Caso de estudio: metodología para el aprendizaje, diseño y desarrollo de sistemas embebidos multi-hilos

Ing. Luis Orlando Ventre[1], Dr. Ing. Orlando Micolini[1], Ing. Mauricio Ludemann[1], Agustin Carranza[1], David D´andrea[1], Enzo Candotti[1].

[1] Laboratorio de Arquitectura de Computadoras, FCEFyN-Universidad Nacional de Córdoba
Av. Velez Sarfield 1601, CP-5000, Córdoba, Argentina
{luis.ventre, orlando.micolini, mauri.ludemann}@unc.edu.ar

Resumen. En este trabajo se presenta la aplicación de una metodología para el diseño y desarrollo de un sistema embebido multi-hilos de control de acceso distribuido. Esta metodología permite desacoplar la lógica, la política de resolución de conflictos y las acciones, lo que resulta en un sistema modular, sencillo, mantenible, formal y flexible. Además, se logra la verificación formal de la lógica en las primeras etapas del desarrollo. Para modelar la lógica del sistema, se utilizan redes de Petri y se convierten en código ejecutable mediante la ecuación de estado generalizada. La implementación incluye un monitor de concurrencia que integra los diferentes componentes de software y hardware del sistema. Esta solución permite mantener las propiedades verificadas mediante el uso de formalismos matemáticos. Además, se exponen las ventajas de aplicar la metodología propuesta al diseño de un sistema crítico y reactivo. Se demuestra su capacidad para abordar problemas complejos de manera sencilla y eficaz, garantizando la escalabilidad y la fiabilidad del sistema desarrollado.

Keywords: Metodología, diseño, sistemas embebidos, Redes de Petri

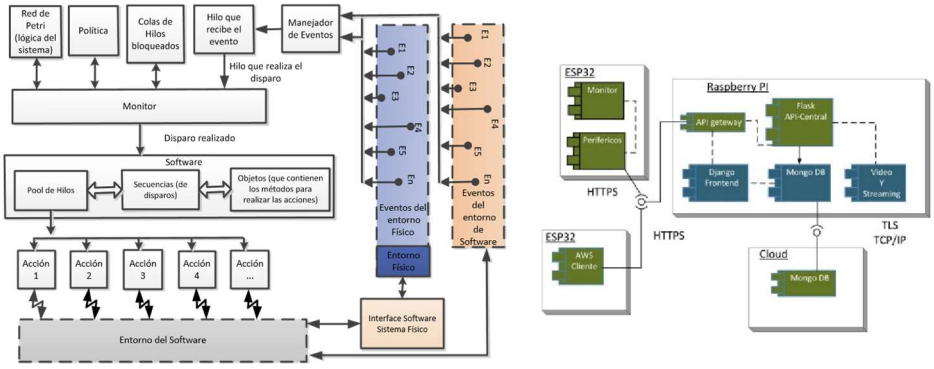
1 Introducción

En el desarrollo de sistemas embebidos se utilizan técnicas de co-diseño hardwaresoftware y lenguajes de programación como C, C++, Java, VHDL y Verilog. Sin embargo, la codificación manual tiene desventajas como la propensión a errores, la limitada mantenibilidad y escalabilidad, la alta volatilidad en los requerimientos y los elevados recursos invertidos para la validación y verificación a través de simulaciones y testing. En sistemas embebidos complejos, esta codificación iterativa por corrección de errores o cambio en requerimientos así como la validación y testing de prototipos, metodología incapaz de garantizar la ausencia de bugs, son todas tareas que impactan significativamente en el tiempo involucrado para la completitud del sistema [1].

El diseño basado en modelos [2] es un método visual y matemático para resolver problemáticas asociadas con sistemas embebidos complejos. Esta metodología engloba otros conceptos importantes como “arquitectura dirigida por modelos” [3]. El diseño

basado en modelos, utiliza los modelos para dar soporte a otras etapas del desarrollo como por ejemplo la simulación, validación, verificación e implementación.

Las redes de Petri (RdP) [4], son un formalismo de modelado que soporta el desarrollo de sistemas basado en modelos. Este formalismo gráfico-matemático soporta explícitamente el modelado de: concurrencia, conflictos, recursos compartidos, exclusión mutua y sincronización. Es importante notar que las RdP tienen claramente definida su semántica de ejecución, y su representación matemática, soportando rigurosa documentación, simulación, verificación y traducción a código de ejecución [5]. Además las RdP son un formalismo matemático abstracto por lo que son intrínsecamente independientes de la plataforma logrando así gran flexibilidad para alcanzar diversos objetivos de performance, costos, consumo de energía u otros.



(a) (b)

Fig. 1 Componentes de la arquitectura de bajo nivel y alto nivel

En este trabajo se expone un caso de estudio y aplicación de una metodología que permite la codificación automática del control de una planta modelada con RdP no autónomas, evitando los inconvenientes de la codificación manual mencionados anteriormente, reduciendo el tiempo de las etapas de simulación y prototipos, garantizando un modelo validado y verificado matemáticamente lo que implica que la especificación documenta la implementación real.

2 Desarrollo

El principal objetivo de la metodología propuesta consiste en descomponer y desacoplar el sistema en: eventos o estímulos, estados, lógica, política y acciones.
El punto de partida es el diseño de un monitor de concurrencia el cual será el componente responsable de la gestión de los eventos en sección crítica con el fin de determinar cuál acción ejecutar y cuando. A continuación se propone establecer la

lógica del sistema a partir de la realización de un modelo del sistema con una RdP, el monitor a través de la RdP determinará las acciones posibles de ejecución.
El sistema requiere de una política para la resolución de los conflictos de la RdP, este componente es utilizado para decidir entre las posibles acciones ejecutables. Además es necesario un módulo para el manejo de los eventos y colas para almacenarlos. Al descomponer el sistema reactivo (RS) o guiado por eventos (EDA) en estos componentes, se busca mejorar la gestión y el rendimiento del sistema embebido logrando simplificar su diseño, desacoplar los componentes y gestionar su control y ejecución en forma centralizada. La arquitectura de bajo nivel resultante de la aplicación de esta metodología puede observarse en la Fig. 1 (a), y puede encontrarse más información respecto de su implementación en [5]. Dada la longitud del artículo se ha omitido la inclusión de la RdP que modela la lógica del sistema de control de acceso.

2.1 Implementación y Selección de Componentes

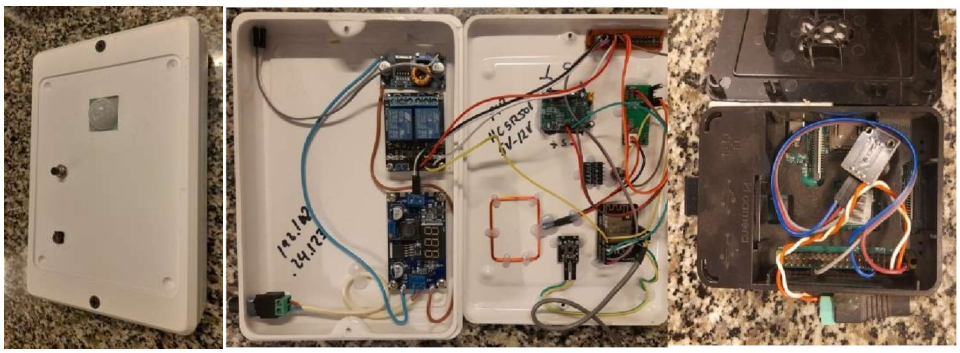
Para el desarrollo del sistema, luego de definir la arquitectura de bajo nivel, se procede a realizar la selección de componentes de hardware. Para la implementación del hardware central se tuvieron en cuenta las siguientes opciones: Raspberry Pi, Orange Pi, Nvidia Jetson, Odroid, BeagleBone y Arduino. Luego de analizar las alternativas y con el criterio de menor costo, simplicidad de uso, soporte, y compatibilidad con los restantes módulos se seleccionó Raspberry Pi.
Para implementar el sistema de punto de acceso (puerta), se consideró la disponibilidad en el Laboratorio de Arq. de Computadoras de la F.C.E.F.y N. U.N.C. de la placa ESP32-CAM, que presenta diversas ventajas como su bajo costo, gran cantidad de puertos de entrada/salida y la disponibilidad de puertos genéricos y de video.
En cuanto a la elección del lenguaje de programación, se optó por Python debido a que es un lenguaje de alto nivel que permite un desarrollo rápido y eficiente. Aunque si bien es cierto que es menos performante que otros lenguajes como Java o Go, en este caso la velocidad no es la prioridad principal. En lo referente a la selección del framework para el desarrollo de la API REST, se evaluaron dos conocidas opciones en el desarrollo de aplicaciones web en Python: Flask y Django. Para esta implementación se decidió utilizar Flask debido a que ofrece mayor flexibilidad y una curva de aprendizaje más sencilla en comparación con Django. Para la selección del IDE/Framework luego de analizar las propiedades de Arduino, Espressif, Mongoose SO, Simba, Pumbaa y nanoFramework se optó por Espressif dado que implementa de forma nativa una versión de FreeRTOS compatible con multicore.
Para el periférico de acceso, se seleccionó RFID 125 KHz debido a su alta disponibilidad, lo que simplifica tanto su adquisición como reemplazo en caso de ser necesario. En cuanto a fuente de alimentación DC-DC se optó por un modelo que implementa un voltímetro en la salida y la entrada con rango de tensión 1.23 – 35 V y 3A con un integrado LM2596. Los restantes periféricos fueron seleccionados de acuerdo a disponibilidad y con criterio de menor costo con el objetivo de obtener un sistema final accesible y replicable. La arquitectura del sistema de control de acceso de distribuido se observa en la Fig. 1 (b), y fue orientada a servicios con protocolo http.

La implementación del sistema para los puntos de acceso (puerta) se realizó en una carcasa genérica en donde se montaron los diferentes componentes. El dispositivo resultante, en su vista exterior puede observarse en la Fig. 2 (a): en la parte superior el sensor de movimiento (HC-SR501), en el centro desplazado a izquierda el pulsador y en la parte inferior el orificio por donde el módulo ESP32-CAM captura las imágenes.

En la Fig. 2 (b) se observa el interior de este dispositivo, conformado por: ficha de alimentación, fuente step down DC-DC LM-2596 con voltímetro, conversor de niveles 5V - 3V3, placa ESP32-CAM con cámara OV2640 2Mp, módulo lector RFID RDM6300 y antena, módulo sensor de movimiento, interruptor, fichas y cables dupont. En la Fig. 2 (c) se puede observar la carcasa para el sistema central implementado con Placa SBC Raspberry PI 3b 1GB RAM 16GB SD y módulo buzzer pasivo.

3 Resultados

El presente trabajo expone como resultado un caso de estudio y aplicación, así como también la validación, de una metodología para diseñar y desarrollar sistemas embebidos, críticos, RS y EDA. De esta forma se obtuvo un sistema de control de acceso distribuido; el cual es capaz de permitir o restringir el acceso a zonas determinada según parámetros de seguridad previamente establecidos. Además el sistema cuenta con funcionalidades de seguridad adicionales tales como: restricciones horarias configurables, sensores de movimiento para la captura de imágenes ante situaciones no esperadas, entre otras. Entre las características del sistema se puede mencionar que es *Flexible* por ser apto para adaptarse a cambios en sus requerimientos minimizando su impacto por ser completamente modular, es validado *Formalmente* debido al fundamento matemático de las RdP el cual avala la lógica del sistema. Es *Mantenible* ya que es modificable efectiva y eficientemente debido a su modularidad, según necesidades evolutivas, correctivas o perfectivas y es *Simple* por permitir visualizar las acciones, independientemente de la lógica y la política que conducen al sistema. Por lo tanto, se logra un código claro y sin responsabilidades solapadas.



(a) (b) (c)

Fig. 2. Vistas externa e interna del dispositivo de acceso y vista interna del dispositivo central.

Referencias

[1] C. Hobbs, *Embedded Soft Dev for Safety-Critical Systems* CRC Press, 2015.
[2] T. Weilkiens, et al. *Model-Based System Architecture*: Wiley & Sons, 2015.
[3] C. J. Roberts, et al, "Preliminary Results from a Model-Driven Arch Methodology for Development of an Event-Driven Space Communications Service Concept," 2017.
[4] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE,* pp. Vol. 77, No. 4, pp. 541-580, 1989.
[5] Eng. Ventre L.O. and Phd. Micolini O. *Extended Petri Net Processor and Threads Quantity Determination Algorithm for Embedded System*. CCIS Springer 2021.