Proyecto 1 - Sistema a la medida para el control y monitoreo de una casa inteligente por medio de servidor web

Julián Camacho Hernández, Kevin Acevedo Rodríguez, Sebastián Calderón Yock
scyock.03@estudiantec.cr, ignaciogramar@gmail.com, kev_sala@estudiantec.cr
CE5303 - Introducción a los sistemas embebidos
Área Académica Ingeniería en Computadores
Tecnológico de Costa Rica

***Abstract*—This paper describes the process of designing and implementing a simple embedded system and it's components. It describes the development of a cross-platform dynamic library for controlling the GPIO interaction with a Raspberry Pi2, as well as a server to work as an application in the embedded system to handle the user interaction and queries made across a web application. It is also explored the process of generating a minimal Linux OS image curated for the project. Additionally the system is tested out in an intelligent house model which uses LEDs and switches to mock the behavior of a real house.**

***Palabras clave*—Embedded systems, IoT, cross-platform development**

I. Introducción

A. Descripción del proyecto

El proyecto en el cual se basa el presente documento trata acerca del tema de los sistemas embebidos en el contexto del internet de las cosas (IoT), específicamente el desarrollo de un sistema a la medida para el control y monitoreo de una casa utilizando una Raspberry Pi2 como plataforma. Esto para el curso de Introducción a los sistemas embebidos de la carrera de Ingeniería en Computadores en el Tecnológico de Costa Rica.

Para este proyecto se creó una biblioteca dinámica en C para la gestión de pines de entradas y salidas (GPIO) de la Raspberry Pi2, un servidor web en Python para interactuar con el sistema, una imagen mínima de Linux por medio de Yocto Project, una interfaz gráfica en React para la interacción con el sistema. Adicionalmente se construyó una maqueta de una casa inteligente para representar el funcionamiento del sistema, por medio de LEDs para simular las luces de la casa, interruptores para las puertas y una cámara web para la cámara de seguridad del patio.

B. Contextualización

En el presente documento se explora el tema del diseño, implementación y aplicación de sistemas embebidos de desarrollo cruzado para sistemas embebidos.

El auge de los sistemas embebidos a la medida, diseñados para espacios inteligentes, ha contribuido significativamente al mejoramiento de la calidad de vida del ser humano. Es aquí donde los conceptos de Internet de las cosas (IoT) y computación ubicua se entrelazan para llevar a cabo aplicaciones embebidas que brinden soluciones específicas, de bajo costo y consumo energético, a problemas reales.

Para este proyecto se debieron aplicar los conceptos de desarrollo cruzado, construcción de sistemas a la medida, tarjetas de evaluación de prototipos e interacción con el hardware en general, para el desarrollo del sistema en cuestión, que implica un manejo adecuado de los

recursos de hardware y software disponibles.

Por todas las razones expuestas y las que más adelante en el documento se analizan, resulta tan importante conocer, comprender y atender este tipo de riesgos, especialmente en el contexto de ingeniería en computadores.

C. Acerca del escrito

Este escrito se compone de una sección de introducción, otra sección de descripción del sistema desarrollado en la cual se detalla el la implementación de los componentes del sistema, una sección de resultados en la cual se muestran y discuten los productos del proyecto, una sección de conclusiones, en donde se muestran los principales hallazgos del proyecto y finalmente, una sección de sugerencias y recomendaciones, donde se brindan algunos consejos que facilitan el desarrollo.

II. Sistema desarrollado

El sistema completo se compone de una serie de componentes desarrollados: biblioteca dinámica, servidor web, aplicación web e imagen Linux. Estos componentes fueron definidos a partir de un documento de especificación y sus requerimientos, tanto funcionales como no funcionales. En la Figura 1 se muestra un diagrama de componentes que ilustra el sistema diseñado.

A. Biblioteca dinámica

La biblioteca dinámica consiste en una colección de funciones en C necesarias para el control de las señales en la plataforma Raspberry Pi 2. Esta biblioteca accede al GPIO de la Raspberry a través del mapeo de su sistema de archivos por medio de descriptores de archivos que correspondan a los

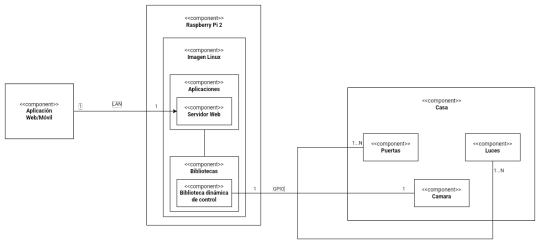

Fig. 1: Diagrama de componentes del sistema

pines definidos. Las funciones principales de esta biblioteca son:

- export_pin(int pin): habilita un pin para para el control de su GPIO como ua interfaz de entrada y salida digital.
- unexport_pin(int pin): revierte el proceso de habilitación de un pin del GPIO como interfaz de entrada y salida digital, con el fin de evitar conflictos y liberar recursos.
- set_pin_mode(int pin, int mode): permite configurar un pin para funcionar en modo lectura o escritura.
- read_digital(int pin): realiza la lectura digital del valor de un pin específico.
- write_digital(int pin, result): realiza la escritura digital de un valor para un pin específico.

Su carácter dinámico se debe al tipo de construcción, en el cual no se vincula la biblioteca directamente con la aplicación durante la compilación, sino en tiempo de ejecución, lo que permite disminuir el consumo de memoria en caso de que sea utilizada por diversos programas. Respecto al desarrollo cruzado, la compilación de esta biblioteca se hizo por medio de una estación de trabajo con las herramientas de GNU Autotools y Meta-toolchain. La biblioteca finalmente era agregada al sistema de archivos del sistema embebido, lo que la hacía utilizable en el sistema por las aplicaciones.

B. *Servidor web*

El servidor web consiste en una colección de funciones en Python para la realización de peticiones, asociadas a las funciones propias de la biblioteca dinámica. Estas funciones están agrupadas en tres partes principales:

- handlerGPIO: este es un wrapper para las funciones de la biblioteca dinámica que está implementada en C. Presenta un mapeo 1:1 de cada una de las funciones que provee la biblioteca dinámica.
- Queries: consisten en las acciones que debe ejecutar el sistema según las peticiones que realice el usuario. Cada función comprende el flujo necesario para cumplir con los requerimientos funcionales del sistema: mapeo y conexión de pines, lectura y escritura de los pines y reportes de errores. Las principales funciones son:
  - initDoors(): realiza el mapeo de las puertas a los pines correspondientes de la Raspberry Pi 2, con base en la configuración definida.
  - initLights(): realiza el mapeo de las luces a los pines correspondientes de la Raspberry Pi 2, con base en la configuración definida.
  - lightOn(): escribe al pin asociado a una cierta luz de la casa, para encenderla.
  - lightOff(): escribe al pin asociado a una cierta luz de la casa, para apagarla.
  - getLightState(): realiza la lectura del pin de una luz, con base en el nombre de la habitación, para determinar si esta se encuentra encendida o apagada.
  - changeAllLightsState(): escribe a los pines asociado de cada luz de la casa, para encenderlas o apagarlas todas.
  - getDoorState(): realiza la lectura del pin de una puerta, con base en el nombre de la habitación, para determinar si esta se encuentra abierta o cerrada.
  - takePhoto(): solicita al sistema operativo que active la cámara y tome una fotografía.
- Routes: corresponde al API para la interacción del servidor por medio de solicitudes de tipo POST y GET. Sus funciones son un mapeo 1:1 de las queries descritas anteriormente.

Este servidor también utiliza el microframework de Pytho3 Flask y Python3 CORS para la gestión de peticiones http. El servidor se ejecuta automáticamente una vez se energice el sistema, lo cual fue configurado de esta forma por medio de la herramienta systemctl de Linux. El servidor se encuentra siempre activo y está programado para poder recibir y realizar las peticiones que el usuario realice por medio de la aplicación web.

C. *Imagen Linux a la medida*

La imagen mínima Linux será generada por medio de Yocto Project, específicamente el branch dunfell. En esta imagen van incluidas tanto la biblioteca dinámica, como el servidor web como aplicación. También incluye las dependencias necesarias para el funcionamiento de ambas, específicamente Python3 flask y Python3 CORS para el funcionamiento del servidor. Esta imagen fue generada por medio de Yocto Project, específicamente el branch dunfell. Para la incorporación de la biblioteca y el servidor se utilizó Meta-toolchain, debido a la naturaleza de desarrollo cruzado. Finalmente generada la imagen mínima e incorporados los demás componentes, se montó la imagen del sistema operativo en una tarjeta SD por medio de Bmaptools y esta se incorporó a la Raspberry Pi2.

D. *Interfaz gráfica*

Se desarrolló una interfaz gráfica para la interacción del usuario con el sistema embebido. Esta interfaz le permite al usuario:

- Encender/apagar una o varias luces
- Conocer el estado de las luces (encendidas/apagadas)
- Comocer el estado de las puertas (abiertas/cerradas)
- Tomar y visualizar una fotografía del patio de la casa

Adicionalmente, tiene un sistema de autenticación para el acceso seguro del usuario propietario. En la Figura 2 y 3 se puede apreciar esto. En la Figura **??** se muestra la visualización general del sistema. Finalmente, en la Figura 5 se ilustra la visualización de la cámara.

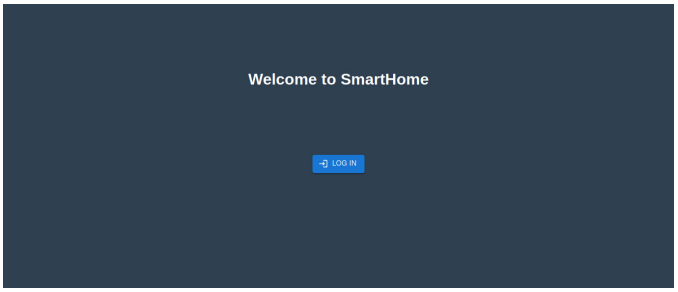

Fig. 2: Landing page de la aplicación

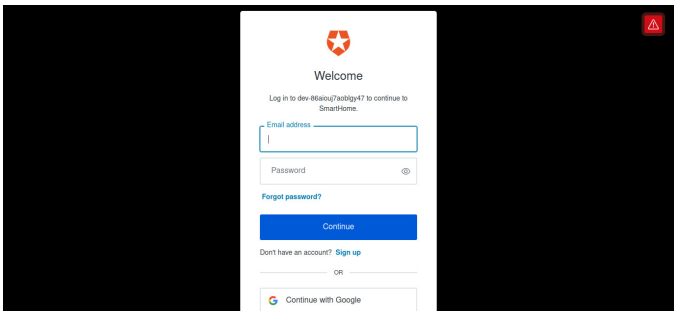

Fig. 3: Autenticación de la aplicación

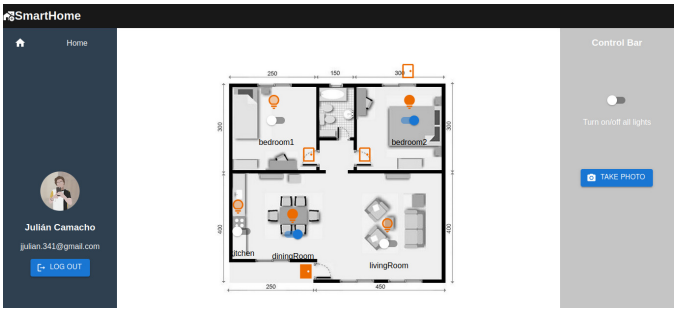

Fig. 4: Vista general de la aplicación

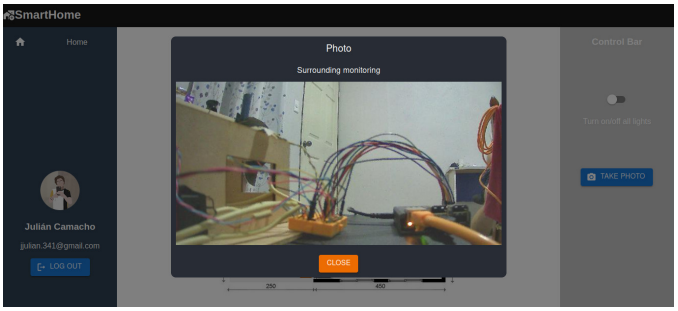

Fig. 5: Vista de la cámara

1) Modelo de la casa inteligente: Finalmente la maqueta de la casa inteligente, la cual consta de 2 habitaciones, sala, comedor y cocina. Hay una luz LED en cada habitación y 4 puertas (una delantera, una trasera y una parabcada habitación). La maqueta se puede apreciar en la Figura 6

III. Resultados

El sistema resultante funciona correctamente y provee todas las funcionalidades descritas. Se inicializa automáticamente una vez este se enciende, permite al usuario autenticarse por medio de sus credenciales e interactuar por medio del sistema embebido con las luces, las puertas y la cámara. El sistema completo se ilustra en la Figura 7

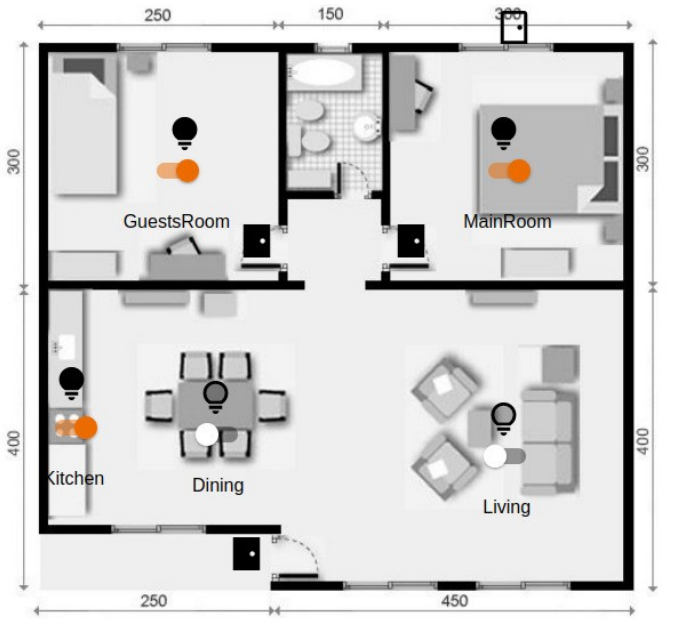

Fig. 6: Modelo de la casa inteligente

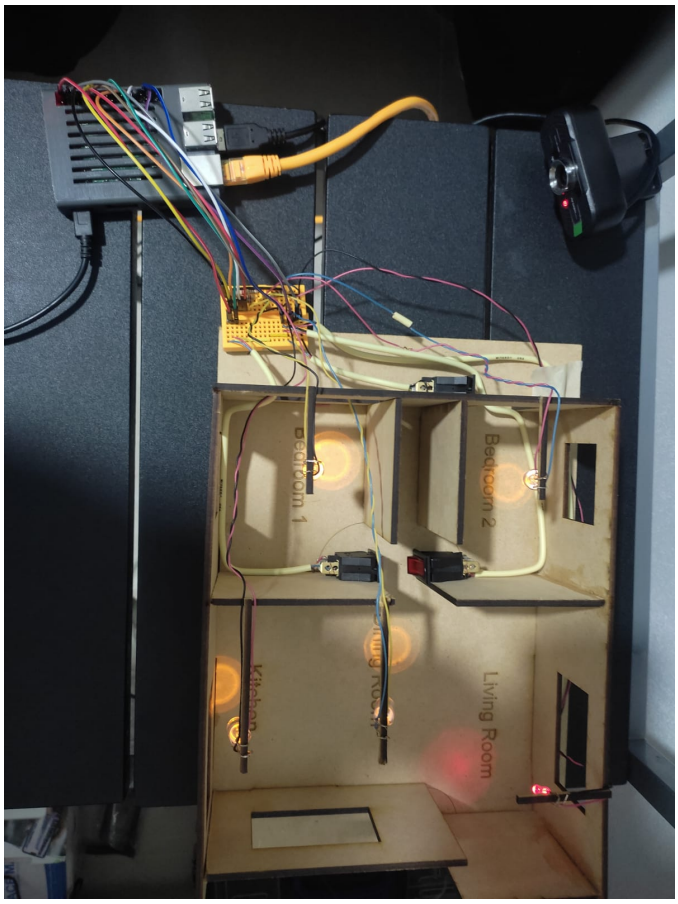

Fig. 7: Maqueta de la casa inteligente

IV. Conclusiones

- Se puede desarrollar sencillamente un sistema de automatización de una casa inteligente por medio de herramientas de software libre como Yocto Project.
- El desarrollo cruzado permite realizar tareas pesadas como compilación y construcción de bibliotecas en una estación de trabajo, descentralizando así mucha esta carga de un sistema de baja potencia como lo son algunos sistemas embebidos.
- Herramientas como Yocto Project permiten generar imágenes a la medida de cualquier sistema y arquitectura.
- El uso de bibliotecas dinámicas permite minimizar el consumo de memoria al enlazar en tiempo de ejecución las bibliotecas con las aplicaciones.

V. Sugerencias y recomendaciones

- Al utilizar la herramienta Yocto Project se debe tener cuidado con la versión o branch que se utilizará, pues esta puede contar con dependencias distintas entre una versión y otra.
- Al trabajar con sistemas embebidos o de baja potencia, se recomienda optar por el desarrollo cruzado pues estos sistemas carecen de la capacidad de construir y compilar programas y aplicaciones, en comparación con una computadora de escritorio o una estación de trabajo.
- Se recomienda la utilización de herramientas como Autotools y Cmake para simplificar la tarea de compilación y construcción de programas, así como resolver dependencias entre ellos.
- Al interactuar con el GPIO de un Raspberry Pi se recomienda acceder a sus pines por medio de archivos y descriptores y no mapear sus pines directamente por simplicidad y facilidad de uso.

References

[1] A. S. Tanenbaum, A. V. R. Elizondo, Ramon Rios Sanchez Jose, and Govea Aaron Jimenez, Sistemas Operativos Modernos, Tercera edición. Mexico: Pearson Educacion, 2009.