Proyecto 2 - Baremetal: Integración de ADC a Empotrado

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 baremetal embedded system based on Raspberry Pi 3 and an Arduino Uno as ADC. It describes the use of Uboot as bootloader for starting up the system and the implementation of a simple ultra sonic sensor for detecting the variation in distance in the environment.**

***Palabras clave*—Embedded systems, Bare-Metal, cross-platform development, Uboot**

I. Introducción

A. Descripción del proyecto

El presente proyecto consiste en el desarrollo de un sistema embebido eficiente, destinado al control y monitoreo de un sensor de proximidad, utilizando la plataforma Raspberry Pi 3, con un enfoque Bare-Metal para maximizar la eficiencia y optimizar los recursos del sistema.

Un sistema Bare-Metal, es decir sin sistema operativo, asegura una ejecución dedicada y eficiente, mientras que el uso de U-Boot como bootloader proporciona una interfaz de arranque flexible. El sistema interactúa con un sensor de proximidad, verificando su capacidad para leer en todo su rango de tensión, y presenta de manera intuitiva los datos en una interfaz desarrollada, que podría ser en Rust, C o C++ (C++ en este caso).

Se empleó un timer para la gestión temporal, en lugar de llamadas del sistema (stalls) y la comunicación serial entre dispositivos, para garantizar un control preciso y una interacción efectiva. Además, el proyecto se ajusta a requisitos no funcionales, como la utilización de la plataforma EVM Raspberry Pi 3 B+, la integración con sistemas empotrados externos con ADC, y la operación a medida con recursos limitados.

La interfaz de usuario, aunque simple y mínima, se diseñó para ser intuitiva y fácil de entender, prescindiendo de la necesidad de un teclado para el inicio del sistema.

En resumen, este proyecto busca lograr un sistema embebido completo, desde el manejo eficiente de hardware hasta una interfaz de usuario amigable, satisfaciendo los requisitos de funcionalidad e integración con un enfoque optimizado para recursos limitados.

B. Contextualización

Los sistemas embebidos Bare-Metal representan una filosofía de diseño en la cual las aplicaciones se ejecutan directamente sobre el hardware sin la presencia de un sistema operativo. Este enfoque otorga a los desarrolladores un control absoluto sobre los recursos de hardware, permitiéndoles gestionar cada aspecto del sistema de forma manual. La simplicidad inherente a los sistemas Bare-Metal se traduce en la ausencia de una capa operativa, brindando eficiencia en términos de tamaño de código y uso de memoria [1].

En este paradigma, la complejidad se reduce al mínimo, ya que no hay un sistema operativo que introduzca abstracciones o servicios preexistentes. Los desarrolladores asumen la responsabilidad total de la gestión de recursos, incluida la manipulación de interrupciones, el acceso a periféricos y la asignación de memoria, partiendo desde cero. Aunque este nivel de control brinda eficiencia, también implica un mayor esfuerzo y experiencia en programación de bajo nivel [1].

La determinación del comportamiento en sistemas Bare-Metal puede ser desafiante, ya que los desarrolladores deben abordar manualmente las restricciones temporales. A pesar de estos desafíos, la elección por Bare-Metal se justifica cuando se busca la máxima eficiencia y control sobre el hardware. Este enfoque es particularmente valioso en situaciones donde la simplicidad y la eficiencia en la ejecución son prioritarias, y los desarrolladores están dispuestos a asumir la responsabilidad total del desarrollo desde la base.

En general, los sistemas embebidos Bare-Metal representan una opción para proyectos donde el control directo y la eficiencia son esenciales, a pesar de la necesidad de asumir mayores responsabilidades en el desarrollo del software [1].

El paper se organizará de la siguiente manera: en la sección II se describirá el diseño propuesto del sistema desarrollado. En la sección III, se presentarán los resultados de la implementación, que incluirán la evidencia del funcionamiento de la aplicación. Finalmente, en la sección IV y V se discutirán las conclusiones y se propondrán posibles direcciones futuras para proyectos similares.

II. Sistema desarrollado

La concepción integral de este sistema se orienta hacia la sincronización fluida de sus diversas partes. Desde la interfaz de lectura del sensor en el Arduino Uno, hasta la presentación clara y eficaz de datos en la pantalla de la Raspberry Pi, cada componente se integra de manera mutuamente armoniosa.

Este diseño, respaldado por un sistema de arranque confiable implementado a través de U-Boot, se destaca por su enfoque modular y planificación meticulosa. Estas características garantizan un rendimiento óptimo y una integración libre de inconvenientes en el contexto específico de la aplicación que se propone. En la Figura 1 se muestra un diagrama de componentes que ilustra el sistema diseñado.

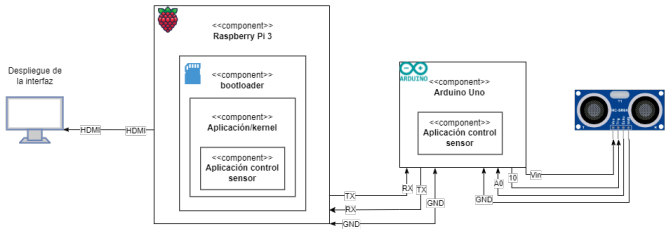

Fig. 1: Diagrama de componentes del sistema

A. Raspberry Pi con Aplicación Bare-Metal

El componente central del sistema reside en una Raspberry Pi 3, que desempeña un papel crucial al ejecutar una aplicación bare-metal desarrollada en C++. La construcción de esta aplicación se lleva a cabo mediante desarrollo cruzado y se incorpora al sistema en forma de un archivo kernel17.img. Para la inicialización del sistema, se utiliza una tarjeta SD que alberga los siguientes archivos:

- start.elf
- config.txt
- bootcode.bin
- kernel17.img

La funcionalidad principal de la aplicación bare-metal se centra en la lectura de una señal UART proveniente de un Arduino Uno, su interpretación y la presentación de la información de manera clara y comprensible en la pantalla. Este enfoque sin sistema operativo proporciona una eficiencia y rendimiento óptimos, especialmente adaptados para la aplicación específica que se busca implementar.

1) Descripción de métodos y bibliotecas :

- Métodos:
  - CKernel::Initialize(): Este método aborda la inicialización completa de las variables de la clase CKernel. Su funcionalidad se extiende a la configuración y puesta en marcha de componentes esenciales, como la pantalla HDMI, la interfaz serie (UART), el sistema de interrupciones y el temporizador del sistema.
  - clearBuffer(void pBuffer, size t nSize): Función auxiliar dedicada a la limpieza de un búfer, recorriendo cada byte y estableciéndolo en 0.
  - my strlen(const char str): Otra función auxiliar que itera sobre una cadena para determinar su longitud.
  - CKernel::Run(): Este método es el corazón de la aplicación. Se ejecuta de forma continua, se encarga de leer datos provenientes del Arduino a través del protocolo UART y, posteriormente, presenta de manera intuitiva la distancia medida desde el sensor de proximidad en la pantalla HDMI. Hace uso del temporizador para introducir retardos y espera antes de realizar nuevas mediciones.
- Bibliotecas:
  - Circle: Proporciona una variedad de herramientas y funcionalidades esenciales para la implementación de la aplicación bare-metal en la Raspberry Pi.
    Entre las capacidades que ofrece, destacan las operaciones relacionadas con cadenas de caracteres mediante circle/string.h, funciones de depuración y registro de información a través de circle/debug.h, y la gestión eficaz de excepciones con circle/exceptionhandler.h.
    Además, circle/screen.h y circle/serial.h facilitan el acceso y control sobre la pantalla HDMI y la interfaz serie (UART), respectivamente. Asimismo, circle/interrupt.h y circle/timer.h desempeñan un papel central en la inicialización y gestión de interrupciones y temporizadores del sistema.
    La utilidad de circle/logger.h radica en simplificar el registro de eventos y mensajes, ofreciendo una herramienta valiosa para el seguimiento y depuración del sistema en su conjunto.
    En resumen, la biblioteca Circle emerge como una caja de herramientas integral, proporcionando las bases sólidas necesarias para la implementación exitosa de la aplicación bare-metal en nuestra plataforma.

B. Arduino Uno como Interfaz de Lectura de Sensor de Proximidad:

El Arduino Uno desempeña un papel crucial en nuestro sistema al actuar como la interfaz encargada de leer la señal proveniente de un sensor de proximidad. Dotado con un convertidor analógico a digital (ADC), el Arduino Uno facilita la lectura de la señal del sensor y su transferencia eficiente a la Raspberry Pi 2 a través de un pin UART.

Esta colaboración estratégica entre el sensor de proximidad y el Arduino Uno permite aprovechar las capacidades particulares de cada dispositivo, estableciendo así una comunicación efectiva.

1) Descripción de métodos y bibliotecas :

- Definiciones de pines y constantes::
  - ECHO PIN y TRIGGER PIN: Define los pines específicos para la entrada del sensor ultrasónico

(ECHO PIN) y la salida del pulso ultrasónico (TRIGGER PIN).
  - SECOND: Representa la duración de un segundo en milisegundos, utilizado para establecer los intervalos de medición.
  - SOUND SPEED: La velocidad del sonido en el aire, esencial para calcular la distancia basada en el tiempo de vuelo del pulso ultrasónico.
- Métodos::
  - setup(): Configura la comunicación serial y los pines del Arduino para el sensor ultrasónico. Inicializa la comunicación serial a 115200 baudios y configura TRIGGER PIN como salida y ECHO PIN como entrada.
  - loop(): Ejecuta repetidamente el proceso de generación de pulsos ultrasónicos, medición de distancia y envío de datos a través de la comunicación serial.
  - generate ultrasonic pulse(): Genera un pulso ultrasónico para activar el sensor de proximidad.
  - measure distance(): Mide la distancia basada en el tiempo de vuelo del eco del pulso ultrasónico.

C. Circuito para Acondicionar un Sensor de Proximidad

Con el objetivo de optimizar la lectura del sensor ultrasónico, se ha diseñado e implementado un circuito de acondicionamiento. Este componente se encarga de preprocesar la señal proveniente del sensor ultrasónico, preparándola de manera óptima para su captura por el Arduino Uno. La información procesada se transfiere posteriormente a la Raspberry Pi mediante la comunicación UART, contribuyendo de manera significativa a una lectura del sensor de proximidad que se distingue por su precisión y confiabilidad

D. Bootloader con U-Boot para la Inicialización del Sistema

Se implementó un bootloader creado con U-Boot, utilizando la configuración rpi 2 defconfig. Este bootloader se ha desarrollado mediante herramientas especializadas de desarrollo cruzado arm-none-eabi-. Su función principal radica en posibilitar el arranque fluido de la aplicación bare-metal en la Raspberry Pi 2. La elección de U-Boot asegura una inicialización del sistema eficiente y confiable, proporcionando la flexibilidad y adaptabilidad necesarias para el entorno específico de la Raspberry Pi.

III. Resultados

El sistema embebido resultante se ilustra en la Figura 2. El sensor ultrasónico, que se observa en la esquina inferior derecha, realiza mediciones de distancia que son leídas por el Arduino Uno. El Arduino Uno remite estas lecturas hacia la Raspberry Pi 3 por medio de comunicación serial.

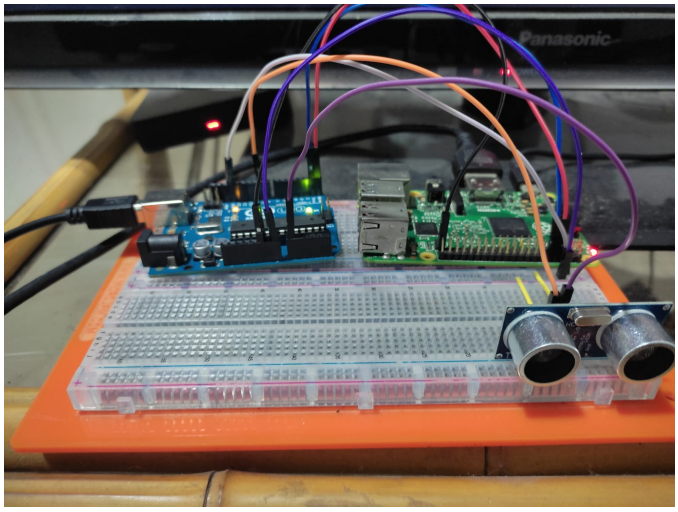

Fig. 2: Maqueta de la casa inteligente

En la figura 3 se observa el resultado de la inicialización del sistema automáticamente, gracias al *bootloader*. Una vez inicializado, la Raspberry Pi despliega en una pantalla los logs de las lecturas realizadas, por medio de una conexión HDMI como se presenta a continuación:

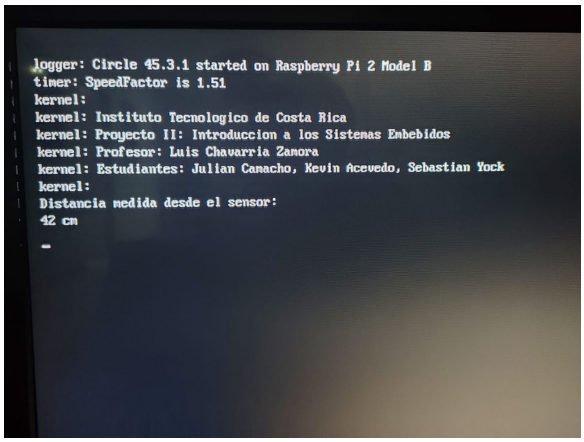

Fig. 3: Interfaz de usuario en la aplicación baremetal.

IV. Conclusiones

- Se puede desarrollar sencillamente un sistema embebido Bare-Metal, exento de requerir un sistema operativo, por medio de un *bootloader* como U-boot.
- 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 es el caso en la mayoría de sistemas embebidos.
- Se puede desarrollar sencillamente un sistema embebido que interface una tarjeta de evaluación relativamente compleja como la Raspberry Pi con una dotada de un ADC como el Arduino Uno, por medio de protocolos de comunicación serial.

V. Sugerencias y recomendaciones

- Se recomienda la aplicación de sistemas Bare-Metal para sistemas embebidos mínimos, para mayor optimizar los recursos y minimizar la complejidad del sistema.
- 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 *bootloaders* probados como U-boot, debido a su amplio soporte y simplicidad.
- Se recomienda utilizar protocolos de comunicación serial de implementación sencilla, como UART para interfazar la Raspberry Pi con el Arduino Uno. Otras alternativas son: SPI y I2C.

References

[1] V. P., "RTOs vs. Bare Metal: Which is right for your embedded system?," LinkedIn, https://www.linkedin.com/pulse/rtos-vs-bare-metal-which-right-your-embedded-system-vijay-panchal (accessed Oct. 11, 2023).