Fait main

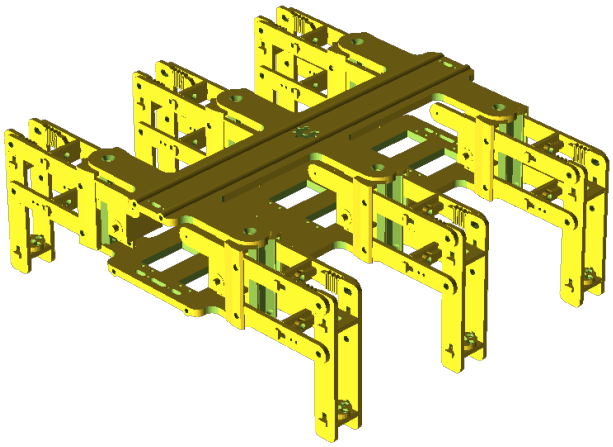
Électronique Informatique Art Cuisine Écologie

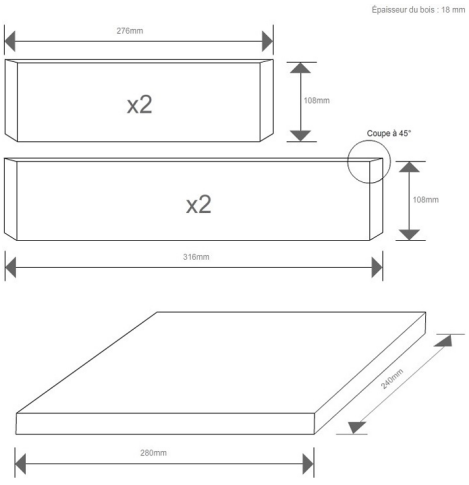
Numéro 3
Août 2013

Fait main

~~PDF~~ ■ Papier ■ Web

faitmain.org





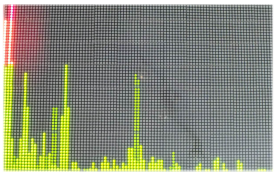


Table des matières

1. Edito Volume 3

Voici le troisième numéro de Fait Main. En terme de quantité d'articles il est légèrement plus court que les deux précédents numéros, mais c'est pour l'instant mon numéro préféré pour plusieurs raisons.

Tout d'abord, tous les articles sont des articles originaux écrits exclusivement par des auteurs francophones. Parfois repris de leurs blogs et adaptés, mais nous n'avons pas eu besoin de traduire ou reprendre des articlesen anglais, ou demander la permission de reproduire un article existant.

Ensuite, la variété des sujets est exactement ce que l'on souhaite, c'est-à-dire ne pas étouffer le magazine dans un contenu qui porte purement sur de l'électronique ou de l'informatique — et s'ouvrir au vaste monde du DIY.

Enfin, le site a une CSS plus propre et le process de licence de contenu est plus clair et simple : chaque auteur a obligation de choisir une licence, qui est affiché en haut de l'article. Le site en lui-même (la garniture) passe en By-SA et le code reste en Apache 2.

Sinon, dans les petits changements :

La partie forum va disparaître — personne ne s'en sert vraiment et c'est une vrai galère à gérer pour les spams.

On a ajouté un lien **modifier** sur les articles, qui atterrit dans Github sur le source. Le gros intérêt c'est que Github non seulement fourni un éditeur en ligne, mais permet aussi aux users de faire une modif, voir une preview et faire une pull request — tout en 3 clicks. C'est magique pour le néophite qui a envie de corriger une typo et ne souhaite pas installer git etc. Pour ceux qui n'ont pas de compte github on accepte les patchs bien sûr.

Le lien **Agenda** dans le menu est une carte des FabLabs et HackerSpace, via un tag OpenStreetMap. Il en existe pleins sur le net, mais le petit plus, c'est le calendrier ou chacun peut ajouter des évènements : conférences, salons, hackathon, etc.

Seul regret : le PDF est encore horriblement moche — nous n'avons pas eu le temps de terminer l'intégration Scribus. Mais Romain & Raphaël sont sur le coup pour le 4.

Bonne lecture et bon été à tous.

— Tarek

Équipe

Le projet *FaitMain* est monté par Tarek Ziadé_ mais est surtout possible grâce aux créateurs d'articles et aux relecteurs.

Ont participé à ce numéro :

— Auteurs : Vincent Becker, David Larlet, Fabien Batteix, Sébastien Riguet, Charles Rincheval, Tarek Ziadé, Alexis Métaireau, Frédéric Sureau, Nerick ;
— Relecteurs : Alexandre Garreau, Cédric Bonhomme, Jean Thomas, Florent Gibaux, Romain Dubos, et tous ceux que nous avons oublié !
— Code & design : Raphael Bastide, Tarek Ziadé, Alexis Métaireau, Alice Lieuter.

2. Horloge Arducomtoise

date 2013-08-01
category électronique,informatique
level vulgarisation, moyen

author Vincent Becker

licence By-Sa-3.0

FIGURE 2.1 – ArduComtoise, l'horloge comtoise la plus précise du monde

Je possède dans ma cuisine une très belle horloge comtoise héritée de mes grands-parents. Si la précision de la mécanique est déjà assez épatante au départ, j'ai chargé une Arduino de l'améliorer encore. Au vu de ses caractéristiques techniques (balancier sur l'avant typique de la deuxième moitié du XIXe siècle mais échappement à couronne et non à ancre typique de la première moitié), cette horloge doit dater du milieu du XIXe siècle.

Le cadran mentionne Munster, en Alsace, comme ville d'origine, mais il s'agit bien d'une comtoise. Les mouvements étaient fabriqués à Morez près de Pontarlier dans le Haut Doubs puis expédiés nus dans toute l'Europe. Les façades, caisses et poids étaient ensuite produits sur le lieu de destination, portant mention de l'horloger local.

Elle possède un mouvement situé assez haut (environ 2m10) avec comme garde-temps un balancier d'environ 1m20. Deux poids assurent la réserve d'énergie, l'un pour le mouvement, l'autre pour la sonnerie. Elle sonne les heures pleine avec répétition et les demi-heures. Il faut la remonter une fois par semaine.

Le balancier est de type lenticulaire, monté sur un ruban métallique simple avec un petit jeton de réglage permettant de faire varier la hauteur, et donc la période, du pendule. Ce type de monture a l'inconvénient, du fait de la dilatation, de rendre la période du balancier sensible à la température, problème qui sera corrigé sur les modèles plus tardifs avec des montures plus complexes auto-compensées. Dans des conditions de température bien stables, la tenue de l'heure est très bonne, avec une dérive de quelques secondes par jour.

Mais on peut faire mieux ! C'est une Arduino qui va s'en charger.

ArduComtoise

<iframe width="800" height="450" src="//www.youtube.com/embed/06flAYL-30Q" frameborder="0" allowfullscreen></iframe>

Pas question, dans ce projet, de toucher à la mécanique très sensible de l'horloge elle-même. Tout se passe sans contact ou presque.

Un module Chronodot sert de garde-temps, avec une dérive typique de moins de 2s par mois. La comtoise est réglée pour avancer légèrement (environ 30 secondes par jour). Il suffit dès lors de retenir le balancier le temps nécessaire grâce à un servo une fois par 24h pour conserver la précision de l'horloge indéfiniment. Cela implique de déterminer précisément quand l'horloge a fait deux tours complets (soit 24h).

Première approche : et Arduino comptait, comptait...

Dans un premier temps, j'ai voulu faire en sorte que le système soit totalement invisible de l'extérieur : pas question donc de mettre un feedback sur les aiguilles. Je suis parti sur une première solution consistant à compter le nombre de passages de balancier effectués par l'horloge en 24h, le tout sans contact.

En bas de la caisse de l'horloge, un capteur optique (un couple émetteur/récepteur IR) compte les passages du balancier. Ce dernier effectue 80 350 passages en 24h, soit une période d'environ 1,07 secondes.

Lorsque les 80 350 passages de balancier ont été comptés et donc que 24h ont passé sur l'horloge, l'Arduino fait pivoter un servo de 90° pour bloquer le balancier. Comme l'horloge avance, l'Arduino maintient le balancier bloqué jusqu'à ce que 24h aient passé sur le Chronodot puis le relâche en faisant pivoter le servo en sens inverse.

Il est important de bloquer le balancier en bout de course afin qu'il reprenne suffisamment d'élan pour repartir. Un deuxième capteur se charge de détecter ce moment. On ne peut pas utiliser ce même capteur pour compter car lorsque le balancier vient d'être relâché, sa course est trop courte pour qu'il revienne jusqu'à ce capteur ; on raterait donc des passages et toute la précision du montage repose sur un comptage précis.

Il arrive, sans que j'arrive à savoir pourquoi, que le détecteur rate un passage. C'est rare, mais le bon comptage est vraiment critique pour le fonctionnement du système. Le temps écoulé entre deux passages est donc calculé et, s'il est supérieur à 2 secondes, c'est qu'on en a raté un et un passage est rajouté « artificiellement ».

Limites du système

Ce système marche parfaitement entre deux remontages. Mais il se heurte à un problème insoluble : lors du remontage, l'horloge recule aléatoirement de 10 à 15 secondes du fait des divers jeux internes de la mécanique.

On pourrait intégrer un temps moyen dans la calibration pour compenser, mais c'est loin d'être idéal. J'ai donc changé d'approche.

Deuxième approche : feedback

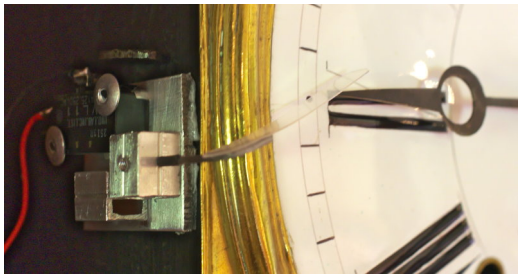


FIGURE 2.2 – Le capteur de feedback

J'ai dû me résoudre à employer la solution rejetée en première instance mais qui semble la seule parfaitement fiable, à savoir un feedback sur les aiguilles.

Celui-ci est constitué d'un micro-switch actionné par une came elle-même actionnée par le passage de la grande aiguille qui appuie sur une palette en plastique transparent rigide (un morceau de blister d'emballage de piles) pour être aussi invisible que possible.

Après tests, le micro-switch se déclenche à 45 minutes et 30 secondes de chaque heure, plus ou moins 3 secondes compte tenu de l'élasticité des divers éléments (palette transparente et micro-switch lui-même). Ce n'est pas gênant, car ce petit décalage n'est par définition pas cumulatif.

Une fois toutes les 24h (un peu avant 19h45), l'Arduino passe en mode réglage, c'est à dire qu'elle va bloquer le balancier dès que le micro-switch va se déclencher (donc quand l'horloge indiquera 19 :45 :30) pour le relâcher au bon moment. L'horloge étant réglée pour avancer de 30s par jour, le système relâche le balancier avec environ 15 secondes de retard sur l'heure juste : en tenant compte de toutes les incertitudes, l'heure indiquée par l'horloge est donc indéfiniment décalée au maximum de 20s par rapport à l'heure du Chronodot.

Mais revoyons la scène au ralenti.

En début de script, l'heure de référence (qui sert au recalage) est définie :

```
// temps de référence pour période de 24h
int heuresRef = 19;
int minutesRef = 45;
int secondesRef = 30;
```

Dans la boucle *loop()* l'action se décompose alors comme suit.

Au début de la séquence, on lit l'heure sur le Chronodot grâce à la fonction *updateHeure()* qui renseigne les variables heures, minutes et secondes (en les convertissant en décimal au passage) :

```
void updateHeure(){
DateTime now = RTC.now();    // on lit l'heure en cours
heures = now.hour(), DEC;
minutes = now.minute(), DEC;
secondes = now.second(), DEC;
}
```

Puis lorsqu'on atteint les deux dernières minutes avant l'heure de référence, l'horloge passe en mode « réglage » :

```
if ( heures == heuresRef && minutes == (minutesRef-2)) {
    // quand on entre dans les deux dernières minutes, on passe en mode réglage
    reglage = 1;
}
```

Deux événements sont alors surveillés : l'activation du microswitch par la grande aiguille (passage à LOW de contactPin) et l'arrivée en bout de course du balancier, détectée grâce à une interruption mise sur le capteur infrarouge correspondant et qui fait passer la variable bitTerminal à 1. Quand ces deux conditions sont remplies, le balancier est bloqué par rotation du servo jusqu'à ce que l'heure de référence soit atteinte, puis le balancier est relâché. On sort alors du mode réglage et on calcule différentes statistiques pour affichage sur les outils de monitoring (écran LCD et indicateur à LEDs, voir plus bas).

```
if (reglage == 1 && digitalRead(contactPin) == LOW && bitTerminal == 1) {
    // quand l'aiguille atteint le contacteur et que le balancier arrive en bout de course
    // on calcule l'avance (delta) pour affichage sur LCD et diodes
    delta=((minutesRef*60)+secondesRef)-((minutes*60)+secondes);

    if (delta > 0) { // si l'horloge avance
      while (heures != heuresRef || minutes != minutesRef || secondes != secondesRef) {
        // on bloque le balancier jusqu'à ce que l'heure de référence soit atteinte
        myservo.write(ferme); // blocage balancier
        updateHeure(); // lecture de l'heure sur le Chronodot
    }

    reglage=0; // on sort du mode réglage
    uptime++; // on incrémente le compteur de jours d'uptime

    if (uptime > 1) {
        // on calcule les stats pour affichage sur le LCD.
        // On ne prend pas en compte le 1er jour car les comptages sont partiels
        compteurTotal = compteurTotal + compteur;
        compteurMoyenne = compteurTotal/(uptime-1); // moyenne des comptages de balancier
        totalDelta = totalDelta+delta;
        deltaMoyenne = totalDelta/(uptime-1); // calcul moyenne du décalage quotidien
    }

    compteur = 0; // le compteur de passages est remis à 0
    myservo.write(ouvert); // on relache le balancier
    }
}
```

Indicateurs et accessoires

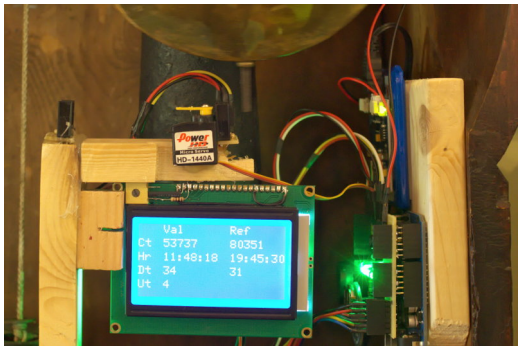


FIGURE 2.3 – Le montage

Histoire de ne pas tout faire à l'aveugle, des indicateurs permettent de monitorer le bon fonctionnement du système.

L'écran LCD

Pour suivre un peu ce qui se passe, un écran LCD est connecté à l'Arduino. Il s'agit d'un écran rétro-éclairé de 128×64 pixels acheté moins de 15 euros sur Dealextreme, sur les bons conseils de Skywodd qui nous fait en plus la grâce d'un tuto complet sur son utilisation.

L'affichage se divise en 2 colonnes : les valeurs courantes et les valeurs de référence.

- la première ligne indique le nombre de passages de balanciers comptés depuis le dernier réglage et le nombre moyen par 24h depuis le lancement du système ;
- la deuxième ligne indique l'heure courante et l'heure de référence ;
- la troisième ligne indique l'avance de l'horloge en secondes et la moyenne depuis la mise en route du système ;
- la dernière ligne indique le nombre de jours écoulés depuis la mise en route (uptime).

Le premier jour de fonctionnement est ignoré dans les statistiques puisqu'il est forcément partiel.

Pour l'affichage sur l'écran, il est nécessaire de formater les données au préalable. L'écran LCD ne comprend que les chaînes caractères en tableau et les données à afficher sont des chiffres. Une fonction *longToChar()* transforme donc le chiffre *valeur* en chaîne de caractère *cible[]* de longueur *taille*.

```
char longToChar(long valeur, int taille, char cible[]) {
  // convertit les long en char affichables par l'écran
  String string = String(valeur);
  string.toCharArray(cible,taille);
}
```

Pour convertir l'heure, c'est le même principe via la fonction *heureToChar()* avec en plus une fonction *subzero()* qui ajoute un zéro aux valeurs inférieures à 10 :

```
char heureToChar(int h, int m, int s, char cible[10]) {
  // convertit l'heure en char affichables par l'écran
  String heureString = String(subzero(h));
  String minuteString = String(subzero(m));
  String secondeString = String(subzero(s));
  String temps = heureString + ":" + minuteString + ":" + secondeString;
  temps.toCharArray(cible, 10);
}

String subzero(int valeur){
  // ajoute une zero aux chiffres horaires < 10
```

```
  String resultat = String(valeur);
  if (valeur < 10) {
      resultat = '0'+ resultat;
  }
  return(resultat);
}
```

Enfin la fonction *draw()* s'occupe de formatter toutes les données pour les placer sur l'écran (voir le tuto de Skywodd pour les détails) :

```
void draw() {
  // affichage écran
  u8g.setFont(u8g_font_6x12); // Utilise la police de caractère standard
  u8g.drawStr(22, 8, "Val");
  u8g.drawStr(80,8, "Ref");
  u8g.drawStr(0, 20, "Ct");
  u8g.drawStr(22,20, compteurChar);
  u8g.drawStr(80,20, compteurMoyenneChar);
  u8g.drawStr(0, 32, "Hr");
  u8g.drawStr(22,32, tempsChar);
  u8g.drawStr(80,32, tempsReferenceChar);
  u8g.drawStr(0, 44, "Dt");
  u8g.drawStr(22,44, deltaChar);
  u8g.drawStr(80, 44, deltaMoyenneChar);
  u8g.drawStr(0,56, "Ut");
  u8g.drawStr(22, 56, uptimeChar);
}
```

Pour générer l'affichage, toutes les conversions sont faites et les caractères sont envoyés à l'écran par appel de la fonction *draw()*. A noter que l'écran ne s'allume que si le bouton-boussoir correspondant a été pressé, faisant passer la variable *ecranAllume* à 1. Si on le presse à nouveau la variable repasse à 0 et l'écran s'éteint.

```
if (ecranAllume == HIGH) {
    // si l'ecran est allume
    // conversions pour l'ecran
    longToChar(compteur,7,compteurChar);
    longToChar(compteurMoyenne,7,compteurMoyenneChar);
    longToChar(delta,5,deltaChar);
    longToChar(uptime,5,uptimeChar);
    longToChar(deltaMoyenne,5,deltaMoyenneChar);
    heureToChar(heures,minutes,secondes,tempsChar);
    heureToChar(heuresRef,minutesRef,secondesRef,tempsReferenceChar);
    u8g.firstPage(); // Sélectionne la 1er page mémoire de l'écran
    do {
        draw(); // Redessine tout l'écran
    }
    while(u8g.nextPage()); // Sélectionne la page mémoire suivante
}
```

L'indicateur lumineux

Un indicateur lumineux constitué de LEDs permet en outre de surveiller la dérive de l'horloge. Lorsque celle-ci est dans la « zone acceptable » (de 20 à 40 secondes) une diode verte s'allume. De 0 à 20 secondes ou de 40 à 60 secondes, une diode orange signale la dérive. En cas de retard, ou d'avance supérieure à 60 secondes, une diode rouge signale le problème.

L'avantage de l'indicateur lumineux est que, contrairement à l'écran LCD, il est visible en permanence par la vitre de la caisse, il n'est donc pas nécessaire d'ouvrir l'horloge pour le consulter.

Pour l'affichage, rien de bien compliqué. les seuils d'activation sont définis en début de script :

```
// Echelle des temps d'avance pour afficheur led
int borneMin = 0;
int borneInf = 20;
int borneSup = 40;
int borneMax = 60;
```

La fonction *indicateur()* se charge d'éteindre toutes les LEDs puis d'allumer la bonne. Celles-ci sont placées sur des pins consécutifs ce qui simplifie un peu le code. Les cas particuliers de l'activation du microswitch par la grande aiguille (allumage des deux LEDs orange) et de la détection des poids de l'horloge (allumage des deux LEDs rouges) sont pris en compte en début de fonction (voir paragraphe suivant).

```
void indicateur(int led) { // eteint toutes les leds
  for (int i=4; i <=8; i++) {
      digitalWrite(i, LOW);
  }
  if (digitalRead(CapteurPoids)==LOW) { // si on voit le poids on allume les 2 leds rouges
      digitalWrite(led1,HIGH);
      digitalWrite(led5,HIGH);
  }
  else {
      digitalWrite(led, HIGH); // sinon on allume la LED d'indication du delta
  }
}
```

On appelle ensuite la fonction à chaque passage du balancier devant le capteur central, l'afficheur est donc mis à jour toutes les 1.07 secondes :

```
// affichage de l'avance/retard sur les leds
if (delta <= borneMin) {
  indicateur(led5); // rouge 1
}
if (delta > borneMin && delta <= borneInf) {
  indicateur(led4); // orange 1
}
if (delta > borneInf && delta <= borneSup) {
  indicateur(led3); // vert
}
if (delta > borneSup && delta <= borneMax) {
  indicateur(led2); // orange 2
}
if (delta > borneMax) {
  indicateur(led1); // rouge 2
}
```

Le remontage des poids

Le remontage des poids doit s'effectuer chaque semaine (enfin plutôt tous les 6 jours et demi, je pense que la caisse de l'horloge a été raccourcie au cours de son histoire pour passer sous un plafond trop bas).

Pour ne pas rater ce moment, un détecteur de proximité (un mini télémètre infrarouge) détecte quand le poids arrive en bas de l'horloge, environ 24h avant qu'il ne touche le sol et que l'horloge s'arrête.

Sitôt le poids détecté, des Leds rouges s'allument sur l'indicateur lumineux. Enfin, un buzzer piézo-électrique émet des bips pendant 2 minutes avant le blocage du balancier.

Chose curieuse, la fonte noire dont est fait le poids était totalement invisible pour le capteur infrarouge. J'ai donc dû l'emballer dans une feuille de papier noir pour le rendre détectable (mais pas trop laid quand même).

L'alimentation

Lors de ma première approche qui impliquait le comptage des passages de balancier, il était très important que l'Arduino ne se remette pas à zéro en cas de coupure de courant par exemple.

J'ai donc intercalé un module LipoRider qui permet à une batterie de 2000 mAh de prendre momentanément le relais en cas de défaillance de l'alimentation principale. Avec le capteur sur aiguille de la deuxième approche cette précaution se justifie moins, mais je l'ai tout de même laissée.

Évolution

Parmi les évolutions possibles, j'envisage l'ajout d'une connexion à un serveur NTP via un shield Ethernet pour recaler le Chronodot périodiquement sur une horloge atomique. On aura ainsi l'horloge comtoise la plus précise de l'Univers.

Le code

Je vous livre enfin le code complet du système. Ma formation universitaire étant l'Histoire, je compte sur l'indulgence des développeurs professionnels !

Le code de l'ArduComtoise.

3. Faire de la bière maison (ou comment transformer l'eau en bière)

date 2013-08-01
category cuisine,écologie
author Frédéric Sureau, Alexis Métaireau
level moyen

On en boit tout le temps, mais on ne prend pas souvent le temps de comprendre comment elle est fabriquée, et encore moins de la fabriquer soi même. Et c'est bien dommage, parce que fabriquer de la bière soi-même est à la portée de n'importe quel assoiffé (s'il est un peu patient) !

Mais d'abord, pourquoi faire de la bière soi-même ?

Il y a plein de bonnes raisons en fait, mais en ce qui nous concerne, c'est principalement qu'on aime bien fabriquer les produits qu'on consomme (et, ça serait mentir que de dire qu'on ne consomme pas de bière !).

Ingrédients

Pour cette recette, il vous faudra :

De l'eau Vous savez, ce truc liquide et transparent.

Du malt Des grains d'orge germés puis séchés voire grillés. Il est responsable de la couleur et du gout de la bière.

Du houblon Des fleurs pour aromatiser et donner de l'amertume.

Des levures Pour transformer le sucre en alcool et faire des bulles.

Tous les ingrédients y compris l'eau ont leur influence sur le goût final de la bière par leur qualité et leur caractéristique, région d'origine, etc.

Matériel

Les grandes brasseries utilisent une cuve spéciale pour chaque étape, mais quand on est pauvres/débutants et que l'on fait des petites quantités, on peut très bien s'en sortir avec très peu de matériel. Dans notre cas, nous faisons des brassins de 20L ce qui est un bon rapport risque/plaisir : si on rate, ce n'est pas si grave, et si elle est bonne il y a quand même de quoi déguster ;)

L'essentiel :

Une source de chaleur réglable Un trépied à gaz pour les paëlla est parfait, sinon une plaque de cuisson quelconque.

Une cuve de brassage Une grande casserole/marmite de minimum 30L. 50L c'est plus confort. En inox ou émaillée, mais surtout pas d'aluminium car il ne réagit pas très bien avec la bière paraît-il... Le mieux est de la récupérer chez Tatie Jacqueline car ça coûte très cher (à partir de 50€ et 150€ facilement selon la qualité)

Une cuve de fermentation Un seau en plastique avec couvercle hermétique de minimum 30L. Utiliser de préférence du plastique alimentaire, sinon la bière développe un goût de pétrole très caractéristique (on a testé pour vous) On trouve chez les distributeurs spécialisés des cuves de fermentation de 30L avec barboteur et robinet pour une dizaine d'euros.

Une touillette Une grande cuillère quoi...

Un thermomètre De cuisine, gradué entre 50°C et 100°C.

Une passoire Très grande si possible.

Recommandé fortement :

Un moulin à malt Du genre de ce qu'il y a dans les fermes pour concasser les céréales qu'on donne aux bêtes, ou un moulin à malt spécial (environ 50€ les premiers prix) Si vous n'en avez pas sous la main, il y a les techniques torchon/marteau (long), rouleau à pâtisserie (fastidieux) ou encore le mixeur (de quoi être radié à vie de l'ordre des brasseurs)

Refroidisseur On peut s'en bricoler un pour pas cher avec des tuyaux de cuivres pour canalisation. Il suffit de le tordre en forme de serpentin et de faire des jolis raccords. Sans cet ustensile, l'étape de refroidissement est longue et risquée pour la qualité de la bière.

Facultatif (mais quand même sympa) :

Un densimètre Pour faire des mesures et savoir au bout de combien de pintes on ne peut plus conduire. Accessoirement c'est pratique pour connaître le moment idéal pour la mise en bouteille.

Un tuyau pour siphonner Lors de la mise en bouteille. Prévoir un bon diamètre (environ 1cm) pour ne pas y passer des plombes.

Processus de fabrication

Cet article explique comment fabriquer une IPA (Indian Pale Ale, une bière un peu plus amère que celle qu'on à l'habitude de trouver en France), mais le principe reste sensiblement le même pour d'autre types de bière. La première étape consiste à concasser le malt. Ce malt est ensuite mélangé à de l'eau. On obtient ainsi un mélange appelé *la maische*. Ce mélange est porté à plusieurs paliers de température.

Ensuite on filtre les grains utilisés (les drèches) et on ne garde que le jus, appelé le moût. Le moût est porté à ébullition en compagnie de houblon dont on extrait l'amertume et les arômes. Le houblon est ensuite retiré du moût. Celui-ci est refroidit à 25°C, la température de prédilection pour les levures qui se font un plaisir de le fermenter pendant 2 à 3 semaines. En fût d'abord, puis en bouteilles où le gaz généré n'aura d'autre solution que de se dissoudre dans la bière pour lui donner ses propriétés bulleuses.

Concassage

La première étape est **le concassage**. On avait 6kg d'orge maltée à concasser (plus exactement 6kg de malt pâle *7EDC* et 500g de malt cara *50EDC*).

Attention, pas trop fin dis donc !

Si on concasse le malt trop fin, on se retrouve avec un dépôt dégeulasse au fond de toutes les bouteilles.

On ne veut pas de la poudre, on veut simplement permettre à l'amidon de se dissoudre dans l'eau.

Empâtage et brassage

La seconde étape est **l'empâtage**. Il s'agit de faire chauffer notre moût à différents paliers de température.

- ~50°C : Transformation des protéines de l'orge malté pour que la bière ne soit pas trouble (protéinase) ;
- ~60-65°C : Création des sucres fermentescibles (dextrose et maltose) ;
- ~68-75°C : Création de sucres non-fermentescibles (Dextrine) ;
- ~78°C : Destruction des enzymes et arrêt des transformations (parce que c'est bon hein).

Les paliers indiqués pour la création des sucres sont les températures idéales, mais ils se créent également à d'autres températures. Il est donc envisageable de n'avoir qu'un palier pour la création des sucres.

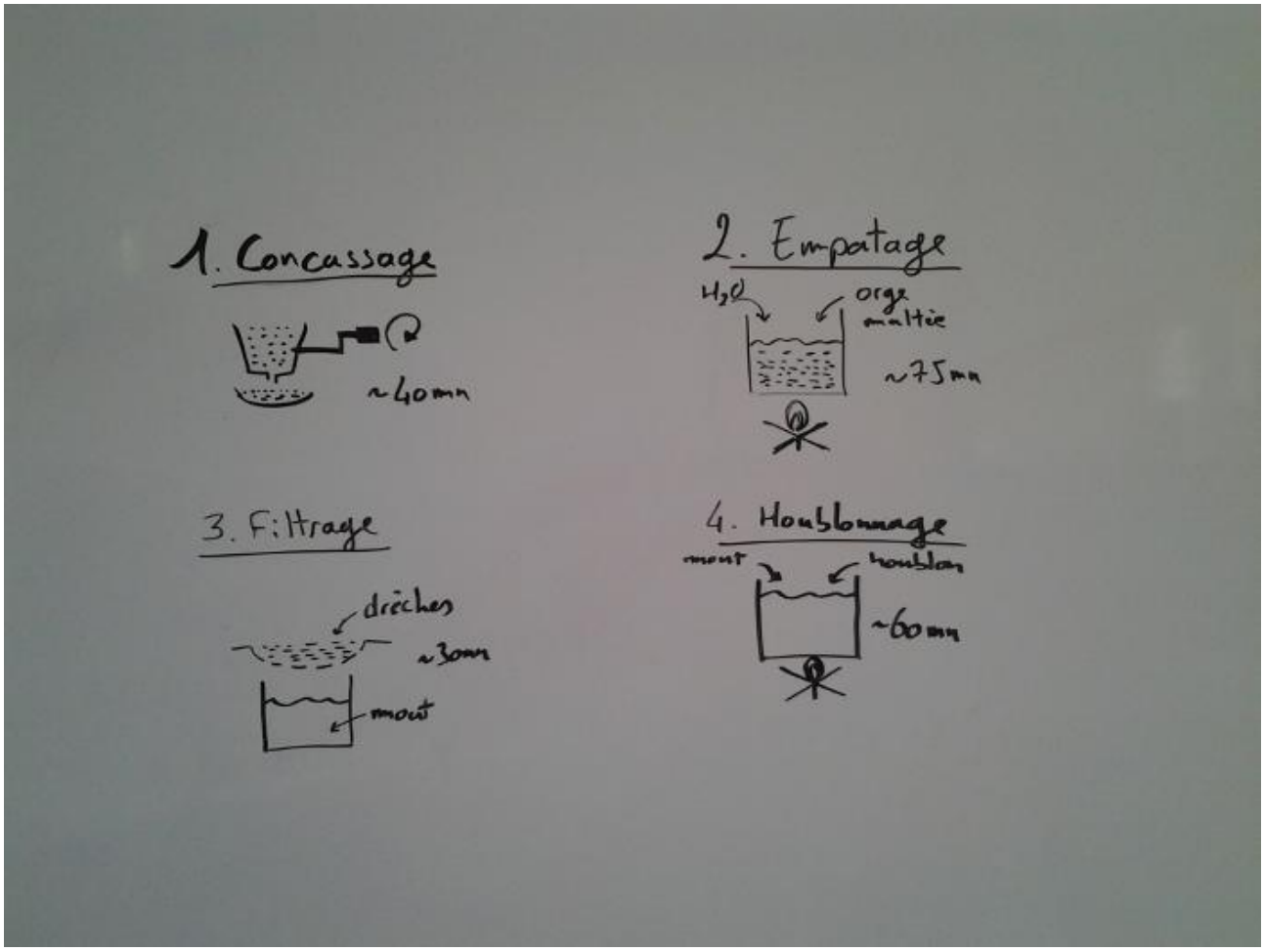


FIGURE 3.1 – Les étapes de fabrication

FIGURE 3.2 – Concassage de l'orge

Rinçage

La troisième étape, c'est **le rinçage**, l'idée est de récupérer l'amidon qui s'est dissout dans l'eau et de mettre de côté l'orge maltée (la partie solide).

Pour ça, il faut faire chauffer de l'eau de rinçage. On a utilisé 10L d'eau de rinçage qu'on a fait chauffer à 78°C, en comptant sur le fait qu'elle perdra de sa température (20°C à peu près) en étant utilisée. On a filtré deux fois pour être sûr de ne rien perdre.

Les drèches (résidus du grain) sont données aux poules pour leur procurer un plumage soyeux.

Houblonnage

L'étape d'après (la quatrième, vous suivez), c'est **le houblonnage**. L'idée c'est de faire infuser notre mout avec du houblon. On fait bouillir le moût pendant 1h.

Lorsqu'on ajoute le houblon dès le début, on extrait l'amertume. Lorsqu'on ajoute le houblon dans les dernière minutes, on extrait ses arômes.

Une troisième technique consiste a ajouter du houblon directement dans la cuve pendant la fermentation, c'est ce qu'on appelle le houblonnage à cru.

Dans notre cas (une bière amère) nous avons mis beaucoup de houblon amer à infuser et également pratiqué le houblonnage à cru.

Refroidissement

Étape suivante : **le refroidissement**.

On dirait que c'est facile comme ça, mais en fait ça ne l'est pas tant que ça : il faut réussir à faire tomber la température de notre liquide en ébullition jusqu'à 25°C en un temps acceptable.

Pour ça, on a utilisé un serpentin confectionné par nos petites mains.

Ce refroidisseur nous a permis d'atteindre la tempéra-

FIGURE 3.3 – Rinçage

ture souhaitée en 35 minutes !

Sans le refroidisseur, il faut compter au moins le double et on s'expose à des risques de contamination par les méchantes bactéries qui peuplent nos garages, squats, cuisines ou autres lieux de brassage.

Fermentation

Dernière étape, haha !

Il faut rajouter les levures qui vont faire tout le travail et transformer le moût en bière, pendant que nous nous reposons, une bonne bouteille de Chimay à la main.

Dans notre cas, on ajoute aussi dans la cuve de fermentation du houblon pour le houblonage à froid.

C'est à cette étape qu'on mesure la densité du liquide. Cette densité représente la quantité de sucre extraite du malt, et donc la quantité d'alcool potentielle dans la bière.

On ferme la cuve de fermentation avec un barboteur pour laisser le gaz s'echapper de son « bloup bloup » caractéristique.

La première fermentation durera approximativement 15 jours, jusqu'à atteindre une densité finale stable (fin de la fermentation).

Notre bière avait une densité initiale de 1046, et une densité finale de 1008 ce qui veut dire environ 5° d'alcool grâce à des calculs scientifiques super compliqués .

Mise en bouteille

Une fois ces deux semaines passées, il faut mettre en bouteille. Dans notre cas nous avons récupéré des bouteilles à bouchon mécanique (vous savez, les bouteilles de limonades) qu'on a bien rincé et nettoyé.

On s'équipe de notre siphon et après avoir ajouté du sucre dans le mélange (pour réactiver les levures), on remplit les bouteilles.

Cette seconde fermentation en bouteille donnera son effervescence à la bière grâce à la dissolution du gaz carbonique dans le liquide. Et voilà ! Il ne reste plus

qu'à mettre les bouteilles de coté durant deux semaines de plus (oui, c'est long) et... à déguster le moment venu.

Dégustation !

La bière en question ne moussait pas trop et avait un goût un peu amer. Pas assez à notre goût, cependant.

En comparaison à nos précédents essais, c'est plutôt positif : la première était ratée puisque trop de pression et trop de dépôt (et donc un fort goût de levures) alors que la seconde (on avait tenté d'ajouter de la lavande) avait un goût de lavande amère, pour ne pas dire de savon.

Celle-ci a un goût de... de bière ! Il nous reste encore à comprendre comment faire pour lui donner la saveur que l'on souhaite.

Conseils/Anecdotes

On a fait quelques petites erreurs en cours de route, voilà rien que pour vous une petite compilation :

- par peur de la contamination bactérienne, on a décidé de faire bouillir nos 26L d'eau pour être sur que les bactéries s'enfuient en courant. Je dis erreur parce que ça nous a pris pas loin de 3h30 pour réussir à chauffer et refroidir ce volume d'eau. Inertie quand tu nous tiens !
- lorsque vous ajoutez le malt dans l'eau, pensez bien qu'il va refroidir la température de l'eau. Comptez perdre approximativement 4°C ;
- lors de l'ébullition, vous allez surement perdre un peu d'eau, pensez à en mettre un peu plus (même si vous couvrez).

Mais alors, c'est quoi les bières brunes, blondes, les stout, etc. ?

La couleur de la bière est déterminée par le mélange de malts choisis. Il existe ainsi des malts bruns, chocolat, noirs, caramel, pâles, etc.

Pour une bière blonde, on utilisera quasiment uniquement du malt pâle. Une bière ambrée sera composé de malts plus foncés, une bière brune sera composée de malts bruns etc. Cependant, quelle que soit la recette, le mélange sera composé d'au moins 80% de malt pâle, même pour une bière stout !

Les bières blanches sont un cas particulier, elles ne sont pas composées uniquement de malt d'orge, mais on leur ajoute également du blé en petite quantité, des épices, et autres secrets bien gardés.

FIGURE 3.4 – Nous deux après notre premier brassin. (admirez la superbe cuve « princesse »)

4. L'hexapode Bleuette

date 2013-08-01
category informatique,électronique
level vulgarisation

author Charles Rincheval

Introduction

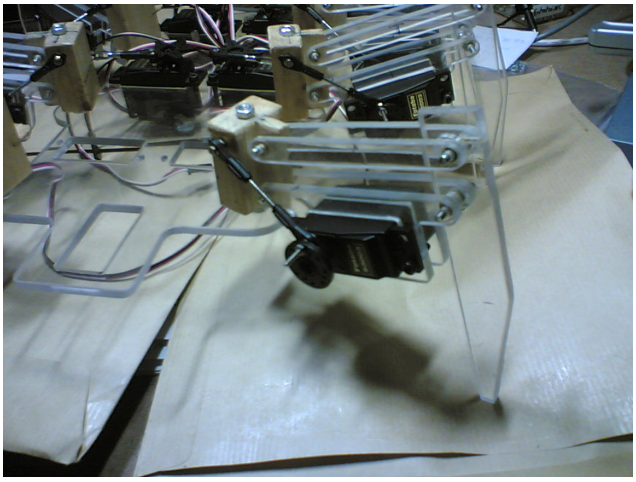
Bleuette est un robot hexapode (6 pattes) entièrement libre, c'est à dire que tous les plans, programmes, informations permettant de le fabriquer sont libres : jouez avec, modifiez les, redistribuez les comme bon vous semble...

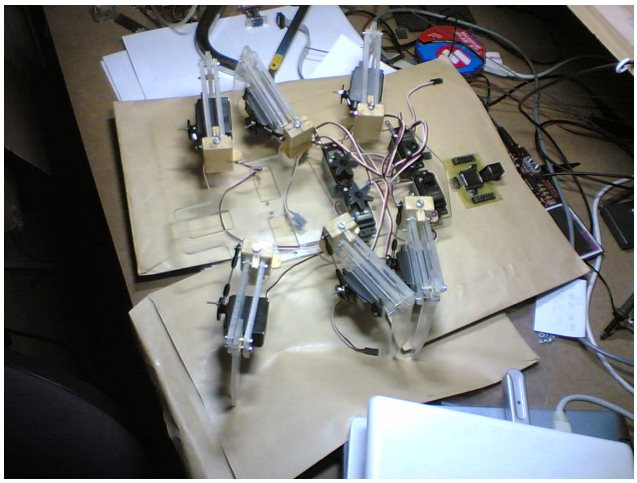
Il est encore en stade de développement mais est déjà parfaitement fonctionnel en terme de déplacement. Son développement se poursuit activement afin de le rendre plus autonome grâce à des capteurs divers.

Histoire

Pour certain d'entre vous, le nom « Bleuette » vous est peut-être familier, en fait, c'est une féminisation de Bleuet, le robot de la série FX, Effets spéciaux, nous nous sommes dit que le pauvre Bleuet devait se sentir seul et qu'il faudrait lui fabriquer un congénère et puis tant qu'à faire femelle... :)

Dans ses premières versions, Bleuette était réalisée en Plexiglas, découpé à la scie sauteuse, un vrai boulot, long et pénible avant de nous rendre compte des limites de cette matière : elle est cassante et se raye trop facilement. Une seconde version a été faite en Lexan, une matière avec des propriétés déjà plus intéressantes...





Quelques années se sont écoulées et, depuis, l'impression 3D s'est considérablement développée, au point d'être devenue attractive pour les particuliers et intéressante pour un projet comme Bleuette, l'achat d'une l'Ultimaker a été guidé par l'arrière-pensée de faire Bleuette entièrement en plastique.

- les tout premiers pas de Bleuette en Plexiglas ;
- quelques années plus tard, Bleuette imprimée en 3D.

Tous les plans de Bleuette ont été refaits à l'aide de OpenSCAD, un logiciel permettant de modéliser des pièces de manière paramétrique, un dépôt GitHub a été créé à l'occasion pour partager tous les documents inérants à la fabrication du robot.

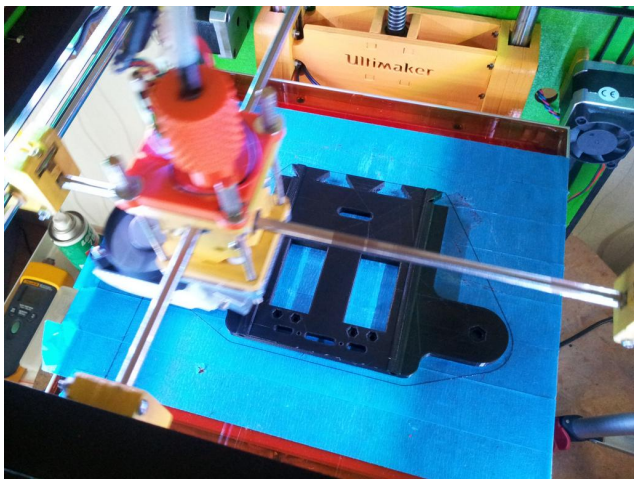
Construction

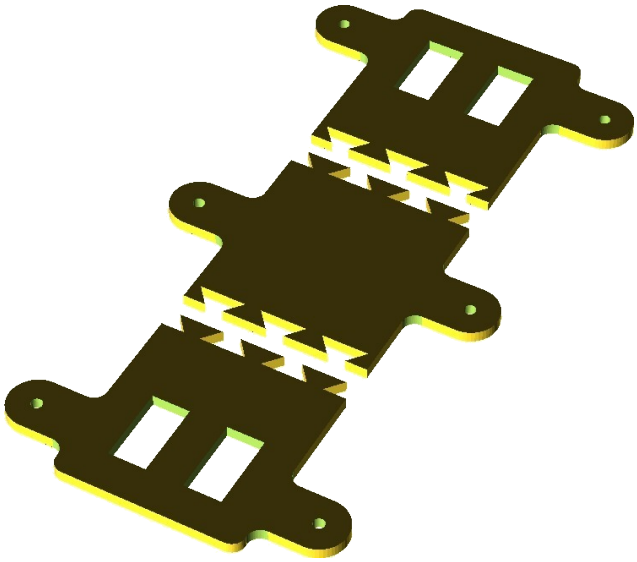
Toute la mécanique de Bleuette est entièrement faite en plastique réalisable avec une imprimante 3D, vous n'avez besoin que de très peu d'éléments autres (vis, écrous, tiges métal).

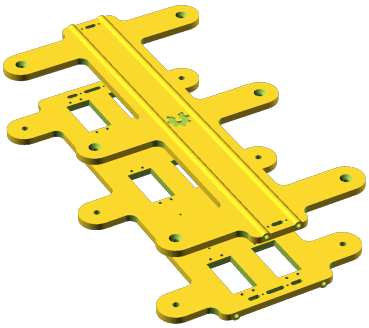
Il est même parfaitement envisageable avec quelques légères adaptations de découper les pièces avec une fraiseuse ou une découpe laser.

Le corps

Le corps de Bleuette étant trop grand pour être imprimé en une seule fois, il est fait en plusieurs parties assemblées par des queues d'arondes, une librairie OpenSCAD a été créée pour l'occasion



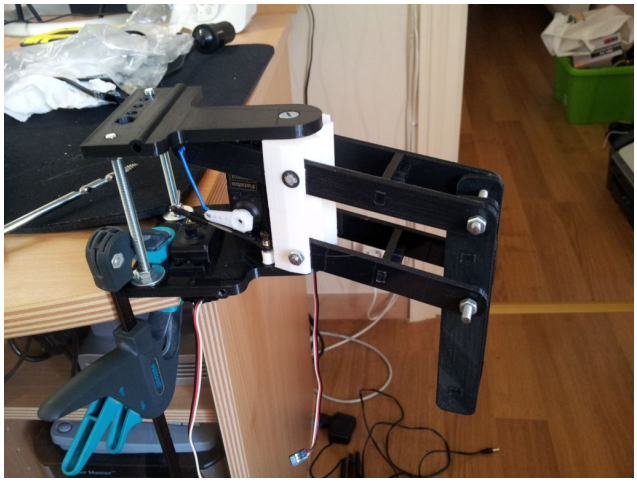




Une fois les différentes parties imprimées, il suffit des les emboiter en force, les jeux étant très faibles, c'est très rigide.

Les pattes

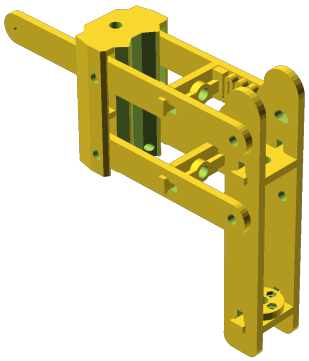
Bleuette possède 6 pattes animées chacune par 2 servos standards (Futaba S3003), un qui permet de lever la patte et un autre qui permet de faire pivoter horizontalement la patte.



Voici une patte montée sur une structure de test pendant la conception :

Détail d'une liaison de patte et vue de l'assemblage : on emboite l'entretoise, on la pivote de 90 degrés et ça ne bouge plus, aussi simple que des Lego !

Une patte entière :



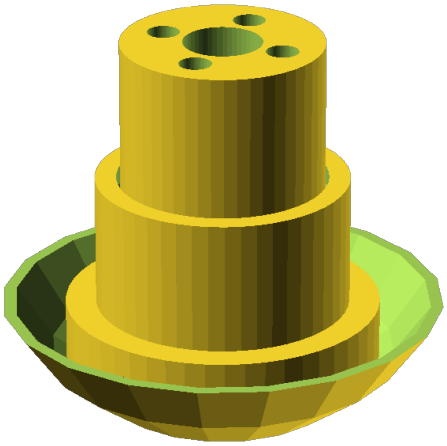
Les palonniers

Les palonniers de servos, une pièce relativement complexe à imprimer car nécessitant une grande précision est aussi réalisée grâce à l'Ultimaker et une librairie OpenSCAD à été créée à l'occasion, grâce à elle, on peut créer des palonniers à 1, 2, 4 bras, voir plus et de tailles diverses.

Les capteurs de sol

Le bout de chaque patte de Bleuette est équipé d'un capteur capable de détecter via un interrupteur une pression verticale correspondant au contact de la patte avec le sol, ainsi, notre hexapode est capable de détecter une absence de sol et donc, par exemple, d'éviter de tomber dans un escalier...

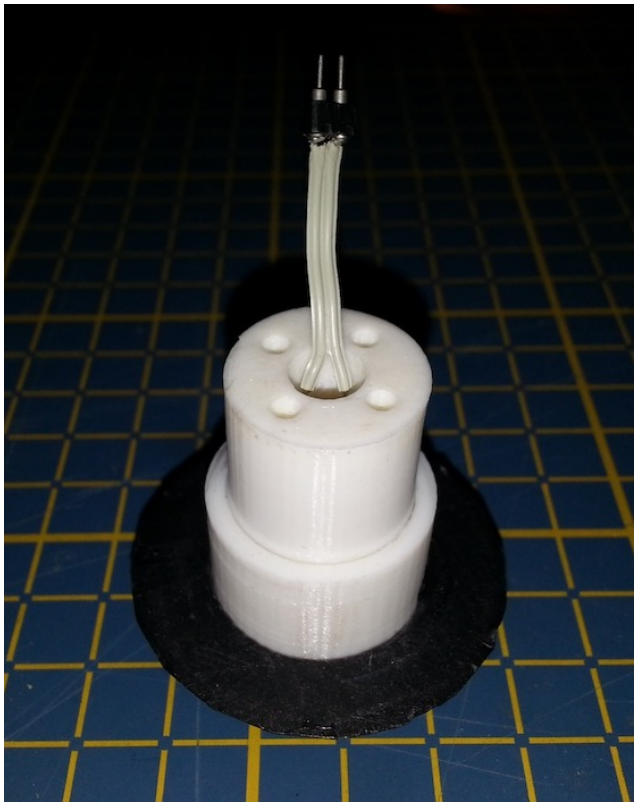
La surface du capteur en contact avec le sol est imprimée en PLA Flex, qui comme son nom l'indique très bien est en PLA mais avec la particularité d'être flexible.



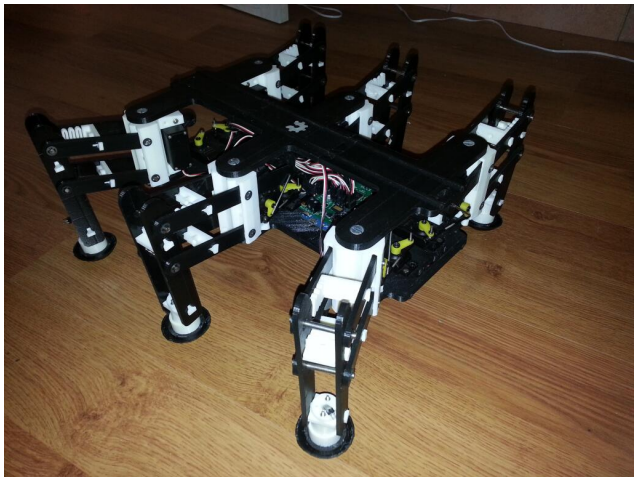
Sur la première photo ci-dessous, nous voyons la différence entre la pièce brute et la pièce trempée dans du PlastiDip pour obtenir plus de grip et meilleure finition.

Sur la seconde photo de droite, on peut voir tous les éléments d'un capteur de sol :

Tous les éléments du capteur sont assemblés :



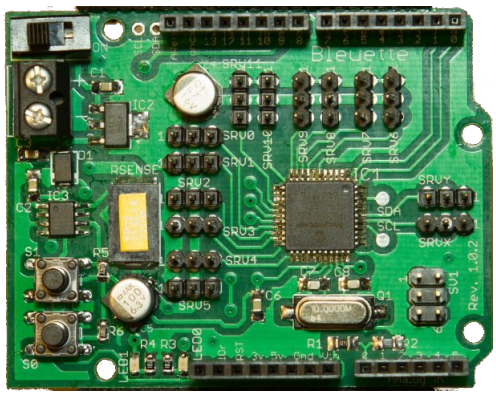
Bleuette et ses capteurs :



Le cerveau

Version Arduino

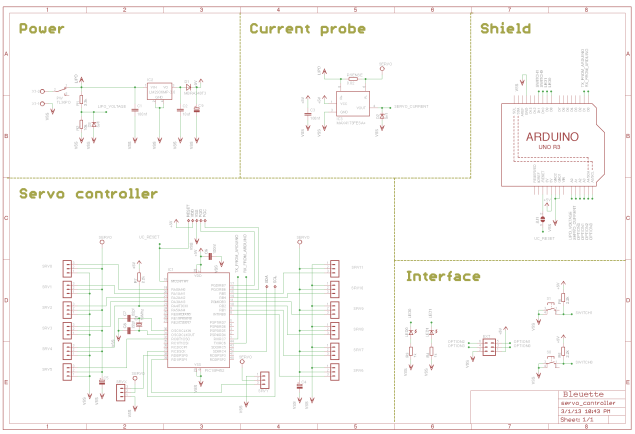
Le cerveau de Bleuette est une carte Arduino avec une shield spécialement conçue pour Bleuette qui sert, entre autres, à piloter les 12 servos des pattes.



Cette carte possède les caractéristiques suivantes :

- génération de la tension de 5V pour l'Arduino ;
- mesure du courant consommé par les servos ;
- surveillance de la tension de la batterie ;
- gestion synchrone de la commande des 12 servos des pattes
  - 2 servos optionnels (basé sur un PIC18F452 et Pic24Servos).

Voici son schéma de principe :



Plus d'informations sur la Bleuette Shield.

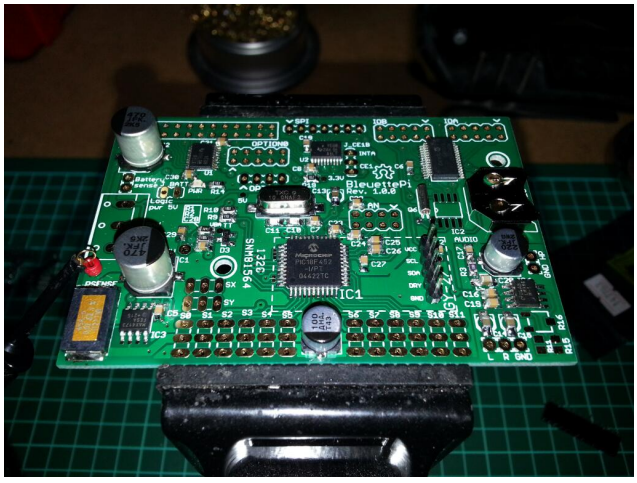
Une autre carte fille (shield) pour Arduino a été créée, il s'agit de Bleuette Sensor Shield qui dispose des caractéristiques suivantes :

- 8 entrées supplémentaires multiplexées utilisant que 4 entrées/sorties (3 d'adressage et une sortie) ;
- Connexion pour une carte GY-27 contenant un accéléromètre et un compas ;
- un module Bluetooth JY-MCU ;
- une connexion pour une guirlande de led RGB à base de LPD8806 ;
- un mosfet pour pouvoir piloter un élément de puissance (je ne sais pas vraiment quoi pour le moment...).

Plus d'informations sur cette carte.

Version Raspberry-Pi

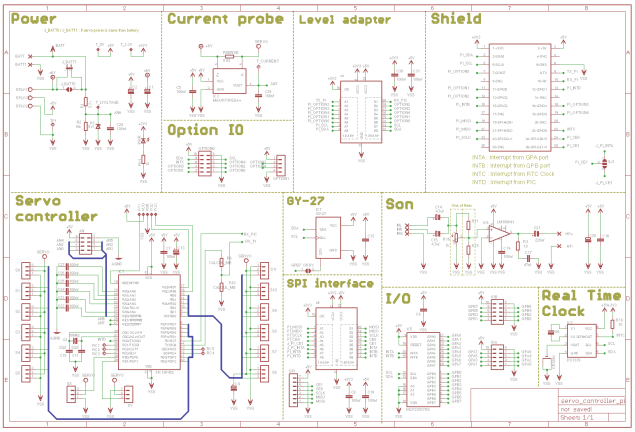
Piloter Bleuette via un Arduino est plutôt aisé mais on se retrouve assez vite limité en place, en puissance et en facilité pour développer. J'ai donc décidé de faire une carte fille pour Raspberry-Pi, cette carte fille nommée simplement Bleuette-Pi propose tout ce que fait la shield Arduino avec plein de choses en plus !



Voici ses caractéristiques :

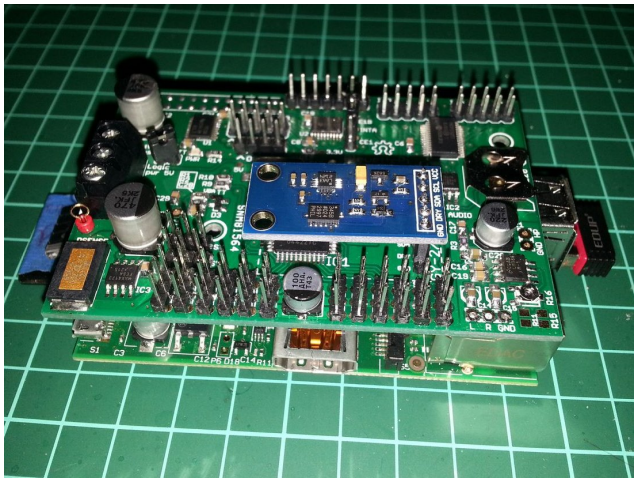
- gestion de **14 servos** (toujours de manière synchrone) ;
- **mesure du courant** consommé par les servos ;
- **mesure de la tension** de la batterie ;
- connexion pour une carte GY-27 contenant un **accéléromètre et un compas** (via I2C) ;
- **16 entrées / sorties** compatibles 5V avec 2 lignes d'interruption, le tout commandé en I2C ;
- **6 entrées analogiques** ;
- **amplificateur audio** pour ajouter le son à votre Raspberry-Pi (à base de LM386) ;
- **5 entrées / sorties généralistes compatibles 5V** direct Raspberry + 5 autres entrées/sorties disponible si le bus SPI n'est pas utilisé ;
- toutes les broches du SPI sont disponibles sur un connecteur et compatible 5V ;
- un module **horloge temps réel (RTC)** pour garder votre Raspberry Pi à l'heure !
- **4 lignes d'interruptions physiques**.

Son schéma de principe :



Toutes les informations sur cette carte sont disponibles sur le wiki

BleuettePi montée sur un Raspberry-Pi (on aperçoit une petite carte bleue, c'est le GY-27 c'est une combinaison d'un accéléromètre et d'un compas) :



Exemple de code en Python

Pour animer les pattes de Bleuette, il faut créer une séquence qui n'est rien d'autre qu'une classe, voici un exemple qui fait faire une pompe à Bleuette :

```
class Seq_PushUp:
    # On donne un nom à notre séquence
    name = "Push Up"

    # La séquence de mouvement
    sequence = [
        [
            # Délai de 0.5 seconde
            0.5,
            [
                FRONT, FRONT, MID, MID, BACK, BACK,     # Positions des servos de pattes (servos d'axe
                UP, UP, UP, UP, UP, UP                  # On place les pattes en haut (servos d'axe ver
            ],
            # Ici, on peut définir une callback qui sera appelée à chaque fois que les pattes
            # seront placées dans leur position voulue
            None
        ],
        [
            0.5,
            [
                __, __, __, __, __, __,                 # On ne touche pas à la position horizontale de
                DOWN, DOWN, DOWN, DOWN, DOWN, DOWN      # On place les pattes en bas
            ],
            None
        ]
    ]
```

Maintenant, pour exécuter la séquence ci-dessus, il suffit du bout de code suivant :

```
from Bleuette import Bleuette

B = Bleuette()

# Fait faire 4 pompes à Bleuette
B.Sequencer.forward(Seq_PushUp, 4)
```

Voilà, c'est aussi simple que ça !

Sur les nouvelles versions du code, j'ai ajouté un thread qui s'occupe uniquement d'envoyer les ordres aux servos, ainsi, il est possible de faire d'autres tâches pendant ce temps.

Tout le code pour jouer avec Bleuette

Une vidéo des premiers mouvements de pattes de Bleuette pilotée par une Raspberry-Pi.

<iframe width="800" height="450" src="//www.youtube.com/embe frameborder="0" allowfullscreen></iframe>

Fabriquer

Pour fabriquer votre propre Bleuette, il vous faudra :

Une CNC

Pour les pièces du corps de Bleuette, il vous faudra avoir accès à une imprimante 3D ou une découpe CNC, inutile d'en posséder une, il vous suffira de trouver le fablab le plus proche de chez vous qui pourra vous orienter et vous aider dans leur réalisation.

Fabriquer les pièces en plastique.

L'électronique

Selon la version choisie, vous devrez vous procurer :

- 1 carte Arduino Leonardo + la Shield Bleuette ;
- 1 Raspberry-Pi + la carte fille Bleuette-Pi.

Pour la Shield Bleuette ou la carte fille Bleuette-Pi, 2 solutions :

- fabriquer l'électronique ;
- ou vous les procurer sur cette page.

Pièces diverses

- 12 servos standard (type Futaba S3003) ;
- visserie, tiges, clips, pièces mécaniques diverses ;
- batterie, divers...

Tout ce matériel doit couter au maximum 250€, ce qui fait de Bleuette un robot hexapode très abordable, notez que l'on trouve dans le commerce des équivalents à plus de 900€...

Participez !

Bleuette est en perpétuel développement, vous pouvez suivre le dépôt GitHub pour vous en rendre compte, le développement se poursuit sur différents axes :

- Logiciel : poursuite du développement en Python sur Raspberry, gestion de la webcam du RaspberryPi avec OpenCV ;
- Mécanique : développemement d'une tourelle 2 axes pour la webcam ;
- Électronique : ajout d'un watchdog sur la carte BleuettePi et développement de la carte de puissance.

Chacun peut apporter ses propres compétences dans un domaine particulier !

- Le blog de développement : ;
- toutes les sources de Bleuette ;
- le wiki en français.

5. Cave à cigares

date 2013-07-07
category autre
level vulgarisation

author Sébastien Riguet

Introduction : une cave à cigares DIY

Devenu amateur de cigares après un cadeau insolite pour un noël, j'ai voulu, comme pour le bon vin, conserver mes acquisitions (de plus en plus nombreuses), dans les meilleures conditions possibles. Je me suis donc affairé à trouver une solution de conservation optimale.

Après quelques recherches sur Internet, j'ai trouvé plusieurs types de cave à cigares, une large fourchette budgétaire (de 20 à plus de 1 000 euros...) et différents design.

Ne voulant pas y mettre un montant pharamineux, mais voulant toutefois que mes cigares soient conservés dans de bonnes conditions, je me suis donc lancé dans une réalisation « fait main » d'une cave à cigares.

Les pré-requis d'une bonne cave à cigares

Commençons par les fondamentaux pour sa conception.

La priorité pour une cave à cigares est l'humidité constante qui doit être maintenue à 70%. Pour arriver à un tel résultat, 3 choses essentielles :

- du cèdre d'Espagne pour le doublage intérieur de la cave,
- un humidificateur (dépendant principalement de la capacité de la cave),
- un hygromètre pour mesurer avec précision le taux d'humidité.

Les autres points sont plus subjectifs car liés à la capacité et au design attendus.

Les matériaux

Nous aurons donc besoin de bois de cèdre d'Espagne, qui sera sans doute la plus grosse difficulté de cette réalisation. En effet malgré son nom, ce bois ne vient pas d'Espagne, mais d'Amérique latine, d'où il tire ses propriétés de régulateur d'humidité. Ce bois, très caractéristique et reconnaissable pas sa forte odeur poivrée, est notamment utilisé pour la fabrication des manches de guitare. Pour une cave à cigares de faible contenance, il est possible d'utiliser uniquement du placage en cèdre d'Espagne, plus facilement achetable en France.

Bien que le cèdre d'Espagne soit conseillé pour la doublure interne, il est préférable de ne pas l'utiliser pour la structure externe. Sa tendance à « gonfler » (un peu) à force de réguler l'humidité entrainerait une déformation disgracieuse.

Plusieurs solutions esthétiques sont possibles pour le rendu extérieur, notamment le rendu en bois massif ou le rendu en MDF plaqué. Personnellement, et n'étant pas un pro du plaquage, je me suis tourné vers une structure en hêtre massif abouté, tout simplement pour son rendu que j'apprécie.

Le concept

En parcourant les nombreux modèles et déclinaisons des caves à cigares du marché, j'ai pu réaliser assez rapidement et facilement des plans de base. Le principe est relativement simple, il s'agit d'un coffret en bois avec un intérieur doublé.

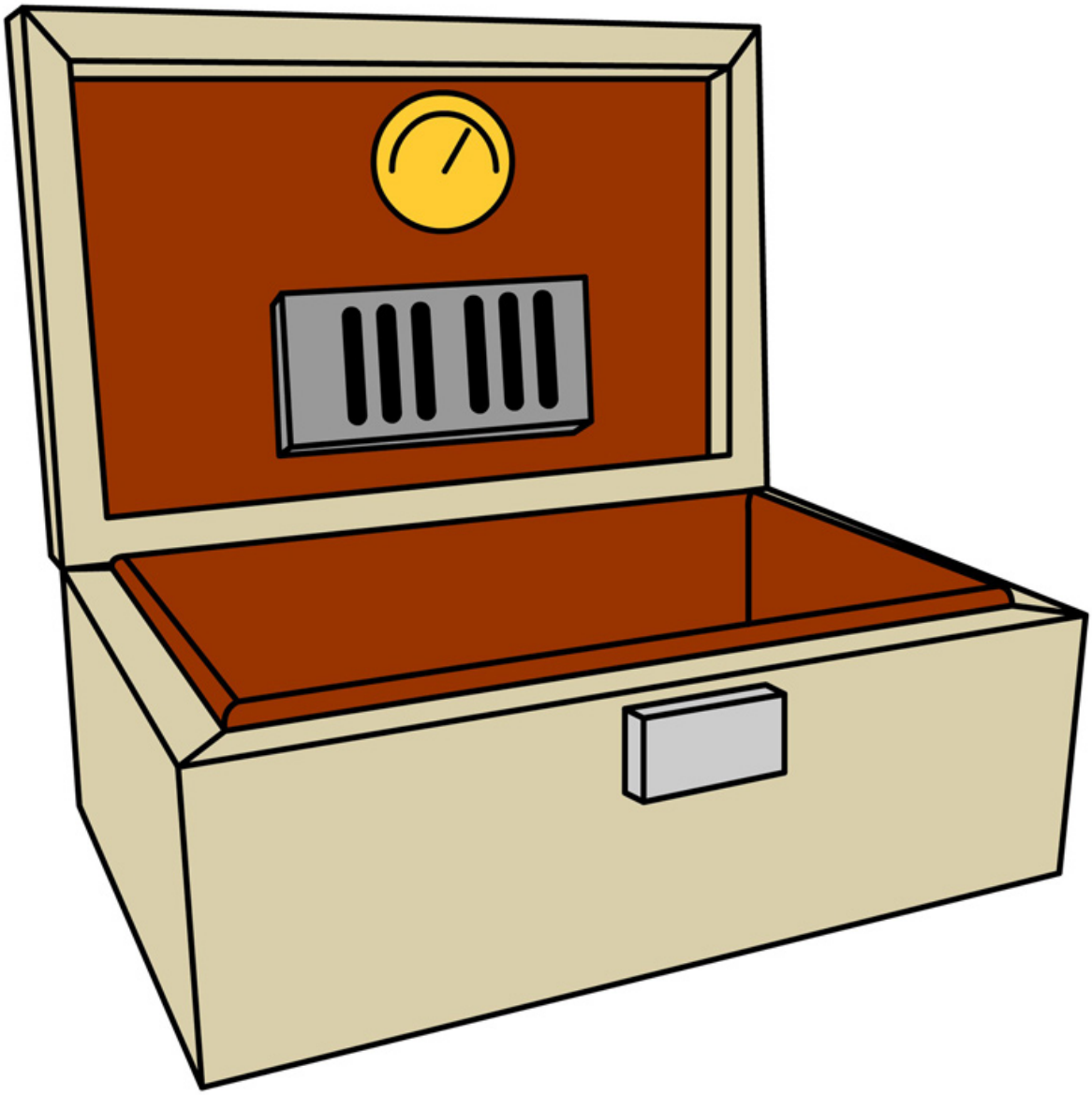


FIGURE 5.1 – Les plans de base

En optant pour un style simple et naturel, et pour une capacité d'environ 40 cigares, cela donnera :

Le petit plus qui peut être bien pratique : des charnières qui se bloquent à 95° environ pour maintenir le couvercle ouvert sans risquer de faire basculer la cave à cigares en arrière.

La réalisation

FIGURE 5.2 – Résultat final

FIGURE 5.3 – Les charnières auto-bloquantes à 95° environ

Matériel et matériaux

Pour réaliser notre cave à cigares, nous aurons besoin de :

- planches en cèdre d'Espagne (95mm de haut, 9mm de large et 900mm de long environ),
- plaquage en cèdre d'Espagne (2 morceaux de 290x250mm),
- un humidificateur,
- un hygromètre,
- du bois (massif ou MDF) pour la structure de la cave,
- deux charnières,
- un fermoir (facultatif).

Au niveau matériel, il nous faudra, électroportatif ou non :

- une scie (à main, sauteuse ou circulaire),
- de la colle à bois,
- un tournevis,
- du papier abrasif ou une ponceuse.

FIGURE 5.4 – Le matériel nécessaire

Les découpes

Pour la réalisation des coffrets en bois, plusieurs concepts sont possibles, avec plus ou moins de facilité et avec différents design :

- l'assemblage en onglet (angle à 45° au niveau des jointures) : esthétique et relativement simple à réaliser, il faut néanmoins couper avec une grande précision à 45° pour avoir de jolies jointures,
- l'assemblage queue droite (ou d'aronde) : style particulier et très fini, mais très long à préparer,
- l'assemblage à plat joint : le moins esthétique mais également le plus simple, on colle les planches champ contre plat,
- l'assemblage à feuillure : assez esthétique, mais nécessite un travail de rabotage assez minutieux.

J'ai choisi un assemblage en onglet pour ma cave à

cigares.

Voici les découpes pour l'assemblage en onglet :

Pour les mesures, je me suis basé sur les dimensions de la doublure intérieure puis j'ai tout simplement rajouté l'épaisseur du bois extérieur (dans mon cas, 18mm) pour obtenir les mesures des découpes.

Assemblage

Assemblage de la structure de base

Après avoir vérifié que mes découpes étaient correctes avec un assemblage à blanc de la structure extérieure et de la doublure en cèdre d'Espagne, j'ai réalisé l'assemblage définitif de la structure de base. Pour cela, il faut coller les bords extérieurs (sur les champs à 45°) et le socle de la cave à cigares avec de la colle à bois.

Il faut bien laisser sécher cette structure, celle-ci étant la base qui tiendra l'ensemble consolidé.

Assemblage du couvercle

De la même manière, j'assemble le couvercle avec la colle. Je renforce la partie qui supportera les charnières avec quelques vis afin de ne pas avoir de surprise, particulièrement avec les charnières auto-bloquantes qui infligeront un choc à chaque ouverture. Pour la « final touch » esthétique, j'ai directement intégré le plaquage en cèdre d'Espagne au niveau du couvercle.

Mise en place de la doublure intérieure

Si les découpes ont été suffisamment précises, on devrait pouvoir intégrer facilement les planches de cèdre d'Espagne découpées dans la structure de base (à vérifier à blanc avant collage).

Une fois toutes les retouches éventuelles effectuées, on colle le plaquage en cèdre d'Espagne sur le socle, puis on intégre la doublure intérieure.

Assemblage de la cave

Une fois les différents temps de séchage terminés (ne pas hésiter à laisser plus de temps qu'indiqué), on réalise l'assemblage de la base et du couvercle en vissant les charnières sur l'arrière de la cave.

Il ne reste plus qu'à mettre en place le fermoir qui donnera la touche finale esthétique à la cave.

Pour cela, on fixe, selon les modèles, avec :

- du collant double face,
- un système magnétique (prévoir éventuellement la pose invisible au moment de fixer le plaquage du couvercle).

Mise en place des éléments d'humidification

La structure de la cave à cigares étant maintenant terminée, il faut positionner l'humidificateur et l'hygromètre.

Et voilà le résultat final :

Je n'ai plus qu'à disposer les cigares et attendre (pas trop longtemps !) la prochaine dégustation :

Ce projet pourrait être (va être ?) poursuivi et amélioré ! Je pourrais le compléter par un système de surveillance du taux d'humidité créé avec Arduino, voire d'un humidificateur électronique, piloté également par Arduino ou système similaire.

Astuce : étalonner son hygromètre. Il est nécessaire, au moins une fois par an de ré-étalonner son hygromètre afin de s'assurer que celui-ci ne s'est pas déréglé. Pour cela, il suffit de l'enfermer dans un sac hermétique avec un bouchon (type bouteille d'eau) de sel humecté (attention, le sel doit faire une pâte et non se liquéfier). Au bout de 8 heures environ, l'hygrométre doit afficher 75% d'humidité relative.

Bonne conception/fabrication !

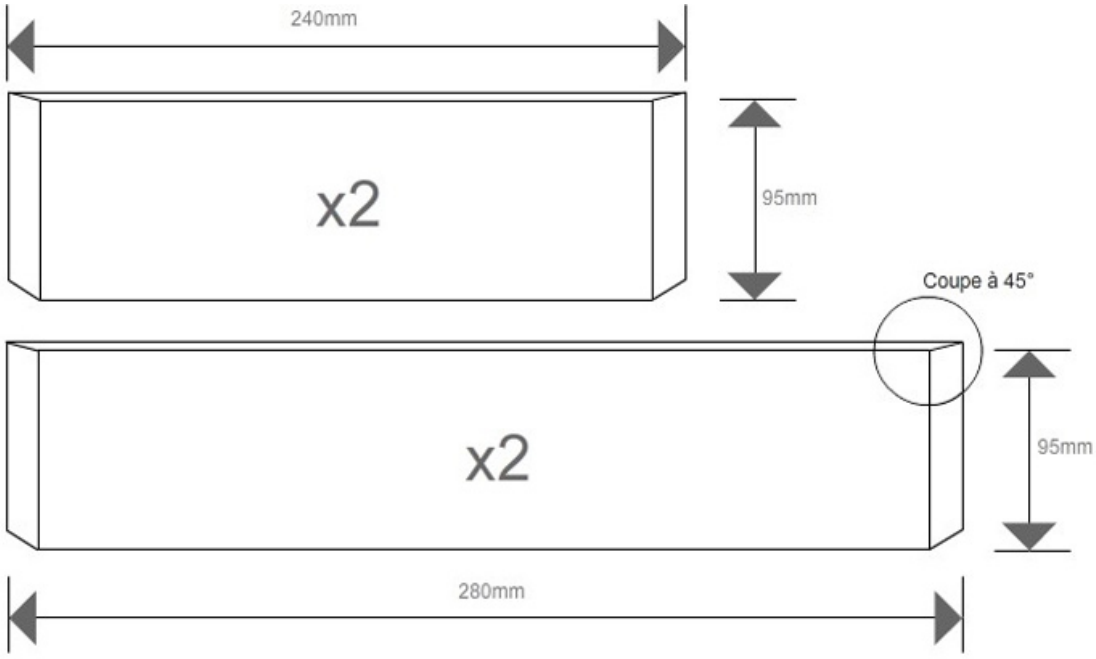


FIGURE 5.5 – Découpe de la doublure intérieure en cèdre d'Espagne

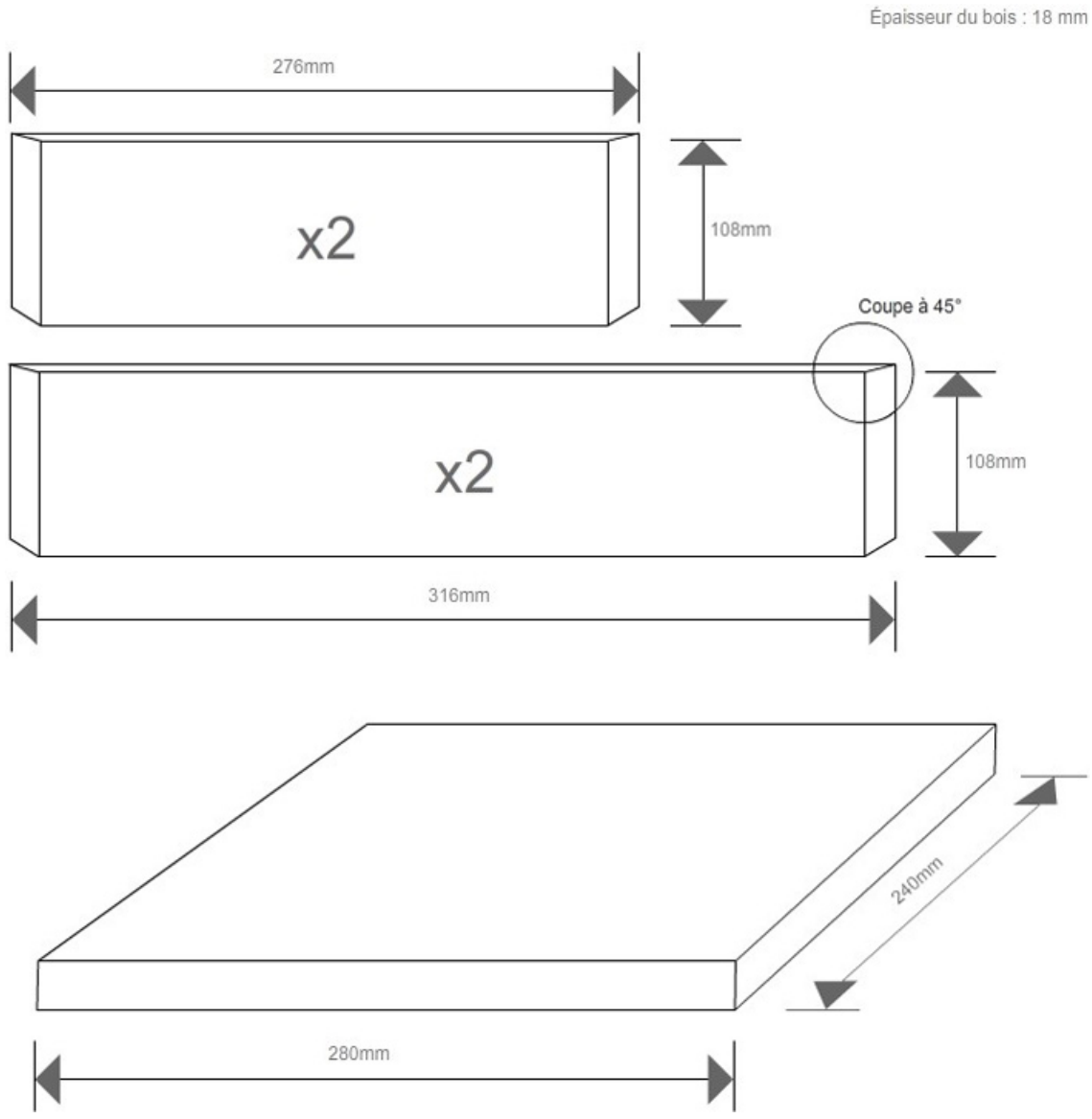


FIGURE 5.6 – Découpe de la structure extérieure et du socle

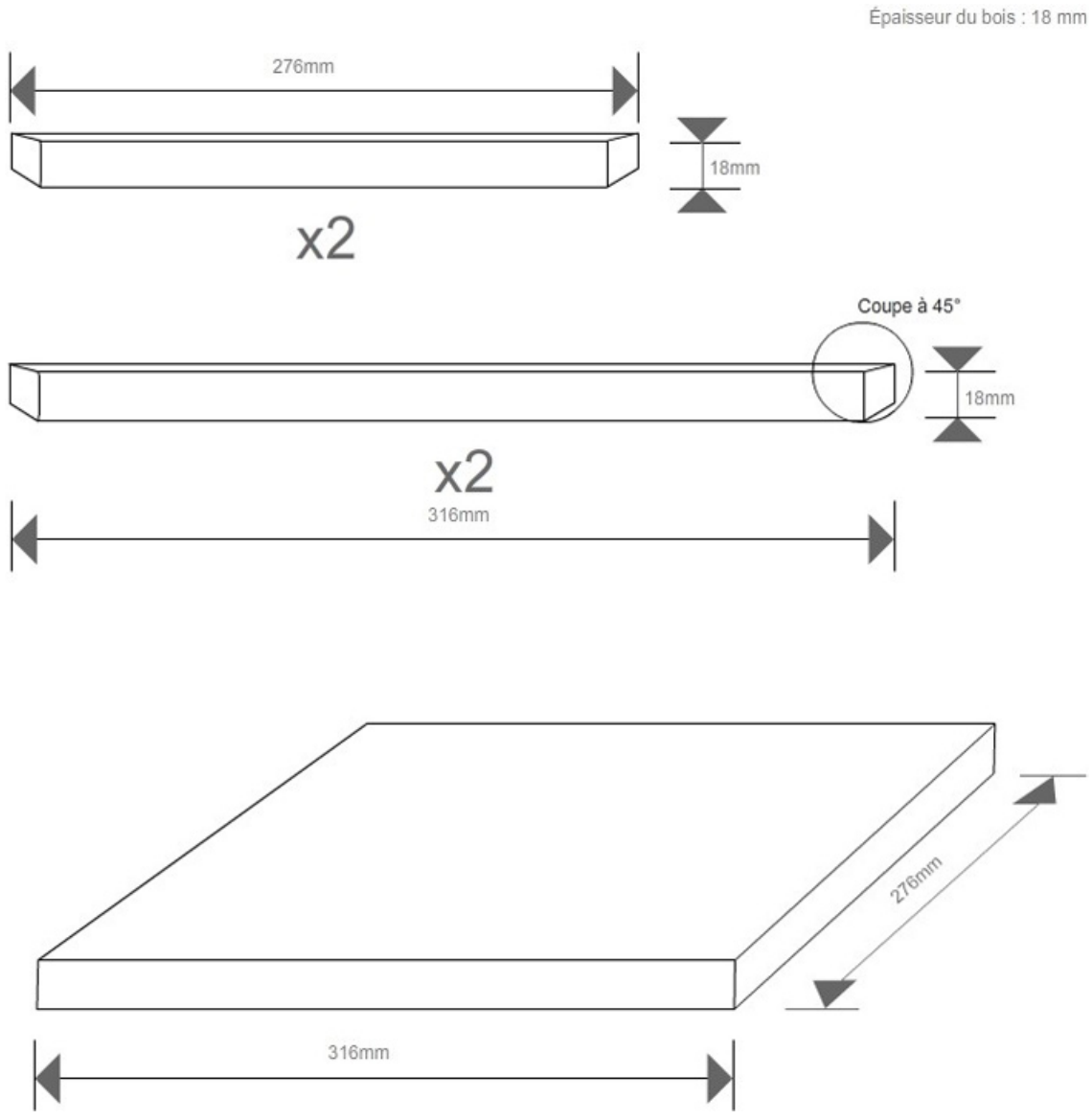


FIGURE 5.7 – Découpe du couvercle

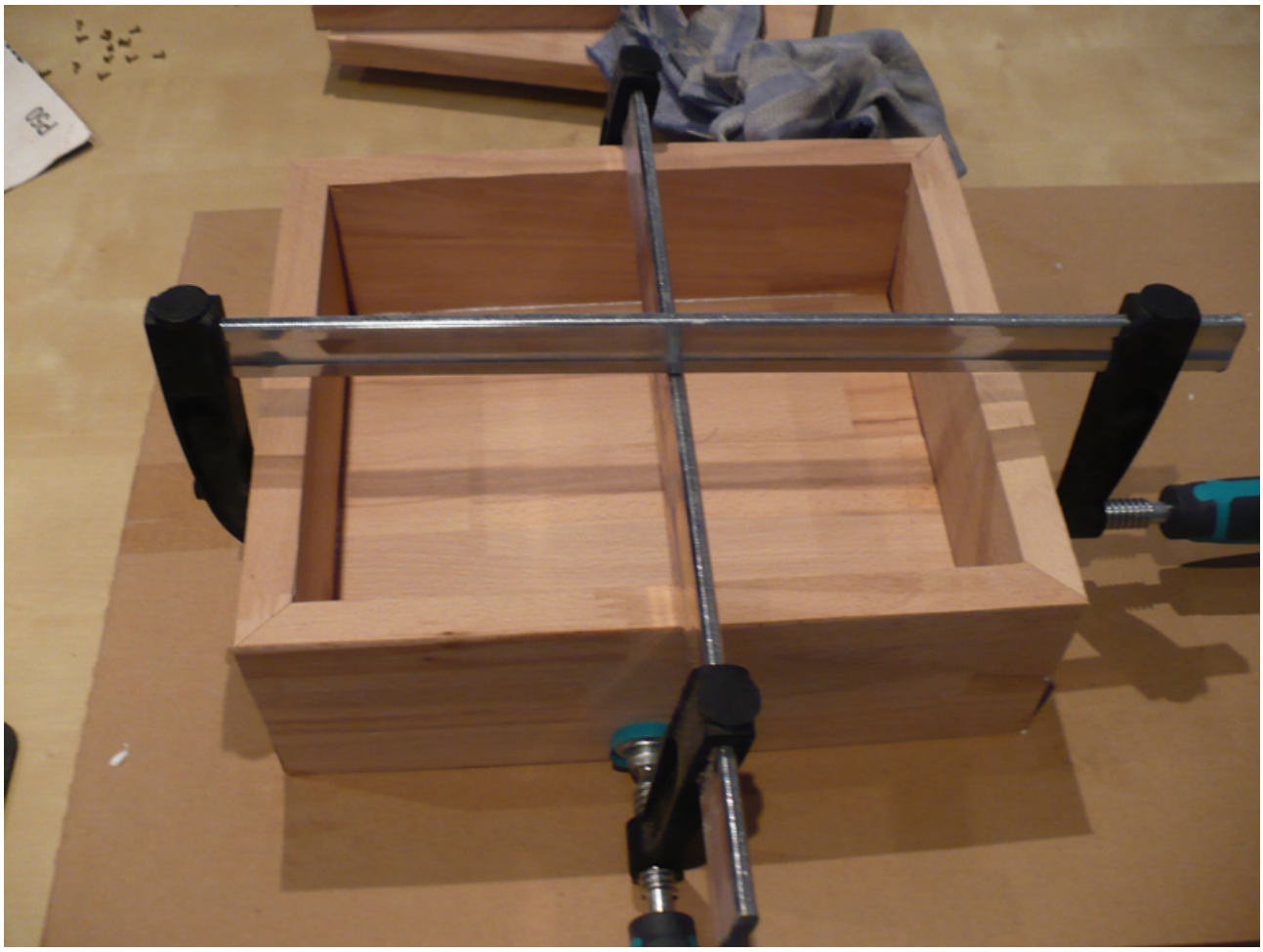


FIGURE 5.8 – Assemblage de la structure de base avec le socle

FIGURE 5.9 – Assemblage du couvercle

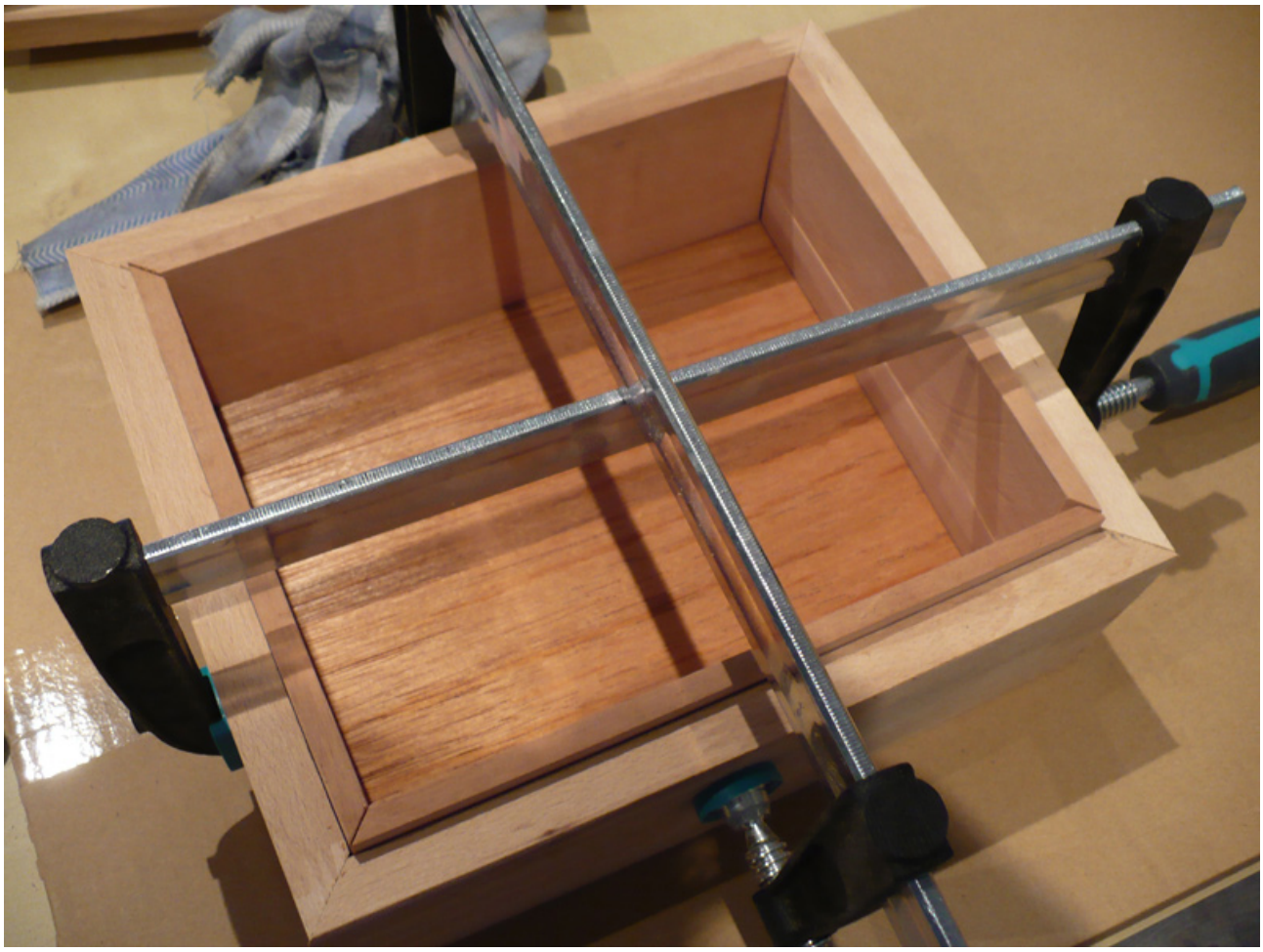


FIGURE 5.10 – Intégration de la doublure intérieure

FIGURE 5.11 – Mise en place des charnières

FIGURE 5.12 – La cave à cigares terminée et fonctionnelle

6. Éduquer pour militer

date 2013-08-01
category écologie,autre
level decouverte
author David Larlet

On nous invite à consommer de saison, à consommer bio, à consommer local dans le domaine de l'alimentaire. Pourrait-on en faire de même dans l'informatique ? Vous êtes-vous déjà posé la question de la distance qui vous sépare des développeurs de vos solutions techniques ? De leur relation avec le monde de l'Open-Source ? De l'importance que vous pouvez avoir dans la viabilité de leur structure ? **Le choix de ses outils et prestataires web peut-il être militant ?**

Peu de personnes ont conscience de la somme de travail qu'il y a derrière un service en ligne ou un logiciel. Demandez autour de vous pendant la prochaine soirée/réunion familiale et vous vous apercevrez probablement que les ordres de grandeurs sont assez délirants, certaines personnes n'ayant même pas conscience qu'il y ait des humains derrière tout ça. Nous — développeurs — avons un devoir éducatif à assumer afin que le grand public réalise que des individus ont apporté leurs compétences et leur temps pour que ces outils existent. Il s'agit aussi de (ré)acquérir une certaine considération, voire une dignité, dans ce processus.

De quels outils disposons-nous pour cela ? Le Web à large échelle mais aussi et surtout le pair-à-pair comme il en a été question pour l'adoption des logiciels libres : chaque geek installant chez ses parents, montrant à ses proches, dépannant parfois. Il pourrait en être de même pour conseiller un prestataire ou un produit ou au moins pour donner les pistes permettant de creuser la question. La proportion de personnes utilisant l'outil informatique est suffisamment importante maintenant pour que chaque groupe puisse discuter de sa propre expérience et échanger sur ce qu'il a pu observer.

Figure 6.1 – Donne fumier.

Au-delà de l'approche éducative, il est nécessaire de faire prendre conscience qu'un tissu économique est concerné, avec son lot de pratiques douteuses, d'entreprises qui jouent le jeu et d'idéalistes incompris. Dans une société capitaliste on a coutume de dire qu'il faut militer avec son argent en achetant des produits issus d'entreprises conformes à nos valeurs. Il pourrait en être de même avec les achats informatiques en intégrant la composante éthique au même titre que celle de la qualité. Cela demande de répertorier les pratiques des entreprises de façon désintéressée et d'en conserver un historique.

Enfin il pourrait être intéressant de recueillir les avis des personnes ayant vraiment été confrontées aux prestataires, pas forcément pour décrire les mauvaises expériences mais bien pour que cet écosystème soit tiré vers le haut… si suffisamment de personnes y prêtent attention.

7. Valise Ghetto Blaster

date 2013-08-01
author Tarek Ziadé
category électronique,informatique
level vulgarisation, moyen
licence By-Sa-3.0

Figure 7.1 – Valise Ghetto Blaster

Après le Juke Box du premier numéro, j'avais envie de pousser un peu plus le projet pour faire un Ghetto Blaster qui puisse streamer de la musique via le wifi de la maison, avec un son digne de ce nom.

Il existe des solutions commerciales comme le Bose Soundlink ou le Jawbone JAMBOX — mais il faut compter un budget de 400 euros — et franchement, quand on sait ce qu'il y a à l'intérieur de ces enceintes, on paye surtout le design et la marque. De plus, une enceinte amplifiée à base de Raspberry-PI offre beaucoup plus de possibilitées vu que c'est programmable.

Enceintes & ampli

En fouillant dans ma cave, j'ai trouvé deux enceintes trois voies Panasonic de 100 Watts (en vrai 25W RMS) qui étaient dans mon ancienne voiture, et une valise en aluminium qui contenait des outils. C'est la valise générique qui est souvent vendue avec une perceuse ou d'autres outils. Elle en jette mais elle est en carton et a juste une mince couche d'aluminium par dessus. La caisse de résonance parfaite pour un prix avoisinant les 10 euros...

Ni une, ni deux, j'ai vidé la valise et retiré sa doublure caoutchouteuse, puis percé deux trous pour les enceintes à l'aide d'un gros cutter. La couche d'aluminium est juste comme il faut pour tenir les enceintes avec des petites vis.

Pour l'amplification j'ai opté pour un petit amplificateur Lepai 20 Watts 2 voies qui permet quelques réglages basses/medium/aigues, et dont le panneau frontal se dévisse — payé 25 euros sur Amazon.

Ça m'a permis de remettre ce panneau sur la valise et de revisser le corps de l'ampli à l'intérieur. Bien sûr, le top du top aurait été de fabriquer mon propre ampli, mais je n'en suis pas encore à ce niveau là.

FIGURE 7.2 – Enceinte vissée dans la valise

FIGURE 7.3 – Ampli Lepai 20w.

FIGURE 7.4 – Panneau de l'ampli repassé à l'extérieur.

Après deux points de soudures pour relier du fil électrique entre les enceintes et l'ampli, j'ai pu essayer le son avec un lecteur mp3.

Le choc. *un son excellent, dépassant de loin la qualité de mes enceintes de salon.* La valise vombrie avec les basses et les rend bien, sans aucune vibration parasite.

J'ai fais des essais en rembourrant la valise avec de l'isolant, mais c'est moins bien — ça étouffe le son et tue les basses.

Alimentation

Avoir une alimentation stable & portative était le gros challenge de ce projet : comment faire pour fournir les 12v que l'amplificateur requiert, ainsi que les 5v pour le Raspberry ?

Au début j'ai pensé à équiper la valise d'une batterie au plomb qui est similaire à celles qui équipent les voitures, mais j'ai très vite déchanté sur le prix et le poids — et aussi le coté pas très écologique.

J'ai ensuite pensé à monter moi-même un train de 4 batteries Li-Po 3.6v, mais ce genre de montage est assez technique car il faut s'assurer que toutes les batteries sont toujours chargées au même niveau et aussi ne jamais descendre en dessous d'une certaine charge. Et puis bon, c'est dangeureux tout ça...

Grâce à Jonathan j'ai finalement trouvé une Li-Po de 12v et 6.8A sur AliBaba très compacte et légère. Guère plus grosse que celle que l'on trouve dans les avions ou voitures radio-commandé, avec toutes les protections nécessaires.

FIGURE 7.5 – Li-Po 12v 6800mA

La batterie se cale parfaitement au dessus de l'ampli, avec un peu de carton entre les deux pour éviter une surchauffe.

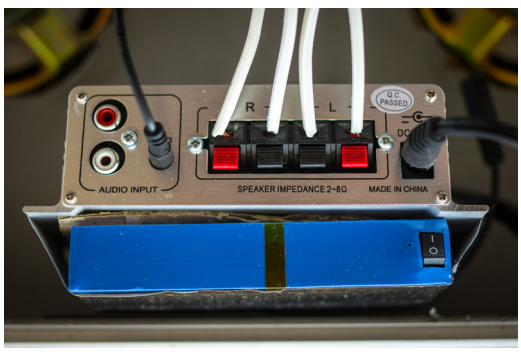


FIGURE 7.6 – L'ampli et la batterie calés dans la valise.

Reste à fabriquer un petit régulateur pour ajouter une deuxième sortie de 5v à la batterie.

Régulation de tension

Pour transformer une tension de 12v en 5v il y a deux méthodes : utiliser une séries de résistances pour simplement diviser le voltage, ou utiliser un semi-conducteur spécialisé comme le LM1117 qui, accompagné de quelques condensateurs, va faire tout le boulot proprement. La deuxième méthode est beaucoup plus fiable et évite les variations de tension, qui peuvent être problématiques lorsqu'on alimente un Raspberry.

Le montage, expliqué ci-dessous, est très simple : le régulateur reçoit les 12 volts sur une patte et renvoie 5v sur l'autre. La troisième patte est la masse. Attention, bien lire le datasheet pour ne pas se tromper de pattes - elles sont différentes en fonction des modèles.

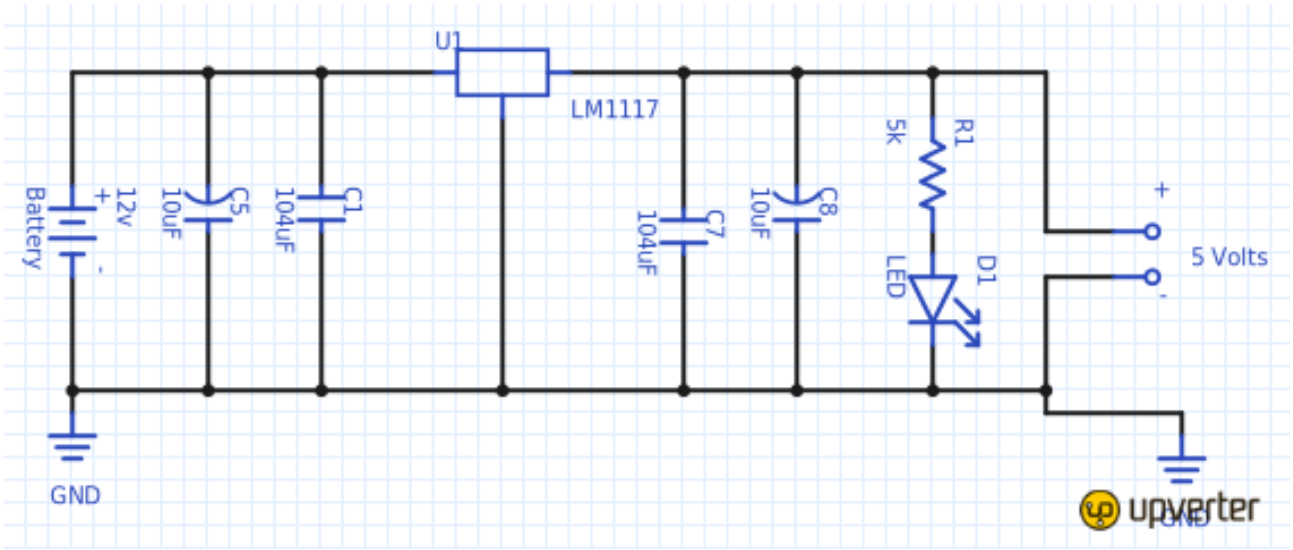


FIGURE 7.7 – Schéma du circuit de régulation de tension.

De chaque côté du montage il y a deux condensateurs, qui stabilisent le circuit. Si vous montez des condensateurs céramiques, attention au sens : ils sont polarisés.

Enfin, une LED est placé sur la partie 5v, avec une petite résistance, histoire de montrer que le circuit tourne.

En utilisation, le régulateur chauffe beaucoup puisqu'il dissipe l'excès de tension, et il peut être équipé d'un petit radiateur à visser — j'en ai récupéré un sur une vieille plaque électronique pour ce montage.

Après quelques soudures, un magnifique régulateur de tension !

Avec un son assez fort, l'ensemble tient 3 à 4 heures, ce qui n'est pas mal du tout.

Les deux évolutions possibles pour la partie alimentation sont :

— un afficheur de charge restante, qui peut être réalisé avec un chip LM3914 qui est capable de gérer jusqu'à 10 LEDs, et ce joli afficheur 10 leds ;
— un bouton pour éteindre le système sans arrêter brutalement le Rapsberry-Pi. Ce petit circuit peut être réalisé en pilotant l'extinction du Raspberry via son port GPIO comme comme expliqué ici, et un timer 555 pour l'extinction finale de la batterie.

Wifi

Le but de la valise étant de se connecter au réseau de la maison pour servir d'enceinte sans fil, il fallait une puce wifi. La puce AirLink que j'avais utilisé lors du jukebox précedent marchait mal car il s'agit d'un simple dongle USB. En effet, lorsque je fermais la valise, le signal se coupait assez vite puisque l'aluminium de la valise fait office de cage de farraday.

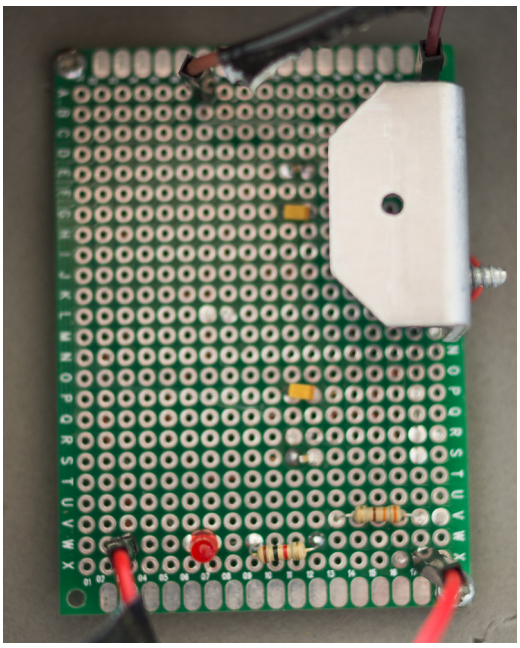


FIGURE 7.8 – Régulateur 12v -> 5v. Le LM1117 est masqué par le radiateur.

J'ai donc opté pour un dongle Logilink WL0151 avec une antenne wifi integrée qui dépasse à l'extérieur de la valise.

Vu que c'est du Ralink, cette puce est plug-n-play sur Raspbian.

Carte son

Une autre fonctionnalité de la valise est de pouvoir jouer de la musique provenant d'une source extérieure comme un lecteur MP3. J'ai donc acheté une carte son Dacomex USB avec une entrée.

Cette carte est reconnue tout de suite sur la Raspbian comme périphérique audio USB, et en modifiant le fichier **/etc/asound.conf** comme suit :

```
pcm.!default {
    type hw
    card 1
    device 0
}
```

Elle sera utilisé comme carte son par défaut.

Logiciels

Coté logiciel, après avoir déployé une Raspbian de base, j'ai suivi la même installation que pour le précédent Jukebox, expliquée ici puis en lieu et place de l'application JukeBox, j'ai déployé le logiciel Mopidy comme suit :

```
wget -q -O - http://apt.mopidy.com/mopidy.gpg | sudo apt-key add -
sudo wget -q -O /etc/apt/sources.list.d/mopidy.list http://apt.mopidy.com/mopidy.list
sudo apt-get update
sudo apt-get install mopidy
```

FIGURE 7.9 – Le dongle Wifi avec antenne extérieure.

FIGURE 7.10 – Carte son Dacomex

Mopidy est un serveur de musique qui permet de jouer de la musique de plusieurs sources différentes : fichiers sur le disque, radios internet, tout type de stream compatible.

Mopidy se base sur un serveur MPD (Music Player Daemon) et est compatible avec tous les clients MPD du marché — il y en a pour Android, GNU/Linux, Mac OS X, Windows.

En d'autre termes, la valise pourra être pilotée via un téléphone une tablette ou un laptop !

J'ai installé l'extension Mopidy-Spotify qui permet à Mopidy de se connecter à un compte Spotify pour streamer de la musique.

```
sudo apt-get install libspotify12 python-spotify
```

Il suffit ensuite de configurer Mopidy en ajoutant une section **spotify** dans le fichier **~/.config/mopidy/mopidy.conf** :

```
[spotify]
username = myusername
password = mysecret
```

Le user et mot de passe s'obtiennent dans l'interface du site de Spotify, en y ajoutant un nouveau device.

Plug-and-play

Le seul petit problème du système est qu'il faut connaître l'adresse IP de la valise sur le réseau de la maison pour pouvoir la piloter.

Le plus simple est de lui attribuer une adresse fixe mais le plus sexy serait d'avoir en plus un accès hot spot sur la valise, pour que chacun puisse s'y connecter pour jouer de la musique.

Il paraît qu'il est possible de configurer certaines puces WIFI pour qu'elles fonctionnent en point d'accès **et** qu'elles se connectent à une borne wifi aussi. Je n'y suis pas arrivé.

En attendant, j'ai opté pour une solution plus geek : j'ai modifié le script de démarrage de Mopidy pour que la valise dicte à haute voix son adresse IP en utilisant eSpeak, installable avec le nom de paquet éponyme.

Quand j'allume ma valise, elle me dit :

> I am ready to play music, my ip address is 192.168.0.20

Conclusion

La valise fonctionne plutôt bien, mais il manque les petits détails pour en faire un produit fini, comme l'affichage de la batterie restante ou le bouton ON/OFF qui respecte la séquence de halt du Raspberry. Le problème de l'IP est aussi un peu pénible.

Mais tout ces problèmes peuvent être résolus, donc je suis assez content du résultat.

Peut-être que j'écrirais un prochain article sur ces upgrades.

8. Lectures de l'été

date 2013-08-01

category électronique,informatique

level decouverte

author Tarek Ziadé

Voici une petite sélection de livres pour l'été. Dans ce numéro, beaucoup d'électronique !

L'électronique en pratique

FIGURE 8.1 – Charles Platt/Eyrolles

Eyrolles lance une nouvelle collection Serial Makers, dédiée « à l'univers des Fab Labs, de l'électronique, du DIY et de l'impression 3D ».

Un des premiers livres dans cette collection est la traduction du livre de Charles Platt, intitulé « Make : Electronics (Learning by Discovery) » en anglais.

La version en français est « L'électronique en pratique » et la traduction est de bonne facture et agréable à lire.

Quand au contenu du livre, je pense que c'est l'un des meilleurs livres actuel pour démarrer en électronique. Il propose de découvrir les principes fomdamentaux de l'électronique par des petites experiences ludiques et simple à réaliser. Aucun bagage en physique ou électronique n'est nécessaire, le livre va réexpliquer simplement les notions de base entre deux expériences, sans que cela ne devienne barbant.

Je suis moi-même assez novice en électronique et je prend beaucoup de plaisir à utiliser ce livre, qui prend le temps d'expliquer en détail comment fonctionne les composants manipulés.

Même si l'on fini par faire la plupart de nos montages complexes avec des micro-controlleurs, il est très intéressant grâce à ce livre de comprendre comment faire des montages avec quelques composants passifs et de simples timers. La section sur les timers 555 est à ce titre extraordinaire. Elle m'a donné envie de refaire quelques-un des mes montages Arduino en remplaçant le micro-controlleur par des timers 555 et des transis-

tors.

L'électronique en pratique est à acheter les yeux fermés.

Encyclopedia of Electronic Components Vol. 1

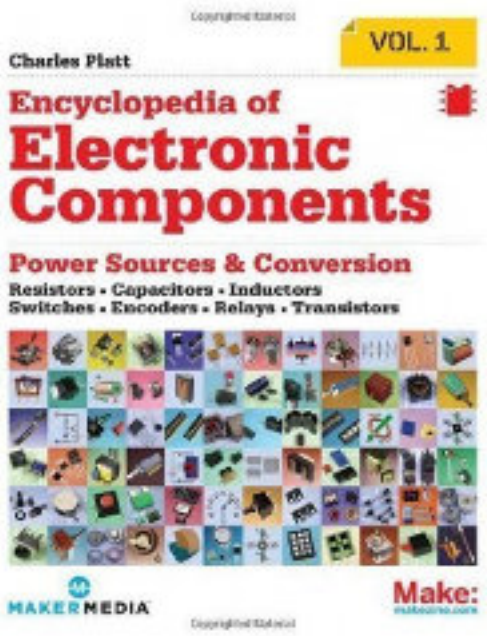


FIGURE 8.2 – Charles Platt / O'Reilly Media

J'ai tellement aimé L'électronique en pratique que lorsque j'ai vu l'Encyclopedia of Electronic Components Vol. 1 au stand O'Reilly d'Europython en Italie, je me suis jeté dessus.

Ce livre est un recueil de composant eléctroniques regroupés par thèmes : résistances, condensateurs, inducteurs, interrupteurs, encodeurs, relais et enfin transistors.

Dans chacun de ces thèmes, une série de composants est présentée. Platt explique pour chacun comment ils fonctionnent, quelle sont leur usages et leur limites, via une section nommée « Ce qui peut mal se passer ».

Un bon exemple est la partie sur les différent types de condensateur : elle est vraiment pédagogique et part du principe que le lecteur n'a aucune connaissance préalable. On comprend le fonctionnement de ces composants, leur utilité, et pourquoi il existe des condensateurs de différent types.

Encyclopedia of Electronic Components Vol. 1 est un excellent complément des datasheets que l'on trouve sur le net, et aussi un bon compagnon du livre précédent.

Je ne crois pas qu'il existe en français par contre, mais j'achèterais le volume 2, c'est sûr.

Raspberry-Pi — Prise en main et premières réalisations

Tout comme Eyrolles, Dunod commence à s'intéresser de prêt au mouvement DIY et sort un ouvrage original sur le Raspberry-Pi intitulé : **Raspberry-Pi — Prise en main et premières réalisations**, écrit par Christian Tavernier, auteur prolifique de livres sur les micro-controlleurs PIC et l'Arduino chez le même éditeur.

Si vous êtes un novice complet, ce livre est parfait pour vous, car l'auteur va vous guider à travers toutes les étapes de découverte et d'installation du Raspberry, et aussi vous donner des notions de programmation en Python.

Si vous êtes déjà familier avec Python et le monde de

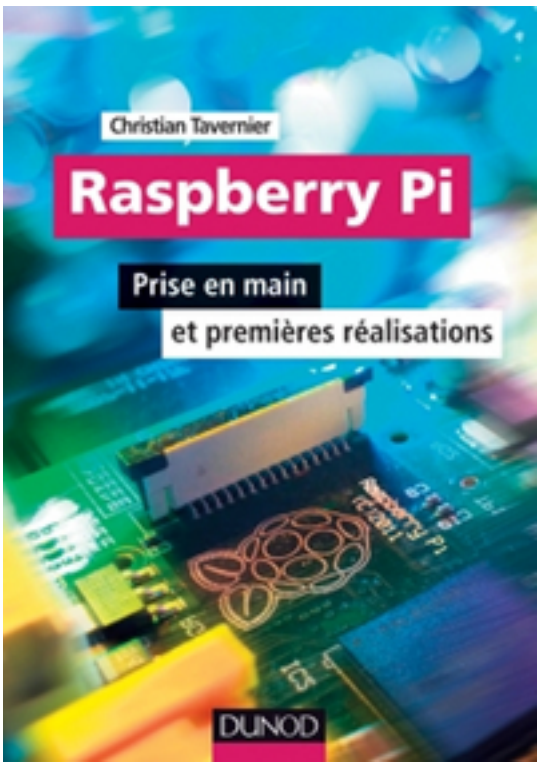


FIGURE 8.3 – Christian Tavernier/Dunod

l'Arduino, cet ouvrage ne vous apportera pas grand chose, si ce n'est la dernière section sur des idées de montages à brancher sur le GPIO.

Ceci étant dit, le niveau du livre est très bon et tout le monde y apprendra quelque chose : l'auteur ajoute une bonne valeur ajouté grâce à ces connaissances en électronique. J'y ai découvert par exemple que le régulateur de tension pour mon ghetto blaster pouvait être remplacé par un régulateur avec un bien meilleur rendement (et qui chauffe moins).

Ca m'a donné envie d'acheter d'autres livres de cet auteur.

Pour débuter avec le Raspberry, je recommande **Raspberry-Pi — Prise en main et premières réalisations**

9. Un cadre numérique pas comme les autres

date 2013-08-01

category électronique,informatique

level moyen

author Fabien Batteix

Pour ce 3ème numéro de Fait Main j'avais le choix entre plusieurs projets tous plus fous les uns que les autres. Finalement celui qui a été retenu est un projet de cadre numérique, mais pas n'importe quel cadre.

Pour ce projet j'ai décidé d'innover en réalisant mon propre cadre numérique « Do It Yourself ». Bien sûr celui-ci ne sera pas aussi conventionnel qu'un cadre du commerce. Le mien sera réalisé avec des matrices leds bicolores et du bois. Celui-ci aura pour but de servir de base pour des applications temps réel diverses et variées que je vous présenterai par la suite.

Des matrices de leds ? Oui, mais lesquels ?

Le choix des matrices de led était bien évidemment un point clé pour ce projet. Mon choix s'est arrêté sur des matrices de leds bicolores (pixels rouge et vert) de 8×8 pixels.

Pourquoi ce choix ?

— tout simplement parce que de telles matrices ne coutent quasiment rien sur ebay, dealextreme... contrairement à des matrices de leds RGB ;
— il est très simple de trouver des circuits imprimés pour en connecter et contrôler un grand nombre à la fois ;
— cela donne un côté très DIY au résultat final, ce qui n'est pas négligeable.

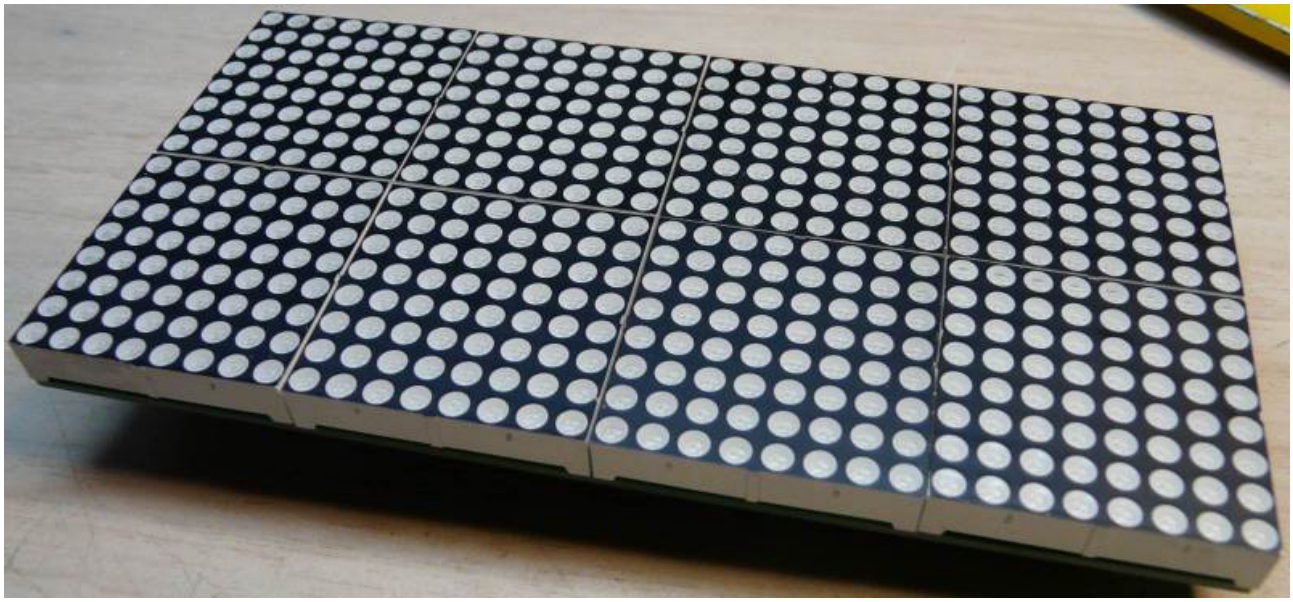


Figure 9.1 – Avant d'une matrice de leds une fois montée.

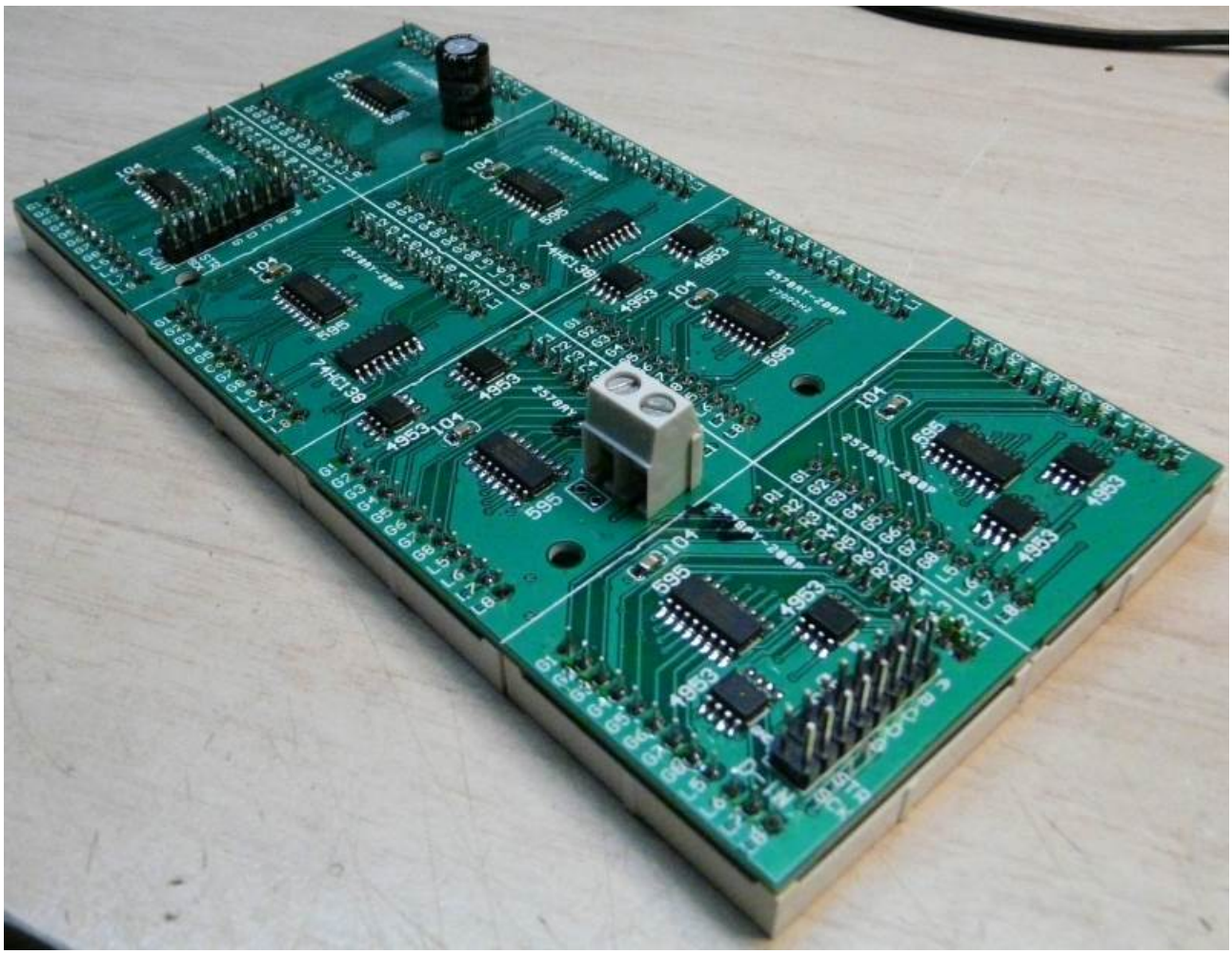


FIGURE 9.2 – Arrière d'une matrice de leds une fois montée.

Après quelques heures de recherche je suis tombé sur des kits à monter soi-même de 32×16 pixels (8 matrices par kit) du fabricant « DIYTJ » sur DealExtreme (sku.202393). Ces kits, bien que comportant des composants CMS, sont relativement simples à monter, même sans connaissances poussées en soudure. De plus leur prix unitaire m'a permis d'en acheter une douzaine et ainsi de former une matrice « géante » de 96×64 pixels au total.

Le cadre

Comme précisé plus haut le cadre entourant les matrices de leds est réalisé en bois.

FIGURE 9.3 – Sans cadre le résultat ne fait pas très professionnel.

Ce cadre a pour but de maintenir les matrices de leds en place et de donner un aspect visuel plus propre au montage. Un cadre numérique n'en serait pas un sans cadre :)

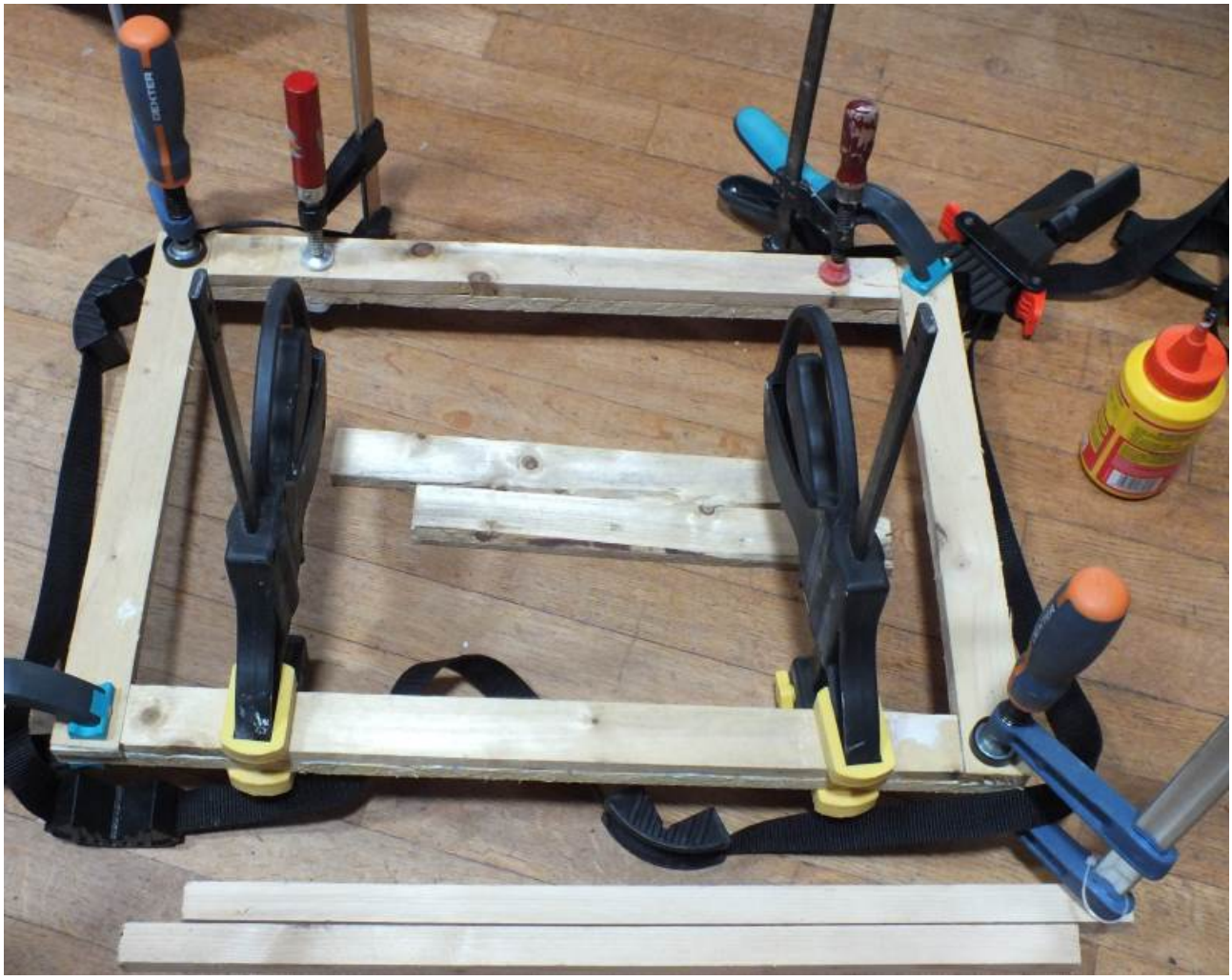


FIGURE 9.4 – Collage de la base du cadre. Ne vous inquiétez pas mon parquet n'a reçu aucune coulure de colle, ou presque.

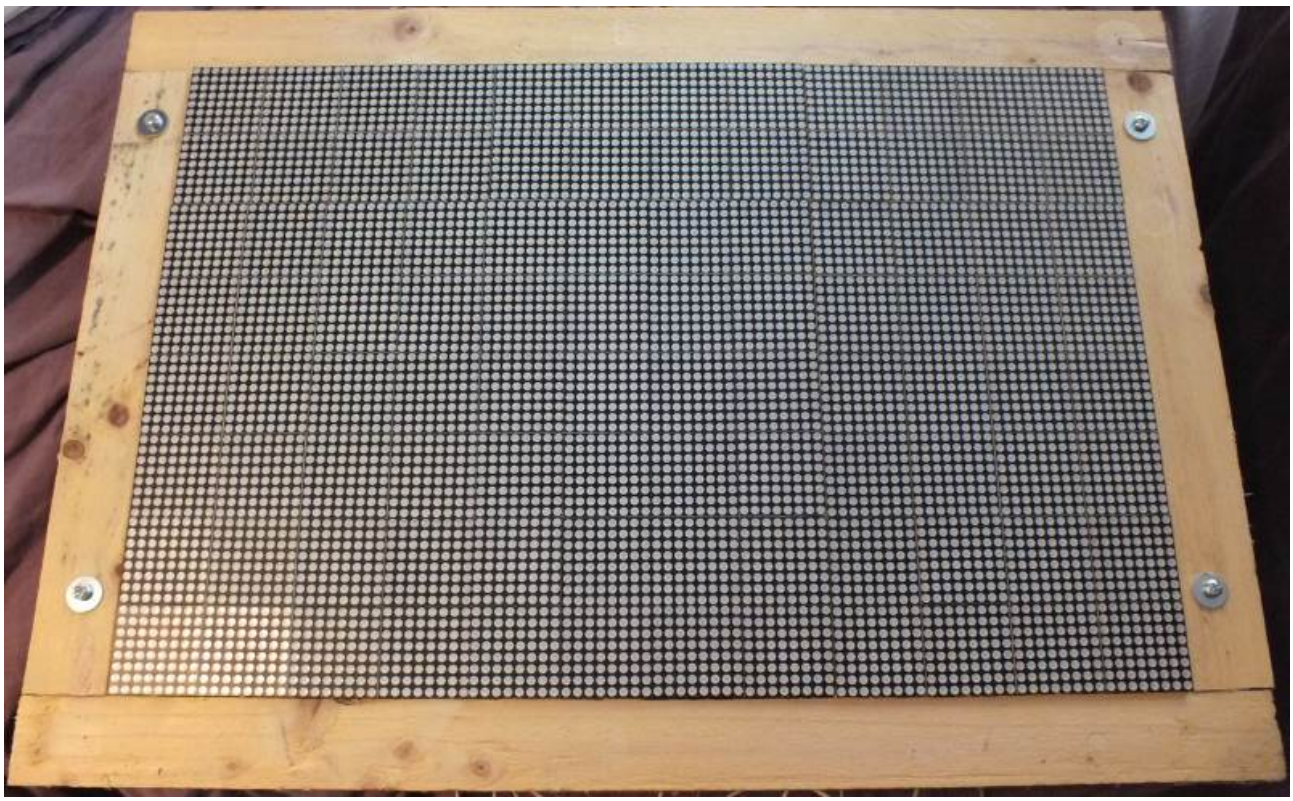


FIGURE 9.5 – La base du cadre une fois la colle sèche et les matrices en place.

Le fond du cadre est réalisé en bois de pin tout ce qui a de plus classique. Deux épaisseurs de planches sont collées en chevauchement pour maximiser la résistance du cadre. Le tout a ensuite été fixé ensemble avec de la colle blanche, des serre-joints et un serre-cadre pour l'équerrage.

Par-dessus la base du cadre vient se poser une plaque de verre synthétique ayant pour but de plaquer les matrices de leds. Les vis sur les quatre coins du cadre ont elles pour but de fixer par l'arrière deux baguettes de bois servant de pressoir pour maintenir les matrices en place. Ces vis ont bien évidement été incrustées dans le bois pour éviter que rien ne dépasse avant la pose définitive du plexiglas.

La face avant du cadre a été réalisée en medium (une espèce de bois aggloméré au grain très fin). Cette face avant comporte des chanfreins sur les bords intérieurs et extérieurs pour un meilleur fini. De même qu'une rainure cachée sur le dessous pour maintenir la plaque de plexiglas. (La plaque de plexiglas est maintenue en place par la seule pression de la face avant sur le cadre de base et un peu de joint à baignoire)

Après un dernier petit coup de peinture noir satiné le cadre est fini !

Principe de fonctionnement des matrices et câblage

Ces matrices de leds sont contrôlées au moyen d'un port SPI un peu spécial. Le connecteur utilisé par ces matrices est un classique connecteur 2x16 broches au pas 2.54mm.

Sur ce connecteur on retrouve les lignes suivantes :

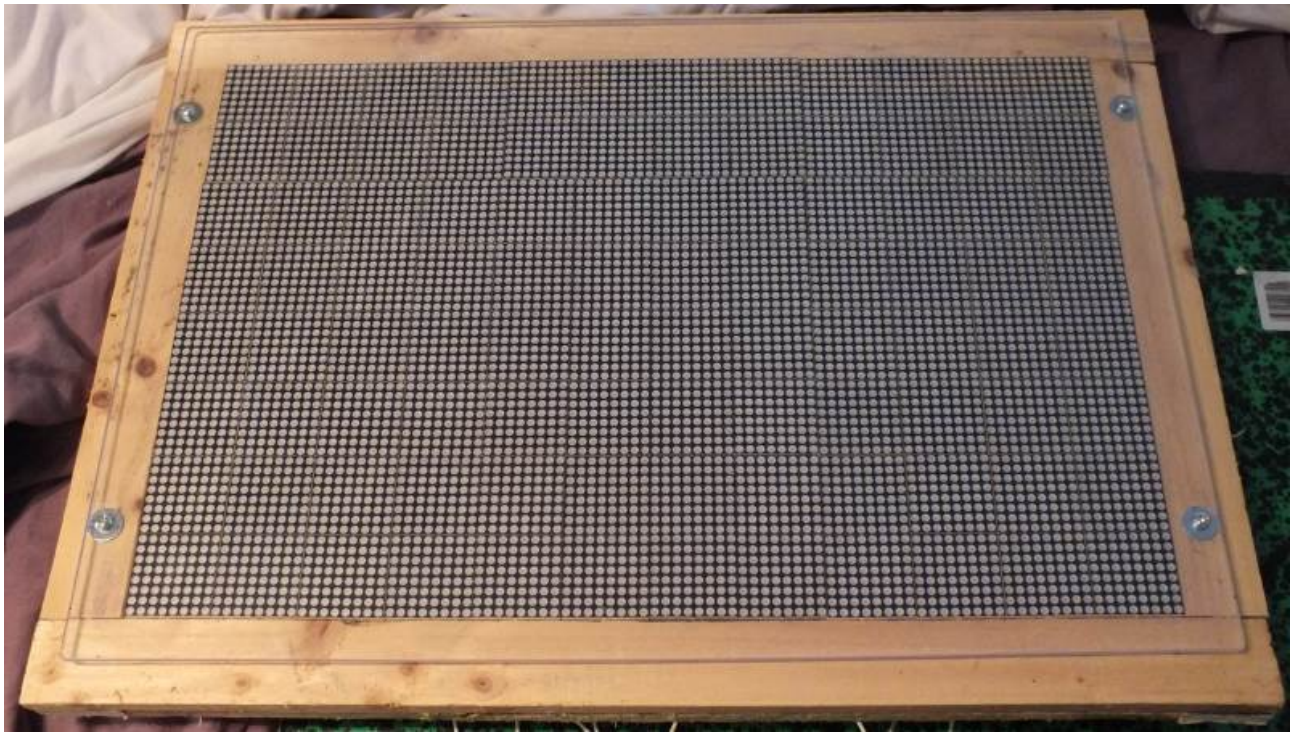


FIGURE 9.6 – La base du cadre avec la plaque de Plexiglass.

FIGURE 9.7 – Face avant du cadre en medium

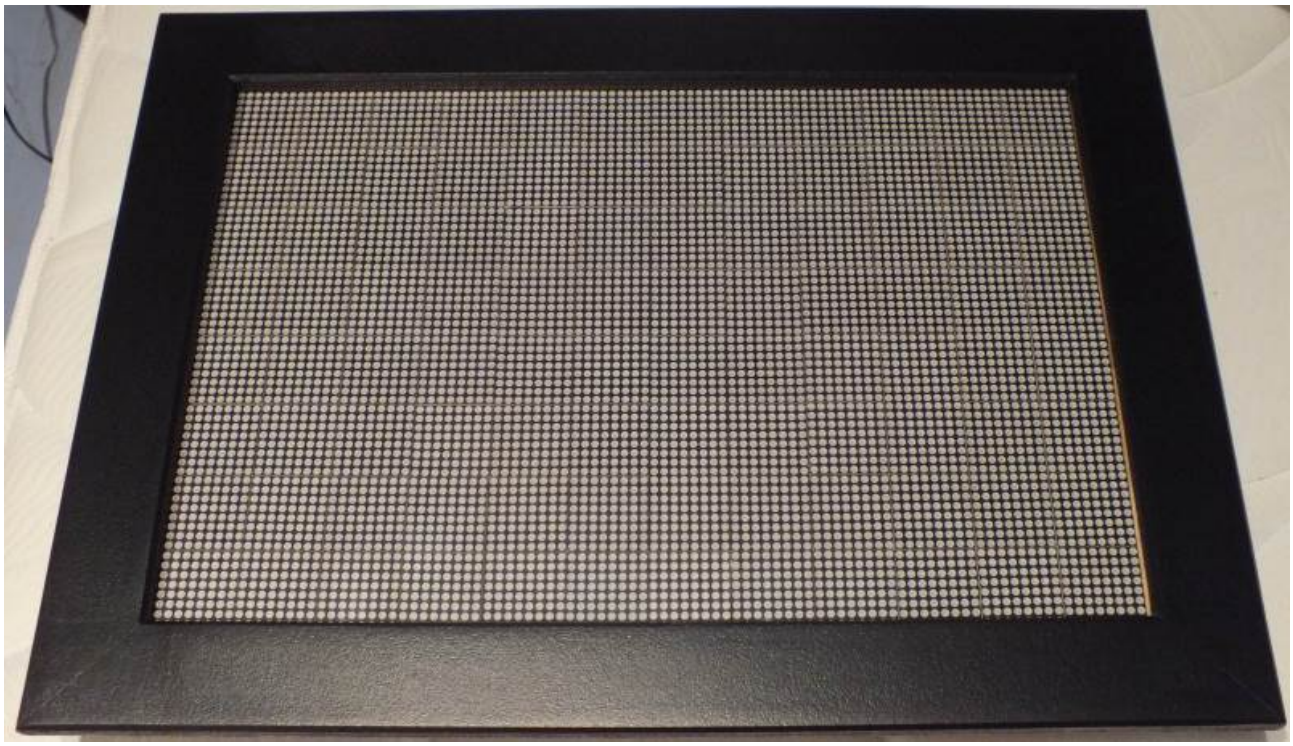


FIGURE 9.8 – Un petit peu de peinture et hop !

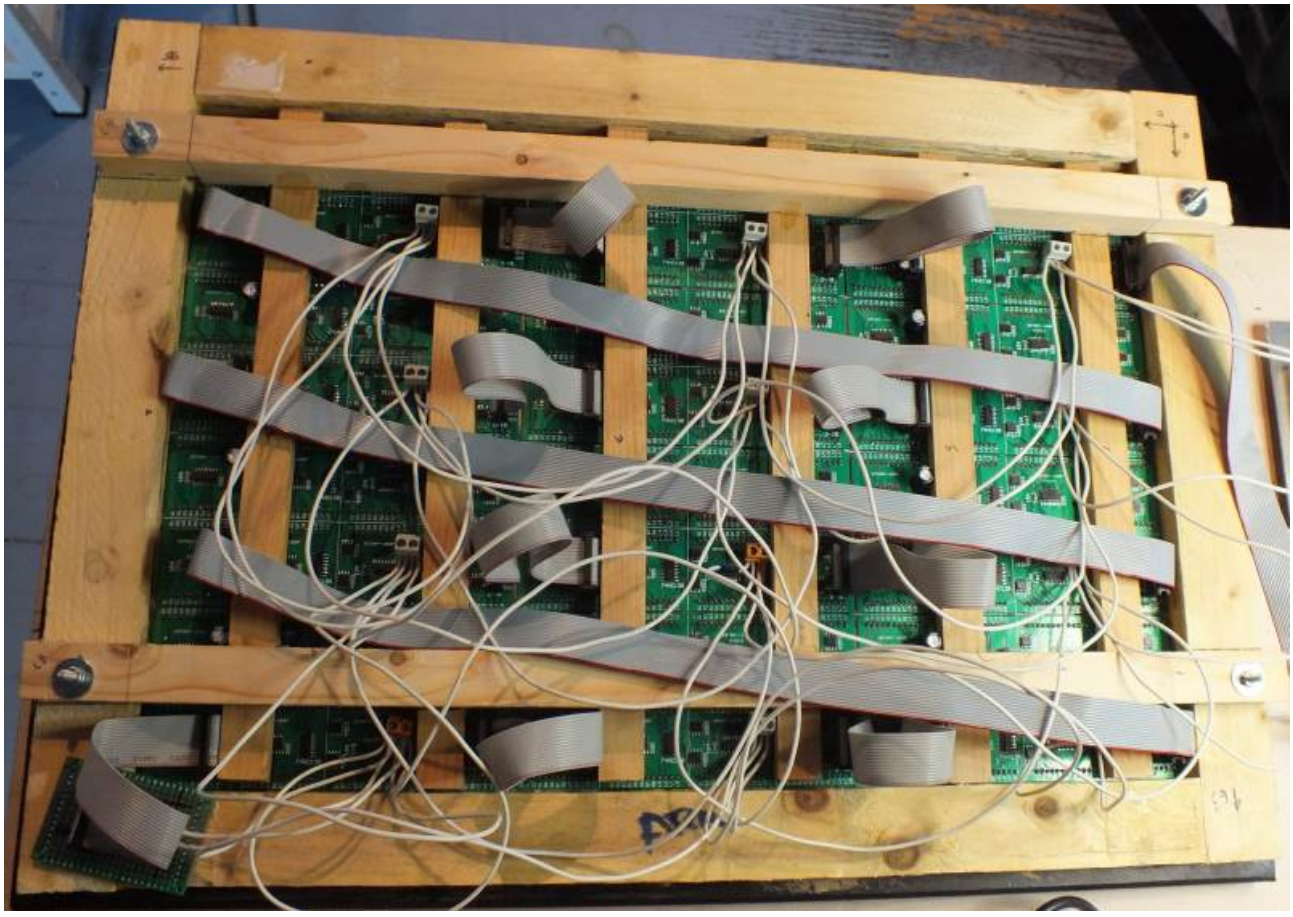


FIGURE 9.9 – Le rangement est mon maitre mot, ou pas.

GND	A
GND	B
GND	C
OE	D
R	G
nc	nc
GND	STR
GND	SCK

Détails :

- GND : masse (0v) ;
- nc : non connecté ;
- A, B, C, D : choix de la ligne à afficher (voir plus bas pour plus de détails) ;
- R, G : entrée de données pour le rouge et le vert (équivalent MOSI en SPI) ;
- STR : latch (permet de mettre en mémoire les données transmises) ;
- SCK : signal d'horloge commun pour les lignes R et G.

Comment marchent ces matrices ?

En réalité ces matrices de leds sont des clones très légèrement modifiés des matrices de leds d'ancienne génération du (très connu) fabricant « SureElectronics ». Par chance j'ai pu mettre la main sur un pdf expliquant le fonctionnement de ces "vieilles" matrices de leds, nommé « LCD matrix display driver – DE-DP029~033_Ver1.0_EN ».

Une copie du pdf est disponible sur ma dropbox, au cas où la version d'origine disparaitrait.

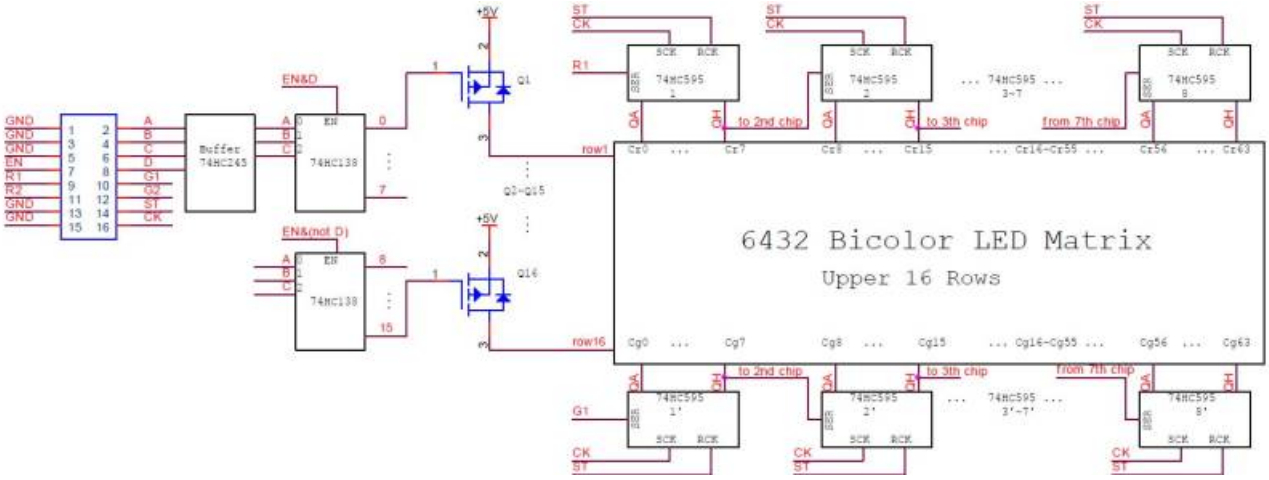


FIGURE 9.10 – Schéma d'origine tiré du pdf de « SureElectronics »

Le principe de fonctionnement est relativement simple :

- Un décodeur binaire -> décimal (74HC138) permet de sélectionner la ligne (entre 0 ~ 15) à afficher ;
- Deux séries de 4 registres à décalage (74HC595) permettent d'allumer/éteindre les pixels voulus de la ligne sélectionnée (« 0 » = allumé, « 1 » = éteint) ;
- L'affichage est multiplexé (affichage ligne par ligne) et doit être rafraichi à 960Hz (soit 60 images par secondes) pour un bon rapport scintillement/luminosité/consommation.

Le problème avec ces matrices c'est qu'il y a deux entrées séries (R et G) pour un seul même signal d'horloge (CK sur le schéma, SCK sur la carte). Il est donc impossible d'utiliser un port SPI matériel, la seule solution possible est donc d'utiliser du SPI software, beaucoup plus lent et couteux en ressource CPU...

Concrètement comment je fais pour allumer mes pixels ?

Prenons une seule matrice de 16×32 pixels. Cette matrice est constituée de 16 lignes de 32 pixels horizontaux, organisés en 4 blocs de 8 pixels. Chaque pixel horizontal est un bit que l'on envoie en SPI (communication série avec horloge synchrone). Ici pour afficher une ligne il faut donc envoyer 4 octets, que ce soit sur le signal de données R (rouge) ou G (vert).

Pour que l'affichage soit correct il faut envoyer les données de chaque ligne tout les 1/60ème de secondes (per-

sistance rétinienne), soit une ligne tous les 1/960ème de seconde (= 16 lignes×60Hz).

Les matrices en question utilisent des registres à décalage montés vers la gauche. Le premier octet envoyé se retrouve donc à droite de la matrice puis est décalé vers la gauche. La seule explication logique à cela serait que le concepteur de la carte ait pris un repère (X = 0, Y = 0) en bas à droite, contrairement au reste des développeurs qui utilisent un repère (0, 0) en haut à gauche.

Dans le cas où l'on souhaite câbler plusieurs matrices en chaine il faut faire un peu de gymnastique pour envoyer les données dans le bon ordre tout en conservant un repère (0, 0) classique. Il faut donc envoyer les blocs de 4 octets de chaque matrice, en commençant par la dernière matrice de la ligne. Ainsi pour envoyer les données d'une ligne de 64 pixels (soit 2 matrices chainées) il faut envoyer les octets dans l'ordre 5, 6, 7, 8, 1, 2, 3, 4.

Quand je vous disais qu'il fallait aimer les casses tête je ne vous avais pas menti ;)

Programme de démonstration

FIGURE 9.11 – Après tant de réflexions il est grand temps de voir ce que ça donne vous ne trouvez pas ?

Programme n°1 : les bases

Avant de courir il faut savoir marcher, ici c'est pareil, avant de vouloir faire des trucs compliqués il faut savoir afficher au moins un pixel.

La première étape pour cela est d'avoir un microcontrôleur pour contrôler la matrice. Lors de mes premiers essais j'utilisais une carte Arduino UNO mais celle-ci n'étant pas adaptée pour la suite (pas assez de RAM) j'ai dû changer mon fusil d'épaule.

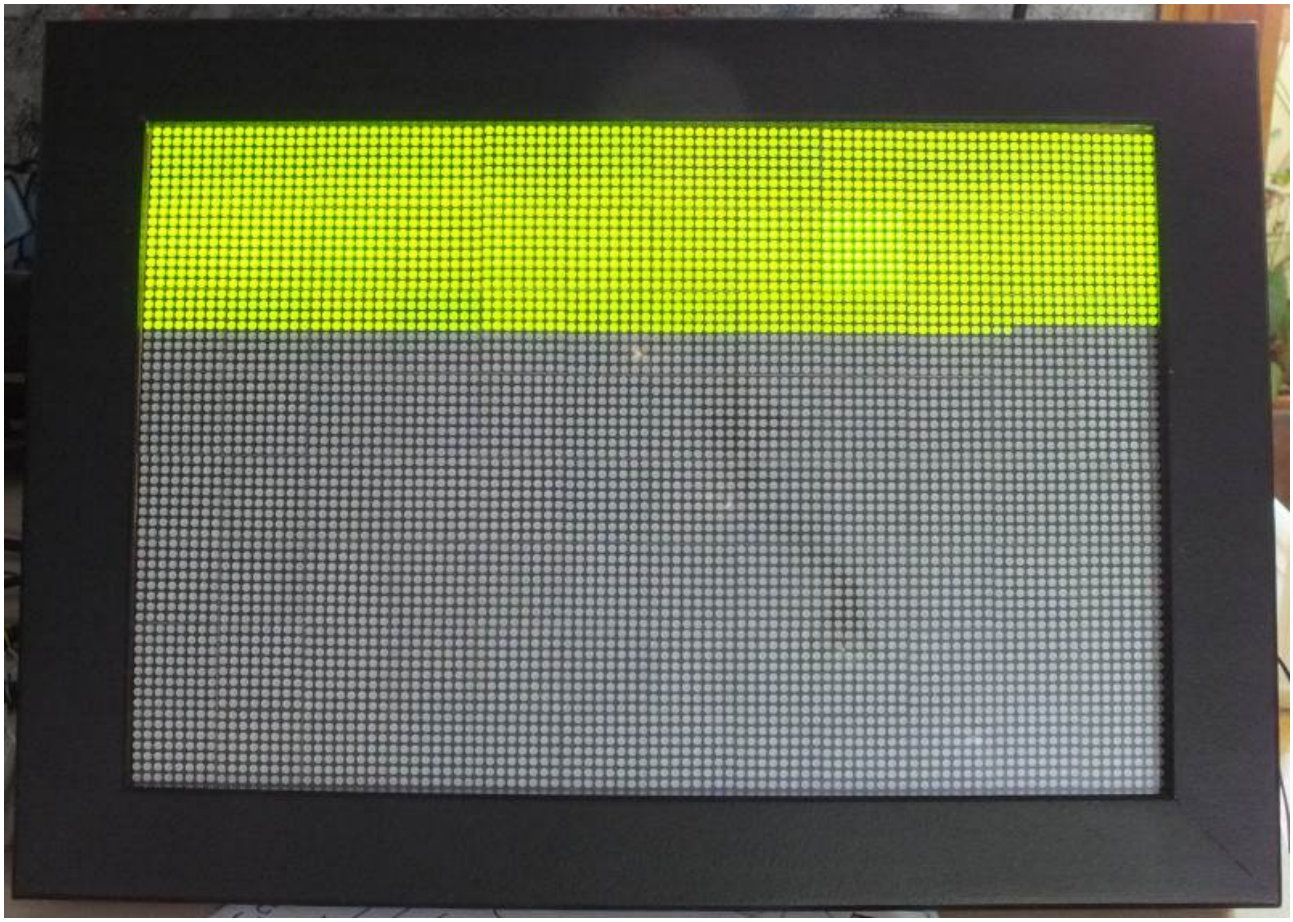


FIGURE 9.12 – Programme de base, rempli l'écran pixel par pixel

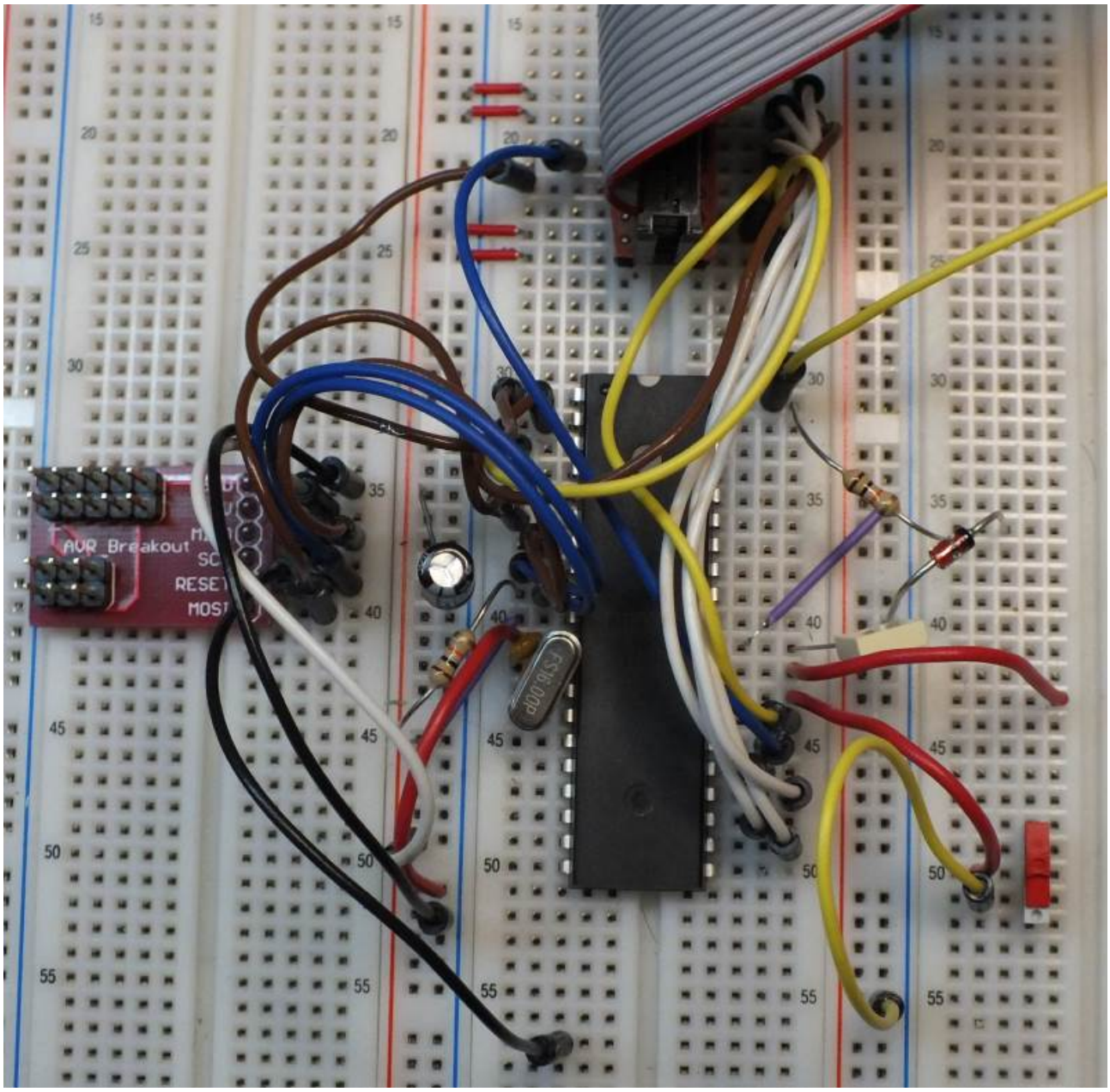


FIGURE 9.13 – Rien ne vaut une breadboard, des fils et un ATmega !

Bien qu'ayant laissé tomber l'idée d'utiliser une carte Arduino je ne suis pas allé chercher loin. Pour contrôler ma matrice de led j'ai utilisé un ATmega1284p (tournant à 16MHz), le même microcontrôleur que dans les cartes Sanguino.

Le montage est relativement trivial, celui ci se compose de :

- un ATmega1284p,
- un quartz à 16MHz + deux condensateurs de 22pF,
- une résistance de 10K sur la broche RESET,
- un condensateur de découplage (100nF) sur le +5v.

Pour me faciliter la vie j'ai aussi câblé un connecteur ICSP pour breadboard (fabriqué par Sparkfun) et un connecteur 2×16 broches pour la nappe reliant le montage à la matrice.

Le code de base est composé de seulement 6 fonctions et de quelques « define », rien de bien extraordinaire.

Le câblage des différentes lignes de contrôle est défini dès les premières lignes du code. Deux ports sont utilisés : le port C et le port B, ces deux ports sont disponibles sur l'ATMega1284p, mais aussi sur l'ATmega328p que l'on retrouve dans les cartes Arduino UNO. Et oui j'ai aussi pensé aux Arduinistes ;)

Au niveau des fonctions bas niveau on trouve :

- **void dualShiftOut(red, green)** : cette fonction permet d'envoyer deux octets suivant le même principe que la fonction Arduino « shiftOut » mais sur deux sorties (R et G) en simultané.
- **void lineShiftOut(line_red_buffer, line_green_buffer)** : cette fonction envoie une ligne complète en utilisant la fonction ci-dessus, elle est aussi responsable de l'ordre d'envoi des blocs de 4 octets.
- **ISR(TIMER2_COMPA_vect)** : cette fonction d'interruption est appelée tous les 1/960ème de seconde pour rafraichir l'affichage.

Cette fonction d'interruption fonctionne suivant le principe ci-dessous :

Vous remarquerez qu'en plus de faire l'affichage des lignes à intervalle régulier je gère aussi la rotation de deux buffers : un d'affichage et un de dessin. Cette méthode d'affichage s'appelle le « double buffering ».

Pour les fonctions haut niveau, elles sont au nombre de deux :

- **void setPixelAt(color, x, y, state)** : permet de manipuler l'état d'un pixel dans le buffer de dessin ;
- **getPixelAt(color, x, y)** : permet de lire l'état d'un pixel du buffer d'affichage.

(on écrit dans le buffer de dessin, mais on lit dans le buffer d'affichage.)

La fonction main() dans ce programme de base est réduite au strict minimum :

- initialisation des entrées / sorties,
- initialisation du timer permettant le rafraichissement à 960Hz,
- boucle infinie remplissant pixel par pixel l'écran.

Programme n°2 : Game of life

<iframe width="800" height="450" src="//www.youtube.com/embed/C1d3ALiDpNY" frameborder="0" allowfullscreen></iframe>

Ce programme est mon préféré, il est très simple et se rapproche énormément du programme de base. Seule une fonction « getAliveNeighbourCount » (compte le nombre de cellules vivantes autour d'une cellule donnée) et une boucle de traitement ont été rajoutées. Le résultat est cependant extraordinairement complexe.

Programme n°3 : Transformée rapide de Fourrier (FFT)

<iframe width="800" height="450" src="//www.youtube.com/embed/qfsvMXD0wb5" frameborder="0" allowfullscreen></iframe>

Ce programme est le plus compliqué des trois. Il réalise en temps réel une transformée rapide de fourrier sur un signal audio et affiche les 96 premières bandes de fréquence qui vont servir au calcul (sur un total de 128 bandes).

À noter que le code associé au calcul de la transformée (écrit en assembleur AVR et optimisé pour travailler sur des nombres réels (transformée connu sous le nom

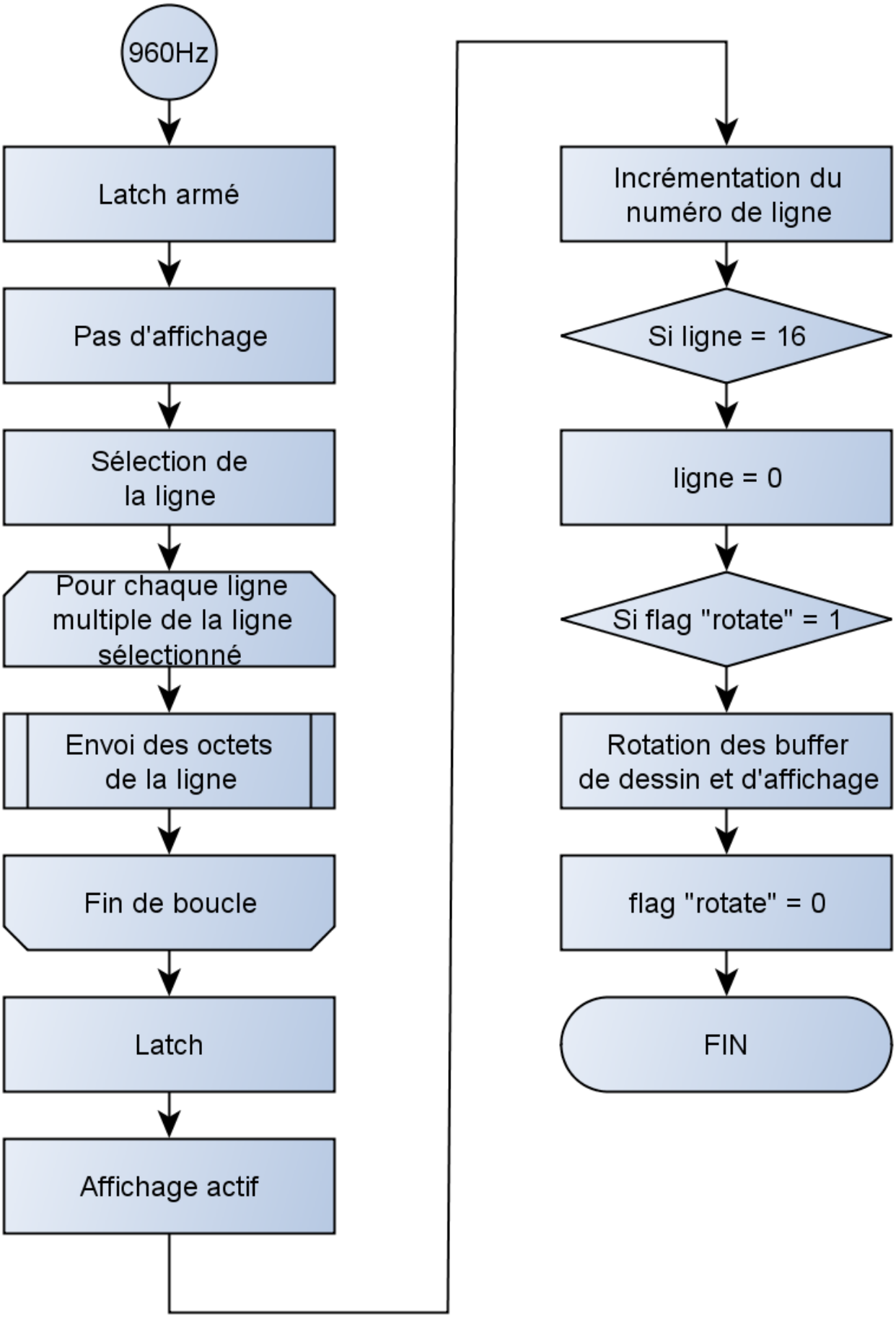


FIGURE 9.14 – Flowchart réalisé au moyen du logiciel yEd

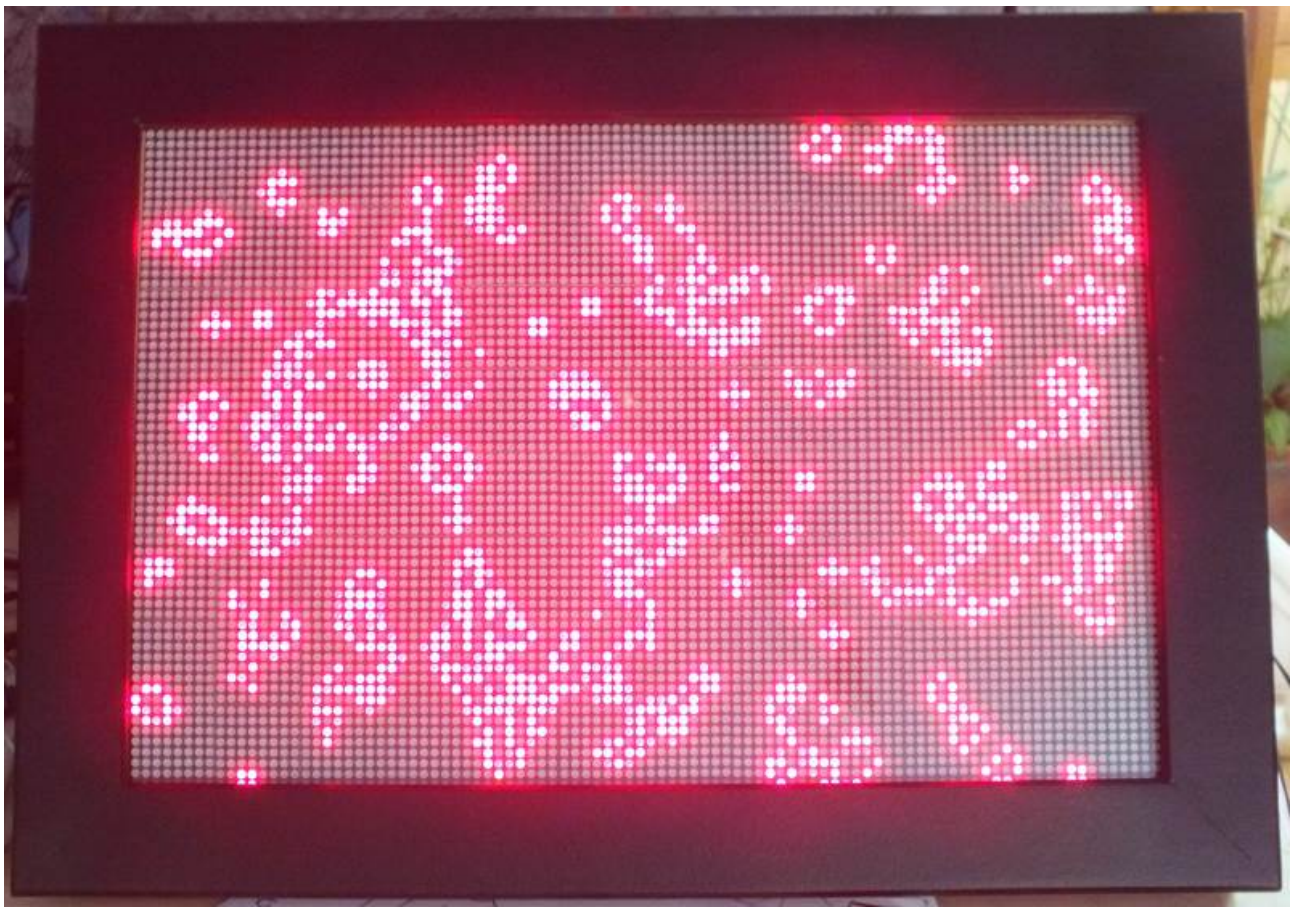


FIGURE 9.15 – Programme « Game of life »

de FHT)) n'est pas de moi mais de OpenMusicLabs. Écrire un tel algorithme n'est pas du tout de mon niveau mathématique. Je me suis juste contenté d'y apporter des modifications mineures pour que celle-ci compile avec ma version d'AVRGCC bien plus récente que celle fournie avec l'ide Arduino.

Le code reprend celui de base en ajoutant simplement une boucle de traitement/dessin par barre dans la fonction main() et une seconde fonction d'interruption par timer pour l'échantillonnage audio.

Pour fonctionner ce programme a besoin de capturer à intervalle régulier un échantillon audio. Les sorties de cartes son travaillant sur une plage de tension entre -1v et +1v il est nécessaire d'adapter ce signal avant de l'envoyer sur l'entrée analogique du microcontrôleur.

Pour ce faire j'utilise un classique montage amplificateur non inverse à ampli-op. Ce montage amplifie le signal 2.5 fois avant de le recentrer sur 2.5v au moyen d'un « bias ».

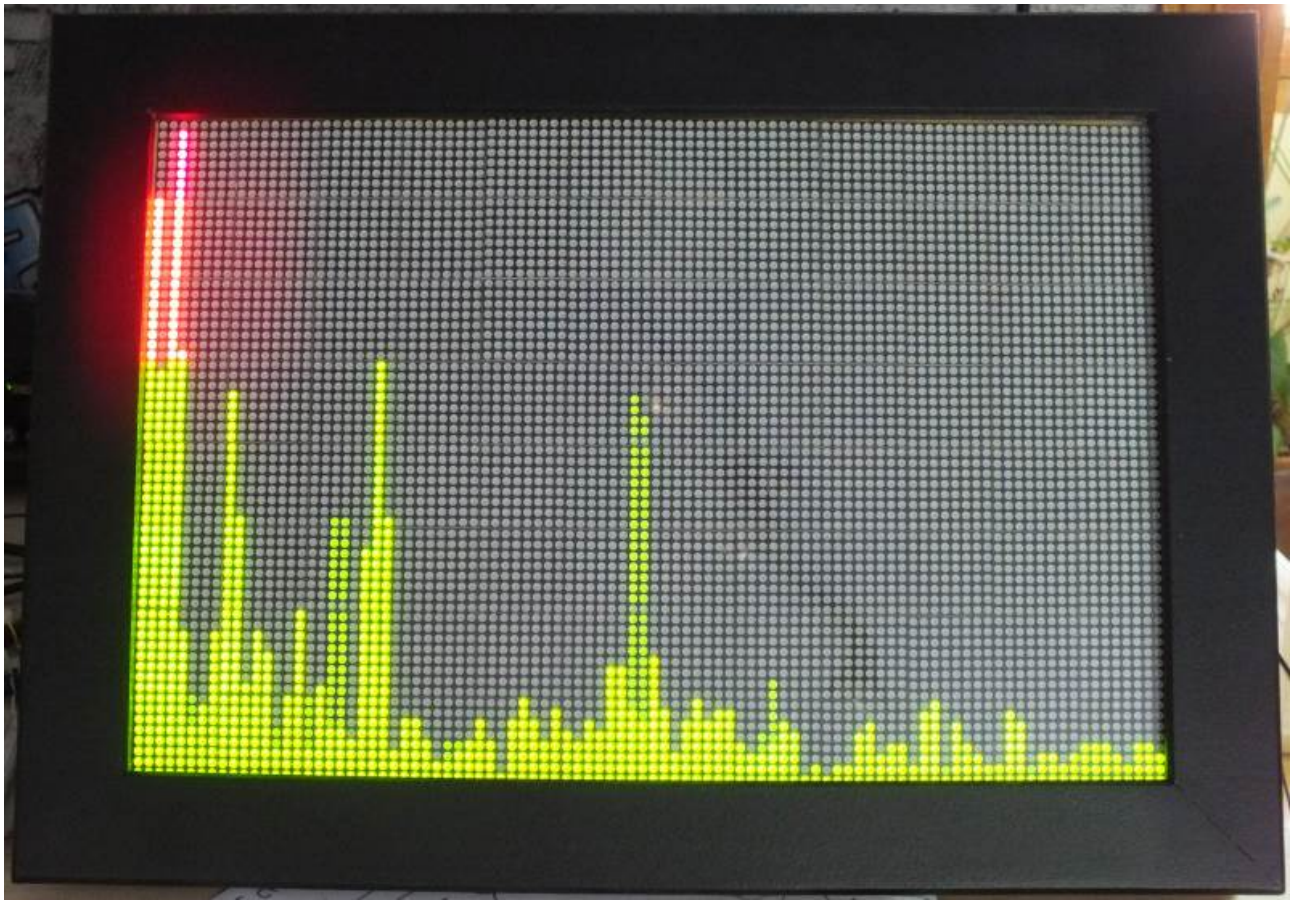


FIGURE 9.16 – Programme « FFT », mode linéaire

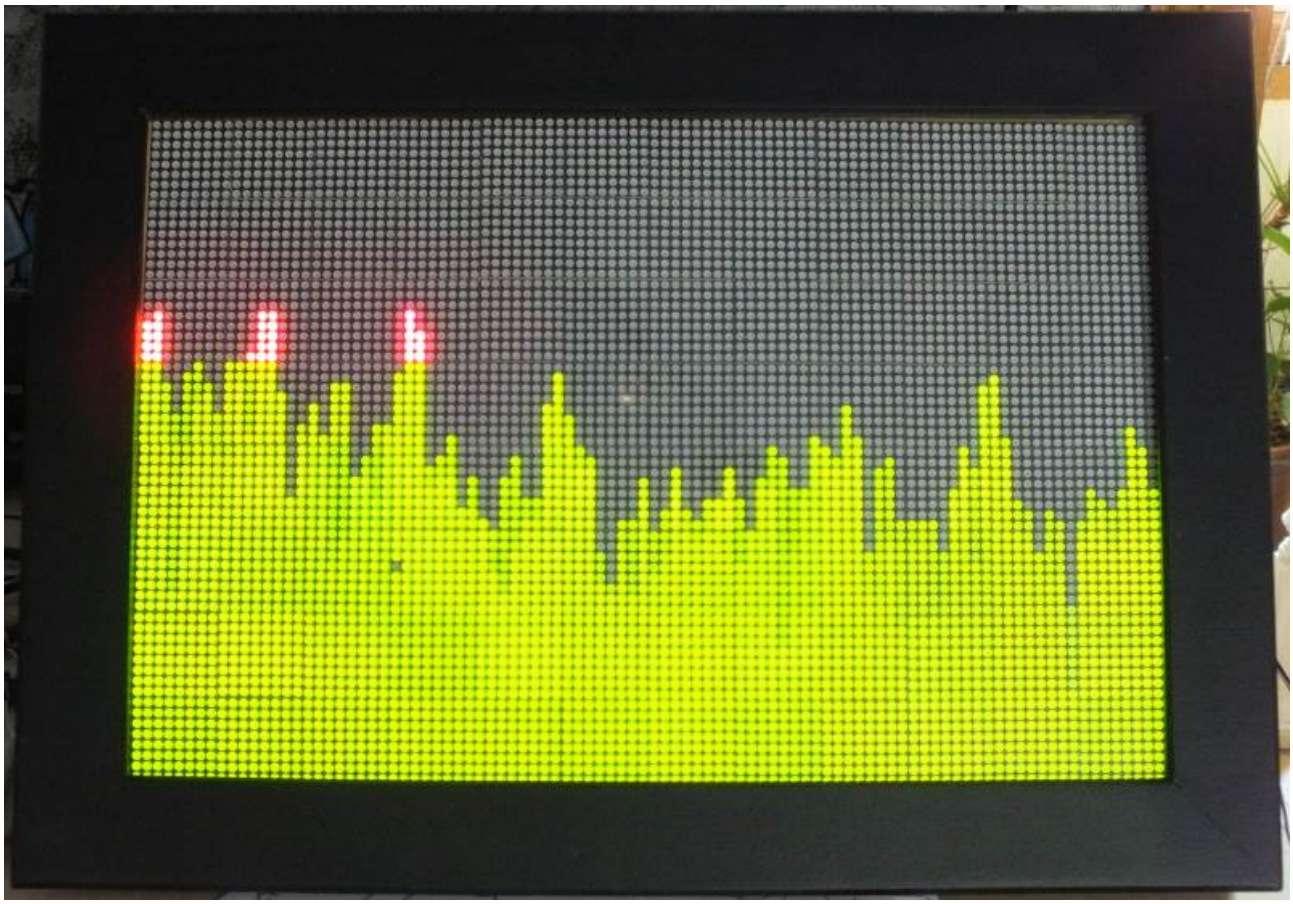


FIGURE 9.17 – Programme « FFT », mode logarithmique (PS : oui il y a bien un pixel mort)

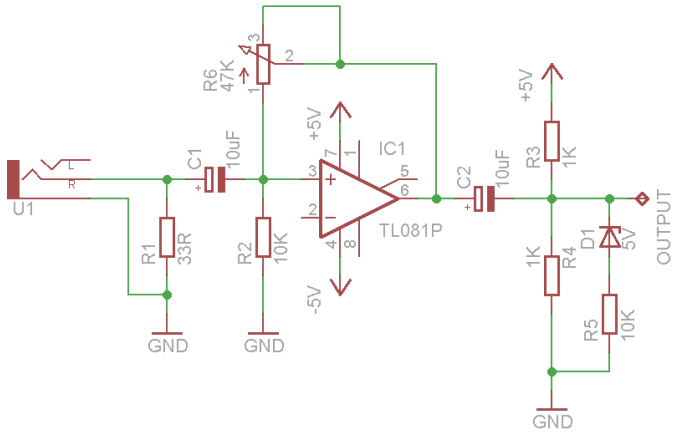


FIGURE 9.18 – Schéma du montage de conversion audio (±1v) vers ADC (0~5v)

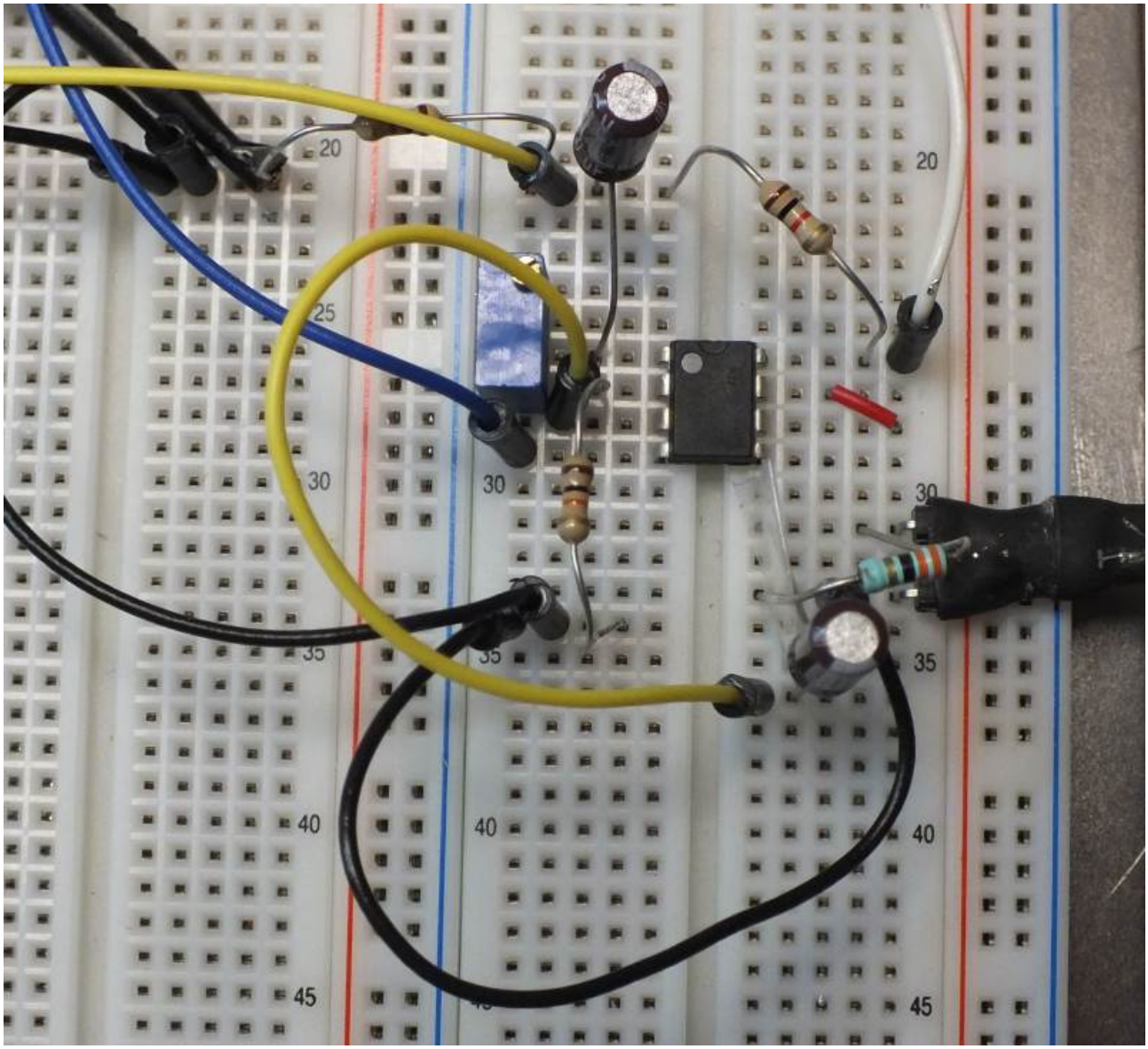


FIGURE 9.19 – Montage de conversion audio vers ADC sur breadboard

10. Du savon fait maison

date 2013-08-01
category cuisine,écologie
level facile
author Nerick
licence By-Sa-3.0

Le savon de marseille... Je suis sûr que vous connaissez tous, mais que personne ne l'utilise. C'est vrai, souvent on en reste aux blocs infâmes trouvés en supermarchés, et qui laissent après utilisation, une désagréable sensation sur la peau.

Personellement, j'ai (re-)découvert le savon en me promenant aux pays-bas, en tombant sur une boutique dont j'ai complètement oublié le nom. Les savons étaient présentés tels des patisseries psychédéliques, tous plus appétissants les uns que les autres. Le savon était redevenu fun.

Et après essai, il est aussi tout a fait possible d'en avoir de beaucoup plus agréables que nos habituels gels douches. Partant de là, l'idée de faire mon propre savon n'était plus très loin, d'autant que mes souvenirs de TP de chimie me rappellaient qu'il s'agissait d'une opération très simple.

Concrètement, pour obtenir du savon, il faut faire agir une base forte sur une matière grasse, ce qui produit du savon et de la glycérine. En jouant sur le ratio corps gras/base forte, on obtient un savon plus où moins « sec », que vous pourrez adapter à votre type de peau. Il est tout à fait possible, voir même recommandé, d'utiliser différentes huiles, les savons obtenus auront des propriétés très différentes. Pour se les procurer, sachez déjà que l'on trouve une grande variété d'huiles différentes en magasin bio, sinon, quelques sites de vente en ligne se sont spécialisés sur la vente d'ingrédients cosmétiques.

Vous pourrez également ajouter des huiles essentielles pour parfumer votre savon, ou le laisser nature (certaines huiles ou beurres sentent naturellement très bon).

Avant de mettre les mains à la patte (façon de parler bien sûr), un petit rappel de sécurité. La soude est un produit très caustique, qui attaquera sans problème votre peau, ou pire, vos yeux. Donc, gants et lunettes de protections obligatoires ! En cas de projection, rincer abondamment à l'eau (s'il s'agit d'une petite zone sur la main, ça devrait suffire, s'il s'agit du visage ou des yeux, faites appeller le centre anti-poison pour connaître la marche à suivre, pendant que vous rincez abondamment à l'eau). La protection du corps, c'est bien, mais il faut penser à un autre point important : les récipients et ustensiles.

En effet, il est possible que certains plastiques soient attaqués par la soude, donc utilisez de préférence un récipient en verre, ça évitera les mauvaises surprises. Pour les ustensiles, évitez l'aluminium, il réagit assez violemment avec la soude.

Maintenant que ce petit rappel est fait, passons aux travaux pratiques.

La précision requise pour les dosages va fortement dépendre de la quantité de savon réalisée. La recette suivante vous permettra de réaliser l'équivalent d'une grosse dizaine de savons, de taille à peu près standard.

Liste des ingrédients :

- 200g d'huile de coco,
- 400g d'huile d'olive,
- 200ml d'eau,
- 85g de soude caustique,
- huile essentielle de lavande.

Cette recette vous donnera un savon assez gras, très doux. À l'origine, j'avais mis de l'huile essentielle de vanille, mais l'huile d'olive ayant une odeur assez marquée, l'odeur de la vanille s'en trouvait masquée. Je recommande donc l'utilisation d'huile essentielle de lavande à la place.

Première étape

Peser les différents ingrédients. Avec un peu de pratique, on se rend compte que c'est beaucoup plus facile d'avoir tous les ingrédients déjà pesés, prêts à être utilisés. N'oubliez pas de tarer vos récipents avant de peser !

.. figure : : savon/3.JPG

alt Huile de coco mis à l'aide d'une cuillière dans un bol posé sur la balance | :target : savon/3.JPG

Le pesage de la soude	. . . et de l'huile de coco

Seconde étape

Il s'agit maintenant de diluer la soude.

Prenez le récipient dans lequel vous avez prévu de mélanger votre pâte à savon, et versez-y les 200ml d'eau. Puis ajoutez très progressivement les cristaux de soude, en mélangeant bien, mais sans faire d'éclaboussures.

Le mélange eau/soude est très exothermique (dégage de la chaleur), je vous conseille de faire ça à l'extérieur, ça évitera les vapeurs, et limitera la casse en cas de fausse manipulation. Si vous y êtes allé trop vite, laissez refroidir un peu (environ 30°), sinon la prise de votre savon sera trop rapide.

Troisième étape

On va maintenant mélanger les différentes huiles à la soude. Commencez par mélanger progressivement l'huile d'olive, en mélangeant bien, puis incorporez l'huile de coco.

Maintenant que vous avez bien mélangé, je vous conseille d'utiliser un mixeur pour bien mélanger votre pâte. Votre savon sera plus homogène, et, surtout, vous allez accélérer très fortement la réaction.

En effet, à partir de maintenant, il va falloir attendre la « trace », c'est-à-dire que le mélange commence à durcir suffisamment pour laisser une trace en surface lorsque l'on en fait couler un peu avec une cuillère. Et pour arriver à la trace, ça peut prendre beaucoup de temps, selon les proportions que vous avez utilisé, la température des ingrédients, ainsi que la méthode utilisée pour le mélange. Si vous brassez à la main, vous êtes partis pour plusieurs heures. Avec un mixeur, et la recette donnée ci-dessus, la trace apparaitra au bout de 30 minutes environ.

Au mixeur, mélangez continuellement pendant 15 minutes, puis laissez reposer 5 minutes, remélangez pendant 5 minutes, etc. jusqu'à la trace.

Quatrième étape

Ça y est, vous avez enfin la trace. Vous allez pouvoir ajouter maintenant les huiles essentielles. Comptez une bonne trentaine de gouttes pour bien parfumer (l'odeur aura tendance à diminuer avec le séchage).

Pourquoi attendre la trace pour ajouter les huiles essentielles ? Et bien parce que certains parfums vont accélérer très fortement la trace, ce qui fait que votre savon risque d'être mal mélangé.

Une fois les huiles essentielles mélangées, vous pouvez mouler vos savons. Personnellement, je fais ça avec des moules à madeleine en silicone, la taille est pas mal (un poil petit quand même), et la forme sympatique. Si vous utilisez des moules fait en une autre matière, graissez les un peu avant de couler le savon, le démoulage en sera facilité.

Dernière étape

Et oui, il y a encore une étape, et c'est la plus importante. Il s'agit de laisser reposer et sécher nos savons. Au bout de quelques jours, vos savons auront déjà bien durcis, et vous pourrez les démouler délicatement (très délicatement, la pâte étant encore molle, si vous appuyez trop fort, vous laisserez des marques).

Mais vos savons ne seront toujours pas utilisables, car encore corrosifs ! En effet, cette méthode de saponification est particulièrement lente, et il vous faudra laisser vos savons reposer entre 6 et 8 semaines pour que la réaction soit bien terminée, et vos savons utilisables.

Laissez les donc reposer à température ambiante, en les entourants éventuellement d'un film alimentaire pour qu'ils ne blanchissent pas, mais dans ce cas, il seront plus mou et s'useront plus rapidement car ils seront moins secs au final.

Pour conclure

Voilà, j'espère que cette petite introduction à la savonnerie vous aura plu, n'hésitez pas à expérimenter vos propres mélanges. Un calculateur pour avoir les bons dosages et pour finir, mon article original.

11. À propos

Le projet

Fait Main est un magazine en ligne conçu par des bénévoles passionnés par la bidouille en général.

Plus d'infos dans l'édito du premier numéro.

Fait Main n'a pas de structure officielle, bien qu'il soit question de monter une association à but non lucratif.

Les articles sont protégés par la licence choisie par l'auteur, si aucune mention n'est apportée, la licence par défaut de tout contenu texte ou photo est CC-By-SA 3.0. Le code du site web est en licence Apache v2.0

Le projet est fondé et maintenu par Tarek Ziadé mais **Fait Main** est un projet open-source et a pour vocation de construire une communauté de contributeurs, que ce soit pour l'écriture d'articles, les relectures ou la conception et la mise à jour du site (code ou design)-ou tout simplement pour proposer un sujet ou un auteur.

Si vous avez envie de participer il suffit de vous inscrire sur la mailing list et de vous présenter - ou bien de forker un de nos repositories sur Github.

N'oubliez pas de lire le guide à l'usage des auteurs

Vous pouvez aussi contacter Tarek : tarek@faitmain.org

Le site web

Le site web est un site statique généré avec le moteur kompost, un script Python créé pour l'occasion, et des templates Mako_.

Le contenu est au format reStructuredText. Ce format permet de générer les pages html mais aussi le PDF (spartiate), et dans le futur un ePub, voir un magazine papier au format un peu plus léché.

Le site utilise Bootstrap pour un rendu correct sur tous les périphériques. Les icônes sont de chez Glyphicons.

Le design à été réalisé par Raphaël Bastide avec joie et amour.

Chaque lien sortant est transformé en short link avec https://github.com/faitmain/short.faitmain.org. Cette redirection permet de corriger d'éventuels liens cassés de manière centralisée.

Le moteur de recherche est un web service écrit avec Cornice, basé sur Xapian, appelé en Javascript depuis l'écran de recherche. La base Xapian est mise à jour à chaque modification de contenu.

Le PDF

Le PDF est généré grâce à rst2pdf, qui lui même utilise ReportLab.

La mise en page est spartiate mais suffisante pour une lecture sur écran.

Informations légales

Magazine publié en France. Numéro ISSN : **2261-8376**.

Contact & adresse :

Tarek Ziadé - tarek@faitmain.org 6 rue de l'Eglise 21540 Turcey France

— **Éditeur** - Tarek Ziadé
— **Directeur de la publication** - Tarek Ziadé

Donnée personnelles stockées : chaque accès au site est stocké dans un fichier de log à des fins de statistiques - mais **les adresses IP ne sont pas collectées** et le serveur ne contact aucun service tiers.

12. Petit guide à l'usage des auteurs

Format général

Le format des articles est le reStructuredText. Ce format est très simple et vous pouvez lire ce document d'introduction.

Pour **Fait Main**, chaque article doit commencer par un titre souligné par des signe = (égal), puis chaque sections par un titre souligné par un signe *:* (deux points).

L'article commence par une série de méta-données :

- **date** — la date de l'article
- **category** — les catégories, en minuscule, séparées par des virgules, sans accents, à prendre dans : ecologie, informatique, electronique, art, hacktivisme, cuisine.
- **level** — le niveau de lecture : avancé, vulgarisation, moyen, etc. Champ libre.
- **author** — l'auteur de l'article au format **Prénom Nom** — Kompost va faire un lien entre l'article et la page */auteurs/prenom_nom.html* donc respectez bien le format.
- **licence** - la licence de l'article

Exemple :

```
What The Feuille ?
==================

:date: 2012-12-12
:category: ecologie,informatique
:level: vulgarisation
:author: Tarek Ziadé
:licence: By-NC-SA 3.0


Du texte...

Section 1
:::::::::

Du Texte...

----

Du texte

Section 2
:::::::::

Du Texte
```

Notez que les saut de section (—-) sont remplacés par des balises **HR** qui apparaissent comme des petites scies dans la CSS faitmain.

Choix d'une licence

Il est important de bien choisir sa licence pour son article. Le plus simple est de suivre le guide à cet adresse : https://creativecommons.org/choose/?lang=fr

Le choix le plus important est de savoir si vous voulez autoriser ou non l'utilisation commerciale. Si vous autorisez l'utilisation commerciale, vous permettez à votre article une diffusion sans limites, sans forcément être rénuméré. C'est un bon choix si vous souhaitez diffuser au maximum votre article mais un mauvais choix si vous souhaitez négocier une rémunération si l'article paraît dans une revue commerciale.

Le choix d'une licence dans Fait Main n'est bien sûr pas gravée dans le marbre - l'article vous appartient et vous pouvez le re-licencier pour d'autre publications.

A partir du numéro 3, Fait Main ne choisit plus de licence par défaut et demande à chaque auteur d'en choisir une en toute connaissance de cause.

Images

Les photos doivent être au format JPEG, en mode progressif en qualité 85%, et d'une largeur de 800 pixels et 72 dpi.

Chaque photo peut avoir une version *retina* avec 1600 pixels. Le nom du fichier dans ce cas doit être **nom_fichier@2x.jpg**, et le site sélectionnera automatiquement la version haute résolution sur les écrans rétina.

Une image est placé à coté de l'article et mise dans une balise **figure** :

```
.. figure:: ../volume-3/nom_fichier.jpg

  :target: ../volume-3/http://un.lien.clickable(optionel)

  La légende de l'image (obligatoire)
```

Pour afficher une image plus petite mais conserver son redimensionnement automatique, il faut utiliser l'option **scale** qui est un pourcentage.

Exemple pour une image qui fera 50% de l'écran et sera centrée

```
.. figure:: ../volume-3/nom_fichier.jpg

  :scale: 50

  La légende de l'image (obligatoire)
```

Pour les petites images, le png est toléré.

13. Mailing List

Fait Main a une Mailing List sur Librelist.

Pour vous abonner, il suffit d'envoyer un mail à faitmain@librelist.com.

Vous recevrez un message de confirmation auquel il suffira de répondre.

Les archives peuvent être consultées ici : http://librelist.com/browser/faitmain

Pour vous désabonner, il suffit d'envoyer un mail à faitmain-unsubscribe@librelist.com avec l'addresse e-mail utilisée pour l'inscription.

14. Partenariats avec Fait Main

Vous êtes constructeur de matériel susceptible d'être présentés dans le magazine Fait Main ?

Vous pouvez devenir partenaire permanent ou ponctuel et fournir du matériel aux auteurs pour le test de matériel et l'écriture d'articles, en échange d'une bannière et de liens sur l'article.

Vous êtes auteur ? Grâce à Fait Main vous pourrez peut être récupérer du matériel à tester. Ne contactez pas directement les partenaires, pour que FaitMain puisse centraliser et synchroniser les propositions.

Contactez tarek@faitmain.org.

Par souci de transparence, tous les partenariats permanents sont détaillés ici, avec les conditions entendues avec le partenaire.

Hackspark.fr

Hackspark est une boutique dédiée au hacking (bidouille) hardware sous toutes ses formes et aspects.

Le partenariat est le suivant : Hackspark peut fournir du matériel électronique pour l'élaboration d'un projet pour un article, sous réserve d'acceptation du sujet et d'un plan détaillé.

Si le prix global dépasse 40 euros, hackspark se réserve le droit de demander une participation financière à l'auteur.

L'article conçu avec le matériel aura des liens vers la boutique Hackspark et une en-tête pour préciser le partenariat avec un lien vers cette page.

Yoctopuce

Yoctopuce est une entreprise Suisse qui conçoit, fabrique et vend des modules USB destinés à tous ceux qui souhaitent interfacer leur ordinateur favori avec le monde réel.

Yoctopuce peut fournir du matériel électronique pour l'élaboration d'un projet pour un article, sous réserve d'acceptation du projet.

L'article conçu avec le matériel Yoctopuce devra comporter un lien sur les produits Yoctopuce utilisés.