Placa demo para desarrolla… aplicaciones de telecontrol y telealarmas vía móvil, se programa fácilmente gracias a un entorno de programación visual en bloques. Segundo y última entrega: el entorno de desarrollo software.

DM Board ICS: EL TELECONTROL UNIVERSAL

DENIS DE GRANDIS Y MARCO SCIPIONI

En el primer episodio de este tutorial sobre la DM Board ICS, hemos introducido el proyecto y explicado que nace para permitir a los apasionados por la electrónica realizar sistemas gestionables desde smartphone en modo simple y rápido; el proyecto se basa en una tarjeta electrónica (DM Board ICS) que con pocas y sencillas instrucciones introducidas a través de procesos guiados (mediante el software DM Design, para ser exactos...) es capaz de ejecutar trabajos incluso muy complejos, como por ejemplo recibir o enviare mensajes SMS o ejecutar una llamada telefónica.

Ya hemos descrito el hardware de la DM Board ICS e ilustrado el circuito a implementar para poder realizar un antirrobo con envió de SMS de alarmas. Ahora nos centraremos en la parte del software de gestión de la DM Board ICS, empezando por entender como estructurar el programa para realizar nuestra alarma GSM.

Para simplificar al máximo la gestión del hardware de la DM Board ICS y hacer simple su programación, se ha decidido introducir en la tarjeta misma una máquina virtual llamada "DM Virtual Machine". Sobre esta máquina virtual, se carga el programa de usuario que es después interpretado y convertido en código máquina para ser ejecutado por el microcon-

Fig. 1

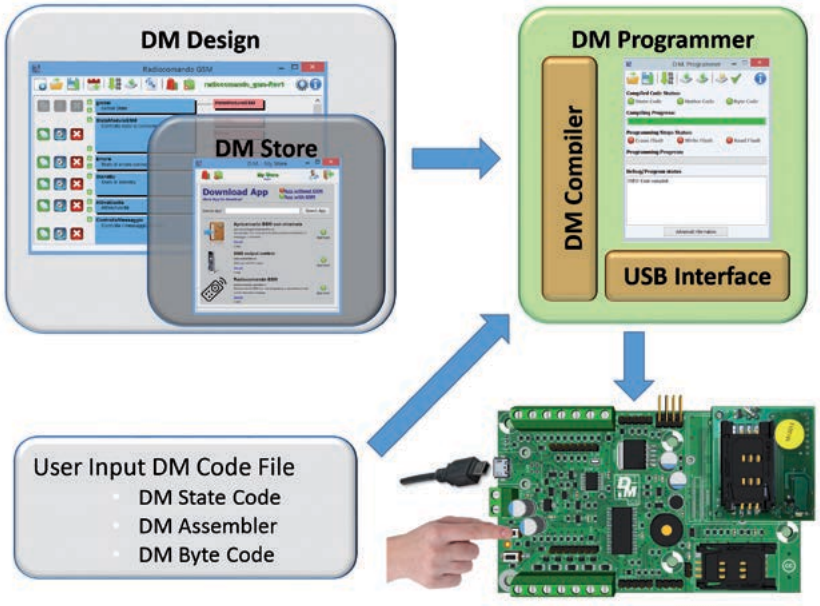
trolador.
La DM Virtual Machine acepta un número finito de instrucciones que sirven para gestionar todas las funciones de la DM Board ICS y funciona como interprete.
La DM Virtual Machine es actualizable a través de un bootloader precargado en la DM Board ICS y permite, mediante un simple proceso, cargar la nueva máquina virtual que podría, por ejemplo, poner a nuestra disposición nuevas instrucciones.
Podemos por tanto resumir las fases de programación de la DM Board ICS, que está estructurada como se muestra en el esquema de bloques de la **Fig. 1**: el programa usuario que se pasa a la DM Virtual Machine consta en un stream de byte que deriva de un lenguaje ensamblador evolucionado (se llama DM Assembler).
Incluso si el DM Assembler resulta ser bastante simple y de alto nivel, pensar en utilizar un lenguaje parecido para la programación de la DM Board ICS no habría supuesto ventaja alguna respecto al uso de un lenguaje convencional como el C. De hecho hemos creado un tercer lenguaje, llamado DM State, que es un lenguaje de estados, o formado por un conjunto de bloques cada uno de los cuales desarrolla una función particular del programa completo.
Este lenguaje ha sido finalmente convertido en un lenguaje gráfico, para hacerlo más fácilmente gestionable al usuario. Todos los lenguajes implicados son gestionados por el IDE DM Design.
La compilación es sin embargo demandada a la herramienta DM Programmer, que es lanzado directamente por DM Design y que se ocupa también de la programación de la DM Board ICS.
El programa de usuario, además de estar generado a través de DM Design, puede ser descargado directamente del DM Store, que es un espacio web que recoge una serie de interesantes aplicaciones creadas por los desarrolladores de la DM Board ICS y puestos a disposición de la correspondiente comunidad. El esquema mostrado en la **Fig. 2** resume como se realiza la fase de compilación/programación.
Veamos ahora cuales son los principios base que regulan el uso del lenguaje DM State.

Como ya se ha dicho anteriormente, DMstate es un lenguaje de estados, y está formado por un conjunto de bloques cada uno de los cuales desarrolla una particular función del programa; cada estado (por tanto cada bloque) tiene la particularidad de que las salidas deben permanecer constantes.
Los estados se conectan entre si a través de los saltos entre un bloque y el otro que dependen de los valores de las entradas o de los valores de algunas variables.
Cada programa, una vez estructurado, se convertirá en una simple máquina de estados.
Para quien no hubiese creado jamás una máquina de estados, mostramos a continuación una pequeña guía sobre cómo convertir nuestras alarmas con envió de SMS en una simple máquina de estados.

COMO ESTRUCTURAR UNA MAQUINA DE ESTADOS

Primero de todo debemos crear una lista de operaciones que la tarjeta debe desarrollar, posiblemente en orden cronológico sin

Fig. 2



preocuparse de la entrada que desencadena la operación.
Para implementar una alarma que envía mensajes SMS necesitaremos enviar los siguientes comandos:

1. encendido GSM;
2. apagado GSM (seguido de un error de encendido del GSM);
3. apagado sirena;
4. envió alarmas SMS;
5. encendido sirena.

En este punto es importante pensar en todos los eventos posibles que puedan desencadenar un comando y reagrupar los comandos sobrescritos para cada evento (**Tabla 1**). Podemos después reagrupar todos los comandos a ejecutar en orden, que no cambian al cambiar el evento: los encontráis en la **Tabla 2**.
Hemos identificado automáticamente el número de estados del programa que en nuestro caso corresponderá a 4 estados más un estado creado por defecto llamado global (el objetivo de este último estado será analizado posteriormente).
El estado se convierte por tanto en un conjunto de comandos que deben ser ejecutados después de un evento.
Antes de pasar a la verdadera y propia implementación del programa sobre DM Design, debemos actualizar nuestra tabla de estados (**Tabla 2**) con dos nuevas columnas: una que indique a qué estado debemos saltar después de haber ejecutado una serie de comandos y una segunda que indique si debe ser respetado un tiempo de permanencia en un determinado estado antes de pasar al siguiente (**Tabla 3**).
Ahora en la **Tabla 3** insertamos, sobre cada línea de la columna "estado a que saltar", el evento que desencadena el salto; deriva la **Tabla 4**.

Tabla 1

Numero evento	Evento	Comando a ejecutar
1	Encendido	Apagado sirena / Encendido GSM
2	Error encendido	Apagado GSM
3	Encendido ocurrido con éxito	Apagado sirena
4	Sensor 1 informa de robo	Envío SMS alarmas / Encendido sirena
5	Sensor 2 informa de robo	Envío SMS alarmas / Encendido sirena
6	Sensor 3 informa de robo	Envío SMS alarmas / Encendido sirena
7	Sensor 4 informa de robo	Envío SMS alarmas / Encendido sirena
8	Sensor 5 informa de robo	Envío SMS alarmas / Encendido sirena

Tabla 2

Numero evento	Evento	Comando a ejecutar
1	Encendido	Apagado sirena / Encendido GSM
2	Error encendido	Apagado GSM
3	Encendido ocurrido con éxito	Apagado sirena
4	Sensor 1 informa de robo Sensor 2 informa de robo Sensor 3 informa de robo Sensor 4 informa de robo Sensor 5 informa de robo	Envío SMS alarmas Encendido sirena

Tabla 3

Numero stato	Evento	Comando a ejecutar	Estado en el cual saltar	Tiempo mínimo permanencia
1	Encendido	Apagado sirena Encendido GSM	2 3	Apagado sirena Encendido GSM
2	Error encendido	Apagado GSM	1	Apagado GSM
3	Encendido con éxito	Apagado sirena	4	Apagado sirena
4	Sensor 1 informa de robo Sensor 2 informa de robo Sensor 3 informa de robo Sensor 4 informa de robo Sensor 5 informa de robo	Envío SMS alarmas Encendido sirena	3	Envío SMS alarmas Encendido sirena

Podemos finalmente eliminar, de esta última tabla, la columna "Evento", ya que resulta ser una repetición, y establecer como estado principal (o el primero a ejecutar) el estado numero 1; se llega así a la **Tabla 5**.
Como se ve en la tabla, al final de la ejecución de una serie de comandos contenidos en un estado, es posible saltar a un nuevo estado solo si se lanza el evento que permite acceder a aquel estado; por ejemplo, en el estado 1 están presentes dos estados a los que saltarsegún el evento que es lanzado al final de la ejecución de la serie de comandos, se puede saltar al estado 2 (en el caso de error de conexión del módulo GSM) o al estado 3 (en el caso de que la conexión tenga éxito).
En el estado 4 podemos ver como está presente un tiempo de permanencia de 10 segundos, que corresponde al tiempo de encendido de la sirena.
Antes de ir a ver como traducir la **Tabla 5** en DM Design, veamos como este último organiza los estados.

ORGANIZACION DE LOS ESTADOS

En DM Design, cada estado está dividido en tres bloques:

- **Variables**: en este bloque se definen las variables numéricas (o numéricas y alfanuméricas para el estado global);
- **Outputs**: en este bloque se define el funcionamiento del estado asignando los valores

Tabla 4

Numero stado	Evento	Comando a ejecutar	Estado en el que saltar	Evento que genera el salto	Tiempo mínimo permanencia
1	Encendido	Apagado sirena Encendido GSM	2 3	Error encendido Encendido con éxito	0 segundos
2	Error encendido	Apagado GSM	1	Ninguno	0 segundos
3	Encendido con éxito	Apagado sirena	4 4 4 4 4	Sensor 1 informa de robo Sensor 2 informa de robo Sensor 3 informa de robo Sensor 4 informa de robo Sensor 5 informa de robo	0 segundos
4	Sensor 1 informa de robo Sensor 2 informa de robo Sensor 3 informa de robo Sensor 4 informa de robo Sensor 5 informa de robo	Envío SMS alarmas Encendido sirena	3	Ninguno	10 segundos

Tabla 5

Numero estado	Comandos a ejecutar	Estado en el cual soltar	Evento que genera el salto	Tiempo mínimo permanencia
1*	Apagado sirena Encendido GSM	2 3	Error encendido Encendido con éxito	0 segundos
2	Apagado GSM	1	Ninguno	0 segundos
3	Apagado sirena	4 4 4 4 4	Sensor 1 informa de robo Sensor 2 informa de robo Sensor 3 informa de robo Sensor 4 informa de robo Sensor 5 informa de robo	0 segundos
4	Envío SMS alarmas Encendido sirena	3	Ninguno	10 segundos

** Estado principal.*

de las salidas o de las variables, insertando eventuales retrasos, enviando mensajes, enviando llamadas, etcétera;

- **Jumps**: este bloque conecta entre ellos los estados a través del valor de las entradas o de las variables definidas anteriormente.

En el interior del bloque **Variables**, pueden ser definidas variables numéricas (llamadas registros) o un valor numérico identificado por un nombre que puede variar en el interior del estado. Estas variables definidas en el interior del estado valen solo para aquel estado.

Para poder definir una variable que sea vista en el interior de todos los estados del programa, deberemos servirnos de un estado particular llamado *global*, que se crea para configuración predefinida sobre todos los programas, y definir en el interior de este estado la variable que nos interesa. Este particular estado, llamado global, tiene también otras funciones que listamos a continuación:

1. definir las variables que deberán ser compartidas (y por tanto vistas) en todos los estados;
2. inicializar la tarjeta o definir como deberán ser las salidas;
3. definir cuál debe ser el estado principal.

Además de las variables numéricas, existen también las variables alfanuméricas. Mientras las variables numéricas, según de donde vienen definidas, asumen un valor local o global, los alfanuméricos pueden ser definidos solo en el estado *global* y por tanto siempre son globales.

Entre las variables numéricas y alfanuméricas, existe también otra distinción: las primeras, una vez que han sido definidas, son guardadas en RAM (una memoria volátil que si se quita la alimentación a la tarjeta se pierde el contenido), mientras las segundas (aquellas alfanuméricas) son guardadas directamente en Flash (una memoria no volátil que si falta la alimentación mantiene el contenido).

La DM Board ICS ejecuta secuencialmente las instrucciones incluidas en el bloque **Outputs** y, posteriormente, ejecuta las instrucciones del bloque **Jumps**, con la diferencia que si en este último no encuentra una instrucción de salto, ejecuta indefinidamente tal bloque; este modo de proceder hace que no se salga de un estado hasta que no se haya comprobado un evento de aquellos contenidos en el bloque **Jumps**.

Bien, explicado también esto, podemos pasar a ilustrar el funcionamiento de DM Design (el IDE de gestión de la DM Board ICS...)

y a implementar el programa necesario para realizar un sistema de alarmas capaz de enviar SMS, llevándolo después a cargar en el microcontrolador que controla la DM Board ICS.
En la práctica se trata de transformar la **Tabla 5 e**n un programa que la DM Board ICS ejecutara.
Asignamos nombres a cada uno de los 5 estados (4 estados + el estado *global*):

0. global
1. Encendido_GSM (Accensione_GSM)
2. Apagado_GSM (Spegnimento_GSM)
3. Espera_Llegada_Alarmas (Attesa_Arrivo_Allarme)
4. Envío_alarmas (Invio_allarme)

Nota: Para facilitar la comprensión del funcionamiento del programa DM Desing hemos traducido los nombres y dejado los nombres originales entre paréntesis para concirdar con las ilustraciones.

Recordamos que cada estado en DM Design debe estar dividido en tres bloques, es decir, **Variables**, **Outputs** y **Jumps**. Por el momento no analizamos las **Variables** en cuanto conviene utilizar DM Design para entender cuántas y cuales son necesarias para la realización de nuestro programa.
Los **Outputs** corresponden a los comandos insertados en la **Tabla 5**, mientras los **Jumps** corresponden a los eventos reportados en la misma tabla, que sirven para hacer saltar de un estado a otro.
Vayamos por tanto a dividir cada uno de los cinco estados en tres bloques e insertar para cada bloque (excepto para las variables) las instrucciones a implementar:

1. global
   - Variables
   - Outputs
   - Jumps
     - Estado principal = Encendido_GSM

2. Encendido_GSM

Fig. 3

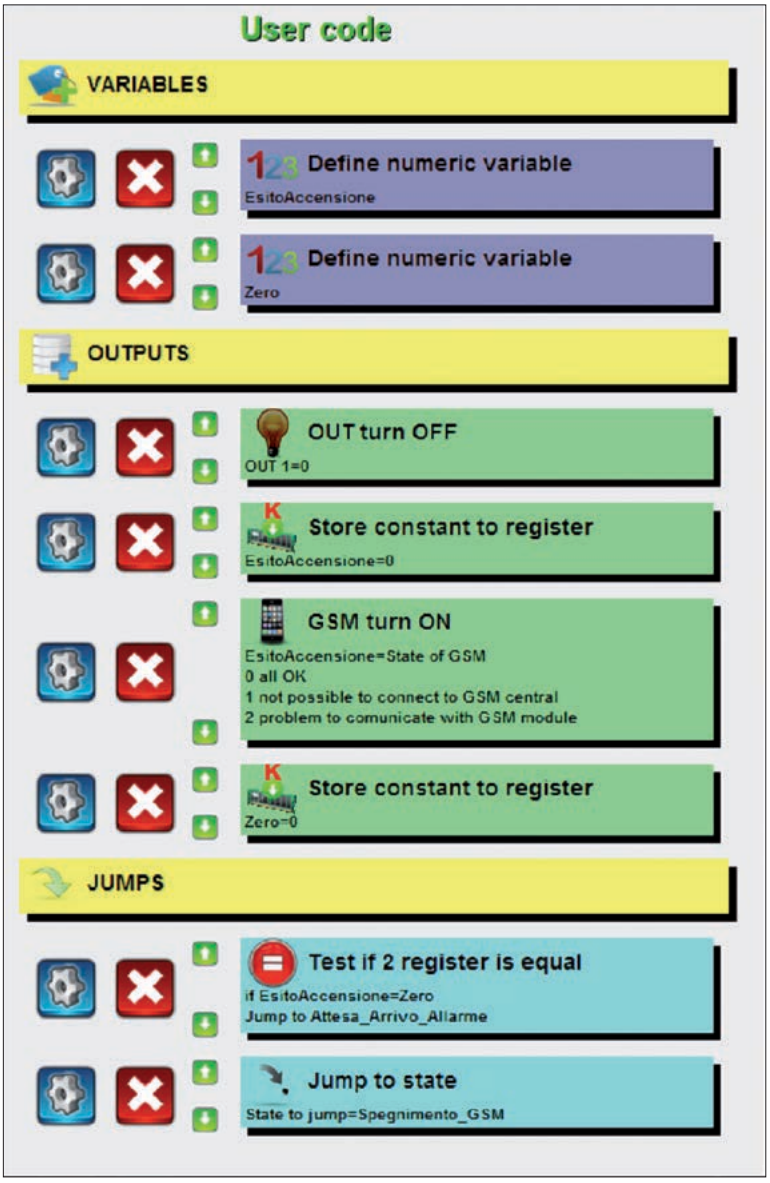
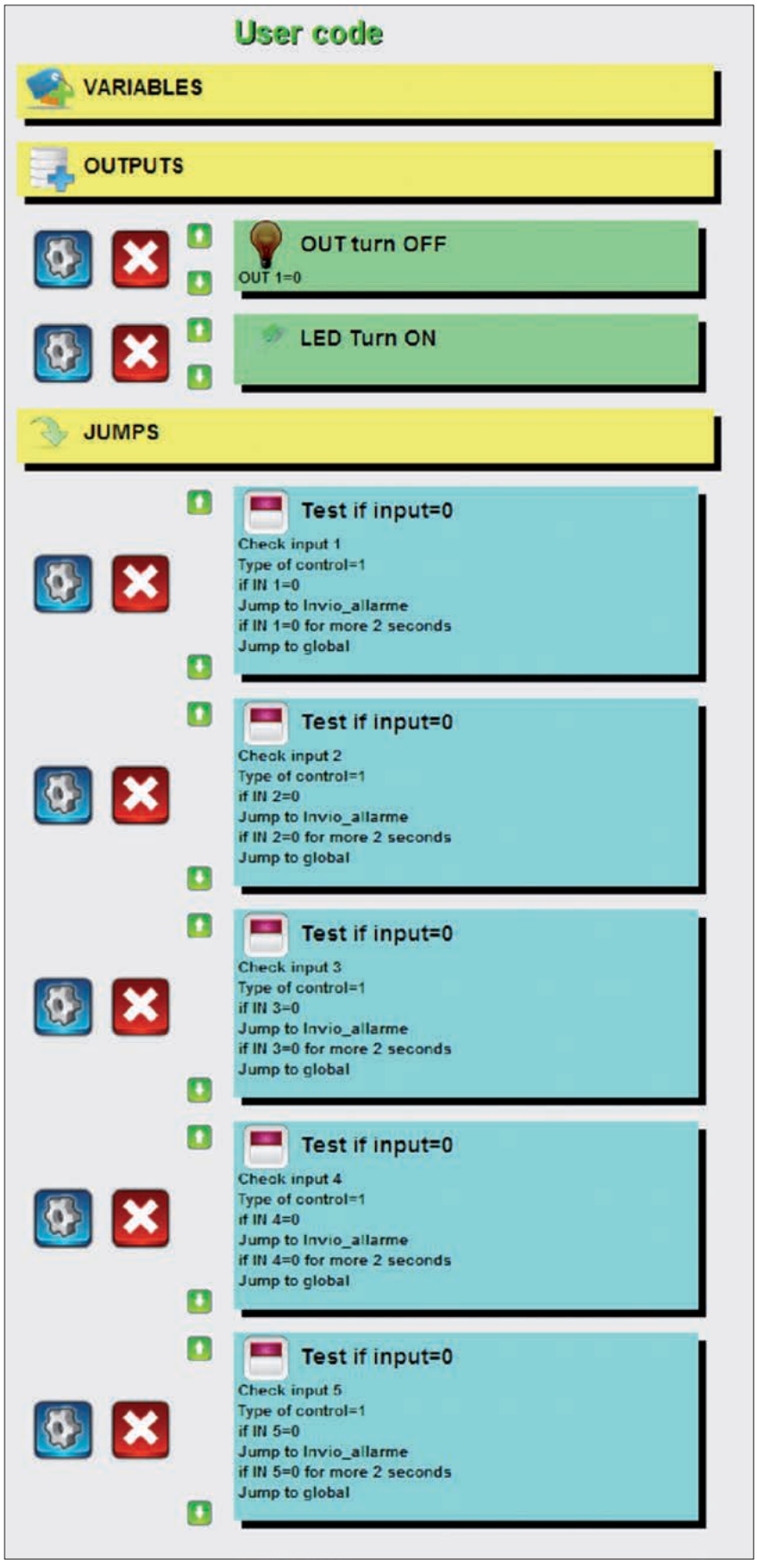


Fig. 4

Fig. 5



Variables
Outputs
Apagado sirena
Encendido GSM
Jumps
Si el GSM se ha encendido,
salta en Espera_Llegada_Alarmas
Si el GSM no se ha encendido, salta en Apagado_GSM

3. Apagado_GSM
Variables
Outputs
Apagado GSM
Jumps
Salta en Encendido_GSM

4. Espera_Llegada_Alarmas
Variables
Outputs
Apagado sirena
Jumps
Si sensor 1 informa de robo, salta en Envío_Alarmas
Si sensor 2 informa de robo, salta en Envío_Alarmas
Si sensor 3 informa de robo, salta en Envío_Alarmas
Si sensor 4 informa de robo, salta en Envío_Alarmas
Si sensor 5 informa de robo, salta en Envío_Alarmas

5. Envío_Alarmas
Variables
Outputs
Envío SMS de alarmas
Encendido Sirena
Espera 10 segundos
Jumps
Salta en Espera_Llegada_Alarmas

Como puedes ver, después de

haber dividido cada estado en tres bloques distintos, hemos reportado en **Outputs** la columna "Comandos a ejecutar" y en **Jumps** la columna "Estado en el cual saltar" unida a la columna "Evento que genera el salto".
En el estado *global*, a diferencia de los otros estados, el bloque **Jumps**, sirve para definir cuál es el estado principal.
Tenemos por tanto estructurado completamente el programa que realiza nuestro sistema de alarmas con envío SMS; ahora debemos simplemente cargar la estructura arriba indicada en DM Design.

IMPLEMENTACION DE LA ALARMA CON ENVIO DE SMS EN DM DESIGN

En este artículo no analizaremos en detalle cómo crear un nuevo proyecto en DM Design; para esto os remitimos a la guía "Il mio primo programma" ("Mi primer programa"), en italiano, descargable de la web *www.dmboard.it* bajo la sección *Tutorial*, la cual ilustra paso a paso como crear un simple programa de parpadeo de un LED, para entender cómo usar los comandos base de DM Design.
Recordamos, aun así, que el módulo GSM y todas las instrucciones de gestión de las entradas/salidas son gestionados nativamente por la tarjeta y por tanto con pocos y simples procesos guiados conseguiremos realizar nuestro programa.
En estas páginas vemos, sin embargo, directamente como cargar la estructura arriba indicada: para empezar deberemos crear los 5 estados de los cuales se compone nuestro programa.
A través del pulsador debemos insertar los cuatro estados (**Fig. 3**); después entramos en el estado "Encendido_GSM" y vamos a insertar esto que hemos visto anteriormente:

1. Encendido_GSM
   Variables
   Outputs
   Apagado sirena
   Enciende GSM
   Jumps
   Si el GSM se ha encendido,
   salta en Espera_Llegada_Alarmas
   Si el GSM no se ha encendido, salta en Apagado_GSM

Primero deberemos añadir las dos instrucciones en el bloque **Outputs**. La sirena está conectada a través del relé a la salida 1, por tanto para apagar la sirena bastara poner a 0 la salida 1.
Todas las instrucciones de salida se encuentran en la opción **Outputs** del menú lateral y están divididas por tipología.
Las instrucciones pueden tener parámetros, como por ejemplo la instrucción de encendido o apagado de una salida; estos parámetros son seleccionables a través de los menús desplegables.
La instrucción de encendido del módulo GSM requiere, por ejemplo, el uso de una variable

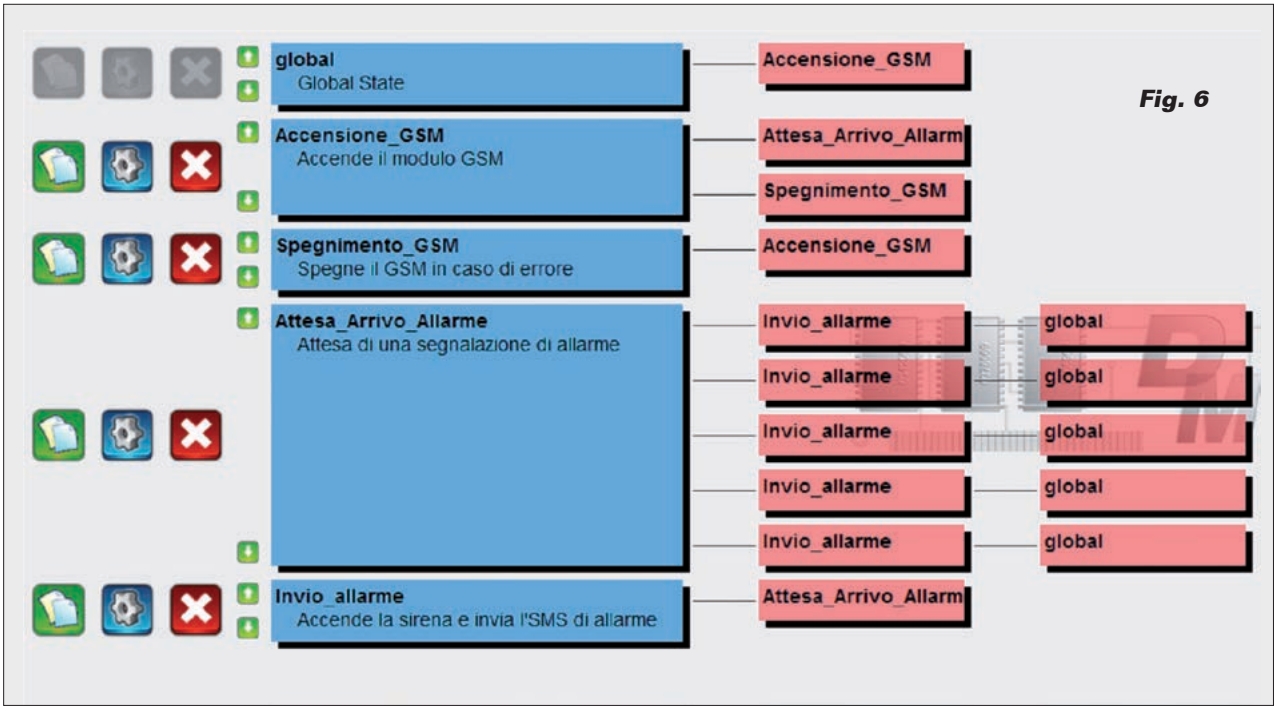


Fig. 6

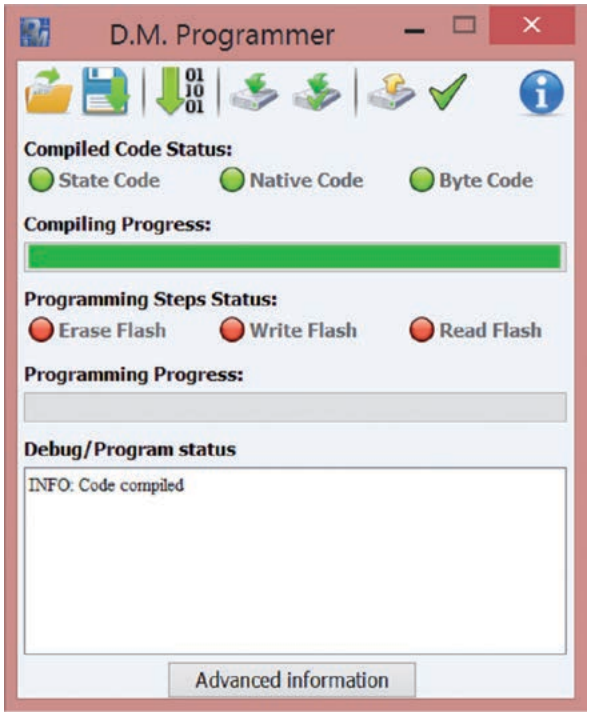


Fig. 7 *Inicio de Dm Programmer con el cual transferir el programa escrito a la memoria del microcontrolador de la DM Board ICS.*

para el guardar el resultado del encendido. Directamente de la instrucción, es posible arrancar un proceso guiado para la creación de la variable. Automáticamente la variable será visualizada en el menú desplegable de la instrucción que la requiere. Según el tipo de variable que se pretende insertar, será automáticamente adjunta la creación de la variable en el estado local o en el estado global.

Para lo que se refiere a las instrucciones de salto, se insertadas a través de opción "Jumps" del menú lateral que se encuentra en el interior de cada estado.

La modalidad de inserción resulta ser la misma que de las instrucciones de **Outputs**.

Una vez insertadas todas las instrucciones, para el primer estado encontraremos una pantalla organizada como se muestra en la **Fig. 4**.

Para los otros estados, el proceso es el mismo; en el estado Espera_Llegada_Alarmas deberemos leer continuamente el estado de las entradas magnéticas conectadas a las 5 entradas de la DM Board ICS. Cuando un sensor informa de la alarma, la entrada pasara de 12V (1 lógico) a 0V (0 lógico).

En el estado de jump de este estado deberemos adjuntar 5 controles para las 5 entradas obteniendo por tanto un estado así organizado (**Fig. 5**).

Podemos indicar que se ha introducido una instrucción de encendido LED (que se encuentra dentro "LED Function") la cual tiene el objetivo de encender el LED contenido en la DM Board ICS cuando el módulo GSM se ha encendido. Esta instrucción no es fundamental, aun así es muy útil ya que el LED nos informa cuando la tarjeta es capaz de recibir las señales provenientes de los sensores; de hecho, desde que es encendido, el módulo GSM utiliza hasta casi 1 minuto para poder operar y conectarse a la red celular.

Una vez completados todos los estados, no queda más que entrar en el estado global y aplicar "Encendido_GSM" como estado principal.

De la pantalla principal podremos ver, finalmente, el diagrama de estados completo del programa (mostrado en la **Fig. 6**).

En este punto no queda más que arrancar la compilación y la programación de la DM Board ICS: haciendo clic sobre el pulsador ("Compile code and program board") es posible arrancar DM Programmer (**Fig. 7**).

Después, haciendo clic sobre el pulsador ("Program Board") es posible programar directamente la tarjeta DM Board ICS insertándole el programa que acabamos de escribir.

Para mayores detalles referentes a la programación os remitimos a la guía "Compilare e scaricare un programma sulla DM Board ICS " ("Compilar y descargar un programa sobre la DM Board ICS"), en italiano, descargable de la web *www.dmboard.it* bajo la sección *Tutorial*.

El programa completo de esta aplicación puede ser recuperado y descargado a través de la tienda de DM Design, con el nombre "Allarme GSM".

(193051)

el MATERIAL

La placa está disponible montada al precio de 134,00 Euros (cod. 7302-DMBOARDICS). La tarjeta no incluye los siguientes productos disponibles por separado: modem móvil montado cod. FT900M, 49,00 Euros; antena GSM cod. ANTS-MAGSM, 8,00 Euros; Cable adaptador cod. CVANT-SMA, 8,00 Euros.

Precios IVA incluido sin gastos de envío. Puede hacer su pedido en:
www.nuevaelectronica.com
pedidos@nuevaelectronica.com