Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

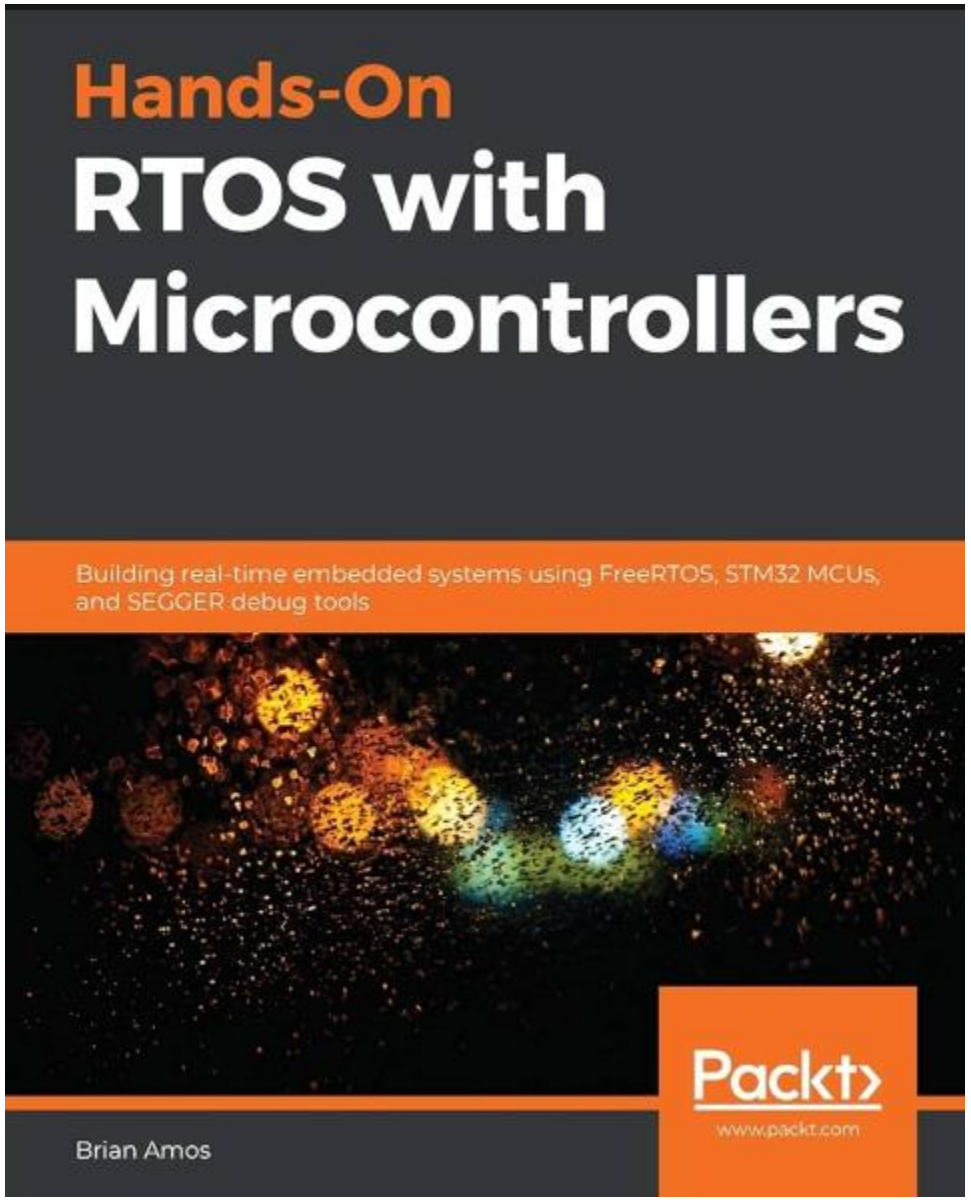
# CEG4566/CSI4541/SEG4545

# Conception de systèmes informatiques en temps réel

Hiver 2024

Professeur : Mohamed Ali Ibrahim, Ph.D., P. Eng.

# Source :


Hands-On
RTOS with
Microcontrollers
Building real-time embedded systems using FreeRTOS, STM32 MCUs, and SEGGER debug tools
Packt
www.packt.com
Brian Amos

# Chapitre 4 :

# Choisir le bon microcontrôleur (MCU)

# Plan

- L'importance du choix d'un MCU
- Considérations sur les MCU
- Considérations relatives au comité de développement
- Présentation de la ligne de produits STM32 MCU
- Comment notre conseil d'administration a-t-il été sélectionné ?

# L'importance du choix d'un MCU

- FreeRTOS est presque exclusivement destiné aux MCU.
- Il s'agit avant tout d'un noyau d'ordonnancement doté d'une API stable, ce qui le rend très adapté à une conception de très bas niveau.
- Contrairement à un système d'unité centrale complet disposant d'un espace d'adressage virtuel pratiquement illimité et de plus de cycles d'horloge que vous ne sauriez en faire, vous allez travailler avec un système aux ressources limitées.
- Si vous développez un micrologiciel sur ce type de système, cela signifie que vous serez beaucoup plus proche du matériel que si vous écriviez un logiciel

# L'importance du choix du MCU

- Les microprogrammes et le matériel sont très étroitement liés, c'est pourquoi il est si important que les ingénieurs en microprogrammes soient impliqués dès le début du processus de développement.
- Dans certaines organisations, il n'y a toujours qu'une seule personne pour effectuer le travail de conception électrique et écrire le micrologiciel.
- Cependant, une tendance croissante pousse les disciplines à se spécialiser de plus en plus dans leurs domaines d'expertise.
- Même dans ce cas, il est important de faire appel à plusieurs membres de l'équipe pour prendre les décisions de conception importantes dès le départ, afin que tout le monde soit conscient des compromis à faire.

# L'importance du choix du MCU

- Si vous n'êtes pas la personne immédiatement responsable de la sélection d'un MCU, il est possible qu'un projet vous soit soumis.
- C'est presque toujours une mauvaise chose, car cela encourage la conception de systèmes sous-optimaux afin d'éviter les retards de calendrier causés par une reconception importante du matériel après la découverte d'un élément essentiel de la fonctionnalité du système.
- Au lieu de s'engager dans une révision importante de la carte pour remédier à un défaut de conception majeur, de nombreuses équipes sont poussées à le corriger avec du code.

# L'importance du choix du MCU

- Ce chapitre n'a pas pour but de dresser une liste exhaustive de tout ce que vous devez savoir et prendre en compte lors de la sélection du matériel pour votre nouveau projet, mais il vise à présenter un grand nombre de facteurs contribuant à la différenciation et à la sélection des dispositifs MCU.

- L'autre chose à garder à l'esprit en lisant ce chapitre est qu'il ne s'applique qu'au choix d'un MCU.

- Comme nous l'avons vu au chapitre 1, Introduction aux systèmes en temps réel, il y a plus d'une façon d'écorcher un système en temps réel - les UC ne sont pas toujours le meilleur choix.

# L'importance du choix du MCU

- Afin de limiter la portée de ce chapitre à ce qui est immédiatement pertinent, pour les exemples présentés dans le reste du cours, nous limiterons notre discussion aux fonctionnalités trouvées dans les appareils basés sur l'ARM Cortex-M.

- Nous nous concentrons sur les MCU ARM Cortex-M parce que les dispositifs basés sur le cœur Cortex-M apportent un mélange utile de caractéristiques qui permettent aux ingénieurs de créer des systèmes embarqués en temps réel de complexité moyenne à élevée utilisant un système d'exploitation en temps réel (RTOS), tout en étant en mesure d'architecturer la solution de manière à ce que les modules soient réutilisables pour d'autres projets.

- Les MCU STM32 ont été sélectionnés en raison de leur popularité, de la large gamme de MCU disponibles, de leur conditionnement en circuit intégré (CI) accessible et des périphériques matériels inclus.

- Bien que nous nous concentrions sur les composants STM32 dans ce chapitre, n'oubliez pas qu'il y a beaucoup d'autres fabricants avec des tonnes de produits excellents et que la grande majorité de ce qui est couvert s'applique également aux composants non-STM32 (et nonARM).

# Considérations sur les MCU

- Il y a quelques considérations à prendre en compte pour choisir un microcontrôleur plutôt qu'une carte de développement.

- En supposant que le projet contienne des composants électroniques entièrement personnalisés, il n'y a pas de limitation sur le choix exact du MCU, comme ce serait le cas si vous ne choisissiez que des cartes de développement.

- Les étudiants et les amateurs se limitent parfois encore plus artificiellement, en restant fidèles à certains écosystèmes et en ne choisissant que des cartes de développement au sein de ces écosystèmes (comme Arduino ou mBed).

- Bien qu'il n'y ait rien d'intrinsèquement mauvais dans aucun des écosystèmes, vous ne parviendrez pas à vous développer en tant qu'ingénieur professionnel si vous êtes incapable d'envisager d'autres solutions ou d'apprécier les atouts uniques que chaque pièce de matériel apporte à un projet particulier.

# Considérations fondamentales

Tout d'abord, nous verrons comment répondre à certaines questions clés qui réduiront immédiatement le champ des candidats MCU potentiels pour un projet :

- Sera-t-il à sa place ?
- Peut-il exécuter tout mon code ?
- Combien cela coûte-t-il ?
- Est-il facilement disponible ?

# Taille physique

- En fonction de la conception, la taille du MCU peut être un facteur important.
- Si vous développez un appareil portable ou à porter sur soi, il est probable que la taille soit en tête de votre liste.
- Parfois, les MCU préemballés sont trop volumineux et les concepteurs doivent recourir à la technologie "chip on board" (où la puce de silicium du MCU est directement collée au circuit imprimé (PCB), au lieu d'être placée dans un boîtier plastique séparé).
- D'autre part, les grands équipements montés en rack ont généralement plus d'espace qu'il n'en faut pour accueillir un MCU de n'importe quelle taille convenant à la tâche.

# ROM

- La mémoire morte (ROM) est un facteur de différenciation assez important entre les MCU d'une même famille, la taille de la ROM étant fortement corrélée au prix.

- En fonction du nombre de modèles différents disponibles dans une famille de produits, il peut y avoir plusieurs MCU avec des ensembles de périphériques très similaires.

- Ces MCU partageront probablement la même empreinte physique, mais disposeront de quantités de mémoire sensiblement différentes.

- Si votre application est sensible aux coûts mais que la ROM requise est inconnue, envisagez l'approche suivante (diapositive suivante).

# ROM

1. Sélectionnez une famille de MCU qui offre plusieurs tailles de flash dans une empreinte compatible.

2. Commencez le développement avec le MCU qui possède le plus de ROM dans la famille. C'est ainsi que l'on obtient la plus grande souplesse pour ajouter des fonctions.

3. Une fois que la taille de l'image finale est connue, le MCU exact (avec une taille de flash plus petite) peut être sélectionné avant de commencer la production de masse.

# ROM

- Si vous adoptez cette approche, vous devrez veiller à laisser suffisamment de place pour les fonctionnalités futures, en supposant que votre produit sera capable de recevoir des mises à jour de son micrologiciel sur le terrain.

- Veillez également à vérifier l'affectation des périphériques d'un modèle à l'autre - la compatibilité des broches n'est pas toujours synonyme de compatibilité des microprogrammes !

# ROM

- La quantité de ROM nécessaire varie considérablement et dépend de la quantité de code à charger sur l'appareil.
- Si vous avez travaillé avec des MCU 8 bits, vous risquez d'avoir une mauvaise surprise en passant à une architecture 32 bits telle que l'ARM.
- Un programme similaire nécessitera plus d'espace flash pour être mis en œuvre sur une architecture 32 bits que sur une architecture 8 bits.
- La bonne nouvelle, c'est que la taille des flashs a évolué, de sorte qu'il est presque toujours possible de trouver un MCU avec suffisamment de flashs intégrés pour répondre aux besoins de votre application.
- L'intégration de bibliothèques tierces dans votre base de code est généralement assez coûteuse en termes de flash, il convient donc d'être prudent si vous choisissez cette voie.

# RAM

- La quantité de mémoire vive (RAM) sur la puce est un autre facteur à prendre en compte : elle suit généralement la quantité de mémoire flash d'un appareil donné.

- Les pièces dotées d'une ROM plus importante ont généralement plus de RAM.

- Quelques exemples où de grandes quantités de RAM seront nécessaires sont le traitement des données qui nécessite de grandes mémoires tampons pour les données, des piles de réseau complexes, des mémoires tampons profondes pour la communication, des interfaces graphiques (en particulier celles qui nécessitent des mémoires tampons), et tous les langages interprétés qui exécutent une machine virtuelle (c'est-à-dire MicroPython et Lua).

# RAM

- Par exemple, supposons que votre application nécessite un écran haute résolution.
- Si l'écran ne dispose pas d'un contrôleur embarqué avec son propre frame buffer, vous êtes probablement déjà dans le domaine de la RAM externe.
- La taille de la mémoire tampon nécessaire pour piloter ce type d'affichage dépassera probablement la RAM disponible sur le MCU.
- En revanche, si vous construisez un système de contrôle simple avec une connectivité et une capacité d'interface utilisateur limitées, une petite quantité de RAM peut suffire.
- Notez également que chaque tâche dans FreeRTOS nécessite sa propre pile (généralement avec un minimum de 512 octets sur le port Cortex-M), donc si un grand nombre de tâches est nécessaire, il sera facile d'utiliser rapidement plusieurs Ko de RAM.

# Fréquence d'horloge de l'unité centrale

- Puisque nous limitons notre discussion aux MCU ayant la même architecture sous-jacente, une MCU avec une fréquence d'horloge plus rapide exécutera le même ensemble de fonctions purement logicielles plus rapidement qu'une MCU avec une fréquence d'horloge plus lente.

- Remarquez le mot clé "pur" dans la déclaration précédente
  - Parfois, des périphériques matériels embarqués peuvent faire une énorme différence en termes de vitesse d'exécution, sans que cela ait un rapport avec la fréquence d'horloge du CPU (par exemple, les fonctions matérielles de virgule flottante et de DSP disponibles sur le cœur Cortex-M4).

- Il faut également tenir compte de la fréquence d'horloge maximale absolue de l'appareil par rapport à la fréquence d'horloge pratique pour une application.

- Par exemple, la fréquence d'horloge maximale de certains MCU est incompatible avec la génération d'une horloge interne de 48 MHz requise pour un périphérique USB, de sorte qu'elle ne peut pas être utilisée à la vitesse maximale si le périphérique USB est également utilisé.

# Traitement des interruptions

- Au sein de la famille ARM Cortex-M, le traitement des interruptions est très similaire.
- Tous les dispositifs comprennent un contrôleur d'interruption vectoriel imbriqué (NVIC) avec une table vectorielle relocalisable et un contrôleur d'interruption externe (EXTI).
- Les considérations spécifiques à l'appareil comprennent les interruptions périphériques exactes qui sont disponibles et la manière dont elles sont mappées au NVIC, ainsi que la manière dont les interruptions externes sont multiplexées dans l'EXTI.

# Prix

- En fonction de l'application, le coût de la nomenclature peut être un facteur important ou ne pas être pris en compte.
- En règle générale, les coûts de nomenclature font l'objet d'un examen de plus en plus minutieux dans les applications à grand volume.
- Toutefois, pour les produits à faible volume, il est souvent plus judicieux de se concentrer sur la réduction du temps et des efforts de développement d'un produit, plutôt que sur l'obtention d'un coût de nomenclature le plus bas possible.

# Prix

- En s'efforçant de minimiser les efforts d'ingénierie et le temps de développement d'un produit à faible volume, le produit sera mis sur le marché plus rapidement.
- Non seulement le produit commence à générer des revenus plus rapidement, mais il accumule également moins de coûts d'ingénierie non récurrents (NRE).
- Moins de NRE se traduit par un retour sur investissement (ROI) plus rapide pour le produit en cours de développement.
- Un retour sur investissement plus rapide fait le bonheur des directeurs et des chefs d'entreprise ! Dans ces situations, s'inquiéter de dépenser quelques dollars sur une nomenclature pour un produit qui se vend par dizaines chaque année - au détriment de semaines ou de mois d'efforts de développement - est rarement un compromis judicieux.

# Prix

- Malgré tous ses inconvénients, la mémoire vive externe offre de nombreuses fonctionnalités, telles que la possibilité de mettre en cache des images entières de microprogrammes dans la mémoire vive pour les mises à jour, des cadres d'interface graphique riches en fonctionnalités, des piles de réseaux complexes et des techniques sophistiquées de traitement des signaux.

- Comme pour toute autre exigence, il faut faire des compromis.

# Disponibilité

- Un aspect souvent négligé par les ingénieurs débutants est la disponibilité attendue et garantie d'un MCU.
- Ce n'est pas parce qu'une pièce est disponible à l'achat au début du projet qu'elle le sera pendant toute la durée de vente du produit final.
- Dans le cas des appareils grand public, ce n'est probablement pas un problème majeur.
- En effet, ces appareils peuvent avoir des volumes extrêmement élevés, mais chaque révision n'est produite que pour une durée limitée (de quelques mois à un an ou deux).

# Disponibilité

- Comparez la mentalité de l'obsolescence planifiée de l'électronique grand public à celle de l'industrie, des télécommunications ou de l'aérospatiale.
- Dans ces secteurs, les délais de développement peuvent se mesurer en années et les périodes de soutien nécessaires sont souvent d'une décennie ou plus.
- C'est pourquoi la disponibilité des pièces est une considération très importante.
- Veillez à vous renseigner sur les garanties de disponibilité offertes par le fabricant et à les mettre en regard de son histoire, de sa réputation et des risques encourus par le projet.
  - Il n'est pas agréable d'achever 80 % d'un projet pour découvrir qu'il est impossible de se procurer le MCU pendant la phase de préproduction !

# Périphériques matériels

- Par rapport à l'univers des CPU, où le processeur lui-même est généralement au centre de l'attention, le choix du bon MCU est plus complexe en raison de l'élargissement de son champ d'application.

- De nombreux éléments matériels différents sont inclus dans la même puce, ce qui nous permet d'optimiser la solution en termes de vitesse, de puissance, d'utilisation de l'unité centrale ou de coût de fabrication.

- Dans le cas d'une conception très contraignante, tous ces facteurs peuvent entrer en jeu et des compromis devront être faits.

# Connectivité

- Dans l'écosystème de plus en plus connecté de l'Internet des objets (IoT), disposer d'une capacité de mise en réseau embarquée sur le MCU peut être une aubaine pour un projet. ... tant qu'il existe le bon micrologiciel pour le piloter.

- Il est important de comprendre que le fait d'avoir un périphérique n'est pas la même chose que d'avoir une fonctionnalité complète.

- Par exemple, ce n'est pas parce qu'un MCU prend en charge une interface réduite indépendante du support (RMII) et une couche physique en réseau (PHY) que vous pouvez immédiatement disposer d'une pile TCP/IP complète
  - toutes les fonctionnalités du micrologiciel doivent provenir de quelque part.

- La connectivité potentielle intégrée dans les appareils peut inclure Ethernet, RMII, 802.11 (WiFi), 802.15.1 (Bluetooth) et 802.15.4 (Zigbee, HART, etc.).

# Unités de protection de la mémoire

- Les unités de protection de la mémoire (MPU) sont utilisées pour garantir que le code n'accède qu'à la plage de la RAM à laquelle il est autorisé.

- Lorsqu'elles sont utilisées correctement, les MPU garantissent une plus grande stabilité du système et une sécurité accrue, car l'application est moins susceptible de provoquer des conséquences involontaires en accédant à une mémoire qu'elle ne devrait pas.

- FreeRTOS inclut un support pour les tâches protégées par le MPU, que nous aborderons dans le Chapitre 15, Gestion de la mémoire sous FreeRTOS.

# Unités matérielles à virgule flottante

- Si votre application doit traiter de nombreux nombres à virgule flottante, une **unité matérielle de virgule flottante (FPU)** peut s'avérer extrêmement utile.
- Jusqu'à la dernière décennie, il était généralement préférable d'éviter les nombres à virgule flottante dans la plupart des systèmes embarqués basés sur des MCU.
- La disponibilité de processeurs plus rapides a commencé à changer la donne.
- Aujourd'hui, les FPU sont souvent implémentées dans le matériel.
- Grâce aux FPU, de nombreuses applications différentes peuvent bénéficier de l'utilisation des mathématiques en virgule flottante, sans subir la pénalité de performance du CPU généralement associée aux implémentations de bibliothèques basées sur des logiciels.
- Les processeurs Cortex-M4 sont équipés en option de FPU à simple précision (32 bits), tandis que les processeurs basés sur le Cortex-M7 ajoutent une prise en charge matérielle optionnelle de l'arithmétique à virgule flottante à double précision (64 bits).

# Fonctions de traitement des signaux numériques

- Outre l'augmentation des performances due à la prise en charge matérielle de la virgule flottante, les MCU Cortex-M4- et Cortex-M7 disposent également d'une fonctionnalité optionnelle de **traitement des signaux numériques (DSP)** intégrée au matériel, qui peut accélérer considérablement certains algorithmes complexes et contribuer à réduire la charge de codage pour les ingénieurs en microprogrammation.

# Canaux d'accès direct à la mémoire

- L'accès direct à la mémoire (DMA) peut s'avérer extrêmement utile dans de nombreuses situations où l'on souhaite disposer d'une grande largeur de bande ou d'un code hautement événementiel.
- Les contrôleurs DMA sont généralement capables d'interagir avec les périphériques de l'unité centrale, ainsi qu'avec différentes parties de la mémoire vive.
- Ils se chargent de remplir les registres périphériques et la mémoire vive sans impliquer le processeur.
- Ces transferts autonomes peuvent libérer une grande partie du temps de l'unité centrale en réduisant considérablement la charge d'interruption et la commutation de contexte.

# Interfaces de communication

- Nous avons déjà introduit la connectivité réseau externe par rapport aux technologies Ethernet et sans fil.

- Il existe de nombreuses interfaces de communication différentes qui sont plus traditionnellement associées aux dispositifs intégrés et qui sont généralement disponibles en tant que périphériques matériels sur un MCU.

- Les interfaces utilisées pour communiquer avec les capteurs et les actionneurs embarqués et non embarqués sont les suivantes :

    1. Communication inter-IC (I2C)
    2. Interface périphérique série (SPI)
    3. Récepteur-émetteur synchrone/asynchrone universel (USART)

# Interfaces de communication

Les périphériques suivants sont régulièrement utilisés dans les environnements automobiles et industriels pour la communication entre les modules :

1. USARTs
2. Réseau de contrôleurs (CAN)
3. Réseau local d'interconnexion (LIN)

# Interfaces de communication

Les périphériques suivants sont régulièrement utilisés dans les environnements automobiles et industriels pour la communication entre les modules :

1. USARTs
2. Réseau de contrôleurs (CAN)
3. Réseau local d'interconnexion (LIN)

# Moteurs cryptographiques matériels

- Si votre application nécessite une connectivité externe, vous devez également penser à la sécurité.

- De la même manière que les FPU rendent les opérations en virgule flottante plus efficaces pour l'unité centrale, des moteurs de cryptographie basés sur le matériel sont disponibles sur certains MCU, ce qui réduira considérablement la charge de travail de l'unité centrale nécessaire pour transférer des données en toute sécurité sur les réseaux publics.

# Matériel de synchronisation

- Un MCU comporte souvent plusieurs périphériques de synchronisation différents.
- Les périphériques eux-mêmes comprennent souvent des fonctions de capture d'entrée, de comparaison de sortie et de modulation de **largeur d'impulsion (PWM) au** minimum.
- Certains appareils comprennent également un matériel de synchronisation pour l'interface avec les codeurs en quadrature.

# Matériel de synchronisation

- La capture d'entrée consiste à capturer le moment où une entrée numérique de l'unité centrale change d'état.
- Les périphériques des MCU le font avec une résolution beaucoup plus élevée que les microprogrammes, car ils utilisent des compteurs à haute fréquence et des portes matérielles pour capturer la transition du signal (plutôt que de s'appuyer sur plusieurs instructions de l'unité centrale).
- Il existe souvent plusieurs canaux de capture d'entrée disponibles, qui peuvent être utilisés en parallèle.
- La comparaison de sortie est en fait l'inverse de la capture d'entrée (un signal est émis avec des caractéristiques temporelles précises).
  - compare fait référence au comparateur matériel utilisé pour déterminer quand effectuer la transition.

# Matériel de synchronisation

- Le PWM et la capture d'entrée sont tous deux très couramment utilisés dans les systèmes de contrôle pour interagir avec les capteurs et les actionneurs.
- Certains actionneurs acceptent des entrées PWM.
- La MLI peut également être utilisée pour fournir un contrôle proportionnel d'un pilote binaire (tel qu'un transistor), qui peut être utilisé pour modifier avec précision la quantité de puissance fournie à une charge.
- De nombreux codeurs différents fournissent également des informations au format PWM, qui peuvent être lues par le MCU à l'aide du mode de capture d'entrée d'un périphérique de temporisation.

# Matériel de synchronisation

- **Les entrées de codeur en quadrature (QEI)** sont extrêmement utiles dans les systèmes de retour de mouvement.

- Bien qu'il soit possible d'obtenir une fonctionnalité similaire en utilisant plusieurs canaux de capture d'entrée (ou lentement, sans matériel dédié), le fait de disposer d'un matériel QEI dédié permet une intervention très minimale de l'unité centrale, même à des taux d'entrée élevés.

# Matériel de synchronisation

- **Les entrées de codeur en quadrature (QEI)** sont extrêmement utiles dans les systèmes de retour de mouvement.

- Bien qu'il soit possible d'obtenir une fonctionnalité similaire en utilisant plusieurs canaux de capture d'entrée (ou lentement, sans matériel dédié), le fait de disposer d'un matériel QEI dédié permet une intervention très minimale de l'unité centrale, même à des taux d'entrée élevés.

# Analogique intégré

- Les convertisseurs numériques-analogiques (CNA) et les convertisseurs analogiques-numériques (CAN) sont utilisés pour convertir des valeurs analogiques variant continuellement en représentations numériques associées à ces valeurs.
- Le plus souvent, ces types de périphériques intégrés ont une résolution et une fréquence inférieures à celles des puces externes.
- Cependant, en fonction des exigences de votre système, ils peuvent s'avérer extrêmement utiles.
- Un autre périphérique utile est constitué par les comparateurs intégrés, qui signalent au processeur qu'une valeur analogique est supérieure ou inférieure à un seuil donné.

# Interfaces tactiles dédiées

- Grâce à la prévalence croissante des interfaces tactiles, des implémentations complètes de contrôleurs tactiles sont désormais incluses dans certains MCU.

- Cela peut réduire considérablement la quantité d'expertise et d'efforts nécessaires à la mise en œuvre d'une interface tactile entièrement fonctionnelle et robuste.

# Interfaces d'affichage

- On les trouve généralement dans des appareils plus performants, dotés d'un plus grand nombre de broches, d'interfaces d'affichage sophistiquées et même d'une accélération graphique, ce qui est de plus en plus courant.

- Il faut s'attendre à trouver des interfaces parallèles LCD/TFT (par exemple, 6800 et 8080) sur un grand nombre de pièces, avec des interfaces telles que MIPI DSI capables de piloter des écrans haute résolution peu coûteux jusqu'à des écrans ne comportant que quelques lignes.

- Les circuits intégrés de conversion de protocole matériel peuvent être utilisés pour s'adapter à un certain nombre de normes d'affichage différentes, telles que LVDS et HDMI.

- Les MCU sont désormais capables d'offrir une expérience utilisateur riche, grâce à une accélération matérielle supplémentaire et à des logiciels intermédiaires et des pilotes écrits de manière efficace.

# Prise en charge de la mémoire externe

- Dans les boîtiers à nombre de broches plus élevé, il faut s'attendre à trouver une prise en charge de la **mémoire statique à accès aléatoire (SRAM).**

- La prise en charge de la **mémoire vive dynamique synchrone (SDRAM)**, avec un contrôleur embarqué prenant en charge les exigences de synchronisation et les cycles de rafraîchissement, peut être trouvée dans des appareils plus performants.

- Les appareils axés sur la performance intègrent généralement la prise en charge de la quadruple interface SPI. Souvent, la mémoire vive externe - et même les dispositifs quad-SPI - peut être mappée en mémoire et utilisée de la même manière que le stockage interne, même si les performances s'en ressentent.

- De nombreux appareils sont également équipés de contrôleurs **MultiMediaCard (MMC)** et **Secure Digital Card (SD), ce qui permet d'**ajouter facilement un espace de stockage amovible grand public.

# Horloge en temps réel

- Des calendriers matériels sont également disponibles sur certains appareils ; il suffit d'un cristal de 32 kHz et d'une source d'alimentation de secours, telle qu'une pile au lithium primaire CR2032.

- Cette capacité s'accompagne généralement d'une quantité limitée de mémoire vive sauvegardée par la batterie.

# Support audio

- Des calendriers matériels sont également disponibles sur certains appareils ; il suffit d'un cristal de 32 kHz et d'une source d'alimentation de secours, telle qu'une pile au lithium primaire CR2032.

- Cette capacité s'accompagne généralement d'une quantité limitée de mémoire vive sauvegardée par la batterie.

# Support audio

- La prise en charge audio haute fidélité par Inter-IC Sound ($I^2S$) est généralement disponible.
- Il faut s'attendre à trouver des canaux DMA attachés au périphérique $I^2S$ afin de minimiser la quantité d'intervention de l'unité centrale requise pour alimenter les CNA gourmands en données et collecter les données des ADC sur ces bus.
- Ceci conclut notre longue liste de périphériques matériels à prendre en compte lors de l'évaluation des MCU.
- Le prochain thème abordé est celui de la consommation d'énergie, qui intéressera tout particulièrement les personnes souhaitant concevoir des batteries ou des dispositifs de récupération d'énergie.

# Consommation électrique

- Les MCU à faible consommation d'énergie sont la tendance depuis plus d'une décennie.
- Cependant, ce qui était historiquement un cas d'utilisation spécialisé avec des options limitées (comme le MSP430 16 bits) est maintenant devenu courant, grâce à la pléthore d'appareils IoT alimentés par batterie qui arrivent sur le marché.
- Aujourd'hui, il existe des MCU 32 bits complets, qui peuvent passer rapidement du mode "deep sleep" (sommeil profond) au mode "run" (exécution) à haut débit et à forte consommation de données.

# Efficacité énergétique

- Cela peut paraître simple, mais un bon moyen de s'assurer que quelque chose consomme moins d'énergie est de l'éteindre (ne riez pas, cela peut être étonnamment compliqué en fonction des pièces concernées, grâce à divers courants de fuite !)

- Si les MCU complexes dotés de dizaines de périphériques ont une chance d'être économes en énergie, il doit y avoir un moyen de désactiver les fonctionnalités qui ne sont pas nécessaires afin de minimiser le gaspillage d'énergie.

- Pour ce faire, on coupe généralement les horloges des périphériques qui ne sont pas utilisés et on s'assure que les broches d'E/S à base de CMOS ne sont pas flottantes (n'oubliez pas que ce sont les transitions dans les dispositifs CMOS qui consomment le plus d'énergie).

# Efficacité énergétique

- Une autre spécification que l'on trouve couramment dans les fiches techniques est la quantité d'énergie consommée par MHz de l'horloge du processeur, généralement spécifiée en µA/MHz.

- Si la quantité de traitement par période de réveil est relativement constante, cela fournit une autre mesure pour comparer les différents modèles de MCU.

# Modes de faible consommation

- Les appareils destinés à des applications à faible consommation d'énergie disposent généralement de plusieurs niveaux d'état d'arrêt différents.

- Ces états permettent au programmeur de faire un compromis entre la consommation de courant, les fonctions disponibles (telles que le maintien du contenu de la mémoire vive et de certains périphériques), le nombre d'interruptions disponibles pour déclencher un événement de réveil et le temps de réveil.

- Heureusement, de nombreuses applications IoT à faible consommation sont assez limitées dans leur champ d'action, de sorte que parfois une combinaison de nouvelles caractéristiques dans un MCU particulier s'avérera très bien adaptée à une application spécifique.

# Heure de réveil

Si un dispositif a un courant d'arrêt étonnamment faible mais prend un temps anormalement long pour se réveiller et se mettre dans un état de fonctionnement utilisable, ce n'est peut-être pas le meilleur choix pour une application qui nécessite des réveils assez fréquents, car il faudra passer beaucoup de temps à mettre le système en marche au lieu de lui faire effectuer le traitement nécessaire, puis de le remettre en veille.

# Tension d'alimentation

- Des tensions d'alimentation plus basses entraînent généralement une consommation de courant plus faible.
- En fonction de la conception, il est possible de faire des compromis entre l'élimination des circuits de conditionnement d'énergie (qui consomment du courant en raison d'un rendement inférieur à 100 %) et l'extension de la plage de tension de fonctionnement utilisable d'un élément de batterie donné.
- La tension requise par le MCU (ainsi que tout circuit auxiliaire) sera un facteur déterminant pour la flexibilité de la régulation.
- Il ne faut donc pas s'attendre à pouvoir faire fonctionner le processeur à la fréquence maximale spécifiée et à la tension d'alimentation la plus basse possible.

# Migration des MCU en milieu de projet

- Il arrive que toutes les exigences du projet ne soient pas connues d'emblée, ou que tout le monde ne soit pas sûr à 100 % de la manière dont il faut résoudre chaque problème détaillé le premier jour.

- Si vous avez la chance de savoir qu'il existe un niveau élevé d'incertitude, il est toujours préférable de s'y préparer plutôt que d'être pris par surprise.

- Voici quelques domaines dans lesquels le choix d'un MCU faisant partie d'une famille ou d'un écosystème plus large peut contribuer à atténuer certains des risques liés à l'incertitude du projet.

# Migration des MCU en milieu de projet

- Lors de la planification d'un éventuel changement de MCU, il convient, si possible, d'identifier à l'avance d'autres MCU dont les broches sont compatibles.
- Par exemple, les pièces du NXP LPC1850 sont compatibles avec les MCU LPC 4350.
- Les appareils STM32 sont tous compatibles au niveau des broches à l'intérieur d'une même famille (et d'un même boîtier), mais ils sont parfois presque compatibles avec les broches d'autres familles (STM32M4 et STM32M7, par exemple).
- ST publie régulièrement des guides de migration pour les ingénieurs qui ont dépassé une famille de MCU et ont besoin de quelque chose d'un peu plus performant.
- Si quelques candidats et alternatives sont sélectionnés dès le départ, de simples cavaliers sur le circuit imprimé peuvent faciliter la migration entre différents MCU dont les performances (et le coût) diffèrent sensiblement, ce qui permet d'éliminer le temps nécessaire pour retravailler le circuit imprimé en cours de projet.

# Similitude périphérique

- La plupart des MCU, au sein d'une famille donnée, hériteront de la même propriété intellectuelle périphérique.

- Les fournisseurs de silicium ne redessinent pas nécessairement les périphériques à partir de zéro chaque fois qu'ils créent une nouvelle famille de MCU, de sorte qu'il y a souvent un chevauchement important dans les cartes de registre et le comportement des périphériques appartenant à un fournisseur donné.

- Souvent, si vos applications n'utilisent qu'un sous-ensemble des fonctions périphériques les plus basiques, vous pouvez avoir la chance d'utiliser en grande partie le même pilote, même si le fournisseur décide de modifier radicalement son API d'une famille de MCU à l'autre.

- Paradoxalement, le matériel brut s'avère parfois plus cohérent au fil du temps que les couches d'abstraction qui le surplombent.

# Le concept de famille de MCU

- De nombreux fournisseurs de silicium ont des concepts de familles de dispositifs, et **STMicroelectronics (STM)** ne fait pas exception - les fiches de données sont généralement rédigées pour des familles entières de dispositifs.

- Les différences les plus notables entre les dispositifs d'une famille STM sont généralement la RAM/ROM et la taille du boîtier.

- Cependant, des périphériques supplémentaires sont également ajoutés aux appareils plus performants - par exemple, les boîtiers plus importants commenceront à inclure des contrôleurs de mémoire vive parallèle.

- Les appareils dotés d'une plus grande quantité de RAM/ROM comprendront des périphériques de temporisation plus performants, des périphériques de communication plus nombreux ou des périphériques spécifiques à un domaine, tels que des modules cryptographiques.

# Considérations relatives à la carte de développement

- Une carte de développement est une pièce de matériel que les ingénieurs utilisent au cours de la première phase de développement d'un projet.

- Les cartes de développement ne sont pas réservées aux MCU ; elles sont utiles pour de nombreux types de matériel, depuis les amplificateurs optoélectroniques jusqu'aux réseaux de cartes programmables (FPGA).

- Les cartes de développement MCU doivent offrir quelques fonctions clés :
  - Circuits auxiliaires, nécessaires à l'alimentation et au fonctionnement de l'unité centrale de commande (MCU)
  - Un moyen de programmer et de communiquer avec le MCU
  - Connecteurs pour une connexion aisée aux circuits externes
  - Éventuellement, quelques circuits intégrés embarqués utiles pour exercer certains des périphériques.

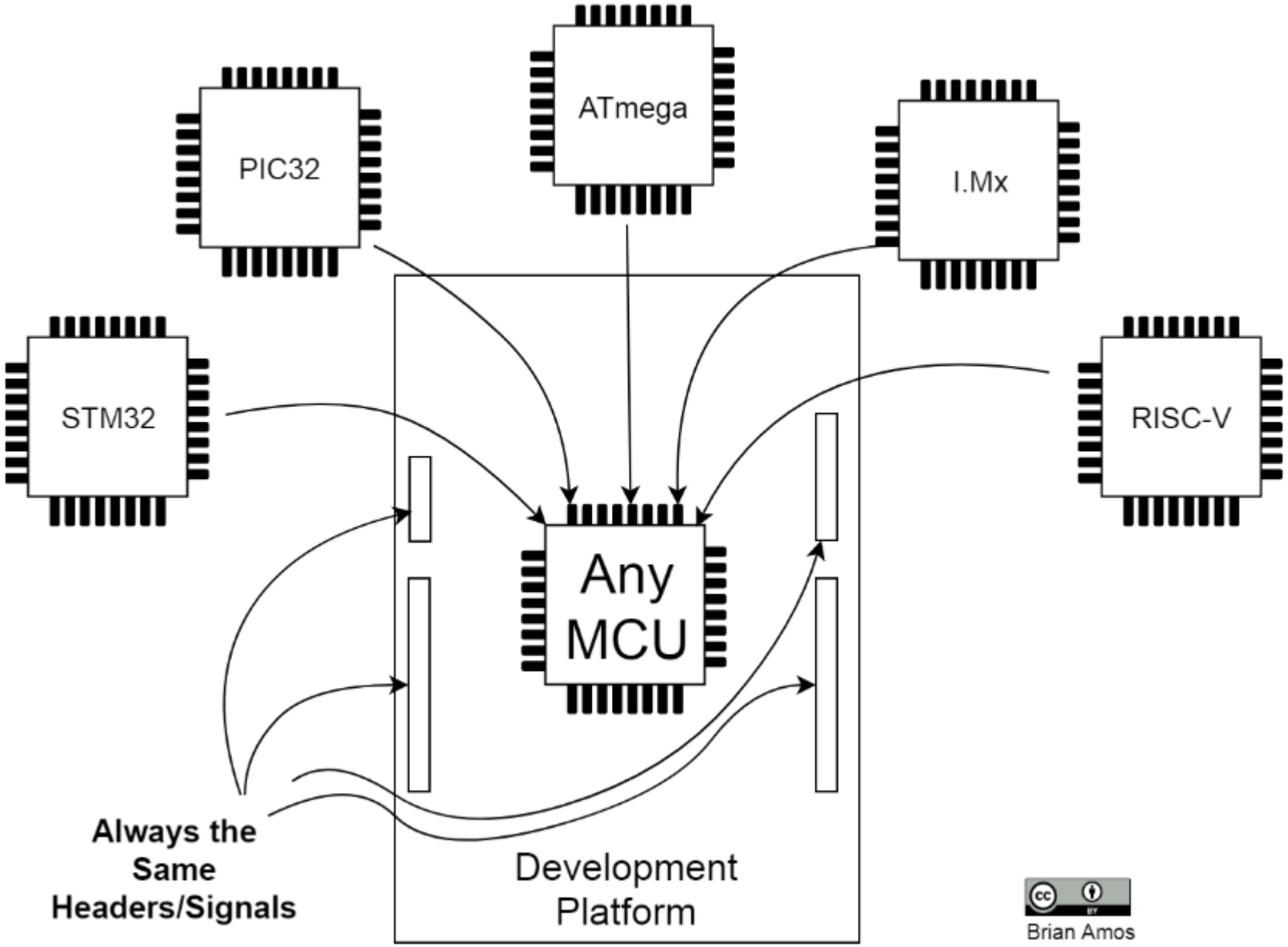
# Qu'est-ce qu'une plateforme de développement et pourquoi est-elle importante ?

- En ce qui nous concerne, une plateforme de développement est un écosystème de produits qui permet un haut degré d'abstraction entre plusieurs fournisseurs.
- L'objectif principal d'une plateforme est de fournir un grand nombre de fonctionnalités avec le moins d'efforts possible, ce qui est excellent lorsque l'objectif principal est de créer un prototype aussi rapidement que possible.

# Qu'est-ce qu'une plateforme de développement et pourquoi est-elle importante ?

- Afin de fournir un grand nombre de fonctionnalités à plusieurs fournisseurs, on tend à mettre l'accent sur les interfaces standardisées, la facilité d'utilisation et la flexibilité.
- Les écosystèmes se développent autour des plates-formes, en guidant doucement les utilisateurs de plates-formes vers les outils, les flux de travail, les ensembles de fonctionnalités du plus petit dénominateur commun et le matériel disponible.
- C'est une bonne chose si l'objectif est de produire une preuve de concept en un minimum de temps.
- Toutefois, si l'on veut trouver une solution de développement et de production à long terme en utilisant l'approche par plate-forme, la plate-forme devra probablement être de haute qualité, extrêmement bien établie et stable.

# Uniformité de la plate-forme



# Uniformité de la plate-forme

- En définitive, l'intérêt d'une plateforme est généralement de faciliter son utilisation afin qu'elle soit davantage adoptée.

- Cette accessibilité peut conduire à des interfaces communes (comme le montre le diagramme précédent), qui peuvent être un atout pour la productivité lors d'efforts de prototypage rapide, mais qui font abstraction de différences considérables dans le matériel sous-jacent.

- Souvent, les fonctionnalités spécifiques à la plate-forme sont si importantes que le code de la plate-forme se contente de virtualiser les interfaces pour les rendre plus accessibles, souvent à des coûts très élevés (c'est-à-dire en modifiant les bits PWM ou SPI pour qu'ils puissent être assignés à des broches prédéterminées spécifiques).

- Par conséquent, si vous choisissez d'utiliser une plate-forme pour évaluer du matériel (ou plus précisément un MCU), vous devez savoir que vous évaluez probablement la plate-forme et sa mise en œuvre sur un matériel donné, plutôt que le matériel lui-même.

# Kits d'évaluation

- Un **kit d'évaluation (eval kit)** se situe à l'autre extrémité du spectre en ce qui concerne les empreintes normalisées par rapport à l'attention portée au matériel lui-même.

- Les kits Eval sont généralement équipés du modèle de MCU le plus grand et le plus riche en fonctionnalités disponible et sont fabriqués dans le seul but de mettre en valeur ce matériel.

- Cela signifie qu'ils ne partageront généralement pas une empreinte ou des connecteurs communs entre les différents MCU cibles (voir le diagramme suivant), car chaque MCU a des caractéristiques primaires différentes et est destiné à un marché différent.

# Kits d'évaluation

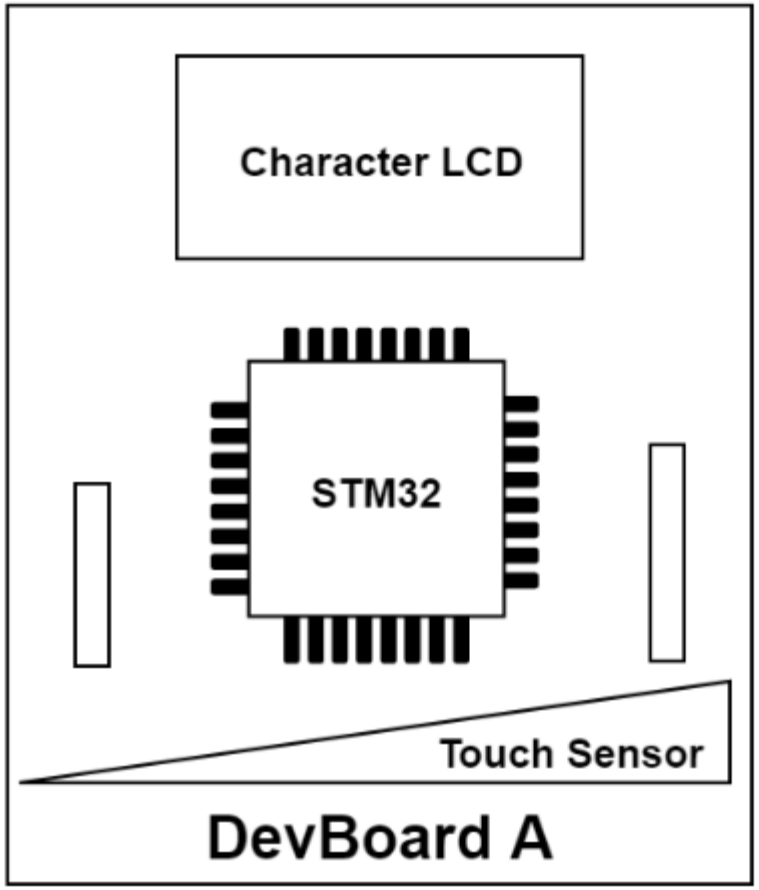
- Les cartes Eval comporteront autant de périphériques que possible répartis sur des connecteurs réels (tels que des cartes série, Ethernet, SD, un bus CAN et plusieurs USB).

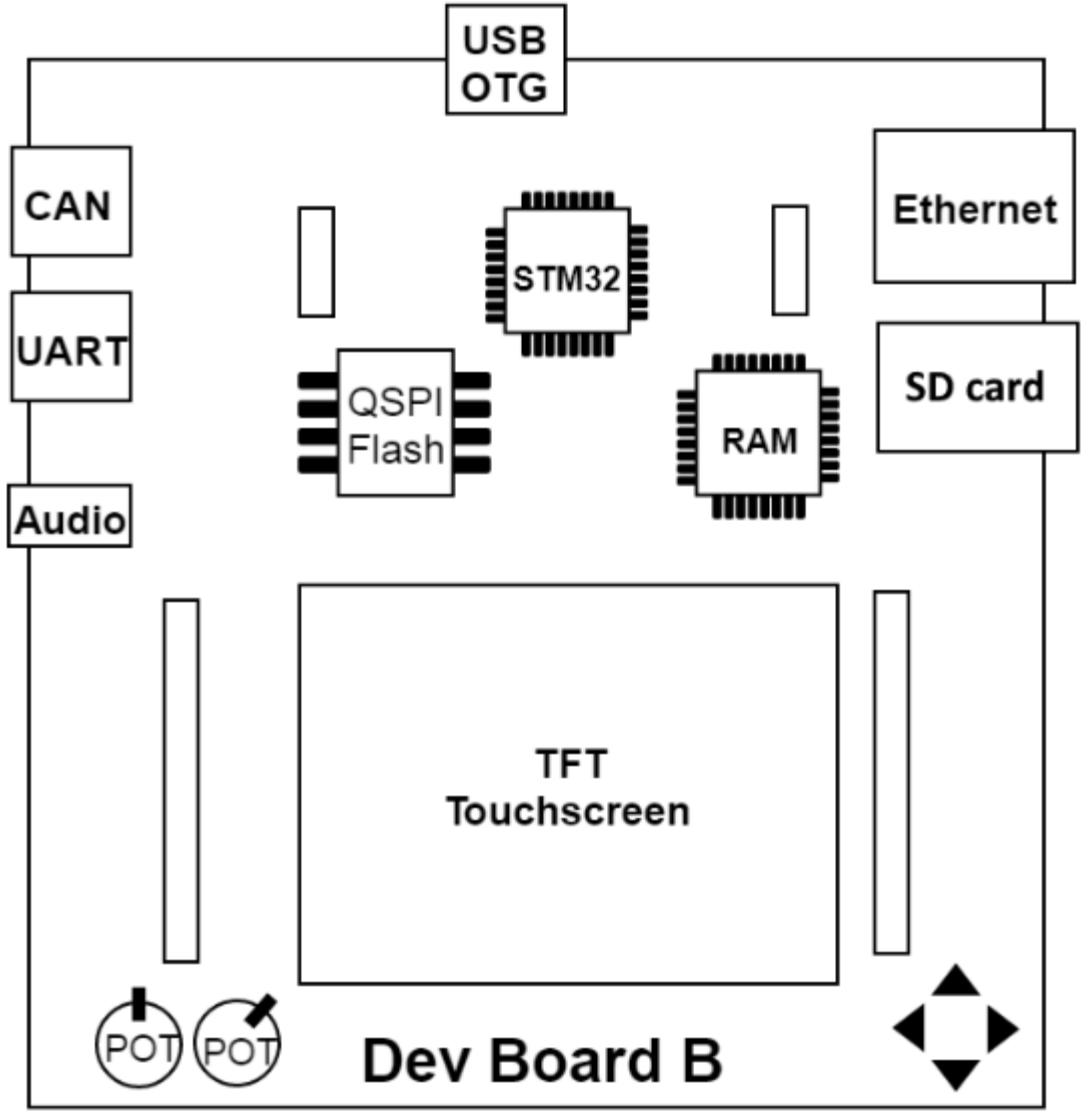
- Ils comprennent aussi généralement une série de périphériques, tels que la RAM, l'eMMC, des boutons, des curseurs, des potentiomètres, des écrans et des codecs audio pour piloter les haut-parleurs.

- Ils répartissent presque toujours toutes les broches intéressantes du MCU sur des connecteurs facilement accessibles, ce qui permet aux développeurs d'essayer rapidement toute configuration matérielle spécifique qu'ils peuvent imaginer.

- Les fabricants présentent généralement leurs autres silicones non-MCU sur les cartes des kits d'évaluation, dans le but d'augmenter les ventes de leurs propres produits.

# Différences entre les DevBoard



# Cartes de démonstration à faible coût

- Les cartes de démonstration à bas prix ont connu un véritable essor ces dernières années.
- Les prix ont considérablement baissé ; les fabricants vendent parfois des cartes de démonstration au même prix que le circuit intégré (bare IC ) qui y est installé (parfois, elles étaient même moins chères que l'achat de circuits individuels !)
- Contrairement aux plates-formes matérielles, ces cartes ont souvent des empreintes similaires, mais pas nécessairement les mêmes connecteurs ou brochages.

# Cartes de démonstration à faible coût

- Récemment, des cartes de démonstration à bas prix qui brouillent la frontière entre les plates-formes et les cartes de démonstration (également appelées cartes de démonstration) sont apparues sur le marché.

- Grâce à l'omniprésence d'Arduino®, la plupart des cartes bon marché disposent au moins d'un jeu de connecteurs Arduino compatibles avec les broches.

- Toutefois, la disponibilité d'en-têtes compatibles et le fait de disposer d'une carte de développement qui intègre pleinement un écosystème sont des choses très différentes.

- La carte de démonstration peut ne pas avoir de logiciel pour accompagner ces en-têtes ; ce n'est pas parce que le matériel existe et peut être branché sur la carte que vous obtiendrez automatiquement des bibliothèques compatibles avec le MCU cible pour piloter le matériel.

- Cela ne les rend pas incompatibles, mais la quantité d'efforts à fournir pour obtenir quelque chose de fonctionnel sera beaucoup plus importante que le simple fait de brancher une carte et de suivre une démo "hello world".

# Cartes de démonstration à faible coût

- Certains fabricants créent également leurs propres en-têtes standardisés qui sont communs aux cartes de démonstration, ce qui est au moins utile lors de la migration entre différentes familles de produits (mais est évidemment limité à ce fabricant).

- Les en-têtes normalisés ST Nucleo et NXP Freedom en sont des exemples.

- Dans un souci de convivialité, ces tableaux sont généralement compatibles avec le mBed.

- Maintenant que vous avez découvert les différents types de cartes de développement, nous allons nous pencher plus en détail sur la gamme de microcontrôleurs d'un seul fabricant, la gamme STM32.

# Présentation de la gamme de produits STM32

- Au cours des dernières années, STM a développé une gamme assez large de MCU.
- Nous verrons ici comment quelques-uns des principaux éléments de ce portefeuille s'inscrivent dans le cadre des considérations exposées dans la section de ce chapitre consacrée aux MCU.
- La plupart des autres vendeurs classent également leurs produits par grands segments, ce qui tend à faciliter le processus de sélection.

# Grand public

- La gamme STM32 a débuté en 2007 avec le STM32F1, un MCU basé sur le Cortex-M3.
- Cette partie de la gamme de produits est destinée à la majorité des applications à haut volume, où le coût et la performance doivent être équilibrés, avec des applications peu complexes.
- Les STM32F0 et STM32G0 de la gamme sont des dispositifs basés sur le Cortex-M0 et le Cortex-M0+ qui sont destinés aux applications à faible coût.
- Le STM32F1 original est basé sur le Cortex-M3 et dispose d'un très large éventail de fonctionnalités, mais il commence à montrer des signes de fatigue lorsque ses performances sont comparées à celles d'autres appareils.

# Grand public

- Le STM32F3 était la première tentative de STM de fournir des périphériques analogiques intégrés de haute précision (d'autres fournisseurs proposent des composants analogiques de plus haute précision que STM) ; cependant, la gamme n'offre pas de véritables performances analogiques.

- Il comprend un CAN sigma delta de 16 bits, mais le **nombre effectif de bits (ENOB)** n'est que de 14 bits, ce qui est légèrement mieux que les périphériques CAN à **approximation successive (SAR)** de 12 bits de STM, qui sont le plus souvent inclus.

# Grand public

- La famille STM32G4, plus récente et basée sur le Cortex-M4+, dispose du plus grand nombre de périphériques analogiques, y compris de nombreux **amplificateurs opératifs à gain programmable (PGA)**, des CNA, des CAN et plusieurs comparateurs, mais aucun CAN intégré de haute précision n'est disponible à l'heure où nous écrivons ces lignes.

- Les familles STM32G0 et STM32G4 devraient voir leur offre s'étoffer jusqu'en 2020. Il s'agit de deux gammes plus récentes et STM devrait commencer à les compléter avec de nombreux autres dispositifs.

# STM haute performance

- Les MCU haute performance de STM ont commencé avec le STM32F4 basé sur le Cortex-M4.

- Le Cortex-M4 est très similaire au Cortex-M3, mais il inclut (en option) des instructions FPU et DSP 32 bits matérielles, qui sont toutes deux présentes sur tous les dispositifs STM3F4.

- Tous les appareils de la gamme haute performance sont capables de piloter confortablement des interfaces graphiques très attrayantes sur des écrans sans contrôleur, pour autant que la RAM nécessaire soit disponible (généralement externe au MCU).

- L'accélération fournie par divers périphériques matériels permet de décharger une bonne partie des efforts liés à la communication avec l'écran sur divers périphériques, de sorte que l'unité centrale puisse consacrer du temps à d'autres tâches.

# Deux membres majeurs de la famille STM32

| MCU line | CPU | Features |
|---|---|---|
| STM32H7 | Cortex-M7 (480 MHz)<br>Cortex-M4 (240 MHz) (opt) | Highest performance MCU from STM<br>64-bit FPU and extended DSP instructions |
| STM32F7 | Cortex-M7 (216 MHz) | 64-bit FPU and extended DSP instructions |
| STM32F2 | Cortex-M3 | Offers trade-off for performance versus cost<br>High level of integration (camera interface and USB OTG) |

- Cette catégorie d'unités MCU peut être considérée comme des unités MCU croisées.
- Ils sont suffisamment puissants pour être utilisés dans certaines applications traditionnellement réservées aux processeurs complets, tout en conservant la facilité d'utilisation d'un MCU.

# L'approche hétérogène multicœur

- En 2019, STM a lancé la gamme STM32MP, sa première entrée dans le domaine des processeurs d'application.

- Cette famille propose un Cortex-A7 à un ou deux cœurs à 650 MHz, ainsi qu'un Cortex-M4 à 209 MHz.

- STM semble viser la partie du marché à haut volume et à faible coût avec ces solutions en se concentrant sur un nombre de noyaux plus faible et des boîtiers qui nécessitent moins de couches de PCB en utilisant des techniques de fabrication moins onéreuses.

# L'approche hétérogène multicœur

- D'un point de vue logiciel, la principale différence entre les offres MCU de STM et la nouvelle unité de microprocesseur (MPU) réside dans le fait que, les MPU étant dotées d'**unités de gestion de mémoire (MMU),** elles sont capables d'exécuter un système d'exploitation complet via des noyaux Linux principaux, ce qui ouvre la voie à un écosystème entièrement différent de logiciels libres (que vous n'avez pas besoin d'écrire personnellement).

- L'approche hétérogène multicœur permet aux concepteurs de diviser des parties d'une conception et de les résoudre dans le domaine qui leur convient le mieux.

- Par exemple, un contrôleur de processus spécialisé doté d'une interface graphique et d'une capacité de mise en réseau pourrait utiliser le Cortex-A7 pour tirer parti de Linux et accéder au cadre Qt et aux piles de mise en réseau complexes, tout en utilisant le Cortex-M4 pour tous les aspects du contrôle en temps réel.

# La faible consommation d'énergie de STM

- La gamme basse consommation de STM est destinée aux appareils alimentés par batterie.
- Voici une comparaison rapide entre les différents membres de la famille :

| MCU version | CPU | Description |
|---|---|---|
| STM32L | Cortex-M0 | Lowest performance and RAM and ROM space of the family, but offers fairly good efficiency. |
| STM32L1 | Cortex-M3 | Higher ROM capacity and offers faster performance with the trade-off of increased power consumption. |
| STM32L4 and STM32L4+ | Cortex-M4 | Have increased number-crunching capability (the STM32L4+ also offers faster clock speeds and larger internal flash storage). |
| STM32L5 | Cortex-M33 | Blends performance and power and incorporates the latest ARM v8 architecture. Cortex-M33 offers additional security features, such as Trust.Zone, and executes instructions more efficiently than Cortex-M4, which allows for some additional performance while still keeping power consumption in check. |

# Sans fil

- Le STM32WB met en œuvre un Cortex-M4 avec un Cortex-M0+ dédié pour exécuter une pile Bluetooth BLE.

- Cette ligne de produits offre de nombreuses possibilités d'intégration, depuis la mémoire flash et la mémoire vive jusqu'aux périphériques analogiques à signaux mixtes, en passant par les capteurs tactiles et les écrans LCD à petit segment.

- Diverses fonctions de sécurité, telles que des algorithmes cryptographiques et un générateur de nombres aléatoires, sont également présentes.

- Comme la certification FCC est requise pour les produits utilisant cette famille d'appareils, ces derniers seront les plus utiles pour les applications à haut volume.

- Avec une large gamme de MCU à choisir, STM a probablement quelque chose que nous pouvons utiliser pour expérimenter un RTOS ! Il est temps de passer au choix de la carte de développement à utiliser.

# Comment notre carte de développement a-t-elle été sélectionnée ?

- Maintenant que nous avons abordé les considérations importantes pour le choix d'un MCU et les types généraux de matériel de développement, voyons comment a été choisie la carte de développement qui sera utilisée dans ce cours.

- STM est le seul fabricant que nous évaluons afin de limiter les exemples à quelque chose de facilement assimilable.

- Dans le cadre de l'ingénierie d'un produit, il incombe au concepteur de jeter un regard neuf sur tous les fournisseurs possibles.

- Bien que chacun ait sa propre méthode pour effectuer des recherches inter-fournisseurs, il est facile d'utiliser les sites web des distributeurs.

# Comment notre carte de développement a-t-elle été sélectionnée ?

- Grâce à des sites web de distributeurs bien conçus et orientés vers le prototypage (tels que Digikey et Mouser), un ingénieur peut effectuer des recherches paramétriques et des comparaisons entre de nombreux fournisseurs différents.

- L'inconvénient de cette approche est que les recherches sont limitées aux gammes de produits proposées par le distributeur concerné.

- Un autre inconvénient potentiel est que les résultats de la recherche paramétrique sont à la merci de la précision de la saisie des données et de la catégorisation du distributeur.

- L'avantage d'utiliser directement le site web d'un distributeur est que de nombreux vendeurs différents sont réunis en un seul endroit, que la disponibilité des produits peut être vérifiée immédiatement et que les prix semi-réels sont facilement visibles et filtrés.

# Exigences

- Jusqu'à présent, nous avons couvert toutes les considérations relatives au choix d'un MCU pour un projet à un niveau théorique.
- Il est maintenant temps de mettre tout cela en pratique et de sélectionner le MCU qui sera utilisé pour tous les exercices pratiques.
- Les éléments indispensables - c'est-à-dire les exigences - sont les suivants :
  - Le MCU cible sur votre carte de développement sera un CPU STM32 basé sur le Cortex-M.
  - Le MCU cible doit disposer d'une unité de protection de la mémoire (MPU).
  - La carte de développement doit avoir un moyen visible d'afficher l'état.
  - Un coût relativement faible (< 50 USD).

# Résumé

- Ce chapitre a présenté les files d'attente, les sémaphores et les mutex.
- Quelques cas d'utilisation courante de chacun de ces éléments de base pour les applications RTOS ont également été discutés à un niveau élevé et certains des comportements plus subtils de chacun ont été mis en évidence.
- Les diagrammes présentés dans ce chapitre devraient servir de point de référence auquel nous pourrons revenir lorsque nous passerons à des exemples réels plus complexes dans les chapitres suivants.
- Nous avons maintenant abordé certains des concepts fondamentaux du RTOS.
- Dans le prochain chapitre, nous nous intéresserons à une autre étape très importante du développement d'un système temps réel solide.
- Cette étape influe sur l'efficacité de l'exécution des microprogrammes et a un impact majeur sur les performances du système - sélection du MCU.

# Exigences

- Ce chapitre a abordé de nombreuses considérations relatives au choix d'un MCU approprié et nous avons expliqué le processus de sélection en passant en revue les compromis effectués lors de la sélection de la carte de développement utilisée dans cet exposé.

- Vous devriez maintenant comprendre l'importance de la sélection des MCU et disposer de suffisamment d'informations pour commencer à rechercher et à sélectionner des MCU pour vos projets.

- Si vous faites partie d'une équipe pluridisciplinaire, vous serez mieux placé pour discuter avec vos pairs des avantages et inconvénients de l'utilisation de différents MCU pour une application donnée.

- Dans le chapitre suivant, nous ferons un exercice similaire, en comparant différentes classes d'environnements de développement intégré (IDE) et en choisissant un IDE approprié pour coder les exercices que vous trouverez dans ce livre, en commençant par le chapitre 7, L'ordonnanceur FreeRTOS.