

Les Universités du Tetal@b



Des Proto (pseudo) Threads pour l'Arduino



Proto Thread : la pratique

- ◆ Vous avez compris la 1ère partie ?
- ◆ Alors rien ne vaut une mise en œuvre !
- ◆ Simulons la gestion simplifiée d'une μ serre* sur une plaque d'essai.

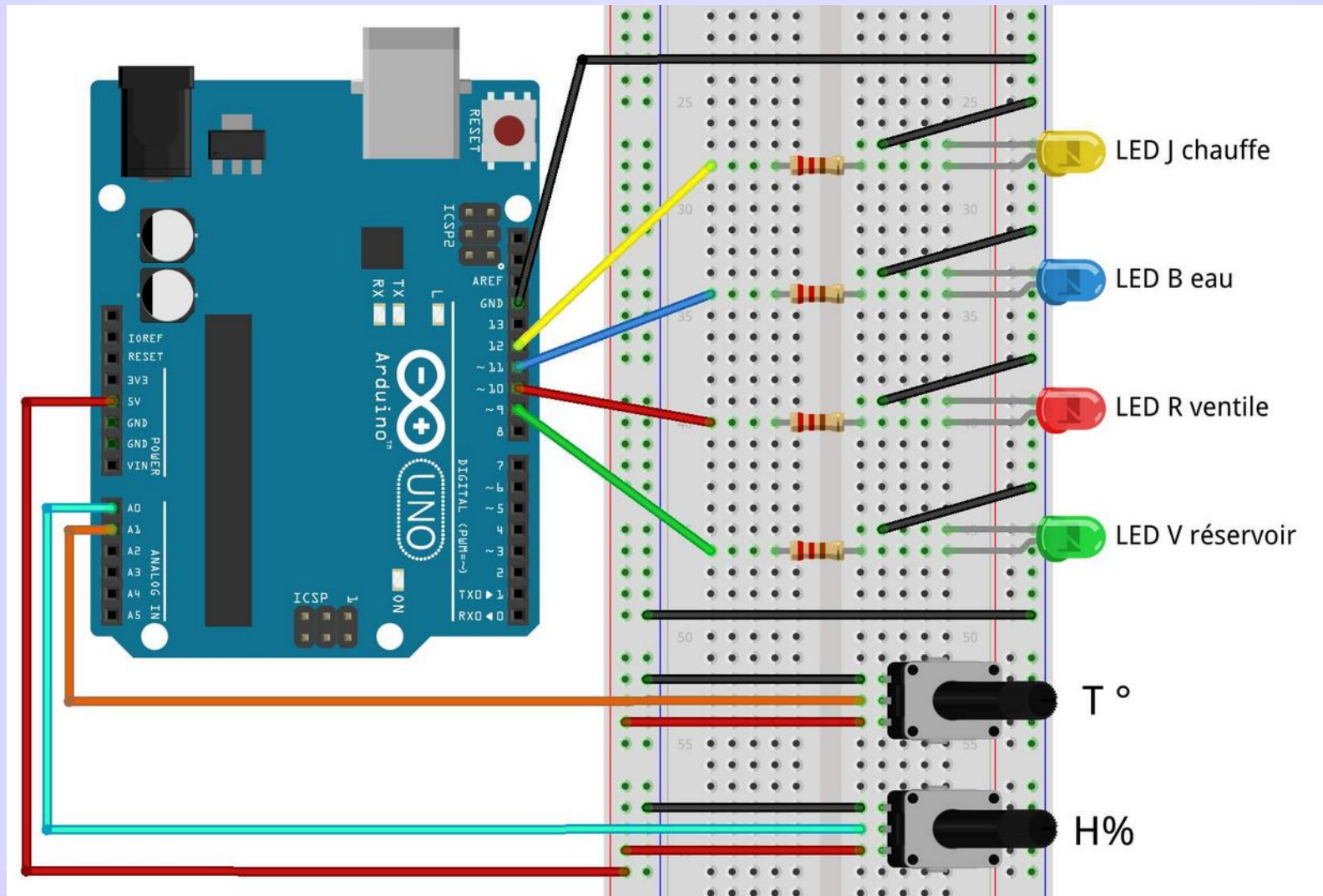


* Le Tetalab ne présume rien de l'usage possible d'une μ serre et se dégage de toute responsabilité en cas d'usage illégal ;-)

Spécifications générales

- ◆ Nous simplifions en négligeant l'éclairage
- ◆ Nous visualisons les actions avec 4 LEDs
 - ◆ La LED jaune signale le chauffage,
 - ◆ La LED bleu signale l'arrosage.
 - ◆ La LED Rouge en mode PWM signale le % du ventilateur
 - ◆ La LED Verte en mode PWM décroît avec le niveau du réservoir, et clignote s'il est vide.
- ◆ Nous simulons la situation avec 2 potentiomètres
 - ◆ Un câblé sur A0 pour la variation d'humidité H%
 - ◆ Un câblé sur A1 pour la variation de température T°

Câblage sur plaque d'essai



Spécifications détaillées

- ◆ Température $< 15^{\circ}$ $\Rightarrow$ cycle de chauffe.
- ◆ Température $> 25^{\circ}$ $\Rightarrow$ ventilateur en proportion.
- ◆ Humidité $< 50\%$ $\Rightarrow$ cycle arrosage.
- ◆ Humidité $> 70\%$ $\Rightarrow$ le ventilateur en proportion.
- ◆ Potentiomètre T° varie de 0V à 5V pour 0° à 50°
- ◆ Potentiomètre $H\%$ varie de 0V à 5V pour 20% à 80%
- ◆ Après 500 arrosages le réservoir est vide, la LED verte clignote pour signaler l'alarme
- ◆ Un cycle comporte un temps actif puis un temps inactif
- ◆ Le RESET remet en conditions initiales

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) ?
 - ◆ Il y a 4 actions et 2 mesures => 6 PT ?
 - ◆ On aurait pu regrouper les mesures dans 1 seul PT, mais si on veut des cadences d'acquisition différentes on sépare.
 - ◆ Il y a 2 modes pour la LED Verte => 7 PT ?
- ◆ Quels sont leurs paramètres ?
 - ◆ Cadence d'acquisition pour T° et H%
 - ◆ Les temps actifs / inactifs pour Jaune & Bleu
 - ◆ La 1/2 période de clignotement de Verte

Constantes et variables globales

◆ Constantes

```
#define CMINT      15    // consigne basse
#define CMAXT      25    // Consigne haute
#define CMINH      50    // consigne basse
#define CMAXH      70    // Consigne haute
#define TMIN       0     // echelle basse
#define TMAX       50    // echelle haute
#define HMIN       20    // echelle basse
#define HMAX       80    // echelle haute
#define MAXCYCLE 500    // nb max de cycles arrosage
```

◆ Température et humidité : temp & humi

◆ Compteur nb_cycles et booléen alarme

```
#include <pt.h>
static struct pt ptT, ptH, ptV, ptC, ptE, ptN, ptA ; // proto threads
int temp=0; int humi=0;
Bool alarme = false ; int nb_cycles = 0 ;
```

Structure principale Arduino

◆ Setup

```
void setup() {  
    initIO();          // initialisations LEDs & potentiomètres  
    // initialisation des proto-threads  
    PT_INIT(&ptT); PT_INIT(&ptH); PT_INIT(&ptV); PT_INIT(&ptC);  
    PT_INIT(&ptE); PT_INIT(&ptN); PT_INIT(&ptA);  
    temp = acquis(AP_T, TMIN, TMAX); // acquis potard T°  
    humi = acquis(AP_H, HMIN, HMAX); // acquis potard H°  
}
```

◆ Loop

```
void loop() {  
    ptTemp(&ptT, 5);          // acquisition t°chaque 5sec  
    ptHumi(&ptH, 10);         // acquisition % chaque 10sec  
    ptVenti(&ptV);            // ventilation  
    ptChauffe(&ptC, 5, 10);    // chauffage 5s, inactivité 10s  
    ptEau(&ptE, 3, 40);       // arrosage 3s, inactivité 40s  
    ptNiveau(&ptN);           // niveau d'eau, PWM Verte  
    ptAlarme(&ptA, 500);      // réserve vide , clignote à 1Hz  
}
```

Les acquisitions

◆ Les proto-threads ptTemp et ptHumi :

```
static int ptTemp(struct pt *pt, int repit) {
static unsigned long ttl = 0;
    PT_BEGIN(pt);
    while(1) {
        ttl = millis() + (repit*1000) ;
        PT_WAIT_UNTIL(pt, millis() > ttl);
        temp = acquis(AP_T, TMIN, TMAX);
    }
    PT_END(pt);
}
```

```
static int ptHumi(struct pt *pt, int repit) {
static unsigned long ttl = 0;
    PT_BEGIN(pt);
    while(1) {
        ttl = millis() + (repit*1000) ;
        PT_WAIT_UNTIL(pt, millis() > ttl);
        humi = acquis(AP_H, HMIN, HMAX);
    }
    PT_END(pt);
}
```

Alarme clignotante

◆ Le proto-thread ptAlarme

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

Visualisation niveau d'eau

◆ Le proto-thread ptNiveau

```
static int ptNiveau(struct pt *pt)
{
    static int nb_old = 0; int ratio;
    nb_old = nb_cycles;
    PT_BEGIN(pt);
    while (!alarme) {
        PT_WAIT_UNTIL(pt, nb_cycles > nb_old ); // attente nouvel arrosage
        ratio = map(nb_cycles, 0, 100, 254, 0);
        ledPWM(VERTE, ratio);
        nb_old = nb_cycles;
    }
    PT_END(pt);
}
```

Gestion du chauffage

◆ Le proto-thread ptChauffe

```
static int ptChauffe(struct pt *pt,  int actif, int inactif )
{
static unsigned long ttl = 0;
BEGIN(pt);
    while ( 1) {
        PT_WAIT_UNTIL(pt,  temp < 15);
        digitalWrite(JAUNE, HIGH);
        ttl = millis() + actif ;
        PT_WAIT_UNTIL (pt, millis() > ttl);
        digitalWrite(JAUNE, LOW);
        ttl = millis() + inactif ;
        PT_WAIT_UNTIL (pt, millis() > ttl);
    }
    PT_END(pt);
} // Chauffe
```

Gestion de l'arrosage

◆ Le proto-thread ptEau

```
static int ptEau(struct pt *pt, int actif, int inactif ) {  
    static unsigned long ttl = 0;  
    PT_BEGIN(pt);  
    while (!alarme) {  
        PT_WAIT_UNTIL(pt, humi < CMINH);  
        digitalWrite(BLEU, HIGH);  
        ttl = millis() + (actif*1000) ;  
        PT_WAIT_UNTIL (pt, millis() > ttl);  
        digitalWrite(BLEU, LOW);  
        nb_cycles++;  
        alarme = ( nb_cycles > MAXCYCLE);  
        ttl = millis() + (inactif*1000) ;  
        PT_WAIT_UNTIL (pt, millis() > ttl);  
    }  
    PT_END(pt);  
} // Eau
```

Gestion de la ventilation

◆ Le proto-thread ptVenti

```
static int ptVenti(struct pt *pt) {  
    static int temp_old = 0, humi_old = 0;  
    PT_BEGIN(pt);  
    while (1) {  
        PT_WAIT_UNTIL(pt, temp > CMAXT || humi > CMAXH);  
        temp_old = temp;  
        humi_old = humi ;  
        ledPWM(ROUGE, vRatio()) ;  
        PT_WAIT_UNTIL(pt, temp != temp_old || humi != humi_old );  
        ledPWM(ROUGE, vRatio()) ;  
    }  
    PT_END(pt);  
} // Ventilateur
```

That's all Folks!



Merci pour votre attention :-)