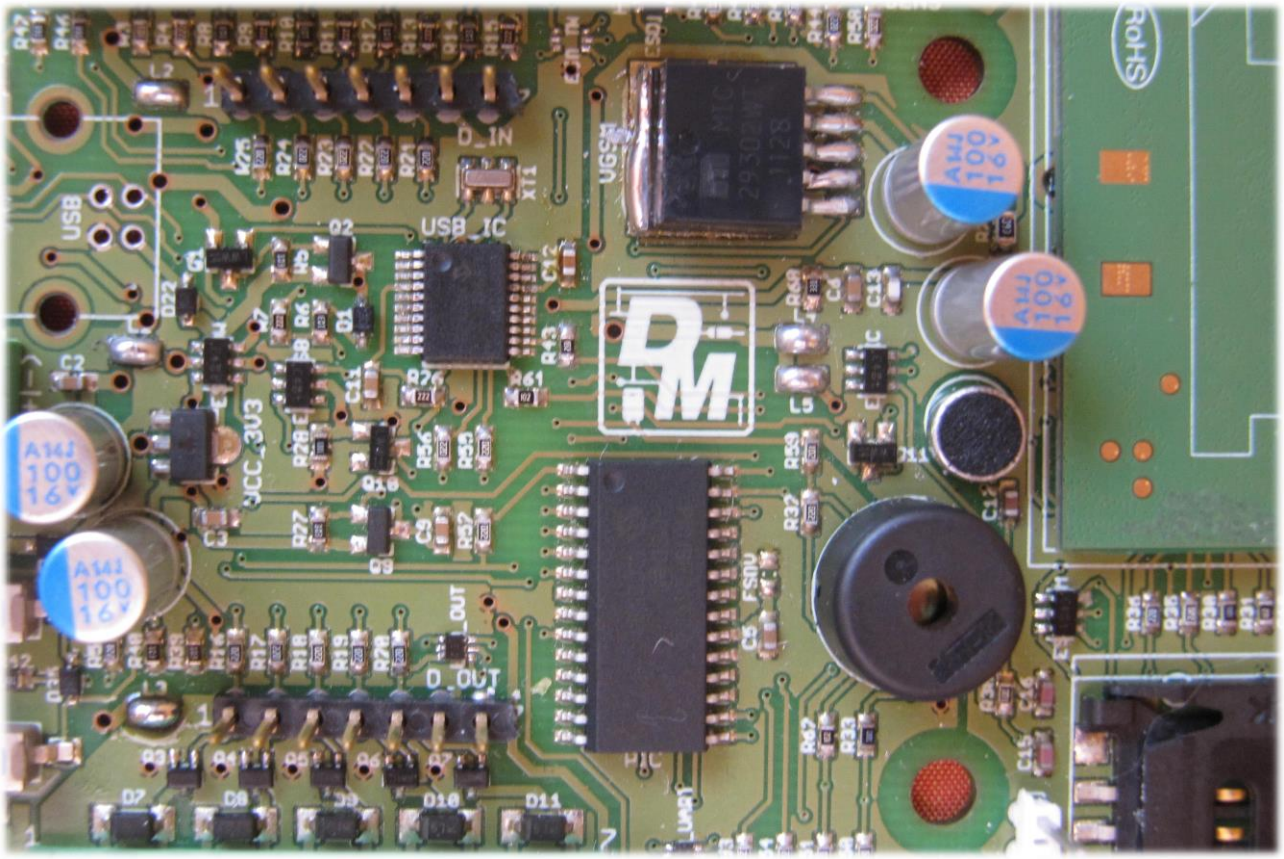
Panoramica del progetto DMBoard ICS

Sommario

1. HARDWARE



1.1 Cos'è DMBoard ICS

DMBoard ICS è un sistema elettronico personalizzabile dall'utente. DMBoard ICS è composto da una serie di dispositivi che possono essere gestiti semplicemente attraverso DMDesign, un tool di sviluppo grafico con procedure guidate che consente la completa gestione di DMDesign.

Attraverso le procedure guidate di DMDesign viene generato un programma in linguaggio DMstate che può essere caricato su DMBoard ICS.

DMBoard ICS è composto da un modulo GSM, da un microfono per l'ascolto ambientale, da un'uscita audio (dedicata al GSM), da 5 ingressi digitali/analogici, da 5 uscite a 12V (che possono pilotare direttamente dei relay), da una porta USB 2.0 e da una serie di porte digitali (I2C, RS232 ttl, SPI, ecc.); tutte queste periferiche sono personalizzabili attraverso DMDesign.

Tutto l'hardware di DMBoard ICS è stato progettato per funzionare a basso consumo e quindi è possibile utilizzare DMBoard ICS anche a batterie.

1.2 DMBoard ICS come strumento didattico

DMBoard ICS è un sistema open source con licenza Creative Commons e tutti gli schemi elettrici vengono forniti in forma completamente gratuita.

DMBoard ICS può essere quindi utilizzato come base per progettare sistemi a basso consumo programmabili o come strumento didattico.

Inoltre all'interno di DMBoard ICS è stata creata una macchina virtuale aggiornabile che converte le macroistruzioni scritte in DMasm (assembler generato da DMDesign) in azioni da eseguire.

2. SOFTWARE

2.1 Cos'è DMDesign

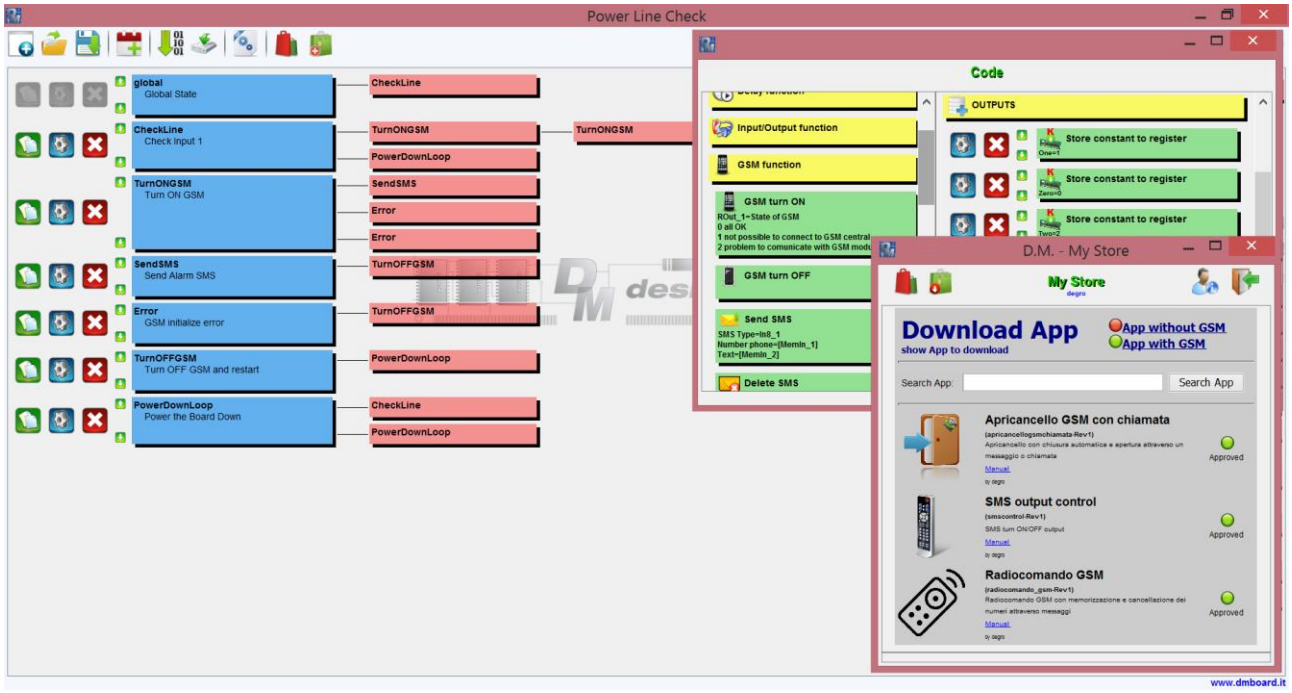
DMDesign è un tool di sviluppo innovativo che consente di creare, scaricare, gestire tutte le applicazioni che possono essere caricate sulla DMBoard ICS.

Quest'ultima è una scheda elettronica che monta un microcontrollore Microchip PIC18F26K22. Grazie al nuovo linguaggio chiamato DMstate, programmare un microcontrollore non è mai stato così semplice.

DMDesign consente l'inserimento delle singole istruzioni attraverso delle semplici procedure guidate riuscendo a svolgere compiti molto complessi in pochi semplici e veloci passaggi.

Tutti i programmi generati da DMDesign sono dei semplici file di testo che possono essere condivisi in rete attraverso uno store raggiungibile direttamente da DMDesign. Tutte le App

presenti nello store sono quindi in codice sorgente e una volta scaricate possono essere modificate a piacere.



2.2 Cos'è DMstate

DMstate è un linguaggio che si basa sugli automi a stati finiti.

Gli automi a stati finiti come la macchina di Mealy, sono ampiamente utilizzati nel campo dell'informatica e dell'ingegneria per risolvere problemi esistenti; grazie alla loro semplicità e chiarezza vengono ampiamente usati anche come strumento didattico in campo accademico.

La macchina di Mealy ha la particolarità di essere un automa a stati finiti che genera un'uscita a partire dagli stati d'ingresso e dallo stato corrente.

DMstate è la naturale traduzione della macchina di Mealy in un linguaggio di programmazione.

2.3 Struttura del linguaggio DMstate

DMstate ha la stessa struttura di una macchina di Mealy: è composto da una serie di blocchi chiamati stati.

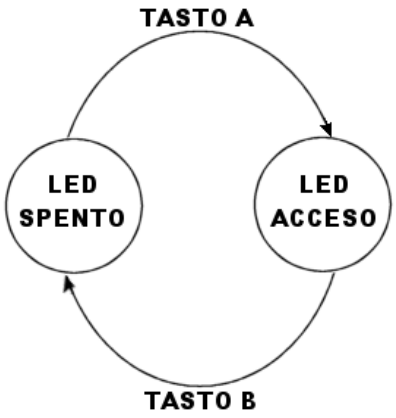
Ogni stato è composto a sua volta da 3 blocchi: variabili, uscite e salti.

Nel blocco variabili vengono definite tutte le variabili dello stato; nel blocco uscite, vengono definite tutte le azioni che dovranno essere compiute dallo stato; infine nel blocco salti, vengono definiti i salti tra uno stato e l'altro a seconda degli ingressi.

Una volta che si entra in uno stato, vengono immediatamente eseguite le azioni definite dal blocco uscite e successivamente vengono continuamente controllate le variabili di ingresso che definiscono i salti tra uno stato e l'altro.

Il linguaggio DMstate viene successivamente interpretato attraverso il tool di sviluppo DMDesign che consente direttamente la programmazione della DMBoard ICS.

Di seguito viene riportato il diagramma a stati di una semplice macchina di Mealy che accende un LED alla pressione del tasto A e lo spegne alla pressione del tasto B.



Per tradurre la macchina di Mealy in DMstate, dobbiamo quindi creare 2 stati (LED SPENTO, LED ACCESO), definire in ogni stato il valore dell'uscita nel blocco output, ed infine definire nel blocco jump a quale stato saltare a seconda dello stato degli ingressi (in questo caso i 2 pulsanti).

Proviamo quindi ad implementare la macchina di Mealy attraverso DMDesign:

Per prima cosa dobbiamo decidere quale sia il LED da controllare e quali siano i pulsanti da utilizzare per il suo controllo.

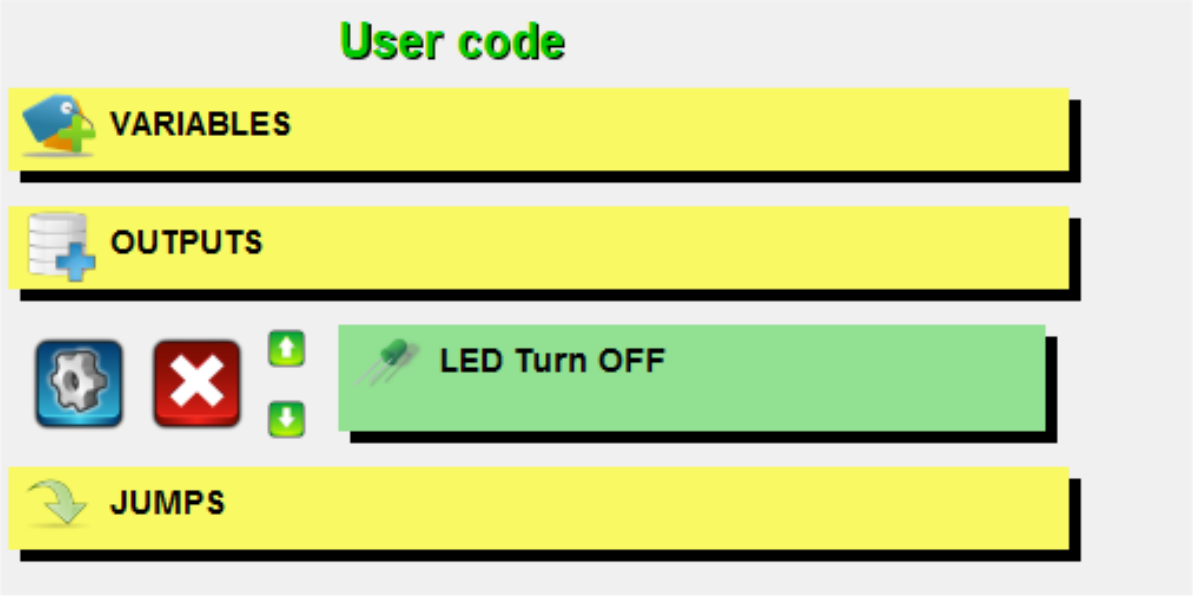
Pensando di implementare la macchina di Mealy sulla DMBoard ICS, utilizziamo il LED ed i 2 pulsanti montati direttamente sulla scheda.

Possiamo quindi iniziare definendo i 2 stati:

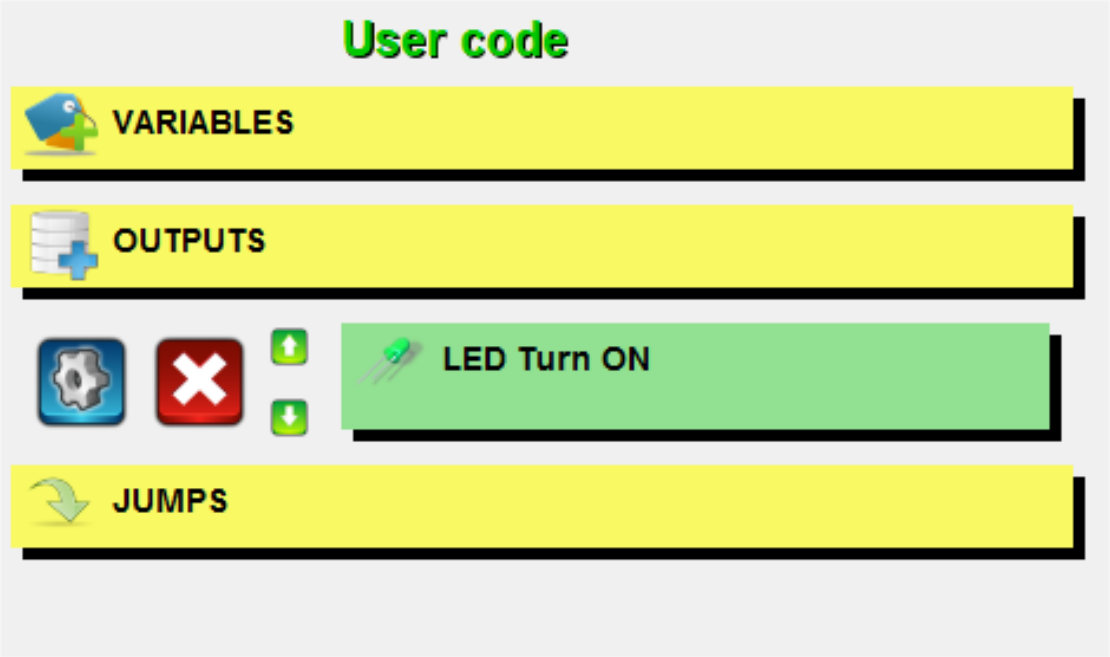
Si può notare come sia presente lo stato global; questo stato viene creato automaticamente ed è strutturato come tutti gli altri stati ma ha una valenza globale, ovvero definisce lo stato iniziale delle uscite e all'interno di questo stato possono essere definite tutte le variabili globali ovvero tutte le variabili che possono essere viste durante tutto lo svolgimento del programma.

Una volta definito gli stati dovremo definire all'interno di essi il valore delle uscite.

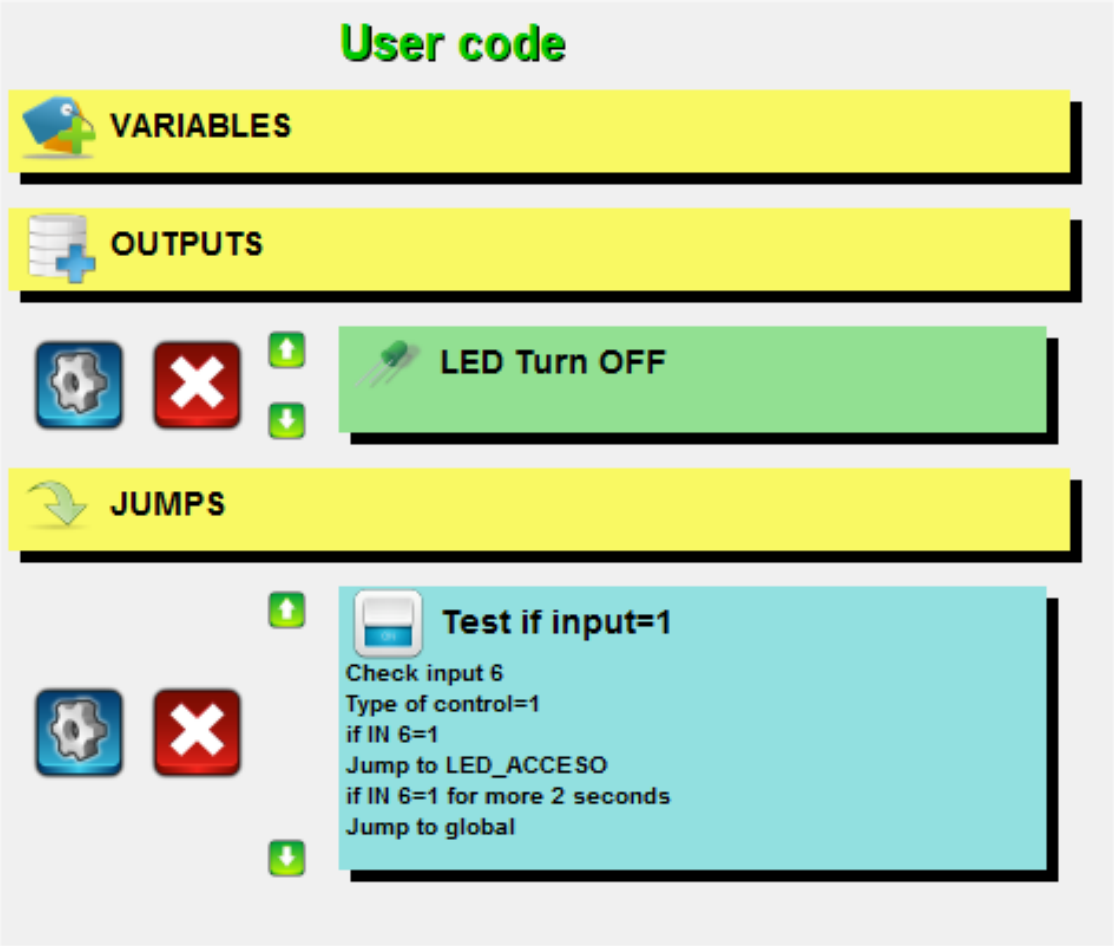
Per lo stato LED_SPENTO avremo:



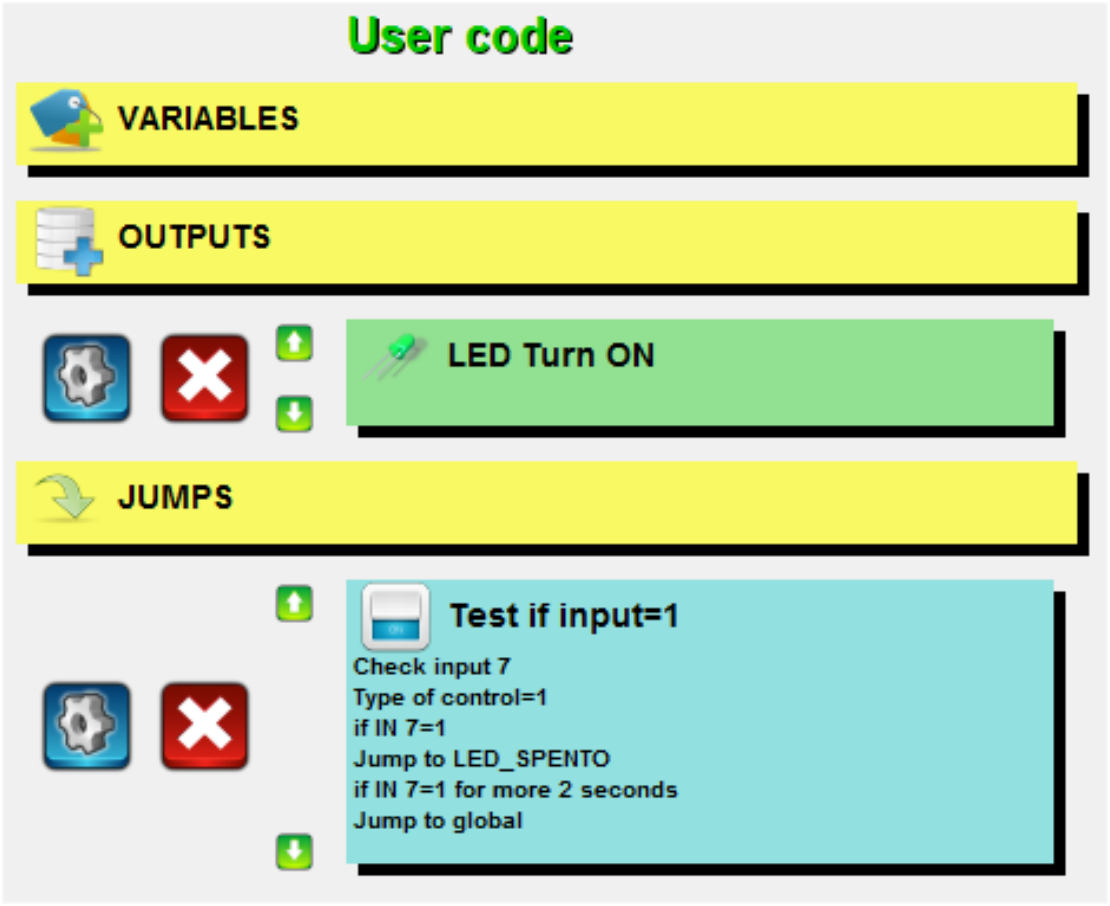
Mentre per lo stato LED_ACCESO avremo:



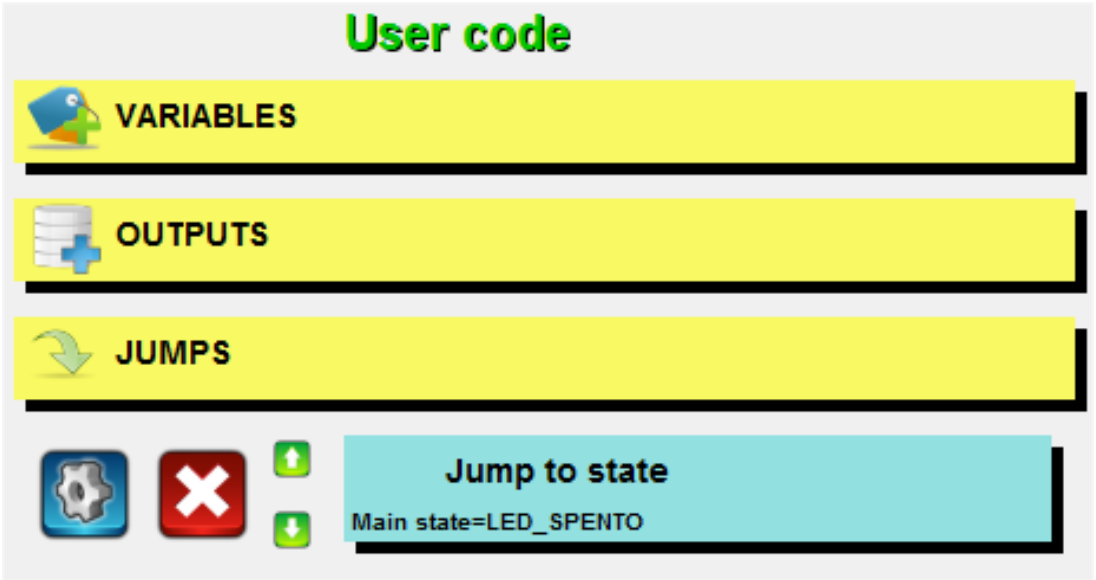
Ora dobbiamo passare a definire i salti tra uno stato ed un altro. Dallo stato LED_SPENTO dovremo passare allo stato LED_ACCESO alla pressione del pulsante A ovvero SW1 (che corrisponde all'ingresso 6 della DMBoard); utilizzando le procedure guidate di DMDesign otterremo:



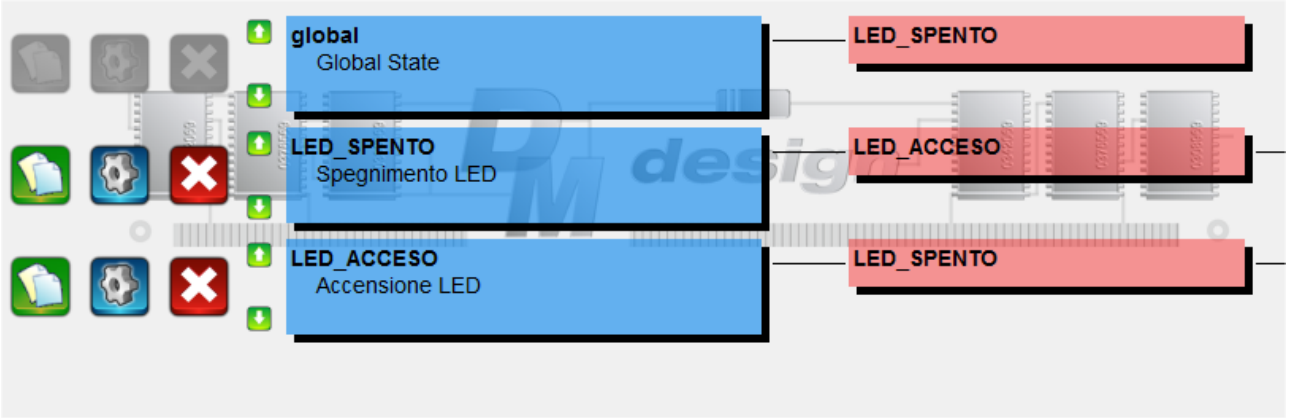
Per passare da LED_ACCESO a LED_SPENTO dovremmo attendere invece la pressione del tasto B ovvero SW2 (che corrisponde all'ingresso 7 della DMBoard):



Infine dovremo definire quale sia lo stato iniziale da cui partire. Questo stato dovrà essere definito all'interno dello stato global:



Il risultato finale sarà quindi il seguente:



Dove vengono rappresentati oltre che gli stati anche i salti condizionati tra uno stato e l'altro.