POLYTECH LILLE

Rapport du Projet

BIJOU ELECTRIQUE

Binômes : Lijie YAO et Keren QIANG

14/05/2018

REMERCIEMENTS

Avant toute chose, nous voudrons ensuite remercier monsieur Xavier Redon et monsieur Alexandre Boé, nous tuteurs de projet qui nous aident beaucoup dans ce projet. Ils nous ont donné beaucoup de conseils pour ce projet ainsi qu’ils ont fait beaucoup de choses pour nous d'avancer ce projet.

Sommaire

Remerciments……2
Introduction……..5
Analyse du projet
Analyse du concurent
1. Bague lumineuse à LED clignotante……6
2. La robe LED……7
3. Collier en or LED | L'or de la baronne……..7
I. L'introduction
I.1 Le microcontrôleur
I.1.1 Introduction d'Atmega328p……..9
I.1.2 l'image d'Atmega328p……10
I.1.3 le circuit appliqué……11
I.2 Le contrôleur
I.2.1 introduction de TLC5947……11
I.2.2 le schéma……..12
I.2.3 le circuit appliqué……12
I.2.4 librairie……..13
I.3 Les LEDs
I.3.1 introduction de LED orange……13
I.3.2 introduction de LED bleu……14
I.3.3 introduction de LED vert ……14
I.3.4 introduction de LED jaune……..15
I.4 DS18B20
I.4.1 introduction ……..16
I.4.2 schéma……..16
I.4.3 le circuit appliqué……17
I.4.4 Utiliser DS18B20……..17
I.5 Le PulseSensor
I.5.1 introduction……18
I.5.2 le circuit appliqué……18
I .5.3 Utiliser Pulsesensor……19
I.6 Le circuit du collier
I.6.1 le circuit du LED……19
I.6.2 le circuit de la pile……..20
I.6.3 le circuit total……..21
I.7 La concept de collier
II. Le programme
II.1 Les définitions
II.1.1 les définition de la fichie tête est la macro définition……..23
II.1.2 les définition pour les différents tableaux……23
II.2 Le programme des capteurs et contrôleur
II.2.1 les valeurs viennent de Pulsesensor……24

II.2.2 la fonction *filtre* de Pulsesensor……..24
II.2.3 la fonction *setup*…….25
II.3 Le programme pour les LEDs
II.3.1 la fonction de s'allumer…….25
II.3.2 la fonction de s'éteindre…….26
II.3.3 la fonction de l'attente……26
II.3.4 la fonction de l'alarme……26
II.3.5 la fonction *loop*……...28
II.4 Le programme globale……..30
La partie de la fonction……37
Conclusion ……38
Bibliothèque ……..39

INTRODUCTION

Notre objectif est créer un bijou de type collier combinant art et jeu de lumières réalisé avec des LED. C’est-à-dire qu'il faut ajouter une dimension dynamique, des capteurs pour remonter la température et le rythme cardiaque et harmoniser l'animation lumineuse et l'humeur du porteur. Pas seulement ça, les LED peuvent être contrôlées par un circuit basé.

Après le recherche, on décide d'utiliser le capteur PulseSensor comme le capteur de rythme cardiaque et le capteur DS18B20 comme le capteur de température. On va faire un collier composé par des LEDs et varié selon le changement de température et de rythme cardiaque.

ANALYSE DU PROJET

Analyse du concurrent

1. Bague lumineuse à LED clignotante

Etre luminescent jusqu'au bout des doigts, c'est l'idéal pour aller en fête. Une façon de raviver à nouveau votre enfance. Cette bague lumineuse à LED clignotante est parfaite pour ce genre de situation, elle fait partie des accessoires de déguisement qu'il faut absolument adopter. Extrêmement lumineuse, elle est parfaite pour une sortie nocturne.

Cette bague lumineuse à LED clignotante est faite en silicone de haute qualité, elle est incroyablement extensible, ce qui vous permet d'avoir un maximum de confort. Etant souple et résistante, elle s'adapte parfaitement à la taille de vos doigts. Pour correspondre aux goûts de chacun, elle est produite en 4 coloris au choix à savoir le vert, le bleu, le rose et l'orange. Ces dernières associent l'élégance et le look relax, vous allez sûrement captiver l'attention de vos convives. Cette bague est facile à utiliser, un seul bouton sert à l'allumage et à la mise hors tension.

Cette bague lumineuse à LED clignotante est économique et possède une faible consommation. Elle fonctionne avec une pile associée à une autonomie de 20 heures, une durée largement suffisante pour profiter de ses effets fantaisies, d'autant plus que la pile peut être changée au cas où elle est usée.

Cette bague lumineuse allie à la fois praticité et plaisir, elle offre une ambiance discothèque à toutes sortes de soirée. La bague clignote dans tous les sens projetant des multiples couleurs pour vous plonger dans une ambiance sur vitaminée.

2. La robe LED

- Télécommande robe LED Vêtements lumineux à LED Robe de mariée Mode double couche robe.

- Utilisez la télécommande pour changer les couleurs de LED : Batterie rechargeable.

- Cette robe lumineuse peut être utilisé à la fois dans la nuit de noce ou en plein jour fête de mariage. En une nuit splendide, il vous fait d'être le centre d'attention!

3. Collier en or LED | L’or de la baronne

La baronne or lumière jusqu'à RAS de cou avec LED or... WOW ! Assez dit !

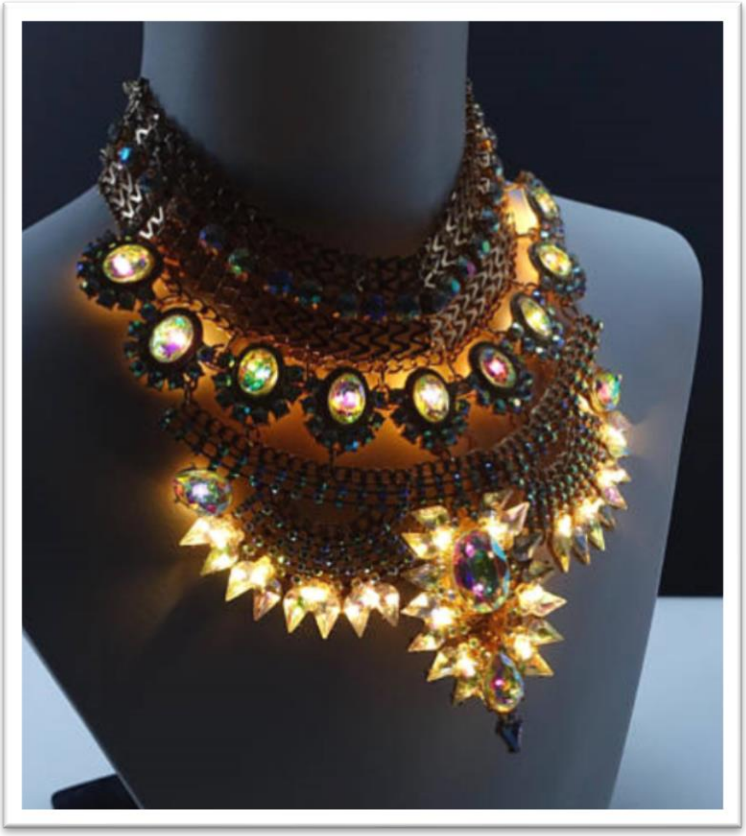
Chrome poli or AB cristal pierres précieuses et un éclairage blanc chaud sont éclairées par une batterie rechargeable intégrée.

- Semble incroyable jour et nuit - construction sans couture ; aucun fil exposé
- Peut être tourné On ou Off tout en le portant - bouton caché à la vue
- Reste allumé de Dusk Till Dawn - 12 + heures de pleine lumière.

- Jamais n’acheter une batterie - batterie rechargeable construit en.

Mini câble de charge USB et quelques auto adhésif mousse rembourrage inclus.

La chaîne tour de cou est ajustable de 20"
Bijoux bavoir est 8 '' long x 8 '' large & pèse seulement 9 onces



I. L’introduction

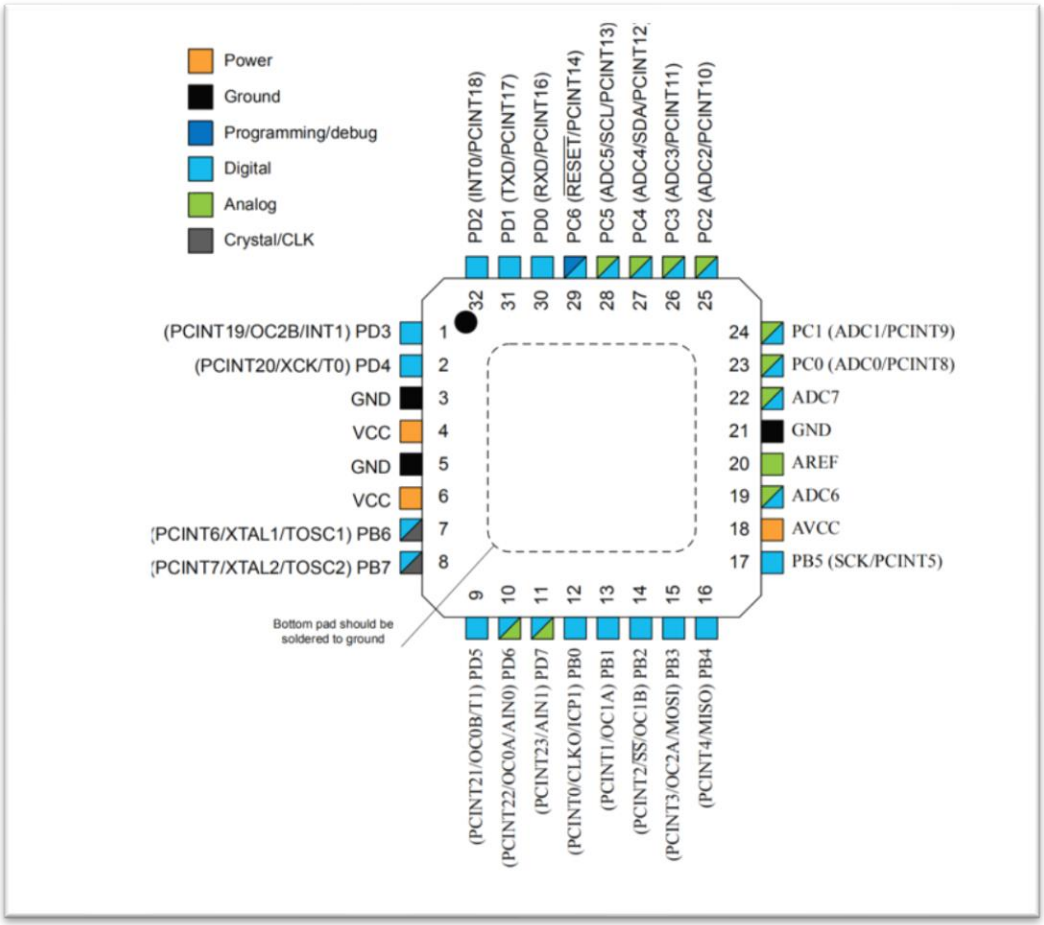
I.1 Le microcontrôleur

I.1.1 Introduction d'Atmega328p

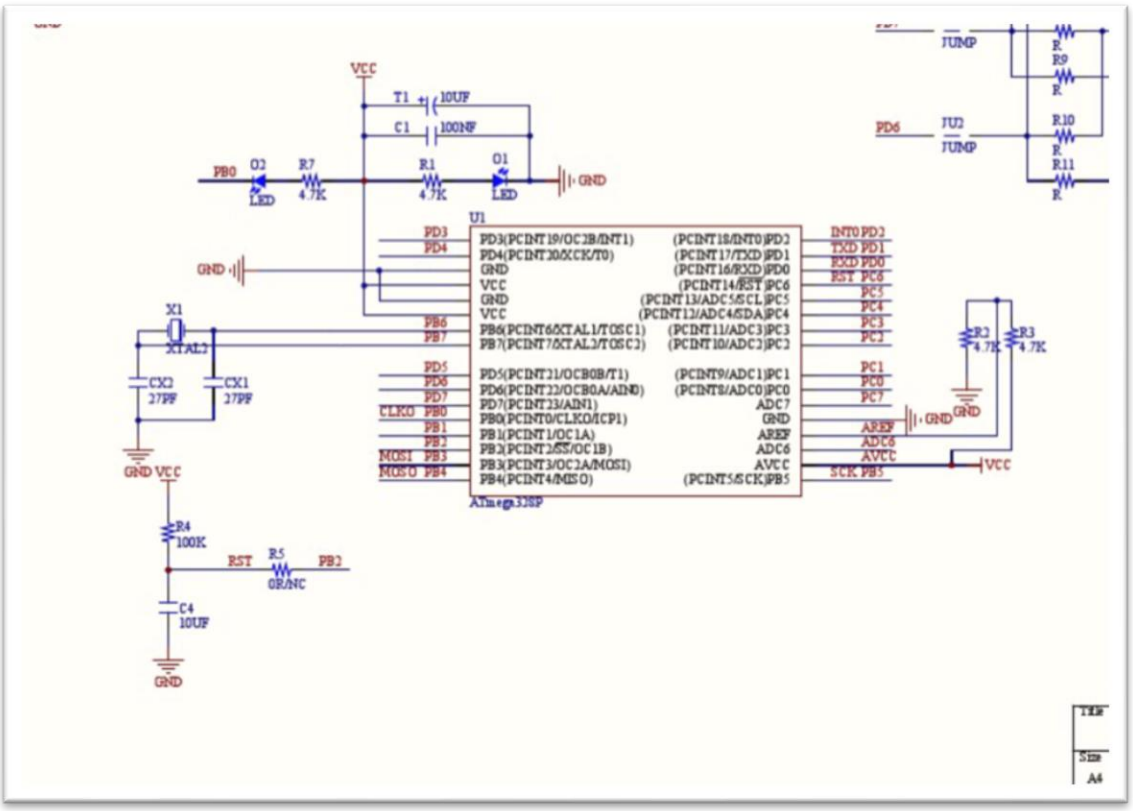
Le noyau Atmel AVR® combine un ensemble d'instructions riche avec 32 registres de travail à usage général. Tous les 32 registres sont directement connectés à l'unité logique arithmétique (ALU), permettant deux registres indépendants être accessible en une seule instruction exécutée en un cycle d'horloge. L'architecture résultante est plus de code efficace tout en atteignant des débits jusqu'à dix fois plus rapides que les microcontrôleurs CISC conventionnels. L'ATmega328 / P offre les fonctionnalités suivantes: 32 Ko de mémoire Flash programmable intégrée Capacités de lecture en écriture, 1 Mo d'EEPROM, 2 Mo de mémoire SRAM, 23 lignes d'E / S à usage général, 32 registres de travail à usage général, compteur temps réel (RTC), trois minuteries / compteurs flexibles avec comparer PWM, 1 USART programmables en série, 1 interface série à 2 fils orientée octet (I2C), un canal ADC 10 bits (8 canaux dans les paquets TQFP et QFN / MLF), un temporisateur de chien de garde programmable avec oscillateur interne, un port série SPI et six modes d'économie d'énergie sélectionnables par logiciel. Le ralenti le mode arrête le processeur tout en permettant à la SRAM, aux minuteries / compteurs, au port SPI et au système d'interruption de continuer à fonctionner. Le mode de mise hors tension enregistre le contenu du registre mais gèle l'oscillateur, désactivation de toutes les autres fonctions de la puce jusqu'à l'interruption suivante ou la réinitialisation matérielle. En mode économie d'énergie, la minuterie asynchrone continue à fonctionner, permettant à l'utilisateur de maintenir une base de minuterie pendant que le reste de la l'appareil est en train de dormir. Le mode ADC Noise Reduction arrête le processeur et tous les modules d'E / S sauf temporisateur asynchrone et ADC pour minimiser le bruit de commutation pendant les conversions ADC. En mode veille, le l'oscillateur de cristal / résonateur fonctionne pendant que le reste de l'appareil dort. Cela permet un démarrage très rapide combiné avec une faible consommation d'énergie. En mode de veille prolongée, l'oscillateur principal et le minuterie asynchrone continue à fonctionner. Atmel propose la bibliothèque QTouch® pour l'intégration de boutons tactiles capacitifs, de curseurs et de roues dans les microcontrôleurs AVR. L'acquisition brevetée du signal de transfert de charge offre une détection robuste inclut un rapport totalement dénudé des touches tactiles et inclut Adjacent Key Suppression® (AKS ™) technologie pour la détection sans ambiguïté des événements clés. La chaîne d'outils QTouch Suite, facile à utiliser, vous permet explorer, développer et déboguer vos propres applications tactiles. L'appareil est fabriqué en utilisant la technologie de mémoire non volatile haute densité d'Atmel. Le fournisseur d'accès Internet

sur puce Flash permet à la mémoire de programme d'être reprogrammée In-System via une interface série SPI, par un programmeur de mémoire non volatile conventionnel, ou par un programme de démarrage sur puce fonctionnant sur le noyau AVR. Le programme de démarrage peut utiliser n'importe quelle interface pour télécharger le programme d'application dans le Flash Application Mémoire. Les logiciels de la section Boot Flash continueront à fonctionner pendant que la section Application Flash mis à jour, fournissant une véritable opération Read-While-Write. En combinant un processeur RISC 8 bits avec In-System Flash auto-programmable sur une puce monolithique, l'Atmel ATmega328 / P est un puissant microcontrôleur fournit une solution hautement flexible et rentable à de nombreuses applications de contrôle embarquées. L'ATmega328 / P est pris en charge avec une suite complète d'outils de développement de programmes et de systèmes, notamment : C Compilateurs, assembleurs de macros, débogueurs / simulateurs de programmes, émulateurs en circuit et kits d'évaluation.

I.1.2 l'image d'Atmega328p



I.1.3 le circuit appliqué

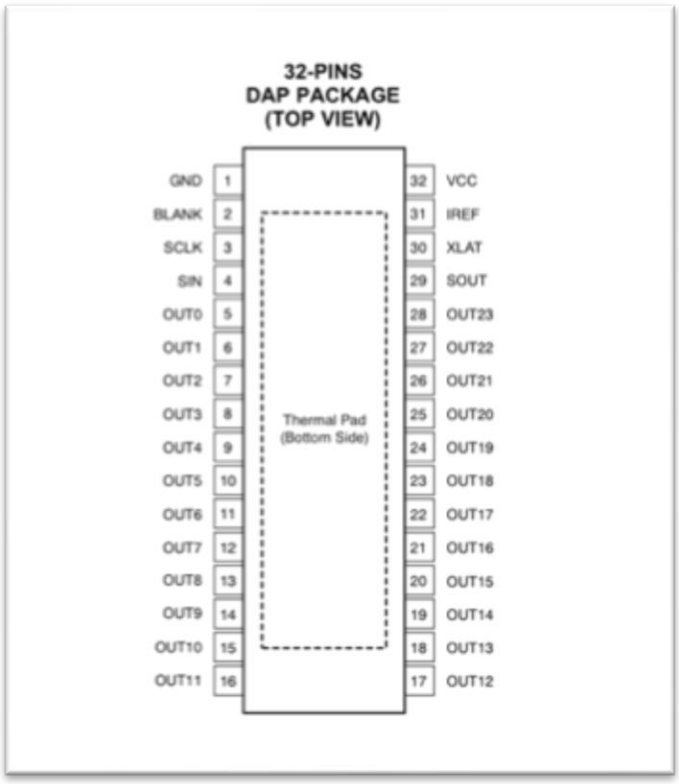


I.2 Le contrôleur

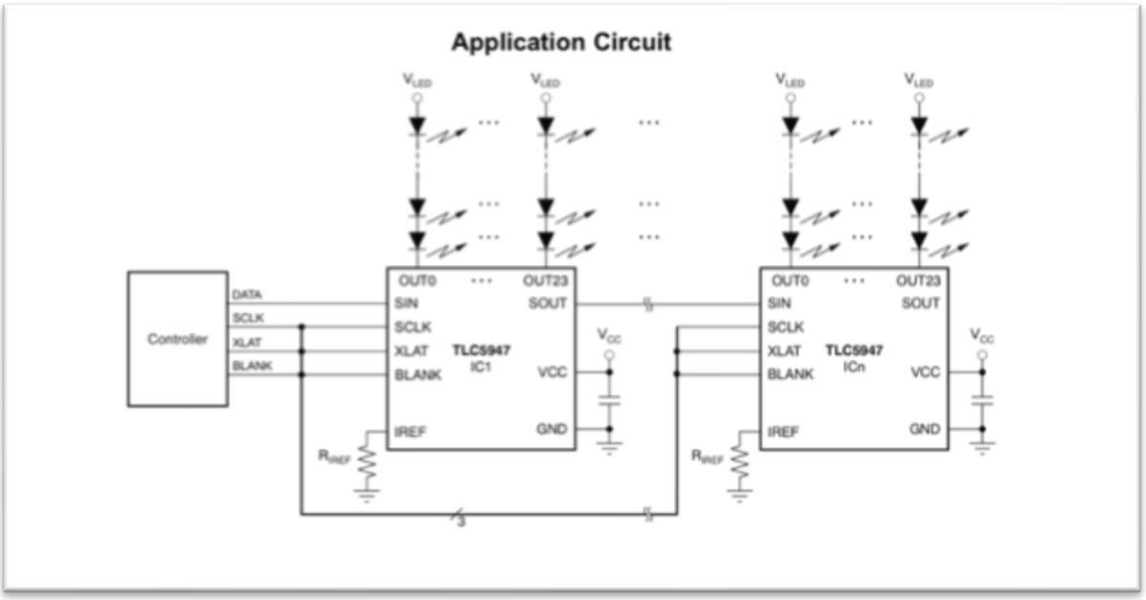
I.2.1 introduction de TLC5947

Le TLC5947 est un pilote de DEL de récepteur à courant constant à 24 canaux. Chaque canal est individuellement réglable avec 4096 pas modulés par largeur d'impulsion (PWM). La commande PWM est répétée automatiquement avec les données de niveaux de gris (GS) programmées. Les données GS sont écrites via un port d'interface série. La valeur actuelle de tous les 24 canaux est définie par une seule résistance externe.
Le TLC5947 dispose d'une fonction d'arrêt thermique (TSD) qui désactive tous les pilotes de sortie lors d'une surchauffe. Tous les pilotes de sortie redémarrent automatiquement lorsque la température redevient normale.

I.2.2 le schéma



I.2.3 le circuit appliqué



I.2.4 librairie

On peut étudier comment utiliser ce contrôleur et télécharger sa librairie dans la web site après :
https://learn.adafruit.com/tlc5947-tlc59711-pwm-led-driver-breakout

Dans notre projet, nous avons utilisé deux fonctions principales pour contrôler les LEDs s'allument et s'éteignent.

void setPWM(uint8_t chan, uint16_t pwm);
Call this to set the PWM level for a channel.

- chan = Channel
- pwm = PWM level (0 = minimum, 4095 = maximum)

void write(void);
Call this after every change to write the new PWM levels to the device.

I.3 Les LEDs

I.3.1 introduction de LED orange

Le HSMD-C170 est une puce SMD orange avec une lentille non teintée / diffuse, en GaP. Il est conçu dans un format standard de 2 x 1,25 mm avec un profil de 0,8 mm. Il est compatible avec les processus de soudure IR par refusion. La petite taille et l'angle de vision large font de cette LED un choix de premier ordre pour les applications de rétroéclairage et l'éclairage du panneau avant, en particulier lorsque l'espace est important.

- Petite taille
- Empreinte standard de l'industrie
- Compatible avec la soudure IR
- Optique diffuse
- -40 à 85 ° C Plage de température de fonctionnement

Couleur	Dimension	type	Courant	tension
Orange	1.25mm*1.4mm	CMS	20mA	2.2V

I.3.2 introduction de LED bleu

La HSMR-C170 est une LED Top Mount Chip qui utilise un matériau InGaN hautement efficace et à haute luminosité pour offrir un bleu haute performance à un prix compétitif. Ce bleu 470nm est une teinte unique qui permet de différencier les couleurs d'un produit.

- Petite taille
- Empreinte industrielle standard
- Optique diffuse
- Emission à angle droit
- Compatible avec la soudure IR

couleur	dimension	type	courant	tension
bleu	1.25mm*1.4mm	CMS	20mA	3.4V

I.3.3 introduction de LED vert

Le HSMG-C170 d'Avago est une LED puce rectangulaire diffuse non teinté. La DEL est de petite taille, son grand angle de vue en fait un premier choix pour les applications de rétroéclairage et l'éclairage de panneau avant particulier où l'espace est une gravité majeure.

- Compatible avec la soudure IR
- Empreinte standard 2 mm x 1,25 mm avec un profil de 0,8 mm

- Kit angle droit et inverseur avec fixation disponible
- LED de couleur verte
- Le matériau de LED est GaP
- LED forme rectangulaire
- Tension directe de 2.2V
- Courant direct (If) de 20mA
- Intensité lumineuse de 15mcd
- Angle de vision de 170 °

couleur	dimension	type	courant	tension
vert	1.25mm*1.4mm	CMS	20mA	2.2V

I.3.4 introduction de LED jaune

Le HSMY-C170 est un LED Surface Mount Chip, de couleur jaune. Cette puce LED est conçue dans un emballage standard de l'industrie pour faciliter la manipulation et l'utilisation. Le HSMY-C170 a l'empreinte de 2,0 x 1,25 mm largement utilisée. Ce paquet est compatible avec les processus de soudure IR par refusion. La petite taille et l'angle de vision large font de cette LED un choix idéal pour les applications de rétroéclairage et l'éclairage du panneau avant, en particulier lorsque l'espace est important.

- Petite taille
- Compatible avec la soudure IR
- Optique diffuse
- -40 à 85 ° C Plage de température de fonctionnement

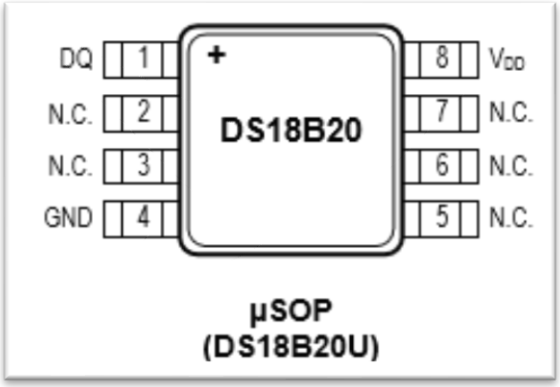
couleur	dimension	type	courant	tension
jaune	1.25mm*1.4mm	CMS	20mA	2.1V

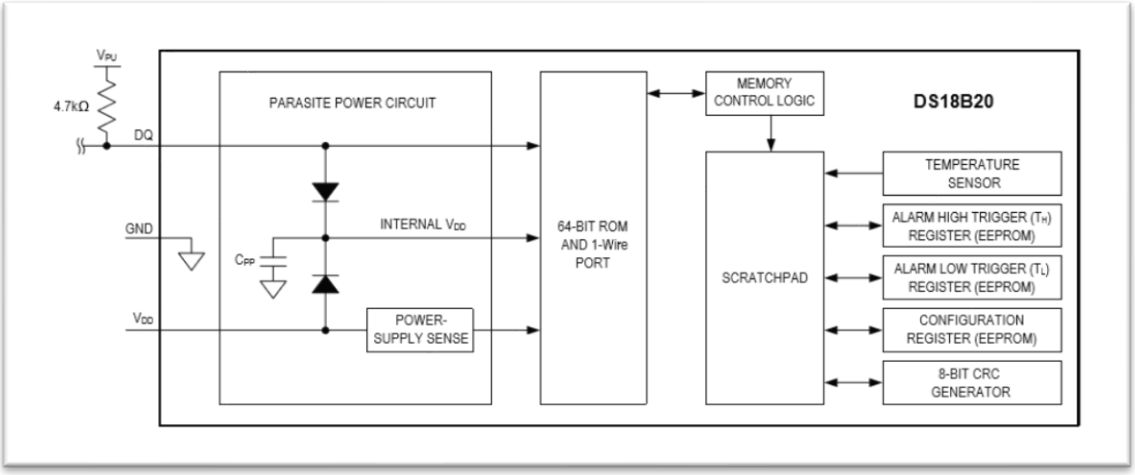
I.4 DS18B20

I.4.1 introduction

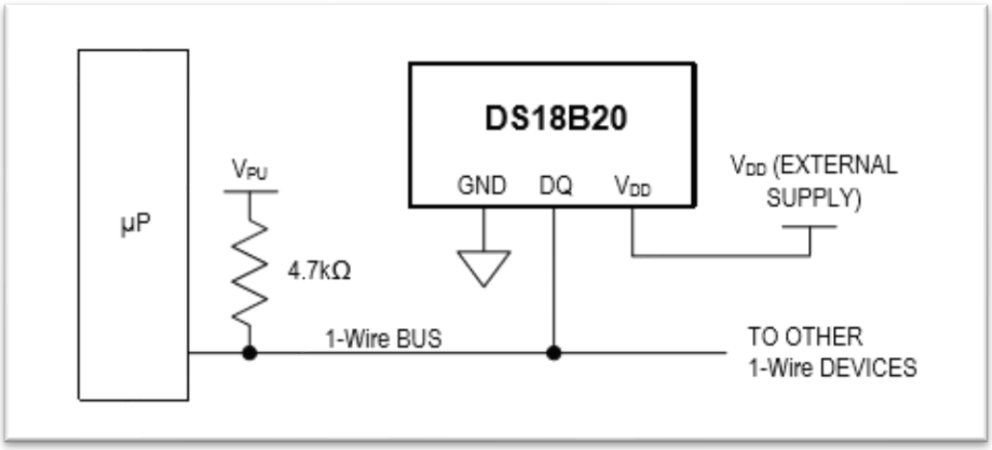
Le thermomètre numérique DS18B20 fournit des mesures de température Celsius de 9 à 12 bits et dispose d'une fonction d'alarme avec des points de déclenchement supérieurs et inférieurs non volatils programmables par l'utilisateur. Le DS18B20 communique sur un bus 1-Wire qui, par définition, ne nécessite qu'une seule ligne de données (et la masse) pour la communication avec un microprocesseur central. De plus, le DS18B20 peut directement tirer son énergie de la ligne de données (« parasite power »), éliminant ainsi le besoin d'une alimentation externe. Chaque DS18B20 possède un code série unique de 64 bits qui permet à plusieurs DS18B20 de fonctionner sur le même bus 1-Wire. Ainsi, il est simple d'utiliser un seul microprocesseur pour contrôler de nombreux DS18B20 répartis sur une grande surface. Les applications qui peuvent bénéficier de cette fonctionnalité comprennent les contrôles de l'environnement HVAC, les systèmes de surveillance de la température à l'intérieur des bâtiments, de l'équipement ou de la machinerie, et les systèmes de surveillance et de contrôle des processus.

I.4.2 schéma





I.4.3 le circuit appliqué



Le schéma appliqué est simple. Il faut juste lier le VCC avec VDD, lier la masse avec GND et le DQ est l'interface de donné qui est lié avec le microcontrôleur.

I.4.4 Utiliser DS18B20

Il faut télécharger et ajouter deux bibliothèques à l'Arduino IDE
https://www.arduinolibraries.info/libraries/dallas-temperature
https://www.arduinolibraries.info/libraries/one-wire
Dans notre projet, nous avons utilisé

```
OneWire oneWire(ONE_WIRE_BUS); //define ds18b20
DallasTemperature sensors(&oneWire); //pass reference to sensor
```

```
//prendre la valeur de temperature
  sensors.requestTemperatures();
  float tempCValue=sensors.getTempCByIndex(0);
  Serial.println(tempCValue);
```

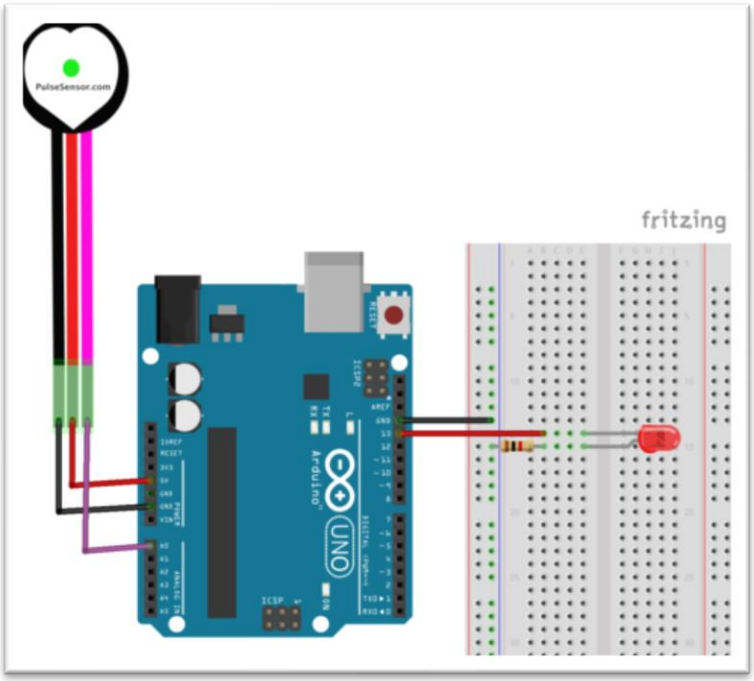
I.5 Le PulseSensor

I.5.1 introduction

Pulse Sensor est un capteur de fréquence cardiaque plug-and-play bien conçu pour Arduino. Il peut être utilisé par les étudiants, les artistes, les athlètes, les fabricants et les développeurs de jeux et mobiles qui souhaitent intégrer facilement des données de rythme cardiaque dans leurs projets. Le capteur se clipse sur le bout du doigt ou sur le lobe de l'oreille et se branche directement sur l'Arduino avec des câbles de démarrage. Il comprend également une application de surveillance open-source qui représente votre pouls en temps réel.

I.5.2 le circuit appliqué

Ce capteur est simple juste trois interfaces. Le fil rouge est de VCC, le fil noir est de GND et le fil violet est le fil de donné. Donc on lie le fil rouge avec VCC sur l'Atmega328p, le fil noir avec la masse. Et le fil violet avec un interface de donné sur l'Atmeag328P.



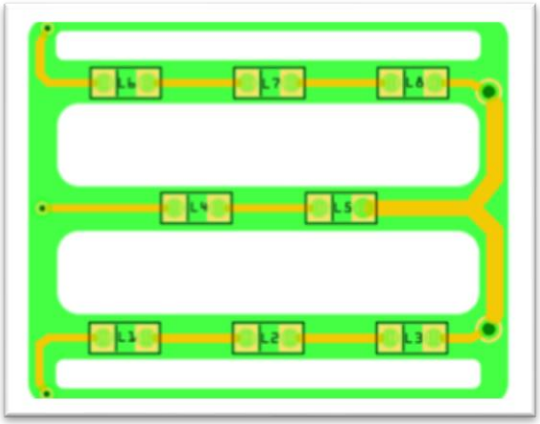
I .5.3 Utiliser Pulsesensor

Il faut télécharger et ajouter une bibliothèque à l'Arduino IDE
https://www.arduinolibraries.info/libraries/adafruit-neo-pixel
Dans notre projet, nous avons utilisé

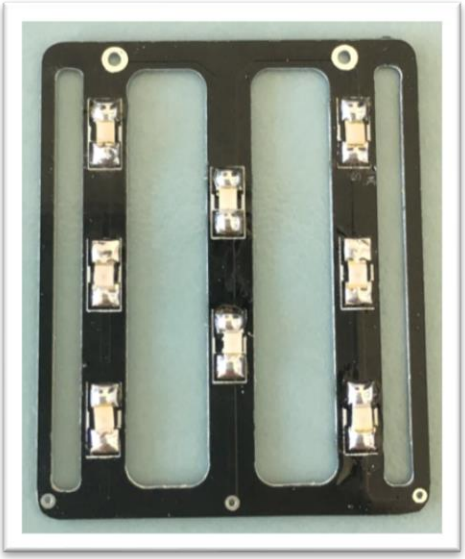
```
//prendre la valeur de pulse
  int heartValue = analogRead(HEART PIN);
```

I.6 Le circuit du collier

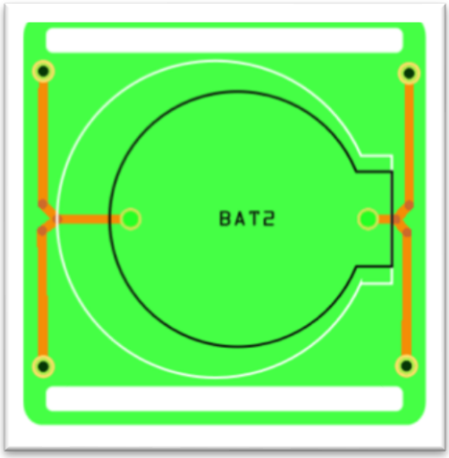
I.6.1 le circuit du LED



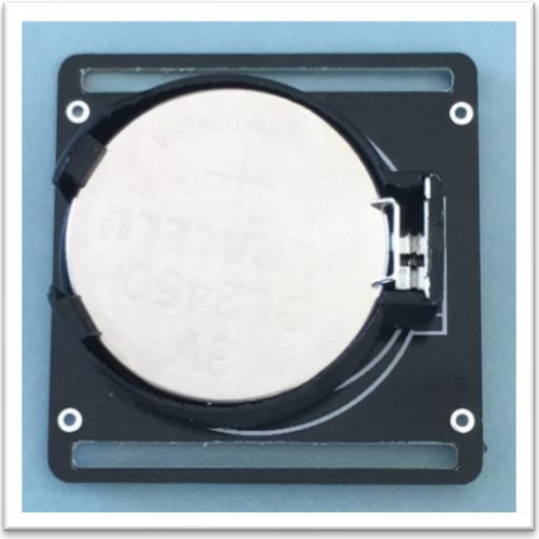
Parce qu'un contrôleur a 24 sorties pour contrôler les LEDs et pour faciliter de construire le collier, on divise 48 sorties en groupe de 3 sorties. Et on soude trois sorties dans un même PCB. En concernant la tension de chaque LED, on choisit mettre deux LEDs bleus ensemble pour une sortie et on mélange le LED orange, le LED vert et le LED jaune dans un sortie. Finalement, on fait le PCB de LED.



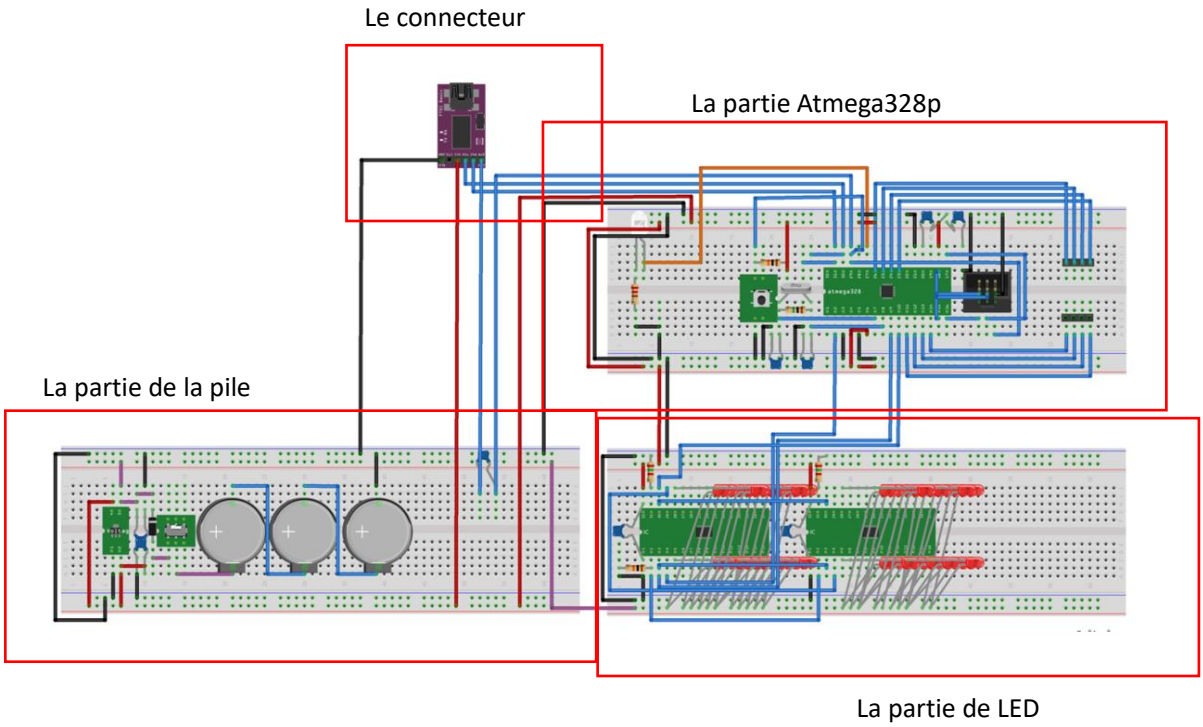
I.6.2 le circuit de la pile

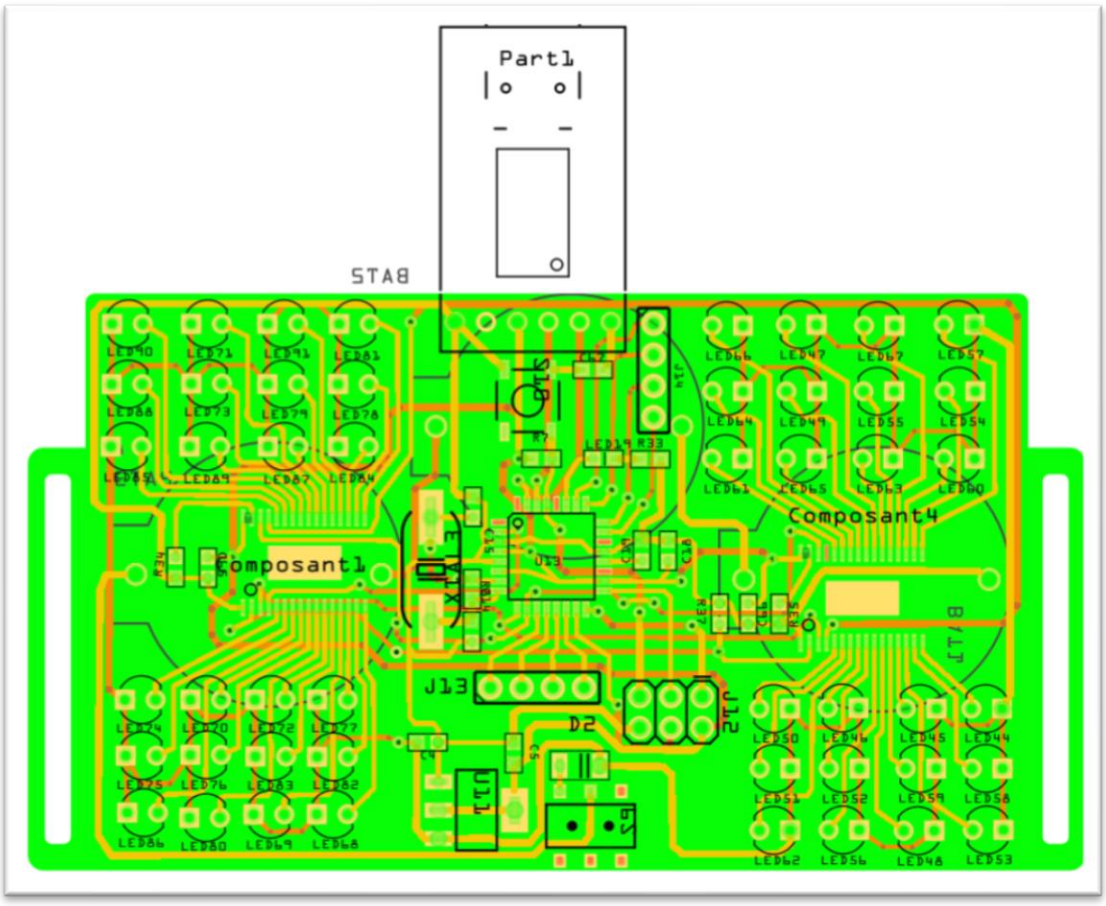


Parce que le pile est grand et lourd, donc on fait un PCB de pile indépendant.
On soude quatre PCB comme ça pour soutenir la tension du circuit globalement.



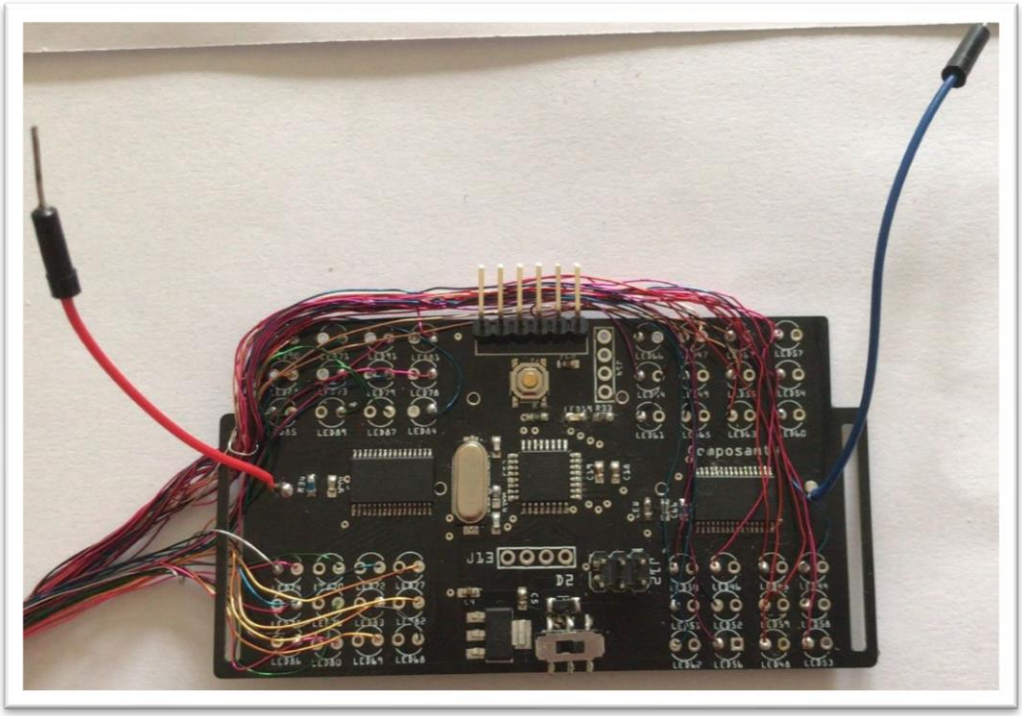
I.6.3 le circuit total





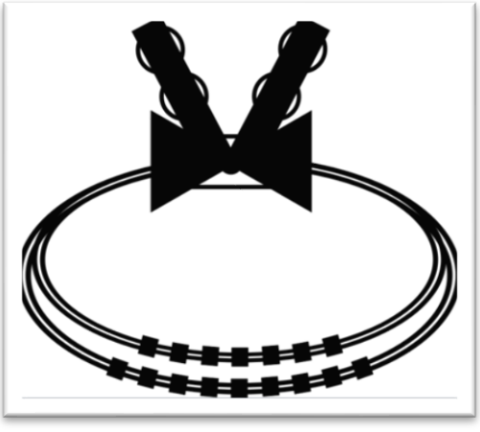
Ce circuit est composé par trois parties. Une partie de pile, une partie du contrôleur Atmega328p et une partie du contrôleur TLC5947. Considérant la partie d'USB est grande, donc on ne met pas cette partie dans ce circuit et on ajoute un connecteur pour connecter la partie d'USB. Quand on a besoin de transmettre les codes au contrôleur Atmega328p, on connecte le PC avec le circuit par le connecteur et quand on finit la transmission, on juste reste le connecteur dans ce circuit. C'est plus simple et simplifie le circuit total.

L'idée du circuit est de deux contrôleurs TLC5947 contrôler tous les LEDs et le Atmega328p contrôler tous ces contrôleurs. Parce que le capteur de température a trois interfaces : VCC, GND et une interface de donné. Donc on lie cette interface de donné avec PD7 sur l'Atmega328p pour transfert les données. Et pour le capteur cardiaque a aussi trois interfaces : VCC, CND et une interface de donné. On lie PC0 sur l'Atmega328p avec cette interface de donné. Du coup, on peut contrôler les deux capteurs par l'Atmega328p.



I.7 La concept de collier

Notre collier est composé par 128 LEDs SMD, dont il y 32 LEDs de chaque couleu. Il y a deux ligne de LEDs, la première ligne a 7 groups des LEDs, c-t-à dire qu'il y a 56 LEDs SMD, et la deuxième ligne a 9 groups des LEDs, c-t-à dire qu'il y 72 LEDs SMD



II. Le programme

II.1 Les définitions

II.1.1 les définition de la fichie tête est la macro définition

Nous avons défini l'interface des deux capteurs, et les paramètres des deux contrôleurs.

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include "Adafruit_TLC5947.h"
#define ONE_WIRE_BUS PC0  // ds18b20_pin
#define HEART_PIN   PD7  //pulse_pin
#define NUM_TLC5974 2    //nombre du controleur
//declaration de TLC5947
#define data   4
#define clock   5
#define latch   6
#define oe  -1
Adafruit_TLC5947 tlc = Adafruit_TLC5947(NUM_TLC5974, clock, data, latch);
OneWire oneWire(ONE_WIRE_BUS); //define_ds18b20
DallasTemperature sensors(&oneWire); //pass reference to sensor
```

II.1.2 les définition pour les différents tableaux

Les différents tableaux sont en fonction de l'ordre s'allumer de tous les LEDs, nous les avons utilisé pour mieux contrôler.

```
uint8_t i=0;
uint8_t k=0;

uint8_t
iarray[48]={7, 23, 1, 5, 17, 22, 2, 18, 0, 20, 8, 11, 19, 13, 3, 9, 6, 16, 15, 4, 10, 45, 40
, 37, 36, 31, 33, 42, 47, 26, 24, 30, 25, 43, 44, 27, 34, 41, 35, 39, 38, 46, 28, 32, 29, 14,
12, 21};

uint8_t
jarray1[21]={10, 4, 15, 16, 6, 9, 3, 13, 19, 11, 8, 20, 0, 18, 2, 22, 17, 5, 1, 23, 7};
uint8_t
jarray2[27]={45, 40, 37, 36, 31, 33, 42, 47, 26, 24, 30, 25, 43, 44, 27, 34, 41, 35, 39,
38, 46, 28, 32, 29, 14, 12, 21};

uint8_t
karray1[32]={7, 1, 5, 22, 2, 0, 20, 11, 19, 3, 9, 16, 15, 10, 45, 37, 36, 33, 42, 26, 24, 2
5, 43, 27, 34, 35, 39, 46, 28, 29, 14, 21};
uint8_t karray2[16]={12, 32, 38, 41, 44, 30, 47, 31, 40, 4, 6, 13, 8, 18, 17, 23};
```

II.2 Le programme des capteurs et contrôleurs

II.2.1 les valeurs viennent de Pulsesensor

Le capteur Pulsesensor a un signal analogique, donc ce n'est pas très stales, mais pas seulement ça, la graphe de la fréquence cardiaque n'est pas une courbe horizontale. Il faut faire filtrer avant appeler ces valeurs.

II.2.2 la fonction *filtre* de Pulsesensor

```
//filtre
int filtre (int input) {
#define FILTER_SIZE 10
static int filterArray[FILTER_SIZE] = {0};
static int fi = 0;
filterArray[fi++] = input;
if (fi >= FILTER_SIZE) {
  fi = 0;
}
int32_t output = 0;
for (int iff = 0; iff < FILTER_SIZE; iff++) {
  output += filterArray[iff];
}
return int(output / FILTER_SIZE);
}
```

Nous avons fait un traitement des valeurs pour laisser les valeurs varient plus incliné. Le résultat après la filtrage est comme ci-dessous

```
472
470
468
469
468
467
471
470
470
470
470
470
470
469
469
```

II.2.3 la fonction *setup*

```
void setup() {
  Serial.begin(9600);
  sensors.begin();
  tlc.begin();
  if (oe >= 0) {
    pinMode(oe, OUTPUT);
    digitalWrite(oe, LOW);
  }
}
```

II.3 Le programme pour les LEDs

II.3.1 la fonction de s'allumer

Nous voulons réaliser de les LEDs s'allument par une façon progressivement, avec le chargement de la fréquence cardiaque. Donc nous avons pris la valeur de la fréquence cardiaque, et nous l'avons utilisé comme une variable pour le changement de la luminance.

```
//LED_allumer
void rise(uint8_t chan, uint16_t pwm) {
//prendre la valeur de pulse
  int heartValue = analogRead(HEART_PIN);
//filtre la valeur de pulse
  int filterValue = filter(heartValue);
  Serial.println(filterValue);
  uint32_t j;
  Serial.println("rise");
//On change la luminance par la valeur de rythme cardiaque
//On peut aussi utilise la valeur *tempCValue* comme une variable pour changer la luminance de LED
  for(j=0; j<pwm; j+=filterValue) {
    tlc.setPWM(chan,j);
    tlc.write();
  }
}
```

II.3.2 la fonction de s’éteindre

Nous voulons réaliser de les LEDs s’éteignent par une façon progressivement, avec le chargement de la fréquence cardiaque. Donc nous avons pris la valeur de la fréquence cardiaque, et nous l’avons utilisé comme une variable pour le changement de la luminance

```
//LED eteindre
void down(uint8_t chan, uint16_t pwm) {
//prendre la valeur de pulse
  int heartValue = analogRead(HEART_PIN);
//filtre la valeur de pulse
  int filterValue = filter(heartValue);
  Serial.println(filterValue);
  int32_t j;
  Serial.println("down");
//On change la luminance par la valeur de rythme cardiaque
//On peut aussi utilise la valeur *tempCValue* comme une variable pour changer la luminance de LED
  for(j=pwm; j>=0; j-=filterValue) {
    tlc.setPWM(chan,j);
    tlc.write();
  }
}
```

II.3.3 la fonction de l’attente

Nous voulons contrôler les deux lignes de LEDs séparément, mais la quantité des LEDs de chaque ligne n’est pas la même. Donc quand la première ligne fini sa partie, elle faut atteindre jusqu’à la deuxième ligne fini sa partie. Et puis, les deux lignes peuvent recommencer, pour maintenir la synchronisé.
Nous avons défini que quand les LEDs sont en attente, ils s’éteignent.

```
//LED atteint
void atteint(uint8_t chan,uint16_t pwm){

  tlc.setPWM(chan,pwm);

  tlc.write();
}
```

II.3.4 la fonction de l’alarme

Nous voulons contrôler la température : quand la valeur de température est supérieur à la valeur normale, tous les LEDs exécutent la fonction de l'alarme pour alarmer ;quand la valeur de température est normale, les LEDs exécutent la fonction normale.
Nous avons plusieurs des idées pour la fonction de l’alarme.
La première est que lorsque la valeur de température est supérieur à la valeur normale, tous les LEDs s'allument durée 30 secondes pour l'alarme.

```
//LED_alarme
void alarme(uint8_t chan,uint16_t pwm){
  tlc.setPWM(chan,pwm);
  tlc.write();
}
```

La deuxième est que lorsque ce collier alarme, tous les LEDs clignotent rapidement, de gauche à droite, de la première ligne à la deuxième ligne, jusqu'à 30 fois, et puis ils s'éteignent pour 5 secondes.

```
//LED_alarme2
void alarme2(){
  for(int nn=0;nn<48;nn++){
    tlc.setPWM(iarray[nn],4095);
    tlc.write();}
  for(int nn=0;nn<48;nn++){
    tlc.setPWM(iarray[nn],0);
    tlc.write();}
}
```

La troisième est que lorsque ce collier alarme, tous les LEDs bleues et autres LEDs colorées traversent le flash jusqu'à 15 fois, et puis ils s'éteignent pour 5 secondes. Les LEDs bleues s'allument de la deuxième ligne à la première ligne, de droite à gauche, et s'éteignent de la première ligne à la deuxième ligne, de gauche à droite. Les autres LEDs colorées s'allument de la première ligne à la deuxième ligne, de gauche à droite, et s'éteignent de la deuxième ligne à la première ligne, de droite à gauche.

```
//LED_alarme3
  void alarme3(){
    //LEDs bleues
    for(int k2=0;k2<16;k2++){
      alarme(karray2[k2],4095);}
    for(int k2=15;k2>=0;k2--){
      atteint(karray2[k2],0);}
    //les autres LEDs colorées
    for(int k1=0;k1<32;k1++){
      alarme(karray1[k1],4095);}
    for(int k1=31;k1>=0;k1--){
      atteint(karray1[k1],0);}
  }
```

II.3.5 la fonction *loop*

Quand tous les valeurs viennent de les deux capteurs sont normaux, le programme marche dans le cas principale. Nous avons deux idées pour contrôler les LEDs dans le cas principale.
La première est que lorsque la valeur de température est normale, tous les LEDs s'allument progressivement et s'étendent progressivement un par un en fonction de la valeur rythme cardiaque. Si la valeur de rythme cardiaque est grande, les LEDs s'allument progressivement et s'étendent progressivement rapide, sinon les LEDs s'allument progressivement et s'étendent progressivement lentement. Ce programme vérifie la température 2 fois par minute(30 secondes par fois) et vérifie la rythme cardiaque chaque 0.2 secondes.

```
void loop() {
//prendre la valeur de temperature
  sensors.requestTemperatures();
  float tempCValue=sensors.getTempCByIndex(0);
  Serial.println(tempCValue);
//si la temperature n'est pas normale
  //La valeur de température peut-être en fonction de la demande
  //On peut aussi utiliser la valeur de pouls pour alarmer, par changer *tempCValue* à *filtervalue*
  if(tempCValue>37)
  {
    for(int ii=0;ii<48;ii++)
    {
```

```
      alarme(iarray[ii],4095);
    }
  //tous les LEDs s'allument durée 30 secondes pour l'alarme
  delay(30000);
  }
//sinon
  else
  {
    //Ce façon de allumer et éteindre peut être changer par les autres, les autres façons peuvent être trouvés après
    rise(iarray[i],4095);
    down(iarray[i],4095);
    i=i+1;
    if(i>47) i=0;
    Serial.println(i);
    //On vérifie la rythme cardiaque chaque 0.2 secondes
    delay(200);
  }
}
```

La deuxième est que lorsque ce collier fonctionne normal, tous les valeurs viennent des capteurs sont normaux, il y une autre façon pour changer les états de tous les LEDs. La première ligne(au dessus) de LEDs s'allument de droite à gauche, et la deuxième ligne(au dessous) de LEDs s'allument de gauche à droite. Quand la première ligne fini, elle s'éteint et atteint jusqu'à la deuxième ligne fini aussi. Et les deux ligne recommencent ensemble.

```
//la fonction principale
void loop() {
  //prendre la valeur de temperature
  sensors.requestTemperatures();
  float tempCValue=sensors.getTempCByIndex(0);
  Serial.println(tempCValue);
  //si la temperature n'est pas normale
  if(tempCValue>30)
  {
    for(int n=0;n<15;n++)
    {
       alarme3();
    }
    delay(5000);
  }
  //sinon
  else
  {
    if(i>20)
    {
      for(int l=0;l<21;l++)
      {
      atteint(jarray1[l],0);
      }
      rise(jarray2[k],4095);
      down(jarray2[k],4095);
      i=i+1;
      k=k+1;
    }
    else
    {
      rise(jarray1[i],4095);
      rise(jarray2[k],4095);
      down(jarray1[i],4095);
      down(jarray2[k],4095);
      i=i+1;
      k=k+1;
    }
```

```
    if( k>26 && i>26)
    {
      for(int j=0;j<27;j++)
      {
        atteint(jarray2[j],0);
      }
      k=0;
      i=0;
    }
    Serial.println(i);
    Serial.println(k);
    delay(200);
  }
}
```

II.4 Le programme globale

La première façon

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include "Adafruit_TLC5947.h"
#define ONE_WIRE_BUS PD7  // ds18b20_pin
#define HEART_PIN   PC0  //pulse_pin
#define NUM_TLC5974 2    //nombre du controleur

//declaration de TLC5947
#define data   4
#define clock   5
#define latch   6
#define oe  -1

Adafruit_TLC5947 tlc = Adafruit_TLC5947(NUM_TLC5974, clock, data, latch);
OneWire oneWire(ONE_WIRE_BUS); //define_ds18b20
DallasTemperature sensors(&oneWire); //pass_reference_to_sensor

void setup() {
  Serial.begin(9600);
  sensors.begin();
  tlc.begin();
  if (oe >= 0) {
    pinMode(oe, OUTPUT);
    digitalWrite(oe, LOW);
  }
}

uint8_t i=0;

uint8_t
iarray[48]={7,23,1,5,17,22,2,18,0,20,8,11,19,13,3,9,6,16,15,4,10,45,40,37,36,3
1,33,42,47,26,24,30,25,43,44,27,34,41,35,39,38,46,28,32,29,14,12,21};

//la fonction principale
void loop() {
  //prendre la valeur de temperature
  sensors.requestTemperatures();
  float tempCValue=sensors.getTempCByIndex(0);
  Serial.println(tempCValue);
```

```
  //si la temperature n'est pas normale
  if(tempCValue>30)
  {
    for(int ii=0;ii<48;ii++)
    {
      alarme(iarray[ii],4095);
    }
    delay(30000);
  }

  //sinon
  else
  {
    rise(iarray[i],4095);
    down(iarray[i],4095);
    i=i+1;
    if(i>47) i=0;
    Serial.println(i);
    delay(200);
  }
}

//LED_allumer

void rise(uint8_t chan, uint16_t pwm) {
  //prendre la valeur de pulse
  int heartValue = analogRead(HEART_PIN);
  //filtre la valeur de pulse
  int filterValue = filter(heartValue);
  Serial.println(filterValue);
  uint32_t j;
  Serial.println("rise");
  for(j=0; j<pwm; j+=filterValue) {
    tlc.setPWM(chan,j);
    tlc.write();
  }
}

//LED_eteindre
void down(uint8_t chan, uint16_t pwm) {
  //prendre la valeur de pulse
  int heartValue = analogRead(HEART_PIN);
  //filtre la valeur de pulse
  int filterValue = filter(heartValue);
```

```
  Serial.println(filterValue);
  int32_t j;
  Serial.println("down");
  for(j=pwm; j>=0; j-=filterValue) {
    tlc.setPWM(chan,j);
    tlc.write();
  }
}

//LED_alarme
void alarme(uint8_t chan,uint16_t pwm){
  tlc.setPWM(chan,pwm);
  tlc.write();
}

//filter
int filter (int input) {
#define FILTER_SIZE 10
  static int filterArray[FILTER_SIZE] = {  0  };
  static int fi = 0;
  filterArray[fi++] = input;
  if (fi >= FILTER_SIZE) {
    fi = 0;
  }
  int32_t output = 0;
  for (int iff = 0; iff < FILTER_SIZE; iff++) {
    output += filterArray[iff];
  }
  return int(output / FILTER_SIZE);
}
```

La deuxième façon

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include "Adafruit_TLC5947.h"
#define ONE_WIRE_BUS PD7  // ds18b20_pin
#define HEART_PIN   PC0  //pulse_pin
#define NUM_TLC5974 2    //nombre du controleur

//declaration de TLC5947
#define data   4
#define clock   5
#define latch   6
#define oe  -1
```

```
Adafruit_TLC5947 tlc = Adafruit_TLC5947(NUM_TLC5974, clock, data, latch);
OneWire oneWire(ONE_WIRE_BUS); //define_ds18b20
DallasTemperature sensors(&oneWire); //pass_reference_to_sensor

void setup() {
  Serial.begin(9600);
  sensors.begin();
  tlc.begin();

  if (oe >= 0) {
    pinMode(oe, OUTPUT);
    digitalWrite(oe, LOW);
  }
}

uint8_t i=0;
uint8_t k=0;

uint8_t
iarray[48]={7,23,1,5,17,22,2,18,0,20,8,11,19,13,3,9,6,16,15,4,10,45,40,37,36,3
1,33,42,47,26,24,30,25,43,44,27,34,41,35,39,38,46,28,32,29,14,12,21};

uint8_t jarray1[21]={10,4,15,16,6,9,3,13,19,11,8,20,0,18,2,22,17,5,1,23,7};
uint8_t
jarray2[27]={45,40,37,36,31,33,42,47,26,24,30,25,43,44,27,34,41,35,39,38,46,28
,32,29,14,12,21};

uint8_t
karray1[32]={7,1,5,22,2,0,20,11,19,3,9,16,15,10,45,37,36,33,42,26,24,25,43,27,
34,35,39,46,28,29,14,21};
uint8_t karray2[16]={12,32,38,41,44,30,47,31,40,4,6,13,8,18,17,23};

//la fonction principale
void loop() {
  //prendre la valeur de temperature
  sensors.requestTemperatures();
  float tempCValue=sensors.getTempCByIndex(0);
  Serial.println(tempCValue);

  //si la temperature n'est pas normale
  if(tempCValue<37)
  {
    for(int n=0;n<15;n++)
```

```
    {
       Alarme2();
    }
    delay(5000);
  }

  //sinon
  else
  {
    if(i>20)
    {
      for(int l=0;l<21;l++)
      {
        atteint(jarray1[l],0);
      }
      rise(jarray2[k],4095);
      down(jarray2[k],4095);
      i=i+1;
      k=k+1;
     }
     else
    {
      rise(jarray1[i],4095);
      rise(jarray2[k],4095);
      down(jarray1[i],4095);
      down(jarray2[k],4095);
      i=i+1;
      k=k+1;
    }
    if( k>26 && i>26)
    {
      for(int j=0;j<27;j++)
      {
        atteint(jarray2[j],0);
      }
      k=0;
      i=0;
    }
    Serial.println(i);
    Serial.println(k);
    delay(200);
  }
}
```

```
//LED_allumer
void rise(uint8_t chan, uint16_t pwm) {
  //prendre la valeur de pulse
  int heartValue = analogRead(HEART_PIN);
  //filtre la valeur de pulse
  int filterValue = filter(heartValue);
  Serial.println(filterValue);
  uint32_t j;
  Serial.println("rise");
  for(j=0; j<pwm; j+= filterValue) {
    tlc.setPWM(chan,j);
    tlc.write();
  }
}

//LED_eteindre
void down(uint8_t chan, uint16_t pwm) {
  //prendre la valeur de pulse
  int heartValue = analogRead(HEART_PIN);
  //filtre la valeur de pulse
  int filterValue = filter(heartValue);
  Serial.println(filterValue);
  int32_t j;
  Serial.println("down");
  for(j=pwm; j>=0; j-= filterValue) {
    tlc.setPWM(chan,j);
    tlc.write();

  }
}

//LED_alarme
void alarme(uint8_t chan,uint16_t pwm){
  tlc.setPWM(chan,pwm);
  tlc.write();
}

//LED atteint
void atteint(uint8_t chan,uint16_t pwm){
  tlc.setPWM(chan,pwm);
  tlc.write();
}

//LED_alarme2
```

```
void alarme2(){
  for(int nn=0;nn<48;nn++){
  tlc.setPWM(iarray[nn],4095);
  tlc.write();}
  for(int nn=0;nn<48;nn++){
  tlc.setPWM(iarray[nn],0);
  tlc.write();}
}

//LED_alarme3
void alarme3(){
  for(int k2=0;k2<16;k2++){
   alarme(karray2[k2],4095);}
  for(int k2=15;k2>=0;k2--){
  atteint(karray2[k2],0);}

  for(int k1=0;k1<32;k1++){
  alarme(karray1[k1],4095);}
  for(int k1=31;k1>=0;k1--){
  atteint(karray1[k1],0);}
}

//filter
int filter (int input) {
#define FILTER_SIZE 10
  static int filterArray[FILTER_SIZE] = {   0  };
  static int fi = 0;
  filterArray[fi++] = input;
  if (fi >= FILTER_SIZE) {
    fi = 0;
  }
  int32_t output = 0;
  for (int iff = 0; iff < FILTER_SIZE; iff++) {
    output += filterArray[iff];
  }
  return int(output / FILTER_SIZE);
}
```

La partie de la fonction

Parce que notre collier peut démontrer le changement de la température et du rythme cardiaque, donc notre collier peut être utilisé dans quel domaines.

S’il y a une personne qui a notre produit, quand il participe une fête, il va être plus joli. Le collier va changer la luminance selon le changement de température de corps et le rythme de radique. Ce collier a plusieurs couleurs et plusieurs façons d’allumer. Donc ça va être super charment.

Si les jeunes voudraient donner un gâteau pour ses grand-parents, ce collier va être un gâteau parfait, particulièrement pour la grand-mère. Ce collier pourrait surveiller la santé tout le temps. Parce qu’il y a un capteur de température et un capteur de radique pour tester la température de corps et le rythme de radique. Les changements peuvent être trouvé par le changement de brillance de LED.

Notre collier aussi peut être utilisé pour vérifier si quelqu’un ment. Parce que si quelqu’un ment, son rythme de cœur va changer. Donc ce sera facile de trouver.

Conclusion

Après ce projet, nous avons appris comment concepter un PCB et comment le réaliser.
Nous avons compris mieux de le circuit d’appliqué de l’Atmega328 et le circuit de l’Arduino uno.
Nous avons appris aussi les deux capteurs et les deux contrôleurs pour LED.
Nous avons écrit quelque programme sur Arduino IDE, c’est pratique réaliser quelques fonctions peuvent être utilisé dans le quotidien.
Il y a encore quelques parties peuvent être améliorées.
L'ensemble du processus est très intéressant.

Bibliothèque

1. DS18B20 https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf
2. PulseSensor https://www.generationrobots.com/media/DetecteurDePoulsAmplifie/PulseSensorAmpedGettingStartedGuide.pdf
3. TLC5947 librairie https ://learn.adafruit.com/tlc5947-tlc59711-pwm-led-driver-breakout/downloads-and-links
4. LED orange http://fr.farnell.com/broadcom-limited/hsmd-c170/led-cms-orange/dp/5790864?MER=bn_para_1TP_LastViewed_1
5. LED bleu http://fr.farnell.com/broadcom-limited/hsmr-c170/led-bleu/dp/8554749
6. LED vert http://fr.farnell.com/broadcom-limited/hsmg-c170/led-verte/dp/5790852
7. LED jaune http://fr.farnell.com/broadcom-limited/hsmy-c170/led-jaune/dp/5790876
8. Atmega328p http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf
9. TLC5947 https://cdn-shop.adafruit.com/datasheets/tlc5947.pdf