

Les Universités du Tetal@b



Des Proto (pseudo) Threads pour l'Arduino



Proto Thread : mise en pratique

- ◆ Vous avez compris la 1ère partie ?
- ◆ Alors rien ne vaut une mise en œuvre !



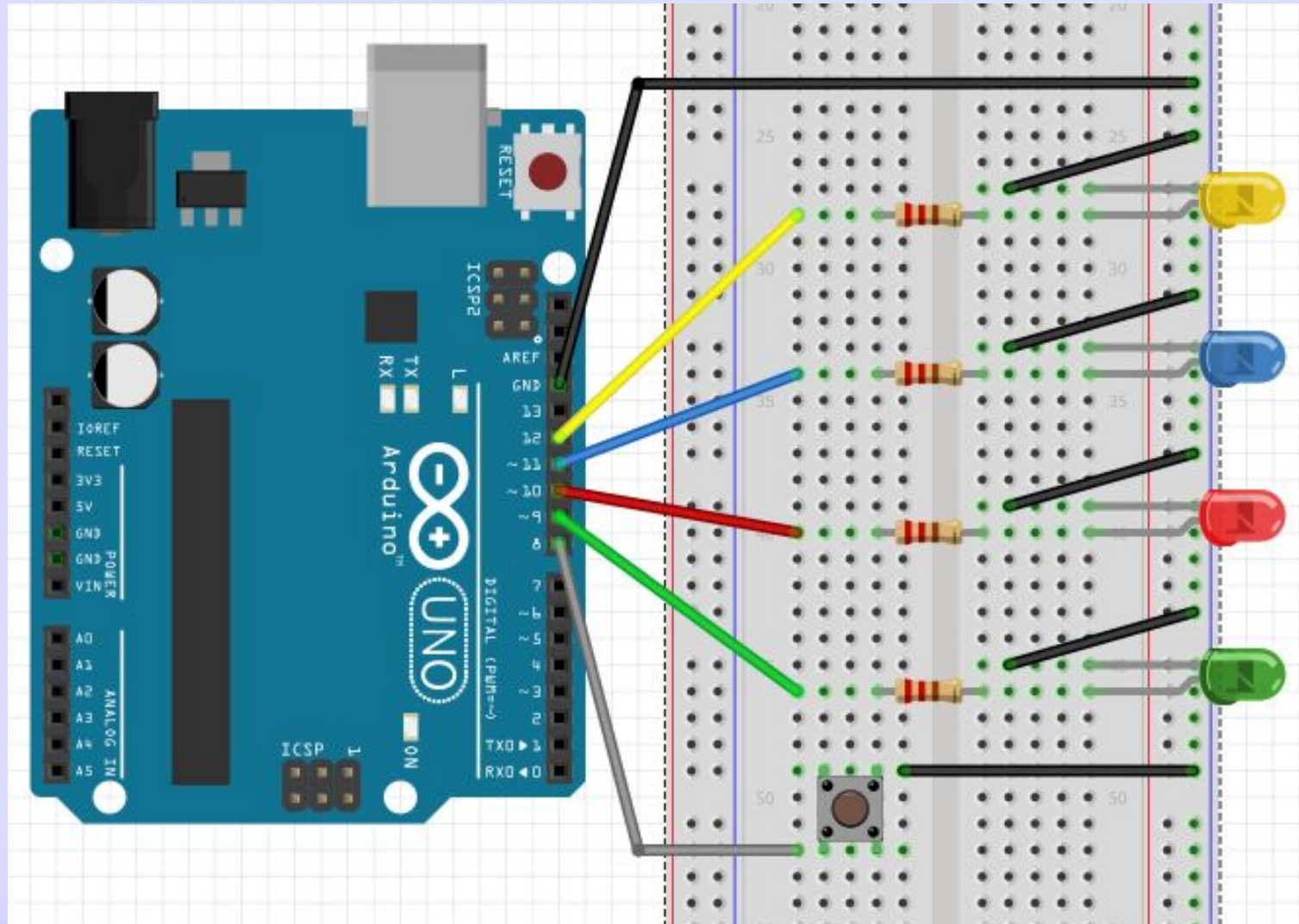
Spécifications générales

Cette proposition d'étude provient d'une réalisation concrète.

◆ Gérer 4 Leds selon des cycles évolutifs:

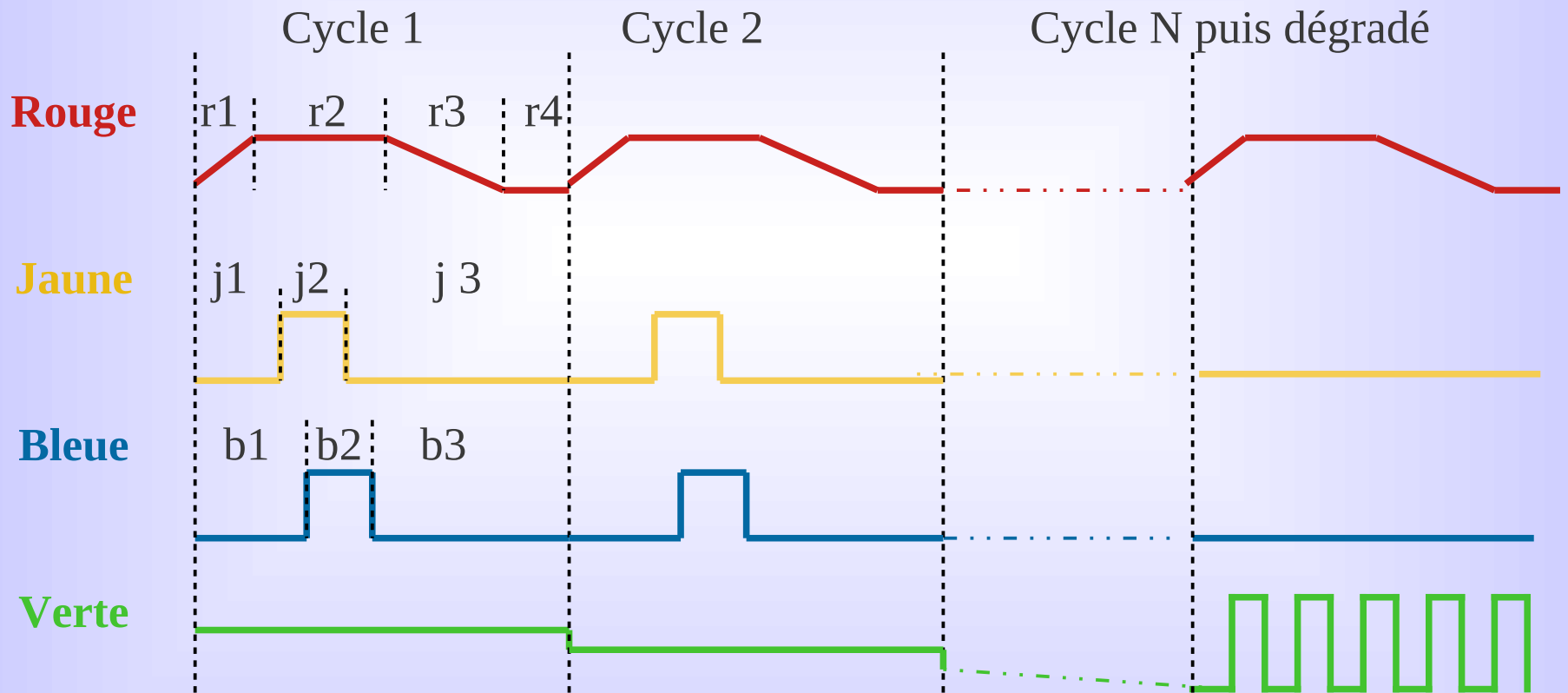
- Les 3 premières sont activées par un poussoir, avec des modalités et des délais variés dans le cycle.
- La 1ère, à l'appui du poussoir, a un allumage progressif, un palier, une extinction, une attente, puis recommence le cycle.
- La 2ème et la 3ème s'allument un certain temps à un instant du cycle tant que le mode dégradé n'est pas actif.
- La 4ème Led commence à 100 % et baisse de luminosité à chaque cycle. Après N cycles le système passe en mode dégradé, inhibe les leds 2 et 3 et la led 4 clignote rapidement.

Câblage sur plaque d'essai



Chronogramme

- t_0 à l'appui du poussoir

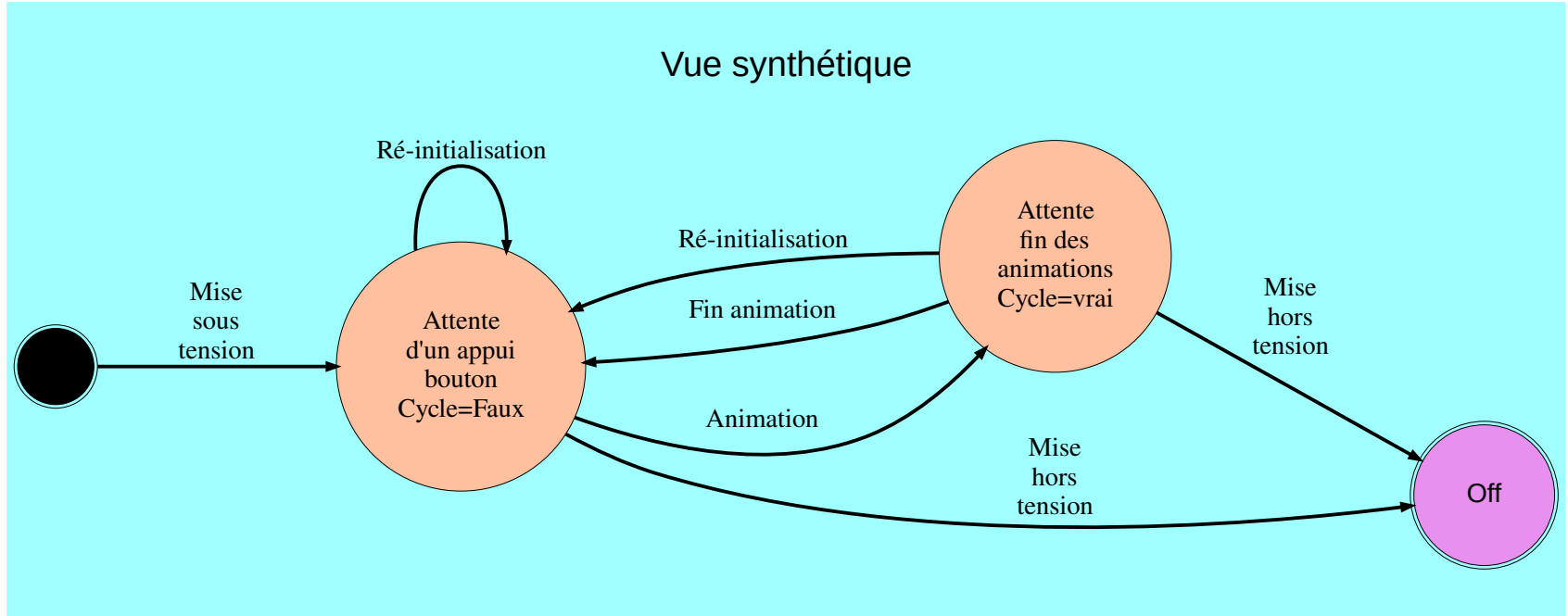


Description / chronogramme

- ◆ le compteur de cycles est initialisé à N et décrémenté à chaque appui sur le bouton
- ◆ **Led1** PWM passe du niveau PWM 0 à 254 sur r1, puis allumée sur r2, passe de PWM 254 à 0 sur r3, éteinte sur r4
- ◆ **Led2** éteinte sur j1, allumée sur j2, éteinte sur j3
- ◆ **Led3** éteinte sur b1, allumée sur b2, éteinte sur b3
- ◆ **Led4** PWM 254 au cycle 1 et passe à PWM 0 quand le compteur est nul, puis clignote à 1hz en mode dégradé.

Diagramme des états

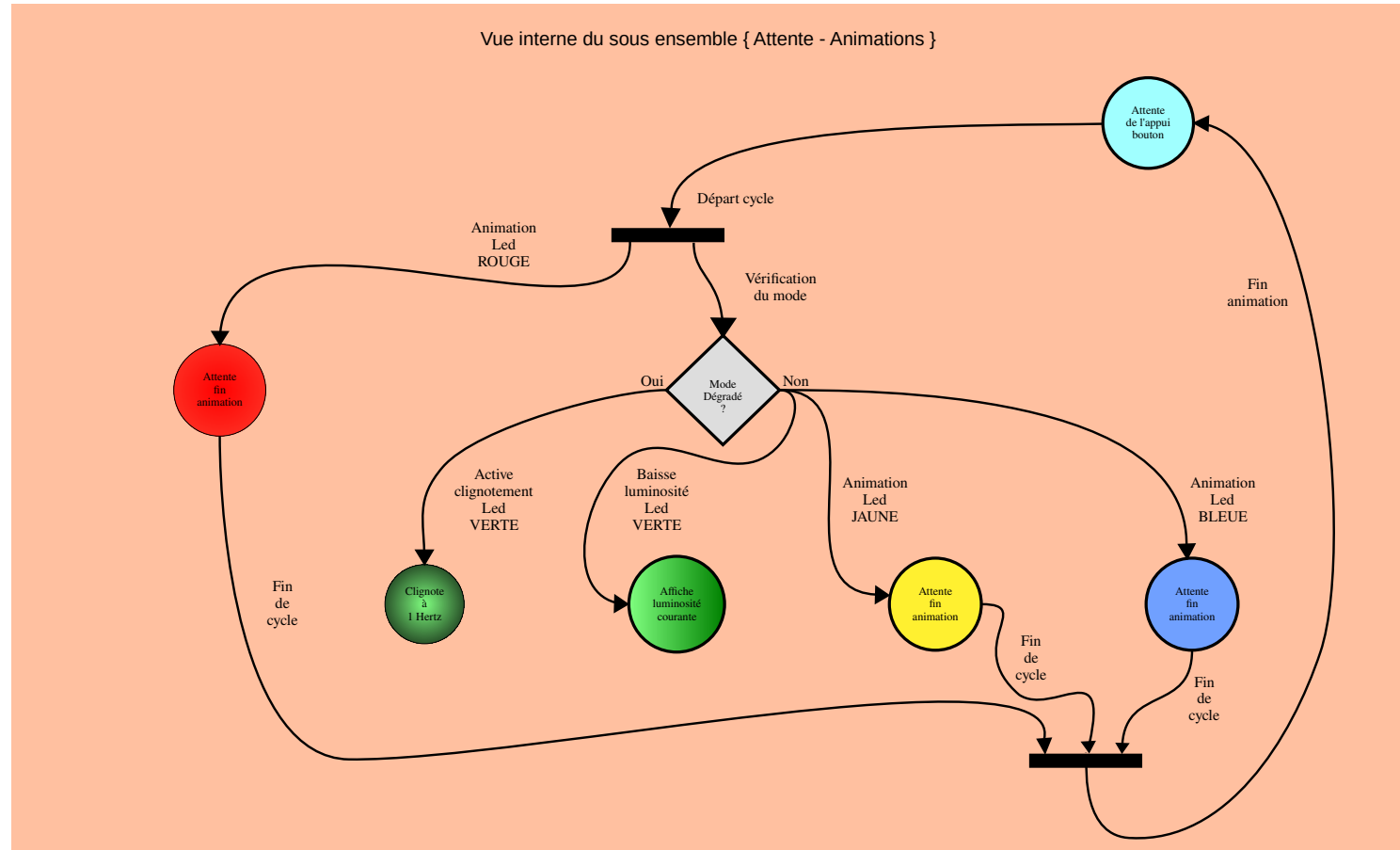
- ◆ L'animation est modélisée avec le contexte associé au micro-contrôleur.



États pour l'animation du dragon.

Diagramme d'activité

- ◆ États & transitions de la boucle de traitement.



États pour l'animation du dragon.

Les questions à se poser

- ◆ Combien de Protothreads ?
- ◆ Synchroniser les opérations dans un cycle ?
- ◆ Paramètres de contrôle / led / action ?
- ◆ Constantes / paramètres globaux ?
- ◆ Quelles dépendances entre eux ?
- ◆ Qu'est-ce qui pose problème ?
- ◆ Vous avez des idées ? Allons-y !

Analyse

- ◆ Combien de Proto Threads (PT) ?
 - 6 états repérés dans le diagramme d'activité
 - Il y a 4 LEDs et le poussoir, => 5 PT ?
 - Et 2 modes sur la Verte => 6 PT ?
 - À chaque variation de la Rouge il faut rendre la main pour laisser du temps aux autre PT => 7 PT
- ◆ Quels sont leurs paramètres ?
 - Les durées des cycles – le dernier pour les LED R, J, B
 - Éventuellement la 1/2 période pour clignoter la Verte
 - On peut alors écrire une boucle principale assez simple

Constantes et variables globales

◆ Variables globales et synchronisation

- On se cale sur une durée de cycle nommée *periode*. Le PT du bouton poussoir initie le début de la période et attend sa terminaison. L'information est connue des autres PT par l'intermédiaire d'un booléen global. Chaque PT attend ce départ et peut ensuite tester où il en est dans la période pour effectuer les actions requises à point nommé. Le mode dégrade requiert un booléen et un compteur.

◆ Ce qui donne dans le code :

```
#include <pt.h>
static struct pt ptP, ptR, ptRv, ptJ, ptB, ptVc, ptVd ; // proto threads
bool cycle = false ; uint16_t periode = 10000 ; // gestion du cycle
Bool degrade = false ; int nb_cycles = 0 ; const int deb_cycle_ko = 255 ;
```

Appel protothreads avec Arduino

◆ Boucle principale avec 6 PT

```
void loop() {  
    ptPoussoir(&ptP);          // gestion appui poussoir  
    ptRouge(&ptR, r1, r2, r3); // pilotage cycle Rouge  
    ptJaune(&ptJ, j1, j2);     // pilotage cycle jaune  
    ptBleue(&ptB, b1, b2);     // pilotage cycle Bleue  
    ptVerteC(&ptVc);           // pilotage Verte compte à rebours  
    ptVerteD(&ptVd, 500);      // pilotage Verte mode dégradé  
}
```

◆ Mais où est passé le 7ème PT ?

- On l'active depuis ptRouge avec PT_WAIT_THREAD :

```
static int ptRouge(struct pt *pt, uint16_t debut, uint16_t plafond, uint16_t fin)  
...  
    // PWM de 0 à 254 par pas de 1 et palier de 7 millisecondes  
    PT_WAIT_THREAD(pt, ptVarRouge(&ptRv, 0, 254, 1, 7));  
...  
}
```

Action bouton poussoir

- ◆ Le décor est en place, passons aux acteurs
- ◆ L'entrée en scène : le PT ptP du poussoir

```
static int ptPoussoir(struct pt *pt) {
    PT_BEGIN(pt);
    while(1) {
        PT_WAIT_UNTIL(pt, capterPoussoir() ); // appel fonction anti rebonds
        cycle=true;  nb_cycles++ ;           // c'est parti pour un tour
        if (nb_cycles == deb_cycle_ko ) mode_degrade = true ;
        started=millis();
        PT_WAIT_UNTIL(pt, millis() - started > periode ); // terminé
        cycle=false;                                // on signale au autres PT
    }
    PT_END(pt);
}
```

Alarme clignotante

◆ Le PT ptVd clignotant du mode dégradé

```
static int ptVerteD(struct pt *pt, int duree)
{
    Static unsigned long = ttl ;
    PT_BEGIN(pt);
    while ( mode_degrade ) {
        ttl = millis + duree ;
        PT_WAIT_UNTIL(pt, millis() > ttl);
        digitalWrite(VERTE, !digitalRead(VERTE));
    }
    PT_END(pt);
}
```

Gestion brillance Led Verte

- ◆ Le PT ptVc du compte à rebours

```
static int ptVerteC(struct pt *pt)
{
    static uint16_t ratio = 0;
    BEGIN(pt);
    While ( ! mode_degrade && cycle ) {
        ratio = map(nb_cycles, 0, 254, 254, 0);
        ledPWM(VERTE, ratio);
        PT_WAIT_UNTIL(pt, millis() - started > periode ); // fin de cycle
    }
    PT_END(pt);
}
```

- ◆ On choisi N=254, le nombre de cycles avant le mode dégradé

Animation Led Jaune

◆ Le PT ptJ de Jaune

```
static int ptJaune(struct pt *pt,  uint16_t offset, uint16_t duree )
{
    static unsigned long ttl = 0;
    PT_BEGIN(pt);
    while( ! degrade && cycle ) {
        ttl = millis() + offset ;
        PT_WAIT_UNTIL(pt, millis() > ttl);
        digitalWrite(JAUNE, HIGH);
        ttl = millis() + duree ;
        PT_WAIT_UNTIL(pt, millis() > ttl);
        digitalWrite(JAUNE, LOW);
        PT_WAIT_UNTIL(pt, ! cycle );
    }
    PT_END(pt);
} // Jaune
```

◆ Le code de Bleue est pareil avec BLEUE en lieu de JAUNE

Animation Led Rouge

◆ Le PT ptR de Rouge

```
static int ptRouge(struct pt *pt, uint16_t debut, uint16_t plafond, uint16_t fin)
{
    static unsigned long ttl = 0;
    PT_BEGIN(pt);
    while ( cycle ) {
        ttl = millis() + debut ;
        PT_WAIT_THREAD(pt, ptVarRouge(&ptRv, 0, 254, 1, 7)); // variation de 0 à 254
        PT_WAIT_UNTIL(pt, millis() > ttl);
        ttl = millis() + plafond ;
        digitalWrite(ROUGE, HIGH); // période allumée à 100%
        PT_WAIT_UNTIL(pt, millis() > ttl);
        ttl = millis() + fin ;
        PT_WAIT_THREAD(pt, ptVarRouge(&ptRv, 254, 0, -1, 7)); // variation de 254 à 0
        PT_WAIT_UNTIL(pt, millis() > ttl);
        digitalWrite(ROUGE, LOW);
        PT_WAIT_UNTIL(pt, ! Cycle ); // attente fin de cycle
    }
    PT_END(pt);
} // rouge
```

Gestion brillance Led Rouge

◆ Le sous PT ptRv, rampe PWM pour Rouge

```
static int ptVarRouge(struct pt *pt, int debut, int fin, int inc, int palier)
{
    Static unsigned long = ttl ;
    static int i=0 ; // niveau PWM conservé entre appels
    PT_BEGIN(pt);
    ledPWM(ROUGE, debut) ;
    do {
        ttl = millis + palier ;
        PT_WAIT_UNTIL(pt, millis() > ttl);
        i=i+inc; // le niveau PWM est incrémenté en + ou -
        ledPWM(ROUGE, i) ;
    } while (i != fin );
    PT_END(pt);
} // gestion PWM rouge
```

That's all Folks!



Merci pour votre attention :-)