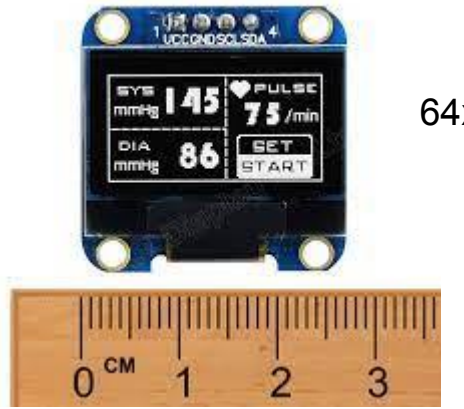
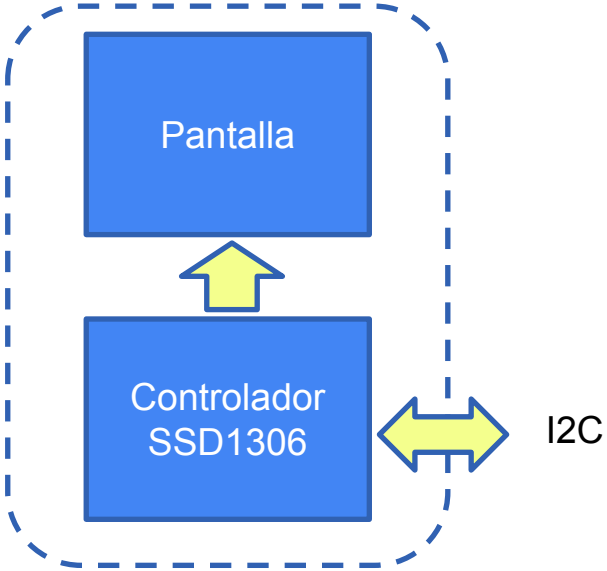
Modulo de pantalla OLED de 0.96 pulgadas monocromático

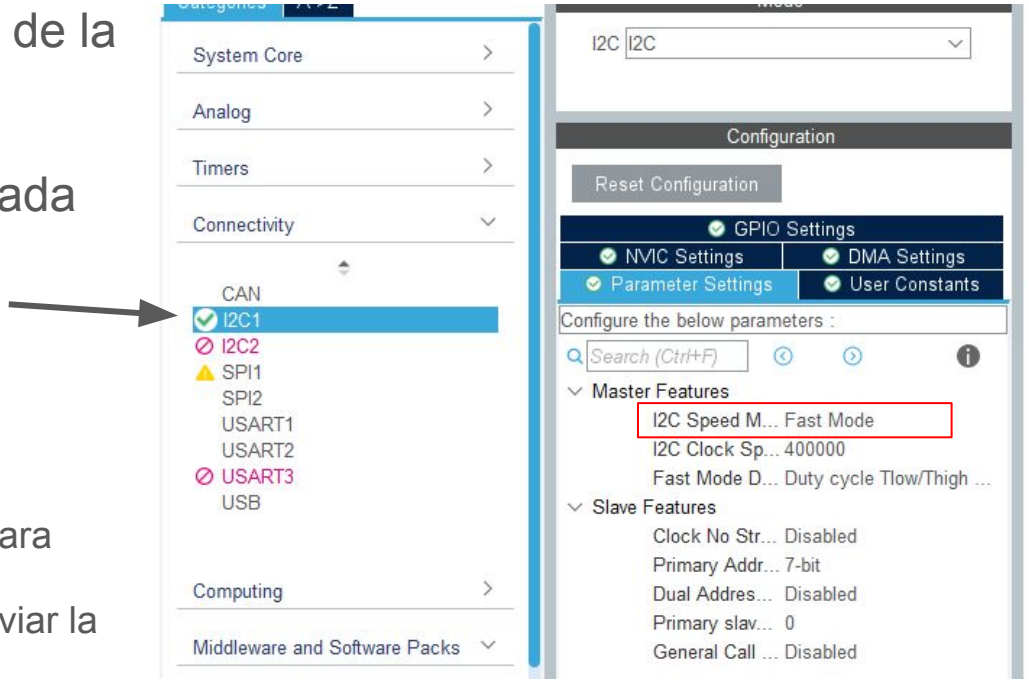


64x128

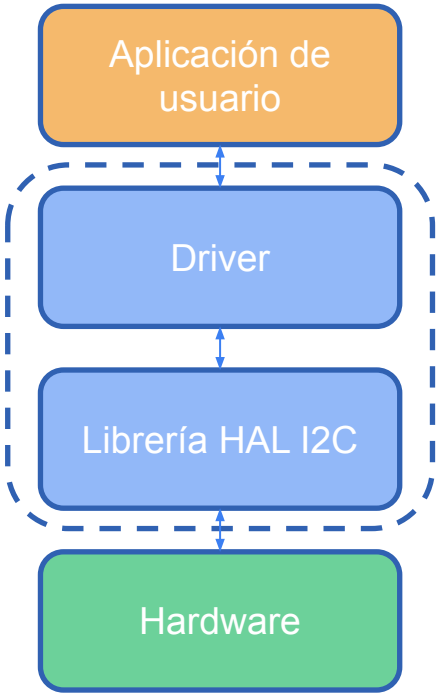


Uso del módulo

- Librería provista para descargar de la página de la cátedra: archivos ssd1306 y fonts .c y .h
- El archivo .c ya tiene implementada la comunicación I2C usando el periférico **I2C1** con la HAL
- Pasos/claves para su uso:
  a. Configurar periférico I2C
  b. Incluir encabezado de librería
  c. Llamar a la función de inicialización
  d. Utilizar las funciones de alto nivel para escribir texto o formas geométricas
  e. Llamar a la función Update para enviar la información a la pantalla



Organización del software



drawLine(int xo, int yo,int x1,int y1)

putpixel(int x, int y)

puts(char* msg, fuente,color)

gotoxy(int x, int y)

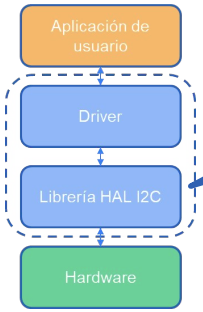
DrawBitmap(…..)

Funciones de la librería HAL I2C

HAL_I2C_Master_Transmit(…..)

Bajo Nivel:

Organizada en archivos apareados (.h y .c): ssd1306.c



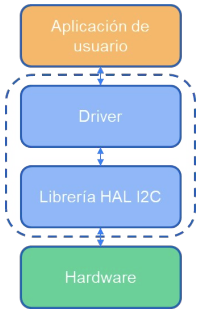

```c
void ssd1306_I2C_Init() {
uint32_t p = 250000;
while(p>0)
p--;
}
```

```c
void ssd1306_I2C_Write(uint8_t address, uint8_t reg, uint8_t data) {
uint8_t dt[2];
dt[0] = reg;
dt[1] = data;
HAL_I2C_Master_Transmit(&hi2c1, address, dt, 2, 10);
}
```

```c
void ssd1306_I2C_WriteMulti(uint8_t address, uint8_t reg, uint8_t* data, uint16_t count) {
uint8_t dt[256];
dt[0] = reg;
uint8_t i;
for(i = 0; i < count; i++)
dt[i+1] = data[i];
HAL_I2C_Master_Transmit(&hi2c1, address, dt, count+1, 10);
}
```

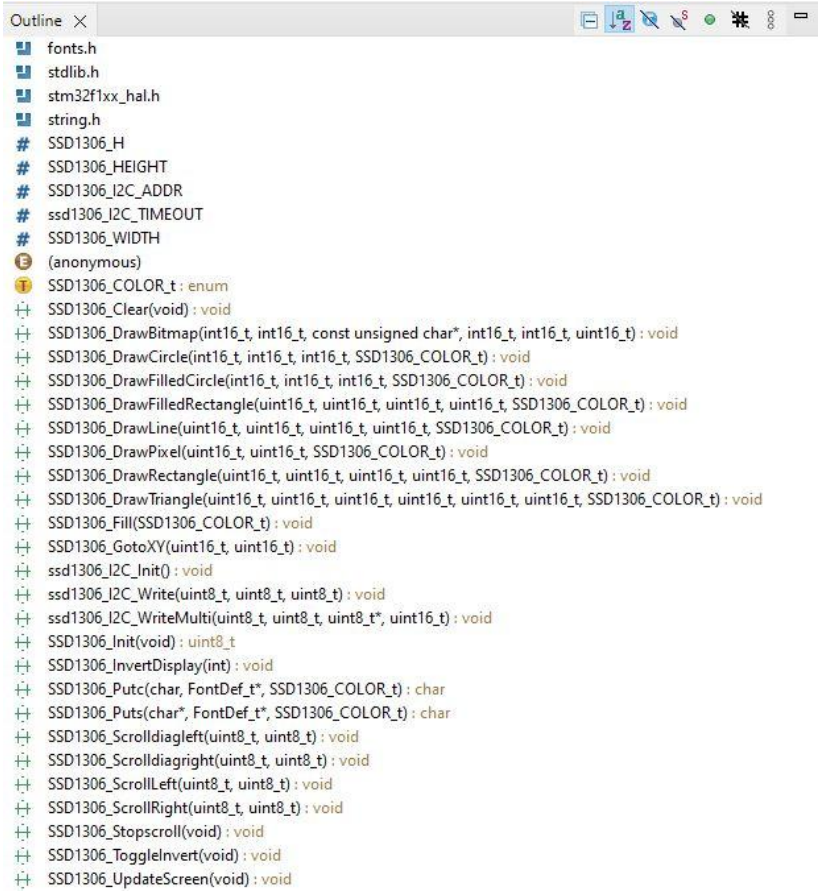
```c
/* Write command */
#define SSD1306_WRITECOMMAND(command)      ssd1306_I2C_Write(SSD1306_I2C_ADDR, 0x00, (command))
/* Write data */
#define SSD1306_WRITEDATA(data)            ssd1306_I2C_Write(SSD1306_I2C_ADDR, 0x40, (data))
```

Driver:

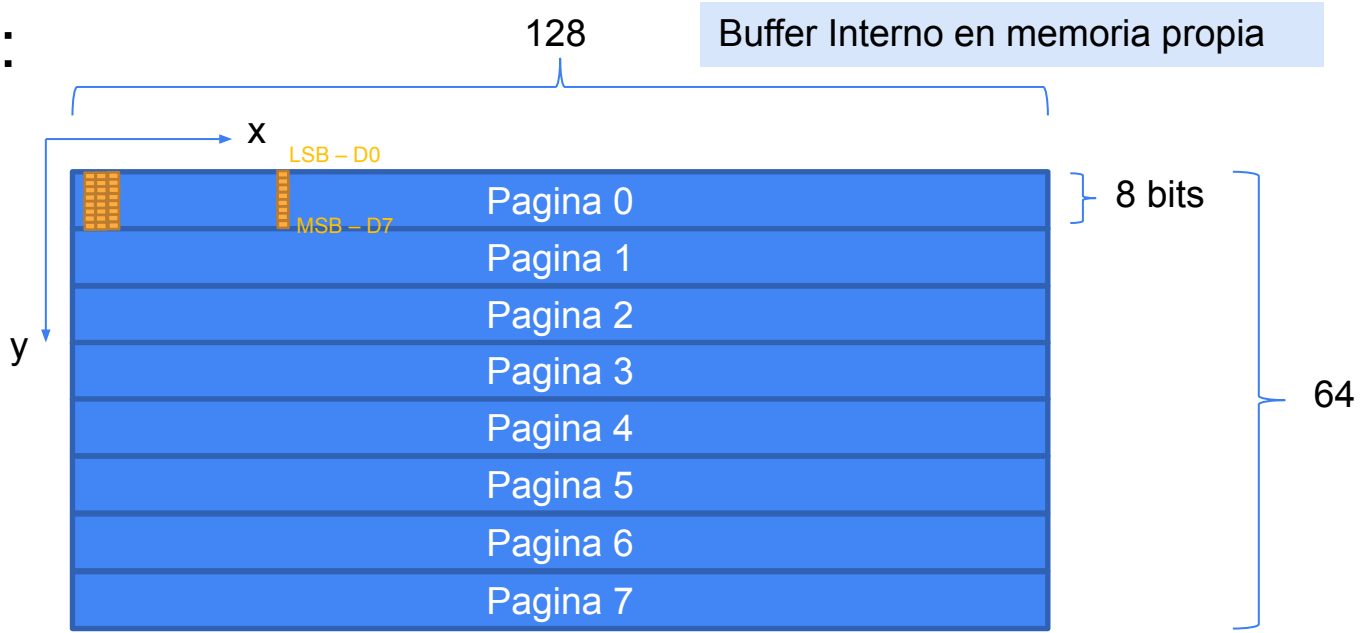


Organizado en 3 archivos:

1) ssd1306
2) fonts
3) header con mapas de bits



Hardware:



En código se modela mediante:

```
/* SSD1306 data buffer */
uint8_t SSD1306_Buffer[SSD1306_WIDTH * SSD1306_HEIGHT / 8];
[128 * 64 / 8] □ uint8_t SSD1306Buffer[1024];
```

Funciones básicas SSD1306_DrawPixel

```c
typedef enum {
SSD1306_COLOR_BLACK = 0x00, /*!< Black color, no pixel */
SSD1306_COLOR_WHITE = 0x01  /*!< Pixel is set*/
} SSD1306_COLOR_t;
```

```c
typedef struct {
uint16_t CurrentX;
uint16_t CurrentY;
uint8_t Inverted;
uint8_t Initialized;
} SSD1306_t;
```

```c
void SSD1306_DrawPixel(uint16_t x, uint16_t y, SSD1306_COLOR_t color) {
if (x >= SSD1306_WIDTH ||y >= SSD1306_HEIGHT) {
      /* Error */
      return;
      }
/* Check if pixels are inverted */
if (SSD1306.Inverted) {
      color = (SSD1306_COLOR_t)!color;
      }
/* Set color */
if (color == SSD1306_COLOR_WHITE) {
      SSD1306_Buffer[x + (y / 8) * SSD1306_WIDTH] |= 1 << (y % 8);
      }
else {
      SSD1306_Buffer[x + (y / 8) * SSD1306_WIDTH] &= ~(1 << (y % 8));
      }
}
```

Coordenada x [0-127]

Página [0-7]

Offset [128]

Función:

SSD1306_DrawBitmap

```
void SSD1306_DrawBitmap(int16_t x, int16_t y, const unsigned char* bitmap, int16_t w, int16_t h, uint16_t color)
```

Posición (x,y)

Tamaño bitmap

En archivo propio “mapa_de_bits.h”

Ejemplo logo 64x64

64 filas

```
const unsigned char logo[]={
255 ,255 ,255 ,255 ,255 ,255 ,255 ,255,
255 ,255 ,255 ,128 ,0 ,255 ,255 ,255,
255 ,255 ,252 ,0 ,0 ,63 ,255 ,255,
255 ,255 ,224 ,0 ,0 ,7 ,255 ,255,
255 ,255 ,130 ,0 ,0 ,1 ,255 ,255,
255 ,255 ,14 ,0 ,0 ,0 ,127 ,255,
.....................................................
255 ,255 ,240 ,0 ,0 ,15 ,255 ,255,
255 ,255 ,252 ,0 ,0 ,63 ,255 ,255,
255 ,255 ,255 ,192 ,3 ,255 ,255 ,255,
255 ,255 ,255 ,255 ,255 ,255 ,255 ,255};
```

8 bytes (8*8=64)

Fuentes:

Implementado en archivos fonts.h y fonts.c

```
typedef struct {
uint8_t FontWidth;    /*!< Font width in pixels */
uint8_t FontHeight;   /*!< Font height in pixels */
const uint16_t *data; /*!< Pointer to data font array */
} FontDef_t;

/**
 * @brief  String length and height
 */
typedef struct {
uint16_t Length;      /*!< String length in units of pixels */
uint16_t Height;      /*!< String height in units of pixels */
} FONTS_SIZE_t;
```

```
FontDef_t Font_7x10 = {
7,
10,
Font7x10
};

FontDef_t Font_11x18 = {
11,
18,
Font11x18
};

FontDef_t Font_16x26 = {
16,
26,
Font16x26
};
```

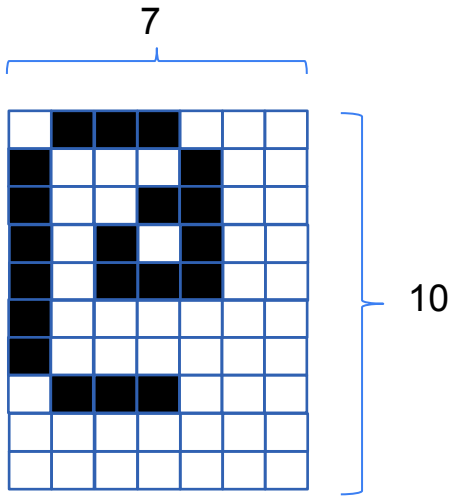
Tamaño de la fuente

Arreglo estático con la construcción de la fuente

Fuentes:

Arreglo estático con la construcción de la fuente

```
const uint16_t Font7x10 [] = {
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,  // sp
0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x0000, 0x1000, 0x0000, 0x0000,  // !
0x2800, 0x2800, 0x2800, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,  // "
0x2400, 0x2400, 0x7C00, 0x2400, 0x4800, 0x7C00, 0x4800, 0x4800, 0x0000, 0x0000,  // #
0x3800, 0x5400, 0x5000, 0x3800, 0x1400, 0x5400, 0x5400, 0x3800, 0x1000, 0x0000,  // $
0x2000, 0x5400, 0x5800, 0x3000, 0x2800, 0x5400, 0x1400, 0x0800, 0x0000, 0x0000,  // %
0x1000, 0x2800, 0x2800, 0x1000, 0x3400, 0x4800, 0x4800, 0x3400, 0x0000, 0x0000,  // &
.....
0x3800, 0x4400, 0x4400, 0x5400, 0x4400, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,  // 0
0x1000, 0x3000, 0x5000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x0000, 0x0000,  // 1
0x3800, 0x4400, 0x4400, 0x0400, 0x0800, 0x1000, 0x2000, 0x7C00, 0x0000, 0x0000,  // 2
0x3800, 0x4400, 0x0400, 0x1800, 0x0400, 0x0400, 0x4400, 0x3800, 0x0000, 0x0000,  // 3
0x0800, 0x1800, 0x2800, 0x2800, 0x4800, 0x7C00, 0x0800, 0x0800, 0x0000, 0x0000,  // 4
0x7C00, 0x4000, 0x4000, 0x7800, 0x0400, 0x0400, 0x4400, 0x3800, 0x0000, 0x0000,  // 5
0x3800, 0x4400, 0x4000, 0x7800, 0x4400, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,  // 6
0x7C00, 0x0400, 0x0800, 0x1000, 0x1000, 0x2000, 0x2000, 0x2000, 0x0000, 0x0000,  // 7
0x3800, 0x4400, 0x4400, 0x3800, 0x4400, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,  // 8
0x3800, 0x4400, 0x4400, 0x4400, 0x3C00, 0x0400, 0x4400, 0x3800, 0x0000, 0x0000,  // 9
0x0000, 0x0000, 0x1000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1000, 0x0000, 0x0000,  // :
0x0000, 0x0000, 0x0000, 0x1000, 0x0000, 0x0000, 0x0000, 0x1000, 0x1000, 0x1000,  // ;
0x0000, 0x0000, 0x0C00, 0x3000, 0x4000, 0x3000, 0x0C00, 0x0000, 0x0000, 0x0000,  // <
0x0000, 0x0000, 0x0000, 0x7C00, 0x0000, 0x7C00, 0x0000, 0x0000, 0x0000, 0x0000,  // =
0x0000, 0x0000, 0x6000, 0x1800, 0x0400, 0x1800, 0x6000, 0x0000, 0x0000, 0x0000,  // >
0x3800, 0x4400, 0x0400, 0x0800, 0x1000, 0x1000, 0x0000, 0x1000, 0x0000, 0x0000,  // ?
0x3800, 0x4400, 0x4C00, 0x5400, 0x5C00, 0x4000, 0x4000, 0x3800, 0x0000, 0x0000,  // @
0x1000, 0x2800, 0x2800, 0x2800, 0x2800, 0x7C00, 0x4400, 0x4400, 0x0000, 0x0000,  // A
0x7800, 0x4400, 0x4400, 0x7800, 0x4400, 0x4400, 0x4400, 0x7800, 0x0000, 0x0000,  // B
0x3800, 0x4400, 0x4000, 0x4000, 0x4000, 0x4000, 0x4400, 0x3800, 0x0000, 0x0000,  // C
0x7000, 0x4800, 0x4400, 0x4400, 0x4400, 0x4400, 0x4800, 0x7000, 0x0000, 0x0000,  // D
0x7C00, 0x4000, 0x4000, 0x7C00, 0x4000, 0x4000, 0x4000, 0x7C00, 0x0000, 0x0000,  // E
```



Funciones para la fuente:

SSD1306_Putc

```
char SSD1306_Putc(char ch, FontDef_t* Font, SSD1306_COLOR_t color) {
uint32_t i, b, j;

/* Check available space in LCD */
if (SSD1306_WIDTH <= (SSD1306.CurrentX + Font->FontWidth) ||
SSD1306_HEIGHT <= (SSD1306.CurrentY + Font->FontHeight))
        {/* Error */
         return 0;}
/* Go through font */
for (i = 0; i < Font->FontHeight; i++) {
    b = Font->data[(ch - 32) * Font->FontHeight + i];
    for (j = 0; j < Font->FontWidth; j++) {
       if ((b << j) & 0x8000) {
             SSD1306_DrawPixel(SSD1306.CurrentX + j, (SSD1306.CurrentY + i), (SSD1306_COLOR_t) color);}
       else {
             SSD1306_DrawPixel(SSD1306.CurrentX + j, (SSD1306.CurrentY + i), (SSD1306_COLOR_t)!color);}
        }
    }
/* Increase pointer */
SSD1306.CurrentX += Font->FontWidth;
/* Return character written */
return ch;
}
```

Combos útiles de funciones:

Modo texto

```
SSD1306_GotoXY (10,10); // goto 10, 10
SSD1306_Puts ("Hello World", &Font_11x18, 1);
SSD1306_GotoXY (10, 30);
SSD1306_Puts ("Hola Sistemas Embebidos", &Font_7x10, 1);
SSD1306_UpdateScreen(); // update screen
```

Mapa de bits con Scroll

```
SSD1306_Clear();
SSD1306_DrawBitmap(0,0,logo_gibic,64,64,1);
SSD1306_UpdateScreen();
SSD1306_ScrollRight(0,7);
HAL_Delay(15000);
SSD1306_Stopscroll();
```