# carte MadMachine SwiftIO

## un langage récent pour du matériel tout nouveau

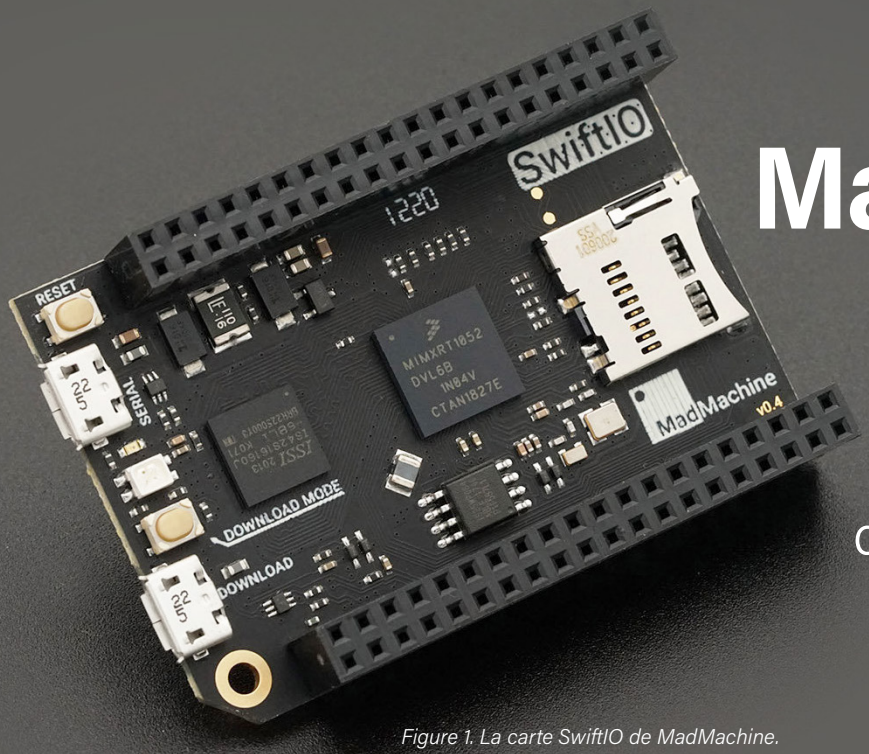


*Figure 1. La carte SwiftIO de MadMachine.*

**Mathias Claußen** (Elektor)

Le langage de programmation Swift qui est bien établi sur les appareils Apple peut désormais être utilisé sur un microcontrôleur. Jetons un œil à la carte SwiftIO qui intègre un processeur ARM Cortex-M7 et 32 Mo de RAM.

MadMachine SwiftIO est une carte de développement au format BeagleBone Black. Ce qui la rend attrayante pour moi est le fait que nous avons un ensemble de matériel intéressant combiné à un cadre logiciel peu commun. Le code peut être écrit en Swift, un langage de programmation bien établi sur les appareils Apple.

### Le matériel

Alors que le langage Swift est destiné au matériel Apple, la carte SwiftIO (**fig. 1**) de MadMachine introduit Swift dans le monde de l'embarqué.

Les principaux composants utilisés sur la carte sont :

- Processeur NXP i.MX RT1052 avec un cœur ARM Cortex-M7 cadencé à 600 MHz
- Flash QSPI externe de 8 Mo
- 32 Mo de SDRAM
- Connecteur pour carte micro-SD, supportant les cartes standard et haute capacité

Avec le i.MX RT1052, la carte est animée par un microcontrôleur robuste, dont les caractéristiques sont détaillées sur la **figure 2**. Elle offre en plus :

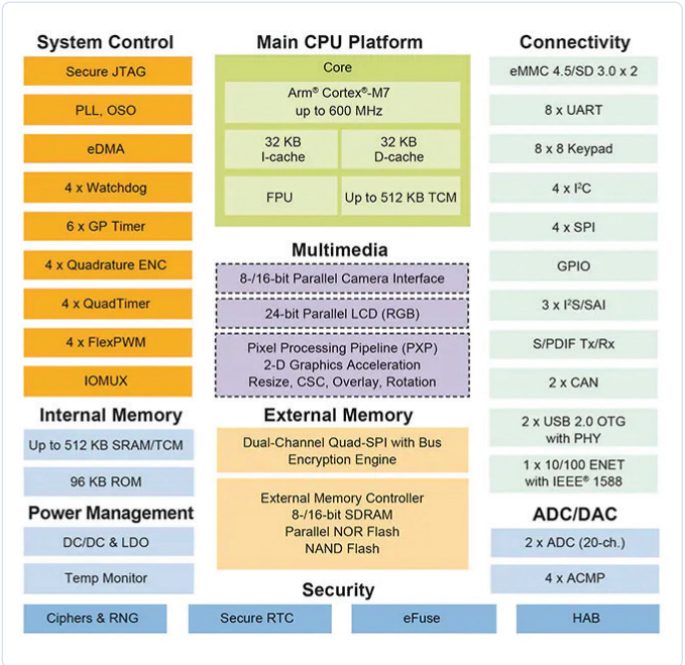


*Figure 2. Vue d'ensemble du i.MX RT1052 (source : NXP).*

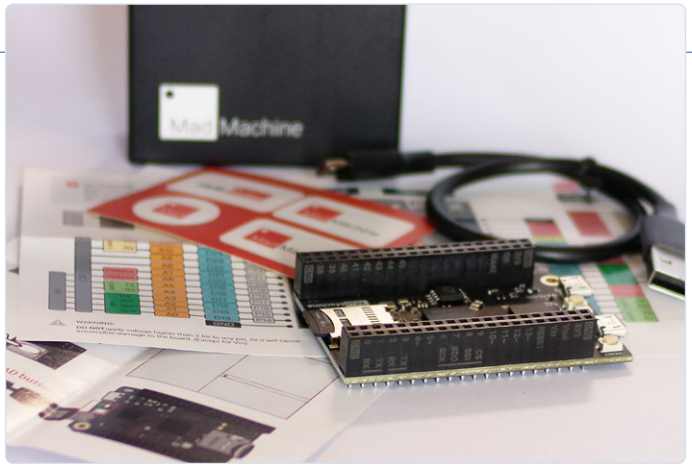


*Figure 3. Contenu de la boîte.*

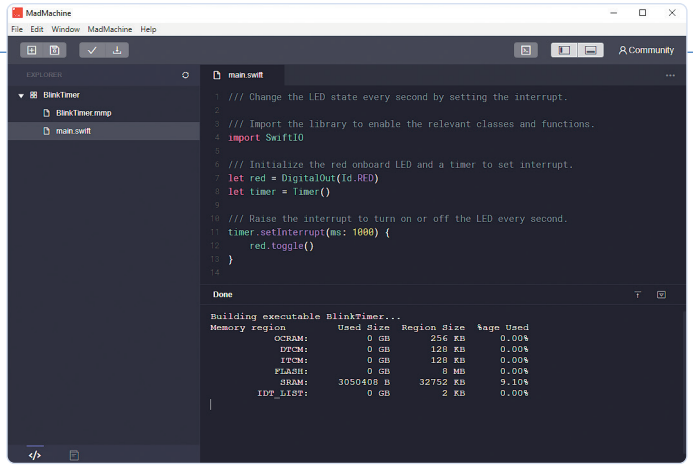


*Figure 4. L'EDI de MadMachine.*

- 46 GPIO (E/S à usage général)
- 12 canaux A/N (12 bits)
- 14 canaux MLI
- 4 UART
- 2 bus CAN 2.0B
- 2 bus I²C (3,4 MHz max)
- Moteur graphique 2D
- Sortie vidéo RVB (1366 × 768 WXGA max)
- Entrée caméra (CSI 8, 16, 24 bits)

Il ne s'agit pas d'une liste exhaustive des caractéristiques, mais elle donne une idée du matériel. Même l'emballage est compact. Vous recevez une carte, un câble USB, un jeu d'autocollants et la carte SwiftIO avec une carte SD insérée (**fig. 3**). Tout ce dont vous avez besoin pour commencer.

## L'EDI

En ce qui concerne le développement de logiciels, MadMachine propose un EDI pour SwiftIO, téléchargeable sur son site Web (voir à quoi il ressemble sur la **figure 4**). Il est disponible pour Windows et macOS, ce qui est une bonne nouvelle pour les utilisateurs Apple. Si vous n'aimez pas l'EDI, vous pouvez utiliser les outils ILC (interface en ligne de commande) fournis pour cette carte ou les associer à XCode ou Visual Studio Code pour votre prochain projet. MadMachines fournit un tutoriel sur la façon de les utiliser.

## Il suffit de copier

La carte comprend dans sa mémoire flash SPI un chargeur d'amorçage qui copiera votre application dans la SDRAM au démarrage puis l'exécutera. Cela facilite la reprogrammation de la carte. Le programmateur intégré permet d'écrire votre code sur la carte micro-SD en l'affichant comme périphérique de stockage de masse. De plus, vous pouvez simplement retirer la carte et écrire votre nouveau micrologiciel directement sur celle-ci.

## SwiftIO

Ce qui a retenu mon attention, c'est le logiciel choisi. Le cadriciel SwiftIO (**fig. 5**) utilise le système d'exploitation en temps réel (RTOS) Zephyr. L'accès au matériel de bas niveau, comme les GPIO, les fonctions analogiques, la MLI, l'I²C et ainsi de suite, passe par une couche d'abstraction, il devrait donc être facile d'écrire son propre logiciel sans avoir à s'occuper de tous les détails du microcontrôleur i.MX RT1052. De plus, cela concerne également l'accès à la SDRAM et son initialisation ainsi que la configuration de la flash SPI. Quant à Swift, le langage de programmation retenu pour cette carte, il est couramment utilisé sur les systèmes Apple et on le retrouve principalement dans l'écosystème Apple. Il n'est pas limité à ce monde et convient pour développer des applications pour Linux et, depuis peu, pour Windows. Comme Swift offre des éléments et des concepts modernes, le fait qu'il soit disponible sur un microcontrôleur permet d'approfondir ses connaissances Swift. Si vous vous demandez à quoi ressemble un programme SwiftIO, le **listage 1** montre comment faire clignoter une LED. Le **listage 2** montre comment utiliser la MLI pour commander sa luminosité. Des exemples plus avancés sont déjà inclus, afin que vous puissiez commencer à explorer SwiftIO.

## Assistance et informations supplémentaires

Pour cette carte, vous trouverez facilement de l'aide sur le serveur Discord de MadMachine. Posez vos questions (même les plus techniques) dans les chats appropriés dans un langage amical et poli ; personne ne veut de trolls. De plus, s'il vous manque des documents, n'hésitez pas à les demander.

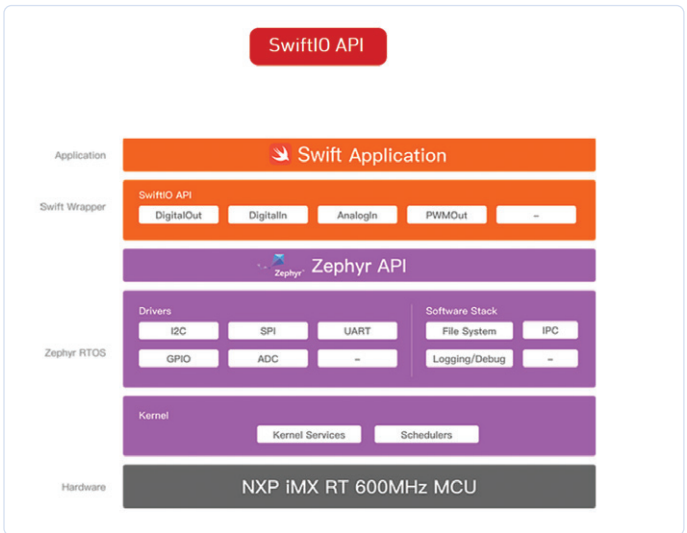


*Figure 5. Pile logicielle (source : MadMachine.io).*

**Listage 1. Faire clignoter une LED.**

```
/// Changer l'état d'une LED toutes les secondes avec une interruption.
/// Importer la bibliothèque pour accéder aux classes et fonctions nécessaires.
  import SwiftIO
/// Initialiser la LED rouge et la minuterie pour déclencher l'interruption.
  let red = DigitalOut(Id.RED)
  let timer = Timer()
/// Déclenchez l'interruption pour allumer ou éteindre la LED toutes les secondes.
  timer.setInterrupt(ms: 1000) {
     red.toggle()
}

  while true {
}
```

## Dernières réflexions

Si vous cherchez une carte compatible Arduino ou qui peut être programmée en C/C++, oubliez SwiftIO. En outre, son prix est plus élevé que celui de la plupart des autres cartes de développement. Toutefois, si vous considérez que vous disposez d'une assistance technique et d'un langage de programmation moderne, et que le développement est possible sur les principaux systèmes d'exploitation, c'est peut-être une option.

(210297-04)

**Listage 2. Avec MLI.**

```
/// Faire varier l'intensité d'une LED en modifiant le rapport cyclique du signal MLI.
/// Importer la bibliothèque pour accéder aux classes et fonctions nécessaires.
  import SwiftIO
/// Initialiser la sortie MLI connectée à la LED.
  let led = PWMOut(Id.PWM0A)
/// Initialiser une variable pour stocker le rapport cyclique.
  var value: Float = 0.0
/// Faire varier la luminosité du min au max, et inversement.
  while true {
     // Allumer la LED sur une durée de deux secondes.
     while value <= 1.0 {
        led.setDutycycle(value)
        sleep(ms: 20)
        value += 0.01
     }
     // Garder le rapport cyclique entre 0,0 et 1,0.
     value = 1.0
     // Eteindre la LED sur une durée de deux secondes.
     while value >= 0 {
        led.setDutycycle(value)
        sleep(ms: 20)
        value -= 0.01
     }
     // Garder le rapport cyclique entre 0,0 et 1,0.
     value = 0.0
}
```


**Contributeurs**

Conception et texte : **Mathias Claußen**

Rédaction : **Jens Nickel**

Mise en page : **Giel Dols**


**Des questions, des commentaires ?**

Envoyez un courriel à l'auteur (mathias.claussen@elektor.com) ou contactez Elektor (redaction@elektor.fr).

## Produits

- **SwiftIO – carte à microcontrôleur, compatible avec le langage Swift**
  www.elektor.fr/swiftio-swift-based-microcontroller-board

## LIENS


[1] **MadMachine** : https://www.madmachine.io/

[2] **Guide pour l'interface en ligne de commande, MadMachine** : https://resources.madmachine.io/about-our-project#how-does-the-building-procedure-work

[3] **Serveur Discord MadMachine** : https://discord.gg/zZ9bFHK

[4] **Ressources de MadMachine** : https://resources.madmachine.io/