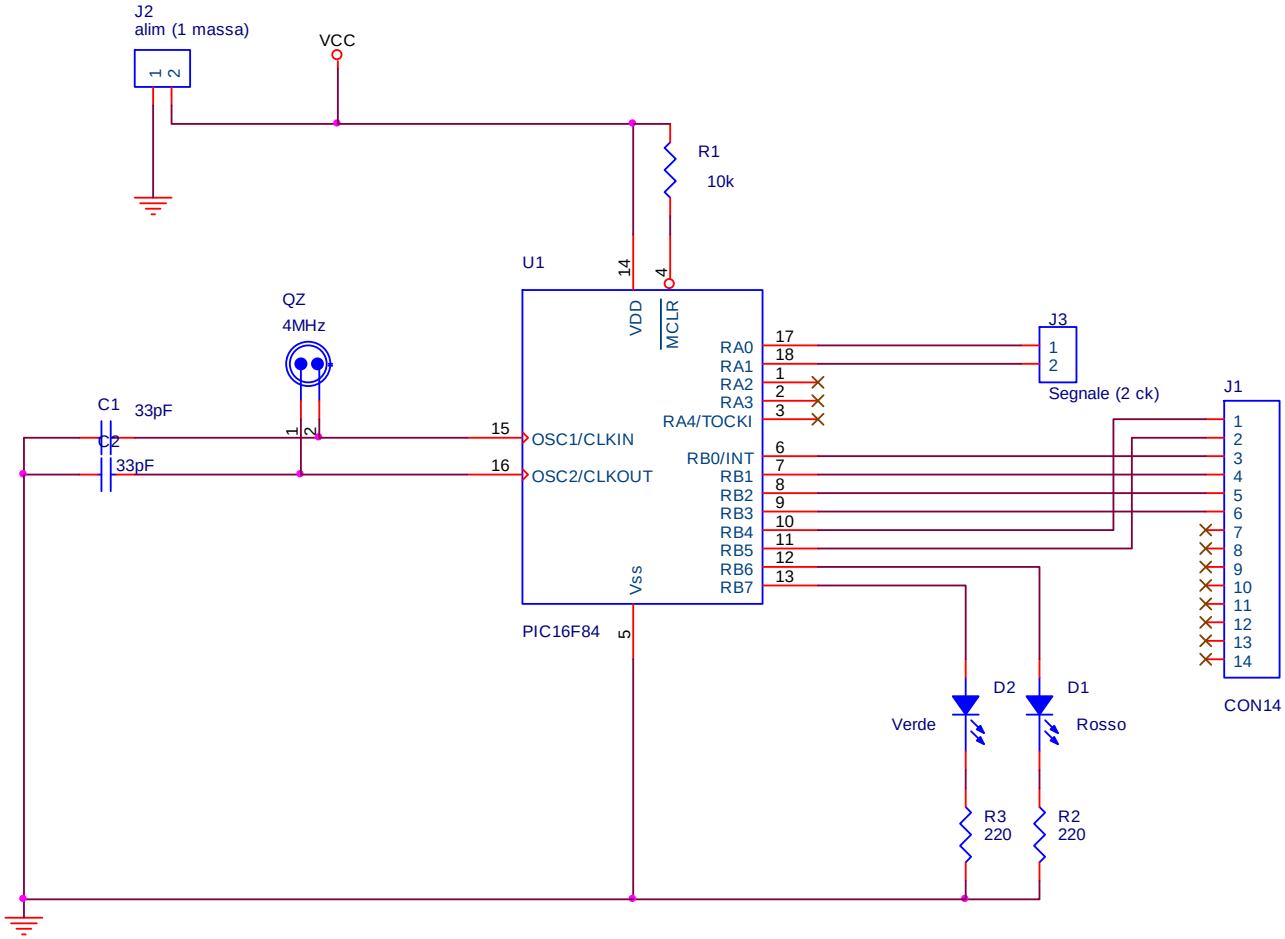
Sistema invio dati:
Questo blocco è il cuore della parte di invio dati, cioè è il circuito in cui è presente il PIC, l'importante microcontrollore cui spetta compiti programmabili.
Il suddetto PIC, difatti dovrà controllare in un ciclo infinito la pressione dei tasti, riconoscere tale carattere e inviarlo tramite due uscite, quella del segnale vero e proprio, e quello del clock, che dà la temporizzazione del rilevamento al sistema ricevente.
A livello software è possibile anche, controllando se viene premuto il tasto MAIUSC, portare a 78 i byte differenti inviabili e riconoscibili.
Inoltre prima dell'invio verrà controllata la parità degli 1 presenti nel byte da spedire (difatti i caratteri vengono spediti in codice ASCII, cioè in gruppi di 8 bit) e in caso di esito negativo, l'ultimo bit (l'ottavo) verrà forzatamente portato a 1, in quanto come è possibile verificare nella tabella, i caratteri da me utilizzati rientrano tutti in 7 bit (cioè fino al 128 carattere), lasciando quindi libero l'ottavo. Questo genere di controllo diventa poi utile in ricezione, permettendo di capire con una media sicurezza se il byte rilevato è corretto o è stato soggetto a qualche errore di trasmissione.

0		32		64	@	96	`	128	Ç	160	á	192	└	224	Ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
9	○	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13	♪	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	☼	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
16	►	48	0	80	P	112	p	144	É	176	░	208	ð	240	-
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
19	‼	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	§	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	▬	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	↑	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	∟	60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
29	↔	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	▲	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	

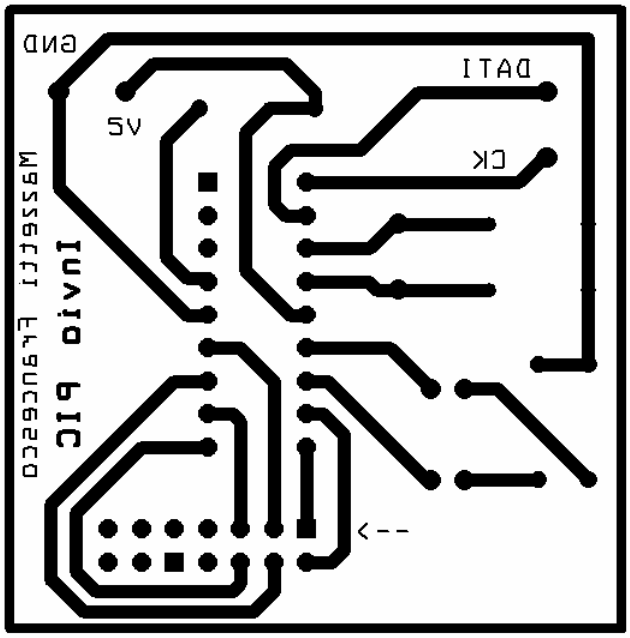
Tale scheda presenta due LED utili ad intendere lo stato in cui si trova il PIC: se è accesa la spia rossa, il microcontrollore è in attesa della pressione di un tasto, contrariamente, a spia verde, il sistema è in "invio dati". Infine in caso di lampeggio simultaneo, il sistema è in pausa, in attesa di inizializzazione.

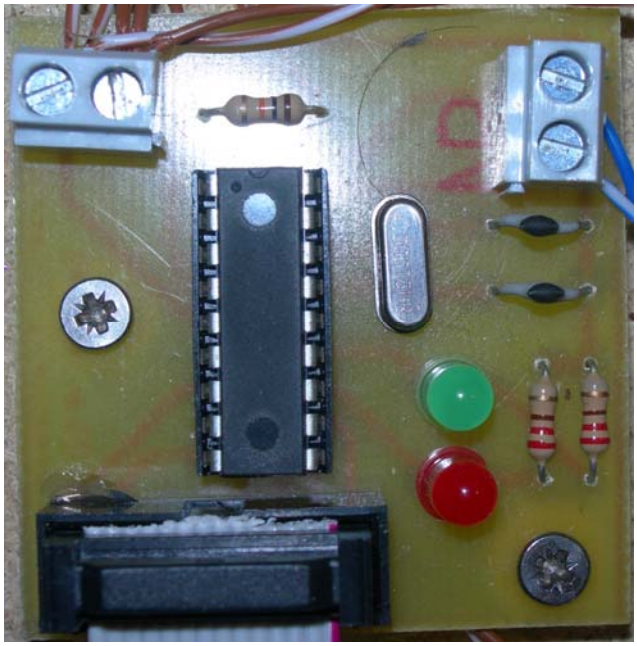
Il listato (programma del PIC), insieme a maggiori informazioni riguardo la trasmissione dati, sono leggibili nei prossimi paragrafi.

Circuito elettrico:



La comunicazione con il blocco “controllo tastiera” è effettuato con un connettore a 14 PIN, anche se, come è visibile, ne vengono utilizzate solamente 6. La motivazione è puramente commerciale, in quanto sono riuscito a trovare connettori soddisfacenti solamente a 14 PIN, la stessa cosa accadrà poi in altre situazioni. Il quarzo utilizzato è a 4MHz, ottimale nel calcolo delle temporizzazioni, in quanto ogni istruzione viene eseguita ogni quattro impulsi di clock, cioè esattamente 1µs.(10^{-9}s)

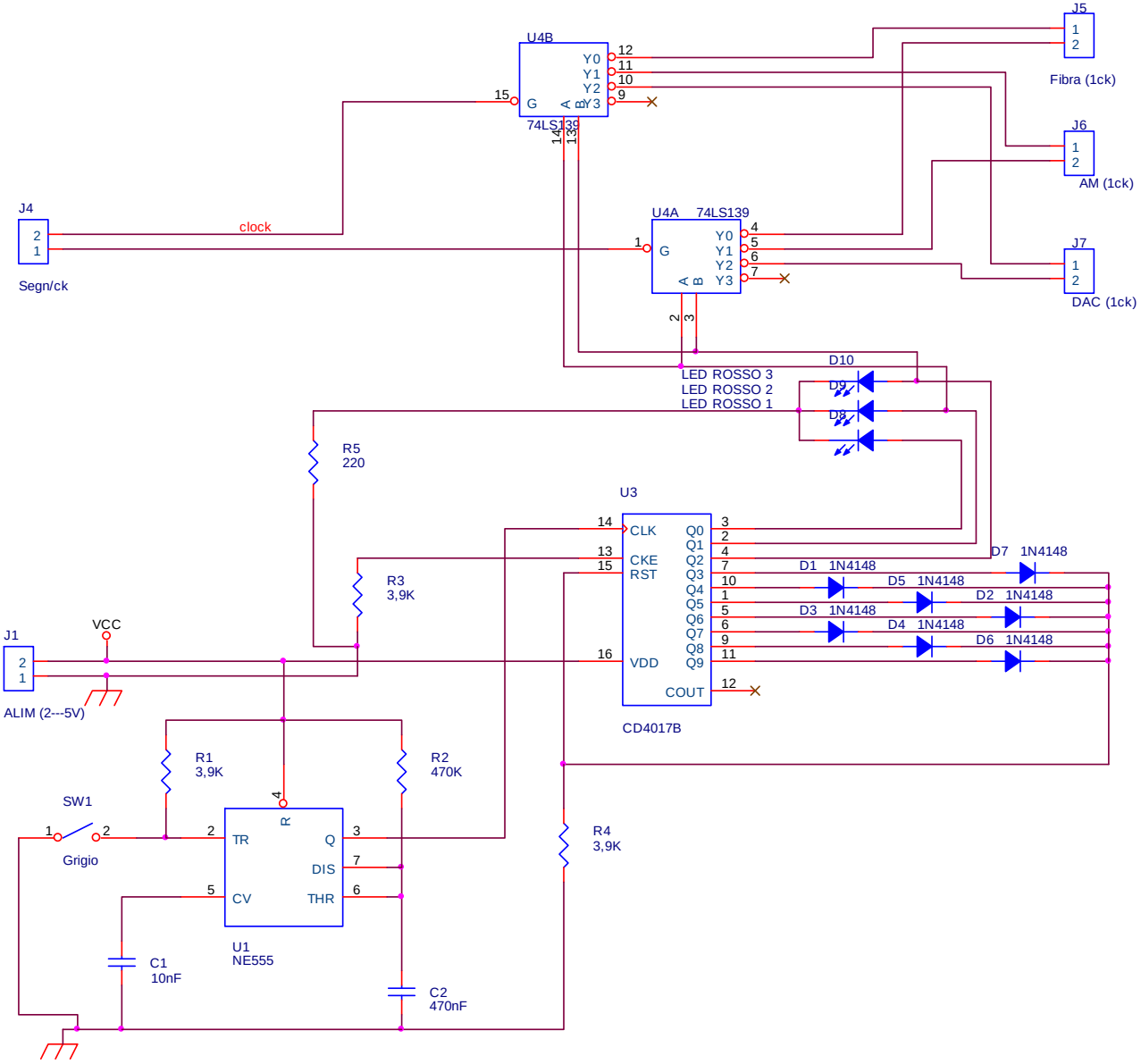




Selettore uscite:

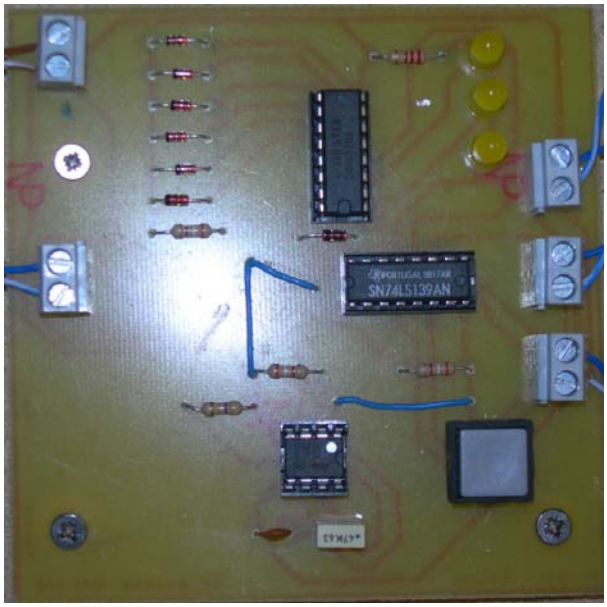
Il compito di questo circuito è molto semplice, nonostante questa semplicità il circuito si è rivelato complesso e di grandi dimensioni. Anche perché ho cercato di utilizzare metodi didattici e elettronicamente corretti (ad esempio il circuito rigenera pure il segnale TTL).

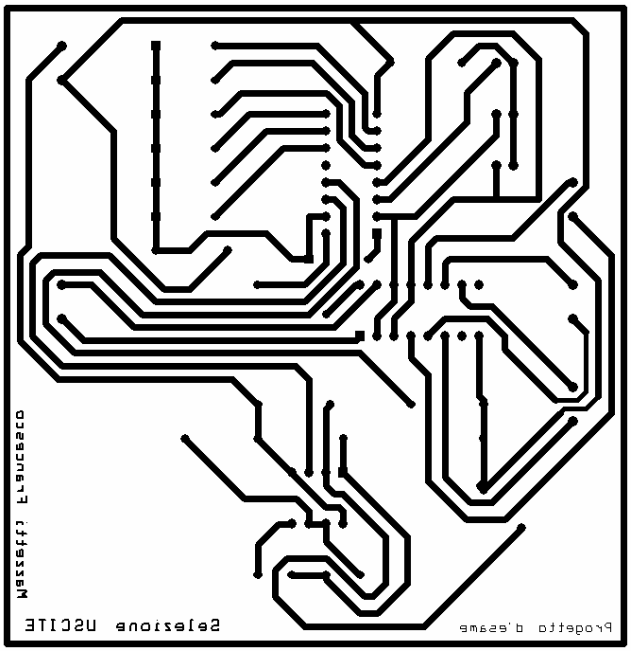
Il segnale in entrata deve essere smistato alle tre tecniche di invio dati. Sia il segnale, che il clock deve quindi passare tramite un demultiplexer, nel mio caso ho scelto un integrato TTL (74LS139).
Per selezionare la commutazione voluta utilizzo l'ormai già conosciuto 4017 (contatore Johnson) con un NE555 in modalità monostabile che attenua i rimbalzi creati dalla pressione del pulsante.
Ci saranno inoltre tre LED che evidenziano la deviazione effettuata.



I diodi in uscita al 4017, servono ad impedire che il circuito si trovi in uno stato non desiderato (oltre al terzo).
In questo modo, nel caso sia attiva la prima uscita, verrà acceso il primo LED, alla pressione del tasto sul monostabile, si attiverà la seconda uscita, portando a 01 l'ingresso del demultiplexer e quindi attivando la seconda fonte. In seguito ad un ulteriore impulso l'uscita sarà 10 (abilitata la terza fonte) e in caso di un successivo impulso, il diodo farà passare un impulso alto al reset che riporterà a 00 la selezione del demultiplexer.

Ho deciso di utilizzare un contatore Johnson piuttosto che un normale contatore molto più standard, unicamente per avere la semplice possibilità di inviare un reset superato il terzo impulso (non esistono contatori modulo 3 ed avrei ulteriormente complicato il circuito ponendo una porta AND che limitasse il conteggio) e perché diventava molto più semplice la visualizzazione tramite spie-LED dell'uscita selezionata.
In basso è possibile vedere il 555 e il pulsante utile per l'antirimbalzo, l'integrato posto in mezzo è il demultiplexer, mentre quello posto più in alto è l'ormai famoso 4017.
Nel circuito, come in altri, è risultato necessario l'utilizzo di ponti (con i fili blu), questa soluzione viene presa in casi di reale necessità, cioè quando non esistono alternative, se non la realizzazione di una complessa scheda doppia faccia.

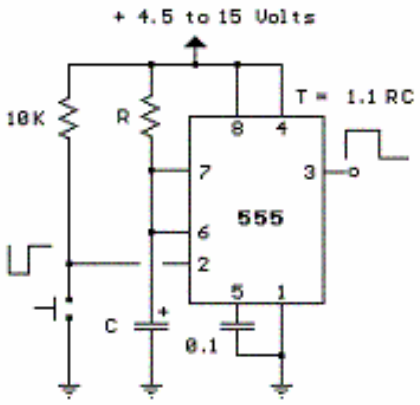




Selettore ingresso:

Il principio di funzionamento del selettore ingresso è praticamente identico al precedente circuito, l'unica differenza risiede nell'utilizzo di un multiplexer piuttosto che un demultiplexer.

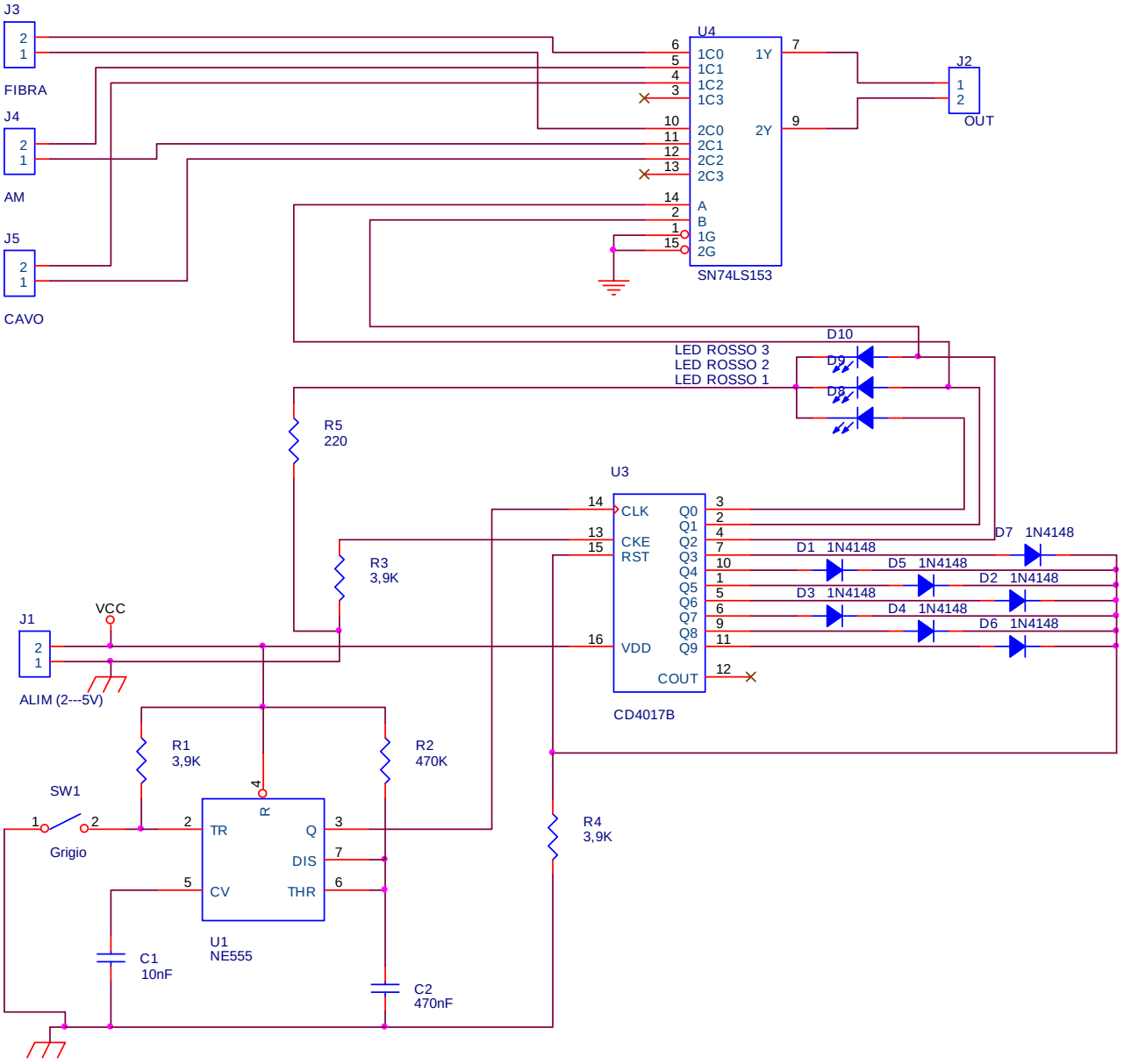
Per approfondire l'argomento, farò accenni al monostrabile con NE555:
il circuito di base è il seguente:



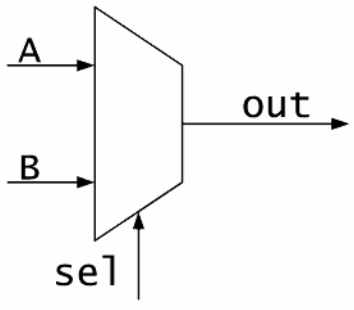
La formula che regola il tempo di salita è 1,1R*C.
Ciò significa che alla pressione dell'interruttore il segnale d'uscita (pin 3) rimarrà alto per un tempo circa uguale al prodotto della resistenza e capacità.
Nel mio caso ho preso come tempo 0,25s, più che sufficienti per limitare un possibile rimbalzo.
Fissando la capacità a 470nF, ho trovato soddisfacente l'utilizzo di una resistenza di 470K

$$R = \frac{0{,}25}{1{,}1 \cdot 470 \cdot 10^{-9}} = 483K\Omega$$

(viene sempre fissata prima la capacità in quanto i valori commerciali sono sempre inferiori rispetto alle resistenze)



Approfondirò pure il funzionamento del multiprexer, base di molti circuiti:

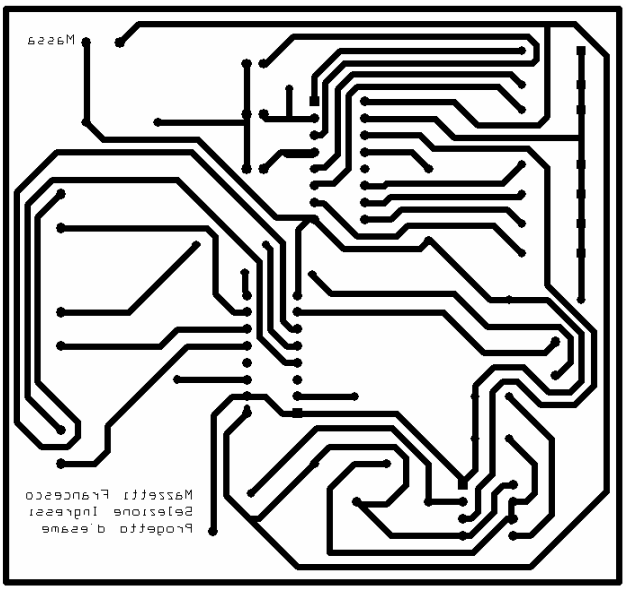


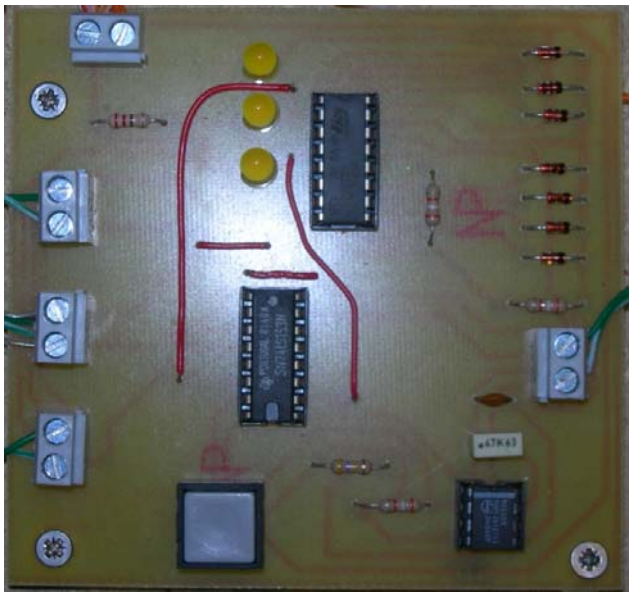
Dall'immagine è facile intuire il funzionamento, cioè grazie alla selezione (solitamente in forma digitale) è possibile permettere di far passare l'ingresso A o B. Normalmente i MUX (abbreviazione di multiplexer) hanno più PIN di selezione(n), così da avere 2^N ingressi possibili.

Esistono pure MUX analogici (ne utilizzerò uno nel codificatore per il segnale AM). Quest'ultimi hanno la caratteristica di avere in uscita esattamente la tensione in ingresso, anche non TTL, quindi variabili e addirittura negative.

Il demultiplexer, non è che l'inverso, cioè possiede un solo ingresso e più uscite abilitate dai pin di selezione. (esistono integrati capaci di fare entrambe le cose).

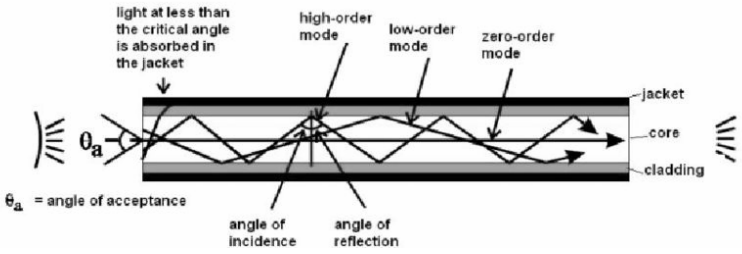
Ecco la foto del circuito e il relativo lato rame



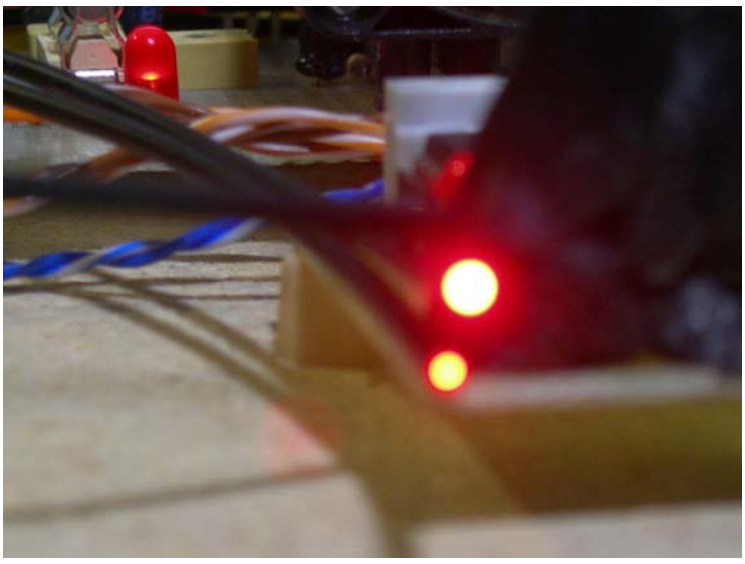


Invio Fibra Ottica

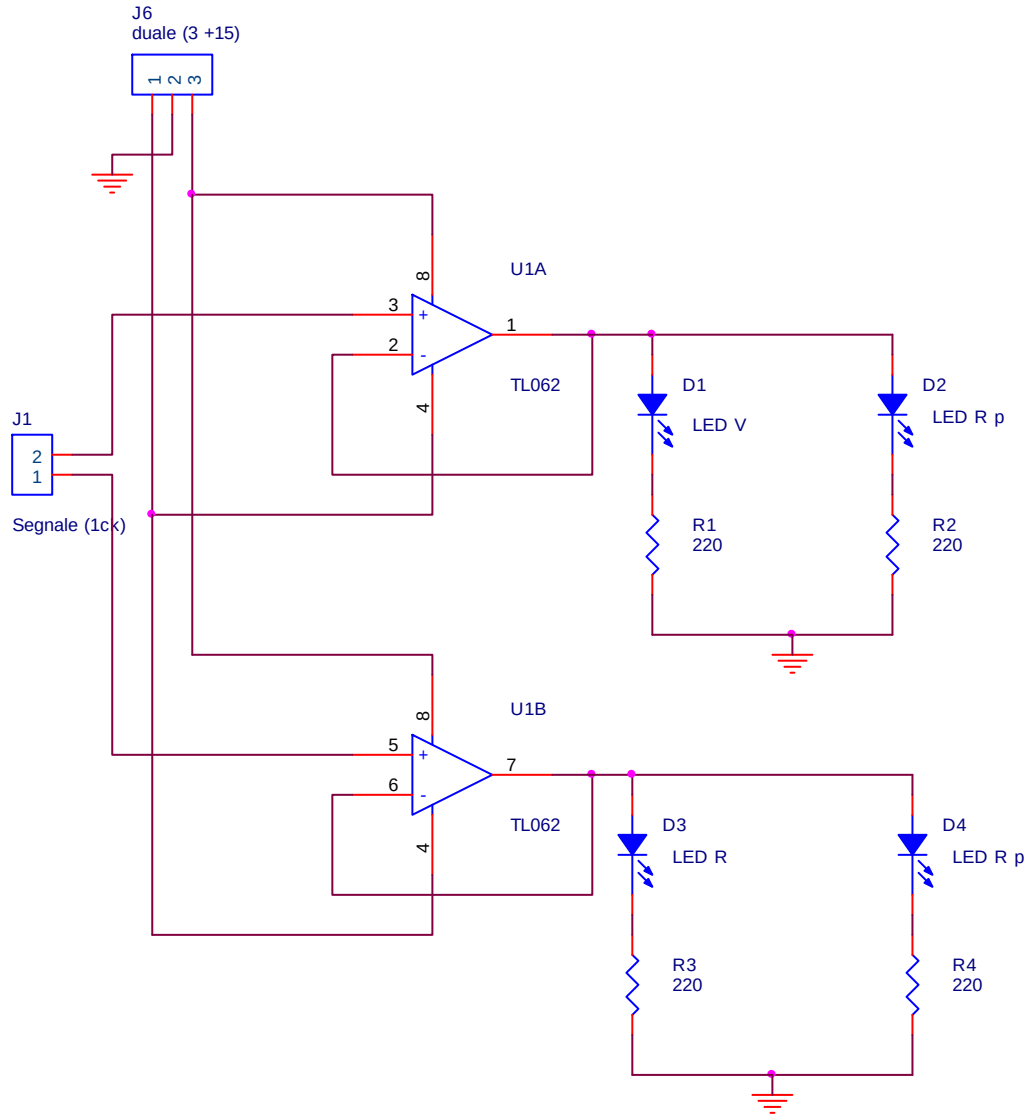
Da qui inizia i veri e propri circuiti che inviano e ricevono il segnale assegnato.
Il primo, e il più semplice è l'invio tramite fibra ottica.
La fibra ottica è un materiale plastico o in vetrite, con la capacità di riprodurre trasmettere un'informazione luminosa da un capo all'altro della stessa, hanno quindi una altissima capacità di trasmissione (naturalmente non sfruttata appieno dal mio sistema).



La loro principale caratteristica è quella di riflettere il segnale luminoso attraverso un canale detto "core", e senza scendere troppo nei dettagli, hanno come difetto la rifrazione. Proprio per questo come fonte luminosa viene normalmente utilizzati diodi laser o LED ad alta efficienza, in quanto posseggono solamente una piccola gamma del segnale luminoso, e quindi in ricezione non potrà che arrivare solo quello, senza rifrazione.
Nonostante che, nel caso di piccole distanze e basse frequenze, questo tipo di problemi, e altri tipici delle fibre ottiche, scompaiano quasi completamente, ho deciso di utilizzare LED ad alta efficienza, ottimali nei consumi e nella potenza luminosa capaci di erogare (quasi direzionale).
Difatti come si riesce a vedere dalla foto, la luce quasi abbaglia in uscita dalla fibra, questo rende molto ottimale la rilevazione dal sistema di ricezione.



Lo schema elettrico è il seguente:

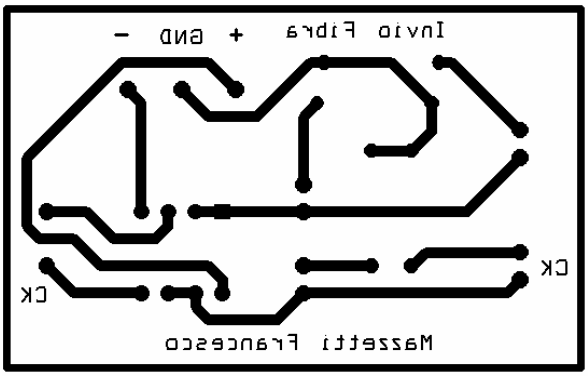


Naturalmente il circuito necessita di alimentazione duale in interesso, in quanto sono presenti amplificatori operazionali. Ho deciso di utilizzare l'integrato TL062, in quanto possiede una buona risposta in frequenza ed eroga nettamente più corrente del TL082 (solo 10mA, contro i 60mA della serie TL06X, e tenendo conto che ogni LED consuma circa 15mA, si avrebbe avuto un riscaldamento eccessivo del componente).

Tale integrato, possiede al suo interno esattamente due operazionali, e si è reso necessario in quanto a monte, il demultiplexer, non sarebbe riuscito ad alimentare tutti e 4 i LED. La sua configurazione difatti di semplice inseguitore: da in uscita esattamente la stessa tensione in ingresso, però senza caricare e quindi modificare ciò che lo precede.

Per riuscire a collegare i due LED ad alta efficienza ho costruito una base in plastica modellabile al calore (vedi foto).
Gli altri LED sono semplicemente di controllo (verde per il segnale e rosso per il clock).

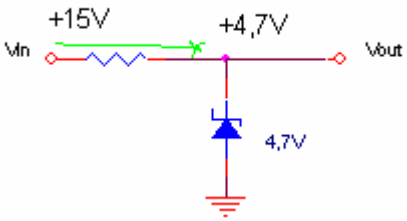
Ecco il lato rame e la foto dall'alto.



Ricezione Fibra Ottica

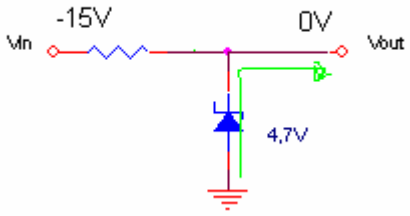
Il blocco relativo alla ricezione è relativamente più complesso, difatti tale sistema dovrà controllare la presenza o meno di luce nella fibra ottica. Il metodo normalmente utilizzato è un fototransistor. Tale componente riesce a variare la corrente in uscita al variare della luce.

La sfortuna è che in commercio ne esistono solamente sensibili ai raggi infrarossi, l'utilizzo di LED infrarossi sarebbe stato meno scenografico, così, anche per semplificare il circuito utilizzo una semplice fotoresistenza (resistenza variabile alla luce) che però presenta una grande inerzia, comunque senza influire nel mio risultato (cioè la resistenza varia valore abbastanza lentamente al variare della luce).
I valori di resistenza rientrano nella gamma 1MΩ (buio) e 2KΩ (luce).
È quindi necessario creare un partitore in modo da rendere "misurabile" la luce presente sulla fibra. Tale misura sarà poi confrontata tramite un operazionale in configurazione di comparatore con una tensione regolabile attraverso un trimmer, per poi essere stabilizzata con uno zener, in quanto il comparatore darà in uscita tensioni unicamente duali e ai limiti della sua alimentazione (cioè ±15V), estremamente pericolose per la logica TTL.

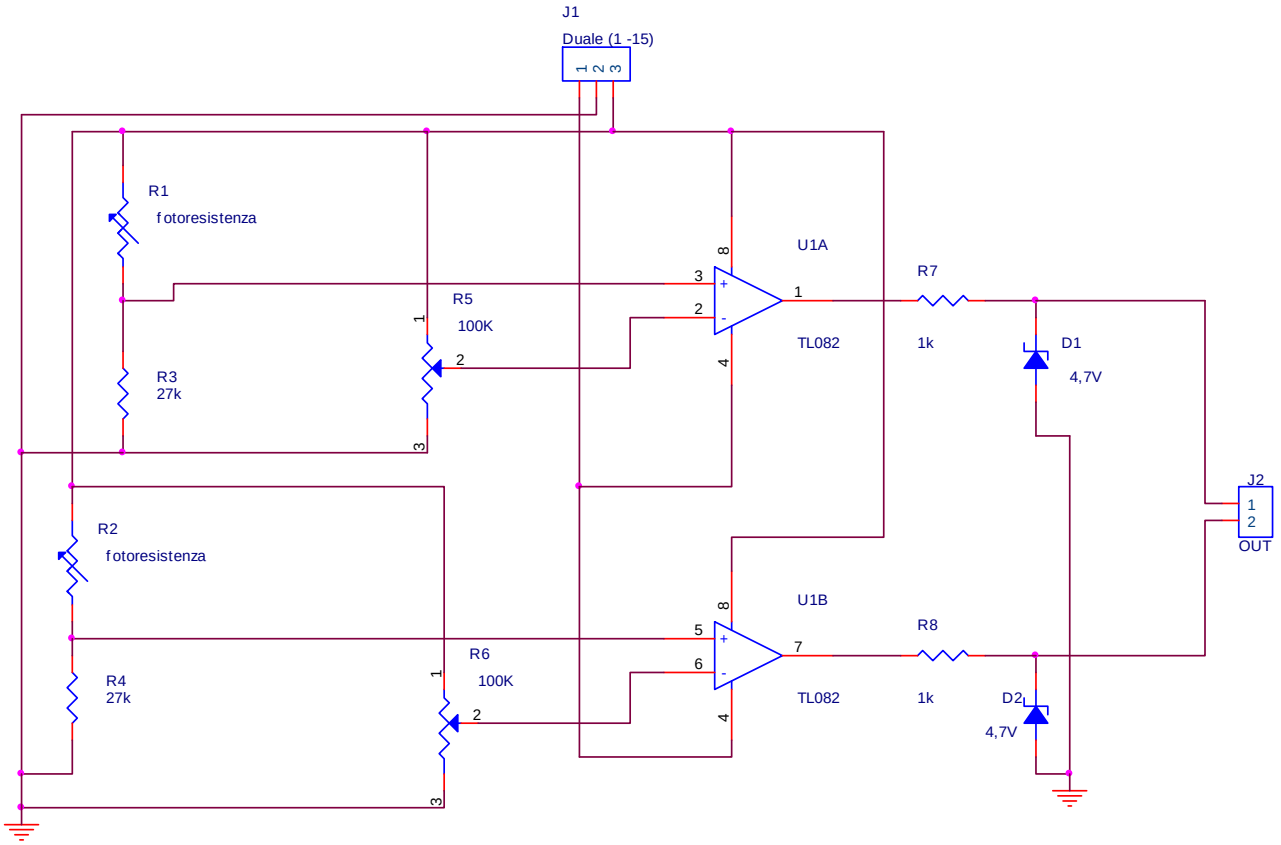


Lo zener ha la capacità di limitare la gamma di tensione tra massa (in caso di tensioni negative) e il suo valore nominale (nel mio caso 4,7V). Cioè se in uscita avrò 15V, lo zener lo regolerà a 4,7V, mentre nel caso di uscita a -15V, lo zener diventerà come un diodo portando a 0V l'uscita.
La resistenza dovrà essere calcolate in modo da garantire il passaggio di al massimo 10-15mA.

$$R_Z = \frac{15 - 4{,}7}{0{,}01} \approx 1K\Omega$$

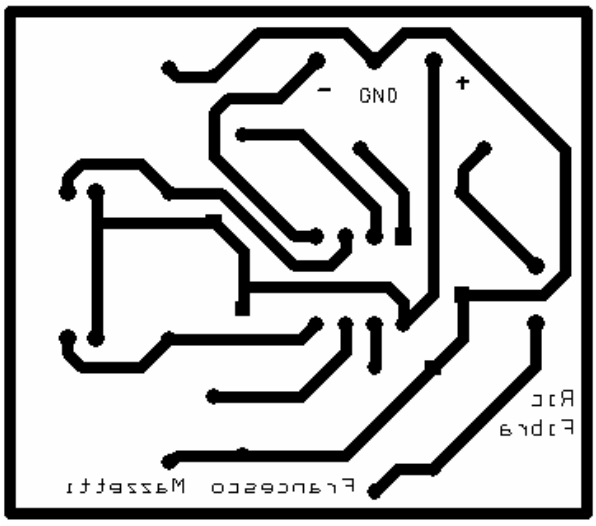


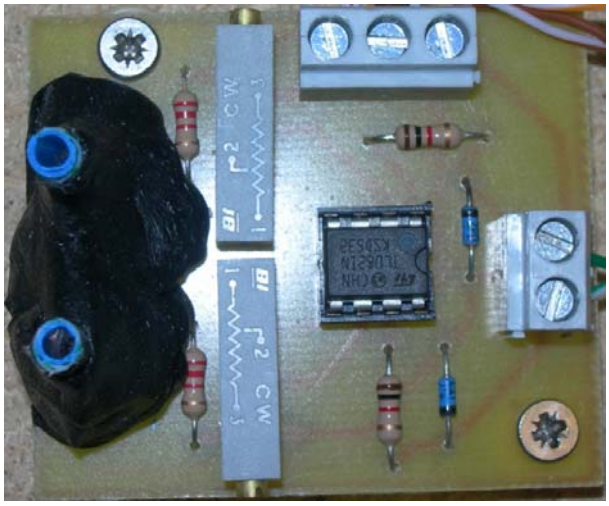
Di seguito è riportato il schema elettrico:



Il partitore è stato composto da una resistenza da 27KΩ, così da avere in caso di luce circa 7V, mentre in caso di buio tensioni sull'ordine dei mV (praticamente 0). In questo modo, regolando i trimmer in modo ottimale (circa sui 3,5V) sarà possibile ricostruire in modo ottimale il segnale e il clock.
In questo caso ho utilizzato degli operazionali TL082, ottimi in frequenza, ma con massime correnti in uscita più modeste, ma sufficienti per lo scopo.
I trimmer sono multigiro, più costosi ma con una maggiore precisione di regolazione, e stabilità nel tempo e negli spostamenti.

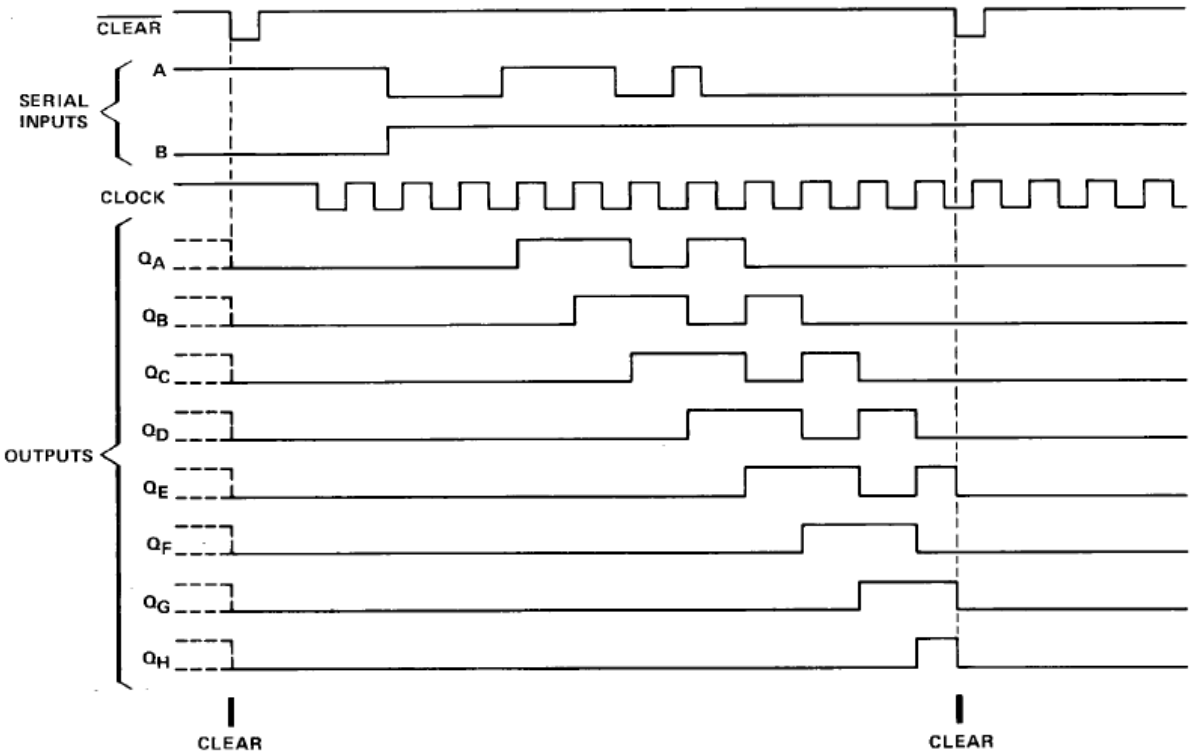
Ecco il lato rame e la foto del circuiti (anche in questo caso la base per la fibra ottica è stata realizzata da me con il materiale plastico malleabile e delle cannucce.





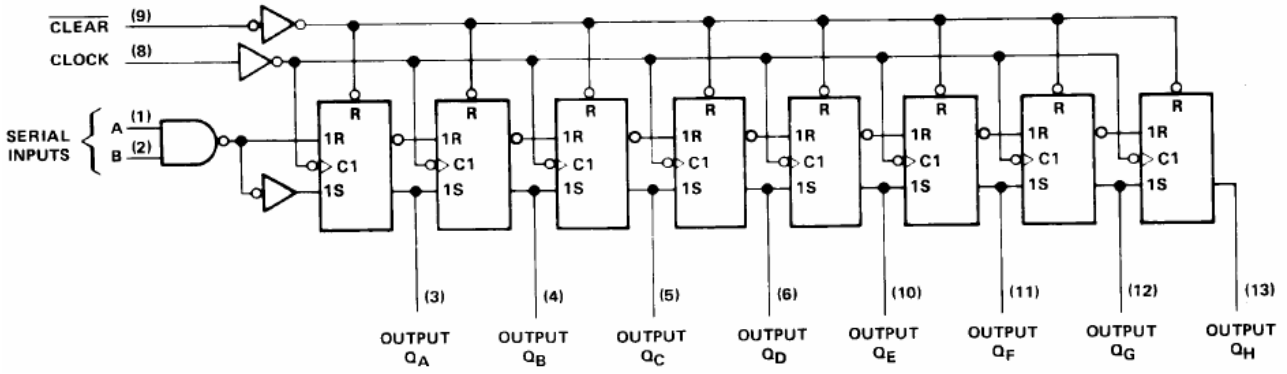
Invio dati Parallelo:

Questo blocco prevede la trasmissione del dato in modo parallelo, difatti normalmente la trasmissione viene incanalata in modo seriale (tramite il collegamento del clock e quello dei dati).
Il progettazione in teoria, potrebbe essere semplice, in quanto basterebbe un registro SIPO (serial-input, parallel - output) con un contatore, che informi in ricezione l'avvenuta "conversione" seriale-parallelo.
Il mio sistema prevede di inviare parole di 8 bit (un byte alla volta), quindi il canale di comunicazione sarà 8+1 (clock). Come è successo precedentemente ho utilizzato una piattina a 14 poli in quanto ho trovato disponibile solo quel tipo di connettore.
Come spesso accade, ciò che in teoria dovrebbe funzionare, a livello pratico potrebbe diventare un rovina. Ad esempio nel caso arrivino due byte consecutivi, o un dato a 16 bit, nel momento in cui il sistema di ricezione incomincia con la lettura del primo byte, questo potrebbe venire scombussolato dall'arrivo troppo rapido del secondo. C'è anche da dire, però, che il sistema di ricezione dovrebbe riuscire lui stesso, caricando in un registro SIPO, a mantenere la memoria. Ma questo dipende esclusivamente, dall'integrato utilizzato, e per rendere più universale il mio sistema, il byte in arrivo, prima di essere inviato, verrà "bloccato" in un registro PIPO (parallel-input, parallel-output, costruito con una fila di 8 flip-flop D).
In questo modo, solamente all'arrivo del nuovo dato, e quindi di 8 impulsi di clock, questo registro verrà aggiornato, consentendo in ricezione una trasmissione ininterrotta, anche di informazioni molto più lunghe di 8 bit.
Il clock, generato dal solito contatore johnson, aspetterà gli otto impulsi, al termine si autoresetterà (tramite diodo e resistenza sul reset) e avrà la duplice utilità di aggiornare il registro PIPO, e quindi di inviare al sistema ricevente l'impulso di conferma che un nuovo dato è disponibile.
Tramite i seguenti grafici spero diventi più facile comprendere il funzionamento di questi registri.
Di seguito verrà analizzato il registro SIPO 74LS164

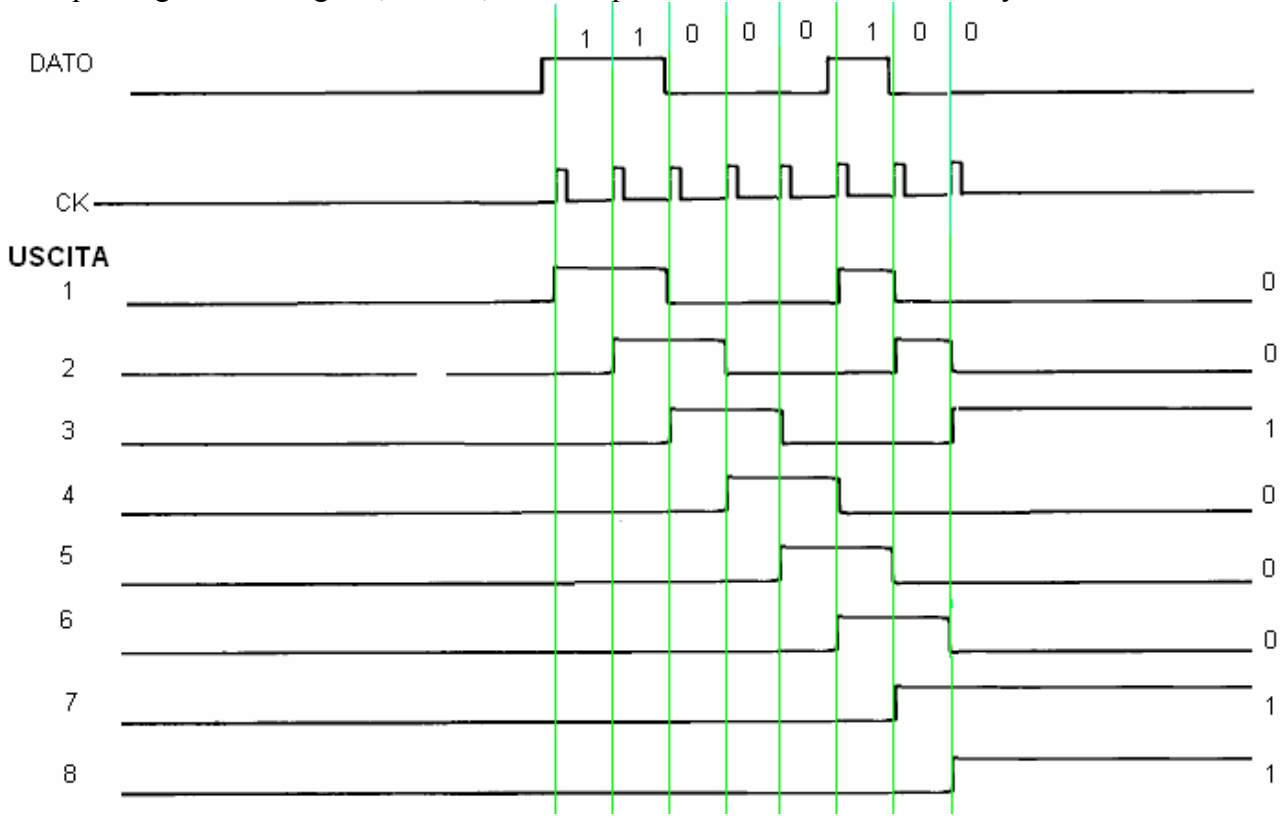


I due *serial input*, sono gli ingressi seriali (uno può essere usato come enable), come è possibile vedere, il dato viene rilevato e quindi spostato nel registro al fronte di salita del clock, mentre il clear è attivo basso (difatti nel mio circuito viene tenuto sempre alto). Il clock viene preso direttamente quello in uscita dal "selettore uscite".

Di seguito lo schema elettrico semplificato dell'integrato



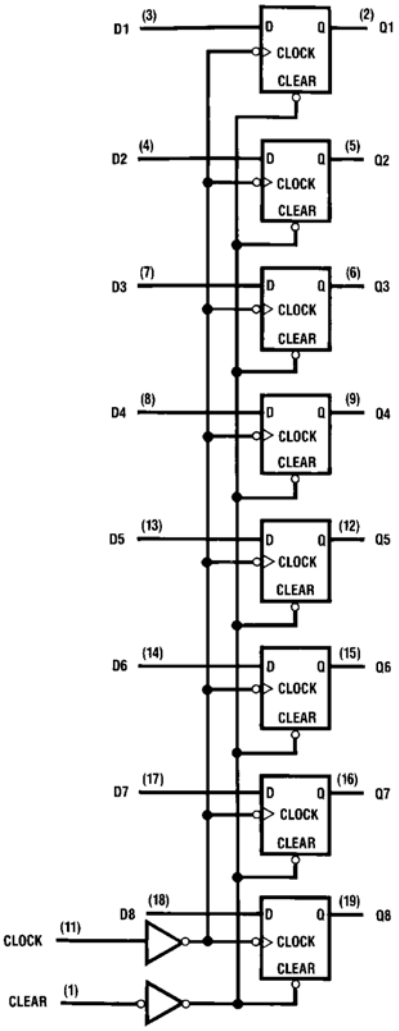
Esso chiarisce ancora di più il movimento del dato attraverso i vari flip-fpop, e le rispettive uscite che si avranno.
Con questo grafico di seguito, simulo, in modo più realistico di l'invio di un byte:



Come si può vedere il primo uno inviato, diventerà ultimo, mentre l'ultimo bit, quindi quello più significativo, sarà nel primo canale del BUS.
Arrivato all'ottavo impulso di clock, il contatore avviserà il PIPO (74LS273) che il dato è pronto per essere collegato al BUS:
Ecco la tabella di verità del 74LS273:

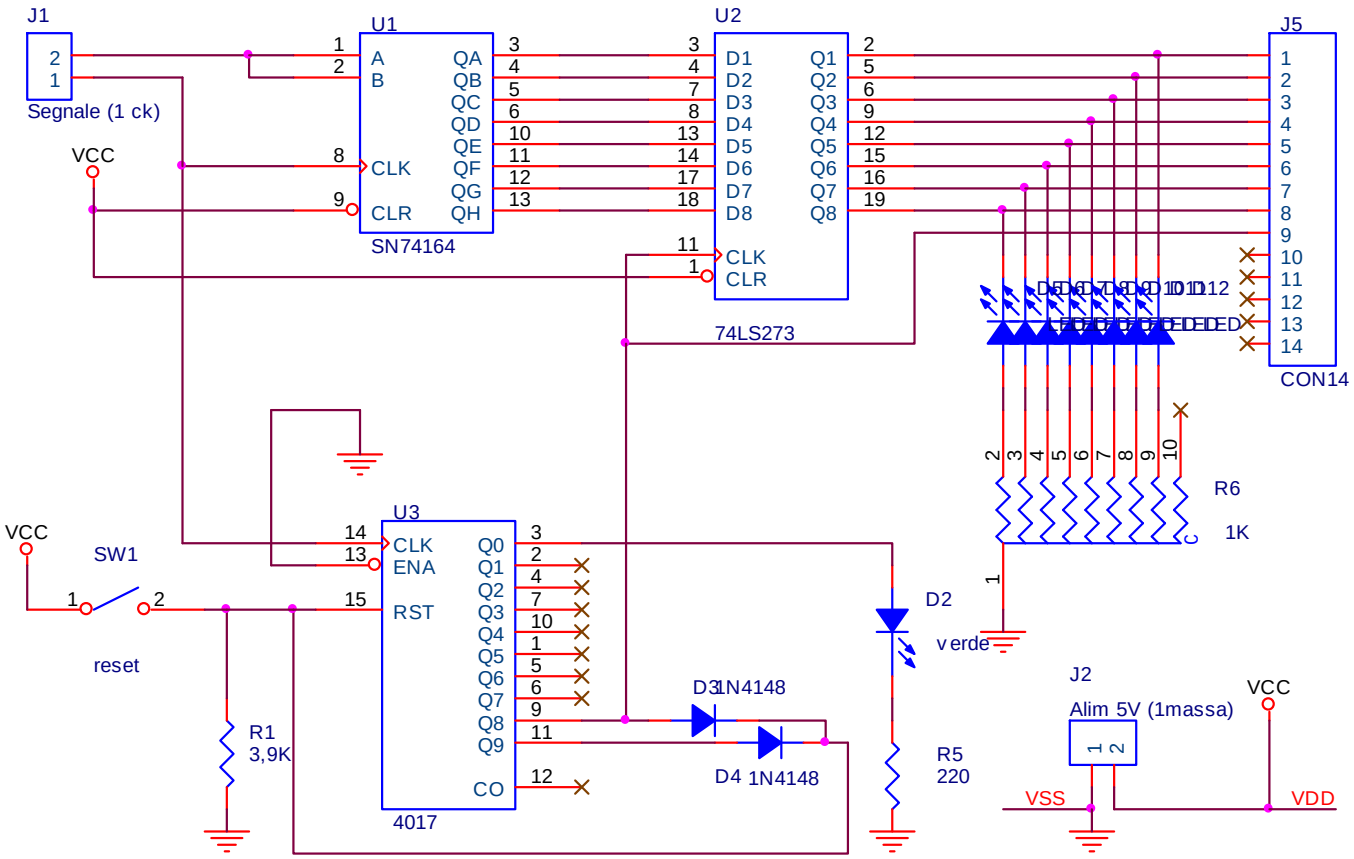
Inputs			Outputs
Clear	Clock	D	Q
L	X	X	L
H	↑	H	H
H	↑	L	L
H	L	X	Q_0

Questo integrato, come detto prima, manterrò in memoria il dato acquisito, quando il livello logico del clock e basso, quando invece è alto, carica ciò che viene letti in ingresso (D).
Questo impulso, brevissimo, verrà generato dal contatore 4017.

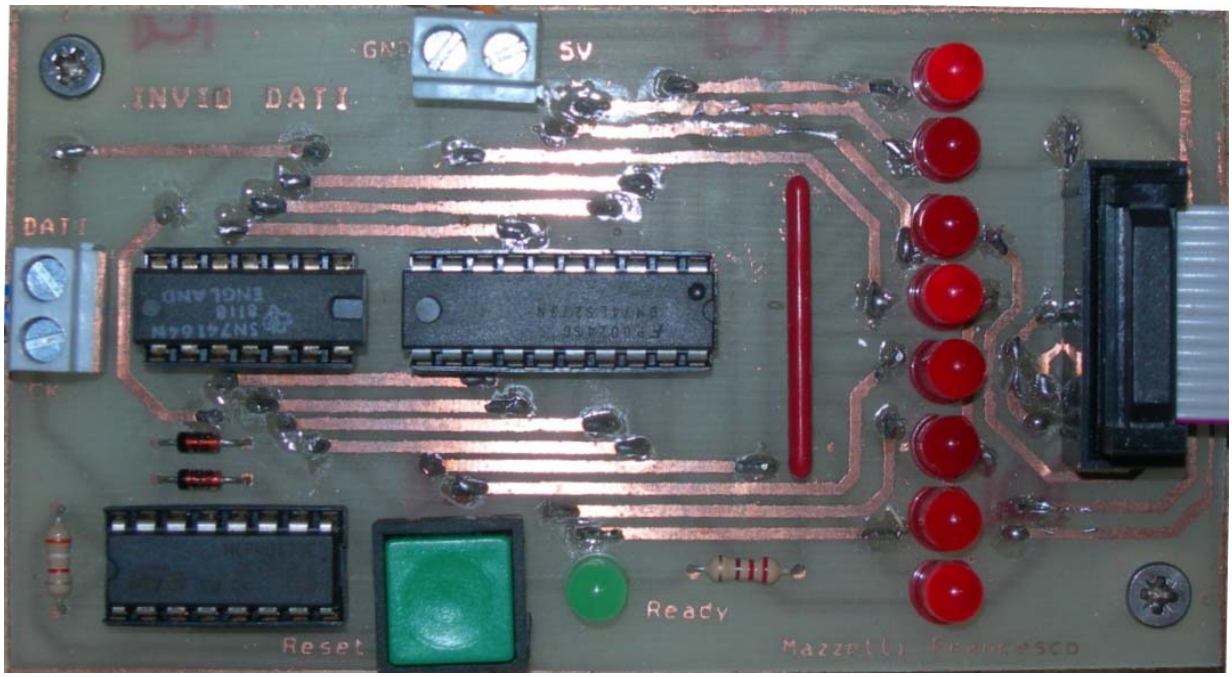


A destra è possibile osservare il semplice schema elettrico del 74LS273, che possiede ben 20 piedini. Un uscita, oltre al collegamento al BUS, sono stati connessi dei LED con una matrice di resistenze. Tale matrice possiede una resistenza più alta del dovuto, così da non rischiare che la troppa caduta sul LED e sulla resistenza, vadano ad intaccare il valore di tensione letto in ingresso.
Questa serie di LED, ritorna molto utile in fase di test, e rende leggibile in binario il carattere inviato.
Nel circuito è stato necessario inserire un pulsante che forza il reset al contatore (e un LED che verifica che il contatore sia in stato di attesa byte), in quanto, disturbi nei canali di clock, o semplicemente lo scambio di selezione dal blocco "selezione uscite" creavano dei fronti di clock, facendo partire il contatore che considerava il primo bit già letto, portando a scombinare tutta la serie di successivi byte di un bit.
Questo tasto poteva venire scavalcato, così da rendere il circuito completamente indipendente dall'operatore, collegando un timer (monostabile) retriggerabile, che controlli il tempo dall'ultimo impulso di clock ricevuto, e agisca sul reset fosse superiore ad una certa soglia.
Questa possibilità non è stata attuata per non complicare ulteriormente il circuito, che già si presentava di difficile realizzazione.

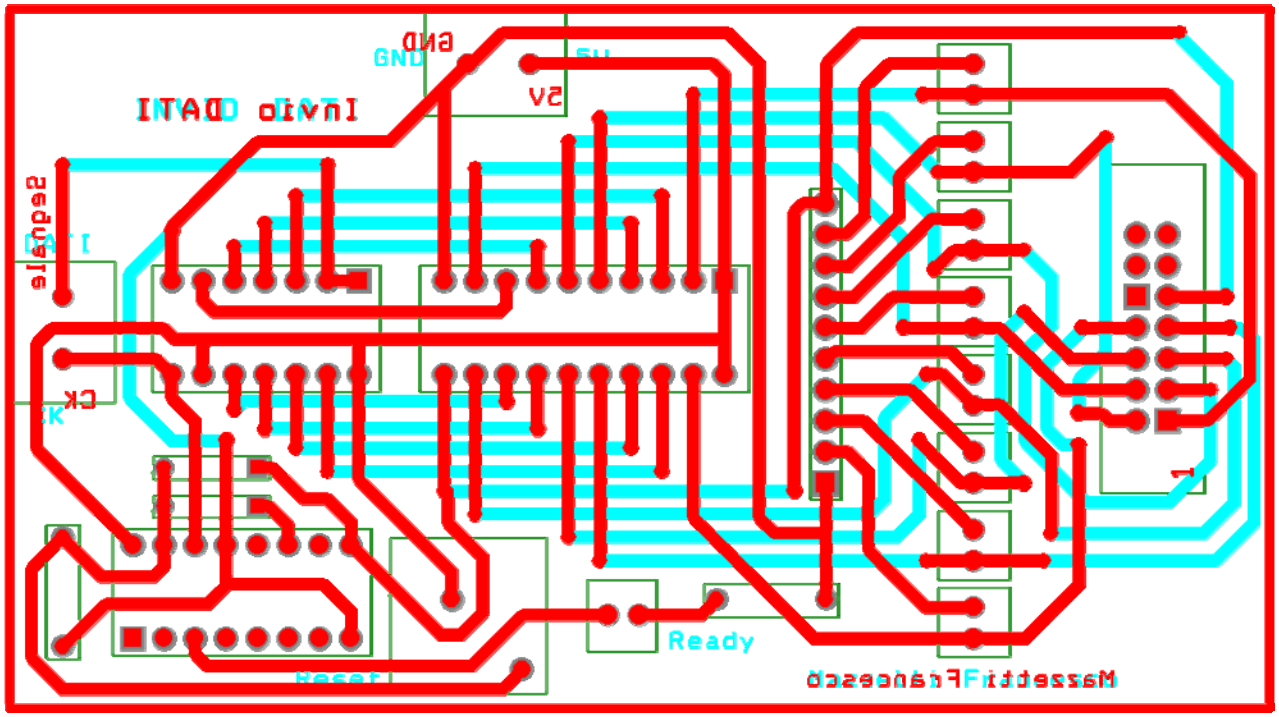
Ecco lo schema elettrico:



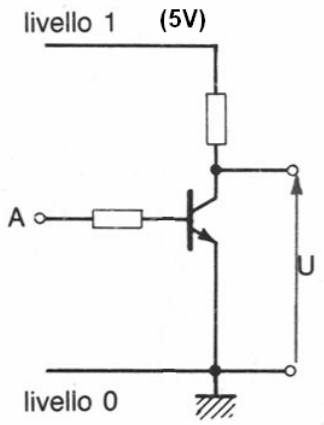
Come si può vedere dalle seguenti foto, questo è il secondo, ed ultimo circuito doppia faccia. Le saldature da uno strato all'altro sono state fatte con un filo di stagno passante e una saldatura da entrambe le parti. La striscia rossa sottile è la matrice di resistenze. Cioè, da un capo (massa) a tutti gli altri, resiste in modo uguale (1KΩ). Gli otto led rappresentano, come detto prima, gli otto bit del byte inviato, controllando il numero di LED accesi, si potrà verificare la loro parità, in quanto, come spiegato precedentemente, il PIC controlla gli "1" del byte ed assegna di conseguenza all'ottavo bit (quello più in alto) un valore attuo a pareggiare essi stesi.



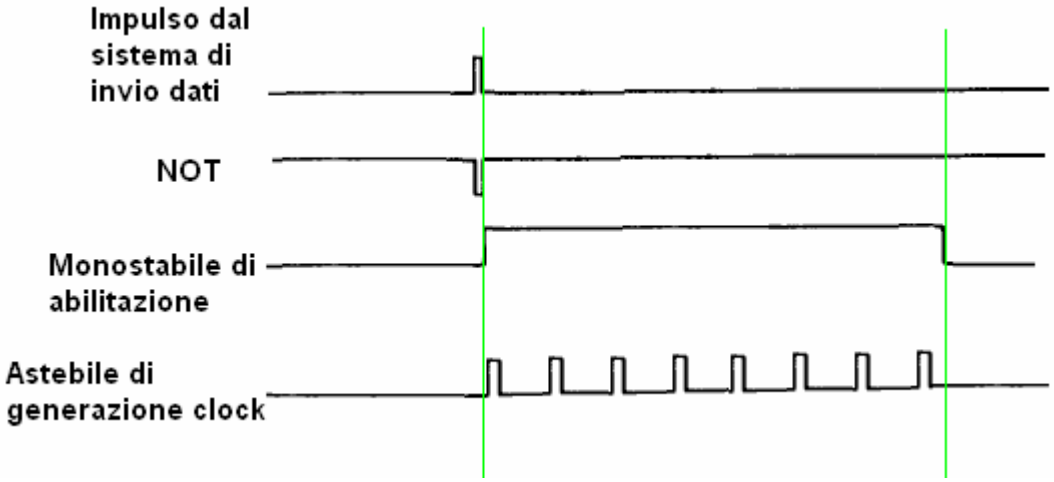
Ecco il lati rame:
In azzurro il alto superiore (quello componenti) e in rosso quello inferiore (saldature)



Ricezione dati Parallelo:
Questo è il blocco che si contrappone in ricezione al circuito visto precedentemente.
Il suo compito è piuttosto semplice: riconvertire il segnale in parallelo in uno seriale e rigenerare il clock.
Proprio quest'ultima parte si è rivelata più complessa, soprattutto nella fase di collaudo, in cui era necessario tarare i due trimmer per ricostruire il clock in modo ottimale.
Per rigenerare il clock sono stati utilizzati due NE555, uno in configurazione monostrabile (visto precedentemente) e l'altro in configurazione astabile, cioè capace di generare un onda rettangolare con duty-cicle regolabile (la regolazione avverrà attraverso un trimmer).
In poche parole, il monostabile riceverà il trigger (impulso di clock) dal sistema di invio, affermando che il bus dati contiene un nuovo byte. Questo manterrà abilitato (per un certo periodo regolabile di circa 150ms) l'astabile, che genererà un nuovo treno di impulsi (cercando di mantenere la forma originaria.
Considerando che ogni bit, possiede un periodo di 20ms, il byte completo è quindi 160ms, ho scelto una rigenerazione più rapida per permettere di ricevere treni di byte consecutivi.
Un problema non da poco, è la necessità di avere un trigger attivo basso, mentre il nostro impulso proveniente dal sistema trasmittente, sarà attivo alto, proprio per questo ho inserito NOT realizzata con un transistor e due resistenze (in questo modo si riesce a evitare l'uso di un integrato molto più ingombrante).
La configurazione della NOT è rappresentato in figura:
Entrambe le resistenze sono state scelte da 1KΩ.
Nella base, per far passare sufficiente corrente da mandare in saturazione il transistor, e sul collettore, sufficiente a non far passare troppa corrente sulla stessa resistenza in coso di conduzione, e nel caso di non-conduzione far riconoscere in uscita la tensione corretta.
Cosa avviene: quando al capo A, viene posta una tensione di 5V, la base viene eccitata, quindi si crea come un collegamento tra l'emettitore (dove c'è la freccia) e il collettore. Quindi al capo U, si avrà una tensione di circa 0V, esattamente il negato logico di 5V.
Mentre se sul capo a non viene applicata tensione (0V), la base non sarà eccitata e il transistor non metterà in conduzione emettitore-collettore. Questo fa si, che prevalga la tensione al capo della resistenza così da avere 5V (il negato di 0V).

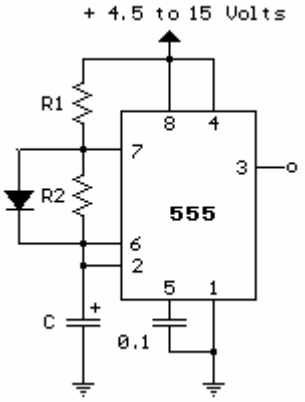


Nel grafico seguente viene simulato l'arrivo dell'impulso di clock, senza tenere conto dei dato.



Sia il tempo del monostabile che la forma del clock, saranno regolabili tramite trimmer

Per migliorare le conoscenze riguardo la configurazione di astabile per l'integrato 555, e quindi comprende al meglio l'utilizzo di certe resistenze piuttosto che altre, trascriverò alcune formule semplificate, utilizzate nei calcoli, tali formula sono in questo caso vere in linea teorica, ma leggermente differenti nella pratica a causa dalla caduta del diodo, necessario per rendere completamente regolabile il duty-cycle (senza diodo sarebbe regolabile solo al di sopra del 50%, quindi non si sarebbe riusciti a regolare i piccoli impulsi di clock a circa un terzo del periodo, come verrà poi spiegato e approfondito successivamente nel paragrafo *tecnica trasmissione*).



In questa configurazione:

$T_{ON} = 0{,}693 \cdot R_1 \cdot C$

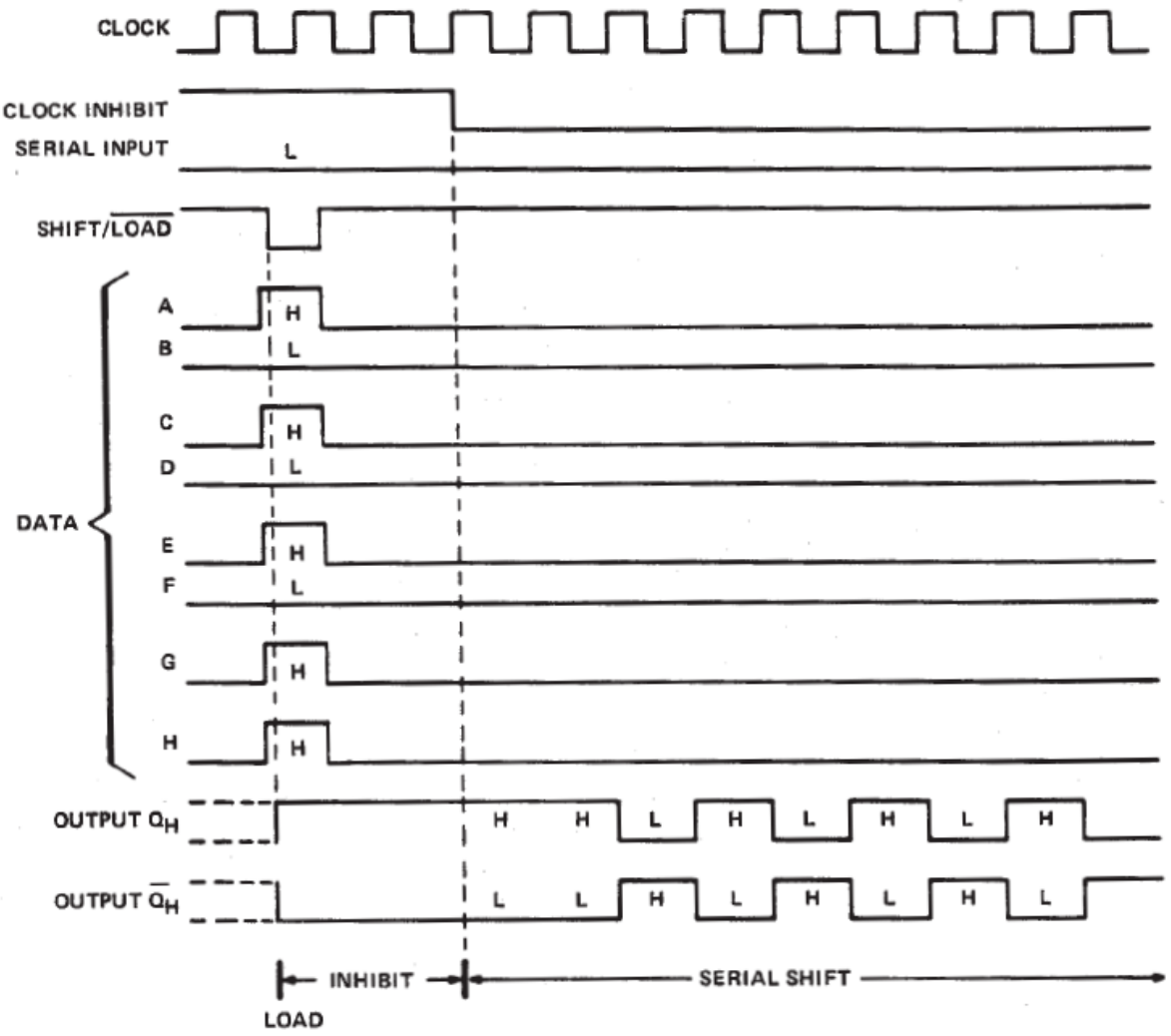
$T_{OFF} = 0{,}693 \cdot R_2 \cdot C$

$T = 0{,}693 \cdot (R_1 + R_2) \cdot C$

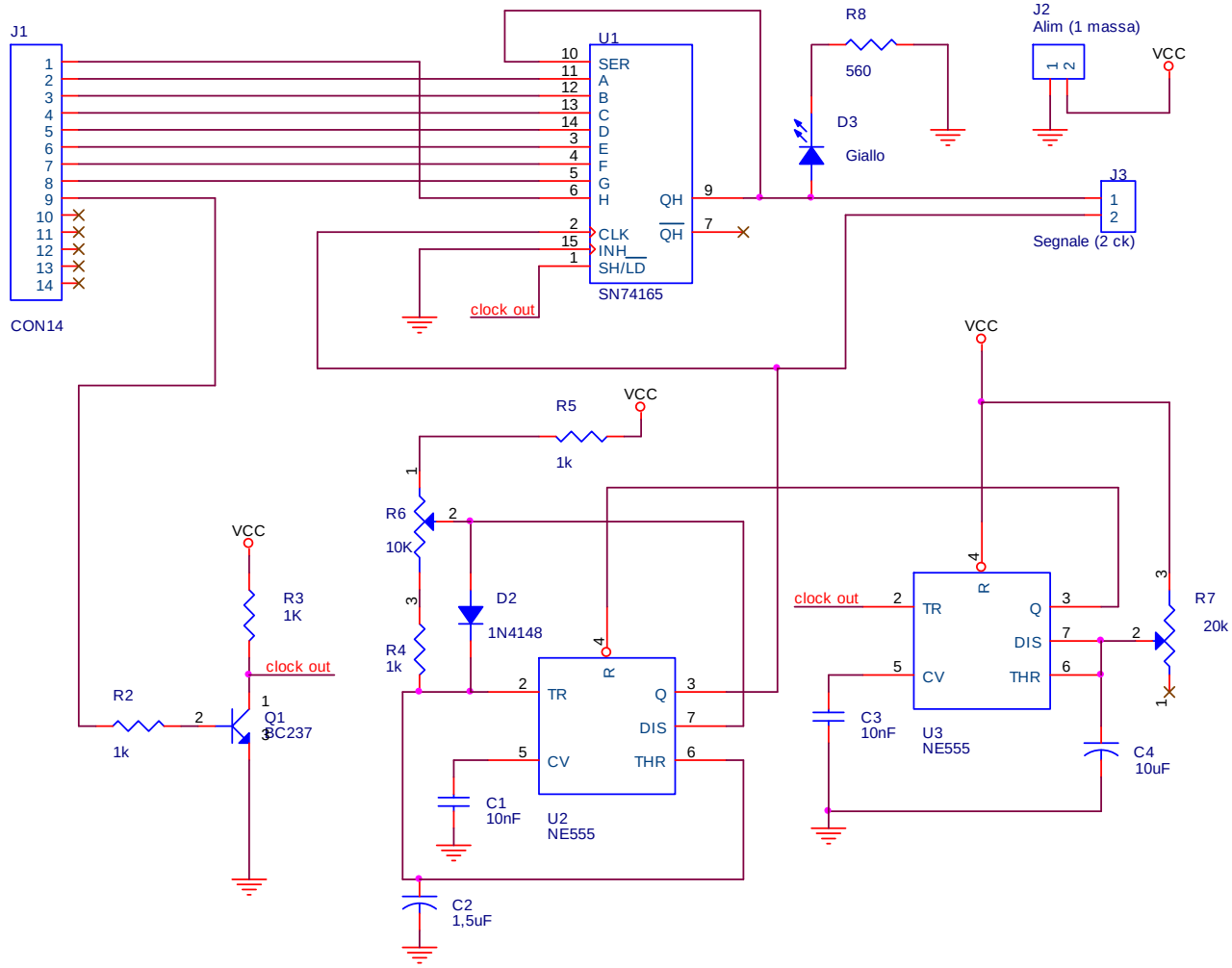
Fissando C a 2,2uF

E cercando di ottenere un periodo poco sotto i 20ms (tempo di trasmissione normale), ho ottenuto la somma delle due resistenze uguale a 12KΩ. Ho fissato due resistenze da un 1KΩ una dalla parte di R_2 e una dalla parte di R_1. In mezzo ho poi posto un trimmer da 10KΩ, cosi da potere modificare e perfezionare on-board il duty-cycle.

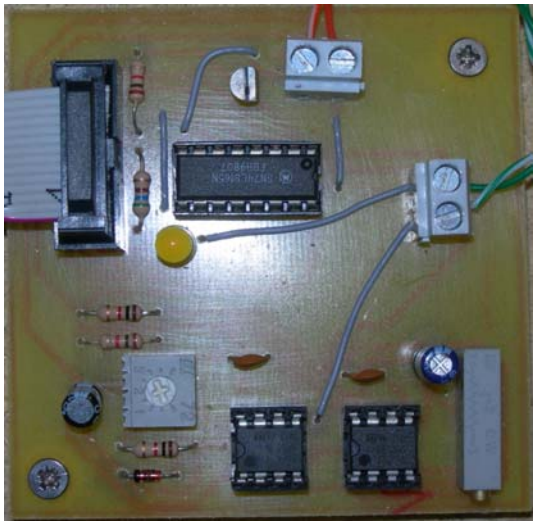
Di seguito un grafico che spiega il funzionamento del PISO utilizzato (74LS165).

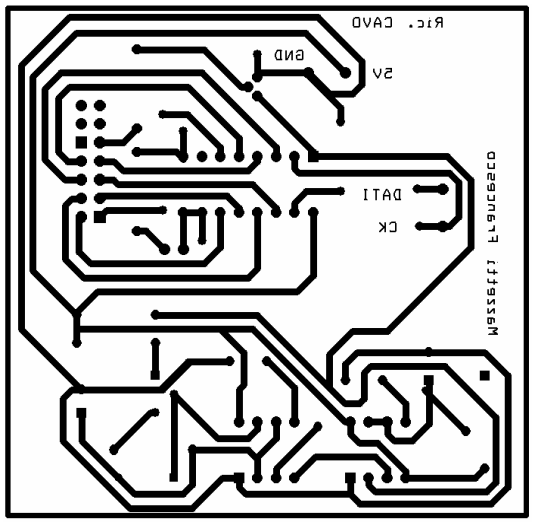


Nel mio caso diventa indispensabile questo accorgimento: una volta caricato il dato (con lo stesso impulso della NOT) al primo impulso di clock generato dall'astabile, il sistema di ricezione non rileverà il primo bit, ma bensì il secondo, difatti in seguito al primo fronte, il PISO, cambierà subito posizione. Per ovviare a questo, come da figura, ho fatto slittare i collegamenti di una posizione, ricollegando l'uscita al PIN 10, che permette di reinserire i dati in modo seriale. Questo vuol dire, che il primo bit sarà in posizione 2, ma verrà letto subito dopo il primo impulso, mentre l'ultimo bit sarà in posizione 1, ma verrà letto per ultimo grazie alla "retroazione" creatagli appositamente.



Il LED giallo viene utilizzato per vedere il dato rigenerato in seriale, utile per lo più per i test.
Nel connettore, com'è visibile, il PIN 9 è quello utilizzato per inviare l'impulso di "dato pronto".
Ecco il lato rame e la foto del circuito:





Codificatore AM:

Per quanto riguarda la trasmissione dati via radio, venivano a crearsi seri problemi nella trasmissione del clock. Come fa il sistema di ricezione a intuire l'arrivo di un nuovo bit, o come fa a capire se quello che si è appena ricevuto è un unico bit o due dello stesso valore?

Esistono nell'ambito delle telecomunicazioni molti metodi (utilizzati nella telefonia via cavo) per trasmettere l'informazione del clock insieme al segnale, nessuno di questi (almeno di quelli analizzati durante l'anno" si adattava ad un trasmissione via radio. In più questi sistemi consistevano in complesse operazioni che dovevano essere compiute attua a modificare il segnale, ciò avrebbe enormemente appesantito i circuiti.

Così ho provato ad immaginare un metodo semplice e alternativo (anche se non universalmente standard) capace di inviare l'informazione del segnale e del clock su un'unica portante.

Proverò ora a ricostruire il mio ragionamento:

Avendo due segnali, potrei avere più stati:

segnale	clock	N° stati
0	0	*1*
0	1	*2*
1	0	*3*
1	1	*4*

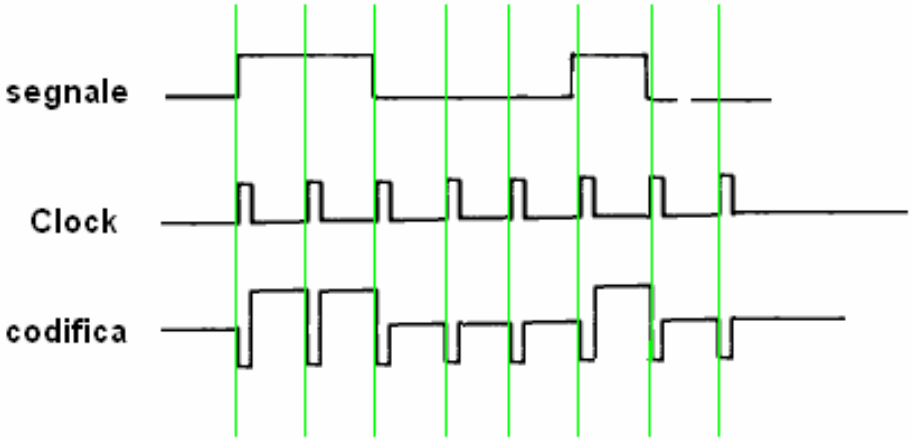
Ho poi notato che il quarto stato (1,1) era scavalcabile, in quanto il sistema di ricezione, prevede di aspettare che l'impulso di clock termini per controllare lo stato del segnale, quindi diventa uno stato eliminabile.

Ho quindi pensato di considerare il livello logico 0-0, ad una certa tensione (ipotizziamo 0V) a segnale alto, portare l'uscita a 5V, e ad impulso di clock, portare l'uscita a -5V(in questo caso è tornata utile la tensione regolabile prevista dal mio alimentatore). Tutto questo deve poi essere modulato e inviato dall'antenna.

Questo è il punto in linea teorica, in linea pratica, sono molto più complessi tutti i passaggi.

Ho deciso di dividere il blocco codificatore da quello modulatore per semplificare i circuiti stessi, per semplificare le fasi di test, e anche perché si possa ben distingue il compito del codificatore (che ha due segnali digitali in ingresso e ne da in uscita uno analogico) e il modulatore che deve modificare l'ampiezza della portante simultaneamente al segnale, amplificarlo in potenza per essere trasmesso "nell'aria".

In questo grafico è possibile vedere come deve lavorare il codificatore.



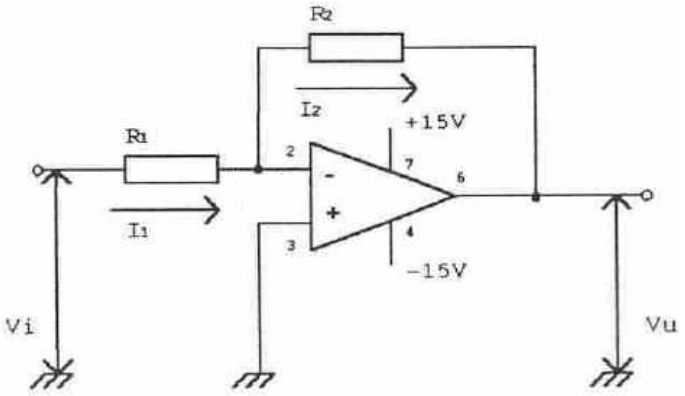
Il breve impulso di clock, porterà (come detto prima) l'uscita negativa per tutta la sua durata, per il resto l'uscita sarà uguale al segnale.

Proprio per questo ho scelto di utilizzare un MUX analogico (4051). Collegando il clock al selettore, in modo che, quando il clock è a zero, passi il segnale, quando invece il clock possiede valore logico "1", verrà abilitato un ingresso a -5V (proveniente direttamente dall'alimentatore.

Per garantite una maggiore flessibilità in fase di collaudo e di realizzazione (questo circuito è stato ideato e realizzato molto prima di avere idee certe sul modulatore utilizzato), ho pensato di rendere tale segnale completamente regolabile in ampiezza (sia attenuando che amplificando) che in tensione, cioè permettendogli di lavorare con un DC-offset completamente regolabile. Ciò si è reso possibile tramite un operazionali utilizzati in configurazione "amplificatore invertente" e "sommatore"
In entrambi i casi il segnale viene invertito, cosicché al termine di questi passaggi, il segnale tornasse regolare.
Breve formulario dei suddetti operazionali, nelle due configurazioni utilizzate, più un'altra adoperata più avanti.
Amplificatore invertente:

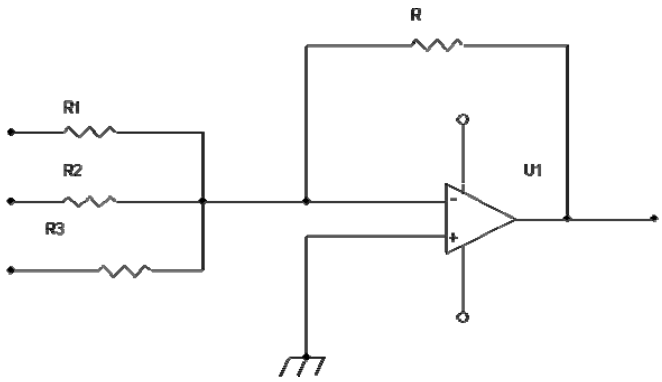
$$Vo = -\frac{R_2}{R_1} \cdot Vi$$

Questo vuol dire che fissando la R_1, modificando con un trimmer la R_2 si potrà amplificare o attenuare il segnale a proprio piacimento (invertendolo).
(le resistenze in gioco, come in tutti gli operazionali, devono essere comprese tra 1KΩ e 1MΩ)

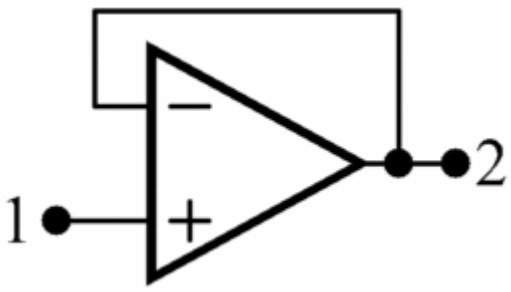


Sommatore:
Tale circuito effettuerà la somma tra i vari ingressi, e restituirà in uscita il risultato di tale operazione aritmetica invertita di segno. Per avere una somma corretta, la R deve essere uguale a R_1, in caso diverso, il segnale può venire attenuato o amplificato.

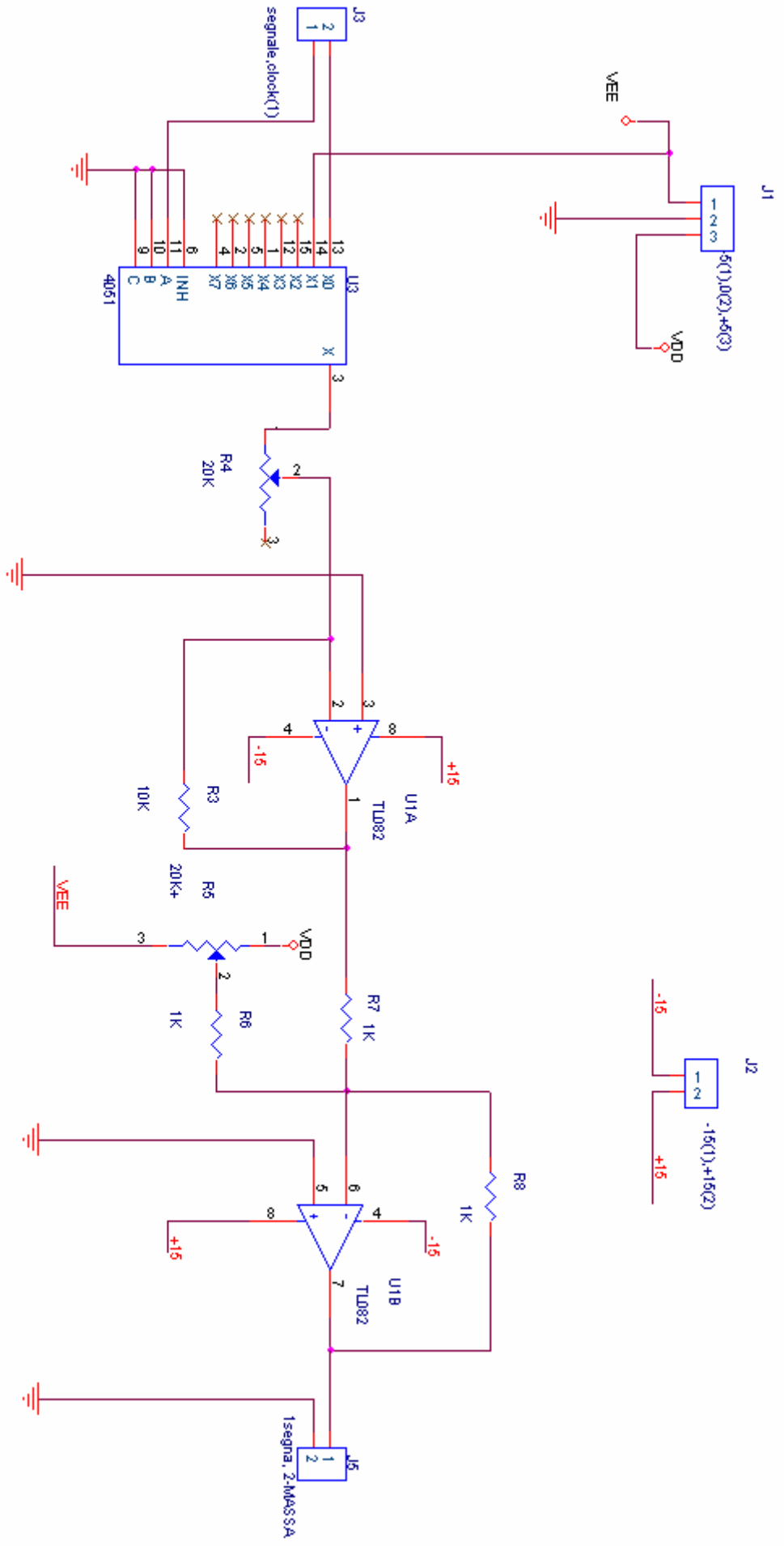
$$Vo = -\frac{R}{R1} \cdot (V_1 + V_2 + V_3)$$



Inseguitore:
Il suo scopo è quello di restituire in uscita, esattamente la stessa tensione presente in ingresso. Nonostante possa sembrare inutile possiede molte applicazioni, soprattutto laddove, non conoscendo la fonte, si voglia rigenerare il segnale senza rischiare di far assorbire troppa corrente al circuito precedente. Difatti la corrente erogata in uscita dall'operazione (come in tutte i suoi utilizzi) viene "prelevata" dall'integrato stesso, e difatti è caratteristica rigida dei singoli componenti.

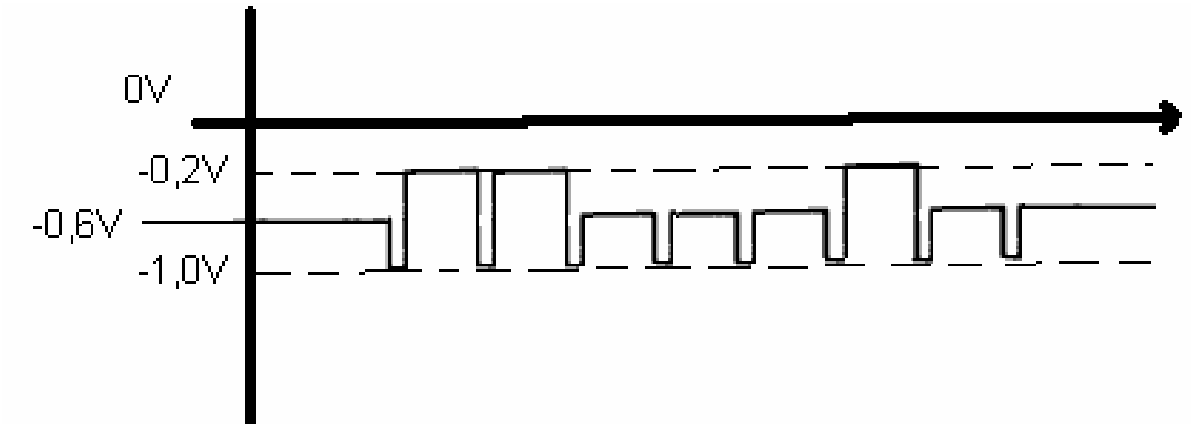


Di seguito lo schema elettrico del circuito:


J3
segnale,clock(1)
VEE
J1
5(1),0(2),+5(3)
VDD
U3
4051
INH
A
B
C
X0
X1
X2
X3
X4
X5
X6
X7
R4
20K
R3
10K
R5
20K+
R6
1K
R7 1K
R8
1K
U1A
TL082
U1B
TL082
-15
+15
J2
-15(1),+15(2)
J5
1segna, 2-MASSA

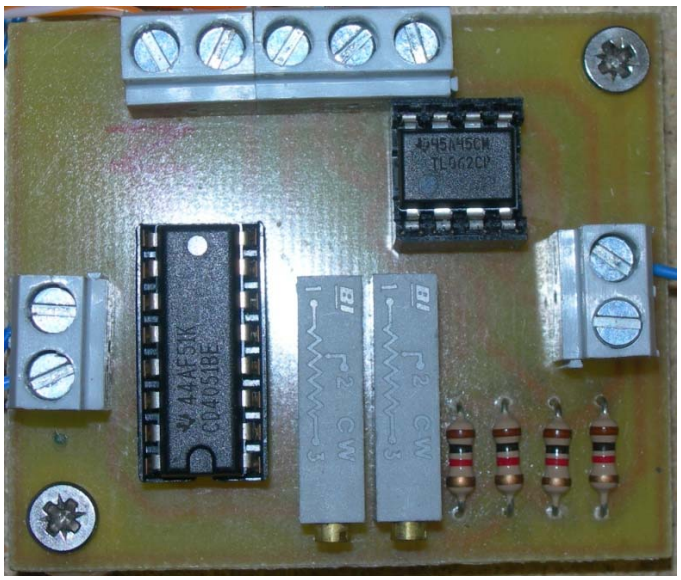
Nello schema è possibile vedere meglio la configurazione del MUX, avendo tre ingressi di selezione, oltre al primo controllato dal clock (A) gli altri pin vengono posti a massa per permettere di selezionare unicamente i primi due ingressi. In questo modo, vengono sprecati i restanti input, perdita obbligata in quanto non sono riuscito a trovare integrati meno ingombranti con le stesse caratteristiche analogiche (anche negative).
Le resistenze da 1KΩ inserite servono a non permettere ai trimmer di scendere sotto questo valore di resistenza, uscendo dall'area di lavoro raccomandata dal costruttore.

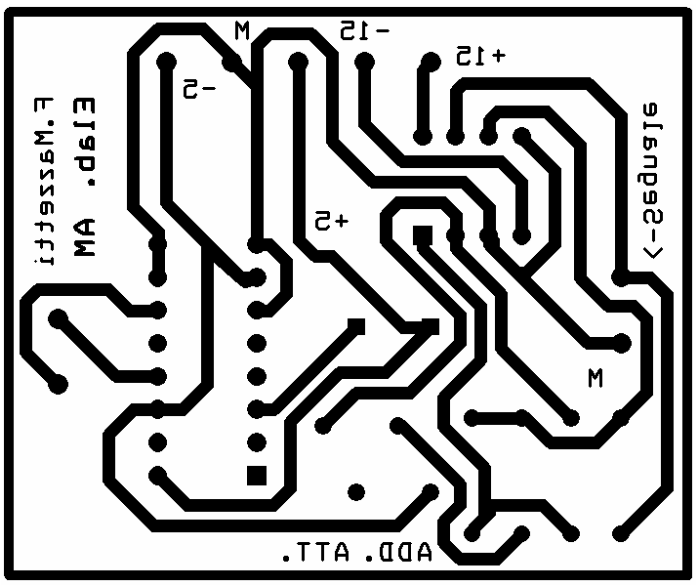
Per ottenere il maggiore guadagno in fase di modulazione, è stato verificato, che all'impulso di clock, la tensione vada a circa -1V, in fase neutra a -0,6V e a fase di uno logico a -0,2V



Dalla foto è possibile notare i due trimmer multigiro ad alta precisione e i ben 5 connettori per l'alimentazione: ±5V, ±15 e massa.

Ed ecco il lato rame

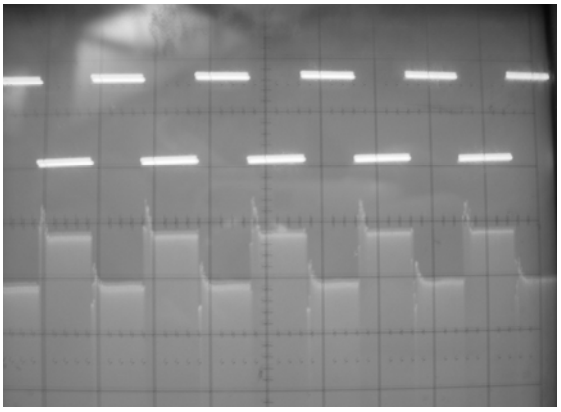




Modulatore AM:

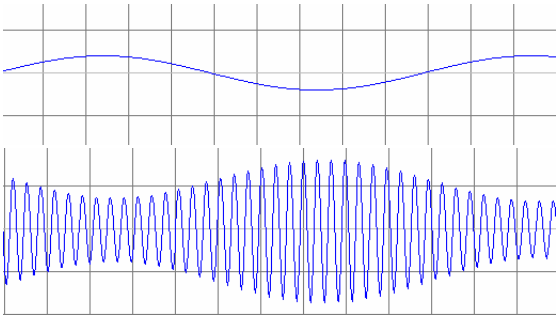
Questo circuito è stato tra gli ultimi realizzati, proprio per questo si presenta con delle piccole modifiche effettuate dopo la stampa del circuito.

Il trincio della modulazione d'ampiezza è semplice, avendo una portante ad alta frequenza (generalmente un'onda sinusoidale), riuscire a modificare la sua ampiezza in modo coerente rispetto al segnale da modulare



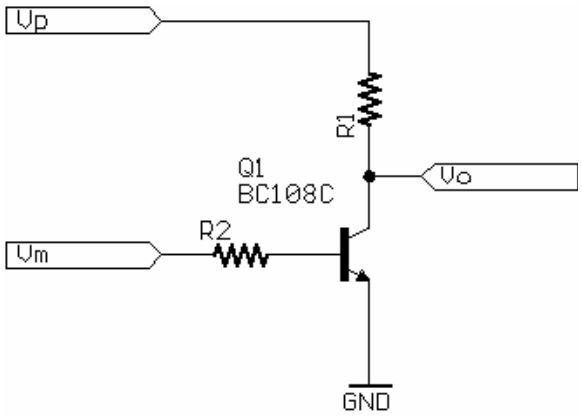
segnale

portante modulata



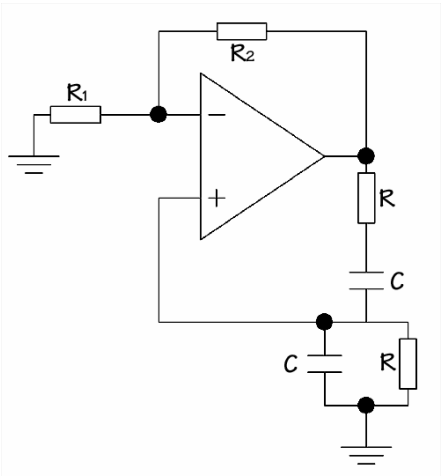
La foto presa dall'oscilloscopio è quella reale sull'antenna del mio circuito, in fase di test, inserivo un segnale quadro, della frequenza il più possibile vicino a quelle utilizzate da me.

(la portante viene chiamata così proprio perché permette al segnale di viaggiare nell'aria, in quanto è necessario un onda ad alta frequenza per poter trasmettere un segnale elettro-magnetico



Esistono molti tipi di modulatori, ma non tutti permettono una modulazione d'onda rettangolare, io ho deciso di modulatore a transistor (2N2222) in cui viene messo in base il segnale da modulare e sul collettore la portante. Il funzionamento è simile alla porta NOT sopra spiegata, anche se in questo caso non si avrà una cambiamento repentino (da 0 a 5V), ma la corrente presente in base è talmente bassa, da non permettere un collegamento diretto con l'emettitore, questo porta ad una leggera, ma rapida, variazione d'ampiezza della portante. Esattamente ciò che si voleva ottenere.

La portante nel mio caso è un onda sinusoidale di circa 70KHz, realizzata tramite un oscillatore a ponte di wien (sempre tramite operazionali) e la cui ampiezza diventa regolabile attraverso un trimmer.

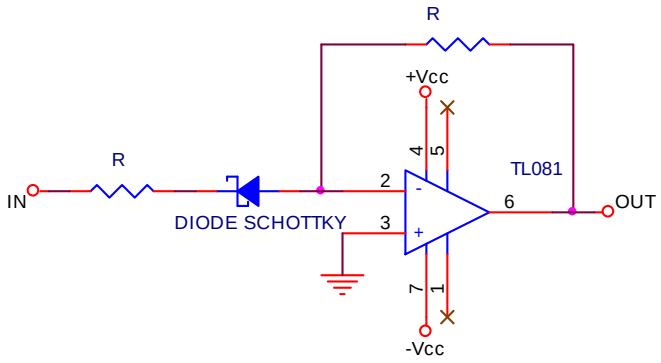


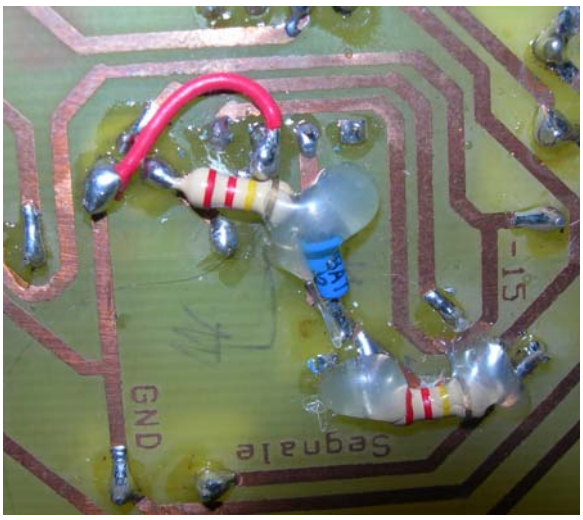
Nel circuito dell'oscillatore a ponte di wien (l'immagine a destra) le due R e C devono essere il più possibili uguali, mentre in teoria $R_2=2R_1$. Ho specificato in teoria, in quanto nella pratica è necessario che la R_2 sia almeno 2,2 volte R_1. Per perfezionare al meglio questa caratteristica, anche in questo caso, ho utilizzato un trimmer per un'ottimale regolazione. Aggiungo inoltre che al momento della taratura, ho deciso di squadrare leggermente l'onda prodotta notando un miglioramento in trasmissione.

$$f_0 = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 22 \cdot 10^3 \cdot 100 \cdot 10^{-12}} \approx 72KHz$$

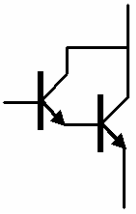
In uscita al modulatore, è stato poi inserito un inseguitore, e un invertitore che taglia il segnale, negativo, questo perché in ingresso al sottoblocco di potenza, era sufficiente portare solamente la parte positiva, anzi, migliorava il segnale inviato.

La parte che inverte il segnale (realizzata con un amplificatore invertente a guadagno 1) è la piccola modifica realizzata sul retro dello stampato, in quanto, dal progetto originale non era stato previsto.
Lo schema per permettere solamente all'onda positiva di passare è il seguente:

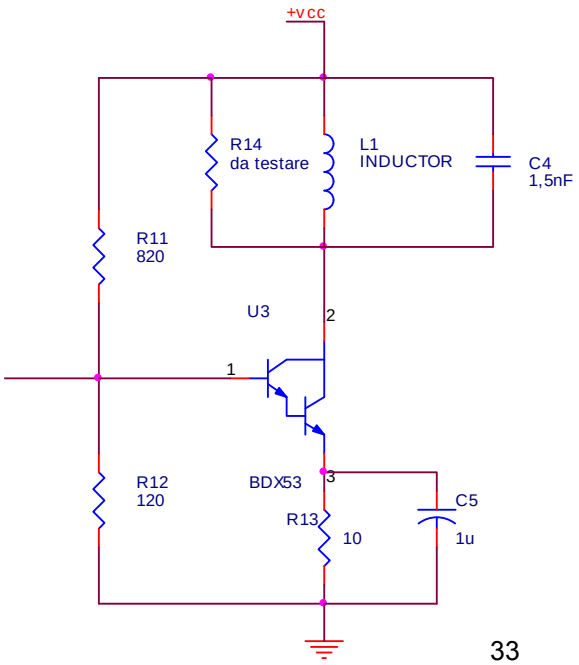




In questo modo nel terminale "–" è presente massa (in quanto in retroazione negativa V+ = V-), riusciranno quindi a sorpassare l'ostacolo del diodo, solamente le tensioni negative, poi rese positive dell'invertente. Il valore delle resistenze è ininfluente, basta che rimanga nell'ordine dei KΩ e che siano uguali (io ho scelto 220KΩ). Come è possibile vedere in figura, ho scelto di utilizzare un diodo schottky, noto per le buone caratteristiche in frequenza, e per avere una caduta di tensione di circa la metà di un diodo comune (0,35V), questo permette di avere ancora più precisione nel taglio dell'onda positiva. La colla posta sopra i componenti è a scopo di protezione contro possibili urti.

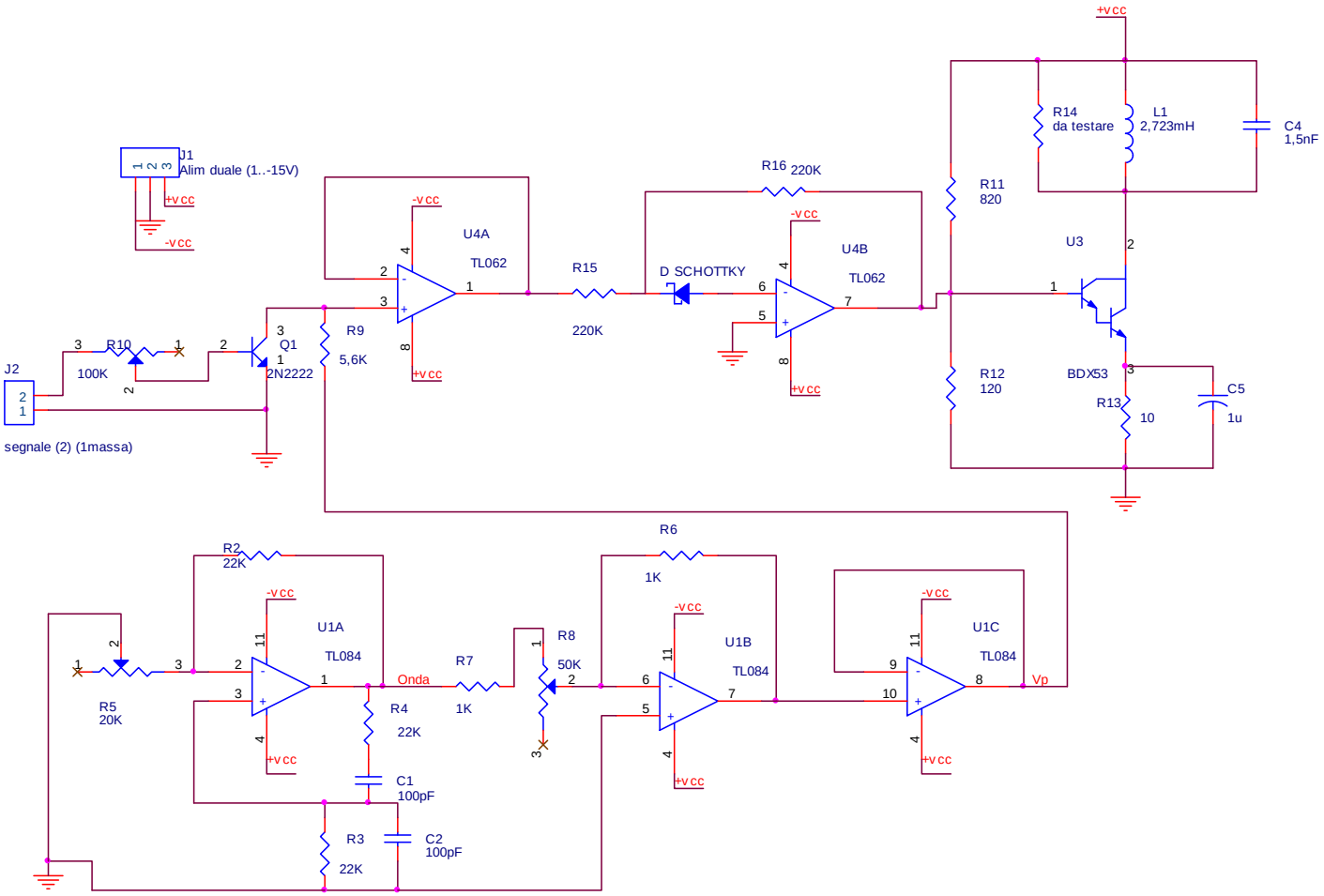


Il blocco che avrà il compito di amplificare ulteriormente il segnale e inviarlo all'antenna è costituito da un transistor in configurazione Darlington. Cioè sono due transistor posti come in fugura.
Questa disposizione interna, permette al transistor di amplificare notevolmente, anche a basse correnti di base. Il che provoca un riscaldamento dello stesso. Proprio per questo il socket (TO220) permette facilmente di essere collegato ad un dissipatore. Il dissipatore da me utilizzato è nettamente sovradimensionato, ma ciò non può fargli che bene.
Ecco lo schema elettrico del blocco di potenza:
Il transistor utilizzato è il BDX53, la R14, in realtà non è mai stata inserita, è stata prevista nel caso che, in fase di collaudo, si fosse rivelato necessario modificare il circuito risonante, che poi è quello che permette all'antenna (che altro non è che una induttanza avvolta in ferrite del valore di 2,723mH, di cui parlerò più avanti).
Il circuito risonante permette il passaggio di una certa frequenza, selezionabile variando il condensatore (nel nostro caso è fisso, in quanto la portante è fissa a 70KHz



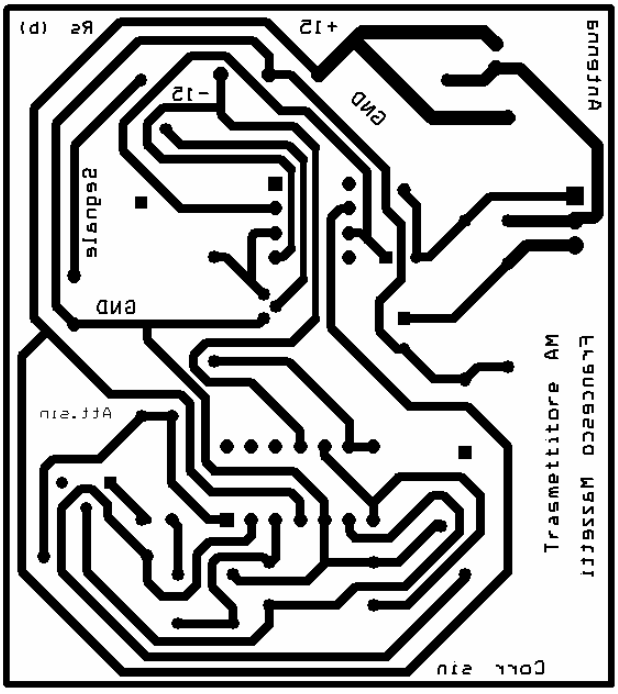
$$f_0 = \frac{1}{2\pi \cdot \sqrt{LC}} \approx 79KHz$$ errore accettabile.

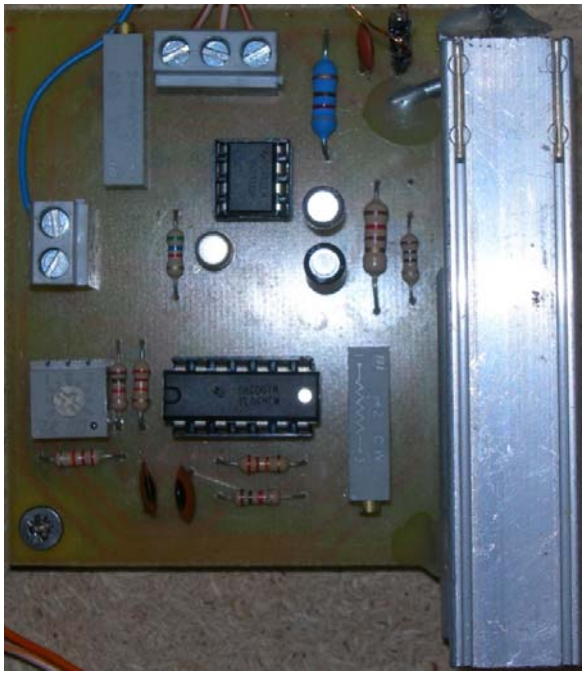
Ecco lo schema elettrico:



In basso è presente tutto il circuito di generazione della portante (ho utilizzato un integrato che contiene 4 operazionali, che erogasse abbastanza corrente, in quanto, i primo utilizzato scaldava leggermente), stessa cosa facevano le due resistenze che vanno in base al BDX53, per cui ho dovuto recuperarne due a ½ di watt.

In questo circuito è già presente la correzione riguardante l'invertente, che permette solo all'onda positiva di passare mentre nel lato rame, non lo è, in quanto la correzione è stata fatta a mano:



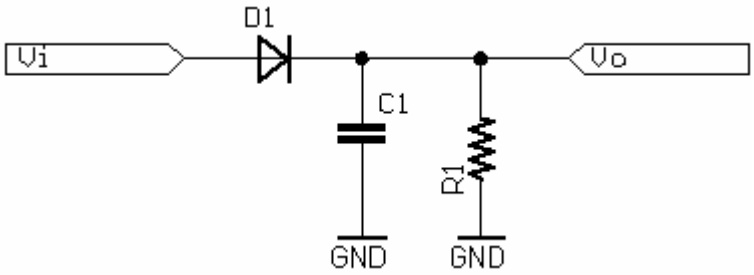


Demodulatore AM:

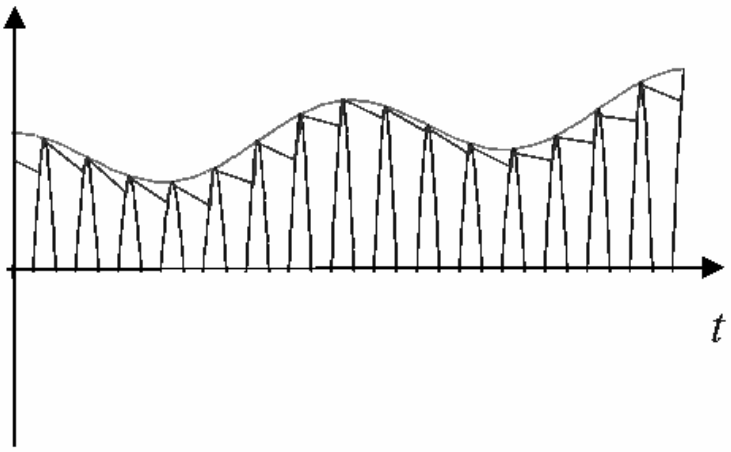
Il principio di funzionamento di una demodulatore è semplice: eliminare la portante dal segnale ricevuto, e ricostruire quindi l'onda originale.

Nel mio caso però le cose si complicano perché, oltre a dover ricevere e demodulare e amplificare, il circuito deve anche riuscire a distingue il clock dal dato.

Il principio del demodulatore AM usato è molto semplice, ed è alla base di tanti altri circuiti che devono svolgere lo stesso compito: non si tratta che di un diodo (schottky) che permette solamente alla parte positiva del segnale di passare (il mio segnale AM è in DSB, cioè presenta sia positivamente che negativamente lo stesso segnale, semplicemente invertito, perciò una delle due parti è inutile).



In seguito al filtro è presente una specie di rete composta da una resistenza e condensatore, che fungono da filtro al segnale, le resistenza serve a permettere al condensatore, in caso di cambio di fronte, di scaricarsi e "acquisire" il nuovo valore. L'immagine seguente chiarisce il lavoro svolto, bisogna però immaginare il segnale rettangolare, e la portante con frequenza molto più alta rispetto al segnale: nel mio caso difatti, la fp è circa 1400 volte più grande del segnale in uscita dal blocco invio dati. Ciò permette di avere una decodifica AM ancora più ottimale.

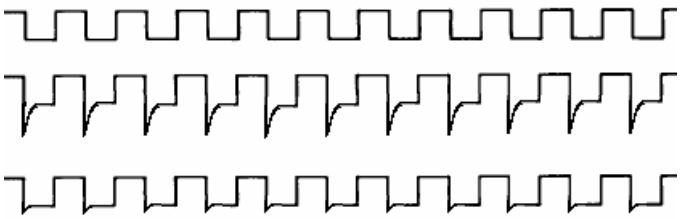


Prima di arrivare a questo sistema, però il segnale in uscita dal circuito risonante (esattamente lo stesso della trasmissione), viene "assicurato" da un due amplificatori, in modo che i circuiti seguenti non incidano su quelli precedenti peggiorando il segnale (antenna).

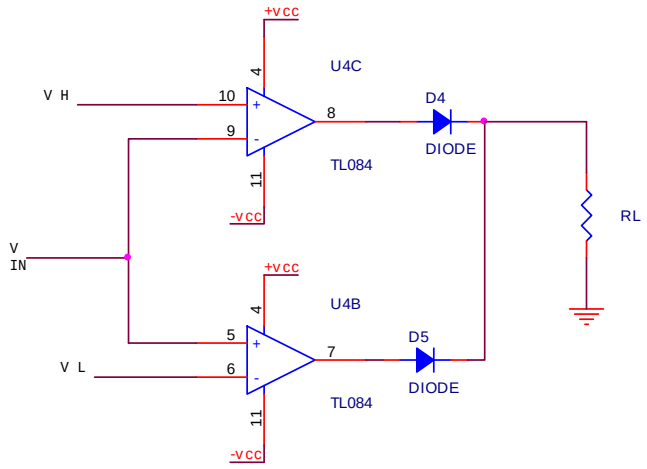
Il primo amplificatore è un semplice invertente a guadagno uno, mentre il secondo ha un trimmer posto sulla R_1, in modo da poter amplificare, o addirittura attenuare il segnale in arrivo.

Una volta demodulato, il segnale si presenta comunque ancora troppo piccolo per essere confrontato, e così si è rivelata necessarie un'ulteriore amplificazione di quattro volte (anche perché il segnale si rileva nuovamente invertito). Aumentando nuovamente l'ampiezza, però, il segnale tenderebbe a uscire dal range di alimentazione degli operazionali (±15V), per questo è necessario inserite un sommatore variabile, tramite trimmer, di modo che regolando ampiezza e amplificazione, così da raggiungere un valore ottimale e di alta fedeltà per il circuito che dovrà separare il segnale dato da quello del clock.

Un ulteriore problema verificato nella fase di collaudo è stato un notevole picco di disturbo che si creava quando l'onda era in fase di discesa. Questo disturbo rischiava di essere confuso con l'impulso negativo di clock. Per risolvere questo ennesimo problema ho inserito una capacità di 10μF in parallelo al segnale con –Vcc. (essendo elettrolitici dovevo essere sicuro di avere un polo sempre negativo rispetto all'altro) in modo da attenuare il disturbo, come da immagine.



Per scindere il segnale dati da quello del clock, ho utilizzato un circuito nato per tutte altre applicazioni, e modificato a mio piacimento. Tale circuito, viene chiamato comparatore a finestra, e permette di dare in uscita +Vcc, quando la tensione in ingresso è oltre a due limiti stabiliti(limite superiore e inferiore, mentre dare tensione 0, quando la tensione "misurata"restava in mezzo a questi due valori imposti.



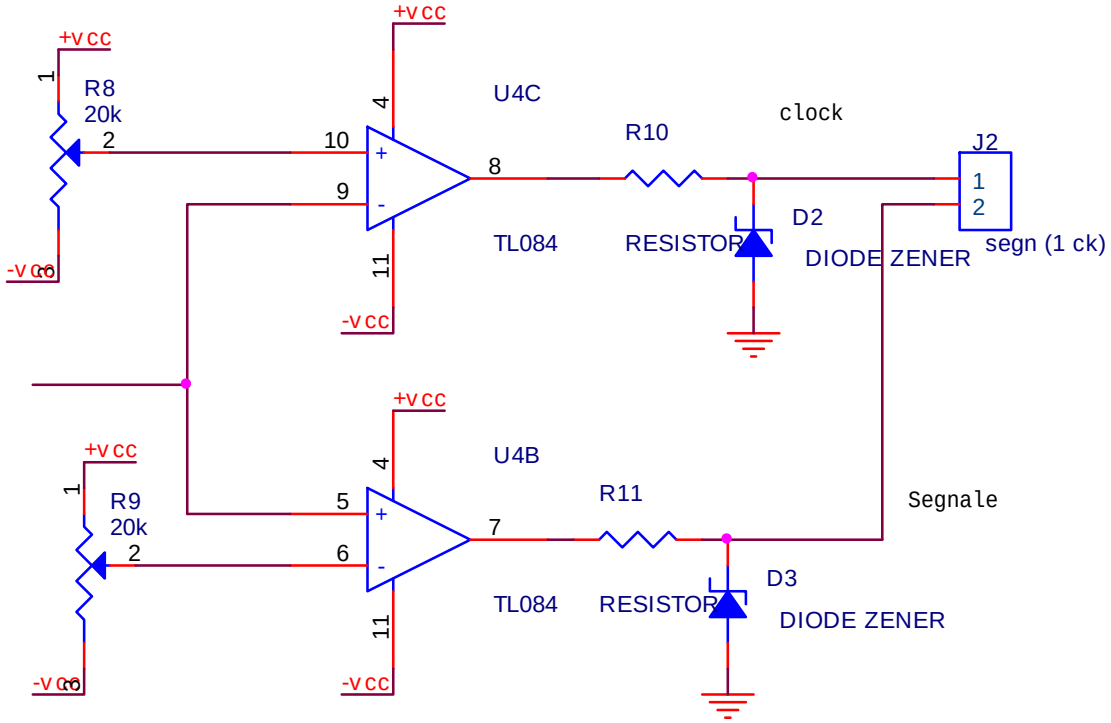
I due valori imposti, sono impostabili ai capi estremi dei due comparatori.
Mentre nel capo centrale ci sarà il segnale da confrontare.

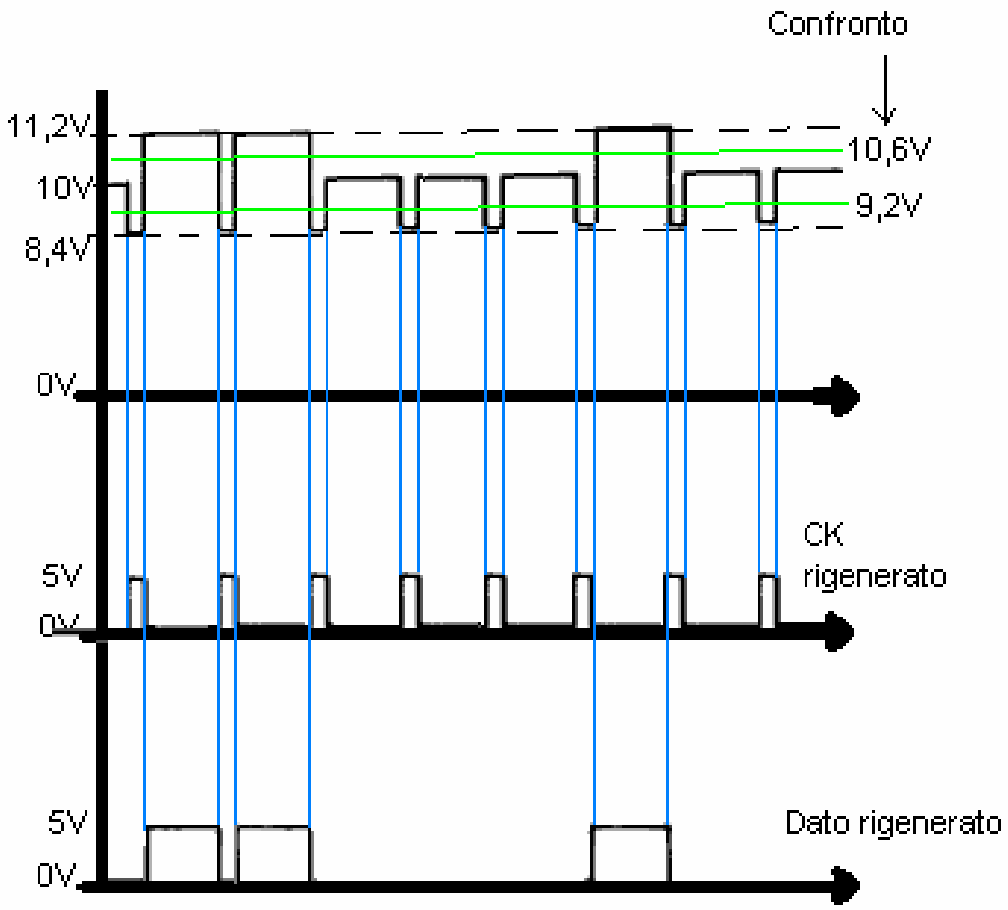
Nel mio caso ho eliminato i due diodi, e quindi eliminato la giunzione, mettendo una configurazione di zener per stabilizzare le tensioni a livello logico TTL, difatti un'uscita corrisponderà al clock, mentre l'altra al segnale dati. L'immagine successiva chiarisce ulteriormente il meccanismo di comparazione.

Immaginiamo (così è) che tensioni relative al segnale demodulate siano le seguenti:
- impulso di clock →≈ - 4,2V
- 0 logico →≈ 0,1V (regolato tramite il trimmer)
- 1 logico →≈ 6,2V

I trimmer sono stati regolati in modo di avere queste tensioni in quanto mi sono reso conto che con questa configurazione, il distacco tra le tensioni tre tensioni era il più rimarcato.

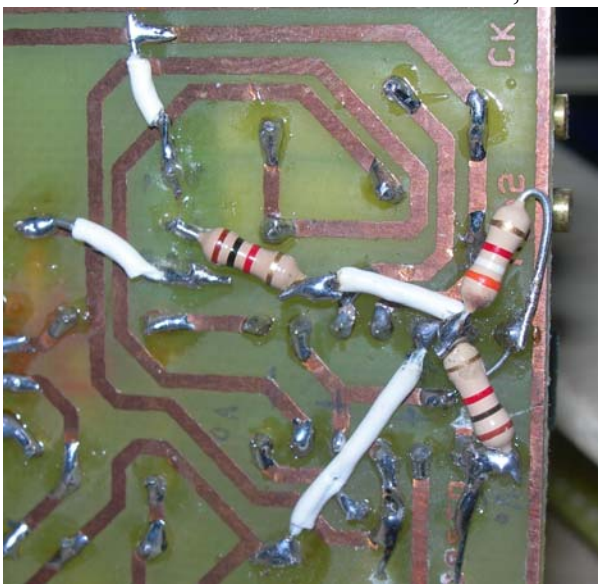
Ora bisogna riuscire ad impostare le due tensioni "limite" circa in metà tra il margine di zero logico e il relativo opposto.
Per esempio il limite inferiore utile per rilevare il clock sarà di circa -2V, mentre quello superiore per rilevare il valore logico alto sarà circa 3V
Il grafico spiega meglio il lavoro svolto dal seguente circuito:





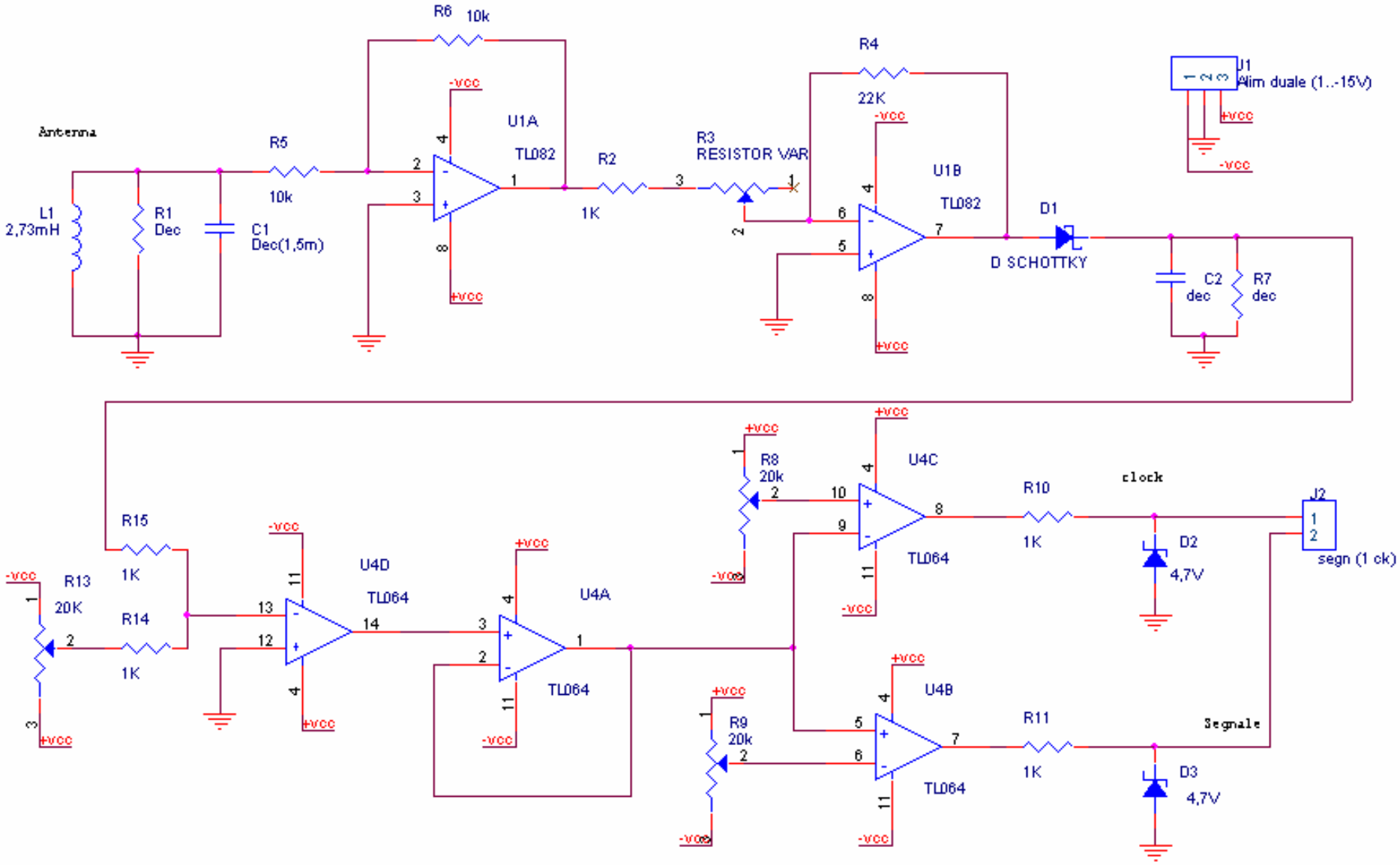
I valori del grafico non sono corretti e non sono in scala, questo perché alla creazione del grafico venivano utilizzati tensioni regolate diversamente. Nonostante questo il grafico evidenzia bene la struttura e il funzionamento del *rigeneratore segnale*

Si vede bene, la teorica perfetta rigenerazione del segnale e del clock. Bisogna notare che inizialmente non erano presenti quei "picchi" negativi sull'asse del *dato* in corrispondenza del clock (nel grafico al secondo impulso), però è una conseguenza obbligatoria dell'eliminazione del quarto stato analizzato nel blocco codifica AM, e comunque ininfluente ai termini di corretta ricezione.

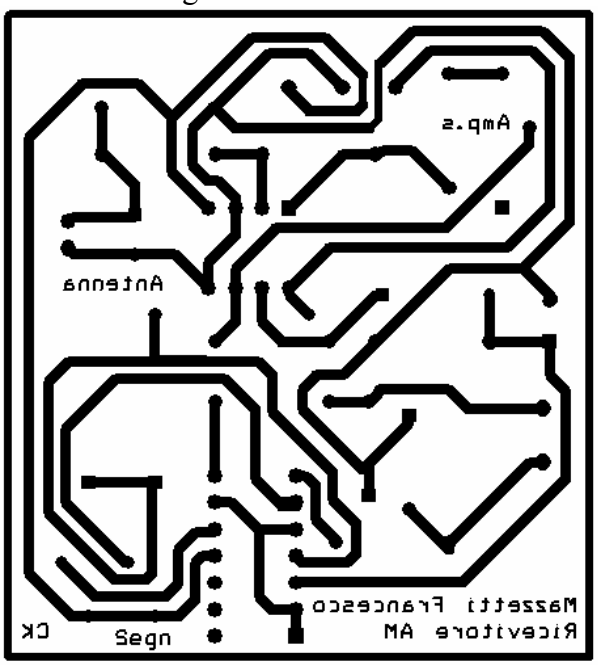


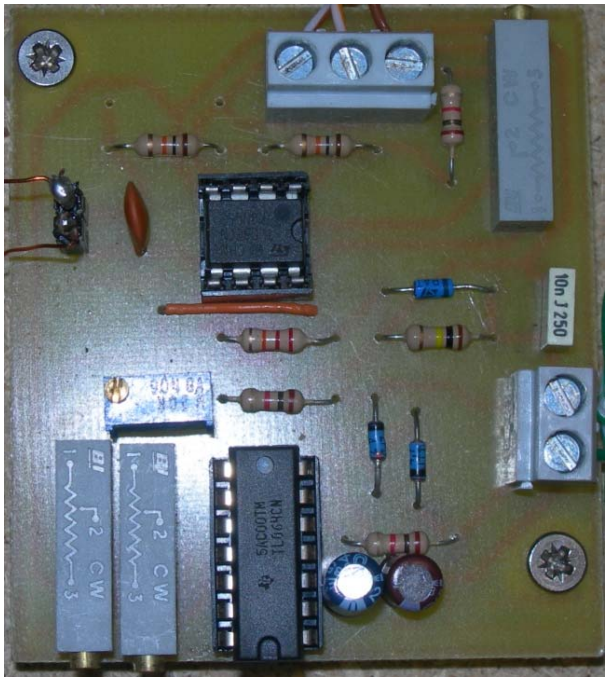
Questa è la foto della modifica effettuata sul retro della scheda, il trimmer e il condensatori aggiunti sono stati inseriti sulla parte superiore con fori sul circuito.

Di seguito il circuito, con le correzioni eseguite manualmente:


Antenna
L1
2,73mH
R1
Dec
C1
Dec(1,5m)
R5
10k
R6
10k
U1A
TL082
R2
1K
R3
RESISTOR VAR
R4
22K
U1B
TL082
D1
D SCHOTTKY
C2
dec
R7
dec
U1
Alim duale (1..-15V)
-vcc
+vcc
R15
1K
R13
20K
R14
1K
U4D
TL064
U4A
TL064
R8
20k
U4C
TL064
R10
1K
clock
D2
4,7V
J2
segn (1 ck)
R9
20k
U4B
TL064
R11
1K
Segnale
D3
4,7V

Inserisco di seguito la foto del circuito e il lato rame:

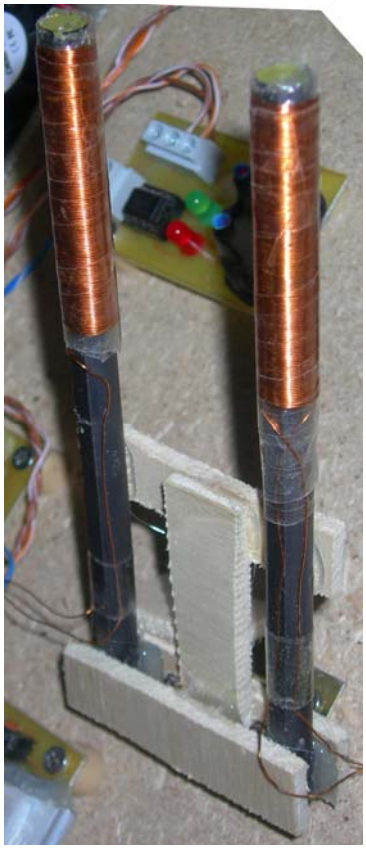




Antenne:

Dedico un piccolo paragrafo alle antenne da me utilizzate. Entrambe le antenne sono state realizzate da me con una barretta di ferrite e un filo di rame del diametro di 0,4mm. Il raggiungimento dell'induttanza corretta, necessaria per il buon funzionamento dell'antenna, è avvenuto in modo sperimentale, cioè provando a raggiungere un certo numero di avvolgimenti e testando se questi raggiungeva circa l'induttanza desiderata (2,723mH).

Gli avvolgimenti sono stati fatti in due strati, e solamente sulla parte superiore della ferrite. Una induttanza del genere è realizzabile solo utilizzando del filo di rame smaltato o isolato.



Il fissaggio delle antenne è avvenuto con delle barrette di legno da me intagliate, con una cerniera che permette il movimento dalla base a 90°.

Utilizzando il mio metodo di trasmissione, la tensione al capo dell'antenna ricevente è molto influente nei termini di corretta ricezione. Anche solo uno spostamento di pochi centimetri potrebbero rovinare il lavoro al sistema che deve dividere il segnale dal clock.

Questo problema poteva essere risolto utilizzando la modulazione di frequenza piuttosto che quella d'ampiezza, scartata in quanto molto più complessa, sia in trasmissione che in ricezione rispetto all'AM, oppure utilizzando un amplificatore con retroazione, capace di aumentare il suo guadagno automaticamente, nel caso in cui avvenisse un allontanamento delle antenne. Quest'ultima soluzione non è stata applicata in quanto ignoravo l'esistenza di tali amplificatori.

Sistema di ricezione dati:

A questo blocco spetta diversi compiti, in principio deve controllare l'arrivo di nuovi dati seriali, inserirli e eseguire vari controlli che poi analizzeremo. In seguito, prelevato il dato, deve trasmetterlo ai display LCD, tramite una comunicazione molto complessa.

All'avvio, per prima cosa, il PIC, provvede ad inizializzare e a far comparire qualche frase sul display. In questo periodo di tempo, non è possibile ricevere dati, d'altronde non è possibile neppure inviarli.

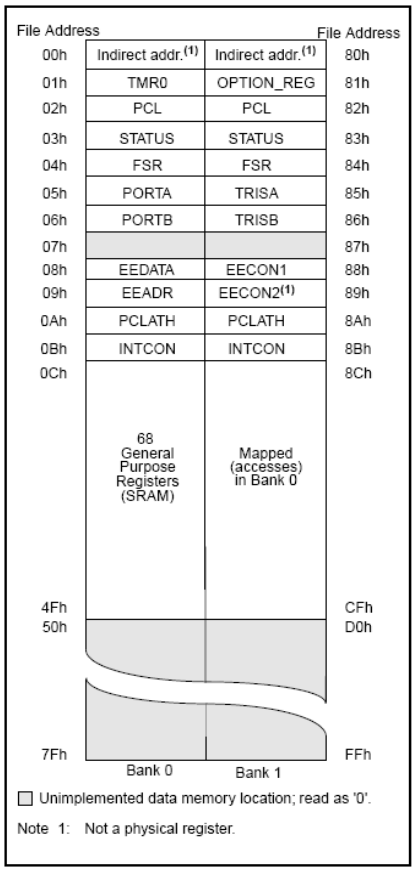
Questo circuito, prevede un tasto di reset (il quale potrebbe tornare utile in caso di calo di tensione) un quarzo a 4 MHz per generare il clock al PIC, due LED di controllo e un connettore per la comunicazione con i tre display LCD.

Inizialmente era previsto l'utilizzo di un unico display, questo mi ha portato a modificare qualche pista sul retro del circuito, ma, grazie alla "programmabilità" dell'integrato PIC16F84A, non è stato necessario aggiungere componenti o realizzare chissà quale altro sistema:

Questo tipo di microcontrollore, è il primo di una lunga famiglia. Difatti possiede una gestione relativamente semplice e una limitato numero di istruzioni. Nonostante questo, in linguaggio assembler, è possibile creare programmi che svolgono operazioni anche molto complesse, grazie anche alle interruzioni, al contatore interno e al buon rapporto numero pin/numero pin IN-OUT.

Di seguito inserisco la lista delle istruzioni utilizzabile con questo PIC, q la gestione di qualche registro, così da rendere forse un po' più chiara la lettura del seguente listato. (per una conoscenza ottimale di questo integrato, è consigliabile leggere il suo datasheet).

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb	LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff		C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff		Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff		Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx		Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff		Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff		Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff			1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff		Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff			1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff		Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff		Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff			
NOP	-	No Operation	1	00 0000 0xx0 0000			
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff		C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff		C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff		C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff			1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff		Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff			1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff			1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff			3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff			3
LITERAL AND CONTROL OPERATIONS							
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk		C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk		Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk			
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100		$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk			
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk		Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk			
RETFIE	-	Return from interrupt	2	00 0000 0000 1001			
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk			
RETURN	-	Return from Subroutine	2	00 0000 0000 1000			
SLEEP	-	Go into standby mode	1	00 0000 0110 0011		$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk		C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk		Z	



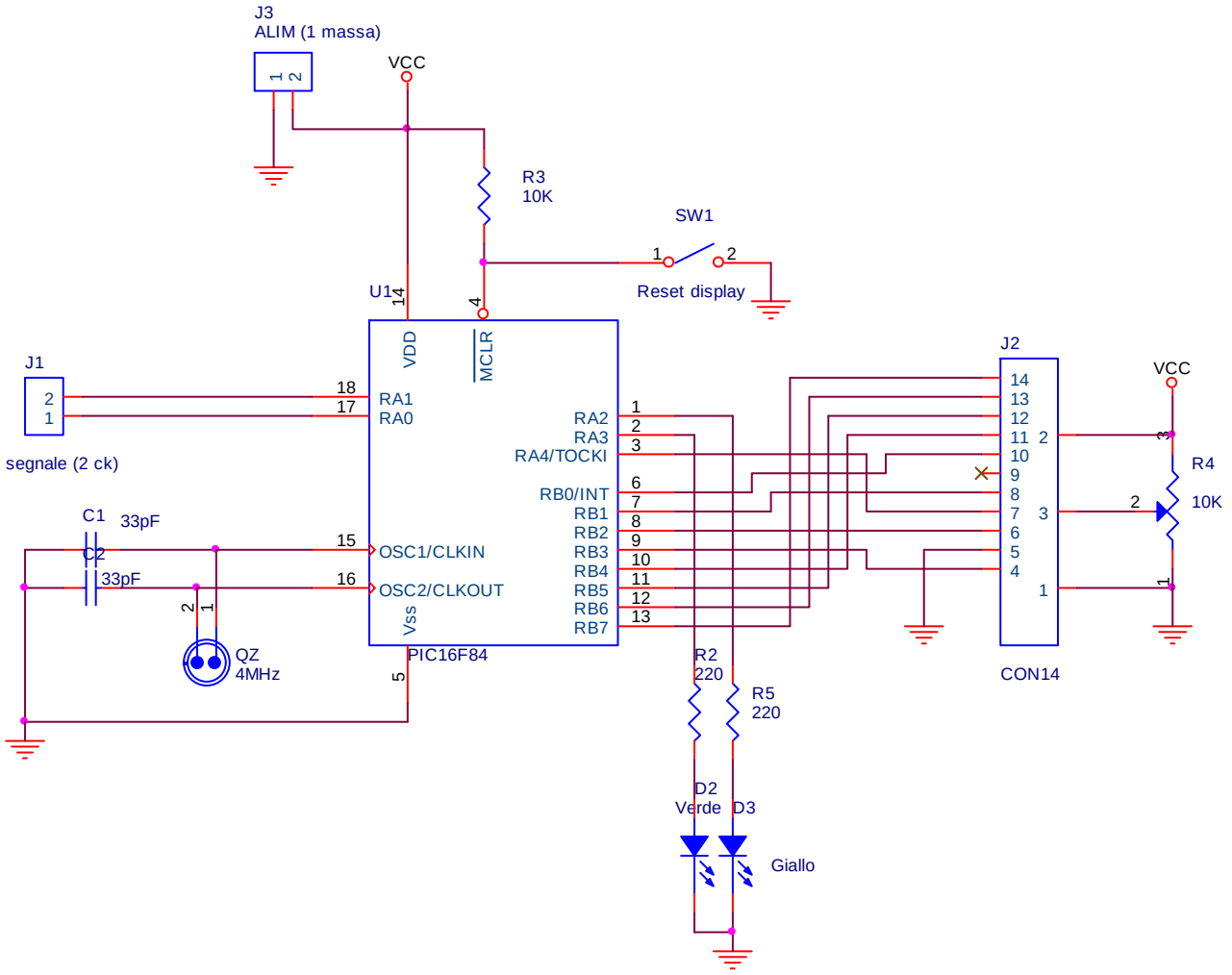
Questo disegno spiega com'è impostata la gestione della memoria.

Nel mio programma si è rivelato necessario l'utilizzo del contatore interno. Tale contatore ha la capacità di inviare un'interruzione software quando capita in overflow, cioè quando arriva al massimo del suo conteggio.
La sua utilità si è rivelata nel controllare il tempo di durata di trasmissione di un singolo byte. Difatti poteva capitare, come blocco invio dati via cavo parallelo, che un disturbo facesse iniziare in anticipo il la ricezione, facendo rovinare tutta l'acquisizione, in quanto il primo bit veniva rilevato come già ricevuto.
Con questo controllo il TMR0 (il suddetto contatore interno) blocca la ricezione nel caso che superasse un secondo di trasmissione, considerando i bit ricevuti fino ad allora, come da scartare.
Il contatore, nel mio caso, preleva gli impulsi di clock dal ciclo macchina (1MHz), ciò significa che se può contare fino a 256, e può dividere il clock per 256 volte. Il TMR0, potrà al massimo controllare una trasmissione di 65ms.
Per ovviare a ciò ho creato un registro, capace di contare, occupando per pochissimi µs il tempo del processore, il numero di interruzioni che genera il TMR0. Questo significa che se voglio ad esempio, un tempo massimo di 650ms, basterà inserire in questo registro il valore di 10.

Riassunto dei registri

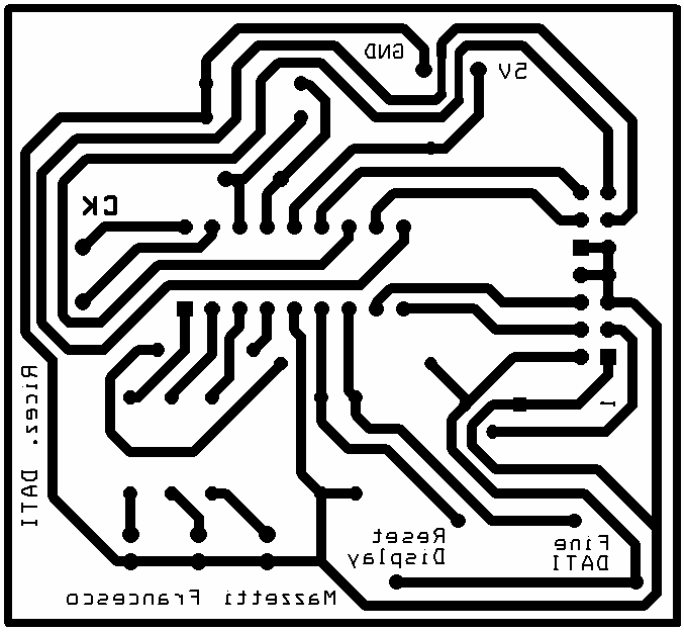
Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note3)
Bank 0											
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
01h	TMR0	8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu
02h	PCL	Low order 8 bits of the Program Counter (PC)								0000 0000	0000 0000
03h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
04h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
05h	PORTA (4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06h	PORTB (5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
07h		Unimplemented location, read as '0'								---- ----	---- ----
08h	EEDATA	EEPROM data register								xxxx xxxx	uuuu uuuu
09h	EEADR	EEPROM address register								xxxx xxxx	uuuu uuuu
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
Bank 1											
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	0000 0000
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
85h	TRISA	—	—	—	PORTA data direction register					---1 1111	---1 1111
86h	TRISB	PORTB data direction register								1111 1111	1111 1111
87h		Unimplemented location, read as '0'								---- ----	---- ----
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	EEPROM control register 2 (not a physical register)								---- ----	---- ----
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

Per quanto riguarda il listato (che trascriverò più avanti) tengo a spiegare come sono riuscito ad ottenere la comunicazione su tre display LCD tutti collegati in cascata.
Guardando le istruzioni e il metodo di comunicazione tra PIC ed LCD, mi sono accorto del fatto che questi dispositivi sono tri-state, cioè necessitano di definiti impulsi di enable gestiti dallo stesso PIC. Difatti, solamente se tali LCD vengono abilitati i precedenza, si potrà visualizzare il carattere desiderato. Ed ecco, che, in questo modo, sarà possibile, tramite una modifica alle librerie che gestiscono la comunicazione PIC-LCD, poter scrivere solamente su nessuno, qualcuno o tutti i display, anche se l'invio del carattere viene eseguito su tutti.
Ci sarà un registro (da me ideato) in cui ai primi tre bit corrisponderanno i tre LCD, se questi sono a 1, l'LCD sarà abilitato, in caso contrario rimarrà ad impedenza infinita, e quindi assolutamente passivo ai dati ricevuti.
Naturalmente perché ciò sia possibile è necessario avere un filo di enable dedicato per ogni display, come ben specificato nello schema a blocchi.
Di seguito riporto lo schema elettrico modificato.



Il trimmer posizionato dietro al connettore è utile per regolare il contrasto del display
Il LED giallo sta ad indicare l'attesa di un impulso di clock, per la ricezione del dato, mentre il led verde, indica che questa procedura è iniziata. Come spiegato prima, all'accensione del LED verde parte il TMR0, per cui, quest'ultima spia non potrà che rimanere accesa il tempo massimo impostato nel programma.

E di seguito il lato rame e la foto del circuito



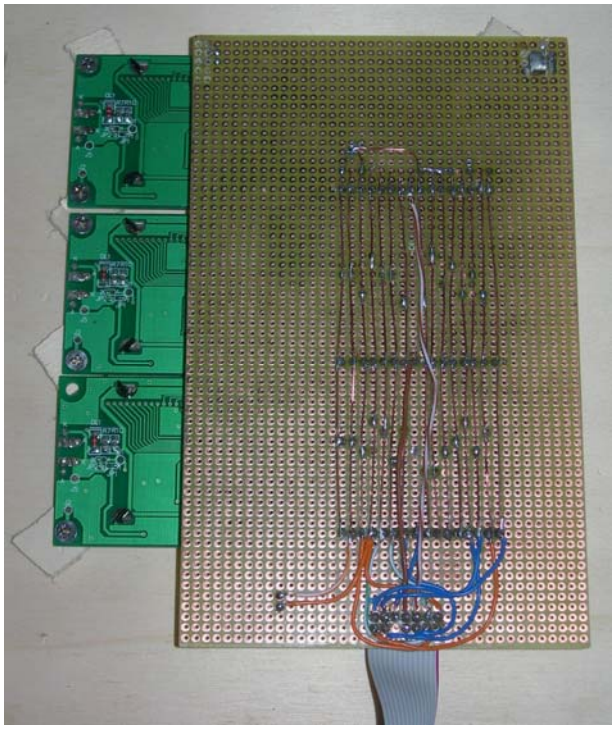
Display LCD:

Quest'ultimo blocco, non è altro che il collegamento fisico (realizzato su semplice millefiori) avvenuto tra il dispositivo di ricezione dati e i tre LCD.

Come spiegato prima, tutti e tre gli LCD sono collegati in cascata, solamente il PIN 6, quello per l'enable, possiede un canale separato per ogni display.

Questi display LCD hanno la possibilità di essere retroilluminati agendo sui primi due PIN di collegamento degli stessi. Non possedendo un contrasto notevole, alla luce del giorno è comunque necessario attivare la retroilluminazione (tramite l'interruttore sul pannello frontale). Nonostante la potente luminosità e le grandi dimensioni, tali display,come tutti, consumano quantità minime di corrente.

Tutti e tre, in caso di retroilluminazione attiva, consumano circa 100mA, mentre in caso contrario meno di 10mA.

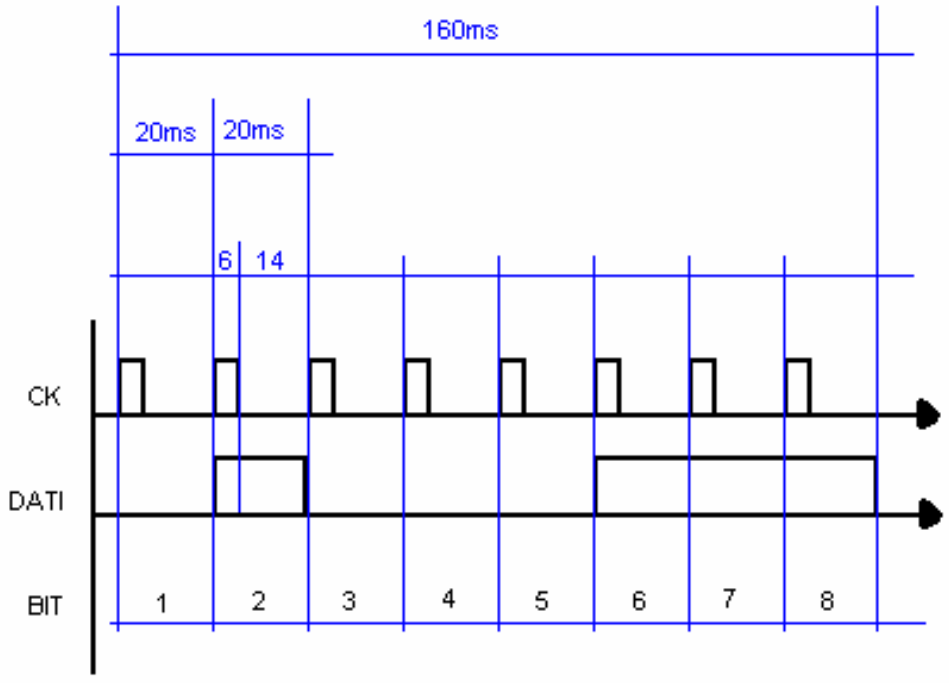


2d Tecnica trasmissione

Il modo in cui ho deciso di trasmettere i dati è stato più volte accennato precedentemente.
In questo paragrafo, tramite l'aiuto di grafici cercherò di spiegare, facendo anche accenni al codice presente nel PIC, come funziona la mia intuitiva trasmissione e ricezione, tralasciando il percorso compiuto dal segnale stesso.

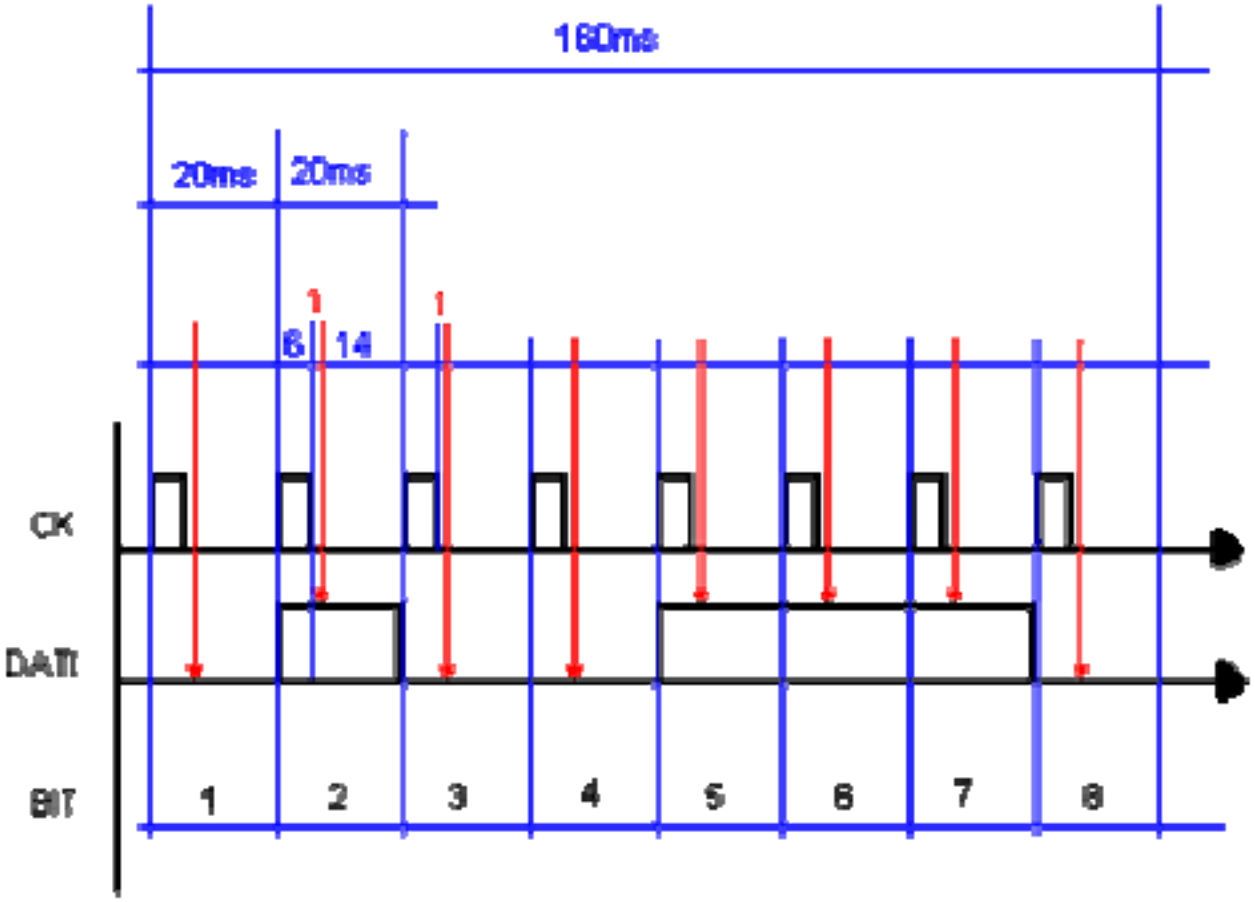
Credo che qualsiasi esperto in telecomunicazioni avrebbe molto da ridire, ma ho trovato questo sistema semplice, adattabile e didattico.
Tutto si basa sull'impulso di clock, generato dal blocco *invio dati*, questo impulso viene emesso, ogni qual volta si presenta un nuovo bit in uscita, scandendo il tempo di trasmissione.
Questo naturalmente il sistema in ricezione lo sa bene, difatti il programma è stato costruito facendo in modo di controllare e quindi memorizzare i bit, solamente dopo la ricezione del suddetto impulso (devo sottolineare che il PIC aspetta pure che suddetto impulso termini, sia per motivi legati al *demodulatore AM* sopra spiegati, che per il blocco *ricezione dati parallelo*, in quanto, nella realizzazione si sarebbe potuto usare un PISO, con clock sul fronte di discesa).
Questo sistema previene il non-riconoscimento di una serie consecutiva di bit a 0 o a 1, classico problema delle trasmissioni.
Sottolineo inoltre il controllo di parità, anche quello precedentemente accennato:
Questo tipo di controllo, è il più banale nelle telecomunicazioni, e previene gli errori con una media fedeltà.
Il tutto si basa, nell'inviare pacchetti (nel mio caso byte) contenenti un numero pari di bit a 1. Questo diventa possibile in quanto l'ottavo bit si è rivelato sempre a 0, in quanto tutti i caratteri che inviavo si presentano prima di 127 (in binario "01111111"). Una volta rivelato dalla tastiera il carattere da inviare, viene controllata la parità e nel caso forzato a 1 l'ultimo bit.
Questo naturalmente il sistema di ricezione lo deve considerare: prima controllando la parità, poi in caso di successo, forzare lo zero nell'ottavo bit.
Se ciò non avvenisse, l'invio al display sarebbe al 50% circa sbagliato, in quanto in ricezione a parte pochi casi, non viene eseguito nessuna operazione del dato acquisito, ma semplicemente (con tutte le complicazioni del caso) smistato al display, che riconosce in automatico i caratteri ASCII.
A parte i pulsanti di controllo, nel riconoscimento del tasto, viene sottratto 32 per le lettere e 12 per i caratteri, nel caso venga rilevata la pressione del tasto maiusc.

Ecco l'esempio dell'invio della lettera "G"(d'71', b'01000111',h'47')

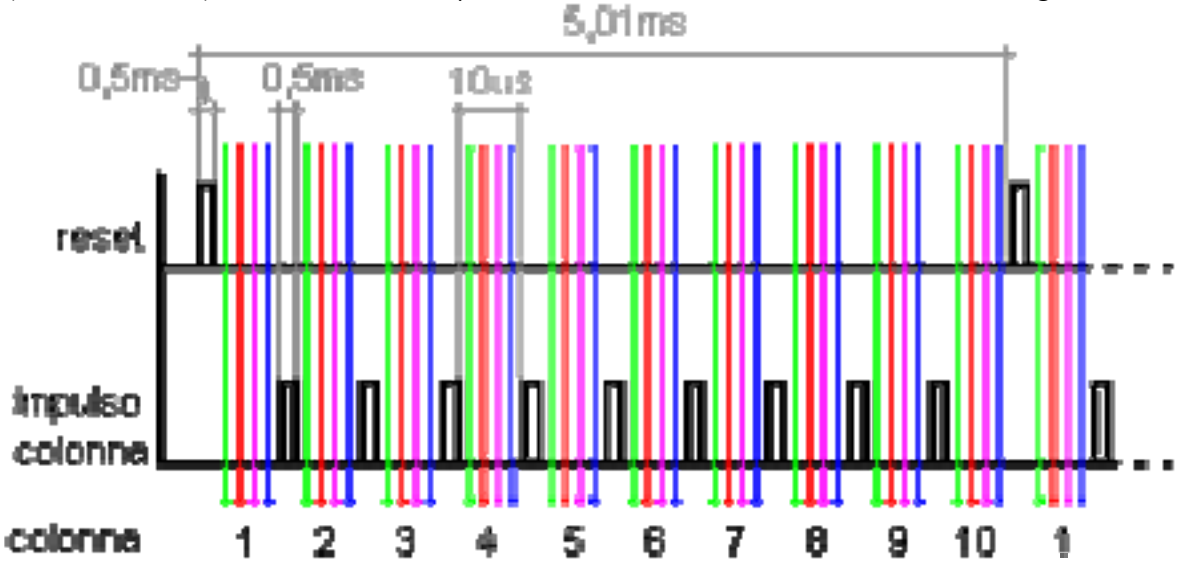


I ritardi per il clock(6ms) e per il segnale(14ms), vengono realizzati con un delay regolabile, cioè un ciclo "inutile", di cui è possibile modificare la durata grazie alle intestazioni a inizio codice. Come si vede il dato viene caricato contemporaneamente all'impulso (che sia 0 o che sia 1) subito dopo viene chiamata la routin di ritardo, così per 8 volte, fino alla durata di 160ms.

In ricezione la distanza tra la rilevazione del segnale e la fine dell'impulso di clock è di un millisecondo, proprio per evitare che nella salita del segnale, nella ricomposizione AM, venga persa qualche informazione.



In questo grafico è possibile vedere il tempo di controllo della tastiera, come viene indicato, il grafico non è in scala, difatti l'impulso dura ben 0,5ms, mentre il controllo delle quattro righe (frecce colorate) avviene in solo 10µs. Il resto diventa chiaro osservando l'immagine.



Non tutti i pulsanti della tastiera, naturalmente, sono assegnati a caratteri o numeri ASCII. Considerando la possibilità di controllare 40 tasti, 4 tasti rimangono liberi (26 lettere più 10 numeri).
Si è creata difatti la possibilità e l'utilità di utilizzare tasti di controllo.
Togliendo il tasto di controllo maiuscolo e lo spazio, rimangono due tasti liberi, uno utilizzato come cancelletto e uno come invio.
Inoltre ho utilizzato la funzione maiuscolo pure su questi pulsanti ottenendo una duplicazione dei tasti funzione.
I codici e le funzioni di questi pulsanti sono riassunti in questa tabella, sono stati prelevati da spazi inutilizzati della tabella ASCII

Combinazione tasto	funzione	Codice binario	Codice decimale
"canc"	Cancella l'ultimo carattere	00111110	62
Shift+canc	Pulizia di tutti gli LCD	00111101	61
Freccia giù	Invio	00111111	63
Shift+freccia giù	Modalità Clone	01000000	64
Spazio+shift	Freccia a destra	00111100	60

La *modalità clone*, è una speciale funzione che permette di scrivere su tutti e tre i display contemporaneamente. Ripremendola, è possibile tornale alla modalità classica. Tutto questo è possibile agendo sul famoso registro di gestione LCD presente nel blocco *ricezione dati*.

2e Listato Assembler Trasmissione

Di seguito, senza interruzioni, tutto il programma assembler del PIC utilizzato in trasmissione. Nel realizzare tale programma, ho inserito molti commenti, anche perché tornandoci su più volte, è necessario mantenere delle spiegazioni di riferimento.
Il programma ha subito variazioni rispetto al primo realizzato, tutte marginali e non sconvolgenti. In particolare è stato inserito dopo il controllo di parità e il controllo del tasto maiuscolo.
Il listato è stato realizzato in un momento in cui diventava impossibili testarlo, proprio per questo ho utilizzato molto le funzioni di debugger di MPLAB (programma della microchip) che si sono rivelate comode e funzionali.

```
;*****************************************************************************************************
;*                    24/4/2008                                                                      *
;* LISTATO RILEVAZIONE DATI TASTIERA E INVIO IN FORMA SERIALE DELLA LETTERA IN CODICE ASCII           *
;* Studio dei mezzi trasmissivi, esame di stato 2008                                                 *
;*                                                                                                   *
;*                                                Mazzetti Francesco ©                               *
;*                                                                                                   *
;*****************************************************************************************************

                    List P=PIC16F84A
                    include "p16F84a.inc"
                    RADIX                   DEC
                    ERRORLEVEL -302
                    __CONFIG    0x3FF9      ;(xt, no wdt, no cont)

                    ORG 0x00
                        BSF STATUS,RP0
                        MOVLW b'00000000' ;(RA0-1, clock e dati)
                        MOVWF TRISA
                        MOVLW b'00001111' ;RB0-3, in riga, RB4-5, colonna, LED
                        MOVWF TRISB
                        BSF OPTION_REG,7
                        BCF STATUS,RP0

                    ORG 0x0C
cont1                   RES1        ;usati dal delay
cont2                   RES1        ;usati dal delay
cont3                   RES 1       ;libero utilizzo
DATO                    RES 1       ;dato da inviare
cpar                    RES 1       ;contatore di 1 utile al controllo di parità
```

```
RIGA1           EQU 3                   ;controllo riga 1           (tastiera usata, composta da 40 tasti,
RIGA2           EQU 2                   ;controllo riga 2            10 colonne e 4 righe:i PIN per la
RIGA3           EQU 1                   ;controllo riga 3            comunicazione sono uno di reset, uno per
RIGA4           EQU 0                   ;controllo riga 4            l'incremento della colonna e 4 righe)
START           EQU 4                   ;reset e inizio controllo tastiera
COLONN          EQU 5                   ;controllo della colonna
LTAST           EQU 6                   ;LED tastiera (rosso)
LINVIO          EQU 7                   ;LED invio dati (verde)
DATI            EQU 0                   ;PIN dati
CK              EQU 1                   ;PIN ck
msCK            EQU d'6'        ;tempo di durata del clock in ms
msDATI          EQU d'14'       ;tempo di durata del livello logico del dato dopo aver finito il ck in ms
CONTROLLO1      EQU b'00111100' ;freccia destra, spazio + shift
CONTROLLO2      EQU b'00111101' ;controllo:clear
CONTROLLO3      EQU b'00111110' ;controllo: cancella l'ultimo carattere
CONTROLLO4      EQU b'00111111' ;controllo: invio
CONTROLLO5      EQU b'01000000' ;controllo: modalità clone

mainloop        nop
                clrf    PORTA
                clrf    PORTB
                nop
                call    iniz
loop            call contast
                call    invio
                goto    loop


iniz            movlw d'23'             ;cavolata grafica
                movwf cont3
ripinz          bsf PORTB,LTAST
                movlw d'200'            ;LED rosso 200ms
                call delay
                bcf PORTB,LTAST
                bsf PORTB,LINVIO
                movlw d'200'            ;LED verde 200ms
                call delay
                bcf PORTB, LINVIO
                decfsz cont3,1
                goto ripinz
                return
;************************************controllo dei 40 tasti della tastiera*******************************
contast         bsf PORTB,LTAST         ;accensione LED di controllo tastiera
                bsf PORTB,START         ;inizializzazione tastiera
                call rit                ;ritardino
                bcf PORTB,START
                btfsc PORTB,RIGA1 ;controllo prima colonna
                retlw '1'
                btfsc PORTB,RIGA2
                retlw 'q'
                btfsc PORTB,RIGA3
                retlw 'a'
                btfsc PORTB,RIGA4
                retlw 'z'
                bsf PORTB,COLONN                ;settaggio seconda colonna
                call rit
                bcf PORTB,COLONN
                btfsc PORTB,RIGA1 ;controllo seconda colonna
                retlw '2'
                btfsc PORTB,RIGA2
                retlw 'w'
                btfsc PORTB,RIGA3
                retlw 's'
                btfsc PORTB,RIGA4
                retlw 'x'
                bsf PORTB,COLONN                ;settaggio terza colonna
                call rit
                bcf PORTB,COLONN
                btfsc PORTB,RIGA1 ;controllo terza colonna
                retlw '3'
                btfsc PORTB,RIGA2
                retlw 'e'
                btfsc PORTB,RIGA3
                retlw 'd'
                btfsc PORTB,RIGA4
                retlw 'c'
```

```
bsf PORTB,COLONN		;settaggio quarta colonna
call rit
bcf PORTB,COLONN
btfsc PORTB,RIGA1 ;controllo quarta colonna
retlw '4'
btfsc PORTB,RIGA2
retlw 'r'
btfsc PORTB,RIGA3
retlw 'f'
btfsc PORTB,RIGA4
retlw 'v'
bsf PORTB,COLONN		;settaggio quinta colonna
call rit
bcf PORTB,COLONN
btfsc PORTB,RIGA1 ;controllo quinta colonna
retlw '5'
btfsc PORTB,RIGA2
retlw 't'
btfsc PORTB,RIGA3
retlw 'g'
btfsc PORTB,RIGA4
retlw 'b'
bsf PORTB,COLONN		;settaggio sesta colonna
call rit
bcf PORTB,COLONN
btfsc PORTB,RIGA1 ;controllo sesta colonna
retlw '6'
btfsc PORTB,RIGA2
retlw 'y'
btfsc PORTB,RIGA3
retlw 'h'
btfsc PORTB,RIGA4
retlw 'n'
bsf PORTB,COLONN		;settaggio settima colonna
call rit
bcf PORTB,COLONN
btfsc PORTB,RIGA1 ;controllo settima colonna
retlw '7'
btfsc PORTB,RIGA2
retlw 'u'
btfsc PORTB,RIGA3
retlw 'j'
btfsc PORTB,RIGA4
retlw 'm'
bsf PORTB,COLONN		;settaggio ottava colonna
call rit
bcf PORTB,COLONN
btfsc PORTB,RIGA1 ;controllo ottava colonna
retlw '8'
btfsc PORTB,RIGA2
retlw 'i'
btfsc PORTB,RIGA3
retlw 'k'
btfsc PORTB,RIGA4
retlw ' '
bsf PORTB,COLONN		;settaggio nona colonna
call rit
bcf PORTB,COLONN
btfsc PORTB,RIGA1 ;controllo nona colonna
retlw '9'
btfsc PORTB,RIGA2
retlw 'o'
btfsc PORTB,RIGA3
retlw 'l'
btfsc PORTB,RIGA4
retlw CONTROLLO4
bsf PORTB,COLONN		;settaggio decima colonna
call rit
bcf PORTB,COLONN
btfsc PORTB,RIGA1 ;controllo decima colonna
retlw '0'
btfsc PORTB,RIGA2
retlw 'p'
btfsc PORTB,RIGA3
retlw CONTROLLO3
goto contast
```

```
invio           movwf DATO              ;registra il tasto premuto (contenuto in W) nel registro DATO
                    bcf PORTB, LTAST  ;spegne il LED di controllo tastiera
                    bsf PORTB, LINVIO       ;accende il LED di invio dati
                    call shift              ;controllo Maiuscolo
                    call bpar               ;chiama il controllo di parità (mette nell'ultimo bit un uno o 0 per pareggiare gli 1)
                    btfsc DATO,0            ;salta se il bit 0 del dato è 0
                    bsf PORTA,DATI          ;pone a 1 il bit in uscita dei dati
                    btfss DATO,0            ;salta se il bit 0 del dato è 1
                    bcf PORTA,DATI          ;pone a 0 il bit in uscita dei dati
                    call clock              ;in uscita c'è il dato, ora viene chiamato il ck
                    movlw msDATI            ;chiama un ritardo uguale al tempo di durata del dato in uscita
                    call delay
                        btfsc DATO,1            ;salta se il bit 1 del dato è 0
                    bsf PORTA,DATI          ;pone a 1 il bit in uscita dei dati
                    btfss DATO,1            ;salta se il bit 1 del dato è 1
                    bcf PORTA,DATI          ;pone a 0 il bit in uscita dei dati
                    call clock              ;in uscita c'è il dato, ora viene chiamato il ck
                    movlw msDATI            ;chiama un ritardo uguale al tempo di durata del dato in uscita
                    call delay
                    btfsc DATO,2            ;salta se il bit 2 del dato è 0
                    bsf PORTA,DATI          ;pone a 1 il bit in uscita dei dati
                    btfss DATO,2            ;salta se il bit 2 del dato è 1
                    bcf PORTA,DATI          ;pone a 0 il bit in uscita dei dati
                    call clock              ;in uscita c'è il dato, ora viene chiamato il ck
                    movlw msDATI            ;chiama un ritardo uguale al tempo di durata del dato in uscita
                    call delay
                    btfsc DATO,3            ;salta se il bit 3 del dato è 0
                    bsf PORTA,DATI          ;pone a 1 il bit in uscita dei dati
                    btfss DATO,3            ;salta se il bit 3 del dato è 1
                    bcf PORTA,DATI          ;pone a 0 il bit in uscita dei dati
                    call clock              ;in uscita c'è il dato, ora viene chiamato il ck
                    movlw msDATI            ;chiama un ritardo uguale al tempo di durata del dato in uscita
                    call delay
                    btfsc DATO,4            ;salta se il bit 4 del dato è 0
                    bsf PORTA,DATI          ;pone a 1 il bit in uscita dei dati
                    btfss DATO,4            ;salta se il bit 4 del dato è 1
                    bcf PORTA,DATI          ;pone a 0 il bit in uscita dei dati
                    call clock              ;in uscita c'è il dato, ora viene chiamato il ck
                    movlw msDATI            ;chiama un ritardo uguale al tempo di durata del dato in uscita
                    call delay
                    btfsc DATO,5            ;salta se il bit 5 del dato è 0
                    bsf PORTA,DATI          ;pone a 1 il bit in uscita dei dati
                    btfss DATO,5            ;salta se il bit 5 del dato è 1
                    bcf PORTA,DATI          ;pone a 0 il bit in uscita dei dati
                    call clock              ;in uscita c'è il dato, ora viene chiamato il ck
                    movlw msDATI            ;chiama un ritardo uguale al tempo di durata del dato in uscita
                    call delay
                    btfsc DATO,6            ;salta se il bit 6 del dato è 0
                    bsf PORTA,DATI          ;pone a 1 il bit in uscita dei dati
                    btfss DATO,6            ;salta se il bit 6 del dato è 1
                    bcf PORTA,DATI          ;pone a 0 il bit in uscita dei dati
                    call clock              ;in uscita c'è il dato, ora viene chiamato il ck
                    movlw msDATI            ;chiama un ritardo uguale al tempo di durata del dato in uscita
                    call delay
                    btfsc DATO,7            ;salta se il bit 7 del dato è 0
                    bsf PORTA,DATI          ;pone a 1 il bit in uscita dei dati
                    btfss DATO,7            ;salta se il bit 7 del dato è 1
                    bcf PORTA,DATI          ;pone a 0 il bit in uscita dei dati
                    call clock              ;in uscita c'è il dato, ora viene chiamato il ck
                    movlw msDATI            ;chiama un ritardo uguale al tempo di durata del dato in uscita
                    call delay
                    bcf PORTA, DATI
                    movlw d'80'             ;ritardo pressione tasto
                    call delay
                    bcf PORTB, LINVIO       ;spegne il LED di invio dati
                    return          ;chiama nuovamente il controllo tastiera

clock           bsf PORTA, CK           ;alza il bit del clock
                    movlw msCK                  ;tempo di durata del clock in ms
                    movwf cont1
                clrf cont2
rit_ck              nop
                decfsz cont2,F
                goto rit_ck
                nop
                decfsz cont1,F
                goto rit_ck
```

```
                        bcf     PORTA, CK               ;abbassa il bit del ck
                    return

shift               bsf PORTB,START     ;inizializzazione tastiera
                        call rit        ;ritardino
                        bcf PORTB,START
                        bsf PORTB,COLONN        ;settaggio seconda colonna
                        call rit
                        bcf PORTB,COLONN
                        bsf PORTB,COLONN        ;settaggio terza colonna
                        call rit
                        bcf PORTB,COLONN
                        bsf PORTB,COLONN        ;settaggio quarta colonna
                        call rit
                        bcf PORTB,COLONN
                        bsf PORTB,COLONN        ;settaggio quinta colonna
                        call rit
                        bcf PORTB,COLONN
                        bsf PORTB,COLONN        ;settaggio sesta colonna
                        call rit
                        bcf PORTB,COLONN
                        bsf PORTB,COLONN        ;settaggio settima colonna
                        call rit
                        bcf PORTB,COLONN
                        bsf PORTB,COLONN        ;settaggio ottava colonna
                        call rit
                        bcf PORTB,COLONN
                        bsf PORTB,COLONN        ;settaggio nona colonna
                        call rit
                        bcf PORTB,COLONN
                        bsf PORTB,COLONN        ;settaggio decima colonna
                        call rit
                        bcf PORTB,COLONN
                        btfss PORTB,RIGA4
                        return                  ;non è stato premuto il maiuscolo, torna indietro
                        movf DATO,0
                        xorlw ' '               ;se è stato premuto spazio con shift = tab
                        btfss STATUS,Z
                        goto shift2             ;passa la contr successivo
                        movlw CONTROLLO1        ;metti il tab
                        movwf DATO              ;metti in dato il controllo1
                        return
shift2              movf DATO,0
                        xorlw CONTROLLO3        ;se è stato premuto canc + shift
                        btfss STATUS,Z
                        goto shift3
                        movlw CONTROLLO2        ;clear
                        movwf DATO              ;metti in dato il controllo2
                        return
shift3              movf DATO,0
                        xorlw CONTROLLO4        ;se è stato premuto ctrl4+shift
                        btfss STATUS,Z
                        goto shift4
                        movlw CONTROLLO5        ;clear
                        movwf DATO              ;metti in dato il controllo2
                        return
shift4              btfsc DATO,6        ;testa il bit 6...è un numero?
                        goto shift5
                        movlw d'16'
                        subwf DATO,1            ;è un numero sottrai 16
                        return
shift5              movlw d'32'             ;è stato premuto con lettere
                        subwf DATO,1            ;somma 32, e ottieni il maiuscolo
                        return
;***************Routine controllo di  parità*****************************
bpar                clrf cpar           ;pulisce il registro del conteggio di parità
                        btfsc DATO,0
                        incf cpar,1     ;conta gli uno...
                        btfsc DATO,1
                        incf cpar,1
                        btfsc DATO,2
                        incf cpar,1
                        btfsc DATO,3
                        incf cpar,1
                        btfsc DATO,4
                        incf cpar,1
                        btfsc DATO,5
```

```
                        incf cpar,1
                        btfsc DATO,6
                        incf cpar,1
                        btfsc DATO,7
                        incf cpar,1
                        btfsc cpar,0        ;tasta l'ultimo bit del contatore degli 1, se 1=dispari, se 0=pari
                        bsf DATO,7                  ;mette a uno il bit 7, per rendere pari gli uno
                        return

;***********************RITARDI UTILI*****************************************
;piccolo delay per l'impulso da inviare al controllo colonna
rit                     clrf cont1          ;ritardo di riconoscimento di circa 0,50ms
ritloop3            decfsz cont1,F
                        goto ritloop3
                        return
;ritardo regolato da W, il  valore di W in decimale è uguale al delay in ms
delay                   movwf cont1
                    clrf cont2
rit_loop            nop
                    decfsz cont2,F
                    goto rit_loop
                    nop
                    decfsz cont1,F
                    goto rit_loop
                        return
                    end
```

L'indentazione del programma non è venuta molto bene, per colpa della copia MPLAB-WORD. (posso garantire, che da bravo programmatore ho indentato bene).
Il programma, a grandi linee occupa circa il 30-40% della memoria codice, questo vuol dire che possiede solo 300-400 istruzioni

2f Listato Assembler Ricezione

Ecco invece il listato del programma in assembler del PIC in ricezione. Tale programma possiede molte più istruzioni(circa 600-700), molte delle quali però non sono indispensabili al programma. Difatti almeno un duecento di righe viene dedicato all'inizializzazione e allo scorrimento di informazioni generali che avviene ad ogni avvio.
Il programma presenta molti commenti, quindi nel complesso chiaro, per aiutare la lettura consiglio di lasciare stare la parte iniziale, e di concentrarsi su quella secondari, molto più delicata e particolare, soprattutto per le tecniche utilizzate (come la ricezione, il controllo parità, il controllo del cursore *CURS3*)
Di seguito il programma generale:

```
;*****************************************************************************
;*                      24/4/2008                                            *
;* LISTATO RICEZIONE DATI TRAMITE BUS CLOCK E DATI, INVIO DATI A DISPLAY LCD  *
;* Studio dei mezzi trasmissivi,   esame di stato 2008                       *
;*                                                                           *
;* Predisposto per massimo 7 display LCD  (nel caso vanno aumentati gli EN)  *
;*                                                          Mazzetti Francesco © *
;*                                                                           *
;*****************************************************************************

                        PROCESSOR  16F84A
                        RADIX               DEC
                        INCLUDE             "p16f84a.inc"
                        ERRORLEVEL          -302
                        __CONFIG    0x3FF1  ;(xt, no wdt, si cont)



                        ORG         0CH
;memoria usata dalla libreria per il display LCD
lsb                             RES     1
msb                             RES     1
tmp                             RES     1
```

```
tmp1                    RES     1
;memoria usata per il seguente sorgente
CONT1                   RES 1                   ;conteggio vario
CONT2                   RES 1                   ;conteggio vario
CONT3                   RES 1                   ;conteggio vario
CURS1                   RES 1                   ;posizione cursore 1
CURS2                   RES 1                   ;Posizione cursore 2
CURS3                   RES 1                   ;posizione cursore durante la ric. dati
DATO                    RES 1                   ;dato ricetuvo
TIMECK                  RES 1                   ;variabile di tempo massimo ricezione dati
STATLCD                 RES 1                   ;registro scelta LCD (ogni bit corrisponde un LCD.1=abilitato), bit7,1=lavoro clone
MEMLCD                  RES 1                   ;registro memoria stato lcd
cpar                    RES 1                   ;conteggio controllo parità
;*************************************************
;Intestazioni
EN3                             EQU 0                   ;Enable LCD 3
EN2                             EQU 1                   ;Enable LCD 2
EN1                             EQU 2                   ;Enable LCD 1
RS                              EQU 3                   ;usato nella libreria LCD
PIN11                   EQU 4                   ;usato nella libreria LCD
PIN12                   EQU 5                   ;usato nella libreria LCD
PIN13                   EQU 6                   ;usato nella libreria LCD
PIN14                   EQU 7                   ;usato nella libreria LCD
DATI                    EQU 0                   ;input DATI
CK                              EQU     1                               ;input clock
LEDATT                  EQU 2                   ;LED attesa del dato
LEDRIC                  EQU     3                       ;LED dato in arrivo
EN4                             EQU 4                   ;Enable LCD 4 (info carattere)
CONTROLLO1              EQU b'00111100'         ;TAB, spazio + shift
CONTROLLO2              EQU b'00111101'         ;controllo:clear
CONTROLLO3              EQU b'00111110'         ;controllo: cancella l'ultimo carattere
CONTROLLO4              EQU b'00111111'         ;controllo: a capo
CONTROLLO5              EQU b'01000000'         ;controllo: /////
CKMAX                   EQU     d'10'                   ;contiene il tempo massimo di attesa segnale
                                                                        ;di clock per un'intera operazione di ricezione.
                                                                        ;sarà data dal CKMAX*256*256[us](più qualche ms)
                        ORG 00H
                            goto        inizio                  ;inizio programma (evita di partire con l'interruzione)
;**********INTERRUZIONE PRESCALER*****************
;questa interruzione avviene dopo 65ms da un impulso di ck e l'altro, ciò è utile
;in quanto normalmente un impulso di clock viene seguito dal secondo dopo soli 20ms
;se ciò non avvine vuol dire che quello precedente era un disturbo o un errore.
;*********************************************
                        ORG 04H
                            bcf     INTCON,T0IF             ;azzera il riconoscimento dell'interruzione
                            bsf                 INTCON, T0IE
                            bsf                 INTCON, GIE
                            decfsz      TIMECK,1                        ;decrementa la variabile di tempo
                            retfie                                      ;ritorna a dove era stato interrotto
                            goto        rick                    ;ritorna alla ricerca di un nuovo impulso di ck

;**************FINE INTERRUZIONE**********************


inizio                  bsf         STATUS,RP0
                                    movlw       b'11000111'             ;impostazioni 1/256   TMR0 interno
                                    movwf       OPTION_REG                      ;copia nel registro option
                                    movlw       b'00000011'             ;gli in/out per portA
                                    movwf       TRISA
                                    movlw       b'00000000'             ;gli in/out per portB
                                    movwf       TRISB
                                    bsf                 INTCON,7
                                    bcf                 STATUS,RP0

mainloop                clrf PORTA              ;nel caso...spegne tutti i LED
                                    clrf        CURS3               ;azzeramento del cursore di posizionamento
                                    clrf        STATLCD             ;azzeramento del reg. inf. LCD
                                    movlw       b'00001111'         ;abilito i 4 lcd
                                    movwf       STATLCD
                                    call        init
                                    call        start
                                    movlw       b'00000001'         ;solo l'ultimo lcd
                                    movwf       STATLCD
                                    call        nome
                                    movlw       b'00000010'         ;il secondo lcd
                                    movwf       STATLCD
                                    call        prog
```

```
                        movlw   b'00000100'     ;il terzo LCD
                        movwf   STATLCD
                        call    titol
                        movlw   d'100'          ;ritardo 2 secondi dalla comparsa delle intestazioni
                        call    delay
                        movlw   b'00001111'     ;Pulizia tutti e 4 i display
                        movwf   STATLCD
                        call    cl
                        movlw   b'00000001'     ;abilita solo il primo display
                        movwf   STATLCD
                        movlw   0x0E            ;rendi visibile il cursore (trattino)
                        call    sc
loop            call    rick
                        goto    loop

start           call    cl
                        movlw   02H
                        call    cp              ;Riga 0 posizione 0
                        movlw   'D'
                        call    sd
                        movlw   'i'
                        call    sd
                        movlw   's'
                        call    sd
                        movlw   'p'
                        call    sd
                        movlw   'l'
                        call    sd
                        movlw   'a'
                        call    sd
                        movlw   'y'
                        call    sd
                        movlw   0BH             ;Riga 1 posizione 11
                        call    cp
                        movlw   'L'
                        call    sd
                        movlw   'C'
                        call    sd
                        movlw   'D'
                        call    sd
                        movlw   11H             ;Riga 1 posizione 11
                        call    cp
                        movlw   'I'
                        call    sd
                        movlw   'n'
                        call    sd
                        movlw   'i'
                        call    sd
                        movlw   'z'
                        call    sd
                        movlw   'i'
                        call    sd
                        movlw   'a'
                        call    sd
                        movlw   'l'
                        call    sd
                        movlw   'i'
                        call    sd
                        movlw   'z'
                        call    sd
                        movlw   'z'
                        call    sd
                        movlw   'a'
                        call    sd
                        movlw   't'
                        call    sd
                        movlw   'i'
                        call    sd
                        movlw   80              ;riitardo di circa 0,5s
                        call    delay
                        return

;*************
;ritardo: modificare il registro w prima di chiamare il ritardo
; il valore di w elevato alla terza sarà il ritardo in us
;********
```

```
nome
                movlw   0FH             ;riga 0
                movwf   CURS1
                movlw   1FH             ;riga 1
                movwf   CURS2
nome1           movf    CURS1,0 ;mette il valore del cursore in W
                call    cp              ;posizionamento cursore
                movlw   'F'             ;Spostamento NOME
                call    sd
                movlw   'r'
                call    sd
                movlw   'a'
                call    sd
                movlw   'n'
                call    sd
                movlw   'c'
                call    sd
                movlw   'e'
                call    sd
                movlw   's'
                call    sd
                movlw   'c'
                call    sd
                movlw   'o'
                call    sd
                movlw   ' '
                call    sd
                movf    CURS2,0 ;mette il valore del cursore in W
                call    cp              ;posizionamento cursore
                movlw   'M'             ;Spostamento COGNOME
                call    sd
                movlw   'a'
                call    sd
                movlw   'z'
                call    sd
                movlw   'z'
                call    sd
                movlw   'e'
                call    sd
                movlw   't'
                call    sd
                movlw   't'
                call    sd
                movlw   'i'
                call    sd
                movlw   ' '
                call    sd
                movf    CURS1,0
                xorlw   0
                btfsc   STATUS,Z        ;controlla se è arrivato alla fine
                return
                decf    CURS1,0
                movwf   CURS1           ;imposta il nuovo cursore per il nome
                decf    CURS2,0
                movwf   CURS2           ;imposta il nuovo cursore per il cognome
                movlw   d'35'                   ;impostazione del ritardo 0,04s, vedi delay
                call    delay
                goto    nome1
prog
                movlw   0FH             ;riga 0
                movwf   CURS1
                movlw   1FH             ;riga 1
                movwf   CURS2
prog2           movf    CURS1,0 ;mette il valore del cursore in W
                call    cp              ;posizionamento cursore
                movlw   'P'             ;Spostamento NOME
                call    sd
                movlw   'r'
                call    sd
                movlw   'o'
                call    sd
                movlw   'g'
                call    sd
                movlw   'e'
                call    sd
                movlw   't'
```

```
                call    sd
                movlw   't'
                call    sd
                movlw   'o'
                call    sd
                movlw   ' '
                call    sd
                movlw   'E'
                call    sd
                movlw   's'
                call    sd
                movlw   'a'
                call    sd
                movlw   'm'
                call    sd
                movlw   'e'
                call    sd
                movlw   ' '
                call    sd
                movf    CURS2,0  ;mette il valore del cursore in W
                call    cp              ;posizionamento cursore
                movlw   'A'             ;Spostamento COGNOME
                call    sd
                movlw   'N'
                call    sd
                movlw   'N'
                call    sd
                movlw   'O'
                call    sd
                movlw   ' '
                call    sd
                movlw   '2'
                call    sd
                movlw   '0'
                call    sd
                movlw   '0'
                call    sd
                movlw   '7'
                call    sd
                movlw   '-'
                call    sd
                movlw   '0'
                call    sd
                movlw   '8'
                call    sd
                movlw   ' '
                call    sd
                movf    CURS1,0
                xorlw   0
                btfsc   STATUS,Z        ;controlla se è arrivato alla fine
                return
                decf    CURS1,0
                movwf   CURS1           ;imposta il nuovo cursore per il nome
                decf    CURS2,0
                movwf   CURS2           ;imposta il nuovo cursore per il cognome
                movlw   d'35'                   ;impostazione del ritardo 0,04s, vedi delay
                call    delay
                goto    prog2
titol
                movlw   0FH             ;riga 0
                movwf   CURS1
                movlw   1FH             ;riga 1
                movwf   CURS2
titol2          movf    CURS1,0  ;mette il valore del cursore in W
                call    cp              ;posizionamento cursore
                movlw   'S'             ;Spostamento NOME
                call    sd
                movlw   'T'
                call    sd
                movlw   'U'
                call    sd
                movlw   'D'
                call    sd
                movlw   'I'
                call    sd
                movlw   'O'
                call    sd
```

```
                movlw   ' '
                call    sd
                movlw   'd'
                call    sd
                movlw   'e'
                call    sd
                movlw   'i'
                call    sd
                movlw   ' '
                call    sd
                movlw   'M'
                call    sd
                movlw   'E'
                call    sd
                movlw   'Z'
                call    sd
                movlw   'Z'
                call    sd
                movlw   'I'
                call    sd
                movf    CURS2,0  ;mette il valore del cursore in W
                call    cp              ;posizionamento cursore
                movlw   ' '             ;Spostamento COGNOME
                call    sd
                movlw   ' '
                call    sd
                movlw   'T'
                call    sd
                movlw   'R'
                call    sd
                movlw   'A'
                call    sd
                movlw   'S'
                call    sd
                movlw   'M'
                call    sd
                movlw   'I'
                call    sd
                movlw   'S'
                call    sd
                movlw   'S'
                call    sd
                movlw   'I'
                call    sd
                movlw   'V'
                call    sd
                movlw   'I'
                call    sd
                movlw   ' '
                call    sd
                movf    CURS1,0
                xorlw   0
                btfsc   STATUS,Z        ;controlla se è arrivato alla fine
                return
                decf    CURS1,0
                movwf   CURS1           ;imposta il nuovo cursore per il nome
                decf    CURS2,0
                movwf   CURS2           ;imposta il nuovo cursore per il cognome
                movlw   d'35'                   ;impostazione del ritardo 0,04s, vedi delay
                call    delay
                goto    titol2


rick            movlw   d'10'                   ;ritardo attesa nuovo dato da quello nuovo
                call    delay
                bcf             PORTA, LEDRIC
                bsf             PORTA, LEDATT   ;accende il LED attesa ck
ctrlck          movlw   CKMAX
                movwf   TIMECK          ;resetta il tempo di massimo attesa ck
                btfsc   PORTA, CK
                call    ricdato                 ;chiama la routine che ricava il dato
                goto    ctrlck


ricdato         bsf     INTCON, T0IE    ;abilitale interruzioni dal TMR0
                bcf             PORTA, LEDATT
                bsf             PORTA, LEDRIC   ;accende LED ricez
```

```
bit0a       btfsc   PORTA, CK
            goto    bit0a
            movlw   d'13'
            call    delay           ;chiamo ritardo di 2ms, per essere sicuro che il ck non interferisca il dato
nell'AM
            btfsc   PORTA, DATI
            bsf             DATO,0          ;mette 1 se necessario al bit 0 del registro del dato
            btfss   PORTA, DATI
            bcf             DATO,0          ;mette 0 se necessario al bit 0 del registro del dato
bit1a       btfss   PORTA, CK       ;controlla il secondo impulso di ck
            goto    bit1a
bit1b       btfsc   PORTA, CK       ;controlla che il secondo impulso sia sceso
            goto    bit1b
            movlw   d'13'
            call    delay           ;chiamo ritardo di 2ms, per essere sicuro che il ck non interferisca il dato
nell'AM
            btfsc   PORTA, DATI
            bsf             DATO,1          ;mette 1 se necessario al bit 1 del registro del dato
            btfss   PORTA, DATI
            bcf             DATO,1          ;mette 0 se necessario al bit 1 del registro del dato
bit2a       btfsc   PORTA, CK       ;controlla il terzo impulso di ck
            call    bit2b
            goto    bit2a
bit2b       btfsc   PORTA, CK       ;controlla che il terzo impulso sia sceso
            goto    bit2b
            movlw   d'13'
            call    delay           ;chiamo ritardo di 2ms, per essere sicuro che il ck non interferisca il dato
nell'AM
            btfsc   PORTA, DATI
            bsf             DATO,2          ;mette 1 se necessario al bit 2 del registro del dato
            btfss   PORTA, DATI
            bcf             DATO,2          ;mette 0 se necessario al bit 2 del registro del dato
bit3a       btfsc   PORTA, CK       ;controlla il quarto impulso di ck
            call    bit3b
            goto    bit3a
bit3b       btfsc   PORTA, CK       ;controlla che il quarto impulso sia sceso
            goto    bit3b
            movlw   d'13'
            call    delay           ;chiamo ritardo di 2ms, per essere sicuro che il ck non interferisca il dato
nell'AM
            btfsc   PORTA, DATI
            bsf             DATO,3          ;mette 1 se necessario al bit 3 del registro del dato
            btfss   PORTA, DATI
            bcf             DATO,3          ;mette 0 se necessario al bit 3 del registro del dato
bit4a       btfsc   PORTA, CK       ;controlla il quinto impulso di ck
            call    bit4b
            goto    bit4a
bit4b       btfsc   PORTA, CK       ;controlla che il quinto impulso sia sceso
            goto    bit4b
            movlw   d'13'
            call    delay           ;chiamo ritardo di 2ms, per essere sicuro che il ck non interferisca il dato
nell'AM
            btfsc   PORTA, DATI
            bsf             DATO,4          ;mette 1 se necessario al bit 4 del registro del dato
            btfss   PORTA, DATI
            bcf             DATO,4          ;mette 0 se necessario al bit 4 del registro del dato
bit5a       btfsc   PORTA, CK       ;controlla il sesto impulso di ck
            call    bit5b
            goto    bit5a
bit5b       btfsc   PORTA, CK       ;controlla che il sesto impulso sia sceso
            goto    bit5b
            movlw   d'13'
            call    delay           ;chiamo ritardo di 2ms, per essere sicuro che il ck non interferisca il dato
nell'AM
            btfsc   PORTA, DATI
            bsf             DATO,5          ;mette 1 se necessario al bit 5 del registro del dato
            btfss   PORTA, DATI
            bcf             DATO,5          ;mette 0 se necessario al bit 5 del registro del dato
bit6a       btfsc   PORTA, CK       ;controlla il settimo impulso di ck
            call    bit6b
            goto    bit6a
bit6b       btfsc   PORTA, CK       ;controlla che il settimo impulso sia sceso
            goto    bit6b
            movlw   d'13'
            call    delay           ;chiamo ritardo di 2ms, per essere sicuro che il ck non interferisca il dato
nell'AM
            btfsc   PORTA, DATI
```

```
                    bsf         DATO,6              ;mette 1 se necessario al bit 6 del registro del dato
                    btfss       PORTA, DATI
                    bcf         DATO,6              ;mette 0 se necessario al bit 6 del registro del dato
bit7a               btfsc       PORTA, CK           ;controlla il ottavo impulso di ck
                    call        bit7b
                    goto        bit7a
bit7b               btfsc       PORTA, CK           ;controlla che il ottavo impulso sia sceso
                    goto        bit7b
                    movlw       d'13'
                    call        delay               ;chiamo ritardo di 2ms, per essere sicuro che il ck non interferisca il dato
nell'AM
                    btfsc       PORTA, DATI
                    bsf         DATO,7              ;mette 1 se necessario al bit 7 del registro del dato
                    btfss       PORTA, DATI
                    bcf         DATO,7              ;mette 0 se necessario al bit 7 del registro del dato
                    bcf         PORTA, LEDRIC
                    bcf         INTCON, T0IE        ;ferma le interruzioni dal TMR0
                    call        invLCD

invLCD              call ctrlpar                    ;chiama il controllo di parità degli 1
                    movf        DATO,0              ;metto il dato in W, pronto per il controllo
                    xorlw       CONTROLLO1          ;esegue il controllo 1
                    btfsc       STATUS,Z            ;controlla il registro Z, se 0 salta
                    call        ES1                 ;se è 1 chiama il controllo 1
                    movf        DATO,0              ;metto il dato in W, pronto per il controllo
                    xorlw       CONTROLLO2          ;esegue il controllo 2
                    btfsc       STATUS,Z            ;controlla il registro Z, se 0 salta
                    call        ES2                 ;se è 1 chiama il controllo 2
                    movf        DATO,0              ;metto il dato in W, pronto per il controllo
                    xorlw       CONTROLLO3          ;esegue il controllo 3
                    btfsc       STATUS,Z            ;controlla il registro Z, se 0 salta
                    call        ES3                 ;se è 1 chiama il controllo 3
                    movf        DATO,0              ;metto il dato in W, pronto per il controllo
                    xorlw       CONTROLLO4          ;esegue il controllo 4
                    btfsc       STATUS,Z            ;controlla il registro Z, se 0 salta
                    call        ES4                 ;se è 1 chiama il controllo 4
                    movf        DATO,0              ;metto il dato in W, pronto per il controllo
                    xorlw       CONTROLLO5          ;esegue il controllo 1
                    btfsc       STATUS,Z            ;controlla il registro Z, se 0 salta
                    call        ES5                 ;se è 1 chiama il controllo 1

;*****************************Lavoro cursore***********************************

                    btfsc       STATLCD,7           ;controlla se lavora in clone
                    goto        jclone
                    movf        CURS3,0             ;cursore in W
                    xorlw       20H
                    btfsc       STATUS,Z            ;se il cursore è alla seconda riga colonna 16
                    call        LCDgiu
                    call        cp                  ;posiziona il cursore
                    movlw       0x0E                ;rendi visibile il cursore (trattino)
                    call        sc
jclone              movf        CURS3,0             ;cursore in W
                    call        cp
                    movf        DATO,0              ;rimetto il dato ricevuto in W
                    call        sd                  ;se arriva fin qua è una lettera, la visualizza
                    incf        CURS3,1             ;incrementa il cursore
                    call        rick                ;ora aspetta un nuovo dato

;controllo dei tre display LCD di visualizzazione caratteri
LCDgiu              btfsc       STATLCD,0           ;testa il primo LCD
                    movlw       b'00000010'         ;abilita l'LCD 2
                    btfsc       STATLCD,1           ;testa il secondo LCD
                    movlw       b'00000100'         ;abilita l'LCD 3
                    btfsc       STATLCD,2           ;testa il terzo LCD
                    movlw       b'00000001'         ;abilita l'LCD 1
                    movwf       MEMLCD              ;copia W nella memoria LCD (che sarà poi copiata
                    movlw       b'00000111'         ;abilita i 3 LCD
                    movwf       STATLCD
                    movlw       0x0C                ;nascondi il cursore in tutti gli LCD
                    call        sc
                    movf        MEMLCD,0            ;in W
                    movwf       STATLCD             ;riabilita il giusto LCD
                    movlw       0x0E                ;cursore visibile
                    call        sc
                    movlw       0x00                ;cursore 0
                    movwf       CURS3
```

```
                    return

LCDsu           btfsc   STATLCD,0           ;testa il primo LCD
                movlw   b'00000100'         ;abilita l'LCD 3
                btfsc   STATLCD,1           ;testa il secondo LCD
                movlw   b'00000001'         ;abilita l'LCD 1
                btfsc   STATLCD,2           ;testa il terzo LCD
                movlw   b'00000010'         ;abilita l'LCD 2
                movwf   MEMLCD              ;copia W nella memoria LCD (che sarà poi copiata
                movlw   b'00000111'         ;abilita i 3 LCD
                movwf   STATLCD
                movlw   0x0C                ;nascondi il cursore in tutti gli LCD
                call    sc
                movf    MEMLCD,0            ;in W
                movwf   STATLCD                     ;riabilita il giusto LCD
                movlw   0x0E                ;cursore visibile
                call    sc
                movlw   0x1F                ;cursore 0
                movwf   CURS3
                return


;Routine di controllo caratteri non standard ASCII
;***********************************************
;Controllo 3, Freccia a destra
ES1             movlw   d'1'                ;incremento di 1 il cursore
                addwf   CURS3,1             ;metto in F
                movf    CURS3,0             ;metto in W
                xorlw   20H
                btfsc   STATUS,Z            ;se il cursore è alla seconda riga colonna 16
                call    LCDgiu
                call    cp                          ;posiziona cursore
                call    rick                ;ora aspetta un nuovo dato
;Controllo 2, clear LCD
ES2             bsf             STATLCD,0           ;abilita i 3 lcd per il clear
                bsf             STATLCD,1
                bsf             STATLCD,2
                call    cl                          ;pulisci lcd
                movlw   0x0C                ;nascondi il cursore in tutti gli LCD
                call    sc
                movlw   0
                movwf   CURS3               ;aggiorna il cursore in 0.0
                btfsc   STATLCD,7           ;funziona in clone?
                call    rick                ;si,ora aspetta un nuovo dato
                movlw   b'00000001'         ;no, abilita solo il primo LCD
                movwf   STATLCD
                movlw   0x0E                ;abilita cursore solo sul primo LCD
                call    sc
                call    rick
;Controllo 3, cancella l'ultimo carattere
ES3             decf    CURS3,1         ;decremento di 1 il registro del cursore
                movf    CURS3,0         ;cursore in W
                xorlw   0xFF
                btfsc   STATUS,Z        ;se il cursore è in overflow rispetto 0.0
                call    LCDsu
                movf    CURS3,0         ;cursore in W
                call    cp                      ;posiziono il cursore
                movlw   0x0E            ;rendi visibile il cursore (trattino)
                call    sc
                movlw   ' '                     ;"scrivo vuoto"
                call    sd
                movf    CURS3,0
                call    cp                      ;riposiziono cursore per effetto
                call    rick            ;ora aspetta un nuovo dato
;cambio riga cursore
ES4             btfss   CURS3,4         ;controlla il bit della riga (se 0 riga1, se 1 riga2)
                goto    ES4b
                call    LCDgiu          ;cambia LCD, +1
                call    cp
                call    rick
ES4b        movlw   0x10                ;posizione cursore in seconda riga pos.0
                movwf   CURS3           ;salvo la nuova posizione
                call    cp
                call    rick            ;ora aspetta un nuovo dato
;entra/esci in modalità clone
ES5             bsf             STATLCD,0
                bsf             STATLCD,1
```

```
                        bsf             STATLCD,2       ;setto gli lcd per il clear
                        call    cl
                        movlw   0x00            ;posiziona il cursore a 0.0
                        movwf   CURS3
                        btfsc   STATLCD,7       ;è già in clone
                        goto    ES5b
                        movlw   b'10000111'     ;non era in clone, ora setta il bit 7, e metti in clone
                        movwf   STATLCD
                        movlw   0x0E            ;rendi visibile il cursore (trattino)
                        call    sc
                        call    rick
ES5b                    movlw   0x0C            ;nascondi cursore in tutti gli LCD
                        call    sc
                        movlw   b'00000001'     ;era già in clone, ora abilita solo LCD1
                        movwf   STATLCD
                        movlw   0x0E            ;rendi visibile il cursore (trattino)
                        call    sc
                        call    rick            ;ora aspetta un nuovo dato

;*********************controllo parità dei bit*****************************
;****************gli uno del byte inviato devono essere pari*****************

ctrlpar                 clrf    cpar            ;pulisce il registro del conteggio di parità
                        btfsc   DATO,0
                        incf    cpar,1          ;conta gli uno...
                        btfsc   DATO,1
                        incf    cpar,1
                        btfsc   DATO,2
                        incf    cpar,1
                        btfsc   DATO,3
                        incf    cpar,1
                        btfsc   DATO,4
                        incf    cpar,1
                        btfsc   DATO,5
                        incf    cpar,1
                        btfsc   DATO,6
                        incf    cpar,1
                        btfsc   DATO,7
                        incf    cpar,1
                        btfsc   cpar,0          ;tasta l'ultimo bit del contatore degli 1, se 1=dispari, se 0=pari
                        call    segnpar         ;invio sagnalazione parità non rispettata
                        bcf             DATO,7          ;rimetti a zero il bit 7 (come da codice ascii)
                        return                                  ;i bit uno sono pari!!torna all'invio


segnpar                 nop                                     ;invio al display stato l'avviso che c'è stata un controllo di parità non
risperrato
                        call    rick
                        return


;*********************************RITARDO************************************
;********moltiplicare alla terza W, per ottenere il ritardo in us)***********
delay                   movwf   CONT3
rit3                    movwf   CONT2
rit2                    movwf   CONT1
rit1                    decfsz  CONT1,1
                        goto    rit1
                        decfsz  CONT2,1
                        goto    rit2
                        decfsz  CONT3,1
                        goto    rit3
                        return

                        INCLUDE"liblcd4.inc"

                        End
```

2h Listato Assembler Libreria LCD

Ed ora il listato delle librerie di controllo LCD (per buona parte modificate). Consiglio la lettura dei commenti iniziali, utili per comprendere il seguito.

```
;*************************************************
;Mazzetti Francesco 5^B4 ©
;Progetto d'esame "STUDIO DEI MEZZI TRASMISSIVI"
;Libreria LCD per comunicazione fino a 4 LCD in cascata
;*************************************************


;ricordarsi di mettere tutti i PIN per gli LCD come output!!!


; COLLEGAMENTI LCD-PIC
;
; LCD DB7 (14)          -->   PIC RB7 (13)
; LCD DB6 (13)          -->   PIC RB6 (12)
; LCD DB5 (12)          -->   PIC RB5 (11)
; LCD DB4 (11)          -->   PIC RB4 (10)
; LCD EN  (6)           -->   PIC RB2,RA0,RA1,RA2,RA4  ;uno per ogni LCD
; LCD RS  (4)           -->   PIC RB3 (9)
;
; INTESTAZIONE da mettere nel programma originale
;
; PROCESSOR             16F84
; RADIX                     DEC
; INCLUDE               "p16f84.inc"
;
; PIN11                     EQU   4
; PIN12                     EQU   5
; PIN13                     EQU   6
; PIN14                     EQU   7
; RS                        EQU   3
; EN4                       EQU     4             ;su RA4 del PIC
; EN3                       EQU     0             ;su RA0 del PIC
; EN2                       EQU     1             ;su RA1 del PIC
; EN1                       EQU     2             ;su RB2 del PIC
;
;                         ORG     0CH
;
; lsb                       RES     1
; msb                       RES     1
; tmp                       RES     1
; tmp1                      RES     1
; STATLCD               RES1        ;questa registro viene utilizzato come stato:
                                    ;in esecuzione, controlla i primi 4 bit
                                    ;del registro e esegue le varie le istruzione
                                    ;richieste, solo negli LCD in cui è presente "1".

;al termine del programma originale, prima di "end"
;copiare tutte le librerie di seguito, o inserire questo file nel progetto
;e scrivere    INCLUDE    "liblcd4.inc"  (comprese virgolette)

;***********************************************STATLCD*****************************
;
;BIT    | 0°| 1°| 2°| 3°| 4°| 5°| 6°|7°|
;
;       | x | x | x | x |  0 |  0 | 0 |  x| ← (Ultimo bit utilizzato per la modalità clone)
;
;LCD    | 1°| 2°| 3°| 4°|             ;corrispondenza LCD abilitati


;ricordarsi prima di tutto di chiamare la routine init su TUTTI i display LCD!!
;in caso contrari, i display LCD non inizializzati, non funzioneranno!
;tutte le altre routine (clear, invio comando, invio carattere e posizionamento
;cursore) possono essere chiamate separatamente


; ************************************************
; init
; Da usare all'inizio del programma per inizializzare l'LCD
; Registri usati:
; \
```

```
; Subroutine richiamate:
; rit
; enable
; sc
; ******************************************************

init                    movlw       30
                        call rit                ; Ritardo 30 ms

                        bcf  PORTB,RS           ; RS = 0 --> Dati

                        bsf    PORTB,PIN11
                        bsf    PORTB,PIN12
                        bcf    PORTB,PIN13
                        bcf    PORTB,PIN14      ; PORTB --> 0011XXXX
                        btfsc       STATLCD,0       ;LCD 1?
                        bsf         PORTB,EN1
                        btfsc       STATLCD,1       ;LCD 2?
                        bsf         PORTB,EN2
                        btfsc       STATLCD,2       ;LCD 3?
                        bsf         PORTB,EN3
                        btfsc       STATLCD,3       ;LCD 4?
                        bsf         PORTA,EN4
                        movlw       5
                        call rit                ; Ritardo 5 ms
                        bcf         PORTB,EN1
                        bcf         PORTB,EN2
                        bcf         PORTB,EN3
                        bcf         PORTA,EN4
                        movlw       1
                        call rit                ; Ritardo 1 ms

                        call enable

                        call enable

                        bcf    PORTB,PIN11
    bsf    PORTB,PIN12
    bcf    PORTB,PIN13
    bcf    PORTB,PIN14  ; PORTB --> 0010XXXX

                        call enable

                        movlw       0x28            ; 4-bit
                        call sc

                        movlw       0x06
                        call sc

                        movlw       0x0C            ;nascondi il cursore
                        call sc
                        call cl

                        return

; ******************************************************
; cp
; Cursor Position - Mette il cursore in una certa posizione
; Registri usati:
; W                     -->  D7-D4 righe,       D3,D0 colonne
; tmp
; tmp1
; Subroutine richiamate:
; sc
; ******************************************************
cp                      movwf       tmp

                        movlw       0x80
                        movwf       tmp1            ; Metti 10000000 in tmp1

                        movf   tmp,W            ; Metti tmp in W
    andlw   0x0F            ; W --> 0000DDDD
    iorwf   tmp1,F          ; tmp1 --> 1000DDDD

    btfsc   tmp,4           ; Se il bit 4 di tmp = 0 salta
    bsf     tmp1,6          ; altrimenti tmp1 diventa 1100DDDD
```

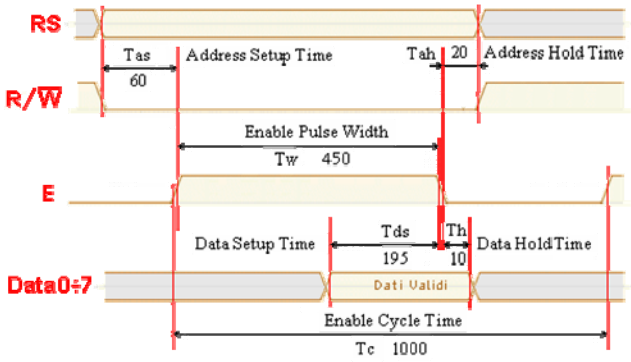
```
        movf    tmp1,W          ; Metti tmp1 in W
        call    sc              ; Manda il comando 1X00DDDD

        return

; ******************************************************
; cl
; Clear - Pulisce il display e mette il cursore all'inizio
; Registri usati:
; \
; Subroutine richiamate:
; sc
; rit
; ******************************************************

cl                      movlw   0x01
                        call sc ; CLEAR

                        movlw   2
                        call rit ; Ritardo 2 ms

                        movlw   80H
                        call sc ; Posizione Iniziale

                        return

; ******************************************************
; sd
; Send data - Invia un dato al display
; Registri usati:
; W                     -->  dato da inviare
; Subroutine richiamate:
; sb
; ******************************************************

sd                      bsf  PORTB,RS
                        call sb
                        return


; ******************************************************
; sc
; Send Command - Invia un comando al display
; Registri usati:
; W                     -->  Comando da inviare
; Subroutine richiamate:
; sb
; ******************************************************

sc                      bcf  PORTB,RS
                        call sb
                        return

; ******************************************************
; sb
; Send byte - Invia un byte sull'msb di PORTB
; Registri usati:
; tmp
; Subroutine richiamate:
; enable
; ******************************************************

sb                      movwf   tmp
                        movlw   B'00001111'
                        andwf   PORTB,F         ;Metto a 0 i 4 bit MSB di PORTB
                        movf    tmp,W
                        andlw   B'11110000'     ;Metto a 0 i 4 bit LSB del byte da inviare in W
                        iorwf   PORTB,F         ;OR fra PORTB (0000XXXX) e W (DDDD0000)
                        call enable             ;Impulso di Enable
                        movlw   B'00001111'
                        andwf   PORTB,F         ;Metto a 0 i 4 bit MSB di PORTB
                        swapf   tmp,W           ;Metto tmp1 in W con LSB invertito con MSB
                        andlw   B'11110000'     ;Metto a 0 i 4 bit LSB del byte da inviare in W
                        iorwf   PORTB,F         ;OR fra PORTB (0000XXXX) e W (DDDD0000)
                        call enable             ;Impulso di Enable

                        return
```

```
; ******************************************************
; enable
; Manda un impulso di ENABLE della durata di 1ms
; Registri usati:
; \
; Subroutine richiamate:
; rit
; ******************************************************

enable                  bcf  STATUS,RP0         ;banco 0
                        btfsc       STATLCD,0           ;LCD 1?
                        bsf                 PORTB,EN1           ;metto a 1 enable
                        btfsc       STATLCD,1           ;LCD 2?
                        bsf                 PORTB,EN2           ;metto a 1 enable
                        btfsc       STATLCD,2           ;LCD 3?
                        bsf                 PORTB,EN3           ;metto a 1 enable
                        btfsc       STATLCD,3           ;LCD 4?
                        bsf                 PORTA,EN4           ;metto a 1 enable
                        movlw       1                   ;metto 1 in w
                        call        rit                 ;ritardo
                        bcf                 PORTB,EN1           ;rimetto a 0 enable
                        bcf                 PORTB,EN2           ;rimetto a 0 enable
                        bcf                 PORTB,EN3           ;rimetto a 0 enable
                        bcf                 PORTA,EN4           ;rimetto a 0 enable
                        movlw       1
                        call        rit

                        return

; ******************************************************
; rit
; Ciclo di ritardo programmabile
; Registri usati:
; W                     -->  Numero di ms (per CLK = 4MHz)
; msb                   -->  (uso interno)
; lsb                   -->  (uso interno)
; Subroutine richiamate:
; \
; ******************************************************

rit                     movwf   msb
        clrf    lsb
rit_loop
                        nop
        decfsz  lsb,F
        goto    rit_loop
        nop
        decfsz  msb,F
        goto    rit_loop

        return
```

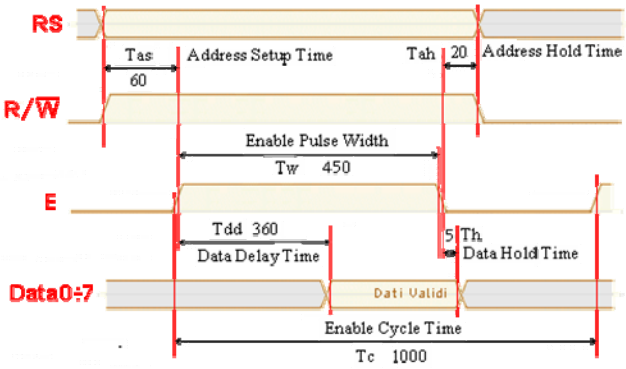
2i Comunicazione display LCD

I display che utilizzo sono di tipo standard, cioè utilizzano una comunicazione semplice e universale per molti di essi. Di seguito riporto alcune tabelle che ne spiegano i registri interni, le procedure di inizializzazione e i comandi:

in scrittura



in lettura



Istruzione	Input		Data Bus								Descrizione	Tempo max
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0		
Nop	0	0	0	0	0	0	0	0	0	0	Nessuna operazione	0
Cancella Display			0	0	0	0	0	0	0	1	Cancella il visualizzatore e fissa Address DDRam Counter a 0	1,52 ms
Cursore a Capo			0	0	0	0	0	0	1	x	fissa Address DDRam Counter a 0 senza modificare la DDRAM display spostato alla posizione iniziale	1,52 ms
Modo d'accesso dei caratteri			0	0	0	0	0	1	I	S	Address >> I=0 Decremen I=1 Incremen display >> S=0 è bloccato S=1 scorre	37 us
Controllo Display			0	0	0	0	1	D	C	B	display >> D=0 spento D=1 acceso cursore >> C=0 invisibile C=1 visibile carattere>>B=0 fisso B=1 lampeggiante	37 us
Scorrimento cursore e display			0	0	0	1	S/C	R/L	x	x	S/C=0 il cursore si muove S/C=1 il display scorre R/L=0 verso sinistra R/L=1 verso destra (non modifica la DDRAM)	37 us
Impostazioni Parametri			0	0	1	DL	N	F	x	x	interfaccia >> DL=0 a 4 bit DL=1 a 8 bit Display >> N=0 a 1 linea N=1 a 2 linee Matrice carattere >> F=0 5x7 F=1 5x10	37 us
Indirizzo CGram			0	1	Character Generator RAM						Imposta indirizzo CGRAM per R/W dati	37 us
Indirizzo DDram			1	Display Data RAM Address							Imposta l'indirizzo DDRAM per R/W dati	37 us
Lettura Bit di Stato Lettura Indirizzo		1	BF	Address Counter							Legge Flag di Busy e'Address Counter BF=0 comando eseguito BF=1 comando in esecuzione	1 us
Scrittura Dato	1	0	Dato da Scrivere								Scrive dati dalla CGRAM o DDRAM	37 us
Lettura Dato		1	Dato da Leggere								Legge dati nella CGRAM o DDRAM	37 us

Istruzione	Input		Codice Operativo								Data Bus				byte	Descrizione
	RS	R/W	7	6	5	4	3	2	1	0	D7	D6	D5	D4		
Function Set	0	0	0	0	1	0	x	x	x	x	0	0	1	0	20H	interfaccia a 4 bit
Effettivo Function Set	0	0	0	0	1	**DL** 0	**N** 0	**F** 0	x	x	0 0	0 0	1 x	0 x	20H 00H	interfaccia a 4 bit una linea matrice 5x7
Display Control	0	0	0	0	0	0	1	**D** 1	**C** 0	**B** 0	0 1	0 1	0 0	0 0	00H C0H	display acceso cursore invisibile cursore a trattino
Display Clear	0	0	0	0	0	0	0	0	0	1	0 0	0 0	0 0	0 1	00H 10H	azzera la DDRan
Enter Mode Set	0	0	0	0	0	0	0	1	**I** 1	**S** 0	0 0	0 1	0 1	0 0	00H 60H	autoincremento cursore a destra display bloccato

Di seguito vengono descritti i principali comandi:

OpCode 08H - 0BH	spegne il display
OpCode 0CH OpCode 0DH	rende invisibile il cursore
OpCode 0EH	rende visibile il cursore come trattino
OpCode 0FH	rende visibile il cursore come blocco lampeggiante

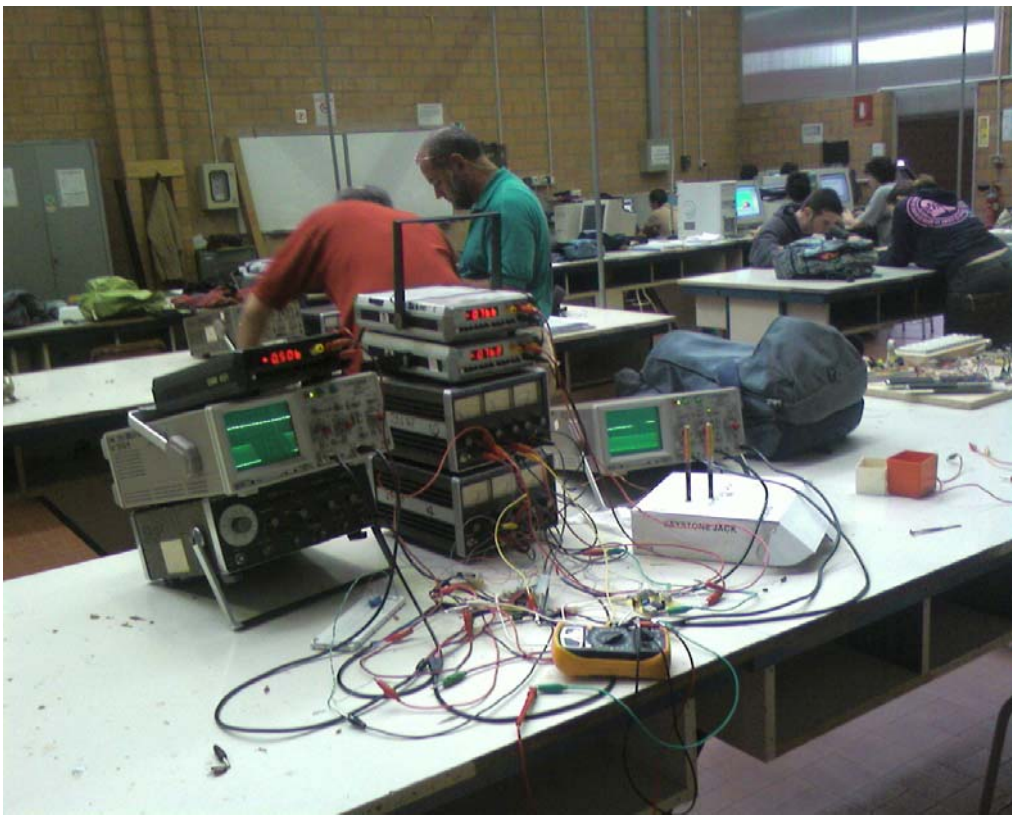
In dettaglio per interfaccia a 4 bit:	
OpCode 20H-23H	predispone display a 1 linea, con matrice di carattere 5x7
OpCode 24H-27H	predispone display a 1 linea, con matrice di carattere 5x10
OpCode 28H-2BH	predispone display a 2 linee, con matrice di carattere 5x7
OpCode 2CH-2FH	predispone display a 2 linee, con matrice di carattere 5x10

OpCode 80H-A7H OpCode C0H-E7H	con visualizzatori a 2 linee (N=1) gli indirizzi per la prima vanno da 00H (=0) a 27H (=39) e per la seconda vanno da 40H (=64) a 67H (=103) per cui i codici che puntano le locazioni DDRam sono: sulla prima linea, a partire dalla prima (10000000 = 80H) fino all'ultima, l'ottantesima (10100111 = A7H). sulla seconda linea, a partire dalla prima (11000000 = C0H) fino all'ultima, l'ottantesima (11100111 = E7H).

2l Regolazioni e test

Una volta realizzati i circuiti, saldati ed alimentati, è necessario eseguire collaudo di funzionamento prima di implementarli nel circuito finale e in alcuni casi anche tararli tramite appositi trimmer.
I circuiti che presentato resistenze variabili sono ben 6:
-Codifica AM
-Trasmissione AM
-Ricezione AM
-Ricezione fibra ottica
-ricezione via cavo parallelo
-ricezione dati
Ben tutti e tre i circuiti riguardanti la radio posseggono dei trimmer di regolazione, questo perché, in fase di realizzazione e progettazione non potevo essere sicuro di certe tensioni e correnti che potevano dipendere da tanti fattori, come la potenza delle antenne o la distanza delle stesse, la corrente in base per il modulatore a transistor ottimale, ecc.
Nel blocco ricezione fibra ottica, i trimmer sono stati utilizzati semplicemente per permettere una rilevazione più accurata, dipendente dal valore di resistenza che donava la fotoresistenza in caso di

“luce” e in caso di “buio”. Inoltre poteva dipendere dall’efficienza della fibra, dalla distanza tra la fibra ottica. Tutto questo ha portato alla necessità di utilizzare un circuito flessibile.
Nella ricezione cavo parallelo, i due trimmer utilizzati sono rispettivamente per “l’astabile di generazione clock”, e il “monostabile di abilitazione clock”.



Teoricamente questa flessibilità sarebbe stata inutile in quanto, con dei semplici calcoli, si sarebbe potuto calcolare esattamente i tempi, regolandoli con precisione ma in modo fisso.
Difatti, se avessi deciso di cambiare i tempi di trasmissione e clock, sarebbe stato molto più difficile agire su questo blocco. Inoltre da elettronico con un minimo di esperienza, so bene che il valore nominale di una resistenza e di un condensatore (componenti utilizzati per la regolazione dei tempi nel NE555) sono puramente teorici e con errori anche considerevoli, difatti, i componenti da me utilizzati posseggono una tolleranza del 5% per le resistenze e 10% per i condensatori.
Nel blocco ricezione dati, il trimmer permette unicamente la regolazione del contrasto nei display LCD. Funzionalità utile e sicuramente necessaria (ad esempio cambiando LCD la tensione necessaria per un contrasto ottimale potrebbe non essere la stessa).
I test e le regolazioni sono state effettuate con strumenti forniti per lo più dalla scuola, in quanto molto costosi:
Alimentatori, Multimetri, Oscilloscopi e generatori di funzione, anche più contemporaneamente.
Nella foto, gli oscilloscopi mostrano figure distorte a causa del mancato sincronismo con la fotocamera.

2m Vincoli

I vincoli sono determinate regole che devono essere rispettate.
Ogni circuito possiede i suoi piccoli vincoli, che dipendono spesso dai tipi di componenti utilizzati, dall’utilità e da altri fattori.
Questi vincoli possono essere sia utili che indispensabili.
Per esempio il posizionamento del connettore dell’alimentazione può essere posizionato ovunque, ma se viene posizionato centralmente, il collegamento sarà scomodo e poco estetico, in più si rischia di creare falsi contatti. Inoltre per migliorare ancora di più l’aspetto grafico, i connettori possono essere rivolti verso la propria fonte.
Tutti questi sono piccoli accorgimenti, che però permettono di migliorare l’aspetto estetico e funzionale.

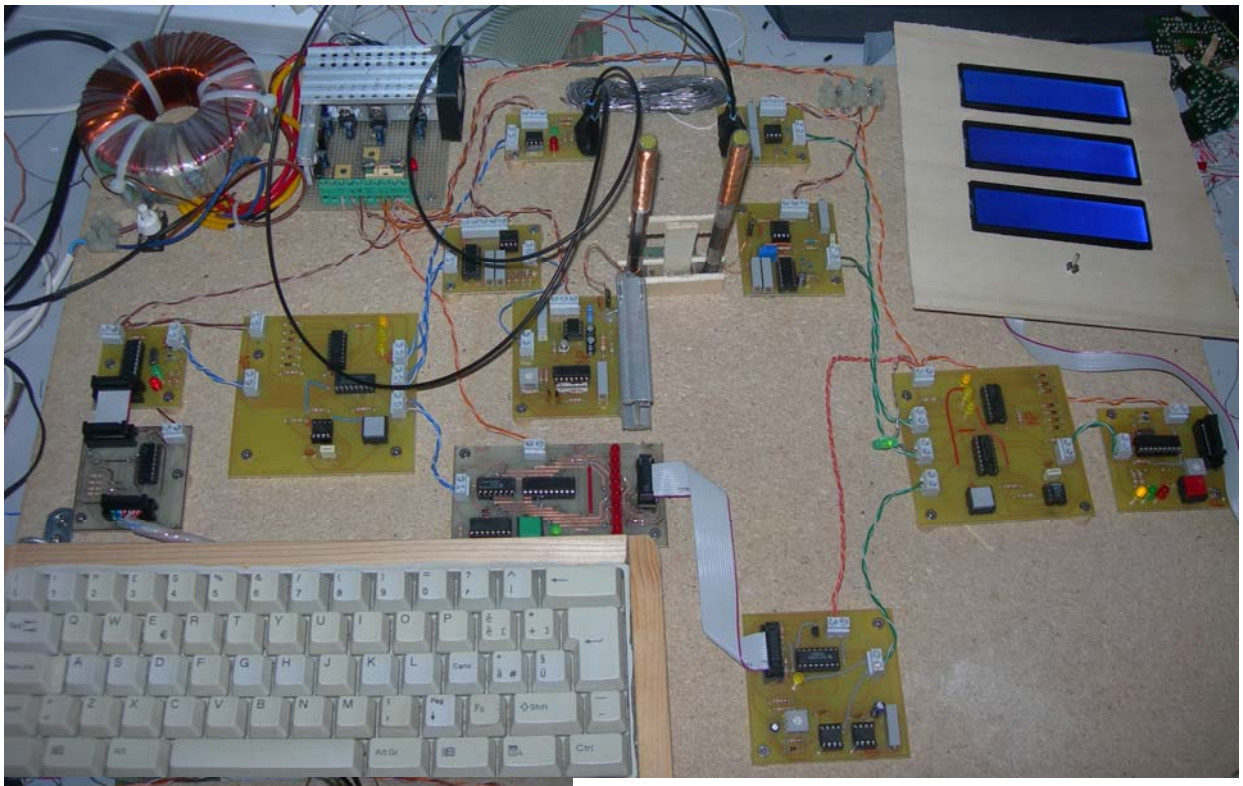
Nel mio progetto sono presenti vari vincoli. I principali e quindi più importanti sono quelli che chiamo rigidi, in quanto necessari, i restanti sono morbidi:
-Posizionamento trimmer (rigido)
Molti trimmer da me utilizzati hanno una regolazione orizzontale, cioè è possibile effettuare le modifiche solamente agendo con un cacciavite in modo orizzontale. Se per esempio la parte della vite viene coperta da un condensatore o da qualsiasi componente leggermente alto, questa regolazione sarà impossibile.
Per questo tutti i trimmer a regolazione orizzontale sono rivolti con la parte della vite verso l'esterno, senza nessuna barriera.
-posizionamento connettori (morbido)
Come spiegato prima, tutti i connettori saranno rivolti verso la loro fonte, un po' come nello schema a blocchi. Cioè i connettori del *selettore uscite* sarà verso destra, in direzione delle fonti di trasmissione. Considerando che l'alimentatore è nella parte alta del pannello, tutti i connettori di alimentazione saranno verso l'alto del circuito.
Una nota particolare va a tutti i connettori delle piattine (cavo con più poli tutti raggruppati in fila). Difatti quest'ultimi posseggono un verso di collegamento. Se ad esempio questi connettori non fossero in linea con il loro estremo, sarebbe necessario piegare il filo, creando difficoltà di collegamento e un groviglio antiestetico.
-posizionamento circuiti (morbido)
Tutti i circuiti sono posizionati seguendo la logica dello schema a blocchi, in modo di avere i blocchi di trasmissione di fronte a quelli di ricezione e seguire il passaggio di byte
da sinistra a destra (come si scrive).
-posizione dissipatori (rigido)
Nel circuito sono stati utilizzato tre dissipatori, essi erano assolutamente necessari. Essi sono stati ricavati da scarti di vari apparecchi rotti (come alimentatori per PC). Essi sono quindi stati segati per rispettare un determinato posizionamento. Del resto, i circuiti devono lasciare lo spazio per il loro posizionamento e allacciamento ai componenti beneficiari (i 5 stabilizzatori di tensione e il transistor BDX53 (nel blocco *trasmissione AM*). Tali dissipatori inoltre non devono essere a contatto con altri componenti in quanto il loro calore potrebbe rovinare i contenitori e quindi il loro funzionamento o potrebbe creare cortocircuiti, in quanto spesso il terminale cui viene collegato il dissipatore è direttamente collegato ad uno dei PIN (ad esempio nel LM7805, il terminale è collegato a massa).
I suddetti vincoli nonostante possano apparire di secondaria importanza, sono da considerarsi indispensabili. Oltre a questi ne esistono molti di più, quasi intrinseci??? nel modo di pensare di un elettronico, quindi non menzionati.

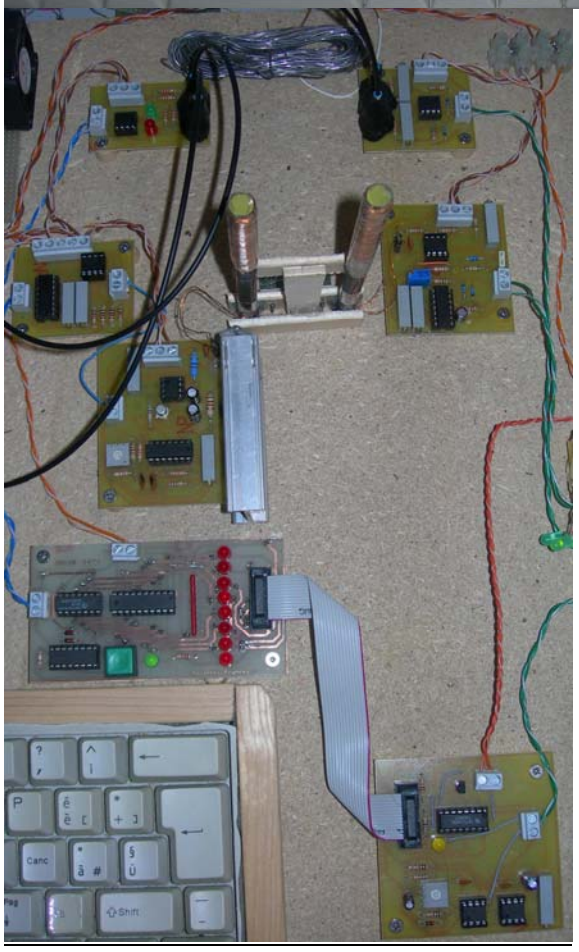
2n Supporti

La base è stata scelta in legno in quanto solida e di facile aggancio per tutti i circuiti. Tale materiale è inoltre completamente isolante, ciò garantisce protezione da falsi contatti tra vari circuiti. Lo spessore è di 2cm, l'altezza di 45cm e la larghezza di 62cm.
Per sorreggere le antenne è stato creato una base in legno agganciata ad una cerniera, che permette di inclinare le stesse e abbassarle al livello della base. Cosa simile è stata fatta per i tre Display LCD, agganciati ad una piccola tavola di compensato tagliata su misura.
Tutti i circuiti sono avvitati alla base in legno grazie a viti e spessori che permettono ai circuiti di non essere completamente a contatto con il piano.
Il trasformatore è fisso grazie a supporti metallici ad "L" e fascette, mentre la tastiera possiede uno spessore laterale (sempre in legno) che permette di mantenere distante i pulsanti sotto i tasti e la base. Inoltre offre un comodo aggancio con la struttura.

Conclusione

3a Foto





3b Problemi

In fase di realizzazione sono sorti molti problemi, per lo più risolti. Il primo fu con l'alimentatore, il quale non riusciva ad erogare sufficientemente corrente nel secondario a 6V. Per ovviare a questo, come detto precedentemente, ho aggiunto un secondario calcolando numero di spire e diametro del filo molto empiricamente, garantendo però una corrente massima nettamente superiore alle necessità.
A parte qualche saldatura fatta male e qualche modifica on-board, tutti i circuiti hanno funzionato subito. L'unico circuito, che fino alla fine dimostrava di avere problemi, è quello relativo alla ricezione AM, in quanto, la serie di amplificatori, e forse anche le condizioni ambientali (umidità, temperatura) in un qualche modo non garantivano una taratura certa. Cioè spesso diventava necessario ri-regolare i trimmer di comparazione. In seguito a molte prove, eseguite in diversi giorni, a diverse ore, sembra essersi raggiunta la condizione ottimale.

3c Ringraziamenti

Banalmente ringrazio tutti i miei professori che spesso hanno risolto problemi o suggerito idee utili al complesso, ed in generale hanno permesso la mia crescita personale grazie a tutte le conoscenze trasmessemi in questi anni:
In particolare il professore Di Silverio, per la disponibilità e fiducia datami durante tutto il secondo quadrimestre, ed inoltre alle capacità di "progettista e programmatore assembler" apprese durante le sue lezioni.
Alla professoressa Bolognesi, per moltissimi aiuti teorici e pratici datemi durante le sue ore di lezione, e all'insegnamento (non solo scolastico) ricevuto in due anni.
Al professore Talerico, alle continue critiche e insegnamenti di vita che mi hanno permesso di crescere con coscienza, consentendomi inoltre di apprendere e apprezzare l'elettronica, la programmazione e lo studio dei sistemi fin dal terzo anno di studio (anche se forse non diventerò mai un buon programmatore…).
A tutti gli assistenti tecnici di laboratorio (proff. Cardamone, Cacciari, Veggetti, Felice e Valentino) che in questi vari anni, ed in particolare in questi ultimi mesi, mi hanno fornito aiuto, strumenti e materiale.
Di seguito elenco tutti i restanti, ma non meno importanti professori avuti in questi 5 anni:
Boninsegni (scienze), Mazzoli, Manfredini, Miliè e molti supplenti(italiano) Marmiroli e Tintorri(inglese) Agostini (TDP) Ragazzini, Baiada e Nicolino(Fisica) Piccinini e Biavati (diritto) Frasca e Soverini (disegno tecnico) Gnaccarini, Govi e Poli (ginnastica) Cimino,Affini ,Di Bonaventura (matematica) Fidanza e Bacci (chimica).

Si ringraziano inoltre molti siti web che mi hanno permesso di ampliare le mie conoscenze e quindi migliorare il progetto, anche graficamente. Oltre ai datasheet elenco i principali siti

www.wikipedia.it (generico per molte informazioni e immagini)
www.ilgrix.it (utile per la comunicazione PIC-LCD)
www.giobe2000.it (informazioni generali e immagini LCD)
www.itisrn.it (dove ho ricavato informazioni e immagini per il modulatore e demodulatore)
www.tanzilli.com (dove ho appreso molte informazioni sul PIC)

Mazzetti Francesco