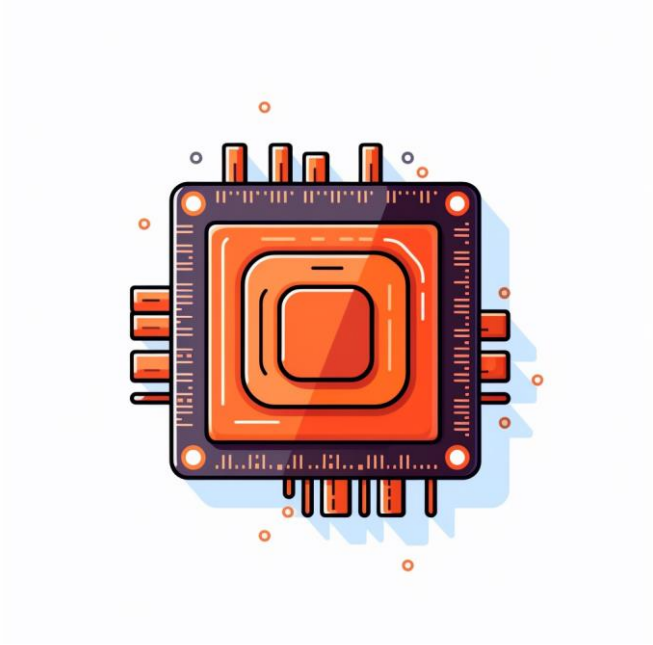
Génie électrique / électronique

3e semestre 2022-2023

Rapport de projet

Programmation objet

2202 Interface carte ARM



Réalisé par :

Miguel Santos

À l'attention de :

Philippe Bovey

Début

2 mars 2023

Fin

2 juin 2023

Table des matières

1. Introduction

L'ETML-ES dispose d'un kit de programmation équipé d'un microcontrôleur ARM. Ce kit est utilisé dans le cadre du cours d'ELNU de première année de génie électrique, dans le but de sensibiliser les étudiants à la programmation sur microcontrôleurs.

2. Cahier des charges

L'objectif du projet est de développer une application en C# qui permettra d'établir une communication bidirectionnelle avec un kit ARM par le biais d'un câble USB.

Celle-ci offrira une interface conviviale pour les étudiants qui permettra d'interagir avec tous les périphériques embarqués sur le kit, comme des LEDS ou des boutons. De plus, l'application devra offrir la possibilité de contrôler des cartes annexes, comme des moteurs et autres composants externes.

Pour le reste du projet, les détails et spécifications sont laissés à libre appréciation.

3. Principe de fonctionnement

L'objectif du projet est de créer une application qui fonctionne comme un "miroir virtuel" de l'état physique de la carte. Cela signifie que chaque partie du système communique entre elles pour partager tous les changements ou actions réalisés.

Lorsqu'un changement physique se produit sur la carte, par exemple l'état d'un bouton qui change ou une action effectuée sur un périphérique, cette information est transmise à l'application logicielle. De même, si l'application logicielle envoie une commande ou une action à la carte, cette information est transmise à la carte.

En utilisant ce mécanisme de communication bidirectionnelle, l'application peut refléter en temps réel l'état physique de la carte et permettre aux utilisateurs d'interagir avec celle-ci de manière virtuelle. Cela permet une synchronisation et une mise à jour constantes entre l'état réel et virtuel, offrant une représentation précise de l'état de la carte dans l'application.

4. Dépôt GitHub

Le code mentionné dans ce rapport est disponible sur le dépôt GitHub suivant :

https://github.com/Miguel-SLO/2202_InterfaceKitARM.git

Les droits d'accès sont fournis à Mr. Bovey. Une copie est réalisée sur les serveurs locaux de l'ES selon les conventions habituelles des projets.

5. Trame de communication

La communication entre l'application C# et le kit ARM est établie via l'UART en utilisant un câble USB. Afin d'optimiser les performances du kit ARM, une trame courte a été choisie afin de réduire le temps de traitement des données.

Header	Command	Data	CRC
1 byte	1 byte	1 byte	1 byte

Figure 1 - Trame de communication USB

- Le header est utilisé pour marquer le début de la trame.
- La commande spécifie l'action à effectuer, par exemple l'allumage d'une LED.
- Les données correspondent aux informations nécessaire pour la commande. Par exemple le numéro de la LED à allumer ou le caractère à afficher à une certaine case du LCD.
- Le CRC est utilisé pour garantir l'intégrité des données reçues.

Chaque commande est assignée à un numéro spécifique. Dans le cas de l'écran LCD, chaque caractère se voit assigné une commande spécifique. La liste des commandes actuellement assignées se trouve sur le dépôt GitHub.

L'ordinateur et la carte communiquent en mode "point à point" sans distinction de rôles (« maître » ou « esclave »). Ils échangent des informations pour maintenir la synchronisation entre l'état réel de la carte et sa représentation virtuelle dans l'application.

A chaque commande envoyée, une confirmation est renvoyée respectivement soi par le kit soit par le PC. De cette manière on garantit que la commande a bien été exécuté et que la synchronisation est constamment respectée.

6. Machines d'état

Dans ce projet, une approche basée sur une triple machines d'états a été adoptée pour la partie concernant le kit ARM.

La première et deuxième machine d'état se concentre sur la communication et la synchronisation entre l'ordinateur et la carte, en utilisant un buffer d'actions pour assurer un flux constant de commandes sans ralentir le système.

La troisième machine d'état est réservée aux étudiants, leur offrant la liberté de développer leur propre code et d'ajouter des fonctionnalités personnalisées à l'application. Ils peuvent ainsi visualiser virtuellement l'état de leur carte et interagir avec elle. Cette approche permet une division claire des responsabilités, favorisant un développement modulaire, une coordination efficace et offrant aux étudiants une plus grande flexibilité dans la création de leur application.

6.1. Machine d'état « USB »

La première machine d'état dans ce projet est spécifiquement conçue pour gérer la communication via le port USB. Son rôle principal est de superviser l'envoi et la réception des trames de données, tout en assurant la vérification de leur intégrité. Elle s'occupe de la réception des trames provenant de l'ordinateur et de leur traitement approprié, ainsi que de l'envoi des trames vers l'ordinateur en garantissant leur fiabilité.

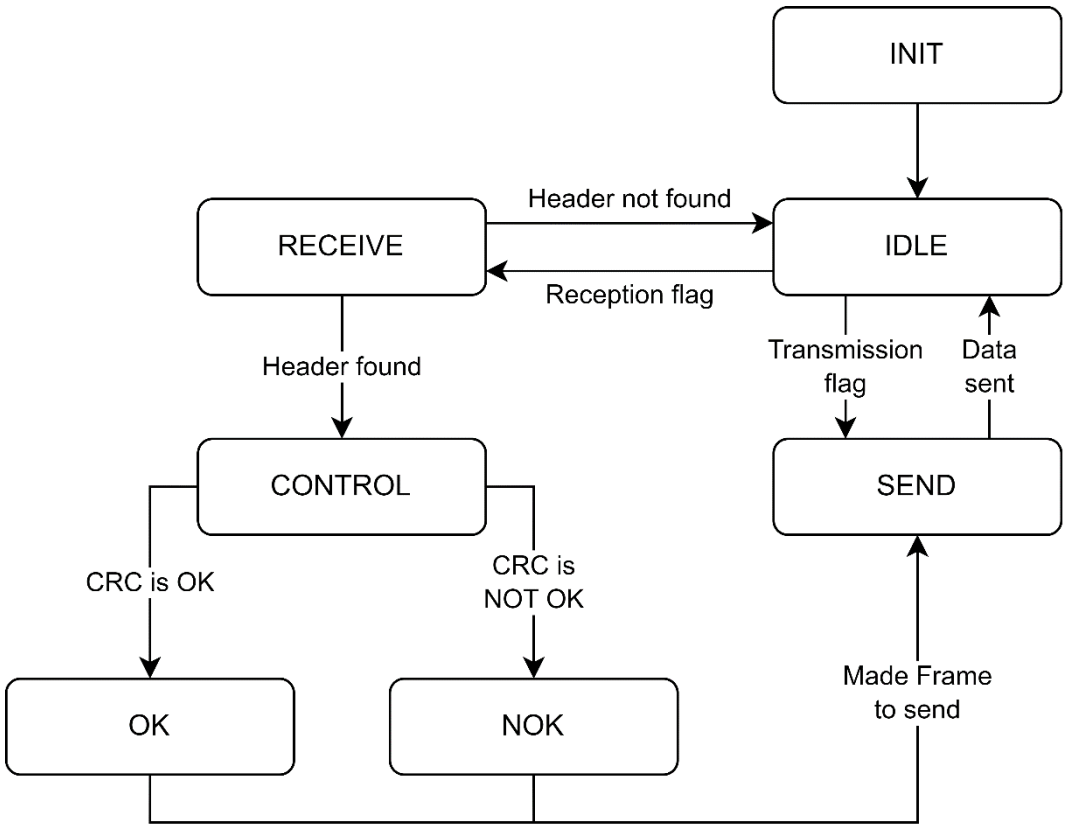


Figure 2 - Machine d'état USB

6.2. Machine d'état « ACTION »

La deuxième machine d'état, chargée de gérer les différentes actions sur le kit ARM, sera associée à une librairie contenant les fonctions nécessaires pour exécuter ces actions. Cette librairie pourra être utilisée par la machine d'état USB et l'application développée par les étudiants. La machine d'état des actions sera responsable de la coordination des actions en fonction des commandes reçues, du suivi de l'état des périphériques et de la gestion des données associées.

6.3. Machine d'état « APP »

La troisième machine d'état représentera l'application spécifique qui sera développée par les étudiants. Elle sera conçue en fonction des besoins du projet et des fonctionnalités souhaitées. Cette machine d'état interagira avec la machine d'état des actions pour envoyer des commandes, recevoir des informations sur l'état des périphériques et maintenir la cohérence entre l'application et l'état physique du kit ARM.

7. Software de l'interface C#

L'application en C# a pour but d'avoir une interface la plus conviviable possible. Elle représente les différents périphériques présents sur la carte du kit.

Les performances ont moins d'importances comme l'application va tourner sur un ordinateur fixe. De ce fait, on peut envisager d'envoyer une trame à chaque événement sur l'interface, comme l'appui d'un bouton.

Figure 3 - Capture d'écran de l'application C#

8. Etat du projet

Actuellement, seule l'application C# a été réalisée.
Les LEDS et les boutons ont été implémentée ainsi qu'une simulation de l'écran LCD. Des animations apportent une certaine convivialité et une part de réalisme à l'application.

L'écran LCD a été simulé en utilisant une certaine quantité de « picture box ». Pour cela, j'ai créé un dossier regroupant chaque caractère de la table ASCII en une image du style d'un écran LCD. En mettant côte à côte les « picture box » dans un « array » on peut créer la sensation d'un écran LCD. Les images ont été nommée pour correspondre à leur valeur en décimal dans la table ASCII.

Pour tester le fonctionnement de l'envoi des trames du kit ARM vers le PC, il est nécessaire d'établir une communication entre les deux appareils et de développer le code requis de chaque côté. La logique doit être implémentée sur le kit ARM afin de détecter les changements physiques et les appliquer et de générer les trames correspondantes, en respectant le protocole de communication défini. Du côté du PC, il est important de configurer la réception des trames en ouvrant un port de communication et en traitant les données reçues.

9. Journal de travail

Voici le travail qui a été réalisé pour ce projet.

Journal de travail	
Mars	Réalisation du cahier des charges.
Avril	Réalisation de l'application C#. Concept et flowchart pour l'application C du kit ARM.
Mai	Ecriture du rapport.

Figure 4 - Journal de travail

10. Perspectives

Désormais, il faudrait implémenter les différentes machines d'états sur le kit ARM. Cela représenterait un exercice intéressant pour les futurs étudiants qui pourront prendre l'habitude de travailler avec des machines d'états en préparation à leur deuxième année.

En parallèle, il faut implémenter les périphériques restants sur l'application C#. La visuelle peut aussi probablement être amélioré.

11. Conclusion

Ce projet m'a permis de m'intéresser plus en profondeur à la programmation objet en C#. J'ai pu approfondir les différentes possibilités de créer une application animée. J'ai dû réfléchir à un protocole de communication le plus performant possible pour le kit ARM afin de permettre à mon application de tourner en parallèle d'une autre sans pour autant la gêner ou réduire ses performances.

J'aurai souhaité m'impliquer un peu plus profondément dans ce projet et je regrette ne pas avoir eu suffisamment de temps pour cela.

Lausanne, le 02 juin 2023

Miguel Santos