Une image contenant Graphique, texte, graphisme, Police

Description générée automatiquement CALENDREAU Axel CREAUZEAU KevinUne image contenant art

Description générée automatiquement avec une confiance moyenne

**ESE partie informatique, compte rendu**

Nous décrivons ici comment fonctionne le code que nous utilisons sur la carte stm32 qui sert à recevoir et afficher les données reçues depuis notre partie électronique

Nous répondons en même temps aux différentes parties de programmation intermédiaire

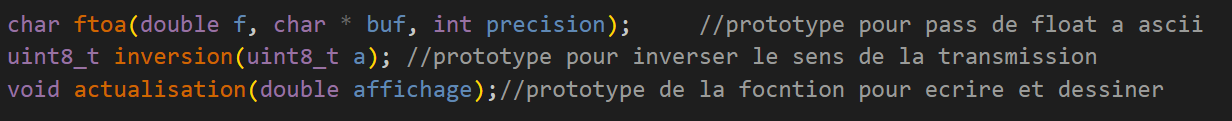
Nous avons premièrement créé notre projet dans stm32cube mx nous y avons posé les bases de notre programme :

Sélection de la carte de développement

Choix des fonctions à activer ou désactiver

On y a activé les interruptions ainsi quel la lecture de l’UART contrôlé par les interruptions, nous avons notamment dû calculer les données nécessaires à l’interruption en calculant les valeurs pour le prescaler et pour de l’autolreload register

Nous avons fait le reste dans stm32cube ide



*Partie prototypage de fonction*

Nous définissons le prototype pour la fonction ftoa qui prend en entrée notre double et en sortie le pointeur de notre chaine de caractère

Le prototype de notre fonction qui nous permet d’inverse le sens des bits

Et notre fonction qui sert à afficher les choses à l’écran

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

Description générée automatiquement

*Utile pour la fonction FTOA*

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

Description générée automatiquementPartie variable*

Avec les valeurs qui servent pour l’aiguille, on peut voir que la moitié des variables sert à retenir les anciennes valeurs, cela nous permet d’effacer ce qu’on a dessiner avant

On a aussi la variable reçue qui va recevoir les donné depuis l’interruption

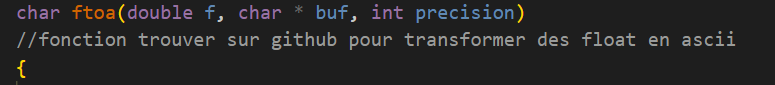
Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

*Partie inversion du sens de la donnée*

Lorsque nous passons par notre carte de test tout fonctionne parfaitement car les bits sont envoyés dans le bon sens

Mais quand on va utiliser la carte que nous avons créer on va avoir besoin d’inverser le sens des bits



Fonction ftoa

Nous définissons une fonction ftoa qui sert à transformer une valeur double en chaine de caractère

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Fonction dessin

Dans cette fonction nous récupérons les données nous les transformons en chaine de caractère

Si la valeur est différente que celle affichée précédemment

Nous calculons l’angle nécessaire a l’affichage de l’aiguille

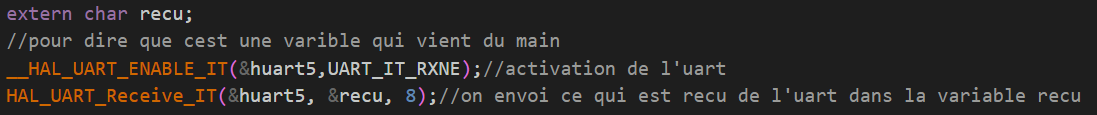
Nous effaçons l’ancienne valeur affichées

Puis nous affichons la nouvelle

Nous effaçons l’ancienne aiguille via les valeurs « old »

Nous dessinons la nouvelle

Et on gardons en mémoire la valeurs actuelle pour pourvoir les effacer au prochaine appel de la fonction actualisation()



*Partie interruption dans le fichier stm32f4xx\_it.c*

On l’on active l’uart en donnant le pointeur du handle

Et on met la valeur lue dans une variable reçu qui est externe car elle vient du main.c



*Le timer 10 effectue ceci*

Ceci permet de faire clignoter la led en même temp que la lecture de l’uart pour être sur que la carte fonctionne

**Plus important le main pour tout relier**

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

*Pour écrire les trucs de base et faire des variables utiles que dans le main*

On affiche notre image

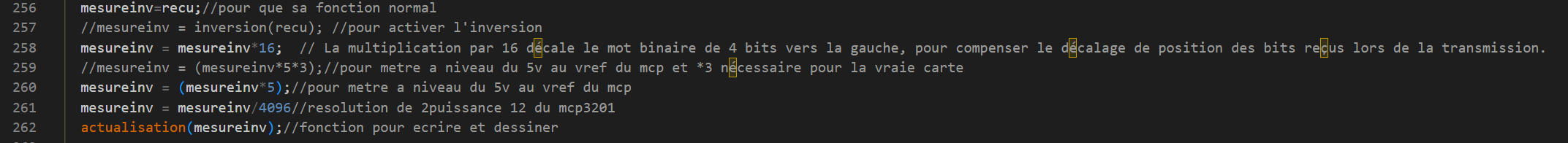
On écrit volts en haut a droite

Nos noms en bas

Et on créer les variables que nous allons utiliser dans nos calculs



Pour démarrer le timer 10 qui fait clignoter la led et lance la lecture des bits



*Partie while*

A chaque itération on récupère la le contenu de la variable remplie depuis l’interruption

On décale les bit vers la gauche pour enlever ce qui sert a rien

On multiplie cette valeur par 5 car c’est notre tension de référence du mcp3201

Si l’on utilise la carte de développement pour envoyer les données il faut commenter la ligne 260 et décommenter les lignes 257 et 259 pour activer l’inversion de bit et la multiplication par 3 a cause de notre mcp et de notre pont diviseur

On multiplie par la résolution du mcp3201 qui est de 12 bit

Puis nous utilisons la fonction actualisation avec la valeur voulu en paramètre

**Affichage**

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Nous avons dessiné notre image via inkscape, ce fut utile notamment pour nos formes et nos texte arrondis car en vectoriel ceci est beaucoup plus simple

Nous sommes ensuite passé par un convertisseur pour transformer notre image en code c, nous avons eu un tableau lisible par du code c

Une image contenant texte, Appareils électroniques, Ingénierie électronique, Matériel d’ordinateur

Description générée automatiquementUne image contenant texte, Appareils électroniques, Ingénierie électronique, Matériel d’ordinateur

Description générée automatiquement

Voici notre résultat sur l’affichage

**Conclusion**

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

Description générée automatiquement

Nous avons dû utiliser le debugger de stm32cubide

Pour voir ce qui ne marchait pas car nous avions des problèmes par rapport au type de certaine de nos variables

Nous avons choisi de passer par le paradigme de la programmation procédurale on l’on divise notre code en fonction pour nous permettre de mieux comprendre comment le code fonctionne en bloc

Ceci nous permet aussi d’activer ou désactiver certaine partie de notre code si nous utilisons la carte de test ou notre carte finale

Nous pensons que ce code est encore améliorable pour économiser plus de ressource ce qui permettrai de prendre une carte plus petite et de consommer moins d’Energie, nous pourrions par exemple dé l’interruption vérifier si la valeur a changé avant de l’envoyer dans le reste du code, de nombreuses autres solutions sont possible et restes à explorer.

Pour un environnent de production lorsque la carte sera programmée pour aller chez le client il faudrait laisser le code final qui fait les bonnes choses ou on ne peut plus sélectionner si l’on souhaite faire l’inversion des bits ou pas, même si le code commenté n’es pas mis dans la carte grâce au compileur on souhaite le strict nécessaire pour éviter les erreurs de manipulation du code.