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

M. Prieto[1], L.Sugezky[1], M. Trujillo[1], M.Giura[1], N. Gonzalez[1], J. Cruz[1]

[1] Universidad Tecnológica Nacional – Facultad Regional Buenos Aires
Buenos Aires, Argentina
ngonzalez@frba.utn.edu.ar

Resumen. 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 (EIUTNBA0002436). 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 en todo momento sincronismo entre 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.

1 Introducción

El uso de modelos se ha vuelto cada más frecuente ya que permiten describir 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.

Dentro de las representaciones gráficas 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 o de 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. Un evento es una ocurrencia que puede causar la transición de un estado a otro del sistema. Esta ocurrencia se puede deber a distintas condiciones como puede ser: 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. Una acción es una operación atómica, que no se puede interrumpir por un evento y que se ejecuta hasta su finalización.

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 es un software didáctico-profesional de modelado 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 (EIUTN-BA0002436). 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 en todo momento sincronismo entre modelo, código generado y documentación (figura 1).

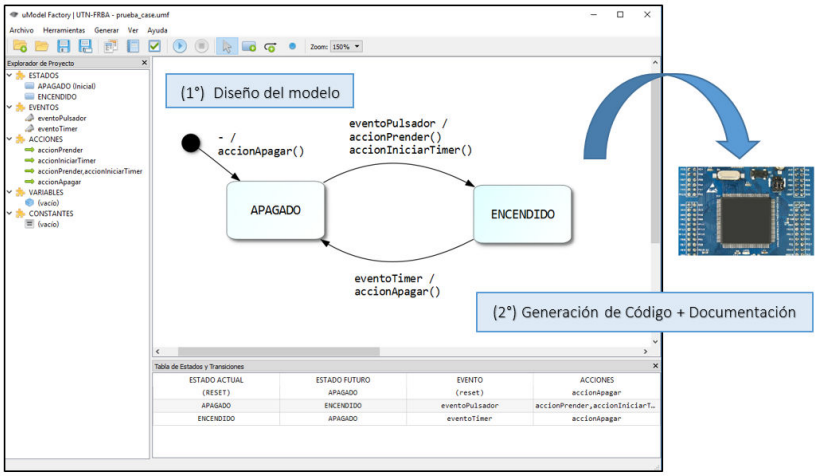

Fig. 1. Vista de la interfaz de uModel Factory

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. Es por ello que esta herramienta posee un enfoque didáctico ya que genera el código en lenguaje C en diferentes implementaciones, 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].

2 Alcance y objetivos

La versión 2016 de uModel Factory permite 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.

El objetivo principal del desarrollo de la nueva versión de uModel Factory 2017 es incorporar herramientas para la depuración del sistema, habiendo pasado previamente por la simulación.

La incorporación de nuevas a 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.

A partir del desarrollo se prevé que se podrán generar eventos o disparar condiciones particulares para guiar la ejecución hacia los puntos a analizar, y ejecutar el modelo en tiempo real para permitir la detección de inconvenientes en el funcionamiento.

Los objetivos específicos son:

A. Desarrollar el software que permita la Simulación y Depuración del diagrama de estados.

B. Elaborar los algoritmos de software necesarios que permitan:

    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.)

C. Fortalecer los aspectos didácticos en el diseño de la interfaz de usuario, asegurando que los conceptos principales se pongan en juego en cada operación.

2.1 Funcionamiento actual y propuesta de mejora

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 permite detectar errores, previo al momento de ejecución sobre el hardware (figura 2), aún no es posible monitorear el funcionamiento del sistema embebido.

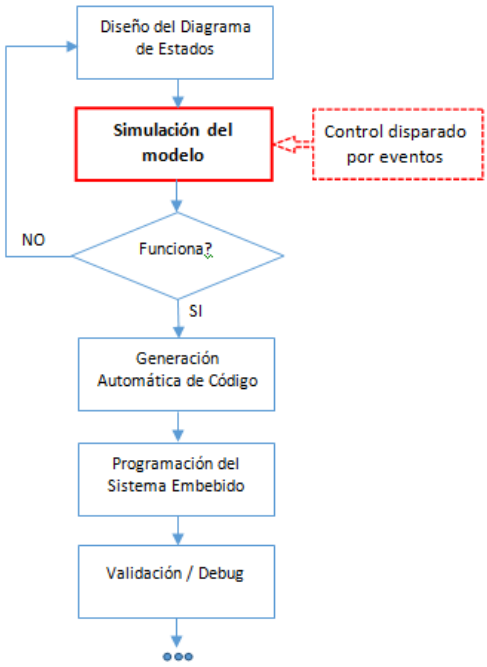

Fig. 2. Secuencia desarrollada con uModel Factory 2016

Propuesta de mejora

- 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.

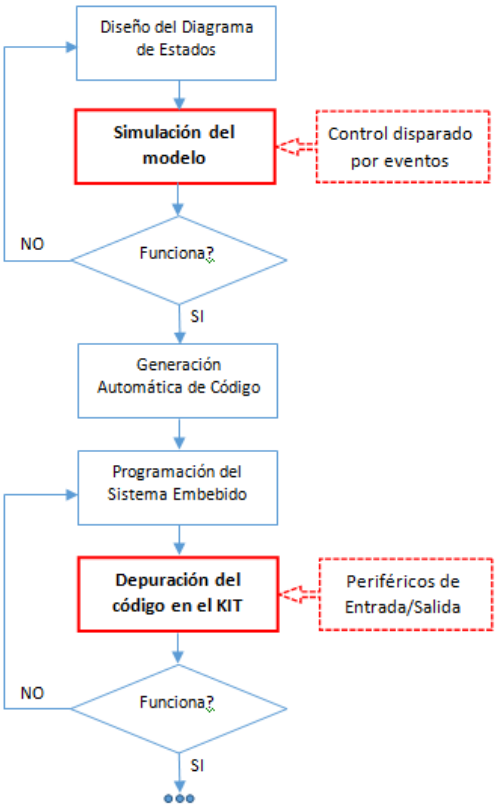

Fig. 3. Secuencia propuesta con uModel Factory 2017

3 Resultados

La nueva versión de uModel Factory (v2017) 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

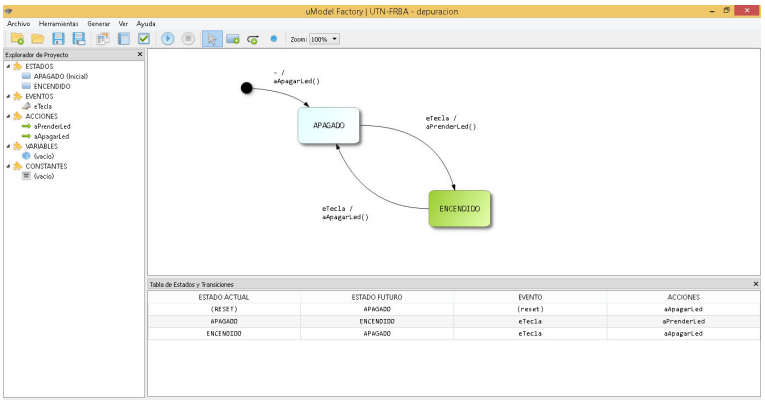

Fig. 4. Interfaz de la aplicación

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

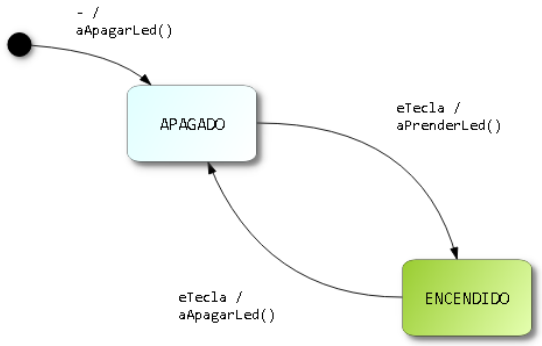

Fig. 5. Diagrama de estado diseñado con la aplicación

3.1 Estructura general del programa

A continuación se expone la estructura general de la codificación en C que se corresponde con el modelo planteado. Dicho módulo se ha generado a partir de la representación del diagrama de estados.

```
void main()
{
    inicializar(); //inicializacion de perifericos

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

3.2 Implementación del proceso de depuración

De forma de llevar adelante el proceso de depuración, se incorpora al código una función de registro, la cual permite establecer de qué máquina de estado se trata, 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_ASIGNADO,DESCON);
                  #endif

                  estado = APAGADO;
            }
  }
```

La función implementada hace uso de la comunicación serie en el dispositivo embebido para enviar una trama hacia la PC de forma de que el software pueda interpretarla y actuar en consecuencia (figura 6).

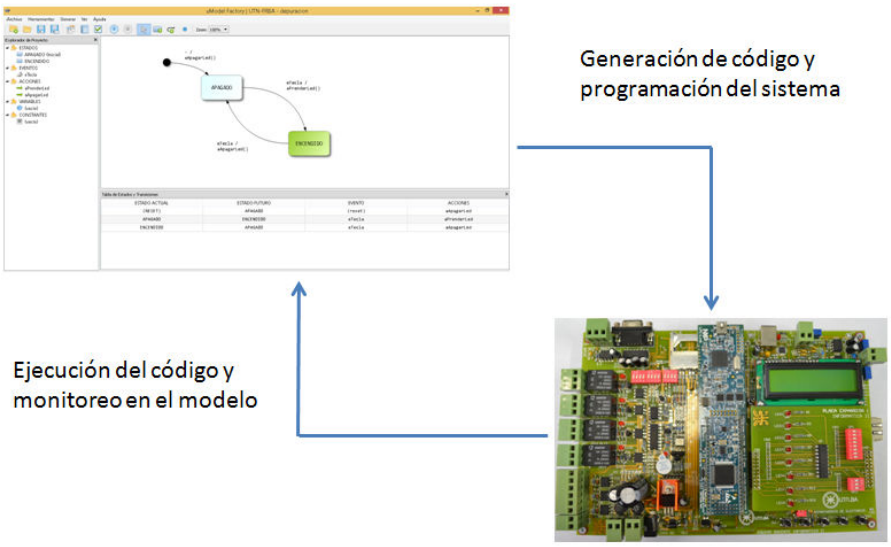

Fig. 6. Monitoreo del código ejecutado a través del modelo

3.3 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).

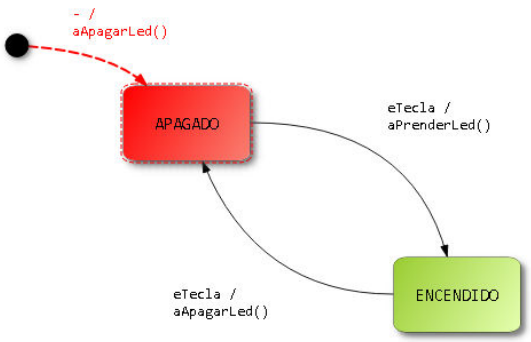

Fig. 7. Proceso de depuración tras el evento RESET

Al producirse el evento “eTecla” correspondiente a la presión de la tecla, se recibe a través de la comunicación serie una trama que brinda información sobre el evento ocurrido. Es así que se descompone la trama, se analiza y se visualiza el resultado en el modelo planteado (figura 8)

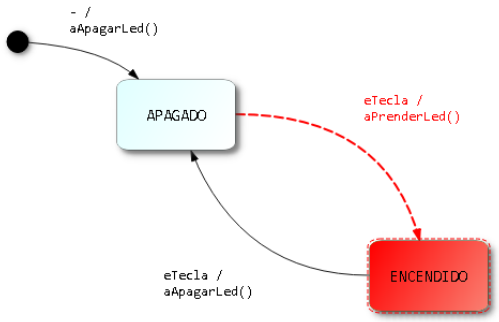

Fig. 8. Proceso de depuración tras el evento eTecla (Transición APAGADO-ENCENDIDO)

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

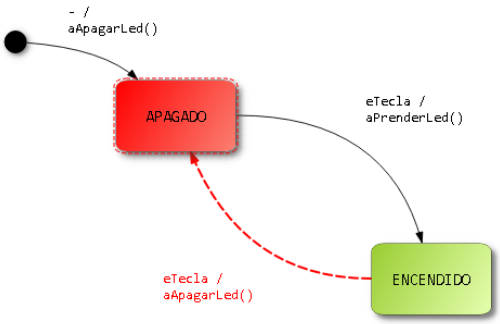

Fig. 9. Proceso de depuración tras el evento eTecla (Transición ENCENDIDO- APAGADO)

4 Conclusiones

En el presente trabajo se diseñaron e implementaron herramientas de software para la depuración del modelo planteado sobre un sistema embebido basado en el micro-controlador 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 recibida en la PC.

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 en el ciclo lectivo 2017 en el contexto de la materia Informática II perteneciente al Departamento de Ingeniería Electrónica (UTN FRBA).

5 Trabajos a futuro

Con relación a los próximos trabajos a realizar, en función de lo indicado en el presente trabajo, podemos destacar el almacenamiento de los eventos recibidos para realizar una "nueva ejecución" y poder 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.

Referencias

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

2. G. Booch, 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. Referencia: https://www.visual-paradigm.com/features/code-engineering/

5. Altova UModel. Referencia: http://www.altova.com/es/umodel/uml-code-generation.html

6. N. González, L. Sugezky, M. Prieto, M. Giura, Y. Kuo, M. Trujillo, J.M. Cruz. "Evaluación del software uModelFactory 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 for Embedded Systems: Lessons Learned from Five Industrial Projects*. In: Barringer H. et al.(eds) Runtime Verification. RV 2010. Lecture Notes in Computer Science, vol 6418. Springer, Berlin, Heidelberg

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.