Smart City Building
Rapport technique

Basile Maddalena Nicolas Héricher Vadym Lozovski Matéo Guy

Introduction

Le projet de Smart city - Building est une maquette d'un appartement (ou une maison) intelligent(e). Il consiste à réaliser plusieurs fonctions automatisées : la gestion énergétique et économe des lumières, la gestion des lumières par claquement des mains ou sifflement, et la simulation de vie dans l'appartement dans le but d'éviter un potentiel cambriolage.
Le but de ce projet est donc de réaliser un lieu de vie miniature et autonome qui proposerait de répondre à des problématiques environnementales et économiques d'économie d'énergie et de sécurité. En effet, un habitant moyen aura tendance par exemple à laisser ses volets fermés et d'allumer les lumières, voire d'oublier d'éteindre ces dernières en partant ou en allant se coucher. De même, les cambriolages sont soumis à un repérage des cambrioleurs, une simulation de vie dans l'appartement est une réponse à ce type de problèmes.

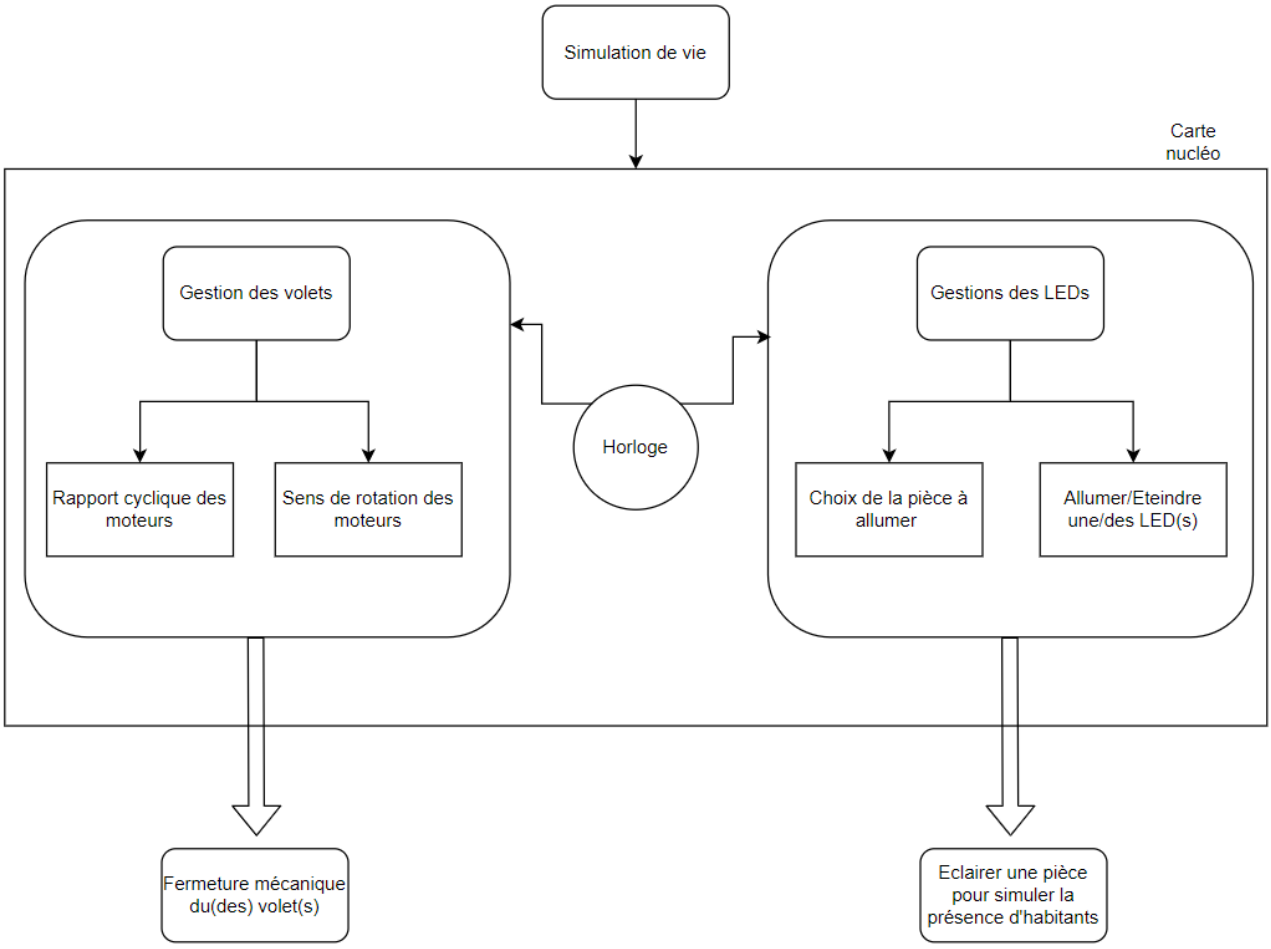
Problématique et motivation

Nous avions cogité sur la manière d'isoler et d'identifier le bruit d'un "clap" dans un environnement bruité. Après avoir envisagé un circuit contenant un passe bande, nous nous sommes vite rendu compte qu'il ne s'agissait pas d'un problème de fréquence, car les claps peuvent avoir des fréquences très variées et sont assez riches en harmoniques.

Découpage fonctionnel

Le découpage fonctionnel du projet se fera en 3 parties. Une première partie destinée à gérer l'éclairage de la maison en fonction de la luminosité extérieure, dans le but de limiter les consommation d'énergie. Une deuxième partie concerne la simulation de vie par l'automatisation de la fermeture et de l'ouverture des stores et par l'allumage de la lumière au sein de chaque pièce en fonction de l'heure de la journée. Le schéma ci-dessous, détaille plus précisément ce système de simulation de vie. Ce même système se décompose en deux parties : une partie liée à la gestion des volets et une partie concernant l'éclairage. Ces deux sous-systèmes sont liés par une horloge dont nous détaillerons les rouages plus loin.
Enfin, une troisième partie consiste à élaborer un système de gestion de la lumière dans l'appartement par simple claquement des mains. Ce système sera également capable de gérer la luminosité ambiante en fonction de la fréquence du signal sonore.

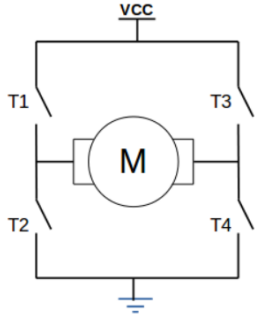
L'objectif de ce projet est de réaliser ces trois parties, et de les réunir pour qu'elles fonctionnent ensembles. Il est alors important d'établir des règles de priorités pour que l'ensemble des trois systèmes fonctionne correctement et sans encombre. Par exemple, si la maison est inhabitée, on mettra en route le programme de simulation de vie. Si quelqu'un y vit, on préfèrera désactiver ce même système et utiliser les deux autres. Enfin, pour des raisons évidentes, le système d'activation lumineuse par claquement de mains sera toujours prioritaire au système d'économie d'énergie.



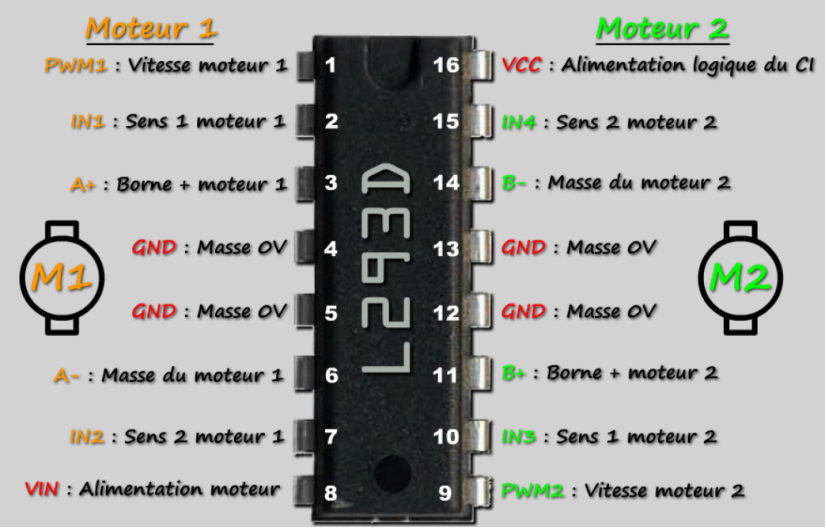
Système de store connecté à l'horloge

1) Présentation

Le système de store est composé d'un moteur que l'on contrôle grâce à un pont en H qui nous permet de faire varier le sens de rotation du moteur. Ce pont en H est réalisé par un composant intégré, le **L293D** dont les broches sont décrites ci-dessous :



Pont en H



Broche du L293D (source : https://electrotoile.eu/arduino-moteur-DC-shield.php)

dans notre cas nous n'utilisons qu'un seul moteur, car un seul store a pu être réalisé à temps. L'alimentation du circuit sera fournie par une batterie de 12V qui alimente tout le système. En dehors du moteur, une pièce cylindrique imprimée et associée au moteur permet de faire monter et descendre un rideau préalablement fixé à cette pièce. Le rideau sera lesté afin d'assurer son bon déroulement. La gestion du store est assurée par un algorithme présenté en annexe. Nous nous assurerons ensuite de modifier légèrement l'algorithme afin de pouvoir l'associer avec le reste du système ensuite.

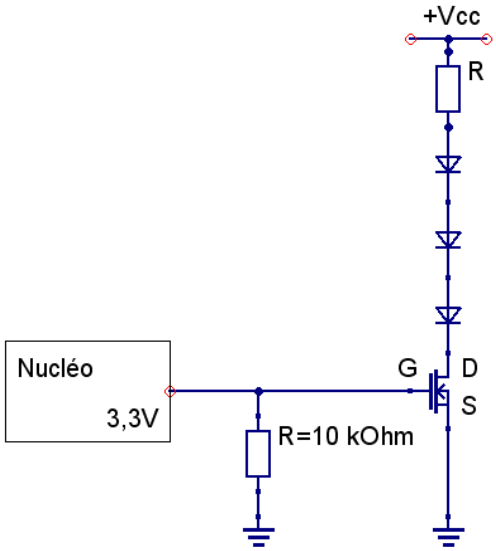
2) Algorithme

L'algorithme est un algorithme basique suivant deux boucles. La première ordonne l'ouverture du volet s'il est fermé. La seconde exécute la fonction inverse. La base du script est présentée en annexe. La course du volet ainsi que le temps de descente et de montée sont déterminés expérimentalement.

Système d'éclairage connecté à l'horloge

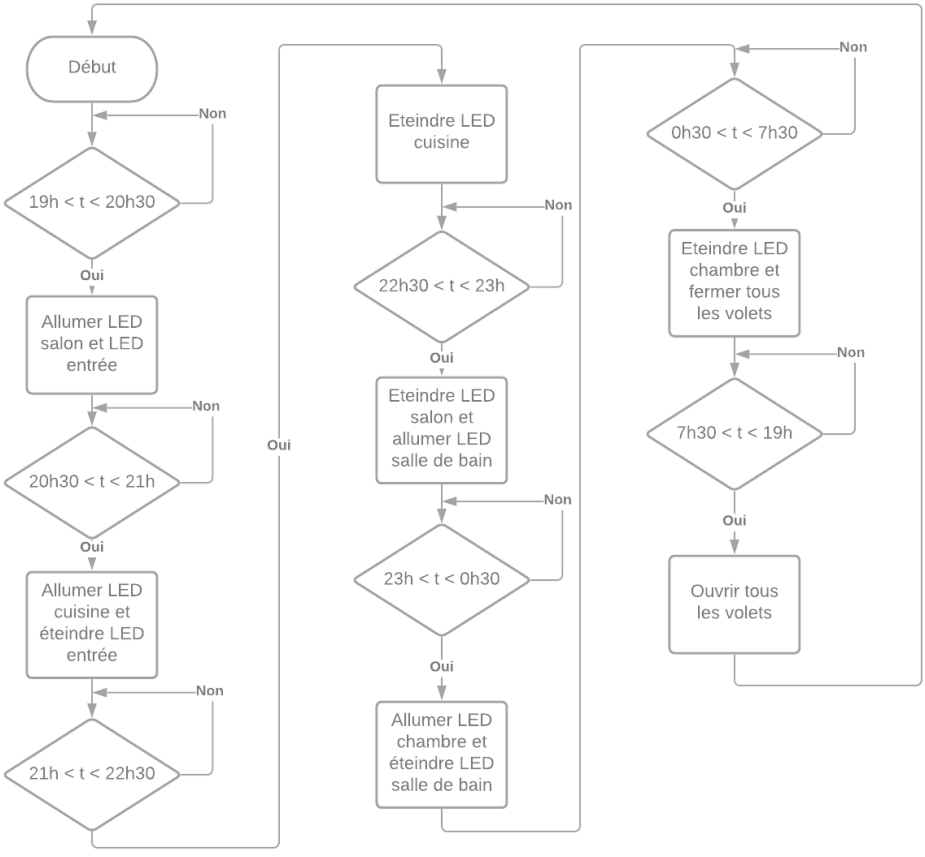
1) Présentation

Notre système d'éclairage périodique est constitué de LED branchées en série (Cf. schéma). Elles sont au nombre de 3 ou de 2. Le transistor permet la répartition de l'énergie électrique sur toute la ligne de LED et permet ainsi à toutes les LED de fonctionner normalement et avec une intensité maximale. Une résistance R de 100 Ohm permet de protéger la série de LED et une batterie de 12V les alimente. On répète ce circuit autant de fois qu'il y a de pièce dans la maison et on branche chaque compartiment à une broche unique de la carte Nucleo. Celle-ci permettra (Cf. Système d'éclairage connecté à l'horloge. Annexe) de commander en fonction de l'horloge, l'allumage et l'extinction des pièces.



2) Algorithme

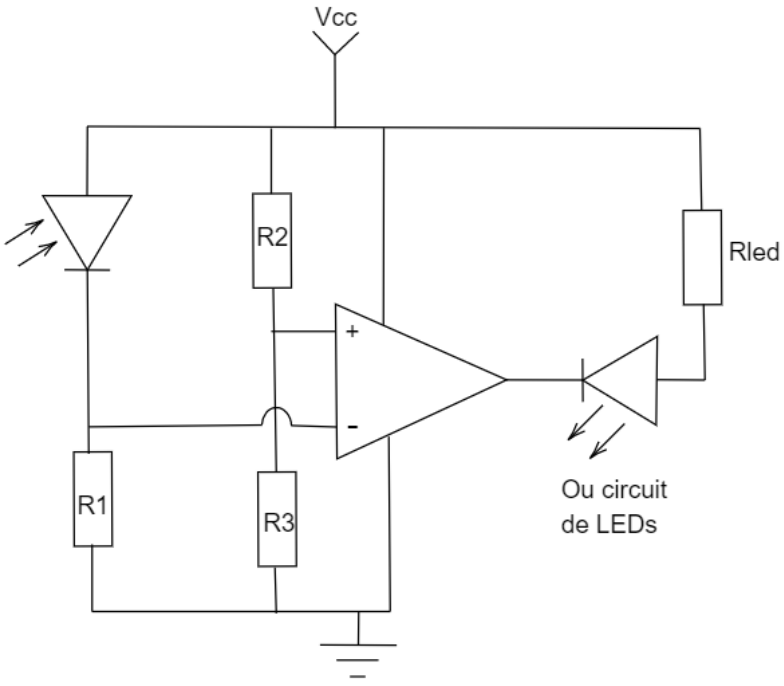
Le diagramme ci-dessous permet de mieux comprendre comment fonctionne notre horloge. On initialise le temps t avec la commande absolute_time (Cf. Annexe) puis on module et par le nombre de secondes dans une journée (86400s) afin que le programme se réinitialise tous les jours. On peut ensuite allumer ou éteindre différentes pièces de la maison et même fermer et ouvrir les stores en conditionnant l'intervalle de temps dans lequel doit se trouver t.



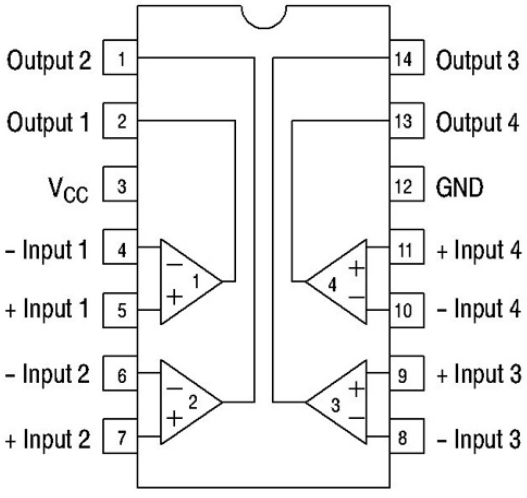
Système d'éclairage en fonction de la luminosité

1) Présentation

On cherche à ajuster l'éclairage à l'intérieur de l'appartement en fonction de l'éclairage extérieur. Ainsi, notre dispositif doit permettre un allumage des lumières à partir du moment ou l'éclairage externe est jugé trop faible pour permettre une certaine luminosité interne. Pour mettre cela en pratique, nous avons utilisé le circuit suivant (inspiré du TD6 de CéTI au semestre 5 : Capter une grandeur physique) :



L'AOP :



(source : microcontrollerslab.com)

Cet AOP contient en réalité 4 ALI TL081 (Notre circuit n'est branché qu'à un de ces ALI, par exemple le 3). Ce circuit contient un AOP LM339 : On appellera V_+ la tension au niveau de la borne + de l'ALI et V_- la tension au niveau de la borne -. Si $\varepsilon = V_+ - V_- > 0$, alors l'ALI est dit passant (la sortie est alors reliée à la masse et sa tension est nulle). Dans le cas contraire ($\varepsilon<0$), il est dit bloquant, et dans ce cas la tension de sortie de l'ALI vaut Vcc.

2) Fonctionnement du circuit

Ainsi, si l'ALI est passant ($V_s = 0$), il y aura une certaine tension aux bornes de la LED représentée sur le schéma, qui permettra de l'allumer. En revanche, s'il est bloquant, la tension en sortie valant Vcc, il n'y aura pas de différence de tension aux bornes de la LED (à la tension aux bornes de Rled près). Le but est ensuite de déterminer comment relier cette propriété à la luminosité extérieure. Avec le circuit représenté ci-dessus, la tension V_- change proportionnellement avec le flux de photons ϕ reçu par la photodiode. On a en effet $V_- = R1*k*\phi$. Les résistances R2 et R3 servent à établir un pont diviseur de tension, de manière à ce que $V_+ = R3/(R2+R3)*Vcc$. Afin de choisir les valeurs de R2 et R3, il nous faut donc fixer R1 pour avoir une tension raisonnable en V_- pour le changement de mode (bloquant/passant). Avec R1=1Mohm et la photodiode utilisée, on avait avec l'éclairage ambiant une tension V_- d'environ 1V. Reste donc à fixer R2 et R3 : on a cherché à avoir V_+=600mV. Cela donne R3/(R2+R3) = 0.5 avec Vcc = 12V. On fixe donc R2=1Mohm et R3=50kohm.

3) Câblage avec un circuit de LED

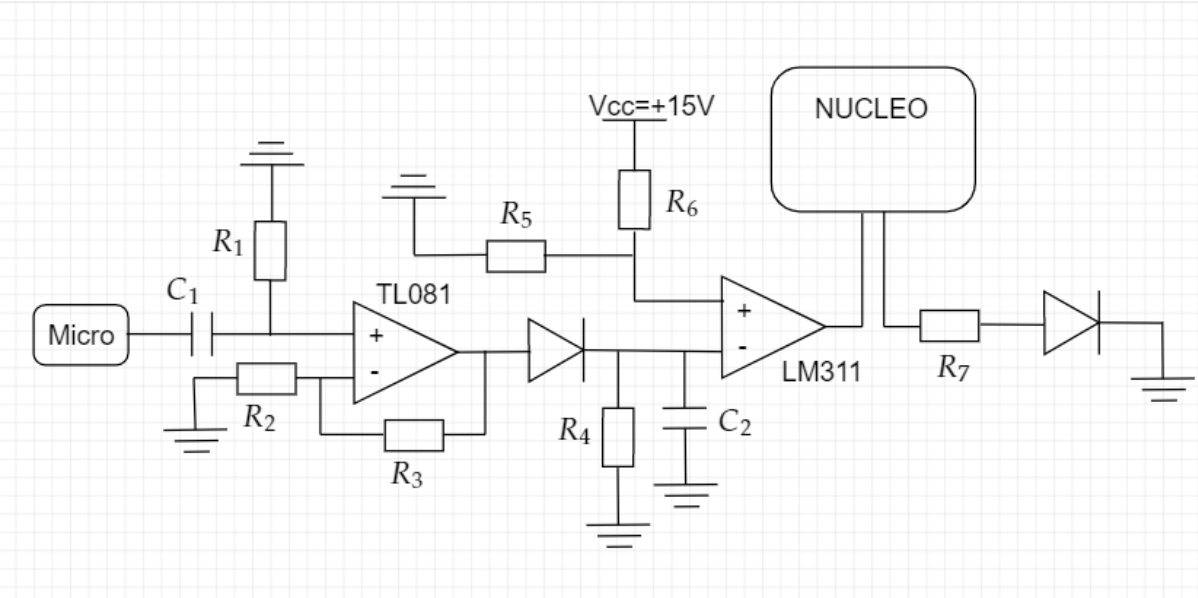
Nous avons par la suite décidé de combiner ce circuit avec le système antivol (Système d'éclairage connecté à l'horloge) permettant d'allumer des LED en différé. On a en effet utilisé le dispositif précédent en tant qu'alimentation du circuit de LED, situé à la place de l'unique LED utilisée auparavant.

Système d'éclairage avec clap

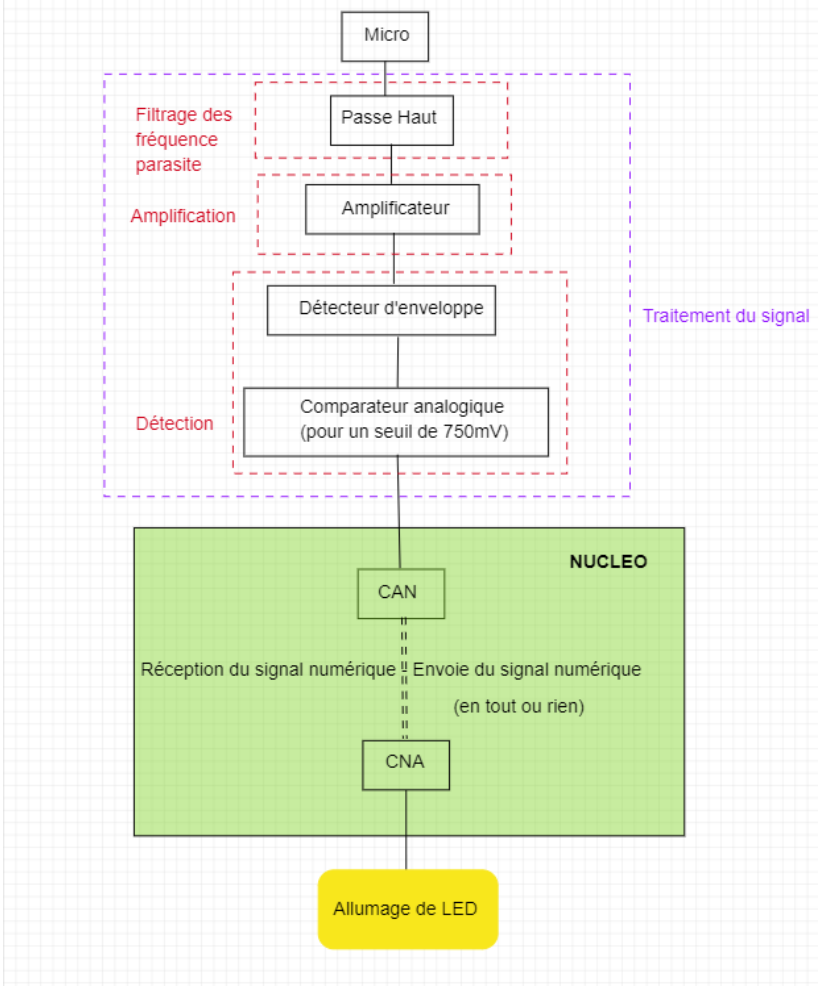
I - Eclairage à interrupteur sonore: Lédaclap

1) Présentation générale

Le système en question permet d'allumer et d'éteindre la lumière (d'une LED, modélisant la lampe) en tapant dans la main et en émettant ainsi un signal sonore déclencheur.



Le schéma bloc de ce système est représenté ci-après.



2) Amplification et filtrage

Le filtre passe-haut supprime la composante continue intrinsèque au microphone; pour ce faire il faut choisir une grande capacité C1=4,7μF (on a prit R1=100Ω}). Le microphone envoie des signaux de très faible intensité, qui doivent être obligatoirement

amplifiés. On ajuste les résistances sur le 1er ALI, amplificateur non-inverseur pour avoir une x4 (R2=1kΩ et R3=3,9KΩ}) .

3) Détecteur de crêtes

Pour repérer un signal sonore assez court mais intense tel que le clap, on peut utiliser un détecteur de crêtes, qui agit sur un signal déjà amplifié et filtré. Il s'agit d'un redresseur (une diode) suivi d'un lisseur (une résistance et une capacité en parallèle). Le redresseur rend le courant positif et le lisseur joue le rôle d'un passe-bas, en gardant que l'enveloppe (associée aux basses fréquences). Il faut bien choisir les valeurs des composants pour ne pas couper les fréquences utiles; après plusieurs essais on a conclu que les valeurs optimales sont: C2=220nF et R4=10kΩ }.

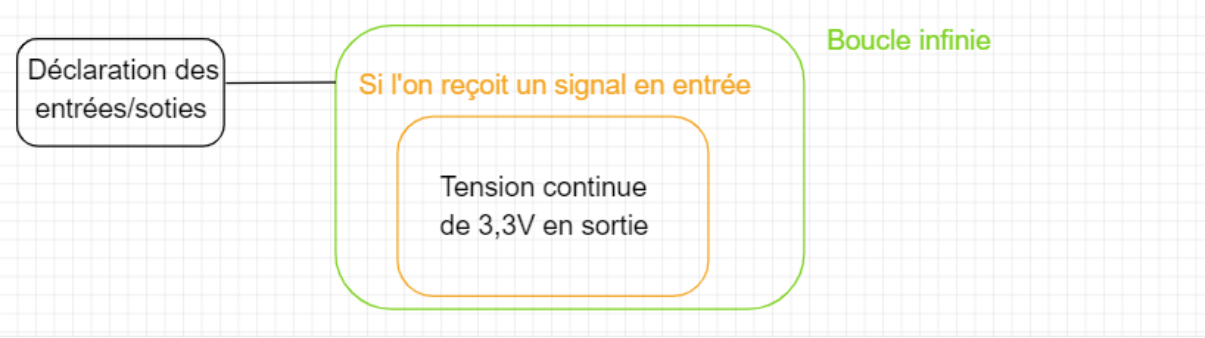
4) Comparateur

Par la suite, il faut comparer la valeur de la tension de l'enveloppe détectée à la valeur seuil. Après de nombreux tests répétitifs, afin de garder une marge entre le bruit ambiant (évaluer à 0,5V environ) et le déclenchement du système on a choisi: Useuil=750mV}. Pour s'approcher au maximum d'un prototype pouvant être mis en vente, on a essayé de minimiser les tâches reposant sur le micro-contrôleur; pour épargner le temps nécessaire à l'exécution des algorithmes, on a privilégié le comparateur analogique. Pour ce faire on s'est servi d'un ALI de modèle LM311, alimenté asymétriquement entre 0 et +15V...

Pour optimiser l'utilisation des sources de tension continue, on ajuste la valeur de la tension seuil grâce à un pont diviseur (R5=3,9kΩ et R6=220Ω) connecté à une source de +15V. En effet: 15/20=0,75. Quand l'enveloppe du signal perçu dépasse 750 mV, LM311 envoie un échelon de courant de 50mA vers le microcontrôleur.

5) Microcontrôleur

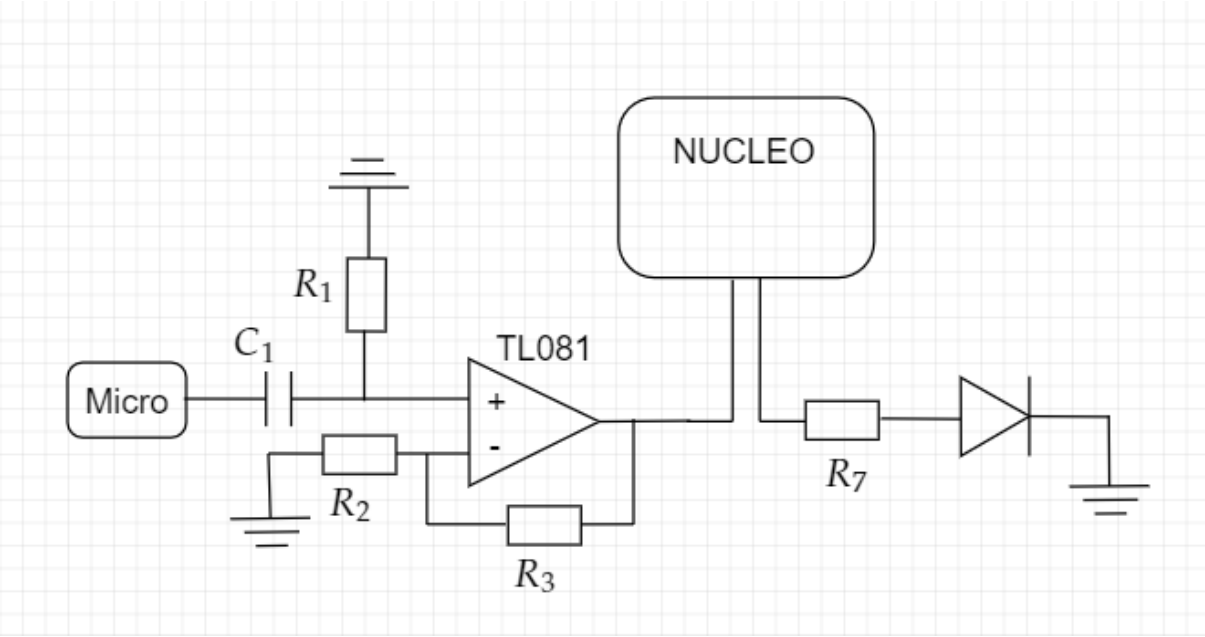
Le traitement numérique se fait à l'aide de la carte Nucleo, dont on n'utilisera qu'une entrée et une sortie analogique (le microcontrôleur doit être également connecter à la masse), en tout ou rien; ce qui augmente la rapidité et la réactivité du système et simplifie beaucoup le code sur mbed. Dès que la carte reçoit l'échelon de courant, elle crée une tension de 3,3V à la sortie connectée à la LED; cette dernière étant protégée par une résistance de 220Ω.



II - Réglage de luminosité en fonction du signal sonore: Whistlight

1) Présentation générale

Pour aller plus loin, on a songé à concevoir un système qui fait varier la luminosité de la LED en fonction de la fréquence de sifflement de l'utilisateur. La première partie du montage est similaire au précédent. Or dans ce cas on se voit dans l'obligation de reposer la plupart des taches sur les fonctionnalités du microcontrôleur. On peut tracer une analogie entre ce système et un appareil de mesures.



2) Codage

Ici les codes nécessitent un plus grand investissement. En effet, il faut d'abord obtenir le spectre du signal amplifié; la première partie du code le fait avec un algorithme de calcul rapide de la transformée de Fourier discrète. D'autre part, il faut identifier les plages de fréquence atteignable par le sifflement du constructeur et les ajuster à ceux des consommateurs. Cette plage nous servira de référence de normalisation pour retrouver la valeur du rapport cyclique qu'on enverra à la LED (compris entre 0 et 1).

Annexe

Code : Gestion du sens de rotation d'un moteur

```
#include "mbed.h"
PwmOut moteur_cc1(D10);
PwmOut moteur_cc2(D9);
Serial pc(USBTX, USBRX);
int main() {
    double rc;
    char c;
    pc.baud(115200);
    moteur_cc1.period_ms(10);
    moteur_cc2.period_ms(10);
    moteur_cc1.write(0);
    moteur_cc2.write(0);
    pc.printf("Nouvelle valeur ? D RC (D = H ou A - RC entre 0.0 et 1.0) \r\n");

    while(1){
        if (pc.readable()) {
            pc.scanf("%c %lf", &c, &rc);
            pc.printf("D = %c, RC = %lf \r\n", c, rc);
            if(c == 'H'){
                moteur_cc2.write(0);
                moteur_cc1.write(rc);
            }
            if(c == 'A'){
                moteur_cc1.write(0);
                moteur_cc2.write(rc);
            }
            pc.printf("Nouvelle valeur ? D RC (D = H ou A - RC entre 0.0 et 1.0) \r\n");
        }
    }
}
```

Code pour le système d'éclairage connecté à l'horloge

```
#include "mbed.h"
Serial pc(USBTX, USBRX);
DigitalOut led_salon(D10);
DigitalOut led_chambre(D9);
DigitalOut led_cuisine(D8);
DigitalOut led_entree(D11);
DigitalOut led_salle_eau(D7);
int main() {

    // Partie LED //
    int t_a;
    int hour;
    int minu;
    set_time(0);  // Set RTC time to the actual date
    while(1) {
        time_t absolute_time;
        absolute_time = time(NULL);
        t_a = absolute_time%13;
        hour = (t_a / 10);
        minu = (t_a % 10)*6;
        printf("heure = %d:%d \n", t_a);
        if (0 <= t_a && t_a < (2));
        {
            led_cuisine = 1;
            led_salon = 0;
            led_chambre = 0;
            led_entree = 1;
            led_salle_eau = 0;
        }
        if ((2) <= t_a && t_a < (4))
        {
            led_cuisine = 0;
            led_salon = 1;
            led_chambre = 0;
            led_entree = 1;
            led_salle_eau = 0;
        }
        if ((4) <= t_a && t_a < (6))
        {
            led_cuisine = 0;
            led_salon = 1;
            led_chambre = 0;
            led_entree = 0;
            led_salle_eau = 1;
        }
        if ((6) <= t_a && t_a < (8))
```

```
        {
            led_cuisine = 0;
            led_salon = 0;
            led_chambre = 1;
            led_entree = 0;
            led_salle_eau = 0;
        }
        if ((8) <= t_a && t_a < (10))
        {
            led_cuisine = 0;
            led_salon = 0;
            led_chambre = 0;
            led_entree = 0;
            led_salle_eau = 0;
        }
        if ((10) <= t_a && t_a < (12))
        {
            led_cuisine = 1;
            led_salon = 1;
            led_chambre = 1;
            led_entree = 1;
            led_salle_eau = 1;
        }
    }
}
```

Code associant la gestion des LEDs et du volet :

```
#include "mbed.h"
Serial pc(USBTX, USBRX);
DigitalOut led_salon(D10);      // Association des LEDs
DigitalOut led_chambre(D9);
DigitalOut led_cuisine(D8);
DigitalOut led_entree(D11);
DigitalOut led_salle_eau(D7);
PwmOut moteur_cc1(D6);          // Association des sens de rotation du moteur
PwmOut moteur_cc2(D5);
int main() {
    double rc;                  // Rapport cyclique
    char vol;                   // chaine de caractère associé à l'état
                                // ouvert/fermé du volet
    moteur_cc1.period_ms(10);   // Imosition des périodes des moteurs
    moteur_cc2.period_ms(10);
    moteur_cc1.write(0);        // Moteurs immobiles au début du programme
    moteur_cc2.write(0);
    rc = 0.1;
    vol = 'f';                  // Le programme commence sur cet état du volet,
                                // accordé au planning d'une journée

    int t_a;        // temps absolu
    int hour;
    int minu;
    set_time(0); // Set RTC time to the actual date
    while(1) {
        time_t absolute_time;
        absolute_time = time(NULL);
        t_a = absolute_time%60; // Division d'une journ´ee sur 4min hour = (t_a / 10);
        minu = (t_a % 10)*6;
        printf("heure = %d:%d \n", hour);
        hour = t_a*24/60;
        if (0.5 <= hour && hour < (7.5));
        {

            led_cuisine = 1;
            led_salon = 0;
            led_chambre = 0;
            led_entree = 0;
            led_salle_eau = 0;
        }
        if ((7.5) <= hour && hour  < (19))
        {
            if (vol=='f'){// Ouverture du volet
            moteur_cc1.write(0);
            moteur_cc2.write(rc);
```

```
    wait_us(2500000);
    moteur_cc1.write(0);
    moteur_cc2.write(0);
    vol='o';}

    led_cuisine = 1;
    led_salon = 0;
    led_chambre = 0;
    led_entree = 0;
    led_salle_eau = 0;
}
if ((19) <= hour && hour  < (20.5))
{
    led_cuisine = 1;
    led_salon = 1;
    led_chambre = 0;
    led_entree = 1;
    led_salle_eau = 0;
}
if ((20.5) <= hour && hour  < (21))
{
    led_cuisine = 1;
    led_salon = 1;
    led_chambre = 0;
    led_entree = 0;
    led_salle_eau = 0;
}
if ((21) <= hour && hour < (22.5)) {
    led_cuisine = 1;
    led_salon = 1;
    led_chambre = 0;
    led_entree = 0;
    led_salle_eau = 0;
}
if ((22.5) <= hour && hour  < (23)) {
    led_cuisine = 1;
    led_salon = 0;
    led_chambre = 0;
    led_entree = 0;
    led_salle_eau = 1;
}
if ((23) <= hour || hour < (0.5)) {
    if(vol=='o'){                    // Fermeture du volet
    moteur_cc2.write(0);
    moteur_cc1.write(rc);
    wait_us(1500000);
```

```
                moteur_cc2.write(0);
                moteur_cc1.write(0);
                vol='f';}

                led_cuisine = 1;
                led_salon = 0;
                led_chambre = 1;
                led_entree = 0;
                led_salle_eau = 0;
            }
            wait(1);
        }
    }
```

Codes pour le Whistlight

```
#include "mbed.h"
#include "arm_math.h"
/* Include mbed-dsp libraries */
#include "dsp.h"
#include "arm_common_tables.h"
#include "arm_const_structs.h"

#define SAMPLES                 512             /* 256 real party and 256 imaginary parts */
#define FFT_SIZE                SAMPLES / 2     /* FFT size is always the same
size as we have samples, so 256 in our case */

float32_t Input[SAMPLES];
float32_t Output[FFT_SIZE];
bool      trig=0;
int       indice = 0;

PwmOut ma_sortie_pwm(D10);
DigitalOut myled(LED1);
AnalogIn   myADC(A0);
AnalogOut  myDAC(A2);
Serial     pc(USBTX, USBRX);
Ticker     timer;

void sample(){
    myled = 1;
    if(indice < SAMPLES){
        Input[indice] = myADC.read() - 0.5f;    //Real part NB removing DC offset
        Input[indice + 1] = 0;                  //Imaginary Part set to zero
        indice += 2;
```

```
    }
    else{ trig = 0; }
    myled = 0;
}

int main() {
    float maxValue;             // Max FFT value is stored here
    uint32_t maxIndex;          // Index in Output array where max value is
    double rc;
    rc = 0.5;
    ma_sortie_pwm.period_ms(10);
    ma_sortie_pwm.write(rc);
    while(1) {
        if(trig == 0){
            timer.detach();
            // Init the Complex FFT module, intFlag = 0, doBitReverse = 1
            //NB using predefined arm_cfft_sR_f32_lenXXX, in this case XXX is 256
            arm_cfft_f32(&arm_cfft_sR_f32_len256, Input, 0, 1);

            // Complex Magniture Module put results into Output(Half size of the Input)
            arm_cmplx_mag_f32(Input, Output, FFT_SIZE);
            Output[0] = 0;
            //Calculates maxValue and returns corresponding value
            arm_max_f32(Output, FFT_SIZE/2, &maxValue, &maxIndex);

            myDAC=1.0;      //SYNC Pulse to DAC Output
            wait_us(20);     //Used on Oscilliscope set trigger level to the highest
            myDAC=0.0;      //point on this pulse

            for(int i=0; i < FFT_SIZE / 2; i++){
                myDAC=(Output[i]) * 0.9;   // Scale to Max Value and scale to 90 / 100
                wait_us(10);                //Each pulse of 10us is 50KHz/256 = 195Hz resolution
            }
            myDAC=0.0;
            pc.printf("MAX = %lf, %d \r\n", maxValue, maxIndex*195);
            if( maxValue>0.2){
                pc.printf("MAX = %lf, %d \r\n", maxValue, maxIndex*195);
                rc = (maxIndex*195.0-2200)/1600.0;      //Normalisation de la plage de fréquence
                if(rc > 1)  rc = 1;                     // Ajustement des limites
                if(rc < 0)  rc = 0;
                pc.printf("\tRC = %lf\r\n", rc);
                ma_sortie_pwm.write(rc);
            }

        wait_ms(200);
            wait(0.2);
            trig = 1;
            indice = 0;
            timer.attach_us(&sample,40);      //20us 50KHz sampling rate
        }
    }
}
```

Codes pour Lédaclap

```
#include "mbed.h"

    DigitalOut led(D13);
    InterruptIn clap(D8);


int main()
{
    while (1) {
        if(clap == 1){
            led = !led;
        wait_ms(15);}
    }
}
```