Comment changer la fréquence PWM d'Arduino

: Guide épique Robert Brun·11 juin 2021

Le microcontrôleur possède plusieurs temporisateurs qui peuvent exécuter différentes fonctions, telles que la génération d'un signal PWM . Pour que le temporisateur génère un signal PWM, il doit être préconfiguré en éditant le registre du temporisateur. Lorsque nous travaillons dans l'IDE Arduino, les minuteries sont configurées à notre insu dans la bibliothèque Arduino.h et obtiennent en fait les paramètres souhaités par les développeurs. Et ces paramètres ne sont pas très bons : la fréquence PWM par défaut est faible et les minuteries ne sont pas utilisées à leur plein potentiel. Regardons le PWM standard de l'ATmega328 (Arduino UNO/ Nano / Pro Mini) :

Minuteur	Épingles	Fréquence	Résolution
Minuterie 0	D5 et D6	976Hz	8 bits (0-255)
Minuterie 1	D9 et D10	488Hz	8 bits (0-255)
Minuterie 2	D3 et D11	488Hz	8 bits (0-255)

En fait, tous les temporisateurs peuvent facilement émettre un signal PWM de 64 kHz , et le temporisateur 1 – c'est même 16 bits, et à la fréquence qui lui a été donnée Arduino, pourrait fonctionner avec une résolution de 15 bits au lieu de 8, et cela, soit dit en passant, 32768 gradations de remplissage au lieu de 256 ! Alors pourquoi cette injustice ? La minuterie 0 est en charge de la synchronisation et est réglée de manière à ce que les millisecondes s'écoulent avec précision. Les autres minuteries sont ramenées à zéro pour éviter que l'amateur d'Arduino n'ait des problèmes inutiles. Cette approche est généralement compréhensible mais aurait fait au moins quelques fonctions standard pour une fréquence plus élevée, eh bien, sérieusement ! D'accord, s'ils ne l'ont pas fait, nous le ferons.

Réglage de la fréquence PWM via les registres

La génération PWM est réglée via les registres de temporisation. Ensuite, vous trouverez des “morceaux” de code prêts à l'emploi, que vous devez insérer dans setup(), et la fréquence PWM sera reconfigurée (le pré-délimiteur et le mode minuterie changeront). Vous pouvez toujours travailler avec le signal PWM avec la analogWrite()fonction, contrôlant le remplissage du PWM sur les broches standard.

Modification de la fréquence PWM sur l'ATmega328 (Arduino UNO/Nano/Pro Mini)

Broches D5 et D6 (Timer 0) – 8 bits

pwmD5D6.ino

```
// Pins D5 and D6 are 62.5kHz
```

```
TCCR0B = 0b00000001; // x1
TCCR0A = 0b00000011; // fast pwm
// Pins D5 and D6 - 31.4 kHz
TCCR0B = 0b00000001; // x1
TCCR0A = 0b00000001; // phase correct
// Pins D5 and D6 - 7.8 kHz
TCCR0B = 0b00000010; // x8
TCCR0A = 0b00000011; // fast pwm
// Pins D5 and D6 - 4 kHz
TCCR0B = 0b00000010; // x8
TCCR0A = 0b00000001; // phase correct
// Pins D5 and D6 - 976 Hz - default
TCCR0B = 0b00000011; // x64
TCCR0A = 0b00000011; // fast pwm
// Pins D5 and D6 - 490 Hz
TCCR0B = 0b00000011; // x64
TCCR0A = 0b00000001; // phase correct
// Pins D5 and D6 - 244 Hz
TCCR0B = 0b00000100; // x256
TCCR0A = 0b00000011; // fast pwm
// Pins D5 and D6 - 122 Hz
TCCR0B = 0b00000100; // x256
TCCR0A = 0b00000001; // phase correct
// Pins D5 and D6 - 61 Hz
TCCR0B = 0b00000101; // x1024
TCCR0A = 0b00000011; // fast pwm
// Pins D5 and D6 - 30 Hz
TCCR0B = 0b00000101; // x1024
TCCR0A = 0b00000001; // phase correct
```