Librería Ubit

Generado por Doxygen 1.9.1

1. Indice de archivos

1.1. Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

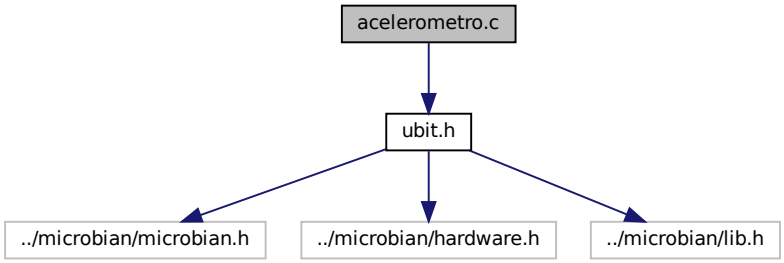
2. Documentación de archivos

2.1. Referencia del Archivo acelerometro.c

Conjunto de rutinas para el manejo del acelerómetro, integrado en el chip LSM303AGR.

```
#include "ubit.h"
```

Dependencia gráfica adjunta para acelerometro.c:



defines

- #define ACC 0x19

  La dirección base del acelerómetro en el bus I2C.
- #define ACC_CTRL_REG1 0x20

  La dirección del registro de control del acelerómetro.
- #define ACC_OUT_X 0x29

  La dirección del registro de datos donde el acelerómetro deposita las lecturas del valor de aceleración en el eje X.
- #define ACC_OUT_Y 0x2B

  La dirección del registro de datos donde el acelerómetro deposita las lecturas del valor de aceleración en el eje X.
- #define ACC_OUT_Z 0x2D

  La dirección del registro de datos donde el acelerómetro deposita las lecturas del valor de aceleración en el eje X.
- #define **PI** 3.14159265358979323846
- #define **RAD_A_GRAD** 180 / PI /∗ = 360 / (2 ∗ PI) ∗/
- #define **abs**(x) (x < 0) ? -x : x

Funciones

- void acelerometro_inicializa ()

  Configura el acelerómetro para que comience a realizar mediciones de la aceleración en los ejes X, Y y Z.
- int acelerometro_lectura_x ()

  Obtiene el valor de la aceleración en el eje X, codificado como un entero en el rango [-128, 128].
- int acelerometro_lectura_y ()

  Obtiene el valor de la aceleración en el eje Y, codificado como un entero en el rango [-128, 128].
- int acelerometro_lectura_z ()

  Obtiene el valor de la aceleración en el eje Z, codificado como un entero en el rango [-128, 128].
- float acelerometro_inclinacion_eje_x ()

  Proporciona el valor de la inclinación en el eje X, en el rango [-90, 90].
- float acelerometro_inclinacion_eje_y ()

  Proporciona el valor de la inclinación en el eje Y, en el rango [-90, 90].

2.1.1. Descripción detallada

Conjunto de rutinas para el manejo del acelerómetro, integrado en el chip LSM303AGR.

Autor

Noé Ruano Gutiérrez (nrg916@alumnos.unican.es)

Versión

1.0

Fecha

jul-2023

2.1.2. Documentación de las funciones

2.1.2.1. acelerometro_inclinacion_eje_x() `float acelerometro_inclinacion_eje_x ( )`

Proporciona el valor de la inclinación en el eje X, en el rango [-90, 90].

Devuelve

float La inclinación en el eje X

2.1.2.2. acelerometro_inclinacion_eje_y() `float acelerometro_inclinacion_eje_y ( )`

Proporciona el valor de la inclinación en el eje Y, en el rango [-90, 90].

Devuelve

float La inclinación en el eje Y

2.1.2.3. acelerometro_lectura_x() `int acelerometro_lectura_x ( )`

Obtiene el valor de la aceleración en el eje X, codificado como un entero en el rango [-128, 128].

Devuelve

int El valor de la aceleración en el eje X

2.1.2.4. acelerometro_lectura_y() `int acelerometro_lectura_y ( )`

Obtiene el valor de la aceleración en el eje Y, codificado como un entero en el rango [-128, 128].

Devuelve

int El valor de la aceleración en el eje Y

2.1.2.5. acelerometro_lectura_z() `int acelerometro_lectura_z ( )`

Obtiene el valor de la aceleración en el eje Z, codificado como un entero en el rango [-128, 128].

Devuelve

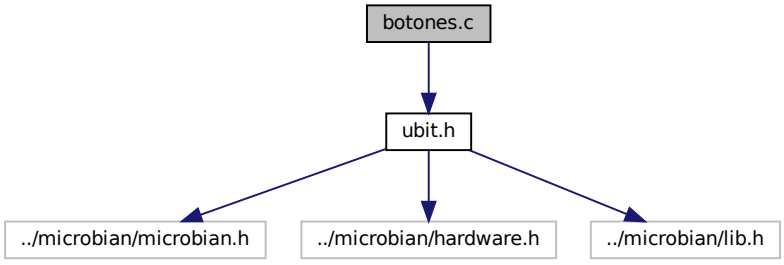
int El valor de la aceleración en el eje Z

2.2. Referencia del Archivo botones.c

Conjunto de rutinas para el manejo de los botones A y B, así como el botón táctil.

```
#include "ubit.h"
```

Dependencia gráfica adjunta para botones.c:



Funciones

- int boton_pulsado (boton_t b)

  Retorna el estado de un botón.
- int boton_espera_pulsacion (boton_t b)

  Realiza una espera activa a la espera de que se pulse un botón concreto.

2.2.1. Descripción detallada

Conjunto de rutinas para el manejo de los botones A y B, así como el botón táctil.

Autor

Noé Ruano Gutiérrez (nrg916@alumnos.unican.es)

Versión

1.0

Fecha

jul-2023

2.2.2. Documentación de las funciones

2.2.2.1. boton_espera_pulsacion() `int boton_espera_pulsacion ( boton_t b )`

Realiza una espera activa a la espera de que se pulse un botón concreto.

Parámetros

b	El botón cuya pulsación debe detectarse

Devuelve

int -1 si el botón indicado no se corresponde con ninguno de los contemplados, o 0 una vez el botón ha sido pulsado

2.2.2.2. boton_pulsado() `int boton_pulsado ( boton_t b )`

Retorna el estado de un botón.

Parámetros

b	El botón cuyo estado quiere conocerse

Devuelve

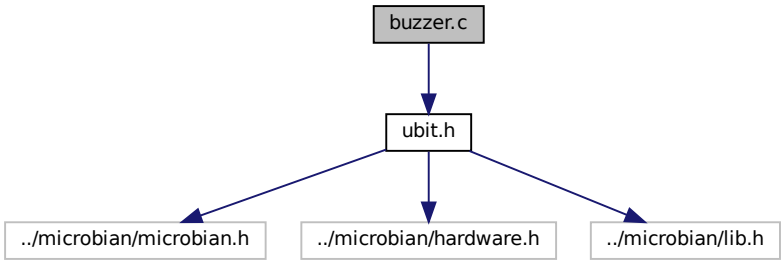
int 1 si el botón está pulsado o 0 si no lo está

2.3. Referencia del Archivo buzzer.c

Conjunto de rutinas para el control del zumbador piezoeléctrico.

```
#include "ubit.h"
```

Dependencia gráfica adjunta para buzzer.c:



Funciones

- void buzzer_reproduce_nota (nota_t n, int t_ms)

  Reproduce un tono de frecuencia igual al inverso del periodo correspondiente a la nota "n" indicada, durante t_ms milisegundos.

Variables

- unsigned int **periodo_us** [ ]

2.3.1. Descripción detallada

Conjunto de rutinas para el control del zumbador piezoeléctrico.

Autor

Noé Ruano (nrg916@alumnos.unican.es)

Versión

1.0

Fecha

jul-2023

2.3.2. Documentación de las funciones

2.3.2.1. buzzer_reproduce_nota() `void buzzer_reproduce_nota (`

```
nota_t n,
int t_ms )
```

Reproduce un tono de frecuencia igual al inverso del periodo correspondiente a la nota "n" indicada, durante t_ms milisegundos.

Parámetros

n	le nota a reproducir
t_ms	la duración de la nota

2.3.3. Documentación de las variables

2.3.3.1. periodo_us `unsigned int periodo_us[]`

Valor inicial:

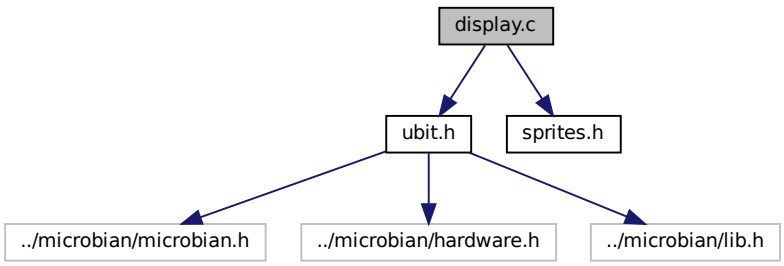
```
= {3822, 3405, 3033, 2863, 2551, 2272, 2024, 1911,
                            1702, 1516, 1431, 1275, 1136, 1012, 955}
```

2.4. Referencia del Archivo display.c

Conjunto de rutinas para el manejo del display de 5x5 LEDs.

```
#include "ubit.h"
#include "sprites.h"
```

Dependencia gráfica adjunta para display.c:



defines

- #define **abs**(x) (x < 0) ? -x : x

Funciones

- int display_cambia_intensidad (intensidad_t intensidad)

  Wrapper para la rutina de inicialización del proceso de refresco del display, proporcionada por la librería Microbian.
- int display_enciende_LED (int x, int y)

  Enciende un único LED en el display.
- int display_apaga_LED (int x, int y)

  Apaga un único LED en el display.
- void display_limpia ()

  Apaga todos los LEDs en el display.
- int display_muestra_imagen (imagen_t img)

  Muestra una imagen 2D utilizando los LEDs del display.
- int display_muestra_secuencia (imagen_t seq[ ], int n_elem_seq, int delay_ms)

  Muestra una secuencia de imágenes en el display.
- int display_muestra_sprite (char *sprite_bin)

  Muestra en el display el sprite correspondiente a la codificación indicada.
- void display_char2codi (char c, char **codi)

  Obtiene la codificación de un carácter. Si la codificación del carácter no se encuentra en la librería de sprites, esta se sustituye por la codificación del signo de cierre de interrogación.
- void display_muestra_texto (char *str, velocidad_texto_t vel)

  Muestra un texto en el display LED de la placa con una animación de deslizamiento hacia la izquierda.

2.4.1. Descripción detallada

Conjunto de rutinas para el manejo del display de 5x5 LEDs.

Autor

Noe Ruano Gutierrez (nrg916@alumnos.unican.es)

Versión

1.0

Fecha

jul-2023

2.4.2. Documentación de las funciones

2.4.2.1. display_apaga_LED() `int display_apaga_LED (`

```
int x,
int y )
```

Apaga un único LED en el display.

Parámetros

x	La coordenada en el eje de abscisas dentro del display
y	La coordenada en el eje de ordenadas dentro del display

Devuelve

int 0 en caso de una terminación correcta de la función, -1 si el valor de alguna coordenada está fuera del rango [0, 4]

2.4.2.2. display_cambia_intensidad()

```
int display_cambia_intensidad (
    intensidad_t intensidad )
```

Wrapper para la rutina de inicialización del proceso de refresco del display, proporcionada por la librería Microbian.

Devuelve

int -1 si el valor de intensidad no es apropiado, 0 en caso de una terminación correcta

2.4.2.3. display_char2codi()

```
void display_char2codi (
    char c,
    char ** codi )
```

Obtiene la codificación de un carácter. Si la codificación del carácter no se encuentra en la librería de sprites, esta se sustituye por la codificación del signo de cierre de interrogación.

Parámetros

c	El carácter cuya codificación quiere obtenerse
codi	El string donde depositar la codificación del carácter

2.4.2.4. display_enciende_LED()

```
int display_enciende_LED (
    int x,
    int y )
```

Enciende un único LED en el display.

Parámetros

x	La coordenada en el eje de abscisas dentro del display
y	La coordenada en el eje de ordenadas dentro del display

Devuelve

int 0 en caso de una terminación correcta de la función, -1 si el valor de alguna coordenada está fuera del rango [0, 4]

2.4.2.5. display_muestra_imagen() `int display_muestra_imagen ( imagen_t img )`

Muestra una imagen 2D utilizando los LEDs del display.

Parámetros

img	El array que contiene los valores que determinarán el estado de cada LED

Devuelve

int 0 si la imagen pudo mostrarse correctamente, o 1 si alguno de los valores de la imagen no es válido

2.4.2.6. display_muestra_secuencia() `int display_muestra_secuencia ( imagen_t seq[], int n_elem_seq, int delay_ms )`

Muestra una secuencia de imágenes en el display.

Parámetros

seq	El array que contiene las imágenes
n_elem_seq	El número de imágenes
delay_ms	El delay entre cada imagen

Devuelve

int -1 si el delay indicado no es válido, o 0 en caso de una terminación correcta

2.4.2.7. display_muestra_sprite() `int display_muestra_sprite ( char * sprite_bin )`

Muestra en el display el sprite correspondiente a la codificación indicada.

Parámetros

sprite_bin	La codificación del sprite

Devuelve

int -1 si la codificación del string no tiene una longitud adecuada (1 carácter por píxel = 25 píxeles) o si alguno de los carácteres de la cadena es distinto de '0' o '1'. Retorna 0 en caso de una terminación correcta

2.4.2.8. display_muestra_texto()

```
void display_muestra_texto (
    char * str,
    velocidad_texto_t vel )
```

Muestra un texto en el display LED de la placa con una animación de deslizamiento hacia la izquierda.

Parámetros

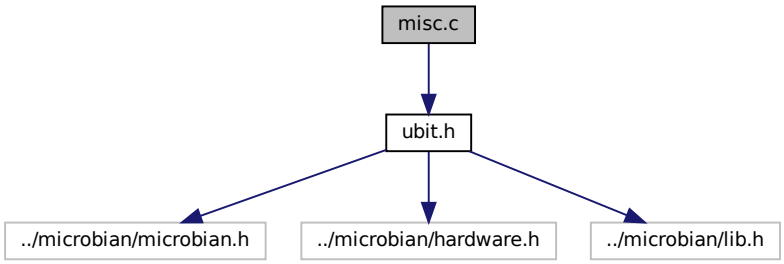
str	El texto a mostrar
v	La velocidad con la que deberá deslizarse el texto

2.5. Referencia del Archivo misc.c

Alberga las funciones init, microbit_inicializa_hardware y termometro_lectura.

```
#include "ubit.h"
```

Dependencia gráfica adjunta para misc.c:



defines

- #define TMP_STEP 0.25

  El paso empleado en la obtención de las medidas del sensor de temperatura.
- #define TEMP_BASE 0x4000C000

  La dirección base donde se encuentra mapeado el sensor de temperatura.
- #define TEMP_TASK_START 0x0

  El offset del registro TASK_START en el sensor de temperatura. Este es el registro empleado para iniciar una lectura.
- #define TEMP_DATARDY 0x100

  El offset del registro DATARDY en el sensor de temperatura. Indica con un valor 1 cuándo se encuentra disponible una lectura del sensor.
- #define TEMP_DATA 0x508

  El offset del registro DATA en el sensor de temperatura. En él deposita el sensor las lecturas obtenidas.

Funciones

- void init ()

  Esta función es el punto de entrada al código del usuario. Dicho código no podrá ejecutar correctamente si no es como parte de un proceso de Microbian. Se incluye como parte del código de la librería para que los alumnos no tengan que declararla en sus códigos, y puedan programar de la forma más parecida posible a como lo vienen haciendo hasta ahora en la asignatura de Introducción al Software, con una función principal main() más las funciones que ellos declaren y definan.
- void microbit_inicializa_hardware ()

  Wrapper para todas las rutinas de inicialización de los dispositivos hardware manejados por la librería Microbian que no pueden ser inicializados desde fuera de un proceso de Microbian.
- float termometro_lectura ()

  Obtiene el valor de temperatura leído por el probe del microcontrolador.

Variables

- volatile int * **temp_base** = (volatile int *)TEMP_BASE
- volatile int * **temp_task_start** = (volatile int *)(TEMP_BASE + TEMP_TASK_START)
- volatile int * **temp_datardy** = (volatile int *)(TEMP_BASE + TEMP_DATARDY)
- volatile int * **temp_data** = (volatile int *)(TEMP_BASE + TEMP_DATA)

2.5.1. Descripción detallada

Alberga las funciones init, microbit_inicializa_hardware y termometro_lectura.

Autor

Noé Ruano Gutiérrez (nrg916@alumnos.unican.es)

Versión

1.0

Fecha

jul-2023

2.5.2. Documentación de las funciones

2.5.2.1. termometro_lectura() `float termometro_lectura ( )`

Obtiene el valor de temperatura leído por el probe del microcontrolador.

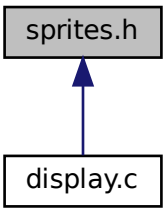
Devuelve

int Una aproximación de la temperatura ambiental

2.6. Referencia del Archivo sprites.h

Una pequeña librería de sprites que pueden ser mostrados en el display.

Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Variables

- char * **sprites** [ ]

2.6.1. Descripción detallada

Una pequeña librería de sprites que pueden ser mostrados en el display.

Autor

Noé Ruano Gutiérrez (nrg916@alumnos.unican.es)

Versión

1.0

Fecha

jul-2023

2.7. Referencia del Archivo ubit.h

Fichero de cabecera de la librería Ubit.

```
#include "../microbian/microbian.h"
#include "../microbian/hardware.h"
#include "../microbian/lib.h"
```

Dependencia gráfica adjunta para ubit.h:

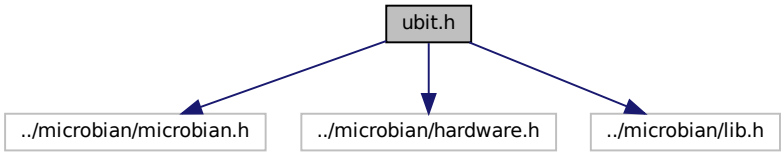
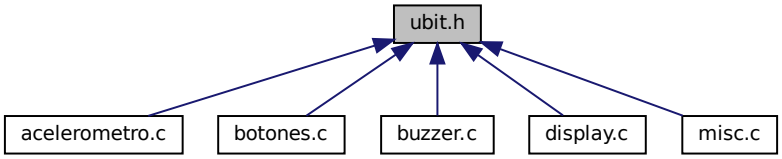


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



defines

- #define **DISPLAY_DIM** 5
- #define **NEGRA** 1000
- #define **CORCHEA** 500
- #define **SEMICORCHEA** 250

typedefs

- typedef int imagen_t[DISPLAY_DIM][DISPLAY_DIM]

  El tipo de dato con que construir una imagen de la librería Ubit que pueda ser mostrada/procesada por las funciones del driver de alto nivel del display.

Enumeraciones

- enum **intensidad_t** { **INT_BAJA** , **INT_MEDIA** , **INT_ALTA** }
- enum **velocidad_texto_t** { **LENTO** , **MEDIO** , **RAPIDO** }
- enum **boton_t** {
  BOTON_A , **BOTON_B** , **BOTON_LOGO** , **BOTON_0** ,
  BOTON_1 , **BOTON_2** }
- enum **nota_t** {
  DO_4 , **RE_4** , **MI_4** , **FA_4** ,
  SOL_4 , **LA_4** , **SI_4** , **DO_5** ,
  RE_5 , **MI_5** , **FA_5** , **SOL_5** ,
  LA_5 , **SI_5** , **DO_6** }

Funciones

- void **main** (int n)
- void microbit_inicializa_hardware ()

  Wrapper para todas las rutinas de inicialización de los dispositivos hardware manejados por la librería Microbian que no pueden ser inicializados desde fuera de un proceso de Microbian.
- float termometro_lectura ()

  Obtiene el valor de temperatura leído por el probe del microcontrolador.
- int display_cambia_intensidad (intensidad_t i)

  Wrapper para la rutina de inicialización del proceso de refresco del display, proporcionada por la librería Microbian.
- int display_enciende_LED (int x, int y)

  Enciende un único LED en el display.
- int display_apaga_LED (int x, int y)

  Apaga un único LED en el display.
- int display_muestra_imagen (imagen_t img)

  Muestra una imagen 2D utilizando los LEDs del display.
- int display_muestra_secuencia (imagen_t seq[ ], int n_elem_seq, int delay_ms)

  Muestra una secuencia de imágenes en el display.
- void display_limpia ()

  Apaga todos los LEDs en el display.
- int display_muestra_sprite (char ∗sprite_bin)

  Muestra en el display el sprite correspondiente a la codificación indicada.
- void display_char2codi (char c, char ∗∗codi)

  Obtiene la codificación de un carácter. Si la codificación del carácter no se encuentra en la librería de sprites, esta se sustituye por la codificación del signo de cierre de interrogación.
- void display_muestra_texto (char ∗str, velocidad_texto_t v)

  Muestra un texto en el display LED de la placa con una animación de deslizamiento hacia la izquierda.
- int boton_pulsado (boton_t b)

  Retorna el estado de un botón.
- int boton_espera_pulsacion (boton_t b)

  Realiza una espera activa a la espera de que se pulse un botón concreto.
- void acelerometro_inicializa ()

  Configura el acelerómetro para que comience a realizar mediciones de la aceleración en los ejes X, Y y Z.
- int acelerometro_lectura_x ()

  Obtiene el valor de la aceleración en el eje X, codificado como un entero en el rango [-128, 128].
- int acelerometro_lectura_y ()

  Obtiene el valor de la aceleración en el eje Y, codificado como un entero en el rango [-128, 128].
- int acelerometro_lectura_z ()

  Obtiene el valor de la aceleración en el eje Z, codificado como un entero en el rango [-128, 128].
- float acelerometro_inclinacion_eje_x ()

  Proporciona el valor de la inclinación en el eje X, en el rango [-90, 90].
- float acelerometro_inclinacion_eje_y ()

  Proporciona el valor de la inclinación en el eje Y, en el rango [-90, 90].
- void **brujula_inicializa** ()
- int **brujula_lectura_x** ()
- int **brujula_lectura_x2** ()
- int **brujula_lectura_y** ()
- int **brujula_lectura_z** ()
- void buzzer_reproduce_nota (nota_t n, int t_ms)

  Reproduce un tono de frecuencia igual al inverso del periodo correspondiente a la nota "n" indicada, durante t_ms milisegundos.

Variables

- image **imagen_actual_microbian**

2.7.1. Descripción detallada

Fichero de cabecera de la librería Ubit.

Autor

Noe Ruano Gutierrez (nrg916@alumnos.unican.es)

Versión

1.0

Fecha

jul-2023

2.7.2. Documentación de las funciones

2.7.2.1. acelerometro_inclinacion_eje_x() `float acelerometro_inclinacion_eje_x ( )`

Proporciona el valor de la inclinación en el eje X, en el rango [-90, 90].

Devuelve

float La inclinación en el eje X

2.7.2.2. acelerometro_inclinacion_eje_y() `float acelerometro_inclinacion_eje_y ( )`

Proporciona el valor de la inclinación en el eje Y, en el rango [-90, 90].

Devuelve

float La inclinación en el eje Y

2.7.2.3. acelerometro_lectura_x() `int acelerometro_lectura_x ( )`

Obtiene el valor de la aceleración en el eje X, codificado como un entero en el rango [-128, 128].

Devuelve

int El valor de la aceleración en el eje X

2.7.2.4. acelerometro_lectura_y() `int acelerometro_lectura_y ( )`

Obtiene el valor de la aceleración en el eje Y, codificado como un entero en el rango [-128, 128].

Devuelve

int El valor de la aceleración en el eje Y

2.7.2.5. acelerometro_lectura_z() `int acelerometro_lectura_z ( )`

Obtiene el valor de la aceleración en el eje Z, codificado como un entero en el rango [-128, 128].

Devuelve

int El valor de la aceleración en el eje Z

2.7.2.6. boton_espera_pulsacion() `int boton_espera_pulsacion ( boton_t b )`

Realiza una espera activa a la espera de que se pulse un botón concreto.

Parámetros

b	El botón cuya pulsación debe detectarse

Devuelve

int -1 si el botón indicado no se corresponde con ninguno de los contemplados, o 0 una vez el botón ha sido pulsado

2.7.2.7. boton_pulsado() `int boton_pulsado ( boton_t b )`

Retorna el estado de un botón.

Parámetros

b	El botón cuyo estado quiere conocerse

Devuelve

int 1 si el botón está pulsado o 0 si no lo está

2.7.2.8. buzzer_reproduce_nota() `void buzzer_reproduce_nota (`

```
nota_t n,
int t_ms )
```

Reproduce un tono de frecuencia igual al inverso del periodo correspondiente a la nota "n" indicada, durante t_ms milisegundos.

Parámetros

n	le nota a reproducir
t_ms	la duración de la nota

2.7.2.9. display_apaga_LED() `int display_apaga_LED (`

```
int x,
int y )
```

Apaga un único LED en el display.

Parámetros

x	La coordenada en el eje de abscisas dentro del display
y	La coordenada en el eje de ordenadas dentro del display

Devuelve

int 0 en caso de una terminación correcta de la función, -1 si el valor de alguna coordenada está fuera del rango [0, 4]

2.7.2.10. display_cambia_intensidad() `int display_cambia_intensidad (`

```
intensidad_t intensidad )
```

Wrapper para la rutina de inicialización del proceso de refresco del display, proporcionada por la librería Microbian.

Devuelve

int -1 si el valor de intensidad no es apropiado, 0 en caso de una terminación correcta

2.7.2.11. display_char2codi() `void display_char2codi (`

```
char c,
char ** codi )
```

Obtiene la codificación de un carácter. Si la codificación del carácter no se encuentra en la librería de sprites, esta se sustituye por la codificación del signo de cierre de interrogación.

Parámetros

c	El carácter cuya codificación quiere obtenerse
codi	El string donde depositar la codificación del carácter

2.7.2.12. display_enciende_LED() `int display_enciende_LED (`

```
int x,
int y )
```

Enciende un único LED en el display.

Parámetros

x	La coordenada en el eje de abscisas dentro del display
y	La coordenada en el eje de ordenadas dentro del display

Devuelve

int 0 en caso de una terminación correcta de la función, -1 si el valor de alguna coordenada está fuera del rango [0, 4]

2.7.2.13. display_muestra_imagen() `int display_muestra_imagen (`

```
imagen_t img )
```

Muestra una imagen 2D utilizando los LEDs del display.

Parámetros

img	El array que contiene los valores que determinarán el estado de cada LED

Devuelve

int 0 si la imagen pudo mostrarse correctamente, o 1 si alguno de los valores de la imagen no es válido

2.7.2.14. display_muestra_secuencia() int display_muestra_secuencia (

```
imagen_t seq[],
int n_elem_seq,
int delay_ms )
```

Muestra una secuencia de imágenes en el display.

Parámetros

seq	El array que contiene las imágenes
n_elem_seq	El número de imágenes
delay_ms	El delay entre cada imagen

Devuelve

int -1 si el delay indicado no es válido, o 0 en caso de una terminación correcta

2.7.2.15. display_muestra_sprite() int display_muestra_sprite (

```
char * sprite_bin )
```

Muestra en el display el sprite correspondiente a la codificación indicada.

Parámetros

sprite_bin	La codificación del sprite

Devuelve

int -1 si la codificación del string no tiene una longitud adecuada (1 carácter por píxel = 25 píxeles) o si alguno de los carácteres de la cadena es distinto de '0' o '1'. Retorna 0 en caso de una terminación correcta

2.7.2.16. display_muestra_texto() void display_muestra_texto (

```
char * str,
velocidad_texto_t vel )
```

Muestra un texto en el display LED de la placa con una animación de deslizamiento hacia la izquierda.

Parámetros

str	El texto a mostrar
v	La velocidad con la que deberá deslizarse el texto

2.7.2.17. termometro_lectura() float termometro_lectura ( )

Obtiene el valor de temperatura leído por el probe del microcontrolador.

Devuelve

int Una aproximación de la temperatura ambiental

Índice alfabético