Proyecto Final

Sistema de Control Remoto de Actuadores con Monitoreo Ambiental

Técnicas Digitales 2

Docentes: Ing. Rubén Darío Mansilla

Ing. Lucas Abdala

Descripción

Este proyecto usa una placa STM32 para medir la luz ambiente con un sensor, controlar Leds y relés de forma remota con Bluetooth y mostrar la información en una pantalla LCD

FUNCIONAMIENTO

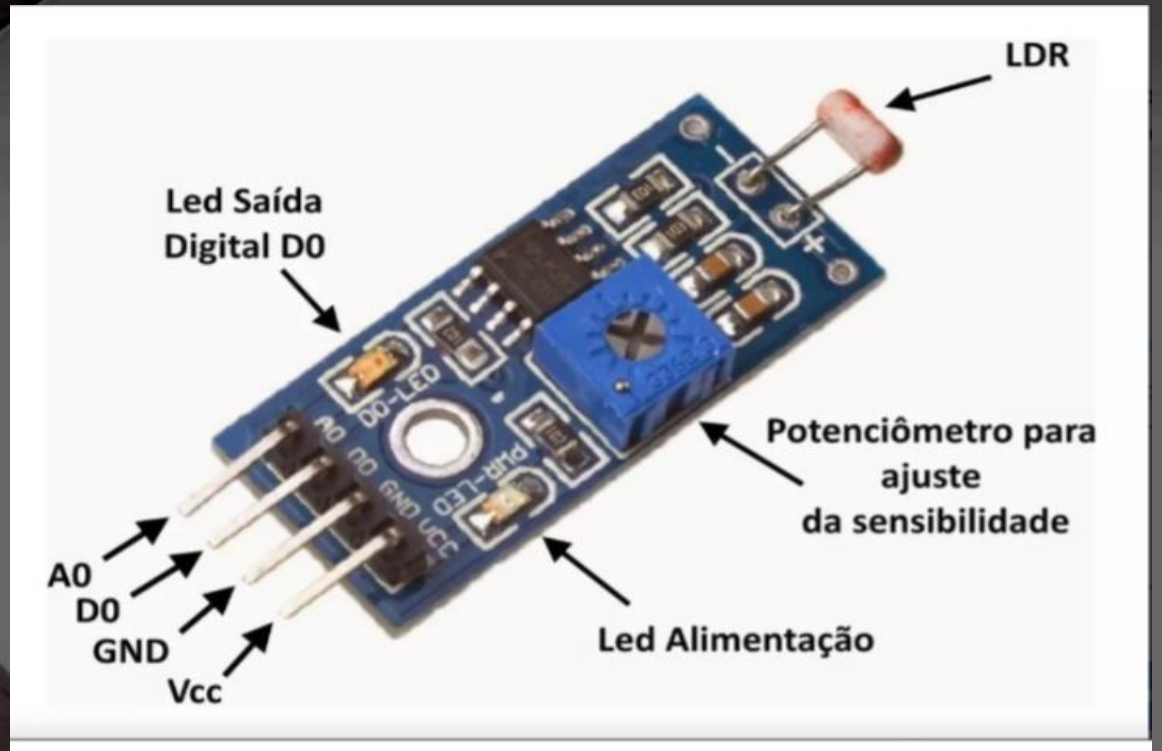
El sistema mide la luz ambiente con un fotosensor y actualiza la pantalla si hay cambios significativos. Recibe comandos por Bluetooth para controlar LEDs y relés, mostrando el estado en la LCD. Además, confirma cada acción recibida y muestra un mensaje de inicio

Módulo de 4 relés con optoacoplador

Placa que permite controlar hasta cuatro cargas de alta potencia (como motores o luces) mediante señales de bajo voltaje. Los optoacopladores aíslan eléctricamente el microcontrolador de la carga, aumentando la seguridad.

Módulo LDR (fotorresistor)

Sensor que varía su resistencia según la intensidad de luz recibida. Se usa comúnmente para medir niveles de iluminación o activar sistemas de manera automática en función de la luz ambiental.

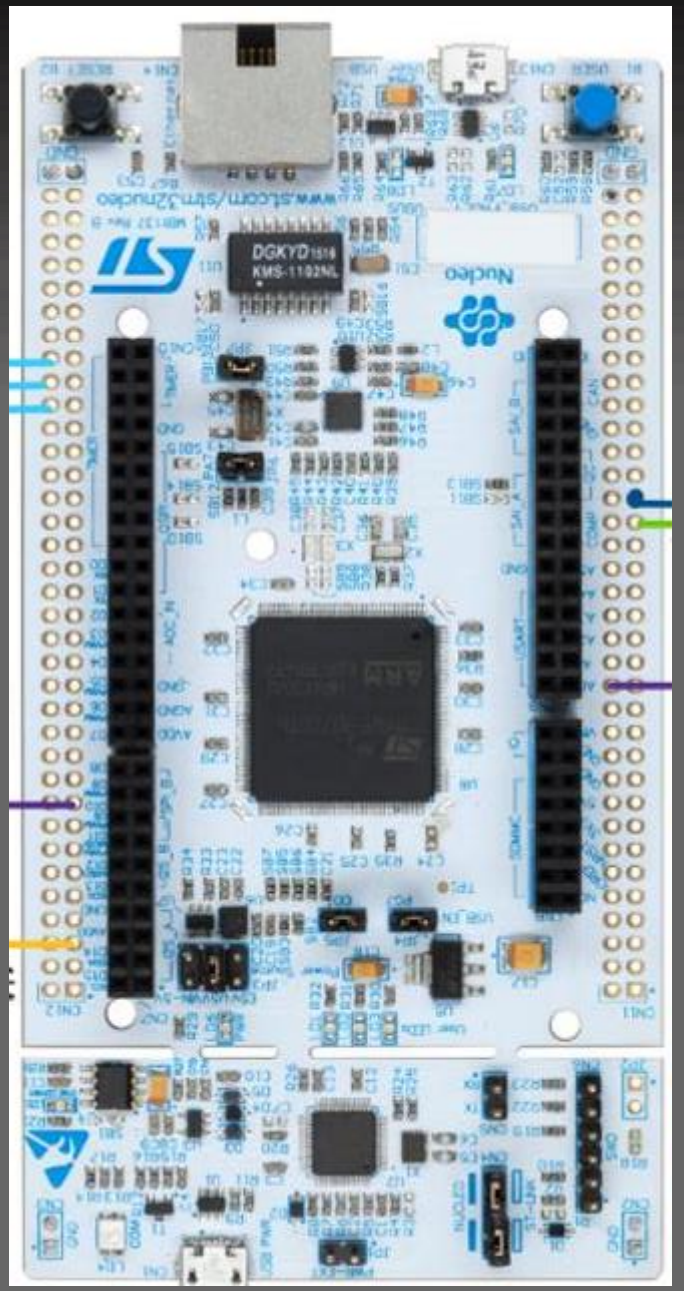


Pantalla 16x2 con interfaz I2C

Display LCD de 16 caracteres por 2 líneas, equipado con un módulo I2C que simplifica su conexión al microcontrolador al reducir el número de pines necesarios. Se usa para mostrar información como mensajes, lecturas de sensores o estados del sistema.

Nucleo-F429

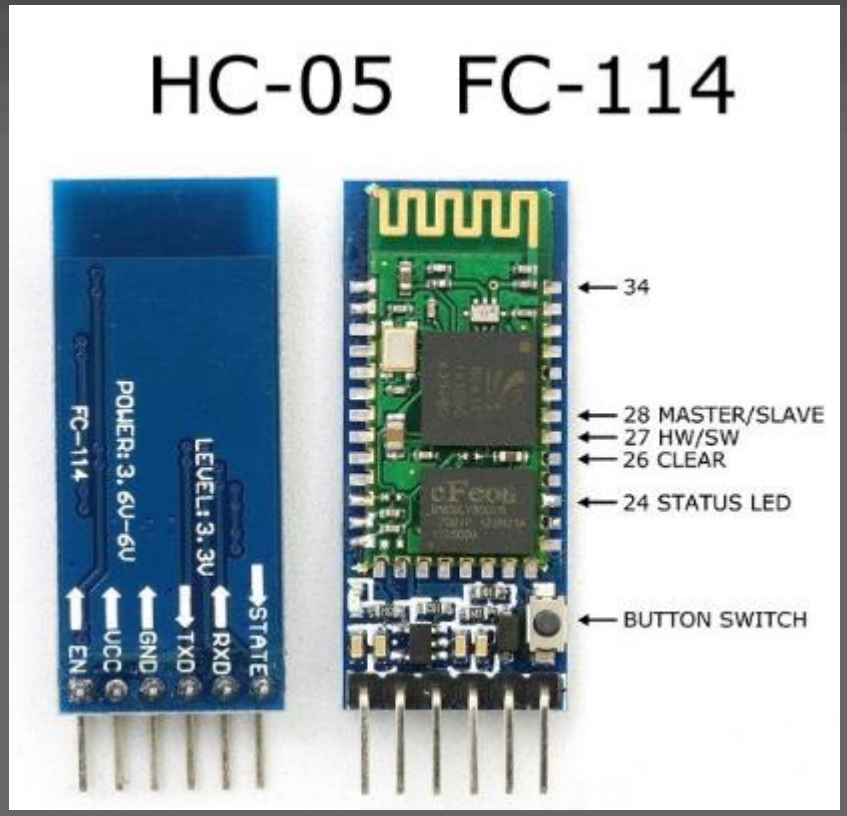
Placa de desarrollo basada en el microcontrolador STM32F429, que ofrece alto rendimiento, procesamiento en tiempo real y múltiples interfaces de comunicación. Ideal para proyectos avanzados de control y procesamiento de datos



Componentes:

Módulo Bluetooth HC-05

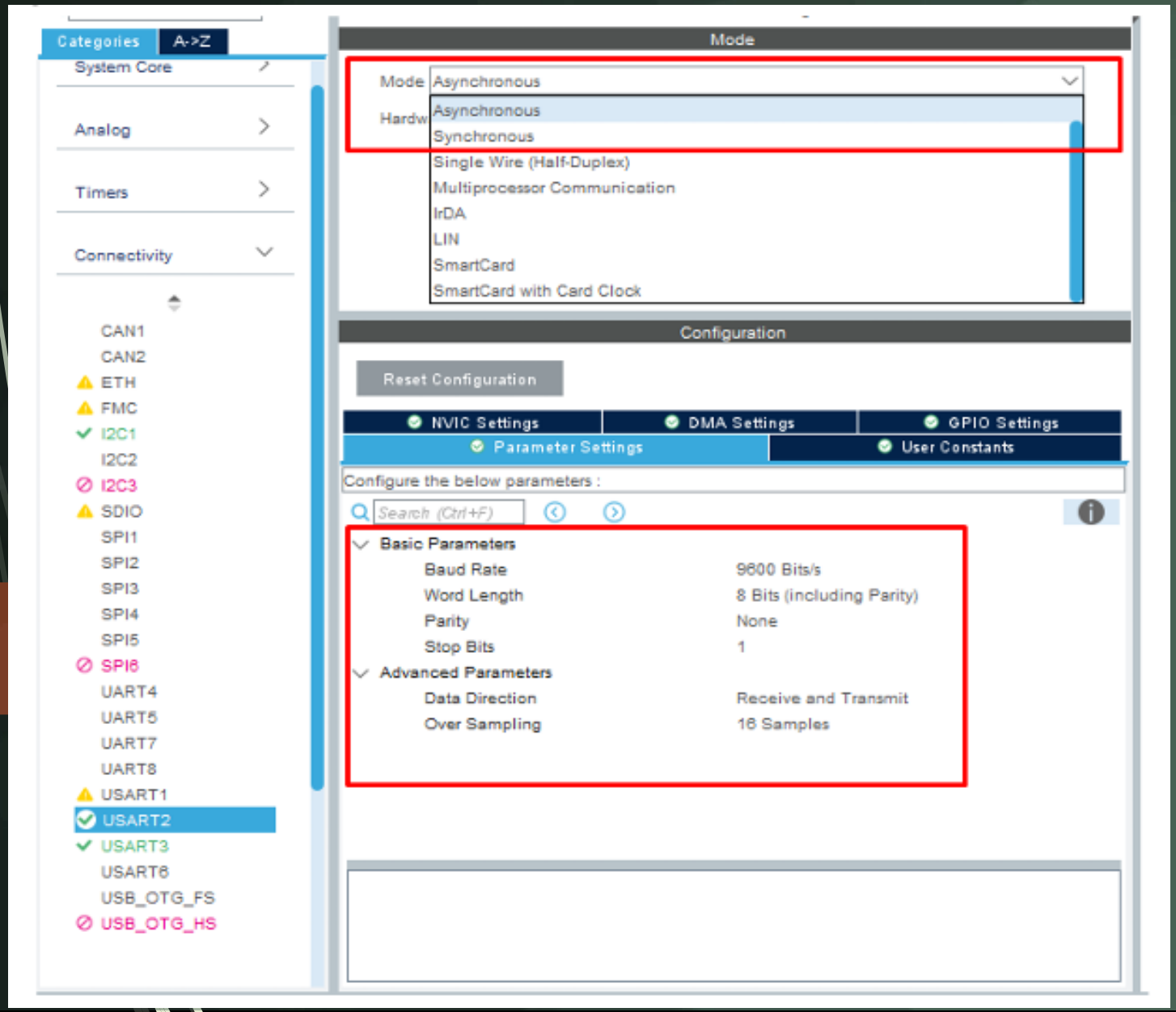
Permite la comunicación inalámbrica mediante el protocolo Bluetooth, ideal para conexiones serie (UART) entre microcontroladores y otros dispositivos. Soporta modos maestro y esclavo



El módulo HC-05 viene por defecto configurado de la siguiente forma:

- Modo: Esclavo
- Nombre por defeco: HC-05
- Código de emparejamiento por defecto: 1234
- La velocidad por defecto (baud rate): 9600

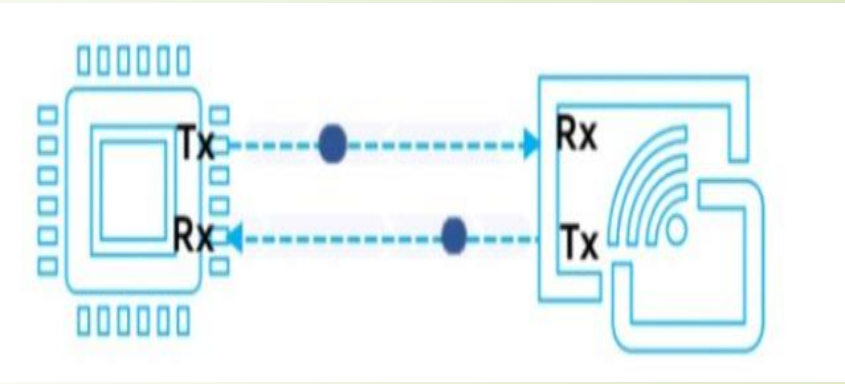
Protocolo Uart:

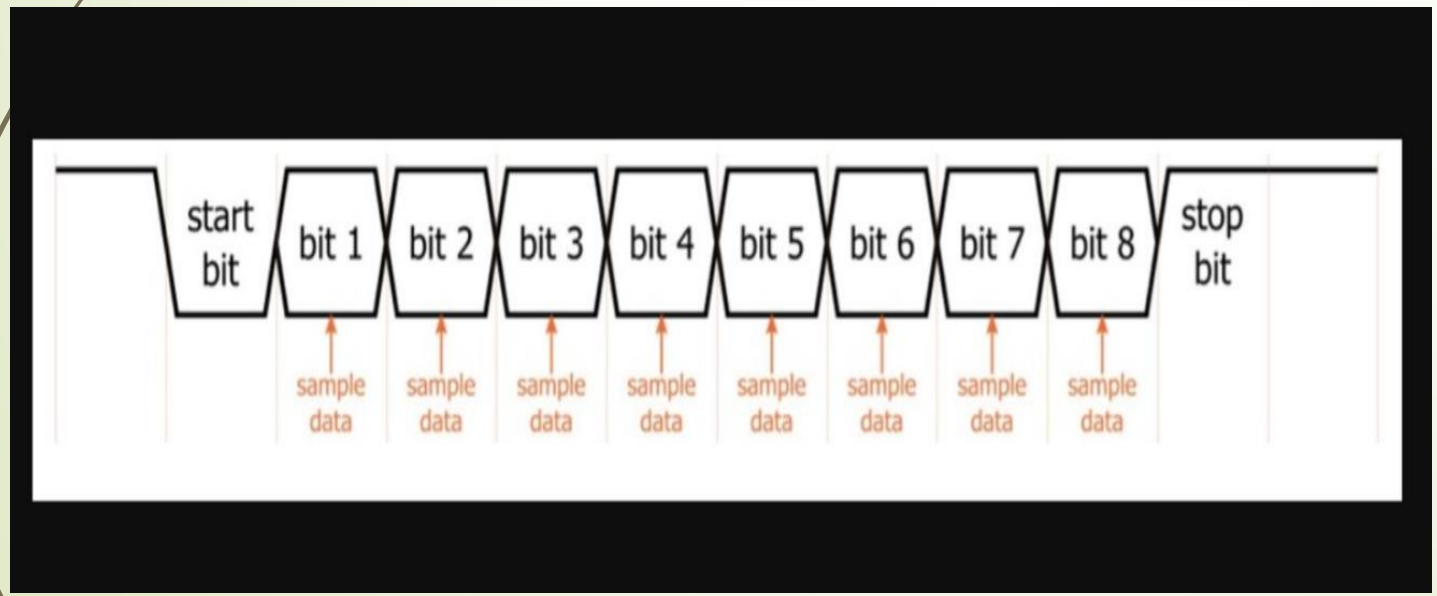


- Comunicación serie asíncrona sin señal de reloj.
- Modo: Asynchronous.
- Baud Rate: 9600 bps.
- Datos: 8 bits, sin paridad, 1 bit de stop.
- Dirección: Transmisión y recepción.
- Over Sampling: 16 samples.

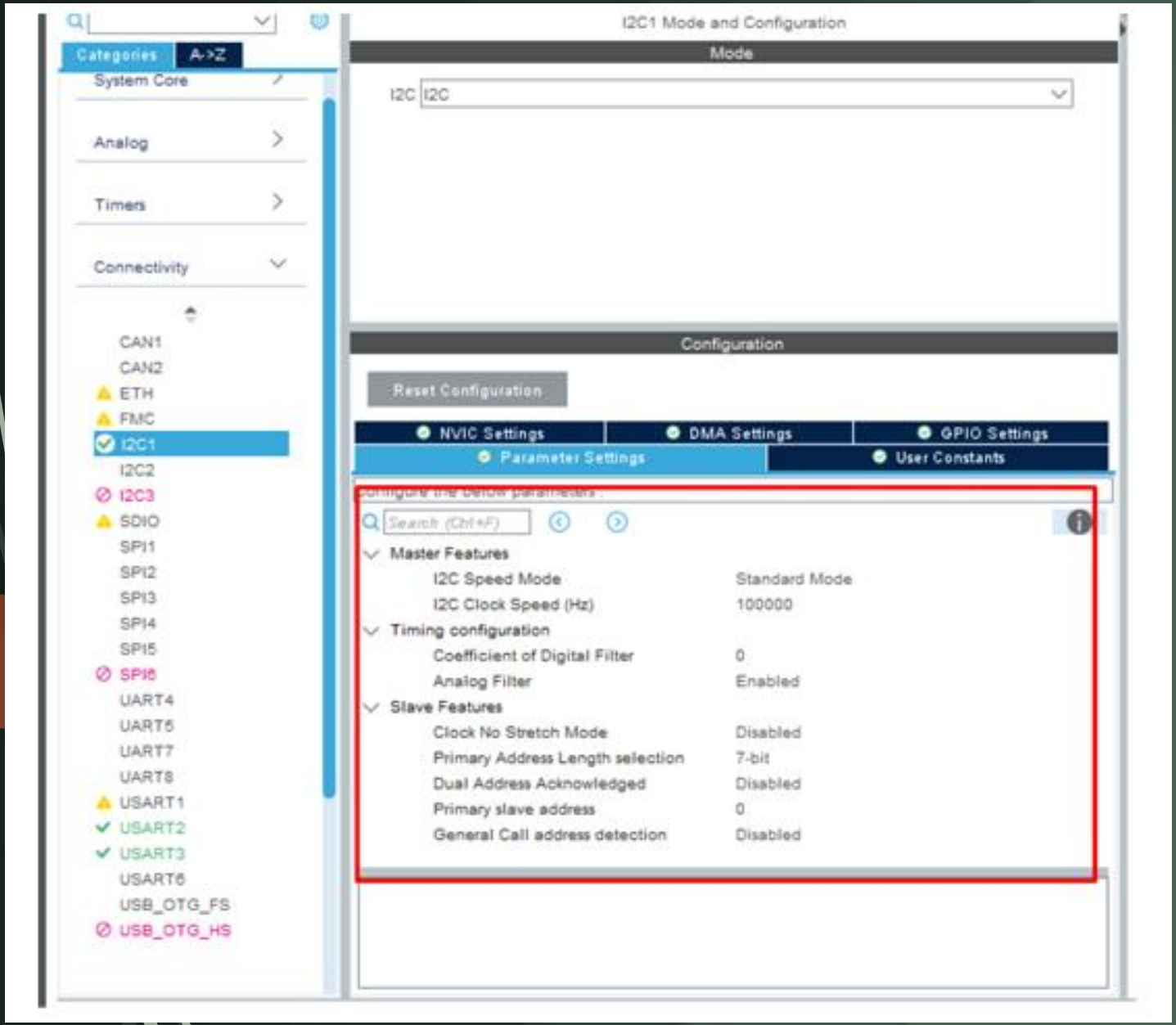
Consideraciones UART

- Tasa de transmisión de datos.
- Formato de trama del mensaje.
- Estándar de comunicación.



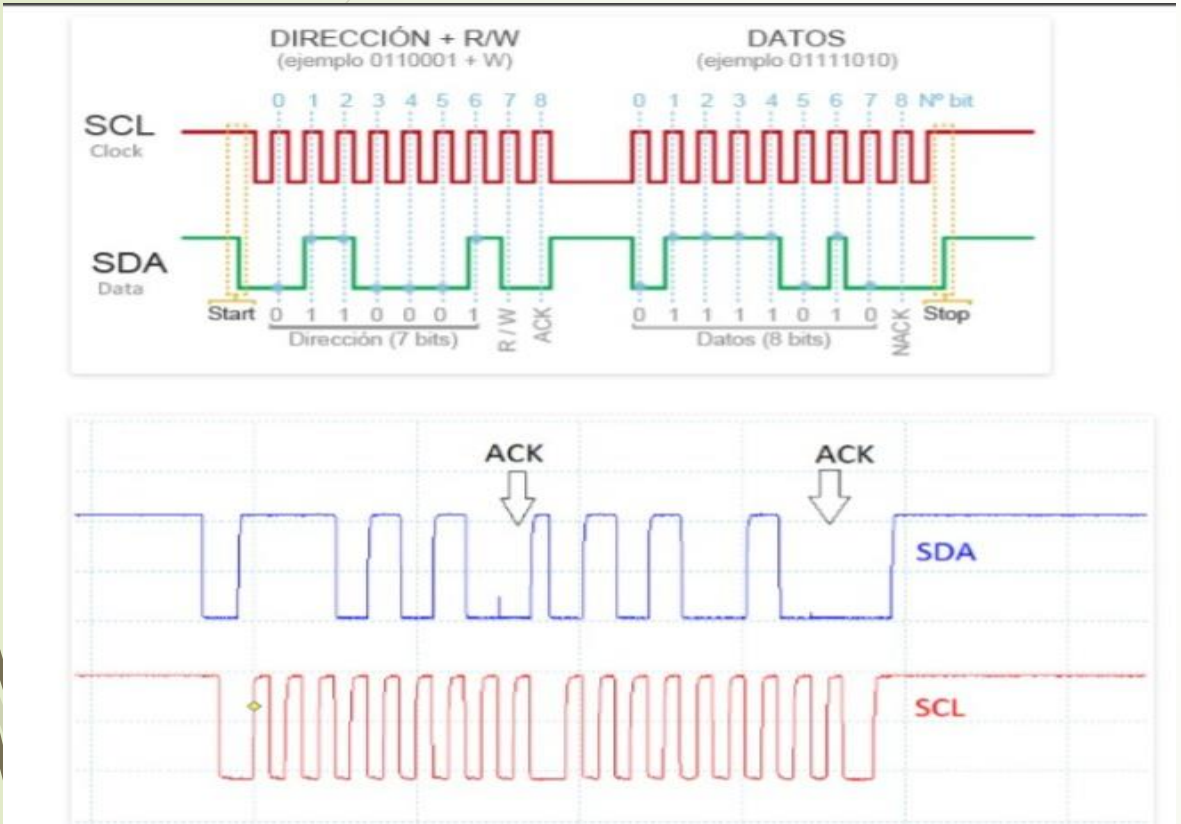


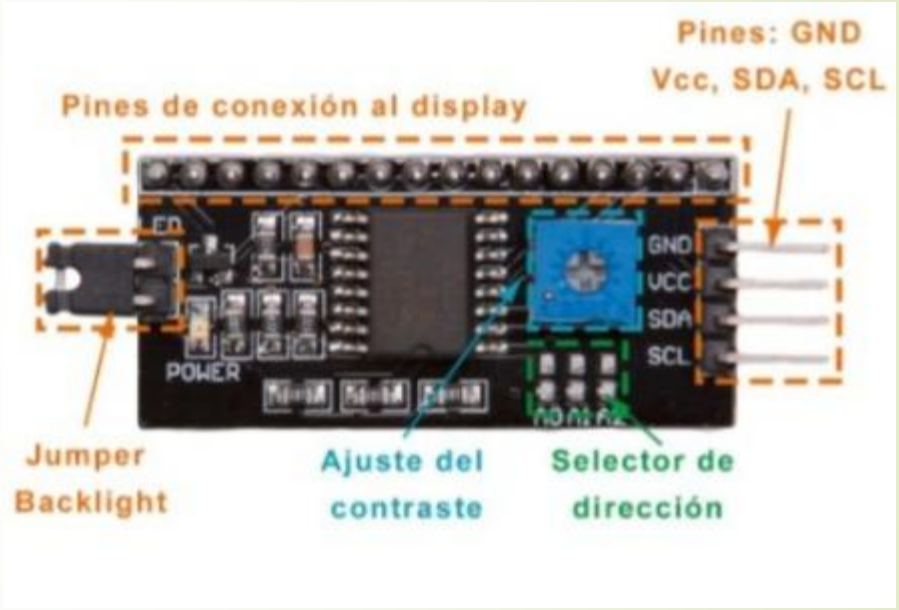
Protocolo I2C:



- Comunicación serie síncrona utilizada para conectar múltiples dispositivos con solo dos líneas: SDA (datos) y SCL (reloj)
- .Modo de operación en la STM32: Master en Standard Mode (100 kHz)
- Filtro analógico habilitado para reducir ruido en la señal.
- Dirección de 7 bits para identificar dispositivos esclavos.
- Clock Stretching" deshabilitado para evitar

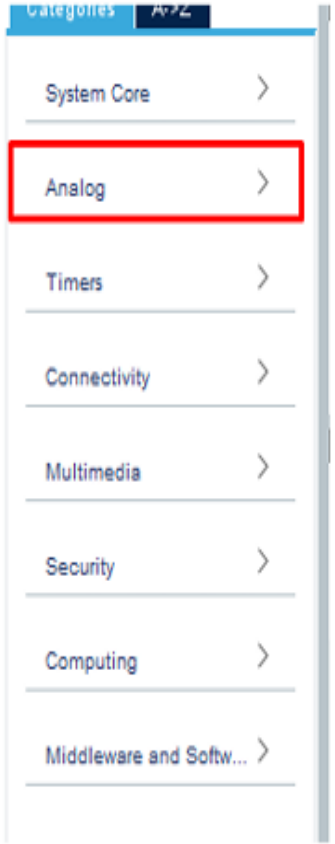
Consideraciones I2C

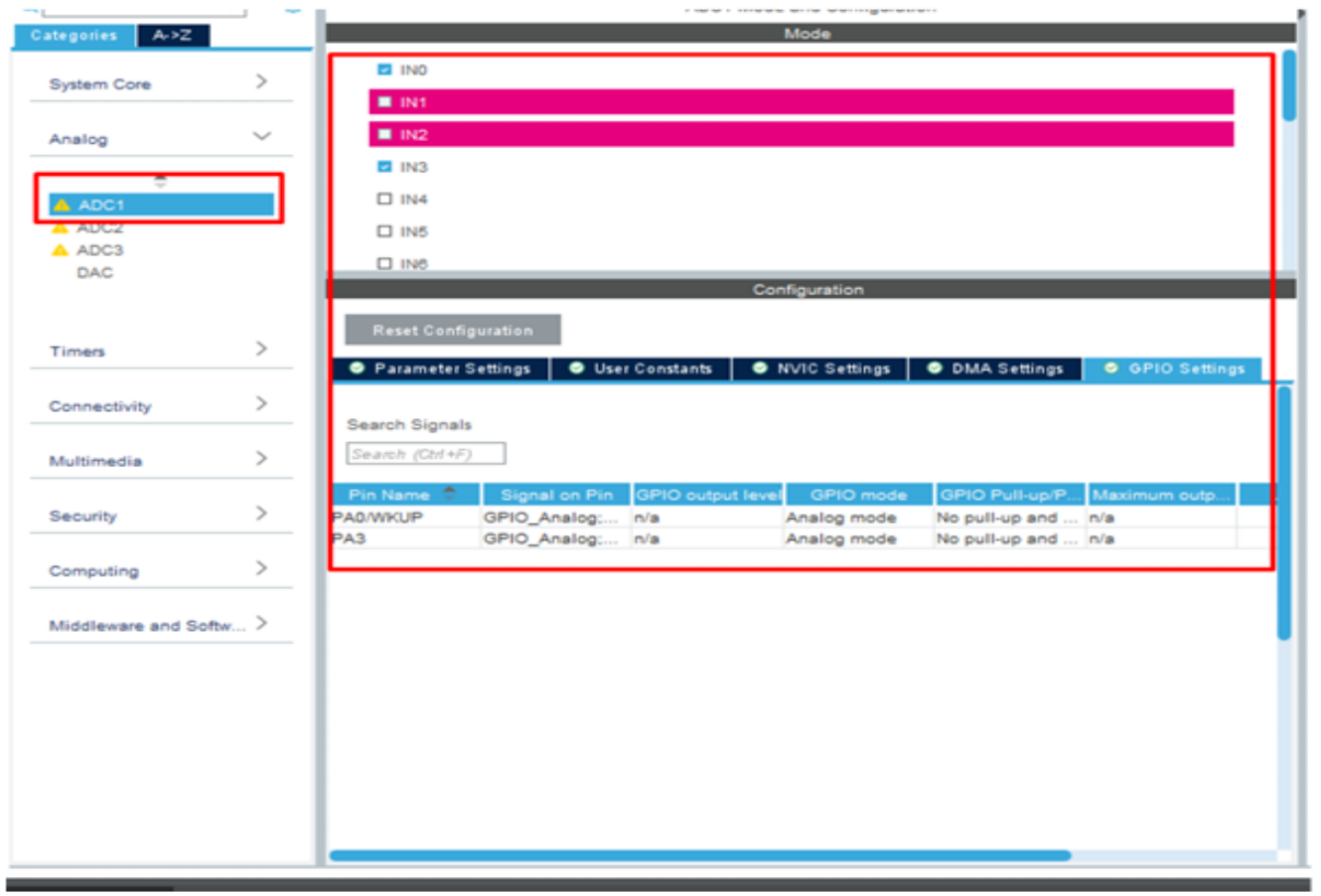




Conversor ADC:

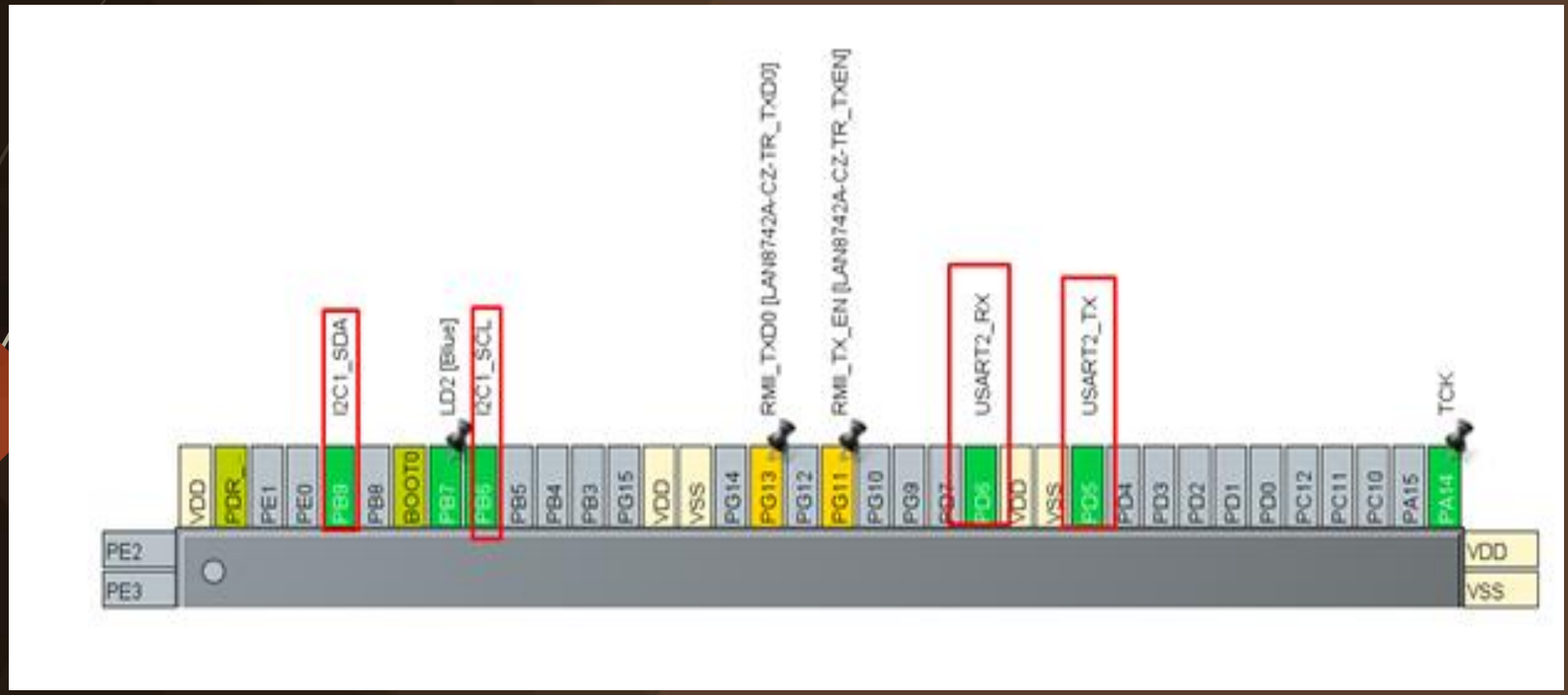
El conversor analógico-digital (ADC) se utiliza para digitalizar señales provenientes de sensores, permitiendo que el microcontrolador STM32F429ZI procese y actúe en función de estos valores. Su implementación es clave para interpretar magnitudes analógicas y utilizarlas en el control del sistema.



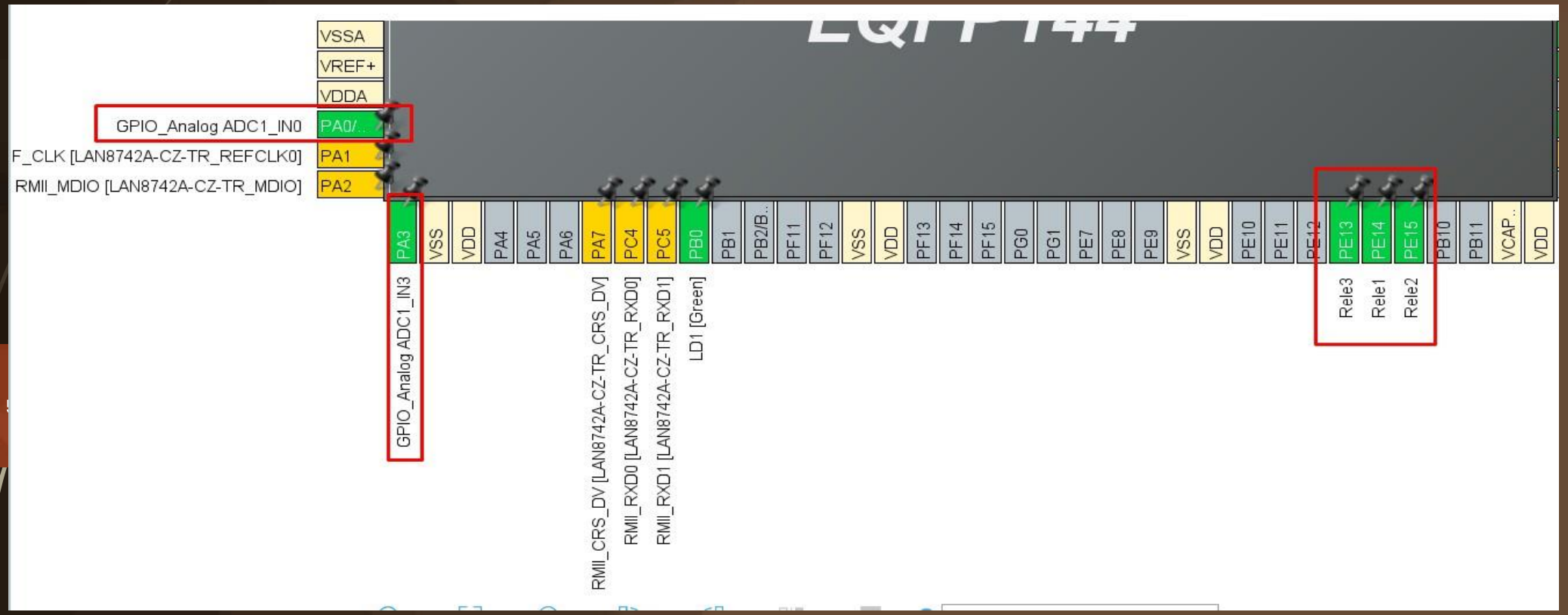


Configuración de pines:

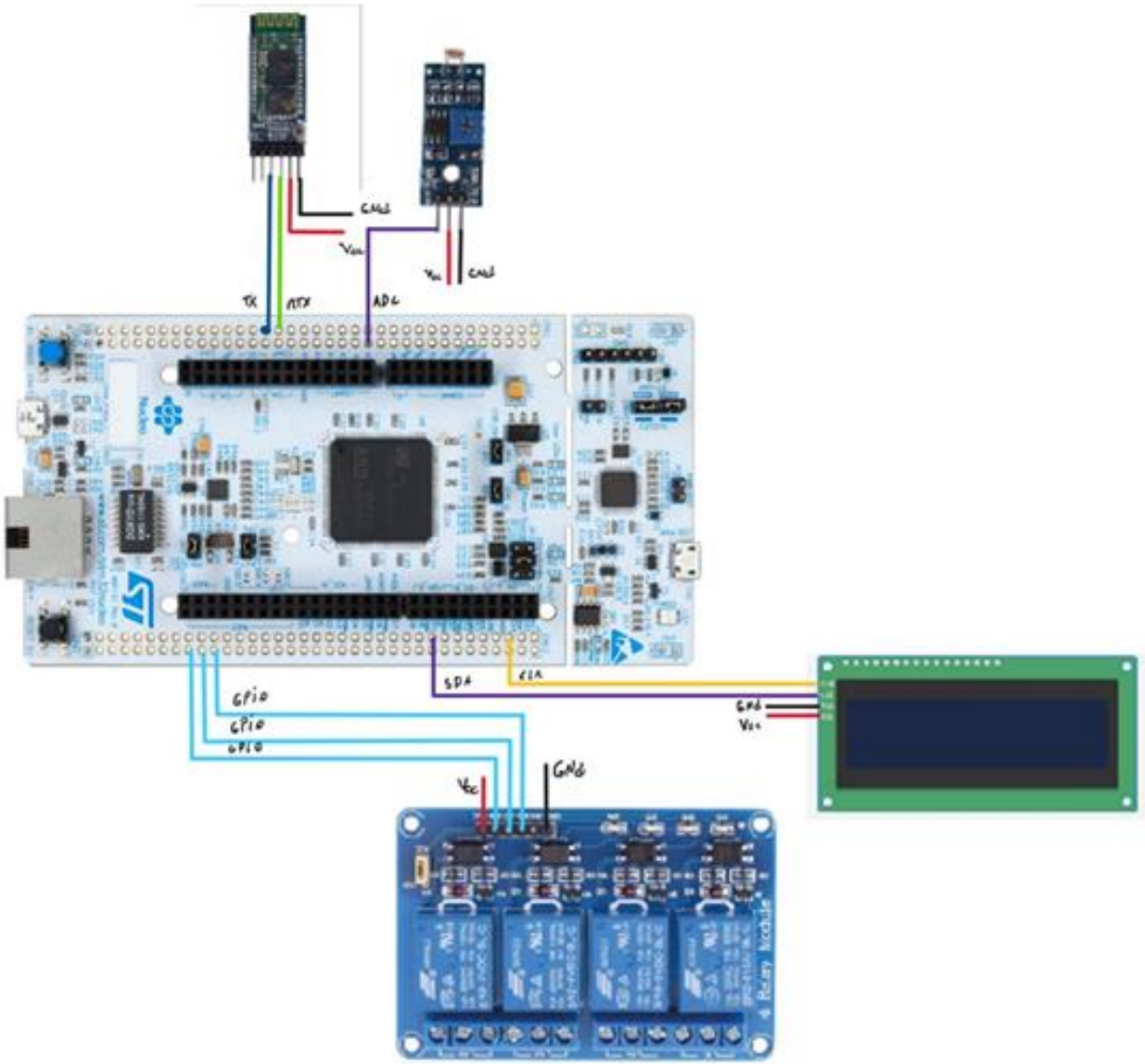
Se puede apreciar cómo están configurados los pines correspondientes a la comunicación I2C y UART



En esta otra podemos ver los pines asignados a los relés y al conversor ADC

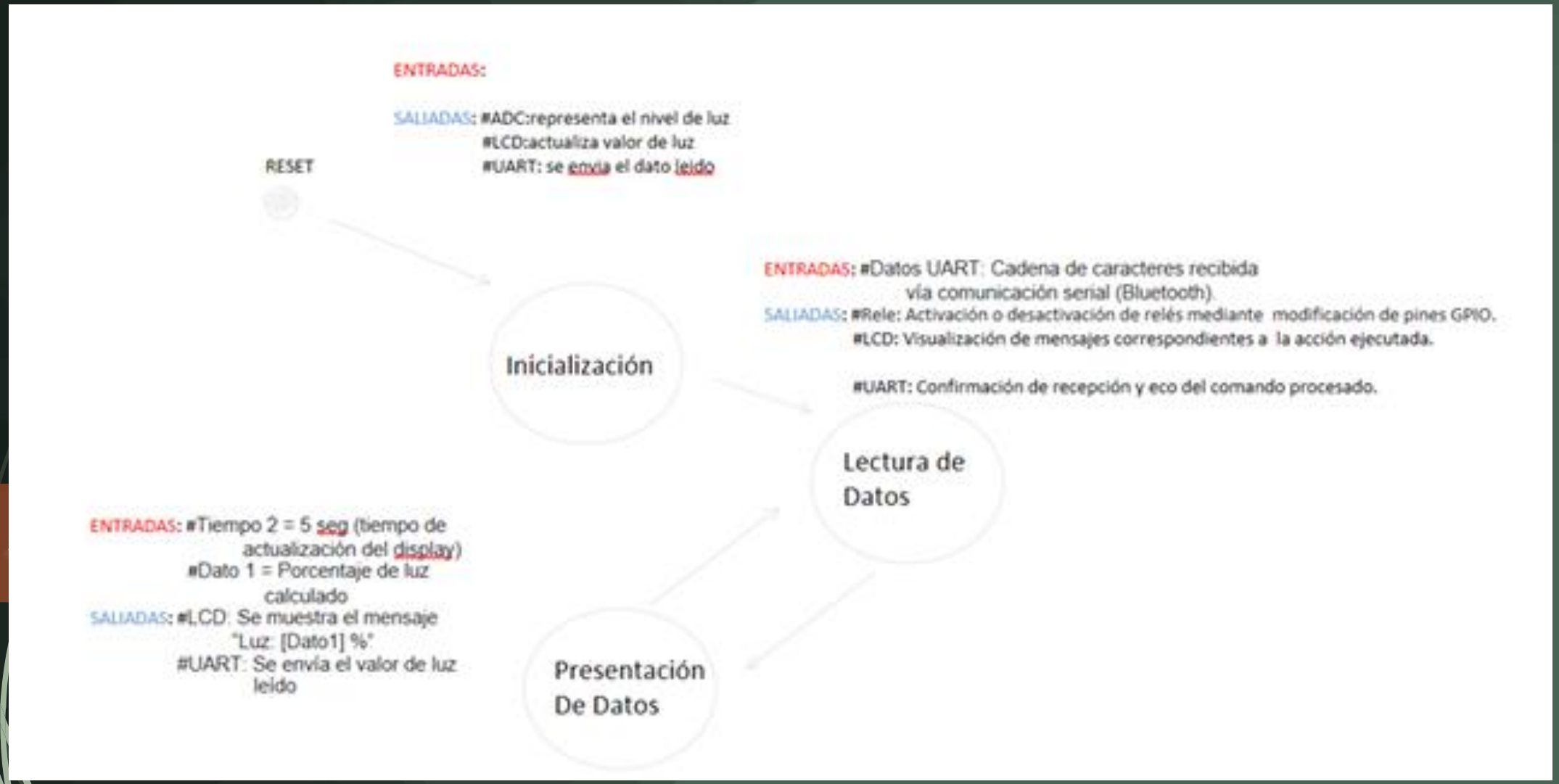


Conexión Física:



NOTA: se utilizará una fuente externa que se conecta en **VCC** y **GND**

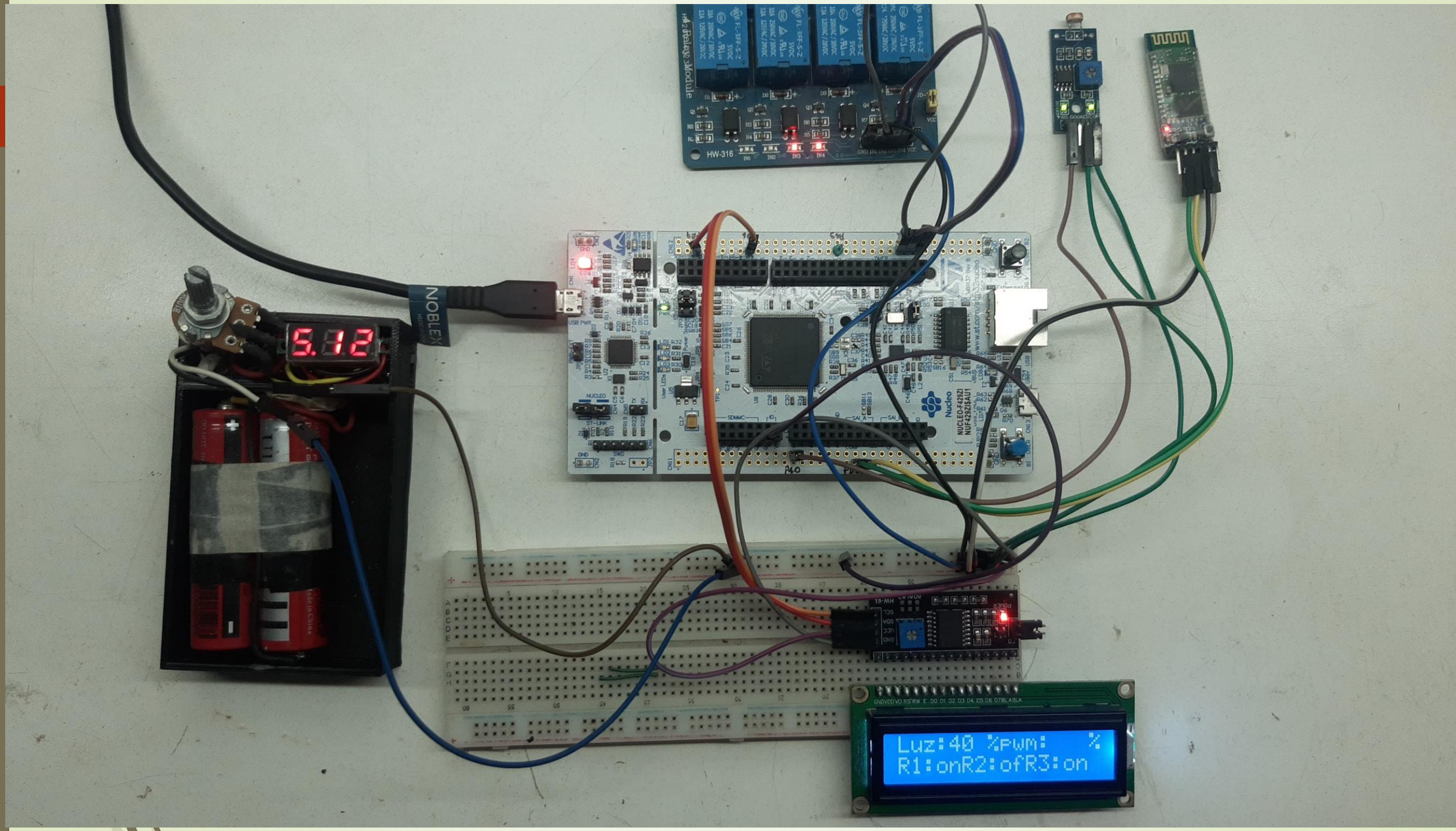
Diagrama de Estados:

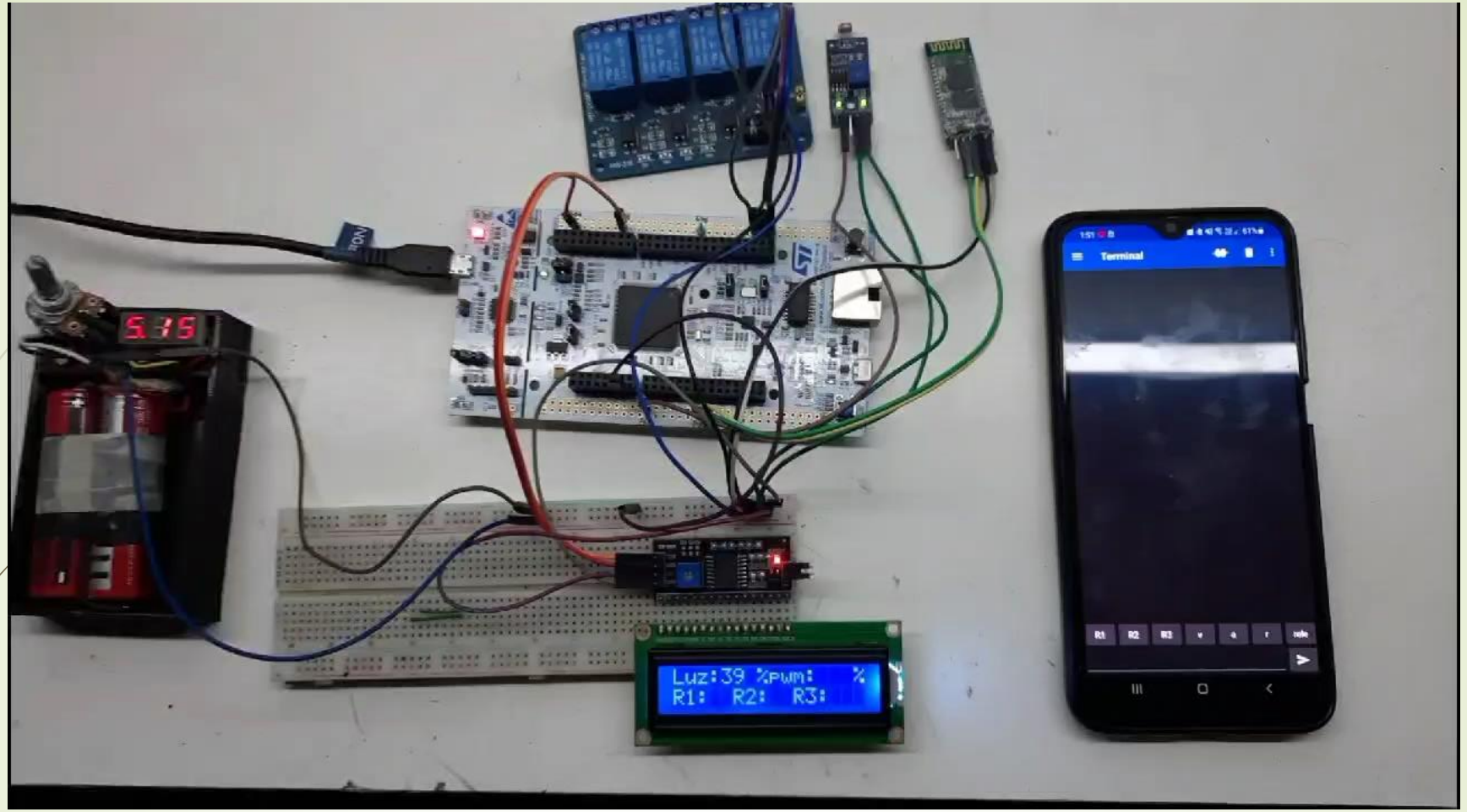


Revision de codigo main

```
while (1) {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */


    // --- Lectura del valor analógico desde el fotosensor ---

       // Inicia el ADC para comenzar una nueva conversión
    HAL_ADC_Start(&hadc1);

    // Espera la conversión
     // Espera a que la conversión finalice, con un timeout de 10 ms
    if (HAL_ADC_PollForConversion(&hadc1, 10) == HAL_OK) {
        // Si la conversión fue exitosa, obtiene el valor convertido
        photoSensorValue = HAL_ADC_GetValue(&hadc1);
        // Usa el valor adcValue
     // En este punto se puede procesar o usar el valor photoSensorValue según necesidad

    }

    // Detiene el ADC para liberar recursos hasta la próxima lectura
    HAL_ADC_Stop(&hadc1);


    // --- Verificación de datos recibidos por UART ---

        // Lee si hay algún dato recibido desde la UAR
    receivedData = uartGetReceivedData();

    //confirmacion de recepcion y comparacion de datos
    // Si hay datos recibidos (se chequea que el string no esté vacío)
    if (strlen((char*) receivedData) > 0) {
         // Envia por UART un mensaje de confirmación de recepción
        uartSendString((uint8_t*) "Recibido: ");
        uartSendString((uint8_t*) receivedData);
        uartSendString((uint8_t*) "\n");

        // Comparar los datos recibidos con los comandos
        if (strcmp((char*) receivedData, "Encender led verde") == 0) {
            writeLedOn_GPIO(LD1_Pin);
        }
        if (strcmp((char*) receivedData, "Encender led rojo") == 0) {
            writeLedOn_GPIO(LD3_Pin);
        }
        if (strcmp((char*) receivedData, "Encender led azul") == 0) {
            writeLedOn_GPIO(LD2_Pin);
        }
        if (strcmp((char*) receivedData, "Apagar led verde") == 0) {
            writeLedOff_GPIO(LD1_Pin);
        }
        if (strcmp((char*) receivedData, "Apagar led rojo") == 0) {
```

```
    }
    if (strcmp((char*) receivedData, "RaN") == 0) {
        HAL_GPIO_WritePin(GPIOE, rele1_Pin, GPIO_PIN_RESET);
        uartSendString((uint8_t*) "rele activado");
        lcd_enviar("R1:on", 1, 0);
    }
    if (strcmp((char*) receivedData, "RbN") == 0) {
        HAL_GPIO_WritePin(GPIOE, rele2_Pin, GPIO_PIN_RESET);
        uartSendString((uint8_t*) "rele activado");
        lcd_enviar("R2:on", 1, 5);
    }
    if (strcmp((char*) receivedData, "RcN") == 0) {
        HAL_GPIO_WritePin(GPIOE, rele3_Pin, GPIO_PIN_RESET);
        uartSendString((uint8_t*) "rele activado");
        lcd_enviar("R3:on", 1, 10);
    }
    if (strcmp((char*) receivedData, "RaFF") == 0) {
        uartSendString((uint8_t*) "rele desactivado");
        HAL_GPIO_WritePin(GPIOE, rele1_Pin, GPIO_PIN_SET);
        lcd_enviar("R1:of", 1, 0);
    }
    if (strcmp((char*) receivedData, "RbFF") == 0) {
        uartSendString((uint8_t*) "rele desactivado");
        HAL_GPIO_WritePin(GPIOE, rele2_Pin, GPIO_PIN_SET);
        lcd_enviar("R2:of", 1, 5);
    }
    if (strcmp((char*) receivedData, "RcFF") == 0) {
        uartSendString((uint8_t*) "rele desactivado");
        HAL_GPIO_WritePin(GPIOE, rele3_Pin, GPIO_PIN_SET);
        lcd_enviar("R3:of", 1, 10);
    }
}

//formamos mensaje en pantalla

cadena = 100 - (photoSensorValue / 4095.0) * 100;
if(abs(cadena - bufferaux) >= 5){
            sprintf((char*) uartRxBuffer, "Luz:%u  ", cadena);
            //lcd_clear();
            lcd_enviar((char*) uartRxBuffer, 0, 0);
            lcd_enviar((char *)"%", 0, 7);
            uartSendString((uint8_t*) uartRxBuffer);
            uartSendString((uint8_t *)"\n");
            bufferaux=cadena;

}
```


HW-316
NOBLEX
5.12
Nucleo
Luz:40 %pwm: %
R1:onR2:ofR3:on



Enlace de video: https://www.youtube.com/shorts/CpNqjV8YPLs

Conclusión:

El proyecto nos permitió integrar hardware y software para crear un sistema que monitorea luz ambiente y controla actuadores de forma remota por Bluetooth. Aplicamos lo aprendido en la materia, trabajamos en equipo y logramos un sistema funcional, modular y fácil de ampliar en el futuro.