Documentación
nextion_comunication_lib

ÍNDICE

1. Resumen de llamadas disponibles

1.1- Actualizar valores en la interfaz

- Actualizar Velocidad

Parámetros: `huart utilizado`
char valor_de_la_velocidad[20]

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

- Actualizar Voltaje la batería

Parámetros: `huart utilizado`
char valor_voltage[20]

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

- Actualizar Temperatura del motor

Parámetros: `huart utilizado`
char valor_engineTemp[20]

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

- Actualizar revoluciones

Parámetros: `huart utilizado`
char valor_revValue[20]

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

- Actualizar Marcha

Parámetros: `huart utilizado`
char valor_gear[20]

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

-Actualizar Freno 1

Parámetros: `huart utilizado`
char valor_break1[20]

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

-Actualizar Freno 2

Parámetros: `huart utilizado`
char valor_break2[20]

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

-Actualizar Freno 3

Parámetros: `huart utilizado`
char valor_break3[20]

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

-Actualizar Freno 4

Parámetros: `huart utilizado`
char valor_break4[20]

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

-Actualizar Barra de freno

Parámetros: `huart utilizado`
char valor_break4[20]

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

-Actualizar Barra de aceleración

Parámetros: `huart utilizado`
int aceleration_value

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

-Actualizar Indicadores de revoluciones

Parámetros: `huart utilizado`
int revs_value

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

* código para parsear variables enteras a texto

```
char text[20]; // Declarar variable donde almacenar la conversión
sprintf(text, "%d", randoom_value); //Inicializar la conversión
```

1.2- Actualizar colores de componentes de la interfaz

1.2.1 Actualizar el color de fondo de un componente

Esta función puede utilizarse cuando se desee mostrar de forma visual y sencilla el estado de

Por ejemplo pueden definirse unos rangos de temperatura par la temperatura del motor

-Si la temperatura del motor

-Cambiar color de fondo de un componente a rojo:

Parámetros: `huart utilizado`
`componente_a_actualizar_fondo`

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

-Cambiar color de fondo de un componente a naranja:

Parámetros: `huart utilizado`
`componente_a_actualizar_fondo`

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

-Cambiar color de fondo de un componente a verde :

Parámetros: `huart utilizado`
`componente_a_actualizar_fondo`

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

1.2.2 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

```
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);
}
```

1.3- 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`

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

2. Explicación de funciones desarrolladas:

2.1- Función para actualizar objeto obj de la interfaz con un valor text

```
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);
}
```

2.2 - Función para actualizar objeto obj de la interfaz con un valor numérico

*Esta función se utiliza para actualizar la barra de frenado, la barra de aceleración y las barras de los indicadores de revoluciones.

```
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);
}
```

2.3 - Función para actualizar los indicadores de revoluciones

El método calcula la intensidad de tres barras de progreso basadas en un valor de entrada val. Divide val en tres rangos y asigna la intensidad a cada barra. Cada barra tiene un máximo de 100, dividido en 5 cuadrados para simular LEDs, por lo que el resultado se redondea a 20 para ajustarse a la medida de cada cuadrado.

```
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);
   NEXTION_SendNumber(huart,"led3", resultado3);
}
```

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

```
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);
   }
}
```

2.5 - Función para cambiar de la loading view a la interfaz con los indicadores

```
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);
}
```

2.6 - Función para cambiar el color de un objeto en específico

```
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(&huart1,"engineTemp", text, "\xB0");
      if (random_value > 0 && random_value <= 50) {
           NEXTION_Alert(&huart1,0); //black
           NEXTION_estado_color(&huart1, "engineTemp", 36609);
       } else if (random_value > 50 && random_value <= 80) {
           NEXTION_Alert(&huart1,0);
          NEXTION_estado_color(&huart1, "engineTemp", 64520);
       } else if (random_value > 91) {
           NEXTION_Alert(&huart1,63488); // red
           NEXTION_estado_color(&huart1, "engineTemp", 63488);
       }
       break;
```