I LETTORI DI BADGE

Per realizzare una chiave elettronica e altro sfruttando una ormai comune tessera magnetica è necessario conoscere la teoria di funzionamento di questi apparecchi, per poterli successivamente interfacciare con un microprocessore in grado di interpretarne i comandi. Vediamo come fare

Paola Sbrana - 1ª parte

Con questo articolo, apriamo una serie di dettagliate documentazioni, con applicazioni pratiche, delle carte magnetiche, dette anche "badge".

Negli ultimi tempi, questo tipo di "chiave" ha avuto un'espansione più ampia del previsto, grazie al fatto di essere facile da leggere, da programmare, poco ingombrante e, per ultimo ma non di minore importanza, poco costosa.

Chi di voi non è mai stato in un albergo dove per chiave della camera non si è trovato un badge? Infatti, la veloce programmazione di queste chiavi consente di cambiare rapidamente le impostazioni di una serratura, senza per questo dover cambiare la serratura stessa.

Vediamolo meglio con un esempio: supponiamo di gestire un albergo e di assegnare ad un nuovo cliente una camera per tre giorni.

Sul badge è possibile inserire informazioni tipo la data di scadenza, l'ora, il nome del possessore.

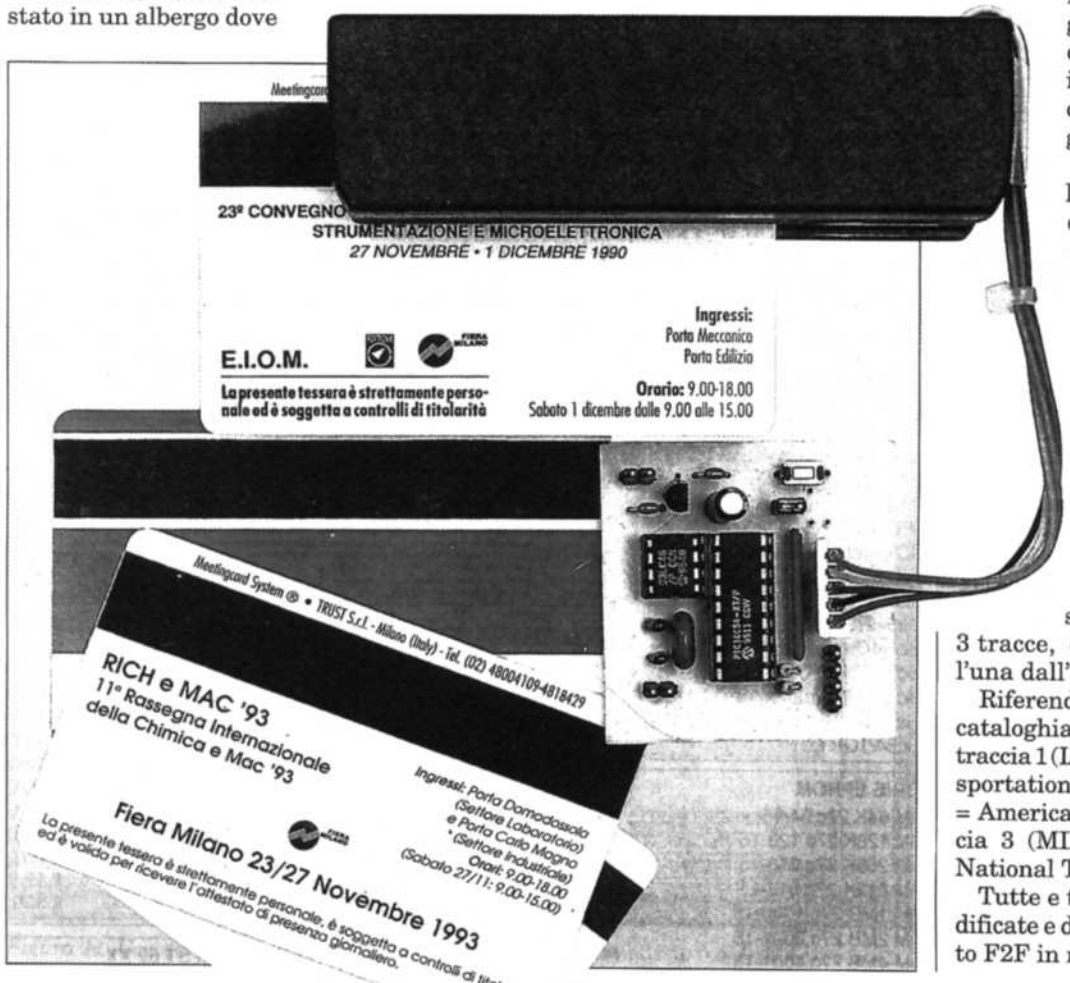
Quando il cliente vuole entrare in camera, deve necessariamente far strisciare il badge nel lettore, e quest'ultimo verifica la correttezza dei dati impostati e memorizzati in un computer centrale. Se uno solo di questi non corrisponde, il cliente non vedrà aprirsi la porta. Altro esempio di carta magnetica oggi comunemente impiegata da molte persone è il bancomat e/o la carta di credito. Anche in questo caso, il lettore deve poter sapere se la tessera inserita è abilitata o meno all'operazione.

Nel primo caso, con il confronto di un numero detto PIN, digitato dal possessore del bancomat, nel secondo caso con la richiesta di informazioni attraverso una linea telematica.

Per ultimo, citiamo il codice fiscale che, da diverso tempo, viene distribuito sotto forma di badge magnetico.

Abbiamo visto quindi che le carte magnetiche sono oggi impiegate un po' in tutti quei casi dove si necessiti di avere un ingresso controllato a basso costo per chiave, facilmente gestibile e personalizzabile.

Se dovessimo scrivere tutte le applicazioni esistenti oggi con le carte magnetiche, non occorrerebbero le pagine di Progetto.



Le carte magnetiche

Vediamo allora come sono fatte queste carte e, soprattutto come possiamo accedervi con semplici operazioni di lettura.

Per prima cosa, diciamo che le carte magnetiche possono essere scritte e lette su 3 tracce, con diverse caratteristiche l'una dall'altra.

Riferendoci allo standard ISO7811, cataloghiamo i tre tipi di traccia in: traccia 1 (IATA = International Air Transportation Association), traccia 2 (ABA = American Banks Association) e traccia 3 (MINTS = Mutual Institutions National Transfor Systems).

Tutte e tre queste tracce vengono codificate e decodificate con il metodo detto F2F in modulazione di frequenza.

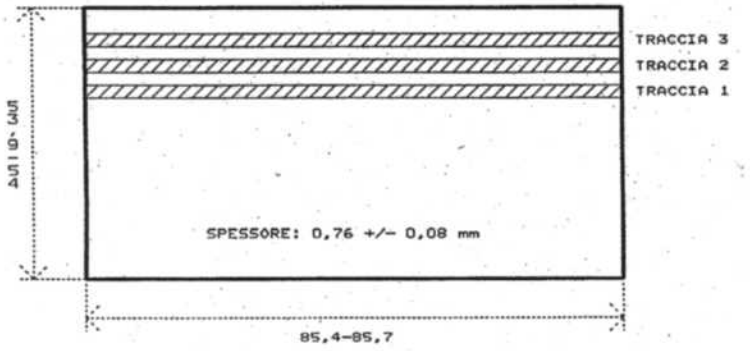


Figura 1. Misure standard ISO7811 per le carte magnetiche

Questo significa che un bit viene codificato con una frequenza se vale 0 e con il doppio di tale frequenza se vale 1 (o viceversa). Se ci pensiamo bene, non è poi una così grossa novità, in quanto i vetusti modem telefonici lavorano con questo metodo già da parecchi anni.

Le tre tracce si differenziano in particolar modo per la densità di scrittura, ovvero per la capacità di contenere più o meno dati digitali.

La traccia 1 ha una densità (come si può vedere nella Tabella 1) di 210 bit per pollice, che corrispondono a 79 caratteri da 7 bit (utili per codifiche ASCII).

La traccia 2 ha una densità di 75 bit per pollice, che corrisponde a 40 caratteri da soli 5 bit (la più impiegata).

La traccia 3 ha una densità di 210 bit per pollice, che corrispondono a 107 caratteri da 5 bit.

Un altro fattore che sembra non influire, ma che invece è di fondamentale importanza, è la grandezza delle carte ed il loro spessore. In Figura 1 potete vedere tali misure. I lettori che abbiamo impiegato per le nostre prove ed applicazioni, forniti dalla EDUE Italia, rispettano gli standard ISO7811 ed in Figura 2 possiamo vederne i timing diagram anche con riferimento al rilevamento degli errori.

Lo standard ISO7811

Dato che tutto ciò che riguarda i badge magnetici deve sottostare allo standard ISO7811, elenchiamo di seguito i riferimenti più importanti.

Tecnica di codifica: la tecnica di codifica è detta delle due frequenze in fase.

Questo metodo permette, nel caso di dati seriali, di sincronizzarsi sul dato stesso. Il dato deve essere registrato come sequenza sincrona di caratteri senza interruzioni.

La registrazione deve avvenire con la saturazione e con la magnetizzazione su una linea parallela al piano della traccia.

La direzione è determinata dall'angolo di registrazione.

In Figura 3 possiamo vedere un esempio di codifica.

Angolo di registrazione: l'angolo di registrazione dovrebbe essere di 0 gradi più o meno 20 rispetto alla normal della striscia magnetica (cioè perpendicolare a questa). Configurazione dei bit: per ogni carattere registrato, viene prima scritto (e quindi letto successivamente) il primo bit meno significante (LSB) e poi gli altri, fino al bit di parità. Set di caratteri delle codifiche: il set di caratteri per la prima traccia, differisce da quello delle tracce 2 e 3.

Nelle Tabelle 2 e 3 vediamo i due set completi.

Le applicazioni più conosciute

Tra le tantissime applicazioni che vedono impiegate, le tessere magnetiche, ne citiamo ora alcune tra le maggiormente sfruttate.

Abbiamo già citato il bancomat, la carta di credito e le serrature degli alberghi. Ma siete mai entrati in club privati, in palestre moderne, in circoli ricreativi moderni?

La maggior parte di questi ha un controllo degli accessi e delle presenze implementato con badge magnetici.

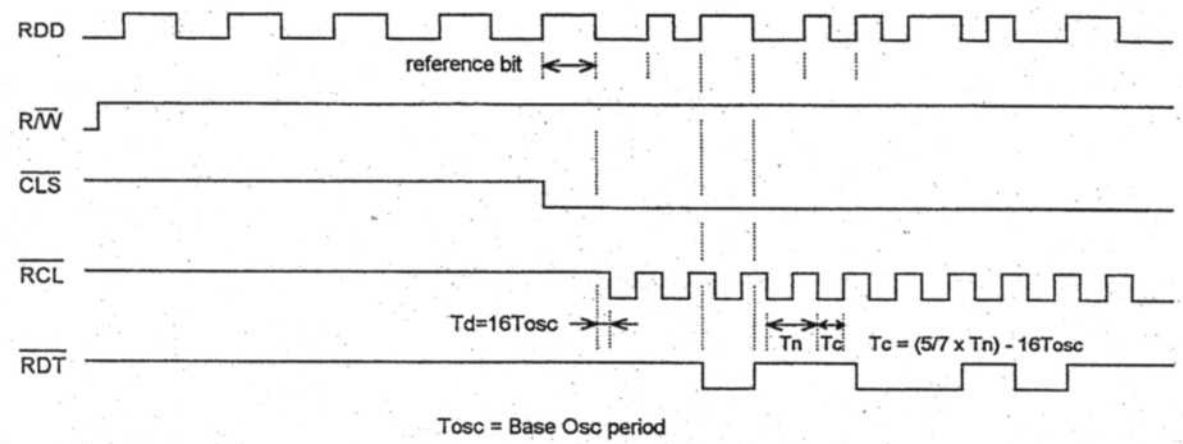
Supponiamo ad esempio di dover gestire un impianto sportivo con campi da tennis, piscina per bambini, piscina per adulti, sauna, palestra, bar, salottini ecc. Con le carte magnetiche l'ingresso a ciascuna zona viene controllato perfettamente.

Tabella 1. Specifiche ISO 7811

I (IATA)	II (ABA)	III (MINTS)
F2F (FM)		
210 BPI	75 BPI	210 BPI
79 Characters (7-bit code)	40 Characters (5-bit code)	107 Characters (5-bit code)
0.76 ±0.08 mm		

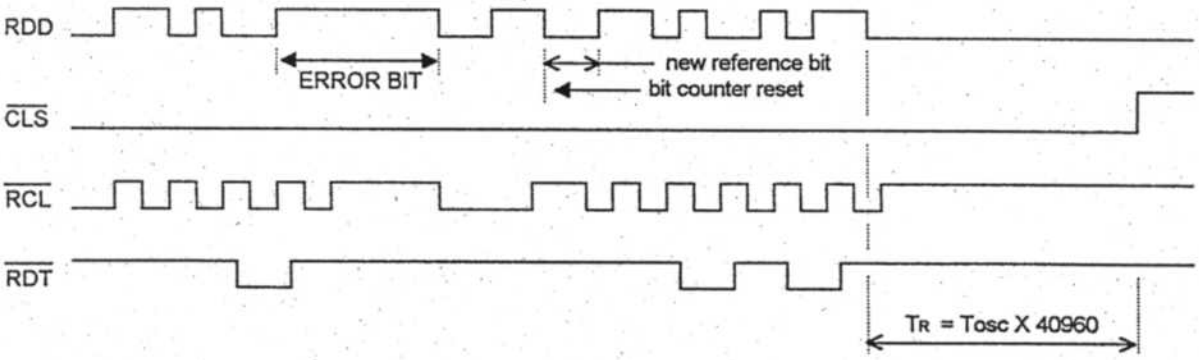
Tabella 2. Set caratteri traccia 1

				b6	0	0	1	1
				b5	0	1	0	1
b4	b3	b2	b1	ROW / COL	0	1	2	3
0	0	0	0	0	SP	0	a	P
0	0	0	1	1	a	1	A	Q
0	0	1	0	2	a	2	B	R
0	0	1	1	3	c	3	C	S
0	1	0	0	4	$	4	D	T
0	1	0	1	5	d	5	E	U
0	1	1	0	6	a	6	F	V
0	1	1	1	7	a	7	G	W
1	0	0	0	8	(	8	H	X
1	0	0	1	9	)	9	I	Y
1	0	1	0	10	a	a	J	Z
1	0	1	1	11	a	a	K	b
1	1	0	0	12	a	a	L	b
1	1	0	1	13	–	a	M	b
1	1	1	0	14	·	a	N	d
1	1	1	1	15	/	d	O	a



Tosc = Base Osc period

BPI	75 BPI		210 BPI	
SPEED	10 Cm/sec	120 Cm/sec	10 Cm/sec	120 Cm/sec
Tn	≒ 3.38 mS	≒ 282 µS	≒ 1.2mS	≒ 100 µS
Tc	≒ 2.4 mS	≒193.4 µS	≒849 µS	≒63.4 µS



CLS generation (SELECT input voltage is low)

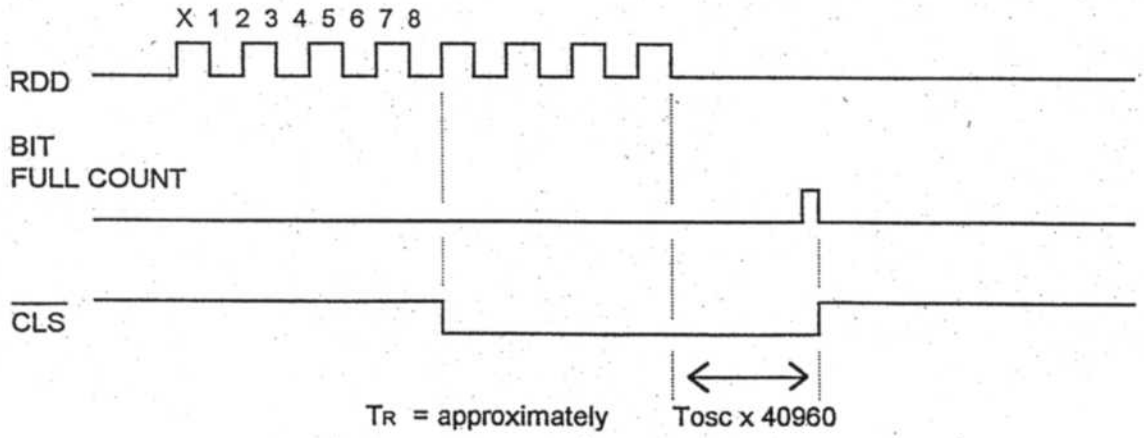
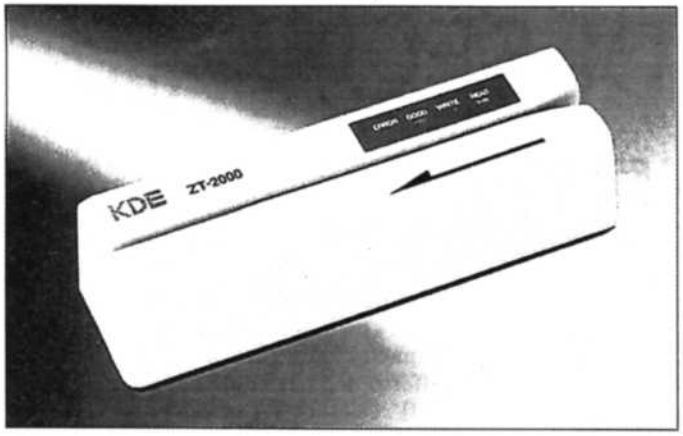
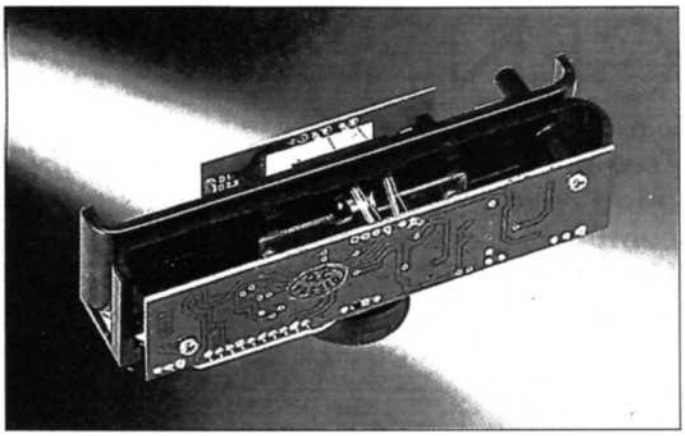


Figura 2. Timing diagram dei lettori ISO7811



Lettori manuali di carte magnetiche



Lettori-scrittori manuali di carte magnetiche

In primo luogo, la tessera viene programmata all'atto dell'iscrizione con i dati anagrafici della persona, poi viene inserita la data di scadenza dell'abbonamento e infine vengono memorizzati i servizi cui accedere: infatti, l'iscrizione può prevedere l'accesso alla piscina per adulti, ma non a quella per bambini, oppure può venire esclusa la sauna oppure i campi da tennis.

Per quest'ultimi poi, la gestione delle prenotazioni e dei pagamenti può avvenire semplicemente sempre con la tessera magnetica, velocizzando l'operazione e permettendo così la riduzione del personale addetto.

Un altro esempio di utilizzo di carte magnetiche si ha nel controllo degli accessi e delle presenze nelle ditte e, da poco, anche negli ambienti pubblici.

Da qualche tempo, inoltre, si trovano sempre più frequentemente macchinette che distribuiscono videocassette a noleggio con tessere magnetiche prepagate. Lo stesso dicasi della Telecom e di tutte le imprese che offrono servizi prepagati, come per esempio parcheggi a pagamento, noleggio di auto, impiego di fotocopiatrici e di distributori di bevande.

Le applicazioni che proporremo noi, invece, sono più ristrette, nel senso che offriremo la possibilità di realizzare chiavi elettroniche con autoapprendimento a una sola uscita e interfacciamento con la seriale RS-232 del computer. Il lato più interessante della nostra proposta è che abbiamo utilizzato la traccia 2 della carta, la più sfruttata, e che quindi permetterà di adoperare come carte magnetiche, anche le carte di credito ed i codici fiscali, senza necessità di dover acquistare altre carte.

In ogni caso, sarà possibile avere anche carte programmate ed anche personalizzate con i vostri contrassegni contattando il numero 0337/259730.

Tabella 3. Set caratteri traccia 2 e 3

P	b4	b3	b2	b1	ROW	CHAR
1	0	0	0	0	0	0
0	0	0	0	1	1	1
0	0	0	1	0	2	2
1	0	0	1	1	3	3
0	0	1	0	0	4	4
1	0	1	0	1	5	5
1	0	1	1	0	6	6
0	0	1	1	1	7	7
0	1	0	0	0	8	8
1	1	1	0	1	9	9
1	1	0	1	0	10	a
0	1	0	1	1	11	SS
1	1	1	0	0	12	a
0	1	1	0	1	13	S
0	1	1	1	0	14	a
1	1	1	1	1	15	ES

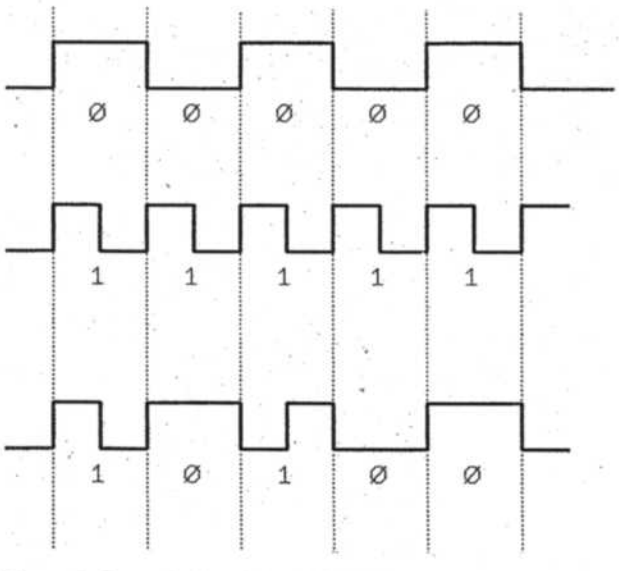


Figura 3. Esempio di codifica ISO7811

Si ringrazia la Ditta EDUE Italia Spa, Via Cassiani, 155 (Modena) per la collaborazione e la documentazione fornitaci per la realizzazione di questo articolo l'azienda fornisce i lettori/scrittori descritti in queste pagine telefonando al numero 059/313403 o via fax al numero 059/314356.

continua

I LETTORI DI BADGE

Continuiamo il nostro viaggio alla scoperta dei lettori di badge, i cui costi ormai sono scesi alla portata di tutti, consentendo da un lato una maggiore sperimentazione e dall'altro la possibilità di utilizzarli nelle più svariate applicazioni

Paola Sbrana - 2ª parte

Dopo aver analizzato in dettaglio la struttura delle carte magnetiche e le loro codifiche, ci cimenteremo nella realizzazione di un circuito dalle mille applicazioni.

L'approccio più semplice con cui iniziare a lavorare con i badge è senza dubbio quello della lettura della traccia 2. Spiegheremo più avanti quanto magnetizzare una tessera magnetica non sia poi così semplice per chi non possiede l'apparecchio adatto e che noi abbiamo potuto testare e collaudare grazie alla EDUE Italia. Detto questo, passiamo alla descrizione del circuito proposto in queste pagine e che implementa una vera e propria chiave elettronica.

Schemi di chiavi elettroniche prima ne abbiamo visti molti nella storia di questa rivista, partendo da quella resistiva, per poi passare a quella a tastiera dapprima senza e poi con microprocessore, successivamente quella con EEPROM seriali ed infine quella con le touch-memory della DALLAS, ma chiavi con carte magnetiche non ne avevamo ancora trattate.

Il motivo per cui abbiamo aspettato del tempo prima di proporre un circuito simile è che, in un recente passato, le applicazioni dei badge erano confinate alle sole operazioni bancarie, e così non abbiamo volute divulgare informazioni che potevano "indurre in tentazione".

Poiché, come abbiamo visto lo scorso mese, da qualche tempo queste carte vengono impiegate con successo anche in applicazioni di tutt'altro genere, abbiamo deciso di farle conoscere e di farne apprezzare i loro vantaggi.

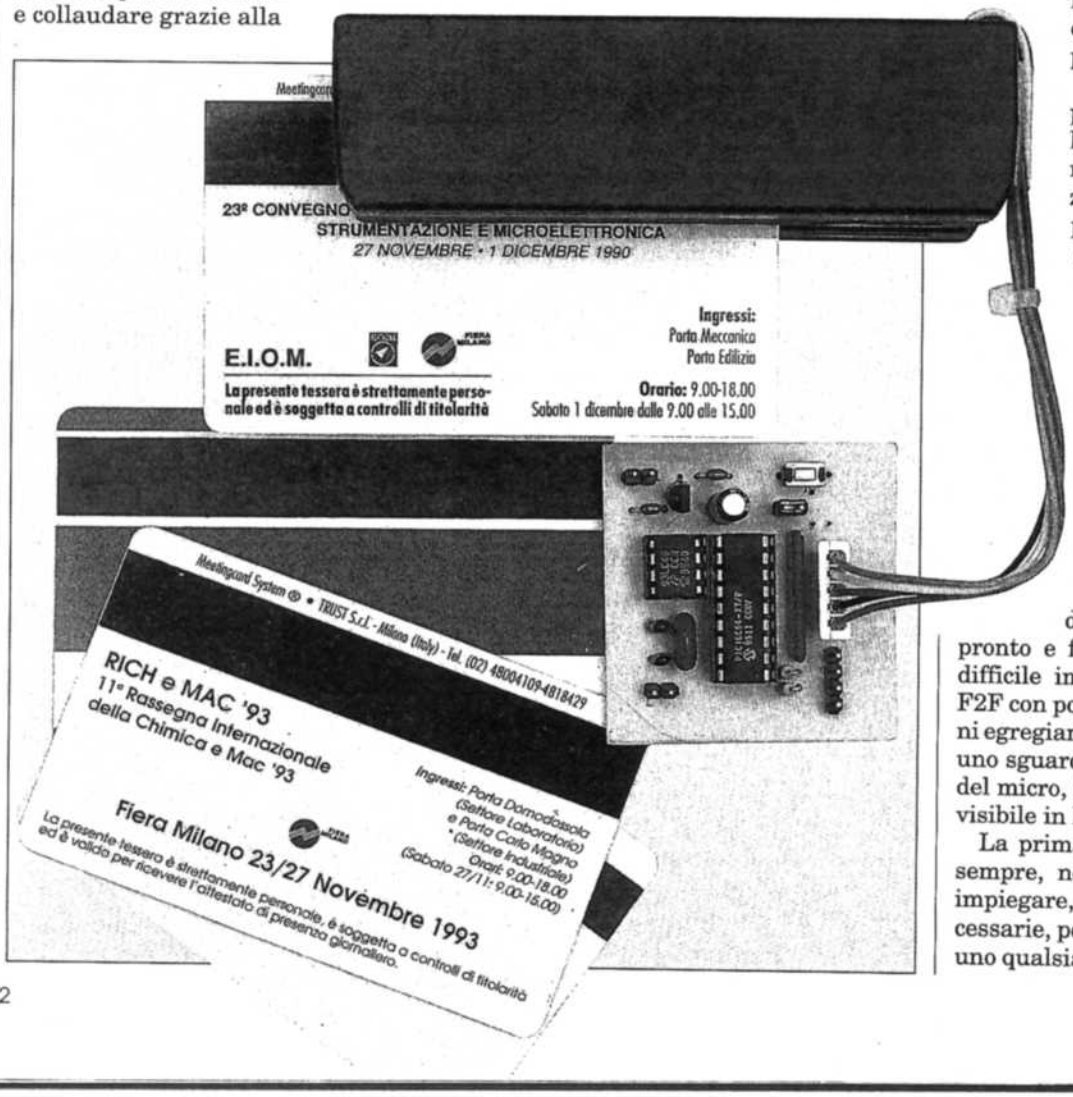
La nostra proposta

Il circuito che andremo tra poco ad analizzare, svolge le mansioni di una vera e propria chiave elettronica con autoapprendimento, ovvero è possibile memorizzare fino a 10 carte diverse senza accorgimenti particolari.

Sono previste delle uscite per la visibilità dello stato della lettura (corretta, non corretta) e una uscita per l'attivazione di un carico. In Figura 4 possiamo vedere lo schema elettrico: il circuito integrato IC1, un PIC16C54XT programmato allo scopo, controlla il funzionamento di tutta la chiave. Il chip IC2 serve per memorizzare i codici relativi alle dieci carte da acquisire, mentre il regolatore IC3 fa in modo di stabilizzare la tensione di alimentazione per tutto il circuito intorno ai 5 volt.

Il lettore di badge invece deve essere acquistato già pronto e funzionante, poiché sarebbe difficile implementare una decodifica F2F con pochi componenti e che funzioni egregiamente. Andiamo allora a dare uno sguardo al programma di gestione del micro, il cui diagramma a blocchi è visibile in Figura 5.

La prima operazione consiste, come sempre, nell'inizializzare le porte da impiegare, i registri e le periferiche necessarie, poi il chip si mette in attesa di uno qualsiasi dei due eventi desiderati:



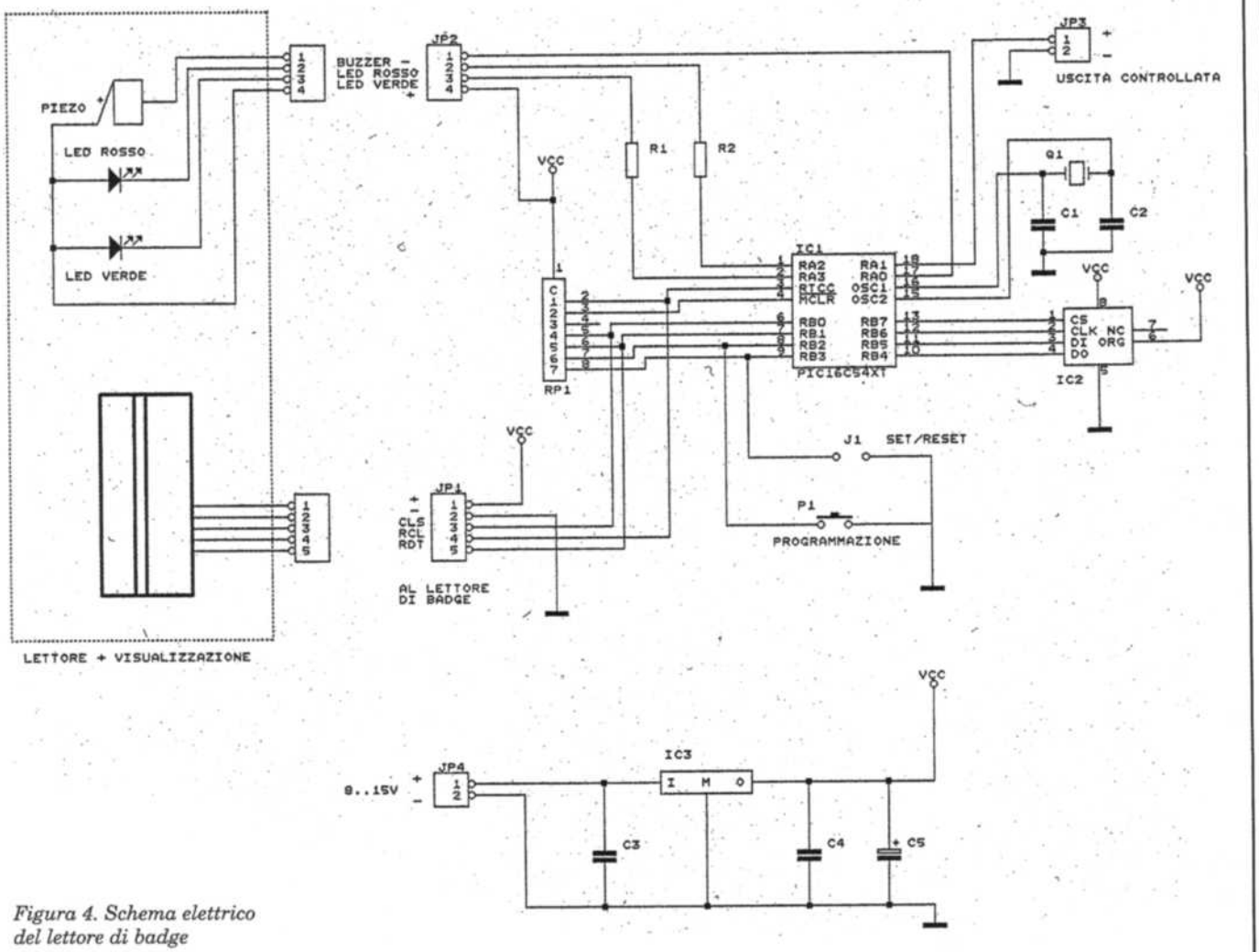


Figura 4. Schema elettrico del lettore di badge

la pressione del pulsante P1, che significa volontà di acquisizione codici, oppure l'arrivo di segnali da parte del lettore di badge, che significa che una carta è stata inserita nel lettore stesso.

Supponiamo che sia stato premuto il pulsante di programmazione P1. Allora entrambi i Led si accendono ed il buzzer emette un suono per circa 3 secondi.

A questo punto si attende l'inserimento di una carta. Quando questa arriva, il Led rosso si spegne per mezzo secondo e contemporaneamente si ode un beep, segno che la carta è stata accettata. Viene così memorizzata in IC2 e poi si passa a vedere se tale carta è la decima inserita.

Se ciò si verifica, entrambi i Led si spengono ed il controllo torna allo stato iniziale, viceversa si prosegue fino alla memorizzazione di tutte e 10 le carte.

Precedentemente abbiamo accennato al fatto che una carta possa non essere accettata: questo fatto può avvenire per diversi motivi. Il primo è che tale carta non rispetti lo standard ISO7811 (che poi vedremo nel dettaglio per la traccia 2). Il secondo è che il badge sia stato inserito o troppo lentamente o troppo velocemente.

In entrambi i casi, si passa ad una routine denominata ERRORE, che spegne il Led verde per un secondo e, contemporaneamente, fa eseguire tre beep veloci al buzzer.

È previsto che le carte da memorizzare possano essere duplicate, nel senso che si possono memorizzare dieci codici identici corrispondenti ad una sola carta. Tale accorgimento risulta utile nel caso in cui si possieda un ristretto numero di tessere magnetiche.

Vediamo ora il ramo del diagramma relativo alla lettura di una carta. Quando la lettura è terminata, IC1 interroga la memoria e cerca di trovare il codice letto tra quelli memorizzati. Se ciò non avviene, il Led rosso si accende per un secondo ed il buzzer entra nella routine di ERRORE vista precedentemente.

Quando ne esce, si torna al test iniziale. Se invece la carta inserita ha il codice uguale ad uno precedentemente memorizzato, il buzzer fa un breve beep, il Led verde si accende per un secondo e l'uscita relativa al carico viene modificata secondo la posizione del jumper J1: se il jumper non è inserito, l'uscita si attiva per circa un secondo, poi torna allo stato inattivo. Viceversa, se il jumper risulta inserito, l'uscita viene "togglata", ovvero il suo stato viene complementato.

In pratica, se l'uscita era attiva viene resa inattiva e viceversa.

Tornando allo schema di Figura 4, notiamo che sia la sezione di visualizzazione, sia quella relativa all'uscita, sono esterne al circuito principale, per dare la possibilità di gestirle nel modo voluto.

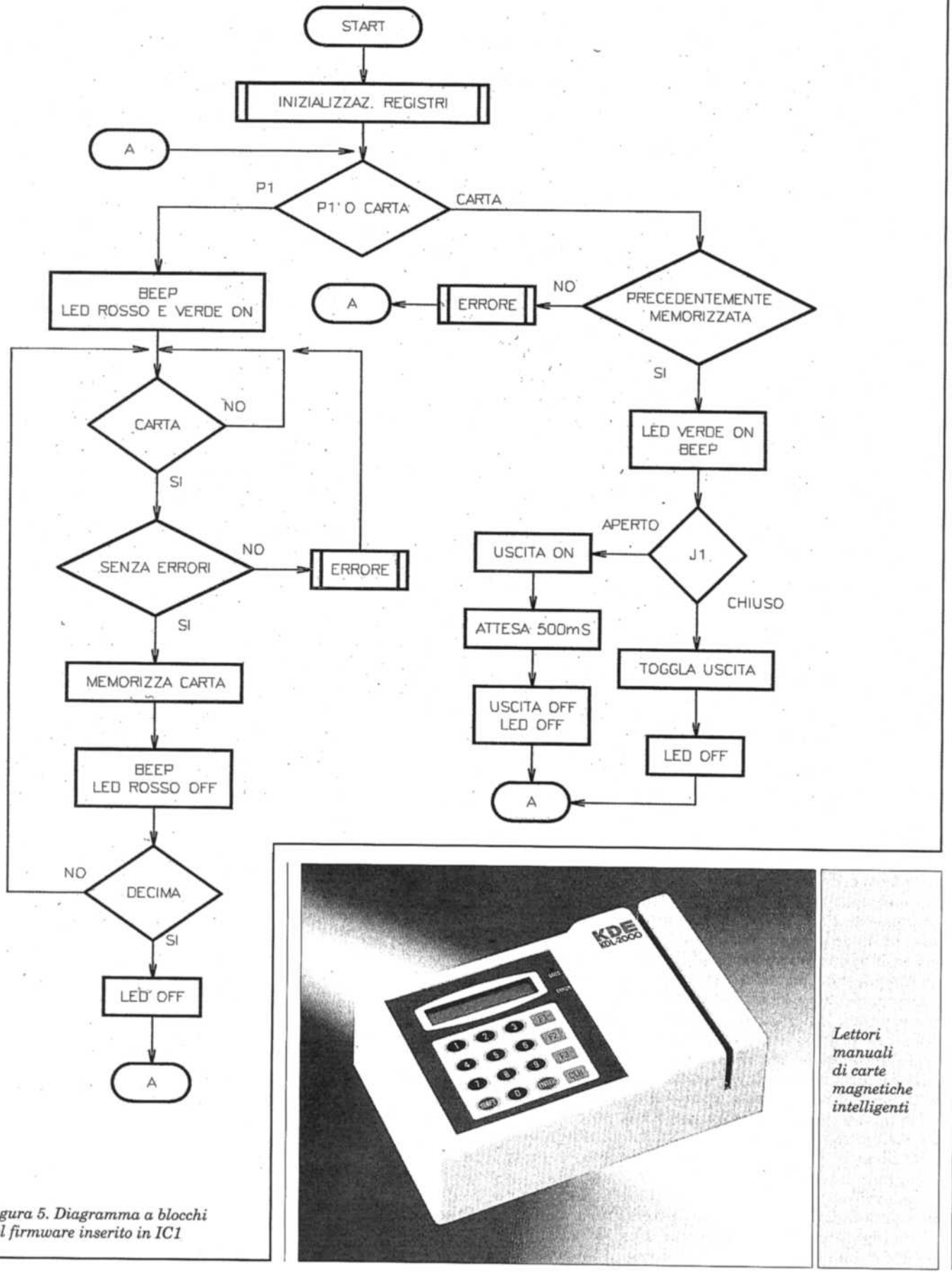


Figura 5. Diagramma a blocchi del firmware inserito in IC1

Lettori manuali di carte magnetiche intelligenti

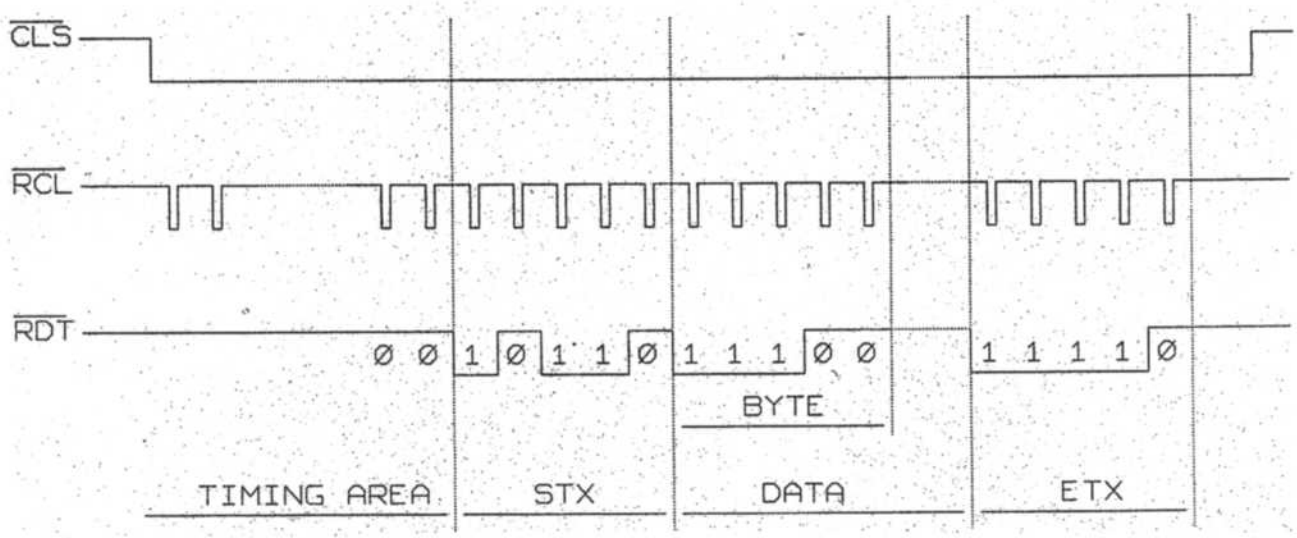


Figura 6. Diagramma dei segnali in uscita dal lettore KDA1121

La codifica della traccia 2

Come promesso, andiamo a vedere come giungono le informazioni dal lettore della EDUE Italia, aiutandoci con il diagramma visibile in Figura 6.

Il segnale più in alto è il CLS, che normalmente sta ad alto livello (come tutti gli altri) e che rimane a livello basso per tutta la durata dello strisciamento della carta.

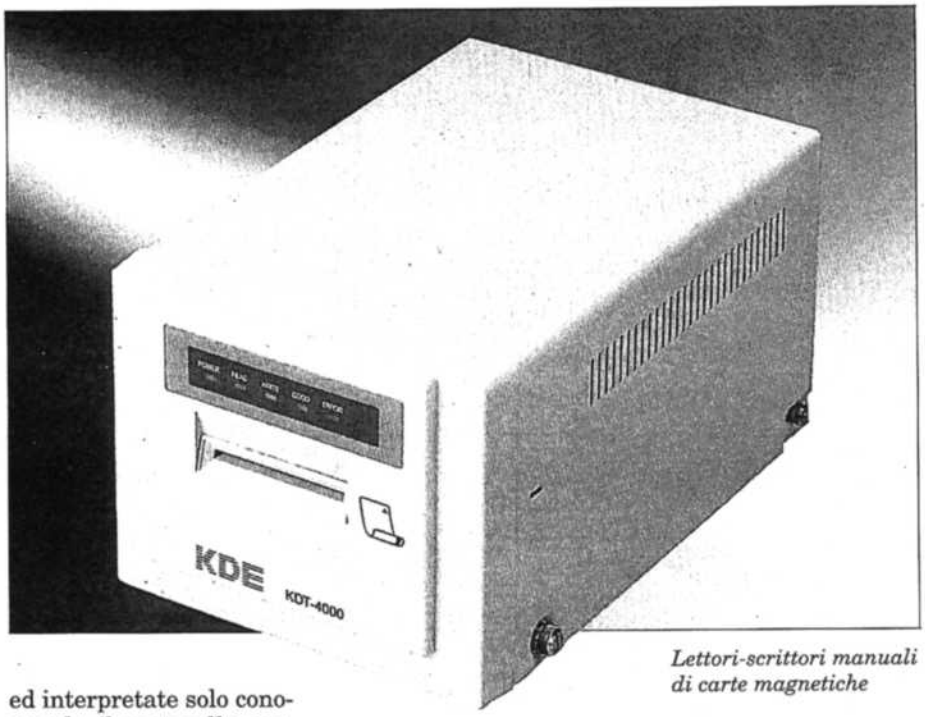
Se, durante tale strisciamento, la lettura risulta falsata, questo segnale torna ad alto livello, permettendo un check in tempo reale della corretta lettura.

Gli altri due segnali utili sono quello dei dati (RDT) e quello del clock (RCL). L'RCL permette la sincronizzazione della lettura su ogni suo fronte di discesa, mentre l'RDT vale 0 quando il livello è alto ed 1 quando il livello è basso.

Per uniformarci allo standard ISO7811, le carte da noi impiegate dovranno essere così memorizzate: una timing area, in cui i dati valgono sempre 0, una STX (detta anche Start Sentinel) che è composta da 4 bit di dato ed uno di parità e che deve valere bh.

Poi ci sono gli eventuali altri 38 caratteri, sempre da 4 bit più uno per la parità ed infine abbiamo la ETX (ovvero la End Transmission) che vale fh più il bit di parità.

Questo è quanto richiesto dallo standard ISO7811, ma ciò non vuol dire che debba necessariamente essere seguito, in quanto è possibile programmare tessere che non rispettano questo standard e che quindi possono poi essere rilette ed interpretate solo conoscendo il protocollo prescelto. Il circuito da noi proposto rispetta ovviamente lo standard IS=7811, anche perché per poter usufruire delle carte già in vostro possesso, era la soluzione migliore.



Lettori-scrittori manuali di carte magnetiche

Il montaggio

Passiamo adesso al montaggio del circuito, sfruttando la traccia per lo stampato proposta in Figura 7. Montiamo i vari componenti seguendo le indicazioni riportate in Figura 8. Non ci sono particolari attenzioni da osservare, eccetto quella di impiegare degli zoccoli per i circuiti integrati.

Le connessioni con il mondo esterno sono semplici e veloci da realizzare.

Per l'alimentazione, consigliamo una tensione compresa tra 8 e 15 volt. Apriamo qui una parentesi per consigliare di non impiegare MAI i cosiddetti alimentatori "a parete", perché la tensione che ne esce raramente è filtrata, raddrizzata bene e stabilizzata.

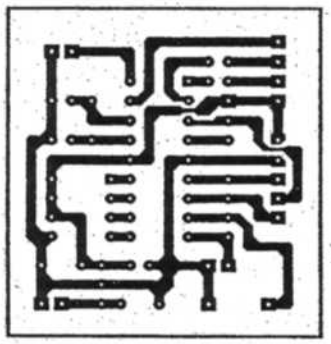


Figura 7. Circuito stampato scala 1:1

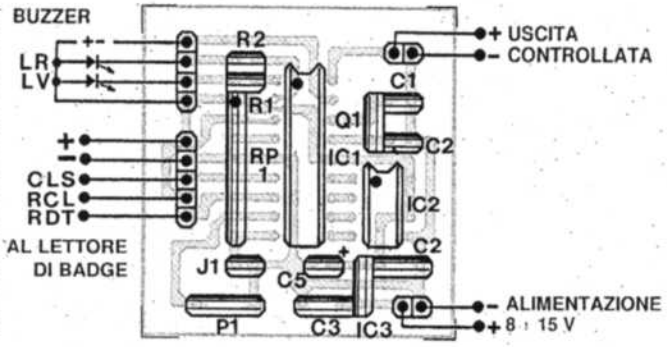


Figura 8. Disposizione dei componenti e collegamenti

ELENCO COMPONENTI

Semiconduttori
IC1: PIC16C54XT programmato
IC2: 93C46
IC3: 78L05

Resistori
R1, R2: 220 Ω
RP1: Rete 10 kΩ

Condensatori
C1, C2: 10 pF
C3, C4: 100 nF
C5: 10 µF 16 V

Varie
Q1: Oscillatore 3,58 MHz
Lettore: KDA1121 (EDUE Italia)

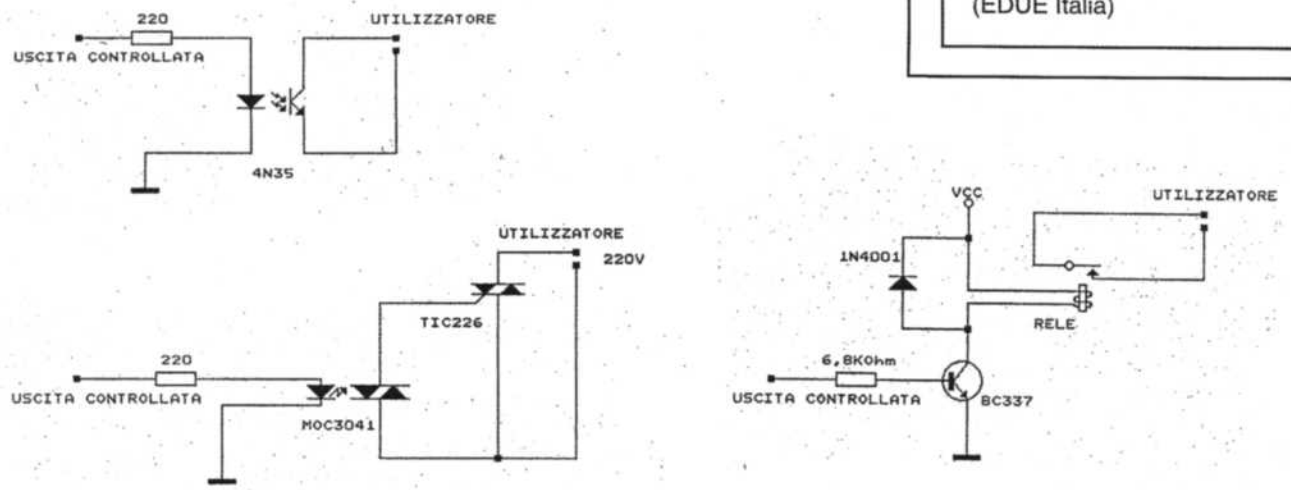


Figura 9. Esempi di connessione a carichi

Spesso infatti ci giungono circuiti con i componenti bruciati per questo motivo.

In Figura 9 abbiamo a disposizione alcuni esempi di come applicare carichi sull'uscita. Vediamo invece il collaudo del lettore.

Prima di connettere alimentazione, assicuratevi che il connettore, non essendo polarizzato, sia inserito correttamente.

Per questo motivo riportiamo i colori corrispondenti ai vari segnali:

ROSSO: +
NERO: –
MARRONE: CLS
GIALLO: RCL
ARANCIO: RDT

Diamo allora alimentazione, avendo ad esempio collegato sull'uscita un relè.

La prima cosa che dobbiamo notare, a conferma che tutto funziona regolarmente, è l'accensione del Led rosso per un secondo e tre beep brevi dal buzzer.

Premiamo allora il pulsante P1 per iniziare a memorizzare le carte in nostro possesso.

A proposito di queste, noi abbiamo provato con le più comuni, e cioè la carta di credito ed il codice fiscale, che ormai tutti possiedono.

Potete provare anche con altre carte, ma otterrete un corretto funzionamento solo se conformi allo standard ISO7811.

Memorizzate quindi le dieci carte seguendo le indicazioni descritte precedentemente (può essere anche la stessa carta per dieci volte), poi notate che entrambi i Led si spengono.

A questo punto potrete utilizzare la chiave, settando opportunamente il jumper J1: chiuso per un funzionamento set-reset, aperto per uno di tipo impulsivo.

In qualsiasi momento desideriate, potete riprogrammare le carte memorizzate, inserendo nuovamente i dieci codici, utilità da apprezzare nel caso di perdita di una tessera.

Le applicazioni di un tale circuito sono infinite: serrature per porte e portoni, chiavi per antifurti, controllo di accessi a determinate strutture, sblocco antifurto in auto, abilitazione a determinate operazioni sul computer o su macchine utensili particolari, ecc. Telefonando allo 0337/259730 è possibile avere modifiche al circuito originale in funzione delle specifiche esigenze, ovviamente per un minimo quantitativo.

Si ringrazia la Ditta EDUE Italia Spa, Via Cassiani, 155 (Modena) per la collaborazione e la documentazione fornitaci per la realizzazione di questo articolo.

Presso la EDUE potrete trovare i lettori/scrittori descritti in queste pagine telefonando al numero 059/313403 o via fax al numero 059/314356.

continua

I LETTORI DI BADGE

Concludiamo la nostra panoramica sui lettori di badge presentando un circuito pratico in grado di realizzare una chiave elettronica, sfruttando una ormai comune, e ai lettori di Progetto nota, tessera magnetica

Paolo Sbrana - 3ª parte

Parlando ancora di carte magnetiche, in queste pagine vedremo una variante alla soluzione proposta il mese scorso: un circuito che non svolge più funzione di chiave elettronica, ma è indispensabile per il controllo degli accessi e delle presenze laddove ci sia un computer che controlla il tutto, infatti abbiamo sviluppato un lettore di traccia 2 (solo per i primi 32 caratteri) che trasmette, dopo la lettura, tutti i dati ad un computer via interfaccia seriale RS-232, che ben conosciamo.

Abbiamo scelto di trasferire solamente 32 caratteri invece di 40 per il semplice motivo che il PIC impiegato è il più piccolo della famiglia e quindi non ha Ram a sufficienza per tutti e 40 i caratteri supportati dalla traccia 2 delle tessere.

Nonostante ciò, il numero di tessere diverse tra loro è uguale a 2 elevato alla 128, come dire miliardi di miliardi di miliardi (una cifra con oltre 36 zeri!), quindi la sicurezza di non avere duplicati è elevatissima.

Se poi le tessere vengono magnetizzate in successione, il problema non sussiste sicuramente.

A differenza del primo circuito, con questo non sarà possibile, a meno di un altro circuito ausiliario, attivare o meno relè o altri carichi, ma sarà invece consentito di implementare uno o più programmi, su personal computer, che gestiscano il controllo degli accessi e delle presenze in uffici, locali particolari, club, aziende di grandi dimensioni.

Tanto per fare un esempio, il Gruppo Editoriale JCE ne ha installato uno proprio all'ingresso.

Ad esempio, è possibile crearsi delle tabelle di corrispondenza e tenere sotto controllo tutti gli ingressi e le uscite degli impiegati, memorizzando data e ora, e, sempre in modo automatico, ottenere a fine mese un grafico delle presenze effettive di una determinata persona in un determinato locale o ufficio.

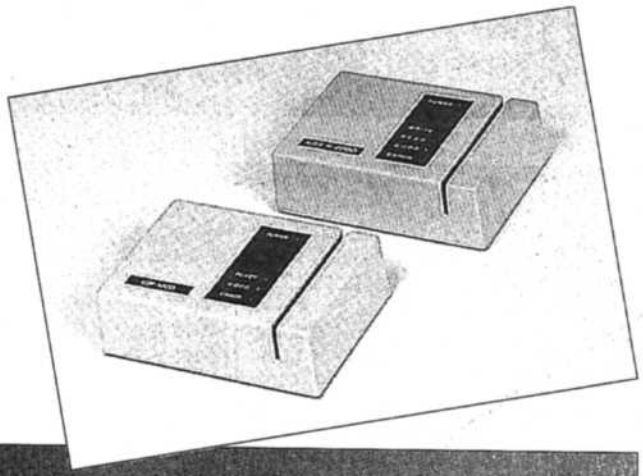
Si ricorda, a scopo informativo, che da poco è stata istituita una legge che prevede che gli uffici pubblici adottino un sistema di controllo presenze proprio di questo tipo.

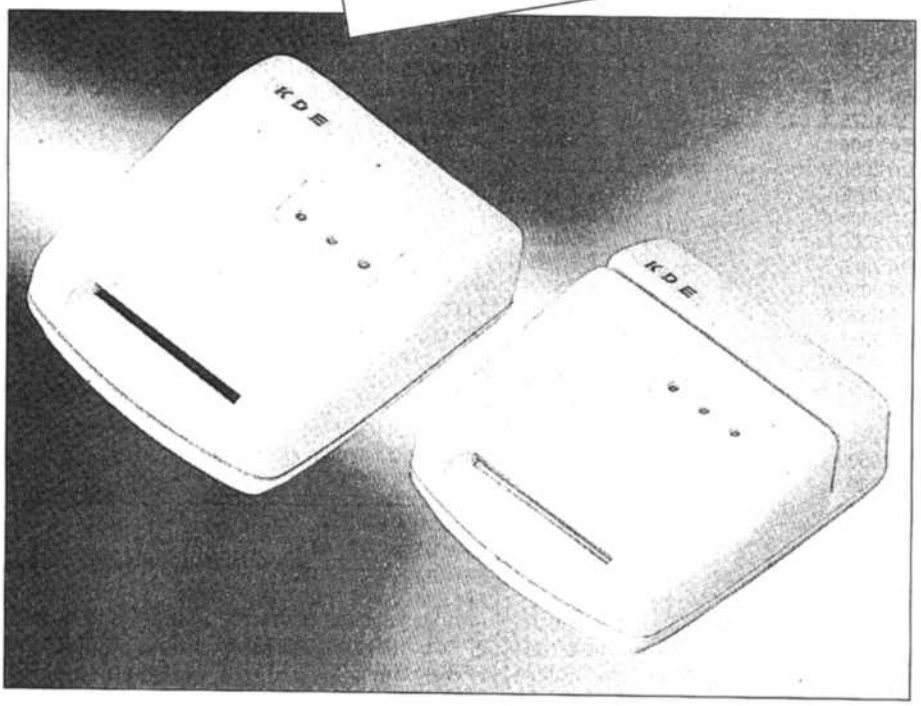
Naturalmente, la gestione del software varierà da ambiente ad ambiente, e per questo abbiamo preferito non implementarla, lasciando ad ognuno la scelta di realizzarselo in proprio con il linguaggio che preferisce.

Il nostro consiglio è di utilizzare il Visual Basic, perché molto facile da gestire e con le potenzialità dell'interfaccia utente superiori a qualsiasi altro linguaggio di programmazione, sebbene anche con Access (che però è una linguaggio per la gestione di database) sia possibile provare.

Ma veniamo alle caratteristiche principali del nostro circuito: alimentazione da 8 a 15 volt, basso consumo, avviso con buzzer e con Led del corretto funzionamento, comunicazione seriale in standard RS-232 a 1.200 8N1.

Qualcuno potrebbe farci notare che ormai sulla seriale tutti i software utilizzano la velocità minima di 9.600, ma in applicazioni "lente" come questa non ne vediamo il motivo, a fronte anche di una maggiore sicurezza del dato trasferito.





```
/*****************************************************/
/* BADGE1        Andrea Sbrana        06/96          */
/*****************************************************/
/* Ricevi i dati via RS232 e visualizzali su schermo */
/*****************************************************/
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <serial.h>
#include <defines.h>
#include <bios.h>
int                rs232;
extern int SError = 0;

int main(int argc, char *argv[])
{
        int         end_cycle = FALSE;
        BYTE        ch;
        static      BYTE      ch1,ch2;
        UINT    value;

        if(argc != 2)
                end_cycle=TRUE;
        if(!strcmp(argv[1],"1"))
                rs232 = 1;
        else
                if(!strcmp(argv[1],"2"))
                        rs232 = 2;
                else
                        end_cycle=TRUE;
        if(!OpenCom(rs232, 1200, NO_PARITY, 8, 1))
                end_cycle=TRUE;
        if(end_cycle==TRUE)
       {printf("\nSintassi non corretta");
        printf("\nSintassi: BADGE1 X");
        printf("\ncon X = numero della porta seriale (1 o 2)");
        CloseCom(rs232);
        return 0;}
        do
        {       if(kbhit())
                {   value=(BYTE)(_bios_keybrd(_KEYBRD_READ));
                    if(value==ESC)
                            end_cycle=TRUE;
                    else
                    {       if(value==ENTER)
                                    printf("\n");}
                }
                if(DataReady())
                {       ReadChar(&ch1);
                        printf("%02X",ch1);}}
        while(!end_cycle);
        printf("\nThanks for using RS232 receiver\n");
        CloseCom(rs232);
        return 0;
}


/*****************************************************/
/* INTER232      Andrea Sbrana          12/94        */
/*****************************************************/

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <bios.h>
#include <dos.h>
#include <conio.h>
#include <time.h>
#include <defines.h>
#include <serial.h>

#pragma check_stack(off)
#define BUFOVFL 1
#define SBUFSIZ 4096                    /* Serial buffer size */
#define PORT_ERROR 1

extern int SError;
int portbase = 0;
void (interrupt far *oldcom_int)();

static char SerialBuff[SBUFSIZ];
unsigned int startbuf = 0;
unsigned int endbuf = 0;

/*****************************************************/
/* COM_INT                                           */
/*****************************************************/
void interrupt far com_int(void)
{
        _disable();
        if((inp(portbase + IIR) & RX_MASK) == RX_ID)
        {
                SerialBuff[endbuf++] = (char)inp(portbase + RXR);
                endbuf %= SBUFSIZ;
                if (endbuf == startbuf)
                        SError = BUFOVFL;
        }
        outp(ICR,EOI);      /* segnalazione fine interrupt */
        _enable();
}

/*****************************************************/
/*  SendChar                                         */
/*****************************************************/
int SendChar(int c)
{
        clock_t time1, time2;

        time1 = clock();

        while((inp(portbase + LSR) & XMTRDY) == 0)
{
time2 = clock();
if((float)time2 - (float)time1 > (float)TIMEOUT) return(FALSE);
}
        _disable();
        outp(portbase + TXR, c);
        _enable();
        return(TRUE);
}

/*****************************************************/
/*  AbleToTrasmit                                    */
/*****************************************************/
int AbleToTrasmiti(void)
{
        clock_t time1, time2;

        outp(portbase + MCR, MC_INT | DTR | RTS);
        time1 = clock();
      while((inp(portbase + MSR) & CTS) == 0) /* CTS a 0 */
{
time2 = clock();
if((float)time2 - (float)time1 > (float)TIMEOUT) return(FALSE);
}
        return(TRUE);
}

/*****************************************************/
/*  Utility                                          */
/*****************************************************/
void StartDataIn(void)
{
        outp(portbase + MCR, MC_INT | DTR | RTS);
}

void SopDataIn(void)
{
        outp(portbase + MCR, MC_INT | DTR | (~RTS));
}

int DataReady(void)
{
        if(endbuf != startbuf) return(TRUE); else return(FALSE);
}

/*****************************************************/
/*  ReadChar                                         */
/*****************************************************/
int ReadChar(int *c)
{
        clock_t time1, time2;

        time1 = clock();
        while((endbuf == startbuf))
{
time2 = clock();
if((float)time2 - (float)time1 > (float)TIMEOUT) return(FALSE);
}
        *c = SerialBuff[startbuf];
        startbuf++;
        startbuf %= SBUFSIZ;
        return(TRUE);
}

/*****************************************************/
/* i_enable & i_disable                              */
/*****************************************************/
void i_enable(int pnum)
```

```
{
        int         c;

        _disable();
        c = inp(portbase + MCR) | MC_INT;
        outp(portbase + MCR, c);
        outp(portbase + IER, RX_INT);
        c = inp(IMR) & (pnum == COM1 ? IRQ4 : IRQ3);
        outp(IMR,c);
        _enable();
}

void i_disable(int pnum)
{
        int         c;

        _disable();
        c = inp(IMR) | (pnum == COM1 ? ~IRQ4 : ~IRQ3);
        outp(IMR,c);
        outp(portbase + IER, 0);
        c = inp(portbase + MCR) & ~MC_INT;
        outp(portbase + MCR, c);
        _enable();
}

/*****************************************************/
/* SetPort & SetSpeed & SetOthers                    */
/*****************************************************/
int SetPort(Port)
{
        int intNumber;

        switch(Port)
        {
                case COM1:
                        intNumber = 0x0C;
                        portbase = COM1BASE;
                        break;
                case COM2:
                        intNumber = 0x0B;
                        portbase = COM2BASE;
                        break;
                default:
                        return(FALSE);
        }
        oldcom_int = _dos_getvect(intNumber);
        _dos_setvect(intNumber, com_int);
        i_enable(Port);
        return(TRUE);
}

int SetSpeed(int Speed)
{
        char c;
        int divisor;

        if(Speed == 0)
                return(FALSE);
        else
                divisor = (int)(115200L/Speed);
        if(portbase == 0)
                return(FALSE);
        _disable();
        c = (char)inp(portbase + LCR);
        outp(portbase + LCR, (c | 0x80));
        outp(portbase + DLL, (divisor & 0x00FF));
        outp(portbase + DLH, ((divisor >> 8) & 0x00FF));
        outp(portbase + LCR, c);
        _enable();
        return(TRUE);
}

int SetOthers(int Parity, int Bits, int StopBit)
{
        int         setting;

        if(portbase == 0)
                return(FALSE);
        if(Bits < 5 || Bits > 8)
                return(FALSE);
        if(StopBit != 1 && StopBit != 2)
                return(FALSE);
        if(Parity != NO_PARITY && Parity != ODD_PARITY &&
                Parity != EVEN_PARITY)
                return(FALSE);
        setting = Bits - 5;
        setting |= ((StopBit == 1) ? 0x00 : 0x04);
        setting |= Parity;
        _disable();
        outp(portbase + LCR, setting);
        _enable();
        return(TRUE);
}

/*****************************************************/
/* CloseCom & OpenCom                                */
/*****************************************************/

int CloseCom(int Port)
{
        int intNumber;

        switch(Port)
        {
                case COM1:
                        intNumber = 0x0C;
                        break;
                case COM2:
                        intNumber = 0x0B;
                        break;
                default:
                        return(FALSE);
        }
        _dos_setvect(intNumber, oldcom_int);
        i_disable(Port);
        return(TRUE);
}

int OpenCom(int Port,int Speed,int Parity,int Bits,int StopBit)
{
        startbuf = endbuf = 0;
        if(SetPort(Port) && SetSpeed(Speed) &&
                SetOthers(Parity, Bits, StopBit))
                        return(TRUE);
        CloseCom(Port);
        return(FALSE);
}

/*****************************************************/
/* fTX                                               */
/*****************************************************/
int     fTX(BYTE data_byte)
{
        int         err_tx;     /* error for tx function */

        err_tx = SendChar((char)data_byte);
        if(err_tx == FALSE)
                return(PORT_ERROR);
        return(OK);                     /* end programming byte */
}

/*****************************************************/
/* fRX                                               */
/*****************************************************/
int fRX()
{
        int         err_rx;                 /* error for rx function */
        int     ricex;
        static BYTE         ch;
        DWORD   count = 0;
        DWORD   start,finish;

        start = clock();
        do
        {
                if(DataReady())
                {
                        ReadChar(&ricex);
                        if(!SError)
                        {
                                ch=(BYTE)ricex;
                                return((int)ch);
                        }
                        else
                        {
                                count = clock();
                                finish = (count - start) / CLK_TCK;
                        }
                }
                else
                {
                        count = clock();
                        finish = (count - start) / CLK_TCK;
                }
        }
        while(finish < 1);
        return(PORT_ERROR);
}
```

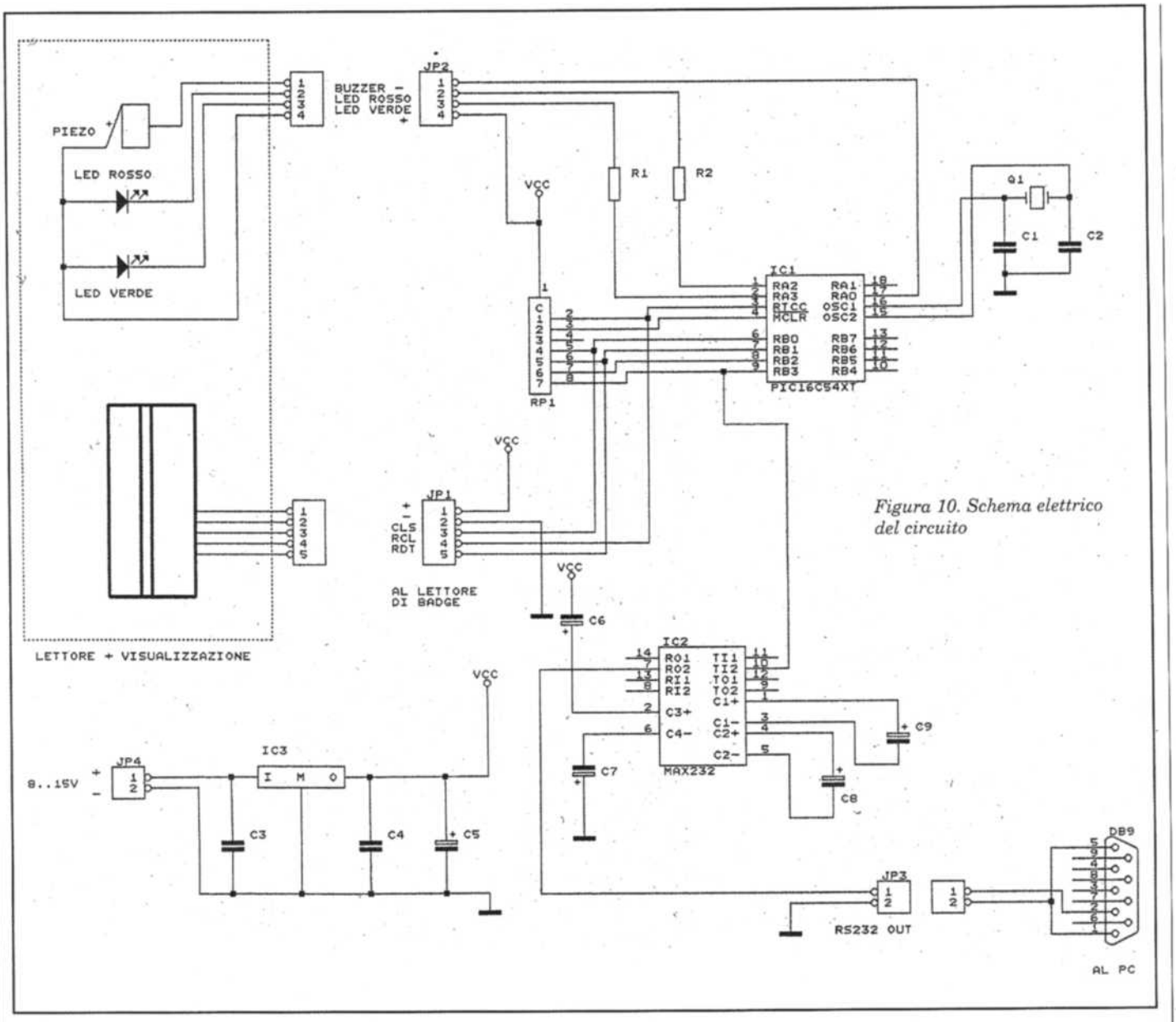


Figura 10. Schema elettrico del circuito

Analisi del circuito

In Figura 1 troviamo lo schema elettrico del circuito. In parte ricalca la struttura del precedente, ovvero ha identica la sezione di interfacciamento sia con il lettore di badge, sia con la parte di visualizzazione. Lo stesso dicasi per la sezione di alimentazione.

Invece, non troviamo più la memoria per i codici, ma al suo posto abbiamo un integrato che serve ad interfacciare i livelli di uscita del PIC con i livelli richiesti dallo standard RS-232.

Avremmo potuto togliere questo integrato, dato che costa abbastanza e sostituirlo con un più economico transistor e qualche altro componente passivo, ma non sarebbe più stato compatibile al 100% con lo standard richiesto.

Passiamo quindi allo studio del firmware, analizzando il diagramma a blocchi di Figura 11. Dopo la consueta e doverosa fase in cui vengono inizializzati i registri e la direzione delle porte, ci si mette in attesa di un carattere.

Quando arriva, si verifica che sia la Start Sentinel, in conformità con lo standard ISO7811 per la traccia 2. Se così non fosse, si attiva una routine di errore che fa eseguire tre beep veloci al buzzer ed attiva per un secondo il Led rosso.

Se, invece, il carattere giunto corrisponde alla Start Sentinel (bh), tale carattere viene letto e memorizzato in una locazione RAM del PIC.

Viene poi confrontato con l'End Transmission e, in caso affermativo, si ha l'invio di tutti e 32 caratteri letti (o non letti) sulla seriale, con il Led verde ed il buzzer che si attivano per circa due secondi. In caso contrario, si testa se il carattere giunto è il 32-esimo ed in caso positivo si esegue la trasmissione come prima.

Viceversa, ci si mette in attesa di un nuovo carattere e si torna alla sua lettura. Per amor di brevità, nel diagramma a blocchi non è stato indicato, ma è presente un controllo di time-out che fa entrare nella routine di ERRORE quando il segnale CLS viene a mancare, oppure il tempo tra un bit (o tra un carattere) ed il successivo, diventa troppo ampio.

In questi casi la lettura viene annullata e si ha la segnalazione dell'errore.

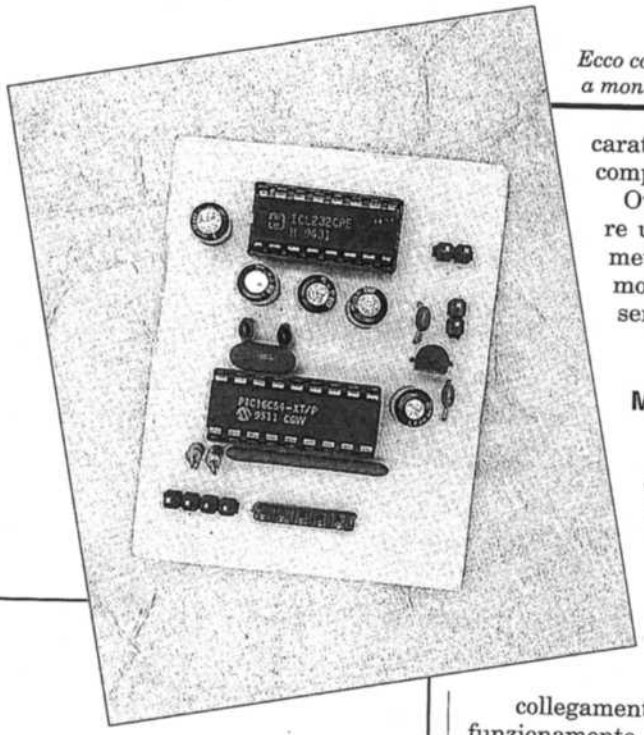
Per completezza, diciamo che quando avviene una trasmissione, vengono sempre spediti tutti e 32 i caratteri possibili.

Se la scheda ne aveva memorizzati in numero minore, quelli non letti verranno trasmessi come 0.

Ricordiamo inoltre che, per compattare i dati, ogni carattere di 5 bit è stato inserito in un nibble (4 bit) togliendo il bit della parità. In questo modo, quando un carattere viene spedito, occupa esattamente la metà di un byte, per cui con la trasmissione di un byte otteniamo l'invio di due caratteri. In questo modo otteniamo direttamente sul monitor del computer la strisciata dei caratteri già "ripuliti" del bit di parità.

Per fare un esempio, se prendete la vostra carta di credito VISA e la passate nel lettore, sul monitor del computer dovrete leggere correttamente il numero scritto sulla carta stessa più altri caratteri che non possiamo comprendere perché cifrati.

Ovviamente, dovrete avere un programma che permette la visualizzazione sul monitor dei dati giunti via seriale.



Ecco come appare la basetta a montaggio ultimato

Montaggio

La realizzazione del circuito è molto facile ed accessibile a tutti, sfruttando ad esempio il circuito stampato consigliato in Figura 12. In Figura 13 invece troviamo la disposizione dei componenti ed i collegamenti necessari al corretto funzionamento.

Ricordiamo di inserire il connettore del lettore di badge correttamente, onde evitarne una prematura fine. Anche la connessione con il computer è stata ridotta ai minimi termini: una massa ed il filo di trasmissione. Nello schema elettrico è riportato il collegamento con una presa DB9. Se avete una DB25 dovete saldare il filo di trasmissione al pin 3 invece che al 2.

Per collaudare il lettore, dovremo fornire per prima cosa un'alimentazione continua compresa tra 8 e 15 volt.

Poi dovremo collegare l'uscita seriale ad un computer, o sulla COM1 o sulla COM2. Per coloro che sono esperti di programmazione, sarà sufficiente scrivere un programma di gestione della porta seriale, anche in basic.

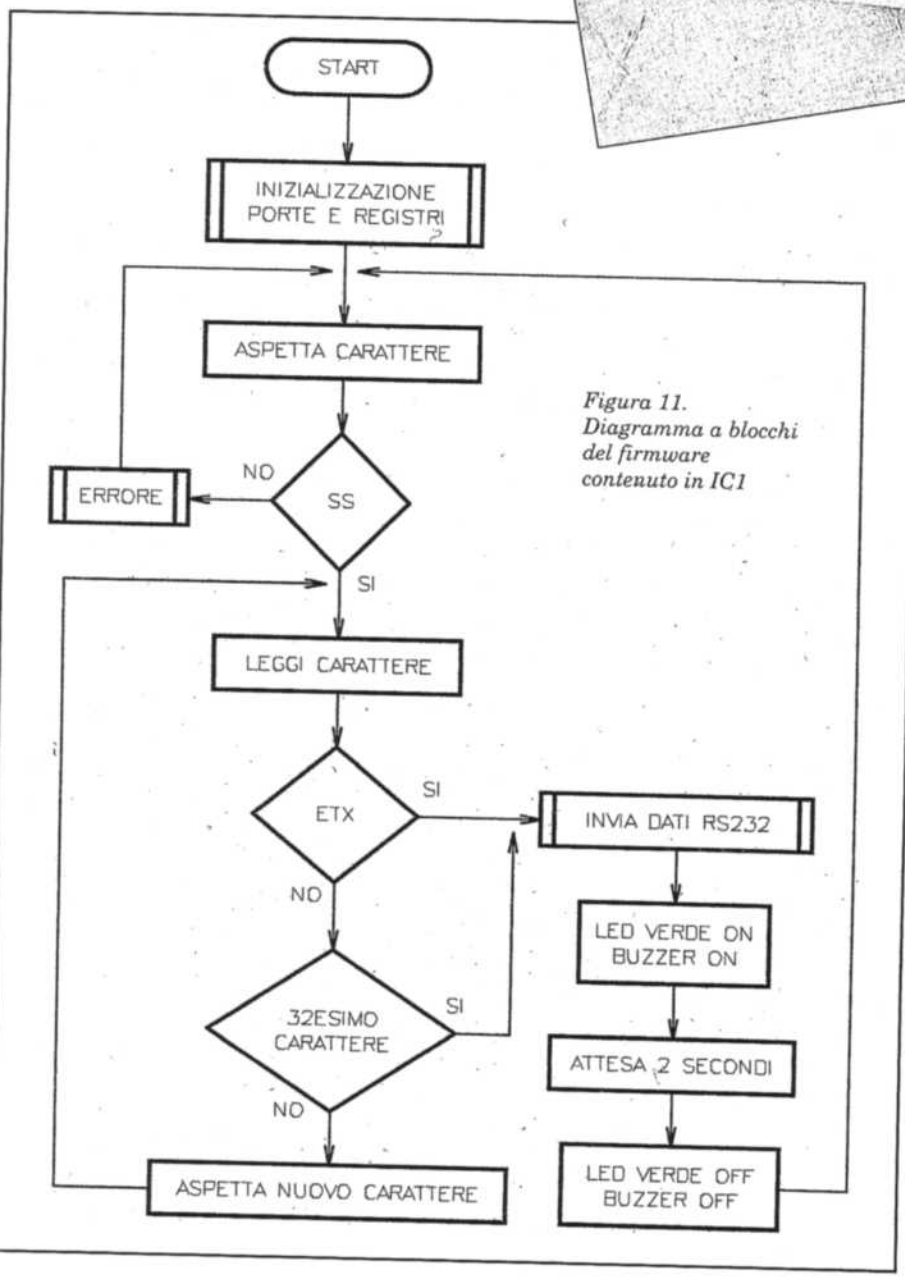


Figura 11. Diagramma a blocchi del firmware contenuto in IC1

ELENCO COMPONENTI

Semiconduttori
IC1: PIC16C54XT programmato (0337/259730)
IC2: MAX232 - **IC3:** 78L05

Resistori
R1, R2: 220 Ω - **RP1:** Rete 10 kΩ

Condensatori
C1, C2: 10 pF
C3, C4: 100 nF
C5: 47 µF 16 V
C6÷C9: 4,7 µF 16 V

Varie
Q1: Oscillatore 3,58 MHz

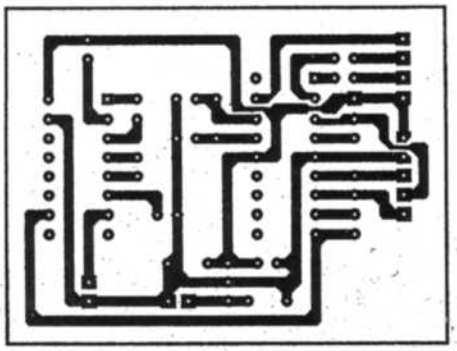


Figura 12. Circuito stampato, scala 1:1

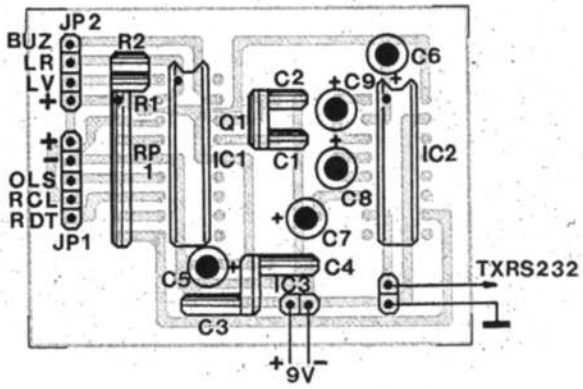


Figura 13. Disposizione dei componenti e collegamenti esterni

Abbiamo preparato un piccolo programma che, sfruttando routine di dialogo seriale pubblicate sul mensile Chip, riesce a visualizzare sul monitor i dati che arrivano dalla porta seriale.

La sintassi di avvio è la seguente: BADGE1 X dove X è il numero della porta seriale impiegata. Poi, sullo schermo appariranno automaticamente tutti i dati in arrivo su quella porta.

Per andare a capo, basta premere ENTER (o INVIO), mentre per uscire basta premere ESCAPE (o ESC). Per provare anche solo questo programma, basta lanciarlo sulla porta del mouse e vedere cosa succede muovendo il "topo".

Con l'applicazione di questo mese, terminiamo la trattazione delle carte magnetiche. Coloro che fossero interessati ad altri articoli, sono invitati a scrivere in redazione elencando le richieste specifiche.

Si ringrazia la Ditta EDUE Italia Spa, via Cassiani,155 (Modena) per la collaborazione e la documentazione fornitaci per la realizzazione di questo articolo e dove potrete trovare i lettori/scrittori descritti in queste pagine telefonando al numero 059/313403 o via fax al numero 059/314356. ■