Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

16 de mayo de 2025

TRABAJO ACADÉMICO

UART

REDES INDUSTRIALES | GRUPO A1

Grado en Informática Industrial y Robótica

Autor/a:

Lourdes Francés Llimerá | lfralli@epsa.upv.es

Ángela Espert Cornejo | aespcor@etsinf.upv.es

Tutor/a:

Lenin Guillermo Lemus Zúñiga | lemus@upvnet.upv.es

TABLA DE CONTENIDOS

PÁGINA EN BLANCO

LISTADO DE TABLAS

No se encuentran elementos de tabla de ilustraciones.

PÁGINA EN BLANCO

LISTADO DE FIGURAS

PÁGINA EN BLANCO

LISTADO DE CÓDIGO

PÁGINA EN BLANCO

1. INTRODUCCIÓN

1.1. INTRODUCCIÓN

En las prácticas de Redes Industriales, se ha trabajado con la ESP32-S3. Durante el desarrollo de las clases, se ha solicitado que los alumnos realicen tres montajes, uno por cada tipo de comunicación dado en teoría, siendo estos RS485, UART y CAN.

1.2. OBJETIVO

El objetivo de este documento es recopilar y organizar de forma estructurada los códigos, montajes y conocimientos adquiridos a lo largo del desarrollo del montaje relacionado con el protocolo UART, presentando un resumen claro y coherente de los contenidos.

PÁGINA EN BLANCO

2. MONTAJE PROTOCOLO UART

2.1. DEFINICIÓN

El protocolo UART (*Universal Asynchronous Receiver-Transmitter*) es un estándar de comunicación serial ampliamente utilizado en la industria electrónica para el intercambio de datos entre dispositivos. Este protocolo resulta particularmente útil en aplicaciones que requieren una comunicación simple y de baja velocidad debido a su sencilla implementación y su capacidad para transmitir datos de manera eficiente en contextos donde la velocidad de trasmisión no es un requisito esencial.

Para realizar el intercambio de información, este protocolo se basa en la transmisión asincrónica de datos. El transmisor envía un bit de inicio seguido del byte de datos y uno o dos bits de parada. El receptor detecta el bit de inicio y se sincroniza con el transmisor para recibir el byte de datos. Esta sincronización permite que los dispositivos se comuniquen de manera eficiente sin la necesidad de un reloj compartido.

CARACTERÍSTICAS

A continuación, se enumeran una serie de características que conforman el protocolo.

- **Comunicación asincrónica:** Como se ha mencionado anteriormente, no requiere un reloj común entre el transmisor y el receptor, lo que simplifica significativamente la implementación. Cada byte de datos se envía con un bit de inicio y uno o dos bits de parada, lo que permite al receptor sincronizarse con el transmisor.
- **Configuración flexible:** Ofrece una gran flexibilidad en términos de configuración. Los parámetros de comunicación, como la velocidad de transmisión (*baud rate*), el número de bits de datos, los bits de paridad y los bits de parada, pueden ser ajustados según las necesidades específicas de la aplicación.
- **Uso común:** Debido a su simplicidad y eficiencia, es ampliamente utilizado en una gran variedad de aplicaciones, ya sea desde la comunicación entre microcontroladores y sensores hasta la interacción con ordenadores personales a través de puertos seriales.
- **Bajo costo:** No requiere hardware adicional complejo, lo que reduce los costos de implementación.
- **Limitaciones de velocidad:** No es adecuado para aplicaciones que requieren altas velocidades de transmisión de datos.
- **Capacidad limitada:** No es ideal para aplicaciones que requieren una gran cantidad de datos o una alta tasa de transferencia.

2.2. CÓDIGO

Para implementar el protocolo UART se ha empleado el entorno de Arduino, donde se utilizan los pines TX (transmisión) y RX (recepción) del microcontrolador. Estos pines están conectados directamente a los pines correspondientes del dispositivo con el que se desea comunicar.

Para la realización del código se ha empleado la biblioteca *HardwareSerial*, que permite crear una comunicación UART adicional en pines digitales arbitrarios. Además, se han empleado dos ESP32-S3 para realizar dicha comunicación simulando una conversación emisor y receptor. Se adjunta a continuación el código.

ESP32-S3 EMISORA

```
#include <HardwareSerial.h>

HardwareSerial MySerial(1); // Definir un Serial para UART
const int MySerialRX = 18;
const int MySerialTX = 17;

void setup()
{
  Serial.begin(9600);

  // Inicializar el Serial con los pines para UART
  MySerial.begin(9600, SERIAL_8N1, MySerialRX, MySerialTX);
}

void loop()
{
  Serial.write("Enviando mensaje...\n");

  String message = "Hola";

  MySerial.println(message); // Envío del mensaje como texto
  delay(1000);               // Espera 1 segundo antes de enviar de nuevo
}
```

Código 1: ESP32-S3 Emisora.

ESP32-S3 RECEPTORA

```
#include <HardwareSerial.h>

HardwareSerial MySerial(1); // Definir un Serial para UART
const int MySerialRX = 18;
const int MySerialTX = 17;

void setup()
```

```
{
  Serial.begin(9600);

  // Inicializar el Serial con los pines para UART
  MySerial.begin(9600, SERIAL_8N1, MySerialRX, MySerialTX);
}

void loop()
{
  if (MySerial.available())
  {
    String message = MySerial.readStringUntil('\n');

    Serial.println("Mensaje recibido: " + message);
  }
}
```

Código 2: ESP32-S3 Receptora.

RESULTADOS OBTENIDOS

Los códigos se han probado siguiendo el siguiente orden:

1. Montaje de la ESP32-S3 emisora.
2. Subida del código de la ESP32-S3 emisora a la misma.
3. Montaje de la ESP32-S3 receptora.
4. Subida del código de la ESP32-S3 receptora a la misma y lectura por terminal de los mensajes recibidos.

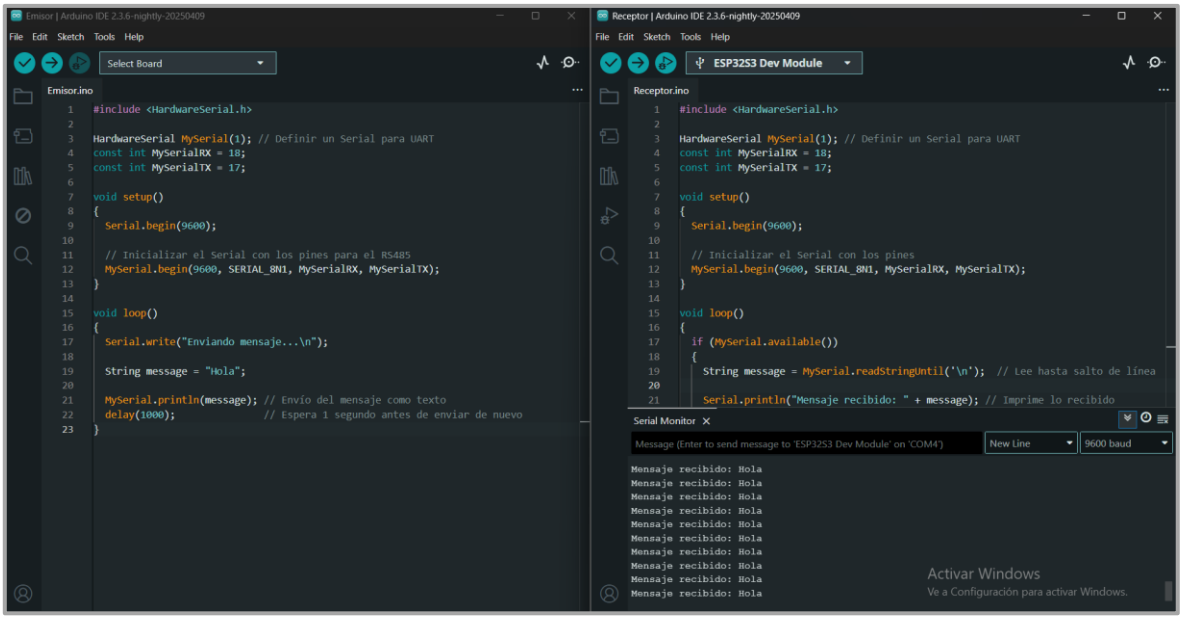

Figura 1: Funcionamiento código UART.

2.3. MONTAJE

MATERIAL NECESARIO

- ESP32-S3-WROOM-1.
- ESP32-S3 GPIO Extension Board.
- 3 cables, para realizar las conexiones pertinentes (en el montaje real, se emplean 3 cables macho-macho).
- Conectores para las ESP32-S3:
    - Uno que permita mantener la ESP32-S3 emisora conectada a la luz para que, tras subirle el código, transmita los datos continuamente.
    - Otro que permita conectar la ESP32-S3 receptora al ordenador, para que puedan comprobarse los resultados recibidos.

CONEXIONES

El pin que actúa como TX (transmisión de datos) de la ESP32-S3 emisora debe conectarse al pin RX (recepción de datos) de la ESP32-S3 receptora. A su vez, es posible realizar la conexión contraria para que ambas puedan actuar como emisoras y receptoras (este es el montaje que se ha realizado y que se muestra a continuación).

En este tipo de comunicación, es crucial conectar las masas (GND) de ambas ESP32-S3.

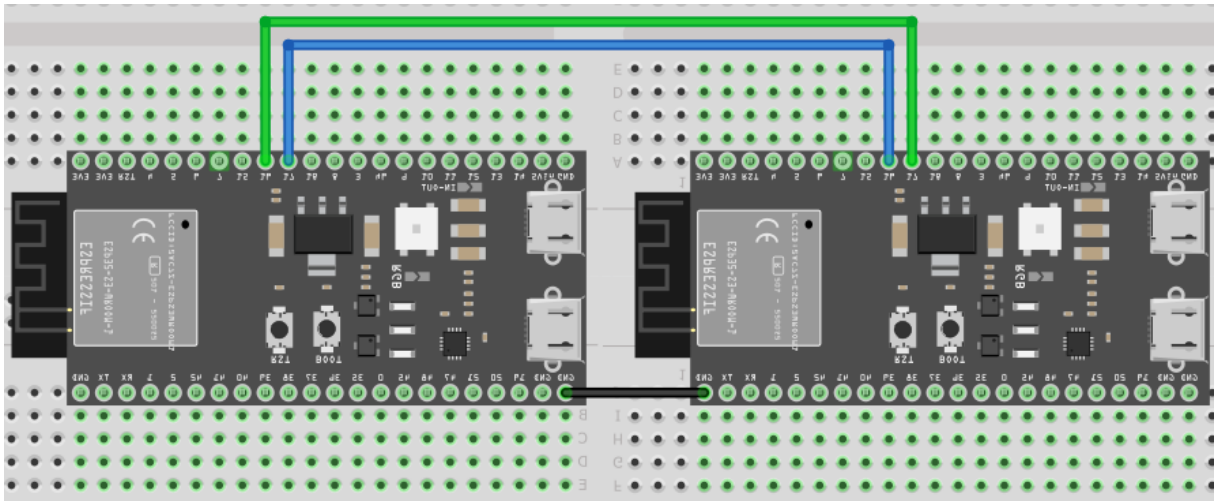

Figura 2: Conexiones UART.

3. BIBLIOGRAFÍA

3.1. INFORMACIÓN SOBRE PROTOCOLOS Y SUS MONTAJES

- Arduino. (n.d.). UART. Arduino Documentation. Recuperado de https://docs.arduino.cc/learn/communication/uart/
- Arduino.cl. (n.d.). Cómo configurar la comunicación UART en Arduino. Recuperado de https://arduino.cl/como-configurar-la-comunicacion-uart-en-arduino/

3.2. ELEMENTOS EMPLEADOS PARA LOS MONTAJES

- Fritzing Forum. (2024). *ESP32-S3 DevKit with USB-C*. Recuperado de https://forum.fritzing.org/t/esp32-s3-devkit-with-usb-c/24673