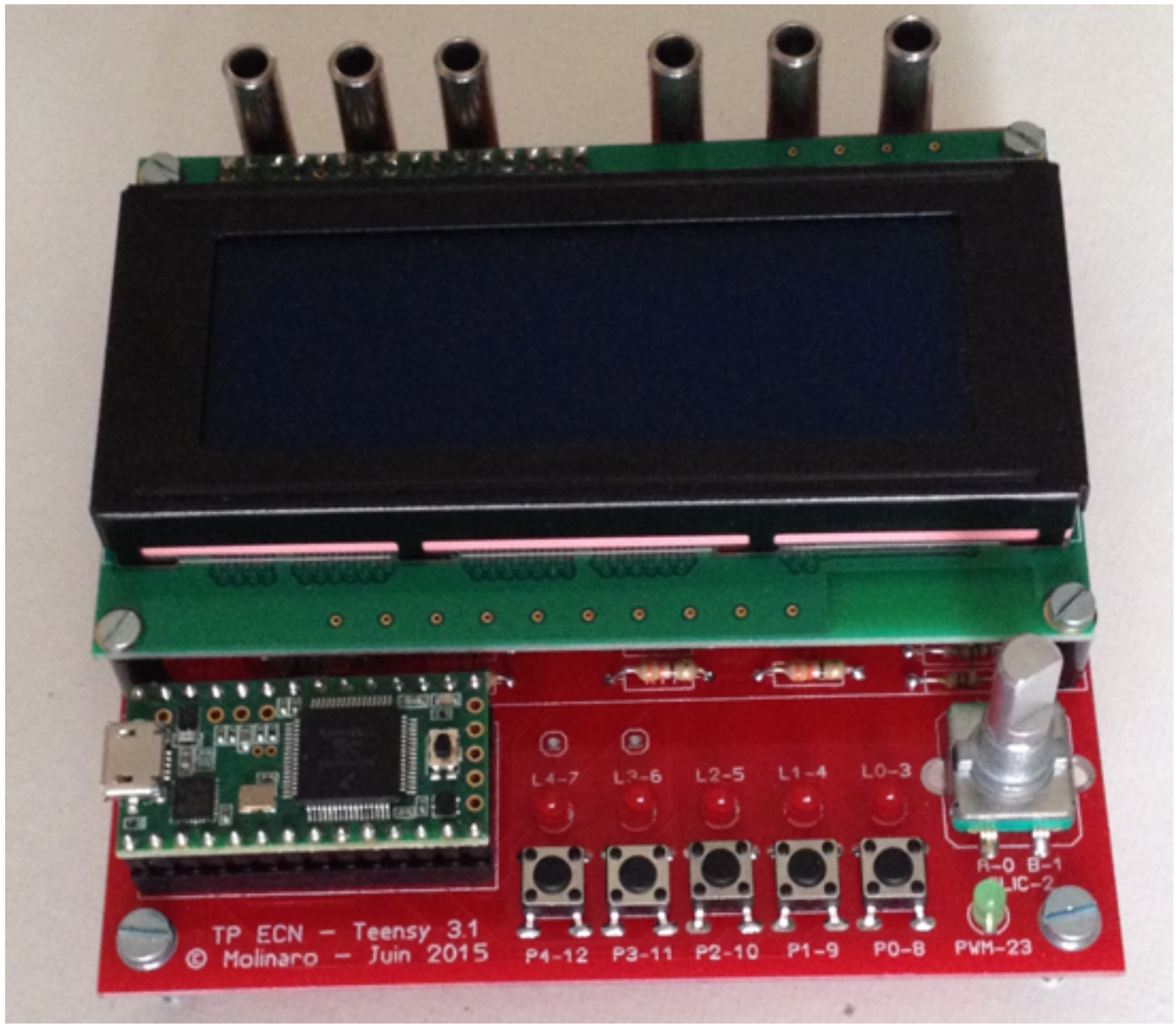
Temps Réel


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 :

- *disposer de routines permettant de tester les boutons poussoir P0 à P4.*

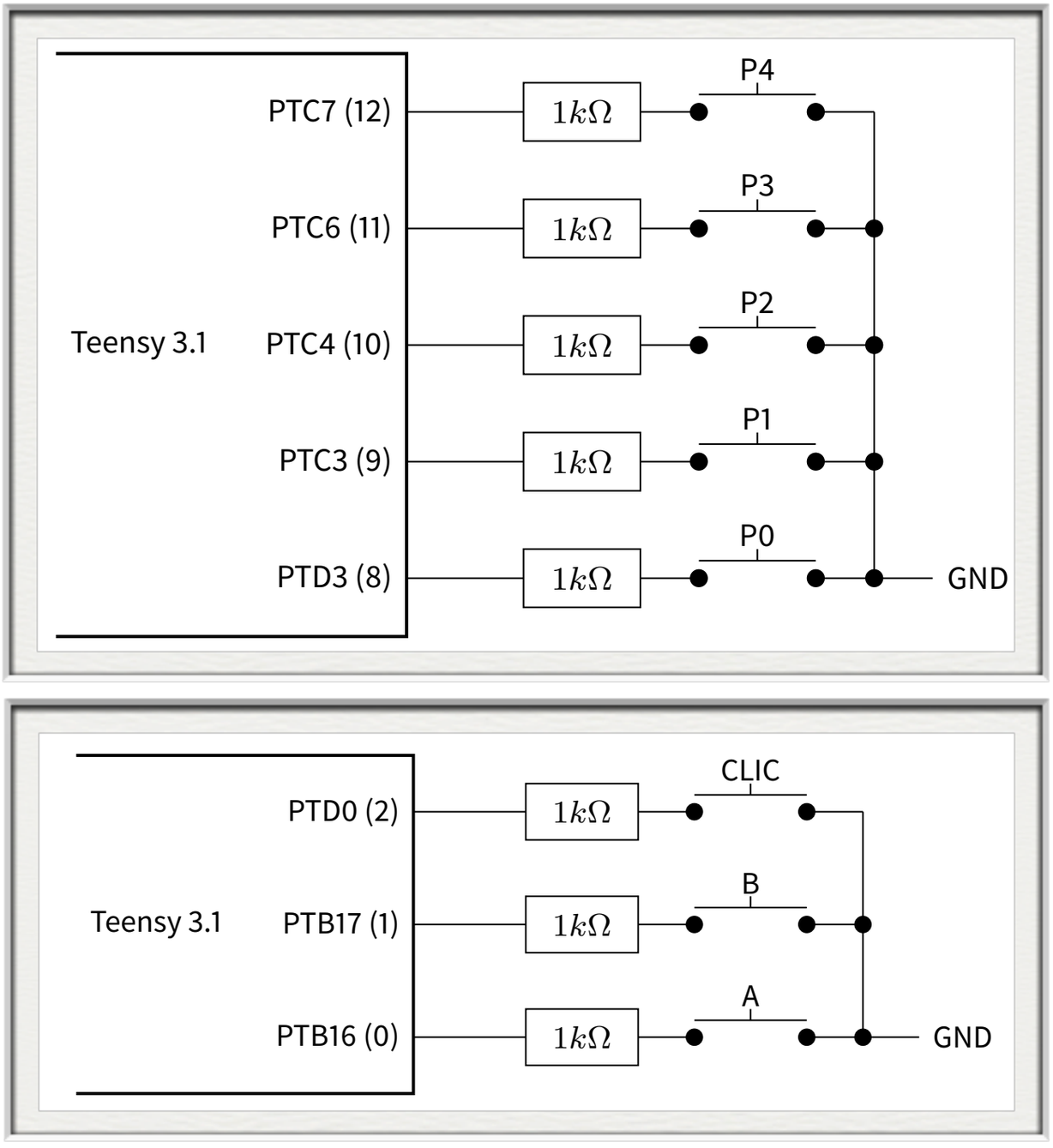
Problèmes à résoudre :

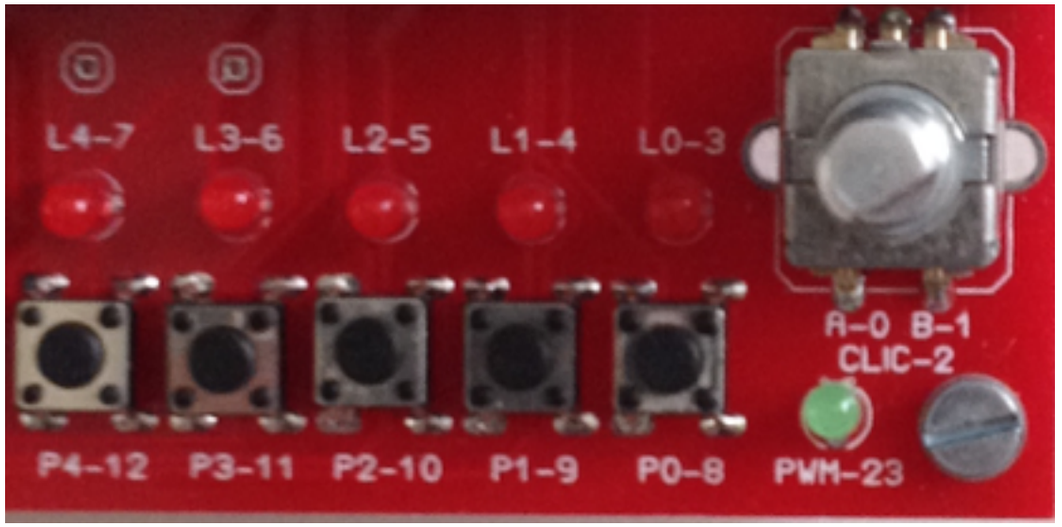
- configuration des ports du micro-contrôleur ;
- écrire la routine `buttons`.

Travail à faire :

- réaliser un programme qui allume ou éteint une led, suivant que le poussoir correspondant est appuyé ou relâché.

Connexion des poussoirs et de l'encodeur





Bouton appuyé : le port correspondant est à une tension de `0V`, sa lecture par programme retourne un booléen **`false`**.

Bouton relâché : une résistance interne *pullup* place le port correspondant à une tension proche de `3,3V`, sa lecture par programme retourne un booléen **`true`**.

Sous *Arduino*, ces ports doivent être programmés en entrée, en utilisant `INPUT_PULLUP`.

Initialisation : routine setupButtons

buttons.c

```
static void setupButtons (void) {
//--- P0
  PORTD_PCR3 = (1 << 8) | (1 << 1) | (1 << 0) ; // Port is GPIO, has a internal pull, pull is pull-up
//--- P1
  PORTC_PCR3 = (1 << 8) | (1 << 1) | (1 << 0) ; // Port is GPIO, has a internal pull, pull is pull-up
//--- P2
  PORTC_PCR4 = (1 << 8) | (1 << 1) | (1 << 0) ; // Port is GPIO, has a internal pull, pull is pull-up
//--- P3
  PORTC_PCR6 = (1 << 8) | (1 << 1) | (1 << 0) ; // Port is GPIO, has a internal pull, pull is pull-up
//--- P4
  PORTC_PCR7 = (1 << 8) | (1 << 1) | (1 << 0) ; // Port is GPIO, has a internal pull, pull is pull-up
//--- CLIC
  PORTD_PCR0 = (1 << 8) | (1 << 1) | (1 << 0) ; // Port is GPIO, has a internal pull, pull is pull-up
}

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

MACRO_INIT_ROUTINE (setupButtons) ;
```

La routine `setupButtons` programme les ports correspondant aux cinq boutons poussoirs `P0`, `P1`, …, `P4` en entrée, ainsi que le port correspondant au poussoir de l'encodeur numérique (« `CLIC` »). Elle est automatiquement exécutée lors du démarrage du micro-contrôleur.

Désignation des poussoirs

buttons.h

```
static const uint32_t BUTTON_P0    = 1 << 0 ;
static const uint32_t BUTTON_P1    = 1 << 1 ;
static const uint32_t BUTTON_P2    = 1 << 2 ;
static const uint32_t BUTTON_P3    = 1 << 3 ;
static const uint32_t BUTTON_P4    = 1 << 4 ;
static const uint32_t ENCODER_CLIC = 1 << 5 ;
```

Les cinq symboles `BUTTON_L0`, … correspondent aux cinq poussoirs, et `ENCODER_CLIC` au poussoir de l'encodeur.

Tester les poussoirs : fonction buttons

buttons.c

```
uint32_t buttons (void) {
  uint32_t result = 0 ;
//--- P0
  if ((GPIOD_PDIR & (1 << 3)) == 0) {
    result |= BUTTON_P0 ;
  }
//--- P1
  if ((GPIOC_PDIR & (1 << 3)) == 0) {
    result |= BUTTON_P1 ;
  }
//--- P2
  if ((GPIOC_PDIR & (1 << 4)) == 0) {
    result |= BUTTON_P2 ;
  }
//--- P3
  if ((GPIOC_PDIR & (1 << 6)) == 0) {
    result |= BUTTON_P3 ;
  }
//--- P4
  if ((GPIOC_PDIR & (1 << 7)) == 0) {
    result |= BUTTON_P4 ;
  }
//--- CLIC
  if ((GPIOD_PDIR & (1 << 0)) == 0) {
    result |= ENCODER_CLIC ;
  }
  return result ;
}
```

Cette fonction permet d'acquérir simultanément l'état de tous les poussoirs. Elle retourne un entier non signé de 32 bits (`uint32_t`) où :

- les bits 6 à 31 sont à `0` ;
- le bit `5` est à `1` si le poussoir de l'encodeur est appuyé, à `0` si il est relâché ;
- le bit `4` est à `1` si le poussoir `P4` est appuyé, à `0` si il est relâché ;
- le bit `3` est à `1` si le poussoir `P3` est appuyé, à `0` si il est relâché ;
- le bit `2` est à `1` si le poussoir `P2` est appuyé, à `0` si il est relâché ;
- le bit `1` est à `1` si le poussoir `P1` est appuyé, à `0` si il est relâché ;
- le bit `0` est à `1` si le poussoir `P0` est appuyé, à `0` si il est relâché.

Exemples d'utilisation de la fonction `buttons`

Tester si le poussoir P0 est appuyé.

```
if ((buttons () & BUTTON_P0) != 0) {
    // Poussoir appuyé
  }else{
    // Poussoir relâché
  }
```

Tester si un des deux poussoirs P1 et P3 au moins est appuyé.

```
if ((buttons () & (BUTTON_P1 | BUTTON_P3) != 0) {
    // Un des deux poussoirs est appuyé
  }else{
    // Les deux poussoirs sont relâchés
  }
```

Tester si les poussoirs P1 et P3 sont simultanément appuyés.

```
if ((buttons () & (BUTTON_P1 | BUTTON_P3) == (BUTTON_P1 | BUTTON_P3)) {
    // Les deux poussoirs sont appuyés
  }else{
    // Au moins un des deux poussoirs est relâché
  }
```

Travail à faire

Écrire un programme qui :

- allume la led `L4` si le poussoir `P4` est appuyé, et l'éteint si il est relâché ;
- allume la led `L3` si le poussoir `P3` est appuyé, et l'éteint si il est relâché ;
- allume la led `L2` si le poussoir `P2` est appuyé, et l'éteint si il est relâché ;
- allume la led `L1` si le poussoir `P1` est appuyé, et l'éteint si il est relâché ;
- allume la led `L0` si le poussoir `P0` est appuyé, et l'éteint si il est relâché ;
- allume la led *Teensy* si le poussoir de l'encodeur est appuyé, et l'éteint si il est relâché ;

Pour cela :

- dupliquer le programme précédent et le renommer ;
- récupérer les fichiers `buttons.h` et `buttons.c` sur le serveur pédagogique, archive `05-sources.tbz`.