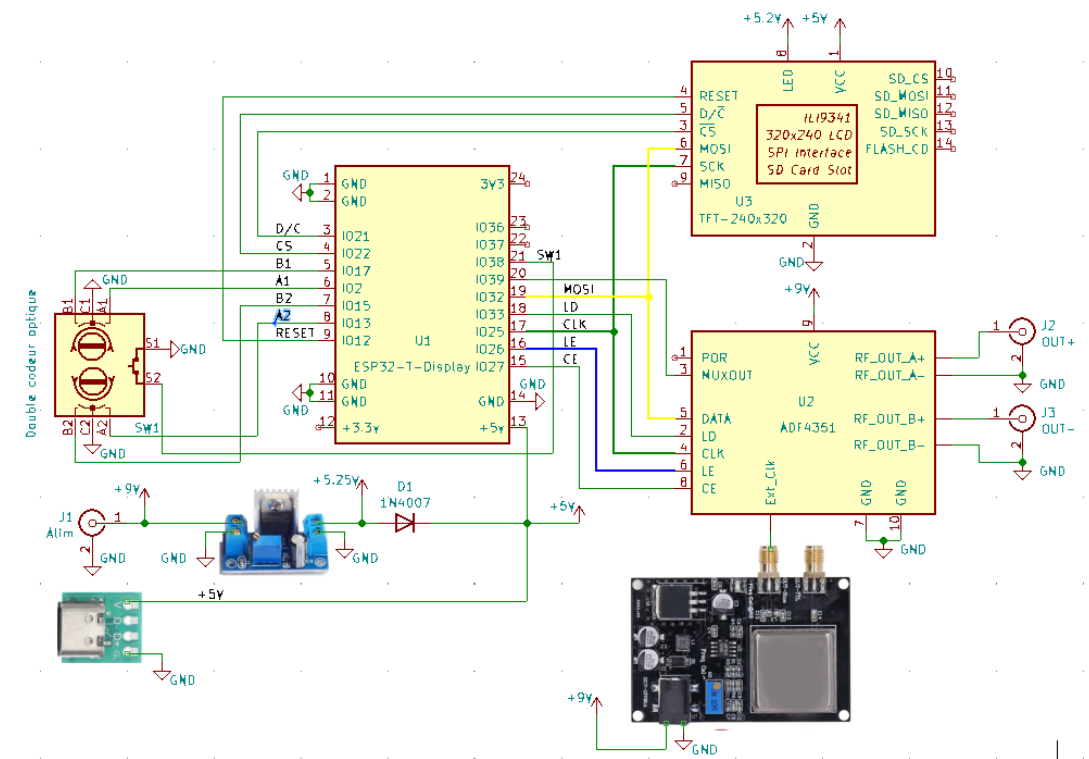
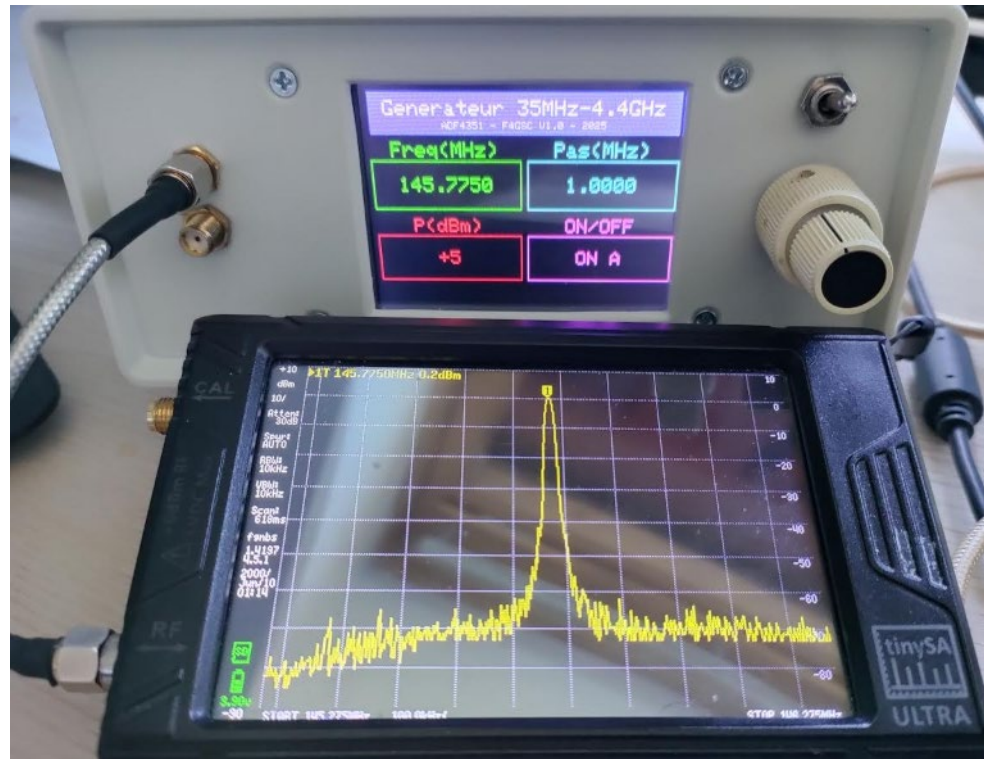


L'intelligence artificielle appliquée aux activités radioamateurs

Jean-Luc Levant – F4GSC



Introduction

- Qui a déjà développé des applications à base de carte Arduino?
- Qui souhaiterait le faire?
- Quels sont les freins?
 - Apprentissage du langage...
- A qui d'adresse cette présentation?
 - Aux RA qui souhaitent découvrir et s'initier au développement d'application à base de cartes Arduino par la pratique et sans avoir à apprendre dans un premier le langage de programmation.

Introduction

- Comment développer des applications Arduino?
 - Faire appel à l'intelligence artificielle pour sous-traiter la partie programmation;
 - Se concentrer sur:
 - Le cahier des charges;
 - La vérification des programmes fournis par l'IA;
 - Développer tout en apprenant la programmation avec l'aide d'un « coach » en programmation;

Avertissement:

- *L'IA facilite la formation du RA qui souhaite développer tout en apprenant;*
- *L'IA n'est pas un outil magique et sans faille;*
- *Ses propositions ne prennent en compte que les ressources existantes;*
- *Il ne faut donc pas lui faire confiance, vérifier ce qu'il propose et à diriger son développement.*

Rôle du radioamateur (RA)

- Il est le chef de projet:
 - Ecrit les cahiers des charges du projet à concevoir;
 - Découpe le projet en tâches;
 - Délègue la programmation à son sous-traitant (IA);
 - Assure le suivi du développement en relation avec son sous-traitant (IA);
 - Vérifie que la conception est conforme au cahier des charges.

Cahier des charges

- Contenu du cahier des charges pour optimiser le nombre de prompt
 - Prompt: canal de communication et de dialogue avec l'IA;
 - Il doit être précis pour éviter le nombre d'aller et retour pour la mise au point du programme:
 - Apprendre à dialoguer avec l'IA;
 - Cahier des charges doit comporter:
 - Règles de conception: Comment l'IA doit organiser son développement et les étapes à suivre;
 - Les contraintes technologiques;
 - Spécifications fonctionnelles.

Cahier des charges

- Contenu du cahier des charges pour optimiser le nombre de prompt

Prompt	Ce que doit contenir le prompt	Exemples
Règles de conception à appliquer strictement:	<ul style="list-style-type: none"> - Exprimer de manière très claires et détaillées le travail et la méthode que doit appliquer l'IA; 	<ul style="list-style-type: none"> - Inventaire littéral de tous les éléments contextuels de l'image; - Mesurer visuellement sur le graphisme au pixel près la taille; - Vérifier les débordements de texte; - ...
Contraintes technologiques	<ul style="list-style-type: none"> - Description matérielle: composants utilisés; - Les connexions entre les circuits et l'ESP32; - Les interfaces utilisées: SPI, I2C, UART, USB, BlueTooth, LORA...; - Les performances à atteindre; - 	<ul style="list-style-type: none"> - Le contrôleur d'écran couleur est un IL9341 - la taille de l'écran est de 320 x 240 pixels; - La Librairie est Adafruit_ILI9341; - Les broches utilisées sont: TFT_MOSI 32, TFT_SCK 25, TFT_CS 22, TFT_DC 21, TFT_RST 12;
Spécifications fonctionnelles	<ul style="list-style-type: none"> - Les fonctions à mettre en œuvre; - Les contraintes fonctionnelles à respecter; - ... 	<ul style="list-style-type: none"> - Le bandeau du titre doit être positionné à 6 pixels des bords de l'écran; - La distance entre un label (freq,pas,P,ON/OFF) et la ligne horizontale d'un cadre doit être en haut et en bas de 2 pixels; - Le label doit être centré par rapport à la ligne horizontale du cadre; - L'écran doit être tourné de 270 degrés.

Différences entre ChatGPT, OPUS et Deepseek

- Limitations en fonction des activités

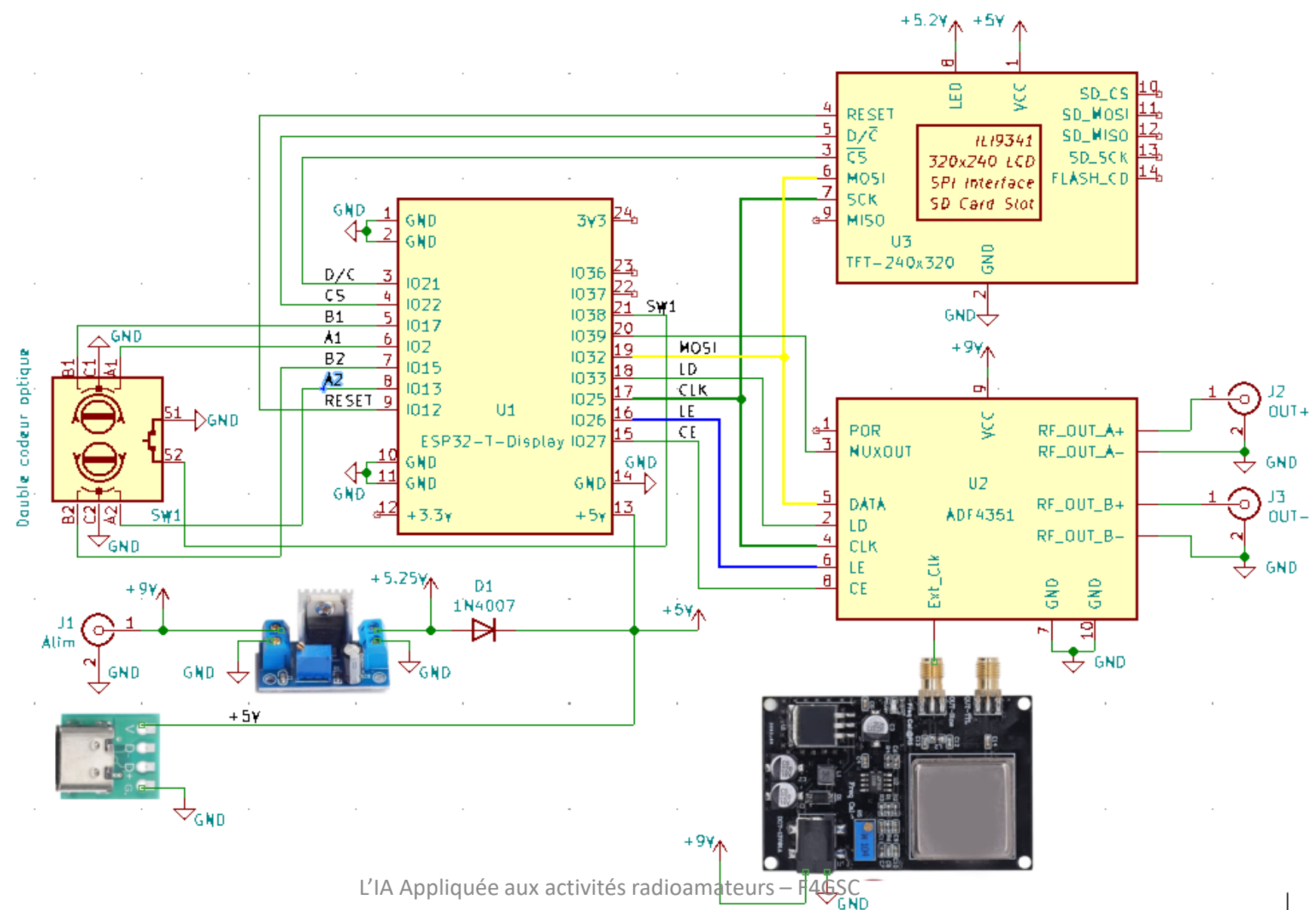
	ChatGpt 5.0	OPUS 4.1	DeepSeek
GUI Eléments contextuels	Tous les éléments contextuels 👍	Tous les éléments contextuels(Texte, cadre, couleurs, police, disposition spatiale, dimensions...). 👍 👍	Texte (OCR) Ne reconnaît pas les couleurs, disposition spatiale et les dimensions.. 👎
Codeurs Optiques	Difficile à converger; Nécessite plusieurs reprises de code. 👎	Nécessite que deux reprises. 👍 👍	Difficile à converger; Nécessite plusieurs reprises de code. 👎
ADF4351	Après 2 à 3 passes	Après 2 à 3 passes	Après 2 à 3 passes
Librerie Arduino	Fournit tous les fichiers unitaires.	Fournit tous les fichiers unitaires.	Fournit tous les fichiers unitaires .

Développement d'un Générateur RF 35 MHz - 4.4 GHz

Contraintes technologiques

- Carte de génération RF: ADF4351
 - 35 MHz - 4.4 GHz;
 - -4,-1,+2,+5 dBm.
- Pilotage: Arduino ESP32,
 - alimentation +5 V;
 - signaux logiques + 3,3 V.
- Interface graphique:
 - Ecran TFT 320 x 240, contrôleur ILI9341, +5V, interface de pilotage +3.3V;
- Double Codeur optique avec bouton poussoir;
- Oscillateur 10 MHz thermostaté, stabilité 0.1 à 0.01 PPM (1 à 0.1 Hz).

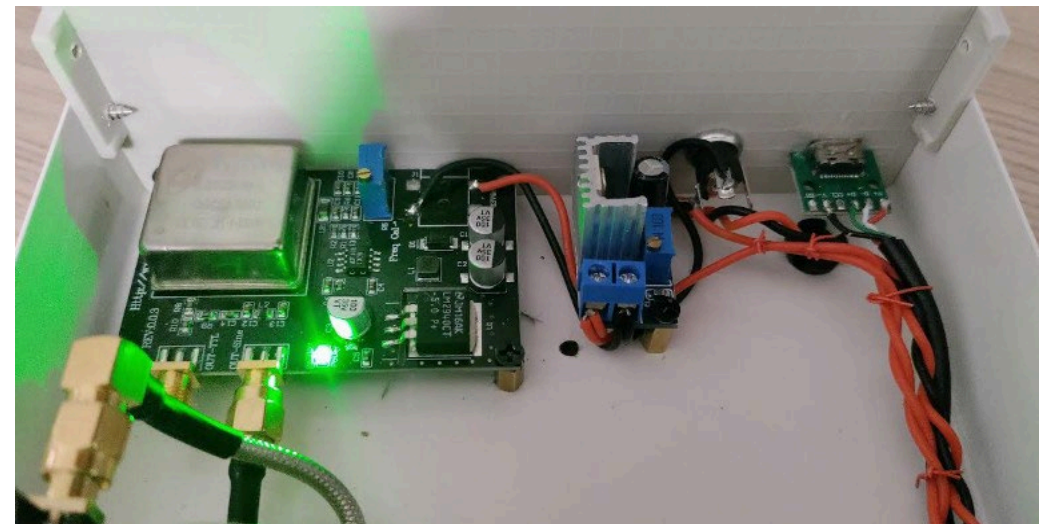
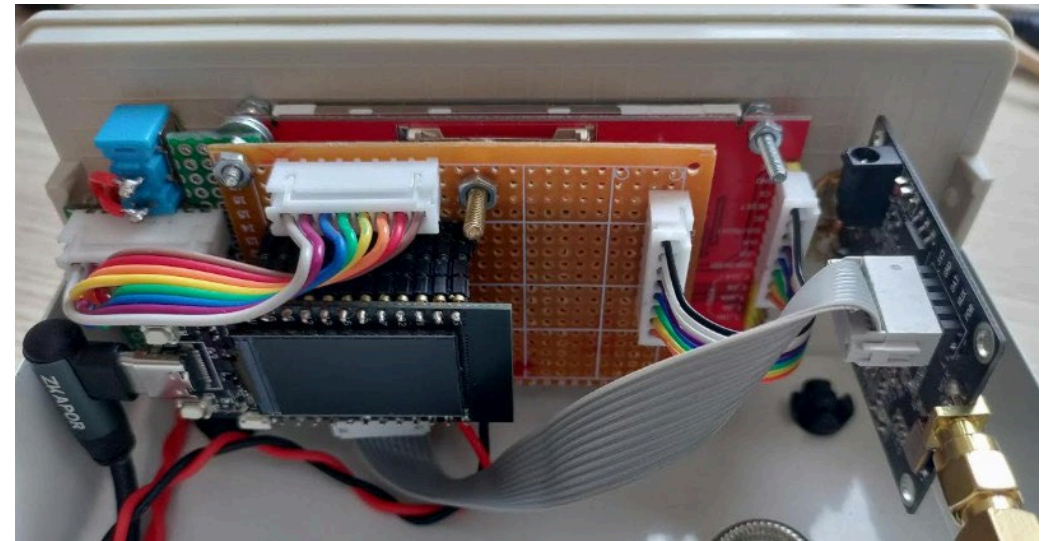
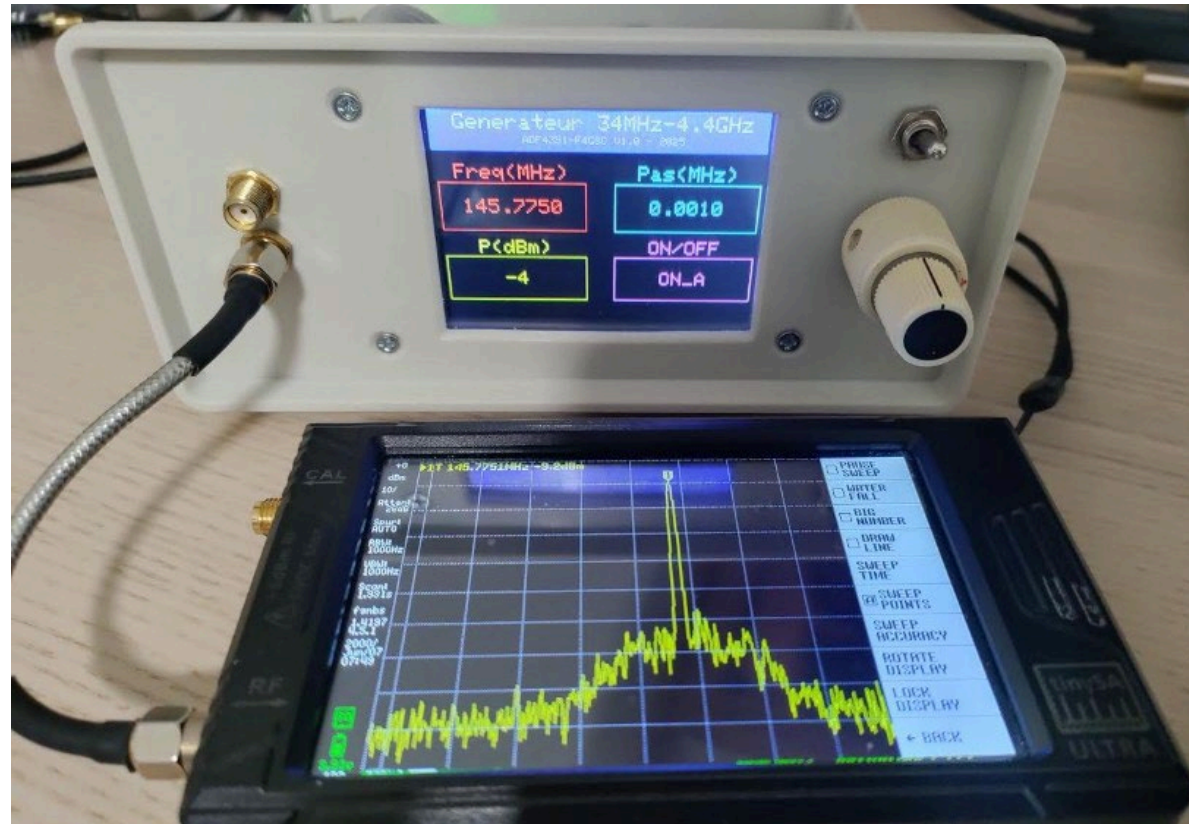
Schéma



Les étapes du projet

- Conception de l'interface graphique;
- Intégration des codeurs optiques;
- Sauvegarde des paramètres en mémoire E²PROM;
- Intégration de l'ADF51;

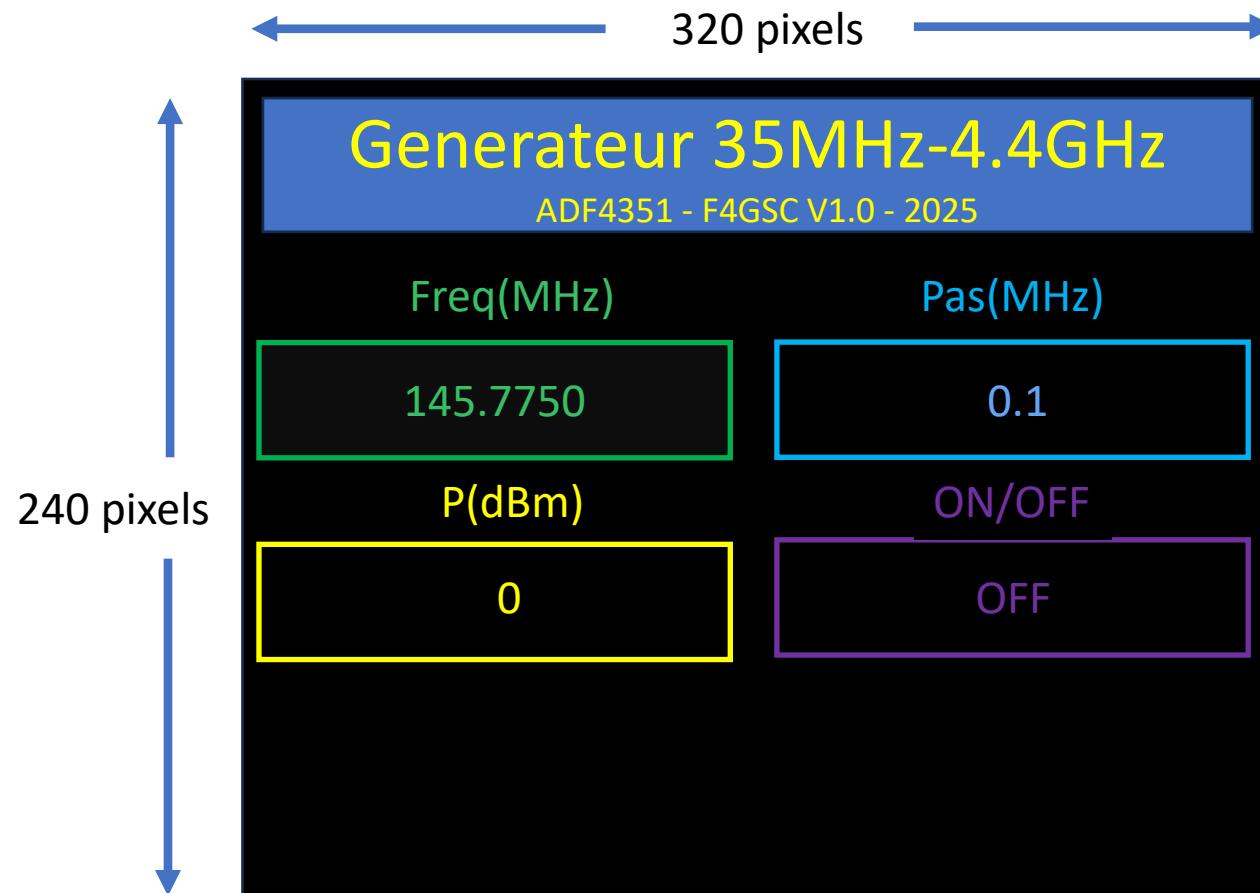
Le produit final



Conception de l'interface graphique (GUI)

Conception de l'interface graphique

- Cahier des charges
 - Description graphique du GUI



Conception de l'interface graphique

- Cahier des charges
 - Description littérale

Développer le programme Arduino ESP32 de l'interface graphique montré sur le dessin inclus:

Règles de conception à appliquer strictement:

- Inventaire littéral de tous les éléments contextuels de l'image (textes, valeurs, couleurs, positions, dimensions en pixels..);
- Mesurer visuellement sur le graphisme au pixel près la taille des différents éléments contextuels: proportions, Tailles de police relatives, espacements visibles;
- Vérifier les débordements de texte: taille des polices et longueurs des chaînes de texte;
- Pour déterminer la bonne taille de la police prendre en compte la hauteur disponible et la longueur du texte;
- Les dimensions et les positions spatiales doivent être faites à partir de l'image fournie;
- Vérifier si le programme correspond à 100% au graphisme.

Contraintes technologiques:

- Contrôleur d'écran couleur est un ILI9341
- Taille de l'écran 320 x 240 pixels;
- La Librairie est Adafruit_ILI9341;
- Interface SPI (ESP32): TFT_MOSI 32, TFT_SCK 25, TFT_CS 22, TFT_DC 21, TFT_RST 12;

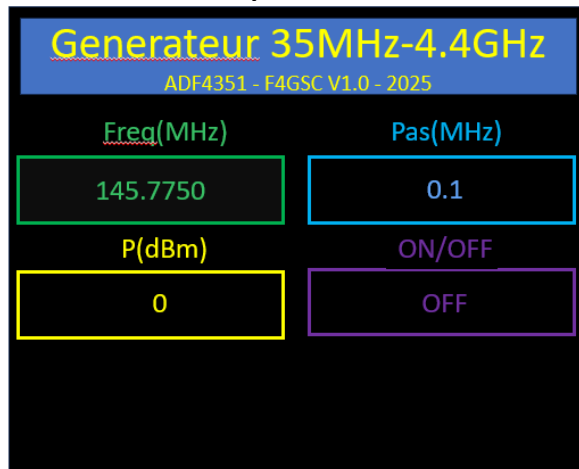
Spécifications fonctionnelles:

- Le bandeau du titre doit être positionné à 6 pixels des bords de l'écran;
- Le label doit être centré par rapport à la ligne horizontale du cadre;
- Les lignes verticales des cadres doivent être à 6 pixels des bords de l'écran;
- La distance entre un label (freq, pas, P,ON/OFF) et la ligne horizontale d'un cadre doit être en haut et en bas de 2 pixels;
- La distance entre le cadre et le prochain label doit être de 8 pixels;
- La valeur numérique dans le cadre doit être centrée et la taille de 2;
- L'écran doit être tourné de 270 degrés;

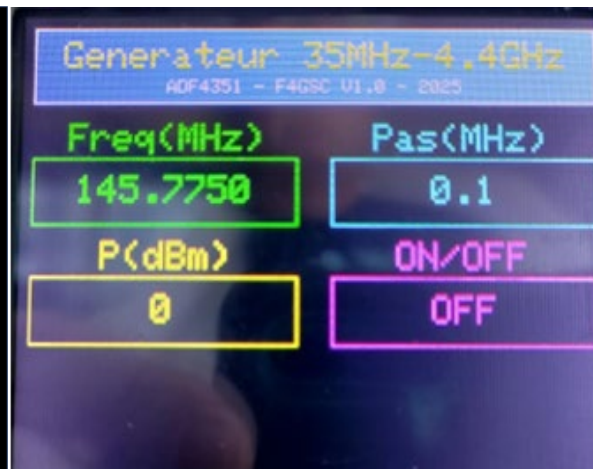
Conception de l'interface graphique

- La conception avec Calude sonnet, Opus41.1 ChatGPT 5.1 et DeepSeek

Description du GUI



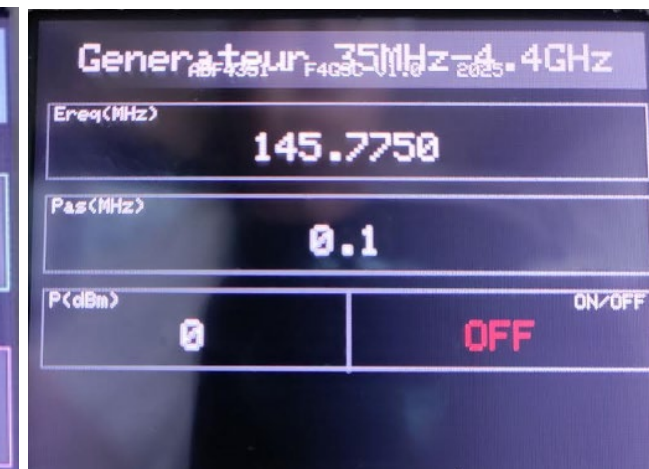
Opus 4.1



Chatgpt 5.1 / Claude Sonnet

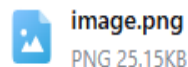


DeepSeek



Conception de l'interface graphique

• Prompt



Développer le programme Arduino ESP32 de l'interface graphique;
 Le contrôleur d'écran couleur est un IL9341 et la taille de l'écran est de 320 x 240 pixels;
 La Librairie est Adafruit_ILI9341;
 Les broches utilisées sont:

TFT_MOSI	32
TFT_SCK	25
TFT_CS	27
TFT_DC	21
TFT_RST	12

Le bandeau du titre doit être positionné à 8 pixels des bords de l'écran;
 La distance entre un label (freq,pas,P,ON/OFF) et la ligne horizontale du cadre doit être de 2 pixels;
 Le label doit être centré par rapport à la ligne horizontale du cadre;
 Les lignes verticales des cadres doivent être à 6 pixels des bords de l'écran;
 La distance entre le cadre et le prochain label doit être de 8 pixels;
 La valeur dans le cadre doit être centré.

L'IA résume le développement réalisé

✓ Conformité aux règles de conception:

Inventaire contextuel respecté:

Bandeau bleu avec "Générateur 35MHz-4.4GHz" (jaune, taille 3)
 Sous-titre "ADF4351 - F4GSC V1.0 - 2025" (jaune, taille 2)
 4 sections colorées: Freq (vert), Pas (cyan), P (jaune), ON/OFF (magenta)

Mesures au pixel près:

Marges de 6 pixels des bords de l'écran
 Labels à 2 pixels des cadres horizontaux
 Espacement de 8 pixels entre sections
 Bandeau: 308×104 pixels (centré)
 Colonnes: 150 pixels de largeur chacune

Configuration technique:

- ✓ Contrôleur ILI9341
- ✓ Écran 320×240 pixels
- ✓ Librairie Adafruit_ILI9341
- ✓ Pins SPI ESP32 configurés
- ✓ Rotation 270 degrés (setRotation(3))

Fonctionnalités:

Textes centrés horizontalement et verticalement
 Valeurs numériques en taille 2
 Bordures doubles pour les cadres
 Fonctions de mise à jour: updateFrequency(), updatePower(), updateStep(), updateStatus()

Conception de l'interface graphique

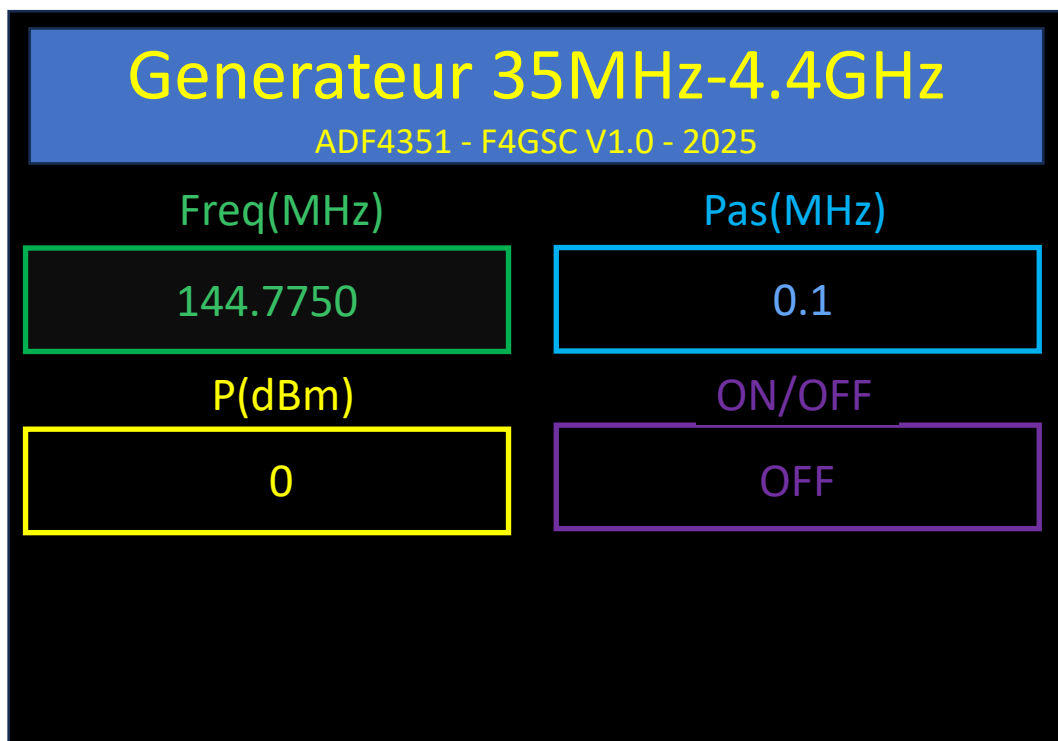
- Téléversement du programme et résultat de la première compilation
 - Une erreur est détectée → Il manque les bibliothèques de pilotage de l'écran.

Erreur pendant la compilation	Réponse et assistance de l'IA
<pre>C:\Users\Jluc\AppData\Local\Temp\.arduinoIDE-unsaved2025829-1424-11v888l.f6fr\sketch_sep29a\sketch_sep29a.ino:1:10: fatal error: Adafruit_GFX.h: No such file or directory 1 #include <Adafruit_GFX.h></pre>	<p>Vous devez installer les bibliothèques nécessaires. Voici comment résoudre cette erreur :</p> <p>Méthode 1 : Installation via le Gestionnaire de Bibliothèques Arduino IDE Ouvrez Arduino IDE Allez dans : Outils → Gérer les bibliothèques...</p> <p>Recherchez et installez : Adafruit GFX Library par Adafruit Adafruit ILI9341 par Adafruit</p> <p>Méthode 2 : Installation manuelle Si la méthode automatique ne fonctionne pas :</p> <p>Téléchargez les bibliothèques : Adafruit GFX Library Adafruit ILI9341</p> <p>Installez-les : Décompressez les fichiers Copiez les dossiers dans : Documents/Arduino/libraries/ Redémarrez Arduino IDE</p> <p style="color: red; text-align: right;">L'IA assiste dans la résolution de l'erreur!!</p>

Conception de l'interface graphique

- Vérification du programme généré
 - Temps de compilation et téléversement ~4 mn;
 - Résultat prometteur! Reprise sur la taille de la police des paramètres;
 - Je n'ai pas écrit de lignes de code. 300 lignes générées par l'IA.

Police trop petite



Conception de l'interface graphique

- **Prompt: Demande d'amélioration.**

peux-tu augmenter la taille de la font des labels Freq, pas, P et ON_OFF en les gardant centrés par rapport aux cadres? Prendre le premier programme que tu m'as fourni.

Voici le programme avec une taille de police augmentée pour les labels, en gardant le centrage :

L'IA résume le développement réalisé

Tailles de police augmentées :

1. Labels (Freq, Pas, P, ON/OFF) :
 - Ancien : `tft.setTextSize(1)`
 - Nouveau : `tft.setTextSize(2)` → 100% plus grand
2. Valeurs dans les cadres :
 - Ancien : `tft.setTextSize(2)`
 - Nouveau : `tft.setTextSize(3)` → 50% plus grand
3. Titre principal :
 - Maintenu : `tft.setTextSize(2)` pour la première ligne
 - Maintenu : `tft.setTextSize(1)` pour la seconde ligne

Centrage préservé :

- Tous les textes restent parfaitement centrés dans leurs cadres respectifs
- La fonction `getTextBounds()` calcule automatiquement la largeur et hauteur pour un centrage précis
- Les positions sont recalculées en fonction des nouvelles tailles de police

L'interface aura maintenant une bien meilleure lisibilité avec des textes plus grands tout en conservant l'alignement et le design d'origine.

Conception de l'interface graphique

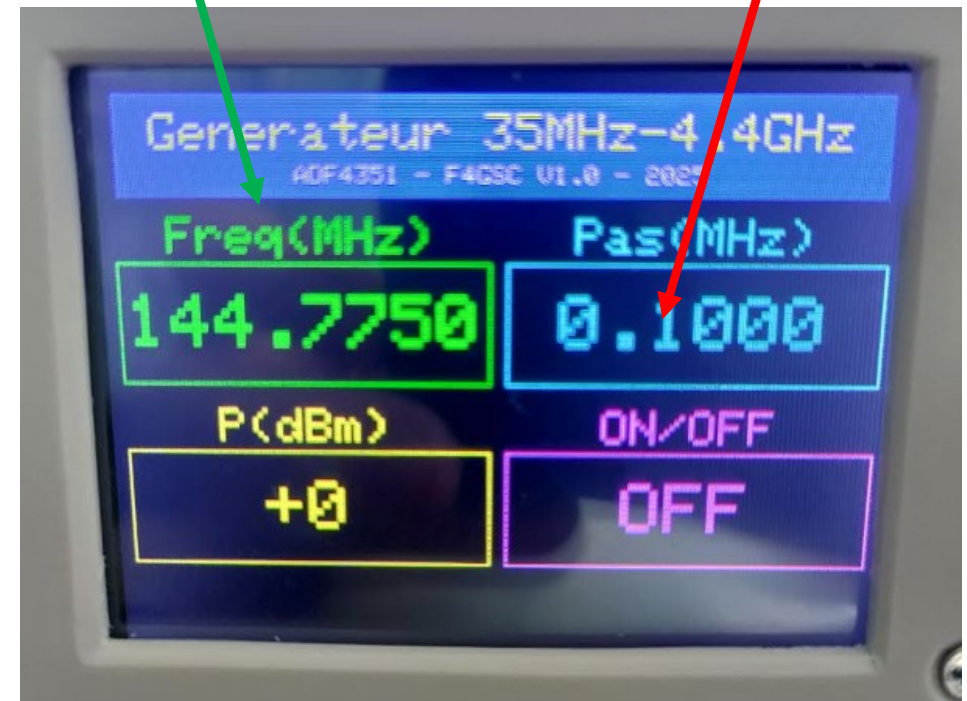
- Il a bien augmenté la taille de la police des labels mais des valeurs aussi.

Avant modification



Taille des labels OK

Police trop grande



Conception de l'interface graphique

- **Prompt:** Modification de la taille de la police des valeurs numériques.

diminuer la taille de police pour les valeurs dans le cadre

Voici le programme avec la taille de police réduite pour les valeurs dans les cadres :

L'IA résume le développement réalisé

Taille des polices maintenant :

- **Labels** (Freq, Pas, P, ON/OFF) : Taille 2 (augmentée)
- **Valeurs dans les cadres** : Taille 2 (réduite de 3 à 2)
- **Titre ligne 1** : Taille 2
- **Titre ligne 2** : Taille 1

Les valeurs dans les cadres seront maintenant plus proportionnelles par rapport aux labels et mieux adaptées à la taille des cadres, tout en restant parfaitement centrées.

Conception de l'interface graphique

- Modification de la taille de la police des valeurs numériques;
- Conception de l'interface graphique validée;
- Je n'ai pas écrit une ligne de code;
- 300 lignes écrites par l'IA.



Intégration des codeurs optiques

Intégration des codeurs optiques

• Cahier des charges

Ajouter deux codeurs optiques pour régler la fréquence, le pas de fréquence la puissance et le « ON/OFF »;

Règles de conception à appliquer strictement:

Vérifier que le sens des codeurs est bien celui attendu;

Le sous-programme d'interruption doit être le court possible.

Contraintes technologiques:

Les signaux des codeurs sont gérés par interruption et par « polling »;

```
#define PIN_A1 2
```

```
#define PIN_B1 17
```

```
#define PIN_A2 15
```

```
#define PIN_B2 13
```

Pas d'utilisation de librairie;

Spécifications fonctionnelles:

A1 et A2 sont gérés par interruption. Le programme d'interruption doit être le plus court possible: détection de la direction et incrémentation/décrémentation d'un compteur;

B1 et B2 sont gérés par « polling »;

A1 et B1 gère la sélection des paramètres;

A2 et B2 gère l'incrément (sens horaire) et la décrémentation (sens anti-horaire);

Fréquence programmable de 35 MHz à 4.4 GHz.

Pas de programmation en (MHz): 0.0001,0.001,0.01,0.0125,0.025,0.1,1,10,100.

Puissance en (dBm): -4,-1,+2 et +5.

Fonction ON/OFF: OFF A et ON A;

Le premier codeur permet de sélectionner le paramètre à modifier. Le label et le cadre doivent passer en rouge lorsque le paramètre est sélectionné et repasse dans sa couleur initiale lorsqu'il n'est plus sélectionné;

Les grandeurs numériques doivent être rafraichies que lorsqu'elles changent pour éviter le phénomène « flicker ».

Intégration des codeurs optiques

- **Prompt: Demande d'intégration des codeurs**

Ajouter deux codeurs optiques pour régler la fréquence, le pas de fréquence la puissance et le « ON/OFF »;

Pas d'utilisation de librairie;

Les signaux des codeurs sont gérés par interruption et par « polling »;

```
#define PIN_A1 2
```

```
#define PIN_B1 17
```

```
#define PIN_A2 15
```

```
#define PIN_B2 13
```

A1 et A2 sont gérés par interruption;

B1 et B2 sont gérés par « polling »;

A1 et B1 gère la sélection des paramètres;

A2 et B2 gère l'incrémentatation (bouton vers la droite) et la décrémentation (bouton vers la gauche);

Fréquence programmable de 35 MHz à 4.4 GHz.

Pas de programmation en (MHz): 0.001,0.01,0.0125,0.025,0.1,1,10,100.

Puissance en (dBm): -4,-1,+2 et +5.

Fonction ON/OFF: OFF A et ON A;

Le premier codeur permet de sélectionner le paramètre à modifier. Il doit passer en rouge lorsqu'il est sélectionné et repasse dans sa couleur initiale lorsqu'il n'est plus sélectionné;

Fonctionnalités implémentées :

Codeurs optiques :

- **Codeur 1 (A1/B1)** : Sélection des paramètres (interruption sur A1, polling sur B1)
- **Codeur 2 (A2/B2)** : Modification des valeurs (interruption sur A2, polling sur B2)

Gestion des paramètres :

- **Fréquence** : 35.0 MHz à 4400.0 MHz
- **Pas** : 0.001, 0.01, 0.0125, 0.025, 0.1, 1, 10, 100 MHz
- **Puissance** : -4, -1, +2, +5 dBm
- **ON/OFF** : OFF A, ON A

Interface visuelle :

- **Cadre rouge** pour le paramètre sélectionné
- **Couleurs normales** quand non sélectionné
- **Rafraîchissement** en temps réel

Sans librairie externe pour les codeurs :

- Gestion manuelle des interruptions et du polling
- Détection des rotations dans les deux sens

Intégration des codeurs optiques

- Vérification
 - Le paramètre sélectionné est bien en rouge;
 - La sélection et la modification du paramètre s'effectue mais l'incrémentatation et la décrémentation sont inversées;
 - Je n'ai toujours pas écrit une ligne de code;
 - Plus de 400 lignes de code généré par l'IA.



Intégration des codeurs optiques

- Prompt: Demande de correction.

inverser la gestion de l'incrémentation et de décrémentation



Voici le programme avec l'inversion de l'incrémentation et décrémentation :

Comportement maintenant :

- Rotation horaire → Incrémentation (augmente les valeurs)
- Rotation anti-horaire → Décrémentation (diminue les valeurs)

Cette inversion correspond au comportement habituel des encodeurs où tourner vers la droite augmente les valeurs et tourner vers la gauche les diminue.

L'IA résume le développement réalisé

Modifications apportées :

Inversion de la logique dans les interruptions :

Ancienne logique :

```
cpp
if (B1 != A1) {
    encSelectPos++; // Incrémentation
} else {
    encSelectPos--; // Décrémentation
}
```

Nouvelle logique (inversée) :

```
cpp
if (B1 != A1) {
    encSelectPos--; // Décrémentation (inversé)
} else {
    encSelectPos++; // Incrémentation (inversé)
}
```

Même inversion pour les deux codeurs :

- Codeur 1 (sélection) : Logique inversée dans `handleA1()`
- Codeur 2 (valeur) : Logique inversée dans `handleA2()`

Sauvegarde des paramètres dans l'E²PROM

Sauvegarde des paramètres dans l'E²PROM

- **Cahier des charges**
 - Sauvegarder les paramètres Freq, Pas, P et ON/OFF dans l'EEPROM.
 - Pour prolonger la durée de vie de l'EEPROM, la sauvegarde est réalisée toutes les secondes et non pas dès qu'un paramètre est changé.

Sauvegarde des paramètres dans l'E²prom

- **Prompt: Demande d'ajout de la fonction sauvegarde**

Sauvegarder les paramètres Freq, Pas, P et ON/OFF dans l'EEPROM.

Pour prolonger la durée de vie de l'EEPROM, organiser la sauvegarde que tous les secondes et non pas dès qu'un paramètre est changer.



4. Chargement au démarrage :

- Vérification du checksum
- Chargement des paramètres sauvegardés
- Valeurs par défaut si checksum invalide

5. Protection des données :

- Vérification des limites après chargement
- Checksum calculé sur tous les paramètres
- Commit explicite après écriture

Le système prolonge considérablement la durée de vie de l'EEPROM en évitant les écritures continues et en groupant les sauvegardes.

L'IA résume le développement réalisé

Fonctionnalités EEPROM ajoutées :

1. Configuration EEPROM :

- Taille : 512 bytes
- Intervalle de sauvegarde : 1 seconde
- Checksum pour vérifier l'intégrité des données

2. Adresses EEPROM :

- ADDR_FREQ (0) : Fréquence (4 bytes)
- ADDR_PAS_INDEX (4) : Index du pas (4 bytes)
- ADDR_PUISSANCE_INDEX (8) : Index de puissance (4 bytes)
- ADDR_ONOFF_INDEX (12) : Index ON/OFF (4 bytes)
- ADDR_CHECKSUM (16) : Checksum (2 bytes)

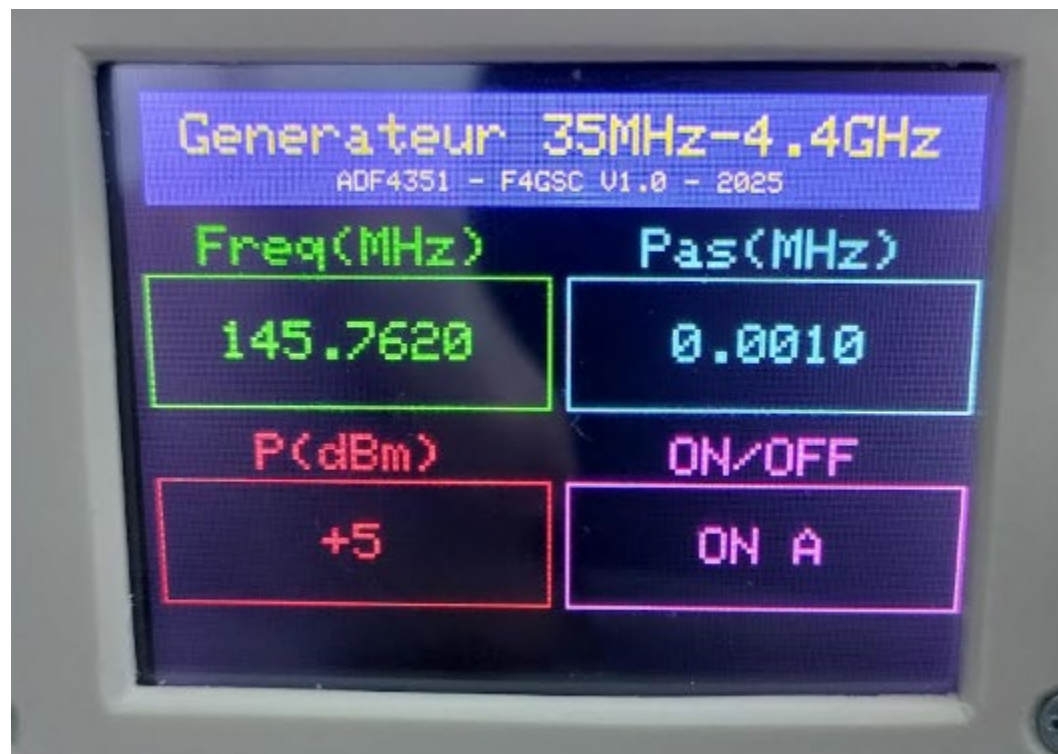
3. Logique de sauvegarde :

- `paramsChanged = true` quand un paramètre est modifié
- Sauvegarde toutes les secondes seulement si changement détecté
- Évite les écritures excessives pour prolonger la durée de vie

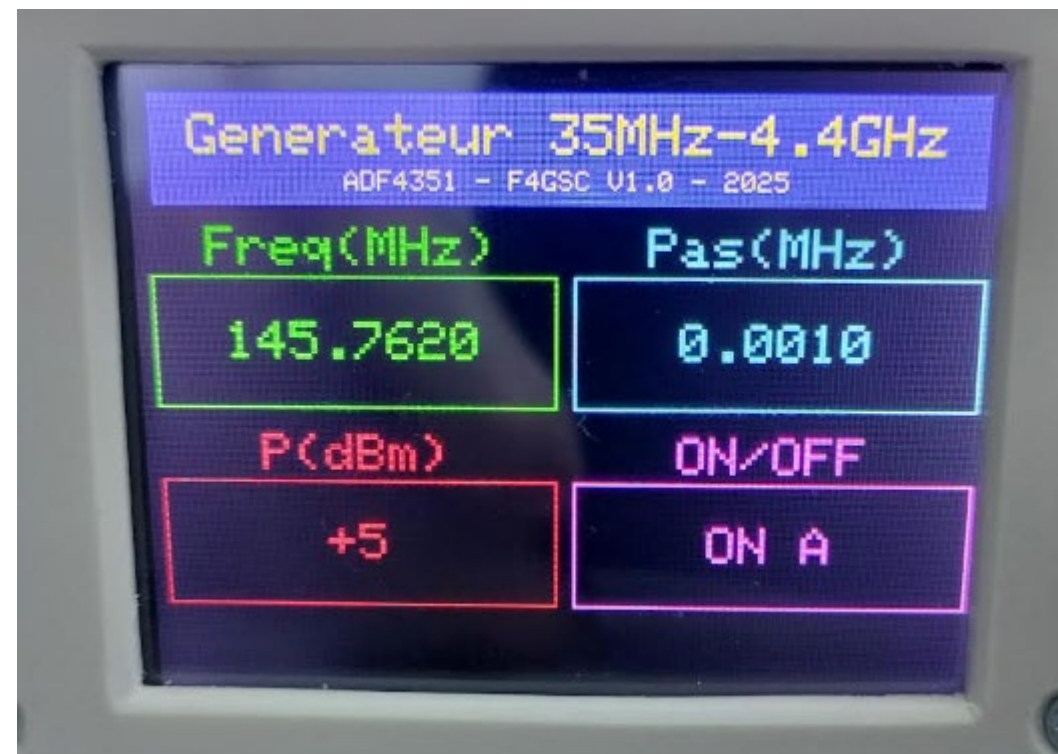
Sauvegarde des paramètres dans l'E²prom

- Vérification
 - Tous les paramètres sont bien restitués y compris celui qui est sélectionné. Ce dernier point n'a pas été demandé dans le cahier des charges.

Avant coupure de l'alimentation



Après coupure de l'alimentation



Intégration de l'ADF4351

Intégration de l'ADF4351

- **Cahier des charges**
 - Ajout du générateur RF ADF4351;
 - Pas d'utilisation de librairie spécifique;
 - Le générateur ADF4351 s'interface à la carte Arduino à partir des broches suivantes:
 - DATA (MOSI), broche 32;
 - CLK, broche 25;
 - LE, broche 26;
 - CE broche 27.
 - Oscillateur 10 MHz;
 - Générer le programme qui permet de piloter l'ADF4351;
 - L'ajouter au programme précédent en le pilotant à partir des paramètres Freq, Pas,P et ON/OFF régler par l'utilisateur.

Intégration de l'ADF4351

- **Prompt: Demande d'ajout de l'ADF4351**
 - Erreur de ma part concernant l'affectation des broches;
 - Correction du cahier des charges.

désolé une erreur sur les broches utilisées. Ajout du générateur RF ADF4351;

Pas d'utilisation de librairie spécifique;

Le générateur ADF4351 s'interface à la Carte Arduino à partir des broches suivantes:

DATA (MOSI), broche 32;

CLK, broche 25;

LE, broche 26;

CE broche 27.

Générer le programme qui permet de piloter l'ADF4351;

L'ajouter au programme précédent en le pilotant à partir des paramètres Freq, Pas,P et ON/OFF régler par l'utilisateur.

Voici le programme complet avec l'ADF4351 corrigé :

L'IA résume le développement réalisé

Broches ADF4351 corrigées :

- DATA : GPIO 32 (identique au TFT_MOSI)
- CLK : GPIO 25 (identique au TFT_SCK)
- LE : GPIO 26 (nouveau)
- CE : GPIO 27 (nouveau)

Fonctionnalités ADF4351 :

- Gestion manuelle SPI sans librairie
- Calcul PLL automatique (INT, FRAC, RFdiv)
- Contrôle puissance (-4, -1, +2, +5 dBm)
- Contrôle ON/OFF sortie A
- Plage fréquence : 35 MHz - 4.4 GHz

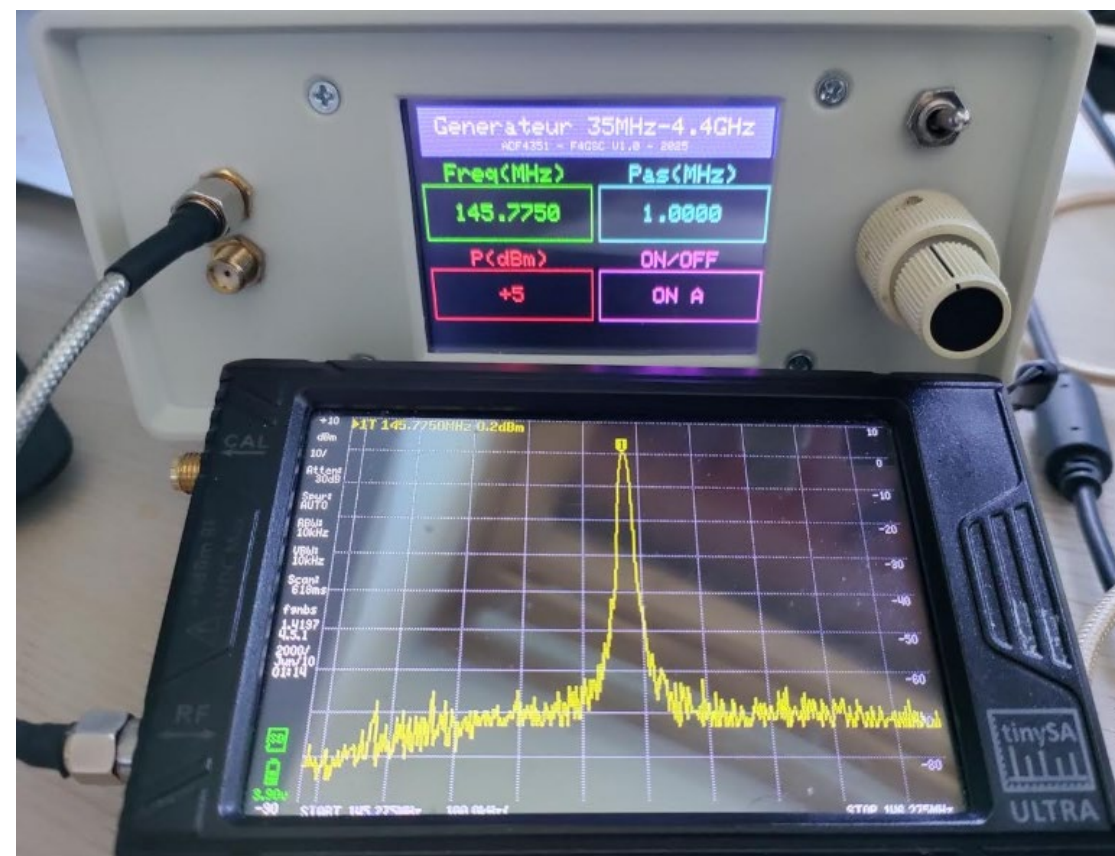
Intégration complète :

- Modification fréquence → mise à jour ADF4351
- Modification puissance → mise à jour ADF4351
- Modification ON/OFF → mise à jour ADF4351
- Modification pas → seulement changement d'incrément

Le programme est maintenant complet avec l'interface utilisateur, les codeurs, la sauvegarde EEPROM et le contrôle du générateur RF ADF4351 !

Intégration de l'ADF4351

- Vérification
 - L'ADF4351 sort bien la bonne fréquence avec la bonne puissance relative (Tiny SA Ultra pas calibré);
 - 578 lignes de code générées par l'IA et qui fonctionne !
 - Je n'ai absolument pas écrit une ligne de code !

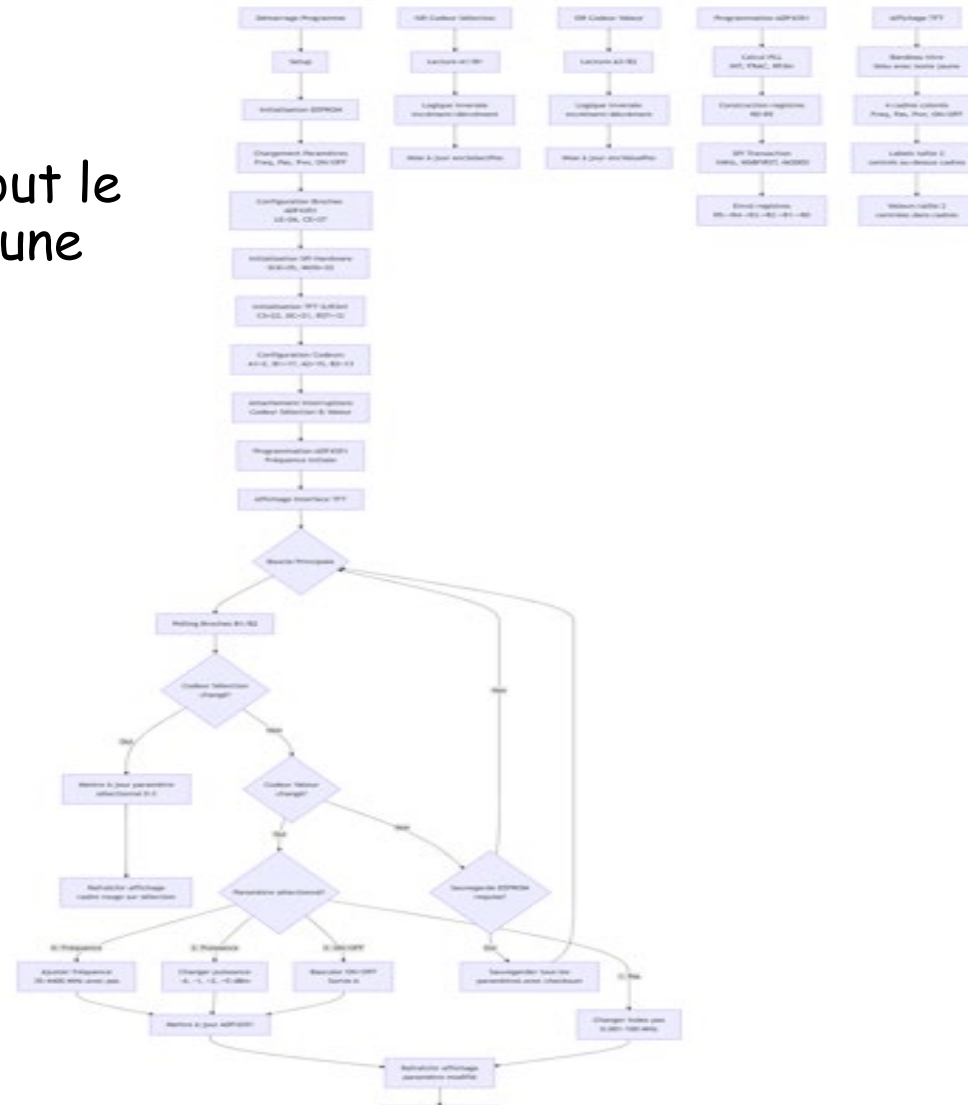


Le plus!!

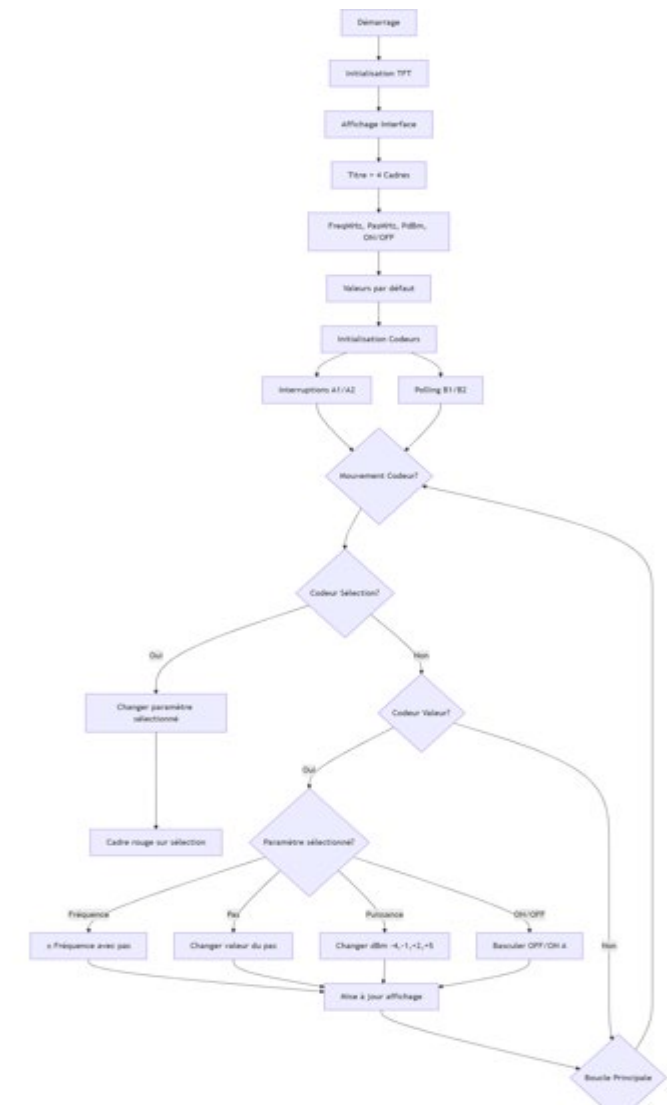
Le plus!!

- Il génère automatiquement l'organigramme de tout le programme ou que d'une partie!

Tout le programme

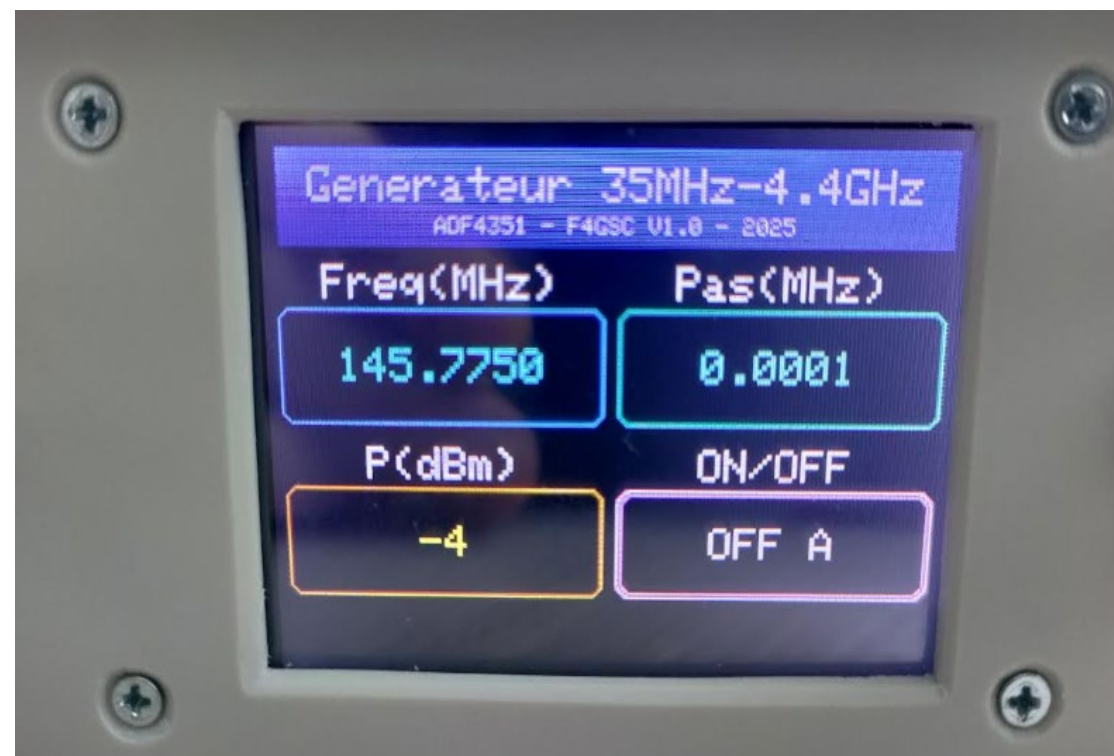
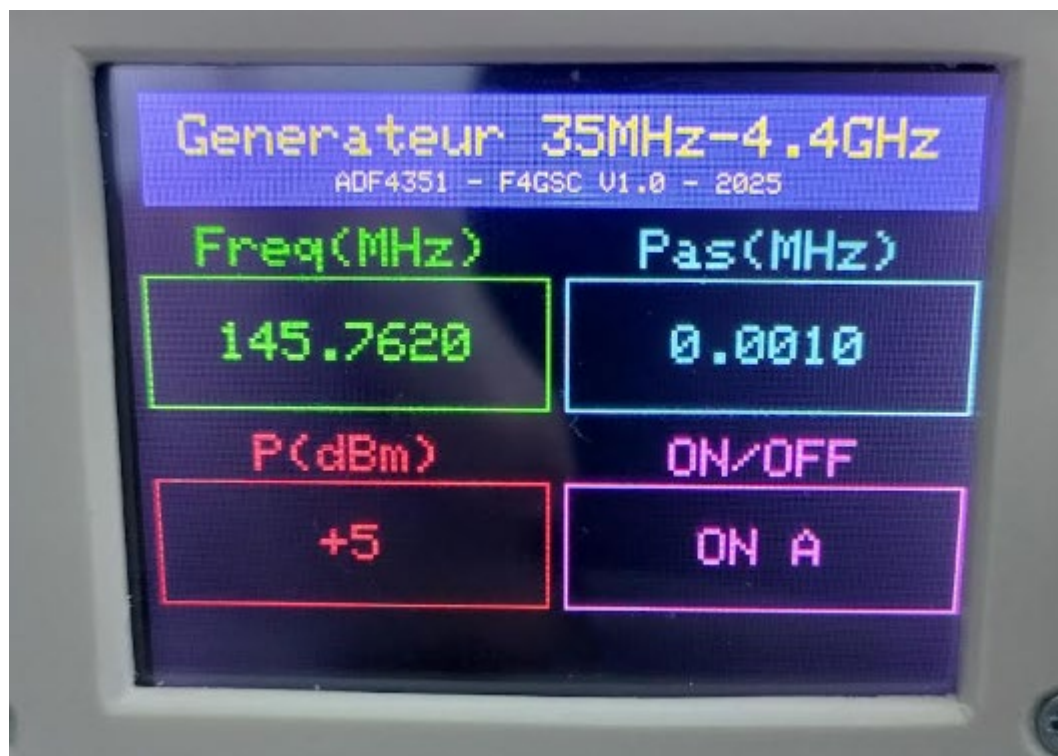


Que de la partie Ecran + codeurs



Le plus!

- Demande
 - Donner un « look » plus moderne au GUI



Génération de la librairie Arduino des fonctions liées à la gestion de l'écran

Génération de la librairie Arduino des fonctions liées à la gestion de l'écran

• Prompt

J générer la librairie Arduino pour toute les fonctions liées à la gestion de l'écran.

3 Exemple_Utilisation.ino (Programme exemple)

Un exemple complet montrant comment utiliser la bibliothèque avec des mises à jour automatiques des valeurs.

4 README_Installation.md (Documentation)

Documentation complète incluant :

- Instructions d'installation
- Configuration matérielle
- API complète de la bibliothèque
- Exemples d'utilisation
- Résolution de problèmes

Fichiers de la bibliothèque

1 ADF4351_Display.h (Fichier header)

Contient :

- Toutes les définitions des couleurs
- Les constantes de positionnement
- La déclaration de la classe `ADF4351_Display`
- Les prototypes de toutes les méthodes publiques et privées

2 ADF4351_Display.cpp (Implémentation)

Contient l'implémentation complète de :

- `begin()` - Initialisation de l'écran
- `drawInterface()` - Affichage de l'interface complète
- `updateFrequency()` - Mise à jour de la fréquence
- `updatePower()` - Mise à jour de la puissance
- `updateStep()` - Mise à jour du pas
- `updateStatus()` - Mise à jour ON/OFF
- `clear()` - Effacement de l'écran
- Toutes les fonctions privées de dessin

Génération de la librairie Arduino des fonctions liées à la gestion de l'écran

- Exemple d'utilisation

```
#include <ADF4351_Display.h>

ADF4351_Display display(TFT_CS, TFT_DC,
TFT_MOSI, TFT_SCK, TFT_RST);

void setup() {
  display.begin();
  display.drawInterface();
}

void loop() {
  display.updateFrequency("146.5250");
  display.updateStatus("ON");
}
```

Conclusions

Conclusions

- Avec l'assistance de l'IA:
 - Pas nécessaire de maîtriser le langage de programmation de la plateforme Arduino;
 - Être précis et détaillé dans la formulation du prompt;
 - Coordonner et organiser le travail de l'IA;
 - Permet de développer des petites applications de quelques centaines de lignes de code;
 - Permet d'apprendre progressivement le langage de programmation par apprentissage et à partir du développement de projets;
 - Génération automatique de l'organigramme et de la documentation;
 - Cela fonctionne aussi avec d'autres langages tel que Python...

Conclusions

- Règles pour utiliser au mieux l'IA:
 - Rédiger un cahier des charges le plus détaillé possible;
 - Décomposer le projet en étapes pour une meilleure vérification;
 - Utiliser les librairies déjà existantes (il les connaît déjà certainement);
 - Demander à l'IA de générer votre propre librairie. Il le fait très bien!!

Oser et franchiser le pas de développer des petits projets Arduino!!