Table des matières

1 Installation de l'environnement Arduino

1.1 Création d'un utilisateur

Les développements se feront sous Linux. Nous allons créer un utilisateur pour faciliter

```
# adduser r204
Nouveau mot de passe : r204
Retapez le nouveau mot de passe : r204

# usermod -a -G dialout r204
```

Prévoir 4 Go pour cet utilisateur (développement Arduino).

Par ailleurs, il faudra penser de temps à temps à faire du ménage dans tmp.

```
$ rm -fr /tmp/arduino-* /tmp/55770067919477794* /tmp/.arduinoIDE-unsaved* /tmp/gdb-server-console-* /tmp/arduino-*
```

1.2 Installation du logiciel

Récupération du paquet https://www.arduino.cc/en/software (version linux x86-64)

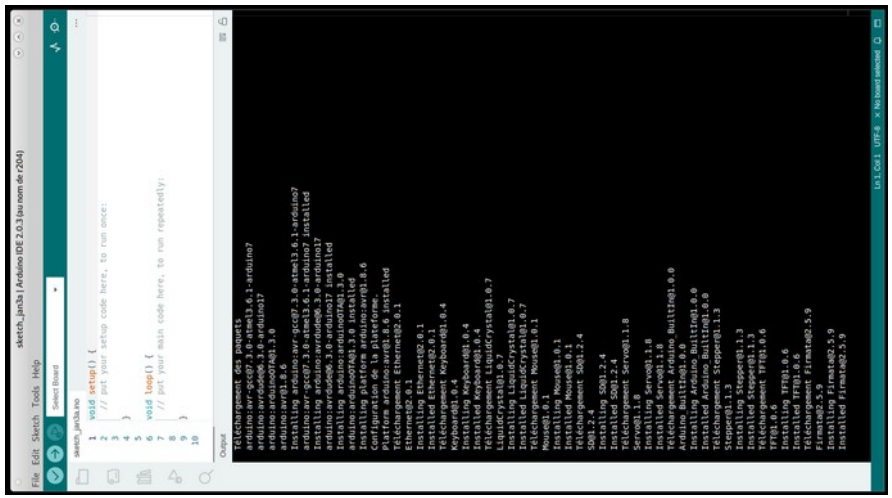
```
# cd /opt
unzip .../arduino-ide_2.0.3_Linux_64bit.zip
```

1.3 Personnalisation de l'IDE

Nous allons maintenant nous connecter avec notre utilisateur dédié et réaliser le premier lancement de l'IDE

```
/opt/arduino-ide_2.0.3_Linux_64bit/arduino-ide
```

Il y a une phase d'initialisation au premier lancement.



1.4 Installation de la carte Espressif

Pour pouvoir utiliser la carte WT32-SC01, il faut regarder la documentation :

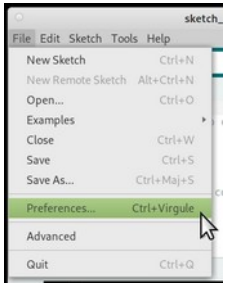
```
https://docs.espressif.com/projects/arduino-esp32/en/latest/installing.html
```

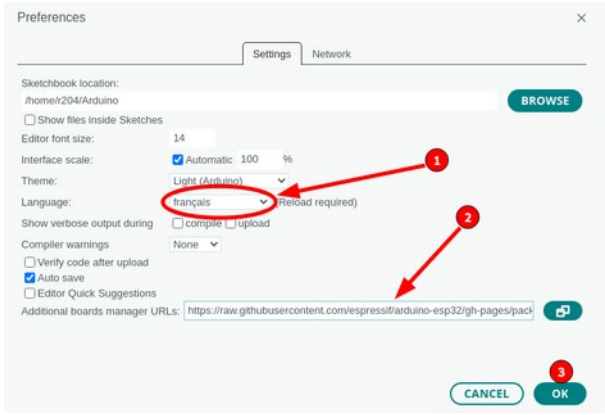
Elle indique le lien qu'il faut ajouter dans « Aditional boards manager URLs »

```
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
```

Pour réaliser cette personnalisation, il faut choisir dans le menu « File » => « Preferences... »

- ❶ choix langue française
- ❷ copier-coller le lien pour ESP32
- ❸ validation





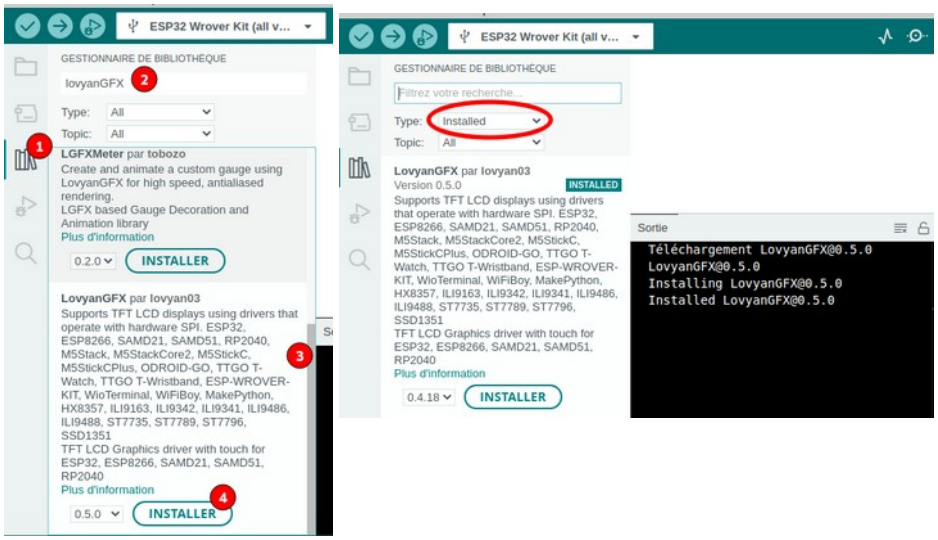
Il faut ensuite sélectionner la carte ESP32

- ❶ choix de l'onglet « Gestionnaire de carte »
- ❷ saisir esp32
- ❸ installer (il faut attendre quelques minutes)

1.5 Bibliothèques graphiques

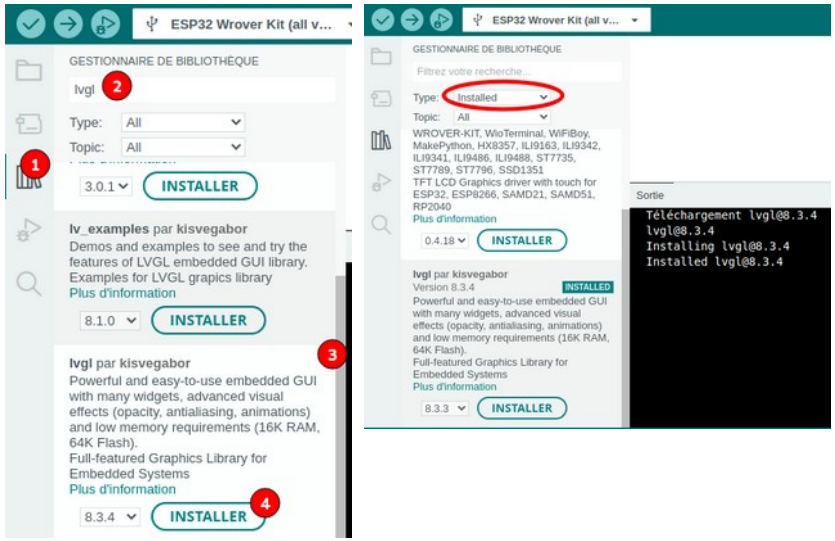
Il faut ensuite ajouter quelques bibliothèques pour gérer l'écran (partie lcd et tactile).

- ❶ choix de l'onglet « Gestionnaire de bibliothèques »
- ❷ saisir « LovyanGFX »
- ❸ descendre avec l'ascenseur
- ❹ installer « LovyanGFX par lovyan03 »



Même chose pour la les fonctions d'interface graphiques (widgets…).

- ❶ choix de l'onglet « Gestionnaire de bibliothèques »
- ❷ saisir « lvgl »
- ❸ descendre avec l'ascenseur
- ❹ installer « lvgl de kisvegabor »



Des répertoires sont apparus dans votre environnement Arduino.

```
$ tree -L 2 Arduino/
Arduino/
└── libraries
    ├── LovyanGFX
    └── lvgl
```

Il faut personnaliser la bibliothèque « lvgl » suivant les indications sur :

```
https://docs.lvgl.io/master/get-started/platforms/arduino.html#configure-lvgl
```

Il faut copier le fichier `lv_conf_template.h et le modifier.`

```
$ cp ~/Arduino/libraries/lvgl/lv_conf_template.h ~/Arduino/libraries/lv_conf.h
$ emacs !$
ligne 15 changer 0 en 1
ligne 88 changer 0 en 1
```

Ne pas installer « TFT_eSPI », nous utiliserons « LovyanGFX » à la place.

Il existe souvent deux façons d'installer les criques logicielles :

- Avec l'interface graphiques

- en clonant l'espace git

Par exemple, nous aurions pu installer la bibliothèque « LovyanGFX » avec les commandes :

```
$ cd ~/Arduino/libraries
$ git clone ttps://github.com/lovyan03/LovyanGFX/
```

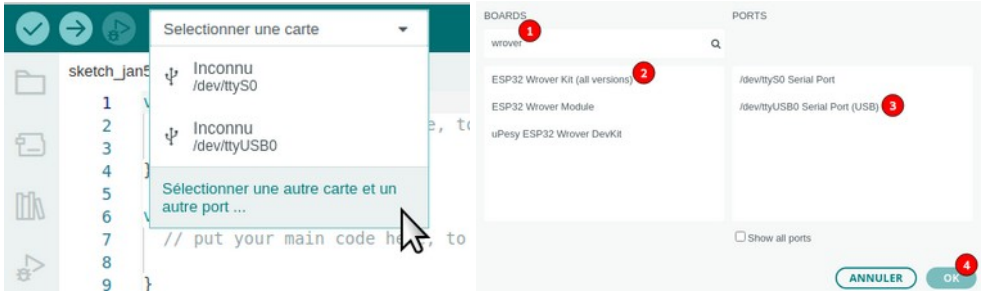
1.6 Test du Wifi

Nous en profitons pour créer un dossier pour sauvegarder nos réalisations.

```
$ mkdir ~/Arduino/softs
```

Les exemples proposés dans le menu fichier dépendent de la carte sélectionnée et des bibliothèques disponibles. Il faut donc commencer par brancher votre matériel sur le port USB puis sélectionner la bonne carte.

- ❶ saisir comme filtre « wrover »
- ❷ choisir « ESP32 Wrover Kit (all versions) »
- ❸ choisir « *dev*ttyUSB0 Serial Port (USB) »
- ❹ cliquer sur « OK »

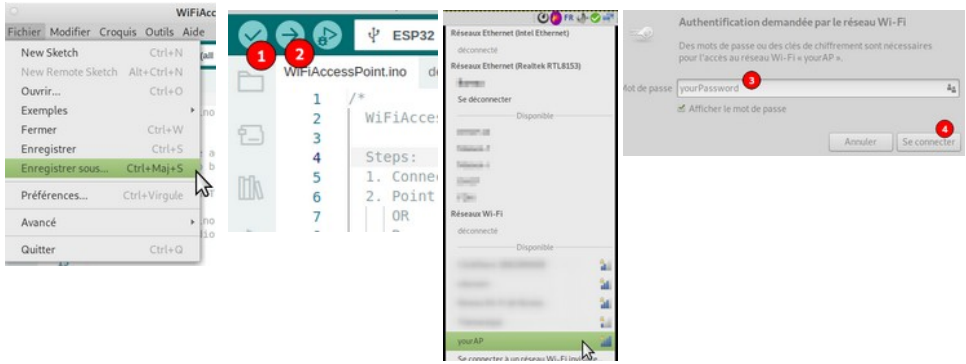


Nous allons récupérer l'application d'exemple de mis en place d'un point d'accès Wifi.

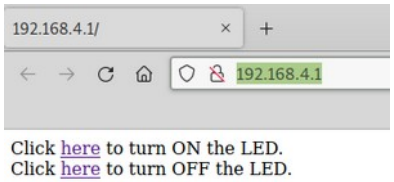
Fichier => Exemples => (Exemples pour ESP32 Wrover Kit (all versions) / Wifi => WifiAccessPoint

Ensuite nous l'enregistrons dans notre répertoire « softs » et

- ❶ compiler, voir
- ❷ téléverser sur le matériel, un nouveau réseau wifi va apparaître de nom « yourAP »
- ❸ saisir le mot de passe « yourPassword »
- ❹ cliquer sur « Se connecter »



Dans un navigateur, on peut vérifier le fonctionne sur l'adresse « http://192.168.4.1/ »



1.7 Test de l'écran

Nous avons récupéré un programme déjà écrit par Sukesh Ashok Kumar sur son git.

```
$ cd Arduino/softs/
$ mkdir Esp31-LovyanGFX-Lvgl
$ cd !$
$  wget  https://raw.githubusercontent.com/sukesh-ak/LVGL8-WT32-SC01-Arduino/main/src/main.cpp -O Esp31-LovyanGFX-Lvgl.ino
$ emacs!$
Ajouter à la ligne 4 la définition suivante
#define LGFX_WT32_SC01
```

Ouvrir le fichier. Ne pas oublier de sélecter la bonne carte et compiler.

2 Exercices

Vous allez pouvoir vous entraîner avec quelques exorcises qui ne seront pas notés. Prenez le temps de bien les comprendre.

2.1 Le Wifi

Téléchargez le ficher « 00-WifiAcessPoint.ino » depuis

```
https://dept-iut-info-vannes-cloud.kaz.bzh/s/xr4DXBn6QK22Fjo?path=%2Fsrc%2Fino
```

Ouvrez-le avec le logiciel arduino-ide.

Pour ne pas avoir de conflit dans les noms de Wifi, modifiez les paramètres en tête du fichier :

```
const char *ssid = "R204-XXX";
```

Changez XXX par votre nom.

Compilez, téléverser sur le matériel.

Connectez-vous à votre réseau et ouvrez un navigateur sur

```
http://192.168.4.1/
```

Pensez à enregistrer votre travail.

2.2 Une première interface graphique

Essayez ensuite l'élément graphique clavier « 15-Keyboard.ino ».

Compilez, téléverser sur le matériel et essayez.

2.3 Autres éléments

Terminez votre TP en choisissant un ou deux exemples à réaliser dans le même répertoire.

3 Quelques références

```
Src : https://github.com/lovyan03/LovyanGFX
Doc : https://lovyangfx.readthedocs.io/en/master/

Src : https://github.com/lvgl/lvgl
Doc : https://lvgl.io/
Doc : https://docs.lvgl.io/master/widgets/index.html

Src : https://github.com/sukesh-ak/LVGL8-WT32-SC01-Arduino
Src : https://github.com/sukesh-ak/ESP32-LVGL8x-SDSPI
```

```
Doc : https://www.arduino.cc/reference/en/libraries/wifi/
Doc : https://www.arduino.cc/reference/en/libraries/ethernet/
```

4 Truc et astuces

4.1 Raccourcis

- ^+R : compilation
- ^+U : compilation+téléversement

4.2 Les erreurs de compilations

Le compilateur utilisé possède des limites :

- Le type de retour des fonctions (y compris les « qualifiers » tel que « static ») doivent se trouver sur la même ligne que le nom de la fonction. Il faut ne pas écrire :

```
static void
event_handler (lv_event_t *e) {
```

mais

```
static void event_handler (lv_event_t *e) {
```

- Il faut utiliser des « cast » explicites. Il faut ne pas écrire

```
lv_obj_t *label = lv_event_get_user_data (e);
```

mais

```
lv_obj_t *label = (lv_obj_t*) lv_event_get_user_data (e);
```

- Comportement spécifique du « #define »

Un certain nombre de directive de compilation sont prépositionnés dans lv_conf.h et il peut être difficile de faire des tests qui utilisent ces variables (tel que le paramètre LV_USE_CALENDAR_HEADER_DROPDOWN)

- Le message suivant indique simplement qu'il n'y a pas suffisamment de mémoire. Il faut diminuer la taille des données.

```
section `.dram0.bss' will not fit in region `dram0_0_seg'
```

- Les messages suivants peuvent indiquer une erreur de paramètre.

```
error: invalid conversion from 'int' to 'lv_meter_scale_t*' [-fpermissive]
```

ou

```
error: cannot convert 'lv_color_t' {aka 'lv_color16_t'} to 'uint16_t' {aka 'short unsigned int'}
```

Cela arrive par exemple lorsque qu'un paramètre a été ajouté.

```
lv_meter_scale_t *scale = lv_meter_add_scale (meter1);
lv_meter_set_scale_ticks (meter1, scale, 41, 2, 10, lv_palette_main (LV_PALETTE_GREY));
```

ou

```
lv_meter_scale_t *scale = lv_meter_add_scale (meter);
lv_meter_set_scale_ticks (meter, scale, 0, 0, 0, lv_color_black());
```

- Le message suivant indique qu'il faut activer une ressource.

```
error: 'lv_font_montserrat_22' was not declared in this scope
```

En changeant

```
#define LV_FONT_MONTSERRAT_22 0
```

Par

```
#define LV_FONT_MONTSERRAT_22 1
```

- Dans le cas où un message indiquant qu'il manque une fonction qui visiblement est une fonction standard C, il faut utiliser la version stand

```
error: 'lv_malloc' was not declared in this scope
```

Dans ce cas, il faut remplacer par

```
malloc
```

- Il arrive souvent qu'un problème de type soit indiqué pour des valeurs constantes (comme 0) comme le message suivant :

```
error: invalid conversion from 'int' to 'lv_style_prop_t' [-fpermissive]
```

Par exemple, il faut remplacer :

```
static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, 0};
```

par

```
static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, (lv_style_prop_t) 0};
```

- SI vous avez un message qui parle de mémoire comme celui ci-dessous

```
Le croquis utilise 571969 octets (43%) de l'espace de stockage de programmes. Le maximum est de 1310720 octets.
Les variables globales utilisent 83712 octets (25%) de mémoire dynamique, ce qui laisse 243968 octets pour les variables locales. Le maximum est de 327680 octets.
```

Alors il est temps de faire du ménage. Il se peut également que vous ayez activé des options inutiles dans lv_conv.h. Parfois, il suffit de compiler à nouveau.

4.3 Conversion d'images en structure C

```
https://github.com/lvgl/lv_img_conv
```

4.4 Intégration de police de caractères

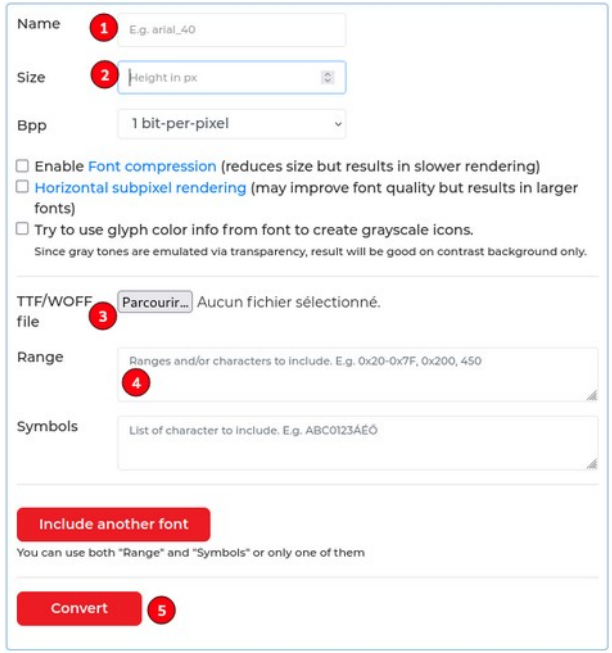
Vous pouvez utiliser n'importe quelle police de caractères avec un outil en ligne qui traduira la spécification TTF en un fichier C à ajouter à vos sources.

Le lien est :

```
https://lvgl.io/tools/fontconverter
```

Les informations sont :

- ❶ un nom court que vous utilisez comme identifiant (exemple : lib_48)
- ❷ la taille en point (exemple : 48)
- ❸ téléverser un fichier TTF qui se trouve sur votre disque (exemple : /usr/share/fonts/truetype/liberation/LiberationMono-Regular.tff)
- ❹ N'oubliez pas intervalle des caractères que vous souhaitez. Les caractères ASCII sans accent se trouve avant 0X7F. Pour le latin accentué, il faudra poursuivre jusqu'à 0xFF. Vous pourrez également utiliser des caractères « symboles » UTF-8. (exemple : 0x20-0xFF).
- ❺ Terminez en cliquant sur « Convert » qui fournira un lien de téléchargement du fichier C.



Dans le fichier obtenu, changer :

```
#ifdef LV_LVGL_H_INCLUDE_SIMPLE
#include "lvgl.h"
#else
#include "lvgl/lvgl.h"
#endif
#ifndef LIB_48
#define LIB_48 1
#endif
#if LIB_48
```

par

```
#include <lvgl.h>
```

et supprimer en fin de fichier :

```
#endif /*#if LIB_48*/
```

Dans le fichier qui utilise cette police de caractère, il faut ajouter au début :

```
extern lv_font_t lib_48;
```