
LVGL project for ESP32

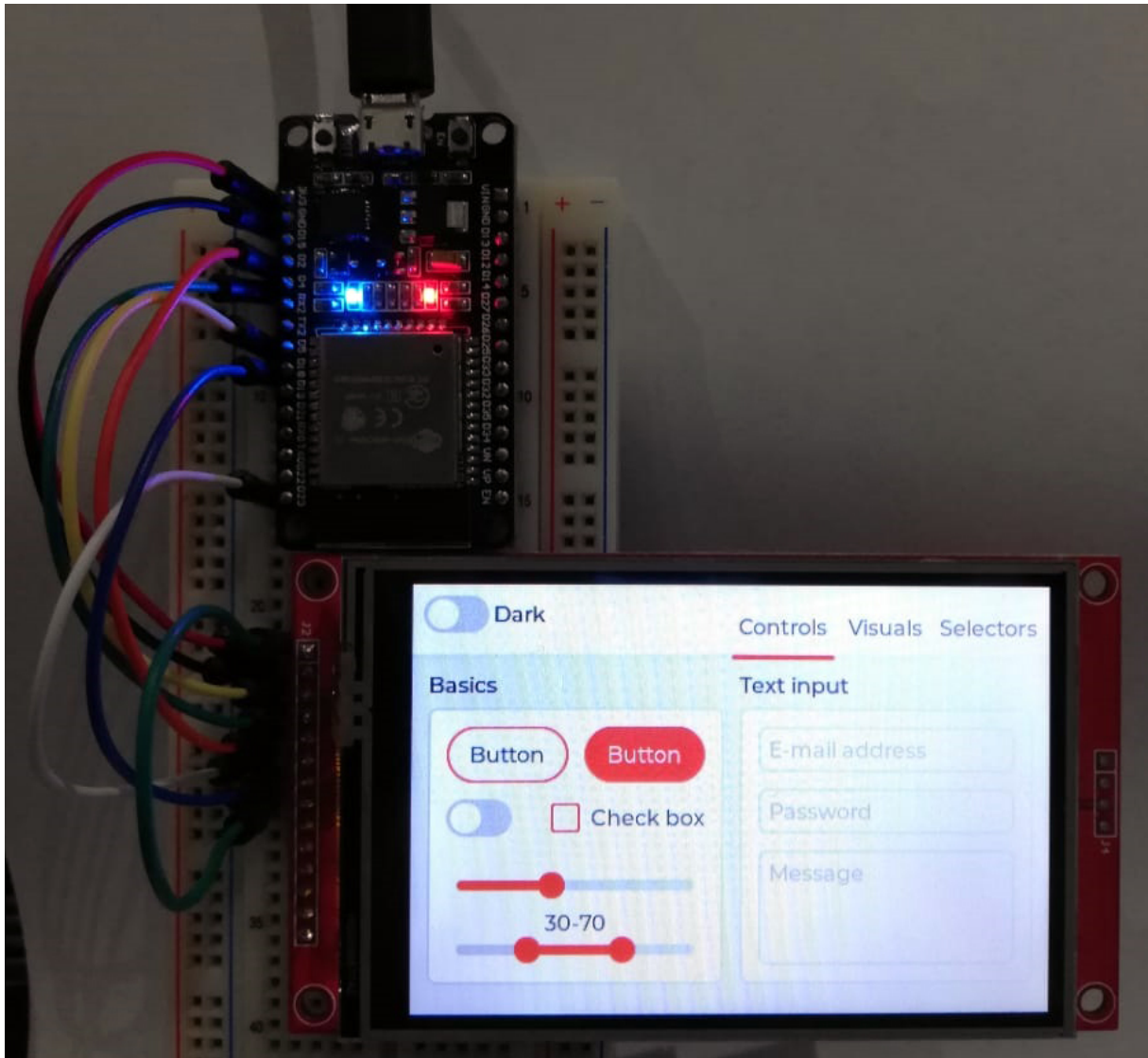
This is an ESP32 demo project showcasing LVGL v7 with support for several display controllers and touch controllers. The demo application is the [lv_demo_widgets](#) project from the lv_examples repository.

- Version of ESP-IDF required 4.2. NOTE: We're trying to make this repo backwards compatible, usage of idf.py is encouraged.
- Version of LVGL used: 7.9.
- Version of lv_examples used: 7.9.

Table of content

- Get started
- Use LVGL in your ESP-IDF project
- Use lvgl_esp32_drivers in your project
- Platformio support
- ESP32-S2 Support

Example demo for TFT displays:



Monochrome support:



Display and touch controllers

The display and touch (indev) controllers are now into it's own repository, you can find it [here](#). To report any issue or add new display or touch (indev) drivers you can do so in the [lvgl_esp32_drivers](#) repo.

Get started

Prerequisites

- ESP-IDF Framework.

Note

This project tries to be compatible with both the ESP-IDF v3.x and v4.0, but using v4.0 is recommended. Instructions assume you are using the v4.x toolchain, otherwise use the make commands, e.g. instead of running `idf.py menuconfig`, run `make menuconfig`.

Build and run the demo.

1. Clone this project by `git clone --recurse-submodules https://github.com/lvgl/lv_port_esp32.git`, this will pull this repo and its submodules.
2. Get into the created `lv_port_esp32` directory.
3. Run `idf.py menuconfig`
4. Configure LVGL in `Components config->LVGL Configuration`. For monochrome displays use the mono theme and we suggest enabling the `unscii` 8 font.
5. Configure your display and/or touch controllers in `Components config->LVGL TFT Display Configuration` and `Components config->LVGL TOUCH Configuration`.
6. Store your project configuration.
7. Build the project with `idf.py build`
8. If the build don't throw any errors, flash the demo with `idf.py -p (YOUR SERIAL PORT) flash` (with `make` this is just `make flash` - in 3.x PORT is configured in `menuconfig`)

Use LVGL in your ESP-IDF project

LVGL now includes a Kconfig file which is used to configure most of the LVGL configuration options via `menuconfig`, so it's not necessary to use a custom `lv_conf.h` file.

It is recommended to add LVGL as a submodule in your IDF project's git repo.

From your project's root directory: 1. Create a directory named `components` (if you don't have one already) with `mkdir -p components`. 2. Clone the lvgl repository inside the `components` directory with `git submodule add https://github.com/lvgl/lvgl.git components/lvgl` 3. Run `idf.py menuconfig`, go to `Component config` then `LVGL configuration` to configure LVGL.

Use lvgl_esp32_drivers in your project

It is recommended to add `lvgl_esp32_drivers` as a submodule in your IDF project's git repo.

From your project's root directory: 1. Create a directory named `components` (if you don't have one already) with `mkdir -p components`. 2. Clone the `lvgl_esp32_drivers` repository inside the `components` directory with `git submodule add https://github.com/lvgl/lvgl_esp32_drivers.git components/lvgl_esp32_drivers` 3. Run `idf.py menuconfig`, go to `Component config` then `LVGL TFT configuration` and `LVGL TFT Display configuration` to configure `lvgl_esp32_drivers`.

Platformio support

Using the `lv_platformio` project add the following lines to `platformio.ini` file:

```
1 [env:esp32]
2 platform = espressif32
3 framework = espidf
4 board = esp-wrover-kit
```

Change the default environment to `default_envs = esp32`.

Modify the `main.c` like this:

```
1 #include "lvgl.h"
2
3 // #include "driver.h"
4
5 #include "demo.h"
6
7 int app_main(void)
8 {
9     lv_init();
10
11     /* Initialize your hardware. */
12
13     /* hw_init(); */
14
15     demo_create();
```

```
16
17     /* Create the UI or start a task for it.
18     * In the end, don't forget to call `lv_task_handler` in a loop. */
19
20     /* hw_loop(); */
21
22     return 0;
23 }
```

For more information see: [platformio with espidf framework compability](#).

ESP32-S2 Support

Support for ESP32-S2 variant is Work In Progress. Smaller displays (e.g. 320x240) work fine, but larger ones need testing.

Background

ESP32-S2 has less on-chip SRAM than its predecessor ESP32 (520kB vs. 320kB). This causes problems with memory allocation with large LVGL display buffers as they don't fit into the on-chip memory and external PSRAM is not accessible by DMA.

Moreover, static allocation to external PSRAM is not yet supported (see [GitHub issue](#)).

At this moment, the buffers are dynamically allocated with DMA capability and memory allocator handles the rest.