Introduction aux miniprojets

Francesco Mondada, Daniel Burnier
IEM - STI - EPFL

Date					
18.Feb.25		Cours 1	**08.Apr.25**		8:15 test valable pour 40%
20.Feb.25		TPIntro	10.Apr.25		miniprojet
25.Feb.25		Cours 2	**15.Apr.25**		miniprojet
27.Feb.25		TP1	17.Apr.25		miniprojet
04.Mar.25		Cours 3	**22.Apr.25**		Vacances de Pâques
06.Mar.25		TP2	24.Apr.25		Vacances de Pâques
11.Mar.25		Cours 4	**29.Apr.25**		miniprojet
13.Mar.25		TP3	01.May.25		miniprojet
18.Mar.25		Cours 5	**06.May.25**		miniprojet
20.Mar.25		TP4	08.May.25		miniprojet
25.Mar.25		Cours 6	**13.May.25**		miniprojet
27.Mar.25		TP5	15.May.25		miniprojet - deadline 23h
01.Apr.25		8:15 test à blanc	**20.May.25**		présentations miniprojet
03.Apr.25		miniprojet	22.May.25		présentations miniprojet
			27.May.25		présentations miniprojet
			29.May.25		ascension

Etape 0:

Dernier délai pour avoir le groupe (deux personnes) pour le miniprojet

Etape 1: Définir

Commencez par définir votre projet sur papier:

- **quels capteurs vous allez utiliser:** est-ce que vous êtes au clair sur les caractéristiques? Est-ce que vous devez valider des aspects?
- **quelle application vous voulez faire:** quels choix sont délicats, où est-ce qu'il peut y avoir des problèmes? Quelle démo finale? Faisable par vidéo interposée
- **quelles ressources vous voulez utiliser:** quelles représentations de nombres, mémoire, FPU, DSP, périphériques?
- **Pas toucher le hardware du robot. Il y a trois supports mécaniques pour visser des éléments, mais pas de soudures, pas de modification mécanique, le robot doit rester identique pour les prochaines utilisations**

Evitez de changer votre projet en cours de route, sinon vous n'allez pas réussir à faire quelque chose de propre.

Etape 2: Organiser

Avant d'écrire le code et les algorithmes en détail, prenez le temps de planifier SUR PAPIER l'architecture de votre code, avec la construction de différentes librairies, gestion correcte et optimale des Threads, etc.

Un conseil: allez voir comment sont faits les exemples de code dans la librairie e-puck2_main-processor (src/main.c).

Attention! Ne partez pas sur la base des TP1-3. L'idéal serait de partir du TP4 ou TP5 qui se base sur e-puck2_main-processor et d'inclure les parties des autres TPs ou de la librairie.

Etape 2: Organiser

- Il est demandé dans la donnée d'avoir un thread par capteur/actuateur. Cependant si vous regardez dans la librairie e-puck2_main-processor, vous verrez que les librairies des ces mêmes capteurs/actuateurs fonctionnent déjà avec des threads dédiés lorsque c'est nécessaire. Il n'est donc pas utile de créer des threads dans votre code juste pour un capteur. En revanche ça peut être nécessaire pour faire du traitement des données d'un capteur par exemple.
- Il faut créer et utiliser un thread uniquement si cela vous paraît utile dans la façon de fonctionner de votre projet (besoin de gérer deux chose en parallèles ou à des fréquences différentes par exemple). Inutile de créer un thread juste pour en créer un. C'est une perte de ressource mémoire inutile et ça n'apporte rien au projet.

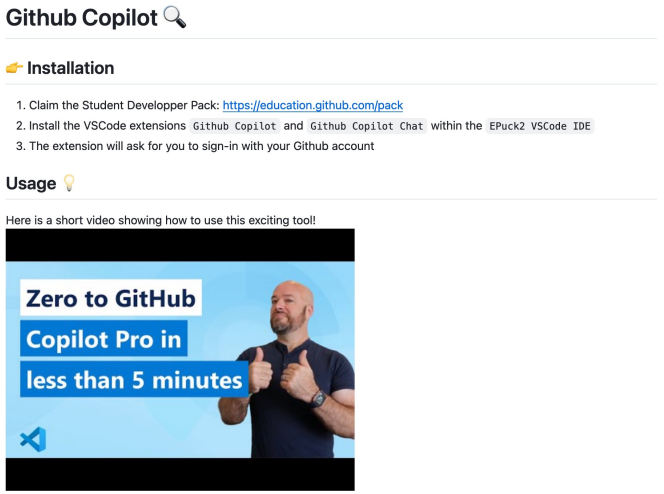
ChatGPT et COPILOT

Code:

- Faites-vous aider, mais en gardant un œil critique
- Des ressources sur copilot sur https://docs.google.com/spreadsheets/d/1SRd_y25MNqwgO1hqiu6LRJ0QIDYWda8hwcxIFRVIGl0/edit?usp=sharing

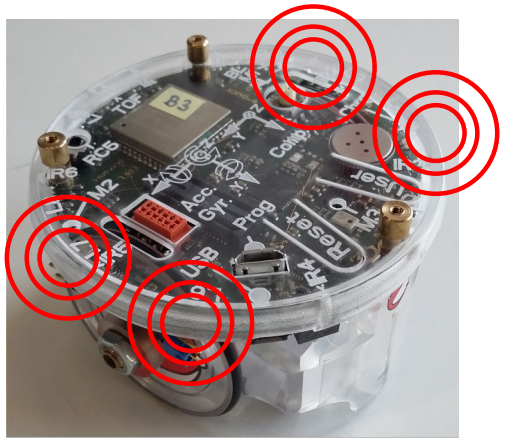
Rapport:

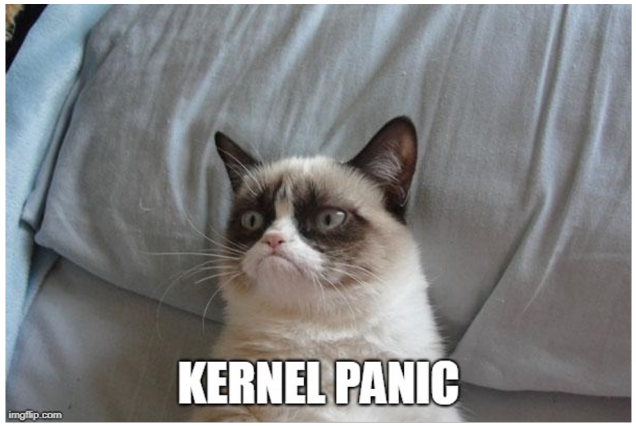
- Ne générez pas de texte avec ChatGPT
- Faites-vous aider à avoir la bonne forme
- Citez!



Pièges à éviter: Kernel Panic

Vu que vous codez avec un RTOS, il y a des risques d'obtenir des erreurs du système (Kernel Panic). Dans ce cas, le robot allume certaines LED en rouge et se met en attente.





Exemple de causes d'un Kernel Panic

- Violation de la mémoire (Exemple pas assez de mémoire allouée à un Thread)

- Appels invalide (exemple Thread non-initialisé)

Si vous avez un Kernel Panic, avant d'appeler un assistant, faites vous même le check de ces points pour vérifier que cela ne vient pas de ça.

Etape 3: Optimiser

- Tenez compte des warnings du compilateur, à la fin il ne devrait pas en rester!
- Évitez d'inclure des librairies non-nécessaires, ou d'initialiser des fonctions (Threads) non-nécessaires
- Ne pas surcharger la pile
- Utilisez les bons types de variables (grosse taille en mémoire vs risque d'overflow)
- Trouvez les bons compromis (justifiable) entre précision et rapidité d'exécution (exemple look-up table, calculs lourds dans un Thread devant s'exécuter rapidement, buffers circulaires pour du filtrage)

Etape 3: Optimiser

- Si vous faites quelque chose qui paraît non optimisé mais que vous pouvez le justifier et expliquer pourquoi c'est nécessaire de faire comme ça alors ok, autrement, ça fait des points en moins…

- Il n'est pas demandé de faire de l'optimisation ultra pointue comme celle qui est implementé dans la librairie arm_cfft_32 vue au TP5 par exemple. Cependant si vous utilisez des float, int32_t, etc pour rien ou que vous itérez dans de nombreuses boucles imbriquées alors qu'on pourrait faire facilement plus simple, là vous pouvez perdre des points, faute d'optimisation évidente. Sinon c'est clair qu'il faut utiliser les librairies existantes qui sont elles optimisées, comme la librairie arm_cfft_32,

Etape 4: Respecter les conventions

- L'utilisation de variables globales est acceptée uniquement si c'est nécessaire et si elles sont déclarées en static (accessibles uniquement depuis le fichier). Par exemple lorsqu'un thread et une autre fonction doivent accéder à la même variable.
- Pas de nombres magiques dans le code. Utilisez des #define pour les constantes que vous utilisez.
- Mettez des petits commentaires dans vos fonctions lorsqu'on ne peut pas comprendre au premier coup d'œil ce qui s'y passe.
- Utilisez des noms de fonctions et de variables qui permettent de comprendre facilement à quoi elles servent.

Soumission du projet

- Soumettre le rapport (format pdf) sur moodle, ne touchez plus github quand vous avez soumis.

- S'inscrire dans le calendrier quand il sera disponible sur moodle

Travail de groupe

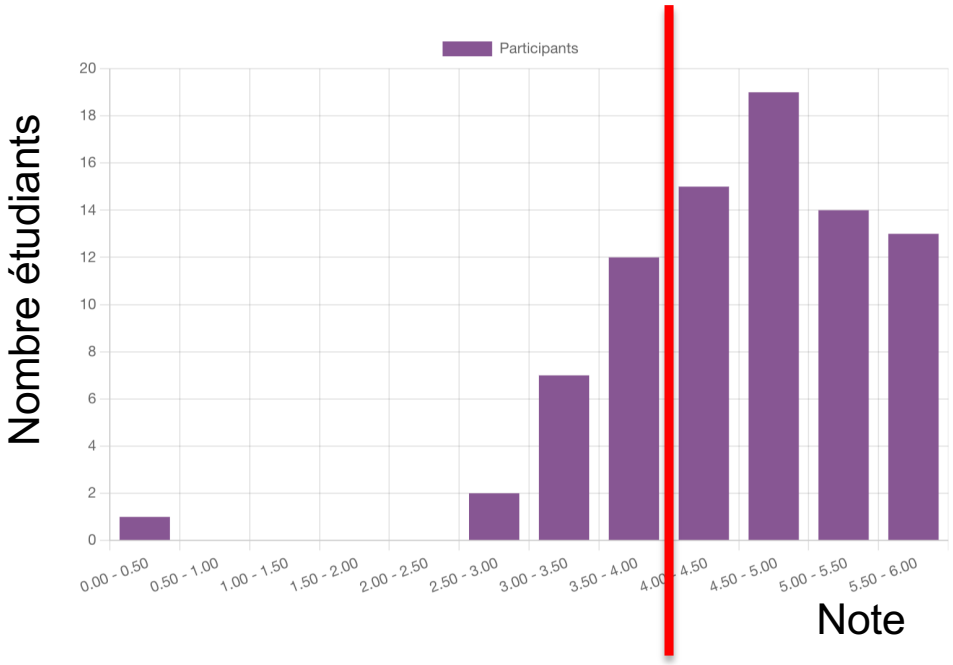
- A la fin de cette semaine, remplissez le formulaire ensemble, PAR GROUPE

  Weekly group self-assessment

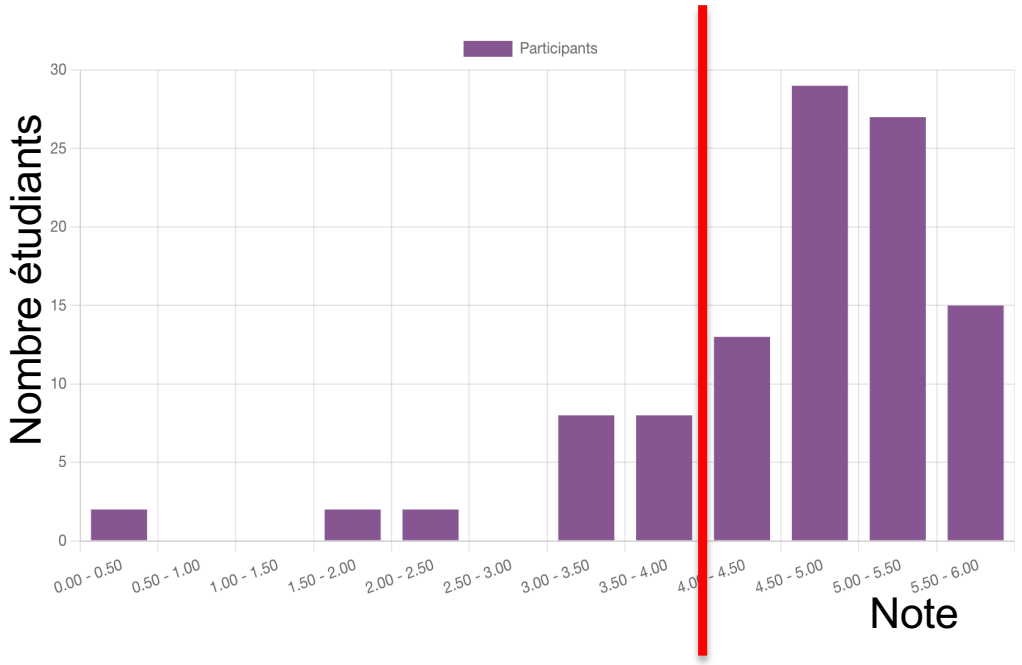
- Ces formulaires sont obligatoires, mais seulement le résumé que vous ferez dans le rapport fera partie de l'évaluation.
- Utilisez le fait de répondre à ce questionnaire pour faire le point sur votre façon de travailler en groupe.
- Utilisez ces questionnaires pour préparer la section du rapport de travail de groupe.

Examen à blanc

Cette année



L' année passée



Examen

Examen : mardi 8 avril

CM 1 2, CM 1 4 , CM 1 5 et CO6 à **8:14**

Enjoy the miniprojet!