Desarrollo e implementación de herramientas de depuración de modelos para sistemas embebidos

M Giura, M. Trujillo, M. Prieto, J. Cruz, L. Francucci, L. Sugezky, N. Gonzalez
Departamento de Ingeniería Electrónica – Facultad Regional Buenos Aires
Universidad Tecnológica Nacional
Buenos Aires, Argentina
mgiura@frba.utn.edu.ar

***Abstract*— uModel Factory es un software didáctico-profesional de modelado y simulación de aplicaciones para sistemas embebidos, desarrollado en la UTN FRBA en el marco de un proyecto de Investigación y Desarrollo del Departamento de Ingeniería Electrónica (SIUTNBA0004734). Permite la creación de un diagrama de estados a través de su interfaz gráfica, la simulación del modelo, como así también la generación automática de código C portable y toda la documentación asociada, manteniendo sincronizado, en tiempo de desarrollo, el modelo, código generado y documentación. En este trabajo se presenta el diseño e implementación de herramientas de software para la depuración del modelo implementado en un microcontrolador ARM Cortex M3 LPC1769.**

Palabras clave—sistemas embebidos; modelo; depuración; Cortex.

I. INTRODUCCIÓN

El uso de modelos se ha vuelto cada más frecuente en el desarrollo de máquinas y equipos que necesiten de algún tipo de automatización, ya que permiten describir claramente el software de los sistemas embebidos, ayudan a comprender el sistema y a diseñar con un nivel de abstracción superior al de los lenguajes de programación [1,2]. Un modelo es una representación simplificada de un sistema que contempla las propiedades importantes del mismo desde un determinado punto de vista. A su vez, permiten lograr un conocimiento más profundo del problema y favorecen el intercambio de ideas entre las personas involucradas en el diseño.

Para el diseño del software de automoción de las máquinas y equipos mencionados anteriormente y en particular dentro de las representaciones gráficas posibles encontramos las máquinas de estados finitos (FSM por Finite State Machine), las cuales constituyen una herramienta que ha sido utilizada tradicionalmente para modelar el comportamiento de sistemas electrónicos e informáticos. Como una amplia extensión al formalismo convencional de estas máquinas surgieron los diagramas de estado (Statecharts) los cuales se convirtieron en parte del estándar UML (Unified Modeling Language) para describir el comportamiento de sistemas en modelos abstractos.

Los diagramas de estado muestran el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación [3]. Para el pasaje de este objeto por los estados del modelo se analiza su respuesta a eventos y se vincula con sus respuestas y acciones.

Estos diagramas normalmente contienen estados, transiciones, eventos, acciones y actividades, a los que en el marco de este trabajo definimos del siguiente modo:

- **Estado:** Identifica los comportamientos estáticos (situaciones en las que el objeto permanecerá hasta que la llegada de un evento específico fuerce su salida) del objeto en análisis.
- **Evento:** es la ocurrencia de un suceso que puede o no causar la transición de un estado a otro del sistema. Por ejemplo, una condición que toma el valor de verdadero o falso (expresión booleana); la recepción de una señal o mensaje externo; o el paso de cierto período de tiempo.
- **Transición:** es el camino (paso) de un estado a otro producida a partir de la ocurrencia de un evento.
- **Acción:** es una operación atómica que ocurre durante una transición, a la que no es posible interrumpir por otro evento y que se ejecuta hasta su finalización.
- **Actividad:** es una tarea que se desarrolla dentro de un estado hasta que ocurra un evento que la obligue a su finalización por el intrínseco cambio de estado.

En este trabajo se presenta el desarrollo e implementación de herramientas de software que permitan la depuración del modelo propuesto a partir de la generación de un lazo cerrado entre el modelo, la generación de código y su funcionamiento.

uModel Factory (uMF) es un software didáctico-profesional de modelado de aplicaciones para sistemas embebidos, desarrollado en la UTN-FRBA (figura 1) en el marco de un proyecto de Investigación y Desarrollo del Departamento de Ingeniería Electrónica (SIUTNBA0004734). Permite la creación de diagramas de estados a través de su interfaz gráfica, la simulación del modelo, como así también la generación automática de código C portable y toda la documentación asociada, manteniendo sincronizado, en tiempo de desarrollo, el modelo, código generado y documentación.

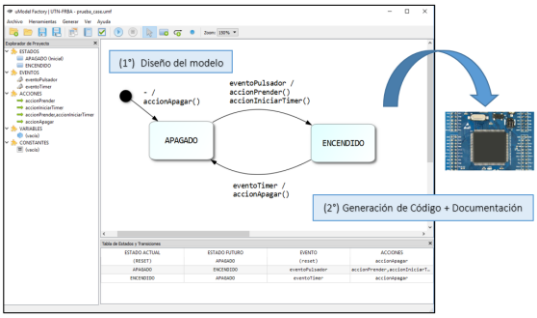

Figura 1. Vista de la interfaz de uModel Factory

La sincronización señalada es unidireccional y se produce a partir del diagrama de estados y transiciones que una vez concluido permite obtener el código C y la documentación pertinente. Si se requieren cambios, y mantener la sincronización, debe modificarse el diagrama para obtener el nuevo código C y su documentación asociada.

Si bien existen otras herramientas que posibilitan la generación de código en diferentes lenguajes a partir de la representación de su modelo [4,5], en su mayoría poseen una curva de aprendizaje pronunciada, licencia paga o no se encuentran orientadas a sistemas embebidos.

Resulta interesante destacar que esta herramienta fue concebida bajo una perspectiva didáctica que se aprecia en el hecho que genera tres implementaciones diferentes del código que representa el modelo creado (if anidados, switch-case y arreglo de punteros a función, en lenguaje C), lo cual favorece la discusión y el análisis dentro del aula [6].

En la actualidad existen diferentes enfoques orientados a la depuración de sistemas embebidos, principalmente pueden clasificarse en intrusivos o no intrusivos. A su vez, podemos evaluar los mismos como de tiempo real o de tiempo diferido [7,8,9].

El enfoque adoptado en el desarrollo aquí presentado es intrusivo y de tiempo real.

II. ALCANCES Y OBJETIVOS

La versión 2016 de uModel Factory ya permitía la creación de diagramas de estados a través de su interfaz gráfica y posteriormente la realización del proceso de simulación, el cual trabaja sobre la validación del modelo realizado [10].

El objetivo principal del desarrollo de la nueva versión de uModel Factory 2018 es incorporar herramientas para la depuración del sistema, habiendo pasado previamente por la simulación.

La incorporación de nuevas herramientas al software de modelado posibilita un enfoque didáctico que permite trabajar sobre todo el proceso de diseño compuesto por la creación del modelo, su simulación y luego su depuración. Este último capaz de ser visualizado gráficamente en el mismo modelo desarrollado.

El plan de desarrollo actual de uModel Factory prevé la incorporación de mecanismos que permitan la generación de eventos y el disparo de condiciones particulares que permitan guiar la ejecución hacia puntos concretos bajo análisis, facilitando así la ejecución del modelo en tiempo real con el objeto de exteriorizar los errores producidos durante su funcionamiento.

Los objetivos específicos sobre los que se trabajó son:

A. Desarrollo del software que permite incorporar la depuración del diagrama de estados.

B. Elaboración de los algoritmos de software necesarios que permitieron:

   1. Evaluar el comportamiento real del modelo a partir de su representación gráfica.

   2. Incorporar elementos gráficos de entrada y salida de información (Pulsadores, Teclados, Leds, Displays, etc.)

En todo momento del desarrollo no se perdió de vista los aspectos didácticos en el diseño de la interfaz de usuario, asegurando que los conceptos principales se pongan en juego explícitamente en cada operación.

Alcance de la versión 2016 de uMF y propuesta de mejora implementada en la versión 2018

Características de la versión 2016

- Generación de un diagrama de estado basado en estados simples
- Generación de código C compatible con el modelo propuesto
- Verificación formal del modelo
- Simulación del modelo mediante eventos
- Nula interacción con el hardware

Si bien el proceso de simulación permitía detectar errores previos al momento de ejecución sobre el hardware (figura 2), aún no era posible monitorear el funcionamiento del sistema embebido.

Mejoras implementadas en la v2018 de uMF

- Se propuso mejorar aspectos de la interfaz de usuario para facilitar el uso de la misma, como ser: tamaño de iconos, declaración y asignación de variables en menor cantidad de pasos.
- Incorporar mecanismos que permitieran rastrear en qué estado de la máquina se encuentra en cada momento (figura 3).

- Considerar aspectos en el diseño del proceso de depuración que permitan escalar el mismo a sistemas más complejos.

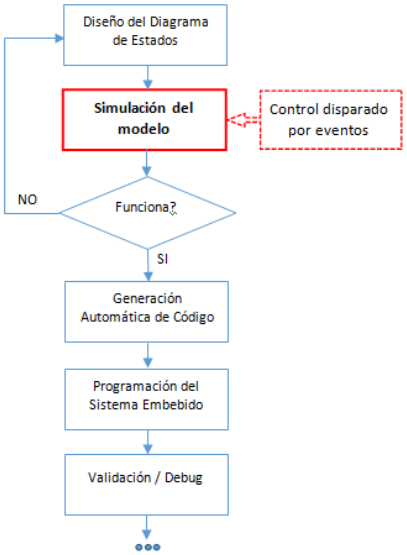

Figura 2. Secuencia desarrollada con la versión 2016

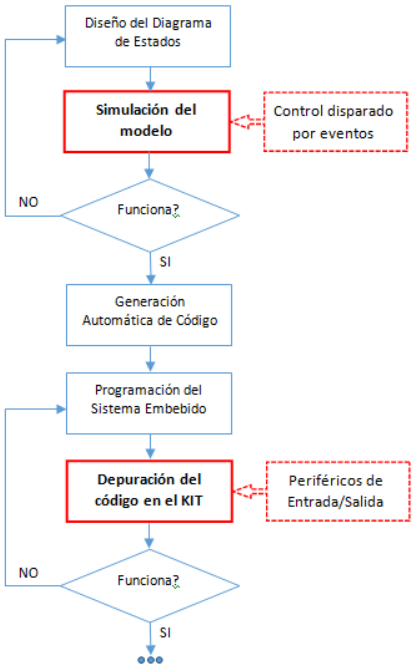

Figura 3. Secuencia propuesta con la versión 2018

III. RESULTADOS

La nueva versión de uModel Factory (v2018) permite la creación del modelo, la generación del código que lo representa, su simulación y posterior depuración. En la figura 4 se observa la interfaz principal del programa donde se encuentra creado el modelo utilizado en el presente trabajo.

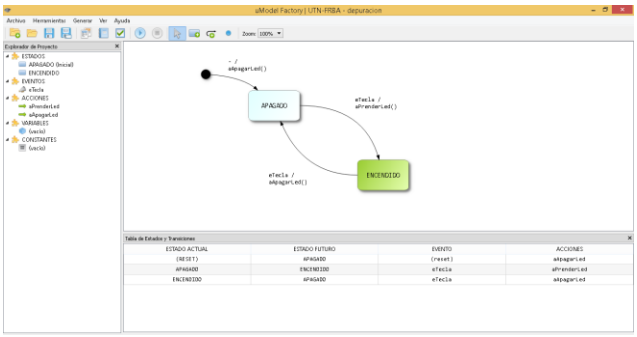

Figura 4. Interfaz de la aplicación

En la figura 5, se observa en detalle la representación de un sistema compuesto por una tecla y un dispositivo lumínico, el cual cambia de estado al presionar la tecla.

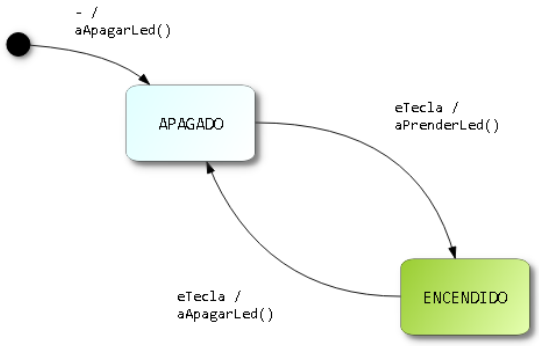

Figura 5. Diagrama de estado diseñado con la aplicación

Estructura general del programa

A continuación, se expone la estructura general de la codificación en C.

```
int main ( ) {
        inicializar ( );     //inicialización de periféricos

        while (1) {
                maquina_estado ( );
        }
        return 0;
}
```

La función "**maquina_estado**", que se observa en la siguiente sección, se ha generado a partir de la representación del diagrama de estados, habiendo optado previamente (en tiempo de desarrollo aún) por su implementación a través de la estructura switch-case. Otras opciones disponibles son la estructura de if-else o la implementación por medio de punteros a función.

Implementación del proceso de depuración

De forma de llevar adelante el proceso de depuración, se incorporó al código una función de registro, la cual permite establecer tres datos fundamentales: de qué máquina de estado se trata (para el caso de que hubiera más de una en juego), el estado del que se parte y el evento que disparó la transición.

```
void maquina_estado ( )
{
static int estado = APAGADO;

switch (estado)
{
        case APAGADO:
            if (eTecla ( ))
            {
                aPrenderLed ( );

                #ifdef DEPURAR
                Log(idMaquina,idEstado,idEvento);
                #endif

                estado = ENCENDIDO;
            }
            break;

        case ENCENDIDO:
             if(eTecla ( ))
            {
                aApagarLed ( );

                #ifdef DEPURAR
                Log(idMaquina,idEstado,idEvento);
                #endif

                estado = APAGADO;
            }
            break;

        default:
            {
                #ifdef DEPURAR
                Log(idMaquina,NO_ASIG,UNK);
                #endif

                estado = APAGADO;
            }
}
```

La función implementada, **"Log"** hace uso de una comunicación serie asincrónica entre el dispositivo embebido y la PC, que permite el envío de tramas que transportan los datos requeridos en el marco de un protocolo de comunicación pre-establecido para que uModel Factory pueda interpretarla y actuar en consecuencia (figura 6). Dicho protocolo se encuentra compuesto por:

- 1 byte de inicio (carácter '$').
- 3 bytes de datos correspondientes a la identificación de la máquina, el código del estado y el del evento que lo disparó. En particular, en el caso por omisión (default) se utilizan dos valores predefinidos (defines) que permiten indicar que se trató de una situación anómala.
- 1 byte de finalización / control (carácter '#')

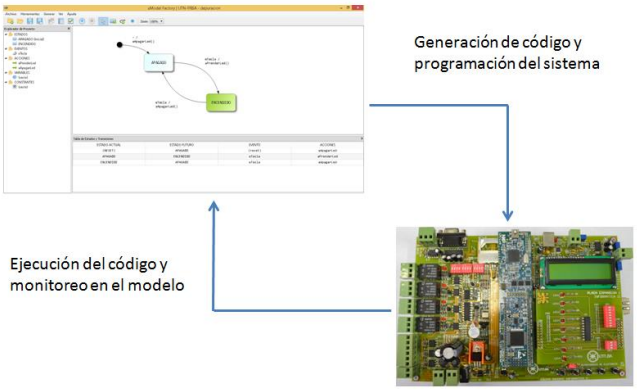

Figura 6. Monitoreo del código ejecutado a través del modelo

Visualización del proceso de depuración

Al iniciar el proceso de depuración se observa en la interfaz gráfica que se ha disparado el evento de RESET, el cual configura a la máquina en su estado inicial (figura 7).

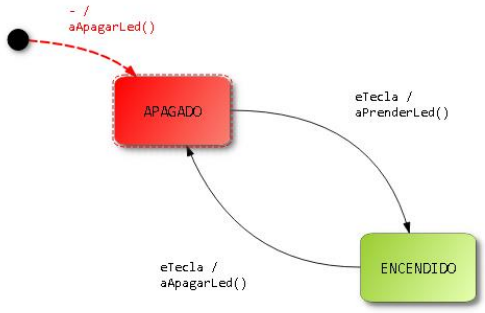

Figura 7. Proceso de depuración tras el evento RESET

Al producirse el evento "eTecla" correspondiente a la presión de la tecla, se recibe en la PC a través de la comunicación serie mencionada una trama que brinda información sobre el evento ocurrido. La trama se analiza, se valida y se extraen los datos que transporta, de modo tal de actuar sobre la interfaz gráfica para que visualice las consecuencias del evento sucedido en el modelo planteado (figura 8).

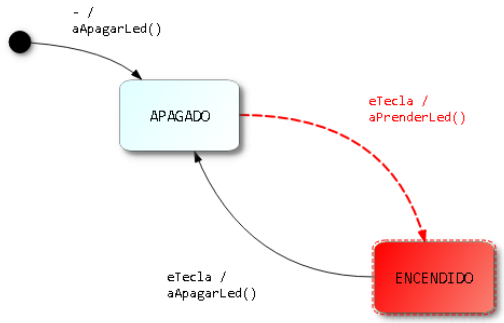

Figura 8. Proceso de depuración tras el evento eTecla (Transición APAGADO-ENCENDIDO)

Al producirse una nueva presión sobre la tecla, se observa una nueva transición desde el estado ENCENDIDO hacia el estado APAGADO (figura 9) repitiendo el procedimiento anteriormente descripto.

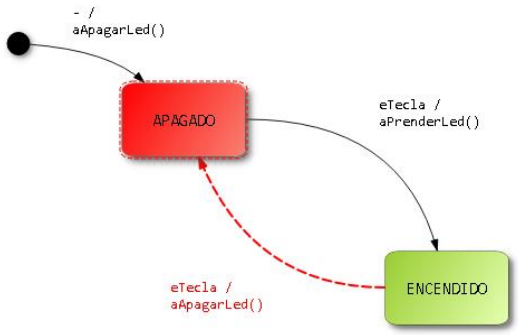

Figura 9. Proceso de depuración tras el evento eTecla (Transición ENCENDIDO- APAGADO)

Caso de estudio

Tomaremos como caso de estudio el de una maquina conformadora de empanadas (figura 10). Dicha máquina se encuentra compuesta por:

- Tolva
- Dosificador
- Molde
- Plato
- Controlador de la máquina

El objeto para nuestro caso será la máquina conformadora de empanadas y la implementación que lograremos a través de uModel Factory es el código de programación necesario para ser ejecutado en un sistema embebido diseñado para tal fin.

El modelo no solo representará al funcionamiento dinámico de la máquina, sino que hará las veces de "puente" ente el desempeño mecánico y el software que lo representa.

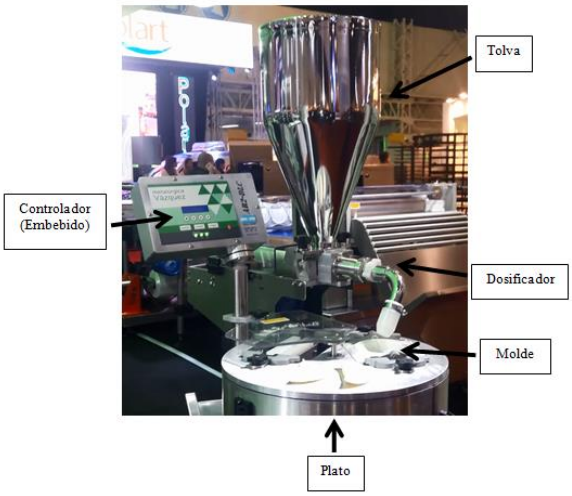

Figura 10. Estructura de la máquina

En nuestro caso los eventos serán:

-Sensor de posicionamiento de plato

-Sensor de fin de dosificación

-Sensor de Alarma por uso indebido del equipo

Con respecto a los estados:

-Avance de Plato

-Dosificar relleno

-Cierre de Molde

-Apertura de Molde

A continuación, se visualiza el modelo que representa a la máquina en análisis (figura 11). De esta forma, en una primera etapa, al momento del diseño del sistema es posible abstraerse de las características mecánicas y evaluar su funcionamiento a través de la verificación formal del modelo y simulación del mismo. Posteriormente, finalizado el firmware y grabado en el microcontrolador, se procede a depurar el mismo utilizando el modelo como centro de monitoreo.

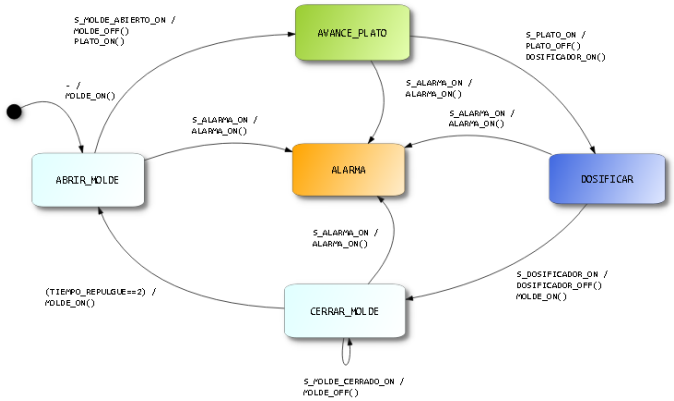

Figura 11. Modelo creado que representa el funcionamiento de la máquina

IV. CONCLUSIONES

En el presente trabajo se diseñaron e implementaron herramientas de software para la depuración de un código C desarrollado a partir de un modelo con uModel Factory sobre un sistema embebido basado en el microcontrolador LPC1769 de la familia ARM Cortex M3.

Para lograr los objetivos propuestos, fue necesario trabajar sobre el código generado a partir del modelo como así también en el procesamiento de la trama serie asincrónica generada por el sistema embebido y recibida en la PC donde corre uMF.

Durante el desarrollo y la codificación, se tuvo en consideración lo importante que resulta que el código incorporado permita que se puedan seguir agregando nuevas prestaciones a futuro y que la aplicación continúe creciendo.

El trabajo realizado generó como resultado una nueva versión de uModel Factory para ser utilizada durante el ciclo lectivo 2018 en el contexto de la materia Informática II perteneciente al Departamento de Ingeniería Electrónica (UTN FRBA).

V. TRABAJOS A FUTURO

Con relación a los próximos trabajos a realizar, a partir de lo presentado en el presente documento, podemos destacar el almacenamiento en secuencia de los eventos de hardware recibidos por medio de la comunicación serie, que permitan realizar una "nueva ejecución" simulada y poder así detectar problemas que a primera vista pasaron desapercibidos, como así también la posibilidad de ejecución del monitoreo en tiempo diferido, en particular atendiendo a aquellos eventos que por su velocidad no pueden apreciarse visualmente en una primera corrida.

A su vez, dentro de las líneas de trabajo que se han evaluado, se contempla la generación de código compatible con el framework RKH (https://www.vortexmakes.com/que-es/), el cual es utilizado en el ámbito industrial. Dicho framework dispone de un módulo interno que permite el registro de la operación de las máquinas con una marca de tiempo de forma de registrar cuando se realizó cada evento. El módulo permite enviar la información mediante comunicación serie o socket como así también su almacenamiento en archivo en función de la plataforma destino utilizada.

REFERENCIAS

[1] G. Booch, J. Rumbaugh, I. Jacobson. *El Lenguaje Unificado de Modelado*. Addison- Wesley 2nd Edition, 2006.

[2] G. ooch, J. Rumbaugh, I. Jacobson. *El Proceso Unificado de Desarrollo de Software*. Addison-Wesley 1st Edition, 2000.

[3] C. Larman. *UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado*. Prentice-Hall, 2003.

[4] Visual Paradigm. Disponible en:

https://www.visual-paradigm.com/features/code-engineering/

(última fecha de acceso: julio de 2018)

[5] Altova UModel. Disponible en: http://www.altova.com/es/umodel/uml-code-generation.html (última fecha de acceso: julio de 2018)

[6] N. González, L. Sugezky, M. Prieto, M. Giura, Y. Kuo, M. Trujillo, J.M. Cruz. "*Evaluación del software uModel Factory como herramienta didáctica*". IEEE Argencon, 2016.

[7] G. Gracioli, S. Fischmeister. "*Tracing Interrupts in Embedded Software*". Journal of Systems Architecture. Volume 58, Issue 9, October 2012, Pages 372–385

[8] J.Campbell, V. Kazantsev, H. O'Keeffe. "*Real-time Trace: A Better Way to Debug Embedded Applications*". White paper. Synopsys, 2014

[9] J. Kraft , A. Wall , H. Kienle. "*Trace Recording forEmbedded Systems: Lessons Learned from Five Industrial Projects*". In: Barringer H. et al.(eds) Runtime Verification. RV

[10] L. Sugezky, M. Prieto, N. González, M. Giura, Y. Kuo, M. Trujillo, J.M. Cruz. "*Desarrollo e implementación de herramientas de simulación de modelos para sistemas embebidos*". Congreso Argentino de Sistemas Embebidos 2016.