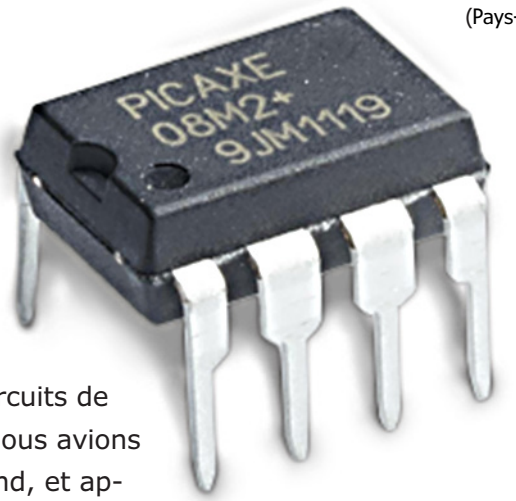
PIC atout AXE (3)

entrées analogiques, MLI, servos et vieilles rengaines

Wouter Spruit
(Pays-Bas)



Dans le premier article de cette série, nous avions vu comment programmer un PICAXE et comment assembler de simples circuits de programmation (Projet Elektor.POST nº 8). Nous avions commandé divers interrupteurs dans le second, et appris à calculer les valeurs des composants qui formaient nos circuits d'interface (Projet Elektor.POST nº 16). Nous allons voir ici comment lire et traiter des valeurs d'entrées analogiques à l'aide de commandes PICAXE, et nous relierons différents types de circuits à une sortie capable d'utiliser la modulation de largeur d'impulsion (MLI). Nous apprendrons à positionner des pièces mécaniques à l'aide de servomoteurs commandés par un PICAXE, et nous créerons même des sonneries monophoniques !

Petit rappel sur les montages

Comme précédemment, tous les montages de cet article utilisent un microcontrôleur PICAXE 08M2 (dont le brochage est reproduit sur la **figure 1**) alimenté par les 5 V d'une alimentation ATX. Les circuits sont assemblés sur une plaque d'essai sans soudure. Les puces PICAXE sont programmées via un câble USB-série selon la méthode décrite dans les articles précédents [2] et avec le logiciel LinAXEpad, version 1.5.0 pour Linux (Arch). Pour éviter toute confusion, tous les numéros de broche des schémas se réfèrent à la numérotation physique sur la puce ; p. ex. la broche 5 est une sortie PWM, et la broche 6 l'entrée analogique 1. Dans les listages, les numéros de broche font référence aux noms internes des broches (fig. 1). Pour des exemples de programmation d'une puce PICAXE, reportez-vous aux articles précédents [2] ou au manuel PICAXE [3]. Vous pouvez vous procurer les puces PICAXE et différents périphériques chez *Revolution Education* [4].

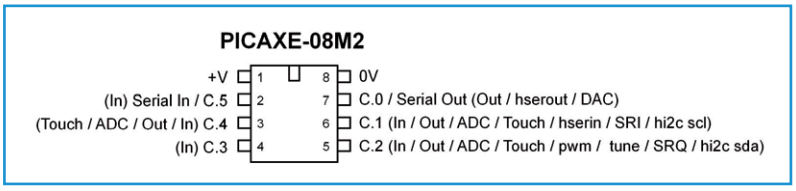


Figure 1.
Brochage du PICAXE 08M2.

Entrée analogique

Nos exemples précédents recouraient tous à des interrupteurs binaires (tout ou rien) pour commander des circuits de sortie. Un contrôle plus fin est souvent nécessaire. Les convertisseurs analogique-vers-numérique (CAN) du PICAXE peuvent lire une tension analogique généralement comprise entre +5 V (tension d'alimentation) et 0 V, et la convertir en une valeur numérique représentée sur 8 bits (ou 10 bits pour certains PICAXE). Utilisons donc comme dispositif d'entrée analogique un potentiomètre ; il agit comme un diviseur de tension réglable. Son curseur (borne centrale) est relié à une broche CAN du PICAXE et sa position détermine la tension.

Lorsque nous tournons le potentiomètre vers l'extrémité 0 Ω, le courant à travers la broche CAN doit être limité pour éviter toute détérioration des composants (cf. épisode précédent [1]). L'ajout de résistances pour limiter le courant affectera toutefois aussi les valeurs CAN. Une résistance de 330 Ω entre le curseur et la broche CAN limitera le courant à :

$$\frac{5V}{330\Omega} = 0{,}0152\ A$$

Cette valeur est inférieure au courant limite de 0,02 A que peuvent drainer les broches. Avec le potentiomètre linéaire de 10 kΩ que nous utilisons, la tension maximale aux bornes du CAN est :

$$5V \times \frac{10k\Omega}{330\Omega + 10k\Omega} = 5V \times 0{,}97 = 4{,}84V$$

L'incidence est mineure et peut être compensée de façon logicielle. Un fil du forum PICAXE traite de la valeur des résistances à connecter à l'entrée CAN du PICAXE [4]. Le consensus général est qu'une valeur de 10 kΩ est optimale pour un potentiomètre relié à l'entrée CAN d'un PICAXE (ou d'un PIC de *Microchip*). L'exemple montre comment allumer ou éteindre une LED en se servant de la position du curseur comme d'un interrupteur. Les schémas des **figures 2** et **3** montrent respectivement les circuits d'entrée et de sortie. Le code correspondant est celui du **listage 1**. La **figure 4** montre l'assemblage sur plaque. Dans cet exemple, la LED est mise sous tension pour une valeur CAN comprise entre 2,5 et 5 V, éteinte sinon. Il est évident qu'utiliser ici un interrupteur aurait été plus pratique, mais ce montage nous servira aussi à réguler la luminosité de la LED. Les entrées CAN du PICAXE sont connues pour se comporter de façon inattendue dans certaines situations, en particulier lorsque l'impédance d'entrée dépasse environ 20 kΩ (cf. [4] ou une recherche sur la toile).

Les bases de la MLI

Les PICAXE ne disposent pas de sorties analogiques mais possèdent une (ou plusieurs) sortie MLI (PWM en anglais). Une sortie MLI peut être exploitée à de nombreuses fins et avoir un comportement très proche d'une sortie analogique. La broche de sortie MLI produit en permanence une onde rectangulaire et sa

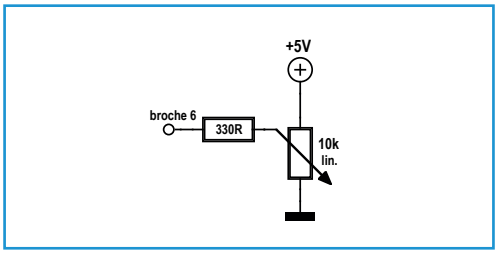


Figure 2.
Schéma pour l'entrée analogique.

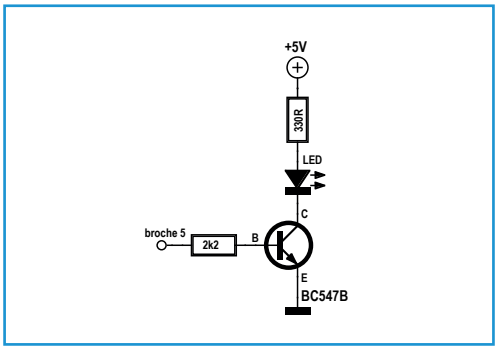


Figure 3.
Schéma pour la sortie MLI.

Listage 1 : entrée analogique

```
init:
low 2
main:
do
  readadc 1,b1
  if b1 > 127 then
    high 2
  else
    low 2
  endif
loop
```

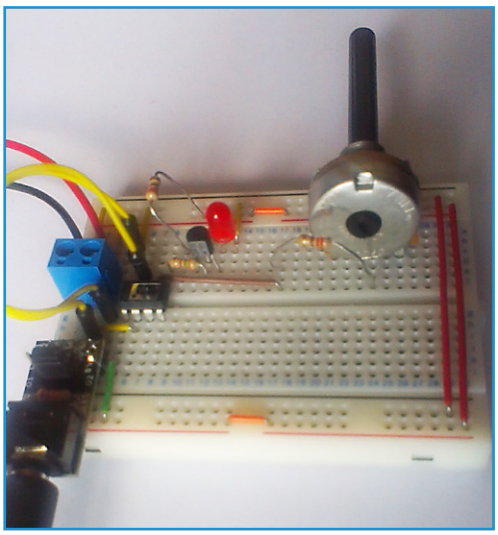


Figure 4.
Montage de la sortie MLI sur plaque d'essai.

commande, contrairement aux commandes PICAXE normales, est exécutée en arrière-plan (parallèlement au reste du programme). Un signal MLI est caractérisé par sa fréquence et son rapport cyclique (**fig. 5**). La fréquence est le nombre de cycles effectués par unité de temps (habituellement la seconde) ; la période équivaut à la durée d'un cycle, et le rapport cyclique est le pourcentage de temps d'un cycle passé à l'état actif (haut). Certaines bibliothèques, p. ex. RPi.GPIO pour le Raspberry Pi [5], permettent de paramétrer directement la fréquence et le rapport cyclique MLI. Les PICAXE ne bénéficient pas de cette facilité logicielle, mais il existe plusieurs commandes, dont `pwmout`, la plus récente et la plus pertinente pour nous. Comme cette commande est complexe, plutôt que de tenter une longue explication nous nous appuierons sur les descriptions du manuel PICAXE [2].
La fréquence du signal MLI est basée sur la fréquence de l'horloge du PICAXE. La plupart des puces PICAXE gèrent plusieurs vitesses d'horloge, mais `pwmout` ne fonctionne qu'avec un nombre limité de fréquences. L'habituelle fréquence d'horloge par défaut, 4 MHz, en fait partie. Les utilisations les plus simples de la commande `pwmout` ont la syntaxe `broche pwmout,période,rapport cyclique` pour activer le signal, et `broche pwmout,OFF` pour le désactiver. Les variables `période` et `rapport cyclique` servent à paramétrer la fréquence MLI (f_{MLI}), la période (P_{MLI}) et le rapport cyclique (R_{MLI}) selon les relations :

$$\frac{1}{f_{MLI}} = T_{MLI} = (période + 1) \times 4 \times vitesse\ du\ résonateur$$

$$R_{MLI} = rapports\ cycliques \times vitesse\ du\ résonateur$$

où la `vitesse du résonateur` vaut 1/4000000 pour une horloge de 4 MHz.

Notez que la commande `pwmout` prend comme argument une période, et non pas un pourcentage comme on pourrait s'y attendre ! L'assistant *PWMout* facilite heureusement les choses. Sous LinAxePad, vous le trouverez sous PICAXE → *Wizards* → *PWMout*. Vous pouvez y entrer la fréquence MLI en Hz et le rapport cyclique en pourcentage (comme c'est l'usage). La commande `pwmout` produit alors un signal MLI aussi proche que possible des spécifications entrées (l'approximation vient de la résolution limitée de la commande `pwmout`).

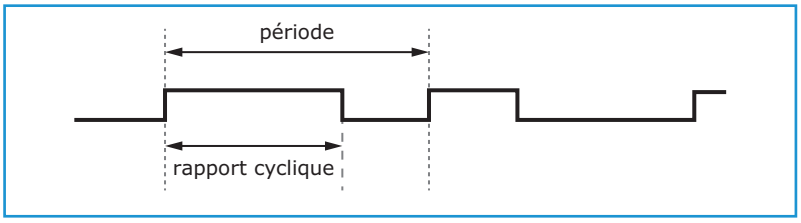


Figure 5.
Structure d'un signal MLI.

Voyons comment commander la luminosité d'une LED par MLI. Nous l'avions dit dans l'article précédent, une LED n'obéit pas à la loi d'Ohm. Pour régler sa luminosité sans utiliser de source de courant (une alimentation qui varie sa tension de sortie pour délivrer un courant constant), on la met rapidement sous et hors tension par MLI. Avec un rapport cyclique de 50 %, la LED ne brille que 50 % du temps. Ce rapport détermine la perception que nous avons de sa luminosité, et la fréquence du signal MLI est si élevée que nous ne percevons que la valeur moyenne de ces commutations.
La fréquence MLI utilisée pour faire varier la luminosité de la LED doit donc être suffisamment élevée pour être imperceptible. La résolution de la sortie doit toutefois être prise en considération – une fréquence proche de la fréquence maximum du PICAXE réduit le nombre de bits disponibles pour le paramétrage du rapport cyclique.
Le principe de la commande de l'intensité d'une sortie par MLI vaut aussi bien pour une LED que pour la vitesse d'un moteur à CC. Le courant que peut délivrer un PICAXE n'est toutefois pas suffisant pour un moteur. Dans ce cas il faut donc recourir à un transistor, comme nous l'avions vu dans l'article précédent [1]. Nous avions également parlé de la vitesse de commutation, une caractéristique à garder à l'esprit lorsqu'on utilise la MLI : si le transistor ne peut pas suivre le rythme des commutations imposé par la fréquence MLI, le signal MLI finit par se distordre et devient alors inutilisable. Ce problème peut être résolu en baissant la fréquence MLI ou en choisissant un transistor plus rapide. Rappelons à ce propos que les paires Darlington sont plus lentes que les transistors seuls.
Le circuit est le même que celui de l'exemple précédent (figures 2 et 3 pour l'entrée et la sortie), mais le programme est un peu plus compliqué (**listage 2**). Les calculs mettent en

correspondance l'intervalle des valeurs d'entrée (0 à 255) et l'intervalle des valeurs de sortie (0 à 100). Nous reprendrons ce principe dans l'exemple suivant.
Le programme utilise le signal d'entrée du CAN (codé sur 8 bits et déterminé par la position du curseur) pour fixer le rapport cyclique d'un signal de sortie MLI chargé de commuter la LED. Et voilà, vous pouvez commander la luminosité de la LED avec un potentiomètre !

De l'utilité du servo

Un servomoteur est un organe de commande mécanique qui permet de commander de façon précise le positionnement d'une pièce. Un servo typique se connecte avec trois fils : noir pour la masse, rouge pour l'alimentation, et blanc (en général) pour le signal de commande. Les servos se commandent par MLI. La largeur d'une impulsion détermine l'angle de rotation du bras de commande. Comme le signal est sans cesse répété, le servo résiste à toute force susceptible de modifier sa position. Les impulsions sont émises toutes les 20 ms (soit à une fréquence de 50 Hz) ; leur durée (comprise entre 0,75 ms et 2,25 ms) est définie par le rapport cyclique. Nous n'avons pas besoin de `pwmout` ici. Nous utiliserons la commande `broche servo,impulsion` pour initialiser une sortie MLI en tant que sortie de servo, et `broche pos_servo, impulsion` pour positionner le servo.
La ligne de commande du servo se relie à la broche de sortie « servo » (à capacité MLI) du PICAXE via une résistance-talon (330 Ω p. ex.), mais le servo a besoin de sa propre alimentation. Un servo peut en effet absorber des courants forts et induire beaucoup de bruit sur la ligne d'alimentation. En pratique, l'absence d'alimentations séparées se traduirait donc par un comportement erratique du PICAXE. Certains servos ont en outre besoin de tensions plus élevées que ce qu'exige le reste du circuit. Le 0 V de l'alimentation du PICAXE et l'alimentation du servo devraient être interconnectés pour fournir un rail de référence commune. Un condensateur est présent entre le V+ du servo et V0 pour réduire les ondulations. Le servo de l'exemple est alimenté en 6 V par 4 piles AA de 1,5 V (non rechargeables car leur tension nominale est d'environ 1,2 V).
La variable `impulsion` doit prendre des valeurs comprises entre 75 et 225 pour correspondre aux largeurs de 0,75 à 2,25 ms attendues par le servo.
Le schéma de la **figure 6** est un exemple de connexion de servo. Le circuit d'entrée analogique est ici encore semblable à celui de la

Listage 2 : sortie MLI

```
init:
low 2
main:
do
  readadc 1,b1  ;get analog value
  if b1 > 250 then  ;clamp value
    b1 = 250
  endif
  w1=b1*10   ;w1 is a 16-bit register (b2 with b3)
  w1=w1/25   ;needed for integer arithmetic
  pwmout 2,24,b2;b2 is the significant part of w1
loop
```

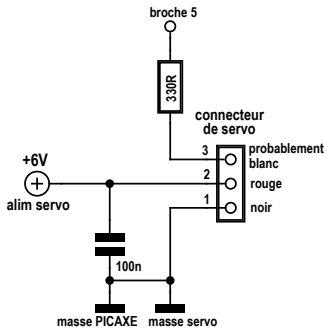


Figure 6.
Schéma du servomoteur.

Listage 3 : commande de servo

```
init:
servo 2,100   ;init servo
main:
do
  readadc 1,b1  ;get analog input
  if b1 > 250 then  ;clamp value
    b1 = 250
  endif
  w1=b1*3  ;scale value
  w1=w1/5
  b2=b2+75
  servopos 2,b2 ;update servo position
loop
```

figure 2. Le **listage 3** contient le code pour commander un servo via le potentiomètre relié à l'entrée analogique. La **figure 7** montre un exemple de montage. Les registres du PICAXE ne stockent que des entiers codés sur 8 ou 16 bits, et nous devons donc contourner cette limitation afin que les valeurs d'entrée du CAN (0 à 255) correspondent à celles de la plage de sortie (0 à 100). Ici l'intervalle des valeurs de sortie court de 75 à 225, soit 225 – 75 = 150 valeurs à mettre en correspondance avec les valeurs d'entrée. Comme 150 / 250 = 3 / 5, nous pouvons multiplier la valeur CAN par 3 (opération qui nécessite une résolution de 16 bits car la valeur maximale codée sur 8 bits est 255, et ici nous avons 3 x 255) puis la diviser par 5. En ajoutant la valeur minimum de 75 nous obtenons une valeur de sortie `b1` qui appartient à l'intervalle des valeurs pouvant positionner le servo, selon la valeur renvoyée par la borne du curseur. Notez que `b2` représente la plus faible valeur sur 8 bits de `w1`, car `w1` est un registre de 16 bits composé des 2 registres de 8 bits `b2` et `b3` (cf. [1]).

Contrôler précisément l'angle du bras de commande est très utile en robotique, mais pour ce genre de projets il faut en général plus de servos que ne peut en commander un seul PICAXE. Une solution pourrait être d'ajouter des puces spéciales pour piloter les servos. Le PICAXE dialoguerait avec ces périphériques, et la carte des pilotes de servo positionnerait et mémoriserait plus de 20 servos par carte. La puce AXE031 permet ainsi de commander jusqu'à 21 canaux [3]. Dans le prochain article, nous verrons comment établir ce dialogue à l'aide du PICAXE.

Du PICAXE au ZICAXE

La MLI est également exploitée par la commande `tune`. Sa syntaxe diffère selon le nombre de sorties MLI présentes sur le PICAXE ; nous utiliserons la version adaptée aux puces à 8 broches. La commande `tune` produit une sonnerie monophonique composée de bips, et peut aussi commuter d'autres sorties (pour faire clignoter des LED au rythme du son produit). Elle nécessite bien sûr une sortie capable d'utiliser la MLI ; le PICAXE 08M2 n'en possède qu'une seule, elle nous suffira. Une autre commande, `play`, permet de jouer des sons préprogrammés.

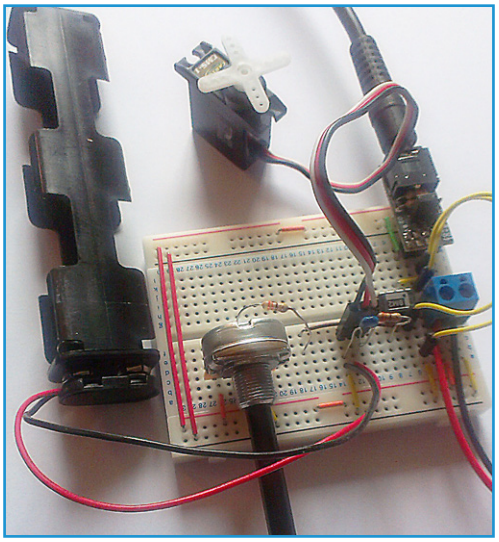


Figure 7.
Le servomoteur sur plaque d'essai.

Le manuel PICAXE [6] décrit de façon exhaustive les possibilités de la commande `tune`, et fournit aussi des exemples de circuits à utiliser avec un haut-parleur, un buzzer piézoélectrique ou encore un amplificateur audio. J'ai quant à moi recyclé une vieille paire d'écouteurs. D'après le manuel, l'impédance du haut-parleur du circuit d'exemple devrait être comprise en 40 et 80 Ω. Si vous mesurez moins, ajoutez une résistance en série de façon à obtenir au moins 40 Ω.
Ici encore un assistant (PICAXE → *Wizards* → *Ring Tone Tunes*) aide à créer des mélodies à l'aide de la commande `tune` (et à importer des sonneries au format RTTTL, p. ex. depuis le site [7]). Cela dit rien n'empêche de passer par `pwmout` pour créer des mélodies ou des effets sonores semblables à ceux des vieux jeux Atari. Conseil : faites varier rapidement les paramètres du signal MLI à l'intérieur de boucles `for` en jouant avec la valeur d'incrément `STEP`, ou imbriquez plusieurs boucles `for`.

La **figure 8** montre le circuit à relier à un haut-parleur (d'après le manuel PICAXE, [6]). Le code associé est celui du **listage 4**. Respectez son format d'une seule ligne, sinon il ne fonctionnera pas. Le montage est reproduit sur la **figure 9**. Peut-être reconnaîtrez-vous la mélodie que produit le code, quoi qu'il en soit amusez-vous avec la commande `tune` (ou `pwmout`) pour jouer aux Beethoven des sonneries de téléphone. Le circuit d'entrée n'est pas nécessaire ici, mais gardez-le, il va à nouveau nous servir.

Une sortie MLI à pulsions analogiques

Les sorties des PICAXE sont toutes numériques. Il est toutefois possible de produire une tension analogique par MLI à l'aide d'un filtre passe-bas. Rappelez-vous du principe du diviseur de tension [1] : lorsqu'un potentiel est appliqué à des résistances reliées en série, la tension aux bornes des résistances individuelles est divisée proportionnellement à leurs résistances respectives. Le même principe vaut pour une onde (rectangulaire) MLI qui traverse un filtre passe-bas. Aux bornes du condensateur, qui remplace la deuxième résistance du diviseur, règne une tension proportionnelle au rapport cyclique du signal MLI. Ceci ne fonctionne toutefois que si la fréquence du signal MLI est très supérieure à la fréquence de coupure du passe-bas. La formule qui donne cette fréquence de coupure est :

$$f_C = \frac{1}{2\pi\ RC}$$

Un produit RC plus grand donne une tension de sortie CC plus lisse, mais un filtre plus lent : lorsque la fréquence MLI change, il faut plus de temps à la tension de sortie analogique pour changer en conséquence. Quels composants utiliser ? Nous pouvons d'abord calculer la valeur de la résistance pour limiter le courant, puis nous servir de la formule du filtre passe-haut pour la valeur de C :

$$C = \frac{facteur}{2\pi\ R\ f}$$

où *facteur* agit comme un multiplicateur qui éloigne la fréquence de coupure de la fréquence MLI. Le site [8] permet de visualiser la forme de la sortie analogique en fonction de la fréquence MLI et des valeurs R et C. Il est recommandé de choisir un facteur d'au moins 100 pour obtenir un signal de sortie sans grande ondulation.

Le PICAXE ne contrôle plus le rapport cyclique de façon précise aux fréquences MLI élevées, donc nous utiliserons une fréquence de 4 MHz / 100 = 40 Hz pour le contrôler sur 100 pas. Le paramètre `rapport cyclique` passé à la commande `pwmout` est maintenant le même que le pourcentage du rapport cyclique : plutôt commode ! Avec une fréquence MLI de 40 kHz et une valeur de 10 kΩ pour

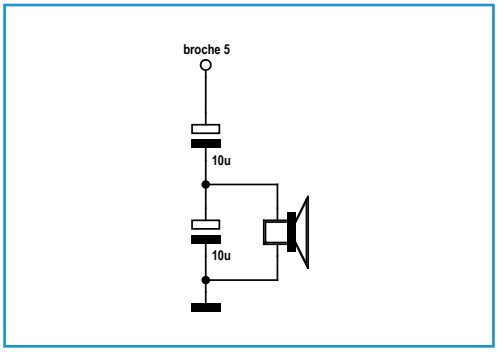


Figure 8.
Schéma du circuit pour la commande `tune`.

Listage 4 : produire un son (une ligne !)

```
tune 1,4,($67,$69,$40,$69,$04,$4C,$04,$22,$4C,$67,$69,$40
,$69,$02,$4C,$02,$00,$4C,$6B,$29,$67,$69,$40,$69,$C0,$02,
$2B,$29,$27,$27,$C2,$E0,$67,$69,$40,$69,$04,$4C,$04,$22,$
4C,$67,$69,$40,$69,$C7,$2B,$20,$6B,$29,$67,$69,$40,$69,$C
0,$02,$2B,$29)
```

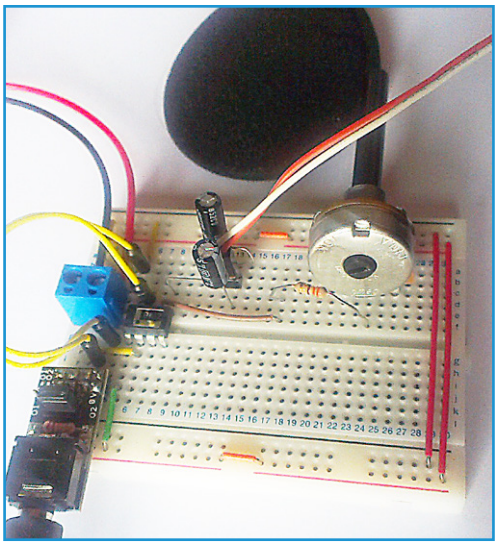


Figure 9.
Montage pour la commande `tune` (avec de vieux écouteurs).

R (pour limiter le courant, comme dit ci-dessus) nous obtenons :

$$C = \frac{100}{2\pi \times 10000 \times 40000} =$$

$$3{,}98 \times 10^{-8} = 39{,}8nF$$

La valeur existante la plus proche est 47 nF. Ici le signal MLI ne sera pas modifié, donc nous pourrions choisir une capacité (bien) plus grande pour obtenir une sortie lisse et dénuée d'effet de bord.

Le circuit de sortie est représenté sur la **figure 10**, celui d'entrée reste le même (fig. 2). Nous pouvons reprendre le code du listage 2 : la fréquence MLI y est déjà de 40 kHz, et le code inclut l'adaptation des valeurs. La **figure 11** montre le montage sur plaque d'essai.

L'entrée analogique est utilisée pour commander la tension de sortie analogique via le rapport cyclique MLI. Le caractère lisse du signal de sortie peut être vérifié et visualisé à l'aide d'un oscilloscope. Un voltmètre suffit si la sortie est suffisamment lisse. Nous pourrions relier la sortie directement à l'entrée analogique du PICAXE pour mesurer la tension de sortie à l'aide du CAN, mais pour l'instant nous ne connaissons aucune méthode pour afficher graphiquement les résultats.

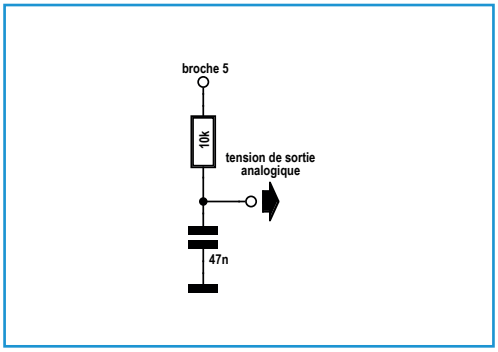


Figure 10.
Schéma pour la sortie analogique.

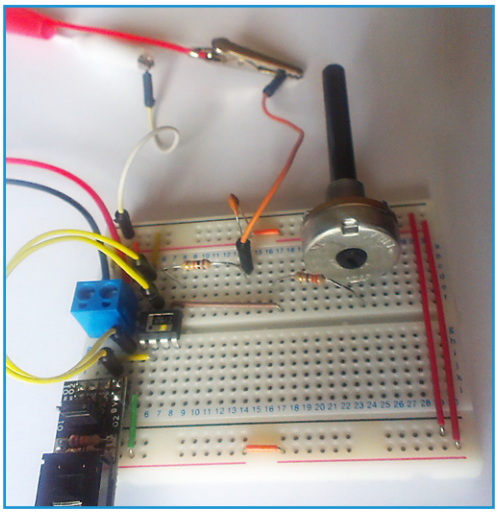


Figure 11.
Montage pour la sortie analogique.

Résumé

Nous savons désormais programmer un PICAXE, commander des périphériques simples, utiliser des entrées analogiques, former un signal analogique à l'aide d'une sortie MLI, régler la luminosité d'une LED, et même produire des sons d'une autre époque et commander précisément la position d'un servo. Comme nous l'avons remarqué à la fin du dernier exemple, nous ne savons pas encore afficher des données, que ce soit pour construire l'interface graphique d'un programme PICAXE, ou tout simplement pour visualiser des données collectées à partir d'une entrée (analogique). Mais pas de panique ! Nous verrons dans le prochain article comment faire communiquer un PICAXE avec différents périphériques, en particulier avec un afficheur OLED. Un bon moyen d'enrichir ses projets !

(130262 – version française : Hervé Moreau)

Liens

[1] PIC atout AXE, (1) et (2), Elektor.POST Projets no 8 et 16
www.elektor-magazine.com/extra/post
[2] www.picaxe.com/Getting-Started/PICAXE-Manuals
[3] www.techsupplies.co.uk/PICAXE
[4] www.picaxeforum.co.uk/forum.php
[5] https://code.google.com/p/raspberry-gpio-python/wiki/PWM
[6] www.picaxe.com/docs/picaxe_manual2.pdf
[7] www.picaxe.com/RTTTL-Ringtones-for-Tune-Command/
[8] http://sim.okawa-denshi.jp/en/PWMtool.php