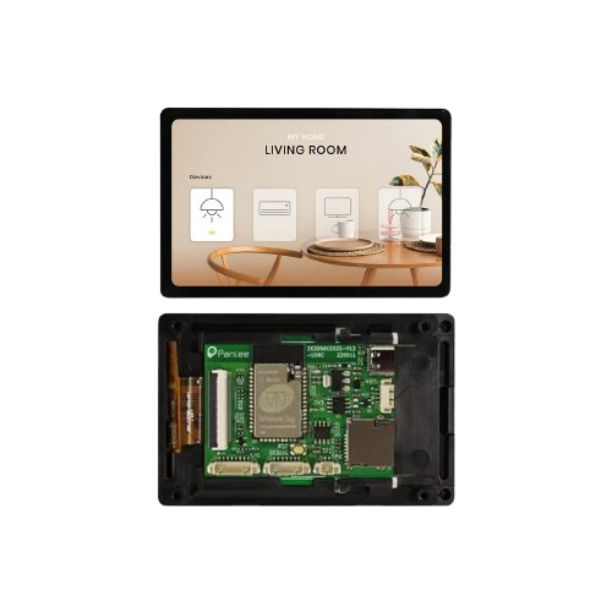
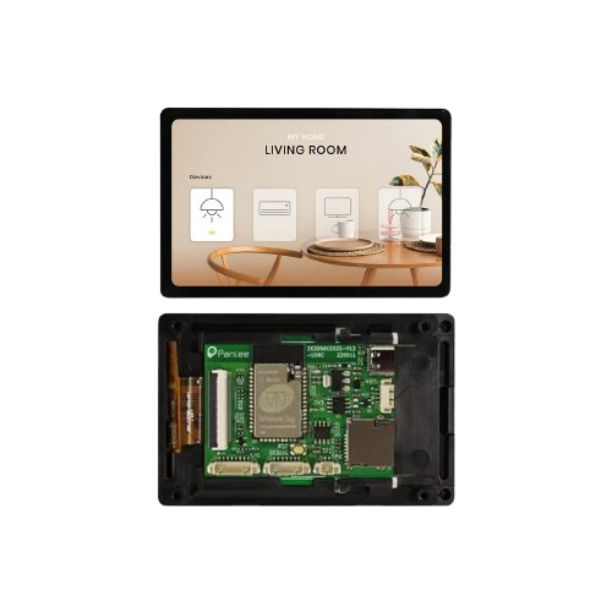


Nathanaël DEBREILLY – Eliez LE HUNSEC – Ilan CHANY

Nathanaël DEBREILLY - Ilan CHANY - Hugo Messager



Projet WT32 :

Tutoriel de prise en main

L2 PEI - Projet

Stéphane BOCHARD

Table des matières

[Introduction 1](#_Toc197262755)

[1. Installation des outils 2](#_Toc197262756)

[1.1. Installation de Arduino IDE 2](#_Toc197262757)

[1.2. Configuration de Arduino IDE 4](#_Toc197262758)

[2. Programmation 7](#_Toc197262759)

[2.1. Utilisation du moniteur série 7](#_Toc197262760)

[2.2. Configuration des pins 9](#_Toc197262761)

[2.3. Programme permettant de paramétrer la bibliothèque 10](#_Toc197262762)

[2.4. Programme principal 11](#_Toc197262763)

[2.5. Commandes de bases pour LovyanGFX 13](#_Toc197262764)

# Introduction

Dans ce tutoriel, nous détaillerons toutes les étapes nécessaires à la programmation d’une carte [WT32-SC01 Plus](https://eu.robotshop.com/fr/products/carte-developpement-wt32-sc01-plus-esp32-ecran-tactile-lcd-ips-35-pouces) à l’aide de Arduino IDE. Nous verrons quels outils sont requis et comment les configurer ainsi que la réalisation d’un programme de base.

Ce tutoriel à une durée estimée à environ 30 minutes et va nécessiter un minimum 1 Go d’espace mémoire, 500 Mo pour le logiciel Arduino IDE et environ 250 Mo pour les outils et configurations.

# Installation des outils

Dans cette première partie, nous verrons quels outils sont nécessaires à la programmation de la carte WT32-SC01 Plus et comment les installer et les configurer.

## Installation de Arduino IDE

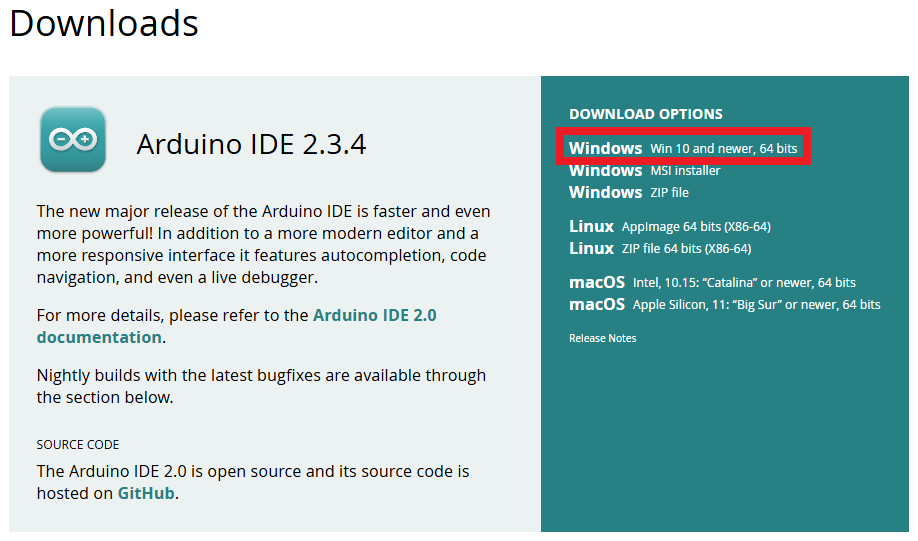
Il existe de nombreux logiciels adaptés à la programmation sur microcontrôleur, dans ce tutoriel, nous privilégierons Arduino IDE (version 2.3.4) qui est gratuit, facile d’accès et intuitif. Commençons donc par installer le logiciel, rendez-vous pour cela sur le [site officiel de Arduino](https://www.arduino.cc/en/software) où vous pourrez le télécharger en suivant les étapes suivantes (nous utiliserons dans notre cas la version Windows 10 - 64 bits) :

Figure 1 : Site Arduino

Vous pouvez alors exécuter le fichier « arduino-ide\_2.3.4\_Windows\_64bits.exe » que vous venez de télécharger.

On arrive alors aux fenêtres suivantes qui vont permettre l’installation du logiciel :

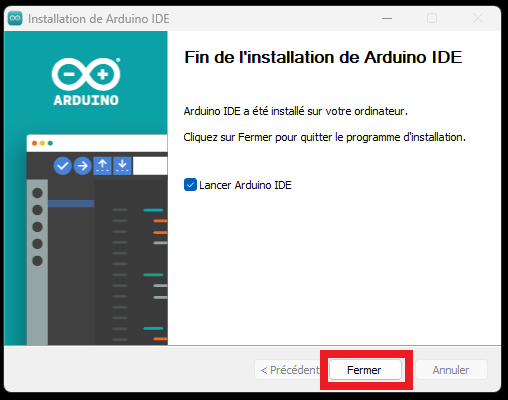
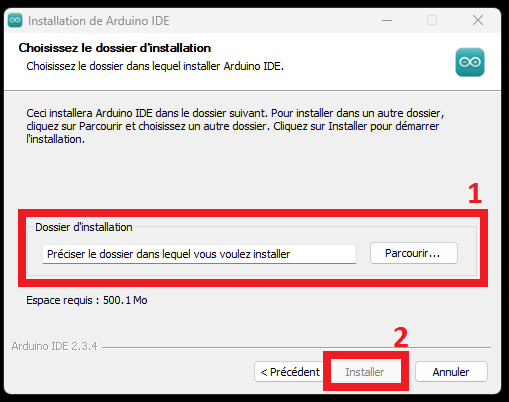
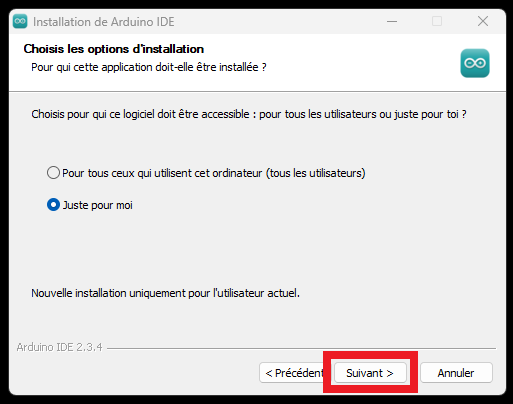
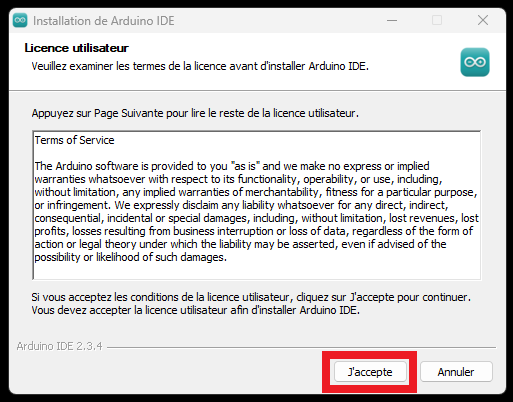


Figure 2 : Installation de Arduino IDE

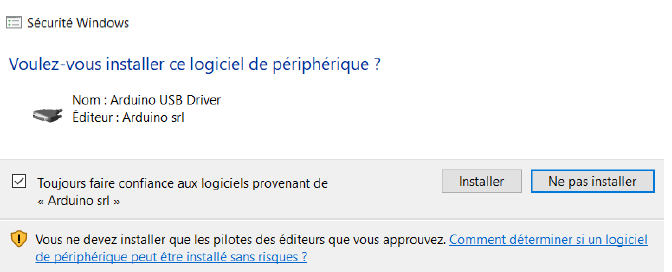
En fonction de votre ordinateur et des driver qui sont déjà installés, vous pouvez avoir des fenêtres supplémentaires pour l’installation de driver complémentaires, leur installation vous sera proposée directement et il suffira d’appuyer sur installer :

Figure 3 : Installation des drivers complémentaires

## Configuration de Arduino IDE

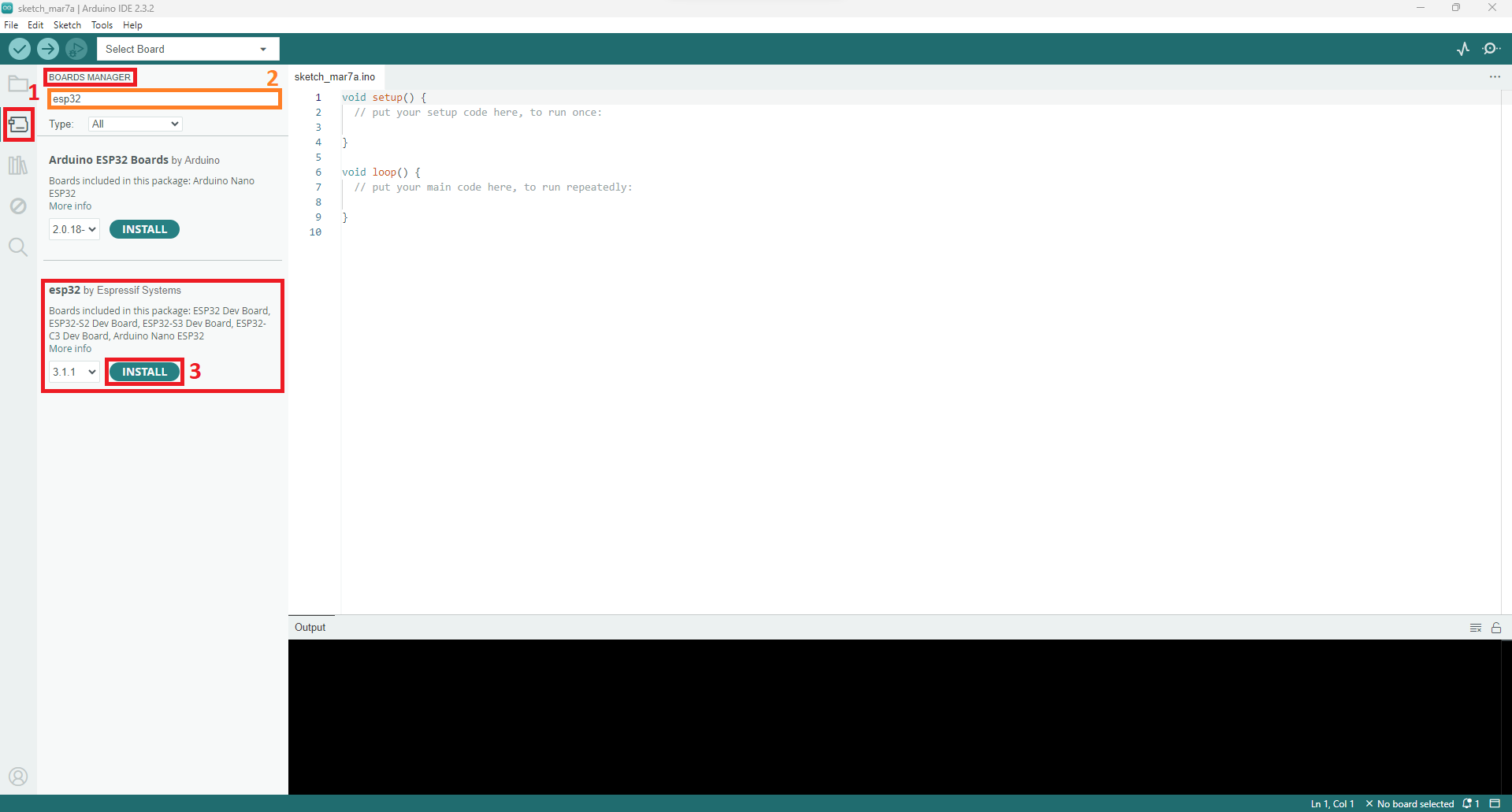
Maintenant que le logiciel est correctement installé, il nous faut le configurer en fonction de la carte que nous souhaitons programmer, dans notre cas une WT32-SC01 Plus qui est une carte utilisant un [microprocesseur ESP32](https://fr.wikipedia.org/wiki/ESP32). Il nous faut donc installer l’extension correspondante aux ESP32, pour cela, entrez dans le Boards manager puis recherchez « esp32 », il vous suffit alors de télécharger « esp32 by Espressif Systems », nous utilisons la version 3.1.1 mais nous conseillons d’utiliser la dernière version comme indiqué ci-dessous :

Figure 4 : Configuration du Boards Manager

Attention, cette étape peut être un peu longue en fonction de votre connexion internet, attendez bien le message de fin d’installation, dont voici un exemple :

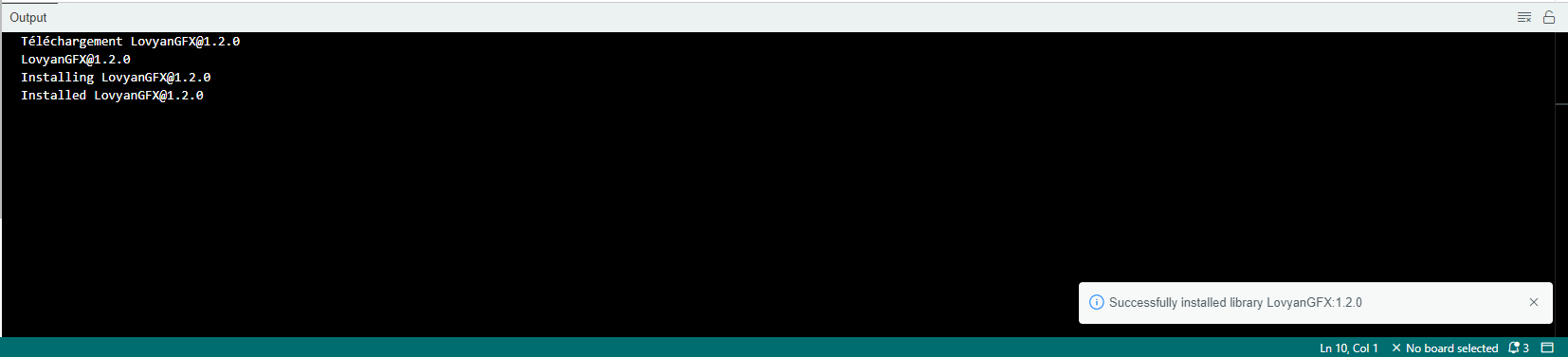
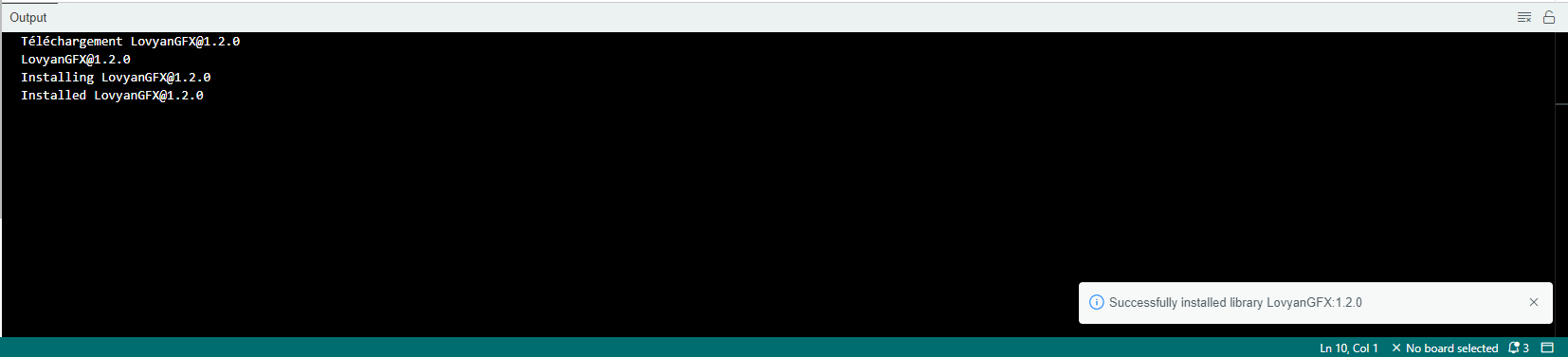


Figure 5 : Message de fin d’installation

Maintenant que l’extension est correctement installée, nous pouvons configurer le port sur lequel nous allons brancher la carte, cela va permettre à Arduino IDE de la reconnaître et de correctement la traiter. Pour ce faire, commencez par brancher la carte à l’un des ports USB de votre ordinateur, à ce stade, la LED bleue devrait s’allumer. Il est possible qu’un programme de base soit déjà installé sur la carte, si ce n’est pas le cas l’écran restera simplement noir, d’où l’utilité d’observer la LED :



**Éteinte**

**Allumée**

Figure 6 : Branchement de la carte

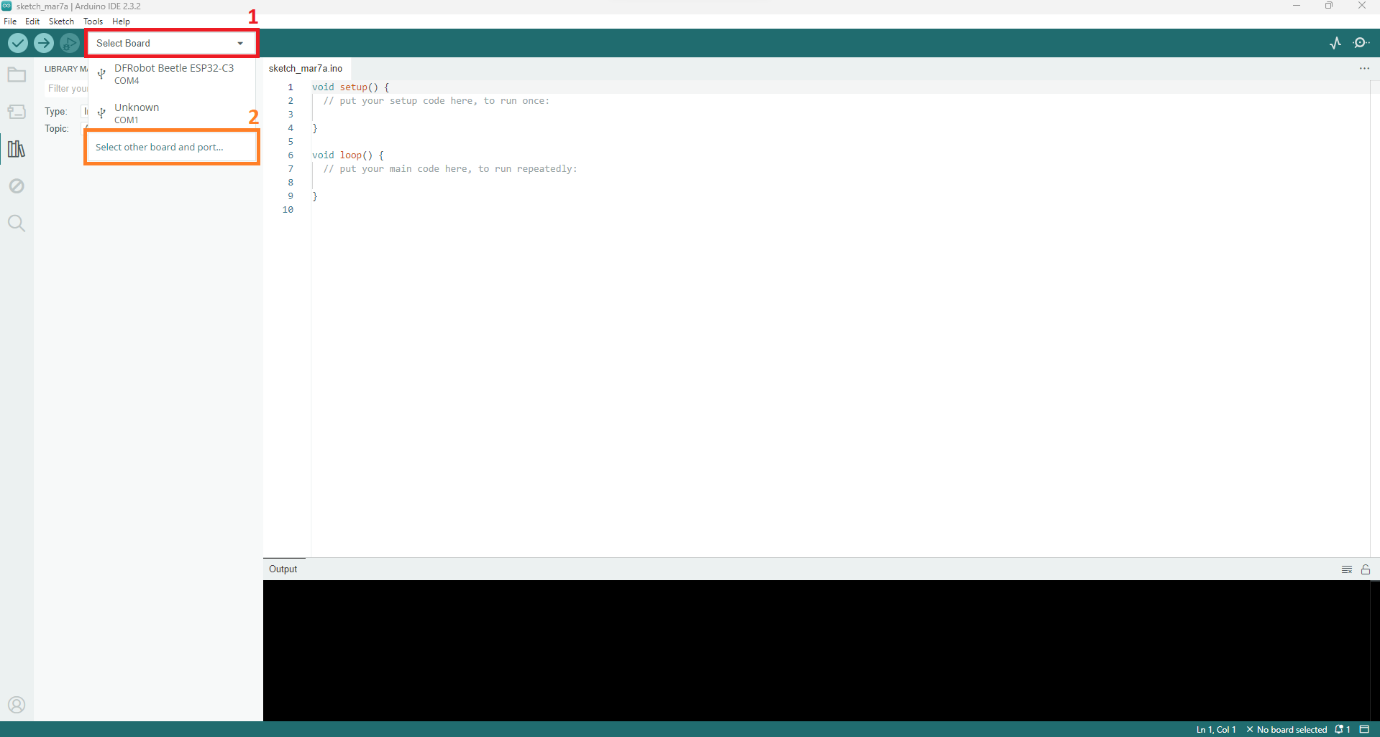
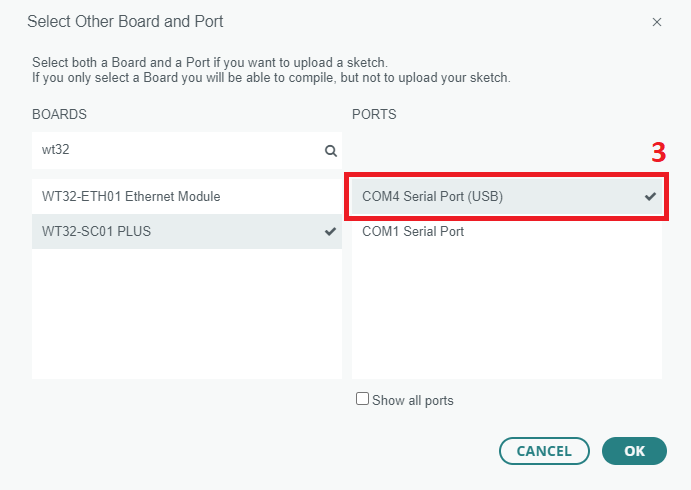
Nous pouvons maintenant revenir sur l’application et accéder à la configuration du port comme indiqué ci-dessous :

Figure 7 : Configuration du port USB

Une fois dans le menu, il suffit de trouver votre carte dans la barre de recherche, dans notre cas WT32-SC01 Plus puis de sélectionner le bon port. Pour déterminer le port sur lequel est branché la carte, vous pouvez la débrancher avant de la rebrancher pour voir apparaître le port concerné comme indiqué ci-dessous :



**Carte débranchée**

**Carte branchée**

Figure 8 : Configuration du port USB suite

Vous pouvez alors valider en appuyant sur « OK » et vous devriez voir apparaître le nom de la carte dans la sélection du port.

Pour finir, il nous reste simplement à installer la bibliothèque permettant d’exploiter l’écran de notre carte, encore une fois, il en existe plusieurs possibles (AdafruitGFX, LovyanGFX, TFT\_eSPI, Arduino\_GFX) mais nous avons décidé d’utiliser LovyanGFX car elle supporte l’écran du WT32-SC01 Plus et elle permet une bonne gestion des images, couleurs et tactile bien qu’elle soit un peu complexe à configurer.

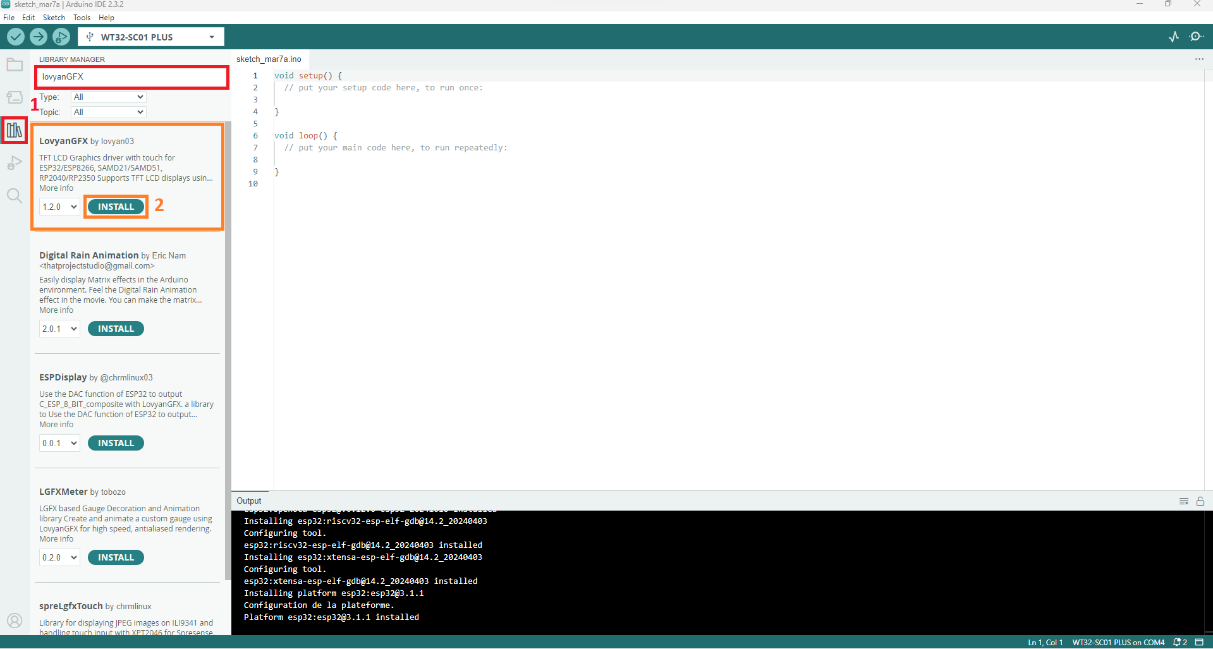
Pour installer LovyanGFX, il vous suffit de vous rendre dans le « Library Manager » où vous pourrez rechercher « lovyanGFX » avant d’installer LovyanGFX by lovyan03. Nous utilisons la version 1.2.0 pour nos programmes, mais une version plus récente devrait également fonctionner :

Figure 9 : installation de la bibliothèque LovyanGFX

Bien sûr, il existe de nombreuses autres bibliothèques qui pourraient être utiles pour notre carte comme Wifi.h pour la connexion wifi, mais dans ce tutoriel nous nous concentrerons sur un affichage basique d’une information sur l’écran.

# Programmation

## Utilisation du moniteur série

Commençons par vérifier que nous avons bien configuré Arduino IDE, pour ce faire, nous pouvons utiliser le moniteur série. Il s’agit d’un outil directement intégré dans l'IDE Arduino (ou d'autres environnements) qui permet d’échanger des messages texte entre carte et ordinateur via le port USB. Ce procédé est très pratique pour vérifier la bonne exécution d’un programme, on pourrait le comparer à un print dans d’autres langages.

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Le contenu généré par l’IA peut être incorrect.Ici, nous l’utiliserons pour vérifier que la carte est bien reconnue et que l’on peut communiquer avec. Commençons par ouvrir le moniteur série qui apparaîtra comme un onglet avec la console :

Figure 10 : Ouverture du moniteur série

Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.Vous devriez alors avoir un nouvel onglet à côté de la console (sortie) appelé moniteur série :

Figure 11: Onglet moniteur série

Nous pouvons observer un message en orange, ce dernier apparaît lorsque la carte est débranchée. Une fois la carte branchée, le message devrait disparaître.

À droite de l’onglet moniteur série, nous avons plusieurs options :

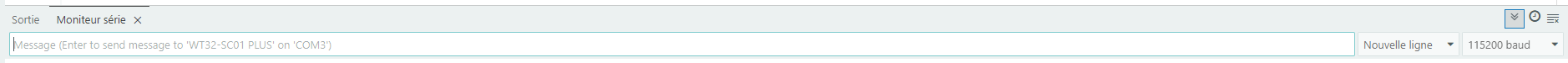
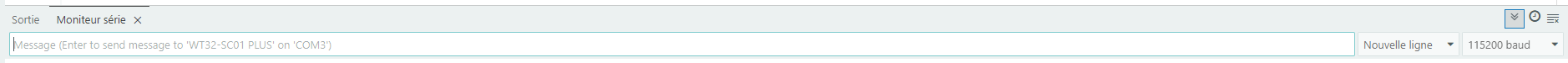


Figure 12 : Options du moniteur série

L’option qui va nous intéresser ici est le menu déroulé 115 200 baud. Il s’agit du débit auquel la communication série va s’effectuer (ici 115 200 bits par seconde). Ce paramètre devrait être sur cette valeur par défaut, si ce n’est pas le cas, vous pouvez la modifier.

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.Nous pouvons maintenant réaliser un programme d’exemple simple qui va renvoyer des messages au moniteur série. Dans Arduino IDE, le programme est divisé en deux parties, le setup ou initialisation dont le contenu ne sera exécuté qu’une seule fois au lancement du programme et la loop ou boucle dont le contenu sera exécuté en boucle, nous reviendrons plus en détails sur ce sujet dans la partie 2.3 :

Figure 13 : Programme basique pour le moniteur série

Comme dit plus tôt, le programme est divisé en deux parties, dans le setup, nous démarrons la communication avec le moniteur série avec *Serial.begin()* en précisant la valeur 115 200 correspondant à l’option vu précédemment.

Nous pouvons alors réaliser un *Serial.print()* pour afficher une chaîne de caractère dans le moniteur série (la précision ln après le print permet simplement un retour à la ligne après l’impression). On ajoute ensuite un délai d’une seconde avec *delay()* (équivalent de sleep ou wait).

Nous pouvons alors téléverser le programme vers la carte à l’aide de la flèche indiquée en jaune en faisant bien attention que le port usb soit bien indiqué comme étant celui de la WT32-SC01 Plus.

Une image contenant texte, capture d’écran, logiciel, Police

Le contenu généré par l’IA peut être incorrect.Une fois le téléversement terminé, nous obtenons alors le résultat suivant :

Figure 14 : Sortie du moniteur série

Nous avons bien le résultat escompté, la carte est donc bien en communication avec notre ordinateur. Nous pourrons ainsi réutiliser ce principe pour vérifier la bonne exécution d’un programme.

## Configuration des pins

Une image contenant texte, capture d’écran, nombre, Police

Le contenu généré par l’IA peut être incorrect.Maintenant que Arduino IDE est correctement installée, nous pouvons réaliser un programme basique permettant d’afficher un texte sur l’écran de la carte. Afin que notre programme puisse utiliser correctement l’écran de la carte, il nous faut d’abord déterminer les pins responsables de l’affichage, pour ce faire, il suffit de trouver leur numéro sur la [documentation de la carte](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwicsNrOw7uMAxVkV6QEHUk1AOAQFnoECB0QAQ&url=https%3A%2F%2Fwww.marutsu.co.jp%2Fcontents%2Fshop%2Fmarutsu%2Fdatasheet%2Fkhe_WT32-SC01-PLUS.pdf%3Fsrsltid%3DAfmBOors3LYLIEc584BJastMvIJpubMIoIMYdDiNEeolOjPPH_6afdv3&usg=AOvVaw3NTLLT8Zsy5ep-wt6Q2P4t&opi=89978449) dans la partie « interface description » aux pages 5 et 6. Voici dans notre cas les pins dont nous nous servirons pour l’affichage :

Figure 15 : Tableau des pins de l’affichage

Attention, ces numéros peuvent varier d’une carte à l’autre, il est donc nécessaire de vérifier directement dans la documentation.

## Programme permettant de paramétrer la bibliothèque

Une image contenant texte, Police, ligne, nombre

Le contenu généré par l’IA peut être incorrect.Passons maintenant à la partie programmation, certaines parties du programme resteront toujours les mêmes étant donné qu’il s’agit de l’initialisation de la bibliothèque LovyanGFX choisie précédemment. Dans un premier temps, commençons par ajouter la bibliothèque au programme et créons la classe LGFX qui nous servira à paramétrer l’écran :

Figure 16 : Ajout de la bibliothèque

Le panel représente l'écran physique que nous utilisons. Il contient toutes les informations spécifiques à l'écran, telles que la résolution, la profondeur de couleur, et les fonctions de contrôle de l'écran.

Le bus quant à lui, représente le moyen de communication entre le microcontrôleur et le panneau d'affichage. Il peut s'agir d'un bus parallèle, d'un bus SPI, d'un bus I2C, etc. Le bus est responsable de l'envoi et de la réception des données entre le microcontrôleur et le panneau.

Nous allons donc configurer le panel et le bus en fonction de notre carte (pour une autre carte, ces paramètres pourront être différents). Nous allons notamment préciser la fréquence d’écriture et les pins indispensables que nous avons vu précédemment pour le bus et la dimension de l’écran pour le panneau d’affichage :

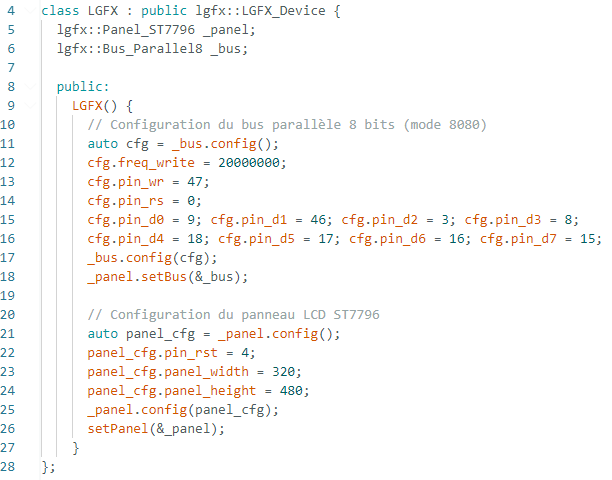


Figure 17 : Classe LGFX complétée avec pins et dimensions

Une image contenant texte, capture d’écran, Police, ligne

Le contenu généré par l’IA peut être incorrect.Une fois cette paramétrisation terminée, nous pouvons écrire le cœur de notre programme. La façon dont fonctionne Arduino IDE est simple, comme expliqué plus tôt, les programmes sont divisés en deux parties, l’initialisation (setup) et la boucle principale (loop). Ces deux parties composent le programme complet, l’initialisation n’est exécutée qu’une fois là où la boucle principale restera active tant que la carte sera alimentée :

Figure 18 : Différentes parties du programme

## Programme principal

Étant donné que nous souhaitons simplement afficher un texte, nous ne nous servirons pas de la boucle principale dans cet exemple, mais elle est très utile pour réaliser des programmes interactifs.

Afin d’afficher un texte, il nous faut d’abord commencer par activer le rétroéclairage (encadré en bleu). Une fois cela fait, nous pouvons paramétrer comme nous le souhaitons notre écran en changeant le format (paysage/portrait), le mode des couleurs (RGB) etc. (encadré en vert). Finalement, nous pouvons nous servir des commandes fournies par la bibliothèque LovyanGFX afin de mettre l’écran en blanc et d’afficher un texte surligné en jaune (encadré en rouge) :

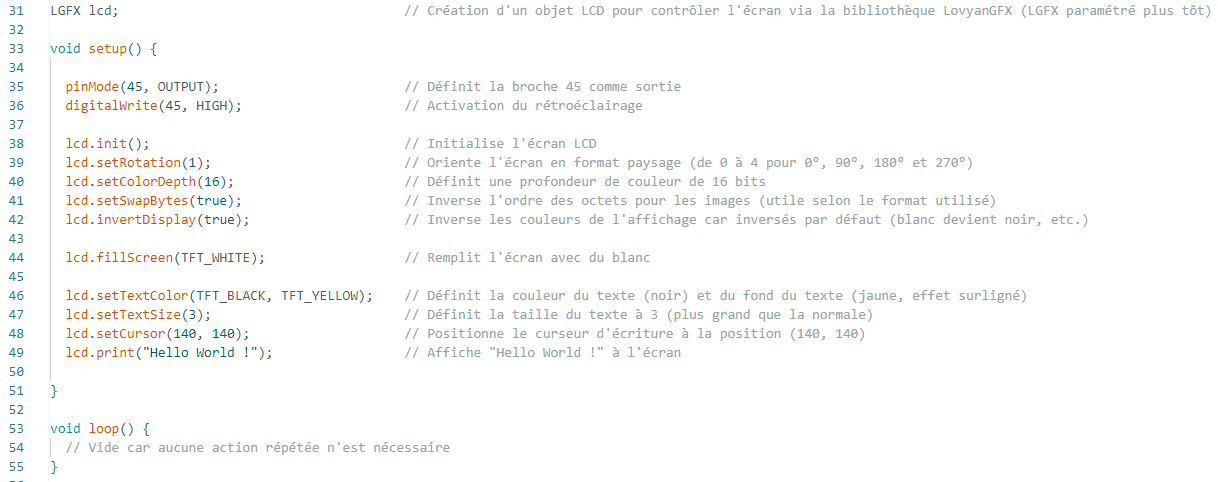
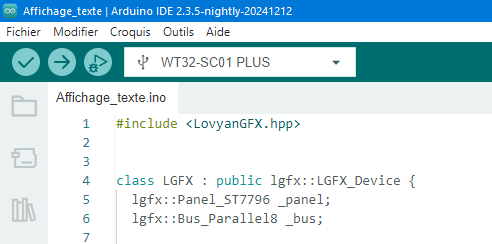


Figure 19 : Programme d’affichage de texte

Maintenant que le programme est réalisé, nous pouvons le compiler et l’envoyer sur notre carte afin qu’il soit exécuté. Pour ce faire, une fois la carte branchée, il suffit de sélectionner le port correspondant et de téléverser le programme comme indiqué ci-dessous :

Figure 20 : Téléversement du programme sur la carte



**1**

**2**

Une image contenant texte, capture d’écran, gadget, multimédia

Le contenu généré par l’IA peut être incorrect.Après compilation et téléversement, nous devrions obtenir le résultat suivant sur la carte :

Figure 21 : Résultat d’exécution du programme

À ce stade, vous connaissez les bases de la programmation de la carte WT32-SC01 Plus et pouvez maintenant expérimenter en modifiant la partie initialisation et boucle du programme. Vous pouvez utiliser de nombreuses bibliothèques afin de réaliser différents programmes. Attention toutefois, pour pouvoir utiliser certains aspects de l’écran comme le tactile par exemple, il est important de correctement configurer les pins adéquats (nous avons simplement configuré les pins d’affichage).

Vous pourrez trouver plus d’informations et d’exemples de programmes sur le Github du projet de contrôle de base holonome à ce [lien](https://github.com/Naellll/Controle-de-base-holonome-via-WT32-SC01-Plus).

## Commandes de bases pour LovyanGFX

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.Pour finir, voici une liste de quelques fonctions de base de la bibliothèque LovyanGFX vous permettant d’expérimenter (vous pourrez retrouver sa version compète sur le Github) :

Figure 22 : Fonctions de base