**IDE, LCD ve HAL**

**(Integrated Development Environment)**

**İÇİNDEKİLER**

|  |  |
| --- | --- |
|  | **Sayfa No** |
| IDEler | **1** |
| LCD Uygulamaları | **2** |
| STM32 HAL Kütüphanesi | **16** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**IDEler**

CoIDE, SW4STM32, STM32CubeIDE, Eclipse ve Atollic TrueSTUDIO, STM32 işlemcilerinde yazılım geliştirmek için kullanılan popüler Integrated Development Environment (IDE) araçlarıdır. Aşağıda her bir IDE'nin özellikleri ve avantajları/dezavantajları hakkında kısa bilgiler verilmiştir:

CoIDE: CooCox tarafından geliştirilen ücretsiz bir IDE'dir. STM32 işlemcilerinde yazılım geliştirmek için kullanılabilir. Avantajları arasında ücretsiz olması, kolay kurulum ve kullanımı, kapsamlı dokümantasyonu ve açık kaynak kodlu olmasıdır. Dezavantajları arasında ise daha az popüler olması ve daha az desteklenen olmasıdır.

SW4STM32: Ac6 tarafından geliştirilen bir IDE'dir. STM32 işlemcilerinde yazılım geliştirmek için kullanılabilir. Avantajları arasında ücretsiz olması, Eclipse tabanlı olması ve kapsamlı dokümantasyonu vardır. Dezavantajları arasında ise daha az popüler olması ve daha az desteklenen olmasıdır.

STM32CubeIDE: ST Microelectronics tarafından geliştirilen bir IDE'dir. STM32 işlemcilerinde yazılım geliştirmek için kullanılabilir. Avantajları arasında ücretsiz olması, kapsamlı dokümantasyonu, STM32CubeMX tool'unun entegrasyonu ve STM32 işlemcisi için optimize edilmiş olmasıdır. Dezavantajları arasında ise daha az popüler olması ve daha az desteklenen olmasıdır.

Eclipse: Eclipse Foundation tarafından geliştirilen bir IDE'dir. STM32 işlemcilerinde yazılım geliştirmek için kullanılabilir. Avantajları arasında ücretsiz olması, kapsamlı dokümantasyonu, geniş bir eklenti ekosistemi ve çoklu platform desteği vardır. Dezavantajları arasında ise kurulum ve kullanımın zorluklarıdır.

Atollic TrueSTUDIO: Atollic tarafından geliştirilen bir IDE Atollic TrueSTUDIO, STM32 işlemcilerinde yazılım geliştirmek için kullanılan bir IDE'dir. Avantajları arasında ücretsiz ve ücretli sürümlerinin olması, kapsamlı dokümantasyonu, geniş bir eklenti ekosistemi, çoklu platform desteği ve optimize edilmiş performansıdır. Dezavantajları arasında ise ücretli sürümün maliyeti ve belirli özelliklerin sadece ücretli sürümde mevcut olmasıdır.

**LCD Uygulamaları**

STM32 işlemcilerinde LCD uygulamaları yapmak için birçok farklı yöntem ve kütüphane mevcuttur. Örneğin, HAL (Hardware Abstraction Layer) kütüphanesi ile STM32 işlemcisi arayüzünü kullanarak LCD ekranına veri yazabilirsiniz. Ayrıca, STM32 işlemcisi ile LCD ekranı arasındaki veri transferini gerçekleştirmek için birçok farklı yöntem mevcuttur. Örneğin, I2C, SPI veya parallel veri transferi gibi.

Örnek olarak, bir STM32F4 Discovery kartı ve bir 16x2 LCD ekran kullanarak basit bir saat uygulaması yapabilirsiniz. İlk olarak, STM32F4 Discovery kartının I2C arayüzünü kullanarak LCD ekranına veri yazabilirsiniz. Ardından, RTC (Real Time Clock) modülü kullanarak saat bilgisini okuyabilirsiniz ve LCD ekranına güncel saat bilgisini yazabilirsiniz. Bu uygulama için, HAL kütüphanesi ve I2C arayüzünü kullanmanız gerekecektir.

Bu sadece bir örnektir, STM32 işlemcilerinde LCD uygulamaları yaparken daha fazla özellik ve fonksiyonlar kullanabilirsiniz. Örneğin, ekranın arka ışığını kontrol etmek, ekranın parlaklığını ayarlamak, ekrandaki metinleri formatlamak, vb.

Ayrıca, STM32 işlemcisi ile LCD ekranı arasında veri transferini gerçekleştirmek için kullanabileceğiniz diğer yöntemler de mevcuttur. Örneğin, parallel veri transferi yöntemi ile LCD ekranına direk olarak veri gönderebilirsiniz. Bu yöntem, veri transfer hızı açısından daha yüksek olmasına rağmen, daha fazla pin gerektirir.

Ayrıca, mevcut olan ücretsiz açık kaynak kodlu kütüphaneler ve örnekleri kullanarak LCD ekranına veri yazmak için başka yollar da mevcut. Örneğin, Adafruit kütüphanesi kullanarak, birçok farklı tipte LCD ekranlarını STM32 işlemcisi ile kullanabilirsiniz.

Sonuç olarak, STM32 işlemcilerinde LCD uygulamaları yapmak için birçok farklı yöntem ve kütüphane mevcuttur. Kullanmak istediğiniz yönteme ve kullanmak istediğiniz LCD ekranın özelliklerine göre seçim yapabilirsiniz. Örnek uygulamalar ve kullanabileceğiniz ücretsiz açık kaynak kodlu kütüphaneler ile ilerlemek daha kolay hale gelecektir. Ayrıca, STM32 işlemcisi ile LCD ekranı arasındaki veri transferi için kullanabileceğiniz yöntemleri de dikkate almalısınız. Örneğin, veri transfer hızını ve gerekli olan pin sayısını dikkate almalısınız.

Aşağıdaki örnek uygulama STM32 tabanında Touchscreen li bir LCD için örnek koddur.

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* Enable I-Cache-------------------------------------------------------------\*/

SCB\_EnableICache();

/\* Enable D-Cache-------------------------------------------------------------\*/

SCB\_EnableDCache();

/\* MCU Configuration----------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* Configure the system clock \*/

SystemClock\_Config();

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_USART3\_UART\_Init();

MX\_SPI5\_Init();

MX\_RNG\_Init();

MX\_TIM1\_Init();

/\* USER CODE BEGIN 2 \*/

ILI9341\_Init();//initial driver setup to drive ili9341

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

//----------------------------------------------------------PERFORMANCE TEST

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("FPS TEST, 40 loop 2 screens", 10, 10, BLACK, 1, WHITE);

HAL\_Delay(2000);

ILI9341\_Fill\_Screen(WHITE);

uint32\_t Timer\_Counter = 0;

for(uint32\_t j = 0; j < 2; j++)

{

HAL\_TIM\_Base\_Start(&htim1);

for(uint16\_t i = 0; i < 10; i++)

{

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Fill\_Screen(BLACK);

}

//20.000 per second!

HAL\_TIM\_Base\_Stop(&htim1);

Timer\_Counter += \_\_HAL\_TIM\_GET\_COUNTER(&htim1);

\_\_HAL\_TIM\_SET\_COUNTER(&htim1, 0);

}

Timer\_Counter /= 2;

char counter\_buff[30];

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

sprintf(counter\_buff, "Timer counter value: %d", Timer\_Counter\*2);

ILI9341\_Draw\_Text(counter\_buff, 10, 10, BLACK, 1, WHITE);

double seconds\_passed = 2\*((float)Timer\_Counter / 20000);

sprintf(counter\_buff, "Time: %.3f Sec", seconds\_passed);

ILI9341\_Draw\_Text(counter\_buff, 10, 30, BLACK, 2, WHITE);

double timer\_float = 20/(((float)Timer\_Counter)/20000); //Frames per sec

sprintf(counter\_buff, "FPS: %.2f", timer\_float);

ILI9341\_Draw\_Text(counter\_buff, 10, 50, BLACK, 2, WHITE);

double MB\_PS = timer\_float\*240\*320\*2/1000000;

sprintf(counter\_buff, "MB/S: %.2f", MB\_PS);

ILI9341\_Draw\_Text(counter\_buff, 10, 70, BLACK, 2, WHITE);

double SPI\_utilized\_percentage = (MB\_PS/(6.25 ))\*100; //50mbits / 8 bits

sprintf(counter\_buff, "SPI Utilized: %.2f", SPI\_utilized\_percentage);

ILI9341\_Draw\_Text(counter\_buff, 10, 90, BLACK, 2, WHITE);

HAL\_Delay(10000);

static uint16\_t x = 0;

static uint16\_t y = 0;

char Temp\_Buffer\_text[40];

//----------------------------------------------------------COUNTING MULTIPLE SEGMENTS

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("Counting multiple segments at once", 10, 10, BLACK, 1, WHITE);

HAL\_Delay(2000);

ILI9341\_Fill\_Screen(WHITE);

for(uint16\_t i = 0; i <= 10; i++)

{

sprintf(Temp\_Buffer\_text, "Counting: %d", i);

ILI9341\_Draw\_Text(Temp\_Buffer\_text, 10, 10, BLACK, 2, WHITE);

ILI9341\_Draw\_Text(Temp\_Buffer\_text, 10, 30, BLUE, 2, WHITE);

ILI9341\_Draw\_Text(Temp\_Buffer\_text, 10, 50, RED, 2, WHITE);

ILI9341\_Draw\_Text(Temp\_Buffer\_text, 10, 70, GREEN, 2, WHITE);

ILI9341\_Draw\_Text(Temp\_Buffer\_text, 10, 90, BLACK, 2, WHITE);

ILI9341\_Draw\_Text(Temp\_Buffer\_text, 10, 110, BLUE, 2, WHITE);

ILI9341\_Draw\_Text(Temp\_Buffer\_text, 10, 130, RED, 2, WHITE);

ILI9341\_Draw\_Text(Temp\_Buffer\_text, 10, 150, GREEN, 2, WHITE);

ILI9341\_Draw\_Text(Temp\_Buffer\_text, 10, 170, WHITE, 2, BLACK);

ILI9341\_Draw\_Text(Temp\_Buffer\_text, 10, 190, BLUE, 2, BLACK);

ILI9341\_Draw\_Text(Temp\_Buffer\_text, 10, 210, RED, 2, BLACK);

}

HAL\_Delay(1000);

//----------------------------------------------------------COUNTING SINGLE SEGMENT

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("Counting single segment", 10, 10, BLACK, 1, WHITE);

HAL\_Delay(2000);

ILI9341\_Fill\_Screen(WHITE);

for(uint16\_t i = 0; i <= 100; i++)

{

sprintf(Temp\_Buffer\_text, "Counting: %d", i);

ILI9341\_Draw\_Text(Temp\_Buffer\_text, 10, 10, BLACK, 3, WHITE);

}

HAL\_Delay(1000);

//----------------------------------------------------------ALIGNMENT TEST

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("Rectangle alignment check", 10, 10, BLACK, 1, WHITE);

HAL\_Delay(2000);

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Draw\_Hollow\_Rectangle\_Coord(50, 50, 100, 100, BLACK);

ILI9341\_Draw\_Filled\_Rectangle\_Coord(20, 20, 50, 50, BLACK);

ILI9341\_Draw\_Hollow\_Rectangle\_Coord(10, 10, 19, 19, BLACK);

HAL\_Delay(1000);

//----------------------------------------------------------LINES EXAMPLE

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("Randomly placed and sized", 10, 10, BLACK, 1, WHITE);

ILI9341\_Draw\_Text("Horizontal and Vertical lines", 10, 20, BLACK, 1, WHITE);

HAL\_Delay(2000);

ILI9341\_Fill\_Screen(WHITE);

for(uint32\_t i = 0; i < 30000; i++)

{

uint32\_t random\_num = 0;

uint16\_t xr = 0;

uint16\_t yr = 0;

uint16\_t radiusr = 0;

uint16\_t colourr = 0;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

xr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

yr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

radiusr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

colourr = random\_num;

xr &= 0x01FF;

yr &= 0x01FF;

radiusr &= 0x001F;

//ili9341\_drawpixel(xr, yr, WHITE);

ILI9341\_Draw\_Horizontal\_Line(xr, yr, radiusr, colourr);

ILI9341\_Draw\_Vertical\_Line(xr, yr, radiusr, colourr);

}

HAL\_Delay(1000);

//----------------------------------------------------------HOLLOW CIRCLES EXAMPLE

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("Randomly placed and sized", 10, 10, BLACK, 1, WHITE);

ILI9341\_Draw\_Text("Circles", 10, 20, BLACK, 1, WHITE);

HAL\_Delay(2000);

ILI9341\_Fill\_Screen(WHITE);

for(uint32\_t i = 0; i < 3000; i++)

{

uint32\_t random\_num = 0;

uint16\_t xr = 0;

uint16\_t yr = 0;

uint16\_t radiusr = 0;

uint16\_t colourr = 0;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

xr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

yr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

radiusr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

colourr = random\_num;

xr &= 0x01FF;

yr &= 0x01FF;

radiusr &= 0x001F;

//ili9341\_drawpixel(xr, yr, WHITE);

ILI9341\_Draw\_Hollow\_Circle(xr, yr, radiusr\*2, colourr);

}

HAL\_Delay(1000);

//----------------------------------------------------------FILLED CIRCLES EXAMPLE

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("Randomly placed and sized", 10, 10, BLACK, 1, WHITE);

ILI9341\_Draw\_Text("Filled Circles", 10, 20, BLACK, 1, WHITE);

HAL\_Delay(2000);

ILI9341\_Fill\_Screen(WHITE);

for(uint32\_t i = 0; i < 1000; i++)

{

uint32\_t random\_num = 0;

uint16\_t xr = 0;

uint16\_t yr = 0;

uint16\_t radiusr = 0;

uint16\_t colourr = 0;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

xr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

yr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

radiusr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

colourr = random\_num;

xr &= 0x01FF;

yr &= 0x01FF;

radiusr &= 0x001F;

//ili9341\_drawpixel(xr, yr, WHITE);

ILI9341\_Draw\_Filled\_Circle(xr, yr, radiusr/2, colourr);

}

HAL\_Delay(1000);

//----------------------------------------------------------HOLLOW RECTANGLES EXAMPLE

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("Randomly placed and sized", 10, 10, BLACK, 1, WHITE);

ILI9341\_Draw\_Text("Rectangles", 10, 20, BLACK, 1, WHITE);

HAL\_Delay(2000);

ILI9341\_Fill\_Screen(WHITE);

for(uint32\_t i = 0; i < 20000; i++)

{

uint32\_t random\_num = 0;

uint16\_t xr = 0;

uint16\_t yr = 0;

uint16\_t radiusr = 0;

uint16\_t colourr = 0;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

xr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

yr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

radiusr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

colourr = random\_num;

xr &= 0x01FF;

yr &= 0x01FF;

radiusr &= 0x001F;

//ili9341\_drawpixel(xr, yr, WHITE);

ILI9341\_Draw\_Hollow\_Rectangle\_Coord(xr, yr, xr+radiusr, yr+radiusr, colourr);

}

HAL\_Delay(1000);

//----------------------------------------------------------FILLED RECTANGLES EXAMPLE

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("Randomly placed and sized", 10, 10, BLACK, 1, WHITE);

ILI9341\_Draw\_Text("Filled Rectangles", 10, 20, BLACK, 1, WHITE);

HAL\_Delay(2000);

ILI9341\_Fill\_Screen(WHITE);

for(uint32\_t i = 0; i < 20000; i++)

{

uint32\_t random\_num = 0;

uint16\_t xr = 0;

uint16\_t yr = 0;

uint16\_t radiusr = 0;

uint16\_t colourr = 0;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

xr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

yr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

radiusr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

colourr = random\_num;

xr &= 0x01FF;

yr &= 0x01FF;

radiusr &= 0x001F;

//ili9341\_drawpixel(xr, yr, WHITE);

ILI9341\_Draw\_Rectangle(xr, yr, radiusr, radiusr, colourr);

}

HAL\_Delay(1000);

//----------------------------------------------------------INDIVIDUAL PIXEL EXAMPLE

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("Slow draw by selecting", 10, 10, BLACK, 1, WHITE);

ILI9341\_Draw\_Text("and adressing pixels", 10, 20, BLACK, 1, WHITE);

HAL\_Delay(2000);

ILI9341\_Fill\_Screen(WHITE);

x = 0;

y = 0;

while (y < 240)

{

while ((x < 320) && (y < 240))

{

if(x % 2)

{

ILI9341\_Draw\_Pixel(x, y, BLACK);

}

x++;

}

y++;

x = 0;

}

x = 0;

y = 0;

while (y < 240)

{

while ((x < 320) && (y < 240))

{

if(y % 2)

{

ILI9341\_Draw\_Pixel(x, y, BLACK);

}

x++;

}

y++;

x = 0;

}

HAL\_Delay(2000);

//----------------------------------------------------------INDIVIDUAL PIXEL EXAMPLE

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("Random position and colour", 10, 10, BLACK, 1, WHITE);

ILI9341\_Draw\_Text("500000 pixels", 10, 20, BLACK, 1, WHITE);

HAL\_Delay(2000);

ILI9341\_Fill\_Screen(WHITE);

for(uint32\_t i = 0; i < 500000; i++)

{

uint32\_t random\_num = 0;

uint16\_t xr = 0;

uint16\_t yr = 0;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

xr = random\_num;

random\_num = HAL\_RNG\_GetRandomNumber(&hrng);

yr = random\_num;

uint16\_t color = HAL\_RNG\_GetRandomNumber(&hrng);

xr &= 0x01FF;

yr &= 0x01FF;

ILI9341\_Draw\_Pixel(xr, yr, color);

}

HAL\_Delay(2000);

//----------------------------------------------------------565 COLOUR EXAMPLE, Grayscale

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("Colour gradient", 10, 10, BLACK, 1, WHITE);

ILI9341\_Draw\_Text("Grayscale", 10, 20, BLACK, 1, WHITE);

HAL\_Delay(2000);

for(uint16\_t i = 0; i <= (320); i++)

{

uint16\_t Red = 0;

uint16\_t Green = 0;

uint16\_t Blue = 0;

Red = i/(10);

Red <<= 11;

Green = i/(5);

Green <<= 5;

Blue = i/(10);

uint16\_t RGB\_color = Red + Green + Blue;

ILI9341\_Draw\_Rectangle(i, x, 1, 240, RGB\_color);

}

HAL\_Delay(2000);

//----------------------------------------------------------IMAGE EXAMPLE, Snow Tiger

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("RGB Picture", 10, 10, BLACK, 1, WHITE);

ILI9341\_Draw\_Text("TIGER", 10, 20, BLACK, 1, WHITE);

HAL\_Delay(2000);

ILI9341\_Draw\_Image((const char\*)snow\_tiger, SCREEN\_VERTICAL\_2);

ILI9341\_Set\_Rotation(SCREEN\_VERTICAL\_1);

HAL\_Delay(10000);

//----------------------------------------------------------TOUCHSCREEN EXAMPLE

ILI9341\_Fill\_Screen(WHITE);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

ILI9341\_Draw\_Text("Touchscreen", 10, 10, BLACK, 2, WHITE);

ILI9341\_Draw\_Text("Touch to draw", 10, 30, BLACK, 2, WHITE);

ILI9341\_Set\_Rotation(SCREEN\_VERTICAL\_1);

while(1)

{

if(TP\_Touchpad\_Pressed())

{

uint16\_t x\_pos = 0;

uint16\_t y\_pos = 0;

HAL\_GPIO\_WritePin(GPIOB, LD3\_Pin|LD2\_Pin, GPIO\_PIN\_SET);

uint16\_t position\_array[2];

if(TP\_Read\_Coordinates(position\_array) == TOUCHPAD\_DATA\_OK)

{

x\_pos = position\_array[0];

y\_pos = position\_array[1];

ILI9341\_Draw\_Filled\_Circle(x\_pos, y\_pos, 2, BLACK);

ILI9341\_Set\_Rotation(SCREEN\_HORIZONTAL\_2);

char counter\_buff[30];

sprintf(counter\_buff, "POS X: %.3d", x\_pos);

ILI9341\_Draw\_Text(counter\_buff, 10, 80, BLACK, 2, WHITE);

sprintf(counter\_buff, "POS Y: %.3d", y\_pos);

ILI9341\_Draw\_Text(counter\_buff, 10, 120, BLACK, 2, WHITE);

ILI9341\_Set\_Rotation(SCREEN\_VERTICAL\_1);

}

//ILI9341\_Draw\_Pixel(x\_pos, y\_pos, BLACK);

}

else

{

HAL\_GPIO\_WritePin(GPIOB, LD3\_Pin|LD2\_Pin, GPIO\_PIN\_RESET);

}

}

}

/\* USER CODE END 3 \*/

}

**STM32 HAL Kütüphanesi**

STM32 HAL (Hardware Abstraction Layer) kütüphanesi, STMicroelectronics tarafından sağlanan bir kütüphanedir. Bu kütüphane, STM32 işlemcilerin çeşitli periferiklerini kontrol etmek için kullanılabilen fonksiyonları içermektedir. HAL kütüphanesi, STM32 işlemcilerin çeşitli periferiklerine erişmek için kullanabileceğiniz birçok fonksiyonu içerir. Bu fonksiyonlar arasında,

* GPIO (General Purpose Input/Output) pinlerini kontrol etmek,
* USART (Universal Synchronous/Asynchronous Receiver/Transmitter) modülünü kullanarak veri transferi yapmak,
* I2C (Inter-Integrated Circuit) ve SPI (Serial Peripheral Interface) arayüzlerini kullanarak veri transferi yapmak,
* ADC (Analog-to-Digital Converter) ve DAC (Digital-to-Analog Converter) modüllerini kullanarak analog-digital dönüşümleri yapmak,
* Timer ve PWM modüllerini kullanarak zamanlama ve pulse-width modulation işlemlerini gerçekleştirmek,
* RTC (Real Time Clock) modülünü kullanarak zaman bilgisini okumak ve ayarlamak,
* Flash bellek okuma ve yazma işlemlerini gerçekleştirmek gibi fonksiyonlar mevcut.

Bu fonksiyonlar arasında en kritik olanlarından bazıları şunlar olabilir:

* GPIO kontrol fonksiyonları: Bu fonksiyonlar sayesinde, STM32 işlemcisi üzerindeki GPIO pinlerini kontrol edebilirsiniz. Örneğin, pinleri input veya output olarak ayarlayabilir, pinlerin durumunu okuyabilirsiniz.
* USART fonksiyonları: Bu fonksiyonlar sayesinde, STM32 işlemcisi ile diğer cihazlar arasında seri veri transferi yapabilirsiniz. Örneğin, USART modülünü kullanarak veri gönderebilir veya alabilirsiniz.
* I2C ve SPI fonksiyonları: Bu fonksiyonlar sayesinde, STM32 işlemcisi ile diğer cihazlar arasında

I2C veya SPI arayüzleri ile veri transferi yapabilirsiniz. Örneğin, bir sensör veya ekran gibi periferik cihazların verilerini okuyabilir veya cihazlara komut gönderebilirsiniz.

* Timer ve PWM fonksiyonları: Bu fonksiyonlar sayesinde, STM32 işlemcisi üzerindeki timer ve PWM modüllerini kullanarak zamanlama ve pulse-width modulation işlemlerini gerçekleştirebilirsiniz. Örneğin, bir LED'i blink ettirmek için PWM modülünü kullanabilirsiniz.
* RTC fonksiyonları: Bu fonksiyonlar sayesinde, STM32 işlemcisi üzerindeki RTC modülünü kullanarak gerçek zaman bilgisini okuyabilir ve ayarlayabilirsiniz. Örneğin, bir alarm uygulaması yazarken RTC modülünü kullanabilirsiniz.

HAL kütüphanesinin alternatifleri arasında LL (Low Level) kütüphanesi, SPL (Standard Peripheral Library) ve CMSIS (Cortex Microcontroller Software Interface Standard) gibi kütüphaneler bulunmaktadır.

LL kütüphanesi, STM32 mikroişlemcilerin özelliklerini en temel seviyede kullanmayı sağlar. Bu kütüphane, HAL kütüphanesinden daha az katmanlıdır ve daha az bellek kullanır. Ancak, LL kütüphanesi kullanımı daha zor olabilir ve daha fazla kod yazma gerektirebilir.

SPL kütüphanesi ise, STM32 mikroişlemcilerin özelliklerini kullanmak için daha esnek bir seçenek sunar. SPL, HAL kütüphanesinden daha az katmanlıdır ve daha az bellek kullanır. Ancak, SPL kütüphanesi kullanımı daha zor olabilir ve daha fazla kod yazma gerektirebilir.

CMSIS, ARM tarafından geliştirilen bir kütüphanedir ve çeşitli ARM mikroişlemciler için kullanılabilir. CMSIS, STM32 mikroişlemciler için kullanılırken daha yüksek seviyede işlevsellik sunar. Ancak, CMSIS kullanımı daha zor olabilir ve daha fazla bellek kullanabilir.

Sonuç olarak, STM32 HAL kütüphanesi, STM32 mikroişlemciler için en kolay kullanımı sağlayan seçenektir. Ancak, alternatif kütüphaneler de kullanılabilir ve kullanıcının ihtiyaçlarına göre en uygun seçenek seçilebilir. Örneğin, bellek veya performans gibi konularda sınırlamalar olan projeler için LL veya SPL kullanılabilir. Ancak, daha yüksek seviyede işlevsellik ve daha fazla özellik isteyen projeler için CMSIS veya HAL kütüphaneleri daha uygun seçenekler olabilir. Ayrıca, kullanılan kütüphanenin kullanımının iyi öğrenilmesi ve anlaşılması, projelerin daha hızlı ve etkili bir şekilde gerçekleştirilmesini sağlayacaktır. Özellikle HAL kütüphanesi kullanımını iyi bilmeyen bir kullanıcı için, SPL veya LL kütüphanelerin kullanımı daha kolay olabilir. Ancak, kullanılan kütüphane her ne olursa olsun, her zaman tüm özelliklerin ve fonksiyonların iyi anlaşılması ve doğru kullanılması önemlidir.