intelligence artificielle pour débutants (1)

Reconnaissance d'objets à l'aide de la carte Maixduino

Walter Trojan (Allemagne)

Il n'y a pas de sujet plus brûlant en informatique que l'intelligence artificielle (IA). La carte *Maixduino* est une bonne introduction peu coûteuse, à la reconnaissance de la parole et de l'image ainsi qu'à d'autres aspects de l'IA. Équipée d'une caméra et d'un petit afficheur LCD, elle ne coûte qu'environ 30 €. Elle est programmable dans le célèbre IDE Arduino Le premier épisode de cette série présente la richesse matérielle du Maixduino et propose quelques programmes de démonstration. L'un d'entre eux reconnaît 1000 objets différents !

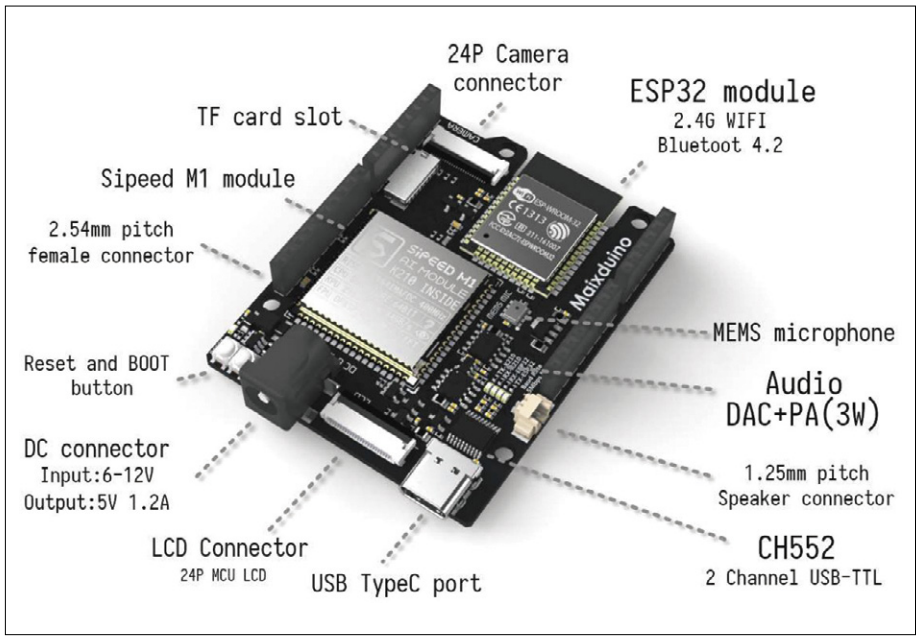

Figure 1. Le matériel disponible sur Maixduino.

L'utilisation de l'intelligence artificielle progresse partout. On trouve de l'IA dans de nombreuses applications. Pour l'année en cours, les ventes mondiales d'applications d'entreprise dans le domaine de l'IA devraient atteindre 4,8 milliards de dollars US, et en 2025, ce chiffre devrait dépasser les 30 milliards de dollars US. L'expertise en IA est très recherchée. Toute personne dotée de connaissances dans ce domaine peut s'attendre à de meilleures perspectives d'emploi. Il est donc utile de prendre le temps de se familiariser avec le sujet et les nouveaux horizons qu'il ouvre. Ce qui ne gâche rien, c'est qu'en plus c'est très amusant ! On entend par intelligence artificielle, entre autres, la reproduction d'un comportement *intelligent*. L'apprentissage *machine* est un domaine particulier, dans lequel une application n'est plus pensée dans tous ses détails par le développeur ; elle est un cadre de programme d'application universelle, sous la forme d'un réseau neuronal (NN) qui s'enseigne lui-même les fonctions nécessaires à l'aide de nombreuses données d'exemple. Une catégorie particulière de ce programme est l'apprentissage *approfondi*, dans lequel des structures de programme complexes et optimisées sont utilisées pour améliorer encore les résultats obtenus. Nous reviendrons sur cette terminologie dans la suite de cette série d'articles.

Cependant, l'IA n'est pas un sujet nouveau : ses débuts remontent aux années 1950. La percée récente résulte de l'augmentation vertigineuse de la puissance de calcul disponible. Des cartes graphiques avec des centaines de cœurs de processeur fonctionnant en parallèle. Des puces d'IA hautement spécialisées ont permis de mettre en place et de former de véritables systèmes d'IA. Cela a ouvert la porte à la reconnaissance automatique de la parole, utilisée dans de nombreux assistants personnels, et à la reconnaissance d'images d'objets de toutes sortes. L'IA joue également un rôle important dans la conduite de véhicules autonomes. Dans certains domaines, l'IA a dépassé les capacités humaines : des systèmes d'IA peuvent battre les meilleurs champions aux échecs et au go, et détecter les tumeurs de manière plus fiable que les spécialistes humains. Les systèmes ne sont supérieurs aux humains que dans les domaines particuliers dans lesquels ils ont été formés, et les domaines dans lesquels les systèmes sont formés continueront à être choisis par les humains.

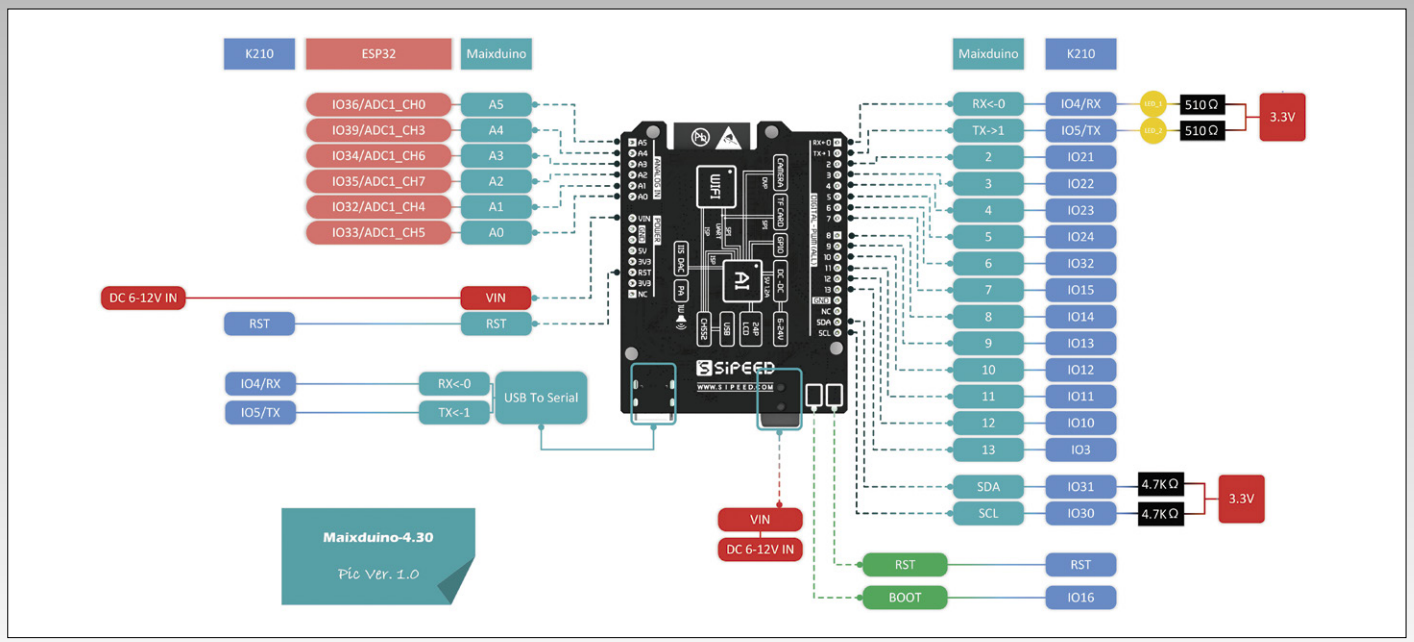

Figure 2. Brochage de Maixduino.

Une initiation qui ne ruinera personne

Maixduino est une manière peu coûteuse de s'initier à l'IA. Le MAix BiT, qui comprend la carte, l'appareil photo et le petit afficheur LCD, est disponible chez Elektor (cf l'encadré en fin d'article). La carte est du format Arduino Uno, mais le matériel est beaucoup plus complet. Elle est fabriquée par Sipeed. Parmi les alternatives au Maixduino, on trouve le Nvidia Jetson Nano, le ROCK PI N10 Model A, le Intel Neural Compute Stick 2 et d'autres, mais ils sont tous beaucoup plus chers.

Dans ce 1er épisode, examinons le riche environnement matériel de Maixduino et sa programmation dans l'IDE Arduino. Outre quelques applications typiques de l'Arduino, nous examinerons comment utiliser la caméra et l'afficheur. Nous nous quitterons sur une démonstration de reconnaissance d'objets.

Dans les 2e et 3e volets, nous approfondirons le thème de l'intelligence artificielle, nous décrirons la structure des réseaux neuronaux, nous installerons MicroPython et l'IDE qui l'accompagne et nous montrerons comment fonctionne la reconnaissance faciale. Nous examinerons également comment programmer vos propres applications d'IA et comment communiquer avec l'internet.

Super Arduino

Les caractéristiques de Maixduino font saliver ! Sa taille et sa construction sont alignées sur celles de l'Arduino Uno, mais vous remarquerez immédiatement la densité plus élevée des composants et les nombreuses connexions supplémentaires disponibles. Au cœur de la carte, il y a deux grands modules. Le premier est le processeur Sipeed M1 AI basé sur le dispositif Kendryte K210, dont nous examinerons plus tard le fonctionnement. Le second est un module ESP32 pour communiquer sur WLAN et Bluetooth et pour acquérir des signaux analogiques. L'ESP32 contient deux cœurs de processeur cadencés à 240 MHz et offre donc à lui seul une puissance de traitement considérable, et il peut être utilisé pour décharger les processeurs principaux de leurs fonctions de communication. L'ESP32 a déjà été bien présenté dans Elektor, je n'entrerai donc pas dans les détails.

Les connecteurs correspondent à ceux de l'Arduino original, et l'affectation des broches est largement la même. Attention, les entrées et les sorties sont conçues pour fonctionner sous 3,3 V, voire 1,8 V. **L'application aux circuits d'entrée d'une tension de 5 V serait irrémédiablement destructrice !**

La **figure 1** donne plus de détails à ce sujet. Deux connecteurs à 24 broches sont prévues pour l'interface avec la caméra et l'afficheur LCD. Une carte microSD peut être utilisée pour étendre la capacité disposition de stockage du Maixduino. Le connecteur USB, de type C, est utilisé pour la programmation et le contrôle. Pour le traitement des données audio, la carte comprend un microphone numérique et d'un convertisseur N/A suivi d'un ampli audio de 3 W. Le matériel riche et varié de la carte vous permettra de vous lancer dans toutes sortes de projets sans recours à des cartes supplémentaires.

Le brochage de la carte (**fig. 2**) ressemble beaucoup à l'original. La tension d'alimentation peut être fournie par l'entrée DC ou en utilisant la broche V_{IN} entre 6 V et 12 V. On peut aussi l'alimenter sous 5 V par le connecteur USB. Six des GPIO de l'ESP32 sont accessibles (32 à 36 et 39), utilisables comme entrées analogiques A0 à A5. Sur le bord opposé de la carte se trouvent les entrées et les sorties du module K210. Celles-ci répondent aux commandes Arduino habituelles, le numéro de broche Arduino étant le numéro du bit d'entrée ou de sortie du K210.

```
#define LED 12 // K210 IO12,
   Maixduino broche 10
pinMode(LED, OUTPUT) ; //
   configurer le port comme sortie
digitalWrite(LED, HIGH) ; // port
   au niveau haut (3,3 V)
```

La broche RST fonctionner sous 1,8 V. Il est hors de question d'y appliquer une tension supérieure. Les sorties IO36 à IO47 sont également conçues pour un fonctionnement sous basse tension : elles ne sont pas sur les connecteurs, mais

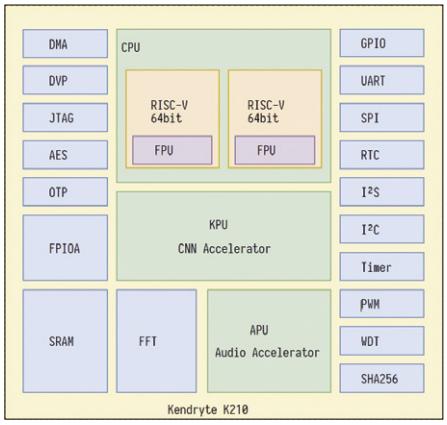

Figure 3. Synoptique des fonctions du Kendryte K210.

utilisées en interne, p. ex. pour piloter l'afficheur LCD. Les ports série RX et TX et l'interface I²C sont équipés des résistances de polarisation appropriées.
Malheureusement, la fiche technique du K210 ne donne pas le courant de sortie maximal disponible. Il devrait y avoir de quoi piloter une LED (<10 mA) ; pour des courants de plus forte intensité, prévoir un étage de puissance.

La grande marmite : Kendryte K210

Au cœur du Maixduino, le SoC K210 (système sur puce) de la société chinoise Kendryte, en technologie 28 nm à faible consommation, disponible depuis septembre 2018. Pour une charge de travail normale, ses deux cœurs de processeur à 64 bits sont cadencés à 400 MHz, une fréquence qui au besoin peut être forcée jusqu'à 800 MHz. Tout ceci répond à la spécification ouverte RISC-V pour éviter au fabricant les frais de licence ARM et contribue ainsi au faible coût total de l'appareil. Les deux processeurs sont équipés d'un FPU (unité de calcul à virgule flottante) qui fonctionne à la fois en simple et en double précision (**fig. 3**).
La particularité du K210 est son unité de traitement des connaissances KPU (avec le k de *knowledge*) pour construire et exécuter des réseaux de neurones.

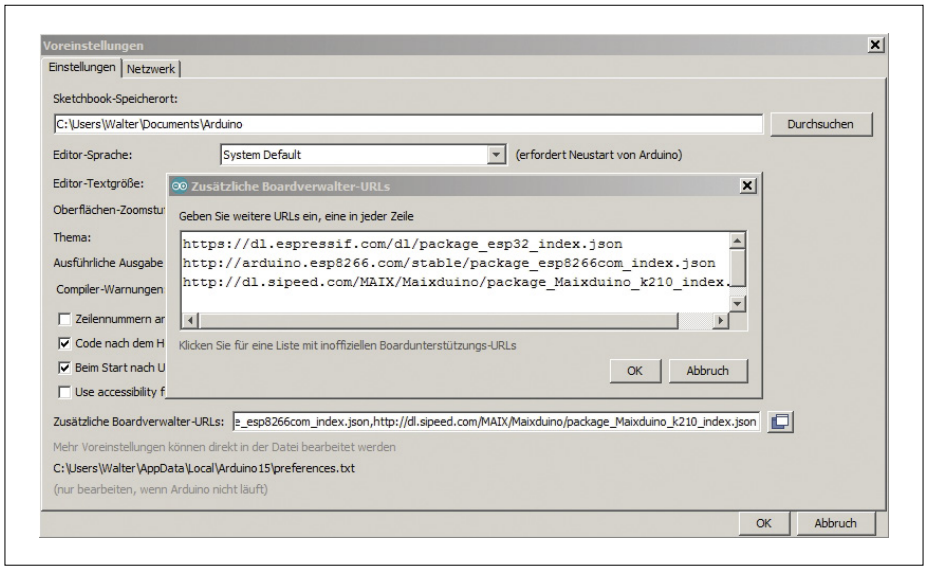

Figure 4. Reconnaissance et configuration de Maixduino.

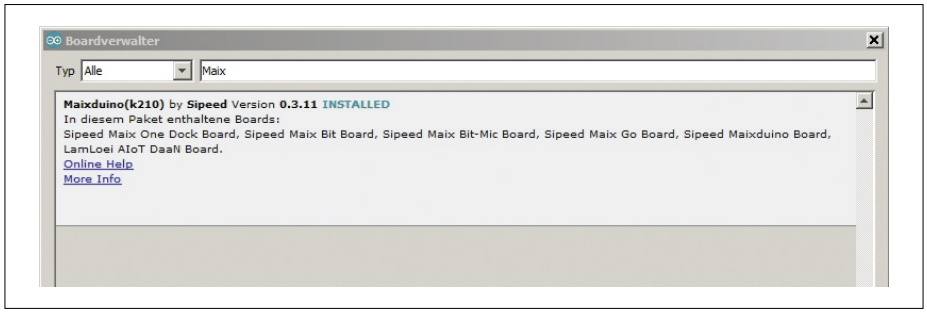

Figure 5. Installation des outils de base de Maixduino.

La puissance de calcul totale disponible est étonnante à ce prix : 0,46 Tops, soit 460 milliards d'opérations par seconde. Avec le surfréquençage de l'horloge, ce nombre déjà astronomique pourra même être doublé, ce qui permet de reconnaître jusqu'à 60 objets par seconde. La performance exceptionnelle est obtenue grâce à 64 unités arithmétiques en parallèle et un bus de 576 bits. Comparée à celle d'autres systèmes d'IA, la dissipation de puissance de seulement 0,3 W est très faible : *Nvidia* recommande d'utiliser une alimentation 5 V 4 A (donc 20 W) pour son *Jetson Nano*, comparable par sa puissance de calcul de 0,4 Tops.
La KPU peut mettre en œuvre des architectures avancées de réseaux neuronaux, y compris des réseaux convolutifs. Ceux-ci ont une structure de filtrage particulièrement efficace dans les applications de traitement d'images : nous y revenons bientôt dans le deuxième volet de cette série.
La mémoire principale incluse dans le SoC a une capacité de 8 Mo, divisée en 2 Mo pour les processeurs principaux et 6 Mo pour la KPU. Jusqu'à 5,9 Mo sont donc disponibles pour stocker la configuration du réseau neuronal, ce qui est suffisant pour mettre en œuvre un réseau de taille moyenne.
La puce K210 a d'autres atouts matériels. Notamment une unité de traitement audio (APU), particulièrement utile pour le prétraitement dans les applications de reconnaissance vocale. L'unité peut traiter jusqu'à 8 canaux (ou 4 canaux stéréo) à des fréquences d'échantillonnage d'entrée jusqu'à 192 kHz. Une unité FFT est disponible pour analyser le spectre du signal par transformée de Fourier rapide.

Notez la présence d'accélérateurs AES et SHA-256 pour les fonctions cryptographiques.
Sans oublier la collection des périphériques habituels : UART, I²C, SPI, I²S, temporisateur, RTC et PWM.
Sipeed a ajouté 16 Mo de mémoire flash sur la carte, ainsi que le circuit d'alimentation, le microphone et l'amplificateur de puissance de 3 W, et l'excellent module ESP32.
Que souhaiter de plus ? Vous trouverez de plus amples informations sur le fonctionnement de la carte, le schéma et d'autres détails en suivant les liens [1], [2] et [3].

Programmation en terrain connu

Ce n'est pas seulement par sa forme que notre carte est similaire à l'Arduino original : elle utilise l'environnement IDE Arduino, dans lequel le noyau Maixduino est intégré de la même manière que l'ESP8266 ou l'ESP32. Sous *Fichier -> Préférences*, il est nécessaire d'ajouter une nouvelle URL pour le gestionnaire de cartes : si vous faites un clic droit sur le bouton à droite de la zone de saisie de texte, une petite fenêtre s'ouvrira pour faciliter la saisie de l'URL (**fig. 4).** Si vous souhaitez programmer uniquement le Maixduino, il suffit d'ajouter la ligne «sipeed» ; la programmation de l'ESP32 embarqué passera par là aussi.

Pour installer les cœurs Maixduino, sélectionnez l'option de menu *Outils -> Gestionnaire de cartes*. Saisissez le terme de recherche «Maix» (**fig. 5**) et procédez à l'installation.

Ça y est, nous pouvons déjà commencer à programmer ! Plutôt que le programme classique *hello world*, nous allons passer directement à l'essai de la caméra et de l'afficheur. Connectez le Maixduino par USB et dans le menu *Outils*, réglez les paramètres suivants.

Board : `Sipeed Maixduino`
CPU Clock Frequency : `400 MHz`
Burn Tool Firmware : `open-ec`
Burn Baud Rate : `1.5 Mbps`
Tool Install Location : `Standard`
Port : `<port COM utilisé>`
Programmer : `k-flash`

Un programme de démonstration permettant de capturer une image et de l'afficher sur l'afficheur LCD est déjà disponible dans l'IDE. Appelez-le : *Fichier -> Exemples -> Sipeed_OV2640 -> selfie* et une fois qu'il est téléchargé sur la carte (vous devrez peut-être appuyer sur le bouton de réinitialisation à ce stade), vous pouvez l'exécuter. Le code est indiqué dans la **figure 6**.

Le programme commence par intégrer les fonctions des bibliothèques Sipeed pour le pilotage de la caméra et du LCD, connectées à la carte à l'aide d'un bus SPI. Le format d'image choisi est la résolution QVGA (320 x 240 pixels) avec une couleur RGB565. La routine de configuration initialise les deux appareils, et dans la boucle principale, les images capturées sont transférées directement au LCD : on ne pourrait pas faire plus simple.

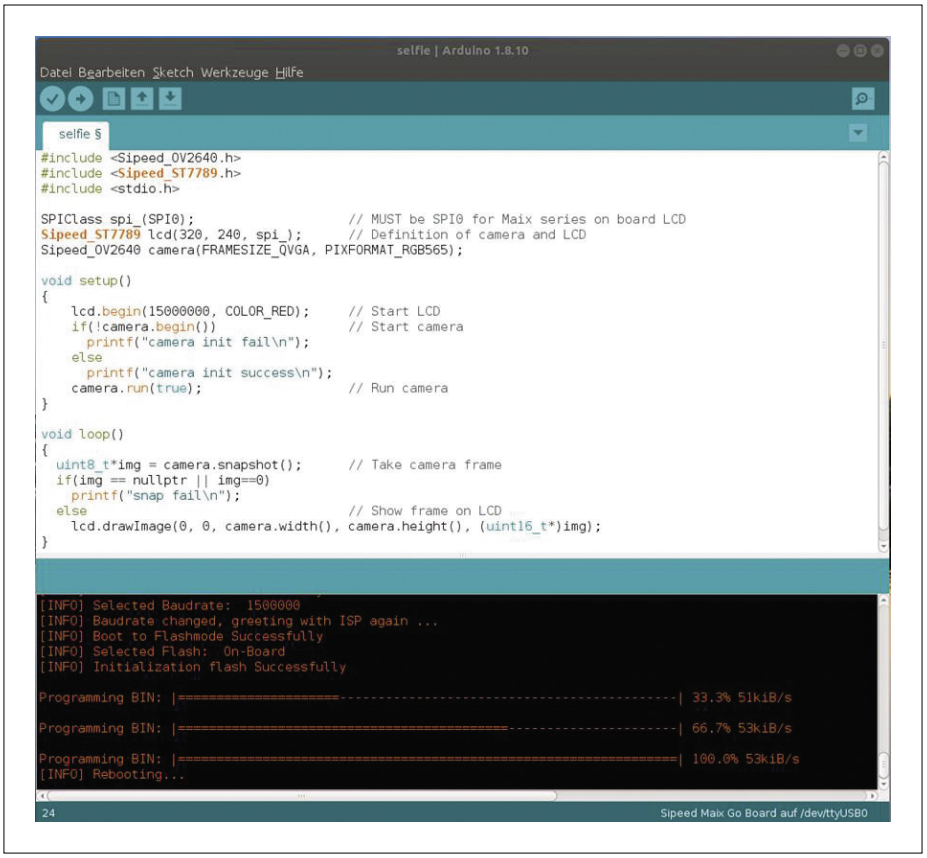

Figure 6. Arduino IDE avec le programme «selfie».

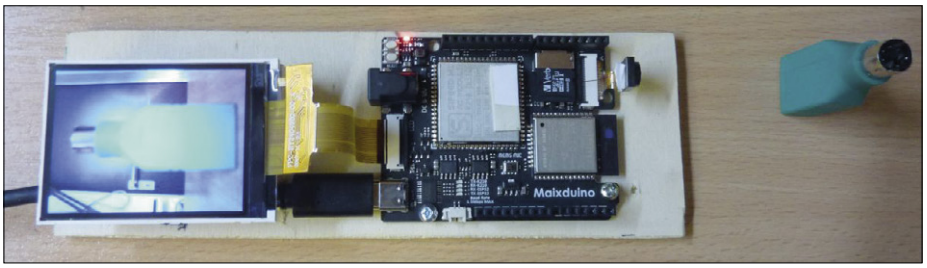

Figure 7. Image (à gauche) capturée d'une fiche d'adaptation (à droite).

L'image n'est pas particulièrement contrastée, mais elle est nette et le rafraîchissement fluide (**fig. 7**).

Maixduino est donc aussi facile à utiliser qu'un Arduino Uno, mais il offre beaucoup plus de possibilités. Nous pouvons commencer à construire des applications plus exigeantes.

Premier modèle d'IA

L'apprentissage approfondi de l'intelligence artificielle sera au menu du prochain article de cette série, mais nous pouvons commencer par une application simple, avec *MobileNet.* Rien à voir avec les téléphones portables, mais plutôt un classificateur d'images qui reconnaît et identifie 1000 types d'objets quotidiens. Il utilise un réseau neuronal, qui est une structure logicielle construite à partir de nœuds organisés en couches formées par un long processus au cours duquel on lui présente des milliers d'images.

Comme le fichier contenant le jeu d'images d'apprentissage occupe environ 200 Go et que le processus de formation lui-même est très long, il n'est pas pratique de le tenter sur le Maixduino. Cependant, il est possible d'y installer un modèle prêt à l'emploi pour le mettre immédiatement au travail de reconnaissance d'objets. C'est ce que je propose maintenant. Le logiciel nécessaire [4] se compose des éléments suivants :

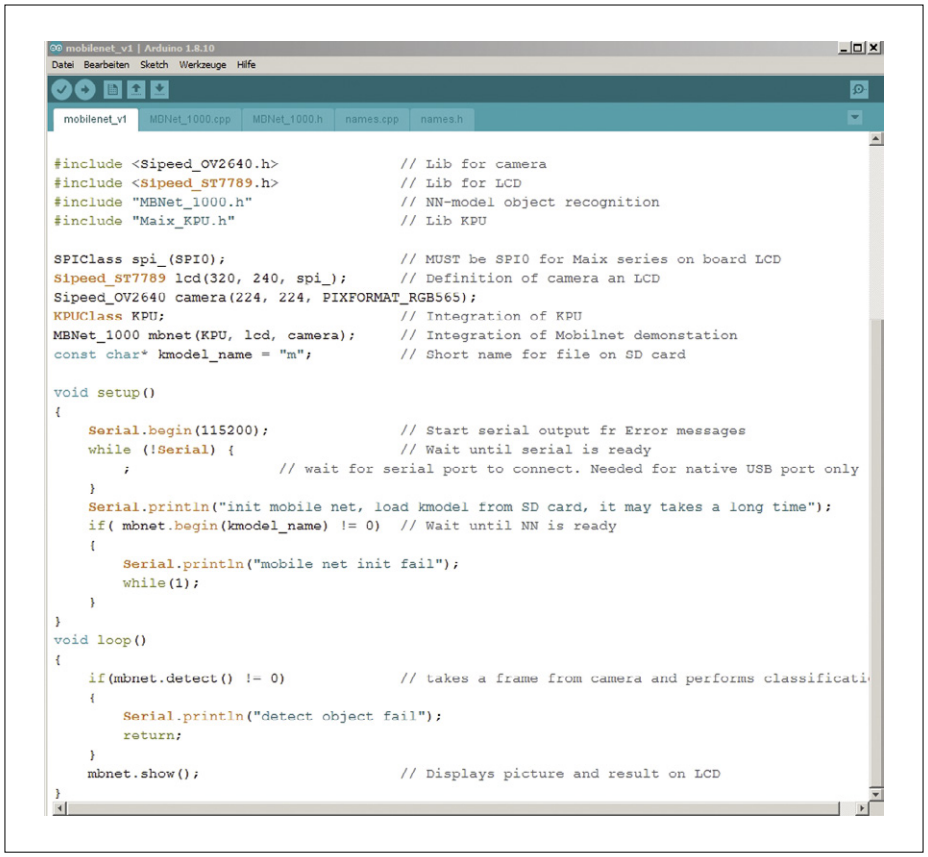

Figure 8. Programme de démonstration MobileNet.

- `mobilenet_v1.ino` : programme C++ principal pour la démonstration
- `MBNet_1000.h` : fichier d'en-tête pour les routines de démonstration
- `MBNet_1000.cpp` : routines C++ pour l'acquisition, la reconnaissance et l'affichage d'objets
- `names.h` : fichier d'en-tête avec description des objets
- `names.cpp` : routine C++ pour décrire les objets reconnus

Téléchargez ces fichiers et placez-les tous dans le même répertoire. Le modèle d'IA préformé mobilenet_0x300000.kfpkg peut être téléchargé [5] : compressé, il ne fait que quelques mégaoctets. Après l'avoir décompressé avec 7zip, vous devriez trouver un dossier mobilenet_0x300000 dans lequel il y a deux fichiers : le modèle est celui appelé 'm'. Copiez ce fichier sur une carte microSD dans le répertoire racine et insérez la carte sur le Maixduino.

Un coup d'œil au code du programme (**fig. 8**) montre qu'il est possible de programmer facilement des applications même très complexes, grâce aux puissantes bibliothèques : dans le domaine de l'IA, il existe de nombreuses bibliothèques très efficaces. La première partie du programme déclare la caméra, le LCD et la KPU ainsi que leurs paramètres. La résolution de la caméra est configurée au même format que les images d'entraînement, soit 224 par 224 pixels. Enfin, l'objet `mbnet` rassemble les ressources de la KPU, de la caméra et du LCD.

La routine de configuration initialise le code de démonstration, puis dans la boucle principale, nous avons la classification des objets dans les images acquises par la caméra et leur affichage sur le LCD. Imaginiez-vous qu'un programme de reconnaissance d'objets puisse être aussi simple ?

Comme objet de test, j'ai présenté au Maixduino une photo de chat (**fig. 9**), immédiatement identifié comme chat tigré ou égyptien (**fig. 10**). L'application est capable de traiter environ cinq images par seconde et peut donc produire des résultats utiles même si la caméra est en mouvement, surtout sous un bon éclairage et devant un arrière-plan pas trop chargé. Comme le montre la figure 10, la palette de couleurs est limitée (bleu-blanc) pour augmenter le taux de reconnaissance, selon une méthode courante. Au lieu d'utiliser les trois canaux de couleur, on n'en utilise qu'un seul pour réduire la quantité de données et donc la puissance nécessaire au traitement des pixels. La technique consistant à utiliser un réseau déjà formé, c'est-à-dire à effectuer la formation et la classification sur des plateformes différentes, est courante aussi dans ce type d'applica-

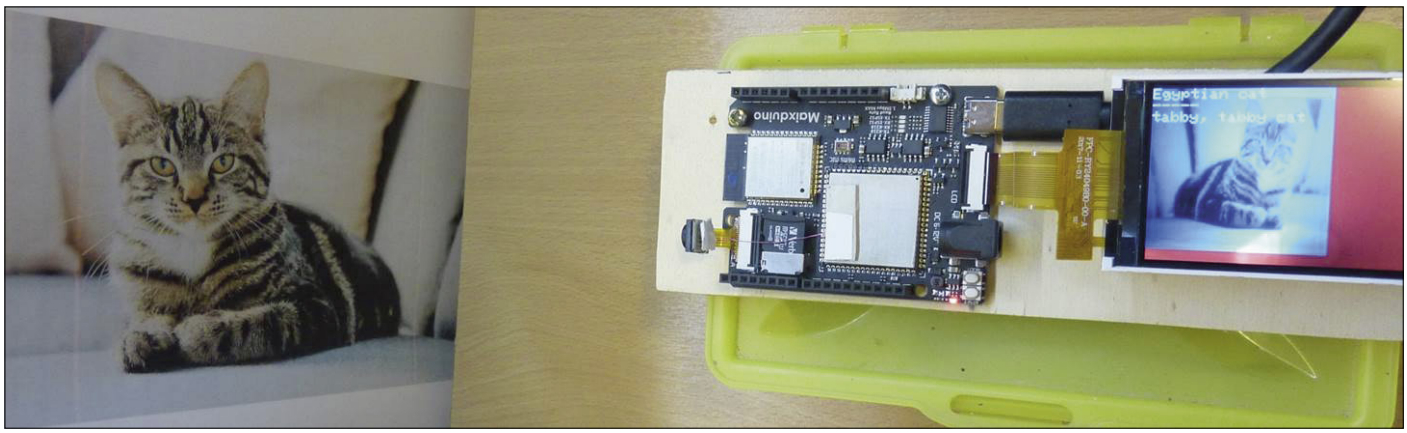

Figure 9. Acquisition de l'image d'un « objet ».

tion. De nombreux utilisateurs profitent de la grande puissance de traitement disponible sur AWS, Microsoft Azure ou Google Cloud pour former leurs réseaux de neurones, pour les faire fonctionner – ce qui nécessite beaucoup moins de ressources que l'apprentissage – sur des plateformes à plus petite échelle. L'industrie des semi-conducteurs répond déjà à cette demande : Intel a récemment annoncé ses processeurs pour réseaux de neurones Nervana NNP-T et NNP-I, dont la version «T» offre une puissance de traitement plus élevée pour la formation, et la version «I» est un dispositif moins puissant, destiné aux applications d'inférence et de classification.

Vous allez voir ce qu'on va voir...

L'application de démonstration examinée dans cet article effleurer la surface du sujet de l'intelligence artificielle en matière de reconnaissance d'objets, mais j'espère avoir suscité votre intérêt.
Dans le prochain épisode, nous nous pencherons sur la structure et le fonctionnement d'un réseau de neurones. Avec un nouvel environnement de développement, nous goûterons aux tentations de Linux et du langage Python, car c'est là que l'on trouve les bibliothèques et les infrastructures les plus prometteuses. Dans une telle infrastructure, la création d'un réseau neuronal est aussi simple que l'assemblage de briques Lego. Nous essayerons une application de reconnaissance des visages et vous développerez vos propres structures de réseau neuronal. Aux impatients, je recommande le livre *Make Your Own Neural Network de* Tariq Rashid. Si vous doutez encore de leur intérêt, sachez que la traduction de l'article que vous venez de lire a largement bénéficié d'outils de traduction automatique basés sur des réseaux neuronaux.

200023-03

Figure 10. L'objet est identifié et classé.

Liens

[1] Description de Maixduino : https://wiki.sipeed.com/en/maix/board/maixduino.html
[2] Forum Sipeed : https://bbs.sipeed.com/
[3] Schéma de la carte Maixduino : http://dl.sipeed.com/MAIX/HDK/Maixduino/Maixduino-4.30/
[4] Démonstration MobileNet : https://github.com/sipeed/Maixduino/tree/master/libraries/Maix_KPU/examples/mobilenet_v1
[5] Fichier de données modèle MobileNet : http://dl.sipeed.com/MAIX/MaixPy/model/mobilenet_0x300000.kfpkg