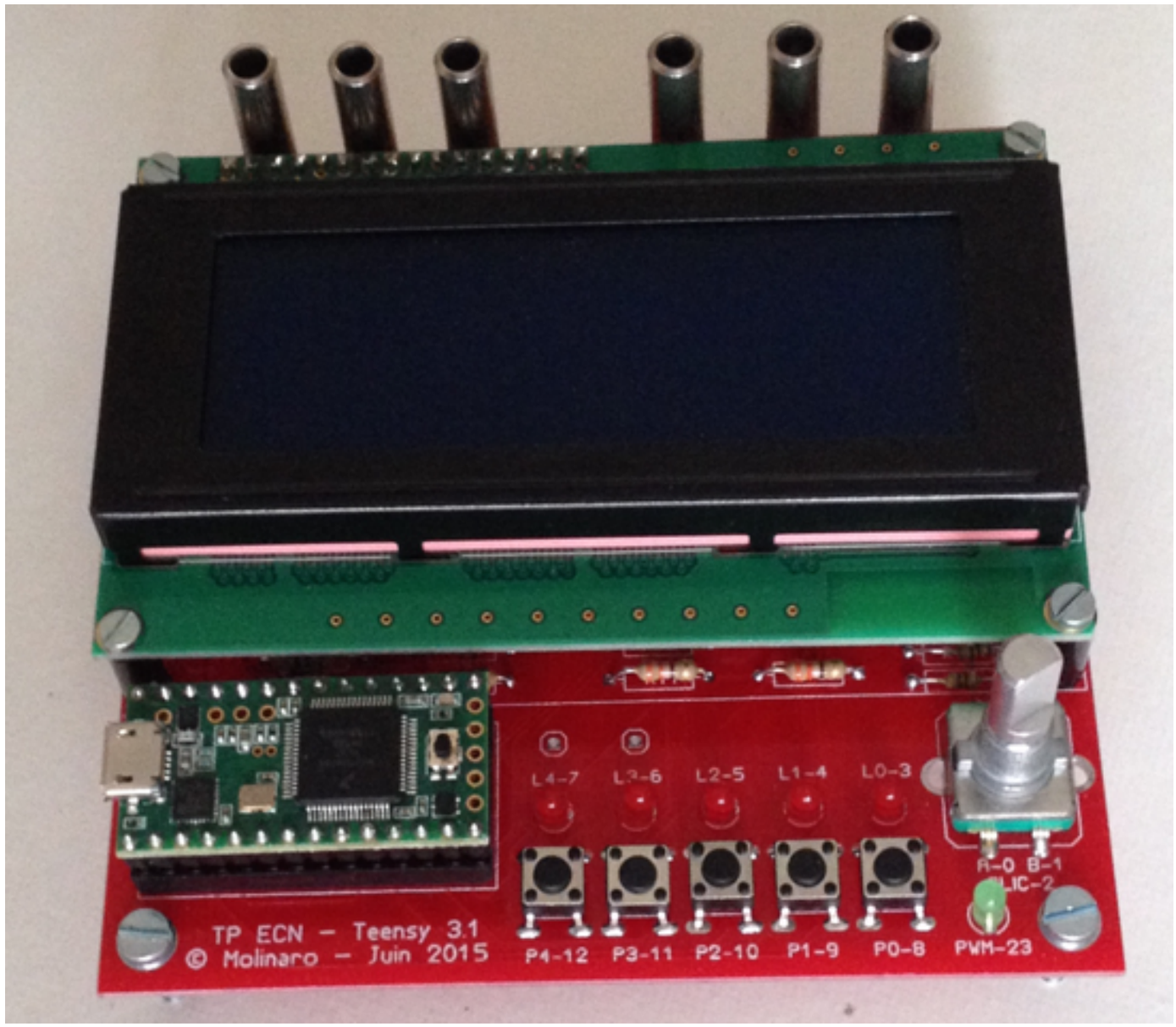
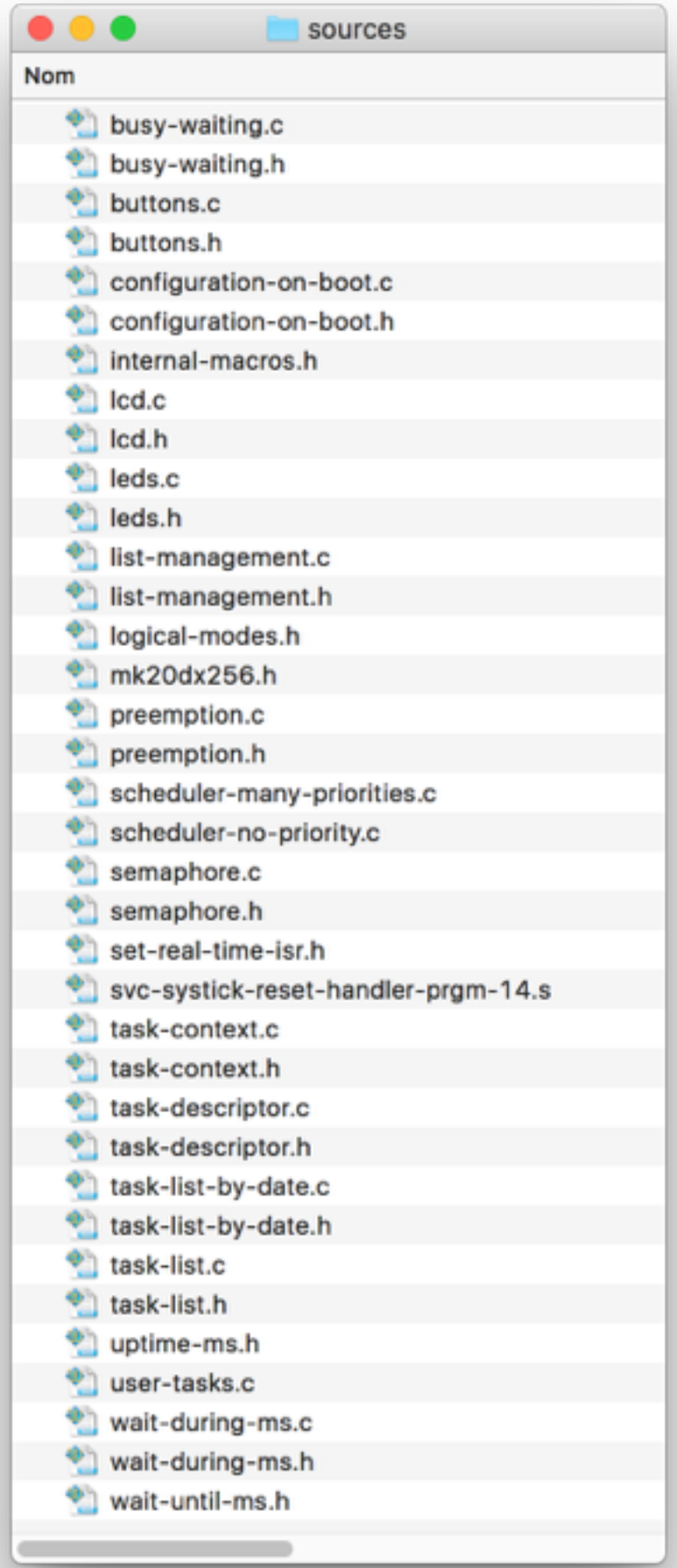
Temps Réel


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 :

- *ajout de l'attente sur échéance au sémaphore de Dijkstra.*

Travail à faire :

Ajouter au sémaphore l'attente sur échéance, et imaginer un jeu de tâches qui met en évidence ce type d'attente.

Primitive `P_until` d'un sémaphore

```
bool P_until (USER_MODE_
              Semaphore * inSemaphore,
              const uint32_t inDeadline) ;

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

bool svc_P_until (KERNEL_MODE_
                  Semaphore * inSemaphore,
                  const uint32_t inDeadline) {
  const bool result = inSemaphore->mValue > 0 ;
  if (result) {
    inSemaphore->mValue -- ;
  }else if (inDeadline > uptimeMS ()) {
    kernel_runningTaskWaitsOnFIFOListAndListOrderedByDate (MODE_
                                                           & inSemaphore->mWaitingTaskList,
                                                           inDeadline) ;
  }
  return result ;
}
```

La primitive `P_until` effectue une attente sur sémaphore jusqu'à une certaine date. La valeur retournée est un booléen :

- `true` si le sémaphore a permis le passage ;
- `false` si l'échéance a été atteinte.

Comment fonctionne la primitive `P_until` (1/4)

Quand la primitive `P_until` est appelée :

- soit `mValue` est nul, soit il est strictement positif (dans le premier cas, si la primitive P était appelée à la place de `P_until`, la tâche appelante se bloquerait ; dans le second, `mValue` serait décrémenté) ;
- soit l'échéance spécifiée par `inDeadline` est atteinte, soit elle n'est pas atteinte.

Ces deux possibilités se combinant, il y a quatre situations possibles.

Comment fonctionne la primitive `P_until` (2/4)

```
bool svc_P_until (KERNEL_MODE_
                  Semaphore * inSemaphore,
                  const uint32_t inDeadline) {
  const bool result = inSemaphore->mValue > 0 ;
  if (result) {
    inSemaphore->mValue -- ;
  }else if (inDeadline > uptimeMS ()) {
    kernel_runningTaskWaitsOnFIFOListAndListOrderedByDate (MODE_
                                                           & inSemaphore->mWaitingTaskList,
                                                           inDeadline) ;
  }
  return result ;
}
```

Si `mValue` est strictement positif, il est décrémenté, et la primitive retourne immédiatement en renvoyant la valeur `true` ; l'échéance n'est pas comparée avec la date courante [2 situations prises en compte sur les 4].

Si `mValue` est nul et si l'échéance est atteinte (c'est-à-dire les deux tests sont faux), la primitive retourne immédiatement en renvoyant la valeur `false` ; l'échéance n'est pas comparée avec la date courante [3e situation prise en compte sur les 4].

Comment fonctionne la primitive `P_until` (3/4)

Si `mValue` est nul et si l'échéance n'est pas atteinte, alors la fonction `kernel_runningTaskWaitsOnFIFO-ListAndListOrderedByDate` est appelée, et la primitive ne retourne pas [4e situation prise en compte]. La tâche est donc bloquée. Son déblocage intervient dans deux cas :

- soit la primitive `V` du sémaphore est appelée ;
- soit l'échéance est atteinte.

Comment fonctionne la primitive `P_until` (4/4)

Déblocage par appel de la primitive V du sémaphore. Rappelons le code de cette primitive :

```
void svc_V (IRQ_MODE_ Semaphore * inSemaphore) {
  const bool found = kernel_firstWaitingTaskBecomesReady (MODE_ & inSemaphore->mWaitingTaskList, 1) ;
  if (! found) {
    inSemaphore->mValue ++ ;
  }
}
```

La tâche est rendue prête par l'appel de `kernel_firstWaitingTaskBecomesReady`. Remarquer le dernier argument : `1`. C'est cette valeur qui sera la valeur de retour de la primitive `P_until`. La valeur entière `1` sera vue comme la valeur booléenne correspondante, c'est-à-dire `true`.

Déblocage par l'échéance. Rappelons le code de la routine d'interruption `systickHandler` :

```
void systickHandler (IRQ_MODE) {
  gUpTimeInMilliseconds ++ ;
  kernel_tasksWithEarlierDateBecomeReady (MODE_ uptimeMS (), 0) ;
  if (NULL != gRealTimeISR) {
    gRealTimeISR (MODE) ;
  }
}
```

Les tâches dont l'échéance est atteinte sont rendues prêtes par l'appel de `kernel_tasksWithEarlierDateBecomeReady`. Remarquer le dernier argument : `0`. C'est cette valeur qui sera la valeur de retour de la primitive `P_until`. La valeur entière `0` sera vue comme la valeur booléenne correspondante, c'est-à-dire `false`.