Documentación nextion_comunication_lib

Por Daniel Gutiérrez Torres

ÍNDICE DE CONTENIDOS

CAPÍTULO 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.

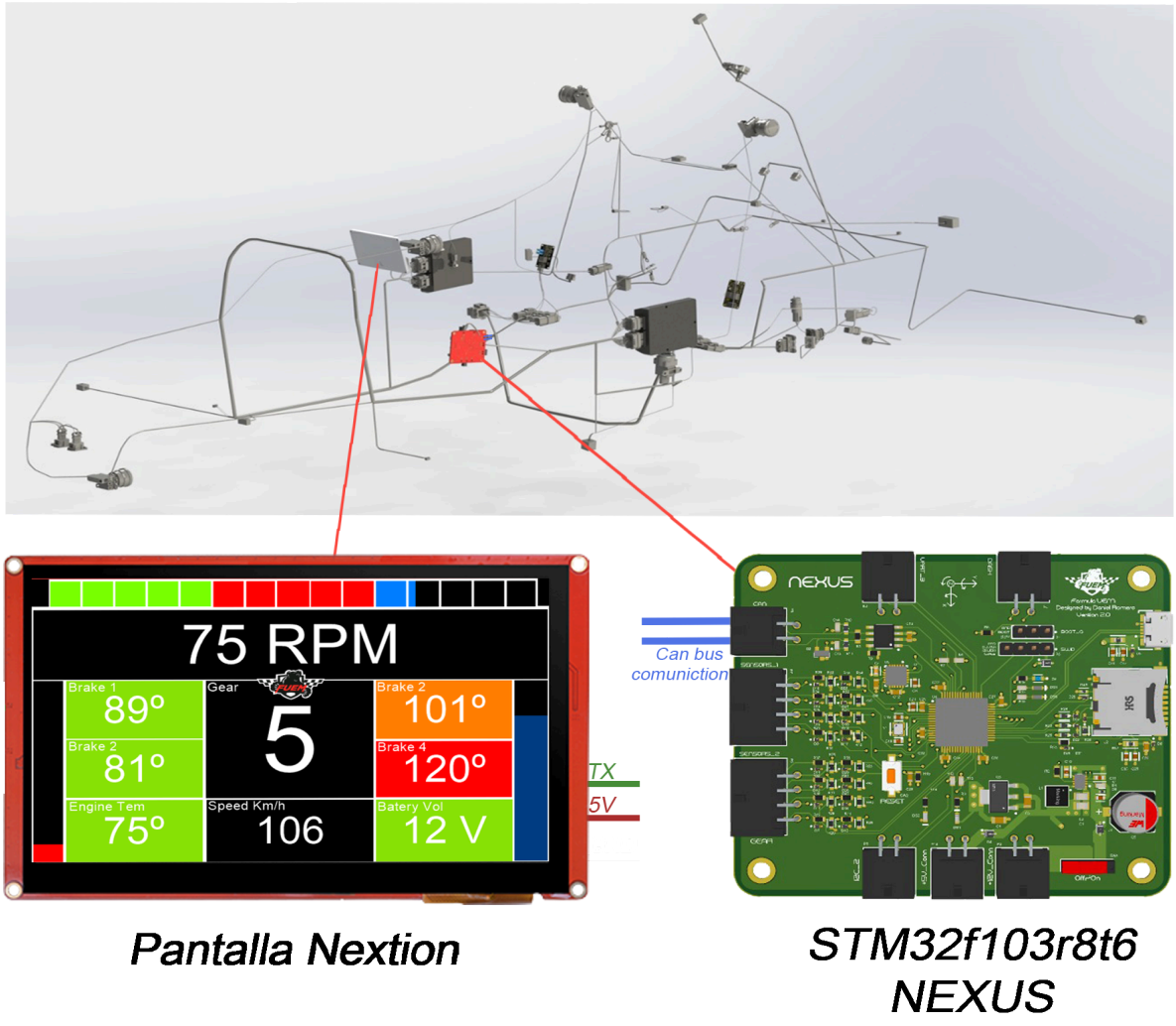

Figura 1: Esquema de conexión de los componentes

CAPÍTULO 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.

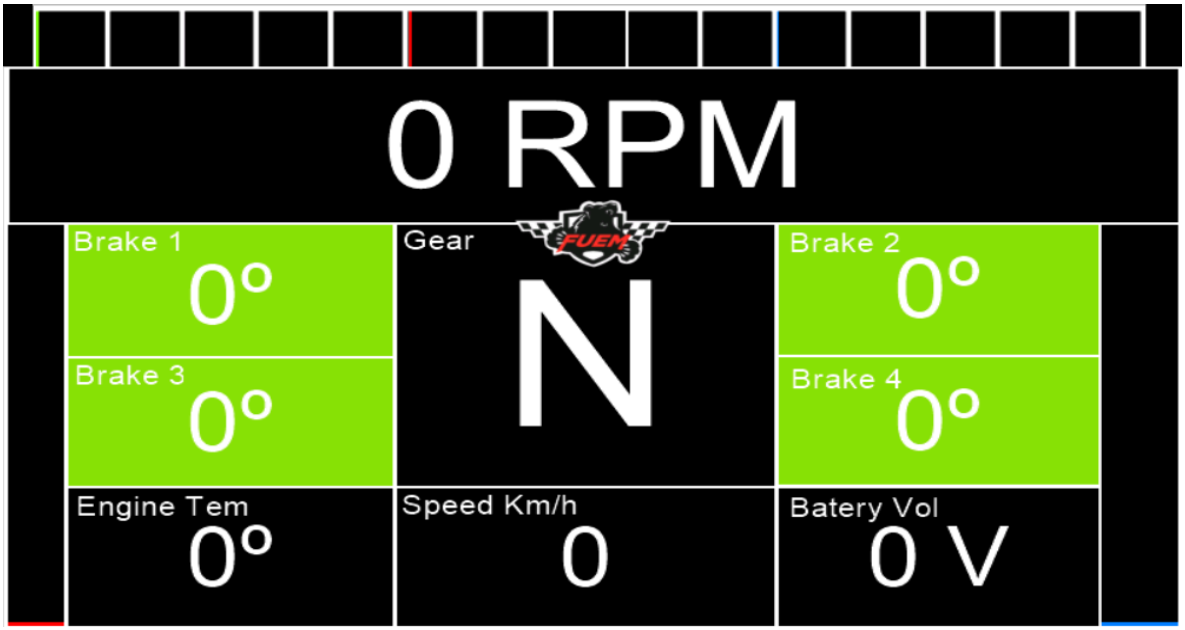

Figura 2: Interfaz del dashware versión 1 con indicadores de temperatura de los frenos.

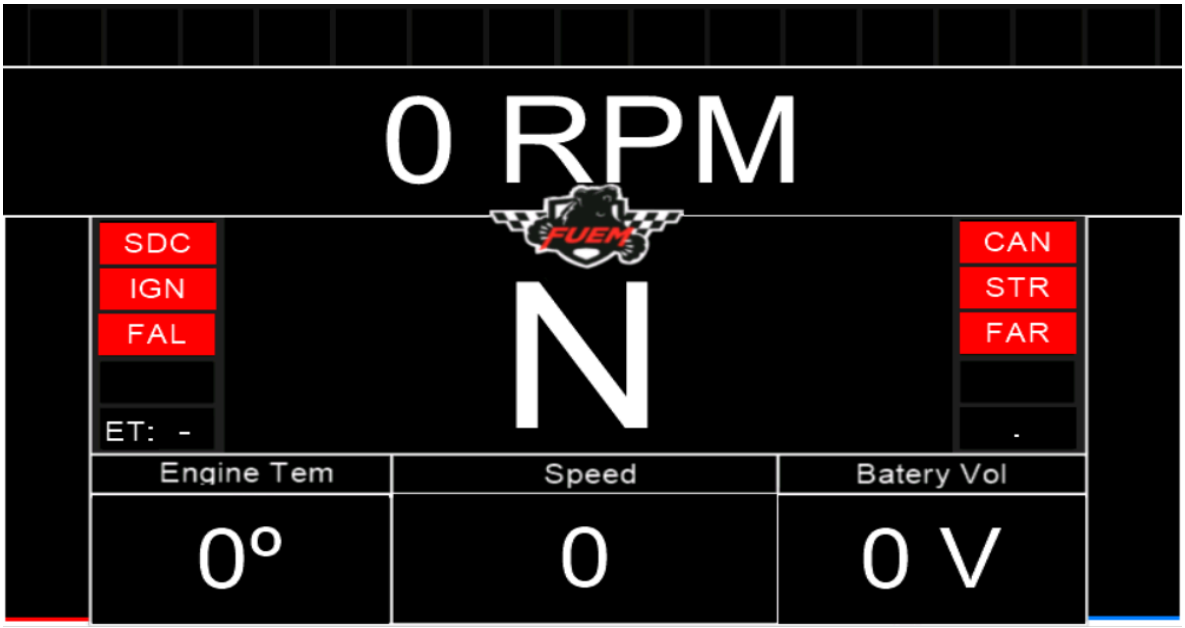

Figura 3: Interfaz del dashware versión 2 con testigos de estados se sistemas del coche.

CAPÍTULO 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 cómo inicializar la comunicación UART dentro del bloque principal de nuestro código. Primero, es necesario inicializar la comunicación UART llamando a la función `MX_USART1_UART_Init();`. Esta función configura los parámetros necesarios para la comunicación serial. Luego, debemos definir una estructura para almacenar la configuración del UART utilizando la declaración `UART_HandleTypeDef huart1;`. Esta estructura permite gestionar y acceder a la configuración de la interfaz UART durante la operación del programa.

En segundo lugar debe de verificarse que el Baud Rate establecido para la comunicación entre pantalla y placa sean los mismos: baud = 115200;.

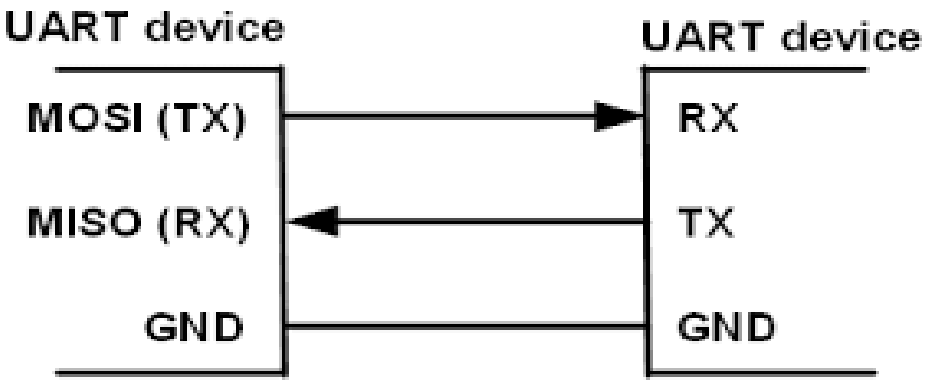

Figura 4: Representación comunicación huart

CAPÍTULO 4: FUNCIONES DE LA LIBRERÍA

4.1 Actualizar valores de componentes de la interfaz

4.1.1 Actualizar Voltaje la batería

- Llamada a la función: NEXTION_SendText(&huart1, "voltage", valor_voltage, "V");
- Parámetros utilizados:
    - huart utilizado
    - char valor_voltage[20]
- Elemento a actualizar en la interfaz:

Figura 5: Indicador de voltaje en la interfaz.

4.1.2 Actualizar Temperatura del motor

- Llamada a la función: NEXTION_SendText(&huart1,"engineTemp", valor_engineTemp," RPM");
- Parámetros utilizados:
    - huart utilizado
    - char valor_de_la_velocidad[20]
- Elemento a actualizar en la interfaz:

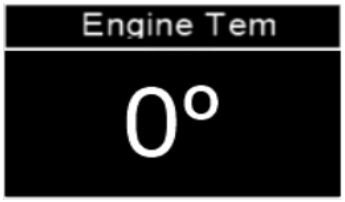

Figura 6: Indicador de temperatura del motor en la interfaz.

4.1.3 Actualizar Velocidad

- Llamada a la función: NEXTION_SendText(&huart1, "speed", valor_velocidad, NULL);
- Parámetros utilizados:
    - huart utilizado
    - char valor_de_la_velocidad[20]
- Elemento a actualizar en la interfaz:

Figura 7: Indicador de velocidad en la interfaz.

4.1.4 Actualizar revoluciones

- Llamada a la función: NEXTION_SendText(&huart1, "revValue", valor_revValue, " RPM");
- Parámetros utilizados:
  - huart utilizado
  - char valor_revValue[20]
- Elemento a actualizar en la interfaz:

Figura 8: Indicador numérico de revoluciones en la interfaz.

4.1.5 Actualizar Indicadores de revoluciones

- Llamada a la función: NEXTION_Send_Revs(&huart1, revs_value);
- Parámetros utilizados:
  - huart utilizado
  - char revs_value[20]
- Elemento a actualizar en la interfaz:

Figura 9: Indicador led de revoluciones en la interfaz.

4.1.6 Actualizar Marcha

- Llamada a la función: NEXTION_SendText(&huart1, "gear", valor_gear, NULL);
- Parámetros utilizados:
  - huart utilizado

- char valor_de_la_velocidad[20]

- Elemento a actualizar en la interfaz:

Figura 10: Indicador de marcha en la interfaz.

4.1.7 Actualizar Freno 1

- Llamada a la función: NEXTION_SendText(&huart1, "brake1", valor_break1, "\xB0");
- Parámetros utilizados:
  - huart utilizado
  - char valor_temp_break2[20]
- Elemento a actualizar en la interfaz:

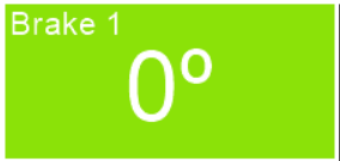

Figura 11: Indicador de temperatura del freno 1 en la interfaz.

4.1.8 Actualizar Freno 2

- Llamada a la función: NEXTION_SendText(&huart1, "brake2", valor_break2, "\xB0");
- Parámetros utilizados:
  - huart utilizado
  - char valor_temp_break2[20]
- Elemento a actualizar en la interfaz:

Figura 12: Indicador de temperatura del freno 2 en la interfaz.

4.1.9 Actualizar Freno 3

- Llamada a la función: NEXTION_SendText(&huart1, "brake3", valor_break3, "\xB0");
- Parámetros utilizados:
  - huart utilizado
  - char valor_temp_break3[20]
- Elemento a actualizar en la interfaz:

Figura 13: Indicador de temperatura del freno 3 en la interfaz.

4.1.10 Actualizar Freno 4

- Llamada a la función: NEXTION_SendText(&huart1, "brake4", valor_break4, "\xB0");
- Parámetros utilizados:
  - huart utilizado
  - char valor_temp_break4[20]
- Elemento a actualizar en la interfaz:

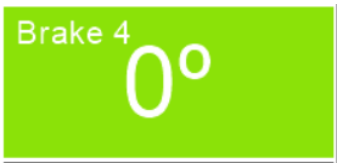

Figura 14: Indicador de temperatura del freno 4 en la interfaz.

4.1.11 Actualizar indicador temperatura ambiente

- Llamada a la función: NEXTION_SendText(&huart1, "ambtemp", valor_ambtemp, "\xB0");
- Parámetros utilizados:
  - huart utilizado
  - char valor_temp_break4[20]
- Elemento a actualizar en la interfaz:

Figura 15: Indicador de temperatura ambiente.

4.1.12 Actualizar Barra de freno

- Llamada a la función: NEXTION_SendNumber(&huart1, "brakePedal", random_value);
- Parámetros utilizados:
  - huart utilizado
  - char valor_temp_break4[20]
- Elemento a actualizar en la interfaz:

Figura 16: Barra de frenado de la interfaz.

4.1.13:Actualizar Barra de aceleración

- Llamada a la función: NEXTION_SendNumber(&huart1, "acePedal", aceleration_value);
- Parámetros utilizados:
  - huart utilizado
  - char valor_temp_break4[20]
- Elemento a actualizar en la interfaz:

Figura 17: Indicador de temperatura del freno 4 en la interfaz.

4.2 Actualizar el color de un único componente de la interfaz

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 colores para los siguientes componentes:

4.2.1 Actualizar indicador de estado de la Comunicación CAN BUS

- Llamada a la función si la comunicación CAN BUS transmite datos correctamente: NEXTION_estado_color(&huart1, "can", 1024);
- Llamada a la función si la comunicación CAN BUS NO transmite datos correctamente: NEXTION_estado_color(&huart1, "can", 63488);
- Elemento a actualizar en la interfaz:

Figura 18: Estados del indicador del funcionamiento de la comunicación CAN BUS en la interfaz.

- Parámetros utilizados:
  - huart utilizado
  - int código_de_color[20] (63488-> rojo, 1024->verde)

4.2.2 Actualizar indicador de estado del Ignition

- Llamada a la función si el estado del ignition esta activado: NEXTION_estado_color(&huart1, "ign", 1024);
- Llamada a la función si el estado del ignition No esta activado: correctamente: NEXTION_estado_color(&huart1, "ign", 63488);
- Parámetros utilizados:
  - huart utilizado
  - int código_de_color[20] (63488-> rojo, 1024->verde)
- Elemento a actualizar en la interfaz:

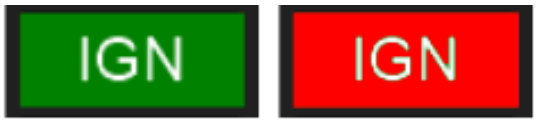

Figura 19: Estados del indicador del correcto funcionamiento de Ignition en la interfaz.

4.2.3 Actualizar indicador de estado del SDC

- Llamada a la función si el estado del SDC esta activado: NEXTION_estado_color(&huart1, "sdc", 1024);
- Llamada a la función si el estado del SDC NO esta activado: correctamente: NEXTION_estado_color(&huart1, "sdc", 63488);
- Elemento a actualizar en la interfaz:

Figura 20: Estados del indicador de funcionamiento del sistema SDC en la interfaz.

- Parámetros utilizados:
  - huart utilizado
  - int código_de_color[20] (63488-> rojo, 1024->verde)

4.2.4 Actualizar indicador de estado del STR

- Llamada a la función si el estado del STR esta activado: NEXTION_estado_color(&huart1, "sdc", 1024);
- Llamada a la función si el estado del STR NO esta activado: correctamente: NEXTION_estado_color(&huart1, "str", 63488);
- Elemento a actualizar en la interfaz:

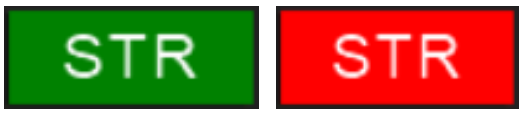

Figura 21: Estados del indicador de funcionamiento del sistema STR en la interfaz.

- Parámetros utilizados:
  - huart utilizado
  - int código_de_color[20] (63488-> rojo, 1024->verde)

4.2.5 Actualizar indicador de estado del ventilador derecho

- Llamada a la función si el ventilador derecho está funcionando: NEXTION_estado_color(&huart1, "fanRight", 1024);
- Llamada a la función si el estado del STR NO esta activado: correctamente: NEXTION_estado_color(&huart1, "fanRight", 63488);
- Elemento a actualizar en la interfaz:

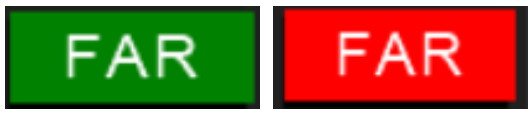

Figura 22: Estados del indicador de funcionamiento del ventilador derecho en la interfaz.

- Parámetros utilizados:
  - huart utilizado
  - int código_de_color[20] (63488-> rojo, 1024->verde)

4.2.6 Actualizar indicador de estado del ventilador izquierdo

- Llamada a la función si el ventilador izquierdo está funcionando: NEXTION_estado_color(&huart1, "sdc", 1024);

- Llamada a la función si el ventilador izquierdo NO está funcionando: correctamente: NEXTION_estado_color(&huart1, "sdc", 63488);
- Elemento a actualizar en la interfaz:

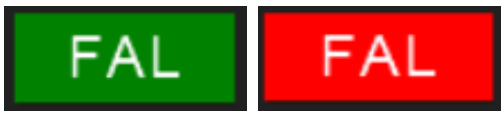

Figura 23: Estados del indicador de funcionamiento ventilador izquierdo en la interfaz.

- Parámetros utilizados:
  - huart utilizado
  - int código_de_color[20] (63488-> rojo, 1024->verde)

4.2.7 Actualizar el color de fondo del freno 1

- Llamada a la función si la temperatura de los frenos es adecuada: NEXTION_estado_color(&huart1, "breake1", 1024);
- Llamada a la función si la temperatura de los frenos es MEDIANAMENTE adecuada: NEXTION_estado_color(&huart1, "breake1", 64520);
- Llamada a la función si la temperatura de los frenos NO es adecuada: NEXTION_estado_color(&huart1, "breake1", 63488);
- Parámetros:
  - huart utilizado
  - int código_de_color[20] (63488-> rojo, 1024->verde, naranja->64520)
- Elemento a actualizar en la interfaz:

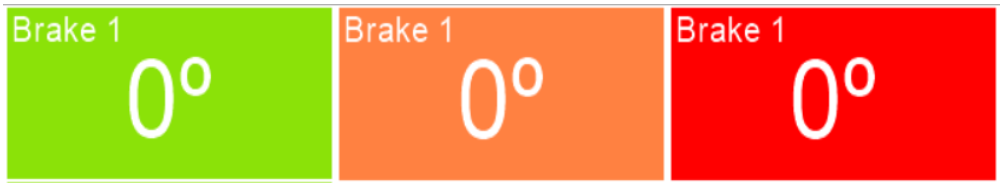

Figura 24: Estados de colores del freno 1.

4.2.8 Actualizar el color de fondo del freno 2

- Llamada a la función si la temperatura de los frenos es adecuada: NEXTION_estado_color(&huart1, "breake2", 1024);
- Llamada a la función si la temperatura de los frenos es MEDIANAMENTE adecuada: NEXTION_estado_color(&huart1, "breake2", 64520);
- Llamada a la función si la temperatura de los frenos NO es adecuada: NEXTION_estado_color(&huart1, "breake2", 63488);
- Parámetros:
  - huart utilizado
  - int código_de_color[20] (63488-> rojo, 1024->verde, naranja->64520)

- Elemento a actualizar en la interfaz:

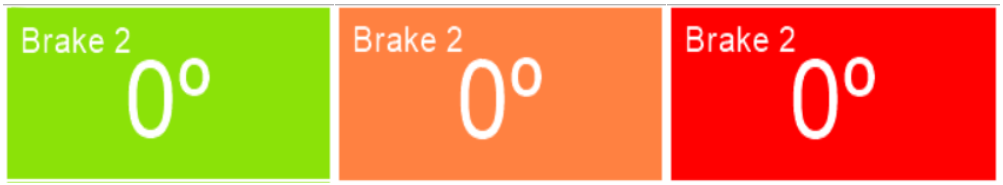

Figura 25: Estados de colores del freno 2.

4.2.9 Actualizar el color de fondo del freno 3

- Llamada a la función si la temperatura de los frenos es adecuada: NEXTION_estado_color(&huart1, "breake3", 1024);
- Llamada a la función si la temperatura de los frenos es MEDIANAMENTE adecuada: NEXTION_estado_color(&huart1, "breake3", 64520);
- Llamada a la función si la temperatura de los frenos NO es adecuada: NEXTION_estado_color(&huart1, "breake3", 63488);
- Parámetros:
  - huart utilizado
  - int código_de_color[20] (63488-> rojo, 1024->verde, naranja->64520)
- Elemento a actualizar en la interfaz:

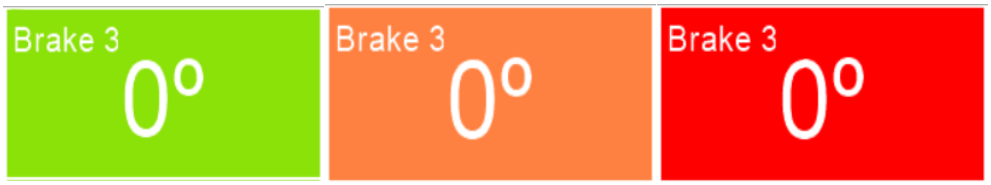

Figura 26: Estados de colores del freno 3.

4.2.10 Actualizar el color de fondo del freno 4

- Llamada a la función si la temperatura de los frenos es adecuada: NEXTION_estado_color(&huart1, "breake4", 1024);
- Llamada a la función si la temperatura de los frenos es MEDIANAMENTE adecuada: NEXTION_estado_color(&huart1, "breake4", 64520);
- Llamada a la función si la temperatura de los frenos NO es adecuada: NEXTION_estado_color(&huart1, "breake4", 63488);
- Parámetros:
  - huart utilizado
  - int código_de_color[20] (63488-> rojo, 1024->verde, naranja->64520)

- Elemento a actualizar en la interfaz:

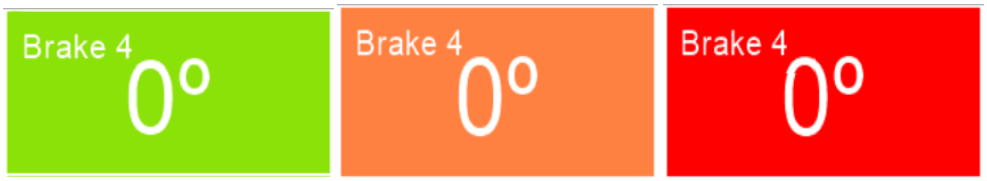

Figura 27: Estados de colores del freno 4.

4.2.11 Actualizar el color de fondo del voltaje de la bateria

- Llamada a la función si el voltaje de la bateria es adecuado: NEXTION_estado_color(&huart1, "voltage", 1024);
- Llamada a la función si el voltaje de la bateria es MEDIANAMENTE adecuada: NEXTION_estado_color(&huart1, "voltage", 64520);
- Llamada a la función si el voltaje de la bateria NO es adecuada: NEXTION_estado_color(&huart1, "voltage", 63488);
- Parámetros:
  - huart utilizado
  - int código_de_color[20] (63488-> rojo, 1024->verde, naranja->64520)
- Elemento a actualizar en la interfaz:

Figura 28: Estados de colores del voltaje de la batería

4.2.12 Actualizar el color de fondo de la temperatura del motor

- Llamada a la función si la temperatura del motor es adecuado: NEXTION_estado_color(&huart1, "engineTemp", 1024);
- Llamada a la funciónsi la temperatura del motor es MEDIANAMENTE adecuada: NEXTION_estado_color(&huart1, "engineTemp", 64520);
- Llamada a la función si la temperatura del motor NO es adecuada: NEXTION_estado_color(&huart1, "engineTemp", 63488);
- Parámetros:
  - huart utilizado
  - int código_de_color[20] (63488-> rojo, 1024->verde, naranja->64520)

- Elemento a actualizar en la interfaz:

Figura 29: Estados de colores de la temperatura del motor.

4.3 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. 7

- 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 en la interfaz:

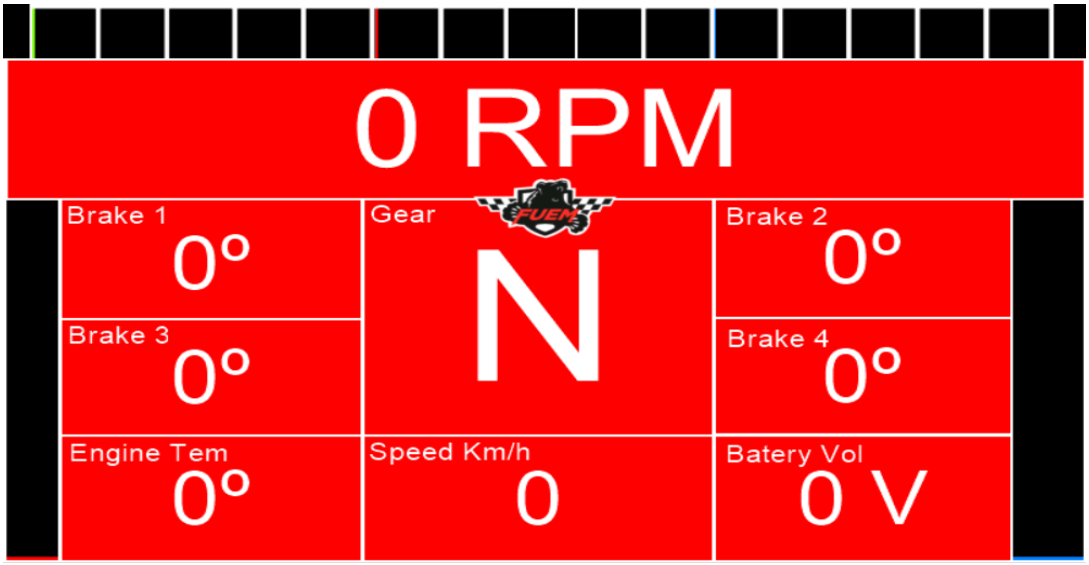

Figura 30:Indicador de alerta por anomalia en algún sensor en la pantalla.

4.4 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 o viceversa.

- Llamada a la función: NEXTION_SendPageChange(&huart1,"page0");
- Parámetros utilizados:
  - huart utilizado
  - página_en_la_interfaz_de_Nextion_a_la_que_cambiar
- Visualización en la interfaz:

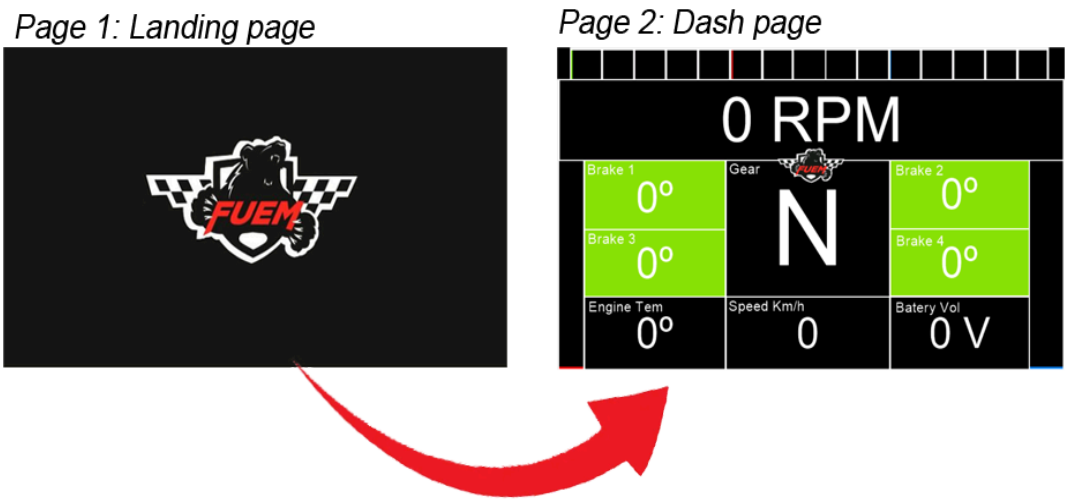

Figura 31: Transición de la interfaz de landing page a dash

CAPÍTULO 5: DESARROLLO DE FUNCIONES DE LA LIBRERÍA

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 las 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);
```