Proyecto Redes

Interfaz de Red

Ciencias de la Computación VIII

Resumen

El proyecto consiste en crear protocolos en la capa de Medio y Host utilizando un microcontrolador compatible con I2C y UART. Se debe de implementar en C + +, Micro Python u otro lenguaje compatible con su tarjeta de desarrollo, y se espera que implementen la comunicación desde la capa física hasta la capa de aplicación. La implementación incluye el control del dispositivo mediante un driver y la creación de una interfaz de usuario. Además, los estudiantes deben investigar y utilizar un Protocolo del Virtual Devices dato como implementación para consolidar la comunicación.

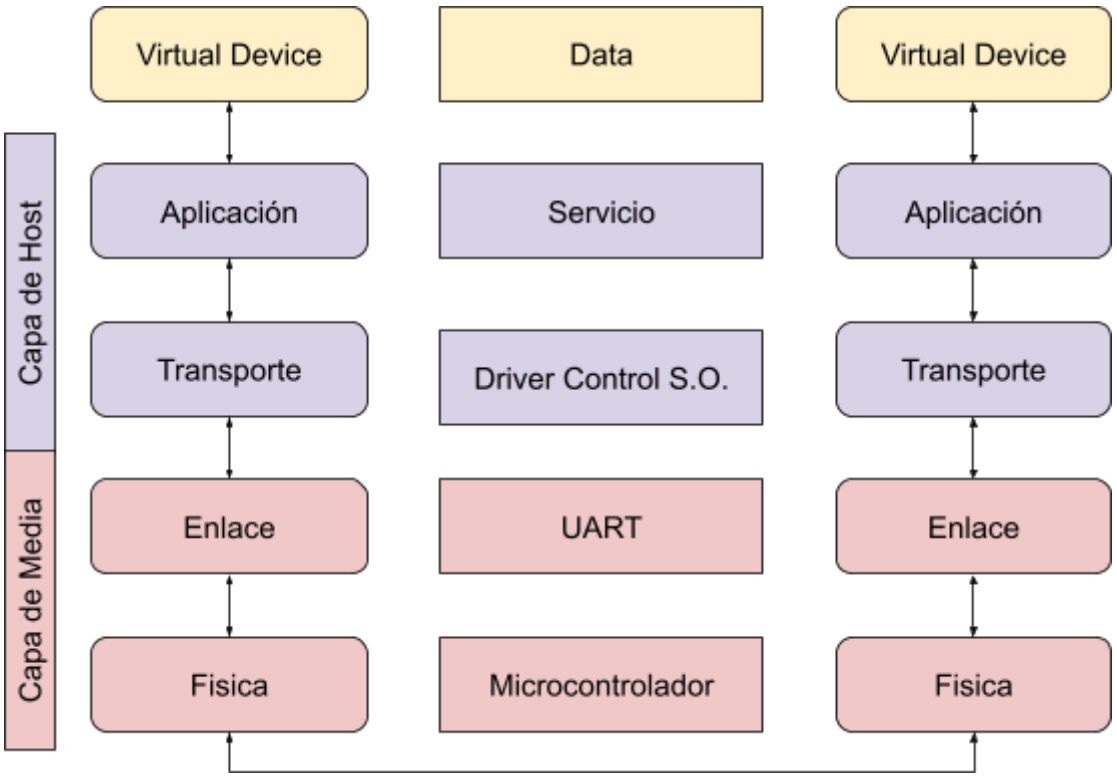
Introducción

En este proyecto, trabajaremos en la creación de protocolos en la capa de enlace de datos utilizando un microcontrolador que soporte conexiones múltiples o físicas mediante I2C. Esta capacidad es esencial para permitir la comunicación entre los proyectos que se desarrollarán en el entorno de red de la clase de manera física. (Debe de presentarse físicamente para tener opción a la calificación completa)

En el microcontrolador, deberán realizar la programación en C++, MicroPython, o cualquier otro lenguaje compatible. Su tarea será implementar la conversión de mensajes desde la capa física, formando los paquetes que serán interpretados por la siguiente capa. La comunicación con la computadora se establecerá mediante una conexión USB, utilizando como protocolo UART, la cual se encargará de transmitir los datos hacia el siguiente nivel.

El siguiente componente de su proyecto es la implementación de un driver de control del dispositivo, que gestionará el direccionamiento del protocolo y permitirá la conexión de múltiples aplicaciones. Este driver puede ser desarrollado en Java o Python, ambos lenguajes ofrecen librerias para UART que facilitarán la comunicación con el microcontrolador. A través de esta capa, podrán controlar qué dispositivos y aplicaciones están interactuando.

Finalmente, trabajarán en la capa de aplicación, que deberá incluir una interfaz de registro (log) y cualquier otra funcionalidad que consideren útil para gestionar el Dispositivo Virtual, es un ejecutable JAR. Es crucial que comprendan y utilicen el Protocolo del Dispositivo Virtual. Aunque la documentación de la Manufactura en China está disponible realmente es mínima, se espera que realicen investigación adicional para asegurar su funcionalidad completa.



Implementación

Los integrantes del grupo deben dividir sus esfuerzos y asignar tiempo a cada parte del proyecto de manera que puedan avanzar en cada aspecto. La clave para unificar su trabajo será la creación de protocolos, los cuales deben estar altamente detallados y documentados internamente en el grupo, estas implementaciones no deben de ser compartidas con otros grupos.

Uno de los aspectos más importantes es el Protocolo de Comunicación entre Microcontroladores. **Toda la clase debe llegar a un acuerdo sobre cómo se comunicarán los microcontroladores entre sí.** Este protocolo debe ser estándar para todos los proyectos y servirá como la base que garantizará una comunicación uniforme y coherente entre los dispositivos de cada grupo.

Opciones de Comunicación

Existen diferentes opciones para implementar la conexión física que facilite la comunicación entre microcontroladores. Pueden optar por una conexión personalizada o utilizar protocolos establecidos. Dos de las opciones recomendadas son SPI e I2C, (el protocolo UART se utilizará únicamente para la comunicación hacia la computadora a través de USB):

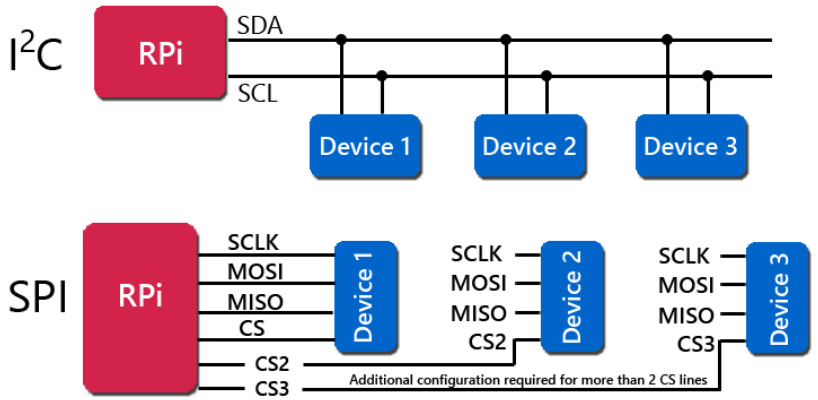
1. **SPI (Serial Peripheral Interface):** Es un protocolo ampliamente utilizado para la comunicación entre múltiples dispositivos. En SPI, el dispositivo maestro controla el reloj y selecciona a los dispositivos esclavos mediante pines dedicados (CS/SS). Para la comunicación bidireccional con múltiples dispositivos, cada uno necesita su propia línea de selección de chip, o bien, se puede usar un bus común si todos los dispositivos deben recibir el mismo mensaje.

2. **I2C (Inter-Integrated Circuit):** Este protocolo permite la comunicación entre múltiples dispositivos en un bus compartido, donde cada uno tiene una dirección única. Un dispositivo actúa como maestro y los demás como esclavos, aunque es posible tener múltiples maestros si se coordina correctamente la comunicación.

Tanto SPI como I2C soportan el broadcasting (envío de mensajes a múltiples dispositivos), pero presentan desafíos de implementación. Sin embargo, I2C es más sencillo en términos de cableado, ya que utiliza menos líneas de conexión.

Recomendación

Es importante que analicen las ventajas y desventajas a nivel de clase de cada protocolo antes de decidir cuál implementar, o si se deciden por uno personalizado. Mientras que SPI ofrece un control más preciso sobre los dispositivos esclavos, I2C facilita la implementación debido a su menor cantidad de cables necesarios. Ambos protocolos conllevan retos técnicos, por lo que deben estar preparados para documentar cuidadosamente la implementación y resolver cualquier conflicto que surja en la integración con otros dispositivos de la clase. Recuerde que todos los dispositivos deben ser capaces de configurarse para determinar su modo de operación o si optarán por un modelo híbrido.



Programación en el Microcontrolador

En el microcontrolador pueden optar por C++, MicroPython u otro lenguaje soportado por el microcontrolador, los microcontroladores recomendados son: Raspberry Pi Pico, Arduino Uno o Nano, ESP32, NUCLEO u otro que tenga soporte a lo que llegarán a convenio en la clase.

El objetivo es implementar la conversión de mensajes desde la capa física hacia la capa de enlace. Esto implica la creación de paquetes que serán enviados a través de la red conformada en clase.

La comunicación se logra por el protocolo elegido para conectar varios dispositivos dentro de la red. Esta conexión es fundamental para la transferencia de datos entre los proyectos y el descarte de información innecesaria a través de los venios estipulados en el protocolo de comunicación que llegaron a nivel de clase.

NOTA: Todos los grupos deben de seguir el protocolo que realizaron a nivel de clase y que a nivel de grupos estipularon, no se calificarán proyectos que implementen protocolos en solitario o tomaron otro rumbo o sean la minoría por diferencias creativas. A nivel del Microcontrolador.

Interfaz de Comunicación UART

La comunicación entre el microcontrolador y la computadora se realizará a través de USB utilizando UART.

Configurar y gestionar la transmisión de datos desde el microcontrolador hacia la computadora, donde se conectará al driver de control del dispositivo. Esto ya dependerá del grupo por el dispositivo que eligieron para implementar, la elección del lenguaje de programación a utilizar sobre el dispositivo C++, MicroPython, Javascript etc, y la forma de lectura y escritura que tiene el SDK hacia el protocolo Serial UART.

Driver de Control

Esta parte se puede desarrollar en Java, Python u otro lenguaje de programación que se le acomode para implementar y que tenga soporte de gestión de puertos seriales.

Este driver gestionará el direccionamiento de paquetes que vienen desde el microcontrolador y hacia qué instancia servicio es requerido, permitiendo que múltiples aplicaciones interactúen con el dispositivo. Será responsable de identificar qué dispositivo y qué aplicación están comunicándose, garantizando un flujo de datos correcto, todo esto debe estar definido en la parte del protocolo que

utilizarán en esa sección a nivel de grupo. Utilizarán la biblioteca UART para facilitar la comunicación con el microcontrolador, asegurando la transmisión y recepción de datos, pero será responsable del control de congestión entre las instancias de la parte de aplicación que soportará el protocolo del virtual device.

Capa de Aplicación

Desarrollarán una interfaz/consola de registro (log) y otras funcionalidades según las necesidades del proyecto y que el grupo necesite para facilitar el control.

Recuerde que esta parte es la encargada de gestionar el Dispositivo Virtual que se entregará en formato JAR como ejecutable. La capa de Aplicación debe de soportar múltiples conexiones o ejecutar múltiples veces, esto dependerá de la implementación que va a realizar el grupo en la parte del protocolo que se estará implementando, recuerde que es responsable de gestionar todas las instancias de los dispositivos virtuales que se estén activas.

Protocolo del Dispositivo Virtual

El objetivo principal es comprender y utilizar el Protocolo del Dispositivo Virtual, unificando la comunicación de su Aplicación. Dado que la documentación disponible es limitada, producto chino con instrucciones en chino, será necesario que investiguen y experimenten para lograr una implementación funcional. La implementación está asociada a varios dispositivos virtuales, cada uno controlado a través de una estructura específica a nivel de bits. Este protocolo permite activar, desactivar o configurar los dispositivos. Es importante tener en cuenta que, aunque no está detallado en la documentación del fabricante, los campos de cada dispositivo podrían funcionar en ambas direcciones.

La documentación establece que el tamaño de los paquetes debe ser de 88 bytes como máximo y 56 bytes como mínimo. Si el mensaje (Msg) no alcanza el tamaño mínimo, es necesario rellenarlo con algún tipo de dato adicional para asegurar el correcto funcionamiento del protocolo. Sin embargo, no se especifica claramente si esta regla se aplica tanto para la transmisión como para la recepción de datos en el protocolo.

El protocolo define que, si el valor del campo de velocidad del ventilador es cero, el ventilador funcionará a baja velocidad. Un valor de quince detendrá por completo el ventilador. Aunque no se especifica sus valores intermedios.

La solicitudes TCP/IP que establece el Dispositivo Virtual es a través del **método POST**, con un **content-type text/plain** en formato **hexadecimal** (HEX). El fabricante sugiere que las respuestas sigan un formato similar, aunque no parecen existir limitaciones claras en cuanto a la estructura de las respuestas. Un joven que hizo prácticas profesionales de su carrera técnica de bachillerato nos mencionó que a veces presenta un error a nivel de protocolo de HTTP de Origen, pero no logró determinar cómo solucionarlo por su poca experiencia, el comando que utilizó para ejecutarlo es:

java -cp *.jar CCVIII.IoT.VirtualDevice 1850

Y se determinó que el argumento final es para cambiar el el puerto en el que se inicia el dispositivo.

Un ejemplo de solicitud podría ser el siguiente: **00F02233440AABBCC0556677.** Este formato representa un conjunto de datos que se envían al dispositivo para activar o configurar ciertas funciones.

Nos dejan una imagen de como se ve el dispositivo virtual.

Tabla de Formato de bits

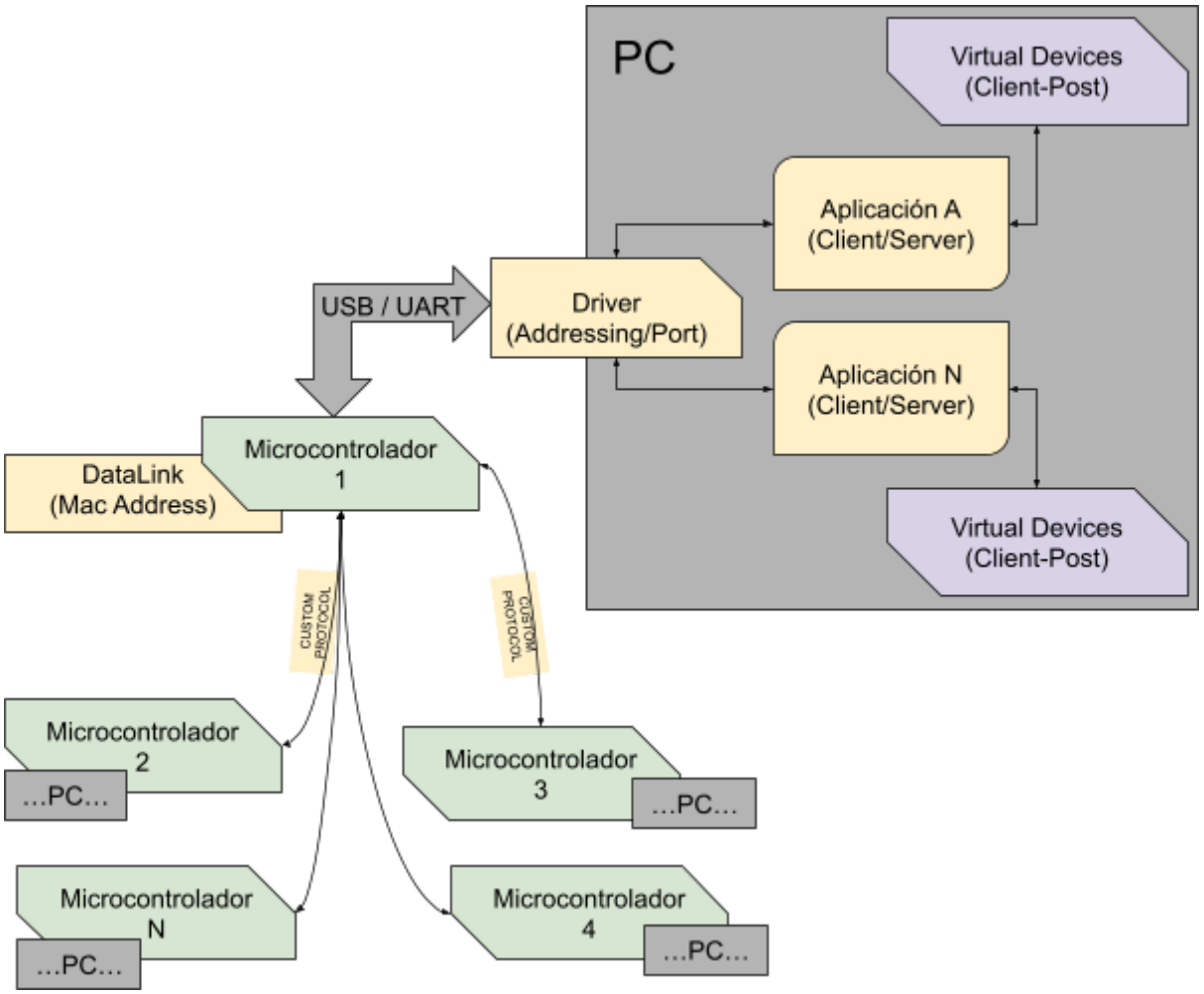
La documentación en chino del fabricante es limitada y lo único útil del manual es la siguiente tabla.

Función	#Bits	Máximo	HEX	Mínimo	HEX
LCD	1b	1	F	0	0
SW0	1b	1		0	
SW1	1b	1		0	
FAN	1b	1		0	
LEDRGB	1b	1	F	0	0
LEDRED	1b	1		0	
LEDGRN	1b	1		0	
HEAT	1b	1		0	
SPEED	4b	1111	F	0000	0
Space	4b	0000	0	0000	0
SLIDER0	8b	11111111	FF	00000000	00
SLIDER1	8b	11111111	FF	00000000	00
SLIDER2	8b	11111111	FF	00000000	00
Space	4b	0000	0	0000	0
LED RGB	8b	11111111	FF	00000000	00
	8b	11111111	FF	00000000	00
	8b	11111111	FF	00000000	00
Space	4b	0000	0	0000	0
PickColor	8b	11111111	FF	00000000	00
	8b	11111111	FF	00000000	00
	8b	11111111	FF	00000000	00
Msg	256b	32 Bytes	32 chars	16 Bytes	16 chars

Tip:
Antes de comenzar a programar es necesario realizar debugging en el protocolo para comprender el funcionamiento de los bits de ida y vuelta.

Diagrama de implementación

Los cuadros Amarillos son las implementaciónes que debe de realizar, lo que esta entre parentesis es una idea de lo que deberia de existir en esa parte de la implementación. El cuadro amarillo de Aplicación (A,B,...,N) piense que estuviera implementando un browser donde puede abrir multiples tabs y estos generan un puerto diferente para que se conecte el Virtual Devices (esto dependera de su implementación). La conexión de los Microcontroladores el Custom Protocol deberá soportar al menos 2 proyectos conectados, siendo 3 el total de dispositivos.



Este proyecto les permitirá profundizar en el diseño y la implementación de protocolos de comunicación, así como en la integración de múltiples capas dentro de un sistema complejo.

¡Espero ver su creatividad y habilidades técnicas reflejadas en sus trabajos!

Entrega

- Suba un archivo ZIP al GES con todos los archivos:
  - Documentos PDF por cada Protocolo y sus versiones, mantenga un formato como de RFC (al final de documento define lo minimo que deberia de llevar el documento), mantenga textos claros y cortos para cada explicación apoyado de un grafo, formato y/o ejemplo, los documentos a entregar son:
    - Protocolo en Capa Física (a nivel de Clase)
    - Protocolo en Capa de Transporte (Grupo)
    - Protocolo en Capa de Aplicación (Grupo)
    - Documentación de Protocolo de Virtual Devices (Grupo)

  - Código fuente de cada Capa implementada, las implementaciones deben de ser No deben ser consolidadas como un gran programa, cada una debe existir por separado.
    - Capa de Aplicación
    - Capa de Transporte
    - Capa Física (única con estándar de Protocolo a nivel de Clase)

- El Proyecto puede tener una calificación de cero si:
  - No está la persona de manera presencial al momento de la calificación, la nota no es grupal.
  - Si en el protocolo que se implementará a nivel de clase no es seguido y se implementa uno en solitario.
  - Si utiliza alguna librería que solucione algún problema completo o de conexión, a excepción de UART.
  - Si no entrega Documentación de Protocolo Estructurado y versiones de su progreso/cambios.

Formato Mínimo de RFC

1. **RFC Header**

Vamos a tener un mínimo de campos para el Header del RFC que serán:

RFC Number: Número de identificación único del RFC.
Versión: Número de versión de RFC iniciando en V 1.0
Título: Título del RFC que describe brevemente el contenido.
Autor(es): Nombre y afiliación del autor o autores del RFC.
Fecha: Fecha de publicación del RFC.
Categoría: Categoría del RFC (ej., Estándar, Experimental).

Se deben generar versiones experimentales y cuando ya sea la final la cambiaremos a Estándar.

2. **Abstract**

Un resumen conciso del propósito y contenido del RFC.

3. **Tabla de Contenido (El índice)**

Incluye las secciones y subsecciones del documento.

4. **Introducción**

Presentación del problema o necesidad que aborda el RFC para el protocolo que se va a definir, junto con los objetivos generales.

5. **Terminología**

Definiciones de términos y acrónimos utilizados en el documento.

6. **Especificación**

Descripción técnica detallada del estándar o protocolo propuesto. Esta sección puede dividirse en varias subsecciones según la complejidad del tema. Especificaciones de formato, transmisión, partes involucradas, Headers, definición de actuadores, etc.

7. **Consideraciones de Implementación**

Aquí puede definir consideraciones prácticas para la implementación del estándar o protocolo, incluyendo requisitos, dependencias, y limitaciones, ya sea a nivel de software o hardware.

8. **Consideraciones de Seguridad**

Aspectos de seguridad relacionados con el estándar o protocolo a nivel de software o hardware.

9. **References**

Lista de documentos, estándares, y RFCs mencionados en el texto. Todo documento es autorreferencial a su versión predecesora, como si de un gestor de control de código estuviéramos ejecutando.

10. **Expresiones de gratitud**

Reconocimiento a personas o entidades que contribuyeron al desarrollo del RFC.

11. **Dirección del autor**

Información de contacto del autor o autores,

12. **Appendix (Opcional)**

Información adicional, como ejemplos de código, que complementen la especificación técnica, no incluir imágenes en vez crear representaciones ASCII de tablas o formatos para que el documento sea liviano.