SuRG Start Here Guide

SuRG42-CP

Introduction

At Future Designs, Inc. our goal is to make it easy for our customers to get their projects up and running as quickly as possible. In this guide, you will develop a simple graphical user interface (GUI) on the SuRG platform to demonstrate how to use some of the core features of LVGL and SuRG. Using this "Hello World" type of walkthrough as a starting point, we hope to shorten the learning curve for GUI development. Aside from this document, there are many additional resources available at https://www.teamfdi.com/product/surg42-cp/. If you ever need more help, contact us at support@TeamFDI.com and we will be happy to assist you.

Hardware Used in This Guide: (Included in Kit)

- 4.2" PCAP Touch Screen LCD GUI Production Module (PN: SuRG42-CP)
- One USB Type A to USB Type C Cable
- AC to 5V USB Wall Power Adapter
- High Speed microSDHC Card (4GB or larger)
- J-Link Debugger/Programmer Probe (PN: J-link Lite CortexM-9)
- Tag-Connect 10 to 6 pin Cable with leg clamps (PN: TC2030-CTX 6-Pin Cable for ARM Cortex)
- One USB Type A to USB Type Mini-B Cable
- One SuRG Demo Board with Cable

Software Used in This Guide:

(Installation and usage instructions are provided within the guide)

- SEGGER J-Link Software
- SuRG42-CP Source
- STM32CubeIDE 1.19.0

Files Used in This Guide:

(Installation and usage instructions are provided within this guide)

- SuRG Project Maker
- Start Here Project Files (Generated with the SuRG Project Maker)

Contents

Document Revision History

Number	Change	Author	Manager	Date
1.0	Initial Release	Caleb Cox	Todd DeBoer	11/17/2025
1.1	Specification Updates	Todd DeBoer	Todd DeBoer	03/18/2026

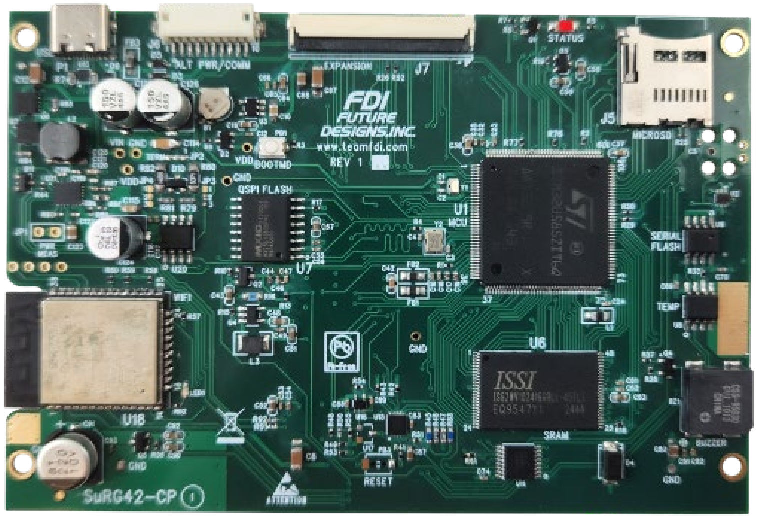
Important Legal Information

For more information on FDI or our products please visit https://www.teamfdi.com/.

FDI Document Part Number: MA00112
Revision: 1.1, 3 March 2026
Printed in the United States of America

1. *Hardware Verification*

The SuRG kit comes with a pre-installed SD card that contains files (Demos and images) required for some demo applications to run. FDI recommends that you visit the documentation tab of the product page of your SuRG to get the latest documentation.



Power is supplied via the USB power adapter and cable provided in the kit.

1. Connect the USB cable to the USB C connector P1 on the SuRG board assembly.
2. Connect the other end of the USB cable to the provided universal AC power supply's 5V USB power output.

To verify Startup Procedure, and USB-C power input (P1), please look at Sections **6**, and **10.d** in SuRG42-CP User's Manual(https://tinyurl.com/SuRG-UM)

The following screen will appear once power has been connected to the device:

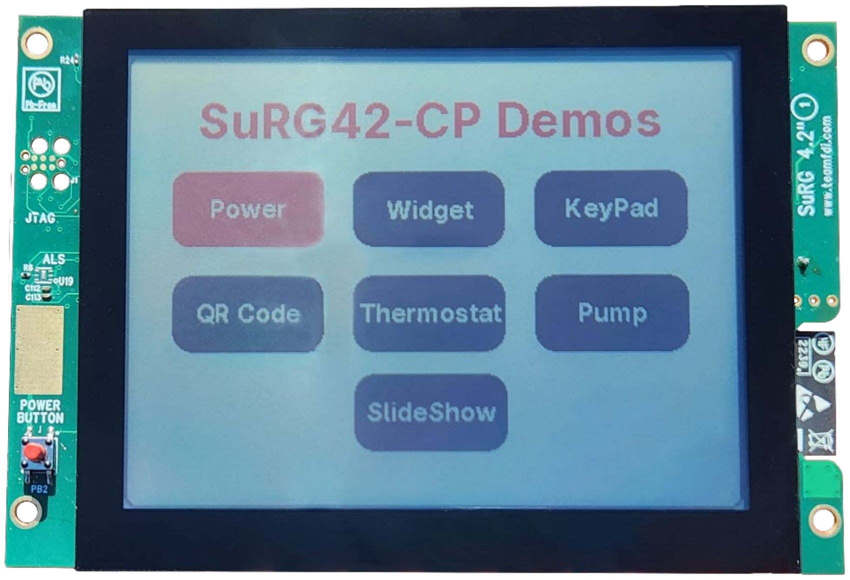


Figure 2: *Out-Of-Box (OOB) Demo*

You can now explore the out-of-box demos. Test the GUI by selecting "Thermostat" and dragging the thermostat slider or dial on the screen with your finger.

2. *Software Installation*

A. IDE Installation

All SuRG projects require an STM development environment to compile and debug the projects. SuRG example projects were developed in STM32CubeIDE on the STM32U585. Download STM32CubeIDE at the link provided below and install according to the instructions provided on their website.

- STM32CubeIDE (v1.19.0 or later):
  https://tinyurl.com/4e3m8emt
  This URL contains downloads for:
  - Windows
  - Linux (Generic)
  - Mac

B. J-Link Installation

SuRG projects were designed to be downloaded and debugged using a J-Link debug probe (J-Link debug probe purchased separately). The SEGGER version 8.38 or above driver must be installed on your computer to continue. STM32CubeIDE will include the minimum J-Link drivers, but the full "Software and Documentation" package should be downloaded to access all SEGGER J-Link applications (including J-Link configurator and RTT applications) and to receive support for the newest J-Link models. Download and install the software from the link below:

- SEGGER J-Link Software for Windows, Mac, and Linux:
  - https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack

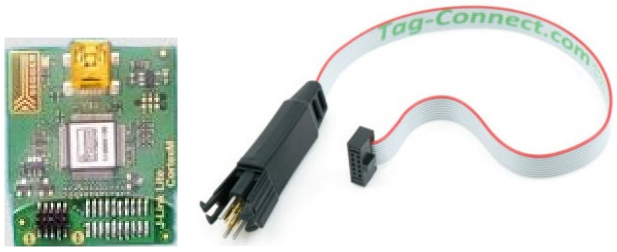
C. SuRG Example Project Installation

- SuRG (v1.00.00): https://sourceforge.net/projects/fdi-surg/files

Extract the SuRG 7z, or zip file, to the desired location. This folder contains the SuRG source library, helpful tools, documentation, project files referenced in this guide, a branch of LVGL modified for use with SuRG, and a demo project.

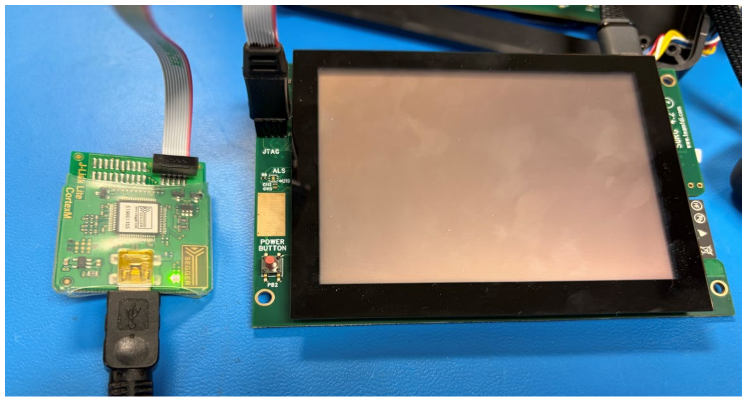
3. *Connecting the J-Link Debugger to the SuRG Board Assembly*

Figure 3:
Jlink Hardware



1. Connect the J-Link Cortex-M V9 to a PC with a USB A to Type Mini-B cable.
2. Connect the 6-pin side of the 10-to-6-pin JTAG cable to the JTAG connector (J1) on the top side of the SuRG, and the 10-pin side on the J-Link Cortex-M V9. The J-Link Cortex-M V9 , USB cable, and 10-to-6-pin Tag-Connect Cable for the SuRG are supplied in the kit.
3. If not already done, connect the SuRG to the power supply with the second included USB cable.

Figure 5
Jlink hardware connected to SuRG42-CP



For Pinout Diagram and description of Tag connect (J1), please see section **11.a** in the SuRG42-CP User's Manual(https://tinyurl.com/SuRG-Users-Manual)

4. Developing a Simple GUI Application with LVGL

A. LVGL Introduction

LVGL (https://lvgl.io/) is a free and open-source software library that is processor independent. LVGL enables you to easily add graphics to an application and is included with SuRG. Some of the features of LVGL include:

- Basic drawing functions such as drawing lines, squares, circles and polygons.
- More complex functions such as managing windows, button widgets, list-view widgets, edit widgets, etc.
- Displaying images.
- Support for multiple displays*
- Support for multiple layers and transparency settings.
- Control of GUI by mouse and touch screen.

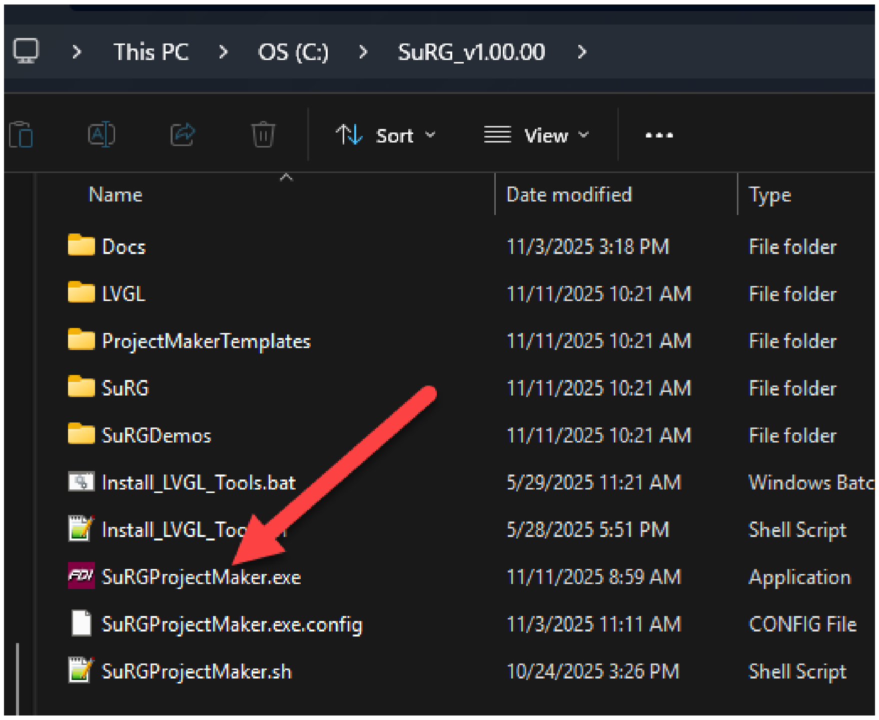
***NOTE:** With the SuRG42-CP, the framebuffer color depth will always be set to 8-bits per pixel, and not all bits are used. Per LVGL documentation, each display needs to have the same color depth as defined in LV_COLOR_DEPTH. However, if the displays differ in this regard, the rendered image can be converted to the correct format in the drivers “flush_cb”.

This section of the guide will take a simple screen and develop it into a GUI to demonstrate a few of the core features of SuRG and the LVGL library.

B. Using the SuRG Project Maker

SuRG comes packaged with a Project Maker that streamlines project creation and gives users a quick place to start and build up from.

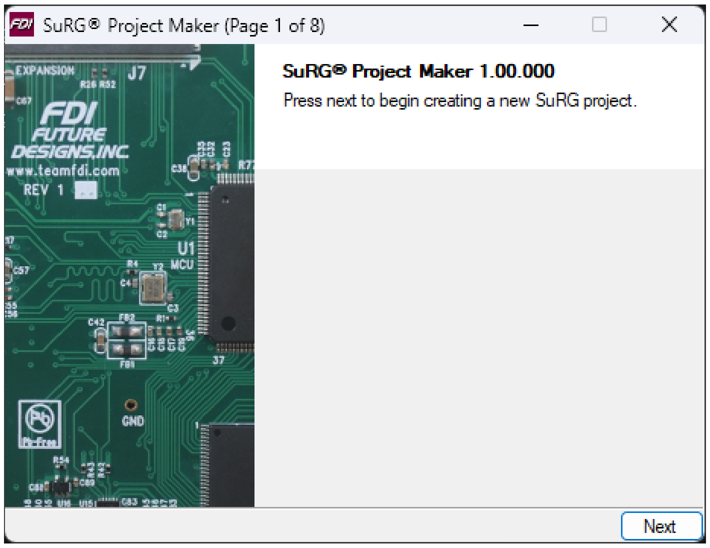
1. In your SuRG download folder, locate “SuRGProjectMaker.exe”, and run it.



This PC > OS (C:) > SuRG_v1.00.00 >

Name	Date modified	Type
Docs	11/3/2025 3:18 PM	File folder
LVGL	11/11/2025 10:21 AM	File folder
ProjectMakerTemplates	11/11/2025 10:21 AM	File folder
SuRG	11/11/2025 10:21 AM	File folder
SuRGDemos	11/11/2025 10:21 AM	File folder
Install_LVGL_Tools.bat	5/29/2025 11:21 AM	Windows Batc
Install_LVGL_Too	5/28/2025 5:51 PM	Shell Script
SuRGProjectMaker.exe	11/11/2025 8:59 AM	Application
SuRGProjectMaker.exe.config	11/3/2025 11:11 AM	CONFIG File
SuRGProjectMaker.sh	10/24/2025 3:26 PM	Shell Script

2. A dialog will appear to create your project.



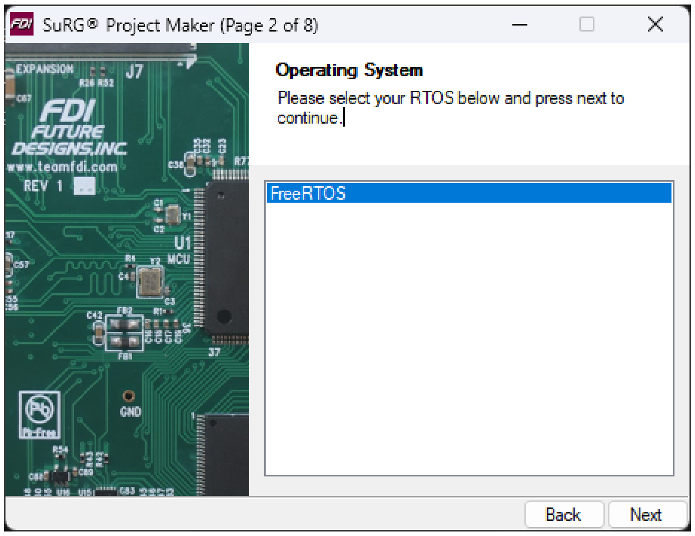
SuRG® Project Maker (Page 1 of 8)

SuRG® Project Maker 1.00.000

Press next to begin creating a new SuRG project.

3. Click <Next>.

4. On Page 2 of 8, select <FreeRTOS>, then click <Next>.



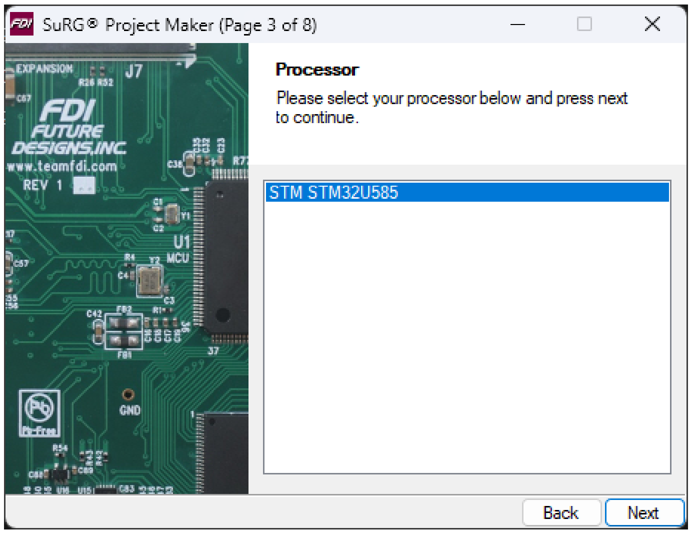
SuRG® Project Maker (Page 2 of 8)

Operating System

Please select your RTOS below and press next to continue.

FreeRTOS

5. On Page 3 of 8, select <STM STM32U585>, then click <Next>.



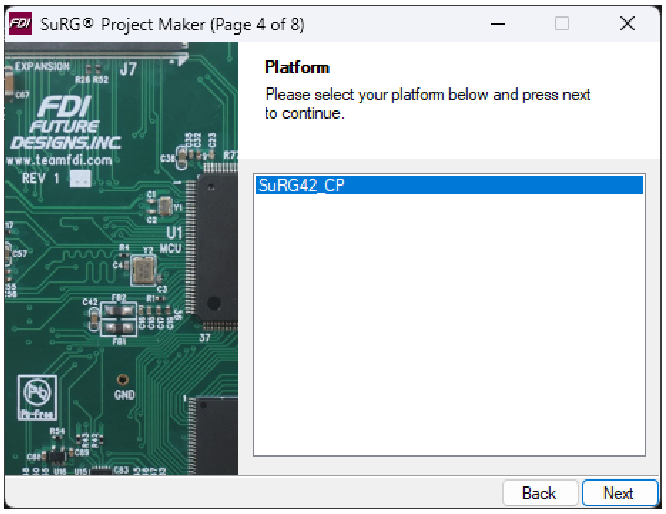
SuRG® Project Maker (Page 3 of 8)

Processor

Please select your processor below and press next to continue.

STM STM32U585

6. On Page 4 of 8, select <SuRG42_CP>, then click <Next>.



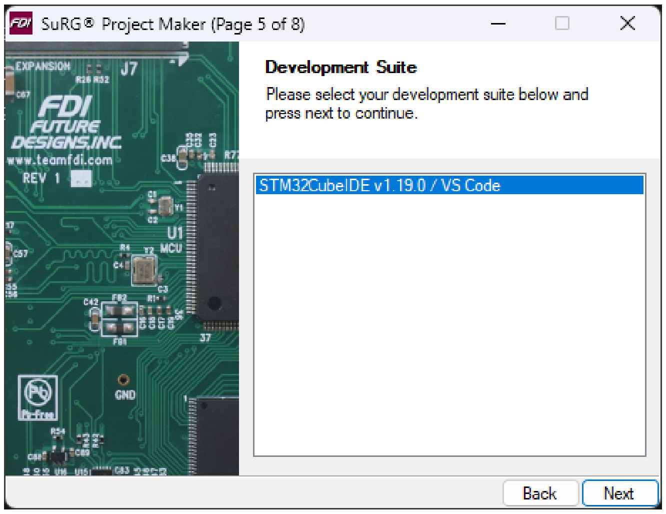
SuRG® Project Maker (Page 4 of 8)

Platform

Please select your platform below and press next to continue.

SuRG42_CP

7. On Page 5 of 8, select <STM32Cube"IDE v1.19.0 / VS Code>, then click <Next>.



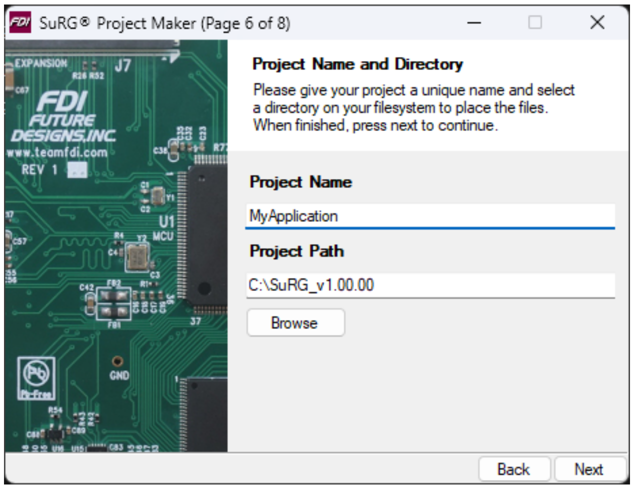
SuRG® Project Maker (Page 5 of 8)

Development Suite

Please select your development suite below and press next to continue.

STM32CubeIDE v1.19.0 / VS Code

8. On Page 6 of 8, give your project the name "MyApplication" and leave the Project Path as default. The default location is where the Project Maker .exe was launched from.



Project Name and Directory

Please give your project a unique name and select a directory on your filesystem to place the files. When finished, press next to continue.

Project Name

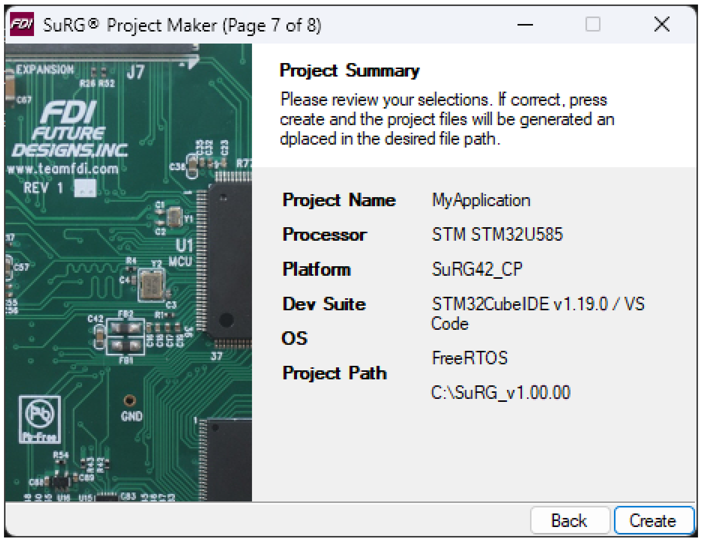
MyApplication

Project Path

C:\SuRG_v1.00.00

9. Click <Next>.

10. On Page 7 of 8, verify your settings look correct, then click <Create> to generate the project.

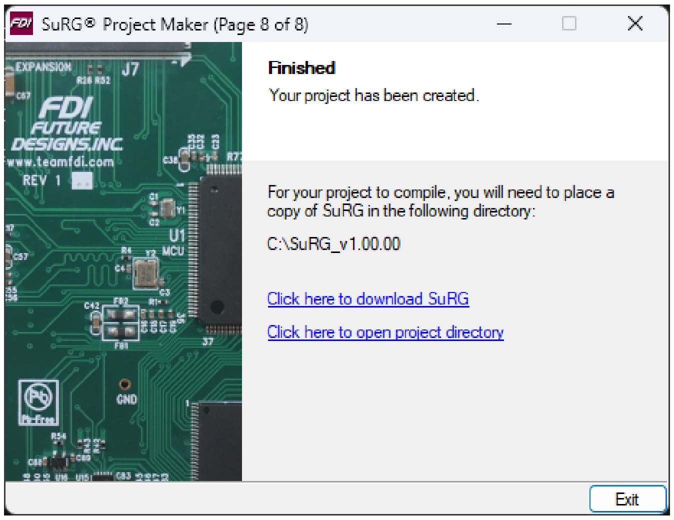


Project Summary

Please review your selections. If correct, press create and the project files will be generated an dplaced in the desired file path.

Project Name	MyApplication
Processor	STM STM32U585
Platform	SuRG42_CP
Dev Suite	STM32CubeIDE v1.19.0 / VS Code
OS	FreeRTOS
Project Path	C:\SuRG_v1.00.00

11. After a very brief pause, your project will be created and Page 8 of 8 will provide you a couple URLs – one for downloading SuRG, and another for opening your project directory.


SuRG® Project Maker (Page 8 of 8)

Finished

Your project has been created.

For your project to compile, you will need to place a copy of SuRG in the following directory:

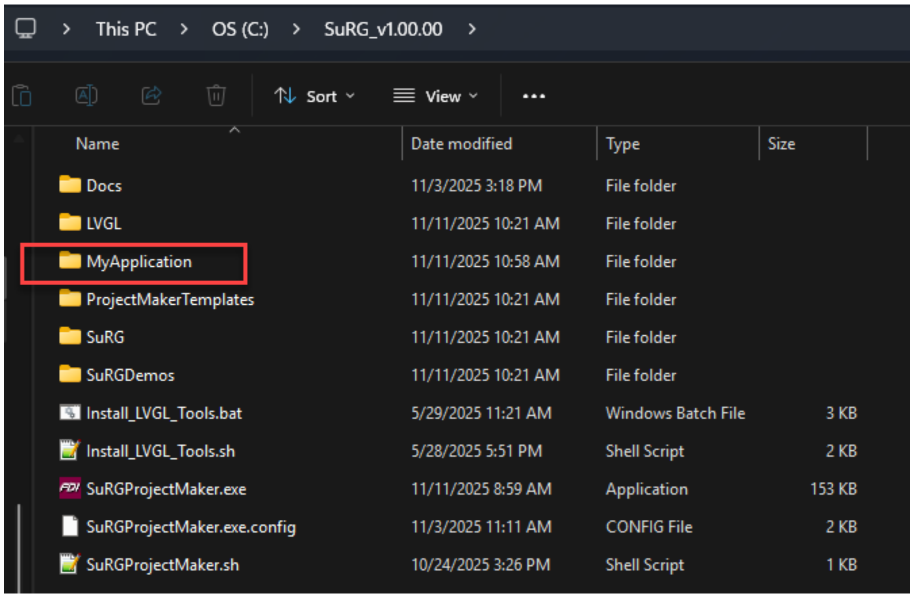
C:\SuRG_v1.00.00

Click here to download SuRG

Click here to open project directory

Exit

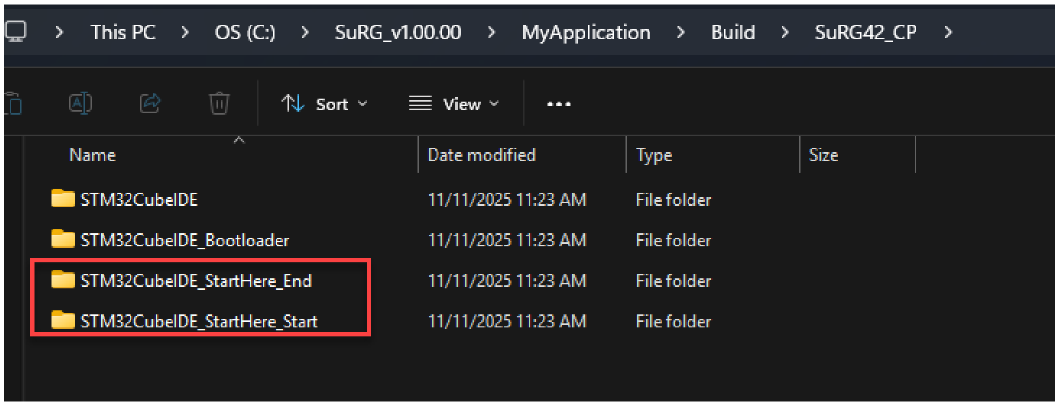
12. Since the project path was left as the default location, look again at C:/SuRG_v1.00.00 and see a folder for your project "MyApplication" was generated:


This PC > OS (C:) > SuRG_v1.00.00 >

Name	Date modified	Type	Size
Docs	11/3/2025 3:18 PM	File folder	
LVGL	11/11/2025 10:21 AM	File folder	
MyApplication	11/11/2025 10:58 AM	File folder	
ProjectMakerTemplates	11/11/2025 10:21 AM	File folder	
SuRG	11/11/2025 10:21 AM	File folder	
SuRGDemos	11/11/2025 10:21 AM	File folder	
Install_LVGL_Tools.bat	5/29/2025 11:21 AM	Windows Batch File	3 KB
Install_LVGL_Tools.sh	5/28/2025 5:51 PM	Shell Script	2 KB
SuRGProjectMaker.exe	11/11/2025 8:59 AM	Application	153 KB
SuRGProjectMaker.exe.config	11/3/2025 11:11 AM	CONFIG File	2 KB
SuRGProjectMaker.sh	10/24/2025 3:26 PM	Shell Script	1 KB

13. Navigate into MyApplication > Build > SuRG42_CP.

14. You will see 4 projects here:


This PC > OS (C:) > SuRG_v1.00.00 > MyApplication > Build > SuRG42_CP >

Name	Date modified	Type	Size
STM32CubeIDE	11/11/2025 11:23 AM	File folder	
STM32CubeIDE_Bootloader	11/11/2025 11:23 AM	File folder	
STM32CubeIDE_StartHere_End	11/11/2025 11:23 AM	File folder	
STM32CubeIDE_StartHere_Start	11/11/2025 11:23 AM	File folder	

"STM32CubeIDE" and "STM32CubeIDE_Bootloader" are the standard demo projects intended as quick start points for actual applications.

The "STM32CubeID_StartHere_Start" project however is the focus on this guide and is intended to make new SuRG users comfortable with the software before diving into a bigger project.

By the end of this guide, the user should have a project that looks and runs identically to the one in "STM32CubeID_StartHere_End".

NOTE: All 4 projects reference the same "Source" folder where much of the source code resides, so changes you make in, for example, "STM32CubeID_StartHere_Start" will affect "STM32CubeID_StartHere_End".

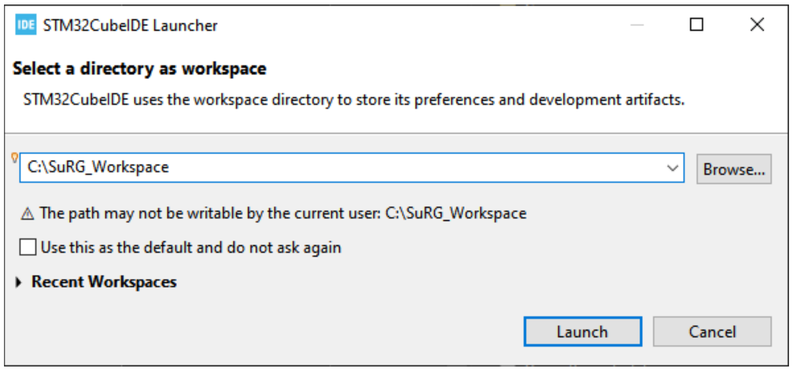
If you want to see how the final application should look at the end of this walkthrough, either create a second project using the Project Maker or build and debug the STM32CubeID_StartHere_Start project first.

C. Building and Running the Initial StartHere_Start Project

Before digging into the code of the project, examine the output by building and running the project as outlined in the steps below:

NOTE: The following sections refer to the SuRG files from SourceForge. In the following instructions, the SuRG source files were unpackaged to the C:\SuRG_Workspace folder on a Windows PC.

1. Start STM32CubeIDE.
2. Select a workspace where you want to build your SuRG application. For example: enter "C:\SuRG_Workspace".



Select a directory as workspace

STM32CubeIDE uses the workspace directory to store its preferences and development artifacts.

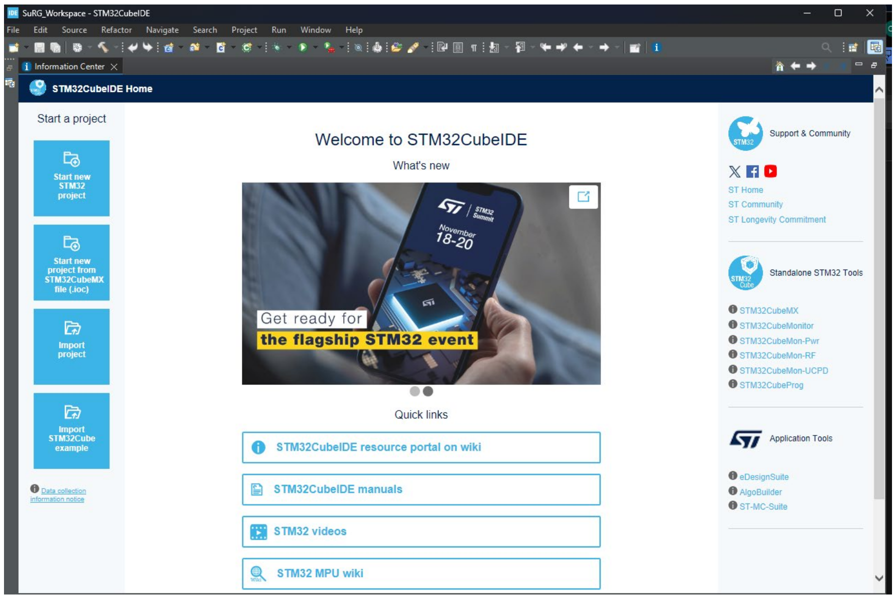
C:\SuRG_Workspace

⚠ The path may not be writable by the current user: C:\SuRG_Workspace

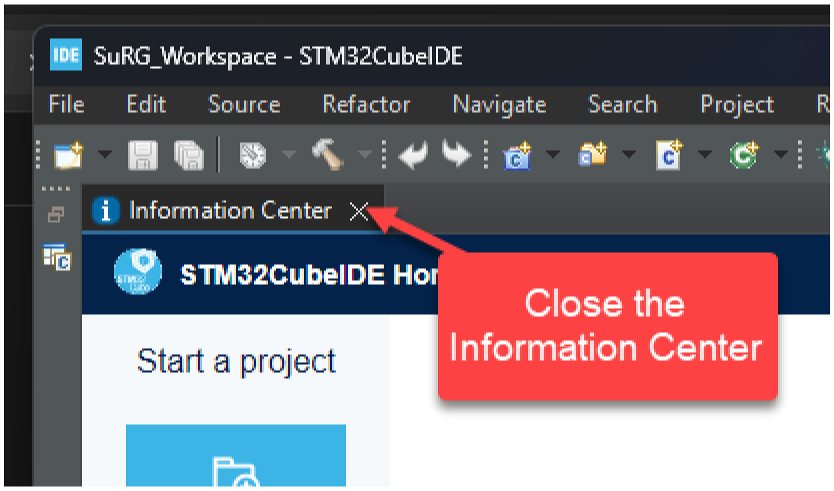
☐ Use this as the default and do not ask again

Figure 6
Choosing a Workspace

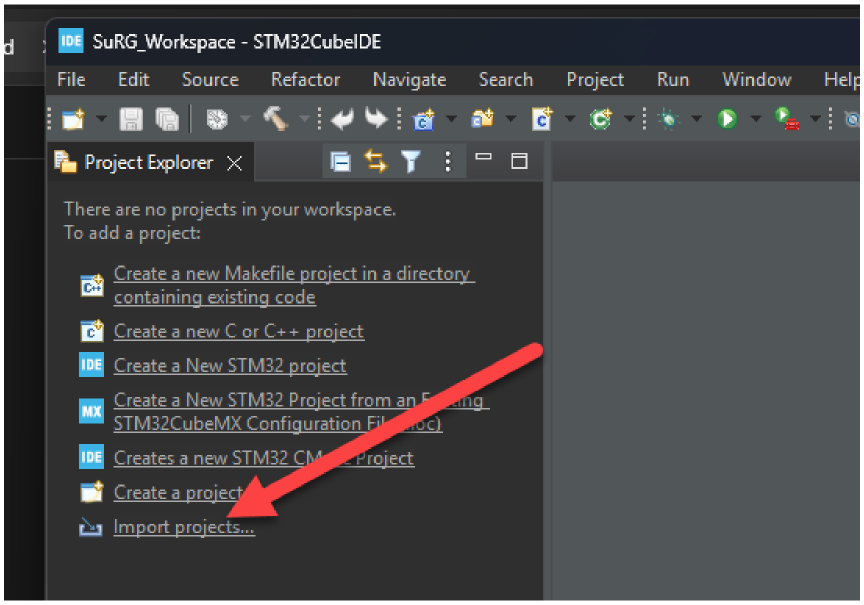
3. If this is your first time opening this workspace, you will be greeted with a welcome screen:



4. If this welcome screen appears, close the “Information Center” tab:


SuRG_Workspace - STM32CubeIDE
File Edit Source Refactor Navigate Search Project
Information Center
STM32CubeIDE Hor
Start a project
Close the Information Center

5. The Project Explorer panel will now be visible. Click <Import Projects…>

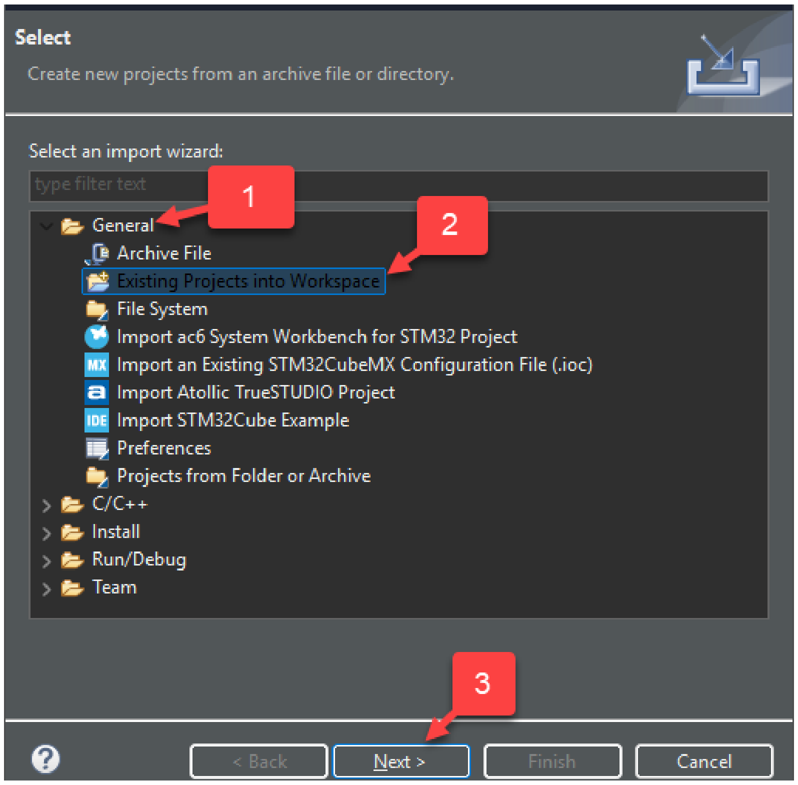

SuRG_Workspace - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer

There are no projects in your workspace.
To add a project:

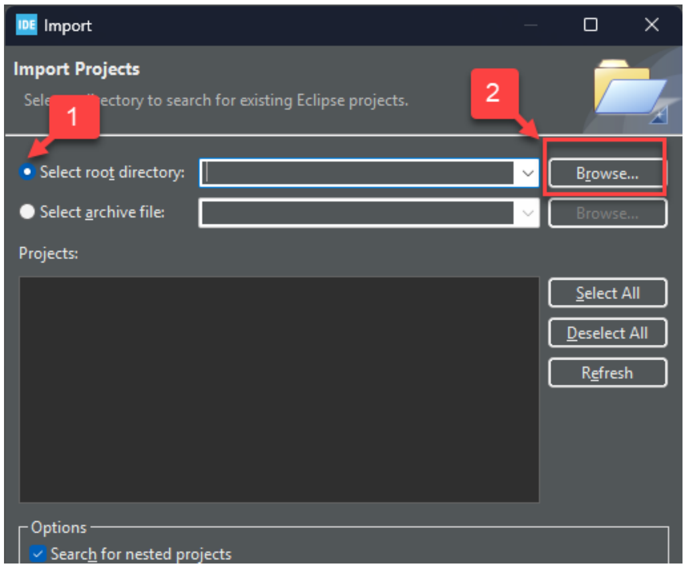
- Create a new Makefile project in a directory containing existing code
- Create a new C or C++ project
- Create a New STM32 project
- Create a New STM32 Project from an Existing STM32CubeMX Configuration File (.ioc)
- Creates a new STM32 CMake Project
- Create a project...
- Import projects...

NOTE: If the project explorer does not open for some reason, you may still import a project by clicking “File > Import…”

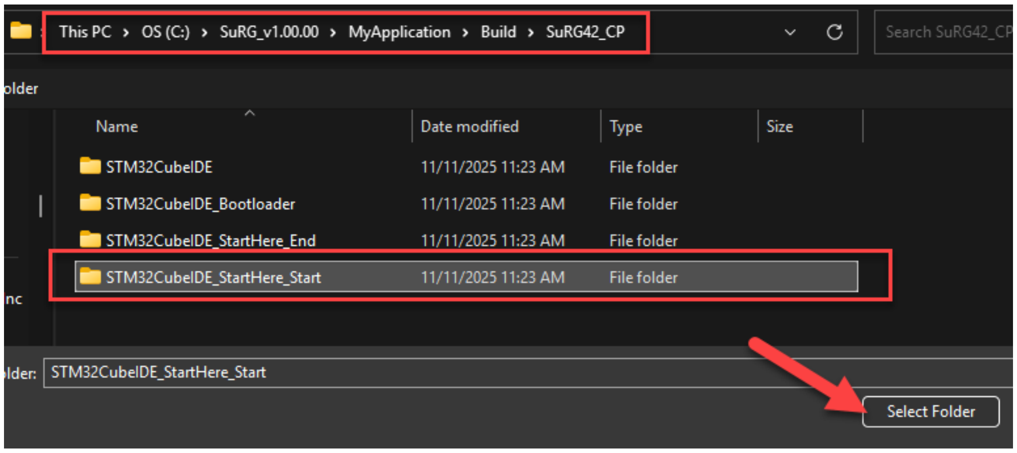
6. In the Selection dialog:
   a. Click General
   b. Existing Projects into Workspace
   c. Next


Select
Create new projects from an archive file or directory.
Select an import wizard:
type filter text
1
2
General
Archive File
Existing Projects into Workspace
File System
Import ac6 System Workbench for STM32 Project
Import an Existing STM32CubeMX Configuration File (.ioc)
Import Atollic TrueSTUDIO Project
Import STM32Cube Example
Preferences
Projects from Folder or Archive
C/C++
Install
Run/Debug
Team
3
< Back
Next >
Finish
Cancel

7. In the Import Projects dialog,
   tick the “Select root directory” radio button,
   and click <Browse...>



8. Navigate into the SuRG42_CP folder and select the "STM32CubeIDE_StartHere_Start" folder.

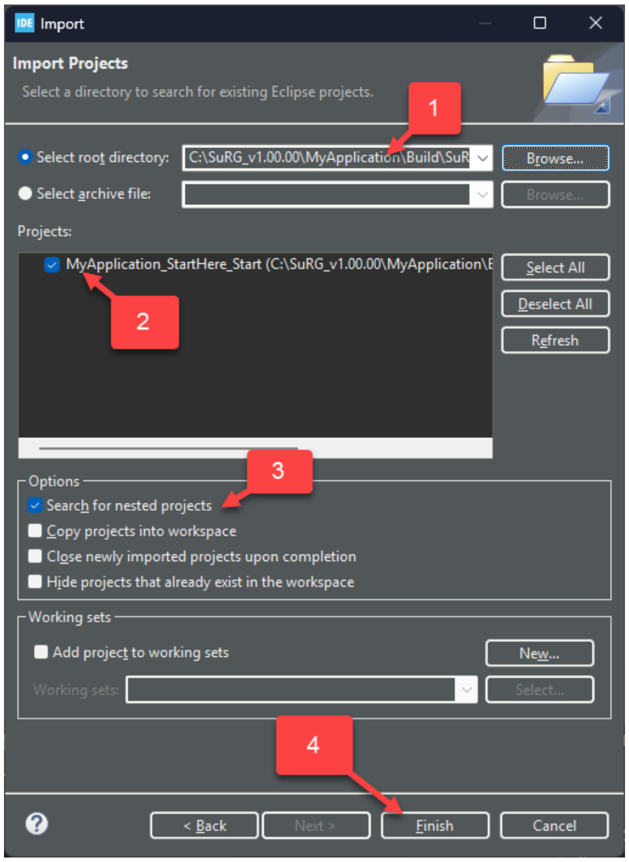

This PC › OS (C:) › SuRG_v1.00.00 › MyApplication › Build › SuRG42_CP

Name	Date modified	Type	Size
STM32CubeIDE	11/11/2025 11:23 AM	File folder	
STM32CubeIDE_Bootloader	11/11/2025 11:23 AM	File folder	
STM32CubeIDE_StartHere_End	11/11/2025 11:23 AM	File folder	
STM32CubeIDE_StartHere_Start	11/11/2025 11:23 AM	File folder	

lder: STM32CubeIDE_StartHere_Start

9. Once you have selected the folder, make the following selections in the Import Projects dialog:

Figure 9: *Finalizing the Import*


Import Projects

Select a directory to search for existing Eclipse projects.

Select root directory: C:\SuRG_v1.00.00\MyApplication\Build\SuR

Select archive file:

Projects:

☑ MyApplication_StartHere_Start (C:\SuRG_v1.00.00\MyApplication\E

Options

☑ Search for nested projects

☐ Copy projects into workspace

☐ Close newly imported projects upon completion

☐ Hide projects that already exist in the workspace

Working sets

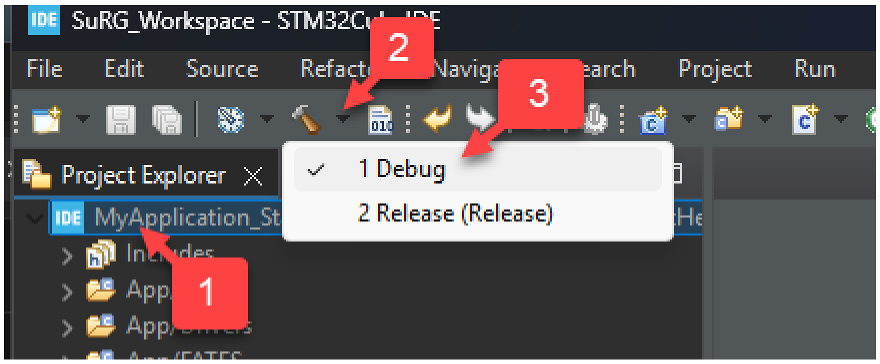
☐ Add project to working sets

Working sets:

NOTE: It is important to ensure "Copy projects into workspace" is unchecked or else you will use up a lot of disk space and might edit files in the copied project rather than the original.

10. Once the project opens –
    a. Click on the project name within the file explorer to make it the active project.
    b. Click the dropdown arrow next to the Build icon (indicated by a hammer).
    c. After clicking the dropdown arrow, select “Debug”.

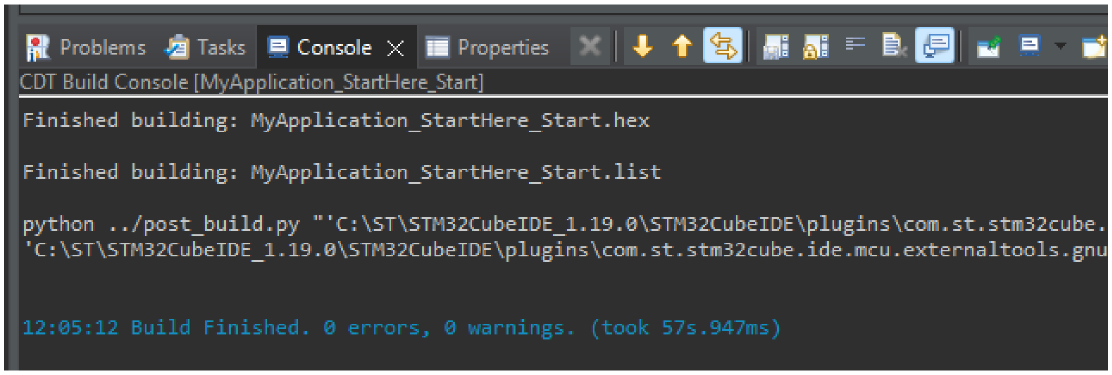
Figure 10: *Building the Project*



NOTE: After doing this once, you can also press CTRL + B on your keyboard to build for the most recently used Build configuration.

11. The StartHere_Start project should build with zero errors.

Figure 11: *STMCube32IDE Console, showing no errors*



```
Finished building: MyApplication_StartHere_Start.hex

Finished building: MyApplication_StartHere_Start.list

python ../post_build.py "'C:\ST\STM32CubeIDE_1.19.0\STM32CubeIDE\plugins\com.st.stm32cube.
'C:\ST\STM32CubeIDE_1.19.0\STM32CubeIDE\plugins\com.st.stm32cube.ide.mcu.externaltools.gnu


12:05:12 Build Finished. 0 errors, 0 warnings. (took 57s.947ms)
```

12. At this point, power on your SuRG42-CP unit by pressing PB2 and make sure the J-Link cable is connected to the JTAG header J1.

13. Now you need to select a Debug Configuration to download + debug the project. Click the dropdown arrow next to the Bug icon, and select “Jlink_Debug”:

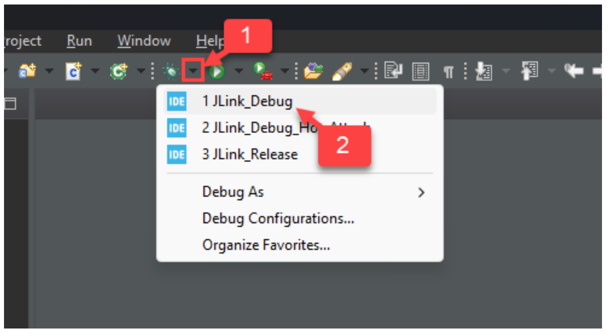
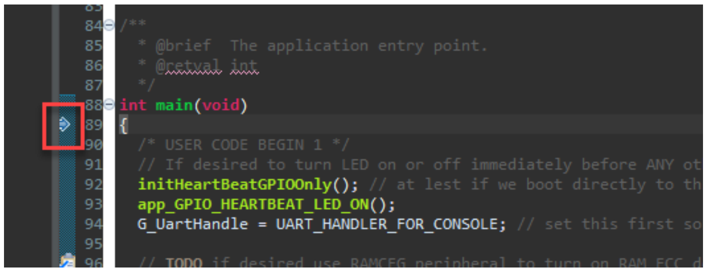


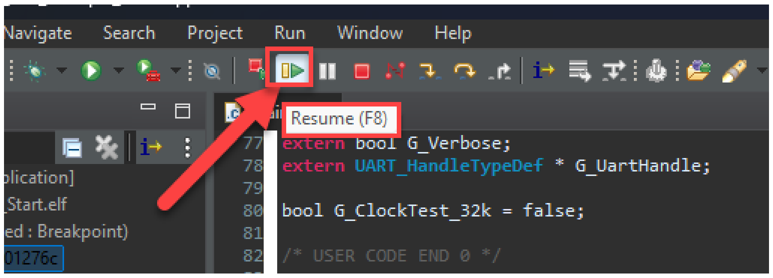
Figure 12: Debug Launch Options for SuRG42-CP

14. After downloading the code to your SuRG42-CP unit, execution will pause at a breakpoint:



```
/**
 * @brief  The application entry point.
 * @retval int
 */
int main(void)
{
  /* USER CODE BEGIN 1 */
  // If desired to turn LED on or off immediately before ANY ot
  initHeartBeatGPIOOnly(); // at lest if we boot directly to th
  app_GPIO_HEARTBEAT_LED_ON();
  G_UartHandle = UART_HANDLER_FOR_CONSOLE; // set this first so
  // TODO if desired use RAMCFG peripheral to turn on RAM ECC d
```

15. Press F8 on your keyboard or click the Continue icon to continue.



```
extern bool G_Verbose;
extern UART_HandleTypeDef * G_UartHandle;

bool G_ClockTest_32k = false;

/* USER CODE END 0 */
```

16. will see the Main Menu screen appear on the SuRG once again.

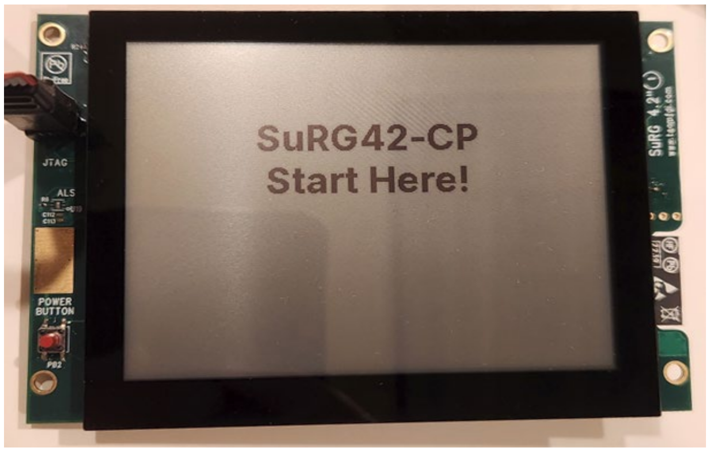
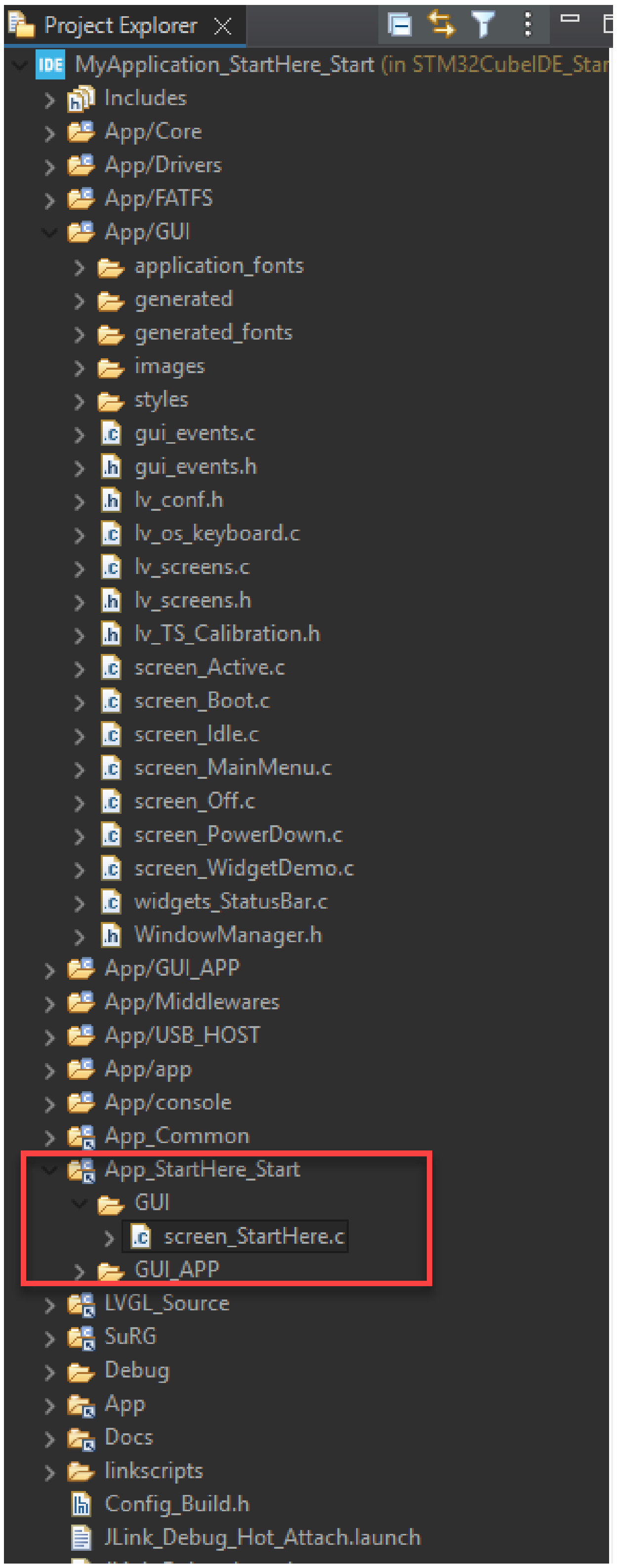


Figure 16: SuRG42-CP StartHere first screen

To see where the "Start Here!" screen is developed, examine the folder structure of the project within STM32Cube:



```
Project Explorer
MyApplication_StartHere_Start (in STM32CubeIDE_Star
  Includes
  App/Core
  App/Drivers
  App/FATFS
  App/GUI
    application_fonts
    generated
    generated_fonts
    images
    styles
    gui_events.c
    gui_events.h
    lv_conf.h
    lv_os_keyboard.c
    lv_screens.c
    lv_screens.h
    lv_TS_Calibration.h
    screen_Active.c
    screen_Boot.c
    screen_Idle.c
    screen_MainMenu.c
    screen_Off.c
    screen_PowerDown.c
    screen_WidgetDemo.c
    widgets_StatusBar.c
    WindowManager.h
  App/GUI_APP
  App/Middlewares
  App/USB_HOST
  App/app
  App/console
  App_Common
  App_StartHere_Start
    GUI
      screen_StartHere.c
    GUI_APP
  LVGL_Source
  SuRG
  Debug
  App
  Docs
  linkscripts
  Config_Build.h
  JLink_Debug_Hot_Attach.launch
```

Figure 17: *File Hierarchy*

D. Adding a Button to the Start Here Screen

To demonstrate the process for adding widgets to a window, this section will walk you through the steps to create a new button on the main menu, which will create a new window we will build and populate from scratch.

NOTE: Most of the code segments in this guide are formatted to be copied and pasted to assist you in easily applying the code to your project.

1. In the App/GUI folder, open “lv_screens.h”.
2. Do a search for the struct “**lv_ui”** defined around *line 68*. This is a struct that stores pointers to windows and widgets.

NOTE: There are a few different declarations of “lv_ui” in this file; the one around ***line 68*** is the one you want to change; the rest are declarations used by the main demo/quickstart applications, and are deactivated for this StartHere project.

3. Within this struct, scroll up and search for the section declaring Main Menu widgets.
   a. It will look like this:

```
// Start Here screen
lv_screen_struct_t screen_StartHere
bool screen_ StartHere _del
lv_obj_t  screen_StartHere_label

// Add new windows and widgets here!
```

4. Add the following code beneath the comment “add new windows and widgets here!”:

```
// Add new windows and widgets here!
lv_obj_t *screen_StartHere_label_button;
lv_obj_t *screen_StartHere_button;
```

5. Save lv_screens.h.
6. Open the file “screen_StartHere.c” in App_StartHere_Start/GUI.
7. This file contains the information for creating the screen and its widgets, as well as their layouts.
8. At the top of the file you will see these 4 #defines:

```
#define LABEL_WIDTH             DISPLAY_1_X_RESOLUTION
#define LABEL_HEIGHT            64
#define LABEL_X                 (DISPLAY_1_X_RESOLUTION / 2) - (LABEL_WIDTH / 2)
#define LABEL_Y                 (DISPLAY_1_Y_RESOLUTION / 3) - (LABEL_HEIGHT / 2)
```

9. **Add** the following **new** #defines under the comment “Add new layout/organizational definitions here!”:

```
#define BUTTON_WIDTH            DISPLAY_1_X_RESOLUTION / 4
#define BUTTON_HEIGHT           DISPLAY_1_Y_RESOLUTION / 6
#define BUTTON_X                (DISPLAY_1_X_RESOLUTION / 2) - (BUTTON_WIDTH / 2)
#define BUTTON_Y                LABEL_Y + LABEL_HEIGHT + (BUTTON_HEIGHT / 2)
```

10. The function “setup_screen_StartHere” is where the screen and its widgets are properly set up.
11. Within “setup_screen_StartHere”, add the following code beneath the comment “add new widgets here!”

```
// Add new widgets here!
// Create the new button
obj = ui->screen_StartHere_button = lv_btn_create(screenPtr);
lv_obj_set_scrollbar_mode(obj, LV_SCROLLBAR_MODE_OFF);
lv_obj_set_size(obj, BUTTON_WIDTH, BUTTON_HEIGHT);
lv_obj_add_flag(obj, LV_OBJ_FLAG_CLICKABLE);
lv_obj_add_style(obj, &G_style_btn_white_on_red_xxsmall, 0);
lv_obj_set_pos(obj, BUTTON_X, BUTTON_Y);

// Create the new label for the button text
obj = ui->screen_StartHere_label_button = lv_label_create(ui->screen_StartHere_button);
lv_label_set_text(obj, "Temp");
lv_obj_add_style(obj, &G_style_btn_white_on_red_xxsmall, 0);
lv_obj_center(obj);
```

12. Save “screen_StartHere.c”.
13. Now build the StartHere project again. It should build with 0 errors.
14. Since you have already run a Debug Configuration previously (Jlink_Debug), you can press F11 to run the last debug configuration used.
15. Press F11 now to download and debug the project.
16. Once downloading completes, continue past the breakpoint in main, and you will see a new button on the StartHere screen:

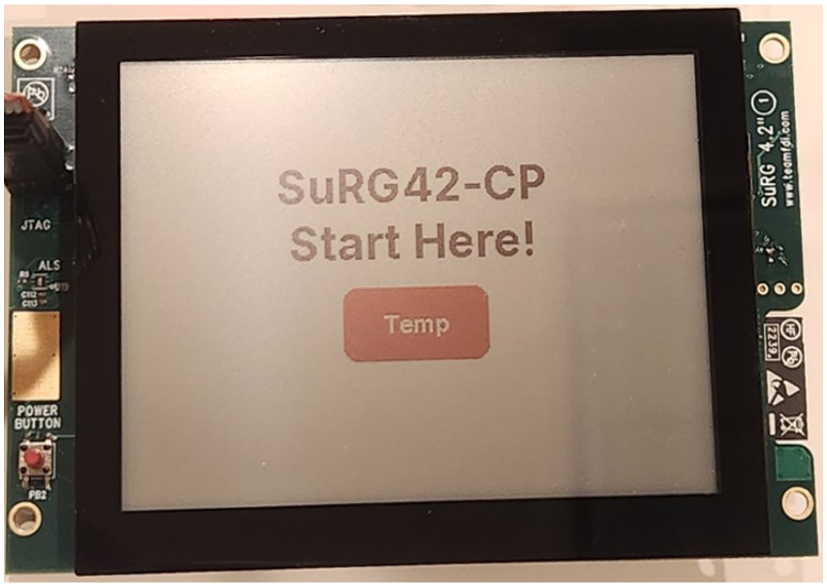


Figure 18: *Button Added*

E. Adding Callback Functionality to the New Button

1. Despite the code we added giving the button the "clickable" flag, the button currently does not "know" what to do when clicked. This section will focus on solving this by adding a button click callback.

NOTE: A callback is a function which is called when an associated event occurs. For buttons, this event can be either a press or release event. When the button is pressed (or released), the callback function is run to perform the desired task.

2. In "screen_StartHere.c", at the end of the function "setup_screen_StartHere", add the following line:

```
events_init_screen_StartHere(ui);  // Setup events for this screen
```

3. This function is already "defined" in "App/Gui/gui_events.h" and "implemented" in "App_StartHere_Start/GUI_APP/callbacks_StartHere.c" respectively, but it is currently empty.
4. Open "App_StartHere_Start/GUI_APP/callbacks_StartHere.c".
5. Add the following to the function "events_init_screen_StartHere":

```
// Get the reference to the widget
lv_obj_t* obj = ui->screen_StartHere_button;
// Add a callback event to the widget
lv_obj_add_event_cb(obj, screen_StartHere_button_cb, LV_EVENT_CLICKED, ui);
```

6. Now we must add the callback function we passed in.
7. Open "App/GUI/gui_events.h".
8. Under the comment "Add new events and callback definitions here!", add the following:

```
// Add new events and callback definitions here!
void screen_StartHere_button_cb(lv_event_t * e);
```

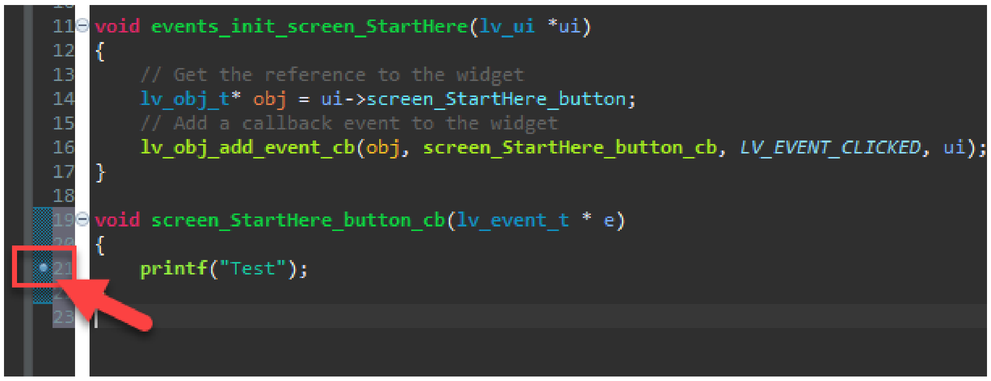
9. Save gui_events.h.

NOTE: Anytime we want a screen to have callback events, we will define them in gui_events.h and implement them in the appropriate "callbacks_<screenName>.c" files respectively to keep the project organized. Bear this in mind for future sections.

10. Open “App_StartHere_Start/GUI_APP/callbacks_StartHere.c” again.
11. Add the function implementation for screen_StartHere_button_cb:

```
void screen_StartHere_button_cb(lv_event_t * e)
{
    printf("Test");
}
```

12. You can test if your callback is working by double clicking on the line numbers on the far left of the code/text editor to add a breakpoint on the “printf(“Test”)” line.



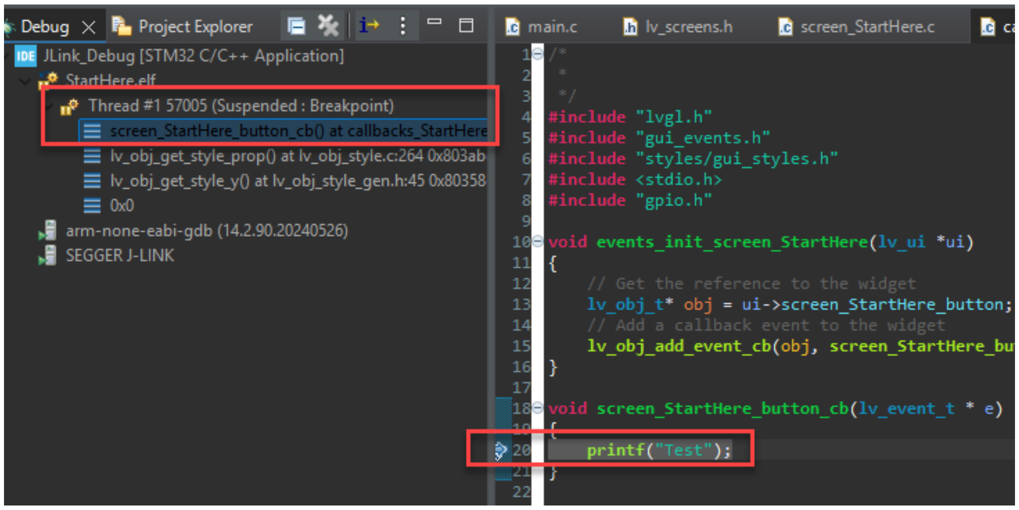
```
void events_init_screen_StartHere(lv_ui *ui)
{
    // Get the reference to the widget
    lv_obj_t* obj = ui->screen_StartHere_button;
    // Add a callback event to the widget
    lv_obj_add_event_cb(obj, screen_StartHere_button_cb, LV_EVENT_CLICKED, ui);
}

void screen_StartHere_button_cb(lv_event_t * e)
{
    printf("Test");

```

Figure 19: *Testing button callback with breakpoints*

13. Build and download + debug the project again.
14. Try clicking the “Temp” button on your SuRG42-CP screen now. Your code execution should pause at the breakpoint. This will show that your callback is working as expected.



```
JLink_Debug [STM32 C/C++ Application]
  StartHere.elf
    Thread #1 57005 (Suspended : Breakpoint)
      screen_StartHere_button_cb() at callbacks_StartHere
      lv_obj_get_style_prop() at lv_obj_style.c:264 0x803ab
      lv_obj_get_style_y() at lv_obj_style_gen.h:45 0x80358
      0x0
  arm-none-eabi-gdb (14.2.90.20240526)
  SEGGER J-LINK
```

```
/*
 *
 */
#include "lvgl.h"
#include "gui_events.h"
#include "styles/gui_styles.h"
#include <stdio.h>
#include "gpio.h"

void events_init_screen_StartHere(lv_ui *ui)
{
    // Get the reference to the widget
    lv_obj_t* obj = ui->screen_StartHere_button;
    // Add a callback event to the widget
    lv_obj_add_event_cb(obj, screen_StartHere_bu
}

void screen_StartHere_button_cb(lv_event_t * e)
{
    printf("Test");
}
```

15. Double click your breakpoint again to remove it.

F. Adding and Switching to a New Window

Now that we have a functioning button, we can use it to change from the StartHere screen to another, new screen. On this screen, we will go a step further and have it interact with the SuRG42-CP Temp Sensor to report the current temperature of the unit.

1. To do this, we will first need to open "App/GUI/lv_screens.h" again and add some new pointer variables to the struct "lv_ui".
2. Within lv_ui, add the following members to the struct:

```
  // Temp screen
  lv_screen_struct_t screen_Temp;
  bool screen_Temp_del;                   // Keeps track of the screen's state
  lv_obj_t *screen_Temp_label_title;      // Label for the screen's title
  lv_obj_t *screen_Temp_label_button;     // Label for the Back button
  lv_obj_t *screen_Temp_label_temp;       // Label for displaying current temp
  lv_obj_t *screen_Temp_button;           // 'Back Button'
  lv_timer_t *screen_Temp_timer_update;// Timer for updating the Temp Label
```

3. Scroll down to the comment "Add new screen startup definitions here!" and add the following line:

```
// Add new screen startup definitions here!
void setup_screen_Temp(lv_ui *ui);
```

4. Save "lv_screens.h".
5. Now, in "App/GUI/lv_screens.c", scroll down to the function "init_scr_del_flag" around *line 189*, and add the following beneath the comment "Add new screen deleted status variables here!":

```
// Add new screen deleted status variables here!
ui->screen_Temp.deleted = true;
```

6. Now in the function "clear_window_pts" right below the previous function, add the following beneath the comment "Add clear-window-pointer calls for new screens here!":

```
// Add clear-window-pointer calls for new screens here!
CLEAR_WINDOW_PTR_IF_DELETED(lv_gui_screens.screen_Temp);
```

7. Save "lv_screens.c".
8. Back in "APP_StartHere_Start/GUI_APP/callbacks_StartHere.c", replace the "printf" line in "screen_startHere_button_cb" with the following:

```
switch_to_screen(&lv_gui_screens.screen_Temp, setup_screen_Temp, false,
LV_SCR_LOAD_ANIM_NONE, 0);
```

9. Save "callbacks_StartHere.c".

10. To switch to our new screen, a new screen file like the screen_StartHere.c file needs to be created.
    Since each window requires the same general structure, you can just copy and paste screen_StartHere.c and rename it:
    a. Right-click on "screen_StartHere.c" and click "copy".

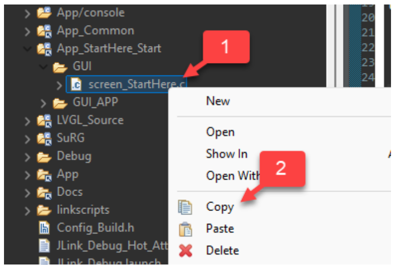
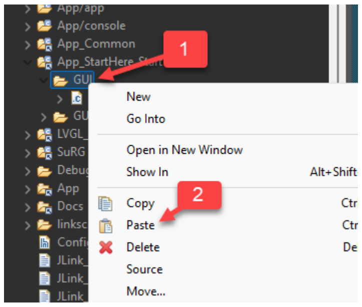
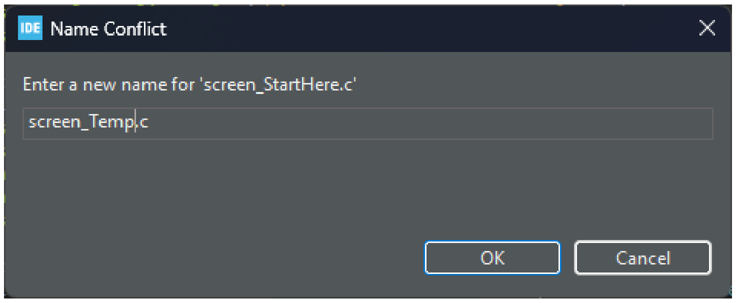


Figure 20: Adding a new Screen

    b. Right-click on "App_StartHere_Start/GUI" and click paste.



    c. Name the pasted file "screen_Temp.c".



11. Open "screen_Temp.c"
12. The first thing to change will be the #defines at the top for setting up the size and positions of the widgets.
13. Replace **all** the defines with these new ones:

```
#define LABEL_TITLE_WIDTH      DISPLAY_1_X_RESOLUTION
#define LABEL_TITLE_HEIGHT     64
#define LABEL_TITLE_X          (DISPLAY_1_X_RESOLUTION / 2) - (LABEL_TITLE_WIDTH / 2)
#define LABEL_TITLE_Y          (DISPLAY_1_Y_RESOLUTION / 3) - (LABEL_TITLE_HEIGHT / 2)

#define LABEL_TEMP_WIDTH       DISPLAY_1_X_RESOLUTION
#define LABEL_TEMP_HEIGHT      64
#define LABEL_TEMP_X           (DISPLAY_1_X_RESOLUTION / 2) - (LABEL_TEMP_WIDTH / 2)
#define LABEL_TEMP_Y           (DISPLAY_1_Y_RESOLUTION / 3) - (LABEL_TEMP_HEIGHT / 2)

#define BUTTON_WIDTH           DISPLAY_1_X_RESOLUTION / 4
#define BUTTON_HEIGHT          DISPLAY_1_Y_RESOLUTION / 6
#define BUTTON_X               0
#define BUTTON_Y               DISPLAY_1_Y_RESOLUTION - BUTTON_HEIGHT
```

14. Rename the function "setup_screen_StartHere" to "setup_screen_Temp".

15. Replace everything in “setup_screen_Temp” with the following code block:

```
lv_obj_t* screenPtr;
lv_obj_t* obj;

screenPtr = ui->screen_Temp.screenPtr = lv_obj_create(NULL);
lv_obj_set_scrollbar_mode(screenPtr, LV_SCROLLBAR_MODE_OFF);
lv_obj_clear_flag(screenPtr, LV_OBJ_FLAG_SCROLLABLE);
lv_obj_add_style(screenPtr, &G_style_screen_white, LV_PART_MAIN|LV_STATE_DEFAULT);

obj = ui->screen_Temp_label_title = lv_label_create(screenPtr);
lv_obj_set_scrollbar_mode(obj, LV_SCROLLBAR_MODE_OFF);
lv_label_set_long_mode(obj, LV_LABEL_LONG_WRAP);
lv_obj_set_size(obj, LABEL_TITLE_WIDTH, LABEL_TITLE_HEIGHT);
lv_label_set_text(obj, "Temp Sensor");
lv_obj_add_style(obj, &G_style_text_red_on_white_xlarge, LV_PART_MAIN|LV_STATE_DEFAULT);
lv_obj_set_pos(obj, LABEL_TITLE_X, LABEL_TITLE_Y);

obj = ui->screen_Temp_button = lv_btn_create(screenPtr);
lv_obj_set_size(obj, BUTTON_WIDTH, BUTTON_HEIGHT);
lv_obj_add_flag(obj, LV_OBJ_FLAG_CLICKABLE);
lv_obj_set_pos(obj, BUTTON_X, BUTTON_Y);

obj = ui->screen_Temp_label_button = lv_label_create(ui->screen_Temp_button);
lv_obj_add_style(obj, &G_style_btn_white_on_black_xsmall, 0);
lv_label_set_text(obj, "Back");
lv_obj_center(obj);

obj = ui->screen_Temp_label_temp = lv_label_create(screenPtr);
lv_obj_set_scrollbar_mode(obj, LV_SCROLLBAR_MODE_OFF);
lv_label_set_long_mode(obj, LV_LABEL_LONG_WRAP);
lv_label_set_text(obj, "0.0 C");
lv_obj_set_size(obj, LABEL_TEMP_WIDTH, LABEL_TEMP_HEIGHT);
lv_obj_add_style(obj, &G_style_btn_black_on_white_medium, 0);
lv_obj_center(obj);
lv_obj_set_style_border_width(obj, 0, 0);
```

16. Build, download, and debug the project.
17. When you click on the Temp button on the StartHere screen, we should be taken to our new “Temp Sensor” screen:

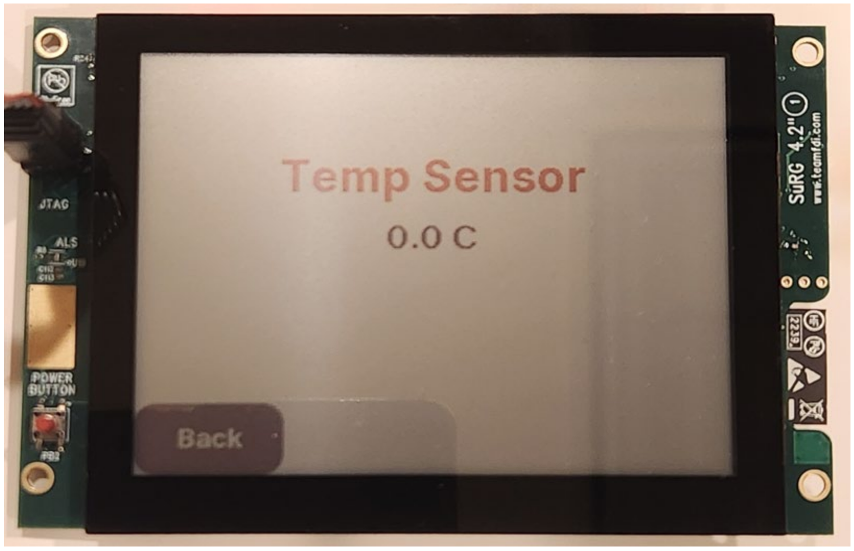


Figure 21: *Temp Sensor Screen*

G. Reading the SuRG Temp Sensor

In this section we will show how we can read from the sensors on SuRG and display their feedback within our app.

In SuRG, the Temp Sensor is already initialized within “App/app/thread_Sensors.c”.

The Sensors thread, or “task”, periodically polls the temp sensor, the touch screen, and (if installed) the ALS sensor. The table below demonstrates the execution flow for the sensors task:

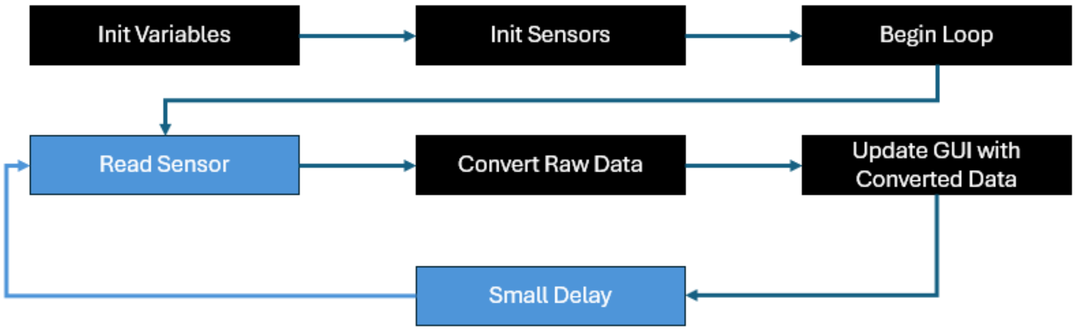


Figure 22: *General Sensor Task Flow*

1. Now much like with the StartHere screen, we need to add callbacks for our button on the Temp Screen to work, and we also need a callback to periodically update the temperature reading text.
2. In “screen_Temp.c”, add this line to the end of the function “setup_screen_Temp”**:**

```
events_init_screen_Temp(ui);
```

3. Navigate into “App/GUI/gui_events.h” and add these two function definitions:

```
void screen_Temp_update_event(lv_timer_t * timer);
void events_init_screen_Temp(lv_ui * ui);
```

4. Save “gui_events.h”.
5. In App_StartHere_Start/GUI_APP, copy the file “callbacks_StartHere.c” and paste the copy in the same folder.
6. Name the copy “callbacks_Temp.c”.
7. Open “callbacks_Temp.c”, and add these lines **after** the final #include:

```
#include <drivers/LM75BD/Temperature_LM75A.h>
extern T_Temp_NXP_LM75A_Workspace G_LM75_Workspace;
```

G_LM75_Workspace is a global variable used for accessing the temp sensor (which is an LM75). This variable is already initialized in “App/app/thread_Sensors.c”, so we just need access to it within this file.

8. Still inside “callbacks_Temp.c” replace everything else with the following:

```
void screen_Temp_update_event(lv_timer_t * timer)
{
        lv_ui *ui = (lv_ui*)timer->user_data;
        HAL_StatusTypeDef error;
        int32_t temp, whole;
        uint32_t frac;
        char buf[16];

        // Reset the timer and start it over once it expires
        lv_timer_reset(ui->screen_Temp_timer_update);
        lv_timer_resume(ui->screen_Temp_timer_update);

        // Try to read the Temp Sensor
        error = Temp_NXP_LM75A_Read(&G_LM75_Workspace, &temp);
        if (error == HAL_OK)
        {
                // We should only attempt updating the text if the active screen is the
Temp Sensor screen
                if(lv_scr_act() == lv_gui_screens.screen_Temp.screenPtr)
                {
                        // Convert the data from the temp sensor to human-readable info
                        if (temp < 0) frac = -temp;
                        else frac = temp;
                        whole = temp >> 16;
                        frac = ((((uint32_t)(frac)) & 0xFFFF) >> 13) * 125;
                        snprintf(buf, sizeof(buf), "%ld.%03lu C", whole, frac);
                        lv_label_set_text(ui->screen_Temp_label_temp, buf);
                }
        }
}

void screen_Temp_button_cb(lv_event_t * e)
{
    switch_to_screen(&lv_gui_screens.screen_StartHere, setup_screen_StartHere, false,
LV_SCR_LOAD_ANIM_NONE, 0);
}

void events_init_screen_Temp(lv_ui *ui)
{
    // Add a callback to our button on the Temp Screen
    lv_obj_t* obj = ui->screen_Temp_button;
    lv_obj_add_event_cb(obj, screen_Temp_button_cb, LV_EVENT_CLICKED, ui);

    // Create the timer used to update the Temp Sensor text on the GUI
    ui->screen_Temp_timer_update = lv_timer_create(screen_Temp_update_event, 1000,
ui);
}
```

9. Here is a brief breakdown of what each function does:
   a. **void screen_Temp_update_event**
      i. This function is a timer callback event. Every time the timer finishes, it attempts to read the Temp Sensor. If the active screen is "screen_Temp" and there were no errors in reading the Sensor, the GUI is updated with the data converted into human-readable format.
   b. **void screen_Temp_button_cb**
      i. This is our button callback function, that will switch from the Temp screen back to the StartHere screen.
   c. **void events_init_screen_Temp**
      i. This function, like for the StartHere screen's variant, initializes the events for this screen's widgets. We also initialize our Temp Sensor GUI update timer and its respective callback function and the interval of the timer in milliseconds here.
10. Open "App/GUI/gui_events.h" once more and add the Temp button callback:

```
void screen_Temp_button_cb(lv_event_t * e);
```

11. Build, download + debug the application.
12. When you click the Temp button, you should now see the Temp Sensor value label periodically update with the SuRG42-CP's temperature.

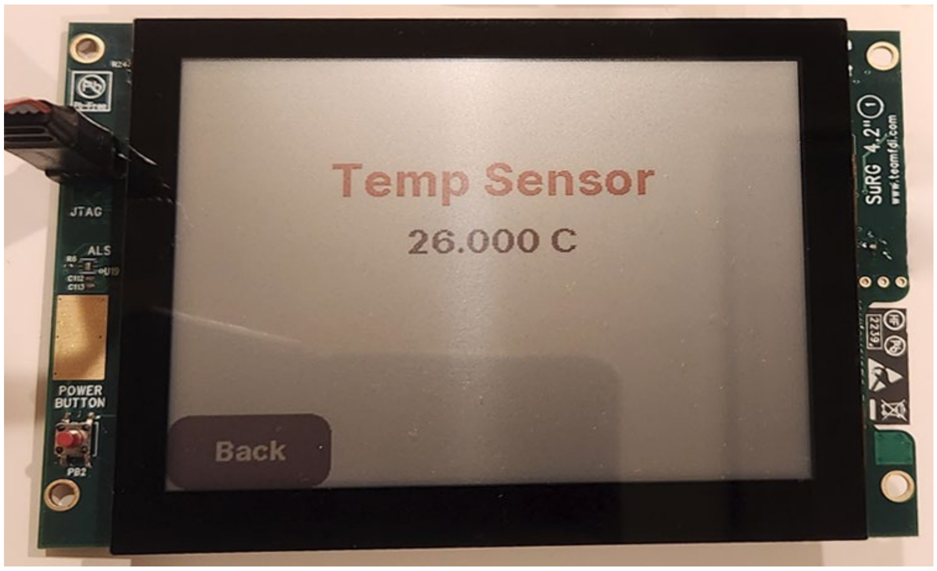


Figure 23: *Final Temp Screen*

13. Click the back button, and you will be taken back to the Start-Here screen.

Congratulations! This concludes the walkthrough for building a simple GUI! Professional looking designs with complex functionality can be designed using advanced features provided in LVGL and SuRG. Documentation and support for learning these features is available through Future Designs, Inc. for LVGL and SuRG.

5. *Next Steps*

Additional examples can be found on the https://www.teamfdi.com/ website.

- SuRG Image Conversion Guide: ....https://tinyurl.com/SuRG-Image-Conversion-Guide
- SuRG Quick Start Guide: https://tinyurl.com/SuRG42-CP-Quick-Start-Guide
- SuRG User's Manual: ....https://tinyurl.com/SuRG-UM

6. *Website and Support*

Documentation:

- SuRG Library Online Documentation .... https://www.teamfdi.com/products/surg/
- LVGL Documentation https://docs.lvgl.io/8.0/

Support & Downloads:

- FDI Support Home Page .... https://www.teamfdi.com/support
- FDI Forums ....https://www.teamfdi.com/forums
- SuRG42-CP Product Page https://www.teamfdi.com/product/surg42-cp/
- SuRG Source w/ Project Maker .... https://sourceforge.net/projects/fdi-surg/files
- Start Here Guide ....https://www.teamfdi.com/StartHere

Contact Information:

Future Designs, Inc.

996 A Cleaner Way SW
Huntsville, AL 35805
Phone: (256) 883-1240
Fax: (256) 883-1241
Email: Sales@TeamFDI.com

https://www.teamfdi.com/