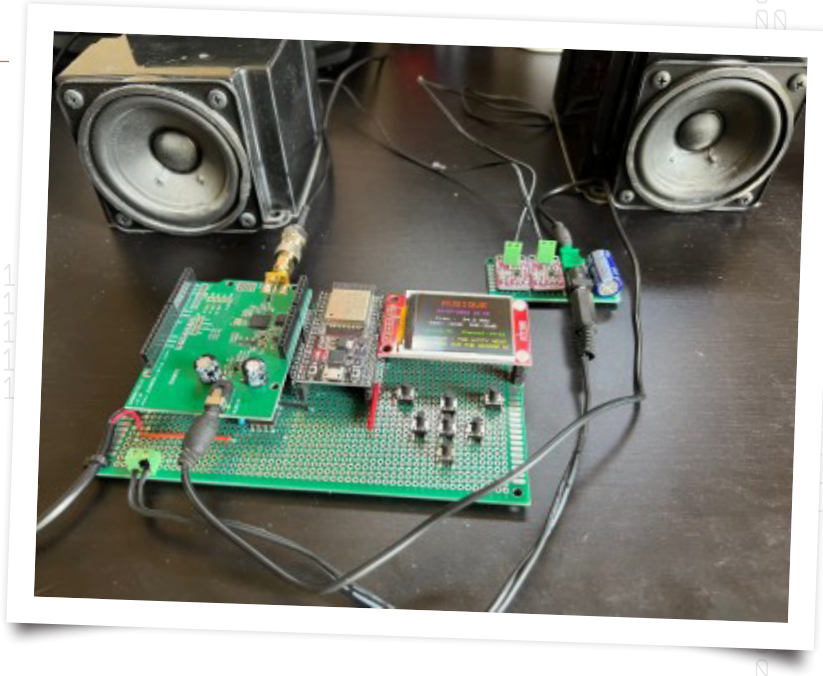
récepteur FM/DAB+

le meilleur des deux mondes



Yves Bourdon (France)

Si vous étiez passionné d'électronique dans les années 1960 et 70, il y a fort à parier que vous vouliez absolument construire deux appareils : un récepteur radio et une horloge numérique. 50 ans plus tard, c'est toujours vrai, mais aujourd'hui, la radio doit aussi être numérique. Cet article vous explique comment la construire avec un ESP32 et une extension Arduino basée sur la puce de réception numérique *Si4684* de *Skyworks*.

J'ai construit ma première radio à moins de 10 ans : Elektor n'existait pas encore et dans la salle d'attente de notre médecin, je suis tombé sur Le *Haut-Parleur*, un magazine principalement axé sur la hi-fi à tubes. Un article qui parlait de diodes, de transistors et de récepteurs GO attira mon attention. J'ai chipé le magasine sans permission aucune. (Cher docteur, veuillez accepter mes sincères excuses).

J'ai mis presque un an pour construire ma première radio, et un matin j'entendis enfin pour la première fois « Nights in White Satin » des Moody Blues. Ma radio comportait une diode de détection, une énorme antenne (à coup sûr pas du tout accordée), un ampli BF (1 transistor OC71) et un petit haut-parleur (HP) emprunté. un condensateur variable faisait l'accord. Ce fut la première d'une longue série de radios de plus en plus complexes, l'ultime étant un récepteur FM entièrement à transistors avec décodeur stéréo. En 2010, Elektor publia la réalisation d'un récepteur radio FM tout numérique basé sur un processeur Skyworks Si4735 de Silicon Labs [1]. Je l'ai bien sûr construite, j'en ai même réalisé plus de cinq versions. Une fois ma région (Aix-en-Provence, France) enfin équipée d'émetteurs DAB+, je commençai à chercher un successeur au Si4735 et tombai sur le *Si4684* (Silicon Labs était devenu *Skyworks*). Ouf ! Je pouvais construire mon récepteur FM/DAB+.

À propos de la norme DAB+

La norme DAB+ (*Digital Audio Broadcasting*) définit la radio numérique terrestre (RNT). C'est l'équivalent radio de la TNT : le signal analogique de la bonne vieille bande FM devient numérique. La DAB+ occupe la bande VHF III (174 MHz à 240 MHz), qui fut utilisée par la télévision analogique. Les principaux avantages de la DAB+ sont :

- Service public gratuit (aucun abonnement requis)
- Aucun bruit de fond (souffle) en cas de mauvaise réception ou d'interférences. La norme DAB+ utilise le codec AAC+ (MPEG-4 partie 3).
- La qualité audio est excellente (bande passante supérieure à celle de la FM, débit de 88 kbps sur la plupart des stations).
- Présélection automatique des stations, comme la TNT. Chaque ville dispose de six à sept multiplex d'environ quinze stations. À Aix-en-Provence, je reçois 83 stations DAB+ !
- Affichage d'informations liées à l'émission en cours (titre, texte défilant, pochette d'album, carte météo, etc. selon les capacités du récepteur).

Le moins : la réception est faible en intérieur et une bonne antenne est nécessaire. J'utilise une antenne ¼ d'onde à polarisation verticale (34,5 cm) installée près d'une fenêtre.

Le Si4684

J'avais prévu de concevoir un circuit imprimé (PCB) pour la puce Si4684. Cependant, en étudiant de près la fiche technique, j'ai réalisé que le micrologiciel du composant (généralement stocké dans une EEPROM série externe) était protégé par des droits de propriété intellectuelle (*PI*). L'utilisation de ce micrologiciel requiert

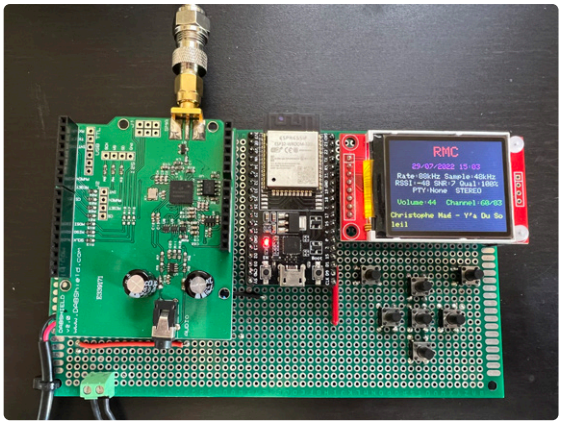


Figure 1. Mon prototype. Basé sur un module ESP32-DevKitC, cet ensemble tient dans un boîtier transparent (Hammond 1591C).

la signature d'un accord avec Skyworks - le fabricant de la puce - moyennant une redevance.
Par conséquent, pour un individu comme moi, obtenir le micrologiciel légalement est impossible. Même si des copies pirates existent en ligne, mon passé d'ingénieur d'études ayant toujours défendu la notion de *PI* m'interdisait d'en utiliser. Je cherchai donc un produit du commerce disponible auprès d'un fabricant approuvé par Skyworks.
C'est ainsi que je retins le *DAB Shield* vendu par *AVIT Research* (société anglaise). Adrian, PDG de la société, m'autorisa à utiliser son excellente bibliothèque *DABShield* pour mon projet. Je tiens ici à le remercier pour son soutien dans ce projet !

Le circuit du récepteur FM/DAB

Pour mon prototype, j'ai utilisé un *Espressif ESP32-DevKitC* (**figure 1**). Le schéma (**figure 2**), vous montre que presque toutes les broches d'E/S disponibles sont prises. J'ai choisi le module ESP32 aussi pour sa puissance de calcul et surtout pour sa mémoire disponible (512 ko de RAM et 4 Mo de mémoire flash).

Problème d'EEPROM

Malgré toute cette mémoire, j'ai dû ajouter une EEPROM externe. C'est parce qu'au début du projet, j'ai utilisé la bibliothèque *EEPROM* de l'ESP32 pour stocker les stations FM et DAB+. Comme l'ESP32 ne

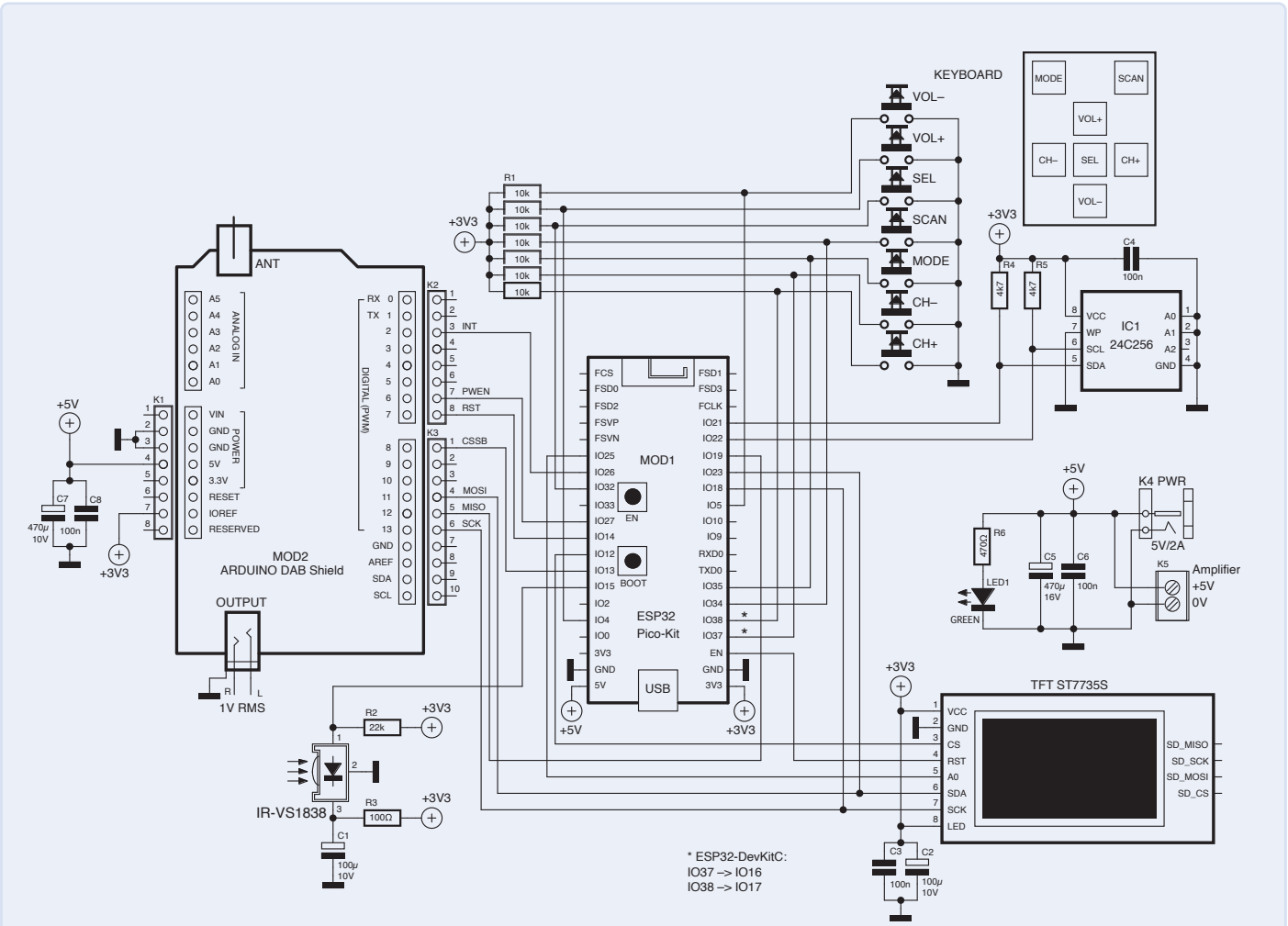


Figure 2. Schéma du récepteur FM/DAB+.

possède pas de vraie EEPROM, cette bibliothèque utilise en partie la mémoire flash pour émuler une EEPROM. Malheureusement, cette bibliothèque devient progressivement obsolète et ne fonctionne bien que si la mémoire ne dépasse pas 512 octets. (J'ai bien essayé une taille max. de 4 ko comme suggéré sur certains forums, mais le programme plante inévitablement car les données de l'EEPROM sont modifiées assez aléatoirement). Mieux vaut utiliser la bibliothèque *Preference* à la place.
Cependant, j'ai découvert les problèmes de la bibliothèque *EEPROM* alors que la majeure partie de mon programme était déjà prête et je ne voulais pas tout réécrire avec la bibliothèque *Preference*. J'ai donc opté pour une EEPROM externe de 32 ko (24C256) gérée par la bibliothèque *SparkFun_External_EEPROM*. Cette puce (environ 50 cts) utilise le bus I²C de l'ESP32 (SDA sur IO21 et SCL sur IO22). Relier ses broches A0 et A1 au 0 V, fixe l'adresse I²C du dispositif à 0xC0. La broche A2 peut rester non connectée.
J'avais prévu de stocker au plus 250 stations FM et 250 stations DAB+. Pour chaque station, on stocke le nom (huit caractères maximum pour la FM ; 16 pour la DAB+), la fréquence FM, le multiplex DAB+ et le numéro. En outre, 10 octets sont nécessaires pour mémoriser la dernière station écoutée : volume, mode (FM ou DAB+), fréquence FM, multiplex DAB+ et canal. Une taille mémoire min. de 10 + 13 × 250 (FM) + 20 × 250 (DAB+) est donc nécessaire, soit 8260 octets, ou 66 080 bits. Une EEPROM de 16 ko de type 24C128 aurait été suffisante.

Écran et clavier

Comme écran du récepteur, j'ai utilisé un écran couleur TFT 160 × 128 à base de ST7735S avec bus SPI. Il se gère facilement avec les excellentes bibliothèques *Adafruit_ST7735* et *Adafruit_GFX*. Cet écran mobilise quatre broches IO de l'ESP32 : 10 (CS),18 & 23 (MOSI & SCK du port SPI), 25 (A0). Le driver Adafruit peut piloter le signal RST, mais pour économiser un port, j'ai mis `TFT_RST` à -1 (dans la déclaration *Screen Connections* du logiciel) et l'ai relié directement à la broche de reset de l'ESP32 (EN).
Notez que le marquage SDA et SCL sur l'écran (reproduit sur la **figure 2**) est assez confus car c'est bien le bus SPI (sans MISO) qui pilote l'écran.
C2 et C3 découplent l'alimentation de l'écran ; le rétro-éclairage est toujours allumé (broche LED connectée à +3,3 V).
Sept boutons contrôlent ma radio : Volume haut/bas (VOL+/VOL-), Canal haut/bas (CH+/CH-), Sélection (SEL), Mode et Balayage. Ils sont polarisés au rail d'alimentation +3,3 V par des résistances de 10 kΩ.

DevKitC ou Pico-Kit ?

Le schéma montre le très petit module ESP32-Pico-Kit, mais un ESP32-DevKitC peut aussi être utilisé, comme je l'ai fait sur mon prototype. Cependant, l'ESP32-DevKitC n'expose pas IO37 et IO38. À la place, il faut utiliser IO16 et IO17, eux indisponibles sur l'ESP32-Pico-Kit. Il faut donc sélectionner le bon module dans le logiciel avant compilation (lignes 76 et 77 à l'heure actuelle). Pour le DevKitC, utiliser :

```
//#define ESP32_PICO
#define ESP32_DEVKIT
```

Pour le Pico-Kit, utiliser :

```
#define ESP32_PICO
//#define ESP32_DEVKIT
```

Télécommande

Le circuit prend en charge une télécommande infrarouge via un récepteur IR VS1838. Le réseau R3/C1 filtre l'alimentation du dispositif pour éviter d'endommager des trames sur sa sortie reliée à IO15 et polarisée par une résistance de 22 kΩ). La prise en charge logicielle de la télécommande n'était pas prête au moment de la rédaction de cet article, mais le sera peut-être lorsque vous lirez ces lignes.

L'extension DAB

L'Arduino M0 était le compagnon idéal de l'extension DAB, mais il a été abandonné. C'est pourquoi, ce projet utilise un module ESP32 à la place. L'ESP32 peut facilement contrôler l'extension DAB sept signaux suffisent.
Même si l'extension DAB nécessite une alimentation 5 V, ses lignes d'E/S sont compatibles 3,3 V. Comme l'ESP32 utilise 3,3 V, les deux peuvent être connectés sans décalage de niveau. Il faut connecter la broche IOREF de l'extension au 3,3 V pour assurer cette compatibilité.
L'extension DAB communique avec l'ESP32 via le bus SPI (MOSI, MISO et SCK). La bibliothèque *DABShield* ne permet que de changer l'affectation du signal CSSB (j'ai pris IO13). Connecter respectivement les signaux INT, PWEN et RST aux ports IO26, IO27 et IO14.

Alimentation électrique

L'alimentation +5 V du récepteur FM/DAB+ entre par le connecteur K4. Il n'y a pas de protection contre l'inversion de polarité, donc prudence (vous pouvez en ajouter une). C5 et C6 filtrent en partie le bruit et la LED1 (verte) sert de témoin d'alimentation.
Un bornier à vis à deux voies (K5) est prévu pour alimenter une carte d'amplification (option, voir ci-dessous). Sans amplificateur externe, une alimentation de 5 V/500 mA suffit car nous n'utilisons pas le wifi de l'ESP32, gourmand en énergie. En revanche, avec l'amplificateur optionnel, il faut une alim 5 V d'au moins 2 A.

Carte d'amplificateur (option)

Positivement surpris par la qualité sonore de la console de jeu Nintendo Wii, j'enquêtai un peu et appris qu'elle utilisait un PAM8302 3 WRMS, un ampli en classe D. Celui-ci se trouve facilement en ligne monté sur une petite carte, et j'ai donc connecté deux de ces modules au récepteur FM/DAB+ (voir **figure 3** et **4**). La qualité du son est étonnante (avec de bons HP, bien sûr).
La sortie audio de l'extension DAB se connecte à l'ampli par un

câble court muni d'un jack stéréo de 3,5 mm à chaque extrémité. Les boutons VOL+ et VOL- contrôlent le volume.

L'environnement de développement logiciel

Pour développer le logiciel du récepteur FM/DAB+, j'ai utilisé l'EDI Arduino. Après l'avoir configuré pour le module ESP32, assurez-vous de sélectionner les paramètres suivants :

- Module de développement ESP32
- Fréquence CPU de 240 MHz
- Schéma de partition : par défaut 4 MB avec SPIFFS

Il faut installer quelques bibliothèques externes ; j'ai indiqué dans le code source où les obtenir :

- *DABShield*
- *Adafruit_GFX*
- *Adafruit_ST7735*
- *SparkFun_External_EEPROM*

Enfin, après sélection du bon port série, la compilation et le téléchargement vers le module ESP32 doivent fonctionner.

Le logiciel

Avant d'aller plus loin, sachez que l'écriture de logiciels n'est pas mon cœur de métier. (Je suis ce que l'on appelle un ingénieur matériel.) Je remercie donc par avance tous ceux qui, s'intéressant à mon projet, veulent et peuvent étendre mon travail ou refaire ce que j'aurais pu mieux faire. Cela dit, mon programme (v1.63 au moment d'écrire ces lignes [2]) est très fiable sans aucun dysfonctionnement à ma connaissance.

Le logiciel est assez largement documenté, et après avoir lu ce qui suit, il devrait être assez facile à comprendre. Il est payant de surveiller la sortie du port série de l'ESP32 car il reçoit beaucoup d'informations de débogage. À la mise sous tension, l'écran affiche l'état de connexion des composants du Si4684 et de l'EEPROM série (**figure 5**). S'il y a un problème, un message d'erreur s'affiche et le programme se bloque.

Formatage de l'EEPROM

Il faut formater l'EEPROM avant le tout premier lancement du programme. À cet effet, appuyez sur le bouton SEL tout en allumant le récepteur. Dès que l'écran affiche « *Init EEPROM …* », vous pouvez libérer le bouton SEL. L'initialisation dure environ quinze secondes.

L'interface utilisateur

Je voulais une interface utilisateur réactive et conviviale. Les boutons sont scrutés toutes les 200 ms. Cela permet d'effectuer des frappes répétitives en maintenant le bouton enfoncé et d'éviter les rebonds de contact (variable `debounceDelay`).

Le bouton Mode indique au logiciel de basculer entre FM et DAB+ (`dabMode` dans le logiciel). S'il n'y a pas de station enregistrée dans le mode sélectionné, le logiciel demande d'appuyer sur le bouton Scan (**figure 6**).

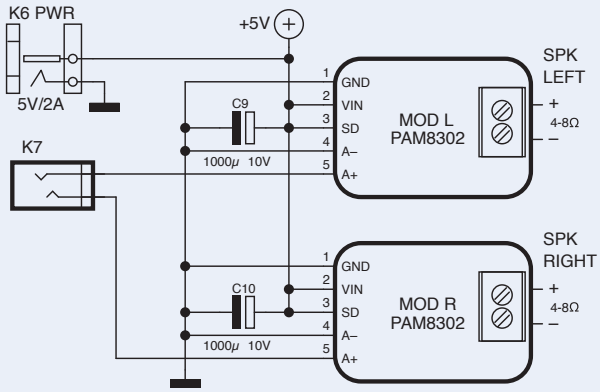


Figure 3. Schéma de la carte d'amplification optionnelle.

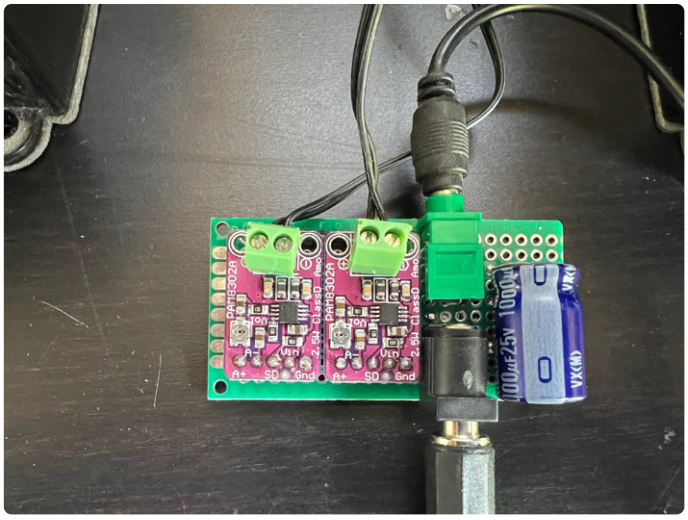


Figure 4. La carte d'amplification optionnelle réalisée carte de prototypage.

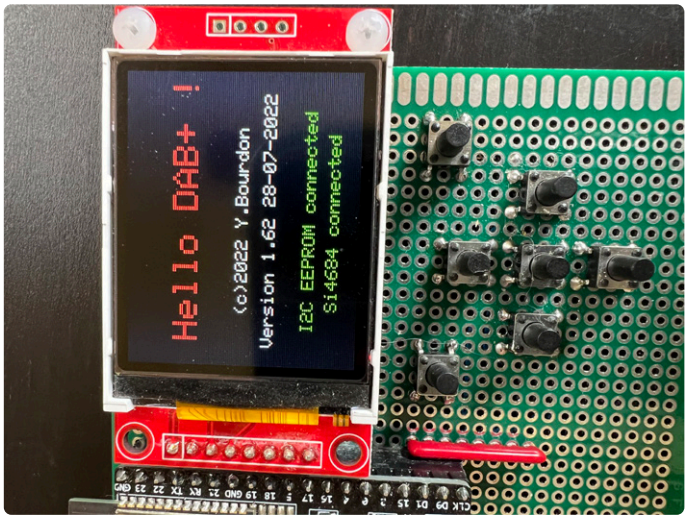


Figure 5. La version du logiciel et l'état de la connexion du matériel sont indiqués.

Figure 6. Ici nous voyons que le multiplex MARSEILLE 8A est trouvé et qu'il contient 13 stations. HELLO PROVENCE est la première.

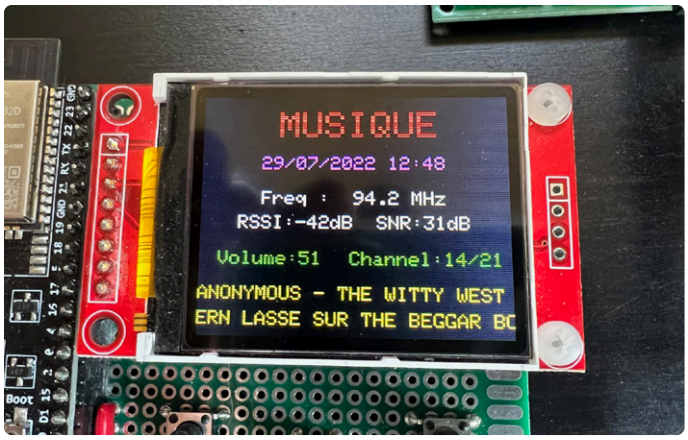


Figure 7. De nombreuses informations sont affichées sur le petit écran couleur.

Tri des stations

En mode DAB+ (`DAB_scan` dans le programme), tous les multiplex sont balayés, et si des stations sont trouvées (RSSI ≥ 30 dB), elles sont mémorisées.

Même si chacun des multiplex envoie les stations par ordre alphabétique, il faut retrier l'ensemble des stations trouvées (`sortStations` dans le programme). Parfois, deux stations identiques sont trouvées sur deux multiplex différents, mais comme les caractéristiques de réception ne sont pas les mêmes, j'ai décidé de ne rien filtrer (avec 83 stations, je n'ai vu cela qu'une fois). Si le nom d'une station n'est pas trouvé (par ex. parce que le champ est resté vide), le programme le remplace par « Unknown? ».

Pour des raisons de performance, les stations sont d'abord stockées et triées en RAM avant d'être copiées dans l'EEPROM (`saveDABchannelToEEPROM`).

Balayage FM

En mode FM, les choses sont un peu plus complexes. En effet, le balayage est très rapide (`FMscan`) et toutes les fréquences entre 87,5 MHz et 108 MHz présentant un RSSI ≥ 30 dB sont mémorisées. Cependant, aucun nom de station n'est trouvé à ce stade. Par conséquent, à la fin du balayage, les fréquences mémorisées sont sélectionnées une à une, et les données RDS renvoyées par la station (heure et nom de la station, entre autres) sont examinées. Si un nom est trouvé, il est stocké en RAM avec la fréquence. Au bout de 30 s, si aucun nom n'est trouvé, le système affiche « Unknown ».

À Aix-en-Provence je reçois 27 stations FM. Avec jusqu'à 30 s par station ce processus peut être long : le bouton Scan permet de l'interrompre à tout moment.

Lorsque la recherche de nom se termine (normalement ou sur interruption), un tri alphabétique est lancé dans la RAM et le numéro de la station, la fréquence et le nom sont stockés dans l'EEPROM (`addStation`).

Lorsqu'une station est sélectionnée, son statut RDS est relu (`Dab.task`), et si le nom trouvé ne correspond pas au nom en mémoire, alors ce dernier est enregistré dans l'EEPROM. Cela se produit lorsque, par exemple, « Unknown? » s'affiche parce que le nom n'a pas été trouvé pendant le scan ou parce que le nom RDS a changé. (Parfois certaines stations utilisent malencontreusement le champ du nom pour afficher le titre du morceau alors que le RDS fournit le champ de données idoine !)

En outre, en FM comme en DAB+, vous pouvez faire défiler les noms des stations (en maintenant l'appui sur CH+ ou CH-) et sélectionner celle que vous avez choisie avec le bouton SEL (`FMsetChannel` ou `DAB_SetChannel`). L'écran affiche les détails de la station reçue, dont l'heure et des informations supplémentaires (rafraîchies toutes les 30 s, **figure 7**).

À chaque mise sous tension, l'EEPROM est lue (`lastEEPROM`) pour retrouver la dernière station écoutée (DAB+ ou FM) et son volume d'écoute.

J'ai délibérément choisi de ne pas implémenter de menu « Ajouter une station ». Mieux vaut attendre que j'écrive la partie télécommande où je prévois d'ajouter, entre autres, cette fonctionnalité.

Radio complète

J'ai finalement réussi à construire la radio que je voulais. Elle est très pratique à utiliser et trône fièrement sur mon bureau. Sans avoir reçu d'instructions de ma part, ma famille l'utilise aussi, ce qui est bon signe. Et cela nous a rapidement fait réaliser que chacun écoutait presque toujours *sa* station (heureusement que j'ai alloué une mémoire pour 500 stations).

Je tiens à remercier Adrian d'AVIT Research, mon ami Stephan Calderoni qui m'a aidé pour la fonction de tri des noms de stations et Elektor pour la publication de cet article et pour avoir redessiné mes schémas faits à la main. Merci également à Adafruit pour la publication de toutes leurs sources et schémas.

J'espère que cet article m'a permis de partager avec certains lecteurs la passion qui m'anime depuis mon plus jeune âge et qui perdure aujourd'hui, même à 64 ans.

(220249-04) — VF : Yves George

DAB+ en France

En France (à la traîne des pays voisins), la diffusion DAB+ est autorisée depuis le 15 juillet 2021, et 40 % de la population française devrait pouvoir recevoir la DAB+ en 2023. La couverture DAB+ actuelle de la France est plutôt faible, surtout quand on sait que la compatibilité DAB+ est obligatoire depuis décembre 2019 pour les autoradios vendus seuls et depuis juillet 2020 sur ceux des voitures neuves. Cela va certainement accélérer le déploiement. La FM disparaîtra probablement progressivement en France comme c'est déjà le cas en Suisse, en Angleterre et en Norvège.

Pour voir la disponibilité du DAB+ dans votre région, vous pouvez utiliser le site : www.csa.fr/Ma-radio-DAB-Plus.

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (yb.electronique@orange.fr) ou contactez Elektor (redaction@elektor.fr).

PRODUITS

- **ESP32-Pico-Kit (SKU 18423)**
  www.elektor.fr/18423
- **Kit Raspberry Pi RTL-SDR d'Elektor (SKU 19518)**
  www.elektor.fr/19518

LIENS

[1] « La radio DSP d'Elektor », B. Kainka, Elektor 7/2010 : https://www.elektormagazine.fr/magazine/elektor-201007/11614

[2] Téléchargements et mises à jour : https://www.elektormagazine.fr/labs/radio-dab