INNOVATIONS... MONTAGES FIABLES... ÉTUDES DÉTAILLÉES... ASSISTANCE LECTEUR

ELECTRONIQUE

ELECTRONIQUE

ET LOISIRS

magazine

http://www.electroniquemagazine.com

n°149 HIVER 2019

Écrans tactiles Nextion

- L'importance du bootloader

- Sélecteur 3 sorties pour alimentation
- Chargeur de batterie LiPo
- Haut-parleur Tesla
- Capteur de pluie & d'humidité
- Cornemuse électronique
- Attrapez la taupe

N° 149 Décembre 2019

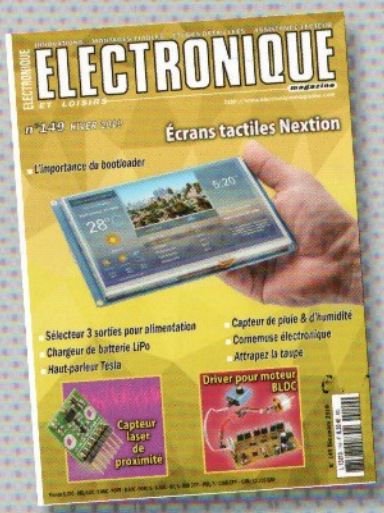
L 13270 - 149 - F: 8,30 € - RD

France 8,30€ - BEL/LUX : 8.80€ - PORT : 8.80€ - DOM/S : 8.30€ - NC/S : 990 CFP – POL/S : 1300 CFP – CAN : 12.70$ CAD

Sommaire

ARTICLES

Numéro 149
Hiver 2019



Les typons des circuits imprimés et les programmes **lorsqu'ils sont libres de droits** sont téléchargeables

Directeur de Publication
Rédacteur en chef
Jean Marc MOSCATI
CD908 13720 Belcodène

Direction - Administration

JMJ éditions
B.P. 20025
13720 LA BOUILLADISSE
Tél.: +334 427 063 96

Secrétariat - Abonnements
Petites-annonces - Ventes
À la revue

Vente au numéro
À la revue

Publicité
À la revue

Maquette - Illustration
Composition - Photogravure
JMJ Editions SARL

Impression
Rotimpres
C/ Pla de l'Estany sn
17181 Aiguaviva (Girona)
Espagne

Distribution
MLP
55 Boulevard de la Noirée
38070 Saint-Quentin-Fallavier

Hot Line Technique
+334 427 063 96 non surtaxé
du lundi au vendredi de 15 h à 17 h

Web
www.electroniquemagazine.com

E-mail
support@electroniquemagazine.com

JMJ éditions

Sarl au capital social de 7800 €
RCS MARSEILLE: 421 860 925
APE 221E

Commission paritaire: 1221 K 79056
ISSN: 1295-9693
Dépôt légal à parution

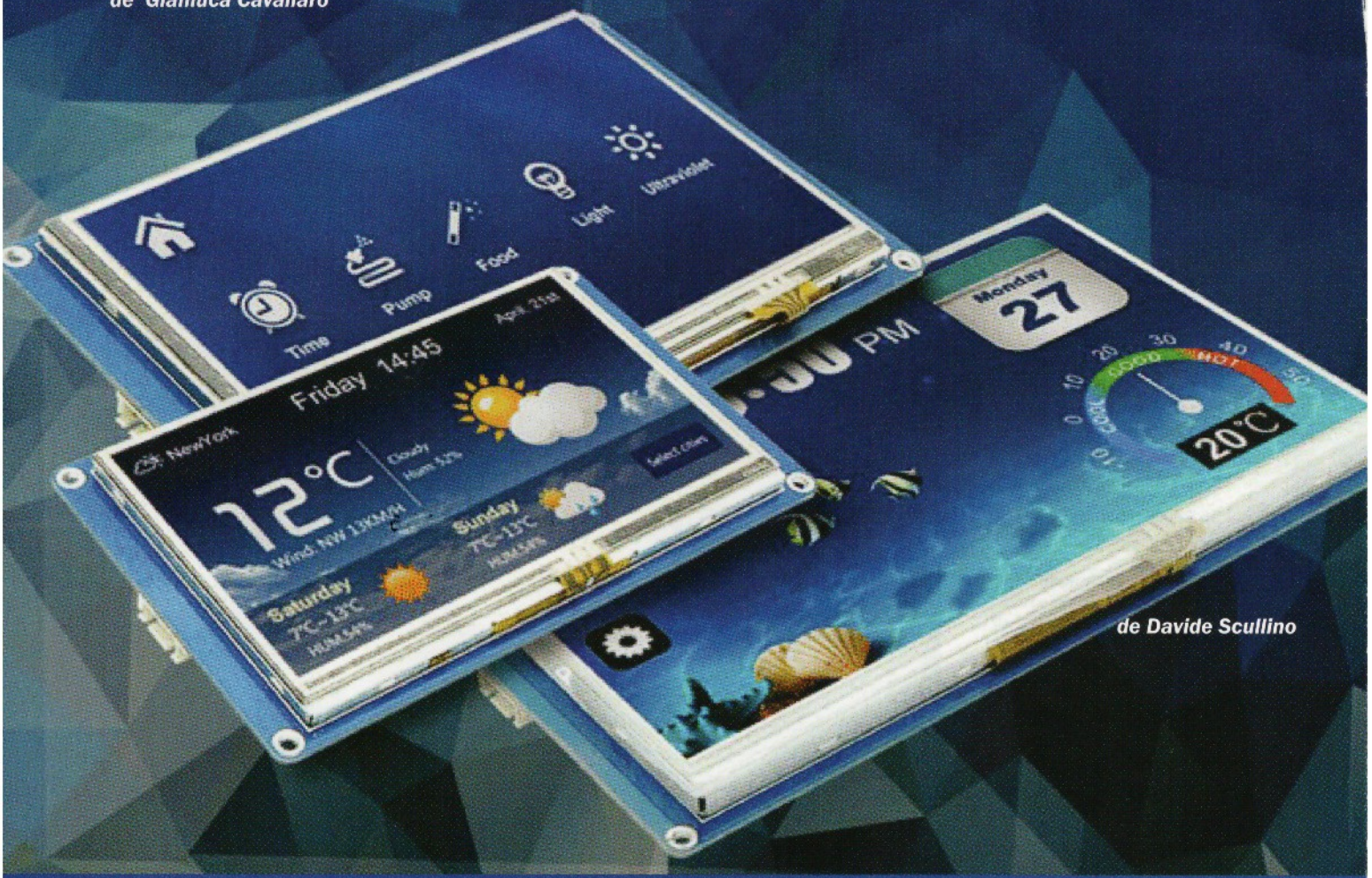
EST RÉALISÉ
EN COLLABORATION AVEC :

NUOVA ELETTRONICA
Elettronica In

à l'adresse **www.electroniquemagazine.com** dans le sommaire de la revue 149 et à l'onglet « Télécharger » (bas de page)

NEXTION : L'ÉCRAN TACTILE

de Gianluca Cavallaro



de Davide Scullino

Dans cet article, nous allons vous montrer comment interfacer des écrans LCD couleur tactiles hautes performances avec Arduino. Voici comment les utiliser.

Grâce à son potentiel et à sa facilité d'utilisation, Arduino est devenu la plate-forme de développement matériel et logiciel Open Source la plus utilisée dans le monde.

Comme vous le savez, il suffit de le connecter à un ordinateur à l'aide d'un câble USB pour pouvoir utiliser immédiatement la carte.

Par exemple, en activant le port série avec l'IDE d'Arduino ou tout autre terminal. Nous pouvons connecter à Arduino de nombreux systèmes électroniques, comme des capteurs, des « breakout board », directement ou par l'intermédiaire de cartes d'extension bien connues qui l'accompagnent.

Mais si nous voulons interagir avec une carte Arduino via un écran tactile, comment pouvons-nous faire ? Le marché offre une variété de solutions qui peuvent parfois compliquer le choix. Certaines utilisent le protocole I2C, d'autres le bus SPI, d'autres encore la communication série RS232 ou TTL.

Nous devons également garder à l'esprit la manière dont il faut charger les images et enfin, le plus important, comment gérer correctement l'événement du « touché » de l'écran.

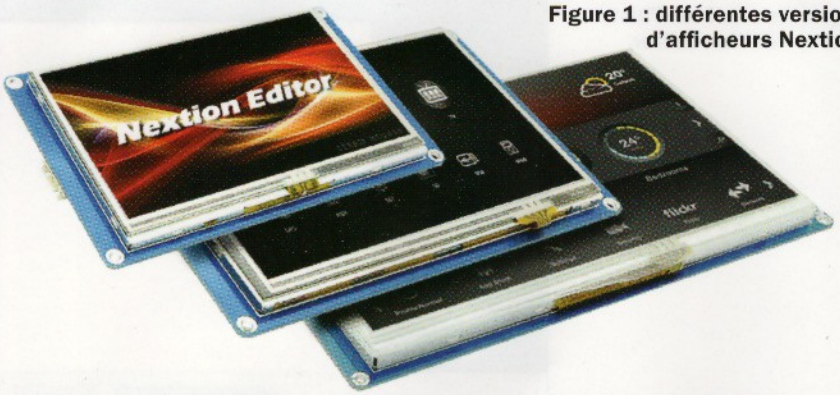


Figure 1 : différentes versions d'afficheurs Nextion.

Si vous vous êtes posés ces questions et que vous vous demandez comment relier des cartes Arduino ou des cartes de développement similaires à un écran tactile de dernière génération et que vous ne savez pas par où commencer, nous avons trouvé la solution à votre problème et nous vous la proposons dans cet article.

Nous profitons de l'occasion pour vous présenter les écrans tactiles de la **série Nextion** (voir la figure 1).

L'objectif de ces pages est multiple. Tout d'abord, nous allons vous montrer les avantages de ces écrans, ainsi que la manière de configurer un écran à la première mise sous tension.

D'autre part, nous vous montrerons comment télécharger les ressources nécessaires et connecter physiquement ces derniers à une carte Arduino.

Enfin, nous allons essayer de créer une interface simple sur l'afficheur avec des boutons pour faire fonctionner le port GPIO d'Arduino en touchant l'écran tactile.

Pourquoi choisir Nextion ?

Nextion est un produit **HMI** (Human Machine Interface) combinant un **écran tactile TFT avec un processeur et une mémoire intégrée**. Pour dialoguer avec une carte Arduino, par exemple, il utilise le **protocole série TTL**. Il gère également les instructions ASCII basées sur le texte afin de coder la manière

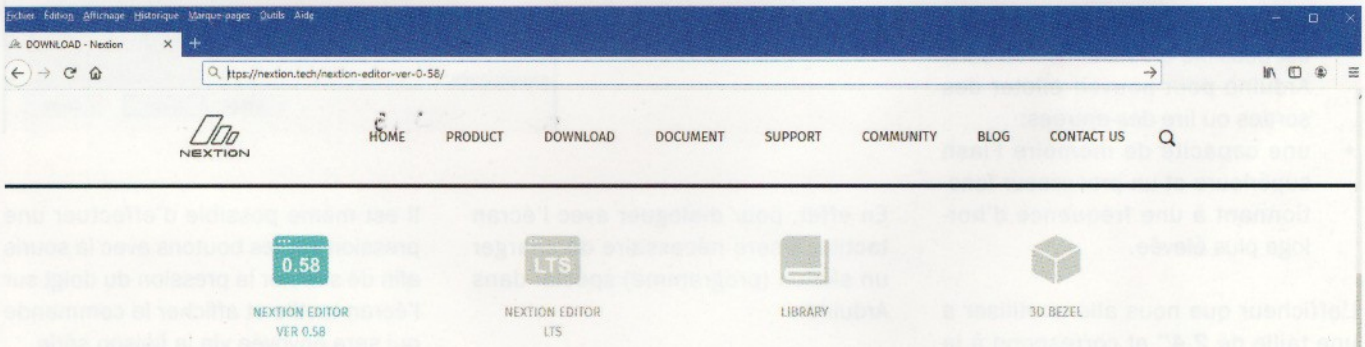


Figure 2 : la page web de téléchargement du logiciel « NEXTION EDITOR ».

dont interagissent les composants avec l'afficheur.

Il offre deux types de mise à jour, le premier en connectant un câble USB au PC avec un adaptateur TTL-RS232 et le second via une carte SD. Dans ce dernier cas, elle est insérée à la première mise sous tension et retirée après la mise à jour.

Dans le commerce, les écrans Nextion sont disponibles sous différents formats, allant d'un minimum de 2,4'' (pouces) à 7'' (pouces). Il existe deux types de versions : la version de base et la version améliorée. Cette dernière, par rapport à la version de base, a en plus :

- une horloge temps réel (RTC) incorporée avec une batterie de sauvegarde ;
- prise en charge de la sauvegarde des données sur une mémoire Flash ;
- supporte le GPIO, dans ce cas, il est possible d'utiliser l'écran sans Arduino pour pouvoir piloter des sorties ou lire des entrées ;
- une capacité de mémoire Flash supérieure et un processeur fonctionnant à une fréquence d'horloge plus élevée.

L'afficheur que nous allons utiliser a une taille de **2,4''** et correspond à la version « **Basic** », car nous voulons l'utiliser via le protocole série afin de piloter les GPIO d'Arduino.

Pour démarrer la démonstration, nous devons avoir les composants suivants :

- un **écran Nextion** avec un câble USB ;
- une **carte Arduino Méga** avec un câble USB ;
- **4 cavaliers** mâles/femelles ;
- un **convertisseur** USB/TTL ;
- le **logiciel d'édition** Nextion.

Commençons par travailler avec Nextion

Passons maintenant à l'utilisation de ces écrans tactiles, nous considérons que nous travaillons dans un environnement Windows et que vous avez déjà installé et utilisé l'environnement de développement d'Arduino sur votre PC.

Figure 2a : fenêtre d'installation de Nextion Editor.

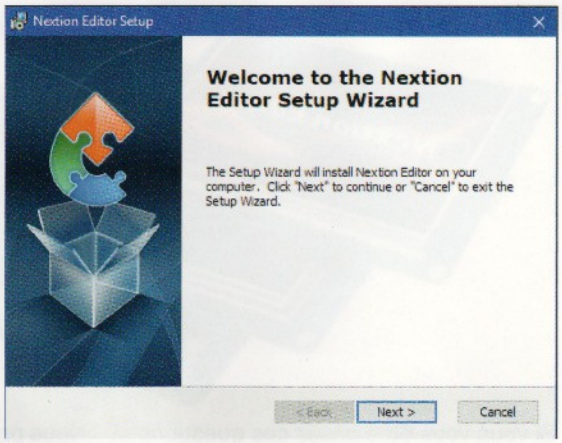
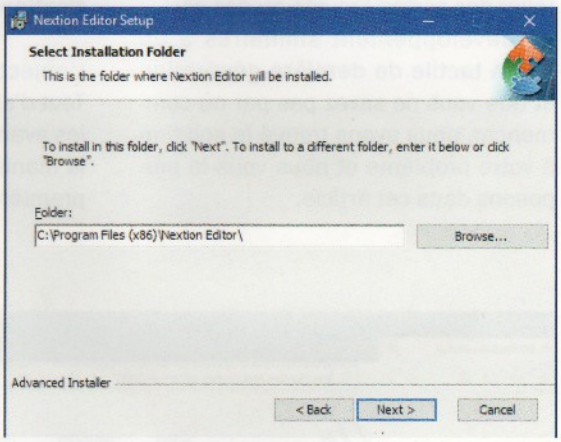


Figure 2b : fenêtre du choix de l'emplacement d'installation de Nextion Editor.



En effet, pour dialoguer avec l'écran tactile, il sera nécessaire de charger un sketch (programme) spécial dans Arduino.

Si vous utilisez pour la première fois Arduino, nous vous conseillons de visiter le site officiel d'Arduino (www.arduino.cc) où vous trouverez des tutoriels et exemples pour en tirer le meilleur parti.

Pour concevoir l'interface graphique, le fabricant a conçu un logiciel appelé « **NEXTION EDITOR** » qui est gratuit et téléchargeable à l'adresse : https://nextion.tech/nextion-editor-ver-0-58/.

Ce logiciel permet de développer rapidement une interface graphique à l'aide de la méthode simple de « **glisser/déposer** » des composants graphiques tels que des zones de texte, des boutons, etc.

Il possède également une fonction de débogage, qui peut également être utilisée sans écran tactile, afin de concevoir tous les graphiques en exécutant une simulation de ces derniers.

Il est même possible d'effectuer une pression sur les boutons avec la souris afin de simuler la pression du doigt sur l'écran tactile et afficher la commande qui sera envoyée via la liaison série.

Pour procéder au téléchargement du logiciel, allez sur le site de Nextion à l'adresse suivante : https://nextion.tech/.

Cliquez ensuite sur l'onglet « DOWNLOAD » en haut dans le menu horizontal. Vous devez voir s'afficher la page web de téléchargement qui doit ressembler à celle de la figure 2.

Enfin cliquez sur le bouton « **EXE Download** » pour ce qui concerne la version fonctionnant sous Windows. Enregistrez sur votre PC le fichier « **nextion-setup-v058.exe** » (au moment où ces pages sont publiées).

Pour installer le logiciel sur l'ordinateur, exécutez le fichier que vous venez de télécharger **en tant qu'administrateur** (clic droit de la souris). La fenêtre de la figure 2a s'affiche, cliquez sur le bouton « **Next** ».

La figure suivante vous permet de choisir l'emplacement d'installation du logiciel. Par défaut l'emplacement est : « C:\ Program Files (x86)\Nextion Editor\ », vous pouvez sélectionner un autre dossier de votre choix à l'aide du bouton « Browse... » (voir la figure 2b).

Nous sélectionnons l'emplacement par défaut et cliquons sur le bouton « **Next** ». La fenêtre de la figure 2c s'affiche alors et vous invite à cliquer sur le bouton « **Install** » pour commencer l'installation du logiciel.

Vous pouvez modifier vos choix précédents en cliquant sur le bouton « **Back** » ou encore annuler l'installation en cliquant sur le bouton « **Cancel** ». Le processus d'installation du programme démarre (voir la figure 2d), à la fin l'icône « **Nextion Editor** » apparaît sur le Bureau de windows.

Une fois le logiciel « Nextion Editor » installé, nous allons créer une interface afin de tester les possibilités de l'afficheur.

Pour ceux qui veulent commencer avec un exemple prêt à l'emploi, sur notre site web vous trouverez un fichier nommé « **Test.HMI** ». Il suffit de le télécharger et d'importer le projet dans l'éditeur.

L'éditeur est très simple et intuitif à utiliser. Commençons par sélectionner l'option « **Open** » en haut à gauche si vous voulez ouvrir le fichier d'exemple.

Si vous voulez commencer un nouveau projet cliquez alors sur « **File** » → « **New** » et attribuez un nom et un emplacement à votre projet. Choisissez ensuite la taille et la version de l'écran tactile utilisé.

Dans notre cas, dans le menu « **Device** » (voir la figure 3), nous sélectionnons celui de base dont la taille est de 2,4 pouces (si vous avez un autre type d'afficheur avec une taille différente, il suffit simplement de le sélectionner).

Vous pouvez également choisir l'orientation à l'aide du menu « **Display** » (voir la figure 3a).

Cliquez sur « **OK** » pour afficher une boîte blanche avec le nom « **PAGE0** ».

Figure 2c : cliquez sur le bouton « Install » pour commencer l'installation de Nextion Editor.

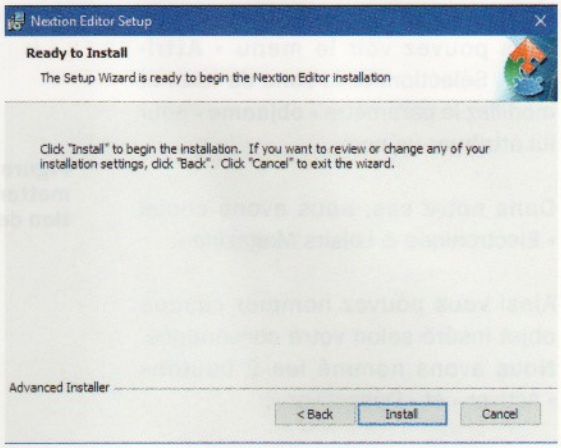
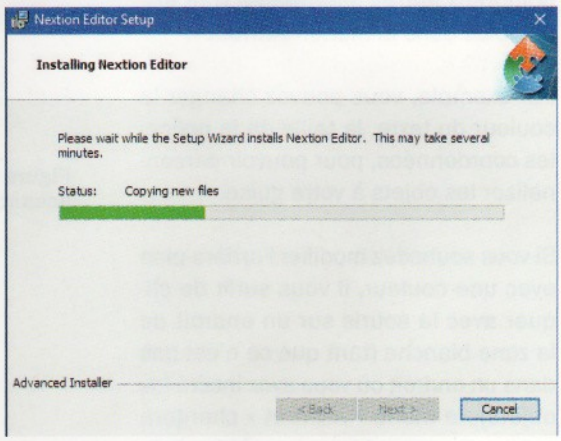


Figure 2d : processus d'installation de Nextion Editor.



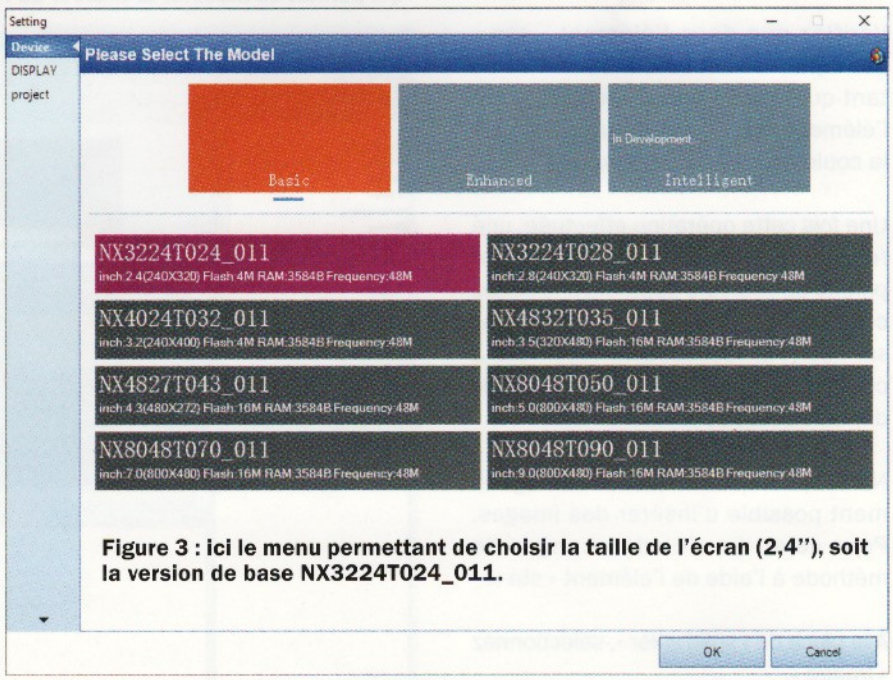


Figure 3 : ici le menu permettant de choisir la taille de l'écran (2,4''), soit la version de base NX3224T024_011.

Grâce à la méthode **glisser/déposer**, vous pouvez sélectionner un composant (qui peut être par exemple un bouton ou une barre de progression) avec le bouton gauche de la souris tout en le faisant glisser dans la zone blanche, puis vous le relâchez dans cette même zone. De la même manière, insérez deux composants « **Button** » et un composant « **Text** » dans la zone blanche, comme illustré en figure 4.

Dans la partie de droite de l'éditeur, vous pouvez voir le menu « **Attribut** ». Sélectionnez la zone de texte et modifiez le paramètre « **objname** » pour lui attribuer un nom.

Dans notre cas, nous avons choisi « Electronique & Loisirs Magazine ».

Ainsi vous pouvez nommer chaque objet inséré selon votre convenance. Nous avons nommé les 2 boutons « Activer » et « Désactiver ».

Comme vous pouvez le constater, de nombreux autres paramètres sont disponibles dans le menu « Attribut ».

Par exemple, vous pouvez changer la couleur du texte, la taille de la police, les coordonnées, pour pouvoir personnaliser les objets à votre guise.

Si vous souhaitez modifier l'arrière-plan avec une couleur, il vous suffit de cliquer avec la souris sur un endroit de la zone blanche (tant que ce n'est pas dans un endroit où vous avez inséré les objets), le menu « Attribut » chargera automatiquement les paramètres de la page (PAGE0).

Vérifiez que dans l'élément « **sta** » du menu « **Attribut** », il est défini en tant que « **solid color** » et à l'aide de l'élément « **bco** », vous pouvez modifier la couleur de l'arrière-plan.

Une fois cette opération effectuée, une fenêtre contextuelle contenant une palette de couleurs apparaît, choisissez celle que vous préférez, puis cliquez sur « **OK** ». Le résultat doit ressembler à celui de la figure 5 en fonction du choix de votre couleur.

Notez qu'en arrière-plan, il est également possible d'insérer des images. Pour cela, il suffit de changer de méthode à l'aide de l'élément « sta ».

À la place de « solid color », sélectionnez « **image** ».

Une fois le menu terminé et personnalisé, cliquez sur le bouton « **Debug** » (barre d'outils en haut vers la gauche). Cela ouvre une fenêtre dans laquelle nous verrons le résultat de la compilation de notre menu.

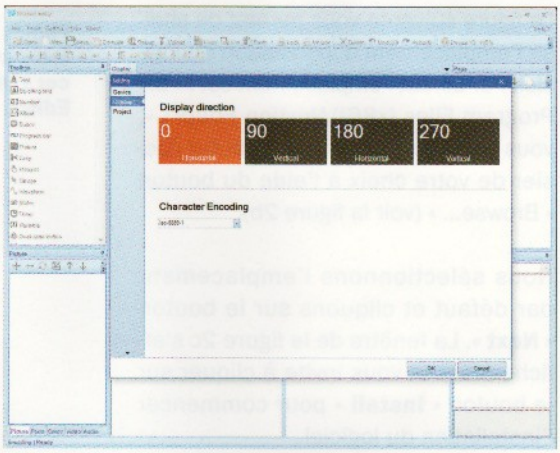


Figure 3a : ici le menu permettant de choisir l'orientation de l'écran (horizontal).

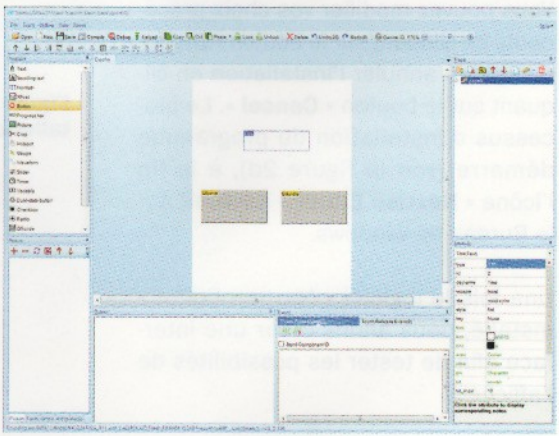
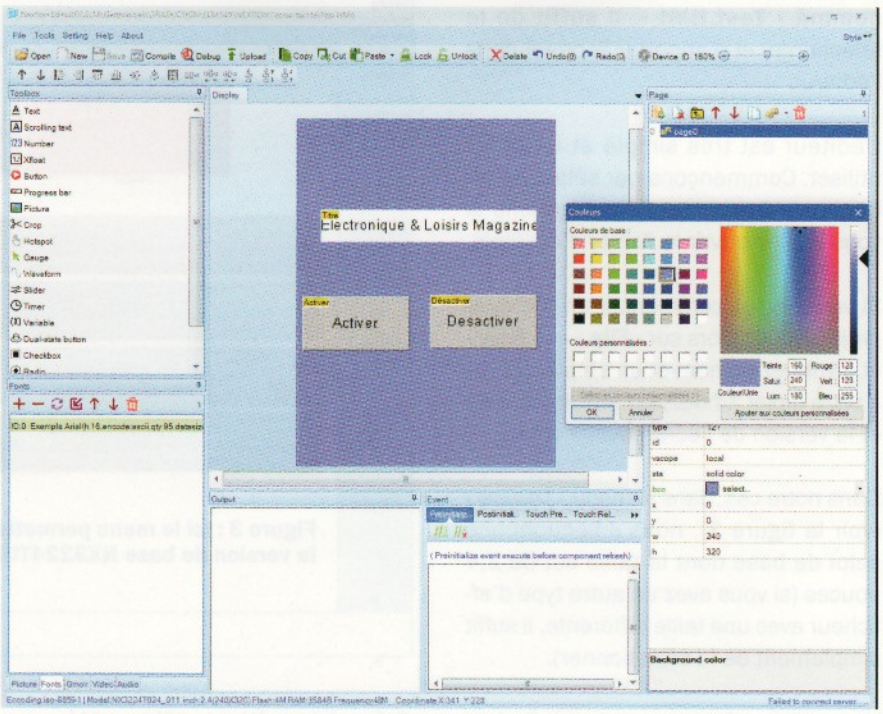


Figure 4 : insertion des composants « Button » et « Text ».

Figure 5 : l'interface avec les éléments positionnés. À droite la palette de couleur qui permet de modifier la couleur de l'arrière-plan.



S'il n'y a pas d'erreurs, nous pouvons enfin charger notre menu dans l'afficheur. Il est possible que des erreurs apparaissent lors de la première compilation.

Ceci est dû au fait que la police de caractère n'est pas chargée. Pour cela, sélectionnez le menu « **Tools** » → « **Font Generator** » et définissez, par exemple, « **Arial** » en donnant un nom à la police,

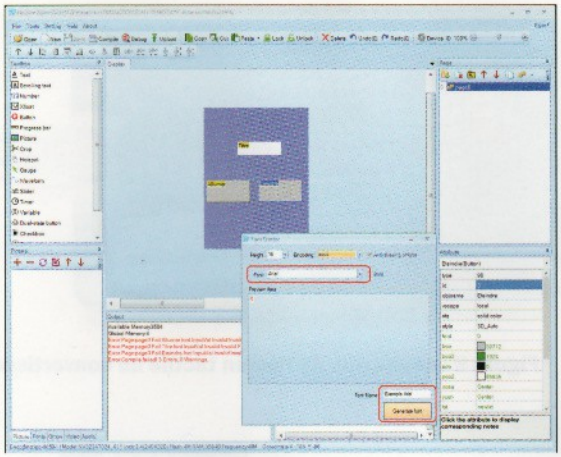


Figure 6 : utilisation de « Font Generator ».

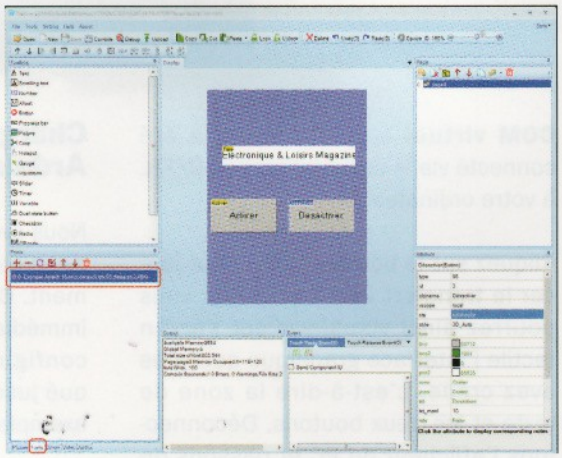
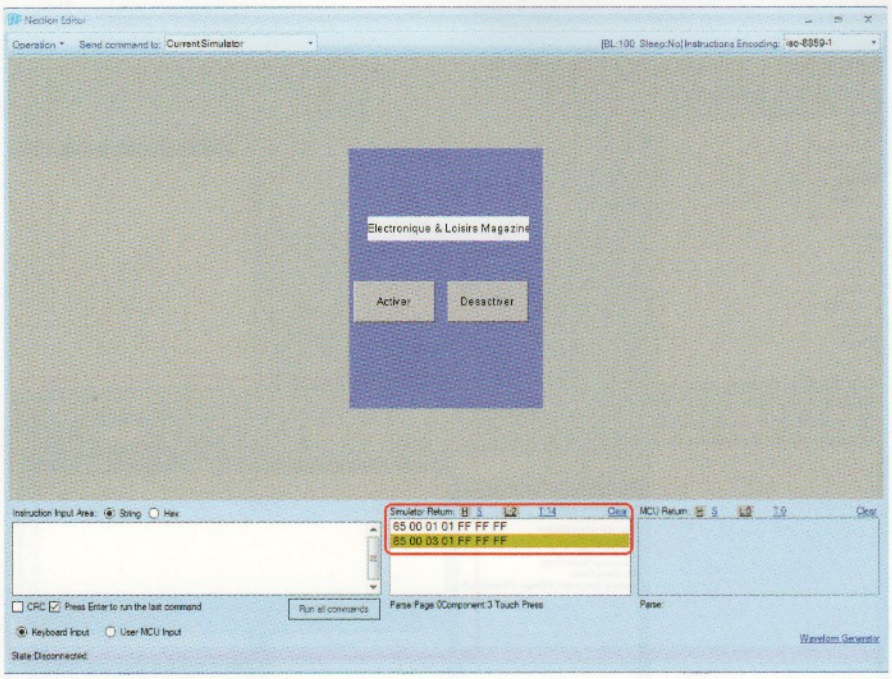


Figure 7 : la police de caractère correctement chargée.

Figure 8 : simulation des messages (chaînes envoyées encadrées en rouge) chaque fois que vous appuyez sur un bouton. Vous pouvez constater que les chaînes sont différentes (65 00 01 et 65 00 03).



« **Exemple Arial** » dans notre cas. Ensuite, cliquez sur « **Generate font** » comme illustré en figure 6. Une nouvelle fenêtre contextuelle s'ouvre pour vous demander où sauvegarder la police.

Nous avons nommé le fichier « **Arial.zi** ». L'extension « **.zi** » **est créée automatiquement**. Enfin, l'éditeur vous demandera si vous voulez **charger la police dans le projet** (Add the generated font ?), répondez en cliquant sur « **Yes** ».

Pour vérifier que la police de caractère a été chargée correctement, cliquez sur l'onglet « **Fonts** » dans la boîte de dialogue en bas à gauche (encadré en rouge en figure 6).

Vous devriez voir afficher **en position 0 la police que vous avez ajoutée** (encadré en rouge en figure 7).

Si la police n'est pas chargée, cliquez sur le bouton « **+** » (Add) dans la fenêtre « **Fonts** » puis sélectionnez le fichier « **Arial.zi** » sur votre ordinateur.

La police est ensuite ajoutée au projet, vous devez voir la syntaxe suivante : **ID:0 Exemple Arial (.....)**.

À ce stade, il sera nécessaire de **cliquer de nouveau sur** « **Debug** » et vérifier que cette fois-ci, il n'y a pas d'erreur.

Fermons la fenêtre de débogage et examinons comment envoyer une instruction à Arduino chaque fois que nous appuyons sur un bouton de l'écran.

Pour cela, nous sélectionnons avec la souris le premier bouton « **Activer** » et dans la boîte de dialogue « **Event** » (en bas vers la droite) nous sélectionnons « **Touch Press Event(0)** » et **cochons la case** « **Send componante ID** » (envoyer l'identifiant du composant).

Effectuons la même opération avec le deuxième bouton (« Desactiver ») et, enfin, cliquons de nouveau sur le bouton « Debug ».

Comme vous pouvez le constater en figure 8, chaque fois que vous cliquez sur l'un des 2 boutons dans la fenêtre de débogage, l'éditeur affiche une chaîne, qui est logiquement différente pour chaque bouton. La chaîne sera envoyée sur le port série, de l'écran vers la carte Arduino.

Il faut maintenant charger dans Arduino un programme permettant à la carte de comprendre la chaîne reçue et d'activer ou de désactiver une ligne GPIO.

Par exemple, la ligne correspondant à la LED présente sur la carte Arduino, ou

de faire commuter une autre ligne en la faisant passer d'un niveau logique 0 à 1 et vice versa.

Concentrons-nous maintenant sur l'aspect matériel

Il est maintenant temps de travailler avec l'afficheur Nextion afin de le préparer à interagir avec la carte Arduino.

Commencez par connecter le convertisseur USB/TTL à l'écran tactile, comme indiqué en figure 9, puis insérez le connecteur du câble USB dans la prise USB de l'ordinateur.

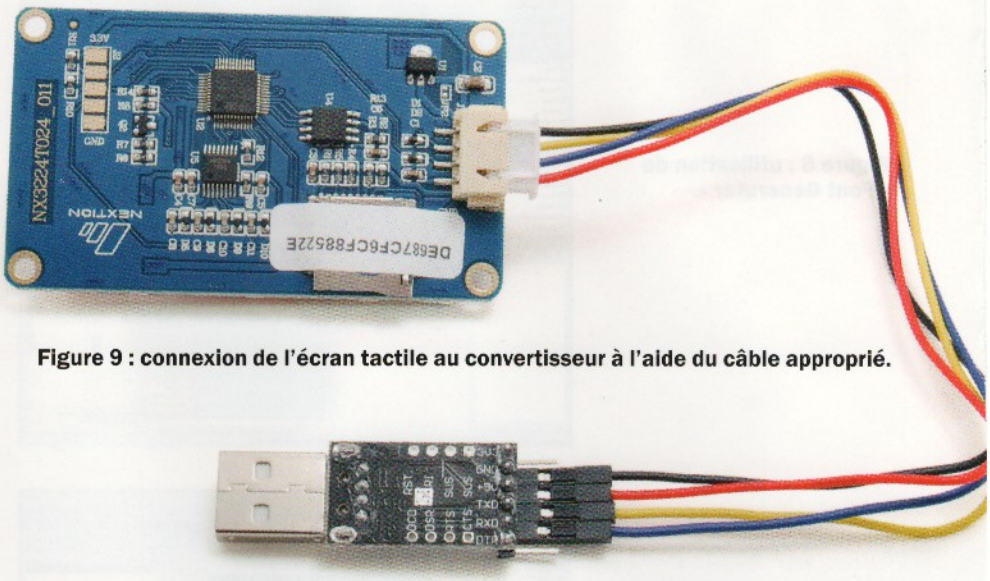


Figure 9 : connexion de l'écran tactile au convertisseur à l'aide du câble approprié.

Le convertisseur USB/TTL utilisé permet d'alimenter l'afficheur tactile car il dispose d'une sortie 5 V.

Veuillez noter que les connexions suivantes sont réalisées :

- sortie +5 V du convertisseur sur la broche 5 V du connecteur de l'écran tactile ;
- GND (masse) du convertisseur à la broche GND du connecteur de l'écran tactile ;
- sortie TX du convertisseur vers la broche RX du connecteur de l'écran tactile ;
- entrée RX du convertisseur vers la broche TX du connecteur de l'écran tactile.

Maintenant, dans le menu, cliquez sur le bouton « **Upload** » (à côté « Debug »). Comme le montre la figure 10, une fenêtre de dialogue relative aux ports de communication s'affiche.

En cliquant sur « **Com Port** », vous ouvrez un menu déroulant dans lequel doit apparaître votre port « COM » utilisé sur votre ordinateur (cela peut être COM3 ou COM7, etc.).

À côté, vous pouvez définir la vitesse de transfert en bauds (Baud Rate), il s'agit de la vitesse de transfert de votre interface graphique de l'éditeur vers la mémoire de l'écran tactile.

Notez que le choix de la commande « **Auto search** » du « Com Port » **détectera automatiquement** le port **COM virtuel** auquel l'écran a été connecté via le convertisseur USB/TTL à votre ordinateur.

Cliquez sur le bouton « **GO** » pour lancer le transfert vers l'afficheur, vous pourrez ainsi visualiser sur l'écran tactile l'interface graphique que vous avez créée, c'est-à-dire la zone de texte et les deux boutons. Déconnectons l'afficheur du PC et passons du côté d'Arduino afin qu'il gère les commandes de l'écran tactile.

Chargeons le code dans Arduino

Nous avons créé pour vous un exemple de sketch disponible en téléchargement. Cela vous permettra d'essayer immédiatement l'afficheur sans devoir configurer tout ce qui a été expliqué jusqu'à présent. Pour utiliser cet exemple, vous devez le téléverser.

Connectez donc la carte Arduino à l'ordinateur à l'aide du câble USB

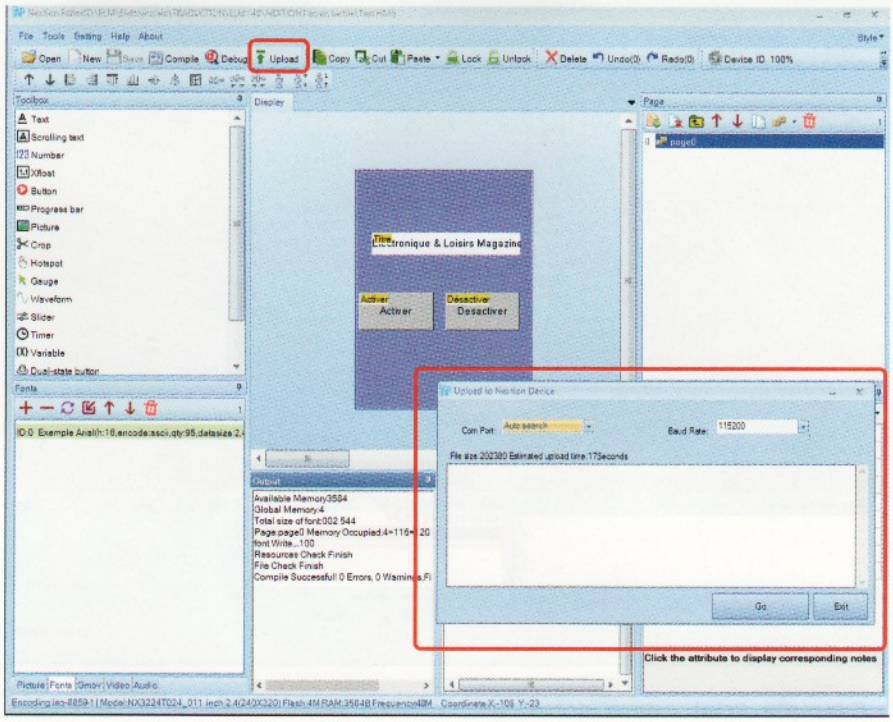


Figure 10 : cliquez sur le bouton « Upload » (cerclé de rouge), la fenêtre de dialogue « Upload to Nextion Device » s'affiche (en bas à droite de l'image encadrée en rouge). Avec la commande « Auto search », le port COM connecté à l'écran sera automatiquement détecté.

Listing 1

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);
  Serial.begin(BAUDRATE_SERIAL);                    /* vitesse de transmission série */
  delay(10);
  Serial.flush();
}

void loop() {
  if(Serial.available()>0) {
    char inChar = (char)Serial.read();
    if (idx< STRING_MAX) {                          /* nettoyage du buffer*/
      inputString[idx]=inChar;
      idx++;
    }
    if  (idx>6)                                     /* s'il y a plus de 6 caractères reçus, exécute l'analyse
    {       stringComplete=true;    }
  }
  parse_menu();                                     /* effectue l'analyse des commandes reçues */
}


void parse_menu() {
  uint8_t index=0;
  if (stringComplete)                               /*chaîne reçue*/
  {
    delay(2);
    index=strlen(inputString);
    if((inputString[0]==0x65)&& (inputString[1]==0x00))
    {
      if(inputString[2]==0x01)
      {       digitalWrite(LED_BUILTIN, HIGH);        }
      else if(inputString[2]==0x03)
      {       digitalWrite(LED_BUILTIN, LOW);     }
    }
    for(idx=0;idx<STRING_MAX;idx++)                 /* nettoyage de la chaîne */
    {     inputString[idx]=0;     }
    idx=0;
    stringComplete=false;
  }
}
```

habituel et ouvrez l'environnement de développement IDE d'Arduino.

Vous devez ouvrir le fichier et à partir du menu « **Outils** » → « **Type de carte** » sélectionnez « **Arduino Mega** ».

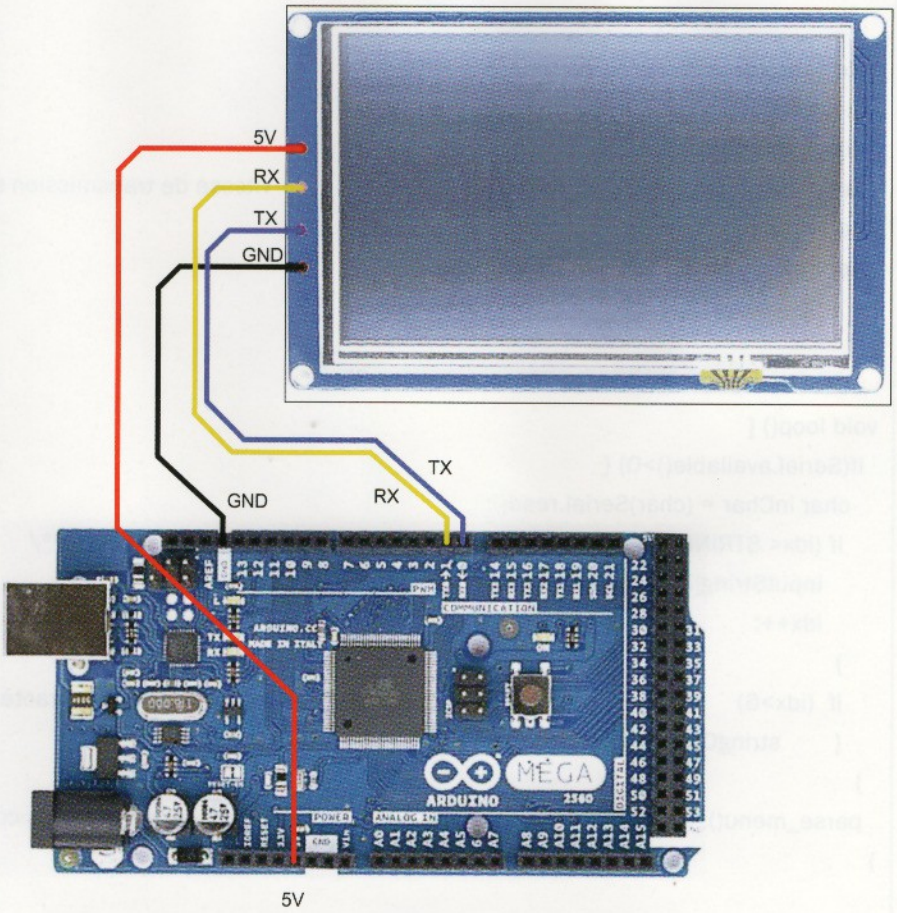
Paramétrez le port « COM » de manière correcte et cliquez sur la flèche ronde « **Téléverser** » (deuxième bouton en haut à gauche en dessous de « Fichier »).

Le code que vous allez télécharger permettra à Arduino de recevoir depuis le port série les commandes envoyées par l'écran tactile lorsque vous appuierez sur l'un des boutons de l'écran.

Si l'analyse de la commande « Activer » est correcte (code hexadécimal 65 00 01 01 FF FF FF), la LED de la carte s'allume, tandis qu'avec la commande « Desactiver » (65 00 03 01 FF FF FF), la LED s'éteint.

Le code source qui effectue ces opérations est visible au Listing 1.

Figure 11 : connexions entre l'afficheur et Arduino. Pour l'alimentation, connectez le câble USB d'Arduino au PC ou alimentez-le avec une source externe.



Connectons Arduino à l'écran tactile

Maintenant que nous avons également expérimenté l'exemple d'application et testé le fonctionnement de l'afficheur, nous pouvons assembler le matériel, ce qui est le but de cet article.

Commençons donc par connecter l'écran à Arduino et alimentons ce dernier, comme illustré dans le schéma de câblage de la figure 11.

Comme vous pouvez le constater, le +5V et la masse (GND) de la carte Arduino (qui dans notre cas est un modèle MEGA) alimentent l'écran tactile, alors que pour la communication, l'UART matériel (hardware) est utilisé.

La ligne TX d'Arduino est reliée à ligne RX de l'écran tactile et la ligne RX de ce dernier est reliée à la ligne TX de l'UART de la carte Arduino MEGA.

À partir de notre exemple, vous pourrez créer des systèmes de commandes à partir d'une interface tactile. Par exemple, vous pouvez ajouter des boutons personnalisés à l'écran tactile, et gérer la pression et le relâchement du bouton (avec dans ce dernier cas une commande différente envoyée), puis charger le nouveau projet dans l'écran.

Chaque fois qu'une commande est ajoutée/modifiée pour l'écran, rappelez-vous que vous devez également modifier le sketch dans Arduino, en ajoutant une analyse des nouvelles commandes reçues afin de faire fonctionner d'autres lignes GPIO (dans ce cas également, vous devrez téléverser de nouveau le nouveau sketch dans Arduino).

Conclusion

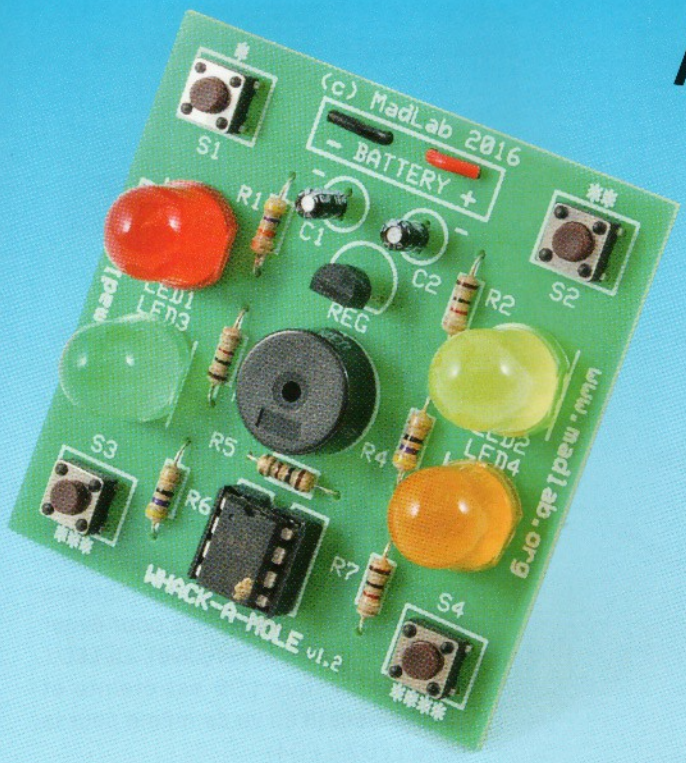
Nous vous rappelons que par rapport aux autres écrans tactiles du marché, l'offre du fabricant Nextion vous permet de mieux couvrir tous les aspects de la conception liés à l'interface utilisateur.

De plus, avec les afficheurs en version « Nextion Enhanced », il est possible de programmer l'écran directement à l'aide d'un code (langage pseudo C propriétaire de Nextion) sans utiliser Arduino et de commander des lignes GPIO disponibles directement sur l'écran tactile (par exemple, il est possible de connecter un relais directement sur l'écran).

Nous vous laissons découvrir les fonctionnalités supplémentaires offertes par ces écrans hautes performances de dernière génération et leur expérimentation. Vous trouverez sur le site web, la documentation utile pour approfondir les aspects que nous ne pouvions pas aborder ici, pour des raisons de place.

Le matériel présenté dans cet article est disponible sur le site futurashop.it. L'écran NEXTION 2,4 pouces est disponible sous la référence NX3224T024 ainsi que l'écran 7 pouces sous la référence NX8048T070. ■

ATTRAPEZ LA TAUPE



Ce montage ludique est un passe-temps agréable qui permet de tester votre vitesse de réaction grâce à des LED et des boutons.

Il existe aujourd'hui dans le commerce des jeux à très haute technologie. Ceux qui désirent des jeux électroniques se tournent maintenant vers des consoles ou des smartphones.

Ces derniers sont devenus une plate-forme universelle avec laquelle il est possible de jouer n'importe où, par exemple dans une salle d'attente ou lors d'un voyage (pas au volant ... bien sûr !!!). Il y a malheureusement beaucoup de personnes qui conduisent avec un smartphone à la main !

C'est la raison pour laquelle nous voulons vous proposer un petit montage qui nous donne l'occasion d'enseigner les bases de l'électronique et d'offrir à ceux qui en ont marre de ne voir qu'Arduino ou RaspberryPi, l'opportunité de continuer à s'entraîner avec un fer à souder.

Le circuit met en œuvre un jeu d'adresse électronique qui sert à tester les réflexes des joueurs. Nous lui avons donné le nom suivant : « Attrapez la taupe », en supposant que l'allumage d'une LED équivaut à voir sortir une taupe de sa cachette et de la saisir avant qu'elle ne disparaisse. En résumé, lorsqu'une LED s'allume, la taupe surgit et vous devez appuyer sur le bouton le plus proche pour « l'attraper ».

Notre passe-temps électronique consiste en fait à mettre en œuvre deux jeux, pour chacun desquels deux modes sont disponibles.

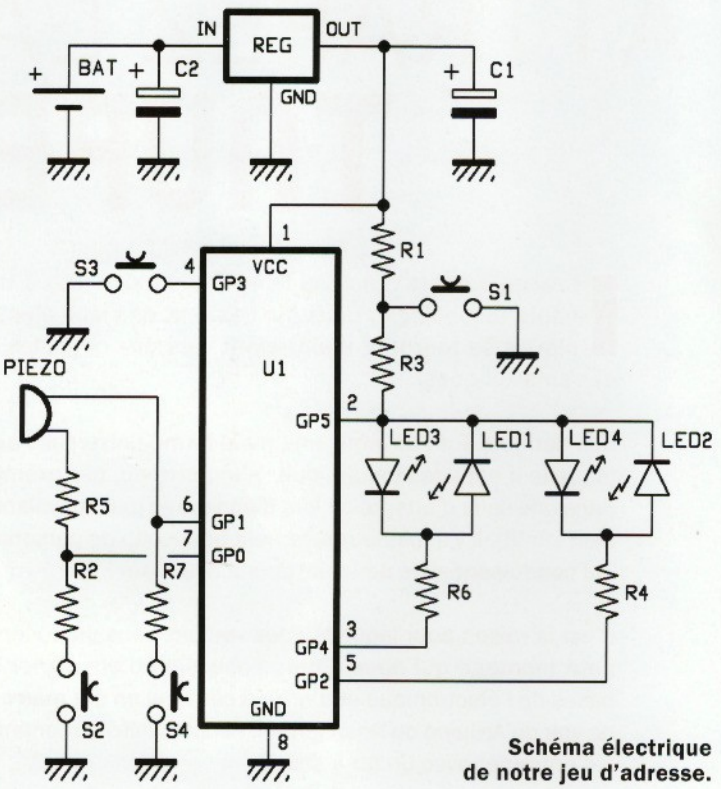


Schéma électrique de notre jeu d'adresse.

Caractéristiques techniques

- Tension d'alimentation : 9 VDC ;
- Consommation : 60 mA ;
- 2 jeux avec 2 niveaux de difficulté ;
- Visualisation à l'aide de LED de 10 mm ;
- Interface à boutons poussoirs.

Schéma électrique

Analysons donc le circuit qui implémente le jeu. Comme le montre le schéma électrique, il est très simple car la gestion de la logique est confiée à un microcontrôleur. Il s'agit d'un microcontrôleur 8 bits dans un boîtier DIP compact de 8 broches. Il est produit par Microchip, il s'agit d'un PIC12F508.

Comme vous pouvez le constater, le microcontrôleur est le cœur du montage et pour que tout fonctionne correctement, très peu de composants externes sont nécessaires. Il y a seulement 4 boutons normalement ouverts, et autant de LED, avec en plus quelques résistances et un buzzer piézoélectrique avec l'électronique intégrée.

Le montage est alimenté avec une pile de 9 V, le régulateur « REG » à 3 broches est un classique 7805 dans sa version basse consommation (78L05). À partir de ce dernier, nous obtenons une tension stabilisée de 5 VDC qui sert principalement à alimenter le microcontrôleur U1.

Notez que pour utiliser au maximum les broches d'entrées/sorties disponibles, nous avons adopté quelques astuces telles que le pilotage des LED de manière « antiparallèle », afin d'en allumer une tout en gardant l'autre éteinte et inversement, en fonction du niveau logique présent sur la broche correspondante. Toutes les LED ont la ligne GP5 en commun, mais les LED1 et LED3 sont contrôlées par la broche GP4 et LED2/LED4 sont pilotées par la ligne GP2.

Notez également que les broches GP2 et GP4 sont configurées en sorties, la broche GP5 est configurée par le programme en entrée et en sortie (ligne bidirectionnelle) selon les besoins. Par exemple, lors de la lecture d'un bouton, elle est configurée en entrée.

Alternativement, elle est configurée en sortie lorsqu'une ou des LED doivent être allumées. L'allumage des LED LED3 et LED4 est obtenu en portant la ligne GP5 à un niveau logique haut et en faisant varier les niveaux logiques sur les lignes GP2 et GP4. Lorsque GP4 est à un niveau bas, LED3 s'allume et si un niveau logique 0 est présent sur la ligne GP2, LED4 s'allume.

Pour allumer les LED LED1 et LED2, la ligne GP5 est maintenue à un niveau logique bas (0) et la ligne GP2 est mise à un niveau logique haut (1) pour allumer LED2. Il en est de même pour la ligne GP4 qui contrôle LED1, elle est portée à un niveau logique haut pour allumer LED1.

Ce mode de commande implique que les LED ne peuvent pas être allumées de façon continue mais de manière pulsée (à haute fréquence pour éviter que notre œil ne le perçoive). La LED1 s'allume lorsque LD3 est éteinte et inversement (il en va de même pour la paire LED2/LED4).

Pour lire l'état du bouton S1, la ligne GP5 est périodiquement configurée en entrée avec sa résistance de tirage (pull-up) activée. La lecture a lieu normalement en maintenant la broche 2 du microcontrôleur à un niveau haut par le biais de la résistance R1. La résistance R3 dérive vers la masse (donc à un niveau bas) la broche GP5 lorsque le bouton est pressé.

Notez que la présence de la résistance R3 permet d'éviter que la ligne GP5 du microcontrôleur ne soit endommagée lorsque le bouton S1 est maintenu appuyé et que la ligne GP5 change de configuration (se comporte comme une sortie).

Cela n'est pas du tout improbable puisque le changement de configuration de la broche 2 (GP5) de la condition de sortie à celle d'entrée a lieu très rapidement, plusieurs fois par seconde, de manière à rendre les impulsions des LED imperceptibles, du fait qu'elles ne peuvent pas être allumées constamment.

S'agissant des boutons, notons que le seul parmi les quatre à disposer d'une ligne d'entrée/sortie dédiée est S3, connecté à la ligne GP3, le programme l'initialise en entrée avec une résistance de pull-up interne.

La souris se met aux jeux vidéo

Pour les passionnés de jeux vidéo, Logitech a créé la souris « G Pro », spécialement conçue pour les jeux vidéo. Elle est née de l'expérience et de la collaboration d'experts du secteur. Pendant 2 ans, Logitech a collaboré avec plus de 50 joueurs professionnels pour atteindre la perfection en termes de forme, de poids et de sensation allant de pair avec les technologies des capteurs sans fil. Le résultat est une souris gaming se distinguant par une précision et des performances inégalées, procurant les outils et la confiance nécessaires pour gagner.

Son capteur « Hero 16K » permet de suivre de 100 à 16 000 mouvements par seconde, avec un IPS supérieur à 400. Ainsi, le pointeur se déplace dans la position exacte du poignet en temps réel. Elle fonctionne sur batterie et a une autonomie de 48 heures en fonctionnement et 60 heures en veille.

La connexion est de type « wireless » (sans fil) et permet aux données d'être transmises en une milliseconde seulement, ce qui élimine les phénomènes de latence.

Grâce au logiciel fourni, il est possible de programmer le fonctionnement des 6 boutons et une mémoire interne permet de sauvegarder les préférences de l'utilisateur. Il est également possible de définir la force à appliquer sur chaque bouton. Le tout pèse 80 grammes et est encapsulé dans une coque de seulement 1 millimètre d'épaisseur.

Les boutons S2 et S4 sont connectés aux lignes GP0 et GP1, partagées avec le buzzer « PIEZO » qui est utilisé par le microcontrôleur pour transmettre les signaux prévus lors des différentes phases et modes du jeu.

Les différents modes du jeu

Ce jeu « attrapez la taupe » contient deux modes, chacun comprenant autant de niveaux de difficulté. La sélection s'effectue à l'aide des boutons, après avoir alimenté le circuit.

Notez qu'à la mise sous tension, le microcontrôleur ordonne au buzzer d'émettre deux courtes notes acoustiques en séquence, puis attend les évènements suivants :

- une pression sur S1 permet d'entrer dans le mode standard. Dans ce cas, une LED s'allumera de manière aléatoire et vous devrez appuyer rapidement sur le bouton le plus proche de la LED allumée. Si vous appuyez sur la mauvaise touche ou si vous n'appuyez pas assez vite (vous mettez trop de temps à réagir), vous avez perdu (ceci est indiqué par le clignotement alterné des LED). Notez que le jeu devient de plus en plus rapide au fil des parties, c'est-à-dire à chaque fois que vous gagnez. Vous avez également moins de temps pour réagir ;
- une pression sur S2 permet d'entrer dans le mode le plus difficile du jeu, dans lequel parfois deux LED s'allument en même temps et vous devez ensuite appuyer simultanément sur deux touches pour confirmer que vous les avez identifiées (dans ce cas aussi, vous devez essayer de gagner le plus de parties possible).

Dans les deux modes du jeu, chaque pression sur une touche est accompagnée de l'émission d'une brève note acoustique par le buzzer. Lorsque vous ne parvenez pas à suivre le jeu et que vous êtes éliminé, la condition est signalée par une sonnerie et par le clignotement alterné des LED.

Avec ce montage, vous pouvez également jouer à un autre jeu appelé « Moley Says », et qui consiste essentiellement à tester la qualité de votre mémoire. Dans ce cas également, vous avez la possibilité de choisir entre un niveau de difficulté standard ou avancé.

Pour jouer à « Moley Says », vous devez effectuer les opérations suivantes :

- appuyez sur le bouton S3 pour démarrer le mode standard. Dès qu'une séquence aléatoire d'illumination des LED se produit, vous devez répéter la séquence en appuyant sur les boutons correspondants dans le même ordre (une fois la séquence des LED terminée, sinon la séquence ne compte pas). Si vous avez appuyé sur les boutons dans le bon ordre, le microcontrôleur fait clignoter deux fois les LED, tandis que si vous faites une erreur ou prenez trop de temps, vous aurez perdu (ceci est indiqué par les LED qui clignotent en alternance). Vous devez être de plus en plus attentif car les séquences deviennent de plus en plus compliquées au fur et à mesure que vous avancez dans les parties. Un clignotement des LED indique que la séquence devient plus longue ;
- appuyez sur le bouton S4 si vous voulez entrer dans le niveau de difficulté avancé. Dans ce mode, parfois deux LED sont allumées en même temps et deux boutons doivent être pressés simultanément lors de la répétition de la séquence.

Pour économiser la pile et donc augmenter sa durée de vie, le microcontrôleur entre en mode veille (sleep-mode) avec

Plan de montage

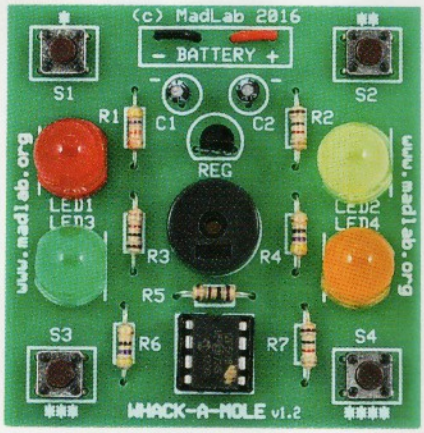
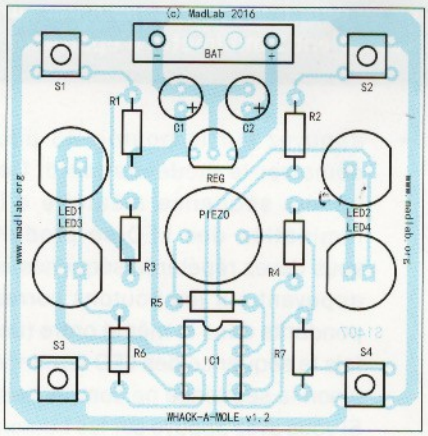


Photo de l'un de nos prototypes.



Plan de câblage des composants.

Liste des composants

R1..... 47 kΩ
R2..... 1 kΩ
R3..... 1 kΩ
R4..... 47 Ω
R5..... 100 Ω
R6..... 47 Ω
R7..... 1 kΩ

C1..... 10 µF/63 V électrolytique
C2..... 10 µF/63 V électrolytique

LED1.. LED 10 mm rouge
LED2.. LED 10 mm verte
LED3.. LED 10 mm jaune
LED4.. LED 10 mm orange

IC1..... PIC12F508-I/P

S1..... microswitch
S2..... microswitch
S3..... microswitch
S4..... microswitch

PIEZO. Buzzer
REG ... 78L05

Divers

Support circuit intégré 2 x 4 broches
Clip pour pile 9 V

une très faible consommation lorsque le circuit n'est pas utilisé ou une minute environ après la fin d'une partie.

Le « réveil » du circuit est obtenu en appuyant sur n'importe quel bouton et est confirmé par l'émission de deux notes acoustiques et par le clignotement alterné des LED situées de chaque côté. La même chose se produit lorsque la pile est connectée.

Réalisation pratique

Après les explications de fonctionnement du montage, nous devons passer à la construction de notre jeu électronique. Le circuit imprimé comporte une seule face qui reçoit tous les composants nécessaires. Vous pouvez le fabriquer facilement par photogravure. Le typon du circuit imprimé à l'échelle 1 est téléchargeable dans le sommaire détaillé de la revue sur notre site web.

Notez que tous les composants sont du type traversant (traditionnels), le montage est donc facilement reproductible même pour ceux qui ne possèdent pas les compétences manuelles et l'équipement nécessaire pour construire des circuits avec des composants CMS (montage en surface).

Une fois le circuit imprimé gravé et percé, comme d'habitude commencez par souder les composants ayant un profil bas (les résistances, le support du microcontrôleur).

Continuez avec les 2 condensateurs en respectant leur polarité (la patte la plus longue correspond au +). Ensuite, soudez les boutons, le régulateur 78L05 en respectant son orientation (son méplat se situe vers le buzzer). Terminez le montage en soudant le buzzer, les 4 LED en respectant leur polarité.

Assurez-vous qu'elle soit correcte, car si elle était inversée, les LED ne s'allumeraient pas correctement pendant le fonctionnement. Pour l'alimentation, soudez les fils rouge (positif) et noir (négatif) aux point « BAT » d'un clip pour piles 9 V. Enfin, insérez le microcontrôleur dans son support, son détrompeur en forme de « U » doit se situer vers le buzzer.

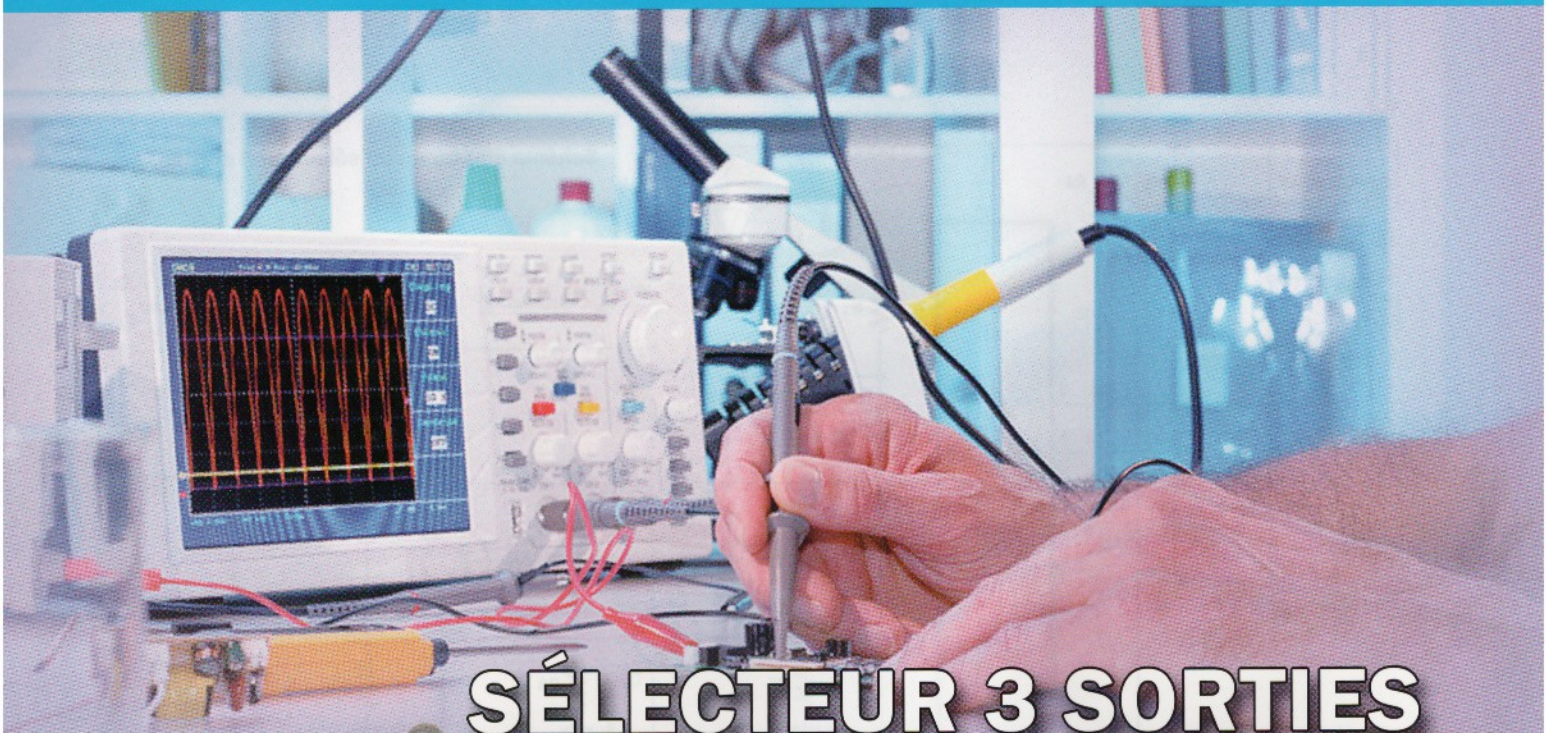
Vérifiez l'implantation et l'orientation des composants polarisés en vous reportant au plan de montage présenté dans ces pages. Vous pouvez ensuite utiliser votre circuit, car il ne nécessite pas d'étalonnage.

Rappelez-vous que pendant le jeu, vous devez faire attention à ne pas toucher le dos du circuit imprimé, ce qui serait une manière naturelle pour le tenir afin de mieux pouvoir appuyer rapidement sur les boutons. Cela ne devrait pas être le cas car cela pourrait affecter le fonctionnement du montage.

Vous pouvez isoler le dos du circuit imprimé avec une feuille de plastique ou de carton collé sur les bords avec du ruban adhésif, ou bien d'une manière plus élégante, mettre en boîter le circuit imprimé en pensant à faire sortir les boutons.

N'oubliez pas que, malgré les stratégies d'économie d'énergie mises en œuvre dans le programme, il est toutefois conseillé de retirer la pile lors des longues périodes d'inactivité, car dans tous les cas, le circuit consomme un peu de courant en veille ce qui, à long terme, déchargerait la pile. Vous pouvez insérer un interrupteur M/A en série avec le positif de l'alimentation provenant de la pile.

Ce jeu est disponible sous la forme d'un kit avec le microcontrôleur programmé sur le site www.futurashop.it. Sa référence est : 8220-MLP111IT



SÉLECTEUR 3 SORTIES
Pour l'alimentation multi-tensions

de Boris Landoni

Nous vous présentons dans cet article une interface pour l'alimentation multi-tension décrite dans le numéro 148. Il s'agit d'un sélecteur qui permet de régler la tension de sortie à l'aide de boutons. L'utilisation d'un microcontrôleur de type Atmel sera l'occasion d'une étude approfondie de l'ATTiny.

En conclusion de l'article sur l'alimentation multi tensions de laboratoire décrite dans le numéro 148, si vous vous en souvenez, nous avions évoqué la possibilité d'ajuster la tension de sortie sélectionnable « VOUT1 » à l'aide d'une carte de commande au lieu d'utiliser le commutateur rotatif.

Cette carte est assimilable à un commutateur électronique qui permet d'activer des sorties de type « collecteur ouvert » à l'aide de deux boutons. L'un permet d'augmenter la tension et l'autre de la baisser. Cette carte de commande est raccordée à l'alimentation multi tensions au moyen de connecteurs appropriés. Bien que simple et réalisable à l'aide de circuits logiques, le circuit est basé sur un microcontrôleur de marque Atmel, il s'agit de l'ATTiny13A. Vous pouvez télécharger sa documentation technique à l'adresse suivante : http://ww1.microchip.com/downloads/en/DeviceDoc/doc8126.pdf.

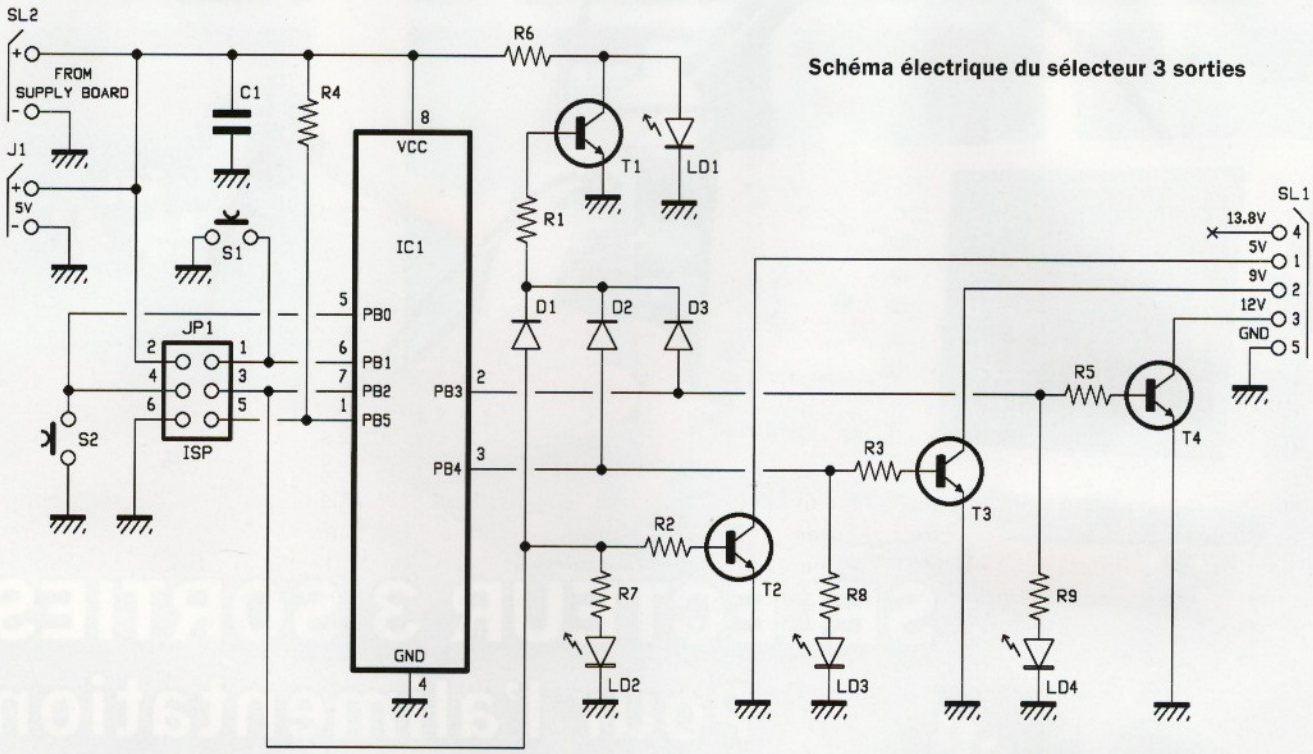


Schéma électrique du sélecteur 3 sorties

Ce microcontrôleur dispose de 8 broches, dont 3 sont réservées pour des fonctions de base telles que : VCC (alimentation), GND (masse) et RESET (réinitialisation).

L'utilisation du microcontrôleur est destinée à nous donner l'opportunité d'enseigner l'utilisation de ce type de composant grâce à la mise en œuvre d'une application pratique. Examinons d'abord le schéma électrique.

Schéma électrique

D'un point de vue matériel, notons tout d'abord l'astuce utilisée pour gérer les entrées et les sorties du circuit à partir de l'ATTiny. En effet, nous ne pouvons utiliser que 5 broches et, comme deux d'entre elles sont réservées pour lire les boutons, les sorties disponibles sont au nombre de trois. Elles servent à régler la tension de sortie de l'alimentation.

Toutefois, les tensions de sortie pouvant être configurées au niveau de l'alimentation sont au nombre de quatre. Pour obtenir les trois premières tensions, nous fermons les trois lignes de commande une à la fois et, pour la quatrième tension, nous laissons toutes les lignes « ouvertes » (déconnectées).

De cette manière, la résistance connectée en contre réaction au régulateur de l'alimentation permet de fixer la sortie à 3,3 V, ce qui équivaut à la tension de référence.

Notre carte permet de remplacer le commutateur mécanique et d'obtenir en sortie 4 tensions différentes : 3,3 V ; 5 V ; 9 V et 12 V. Cela, simplement à l'aide de deux boutons utilisés pour régler séquentiellement la tension vers le bas ou vers le haut.

Le réglage est effectué en contrôlant à l'aide des sorties du microcontrôleur (c'est-à-dire les entrées/sorties PB2, PB3 et PB4 configurées en sortie) trois transistors NPN dénommés T2, T3 et T4.

Sur les bases de ces derniers, des LED sont connectées avec en série, pour chacune, une résistance de limitation de courant. Les LED s'allument lorsque le transistor correspondant passe en saturation et que la tension sélectionnée est présente en sortie de l'alimentation.

Comme nous n'avons pas de broche disponible pour régler le 3,3 V, pour détecter la condition du 3,3 V, une porte logique de type NOR est réalisée à l'aide de trois diodes et d'un transistor NPN (T1).

Le fonctionnement de cette partie du circuit est tel que, lorsque toutes les sorties du microcontrôleur sont à un niveau logique 0 (donc T2, T3 et T4 sont bloqués), T1 n'est pas conducteur et la LED LD1 s'allume (car elle est alimentée par la ligne + 5 V via la résistance R6). Lorsque l'une des lignes PB2, PB3 ou PB4 met le transistor T1 en saturation, ce dernier est dans un état passant et dérive le courant qui normalement irait dans la LED LD1 (qui reste éteinte).

Les transistors T2, T3 et T4 ne sont saturés qu'un à la fois et servent donc d'interrupteurs statiques en remplaçant les contacts de l'interrupteur rotatif présent dans l'alimentation. Pour régler la tension de sortie à 3,3 V, les trois transistors se trouvent tous dans un état bloqué.

Caractéristiques techniques :

- **Alimentation : 5 VDC ;**
- **Gestion par microcontrôleur ;**
- **Sorties à collecteur ouvert ;**
- **Signalisation par LED.**

Examinons maintenant les entrées, où se trouvent les deux boutons S1 et S2 qui sont reliés aux broches 5 (PB0) et 6 (PB1). Ils permettent au microcontrôleur d'effectuer les commandes d'incrémentation de la tension de sortie (de 3,3 V à 12 V) et de décrémentation (de 12 V à 3,3 V) de la tension de sortie de l'alimentation.

L'alimentation de la carte de commande est réalisée au moyen du connecteur SL2 qui provient de l'alimentation (voir le schéma électrique du numéro 148). En parallèle avec le connecteur SL2 se trouve le connecteur J1, à partir duquel la tension de 5 V peut être prélevée pour alimenter, par exemple, un voltmètre électronique qui affichera la tension de sortie de l'alimentation.

Le condensateur de filtrage C1 est placé sur la ligne d'alimentation provenant de SL2, il permet d'atténuer d'éventuelles perturbations présentes sur la tension secteur. Le connecteur SL1 relie la carte de commande au circuit de sélection de la tension de sortie de la partie réglable de l'alimentation de laboratoire publiée dans le précédent numéro. Pour compléter la description du circuit, vous pouvez voir sur le schéma électrique le connecteur IDC (JP1) qui permet la programmation du microcontrôleur à l'aide d'un programmateur compatible ISP.

La résistance de tirage R4 présente sur la broche de réinitialisation (PB5), permet de maintenir cette dernière à un niveau haut. Lorsque la ligne PB5 est amenée à un niveau logique 0, le microcontrôleur entre en mode programmation.

Notez que la ligne PB2, en utilisation normale, est destinée à la commande du transistor T2. Lors de la programmation du microcontrôleur, elle fonctionne comme une broche d'horloge (SCK) afin d'établir la communication série avec le programmateur.

Le programme ou firmware

Jusqu'à présent, nous avons étudié la partie matérielle, qui peut être considérée comme la partie la plus facile du projet. Nous devons maintenant programmer notre microcontrôleur afin qu'il exécute les fonctions décrites.

Contrairement aux cartes Arduino, nous avons ici un microcontrôleur avec seulement 1 ko de mémoire flash pour le programme.

La tâche n'est pas facile et nous devons utiliser quelques astuces pour réduire et optimiser le code. Ce dernier est visible dans le Listing 1. Le code qui gère notre carte de commande est écrit en C, ce langage constitue l'environnement de base des systèmes de développement Arduino et de tous les microcontrôleurs Atmel AVR en général.

Nous avons essayé de le commenter de manière détaillée pour chaque section afin de faciliter la compréhension du lecteur. Examinons la première ligne du programme : **#include <avr/io.h>**

Cette instruction ouvre la première porte importante de l'environnement AVR, c'est-à-dire la configuration des entrées/sorties qui permet de définir les registres d'entrées/sorties correspondants.

Ensuite, il existe d'autres définitions (define) dans la routine principale (main). Après l'instruction « cli() », se trouvent celles liées aux registres PORTB et DDRB.

Vous observez la présence d'un « égal », c'est ce que l'on appelle l'affectation absolue, c'est-à-dire que les bits qui ne sont pas explicitement affectés à 1, sont affectés à 0.

Pour l'affectation, nous observons une fonction « _BV », qui est une macro incluse dans le compilateur AVR et qui signifie « Bit Value ». Entre parenthèses se trouve, par exemple, « PORTB0 ». Il s'agit d'une des constantes définies implicitement dans le fichier inclus dans la première ligne du programme.

Au « PORTB0 » correspond l'expression binaire 00000001, immédiatement après se trouve le signe « | » qui signifie OR (pout bit OU) et l'instruction « _BV(PORTB1) ». Il s'agit d'une somme logique, ensuite on communique à la variable que l'on affecte 1 sur le bit 2.

L'instruction « PORTB = (_BV(PORTB0) | _BV(PORTB1) | _BV(PORTB2) | _BV(PORTB3) | _BV(PORTB4)); » signifie qu'au registre PORTB est affectée la valeur binaire « 00011111 ».

L'instruction suivante concerne le registre « DDRB » et est analogue à la première, on attribue au registre la valeur « 00011100 ». L'argument correspond à une affectation (absolue) à 0 ou 1 des bits d'une variable (bit value).

Examinons deux autres types d'assignations. La première est celle représentée ci-après :

```
GIMSK |= (_BV(INT0) | _BV(PCIE));
```

GIMSK est le registre général des interruptions.

Ce registre configure le comportement des interruptions du microcontrôleur. Le bit INT0 est utilisé pour les interruptions externes, tandis le bit PCIE active le vecteur d'interruption sur les broches.

Il s'agit d'une affectation de type OR (OU), en ce sens que l'opération OR s'effectue bit par bit entre le registre de gauche et la variable créée à droite. En résumé, les bits représentant les variables « INT0 » et « PCIE » sont configurés à 1.

La seconde assignation est complémentaire à la précédente et est représentée par la ligne :

```
TCCR0B &= ~(_BV(CS00) | _BV(CS01)
| _BV(CS02));
```

qui est utilisée pour configurer à 0 les bits relatifs à CS00, CS01, CS02.

À ce stade, nous n'expliquerons pas la ligne qui définit le registre TCCR0B. Il suffit de dire que cela concerne la configuration du Timer0.

```
GIMSK |= (_BV(INT0) | _BV(PCIE));
PCMSK |= _BV(PCINT0);
MCUCR |= _BV(ISC00);
```

Ces trois lignes doivent être associées à la deuxième ligne de programme « **#include <avr/interrupt.h>** ». Cette instruction, si elle n'est pas bien gérée, peut causer de graves problèmes, car elle gère les interruptions du programme.

Cela est valable non seulement dans notre cas, mais également dans l'environnement Arduino.

Listing 1

```
/*************************************************************************** */
/* Routine pour créer un sélecteur de type collecteur ouvert à 2 boutons    */
/* Utilise un ATTiny 13A                                                    */
/* Version 1.2 du 08-08-2019                                                */
/* Sortie 1 sur la broche 7 (PB2)
/* Sortie 2 sur la broche 3 (PB4) */
/* Sortie 3 sur la broche 2 (PB3) */

/*************************************************************************** */
#include <avr/io.h>
#include <avr/interrupt.h>
#define HIGH 1
#define LOW 0
#define EXIT1 2 //PB2
#define EXIT2 4 //PB2
#define EXIT3 3 //PB2
unsigned char pari1,pari2,posizione;

void digitalWrite(char Pin, char stato);
void myDelay(int prescaler, char numpulse);

int main(){
cli();                  // désactive toutes les interruptions
                        // configure les broches 5(PB0) et 6(PB1) du PORTB en entrées ; configure les broches 7(PB2),
2(PB3) et 3(PB4) en sorties.

PORTB = (_BV(PORTB0) | _BV(PORTB1) | _BV(PORTB2) | _BV(PORTB3) | _BV(PORTB4));
DDRB = (_BV(DDB2) | _BV(DDB3) | _BV(DDB4));                    //configure les entrées en pull-up
TCCR0B &= ~(_BV(CS00) | _BV(CS01) | _BV(CS02));                 //désactive le timer Timer0

GIMSK |= (_BV(INT0) | _BV(PCIE));                   // Active les interruptions INT0 et PCINT
PCMSK |= _BV(PCINT0);                               // on considère PCINT0
MCUCR |= _BV(ISC00);                                // INT0 se comporte comme PCINT0

pari1 = 1;

pari2 = 1;

posizione = 0;
digitalWrite(EXIT1,LOW);
digitalWrite(EXIT2,LOW);
digitalWrite(EXIT3,LOW);
myDelay(1024,200);
sei();                  //Active toutes les interruptions

while(1);
}

void digitalWrite(char Pin, char stato) {
        if (stato) {
                switch (Pin) {
                        case 0 : if (!(PINB & _BV(PINB0))) PINB |= _BV(PINB0); break;   //configure la broche B0 à 1
                        case 1 : if (!(PINB & _BV(PINB1))) PINB |= _BV(PINB1); break;   // configure la broche B1 à 1
                        case 2 : if (!(PINB & _BV(PINB2))) PINB |= _BV(PINB2); break;   // configure la broche B2 à 1
                        case 3 : if (!(PINB & _BV(PINB3))) PINB |= _BV(PINB3); break;   // configure la broche B3 à 1
                        case 4 : if (!(PINB & _BV(PINB4))) PINB |= _BV(PINB4); break;   // configure la broche B4 à 1
                }
        }
        else {
                switch (Pin) {
                        case 0 : if ((PINB & _BV(PINB0))) PINB |= _BV(PINB0); break;   // configure la broche B0 à 0
                        case 1 : if ((PINB & _BV(PINB1))) PINB |= _BV(PINB1); break;   // configure la broche B1 à 0
```

```
                        case 2 : if ((PINB & _BV(PINB2))) PINB |= _BV(PINB2); break;  // configure la broche B2 à 0
                        case 3 : if ((PINB & _BV(PINB3))) PINB |= _BV(PINB3); break;  // configure la broche B3 à 0
                        case 4 : if ((PINB & _BV(PINB4))) PINB |= _BV(PINB4); break;  // configure la broche B4 à 0
                }
        }
}

void myDelay(int prescaler, char numpulse) {
        TCNT0 = 0;                      // réinitialise le compteur
        // configuration du prescaler
        switch (prescaler) {
                case 1 : TCCR0B |= _BV(CS00); break;
                case 8 : TCCR0B |= _BV(CS01); break;
                case 64 : TCCR0B |= (_BV(CS01) | _BV(CS00)); break;
                case 256 : TCCR0B |= _BV(CS02); break;
                case 1024 : TCCR0B |= (_BV(CS02) | _BV(CS00)); break;
        }
        while (TCNT0 < numpulse);                                       // sort lorsque TCNT0 >= numpulse
        TCCR0B &= ~(_BV(CS00) | _BV(CS01) | _BV(CS02));                 //désactive le timer0
}

ISR(PCINT0_vect) {   // incrémentation
        GIMSK &= ~(_BV(INT0) | _BV(PCIE));                              // désactive les interruptions INT0 et PCINT0
        if (pari1) {
                pari1 = 0;
        switch (posizione) {
                case 0:
                digitalWrite(EXIT1,HIGH);
                digitalWrite(EXIT2,LOW);
                digitalWrite(EXIT3,LOW);
                posizione++;
                break;
                case 1:
                digitalWrite(EXIT1,LOW);
                digitalWrite(EXIT2,HIGH);
                digitalWrite(EXIT3,LOW);
                posizione++;
                break;
                case 2:
                digitalWrite(EXIT1,LOW);
                digitalWrite(EXIT2,LOW);
                digitalWrite(EXIT3,HIGH);
                posizione++;
                break;
                }
        }
        else pari1 = 1;
        myDelay(1024,250);
        GIMSK |= (_BV(INT0) | _BV(PCIE));               // active les interruptions INT0 et PCINT0
}

ISR(INT0_vect) { //Décrémentation
        GIMSK &= ~(_BV(INT0) | _BV(PCIE));              // désactive les interruptions INT0 et PCINT0
        if (pari2) {
                pari2 = 0;
        switch (posizione) {
                case 1:
                digitalWrite(EXIT1,LOW);
                digitalWrite(EXIT2,LOW);
                digitalWrite(EXIT3,LOW);
                posizione--;
                break;
                case 2:
                digitalWrite(EXIT1,HIGH);
```

```
                digitalWrite(EXIT2,LOW);
                digitalWrite(EXIT3,LOW);
                posizione--;
                break;
                case 3:
                digitalWrite(EXIT1,LOW);
                digitalWrite(EXIT2,HIGH);
                digitalWrite(EXIT3,LOW);
                posizione--;
                break;
                }
        }
        else pari2 = 1;
        myDelay(1024,250);
        GIMSK |= (_BV(INT0) | _BV(PCIE));    // active les interruptions INT0 et PCINT0
}
```

Tableau 1 - Configuration de INT0

ISC01	ISC00	Description
0	0	un niveau bas sur INT0 génère une interruption
0	1	tout changement de niveau sur INT0 génère une interruption
1	0	un front descendant sur INT0 génère une interruption
1	1	un front montant sur INT0 génère une interruption

Mais de quel type d'interruption s'agit-il ?

Tout simplement, cette interruption stoppe le déroulement normal du programme pour exécuter un morceau de code (routine d'interruption) puis reprendre là où le programme s'est arrêté.

Pour cela, le microcontrôleur enregistre le pointeur de l'instruction qu'il doit exécuter, ensuite il sauvegarde les variables en cours et effectue un saut pour exécuter le code écrit dans la routine d'interruption.

Les interruptions peuvent être masquables ou non. Si elles le sont, elles peuvent être activées et désactivées pendant le déroulement du programme, ou il est possible d'imposer une condition pour qu'elles soient ignorées, sinon dans la plupart des cas elles sont quand même exécutées.

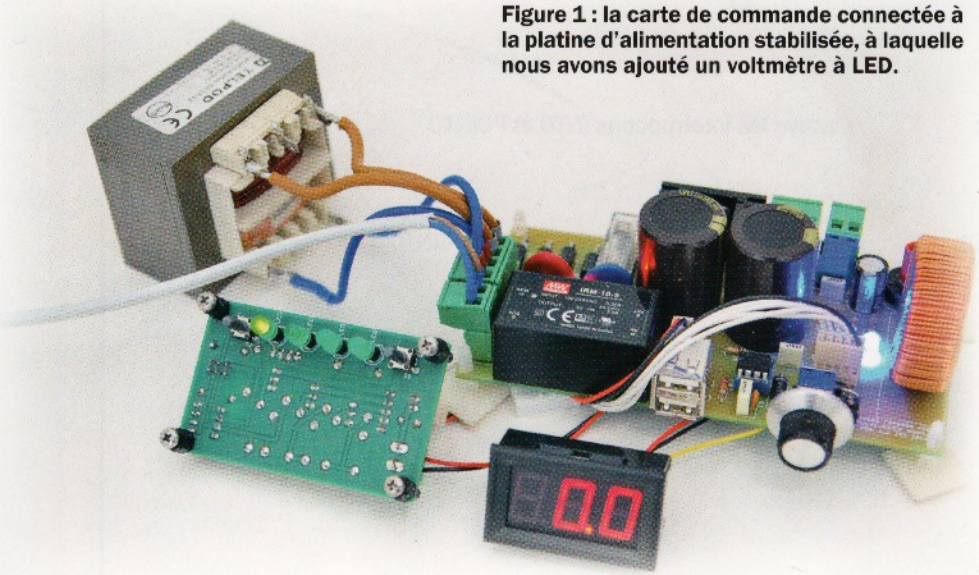


Figure 1 : la carte de commande connectée à la platine d'alimentation stabilisée, à laquelle nous avons ajouté un voltmètre à LED.

Celles que nous utiliserons seront toujours masquables.

Les interruptions sont constituées d'un vecteur qui se nomme « vecteur d'interruption ». Il existe un ordre de priorité qui, dans les microcontrôleurs de la classe AVR Tiny et Mega, ne peut pas être modifié. Cet ordre est défini par le fabricant.

Plus l'indice est bas, plus la priorité est élevée. La valeur « 0 » est attribuée à la réinitialisation et immédiatement après dans le cas de l'ATTiny13A, il existe « INT0 » qui sert également à « réveiller » le microcontrôleur qui a été mis dans un état de repos pour consommer moins. Par exemple, lorsqu'il est utilisé dans des applications alimentées par batterie.

Pour de plus amples informations, vous pouvez consulter la documentation technique de l'ATmega, où se trouve la table des interruptions.

Au niveau du code, une fonction spécifique de l'environnement AVR est appelée, elle se nomme ISR (Interrupt Service Routine). Elle se compose d'un paramètre correspondant au vecteur d'interruption que vous souhaitez gérer.

Par exemple, pour gérer l'interruption INT0, il suffit d'écrire : ISR(INT0_vect). Notez que le code est écrit entre parenthèses.

Nous devons nous rappeler une chose importante : les ISR ne peuvent pas elles-mêmes être interrompues par une autre interruption.

Ainsi, tant que la routine n'est pas terminée, une autre interruption ne peut pas être exécutée. À ce stade, nous comprenons pourquoi il peut y avoir des problèmes graves si une routine d'interruption « tourne en boucle », seule une réinitialisation du système peut l'arrêter.

Examinons maintenant la gestion des boutons à l'aide des interruptions.

Les interruptions disponibles pour les entrées digitales de l'ATTiny13A sont au nombre de 2 : INT0 et PCINT. La documentation technique montre que INT0

Plan de montage du sélecteur pour alimentation

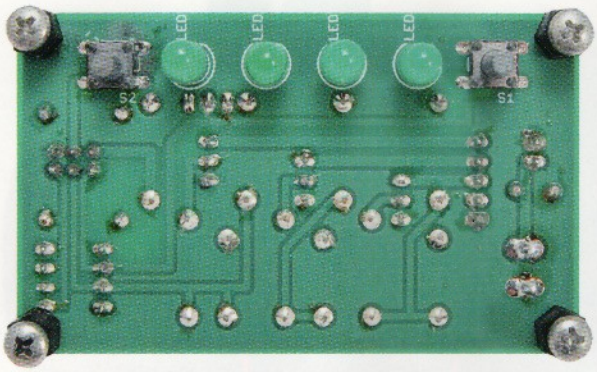


Photo de l'un de nos prototypes vu côté soudures.

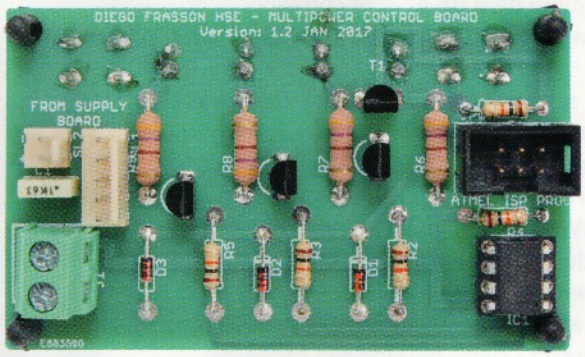
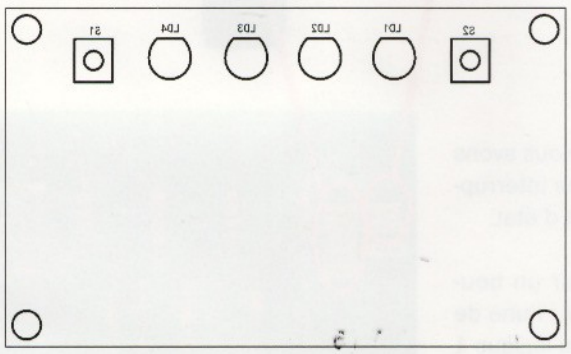
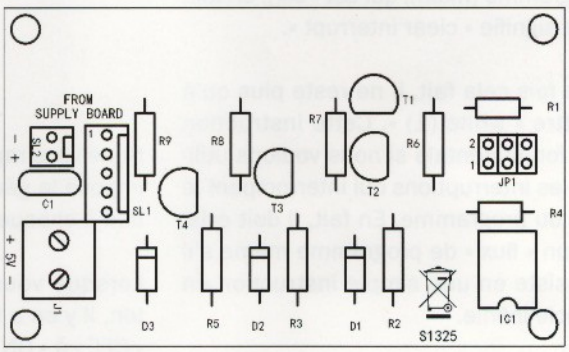


Photo de l'un de nos prototypes vu côté composants.



Plan de câblage des composants côté soudures.



Plan de câblage des composants.

Liste des composants

R1..... 10 kΩ
R2..... 10 kΩ
R3..... 10 kΩ
R4..... 10 kΩ
R5..... 10 kΩ
R6..... 1 kΩ
R7..... 1 kΩ
R8..... 1 kΩ
R9..... 1 kΩ

C1..... 100 nF céramique
D1..... 1N4148
D2..... 1N4148
D3..... 1N4148
T1 ..... 2N3904
T2 ..... 2N3904
T3 ..... 2N3904
T4 ..... 2N3904
IC1..... ATTiny13A-20PU
LD1.... LED 5 mm verte
LD2.... LED 5 mm verte
LD3.... LED 5 mm verte
LD4.... LED 5 mm verte

S1..... microswitch
S2..... microswitch

Divers

Connecteur HE6 pour circuit imprimé
Bornier 2 pôles
Connecteur fil-câble 2 pôles
Connecteur fil-câble 5 pôles
Support circuit intégré 2 x 4 broches

est configurable comme mentionné dans le tableau 1.

Ceci, en raison du fait qu'il existe une résistance de pull-up, en fait, le cas avec INT0 à un niveau haut n'est pas envisagé. Par contre, PCINT répond toujours à chaque changement de niveau, en générant une interruption. Elle ne peut pas être configurée.

Donc, nous devons définir la séquence suivante dans le programme :

- activer l'interruption sur INT0 et PCINT0 ;
- utiliser la broche à laquelle PCINT0 est associée pour activer l'interruption ;
- configurer à 1 le bit ISC00 du registre MCUCR (ISC01 n'a pas besoin d'être configuré car il est à 0 par défaut).

Nous attirons maintenant votre attention sur la première ligne. Pour pouvoir les utiliser, nous devons activer les interruptions INT0 et PCINT0.

Notez également que l'interruption est appelée PCINT0 (comme vous pouvez le voir dans le tableau des vecteurs d'interruption) et vous voyez à la place les broches PCINT0, PCINT1, PCINT2, PCINT3, PCINT4, PCINT5.

Cependant, il n'y a qu'une seule interruption, elle doit être combinée avec une broche, parmi celles disponibles dans le microcontrôleur. Dans notre cas, nous utilisons la broche 5 (PB0), qui dispose de PCINT0.

Si vous consultez la documentation technique de l'ATMega328P, vous constaterez que les interruptions PCINT sont au nombre de 3 (0, 1, 2) et correspondent chacune à un port d'E/S (B, C, D).

Le reste du programme, compris dans la routine principale, initialise certaines variables en configurant les broches de sortie à 0 grâce à l'instruction « digitalWrite » (de même style que pour Arduino). Ensuite, le programme appelle une fonction de temporisation « delay » et active enfin l'interruption avec l'instruction « sei() » qui signifie « configurer » l'interruption (set interruption) et qui est à l'opposé de celle définie au début du programme (main), qui est « cli(), en fait, cela signifie « clear interrupt ».

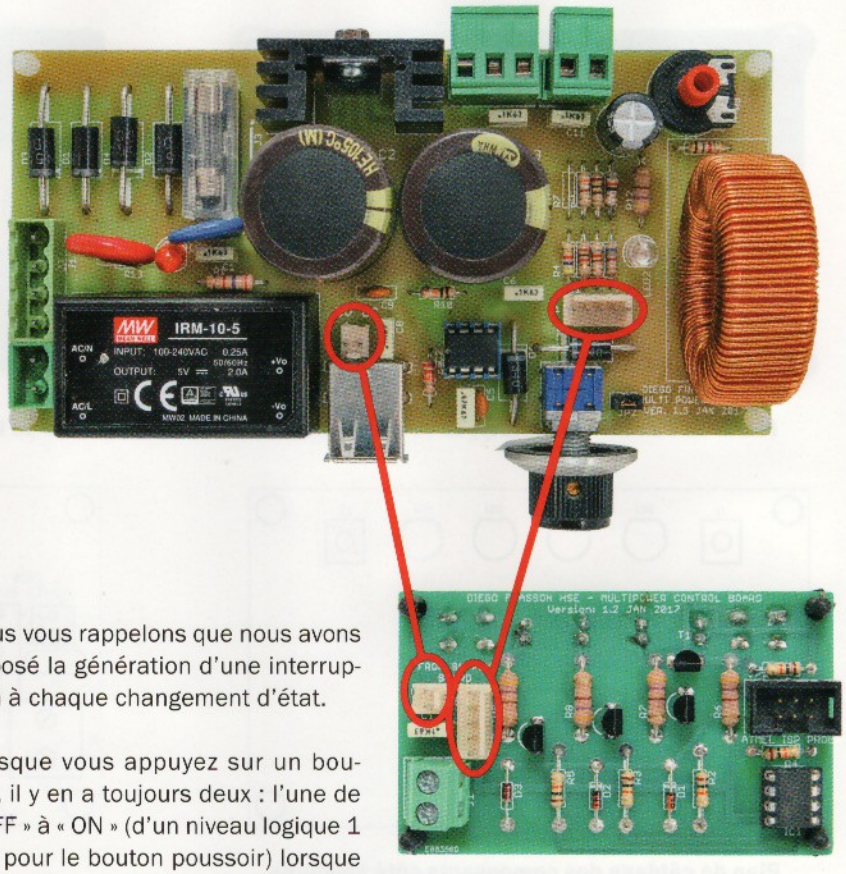
Une fois cela fait, il ne reste plus qu'à mettre « while (1) ». Cette instruction est fondamentale si nous voulons utiliser les interruptions qui interrompent le flux du programme. En fait, il doit exister un « flux » de programme même s'il consiste en une simple instruction en boucle infinie.

Nous allons décrire maintenant les routines d'interruption qui constituent le véritable cœur du programme. Ces routines peuvent être considérées comme une machine à états simples où l'état suivant dépend de l'état précédent. Nous connaissons l'état initial car nous l'imposons. Il correspond à une configuration à un niveau logique bas de toutes les sorties de contrôle du réglage de la tension (PB2, PB3, PB4).

Ensuite, en appuyant sur le bouton associé à PCINT0, la variable est incrémentée d'une unité, ce qui porte la première sortie du microcontrôleur (PB2) à un niveau logique haut. La LED LD1 s'éteint et LD2 s'allume (car le niveau logique 1 sur PB2 la polarise) et ainsi de suite jusqu'à la quatrième LED (LD4). Les états sont donc au nombre de 4. Le bouton associé à INT0 est complémentaire et sert à décrémenter la variable.

Dans les routines, vous trouverez également deux variables appelées « pari1 » et « pari2 ». Le délai défini (environ 0,3 seconde) est utilisé pour produire une impulsion à chaque pression sur une touche et éviter ainsi le rebond des contacts (appelé anti-rebond) dû à l'usure.

Figure 2 : la carte de commande est connectée à l'alimentation stabilisée via les deux connecteurs de type S.I.L. nommés SL1 et SL2.



Nous vous rappelons que nous avons imposé la génération d'une interruption à chaque changement d'état.

Lorsque vous appuyez sur un bouton, il y en a toujours deux : l'une de « OFF » à « ON » (d'un niveau logique 1 à 0 pour le bouton poussoir) lorsque vous appuyez, et l'autre lorsque vous relâchez le bouton de « ON » à « OFF » (d'un niveau logique 0 à 1).

Nous allons maintenant examiner la fonction « digitalWrite ». Celle-ci permet de « transmettre » un état logique sur la broche que vous voulez « contrôler ».

Au niveau des entrées de l'AVR, il est toujours possible de lire l'état via le registre PINB, que la broche soit configurée en entrée ou qu'elle soit configurée en sortie.

Nous en avons besoin car, comme nous l'avons déjà mentionné, si nous écrivons sur un seul bit du registre PINB, les sorties relatives sont déjà dans l'état souhaité, c'est une erreur de le refaire car cela les ramènerait dans un état non désiré.

C'est donc ici que la fonction discrimine la broche puis lit son état et si celle-ci se trouve dans un état opposé à celui imposé (paramétré), elle écrit 1 et la fait commuter, sinon elle ignore la commande puisque l'état est déjà conforme aux attentes.

Réalisation pratique

Passons maintenant à la construction du circuit. Pour cela vous devez réaliser le circuit imprimé double face. Nous mettons à disposition le fichier Eagle (.brd) que vous pourrez télécharger dans le sommaire détaillé de la revue. Vous pouvez voir le circuit imprimé dans Eagle en figure 3. Les pistes de couleur bleu représentent la couche inférieure (bottom) et celles en rouge les pistes de la couche supérieure.

En cliquant sur le bouton « Manufacturing » en haut à droite, vous pouvez voir un aperçu de la face supérieure du circuit une fois fabriqué (voir la figure 4). En sélectionnant « Bottom Side » dans le coin inférieur gauche, vous apercevrez un aperçu de la couche inférieure.

Nous allons maintenant générer les fichiers Gerber pour une fabrication professionnelle. Dans la fenêtre « Manufacturing », cliquez sur le bouton « CAM » en bas à droite afin de générer les fichiers Gerber. Une fenêtre s'ouvre dans laquelle vous pouvez visualiser l'ensemble des fichiers Gerber.

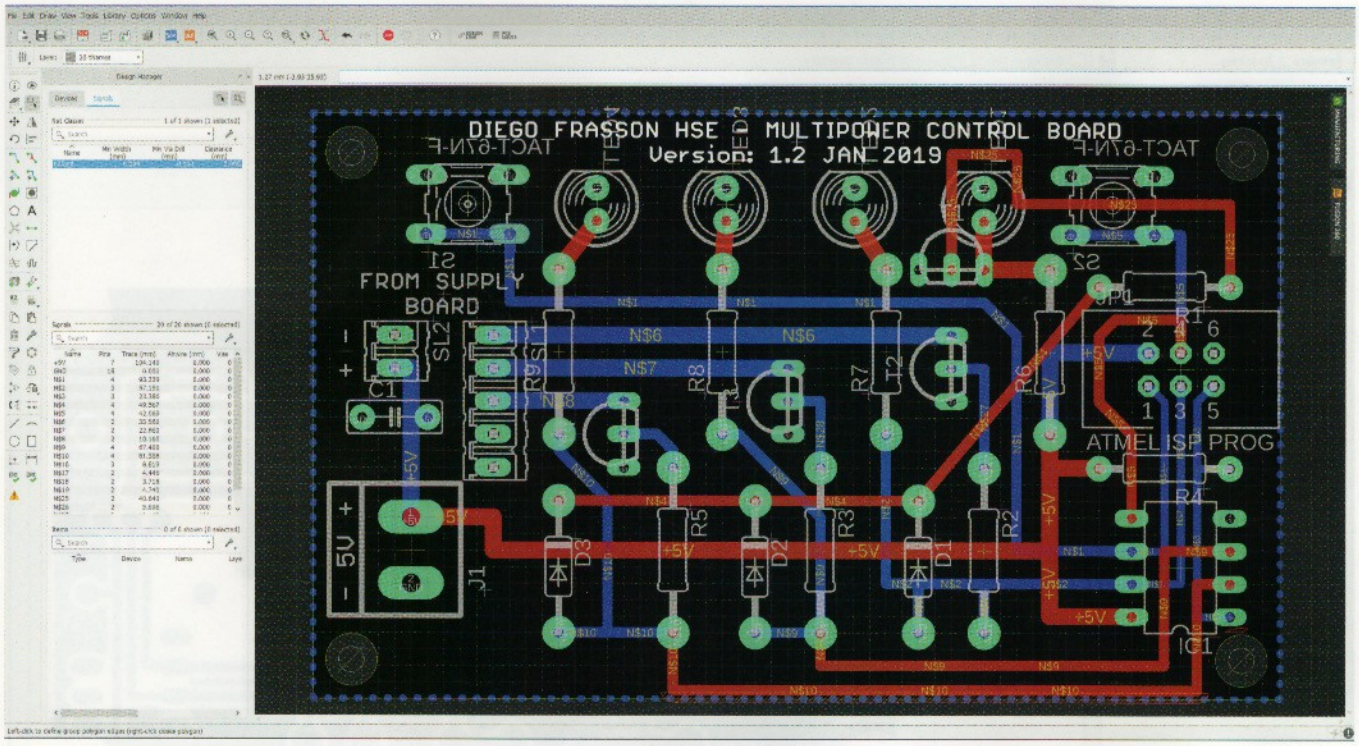


Figure 3 : le fichier « Attiny13A interrupt selector 1_2.brd » ouvert dans Eagle.

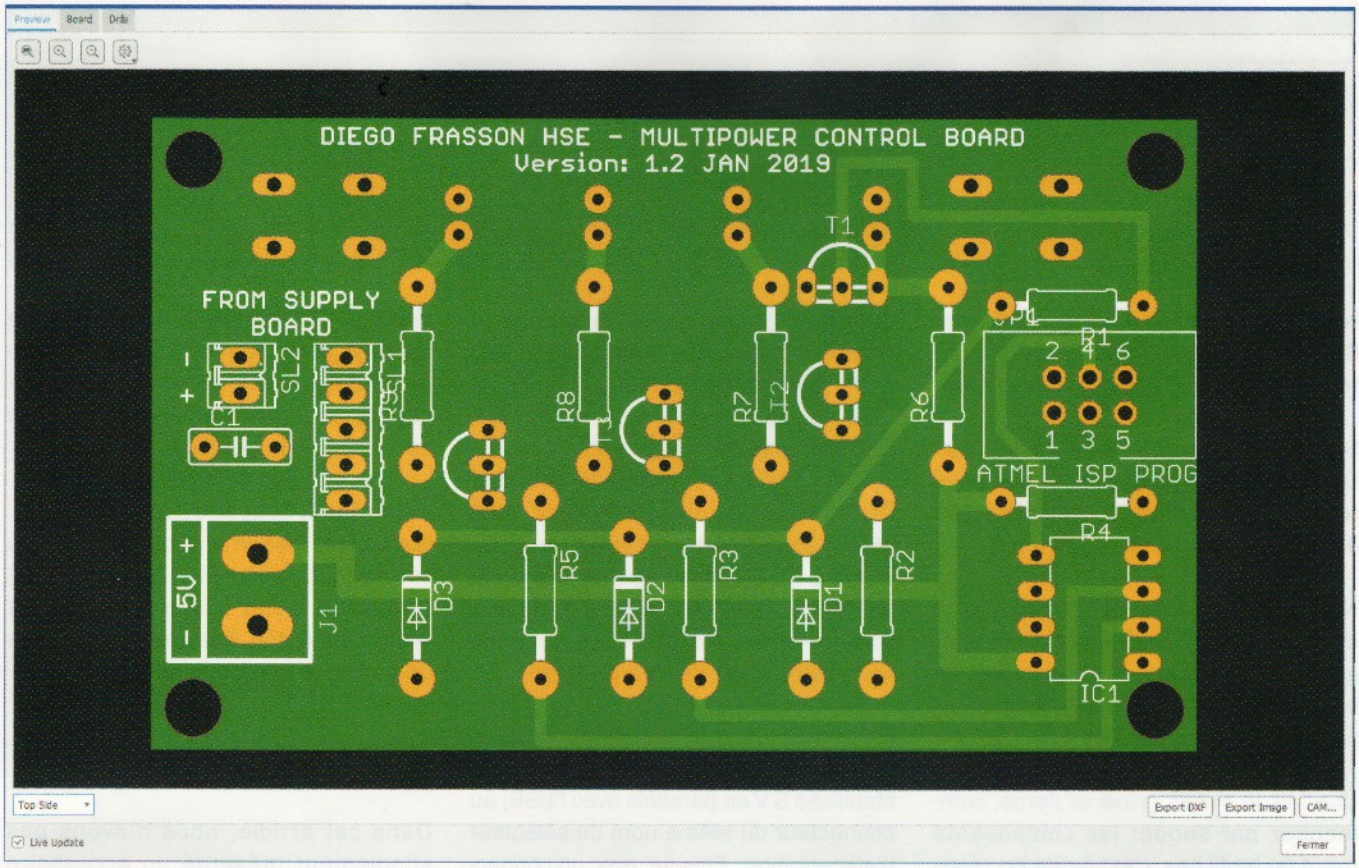


Figure 4 : en cliquant sur le bouton « Manufacturing » en haut à droite, vous pouvez voir un aperçu de la face supérieure du circuit une fois fabriqué.

En sélectionnant à gauche « Gerber » → « Bottom Copper », vous devez obtenir un résultat semblable à celui de la figure 5 qui représente la couche inférieure (bottom). En sélectionnant les autres couches, vous visualiserez les différents fichiers correspondants. Pour générer l'ensemble des fichiers Gerber, cliquez sur le bouton « **Process job** » en bas à droite. Ensuite, vous pouvez faire fabriquer votre circuit par un professionnel du circuit imprimé.

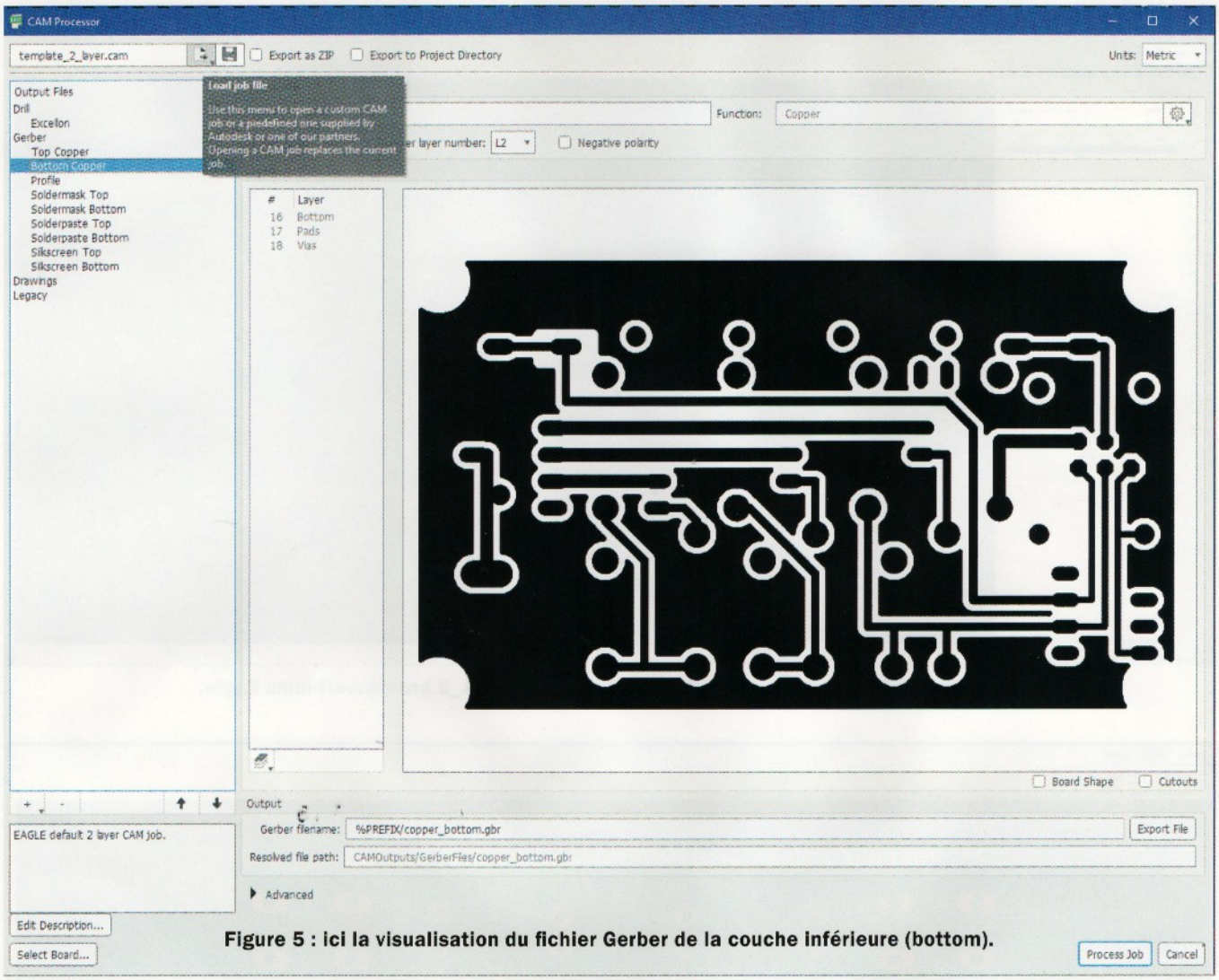


Figure 5 : ici la visualisation du fichier Gerber de la couche inférieure (bottom).

Nous avons nommé le dossier « Gerber Selecteur Alimentation » dans lequel se trouve l'ensemble des fichiers Gerber.

Si vous préférez une fabrication traditionnelle avec gravure chimique, vous trouverez les typons du circuit imprimé à l'échelle 1 au format « .pdf ».

Notez que le circuit imprimé comporte deux faces afin de permettre d'être intégré à une face avant avec les LED et les boutons orientés vers l'extérieur et les connecteurs vers l'intérieur.

Une fois le circuit gravé et percé, commencez par souder les composants ayant un profil bas, c'est-à-dire les résistances et les diodes, continuez avec le support du microcontrôleur, le condensateur, les transistors, le connecteur mâle 2 x 3 broches pour la programmation. Soudez ensuite les 4 LED, le bornier et les connecteurs. Pour être sûr de ne pas avoir de problèmes de contact, utilisez des boutons avec un axe long et des entretoises pour maintenir les LED à une distance correcte.

Une fois tous les composants soudés, vérifiez l'orientation des diodes, des transistors et du microcontrôleur. Vous pouvez alors insérer ce dernier dans son support. N'oubliez pas que les LED et les boutons doivent être soudés de l'autre côté du circuit imprimé (reportez-vous aux photos et au plan de câblage).

En ce qui concerne le câblage, reliez le connecteur SL2 de l'alimentation (sortie stabilisée 5 V en parallèle avec l'USB) au connecteur du même nom du sélecteur d'alimentation. Ensuite, avec un connecteur SIL au pas de 2,54 mm comportant 5 fils, connectez « SL1 » de l'alimentation à celui du sélecteur d'alimentation.

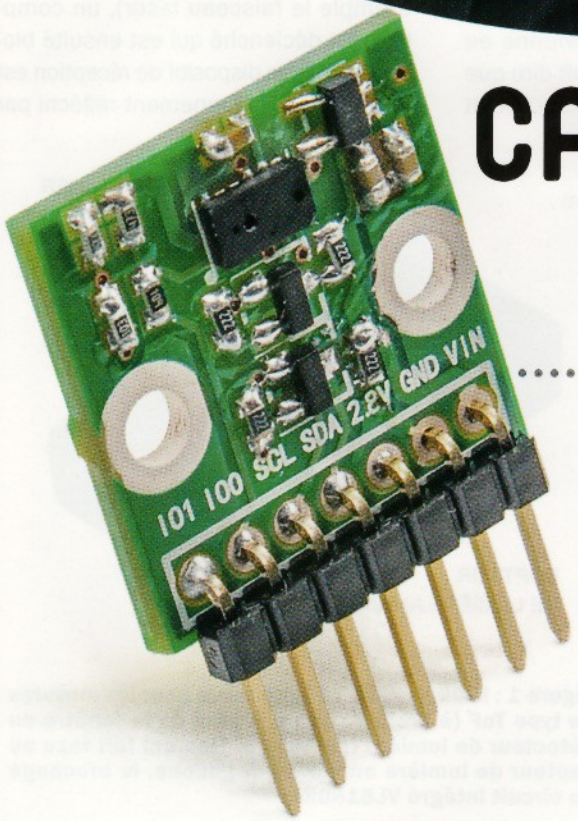
En figure 1, vous pouvez voir l'ensemble comprenant l'alimentation, le sélecteur et éventuellement un voltmètre relié à la sortie variable de l'alimentation. En figure 2, vous pouvez voir les connecteurs permettant de réaliser l'interconnexion entre l'alimentation et le sélecteur d'alimentation.

À ce stade, tout est prêt à l'emploi. Vous devez insérer le circuit imprimé du sélecteur dans le même boîtier de l'alimentation. Notez que le voltmètre est optionnel, il doit être relié aux bornes de sortie de la carte d'alimentation.

Conclusion

Dans cet article, nous n'avons pas simplement présenté un accessoire utile pour l'alimentation décrite dans le numéro 148, nous vous avons aussi enseigné les bases de l'utilisation des microcontrôleurs Atmel AVR. Nous espérons que les concepts décrits ici vous seront utiles pour mener à bien vos propres projets.

Basé sur la technologie « Time of Flight », ce montage fournit à un microcontrôleur des informations sur la distance parcourue par un faisceau lumineux grâce à son capteur optique en fonction du temps de propagation de l'onde lumineuse.



CAPTEUR LASER DE PROXIMITÉ

de Boris Landoni

La mesure d'une distance sans contact, et donc avec des instruments électroniques, est un problème auquel la technologie a offert, au fil des ans, diverses solutions. Celles basées sur la mesure d'un faisceau infrarouge ou d'une onde ultrasonique, jusqu'au radar traditionnel, qui fonctionne avec les micro-ondes radio.

Dans tous les cas, la détection utilise la mesure du « ***temps de vol*** » ou « **ToF** » (Time of Flight), qui est le **temps mis par une impulsion lumineuse pour atteindre l'objet à détecter et revenir au capteur.**

Les solutions existantes sont choisies en fonction de la distance à détecter, de l'élément permettant de déterminer la distance et les conditions de l'environnement dans lequel s'effectue la mesure, donc, en général, en fonction de l'application finale.

Dans cet article, nous souhaitons vous proposer un appareil de mesure, interfaçable avec un microcontrôleur et également avec Arduino, permettant d'exploiter la technologie **ST Flight Sense** (ToF) pour la détection de proximité, basée sur le circuit intégré VL6180XV0NR/1.

Ce montage permet la mesure d'une distance allant jusqu'à 10 cm (cela convient parfaitement pour une application de type robotique, ou pour un capteur de proximité, avec une précision de 1 mm.

Une librairie pour Arduino est disponible en téléchargement sur notre site dans le sommaire détaillé de la revue, afin de pouvoir tester les qualités et réaliser diverses expériences.

La technologie

Avant d'examiner en détail le projet, faisons un peu de théorie. Pour effectuer une mesure avec la technique « ToF », un rayon ou une vibration sonore est envoyé(e) vers un objet ou un mur, dont il faut déterminer la distance, puis le (la) détecter par réflexion sur un capteur.

CARACTÉRISTIQUES TECHNIQUES

- **Tension d'alimentation : 5 VDC stabilisée ;**
- **Consommation de courant : 10 mA ;**
- **Système de mesure : distance parcourue par le faisceau laser vers le capteur IR (et retour) selon une durée déterminée ;**
- **Distance maximale : 10 cm ;**
- **Longueur d'onde du laser et du capteur : 850 nm ;**
- **Interface : bus I2C.**

Cela fonctionne que si la surface de l'objet ou du mur n'absorbe pas trop le rayon émis.

La mesure est basée, comme mentionné, sur le temps mis par le faisceau pour aller de l'émetteur vers la surface de l'objet à détecter, puis pour revenir vers le capteur.

La distance (d) est déterminée en tant que produit de la vitesse de propagation de l'impulsion (v) pendant la moitié de ce temps (t) selon la relation :

$d = v * t/2$

La **vitesse de propagation dépend du milieu** dans lequel la mesure a lieu et, dans notre cas, nous supposons travailler dans l'air. Si nous utilisons des ultrasons, dont la vitesse de propagation est de 340 m/s, en supposant qu'une impulsion émise revienne au bout de 100 ms, cela voudrait dire que la distance à laquelle se trouve l'objet est égale à :

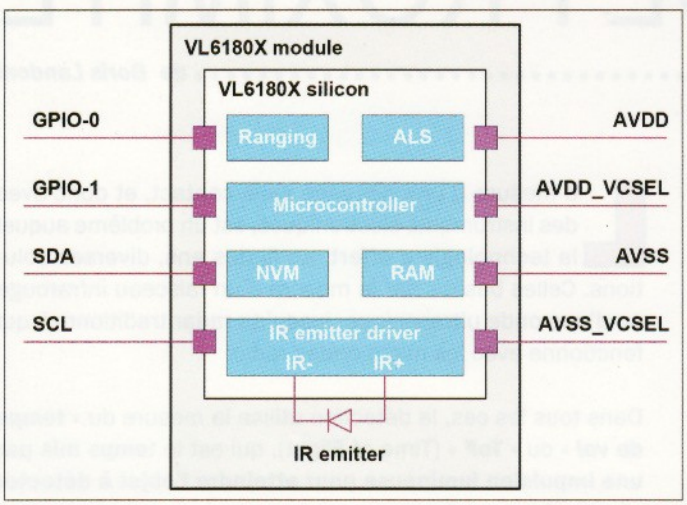
d = 340 m/s * 50 ms = 17 m

Si la mesure est effectuée avec un faisceau lumineux (par convention en prenant la vitesse de la lumière égale à 300 000 km/s), les intervalles de temps sont très petits. Par exemple, un intervalle de temps de 100 ms correspondrait à une distance de :

d = 300000 km/s x 50 ms = 15000 km

Dans le cas de notre capteur, devant mesurer des distances très courtes avec un faisceau lumineux, **il est essentiel d'utiliser des dispositifs électroniques avec des temps de latence très courts**, afin d'empêcher que leur retard de réponse ait une influence sur la mesure.

Pour comprendre cela, il est nécessaire de considérer le déroulement de la mesure. Lorsque l'alimentation est appliquée à l'élément rayonnant (par exemple le faisceau laser), un compteur est déclenché qui est ensuite bloqué lorsque le dispositif de réception est frappé par le rayonnement réfléchi par l'objet.



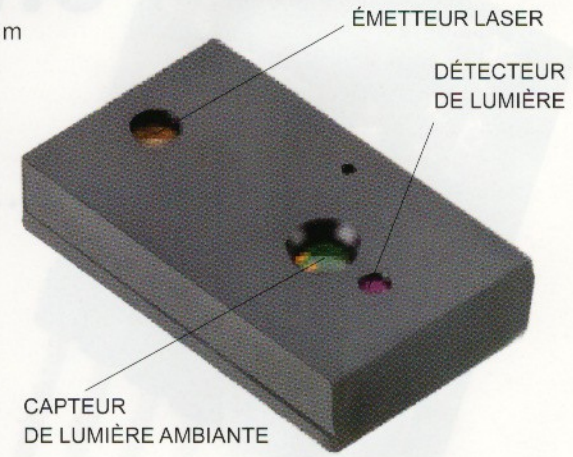


Figure 1 : le dispositif ST Flight Sense pour les mesures de type ToF (à droite) : notez, à côté de la fenêtre du détecteur de lumière (en bas), le trou qui fait face au capteur de lumière ambiante. À gauche, le brochage du circuit intégré VL6180X.

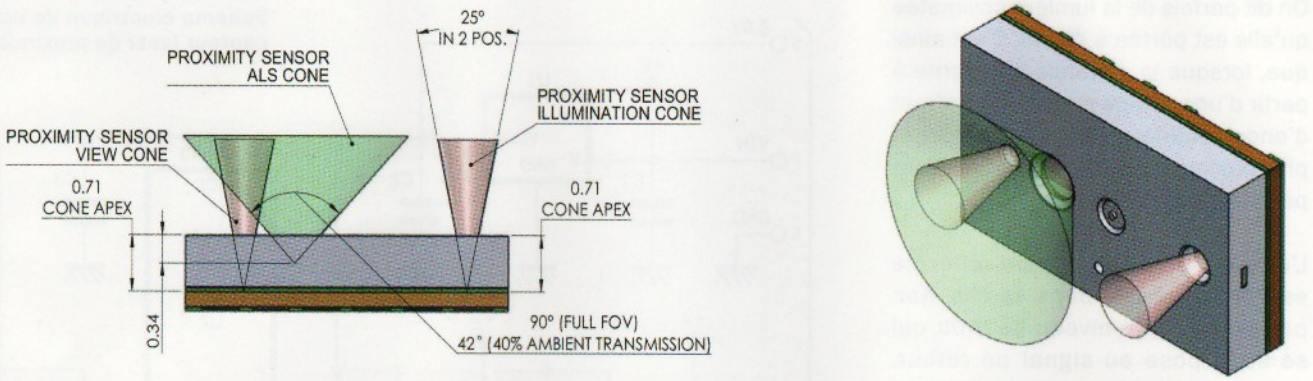


Figure 2 : structure du capteur ST : le cône de lumière émise par la LED est limité par rapport à l'ouverture du capteur SPAD.

La mesure est affectée par le temps de réponse de l'émetteur (de l'instant où l'alimentation est appliquée à celui où le faisceau est émis par la surface émettrice) et le temps de réponse du capteur (le temps écoulé entre le moment où les photons, qui composent le faisceau réfléchi, frappent la surface sensible du capteur et le signal électrique correspondant est mesuré sur les bornes du détecteur).

Une erreur de mesure peut provenir de la latence du compteur et des périphériques logiques interposés. Ces temps, étant donné les trajets très courts de l'impulsion lumineuse, doivent être très petits.

Une méthode utile **pour limiter l'erreur** (indispensable pour obtenir une très grande précision sur de petites distances) consiste à **positionner un deuxième capteur de lumière en regard de l'émetteur** (sans toutefois couvrir la surface émettrice de ce dernier) de manière à ce qu'il détecte la lumière quand elle sort. En théorie, cela compense à la fois le temps de réponse de l'émetteur et celui du capteur.

Cela dit, nous pouvons consacrer quelques paragraphes à la technologie ST, qui est brevetée et nommée Flightsense.

Ces dispositifs sont encapsulés dans un seul module CMS intégrant le capteur de lumière à **diode avalanche à photons** (Photon Avalanche Diode ou SPAD), un capteur de lumière ambiante traditionnel (Ambient Light Sensor ou ALS) et un **émetteur à diode laser de type à émission à cavité verticale** (Vertical Cavity Surface-Emitting Laser ou VCSEL) fonctionnant avec une longueur d'onde de 850 nm, donc dans l'infrarouge.

Cette technologie est idéale pour réaliser des détecteurs de proximité à courte distance, mais aussi des détecteurs de gestes. Le capteur de lumière ambiante a une excellente sensibilité allant de moins de 1 lux à 100 000 lux.

Une particularité innovante de **cette technique** est qu'elle permet d'**effectuer des mesures** pratiquement **indépendamment du coefficient de réflexion** de la lumière (réfléchie par la surface). Ceci est possible grâce à un circuit spécial permettant de réguler le gain de l'étage de détection de la lumière ambiante et d'émettre des impulsions très courtes.

La particularité de ce module réside dans le fait qu'il intègre, du côté du capteur de lumière (voir la figure 1), un deuxième détecteur de lumière servant à déterminer la luminosité ambiante, de manière à obtenir un signal qui est soustrait de l'impulsion résultant du retour du faisceau laser réfléchi.

De cette façon, l'amplitude du signal obtenu reflète fidèlement la distance. En d'autres termes, il est facile de discriminer la luminosité ambiante en connaissant le début des impulsions lumineuses et en attendant leur arrivée, car la lumière ambiante est relativement constante (uniforme).

La présence du capteur est primordiale dans des applications telles que les détecteurs de proximité placés derrière l'écran d'une tablette, d'un smartphone ou d'un écran tactile. Il permet de discriminer la réflexion d'une partie du rayonnement lumineux émis par la diode laser, à l'intérieur du verre, cela ne fonctionnerait pas avec un photodétecteur.

La figure 2 visualise une simulation du fonctionnement du capteur, avec la lumière infrarouge émise par le laser en rose et le cône de réception du capteur en vert, avec les angles d'ouverture relatifs.

Comme vous pouvez le constater, l'angle de réception du capteur est nettement supérieur à celui de l'émission laser, qui devrait idéalement être de zéro, car la meilleure solution consisterait à émettre un rayon laser qui ne s'ouvre pas, c'est-à-dire qu'il soit collimaté à l'infini.

Dans la pratique, un faisceau conique facilite la détection des objets, en particulier lorsqu'ils sont petits.

NB : une lumière collimatée est une lumière dont les rayonnements sont quasiment parallèles et se déploient lentement quand ils se propagent.

Le mot est relatif à « colinéaire » et implique une lumière qui ne se disperse pas avec la distance, ou qui serait très peu dispersée (dans la réalité). Un faisceau parfaitement collimaté, sans divergence, ne peut pas être créé à cause du principe de diffraction, mais la lumière ne peut qu'être approximativement collimatée par un certain nombre de processus, par exemple au moyen d'un collimateur.

On dit parfois de la lumière collimatée qu'elle est portée à l'infini. C'est ainsi que, lorsque la distance augmente à partir d'une source ponctuelle, le front d'onde sphérique devient plus plat et plus proche des ondes planes qui sont parfaitement collimatées.

L'élimination de la lumière réfléchie est fondamentale dans ce cas, non pas tant pour le niveau de bruit qui se superpose au signal de retour, mais, pour le calcul de l'impulsion du « temps de vol » de la lumière, étant donné que l'onde réfléchie à l'intérieur du verre arriverait avant la lumière réfléchie par l'objet dont la distance doit être mesurée. Il en résulterait une erreur de mesure et donc la distance mesurée serait inférieure.

La figure 3 schématise le fonctionnement du dispositif. La diode laser (émetteur) émet des impulsions périodiques de lumière dont une partie est réfléchie jusqu'à atteindre le capteur.

En synchronisant l'émission avec l'attente des impulsions (ceci peut être fait en considérant une plage d'intervalles de temps et une fréquence d'émission), il est possible d'éliminer les perturbations causées par la luminosité ambiante, augmentant ainsi la sensibilité du dispositif.

Contrairement, **les capteurs infrarouges classiques de proximité sont basés sur l'intensité de l'onde réfléchie**, ce qui les **rend peu fiables** car elle dépend du coefficient de réflexion de la surface de l'objet à détecter.

La particularité de la diode utilisée est qu'il s'agit d'une diode à photon unique

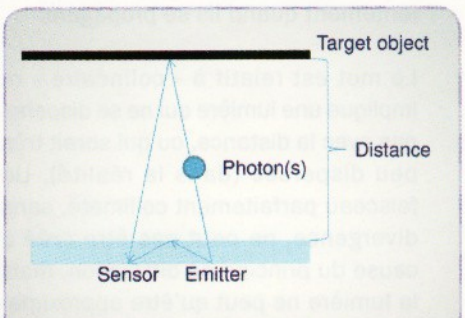


Figure 3 : schématisation du fonctionnement de la famille des circuits VL6180x.

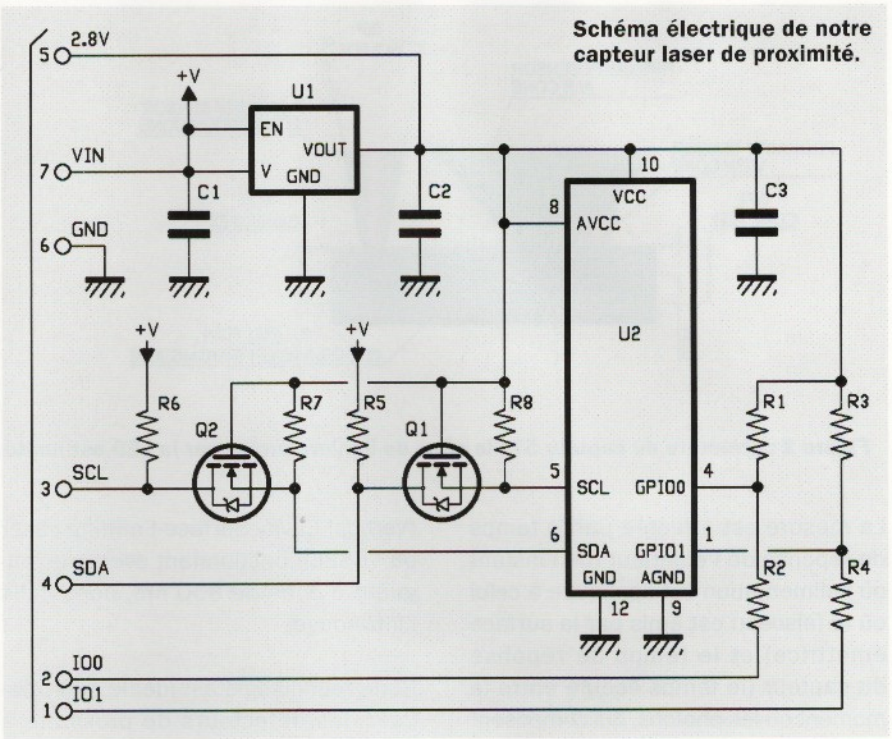


Schéma électrique de notre capteur laser de proximité.

(Single Photon Avalance Diode), c'est-à-dire une photodiode à polarisation inverse comme toutes les photodiodes (avec une résistance de charge placée en série). Elle fonctionne dans la zone où se produit l'avalanche, qui se situe généralement dans le troisième quadrant.

Le cycle de fonctionnement est décrit en figure 4. En maintenant la diode à une tension anode-cathode inverse proche de celle du claquage, lorsque la surface est frappée par la lumière, le premier photon déclenche la conduction due à l'effet d'avalanche et le courant augmente fortement, pour être ensuite annulé uniquement lorsque la tension tombe en dessous de celle de claquage (tension Vbd dans le diagramme de la figure 4).

À ce stade, un photon supplémentaire est nécessaire pour déclencher la conduction.

Ainsi, avec un réseau de diodes SPAD, il est possible de compter les photons. Aux bornes de la résistance de charge, une impulsion très courte peut donc être enregistrée à chaque photon, ce qui permet théoriquement la détection et le comptage individuel des photons, qui sont, selon la théorie quantique, les particules énergétiques constituant les rayonnements lumineux.

Le circuit intégré de chez ST est alimenté par les broches « AVDD » (positif) et AVSS (GND ou masse) pour la partie détection et par les broches « AVDD_VCSEL » et « AVSS_VCSEL » pour la diode laser.

De plus, il dispose de deux broches GPIO qui sont généralement utilisées comme entrées (il faut ajouter, comme dans le schéma électrique de notre circuit, des résistances de pull-up) pour gérer certaines fonctions.

La communication avec le dispositif hôte, généralement un microcontrôleur, s'effectue via le bus I2C à travers les broches SCL(5) et SDA(6), respectivement la ligne d'horloge et la ligne bidirectionnelle pour les données.

Le bus fonctionne à une fréquence de 400 kHz, il est possible de définir les modes de fonctionnement, la sensibilité, etc., mais également de lire le résultat de la mesure.

En termes de mode de fonctionnement, le circuit intégré peut effectuer la mesure soit avec une certaine périodicité réglable, soit sur demande ou requête.

Cette dernière méthode optimise la consommation, en grande partie grâce à l'émission laser qui est active que lors de la requête.

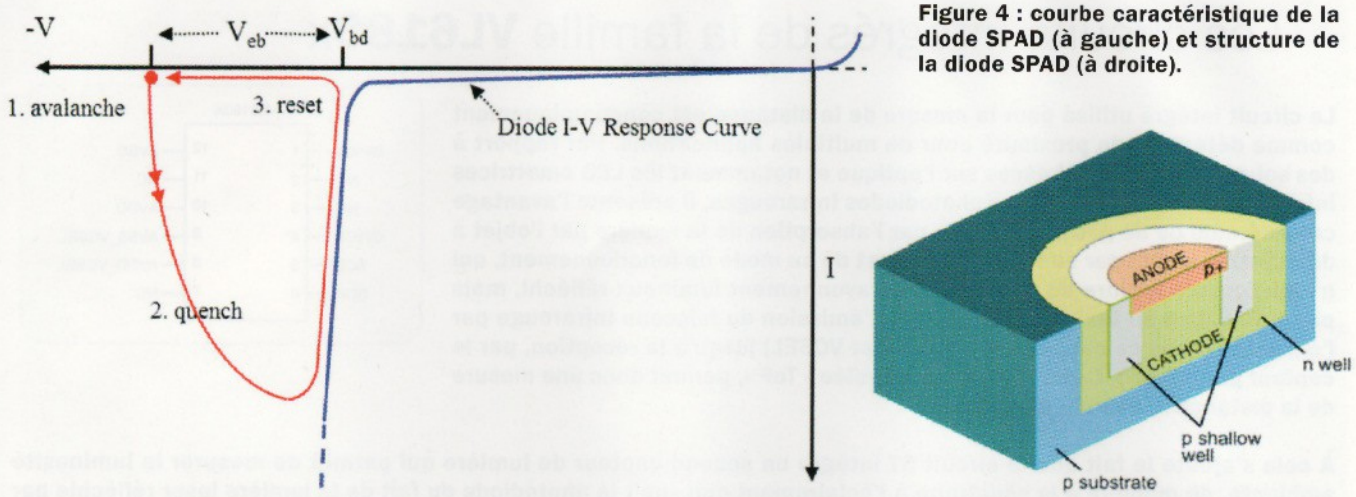


Figure 4 : courbe caractéristique de la diode SPAD (à gauche) et structure de la diode SPAD (à droite).

Pour adapter les niveaux de tension du circuit intégré ST à des niveaux TTL, un translateur bidirectionnel est utilisé. Il est constitué en partie par les deux transistors MOSFET, dont le fonctionnement peut être analysé ligne par ligne et niveau logique par niveau logique. Nous l'étudierons lors de l'analyse du schéma électrique.

Lorsque le niveau SCL de la broche 3 du connecteur est à un niveau logique haut, la diode de protection à l'intérieur du MOSFET (placée entre le drain et la source) est bloquée car la source est négative (2,8 V) par rapport au drain (qui est maintenu à 5 V à travers R6), même si l'impulsion n'atteint pas la broche SCL du circuit U2, car cette dernière est à un niveau logique 1.

Inversement, si le niveau SCL de la broche 3 du connecteur est à un niveau logique bas (0 V), le MOSFET est toujours bloqué (car sa grille est polarisée avec une tension de 2,8 V) mais sa diode de protection conduit, car la source (et donc l'anode de la diode) est polarisée par une tension de 2,8 V alors que la cathode est à une tension de 0 V (car SCL à 0 V).

Schéma électrique

Nous allons expliquer le fonctionnement de la technologie qui se cache dans notre montage. Ce dernier se présente sous la forme d'une carte d'expérimentation (breakout board) afin de mieux l'intégrer dans divers types de systèmes.

Le schéma électrique, outre le circuit intégré **VL6180XVONR/1**, comprend un régulateur de tension linéaire à faible chute de tension (U2) et un adaptateur à MOSFET pour l'interface I2C. Il permet la conversion des niveaux logiques 0/2,8 V vers des niveaux TTL standard comme ceux des cartes électroniques traditionnelles ou encore Arduino.

Pour expliquer le fonctionnement du translateur de niveaux, commençons par le signal d'horloge, qui du côté de U2 est connecté à la broche 5, tandis que du côté du connecteur il est relié à la ligne SCL (broche 3 du connecteur du côté de R6).

Les deux lignes SCL sont dotées chacune de résistances de pull-up (R8 pour SCL de U2 et R6 pour la broche 3 du connecteur de la carte).

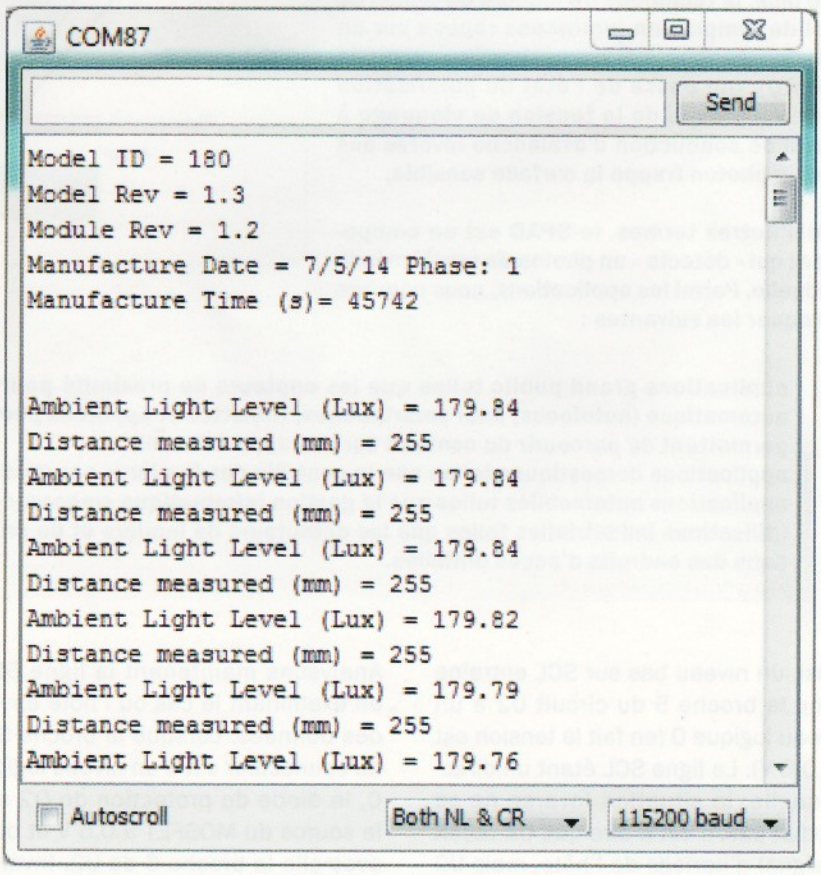
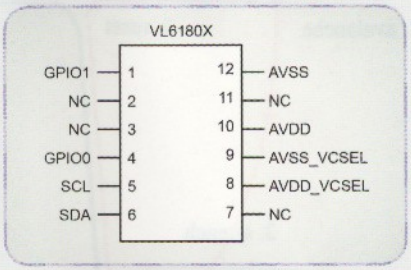


Figure 6 : fenêtre du moniteur série d'Arduino, vous pouvez voir les valeurs provenant de la breakout board ToF s'afficher.

Les circuits intégrés de la famille **VL6180x**



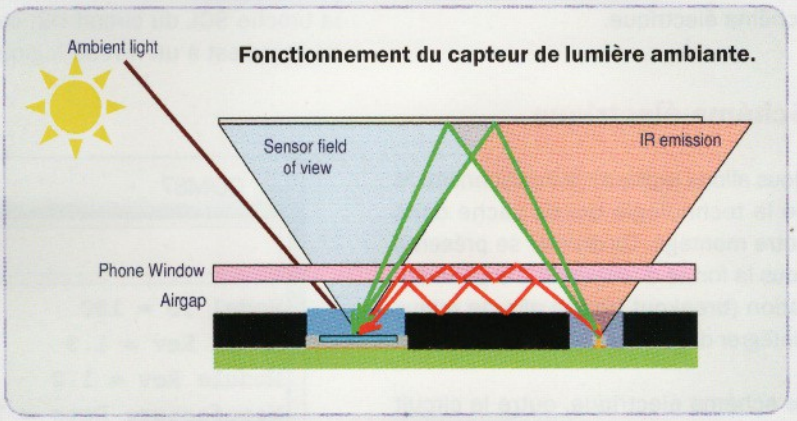
Le circuit intégré utilisé pour la mesure de la distance est conçu uniquement comme détecteur de proximité pour de multiples applications. Par rapport à des solutions similaires basées sur l'optique et notamment les LED émettrices infrarouges et les détecteurs à photodiodes infrarouges, il présente l'avantage considérable de ne pas être affecté par l'absorption de la lumière par l'objet à détecter, ou plutôt par sa surface. Cela est dû au mode de fonctionnement, qui n'inclut pas la lecture de l'intensité du rayonnement lumineux réfléchi, mais plutôt le calcul du temps écoulé depuis l'émission du faisceau infrarouge par l'émetteur (dans ce cas, il s'agit d'un laser VCSEL) jusqu'à la réception, par le capteur photodiode. Cette technique, appelée « ToF », permet donc une mesure de la distance beaucoup plus fiable.

À cela s'ajoute le fait que le circuit ST intègre un second capteur de lumière qui permet de mesurer la luminosité ambiante, de manière à la soustraire à l'éclairement que subit la photodiode du fait de la lumière laser réfléchie par l'objet ou l'obstacle à détecter.

La présence du second capteur est également idéale pour les applications intégrées comme un capteur de proximité présent dans les tablettes et les smartphones, ou encore en disposant le circuit intégré derrière un écran tactile.

En effet, avec des capteurs conventionnels, les réflexions et les réfractions produites ramènent une bonne partie de la lumière émise par le laser sur la photodiode, ce qui fausse la mesure. Le phénomène est décrit dans la figure ci-après.

Une autre caractéristique, contribuant à rendre le circuit peu sensible à l'absorption du faisceau lumineux par la surface de l'objet dont il faut mesurer la distance, est la présence d'un circuit spécial qui permet de réguler le gain de l'étage de réception et donc de la lumière ambiante et d'émettre des impulsions très courtes.



Fonctionnement du capteur de lumière ambiante.

De plus, la technique de mesure du temps de vol de l'impulsion lumineuse repose sur un photodétecteur à diode avalanche à photons (SPAD), qui passe de l'état de polarisation inverse proche de la tension de claquage à celui de conduction d'avalanche inverse dès qu'un photon frappe la surface sensible.

En d'autres termes, le SPAD est un composant qui « détecte » un photon de manière individuelle. Parmi les applications, nous pouvons évoquer les suivantes :

- **applications grand public telles que les capteurs de proximité pour smartphones, systèmes de mise au point automatique (autofocus) pour smartphones, tablettes et appareils photo, systèmes de reconnaissance des gestes permettant de parcourir du contenu sur des appareils mobiles ;**
- **applications domestiques telles que le contrôle des lumières par geste, les robots, les jouets, etc. ;**
- **applications automobiles telles que la gestion informatique embarquée et l'infodivertissement pour l'automobile ;**
- **utilisations industrielles telles que les détecteurs de lumière et de proximité, commandes de portes et robotique dans des endroits d'accès difficiles.**

Ainsi, un niveau bas sur SCL entraîne donc la broche 5 du circuit U2 à un niveau logique 0 (en fait la tension est de 0,6 V). La ligne SCL étant unidirectionnelle, la situation inverse ne se produit pas, c'est-à-dire que U2 reçoit le signal d'horloge de l'hôte, mais U2 ne fournit pas de signal d'horloge au microcontrôleur (hôte).

Analysons maintenant la ligne SDA, en examinant le cas où l'hôte envoie des données. Lorsque la broche SDA du connecteur est à un niveau logique 0, la diode de protection de Q2 met la source du MOSFET à 0,6 V et donc avec elle la broche 6 de U2. Inversement, quand elle est à 5 V, le MOSFET et la diode sont bloqués, donc la broche SDA de U2 est amenée à une tension de 2,8 V (niveau haut) par la résistance de tirage R7.

Si le VL6180 est en train de transmettre, lorsqu'il met sa broche 6 à un niveau logique 0, le MOSFET Q2 devient conducteur entre son drain et sa source, car sa source est à 0 V tandis que sa

grille (gate) est soumise à une tension de 2,8 V. Par conséquent, son courant de drain tire la broche SDA du connecteur à environ 0 V, en raison de la chute de tension dans la résistance de pull-up R6.

Inversement, lorsque la broche 6 de U1 est à un niveau logique haut (2,8 V), la source et la grille sont à un même potentiel (elles sont équipotentielles) et le MOSFET reste bloqué.

Ainsi, la résistance de pull-up R6 maintient à un niveau logique 1 (mais à 5 V) la ligne SDA présente sur le connecteur. Quant au régulateur de tension, il s'agit d'un MIC5219 de chez Micrel.

Listing 1

```
// Réglages recommandés par la documentation technique
//http://www.st.com/st-web-ui/static/active/en/resource/technical/document/application_note/DM00122600.pdf

// Activation des interruptions lorsque la conversion est terminée (toutes les sources)
VL6180x_setRegister(VL6180X_SYSTEM_INTERRUPT_CONFIG_GPIO, (4 << 3)|(4) ); // configure GPIO1 à un niveau logique haut à la fin de l'échantillonnage
VL6180x_setRegister(VL6180X_SYSTEM_MODE_GPIO1, 0x10); // configure GPIO1 à un niveau logique haut à la fin de l'échantillonnage
 VL6180x_setRegister(VL6180X_READOUT_AVERAGING_SAMPLE_PERIOD, 0x30); // Définit la période d'échantillonnage
 VL6180x_setRegister(VL6180X_SYSALS_ANALOGUE_GAIN, 0x46); // Configure le gain ALS
 VL6180x_setRegister(VL6180X_SYSRANGE_VHV_REPEAT_RATE, 0xFF); // Définit la période de calibration automatique (Max = 255)/(OFF = 0)
VL6180x_setRegister(VL6180X_SYSALS_INTEGRATION_PERIOD, 0x63); // Configure le temps d'intégration d'ALS à 100 ms
VL6180x_setRegister(VL6180X_SYSRANGE_VHV_RECALIBRATE, 0x01); // Effectue un seul étalonnage de la température

// Paramètres facultatifs de la documentation technique
//http://www.st.com/st-web-ui/static/active/en/resource/technical/document/application_note/DM00122600.pdf

VL6180x_setRegister(VL6180X_SYSRANGE_INTERMEASUREMENT_PERIOD, 0x09); // Définit par défaut la période entre 2 mesures à 100 ms
 VL6180x_setRegister(VL6180X_SYSALS_INTERMEASUREMENT_PERIOD, 0x0A); // Définit par défaut la période entre 2 mesures d'ALS à 100 ms
 VL6180x_setRegister(VL6180X_SYSTEM_INTERRUPT_CONFIG_GPIO, 0x24); // Configure l'interruption lors de l'événement « Nouvel échantillon prêt »

// Paramètres supplémentaires par défaut de la communauté
 VL6180x_setRegister(VL6180X_SYSRANGE_MAX_CONVERGENCE_TIME, 0x32);
 VL6180x_setRegister(VL6180X_SYSRANGE_RANGE_CHECK_ENABLES, 0x10 | 0x01);
 VL6180x_setRegister16bit(VL6180X_SYSRANGE_EARLY_CONVERGENCE_ESTIMATE, 0x7B );
 VL6180x_setRegister16bit(VL6180X_SYSALS_INTEGRATION_PERIOD, 0x64);

 VL6180x_setRegister(VL6180X_READOUT_AVERAGING_SAMPLE_PERIOD,0x30);
 VL6180x_setRegister(VL6180X_SYSALS_ANALOGUE_GAIN,0x40);
 VL6180x_setRegister(VL6180X_FIRMWARE_RESULT_SCALER,0x01);
}
```

Détecteur de gestes pour l'**automobile**

MediaTek et Pixart Imaging ont entamé une collaboration visant à développer des détecteurs de gestes en 3D, basés sur la combinaison d'un circuit intégré MT8665 (quadricœur basé sur ARM Cortex-A53) et de capteurs Pixart Imaging PAJ7620U2 communiquant via le bus I2C et qui sont capables de reconnaître jusqu'à 9 gestes, y compris les mouvements « monter », « descendre », « aller à gauche », « aller à droite », etc. avec simplement un toucher de la main. Cette solution permettra de mettre au point des systèmes de reconnaissance des gestes, par exemple dans les systèmes multimédias présents dans les automobiles, en remplacement des commandes et des écrans tactiles actuels.

Listing 2

```
#include <Wire.h>
#include <SparkFun_VL6180X.h>

#define VL6180X_ADDRESS 0x29

VL6180xIdentification identification;
VL6180x sensor(VL6180X_ADDRESS);

void setup()
{
  Serial.begin(115200);                    //début de la communication série à 115200 bps
  Wire.begin();                            //démarrage de la librairie I2C
  delay(100);                              // délai 0,1 s (100 ms)
  sensor.getIdentification(&identification);   // récupère les informations du fabricant depuis la mémoire
  printIdentification(&identification);    // fonction d'assistance pour afficher toutes les informations du module

   if(sensor.VL6180xInit() != 0)
   {
     Serial.println("FAILED TO INITALIZE");    // initialise le périphérique et recherche les erreurs

   };
  sensor.VL6180xDefaultSettings();         // charge les paramètres par défaut pour démarrer.
```

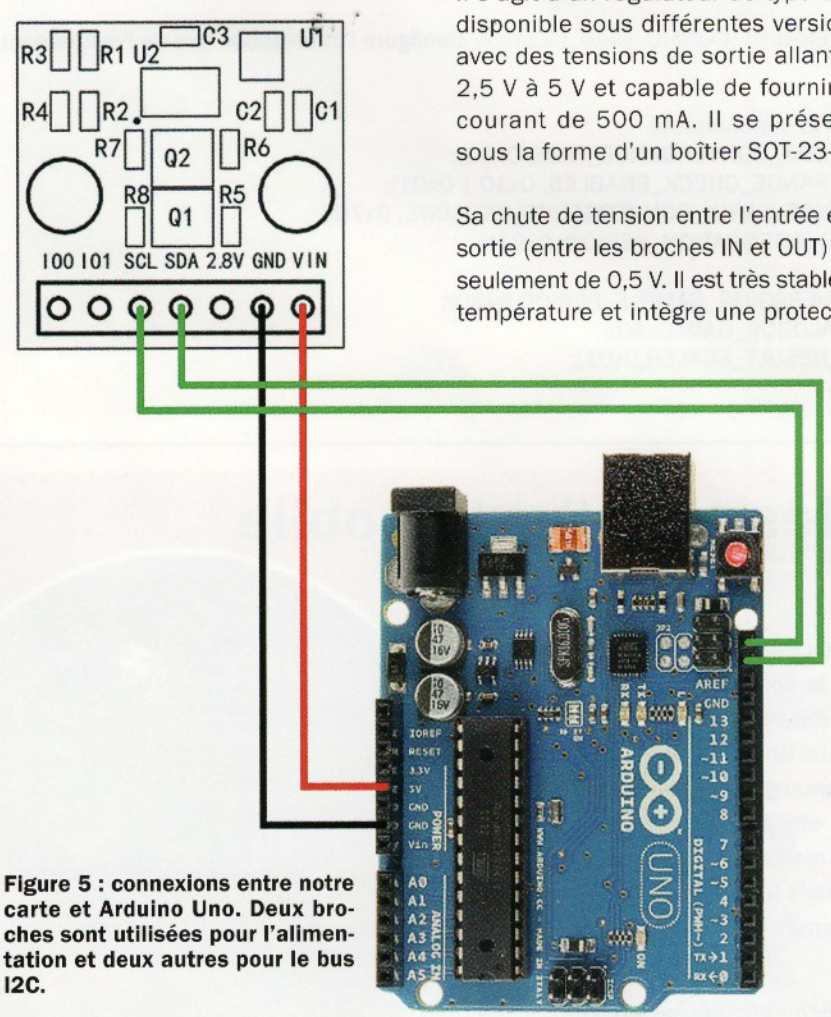


Figure 5 : connexions entre notre carte et Arduino Uno. Deux broches sont utilisées pour l'alimentation et deux autres pour le bus I2C.

Il s'agit d'un régulateur de type LDO, disponible sous différentes versions, avec des tensions de sortie allant de 2,5 V à 5 V et capable de fournir un courant de 500 mA. Il se présente sous la forme d'un boîtier SOT-23-5.

Sa chute de tension entre l'entrée et la sortie (entre les broches IN et OUT) est seulement de 0,5 V. Il est très stable en température et intègre une protection contre les courts-circuits en sortie et une protection contre une température excessive, ainsi qu'une protection contre l'inversion de polarité sur l'entrée.

Ce régulateur peut être mis en vielle (shutdown) à l'aide de la broche « EN ». Dans ce mode, il ne consomme que quelques microampères. Lorsque la broche « EN » est mise à un niveau logique 1, le régulateur est en fonctionnement normal, alors qu'à un niveau logique 0, le régulateur est en veille. Cette broche accepte des niveaux logiques TTL jusqu'à 5 V.

La broche « BYP » permet de connecter un condensateur de 470 pF typiquement vers la masse, ce qui permet de mieux filtrer la tension de référence de l'amplificateur d'erreur et donc de minimiser le bruit qui pourrait se superposer à la tension de sortie.

Le condensateur doit avoir une valeur comprise entre 470 pF et 1 nF, mais cette broche peut être laissée non connectée (comme dans notre montage).

Avant de terminer la description du schéma électrique, arrêtons-nous

```
  delay(1000); // delay 1s
}

void loop()
{
 //Mesure du niveau de la lumière ambiante en lux
 // GAIN en entrée pour les niveaux de lumière,
 // GAIN_20    // gain actuel d'ALS : 20
 // GAIN_10    // gain actuel d'ALS  : 10.32
 // GAIN_5     // gain actuel d'ALS : 5.21
 // GAIN_2_5   // gain actuel d'ALS : 2.60
 // GAIN_1_67  // gain actuel d'ALS :  1.72
 // GAIN_1_25  // gain actuel d'ALS : 1.28
 // GAIN_1     // gain actuel d'ALS : 1.01
 // GAIN_40    // gain actuel d'ALS  : 40

 Serial.print("Ambient Light Level (Lux) = ");
 Serial.println( sensor.getAmbientLight(GAIN_1) );

//mesure de la distance et report en mm
 Serial.print("Distance measured (mm) = ");
 Serial.println( sensor.getDistance() );
 delay(500);
};

void printIdentification(struct VL6180xIdentification *temp)
{
 Serial.print("Model ID = ");
 Serial.println(temp->idModel);
 Serial.print("Model Rev = ");
 Serial.print(temp->idModelRevMajor);
 Serial.print(".");
 Serial.println(temp->idModelRevMinor);
 Serial.print("Module Rev = ");
 Serial.print(temp->idModuleRevMajor);
 Serial.print(".");
 Serial.println(temp->idModuleRevMinor);
 Serial.print("Manufacture Date = ");
 Serial.print((temp->idDate >> 3) & 0x001F);
 Serial.print("/");
 Serial.print((temp->idDate >> 8) & 0x000F);
 Serial.print("/1");
 Serial.print((temp->idDate >> 12) & 0x000F);
 Serial.print(" Phase: ");
 Serial.println(temp->idDate & 0x0007);
 Serial.print("Manufacture Time (s)= ");
 Serial.println(temp->idTime * 2);
 Serial.println();
 Serial.println();
}
```

un instant sur les broches GPIO du VL6180XVONR/1, qui, comme indiqué, peuvent être configurées à l'aide de commandes spéciales envoyées via le bus I2C.

La configuration de notre projet est incluse dans la librairie spéciale qui peut être téléchargée sur notre site web dans le sommaire détaillé de la revue.

La librairie est reportée dans le Listing 1.

Les broches GPIO0 et GPIO1 remplissent les fonctions suivantes :

- **GPIO0** : change d'état une fois l'acquisition terminée, elle peut être utilisée pour communiquer avec le microcontrôleur hôte afin de lire la mesure ;

- **GPIO1** : si elle est mise à la masse (GND), le circuit intégré ST est en veille. Une fois dans cette condition, le circuit **U2 peut être réactivé en renvoyant sur le bus I2C la configuration requise** pour la mesure (**il ne suffit donc pas de ramener la broche GPIO1 à un niveau élevé pour activer U2**).

La librairie de gestion du VL6180XVONR/1 provient du site **SparkFun** et est également téléchargeable à l'adresse suivante : https://github.com/sparkfun/SparkFun_ToF_Range_Finder-VL6180_Arduino_Library.

Pour pouvoir utiliser la librairie, après l'avoir téléchargée, vous devez copier le dossier contenu dans le fichier « .zip » dans le répertoire « **libraries** » où est installé (sur votre disque dur) l'environnement Arduino.

Démonstration avec Arduino

Nous allons maintenant tester notre breakout board en combinaison avec une carte Arduino Uno.

La figure 5 visualise l'ensemble des connexions des deux cartes. Comme vous pouvez le constater, nous n'utilisons que 4 fils, car nous nous limitons à l'alimentation et aux deux lignes du bus I2C.

En effet, il s'agit d'effectuer un test d'évaluation des caractéristiques du capteur ST, il suffit donc de pouvoir réaliser quelques mesures.

Une fois la librairie chargée dans le répertoire « **libraries** » où est installé l'environnement Arduino, vous pouvez ouvrir l'IDE, puis dans le menu « Fichier » → « Exemples ».

Vous trouverez « **SparkFun VL6180 Sensor** », qui servira d'exemple pour gérer le capteur.

Plan de montage

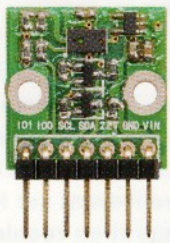
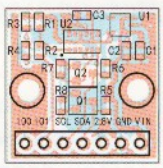


Photo de l'un de nos prototypes du capteur laser de proximité.



Plan de câblage des composants du capteur laser de proximité.

Liste des composants

R1..... 47 kΩ boîtier CMS 0603
R2..... 10 kΩ boîtier CMS 0603
R3..... 47 kΩ boîtier CMS 0603
R4..... 10 kΩ boîtier CMS 0603
R5..... 2,2 kΩ boîtier CMS 0603
R6..... 2,2 kΩ boîtier CMS 0603
R7..... 2,2 kΩ boîtier CMS 0603
R8..... 2,2 kΩ boîtier CMS 0603

C1..... 1 µF céramique boîtier CMS 0603
C2..... 1 µF céramique boîtier CMS 0603
C3..... 1 µF céramique boîtier CMS 0603

Q1..... BSS138W-7-F
Q2..... BSS138W-7-F

U1..... MIC5504-2.8YM5-TR
U2..... VL6180XV0NR/1

Le sketch d'exemple (programme) est reporté au Listing 2. Il est prêt à être utilisé, aucune modification n'est nécessaire.

La communication sur le bus I2C est initialisée avec une horloge à 400 kHz, conformément aux spécifications du circuit VL6180XV0NR/1. Vous pouvez maintenant charger (téléverser) le sketch dans Arduino.

Le sketch d'exemple initialise la connexion série, puis charge la librairie du bus I2C pour communiquer avec la breakout board, puis ensuite il commence à interroger celle-ci de manière cyclique afin de retourner les données correspondantes.

Le même sketch envoie sur le port série ouvert pour Arduino les données relatives en lux et la distance de l'objet.

Pour visualiser ces données, vous devez donc ouvrir le moniteur série d'Arduino et configurer un « baud-rate » de 115 200 bps.

De cette manière, la fenêtre du moniteur série affichera les données, comme vous pouvez le voir en figure 6, où un exemple de données provenant de la breakout board apparaît.

Réalisation pratique

Nous concluons cet article en abordant la réalisation pratique de ce capteur laser de proximité.

La construction commence par la préparation du circuit imprimé double face, pour lequel nous proposons en téléchargement sur notre site web dans le sommaire détaillé de la revue les fichiers gerber pour une fabrication professionnelle ainsi que les typons à l'échelle 1 pour la gravure chimique classique.

Une fois le circuit gravé et percé, vous devez prendre une attention particulière pour souder les composants. Le montage est un peu difficile étant donné que les composants utilisés sont tous de type CMS (montage en surface).

Procurez-vous donc un fer à souder à pointe fine de 0,2 mm de Ø, de la soudure pour composants CMS d'un diamètre maximal de 0,5 mm, de la pâte à flux et un petit pinceau pour l'étaler.

Vous devez aussi vous munir d'une pince à épiler pour manipuler les composants et d'une loupe pour vérifier qu'ils sont correctement positionnés, ensuite vous pouvez les souder délicatement.

Vous pouvez éventuellement utiliser de la tresse à dessouder pour éliminer tout excès d'étain qui pourrait court-circuiter des broches adjacentes.

Comme d'habitude, commencez par souder les composants les plus petits, c'est-à-dire les résistances et les condensateurs, puis positionnez le circuit intégré en le centrant correctement.

En effet, ses broches se situent en dessous du boîtier, vous devez faire attention à ne pas provoquer de courts-circuits.

Avant de souder le circuit, vérifiez son orientation en vous aidant du plan de montage.

Terminez en soudant les MOSFET et la barrette qui doit être à angle droit, afin de pouvoir monter verticalement la carte dans tout système existant.

Lorsque vous utilisez le flux, veillez à ne pas salir les trous de la surface du VL6180XV0NR/1, qui doivent rester libres et propres pour émettre et capter correctement la lumière, faute de quoi le système de détection et de mesure risquent de mal fonctionner.

Une fois le montage terminé, vérifiez à la loupe que tous les composants sont bien orientés et soudés (les soudures réussies sont brillantes, alors que les soudures opaques risquent d'être oxydées et de provoquer des faux contacts).

NB : la breakout board « ToF » avec le capteur VL6180 déjà monté et testé est disponible sur le site web www.futurashop.it sous la référence 7100-BREAKOUT017.

Plan de montage

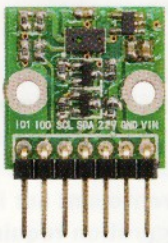
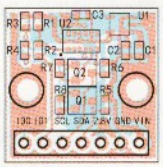


Photo de l'un de nos prototypes du capteur laser de proximité.



Plan de câblage des composants du capteur laser de proximité.

Liste des composants

R1..... 47 kΩ boîtier CMS 0603
R2..... 10 kΩ boîtier CMS 0603
R3..... 47 kΩ boîtier CMS 0603
R4..... 10 kΩ boîtier CMS 0603
R5..... 2,2 kΩ boîtier CMS 0603
R6..... 2,2 kΩ boîtier CMS 0603
R7..... 2,2 kΩ boîtier CMS 0603
R8..... 2,2 kΩ boîtier CMS 0603

C1..... 1 µF céramique boîtier CMS 0603
C2..... 1 µF céramique boîtier CMS 0603
C3..... 1 µF céramique boîtier CMS 0603

Q1..... BSS138W-7-F
Q2..... BSS138W-7-F

U1..... MIC5504-2.8YM5-TR
U2..... VL6180XV0NR/1

Le sketch d'exemple (programme) est reporté au Listing 2. Il est prêt à être utilisé, aucune modification n'est nécessaire.

La communication sur le bus I2C est initialisée avec une horloge à 400 kHz, conformément aux spécifications du circuit VL6180XV0NR/1. Vous pouvez maintenant charger (téléverser) le sketch dans Arduino.

Le sketch d'exemple initialise la connexion série, puis charge la librairie du bus I2C pour communiquer avec la breakout board, puis ensuite il commence à interroger celle-ci de manière cyclique afin de retourner les données correspondantes.

Le même sketch envoie sur le port série ouvert pour Arduino les données relatives en lux et la distance de l'objet.

Pour visualiser ces données, vous devez donc ouvrir le moniteur série d'Arduino et configurer un « baud-rate » de 115 200 bps.

De cette manière, la fenêtre du moniteur série affichera les données, comme vous pouvez le voir en figure 6, où un exemple de données provenant de la breakout board apparaît.

Réalisation pratique

Nous concluons cet article en abordant la réalisation pratique de ce capteur laser de proximité.

La construction commence par la préparation du circuit imprimé double face, pour lequel nous proposons en téléchargement sur notre site web dans le sommaire détaillé de la revue les fichiers gerber pour une fabrication professionnelle ainsi que les typons à l'échelle 1 pour la gravure chimique classique.

Une fois le circuit gravé et percé, vous devez prendre une attention particulière pour souder les composants. Le montage est un peu difficile étant donné que les composants utilisés sont tous de type CMS (montage en surface).

Procurez-vous donc un fer à souder à pointe fine de 0,2 mm de Ø, de la soudure pour composants CMS d'un diamètre maximal de 0,5 mm, de la pâte à flux et un petit pinceau pour l'étaler.

Vous devez aussi vous munir d'une pince à épiler pour manipuler les composants et d'une loupe pour vérifier qu'ils sont correctement positionnés, ensuite vous pouvez les souder délicatement.

Vous pouvez éventuellement utiliser de la tresse à dessouder pour éliminer tout excès d'étain qui pourrait court-circuiter des broches adjacentes.

Comme d'habitude, commencez par souder les composants les plus petits, c'est-à-dire les résistances et les condensateurs, puis positionnez le circuit intégré en le centrant correctement.

En effet, ses broches se situent en dessous du boîtier, vous devez faire attention à ne pas provoquer de courts-circuits.

Avant de souder le circuit, vérifiez son orientation en vous aidant du plan de montage.

Terminez en soudant les MOSFET et la barrette qui doit être à angle droit, afin de pouvoir monter verticalement la carte dans tout système existant.

Lorsque vous utilisez le flux, veillez à ne pas salir les trous de la surface du VL6180XV0NR/1, qui doivent rester libres et propres pour émettre et capter correctement la lumière, faute de quoi le système de détection et de mesure risquent de mal fonctionner.

Une fois le montage terminé, vérifiez à la loupe que tous les composants sont bien orientés et soudés (les soudures réussies sont brillantes, alors que les soudures opaques risquent d'être oxydées et de provoquer des faux contacts).

NB : la breakout board « ToF » avec le capteur VL6180 déjà monté et testé est disponible sur le site web www.futurashop.it sous la référence 7100-BREAKOUT017.

DRIVER POUR MOTEUR BLDC

de Davide Gironi

Dans cet article, nous vous proposons de réaliser un driver pour moteur à courant continu sans balais ou BLDC (BrushLess DC). Son architecture est ouverte, elle est basée sur un microcontrôleur Atmel pour ce qui est de la gestion et sur des MOSFET pour la partie puissance.

Tant dans le secteur industriel que dans le secteur grand public, les moteurs électriques sont des éléments fondamentaux pour le développement d'un grand nombre de dispositifs technologiques.

Par exemple, les moteurs présents dans les machines à laver, les réfrigérateurs, les perceuses, les ventilateurs, les imprimantes de toutes sortes, mais aussi dans les voitures électriques récentes.

Les moteurs à courant alternatif traditionnels sont répandus, simples à produire et n'ont besoin d'aucun composant électronique pour fonctionner. Mais lorsqu'il est nécessaire de faire varier le régime ou le couple, ils nécessitent des circuits de découpage qui n'ont pas un très haut rendement. Pour cette raison, nous nous tournons de plus en plus vers les moteurs sans balais (littéralement « brushless »), qui

constituent la famille des moteurs auxquels nous dédions cet article.

Techniquement, les **moteurs sans balais** ou **BLDC** (Brushless DC electric motor) sont des **moteurs synchrones alimentés par une source de courant continu** au **moyen** d'un **onduleur qui produit un signal alternatif**. Les moteurs BLDC présentent plusieurs avantages par rapport aux moteurs à balais, ils fonctionnent aussi bien en courant continu qu'alternatif.

Par rapport aux moteurs continus, l'**absence de balais**, et donc leur action mécanique sur le collecteur, **réduit les perturbations radioélectriques** émises dans l'environnement du fait de l'ouverture et de la fermeture des contacts.

De plus, le fonctionnement du collecteur implique que chaque balai entre en contact avec deux lamelles consécutives pendant une période limitée, ce qui provoque un petit court-circuit et donc une perte d'énergie, d'où une diminution du rendement.

La présence des balais provoque également une augmentation de la masse et de l'encombrement à puissance égale en sortie d'arbre. De plus, les frottements entre les balais et le collecteur provoquent une usure de ces derniers et donc un remplacement périodique des balais.

Cependant, tous les avantages énumérés ci-dessus ne sont pas gratuits et le prix à payer pour ceux-ci est la nécessité d'une carte électronique qui soit capable de piloter et d'alimenter correctement les bobines du moteur selon des séquences particulières pour permettre la rotation de l'axe.

En fait, les moteurs à balais fonctionnent simplement en leur appliquant une tension continue ou alternative, alors que les **moteurs sans balais ont besoin de trois tensions déphasées** de manière appropriée pour générer le champ magnétique du stator en rotation afin de faire tourner le rotor.

Dans le cas du moteur à balais à courant continu, le **champ tournant est généré en inversant la tension** par le biais d'une disposition appropriée des lamelles du collecteur et dans le moteur à balais à courant alternatif avec un collecteur en anneau dans lequel un champ tournant est obtenu du fait de l'alimentation en courant alternatif.

Dans les moteurs asynchrones, le champ du stator qui est tournant est obtenu en réalisant deux enroulements intercalés et en alimentant l'un avec le courant alternatif direct et l'autre traversé par un condensateur afin d'obtenir un déphasage.

Revenons donc à notre moteur « **brushless** », qui est physiquement **formé par un ou plusieurs aimants permanents placés sur le rotor et des bobines d'induction disposées sur le stator**.

Les deux sont des multiples de trois, c'est-à-dire qu'**il y en a autant qu'il existe des tensions nécessaires au fonctionnement** (le moteur peut donc être considéré comme un moteur triphasé). Le **moteur est synchrone** car le **champ magnétique généré par le stator détermine la fréquence de rotation du rotor**.

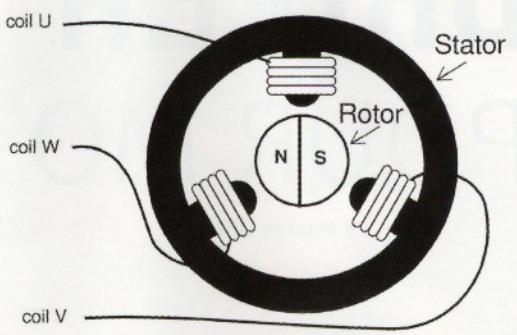
Dans ces pages, nous allons réaliser un driver (c'est-à-dire un circuit de commande) simple pour moteurs sans balais basé sur le microcontrôleur ATmega8 d'Atmel. La carte sera capable de gérer des moteurs de petite puissance.

Notre projet

L'objectif de cette carte driver est de fournir un outil pouvant être facilement intégré à vos projets et pouvant être personnalisé afin de répondre à vos besoins.

Le code source fourni est développé en langage C et compilé avec « avr-gcc ». Il a été écrit de manière à pouvoir être adapté sans effort aux autres microcontrôleurs de la même série ou même à ceux de fabricants différents, par exemple les microcontrôleurs ATmega328 ou même STM32 de chez STMictroelectronics.

Figure 1 : schématisation d'un moteur BLDC.



En général, pour contrôler un système, nous pouvons utiliser deux modes :

- le **contrôle en boucle ouverte** (open-loop), c'est-à-dire sans contre-réaction ;
- le **contrôle en boucle fermée** (closed-loop),c'est-à-dire **avec une contre-réaction**.

Dans le premier modèle, la commande du système est indépendante de la mesure du système. Au contraire, dans le deuxième modèle, il existe un **signal de retour qui informe le contrôleur de l'état du système** afin qu'il puisse utiliser les informations détectées pour **adapter la commande**.

La carte driver proposée est du **type à boucle fermée**, c'est-à-dire qu'elle reçoit certains paramètres liés au moteur pendant la phase de contrôle.

En particulier, la carte driver doit connaître la position du rotor.

Pour **connaître l'état du rotor** pour un driver de type BLDC, il existe deux techniques principales.

L'utilisation de **capteurs à effet Hall** placés sur le moteur ou la **mesure de la force électromotrice induite dans les bobines** (Back Electromotive Force, Back EMF, ou BEMF).

Dans le premier cas, il est nécessaire que trois capteurs à effet Hall soient disposés sur le moteur.

Dans le second cas, cependant, il suffit que la carte driver dispose des circuits adéquats pour mesurer la force électromotrice induite.

Figure 2 : séquence de fonctionnement du moteur.

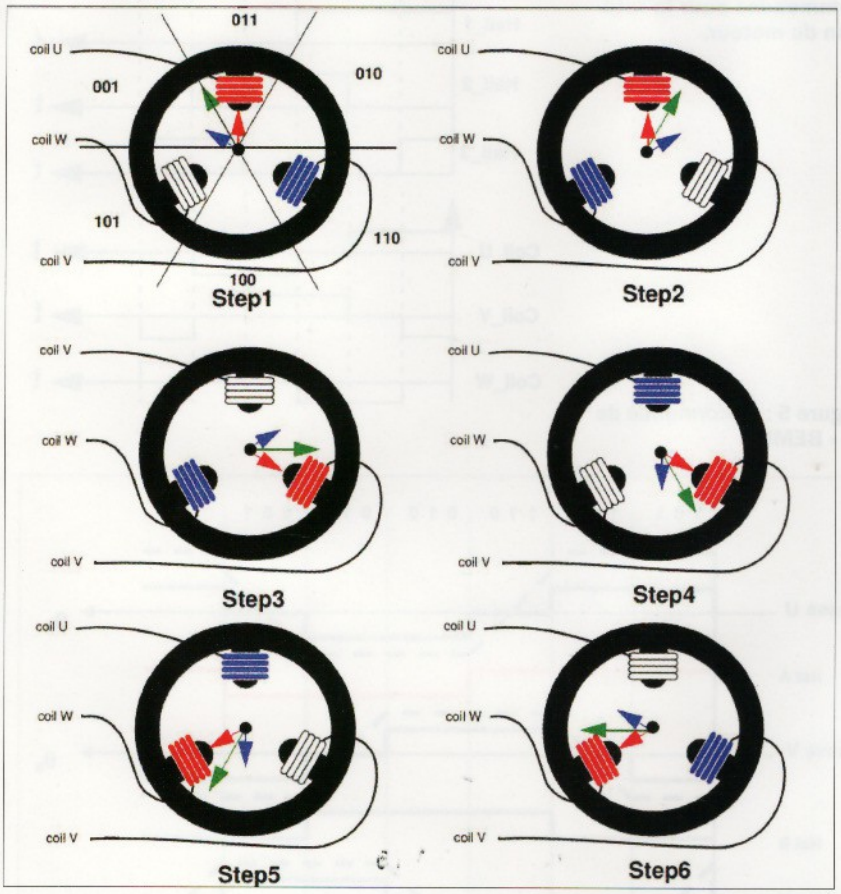
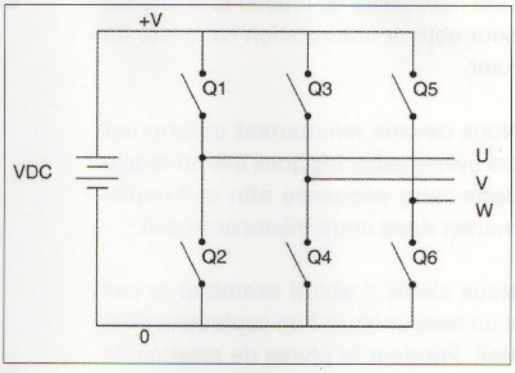


Figure 3 : section d'alimentation d'un driver pour moteur BLDC.



Dans ce dernier cas, la technique de pilotage du moteur devient légèrement plus compliquée, mais la nécessité de composants supplémentaires, tels que les capteurs à effet Hall, est éliminée.

Les **moteurs sans balais les plus courants ont trois phases** et, comme nous l'avons déjà mentionné, un moteur BLDC est composé d'un certain nombre d'aimants permanents disposés sur le rotor et d'un certain nombre de bobines placées sur le stator.

Comme il s'agit de **moteurs triphasés**, le **nombre de bobines** du stator, comme celui des **aimants permanent** sont des **multiples de trois**.

Selon le type de moteur et son domaine d'application, il est construit avec plus ou moins d'aimants et d'enroulements sur le stator.

Généralement, avec le même courant en entrée, plus le nombre d'enroulements (bobines) du stator est grand, plus le couple est important. La figure 1 montre un exemple de ce type de moteur.

Comment fonctionne un moteur BLDC ?

Pour comprendre cela, considérons le plus simple des moteurs BLDC triphasés, c'est-à-dire celui composé de trois bobines et d'un aimant permanent. Nous appelons les **trois bobines U**, **V** et **W**.

Examinons maintenant le moteur pour comprendre quelle séquence d'activation nous devons générer sur les trois bobines pour produire un mouvement de rotation. **Pour faire tourner un moteur BLDC**, il est **nécessaire d'alimenter les bobines en suivant une séquence particulière**.

En alimentant les bobines dans l'ordre inverse, le rotor tourne dans le sens opposé.

Nous savons qu'en faisant **circuler un courant électrique dans une bobine**, **il se crée un champ magnétique** qui, selon le sens, **attire ou repousse** l'aimant du stator. Pour clarifier le fonctionnement de notre moteur, reportez-vous à la figure 2.

Au cours de l'étape 1, la bobine W n'est pas connectée, la bobine U est alimentée de manière à produire un champ magnétique qui est représenté en rouge. La bobine V, à l'inverse, génère un champ magnétique (en bleu), la somme vectorielle des deux champs est représentée par la flèche verte. Cela agit sur l'aimant placé sur le rotor, provoquant ainsi un mouvement de rotation.

Pour permettre la prochaine étape de la rotation, les bobines U et W seront alimentées comme indiqué en figure 2 à l'étape 2 (step 2) et ainsi de suite. Le microcontrôleur aura donc pour tâche de déterminer le moment exact où les bobines doivent être alimentées.

Dans les moteurs BLDC, les bobines sont enroulées de telle sorte que trois connexions suffisent pour leur permettre de fonctionner. Si cette technique de bobinage est utilisée, la section puissance d'un driver BLDC peut être schématisée comme illustré en figure 3.

Dans le circuit que nous allons réaliser, les commutateurs Q1, Q2, Q3, Q4, Q5, Q6 seront remplacés par des transistors MOSFET de type canal N, de manière à former une branche d'un montage en pont en « H » (H-Bridge).

La séquence de rotation complète est celle visualisée dans le graphique de la figure 4. Comme vous pouvez le constater, il s'agit d'une séquence divisée en six étapes. **Chaque étape correspond donc à une rotation de 60 degrés**, soit au total 360 degrés (1 tour complet).

Précisons cependant que dans le cas des moteurs ayant plus de trois pôles, il sera nécessaire de répéter la séquence pour obtenir une rotation complète du rotor.

Nous devons maintenant déterminer les évènements logiques mis en œuvre dans cette séquence afin de l'implémenter dans notre microcontrôleur.

Nous allons d'abord examiner le cas d'un moteur équipé de capteurs à effet Hall. Pendant la phase de rotation, le microcontrôleur doit être capable de lire l'état des capteurs à effet Hall afin de déterminer la position du rotor, puis de sélectionner l'étape à produire pour effectuer la rotation.

La lecture de l'état du capteur est effectuée via une broche configurée en entrée digitale.

Si, au contraire, le **moteur n'est pas équipé de capteurs à effet Hall**, il sera nécessaire d'utiliser une technique plus fine pour déterminer la position du rotor. Cette technique utilisera la **force électromotrice induite dans les bobines** après la rotation, c'est-à-dire la mesure de la « BEMF ».

Le principe de cette méthode repose sur l'application de la loi physique selon laquelle, **lorsqu'un moteur tourne**, d**ans chaque bobine** du stator **est généré une tension de sens opposé** à celle appliquée aux bobines.

Dans un premier test, pour déterminer la qualité de la méthode « BEMF », nous avons utilisé un moteur également équipé de capteurs à effet Hall.

En mesurant les tensions et l'état des capteurs à effet Hall par rapport à la position du rotor et à l'alimentation des bobines, nous avons obtenu les données représentées en figure 5.

Sur ce graphique, vous pouvez constater que la « BEMF » change de polarité lorsque les capteurs à effet Hall changent d'état. Nous devons donc nous préoccuper de trouver un moyen de lire avec notre microcontrôleur le moment où la « BEMF » coupe le zéro. Ce moment est appelé « **passage par zéro** » (zero-crossing).

Figure 4 : séquence de commandes pour la rotation du moteur.

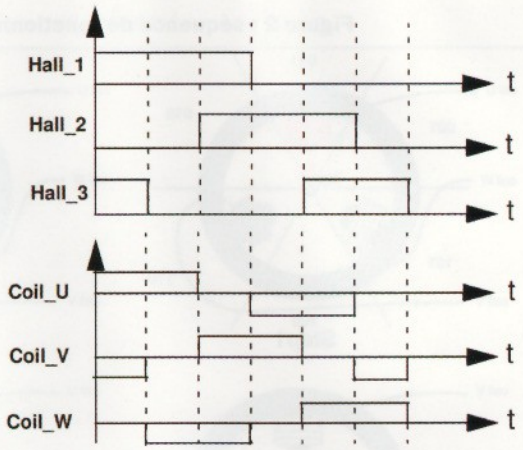
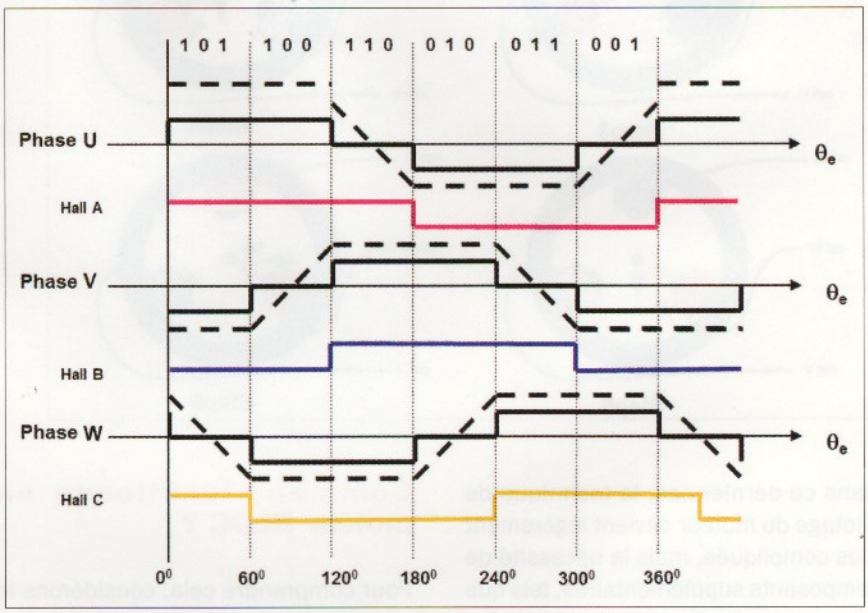


Figure 5 : performance de la « BEMF ».



Nous devons donc mesurer la tension présente sur les trois pôles du moteur.

Bien sûr, nous devons utiliser un diviseur de tension pour pouvoir ensuite appliquer le signal sur l'entrée du microcontrôleur, car les ports analogiques de notre microcontrôleur n'acceptent généralement pas des tensions supérieures à 5 V, tandis que les moteurs sont alimentés avec une tension, par exemple, de 12 V.

À première vue, nous pourrions penser que le convertisseur analogique/digital (ADC) présent dans le microcontrôleur est un bon moyen. Cependant, même si cela est réalisable, nous savons que lire une valeur dans l'ADC nécessite un cycle de calcul pour produire un résultat valide, et que le nombre de cycles dépend de l'architecture choisie.

Nous devons donc trouver une **méthode plus rapide** pour obtenir cette information. Nous pouvons utiliser un **comparateur ADC**, lui aussi présent à l'intérieur de l'ATmega.

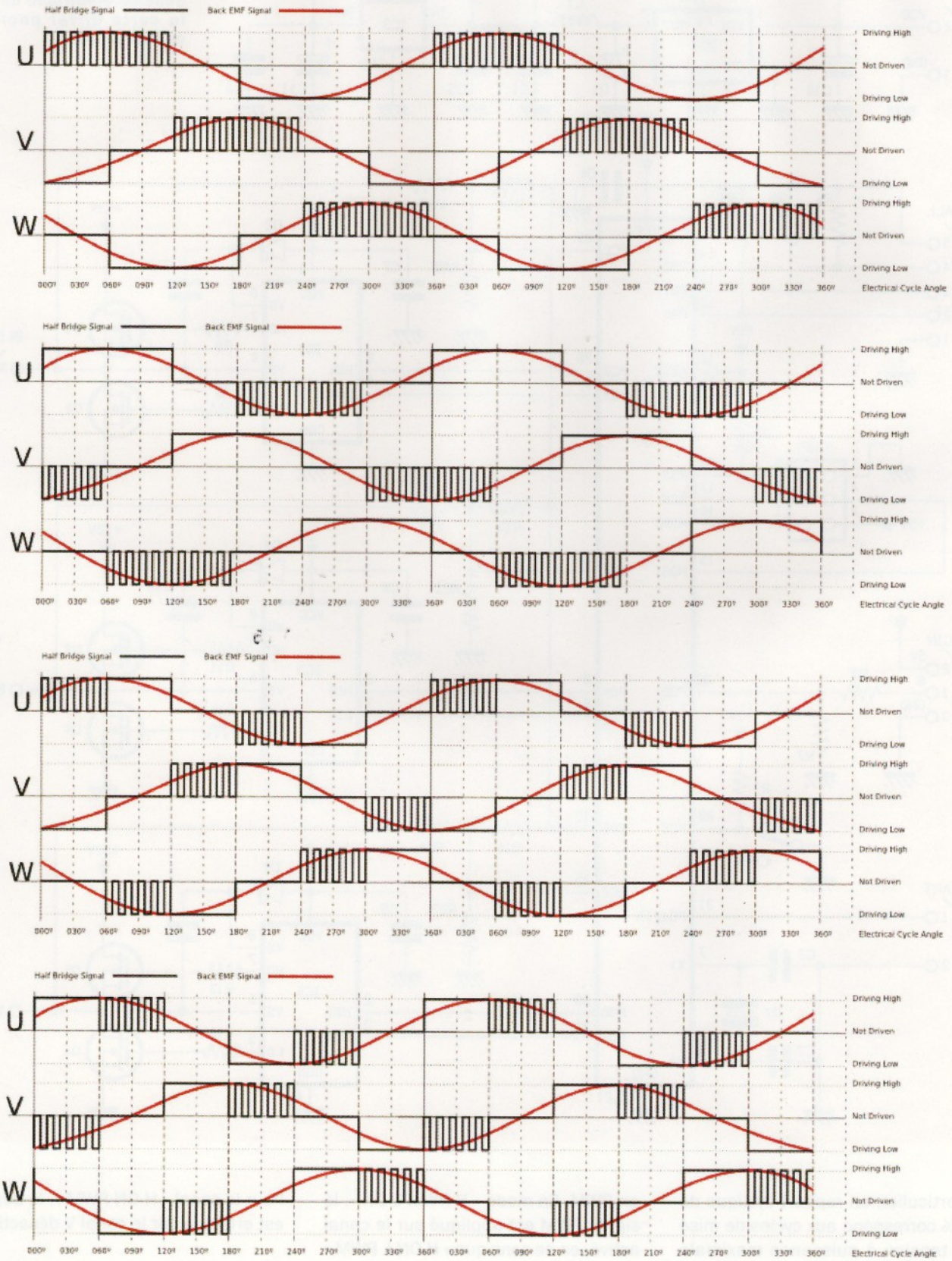
Avec cette technique, nous obtiendrons un résultat avec un seul cycle de lecture.

Si nous considérons maintenant la séquence examinée en figure 5, au moment de la première étape nous devons comparer la sortie avec celle du MOSFET W, c'est-à-dire l'étape 2. Pour l'étape suivante, nous devons comparer la sortie à celle du MOSFET V.

Nous devons maintenant comprendre comment commuter l'allumage et l'extinction des pôles du moteur.

La méthode d'activation la plus courante pour les bobines est un schéma simple

Figure 6 : cycles PWM utilisés dans le circuit : de haut en bas « H PWM L ON », « H ON L PWM », « H PWM ON » et « H ON PWM ».



ON/OFF qui dure tout le demi-cycle de rotation. Cependant, cette méthode n'est pas efficace car le MOSFET reste passant pendant toute la phase de rotation, il en résulte une surchauffe des transistors de puissance. L'alternative proposée consiste à utiliser des schémas d'allumage et d'extinction par **modulation en largeur d'impulsion** (Pulse With Modulation ou PWM).

De cette manière, nous déterminerons l'énergie à fournir aux bobines en définissant le **rapport cyclique** (duty-cycle) de la modulation en largeur d'impulsion.

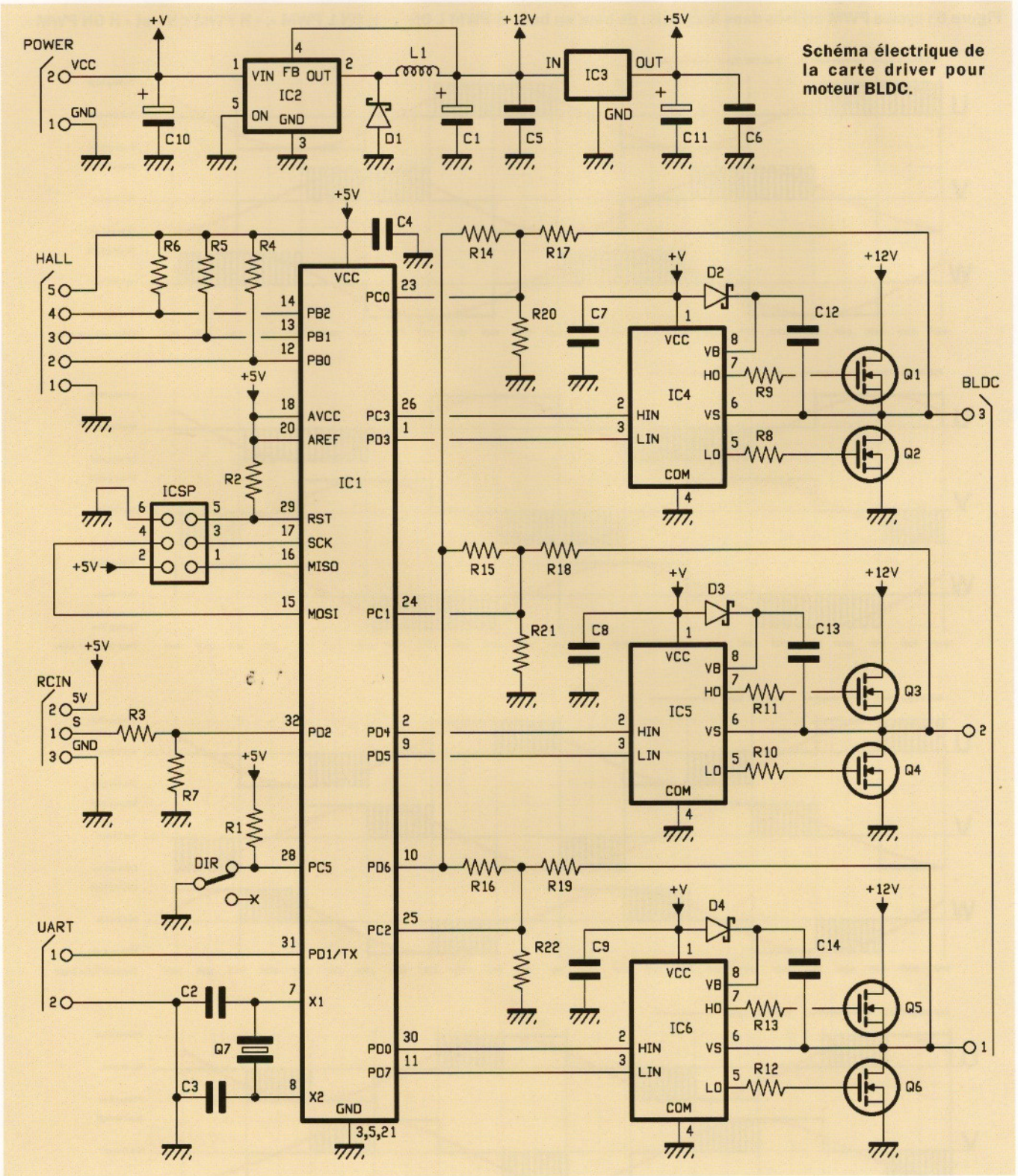


Schéma électrique de la carte driver pour moteur BLDC.

En particulier, un rapport cyclique de 100 % correspond aux cycles de mise sous tension à puissance maximale et d'extinction. Les quatre schémas PWM utilisés sont illustrés dans les graphiques de la figure 6.

Comme vous pouvez le constater, les MOSFET sont maintenant commandés en PWM. En mode « H PWM L ON », le signal PWM est appliqué sur le canal activé, en ce sens que « H ON L PWM » est appliqué sur le canal bloqué.

En mode « H PWM ON », le signal PWM est appliqué sur les canaux U, W et V tous les 60 degrés en partant du canal U activé.

Pour le canal « H ON PWM », le départ est effectué sur le canal V désactivé.

Schéma électrique

Analysons maintenant le circuit proposé, dont vous apercevez le schéma électrique dans ces pages.

Le cœur du montage est le microcontrôleur **ATmega8**.

La section alimentation utilise un **régulateur à découpage** (commutation) de la série LM2596 qui génère une tension de 12 VDC afin de piloter les trois circuits intégrés **IR2101**.

Nous trouvons également un régulateur AMS1117 utilisé pour générer la tension stabilisée qui alimente le microcontrôleur. Les transistors MOSFET de commande du moteur sont directement reliés à la tension d'entrée d'alimentation (connecteur « POWER »), qui peut atteindre 35 V.

Le microcontrôleur est connecté aux MOSFET de puissance par l'intermédiaire de trois circuits intégrés IR2101, qui fonctionnent comme des circuits d'isolement (HIGH/LOW) entre les grilles des MOSFET et les sorties de l'ATmega8.

Les sorties du moteur retournent alors vers les entrées analogiques de l'ATmega à travers trois diviseurs de tension. Ces derniers permettent la mesure de la « BEMF » pour un moteur de type sans balais et ne disposant pas de capteur à effet Hall.

Trois broches du microcontrôleur sont également configurées en entrées digitales avec trois résistances de tirage afin de lire l'état des capteurs à effet Hall du moteur.

L'horloge cadencée à 16 MHz est générée par un oscillateur à quartz Q7 afin de stabiliser la fréquence de travail.

Nous avons également testé le fonctionnement avec un quartz de 8 Mhz sans rencontrer de problèmes particuliers.

Notez que l'entrée RCIN, que nous voulions, car le contrôleur décrit ici permet à l'aide de ses trois broches de piloter des moteurs sans balais à partir des signaux issus de servocommandes, qui respectent le standard des radiocommandes pour modélisme.

Le signal correspondant est de type PWM qui est un standard pour pratiquement tous les servomoteurs du commerce.

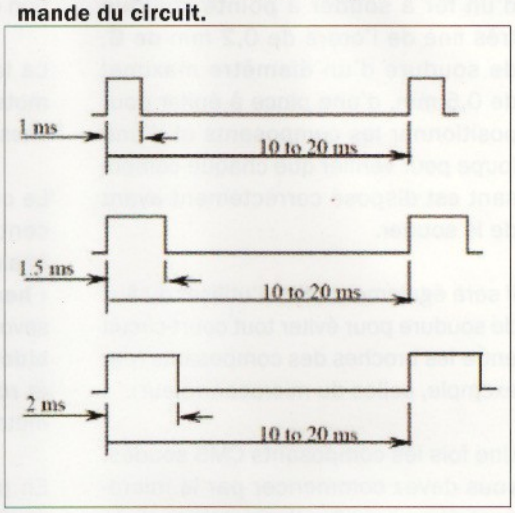
En particulier, le signal RC qui est à des niveaux TTL avec une fréquence de 50 Hz, celui-ci génère donc une impulsion toutes les 20 ms.

Le rapport cyclique de ce signal est tel que la largeur de l'impulsion dans la période varie de 1 ms à 2 ms.

En fonction de sa durée, le servomoteur fonctionnera dans un sens ou dans l'autre.

Lorsque le signal est de 1,5 ms, le servomoteur reste immobile dans sa position centrale.

Figure 7 : synchronisation du signal de commande du circuit.



Le contrôleur proposé ici implémente une lecture pour ce type de signal en utilisant un TIMER et une interruption du microcontrôleur et produit en sortie une variable dont la valeur est comprise entre 0 et 100. Nous allons utiliser cette valeur pour régler la vitesse de rotation de notre moteur (voir la figure 7).

En fait, ce même signal des servocommandes est utilisé dans le contrôleur de vitesse ou ESC (Electronic Speed Controller) afin de faire varier la vitesse du moteur, en fonction de la largeur de l'impulsion, de zéro au maximum.

Réalisation pratique

Passons maintenant à la partie pratique du projet, à savoir la construction de la carte contrôleur, pour laquelle un circuit imprimé double face est nécessaire. Vous pouvez télécharger sur notre site, dans le sommaire détaillé de la revue, les typons du circuit imprimé.

Nous mettons à disposition le fichier « avr_bldc_04.brd » qui peut être modifié avec le logiciel Eagle. Vous pourrez éventuellement générer les fichiers gerber pour une fabrication professionnelle.

Nous avons déjà expliqué dans notre revue la procédure pour générer des fichiers gerber à partir de Eagle. Reportez-vous par exemple au numéro 148 (pages 41 et 42).

Une fois le circuit imprimé gravé et percé, commencez d'abord par souder les composants CMS. Ils nécessitent une grande attention.

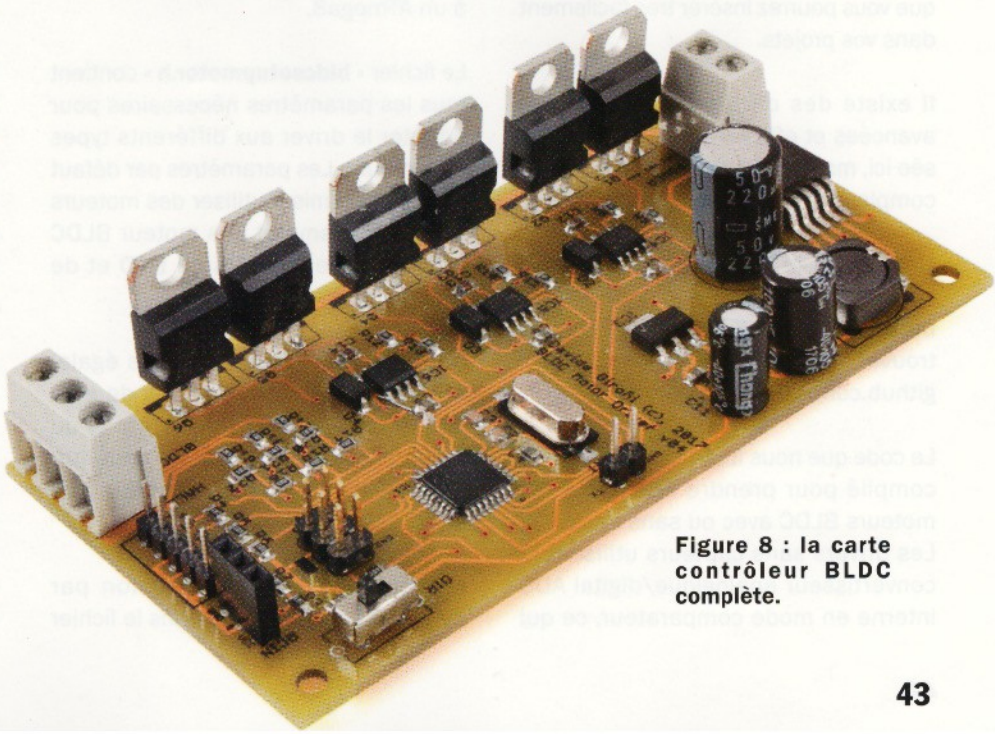


Figure 8 : la carte contrôleur BLDC complète.

Pour le montage, vous aurez besoin d'un fer à souder à pointe conique très fine de l'ordre de 0,2 mm de Ø, de soudure d'un diamètre maximal de 0,5 mm, d'une pince à épiler pour positionner les composants et d'une loupe pour vérifier que chaque composant est disposé correctement avant de le souder.

Il sera également utile d'utiliser du flux de soudure pour éviter tout court-circuit entre les broches des composants (par exemple, celles du microcontrôleur).

Une fois les composants CMS soudés, vous devez commencer par le microcontrôleur (vérifier l'orientation de la broche 1 qui correspond à une marque sur le boîtier), soudez les condensateurs électrolytiques en vérifiant leur orientation, l'interrupteur à glissière, les barrettes et les borniers.

Terminez par les transistors MOSFET, leur côté métallique est orienté vers l'extérieur du circuit imprimé. Une fois tous les composants soudés, le circuit doit être identique à celui visible en figure 8.

Le firmware

Nous pouvons maintenant passer à l'analyse du code source du programme. Nous rappelons que le programme, disponible en téléchargement, est un point de départ à partir duquel vous pouvez créer une carte driver personnalisée que vous pourrez insérer très facilement dans vos projets.

Il existe des cartes beaucoup plus avancées et efficaces que celle proposée ici, mais leur code source est assez complexe et difficile à importer.

Nous nous référons par exemple au célèbre driver pour ESC écrit en assembleur par SimonK, que vous pouvez trouver à l'adresse suivante : https://github.com/sim-/tgy.

Le code que nous fournissons peut être compilé pour prendre en charge les moteurs BLDC avec ou sans capteurs. Les drivers sans capteurs utilisent le convertisseur analogique/digital ADC interne en mode comparateur, ce qui permet une lecture rapide de la position du rotor.

La lecture d'une entrée d'un servomoteur RC permet la sélection de la vitesse du moteur.

Le code utilise la librairie « **bldc** » et contient un exemple de programme (main.c) pour la commande. Les fichiers « **header** » de la librairie « **bldc.h** », à savoir **bldcsetup.h**, **bldcsetupboard.h**, **bldcsetupmicro.h**, **bldcsetupmotor.h** et **rcin2.h** contiennent tous les paramètres configurables.

En particulier, à partir du fichier de configuration principal « bldcsetup.h », il est possible de sélectionner le mode de fonctionnement du driver pour les moteurs sans capteur ou avec capteur, le signal de sortie PWM pour les MOSFET, ainsi que d'autres paramètres supplémentaires tels que le « watchdog » ou le type de démarrage (startup).

Le fichier « **bldcsetupboard.h** » configure tous les ports d'entrée et de sortie liés à la carte matérielle, nous pouvons adapter le driver pour pouvoir l'utiliser avec différents systèmes, par exemple un ESC du commerce ou d'autres cartes de développement, en modifiant ce fichier.

Si nous voulons adapter le code source à un autre microcontrôleur, le fichier « **bldcsetupmicro.h** » contient les principales définitions de l'adaptation. Dans notre cas, l'ensemble se réfère à un ATmega8.

Le fichier « **bldcsetupmotor.h** » contient tous les paramètres nécessaires pour adapter le driver aux différents types de moteurs. Les paramètres par défaut nous ont permis d'utiliser des moteurs pour modélisme et des moteur BLDC provenant de lecteurs de DVD et de disques durs.

Dans ce fichier, nous pouvons également définir les paramètres de gestion de la vitesse et de démarrage du moteur. Le fichier « rcin2.h » peut être utilisé pour configurer la lecture du signal de la vitesse d'un servo RC.

La séquence de commutation par défaut est implémentée dans le fichier « **bldc.h** », qui contient également toutes les commandes de gestion de la librairie.

Les fonctions principales sont les suivantes : **initialisation** « bldc_init » ; **sélection de la direction** « bldc_setdirection », **démarrage** « bldc_start » ; **arrêt** « bldc_stop » ; **vitesse de rotation** « bldc_setspeed ». La librairie complète utilise trois TIMERS physiques du microcontrôleur.

Le TIMER utilisé pour la lecture du signal RC peut éventuellement être omis si le signal de la vitesse provient d'un autre endroit comme le bus I2C, l'UART ou une partie du code contenu dans le programme.

Liste des composants

R1..... 10 kΩ boîtier CMS 0805
R2..... 10 kΩ boîtier CMS 0805
R3..... 10 kΩ boîtier CMS 0805
R4..... 10 kΩ boîtier CMS 0805
R5..... 10 kΩ boîtier CMS 0805
R6..... 10 kΩ boîtier CMS 0805
R7..... 47 kΩ boîtier CMS 0805
R8..... 10 Ω boîtier CMS 0805
R9..... 10 Ω boîtier CMS 0805
R10.... 10 Ω boîtier CMS 0805
R11.... 10 Ω boîtier CMS 0805
R12.... 10 Ω boîtier CMS 0805
R13.... 10 Ω boîtier CMS 0805
R14.... 47 kΩ boîtier CMS 0805
R15.... 47 kΩ boîtier CMS 0805
R16.... 47 kΩ boîtier CMS 0805
R17.... 47 kΩ boîtier CMS 0805
R18.... 47 kΩ boîtier CMS 0805
R19.... 47 kΩ boîtier CMS 0805
R20.... 10 kΩ boîtier CMS 0805
R21.... 10 kΩ boîtier CMS 0805
R22.... 10 kΩ boîtier CMS 0805

C1..... 100 µF/50 V électrolytique
C2..... 22 pF céramique boîtier CMS 0805
C3..... 22 pF céramique boîtier CMS 0805
C4..... 100 nF céramique boîtier CMS 0805
C5..... 100 nF céramique boîtier CMS 0805
C6..... 100 nF céramique boîtier CMS 0805
C7..... 100 nF céramique boîtier CMS 0805
C8..... 100 nF céramique boîtier CMS 0805

C9 ..... 100 nF céramique boîtier CMS 0805
C10 .... 220 µF/50 V électrolytique
C11 .... 470 µF/16 V électrolytique
C12 .... 1 µF céramique boîtier CMS 0805
C13 .... 1 µF céramique boîtier CMS 0805
C14 .... 1 µF céramique boîtier CMS 0805

Q1 ..... IRF640
Q2 ..... IRF640
Q3 ..... IRF640
Q4 ..... IRF640
Q5 ..... IRF640
Q6 ..... IRF640
Q7 ..... quartz 16 MHz

D1 ..... 1N5819
D2 ..... 1N5819
D3 ..... 1N5819
D4 ..... 1N5819

IC1 ..... ATMEGA8-16AU
IC2 ..... LMS2596S-12
IC3 ..... REG1117-5.0
IC4 ..... IR2101SMD
IC5 ..... IR2101SMD
IC6 ..... IR2101SMD

L1 ..... bobine 33 µH
DIR ..... interrupteur à glissière

Divers :
Bornier 2 pôles au pas de 5,08 mm
Bornier 3 pôles au pas de 5,08 mm
Barrette mâle 2 pôles
Barrette mâle 3 pôles
Barrette mâle 5 pôles
Barrette mâle 2 * 3 pôles

Plan de montage

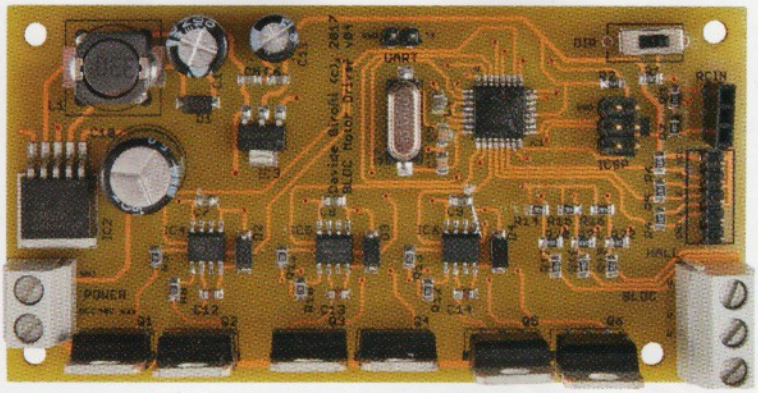


Photo de l'un de nos prototypes de la carte driver pour moteur BLDC.

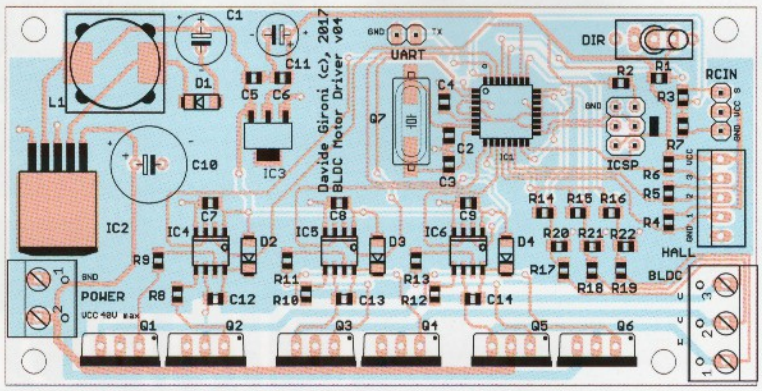


Schéma d'implantation des composants de la carte driver pour moteur BLDC.

Les deux autres TIMER gèrent la commande PWM envoyée aux MOSFET et la logique de génération du signal.

Le TIMER de gestion des signaux PWM n'est pas très complexe, il active ou désactive simplement les portes logiques du microcontrôleur en fonction de l'étape sélectionnée par le TIMER principal et du modèle de génération de la commande PWM sélectionnée.

Le dernier TIMER gère la logique de fonctionnement de la librairie.

Pour simplifier, ce TIMER contrôle en permanence l'état du rotor, en utilisant pour cette fonction la lecture des ports digitaux dans le cas des moteurs avec capteurs ou la configuration du comparateur ADC dans le cas des moteurs sans capteur.

La lecture de la position du rotor est modifiée en permanence en fonction de l'étape de commutation.

Lorsque le mode de démarrage de rotation du moteur est activé, le TIMER principal génère le cycle de démarrage.

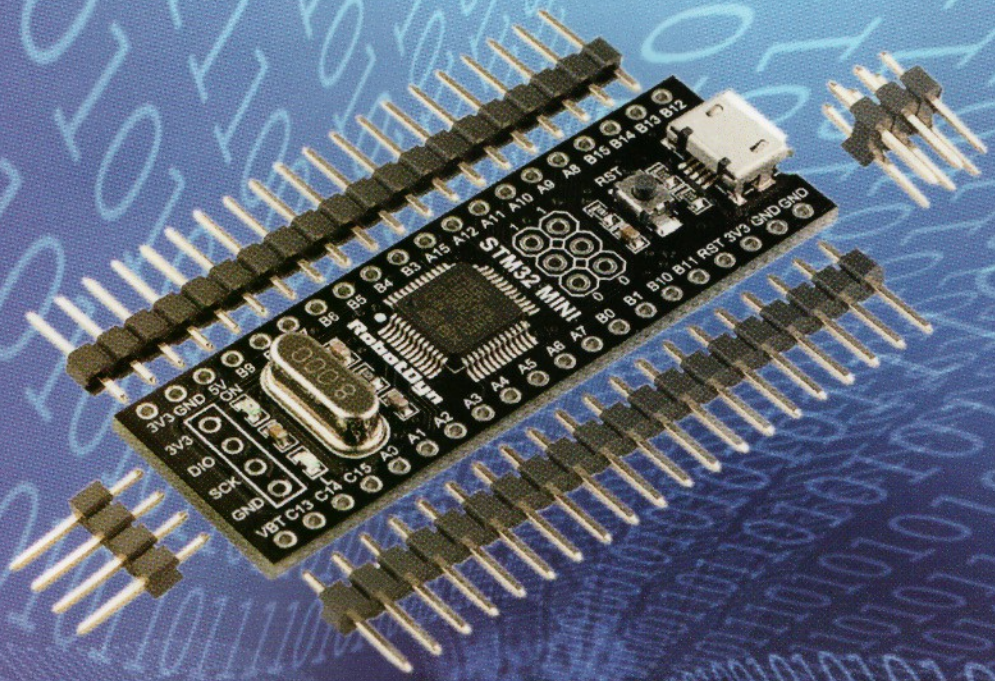
Conclusion

Dans cet article, nous avons analysé les bases théoriques du fonctionnement des moteurs sans balais en définissant deux grandes familles existantes, à savoir ceux équipés de capteurs à effet Hall et ceux sans capteurs, avec un contrôle de contre-réaction.

Nous avons implémenté la logique de fonctionnement d'un moteur dans un circuit de contrôle simple, qui peut être programmé afin de constituer une carte de développement pour des applications de commandes de moteurs sans balais.

Le programme a été écrit en langage C et compilé pour le microcontrôleur ATmega8, qui est le cœur de notre montage.

Ce firmware est un point de départ pour ceux qui veulent le modifier afin de l'adapter à leurs besoins.



Cet article didactique traite tout particulièrement du bootloader, un sujet mal connu de la plupart des utilisateurs de microcontrôleurs. Nous allons aborder les séquences de démarrage du firmware et de la programmation associée.

L'IMPORTANCE DU BOOTLOADER

de Piero Boccadoro

Dans la programmation des microcontrôleurs, il existe des codes, des routines et des « framework » de référence qui permettent l'exécution séquentielle d'instructions qui constituent le fonctionnement général du système, une fois son développement terminé.

Mais, pour programmer un microcontrôleur, il est nécessaire d'effectuer des opérations selon une séquence bien précise et dans un certain ordre, de manière à agir sur des registres spécifiques, dont les paramètres corrects permettent l'exécution effective du programme.

Ce que nous proposons dans cet article est une étude sur la programmation des microcontrôleurs, en mettant l'accent sur les routines nécessaires.

Nous allons décrire le bootloader et les séquences de démarrage afin de comprendre le fonctionnement des microcontrôleurs.

Et pour bien comprendre chaque aspect, nous allons donner des exemples en analysant des cas particulièrement utiles et significatifs, tels que les microcontrôleurs de la famille ST Microelectronics, ou encore le fonctionnement particulier du bootloader dans les microcontrôleurs ARM Cortex ou ceux de la famille XMC1000 de chez Infineon.

La comparaison montrera les aspects communs et les différences dont la compréhension est fondamentale pour pouvoir programmer pratiquement n'importe quel microcontrôleur.

Enfin, nous tenterons de déterminer la meilleure solution à adopter pour débuter, en fonction de la simplicité du système, des pratiques et des logiciels.

Le bootloader

En informatique, le **bootloader est le programme** qui, dans la phase de démarrage (boot) de l'ordinateur, **charge le noyau** (kernel) du système d'exploitation de la mémoire « secondaire » (c'est-à-dire du disque dur ou d'un support de stockage de masse) vers la mémoire « principale », qui est dans la plupart des cas la mémoire vive ou RAM. Cela permet l'exécution par le processeur et le démarrage du système qui s'ensuit.

Le terme dérive du fait que le processus de démarrage d'un ordinateur s'appelle un « bootstrap », expression provenant d'un dicton anglais. Dans le cas où plusieurs systèmes d'exploitation sont installés sur un même ordinateur, le chargement du bootloader est précédé de la sélection du système d'exploitation souhaité par l'utilisateur, via un gestionnaire de démarrage spécial (une solution bien connue des utilisateurs de Linux est le programme GRUB).

La **fonction fondamentale du bootloader consiste donc à charger et à exécuter le noyau d'un système d'exploitation**, ainsi que des processus et des services secondaires. Dans la plupart des cas, cela nécessite un accès à la mémoire de masse, afin de lire le noyau du système d'exploitation, ainsi que d'autres fichiers.

Dans les cas les plus simples, le bootloader contient l'adresse des blocs de mémoire dans lesquels sont stockés les fichiers à charger et doit donc être mis à jour si ces fichiers sont modifiés. L'accès au disque se fait souvent par des fonctions disponibles dans le « firmware » (le BIOS dans le cas des systèmes PC compatibles IBM).

Certains bootloader ont la capacité d'interpréter un ou plusieurs fichiers système pour trouver les fichiers à télécharger. Dans ce cas, ils peuvent également charger un fichier de configuration à partir du disque dur ou permettre à un utilisateur (expérimenté) d'explorer le disque à la recherche des fichiers à charger pour un lancement sélectif de services spécifiques.

De plus, certains bootloader peuvent utiliser des fonctionnalités disponibles sur certaines cartes réseau (généralement Ethernet) pour télécharger un noyau à partir du réseau.

Il existe également des fonctionnalités supplémentaires présentes dans certains bootloader :

- de nombreux noyaux prennent en charge la possibilité de recevoir des paramètres d'amorçage pour configurer leur comportement. Le bootloader se charge alors de les transmettre au noyau et, dans certains cas, permet à l'utilisateur d'éditer certaines fonctions ;
- certains noyaux s'attendent à ce que d'autres fichiers soient disponibles au démarrage. Par exemple, dans les systèmes de type Unix, l'utilisation d'« initrd » (pour « INITial RamDisk », c'est une image du système d'exploitation minimal initialisé au démarrage du système) est courante. Leur chargement est géré par le bootloader ;
- un bootloader peut afficher à l'utilisateur un menu contenant plusieurs noyaux à charger, avec les paramètres associés, permettant ainsi de choisir le système d'exploitation à démarrer ;
- un bootloader peut demander un mot de passe pour permettre au système de démarrer.

Le bootloader est, dans de nombreux cas, installé dans une position spécifique d'un périphérique de mémoire de masse, généralement dans le premier bloc du premier disque (disque primaire, en fonction de l'ordre dans lequel les périphériques sont connectés à l'ordinateur), à partir duquel le firmware est chargé.

Par exemple, dans l'architecture d'un PC compatible IBM, le bootloader peut être stocké dans le MBR (Master Boot Record) du disque d'amorçage ou dans le premier secteur de la partition hébergeant le système d'exploitation (dans ce cas, il doit être appelé par un gestionnaire de démarrage ou « boot manager » installé dans le MBR). Il peut être situé dans le premier secteur d'un volume amorçable, tel qu'un environnement de récupération (par exemple, Windows Recovery Environment).

Il existe certaines limitations imposées par la technique et par les fabricants de matériel et de programmes qui empêchent de nombreux bootloader modernes de fonctionner en utilisant le démarrage dit à « deux étapes » : étape 1 et étape 2. Elles peuvent être décrites de la manière suivante :

- l'étape 1 correspond à ce que nous avons appelé précédemment le gestionnaire de démarrage (boot manager). Elle traite les tâches très simples pouvant être codées dans un espace relativement restreint, telles que, par exemple, la recherche du premier volume du système sur le disque de démarrage, la recherche et le chargement l'étape 2, plus complexe ;
- l'étape 2 permet le véritable démarrage du système d'exploitation. Elle est beaucoup plus sophistiquée, complexe et plus étendue que la première et consiste principalement à choisir le noyau à charger en mémoire, puis à lui donner le contrôle. Elle s'apparente à un programme exécutif, sans les nombreuses fonctionnalités des systèmes d'exploitation, telles que la multiprogrammation et la gestion des ressources.

Ces étapes correspondent à une subdivision logique qui implémente les différentes phases du démarrage.

En ce qui concerne les limitations, par exemple, il est possible de charger le dispositif de démarrage (boot) dans la mémoire principale (seulement une petite quantité d'octets).

Il se trouve positionné dans un emplacement mémoire peu commode, comme cela est le cas des systèmes « x86 », seulement les 512 premiers octets commençant à l'adresse 0x7C00 jusqu'à l'adresse 0x7DFF.

Il est utile de préciser qu'un seul bootloader est capable de charger de nombreux types de noyaux différents (et donc de nombreux systèmes d'exploitation) sur différentes partitions. Bien entendu, un seul noyau peut être chargé à la fois.

Le bootloader dans les microcontrôleurs

L'environnement de développement s'occupe de l'écriture du code qui doit être chargé et exécuté sur des dispositifs beaucoup plus petits qu'un ordinateur personnel.

Alors, quelles sont les différences entre un microprocesseur et un microcontrôleur ? Le bootloader est-il le même ?

Bien sûr que non ! Le bootloader remplit des fonctions similaires, fonctionne de manière équivalente, mais celui des microcontrôleurs est beaucoup plus simple.

En fait, **le bootloader est un programme qui s'exécute dans le microcontrôleur pour pouvoir le programmer**. Cela semble être une définition quelque peu récursive, mais il s'agit fondamentalement d'une routine qui reçoit un code (programme) d'une source externe, via un bus de communication dédié, afin de pouvoir le charger correctement dans la mémoire du microcontrôleur.

L'une des notions qu'il est important de garder à l'esprit lors de la programmation d'un microcontrôleur à l'aide de programmateurs externes dédiés est le fait qu'ils fonctionnent indépendamment du contenu déjà présent à l'intérieur du microcontrôleur.

En fait, les microcontrôleurs commencent le processus de réécriture des zones de la mémoire de manière ordonnée par rapport au contenu précédemment présent dans leur mémoire.

Si vous souhaitez utiliser un ordinateur, le programme qui communique avec le bootloader doit être correctement configuré pour envoyer un fichier de type « ***.bin** » ou « ***.hex** ». En fait, cela semble simple.

Mais que se passe-t-il s'il y a une erreur de communication ? Que se passe-t-il si les données, ou une partie de celles-ci, sont perdues ? Et si par hasard la communication est perdue, qu'advient-il des zones de la mémoire ? Et s'il y a un bug ou si le bootloader est corrompu que se passe-t-il ?

Tous ces problèmes sont gérés par le bootloader. Le bootloader commence par une première instruction visant à réinitialiser le microcontrôleur, ce qui est essentiel pour interrompre l'exécution d'un programme déjà présent.

Le bootloader communique avec l'hôte, qui répond avec le nouveau code. Dès qu'il est envoyé, celui précédemment en mémoire est écrasé.

Dans chaque code, il existe toujours un mécanisme de vérification comprenant une somme de contrôle (checksum) pour vérifier l'intégrité des données. C'est l'une des fonctionnalités associées au bootloader.

Effectivement, si l'image reçue n'est pas intacte, il doit être possible de gérer le remplacement ou la restauration de l'ancienne image.

C'est précisément pour cette raison que deux modules distincts sont généralement fournis : l'uploader et le bootloader.

En effet, le premier est responsable de l'écriture dans la mémoire de la nouvelle image uniquement si celle-ci a été reçue intacte et si elle a été choisie pour remplacer l'ancienne image par celle qui a été mise à jour.

L'uploader fait partie de l'application principale et interagit avec l'hôte afin de charger le nouveau programme. Cela nécessite des zones de mémoires distinctes qui doivent être nécessairement différentes de celles où se trouve le programme principal.

Par exemple, une mémoire EEPROM externe peut être utilisée. Elle est reliée via une interface série. Il est également possible d'utiliser un autre microcontrôleur disposant d'un espace de stockage supplémentaire. L'uploader gère simplement l'écriture du nouveau fichier reçu, mais n'est pas impliqué dans son exécution.

Dès que le téléchargement a été effectué, pour que le programme s'exécute, il est impératif que le bootloader fonctionne réellement. Ce dernier se charge en effet du contrôle de la mémoire, de la gestion des instructions et de leur exécution séquentielle.

Il est important de noter que chaque microcontrôleur diffère des autres, même au sein d'une même famille (par exemple, la famille PIC 18F). Pour cette raison, il n'est pas possible de concevoir un bootloader adapté à tous les microcontrôleurs existants.

La solution unique dite « one size fits all » n'existe pas car les commandes sont différentes, les zones de mémoire ont des adresses différentes, les instructions nécessaires sont différentes. Bien que les opérations élémentaires, d'un point de vue informatique, soient identiques (par exemple, l'arrêt du microcontrôleur, la réinitialisation et le redémarrage), des routines différentes sont nécessaires pour que chacun microcontrôleur puisse fonctionner correctement.

Les solutions à ces problématiques peuvent être celles décrites ci-après :

- **De base** : le dispositif dispose d'une ligne de communication série et souhaite se connecter à un hôte. Le bootloader est exécuté à partir de l'instruction de réinitialisation (reset) pour commencer un échange de messages afin d'écrire dans les zones de mémoire et ensuite exécuter le nouveau code. Habituellement, après un intervalle de temps de 500 ms sans communication (par exemple, aucune réponse), le bootloader cesse d'effectuer des « tentatives » et exécute le code précédemment chargé dans la zone de la mémoire. Dans ce cas, le flashage doit être effectué de nouveau ;
- **Program memory uploader** : dans ce cas, la mémoire disponible dans le microcontrôleur est divisée de telle manière qu'elle contient le programme déjà présent, la nouvelle image et le bootloader. Ce

dernier fonctionnera de manière à réinitialiser et à charger la nouvelle image dans les premières zones de la mémoire ;

- **External EEPROM Image** : comme dans le cas précédent, il est possible d'utiliser une mémoire externe en tant que copie de la nouvelle application, mais fondamentalement, cela ne change pas beaucoup par rapport au cas précédent, à moins que vous envisagiez de changer le microcontrôleur pour avoir une plus grande mémoire ou avec un type ou un nombre différent de périphériques ;
- **TCP Bootloader** : cela implique l'utilisation du protocole TCP, et donc d'une connexion via une adresse IP, pour charger l'image. Généralement, ce mode de chargement est choisi si des serveurs dédiés sont utilisés pour l'ensemble du système. Ce mode peut être utilisé lorsque plusieurs microcontrôleurs différents doivent être programmés. Dans ce cas, avant d'écrire dans le dispositif suivant, une vérification des opérations effectuées doit être exécutée. Cela pose certains problèmes techniques en termes de retards possibles dans les contrôles, qui doivent pouvoir être gérés de manière appropriée. Bien entendu, le bootloader, dans ce cas, doit pouvoir être mis à jour, ce qui implique qu'il doit contenir la pile TCP.

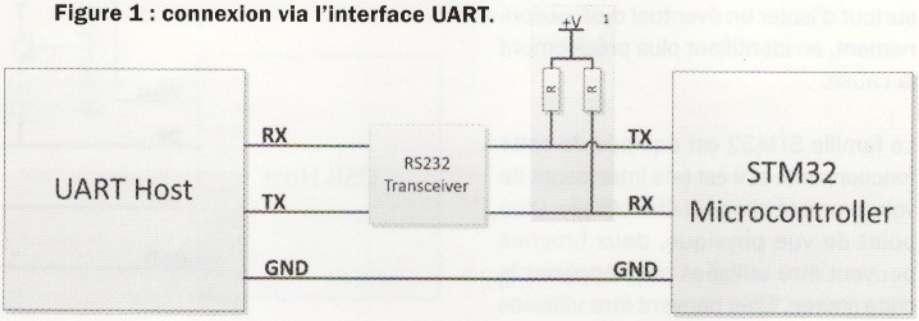
De plus, chacune de ces possibilités doit être choisie sur la base d'une évaluation précise des paramètres suivants :

- la vitesse de transmission ;
- le coût des composants ;
- la facilité d'utilisation;
- la maintenance.

Le bootloader de l'ARM Cortex

L'une des caractéristiques les plus intéressantes des microcontrôleurs de la famille STM32 de chez ST Microelectronics est la présence d'un bootloader intégré, du moins pour leur dernière révision. Pour tous ceux qui ont fait de la programmation, l'écriture du firmware et le débogage étaient des opérations fastidieuses.

Figure 1 : connexion via l'interface UART.



Soulignons que le débogage est l'une des fonctionnalités les plus utiles pour le développeur, dans les STM32 le firmware et le matériel nécessaires sont déjà présents, ils doivent simplement être configurés.

BOOT1	BOOT0	Mode de démarrage
X	0	Démarrage depuis la mémoire Flash
0	1	Démarrage à partir de la mémoire système
1	1	Démarrage à partir de la RAM

En particulier, l'interface **JTAG** représente l'un des principaux, sinon le principal standard de référence pour les communications et l'interfaçage avec les microcontrôleurs. Elle est tout d'abord utilisée pour programmer le microcontrôleur.

Une fois le code compilé, il peut être chargé à l'aide de l'interface JTAG directement dans le microcontrôleur via une séquence de contrôle spécifique en fonction de la version et de la révision du circuit intégré.

Mais les avantages de l'utilisation de cette interface concernent principalement le contrôle et la surveillance du fonctionnement, même pendant la phase de « Runtime » (exécution).

En fait, l'**interface JTAG permet un accès direct aux zones de la mémoire** du microcontrôleur de manière précise, c'est-à-dire qu'elle permet la **lecture de la mémoire octet par octet**.

En conséquence, il est également possible d'effectuer des contrôles pour vérifier l'intégrité des zones de la mémoire afin de vérifier qu'elles sont toujours accessibles et que les données qu'elles contiennent ne sont pas erronées.

Grâce à l'interface JTAG, il est également possible de vérifier les fonctions du microcontrôleur en termes de RAM, de vitesse d'accès et de communication. Cependant, pour effectuer toutes ces opérations, il est généralement nécessaire d'utiliser :

- des composants matériels supplémentaires (facultatif) ;
- un outil logiciel dédié ;
- un « header » (connecteur physique) permettant l'interfaçage physique, si la carte en question ne le prévoit pas.

Cette liste est suffisante pour tous les microcontrôleurs qui ont déjà un bootloader, ce qui n'est pas toujours le cas. En effet, il existe des microcontrôleurs qui en sont équipés et d'autres qui n'ont pas de bootloader interne. Il est donc très intéressant et utile de comprendre ce qui change dans un cas ou dans l'autre.

L'une des fonctionnalités avancées les plus intéressantes d'un bootloader intégré dans un microcontrôleur est le « **débogage en circuit** ». Cela permet d'effectuer la vérification du fonctionnement sans devoir interrompre l'exécution d'une routine.

Il est possible, en particulier, **de faire passer le microcontrôleur dans un mode de fonctionnement pas à pas**, de sorte qu'il soit possible de vérifier de manière appropriée chaque opération, chaque fonction appelée et la valeur de chaque variable.

Ce type de fonctionnalité est extrêmement utile pour tenter de vérifier le bon

fonctionnement du circuit, mais permet surtout d'isoler un éventuel dysfonctionnement, en identifiant plus précisément la cause.

La famille STM32 est équipée de cette fonctionnalité et il est très intéressant de voir comment elle peut être utilisée. D'un point de vue physique, deux broches peuvent être utilisées pour analyser le code source. Elles peuvent être utilisées dans différentes configurations.

Si nous devons exécuter le code qui réside dans la mémoire RAM, « BOOT1 » peut être maintenu à un niveau logique bas et nous pouvons ajouter un commutateur ou un cavalier à « BOOT0 ».

Si, après réinitialisation, « BOOT0 » est à un niveau logique haut, le microcontrôleur commence à utiliser le code situé dans la mémoire système. En pratique, directement à partir du bootloader. S'il se trouve à un niveau logique bas, le microcontrôleur commence par le code contenu dans la mémoire Flash.

Une fois que vous avez effectué votre choix, il est nécessaire d'étudier la réalisation des connexions. Pour ce faire, il est essentiel de comprendre tout d'abord le type d'interface que vous souhaitez utiliser. En particulier, il est possible de choisir entre l'USART, l'USB, le bus SPI, le bus I²C ou le bus CAN.

Si vous souhaitez utiliser l'interface USART pour accéder au bootloader, il est nécessaire de connecter les broches « Tx » et « Rx » à l'aide d'un câble série (voir la figure 1).

Il est essentiel de noter que des résistances de tirage (pull-up) doivent être utilisées pour que tout fonctionne correctement. Par défaut, le niveau haut ne peut jamais être atteint et, par conséquent, il ne peut jamais être lu.

Le dimensionnement correct de la résistance de tirage est un facteur parfois sous-estimé dans la phase de conception. Il faut porter une attention particulière au niveau logique et à la valeur du courant que la résistance peut garantir, afin d'éviter toute sorte de problème.

Il convient de souligner qu'il n'est pas possible de communiquer directement avec le microcontrôleur, mais qu'il est nécessaire d'utiliser un composant intermédiaire, indiqué en figure 1. Il s'agit d'un émetteur/récepteur de type RS232. Son but est d'agir en tant qu'interprète au niveau de la communication.

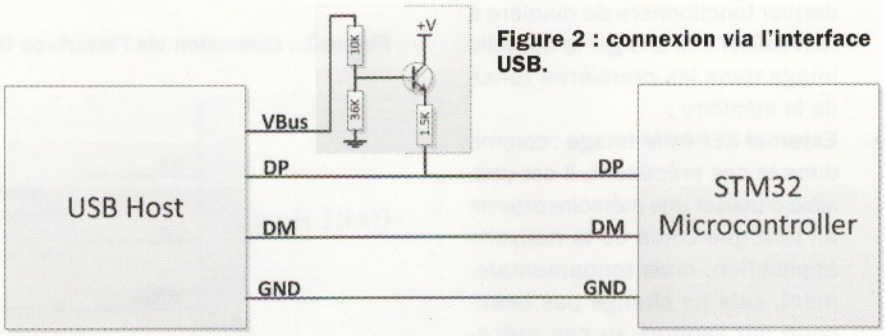


Figure 2 : connexion via l'interface USB.

Pour tous ceux qui ne sont pas familiarisés avec ce type d'interface, il est également important de noter que les broches « Tx » et « Rx » sont indépendantes, précisément parce que les lignes sont dédiées.

Il existe également un mode de connexion appelé « DFU », qui signifie « Device Firmware Update ». Dans ce mode, comme illustré en figure 2, le microcontrôleur est directement interfacé avec l'hôte USB.

Il est clair que l'alimentation du bus nécessite des composants dédiés, qui mettent en œuvre un réseau de type « pull-up ». Dans ce cas, la différence réside dans le fait que le réseau est dédié à l'alimentation et non aux lignes de données.

Lors de l'utilisation du bus I2C, la configuration implique l'utilisation de deux lignes dédiées, l'une servant à transmettre les signaux de synchronisation (SCL) et l'autre à la transmission des données (SDA).

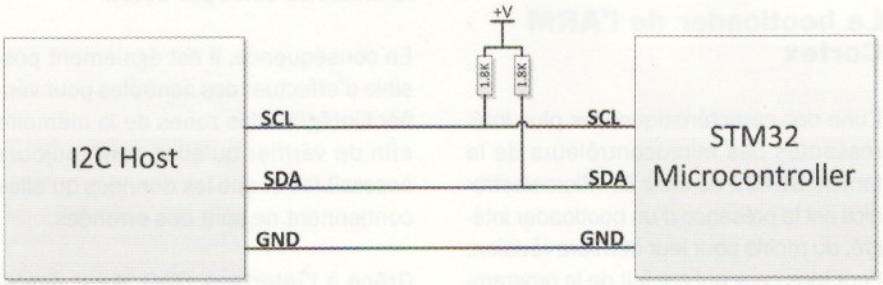


Figure 3 : connexion via l'interface I2C.

Également dans ce cas, comme dans la première configuration, les composants qui implémentent les réseaux de pull-up sont dédiés aux lignes de données (voir la figure 3).

Avec l'interface SPI, les connexions sont composées d'une ligne de synchronisation dédiée pour les signaux d'horloge (SCK), et de deux lignes dédiées pour la transmission des données.

Dans ce cas, en particulier, l'interface exige que le périphérique qui agit en tant qu'hôte soit le maître (MASTER) de la communication, tandis que celui auquel les données sont transmises se comporte comme un esclave (SLAVE). Une simple résistance est généralement utilisée qui référence à la masse le signal d'horloge (voir la figure 4).

Enfin, la figure 5 montre la connexion via l'interface CAN. Ici, l'hôte communique avec le périphérique via l'utilisation de lignes dédiées passant par deux émetteurs/récepteurs et un câble série. En règle générale, il peut être utile d'utiliser une résistance de terminaison de 120 Ω.

Il est à noter que l'exécution des opérations dans cette phase inclut une série de conditions et de contrôles qui sont résumés en figure 6.

C'est précisément dans la séquence des opérations que ce mode de fonctionnement est particulier.

En fait, l'ensemble de la séquence de démarrage prévoit, après le choix de l'interface de communication à utiliser, un « handshake » c'est-à-dire l'établissement d'une liaison pour la synchronisation. Cette dernière est suivie de la vérification de l'adresse, dans le cas du bus I2C, et de l'interface de communication choisie pour la programmation. Il sera donc nécessaire de vérifier l'adresse spécifique.

Dès qu'il est établi que la cible est correcte, il est nécessaire de désactiver chaque source d'interruption afin d'interrompre l'exécution de toute opération, puis d'activer ensuite l'interface de communication. Il s'agit d'une phase préliminaire qui dépend de la famille à laquelle appartient le microcontrôleur. En particulier, dans ce cas, nous parlons de la famille STM32F030xC.

Il est très important de noter que des considérations similaires, comparées à celles observées jusqu'à présent, peuvent être apportées aux logiciels fournis par les différents fabricants, Texas Instruments, Analog Device, NXP, Atmel, etc. Pour l'instant, nous allons nous concentrer sur le logiciel fourni par ST Microelectronics.

En particulier, il existe une solution logicielle qui permet de réaliser le « **flash loader** » via une interface graphique, sous Windows. Bien sûr, vous pouvez également l'utiliser sous la forme de lignes de commandes. Il s'agit d'un script en langage Python que vous pouvez télécharger à partir de l'adresse suivante : https://github.com/jsnyder/stm32loader.

En lisant la documentation, vous vous rendrez facilement compte que pour pouvoir utiliser cette routine, vous devez avoir installé le langage Python sur votre ordinateur. Pour cela, nous vous recommandons d'utiliser le site web officiel à l'adresse : https://www.python.org/downloads/.

Bien sûr, vous pouvez exécuter un script Python dans n'importe quel environnement.

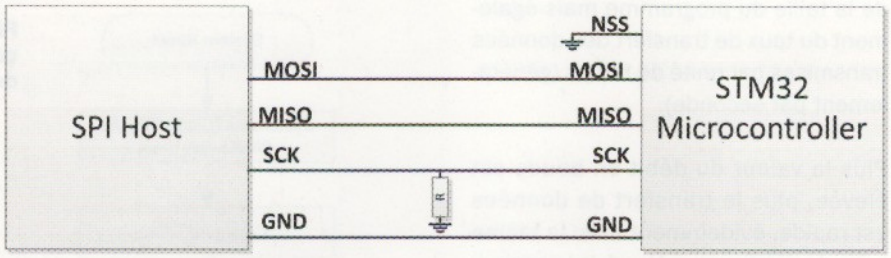


Figure 4 : connexion via l'interface SPI.

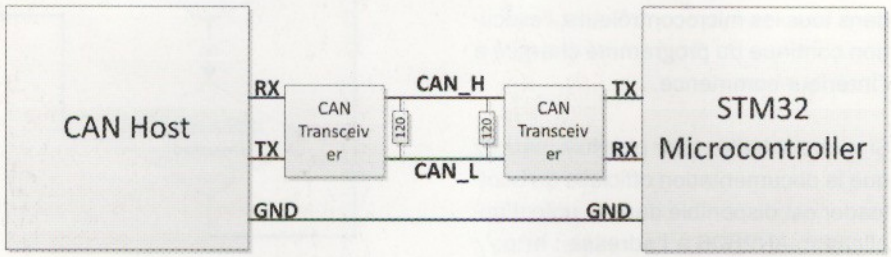


Figure 5 : connexion via le bus CAN.

La syntaxe d'appel est la suivante :

stm32loader.py [-hqVewvr] [-l length] [-p port]
[-b baud] [-a addr] [file.bin]

Dans le tableau 1 sont reportés tous les paramètres. Comme toutes les routines, celle-ci possède également des paramètres par défaut, notamment ceux concernant le port série.

À cet égard, il a été choisi de définir la valeur du débit à 115200 bauds, ce qui accélère considérablement le transfert des données par rapport à une valeur de 9600 bauds, par exemple.

Cependant, il est toujours possible de redéfinir à la fois l'adresse du port et le nombre de données envoyées par seconde (taux de transfert) via l'interface série. Cela dépend du fait que tous les dispositifs ne sont pas nécessairement capables de prendre en charge ces caractéristiques. Pour cette raison, il est possible de redéfinir le port, en utilisant le paramètre « p », et le débit en bauds à l'aide du paramètre « b ».

La sortie de la console suggère également que, quels que soient le nombre et le type de paramètres que vous souhaitez définir, le dernier champ à renseigner sera le nom du fichier binaire que vous souhaitez télécharger dans le microcontrôleur.

Tableau 1 - Paramètres d'appel.

-h	Help
-q	Quiet
-V	Verbose
-e	Erase
-w	Write
-v	Verify
-r	Read
-l	Length
-p	Serial Port
-b	Baud rate (speed, default: 115200)
-a	Target Address

Une fois que vous aurez choisi le mode de téléchargement approprié, le microcontrôleur attendra les données provenant de l'ordinateur.

Le débit en bauds étant un paramètre fondamental dans ce type de communication, sachez que si vous ne le spécifiez pas, il sera défini automatiquement. Ce type de fonctionnalité est judicieux, car les communications sont cohérentes et optimisées, en particulier dans le cas où les programmeurs ont moins d'expérience dans la gestion de ces paramètres.

Pour utiliser cette routine en Python, il est important de sélectionner « BOOT0 » à un niveau logique haut et d'exécuter le script. Le processus prend un certain temps, qui dépend évidemment

de la taille du programme mais également du taux de transfert des données transmises par unité de temps (généralement par seconde).

Plus la valeur du débit en bauds est élevée, plus le transfert de données est rapide, évidemment avec la même taille de programme. Une fois le processus terminé, il est possible de ramener « BOOT0 » à un niveau logique bas et de réinitialiser la carte. Ensuite, comme dans tous les microcontrôleurs, l'exécution continue du programme chargée à l'intérieur commence.

Si vous souhaitez aller plus loin, sachez que la documentation officielle du bootloader est disponible dans la note d'application AN2606 à l'adresse : http://www.st.com/stonline/products/literature/an/13801.pdf.

Un exemple concret : le BSL XMC1000

En ce qui concerne la famille XMC1000 de Infineon, les choses changent légèrement. Nous allons concentrer notre attention sur les familles XMC1100/1200/1300/1400.

Les microcontrôleurs auxquels nous faisons référence ont un mécanisme de Bootstrap (Bootstrap Loader ou BSL) intégré, qui peut être utilisé pour la programmation de la mémoire Flash.

En fait, comme nous l'avons déjà évoqué dans les paragraphes précédents, le but des fabricants est de fournir des solutions logicielles permettant une automatisation du processus.

Cependant, il est très important de comprendre le fonctionnement de ces routines afin de pouvoir modifier, même matériellement, les solutions adoptées.

Le mécanisme de fonctionnement n'inclut pas de routines codées en dur dans la « BootROM », c'est précisément pour cette raison que les routines Flash doivent être implémentées par l'utilisateur.

Afin de travailler sur la famille spécifique dont nous parlons dans ce cas, il est nécessaire d'utiliser une suite d'outils

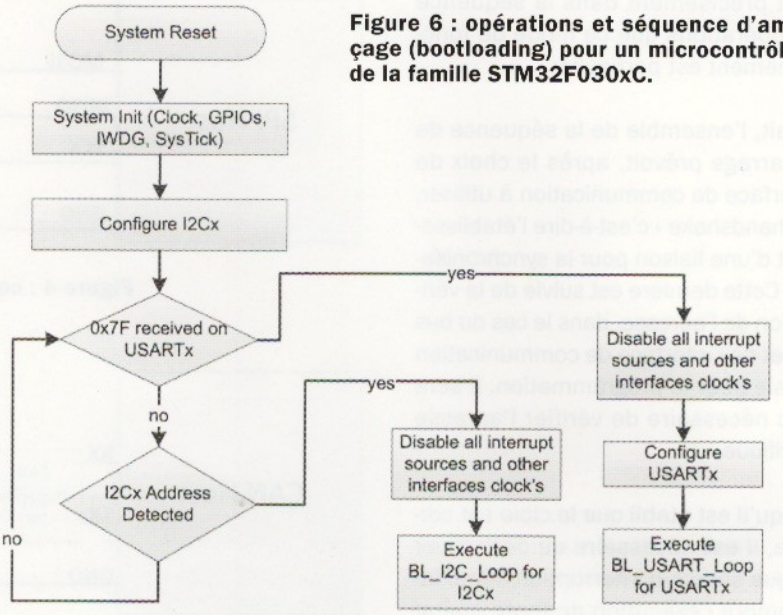


Figure 6 : opérations et séquence d'amorçage (bootloading) pour un microcontrôleur de la famille STM32F030xC.

développée directement par Infineon, l'environnement de développement s'appelle DAVE (à ne pas confondre avec le célèbre chanteur), actuellement dans sa version 4.4.2.

La famille XMC1000 prend en charge le mode BSL (Asynchronous Serial Interface (ASC) pour interface série asynchrone) et l'interface série synchrone SSC (Synchronous Serial Interface).

En général, lorsque le microcontrôleur est connecté à l'ordinateur via une interface ASC, deux sections sont utilisées :

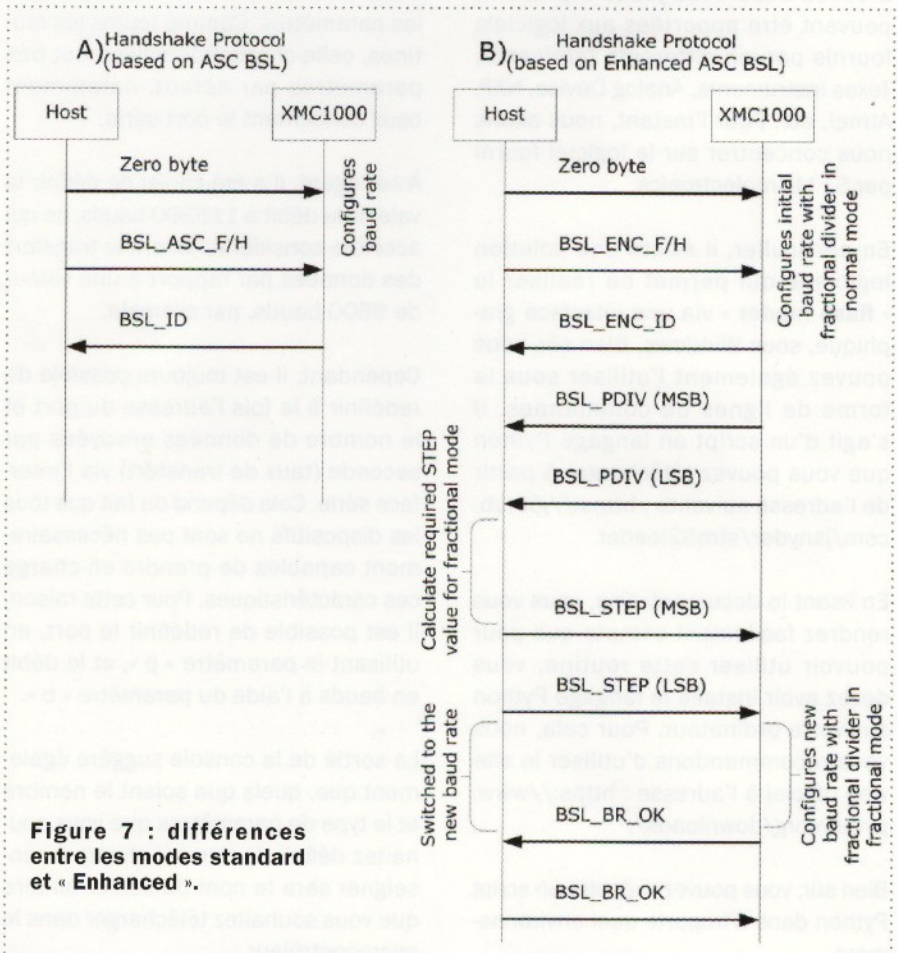


Figure 7 : différences entre les modes standard et « Enhanced ».

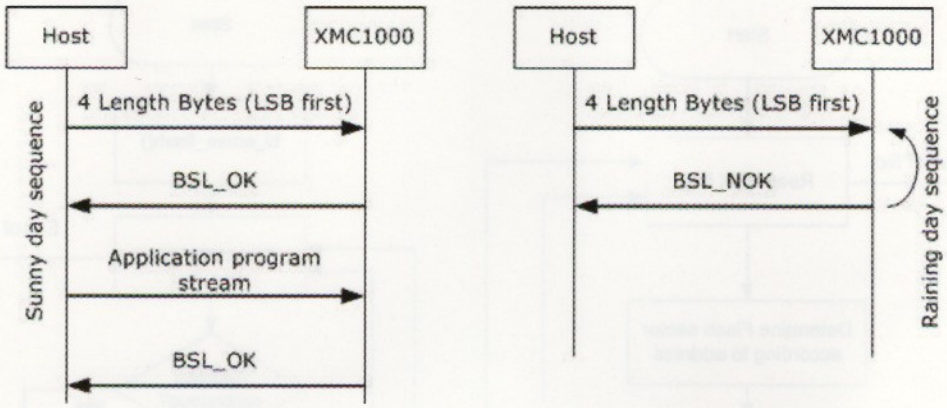


Figure 8 : séquence de téléchargement dans les deux cas.

- le **programme Flash Loader**, qui est envoyé au dispositif à l'aide du mécanisme de Bootstrap intégré. Une fois le programme envoyé et exécuté, le programme Flash Loader établit la communication pour recevoir les commandes du programme hôte (Host Program) tournant sur l'ordinateur ;
- le **programme hôte**, exécuté sur le PC, utilise le protocole de communication défini pour envoyer les commandes et les codes spécifiques à programmer. Il va sans dire que cela signifie qu'il s'agit d'un programme écrit pour une application donnée et qu'il faudra donc réécrire et éventuellement le réinterpréter si l'interface ou le microcontrôleur sont modifiés.

Maintenant, entrons dans les détails du Bootstrap. En mode « ASC_BSL », l'utilisateur peut télécharger un programme de monitoring à l'intérieur de la mémoire SRAM du XMC1000 à partir d'une adresse mémoire spécifique qui est : 0x20000200.

Une fois reçu et mémorisé, le programme sera exécuté de manière à permettre les communications avec le PC hôte afin de pouvoir supprimer, programmer et vérifier les données de la mémoire Flash. Vous devez effectuer au préalable deux opérations, à savoir l'allumage et la réinitialisation du dispositif (ici le microcontrôleur), puis ensuite il restera en attente.

L'« ASC_BSL » peut communiquer avec le PC hôte en mode full-duplex ou half-duplex, ce choix dépend de la manière dont l'octet « Header Byte » est rempli. En particulier, la valeur « 0x12 » implique le mode half-duplex alors que la valeur « 0x6C » correspond au mode full-duplex.

À partir de là, il est assez immédiat de comprendre quelles sont les différences entre les modes asynchrone et synchrone, étant donné les différences entre les exigences d'une communication asynchrone par rapport à une communication synchrone. Le « loader » ASC prend effectivement en charge les deux modes via le protocole UART.

En réalité, l'ASC Bootstrap Loader fonctionne de deux manières : une standard et l'autre étendue (Enhanced). La principale différence entre les deux réside dans le fait que, dans le cas du mode « Enchanced », des débits en bauds nettement plus élevés sont pris en charge.

Quel que soit le mode que vous décidez d'utiliser, dans les deux cas, vous aurez besoin de :

- détecter le débit en bauds et exécuter la séquence de sélection du mode ;
- établir la séquence de téléchargement.

En ce qui concerne la première action, l'interaction entre le bootloader et l'hôte s'effectue via un protocole. Il existe donc un « handshake » suivi d'une requête, d'un accusé de réception (acknowledgment) et d'un transfert de données.

En figure 7, vous pouvez clairement voir qu'il existe une grande différence entre le **mode standard** beaucoup plus simple et le **mode « Enchanced »**, dans lequel une nouvelle valeur possible pour le débit en bauds est également négociée.

Une fois le débit en bauds et le canal/mode déterminés, l'ASC Bootstrap attend 4 octets indiquant la longueur de l'application ou sa taille. Ensuite, la deuxième phase a lieu, et correspond à la séquence de téléchargement, comme illustré en figure 8.

Le Flash Loader conçu pour la famille XMC1000 implémente des routines servant à établir la communication entre le PC et le périphérique cible, ainsi que les fonctionnalités suivantes :

- suppression des zones de mémoire ;
- suppression, programmation et vérification des pages de mémoire programmées ;
- changement de mode du « Boot Mode Index » (BMI) du dispositif cible.

En particulier, dans le processus d'effacement de la mémoire Flash (Flash Erase), chaque secteur de la mémoire est contrôlé afin de vérifier s'il est plein ou non (voir la figure 9).

Si ce n'est pas le cas, le mode approprié est défini, puis on passe au secteur suivant. La procédure est itérée jusqu'au parcours complet du fichier.

Après l'exécution du processus d'effacement de la mémoire Flash, le fichier est lu afin de vérifier d'abord les zones de mémoire, puis leur contiguïté mais

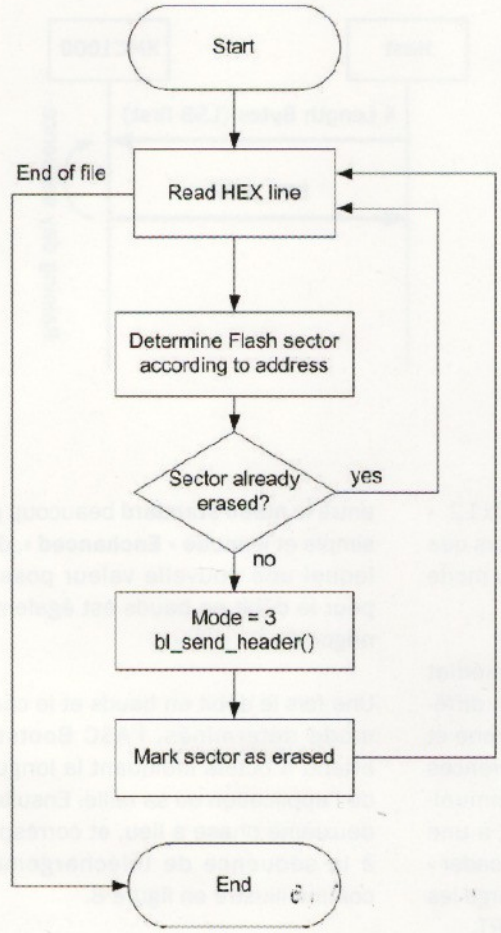


Figure 9 : procédure d'effacement de la mémoire Flash.

surtout s'il s'agit de la première tentative d'écriture ou non. Bien sûr, ce contrôle est utilisé pour vérifier si des secteurs défectueux sont présents.

Si toutes les vérifications ont donné un résultat négatif, les zones de mémoire seront écrites de manière appropriée jusqu'à la taille souhaitée (voir la figure 10). L'opération est itérée jusqu'à la fin de l'écriture complète du fichier.

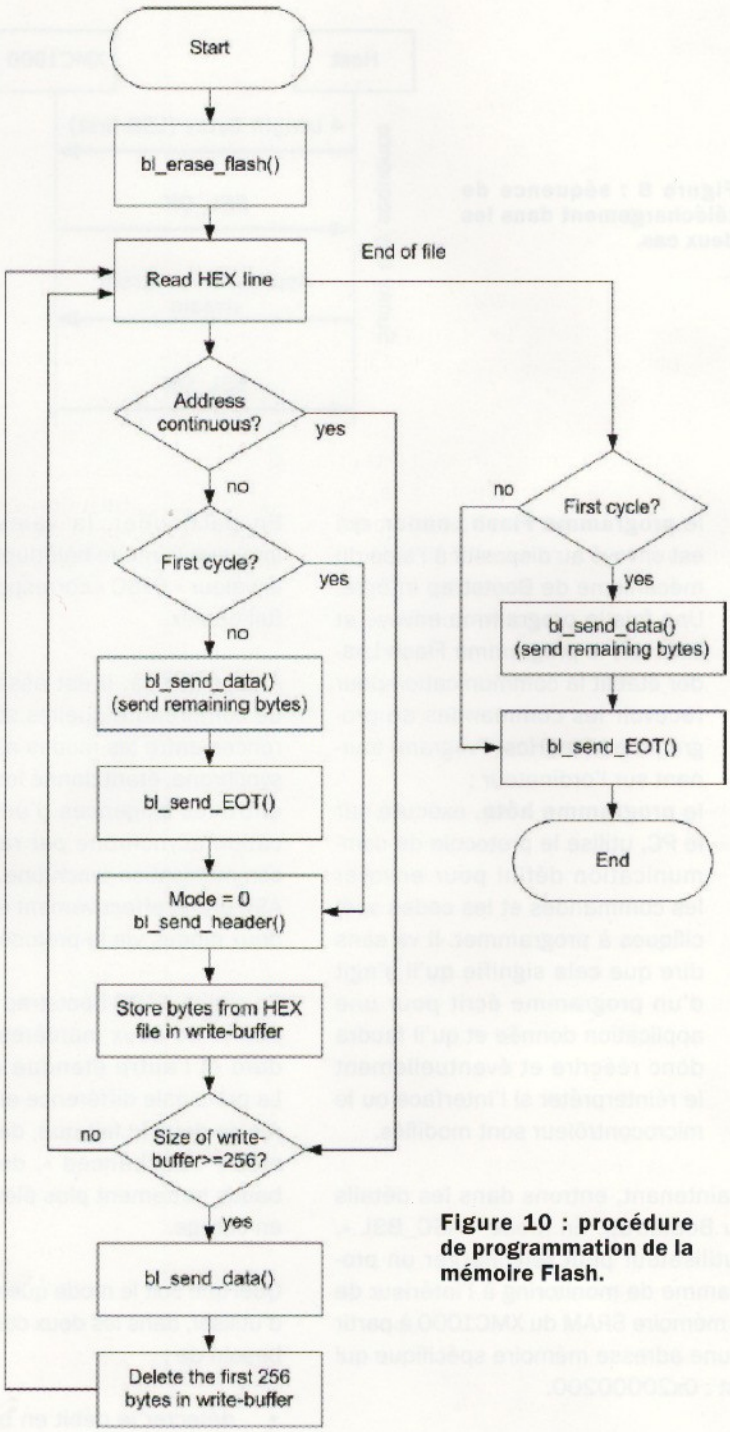


Figure 10 : procédure de programmation de la mémoire Flash.

Ces opérations, toujours indispensables et communes à tous les microcontrôleurs, s'effectuent de manière transparente pour l'utilisateur et c'est précisément là que réside, du moins pour cette famille de microcontrôleurs, la principale différence.

Tout ce que nous avons décrit peut également être représenté graphiquement, pour cela nous allons examiner maintenant l'interface logicielle à travers laquelle il est possible de définir la valeur du « BMI » (voir la figure 11).

Le cas de l'utilisateur

L'un des aspects fondamentaux de la philosophie de l'utilisateur est la simplification des concepts complexes. Cet exemple peut être à la fois pratique et théorique.

Alors que les microcontrôleurs se diversifient et que les fabricants tentent de proposer des solutions variées, actuellement des efforts sont faits par l'ensemble de la communauté pour créer des solutions polyvalentes et adaptables à des besoins multiples.

Plus précisément, un utilisateur peut écrire du code et le télécharger dans un microcontrôleur sans trop s'inquiéter de la technique de communication, de l'interface utilisée, et en ignorant, parfois à son insu, tout ce qui se cache derrière la technique de bootloader. Ceux qui utilisent Arduino ignorent très souvent la configuration de la séquence de démarrage (boot), ou ce qu'est un « handshake ».

Par conséquent, l'utilisateur lambda n'est pas au courant des nombreuses autres variables, et fonctions dont nous avons parlé jusqu'à présent. Cependant, cela reste indispensable.

Comment Arduino garantit-il une approche didactique du problème ?

La réponse est qu'il est relativement facile de configurer le microcontrôleur ATMega328 présent dans la carte Arduino UNO.

Examinons comment un utilisateur aborde ce type de problématique, en résolvant le problème du chargement du firmware dans le microcontrôleur.

La figure 12 montre le schéma de câblage du microcontrôleur, que nous pouvons directement comparer aux instructions de chargement du bootloader visibles en figure 13.

Mais à quoi servent ces schémas ? Et comment sont-ils utilisés ?

Comme nous l'avons déjà mentionné, les microcontrôleurs sont généralement programmés via un programmateur dédié, sauf s'il existe un firmware dans le microcontrôleur permettant l'installation de nouvelles fonctionnalités ou la réécriture du programme présent dans un composant externe. Ce dernier est le bootloader.

Si vous ne souhaitez pas utiliser une partie de la mémoire Flash et que vous souhaitez également éviter les problèmes de retard dus au bootloader, vous pouvez utiliser un programmateur externe.

Pour pouvoir flasher le bootloader, vous avez besoin d'un programmateur AVR-ISP (ce dernier est l'acronyme de In System Programmer), ou d'un USBtinyISP, ou encore un « programmateur parallèle ». Quelle que soit la solution choisie, sachez que vous devrez utiliser le connecteur ICSP à 6 broches (sous la forme de deux rangées de trois broches).

L'utilisation de ce connecteur nécessite tout d'abord de faire très attention aux connexions, car chaque broche a sa propre fonction spécifique.

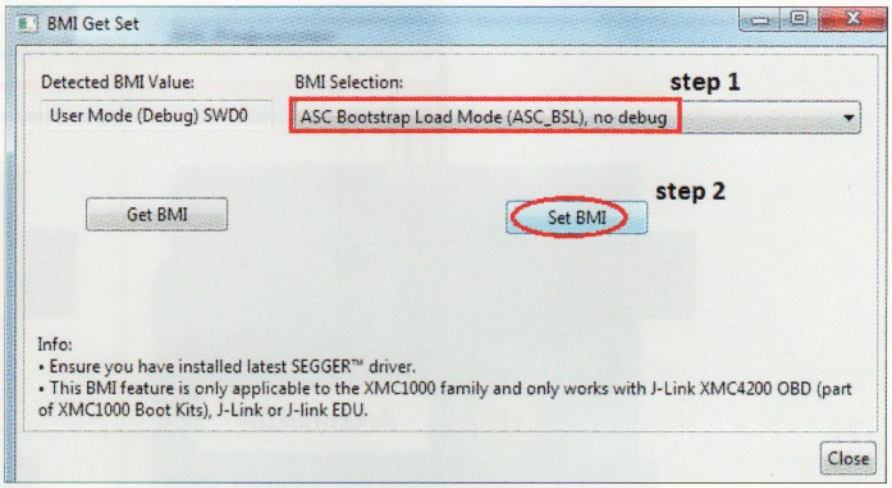


Figure 11 : utilisation de l'outil spécifique pour définir la valeur du BMI sur les microcontrôleurs XMC1x00.

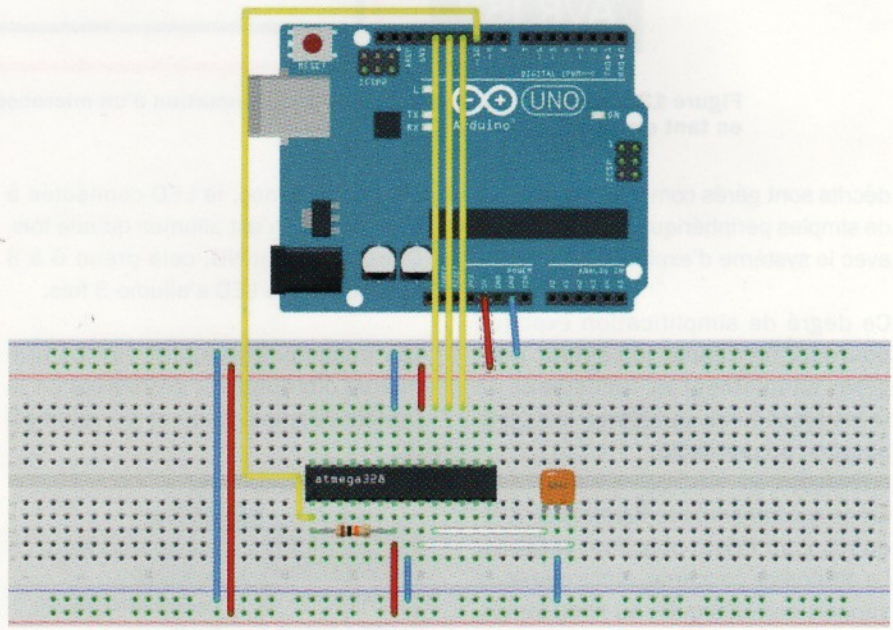


Figure 12 : schéma de câblage des connexions pour la programmation d'un microcontrôleur.

Dans tous les cas, gardez à l'esprit que vous aurez besoin d'une alimentation externe pour la carte.

Une fois cela effectué, vous devez simplement utiliser l'environnement de développement IDE Arduino, en sélectionnant dans le menu « Outils » l'onglet « Graver la séquence d'initialisation ».

Pourquoi Arduino rend-il cette simplification possible ?

Parce que l'IDE d'Arduino n'intègre pas seulement un interpréteur de commandes ou certaines des fonctionnalités les plus utiles pour les programmeurs en termes de lisibilité du code.

En fait, il possède et intègre une série de fonctionnalités inhérentes à l'interprétation du code et au lancement des séquences de chargement des fichiers compilés vers la carte.

De plus, l'interaction entre l'utilisateur et la carte Arduino est gérée comme si elle était de plus haut niveau, car l'échange des messages s'effectue à l'aide d'un circuit intégré qui gère les communications via l'interface RS232.

En outre, il convient de noter que lors de l'installation de l'IDE Arduino, des pilotes (drivers) spécifiques sont installés et que, par conséquent, des processus complexes tels que ceux déjà

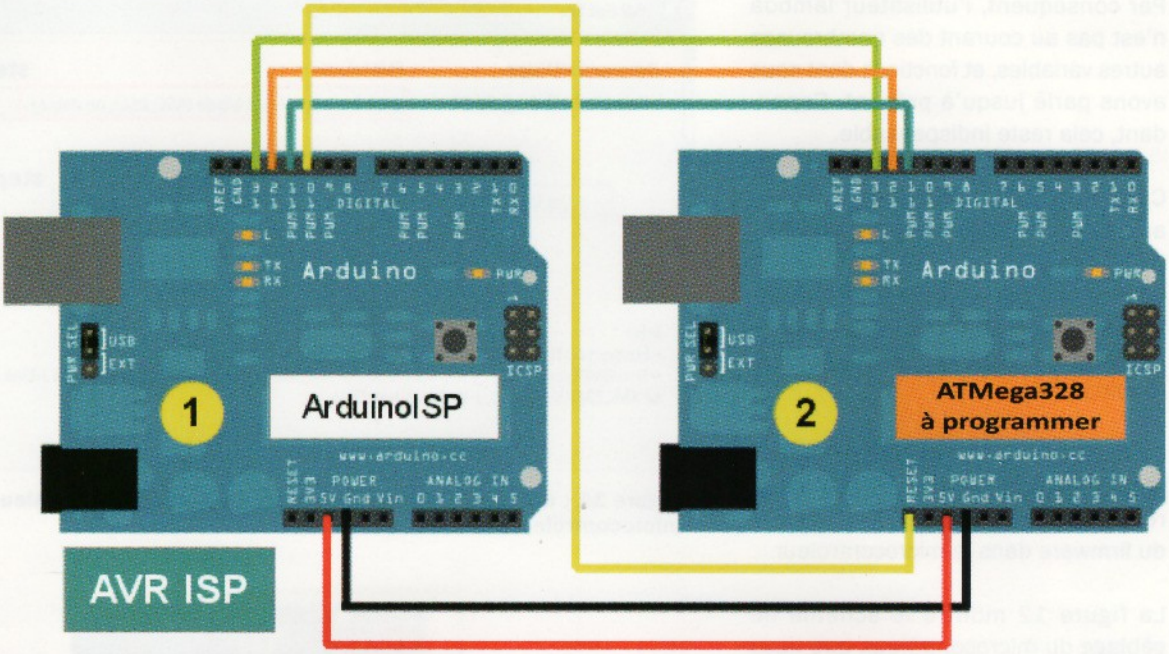


Figure 13 : schéma de câblage pour la programmation d'un microcontrôleur utilisant une carte Arduino UNO en tant que programmateur.

décrits sont gérés comme s'il s'agissait de simples périphériques en interaction avec le système d'exploitation.

Ce degré de simplification explique pourquoi un simple utilisateur peut gérer simplement et facilement de telles procédures complexes sans préparation au préalable.

En ce qui concerne le bootloader d'Arduino, il en existe plusieurs versions. La différence concerne, entre autres, le support matériel qui+, au fil du temps, continue évidemment d'évoluer. Dans le cas d'Arduino, au moment de la rédaction de cet article, les différentes versions sont : **NG** et **Diecimila**.

Les deux fonctionnent avec un débit de 19200 bauds et occupent environ 2 ko de mémoire Flash dans un ATMega168 (NG).

La différence fondamentale réside dans le temps d'attente de la nouvelle version du firmware et dans le nombre de fois que la LED connectée à la broche 13 s'allume.

Étant donné que dans la version Diecimila (et ultérieure), la réinitialisation automatique est implémentée, son bootloader doit attendre très peu de temps, moins d'une seconde, pour être sauvegardé.

En conséquence, la LED connectée à la broche 13 n'est allumée qu'une fois. Dans le cas du NG, cela prend 6 à 8 secondes et la LED s'allume 3 fois.

Les anciennes versions, sur les premières distributions des cartes de la famille Arduino, avaient un débit en bauds nettement inférieur, fixé à la valeur classique de 9600 bauds.

Bien entendu, il est toujours possible d'utiliser un bootloader externe, produit par des tiers ou d'autres utilisateurs de la communauté. Naturellement, nous devons tenir compte de la différence

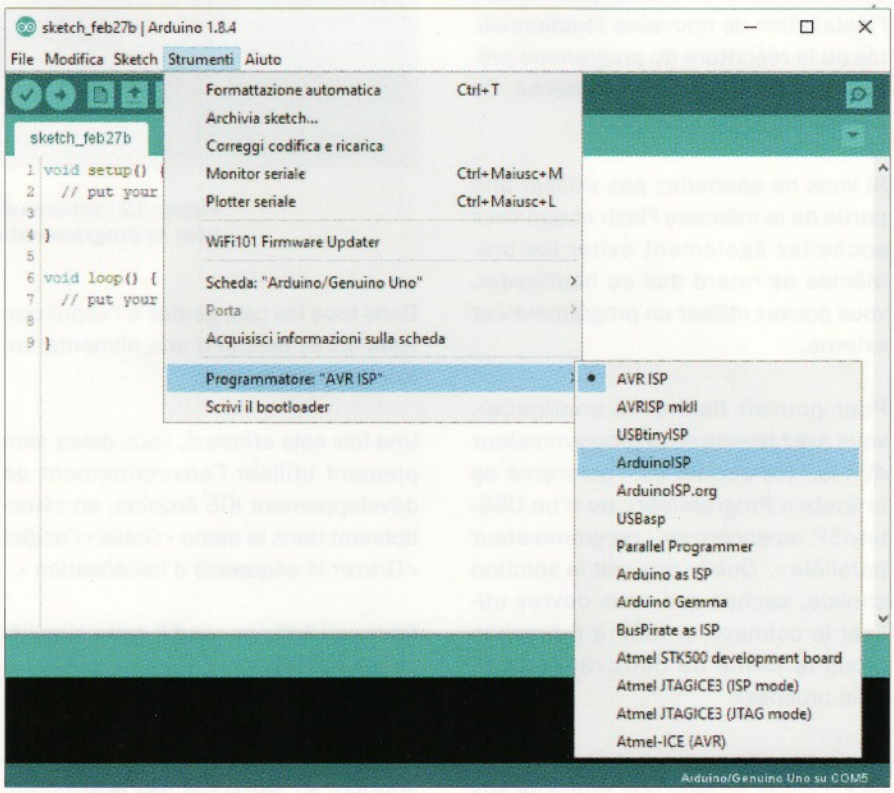


Figure 14 : sélection de la routine « Arduino ISP » dans l'IDE d'Arduino utilisé comme environnement de programmation pour microcontrôleur.

et l'étudier correctement avant de l'utiliser, de manière à vérifier la configuration des paramètres.

Dans l'environnement Arduino, la commande « Graver la séquence d'initialisation » est associée à l'exécution d'un outil Open Source appelé « **AVRDUDE** ».

Toute personne familiarisée avec cet environnement de développement ou avec cette famille de cartes trouvera sûrement des messages d'erreur mentionnant le nom de cette routine. « AVRDUDE » fonctionne en effectuant quatre étapes de base :

- débloque le bootloader sur la puce ;
- définit le cycle pour l'écriture ;
- charge le code du bootloader ;
- bloque le bootloader.

Naturellement, les quatre opérations sont gérées selon les paramètres présents dans les fichiers de configuration d'Arduino. Pour cette raison, ils peuvent être personnalisés en modifiant les valeurs par défaut des routines de déblocage.

Concrètement, pour effectuer cette opération, il est nécessaire d'utiliser la routine « Arduino ISP » (voir la figure 14).

Ce mode permet d'utiliser une carte Arduino générique en tant que programmateur pour d'autres microcontrôleurs, là encore de type AVR.

La routine peut être utilisée avec succès. En particulier, comme nous l'avons dit précédemment, dans les versions les plus récentes du bootloader, une fonction de réinitialisation automatique a été implémentée, qui sert précisément à garantir l'interruption de l'exécution de chaque opération en cours par le microcontrôleur, l'arrêt et la validation des opérations d'écriture.

Pour gérer correctement tout conflit avec cette fonction automatique, il est possible d'utiliser un circuit très simple composé d'une résistance d'une valeur de 120 Ω et d'un condensateur de 10 µF connecté directement sur la broche de réinitialisation (pour ainsi dire, la première de la section d'alimentation sur la version UNO R3).

En principe, une résistance d'une valeur suffisamment basse, inférieure à 200 Ω, permet de résoudre le problème.

La solution la plus élégante et la plus complète pour résoudre le problème de l'utilisation d'une carte Arduino en tant que programmateur pour d'autres microcontrôleurs est l'utilisation de la configuration illustrée en figure 13.

Une des deux cartes, qui fonctionnera en tant que programmateur, est configurée avec le firmware Arduino ISP programmé.

L'autre carte sera la cible dans laquelle il faudra télécharger le bootloader ou mettre à jour celui-ci s'il est déjà présent.

Il convient de souligner que la version UNO R3 permet le remplacement du microcontrôleur car il ne s'agit pas d'un composant de type CMS (montage en surface), mais bien d'un boîtier DIP avec un support. Cela facilite l'extraction et donc l'utilisation en tant que programmateur.

De toute évidence, la carte qui servira de « source » doit être reliée au PC et alimentée par le port USB. La même alimentation est également fournie à la carte de destination.

Après avoir abordé les connexions physiques, passons à l'utilisation de l'environnement IDE d'Arduino.

Dans ce dernier, il est essentiel de vérifier la version de la carte connectée et l'indication du port série via le menu « Outils ».

Une fois la routine « Arduino ISP » sélectionnée, à la fin de la procédure, notre carte deviendra la « source » et contiendra le firmware. Il ne sera plus nécessaire de la reprogrammer, à condition que vous n'ayez pas à l'utiliser pour d'autres projets.

Désormais, il est possible de programmer correctement le microcontrôleur Atmel AVR présent sur la carte cible.
Il suffit de connecter cette dernière et d'observer le transfert de données via l'interface série grâce aux LED « Tx » et « Rx » utilisées pour donner un retour visuel des transmissions en cours.

À partir de cette brève description, il est possible d'aborder, pour le particulier amateur, un problème extrêmement complexe en utilisant des instruments très simples et en exploitant tout le potentiel des microcontrôleurs.

Conclusion

Nous arrivons à la fin de ce cours didactique dédié au monde de la programmation embarquée (embedded).

Nous avons étudié le bootloader, les opérations et les séquences de démarrage des microcontrôleurs sous différents angles, en analysant la programmation, la mise en œuvre et la configuration de diverses solutions.

L'avantage est évident, car différents membres d'une même famille, différentes familles de microcontrôleurs, mais aussi différentes générations de microcontrôleurs utilisent différentes méthodes et techniques de programmation.

Pour l'amateur qui entre dans le monde de la programmation embarquée, il est très important de connaître les principes de base et de pouvoir ensuite les adapter.

En fait, ce sont des connaissances techniques, en général, qui dérivent des compétences variées sur les microcontrôleurs.

Elles peuvent être utilisées même si un microcontrôleur n'est pas connu.

De plus, nous avons également vu comment et dans quelle mesure l'approche et les solutions adoptées par l'amateur peuvent avoir un impact significatif en termes de simplification et d'immédiateté.

En fin de compte, nous pouvons dire que la compréhension du bootloader, ainsi que sa mise en œuvre, est une condition indispensable à l'utilisation correcte d'un microcontrôleur, quel qu'il soit. ■

Nous vous proposons un montage qui détecte l'humidité à l'aide de deux capteurs à utiliser alternativement l'un par rapport à l'autre. Un capteur pour indiquer la présence d'eau sur le sol après un orage, l'autre pour mesurer le taux d'humidité d'une plante, par exemple, qui est trop bas et nécessite donc un arrosage.

CAPTEUR DE PLUIE ET D'HUMIDITÉ

de Davide Scullino

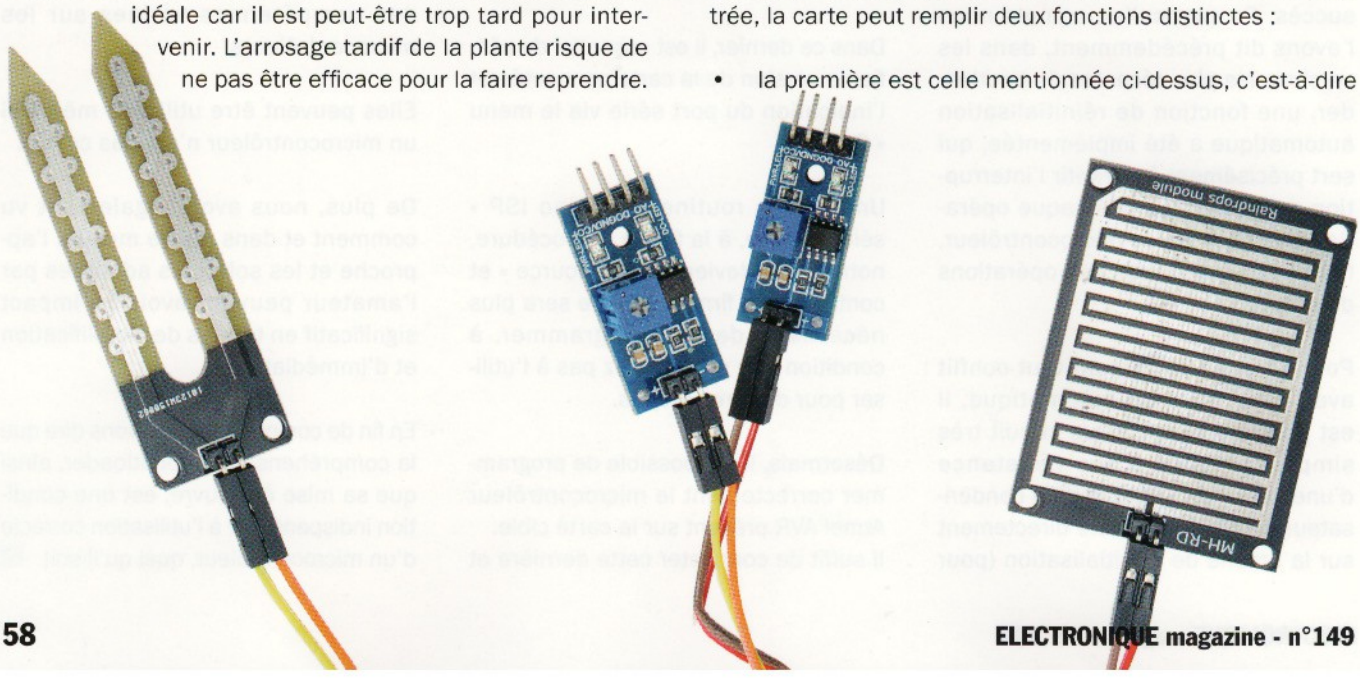
Comment connaître la bonne teneur en humidité du sol ou d'un pot de fleur ?

L'expérience est certainement utile, tout comme de constater que la plante commence à perdre son tonus et à se dessécher, mais dans ce cas, la méthode ne serait pas idéale car il est peut-être trop tard pour intervenir. L'arrosage tardif de la plante risque de ne pas être efficace pour la faire reprendre.

Les produits de jardinage du commerce indiquant l'humidité de la terre peuvent aider. Nous souhaitons réaliser dans ces pages quelque chose de plus sophistiqué sous la forme d'une carte électronique, avec un circuit très simple.

En fonction du capteur utilisé que nous connectons sur l'entrée, la carte peut remplir deux fonctions distinctes :

- la première est celle mentionnée ci-dessus, c'est-à-dire



qu'avec **deux pointes métalliques enfoncées dans le sol**, nous allons vérifier le **taux d'humidité** de la terre. Est-elle suffisamment humide ?

- la seconde est toujours basée sur la détection d'eau, mais dans ce cas, il s'agit d'**un capteur de pluie** (en immersion, en général ...) qui utilise un circuit imprimé avec des **électrodes qui se présentent sous la forme d'un peigne**. Il s'agit d'un **capteur interdigité** permettant la détection d'eau.

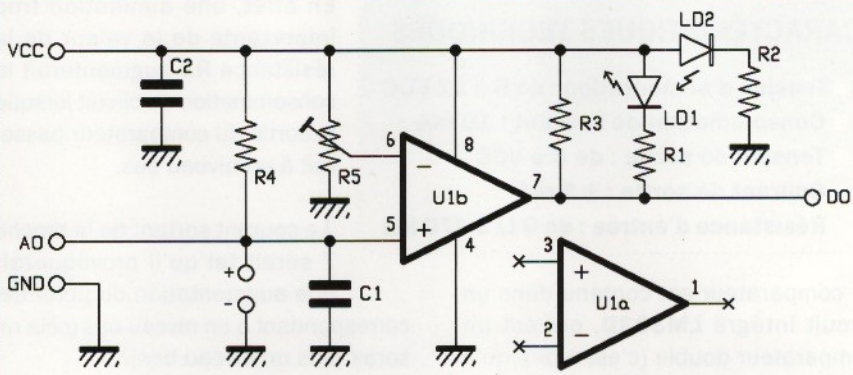


Schéma électrique du capteur de pluie et d'humidité.

Au final, le circuit est toujours identique et repose sur la détection d'une quantité d'eau suffisante.

Comme nous le verrons dans le schéma électrique, il s'agit d'un **circuit à déclenchement de seuil exploitant la conductivité électrique intrinsèque de l'eau minéralisée** (c'est de l'eau courante, certainement pas de l'eau distillée, qui est théoriquement isolante).

La différence est que dans le premier cas, nous avons besoin de détecter l'instant où il y a très peu d'eau pour joindre les contacts du capteur et dans le second lorsqu'il y en a beaucoup.

Le capteur de pluie peut être combiné avec des unités de contrôle d'irrigation de jardin, car il est inutile d'effectuer des séquences d'arrosage lorsque la pelouse est déjà mouillée par la pluie.

De plus, après que la pluie soit terminée, même dans ce cas, il est inutile d'irriguer le jardin à cause de la stagnation de l'eau, étant donné qu'il y en a suffisamment (la terre est détrempée).

Schéma électrique

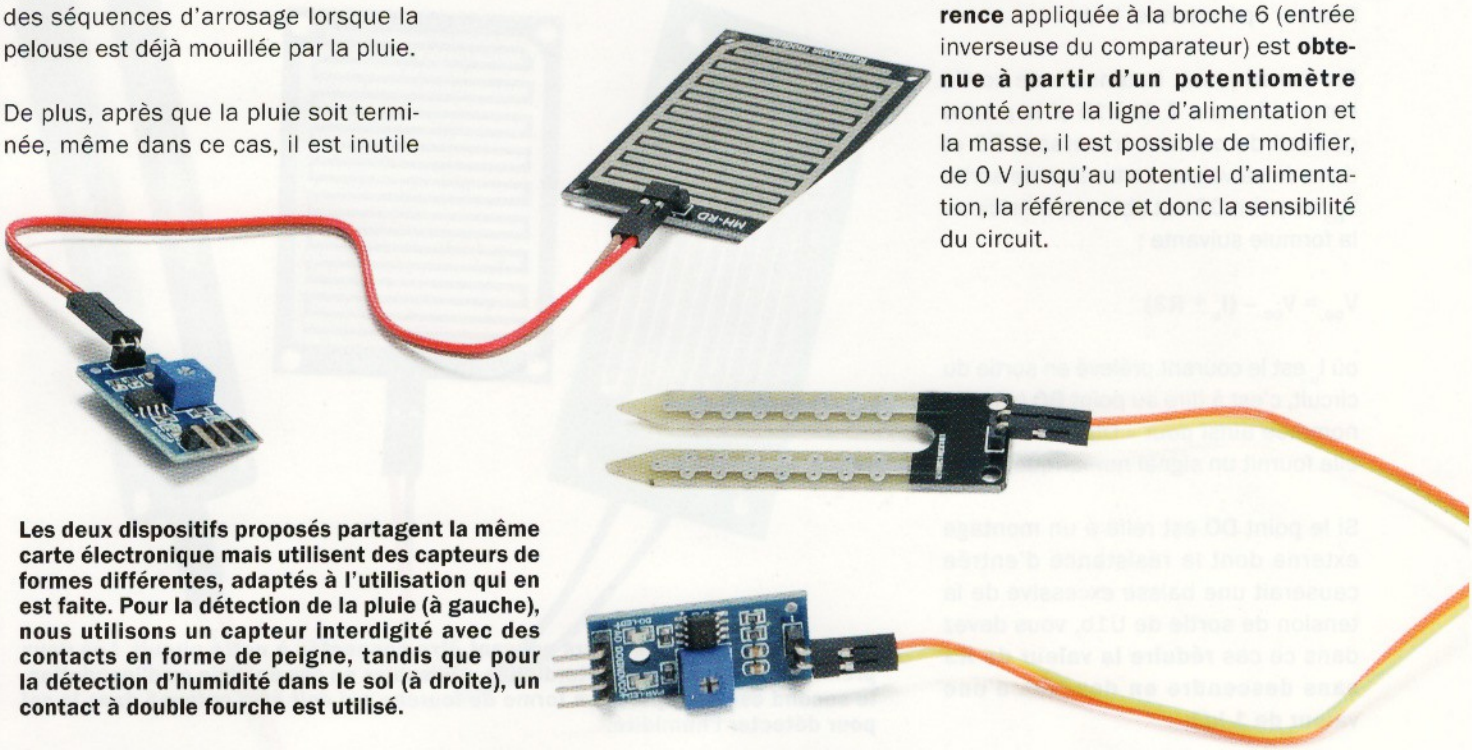
Le circuit est essentiellement composé d'un **comparateur de tension**, l'un des deux amplificateurs opérationnels contenus dans le circuit intégré U1. L'autre, U1a, est laissé déconnecté. Cela permet de réaliser deux fonctions différentes lorsqu'un même événement est détecté.

L'événement correspond à la **diminution de la résistance** entre les points « + » et « - » (c'est-à-dire entre les points A0 et GND).

Ces derniers correspondent dans la pratique au contact du capteur (soit celles du capteur interdigité ou celui du capteur à fourche) avec l'eau, lorsque celle-ci est en quantité suffisante pour établir un contact (électrique) entre les deux électrodes d'un des capteurs.

Le comparateur fonctionne en **mode non inverseur**, en ce sens qu'il reçoit la tension de référence sur l'entrée inverseuse (broche 6) et celle à comparer, sur l'entrée non inverseuse. De plus, **il n'y a pas d'hystérésis**, donc la commutation de la tension de sortie d'un niveau bas vers un niveau haut se produit dès que le potentiel sur l'entrée non inverseuse passe d'un niveau bas vers un niveau haut.

Etant donné que la **tension de référence** appliquée à la broche 6 (entrée inverseuse du comparateur) est **obtenue à partir d'un potentiomètre** monté entre la ligne d'alimentation et la masse, il est possible de modifier, de 0 V jusqu'au potentiel d'alimentation, la référence et donc la sensibilité du circuit.



Les deux dispositifs proposés partagent la même carte électronique mais utilisent des capteurs de formes différentes, adaptés à l'utilisation qui en est faite. Pour la détection de la pluie (à gauche), nous utilisons un capteur interdigité avec des contacts en forme de peigne, tandis que pour la détection d'humidité dans le sol (à droite), un contact à double fourche est utilisé.

CARACTÉRISTIQUES TECHNIQUES

- **Tension d'alimentation : de 5 à 12 VDC**
- **Consommation de courant : 10 mA**
- **Tension de sortie : de 0 à VCC**
- **Courant de sortie : ± 1 mA**
- **Résistance d'entrée : de 0 Ω à 470 kΩ**

Le comparateur est contenu dans un circuit intégré **LM393D**, qui est un comparateur double (c'est-à-dire qu'il contient deux amplificateurs opérationnels spécifiquement conçus pour fonctionner en tant que comparateur).

Il possède donc une sortie capable d'effectuer l'excursion maximale, bien que, lorsque le niveau est bas, la sortie ne descend pas exactement à 0 V (c'est l'une des particularités des amplificateurs opérationnels rail-to-rail) mais reste à environ une centaine de millivolts.

La **sortie** de l'amplificateur opérationnel U1b est de type à « **collecteur ouvert** ». Par conséquent, un niveau haut en sortie nécessite la présence d'une résistance de pull-up qui est dans notre cas R3. Lorsque la sortie passe à un niveau bas (c'est-à-dire qu'elle se trouve au même potentiel que la masse), la résistance R3 permet d'éviter tout court-circuit entre l'alimentation et la masse et permet un fonctionnement correct de l'amplificateur opérationnel U1b.

Par conséquent, la tension de sortie entre la broche 7 de U1b et la masse dépend du courant traversant R3 et donc de sa valeur. La tension de sortie V_{DO} au point DO est obtenue à l'aide de la formule suivante :

$$V_{DO} = V_{CC} - (I_u * R3)$$

où I_u est le courant prélevé en sortie du circuit, c'est à dire au point **DO** (elle est nommée ainsi pour « Digital Output », elle fournit un signal numérique).

Si le point DO est relié à un montage externe dont la résistance d'entrée causerait une baisse excessive de la tension de sortie de U1b, vous devez dans ce cas **réduire la valeur de R3 sans descendre en dessous d'une valeur de 1 kΩ**.

En effet, une diminution trop importante de la valeur de la résistance R3 augmenterait la consommation du circuit lorsque la sortie du comparateur passerait à un niveau bas.

Le courant sortant de la broche 7 serait tel qu'il provoquerait une augmentation du potentiel correspondant à un niveau bas (cela ne serait plus un niveau bas).

Examinons maintenant le fonctionnement du montage. Si entre les points « AO » et GND, c'est-à-dire sur l'entrée entre les contacts « + » et « - », nous connectons un **capteur à fourche** composé d'un circuit imprimé comportant deux électrodes en forme de pointes que nous insérons dans le sol, l'humidité de ce dernier va déterminer une certaine résistance entre les pointes, ce qui formera un **diviseur de tension avec la résistance R4** qui est connectée entre l'alimentation VCC et la **broche 5** du **U1b**.

Pour résumer, lorsque le capteur est introduit dans le sol, une résistance est insérée aux points « + » et « - » qui se trouve alors reliée en série et forme un diviseur de tension avec la résistance R4.

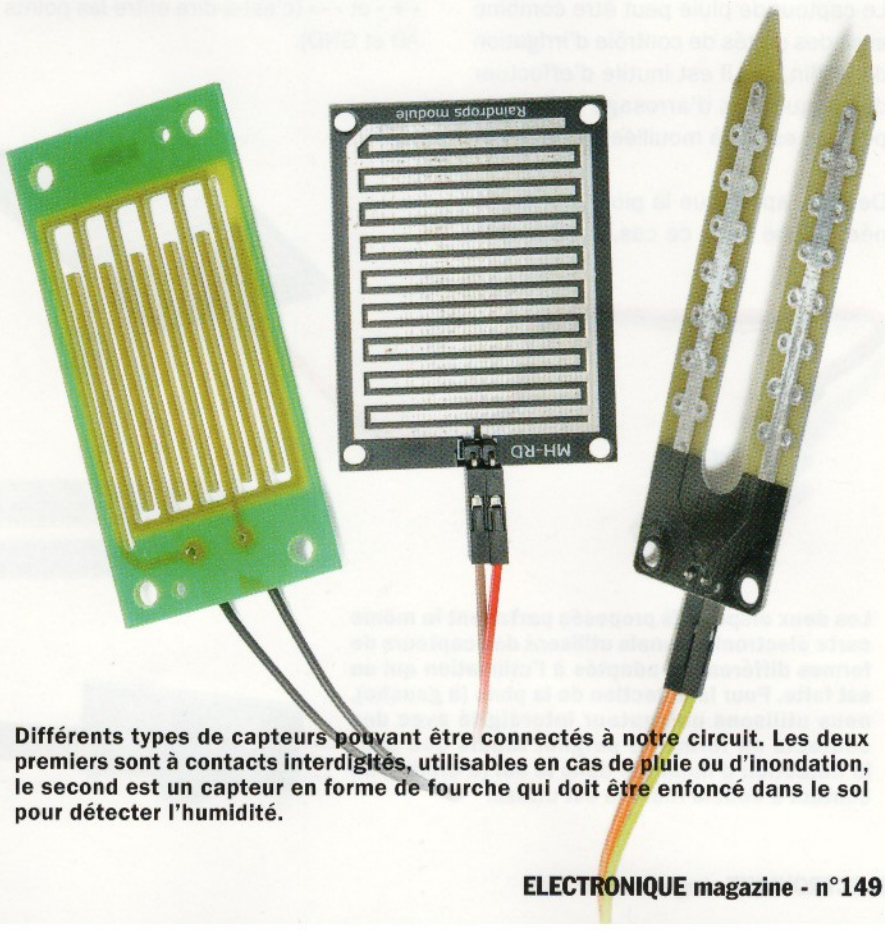
Le **rapport du diviseur de tension** ainsi formé **détermine le potentiel appliqué sur l'entrée** du comparateur.

Par conséquent, à une même tension de référence déterminée par la position du trimmer, plus le sol est humide, plus la tension à laquelle la sortie du comparateur commute est faible.

Cette dernière est à un niveau bas (ce qui allume la LED LD1) lorsque le sol est très humide, la résistance résultante est telle que la tension présente sur la broche 5 de U1b est inférieure à celle appliquée par le trimmer sur la broche 6. La sortie au point DO se trouve alors dans le même état.

Inversement, lorsque le sol commence à se dessécher, le pourcentage d'eau est considérablement réduit et la tension entre les points AO et GND augmente, jusqu'à dépasser le niveau de référence du comparateur, ce qui provoque le passage de la sortie d'un niveau bas vers niveau haut. Ainsi, la LED s'éteint, ce qui indique que nous devons rajouter de l'eau.

Avec le trimmer, nous pouvons **adapter le seuil de déclenchement** à un type de terrain (sol) ou à un type de plante que nous voulons surveiller.



Différents types de capteurs pouvant être connectés à notre circuit. Les deux premiers sont à contacts interdigités, utilisables en cas de pluie ou d'inondation, le second est un capteur en forme de fourche qui doit être enfoncé dans le sol pour détecter l'humidité.

Plan de montage

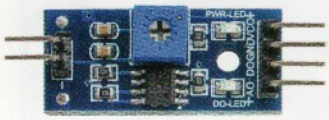
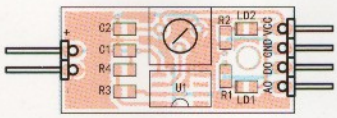


Photo de l'un de nos prototypes du capteur de pluie.



Plan de câblage des composants du capteur de pluie.

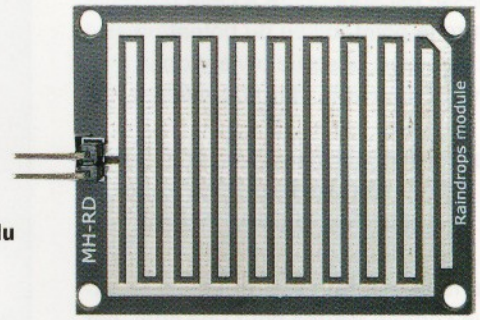


Photo du capteur interdigité avec des contacts en forme de peigne.

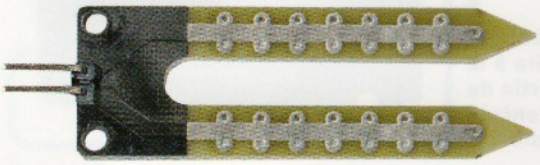
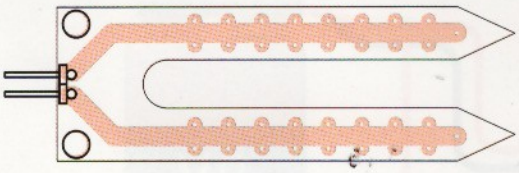
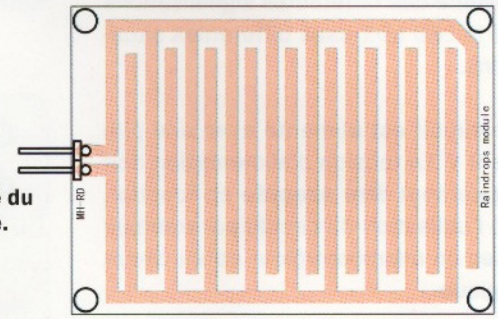


Photo du capteur à double fourche.



Forme du circuit imprimé du capteur à double fourche.



Circuit imprimé du capteur interdigité en forme de peigne.

Liste des composants

R1..... 1 kΩ boîtier CMS 0805
R2..... 1 kΩ boîtier CMS 0805
R3..... 10 kΩ boîtier CMS 0805
R4..... 10 kΩ boîtier CMS 0805
R5..... trimmer 10 kΩ

C1..... 100 nF céramique boîtier CMS 0805
C2..... 100 nF céramique boîtier CMS 0805

U1..... LM393D
LD1.... LED verte boîtier CMS 0805
LD2.... LED verte boîtier CMS 0805

Divers

Barrette mâle 2 pôles coudée à 90°

Barrette mâle 4 pôles coudée à 90°

Rappelons-nous que plus le curseur (point milieu) du trimmer est proche de la masse, plus le sol doit être humide pour allumer la LED et vice-versa (la LED s'éteint lorsque le sol devient de plus en plus sec).

Examinons maintenant comment le circuit se comporte avec un capteur de pluie. Dans ce cas, nous connectons le capteur approprié constitué d'un circuit imprimé à pistes interdigitées aux points AO et GND (ou + et -), en général tout capteur de ce type, même s'il n'est pas constitué par un circuit imprimé. Il existe des capteurs de ce type fabriqués sur de l'époxy.

Le fonctionnement dans ce cas doit être inversé, car nous devons voir la LED s'allumer en présence d'eau. La densité d'eau sur les électrodes détermine la résistance et donc la tension appliquée sur l'entrée du comparateur U1b.

Par conséquent, lorsque l'eau atteint les électrodes, le circuit électrique se ferme et détermine une résistance suffisamment faible pour amener le potentiel sur A0 à une valeur inférieure à celle présente sur la broche 6.

La sortie du comparateur DO passe à un niveau haut lorsqu'il s'arrête de pleuvoir et donc qu'il n'y a plus d'eau sur les contacts du capteur.

Tableau 1 - les différentes signalisations de LD1

Utilisation	LD1 allumée	LD1 éteinte
Capteur d'humidité pour un vase	terre humide	nécessite un arrosage
Capteur de pluie	il pleut	il ne pleut pas
Capteur d'inondation	sol inondé	sol sec
Détecteur de niveau de liquide	niveau OK	niveau insuffisant

La tension entre la broche 5 et la masse est supérieure à celle présente entre le point milieu du potentiomètre et la masse, la sortie revient donc à un niveau haut et la LED LD1 s'éteint.

Par conséquent, dans cette application, la LED éteinte signifie qu'il ne pleut pas et lorsqu'elle est allumée, il pleut.

Dans le cas où nous utilisons le circuit comme alarme pour une crue, par exemple, si la LED s'allume cela signifie qu'il y a de l'eau sur le sol, sinon dans le cas contraire (éteinte) le sol est sec.

Il est ainsi possible d'utiliser ce montage comme un détecteur de niveau de liquide. Le tableau 1 représente un résumé des différentes signalisations de la LED LD1 dans différentes situations.

Le circuit est alimenté avec une pile de 9 V. La tension d'alimentation doit également être adaptée en fonction de l'utilisation du circuit, par exemple avec un microcontrôleur ou un circuit logique.

Dans ce cas, la sortie DO ne doit pas dépasser 5 V voir 3,3 V. Il est possible d'alimenter le montage avec 3 piles LR03 ou LR06 en série.

Ce montage peut être utilisé, par exemple, dans une serre pour surveiller l'état des plantes et recevoir une alarme lorsqu'une ou plusieurs plantes ont besoin d'eau.

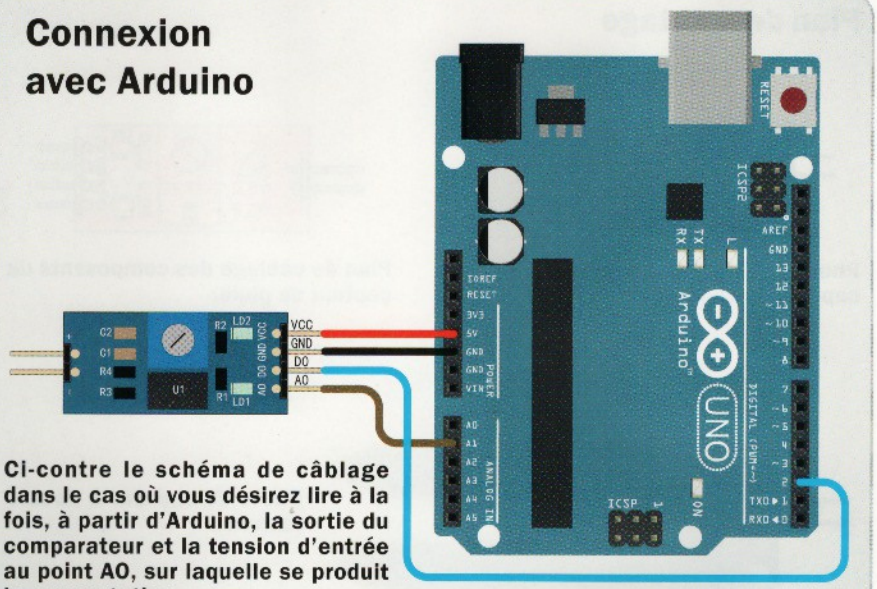
Un logiciel de monitoring peut être créé sur un PC afin d'afficher l'état de la situation.

Ce montage peut aussi détecter la présence de pluie ou d'inondations ou le niveau maximal d'un liquide dans un réservoir.

La présence de l'alimentation est signalée par l'allumage de la LED LD2 (placée sur le positif VCC), et dont le courant est limité (déterminé) par la résistance R2.

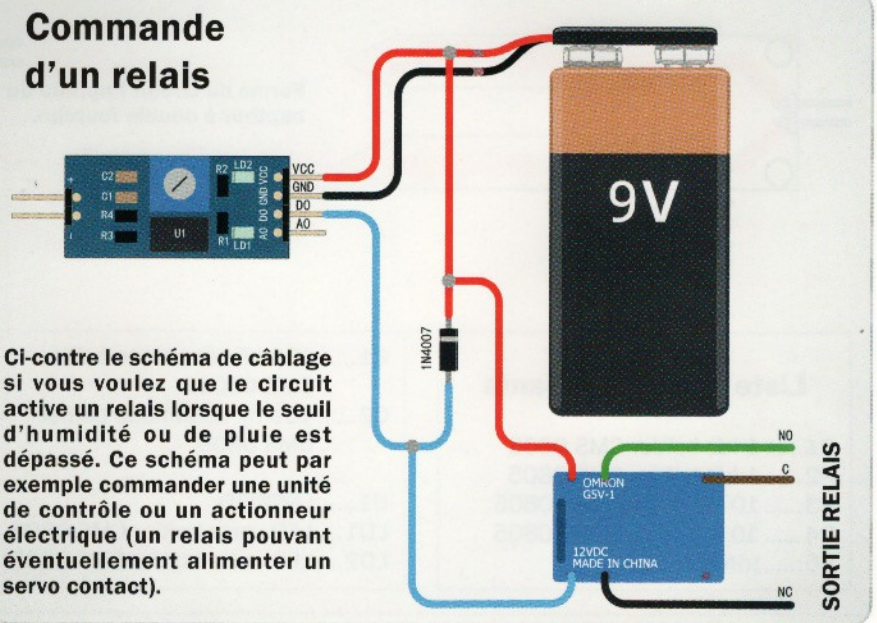
Dans le cas où le montage est alimenté par 3 piles de 1,5 V, le courant de LD2 vaut environ 4,5 mA et le double avec une pile de 9 V.

Connexion avec Arduino



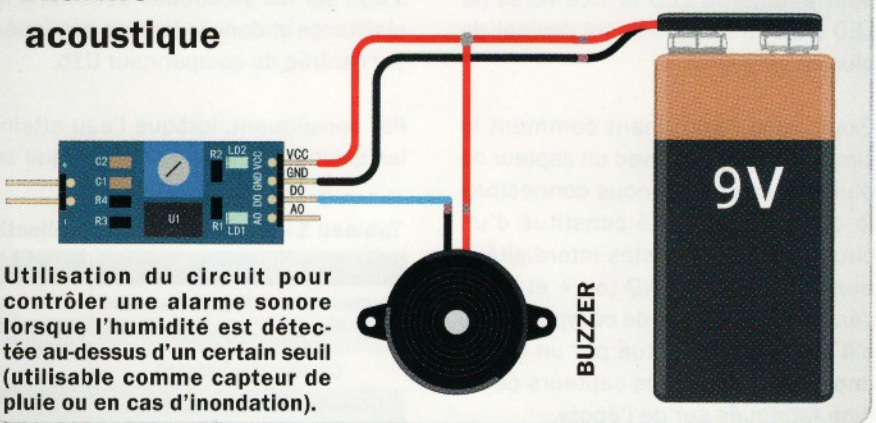
Ci-contre le schéma de câblage dans le cas où vous désirez lire à la fois, à partir d'Arduino, la sortie du comparateur et la tension d'entrée au point AO, sur laquelle se produit la commutation.

Commande d'un relais



Ci-contre le schéma de câblage si vous voulez que le circuit active un relais lorsque le seuil d'humidité ou de pluie est dépassé. Ce schéma peut par exemple commander une unité de contrôle ou un actionneur électrique (un relais pouvant éventuellement alimenter un servo contact).

Alarme acoustique



Utilisation du circuit pour contrôler une alarme sonore lorsque l'humidité est détectée au-dessus d'un certain seuil (utilisable comme capteur de pluie ou en cas d'inondation).

Réalisation pratique

Passons maintenant à la construction de la carte électronique, qui nécessite la préparation d'un circuit imprimé double face dont les typons sont téléchargeables sur notre site web dans le sommaire détaillé de la revue.

Les composants sont de type CMS, c'est-à-dire montés en surface, il vous faudra donc un peu de dextérité.

Vous aurez besoin d'un fer à souder de 20 W avec une pointe fine (0,2 mm de Ø) et de la soudure d'un diamètre maximal de 0,5 mm.

Munissez-vous d'une pince pour positionner les composants, de flux de soudure et d'une loupe afin de vérifier qu'il n'y a pas de courts-circuits entre les broches des composants (surtout au niveau de U1).

Une fois le circuit gravé et percé, commencez par souder le circuit intégré U1.

Vérifiez l'orientation du circuit, son repère en forme de « U » doit être positionné vers l'extérieur du circuit imprimé.

Ensuite, continuez en soudant les résistances, les condensateurs et les LED. Continuez avec le trimmer qui lui, est un composant traversant ainsi que les barrettes mâles.

Aux points marqués « + » et « - », insérez deux barrettes males au pas de 2,54 mm à angle droit (coudées à 90°), de manière à connecter facilement le capteur souhaité avec un connecteur approprié.

Une fois les soudures terminées, vous devez vous procurer un boîtier en plastique dans lequel vous placerez le circuit et l'alimentation, ainsi que des encoches pour visualiser les LED.

Pour ce qui est de la connexion d'un des capteurs avec la carte, notez que la longueur du fil peut facilement atteindre plusieurs mètres (nous avons effectué des tests avec un câble de 5 m, nous n'avons pas rencontré de problème).

Notez aussi que **vous pouvez vous-même vous fabriquer les capteurs**, vous trouverez les typons en téléchargement qui vous permettrons de les fabriquer sous la forme d'un circuit imprimé.

Les deux capteurs sont en fait des circuits imprimés double face et sont réalisables par photogravure classique.

Dans le plan de montage décrit dans les pages précédentes, vous pouvez voir les capteurs photographiés d'un seul côté, mais l'autre côté est identique, car les contacts pour la connexion avec l'extérieur (sol ou liquide) sont dans la même position et n'ont aucune polarité à respecter.

Si vous souhaitez utiliser le système pour communiquer les conditions de pluie à une station météo, connectez le contact DO et la masse sur l'entrée de celle-ci, en vous assurant que la tension requise soit compatible avec celle de notre montage. Adaptez éventuellement l'alimentation du circuit en respectant les limites (12 VDC) du comparateur.

La même remarque s'applique dans le cas où vous souhaitez connecter le circuit à une alarme ou à tout autre système électronique (par exemple basé sur Arduino) pour collecter des informations.

Reportez-vous aux différents schémas de câblage visibles dans la page précédente, vous y trouverez divers exemples d'applications et leurs connexions.

Les capteurs à contacts interdigités sont placés sur le sol et, lorsqu'il pleut, l'eau réunit les contacts d'un côté à ceux de l'autre, déterminant ainsi entre les électrodes une forte baisse de la résistance par rapport à une condition sèche (dans ce cas la résistance est théoriquement infinie).

La lecture de l'état du capteur s'effectue à l'aide d'une entrée digitale équipée d'une résistance de pull-up. Au repos, l'entrée se trouve à un niveau logique haut. Lorsque le capteur devient humide, la présence de l'eau introduit une résistance qui ferme le circuit aux points « + » et « - », l'entrée se trouve alors à un niveau logique bas (aux alentours de 0 V).

Rappelez-vous que pour détecter de la pluie ou une inondation, vous devez connecter aux points « + » et « - » un capteur interdigité, tandis que pour surveiller l'humidité du sol ou d'une plante, vous devez utiliser un capteur à fourche.

Le capteur à fourche est également utile si vous souhaitez créer un détecteur de niveau de liquide dans un réservoir ou un autre conteneur.

N'oubliez pas non plus que vous pouvez adapter la sensibilité du circuit à votre guise (par exemple, décider de la quantité d'eau ou d'humidité que votre circuit doit détecter, via LD1) en ajustant le potentiomètre.

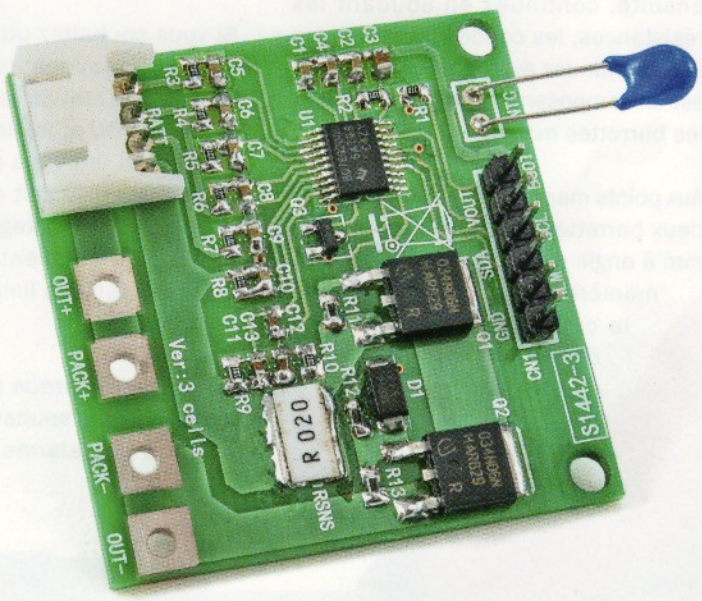
Lorsque le curseur du trimmer se trouve vers l'alimentation positive, une plus grande quantité d'eau est nécessaire pour déclencher l'alarme.

En le tournant dans l'autre sens, vers la masse, une petite quantité d'eau ou d'humidité déclenchera la condition d'alarme.

CHARGEUR DE BATTERIE LIPO

d' Alessandro Sottocornola

Nous vous proposons de réaliser un chargeur de batterie LiPo complet, basé sur le circuit intégré BQ76920, capable de gérer jusqu'à 3 cellules.



La charge des batteries au lithium, qu'elles soient de type Li-ion, LiPo ou LiFePO4, nécessite des précautions particulières qui les rendent plus difficiles et délicates que des accumulateurs tels que les batteries au plomb (simplement mises sous tension avec un limiteur de courant) ou tels que les batteries NiCd ou NiMH.

Ceci, parce que la chimie des batteries au lithium est plus exigeante que celle des autres batteries, et parce qu'elle comporte un risque d'incendie ou d'explosion si le courant maximum autorisé est dépassé ou si la température atteinte par les cellules lors d'un cycle de recharge dépasse un niveau critique.

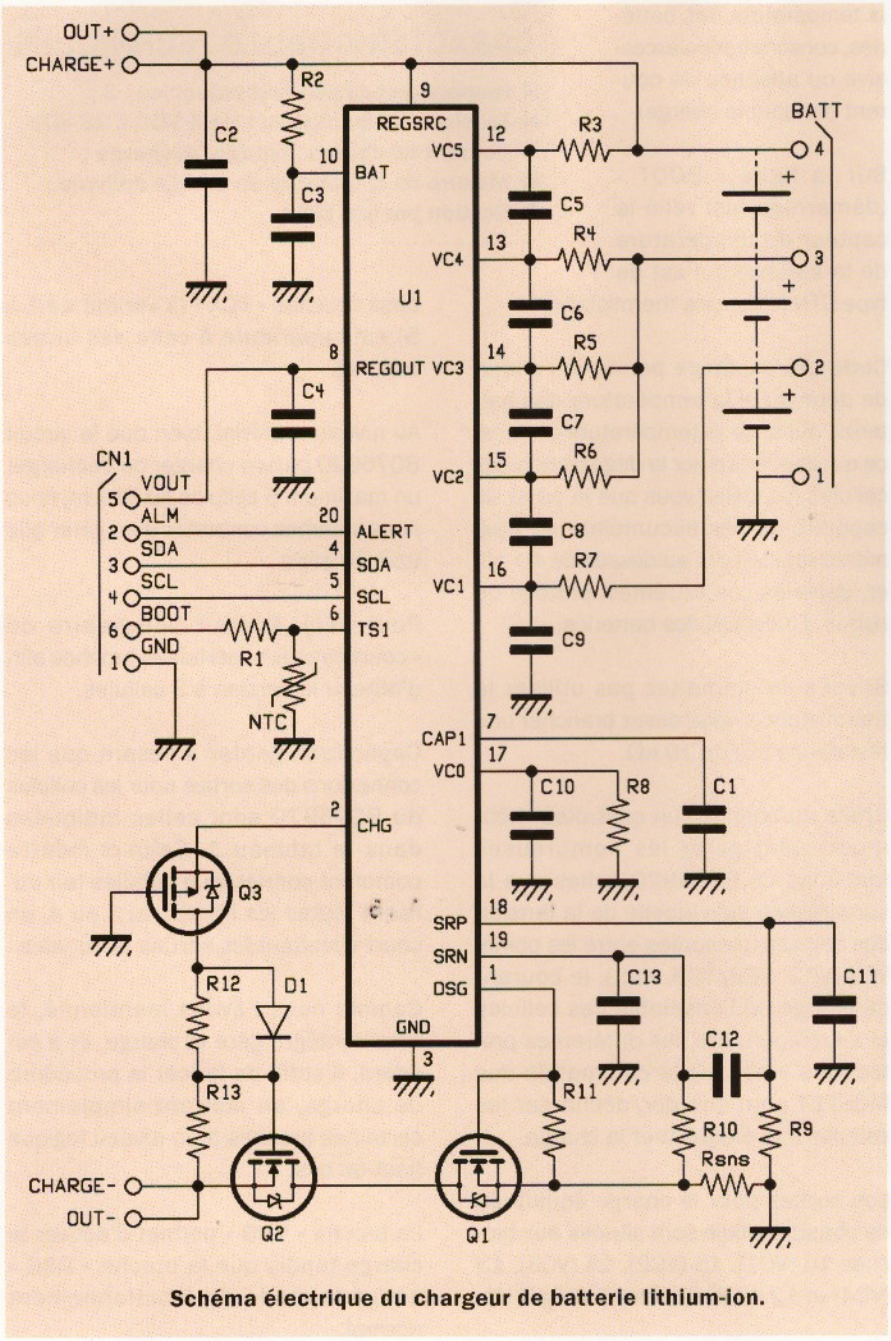


Schéma électrique du chargeur de batterie lithium-ion.

Donc, si la charge n'est pas gérée correctement, il y a des risques pour la batterie et les personnes se trouvant à proximité. Si la recharge n'est pas gérée correctement, cela entraîne une perte partielle de la capacité nominale de la batterie au lithium. De plus, si la batterie au lithium est composée de deux ou plusieurs cellules en série, il est nécessaire de limiter le déséquilibre de charge, étant donné que la tension aux bornes de chaque cellule est élevée et peut endommager un élément si elle a tendance à trop augmenter. Tout cela demande au concepteur un certain effort pour trouver la solution idéale.

Cependant le fabricant Texas Instruments, propose des circuits intégrés contenant tous les composants nécessaires pour assurer la charge optimale des batteries au lithium.

Ces circuits peuvent s'interfacer avec des microcontrôleurs ou des microprocesseurs (comme dans les ordinateurs portables, les tablettes et les smartphones) via des bus standard tels que le bus I2C ou le SM-Bus (qui est une variante) afin de surveiller la charge.

Nous avons opté pour le circuit intégré BQ76920, que nous avons utilisé dans le projet décrit dans ces pages.

Nous avons développé une carte d'évaluation de type « breakout board » qui permet d'expérimenter et/ou d'intégrer le BQ76920 dans des systèmes existants devant recharger des batteries de type Li-ion ou LiPo.

Le projet

La « breakout board » permet d'utiliser le circuit intégré qui, en fonction de la configuration matérielle, permet de charger ou de décharger 3 cellules. Ce n'est pas un hasard si le connecteur « BATT » n'a que 4 contacts: un pour chaque cellule et le commun pour la masse.

Il existe également d'autres versions du circuit intégré qui permettent de gérer jusqu'à 15 cellules, sur la base du même schéma proposé. Vous pouvez, en suivant les instructions du fabricant, réaliser un chargeur modulaire pour batterie au lithium.

Le contrôle du circuit intégré s'effectue via le bus I2C, cependant il est nécessaire de porter une attention particulière à la version utilisée car, selon le modèle, les commandes envoyées doivent avoir une adresse différente.

Dans notre projet, nous utilisons le BQ76920. Si vous expérimentez un autre circuit de la famille, reportez-vous à la documentation technique du composant.

Une particularité de notre circuit intégré est qu'il mesure et communique via le bus I2C la quantité de charge en Coulombs transférée à la batterie. Pour cette raison, nous devons tenir compte de la charge équilibrée des cellules, à partir d'une condition de batterie faible.

Nous rappelons que la quantité de charge accumulée dans une batterie, qui est généralement exprimée en ampères/heure (Ah) ou en watts/heure (Wh), peut également être exprimée en Coulomb (C), en considérant que 1C vaut :

$$Q = i \times t$$

où **Q** est la quantité de charge exprimée en Coulomb, **i** le courant de charge et

t la durée pendant laquelle la batterie est en charge. En résumé, c'est un peu comme la charge d'un condensateur.

Une autre caractéristique de notre « breakout board » est la disponibilité d'un connecteur de charge équilibrée, que le BQ76920 est bien sûr capable de gérer. Les bornes du connecteur de charge équilibrée sont reliées aux bornes VC1 à VC5 du circuit intégré.

Notre prototype permet de gérer 3 cellules. En analysant le schéma électrique, nous expliquerons comment configurer le nombre de cellules.

Schéma électrique

Examinons maintenant ce qui se passe au niveau matériel et comment fonctionne ce chargeur. Comme le montre le schéma électrique, le cœur du montage est le circuit intégré U1, à savoir la puce de chez Texas Instruments.

Le circuit est alimenté par les points « CHARGE+ » et « CHARGE- » (respectivement le pôle positif et le pôle négatif) qui sont en parallèle avec deux contacts auxiliaires qui fournissent la même tension reçue par le montage (une sorte de duplication de l'alimentation).

L'interface de contrôle externe est reliée au connecteur « CN1 », sur lequel nous retrouvons les points « SCL », « SDA » et « GND » du bus I2C. Le connecteur comporte en plus des lignes supplémentaires, telle que la ligne « ALERT » avec laquelle le circuit intégré communique avec le microcontrôleur.

Lorsqu'elle génère une impulsion à niveau logique haut, cela indique une condition d'alarme (dépassement de la température des batteries, consommation excessive ou absence de courant pendant la charge).

Sur la ligne « BOOT » (démarrage) est relié le capteur de température de la batterie qui est de type CTN (c'est une thermistance).

Cette thermistance permet au circuit de détecter si la température des batteries dépasse la température critique, ce qui peut entraîner la dégradation des cellules (rappelez-vous que la perte de capacité de ces accumulateurs augmente rapidement au-dessus de 40 °C) et, dans les cas extrêmes, protège du risque d'incendie des batteries.

Si vous ne souhaitez pas utiliser la thermistance, vous devez brancher une résistance fixe de 10 kΩ.

Grâce au bus I2C, un contrôleur hôte (host) peut gérer les nombreuses fonctions du BQ76920, telles que la surveillance individuelle de la tension des cellules (mesurées entre les points VC1/VC2, VC2/VC3, etc.), le courant de charge de l'ensemble des cellules et sa température, les différentes protections ainsi que la commande des MOSFET pour charger/décharger les cellules afin d'équilibrer la charge.

Les sorties pour la charge équilibrée de chaque cellule sont situées aux broches 16 (VC1), 15 (VC2), 14 (VC3), 13 (VC4) et 12 (VC5) du circuit intégré U1.

Entre chaque broche, en interne, se trouve un MOSFET faisant office de shunt afin de dériver une partie du courant de la cellule qui est en parallèle, lorsqu'une tension mesurée entre deux broches « VCX » (X variant de 1 à 5) est supérieure à celle des autres cellules.

Au niveau matériel, bien que le circuit BQ76920 puisse charger ou décharger un maximum 5 cellules au lithium, nous nous sommes contentés d'en gérer que trois en série.

Pour cela, il était nécessaire de « court-circuiter » certaines broches afin d'obtenir la version à 3 cellules.

Cependant, gardez à l'esprit que les connexions des sorties pour les cellules du BQ76920 sont celles indiquées dans le tableau 1. Celui-ci montre comment connecter 5 cellules (en utilisant toutes les lignes) ou 3 ou 4, en court-circuitant les sorties adéquates.

Comme nous l'avons mentionné, le circuit intégré gère la charge, et à cet égard, il suffit de lancer la procédure de charge, en mettant simplement certaines broches à un niveau logique haut ou bas.

La broche « CHG » permet d'activer la charge tandis que la broche « DSG » active le mode de fonctionnement normal.

Ces broches sont reliées aux grilles (gate) des MOSFET, qui gèrent le courant de charge/décharge de la batterie (constitué d'un pack de cellules).

CARACTÉRISTIQUES TECHNIQUES

- **Nombre de cellules rechargeables : 3 ;**
- **Tension d'alimentation : de 4 VDC à 25 VDC ;**
- **Courant de charge : charge/décharge ;**
- **Mesure de la quantité de charge délivrée ;**
- **Gestion par bus I2C.**

Cell Input	3 Cells	4 Cells	5 Cells
VC5–VC4	CELL 3	CELL 4	CELL 5
VC4–VC3	short	short	CELL 4
VC3–VC2	short	CELL 3	CELL 3
VC2–VC1	CELL 2	CELL 2	CELL 2
VC1–VC0	CELL 1	CELL 1	CELL 1

Tableau 1 : connexions et pontages à effectuer pour adapter le schéma proposé à un nombre différent de cellules.

Plan de montage

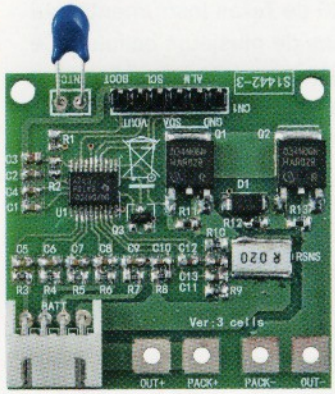


Photo de l'un de nos prototypes du chargeur de batterie lithium-ion.

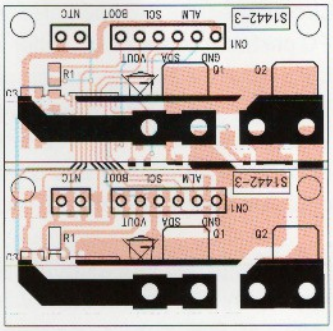


Schéma d'implantation des composants du chargeur de batterie lithium-ion.

Liste des composants

R1..... 10 kΩ boîtier CMS 0805
R2..... 100 Ω boîtier CMS 0805
R3..... 100 Ω boîtier CMS 0805
R4..... 100 Ω boîtier CMS 0805
R5..... 100 Ω boîtier CMS 0805
R6..... 100 Ω boîtier CMS 0805
R7..... 100 Ω boîtier CMS 0805
R8..... 100 Ω boîtier CMS 0805
R9..... 100 Ω boîtier CMS 0805
R10.... 100 Ω boîtier CMS 0805
R11 ... 1 MΩ boîtier CMS 0805
R12 ... 1 MΩ boîtier CMS 0805
R13 ... 1 MΩ boîtier CMS 0805
R_{SNS}..... 0, 05 Ω 3 W boîtier CMS 3015
NTC CTN 10 kΩ
C1..... 1 µF céramique 35V boîtier CMS 0805
C2..... 1 µF céramique 35V boîtier CMS 0805
C3..... 4,7 µF/25 V céramique boîtier CMS 0805
C4..... 1 µF céramique 35V boîtier CMS 0805
C5..... 1 µF céramique 35V boîtier CMS 0805
C6..... 1 µF céramique 35V boîtier CMS 0805
C7..... 1 µF céramique 35V boîtier CMS 0805
C8..... 1 µF céramique 35V boîtier CMS 0805
C9..... 1 µF céramique 35V boîtier CMS 0805
C10.... 1 µF céramique 35V boîtier CMS 0805
C11.... 100 nF/50V céramique boîtier CMS 0805
C12.... 100 nF/50V céramique boîtier CMS 0805
C13.... 100 nF/50V céramique boîtier CMS 0805
Q1..... IPD034N06N3 G
Q2..... IPD034N06N3 G
Q3..... NTR1P02T1G
U1..... BQ7692006PWR
D1..... MBRA140TRPBF

Divers

BATT connecteur JST 4 pôles au pas de 2,5mm
PACK connecteur XT60 mâle avec 10 cm de fil
OUT connecteur XT60 mâle avec 10 cm de fil
connecteur XT60 femelle avec 10 cm de fil
barrette mâle 2 pôles
barrette femelle 2 pôles

SYS_STAT (0x00)

BIT	7	6	5	4	3	2	1	0
NAME	CC_READY	RSVD	DEVICE_XREADY	OVRD_ALERT	UV	OV	SCD	OCD
RESET	0	0	0	0	0	0	0	0
ACCESS	RW	RW	RW	RW	RW	RW	RW	RW

Figure 1 : structure du registre « SYS_STAT ».

Notez que la batterie est connectée à sa propre masse et ne coïncide pas directement avec le négatif de l'alimentation. Ce dernier ne concerne que le circuit d'entrée (« CHARGE+ » et « CHARGE- »).

Par contre, le négatif de l'alimentation est commuté à la batterie lorsque les MOSFET Q1 et Q2 sont conducteurs (commandés par les broches CHG et DSG de U1). En mesurant la chute de tension aux bornes de la résistance « Rsns », le circuit intégré mesure le courant circulant dans la batterie. La mesure est effectuée à l'aide des broches « SRP » et « SRN » (respectivement positive et négative de l'entrée correspondante) du convertisseur A/N interne du BQ76920.

Nous allons maintenant décrire le fonctionnement du bloc des MOSFET Q1, Q2 et Q3, qui régit la charge et la décharge.

Commençons par la charge, la broche « CHG », qui au repos se trouve à un niveau logique bas, est portée à un niveau logique haut. Le MOSFET Q3 est connecté à la broche 2 de U1, il s'agit d'un canal P.

Le contrôleur de charge BQ76920

Le circuit intégré sur lequel repose notre chargeur appartient à la famille des chargeurs/contrôleurs de batterie BQ769x0 de Texas Instruments, qui comprend des circuits capables de gérer la recharge de batteries au lithium-ion composées de plusieurs cellules. Ce circuit intégré a été spécialement conçu pour la charge (mais aussi la décharge) de batteries qui peuvent être celles d'un PC portable, mais aussi celles d'un petit véhicule électrique comme une moto, car il permet de gérer de 3 à 15 cellules. Cela correspond respectivement à des tensions de batterie complètement chargées allant de 12,6 V à 63 V.

Une des particularités de ce circuit intégré est que, contrairement aux chargeurs de batterie au lithium classiques, il mesure et communique via le bus I2C la quantité de charge transférée à la batterie, exprimée en Coulomb (rappelez-vous qu'un coulomb vaut 1 ampère par seconde). Pour cette raison, le circuit doit lui-même gérer la charge équilibrée des cellules.

Le montage doit être alimenté par les points « CHARGE+ » et « CHARGE- » (respectivement positif et négatif) avec une tension supérieure d'au moins 3 à 4 V à celle de la charge complète des cellules en série. À cet égard, il convient de rappeler qu'une cellule au lithium-ion a une tension nominale de 3,6 V (3,7 pour le LiPo) et qu'en fin de charge sa tension atteint 4,2 V à 4,3 V.

L'interfaçage du contrôle externe s'effectue par le bus I2C à travers les lignes SCL, SDA et GND. La ligne « ALERT », avec laquelle le circuit communique avec le microcontrôleur hôte, lorsqu'elle génère une impulsion d'un niveau logique haut indique une condition d'alarme (dépassement de la température maximale de la batterie, consommation excessive ou manque de charge).

La broche « BOOT » permet de relier un capteur de température externe de type CTN sur la batterie (il s'agit d'une thermistance). Celle-ci permet au circuit de détecter si la température des batteries dépasse un seuil critique, ce qui peut entraîner la dégradation des cellules (rappelez-vous que la perte de capacité de ces accumulateurs augmente rapidement au-dessus de 40 ° C) et, dans des cas extrêmes, la batterie peut prendre feu avec des risques d'explosion. Si vous ne souhaitez pas utiliser la thermistance, vous devez connecter une résistance fixe de 10 kΩ.

Grâce au bus I2C, le microcontrôleur hôte peut gérer de nombreuses fonctions du BQ76920, telles que la surveillance de la tension des cellules de manière individuelle (mesurée entre des paires de broches VC « voisines »), le courant de charge global du pack de cellules et sa température, les protections, la commande des MOSFET pour charger/décharger les cellules afin d'équilbrer la charge.

Les sorties pour la charge équilibrée de chaque cellule sont situées sur les broches 16 (VC1), 15 (VC2), 14 (VC3), 13 (VC4), et 12 (VC5). Entre chaque paire de broches consécutives se trouve un MOSFET faisant office de shunt. Il dérive une partie du courant de la cellule (si nécessaire) qui est en parallèle, lorsque la tension mesurée par le convertisseur A/N interne du BQ76920 entre deux broches VCX est supérieure à celle des autres cellules.

Le tableau ci-dessous présente l'ensemble de la famille BQ769x0, avec les différentes fonctions, y compris l'adressage du bus I2C. Comme vous pouvez le constater, l'adresse n'est pas la même pour tous les circuits intégrés et doit être respectée pour le bon fonctionnement du montage.

TUBE	TAPE & REEL	CELLS	I²C ADDRESS (7-Bit)	LDO (V)	CRC
bq7692000PW	bq7692000PWR	3–5	0x08	2.5	No
bq7692001PW(1)	bq7692001PWR(1)				Yes
bq7692002PW(1)	bq7692002PWR(1)			3.3	No
bq7692003PW	bq7692003PWR				Yes
bq7692006PW	bq7692006PWR		0x18		No
bq7693000DBT	bq7693000DBTR	6–10	0x08	2.5	No
bq7693001DBT	bq7693001DBTR				Yes
bq7693002DBT	bq7693002DBTR			3.3	No
bq7693003DBT	bq7693003DBTR				Yes
bq7693006DBT	bq7693006DBTR		0x18		No
bq7693007DBT	bq7693007DBTR				Yes
bq7694000DBT	bq7694000DBTR	9–15	0x08	2.5	No
bq7694001DBT	bq7694001DBTR				Yes
bq7694002DBT	bq7694002DBTR			3.3	No
bq7694003DBT	bq7694003DBTR				Yes
bq7694006DBT	bq7694006DBTR		0x18		No

Listing 1

```
#include <bq769x0.h>              // Librairie pour le circuit intégré de gestion de batterie Texas Instruments BQ76920
#define BMS_ALERT_PIN 2           // Broche connectée à ALN (alarme)
#define BMS_BOOT_PIN 7            // Broche BOOT connectée à NTC
#define BMS_I2C_ADDRESS 0x08      // Adresse I2C du circuit intégré
#define CC_CHANGED_PIN 13         // Sortie indiquant qu'une nouvelle valeur en Coulomb est disponible. Non utilisé
bq769x0 BMS(bq76920, BMS_I2C_ADDRESS);  // Déclaration de la gestion de la batterie
float num_coulomb=0;
boolean CHARGE_ON=false;                // Faux : pas de charge en cours
const float MIN_TENS_CHARGE = 11000;    // Tension minimale en dessous de laquelle la charge a lieu (somme des cellules)
void setup()
{
 pinMode(BMS_ALERT_PIN, INPUT);
 pinMode(CC_CHANGED_PIN, OUTPUT);
 digitalWrite(CC_CHANGED_PIN,LOW);
 int err = BMS.begin(BMS_ALERT_PIN,BMS_BOOT_PIN);
 Serial.begin(9600);
 BMS.setTemperatureLimits(-20, 45, 0, 45);
 BMS.setShuntResistorValue(5);
 BMS.setShortCircuitProtection(14000, 200);                  // délai en µs
 BMS.setOvercurrentChargeProtection(8000, 200);              // délai en ms
 BMS.setOvercurrentDischargeProtection(8000, 320);           // délai en ms
 BMS.setCellUndervoltageProtection(2600, 2);                 // délai en s
 BMS.setCellOvervoltageProtection(3750, 2);                  // délai en s
 BMS.setBalancingThresholds(0, 3300, 20);                    // minIdleTime_min, minCellV_mV, maxVoltageDiff_mV
 BMS.setIdleCurrentThreshold(100);
 BMS.enableAutoBalancing();                                  // Activation de l'équilibrage de la charge de la batterie
 BMS.setBatteryCapacity(2200);                               // Capacité de la batterie, utilisée pour le calcul en cours
 }

void loop()
{
 int dato;
 byte registro;
 BMS.update();                  // Cette fonction doit être appelée toutes les 250 ms pour obtenir les indications correctes
 Serial.print("Vbat: ");        // Visualise la tension de chaque cellule
 Serial.print(BMS.getCellVoltage(1));
 Serial.print(" - ");
 Serial.print(BMS.getCellVoltage(2));
 Serial.print(" - ");
 Serial.print(BMS.getCellVoltage(5));
 Serial.print(" = ");
 Serial.print(BMS.getBatteryVoltage());
 Serial.print("mV - ");
 Serial.print("Temp: ");        // Visualise la température
 Serial.println(BMS.getTemperatureDegC(1));
 Serial.print("CHARGE ");       // Indique si la charge est en cours ou non
 Serial.println(CHARGE_ON);

 // Si la charge n'est pas en cours et si la tension tombe en dessous de MIN_TENS_CHARGE, la charge est activée.
if (CHARGE_ON==false)
 {
  if (BMS.getBatteryVoltage() < MIN_TENS_CHARGE) CHARGE_ON=true;
 }
 // Si la charge n'est pas en cours, le fonctionnement normal est activé
```

```
  else
  {
   if (BMS.enableCharging()==false)
   {
    CHARGE_ON=false;
    BMS.enableDischarging();
    delay(500);
   }
  }
   // Lecture de la valeur Coulomb si une nouvelle donnée est disponible, à utiliser si vous souhaitez lire le registre de manière
autonome sans passer par la librairie
   Serial.print("Coulomb: ");
  Serial.print(BMS.getSOC());
  Serial.print(" / Corrente: ");
  Serial.println(BMS.getBatteryCurrent());
  delay(250);
}
```

Il entre donc en conduction car sa grille est reliée à la masse, c'est-à-dire à un potentiel inférieur à celui de la source.

Ainsi, Q3 porte alors sa tension de drain sur la grille de Q2, qui est un canal N .Elle est polarisée par la diode D1 qui court-circuite la résistance R12.

Comme la source de Q2 est à la masse, le potentiel sur sa grille le fait entrer en conduction (état ON) et son drain dérive à la masse celui du MOSFET Q1 qui, comme la broche DSG du BQ76920 à un niveau logique 0, passe en conduction car le potentiel de sa source est positif par rapport à celui de la broche 1.

En effet, la masse de la batterie est plus positive que le potentiel du point « CHARGE- ».

Ainsi, la charge de la batterie commence, elle est équilibrée et régulée par le circuit U1 en fonction des réglages imposés par le bus I2C.

Le courant traverse la résistance « Rsns » qui permet ainsi de mesurer en temps réel la consommation de la batterie et donc le courant fourni à cette dernière. Cela permet de réguler l'intensité de charge et de la communiquer via le bus I2C.

Ainsi, il est possible de contrôler les différentes phases de la charge, qui, rappelons-le, pour les batteries au lithium-ion est divisée en trois phases :

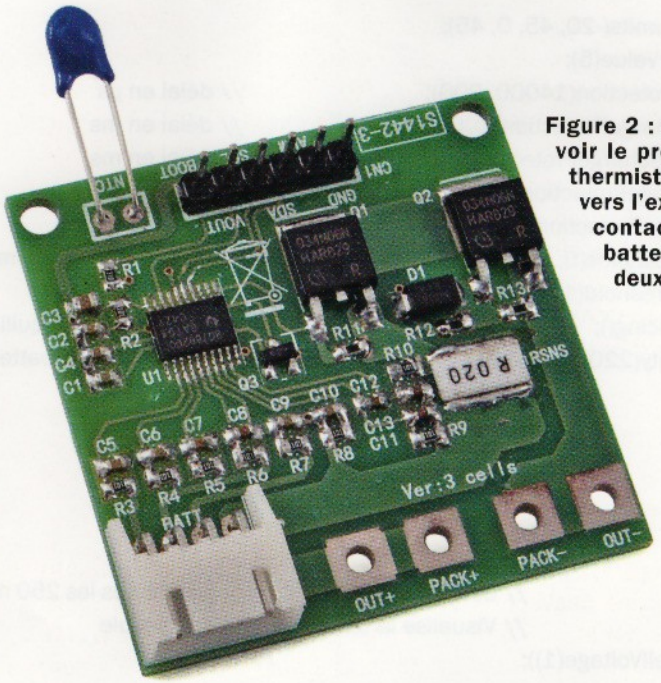


Figure 2 : ci-contre vous pouvez voir le prototype assemblé. La thermistance doit être montée vers l'extérieur et doit être en contact (thermique) avec la batterie. Elle est reliée par deux fils au circuit imprimé.

1. Une phase de pré-conditionnement durant laquelle le courant est progressivement augmenté, surtout si la tension mesurée aux bornes des cellules est très basse ;

2. La seconde phase correspond à une charge à courant constant jusqu'à atteindre une tension de 4 V par cellule ;

3. La troisième phase correspond à une charge à tension constante, pour atteindre une valeur d'environ 4,2 V.

De plus, mesurer le courant qui passe à travers la résistance « Rsns » permet de connaître la quantité de charge emmagasinée dans la batterie pendant la phase de charge, en effectuant la moyenne du courant et en multipliant la valeur par le temps écoulé.

Une fois la charge terminée, la broche 2 du circuit U1 repasse à un niveau logique 0, de sorte que le MOSFET Q3 est bloqué entrainant les blocages des MOSFET Q2 et Q1.

Quant à la décharge, elle est effectuée pour préparer la batterie à une nouvelle charge, en déchargeant éventuellement les cellules de la même manière pour les équilibrer. Dans ce cas également, la résistance « Rsns » permet de

détecter le courant et de déterminer la quantité de charge soustraite (enlevée) à la batterie.

Pendant le cycle de charge, il est possible de surveiller le bit « CC_READY » du registre « SYS_STAT » (voir la figure 1) pour savoir si de nouvelles données en Coulomb sont présentes.

Si une nouvelle donnée est disponible, il suffit de lire les registres « CC_HI » et « CC_LO » et d'ajouter cette valeur à celle précédemment lue pour obtenir le nombre total de la charge en Coulombs stockée dans la batterie.

Grâce au registre d'état, il est également possible d'avoir d'autres informations, par exemple si le circuit intégré est prêt pour effectuer une lecture, ou s'il y a des erreurs ou des alarmes, telles qu'une température trop élevée, et bien plus encore (reportez-vous à la documentation technique).

C'est précisément à travers ce registre qu'il est possible de surveiller la température des batteries dès lors qu'un capteur externe de type CTN d'une valeur de 10 kΩ est connecté (voir la figure 2).

Dans notre cas, la CTN doit être reliée au connecteur dénommé « NTC ».

Si cette thermistance est présente dans le circuit, la température peut être mesurée à l'aide d'un registre spécial et des notifications d'alarme peuvent être reçues via le registre d'état.

La librairie Arduino

Comme mentionné précédemment, le BQ76920 doit être géré via le bus I2C et ce mode de fonctionnement fait partie du monde d'Arduino.

Nous avons créé et mis à disposition une librairie pour Arduino qui gère cette « breakout board » à l'aide du bus I2C.

La librairie que nous vous proposons est téléchargeable dans le sommaire détaillé de la revue (en bas de page).

Cette librairie implémente la quantité de charge (en Coulombs) qui est transférée à la batterie. D'autre part, elle vous permettra d'exécuter les fonctions du projet avec Arduino. Le programme de test des différents modes de fonctionnement du BQ76920 est reporté au Listing 1.

Réalisation pratique

Passons maintenant à la réalisation pratique du projet, c'est-à-dire la construction du circuit imprimé. Il s'agit d'un circuit double face.

Nous mettons en téléchargement dans le sommaire détaillé de la revue les typons ainsi que les fichiers Gerber pour une fabrication professionnelle.

Les composants sont tous de type CMS (montage en surface), à l'exception du connecteur CN1 et de la thermistance (qui doit être montée à l'extérieur pour être en contact avec le bloc de batterie).

Le soudage des composants requiert un minimum d'équipement et de soins. Il vous faudra un fer à souder à pointe très fine (0,2 mm) avec de la soudure pour composants CMS de 0,5 mm de Ø.

D'autres part, pour manipuler les composants vous aurez besoin de pinces Bruxelles, d'un flux pour faciliter le soudage, ainsi que d'une loupe éclairée si possible.

Vous devez souder en premier le circuit intégré U1 en utilisant un peu de colle pour le maintenir. Son repère en forme de « U » doit se situer vers les transistors Q1 et Q2 (vers la droite vue de dessus).

Soudez en premier une patte, laisser refroidir puis soudez la patte diagonalement opposée. Continuez ainsi de suite en laissant refroidir à chaque soudure.

Vérifiez soigneusement que de la soudure ne se soit pas introduite sous le composant ou que des ponts de soudure soient présents entre 2 pattes successives.

Notez que le flux permet d'éviter ce dernier phénomène en évitant que la soudure ne joigne une patte voisine.

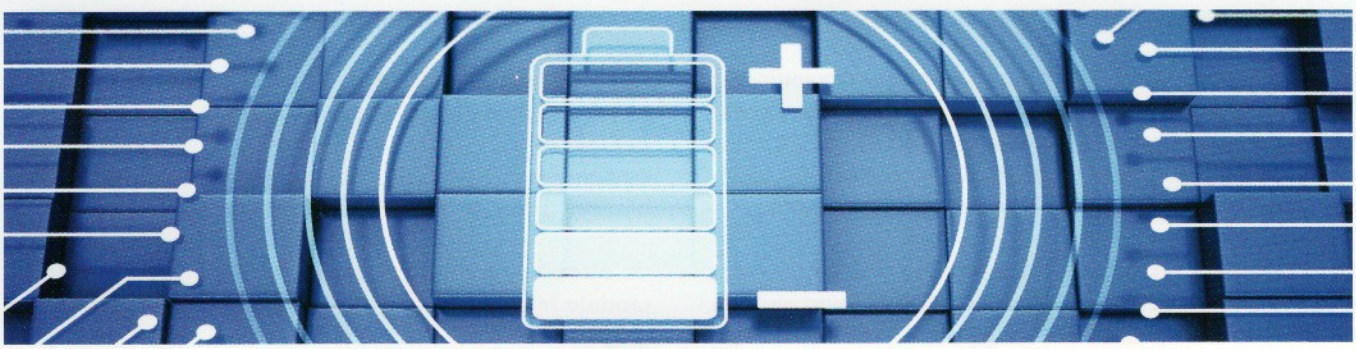
Une fois le circuit intégré soudé, continuez avec les autres composants restants, en respectant la polarité de la diode.

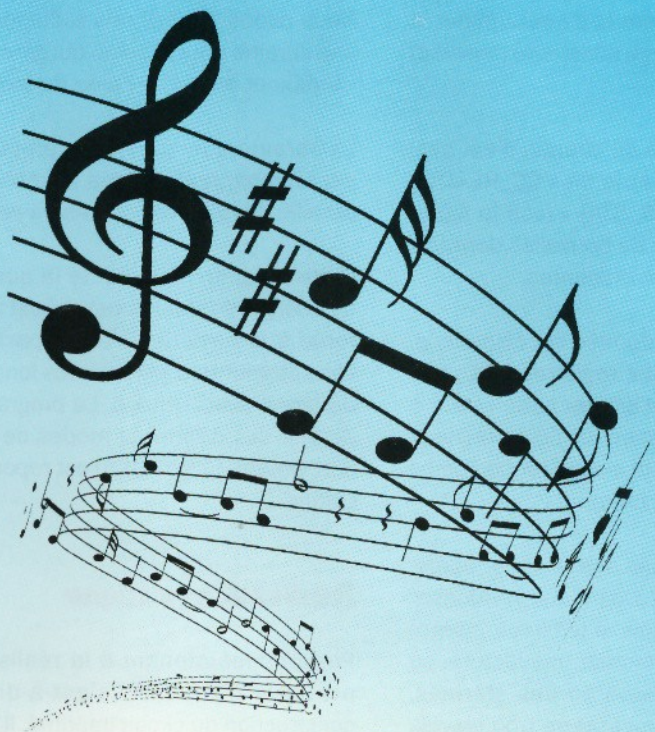
Les transistors ne peuvent être soudés que d'une seule manière. Enfin, terminez en soudant le connecteur « BATT » et la barrette mâle.

Une fois les soudures terminées, vous devez vérifier soigneusement avec une loupe que tous les composants sont correctement positionnés et soudés.

Le matériel présenté dans cet article est disponible sur le site www.futurashop.it. La « breakout board » est disponible sous la forme d'un kit (référence FT1442) monté et testé avec le BQ76920.

Vous y trouverez aussi la batterie LiPo 7,4V 2000mAh (référence : 25C2000MAH7V4) et la batterie LiPo 11,1V 2000mAh (référence : 25C2200MAH11V1). ■

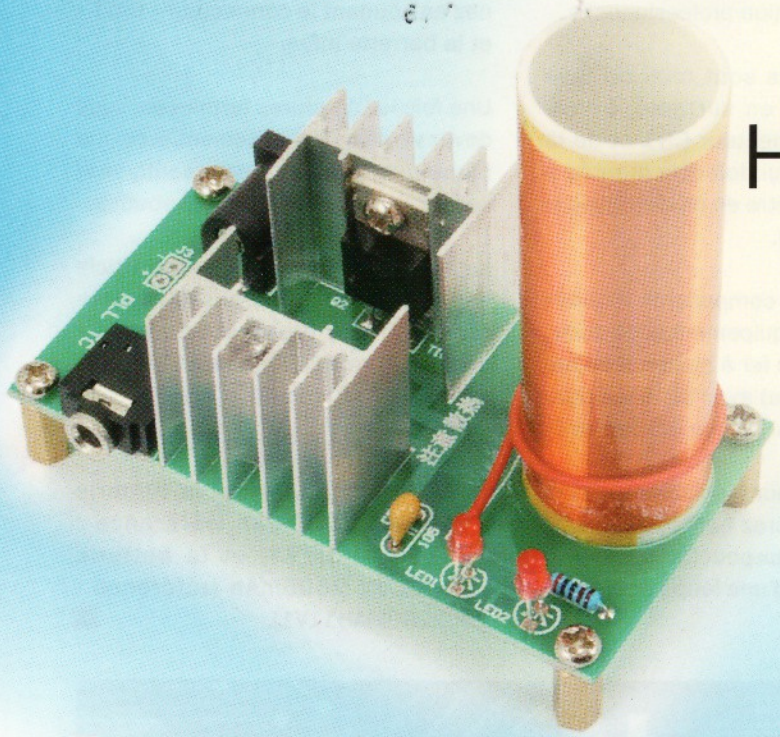




Dans cet article, nous vous proposons de réaliser un montage vous permettant d'expérimenter le monde fascinant de la haute tension, avec enthousiasme et une pincée d'attention !
Ce montage dispose d'une entrée audio qui permet de moduler l'arc électrique en fonction du signal audio.

de Boris Landoni

HAUT-PARLEUR TESLA



Les expériences avec la haute tension sont toujours très intéressantes et fascinantes, car elles nous permettent d'observer des phénomènes physiques qui sont des répliques dans le laboratoire de celles auxquelles nous assistons dans la nature. Un exemple pour tous est la foudre.

Dans ces pages, nous souhaitons décrire un circuit permettant d'expérimenter les effets de la haute tension en toute sécurité, en observant des phénomènes suggestifs et instructifs.

Ce circuit permet de réaliser une **bobine Tesla** miniature qui, en générant une haute tension, crée un petit éclair qui permet par exemple d'allumer une lampe au néon simplement en l'amenant près de la bobine ou encore en ionisant l'air ambiant avec une odeur caractéristique de ce gaz.

Le **circuit comprend** également une **entrée audio qui module le générateur en allumant et éteignant l'arc au rythme de la musique**.

Le signal audio appliqué sur l'entrée module l'amplitude de la haute tension produite, agissant ainsi sur le champ électrostatique présent dans le secondaire du transformateur haute tension et faisant donc « vibrer » l'air.

Le circuit peut donc réaliser efficacement ce que l'on appelle dans la technique du son un **haut-parleur électrostatique**. Sur l'entrée BF, vous pouvez connecter votre smartphone, un lecteur MP3, la sortie audio de l'ordinateur, etc.

Le schéma électrique

Analysons le schéma électrique du générateur haute tension proposé dans ces pages.

Le circuit génère une **tension pulsée** dont la **fréquence est déterminée** par les oscillations d'un **oscillateur** à transistors. Ce dernier est formé par un transistor MOSFET à canal N et un transistor BJT de type NPN.

En se référant au schéma électrique, expliquons son fonctionnement à partir de la mise sous tension.

Initialement, le MOSFET **Q1** (un IRF530 dont la tension V_{DS} est de 100 V avec un courant de drain de 30 A) ainsi que le transistor BJT **Q2** (un NPN de type TIP41 avec une tension V_{CE} de 100 V et une puissance maximale de 40 W) **sont dans un état bloqué**.

Le condensateur **C2**, que nous supposons être déchargé lorsque le circuit est allumé, **met à la masse la source** du MOSFET et **fait circuler un courant** dans le **diviseur de tension** formé par les résistances **R1** et **R3** afin d'**alimenter la grille de Q1**.

L'IRF530 devient conducteur et passe alors à l'état « ON », et alimente ainsi l'étage sous-jacent appartenant au transistor NPN. La résistance R4 polarise la LED rouge LED1, aux extrémités de laquelle se trouve une **tension suffisante** pour polariser la **jonction base-émetteur** du transistor **Q2**.

À ce stade, le **transistor passe de l'état bloqué à l'état de saturation** et le potentiel de son collecteur, comme celui situé à **l'extrémité supérieure du primaire** du transformateur de Tesla, se trouve à un potentiel initialement **légèrement supérieur à la masse**, alors que la grille du MOSFET est maintenue approximativement au potentiel de la ligne d'alimentation positive du circuit.

Cette **condition** permet à **Q1 de devenir passant** et la source de celui-ci fournit à la bobine et au TIP41 tout le courant dont il est capable de fournir (comme un interrupteur qui se ferme) de telle sorte qu'une tension apparaisse aux extrémités du primaire L1.

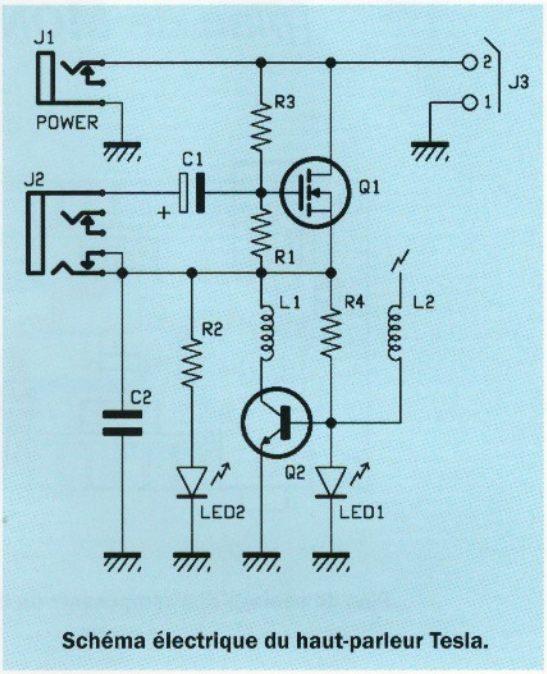


Schéma électrique du haut-parleur Tesla.

L'amplitude de la tension est inférieure d'environ 1,5 V à la tension d'alimentation du circuit.

La **tension ainsi appliquée au primaire induit au secondaire du transformateur** (L2) une tension de polarité opposée dont la **valeur est égale au rapport entre les spires**, soit **L2 / L1**.

Plus précisément, la tension générée « V_{ht} » aura une amplitude égale à :

$$V_{ht} = V_{L1} * L2 / L1$$

où V_{L1} est la tension que le TIP41 génère sur le primaire.

En supposant que le circuit soit alimenté avec une tension de 13,5 VDC, nous obtenons environ 12 V sur le primaire L1. Pour L1, nous enroulons deux spires (soit 6 V/spire) et pour le secondaire nous enroulons 350 spires, nous aurons sur le secondaire L2 : 350 * 6 : **2,1 kV** ou 2100 V !

Si la tension d'alimentation monte à 20 VDC, la tension aux extrémités du primaire sera d'environ 18 V et, en travaillant cette fois à 9 V/spire, la tension aux extrémités du secondaire sera de 3,15 kV.

Pour augmenter encore la tension sur le secondaire du transformateur, il est possible de réduire le nombre de spire au primaire, en n'utilisant qu'une seule spire. Dans ce cas, les tensions doublent théoriquement.

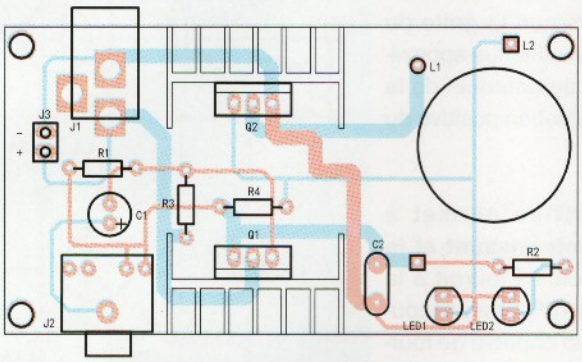
C'est théorique car le courant augmente et avec lui les chutes de tensions dans le transistor et dans les pistes du circuit imprimé, sans oublier que, avec une seule spire, le champ magnétique n'est pas distribué uniformément dans la bobine, ce qui augmente les pertes dans celle-ci. Sur les photos du prototype, vous vous apercevez que pour le primaire, nous avons utilisé qu'une seule spire.

Continuons l'analyse du circuit, une fois que Q2 a alimenté la bobine, le transistor BJT NPN est bloqué en raison de la chute de tension sur sa base, car en régime permanent, c'est-à-dire lorsque la **charge transitoire** de l'inductance représentée par la bobine L1 est **terminée**, la différence de potentiel entre le drain et la source du MOSFET (qui

CARACTÉRISTIQUES TECHNIQUES

- **Tension d'alimentation : de 12 VDC à 24 VDC ;**
- **Courant maximal : 2 A ;**
- **Haute tension générée : 2,1 kV à 13 VDC ;**
- **Dimensions : 76 * 80 * 40 (en mm) ;**
- **Entrée audio : jack 3,5 mm ;**
- **Amplitude maximale du signal audio : 2 V_{eff}.**

[plan de MONTAGE]



Plan de câblage des composants du haut-parleur Tesla.

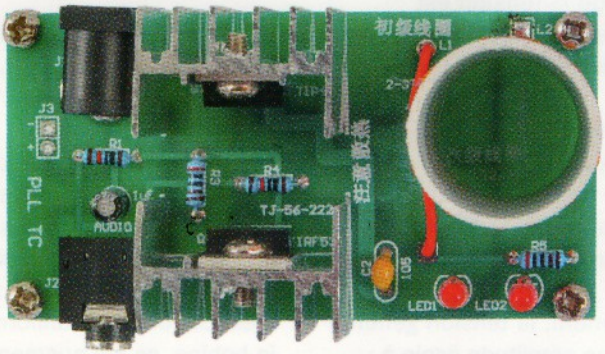


Photo de l'un de nos prototypes du haut-parleur Tesla.

Liste des composants

R1..... 10 kΩ 1%
R2..... 2 kΩ 1%
R3..... 2 kΩ 1%
R4..... 10 kΩ 1%
C1..... 1 µF / 50 V électrolytique
C2..... 1 µF céramique
L1 ..... voir texte
L2 ..... voir texte

Q1..... IRF530
Q2..... TIP41
LED1.. LED 3mm rouge
LED2.. LED 3mm rouge
J1..... fiche alimentation
J2..... connecteur jack stéréo 3,5mm

Divers :
Dissipateur (x2)
Vis 10 mm 3 MA (x2)

fonctionne comme un commutateur statique) devient insuffisante pour le maintenir dans un état passant (« ON »).

Ainsi, Q1 tend à se bloquer, privant ainsi Q2 et l'inductance de la tension d'alimentation.

L'induction a tendance à réagir en maintenant le courant pendant quelques instants, elle **génère une surtension inverse** qui est absorbée par la diode de protection interne du MOSFET.

Lorsque Q1 est bloqué, la tension qui arrive sur la base de Q2 est suffisante pour le faire entrer en conduction. Ainsi, la bobine L1 est alimentée et génère une nouvelle impulsion, le cycle est relancé.

Nous avons donc un **phénomène cyclique qui génère aux extrémités du secondaire** (haute tension) des **impulsions** de fréquence égales à celles présentes dans l'enroulement primaire.

De plus, lorsque Q2 se trouve dans un état saturé, L1 se trouve en parallèle avec C2, ce qui implique un phénomène oscillatoire qui augmente le pic de tension généré par la bobine de Tesla, permettant d'obtenir beaucoup plus que les valeurs théoriques calculées, c'est-à-dire des tensions au secondaire qui peuvent atteindre des milliers de volts (le courant reste très faible).

La LED **LD2** est alimentée à travers la résistance R5, elle indique lorsqu'elle est allumée, que le **montage est sous tension**. Une deuxième LED est placée dans le circuit d'enroulement secondaire et s'allume lorsqu'une décharge est produite.

Etudions maintenant comment une modulation haute tension peut être obtenue à partir d'un signal basse fréquence variable tel qu'un signal audio provenant d'un lecteur MP3, un récepteur radio, un smartphone, etc. Ce signal arrive sur la prise stéréo 3,5 mm du circuit (J2), dont l'un des canaux est mis à la masse, tandis que l'autre n'est pas connecté.

Par contre, la masse du connecteur est reliée au condensateur C1, qui bloque la composante continue du signal audio et laisse passer la composante alternative, nous obtenons ainsi une **tension variable** constituant le signal utilisé.

L'inversion des connexions par rapport à ce qui semblerait logique vise à éviter que la masse de l'entrée audio ne soit continuellement couplée au circuit.

La séparation est valable aussi dans le cas d'une connexion avec un PC ou un autre appareil BF alimenté en 230 VAC. Il est probable que son boîtier soit relié à la terre. Dans notre circuit, **le signal est appliqué sur un point intermédiaire de l'alimentation**, il est bon que **la masse ne soit pas couplée** en continu, afin d'éviter des bouclages par la terre, avec des conséquences inévitables.

Lorsqu'un signal, variant dans le temps, est appliqué sur le condensateur électrolytique C1, il passe vers la grille du MOSFET en se **superposant à la tension de polarisation**. Cela entraine une variation de la polarisation qui est fonction de l'amplitude du signal analogique.

Il en résulte une modification analogue du courant délivré à l'étage composé de Q2 et donc une **modulation de l'amplitude de la tension au primaire de L1**, suivie d'une variation identique proportionnelle aux extrémités du secondaire (haute tension).

Si l'extrémité libre du secondaire du transformateur se décharge dans l'air, le « vent » électrostatique qu'elle produit se déplace de la même manière que la variation d'amplitude du signal audio, de sorte qu'en connectant cette extrémité (évidemment lorsque le circuit est éteint !) à une plaque métallique, cela permet de reproduire le son correspondant au signal appliqué sur l'entrée audio, c'est-à-dire sur la prise J2.

Expérimentez, mais avec précaution !

Le circuit fonctionne sous une haute tension et son utilisation requiert donc les précautions suivantes : ne pas toucher le circuit imprimé, ni les composants pendant son fonctionnement et pendant quelques minutes après sa mise hors tension. Cela, afin d'éviter de subir un choc douloureux pour la plupart des personnes. Pour un patient cardiaque, le choc peut être dangereux.

Ne placez pas de téléphones portables, de lecteurs MP3 et autres équipements électroniques à proximité de la bobine, car celle-ci génère un champ électromagnétique à haute fréquence qui pourrait provoquer des interférences et les endommager.

Bien que le courant délivré par le transformateur soit très faible (autour du milliampère), la tension générée est très élevée et peut provoquer un choc douloureux. En se rapprochant de l'extrémité libre du secondaire, tout corps peut déclencher une décharge, ce phénomène peut provoquer une sensation de brûlure.

Après une utilisation prolongée, vous ne devez pas toucher les transistors TIP41 et MOSFET, car leur température peut être très élevée, en particulier si le circuit est alimenté en 24 V.

L'ensemble du circuit est alimenté via le bornier d'alimentation J3 ou la prise d'alimentation nommée J1 (les deux sont connectées en parallèle).

Réalisation pratique

Abordons maintenant la construction du générateur haute tension.

La première des choses à faire est de préparer le circuit imprimé double face dont vous pouvez télécharger les typons sur notre site dans le sommaire détaillé de la revue les typons des circuits imprimés.

Une fois le circuit imprimé gravé et percé, commencez par souder comme d'habitude les composants ayant un profil bas.

Commencez par les résistances, ensuite soudez le condensateur non polarisé C2, puis la prise audio et le condensateur électrolytique C1.

Continuez avec la prise d'alimentation J1 et le bornier J3 (si vous préférez alimenter le circuit à partir de celui-ci).

Ensuite, soudez les deux LED en les orientant correctement, leurs méplats doivent se situer du côté de la bobine.

Continuez en soudant les transistors IRF530 et TIP41, orientez leurs parties métalliques vers l'extérieur du circuit imprimé. Fixez, sur chaque transistor, un radiateur de 13 ° C/W, en interposant de la pâte thermique pour faciliter la dissipation de chaleur produite pendant le fonctionnement.

Les composants nécessaires à la réalisation du circuit sont faciles à trouver dans le commerce, à l'exception du transformateur, qui s'apparente à une self à air.

Pour cela, utilisez un support cylindrique en plastique ou en papier d'un diamètre de 20 mm et d'une longueur de 60 mm. Par exemple, un morceau de tuyau en PVC convient.

Bobinez 350 tours de fil de cuivre émaillé de 0,2 mm de diamètre, que vous bloquerez avec du ruban adhésif.

Vous venez de fabriquer le secondaire HT du transformateur (haute tension).

Grattez délicatement le fil émaillé aux extrémités avec un cutter afin de retirer l'isolant et faciliter le soudage sur le circuit imprimé. L'autre extrémité restera libre, elle sert d'électrode.

Pour l'enroulement primaire, vous devez bobiner une ou deux spires de fil rigide isolé de 0,5 mm de diamètre, directement sur le secondaire HT.

Les extrémités du primaire doivent ensuite être soudées aux endroits respectifs du circuit imprimé, marqués L1.

Une fois les enroulements du transformateur terminés, vous devez le fixer sur le circuit imprimé en collant la base avec de la colle forte. Cela assurera une certaine stabilité du transformateur et l'empêchera de tomber sur le reste du circuit, en causant des dommages.

Le circuit imprimé sera disposé dans un boîtier en plastique (ou en bois) pour plus de sécurité. Faites une encoche pour laisser passer l'extrémité libre du secondaire du transformateur, de manière à favoriser la décharge pour vos expérimentations.

Les expériences

Passons maintenant à la partie la plus intéressante du projet, à savoir l'expérimentation du générateur haute tension. Procurez-vous une alimentation secteur fournissant une tension continue de 15 VDC en sortie et pouvant fournir un courant de 2 A.

Munissez-vous d'un câble se terminant par une fiche coaxiale ayant un diamètre adapté à la prise J1 du circuit. La partie centrale de la prise correspond au positif, la partie externe au négatif, vérifiez bien la polarité avant la mise sous tension.

Le fil de l'extrémité du secondaire laissée libre doit être sortie de telle sorte qu'en alimentant le circuit, elle produise un halo lumineux dû à une décharge de type « **corona** » qui provoque l'**ionisation de l'air** et peut entraîner la production d'ozone.

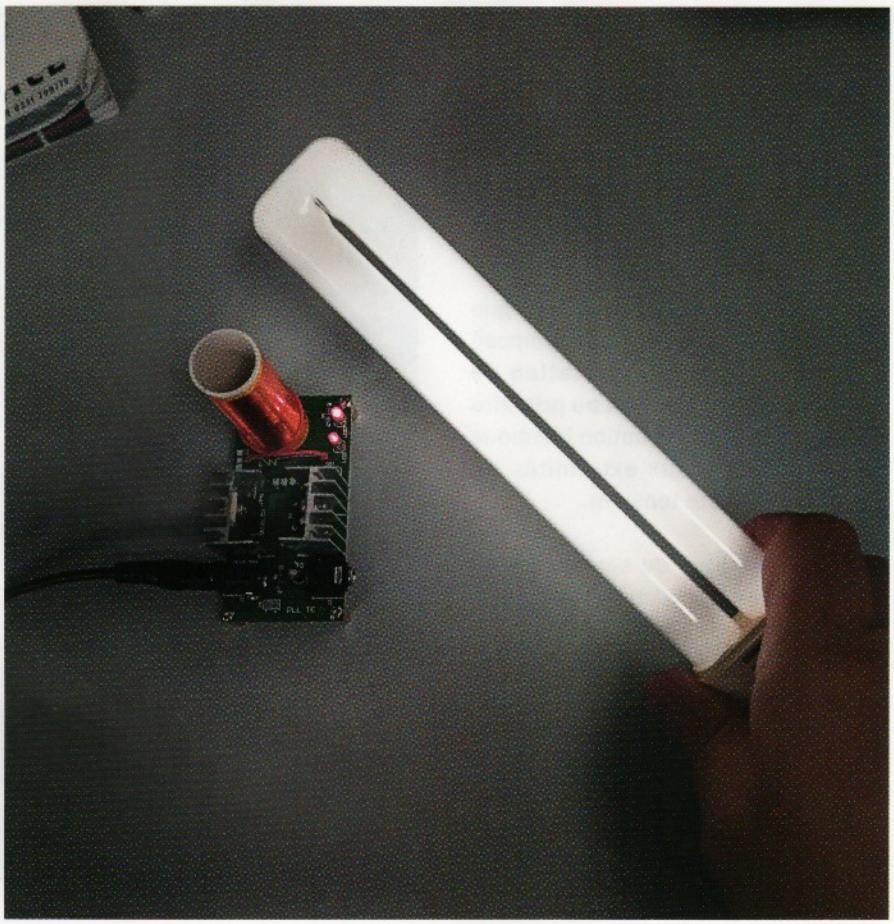
L'effet « corona », aussi appelé « effet couronne » ou « effet de couronne », est un phénomène de décharge électrique partielle entraînée par l'ionisation du milieu entourant un conducteur.

Les champs électrostatiques libérés de l'extrémité du fil sont visibles sous la forme d'un halo en forme de cône avec une lumière très intense près du fil et de plus en plus sombre en s'éloignant.

La production de la décharge sera associée à un bruit caractéristique semblable à un « souffle rythmique ».

Les deux phénomènes seront plus accentués si la masse du circuit est connectée à une plaque métallique placée sur le plan du support du générateur, étant entendu qu'elle doit rester à au moins dix centimètres de l'électrode libre du secondaire du transformateur.

En plaçant une sphère métallique au-dessus du transformateur ou en l'appuyant contre le support des enroulements après avoir replié l'extrémité libre de la haute tension secondaire afin de la placer entre le haut du support et la sphère (qui l'écrase), en relâchant cette dernière, vous apercevrez une décharge électrostatique uniforme,



En approchant un tube ou une lampe au néon, il s'allume presque par magie.

et dans l'obscurité, il sera possible de voir autour un halo bleu pâle.

En approchant du générateur une ampoule au néon vous la verrez s'allumer, de manière plus faible que si elle était alimentée normalement. Ce phénomène est dû à l'ionisation du gaz contenu dans l'ampoule par le champ électrique généré par le transformateur.

En appliquant **un signal BF sur l'entrée audio**, vous pouvez **moduler le champ électrostatique** et obtenir le son correspondant à partir d'une sphère ou d'une plaque métallique.

Pour cela, prenez un câble audio stéréo avec deux prises jack de diamètre 3,5 mm aux extrémités et insérez une extrémité dans la prise de sortie audio d'un lecteur MP3, d'un smartphone, d'une tablette etc., puis insérez l'autre côté de la prise dans l'entrée audio du circuit.

Commencez la lecture audio et allumez le générateur haute tension.

En faisant varier le volume, vous pourrez entendre le faible son généré par la modulation du champ électrostatique (c'est-à-dire le « vent » électrique) en fonction du signal audiofréquence.

Il est entendu **qu'en cours d'utilisation, vous ne devez pas approcher de trop près votre oreille de la bobine ou de la partie métallique pour essayer d'entendre le son**, sinon vous risquez de **subir un choc électrique** qui, bien que théoriquement inoffensif, est douloureux. Si vous n'entendez pas bien, essayez d'augmenter le niveau du signal d'entrée.

À cet égard, nous soulignons que la sensibilité de l'entrée audio est de quelques volts, en ce sens qu'il est possible d'appliquer un signal d'une tension de 5,7 Vpp.

En effet, l'entrée est assez « difficile » à piloter et aura certainement besoin de signaux d'amplitude importante, tels que ceux produit par un petit amplificateur BF alimentant des haut-parleurs de quelques watts.

UNE CORNEMUSE ÉLECTRONIQUE

de Davide Scullino

Nous vous proposons de fabriquer un petit instrument de musique qui ravira les enfants et leur permettra de jouer 8 notes tout en ajustant leur fréquence à l'aide d'un potentiomètre.

Nous allons réaliser, dans cet article, un petit instrument de musique sans utiliser des composants complexes basés sur des synthétiseurs et des microcontrôleurs pour produire des notes. Le montage ne nécessite qu'une poignée de composants classiques.

Il s'agit d'un simple instrument de musique que nous pouvons considérer comme un remake électronique d'une cornemuse, un instrument très ancien et populaire, ancré de plus dans la tradition de Noël, qui permet de modifier la tonalité de quelques notes pendant que vous jouez.

Voici un petit circuit, à la portée de tous, facile à réaliser, de plus, cette réalisation est économique. Ce montage peut devenir le premier instrument de musique d'un enfant ou un gadget ludique pour accompagner les fêtes de fin d'année.

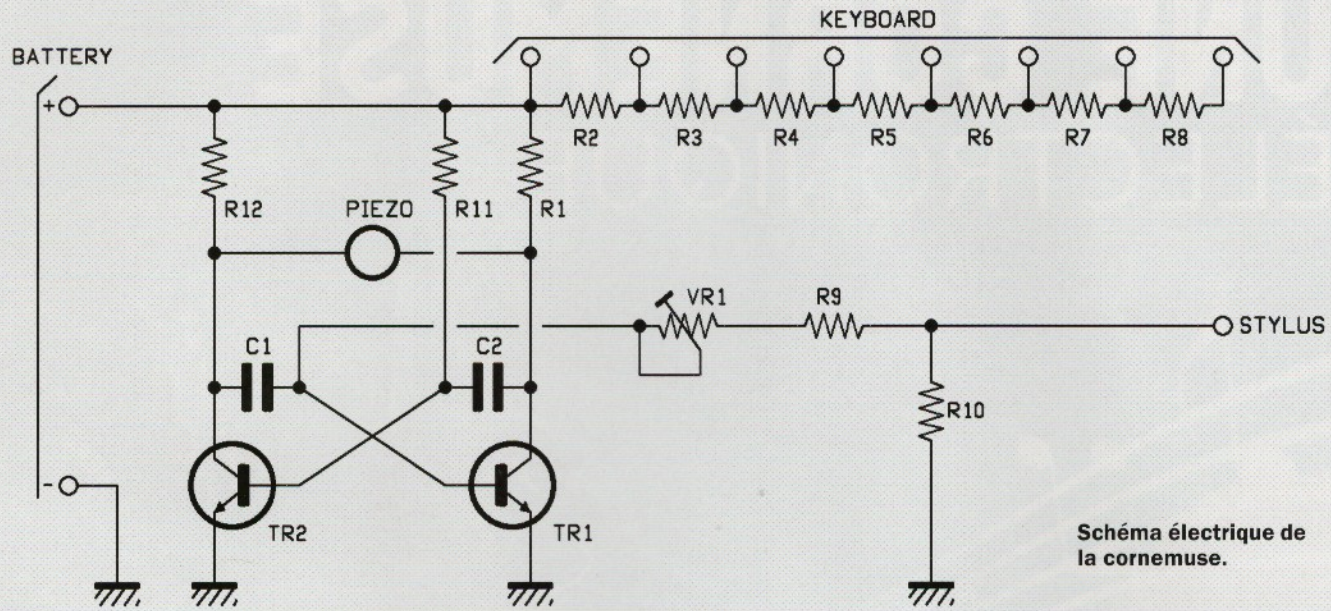


Schéma électrique de la cornemuse.

Schéma électrique

Analysons le schéma électrique de l'instrument de musique. Le cœur du montage est constitué d'un **multivibrateur** composé de transistors BJT de type NPN. Nous avons ajouté un buzzer piézoélectrique (sans électronique, car il sera piloté par le signal variable produit par le circuit astable) et un trimmer afin de pouvoir modifier la fréquence sur un octave.

Grâce à une échelle de résistances et à un fil que nous faisons toucher sur huit pastilles (pads) correspondant aux intersections des résistances de ladite échelle, nous pouvons configurer le multivibrateur de manière astable pour le laisser générer autant de notes, simplement en touchant avec un fil connecté au point « STYLUS » les pastilles qui, dans le schéma électrique, sont regroupées sous la légende « KEYBOARD ».

Celui qui le souhaite peut rendre le petit instrument plus pratique en créant un véritable clavier, qui pourrait simplement être composé de huit boutons normalement ouverts, reliés entre eux par une extrémité et avec les contacts libres connectés chacun à l'un des points de « KEYBOARD ».

Le circuit est un **générateur de signal rectangulaire**, en effet il ne peut pas maintenir un état stable et ses transistors commutent en permanence et de manière alternative. Les transistors **TR1** et **TR2** présentent toujours des **niveaux opposés**. Ce phénomène commence lorsque le circuit est alimenté.

À l'allumage, les deux condensateurs sont déchargés, le transistor ayant une tension de seuil inférieure entre en premier en conduction et bloque l'autre transistor, du moins tant que le condensateur qui est connecté à la base de ce dernier ne se charge pas.

Pour comprendre le fonctionnement du circuit, étudions d'abord le **multivibrateur astable** qui en est le cœur, en partant du principe qu'il peut fonctionner, bien qu'il soit théoriquement composé de deux étages à transistors symétriques BJT de même modèle, que si les transistors ont des caractéristiques légèrement différentes (ce qui est le cas dans la pratique).

Dès que le circuit reçoit l'alimentation, l'**un des deux transistors entre en conduction en bloquant l'autre**. En fait, dans un montage astable, les conditions des deux transistors qui le composent sont étroitement liées et l'un influence l'autre.

Supposons que le point « STYLUS » soit relié à l'une des résistances R2, R3, R4, R5, R6, R7 ou R8 et qu'à l'instant où nous appliquons une tension au circuit via le connecteur « BATTERY » (points + et -), le transistor TR1 entre en conduction en premier.

De ce fait, le collecteur de ce dernier passe à environ 0 V (mettant à la masse l'armature du condensateur C2 qui est connectée à son collecteur) et ne permet pas la polarisation de la base du transistor TR2, car le courant du collecteur de TR1 provoque une chute de tension à travers la résistance R11 de manière à maintenir la tension V_{be} en dessous de la valeur de seuil.

Caractéristiques techniques

- Tension d'alimentation : 9 VDC ;
- Consommation de courant : 40 mA ;
- Fréquence minimale reproduite : 580 Hz ;
- Fréquence maximale reproduite : 2 400 Hz ;
- Nombre de notes exécutables : 8 ;
- Extension : environ sur 2 octaves ;
- Possibilité d'utiliser le « Pitch Bend » (modification de la hauteur du son par potentiomètre).

En effet, **C2 est initialement déchargé** et fait passer le courant.

Dans cette condition, le transistor TR2 reste bloqué et, par l'intermédiaire de la résistance R11, le condensateur C2 commence à se charger, jusqu'à ce que la base du transistor TR2 soit à un potentiel tel que ce dernier devienne conducteur jusqu'à ce qu'il soit saturé.

Le collecteur du transistor TR2 se trouve alors à un niveau de tension proche de 0 V ce qui décharge le condensateur C1, imposant sur la base du transistor TR1 une tension de niveau bas qui provoque le blocage de ce dernier. Ainsi, le collecteur de TR1 passe à un niveau haut car il est bloqué.

Maintenant, le condensateur C1 commence à se charger et porte la base de TR1 à un potentiel tel qu'il provoque de nouveau la conduction de ce dernier.

Le collecteur de TR1 repasse alors à 0 V et décharge immédiatement C2, ce qui implique une tension proche de 0 V sur la base TR2 qui se bloque.

Le cycle décrit se répète tant que le circuit reste alimenté et aboutit à la production de deux ondes rectangulaires de phases opposées sur les transistors TR1 et TR2.

Par conséquent, en connectant un buzzer piézoélectrique entre les deux collecteurs, on obtient un signal de forme d'onde rectangulaire et d'amplitude théoriquement égale à deux fois celle des impulsions présentes entre les collecteurs et les émetteurs des transistors.

Le montage génère une amplitude de valeur suffisante pour piloter le circuit interne du buzzer piézoélectrique et lui permet d'émettre une vibration acoustique suffisamment intense pour être entendue à un bon niveau sonore dans le milieu ambiant.

Le son émis est suffisant pour répondre aux besoins d'une personne jouant d'un instrument.

La durée des cycles de charge et de décharge des condensateurs du circuit astable détermine la fréquence de la

Plan de montage

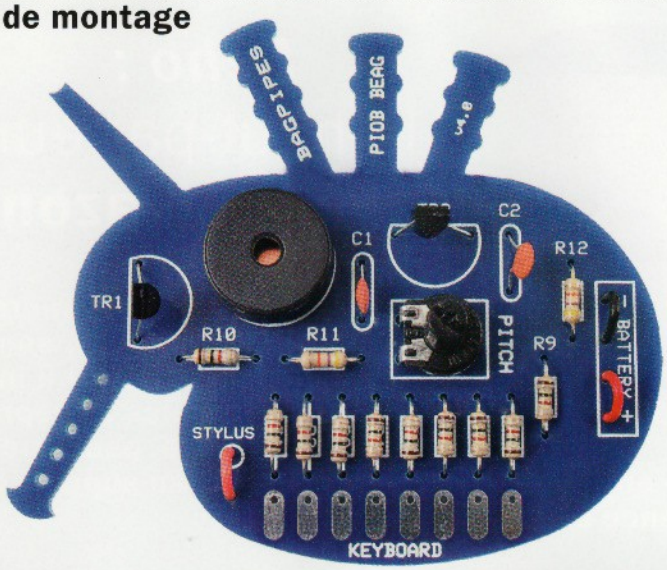


Photo du circuit de notre cornemuse.

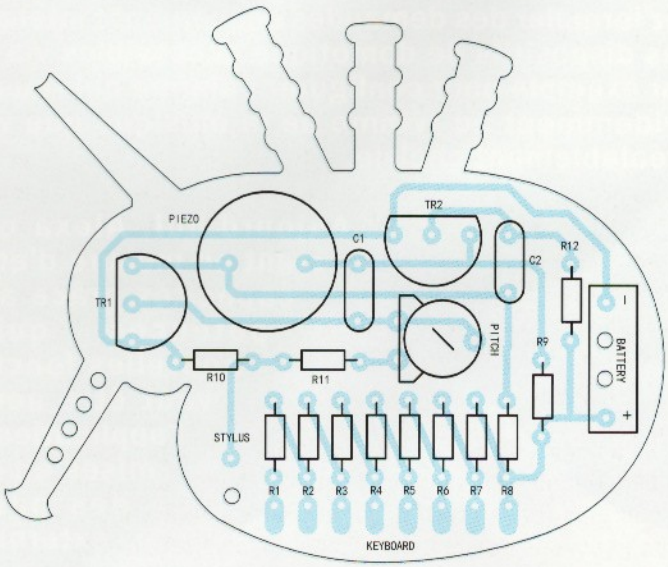


Schéma de câblage des composants du circuit de notre cornemuse.

Liste des composants

R1..... 1 kΩ
R2..... 1 kΩ
R3..... 1 kΩ
R4..... 1 kΩ
R5..... 1 kΩ
R6..... 1 kΩ
R7..... 1 kΩ
R8..... 1 kΩ
R9..... 1 kΩ
R10.... 1 kΩ
R11.... 47 kΩ
R12.... 47 kΩ

C1..... 4,7 nF céramique
C2..... 4,7 nF céramique

TR1.... BC547
TR2.... BC547

VR1.... trimmer 47 K mono tour
PIEZO. buzzer sans électronique

Divers

Connecteur pour pile 6F22 9 V
Morceau de fil de 10 cm de longueur

Echo : le haut-parleur 2.0 d'**Amazon**

Amazon a récemment commercialisé la dernière version d'Echo, son haut-parleur intelligent connecté à Internet via le Wi-Fi, qui, en plus de reproduire une excellente musique, peut être transformé en assistant personnel grâce au support de l'intelligence artificielle d'Alexa.

Grâce à un microphone intégré, vous pouvez demander à l'assistant virtuel Amazon Alexa vos envies, qui peuvent consister par exemple à rechercher un morceau de musique sur le web et à l'écouter. Ou encore, vous pouvez formuler des demandes concernant l'interaction avec des dispositifs de domotique accessibles via le réseau, par exemple pour configurer l'alarme qui vous réveillera évidemment au rythme de la musique que vous aurez préalablement choisie.

Aujourd'hui, Alexa est également en mesure d'étendre ses capacités grâce aux « compétences », qui sont de nouvelles fonctions pouvant être activées via l'application dédiée et mises en œuvre par des partenaires externes.

note reproduite, en ce sens que la fréquence de cette dernière est égale à :

$$f = 1/ (t1 + t2)$$

où **t1** représente la **durée d'un niveau haut sur le collecteur** du transistor TR1 et donc la durée de la charge du condensateur C1, tandis que **t2 désigne la durée d'un niveau haut sur le collecteur de l'autre transistor** (TR2) et donc de la durée de la charge de C2.

Les cycles dépendent des valeurs des résistances et des condensateurs impliqués, soit la résistance R11 pour C2 (t2) et la somme de R9, VR1 et celle entre R2 à R8 reliée par le fil au point « STYLUS », c'est-à-dire la charge de C1 pour la durée t1.

La détermination de la durée t1 correspond au rapport entre la résistance insérée par celle de R2 à R8 et la résistance R10. Cette formule est simplifiée et permet une bonne approximation car sinon nous devrions prendre en compte les courants de base de TR1 et TR2 ce qui compliquerait les calculs.

Cependant, pour les calculs nous nous en tiendrons à la formule ci-dessus, à partir de laquelle nous constatons que la durée t2 est conditionnée par les valeurs des résistances insérées. Il est possible d'obtenir du buzzer autant de fréquences qu'il y a de notes de 1 à 8.

C'est-à-dire autant de notes de musique composant une octave, soit DO, RE, MI, FA, SOL, LA, SI, DO. Il est entendu que les fréquences exactes sont obtenues en accordant l'instrument avec un dispositif similaire à ceux utilisés pour accorder les guitares.

Le diapason classique ne convient pas, car il émet généralement une note LA plus haute que la note DO centrale (par centrale, nous entendons celle de l'octave centrale du clavier du piano, composée de 88 touches réparties en trois touches à gauche, 7 octaves et une autre DO à droite) qui par convention se situe à 440 Hz.

Dans notre cas, comme la fréquence minimale reproductible est de l'ordre de 580 Hz, il faut un dispositif qui génère une fréquence supérieure ou égale.

L'accord peut être effectué avec le trimmer VR1 qui, monté en rhéostat, permet de faire varier la résistance mise en série avec le circuit de charge du condensateur C1 et, plus précisément, de l'augmenter en déplaçant le curseur vers la base du transistor TR1 (correspond à l'augmentation du temps t1 et donc à la diminution de la tonalité des notes) ou de le diminuer si le curseur se déplace dans le sens inverse (la fréquence reproduite augmente car la durée t1 diminue).

Enfin, notez (et vous pouvez le vérifier en jouant de l'instrument) que la note reproduite par notre petit instrument de musique est d'autant plus aiguë que le point « STYLUS » est connecté à proximité de l'alimentation positive, c'est-à-dire que la note devient plus grave lorsqu'elle passe du nœud R1/R2 au nœud R2/R3, jusqu'à l'extrémité libre de la résistance R8. Cette dernière condition correspond à la note la plus basse.

Comme nous parlons de fréquences, le minimum reproductible (en reliant le point « STYLUS » à l'extrémité libre de la

résistance R8 et avec le trimmer VR1 au maximum) est d'environ 580 Hz, comme nous venons de l'expliquer, alors que le maximum est de l'ordre de 2400 Hz. Dans ce dernier cas, le trimmer VR1 est court-circuité et le point « STYLUS » est relié à l'alimentation positive.

En résumé, la note la plus basse se situe aux environs de 580 Hz (note relative au point le plus à gauche du circuit imprimé) et la note la plus haute aux environs de 1 680 Hz (note relative au point le plus à droite) en reliant le point « STYLUS » de l'alimentation positive à l'extrémité libre de la résistance R8 avec VR1 au maximum.

Les notes plus aigües, par contre, vont d'environ 990 Hz (point le plus à gauche) à 2400 Hz (point le plus à droite) avec VR1 court-circuité.

Dans les positions intermédiaires du trimmer VR1, nous obtenons des fréquences intermédiaires, ce qui nous permet, si nous le souhaitons, de glisser la note (fonction Pitch Bend) pendant que nous la jouons, un peu comme nous le ferions avec la distorsion d'une guitare.

Réalisation pratique

Après avoir expliqué le fonctionnement du circuit, passons à la construction de cet agréable instrument de musique, accessible à tous, à la fois parce qu'il utilise exclusivement des composants électroniques traditionnels et qu'il est facile à souder (sans la complexité et la nécessité d'un équipement particulier des composants CMS).

Pour le circuit imprimé, nous avons opté pour une forme de cornemuse, étant donné la simplicité du PCB qui ne contient qu'une seule face.

Vous pouvez facilement le graver par la méthode classique chimique au perchlorure de fer. Nous mettons en téléchargement dans le sommaire détaillé de la revue le typon du circuit imprimé à l'échelle 1.

Notez la forme particulière du PCB, qui rappelle celle de la cornemuse avec ses multiples porte-vent (ou sutel) sortant du réservoir constitué de peau de chèvre.

Toutefois, ceux qui le souhaitent peuvent facilement conserver les pistes du circuit imprimé et les intégrer dans une plaque de cuivre rectangulaire.

Une fois le circuit imprimé gravé et percé, commencez par souder, comme d'habitude, les composants ayant un profil bas, c'est-à-dire les résistances et les condensateurs.

Ensuite, continuez avec les transistors en les orientant correctement comme indiqué dans le plan de montage présenté dans ces pages.

Le buzzer piézoélectrique est du type sans électronique de commande, car il lui sera appliqué le signal variable provenant du circuit. Comme le buzzer ne contient pas d'électronique, il peut être monté sans tenir compte de la polarité.

Vous pouvez remplacer le trimmer par un potentiomètre de même valeur afin d'avoir plus de facilité pour jouer les notes.

Pour compléter le montage, soudez aux emplacements prévus (point + et – du connecteur BATTERY) les deux fils (respectivement rouge et noir) du porte pile pour piles 9 V ou 6F22.

L'ensemble du circuit doit être alimenté par une pile de 9 V (ou une alimentation fournissant cette tension stabilisée, ainsi qu'un courant continu d'au moins 60 mA) et inséré dans un boîtier en plastique en laissant accessible la rangée des pastilles correspondant aux notes.

Soudez un morceau de fil de cuivre souple au points « STYLUS » puis dénudez et étamez l'autre côté du fil afin de le faire entrer en contact avec les pastilles pour jouer les notes.

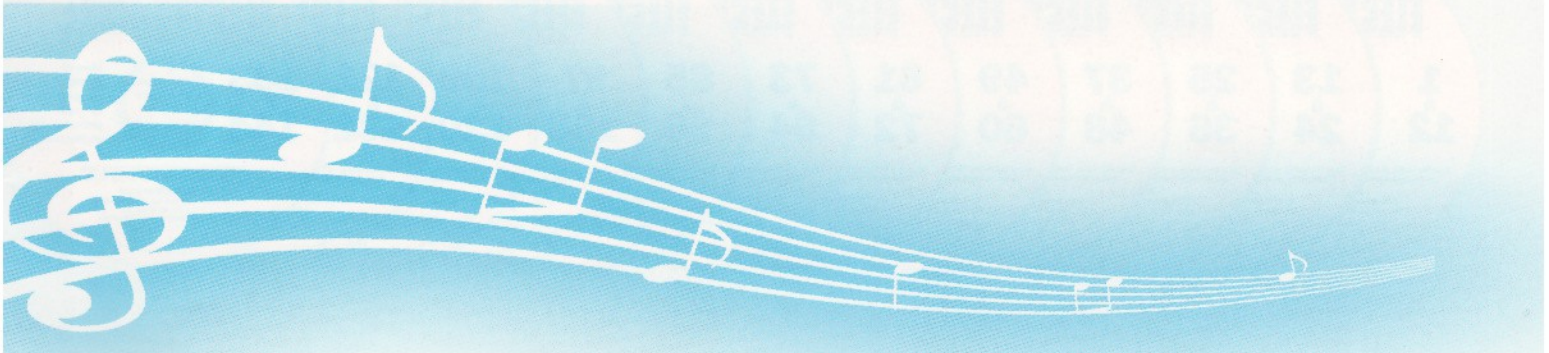
Si vous le souhaitez, vous pouvez fabriquer un clavier composé de huit boutons normalement ouverts, reliés tous ensemble d'un côté au point « STYLUS » du PCB et de l'autre aux pastilles correspondantes.

Les boutons peuvent éventuellement être rectangulaires, même mieux si vous trouvez un petit clavier à recycler d'un vieil orgue électrique ou d'un jouet musical.

Les boutons doivent être placés sur la partie supérieure du boîtier, de même que le potentiomètre. Vous pouvez ajouter un interrupteur marche/arrêt, que vous connecterez en série avec le fil rouge (positif) du support de pile pour économiser cette dernière.

Vous voulez voir notre cornemuse électronique en action ? Rien de plus simple, rendez-vous sur la page : https://youtu.be/rHyYBjY9lvw

Le kit complet est disponible sur le site futurashop.it sous la référence MLP104IT. ■



13 ,58 €* la revue **frais de port inclus pour la France Métropolitaine**

* Frais d'expédition (CEE, les DOM-TOM et autres pays), contactez-nous pour un devis ou bien à calculer directement sur notre site

Au sommaire : Construisons une tondeuse autonome à l'aide de composants de récupérations - Amplificateur 4 W pour guitare qui dispose d'un réglage de volume, de tonalité et un système de distorsion. Notification push avec RaspberryPi - Whiff, le purificateur/ventilateur d'air pour la maison, en plus de l'absence de pale, il faut comprendre que ce n'est pas un ventilateur commun. Transformez votre oscilloscope en analyseur de spectre-2, nous allons aborder la construction et l'utilisation - Imprimante 3D VERTEX - 5. Variateur de vitesse pour moteur à courant continu - Cours Arduino 9 ème partie - Cours de programmation sur iPhone 5ème partie, comment utiliser la « Navigation Bar »

Au sommaire : Sustentation magnétique : Suspendre une ampoule sans fil pour la maintenir, l'allumer sans aucune pile ou câble - Compteur d'énergie électrique 220 VAC, doté d'un afficheur LCD qui visualise la consommation instantanée et globale Générateur de bruit blanc et rose - Vision numérique avec RaspberryPi et Caméra Pi - Chargeur pour smartphone intelligent, idéal pour faire fonctionner et recharger des smartphones ou d'autres dispositifs mobiles - Imprimante 3D VERTEX, le logiciel Repetier-Host - Cours Arduino X - CHIPKIT PI : comment rendre le RaspberryPi plus polyvalent - Veilleuse à LED universelle Cours de programmation sur iPhone – VI

Au sommaire : Reconnaissance faciale avec RaspberryPI, nous allons utiliser les fonctionnalités de la librairie SimpleCV afin de reconnaître les visages et les autres détails du corps humain dans des images - Programmateur hebdomadaire, un temporisateur hebdomadaire avec une sortie relais, un clavier et un afficheur LCD pour les réglages des différentes fonctions - Radio FM & lecteur MP3 USB - Enregistreur vocal avec PIC & EEPROM - 1. Apprenez à maîtriser KiCad EDA-1 - Effet trémolo pour guitare, faire vibrer le son des cordes à travers un système optique original de modulation d'amplitude. Cours MPLAB X IDE-1. Étoile de Noël - Boule de Noël à changement de couleur

Au sommaire : Un amplificateur Hi-Fi 200 W à MOSFET, il pourra sonoriser correctement des pièces volumineuses sans la moindre distorsion - Chambre d'écho digitale - RandA : connectons RaspberryPi avec Arduino avec une carte de développement qui permet de créer une passerelle entre le monde du RaspberryPi et le monde d'Arduino - Enregistreur vocal avec PIC & EEPROM - 2 Cours MPLAB X IDE - 2 pour découvrir MPLABX IDE, le nouvel environnement de développement intégré de Microchip - Apprenez à maîtriser KiCad EDA - 2 - Comment ponter un amplificateur - Contrôleur pour moteur DC - Télécommande infrarouge 4 canaux.

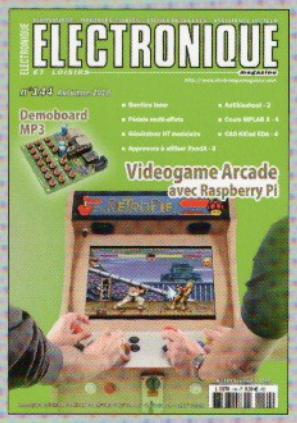
Au sommaire : Apprenons à utiliser RandA, programmation et l'utilisation de la carte - STOP au BLACKOUT ! , système de mesure et de limitation automatique de la consommation électrique jusqu'à 6 kW sous 220 VAC. - Chauffage par induction 1 kW, La technologie ZVS permet une régulation de la tension à l'aide d'une « commutation douce-Détecteur de flamme - Enregistreur vocal avec PIC & EEPROM - 3. Cours MPLAB X IDE – 3, le nouvel environnement de développement intégré produit et distribué par Microchip Technology.. Récepteur 1 & 2 canaux 433,92 MHz compatible MM53200, UM86409 et UM3759, Chaque récepteur peut être combiné jusqu'à 10 télécommandes

CD-ROM ENTIÈREMENT IMPRIMABLE

***CD - FRAIS DE PORT INCLUS POUR LA FRANCE METROPOLITAINE (DOM-TOM ET AUTRES PAYS : NOUS CONSULTER).**

Articles, Revues et CD téléchargeables au format PDF sur Internet : www.electroniquemagazine.com

LISEZ ET IMPRIMEZ VOTRE REVUE SUR VOTRE ORDINATEUR PC OU MACINTOSH



Au sommaire : Jeu vidéo d'arcade avec RaspberryPi - Demoboard MP3, testons l'énorme potentiel du module lecteur audio DFR0299 - Barrière laser, ce montage sert à détecter la présence - Pédale multi-effets, ce montage avec une pédale multi-effets proposant deux types de distorsions fuzz - STOP au BLACKOUT ! (II), cet électronique permet en cas de dépassement de la puissance maximale, de désactiver un ou plusieurs appareils - Générateur haute tension modulaire - Apprenez à maîtriser KiCad EDA – 4 , commençez par l'analyse de Pcbnew - Apprenons à utiliser RandA – 3 - Cours MPLAB X IDE – 4 - Indicateur d'état de la batterie

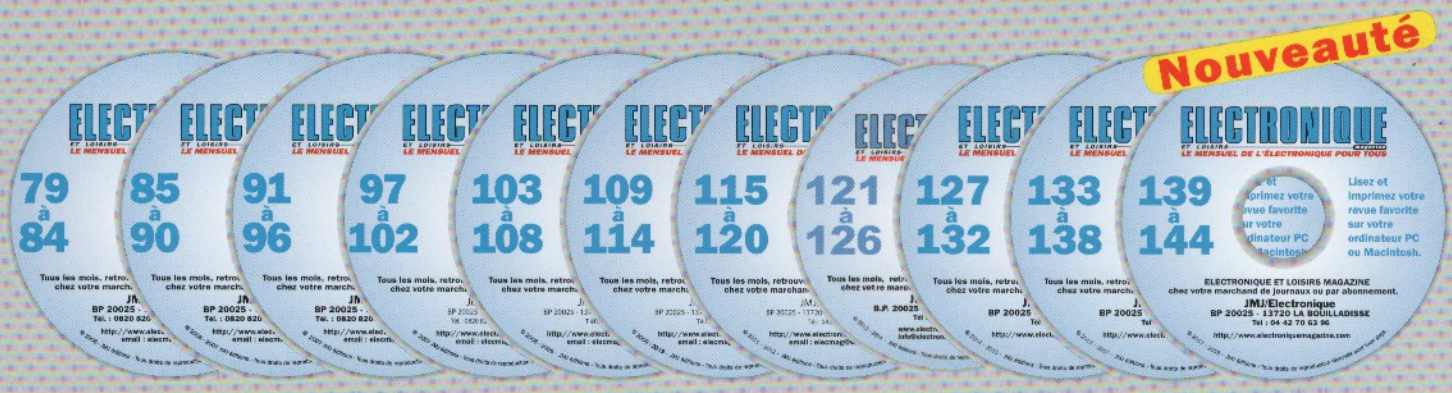
Au sommaire : Unité VoIP sur RaspberryPi - 1, nous vous proposons de réaliser un standard téléphonique doté de toutes les fonctionnalités les plus avancées. La 3DRAG+, une version améliorée capable d'imprimer en 3D des pièces de dimensions 400 mm x 400 mm x 400 mm - Cartes de prototypage NE555 - Cours MPLAB X ID– 5 - Une crèche avec Arduino - Maîtriser KiCad EDA–5 . Amplificateur BF 2 x 15 W à TDA7297 - Détecteur de pluie - Thermostat à microcontrôleur, testez ses fonctionnalités Apprenons à utiliser RandA-4 qui sert de passerelle entre les deux mondes d'Arduino et du RaspberryPi–4 - Un sapin de Noël électronique.

Au sommaire : Découvrir et concevoir un chargeur sans fil, nous vous proposons de concevoir un système d'alimentation sans fil pour vos projets. - Roulette électronique à LED, nous mettons au goût du jour une version électronique de l'un des jeux de casino les plus célèbres. - La 3DRAG+ : imprimez en grand ! - 2, nous décrirons la configuration logicielle de l'imprimante. Décodeur universel pour télécommandes radio - Analyseur de semi-conducteurs - Cartes de prototypage pour amplificateur opérationnel - Cours MPLAB X IDE – 6 - Tutoriel OpenSCAD - Apprenez à maîtriser KiCad EDA – 6 - Unité VoIP sur RaspberryPi - 2

Au sommaire : Extrudeuse de filaments ou comment recycler vos objets 3D, fabriquez un filament d'impression FDM qui vous permet de donner une nouvelle vie aux restes de PLA, d'ABS, de PET - Scanner laser 3D, en utilisant un RaspberryPi, fabriquez un scanner pour acquérir des modèles d'objets en trois dimensions - Relais de puissance à MOSFET à commande isolée - Détecteur 6 gaz - Préamplificateur micro à alimentation Phantom, idéal pour tous les types de microphones, y compris les types à condensateur nécessitant une tension de polarisation externe - Lecteur MP3 pour automobile - Tutoriel OpenSCAD - 2 Amplificateur classe D 2,8 W - Cours MPLAB X IDE – 7

Au sommaire : Détecteur de séismes, dans cet article, nous allons mettre en œuvre le circuit « D7S » de Omron - Machine à sous, réalisez une version portable - Electromètre avec bargraphe à LED, nous vous proposons de construire un appareil simple et d'une grande sensibilité afin de mesurer l'électricité statique - Alimentation multi-tensions pour laboratoire, très utile pour tester des prototypes et alimenter des appareils en réparation - Analyseur de la qualité de l'air, capteur capable de détecter la concentration de particules polluantes - Le guide de la CAO assistée par ordinateur - Scanner 3D avec caméra - Mini récepteur FM - Connaître les régulateurs LDO

CD-ROM ENTIÈREMENT IMPRIMABLE



CD 6 Numéros 25,76 €* / CD 12 Numéros 45,76 €*

50% de remise pour nos abonnés.

JMJ EDITIONS - CD 908 - 13720 BELCODÈNE. Règlement par chèque à l'ordre de JMJ ÉDITIONS et/ou règlement par Paypal ou CB sur notre site : www.electroniquemagazine.com - Tél. : 04 42 70 63 96

AFX 9

L'AFX-9 constitue la nouvelle génération d'afficheurs de laboratoire développés par la société COMELEC. Il est destiné aux laboratoires, aux salles propres et aux blocs opératoires. Sa nouvelle façade design est disponible en aluminium anodisé ou en inox répondant aux dernières normes d'ultra-propreté.

Gamme	AFX-9 SENSOR	AFX-9 SENSOR+	AFX-9 PID	AFX-9 PID+	AFX-9 UNIT	AFX-9 UNIT+
Capteurs différentiel de pression embarqué - Gamme de fonctionnement : -156 Pa à + 156 Pa	OUI	OUI	OUI	OUI	OUI	OUI
Capteur de température/hygrométrie déporté* (câble de 50cm) - Gamme de fonctionnement température : -45°C à +130°C - Gamme de fonctionnement hygrométrie : 0 à 100%	-	OUI	-	OUI	-	OUI
*Il est impératif de positionner le capteur en dessous ou sur les côtés de l'AFX-9.						
Capteurs externes (entrée en tension)	3	3	3	3	3	3
Capteurs externes (entrée en courant)	3	3	3	3	3	3
Entrées contact sec	2	2	2	2	2	2
Affichage des mesures en temps réel sur l'écran	1 à 9	1 à 9	1 à 9	1 à 9	1 à 9	1 à 9
Sorties analogiques 0-10V isolées	3	3	3	3	2*	2*
*la sortie analogique 0-10V numéro 3 n'est plus disponible car elle est dédiée pour la température bloc.						
Sortie analogique 0-10V isolée dédiée bloc opératoire	-	-	-	-	1	1
Sorties contact sec paramétrables	3	3	3	3	2	2
Sortie contact sec dédiée bloc opératoire	-	-	-	-	1	1
Alarmes visuelles paramétrables	OUI	OUI	OUI	OUI	OUI	OUI
Alarmes sonores paramétrables	OUI	OUI	OUI	OUI	OUI	OUI
Régulation multi-capteurs de type PID	-	-	0 à 3	0 à 3	0 à 2	0 à 2
Lisibilité à distance (jusqu'à 8m)	OUI	OUI	OUI	OUI	OUI	OUI
Interface de communication ModBus	OUI	OUI	OUI	OUI	OUI	OUI
Face avant étanche	OUI	OUI	OUI	OUI	OUI	OUI

Caractéristiques techniques sur : *www.ecro.fr*