Paolo Rognoni (Paolino)

LO HAI MAI REALIZZATO CON UN PIC?
ALEA ELECTRONICA IACTA EST, IL DADO E' TRATTO!

27 February 2010

No, non ci troviamo davanti al Rubicone con un PIC in mano... Ma chissà: una partita a Risiko la si può fare ugualmente. O, perché no, ad un altro gioco di società. Certo, ma non prima di aver realizzato un semplice **doppio dado elettronico**, controllato da **microcontrollore PIC**. Per continuare la rassegna de "Lo hai mai realizzato con un PIC?" propongo oggi un progetto semplice e ludico. Il progetto di per sè non presenta troppe difficoltà ma racchiude al proprio interno diversi aspetti che ritengo interessanti; alcuni di essi esulano dal mondo "PIC" ma senz'altro offrono spunti di riflessione. **Il progetto è corredato di schema, PCB, lista componenti, codice sorgente e firmware già compilato**, cosicché chiunque possa replicarlo in modo autonomo.

LE SPECIFICHE

Aprendo una scatola di un gioco di società è facile scoprire che per muovere le pedine sia necessario utilizzare uno o due dadi. A volte i giocatori si devono contrapporre lanciando entrambi (e magari contemporaneamente) un dado a testa. Volendo progettare un dado elettronico, questo è senz'altro un aspetto da tenere presente. Ragionando un po' su come fare, ho steso questo decalogo di specifiche:

1. il circuito deve prevedere due dadi a LED con colori differenti, uno rosso e uno verde;
2. deve essere previsto l'uso di LED ad alta luminosità al posto di quelli tradizionali;
3. il "lancio" deve essere sul singolo dado o su entrambi, distinguendo il caso di due persone che lanciano ciascuno il proprio dado oppure dalla singola persona che ne lancia due;
4. prevedere un cicalino (buzzer) eventualmente disinseribile;
5. prevedere di poter utilizzare un cicalino con o senza oscillatore interno;
6. la routine per l'estrazione dei numeri deve avere una buona affidabilità circa la casualità dei numeri estratti;
7. l'alimentazione del circuito deve avenire a batteria;

8. va utilizzato un microcontrollore con un numero sufficientemente ridotto di pin;
9. il micocontrollore deve portarsi in funzionamento a basso consumo quando il circuito è inattivo;
10. inserire un sistema per rivedere l'ultima estrazione effettuata.

RAPPRESENTAZIONE GRAFICA DEL DADO

Ciascun dado è costituito da 7 LED che, accendendosi in modo opportuno, mostrano i numeri da 1 a 6; è palese il fatto che con due dadi i LED da porre sono 14. Agendo di impulso si potrebbe pensare subito di impiegare un pin di I/O per pilotare il singolo LED. Ma si può fare di meglio? Non dimentichiamo il punto 8 della lista delle specifiche che chiede di utilizzare un microcontrollore con un ridotto numero di pin. Dotandoci di carta e penna si può cercare di ottimizzare le connessioni per cercare di rimanere in specifica. Si osservi la seguente figura, in cui sono riportati i LED che andranno a disegnare il dado. Ciascun punto (che poi diventerà un LED) del dado è stato contraddistinto da una lettera; sette punti, sette lettere, dalla A alla G.

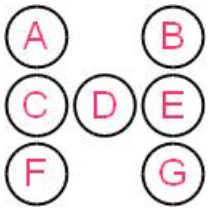


Figura_01.JPG

Ora, cosa accade se provassimo a disegnare i numeri del dado? Ecco qui rappresentati i numeri da 1 a 6, così come li vediamo abitualmente. Da una analisi attenta emerge che, fatto salvo il punto centrale (colorato di rosso), è possibile identificare tre coppie di LED: coppia azzurra, coppia verde e coppia arancione.

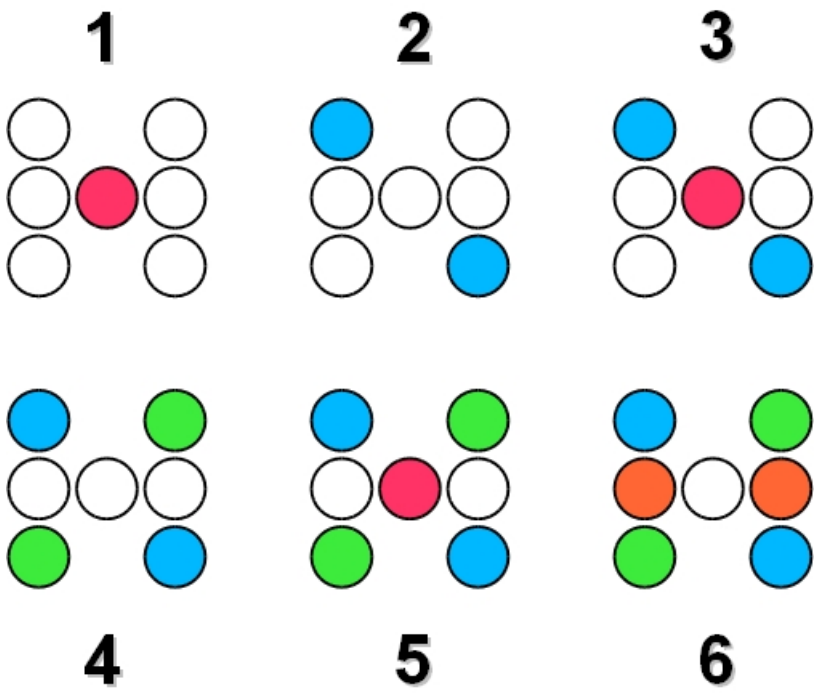


Figura_02.JPG

Questo aspetto permette di realizzare le combinazioni da 1 a 6 con sole quattro uscite del microcontrollore, avendo cura di **accorpare i LED** secondo quanto indicato nello schema seguente.

Così facendo, **con 8 pin di una sola porta si riescono a pilotare due dadi** con numerazione da 1 a 12.

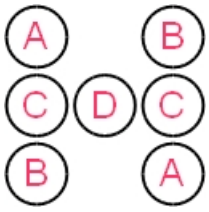


Figura_03.JPG

SCHEMA ELETTRICO

Il *brainstorming* ha portato allo sviluppo di uno schema al quale è seguito un circuito stampato. Vediamo in dettaglio lo schema.

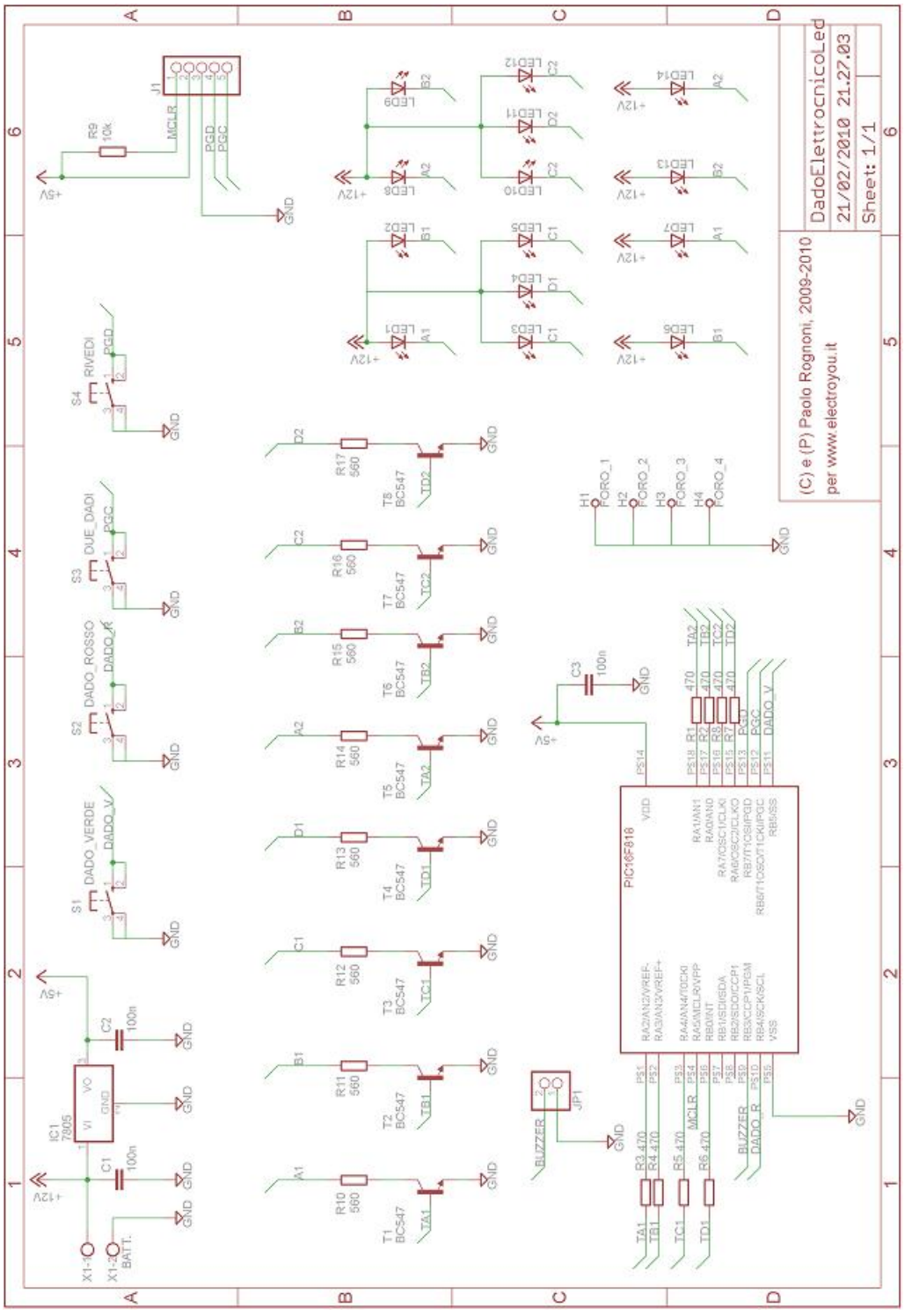


Figura_04.JPG

Lo stadio di alimentazione è costituito da una batteria a 9V in ingresso ad un regolatore di tensione 7805. Il microcontrollore scelto è un PIC16F819, PIC con 18 pin, tre timer, oscillatore integrato, sorgente di interrupt on change, PWM, ecc.

Quattro pulsanti si collegano al PIC (abilitando i pull-up interni); ciascun pulsante svolge le seguenti funzioni:

- S1: lancio del solo dado verde;
- S2: lancio del solo dado rosso;
- S3: lancio contemporaneo dei dadi verde e rosso;
- S4: visualizzazione dell'ultimo risultato di estrazione di numeri casuali.

I LED sono pilotati a coppie mediante i transistor T1, ..., T8 che hanno la base collegata al PIC (mediate una resistenza di polarizzazione), l'emitter a massa ed il collettore va ai LED, mediante una resistenza di limitazione di corrente. La scelta del valore delle resistenze di limitazione è funzione della corrente che attraverserà il LED. Per i LED "comuni", i valori suggeriti sono più che accettabili.

Dopo aver sbrogliato le piste ne è venuto fuori un circuito stampato che prima è stato visualizzato in modalità 3D e poi fisicamente realizzato. Si noti che nel protitpo **i LED sono stati montati senza accorciare i reofori**, per permettere l'inserimento in una scatoletta.

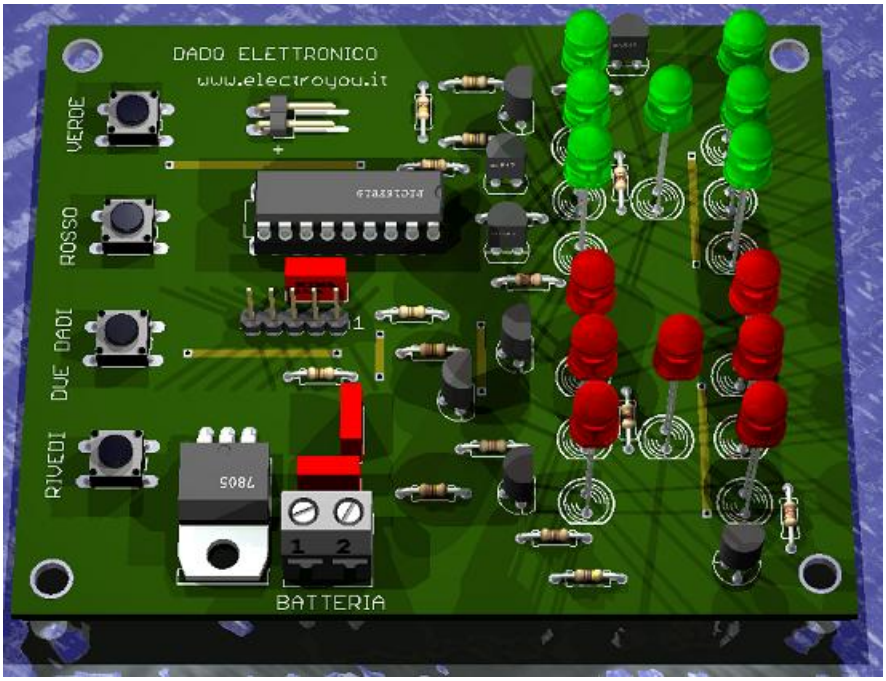


Figura_05.JPG

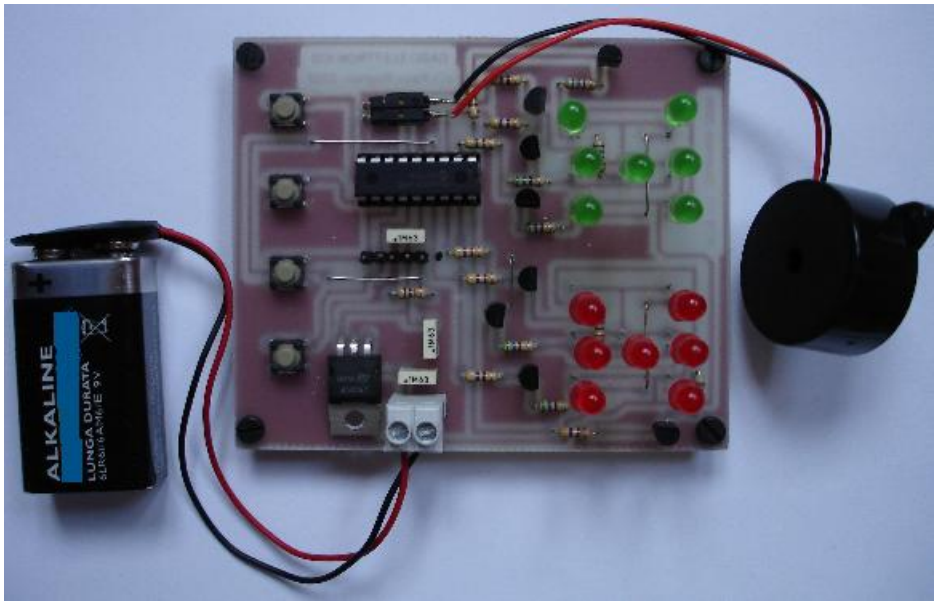


Figura_06.JPG

La **lista dei componenti** è la seguente:

- R1, ..., R8: 470 1/4 W
- R9 10k 1/4 W
- R10, ..., R17: 560, 1/4 W
- C1, C2, C3: 100n poliestere
- LED1, ..., LED7: LED 5mm verde
- LED8, ..., LED14: LED 5mm rosso
- T1, ..., T8: transistor BC547
- IC1: LM7805
- IC2: PIC16F819
- J1: pin strip 5 poli passo 2.54
- JP1: pin strip 2 pol, passo 2.54 (a 90°)
- S1, ..., S4: pulsanti N.O.
- X1: connettore a vite a due poli, passo 5.08

GENERAZIONE DEI NUMERI CASUALI

Questo è forse l'aspetto più interessante di tutto il progetto: riuscire a identificare un metodo in grado di generare numeri "abbastanza" casuali, cioè dove il rischio di periodicità sia effettivamente ridotto al minimo. Operando con i microcontrollori, la cosa più semplice che si può immaginare è osservare il valore di un timer lasciato libero di ciclare (gergalmente, lasciato in free-running): a seguito di un evento riconosciuto come "pulsante premuto", viene letto il valore del timer ed il suo valore viene scritto sul dado. Il principio è senz'altro valido ma nasconde delle

insidie se si pensa di dover estrarre due numeri casuali in modo consecutivo. per capire meglio come poter operare. supponiamo di voler estrarre in modo casuale un numero da 0 a 9 impiegando il timer TMR1; quando viene premuto un pulsante è sufficiente leggere il valore del timer e considerare solamente il valore delle unità. Supponendo per semplicità che ad ogni istruzione eseguita il PIC facesse avanzare di un tick il timer, se devono essere estratti due numeri in sequenza, il timer verrebbe sempre incrementato del medesimo valora. Nel caso in cui la chiamata alla routine di estrazione avvenisse in 10 colpi di clock, il timer si andrebbe a incrementare di 10 unità il che fa pensare che il secondo numero estratto coinciderebbe sempre con il primo (periodicità 10). Credo che un approccio interessante sia quello di andare ad accelerare o rallentare il timer ogni qual volta viene etratto un numero, in questo modo il secondo numero non dovrebbe essere necessariamente identico al primo. Ma come fare? Un metodo a mio avviso valido è quello di modificare la base dei tempi del timer in funzione del numero appena estratto, agendo sul prescaler. Quello che è venuto fuori dal ragionamento contorto che ho appena fatto, è espresso nella seguente routine in C.

```
char RandomNumber (void)
{
     char chNumeroCasuale;
// Estrazione del numero casuale
     chNumeroCasuale = ((TMR1L + (TMR1H>>8))>>(TMR0&1))%6 + 1;

// Ridefinizione del prescaler di TMR1 sulla base del numero estratto
     T1CON.T1CKPS0 = TMR0&1;
     T1CON.T1CKPS1 = TMR1L&1;

     return chNumeroCasuale;
}
```

Si osservi che il numero da rappresentare va da 1 a 6, quindi il valore estratto viene diviso per 6 mediante l'operatore modulo (%) che restituisce il resto della divisione come valore da 0 a 5. Basta aggiungere 1 al risultato ottenuto et voilà, il gioco è fatto.

IL FIRMWARE

Il firmware è stato scritto in **linguaggio C** e compilato mediante il compilatore **MikroC PRO**. Il codice propone alcuni aspetti legati all'antirimbalzo già discussi in questo articolo. Ho cercato di inserire molti commenti per permettere una facile lettura del codice. Il firmware, dopo la fase di inizializzazione del PIC (abilitazione dell'oscillatore interno, setup delle porte, impostazione dei timer, ecc). esegue un test di verifica e successivamente porta il microcontrollore nello stato di **SLEEP**,

stato dal quale si "risveglierà" solamente a fronte di particolari eventi, quali la risposta all'interrupt da evento esterno. I quattro pulsanti disponibili, sono gestiti con l'interrupt-on-change: quando un pulsante viene premuto il PIC si risveglia dallo stato di sleep ed esegue i controlli di antirimbalzo. Il timer impiegato per l'antirimbalzo è TMR0. Per quanto concerne l'aspetto della generazione di numeri casuali si è avuto modo di parlare nel paragrafo precedente. **Una breve nota circa il cicalino**; in commercio esistono fondamentalmente due tipi di buzzer, quelli con oscillatore integrato e quelli privi di oscillatore. Con i primi, è sufficiente fornire la giusta tensione ai suoi capi per sentirlo emettere una nota; nel secondo caso, invece, è necessario fornire un'onda quadra a frequenza sufficientemente bassa (qualche kHz) per permettere al buzzer di suonare. Ora, **il firmware prevede la gesione di entrambi**; nel caso in cui si disponga di un buzzer privo di oscillatore interno viene abilitato il modulo PWM del PIC sulla porta RB3 avendo cura di segnalare questo aspetto nei configuration bits, come indicato nel seguito.

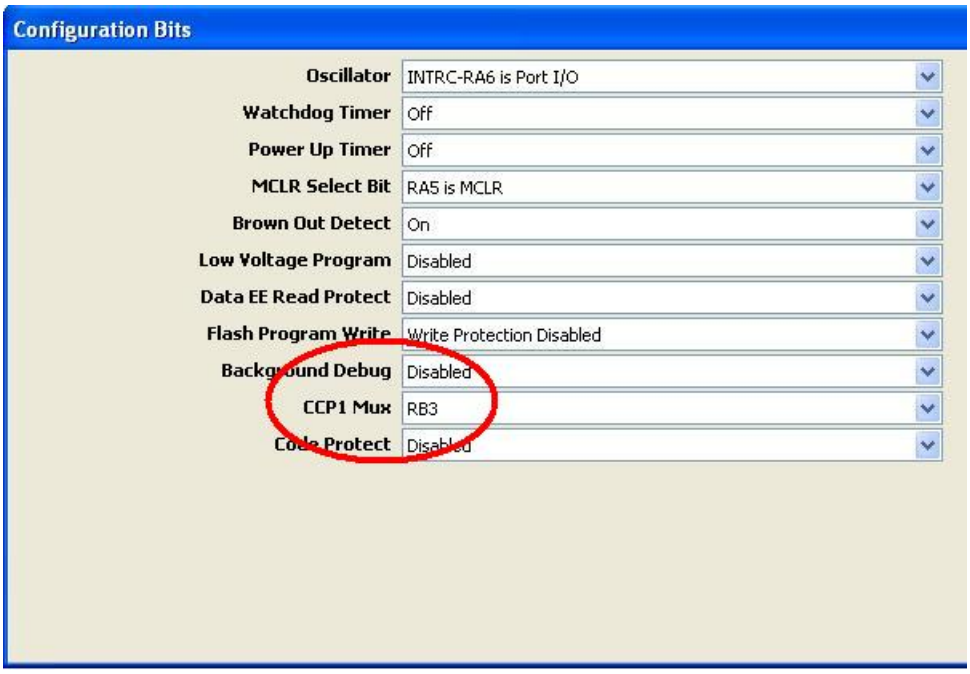
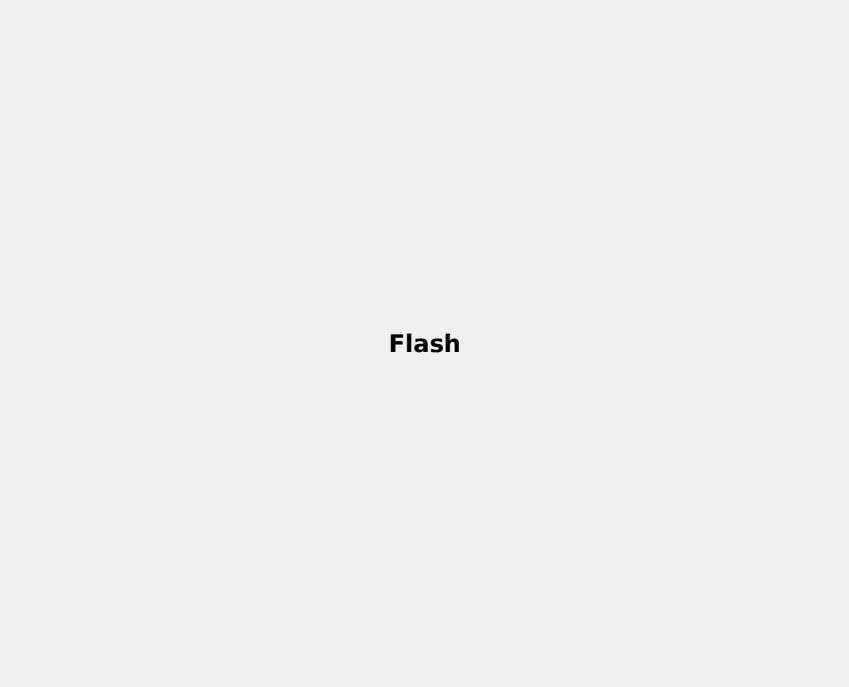


Figura_07.JPG

COME FUNZIONA

Una volta acceso, il circuito esegue il self-test, ossia va ad accendere in sequenza i LED e comanda il cicalino per un semplice "beep". Ogni qual volta il numero è estratto (e rappresentato sul/sui dado/i), il cicalino emette un beep di conferma. Il suono potrebbe essere fastidioso, quindi è possibile spegnerlo elettricamente (senza dover aprire la scatola per disconnettere il buzzer), avendo cura di accendere il dado mantenendo premuti contemporaneamente i pulsanti DADO VERDE e RIVEDI.

Dopo ciscun lancio, il circuito mostrerà il valore estratto (o i valori estratti, nel caso si lanciassero due dadi) per un tempo di circa due secondi, allo scadere dei quali inizia lo stato di sleep, stato di funzionamento va a basso consumo. Qualora si decidesse di lanciare entrambi i dadi ma con due giocatori, diversi, ciascun giocatore deciderà quale dado lanciare. Non è richiesta la simultaneità del tiro ma è sufficiente che entro due secondi dal lancio del primo dado venga lanciato il secondo. Per rivedere il punteggio ottenuto è sufficiente premere il pulsante RIVEDI. Ed eccolo qui funzionante, il dado elettronico.



FILE DI PROGETTO

Anche per questo progetto allego tutta la documentazione necessaria alla sua realizzazione. Oltre al codice sorgente e allo schema elettrico, allego anche il PCB in formato PDF da stampare in formato 1:1, per la realizzazione della scheda, così come lo schema di montaggio per la disposizione dei componenti. Per scaricare il materiale, fare click su questo link **http://www.electroportal.net/users/files/DadoElettronico.zip**

RINGRAZIAMENTI

Per la realizzazione della scatola devo ringraziare la pazienza e l'esperienza di **mio papà** che mi ha aiutato nel tagliare e incollare i vari pezzi di legno che la compongono. Un grazie va a **Zeno** e a **Nicolò** che mi offrono questo spazio per la pubblicazione on-line.

RIFERIMENTI

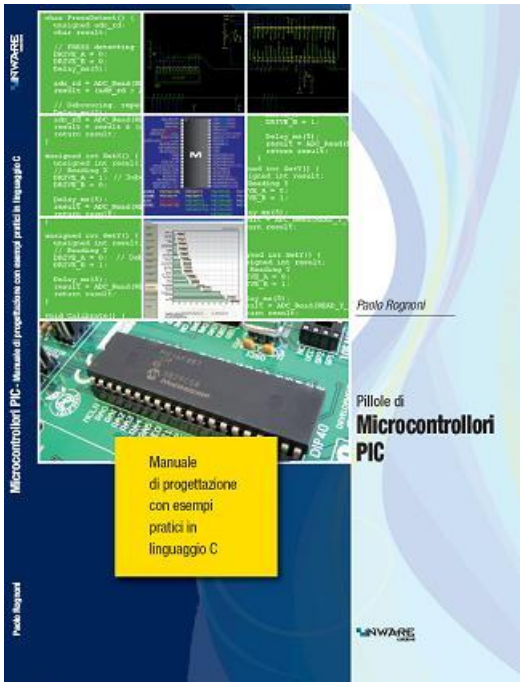
Lo hai mai realizzato con un PIC?:

- **Il conta marce**
- **Una sorpresa musicale per Babbo Natale!**
- **Una tecnica antirimbalzo**

Data sheet **PIC16F819**: http://ww1.microchip.com/downloads/en/DeviceDoc/39598e.pdf

MikroC PRO: http://www.mikroe.com/en/compilers/mikroc/pro/pic/

Pillole di microcontrollori PIC: http://www.inwaredizioni.it/pic2/



Pillole di Microcontrollori PIC.JPG

Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Paolino:dadoelettronico"