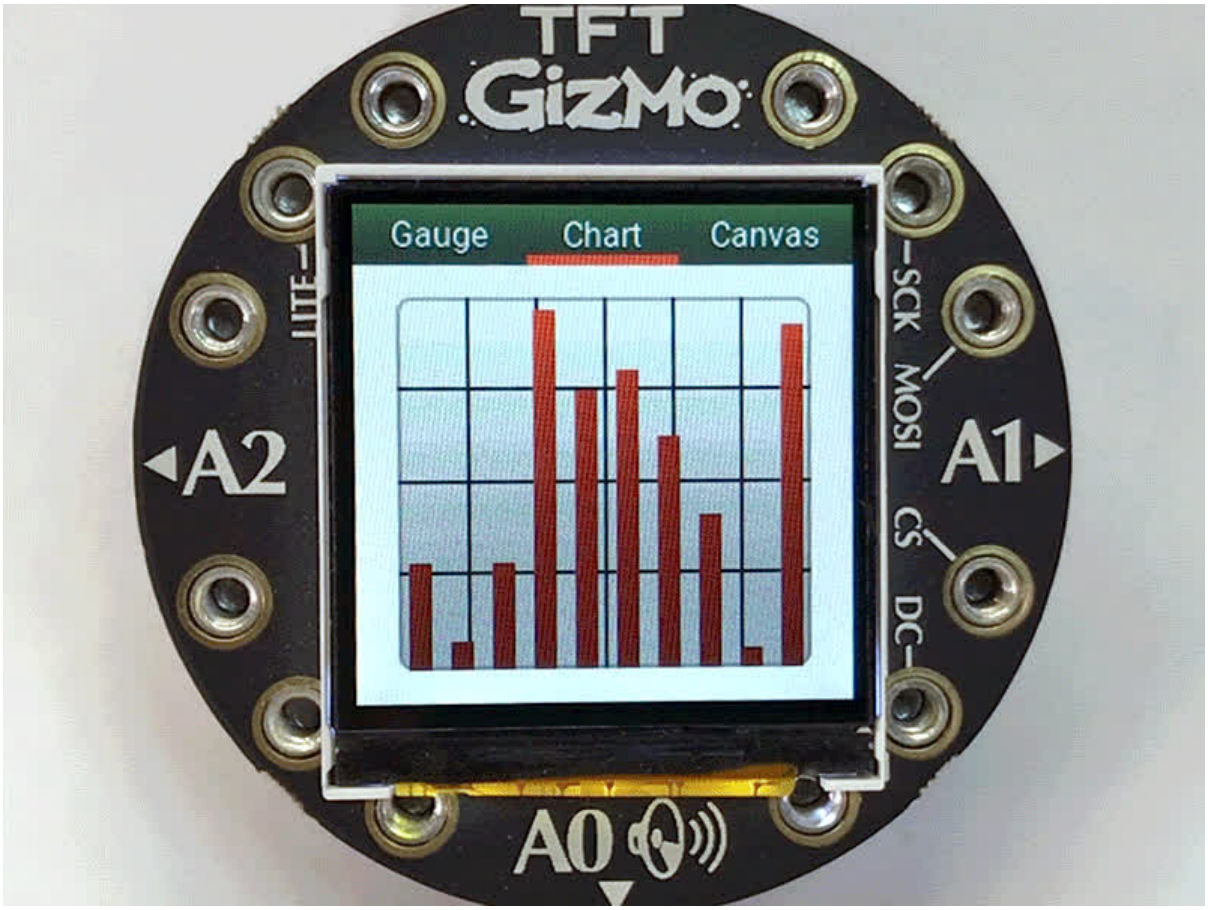
Using LittlevGL with Adafruit Displays

Created by Phillip Burgess

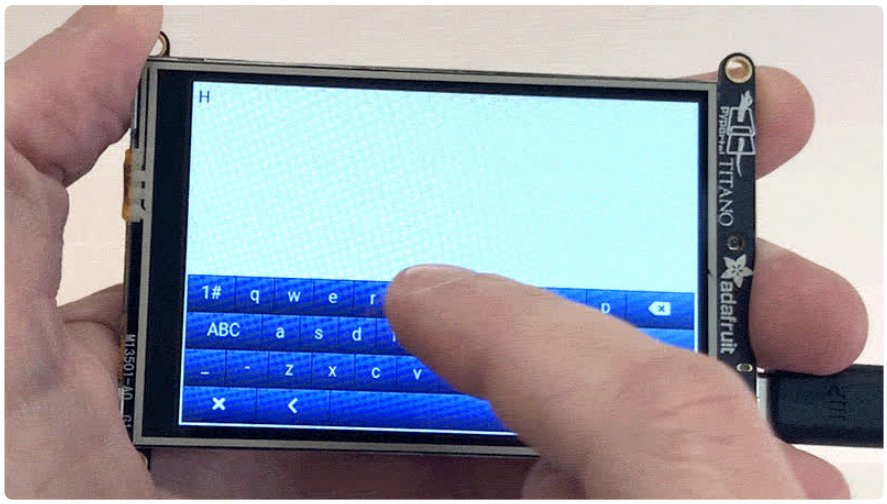


https://learn.adafruit.com/using-littlevgl-with-adafruit-displays

Last updated on 2024-03-08 03:45:33 PM EST

Table of Contents

Overview



LittlevGL is a nice open source graphics library for generating graphical **user interfaces** (buttons, sliders, graphs and so forth) on microcontrollers. It provides most of the UI "smarts" behind the scenes, but on its own doesn't communicate with any specific display or input device. Instead, one normally must write their own intermediary code to tie these together.

We've made an Arduino library — **Adafruit_LvGL_Glue** — that simplifies the task of sticking LittlevGL on many Adafruit displays: PyPortal, TFT FeatherWings, and most other Adafruit devices with (or connected to) a color TFT or OLED screen, using fast DMA transfers when possible. Touchscreens can be used for responsive interfaces. For non-touch displays, LittlevGL can still be useful for its nice graphing and text display capabilities. Mostly, though, we think this will shine on Adafruit PyPortal devices, with their extra fast displays and touch sensitivity.

The following microcontrollers are supported: SAMD51 ("M4"), nRF52840 and ESP32. More 32-bit devices might get added in the future. SAMD21 ("M0") may work for very simple LittlevGL tasks, but generally isn't recommended. And LittlevGL vastly exceeds the capabilities of 8-bit AVR devices (such as Arduino Uno), so don't hold out for support there.

This guide is not an extensive tutorial into LittlevGL. We'll get things set up on Adafruit hardware and run a basic "hello world" type of demo. Going forward from there, you'll find explanations and examples on the LittlevGL documentation site (https://adafru.it/Rva) and elsewhere. Many existing examples can be copied-and-pasted with only minor changes needed!

Installing

For first time LittlevGL users (existing users, see notes later): we'll assume you already have basic Arduino usage set up for your board — SAMD, nRF and so forth — the corresponding introduction guide for each board explains that part. If you can get the "blink" sketch running, you're in good shape.

The required software components for LittlevGL usage can then be installed with the **Arduino Library Manager**...

Sketch→Include Library→Manage Libraries...

Search for and install the following libraries:

- **lvgl** (not the similar lv_arduino library — that's now out of date)
- **Adafruit_LvGL_Glue**
- **Adafruit_GFX**
- **Adafruit_BusIO**
- **Adafruit_Touchscreen**
- **Adafruit_STMPE610**

Even if you're not using a touchscreen, those latter two libraries are still required for the code to compile.

Recent versions of the Arduino IDE install library dependencies automatically, so you'll only need to manually request the first two of these and the rest will come along for the ride. But just in case, there's the whole list.

In addition, you'll need to install one or more libraries **specific to the display** you're using...

- **Adafruit_ILI9341** (320x240 PyPortal, TFT FeatherWing, etc.)
- **Adafruit_HX8357** (480x320 PyPortal Titano, TFT FeatherWing, etc.)
- **Adafruit_ST7735** (includes ST7789 support) (CLUE, TFT Gizmo, HalloWing, PyGamer, etc.)
- Other color Adafruit displays can usually work, see the "Using" page for further details

Prior Adafruit_LvGL_Glue Users

If you've previously used LittlevGL and Adafruit_LvGL_Glue, there are some configuration and code changes you'll need to make.

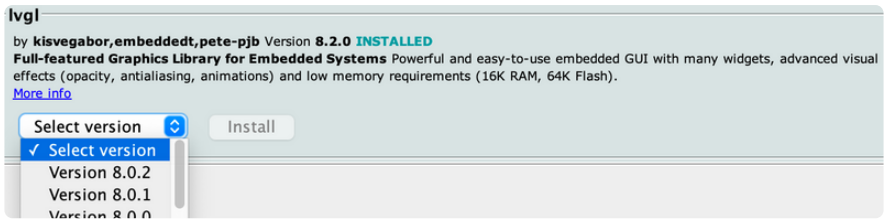
First, **uninstall the lv_arduino library** and **install lvgl** in its place. lv_arduino was an older release of LittlevGL for Arduino...it's quite deprecated now and will not be updated. Use lvgl going forward.

Second, LittlevGL has evolved considerably, and code written with the older library won't compile without changes. A lot of functions and constants have different names, and styles are handled through "setter" functions now. You can look through the Adafruit_LvGL_Glue examples for some insights, but your best reference will be the official LittlevGL documentation (https://adafru.it/Rva).

Additionally, we no longer recommend LittlevGL for SAMD21 ("M0") boards. Simple code might work, but it's generally better to use a SAMD51 ("M4"), nRF52 or ESP32 controller with their spacious RAM.

If the Examples Won't Compile

LittlevGL moves fast, and its developers aren't shy about making changes, even if that means breaking existing user code. If you find that the Adafruit_LvGL_Glue examples won't compile, it might be necessary to rewind the lvgl library version to the last known compatible state (**8.2.0** when last checked). There's a "Select version" option for this in the Arduino Library Manager:



LittlevGL Configuration

By default, our glue library already has LittlevGL set up for our boards and displays. But if you'd like to get in there and tune some settings, the LittlevGL configuration file can be found at:

`(home directory)/Documents/Arduino/libraries/Adafruit_LvGL_Glue/lv_conf.h`

Of possible interest in there, `LV_USE_LOG` and `LV_LOG_LEVEL` can be edited if you want to customize LittlevGL's status logging capabilities (which will print to the Serial Console when using **Adafruit_LvGL_Glue**). `LV_USE_LOG` is set to `1` by default, enabling this capability, but to fully activate it you must pass `true` as an optional extra argument to our library's `begin()` function. If not needed, you can set `LV_USE_LOG` to `0` to free up a little program space. `LV_LOG_LEVEL` is set to `LV_LOG_LEVEL_INFO` by default, displaying the most important events only, but you can step this up to `LV_LOG_LEVEL_TRACE` if you require verbose and detailed information.

Hello Arduino example

To verify that everything's installed and working properly, you can try the minimal "Hello" program...

File→Examples→Adafruit LittlevGL Glue Library→Hello *

There are different versions for **Adafruit CLUE**, **TFT FeatherWing**, **TFT Gizmo** and **PyPortal** (all versions). **Other Adafruit boards and displays can also work**...see the "Using" page. It's fairly straightforward to "mash up" an existing graphics example with the LittlevGL hello example and verify that it's working.

Most of these need no modification. Only the **hello_featherwing** example, where you'll need to edit this line if using a 3.5" (480x320) TFT FeatherWing:

```
#define BIG_FEATHERWING 0 // Set this to 1 for 3.5" (480x320) FeatherWing!
```

Select your board type from the Tools menu, verify the code compiles, and upload. If you get a legible “Hello Arduino!” centered on the screen, everything’s in good shape!

We provide just these basic test examples because of LittlevGL’s fast-paced development...any actual UI “widget” examples tended to break as the library evolved quicker than we could revisit. Instead, have a look at the examples included with the lvgl library, and “mash up” that code with the initialization from our hello examples to produce a working combination. This is detailed a little further on the next page.

Using

As explained in the Overview, this guide is not an in-depth tutorial to writing LittlevGL applications. For that, have a look through the explanations and examples on the LittlevGL documentation site (https://adafru.it/Rva).

Let’s look at the basics of using **Adafruit_LvGL_Glue** though. Once that’s understood, many LittlevGL examples can be copied-and-pasted and work with only minor modification.

Something common to **ALL** of the examples is the inclusion of the Glue library and LittlevGL **header files** (in that order!), and then a few lines down a global `Adafruit_LvGL_Glue` object declaration (which has no arguments):

```
#include &lt;Adafruit_LvGL_Glue.h&gt; // Glue library header INCLUDE THIS FIRST!
#include &lt;lvgl.h&gt;               // LittlevGL header

Adafruit_LvGL_Glue glue;
```

For the PyPortal's touchscreen, the `TouchScreen.h` header is included, and a global `TouchScreen` object is declared. We've abbreviated here...in the examples, you'll see all the values for the `XP`, `YP` and other arguments are `#defined`.

```
#include &lt;TouchScreen.h&gt;

TouchScreen ts(XP, YP, XM, YM, 300);
```

A TFT FeatherWing's touchscreen works a little differently. `Adafruit_STMPE610.h` is included, and a global `Adafruit_STMPE610` object declared, with different arguments (again, abbreviated here, we're not showing all the `#defines`):

```
#include &lt;Adafruit_STMPE610.h&gt;

Adafruit_STMPE610 ts(STMPE_CS);
```

Various displays will have their own device-specific header files. For the PyPortal Titano and 480x320 TFT FeatherWing, that would be `Adafruit_HX8357.h`. A global `Adafruit_HX8357` object is then declared, but the arguments are totally different between PyPortal and FeatherWing. See the example sketches. For FeatherWing, it's like so:

```
#include &lt;Adafruit_HX8357.h&gt;

Adafruit_HX8357 tft(TFT_CS, TFT_DC, TFT_RST);
```

For other PyPortal models and 320x240 TFT FeatherWings, it's `Adafruit_ILI9341.h` and an `Adafruit_ILI9341` object. Again, the arguments vary between PyPortal and FeatherWing, see the examples. For PyPortal it's:

```
#include &lt;Adafruit_ILI9341.h&gt;

Adafruit_ILI9341 tft(tft8bitbus, TFT_D0, TFT_WR, TFT_DC, TFT_CS, TFT_RST, TFT_RD);
```

That's it for the header includes and global objects.

Then, in the `begin()` function, the display is initialized...calling its `begin()` (and optionally `setRotation()`) functions, and (on some devices) switching on the backlight:

```
// Initialize display BEFORE glue setup
  tft.begin();
  tft.setRotation(TFT_ROTATION);
  pinMode(TFT_BACKLIGHT, OUTPUT);    // On devices with controllable backlight
  digitalWrite(TFT_BACKLIGHT, HIGH); // otherwise omit these lines
```

With the TFT Gizmo for Circuit Playground, the backlight control is a little different:

```
analogWrite(TFT_BACKLIGHT, 255); // USE analogWrite() FOR GIZMO BACKLIGHT!
```

If using a **TFT FeatherWing** (or other screen with STMPE610 touchscreen controller), that library also needs a `begin()` call. On PyPortal, this isn't needed.

```
if(!ts.begin()) {
    Serial.println("Couldn't start touchscreen controller");
    for(;;);
  }
```

Finally, with the display and touch initialized, we call our Glue library's `begin()` function, passing in the address of the TFT and (optionally) touchscreen objects (the `&` before each means "address of" in C). If using a non-touch device (as in the CLUE and Gizmo examples), the touchscreen object can be omitted.

```
// Initialize glue, passing in address of display &amp; touchscreen
  LvGLStatus status = glue.begin(&amp;tft, &amp;ts);
  if(status != LVGL_OK) {
    Serial.printf("Glue error %d\r\n", (int)status);
    for(;;);
  }
```

You can also optionally add a "`true`" last argument to `begin()`, which enables logging to the Serial Console (also requires some configuration of LittlevGL, as explained on the previous page).

All of our examples then have this call at the end of the `setup()` function:

```
lvgl_setup(); // Call UI-building function above
```

In the examples, all of the LittlevGL interface-building happens inside a `lvgl_setup()` function, not in `setup()`. This makes it easier to just copy-and-paste the boilerplate Arduino code that sets up the display and touchscreen...you don't need to snip pieces out of `setup()` every time. But you can structure your UI code however you like.

The `loop()` function often then has very little to do. Just need to periodically call one of the LittlevGL internal functions every few milliseconds. Any special handling of events (such as button presses) is usually handled with LittlevGL callback functions.

```
void loop(void) {
  lv_task_handler(); // Call LittleVGL task handler periodically
  delay(5);
}
```

For devices without touchscreens (such as CLUE or Gizmo), but requiring some basic UI interaction, you'd read the board's buttons with common `digitalRead()` calls in

your `loop()` function and then take some action accordingly, such as switching among tabs in a LittlevGL user interface. This is extremely specific to your own code and UI design though, there is no one-size-fits-all example we can provide here...you'll need some familiarity with basic Arduino-ing and LittlevGL-ing.

Using LittlevGL with Other Displays

Most of our color TFT and OLED displays could be used with LittlevGL, but the examples are all written for devices that offer lots of pixels... the utility of LittlevGL might be lost on smaller displays, but you're welcome to experiment. PyGamer, for example, the screen is only 160x128 pixels and non-touch...but the joystick and buttons might still make it useful in unexpected ways.

To try out a different display, you can usually create a mashup starting with one of our **LvGL_Glue** examples, plus a simple graphics test from that particular display library. Extract those parts that define and initialize the display, then pass the corresponding display object to the Glue library's `begin()` function.

For example, using a SSD1351-based OLED display, one might start with this:

```
#include &lt;Adafruit_GFX.h&gt;
#include &lt;Adafruit_SSD1351.h&gt;
#include &lt;SPI.h&gt;

#define SCREEN_WIDTH  128
#define SCREEN_HEIGHT 128 // Change this to 96 for 1.27" OLED.

#define DC_PIN   4
#define CS_PIN   5
#define RST_PIN  6

Adafruit_SSD1351 tft(SCREEN_WIDTH, SCREEN_HEIGHT, &amp;SPI, CS_PIN, DC_PIN,
RST_PIN);
```

And then, in the `setup()` function, initialize the screen and hand it off to the glue library:

```
tft.begin();

  // Initialize glue, passing in address of display
  LvGLStatus status = glue.begin(&amp;tft);
  if(status != LVGL_OK) {
    Serial.printf("Glue error %d\r\n", (int)status);
    for(;;);
  }
```

That's just the setup...there would be additional code to create the LittlevGL interface and periodically call that library's task handler, exactly like our examples for other devices.

The lesser resolution of these displays will limit what can be done there with LittlevGL, but it might still prove useful for a few small status indicators. If you really want to work at it, LittlevGL can be “themed” with different visual styles, and you might find something less busy that works well with less screen real estate.

See the LittlevGL documentation (https://adafru.it/Rva) for more ideas!