# Phare maritime à éclat programmable

Institut Gériatrique des Solex
Bernique

## Abstract

Construction d'un phare maritime miniature en impression 3D avec sa lanterne à éclat programmable.

**Keywords**: ATtiny85, led, impression 3D, iTopie, sleep mode, WDT

---

## Table des matières



---

# I. Introduction

Profitant de l'occasion offerte par l'anniversaire d'un enfant, j'avais réalisé une miniature d'un phare maritime, sur la base d'un corps en carton et d'une led qui clignotait selon un schéma prédéterminé, le tout piloté par un petit µ-processeur.
Bref, le truc qui ne sert à rien mais que j'ai remis au goût du jour en gardant l'électronique et en revisitant la structure du phare désormais imprimée en 3D sur mon iTopie maison.

# II. Composants et dimensionnement

## 1 Le cerveau du système

Bien connu des bidouilleurs de tous poils, le µ-processeur ATtiny85 de chez Atmel® est d'un accès très aisé. Il est disponible partout en version nue DIP-8.


Il bénéficie d'une immense communauté pour le support, est directement programmable en C et donne ainsi accès à toute sa puissance sans compromis sur l'efficacité dès lors que l'on se donne la peine de maîtriser ses registres intimes.
Il peut aussi être programmé en langage Arduino à l'aide de bibliothèques dédiées et restreintes adaptées à sa petite mémoire… il faut alors accepter de perdre un peu du contrôle strict de son comportement dicté par le contenu figé de ces mêmes bibliothèques !
J'ai choisi ici sa déclinaison ATtiny85-20PU pour ses dimensions restreintes et un nombre d'entrées/sorties largement suffisantes. On pourrait opter pour la version 10PU qui présente l'avantage de fonctionner à tension plus faible (jusqu'à 1,8v), plutôt intéressante dans des applications où les réserves d'énergie ne sont pas infinies!

Le programme à charger en mémoire est donné en annexe (les conditions de transfert sont données en en-tête du programme).

## 2 Alimentation

L'AT85 support une alimentation directe par piles ou accus tant qu'on ne sort pas de ses spécifications.
J'ai opté pour deux piles AA qui délivrent un bon 3,6V lorsque toutes neuves. Le reste du dimensionnement sera basé sur cette tension. Un boîtier chinois avec interrupteur incorporé fera des merveilles pour quelques centimes de piastres !

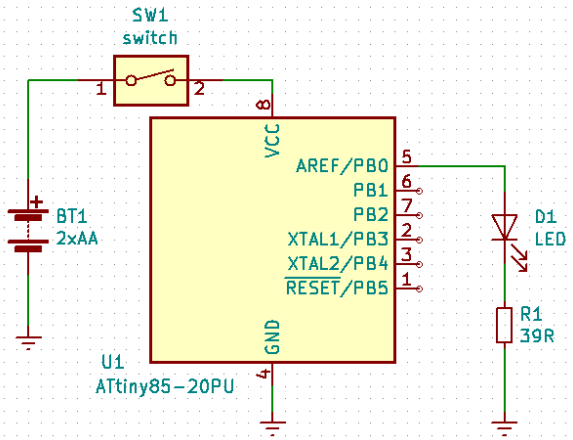
## 3 Lanterne du phare

La lanterne du phare est simulée par une led blanche (tension directe environ 3v). On lui adjoindra une résistance de protection 39Ω.
L'ensemble sera connecté sur la broche PB0 de l'AT85.

Plus simple, tu meurs...

# III. Le circuit

Plus simple, ça va être compliqué… un bon schéma valant tous les commentaires !

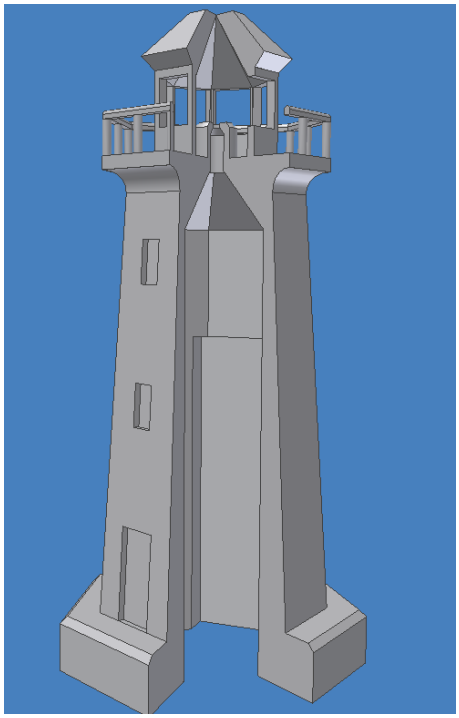


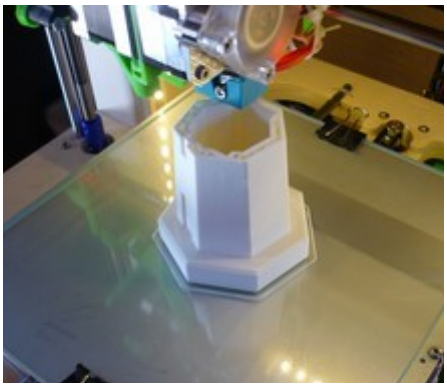
# IV. Mise en œuvre

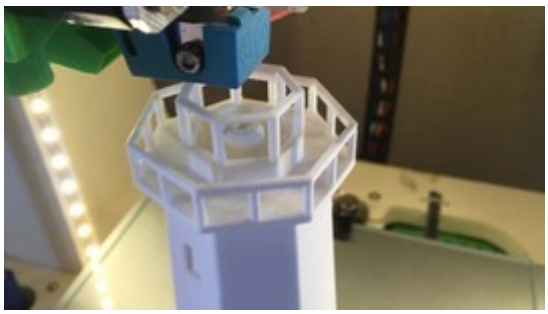
Après avoir longtemps cherché un modèle libre de droits sur le Net, j'ai finalement choisi de dessiner mon propre phare !

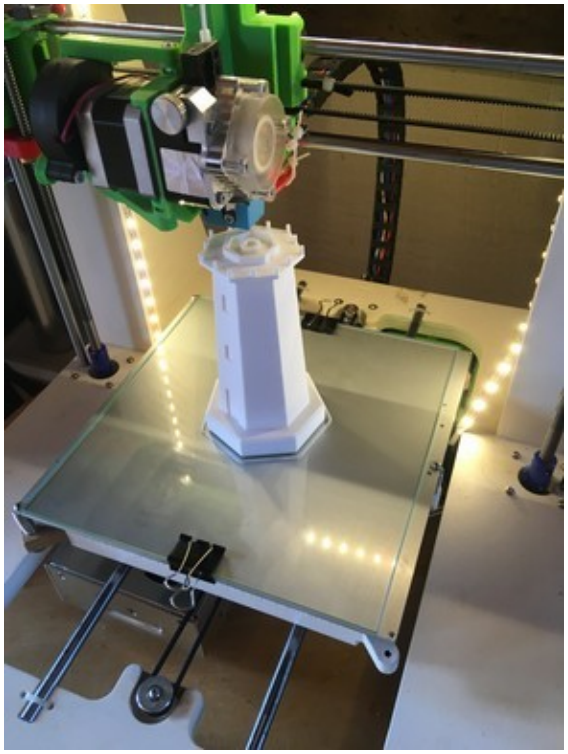
En effet, tous les modèles avec un rendu sympa étaient dessinés dans des proportions telles qu'il fallait les réduire à environ 30 % de leur taille nominale dans le logiciel de tranchage, créant des problèmes de continuité de surfaces. Tant qu'à tout refaire, j'ai créé un modèle qui ne nécessite pas de support, et qui peut être imprimé en un seul morceau. J'y ai aussi logé le boîtier chinois pour les piles. Enfin, pour le fun, j'ai opté pour une base heptagonale ;-)

Au final, j'ai obtenu après 4 heures d'impression un petit phare d'une quinzaine de centimètres de haut et dont la base mesure environ 7 cm.



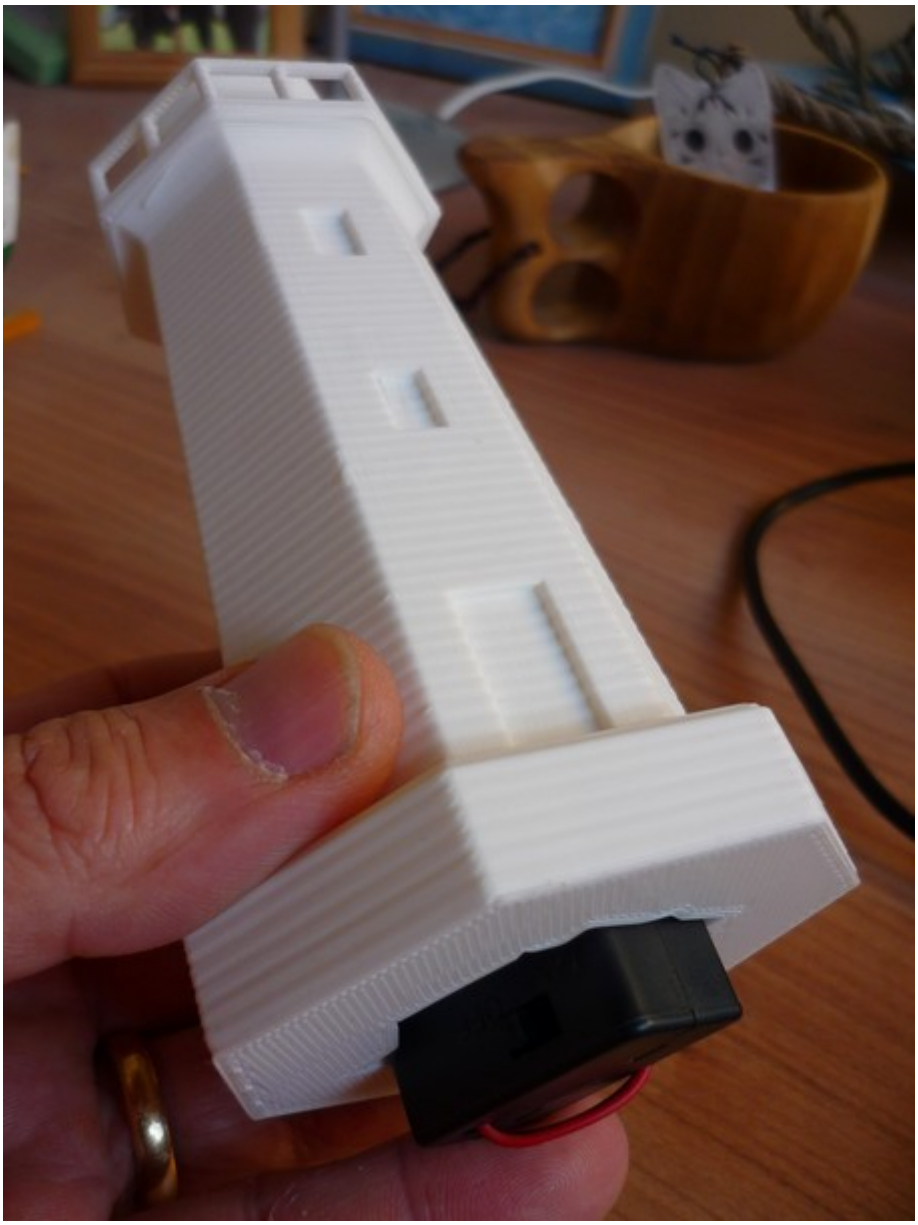






Ensuite, j'ai glissé le support des piles et l'AT85 avec sa résistance et la led à l'intérieur de la colonne. L'interrupteur sur le boîtier permet de mettre le phare facilement en fonctionnement.

Le code, chargé sur le µ-processeur, endort le µ-processeur entre chaque changement d'état de la led, permettant d'économiser les piles (*power down* sleep mode et le watchdog timer )
J'ai choisi la séquence d'allumage de la led suivante : un éclat long suivi de 3 éclats courts (voir code). C'est évidemment modifiable à votre guise en jouant avec le paramètre de la fonction OFF()  ;-)



## V. Conclusion

Woulà… un p'tit montage à quatre composants pour un résultat sympa… yapluka se lancer dans la confection des bateaux en papier pour décorer autour :-)
A vous de jouer !

---

# Annexe : le code

```
/*****************************************************************************
*  Blink a led to a defined patern using power_down sleep mode & WDT         *
*                                                                            *
*  Processor: ATTiny85 @ 1MHz                                                *
*  Compiler avr-gcc 5.4.0                                                    *
*  Programmer: USBasp                                                        *
*                                                                            *
*                  PB5  *|+      |*  VCC                                    *
*                  PB3  *|       |*  PB2                                    *
*                  PB4  *|       |*  PB1                                    *
*                  GND *|___|*  PB0  --> LED1                               *
*                                                                            *
* WatchDog Timer:          wake up from Power Down sleep mode                *
*                                                                            *
* Lfuse: 0x62 / Hfuse: 0xDF / Efuse: 0xFF                                    *
*                                                                            *
* Bernique BGY#012                                                           *
* Common Creative BY-NC-SA                                                   *
* 2018/03/03                                                                 *
*                                                                            *
* Status: OK                                                                 *
*****************************************************************************/

#define F_CPU 1000000L    // 1Mhz (8Mhz, prescaled by fuse = 8), à changer lorsqu'à 125kHz

#include <avr/io.h>
#include <avr/wdt.h>
#include <avr/sleep.h>
#include <avr/interrupt.h>

#define LED_1  PB0          // LED 1 on pin5

volatile uint8_t wdt_count; // counter for Watchdog
int8_t _prescaler;
uint8_t i = 0;              // internal counter

void main(void) {
   // setup
   // I/O configuration
   DDRB |= (1<<LED_1);                          // Pin configuration as output
   // switch Analog Comparator OFF
   ACSR  |= (1<<ACD);                           // switch Analog Comparator OFF to save power
   // switch ADC OFF
   ADCSRA &= ~(1<<ADEN);                        // switch ADC OFF
   // Configure attiny85 sleep mode
   set_sleep_mode(SLEEP_MODE_PWR_DOWN);
   // make interrupt active
   sei();
   // end of setup

   while(1) {                                   // main loop
            PORTB |= (1<<LED_1);                // turns LEDs ON
            OFF(8);
            PORTB &=~(1<<LED_1);                // turns LEDs OFF
            OFF(5);
            for (i = 0; i < 3; i++) {
                     PORTB |= (1<<LED_1);       // turns LEDs ON
                     OFF(1);
                     PORTB &=~(1<<LED_1);       // turns LEDs OFF
                     OFF(5);
            }
            OFF(11);
```

```
    }
}   //end of main

void OFF(uint16_t duree){               // turns devices for "duree" x 1 seconds
    wdt_count = 0;                      // usual delay function is replaced by putting processor into sleep power down mode
                                        // for "D" milli-second using watchdog
    watchdog_start_interrupt(3);        // prescale of 3 ~= 0.125sec
    while(wdt_count < duree){           // Wait "duree" watchdog interupts
            sleep_mode();               // Make CPU sleep until next WDT interrupt
    }
    watchdog_stop();                    // end of watchdog

} // end of OFF

void watchdog_stop() {                  // Turn OFF Watchdog
    WDTCR |= (1<<WDCE) | (1<<WDE);
    WDTCR = 0x00;
}

void watchdog_start_interrupt(uint8_t wd_prescaler) {           // Turn ON Watchdog with Interrupt
    if(wd_prescaler > 9) wd_prescaler = 9;
    //byte _prescaler = wd_prescaler & 0x7;
    _prescaler = wd_prescaler & 0x7;
    if (wd_prescaler > 7 ) _prescaler |= (1<<WDP3);             // ^ fourth bit of the prescaler is somewhere else in the register...
    WDTCR = _prescaler;                                         // set new watchdog timer prescaler valuee
    WDTCR |= (1<<WDIE) | (1<<WDCE) | (1<<WDE);                  // start timed sequence
}

ISR(WDT_vect) {                         // Watchdog Interrupt Service / is executed when watchdog timed out
    wdt_count++;
    WDTCR |= (1<<WDIE);     // Watchdog goes to interrupt not reset
}
```