Documentación nextion_comunication_lib

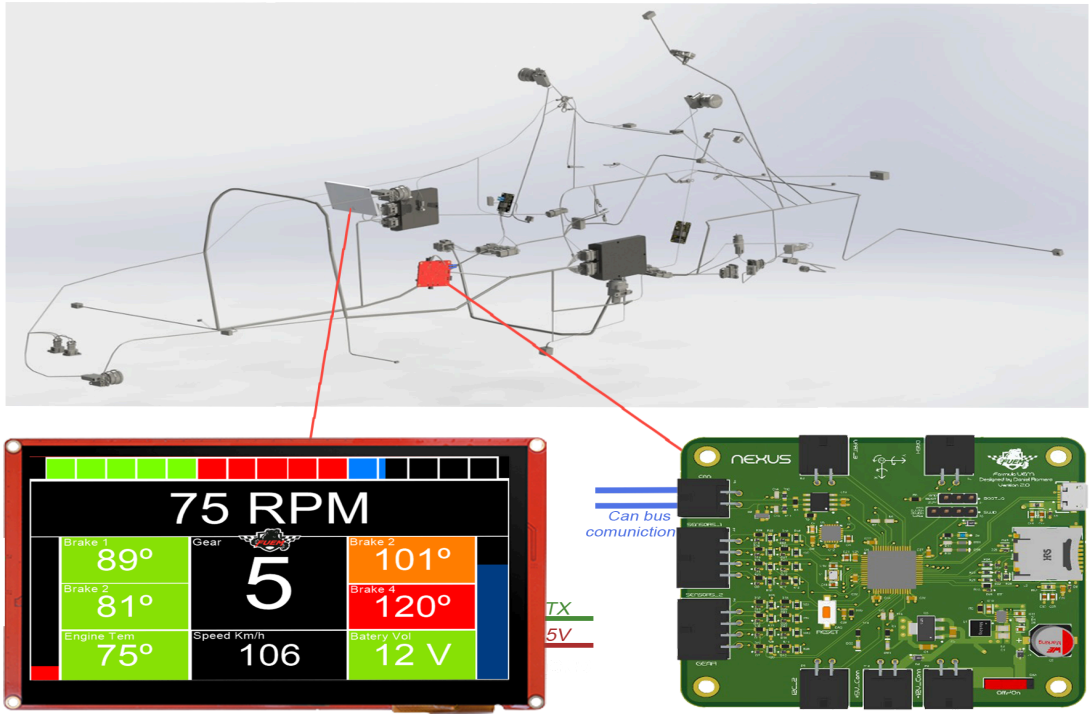
Por Daniel Gutiérrez Torres

Índice de contenido

1. Sobre la librería

Esta librería se ha desarrollado para la comunicación entre una interfaz diseñada para una pantalla Nextion (NX4827P043) y una placa de desarrollo STM32. Su objetivo es enviar y mostrar en tiempo real información relevante al piloto sobre la velocidad, la marcha y las revoluciones del coche, así como la temperatura de los frenos, la temperatura del motor, el voltaje de la batería y la aceleración y frenado del coche durante la conducción.

Además, permite mostrar alertas de forma visual y sencilla al piloto cuando es necesario detener el monoplaza debido a lecturas peligrosas de los sensores del vehículo, como una temperatura muy elevada del motor o un voltaje de la batería inadecuado.

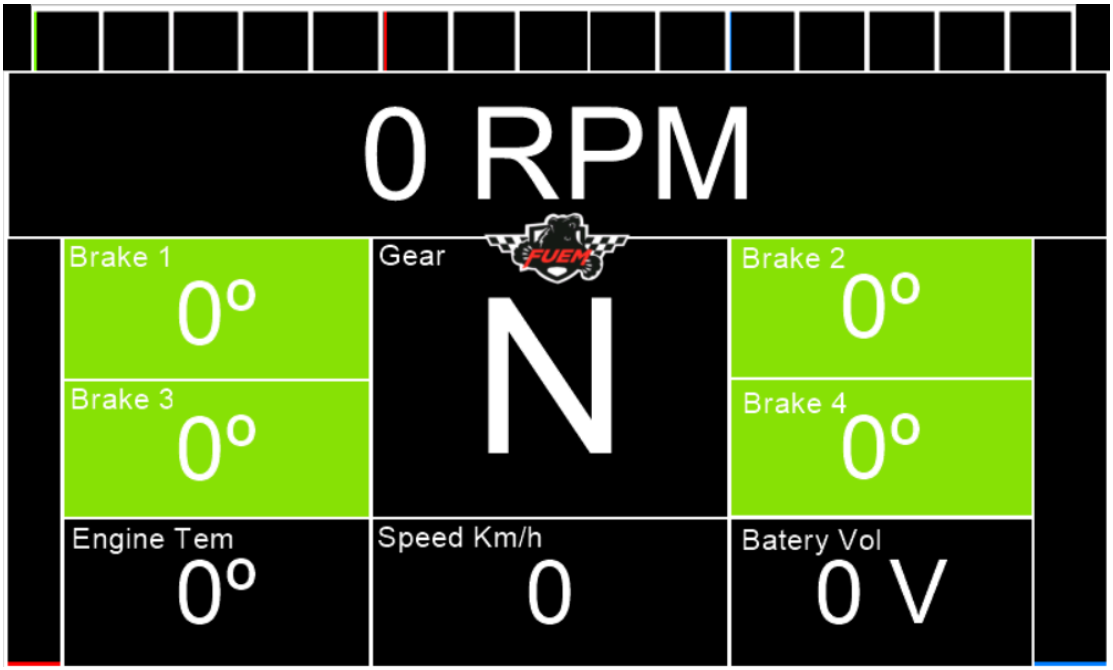


Pantalla Nextion

STM32f103r8t6 NEXUS

2. Interfaz utilizada

A continuación se muestra un ejemplo de interfaz desarrollada para una pantalla Nextion (NX4827P043). En ella podemos visualizar los diferentes campos que, mediante el uso de las diferentes funciones que la librería contiene, permitirán actualizar los datos de temperatura de frenos, velocidad, revoluciones, temperatura de motor y voltaje entre otros.



3. Precondiciones para utilizar la librería

En primer lugar antes de emplear la biblioteca, es esencial configurar y activar una conexión UART. Este es un protocolo de comunicación serial que facilita la transferencia de datos entre dispositivos electrónicos de forma directa. Este protocolo se utilizará para la transferencia de datos entre la pantalla y la placa seleccionada. Es en consecuencia fundamental contar con una placa compatible con UART, comprender los pines RX y TX utilizados para esta comunicación, ajustar correctamente la velocidad de transmisión (baud rate) y tener familiaridad con la inicialización y uso de la comunicación UART en Arduino.

A continuación se muestra como inicializar una comunicación UART, dentro del bloque main de nuestro código:

1-Inicialización de la comunicación UART:

```
MX_USART1_UART_Init();
```

2-Definimos la estructura para almacenar la configuración del UART:

```
UART_HandleTypeDef huart1;
```

En segundo lugar debe de verificarse que el Baud Rate establecido para la comunicación entre pantalla y placa sean los mismos.

baud = 115200;

4. Funciones:

A continuación se muestran el conjunto de funciones implementadas en la librería, las cuales pueden ser importadas y utilizadas para:

4.1 Actualizar valores en la interfaz

4.1.1 Actualizar Voltaje la batería

Parámetros:

- huart utilizado
- char valor_voltage[20]

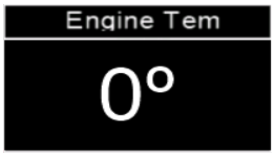
Llamada a la función:

```
NEXTION_SendText(&huart1, "voltage", valor_voltafe, "V");
```

4.1.2 Actualizar Temperatura del motor

Parámetros:

- huart utilizado
- char valor_engineTemp[20]



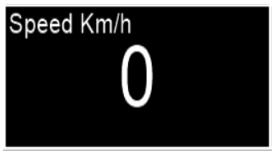
Llamada a la función:

```
NEXTION_SendText(&huart1,"engineTemp",valor_engineTemp," RPM");
```

4.1.3 Actualizar Velocidad

Parámetros:

- huart utilizado
- char valor_de_la_velocidad[20]



Llamada a la función:

```
NEXTION_SendText(&huart1, "speed", valor_velocidad, NULL);
```

4.1.4 Actualizar revoluciones

Parámetros:

- huart utilizado
- char valor_revValue[20]

Llamada a la función:

```
NEXTION_SendText(&huart1, "revValue",  valor_revValue, " RPM");
```

4.1.5 Actualizar Indicadores de revoluciones

Parámetros:

- huart utilizado
- int revs_value

Llamada a la función:

```
NEXTION_Send_Revs(&huart1, revs_value);
```

4.1.6 Actualizar Marcha

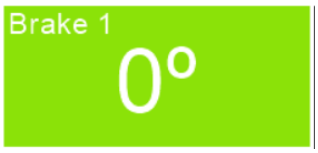
Parámetros:

- huart utilizado
- char valor_gear[20]

Llamada a la función:

```
NEXTION_SendText&huart1, "gear",  valor_gear, NULL);
```

4.1.7 Actualizar Freno 1



Parámetros:

- huart utilizado
- char valor_break1[20]

Llamada a la función:

```
NEXTION_SendText(&huart1, "brake1",  valor_break1, "\xB0");
```

4.1.8 Actualizar Freno 2

Parámetros:

- huart utilizado
- char valor_break2[20]

Llamada a la función:

```
NEXTION_SendText(&huart1, "brake2",  valor_break2, "\xB0");
```

4.1.9 Actualizar Freno 3

Parámetros:

- huart utilizado
- char valor_break3[20]

Llamada a la función:

```
NEXTION_SendText(&huart1, "brake3",  valor_break3, "\xB0");
```

4.1.10 Actualizar Freno 4

Parámetros:

- huart utilizado
- char valor_break4[20]

Llamada a la función:

```
NEXTION_SendText(&huart1, "brake4",  valor_break4, "\xB0");
```

4.1.11 Actualizar Barra de freno

Parámetros:

- huart utilizado
- char valor_break4[20]

Llamada a la función:

```
NEXTION_SendNumber(&huart1, "brakePedal", random_value);
```

4.1.12 Actualizar Barra de aceleración

Parámetros:

- huart utilizado
- int aceleration_value

Llamada a la función:

```
NEXTION_SendNumber(&huart1, "acePedal", aceleration_value);
```

4.2 Actualizar color los componentes de la interfaz

4.2.1 Actualizar el color de fondo de un componente

Esta función puede utilizarse cuando se desee mostrar de forma visual y sencilla el estado de un sensor del coche en la interfaz. Por ejemplo pueden definirse unos rangos de temperatura par la temperatura del motor

4.2.1.3 Cambiar color de fondo de un componente a verde :

Parámetros:

- huart utilizado
- componente_a_actualizar_fondo

Llamada a la función:

```
NEXTION_estado_color(&huart1, "componente", 36609);
```

4.2.1.2 Cambiar color de fondo de un componente a naranja:

Parámetros:

- huart utilizado
- componente_a_actualizar_fondo

Llamada a la función:

```
NEXTION_estado_color(&huart1, "componente", 64520);
```

4.2.1.3 Cambiar color de fondo de un componente a rojo:

Parámetros:

- huart utilizado
- componente_a_actualizar_fondo

Llamada a la función:

```
NEXTION_estado_color(&huart1, "componente", 63488);
```

4.2.2 Actualizar el color de fondo de todos los componentes del dash

Esta función puede utilizarse para mostrar de forma visual y sencilla al piloto que debe detener el coche en consecuencia de estar obteniendo lecturas de los sensores del coche peligrosas, como por ejemplo una temperatura muy elevada del motor.

Parámetros: huart utilizado
int código_color = 63488 // color rojo

Llamada a la función:

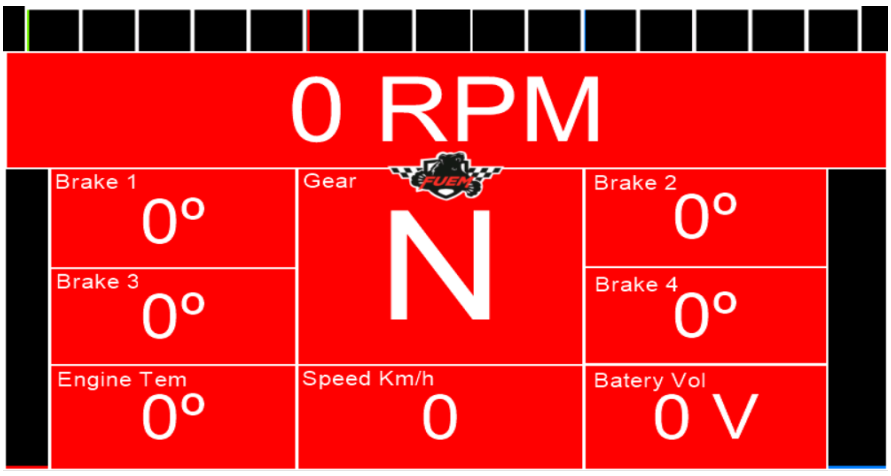
```
NEXTION_Alert(&huart1, código_color);
```

Ejemplo de uso:

Si la temperatura del motor es superior a 95º se cambia el color de todos los componentes de la interfaz a rojo

```
if (valor_engine_temp > 95) {
NEXTION_estado_color(&huart1, "engineTemp", 63488);
}
```

Visualización:



4.2.3 Cambio de interfaz mostrada en la pantalla

Esta función puede utilizarse cuando se desee cambiar por ejemplo de la interfaz de carga a la interfaz con el dash

Parámetros:

- huart utilizado
- página_en_la_interfaz_de_Nextion_a_la_que_cambiar

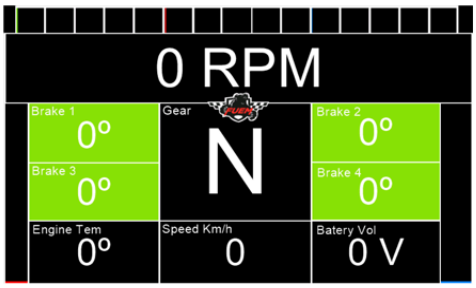
Llamada a la función:

```
NEXTION_SendPageChange(&huart1,"page0");
```

Visualización:

Page 1: Landing page

Page 2: Dash page



5. Desarrollo de las funciones:

A continuación se muestran y desarrollan las diferentes funciones que la librería implementa.

5.1- Función para actualizar el valor de Etiquetas en la interfaz

Esta función se utiliza para actualizar los valores de la temperatura de los frenos y del motor, el voltaje de la batería y el número de revoluciones, pasándole como parámetro el valor medido por el sensor en cuestión, convertido a texto con el valor y las unidades asociadas al componente correspondiente. Además, se especifica el componente al que se le quiere actualizar el valor y el UART que utilizamos para comunicarnos con la interfaz y la placa.

```
void NEXTION_SendText(UART_HandleTypeDef *huart, char *obj, char *text, char
*units) {
  // Reserva memoria para un buffer de 50 bytes
  uint8_t *buffer = malloc(50 * sizeof(char));
  int len= 0;
  if (units == NULL || units[0] == '\0') {
    // Agregar el texto al objeto
    len= sprintf((char *)buffer, "%s.txt=\"%s\"", obj, text);
  } else {
    // Agrega las unidades al texto del objeto
    len= sprintf((char *)buffer, "%s.txt=\"%s%s\"", obj,
    text, units);
  }
  // Transmite el buffer a través de UART
  HAL_UART_Transmit(huart, buffer, len, 1000);
  // Transmite Cmd_End para indicar que finalizó el mensaje
  HAL_UART_Transmit(huart, Cmd_End, 3, 100);
  // Libera la memoria asignada al buffer
  free(buffer);
}
```

5.2 - Función para actualizar el valor de Barras de progreso en la interfaz

Esta función se utiliza para actualizar la barra de frenado, la barra de aceleración y las barras de los indicadores de revoluciones con un valor entero entre 0 y 100. Para ello, se le pasan como parámetros uno de los componentes mencionados, el valor numérico ajustado dentro del rango mencionado y el UART que utilizamos para comunicarnos con la interfaz y la placa.

```
void NEXTION_SendNumber(UART_HandleTypeDef *huart, char *obj, int number) {
  // Reserva memoria para un buffer de 50 bytes
  uint8_t *buffer = malloc(50 * sizeof(char));
  // Inicializa el buffer con el objeto y el valor a inicializar
  int len = sprintf((char *)buffer, "%s.val=%d", obj, number);
  // Transmite el buffer a través de UART
  HAL_UART_Transmit(huart, buffer, len, 1000);
  // Transmite Cmd_End para indicar que finalizó el mensaje
  HAL_UART_Transmit(huart, Cmd_End, 3, 100);
  // Libera la memoria asignada al buffer
  free(buffer);
}
```

5.3 - Función para cambiar la interfaz de la pantalla

Esta función permite cambiar la interfaz mostrada en la pantalla de forma rápida pasandole como parámetro la página (interfaz) que se desea mostrar y el huart que utilizamos para comunicarnos con la interfaz y la placa..

```
void NEXTION_SendPageChange(UART_HandleTypeDef *huart,char *page_name) {
  // Reserva memoria para un buffer de 50 bytes
  uint8_t *buffer = malloc(50 * sizeof(char));
  // Inicializa el buffer con instrucción de cambiar de página
  int len = sprintf((char *)buffer, "page %s", page_name);
  // Transmite el buffer a través de UART
  HAL_UART_Transmit(huart, buffer, len, 1000);
  // Transmite un comando para indicar el final del mensaje
  HAL_UART_Transmit(huart, Cmd_End, 3, 100);
  // Libera la memoria asignada al buffer
  free(buffer);
}
```

5.4 - Función para actualizar el color de un componente de la interfaz

Esta función permite cambiar la propiedad BCO (color de fondo de un componente) pasándole como parámetro el componente al que se le quiere cambiar el color, el código de color del color al que se queire cambiar el fondo del componente y el huart que utilizamos para comunicarnos con la interfaz y la placa.

```
void NEXTION_estado_color(UART_HandleTypeDef *huart, char *obj, int color) {
  uint8_t *buffer = malloc(50 * sizeof(char));
  // Formatea y transmite el mensaje para el elemento actual
  int len = sprintf((char *)buffer, "%s.bco=%d", obj, color);
HAL_UART_Transmit(huart, buffer, len, 1000);
HAL_UART_Transmit(huart, Cmd_End, 3, 100);
// Libera el buffer
free(buffer);
}
```

Dentro del switch usamos una serie de condiciones para determinar cuándo y a qué color cambiar el componente

```
case 0x647:
    NEXTION_SendText(UART_HandleTypeDef *huart,"engineTemp", text, "\xB0");
    if (random_value > 0 && random_value <= 50) {
            NEXTION_Alert(huart,0); //black
            NEXTION_estado_color(huart, "engineTemp", 36609);
    } else if (random_value > 50 && random_value <= 80) {
            NEXTION_Alert(huart,0);
        NEXTION_estado_color(huart, "engineTemp", 64520);
    } else if (random_value > 91) {
            NEXTION_Alert(huart,63488); // red
            NEXTION_estado_color(huart, "engineTemp", 63488);
    }
   break;
```

5.5 - Función que actualiza el color de todos los componentes de la interfaz a rojo

Esta función cambia el color de todos los elementos contenidos en array_elementos_a_poner_rojo_por_alerta , pasándole como parámetros el color al que se quieren cambiar los elementos y el UART que utilizamos para comunicarnos con la interfaz y la placa, con el objetivo de mostrar de forma visual y sencilla la orden de detener el vehículo al piloto.

```
void NEXTION_Alert(UART_HandleTypeDef *huart,int color) {
  uint8_t *buffer = malloc(50 * sizeof(char));
  for (int i = 0; i < 7; i++) {
     // Formatea la propiedad bco (color) del elemento actual
      int len = sprintf((char *)buffer, "%s.bco=%d",
      array_elementos_a_poner_rojo_por_alerta[i], color);
     HAL_UART_Transmit(huart, buffer, len, 1000);
     HAL_UART_Transmit(huart, Cmd_End, 3, 100);
     // Libera el buffer
     free(buffer);
  }
}
```

5.6 Función para actualizar los indicadores de revoluciones versión 1

Esta función permite calcular la posición de las tres barras de progreso basandose en un valor de entrada de revoluciones recibido. Para ello divide este valor en tres rangos y para cada cada barra de progreso y a su vez a en 5 subpartes para la propia barra. Para ello hasta las 3000 revoluciones muestra el valor en la primera barra de progreso verde. A partir de 6000 muestra el valor en la primera barra de progreso verde y en la segunda barra de progreso roja. A partir de 9000 muestra el valor en la primera barra de progreso verde, de la segunda barra de progreso roja y de la tercera barra de progreso azul. Para ello recibe como parámetros el huart utilizado, además del valor de revoluciones a mostrar en la interfaz.

```
void NEXTION_Send_Revs(UART_HandleTypeDef *huart, int val) {
  int resultado1 = 0;
  int resultado2 = 0;
  int resultado3 = 0;
  if (val >= 0 && val < 3000) {
     resultado1 = val / 30.0; // Rango 0-3000
     resultado1 = (resultado1 + 10) / 20 * 20;
     resultado2 = 0;
     resultado3 = 0;
  } else if (val >= 3000 && val < 6000) {
     resultado1 = 100;
     resultado2 = (val - 3000) / 30.0; // Rango 3000-6000
     resultado2 = (resultado2 + 10) / 20 * 20;
     resultado3 = 0;
  } else if (val >= 6000 && val <= 9000) {
     resultado1 = 100;
     resultado2 = 100;
     resultado3 = (val - 6000) / 30.0; // Rango 6000-9000
     resultado3 = (resultado3 + 10) / 20 * 20;
  }
  // Envía los resultados a las barras correspondientes
  NEXTION_SendNumber(huart,"led1", resultado1);
  NEXTION_SendNumber(huart,"led2", resultado2);
```

5.7 Función para actualizar los indicadores de revoluciones versión 2

Esta segunda función optimizada permite emular las tres barras de progreso manejadas con la versión 1, las cuales han sido sustituidas por tres rectángulos de colores. En vez de calcular los valores de las 3 barras y a su vez el valor de sus 5 subconjuntos, esta únicamente cambia el color de los 3 rectángulos. Para ello hasta las 3000 revoluciones establece el color del primer rectángulo a verde y el resto negro. A partir de 6000 cambia el color del primer rectángulo a verde, del segundo a rojo y el tercero e negro. A partir de 9000 cambia el color del primer rectángulo a verde, del segundo a rojo y el tercero a azul. Para ello recibe como parámetros el huart utilizado, además del valor de revoluciones a mostrar en la interfaz.

```
void NEXTION_Send_Revs_v2(UART_HandleTypeDef *huart, int val) {
  int resultado1 = 0;
  int resultado2 = 0;
  int resultado3 = 0;
  // Check the value ranges and assign corresponding color values
  if (val >= 0 && val < 3000) {
     resultado1 = 32736; // Color green for range 0-3000
     resultado2 = 0;
     resultado3 = 0;
  } else if (val >= 3000 && val < 6000) {
     resultado1 = 32736; // Color red for range 3000-6000
     resultado2 = 63488;
     resultado3 = 0;
  } else if (val >= 6000 && val <= 9000) {
     resultado1 = 32736; // Color blue for range 6000-9000
     resultado2 = 63488;
     resultado3 = 1055;
  }
  // Send the color values to the corresponding LEDs on the Nextion display
  NEXTION_estado_color(huart, "led1", resultado1);
  NEXTION_estado_color(huart, "led2", resultado2);
  NEXTION_estado_color(huart, "led3", resultado3);
}
```