21 charge électronique pour les tests à haute intensité

de la nécessité à l'innovation

Saad Imtiaz (Elektor)

Vous est-il déjà arrivé de vous lancer dans un projet pour réaliser que vous n'aviez pas l'équipement adéquat ? C'est exactement ce qui m'est arrivé : mon module de mesure de puissance AmpVolt devait être poussé à ses limites, mais je n'avais qu'une charge CC de 2 A. Face au manque de temps et de ressources, j'ai dû innover. Avec une configuration basée sur des MOSFET, j'ai pu atteindre plus de 8 A.

L'importance des équipements de test est cruciale dans tous les espaces de travail des électroniciens. Ils marquent souvent la différence entre un résultat « suffisamment bon » et la perfection. Un équipement de test de haute qualité garantit que le matériel conçu fonctionne conformément aux attentes, même dans des conditions variées, et respecte les standards exigeants de l'électronique moderne.

Mais si l'on n'a pas le matériel de test sous adéquat, que l'on manque de temps ou que l'on souhaite simplement le tester immédiatement, il peut être nécessaire de concevoir sa propre solution. C'est ce que j'ai dû faire lors du test des performances de l'*AmpVolt Power Meter* [1], et que je n'avais pas de charge capable d'atteindre plus de 2 A. Cette contrainte m'a poussé à développer une solution petite mais efficace qui peut être utilisée quand on ne dispose pas d'une charge DC adéquate.

Lorsque l'on parle de « charge » en électronique, on pense souvent d'abord à une charge résistive. Cependant, les charges électroniques présentent des avancées significatives par rapport aux modèles résistives classiques, offrant des avantages significatifs dans les environnements de test modernes. Les charges électroniques utilisent des semi-conducteurs de puissance tels que les MOSFET, les IGBT, les BJT, etc. pour s'adapter dynamiquement à des conditions électriques variables, permettant ainsi une simulation précise des comportements électriques sans nécessiter de réglages manuels. Cette capacité permet d'optimiser les processus de test et d'améliorer la précision. De plus, les charges électroniques sont souvent dotées de fonctions de mesure et d'enregistrement de données, associées à une gestion thermique efficace et à des fonctions de sécurité intégrées.

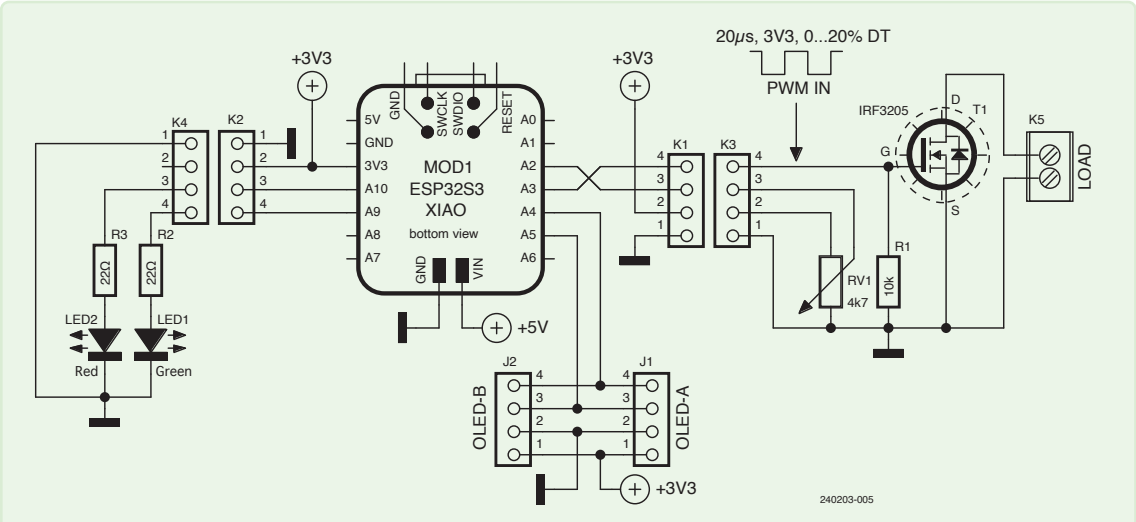


Figure 1.
Schéma du projet.

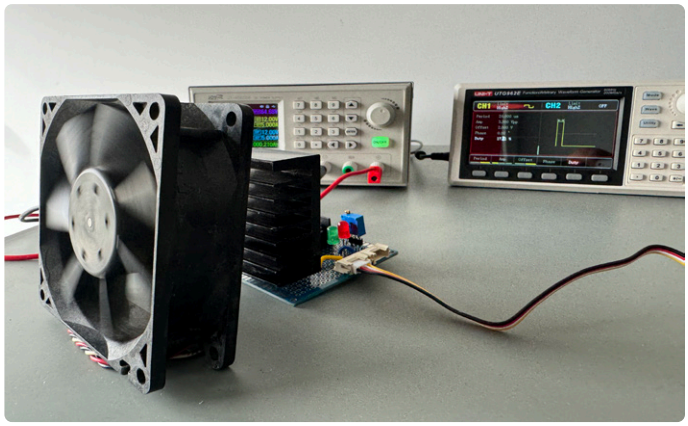


Figure 2. Configuration de test, le générateur de fonctions est utilisé pour contrôler le circuit avec un signal PWM.

Circuit

Le schéma est représenté dans la **figure 1**. Le choix du MOSFET IRF3205 [2] est délibéré. Avec un seuil de 55 V et un courant nominal de 110 A, il est adapté à des tests rigoureux. Avec un dissipateur thermique, il est capable de dissiper efficacement la chaleur, malgré les exigences strictes de la dissipation de puissance continue.

Au cœur du circuit, nous utilisons la petite carte d'extension pour le microcontrôleur XIAO ESP32S3 [3] de Seeed. Pour faciliter le prototypage, j'ai choisi une carte d'extension particulièrement compacte adaptée à ce module XIAO. (Nous avons entretemps développé notre propre carte d'extension Elektor XIAO, détaillée dans [4].) Cette carte d'extension connecte les GPIO du module XIAO à des connecteurs Grove. Chaque connecteur Grove inclut une broche pour GND, une pour 3.3 V, et deux broches pour les entrées/sorties ou le bus. Ces connecteurs sont indiqués sur le schéma du circuit par K2, K1, J1 et J2, avec J1 et J2 spécifiquement dédiés au bus I2C.

K2/K4 sont connectés pour contrôler LED1 et LED2, utilisées pour indiquer l'état de fonctionnement et fournir un retour visuel sur l'état de la charge. K1/K3 sont connectés à une broche de sortie PWM qui commute le MOSFET, et une broche d'entrée analogique qui lit la valeur du potentiomètre. Grâce à ce potentiomètre, le micrologiciel peut ajuster le rapport cyclique du signal PWM, offrant ainsi une modulation précise de la charge.

R1 est une résistance *pull-down* connectée entre la grille et la source du MOSFET, assurant que la grille est correctement déchargée en absence de signal PWM empêchant ainsi toute commutation involontaire du MOSFET.

Le schéma de la charge électronique offre des possibilités d'améliorations, telles l'ajout d'un écran OLED et d'un capteur de courant. Cependant, pour conserver la simplicité du circuit initiale, nous utilisons un potentiomètre pour le contrôle manuel direct des paramètres de la charge. Cette méthode assure un fonctionnement de base tout en permettant l'ajout ultérieur de fonctionnalités plus sophistiquées.

Test

J'ai réalisé un test sur la partie droite du circuit en injectant un signal PWM, généré par un générateur d'ondes, sur la broche 4 de K3. J'ai connecté une alimentation électronique à K5, la source du T1 à la borne négative, et le drain du T1 à la borne positive de l'alimentation. J'ai appliqué un signal PWM avec une période de 20 µs, une amplitude de 3,3 Vpp, et un rapport cyclique variant de 3% à 50% à la grille. J'ai ensuite mesuré le courant traversant le MOSFET à l'aide de l'alimentation numérique et d'un multimètre.

À environ 21,3 % du rapport cyclique, la charge électronique atteignait presque 4,9 A (voir **figure 2)**. En augmentant encore le cycle de travail, on peut augmenter le flux de courant, mais cela conduit aussi à une température élevée du boîtier, atteignant 70°C, sans aucun refroidissement actif, il est donc recommandé d'utiliser un dissipateur de chaleur plus grand avec un refroidissement actif. L'utilisation de deux ou plusieurs MOSFET en parallèle permet également d'augmenter la limite de courant et de réduire les températures élevées.

Logiciel

Le micrologiciel a été développé avec l'EDI Arduino, pour contrôler un signal PWM à l'aide d'un ESP32, un potentiomètre permettant de régler le rapport cyclique et des LED pour indiquer l'état du système (**listage 1**). On commence par définir les broches nécessaires pour le potentiomètre, la sortie PWM et les LED, ainsi que les paramètres du signal PWM, notamment une fréquence de 50 kHz, une résolution de 8 bits et une valeur PWM maximale de 255. Dans la fonction setup, l'interface série est initialisée pour le débogage et les broches des LED sont configurées en sorties. Le signal PWM est appliqué à la broche spécifiée avec la fréquence et la résolution définies. Le code commence par lire la position du potentiomètre et vérifie si elle dépasse le seuil de tension minimal. Si c'est le cas, la LED rouge s'allume et le système attend que l'utilisateur ajuste le potentiomètre à sa valeur minimale.

Dans la boucle main, le code lit en permanence la valeur du potentiomètre, la convertit en tension et associe cette tension à un rapport cyclique PWM allant de 0 à 50 % de l'amplitude du signal PWM maximale. Ce rapport cyclique est ensuite écrit sur la broche PWM. La LED verte s'allume pour indiquer un fonctionnement normal.

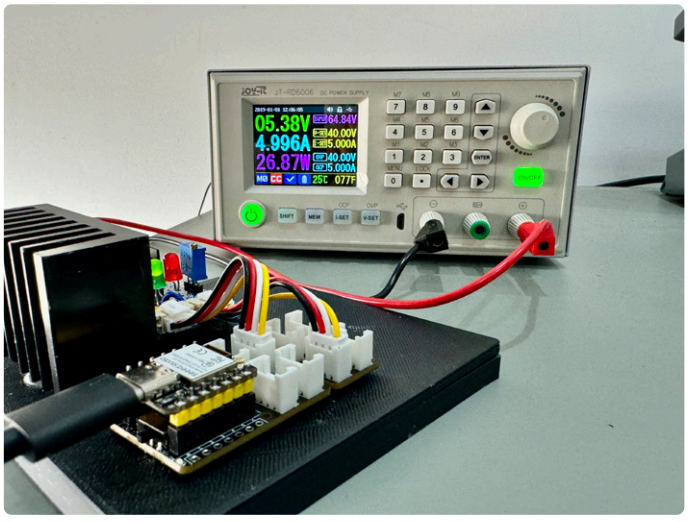
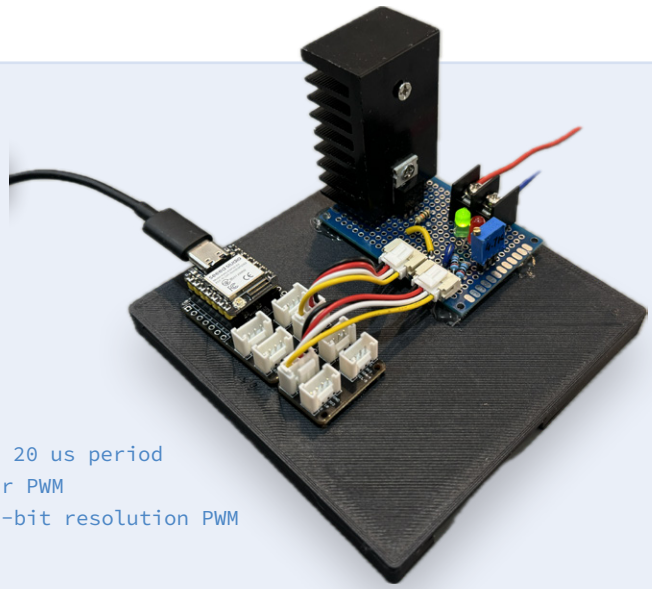


Figure 3. La charge électronique ESP32 est reliée à une alimentation électrique, mesurant et contrôlant le courant.

En outre, le code affiche des informations de débogage, y compris le pourcentage de rapport cyclique et la tension du potentiomètre, sur le moniteur série. Dans la **figure 3**, vous pouvez voir le projet en action, ajustant la charge en fonction du réglage du potentiomètre. Pour ceux qui souhaitent reproduire ou personnaliser ce projet, le code et les schémas sont partagés sur GitHub [5]. Vous disposez ainsi de toutes les ressources nécessaires pour réaliser ce projet et adapter la charge électronique à vos besoins. Comme nous utilisons un ESP32, il est possible de contrôler la charge numérique à distance, sans fil, en installant un serveur web sur l'ESP32 pour contrôler la charge électronique via une interface web en Wifi, ou même à distance via Internet. Les possibilités sont infinies !

240203-04



Listage 1. Micrologiciel.

```
#include <Arduino.h>

// Pin definitions
#define POT_PIN 3
#define PWM_PIN 2
#define RED_LED_PIN 9
#define GREEN_LED_PIN 8

// PWM parameters
const long pwmFrequency = 50000; // 50 kHz to achieve a 20 us period
const uint8_t pwmResolution = 8; // 8-bit resolution for PWM
const uint8_t maxPwmValue = 255; // Maximum value for 8-bit resolution PWM

// ADC and Voltage settings
const float referenceVoltage = 3.3; // ADC reference voltage in volts
const int adcMaxValue = 4095;       // Maximum ADC value for 12-bit resolution
const float minimumVoltage = 0.1;   // Minimum voltage threshold to start PWM

void setup() {
  // Initialize Serial for debug output
  Serial.begin(9600);

  // Set up LED pins
  pinMode(RED_LED_PIN, OUTPUT);
  pinMode(GREEN_LED_PIN, OUTPUT);

  // Initialize PWM on pin
  ledcSetup(0, pwmFrequency, pwmResolution);
  ledcAttachPin(PWM_PIN, 0);

  // Check initial position of the potentiometer
  float initialVoltage = (analogRead(POT_PIN) * referenceVoltage) / adcMaxValue;
  if (initialVoltage > minimumVoltage) {
    digitalWrite(RED_LED_PIN, HIGH); // Turn on red LED
    digitalWrite(GREEN_LED_PIN, LOW); // Make sure green LED is off
    while ((analogRead(POT_PIN) * referenceVoltage / adcMaxValue) > minimumVoltage) {
      // Wait for the user to adjust the potentiometer to minimum
      delay(100); // Delay to avoid excessive reading
      Serial.println("Adjust potentiometer to minimum to start.");
      Serial.println((analogRead(POT_PIN) * referenceVoltage / adcMaxValue));

    }
  }
```

```
  // Potentiometer is at minimum value, proceed with normal operation
  digitalWrite(RED_LED_PIN, LOW);
  digitalWrite(GREEN_LED_PIN, HIGH);
}

void loop() {
  // Read the potentiometer value and convert to voltage
  float potVoltage = (analogRead(POT_PIN) * referenceVoltage) / adcMaxValue;

  // Calculate the PWM duty cycle (0 - 50% of maximum PWM value)
  int pwmDutyCycle = map(potVoltage * 1000, 0, referenceVoltage * 1000, 0, 128);

  // Set the PWM duty cycle
  ledcWrite(0, pwmDutyCycle);
  digitalWrite(GREEN_LED_PIN, HIGH);
  // Debug output to Serial Monitor
  Serial.print("Duty Cycle: ");
  Serial.print((float)pwmDutyCycle / maxPwmValue * 100);
  Serial.println("%");
  Serial.print("Potentiometer Voltage: ");
  Serial.print(potVoltage);
  Serial.println(" V");
}
```

À propos de l'auteur

Saad Imtiaz, ingénieur senior chez Elektor, est spécialisé en mécatronique. Il possède une solide expérience dans les systèmes embarqués, la mécatronique, et le développement de produits. Saad a collaboré avec diverses entreprises, allant de startups à des multinationales, sur des projets de prototypage et de développement. Il a également acquis de l'expérience dans l'industrie aéronautique et a dirigé une startup technologique. Actuellement, il est responsable du développement de projets logiciels et matériels chez Elektor.

Questions ou commentaires ?

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

Produits

- **Joy-IT HD35 Résistance de charge USB (35 W)**
  www.elektor.fr/19164
- **Owon DGE3062 générateur d'ondes arbitraires à 2 canaux (60 MHz)**
  www.elektor.fr/20500
- **Owon SPE6102 Alimentation électrique CC (200 W)**
  www.elektor.fr/20501

LIENS

[1] Saad Imtiaz, « AmpVolt : module de mesure de puissance (1) », Elektor 5-6/2024 : https://www.elektormagazine.fr/magazine/elektor-344/62820

[2] Fiche technique du IRF3205PbF : https://infineon.com/dgdl/Infineon-IRF3205-DataSheet-v01_01-EN.pdf?fileId=5546d462533600a4015355def244190a

[3] Seeed Studio XIAO ESP32S3 : https://seeedstudio.com/XIAO-ESP32S3-p-5627.html

[4] Saad Imtiaz and Jens Nickel, « carte eXpansion d'Elektor v1.0 », Elektor 7-8/2024: https://elektormagazine. fr/230637-04

[5] ESP32 Digital Load, dépôt Github : https://github.com/ElektorLabs/esp32-digital-load/