détecteur de flux d'air Arduino

aucun capteur externe n'est nécessaire !

Source : Adobe Stock

Raymond Schouten (Pays-Bas)

Ce circuit de détection utilise uniquement l'Arduino Nano, sans capteur supplémentaire. Il permet de détecter les flux d'air provenant d'une distance de 30 cm, en mesurant la température interne de la puce chauffée avec une résolution de 0,01°C et en détectant les variations rapides. Ce circuit utilise une technique de **dithering**, permettant d'augmenter la résolution d'un CAN au-delà de celle définie par le LSB.

La détection de flux d'air dans ce circuit repose sur un principe relativement simple. Lorsqu'il est en fonction, le contrôleur ATmega328P de l'Arduino Nano chauffe à environ 6...10°C supérieure de la température ambiante. Une augmentation du flux d'air autour de cette puce entraine son refroidissement : c'est ainsi détection qu'un flux d'air soudain autour de la carte Arduino Nano est détecté.
Cette détection se fait en comparant la nouvelle température mesurée avec celle relevée précédemment, deux fois par seconde. Outre la détection du flux d'air, le système est capable d'identifier d'autres variations de température, tels que celles dues au déplacement de la carte ou au contact avec le boîtier de la puce (**figure 1**).
Veuillez noter que :

- Pour que l'Arduino fonctionne en tant que détecteur, après un boot à froid, il faut lui laisser le temps de se réchauffer (5...10 min). Une fois qu'il a atteint une température « stable », il devient possible de détecter ces flux d'air soudains.
- Ce détecteur est très sensible – les fluctuations de lecture peuvent être <0,02°C - mais un flux d'air provoque des fluctuations plus élevées. Pour évaluer la stabilité de lecture, vous pouvez protéger le détecteur des flux d'air, par exemple en le plaçant dans un sac en plastique. Voir la **figure 2**.

Le concept de lecture de la température

Le CAN interne de la puce peut non seulement lire ses broches d'entrée analogique, mais peut également être configuré pour lire le capteur de température intégré à la puce. Cette tension sera toujours <1 V. Pour augmenter la sensibilité du CAN, la tension de référence est réglée (via le logiciel) à 1,1 V au lieu de l'alimentation standard de 5 V. Avec un CAN de 10 bits (1 024 pas), le plus petit pas mesurable passe alors de 5 mV à 1 mV (chiffres arrondis).
Ce plus petit pas est exprimé par le LSB (bit de poids faible). Selon la fiche technique, la sortie du capteur est de 1 mV/°C, ce qui signifie qu'avec un pas de 1 mV, nous obtenons une résolution de mesure de 1°C - une valeur trop imprécise pour nos besoins. Le paragraphe suivant explique comment améliorer cette résolution à environ 0,01°C/pas.

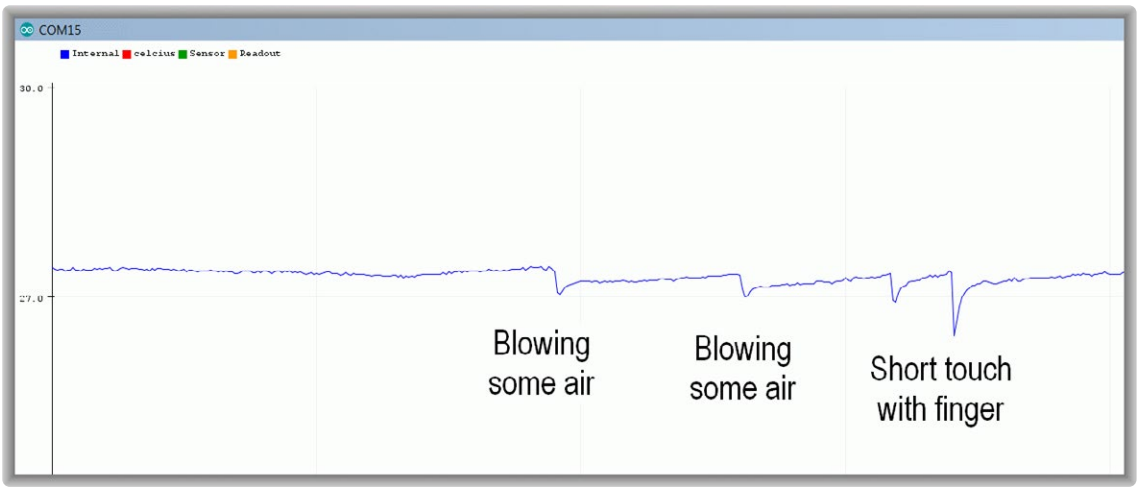

Figure 1. La réponse rapide aux relevés de température du capteur thermique intégré au contrôleur ATmega328 de la carte Arduino Nano.

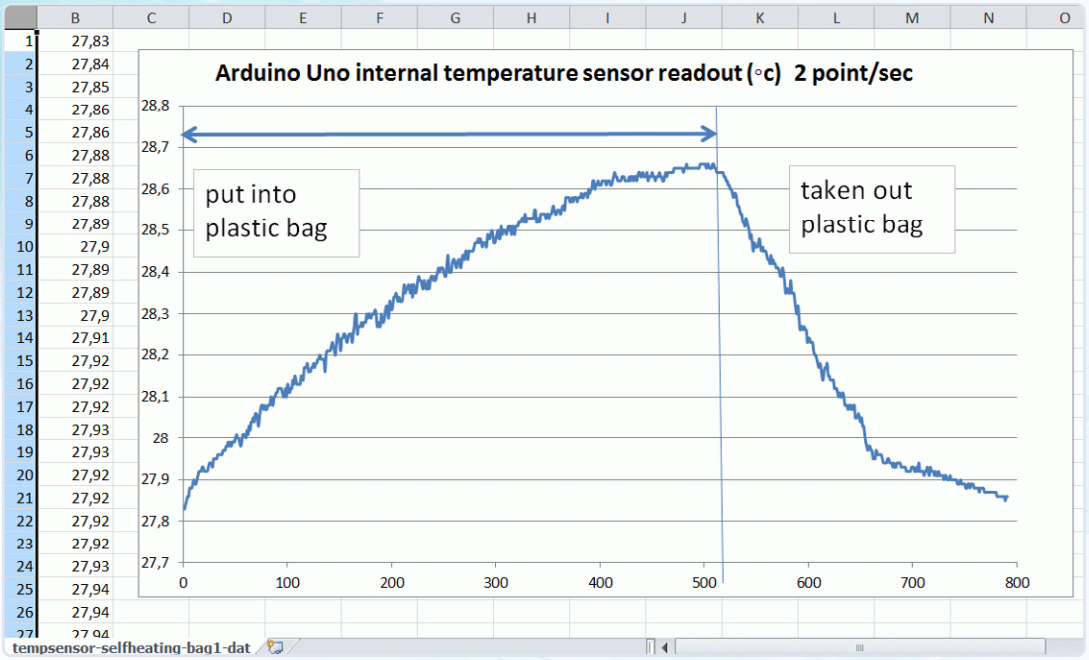

Figure 2. Pour évaluer la stabilité de la lecture, il est conseillé de placer la carte dans un sac en plastique, afin de la protéger de tout changement de flux d'air indésirable.

Augmentation de la résolution de lecture du CAN par dithering

Comme mentionné précédemment, un CAN est généralement limité à la détection des variations (pas) équivalentes à un LSB. Ayant réduit la tension de référence à 1,1 V, chaque LSB correspond maintenant à environ 1 mV.

Le CAN donne alors une lecture échelonnée en multiples de 1 mV. Si. Par exemple, si la tension d'entrée est entre 99,5 mV et 100,5 mV, il donnera toujours des lectures de « 100 mV ». Ainsi, si nous avons un signal d'entrée de 100,1 mV, la lecture indiquera simplement « 100 mV ». Ici, le concept de dithering s'avère utile. Cette technique est l'une des rares applications où la présence de bruit peut être une bénédiction. Pour mieux comprendre le processus du dithering, examinons un exemple pour notre cas spécifique, comme le montre la **figure 3**. Si nous ajoutons un bruit aléatoire au signal de 100,1 mV (supposons que nous ajoutons un bruit d'une amplitude 1 LSB = 1 mV) et effectuons de nombreuses mesures rapides et répétées, nous obtiendrons environ 90 % de lectures à « 100 mV » et 10 % de lectures à « 101 mV ». En calculant la moyenne, nous pouvons déduire que le signal d'entrée est en fait de 100,1 mV. Pour des bruits d'amplitudes plus élevées, la moyenne des mesures converge également vers cette valeur.

Les fluctuations causées par le bruit rendent ce résultat légèrement instable, mais, en général, le calcul de la moyenne de 100 échantillons permet une amélioration de 10×. L'amélioration correspond à la racine carrée du nombre d'échantillons. Dans cette application, la moyenne est calculée sur 6 400 échantillons pour chaque mesure de température. La racine carrée de ce chiffre permet de prévoir une amélioration de 80 fois.

La résolution initiale de notre mesure était de 1°C, de sorte que nous pourrions théoriquement améliorer la résolution à 0,0125°C. Les données de test visibles à droite de la **figure 4** ont tendance à fluctuer de 0,01...0,02°C, ce qui démontre que nous nous en sommes approchés de cette valeur dans la pratique. Vous pouvez télécharger le fichier de données sur la page de ce projet sur Elektor Labs [1] et consulter les résultats dans la vidéo de démo [2].

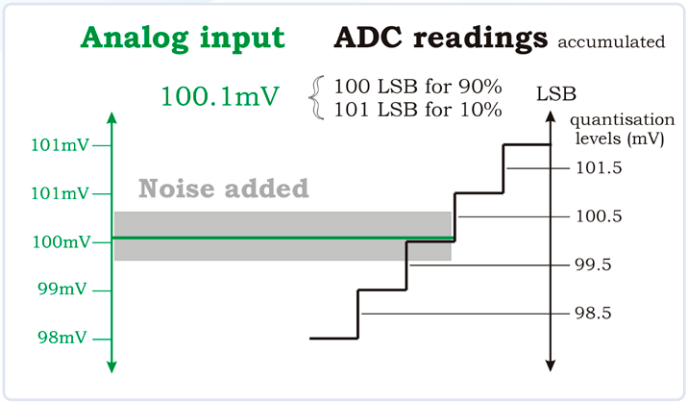

Figure 3. Les relevés de la composante bruit sont ajoutés à ceux du signal primaire, puis une moyenne est calculée à partir de ces valeurs.

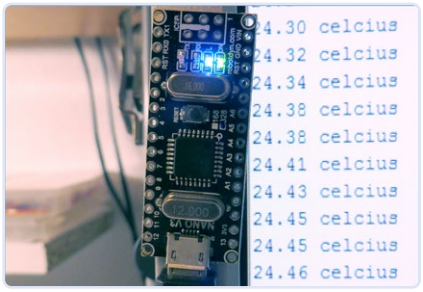

Figure 4. La carte Arduino Nano utilisée dans ce projet, avec les lectures indiquées à droite.

Précision et exactitude

Il est important de noter que les lectures se caractérisent par une grande précision (fluctuations <0,02°C) mais elles ne sont pas très exactes à ce niveau, ce qui signifie que les lectures à faible fluctuation (précision) peuvent être décalées de plusieurs degrés par rapport à la valeur réelle (manque d'exactitude), comme illustré dans la **figure 5**. Pour les applications détectant des variations relatives de la température, sans nécessiter une précision absolue, une grande précision est suffisante.

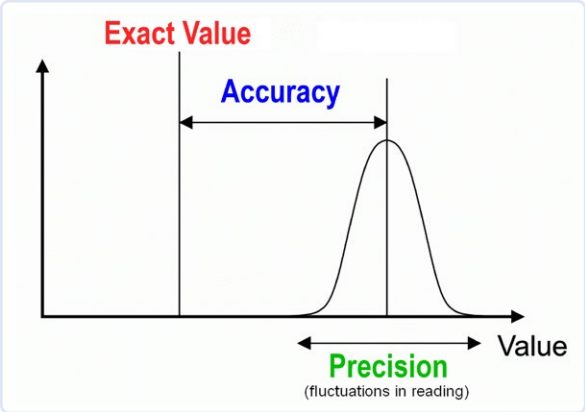

Figure 5. Relation entre la précision et l'exactitude (Source : Wikimedia Commons, Pekaje / Anthony Cutler, https://commons.wikimedia.org/wiki/File:Accuracy_and_precision.svg)

Logiciel

Nous avons utilisé un Arduino Nano, mais vous pouvez utiliser d'autres cartes avec le même contrôleur ATmega328P (Arduino UNO ou Arduino Pro Mini). Le code source de ce projet est disponible sur [1], et les étapes sont plutôt simples :

1. Initialiser en définissant la tension de référence du CAN et l'entrée du multiplexeur de manière appropriée pour la lecture du capteur de température interne.
2. Accumuler 6 400 lectures du CAN afin de déterminer la température avec une haute précision.
3. Envoyer le résultat via USB.
4. Vérifier si la lecture a chuté soudainement. Si c'est le cas, éteindre le voyant indicateur ; sinon, l'allumer.
5. Passer à la phase 2.

Limites

- Ce projet fonctionne mieux avec les cartes Arduino équipées de la version CMS du microcontrôleur. Le boîtier DIL, plus volumineux, résiste plus longtemps aux variations de température, a une meilleure isolation thermique et n'est donc pas adapté à cette application.
- Le processus de dithering ralentit la vitesse de mesure, car nous devons collecter de nombreuses lectures. Pour cela, nous utilisons un CAN de 10 kHz, ralenti à environ deux lectures par seconde, ce qui est suffisant pour les mesures de température.
- Un CAN n'est pas parfait, et peut présenter des inégalités dans ses pas les plus fins, ce qui peut entrainer des erreurs supplémentaires dans les résultats. Pour en savoir plus sur la DNL (non-linéarité différentielle), consultez [3].
- Pour les utilisateurs avancés : un bruit d'amplitude supérieure à 1 LSB contribue également à atténuer les erreurs de DNL du CAN.
- Si le bruit n'est pas uniformément réparti parmi toutes les valeurs mesurables, cela peut affecter les résultats.

Comment avons-nous introduit la source de bruit nécessaire au dithering ?

En réalité, nous n'avons pas eu besoin de le faire. À ces faibles niveaux de LSB de 1 mV, le bruit est naturellement présent dans les composants du circuit, y compris le capteur lui-même. À titre de comparaison : lors de la lecture d'une tension d'entrée en utilisant la tension de référence de 5 V du CAN , j'ai constaté que le bruit ambiant « naturel » était trop faible pour que le dithering soit efficace.
Comment en suis-je arrivé à cette conclusion ? La plupart des lectures restaient centrées autour des pas de 5 mV LSB. Je compte démontrer comment résoudre cela pour réaliser un voltmètre à haute résolution dans un futur article.

Quelques idées (non testées)

Utilisation comme thermomètre

En réduisant la dissipation thermique moyenne du contrôleur, vous pouvez utiliser ce circuit comme thermomètre. Pour cela, mettez le contrôleur en mode veille et réactivez-le toutes les quelques secondes pour effectuer des mesures.

Lecture des capteurs externes

Vous pourriez connecter des diodes externes aux broches d'entrée analogique, en utilisant une faible résistance de tirage vers le haut pour la polariser avec un faible courant. Ainsi, vous pourriez mesurer plusieurs températures externes avec une grande précision (mais une faible exactitude) en utilisant le même concept de code.

220615-04

À propos de l'auteur

Outre son emploi principal où il développe des instruments électroniques à faible bruit, Raymond Schouten s'adonne à des projets personnels en concevant de petits synthétiseurs musicaux et d'autres circuits compacts. La plupart de ses projets visent à obtenir des résultats optimaux avec le matériel le plus simple. Il partage souvent ses projets sur Elektor Labs, YouTube et son site web personnel rs-elc.nl.

Questions ou commentaires ?

Envoyez un courriel à l'auteur (rs.elc.projects@gmail.com) ou contactez Elektor (redaction@elektor.fr).

Produits

- **Arduino Nano**
  www.elektor.fr/17002
- **Livre « *Kickstart to Arduino Nano* », Ashwin Pajankar (Elektor, 2022)**
  version papier : www.elektor.fr/20241
  Version numérique : www.elektor.fr/20242

LIENS

[1] Page du projet sur Elektor Labs : https://tinyurl.com/airflowdetector
[2] Vidéo de démonstration sur YouTube : https://youtu.be/0p6hssAf7Xs
[3] Short wiki-note about DNL : https://en.wikipedia.org/wiki/Differential_nonlinearity