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

ELECTRONIQUE ET LOISIRS

ELECTRONIQUE

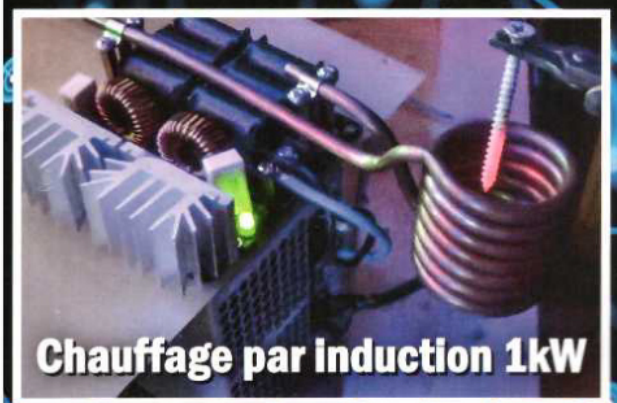
magazine

www.electroniquemagazine.com

n° 143 ÉTÉ 2018

STOP au blackout

- Détecteur de flamme Infrarouge
- RandA - 2 : connectons RaspberryPi et Arduino
- Enregistreur vocal avec PIC et EEPROM - 3
- Cours MPLAB X - 3
- CAO KiCad EDA - 3



N° 143 Juin 2018

L 13270 - 143 - 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 143
Été 2018
100 pages

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

L'EDITO ÉTÉ 2018

Chères lectrices et lecteurs, c'est bientôt l'été et il faut déjà préparer votre départ en vacances. Pour éviter tout désagrément lors du retour, nous vous proposons un système anti-blackout de mesure de la consommation électrique de votre maison ou appartement et de limitation automatique de la consommation en désactivant les appareils qui ne sont pas indispensables. Ainsi, vous ne risquerez pas de retrouver le congélateur arrêté ou la batterie de l'alarme à plat.

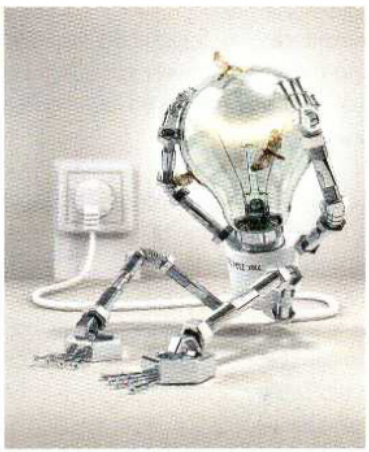
D'autre part, ceux d'entre vous qui disposez d'un camping-car pourront cuisiner quelques petits plats grâce au chauffage par induction qui peut être facilement adapté en plaque de cuisson.

Toujours pour les adaptes du RaspberryPi et d'Arduino, nous continuons la description de la suite de la passerelle « RandA » qui permet de tirer le meilleur parti de la puissance matérielle et informatique du RaspberryPi avec les nombreuses cartes et applications disponibles sur Arduino.

Nous continuons nos cours très détaillés, avec encore une fois près de 30 pages (28 exactement), concernant la programmation en C des PIC32 et la réalisation de circuits imprimés à l'aide du logiciel Open Source KiCad.

Enfin, pour ceux d'entre vous qui possédez des télécommandes au fond d'un tiroir, vous pourrez fabriquer un récepteur universel à 433,92 MHz compatible avec les principaux codages MM53200, UM86409 et UM3759, afin de les utiliser de nouveau.

La Rédaction

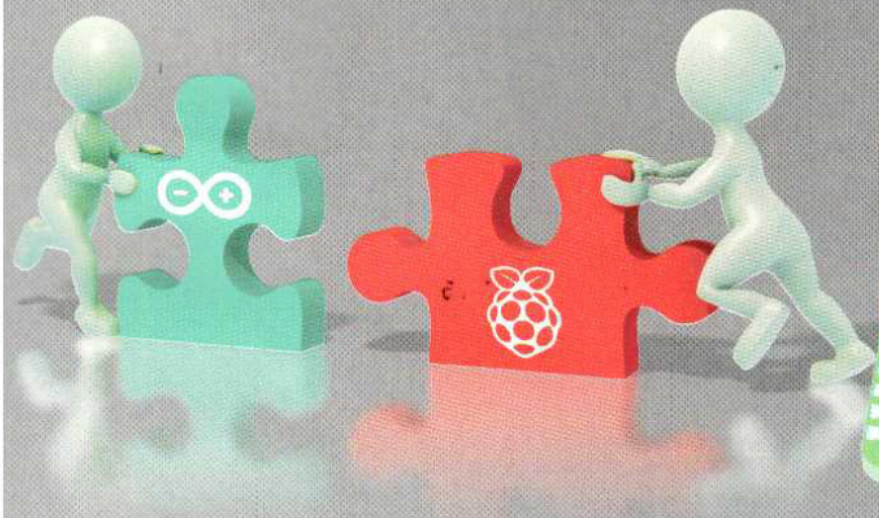


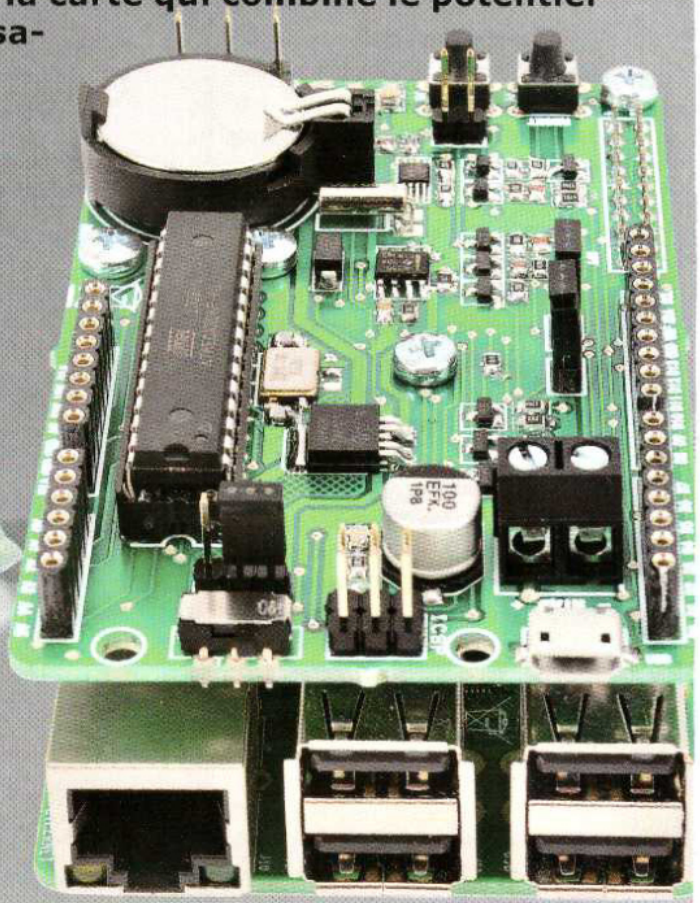
à l'adresse **www.electroniquemagazine.com** dans le sommaire de la revue 143 et à l'onglet « Télécharger ».

Apprenons à utiliser RANDA

Deuxième partie

Dans le numéro 142 d'Electronique et Loisirs Magazine, nous vous avons présenté la carte « RandA » qui permet d'interfacer le Raspberry Pi et Arduino. Le moment est venu d'utiliser la carte qui combine le potentiel du Raspberry Pi avec la facilité d'utilisation d'Arduino.





de Daniele DENARO

Les univers du RaspberryPi et d'Arduino sont proches, mais pas compatibles. Nous avons donc pensé à développer une carte hybride à fort potentiel qui permet de réunir les deux univers. Nous avons commencé à la décrire dans le numéro 142, en présentant la carte « RandA » (ce nom signifie Raspberry and Arduino).

Il s'agit d'une carte de développement équipée d'un « noyau » Arduino. Elle dispose donc des connecteurs classiques pour supporter des cartes Arduino, mais aussi un connecteur compatible avec celui du RaspberryPi.

Dans le numéro 142 d'Electronique et Loisirs Magazine, nous avons abordé les principales caractéristiques de la carte « RandA », maintenant nous allons examiner certains aspects avec des exemples concrets d'utilisation.

Nous considérons que l'installation du logiciel de « RandA », qui comprend l'IDE Arduino modifié à utiliser sur votre ordinateur et tous les logiciels fournis pour le RaspberryPi oa été correctement effectuée.

Pour l'installation, reportez-vous à l'article précédent. Rappelez-vous que vous pouvez acheter une carte SD pour RaspberryPi contenant la distribution déjà installée.

En figure 1, nous résumons le schéma synoptique de « RandA » ainsi que le système de commutation. Comme nous l'avons évoqué, la programmation et l'utilisation de la carte « RandA » offrent deux types d'approches :

- une approche dont **Arduino est le « noyau »**, c'est-à-dire qu'il gère l'ensemble du système ;

- une approche dont **RaspberryPi est le « noyau »**, c'est-à-dire que l'environnement de référence est le système d'exploitation Linux.

Approche dont Arduino est le « noyau »

Commençons par l'approche où Arduino est le centre du système et commençons par développer un sketch sur notre ordinateur en utilisant l'IDE Arduino modifié. L'IDE est parfaitement compatible avec l'utilisation traditionnelle d'une carte Arduino connectée à un port USB de l'ordinateur, donc pour ceux qui savent déjà utiliser cet environnement de programmation, il n'y a rien à ajouter.

La surprise vient alors de la sélection des ports qui sont disponibles pour réaliser une connexion avec Arduino. Parmi tous les ports locaux (port série et port COM virtuel sur USB), un nouveau port apparaît qui est défini par une adresse réseau.

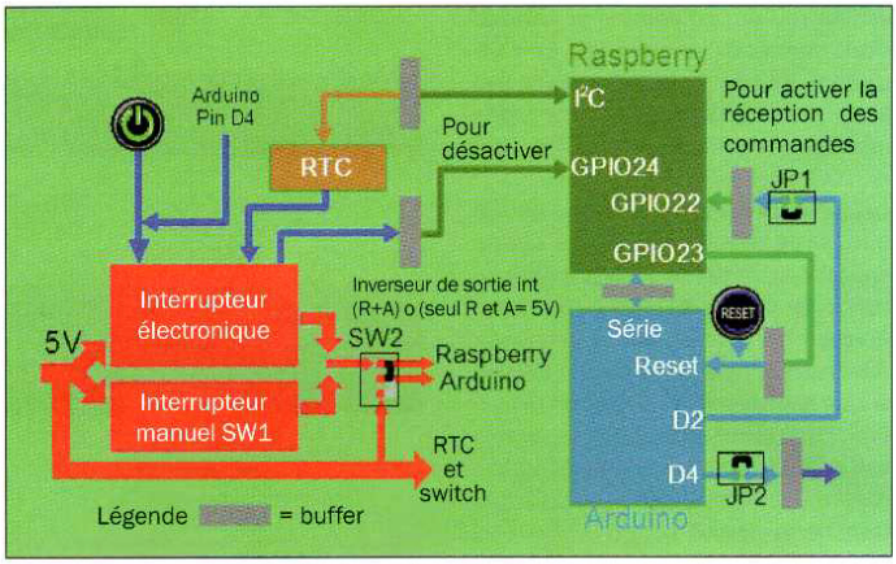


Figure 1 : la partie matérielle (hardware) de « RandA ».

nous pouvons utiliser l'IDE modifié pour détecter l'adresse attribuée par le routeur.

Ensuite, il suffit de se **connecter** avec un **client SSH** comme « **MobaXterm** » ou via un **navigateur** au **serveur WEB** installé sur la carte « RandA ».

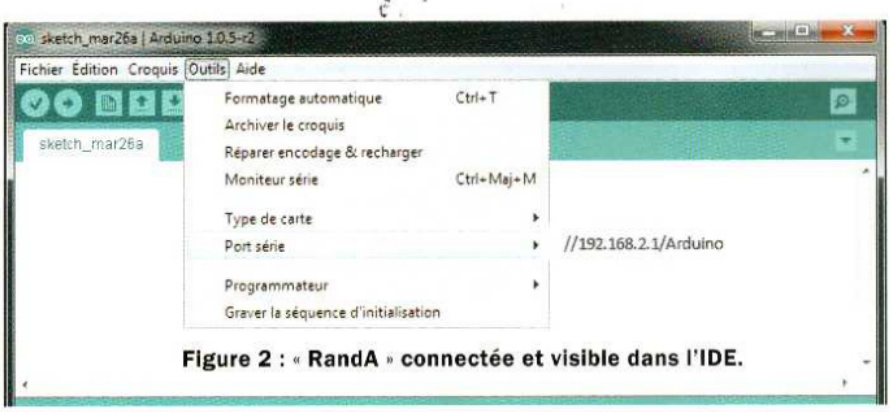


Figure 2 : « RandA » connectée et visible dans l'IDE.

Pour effectuer un premier test, nous pouvons utiliser notre propre sketch écrit pour une carte Arduino standard, ou l'un des exemples présents dans l'IDE modifié. Une fois le port distant sélectionné et le téléversement effectué, le sketch est opérationnel.

Par exemple, si nous choisissons le sketch « Blink » (clignotement), nous pourrions voir la LED sur la carte « RandA » clignoter. Rappelez-vous, en effet, que la carte « RandA » est complètement compatible avec la carte Arduino Uno.

Mais si nous voulons charger un sketch qui utilise le port série, que se passe-t-il ? Ne vous inquiétez pas, en activant la console IDE, comme d'habitude, nous verrons les données

Par exemple « //192.168.10/Arduino ». Ceci, bien sûr, si la carte « RandA » est allumée et connectée au réseau, via le port Ethernet disponible sur le RaspberryPi, ou via une clé WiFi également connectée au RaspberryPi.

En fait, nous pouvons avoir différents ports dans la liste dans le cas où il y a plusieurs cartes « RandA » connectés sur le réseau, donc il est possible de charger plusieurs sketchs dans les différentes cartes « RandA ».

De plus, si nous n'avons pas assigné de manière permanente une adresse du réseau local à « RandA » sur le routeur,

```
curl -i http://api.openweathermap.org/data/2.5/weather?q=Roma,it

HTTP/1.1 200 OK
Server: nginx
Date: Sun, 02 Nov 2014 14:56:22 GMT
Content-Type: application/json; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Source: redis
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: GET, POST

{"coord":{"lon":12.48,"lat":41.89},"sys":{"type":1,"id":5851,"
message":0.1529,"country":"IT","sunrise":1414907053,"sunset":1
414944182},"weather":[{"id":800,"main":"Clear","description":"
Sky is Clear","icon":"01d"}],"base":"cmc stations","main":{"te
mp":292.84,"pressure":1021,"humidity":63,"temp_min":292.15,"te
mp_max":293.15},"wind":{"speed":3.6,"deg":280},"clouds":{"all"
:0},"dt":1414938000,"id":3169070,"name":"Roma","cod":200}
```

Figure 3 : commande « curl » en cours d'exécution.

Commande Linux et réponse
NB : la réponse a été tronquée à un maximum de 64 caractères, à partir du code du sketch, mais en réalité les données au format JSON ne comportent pas de retour chariot.

arriver (ou partir) comme si Arduino était connecté localement. En fait, dans « RandA », le port série est entièrement géré, donc nous pouvons interagir complètement avec Arduino.

Jusqu'à présent, nous avons vérifié que, sans devoir utiliser des cartes spécifiques Arduino, il est possible de connecter celui-ci au réseau car il s'appuie sur les fonctionnalités réseau du RaspberryPi. Ainsi, vous pouvez téléverser un sketch contrairement à l'utilisation d'une carte réseau. Bien que cela soit déjà une fonction intéressante, ce n'est qu'une petite partie du potentiel de la carte « RandA ».

Sur la carte « RandA », il est possible de connecter l'une des nombreuses cartes d'extension disponibles pour Arduino, afin de gérer ou d'étendre le nombre d'entrées/sortie pour des systèmes à relais, pour des systèmes opto-isolés, ou encore pour piloter des moteurs, etc.

Mais vous pouvez faire beaucoup plus et dépasser les limites matérielles imposées par Arduino. En fait, **il est possible d'utiliser le RaspberryPi pour faire effectuer des tâches à Arduino**. Pour cela, il est nécessaire qu'Arduino puisse lancer des commandes sur le RaspberryPi et éventuellement lire la réponse de ce dernier.

Dans ce but, une librairie a été intégrée dans l'IDE Arduino modifié. Cette librairie (**RAComm**) permet de lancer des commandes sur le RaspberryPi et de recevoir (comprendre) la réponse. Cela permet également d'ouvrir un fichier, de le lire ou d'écrire à l'intérieur.

Comme Arduino communique avec le RaspberryPi via le port série, l'IDE ne peut pas dialoguer avec Arduino (le port série n'est pas disponible). Par conséquent, il n'est pas possible d'utiliser la console IDE.

C'est pour cette raison que nous avons ajouté la possibilité d'ouvrir une console sur le RaspberryPi afin d'écrire des messages, de lire l'état d'une ou plusieurs entrées ou simplement de déboguer. Cependant, cette console, construite en mode terminal, nécessite une connexion avec le RaspberryPi pour être visible, soit via un réseau avec le protocole SSH ou directement depuis le terminal.

Figure 4 : synoptique du sketch.

Examinons maintenant un exemple pratique de cette synergie entre Arduino et RaspberryPi. Nous utilisons une commande Linux utile qui permet d'**effectuer une requête** (**GET**) d'une ressource sur le réseau à l'aide de la commande « **curl** ».

Essayons d'utiliser cette commande pour demander au site « openweathermap.org » des informations atmosphériques relatives à une localité.

En suivant les indications du site, nous savons que la requête doit être effectuée au format représenté en figure 3, où la réponse apparaît également.

Dans l'exemple, nous avons utilisé Rome (toutes les routes y mènent), mais la requête peut être faite pour n'importe quelle autre localité du monde. En mode basique, les données sont retournées au format « **JSON** », mais elles peuvent aussi être au format **XML**. Le sketch, fourni à titre d'exemple dans la librairie, s'appelle « **ReadWeb** ». Il est constitué de quelques lignes de code, mais il ne s'agit pas de la partie concernant l'extraction des données restituées par le site. En effet, en plus du code provenant du protocole HTTP, les données fournies sont nombreuses.

Listing 5A : corps du sketch.

Définitions générales

```
#include «RAComm.h»                    // Utilise la bibliothèque des commandes de RaspberryPi

/* Choix de la suppression d'une ville ou de l'insertion d'une nouvelle ville */
//#define CITY «London,uk»
//#define CITY «Paris,fr»
//#define CITY «Berlin,de»
//#define CITY «Madrid,es»
#define CITY «Roma,it»
//#define CITY «Milano,it»
//#define CITY «Napoli,it»
//#define CITY «Catania,it»

// Commande RaspberryPi pour envoyer une requête à une URL
#define CMD1 «curl -i http://api.openweathermap.org/data/2.5/weather?q=» CITY

#define DEB true                       // flag de débogage
#define LENBUFF 64                     // longueur du buffer pour la lecture de la réponse
RAComm Cmd;                            // Cmd est la classe RAComm

char rbuff[LENBUFF];                   // buffer pour la réponse de Raspberry

int flagStatus=0;                      // signalisation des états à l'aide de la LED Arduino
int led;
```

Fonctions setup et loop

```
void setup()
{
 initLed();                            // LED Arduino utilisée comme flag

 createPattArray(5);                   // définit un tableau de modèles
 setPattern(0,»\»temp\»:»,',',10);     // définit chaque modèle
 setPattern(1,»\»temp_min\»:»,',',10);
 setPattern(2,»\»temp_max\»:»,'}',10);
 setPattern(3,»\»humidity\»:»,',',10);
 setPattern(4,»\»pressure\»:»,',',10);

 getWeather();

}

void loop()                            // signalisation de l'état (clignotant : n fois avec la période en millisecondes)
{
 switch (flagStatus)                   // 1 : pas de communication avec Raspberry ; 2: erreur de communication
 {
  case 1 : blinking(50,100);break;
  case 2 : blinking(10,500);break;
 }
 delay(2000);
}
```

Listing 5 B : Fonction getWeather

Fonction principale

```
int getWeather()
{
  boolean ok;
  Cmd.begin();                                          // Initialisation de la commande de communication du
                                                        // RaspberryPi
  delay(500);                                           // délai (pour plus de clarté)
  for (int i=0;i<20;i++)                                // Max 20 contrôles. Si cela n'est pas vérifié, quelque chose
                                                        // ne va pas !
    {ok=Cmd.isReady();if (ok) break;Cmd.begin();delay(100);}
  if (!ok) {noListen();return -1;}                      // si pas de communication  alors aucune action

  Cmd.openFileWrite(«/home/pi/wheather.log»);           // ouvre le fichier pour enregistrer les données
  ok=Cmd.sendCommand(CMD1);                             // envoi une commande
  if (!ok) {error();return -1;}
  while (Cmd.getAnswer(rbuff,LENBUFF)!=NULL)            // lit la réponse par bloc de 64 octets à la fois
  {
    if (DEB){Cmd.writeRec(rbuff);}                      // si DEB écrit toutes les réponses sur le fichier
    checkPattern(pattarray.ptn,rbuff);                  // vérifie tous les modèles et collecte les données
  }
  // Écrit les données dans un fichier : température, température minimale, température maximale, humidité, pression
  char rec[30];                                         // utiliser le buffer pour créer un enregistrement pour le
                                                        // fichier float v=0;
char sval[10];                                          // lit la température en virgule flottante et obtient une nouvelle
chaîne
                                                        // car le compilateur Atmel n'utilise pas le format « %f »
  v=getPattValN(0);v=v-273.15;dtostrf(v,5,1,sval);
  snprintf(rec,30,»Temperature: %s C»,sval);Cmd.writeRec(rec);
  v=getPattValN(1);v=v-273.15;dtostrf(v,5,1,sval);
  snprintf(rec,30,»Temp-min   : %s C»,sval);Cmd.writeRec(rec);
  v=getPattValN(2);v=v-273.15;dtostrf(v,5,1,sval);
  snprintf(rec,30,»Temp-max   : %s C»,sval);Cmd.writeRec(rec);
  snprintf(rec,30,»Humidity   : %s%%»,getPattVal(3));Cmd.writeRec(rec);
  snprintf(rec,30,»Pressure   : %s P»,getPattVal(4));Cmd.writeRec(rec);
  Cmd.close();                                          // termine la communication
}
```

À ce stade, nous devons garder à l'esprit que la mémoire d'Arduino est limitée. En fait, il est impensable de mémoriser toute la réponse dans un buffer. De plus, le **buffer du port série ne peut contenir au maximum que 64 octets**, il n'est pas conseillé de lire plus de données afin d'éviter des pertes de données.

Nous devons donc demander la réponse par petits bouts et effectuer une analyse caractère par caractère afin de détecter le début des données que nous voulons extraire. La figure 4 montre un synoptique de principe du sketch.

Les listings 5a et 5b montrent les parties essentielles du sketch. Comme vous pouvez le voir, la requête est exécutée une seule fois, pour la répéter nous devons réinitialiser Arduino.

En effet, ce sketch est un exemple, alors que pour une utilisation réelle, il faudrait probablement répéter la requête à intervalles réguliers.

Pour ce faire, il suffit d'ajouter une commande supplémentaire qui gère le mode veille, et une autre commande de « **shutdown** » (arrêt).

Par exemple, pour lire des données météorologiques toutes les heures, il suffit d'insérer :

- **RAComm.sendCommand("SetRestartAt -sd 00")**
- **RAComm.sendCommand("sudo shutdown -h now")**

La première détermine la sortie du mode veille à la fin de l'heure, tandis que la seconde provoque l'arrêt du système.

Rappelons que pour « RandA », la commande « **shutdown –h** » provoque l'interruption de l'alimentation à la fin du processus de fermeture.

De plus, le sketch écrit la réponse dans un fichier, créant ainsi une sorte de journal (log). Pour cela, nous utilisons des fonctions de la librairie qui nous permettent d'ouvrir un fichier (un à la fois !), d'écrire à l'intérieur et de le lire.

Les commandes utilisées dans l'exemple sont :

- **RAComm.openFileWrite("home/pi/weather.log")**
- **RAComm.writeRec(rbuff)**

Le fichier est fermé automatiquement par la commande de fermeture de la communication avec le RaspberryPi

(même s'il existe une commande spécifique). Les données extraites de la réponse sont visibles en figure 6.

Pour la température, une conversion de la température absolue est effectuée pour obtenir une température en degrés centigrades (T absolue = - 273,15 K).

Les données sont sauvegardées dans le fichier, mais, pour implémenter une application pratique, vous pouvez par exemple activer ou réguler un système de chauffage ou de refroidissement via les sorties numériques ou analogiques d'Arduino.

Comme nous l'avons déjà mentionné, la partie la plus consistante du sketch est celle liée à un problème qui ne concerne pas l'interaction Arduino/RaspberryPi, mais plutôt le dépassement des limitations de la mémoire RAM d'Arduino.

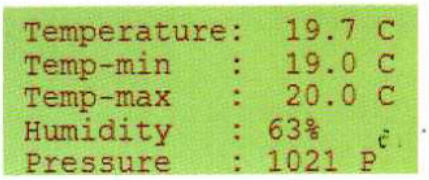


Figure 6 : données météorologiques extraites.

La réponse est divisée en enregistrements de 64 caractères maximum, mais le modèle à détecter (c'est-à-dire le bloc de caractères à identifier), par exemple l'expression « "temp": » peut se trouver sur deux lignes. Même si ce problème ne concerne pas directement cet article, nous donnons ici une brève description de l'algorithme adopté.

La problématique a été généralisée et il est donc possible de l'utiliser pour lire, par exemple, un texte HTML en recherchant des expressions (modèles) particulières dans une logique de : « **clé/valeur** ».

Tout d'abord, nous définissons une structure modèle et un tableau. Chaque élément du tableau contient le modèle à rechercher qui fonctionne comme un démarrage des données à lire et un caractère qui indique la fin de celui-ci.

De plus, nous définissons certaines variables de l'algorithme. L'algorithme fonctionne en comparant tous les modèles définis en parallèle, en marquant individuellement chaque caractère.

Lorsqu'un modèle correspond, il commence à lire les données indépendamment des autres modèles. Les données sont mémorisées sous la forme de chaînes de caractères dans un buffer de la structure correspondante.

Avant de passer à un autre exemple d'utilisation de la librairie « RAComm », il convient de préciser que dans « **RAComm.sendCommand()** », il est possible d'**insérer toutes les commandes standard** des distributions **Linux** (Raspbian), les programmes que nous avons préparés pour « RandA » et tout script ou exécutable que vous avez implémenté.

Tableau 1

Commande	Signification
ArduIO	Définit et écrit ou lit les broches d'Arduino.
Arduload	Charge un sketch compilé (.hex) enregistré sur le Raspberry.
ArduInterrupt	S'arrête si une condition est présente sur les broches d'Arduino.
ResetRandA	Réinitialisation d'Arduino.
GetRTC	Lit l'horloge de RandA.
SetRTC	Règle l'horloge de RandA.
SetRestartAt	Définit l'alarme pour le réallumage.
ResetAlarm	Supprime l'alarme activée, si nécessaire.
SetSysClock	Règle l'horloge en utilisant la RTC.
SendMail	Envoie un e-mail.

Nous avons déjà abordé l'utilisation d'une commande configurée pour « RandA », il s'agit de la commande « **SetRestartAt** » (c'est un exécutable C).

Il est maintenant temps de lister brièvement toutes les autres commandes, que vous pouvez voir dans le tableau 1. Elles peuvent également être visualisées depuis la console du RaspberryPi en tapant la commande « commands ». En ajoutant « **-h** », cela affiche toute l'aide.

En réalité, certaines d'entre elles, comme « **ArduLoad** », « **ArduIO** » et « **ArduInterrupt** », sont conçues pour une utilisation centrée sur le RaspberryPi. En effet, elles fournissent un accès aux fonctionnalités d'Arduino à partir de scripts Linux.

Le deuxième exemple que nous allons étudier est l'envoi d'un e-mail. Supposons que nous voulions détecter la présence d'un signal d'alarme et avertir quelqu'un de l'événement.

La partie liée à la détection de la condition d'alarme est généralement une tâche pour Arduino. Par exemple, nous pourrions détecter le dépassement d'un seuil analogique ou d'une condition ON/OFF. Cependant, l'envoi d'un e-mail avec Arduino est une tâche difficile, notamment à cause des serveurs SMTP qui utilisent maintenant des protocoles protégés, donc l'implémentation avec Arduino est pratiquement impossible.

La commande à utiliser pourrait être, par exemple :

```
RAComm.sendCommand("
SendMail mailto=\"lulu@aol.com\"
subject=\" Détection \"
filemess=\"/home/pi/message.txt\" ")
```

La commande est visible sur plusieurs lignes pour faciliter la lecture. Le modèle « \» » indique la présence de guillemets dans la chaîne (la commande à passer en tant qu'argument de la fonction).

Le texte du message est automatiquement formaté par « SendMail », à partir du fichier dans lequel il a été placé.

La commande autorise la possibilité d'envoyer un texte court au lieu d'un texte placé dans un fichier, de cette manière :

```
RAComm.sendCommand("
SendMail mailto=\" lulu@aol.com\"
subject=\" Détection \"
message=\"Valeur : 320\" ")
```

Listing 7 : le sketch « SendMail » (définitions principales)

Définitions

```
#include «RAComm.h»                          // utilise la bibliothèque de commandes du RaspberryPi

#define analog 1                             // broche analogique utilisée
#define LOGFILE «/home/pi/values.dat»        // nom du fichier log
#define TEXTMAIL «/home/pi/mailval.txt»      // nom du fichier log


#define Q «\»»                               // caractère d'échappement pour les guillemets («)
#define SENDTO «lulu@aol.com»                // adresse mail
#define SUBJECT «Value»

//Commandes :

// Commande RaspberryPi pour obtenir l'horodatage (timestamp)
#define CMD1 «GetRTC -s»

// Commande qui utilise l'instruction « tail » de Linux afin d'extraire les 5 dernières lignes du fichier log et mettant en forme le texte à envoyer
#define CMD2 «tail -n 5 « LOGFILE « > « TEXTMAIL

// Commande RaspberryPi de texte « SendMail ». (NB : le texte entre guillemets doit être inséré en utilisant le caractère d'échappement)
#define CMD3 «SendMail mailto=» SENDTO « subject=» Q SUBJECT Q « filemess=» Q TEXTMAIL Q

// Commande RaspberryPi pour le réglage de l'alarme (détection)
#define CMD4 «SetRestartAt -sd %d %d»        // utiliser dans « sprintf » pour insérer des variables

// Shutdown command
#define CMD5 «sudo shutdown -h now»


RAComm Cmd;                                  // Cmd est la classe RAComm
```

Il est possible également d'ajouter une pièce jointe ou plusieurs destinataires. Pour plus de détails, référez-vous à l'aide de la commande.

À ce stade, cependant, il est nécessaire de spécifier que le fichier « **/home/pi/bin/Mail.properties** » doit être initialisé avec ses propres données, c'est-à-dire avec le nom d'utilisateur et le mot de passe reconnus par le serveur SMTP utilisé et l'adresse du serveur lui-même.

De plus, gardez à l'esprit que ceux qui ne sont pas habitués à utiliser un client de messagerie (comme Outlook ou Thunderbird), pourraient être confrontés, par exemple, au problème du serveur Gmail qui n'est pas habilité à recevoir certains types de commandes. Dans ce cas, vous devez définir les paramètres appropriés sur le site Gmail.

Dans l'exemple, inclus dans la librairie « **SendMail** », une valeur analogique est détectée périodiquement, qui, avec l'horodatage provenant de la RTC, sont mémorisés dans un fichier puis envoyés à une adresse mail.

Notez que, ce n'est pas seulement la dernière donnée lue qui est envoyée, ce sont les 5 derniers enregistrements qui sont envoyés par e-mail. Pour cela, la commande Linux « tail » est utilisée pour extraire les « n » derniers enregistrements d'un fichier.

Dans le listing 7, vous pouvez voir les principales définitions présentes dans le sketch. À travers ces exemples, vous constatez que le potentiel de la librairie de liaison avec le RaspberryPi est évident.

En outre, notez qu'en cas d'erreur logique (par exemple un sketch qui provoque l'arrêt du système au démarrage), l'**ouverture du cavalier JP1 permet de désactiver la liaison** (communication) avec le RaspberryPi. Le cavalier JP1 signale au RaspberryPi l'intention de « dialoguer ».

Le script qui détecte le signal est lancé au démarrage (dans le dossier : file/etc/rc.local) et s'appelle « **StartListenCmd** ». Il peut également être lancé à partir d'un fichier log. Il est aussi possible, en cas de problèmes de débogage, de lancer directement le programme C, nommé « **ExecSCmd** », qui communique, après avoir désactivé « StartListenCmd ».

« ExecSCmd » peut être lancé avec le flag de débogage « -s », il retourne sur la console tout se qui passe sur le port série. Le programme « ExecScmd » est inclus dans l'archive « **RandAinstall.tar.gz** », avec les autres programmes

qui se trouvent dans le dossier de téléchargement « Programmes RandA-V1.3 ». Le chemin est : « **/home/pi/workspace/cworkspace** ».

Il existe encore une autre possibilité d'exploiter l'approche Arduino en tant que noyau. Il suffit de garder Arduino sous tension et d'agir sur le déviateur SW2 qui le sépare de la gestion de l'interrupteur électronique. Arduino, privé de son alimentation et de l'USB, consomme très peu et peut être mis dans une condition de consommation très faible.

Ainsi, dans cette condition, Arduino peut activer le RaspberryPi en générant une impulsion sur sa broche D4 (à condition que le cavalier correspondant JP2 soit fermé).

Un autre exemple, contenu dans la librairie, met en œuvre cette utilisation. Il allume le RaspberryPi si une valeur supérieure à 300 est présente sur son entrée analogique A0.

Approche dont RaspberryPi est le noyau

Ceux d'entre vous qui êtes familiers avec le système d'exploitation Linux, trouveront certainement plus facile d'utiliser le RaspberryPi comme base de programmation. Pour cela, nous pouvons utiliser un puissant environnement de développement en langage C tel que les « **codesblocks** », que nous avons installé sur le RaspberryPi, ou encore l'environnement de développement « **Idle** » pour le langage Python. Comme Java est également installé sur le RaspberryPi, ce langage puissant peut également être utilisé.

Tableau 2

Option	Signification
–set n x	Configuration de la broche n en tant que x = inp, out, ipl (ipl : entrée pull-up).
-rda n	Lit la broche analogique n.
-rdd n	Lit la broche digitale n.
-wra n v	Configuration des broches PWM.
-wrd n v	Configuration des broches digitales (0/1) ; attention à la broche 4 : **retirez le cavalier pour l'utiliser**. La broche 4 est connectée à l'interrupteur ON / OFF.
-pou n v	Impulsion sur la broche n d'une fréquence v (tonalité).
-poi n v	Lit la durée d'impulsion sur la broche n (v = 0 ou 1 ; définit une impulsion haute ou basse).

En fait, il existe déjà des programmes Java utilisés dans « RandA », tels que la commande « SendMail » ou encore les servlets sur le serveur Web que nous aborderons plus tard.

Pour rappel, une servlet est une classe Java qui permet de créer dynamiquement des données au sein d'un serveur HTTP.

Cependant, nous n'avons pas installé d'environnement de développement pour Java, car nous n'avons pas été satisfaits des IDE Open Source pour le RaspberryPi. Bien qu'il soit possible de l'installer à partir de Raspbian, Eclipse ne nous a pas donné entièrement satisfaction en raison de problèmes d'installation et de performances. Les programmes Java complets avec les sources se trouvent dans le répertoire : « **/home/pi/workspace/jworkspace** ».

Les programmes (ou scripts) réalisés peuvent également être utilisés dans l'approche Arduino en tant que noyau, comme nous l'avons déjà vu. Ils peuvent utiliser le port série pour communiquer avec des sketchs réservés à la communication (c'est-à-dire en dehors de la librairie RAComm). De cette façon, Arduino devient un puissant périphérique programmable du RaspberryPi.

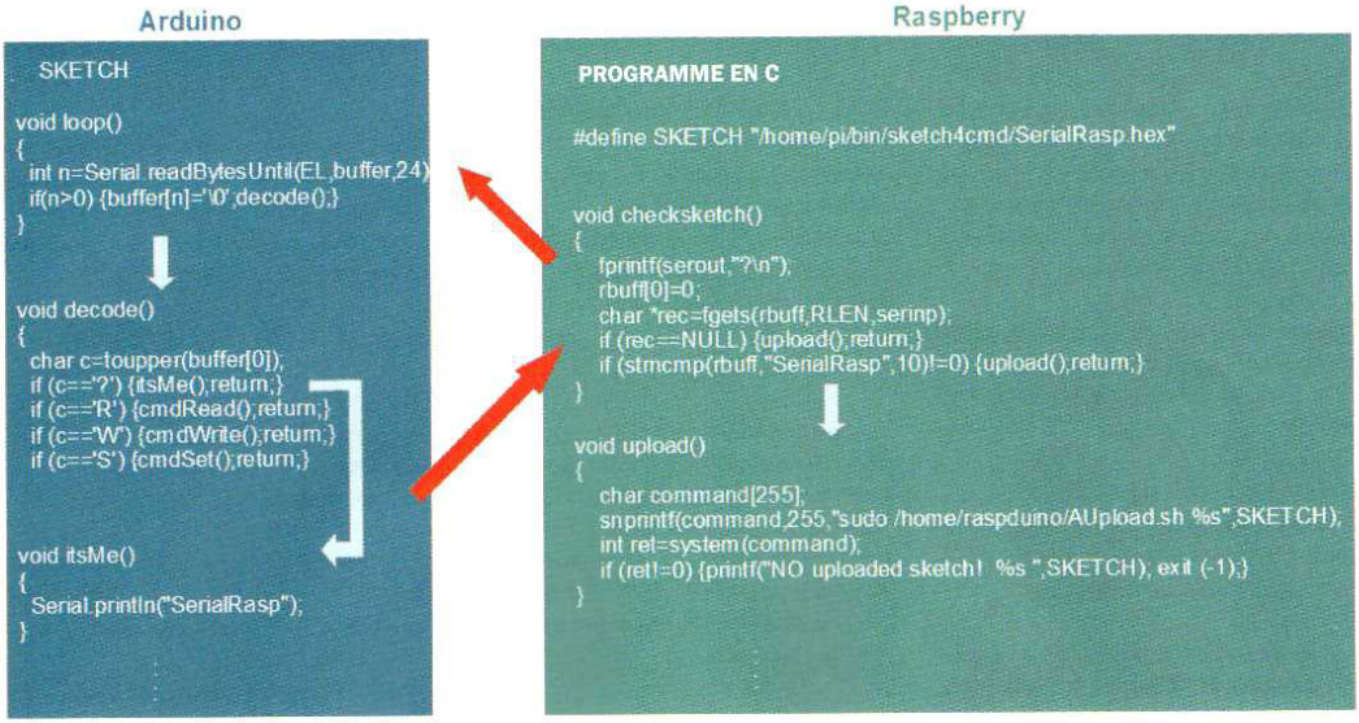


Figure 8 : « SerialRasp » fonctionne avec Arduino.

Listing 9a : script bash pour un allumage conditionné.

```
#!/bin/bash
ArduIO -set 13 out                    # configuration de la broche de la LED
out=»$(ArduIO -rda 1)»                # valeur de A1 dans la variable out
if [[ $out == NO* ]]                  # s'il ne peut pas lire alors sortie
then echo «Err!»; exit 0; fi
out=${out//[^0-9]/}                   # supprime tout caractère qui n'est pas un chiffre
echo $out                             # permet d'obtenir une variable numérique claire
if [ «$out» -lt «200» ]               # si < au seuil
then
 echo « Allumée !»
 ArduIO -wrd 13 1                     #LED allumée (ON)
else
 ArduIO -wrd 13 0                     #LED éteinte (OFF)
 echo « Rien à faire !»
fi
```

Listing 9b : script Python équivalent à celui du script bash (listing 9a)

```
#!/usr/bin/python
import os
import commands
ret=os.system(«ArduIO -set 13 out»)
out=commands.getoutput(«ArduIO -rda 1»)
if out[0:1] == «NO»: quit()
out=int(out)
print out
if out < 200:
   print « Allumée !»
   ret=os.system(«ArduIO -wrd 13 1»)
else:
   print « Rien à faire «
   ret=os.system(«ArduIO -wrd 13 0»)
```

La commande « Arduload », qui charge un sketch compilé au format « .hex » enregistré sur le RaspberryPi, permet de modifier le comportement d'Arduino, car le sketch est installé par l'application sur Arduino.

Ce sketch, au format exécutable, peut avoir été créé par l'IDE distant ou par l'IDE installé localement sur le RaspberryPi.

En fait, les deux versions de l'IDE, en plus de télécharger le sketch, sauvegardent l'exécutable sur le RaspberryPi dans deux répertoires distincts :

- **/home/ArduinoUpload** : pour les sketchs élaborés localement ;
- **/home/RArduinoUploads** : pour les sketchs élaborés à distance.

C'est un ajout supplémentaire à l'IDE que nous avons modifié. Comme exemple d'application qui gère Arduino en tant que périphérique, nous faisons référence à la commande « ArduIO » présente dans le tableau 1. Ce programme en C peut également être trouvé dans le dossier « cworkspace » avec le fichier source.

La commande « **ArduIO** » peut être considérée comme l'**équivalent de l'utilitaire GPIO** (wiringPi). En fait, elle permet de gérer les pins d'Arduino à partir d'une ligne de commande. Par exemple, pour allumer la LED Arduino (broche 13), vous devez taper les commandes suivantes :

- **ArduIO -set 13 out** : définit en sortie la broche 13 ;
- **ArduIO -wrd 13 1** : met à 1 (niveau logique haut) la broche 13 ;
- **ArduIO -wrd 13 0** : éteint la LED.

Vous pouvez utiliser les méthodes habituelles d'Arduino, c'est-à-dire la lecture des broches analogiques et numériques, l'activation/désactivation des sorties numériques et la configuration des broches PWM. En tapant la commande « ArduIO –h », l'aide détaillée est obtenue. Plus précisément, vous pouvez voir dans le tableau 2 les différentes options possibles.

Le programme utilise un sketch (SerialRasp) avec lequel il coopère en lui envoyant les commandes correspondantes. C'est ce dernier qui active les pins.

Donc le sketch doit avoir été chargé au préalable dans Arduino. En réalité, la commande « ArduIO » surmonte ce problème, en chargeant le sketch s'il n'est pas présent dans Arduino.

Pour permettre cela il faut :

- identifier le sketch sur Arduino ;
- trouver le sketch dans une situation où il est nécessaire de le télécharger (sketch endommagé).

Le premier point est facilement résolu en préparant le sketch afin que, s'il est interrogé, il réponde avec son nom.

Pour le second point, un répertoire nommé « /home/pi/bin/sketch4cmd » est mis en place pour le récupérer.

Applications Web et CGI

Au début du web, les pages fournissaient des contenus structurés et multimédias, mais statiques. Cependant, au fil du temps, l'internet a commencé à interagir avec les pages en entrant des données et en attendant des réponses. C'est comme cela que son nés les moteurs de recherche. Pour cela, il était nécessaire que le serveur Web ait la capacité d'activer les programmes qui recevaient les demandes afin de savoir comment générer la réponse au format HTML pour la renvoyer au demandeur.

Les programmes qui exécutaient ces fonctions étaient intégrés dans ce qu'on appelle un CGI (Common Gateway Interface) ou encore « Interface de passerelle commune ».

Grâce à cette interface, le serveur WEB permet d'interpréter de manière différente une requête dédiée à un traitement (demande) d'une requête provenant d'une simple page.

Dans le premier cas, il doit savoir quel programme activer et lui transmettre la demande. Les programmes utilisés pour cette fonctionnalité sont généralement des interpréteurs de script.

Ainsi, l'augmentation de la demande d'interaction, telle que l'utilisation de la technologie AJAX qui interagit en arrière-plan, a conduit à intégrer davantage de capacité de traitement au niveau du serveur Web, d'où la transition vers le serveur d'applications Web.

Il existe deux technologies différentes : « dot net » (aujourd'hui .NET Framework) et « Java », la première a été développée par Microsoft qui en est donc propriétaire, tandis que la seconde est Open Source. Dans le cas de Java, nous parlons d'une intégration constituée de tâches (threads) appelées Servlets ou de leur utilisation plus complexe, similaire aux scripts PHP, représentées par des pages JSP.

Avec Servlet et JSP, l'interaction est beaucoup plus fluide car il n'y a pas de processus entier à activer, ni de script à interpréter (en fait les pages JSP sont automatiquement compilées dans Servlet). Cependant comme l'interaction est légère, la complexité de la programmation de Servlet peut être excessive par rapport à la tâche à accomplir, ce qui explique pourquoi, parfois, l'utilisation de la technologie CGI est pratique.

Avec la technologie CGI, le programme (script) est activé en appelant la ressource avec le nom du script. Cela doit par contre être placé avec une référence précise dans la structure du site.

Le script reçoit les paramètres de la requête via la variable d'environnement « QUERYSTRING » et sa sortie est renvoyée du serveur Web vers le demandeur.

La figure 8 met en évidence la collaboration entre les deux environnements, avec une référence particulière à la reconnaissance du sketch et à son éventuel téléchargement automatique.

Cependant, il n'est pas vraiment nécessaire d'utiliser des langages compilés, tels que le C, pour construire des applications sur le RaspberryPi qui utilisent les entrées/sorties d'Arduino. Vous pouvez également utiliser des scripts bash (Linux) ou des scripts en Python.

Il suffit d'utiliser les commandes préparées pour « RandA « (voir le tableau 1) afin de simplifier la tâche.

L'exemple suivant, qui est une petite application de démonstration, indique la valeur de la tension présente sur la broche A1 (provenant d'un photodétecteur, par exemple) et active la LED d'Arduino lorsque la valeur lue est inférieure à un certain seuil.

Le listing 9a reporte la version réalisée avec un script bash, tandis que le listing 9b reporte l'équivalent en Python. Bien sûr, le script est essentiellement didactique, mais avec de petits ajouts, vous pouvez effectuer une détection périodique ou vous pouvez effectuer une vérification à une heure donnée et envoyer une alerte par e-mail.

Ces deux fichiers (« ScriptExample4IO.sh » et « PyExample4IO.py ») se trouvent dans le répertoire : « **/home/pi/bin/examples** ».

Il existe une commande supplémentaire qui permet au RaspberryPi d'utiliser le potentiel d'Arduino, et qui peut être utilisée dans les scripts. Il s'agit de « **ArduInterrupt** ». Ce programme utilise le sketch « **SerialStop** » présent dans le répertoire : « /home/pi/bin/sketch4cmd ». Il s'arrête lorsqu'une certaine condition n'est pas remplie.

La condition peut avoir lieu sur une broche digitale lorsqu'elle prend la valeur 1 ou 0, ou sur une broche analogique lorsque la valeur mesurée est supérieure ou inférieure à un certain seuil.

Par exemple, « ArduInterrupt -ana 2 lt 200 » est suspendue tant que la valeur ne tombe pas en dessous de 200 sur la broche analogique 2 d'Arduino. Il est évident que l'utilisation la plus appropriée de cette commande se situe dans un script où les instructions suivantes implémentent une action.

De plus, le script sera probablement lancé en tâche de fond. Mais comment développer de manière efficace des applications sur le RaspberryPi ?

Avec l'approche Arduino en tant que noyau, nous avons utilisé un emplacement distant (n'importe quel PC connecté au réseau local) pour développer des sketchs qui utilisent le potentiel du RaspberryPi.

De manière inverse, il est aussi possible d'utiliser un emplacement distant

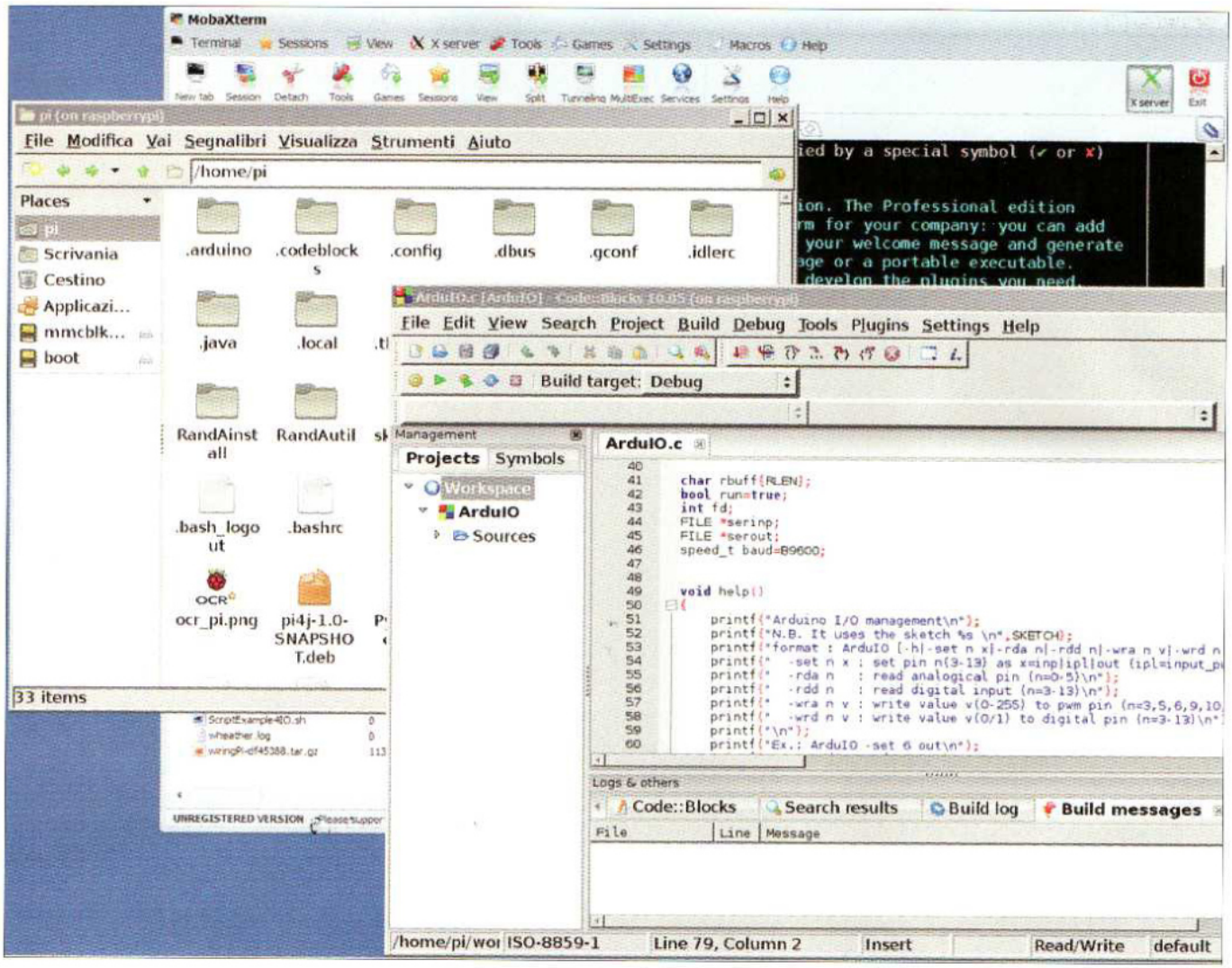


Figure 10 : environnement de gestion centré sur le RaspberryPi.

(toujours notre PC connecté au réseau local), mais cette fois en utilisant une console distante Linux avec le protocole SSH et un environnement graphique.

Pour cela, nous pouvons utiliser un logiciel qui nous permet d'agir comme sur un terminal « X Window » (c'est l'environnement graphique de base de Linux). L'un d'entre eux est « MobaXterm », vous pouvez le télécharger gratuitement à l'adresse suivante : **http://mobaxterm.mobatek.net**.

La figure 10 montre une situation typique dans laquelle sont superposées, grâce à la console « MobaXterm », la fenêtre du système de fichiers et la fenêtre de l'environnement de développement « codeblocks » pour la programmation en C.

Ceux d'entre vous qui sont habitués à utiliser Linux préféreront cette approche, c'est-à-dire le RaspberryPi en tant que noyau, car elle permet une plus grande flexibilité et une utilisation plus efficace du potentiel de la carte « RandA ».

Ceux qui sont plus habitués à l'environnement Arduino, préféreront l'approche précédente. En réalité, il est judicieux de mélanger les deux approches en fonction de l'application.

« RandA » est un système idéal pour l'enseignement et l'approche des technologies de l'information ainsi que des systèmes numériques, car elle permet de couvrir une quantité incroyable de sujets.

Le serveur WEB

La partie logicielle ne se termine pas avec ce que nous venons de décrire, car nous avons également pensé à ajouter un serveur Web. Ce dernier permet une gestion de base du système, mais aussi de la topologie d'un réseau.

En fait, jusqu'à présent, nous avons limité l'utilisation de l'approche à distance exclusivement à un réseau local, à cause du type de logiciel réseau utilisé et pour des raisons de sécurité.

Avec un serveur Web, il est possible de dépasser cette limite, à condition de définir un port de sortie sur le routeur local (port forwarding), c'est-à-dire d'avoir une adresse réseau visible depuis l'extérieur. Le logiciel serveur utilisé n'est pas seulement un serveur Web, mais plutôt une « Web Application Server ».

Celle-ci utilise une approche plus orientée vers l'interaction. Le serveur est basé sur l'un des logiciels gratuits les plus populaires, à savoir « **Tomcat** ».

Malheureusement, un tel logiciel implique l'utilisation de Servlet Java (ou de scripts JSP) pour des applications interactives.

Mais la complexité du logiciel utilisé est compensée par l'efficacité remarquable des applications. Il suffit d'imaginer que tous les serveurs professionnels utilisent ce type d'environnement ou son concurrent « dot-net » (voir encadré « Applications Web et CGI »).

Si vous n'êtes pas un expert en Servlet ou en JSP ne vous inquiétez pas, le serveur a été configuré pour gérer des scripts CGI, c'est-à-dire des scripts Linux ou Python. De cette façon, vous pouvez créer vos applications interactives et les utiliser depuis un navigateur, n'importe où dans le monde.

La version 7 de Tomcat est installée, c'est une version stable. Elle se trouve dans le répertoire « /home/apache-tomcat-7.0.47 ».

Cette version est amplement suffisante pour le type d'application que nous allons développer par la suite, car nous avons rencontré des problèmes avec la version 9 de Tomcat et Eclipse.

Dans ce même répertoire, il existe deux scripts « **startWebS.sh** » et « **stopWebS.sh** » qui permettent de démarrer et d'arrêter le serveur Web. Le premier est appelé dans le fichier système de démarrage « /etc/rc.local ».

Si vous souhaitez désactiver le démarrage automatique du serveur Web, mettez simplement en commentaire cette commande dans le fichier système.

Le serveur Tomcat doit être configuré par le port 80 (au lieu du 8080), cela permet d'afficher directement un site. Par exemple http://www.monsite.com (port 80) au lieu de http://www.monsite.com:8080 (pour le port 8080, il faut à chaque fois ajouter « :8080 »).

Pour cela vous devez modifier le fichier qui se trouve dans le répertoire « /home/apache-tomcat-7.0.47/conf/server.xml ». Modifiez « 8080 » en « 80 ».
Dans le répertoire Tomcat se trouve l'ensemble du serveur Web, c'est-à-dire les applications et les pages statiques.

En particulier, le répertoire « .../webapps/ROOT » contient la page de démarrage « index.html ».

L'application pour la gestion du RaspberryPi et d'Arduino se trouve dans le répertoire « ../webapps/RandA ».

Pour ceux qui souhaitent charger leurs propres applications ou des applications tierces, le plus simple est d'utiliser les fichiers d'archive « .war » (comme sont distribuées généralement les applications Web Java) et les installer via le gestionnaire d'applications de Tomcat.

Cette console est accessible à partir de l'adresse « http://...../manager » et est protégée par nom d'utilisateur (Tomcat est la valeur par défaut) et par un mot de passe (Tomcat est la valeur par défaut).

Le fichier « .war » peut également résider sur l'ordinateur connecté au serveur Web, car la console permet le téléchargement et l'installation automatique en une seule fois. Donc, l'installation des applications s'effectue essentiellement à distance.

Ceux qui souhaitent utiliser des pages HTML simples peuvent les placer dans le répertoire ROOT, où elles seront immédiatement accessibles. Évidemment, des sous-répertoires de ROOT peuvent également être créés.

Par contre, lorsque vous utilisez le **mode CGI**, les **scripts bash doivent être placés** dans le répertoire : « **/home/apache-tomcat-7.0.47/webapps/ROOT/WEB-INF/cgi** ».

Mais, ils doivent être référencés comme : « http: //...../bin-cgi/nomduscript ».

Exemple : http://192.168.1.8/bin-cgi/testcgi2.sh

Les fichiers « testcgi.sh » et « testcgi2.sh » se trouvent dans le répertoire : « ...../ROOT/WEB-INF/cgi ».

Il s'agit de deux exemples de fichiers qui montrent l'utilisation de simples scripts bash pour le dialogue web et l'exécution des commandes Linux.

Si vous voulez utiliser un interpréteur de script différent, vous devez éditer le fichier « /home/apache-tomcat-7.0.47/conf/web.xml » (bloc <servlet> ; paramètre "executable").

La page de démarrage, qui s'affiche en insérant l'adresse de « RandA » dans le navigateur, donne accès à l'application de gestion et à une console web afin d'accéder à Linux depuis le réseau (application tierce).

L'application de gestion n'est pas protégée par mot de passe, mais vous pouvez en ajouter un en éditant le fichier « /home/apache-tomcat-7.0.47/webapps/RandA/WEB-INF/web.xml » et en supprimant le commentaire relatif au bloc d'authentification.

Alors, la console Web a comme nom d'utilisateur « randa » et comme mot de passe « randa ».

L'application de gestion donne accès aux pages et/ou aux applications suivantes :

- sketch de chargement sur Arduino ;
- console d'Arduino ;
- gestion des entrées/sorties d'Arduino ;
- configuration de l'alarme ;
- réglage de l'horloge ;
- extinction de « RandA ».

Les applications listées ci-dessus vous permettent d'avoir un accès distant au système « RandA » à partir d'un réseau, mais le serveur a été installé de manière à être personnalisé par les utilisateurs à l'aide de scripts CGI.

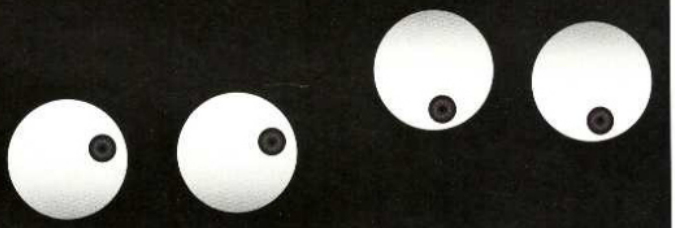
Conclusion

Nous avons décrit dans cet article un système prêt et fonctionnel, pouvant être utilisé dans diverses applications, même avec l'ajout de matériel spécialisé. Tout cela, en maintenant une vision didactique, de loisir et de développement, en commentant le plus possible le logiciel afin de faciliter la personnalisation.

Dans le prochain numéro, nous aborderons l'implémentation d'un serveur web.

STOP AU BLACKOUT!

Première partie



de Roberto Prestianni

Suite aux nombreuses réflexions de nos lecteurs concernant le compteur d'énergie 220 VAC décrit dans le numéro 140 d'Electronique et Loisirs Magazine, nous vous proposons une variante du montage qui répondra aux attentes de ces derniers. Il s'agit donc d'un système de mesure (monitoring) de la consommation électrique de votre maison ou appartement (ou autre) et de limitation automatique de la consommation jusqu'à 6 kW sous 220 VAC. Le montage mesure continuellement la puissance absorbée par les appareils connectés sur le réseau électrique et, en cas de dépassement de la puissance maximale, selon une priorité précise, désactive les appareils qui ne sont pas indispensables afin d'éviter un blackout. Première partie.

Combien de fois sommes-nous brusquement tombés en panne d'électricité ? Surtout lorsque la consommation est particulièrement élevée en raison de l'utilisation massive de climatiseurs l'été ou de radiateurs électriques l'hiver. Il n'est pas rare de voir le disjoncteur se couper subitement. Les pannes dues aux problèmes techniques de notre fournisseur d'électricité sont, heureusement, relativement courtes et sporadiques.

Les pannes d'électricité les plus fréquentes sont celles que nous provoquons lorsque nous consommons plus de puissance que notre abonnement le permet. Par exemple, lorsque nous utilisons la machine à laver et que nous allumons également le four électrique et si en plus nous ajoutons des appareils tels qu'un sèche-cheveux, un radiateur ou une climatisation, (nous ne parlons même pas de la voiture électrique à recharger !), notre fameux compteur électronique Linky décide de déconnecter le réseau électrique.

Pour éviter ces désagréments, nous vous proposons de réaliser un **système anti-blackout** (anti-coupure) décrit dans ces pages. Il s'agit d'un système de contrôle électronique qui **s'occupe de la gestion de la consommation électrique et vérifie que la limite n'est pas dépassée afin d'éviter le déclenchement du disjoncteur**.

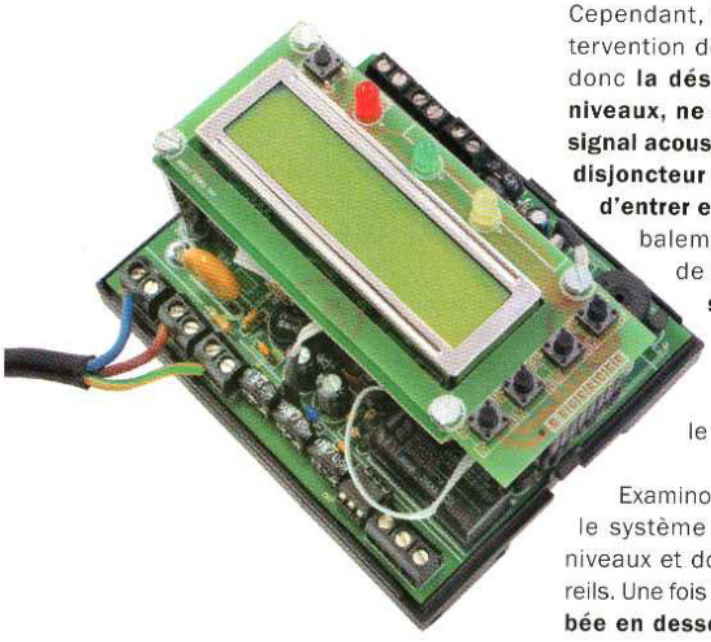
Lorsque la demande en courant de l'installation électrique est trop importante, **il déconnecte automatiquement un ou plusieurs appareils non essentiels** afin d'assurer une consommation électrique dans la limite autorisée.

Cependant, le système ne désactive pas les appareils sans discernement. Grâce à une gestion basée sur des priorités d'utilisation, **il décide d'éteindre en premier les appareils non indispensables**.

Le système se compose d'une unité de contrôle, installée entre le disjoncteur et le reste de l'installation électrique. Cette unité mesure la consommation globale d'électricité et, si nécessaire via un ou plusieurs récepteurs radio, désactive l'alimentation des appareils facultatifs. Par conséquent, ces appareils que vous considérez comme facultatifs et dont vous pouvez vous passer, sont alimentés par l'intermédiaire de modules, de sorte que, si nécessaire, l'unité de contrôle peut les désactiver.

Le système commande les **modules au moyen de signaux radio codés** selon la norme **Motorola**, ce qui permet de les associer à tout récepteur basé sur ce type de codage, à condition que sa partie radio fonctionne à la même fréquence.

L'unité de contrôle se présente sous la forme d'un boîtier en plastique qui s'encastre sur un rail DIN. Elle possède un afficheur LCD, des boutons et des LED. Les modules de commande (« Smart-Rx », c'est-à-dire les récepteurs radio reliés aux appareils) sont intégrés dans un boîtier plastique avec une prise classique pour le raccordement de l'appareil électrique à commander. Ils disposent d'une antenne pour recevoir les commandes émises par l'unité de contrôle, ainsi que des LED et un bouton.



Pour expliquer le fonctionnement du système, supposons que parmi les appareils à commander, nous ayons des lampes, un PC, une machine à laver et un sèche-linge. En cas de consommation excessive, nous définissons un ordre selon lequel ils doivent être déconnectés : 0 (sèche-linge), 1 (machine à laver), 2 (les lampes), 3 (le PC).

Nous avons prévu **7 niveaux de priorité**, soit de 0 à 6. Ainsi, si à un certain moment les 4 appareils sont alimentés et opérationnels et que l'unité de contrôle enregistre une consommation d'énergie excessive, celle-ci émet une commande d'arrêt en premier pour le niveau 0 (sèche-linge). Si cela ne suffit pas, elle passera au niveau 1 (arrêt de la machine à laver).

Ensuite une autre mesure de la consommation est effectuée et si, malgré la déconnexion des charges 0 et 1, la puissance maximale est toujours dépassée, le module de niveau 2 (les lampes) recevra également une commande d'extinction (des lampes).

De cette manière, le PC ne sera pas éteint, à condition que la désactivation des appareils précédents a permis une consommation d'énergie dans les limites de la valeur maximale autorisée.

Cependant, il peut arriver que l'intervention de l'unité de contrôle, et donc **la désactivation de tous les niveaux, ne soit pas suffisante**. Un **signal acoustique** avertit alors que le **disjoncteur de l'installation risque d'entrer en action** en coupant globalement l'électricité. L'unité de contrôle **isole alors le système au moyen d'un relais** conçu pour commander un contacteur optionnel à insérer dans le tableau électrique.

Examinons maintenant comment le système réactive les différents niveaux et donc les différents appareils. Une fois la **consommation retombée en dessous de la valeur limite**, la centrale **tente de reconnecter les appareils dans l'ordre inverse dans lequel elle les a déconnectés** (c'est-à-dire qu'elle réactive le niveau 3 de notre exemple qui correspond au PC). Lors de la reconnexion, elle vérifie si la consommation résultante est dans la limite, elle tente ensuite de reconnecter l'appareil de niveau 2, c'est à dire les lampes.

Supposons maintenant que tous les récepteurs soient éteints et que le relais d'arrêt général ait déconnecté le système afin de ne pas déclencher le disjoncteur. Après environ 30 secondes, le relais est relâché (au repos) pour rétablir l'alimentation du système, ce dernier étant représenté par les différents appareils. L'unité de contrôle mesure de nouveau la puissance consommée et si elle détecte une surcharge persistante, le relais d'arrêt général est activé de nouveau, sinon elle réalimente les différents niveaux désactivés précédemment selon l'ordre inverse.

Si les appareils du premier niveau réactivé ont été débranchés par l'utilisateur et si la surcharge a disparu après 30 secondes supplémentaires, le niveau suivant (par ordre de priorité) est également réactivé et ainsi de suite jusqu'à ce que le système revienne à un fonctionnement normal.

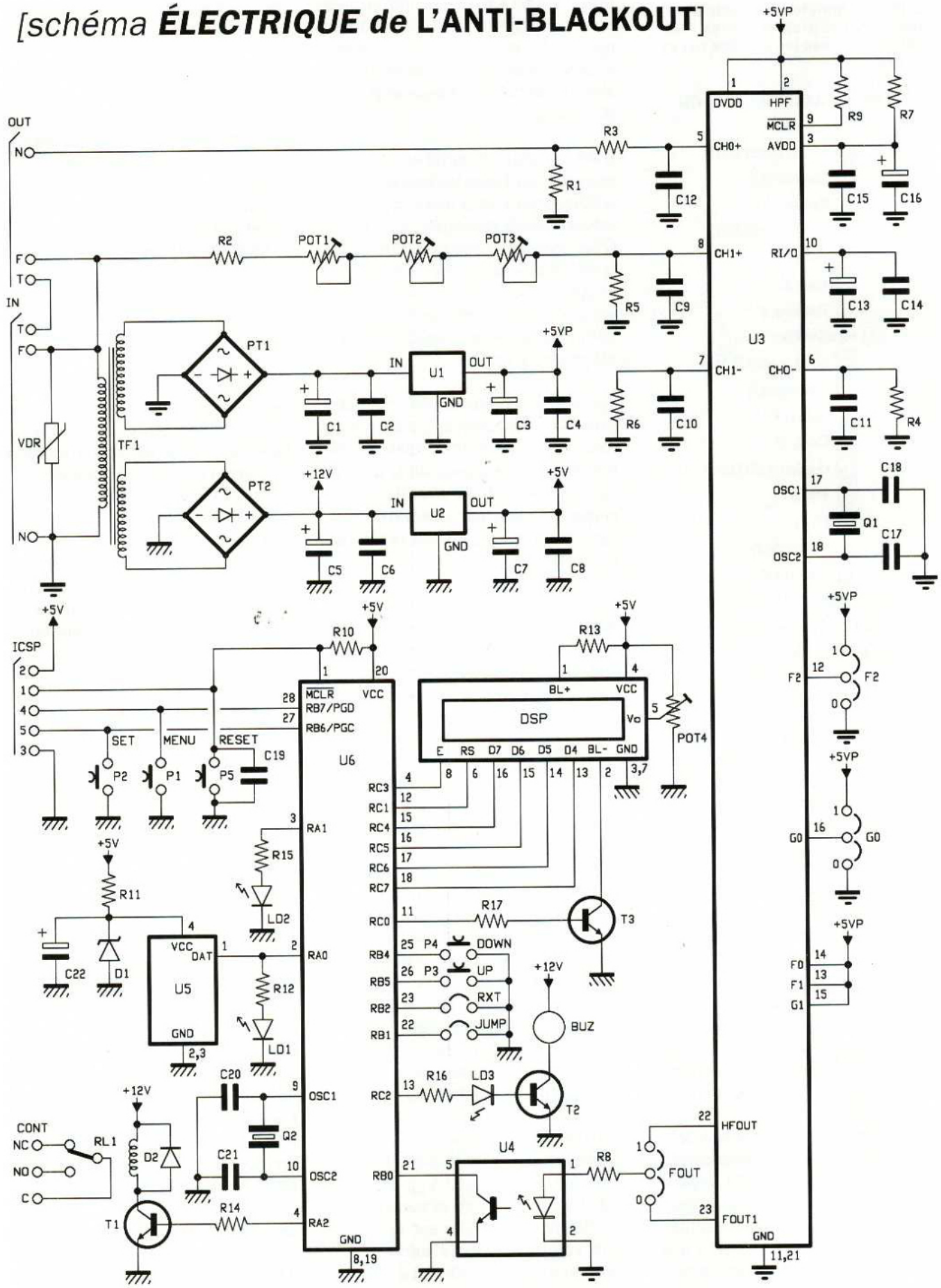
L'unité de contrôle mesure la puissance consommée, maximale et moyenne. Elle permet de définir une puissance maximale pouvant être consommée et active les récepteurs correspondants.

Le schéma électrique

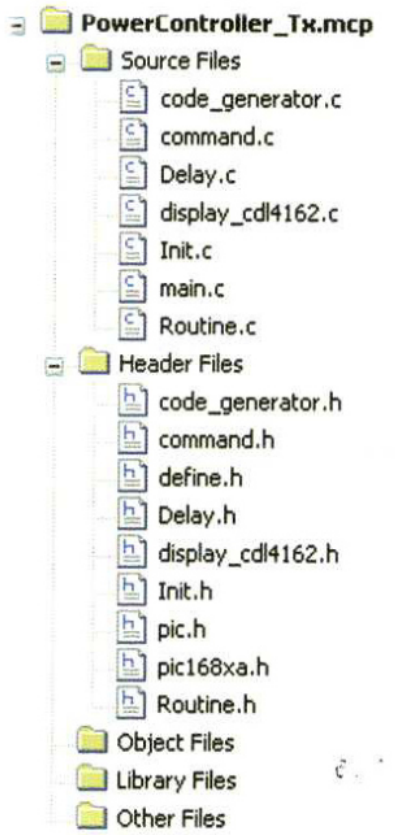
Les principaux composants sont le circuit intégré **MCP3905A** et le microcontrôleur **PIC16F876A**, tous deux fabriqués par Microchip. Le MCP3905A mesure l'énergie consommée d'un réseau électrique monophasé. Ce dernier, dénommé U3 sur le schéma électrique, s'occupe de calculer la puissance instantanée consommée par le réseau électrique en mesurant le courant et la tension instantanés et en calculant le produit.

Le microcontrôleur U6 traite les signaux provenant de U3, pilote l'afficheur LCD

[schéma **ÉLECTRIQUE de L'ANTI-BLACKOUT**]



La liste complète des fichiers qui composent le programme de l'anti-blackout, avec les sources (.c) et les header (.h).



ainsi que l'émetteur HF, gère les boutons, les LED, le relais et toutes les autres parties du montage qui mettent en œuvre les fonctionnalités du système anti-blackout.

Le point de jonction entre les deux parties du montage est réalisé optiquement grâce à l'optocoupleur U4. Ce dernier permet une isolation galvanique entre le PIC et le MCP3905A. Cette condition est nécessaire pour séparer le réseau électrique du reste du montage, car les commandes étant manuelles (boutons), elles sont accessibles par l'utilisateur. Il faut prévenir contre tout risque d'électrocution.

Chaque partie du montage a sa propre alimentation séparée galvaniquement. En effet le transformateur TF1 dispose de 2 enroulements secondaires indépendants. L'alimentation, constituée autour du régulateur U1, fournit une tension de +5 V (+5VP) qui alimente le circuit intégré MCP3905A, ce dernier **n'étant pas isolé du secteur.**

D'autre part l'alimentation formée autour du régulateur U2 fournit 2 tensions +12 V et + 5 V. Le 12 V alimente le buzzer et le relais, tandis que le 5 V alimente le microcontrôleur et le reste du montage.

Il est important de noter que les références, c'est-à-dire les masses des 2 alimentations sont distinctes et ne doivent jamais être reliées ensemble. C'est pour cette raison qu'elles sont représentées de manière différente sur le schéma électrique. La masse de la tension « +5VP » provenant de U1 est différente de celle la tension « +5V » provenant de U2.

Les lignes d'alimentation en entrée (c'est-à-dire les point IN T, F et N) sont reportées en sortie. Une **résistance de 0,001 Ω** est **interposée sur le neutre**, elle est utilisée par le MCP3905 pour **mesurer le courant alternatif** de la ligne sur laquelle seront reliés les appareils.

Cette disposition de R1 fait que **le neutre agit comme une masse** pour le circuit MCP3905 qui mesure le courant, ce qui fournit une sécurité supplémentaire, car la phase ne se propage pas à travers le circuit.

La chute de tension produite dans la résistance de 0,001 Ω pour mesurer le **courant** est appliquée à l'entrée « **CH0+** » du MCP3905 à travers un filtre passe-bas constitué par R3 et C12. La deuxième entrée « **CH1+** » mesure la **tension du secteur** qui est prélevée sur la phase et référencée à la masse (donc au neutre) à travers le diviseur de tension formé par R2, POT1, POT2, POT3 et R5. Le facteur de réduction, une fois le circuit calibré, a une valeur de plusieurs centaines de fois et, dans le pire des cas (POT1, POT2 et POT3 courts-circuités), est égal à 560 (R2 vaut 560 kΩ).

Ceci implique une tension efficace maximale sur l'entrée différentielle CH1 (pour V_{in} = 230 V_{eff}) égale à 0,411 V_{eff} et donc une valeur crête de ± 0,580 V. Cette valeur est inférieure à la valeur maximale admise sur cette entrée (± 0,660 V). La tension différentielle applicable sur l'entrée « CH0+ » a également une limite, elle est égale à ± 470 mV/G, où **G** représente le **gain**. Dans notre projet G1 = 1 et G0 = 0, alors le gain sera égal à 8 et donc la limite de la tension est de ± 60 mVcc.

Les condensateurs C9 à C12 servent à filtrer les tensions sur les entrées « CH1+ », « CH1- », « CH0+ » et « CH0- ». Les entrées « CH1 » détectent l'**angle de phase de la tension**, ainsi le MCP3905 le **compare à celui du courant** qui est mesuré à l'aide des entrées « **CH0** ». Cela permet donc de déterminer le **déphasage entre la tension et le courant**.

La fréquence d'horloge du MCP3905 est générée par le quartz Q1 et les deux condensateurs C17 et C18. La ligne F2 (broche 12) doit être mise à zéro via le cavalier F2, de manière à obtenir une fréquence de 27968,75 Hz qui correspond à la fréquence de référence de la sortie « **HFc** » (impulsions). Cette particularité permet d'effectuer une lecture de la puissance instantanée en seulement 2 secondes sans perte de définition due à la conversion numérique.

Le cavalier « FOUT », d'autre part, permet de sélectionner l'une des deux sorties impulsives du circuit intégré, nous utilisons pour notre projet la sortie « HFOUT ».

Enfin, les composants R7, C15 et C16 réalisent un simple filtre qui alimente la partie analogique du circuit MCP3905. Ainsi la tension de 5 V est correctement filtrée pour supprimer toute perturbation sur la broche « Avdd ». La **varistance VDR protège d'éventuels pics de tension** qui pourraient se produire sur le réseau.

Examinons maintenant le microcontrôleur PIC. Les **impulsions** qui doivent être **comptées pour calculer la puissance** sont appliquées sur la broche **RB0**. Les impulsions parviennent indirectement de l'**optocoupleur U4** afin d'obtenir une **isolation galvanique**.

Les **boutons** de gestion du menu (MENU, SET, UP, DOWN) sont reliés aux 4 broches **RB4** à **RB7**, qui sont **configurées en entrées** et dont les **résistances internes de pull-up sont activées**.

Deux de ces lignes sont connectées au connecteur **ICSP** à 6 pôles permettant la **programmation en circuit du PIC**. La réinitialisation (reset) du microcontrôleur au démarrage est obtenue à l'aide de R10 et C19. Il est possible d'effectuer un **reset manuel** en appuyant sur le bouton **P5**.

L'unité de contrôle transmet les codes Motorola pour contrôler la mise en marche et l'arrêt des charges connectées aux modules actionneurs. La **liaison radio** est de type à **modulation d'amplitude** et utilise le module hybride **Aurel TX-4M10HA**.

Celui-ci est équipé d'une antenne intégrée et d'un oscillateur d'une puissance de **10 mW**. Il est alimenté par une tension de 3 V obtenue par le réseau R11, D1 et C22. Chaque fois qu'une **transmission** se produit, la **LED jaune LD1 clignote** pendant toute la durée.

La déconnexion complète de l'installation électrique s'effectue via le relais **RL1**, piloté par le transistor T1. La LED LD2, pilotée directement par la broche RA1 à travers la résistance de limitation R15, signale chaque mesure de la consommation électrique.

Le **buzzer est de type à oscillateur intégré** et avec LD3, ils indiquent acoustiquement et visuellement toute **surcharge**. L'afficheur LCD est de type rétroéclairé à 2 lignes de 16 caractères chacune. Il est piloté par le PIC sur 4 bits de données (D4 à D7) ainsi que les lignes de contrôle E et RS. La ligne « BL- » de l'afficheur est relié au collecteur de T3. Lorsque ce dernier est rendu conducteur via la broche RC0 du PIC, le rétroéclairage du LCD s'allume. La résistance R13 permet une limitation du courant de rétroéclairage et le trimmer « POT4 » permet le réglage du contraste de l'afficheur.

Le PIC est cadencé à l'aide du quartz Q2 de 16 MHz ainsi que des 2 condensateurs C20 et C21.

Le programme (firmware)

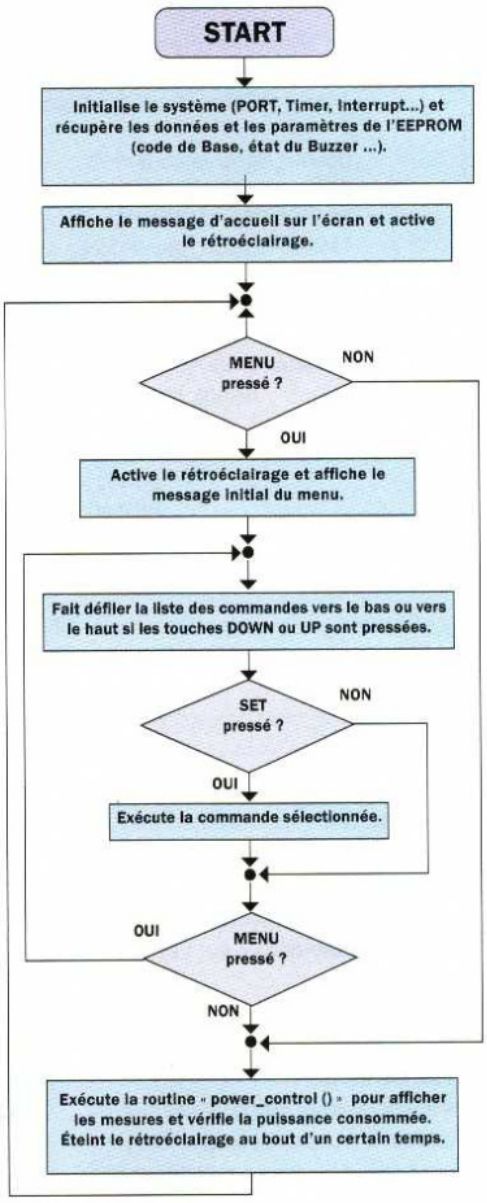
Le programme sera disponible en téléchargement sur notre site web, vous y trouverez le programme au format « .hex » du PIC16F876A mais aussi l'ensemble du projet avec tous les fichiers source en langage ANSI C, dont la liste (y compris les fichiers header avec l'extension « .h ») est indiquée dans la figure appropriée. Le fichier « **main.c** » contient à la fois le programme principal « main () » du projet et la routine d'interruption.

Le programme principal s'occupe essentiellement de la gestion du menu et de l'appel des commandes correspondantes. Il effectue la lecture de l'état des boutons (MENU, SET, UP, DOWN) pour faire défiler vers le bas, vers le haut, et sélectionner l'une des 14 commandes. Lorsque la commande est sélectionnée, la routine respective est appelée et exécutée immédiatement.

La routine d'interruption effectue un comptage long en utilisant le TIMER1 avec un registre de 16 bits. Cela est nécessaire pour l'extinction automatique du rétroéclairage de l'afficheur et l'activation de ce dernier si une pression sur l'un des 4 boutons est détectée.

Le fichier « command.c » définit toutes les commandes implémentées par le système, et donc les fonctions relatives telles que :

- « **print_power_mas()** » qui calcule et affiche la puissance maximale mesurée depuis le dernier reset ;
- « **print_power_average()** » qui calcule la moyenne des mesures des puissances instantanées ;
- « **power_control()** » qui effectue le contrôle de la puissance en commandant les modules actionneurs des différents niveaux de priorité, ainsi que le relais de surcharge, le buzzer et la LED rouge qui signalent cette condition (cette fonction sera mieux examinée ci-après) ;
- « **active_all_Rx()** » permet l'activation forcée de tous les modules actionneurs ;
- « **disactive_all_Rx()** » de manière complémentaire à la précédente, elle désactive de manière forcée tous les modules actionneurs ;
- « **print_level_Rx()** » affiche à plusieurs reprises le niveau de priorité des « Smart-Rx » uniquement à travers leur LED verte ;
- « **reset_system()** » permet une réinitialisation complète du système après la restauration des valeurs par défaut ;
- « **insert_base_code()** » permet de modifier le code de base ainsi que d'entrer les niveaux de priorité à



Structure générale du firmware, l'organigramme reflète le contenu du programme principal du microcontrôleur.

La routine suivante a comme argument le « flag_menu », défini à l'aide d'une variable d'un octet (unsigned char). Le flag (drapeau) est mis à « 1 » par la fonction d'appel, cela permet l'interruption de la lecture de l'état des boutons au moment où la touche « MENU » est pressée. Si le flag est mis à « 0 », la lecture n'est pas interrompue, même si ce bouton est pressé. Cela est nécessaire lorsque l'anti-blackout est dans une phase de limitation de la puissance instantanée et, par conséquent, dans la phase de désactivation et de contrôle des modules actionneurs (voir dans le listing la première instruction « if(...) »).

D'abord, les variables de comptage « power_ist » et « counter » sont réinitialisées. La première est de type « unsigned short », il s'agit d'une variable sur 2 octets sans signe. Elle compte, à chaque cycle de lecture, le nombre d'impulsions qui se présentent sur la broche « HFout » du MCP3905A.

La seconde variable est utilisée pour compter la durée de la lecture qui est de 2 secondes. Au maximum, la variable « power_ist » atteindra une valeur un peu supérieure à 6000, ce qui correspond à la valeur maximale de la puissance active nominale attendue par l'anti-blackout.

Listing 1

```
void power_read(unsigned char flag_menu)
{
        //**** Cacul de la puissance ***********************************************

        power_ist=0;counter=0;                  // réinitialisation des variables
        INTF=0;T0IF=0                           // réinitialisation initiale des flags
        PS2=1;PS1=1;PS0=1;                      //prescaler TIMER0 1:256
        TMR0=5;num_sec = 125;                   //initialisation pour le comptage des 2 secondes

//*** Comptage du nombre d'impulsions de « HFout » durant les 2 secondes ***

        while(counter<num_sec)                                          {
                if(flag_menu==1 && menu==0)
                {
                        flag_menu=0;            //reset du flag
                        power_ist=0;counter=0;  //reset des variables de comptage
                        PS2=0;PS1=1;PS0=1;      //restauration du prescaler TIMER0 à 1:16
                        return;                 // sort si le bouton MENU est pressé
                }

// Comptage des impulsions sur « HFout » ***************************************

if(INTF)
```

transmettre aux modules actionneurs, lors de la phase d'auto-apprentissage ;

- « **print_base_code()** » affiche le code de base actuel sur l'afficheur LCD ;
- « **buzzer_select()** » active/désactive le buzzer en cas de surcharge non contrôlée ;
- « **write_power_max()** » permet de définir la puissance maximale pouvant être délivrée sans intervention des modules actionneurs ;
- « **write_priority_max()** » permet de définir le plus haut niveau de priorité géré par le système en cas de surcharge ;
- « **power_read()** » mesure la puissance instantanée, elle est appelée par plusieurs fonctions ;
- « **print_power()** » affiche la puissance mesurée.

Le fichier « code_generator.c » gère et implémente toutes les transmissions des codes Motorola, il contient les routines suivantes :

- « **init_code_generator()** » récupère de l'EEPROM du PIC le paramètre qui définit la fréquence d'émission des bits de chaque code (delay_timer0) ;
- « **set_freq()** » permet de faire varier cette fréquence en modifiant le même paramètre que précédemment et permet de plus l'acquisition du code de base et du niveau de priorité des récepteurs avec auto-apprentissage ;
- « **trans_code()** » est la routine qui implémente le codage Motorola et prend en charge toutes les autres fonctions ;
- « **code_TX()** » génère individuellement les bits à 3 états et les transmet ;
- « **command_code()** » code les commandes qui doivent être envoyées aux différents modules actionneurs (y compris les commandes spéciales « Smart-Rx ») ;

Ensuite, les flags « INTF » et « TOIF » sont réinitialisés, puis ils sont respectivement utilisés pour signaler un front descendant sur la ligne « INT/RB0 » du PIC (c'est-à-dire sur la sortie « HFout ») et pour le comptage du « TIMER1 ». Ce dernier compte le nombre d'intervalles de 16 ms, au total num_sec = 125 soit 16 ms * 125 = 2 s).

Après ces initialisations, la boucle « while () » commence le comptage des 125 cycles. La première instruction rencontrée est « if(flag_menu==1 && menu==0) » décrite précédemment et la suivante est « if(INTF) » qui interroge le statut du flag « INTF ». Chaque fois que ce flag est mis à « 1 », une impulsion apparaît et la variable « power_ist » est incrémentée.

La dernière instruction « if(TOIF) » teste si le flag « TOIF » est mis à « 1 », dans ce cas la variable « counter » est incrémentée.

Une fois la série de cycles terminée, la LED verte clignote signalant la fin de la mesure. La dernière instruction est utilisée pour calculer la puissance maximale enregistrée, cela s'effectue en comparant la mesure en cours avec la puissance maximale enregistrée jusqu'à présent (power_mas).

```
                {
                        ++power_ist;        //incrémentation du comptage des impulsions du capteur de courant

                        INTF=0;
                }

//Fin du comptage du TIMER0 ********************************************

                if(TOIF)
                {
                        ++counter;                  // incrémentation du comptage pour attendre 2 secondes
                        TOIF=0;TMR0=5;
                }
        }
        PS2=0;PS1=1;PS0=1;                          // restauration du prescaler TIMER0 à 1:16
        LED=1;DelayMs(100);LED=0;                   // impulsion lumineuse de la LED pour la fin de la mesure
        if(power_ist>power_mas) power_mas=power_ist;    // détection de puissance maximale

//*************************************************************************
//NOTE : Pour chaque W la fréquence de la sortie « HFout » est augmentée de 0, 5 Hz      *
//       (Exemple : 1 000 W correspondent à 500 Hz)                                     *
//       Cela signifie que pendant les 2 secondes, la lecture enregistre un nombre d'impulsions  *
//       égal à la mesure de la puissance (par exemple : 1 000 W → 1 000 impulsions en 2 secondes)  *
//*************************************************************************
}
```

- « **halfperiod()** » modifie la fréquence d'émission des codes ;
- « **write_new_code()** » permet la définition d'un nouveau code de base.

Le fichier « Routine.c » définit certaines fonctions secondaires telles que :

- « **welcome_message()** » affiche le message de bienvenue sur l'écran LCD chaque fois que le système est réinitialisé ;
- « **lamp_led()** » et « **more_lamp()** » produisent respectivement un seul clignotement rapide et une série de clignotements rapides de la LED ;
- « **print_menu_command()** » affiche le message d'entrée dans le menu.

Les fichiers sources de la liste non encore examinés sont « Init.c » et « Delay.c ». Le premier définit la routine « **Init_system()** » qui permet la **configuration initiale** de tous les périphériques du microcontrôleur (PORT, TIMER, oscillateur, interruption, etc.). Le second contient les fonctions qui génèrent les retards. Plus précisément, « **DelayMs()** » réalise un retard **multiple de 1ms**. « DelayMss() », identique à la première, est utilisée seulement par la routine d'interruption. « **Delay100Us()** » génère un **retard multiple de 100 µs**, tandis que « **Delay1S()** » génère un **retard multiple de 1s**.

Le fichier qui contient toutes les routines de gestion de l'afficheur CDL4162 (2 lignes de 16 caractères) se nomme « **display_cdl4162.c** » qui, dans ce contexte, n'est pas examiné.

Pour terminer la description des rôles de tous les fichiers du projet, examinons maintenant les « **header files** » (fichiers d'en-tête) qui recueillent tous les fichiers header associés aux fichiers « .c » examinés précédemment, à l'exception des fichiers « pic168xa.h », « pic.h » et « define.h ».

La première instruction « if (...) » vérifie si la variable « power_ist » est inférieure à 400 W et, dans ce cas, un message est envoyé à l'afficheur. Dans le cas contraire (power_ist > 400 W), l'afficheur doit indiquer la valeur numérique exacte de la puissance instantanée. Si la mesure est supérieure à 999 W, la routine affichera le nombre de kW sur l'écran LCD, sinon le nombre complet de watts est affiché (nombre sans virgule).

La même fonction permet l'affichage de tous les niveaux de priorité désactivés (mis sur OFF) par le système, lorsqu'un événement de surcharge se produit (voir l'instruction « switch(counter) »). Cette routine d'affichage est en fait utilisée par la fonction « power_control() » lorsque la puissance maximale est dépassée. Cependant, dans d'autres parties du code source, il y a le même morceau de code lié à l'affichage instantané de la puissance.

Listing 2

```
void print_power(unsigned char *string)
{
        if(power_ist<=400)                  //puissance inférieure à 400 W
        {
                strncpy(string, "Potenza < 400W  ",16);LCD_Print_string16(string,1);   //première ligne sur le LCD
        }
        else                                // puissance supérieure à 400 W
        {
                if(power_ist>=1000)         // affichage du nombre de kW (plus de 999 W)
                {
                        calc2=power_ist-(power_ist/1000)*1000;calc1=power_ist/1000;
                        if(calc2<100)
                        {
                                if(calc2<10)
                                {
                                        if(calc2==0) sprintf(string,"Power = %1d,000 KW",  calc1);
                                        else   sprintf(string,"Power = %1d,00%1d KW",calc1,calc2);
                                }
                                else      sprintf(string,"Power = %1d,0%2d KW", calc1,calc2);
                        }
                        else sprintf(string,"Power = %1d,%3d KW", calc1,calc2);
                        LCD_Print_string16(string,1);                   //écrit sur la 1ère ligne
                        }
                else                                    //affiche le nombre de W (jusqu'à 999 W)
                {
                        sprintf(string,"Power = %5d W ", power_ist);LCD_Print_string16(string,1);        //première
ligne sur le LCD
                }
        }
        strncpy(string, "LIV.OFF:   ",16);LCD_Print_string16(string,2);   //affichage des niveaux désactivés (OFF) sur
la 2ème ligne du LCD

for(counter=0;counter<level;counter++)
        {
                LCD_Locate(2,9+counter);
                switch(counter)
                {
                        case 0:LCD_Putc('0');break;
                        case 1:LCD_Putc('1');break;
                        case 2:LCD_Putc('2');break;
                        case 3:LCD_Putc('3');break;
                        case 4:LCD_Putc('4');break;
                        case 5:LCD_Putc('5');break;
                        case 6:LCD_Putc('6');break;
                        default:break;
                }
        }
    }

}
```

Le premier définit les **noms et adresses des registres**, des **ports** et divers **périphériques** du PIC16F876A (U6), tandis que le second est un fichier de support. Le troisième définit toutes les constantes du projet, c'est-à-dire les **adresses EEPROM**, les pseudonymes des différents ports, les **valeurs des paramètres** de fonctionnement et les chaînes de caractères du menu. Par souci de simplicité, les arguments et les types de variables retournés par chaque routine ne sont pas spécifiés.

Maintenant, regardons l'organigramme ou diagramme de flux (flow-chart) du projet. Il reflète à peu près le contenu du programme principal « main () ».

Lorsque le microcontrôleur est mis sous tension, le programme interne exécute la routine « Init_system() » qui fournit les configurations des registres et des périphériques internes, en particulier, elle attribue une valeur logique aux ports d'entrées/sorties connectés aux périphériques externes tels que les LED, le buzzer, le relais et le module radio HF.

Une fois que les données et les paramètres provenant de l'EEPROM sont chargés, un message de bienvenue s'affiche.

À ce stade, si la touche est enfoncée, l'utilisateur accède au menu des commandes, qui présente un message de bienvenue.

Après environ 20 secondes, la fonction « power_control() » est exécutée, elle mesure la puissance instantanée et affiche la valeur sur l'écran LCD.

Une fois dans le menu, si l'utilisateur appuie sur le bouton « DOWN » (bas), la première des 2 commandes s'affiche.

Pour **faire défiler les commandes**, il faut appuyer sur les boutons « **UP** » ou « **DOWN** » (haut ou bas).

Une fois que la commande désirée s'affiche, il faut appuyer sur le bouton « **SET** » pour qu'elle soit sélectionnée et **exécutée**.

À la fin de la routine, le programme retourne à la fonction principale « main () », qui appellera alors immédiatement la routine « power_control() ».

Il convient de noter que chaque fois que le bouton MENU est relâché, la sélection de la commande en cours est automatiquement abandonnée et le cycle de mesure de la puissance démarre.

En cas de surcharge, une partie de la routine « power_control() » est exécutée pour contrôler les modules actionneurs.

Avant d'aborder cette routine importante, examinons les **Listing 1** et **Listing 2**, qui reportent respectivement la fonction de mesure instantanée « power_read() », et celle de l'affichage de ces mesures, c'est-à-dire « print_power() ».

En ce qui concerne la routine « power_control() », si aucune surcharge n'est détectée sur le réseau électrique, elle effectue simplement la mesure et l'affichage de la puissance active instantanée sur l'écran LCD.

Si, par contre, la puissance maximale est dépassée, la routine adopte toutes les mesures nécessaires pour que la consommation du système soit dans les limites autorisées.

Par conséquent, aussi longtemps que l'ensemble des appareils électriques allumés nécessite une puissance totale supérieure à la valeur maximale, cette routine restera continuellement opérationnelle.

Initialement, tous les niveaux de priorité sont activés (tous les appareils sont alimentés), et après une première mesure de la puissance instantanée et l'initialisation de certaines variables, la **valeur détectée** par « **power_ist** » est **comparée à la valeur de la puissance maximale mesurée précédemment** (c'est-à-dire la valeur de la fonction « power_max »).

Si cette dernière n'est pas dépassée, la routine se termine et le programme revient à la boucle principale « main() », sinon le programme cherche le dernier niveau de priorité désactivé (s'il y en a un).

Si c'est le cas, le niveau de priorité le plus bas qui se trouve encore sur ON est désactivé afin de limiter immédiatement la consommation d'énergie.

Ensuite le programme passe au niveau de priorité supérieur. Lorsque tous les niveaux sont désactivés (sur OFF), l'anti-blackout effectue la déconnexion forcée du système électrique.

Lorsque la variable de comptage « flag_power_max » est égale à la valeur définie avec « PMX_mess_num « (nombre de fois que le message d'avertissement doit être affiché avant la déconnexion forcée du système électrique), le relais reste enclenché pendant 30 secondes.

Après ce délai, l'anti-blackout remet le système sous tension (relais sur OFF) afin de vérifier si la surcharge persiste. Si c'est encore le cas, le cycle est répété avec les messages d'avertissement et une nouvelle activation du relais.

Si « **power_ist ≤ power_mas** », l'anti-blackout tente de **réactiver** les modules actionneurs de tous les niveaux un à la fois, **en commençant par celui qui a la plus haute priorité** (voir le côté gauche de l'organigramme).

Le module actionneur réactivé est maintenu sur ON si « **power_ist ≤ power_mas** », sinon il est de nouveau désactivé.

Tableau 1 - Les cavaliers de l'unité de contrôle. En gras, les paramètres par défaut. G0 est à 0 pour les systèmes jusqu'à 4,5 kW et sur 1 pour les puissances de 4,5 kW à 6 kW. RXT permet de régler de la compatibilité avec les récepteurs.

CAVALIER	1	0	OUVERT	FERMÉ
G0	G=16	G=8	-	-
F2	HFc = 219,51 Hz	**HFc = 27968,75 Hz**	-	-
Fout	**Fout=HF_{OUT}**	Fout=Fout1	-	-
RXT	-	-	seulement les Smart-RX	**tous les RX**

Listing 3

```
void command_code(unsigned char command,unsigned char level, unsigned char *string)
{
        // Chargement du code de base

for(counter=0;counter<6;counter++)
        {
                code_temp1[counter]=EEPROM_READ(EEADD_base_code+counter);
                code_temp2[counter]=code_temp1[counter];
        }
        //Commandes d'activation/désactivation des actionneurs/Rx

if(command == ON_level || command == OFF_level)
        {
                switch (level)
                {
                        case  0: code_temp1[6]=ZERO;   code_temp1[7]=ZERO;   break;
                        case  1: code_temp1[6]=ZERO;   code_temp1[7]=ONE;    break;
                        case  2: code_temp1[6]=ZERO;   code_temp1[7]=OPEN;   break;
                        case  3: code_temp1[6]=ONE;    code_temp1[7]=ZERO;   break;
                        case  4: code_temp1[6]=ONE;.   code_temp1[7]=ONE;    break;
                        case  5: code_temp1[6]=ONE;    code_temp1[7]=OPEN;   break;
                        case  6: code_temp1[6]=OPEN;   code_temp1[7]=ZERO;   break;
                        default: code_temp1[6]=ZERO;   code_temp1[7]=ZERO;   break;
                }

// Copie du code pour la deuxième transmission

code_temp2[6]=code_temp1[6];  code_temp2[7]=code_temp1[7];
                if(command == ON_level)
                {
                        code_temp1[8]=ONE;     code_temp2[8]=OPEN;
```

La routine « command_code() » implémente la transmission des commandes aux modules actionneurs. Cette routine a comme arguments « level » et « * string », c'est à dire la commande à transmettre vers le niveau auquel elle doit être adressée, et l'adresse d'un tableau (array) pour stocker les chaînes de caractères à afficher sur l'écran LCD.

Les commandes implémentées sont : « ON_level », « OFF_level », « print_priority », « ON_all », « OFF_all » et « chg_base_code ».

Les deux premières commandes permettent l'activation et la désactivation d'un niveau particulier, « print_priority » effectue la commande 9 pour visualiser le niveau de priorité sur le « Smart-Rx », tandis que « ON_all » et « OFF_all » active et désactive tous les niveaux (du niveau 0 jusqu'à la valeur la plus élevée spécifiée dans la phase de configuration initiale).

Enfin, « chg_base_code » modifie automatiquement le code de base des « Smart-Rx » chaque fois qu'avec la commande 7, le code de base est modifié.

Les premières lignes de code de la routine permettent de charger le code de base dans les 6 premiers bits des variables « code_temp1 » et « code_temp_2 » stockées dans la mémoire EEPROM à partir de l'adresse de « EEADD_base_code ».

Le double code est dû au fait que, pour certaines commandes, il est nécessaire d'utiliser un code spécifique pour les modules de types « Smart-Rx » et un autre code pour les récepteurs de types communs.

Pour maintenir la compatibilité avec des récepteurs à deux canaux, il est nécessaire d'émettre deux codes qui diffèrent par le dernier bit.

Le code de la commande « ON_level » signale qu'après le codage des bits 7 et 8 (code_temp1[6] et code_temp1[7]), basé sur la valeur de « level » (le niveau requis), le bit 9 dans « code_temp1 » sera à « 1 » et dans « code_temp2 » il vaudra « H ».

```
                }
                else if(command == OFF_level)
                {
                        code_temp1[8]=ONE;      code_temp2[8]=ZERO;
                }

// Transmission du code d'activation/désactivation pour les Smart_Rx

                trans_code(code_temp2,7);
                        DelayMs(250);DelayMs(250);      // retard

// Transmission du code d'activation/désactivation pour tous les autres types d'actionneurs

                if(rx_en==0)trans_code(code_temp1,7);   // seulement après avoir inséré le cavalier

// NOTE : cette séquence est importante car le Rx du projet lit la commande du premier code reçu

}
        // Commandes spéciales pour les actionneurs / Rx

else

{

//********* Visualisation des niveaux de priorité avec les LED **************************

if(command == print_level)
                {
                        code_temp1[6]=OPEN;    code_temp1[7]=ONE;      code_temp1[8]=ZERO;
                        trans_code(code_temp1,7);          // Transmission du code de commande
                }
```

Le premier active le canal 1 du récepteur 2 canaux, tandis que l'autre active le canal 2 (également celui du « Smart-Rx »). Il en va de même pour la commande « OFF_level » où le bit 9 prendra les valeurs « 1 » et « 0 ».

La transmission des deux codes s'effectue avec un retard de 0,5 seconde, afin de permettre à tous les récepteurs à deux canaux de faire commuter le premier canal puis le deuxième.

Les dernières lignes du code concernent l'affichage du niveau de priorité attribué à tous les « Smart-Rx ». Un code spécial est utilisé sur les 6 premiers bits du code de base, les 3 derniers bits étant égaux à « H-1-0 ».

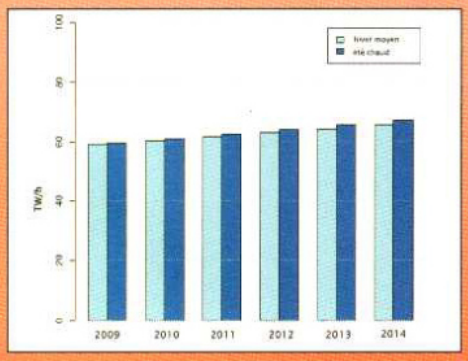
En étant à proximité d'un « Smart-Rx » ayant le même code de base, lorsque le bouton de transmission d'une radiocommande configurée avec ce code spécial est pressé, la LED verte du module actionneur clignote un nombre de fois égal à son niveau de priorité.

Le cavalier peut être ouvert ou fermé (absent ou présent). Les valeurs en gras dans le Tableau 1 sont les valeurs préférentielles.

Le cavalier « G0 » doit être à « 0 » si l'anti-blackout est installé dans des systèmes de 4,5 kW ou moins, sinon il faut le mettre à « 1 ».

Le cavalier « F2 » doit être à « 0 » pour que U3 fonctionne à la fréquence maximale. Le cavalier « FOUT » doit être à un niveau bas pour sélectionner « HFOUT ».

Le cavalier « RXT » doit être fermé pour que l'unité de contrôle puisse commander tous les récepteurs Motorola (et non pas seulement les types « Smart-Rx »).



Le blackout : aujourd'hui pire qu'hier

Quelle est la cause de l'augmentation des pannes électriques ? Le problème réside dans le réseau de transport et de surveillance de l'énergie, auquel l'ouverture du marché a demandé de fournir de l'électricité à n'importe qui et partout. Pour limiter les pertes et les coûts des centrales électriques, le courant injecté dans le réseau doit être consommé relativement proche afin de diminuer les pertes par effet Joule.

Si 100 MW manquent dans le sud de la France et que l'Allemagne fournit 200 MW (avec des centrales à charbon !), le problème se pose dans le transport de l'électricité sur des centaines de kilomètres qui provoque des pertes et donc des coûts importants.

Si sur le papier le système électrique national a assez d'énergie, il peut arriver que localement il soit mal distribué. Cela provoque des pannes électriques. À la lumière des augmentations estimées de la consommation dans les années à venir (le graphique montre l'évolution de la consommation de 2009 à 2014) et en supposant que de plus en plus de voitures électriques seront reliées au réseau, la solution est de disposer d'un système automatique capable de suivre localement l'approvisionnement et la distribution territoriale en fonction de la consommation locale.

Réalisation pratique

L'unité de contrôle de l'anti-blackout est composée de 2 circuits imprimés. Le principal, où seront installés la plupart des composants, et celui de l'afficheur LCD qui est relié au circuit imprimé principal par l'intermédiaire d'une nappe et sur lequel seront montés les boutons et les LED.

Les typons des circuits sont téléchargeables dans le sommaire détaillé de la revue 143.

Les circuits imprimés sont facilement reproductibles, bien que le circuit principal soit un circuit double face il est relativement simple à fabriquer. Il faut être soigneux et ne pas oublier de souder les « vias ».

Un via est un trou métallisé qui permet d'établir une liaison électrique entre deux ou plusieurs couches. À l'aide de morceaux de pattes de composants, soudez les trous qui sont en correspondance avec les faces inférieures et supérieures.

Puisque l'unité de commande est traversée par l'ensemble du courant circulant dans les appareils électriques reliés (le courant peut atteindre environ 27 A sous 220 VAC), les pistes de cuivre de la phase et du neutre doivent être renforcées, en particulier le segment de la piste du neutre qui relie le bornier au shunt.

Les premiers composants à monter sont le circuit intégré MCP3905A (U3) et la résistance R1.

Faites attention au circuit intégré, car il s'agit d'un boîtier CMS disposé sur la face « côté soudures » du circuit imprimé (en dessous). Ces 24 broches sont très proches les unes des autres.

Les autres composants sont de type traversant, toutes les résistances doivent être montées verticalement à l'exception de R2, R3 et R11. Vous devez souder ces dernières, puis ensuite les diodes en respectant l'orientation.

Continuez à souder les condensateurs non polarisés, la varistance « VDR » puis les condensateurs électrolytiques en respectant la polarité (la patte la plus longue correspond au positif).

Continuez avec les transistors, les 2 régulateurs toujours en vérifiant plusieurs fois avant de souder l'orientation que les composants doivent prendre.

Pour l'optocoupleur et le PIC nous vous conseillons d'utiliser des supports de circuits intégrés. Continuez en soudant les potentiomètres, l'émetteur U5, le relais, les borniers.

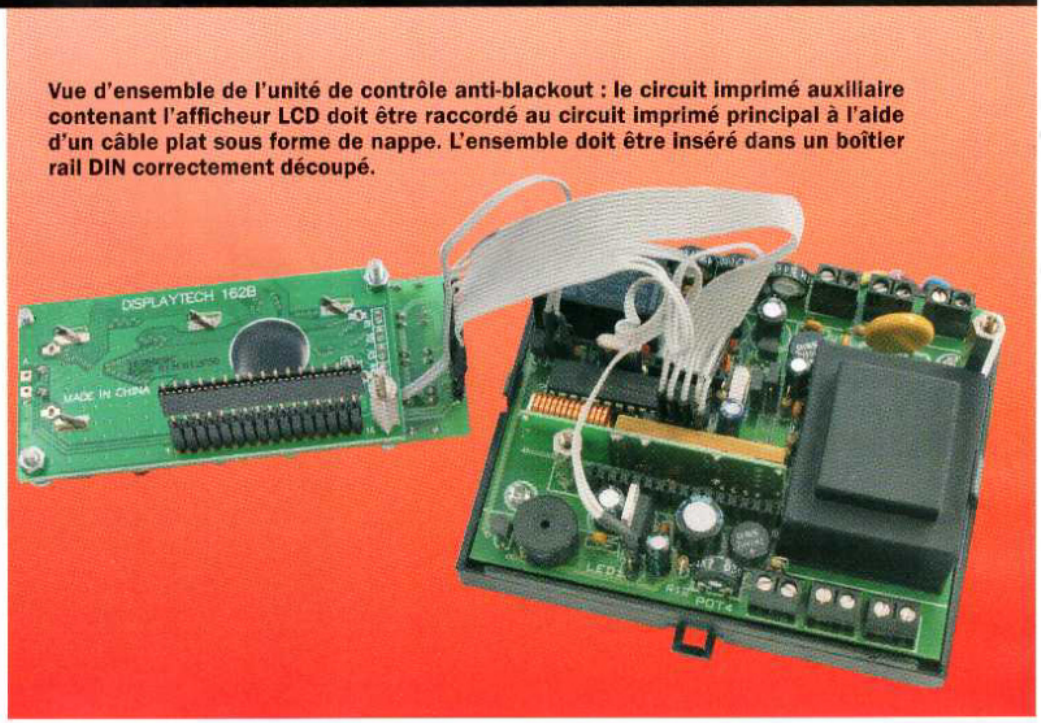
Le transformateur doit être soudé en dernier, car sinon il serait difficile de manipuler le circuit imprimé à cause du poids.

En ce qui concerne le module LCD, les LED et les boutons poussoirs, ils doivent être soudés sur le circuit imprimé auxiliaire. L'afficheur sera relié au circuit principal à l'aide d'une nappe de fil terminée par un connecteur femelle à 16 broches d'environ 8 mm de haut.

Le brochage du connecteur « CONN » du circuit imprimé auxiliaire est le suivant :

- 1 : masse ;
- 2 : MENU ;
- 3 : SET ;
- 4 : UP ;
- 5 : DOWN ;
- 6 : LED1 (jaune) ;
- 7 : LED2 (verte) ;
- 8 : RESET.

Notez aussi que le corps du bouton « RESET » doit être plus court afin d'éviter toute manœuvre accidentelle.



Vue d'ensemble de l'unité de contrôle anti-blackout : le circuit imprimé auxiliaire contenant l'afficheur LCD doit être raccordé au circuit imprimé principal à l'aide d'un câble plat sous forme de nappe. L'ensemble doit être inséré dans un boîtier rail DIN correctement découpé.

De même le connecteur « CONN » sera relié au circuit principal à l'aide d'une nappe de fil.

Une fois l'assemblage terminé, vérifiez qu'il n'y a pas de court-circuit entre les lignes de la phase et du neutre. Mesurez la résistance entre ces deux pistes avec un multimètre. L'instrument doit indiquer la présence de l'enroulement primaire du transformateur, qui doit avoir une résistance d'environ 1,4 kΩ.

Si vous trouvez une résistance de quelques ohms alors il y a un danger, revérifiez intégralement le montage et **surtout ne tentez pas de le connecter** au réseau électrique 220 VAC.

Avant d'insérer les circuit intégrés U4 et U6, vous devez vérifier les tensions (en se référant aux masses correspondantes) des régulateurs qui doivent avoir une valeur de 5 V. Pour cela vous devez relier le montage au réseau en évitant de le toucher avec les mains, car une partie du circuit imprimé n'est pas isolée du secteur. Il y a un risque d'électrocution.

Mesurez soigneusement à l'aide d'un multimètre la tension continue entre les broches 20 (+) et 8 (-) de U6 et entre la sortie et la masse du régulateur U1.

Si tout est correct, débranchez le montage et insérez le microcontrôleur sur son support.

À l'aide d'un programmateur approprié via le connecteur ICSP (la broche « 1 » qui est Vpp est orientée vers le quartz Q1), programmez le microcontrôleur.

Selon le programmateur dont vous disposez, vous pouvez programmer le PIC puis l'insérer ensuite sur son support en respectant l'orientation.

La résistance **R1** est un shunt de 0,001 Ω. Si vous ne pouvez pas la trouver dans le commerce, vous pouvez la fabriquer en pliant en forme de « U » un morceau de fil de cuivre nu de **0,6 mm²** et de **4 cm de longueur**.

Si vous avez l'intention d'utiliser l'anti-blackout dans des systèmes de 6 kW, vous devrez utiliser une résistance de shunt d'une valeur d'environ 0,75 mΩ.

Dans ce cas, il est nécessaire de porter la broche G0 de U3 à 5 V, cela double le gain du PGA interne (G = 16).

De plus, il est judicieux de vérifier la valeur résistive obtenue. Pour cela, connectez une prise volante sur la sortie (ACOUT) avec une charge électrique qui absorbe 10 A.

En série avec l'un des câbles de l'alimentation AC, insérez un multimètre calibré pour la mesure de courants alternatifs avec une pleine échelle d'au moins 10 A.

Avec un second multimètre mesurez la tension entre le nœud R3-R1 et la masse de U3, après avoir connecté le dispositif au réseau.

Lisez la tension et le courant en même temps et vérifiez que leur rapport ne soit pas supérieur à 0,8 mΩ. Si cela n'est pas le cas, déconnectez tout et renforcez encore plus la piste affectée.

Pour conclure, configurez les cavaliers comme indiqué dans le Tableau 1, le cavalier « JUMP » n'est pas utilisé.

À ce stade, vous devez insérez le circuit de l'anti-blackout dans un boîtier en plastique.

N'oubliez pas de faire la découpe du boîtier pour les boutons, les LED et l'afficheur LCD.

Etalonnage et tests généraux

À ce stade, il sera nécessaire d'étalonner le système afin que la mesure réelle de la consommation instantanée apparaisse sur l'afficheur. Pour cela, vous pouvez mesurer la fréquence du signal « HFOUT » avec un fréquencemètre et calibrer les potentiomètres pour que chaque watt corresponde à 0,5 Hz.

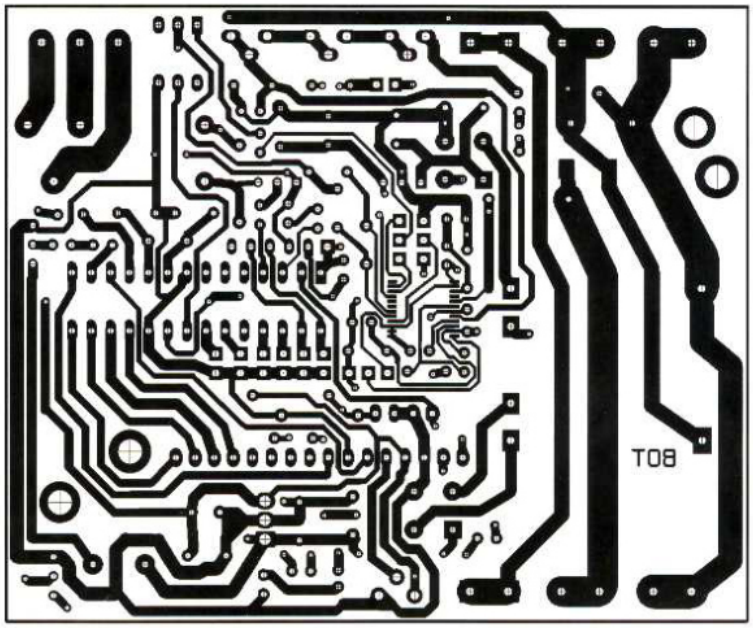
Cependant, nous vous **conseillons d'effectuer directement des mesures de puissances et d'ajuster les trimmers** jusqu'à ce que la même valeur s'affiche sur l'écran LCD.

L'idéal serait d'utiliser un wattmètre à courant alternatif monophasé, mais il est également possible de recourir à deux instruments distincts : un voltmètre et un ampèremètre. En multipliant les valeurs mesurées, nous pouvons obtenir la puissance réelle.

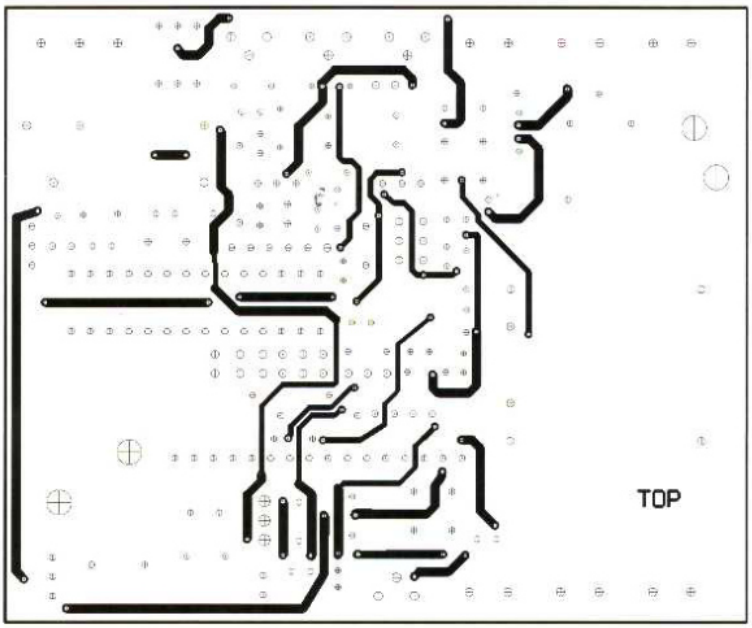
Connectez un ampèremètre calibré pour les courants alternatifs avec une pleine échelle d'au moins 20 A en série avec la sortie (OUT), et un voltmètre calibré pour les tensions alternatives sur l'entrée (IN).

De droite à gauche (vue de dessus), les trimmers permettent un réglage grossier, moyen et fin. Mettez le montage sous tension et attendez que le message de bienvenue s'affiche.

Plan de montage du circuit principal de l'anti-blackout



Circuit imprimé à l'échelle 1 : 1 côté soudures du circuit principal de l'anti-blackout.



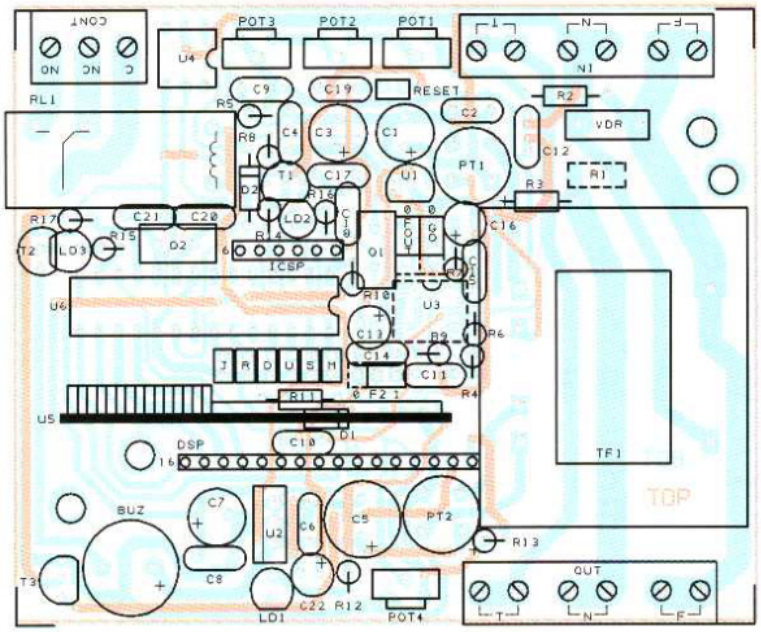
Circuit imprimé à l'échelle 1 : 1 côté composants du circuit principal de l'anti-blackout.

Liste des composants

R1.....0,001 Ω 1W
R2.....560 kΩ
R3.....1 kΩ
R4.....1 kΩ
R5.....1 kΩ
R6.....1 kΩ
R7.....10 Ω
R8.....820 Ω
R9.....1 kΩ
R10....10 kΩ
R11....82 Ω
R12....68 Ω
R13....22 Ω
R14....10 kΩ
R15....220 Ω
R16....220 Ω
R17....10 kΩ

POT1..trimmer 1 MΩ
POT2..trimmer 470 kΩ
POT3..trimmer 47 kΩ
POT4..trimmer 4,7 kΩ

C1..... 100 µF/25 V électrolytique
C2..... 33 nF multicouche
C3..... 100 µF/25 V électrolytique
C4..... 100 nF multicouche
C5..... 470 µF/25 V électrolytique
C6..... 33 nF multicouche
C7..... 100 µF/25 V électrolytique
C8..... 100 nF multicouche
C9..... 33 nF multicouche
C10.... 33 nF multicouche
C11.... 33 nF multicouche
C12.... 33 nF multicouche
C13.... 10 µF/25 V électrolytique
C14.... 100 nF multicouche
C15.... 100 nF multicouche
C16.... 47 µF/25 V électrolytique
C17.... 22 pF céramique
C18.... 22 pF céramique
C19.... 10 nF multicouche
C20.... 22 pF céramique



Plan de câblage des composants du circuit principal de l'anti-blackout.

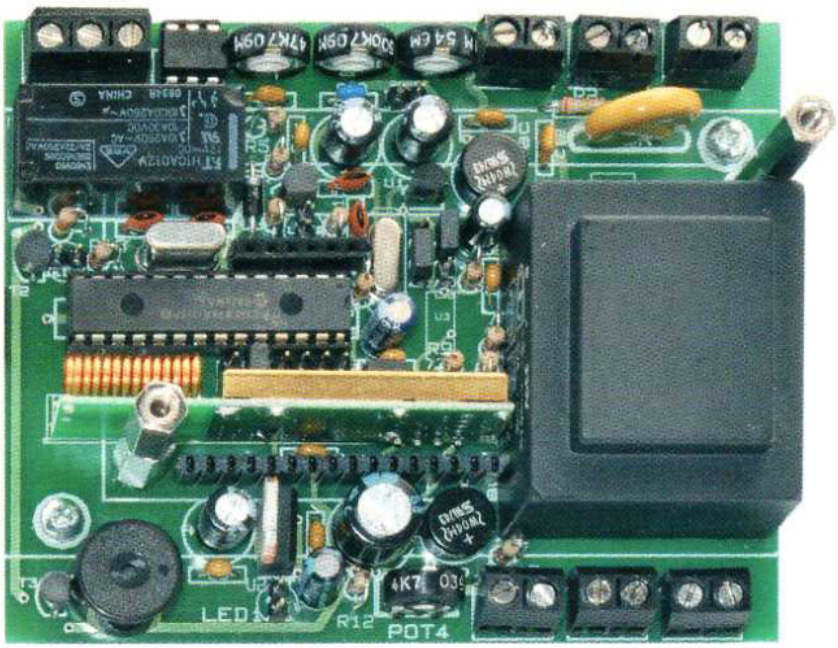


Photo de l'un de nos prototypes du circuit principal de l'anti-blackout.

C21.... 22 pF céramique
C22.... 47 µF/6,3 V électrolytique

D1..... zener 3 V / 1 W
D2..... 1N4007

PT1 .... pont de diode W06M
PT2 .... pont de diode W06M

T1 ..... BC547
T2 ..... BC547

U1..... 78L05
U2..... 7805
U3..... MCP3905A

U4..... 4N35
U5..... TX-4M10HA
U6..... PIC16F876A-I/SP
Q1..... quartz 3,579545 MHz
Q2..... quartz 16 MHz

DSP.... afficheur 16 x 2 (CDL4162)
TF1 .... transformateur 230 V/2 x 9 VAC / 3 VA
RL1 relais 12 V/10 A type H1CA012V
VDR varistance 275 VCA
BUZ buzzer 12 V pour circuit imprimé avec oscillateur

Divers

Support circuit intégré 2 x 14 broches
Support circuit intégré 2 x 3 broches
Bornier 2 pôles (x 6)
Bornier 3 pôles
Barrette mâle 2 pôles (x 7)
Barrette mâle 3 pôles (x 3)
Barrette mâle 4 pôles
Barrette mâle 16 pôles
Barrette femelle 6 pôles
Barrette femelle 16 pôles (x 2)
Boîtier 6 modules rail DIN ou autre

Appuyez alors sur le bouton « MENU » pour commencer la première mesure.

Maintenant, connectez une charge d'environ 2000 W et attendez que la LED verte atteigne le deuxième clignotement, afin d'afficher sur l'écran LCD une mesure correcte.

Calculez la puissance par rapport aux valeurs mesurées par les deux instruments et commencez à tourner le potentiomètre de droite (POT1) pour un premier réglage grossier (en tournant de droite à gauche cela augmentera la valeur sur l'afficheur), et aussi le potentiomètre central (POT2).

Une fois que les deux valeurs sont égales aux unités et aux dixièmes de kW (c'est-à-dire les chiffres immédiatement à gauche et à droite de la virgule), connectez une charge électrique supplémentaire de 1800 à 2000 W.

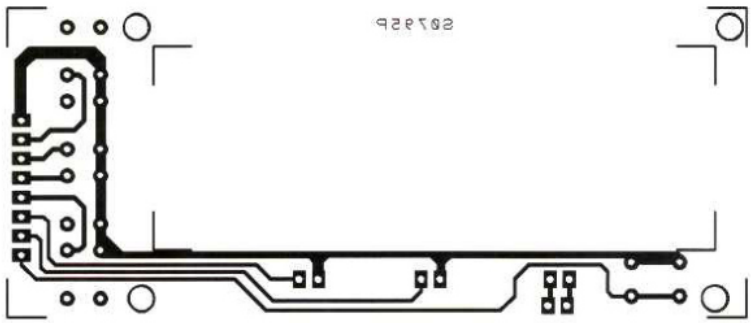
De cette manière, il est possible de dépasser la puissance maximale autorisée par le disjoncteur du circuit sur lequel vous avez connecté l'anti-blackout.

Par conséquent, soyez rapide au niveau de l'étalonnage avec cette mesure supplémentaire. Il s'agit d'ajuster le potentiomètre de gauche (POT3) afin d'obtenir un réglage précis des mesures.

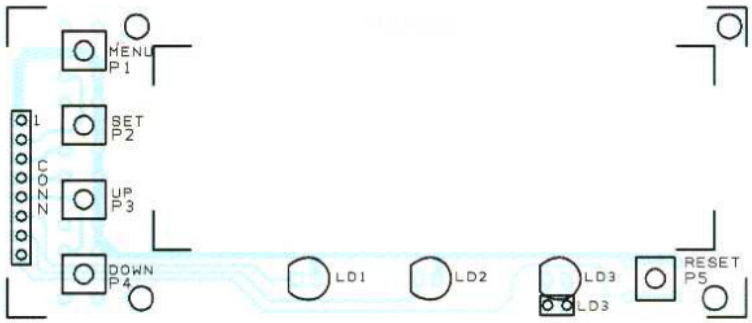
Une fois cela réalisé, l'unité de contrôle anti-blackout est prête à être installée.

Les pistes de cuivre du circuit imprimé, qui vont du bornier d'entrée à celui de sortie vers le réseau domestique, doivent être étamées pour augmenter la section et permettre le passage du courant attendu.

Plan de montage de la carte afficheur



Circuit imprimé à l'échelle 1 : 1 coté soudures de la carte afficheur. Il s'agit d'un circuit comportant une seule face.



Plan de câblage des composants de la carte afficheur.

Installation et mise en service

Maintenant, éteignez le montage, puis insérez l'unité de contrôle anti-blackout dans un rail DIN du panneau électrique de votre maison. L'entrée (IN) est reliée en aval du coupe-circuit.

Quant aux bornes de sorties (OUT), elles sont reliées au reste du circuit électrique.

Si vous souhaitez que l'unité de contrôle anti-blackout isole le système électrique en cas de surcharge, connectez le bornier du relais à un contacteur sur rail DIN que vous devez insérer entre la sortie de l'anti-blackout et le circuit sur lequel sont reliés les appareils à commander.

Ne pas utiliser directement le contact du relais RL1, il doit obligatoirement commander la bobine d'un relais de puissance.

Après avoir soigneusement vérifié tout le câblage, remettez l'interrupteur sous tension. Le message de bienvenue apparaît sur l'afficheur et disparaît au bout de 20 secondes environ, sauf si vous appuyez sur le bouton MENU.

Ensuite, toutes les deux secondes, la mesure de la puissance active consommée par les appareils connectés au système est mise à jour et affichée. Notez que lorsque le système est occupé à gérer une surcharge, le menu n'est pas disponible.

Les commandes du « MENU »

Pour accéder aux commandes du « MENU », maintenez le bouton « MENU » enfoncé puis appuyez sur le bouton « DOWN » (bas). Une fois que les deux premières commandes s'affichent, il est possible de se déplacer

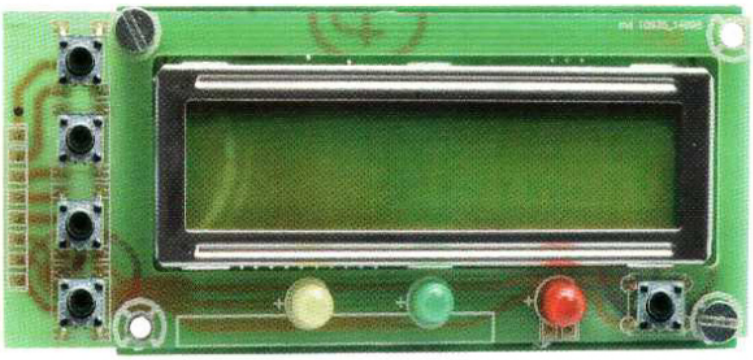


Photo de l'un de nos prototypes de la carte afficheur.

Liste des composants

LD1.... LED 5 mm jaune
LD2.... LED 5 mm verte
LD3.... LED 5 mm rouge

P1..... micro switch hauteur 3,5 mm
P2..... micro switch hauteur 3,5 mm
P3..... micro switch hauteur 3,5 mm
P4..... micro switch hauteur 3,5 mm
P5..... micro switch hauteur 1 mm

Divers :

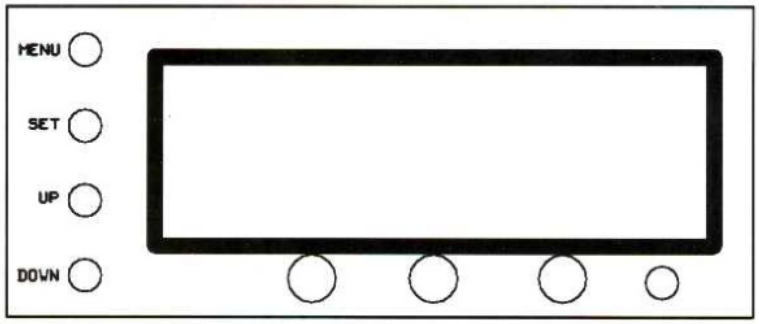
Support de LED 5 mm pour face avant (x 3)
Câble en nappe longueur 15 cm 8 pôles
Câble en nappe longueur 17 cm 2 pôles
Barrette femelle 8 pôles
Barrette femelle 1 pôles (x 4)
Barrette femelle 4 pôles
Barrette femelle 2 pôles (x 4)
Barrette mâle 6 pôles
Entretoise M/F 15 mm (x 4)
Ecrou 3 MA (x 8)
Vis 20 mm 3 MA (x 4)

vers le haut (avec le bouton « UP ») ou vers le bas (avec le bouton « DOWN »).

Pour sélectionner une commande, appuyez sur la touche « SET », puis relâchez la touche « MENU ».

La libération de cette dernière, lors du défilement des commandes, permet de quitter le menu et de démarrer le cycle des mesures de la puissance instantanée. Les commandes sont les suivantes :

- « **1. Puissance MAX** » : indication de la valeur maximale parmi toutes les mesures de puissances effectuées. En accédant à cette commande, il vous sera demandé d'appuyer sur UP/DOWN, pour respectivement soit effectuer un reset de la valeur actuelle (afin de commencer une nouvelle période de surveillance), soit lire (« Read ») la valeur enregistrée jusqu'à ce moment ;
- « **2. Puissance moyenne** » : calcul de la puissance moyenne pendant une période d'une minute. À la fin, la puissance moyenne est affichée ;
- « **3. Ins.Pot.Max.** » : insertion de la puissance maximale autorisée en sortie avant l'intervention de limitation du système. Pour changer cette valeur qui va d'un minimum de 400 W à un maximum de 6000 W, appuyez plusieurs fois sur « UP » (pour augmenter) ou « DOWN » (pour diminuer) et pour confirmer appuyez sur la touche « SET ». L'augmentation ou la diminution s'effectue par pas de 50 W ;
- « **4. Priorité MAX** » : insertion du niveau de priorité le plus élevé des récepteurs. La commande prévoit l'insertion d'une nouvelle valeur d'un minimum de 1 à un maximum de 6 en appuyant sur la touche « UP ». La lecture de la valeur actuelle s'effectue en appuyant sur la touche « DOWN ». La modification de la valeur est effectuée en appuyant plusieurs fois sur la touche « UP » jusqu'à la valeur désirée, puis en confirmant avec la touche « SET » ;
- « **5. Force ON liv.** » : permet l'activation forcée de tous les modules actionneurs, et donc des appareils électriques qui y sont connectés. Les « Smart-Rx » sont d'abord activés via un code spécial, suivi de tous les récepteurs Motorola en fonction de leur niveau de priorité (à condition que le cavalier de configuration « RXT » soit fermé) ;
- « **6. Force OFF liv.** » : permet la désactivation forcée de tous les modules actionneurs. Les « Smart-Rx » sont d'abord désactivés via un code spécial, suivi de tous les récepteurs Motorola en fonction de leur niveau de priorité (à condition que le cavalier de configuration « RXT » soit fermé) ;
- « **7. NewCod.Base** » : permet la modification du code de base Motorola (6 bits MSB) émis par l'anti-blackout. En appuyant sur le bouton « UP », vous pouvez personnaliser bit par bit le code de base (Zero, One, Open) avec un maximum de 729 combinaisons possibles (SET pour confirmer). En appuyant sur « DOWN », il est possible de modifier le niveau de priorité émis immédiatement après le code de base, chaque fois que la commande « 12.Calibration HF » est exécutée. La procédure se termine par la modification automatique du code de base des « Smart-RX » (les autres doivent être codés manuellement) ;
- « **8. Vis.Cod.Base** » : permet l'affichage du code de base (Zero, One, Open) ;
- « **9.Vis.liv.prior** » : permet la visualisation directe des niveaux de priorité des « Smart-RX » (et sur chacun d'eux) grâce à leur LED verte. Une série de clignotements rapides correspond à la priorité « 0 », sinon « n » clignotements correspondent à une priorité de niveau « n ». Cette visualisation se répète 5 fois avec un intervalle de 4 secondes. Par exemple : 2 clignotements → niveau de priorité = 2 ;



Gabarit du panneau de commande dont les dimensions sont de 100 mm x 41 mm.

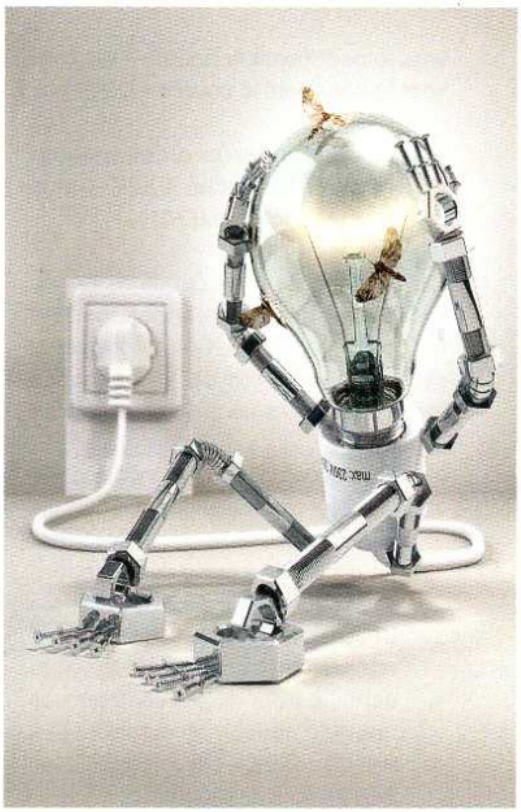
- « **10.BuzzerON/OFF** » : permet l'activation/désactivation du buzzer en cas de signalement de la présence d'une surcharge. Il est nécessaire de modifier l'état ON/OFF à l'aide du bouton « UP », puis de confirmer avec « SET » ;
- « **11.ResetSystem** » : permet de réinitialiser le système en restaurant les valeurs par défaut. Vous devez simultanément appuyer sur « UP » et « DOWN » pendant 5 secondes, la LED verte clignote toutes les 0,5 secondes. Sinon, la séquence de réinitialisation n'est pas exécutée. Les valeurs par défaut sont :
  - Code de base : **H H H H H H** ;
  - Niveau de priorité pour le codage « Rx » : **0** ;
  - Buzzer : **ON** ;
  - Puissance MAX : **6 000 W** ;
  - Niveau de priorité le plus élevé : **6** ;
- « **12.Calibration HF** » : permet l'émission continue d'un code Motorola constitué du code de base et du code de priorité défini par la commande 7. Cette commande permet le codage des « Smart-Rx » (auto-apprentissage du code de base et du niveau de priorité) et le test de réception du signal HF. La procédure se termine en maintenant le bouton « SET » enfoncé jusqu'à ce que la LED jaune s'éteigne pendant environ une seconde. Immédiatement après, un second code est émis pendant quelques secondes, capable de désactiver les récepteurs activés précédemment et ayant le même niveau de priorité. Une fois qu'un module « Smart-Rx » (ou un récepteur Motorola) a été codé, il peut être installé en le reliant à une prise électrique contrôlée par le système anti-blackout. Cette commande permet également de modifier la fréquence d'émission du code Motorola, elle peut être réglée à l'aide des touches « UP » et « DOWN ». Ce réglage n'est pas forcément nécessaire, mais il peut être utile pour maintenir une compatibilité de fonctionnement avec tous les récepteurs classiques Motorola, c'est-à-dire sans un décodeur à base de PIC.

Configuration initiale

Le système a besoin de réglages initiaux pour intervenir au bon moment. La séquence recommandée est la suivante :

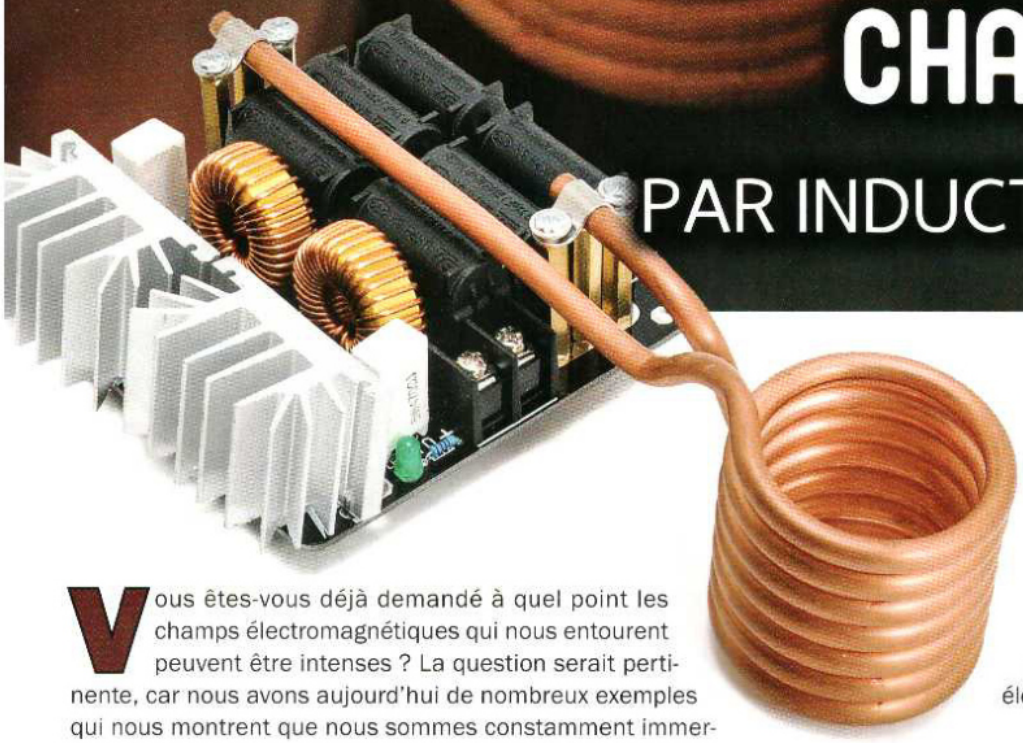
1. Entrez la puissance maximale autorisée à l'aide de la commande « n° 3 », en entrant le seuil de puissance pour lequel le système interviendra lors d'une surcharge. Il désactivera les modules actionneurs d'un ou plusieurs niveaux de priorité. Si, par exemple, la puissance électrique de l'installation est limitée à 3 kW, cette valeur peut être utilisée comme limite pour l'unité de contrôle anti-blackout. La valeur par défaut est 6 kW ;
2. Entrez le niveau de priorité le plus élevé en utilisant la commande « n° 4 », cela correspond au niveau de priorité le plus élevé (de 1 à 6) géré par l'anti-blackout. Le nombre maximal de niveaux est de 7, mais ce nombre peut être réduit en fonction de vos besoins. Cela permet d'éviter que le système ne perde du temps à désactiver des niveaux inutilisés. La valeur par défaut est 6 ;
3. Saisie du code de base : tous les codes de commande émis par l'unité de contrôle sont de type Motorola avec un codage à 3 états sur 9 bits. Les 6 premiers bits sont réservés au code de base, c'est-à-dire l'identification du système anti-blackout. Pour le modifier, utilisez la commande « n° 7 ». La valeur par défaut est « H H H H H H » ;
4. Activation du buzzer : elle s'effectue à l'aide de la commande « n° 10 ». Il est possible d'activer ou de désactiver le buzzer. Si elle est sur « ON » (valeur par défaut), le buzzer signale la condition de surcharge lorsque la désactivation de tous les niveaux de priorité ne ramène pas la consommation en dessous de la puissance maximale autorisée.

Dans le prochain numéro, nous aborderons la description des modules actionneurs (récepteurs.) ■



Nous portons à fusion des métaux grâce à la technologie de commutation au zéro de tension ou ZVS (Zero Voltage Switching), appliquée dans notre projet à un circuit résonnant RLC d'une puissance de 1 kW. La technologie ZVS permet une régulation de la tension à l'aide d'une « commutation douce », évitant ainsi les pertes de commutation qui se produisent lors du fonctionnement en commutation classique.

CHAUFFAGE PAR INDUCTION 1 KW



de Pier Alessandro AISA

Vous êtes-vous déjà demandé à quel point les champs électromagnétiques qui nous entourent peuvent être intenses ? La question serait pertinente, car nous avons aujourd'hui de nombreux exemples qui nous montrent que nous sommes constamment immergés dans une myriade d'ondes électromagnétiques, due par exemple aux câbles de distribution électrique, aux émetteurs radio FM, aux signaux TNT de la télévision, au réseau GSM des mobiles, au Wi-Fi des box, etc.

Placez simplement un téléphone portable à côté d'un haut-parleur, vous entendrez des grésillements; ou alors prenez un tube néon dans votre main et approchez-le dans l'obscurité sous une ligne à haute tension, vous remarquerez que le néon s'allume alors qu'il n'est pas relié à un circuit électrique !

Ces ondes invisibles pour l'homme sont en réalité très présentes et même envahissantes dans notre vie quotidienne, bien que leur présence devienne dangereuse dans certains cas en fonction de la puissance rayonnée et de la distance de la source à laquelle nous nous trouvons.

Mais à quel point un champ électromagnétique peut-il être puissant ? Le four à micro-ondes, présent dans beaucoup

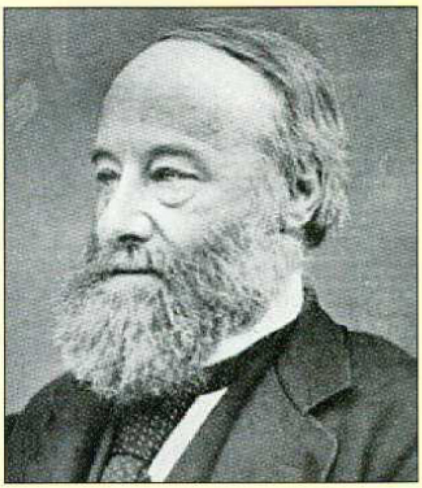
Nos **inspirateurs**

Michael Faraday (1791-1867)

C'était un physicien et chimiste britannique qui a été responsable de la découverte du phénomène d'induction électromagnétique, en 1831. Connu pour ses travaux fondamentaux dans le domaine de l'électromagnétisme, l'électrochimie, le diamagnétisme, et l'électrolyse, il donna son nom à de multiples lois et phénomènes, notamment la loi de Faraday (ou Lenz-Faraday) pour l'induction électromagnétique, les lois de Faraday en électrochimie, l'effet Faraday, ou encore à des dispositifs expérimentaux comme la cage de Faraday. Le farad, unité de capacité électrique, est également nommé en son honneur.

Ses découvertes sont actuellement la base de l'exploitation des moteurs électriques, alternateurs, transformateurs, haut-parleurs, microphones, etc.

La découverte de l'électromagnétisme a eu lieu lorsqu'il enroula deux bobines isolées, constituées de fil électrique, autour d'une grande bague en acier, fixée à une table. Il s'aperçut qu'en faisant passer du courant à travers l'une des bobines, un courant et donc une tension était induite momentanément dans l'autre bobine. Dans d'autres expériences, il découvrit qu'en déplaçant un aimant à travers une boucle de fil, un courant électrique circulait dans le fil et également qu'un courant circulait en déplaçant un solénoïde le long d'un aimant stationnaire.



La **loi de Faraday-Neumann-Lenz** devint par la suite l'une des **4 équations de Maxwell régissant les champs électromagnétiques**.

James Prescott Joule (1818-1889)

Il était un physicien anglais et a donné son nom à « l'**effet Joule** ». En 1848, en étudiant la nature de la chaleur, il a démontré l'existence d'une relation entre le courant circulant dans une résistance et la chaleur dissipée, plus précisément il trouva qu'il existait une proportionnalité entre le courant circulant dans un circuit électrique et la chaleur qui en émanait. Son étude sur la nature de la chaleur et sa découverte de la relation avec le travail mécanique l'ont conduit à la théorie de la conservation de l'énergie (première loi de la thermodynamique).

de cuisines, peut chauffer des aliments et faire apparaître des arcs électriques entre les extrémités d'objets métalliques insérés par inadvertance.

Mais un champ électromagnétique pourrait-il être assez puissant pour amener un conducteur électrique à l'incandescence en quelques secondes ?

La réponse est oui et le projet que nous vous présentons dans cet article le démontre clairement. Il s'agit en fait d'un système qui fonctionne en exploitant trois phénomènes physiques que nous avons étudiés à l'école, du moins dans la filière scientifique.

Ces 3 phénomènes sont :

- le principe de l'**induction magnétique** ;
- le principe des **courants de Foucault** ;
- l'**effet Joule**.

Notre projet montre comment échauffer et éventuellement porter à fusion **un matériau électriquement conducteur** et notamment les matériaux ferromagnétiques, grâce à l'utilisation d'un circuit de type « **commutation au zéro de tension** » (ZVS) d'une puissance nominale de **1000 W**, pouvant aller jusqu'à 1500 W dans certaines conditions.

ZVS est l'acronyme de « **Zero Voltage Switching** », c'est une technique utilisée dans les convertisseurs électroniques de puissance pour augmenter l'efficacité, car **la commutation des transistors de puissance s'effectue à une tension presque nulle à leurs broches**.

Par conséquent, la puissance représentée par le produit **V * I** diminue et, **donc les pertes par commutation sont très faibles**, comme nous le décrirons plus en détail dans le paragraphe consacré au principe de fonctionnement.

Le même concept de base de fonctionnement des plaques de cuisson à induction, qui sont largement utilisées dans les pays où l'électricité est bon marché, est appliqué à ce montage.

Une table de cuisson à induction est constituée par une bobine dans laquelle circule un très fort courant électrique alternatif ou en tout cas

Jean Bernard Léon Foucault (1819-1868)

C'était un physicien et astronome français, principalement connu pour l'invention du « **pendule de Foucault** ». Mais il fut aussi responsable de la découverte du phénomène des courants parasites en 1851, quand il s'aperçut que les courants parasites étaient causés par la variation du champ magnétique qui traverse un conducteur et que le mouvement relatif engendre la circulation des électrons à l'intérieur du conducteur. Les électrons, qui se déplacent sous la forme de vortex, génèrent à leur tour un champ magnétique qui s'oppose (c'est-à-dire ayant une direction opposée ou un sens opposé) à la variation du champ magnétique dû à la loi de Lenz, puis par effet Joule le conducteur est chauffé.

variable au cours du temps, ce qui produit un champ magnétique proportionnel au courant qui le génère.

Selon la **loi de Faraday**, une **variation du flux du champ magnétique dans le temps produit une force électromotrice induite dans tout corps électriquement conducteur**, ce dernier étant affecté par les lignes de champ résultantes.

Cette force électromotrice engendre des courants électriques induits qui circulent dans la matière constituant les ustensiles posés sur la plaque (attention il faut des ustensiles adaptés aux plaques à induction). Ces **courants induits** sont appelés « **courants de Foucault** » ou « **courants d'eddy** » (« eddy » en anglais signifie « vortex », en raison de la forme caractéristique que prend le courant à l'intérieur du conducteur). Ces courants parasites sont la cause de l'effet Joule, c'est-à-dire la production de chaleur due à la perte d'énergie qui en résulte.

Dans le cas de la plaque à induction, la dissipation de l'énergie sous forme de chaleur provoque l'échauffement de l'ustensile, comme cela arrive en mécanique avec les frottements. Ces derniers constituent une force résistante qui se traduit par la dissipation de l'énergie cinétique sous forme de chaleur.

Dans tous les cas, la cuisson à induction est un cas particulier, car l'effet des courants induits par les champs électromagnétiques diffère selon la perméabilité magnétique des matériaux qui y sont immergés, leur réluctance magnétique (la réluctance est l'aptitude d'un circuit magnétique à s'opposer à sa pénétration par un champ magnétique) et leur conductivité électrique.

Cela explique pourquoi dans un four à micro-ondes, il n'est pas possible d'introduire un récipient métallique (encore moins en acier).

Par exemple, pour cuire de la viande, il faut la mettre dans un plat en verre, alors que pour une table à induction la viande doit être obligatoirement placée dans un ustensile métallique pour cuire.

En effet, dans le premier cas la chaleur se développe à l'intérieur de la nourriture alors que dans le second cas la chaleur apparaît dans le fond de l'ustensile, car certains matériaux chauffent déjà aux basses fréquences alors que d'autres nécessitent des micro-ondes (très hautes fréquences).

Généralement en électricité et en électronique, l'effet des courants de Foucault doit être évité, car il entraîne des pertes. Par exemple, les pertes dans le noyau d'un transformateur.

Dans notre cas, au contraire, nous allons utiliser les courants de Foucault pour générer de la chaleur dans un matériau exposé à un champ magnétique. Spécifiquement, nous parlons de matériaux ferromagnétiques, qui chauffent fortement à la fréquence de fonctionnement du montage.

En plus du chauffage et de la fusion possible des métaux, le circuit peut trouver d'autres applications intéressantes, telles que la transmission d'énergie dans le milieu ambiant en utilisant des bobines couplées et accordées à la fréquence de résonance (transmission de la puissance sans fil ou « wireless power ») ou les circuits de décharge pour les bobines Tesla.

Le schéma électrique

En observant le schéma électrique, nous pouvons constater dans le circuit de commande une structure symétrique à deux branches, appelée **oscillateur de Royer**, qui permet **l'auto-oscillation d'un bloc RLC** à sa **fréquence de résonance naturelle**.

Un **oscillateur de Royer est un oscillateur à relaxation électronique qui utilise un transformateur à noyau saturable**. Il a été inventé et breveté en 1954 par George H. Royer. Il présente les avantages de la simplicité, du faible nombre de composants, des formes d'ondes rectangulaires et de l'isolation facile du transformateur.

En utilisant au maximum le noyau du transformateur, il minimise la taille et le poids de ce dernier.

Le circuit classique de Royer produit des ondes carrées, une version modifiée en ajoutant un condensateur le transforme en oscillateur harmonique produisant ainsi des ondes sinusoïdales. Les deux versions sont largement utilisées, principalement dans les convertisseurs de puissance.

Le bloc RLC est constitué pour la partie « **L** » (inductance) d'une bobine communément appelée « **work-coil** ou **bobine de travail**, pour la partie « **C** » (capacité) communément appelée « **tank-condensateur** » (condensateur de stockage) de différents condensateurs placés en parallèle et pour la partie « **R** » (résistance) des résistances séries introduites par les composants et les connexions.

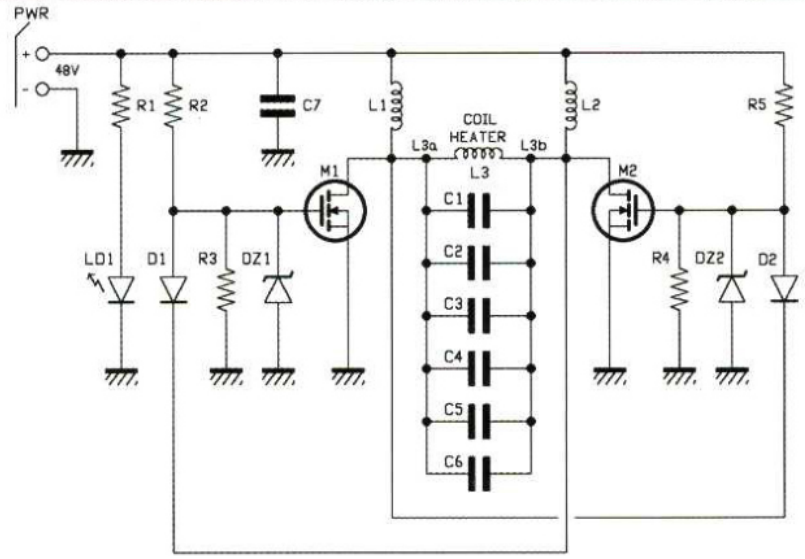


Schéma électrique de notre chauffage par induction.

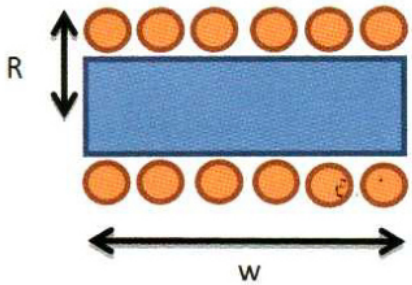


Figure 1 : la bobine est composée de spires espacées de fil de cuivre.

Figure 2 : la bobine a un rayon R de 2,5 cm et une longueur « w » de 7 cm.

Une fois alimenté, l'oscillateur entre en résonance et grâce au courant élevé d'extra-résonance circulant dans la bobine de travail, il est capable de créer un champ magnétique intense.

Lorsque l'oscillateur de Royer est alimenté, bien que les deux branches soient symétriques, l'un des deux MOSFET M1 ou M2 deviendra conducteur en premier car les deux MOSFET n'ont pas des caractéristiques parfaitement égales.

Supposons que le **transistor M1 entre en premier en conduction**, son **drain se retrouve au potentiel de la masse**, ce qui a pour effet de forcer le **blocage** du transistor **M2**.

Cependant, grâce à la **diode** de **feedback D1** (diode de retour ou de contre-réaction) qui **entre en conduction**, la charge de la grille de M2 est extraite rapidement. Le circuit résonnant, constitué par l'inductance L3 et les condensateurs en parallèles C1, C2, C3, C4, C5, C6, **génère sur le drain de M2 une demi-onde sinusoïdale** qui passe de la valeur zéro au maximum puis revient à zéro.

Lorsque la **demi-onde revient à zéro**, la diode **D2 conduit**, ce qui force le **blocage de M1** et **M2 se met à conduire**, provoquant la génération de l'autre demi-onde sinusoïdale au niveau du circuit résonant.

À partir de ce moment, le cycle se répète avec la fréquence de résonance du bloc RLC et sur la bobine de travail L3, nous obtenons un fort courant sinusoïdal avec une **fréquence d'environ 100 kHz**. Cette valeur produirait un réchauffement modeste pour de la nourriture, mais déjà un réchauffement appréciable pour de l'acier. L'échauffement peut être augmenté avec une puissance électrique plus importante et donc une induction électromagnétique plus grande.

Dans l'oscillateur, l**es MOSFET travaillent en opposition de phase**, c'est à dire lorsque M1 conduit M2 est bloqué et vice versa. Cela est garanti par la présence du circuit résonnant et des diodes de contre-réaction D1 et D2.

S'il y avait une conduction simultanée des deux transistors MOSFET, il n'y aurait pas de limitation du courant de court-circuit et donc il y aurait destruction des MOSFET en raison du courant excessif qui apparaîtrait.

La commutation des MOSFET s'effectue avec une tension entre le drain et la source (Vds) **pratiquement nulle**, c'est-à-dire dans la condition du « **Zero Voltage Switching** » (d'où l'acronyme ZVS) et donc **la puissance dissipée lors de la commutation est minimisée** et est égale à :

$$Pd = Vds * Id$$

où Id est le courant de drain.

De plus, grâce à la technique ZVS, les perturbations radiofréquences produites lors de la commutation des MOSFET sont fortement réduites, le circuit perturbe très peu le milieu environnant.

Les MOSFET M1 et M2 sont des **IRFP260N** d'Infineon.

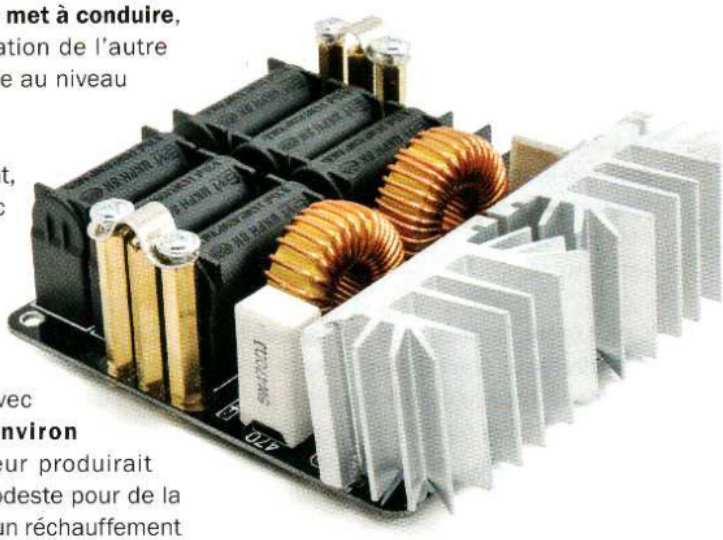


Figure 3 : ici vous pouvez voir la carte montée.

Figure 4 : les entretoises permettent la fixation de la bobine.

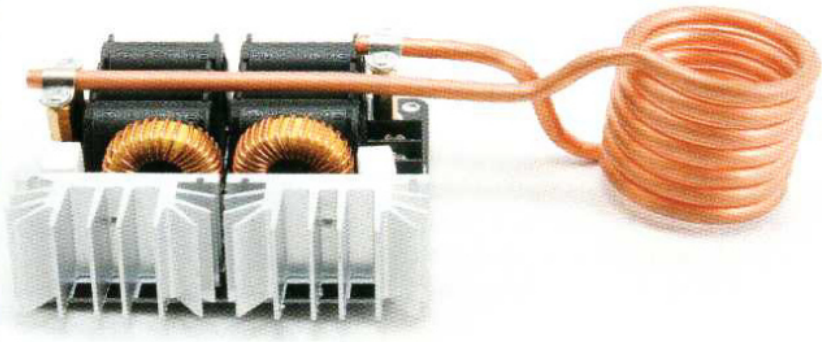


Figure 5 : La bobine de travail (work-coil) montée sur le circuit.

Ils sont caractérisés par une « Rdson » très faible (seulement 40 mΩ) et un courant de drain élevé (Id = 50 A). Ce sont des MOSFET de type « Fast Switching », c'est-à-dire à commutation rapide. Ils passent de l'état bloqué à l'état de conduction complète très rapidement et cela s'applique également aux diodes de protection intégrées placées en antiparallèles, pour lesquelles le temps de recouvrement inverse (TRR) doit être de l'ordre de 400 ns.

En effet, lorsqu'une diode est parcourue par un courant direct et qu'elle est brusquement soumise à une tension inverse, le blocage de la diode ne se produit pas instantanément. Un courant bref parcourt alors la diode de la cathode vers l'anode pendant une durée appelée temps de recouvrement inverse.

La tension de crête maximale qui se produit dans un circuit en commutation de type ZVS est :

$$V_{crête} = V_{allm} * \pi$$

Avec une tension d'alimentation de 48 V, le paramètre de la tension maximale entre le drain et la source (Vdss) a été choisi égal à 200 V.

Les diodes de retour D1 et D2 sont des MUR420 et doivent également être de type à commutation rapide (Fast Switching), pour bloquer à temps le MOSFET auquel leur anode est reliée. Elles doivent supporter un courant d'au moins 4 A.

Pour piloter les grilles (gate) des deux MOSFETS, nous avons inséré deux résistances de puissance R2 et R5 d'une valeur de 470 Ω/5 W.

Pour protéger les MOSFET contre les surtensions et les surintensités, il existe deux couples de composants constitués chacun d'une résistance et d'une diodes zener. Il s'agit de R3/DZ1 et R4/DZ2.

Les résistances R3 et R4 ont une valeur de 10 kΩ et les diodes zener sont des modèles 12 V/1 W.

La LED LD1 signale la présence de l'alimentation et est alimentée à travers la résistance R1 de 4,7 kΩ. Son rôle est de limiter le courant circulant dans la LED.

Les inductances L1 et L2 ont une valeur de 100 µH et sont utilisées pour limiter les pics de tension lors des phases de commutation, car sinon les MOSFET pourraient être détruits. Les inductances sont enroulées sur des noyaux toroïdaux avec une capacité de courant maximal de 13 A.

Les condensateurs C1, C2, C3, C4, C5, C6 sont de type polypropylène MKP afin de supporter un courant élevé et une tension de résonance importante et aussi pour limiter les pertes. La capacité de résonance est d'environ 2 µF (c'est la valeur que nous avons choisie), cette capacité est obtenue par la connexion de 6 condensateurs de 0,33 µF en parallèle. Chaque condensateur a une tension de service de 630 V.

La **bobine de travail** ou « work-coil » (voir la figure 1), c'est-à-dire la bobine à l'intérieur de laquelle nous introduirons l'objet à chauffer/fondre, **est constituée de 6 spires de fil de cuivre de 6 mm de diamètre** pouvant supporter des courants élevés, dissiper la chaleur produite et minimiser les pertes dues à l'effet de peau qui à la fréquence de 100 kHz tend à faire circuler le courant dans la couronne circulaire du conducteur et non pas au centre.

Figure 6 : positionnement de l'objet à chauffer à l'intérieur de la bobine de travail.

En effet, l'effet de peau ou effet pelliculaire est un phénomène électromagnétique qui fait qu'à fréquence élevée, le courant circule uniquement sur la surface du conducteur.

Ce phénomène d'origine électromagnétique existe pour tous les conducteurs parcourus par des courants alternatifs. Il en résulte une augmentation de la résistance du conducteur.

Ce phénomène a été mis en évidence par Nikola Tesla (inventeur et ingénieur naturalisé américain d'origine serbe 1856-1943).

Les spires ne doivent pas se toucher car le cuivre est nu, sinon un court-circuit pourrait se créer et modifier la valeur de l'inductance.

La bobine de travail a une inductance théorique d'environ 1,26 µH, mais celle-ci peut varier en fonction de la longueur des connexions horizontales qui la relient aux condensateurs.

Pour calculer la valeur théorique de l'inductance, vous pouvez utiliser la formule suivante (voir la figure 2) :

$L = \mu_0 N^2 \pi R^2 / w$

où μ_0 est la perméabilité magnétique du vide : $\mu_0 = 4 \pi 10^{-7}$ H / m (l'unité est en henry par mètre H m^{-1} ou H / m), N est le nombre de spires de la bobine (dans notre cas 6), R vaut 2,5 * 10^{-2} m et w = 7 * 10^{-2} m (10^{-2} m = 1 cm). Donc l'inductance L vaut 1,26 µH.

Pour calculer la valeur théorique de la fréquence de résonance, vous pouvez utiliser la formule suivante :

$f_R = 1 / (2\pi * \sqrt{(LC)}) = 100{,}3$ kHz

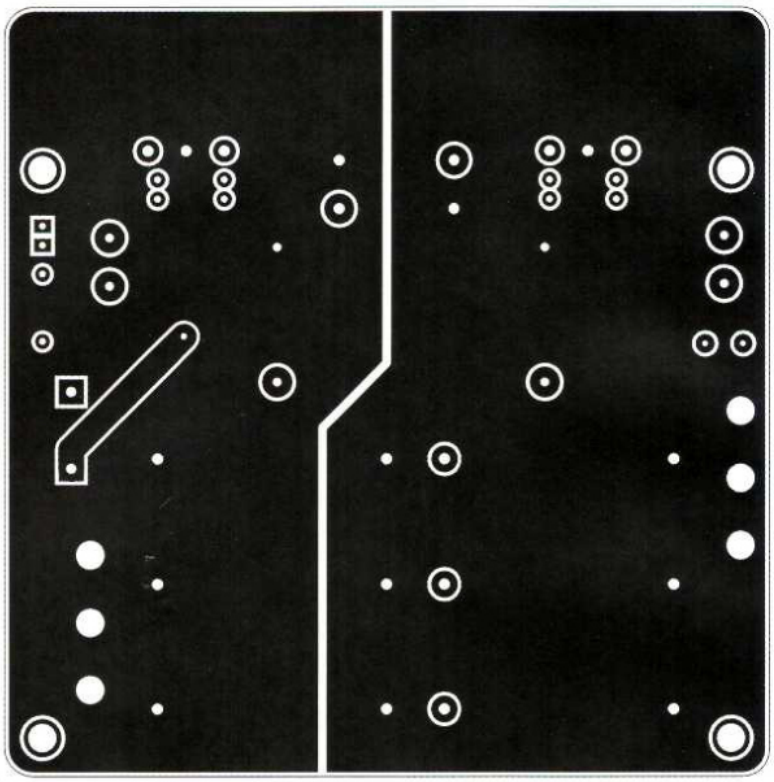
où L = 1,26 µH et C = 2µF.

Réalisation pratique

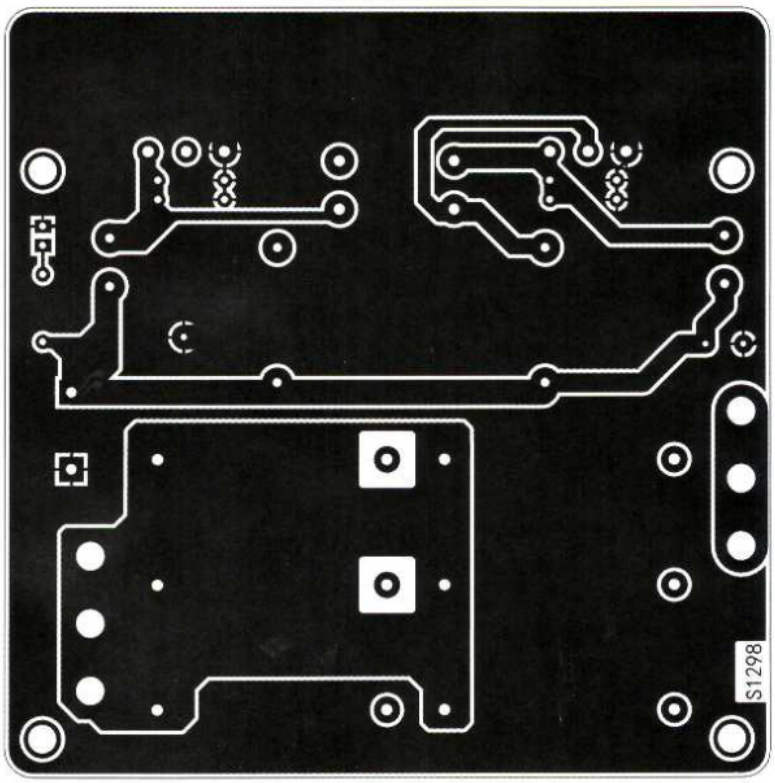
Nous avons fabriqué notre chauffage à induction ZVS sur un circuit imprimé double face.

Vous pourrez télécharger gratuitement sur notre site www.electroniquemagazine.com dans le sommaire détaillé de

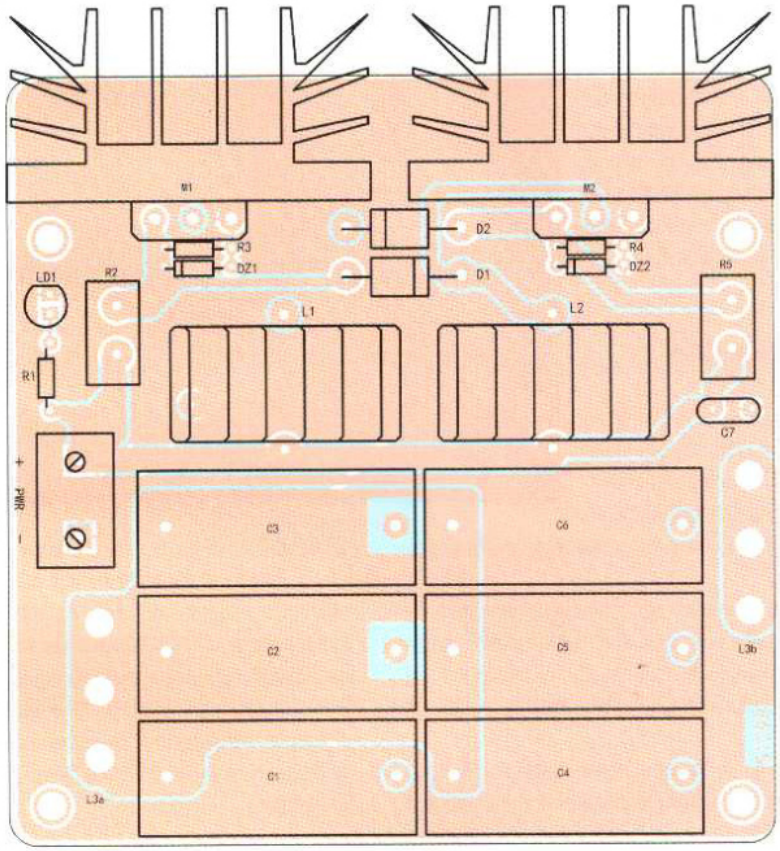
Plan de montage du chauffage par induction



Circuit imprimé à l'échelle 1 : 1 côté soudures (face inférieure) du chauffage par induction.



Circuit imprimé à l'échelle 1 : 1 côté composants (face supérieure) du chauffage par induction.



Plan de câblage des composants du chauffage par induction.

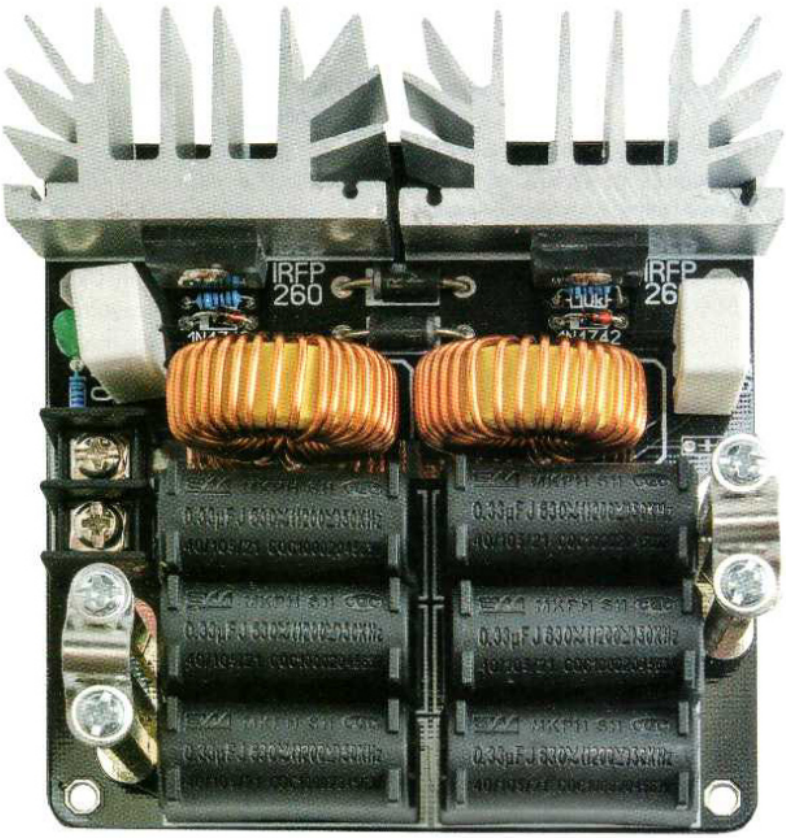


Photo de l'un de nos prototypes du chauffage par induction.

Liste des composants du chauffage par induction

R1..... 4,7 kΩ
R2..... 470 Ω / 5W
R3..... 10 kΩ 1%
R4..... 10 kΩ 1%
R5..... 470 Ω / 5W

C1..... 0,33 µF 630 VAC au pas de 30 mm
C2..... 0,33 µF 630 VAC au pas de 30 mm
C3..... 0,33 µF 630 VAC au pas de 30 mm
C4..... 0,33 µF 630 VAC au pas de 30 mm
C5..... 0,33 µF 630 VAC au pas de 30 mm
C6..... 0,33 µF 630 VAC au pas de 30 mm
C7..... 100 nF céramique

L1 ..... bobine 100 µH
L2 ..... bobine 100 µH
L3 ..... bobine 1,26 µH

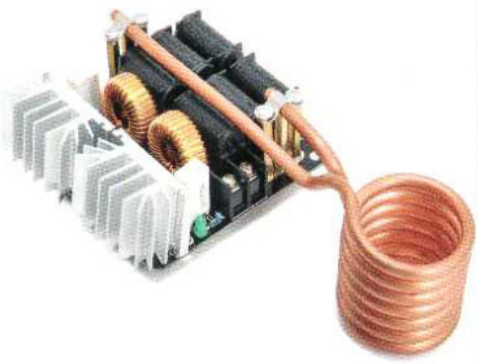
M1 ..... IRFP260N ou équivalent
M2 ..... IRFP260N ou équivalent

D1..... MUR420
D2..... MUR420
DZ1.... 1N4742
DZ2.... 1N4742

LD1.... LED 5 mm verte

Divers

Bornier 2 pôles 10 mm
Vis 20 mm 3 MA (x 2)
Vis 10 mm 4 MA (x 10)
Entretoise F/F 30 mm 4 MA (x 6)
Radiateur 1.25GY-50 (x2) code : 175009 chez www.farnell.fr
Collier pour tuyau de 6 mm (x2)



la revue (descendez tout en bas de la page web pour visualiser les fichiers disponibles) les typons au format « pdf » et les fichiers GERBER pour une fabrication professionnelle.

Les composants à monter sur le circuit imprimé sont de type traversant et sont peu nombreux. Les seuls composants polarisés sont les diodes et les MOSFET, il faudra vérifier leur orientation avant de les souder.

Les MOSFET doivent être montés chacun sur un dissipateur de 6 °C / W.

Comme d'habitude, commencez par souder les composants ayant un bas profil, c'est à dire les résistances.

Ensuite continuez avec les diodes en prêtant une attention particulière à leur orientation. Pour cela consultez le plan de câblage des composants avant de les souder.

Continuez avec les 2 résistances de puissance R2 et R5, puis la LED dont le méplat doit être orienté vers les MOSFET. Ensuite, soudez le bornier, les condensateurs, les selfs L1 et L2.

Fixez les transistors sur leurs radiateurs avant de les souder, sans bloquer les vis. Cela vous aidera à les positionner correctement avant de les souder sur le circuit imprimé.

Terminez le montage par l'insertion des entretoises. Une fois l'assemblage terminé, la carte sans la bobine de travail devrait ressembler à celle visible en figure 3.

Pour fixer la bobine de travail, nous avons utilisé de chaque côté 3 entretoises hexagonales dorées de 6 mm de diamètre et de 40 mm de longueur.

Deux colliers viennent se fixer sur les entretoises à l'aide de vis 4 MA qui maintiendront solidement les extrémités de la bobine de travail.

Pour insérer les extrémités de la bobine de travail dans les colliers (voir la figure 4), vous devez desserrer les vis puis introduire les broches de la bobine de façon à ce qu'elle se trouve du côté du bornier d'alimentation.

Positionnez la bobine de travail de façon à laisser moins de 1 cm de fil de cuivre en porte-à-faux par rapport au côté du circuit imprimé, ensuite serrez les vis des colliers (voir la figure 5).

Utilisation

Le circuit doit être alimenté avec une tension continue comprise entre 10 VDC et 48 VDC avec une puissance suffisante.

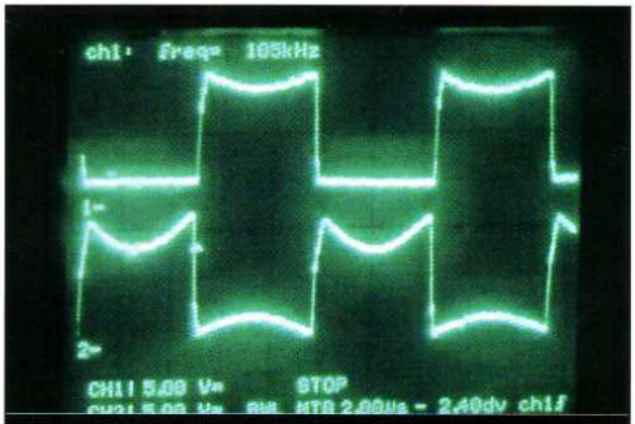


Figure 7 : formes des signaux sur les grilles des deux MOSFET.

Si la tension d'alimentation maximale de 48 VDC est appliquée, nous recommandons une alimentation dont la puissance soit d'au moins 1500 W.

Insérez le fil positif et négatif de l'alimentation dans le bornier en respectant la polarité, puis serrez les vis. Mettez sous tension le circuit et vérifiez que la LED verte LD1 s'allume pour indiquer la présence de la tension d'alimentation.

À ce stade, à l'intérieur de la bobine de travail, un courant de résonance circule avec une fréquence d'environ 100 kHz.

Si le montage est alimenté avec la tension maximale de 48 VDC et qu'aucun corps métallique ne soit introduit à l'intérieur de la bobine de travail, la consommation d'énergie doit être inférieure à 500 W.

Si vous placez un objet à l'intérieur de la bobine de travail, en fonction du matériau dont il est composé, de ses dimensions, de sa géométrie et de sa position relative, la puissance consommée peut atteindre près de 1500 W.

Après avoir choisi un objet conducteur à insérer dans la bobine de travail, comme par exemple une vis métallique, saisissez-la et maintenez-la en place à l'aide d'une pince pour éviter les brûlures, sans trop approcher cette dernière de la bobine. Insérez lentement le conducteur à l'intérieur de la bobine de travail, en essayant de ne pas toucher les parois internes de la bobine. Si possible, maintenez l'objet au centre de la bobine de travail dans une position verticale.

L'objet introduit constitue le secondaire d'un transformateur virtuel dont la bobine de travail correspond à l'enroulement primaire. Le matériau conducteur (objet) est soumis aux lignes du champ magnétique qui sont très intenses par rapport aux lignes de champ présentes à l'extérieur de la bobine.

Au bout d'un moment, l'objet s'échauffe par effet Joule et change de couleur, en prenant une teinte typique jaune-orangé (voir la figure 6). Cette condition peut être maintenue pendant quelques dizaines de secondes, mais il faut tenir compte du fait que la température peut

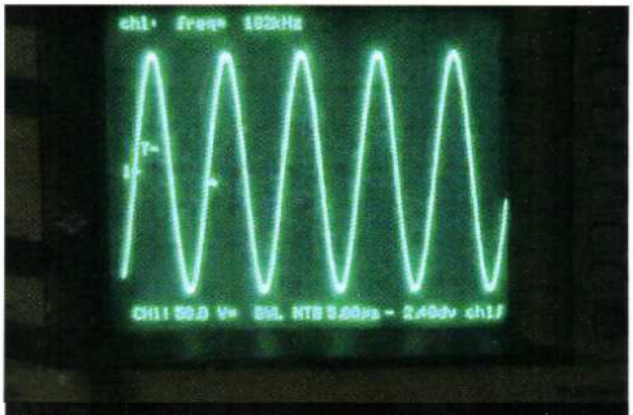


Figure 8 : forme du signal aux bornes de la bobine de travail.

Précautions d'utilisation

Le dispositif présenté dans cet article vous permet de réaliser des expériences intéressantes et vous procurer des fonctionnalités utiles, mais il doit être utilisé avec précaution.

- **Assurez-vous qu'il n'y a pas de substance inflammable à proximité**, car le montage dégage une forte quantité de chaleur au niveau de ses composants et des objets introduits à l'intérieur de la bobine de travail.
- **Assurez-vous qu'il n'y a pas d'appareil électronique à proximité, tels que des appareils médicaux.** Le chauffage par induction ZVS produit un champ magnétique élevé d'environ 10 mT (soit 100 gauss) à une fréquence de 100 kHz, cela pourrait perturber des appareils électroniques tel qu'un pacemaker, etc.
- **Vous devez toujours utiliser des pinces avec une poignée isolée pour manipuler les objets** introduits à l'intérieur de la bobine de travail **afin d'éviter des brûlures.**
- **Ne touchez pas à main nue les composants du chauffage par induction ZVS et les objets introduits dans la bobine de travail**, après ou pendant le fonctionnement, afin d'éviter des brûlures et/ou des chocs électriques.

s'élever fortement. Il faut donc faire particulièrement attention à ne pas entrer en contact direct avec l'objet ou la bobine de travail afin d'éviter des brûlures.

Notez aussi qu'après quelques minutes d'utilisation du chauffage par induction ZVS, l'enroulement de la bobine de travail aura tendance à s'assombrir, jusqu'à ce qu'il devienne pratiquement noir à cause de la chaleur produite. Cependant, cela n'empêchera pas son fonctionnement.

Mesures

Le courant circulant à l'intérieur de la bobine de travail a une valeur efficace d'environ 100 A.

Avec un courant aussi élevé, **le champ magnétique est considérable** et donc tous les objets à proximité sont soumis aux lignes de champ du champ magnétique, de sorte que les dispositifs sensibles au magnétisme pourraient être perturbés.

Le champ magnétique B peut être calculé avec la formule suivante :

$$B = \mu_0 N I / L = 10{,}7 \text{ mT} = 107 \text{ gauss}$$

où N = 6, I = 100 A et L = 1,26 µH.

La figure 8 montre la forme de l'onde à l'oscilloscope des tensions entre les grilles et sources des transistors MOSFET M1 et M2. Comme vous pouvez le voir, les tensions sont parfaitement en phase.

Avant que la tension de la grille du transistor M2 prenne un niveau haut et fasse conduire complètement le transistor M2, la tension de l'autre grille (de M1) est déjà à un niveau bas assurant ainsi que M1 soit bloqué. Cela évite la conduction simultanée des deux MOSFET, ce qui provoquerait la destruction de ces derniers.

La figure 8 montre la tension aux bornes de la bobine de travail, qui atteint des niveaux crêtes d'environ 150 V. Sa forme est parfaitement sinusoïdale grâce au circuit résonnant RLC parallèle.

Enfin, la figure 9 montre la consommation de courant du circuit alimenté par une tension de 48 VDC, le courant est égal à 17 A.

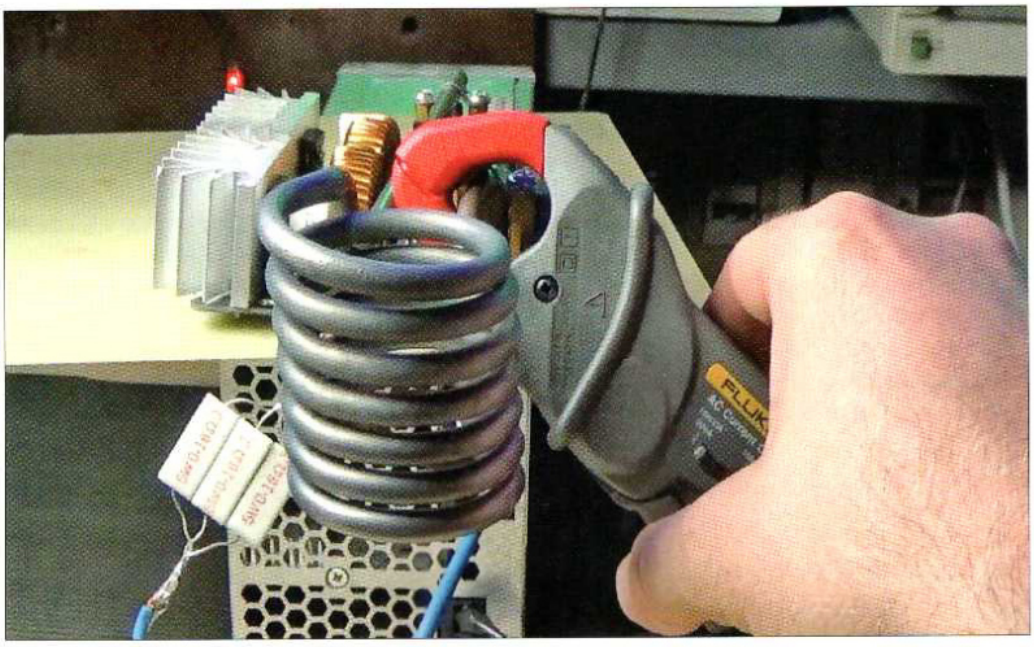


Figure 9 : la pince ampèremétrique permet de vérifier le courant absorbé par le circuit.

La mesure du courant est effectuée sur la base de la chute de tension mesurée avec un multimètre numérique aux extrémités d'une résistance de shunt de 0,06 Ω (constituée par 3 résistances de 0,18 Ω montées en parallèle) et avec un objet inséré dans le bobine.

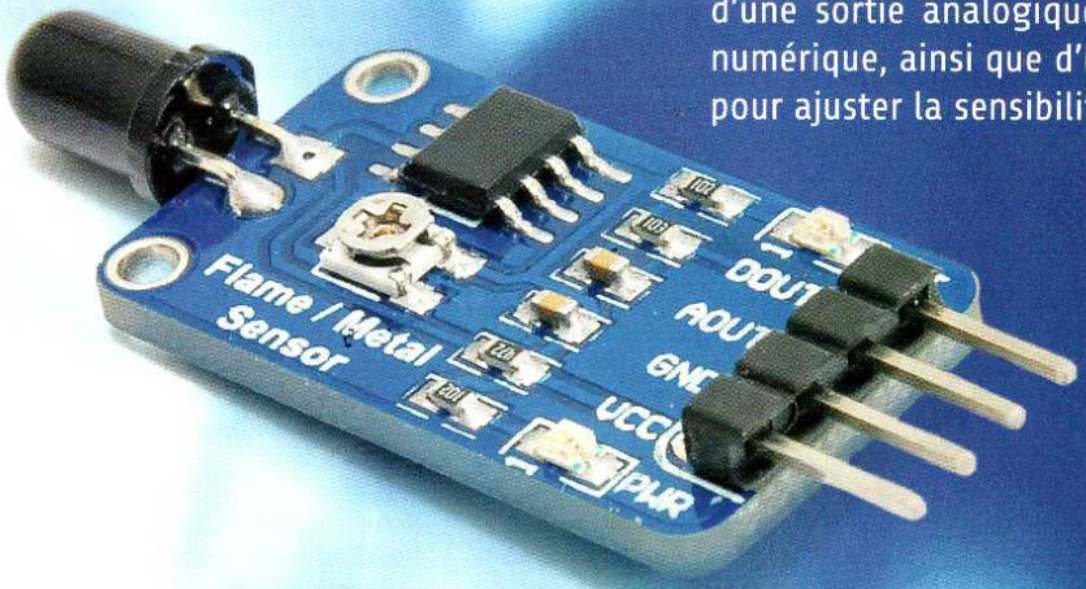
Sinon, vous pouvez utiliser une pince ampèremétrique.

Nous concluons en vous rappelant une fois de plus que vous devez faire très attention dans l'utilisation du montage, il vous permettra de mettre en œuvre des expériences intéressantes. ■

DÉTECTEUR DE FLAMME

Ce montage permet d'identifier la présence d'une flamme grâce à une photodiode IR qui détecte les infrarouges correspondant à la chaleur qui en émane, et cela dans un angle de ± 60 °. Il dispose d'une sortie analogique et d'une sortie numérique, ainsi que d'un potentiomètre pour ajuster la sensibilité.

de Boris Landoni



Il existe de nombreuses situations parmi lesquelles il est nécessaire de vérifier la présence d'une flamme. Par exemple, dans les brûleurs des chaudières et des fours industriels ou encore dans les ateliers de carrosserie, dans les feux des cuisines et dans d'autres zones. Dans les systèmes où l'électronique n'était pas largement utilisée, la principale méthode de détection d'une flamme consistait à mettre en contact un capteur thermique formé typiquement par un thermocouple, dont la tension était lue par un circuit constitué d'un amplificateur analogique et d'un comparateur. Ce circuit était capable de fournir une tension dans le cas de la présence d'une flamme et dans le cas contraire (pas de flamme), la tension était nulle.

Ce type de détection était réalisée par la propagation directe de la chaleur. Aujourd'hui, nous avons tendance à privilégier la détection sans contact (contactless) à l'aide de détecteurs ioniques fonctionnant sur le principe du courant ionique lié à la flamme, ou à l'aide de détecteurs optiques à infrarouges. En effet, pour des raisons de fiabilité et de sécurité, il convient d'éviter que le capteur soit directement en contact avec la flamme, car avec le temps il s'oxyde et le métal, qui constitue les capteurs thermiques, se corrode. La détection optique est réalisée en exploitant le fait que la chaleur qui est irradiée par la flamme (comme toute source de chaleur en général) produit un rayonnement électromagnétique dont le spectre se situe dans la longueur d'onde du spectre infrarouge. La longueur d'onde des infrarouges est comprise entre le domaine visible (environ 0,7 µm) et le domaine des micro-ondes (environ 1 mm). Il est donc possible d'utiliser des dispositifs électroniques sensibles aux infrarouges, comme nous allons le décrire dans le projet ci-après.

Le schéma électrique

Pour comprendre le fonctionnement du montage, regardons le schéma électrique qui repose sur une **photodiode sensible à l'infrarouge** et plus précisément, dans la gamme comprise entre **760** et **1100 nanomètres**.

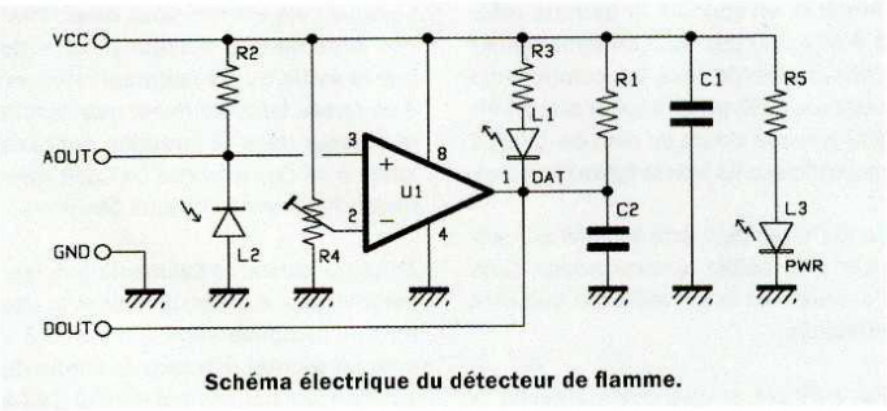


Schéma électrique du détecteur de flamme.

Cela permet de détecter la présence d'une flamme dans le voisinage mais à une distance telle qu'elle n'est pas exposée à la chaleur. Le boîtier en plastique bleu de la photodiode la rend insensible à la lumière visible, augmentant ainsi sa sensibilité et sa capacité de distinguer une source de chaleur même à des distances assez grandes.

Dans le circuit, la photodiode est montée dans une configuration classique, c'est-à-dire qu'elle est **polarisée en inverse** avec une résistance en série du côté positif de l'alimentation.

Dans ces conditions, **son courant de saturation inverse** dans une situation de lumière très faible (obscurité) n'est que de **quelques dizaines de nano ampères**. Il **croît très rapidement** lorsque la jonction de la photodiode **capte un rayonnement infrarouge**, tel que celui d'une flamme par exemple. Ainsi, si dans l'angle de détection (qui est de ± 60° par rapport à l'axe), est détecté en face de la photodiode un rayonnement infrarouge d'intensité suffisante (c'est-à-dire si la photodiode « voit » une flamme petite ou grande), le courant inverse dans celle-ci croît très rapidement et provoque une chute de tension aux bornes de la résistance R2 qui augmente selon la **loi d'Ohm**.

La différence de potentiel entre la tension d'alimentation VCC et la chute de tension aux bornes de R2 correspond à la différence de potentiel présente entre l'anode et la cathode de la photodiode. Cette différence de potentiel est ainsi appliquée sur l'entrée non-inverseuse (broche 3) du **comparateur LM393** alors que sur son entrée inverseuse (broche 4) est appliquée une tension de référence réglable au moyen du trimmer R4.

Cela permet de régler le seuil d'activation du capteur, car en fonction de la tension appliquée à l'entrée inverseuse du LM393, une tension plus ou moins élevée aux extrémités de la photodiode fera basculer le comparateur. Le circuit U1 a sa sortie à un niveau haut lorsque la broche 2 est à un potentiel supérieur à celui de la broche 3 et vice versa. Lorsque le trimmer R4 provoque une augmentation de la tension appliquée sur l'entrée inverseuse, la sensibilité augmente.

Même avec un rayonnement infrarouge faible, la tension entre l'anode et la cathode de la photodiode diminue, cela suffit à commuter la sortie du LM393 d'un niveau haut (environ VCC) vers un niveau haut (environ 0 V). En amenant le curseur de R4 vers la masse, la lumière infrarouge doit être plus intense pour faire commuter la sortie du comparateur à un niveau bas.

Il en découle que la sortie (broche 1) du LM393 est à un niveau haut au repos et à un niveau zéro lorsque la photodiode détecte le rayonnement infrarouge d'une flamme. Dans cette condition, la LED L1 est polarisée lorsque la sortie du circuit U1 est à un niveau bas (0 V). La LED L1 s'allume donc (sa cathode se trouve à un potentiel proche de celui de la masse car la sortie de U1 est à 0 V) grâce à la résistance R3 de limitation du courant.

Notez que la sortie du circuit intégré se trouve à un niveau haut car son étage de sortie est composé d'un transistor NPN configuré en collecteur ouvert, il faut donc connecter la résistance R1 de tirage (pull-up) entre la sortie et l'alimentation positive VCC.

Le condensateur C2, connecté en parallèle avec la sortie du LM393, forme avec la résistance R1 un circuit RC dont la constante de temps permet d'éviter de faux changements d'états du comparateur, provoqués par d'éventuelles perturbations sur l'alimentation ou par des ondes lumineuses ou encore par des rayonnements thermiques provenant du milieu extérieur et qui pourraient perturber le fonctionnement de la photodiode.

L'alimentation du circuit est filtrée par le condensateur C1 placé entre l'alimentation VCC et la masse GND. La LED L3, avec la résistance R5, indique que le montage est sous tension.

Complétons la description du schéma électrique avec les broches « AOUT » et « DAT », dont la première permet de prélever le potentiel présent sur l'anode de la photodiode (qui est proportionnel à l'intensité du rayonnement infrarouge détecté). La seconde fournit un niveau logique correspondant au seuil détecté, c'est-à-dire à un niveau haut si aucune flamme n'est détectée ou si l'infrarouge détecté n'a pas une intensité suffisante ou à un niveau bas en cas de détection infrarouge.

Réalisation pratique

Après avoir étudié le fonctionnement du montage, passons à la fabrication de ce détecteur. Il s'agit d'un circuit imprimé double face de petites dimensions. Les typons et les fichiers GERBER du circuit sont disponibles en téléchargement sur

CARACTÉRISTIQUES TECHNIQUES :

- **Sensibilité : de 760 nm à 1100 nm ;**
- **Angle de détection : ± 60° ;**
- **Tension d'alimentation : de 3,3 V à 5,3 V ;**
- **Courant consommé : 8 mA (avec la LED allumée) ;**
- **Température de fonctionnement : de - 25°C à + 85°C ;**
- **Dimensions : 27,3 mm x 15,4 mm.**

notre site www.electroniquemagazine.com dans le sommaire détaillé de la revue.

Le circuit est relativement simple à fabriquer, cependant les composants sont des CMS. Il faudra être soigneux et patient pour souder les composants. Sachez que le circuit est disponible sous forme de kit.

Commencez par souder délicatement les résistances et les condensateurs. Pour les résistances, nous vous conseillons de mesurer la valeur avec un multimètre avant de les souder. Cela évitera des erreurs car parfois le marquage est difficile à identifier.

Ensuite soudez le comparateur LM393, son repère en forme de « U » doit être orienté vers l'extérieur du circuit imprimé. Puis passez au trimmer, aux LED en faisant attention à leur orientation. La cathode de chaque LED est repérée à l'aide d'un trait de couleur verte, il doit être orienté vers la barrette mâle à 4 pôles.

Enfin, il ne vous reste plus que la photodiode à souder, en pliant au préalable ses pattes à 90 °. La patte la plus longue correspond à l'anode.

Terminez en soudant la barrette mâle à 4 pôles au pas de 2,54 mm. Vérifiez l'orientation de tous les composants polarisés (LED, photodiode et circuit intégré) en vous aidant du plan de câblage des composants (voir la figure ci-contre).

Le montage peut être installé à l'intérieur d'un boîtier ou directement dans l'appareil où la surveillance doit être effectuée.

Par exemple, si vous devez détecter la présence d'un incendie, l'idéal est de placer le circuit dans un boîtier en plastique et de laisser en dehors la photodiode contre un mur mais pas trop près du plafond. La fumée a tendance à faire obstacle à la propagation des infrarouges.

Si vous devez détecter la présence d'une flamme dans un appareil à gaz, le montage doit être fixé sur un support en pointant la photodiode vers la flamme.

Pour l'étalonnage, vous devez pointer la photodiode vers la flamme que vous voulez détecter et régler le trimmer R4 en partant avec le curseur vers VCC et en tournant pour aller vers la masse, jusqu'à ce que la sortie « DAT » passe d'un niveau haut à un niveau bas (0 V).

Lorsque cela est fait, vous devez tourner lentement le curseur jusqu'à ce que la sortie du comparateur revienne à un niveau haut. Ramenez maintenant le curseur dans la direction opposée jusqu'à ce que la sortie DAT soit commutée à un niveau logique bas (0 V).

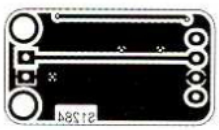
En ce qui concerne l'alimentation, rappelez-vous que le circuit nécessite une tension comprise entre 3,3 V et 5,3 V avec un courant dérisoire de l'ordre de 1 mA en fonctionnement normal (DAT à « 1 ») et de l'ordre de 8 mA avec la sortie à un niveau bas et la LED allumée.

Applications

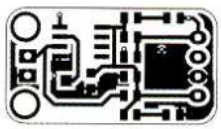
En plus de détecter la présence d'une flamme dans une chaudière ou dans un brûleur en général, notre capteur peut être utilisé dans des systèmes de détection d'incendie.

Par exemple, plusieurs capteurs connectés peuvent être installés et placés dans des locaux ou dans un entrepôt disposant d'une centrale d'alarme, dans des robots de lutte contre l'incendie, dans des systèmes automatiques d'anti-incendie reliés à un mécanisme d'extinction au dioxyde de carbone, etc. ■

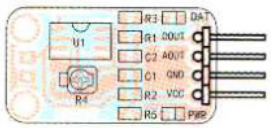
Plan de montage du détecteur de flamme



Circuit imprimé à l'échelle 1 : 1 côté soudures (face inférieure du circuit) du détecteur de flamme.



Circuit imprimé à l'échelle 1 : 1 côté composants (face supérieure du circuit où les composants sont soudés) du détecteur de flamme.



Plan de câblage des composants du détecteur de flamme.

Photo de l'un de nos prototypes du détecteur de flamme.

Liste des composants du détecteur de flamme

R1..... 10 kΩ boîtier 0805
R2..... 4,7 kΩ boîtier 0805
R3..... 1 kΩ boîtier 0805
R4..... trimmer 10 kΩ
R5..... 1 kΩ boîtier 0805

C1..... 100 nF céramique boîtier 0805
C2..... 100 nF céramique boîtier 0805
L1 ..... LED rouge boîtier 0805
L2 ..... SFH213
L3 ..... LED verte boîtier 0805

U1..... LM393

Divers

Barrette mâle 4 pôles 90°

ENREGISTREUR VOCAL AVEC PIC & EEPROM

Troisième partie

de Marco LANDONI

Nous terminons la description de notre enregistreur vocal à PIC en décrivant le module le plus simple et le logiciel de gestion avec l'interface série nécessaire pour une connexion à un PC. Troisième et dernière partie.

Dans les numéros 141 et 142 de la revue, nous vous avons présenté et décrit notre idée de réaliser, à l'aide d'un microcontrôleur Microchip, des modules de synthèse vocale (Speech Module) contrôlés par des signaux « TOR » (tout ou rien) ou par des impulsions, tout comme les célèbres circuits ISD de Winbond.

La raison de ce projet est que les anciens circuits intégrés ISD (qui nécessitaient un certain nombre de composants externes) sont de plus en plus difficiles à trouver sur le marché.

Nous avons donc pensé à implémenter les fonctions de ces puces ISD à l'aide de modules simples et peu coûteux qui trouveront sûrement une large plage d'applications dans tous les appareils électroniques « parlants », en utilisant des composants très communs comme les microcontrôleurs PIC et des mémoires EEPROM I²C.

Après avoir introduit la philosophie de conception et la structure des enregistreurs dans le numéro 141 et en ayant décrit en détail le premier module SPC01 dans le numéro 142,

nous allons maintenant, dans ce 3ème et dernier article de la série, décrire en détail le **module enregistreur/lecteur miniature SPC02**.

Nous aborderons d'abord dans le détail le schéma électrique du module SPC02, puis nous expliquerons comment l'utiliser. Nous conclurons ensuite en décrivant l'interface série qui est commune au module SPC01 que nous avons étudié en détail dans le numéro précédent.

En figure 1, nous présentons le schéma synoptique du module SPC02 ainsi que ses caractéristiques. Vous remarquez immédiatement que le module est plus simple que le module SPC01 déjà décrit. La partie matérielle du module SPC02 n'est pas personnalisable comme celle du module SPC01, car il est dédié à un usage spécifique et pas aussi générique que le précédent.

Toutefois, la structure et le fonctionnement sont similaires à ceux du module SPC01 puisque le microphone, les mémoires, les étages de filtrages analogiques et la structure du programme (firmware) sont quasiment identiques pour les deux modèles.

Dans cette configuration minimale, nous trouvons **seulement deux mémoires EEPROM**, qui permettent une capacité **d'enregistrement** maximale de **65** s avec une fréquence d'échantillonnage de 8 ksps (la fréquence d'échantillonnage du module SPC02 ne peut pas être modifiée).

Par contre, nous avons intégré dans le module un **préamplificateur microphonique** et un **amplificateur BF**. Il ne vous reste plus qu'à connecter un microphone, un haut-parleur, 3 boutons et alimenter le circuit pour commencer à l'utiliser.

Toutes les connexions sont disponibles sur un connecteur au pas de 2,54 mm qui se trouve sur un côté du circuit imprimé afin de permettre l'insertion du module dans un circuit existant, comme par exemple une alarme qui reproduit un message vocal, ou encore un détecteur de passage avec alerte vocale, etc.

Schéma électrique

Si vous vous reportez à la revue précédente qui traite de la description du module SPC01, vous remarquerez immédiatement une grande similitude entre les deux circuits. En effet, les schémas synoptiques des deux modules sont pratiquement identiques, cependant le module SPC02 est en fait un compromis entre le module SPC01 qui est plus complexe et la Demoboard.

En partant de l'entrée MIC (microphone), nous trouvons un premier amplificateur opérationnel (**U5a**) monté en **configuration inverseuse**, c'est-à-dire que le signal du microphone passe à travers la broche « - » (2).

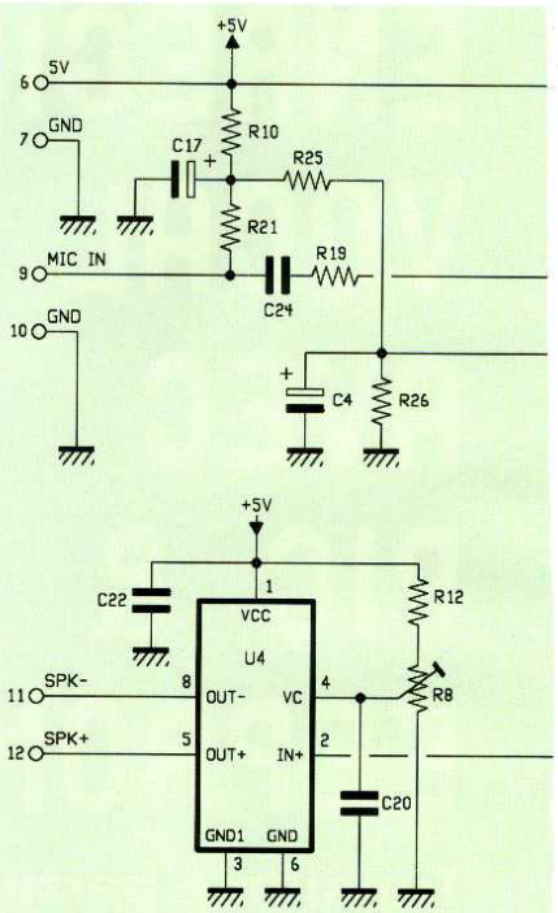


Figure 1A : notre enregistreur vocal numérique se compose d'un microcontrôleur Microchip qui numérise le signal audio correctement filtré pour éviter les distorsions, provenant du microphone. Il effectue aussi les opérations inverses, c'est à dire la lecture des données en mémoire, puis la conversion des données échantillonnées via la sortie PWM en un signal audio analogique qui est amplifié par le circuit U4.

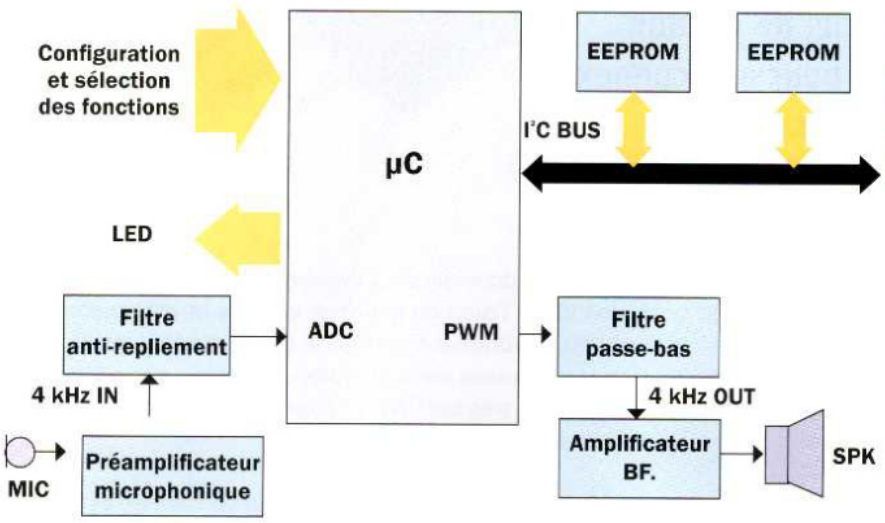
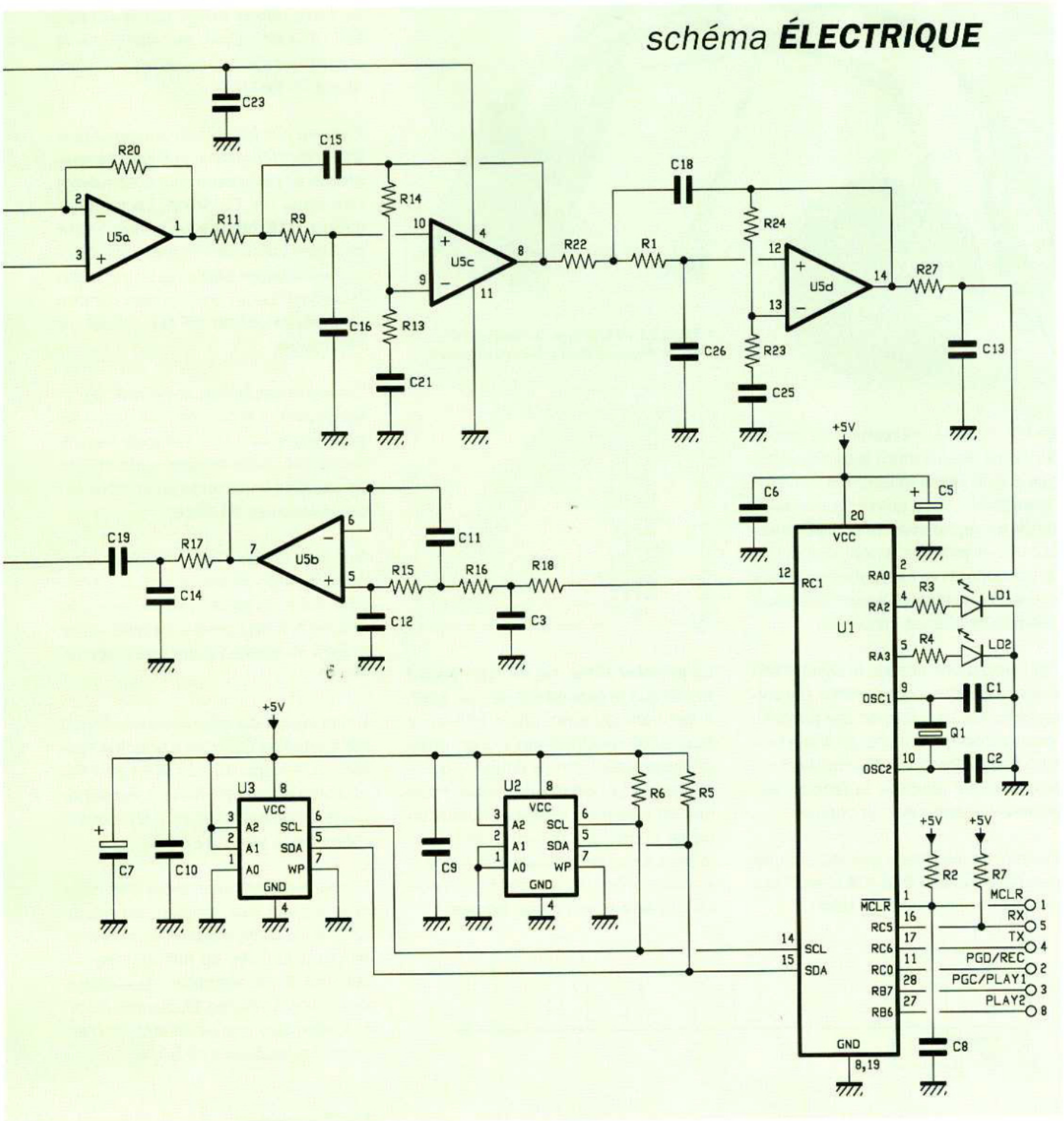


Figure 1 : schéma synoptique du module SPC02.

Cet AOP permet d'**amplifier** le signal provenant du microphone d'environ **25 dB**. Ce premier étage est suivi par deux autres amplificateurs opérationnels « **U5c** » et « **U5d** » qui forment deux **cellules de Sallen-Key** montées en cascade et utilisées comme des **filtres actifs passe-bas** dont la **fréquence de coupure** de chaque filtre est de **3,1 kHz**. Ces deux filtres actifs sont reliés à un **filtre passif passe-bas** constitué par

schéma **ÉLECTRIQUE**

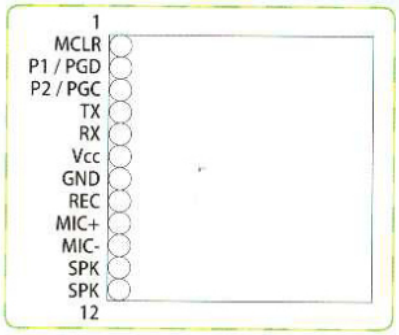


la résistance **R27** et le condensateur **C13**. L'ensemble forme, comme pour le module SPC01, un **filtre d'anti-repliement passe-bas** efficace du **5ème ordre**.

Le signal, ainsi filtré, est **prélevé aux bornes de C13** pour **rejoindre l'entrée du convertisseur A/D** qui se trouve à l'intérieur du microcontrôleur PIC18F2420. Celui-ci échantillonne le signal, le traite, enregistre les données dans la mémoire et effectue toutes les opérations inverses : lecture de la mémoire (extraction des données), puis conversion en un signal analogique dirigé vers la sortie audio.

Lors de la lecture d'une piste, le microcontrôleur utilise le périphérique PWM(2) pour générer le signal analogique. Sur la broche 12 de U1 est présent un signal carré à la fréquence fixe de 195 kHz. Le programme fait varier, échantillon par échantillon, le rapport cyclique D de ce signal proportionnellement à l'amplitude du signal audio décompressé qui doit être reproduit. Afin de pouvoir extraire le signal modulé (en l'occurrence la piste enregistrée) de l'onde PWM, il est nécessaire d'effectuer une moyenne du signal, c'est-à-dire le filtrer à travers une cellule passe-bas.

Figure 2 : brochage du module SPC02 vu de dessus (côté trimmer et quartz).



En fait, il existe une relation de proportionnalité directe entre le rapport cyclique D et la valeur moyenne d'un signal rectangulaire. En d'autres termes, **le filtrage du signal supprime la porteuse** ou, si vous préférez, exploite le fait que le condensateur est instantanément chargé à une tension proportionnelle à la durée de chaque impulsion.

Par conséquent, lorsque le signal PWM a un rapport cyclique élevé (impulsions longues par rapport aux pauses), le condensateur est chargé à une tension élevée. Par contre, lorsque le rapport cyclique diminue, la tension aux bornes du condensateur diminue.

Examinons les filtres constitués d'une part par le réseau **R18/C3** et par l'amplificateur opérationnel « **U5b** ».

Le **premier filtre** est de type **passif** tandis que le **second** est de type **actif**. Il est indispensable de préfiltrer le filtre actif constitué par « U5b » avec un étage passif afin de limiter la dynamique de la tension du signal PWM (qui est d'un point de vue électrique un signal numérique pouvant varier seulement entre deux niveaux, à savoir 0 et Vcc) car l'entrée (broche 5) de l'AOP « U5b » serait sans doute saturée.

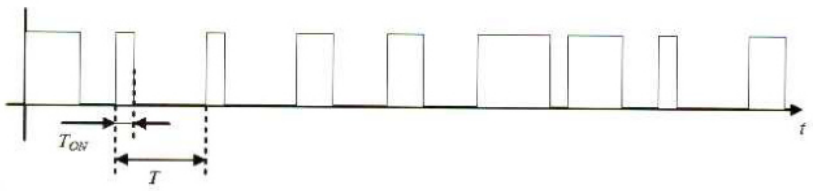


Figure 3 : le rapport cyclique d'un signal rectangulaire est défini par le rapport $D = T_{ON} / T$. La moyenne du signal PWM est proportionnelle à D, par conséquent, en faisant varier D, la valeur moyenne du signal PWM varie dans le temps. Pour obtenir la valeur moyenne relative au signal PWM, il suffit de le filtrer avec une cellule passe-bas. La partie analogique du signal, représentée par V_{OUT}, est ainsi disponible en sortie du filtre.

Le filtre passif formé par le réseau R17/C14 est utilisé pour éliminer tous les résidus de la fréquence porteuse du signal PWM.

Ensuite, aux **bornes** du condensateur **C14**, nous retrouvons le **signal décompressé et reconstruit** qui était mémorisé dans les EEPROM. Le condensateur **C19 bloque la composante continue** présente en sortie de « U5b » tout en laissant passer le signal audio (alternatif) pour l'envoyer vers l'entrée de l'**amplificateur BF U4** qui est un **TDA7052A**.

Ce circuit est fabriqué par NXP Semiconductors, il est capable de piloter un petit haut-parleur de 1 W avec une alimentation mono tension sans utiliser de capacité importante (encombrante) de découplage en sortie.

De plus, le TDA7052A dispose d'un contrôle de volume externe. En faisant varier la tension présente sur la broche 4, il est possible de faire varier le gain en tension dans une plage de 80 dB.

L'interfaçage du microcontrôleur avec les mémoires EEPROM s'effectue également à l'aide du bus SPI sur 2 fils et, comme nous pouvons le constater sur le schéma électrique, il n'y a que 2 mémoires, à savoir U2 et U3.

Les boutons des commandes sont reliés directement aux broches du microcontrôleur, configurées en entrées avec leurs résistances de pull-up internes activées afin de déterminer l'état logique de repos des broches. L'ensemble du circuit nécessite une seule tension d'alimentation stabilisée de 5 VDC.

Le firmware

Nous allons maintenant analyser le firmware (programme) implémenté dans le microcontrôleur. Nous décrirons uniquement la séquence des opérations effectuées pendant la phase d'enregistrement (la reproduction est très similaire) et nous illustrerons brièvement les astuces qui accélèrent l'exécution du programme. Dans ce type d'application, il est essentiel d'effectuer un certain nombre d'opérations

Caractéristiques techniques du module SPC02

- format d'enregistrement : ADPCM 4 bits ;
- durée d'enregistrement : 65 s à 8 ksps ;
- fréquence d'échantillonnage : 8 ksps ;
- résolution effective : 8 bits ;
- entrée analogique : microphone (40 mVppmax) ;
- sortie analogique : haut-parleur 500 mW / 16 Ω ;
- commandes : PLAY piste 1, PLAY piste 2, REC ;
- commandes et fonctions via le port série (RS232), LED rouge et verte de signalisation.

(et non d'instructions du point de vue programmation) dans les plus brefs délais.

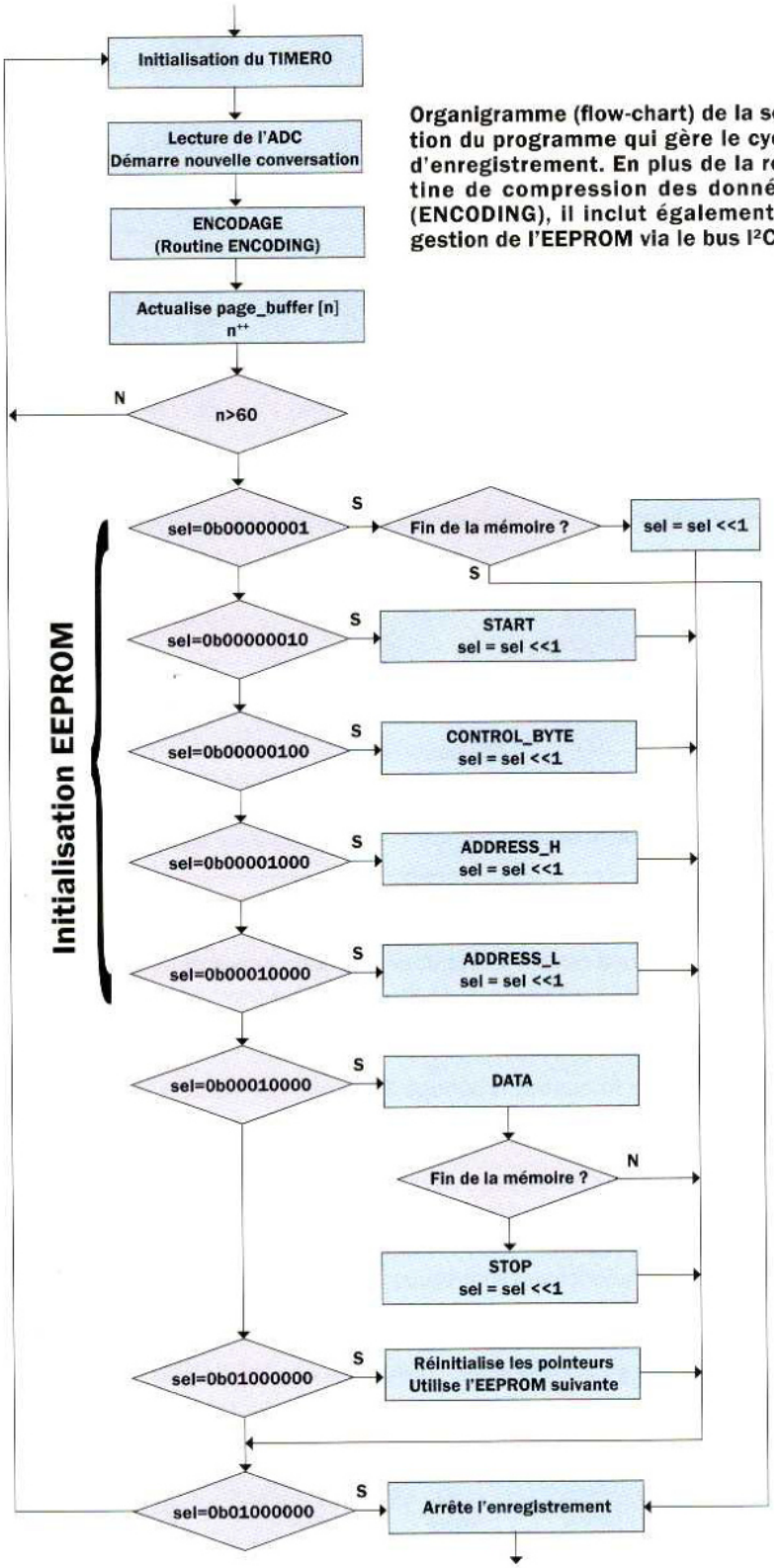
Nous allons nous référer aux concepts décrits dans le premier article, il est peut-être utile de les relire pour une compréhension plus facile du code source.

Tout d'abord, il est nécessaire de préciser que nous **utilisons** un « timer » interne au PIC, à savoir le « **TIMER0** ». Ce dernier cadencera avec précision la succession des opérations.

Ce « timer » est configuré pour générer un événement dans un intervalle de temps égal à : **ts = 1 / fs**. C'est-à-dire, chaque fois qu'un échantillon audio doit être acquis.

Nous n'avons pas choisi d'associer une routine d'interruption au « timer », mais d'aller **interroger** dans le programme le « **flag** » d'**interruption**. En effet, en écrivant le programme en langage C, chaque fois qu'une interruption est appelée, plusieurs instructions sont exécutées automatiquement (en plus de celles du programme) par le compilateur pour la gestion de l'instruction elle-même, ce qui aurait soustrait du temps d'exécution au programme.

Après vérification de la condition de démarrage (START) de l'enregistrement, le cycle des opérations effectué pendant l'enregistrement est représenté par l'organigramme ci-après.



Organigramme (flow-chart) de la section du programme qui gère le cycle d'enregistrement. En plus de la routine de compression des données (ENCODING), il inclut également la gestion de l'EEPROM via le bus I²C.

Le cycle principal est activé à chaque échantillonnage « ts » via le « TIMER0 » et la première opération consiste à lire les données converties par l'ADC.

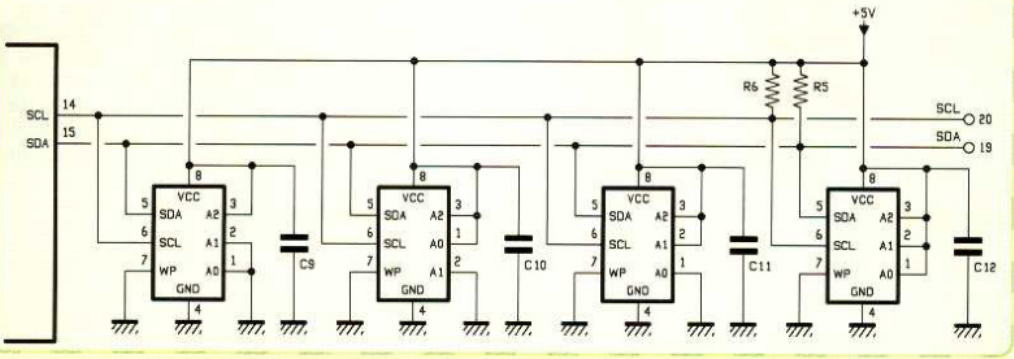
La même opération est redémarrée immédiatement après pour une nouvelle acquisition.

De cette manière, l'ADC peut effectuer la conversion en arrière-plan par rapport à l'exécution du programme, puisque les données ne seront demandées qu'au début du cycle suivant.

L'échantillon prélevé sur l'ADC est ensuite compressé et ajouté au buffer

Gestion de l'EEPROM

Figure A : le bus I²C utilisé dans le module SPC01.



Les mémoires utilisées dans les modules SPC01 et SPC02 sont de type série avec une interface de type bus I²C. Cette interface utilise seulement deux fils, un pour l'horloge (SCL) qui analyse la communication, et l'autre pour les données (SDA). Il s'agit d'un bus synchrone et adressable, en ce sens que les mêmes lignes peuvent être partagées par plusieurs périphériques connectés sur le bus, chacun d'entre eux se distinguant par sa propre adresse. Les périphériques connectés sur le bus peuvent être de type maître (MASTER) ou de type esclave (SLAVE). Le maître est celui qui décide quand commencer une session de communication et il y en a normalement qu'un seul sur le bus. L'esclave répond simplement aux demandes du maître, c'est-à-dire exécute les commandes et reçoit ou transmet les données demandées.

Listing 1

```
//          +----------        I2C bus ----------+
            SSPCON1 = 0b00101000;       // configuration MASTER, active l'I2C
            SSPCON2 = 0b00000000;       // au repos...
            SSPADD = 0x18;              // fréquence du bus = 400 kHz
```

Au niveau électrique, tous les périphériques sont connectés avec des ports bidirectionnels configurés en sortie, ils doivent être de type à drain ouvert. Il faut donc des résistances de tirage (pull-up) communes à tout le bus. Plus la vitesse du bus est élevée, plus les résistances de tirage doivent avoir des valeurs faibles, sinon en raison des capacités parasites des structures MOS contenues dans les puces, la communication ralentira.

Le bus I²C a une fréquence d'horloge SCL nominale de 100 kHz (mode standard) jusqu'à un maximum de 400 kHz (fast mode), même si, dans certaines conditions, il est possible d'atteindre une fréquence de 1 MHz (high-speed mode).

Comme la ligne de données est unique, il n'est possible de transmettre qu'une donnée à la fois et de manière synchrone avec le signal d'horloge. De plus, le protocole prévoit l'envoi de signaux d'accusé de réception (ACK) et de temporisations particulières qui signalent le début d'une transmission, la poursuite et la fin. Heureusement, avec les microcontrôleurs actuels, plusieurs fonctions sont transparentes pour le programmeur et sont réduites à une simple configuration ou à une réinitialisation de certains flags spécifiques.

Examinons maintenant comment s'effectue la lecture/écriture des cellules ou des bancs de mémoire dans notre projet. Pour cela, nous vous conseillons de consulter la documentation technique du PIC18F2420 et de la mémoire 24XX1025.

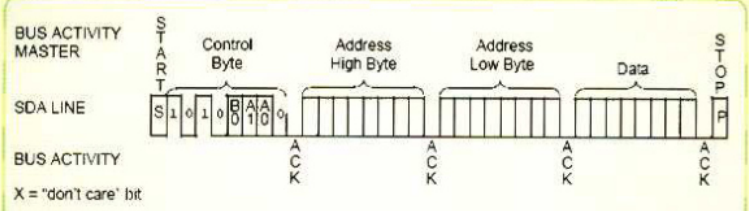


Figure B : commandes et données à envoyer pour effectuer une écriture en mémoire. L'écriture réelle des données commence après l'octet « Adress Low Byte » et se termine lorsque le bit de STOP apparaît.

Tout d'abord, le MSSP (Master Synchronous Serial Port) du PIC doit être initialisé de manière appropriée. La première instruction définit l'utilisation de broches spécifiques, active le périphérique et le configure en tant que maître (MASTER).

La seconde instruction place le port dans la condition de repos tandis que la troisième règle la fréquence de fonctionnement du bus. À ce stade, le port est opérationnel.

Pour pouvoir écrire des données dans l'une des mémoires EEPROM du bus, nous devons effectuer

des opérations de lecture/écriture de données sur le bus en fonction de ce qui est indiqué dans la documentation de la mémoire. En particulier, si nous voulons écrire un seul octet dans un emplacement spécifique de la mémoire, nous devons :

- envoyer un message de démarrage (bit de START) ;
- envoyer un octet de contrôle (Control Byte) dans lequel est spécifié à quelle mémoire (à quel circuit intégré) nous souhaitons accéder ;
- envoyer deux octets d'adressage ;
- envoyer l'octet à écrire ;
- envoyer le message d'arrêt (bit de STOP).

Listing 2

```
// START ----------
SSPCON2bits.SEN = ENABLE;        // bit de START
while(SSPCON2bits.SEN);          // attend l'esclave...

// configuration du mode d'écriture ----------
SSPCON2bits.ACKDT = ZERO;        // répond avec ACK
PIR1bits.SSPIF = ZERO;
SSPBUF = 0xA0 | (M << 1);        // octet de contrôle, M = mémoire à adresser
while(!PIR1bits.SSPIF);
while(SSPCON2bits.ACKSTAT);      // attend l'ACK

// configuration de l'adresse d'écriture ----------
PIR1bits.SSPIF = ZERO;
SSPBUF = ADR_H;                  // b15 - b8
while(!PIR1bits.SSPIF);
while(SSPCON2bits.ACKSTAT);      // attend l'ACK

PIR1bits.SSPIF = ZERO;
SSPBUF = ADR_L;                  // b7 - b0
while(!PIR1bits.SSPIF);
while(SSPCON2bits.ACKSTAT);      // attend l'ACK

// écrit les données ----------
PIR1bits.SSPIF = ZERO;
SSPBUF = DATA;                   // écrit dans la cellule (bloc mémoire)
while(!PIR1bits.SSPIF);
while(SSPCON2bits.ACKSTAT);      // attend l'ACK

// STOP ----------
SSPCON2bits.PEN = ENABLE;        // bit de STOP
while(SSPCON2bits.PEN);          // attend l'esclave ...
```

L'octet de contrôle (c8 à c0) est structuré de la manière suivante : les 4 premiers bits « c7 » à « c4 » sont toujours fixes et consistent en une séquence fixe de 1 et de 0 alternés. Le bit « c3 » est le 17ème bit d'adresse qui, avec les 16 bits spécifiques aux deux octets d'adressage, permet d'adresser 217 cellules pour un total de 128 kilooctets (1 Mbit), c'est-à-dire toute la mémoire.

Les deux bits « c2 » et « c1 » spécifient l'adresse attribuée à l'EEPROM via les broches A0 et A1 (la broche A2 doit toujours être maintenue à Vdd). Enfin, si le bit « c0 » est réinitialisé, cela indique le mode lecture à la mémoire. S'il est réinitialisé de nouveau, cela indiquera le mode écriture à la mémoire. Le code qui effectue toutes ces opérations est réduit à quelques lignes grâce à l'intégration des technologiques dans le microcontrôleur (voir le listing 2).

À partir du moment où le signal de STOP est généré, il faut attendre au moins 5 ms pour laisser le temps de sauvegarder les données avant de pouvoir accéder au signal. Au contraire, il est possible d'accéder immédiatement à tous les autres périphériques connectés au bus.

La lecture d'une cellule de la mémoire est légèrement plus compliquée. Nous devons spécifier deux octets de contrôle pour chaque cellule à lire. Le premier est utilisé pour écrire dans le buffer de la mémoire quelle adresse nous voulons lire, tandis que le second est nécessaire pour configurer la mémoire en mode lecture.

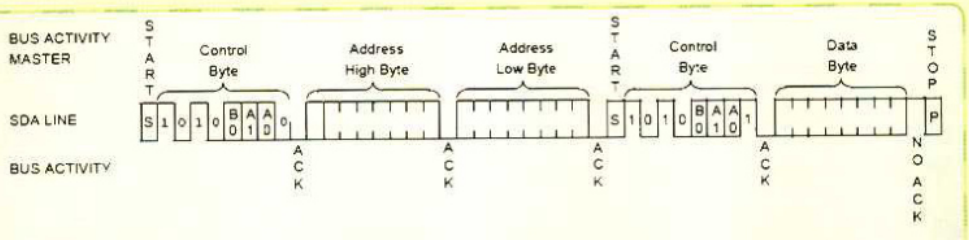


Figure C : commandes et données à envoyer pour effectuer une lecture de la mémoire.

De même, dans cette séquence de commandes sur le bus I²C, il est nécessaire de spécifier les conditions de START, de STOP et d'ACK.

La condition marquée comme un START, avant le second octet de contrôle de la figure C, est en fait une condition de redémarrage (RESTART) car elle intervient entre le premier START (tout à gauche de la figure C) et le STOP final.

Listing 3

```
// START ----------
SSPCON2bits.SEN = ENABLE;                // bit de START
while(SSPCON2bits.SEN);                  // attend l'esclave...

// configuration du mode d'écriture ----------
SSPCON2bits.ACKDT = ZERO;                // répond avec ACK
PIR1bits.SSPIF = ZERO;
SSPBUF = 0xA0 | (M << 1);                // octet de contrôle, M = mémoire à adresser
while(!PIR1bits.SSPIF);
while(SSPCON2bits.ACKSTAT);              // attend l'ACK

// configuration de l'adresse de début de lecture ----------
PIR1bits.SSPIF = ZERO;
SSPBUF = ADR_H;                          // b15 - b8
while(!PIR1bits.SSPIF);
while(SSPCON2bits.ACKSTAT);              // attend l'ACK

PIR1bits.SSPIF = ZERO;
SSPBUF = ADR_L;                          // b7 - b0
while(!PIR1bits.SSPIF);
while(SSPCON2bits.ACKSTAT);              // attend l'ACK

// configuration du mode lecture ----------
SSPCON2bits.RSEN = ENABLE;               // bit de RESTART
while(SSPCON2bits.RSEN);                 // attend le début de la confirmation
PIR1bits.SSPIF = ZERO;

SSPBUF = 0xA1 | (M << 1);                // reconfirme l'octet de contrôle en mode lecture

while(!PIR1bits.SSPIF);
while(SSPCON2bits.ACKSTAT);              // attend l'ACK
SSPCON2bits.ACKDT = ONE;                 // répond sans ACK

// lit à l'adresse ----------
PIR1bits.SSPIF = ZERO;
SSPCON2bits.RCEN = ENABLE;               // active la lecture
SSPSTATbits.BF = ZERO;
while(SSPCON2bits.RCEN);                 // attend la fin de la lecture

DATA = SSPBUF;                           // lit les données

SSPCON2bits.ACKEN = ENABLE;              // confirme NACK
while(SSPCON2bits.ACKEN);                // attend ...

// STOP ----------
SSPCON2bits.PEN = ENABLE;                // bit de STOP
while(SSPCON2bits.PEN);
SSPCON2bits.ACKDT = ZERO;                // répond avec ACK
```

Il est possible de lire ou d'écrire plusieurs données en séquence sans devoir accéder à chaque fois à chaque cellule. Pour l'écriture, nous procédons exactement comme expliqué, avec une différence. La partie du code appelée « écrit les données » doit être répétée autant de fois qu'il y a de cellules à écrire. Cependant, l'écriture doit toujours se terminer par l'envoi de la commande STOP.

Notez qu'il est possible d'écrire dans l'ordre à condition de ne pas déborder de la page mémoire. En partant de l'adresse « 0x0000 », la première page se termine à l'adresse « 0x007F ». La seconde part de l'adresse « 0x0080 » et se termine à l'adresse « 0x00FF », et ainsi de suite par pas de 128 octets pour toute la mémoire.

En mode lecture, il est aussi possible de lire en séquence. De la même manière, il sera d'abord nécessaire de répéter ce qui est contenu dans la partie du code « lecture des données » autant de fois qu'il y a de cellules à lire.

La seule précaution à prendre est que chaque octet lu doit être confirmé avec un ACK, alors qu'à la fin de la lecture du dernier octet il doit y avoir un acquittement négatif (NACK) ou pas d'acquittement, comme dans l'exemple du listing 3.

La lecture contiguë est possible à partir de l'adresse « 0x00000 » jusqu'à l'adresse « 0x0FFFF » sans interruption et de « 0x10000 » à « 0x1FFFF ».

de la mémoire interne du microcontrôleur qui est appelée « **page_buffer** ».

Nous vous rappelons que dans la « page_buffer », chaque échantillon est représenté sur **4 bits**. Par conséquent, dans les **128 octets de la mémoire** nous avons donc **256 échantillons vocaux**. La taille de 128 octets pour la « page_buffer » n'a pas été choisie par hasard. En fait, c'est précisément la taille d'une page mémoire à l'intérieur de l'EEPROM, qui peut être écrite par bloc. Par conséquent, les 256 échantillons vocaux acquis sont écrits dans une page de 128 octets de l'EEPROM.

Ainsi, tant que la quantité des données dans le buffer (qui fonctionne sur 8 bits) est inférieure à 60, soit moins de la moitié des 128 octets, le programme effectue les opérations décrites ci-dessus puis revient au début du cycle en attendant le moment exact pour commencer une nouvelle conversion.

Inversement, si le nombre d'octets dans le buffer est supérieur à 60, le programme configure la mémoire EEPROM pour recevoir un bloc de données.

Comme il n'est pas possible de transférer consécutivement 128 octets dans la mémoire sans interférer temporellement avec l'opération d'échantillonnage, le transfert des données a été « divisé » en plusieurs « étapes » (ou en plusieurs « pas ») qui sont gérées par une machine d'état.

En d'autres termes, une seule donnée (de 8 bits) est transférée à chaque cycle de la routine d'enregistrement. Il ne faut pas oublier que pour écrire dans l'EEPROM, il faut d'abord envoyer quelques octets de contrôle et spécifier l'adresse de la mémoire (voir l'encadré intitulé « Gestion de l'EEPROM). Ces opérations ont donc été séparées les unes des autres et exécutées une à la fois à chaque cycle de la routine principale.

Dans la pratique, après que le 60ème octet a été écrit dans la « page_buffer », le programme commence à dialoguer avec l'EEPROM, en envoyant un octet ou une commande à chaque cycle de la routine d'enregistrement.

Ainsi, lorsque nous commençons à remplir la seconde moitié de la « page_buffer », nous envoyons en même temps, à partir de cette même « page_buffer », le premier octet vers l'EEPROM.

À ce stade, pour compléter le buffer, 128 échantillons audio (64 octets compressés) restent à acquérir et, 128 octets doivent être transférés vers la mémoire.

Maintenant, en transférant un octet à chaque cycle d'acquisition A/D, le dernier échantillon compressé est mis à jour juste avant d'être transféré vers la mémoire (cela se produit dans le même cycle que celui de la routine d'enregistrement).

Dans la pratique, l'acquisition et le positionnement des 256 échantillons vocaux dans la « page_buffer » se termine au moment du transfert vers l'EEPROM du 128ème octet présent dans ce même buffer.

À ce stade, la communication avec la mémoire est terminée, ce qui a pour effet d'enregistrer la page qui vient d'être transférée. Le prochain cycle de la routine principal commencera à remplir le buffer depuis le début.

Cette procédure continue tant que le bouton STOP n'est pas pressé ou que le microcontrôleur détecte qu'il n'y a plus de mémoire disponible.

Vous pouvez vous demander, en regardant le code source, pourquoi la structure de la machine d'état n'a pas été construite en utilisant des instructions « case » et en incrémentant une variable. Cela aurait était plus naturel, mais en utilisant une série d'instructions « if » et une variable initialisée à « 0x01 », chaque pas est décalé d'un bit.

Eh bien, la raison est qu'un test de type « if(sel & 0b00000100) » est compilé avec une seule instruction de test sur le bit « b2 » de la variable « sel » (1 seule instruction), alors que pour effectuer la même opération avec une construction de type « case », cela aurait nécessité au moins 5 à 6 instructions machine.

Le fait de décaler la variable « sel » au lieu de l'incrémenter permet d'effectuer le test sur un seul bit. Par contre, il n'est pas possible de « briser » plus de 8 fois le flux du programme.

Réalisation pratique

Pour limiter les dimensions du module nous avons opté pour une construction avec des composants montés en surface (CMS), en choisissant des boîtiers de type 0805 afin de limiter les difficultés que vous pourriez rencontrer lors du montage.

Comme d'habitude, nous mettons en téléchargement sur notre site dans le sommaire détaillé de la revue les typons des circuits imprimés et les fichiers Gerber pour une fabrication professionnelle.

Une fois le circuit imprimé double face gravé et percé, commencez par souder le microcontrôleur U1. Pour cela, vous devez utiliser un fer avec une pointe de 0,2 mm de Ø et de la soudure spéciale pour CMS de 0,5 mm de Ø. Positionnez-le correctement à l'aide d'un point de colle.

Une fois les pattes en correspondance avec les pastilles, soudez une patte d'un côté, laissez refroidir puis souder la patte qui se trouve diagonalement opposée vers l'autre extrémité. Continuez ainsi en diagonale, en laissant refroidir entre chaque soudure.

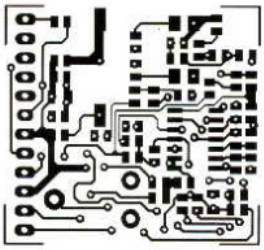
Procédez ensuite de la même manière au montage de tous les autres circuits intégrés : les mémoires, les deux amplificateurs opérationnels et l'amplificateur BF pour le haut-parleur.

Dans tous les cas, prêtez une attention particulière à l'orientation de tous les circuits intégrés, reportez-vous au plan de câblage et vérifiez plusieurs fois avant de souder la première patte de chaque circuit.

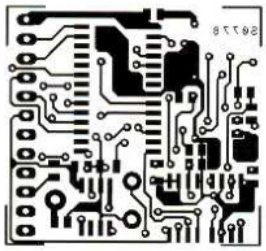
Montez ensuite tous les autres composants passifs, en prenant soin de les insérer au bon endroit.

Pour les résistances, vérifiez si possible la valeur à l'aide d'un multimètre avant de les souder, cela évite de mauvaises surprises.

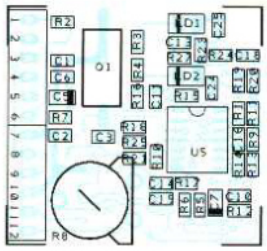
Plan de montage du module SPC02



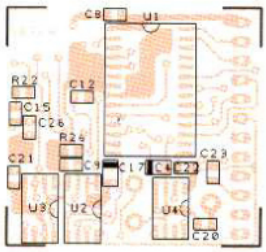
Circuit imprimé à l'échelle 1 : 1 du côté de la face supérieure (du côté du quartz) du module SPC02.



Circuit imprimé à l'échelle 1 : 1 du côté de la face inférieure (où est soudé le PIC) du module SPC02.



Plan de câblage des composants du côté de la face supérieure (du côté du quartz) du module SPC02.



Plan de câblage des composants du côté de la face inférieure (où est soudé le PIC) du module SPC02.

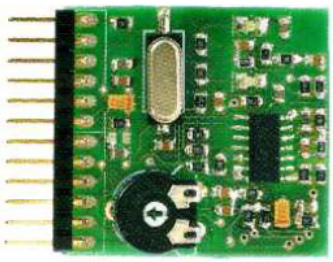


Photo de l'un de nos prototypes du côté de la face supérieure (du côté du quartz) du module SPC02.

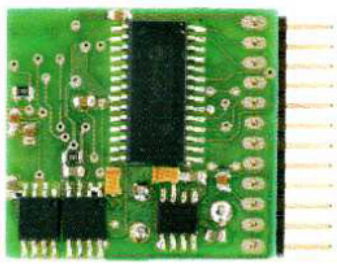


Photo de l'un de nos prototypes du côté de la face inférieure (où est soudé le PIC) du module SPC02.

Liste des composants du module SPC02

R1..... 27 kΩ 5 % boîtier 0805
R2..... 100 kΩ 5 % boîtier 0805
R3..... 1 kΩ 5 % boîtier 0805
R4..... 1 kΩ 5% boîtier 0805
R5..... 1,5 kΩ 5 % boîtier 0805
R6..... 1,5 kΩ 5 % boîtier 0805
R7..... 10 kΩ 5 % boîtier 0805
R8..... trimmer mono-tour 10 kΩ
R9..... 56 kΩ 5 % boîtier 0805
R10.... 150 Ω 5 % boîtier 0805
R11.... 22 kΩ 5 % boîtier 0805
R12.... 12 kΩ 5 % boîtier 0805
R13.... 47 kΩ 5 % boîtier 0805
R14.... 47 kΩ 5 % boîtier 0805
R15.... 27 kΩ 5 % boîtier 0805
R16.... 27 kΩ 5 % boîtier 0805
R17.... 2,2 kΩ 5 % boîtier 0805
R18.... 3,3 kΩ 5 % boîtier 0805
R19.... 27 kΩ 5 % boîtier 0805
R20.... 1 MΩ 5 % boîtier 0805
R21.... 10 kΩ 5 % boîtier 0805
R22.... 27 kΩ 5 % boîtier 0805
R23.... 150 kΩ 5 % boîtier 0805
R24.... 100 kΩ 5 % boîtier 0805
R25.... 100 kΩ 5 % boîtier 0805
R26.... 100 kΩ 5 % boîtier 0805
R27.... 56 kΩ 5 % boîtier 0805

C1..... 15 pF céramique boîtier 0805
C2..... 15 pF céramique boîtier 0805
C3..... 4.7 nF céramique boîtier 0805
C4..... 22 µF/6,3 V tantale
C5..... 10 µF/6,3 V tantale
C6..... 100 nF céramique boîtier 0805
C7..... 10 µF/6,3 V tantale
C8..... 1 nF céramique boîtier 0805
C9..... 100 nF céramique boîtier 0805

Continuez avec les condensateurs non polarisés puis les condensateurs au tantale (C4, C5, C7 et C17) polarisés en orientant correctement leur repère. Ensuite, soudez les 2 diodes en orientant, là aussi, correctement leurs repères et le quartz CMS de 10 MHz.

Veuillez noter que les composants CMS se trouvent sur les deux faces du circuit imprimé, n'oubliez pas de souder ceux qui se trouvent sur la face inférieure où se situe le microcontrôleur U1 (couche « bottom »).

Finissez en insérant le trimmer de réglage du volume et la barrette mâle qui établit la connexion avec l'extérieur et qui permet aussi la programmation du PIC (broches MCLR, PGD, PGC, Vcc et GND).

À ce stade, le circuit est prêt à fonctionner et ne nécessite pas d'étalonnage. Si vous n'avez pas fait d'erreurs et que le PIC est correctement programmé, il doit fonctionner immédiatement. Il suffit de connecter un microphone, un haut-parleur, des boutons aux broches correspondantes et d'alimenter le tout comme indiqué en figure 4. Les broches 4 (TX) et 5 (RX) peuvent être utilisées si vous souhaitez profiter des fonctions contrôlées par PC pour la gestion de la mémoire de l'enregistreur (voir le paragraphe « Fonctions gérées par le PC »).

Utilisation de l'enregistreur

Comme nous l'avons déjà mentionné, ce modèle d'enregistreur n'a pas toutes les fonctions du module SPC01.

C10..... 100 nF céramique boîtier 0805
C11 .... 2,2 nF céramique boîtier 0805
C12 .... 2,2 nF céramique boîtier 0805
C13 .... 1 nF céramique boîtier 0805
C14..... 1 nF céramique boîtier 0805
C15 .... 2,2 nF céramique boîtier 0805
C16..... 1 nF céramique boîtier 0805
C17..... 47 µF/6,3 V tantale
C18 .... 2,2 nF céramique boîtier 0805
C19 .... 100 nF céramique boîtier 0805
C20 .... 100 nF céramique boîtier 0805
C21..... 100 nF céramique boîtier 0805
C22 .... 100 nF céramique boîtier 0805
C23 .... 100 nF céramique boîtier 0805
C24..... 100 nF céramique boîtier 0805
C25 .... 100 nF céramique boîtier 0805
C26 .... 2,2 nF céramique boîtier 0805
LD1 .... LED rouge boîtier 0805
LD2 .... LED verte boîtier 0805
U1..... PIC18F2420 boîtier SOIC28
U2..... 24LC1025-I/SM boîtier SOIC8
U3..... 24LC1025-I/SM boîtier SOIC8
U4..... TDA7052A boîtier SOIC8
U5..... MCP6002 boîtier SOIC8
Q1..... quartz 10 MHz boîtier HC49/4H
Divers
Barrette mâle 12 pôles coudée

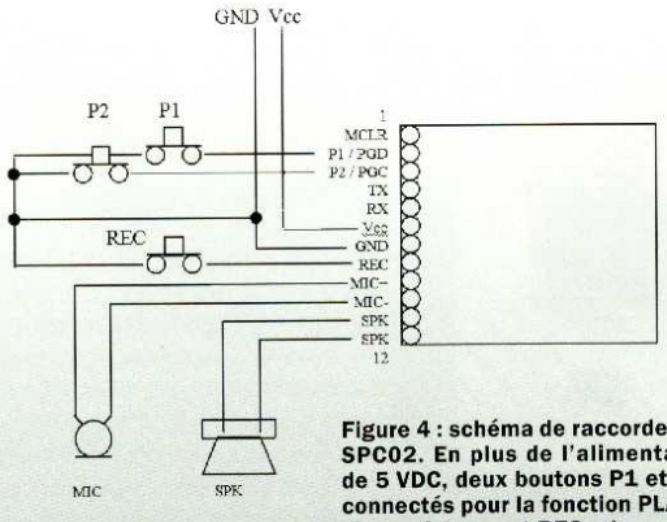


Figure 4 : schéma de raccordement du module SPC02. En plus de l'alimentation stabilisée de 5 VDC, deux boutons P1 et P2 doivent être connectés pour la fonction PLAY et la fonction d'enregistrement REC, plus un microphone et un haut-parleur.

Il n'est donc pas possible de modifier la fréquence d'échantillonnage, de **répéter les pistes**, de **sauter d'une piste à l'autre**. Vous avez la possibilité **d'enregistrer seulement deux messages** de n'importe quelle durée inférieure ou égale à la durée maximale d'enregistrement.

Pour enregistrer le premier message, appuyez simplement sur le bouton REC. La LED rouge s'allume pour indiquer qu'un enregistrement est en cours et, en même temps, la LED verte clignote, indiquant que l'enregistreur attend le signal de départ de l'enregistrement. Ce signal est donné en appuyant sur l'un des boutons P1 ou P2.

À partir de ce moment, tout ce qui est capté par le microphone est enregistré. Nous vous conseillons **de parler à une distance comprise entre 50 cm et 1 m du microphone**, avec une voix stable.

Pour terminer l'enregistrement, vous devez appuyer de nouveau sur le bouton REC, les LED s'éteignent.

Les boutons P1 et P2, en cours d'enregistrement, servent également à insérer une pause, c'est-à-dire en appuyant sur l'un d'entre eux, l'enregistrement est suspendu. Une nouvelle pression sur l'un d'entre eux effectue une reprise de l'enregistrement.

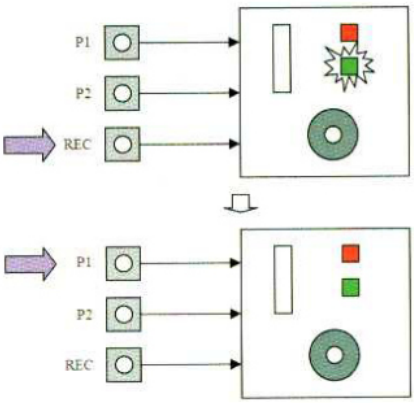
À ce stade, si vous appuyez sur le bouton P1, vous pouvez écouter ce qui est enregistré en position 1.

La lecture est indiquée par l'allumage de la LED verte uniquement.

Pendant la lecture, si vous appuyez sur l'un des deux boutons PLAY (P1 ou P2), l'enregistreur se met en pause.

Pour reprendre la lecture de la piste, appuyez de nouveau sur l'un des boutons P1 ou P2.

Figure 5 : appuyez sur le bouton REC pour passer en mode enregistrement. Le message est enregistré en appuyant sur P1 ou P2.

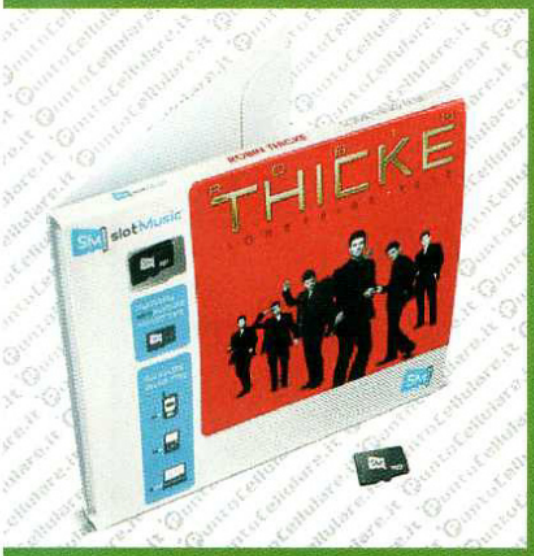


Pour enregistrer la deuxième piste, il suffit de répéter les mêmes opérations que celles décrites précédemment pour l'enregistrement de la première piste.

Le microcontrôleur s'occupe d'ajouter la nouvelle piste à celle déjà existante.

Si vous avez enregistré deux pistes et que vous essayez d'en enregistrer une autre, l'enregistreur effacera automatiquement toute la mémoire et commencera à enregistrer une nouvelle piste en position 1.

Le potentiomètre R8 permet de régler le volume pendant la lecture.



Après le vinyle et le CD, voici l'album numérique

Cela fait 30 ans, qu'avec étonnement et émerveillement, nous avons écouté pour la première fois la musique issue d'un compact disc laser. Pourtant, l'ère du CD semble être terminée. L'apparition du format audio compressé « mp3 » a révolutionné le monde de la musique. En effet, il existe des sites web qui proposent de la musique à la demande avec des milliers de titres disponibles sous la forme d'abonnement. Il est ainsi possible d'écouter de la musique sous différents périphériques allant du smartphone à la tablette ou encore au baladeur numérique.

Si bien qu'aujourd'hui, aller acheter un CD audio dans un magasin est devenu ringard.

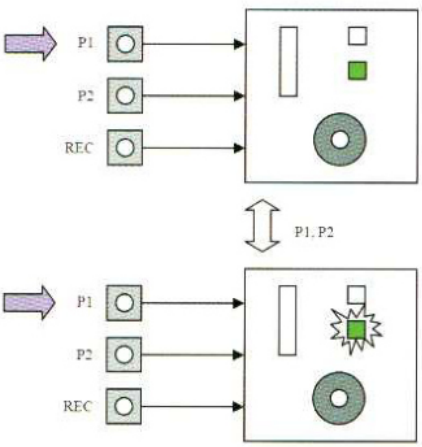
Cependant, la qualité sonore du vinyle est le nec plus ultra. Grâce à la précision des microsillons, l'onde sonore est reproduite analogiquement donc fidèlement, jusqu'à restituer parfois l'acoustique du lieu d'enregistrement (si l'ingénieur du son a fait un bon travail). Si l'on excepte un certain grésillement et une relative tendance à se dégrader à long terme, il propose la meilleure qualité d'écoute disponible aujourd'hui.

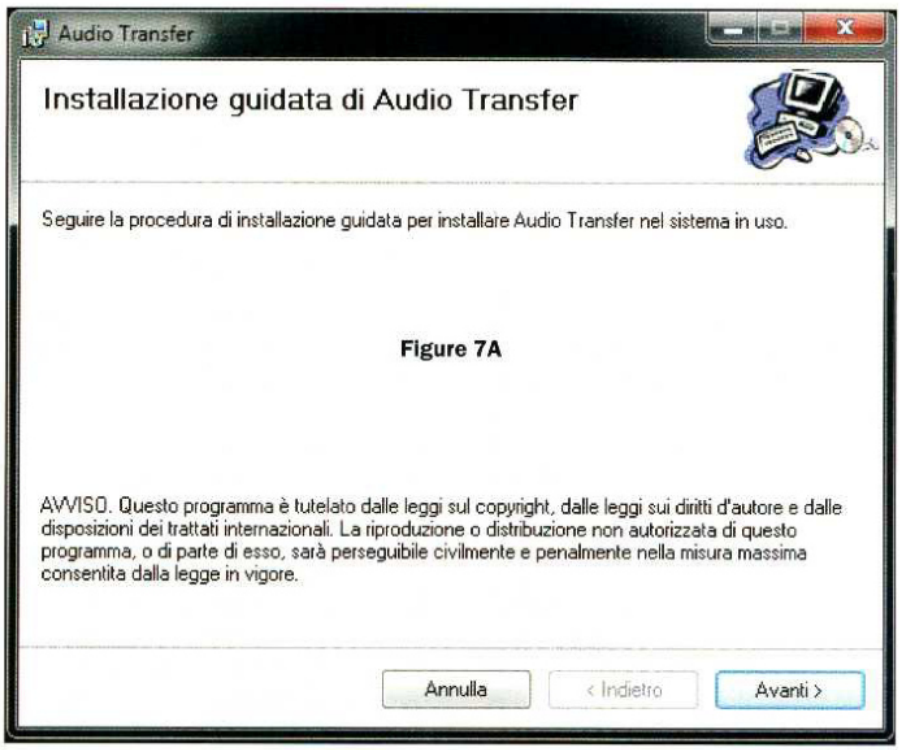
Le CD propose une qualité à mi-chemin entre le format MP3 et le disque vinyle. Il permet de retranscrire la musique selon un échantillonnage de 44,1KHz, ce qui signifie que le son est capturé 44 100 fois par seconde. Ce taux donne une précision proche de celle du disque vinyle. La plage dynamique du CD est même supérieure à celle du vinyle, avec une valeur de 100 décibels (contre 60 pour le disque microsillon). Cela signifie que le CD peut retranscrire des sons d'une plus grande amplitude, mais en réalité, cette valeur est peu exploitée car rares sont les systèmes audio qui permettent de tirer profit d'une telle dynamique. Bien que d'excellente qualité, le CD est condamné à n'être qu'une interprétation du son analogique en version numérique, forcément moins bon que l'original.

Le MP3 est celui qui offre la moins bonne qualité audio. Pour réduire la taille des fichiers afin d'être téléchargeables avec des connexions bas débit, et tenir sur des cartes SD ou des clés USB, le format MP3 compresse les données audio, qui sont ensuite décompressées lors de l'écoute.

Il en résulte des pertes, et un son dégradé si l'on y prête une oreille attentive (à condition d'avoir des haut-parleurs haute-fidélité).

Figure 6 : la lecture s'effectue simplement en appuyant sur le bouton correspondant à la piste à lire (P1 pour la première ou P2 pour la seconde). Les mêmes boutons sont utilisés pour mettre en pause la lecture.





Le format MP3 gomme les crêtes du signal audio et supprime la stéréo pour les fréquences inférieures à 300 Hz, ceci pour occuper moins de place. Nous pouvons dire que le MP3 est l'équivalent des cassettes audio des années 80.

L'idée de proposer des albums dans un format numérique compressé permet de les mettre sur des cartes mémoire et de les insérer dans un lecteur mp3 ou un smartphone afin d'écouter partout de la musique avec un casque. En effet, une minute de signal audio au format mp3 occupe environ 1 Mo de mémoire. Cependant, pour écouter de la musique sur un baladeur mp3, il faut soit acheter un CD et le convertir soit télécharger sur le web légalement des fichiers audio.

Il existe plusieurs plateformes web proposant de la musique en streaming ou téléchargement, tels que Spotify, Apple Music, Google Play Music, Amazon Music, Qobuz. Cependant, certaines proposent les fichiers audio avec des verrous numériques (DRM) qui empêchent la copie des chansons téléchargées, comme par exemple iTunes.

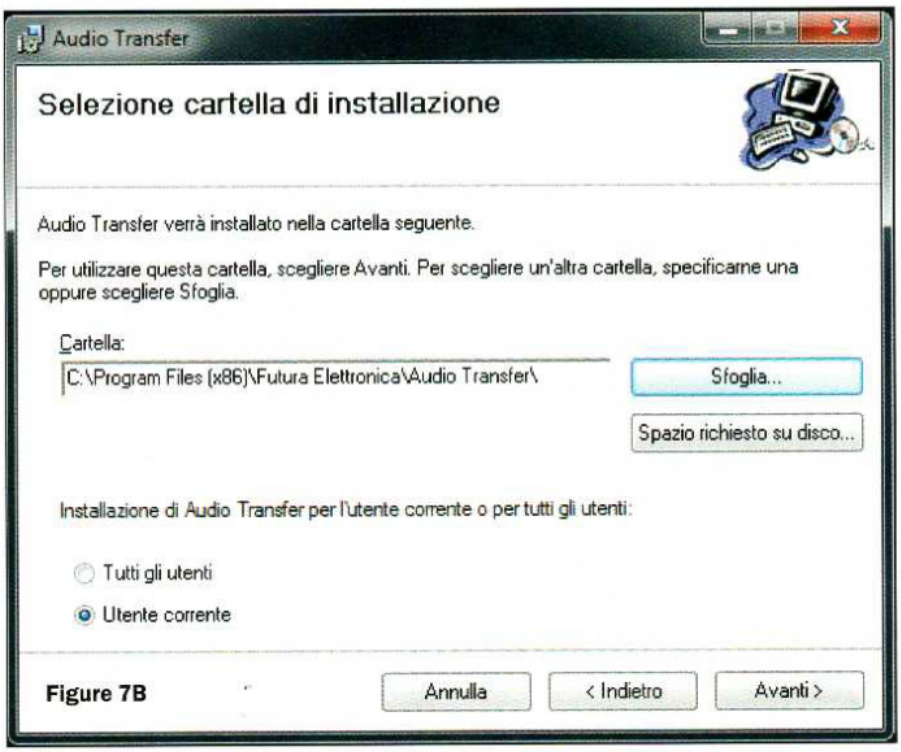


Figure 7B

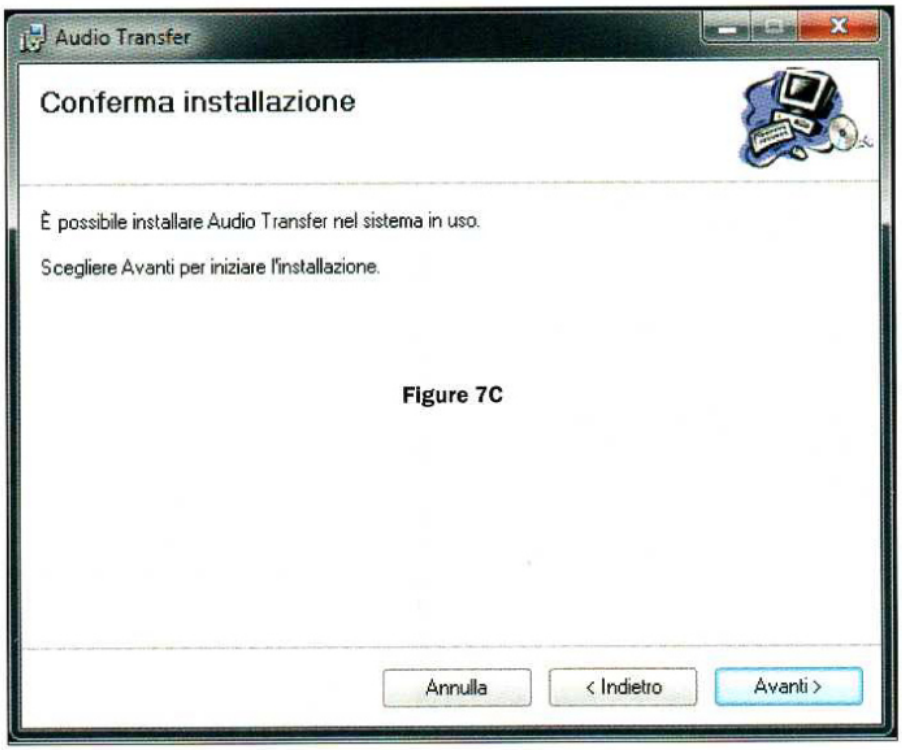


Figure 7C

Fonctions gérées par le PC

Nous allons décrire maintenant la manière d'utiliser le programme conçu pour lire et écrire dans les mémoires des deux modules d'enregistrement SPC01 et SPC02 à partir d'un PC. Toutes les procédures que nous allons décrire sont identiques pour les deux modules.

Le programme sur le PC permet, via le port série, de lire et d'écrire entièrement l'espace mémoire de l'enregistreur.

Cette fonction est utile, par exemple, pour créer une sorte de bibliothèque de messages pour votre module enregistreur, dans laquelle vous pouvez en choisir un de temps en temps.

Une autre possibilité du programme consiste à créer une série de modules qui reproduisent un même message, sans devoir enregistrer de nouveau le même message pour chaque module. Dans ce cas, une fois que vous avez enregistré le ou les messages sur l'un des modules, vous pouvez lire et sauvegarder le contenu de toute la mémoire dans un fichier sur le PC. Ensuite, vous pouvez copier le fichier entier dans la mémoire non programmée de tous les autres modules. Il s'agit de clones pour ainsi dire.

Mais examinons la procédure. Après avoir alimenté et connecté le module au PC via le port série, la première des choses à faire est d'installer le logiciel que vous pouvez télécharger sur notre site.

Une fois l'archive décompressée, vous devez obtenir un dossier nommé « **Audio Transfer** ».

Lancez l'installation en exécutant **en tant qu'administrateur** le fichier « **setup.exe** », qui se trouve dans le dossier. Au bout de quelques instants, la fenêtre de la figure 7A apparaît.

Cliquez sur le bouton « Avanti » (Suivant), une nouvelle fenêtre visible en figure 7B s'ouvre. Elle permet de choisir l'emplacement d'installation du programme sur le disque dur.

Par défaut, le chemin d'installation est :

C:\Program Files(x86)\Futura Elettronica\Audio Transfer\.

En cliquant sur le bouton « Sfoglia », vous pouvez choisir un autre emplacement. Par défaut, en bas de la fenêtre, l'expression « Utente corrente » est cochée.

Cela veut dire que l'exécution du programme n'est possible que par l'utilisateur courant. Si vous voulez donner l'accès à tous les utilisateurs de votre PC, cochez l'expression « Tutti gli utenti ».

Ensuite, cliquez sur le bouton « Avanti ». La fenêtre de la figure 7C apparaît vous indiquant que l'installation peut commencer.

Cliquez de nouveau sur le bouton « Avanti », le processus d'installation commence (voir la figure 7D).

À la fin de l'installation, vous obtenez la figure 7E. Cliquez alors sur « Chiudi » pour fermer la fenêtre.

Maintenant, le programme est correctement installé. Pour le lancer, vous devez aller dans le menu « Démarrer », puis « Tous les programmes ».

Cherchez le dossier « Futura Elettronica », à l'intérieur se trouve le dossier « Audio Transfer ».

Cliquez sur ce dernier pour avoir accès au programme « Audio Transfer » (voir la figure 7). Une fois le programme lancé, vous obtenez immédiatement une fenêtre semblable à celle de la figure 8.

Sélectionnez ensuite le port COM auquel vous avez connecté l'enregistreur et

Figure 7D

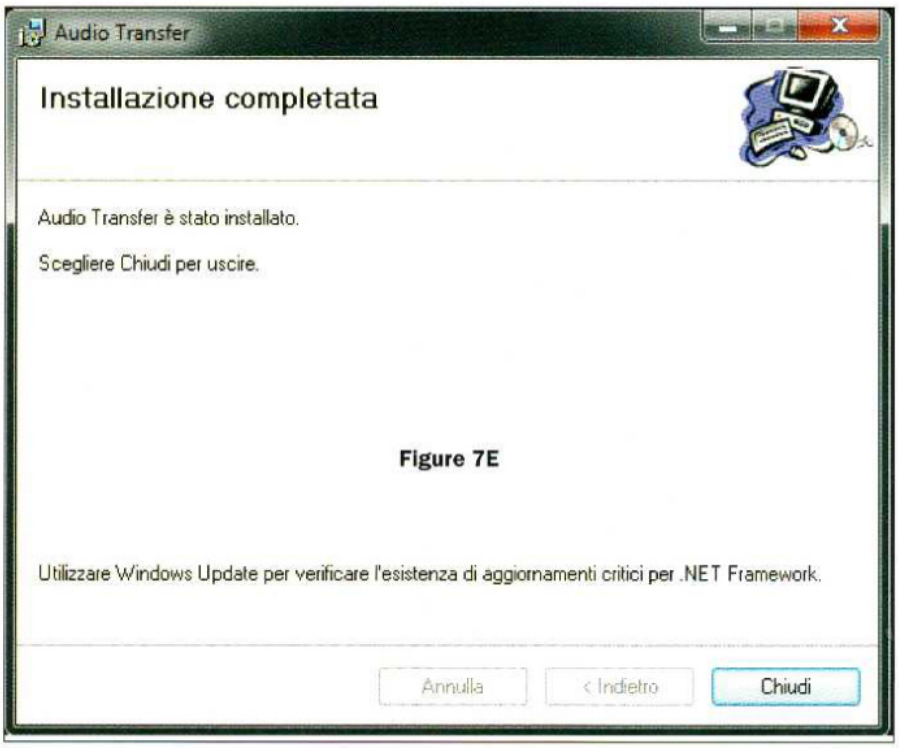


Figure 7E

réglez la vitesse de communication sur 115200 bauds.

Cliquez sur le bouton « Connetti » qui veut dire « Connexion ». Si le port sélectionné est correct, tous les autres boutons sont déverrouillés (ils n'apparaissent plus en grisé), sinon un message d'erreur apparaît. Choisissez alors un autre port de communication.

Dans le cas où vous voulez quitter le programme, nous vous conseillons toujours de déconnecter l'enregistreur en cliquant d'abord sur le bouton « Disconnetti » (Déconnexion), puis en fermant le programme. Après avoir ouvert le port COM, vous pouvez cliquer sur le bouton « Richiesta ID Modulo » (Demande d'identité du module).

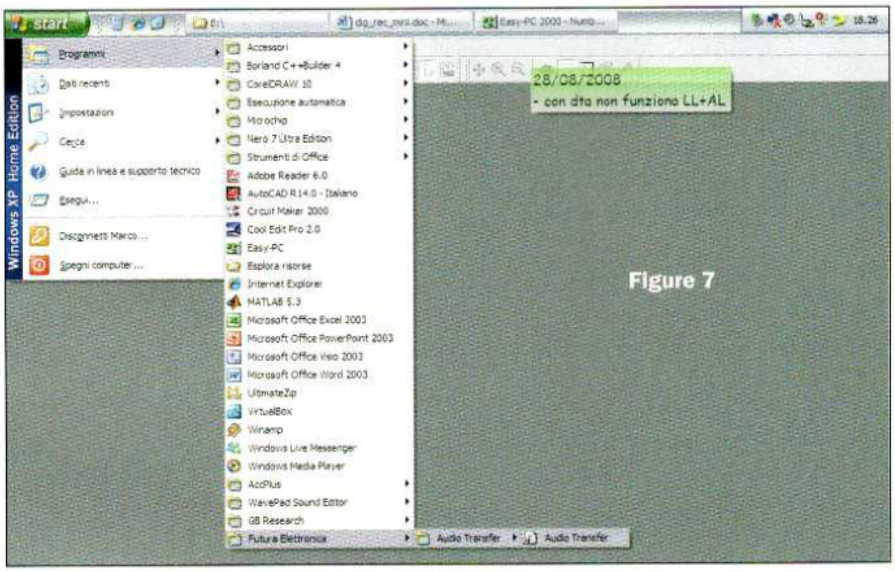


Figure 7

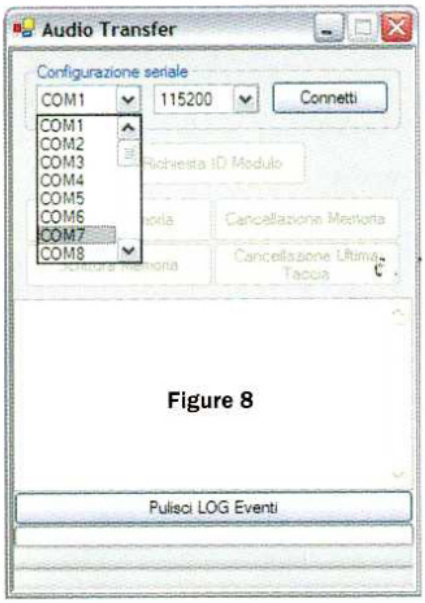


Figure 8

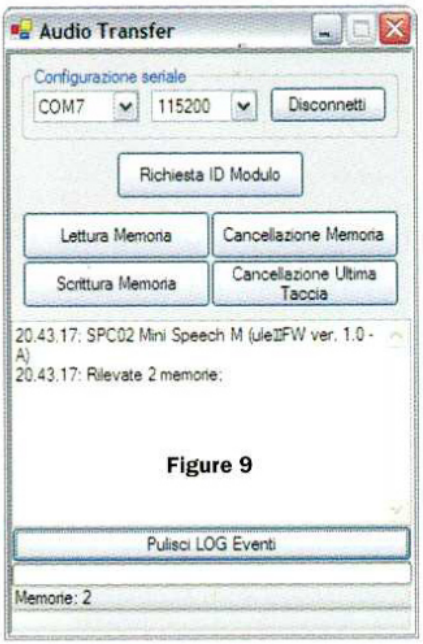


Figure 9

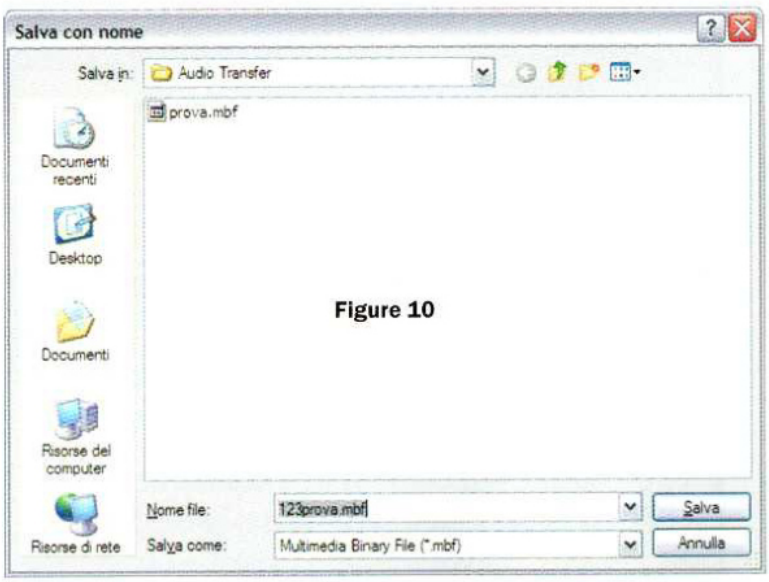


Figure 10

Cette commande vous permettra de savoir quel type de module vous avez connecté au PC (module SPC01 ou SPC02) et le nombre de mémoires installées sur le module (voir la figure 9).

Cette commande permet aussi de vérifier que la communication série fonctionne correctement.

À l'aide des deux boutons situés à droite de la fenêtre de la figure 9, vous pouvez choisir d'effacer complètement la mémoire (bouton « Cancellazione Memoria ») de l'enregistreur ou seulement la dernière piste enregistrée (bouton « Cancellazione Ultima Traccia »).

Dans les deux cas, un message vous avertira de l'exécution correcte de l'opération. Pendant l'exécution de la commande, les LED du module clignoteront simultanément.

Notez qu'une fois que vous avez supprimé une ou plusieurs pistes de la mémoire, elles ne pourront pas être récupérées.

Enfin, en cliquant de façon répétée sur la commande « Cancellazione Ultima Traccia », vous pouvez également effacer toute la mémoire, mais cela est plus long.

Le bouton « Lettura memoria » (Lecture Mémoire) vous permet de lire tout le contenu des mémoires installées sur l'enregistreur. En cliquant sur ce bouton, une boîte de dialogue apparaît vous permettant d'enregistrer le contenu de la mémoire du module dans un dossier de votre disque dur.

Il suffit ensuite de nommer le fichier (voir la figure 10). Une fois le fichier nommé, cliquez sur le bouton « Salva » (Enregistrer).

Le transfert des données vers le PC commence immédiatement. Pendant le transfert (qui dure environ 15 s pour chaque mémoire installée), vous verrez toujours la LED verte allumée, tandis que la LED rouge clignotera à chaque bloc de données envoyé. Une barre de progression en bas de la fenêtre indique visuellement le pourcentage de transmission effectué (voir la figure 11).

À la fin de l'opération, le message « Lettura OK » (Lecture OK) est affiché.

L'écriture dans la mémoire d'un fichier précédemment enregistré sur le PC s'effectue de la même manière. En premier lieu, vous devez cliquer sur le bouton « Scrittura memoria » (Ecriture Mémoire) et immédiatement une autre fenêtre de sélection apparaît.

Vous devrez alors choisir l'un des fichiers « .mbf » que vous avez enregistré précédemment (voir la figure 12). À ce stade, en cliquant sur le bouton « Apri » (Ouvrir), le fichier est téléchargé dans la mémoire de l'enregistreur.

Dans ce cas également, une barre d'état indique la progression de l'opération en cours, tout comme pour la lecture et, en même temps, sur l'enregistreur, la LED rouge reste allumée pendant que la verte clignote en fonction de chaque bloc de données reçu.

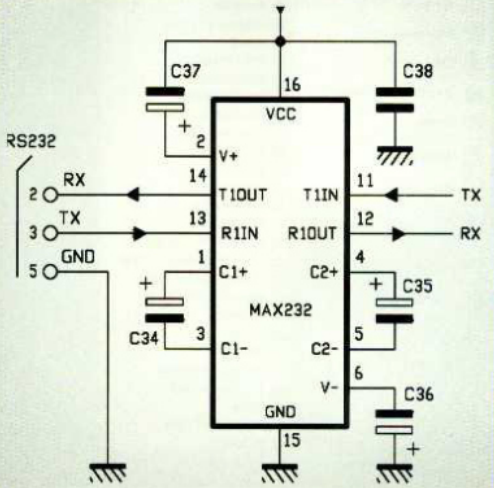
En conclusion, nous précisons que les fichiers lus par un enregistreur (quel que soit le module SPC01 ou SPC02) ayant deux mémoires ne peuvent pas être utilisés avec un enregistreur à trois ou quatre mémoires et vice versa.

De plus, si vous essayez d'écrire un fichier avec plus de deux pistes sur un module SPC02, vous ne pourrez pas accéder aux pistes situées après la seconde.

L'interface série

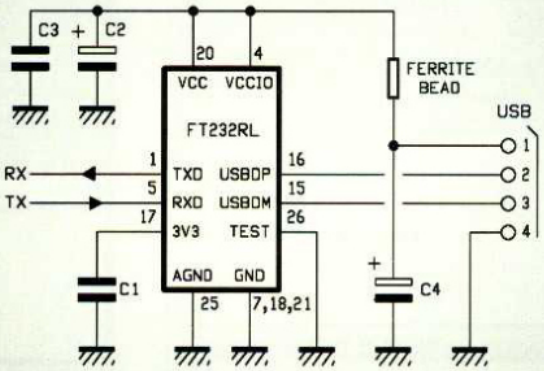
Pour utiliser les fonctions de contrôle des modules SPC01 et SPC02 à partir d'un PC, vous devez utiliser le port série. Comme les broches séries RX et TX du module ne sont pas directement compatibles avec les signaux d'un port standard d'un PC (ils sont au niveau TTL, alors que l'ordinateur communique au niveau RS232), vous devez utiliser un circuit de conversion. Vous pouvez choisir d'utiliser un port série RS232 ou un port USB.

Convertisseur de niveau RS232



Dans le premier cas, vous devrez utiliser un simple convertisseur de niveau comme le classique MAX232, tandis que si vous voulez utiliser un port USB, vous aurez besoin d'une interface plus complexe basée, par exemple, sur le circuit FTDI FT232R.

Convertisseur série - USB



Vous pouvez voir les schémas de ces deux interfaces ci-contre. Gardez à l'esprit que les broches RX et TX de l'interface sont directement connectées aux modules SPC01 ou SPC02. De la même manière, l'alimentation de l'interface est également directement reliée, vous pouvez la prendre à partir du module.

Figure 11

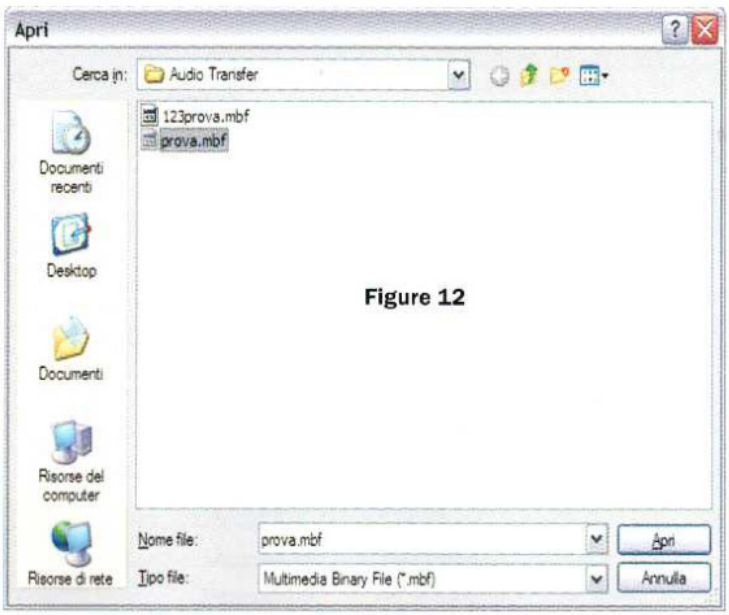


Figure 12

Récepteur 1 & 2 canaux 433,92 MHz
compatible MM53200, UM86409 et UM3759

de Alessandro Sottocornola

Dans cet article, nous vous proposons de réaliser un récepteur 433,92 MHz à un ou deux canaux compatible avec les codes des télécommandes MM53200, UM86409 et UM3759. Chaque récepteur peut être combiné jusqu'à 10 télécommandes. Le fonctionnement du récepteur pouvant être monostable ou bistable.

Ce récepteur est basé sur le **codage** de type **MM53200** qui est l'un des codes les plus anciens adoptés par les télécommandes radio génériques et ouvre-portes. Il existe de nombreux émetteurs disponibles sur le marché pour de telles applications. Au fil du temps, les encodeurs de type MM53200 ont été remplacés par le circuit UM3750 puis par le circuit UM86409 (tous deux fabriqués par UMC).

Pour cette raison, il est préférable d'avoir un récepteur à base de microcontrôleur qui peut s'adapter à ces codages, en l'apprenant automatiquement. Cela permet de réutiliser de vieilles télécommandes dormant au fond d'un tiroir et dont les récepteurs ne sont plus opérationnels.

C'est l'une des raisons pour lesquelles nous nous sommes aventurés dans la réalisation de ce récepteur en version monocanal et bicanal. L'autre raison est que le récepteur apprend le code directement à partir d'une transmission provenant de la télécommande, et donc cela permet de limiter la taille du récepteur car vous n'avez pas besoin d'un commutateur DIP à 12 voies (comme le réglage manuel l'exigeait). Ainsi, l'appairage entre le récepteur et l'émetteur s'effectue rapidement et sans erreur.

Les **deux récepteurs** (1 et 2 canaux) décrits dans cet article **sont compatibles** avec les **encodeurs MM53200**, **UM86409** et **UM3759**, mais également avec le dernier circuit intégré **Holtek HT12**.

Schéma électrique du récepteur monocanal

Commençons par l'analyse du **récepteur monocanal**, basé sur un microcontrôleur **PIC16F683** et sur un module récepteur hybride **AC-RX2**, accordé sur la fréquence de **433,92 MHz**. Le module U2 contient la partie radio du circuit, il s'agit d'un **récepteur à super-réaction Aurel AC-RX2**.

Ce dernier est équipé d'un amplificateur du signal d'antenne (sa sensibilité est de -106 dB), d'un étage de syntonisation accordé sur 433.92 MHz utilisant une modulation de type ASK avec condensateur ajustable, d'un filtre RF (le filtre sert à améliorer la sélectivité, qui dans le cas d'un récepteur à super-réaction n'est pas élevée) et d'un démodulateur d'amplitude.

Pour compléter la description du module, il dispose d'un comparateur de signal numérique (au niveau TTL) qui fournit sur la broche 14 les données décodées ainsi que d'un amplificateur LF du signal provenant du démodulateur AM.

Immédiatement après la mise sous tension, le **microcontrôleur initialise les entrées/sorties en configurant respectivement** les broches **GP0** et **GP1** en tant que **sorties** utilisées pour contrôler la LED de signalisation **LD2** (qui indique à la fois le mode de fonctionnement et l'état de la procédure d'auto-apprentissage) et le transistor NPN **T1** qui commande le relais auquel est reliée la charge à contrôler.

Ce dernier est activé par l'état de saturation du transistor qui est polarisé via sa base lorsque la broche GP1 est à un niveau logique haut (R1 limite le courant de base de T1 pour éviter d'endommager la jonction base-émetteur).

Chaque fois que le transistor conduit, en plus de la bobine du relais, il alimente la résistance R2 et la LED LD1 signalant que le relais est actif (contact collé).

En parallèle sur la bobine du relais est connecté la diode **D2** dont la fonction est de **court-circuiter la surtension inverse** générée par la bobine lorsque le transistor se bloque en interrompant le courant dans la bobine.

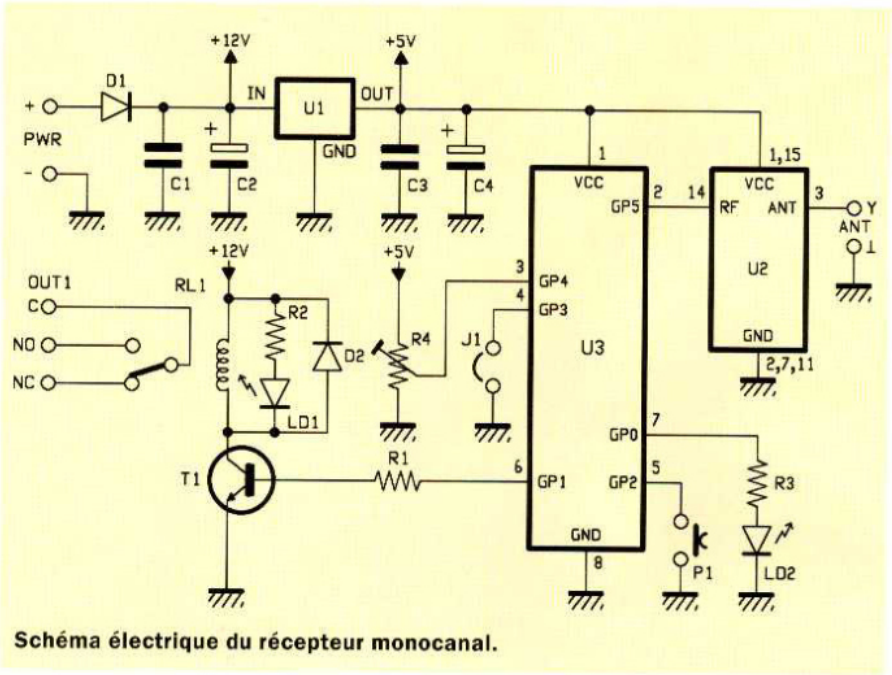


Schéma électrique du récepteur monocanal.

Sans la diode, la tension inverse apparaissant aux bornes de la bobine détruirait la jonction base-collecteur du transistor T1 en peu de temps.

Le PIC continue l'initialisation en configurant les broches **GP2**, **GP3** et **GP5 en entrées**. Respectivement, elles sont dédiées à la détection d'une pression sur le bouton poussoir **P1**, à l'état du cavalier **J1** et à la sortie des **données sortant** du récepteur **AC-RX2**. Pour les deux premières broches GP2 et GP3, les **résistances internes de tirage** (pull-up) sont **activées**.

La broche GP4 est configurée en entrée et affectée au convertisseur A/N. Elle est destinée pour de futures applications. Pour l'instant, elle n'a aucune incidence sur le fonctionnement du circuit.

Complétons la description du schéma par la partie alimentation. Le **montage est alimenté** via les points « **+PWR** » et « **-PWR** », c'est-à-dire via le bornier d'alimentation. La ligne positive passe à travers la diode **D1** qui **protège** le montage **contre les inversions de polarité** qui pourraient causer des dommages irréversibles.

Ensuite, la cathode de D1 est reliée à l'entrée (IN) du **régulateur U1** intégré à 3 broches. L'**entrée est filtrée** par les condensateurs **C1** et **C2**.

Ce dernier réduit les ondulations de la tension d'alimentation et C1 limite les perturbations HF et/ou les impulsions captées par les fils d'alimentation. De façon similaire, les condensateurs C3 et C4 effectuent les mêmes tâches en sortie du régulateur.

Examinons maintenant le fonctionnement du récepteur monocanal. Lorsque le montage est mis **sous tension**, le programme du PIC initialise les entrées/sorties, puis génère **5 clignotements** de la LED **LD2**, indiquant ainsi le bon démarrage et le fonctionnement normal.

Le circuit attend l'exécution d'une commande à distance. Dans la boucle principale du programme, le PIC vérifie en permanence si une variation de niveau a lieu sur la broche 4 (c'est-à-dire l'état du cavalier J1) et si une pression sur P1 a lieu.

Une **routine spécifique** vérifie périodiquement l'état de la broche 2 du

CARACTÉRISTIQUES TECHNIQUES :

- Nombre de sorties : 1 à 2 selon le modèle ;
- Mode de sortie : monostable ou bistable ;
- Tension d'alimentation : 12 VDC
- Consommation de courant : 40 mA ;
- Mémoire : 10 codes par canal ;
- Encodage : MM53200 / HT12.

microcontrôleur, c'est-à-dire **l'arrivée de données** provenant du récepteur AC-RX2. Lors d'une pression sur l'un des boutons de l'émetteur, le signal HF atteint l'antenne du module de réception AC-RX2, celui-ci démodule la trame comportant les données et les transmet, via sa broche 14, sur la broche 2 du microcontrôleur.

Le **microcontrôleur fonctionne comme un décodeur du code correspondant à la touche pressée** de la télécommande. Cependant, le programme permet également une **routine d'auto-apprentissage du code** de base sans configurer de DIP switch.

Cela fonctionne de la manière suivante. Le microcontrôleur acquiert les impulsions TTL, les place dans la RAM et les analyse avec une routine spéciale qui discerne tout d'abord, parmi les nombreux signaux détectés dans le milieu ambiant, celui qui est compatible avec le format de l'encodeur UM3750. Si cela est le cas, il vérifie si le code est l'un de ceux mémorisés pendant la phase d'auto-apprentissage, sinon il supprime les données dans la RAM et se prépare à une nouvelle analyse.

Nous étudierons plus tard comment appairer l'émetteur avec le circuit en utilisant l'auto-apprentissage.

Pour le moment, il suffit de savoir que le **programme permet d'apprendre un maximum de 10 codes**, même avec des émetteurs différents. En fait, tout code est mémorisé, c'est-à-dire les 12 bits qui le composent.

Examinons maintenant ce qui se passe si le signal reçu contient l'un des codes appris et mémorisés dans l'EEPROM du PIC. Dans ce cas, la routine de gestion du relais démarre, elle détermine les différentes actions selon le mode de fonctionnement choisi : monostable ou bistable.

Le mode est configuré grâce au cavalier **J1**, s'il est **ouvert** cela signifie que le relais fonctionne en mode **monostable**. Si le cavalier est **fermé**, cela correspond au mode **bistable**. **En mode bistable, le relais change d'état chaque fois que le microcontrôleur reconnaît un code valide.**

Tableau 1 : signification de l'état des LED du récepteur monocanal.

LED	Fonctionnement normal	Programmation
LD1	Activité de OUT1 : - allumée = RL1 activé - éteinte = RL1 au repos	-
LD2	lors du début ou de la fin de l'apprentissage ou de la suppression des codes, 5 clignotements pour le retour en mode normal.	s'allume pendant 2 secondes lors de l'entrée en programmation ; clignote lorsque le circuit a appris le code transmis ; fixe si l'apprentissage a échoué.

En mode monostable (ou impulsionnel), **le relais reste enclenché tant qu'un code valide est reçu et revient au repos lorsque le bouton de la télécommande correspondant au code est relâché**, c'est-à-dire dès la fin de la réception du signal d'activation. En résumé, l'état du relais suit l'état du bouton de la télécommande.

Si en même temps la transmission s'arrête mais que le code d'une autre télécommande appariée (dont le code est l'un des 10 appris) est reçu avant le délai d'expiration paramétré dans le programme, le microcontrôleur considère que la transmission n'a jamais cessé.

Si le code d'une autre télécommande arrive lorsque le relais est actif, car le récepteur est en train de recevoir une transmission valide d'un autre émetteur, le système ne fonctionne plus car le second signal interfère avec le premier. Le récepteur tente de démoduler des données non valides.

Il en résulte que le microcontrôleur croit que le signal valide a cessé d'être transmis et provoque la mise au repos du relais.

Notez que **le mode de fonctionnement du relais peut être modifié lorsque le montage est sous tension**, c'est-à-dire qu'il n'est pas nécessaire d'éteindre et de rallumer le montage car l'état du cavalier J1 est constamment détecté.

Procédure d'apprentissage du récepteur monocanal

Nous venons d'examiner le comportement du récepteur lorsqu'une commande arrive, examinons maintenant l'apprentissage des codes. Cela peut être fait à tout moment en appuyant sur le bouton P1 et en le maintenant enfoncé jusqu'à ce que la LED verte (LD2) s'allume. La phase d'auto-apprentissage est alors démarrée.

À ce stade, il est nécessaire de transmettre un code pour que le récepteur l'apprenne. Si la télécommande a plusieurs canaux, les codes peuvent tous être appris.

Par conséquent, vous devez **appuyer sur l'un des boutons** de la télécommande que vous désirez **appairer**, et attendre que la LED **LD2 clignote**, indiquant ainsi que l'apprentissage a réussi.

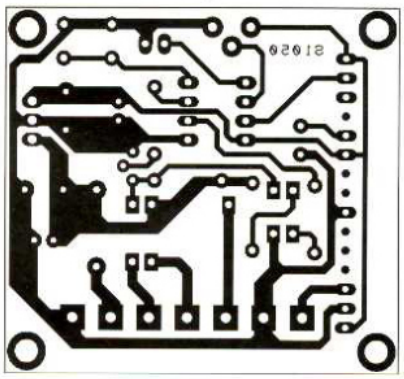
Si la **LED reste allumée**, cela signifie soit que la **mémoire est pleine ou que le code transmis n'est pas valide** (le code ne correspond pas à un encodeur de type MM53200, UM3750, UM86409 ou HT12).

Veuillez prêter une attention particulière à la façon dont fonctionne le programme et à la façon dont l'EEPROM du microcontrôleur est gérée. Le circuit peut mémoriser tous les codes de l'émetteur sans exception, sauf celui dont tous les bits sont à 1 (tous les DIP switch de la télécommande sont positionnés sur ON). Ainsi, 4095 combinaisons sur les 4096 possibles peuvent être utilisées au niveau de l'émetteur.

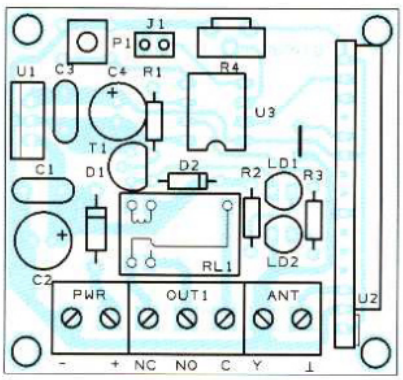
Il n'est pas possible de supprimer un seul code. La procédure prévoit que l**a mémoire est entièrement effacée et non en partie**. Pour effacer l'EEPROM, c'est-à-dire supprimer les codes appris, le **montage doit être mis hors tension puis le bouton P1 doit être pressé et enfin le montage doit être remis sous tension**. Le bouton doit être relâché dès que la LED verte reste allumée pendant 2 secondes, indiquant que la mémoire a été effacée.

À ce stade, le bouton peut être relâché, la LED verte clignote alors 5 fois pour indiquer la sortie de la procédure

Plan de montage du récepteur monocanal



Circuit imprimé à l'échelle 1 : 1 côté soudures du récepteur monocanal.



Plan de câblage des composants du récepteur monocanal.

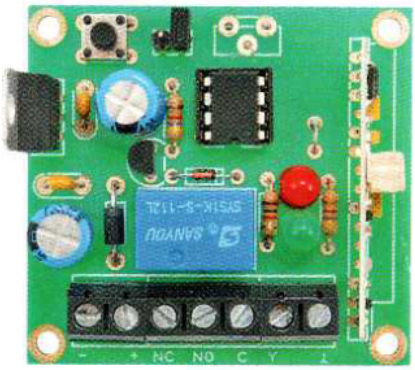


Photo de l'un de nos prototypes du récepteur monocanal.

Liste des composants du récepteur monocanal

R1..... 4,7 kΩ
R2..... 1 kΩ
R3..... 470 Ω
R4..... non montée
C1..... 100 nF multicouche
C2..... 100 µF/35 V électrolytique
C3..... 100 nF multicouche
C4..... 10 µF/35 V électrolytique

U1..... 7805
U2..... AC-RX2
U3..... PIC12F683
D1..... 1N4007
D2..... 1N4148
LD1.... LED 5 mm rouge
LD2.... LED 5 mm verte
T1 ..... BC547
P1..... bouton poussoir
RL1.... relais 12 V/1 contact
(Sanyou Relays SYS-S-112L)

Divers

Bornier 2 pôles (x2)
Bornier 3 pôles
Support circuit intégré 2 x 4 broches
Barrette mâle 2 pôles
Cavalier

d'effacement et le fonctionnement normal du récepteur (voir le tableau 1).

Schéma électrique du récepteur à 2 canaux

Maintenant, examinons le schéma électrique du récepteur à 2 canaux, sans répéter ce qui a déjà été expliqué pour le récepteur monocanal. Nous remarquons que l'étage d'alimentation est identique à celui du récepteur monocanal. Nous retrouvons le même étage de réception autour du module AC-RX2. Par contre, le microcontrôleur est différent, puisqu'il s'agit d'un **PIC16F688**, dont le choix a été dicté par un plus grand nombre d'entrées/sorties disponibles que pour le PIC16F683 utilisé dans la version monocanale.

Bien évidemment, nous avons deux étages à relais, dont le fonctionnement est identique à celui du récepteur monocanal, avec autant de diodes de protection en parallèle sur les bobines et autant de transistors (T1 et T2, un pour chaque relais).

Nous trouvons également en supplément un DIP switch à 2 voies qui sert à configurer le mode de fonctionnement des relais des deux canaux (monostable ou bistable), et deux potentiomètres, qui, comme mentionné dans la description du récepteur monocanal, sont destinés à des développements futurs et ne sont pas utilisés.

Immédiatement après la mise sous tension, signalée par une séquence de 5 clignotements de la LED verte pour indiquer le bon démarrage, le microcontrôleur initialise les entrées/sorties en configurant la broche RA4 en entrée pour l'acquisition des données provenant du module AC-RX2.

Les broches RB4 et RB5 sont configurées en entrées avec leur résistances internes de pull-up activées, elles permettent de lire l'état des DIP switch.

Les broches RB6 et RB7, configurées aussi en entrées avec leurs résistances internes de pull-up activées, permettent de lire l'état des boutons P1 (démarre l'auto-apprentissage pour le canal 1) et P2 (démarre l'auto-apprentissage pour le canal 2).

Le microcontrôleur configure aussi, pendant la phase d'initialisation, les broches RB1, RB2 et RB3 en tant que sorties. Respectivement, elles sont dédiées à la commande de la LED de signalisation LD3 (verte, qui indique les différentes étapes de la procédure d'auto-apprentissage et du fonctionnement normal), à la commande de la LED LD1 qui signale l'activité du canal 1, et enfin à la commande de LD2 qui signale l'activité du canal 2.

Concernant les deux relais, le fonctionnement est analogue à celui du récepteur monocanal, puisque les deux étages sont identiques. Le transistor T1, qui pilote le relais RL2, entre en saturation lorsque la broche RA6 est

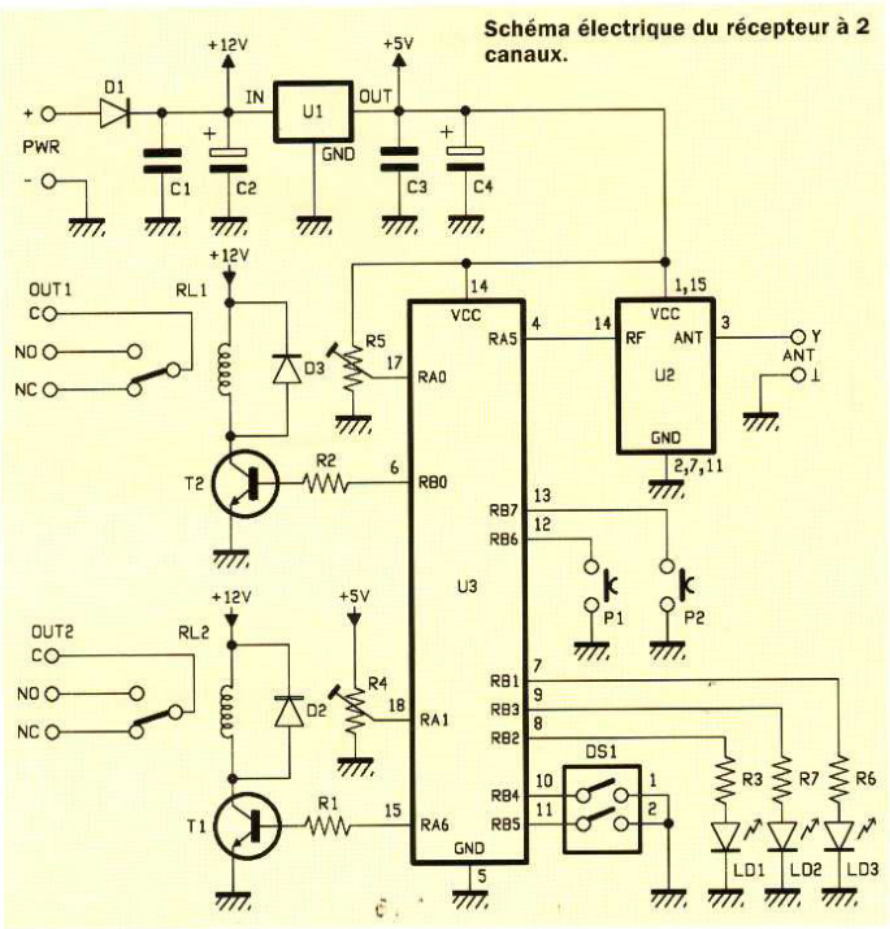


Schéma électrique du récepteur à 2 canaux.

à un niveau logique haut, tandis que transistor T2 est saturé (conducteur) lorsque la broche RB0 est à un niveau logique haut. Les diodes D2 et D3 protègent les transistors T1 et T2 contre les surtensions inverses provoquées par les bobines des relais.

Le DIP switch **DS1 permet de définir le mode de fonctionnement souhaité pour chaque canal**, plus exactement, le DIP1 permet le réglage du mode de fonctionnement de la sortie OUT1 (RL1) et le DIP2 de la sortie OUT2 (RL2).

Le DIP fermé correspond à un fonctionnement bistable, tandis que le DIP ouvert correspond à un fonctionnement monostable. Comme dans le cas du récepteur monocanal, en mode monostable, la sortie est activée en appuyant sur le bouton de l'émetteur et revient au repos lorsque le bouton est relâché.

Les trimmers R4 et R5 ne sont pas montés, ils serviront pour des applications futures. Les deux LED rouges signalent l'état des sorties pendant le fonctionnement.

LD1 s'allume si le relais de la sortie OUT1 est activé et LD2 s'allume si le relais de la sortie OUT2 est activé.

Procédure d'apprentissage du récepteur à 2 canaux

Pour appairer les émetteurs au récepteur à 2 canaux, vous devez procéder de la manière suivante. Appuyez et maintenez pressé le bouton relatif au canal à mémoriser (P1 pour la sortie OUT1 et P2 pour la sortie OUT2). La LED rouge relative à la sortie (LD1 pour OUT1 et LD2 pour OUT2) s'allume pour indiquer que vous êtes en mode auto-apprentissage.

Une fois cela effectué, appuyez sur le bouton de la télécommande que vous souhaitez appairer et attendez que la LED clignote, indiquant que l'apprentissage a réussi.

Si la LED reste allumée de manière fixe, cela signifie que la mémoire est pleine ou que le code transmis est invalide.

Notez qu'il n'est pas possible d'apprendre le code correspondant à tous les DIP switch de l'émetteur positionnés sur ON, dans le sens où le code correspondant à « 111111111111 » n'est pas autorisé à cause de la manière dont la gestion de la mémoire EEPROM est effectuée par le microcontrôleur. Il en résulte que, comme pour le récepteur monocanal, seules 4095 combinaisons sont disponibles au lieu des 4096 possibles.

En ce qui concerne l'effacement des codes, **il n'est pas possible de supprimer un seul code d'un canal**, mais **il est possible de supprimer les codes relatifs à un seul canal**.

Pour effacer la mémoire d'un canal, allumez le circuit en appuyant sur le bouton relatif au canal dont vous souhaitez supprimer les codes et maintenez-le pressé.

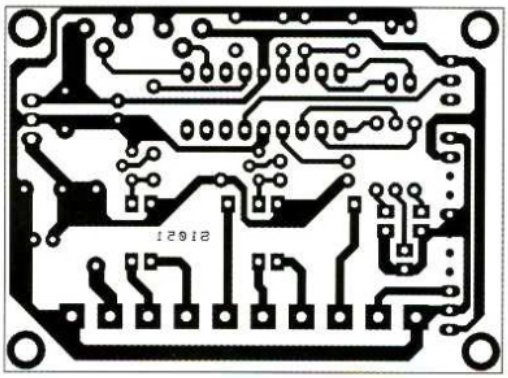
Soit le bouton P1 pour le canal 1 (OUT1), soit le bouton P2 pour le canal 2 (OUT2).

Relâchez le bouton lorsque la LED verte (LD3) reste allumée pendant 2 secondes, indiquant que la mémoire associée au canal sélectionné a été effacée.

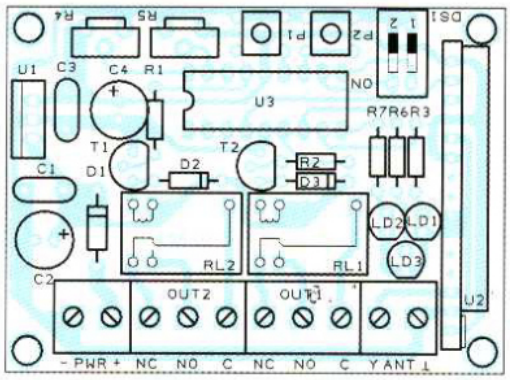
Tableau 2 : signification de l'état des LED du récepteur à 2 canaux.

LED	Fonctionnement normal	Programmation
LD1	Activité de OUT1 : - allumée = RL1 activé - éteinte = RL1 au repos	-
LD2	Activité de OUT2 : - allumée = RL2 activé - éteinte = RL2 au repos	-
LD3	lors du début ou de la fin de l'apprentissage ou de la suppression des codes, 5 clignotements pour le retour en mode normal.	s'allume pendant 2 secondes lors de l'entrée en programmation ; clignote lorsque le circuit a appris le code transmis ; fixe si l'apprentissage a échoué.

Plan de montage du récepteur à 2 canaux



Circuit imprimé à l'échelle 1 : 1 côté soudures du récepteur à 2 canaux.



Plan de câblage des composants du récepteur à 2 canaux.

Liste des composants du récepteur à 2 canaux

R1..... 4,7 kΩ
R2..... 4,7 kΩ
R3..... 470 Ω
R4..... non montée
R5..... non montée
R6..... 470 Ω
R7..... 470 Ω

C1..... 100 nF multicouche
C2..... 100 µF/35 V électrolytique
C3..... 100 nF multicouche
C4..... 100 µF/35 V électrolytique
U1..... 7805
U2..... AC-RX2
U3..... PIC16F88
D1..... 1N4007
D2..... 1N4148
D3..... 1N4148

T1 ..... BC547
T2 ..... BC547
LD1.... LED 3 mm rouge
LD2.... LED 3 mm rouge
LD3.... LED 3 mm verte
P1..... microswitch
P2..... microswitch
RL1.... relais 12 V/1 contact (Sanyou Relays SYS-S-112L)
RL2.... relais 12 V/1 contact (Sanyou Relays SYS-S-112L)
DS1 ... DIP switch 2 voies

Divers :

Bornier 2 pôles (x2)
Bornier 3 pôles (x2)
Support circuit intégré 2 x 9 broches

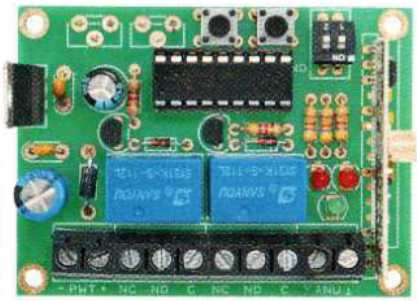


Photo de l'un de nos prototypes du récepteur à 2 canaux.

La LED verte clignote 5 fois pour indiquer la sortie de la procédure d'effacement et le démarrage en mode normal du récepteur (voir le tableau 2).

Réalisation pratique

Pour chacun des récepteurs, nous mettons en téléchargement sur notre site dans le sommaire détaillé de la revue les typons des circuits imprimés ainsi que les fichiers «.hex » de programmation des microcontrôleurs.

Une fois les circuits imprimés gravés et percés, comme d'habitude commencez par souder les composants ayant un profil bas, c'est-à-dire les résistances, les diodes en respectant leur orientation, le support du PIC, le DIP switch (présent uniquement sur la version à 2 canaux), les condensateurs non polarisés puis les condensateurs électrolytiques en respectant l'orientation. Continuez avec les transistors en positionnant correctement leur méplat, de même pour les LED.

Enfin soudez le relais, le régulateur de tension, les borniers et les récepteurs HF AC-RX2 en respectant scrupuleusement leur orientation sous peine de destruction immédiate (le condensateur ajustable présent sur le AC-RX2 doit être dirigé vers l'extérieur du circuit, ceci est valable pour les 2 montages). Pour l'orientation correcte des éléments polarisés (transistors, diodes, condensateurs, régulateurs et microcontrôleurs), reportez-vous aux plans de câblages respectifs dans ces pages.

Une fois l'assemblage terminé, il vous suffit d'insérer le microcontrôleur en prenant soin de l'orientation et en l'ayant au préalable programmé avec le firmware correspondant.

N'oubliez pas d'équiper les récepteurs d'une antenne adéquate accordée sur 434 MHz. La connexion doit, de préférence, être réalisée avec un câble coaxial RG59 dont le conducteur central doit être relié au point Y du bornier ANT, tandis que la tresse de blindage doit être reliée à la masse du bornier.

Vous pouvez aussi **fabriquer l'antenne avec un simple morceau de fil de cuivre de 17 cm de longueur** (pour avoir une antenne d'un quart d'onde) ou de **35 cm de longueur** (pour avoir une antenne de demi-onde) en le connectant au point Y du bornier ANT. Ceci est valable pour les deux récepteurs (1 ou 2 canaux).

Pour l'alimentation des circuits, vous pouvez utiliser une alimentation stabilisée de **12 VDC** à **16 VDC**, avec un courant continu minimal de 70 mA pour le récepteur monocanal et de 100 mA pour le récepteur à 2 canaux.