# Current installation

Paths may not contain spaces

|  |  |
| --- | --- |
| %userprofile%\esp\esp-idf | contains the IoT development framework |
| %userprofile%\esp\ | contains the projects |

# Quick shortcuts

%userprofile%\esp\esp-idf\export.bat

idf.py -p COM7 build flash monitor

# Installation of esp-idf in windows

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/windows-setup.html>

Install latest version of ESP-IDF (currently 4.0) to run lvgl

# Creating a new esp-idf project

**Important! Spaces are not allowed in the project names and directories**

Copy example to project:

**cd** %userprofile%\esp

xcopy /e /i %IDF\_PATH%\examples\get-started\hello\_world hello\_world

# Before each time working on a project

Set the paths when working on a project:

%userprofile%\esp\esp-idf\export.bat

Choose settings for the compiler:

**cd** %userprofile%\esp\hello\_world

idf.py menuconfig

# Building and uploading a project

Build the project:

idf.py build

Flash the project and monitor the serial port:

idf.py -p COM7 flash monitor

# Useful information for project configuration

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/build-system.html>

Concepts

* A **project** is a directory that contains all the files and configuration to build a single **app** (executable), as well as additional supporting elements such as a partition table, data/filesystem partitions, and a bootloader.
* **Project configuration** is held in a single file called sdkconfig in the root directory of the project. This configuration file is modified via idf.py menuconfig to customise the configuration of the project. A single project contains exactly one project configuration.
* An **app** is an executable which is built by ESP-IDF. A single project will usually build two apps - a **project app** (the main executable, ie your custom firmware) and a **bootloader app** (the initial bootloader program which launches the project app).
* **components** are modular pieces of standalone code which are compiled into static libraries (.a files) and linked into an app. Some are provided by ESP-IDF itself, others may be sourced from other places.
* **Target** is the hardware for which an application is built. At the moment, ESP-IDF supports esp32 and esp32s2 targets.

# 

# Adding a custom ESP-IDF component

As you build projects, you may want to build your own custom components. To create a

component, create a directory in the root of your application called "components" and

within there, create a directory named after your new component. In that directory, we

will construct our component. It should contain:

* component.mk
* <your source files>.c
* Kconfig (optional)

# Component cmakelists files

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/build-system.html#component-cmakelists-files>

# Graphics libraries

| **Library** | **IDE** | **Color** | **Touch** | **Full GUI** | **Maintained** | **Remarks** |
| --- | --- | --- | --- | --- | --- | --- |
| [Loboris](https://github.com/loboris/ESP32_TFT_library) | ESP-IDF | Yes | Yes | No | 28-8-2018 | No longer maintained |
| [u8g2](https://github.com/olikraus/u8g2) | Generic | No | No | No | Yes | Simple monochrome library |
| [TFT\_eSPI](https://github.com/Bodmer/TFT_eSPI) | Arduino | Yes | Yes | No | Yes | Maintained by Bodmer  Not easily compatible with ESP-IDF |
| [lvgl](https://github.com/lvgl/lv_port_esp32) | ESP-IDF | Yes | Yes | Yes | Yes | Very professional  Potential risk: memory |

Useful related links:

<https://github.com/loboris/ESP32_TFT_library>

<http://www.lcdwiki.com/2.8inch_SPI_Module_ILI9341_SKU:MSP2807>

<https://github.com/PaulStoffregen/XPT2046_Touchscreen>

On which SPI port to use by the Bodmer driver:

<https://github.com/Bodmer/TFT_eSPI/issues/294>

# Little versatile graphical library (lvgl)

## Installing

The port of lvgl to ESP32 can be found at <https://github.com/lvgl/lv_port_esp32>

Use idf.py menuconfig to configure the touch screen and the pinouts.

When using the ILI9341 320x240 in landscape mode, the touchscreen options should be set as:

* SwapXY should be off
* Invert X should be on
* Invert Y should be on

Looping through the tabs in demo mode is configured in:

…\lv\_port\_esp32\components\lv\_examples\lv\_ex\_conf.h

Change the define in line 45 to:

#define LV\_DEMO\_SLIDE\_SHOW 0 /\*Automatically switch between tabs\*/

The theme can be changed in:

C:\Users\hjvanderpol\esp\lv\_port\_esp32\components\lvgl\lv\_conf.h

## Using the lvgl demos

When using the demo’s, LV\_USE\_TUTORIALS in lv\_ex\_conf.h whould be set to 1:

29] #define LV\_USE\_TUTORIALS 1

## Switch off logging of touch screen events

23] #include "esp\_log.h"

Additionally in app\_main:

51] esp\_log\_level\_set("XPT2046", ESP\_LOG\_ERROR);

To do: this supresses the messages in runtime, perhaps also possible at compile time.

## Prepare the drawing of a map

An overview of LVGL widgets can be found at:

<https://docs.lvgl.io/en/html/object-types/index.html>

“Canvases and images are almost identical under-the-hood; the canvas is just more convenient for drawing on. I’d just use a canvas.”

First impression is that lv\_canvas (<https://docs.lvgl.io/en/html/object-types/canvas.html>) can be used to draw the map.

\*Create a canvas and initialize it’s palette\*/

lv\_obj\_t \* canvas = lv\_canvas\_create(lv\_scr\_act(), NULL);

lv\_canvas\_set\_buffer(canvas, cbuf, CANVAS\_WIDTH, CANVAS\_HEIGHT, LV\_IMG\_CF\_INDEXED\_1BIT);

lv\_canvas\_set\_palette(canvas, 0, LV\_COLOR\_TRANSP);

lv\_canvas\_set\_palette(canvas, 1, LV\_COLOR\_RED);

Linedrawing can be done with the function:

lv\_canvas\_draw\_line(canvas, point\_array, point\_cnt, &style)

## How to deal with multiple screens in one UI

A screen can be created from any object type but, the two most typical types are the Base object and the Image (to create a wallpaper).

To create a screen, use lv\_obj\_t \* scr = lv\_<type>\_create(NULL, copy). copy can be another screen to copy it.

To load a screen, use lv\_scr\_load(scr). To get the active screen, use lv\_scr\_act(). These functions works on the default display. If you want to to specify which display to work on, use lv\_disp\_get\_scr\_act(disp) and lv\_disp\_load\_scr(disp, scr). Screens can be deleted with lv\_obj\_del(scr), but ensure that you do not delete the currently loaded screen.