# Laboratorio token

Laura Camila Blanco Gómez
*Escuela de Ingeniería*
Universidad Sergio Arboleda
Bogotá, Colombia
laura.blanco01@usa.edu.co

Santiago Cáceres Linares
*Escuela de Ingeniería*
Universidad Sergio Arboleda
Bogotá, Colombia
santiago.caceres01@usa.edu.co

## I. Introducción

Este proyecto se enfoca en la creación de un sistema de verificación de TOKEN, para esto se hace uso de un programa en C en un PC y una STM32 conectada a una LCD.

Se tratan temas como la sincronización precisa entre sistemas, el uso eficiente de temporizadores, la comunicación USB entre la PC y el microcontrolador, así como la aplicación de operaciones hash y XOR para garantizar la seguridad de los TOKENs generados.

## II. Marco teórico

Como recapitulación de ciertos de los tópicos tratados durante el laboratorio se parte para el correcto entendimiento de algunos conceptos mencionados en la sección anterior.

**Comunicación USB:** La comunicación USB (Universal Serial Bus) es un estándar de comunicación serial que define los protocolos y conectores para la transmisión de datos entre dispositivos electrónicos. Utiliza un bus jerárquico y se divide en varios tipos de conectores (como USB-A, USB-B, micro USB, USB-C, entre otros). Los dispositivos USB pueden funcionar como hosts o periféricos, y pueden soportar diferentes velocidades de transferencia de datos.

**Operaciones XOR:** La operación XOR (Exclusive OR) es una operación lógica que retorna un valor verdadero (1) cuando el número de bits 1 es impar y un valor falso (0) cuando el número de bits 1 es par. Es utilizada en diversas aplicaciones, como en criptografía, codificación de datos y operaciones aritméticas.

| $A$ | $B$ | $A \oplus B$ |
|---|---|---|
| False | False | False |
| False | True | True |
| True | False | True |
| True | True | False |

Figure 1: Tabla de verdad xor

**Manejo de Tokens:** El manejo de tokens se refiere al proceso de gestión y control de unidades discretas de datos dentro de un sistema. En contextos de seguridad, los tokens pueden ser utilizados para autenticación de usuarios, control de acceso o como elementos de seguridad en protocolos de comunicación.

Estos permiten tener una mejor experiencia de usuario, seguridad digital añadida, control administrativo y una carga tecnológica más ligera.

**Validez Temporal:** La validez temporal se refiere a la vigencia o duración de un dato, un certificado o cualquier entidad que requiera una fecha de inicio y una fecha de expiración. En aplicaciones como certificados digitales, contratos electrónicos y otros sistemas, la validez temporal es crucial para garantizar la seguridad y la autenticidad de la información.

En el caso de un token se establece un vencimiento muy largo, el total de 200 días, sin embargo, para lugares con uso de datos y dinero estos tokens no pueden sobrepasar el día.

**Comunicación Serial desde PC a Microcontrolador:** La comunicación serial es un método de transmisión de datos en serie, uno a la vez, entre un emisor y un receptor. Los microcontroladores a menudo se comunican con un PC u otros dispositivos a través de un puerto serial (como UART) para intercambiar información. Esto es fundamental en aplicaciones de sistemas embebidos.

## III. Procedimiento

Para este apartado se divide en tres partes las cuales son.

- **Montaje** Para este apartado del procedimiento se hizo la conexión entre la STM y la pantalla lcd como se ve en la figura 2, adicionalmente se conecta un cable USB - C para poder alimentar la pantalla LCD.

- **Código PC** Para el código que se realizó en el PC se hizo uso del tutorial anexado en el laboratorio el cual permite y explica el cómo poder enviar datos por medio del puerto serial en este caso el puerto **COM3**, esto se logró gracias al uso de la API **WIN32**, pasos más importantes para este apartado fueron.
  - Uso de la API WIN32

- Configuración del puerto serial COM3
- Uso de la librería **sys/time**

estos tres puntos son los más importantes debido a que gracia a la API WIN32 por medio de sus métodos **WriteFile** y **SetCommState** las cuales nos permite enviar datos y configurar el puerto serial a usar, esto se ve en los siguientes códigos

```
        WriteFile(hSerial, buffer, ...
            strlen(buffer), &bytesWrite, ...
            NULL)
```

Código envió de datos

```
        DCB dcbSerialParams = {0};
        dcbSerialParams.DCBlength = ...
            sizeof(dcbSerialParams);

        if (!GetCommState(hSerial, ...
            &dcbSerialParams)) {
            fprintf(stderr, "Error al ...
                obtener los par metros ...
                del puerto. C digo de ...
                error: %d\n", ...
                GetLastError());
            CloseHandle(hSerial);
            return 1;
        }

        dcbSerialParams.BaudRate = ...
            CBR_9600; // Velocidad de ...
            transmisi n (9600 baudios)
        dcbSerialParams.ByteSize = 8; // ...
            Tama o del byte
        dcbSerialParams.StopBits = ...
            ONESTOPBIT; // 1 bit de parada
        dcbSerialParams.Parity = ...
            NOPARITY; // Sin paridad
```

código configuración de puerto serial. Dentro de las configuraciones del puerto serial se puede encontrar se encuentran las siguientes.

- Velocidad de transmisión
- Tamaño del byte
- bit de parada
- Paridad

Para poder terminar con este apartado se realizó el código, el cual permite consultar el token mostrado en la pantalla LCD en cualquier momento. Este código funciona por medio de un ciclo while infinito el cual se activa una vez se cierre el puerto serial y se haya ingresado el primer token que se quiera validar

```
    while (1) {
        if (tokenEscrito == 0) {
            printf("Digite su token\n");
            scanf("%s", tokenHexString);
            tokenEscrito = 1;
        } else {
            gettimeofday(&start, NULL);
            tokenHex = ...
                strtoul(tokenHexString, ...
                NULL, 16);
            tokenXor = xor_hex(tokenHex, ...
                clave);
            if (tokenXor ≥ (start.tv_sec ...
                - (5 * 3600) -30) && ...
                tokenXor ≤ (start.tv_sec ...
                - (5 * 3600))) {
                printf("Token Valido\n");
            } else {
                printf("Token Invalido\n");
            }
            tokenEscrito = 0;
        }
    }
```

Para evitar que este código este mostrando siempre el resultado o en su defecto siempre este pidiendo el token sin dejar ingresarlo se hace uso de la variable **TokenEscrito** la cual si es 0 es porque ya se validó un token y se está esperando el ingreso de un token nuevo o 1, este último valor lo pide una vez ya se haya ingresado el token a validar.

- **Código STM** Para el apartado de la tarjeta, la cual va a servir como el dispositivo, el cual se va a encargar de recibir los datos enviados para que esto se pueda realizar, se debe configurar el **Milddleware** de la tarjeta como **USB_DEVICE**, adicionalmente en el apartado de conectividad de la tarjeta se le debe escoger la opción **devide_only** una vez se hacen esta configuración se debe configurar el reloj el cual tendrá como frecuencia máxima 96MHz esto debido a la conexión USB.

  Una vez finalizadas estas configuraciones previas se debe realizar la configuración del **TIM** el cual nos permitirá avanzar el tiempo dentro de la tarjeta, a pesar de esta ya no estar recibiendo datos por el puerto serial, la configuración para del TIM para este laboratorio fue la siguiente.
  - Preescaler: 48000 - 1
  - CP: 2000 - 1

  esta configuración se realiza para que se pueda realizar una ISR cada 1 segundo, la cual aumentara el tiempo de la estructura **timeval** la cual se compone de dos características.
  - tv_sec
  - tv_usec

  esta estructura contiene en ambos campos un tiempo **UNIX**, el cual es la cantidad de segundo desde el 1 de enero de 1970 hasta la actualidad, en el caso del **tv_sec**, lo mismo aplica para **tv_usec**, pero en este caso serían los microsegundos desde la fecha antes mencionada.

  Una vez configurado el TIM que no ayudara para aumentar el tiempo de la estructura **timeval** como para poder contar el tiempo válido de un token, se procede a escribir una función en el archivo **usbd_cdc_if.c**. con el modificador **_weak** el cual nos permitirá sobreescribir esta función en el archivo principal main, la función escrita en este archivo será la encarga de recibir la

información enviada en el puerto serial esta función recibe dos parámetros los cuales son.

- buf: El cual es el que lleva la información
- len: Tiene el tamaño del buf

Esta función se ve de la siguiente manera.

```
void recive_CallBack(uint8_t *buf, uint32_t len) {
    tiempo = ((uint32_t) buf[0] << 24) | ((uint32_t) buf[1] << 16)
        | ((uint32_t) buf[2] << 8) | (uint32_t) buf[3];

    clave = ((uint32_t) buf[5] << 24) | ((uint32_t) buf[6] << 16)
        | ((uint32_t) buf[7] << 8) | (uint32_t) buf[8];

    tiempo_actual.tv_sec = (time_t) tiempo - (5 * 3600);
    sincronizado = 1;
    lcd_put_cur(0, 0);
    lcd_send_string("                ");
    lcd_put_cur(1, 0);
    lcd_send_string("                ");
}
```

Esta función no solo recibe los valores enviados por el puerto serial, sino que también asigna este tiempo una estructura de tiempo, en este caso la estructura **tiempo_actual**, para facilidad del el cálculo se decidió solo enviar y manejar el reloj de la STM con el segundo.

Adicionalmente, a este tiempo recibido se le resta lo siguiente (5 * 3600) se hace esto debido a que el GMT de Colombia es de -5, este cálculo también se realiza en la validación del token en el código del PC. una vez finalizado este apartado se procede a crear la función de la ISR la cual es la siguiente.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim) {
    if ( htim -> Instance == TIM1 ) {
       if(tiempo > 0){
          ms++;
          tiempo_actual.tv_sec ++;
          raw_time = tiempo_actual.tv_sec; // Convierte tv_sec a time_t
          timeinfo = localtime(&raw_time);
          if(sincronizado == 1){
              valorMiliSec = (uint32_t) tiempo_actual.tv_sec;
                resultXor = calcular_xor(valorMiliSec, clave);
                sprintf(tokenLcd, "%x", resultXor);
                lcd_put_cur(0, 0);
                lcd_send_string("Su token es :");
                lcd_put_cur(1, 0);
                lcd_send_string(tokenLcd);
                sincronizado = 0;
          }
       }

       if(ms == 30){
            lcd_put_cur(0, 0);
            lcd_send_string("                ");
            lcd_put_cur(1, 0);
            lcd_send_string("                ");
            ms = 0;
            valorMiliSec = (uint32_t)tiempo_actual.tv_sec;
            resultXor = calcular_xor(valorMiliSec, clave);
            sprintf(tokenLcd, "%x", resultXor);
            lcd_put_cur(0, 0);
            lcd_send_string("Su token es :");
            lcd_put_cur(1, 0);
            lcd_send_string(tokenLcd);

       }
    }
}
```

Como se mencionó anteriormente, esta función se encarga de dos cosas, las cuales son aumentar cada un segundo el valor de tv_sec para que el reloj siga contando el tiempo y calcular un nuevo token cada 30 segundos, esto lo hace aumentando la variable **ms** hasta el valor de 30, cuando esto sucede se calcula el nuevo token, este token se calcula usando la siguiente función.

```
uint32_t calcular_xor(uint32_t x1, uint32_t x2){
    return x1 ^ x2;
}
```

la cual realiza la operación XOR bit a bit por medio del operador $\wedge$

## IV. Resultados y análisis de resultados

para empezar con esta sección se va a tratar esta en dos partes

- **Código PC**, para este apartado se van a analizar las partes más importantes para los resultados, los cuales serían el cálculo para la verificación del token ingresado, para poder generar esta validación se necesitan de dos partes importantes las cuales fueron.
  - Cálculo XOR entre la clave y el token ingresado.
  - Rango de valores para la validación.

  Estos dos puntos son esenciales debido a que con el cálculo de la xor se obtendría el tiempo UNIX, el cual sería utilizado posteriormente para el cálculo del rango de valores para la validación. para este rango se debe tener en cuanta que existe un pequeño retraso el cual será dado por la validación del token, por lo cual este rango se definiría de a siguiente manera.

```
[(start.tv\underline{ }sec - (5 * 3600) -30) y (start.tv\underline{ }sec - (5 * 3600)]
```

siendo el valor que se obtenga de la operación XOR el límite superior de este rango, adicionalmente tomando este rango de tiempo, el cual visto lógicamente se comportaría de la siguiente manera

```
tokenXor ≥ (start.tv_sec - (5 * 3600) ...
    -30) && tokenXor ≤ (start.tv_sec - ...
    (5 * 3600))]
```

Otra parte importante para la obtención de los resultados por parte del código del PC es que a pesar de que el token se visualiza en base 16 en la pantalla LCD, esto solo se hace por comodidad tanto a la hora de poder ingresar el token como a la hora de visualizarlo porque lo muestra de una manera más compacta, pero a la horra de realizar los cálculos del token para poder obtener el tiempo este cálculo se debe hacer en las bases de en los cuales fueron enviados las claves y la contraseña.

Finalmente, al tiempo que se obtiene y se almacena en la estructura start hay que aplicarle la misma operación que se tiene en la STM a la hora de asignar los segundos a la estructura **tiemval**, ya que con esto ambos sistemas están sincronizados tanto tiempo con en el GMT.

- **Código STM** En el apartado de la tarjeta se pudo observar por medio de la herramienta de depuración que pose el IDE que a pesar de que el valor que llegaba en la función era el mismo que se mostraba en la consola de programa del PC a la hora de que se asignaba en la estructura de tiempo **tiempo_actual** el valor era diferente, esto se daba a que el valor que se asignaba a la estructura de tiempo era los segundos adaptado al GMT de Colombia, también observamos que antes de aplicar el cálculo para adaptar el GMT a Colombia la hora que se mostraba en la LCD par poder confirmar una sincronización exitosa estaba 5 horas adelantado.

  Finalmente, para este apartado, ya que no se usa en ningún momento llamado de función en el main debido a que todo se ejecutará o por la función que recibe datos del puerto serial o en la ISR se evitó el uso de la función **lcd_clear**, esto se hizo con el fin de evitar delays dados por la librería porque tanto en la ISR con en la función que recibe los datos no se deben usar estos últimos la opción que se usó par poder limpiar la pantalla especialmente en la ISR fue poner un sting vacío de tamaño 16 en ambas líneas cada vez que se iba a poner el nuevo token para posteriormente pone el token calculado y así evitar que el token visto en la LCD estuviera incorrecto por culpa de que se quedó algún valor de otro token pegado en la pantalla.

## V. CONCLUSIONES

- la función de casteo en el lenguaje c solo sirven para poder evitar warnings, pero esta no tendrá ningún efecto sobre una variable, es decir que si se tiene un variable char por el hecho de castearla, esto no hará que la variable ahora posea el tipo del casteo.

- A la hora de sincronizar ambos sistemas no solo es necesario tener el valor del UNIX, también es importante tener en cuenta el valor del GMT según en el país que se encuentra.

- En la función de la ISR como en la función que se recibe datos no se deben tener ni interrupciones ni procesos demasiado largos porque estos pueden afectar a la hora de hacer la sincronización debido a retrasos.

- Se debe tener en cuenta que a la hora de iniciar los TIM's no es lo mismo si se quiere iniciar un Tim para que se ejecute un ISR cada x tiempo, para este caso se usaría el siguiente comando **HAL_TIM_Base_Start_IT(htim1)**, en caso contrario si se quiere usar el Tim como contador únicamente se inicia de la siguiente manera **HAL_TIM_Base_Start(htim2)**.

- Debido a que el IDE **STM32CUBE** no soporta las funciones **gettimeofday** y **settimeofday**, se debió buscar otra forma de poder mantener el reloj de la stm funcionando, esto se logró por medio de estar aumentando el valor de la estructura de tiempo **tv_sec**.

## REFERENCES


[1] U. technology. Usb-if technologies. [Online]. Available: https://www.usb.org/technologies

[2] Wikipedia. Exclusive or. [Online]. Available: https://en.wikipedia.org/wiki/Exclusive_or

[3] Entrust. Proveedores de identidad ¿quÉ es la autenticaciÓn basada en tokens? [Online]. Available: https://www.entrust.com/es/resources/faq/what-is-token-based-authentication

[4] G. cloud. Establece un tiempo de vencimiento prolongado para los tokens de oauth. [Online]. Available: https://cloud.google.com/apigee/docs/api-platform/antipatterns/oauth-long-expiration?hl=es-419#:~:text=El%20token%20de%20actualizaci%C3%B3n%20se,generar%20864%2C000%20tokens%20por%20d%C3%ADa.

[5] Arduino. Serial. [Online]. Available: https://www.arduino.cc/reference/en/language/functions/communication/serial/

[6] xanthium enterprises. Serial port programming using win32 api. [Online]. Available: https://www.xanthium.in/Serial-Port-Programming-using-Win32-API


## VI. ANEXOS

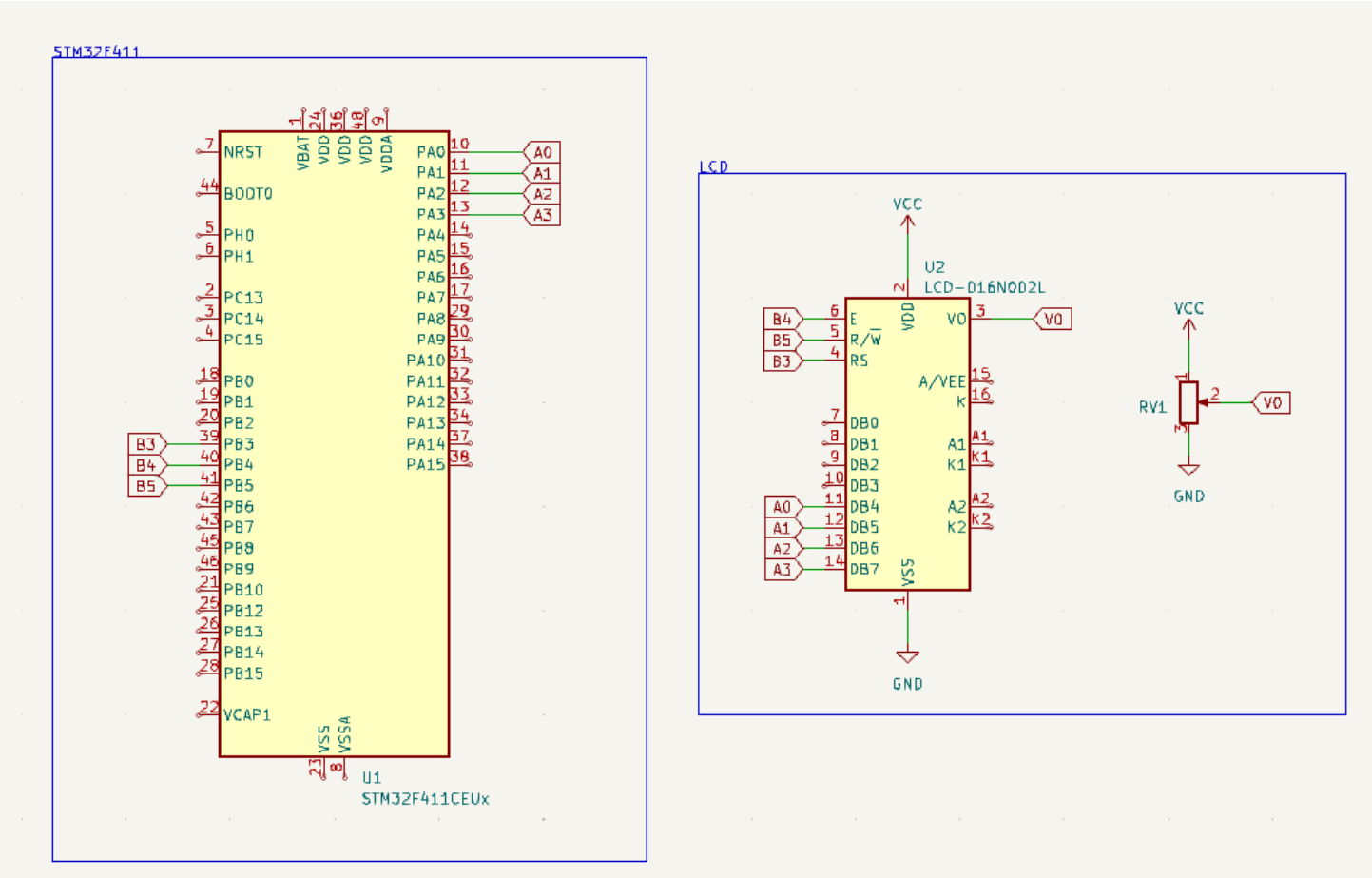


Figure 2: Esquemático