Salutations à toute et à tous,

Vous trouverez ici les étapes à suivre afin de recréer un streamdeck Wi-Fi qui permet de changer les scènes de son OBS à l'aide d'OBS WebSocket et d'afficher le chat de sa chaîne Twitch en temps réel.

Allons-y étape par étape. Dans un premier temps, nous allons récupérer les informations nécessaires afin d'accéder au chat Twitch à l'aide d'un script Python. Par la suite, nous l'implémenterons dans un terminal de commande à l'aide de Serial.print() et d'un code Arduino, en utilisant un ESP32 et le Wi-Fi. Ensuite, nous afficherons le chat sur l'écran ILI9488.

1) Tchat Twitch en temps réel

1.0) Compréhension du fonctionnement des codes pour récupérer le chat Twitch

Dans ce document, je ne rentrerai pas dans les détails de la communication et des interactions entre le code et l'API Twitch (https://dev.twitch.tv/docs/api/) pour récupérer les informations. D'autres personnes le font déjà très bien, de manière plus professionnelle. Cependant, je vais tout de même expliquer les bases de cette communication afin de mieux comprendre l'architecture de mes codes Python et Arduino.

1.0.a) Les toaken

Cette fonction permet une connexion sécurisée avec les services Twitch. Elle permet d'obtenir un token OAuth, qui agit comme une clé d'accès temporaire. Ce token est requis pour authentifier l'utilisateur et prouver que le programme a le droit d'accéder aux fonctionnalités spécifiques de l'API Twitch, comme rejoindre un chat ou récupérer des informations sur une chaîne. En pratique, cette fonction communique directement avec l'API Twitch, en envoyant une requête HTTP avec les identifiants de l'application (*client ID* et *client secret*). Une fois le token reçu, il est utilisé pour les interactions ultérieures.

La fonction utilisée dans mes codes : get_oauth_token (Authentification simplifiée avec Twitch)

1.0.b) Connexion au chat

Une fois le token obtenu, cette fonction établit une connexion avec le serveur IRC de Twitch, qui est le protocole utilisé pour le chat en temps réel. En utilisant le token comme identifiant, cette étape permet au programme de rejoindre un ou plusieurs salons (channels) et de commencer à interagir avec les messages en direct. Concrètement, cette fonction configure un client réseau, gère l'authentification auprès du serveur IRC, et s'assure que la connexion reste active.

La fonction utilisée dans mes codes : connect_to_chat (Connexion au serveur IRC de Twitch)

1.0.c) Lecture du chat

Une fois connecté au chat, cette fonction s'occupe de la réception et du traitement des messages. Elle écoute en continu les données envoyées par le serveur IRC, filtre les messages pertinents, et les traite en fonction des besoins. Par exemple, elle peut afficher les messages dans le moniteur série ou les transmettre à un écran.

La fonction utilisée dans mes codes : listen_to_chat (Lecture et gestion des messages)

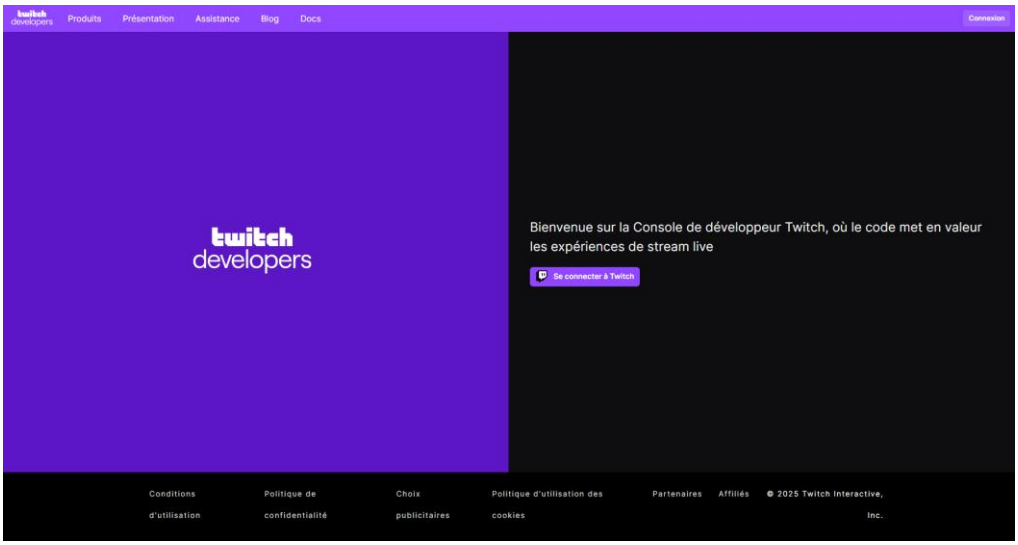
1.1) Les codes

1.1.a) Création d'une application

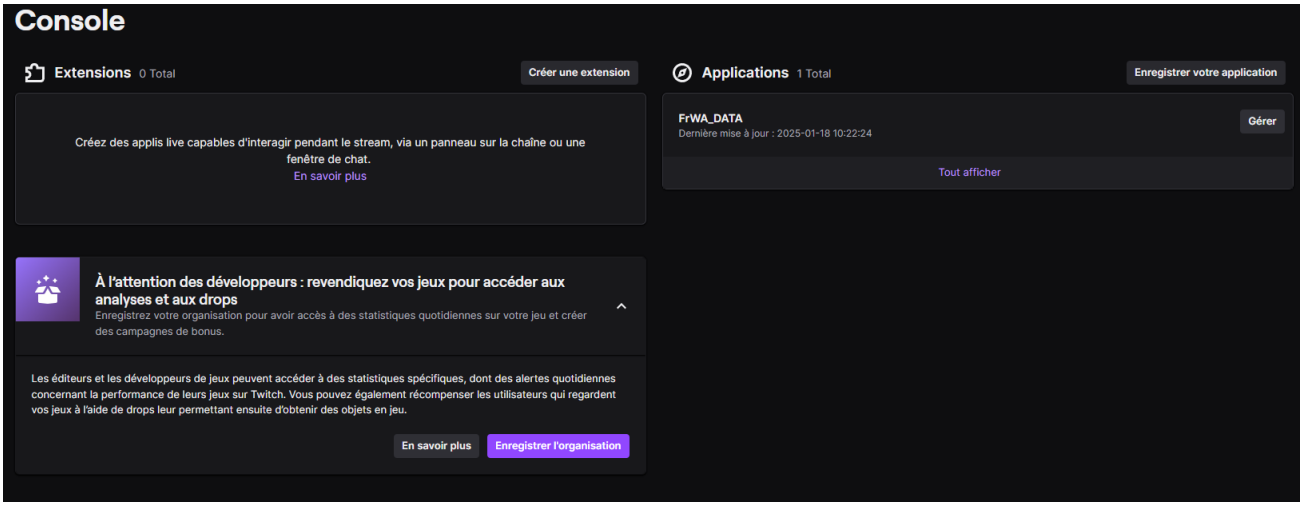
Twitch est une plateforme de streaming qui met à disposition des API pour contrôler de nombreuses fonctionnalités : les événements comme les *followers*, le nombre de spectateurs, la catégorie de jeu, et le chat (la catégorie qui nous intéresse).

Pour récupérer les informations du chat, nous devons d'abord nous connecter à la plateforme Twitch Developers Dashboard (https://dev.twitch.tv/login) qui nous ouvre les portes de l'interface Twitch via le Web.

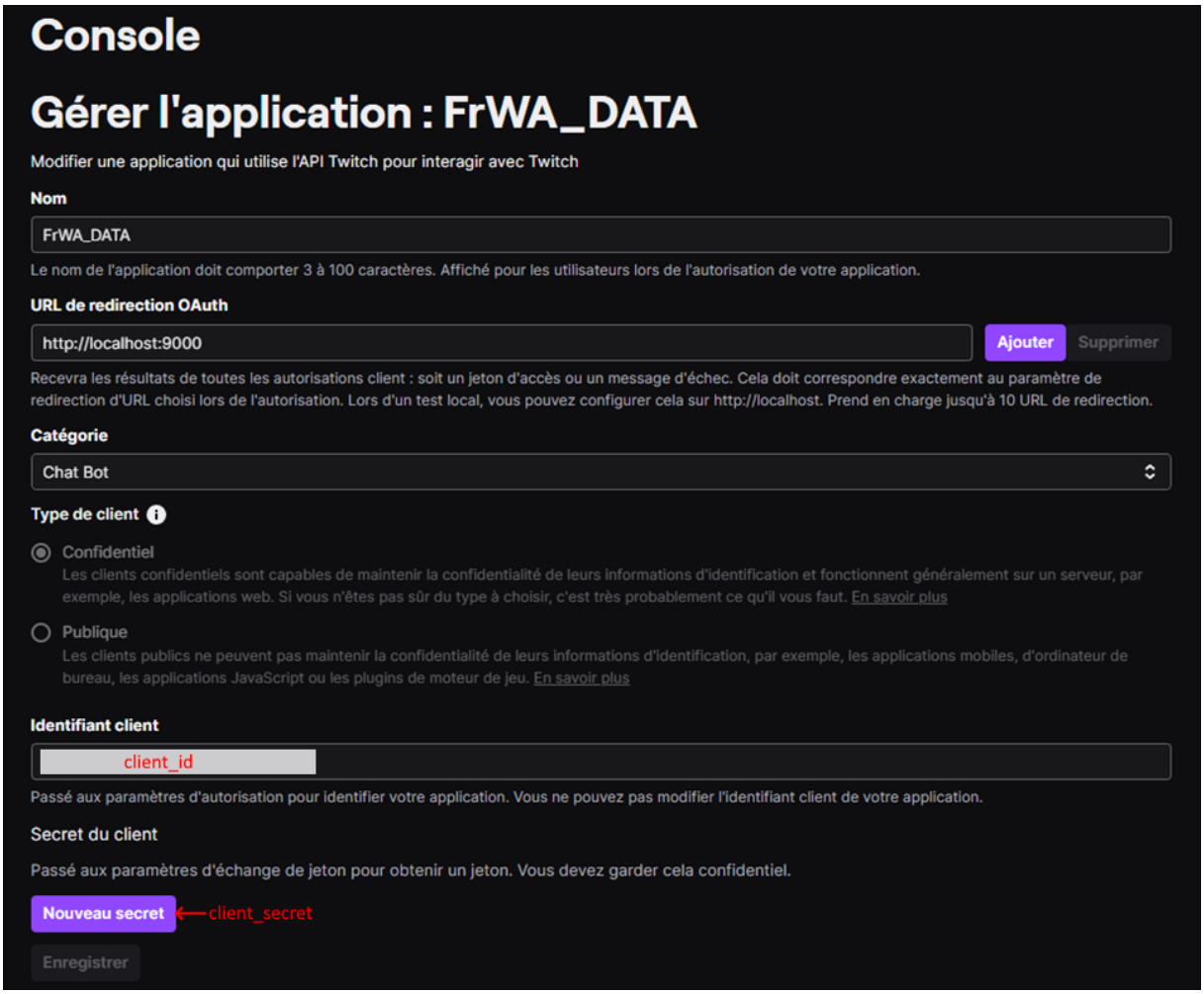
Ouvrir le site :



Se connecter à votre compte Twitch (*login*) :



~~Créer~~ enregistrer une nouvelle application :



Important : Notez dans un bloc-notes votre client_id et le code secret (*client_secret*). Si c'est la première fois, créez un nouveau secret. Le nom de l'application a peu d'importance, et la catégorie correspond à un chatbot.

1.1.b) Code Python et chat Twitch

Dans le dossier Python, un code est nommé Twitch_chat_temps_reel.py. Afin d'avoir dans le terminal le chat en temps réel, vous devrez remplir les informations nécessaires entre les apostrophes (' '). Une fois votre client_id et votre client_secret remplis, le code pourra se connecter à l'API Twitch et retourner en temps réel le chat de la chaîne souhaitée (*twitch_channel*).

```
# Informations de ton application
client_id = "client_id"
client_secret = "client_secret"
twitch_channel = "nom_chaine_twitch"
```

Pour lire et retourner le chat, j'utilise un pseudonyme générique : *justinfan12345*..

1.1.c) Code Arduino et chat twitch

Afin de lire le chat Twitch, j'utilise un microcontrôleur capable d'accéder à Internet via le Wi-Fi : un ESP32 DevKit V1. Le code permettant de lire le chat Twitch en temps réel se nomme ESP32_Twitch_tchat.ino.

Pour se connecter à Internet en Wi-Fi et au chat, nous aurons besoin de quelques bibliothèques :

#include <WiFi.h>

WiFi.h : Permet d'activer le Wi-Fi sur l'ESP32 et de se connecter à un box/routeur.

#include <WiFiClientSecure.h>

#include <ArduinoJson.h>

WiFiClientSecure.h et ArduinoJson.h : Permettent de communiquer avec l'API Twitch, d'envoyer des requêtes pour accéder au chat, et de récupérer les données sous forme JSON. Ces données sont ensuite affichées dans un terminal avec la commande Serial.print().

Pour se connecter à Internet, vous devrez entrer le nom de votre box/routeur et son mot de passe. Pour accéder au chat Twitch, il faudra renseigner votre client_id, votre client_secret et le nom de la chaîne pour laquelle vous souhaitez récupérer le chat en direct.

```
// Informations de connexion Wi-Fi
const char *ssid = "nom_box_wifi";
const char *password = "mot_de_passe_box_wifi";

// Informations Twitch
const char *client_id = "client_id";
const char *client_secret = "client_secret";
const char *twitch_channel = "twitch_channel";
```

Comme pour le code Python, je me suis grandement inspiré d'un pseudonyme générique : *justinfan12345*.

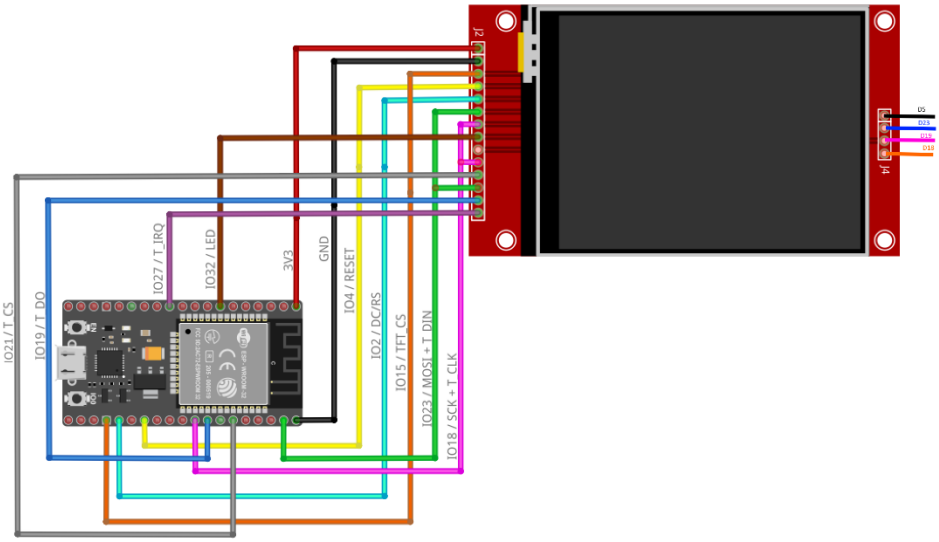
2)L'écran tactile ili9488

2.0) présentation de l'ecran tactile ili9488

Le module ILI9488 est un écran tactile couleur fonctionnant en SPI (protocole de communication), offrant une résolution de 480x320 pixels pour le modèle utilisé. Il permet d'afficher des images, du texte ou des données en temps réel. Cet écran est équipé d'un contrôleur ILI9488 intégré, ce qui facilite la gestion de l'affichage via des commandes simples. Il peut inclure un écran tactile résistif et possède un lecteur de carte SD pour stocker des images.

2.0.a) Connexion à l'ESP32

Afin de pouvoir utiliser l'écran tactile, nous devons le connecter à l'ESP32 Devkit V1. Pour ce faire, voici un schéma de montage que vous trouverez dans le dossier *Hardware*.



2.0.b) Test affichage ILI9488 et ESP32

Pour afficher une image sur l'écran ILI9488, vous aurez besoin de télécharger et d'utiliser plusieurs bibliothèques indispensables, chacune ayant un rôle spécifique :

#include <SPI.h>

Cette bibliothèque est nécessaire pour établir une communication avec l'écran via le protocole SPI. Elle permet de gérer l'échange de données entre l'ESP32 et le contrôleur ILI9488. Sans SPI, il serait impossible d'envoyer les informations d'affichage à l'écran.

#include <TFT_eSPI.h>

Cette bibliothèque est spécialement conçue pour les écrans TFT comme le ILI9488. Elle simplifie grandement l'utilisation de l'écran en fournissant des fonctions prêtes à l'emploi pour dessiner des pixels, du texte, des lignes, ou encore charger des images. Elle est également optimisée pour le matériel comme l'ESP32, garantissant des performances rapides.

#include <SD.h> et **#include <FS.h>**

Ces bibliothèques permettent de lire des fichiers depuis une carte SD, ce qui est essentiel si vous voulez afficher des images stockées sous forme de fichiers. La carte SD est un moyen pratique de stocker des ressources volumineuses (comme des images JPEG) que l'ESP32 peut charger et afficher sur l'écran.

#include <JPEGDecoder.h>

Cette bibliothèque est nécessaire pour décoder des fichiers JPEG en données brutes exploitables par l'écran. Le ILI9488 n'est pas capable de comprendre directement les fichiers JPEG : ils doivent d'abord être décompressés en pixels, et c'est là que *JPEGDecoder* intervient. Avec cette bibliothèque, vous pouvez lire des images haute résolution sans saturer la mémoire de l'ESP32, car elle effectue le traitement image par image.

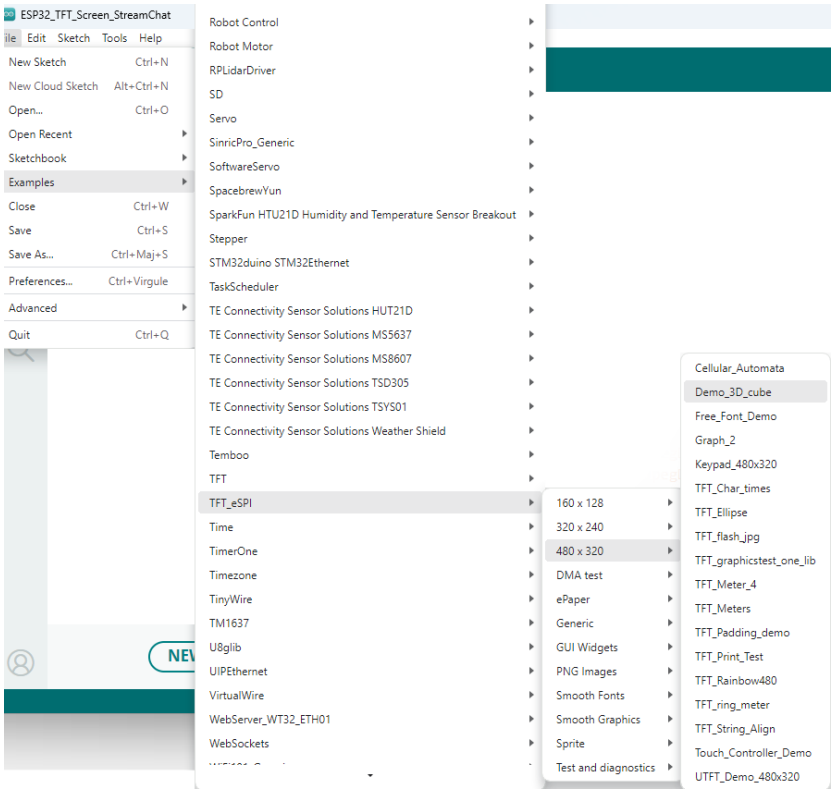
2.0.c) TFT_eSPI

La bibliothèque TFT_eSPI.h permet une compatibilité avec plusieurs microcontrôleurs. Afin de l'adapter au montage présenté précédemment, nous allons modifier certaines lignes dans son fichier de configuration User_Setup.h.

```
// Only define one driver, the other ones must be commented out
//#define ILI9341_DRIVER
//#define ST7735_DRIVER      // Define additional parameters below for this di
//#define ILI9163_DRIVER     // Define additional parameters below for this di
//#define S6D02A1_DRIVER
//#define RPI_ILI9486_DRIVER // 20MHz maximum SPI
//#define HX8357D_DRIVER
//#define ILI9481_DRIVER
//#define ILI9486_DRIVER
#define ILI9488_DRIVER     // WARNING: Do not connect ILI9488 display SDO to M
//#define ST7789_DRIVER      // Full configuration option, define additional p
//#define ST7789_2_DRIVER    // Minimal configuration option, define additiona
//#define R61581_DRIVER
//#define RM68140_DRIVER
//#define ST7796_DRIVER
//#define SSD1963_480_DRIVER    // Untested
//#define SSD1963_800_DRIVER    // Untested
//#define SSD1963_800ALT_DRIVER // Untested


#define TFT_BL   32            // LED back-light control pin
#define TFT_BACKLIGHT_ON HIGH  // Level to turn ON back-light (HIGH or LOW)


#define TFT_MISO 19
#define TFT_MOSI 23
#define TFT_SCLK 18
#define TFT_CS   15  // Chip select control pin
#define TFT_DC    2  // Data Command control pin
#define TFT_RST   4  // Reset pin (could connect to RST pin)
//#define TFT_RST  -1  // Set TFT_RST to -1 if display RESET is connected to ESP32 board RST

#define TOUCH_CS 21     // Chip select pin (T_CS) of touch screen
```

2.0.d) TFT_eSPI

Pour tester l'écran tactile, vous pouvez lancer le code d'exemple de la librairie *TFT_eSPI* pour un écran de 480x320 pixels nommé *Demo 3D cube*.



2.1) Présentation du code pour afficher le chat sur l'écran

2.1.a) Présentation du code pour afficher le chat sur un écran ILI9488

Afin de simplifier la lisibilité, j'ai créé deux fichiers .h qui permettent, d'une part, de configurer l'écran et, d'autre part, d'afficher les images depuis la carte SD. Ces fichiers sont respectivement setup_tft.h et jpeg_fonction_tft.h.

setup_tft.h contient une seule fonction à appeler dans le void setup() du code principal Arduino pour configurer le SPI et l'écran : setupTFT().

jpeg_fonction_tft.h contient une fonction utilisant plusieurs paramètres pour décoder une image JPEG et l'afficher à l'emplacement voulu (x et y) sur l'écran : drawSdJpeg(const char *filename, int xpos, int ypos).

2.1.b) Code Arduino configuration et démarrage

De la même manière que pour le code affichant le chat Twitch dans le terminal, vous devrez entrer les informations nécessaires pour vous connecter à Internet et accéder au chat Twitch. La seule différence ici est que la fonction TFT_eSPI tft = TFT_eSPI() configure l'écran, désigné par le nom de la fonction tft.

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <ArduinoJson.h>
#include "setup_tft.h"
#include "jpeg_fonction_tft.h"

TFT_eSPI tft = TFT_eSPI();

// Informations de connexion Wi-Fi
const char *ssid = "TP-nom_box_wifi";
const char *password = "mot_de_passe_box_wifi";

// Informations Twitch
const char *client_id = "client_id";
const char *client_secret = "client_secret";
const char *twitch_channel = "twitch_channel";
```

Une partie de code facultative a été ajoutée pour afficher un logo au démarrage de l'ESP32. Une image nommée *PP.jpg*, directement placée sur la carte SD, est utilisée. Elle mesure 300x300 pixels, ce qui la rend adaptée à un affichage sur un écran configuré en mode portrait (verticale) ou paysage (horizontale).

Après cet affichage de 2 secondes, les fonctions fillScreen() et setTextColor() permettent de mettre l'écran en noir et de configurer le texte en blanc sur un fond noir.

```
setupTFT();

// Affichage d'une image d'initialisation
tft.setRotation(2);  // 1:paysage //
tft.fillScreen(0xFFFF); //Fond d'écrant en blanc
//position de l'image sur l'écran
int x = (tft.width()  - 300) / 2 - 1;
int y = (tft.height() - 300) / 2 - 1;
//affichage de l'image
drawSdJpeg("/PP.jpg", x, y); // Affiche une image JPEG depuis la carte SD
delay(2000);
tft.fillScreen(0x0000); //Fond d'écrant en noir
tft.setTextColor(0xFFFF, 0x0000);  // Texte blanc sur fond noir
```

2.1.c) Code Arduino affichage du chat Twitch sur l'écran

La fonction permettant d'afficher le chat est addMessageToDisplay(). Elle est située à la fin de la fonction listenToChat(), qui affiche les messages dans le terminal.

Avec la configuration actuelle de 20 pixels entre chaque ligne, il est possible d'afficher 24 messages sur les 480 pixels de hauteur de l'écran. Le nombre total de messages que l'on souhaite afficher est défini par la variable MAX_MESSAGE.

```
#define MAX_MESSAGES 24  // Nombre maximum de messages à afficher
String messages[MAX_MESSAGES];  // Tableau pour stocker les messages
int messageCount = 0;  // Nombre de messages actuels
```

Le code permettant d'afficher le chat Twitch est directement intégré à la fonction listenToChat(), qui récupère le chat en ligne. Vous n'aurez qu'à appeler cette fonction périodiquement pour afficher les nouveaux messages en temps réel sur l'écran.

```
void loop() {
  // Écouter et afficher les messages du chat
  listenToChat();

  // Vérifier si la connexion est toujours active
  if (!client.connected()) {
    Serial.println("[INFO] Reconnexion au serveur Twitch...");
    connectToTwitch();
  }

  delay(10); // Laisser du temps pour éviter la surcharge CPU
}
```