Rapport de projet monotron

Ederle Thomas Chalono Kévin

16 avril 2014

Table des matières

Première partie

Etude du circuit électronique du monotron

0.1 Introduction

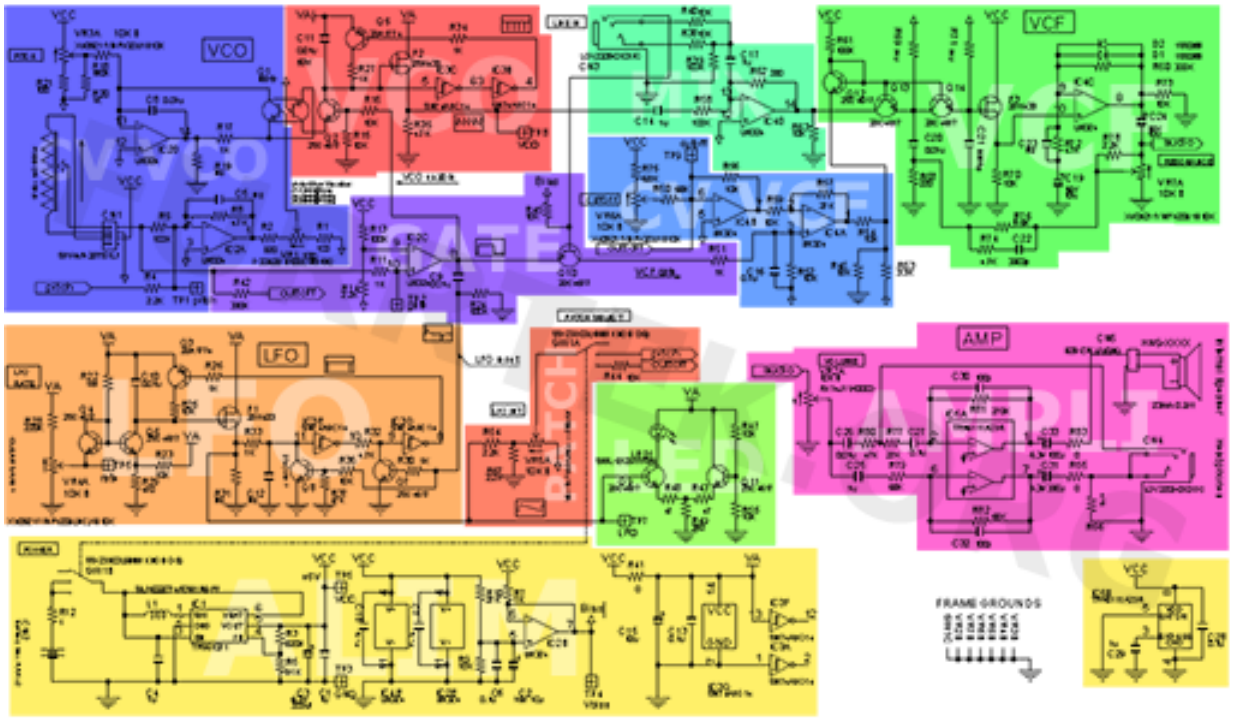

FIGURE 1 – schéma bloc du monotron

Le Monotron est un synthétiseur analogique de poche créé par Korg, utilisant une bande résistive comme clavier. Il comprend un oscillateur saw, un LFO (Low Frequency Oscillator) et un VCF (Voltage Controled Filter), qui peut être utilisé avec une source audio externe. Le LFO peut être "patché" soit au pitch, soit au cutoff du filtre. Le tout fonctionne sur deux piles AA, deux prises jack sont là pour l'entrée et la sortie audio, et une molette de volume règle la sortie haut parleur et casque. En figure Figure 1 le schéma décliné sous forme de bloc du monotron.

0.2 L'alimentation

IC1 est un TPS61071, convertisseur DC-DC step-up Texas Instruments, qui peut sortir 5V à 150mA (600mA max) à partir de 1.8V. Il est utilisé pour convertir le 3V décroissant des piles au 5V stable nécessaire. R3 et R6 servent à fixer la tension de sortie du convertis-

seur. Elles sont dimensionnées pour donner 0.5V (Vref interne de IC1) sur FB quand VCC est à 5V. Vbias est utilisé dans beaucoup d'endroits (symbole en flèche) et provient du diviseur R9, R10, bufferisé par IC2B.

$$Vbias = \frac{VCC.R9}{R10 + R9} = 1.35V \tag{1}$$

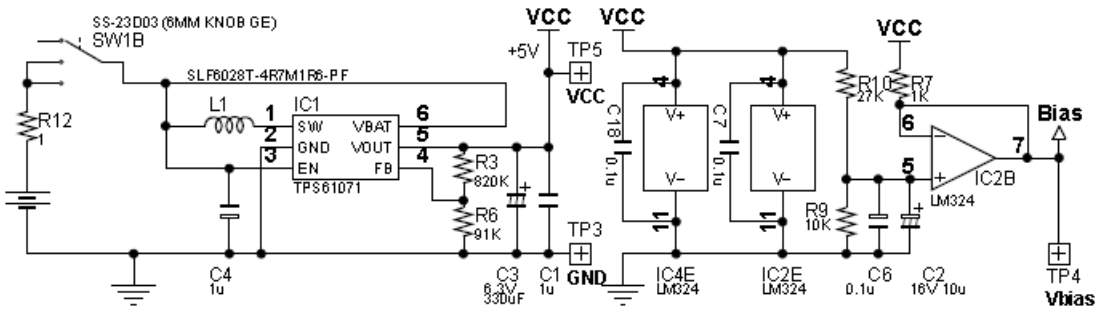

FIGURE 2 – schéma d'alimentation du monotron

En figure Figure 2 le schéma d'alimentation du monotron.

0.3 LFO (low frequency oscillator)

Il s'agit d'un Oscillateur dent de scie décroissant à fréquence variable. Q4 et Q6 constituent un mirroir de courant. Le courant est fixé par R22 (fixe) et la tension variable sur l'émetteur de Q4.

$$T_e = \frac{VA.VR4}{R28 + VR4} \tag{2}$$

$$I = \frac{VA - Ve - 0.7}{R22} \tag{3}$$

$$T_{max} = 0.22V \rightarrow I_{correspondant} = 4.1\mu A \tag{4}$$

$$T_{min} = 0V \rightarrow I_{correspondant} 430\mu A \tag{5}$$

$$\tag{6}$$

C10 sert à temporiser la rampe, son courant de décharge est limité par le mirroir de courant. Il se décharge jusqu'à

$$Ve_{Q6} + Vbe = \frac{VA.R29}{R23 + R29} + Vbe = 0.92V \tag{7}$$

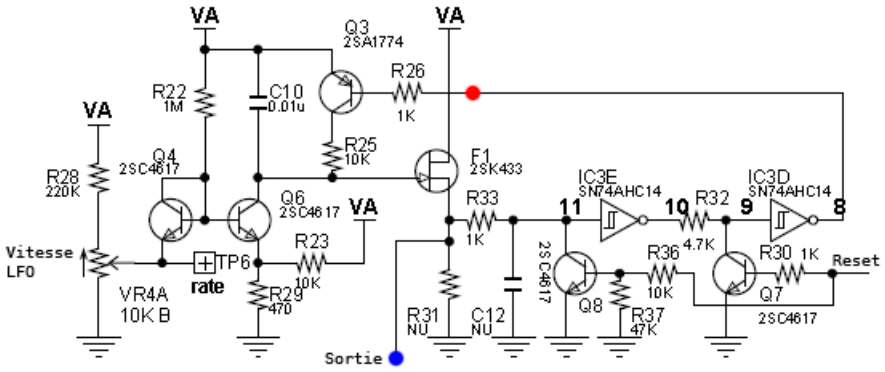

FIGURE 3 – schéma du LFO

F1 sert de suiveur pour la tension de C10, c'est la sortie du VCO. Cette tension passe dans deux inverseurs à trigger IC3. Lorsque la tension est inferieure au seuil bas, le transistor Q3 est commandé par une tension proche de 0V et recharge C10 via R25. La tension remonte donc brusquement, inverse l'état des inverseurs et arrête la charge. La durée de charge dépend de R25, de la capacité de C10 et de l'hysteresis des inverseurs (temps de charge jusqu'à dépasser leur seuil haut). Lorsque le signal Reset dépasse Vbe (0.7V), Q8 annule la tension de sortie, et Q7 bloque Q3 pour laisser le condensateur se décharger. En figure Figure 3 le schéma de l'alimentation.

0.4 VCO et CVCO

0.4.1 control voltage du VCO (control voltage oscillator)

Quelques spécification techniques :

$$Pitch_{min} = 0.05 \tag{8}$$
$$Pitch_{max} = 5V \tag{9}$$
$$Ribbon_{min} = 0V \tag{10}$$
$$Ribbon_{max} = 5V \tag{11}$$

IC2C est un comparateur pour detecter l'appui sur la bande, ou dans notre cas la commande venant du processeur.

$$V_{seuil} = \frac{VCC - Vbias}{R13 + R14}.R14 + Vbias = 1.43V. \tag{12}$$

IC2A est un amplificateur inverseur.

$$Gain_{bande} = \frac{R8}{R5} = 0.047 \quad (13)$$

$$V_{min} = 63mV \quad (14)$$

$$V_{mac} = 235mV \quad (15)$$

$$Gain_{pitch} = \frac{R8}{R4} = 2.1 \quad (16)$$

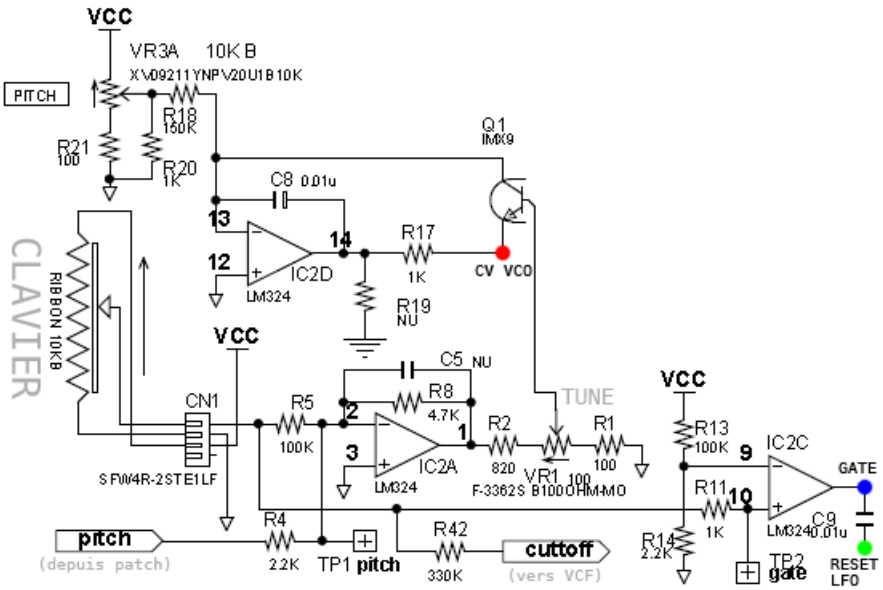

FIGURE 4 – schéma du CVCO

VR1 sert de tunner, IC2D est un VCA de Pitch, commandé par Q1. En figure Figure 4 le schéma de l'étage CVCO.

0.4.2 VCO

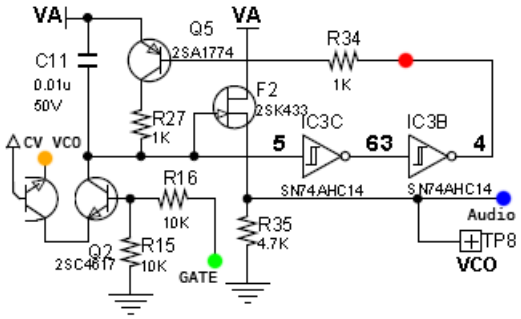

FIGURE 5 – schéma du VCO

Oscillateur : pareil que celui du LFO. Démarrage par Q2 depuis le signal GATE. Décharge de C11 contrôlé en courant par Q1, recharge par Q5 et R27. F2 comme buffer, et retour dans deux inverseurs. En figure Figure 5 le schéma du VCO.

0.5 Mixeur Line in

Mélange des canaux gauche/droite par R38, R39 et R40.
Signal audio centré sur Vbias et réduit par IC4D.
Gain : R62/R58 : 0.0033.

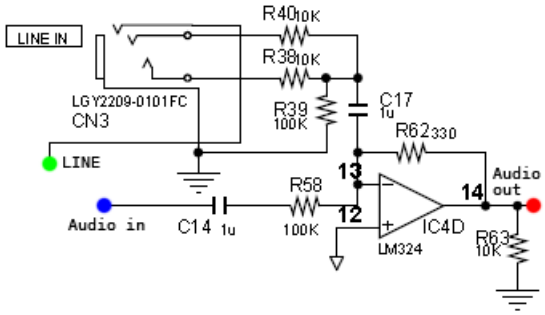

FIGURE 6 – schéma du mixeur line in

En figure Figure 6 le schéma du mixeur line in

0.6 VCF et CVCF

0.6.1 Control voltage du VCF

Le signal Cutoff vient du keyboard track (CV du VCO) et du patch LFO.

$$Cutoff_{min} = 0V \quad (17)$$
$$Cutoff_{max} = 2.97V \quad (18)$$

IC4B est centré sur Vbias et il y a réduction du Gain pour le potard

$$\frac{R56}{R50} = 0.147 \quad (19)$$

cutoff depuis le VCO :

$$\frac{R56}{R44} = 1 \quad (20)$$

Le signal Ligne active constament le filtre quand une source line in est utilisée. Il empêche Gate de tirer VCF Gate à 0V. GATE bas : v+ tiré à la masse par Q10 si on utilise pas line-in, tension de commande nulle. C16 déchargé via R51. GATE haut : C16 charge jusque Vbias via R52 (Attack), IC4A amplifie.

$$Gain = \frac{R57}{R59} = 2.7 \quad (21)$$

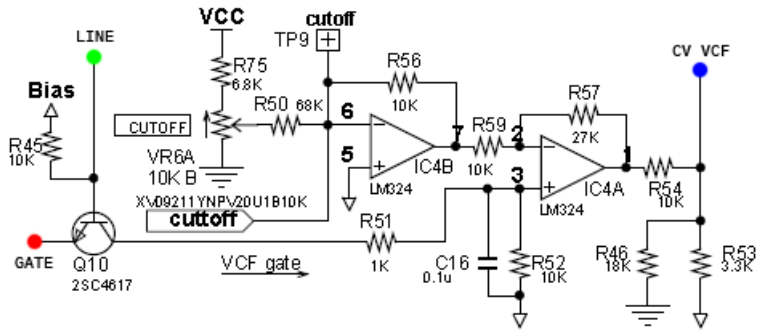

FIGURE 7 – shéma du control voltage du VCF

En figure Figure 7 le schéma du control voltage du VCF

0.6.2 VCF

Trois points : sortie de l'ampli (R54), Vbias (R53) et masse (R46).

$$V_{commande} = \frac{\frac{Vout}{R54} + \frac{Vbias}{R53}}{\frac{1}{R54} + \frac{1}{R53} + \frac{1}{R46}} \tag{22}$$

IC4C : Ampli non-inverseur.

$$Gain = 1 + \frac{R60}{R72} = 71.2. \tag{23}$$

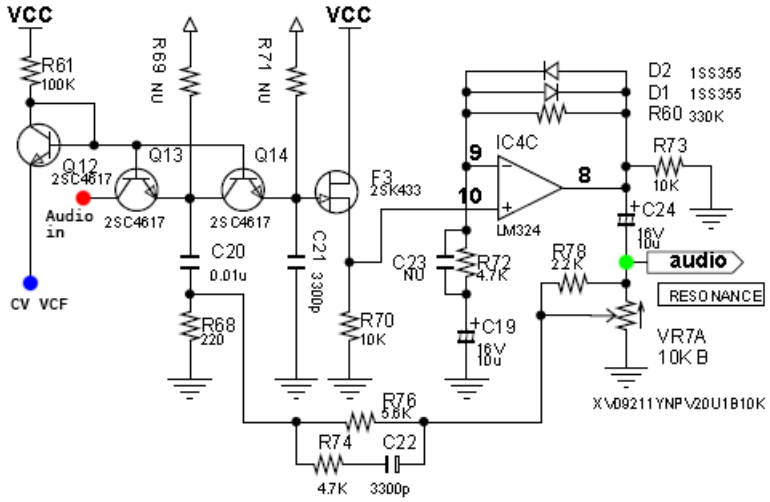

FIGURE 8 – shéma du VCF

Q13 et Q14 : Double filtre RC, avec les transistors comme résistances. Q13 : résonnance. Q14 : RC normal. En figure Figure 8 le schéma du VCF

0.7 Amplificateur de puissance

Gain pour le haut parleur :

$$G = \frac{R81}{R77} = 12.3 \tag{24}$$

$$Gain_{sortiecasque} = \frac{R82}{R79} = 1 \tag{25}$$

Filtre passe-bas sortie HP :

$$\frac{1}{2.\pi.R81.C30} = 5.89kHz \tag{26}$$

Filtre passe-bas sortie casque :

$$\frac{1}{2.\pi.R82.C32} = 23.4kHz \tag{27}$$

Filtre passe-haut avec haut parleur 8 ohms :

$$\frac{1}{2.\pi.Rhp.C33} = 199Hz \tag{28}$$

Filtre passe-haut avec casque 8 ohms :

$$\frac{1}{2.\pi.R66.Rcasque.C31} = 67Hz \tag{29}$$

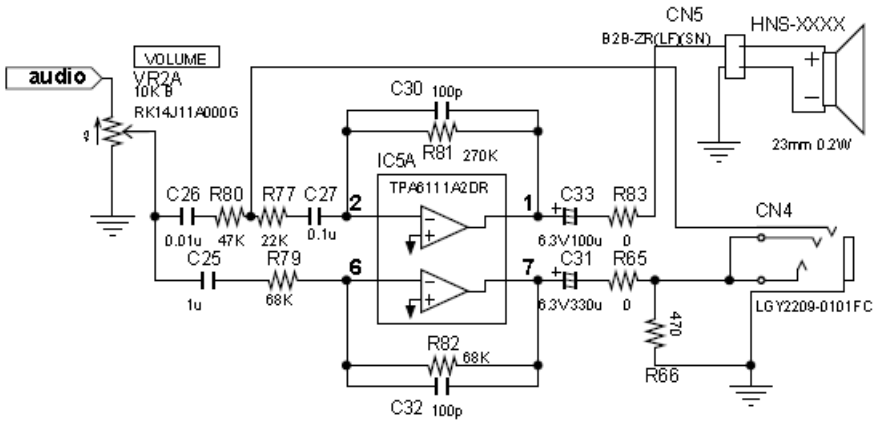

FIGURE 9 – shéma de l'amplificateur

En figure Figure 9 le schéma de l'amplificateur de puissance du montage.

Deuxième partie

Modifications apportées au circuit

Certaines fonctions sur la carte ont été modofiées et d'autres ont été rajoutées pour pouvoir répondre au mieux au cahier des charges.

0.8 ajout de la carte MBED

0.8.1 Le microprocesseur ARM CORTEX

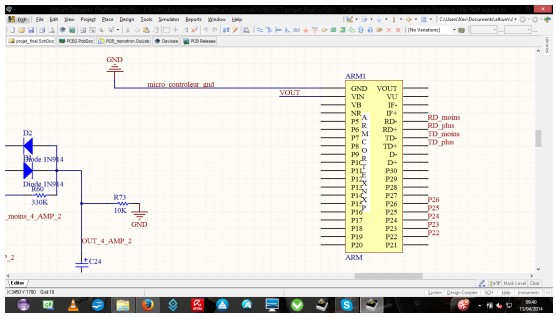

FIGURE 10 – ajout du processeur arm sur le schematic

On ajoute au schematic notre micro processeur ARM CORTEX pour qu'il puisse commander la board voir figure Figure 10.

Puisque les sortie PWM du processeur arm est un signal crénaux, il est nécéssaire de filter la tension de sortie. Le spectre du signal de sortie peut s'exprimer de la manière suivante

$$v_s = \sum_{k=1}^{\infty} \frac{2A}{\pi k} sin\left(\pi k \alpha\right) \tag{30}$$

Ou α représente le rapport cyclique et A l'amplitude du signal créneau. Pour filtrer ce signal nous avons utilisé des filtres RC classiques dont la fréquence de coupure vaut

$$f_{coupure} = \frac{1}{2\pi RC} \tag{31}$$

AInsi on choisi de filtrer toutes les fréquences supérieures à 4 KHz, on choisi donc R = 220 KΩ et C = 180 pF.

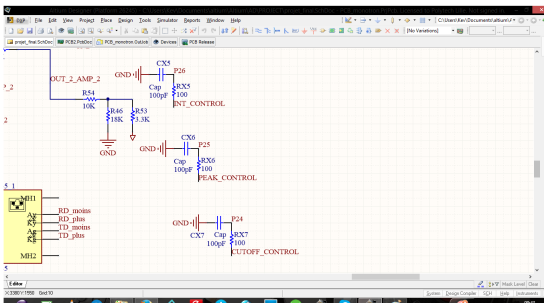

FIGURE 11 – ajout des filtres sur les sorties PWM du monotron

On montre ci dessous l'ajout des filtres à notre schématic Figure 11.

0.9 alimentation

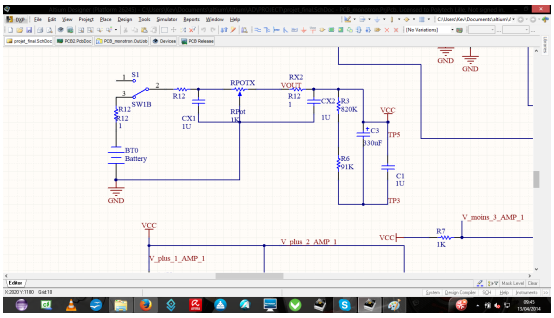

FIGURE 12 – alimentation du monotron

Le schematic original utilisait une PWM suivi de filtre pour pouvoir atteindre la tension d'alimentation de 5 V. Nous avons choisi d'utiliser le régulateur de tension qui va abaisser la tension d'alimentation prévu (4 piles de 1,5 V = 6 V) en une tension d'alimentation de 5 V. J'ai utiliser les empreintes d'une résistance variable pour insérer le régulateur, on peut le voir la figure suivante Figure 12.

0.10 Structure de contrôle mode manuel / mode automatique

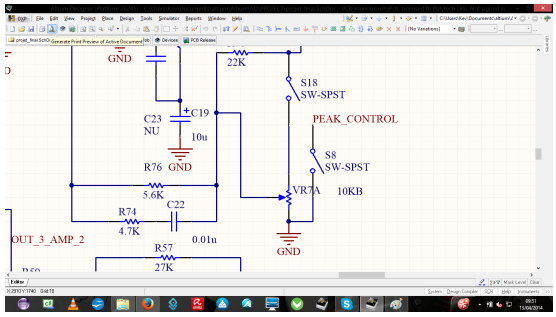

FIGURE 13 – Structure de contrôle mode manuel / mode automatique

Nous avons fait le choix de pouvoir contrôler le monotron en mode manuel ou en mode automatique. Cela nous sera utile notament pour la phase de test comme cela on peut tester si la carte fonctionne indépendament de la partie réseau du projet. Pour réaliser la double commande, il faut que l'on puisse choisir la tension de commande des différents paramètres qui influent sur le rendu sonore on utilise un jeu d'interrupteur pour le faire comme on peut le voir sur la figure suivante Figure 13.

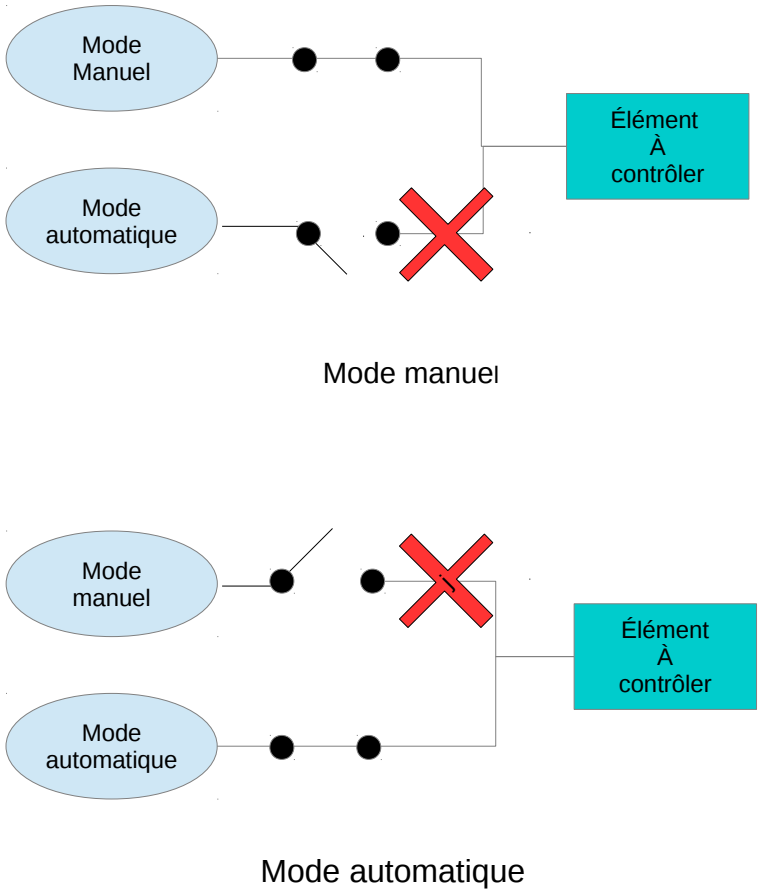

FIGURE 14 – Schéma de principe de la double commande

Nous avons utiliser cette strucure sur tous les éléments que nous souhaitons commander en mode automatique ou en mode manuel. Notons que les deux jeux d'interrupteur fonctionnent de manière complémentaires. En effet les deux ne peuvent pas être fermé en même temps, cela reviendrait à vouloir commander le monotron en mode manuel et automatique en même temps. Si les deux sont ouvert, la carte ne marche pas cartela partie commandée n'est pas alimentée. Une illustration de la stratégie utilisée est donnée en figure Figure 14 .

0.11 Communication ethernet

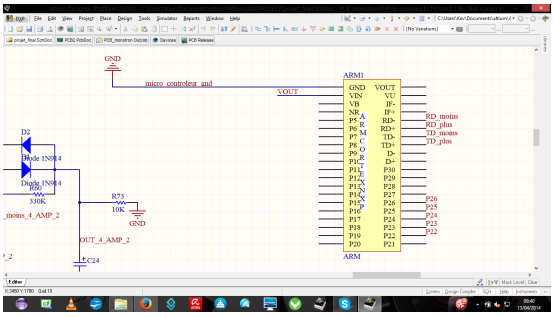

FIGURE 15 – coté micro processeur

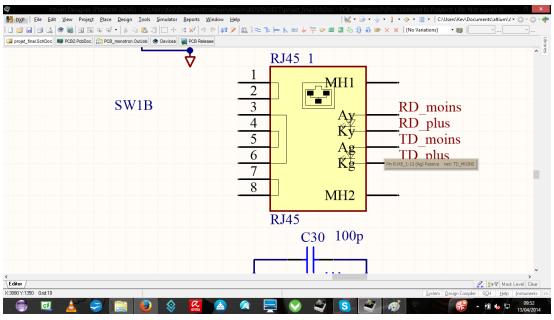

FIGURE 16 – coté connecteur

Il a fallu insérer un connecteur femmelle ethernet sur la carte pour que le micro processeur puisse communiquer en ethernet. On relie les broches Tx aux broches Rx entre elles. voir schéma Figure 15 et Figure 16.

Troisième partie

Partie Réseau

0.12 Présentation des outils pour la réalisation de la partie logiciel

0.12.1 Introduction

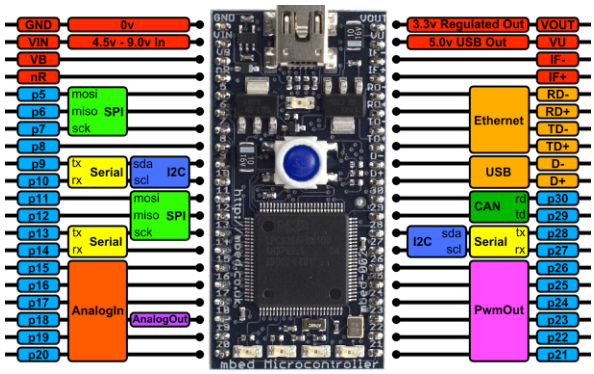

FIGURE 17 – Carte de développement MBED NXP LPC1768

L'objectif de la partie logiciel du projet e-monotron à était de pouvoir controler le monotron à distance par le biais d'une interface utilisateur créer sur une page web dynamique. Les outils essentiel permettant cette réalisation ont était l'utilisation du serveur SMEWS développé par l'équipe 2XS de LIFL, l'utilisation d'une carte de développement MBED NXP LPC1768 qui posséde un processeur 32 bit ARM Cortex-M3 et l'utilisation de la librairie RFLPC en langage C permettant de programmer la carte MBED

0.12.2 Présenttion du serveur SMEWS

SMEWS est un serveur web développé par l'équipe 2XS, la particularité de ce serveur est qu'il a une empreinte mémoire particulièrement faible et adapté pour pour des processeurs de faible puissance (ARM Cortex-M3 par exemple) ou des cartes à puce. Sa particularité réside dans le fait que la pile réseau n'est pas décomposé en couches mais faite d'un bloc est optimisé pour les connections http.

Source http ://wwwlifl/2XS/smews/

0.13 Présentation de la partie logiciel du e-monotron

0.13.1 Une page web statique

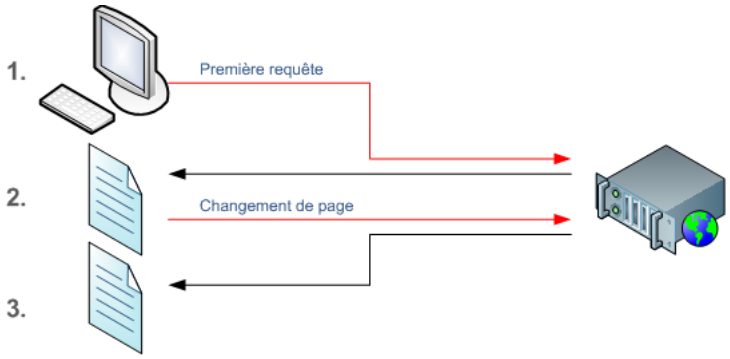

FIGURE 18 – exemple interaction entre un serveur et une page web statique

Afin de pouvoir contrôler le Monotron à distance comme dit précédement. La première étape du projet e-monotron à était de développer une page web statique par le biais du langage HTML et du langage CSS pour la mise en page et la mise en forme et d'implanté cette page sur la carte MBED qui comporte le microcontrôleur ARM ou est implanté le serveur SMEWS.

0.13.2 Développeement d'une interface utilisateur

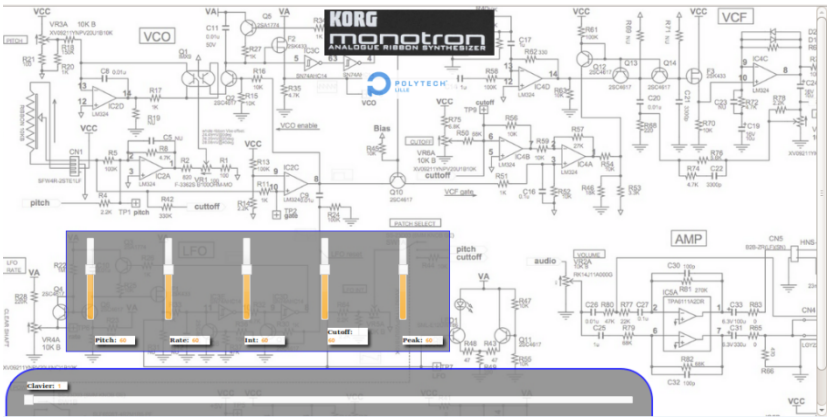

FIGURE 19 – page web et interface de contrôle monotron

FIGURE 20 – interface de contrôle monotron

FIGURE 21 – interface de contrôle page web

Après avoir créé notre page web, la seconde étape du processus de développement était de réfléchir et développé à une interface utilisateur simple qui rappelle les principales fonctionnalités du Monotron, c'est-à-dire le pitch, rate, int, cutoff, peak et le clavier. Pour réaliser cette étape nous avons décidé d'utiliser jQuery qui est une bibliothèque JavaScript gratuite et très pratique, ayant une syntaxe courte et logique, compatible avec tous les navigateurs courants. De plus nous avions déjà utilisé cette bibliothèque auparavant c'est pour cela que nous avons décidé d'utiliser cette bibliothèque. Page web et Interface de contrôle Nous pouvons comparer ci-dessous la ressemblance entre l'interface de contrôle sur la page web et l'interface de contrôle du Monotron.

Remarque : Après avoir implanté la page web dans le microcontrôleur comportant le serveur SMEWS. Nous avons décidé de ne pas insérer le fond de la page car l'espace mémoire ne le permet pas.

0.14 Utilisation de la bibliothèque RFLPC et PWM

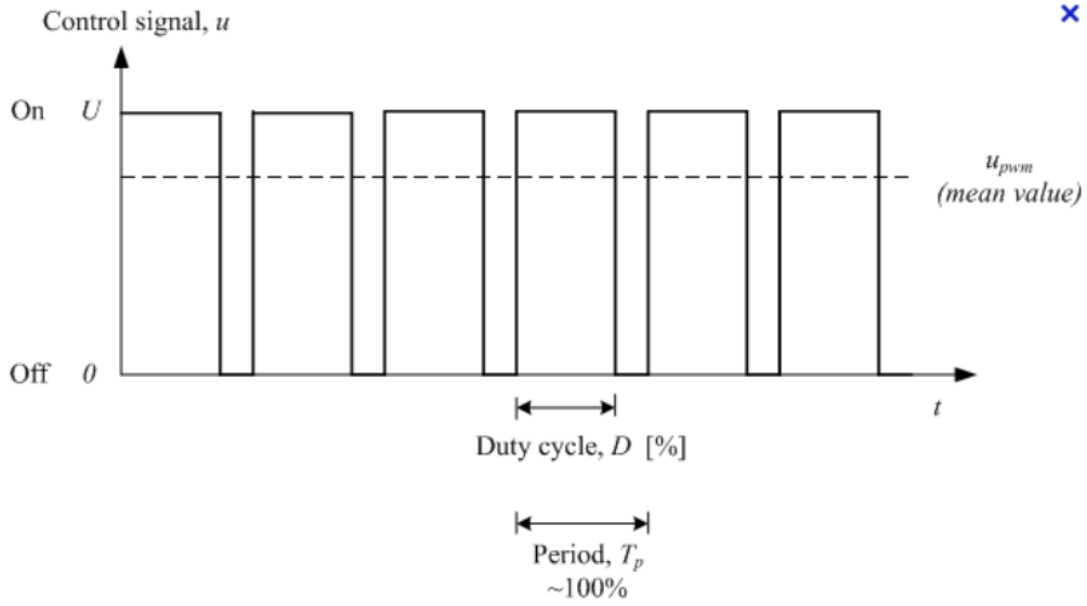

FIGURE 22 – signal d'une PWM

Afin de contrôler le circuit du Monotron par le biais de la carte MBED, nous avons besoin de générer des tensions variables. Pour générer une tension variable par le biais d'un microcontrôleur nous avons deux possibilités, soit utiliser une sortie analogique ou soit générer une PWM (Pulse Width Modulation) sur une broche qui permettra de faire varier la tension la tension moyenne. Sachant que la carte MBED comporte une seul sortie analogique et la possibilité d'utiliser une PWM sur sept broches de la carte. Nous avons décidé de développer six PWM pour les broches 21 à 26 de la carte MBED.

Comme énoncé précédemment nous avons utilisé la librairie RFLPC en langage C pour développer les six pwm et la gestion des entrée/sortie de la carte MBED. Les principales fonctions utilisées sont :

rflpc_gpio_set_pin_mode_output (rflpc_pin_t pin, uint8_t val) ;
// initialise la broche de type rflpc_pin_t en sortie à la valeur val
Int rflpc_pwm_int(rflpc_pin_t pin)
// Initialise le périphérique PWM pour une utilisation sur la broche

séléctionné.
rflpc_pwm_single_edge(rflpc_pin_t pin, uint32_t pulsewidth) ; // Utilisation de la pwm en mode signle_edge, ceci permet de determiner la //période en état haut.
rflpc_pwm_set_period(uint32_t period) ;
// détermine la période de pwm rflpc_pwm_enable (rflpc_pin_t pin) ;
//autorise l'utilisation de la pwm sur la broche selectionné Void rflpc_pwm_start(void) ;
// Commence le cycle de pwm typedef uint8_t rflpc_pin_t
// représente une broche
Remarque : On pourra visualiser tout le development effectué à partir de ces fonctions RFLPC en annexe.

0.15 Une page web dynamique coté client

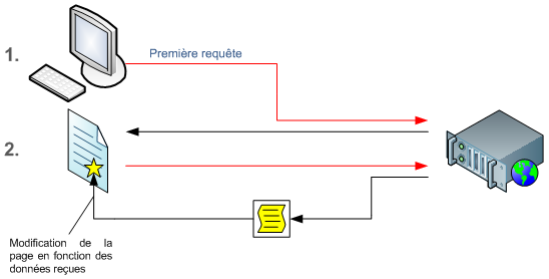

FIGURE 23 – page web dynamique

Après avoir développé les PWM sur la partie microcontrôleur, nous avons à travers cette étape modifiée notre page web statique en dynamique, pour obtenir une application web. La différence va résider dans le fait que quand l'utilisateur cliquera sur un slider la page ne se rechargera pas et le navigateur enverra une requête au serveur contenant la valeur du slider.

Quand l'utilisateur utilisera un slider pour commander le monotron, cela enverra une une requête HTTP vers le serveur. Pour ce faire, il faut disposer d'un objet XHR disposant de cette fonctionnalité. Pour instancie déclarer un objet XHR, on procède de la même façon que pour n'importe quel objet JavaScript à savoir avec le mot-clé new :
var xhr = new XMLHttpRequest() ;
Pour envoyer une requête http il faut définir dans un premier temps les modalités d'envoi avec la méthode open, et on l'enverra ensuite avec la méthode send .

xhr.open("GET", "monotron?var1="+pwm1, true);
xhr.send(null);
GET : méthode de transfert
Monotron.c : programme comportant les pwm développé avec la librairie rflpc True : définit si le mode de transfert est asynchrone ou non

0.16 Une page web dynamique coté serveur

Les applications Web sont prévues par Smews via un ensemble de fonctions handler prédéfinies, par cette ensemble de fonction nous allons utiliser : doGet qui est une fonction utilisé pour générer la réponse HTTP. Cela permet d'associer une fonction C à cette handler. Exemple ci-dessous, la fonction switch_led écrite en c est associé à chaque requete http.

```
/*
<generator>
<handlers doGet="switch_led"/> </generator>
*/

static char switch_led(struct args_t *args) {}
```

Smews permet d'analyser les arguments URL avant d'appeler le handler doGet, cette analyse est traitée dans le noyau. Chaque argument URL est directement accessible . Voir exemple ci-dessous. <generator>

```
<handlers doGet="switch_led"/>
<args>
<arg name="var1" type="uint8"/>
<arg name="var2" type="uint8"/>
<arg name="var3" type="uint8"/>
<arg name="var4" type="uint8"/>
<arg name="var5" type="uint8"/>
<arg name="var6" type="uint8"/>
</args>
</generator>
*/
static char switch_led(struct args_t *args) { pwm1=args->var1;
//accede à la variable
pwm2=args->var2;
pwm3=args->var3;
pwm4=args->var4;
pwm5=args->var5;
pwm6=args->var6;
}
```

0.17 Conclusion

Nous avons réaliser une carte double face et dimensionner les éléments ajoutés (filtres de la PWM). Ce projet nous a permis d'améliorer nos connaissances en développement web, en réseau, et en électronique. Malgré quelques soucis de développements nous avons apprit à travers ce projet à synthétiser, chercher et examiner les informations essentiel pour mener à bien la réalisation d'un projet. Les connaissances et la méthodologie apprise durant ce projet ne pourront être que bénéfique pour notre futur métier d'ingénieur.

Quatrième partie

Annexe

.1 foot print

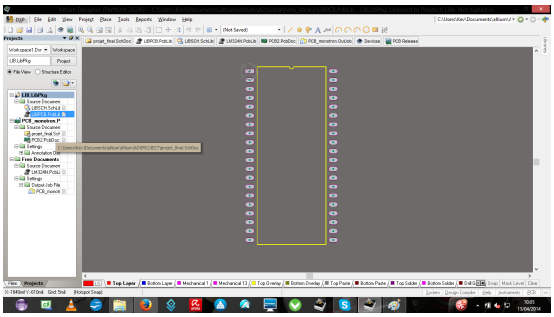

FIGURE 24 – foot print du micro processeur ARM

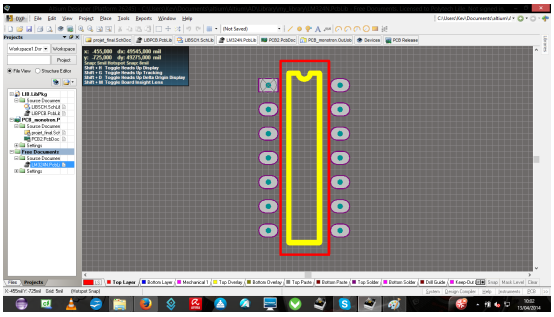

FIGURE 25 – foot print du LM324N

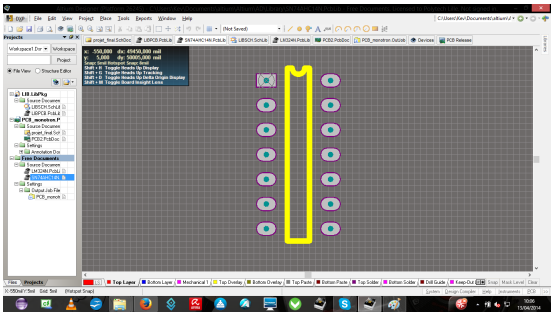

FIGURE 26 – foot print du SN74AHC14N

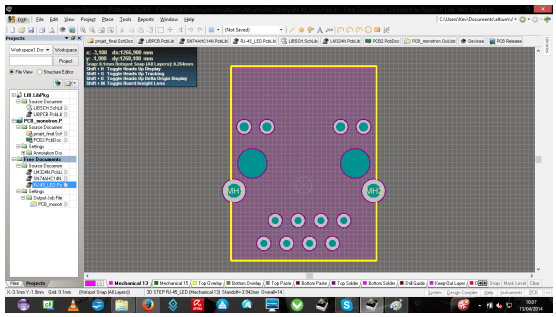

FIGURE 27 – foot print du RJ45

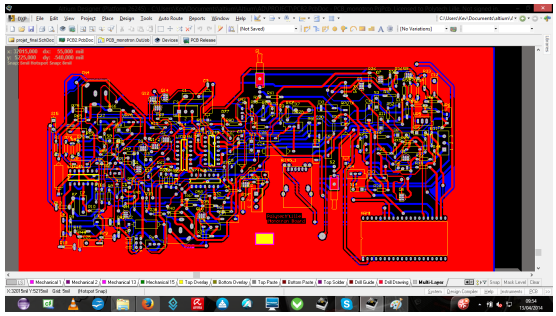

FIGURE 28 – PCB vue top

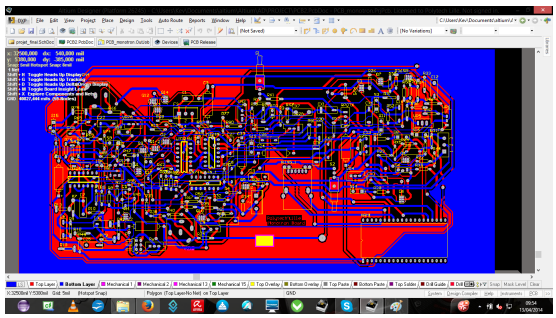

FIGURE 29 – PCB vue bottom

En annexe les empreintes de quelques composants manquants que l'on a du crée

Les codes générés pour le projet ont été rajoutés en lien sur le TWIKI au format TXT et PDF