Diseño e Implementación de un Sistema de Control Remoto Basado en Android para un Vehículo con Bleutooth LE y en FPGA

César Rodríguez, José Bernal y Camilo Torres
Facultad de Ingeniería, Universidad Nacional de Colombia
Bogotá, Colombia

***Resumen*—En este documento se presenta el desarrollo completo de un sistema de control remoto para un vehículo a escala. El sistema integra dos subsistemas principales: una aplicación móvil desarrollada en Android utilizando Kotlin y Jetpack Compose, y un firmware implementado en FPGA mediante Verilog. La comunicación entre ambos subsistemas se establece a través de Bluetooth Low Energy (BLE). La aplicación móvil cuenta con múltiples interfaces que incluyen un joystick virtual para operación en tiempo real y paneles dedicados para pruebas y depuración. El firmware se estructura alrededor de una máquina de estados finitos (FSM) que interpreta comandos ASCII para gestionar el control de motores mediante un puente H. Se detallan la arquitectura del hardware, el diseño del firmware en Verilog, el proceso de desarrollo de la aplicación Android y las pruebas de integración del sistema completo.**

I. Introducción

El avance de los sistemas embebidos y el Internet de las Cosas ha generado nuevas oportunidades en el campo de la robótica y el control remoto, especialmente en contextos académicos. Este proyecto aborda el desarrollo de un sistema completo de operación remota para un vehículo a pequeña escala. El objetivo principal consistió en crear una solución modular que separara la lógica de control (firmware) de la interfaz de usuario (aplicación móvil), lo cual facilita el desarrollo independiente, las pruebas y futuras actualizaciones del sistema.

Se optó por utilizar una FPGA para el control a nivel de hardware debido a su flexibilidad y capacidad de procesamiento paralelo. El firmware se implementó en Verilog, mientras que para la interfaz de usuario se emplearon prácticas modernas de desarrollo Android que garantizan una experiencia de usuario fluida y responsiva.

II. Arquitectura del Sistema

El sistema se organiza en tres capas principales: Hardware, Firmware y Software (Aplicación Android).

II-A. Arquitectura del Hardware del Vehículo

Los componentes principales del vehículo son:

- **Chasis:** Plataforma de cuatro ruedas que constituye la base mecánica del vehículo.
- **FPGA (Field-Programmable Gate Array):** Unidad central de procesamiento del vehículo. Se seleccionó una FPGA por su capacidad de procesamiento paralelo de alta velocidad y generación precisa de señales de modulación por ancho de pulso (PWM).
- **Driver de motores (Puente H):** Circuito dedicado para el control de los motores DC. Recibe las señales de control (dirección y PWM) desde la FPGA para manejar la polaridad y velocidad de las ruedas.
- **Módulo Bluetooth (HC-06):** Interfaz de comunicación inalámbrica configurada para operar en modo BLE. Expone un servicio UART que permite recibir datos seriales desde la aplicación Android y transmitirlos a la FPGA.

III. Desarrollo del Firmware en Verilog

El firmware, sintetizado en la FPGA, se encarga de interpretar los comandos recibidos y controlar el hardware del vehículo. Está compuesto por dos módulos principales implementados en Verilog.

III-A. Módulo Receptor UART

Se implementó un receptor UART estándar para gestionar los datos seriales asíncronos provenientes del módulo HC-06. Este módulo detecta el bit de inicio, muestrerea los 8 bits de datos a la tasa de baudios configurada y verifica el bit de parada. Al recibir un byte válido, transmite el carácter ASCII de 8 bits a la lógica de control principal.

III-B. Lógica de Control y Máquina de Estados Finitos

El núcleo del firmware consiste en una máquina de estados finitos (FSM) que decodifica los comandos ASCII recibidos. Se utiliza un bloque `always @(posedge clk)` con una estructura `case` para determinar la acción correspondiente según el carácter recibido. Esta FSM presenta robustez ante comandos inválidos, los cuales son simplemente ignorados si no están definidos en el protocolo.

A continuación se muestra un fragmento conceptual del decodificador en Verilog:

Listing 1. Fragmento conceptual de la FSM para decodificación de comandos

```
always @(posedge clk or posedge reset)
begin
    if (reset)
    begin
        // Reiniciar senales de control de motores
    end
    else if (new_byte_received)
    begin
```

```
        case (received_byte) // Entrada ASCII de 8
            bits
            "A": // Adelante
                set_motor_direction(FORWARD);
            "B": // Atras
                set_motor_direction(BACKWARD);
            "C": // Detenerse
                set_motor_speed(SPEED_NONE);
            "D": // Derecha
                set_motor_direction(RIGHT);
            "E": // Izquierda
                set_motor_direction(LEFT);

            // Controles de velocidad
            "J": set_motor_speed(SPEED_LOW);
            "K": set_motor_speed(SPEED_MEDIUM);
            "L": set_motor_speed(SPEED_HIGH);

            // Movimientos diagonales (F, G, H, I)
            // ... casos adicionales ...

            default: // Ignorar comandos no
                definidos
                ;
        endcase
    end
end
```

Este diseño garantiza que cada comando válido actualice los registros de control de motores, los cuales determinan las señales PWM generadas para el puente H.

IV. Desarrollo de la Aplicación Android

La interfaz de usuario se desarrolló completamente con Jetpack Compose y se estructura en torno a una arquitectura de múltiples pantallas.

IV-A. Capa de Comunicación BLE

La aplicación utiliza el protocolo Bluetooth GATT. Se optimizó la función `sendData` para control en tiempo real configurando el `writeType` de la característica como `WRITE_TYPE_NO_RESPONSE`. Esto permite enviar un alto volumen de comandos desde el joystick sin esperar confirmación por cada paquete, reduciendo significativamente la latencia. Para mitigar la posible pérdida de paquetes, cada acción del usuario genera el envío del comando correspondiente cuatro veces en rápida sucesión.

IV-B. Estructura de la Aplicación e Interfaz de Usuario

Se implementó un `ModalNavigationDrawer` que proporciona acceso a tres pantallas principales:

- **Depurador:** Vista tipo terminal para monitorear todos los comandos enviados y el estado de la conexión.
- **Control del Carro:** Interfaz principal de control en tiempo real que incluye un joystick virtual personalizado y un control deslizante vertical para la velocidad.
- **Controles de Programación:** Panel con botones discretos basados en iconos para probar individualmente cada movimiento, diagonal y comando de velocidad. Esta pantalla resultó fundamental durante la fase de integración.

V. Integración y Pruebas

El diseño modular permitió realizar pruebas independientes de cada subsistema:

1. **Pruebas del Firmware:** El código Verilog se simuló inicialmente para verificar la lógica de la FSM. Posteriormente a la síntesis, se probaron la FPGA y el puente H enviando comandos ASCII desde una terminal serial de PC.
2. **Pruebas de la Aplicación:** La interfaz de usuario y la navegación de la aplicación Android se probaron de forma independiente al hardware. La pantalla de depuración fue crucial para verificar que se generara la secuencia ASCII correcta para cada entrada del usuario.
3. **Integración del Sistema:** La pantalla de controles de programación se utilizó como herramienta principal para las pruebas iniciales de extremo a extremo. Permitió enviar comandos específicos y controlados para verificar que cada función del hardware (avanzar, retroceder, ajustar velocidad) respondiera según lo esperado. Una vez verificados todos los comandos individuales, se procedió a probar la interfaz del joystick para control en tiempo real.

VI. Conclusiones y Trabajo Futuro

Este proyecto demuestra el desarrollo exitoso de un sistema completo de control remoto, desde el firmware a nivel de hardware hasta una interfaz móvil moderna. La separación de responsabilidades entre la FSM en Verilog implementada en la FPGA y la interfaz de usuario en Jetpack Compose proporciona una arquitectura robusta y escalable.

Como trabajo futuro se plantea expandir el protocolo de comunicación para hacerlo bidireccional. El vehículo podría enviar datos de sensores (voltaje de batería, detección de obstáculos mediante sensores ultrasónicos) de vuelta a la aplicación, la cual podría mostrar esta información en la interfaz, transformando la aplicación de un simple control remoto a un panel completo de diagnóstico y monitoreo.