afficheurs LED avec le MAX7219

une puce excellente

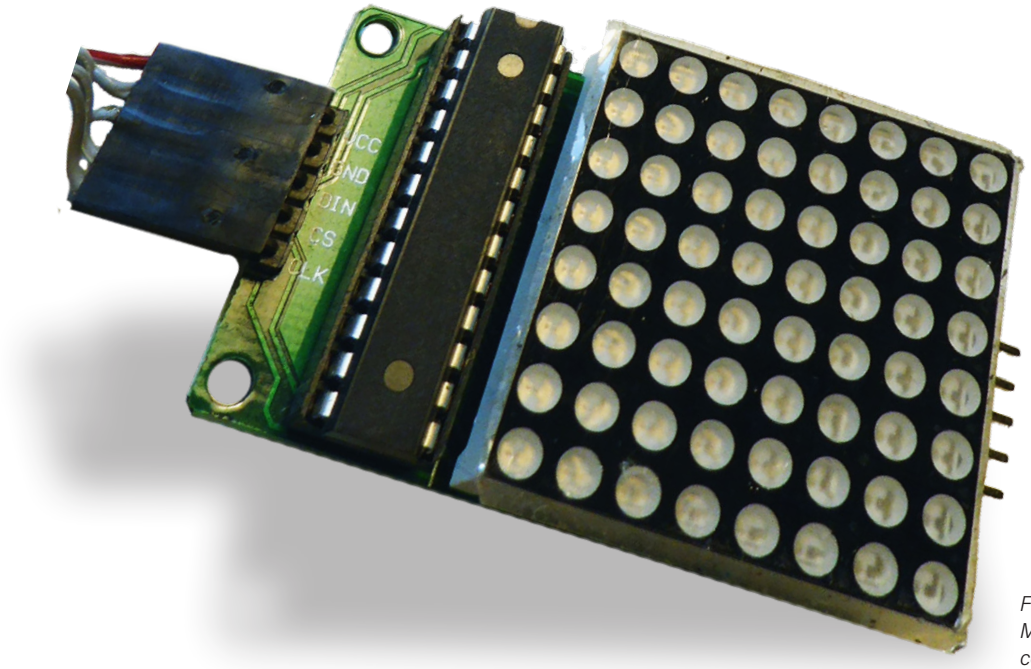


Figure 1. Un module MAX7219 peu coûteux.

Jonathan Hare (Royaume-Uni)

Le MAX7219 est un driver d'affichage utile et polyvalent, capable de contrôler jusqu'à 64 LED indépendamment. Bien qu'il soit largement reconnu, la majorité des tutoriels disponibles se contentent d'exploiter des bibliothèques prêtes à l'emploi. À l'opposé, je vous offre une approche plus fondamentale qui consiste à manipuler directement les registres du composant. Cette méthode enrichit votre compréhension et vous fournit des connaissances essentielles pour réussir vos projets avec cette puce utile, économique et amusante.

Le MAX7219 de Maxim Integrated [1], dans son boîtier de 24 broches, peut contrôler indépendamment jusqu'à huit rangées de huit LED, ou alternativement jusqu'à huit afficheurs à sept segments. Ceci le rend idéal pour des projets nécessitant un affichage visuel à base de LED. J'ai récemment utilisé huit de ces puces pour piloter un cube à 512 LED dans un jeu interactif destiné à enseigner les mathématiques des matrices et des tenseurs [2]. Bien que l'internet regorge de projets intéressants utilisant le MAX7219 que vous pouvez simplement reproduire, ils s'avèrent limités pour ceux désirant innover. Je vous propose donc une approche différente et de travailler directement avec les registres.

Modules MAX7219

Vous pouvez acheter la puce nue en boîtier DIP ou SMD. Toutefois, il est généralement plus simple et économique d'opter pour un module prêt à l'emploi disponible sur eBay ou ou d'autres sites similaires. Ces modules intègrent une matrice de 8 x 8 LED sur le circuit imprimé (**figure 1**). Une carte de développement typique comprend cinq connexions : VCC = 5 V, GND = 0 V, DIN = SPI entrée de données, CS = chip select et CLK = SPI entrée d'horloge. Notez la présence

de deux connecteurs à cinq broches ; le connecteur le plus proche de la puce, à gauche, est utilisé pour la communication avec la carte, tandis que celui à droite permet de connecter plusieurs modules en chaîne créant ainsi un affichage de plus grande dimension.

Schéma

Le schéma du module est illustré dans la **figure 2**. Il est très simple : la puce est alimentée via sa broche 19 (VCC, 5 V) et ses broches 4 et 9 (masse). Par ailleurs, les broches 2, 3, 5, 6, 7, 8, 10 et 11 sont connectées aux cathodes des LED. Elles sont désignées par Dig 0 à Dig 7 sur la fiche technique, car il est également possible de les utiliser pour contrôler des afficheurs à sept segments. Les broches 14, 15, 16, 17, 20, 21, 22 et 23 sont connectées aux anodes des LED. Elles sont désignées par SEG A à SEG DP sur la fiche technique pour la même raison. La broche 18 est utilisée pour ajuster la luminosité maximale de toutes les LED, en y connectant une résistance (habituellement de 10 kΩ) à la source de tension positive. Les broches 1, 12 et 13 servent d'entrées pour l'interface SPI. Si vous réalisez vous-même un circuit, placez les condensateurs C1 et C2 aussi proche que possible des broches d'alimentation de la puce.

À propos des registres

La plupart des pilotes d'affichage comme le MAX7219 n'affichent rien sur leur sortie à la mise sous tension. Une communication est nécessaire pour configurer ces puces et leur transmettre des données. L'utilisation d'un microcontrôleur ou d'un ordinateur via un port série est courante pour cette tâche. Ces puces sont pilotées par des registres plutôt que par des commandes ; elles sont configurées et contrôlées par le chargement de données dans leurs registres. Il existe 13 registres accessibles via un bus série. Cela peut sembler complexe, mais devient simple avec l'utilisation de Python ou d'un autre langage de programmation de haut niveau. Une fois ces registres initialisés, il n'est pas nécessaire de les reconfigurer tous à chaque modification de l'état des LED ou de l'affichage. En pratique, vous pouvez créer une routine de configuration à exécuter au début de votre programme, puis vous envoyez des données uniquement au(x) registre(s) à modifier.

Communications en série avec des puces et des capteurs

Bien que certains circuits intégrés puissent recevoir des données sur un bus parallèle (en utilisant plusieurs lignes de données à la fois,

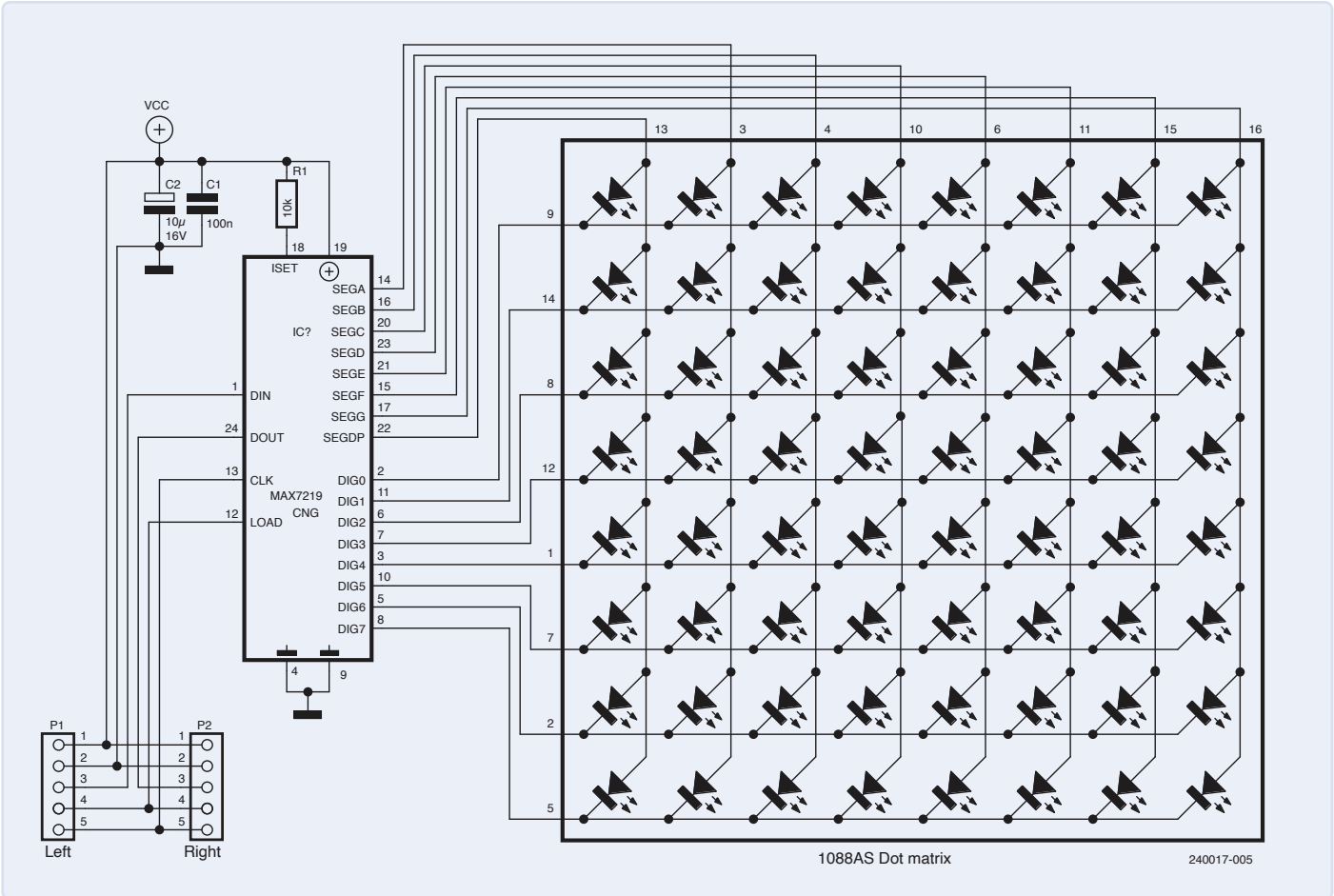


Figure 2. Schéma du module.

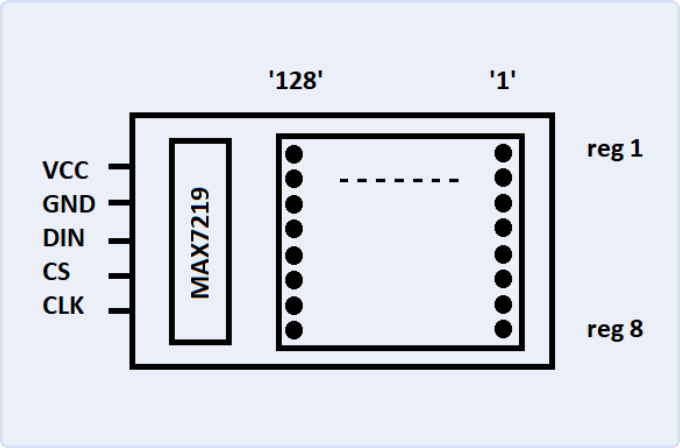


Figure 3. Orientation et adressage des LED.

par exemple huit), la communication série est plus répandue. Dans le domaine des systèmes embarqués, les trois principales interfaces série sont UART [3], I²C [4] et SPI [5]. Le MAX7219 utilise l'interface SPI, qui est relativement simple une fois que l'on a pris le coup de main. La consultation de la fiche technique de la puce s'avère cruciale pour adapter correctement votre application. À la différence de l'UART, le SPI ne requiert pas de configuration de vitesse de transmission (baud rate), bien qu'il puisse être nécessaire de ralentir le bus par rapport à ses paramètres par défaut, par exemple si vous avez de longues liaisons entre le PIC principal et la puce MAX7219.

Outre la connexion de masse commune (GND, broche 4), SPI utilise deux fils pour envoyer des données vers et depuis la puce : une ligne de données (DATA IN/DIN, broche 1) et une ligne d'horloge (CLK, broche 13). Une troisième ligne appelée Chip Select (CS/LOAD, broche 12) est utilisée pour charger les données reçues dans l'appareil. Les données SPI nécessaires pour piloter la puce MAX7219 sont assez simples. La communication avec le MAX7219 via SPI implique l'envoi de 16 bits de données et l'activation temporaire de la broche CS pour charger ces données dans la puce. Lorsque vous utilisez le MAX7219, il suffit d'envoyer des données sans nécessité de recevoir des informations en retour.

Logique 5 V ou 3.3 V

En général, il est crucial de vérifier la compatibilité des tensions de fonctionnement des appareils en communication. Les niveaux de 5 V et 3,3 V sont les plus répandus. La fiche technique du MAX7219 indique une tension de fonctionnement nominale de 5 V (4 V à 5,5 V). Pour les microcontrôleurs fonctionnant à une tension différente, comme 3,3 V, un circuit de décalage de tension pourrait s'avérer nécessaire. La fiche technique indique que la tension minimale pour un niveau logique haut (V_{IH}) est de 3,5 V. Ainsi, un niveau « haut » de 3,3 V fournie par un microcontrôleur alimenté en 3,3 V n'est pas assez élevé. En pratique, j'ai constaté que les modules MAX7219 fonctionnaient correctement avec des microcontrôleurs 3,3 V, tels que les PIC de Microchip ou le Raspberry Pi Pico, sans nécessiter de décalage de niveau, même si c'est en dehors des spécifications de la fiche technique.

Registres du MAX7219

Le MAX7219 est contrôlé par treize registres. Chaque registre a 8 bits de données et une adresse de 8 bits.

- le registre 1 définit l'état (activé ou désactivé) de la première rangée de DEL. Son adresse est 00000001 en binaire.
- les registres suivants 2 à 8, dont les adresses sont respectivement 00000010 et 00001000, définissent les autres rangées de LEDs de la 2ème à la 8ème.
- le registre 9 (adresse : 00001001) définit le mode de décodage des données destinées aux afficheurs à 7 segments.
- registre 10 (adresse : 00001010) règle la luminosité
- registre 11 (adresse : 00001011) règle la limite de balayage
- le registre 12 (adresse : 00001100) définit le mode d'arrêt/de démarrage
- le registre 13 (notez que son adresse est 00001111 et non 00001101) est utilisé pour tester l'affichage.

Il est important de noter que le registre à l'adresse 00000000 est inactif (no-op) lorsqu'un seul module MAX7219 est utilisé. Il est possible d'utiliser ce registre lors de la mise en cascade de plusieurs modules MAX7219 (aspect que j'aborderai plus en détail ultérieurement). Pour plus de détails sur tous ces registres, referez-vous à la fiche technique et aux encadrés **«Plus de détails sur les registres 9 à 13»** et **«Mise en cascade de plusieurs modules MAX7219»**.

Mise en cascade de plusieurs modules MAX7219

Le MAX7219 est doté d'une fonction intelligente intégrée qui peut être utilisée pour piloter plusieurs puces afin de créer de plus grands réseaux de LED. Grâce à cette fonction, vous pouvez relier des cartes entre elles pour créer un grand affichage à LED, utilisé typiquement dans les écrans déroulants visibles sur les bus et les panneaux d'information des gares. Pour ce faire, il faut connecter la broche Data Out du premier module à la broche Data In du module suivant, et ainsi de suite. Chaque puce transmet les données reçues à la puce suivante après seize cycles d'horloge. Par exemple, si vous avez deux modules MAX7219 connectés comme décrit, les données sont envoyées sous forme de deux mots de 16 bits. Les données sont transmises d'une puce à l'autre, un bit à la fois. Ensuite, vous envoyez une impulsion à la broche CS pour verrouiller les données. La première puce (la plus proche du microcontrôleur) stocke donc les 16 derniers bits qui ont été envoyés, tandis que la seconde puce stocke les 16 premiers bits.

Pour adresser uniquement la deuxième puce sans affecter la première, on utilise le registre no-op (adresse 00000000). Il suffit de lui envoyer le mot de 16 bits souhaité, suivi d'une commande no-op (seize zéros). Ainsi, la deuxième puce reçoit les données et la première puce ne change rien. Ceci peut être étendu à n'importe quel nombre de MAX7219, le seul facteur limitant étant qu'il peut devenir peu pratique de manipuler des nombres avec un grand nombre de bits. Bien que cette méthode rende le câblage très simple, elle rend le codage un peu plus complexe. Pour mon cube LED éducatif 8 × 8 × 8, comme le Raspberry Pi Pico a beaucoup de ports de sortie, j'ai choisi de câbler toutes les lignes de données SPI ensemble, de sorte que toutes les puces reçoivent les données en même temps, mais ne sélectionnent que les broches de sélection de puce (LOAD) sur les tableaux 8 × 8 que je souhaite contrôler à chaque instant.

Tableau 1. Broches et connexions.

Numéro de broche Pi Pico	Numéro du Pi Pico Portpin	Connexion à la carte MAX7219
pin 4	GP2	CLK
pin 5	GP3	DIN (DATA)
pin 21	GP16	CS (LOAD)
pin 40	VBUS	VCC (+5 V)
pin 38	GND	GND (0 V)

Orientation du module

Lorsque vous utilisez le module décrit dans cet article, assurez-vous de le maintenir dans l'orientation indiquée, avec la puce à gauche (**figure 3**). Le registre 1 correspond à la rangée supérieure de LED. Pour les huit bits de données envoyés à ce registre, la LED correspondant au bit de poids faible se trouve à l'extrémité droite, la plus éloignée du MAX7219.

Utilisation des registres

La transmission des données au MAX7219 se fait par paquets de 16 bits via le bus SPI. L'adresse du registre doit être envoyée en premier, suivie des données que vous souhaitez stocker dans le registre. Le bit de poids fort (MSB) doit être transmis en premier. Ceci est facile à implémenter dans le code. Par exemple, si vous souhaitez allumer une LED sur deux dans la troisième rangée de LED, vous devez envoyer les données suivantes sur le bus SPI : 00000011 10101010. Le 00000011 est l'adresse du 3e registre, tandis que le 10101010 allumera une LED sur deux.

Le bit de poids fort en premier et la configuration SPI

L'obligation d'envoyer le bit de poids fort en premier peut prêter à confusion. Le périphérique SPI du microcontrôleur que vous utilisez possède d'autres paramètres qui doivent parfois être définis, par exemple si les données sont capturées sur le front descendant ou ascendant des impulsions. Dans la plupart des cas, les langages de haut niveau que vous utilisez peuvent gérer cela pour vous. Il est recommandé de consulter la documentation de votre microcontrôleur pour utiliser les réglages par défaut.

Exemple en PIC BASIC

Je trouve le compilateur PIC BASIC de Crownhill [6] très agréable à utiliser. Il est parfaitement compatible avec les microcontrôleurs PIC16F877 qui, bien qu'anciens, restent extrêmement utiles pour le prototypage rapide. Voici un exemple de ligne de code en PIC BASIC pour envoyer le code SPI du PIC au MAX7219, en utilisant le Port B.2, MSB en premier du PIC au MAX7219 :

```
SHOut PORTB.2, PORTB.0, MsbFirst, [%0000010111111111]
```

Cette commande envoie un flux de 16 bits, avec l'adresse en premier (00000101, qui correspond à 5) et ensuite les données (tous les bits à 1). Cela permet d'allumer toutes les LED de la ligne 5.

Exemple en MicroPython avec le Raspberry Pi Pico

Bien que certaines versions de MicroPython incluent des modules pré-écrits pour le MAX7219, permettant de dessiner des formes ou des caractères, je préfère aborder la programmation des registres directement pour un contrôle plus précis. Pour ce faire, connectez les broches 4 (CLK), 5 (DATA) et 21 (CS) aux broches correspondantes du module MAX7219, comme décrit dans le **tableau 1**. MicroPython

Listage 1. A setup Module.

```
from machine import Pin, SPI
import utime
import urandom
spi_sck = Pin(2)
spi_tx = Pin(3)
spi_rx = Pin(0)
spi = SPI(0, sck=spi_sck, mosi=spi_tx, bits=8,
miso=spi_rx, baudrate=100000, firstbit=SPI.MSB)
cs = Pin(16, Pin.OUT)
cs.value(0)

# chip select 'pulse'
def cs_ping_set():
    cs.value(1)
    utime.sleep(.001)
    cs.value(0)

def setup_max7219():
    for i in range (1,9):
        # Note: zero = nop
        # clear led rows 1 to 8
        # in registers 1 to 8
        buff = [0,0]
        buff[0] = i
        buff[1] = 0b00000000
        spi.write(bytearray(buff))
        cs_ping_set()

    # set shutdown to start
    # register 9
    buff = [0,0]
    buff[0] = 0b00001100
    buff[1] = 0b00000001
    spi.write(bytearray(buff))
    cs_ping_set()
    # adjust brightness
    # register 10
    buff = [0,0]
    buff[0] = 0b00001010
    buff[1] = 0b00001111
    spi.write(bytearray(buff))
    cs_ping_set()
    # scan limit register -set all rows and col.s
    # register 11
    buff = [0,0]
    buff[0] = 0b00001011
    buff[1] = 0b00000111
    spi.write(bytearray(buff))
    cs_ping_set()
    # set decode mode
    # register 12
    buff = [0,0]
    buff[0] = 0b00001001
    buff[1] = 0b00000000
    spi.write(bytearray(buff))
    cs_ping_set()
    # set test mode off
    # register 13 but Note address 15
    buff = [0,0]
    buff[0] = 0b00001111
    buff[1] = 0b00000000
    spi.write(bytearray(buff))
    cs_ping_set()
```

nécessite que vous importiez du code pour gérer la fonction SPI. Cela se fait en ajoutant `from machine import Pin, SPI` au début de votre programme. Le code Python que vous devez utiliser pour envoyer les mêmes données SPI de l'exemple précédent au MAX7219 est :

```
from machine import Pin, SPI
buff = [0,0]
buff[0] = 0b00000101
buff[1] = 0b11111111
spi.write(bytearray(buff))
```

Autres exemples

Les lignes de code ci-dessus sont un extrait qui illustre les parties les plus importantes. Vous pouvez consulter les programmes complets sur mon site web [7]. Dans les deux exemples, vous devez immédiatement fournir une impulsion positive sur la broche Chip Select (CS ou LOAD) pour charger les données dans le registre. Pour ce faire, il suffit d'écrire :

```
cs.value(1)
utime.sleep(0.001)
cs.value(0)
```

Écrire des programmes

Avant de commencer à adresser les LED de votre afficheur, vous devez créer un sous-programme d'installation pour la configuration. Exécutez ce sous-programme au début de votre programme. Par la suite, vous pouvez simplement adresser les LED de chaque ligne, en réécrivant le registre correspondant, sans ordre particulier. Le **listage 1** présente la fonction `setup_max7219()` que j'ai développée, disponible également sur mon site web. Cette fonction commence par une boucle qui envoie zéro aux registres 1 à 8, ce qui éteint toutes les LED. Ensuite, le registre 9 est défini (mode de décodage) à 0 pour une utilisation avec des LED individuelles (et non des afficheurs à sept segments), puis le registre 10 (luminosité) est défini à 15 (1111), le registre 11 (limite de balayage) à 7 (pour utiliser toutes les lignes de LED), le registre 12 (arrêt) à 0, et enfin le registre 13 (mode test) à 0.

Figure 4. Cube à LED.

Listage 2. Lights Flashing Across the Display.

```
# display set-up
setup_max7219()
while True:
# sweep rows
    for j in range(1,9):
        buff = [0,0]
        buff[0] = j    # register no.
        buff[1] = 255  # value (all ON)
        spi.write(bytearray(buff))
        cs_ping_set()
        utime.sleep(.2)

        setup_max7219()  # clear screen
# sweep columns
    for k in range(0,8):
        for l in range(1,9):
            buff = [0,0]
            buff[0] = l    # register no.
            buff[1] = 2**k  # value 2^k
            spi.write(bytearray(buff))
            cs_ping_set()
        utime.sleep(.2)

    setup_max7219()  # clear screen
```

Effacer l'afficheur

Une autre fonction essentielle est de pouvoir effacer tout l'afficheur en une seule fois sans modifier les autres réglages. Cela peut être réalisé en envoyant des «0» aux huit premiers registres, éteignant ainsi toutes les LED simultanément. Grâce à la rapidité du SPI, cette opération ne prend que quelques millisecondes, rendant l'effacement de l'afficheur instantané et imperceptible pour l'œil humain.

Exemples de Light Chaser

Pour créer un effet visuel où une rangée de LED allumées se déplace à travers l'afficheur, commencez par effacer l'afficheur. Envoyez ensuite 11111111 (toutes les LED allumées) au registre 1. Après une courte pause, effacez à nouveau l'afficheur et envoyez les mêmes données au registre 2, et ainsi de suite. Une boucle permet de répéter cette opération pour les huit registres, créant ainsi un effet de balayage des LED sur l'afficheur. Si vous ne réinitialisez pas l'afficheur après chaque étape, vous obtiendrez un effet de rideau de lumière croissant. Le délai déterminera la vitesse à laquelle les lumières clignotent. Ceci est illustré dans le **Listage 2**. Ici, pour simplifier le code, j'ai utilisé à nouveau la fonction `setup_max7219()` pour réinitialiser l'affichage. Cela fonctionne bien ;

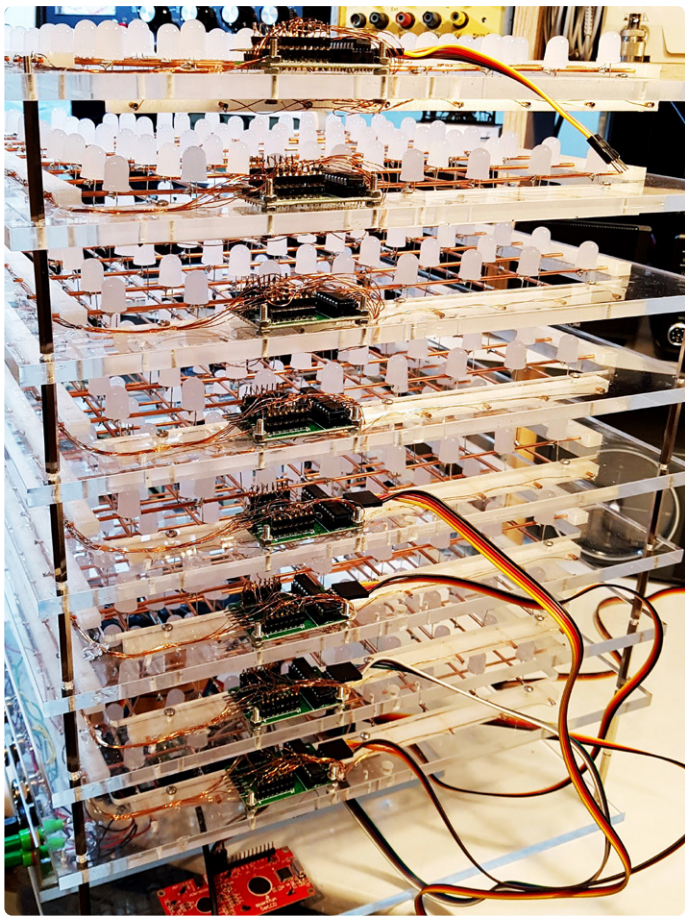


Figure 5. Vue latérale montrant huit modules MAX7219.

Plus de détails sur les registres 9 à 13

- Le registre 9 définit le mode de décodage. Vous pouvez configurer la puce MAX7219 pour contrôler soit des rangées de LED, soit des afficheurs à 7 segments. Le réglage par défaut est pour une seule rangée deLED (voir registre 11 ci-dessous). En mode décodage, vous pouvez contrôler directement les afficheurs à 7 segments, ce qui facilite la programmation : par exemple, envoyer « 00000000 » comme donnée, qui est un zéro en décimal, activera 6 des sorties pour contrôler les segments A à F de l'afficheur à 7 segments afin de créer un zéro ; même chose pour les autres chiffres décimaux 1 à 9 et quelques autres symboles (voir la fiche technique).
- Le registre 10 définit la luminosité des LED. La puce utilise la modulation de largeur d'impulsion (PWM) pour contrôler la luminosité des LED. 32 niveaux sont disponibles, le nibble inférieur contrôlant la luminosité : 0000 est le moins lumineux (1:32 PWM), et 1111 est presque entièrement allumé (31:32 PWM). Vous pouvez utiliser le réglage le moins lumineux comme mode nuit, par exemple, mais n'oubliez pas de le remettre à pleine luminosité plus tard.
- Le registre 11 définit le nombre de rangées de LED à utiliser. Lorsqu'elle est mise sous tension, la puce est réglée pour faire fonctionner une seule ligne de 8 LED. Si vous voulez piloter une seule rangée de 8 LED (ou un seul afficheur à 7 segments), réglez ce registre à 00000000. Pour utiliser les huit rangées (64 LED), réglez ce registre sur 00000111, et ainsi de suite.
- Les registres 12 et 13 peuvent être utilisés pour éteindre l'écran (par exemple, pour économiser de l'énergie) et pour lancer un mode de test, utile pour vérifier que toutes les LED fonctionnent correctement. Notez que la plupart des registres ont une adresse en binaire qui correspond à leur numéro de registre en décimal, mais pas celui-ci : son adresse est 00001111, c'est-à-dire 15, et non 13.

bien que les registres 9 à 13 soient écrits inutilement à chaque fois, cela n'est pas visible à l'œil humain. Une alternative serait d'écrire une fonction `clear_screen()` dédiée.

Si vous voulez faire clignoter les LED en colonnes plutôt qu'en lignes (c'est-à-dire pivoter l'effet à 90 degrés), vous devez envoyer 00000001 à tous les registres en utilisant une boucle. Cela permet d'allumer la première LED de chaque colonne, créant ainsi une colonne de LED. Utilisez la boucle suivante pour envoyer 000000010 à tous les registres, puis 00000100, etc. Vous pourriez utiliser le décalage par bit pour cela, mais dans l'exemple montré, j'ai juste utilisé des puissances croissantes de deux pour le même but, comme 2^1 = 2 = 0b00000010, 2^2 = 4 = 0b00000100, etc. Il n'est pas nécessaire de réinitialiser l'afficheur à chaque fois, puisque les registres sont modifiés progressivement. Là encore, le délai détermine la vitesse à laquelle les lumières clignotent. En jouant avec les valeurs des bits que vous envoyez aux huit registres 1 à 8, vous pouvez créer toutes sortes de motifs captivants, de flashs, de voiles lumineux, etc. Vous pouvez également utiliser la fonction aléatoire de Python pour créer de jolis effets visuels. Jetez un coup d'œil aux autres exemples qui incluent des simulations de vagues, d'éclairs, de flammes, de marches aléatoires et même quelques projets basés sur les mathématiques sur mon site web [7].

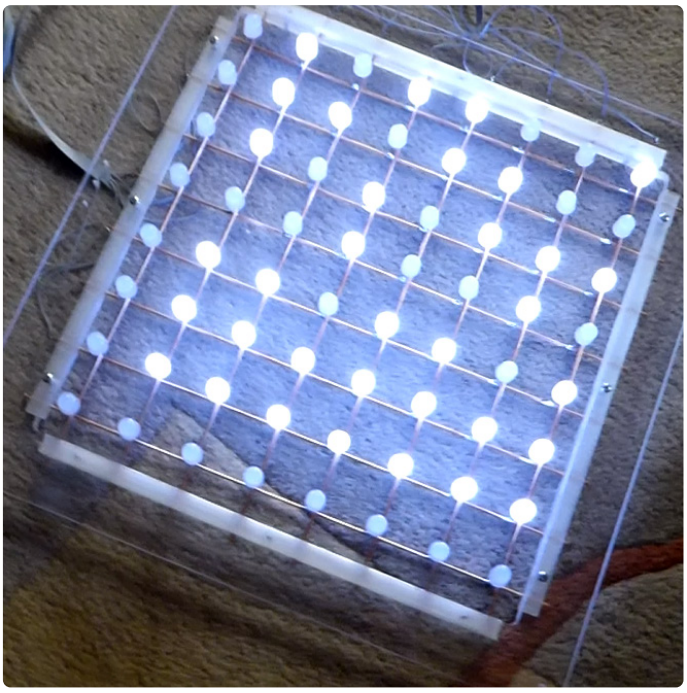


Figure 6. Une des huit couches.

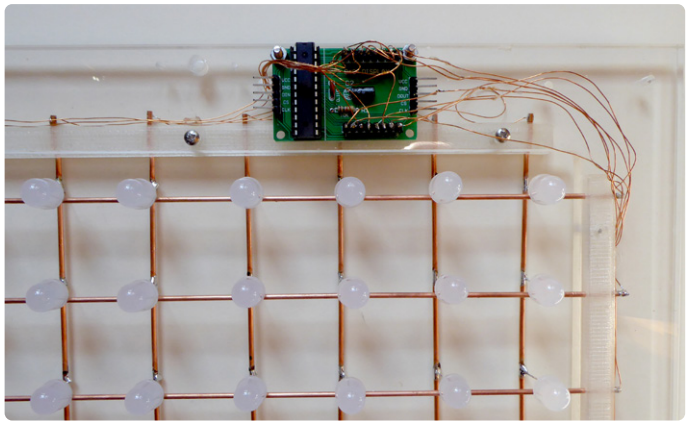


Figure 7. Connexions aux lignes et aux colonnes.

Enseigner les scalaires, les vecteurs, les matrices et les tenseurs de manière intelligente

J'ai récemment collaboré avec l'université de Sussex (Royaume-Uni) pour développer un jeu éducatif destiné à enseigner les vecteurs, les matrices et les tenseurs aux élèves et aux étudiants. Le jeu, qui peut également être utilisé en « mode démo » pour démontrer d'autres principes mathématiques. Le dispositif (**figure 4**) consiste en un cube composé de huit modules MAX7219 (**figure 5**), chacun pilotant 64 LED blanches de 10 mm, soit 8 × 8 × 8 = 512 LED au total ! Les données représentées par les LED, allumées ou éteintes, forment des tableaux binaires. Conçu pour captiver un public de 10 à 16 ans, le jeu a également séduit un large éventail d'âges lors d'événements familiaux STEM. La structure de chaque couche (**figure 6**) est composée de feuilles d'acrylique transparent et de tubes de cuivre, le tout piloté par un Raspberry Pi Pico programmé en MicroPython.

Bien que l'utilisation de LED RGB programmables et adressables en série aurait également été une option, je n'ai pas trouvé de versions traversants de 10 mm nécessaires pour le projet. Le jeu comprend quatre rangées de 8 interrupteurs ; trois d'entre eux sont utilisés par les participants pour entrer les données x, y et z pour jouer. La quatrième est utilisée pour la sélection du menu du jeu. Plus de détails sur mon site web et sur ma chaîne YouTube [8]. Je tiens à remercier l'équipe du professeur Hazel Cox à l'université du Sussex (Royaume-Uni), notamment le professeur Michael Melgaard et le docteur Yevgen Petrov, Brice Kammegne et Tom Baird-Taylor. Merci également au festival STEM de Lewes. Ce projet a été soutenu par un prix d'engagement public APEX de la Royal Society.

Fabriquer un afficheur plus grand

Si vous envisagez de fabriquer votre propre afficheur à LED 8 × 8, comme je l'ai fait pour le cube à LED éducatif 8 × 8, il est toujours intéressant d'utiliser un module MAX7219 prêt à l'emploi. Il suffit de retirer l'afficheur LED embarqué et de souder les connexions pour votre afficheur (**figure 7**). Vous aurez besoin de faire une matrice de LED avec toutes les anodes câblées à huit rails communs (Note : vous n'avez pas besoin de résistances de limitation de courant) et les cathodes câblées à huit rails communs. J'ai utilisé des tubes de cuivre de 2 mm (plutôt que des tiges ou des fils qui sont beaucoup plus faciles à souder), qui ont été soutenus sur un morceau d'acrylique transparent à l'aide de supports imprimés en 3D. Les huit rails d'anode et les huit rails de cathode peuvent être connectés aux broches correspondantes du module avec un fil plus fin. Une fois les 64 LED connectées, connectez les lignes SPI et les connexions d'alimentation (**figure 8**) et le tour est joué. J'espère que cet article vous incitera à prendre un module MAX7219, à commencer à expérimenter et à vous amuser !

240017-04

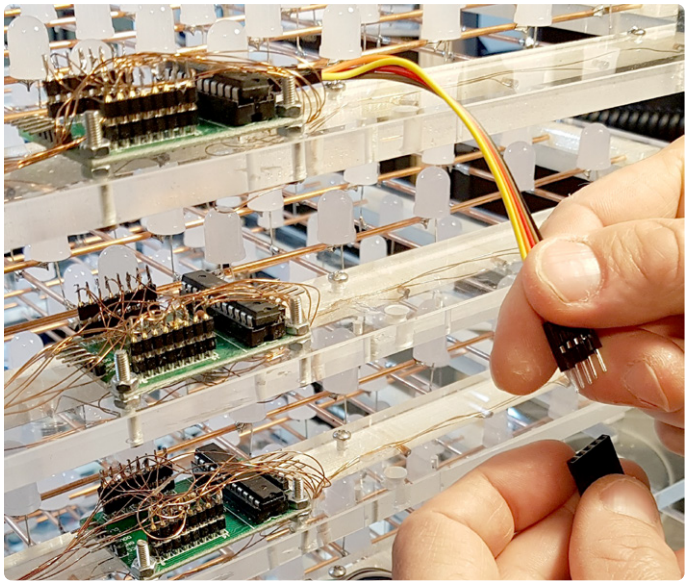


Figure 8. Branchement des modules entre eux.

Questions ou commentaires ?

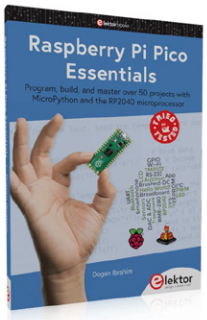
Envoyez un courriel à l'auteur (jphcreativescience@gmail.com), ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Jonathan Hare est un scientifique indépendant et un communicateur scientifique. Lors de son doctorat avec Sir Harry Kroto , il a mis au point une méthode de synthèse de molécules de buckminsterfullerène (C60), souvent appelé « football molecule » en raison de sa forme caractéristique. Actuellement, il est chercheur invité au département de chimie de l'université du Sussex, où il est responsable de la sensibilisation et de l'engagement du public. Jonathan a également contribué à plusieurs émissions de la BBC, notamment Rough Science, Coast et Horizon. Passionné de bricolage, il s'adonne aussi au jonglage, à la randonnée, à la radio amateur, à la peinture et, bien sûr, à l'électronique. Pour plus d'informations, vous pouvez consulter son site web [9].

Produits

- **Dogan Ibrahim, *Raspberry Pi Pico Essentials* (Elektor, 2021)**
  www.elektor.fr/19673
- **MAX7219 Dot Matrix Module (Set of 8)**
  www.elektor.fr/18422



LIENS

[1] Fiche technique du MAX7219 : https://tinyurl.com/5n6nbyv6
[2] Tensor Game demo : https://www.youtube.com/watch?v=VaoEBamceXg
[3] UART (Wikipedia) : https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter
[4] I²C (Wikipedia) : https://en.wikipedia.org/wiki/I%C2%B2C
[5] SPI (Wikipedia) : https://en.wikipedia.org/wiki/Serial_Peripheral_Interface
[6] Compilateur PIC BASIC : https://www.crownhill.co.uk/pic_basic
[7] Exemples en Python sur le site de l'auteur : http://www.creative-science.org.uk/max7219.html
[8] Chaîne YouTube de l'auteur : https://www.youtube.com/@jonathanhare6644
[9] Site web de l'auteur : https://www.zoomscience.co.uk