# 使用 SquareLine Studio 設計 GUI 與執行

NuMicro® 32 位元系列微控制器範例程式碼介紹

## 文件資訊

| | |
|---|---|
| 應用簡述 | 使用 SquareLine Studio 工具設計 LVGL GUI，並將產出的 UI 檔案整合至 Nuvoton MCU 專案中編譯與執行。 |
| BSP 版本 | M55M1_Series_BSP_CMSIS_V3.01.005 |
| 硬體平台 | NuMaker-X-M55M1D |

*The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.*

*Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.*

*All data and specifications are subject to change without notice.*

*For additional information or questions, please contact: Nuvoton Technology Corporation.*

www.nuvoton.com

# 1. 概述

本範例程式與文件說明如何以 SquareLine Studio 設計 LVGL GUI，以及導覽基本的 SquareLine Studio 畫面設計功能，並將產出的 UI 檔案整合至 Nuvoton MCU 專案中執行。

本範例以 NuMicro M55M1 平台為例，亦可應用於表 1-1 所述的其他平台使用。

## 1.1 原理

### 1.1.1 SquareLine Studio 介紹

SquareLine Studio（以下簡稱 SLS）是一套專為 LVGL 打造的所見即所得（WYSIWYG）GUI 設計工具。設計者只要在畫布上以拖曳方式擺放按鈕、標籤、圖片、滑桿等元件，並於屬性面板調整版面、樣式與事件，SLS 便會自動產生對應的 LVGL C 原始碼，省去手寫介面程式的繁瑣工作。

SLS 其中的 LVGL Library 9.2 版內建部分 Nuvoton 開發板選項，如圖 1-1，可直接選擇對應平台建立專案；搭配 “lv_port_nuvoton” BSP，將匯出的 UI 檔案整合進專案即可在 Nuvoton 開發板上編譯與執行，讓畫面設計與韌體開發流程無縫接軌。

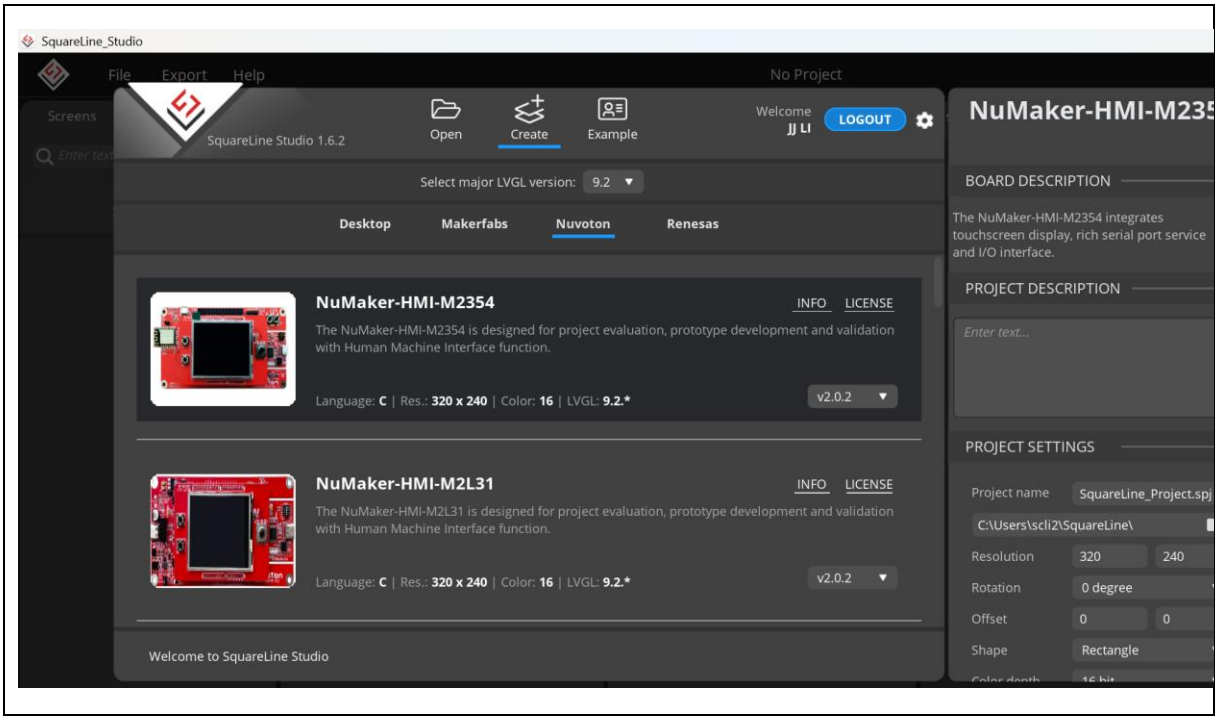



圖 1-1 SquareLine Studio LVGL 9.2 Version

而在“lv_port_nuvoton” BSP 中目前已移植並支援的平台如表 1-1 所示，並且針對對應的開發板移植 LVGL 範例程式，支援平台會根據 LVGL 版本更新而異動，請依 GitHub 官方說明 為參考。

| NuMicro Series | NuMaker Platforms |
|---|---|
| M2354 Series | NuMaker-HMI-M2354 |
| M2L31 Series | NuMaker-HMI-M2L31 |
| M3334 Series | NuMaker-HMI-M3334 |
| M3351 Series | NuMaker-HMI-M3351 |
| M487 Series | NuMaker-HMI-M487 |
| M467 Series | NuMaker-HMI-M467 |
| M55M1 Series | NuMaker-X-M55M1D |
| M5531 Series | NuMaker-HMI-M5531 |
| N9H Series | NuMaker-HMI-N9H20<br>NuMaker-HMI-N9H26<br>NuMaker-HMI-N9H30<br>NuMaker-HMI-N9H31 |
| NUC980 Series | NuMaker-IIoT-NUC980G1D<br>NuMaker-IIoT-NUC980G2D |
| MA35D1 Series | NuMaker-HMI-MA35D1 |
| MA35H0 Series | NuMaker-HMI-MA35H0 |

表 1-1 Supported Platform List

### 1.1.2 執行步驟

1. 下載與註冊 SquareLine Studio，請參考 SLS 官方下載頁面。

2. 建立 SLS 專案：

1) 在 SLS 開啟後，點選 Create，並在 Select major LVGL version 選擇 9.2，如圖 1-2。

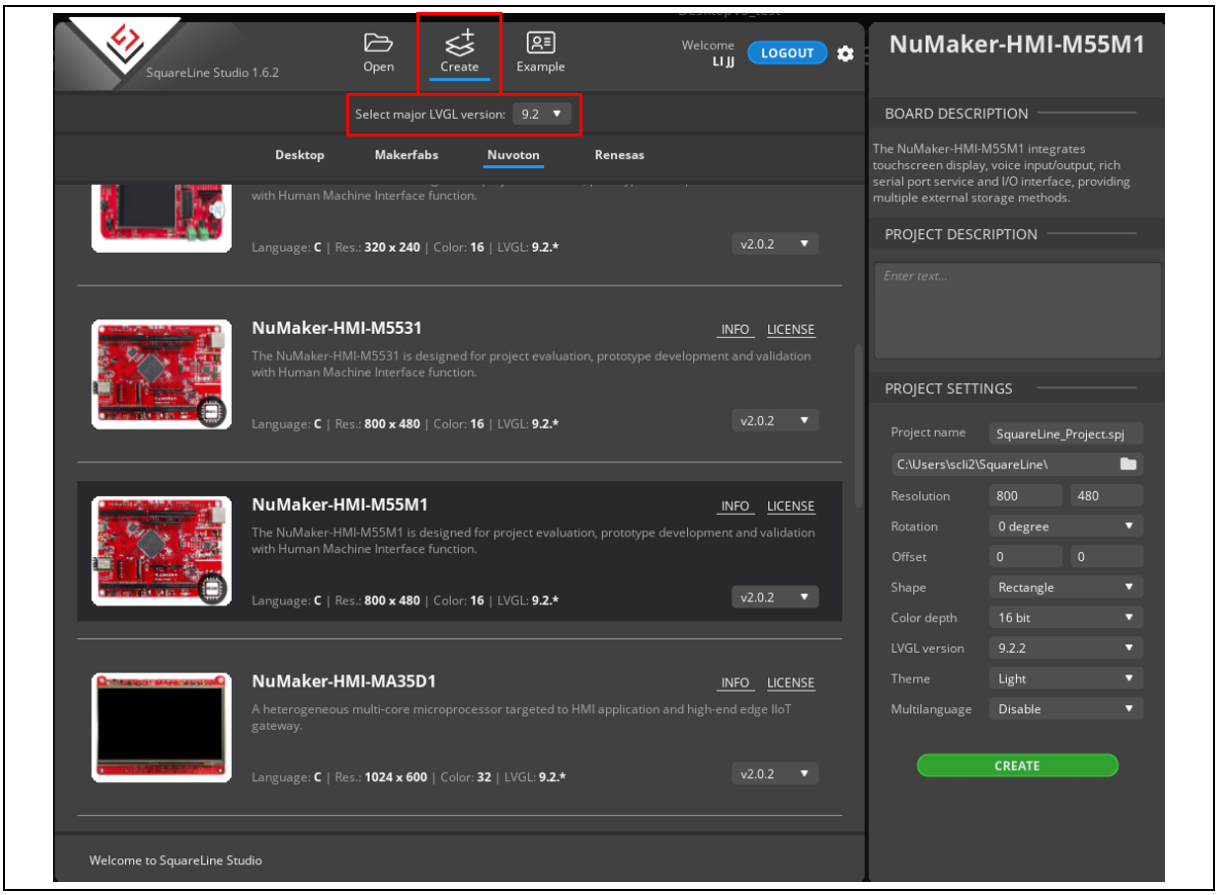



圖 1-2 建立 SLS 專案

2) 若能在選單中找到對應的平台與開發環境，可以直接挑選對應選項，參考圖 1-2，如 M55M1 的 NuMaker-HMI-M55M1，同時可以看到右邊的 Project Settings 區域已按照對應規格設定完成。

若無法在選單中找到對應的平台與開發環境，可以先任選其他開發板或 Desktop 版本，再於 Project Settings 區域設定對應的專案屬性即可。而專案屬性規格設定內容與注意事項可參考表 1-2 說明。

| Properties | Notes |
|---|---|
| Project Name/Path | 名稱與路徑須避免任何空白或非英語的文字 |
| Resolution | 解析度務必與實際 LCD 相同，否則畫面比例會錯誤 |

| Properties | Notes |
|---|---|
| Rotation | 旋轉角度一般皆為 0，若面板實體方向不同可再調整 |
| Offset | 偏移設定保持 0 |
| Shape | 面板形狀可以設定為矩形或圓形 |
| Color depth | 色彩深度依據需求可設定為 16/32 bit |
| LVGL version | 根據 Nuvoton GitHub 說明的同步版本，可向下相容 |
| Theme | 主題可選擇深/淺色，但僅影響外觀預設值 |

表 1-2 Project Settings

3. SLS GUI 介面導覽：

SLS 介面如圖 1-3 所示，各項功能說明可參考表 1-3，進階介面設定與功能可參考 SLS 官方文件。

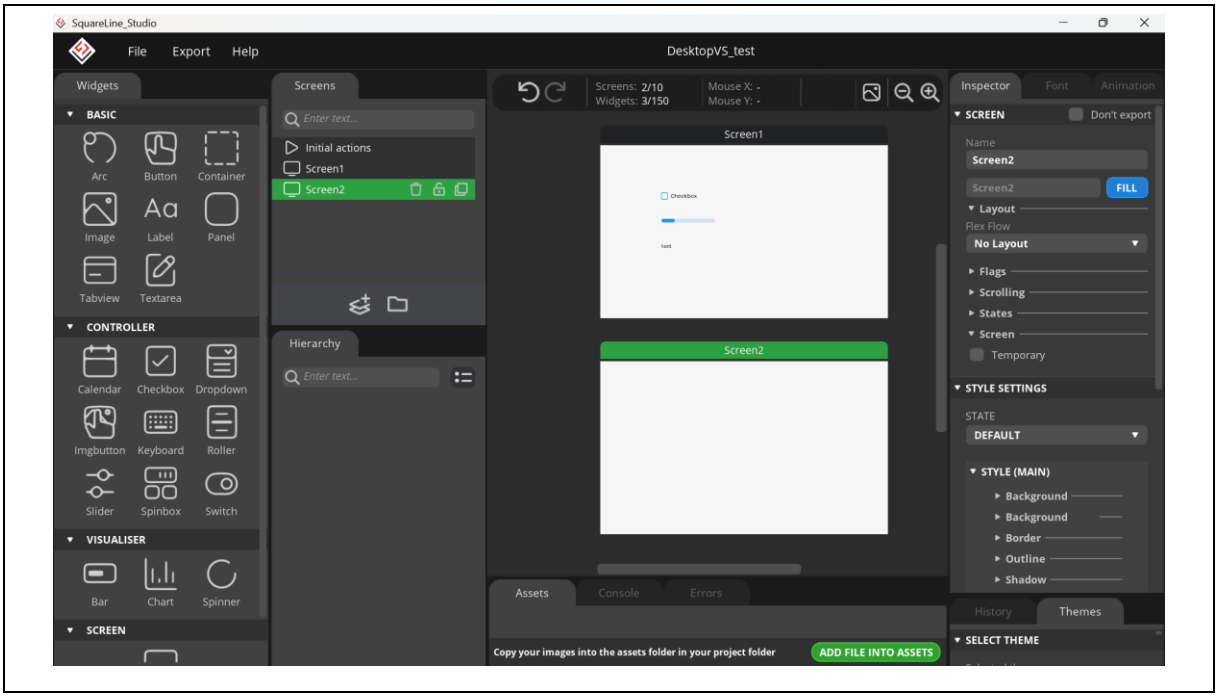



圖 1-3 SLS 介面

| 名稱 | 用途 |
|---|---|
| Widgets | 元件庫，可以直接點擊或拖曳至畫布上 |

| 名稱 | 用途 |
|---|---|
| Screens | 管理多個頁面(Screens)設計 |
| Canvas | 實際擺放位置與排版的所見即所得區域 |
| Hierarchy | 以樹狀結構檢視頁面中所有元件的階層關係 |
| Inspector | 設定選取元件的位置、大小、樣式、事件等屬性 |
| Animations | 為元件加入事件或動畫 |

表 1-3 SLS 介面功能說明

4. SLS UI 畫面設計：實作簡易的儀表板功能

依照以下步驟，可完成如圖 1-4 所示的簡易儀表板畫面設計。此範例包含兩個畫面：Screen1 用於顯示動態數值與 Arc 連動效果；Screen2 則透過 Slider 控制數值顯示。

**Screen1：動態數值與 Arc 顯示**

在 Screen1 中，數值會在 0 至 100 之間變化，並同步帶動 Arc 的顯示位置或進度變化。畫面上同時配置 Button，可切換至 Screen2。

**Screen2：Slider 數值控制**

在 Screen2 中，可拖曳 Slider 調整 0 至 1000 的數值範圍，畫面上的數字會依照 Slider 目前值即時更新。

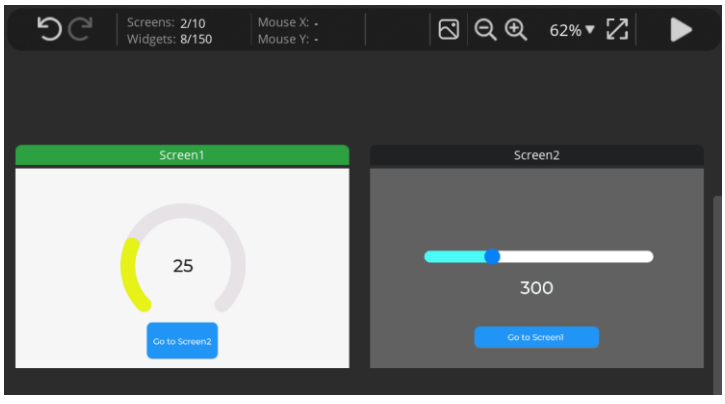



圖 1-4 儀表板實作

1) 新增 Screen2：如圖 1-5，在 Screen 視窗中，增加 Screen2，並在 Inspector 中 STYLE SETTINGS 對背景設定成不同的顏色。

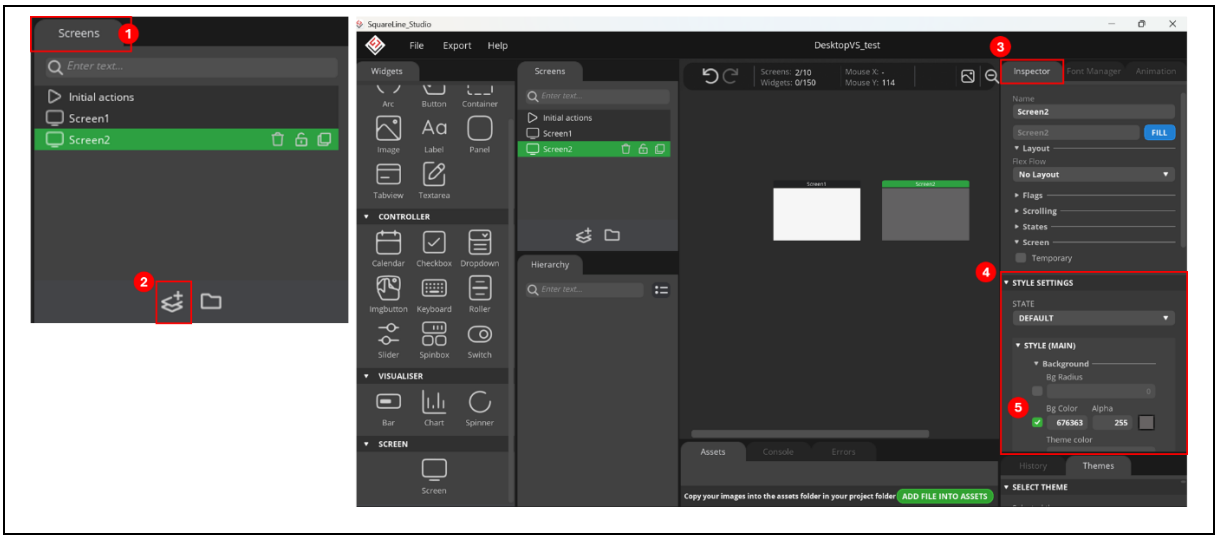



圖 1-5 增加 Screen2，並設定背景色

2) 新增 Arc 元件：點選 Screen1，以增加在 Screen1 上的物件。參考圖 1-6，從 Widgets 中點選 Arc，點選後即會出現在 Screen1 的畫布上。並參考表 1-4 在 Inspector 中設定 Arc 屬性，包括大小、數值的範圍、顯示的角度、顏色設定等。

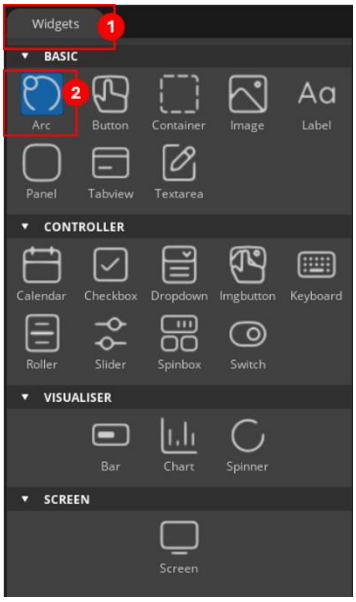
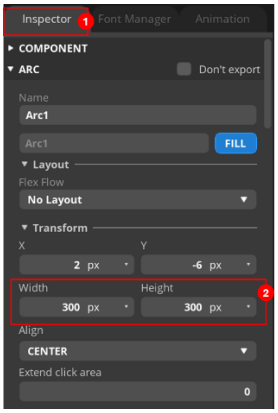
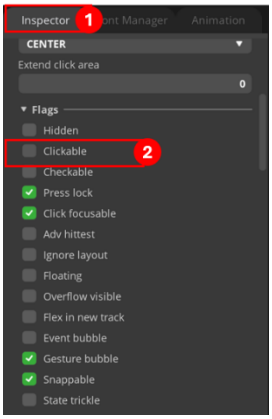
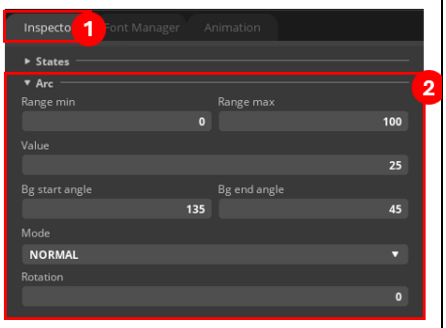



圖 1-6 增加 Widgets – Arc

| Contents | Functions | Settings |
|---|---|---|
| Width and Height | Arc 整體的寬與高 |  |
| Flags-Clickable | 設定是否能被使用者觸控 |  |
| Range, Value, Angle | Range 是 Arc 值的範圍，<br>Value 是初始顯示的值，<br>Angle 是 Arc 顯示的角度 |  |

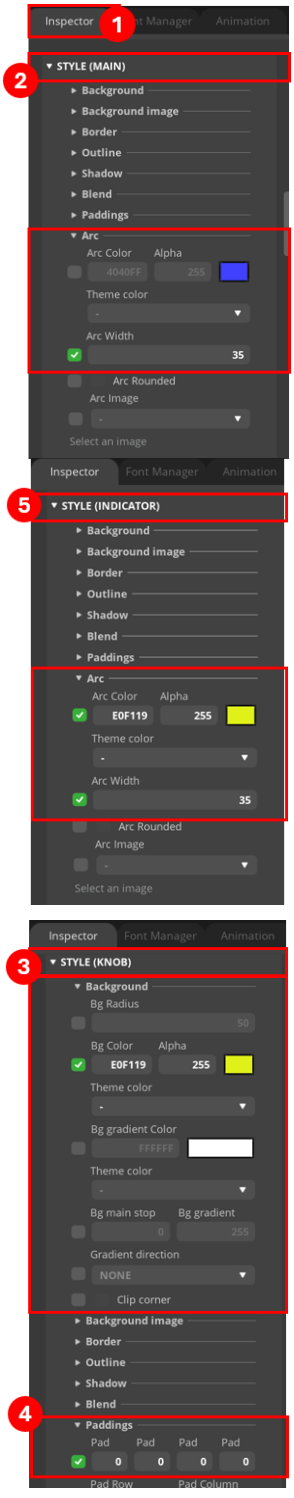
| Contents | Functions | Settings |
|---|---|---|
| STYLE | Style 中有分成三項，MAIN 代表對 Arc 主體的控制設定、KNOB 代表控制旋鈕的設計設定、INDICATOR 代表 Arc 值變動時指示的設計設定。 |  |

表 1-4 Arc 屬性設定

3) 新增 Screen1 的數值顯示：從 Widgets 中點選 Label，點選後即會新增在 Screen1 的畫布上。在畫布上拖曳移動至 Arc 中間，參考表 1-5 說明，設定 Label 相關屬性。

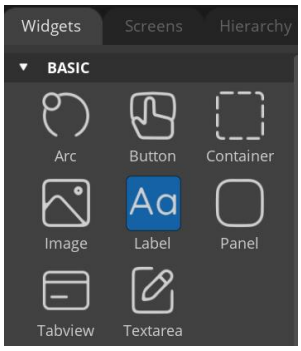



圖 1-7 增加 Widgets – Label

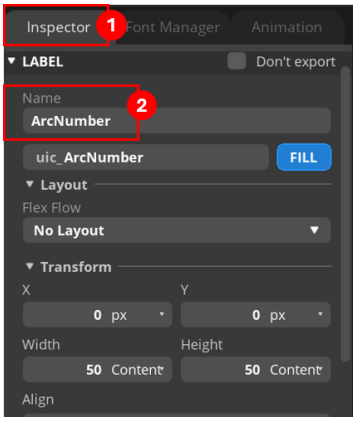
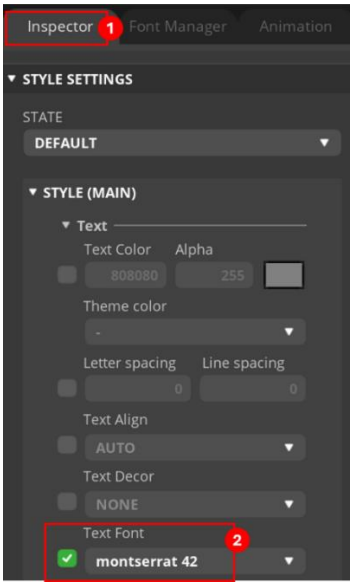
| Contents | Functions | Settings |
|---|---|---|
| Name | 可以自定義物件名稱，在物件連結時或在專案中更方便辨認 |  |
| STYLE-Text Font | Style 中可以設定 Label 的顏色以及字型，字型包括大小。 |  |





表 1-5 Label 屬性設定

4) 新增 Screen1 跳至 Screen2 的 Button：參考圖 1-8 從 Widgets 中點選 Button，並將 Button 改命名為 ”NextBTN”。若 Button 上需要顯示文字，需另外新增 Label 重疊放置在 Button 上方。

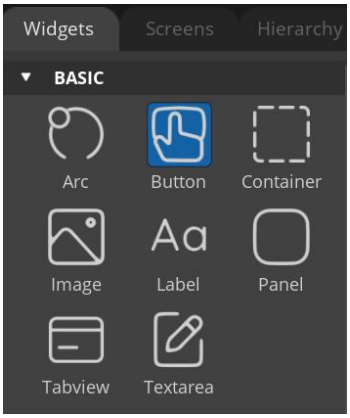



圖 1-8 增加 Widgets – Button

5) 設定當按下 Screen1 的 ”NextBTN” 後，跳至 Screen2：參考圖 1-9 步驟，點選 ”NextBTN” ，在 Inspector 屬性設定按下 ADD EVENT，Trigger 方式是 Clicked，表示按下按鈕時，Action 是選擇 Change Screen，表示表現的動作是切換畫面，加入動作後，可再進一步選擇欲切換的目標-Screen2 以及切換的方式- Move left(往左移)。

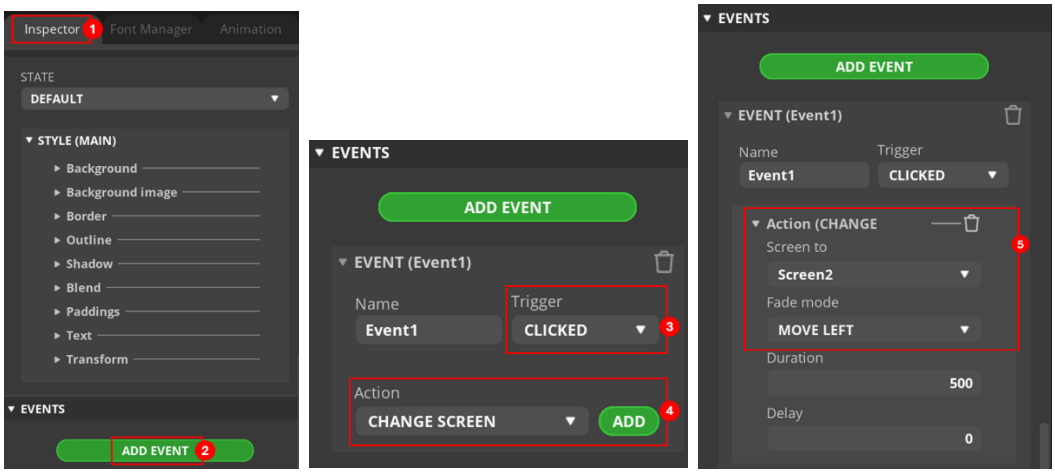



圖 1-9 按鍵動作設定

6) 增加 Screen2 的元件：點選 Screen2，參考圖 1-10 從 Widgets 中點選 Slider，並參考表 1-6 設定 Slider 相關屬性。

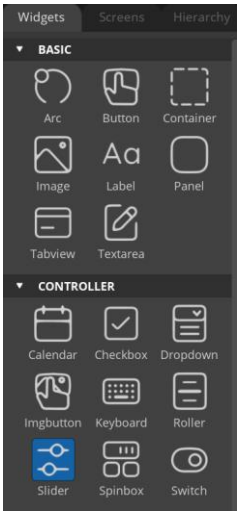



圖 1-10 增加 Widgets – Slider

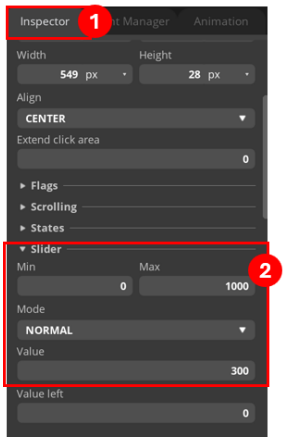
| Contents | Functions | Settings |
|---|---|---|
| Min/Max | 表示 Slider 數值的範圍 |  |
| Value | Slider 顯示的初始位置 | |



表 1-6 Slider 屬性設定

7) 新增 Screen2 的數值顯示：從 Widgets 中點選 Label，點選後即會新增在 Screen2 的畫布上。參考圖 1-7 的設定步驟，將此 Label 命名為 SliderNumber。

8) 將 Slider 的變化跟 SliderNumber 數值連動：點選 Screen2 的 Slider 物件，參考圖 1-11 步驟，在 Inspector 中按下 ADD EVENT，設定 Trigger 方式為 VALUE_CHANGED，Action 動作選為 SET VALUE FROM SLIDER，Target 則是選定為前面設定好的 SliderNumber。

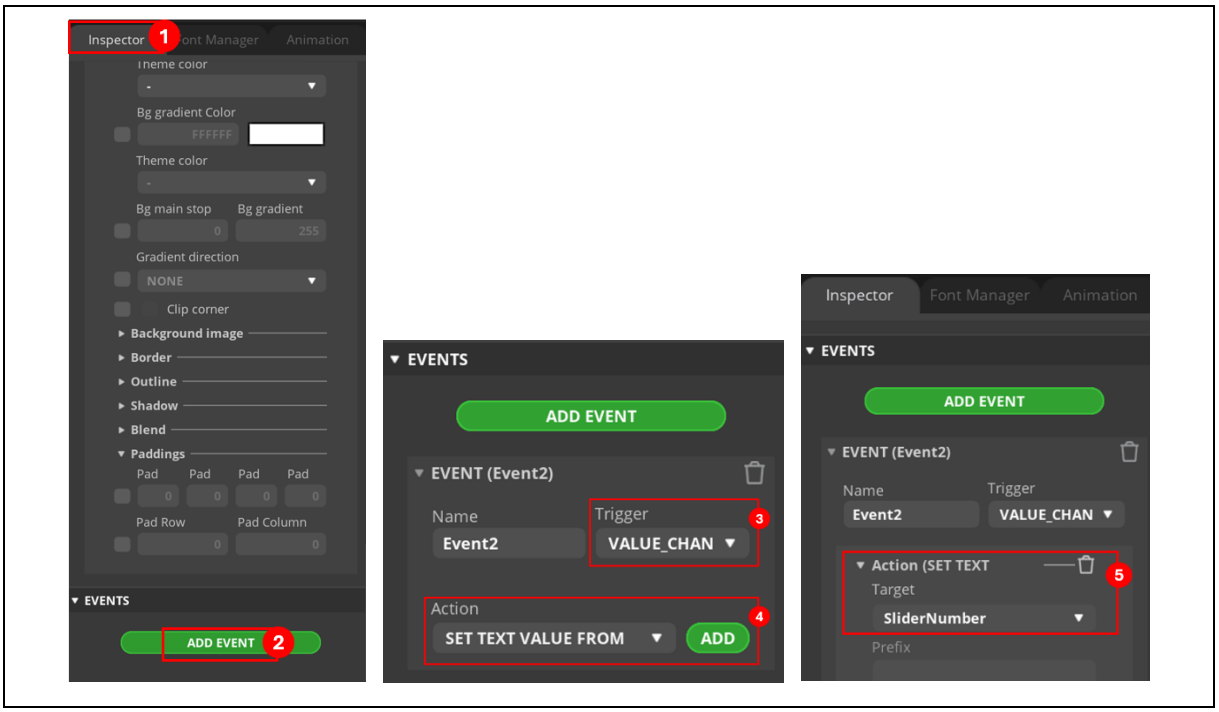



圖 1-11 Slider 與數值連動設定

9) 增加返回 Screen1 的 Button：與步驟 5)相同，新增 Button，並參考圖 1-9 設定連動。

10) Play Mode 模擬：完成設計時，可如圖 1-12 所示，按下 Play，在電腦上模擬畫面。

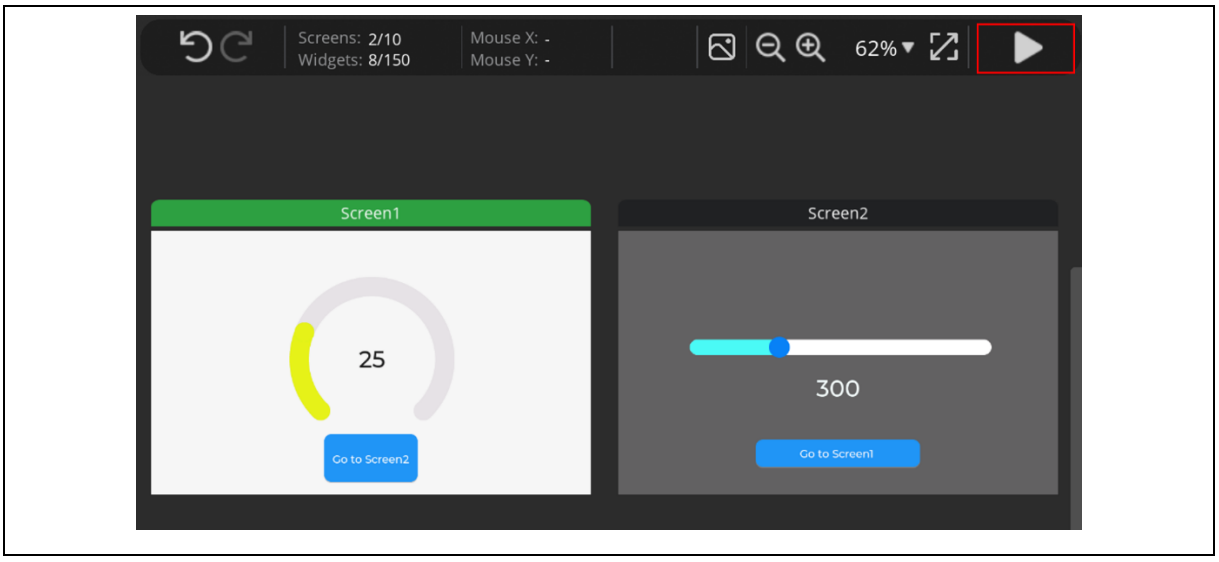



圖 1-12 Play Mode

5. 輸出並開啟程式專案：在 SLS 視窗左上角，選擇 Export，共有兩種輸出方式。Export UI Files – SLS 將只輸出 UI 設定相關檔案。本範例亦以此方式輸出，輸出流程與開啟程式專案的方式為：

1) 參考圖 1-13，若選擇 Export UI Files，會輸出的檔案包括表 1-7 所示。儲存路徑可以於 Project Setting 中設定與查詢，如圖 1-14。

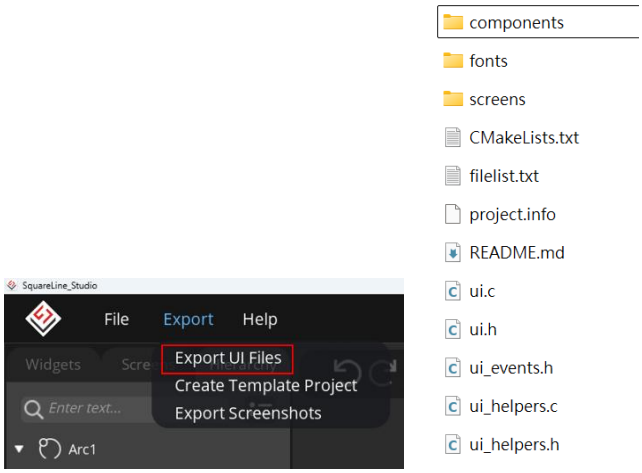



圖 1-13 Export UI Files

| **Contents** | **Notes** |
|---|---|
| Components | 存放 Component（如 Status Bar、Navigation Bar），供多個 Screen 共用。 |
| Fonts | 字型資源 |
| Screens | 每個 Screen 對應一組 ui_ScreenX.c / .h，建立畫面上所有 Widget |
| ui.c / ui.h | 是 ui 的主入口，包括 ui_init()、Screen 初始化、載入預設畫面、Widget 全域變數定義 |
| ui_events.h | event 設定為 CALL FUNCTION 時產生的函式原型宣告，須將程式執行內容由使用者自行寫於 ui_events.c |
| ui_helpers.c<br>ui_helpers.h | 由 SLS 產生的通用輔助函式 |

表 1-7 UI Files List

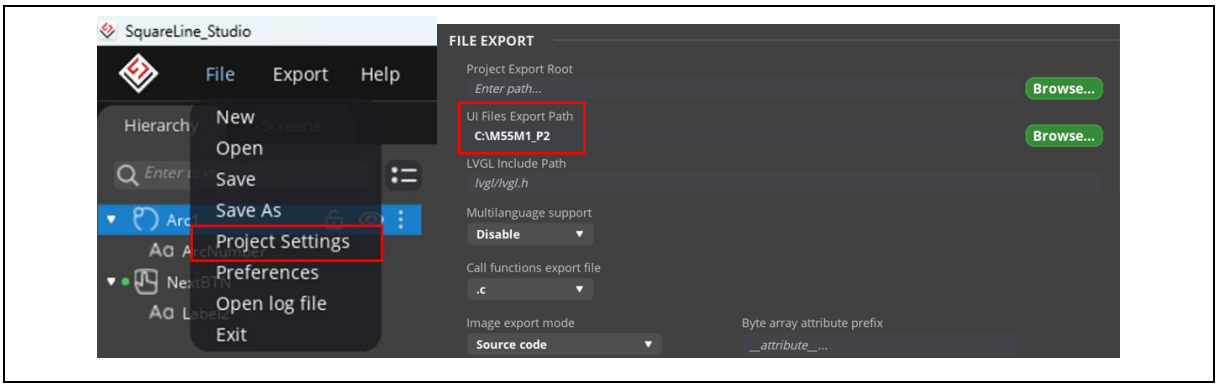



圖 1-14 Project Settings

2) 複製 SLS UI 輸出檔案至專案目錄：將 SLS 產生的所有 UI 相關檔案，完整複製到專案中的以下路徑：\board\numaker-x-m55m1d\sls_files。

3) 如章節 5 所述，開啟專案。

若使用 Keil MDK 開啟，需依照圖 1-15 步驟，將 UI 相關檔案加入專案中。

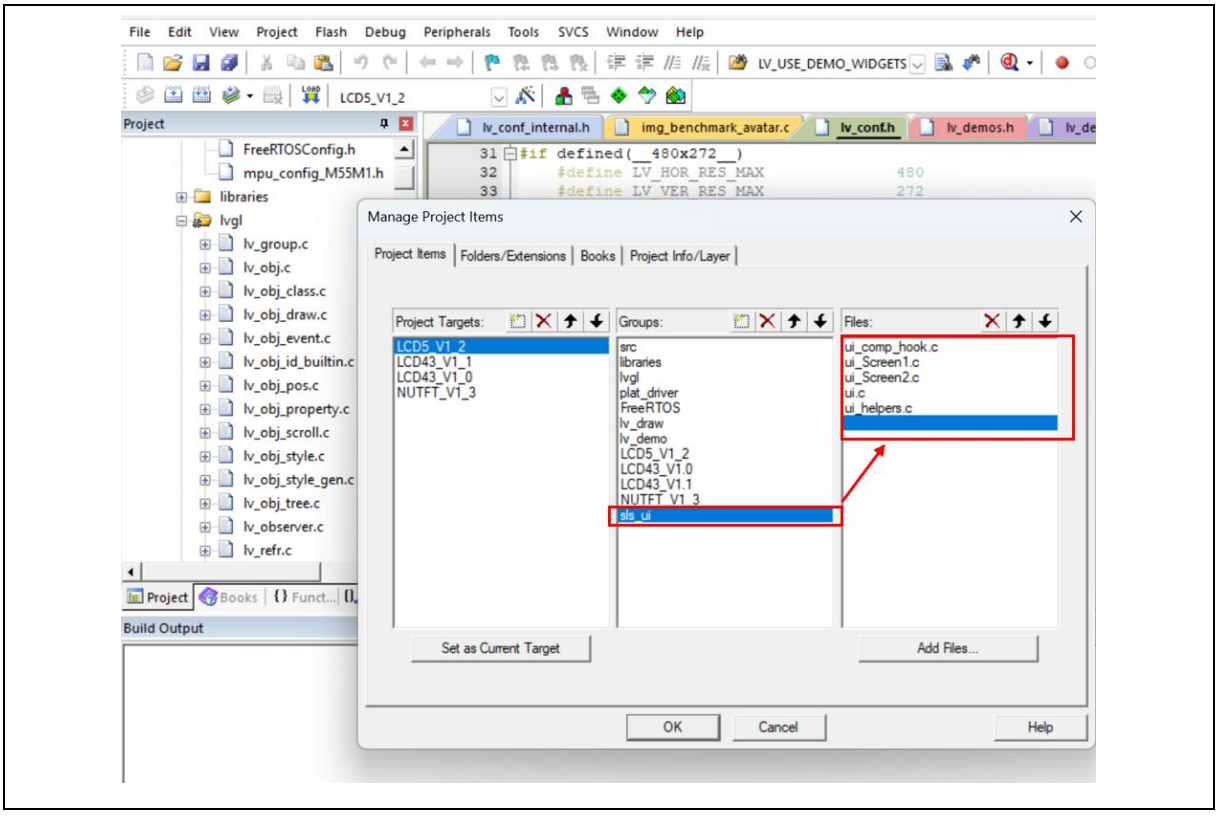



圖 1-15 於 Keil 嵌入 UI 檔案

若使用 VS Code 開啟專案，如圖 1-16 所示，若有新增 UI 檔案需編譯，需手動加入檔案於 sls_files 資料夾內，並確保.cproject.yml 編譯檔亦包括該檔案。

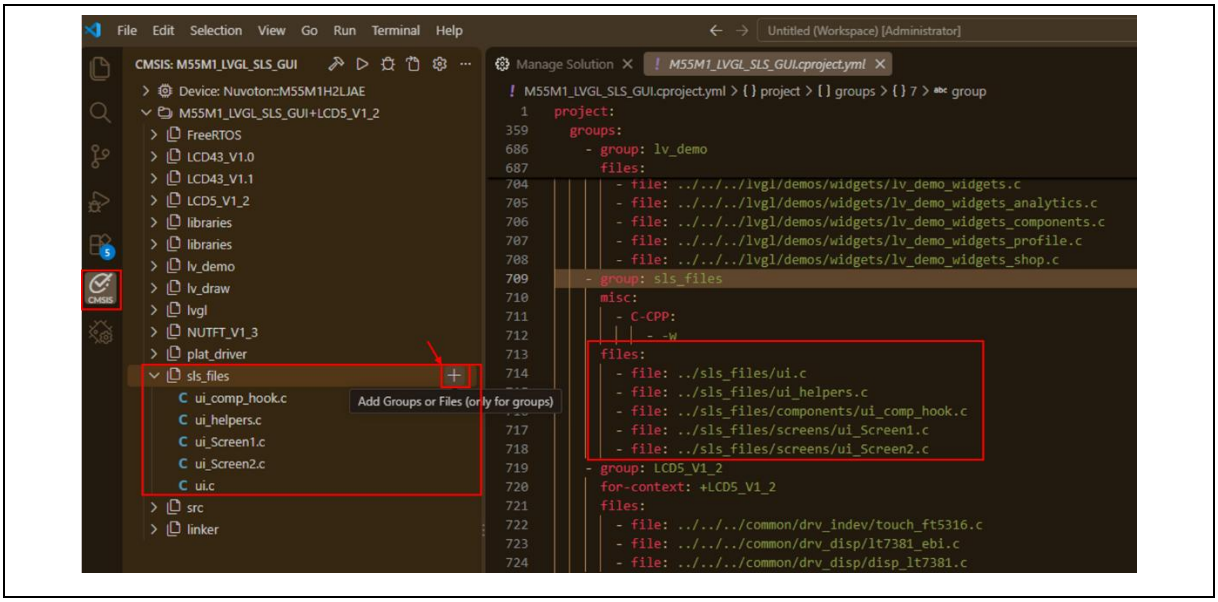



圖 1-16 於 VS Code 嵌入 UI 檔案

6. 加入數值讀取，並顯示於 Screen1 的 Label 物件與 Arc 物件：當 MCU 有接 sensor 或是計算數值，可於 task_lv.c 中寫入相關數值至欲顯示的元件，API 與相關程式流程請參考章節 2。

## 1.2 執行結果

本範例程式碼執行編譯和燒錄後，Screen1 會顯示自動播放動畫，Screen2 則是可以控制畫面，如圖 1-17 所示。

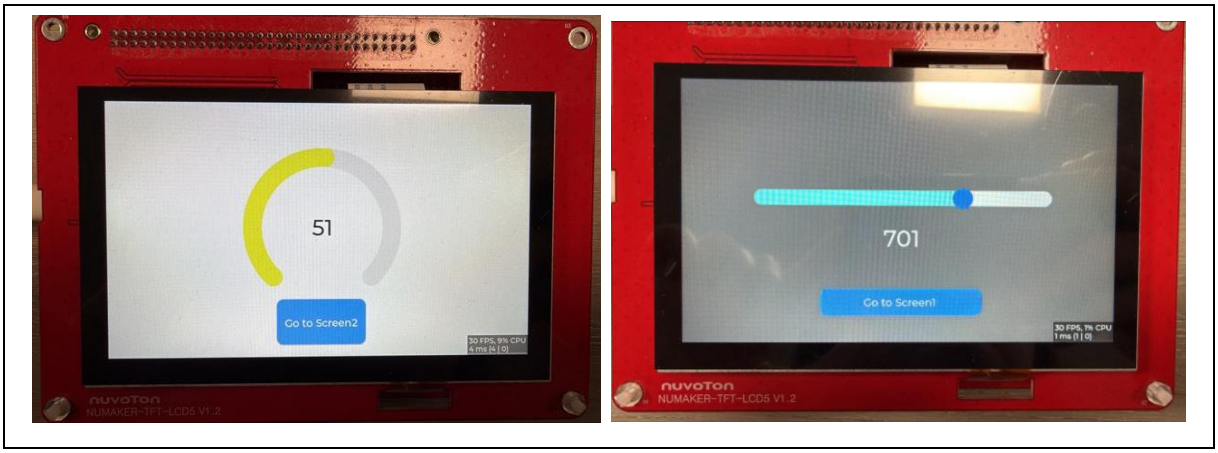



圖 1-17 執行編譯和燒錄結果

## 2. 程式碼介紹

以下將介紹此範例的關鍵程式碼部分。主程式 main.c 流程，包括三部分，分別是系統初始化、LVGL 初始化以及 UI 程式流程。

```
int main(void)
{
    sys_init();

    int task_lv_init(void);
    task_lv_init();

    xTaskCreate(sensor_task, "sensor", configMINIMAL_STACK_SIZE, NULL, tskIDLE_PRIORITY +
1, NULL);

    /* Start scheduling. */
    vTaskStartScheduler();

    for (;;);

    return 0;
}
```

1. 系統初始化 sys_init()

   此範例所使用到的硬體腳位初始化，主要包括 EBI 用於驅動 LCD、I2C1 驅動 LCD 的觸控，SPI2、PDMA0/1、TMR0、EADC0 則供其他周邊功能使用，另外使用 GDMA350，作為 LVGL 繪圖的加速功能。

```
static void sys_init(void)
{
…
    /* Enable EBI clock */
    CLK_EnableModuleClock(EBI0_MODULE);

    /* Enable I2C1 clock */
    CLK_EnableModuleClock(I2C1_MODULE);

    /* Enable SPI2 clock */
    CLK_EnableModuleClock(SPI2_MODULE);

    /* Enable PDMA clock */
    CLK_EnableModuleClock(PDMA0_MODULE);
    CLK_EnableModuleClock(PDMA1_MODULE);

    /* Select TIMER clock source */
    CLK_SetModuleClock(TMR0_MODULE, CLK_TMRSEL_TMR0SEL_HIRC, 0);

    /* Enable TIMER module clock */
    CLK_EnableModuleClock(TMR0_MODULE);

    /* EADC Analog Pin */
```

```
    CLK_EnableModuleClock(EADC0_MODULE);

    /* Select EADC peripheral clock source. */
    CLK_SetModuleClock(EADC0_MODULE, CLK_EADCSEL_EADC0SEL_PCLK0, CLK_EADCDIV_EADC0DIV(8));

    /*---------------------------------------------------------------------------------*/
    /* Init I/O Multi-function                                                         */
    /*---------------------------------------------------------------------------------*/
    SET_SPI2_SS_PA11();
 …
    SET_I2C1_SDA_PB10();
    SET_I2C1_SCL_PB11();

    SET_EBI_AD0_PA5();
  …

    SYS->GPB_MFP1 &= ~(SYS_GPB_MFP1_PB7MFP_Msk | SYS_GPB_MFP1_PB6MFP_Msk);
    SYS->GPB_MFP1 |= (SYS_GPB_MFP1_PB7MFP_EADC0_CH7 | SYS_GPB_MFP1_PB6MFP_EADC0_CH6);
    SYS->GPB_MFP2 &= ~(SYS_GPB_MFP2_PB9MFP_Msk | SYS_GPB_MFP2_PB8MFP_Msk);
    SYS->GPB_MFP2 |= (SYS_GPB_MFP2_PB9MFP_EADC0_CH9 | SYS_GPB_MFP2_PB8MFP_EADC0_CH8);

    /* Disable digital path on these EADC pins */
    GPIO_DISABLE_DIGITAL_PATH(PB, BIT6 | BIT7 | BIT8 | BIT9);

    /* Enable SysTick clock */
    CLK_EnableSysTick(CLK_STSEL_ST0SEL_HIRC_DIV2, 0);

    /*---------------------------------------------------------------------------------*/
    /* Init I/O Multi-function                                                         */
    /*---------------------------------------------------------------------------------*/
    SetDebugUartMFP();

    InitDebugUart();

    /* Enable SPIM module clock */
    CLK_EnableModuleClock(SPIM0_MODULE);

    /* Enable OTFC module clock */
    CLK_EnableModuleClock(OTFC0_MODULE);

    DMA350DevInit();

}
```

2. LVGL 初始化

LVGL 會在 FreeRTOS 排程啟動後，於 lv_hdler 任務內執行函式 lv_nuvoton_task()。

```
int task_lv_init(void)
{
    printf("stack addr = %p\n", lvgl_stack);
    xTaskCreate(lv_tick_task, "lv_tick", configMINIMAL_STACK_SIZE, NULL,
CONFIG_LV_TASK_PRIORITY, NULL);
    xTaskCreateStatic(lv_nuvoton_task, "lv_hdler", CONFIG_LV_TASK_STACKSIZE, NULL,
CONFIG_LV_TASK_PRIORITY, lvgl_stack, &lvgl_tcb);
    return 0;
}
```

函式 lv_nuvoton_task() 依序完成 LVGL 核心初始化、log 註冊、GDMA 繪圖單元註冊、時間與延遲綁定 FreeRTOS 的 tick、顯示裝置移植以對應欲顯示的 LCD 裝置、輸入裝置移植、使用者介面建立，最後建立資料更新用的 LVGL timer 並進入主迴圈。

其中與使用者介面建立有關的 ui_init()，是由 SLS 產生，會直接取得位於路徑 sls_files/ui.c 檔案進入並執行載入。

```
void lv_nuvoton_task(void *pdata)
{
    lv_init();

#if LV_USE_LOG
    lv_log_register_print_cb(lv_nuvoton_log);
#endif

#if LV_USE_DRAW_GDMA
    void lv_draw_gdma_init(void);
    lv_draw_gdma_init();
#endif

    lv_tick_set_cb(xTaskGetTickCount);
    lv_delay_set_cb(vTaskDelay);

    extern void lv_port_disp_init(void);
    lv_port_disp_init();

    extern void lv_port_indev_init(void);
    lv_port_indev_init();

    extern void ui_init(void);
    ui_init();

    /* Sample faster than the producer so no sensor value is skipped */
    lv_timer_create(sensor_ui_timer_cb, 10, NULL);

    while (1)
    {
        lv_task_handler();
        vTaskDelay(pdMS_TO_TICKS(1));
    }
}
```

3. UI 程式流程

要讓 MCU 端取得的數值，例如 EADC 轉換結果、 I²C 感測器讀值、計時器計數等，顯示在 Screen1 的弧形量表與標籤上，共需兩步驟。

1) 在 MCU 端產生數值：在 main.c 中，以 sensor_task 函式模擬一顆 I²C 感測器，模擬數值為 g_i32I2CSensorValue，此值會在 0 與 100 之間來回掃描，中間每 50 ms 前進一步，抵達兩端時停留 1 秒以便觀察。

```
static void sensor_task(void *pdata)
{
    int32_t value = 0;
    int32_t step = 1;

    (void)pdata;

    for (;;)
    {
        g_i32I2CSensorValue = value;

        if (value >= 100)
        {
            step = -1;
            vTaskDelay(pdMS_TO_TICKS(1000));
        }
        else if (value <= 0)
        {
            step = 1;
            vTaskDelay(pdMS_TO_TICKS(1000));
        }
        else
        {
            vTaskDelay(pdMS_TO_TICKS(50));
        }

        value += step;
    }
}
```

2) 在 LVGL 側結合數值：在可客製不會被 SLS 覆蓋的 task_lv.c 中，呼叫 lv_arc_set_value() 更新 Arc 數值，以及 lv_label_set_text_fmt() 更新數字。

其餘 Widgets 的 API 可透過 LVGL 官方文件查詢。

```
static void sensor_ui_timer_cb(lv_timer_t *timer)
{
    static int32_t last_value = -1;
    int32_t value = g_i32I2CSensorValue;

    (void)timer;

    if (value == last_value) return;

    last_value = value;

    if (ui_Arc1)      lv_arc_set_value(ui_Arc1, value);
    if (ui_ArcNumber) lv_label_set_text_fmt(ui_ArcNumber, "%d", (int)value);
}
```

# 3. 軟體與硬體需求

## 3.1 軟體需求

- BSP 版本
  - M55M1_Series_BSP_CMSIS_V3.01.005
- IDE 版本
  - Keil uVision 5.43
  - VS Code 1.138.0
- SquareLine Studio 版本
  - SquareLine Studio 1.6.2

## 3.2 硬體需求

- 電路元件
  - NuMaker-X-M55M1D

# 4. 目錄資訊

目錄結構如下圖所示。

| | |
|---|---|
| EC_M55M1_LVGL_SLS_GUI_V1.00 | |
| board | Board-level files including the main application. |
| numaker-x-m55m1d | |
| Keil | Keil Project |
| VSCode | VSCode Project |
| components | task_lv |
| sls_files | ui files generated from SLS |
| bsp | Board-level for board support package |
| m55m1 | |
| common | Shared source code, including display drivers, drawing, touch input. |
| lvgl | Open-source LVGL graphics library |
| SLS | SquareLine Studio project file |
| EC_M55M1_LVGL_SLS_GUI.spj | |
| thirdparty | Third-party components, primarily the FreeRTOS library |

圖 4-1 目錄資訊

# 5. 範例程式執行

## 5.1 Keil MDK

1. 在 Windows 檔案總管中進入 **Keil** 專案資料夾，再開啟 *M55M1_LVGL_SLS_GUI.uvprojx* 專案檔。
2. 編譯程式碼
3. 下載程式碼
4. 進入偵錯模式
   - 執行程式碼

## 5.2 VS Code

1. 在 **VS Code** IDE 中開啟 **File** 選單
   - 選取 **Open Folder**
   - 選取專案目錄下的 **VSCode** 資料夾
2. 選擇 CMSIS
   - 點選 **Build solution** 編譯專案
   - 點選 **Load & Run application** 下載映像檔至目標晶片
   - 點選 **Load & Debug application** 進入偵錯模式

# 6. 修訂紀錄

| Date | Revision | Description |
|---|---|---|
| 2026.09.30 | 1.00 | 初始發布。 |

## Important Notice

**Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".**

**Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.**

**All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.**

*Please note that all data and specifications are subject to change without notice.*
*All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.*