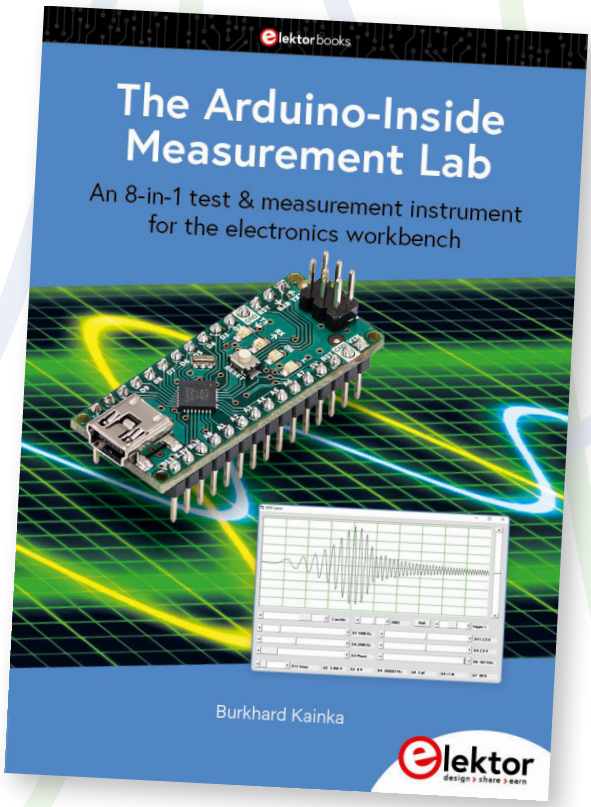
The Arduino-Inside **Measurement Lab**

un instrument de test et de mesure 8 en 1 pour le labo d'électronique

Burkhard Kainka (Allemagne)

Le livre d'Elektor portant le titre ci-dessus explore en détail un ensemble d'instruments de test et de mesure contrôlés par Arduino. Il aborde leur théorie de fonctionnement, leur programmation, et évalue leurs avantages ainsi que leurs limites. Cet article examine une section avancée du livre, où certaines des fonctions essentielles, telles que la génération et la mesure de fréquence, reçoivent des améliorations logicielles dessinées à accroitre la précision et à optimiser leur usage pratique dans le labo domestique. Si vous envisagez d'incorporer l'Arduino dans votre labo, il trouvera parfaitement sa place, sur votre établi, au cœur d'un instrument de test multifonctionnel et entièrement *DIY*, conçu pour une utilisation pratique.



Note de l'éditeur : *cet article est un extrait du livre d'Elektor : The Arduino-Inside Measurement Lab formaté et légèrement modifié pour correspondre aux normes éditoriales et à la mise en page du magazine Elektor. L'auteur et l'éditeur seront heureux de répondre aux questions – pour les contacter, voir l'encadré «* **Questions ou commentaires ?** *». En allemand, l'acronyme « MSR » est l'abréviation de « messen, steuern, regeln », qui se traduit en français par « mesurer, contrôler, réguler ». Cette terminologie est également conservée dans la version anglaise du livre afin de correspondre au logiciel de contrôle spécialement développé par l'auteur pour le projet.*

L'Arduino Nano a beaucoup plus à offrir que ce qui a été abordé jusqu'à présent (dans le livre, NDLR). Plusieurs ports, entrées analogiques et timers sont encore disponibles. Essayons donc de considérer toutes les options disponibles qui peuvent être utilisées simultanément. Le but reste d'utiliser toutes les fonctions comme si elles appartenaient à des appareils indépendants. Le laboratoire *MSR* se développe ainsi sans avoir recours à du matériel supplémentaire. Le but ultime du développement est de combiner plusieurs fonctions :

- Oscilloscope avec jusqu'à deux voies, axes temporels commutables et fonctions de déclenchement
- Deux générateurs d'ondes sinusoïdales DDS
- Sorties d'ondes carrées DDS ajoutées
- Deux sources de tension réglables

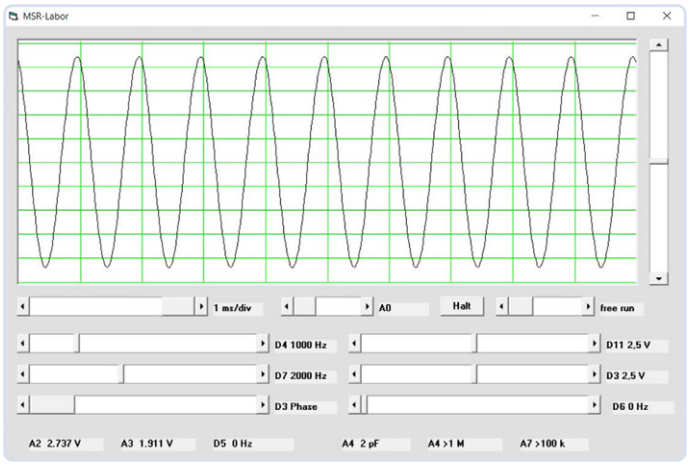


Figure 1. Fonction et connectivité supplémentaires pour «MSR», le noyau logiciel du labo de mesure Arduino-Inside.

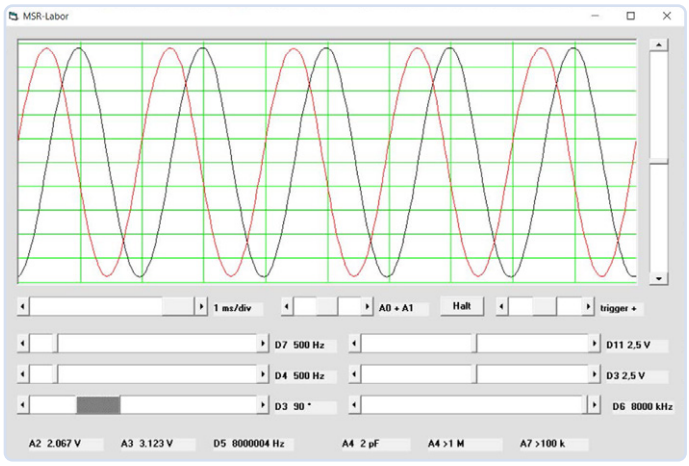


Figure 2. Déphasage de 90 degrés.

En ce qui concerne le logiciel, le programme du labo MSR et d'autres informations sur le projet sont disponibles à l'adresse [1], où vous trouverez également des informations complémentaires dans la publication de l'auteur (en allemand à l'origine).
Pour un bon aperçu, même avec beaucoup plus d'entrées et de sorties, toutes les fonctions sont nommées selon les désignations des broches sur l'Arduino. La **figure 1** montre un exemple d'écran du MSR U/I en fonctionnement et indiquant la convention des noms de broches.

Réglage de la phase du DDS

La table de sinus du générateur DDS (discutée ailleurs dans le livre, NDLR) a une longueur de 256 octets. Pour une période complète, l'octet de poids fort de l'accumulateur de phase doit passer par la plage de 0 à 255. Ainsi, avec la même fréquence sur les deux voies, une relation de phase de 360 degrés au total est représentée dans cette plage. En conséquence, `a1 = 0x0000` est défini, et un paramètre d'un octet reçu est décalé vers l'octet de poids fort de `a2`. La commande `82` déclenche ainsi un saut de phase sur les deux voies, après quoi la relation de phase souhaitée s'applique. Le **listage 1** montre le code du programme concerné.
Dans le programme utilisateur, la différence de phase recherchée est réglée avec le curseur `HScroll9` (**figure 2**). À chaque opération, la commande `82` est envoyée avec le nouvel octet de phase.

Générateur de signaux jusqu'à 8 MHz

Il est possible d'utiliser Timer 0 (avec une résolution de 8 bits) pour générer un signal carré symétrique. Pour l'initialisation, `TCCR0A = 0x42` est défini. Le registre suivant définit le prescaler. Avec `TCCR0B = 0x00`, le générateur est éteint, et avec `TCCR0B = x01` il a la pleine fréquence d'horloge de 16 MHz. D'autres niveaux de prescaler atteignent une division de 1 024. La fréquence exacte est réglée avec OCR0A = 255 (/256, fréquence la plus basse) à `OCR0A = 0` (/1, fréquence la plus élevée). Le compteur s'incrémente et saute à 0 chaque fois que OCR0A est atteint. Simultanément, la sortie OC0A est basculée sur le port D6. Il en résulte que la fréquence la plus élevée est de 8 MHz. La fréquence la plus basse est de 16 MH z / 2 / 1,023 / 256 = 30.528 Hz.

```
TCCR0A = 0x42; // Timer0 Toggle OC0A
TCCR0B = 0x00; // off
OCR0A = 255;
```

Pour régler la fréquence, deux octets doivent être transmis pour le prescaler et le timer. La commande `90` a été définie à cet effet.

```
if (c == 90){ // OC0A frequency
  TCCR0B = USART_Receive();
  OCR0A = USART_Receive();
}
```

Dans le programme utilisateur, chaque changement au niveau de la commande de fréquence `HScroll8` entraîne une nouvelle sortie. Pour un réglage aussi précis que possible, cinq gammes avec des échelles différentes (1, 8, 64, 256, 1 024) sont utilisées. La fréquence de sortie est calculée et affichée sur l'écran de l'utilisateur. Le **listage 2** montre l'extrait de code. La **figure 3** montre les connexions des broches et la **figure 4** montre l'écran utilisateur du MSR.

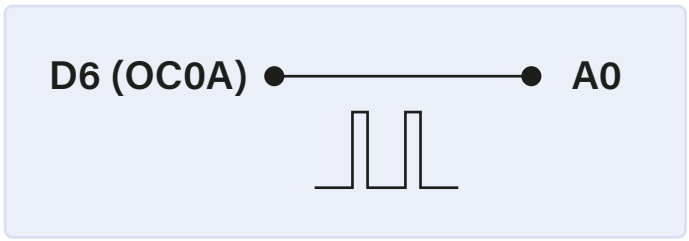


Figure 3. Connexions des broches du générateur de fréquence.

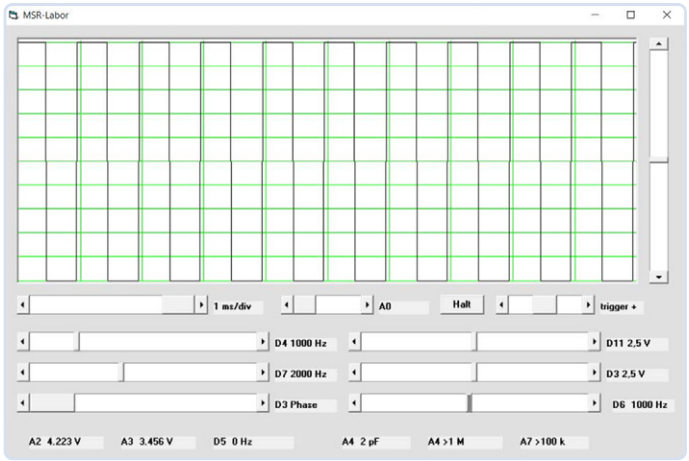


Figure 4. Génération et affichage d'un signal de 1 000 Hz.

Listage 1. Triggering a phase jump.

```
    if (c == 82) { // DDS Phase
      a2 = (USART_Receive()) << 8;
      a1 = 0x0000;
    }

Private Sub HScroll9_Change()
  phase = HScroll9.Value
  Label15 = "D3 " + Str(Round(phase / 256 * 360)) + " °"
  SENDBYTE 82
  SENDBYTE phase
End Sub
```

Listage 2. La fréquence de sortie est calculée et affichée.

```
Private Sub HScroll8_Change()
  d = HScroll8.Value
  If d = 0 Then pre = 0: n = 0
  If d > 0 Then
    pre = 5
    n = 256 - d
    If n > -1 Then f = 8000000 / 1024 / (n + 1)
  End If
  If d > 192 Then
    pre = 4
    n = 448 - d
    If n > -1 Then f = 8000000 / 256 / (n + 1)
  End If
  If d > 384 Then
    pre = 3
    n = 640 - d
    If n > -1 Then f = 8000000 / 64 / (n + 1)
  End If
  If d > 608 Then
    pre = 2
    n = 864 - d
    If n > -1 Then f = 8000000 / 8 / (n + 1)
  End If
  If d > 832 Then
    pre = 1
    n = 1088 - d
    If n > -1 Then f = 8000000 / (n + 1)
  End If

  If f < 100000 Then Label11.Caption = "D6 " + Str(Round(f))
     + " Hz"
  If f >= 100000 Then Label11.Caption = "D6 " + Str(Round(f
     / 1000)) + " kHz"
  SENDBYTE 90
  SENDBYTE pre
  SENDBYTE n
End Sub
```

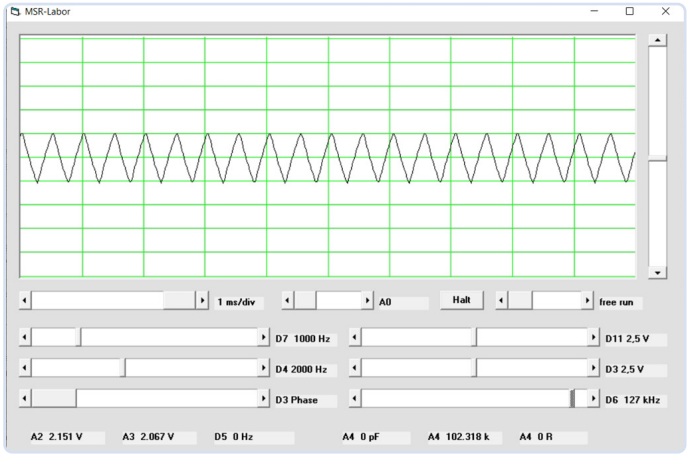


Figure 5. Mesure de l'alias de 2 kHz à 127 kHz.

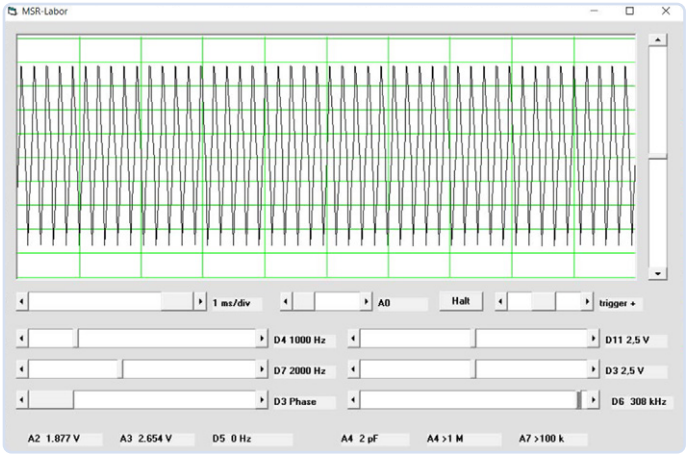


Figure 6. Affichage de la forme d'onde triangulaire à 308 kHz.

Il est possible de régler avec précision de nombreuses valeurs de fréquences importantes, mais la plupart des fréquences sont des fractions impaires de 16 MHz. La résolution est élevée aux basses fréquences et devient plus fine à la limite. Les quatre fréquences les plus élevées sont 2 MHz, 2,667 MHz, 4 MHz et 8 MHz. À titre de comparaison : Le générateur DDS ainsi réalisé a une résolution d'environ 1 Hz sur toute la gamme, mais n'atteint que 5 kHz. Le générateur d'ondes carrées permet de régler des fréquences supérieures à la fréquence d'échantillonnage de l'oscilloscope. Cependant, des mesures fiables avec l'oscilloscope ne sont possibles que jusqu'à la moitié de la fréquence d'échantillonnage, c'est-à-dire jusqu'à environ 31 kHz. Au voisinage du taux d'échantillonnage ou de ses multiples, des courbes complètement erronées sont délivrées. La fréquence d'échantillonnage double est de 125 kHz. Si vous réglez le générateur d'ondes carrées sur 127 kHz, un signal apparent de 2 kHz est généré, c'est-à-dire la fréquence de différence. En principe, ce problème peut être observé avec n'importe quel oscilloscope à mémoire numérique (DSO), alors qu'il ne se produit jamais avec un oscilloscope analogique. La **figure 5** en donne un exemple. On remarque également les bords inclinés de l'oscillogramme, malgré sa véritable forme rectangulaire. Ils sont causés par le temps d'échantillonnage fini du convertisseur AN. Le condensateur d'échantillonnage et de maintien a besoin d'un certain temps pour se charger jusqu'à la tension actuelle. Cependant, à des fréquences très élevées, l'état aura déjà changé pendant le temps d'échantillonnage. Par conséquent, les tensions entre les valeurs limites sont mesurées dans les transitions. À très haute fréquence, des tensions triangulaires sont même affichées, comme l'illustre la **figure 6**.

Mesure de fréquence

Le compteur numérique utilise timer 1, avec une résolution de 16 bits. Comme il n'est possible de compter que jusqu'à 65 535 avec cet arrangement, une interruption est déclenchée à chaque débordement pour incrémenter un compteur supplémentaire. Cela contredit le principe selon lequel il ne devrait y avoir qu'une seule interruption active pour ne pas perturber la sortie DDS en cours. Cependant, l'interruption du Timer1 se produit rarement et seulement lorsque des fréquences supérieures à 65 kHz sont mesurées. Ce listage présente le code correspondant :

```
ISR (TIMER1_OVF_vect)
{
  fh1++;
}
```

```
…
  TCCR1A = 0x00;
  TCCR1B = 0x07; // Timer1 Input
  TIMSK1 = 0x01; // Timer1 Overflow Interrupt
  TCCR1C = 0;
```

Pour l'initialisation, on connecte le compteur à l'entrée T1 du port D5. De plus, l'interruption est activée. L'interruption de Timer2 est également utilisée pour contrôler le temps de porte. Un compteur de temps `t` est configuré ici. À `t = 0`, Timer1 est réinitialisé ainsi que son octet de poids fort `fh1`. A `t = 1`, le timer est démarré. Et exactement une seconde plus tard, il est arrêté et lu :

```
ISR (TIMER2_OVF_vect)
{
  PORTB |= 1;
  …
  t++;
  if (t == 0) { TCCR1B = 0x00; TCNT1 = 0; fh1 = 0; }
  if (t == 1) { TCCR1B = 0x07; }
  if (t == 62501) { TCCR1B = 0x00;f=TCNT1; fh2=fh1; }
  …
  PORTB &= ~1;
}
```

Les 16 bits de poids faible se trouvent alors dans `f`. En outre, les 8 bits de poids fort se trouvent dans `fh2`. La commande `91` a été définie pour la transmission au PC. Dans le programme utilisateur MSR, la lecture de la fréquence est rafraîchie une fois par seconde. Pour ce faire, la valeur mesurée doit être lue dans le cadre de la fonction de temporisation. Un total de trois octets est combiné pour former un nombre de 24 bits. Voici les extraits de code pertinents pour ces opérations :

```
if (c == 91) { // Timer 1 frequency
      USART_Transmit(fh2);
      USART_Transmit(f >> 8);
      USART_Transmit(f & 0xFF);
    }
```

```
CLEARBUFFER
 SENDBYTE 91
 f = READBYTE
 f = 256 * f
 f = f + READBYTE
```

```
f = 256 * f
f = f + READBYTE
Label9 = «D5 « + Str(f) + « Hz»
```

Le fréquencemètre fonctionne en permanence en arrière-plan sans perturber la sortie DDS et l'oscilloscope. Si vous reliez l'entrée D5 à la sortie D8, il est possible de contrôler le taux d'échantillonnage et l'appel régulier de la fonction d'interruption Timer2 (**figure 7**). Ici, 62 500 Hz est affiché (**figure 8**). Si jamais cette fréquence fluctue ou baisse à la suite d'une modification du micrologiciel, cela indique une erreur causée par un délai trop long durant l'interruption.
La sortie de l'onde carrée, D7 ou D4, est la plus adaptée à la mesure de la fréquence du DDS. On peut y trouver des écarts d'un hertz causés en partie par des erreurs d'arrondi. En général, le chiffre le plus bas d'un compteur de fréquence fluctue parce que la fréquence du signal est le plus souvent complètement asynchrone avec la base de temps du compteur.
Dans ce cas, la fréquence de 1 kHz est également confirmée par l'oscilloscope, qui mesure simultanément le signal sinusoïdal correspondant (**figure 9**). Cependant, l'oscilloscope MSR ne peut mesurer que des fréquences inférieures à 31 kHz. Le compteur de fréquence MSR, quant à lui, fonctionne jusqu'à 8 MHz.
Un écart de 4 Hz est observé à la fréquence de mesure la plus élevée de 8 MHz (**figure 10**). Ceci est dû au temps de calcul dans la fonction Timer2, qui provoque une légère prolongation du *gate time*. Plus précisément, il est possible de réduire cette erreur temporelle à 0,5 µs. En général, le compteur de fréquence semble avoir une excellente précision. Cependant, il faut garder à l'esprit que tous les signaux mesurés sont dérivés de la même horloge, à savoir l'horloge système de l'Arduino. Malheureusement, le contrôleur n'utilise pas de quartz mais un résonateur céramique de 16 MHz. Des mesures précises montrent que celui-ci peut avoir un écart allant jusqu'à environ 0,3%, ce qui équivaut à environ 50 kHz à 16 MHz. Par conséquent, avec la fréquence affichée de 8 MHz, une erreur allant jusqu'à 25 kHz est possible.
Il faut tenir compte de ces tolérances lors de toute mesure de fréquence. Cependant, il y a souvent des tâches pour lesquelles seule la précision relative ou l'observation des changements de fréquence (delta-f) est importante. Dans d'autres cas, il faut envisager de retirer le résonateur en céramique et de le remplacer par un quartz et des condensateurs adaptés si nécessaire.

240119-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (b.kainka@t-online.de) ou contactez Elektor (redaction@elektor.fr).

Produits

- **Burkhard Kainka, *The Arduino-Inside Measurement Lab*, Elektor 2024**
  Version papier : www.elektor.de/20818
  Version numérique : www.elektor.com/20819

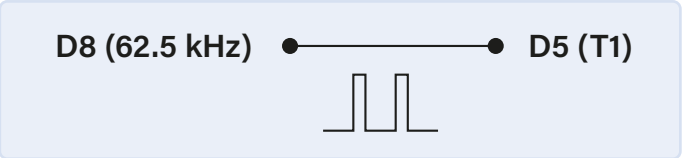


Figure 7. Connexions des broches pour le fréquencemètre.

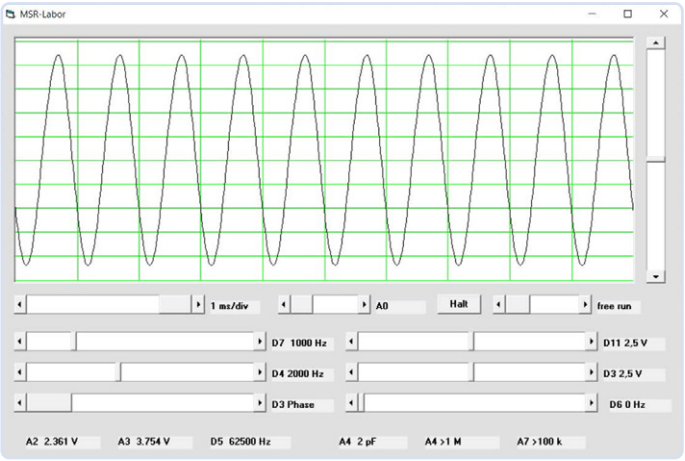


Figure 8. Mesure de la fréquence d'échantillonnage sur D8.

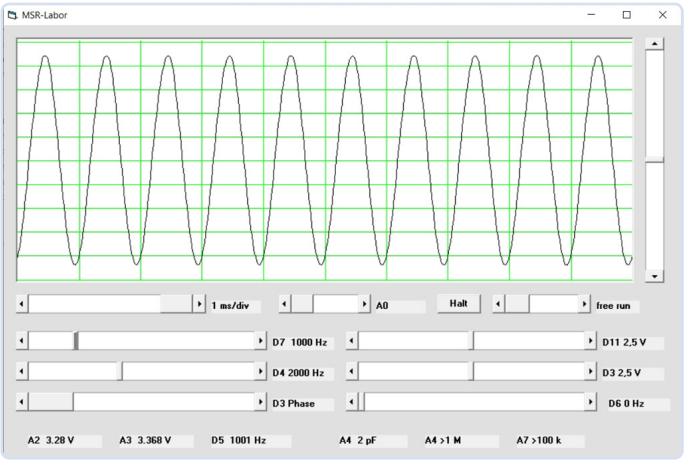


Figure 9. Mesure de la fréquence de sortie de 1 000 Hz générée par le DDS.

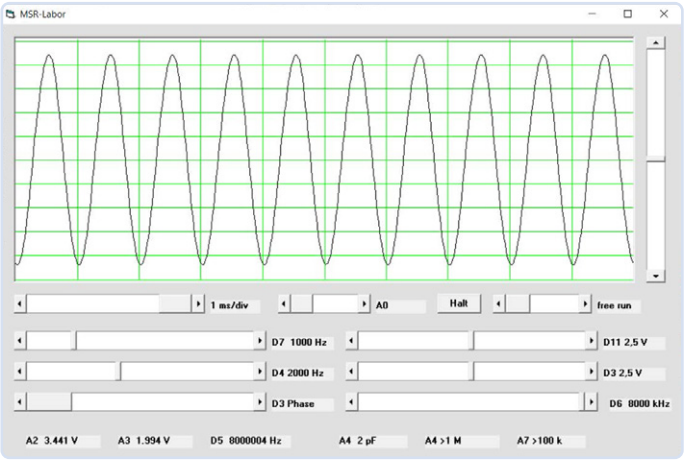


Figure 10. Mesure de 8 MHz sur D6.

LIEN

[1] Logiciel MSR : https://b-kainka.de/MeasurementLab.html