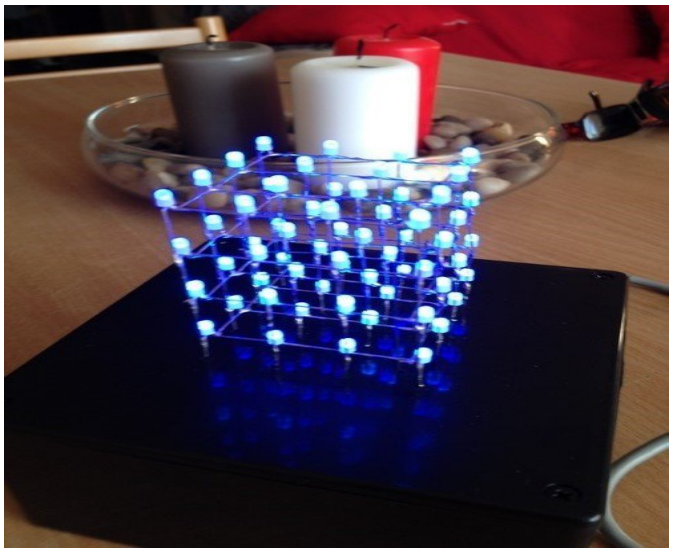
# FABRICATION ET PROGRAMMATION D'UN CUBE LED



**Mise en situation :**

Un prestataire technique évènementiel fait appel à vos compétences dans le domaine de la conception et de la programmation afin de lui concevoir un CUBE LED, d'écrire les programmes informatique nécessaires à sa mise en service et de lui faire une démonstration de son fonctionnement.

**Problématique :**

Etes capable réaliser et de configurer un CUBE LED 3x3x3 et le contrôler en utilisant une carte arduino.

Moyens mis à votre disposition :

- Carte Arduino
- Composants électroniques (LED, Transistor, résistance, Fils électriques)
- Matériels de test
- Connexion Internet.

| **Objectifs** | |
|---|---|
| Activités | Compétences |
| **A1-2** : préparation, intégration, assemblage, interconnexion des matériels.<br>**A1-3** : intégration des logiciels.<br>**A1-4** : test et validation. | **C3-2:** Réaliser l'intégration matérielle ou logicielle d'un équipement.<br>**C3-3**: Effectuer les tests nécessaires à la validation du fonctionnement des équipements. |
| **On demande**<br>- Réaliser un programme permettant de commander le produit<br>- de câbler des composants électroniques<br>- Modéliser et valider le fonctionnement du produit<br>- Valider le fonctionnement réel du produit | **Critères d'évaluation**<br>- Les informations nécessaires sont recueillies<br>- Les difficultés techniques sont repérées<br>- Les documents sont renseignés<br>- Les équipements matériels et logiciels sont identifiés |

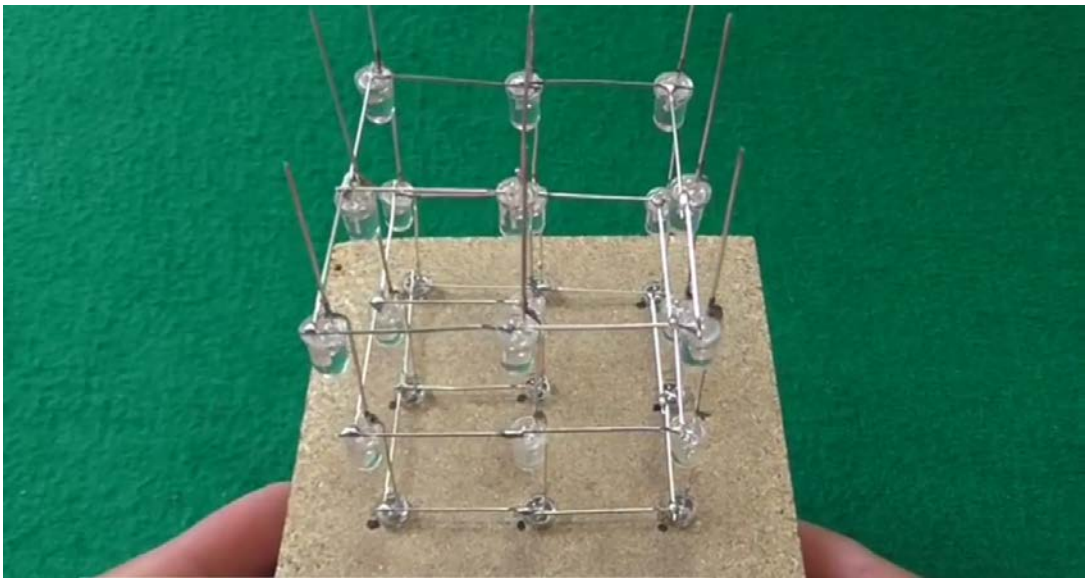
**Etape 1 : Appropriation du système :** (CUBE LED 3X3X3)

Identification des éléments nécessaires à la réalisation (étape déjà réalisé).

**Cube LED 3x3x3 :**

Le Cube LED que vous allez réaliser, utilise peu de composants par rapport à ce qu'on peut trouver dans d'autres systèmes plus complexe.
Pour la conception de ce cube, il faudra :

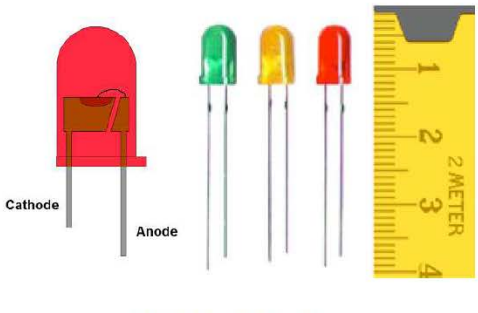
- Une carte arduino.
- 27 diodes électroluminescentes (leds), 12 résistances, 3 transistors, des câbles électriques.
- une  platine d'expérimentation (breadboard).
- Un support 9 trous.
- Un fer à souder
- Une alimentation.

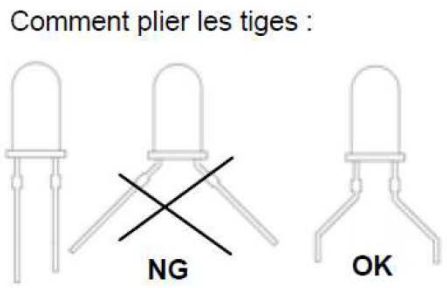


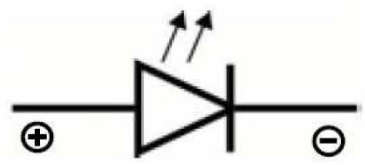
## 1- La diode LED

La diode électroluminescente (LED) émet de la lumière quand elle est traversée par un courant.
Elle est polarisée: La patte "+" est la plus longue, l'autre patte  (la plus courte) est la patte "-".

Les broches numériques de l'Arduino, lorsqu'elles sont configurées en sorties et qu'elles sont à l'état haut (1 logique) , fournissent une tension de 5 volts (supérieure à ce que peut accepter une LED). C'est pour cette raison que Les LED doivent donc être couplées en série avec une résistance.



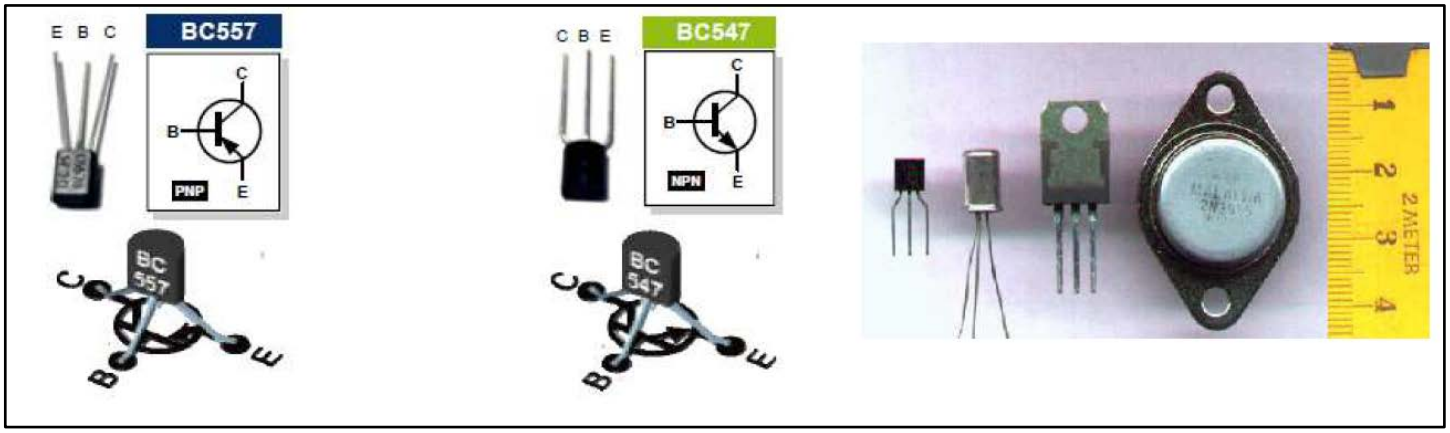



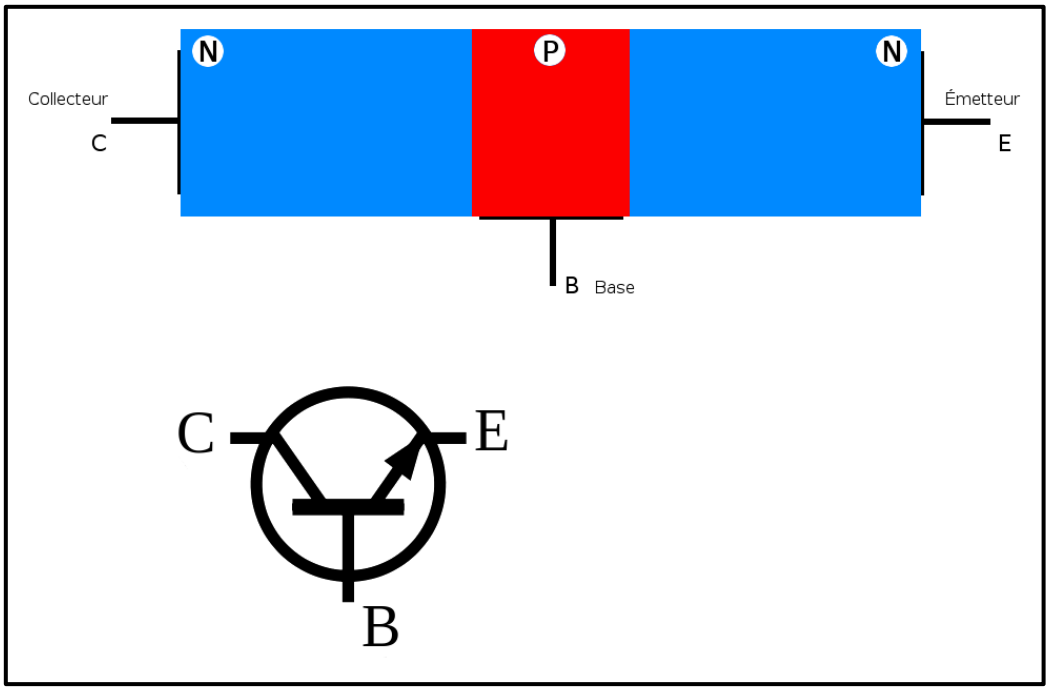



| Couleurs | Tension de seuil ou Vf | If (mA) | Longueur d'onde (nm) |
|---|---|---|---|
| Rouge | 1,6 à 2V | 6 à 20 mA | 650 à 660 nm |
| Jaune | 1,8 à 2V | 6 à 20 mA | 565 à 570 nm |
| Vert | 1,8 à 2V | 6 à 20 mA | 585 à 590 nm |
| Bleu | 2,7 à 3,2V | 6 à 20 mA | 470 nm |
| Blanc | 3,5 à 3,8V | 6 à 20 mA | |

## 2- Le transistor

Le transistor sert à amplifier un signal. Un faible courant de commande peut ainsi être transformé en un courant plus important. On distingue deux types de transistors, selon leur polarité. Le NPN et le PNP. Un transistor possède 3 pattes : la base (B), l'émetteur (E) et le collecteur (C).






Il existe une relation entre courant de base et courant collecteur due à l'effet transistor. Cette relation s'écrit :

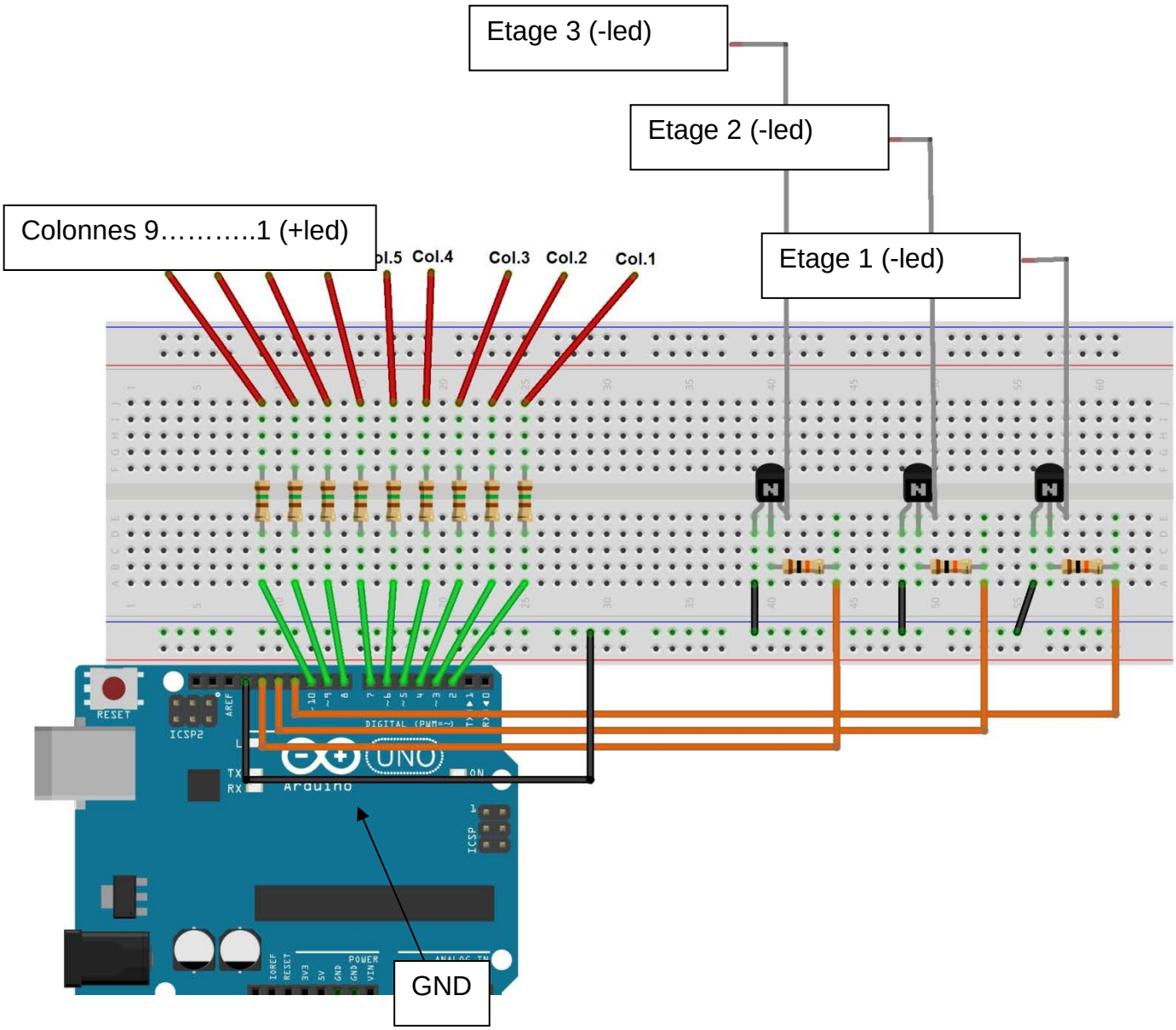
**IC = β x IB** (avec β = gain en courant du transistor)

**β** est parfois aussi appelé coefficient d'amplification statique en courant.

En règle générale **β** varie de 30 à 300 avec pour valeur courante :

- Transistors dit "Petit signaux" : **100 < β < 300**
- Transistors dit de "Puissance" : **30 < β < 100**

## Schéma du montage :



## 3- Réalisation d'un cube à LED :

Le cube est divisé en 3 niveaux de 9 Leds.

Tous les voyants alignés dans la même colonne (verticale) ont une anode commune (+).

Tous les voyants d'un même niveau (horizontal) ont une cathode commune (–).

## Préparation :

## Test des diodes :

Avant de commencer la soudure des diodes, et afin de s'assurer de son bon fonctionnement, et éviter Le dessoudage et le remplacement d'une diode défectueuse il faut tester chacune des diodes avant de l'utiliser.

Utiliser une pile de 3v entre les deux branches de la LED. L'anode (La patte la plus longue) sur la borne positive (+) et la cathode (la patte la plus courte sur la borne négatif (-)).

## Soudure :

### 1- Préparer le support

1. La soudure doit se faire sur un support en bois à 9 trous.

2. Après la fixation du LEDS sur le support, avant de commencer la soudure, il faut faire en sorte qu'elle y rentre légèrement pour qu'elle reste bien en place (Voir images)

3. Afin d'avoir un carré rigide, il faut ajoutez un fil de fer qui va relier les cathodes « – ») de toutes les diodes.

4. Soudé un autre fil de fer plus long (la longueur sera ajustée à la fin) afin de relier les cathodes de cet étage à la plaque électronique. (C'est la tige verticale sur Schéma du montage)

5. Faire 3 fois la même chose (pour les 3 étages) en n'oubliant pas de décaler à chaque fois la tige verticale d'une LED. Voir image avec deux étages assemblés (image 5). Les deux tiges verticales sont en décalées.

6. Bien faire les 3 étages de la même façon afin de faciliter l'assemblage des 3 étages. Sur la photo, vous avez un aperçu de la façon d'assembler.

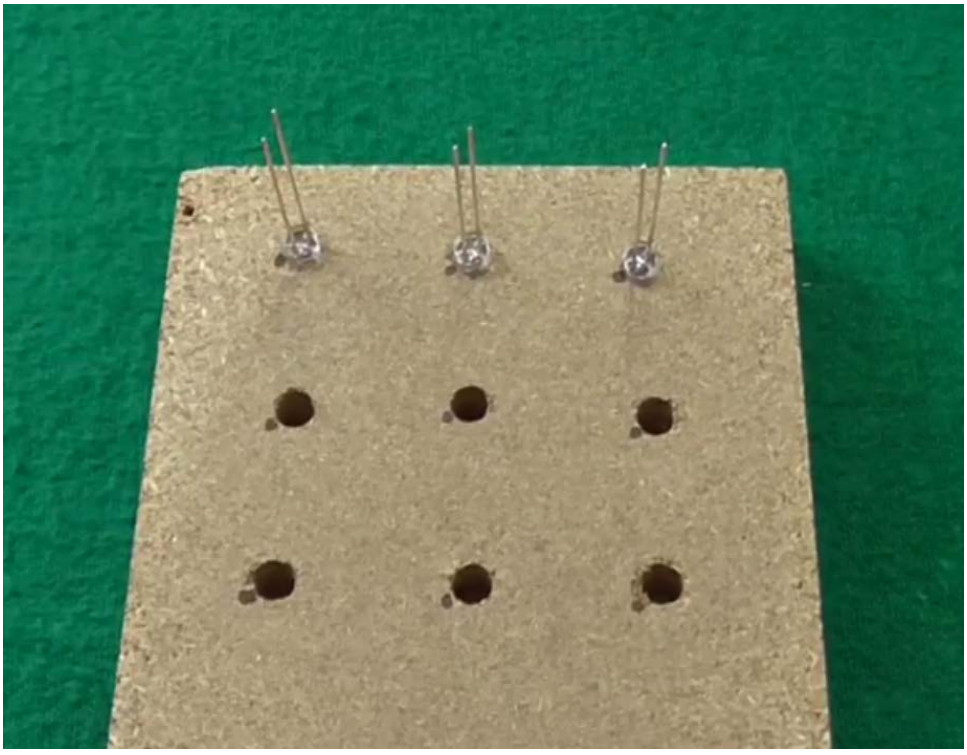


**Image 1**

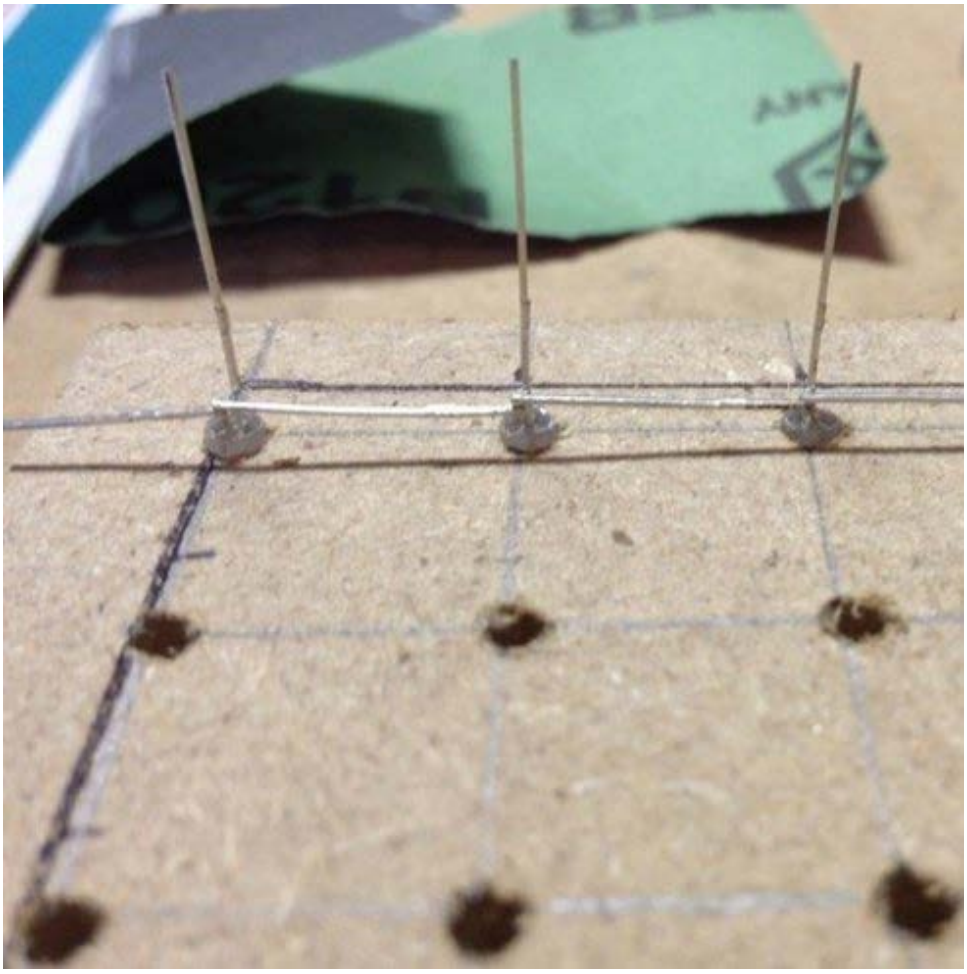


**Image 2**

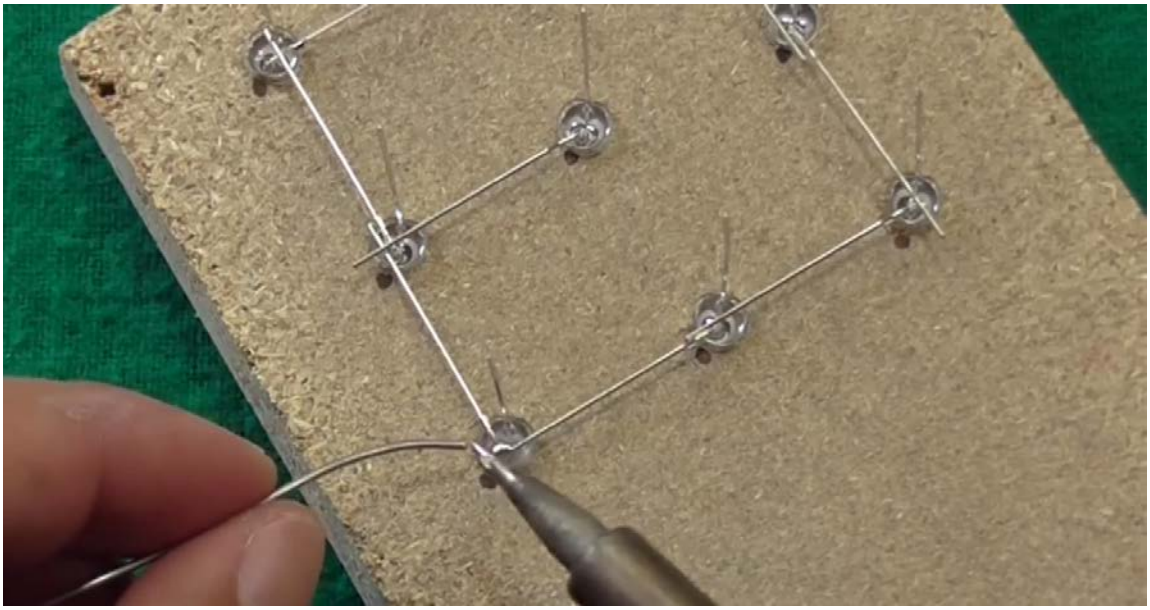


**Image 3**

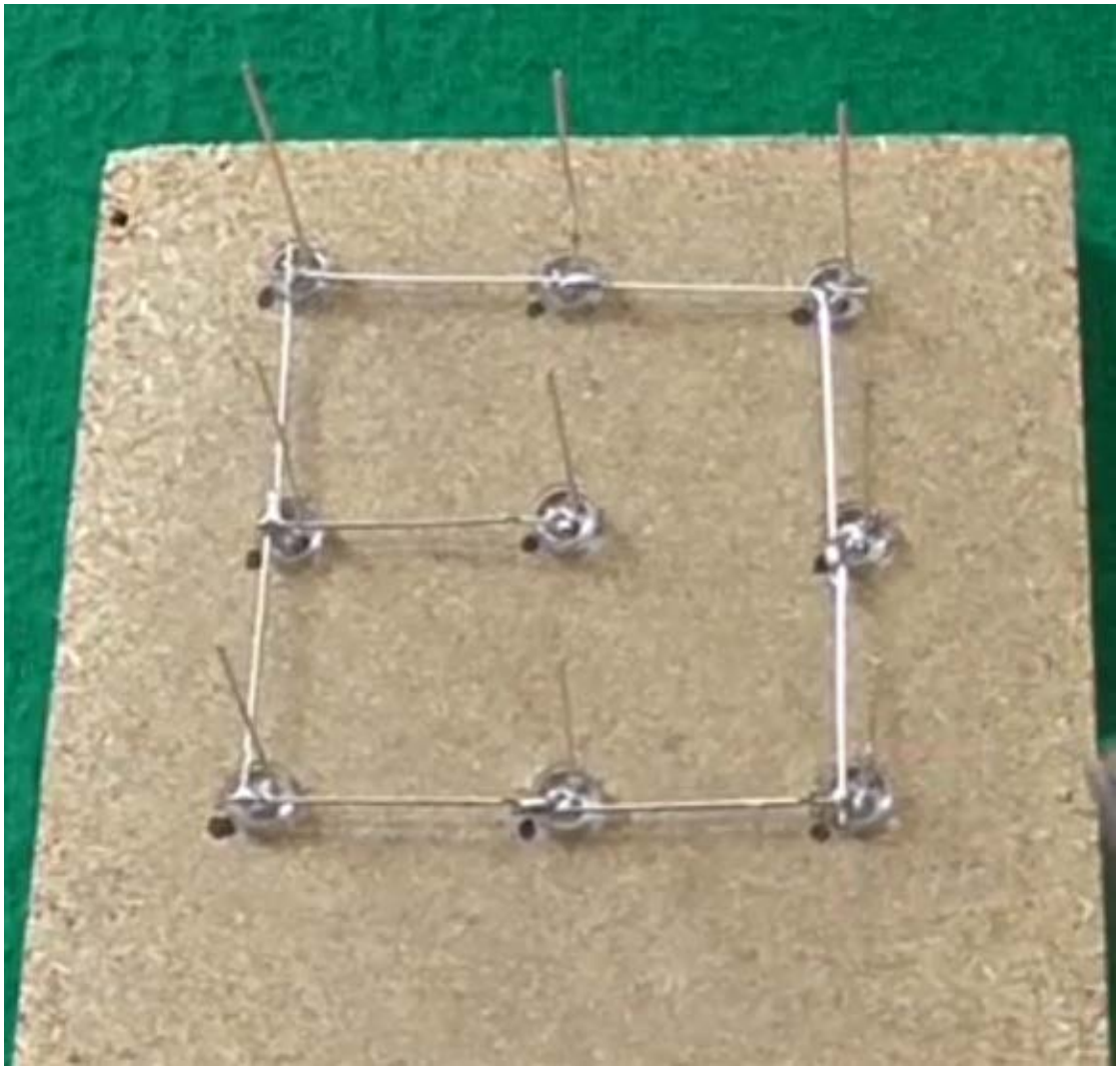


**Image 4**

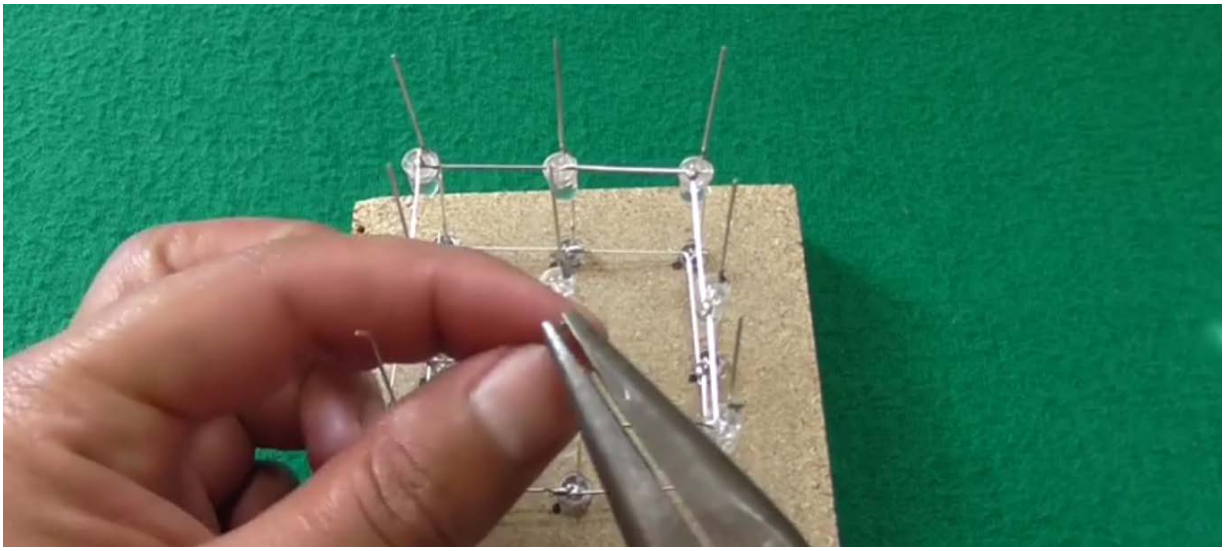


**Image 5**

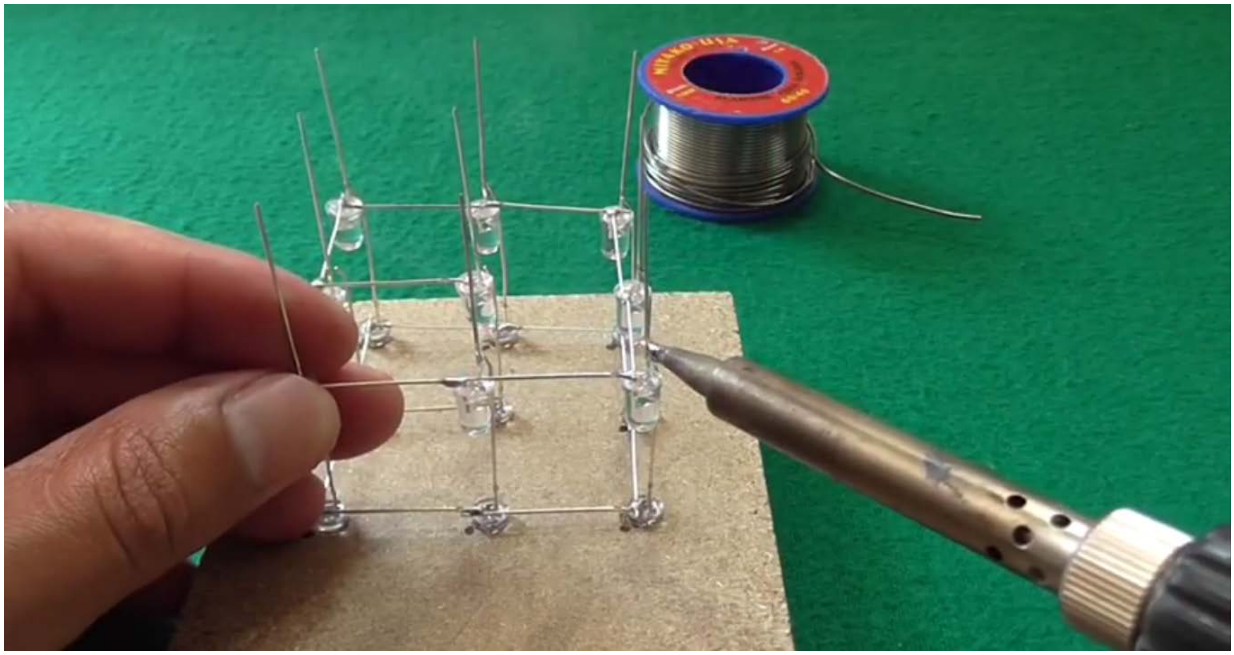


**Image 6**

## 2- Câblages : (voir schéma du montage page5)

Chacune des 9 colonnes du cube led se connectera à une broche sur l'Arduino à travers une résistance (220 ohms) de limitation de courant.

Chacun des trois étages du cube led se connectera à une broche sur l'Arduino à travers un transistor et une résistance.

Le cube est connecté de la manière suivante :

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Colonne :** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | **Etage :** | E1 | E2 | E3 |
| **Broches Arduino :** | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | 11 | 12 | 13 |

## 3- programmation

Pour allumer votre lampe LEDs cube 3x3x3, il vous faut mettre un programme dans votre carte arduino.
Premièrement, il faut allumer une colonne ou un étage.

```
/*
    ledcube.pde - Example sketch for controlling an LED cube.

*/

#include <LedCube.h>

#define SIZE 3
#define COLS (SIZE*SIZE)

byte levelPins[SIZE] = {11,12,13};
byte colPins[COLS] = {2,3,4,5,6,7,8,9,10};

LedCube cube(SIZE, levelPins, colPins);

//#define DEBUG
#ifdef DEBUG
#include <memdebug.h>
void showmem(const char label[] = "")
{
  char buffer[100];

  sprintf(buffer,"%s: %04u %04u %04u : used/free",
      label,
      getMemoryUsed(),
      getFreeMemory()
    );

  Serial.println(buffer);
}
#endif

void setup ()
{
#ifdef DEBUG
    Serial.begin(9600);
#endif
}

void loop ()
{
    delay(10);

#ifdef DEBUG
    showmem("start");
#endif
    cubeFrame* f[] = {
        cube.createFrame((byte[]) {0,6, 1,6, 2,6}, 6, 80),
        cube.createFrame((byte[]) {0,7, 1,7, 2,7}, 6, 70),
        cube.createFrame((byte[]) {0,8, 1,8, 2,8}, 6, 60),
```

```
      cube.createFrame((byte[]) {0,5, 1,5, 2,5}, 6, 50),
        cube.createFrame((byte[]) {0,2, 1,2, 2,2}, 6, 40),
        cube.createFrame((byte[]) {0,1, 1,1, 2,1}, 6, 30),
        cube.createFrame((byte[]) {0,0, 1,0, 2,0}, 6, 20),
        cube.createFrame((byte[]) {0,3, 1,3, 2,3}, 6, 10)
    };
#ifdef DEBUG
    showmem("before free");
#endif
    cube.lightFrames(f, 8);

    // light each light one at a time
    for(byte level=0; level<cube.getLevels(); level++)
    {
        for(byte col=0; col<cube.getCols(); col++)
        {
            cube.lightPulse(level, col, 100);
        }
    }
    // light one level at a time, increasing speed each time
    for(byte d=25; d>2; d-=2)
    {
        for(byte l=1; l <= cube.getLevels(); l++)
        {
            cube.lightLevel(l, d);
        }
    }

    // light each row on each level
    for(byte level=1; level<=cube.getLevels(); level++)
    {
        for(byte row=1; row<=cube.getLevels()*2; row++)
        {
            cube.lightRow(row, level);
        }
    }

    // light each plane
    for(byte i=3; i; i--)
    {
        for(byte row=1; row<=cube.getLevels()*2; row++)
        {
            cube.lightPlane(row, 10*i);
        }
    }

    // single random light at a time
    cube.randomLight(random(25,100),100);

    // random column drop
    for(byte x=0; x<=15; x++)
    {
        cube.lightDrop(random(0,cube.getCols()), random(50,150));
    }

    // circle around cube at a random level
    for(byte x=0; x<=5; x++)
    {
        cube.lightPerimeter(random(0,cube.getLevels()), random(1,5), random(25,100));
    }
```

```
    // light each face
    byte planes[] = {cube.getLevels()+1,cube.getLevels(),cube.getLevels()*2,1};
    for(byte i=5; i; i--)
    {
        for(byte p=0; p<sizeof(planes); p++)
        {
            cube.lightPlane(planes[p], 5*i);
        }
    }

    // random columns
    cube.randomColumn(25);

    // turn off a single column randomly
    cube.enableBuffer();
    for(byte c=0; c<30; c++)
    {
        cube.fillBuffer();
        cube.invertBuffer();
        cube.randomColumn();
        cube.drawBuffer(7);
    }
    cube.enableBuffer(false);

    // cols in and out
    for(byte c=1, d=0; c<=10; c++)
    {
        if(c%2 == 0)
        {
            for(d=0; d<20; d++)
            {
                cube.lightColumn(2,1);
                cube.lightColumn(4,1);
                cube.lightColumn(6,1);
                cube.lightColumn(8,1);
            }
        }
        else if(c%4 == 1)
        {
            for(d=0; d<30; d++)
            {
                cube.lightColumn(1,1);
                cube.lightColumn(3,1);
                cube.lightColumn(7,1);
                cube.lightColumn(9,1);
            }
        }
        else
        {
            for(d=0; d<70; d++)
            {
                cube.lightColumn(5,1);
            }
        }
    }

    // diamond and box
    byte diamond[] = {0,4,  1,1, 1,3, 1,4, 1,5, 1,7,   2,4};
    byte box[] = {
        2,0, 2,1, 2,2, 2,3, 2,5, 2,6, 2,7, 2,8,
```

```
        1,0, 1,2, 1,6, 1,8,
        0,0, 0,1, 0,2, 0,3, 0,5, 0,6, 0,7, 0,8
    };
    cube.lightSequence(box, sizeof(box), 200);
    cube.lightSequence(diamond, sizeof(diamond), 400);

    // helicopter effect
    byte topSeq[8] = {0,3,6,7,8,5,2,1};
    byte botSeq[8] = {8,5,2,1,0,3,6,7};
    for(byte loops = 0, delay = 50; loops<=8; loops++)
    {
        for(byte s=0; s<8; s++)
        {
            byte seq[] = {2,topSeq[s], 1,4, 0,botSeq[s]};
            cube.lightSequence(seq, sizeof(seq), delay);
        }
        if(loops < 5) delay-=10;  else delay += 10;
    }

    // turn off one light at a time
    cube.lightsOut();
}
```

## Conclusion :