INSTITUT
UNIVERSITAIRE
DE TECHNOLOGIE
UNIVERSITÉ D'ANGERS

Rapport de SAE - GEII 3

SAE Ecrans LCD

Rédigé par DRIBEK Adem & Layla QUEYSSAC

Encadrant : Monsieur Philippe LUCIDARME

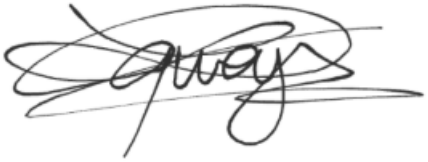
DRIBEK Adem – QUEYSSAC Layla Année 2024-2025

Décharge de responsabilités

Ce rapport présente le travail réalisé par un groupe d'étudiants dans le cadre d'un projet pédagogique. Les auteurs et l'Université d'Angers ne garantissent pas que l'information, les documents, la méthodologie et le matériel présentés dans ce document soient complets, conformes à l'état de l'art et exacts ni n'assurent en toutes circonstances la sécurité des biens, des personnes et des utilisateurs. Les auteurs et l'Université d'Angers ne seront pas tenus responsables des dommages éventuels qui pourraient résulter de l'utilisation du contenu du présent rapport.

Licence

DRIBEK Adem et QUEYSSAC Layla auteurs du présent rapport, publions et divulguons celui-ci sous la Licence Creative Commons suivante « CC BY » pour le monde entier et pendant la durée légale de protection des droits d'auteur. Cette licence autorise la représentation, la reproduction, la modification, la création d'œuvres dérivées et l'utilisation y compris à des fins commerciales sous réserve de mentionner les noms et prénoms des auteurs.



QUEYSSAC Layla DRIBEK Adem

Remerciements :

Tout d'abord, nous tenons à remercier tout particulièrement et à témoigner toute notre reconnaissance aux personnes suivantes, pour leur soutien dans la concrétisation de ce projet :

- M. Philippe LUCIDARME, responsable projet, pour ses conseils éclairés, sa patience, sa disponibilité et sa confiance qu'il nous a accordée lors de ce projet ; ainsi que pour sa mise à notre disposition de tous les moyens disponibles afin de mener à bien ce projet.
- Le département GEII de l'IUT d'Angers-Cholet pour sa coopération professionnelle tout au long de cette expérience et pour s'être mis à notre disposition lors de la réalisation de ce projet.

Introduction

Ce projet s'inscrit dans une démarche collaborative répartie entre plusieurs groupes, chacun étant chargé d'un aspect spécifique du développement. L'objectif global était de concevoir un système interactif permettant de contrôler la navigation sur un téléphone ou une page internet à l'aide d'un capteur TOF (Time of Flight), d'un écran tactile LCD et d'un microcontrôleur ESP32-C6, assurant la communication entre ces éléments via Bluetooth.

Notre groupe s'est concentré sur la conception et la gestion des écrans tactiles, tandis que les deux autres groupes avaient pour mission de travailler respectivement sur le capteur TOF et sur la programmation et l'intégration de l'ESP32-C6. Cette répartition des tâches a permis de couvrir toutes les facettes du projet de manière approfondie tout en favorisant l'intégration des différentes technologies.

Objectif de notre partie du projet :

Notre mission était de développer une interface graphique tactile fonctionnelle et intuitive en utilisant l'écran LCD GC9A01A et un contrôleur tactile capacitif CST816S. Cet écran joue un rôle central dans le système global en permettant :

1. La visualisation et navigation au sein de l'interface utilisateur.
2. L'interaction tactile pour contrôler certaines fonctionnalités, comme la configuration des paramètres du système ou la gestion de la navigation.
3. Une intégration harmonieuse avec les autres modules, notamment le capteur TOF et l'ESP32-C6.

Contexte général :

Dans le système complet :

- Le capteur TOF permet de détecter les mouvements de l'utilisateur ou des gestes, pour interagir avec le téléphone sans contact.
- L'ESP32-C6, en tant que contrôleur principal, gère les données provenant de l'écran tactile et du capteur TOF, et communique avec le téléphone via Bluetooth.
- L'écran GC9A01A, développé par notre groupe, affiche une interface claire et dynamique qui permet de gérer les fonctionnalités du système tout en offrant un retour visuel à l'utilisateur.

Table des matières

I. Cahier des charges

Le cahier des charges présenté se divise en deux parties distinctes : "Écran sans ESP32" et "Écran avec ESP32". Ces deux approches permettent d'aborder différentes étapes de développement et d'intégration entre l'écran et les systèmes ESP32.

C. Écran sans ESP32

Dans cette première partie, les objectifs se concentrent sur la mise en œuvre de l'affichage et des fonctionnalités de l'écran indépendamment des capacités Bluetooth et de communication de l'ESP32. Les étapes suivantes sont prévues :

- Réalisation du câblage : Connexion physique entre l'écran et un module ESP32-C6 DevKit M1.
- Prise en main de l'affichage : Apprentissage et maîtrise des fonctionnalités d'affichage offertes par l'écran.
- Prise en main du tactile : Intégration et exploitation des fonctionnalités tactiles pour permettre une interaction utilisateur.
- Création d'une interface graphique : Conception et développement d'une interface visuelle pour l'utilisateur.

D. Écran avec ESP32

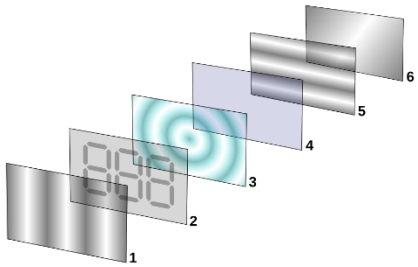
Cette seconde partie intègre l'utilisation des fonctionnalités avancées de l'ESP32-S3 pour améliorer l'expérience utilisateur et la connectivité. Les étapes suivantes sont prévues :

- Prise en main des librairies constructeurs : Utilisation des librairies spécifiques pour l'écran et l'ESP32-S3 afin de faciliter le développement.
- Connexion Bluetooth : Mise en place de la communication sans fil via le mode BLE.
- Communication entre téléphone et ESP32 : Développement d'une interface de communication entre un téléphone et l'ESP32 pour transmettre des informations.
- Création d'une interface graphique : Développement d'une interface visuelle adaptée aux fonctionnalités connectées.

II. Les technologies LCD

A. LCD

Les écrans LCD (Liquid Crystal Display) utilisent la polarisation de la lumière afin d'afficher ce que l'on veut sur l'écran. Ils sont constitués de cristaux liquides qui vont s'orienter en fonction du courant électrique. Ainsi, ils modifient la polarité de la lumière (modification la vibration du champ électrique). Ce type d'écran est composé de deux filtres polarisants, dont les directions forment un angle de 90°, ainsi que de deux plaques de verre composées d'une matrice d'électrodes enserrant des cristaux liquides.

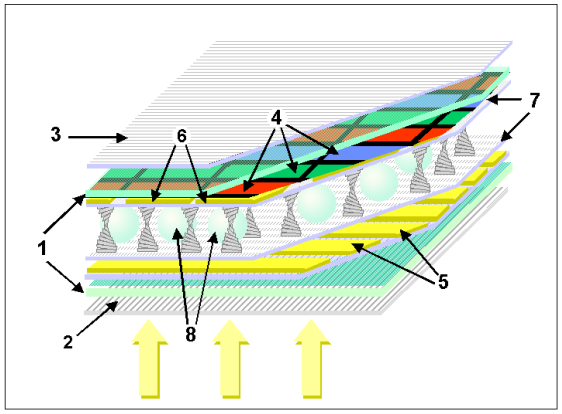


1 et 5 : filtres polarisants ;
2 : électrodes avant ;
4 : électrode arrière ;
3 : cristaux liquides ;
6 : miroir.

Il existe différentes technologies d'écran, avec pour chacune d'entre elles différentes propriétés et technologies.

1. TN (Twisted Nematic)

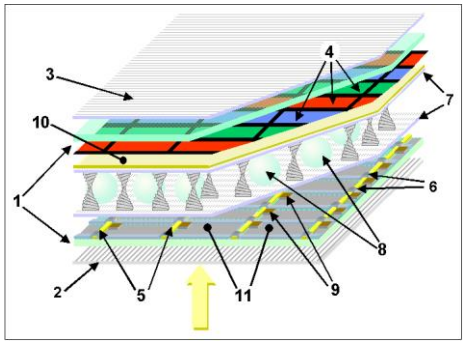
La technologie TN était la technologie la plus répandue et la plus économique puisqu'il s'agissait de la technologie qui offre les meilleures performances en termes de rapidité/réactivité. En effet, la position des cristaux liquides peut varier très rapidement sous l'effet d'un champ électrique. Cela permet d'adapter l'image aux contenus fortement dynamiques. Cependant, cette technologie s'est uniquement centrée sur cet aspect, au détriment de la qualité de l'image (des contrastes notamment) et des angles de vision.



1 : plaque de verre ;
2 et 3 : polarisants vertical et horizontal ;
4 : filtre couleur RVB ;
5 : électrodes verticales ;
6 : électrodes horizontales ;
7 : couches polymère d'alignement ;
8 : billes d'espacement.

2. TFT (Thin-film transistor)

La technologie TFT est quant à elle, la plus utilisée pour les écrans de couleurs et la TV. Cette technologie remplace la grille d'électrodes avant par une seule électrode qui est fabriquée en oxyde d'indium-étain. La grille d'électrodes arrière est supprimée par une matrice de transistors en film mince (TFT). La matrice est composée de trois transistors par pixel de couleurs. Cela permet d'améliorer le temps de réponse et la stabilité de l'affichage.



1 : plaque de verre ;
2 et 3 : polarisants vertical et horizontal ;
4 : filtre couleur RVB ;
5 : lignes de commande horizontales ;
6 : lignes de commande verticales ;
7 : polymère d'alignement ;
8 : billes d'espacement ;

3. IPS (In-Plane Switching)

La technologie IPS est généralement utilisée pour les afficheurs dynamiques. Elle perfectionne la technologie TN et TFT car l'utilisation de l'axe des cristaux liquides est parallèle au plan de l'écran. Cela a permis d'élargir les angles de vision de l'écran. Cependant, comme les autres écrans LCD, le contraste est moindre à cause du rétroéclairage.

4. VA (Vertical Alignment)

La technologie VA permet d'améliorer le contraste qui est moyen sur les autres technologies LCD dû au rétroéclairage. En effet, les cellules de cette technologie permettent de bloquer la lumière de manière beaucoup plus efficace et donc, de créer ainsi une sorte de « rétroéclairage local ». Les couleurs et les contrastes sont beaucoup plus profonds, ce qui permet au à la technologie VA de venir concurrencer les autres technologies sur cet aspect. En revanche, les angles de vision sont beaucoup moins larges, en comparaison à la technologie IPS par exemple. L'image pourra donc paraitre délavée à certains endroits en fonction d'où on se positionne (en fonction de l'angle et de la distance). Quant à la réactivité, la technologie VA offre de meilleures performances que la dalle IPS, mais la différence n'est pas flagrante.

5. QLED (quantum dot light emitting diode)

La technologie QLED offre des performances et une qualité d'image presque identique à l'OLED. C'est technologie exclusivement fabriquée et distribuée par Samsung. Il s'agit d'un moniteur LCD à rétroéclairage LED haut de gamme. Cette technologie repose sur des nano cristaux liquides en alliage de métal, qui ne créent pas l'image, mais qui permettent au rétroéclairage d'être plus efficace. L'écran offre en effet une gamme de couleurs beaucoup plus étendue.

B. OLED

Les écrans OLED (Organic Light-Emitting Diode) utilise cette diode électroluminescente qui permet de produire de la lumière. La structure de cette diode est constituée d'une superposition de couches semi-conductrices organiques entre deux électrodes. Au moins l'une de ces couches doit être transparente. Lorsqu'une tension électrique appropriée est appliquée entre la cathode et l'anode, les électrons et les trous des couches se combinent dans la couche d'émission pour former l'électroluminescence qui apparaît.

La technologie OLED est utilisée pour l'affichage dans le domaine des écrans et aussi comme panneau d'éclairage. La technologie OLED possède des avantages intéressants par rapport à de nombreuses autres techniques utilisées dans les écrans. La propriété électroluminescente de l'OLED ne nécessite pas d'utiliser un rétroéclairage : cela a pour conséquence une consommation moindre d'énergie mais aussi des niveaux de gris et de noir plus subtils, un meilleur affichage des couleurs, et des écrans moins épais.

C. Bilan et choix

	LCD				OLED
	IPS	VA	TN	QLED	QLED
CRISTAUX LIQUIDES	OUI	OUI	OUI	OUI	NON
RETROECLAIRAGE	OUI	OUI	OUI	OUI	NON
LUMINOSITE	BIEN	BIEN	BIEN	EXCELLENT	EXCELLENT
CONTRASTE	MOYEN	BIEN	MOYEN	BIEN	EXCELLENT
RAPIDITE	BIEN	MOYEN	EXCELLENT	EXCELLENT	EXCELLENT
ANGLES DE VISION	EXCELLENT	BIEN	MAUVAIS	EXCELLENT	EXCELLENT
RENDU DES COULEURES	BIEN	BIEN	MOYEN	EXCELLENT	EXCELLENT

Après analyse, les meilleures technologies d'écrans qui existent aujourd'hui sont les technologies OLED ou QLED-LCD. Cependant, leur coût important nous a obligé à nous rabattre sur la technologie LCD-IPS. Ainsi, nous avons trouvé deux écrans répondant à notre cahier des charges. Un écran rond et tactile de 1.28 pouce sans ESP32, et un écran rond et tactile de 2.1 pouces.

III. Ecran sans ESP32

A. Principe de fonctionnement des composants

1. Description des composants

a. Écran GC9A01A : rôle et spécifications

L'écran GC9A01A est un écran LCD circulaire de 1,28 pouce basé sur la technologie IPS. Il est principalement conçu pour des applications embarquées et interactives telles que les montres connectées, les tableaux de bord ou encore les interfaces utilisateur compactes.

Spécifications techniques principales :

- Résolution : 240 x 240 pixels.
- Interface : SPI (Serial Peripheral Interface).
- Profondeur de couleur : Jusqu'à 65 536 couleurs (16 bits).
- Tension d'alimentation : 3,3V.
- Rafraîchissement rapide : Idéal pour des interfaces réactives.

Rôle :

Cet écran sert à afficher des informations visuelles telles que des boutons, des textes ou des graphiques. Dans ce projet, il est utilisé pour implémenter une interface graphique à plusieurs pages (« Menu », « Infos », « Couleurs »).

b. Contrôleur tactile CST816S : rôle et spécifications

Le CST816S est un contrôleur tactile capacitif permettant la détection d'appuis et de mouvements sur une surface tactile associée à l'écran.

Spécifications techniques principales :

- Interface : I2C (Inter-Integrated Circuit).
- Précision : Capable de détecter des positions avec une résolution adaptée à l'écran.
- Détection : Gère un seul point de contact (à une pression à la fois).
- Tension d'alimentation : 3,3V.
- Sensibilité : Optimisée pour des surfaces tactiles capacitives de petite taille.

Rôle :

Le CST816S capture les coordonnées X et Y du point d'interaction sur l'écran tactile. Ces données sont ensuite transmises au microcontrôleur via l'interface I2C pour être interprétées et associées à des actions, comme le changement de page ou de couleur dans notre application.

c. Technologie capacitive tactile

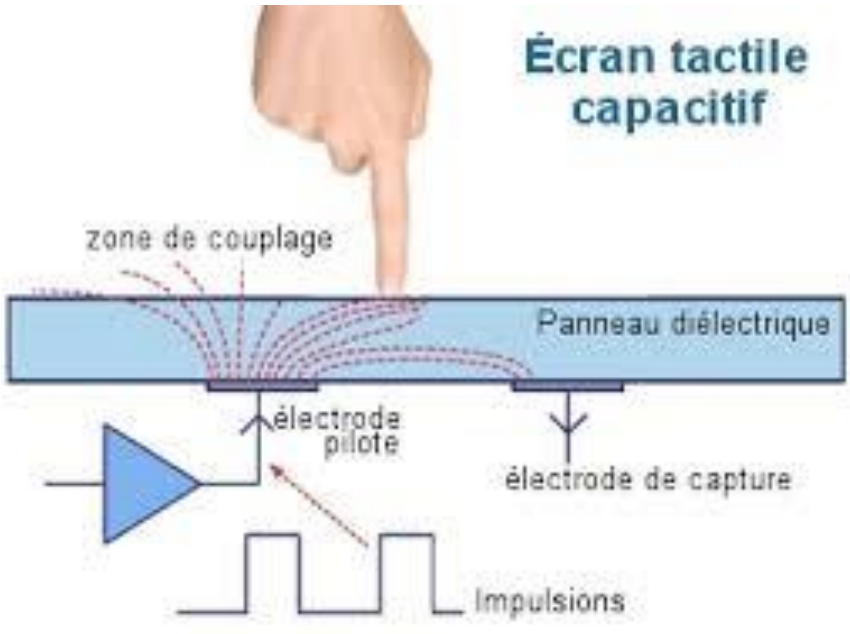
La technologie capacitive tactile repose sur le principe de détection des variations de capacité électrique. Un écran tactile capacitif est recouvert d'une couche conductrice transparente. Lorsqu'un doigt (qui est conducteur) touche l'écran, il modifie le champ électrique local à travers une petite variation de capacité.

Caractéristiques principales :

- Multi-usage : Très utilisé dans les smartphones et tablettes en raison de sa précision et de sa robustesse.
- Détection rapide : Les changements de capacité sont mesurés par des circuits intégrés comme le CST816S.
- Résistance accrue : Contrairement aux écrans résistifs, il n'y a pas de pièces mobiles, ce qui les rend durables.

Phénomène physique :

Le système capacitif repose sur la loi de Coulomb. Lorsqu'un objet conducteur (comme un doigt) s'approche de l'écran, il agit comme une plaque d'un condensateur et provoque une variation de la capacité électrique. Ces variations sont ensuite interprétées par le contrôleur tactile pour localiser précisément le point de contact.



B. Protocoles de communication

1. Explication de SPI (Serial Peripheral Interface) pour l'écran

Le protocole SPI est une interface de communication série rapide et synchrone, souvent utilisée pour des équipements à haut débit tels que les écrans LCD.

Caractéristiques principales du SPI :

- Fonctionne selon un modèle maître/esclave, ici le microcontrôleur (maître) contrôle l'écran (esclave).
- Utilise 4 lignes principales :
  - MOSI (Master Out Slave In) : Transmet les données du maître vers l'écran.
  - MISO (Master In Slave Out) : Non utilisé ici car l'écran ne renvoie pas de données.
  - CLK (Clock) : Fournit une horloge synchronisant les transmissions.
  - CS (Chip Select) : Active l'écran pour la communication.

Avantages dans ce projet :

- Vitesse de transmission élevée, permettant des affichages fluides.
- Simplicité de mise en œuvre avec l'écran GC9A01A via la bibliothèque Adafruit_GC9A01A.

2. Explication de I2C (Inter-Integrated Circuit) pour le tactile

Le protocole I2C est une interface série qui permet à plusieurs périphériques de communiquer avec un microcontrôleur via deux lignes :

- SDA (Serial Data) : Transfère les données.
- SCL (Serial Clock) : Synchronise la communication.

Caractéristiques principales du I2C :

- Chaque appareil (ici le CST816S) possède une adresse unique, permettant une identification facile.
- Fonctionne également en mode maître/esclave. Le microcontrôleur initie la communication et interroge le CST816S pour obtenir les coordonnées X et Y.

Avantages dans ce projet :

- Simplicité du câblage avec seulement deux lignes partagées.
- Réduction du nombre de broches utilisées sur le microcontrôleur.

3. Coordination entre SPI et I2C

Dans ce projet, le SPI et l'I2C fonctionnent de manière indépendante mais complémentaire :

- Le SPI est dédié à l'affichage : il gère le transfert rapide des informations graphiques vers l'écran GC9A01A.
- L'I2C est dédié à la détection tactile : il transmet les informations d'interaction utilisateur du CST816S vers le microcontrôleur.

Pour assurer une coordination fluide :

- Les deux protocoles sont configurés sur des broches différentes du microcontrôleur, évitant tout conflit.
- Les données du CST816S (I2C) sont interprétées pour déclencher les commandes de mise à jour graphique sur l'écran (SPI).
- Un délai (égal à debounceDelay) est utilisé pour synchroniser les réactions au toucher et l'actualisation de l'écran.

Grâce à la combinaison des deux protocoles, l'utilisateur peut interagir de manière fluide et naturelle avec l'interface graphique. Ce système montre une belle coordination entre les différents composants électroniques, offrant une expérience rapide et intuitive.

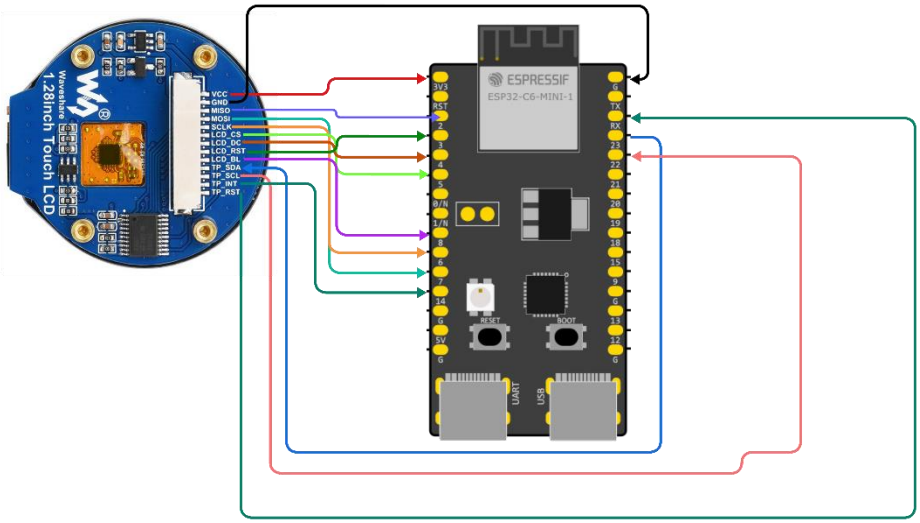
L'écran GC9A01A, avec son affichage réactif assuré par le protocole SPI, fonctionne parfaitement avec le contrôleur tactile CST816S, qui capture avec précision les coordonnées de chaque interaction via I2C.

Chaque geste tactile est immédiatement pris en compte par l'interface, garantissant une utilisation fluide et agréable. Ce duo technologique ne se contente pas d'assurer une performance solide aujourd'hui, il offre également une base robuste et évolutive pour des projets encore plus ambitieux à l'avenir.

C. Réalisation technique

1. Schéma de câblage

Le câblage de l'écran tactile GC9A01A avec le ESP32-C6 DevKit M1 est représenté dans le schéma ci-dessous. Chaque broche de l'écran est connectée à une broche GPIO spécifique de l'ESP32 pour assurer la communication SPI et le fonctionnement du tactile.



Écran LCD	ESP32-C6	Description
VCC	3.3V	Alimentation de l'écran
GND	GND	Masse
MOSI	GPIO10	Données envoyées via SPI
SCK	GPIO8	Horloge SPI
MISO	GPIO9	Données reçues via SPI
LCD_CS	GPIO6	Sélection de l'écran LCD
LCD_DC	GPIO7	Données/Commandes pour LCD
LCD_RST	GPIO5	Réinitialisation de l'écran
LCD_BL	GPIO4	Contrôle du rétroéclairage
TP_SCL	GPIO14	Horloge I^2C pour capteur tactile
TP_SDA	GPIO13	Données I^2C pour capteur tactile
TP_INT	GPIO12	Interruption tactile
TP_RST	GPIO11	Réinitialisation tactile

2. Logiciel

a. Structure globale

L'application est structurée autour de trois pages interactives :

- **Menu** : Page principale avec accès aux autres options.
- **Color** : Permet de changer la couleur de l'écran via des boutons interactifs.
- **Info** : Affiche des informations techniques sur le matériel.

Chaque page est gérée dynamiquement en utilisant des fonctions spécifiques pour l'affichage et les interactions.

b. Fonctions clés

- **readCoordinates()** :
  - Détecte les coordonnées tactiles (x, y) sur l'écran via le capteur tactile CST816S.
  - Compare les positions relevées aux zones de boutons définies pour déclencher les actions.

```
void readCoordinates() {
    if (tactileDetected()) {
        int x = getX();
        int y = getY();
        Serial.print("Coordonnées : ");
        Serial.print(x);
        Serial.print(", ");
        Serial.println(y);
        // Détection des boutons
        checkButton(x, y);
    }
}
```

- **drawScreen()** :
    - Affiche les éléments dynamiques selon la page active (Menu, Color, Info).
    - Utilise la bibliothèque **Adafruit_GFX** pour le rendu des éléments graphiques.

```
void drawScreen(String page) {
    if (page == "Menu") {
        displayMenu();
    } else if (page == "Color") {
        displayColorOptions();
    } else if (page == "Info") {
        displayInfo();
    }
}
```

- **drawButton()** :
    - Rend les boutons interactifs avec des couleurs, du texte et des formes précises.

```
void drawButton(int x1, int y1, int x2, int y2, String text, uint16_t color) {
    tft.fillRoundRect(x1, y1, x2 - x1, y2 - y1, 5, color);
    tft.setCursor(x1 + 10, y1 + 10);
    tft.setTextColor(TFT_WHITE);
    tft.print(text);
}
```

3. Démonstration du fonctionnement

a. Navigation entre les pages

- **Menu** : Accueil principal.
- **Color** : Boutons interactifs pour changer les couleurs (rouge, bleu, vert).
- **Info** : Affichage des informations matérielles telles que :
  - Nom de l'écran : GC9A01A.
  - Microcontrôleur : ESP32-C6 DevKit M1.
  - Auteurs du projet : DRIBEK Adem et QUEYSSAC Layla.

b. Changements de couleurs dynamiques

Lorsque l'utilisateur appuie sur un bouton couleur (ex. Rouge), l'écran change instantanément de fond grâce à l'actualisation rapide des commandes SPI.

Exemple pratique :

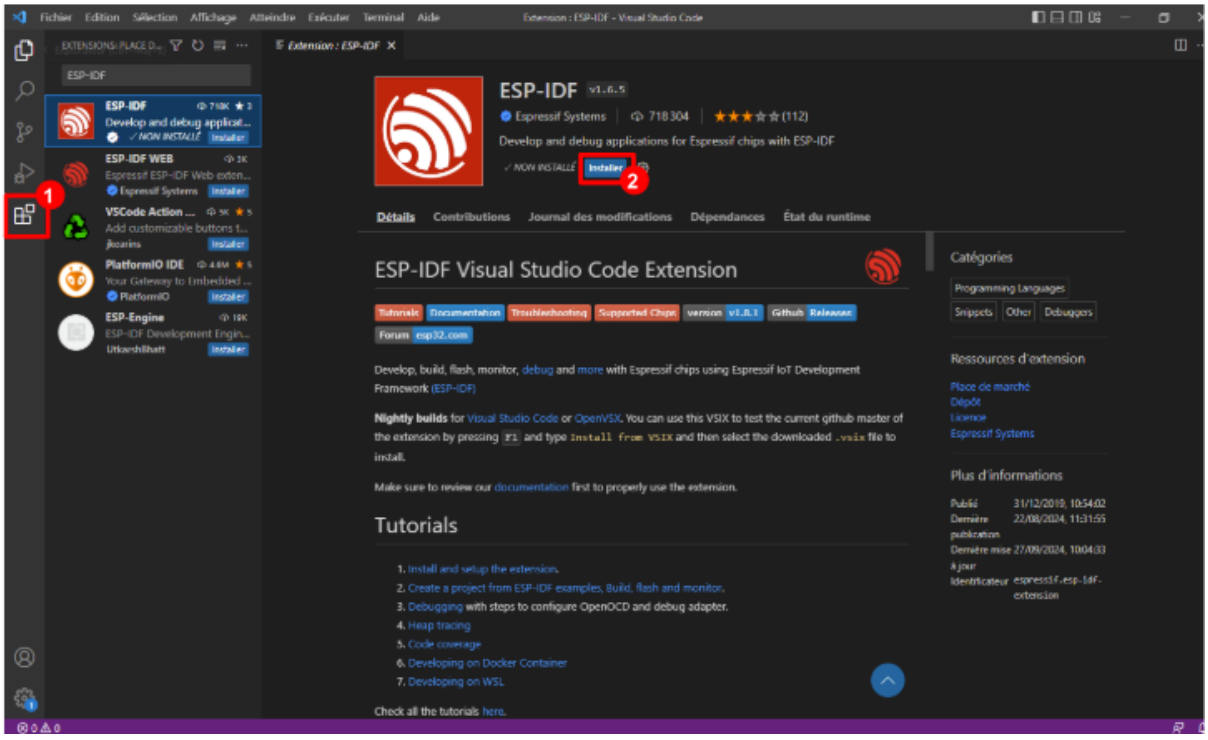
1. **Menu** → Appui tactile sur le bouton **Color**.
2. **Color** → Sélection d'une couleur **rouge**.
3. Retour au **Menu** avec le bouton dédié.

IV. Ecran avec ESP32

Pour la prise en main de l'écran avec ESP32, plusieurs étapes ont été nécessaires. Dans un premier temps, on retrouve l'utilisation du code constructeur de l'écran avec l'installation de l'extension Espressif IDF sur VS Code et l'installation de l'application Espressif-IDF. Ensuite, on a la prise en main de l'écran avec l'utilisation de la librairie *lvgl.h* pour la création d'une interface graphique. Et enfin, la découverte du mode BLE présent sur l'ESP32-S3.

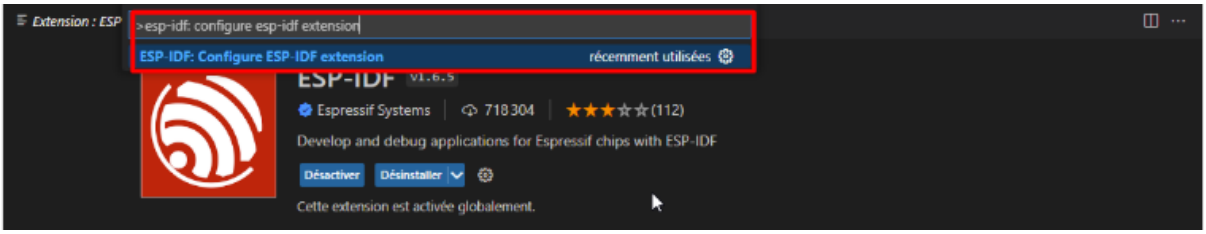
A. Programmation ESP32-S3-Touch-LCD-2.1 Waveshare

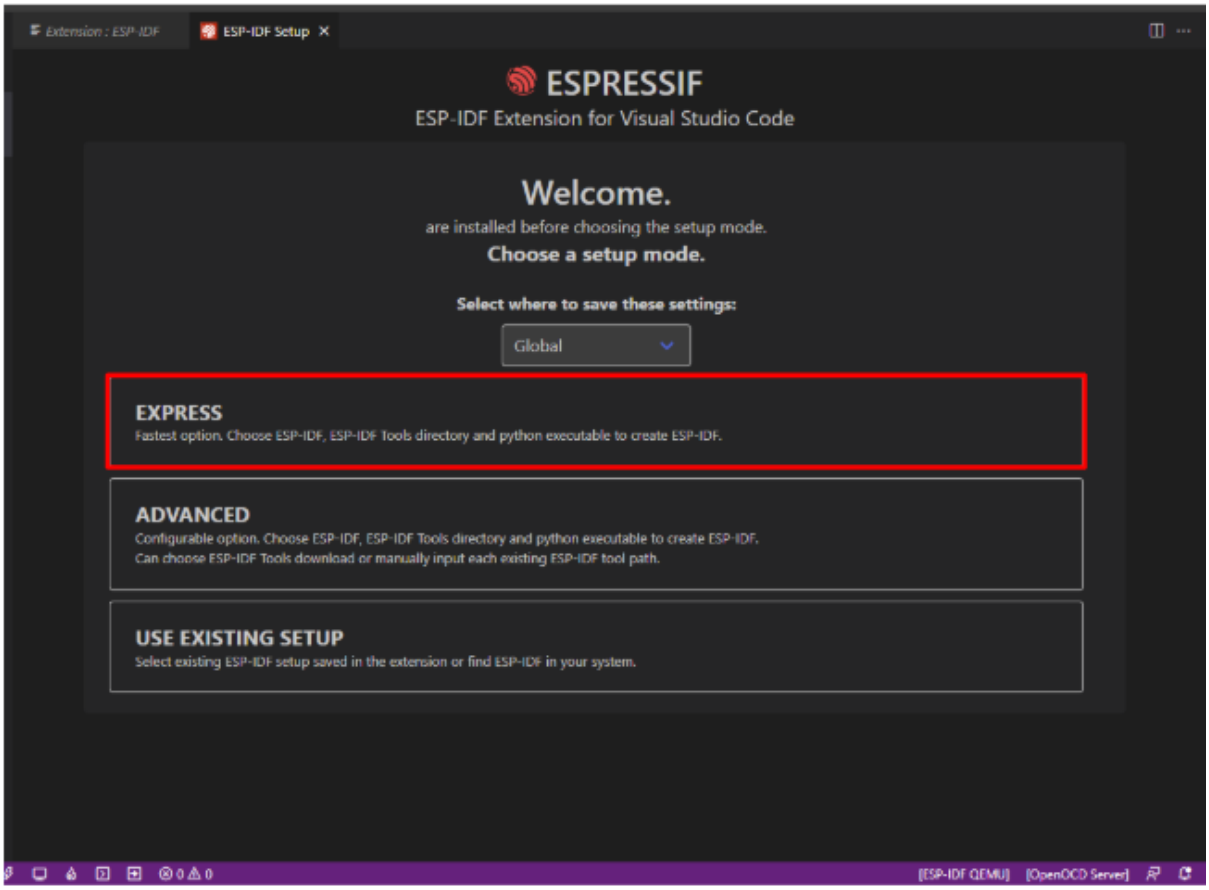
1. Installation de l'extension Expressif IDF sur VS Code

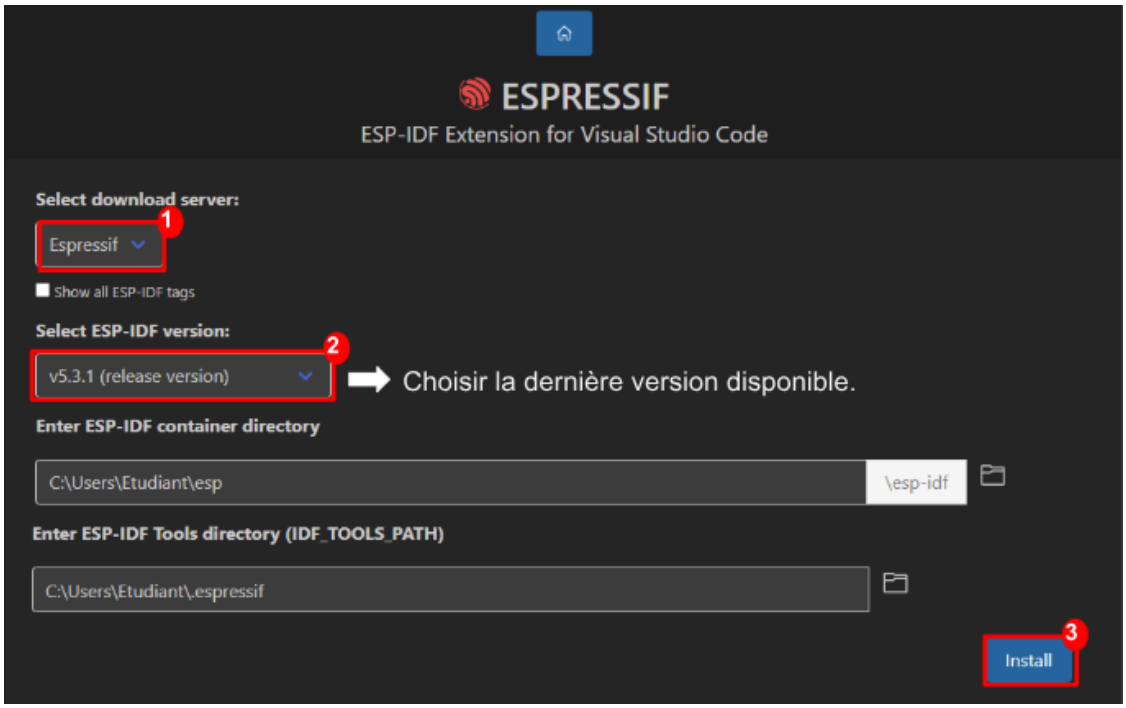


2. Configuration de l'extension ESP-IDF

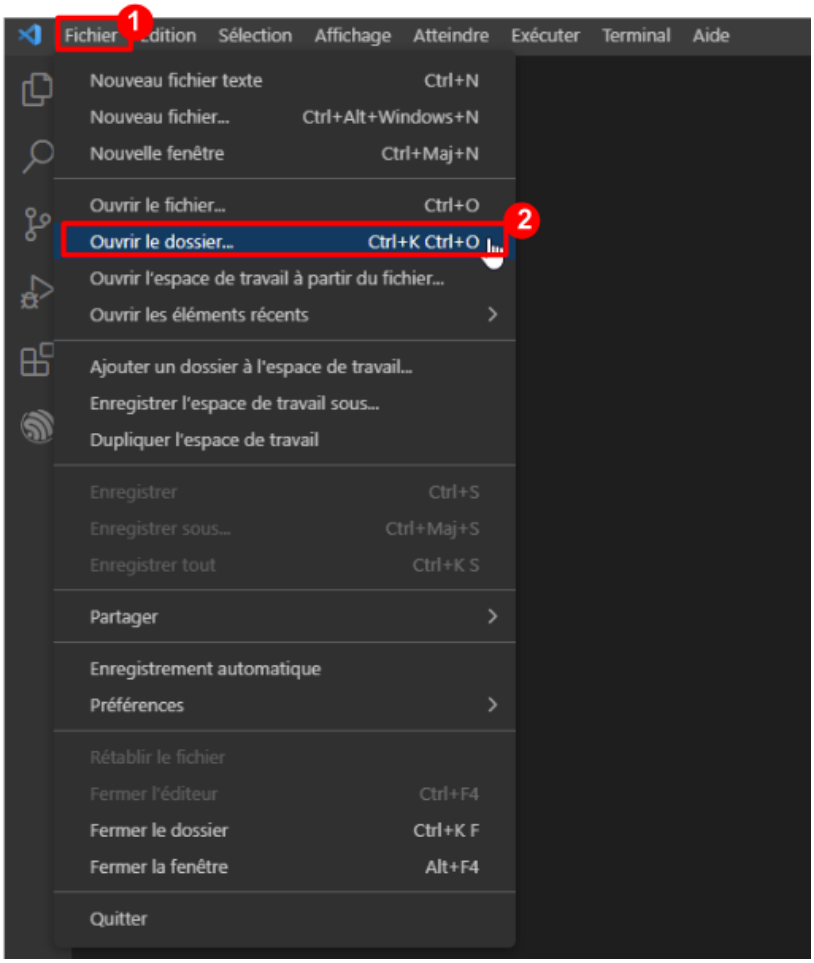
Appui sur F1 : *esp-idf : configure esp-idf extension*

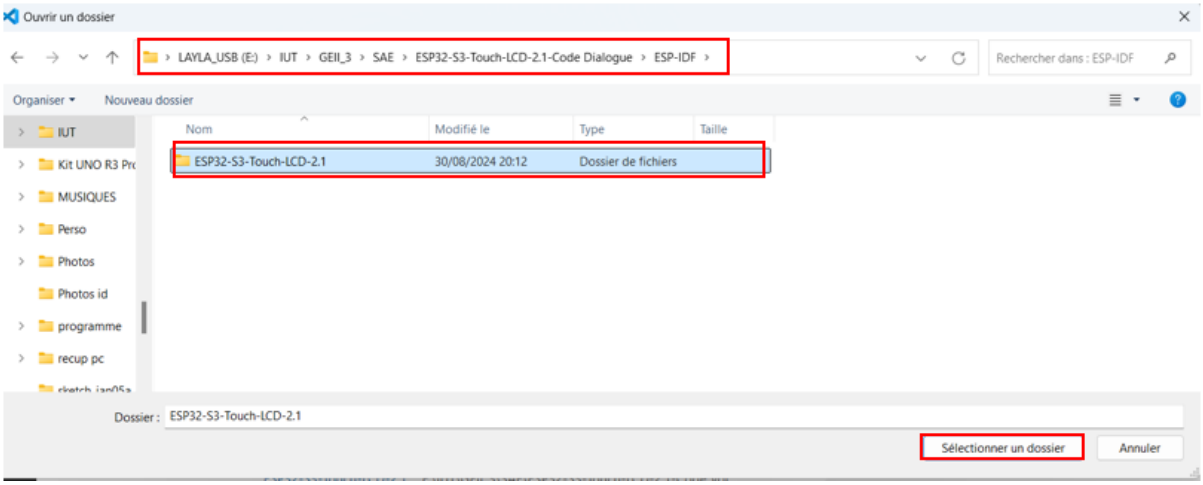






3. Ouvrir le dossier



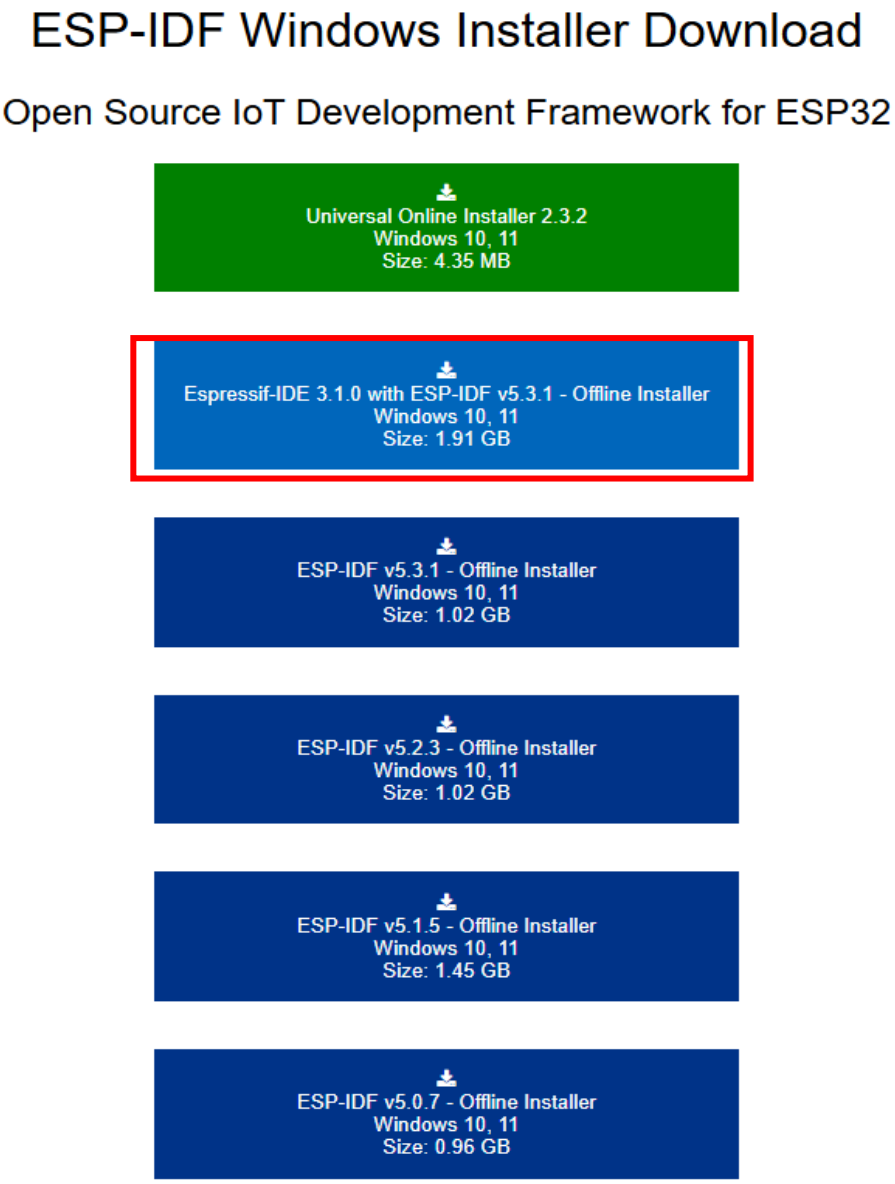


Lien du fichier : https://files.waveshare.com/wiki/ESP32-S3-Touch-LCD-2.1/ESP32-S3-Touch-LCD-2.1-Code.zip

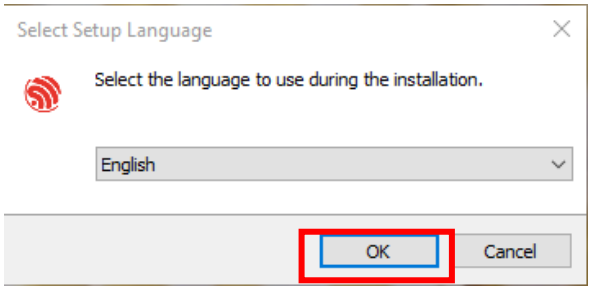
B. Installation de de l'application ESP-IDF

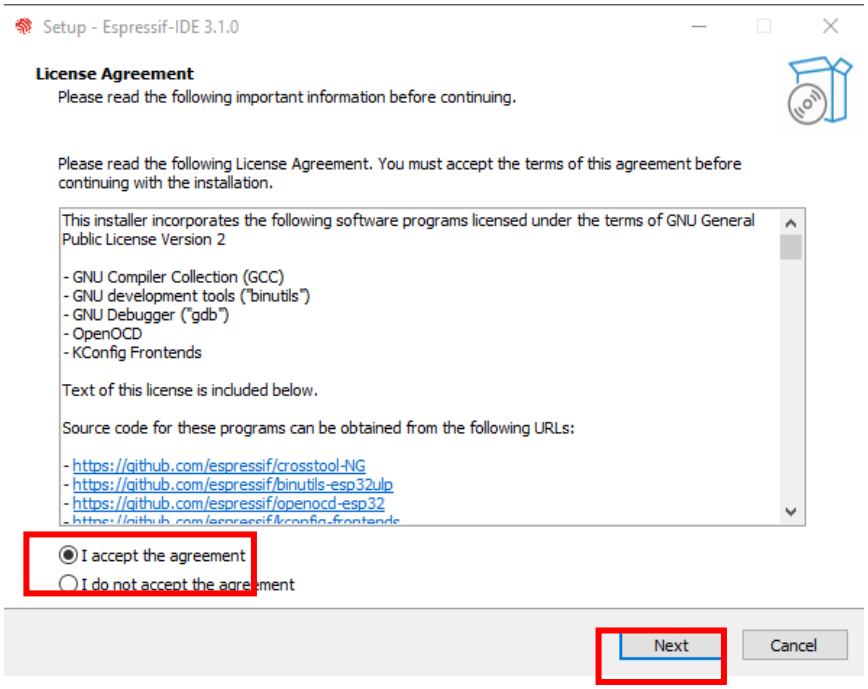
1. Installation de l'application Expressif IDF

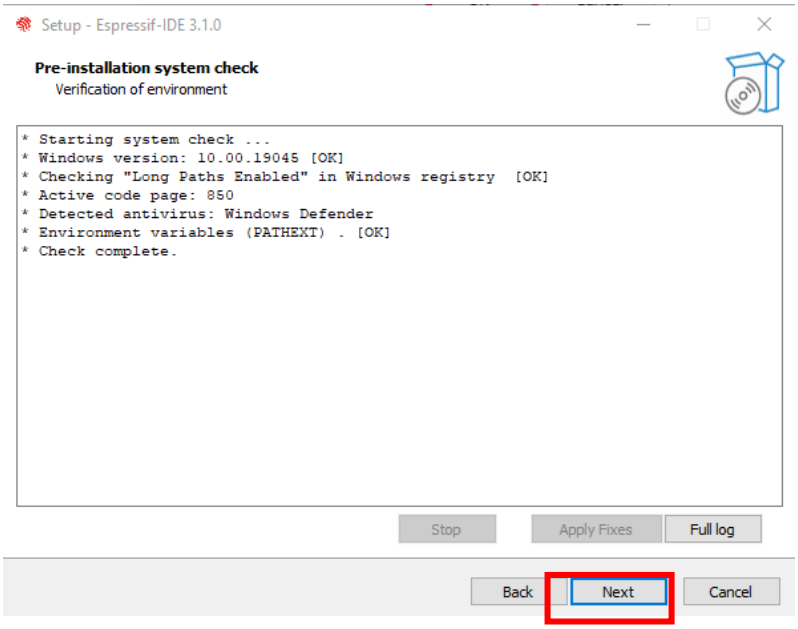
Lien : https://dl.espressif.com/dl/esp-idf/

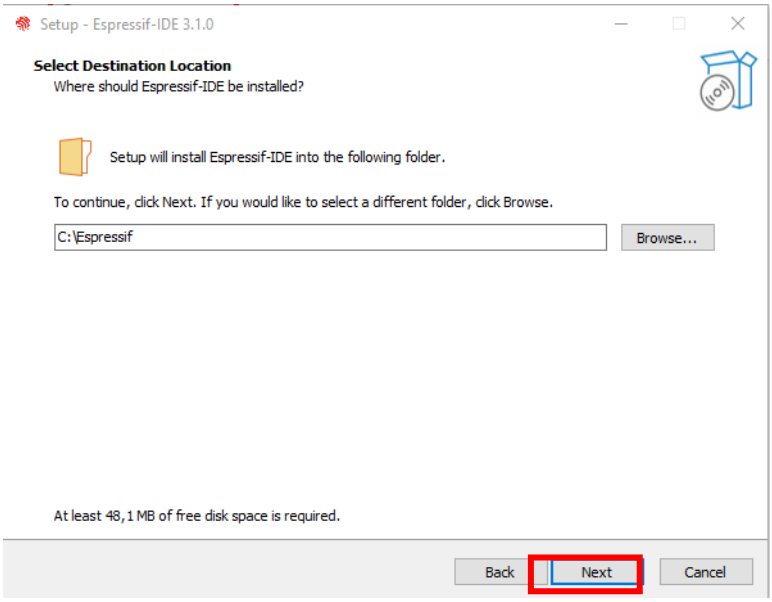


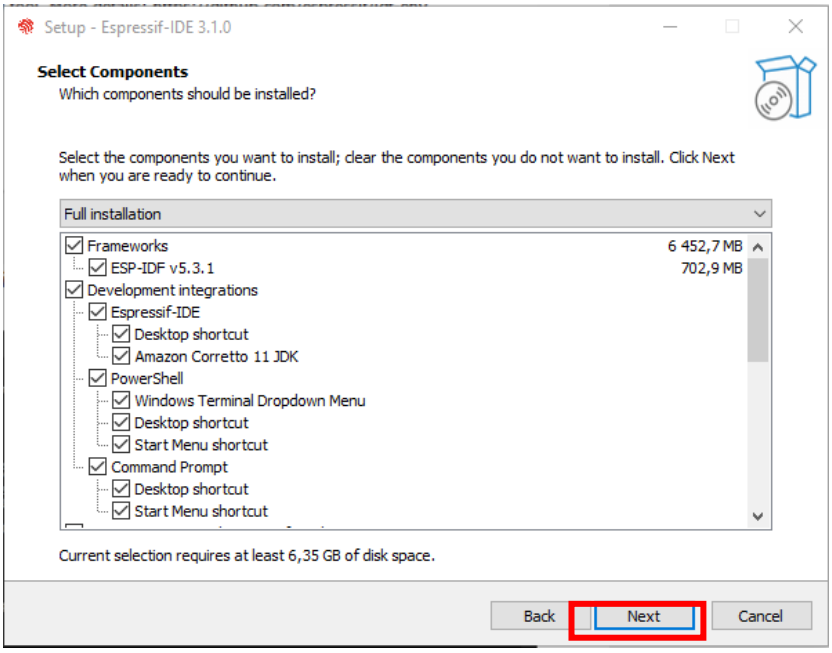
2. Configuration de l'installation

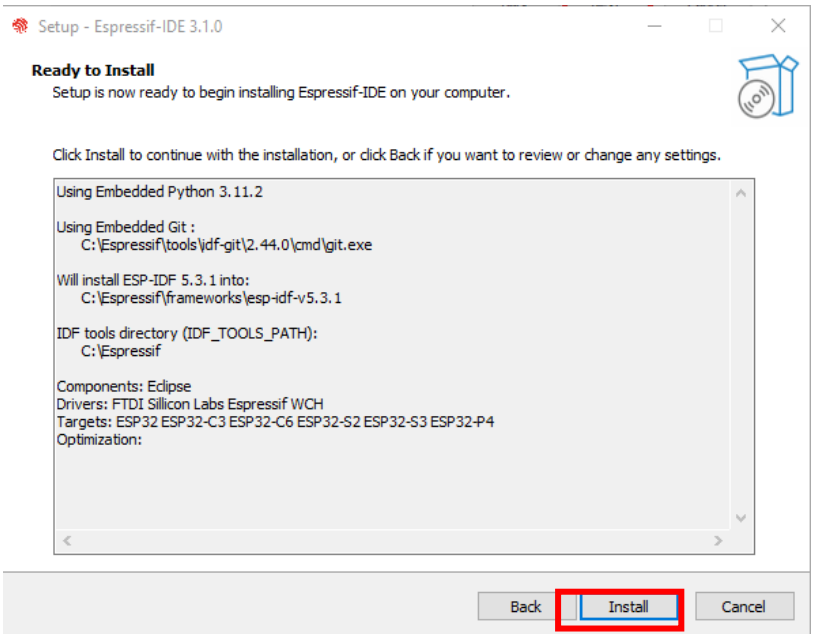


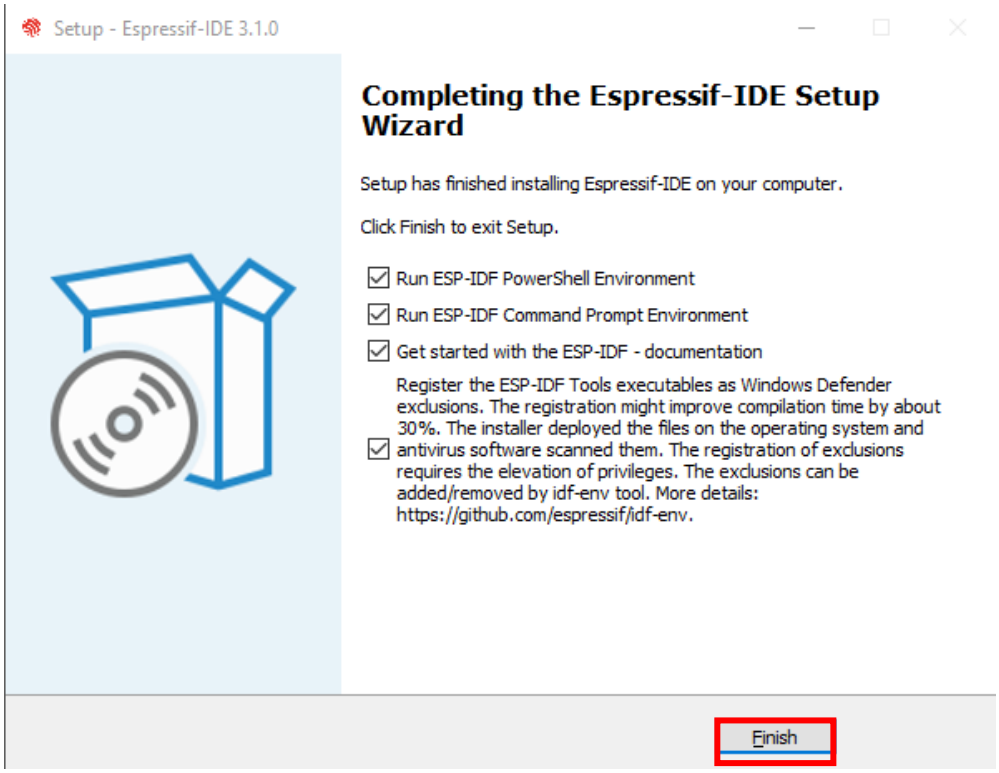


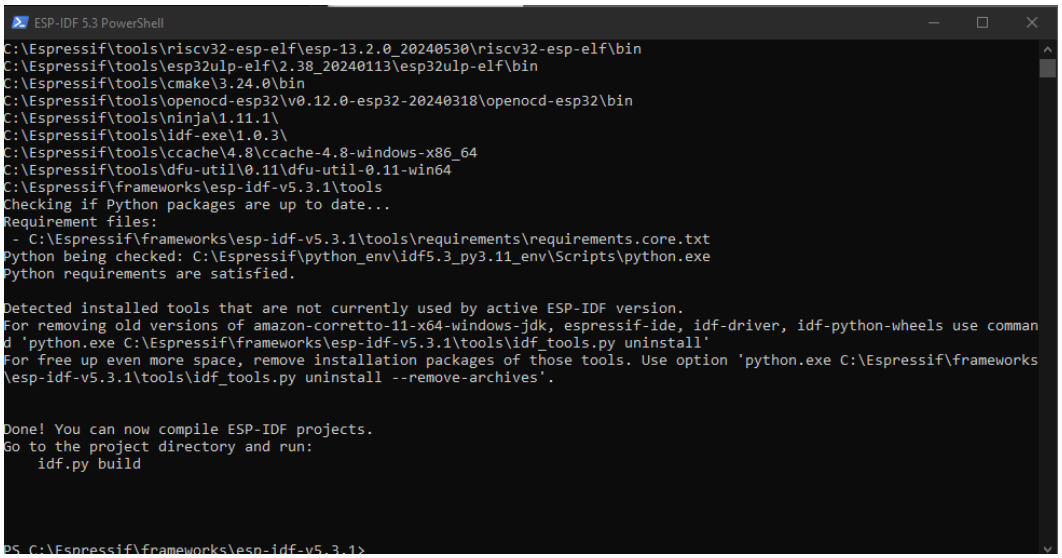


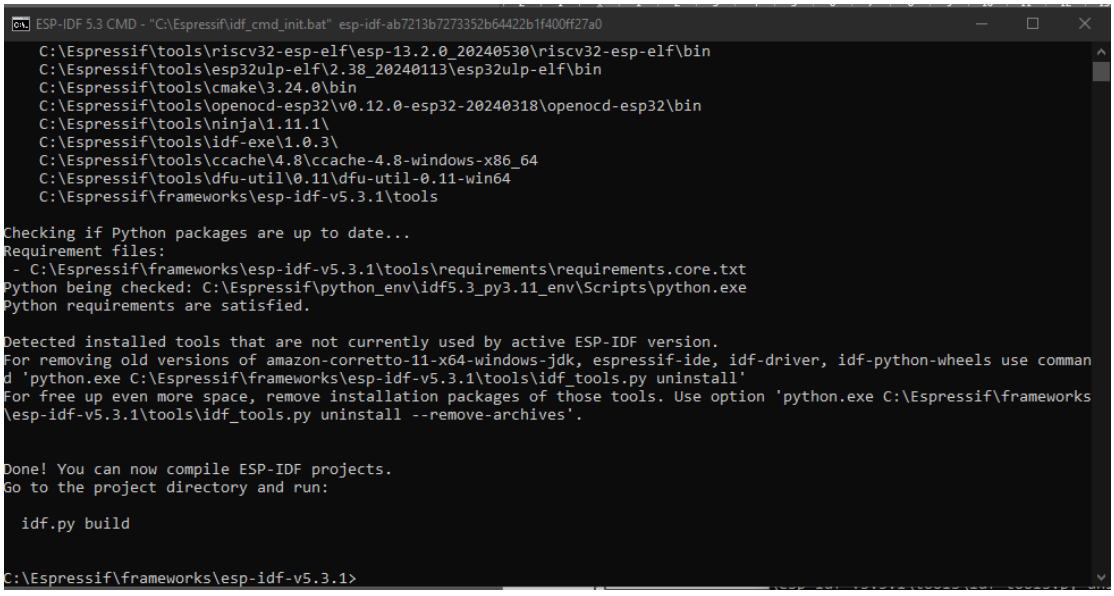

Setup - Espressif-IDE 3.1.0
Select Destination Location
Where should Espressif-IDE be installed?
Setup will install Espressif-IDE into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Espressif
Browse...
At least 48,1 MB of free disk space is required.
Back
Next
Cancel


Setup - Espressif-IDE 3.1.0
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Full installation
Frameworks 6 452,7 MB
ESP-IDF v5.3.1 702,9 MB
Development integrations
Espressif-IDE
Desktop shortcut
Amazon Corretto 11 JDK
PowerShell
Windows Terminal Dropdown Menu
Desktop shortcut
Start Menu shortcut
Command Prompt
Desktop shortcut
Start Menu shortcut
Current selection requires at least 6,35 GB of disk space.
Back
Next
Cancel

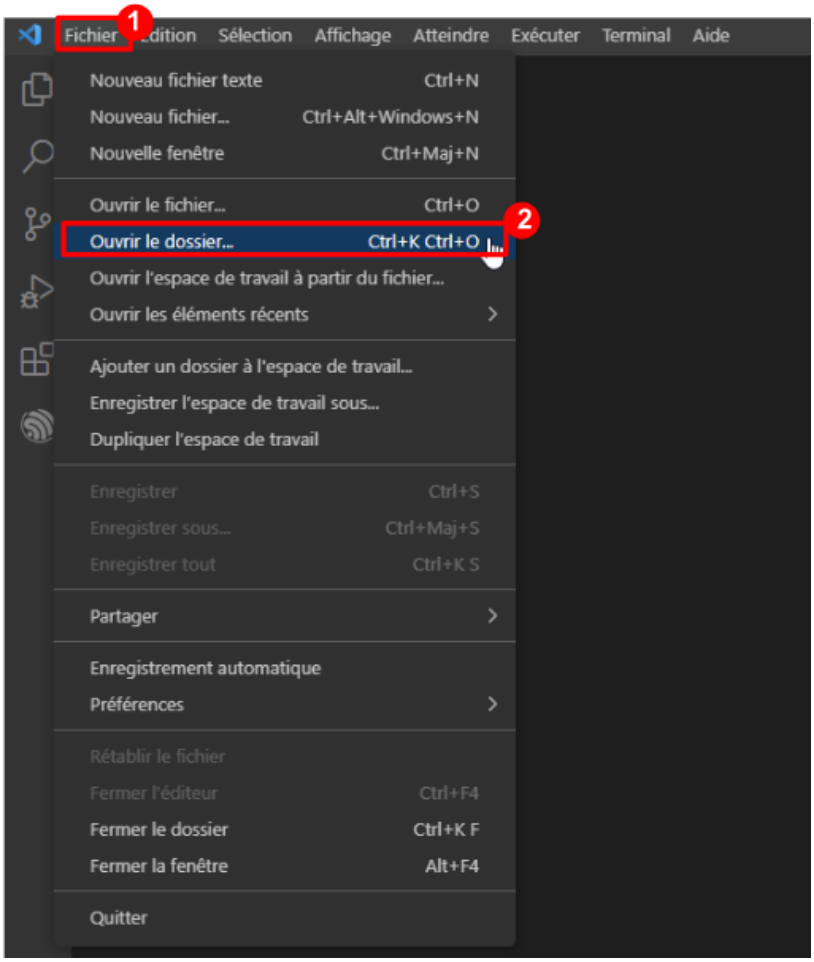

Setup - Espressif-IDE 3.1.0
Ready to Install
Setup is now ready to begin installing Espressif-IDE on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Using Embedded Python 3.11.2
Using Embedded Git :
C:\Espressif\tools\idf-git\2.44.0\cmd\git.exe
Will install ESP-IDF 5.3.1 into:
C:\Espressif\frameworks\esp-idf-v5.3.1
IDF tools directory (IDF_TOOLS_PATH):
C:\Espressif
Components: Eclipse
Drivers: FTDI Sillicon Labs Espressif WCH
Targets: ESP32 ESP32-C3 ESP32-C6 ESP32-S2 ESP32-S3 ESP32-P4
Optimization:
Back
Install
Cancel

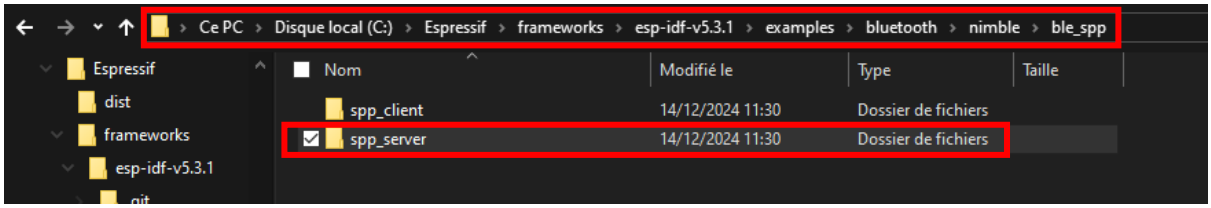

Setup - Espressif-IDE 3.1.0
Completing the Espressif-IDE Setup Wizard
Setup has finished installing Espressif-IDE on your computer.
Click Finish to exit Setup.
Run ESP-IDF PowerShell Environment
Run ESP-IDF Command Prompt Environment
Get started with the ESP-IDF - documentation
Register the ESP-IDF Tools executables as Windows Defender exclusions. The registration might improve compilation time by about 30%. The installer deployed the files on the operating system and antivirus software scanned them. The registration of exclusions requires the elevation of privileges. The exclusions can be added/removed by idf-env tool. More details: https://github.com/espressif/idf-env.
Finish


ESP-IDF 5.3 PowerShell
C:\Espressif\tools\riscv32-esp-elf\esp-13.2.0_20240530\riscv32-esp-elf\bin
C:\Espressif\tools\esp32ulp-elf\2.38_20240113\esp32ulp-elf\bin
C:\Espressif\tools\cmake\3.24.0\bin
C:\Espressif\tools\openocd-esp32\v0.12.0-esp32-20240318\openocd-esp32\bin
C:\Espressif\tools\ninja\1.11.1\
C:\Espressif\tools\idf-exe\1.0.3\
C:\Espressif\tools\ccache\4.8\ccache-4.8-windows-x86_64
C:\Espressif\tools\dfu-util\0.11\dfu-util-0.11-win64
C:\Espressif\frameworks\esp-idf-v5.3.1\tools
Checking if Python packages are up to date...
Requirement files:
- C:\Espressif\frameworks\esp-idf-v5.3.1\tools\requirements\requirements.core.txt
Python being checked: C:\Espressif\python_env\idf5.3_py3.11_env\Scripts\python.exe
Python requirements are satisfied.
Detected installed tools that are not currently used by active ESP-IDF version.
For removing old versions of amazon-corretto-11-x64-windows-jdk, espressif-ide, idf-driver, idf-python-wheels use comman
d 'python.exe C:\Espressif\frameworks\esp-idf-v5.3.1\tools\idf_tools.py uninstall'
For free up even more space, remove installation packages of those tools. Use option 'python.exe C:\Espressif\frameworks
\esp-idf-v5.3.1\tools\idf_tools.py uninstall --remove-archives'.
Done! You can now compile ESP-IDF projects.
Go to the project directory and run:
idf.py build
PS C:\Espressif\frameworks\esp-idf-v5.3.1>


ESP-IDF 5.3 CMD - "C:\Espressif\idf_cmd_init.bat" esp-idf-ab7213b7273352b64422b1f400ff27a0
C:\Espressif\tools\riscv32-esp-elf\esp-13.2.0_20240530\riscv32-esp-elf\bin
C:\Espressif\tools\esp32ulp-elf\2.38_20240113\esp32ulp-elf\bin
C:\Espressif\tools\cmake\3.24.0\bin
C:\Espressif\tools\openocd-esp32\v0.12.0-esp32-20240318\openocd-esp32\bin
C:\Espressif\tools\ninja\1.11.1\
C:\Espressif\tools\idf-exe\1.0.3\
C:\Espressif\tools\ccache\4.8\ccache-4.8-windows-x86_64
C:\Espressif\tools\dfu-util\0.11\dfu-util-0.11-win64
C:\Espressif\frameworks\esp-idf-v5.3.1\tools
Checking if Python packages are up to date...
Requirement files:
- C:\Espressif\frameworks\esp-idf-v5.3.1\tools\requirements\requirements.core.txt
Python being checked: C:\Espressif\python_env\idf5.3_py3.11_env\Scripts\python.exe
Python requirements are satisfied.
Detected installed tools that are not currently used by active ESP-IDF version.
For removing old versions of amazon-corretto-11-x64-windows-jdk, espressif-ide, idf-driver, idf-python-wheels use comman
d 'python.exe C:\Espressif\frameworks\esp-idf-v5.3.1\tools\idf_tools.py uninstall'
For free up even more space, remove installation packages of those tools. Use option 'python.exe C:\Espressif\frameworks
\esp-idf-v5.3.1\tools\idf_tools.py uninstall --remove-archives'.
Done! You can now compile ESP-IDF projects.
Go to the project directory and run:
idf.py build
C:\Espressif\frameworks\esp-idf-v5.3.1>

3. Ouverture d'un exemple



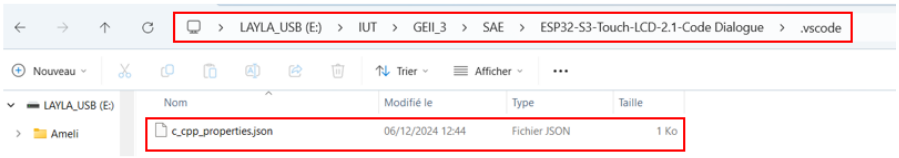


4. Modification du fichier

Lors de l'ouverture d'un exemple, des erreurs sur les #include apparaissent car le chemin pour les trouver est inconnu pour le compilateur. Il faut donc lui indiquer les chemins.

```
7   #include "esp_log.h"
8   #include "nvs_flash.h"
9   /* BLE */
10  #include "nimble/nimble_port.h"
11  #include "nimble/nimble_port_freertos.h"
12  #include "host/ble_hs.h"
13  #include "host/util/util.h"
14  #include "console/console.h"
15  #include "services/gap/ble_svc_gap.h"
16  #include "services/gatt/ble_svc_gatt.h"
17  #include "ble_spp_server.h"
18  #include "driver/uart.h"
```

Pour pallier ce problème, il faut modifier le fichier c_cpp_properties.json avec ce code :




```
{
  "configurations": [
    {
      "name": "ESP-IDF",
      "compilerPath": null,
      "compileCommands": "${workspaceFolder}/build/compile_commands.json",
      "includePath": [
        "${config:idf.espIdfPath}/components/**",
        "${config:idf.espIdfPathWin}/components/**",
        "${config:idf.espAdfPath}/components/**",
        "${config:idf.espAdfPathWin}/components/**",
        "${workspaceFolder}/**",
        "C:/Users/.../esp/v5.3.1/esp-idf/components/freertos/FreeRTOS-Kernel-SMP/include",
        "C:/Espressif/frameworks/esp-idf-v5.3.1/components/bt/host/bluedroid/api/include/api",
        "${workspaceFolder}/main/Wireless"
      ],
      "browse": {
        "path": [
          "${config:idf.espIdfPath}/components",
          "${config:idf.espIdfPathWin}/components",
          "${config:idf.espAdfPath}/components/**",
          "${config:idf.espAdfPathWin}/components/**",
          "${workspaceFolder}"
        ],
        "limitSymbolsToIncludedHeaders": false
      }
    }
  ],
  "version": 4
}
```

Attention : penser à modifier les "..." du premier chemin C:/ en fonction de son nom sur le PC.

C. La librairie *lvgl.h*

La librairie lvgl est une librairie open source pour créer des interfaces graphiques. En effet, elle permet d'écrire du texte, de créer des formes ou encore de créer des curseurs.

1. Afficher du texte

```
// Label du titre
lv_obj_t *title = lv_label_create(scr);
lv_label_set_text(title, "Controle du Volume");
lv_obj_align(title, LV_ALIGN_CENTER, 0, -200);
```

Le principe pour écrire du texte est de créer des labels. C'est à dire que chaque ligne de texte a un nom. Par exemple ici, la ligne "Contrôle du Volume" a pour nom "*title*". Enfin, plusieurs fonctions peuvent être utilisées afin de modifier la ligne. On peut par exemple modifier l'alignement du texte en alignant au centre, en haut, ou encore en bas.

2. Création de formes

Grâce à cette librairie, on peut aussi dessiner de nombreuses formes, que ce soient des cercles, des carrés, des rectangles ou encore des rectangles aux angles arrondis. Pour cela, le fonctionnement est le même que pour le texte, on crée un label pour une forme.

```
lv_obj_t * rec = lv_obj_create(scr);
lv_obj_set_size(rec, 150, 50);  // Taille du rect (largeur et hauteur)
lv_obj_set_pos(rec, 165, 55);   // Position du rect
lv_obj_set_style_radius(rec, 0,LV_STATE_DEFAULT);  //  => contour X arrondi

lv_obj_t * circle = lv_obj_create(scr);
lv_obj_set_size(circle, 100, 100);  // Taille du cercle (largeur et hauteur égales)
lv_obj_set_pos(circle, 130, 150);   // Position du cercle
lv_obj_set_style_radius(circle, LV_RADIUS_CIRCLE, LV_STATE_DEFAULT);

lv_obj_t * carre = lv_obj_create(scr);
lv_obj_set_size(carre, 50, 50);  // Taille du rect (largeur et hauteur égales)
lv_obj_set_pos(carre, 250, 150);   // Position du rect
lv_obj_set_style_radius(carre, 0,LV_STATE_DEFAULT);  //  => contour X arrondi
```

Par exemple, ici, on trace un rectangle, un cercle ainsi qu'un carré. On les positionne à des endroits différents, et on les dimensionne à notre guise. On obtient donc ceci :

Image de l’écran avec le rectangle, le cercle et le carré.

D. Le mode BLE

Le Bluetooth à Faible Consommation d'Énergie (Bluetooth Low Energy ou BLE) désigne un type spécifique de technologie Bluetooth à faible consommation. Le Bluetooth Low Energy a été créé pour pouvoir transmettre des données entre deux appareils sur de longues périodes, ce qui permet d'économiser au maximum la batterie. Ainsi, un appareil BLE fonctionne, presque toujours, en mode veille, s'activant uniquement lorsqu'il établit des connexions qui ne durent que quelques millisecondes. Il est présent sur de nombreux appareils utilisez en sur des applications médicales ou encore de sécurité. Mais, on peut aussi le retrouver sur les ESP32, et notamment sur ESP32-S3 présent avec notre écran.

L'utilisation de ce mode BLE les protocoles GAP et GATT. GAP (Generic Access Profile) contrôle la connexion. C'est à dire qu'il rend visible l'ESP32 sur notre téléphone. Il détermine également si celui-ci peut interagir avec des appareils. Le protocole GATT (Generic ATTribute) permet de gérer la communication entre les deux appareils (Client-Serveur).

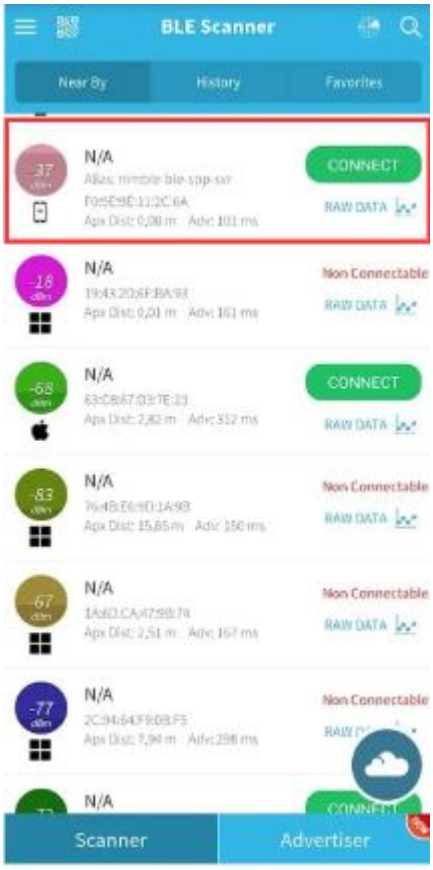
Afin de pouvoir interagir avec l'ESP32 en BLE, et grâce aux protocoles GAP et GATT, on doit utiliser une application comme BLE_Scanner. Elle permet d'effectuer un scan des périphériques BLE à proximité, afficher leurs informations principales, et interagir avec eux pour explorer leurs services et caractéristiques.

Logo application BLE_Scanner

Pour réaliser tout cela, on commence par activer le protocole GAP afin d'observer notre ESP32 depuis l'application.

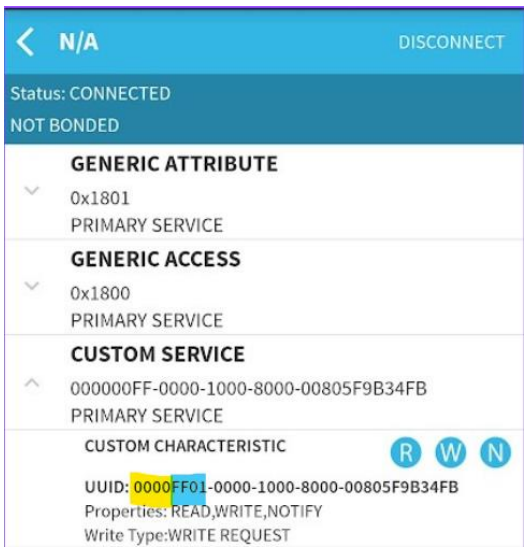
```
case ESP_GAP_BLE_ADV_START_COMPLETE_EVT:
    ESP_LOGI(TAG, "Connexion démarré");
    break;
```

Un fois que le mode BLE a été activé, on observe donc l'ESP32 sur le téléphone et on peut s'y connecter.



Enfin, il faut activer le protocole GAPP pour pouvoir communiquer entre le téléphone et l'ESP. Pour cela dans un premier temps, il faut indiquer dans notre code les valeurs de caractéristiques du produit :

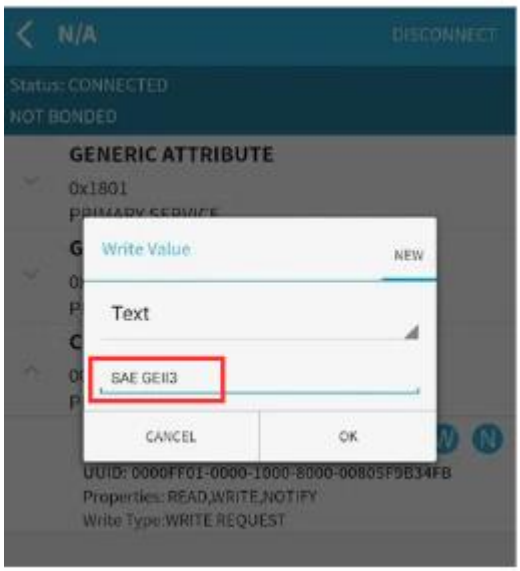
```
#define GATT_SERVICE_UUID           0x0000
#define GATT_CHARACTERISTIC_UUID    0xFF01
```



Ensuite on peut activer le protocole GATT et récupérer des données écrites sur le téléphone pour les afficher sur l'écran :

```
case ESP_GATTS_WRITE_EVT:
    ESP_LOGI(TAG, "Données reçues: handle=%d, valeur=%.*s",
             param->write.handle, param->write.value); // Récupération et écriture de la valeur (handle =identifiant de la variable)
    valeur = param->write.value;
    lv_obj_t *scr = lv_scr_act();  // Obtenir l'écran actif
    // Label du titre
    lv_obj_t *title1 = lv_label_create(scr);
    lv_obj_t *title2 = lv_label_create(scr);
    lv_label_set_text(title1,"Donnees recues:" );
    lv_label_set_text(title2, valeur); //Ecris la valeur sur le LCD
    lv_obj_align(title1, LV_ALIGN_CENTER, -70, 0);
    lv_obj_align(title2, LV_ALIGN_CENTER, 50, 0);

    break;
```



V. Conclusion et perspectives

Ce projet a permis de développer une compréhension approfondie des technologies associées à l'utilisation d'écrans LCD et de l'ESP32. Grâce à l'intégration des bibliothèques graphiques comme *Adafruit_GFX* et *lvgl.h*, ainsi qu'à la gestion tactile avec le capteur *CST816S*, nous avons pu créer une interface utilisateur simple mais fonctionnelle. Le travail a également mis en lumière l'importance des protocoles de communication, notamment SPI, I²C, GAP et GATT, essentiels pour l'affichage et le dialogue Bluetooth.

D'un point de vue technique, l'organisation du code a permis de structurer efficacement les fonctionnalités telles que :

- Gestion des pages : Menu, Color, Info.
- Gestion des boutons interactifs : Création de zones tactiles précises.
- Navigation intuitive : Interaction fluide entre l'écran et l'utilisateur.

En travaillant avec l'ESP32 et les écrans tactiles, nous avons également exploré les possibilités offertes par le Bluetooth Low Energy (BLE), ce qui ouvre des portes pour des applications plus connectées et interactives.

Les pistes d'amélioration identifiées pour ce projet incluent :

1. Fonctionnalités avancées :
    a. Intégrer une horloge en temps réel (RTC) pour afficher l'heure sur l'écran.
    b. Ajouter des paramètres utilisateur pour personnaliser l'interface (couleurs, langue, etc.).
    c. Vérifier le fonctionnement des coordonnées données pour l'emplacement des boutons.
2. Améliorations graphiques :
    a. Utiliser davantage la bibliothèque *lvgl.h* pour enrichir l'interface avec des animations, des transitions fluides, et un style plus moderne.
3. Connectivité et interaction :
    a. Implémenter AVRCP (Audio/Video Remote Control Profile) afin de contrôler des fonctions multimédia sur un appareil connecté (par exemple : ajustement du volume via l'écran tactile).
    b. Ajouter des fonctionnalités interactives via le BLE, comme le contrôle d'autres appareils connectés.

VI. Résumé

Layla Queyssac et Adem Dribek, étudiants en 3e année de BUT GEII à l'IUT d'Angers, ont développé un projet axé sur l'utilisation d'écrans LCD tactiles. Le projet s'est articulé autour de deux axes principaux : l'intégration d'un écran seul et son utilisation en combinaison avec un microcontrôleur ESP32. Ils ont conçu une interface graphique simple pour permettre la navigation entre plusieurs pages interactives et ont utilisé des bibliothèques spécifiques pour gérer l'affichage et le tactile.

Avec l'ESP32, ils ont implémenté une communication Bluetooth entre l'écran et un téléphone, en exploitant des protocoles comme GAP et GATT. Ce travail leur a permis de maîtriser des protocoles de communication et de découvrir des outils graphiques avancés, tout en ouvrant des perspectives pour des fonctionnalités supplémentaires comme des animations ou le contrôle du volume via l'écran.

VII. Summary

Layla Queyssac and Adem Dribek, third-year students in the GEII program at IUT Angers, developed a project focusing on the use of touchscreen LCDs. The project revolved around two main aspects: integrating a standalone screen and using it with an ESP32 microcontroller. They designed a simple graphical interface to navigate through interactive pages and utilized specific libraries to manage display and touch functionalities.

With the ESP32, they implemented Bluetooth communication between the screen and a phone, leveraging protocols such as GAP and GATT. This project allowed them to master communication protocols and explore advanced graphical tools while paving the way for additional features like animations or volume control via the screen.