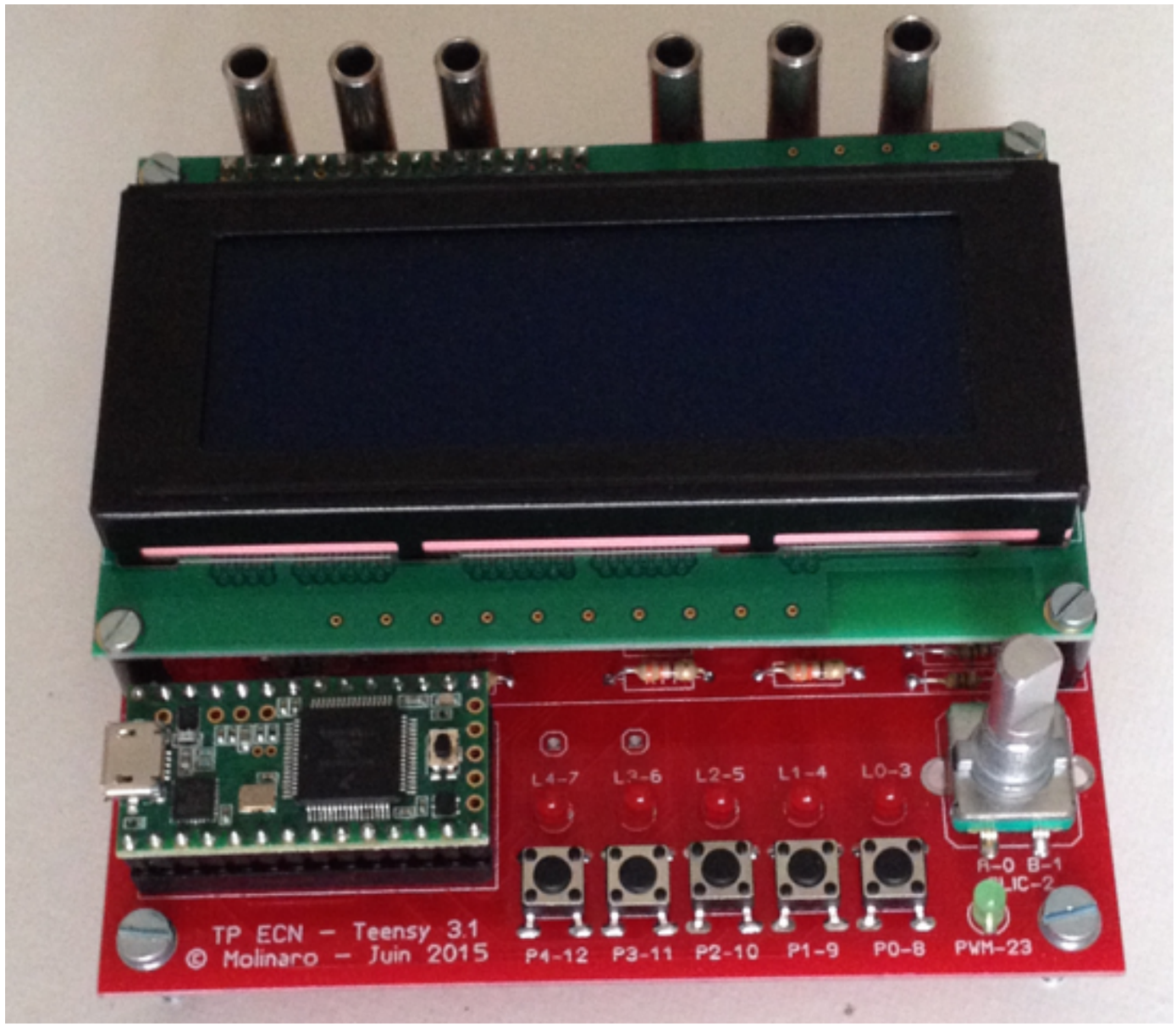
Temps Réel


L4-7
L3-6
L2-5
L1-4
L0-3
R-0 B-1
LIC-2
TP ECN - Teensy 3.1
© Molinaro - Juin 2015
P4-12
P3-11
P2-10
P1-9
P0-8
PWM-23

But de cette partie

Objectif :

- *mettre en évidence la non atomicité de l'incrémentation d'une variable ; utiliser les constructions permettant l'incrémentation atomique.*

Problèmes à résoudre :

- permettre à l'utilisateur d'installer une routine d'interruption périodique ;
- non atomicité de l'incrémentation d'une variable.

Travail à faire :

Réaliser un programme qui incrémente quatre variables globales :

- dans une routine d'interruption périodique ;
- dans la routine `loop`.

Au bout de 5 secondes, la routine d'interruption périodique est désinstallée et les quatre variables sont affichées.

Routine d'interruption périodique

On veut que l'utilisateur puisse installer une routine d'interruption qui sera exécutée toutes les milli-secondes.

Installer une routine d'interruption périodique

startup-sequential.c

```
//-------------------------------------------------------------------*

static void (* gRealTimeISR) (void) ;

//-------------------------------------------------------------------*

void setRealTimeISR (void (* inISR) (void)) {
  gRealTimeISR = inISR ;
}

//-------------------------------------------------------------------*

static void systickHandler (void) {
  gUpTimeInMilliseconds ++ ;
  if (NULL != gRealTimeISR) {
    gRealTimeISR () ;
  }
}
```

Modifier le fichier `startup-sequential.c` :

- insérer la déclaration de la variable `gRealTimeISR` ;
- insérer la routine `setRealTimeISR` ;
- modifier la routine `systickHandler`.

Utilisation de la routine `setRealTimeISR`

Installer une routine d'interruption périodique.

```
void isr (void) {
  // Code exécuté toutes les milli-secondes
}
void setup (void) {
  …
  setRealTimeISR (isr) ;
}
```

Désinstaller une routine d'interruption périodique.

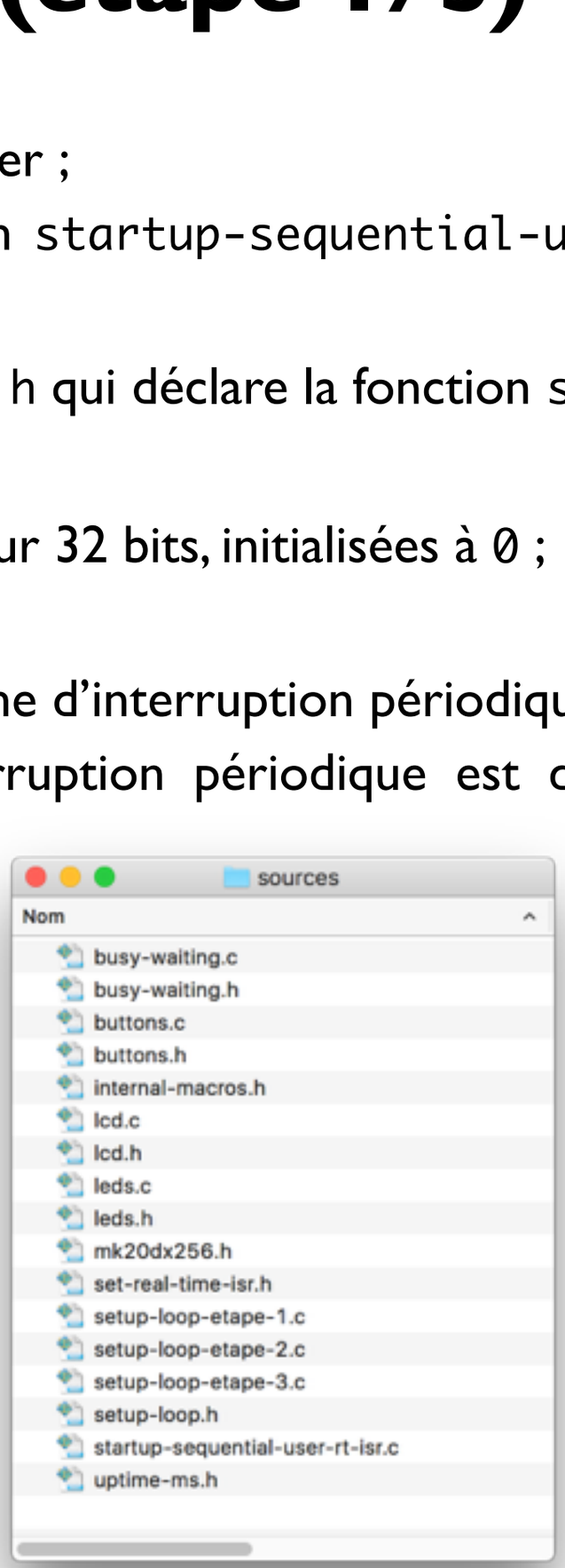
```
  setRealTimeISR (NULL) ;
```

À écrire dans la routine `loop` (ou `setup`), pour que la routine précédemment installée ne soit plus appelée.

Travail à faire (étape 1/3)

Première étape :

- dupliquer le programme précédent et le renommer ;
- renommer le fichier `startup-sequential.c` en `startup-sequential-user-it-isr.c`, et le modifier comme indiqué ;
- ajouter un fichier d'en-tête `set-real-time-isr.h` qui déclare la fonction `setRealTimeISR` ;
- écrire dans `setup-loop.c` un programme qui :
  - déclare 4 variables globales de type non signé sur 32 bits, initialisées à `0` ;
  - installe une routine d'interruption périodique ;
  - ces 4 variables sont incrémentées dans la routine d'interruption périodique ;
  - au bout de cinq secondes, la routine d'interruption périodique est désinstallée, et les valeurs des quatre variables sont affichées.

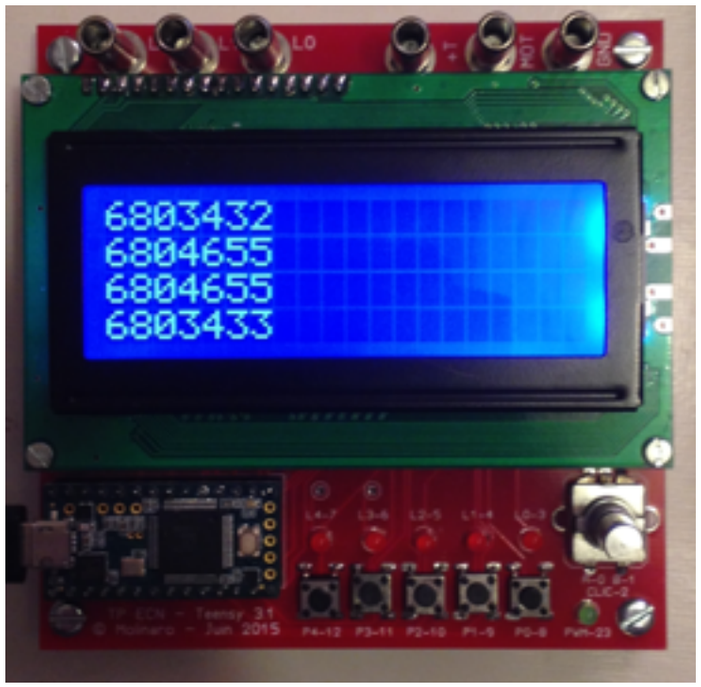


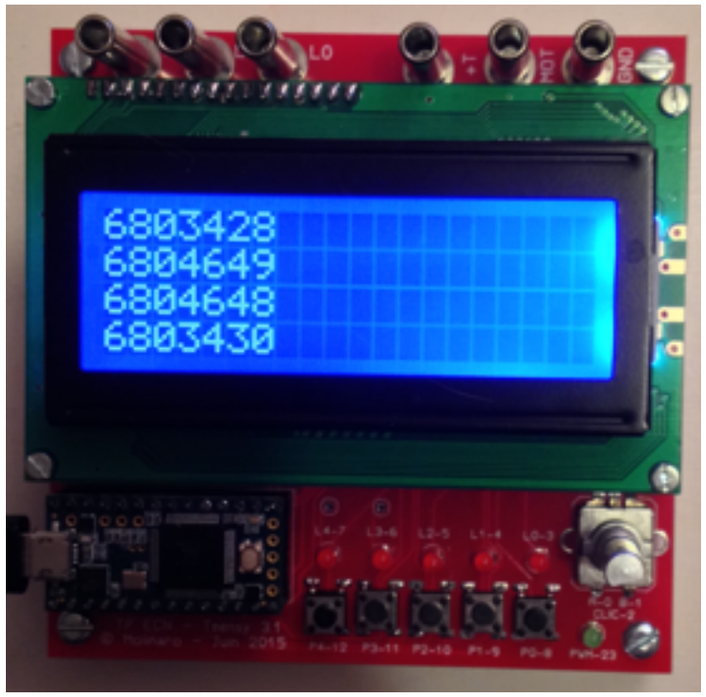
Travail à faire (étape 2/3)

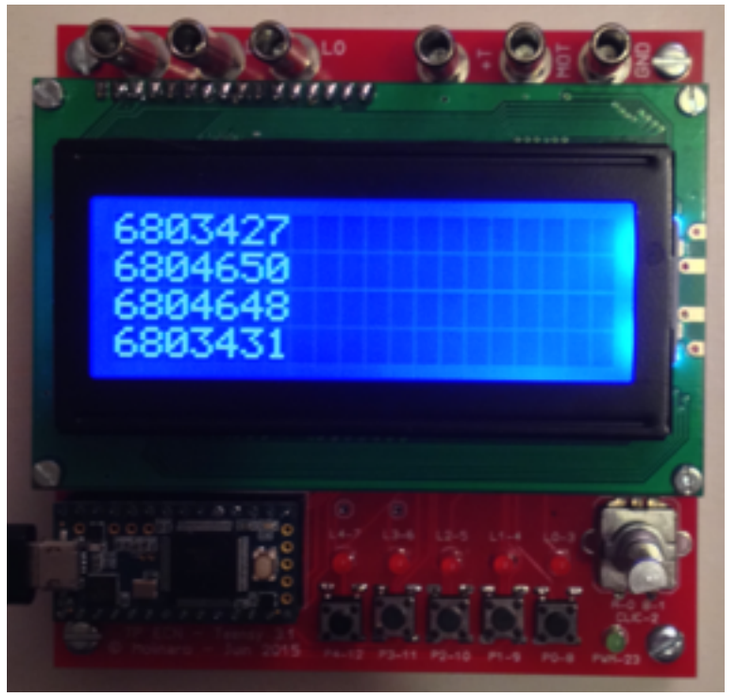
Deuxième étape :

- ajouter l'incrémentation des quatre variables au début de la routine `loop` ;

Observer les valeurs affichées. Pourquoi sont-elles différentes ?







Travail à faire (étape 3/3)

Troisième étape :

- dans la routine `loop`, effectuer l'incrémentation atomique des quatre variables.

GCC possède des fonctions intrinsèques qui réalisent l'incrémentation atomique.

L'incrémentation est habituellement écrite :

```
variable ++ ;
```

L'incrémentation atomique s'écrit :

```
__atomic_fetch_add (& variable, 1, __ATOMIC_ACQ_REL) ;
```