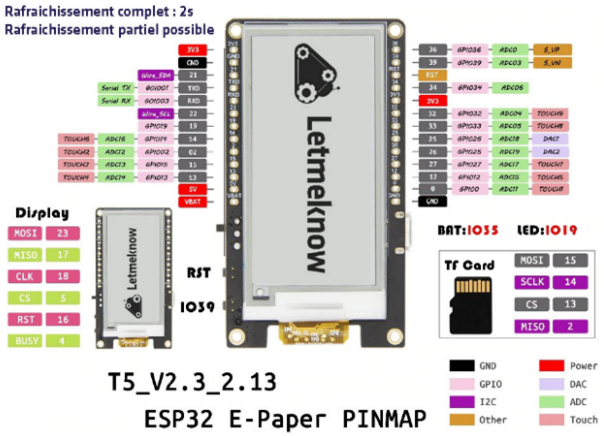
Badge statique E. Paper pour salons OM

Edgar Rouquet F5FDR

INTRODUCTION

Pour certains d'entre-vous, la technologie du « E. Paper » est sans doute inconnue et pour vous la faire découvrir, quoi de mieux qu'un exemple d'application que vous pourrez mener à bien sans trop de complication ? J'ai découvert il y a quelque années ce type d'afficheurs lors d'un salon professionnel et j'ai trouvé cette technologie vraiment bluffante. Avoir un afficheur qui conserve les données affichées même sans alimentation peut nous être très utile lorsque la consommation électrique est critique sur certaines de nos réalisations. En fait, on parle ici de la technologie de « l'encre électronique » très utilisée par les « liseuses électroniques » ou appelées aussi « Ebook ».

Aujourd'hui on trouve de petits afficheurs à un QSJ abordable, un peu plus de 10 euros. Il suffit de se rendre sur certains sites Internet et de faire son choix. Pour cet article, j'ai retenu un type d'afficheurs trouvé sur un site chinois (les divers liens sont proposés en fin d'article). Le fabricant et la référence sont « LilyGO T5 V2.3.1 ». C'est un afficheur de « 2,3 » que vous croisez régulièrement lorsque vous vous rendez dans certaines grandes surfaces commerciales. Ils sont généralement utilisés pour indiquer les prix des articles mis en vente.



La carte électronique proposée avec son afficheur est très performante et elle peut être utilisée par un grand nombre d'applications OM. Dans cet article, nous nous contenterons d'une application très simple mais ludique, accessible à tous.
Le but étant :

- De permettre au plus grand nombre d'entre-vous de réaliser ce gadget.
- De vous simplifier au maximum la programmation des textes à afficher tout en vous laissant le choix de la configuration finale de l'affichage.
- De ne pas vous obliger à écrire du « code de programmation » tout en vous donnant accès à cette réalisation, même si vous n'avez aucune compétence en la matière.
- De supprimer une bonne fois pour toute le problème d'autonomie rencontré sur certains types de badges électroniques.

LES ÉTAPES

Malgré toute ma volonté de vous simplifier au maximum cette réalisation, certaines étapes vous seront, pour certains, inconnues mais elles sont nécessaires afin de mener à bien ce projet. Cet article est là pour vous guider. En effet, l'afficheur est livré programmé avec un logiciel de démonstration. Il faudra donc lui « injecter » un autre programme pour qu'il puisse nous être utile et répondre à nos besoins. Heureusement il est compatible avec l'écosystème « Arduino » très prisé parmi les radioamateurs. Les caractéristiques de l'ensemble sont :

- Carte de développement compatibles « ESP32 Dev module ».
- Afficheur E. Paper de 2,3 pouces (250 x 122 pixels) blanc/noir.

- Interface Wi-Fi (non utilisée ici).
- Interface carte µSD.
- Connecteur pour batterie lithium 3,7 V (avec électronique de charge intégrée sur la carte, batterie non fournie et pas nécessaire ici).
- Interface série USB. Attention au modèle que vous commanderez. De préférence, prenez celui que j'ai utilisé pour cet article.
- Alimentation 3,3 V.
- Poids : 11 gr.
- Taille rectangulaire 66 mm x 37 mm x 6 mm (hors connecteur USB et bouton ON/OFF).
- 1 x bouton poussoir de RESET.
- 1 x bouton poussoir programmable.
- 26 plots à souder pour applications personnelles (non utilisés ici).

Le fichier « badge.zip » disponible en téléchargement contient tous les fichiers nécessaires à la réalisation. Dans un premier temps et pour ceux qui n'ont jamais fait de réalisation à base de systèmes « Arduino », il vous faut télécharger l'IDE Arduino et l'installer (Environnement de Développement Intégré). C'est l'outil obligatoire pour « injecter » le programme dans l'afficheur et pour vous permettre de modifier à volonté les paramètres par défaut.

https://www.arduino.cc/en/software

Le lien ci-dessus permet le téléchargement correspondant à votre système d'exploitation (PC, Linux, Mac). La version 2.0.3 est la dernière version disponible, malheureusement et pour le moment cette version ne prend pas en compte correctement notre carte « ESP32 Dev module ». Il faut donc se rabattre sur la version précédente appelée « Legacy IDE (1.8.X) » disponible sur la même page. Pour ne pas surcharger cet article, l'installation et la configuration de l'IDE sont décrites dans un autre document proposé en téléchargement en fin de cet article.

Je ne vais décrire que les étapes pour des PC Windows mais le principe est le même pour les versions Linux et Mac.

LA COMPILATION

Une fois l'IDE installé et correctement configuré, un nouveau répertoire «Arduino» a été créé dans votre répertoire « {utilisateur}\Arduino ». C'est dans ce répertoire qu'il vous faut décompresser le fichier « badgeV2.zip ». Vous devez obtenir un répertoire supplémentaire : « {utilisateur}\Arduino\BadgeV2 ».

Déballez votre afficheur et à l'aide d'un cordon USB adapté, connectez-le à votre PC. Mettez sous tension l'afficheur en basculant le petit interrupteur prévu à cet effet. Au bout de quelques secondes un « Port COM » supplémentaire doit apparaître. Notez-le, vous en aurez besoin pour la programmation.

Si tout se passe bien, votre afficheur doit faire apparaître des images et des textes de démonstration. Nous allons maintenant supprimer ce programme de démo en le remplaçant par celui de notre badge, mais avant nous devons nous assurer que la compilation du logiciel s'effectue correctement. Dans le répertoire « BadgeV2 » faites un « double-clic » sur le fichier « BadgeV2.ino ». Votre IDE Arduino doit démarrer et les fichiers du projet seront chargés. Si vous aviez déjà une instance de l'IDE active, fermez-la, elle ne vous servira à rien.

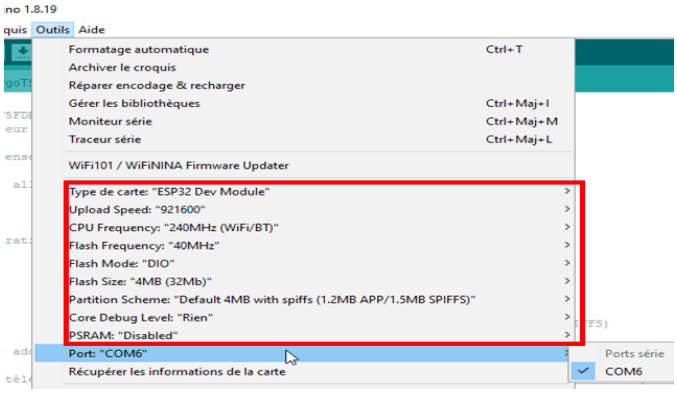
Le bouton « Vérifier » va nous permettre de tester le bon déroulement de la compilation du programme. Pour les néophytes, une compilation permet de convertir le programme « texte » écrit dans un langage spécifique en un bloc de données binaires compatibles avec notre cible « LiliGO T5 ». Cliquez sur le bouton « Vérifier », la compilation doit s'exécuter.

Patientez, la première compilation est relativement longue, plusieurs minutes mais à la fin vous devez obtenir le message suivant :

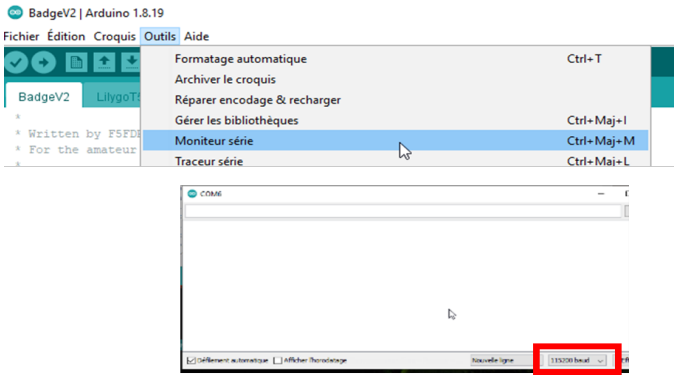
« Le croquis utilise 343138 octets (26 %) de l'espace de stockage de programmes. Le maximum est de 1310720 octets. Les variables globales utilisent 19128 octets (5 %) de mémoire dynamique, ce qui laisse 308552 octets pour les variables locales. Le maximum est de 327680 octets. »

Si vous n'obtenez pas ce message et qu'à la place vous avez un certain nombre d'erreurs, il vous faut revoir l'installation de l'IDE et surtout vérifier que vous avez bien ajouté toutes les librairies supplémentaires requises par le projet. Reportez-vous au document d'installation de l'IDE.

Avant d'injecter le programme dans la cible (téléverser) il vous faut définir le « port COM » dans l'IDE. Utilisez les menus « Outils », « Port COM(x) » afin de valider celui qui a été ajouté par notre afficheur. Dans notre exemple c'est le « COM6 » :



Les autres paramètres spécifiques à la carte ESP32 (rectangle rouge) ont, normalement, été correctement définis lors de l'installation de l'IDE. Pour nous permettre de suivre le bon fonctionnement du programme compilé, vous devez activer le terminal série qui vous permettra d'afficher les informations de « debug » :

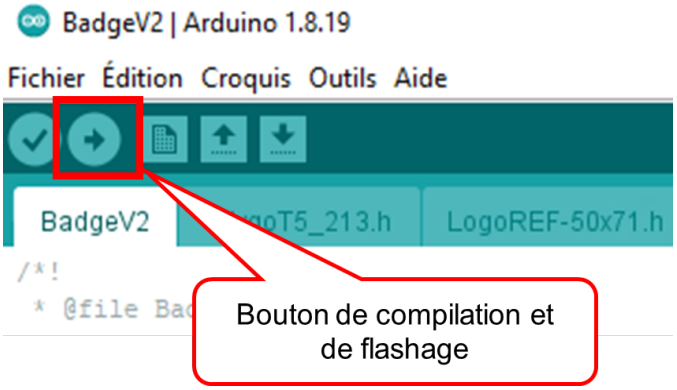


N'oubliez pas de définir la vitesse de la liaison série à 115200 bds.

Le moment est venu de programmer l'afficheur et de lui faire écrire les textes définis par défaut. L'étape est la même que celle de la vérification mais elle sera suivie automatiquement du « flashage » des données du programme dans la mémoire « flash » de notre ESP32.

LE FLASHAGE ET LE CONTRÔLE

Cliquez sur le bouton de « flashage » (Téléverser) pour relancer une compilation qui sera bien plus rapide puisque déjà effectuée une fois et qui enchaînera par la programmation du fichier binaire dans la cible.



Notez qu'il faut maintenir l'afficheur sous tension et connecté au PC pour que toutes les phases soient effectuées correctement.

La fin de la programmation sera matérialisée par le message suivant :

...

Writing at 0x00030000... (90 %)

Writing at 0x00034000... (100 %)

Wrote 349360 bytes (156154 compressed) at 0x00010000 in 3.0 seconds (effective 931.6 kbit/s)...

Hash of data verified.

Compressed 3072 bytes to 128...

Writing at 0x00008000... (100 %)

Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds...

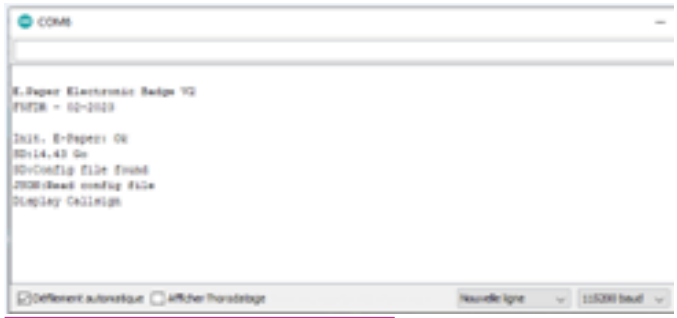
Hash of data verified

Leaving...

Hard resetting via RTS pin...

Si tout s'est bien passé, le « flashage » se termine par un « RESET » de notre afficheur qui doit se mettre à clignoter plusieurs fois correspondant à l'effacement du précèdent message.

Une ou deux secondes après que l'afficheur ait été complètement effacé, le texte par défaut identique à l'image ci-dessous doit apparaître sur le e.paper :

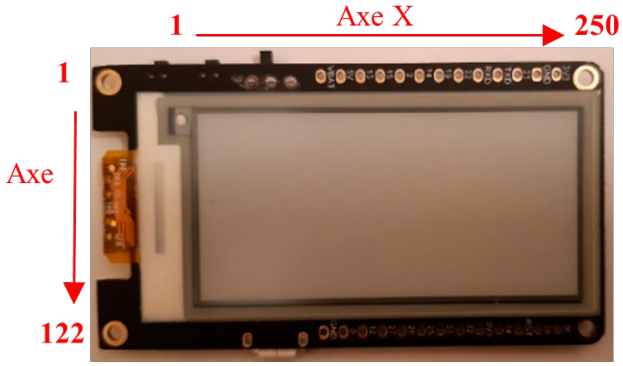


Affichage du mode « debug »

Le plus compliqué étant réalisé, maintenant vous êtes prêt à aborder la partie modification / adaptation des textes pas défaut.

ÉDITION DES TEXTES

Avant toute modification il est nécessaire de rappeler comment le positionnement des textes a été pris en compte par le programme :

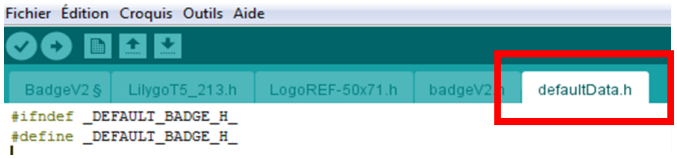


Les axes X et Y sont définis comme indiqués sur l'image ci-dessus et il vous sera possible de positionner vos textes en tenant compte de ces deux axes.

Il existe deux possibilités de définir vos propres textes :

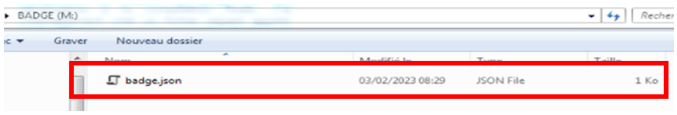
- Par compilation du programme.
- Par l'utilisation d'une carte SD contenant un fichier au format « JSON ».

La première solution vous est accessible en utilisant l'IDE Arduino. Lorsque vous avez sélectionné le fichier « BadgeV2.ino », l'IDE vous a ouvert plusieurs fichiers textes. Ces fichiers composent la totalité du projet et sont nécessaires à la compilation finale. J'ai volontairement créé un fichier séparé appelé « defaultData.h » pour vous permettre facilement de l'identifier et vous autoriser la modification et la création de vos textes. Dans ce fichier, vous trouverez toutes les informations vous permettant de définir vos propres textes à afficher.

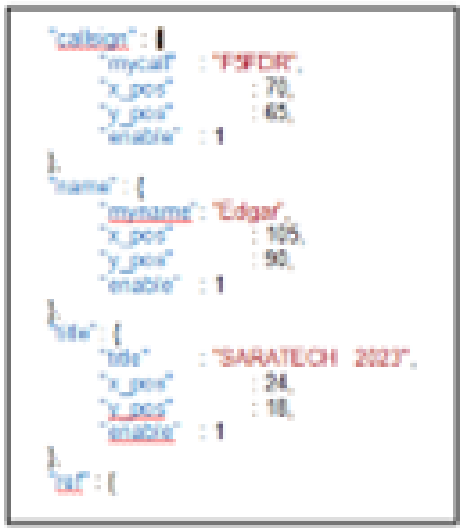


Une fois les modifications effectuées, faites-en une sauvegarde en utilisant le menu « Fichier » et « Enregistrer » (ou utilisez la séquence « CTRL+S »). Ensuite, relancez la compilation décrite au chapitre « Flashage et contrôle ». Après quelques dizaines de secondes vos modifications doivent être prises en compte sur l'afficheur.

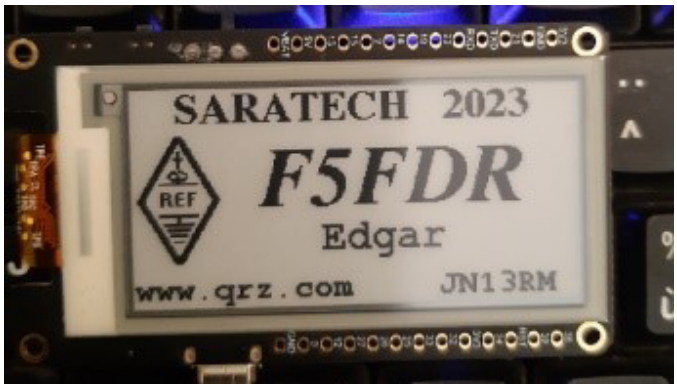
La seconde solution utilise une carte SD préalablement formatée en FAT32 et sur la quelle sera stockée un fichier qui devra être nommé « badge.json ». Ce fichier est fourni en téléchargement. Le simple fait de mettre la carte SD dans le support de l'afficheur et de lui faire exécuter un RESET suffira à prendre en compte les données que vous aurez définies à l'aide d'un simple éditeur de texte comme « Notepad++ ».



Ce fichier devra être stocké dans la racine de la carte SD. Une partie de ce fichier est décrite ci-dessous :



Comme vous pouvez le constater, ce fichier est composé de sections entre { } avec une syntaxe et une ponctuation bien précise qu'il vous faut obligatoirement respecter. En fait, il vous suffit de modifier les textes entre les « », les positions X et Y et les options d'affichage qui sont situées à droite des « : ». Notez que la moindre erreur de ponctuation fera échouer le chargement du fichier. La modification de ces paramètres et les diverses limites qui vous sont imposées par le programme sont décrites dans le fichier « defaultData.h » et vous devez vous y référer sous peine de ne pas obtenir un fonctionnement correct du badge.

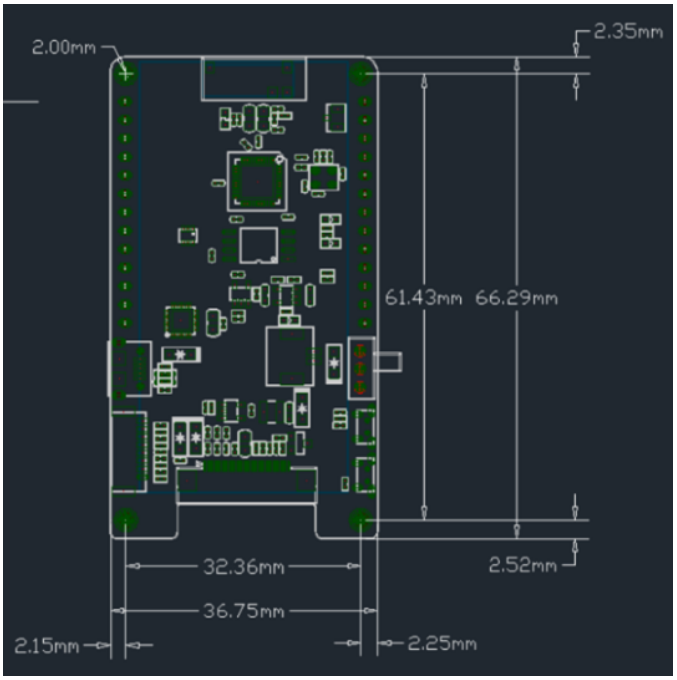


Notez que le fichier « JSON » proposé en téléchargement contient les mêmes paramètres par défaut que ceux définis dans le fichier « defaultData.h » utilisés lors de la compilation du programme.

CONCLUSION

Une fois vos textes correctement affichés, vous pouvez mettre hors tension l'afficheur, le déconnecter du PC et enlever éventuellement la carte SD.

Il ne vous reste plus qu'à trouver un porte badge de taille adaptée et vous êtes fin prêt pour vous rendre à votre salon OM. Pour les passionnés d'impression 3D, voici les cotes mécaniques de notre « gadget » si vous souhaitez réaliser un petit support spécifique. L'épaisseur totale est de 6 mm.



Bonne réalisation.

Afficheur à commander : **LilyGo T5 V2.3.1** modèle **DEPG0213BN 9102 Chip.**

Attention ! Sélectionnez bien **ce modèle** qui est équipé du circuit USB pour la connexion directe au PC.

Lien Internet :
http://urls.r-e-f.org/wr983pq