Compteur Beursault

Compagnie Arc
1873
ROMAINVILLE

Carte contrôleur

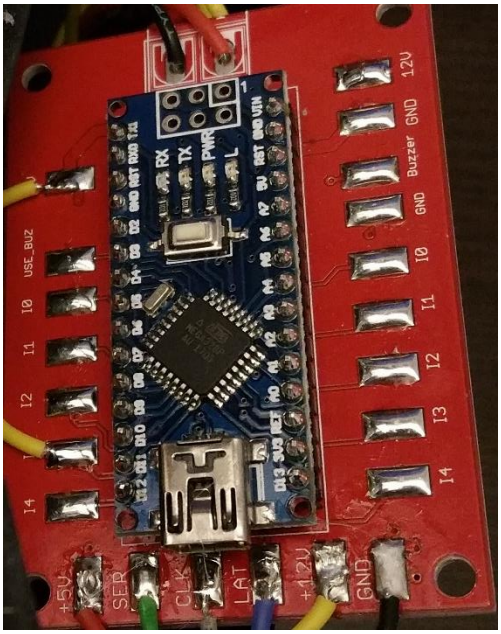
Matériel nécessaire

- Arduino nano
- PCB d'interface (fabrication Antoine)
- Résistance de 10k CMS 0804

Fabrication

Souder les résistances qui seront présente sous la carte Arduino.

Souder la carte Arduino sur le PCB d'interface. Le connecteur USB se trouve au niveau des pads de soudure pour la carte Sparkfun (donc à l'opposé du connecteur 12V !)



Étamer les différents plots de soudure pour faciliter le travail futur (et protéger les pads)

Flasher le code sur la carte Arduino grâce au connecteur USB. Celui-ci est présent en annexe au besoin.

Afficheur

Matériel nécessaire

- Boitier plastique (ou autre) : 150x100x55mm minimum afin de pouvoir tout faire rentrer dedans ➔ *8€ environ fdpin*
- Afficheur 7 segment 12V : par exemple le CM1-4002LR00-W (disponible sur ebay en cherchant « big 7 segment ») ➔ *10€ environ fdpin*
- Drivers 7 segment 12V (Sparkfun): Breakout Board TPIC6C596 (disponible sur ebay en cherchant “7 segment driver 12v”) ➔ *2.18€*
- Convertisseur 220V 12V (alimentation de LED) 12w min ➔ *5€*
- Prise Molex 6 points (et sa pince à sertir)

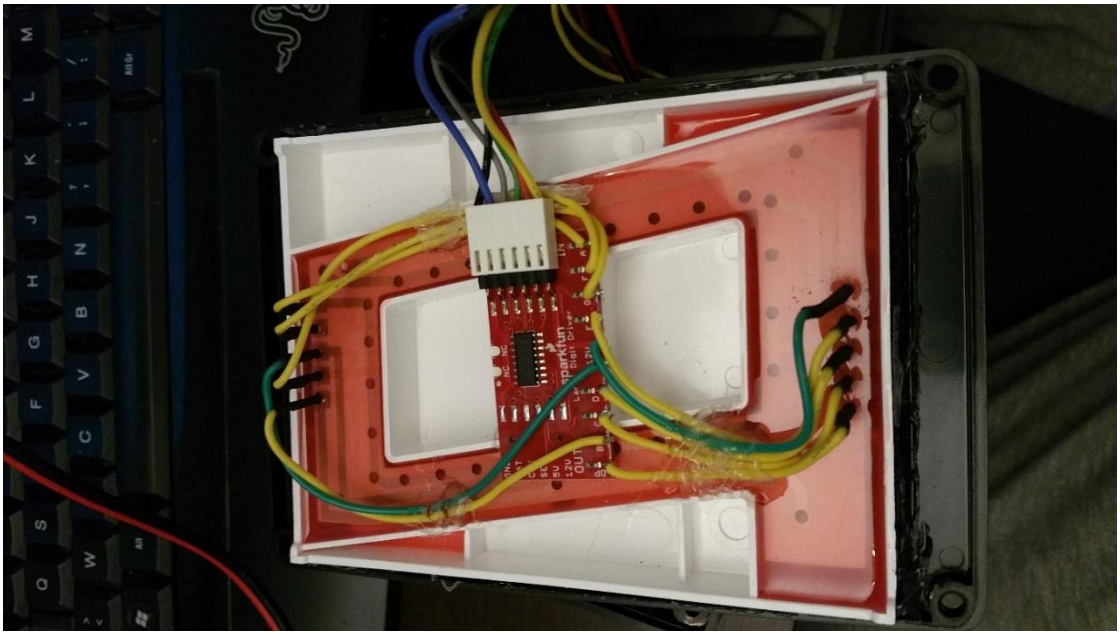
Fabrication

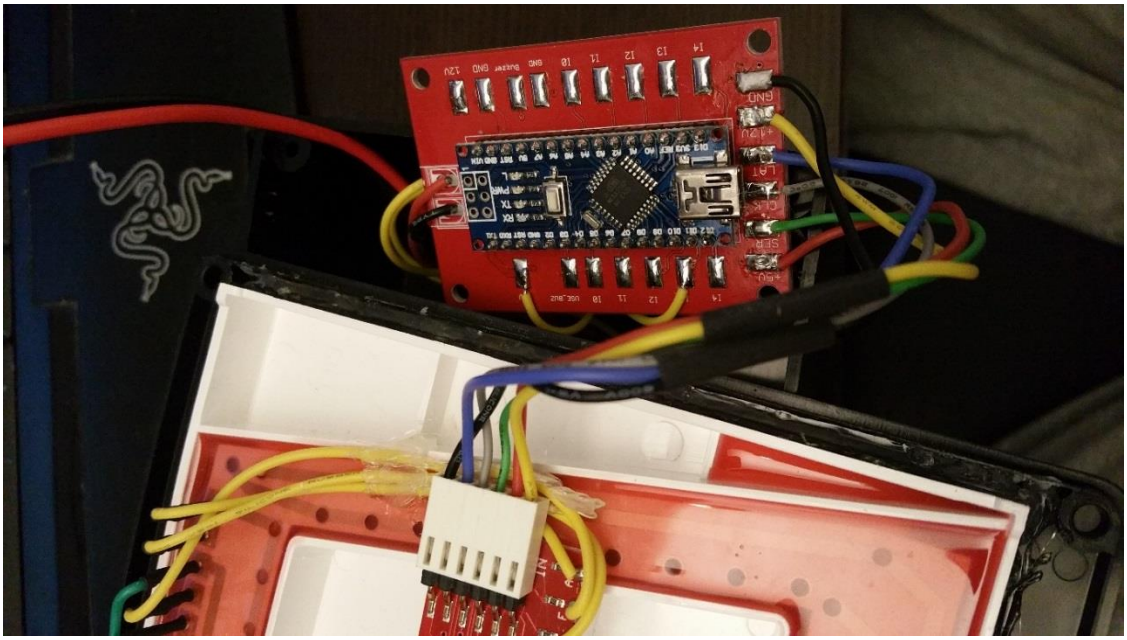
Découper le boitier plastique afin de faire ressortir l’afficheur 7 segments. Fixer l’afficheur avec de la colle chaude afin d’assurer un minimum d’étanchéité. Une fois ceci fait, coller le driver 7 segment au dos de l’afficheur, les pads de soudure vers le bas de l’afficheur (le point est vers le bas ☺)

Utiliser des fils silicone afin d’éviter toute cassure sur le long terme. Dénuder et souder les fils sur le boitier. Protéger les soudures avec de la gaine thermo rétractable. Une fois tous les fils parfaitement préparés, souder les fils au drivers Sparkfun en suivant ce câblage (la datasheet est en annexe).

Câblage 7 segment (Le 1 est en bas à droite et le 6 en haut à gauche) :

1. 12v
2. E
3. D
4. C
5. DP
6. B
7. A
8. 12v
9. F
10. G





Dessouder le connecteur OUT de la carte Sparkfun pour éviter tout faux contact.

Câblage Connecteur de la carte Drivers LED (Sparkfun) :

1. GND
2. LAT
3. CLK
4. SER
5. +5V
6. +12V

Télécommande

Matériel nécessaire

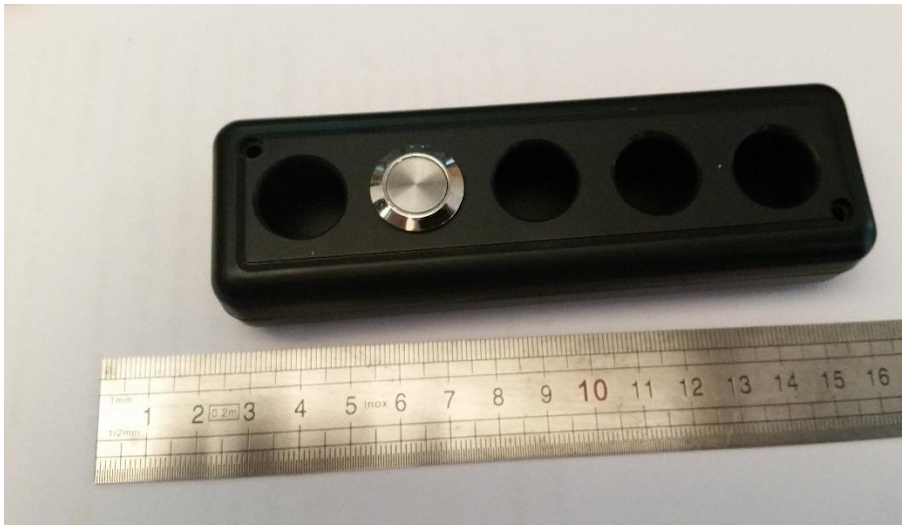
- Boitier plastique aux dimensions : 129X40X25.5MM (disponible sur ebay en cherchant « boitier long noir 129 ») ➔ *15€ fdpin*
- 5 boutons anti-vandalisme 16mm (disponible sur ebay en cherchant « 16mm anti vandal ») ➔ *12€ fdpin*

Fabrication

Pour placer 5 boutons à distance équivalentes, il faut faire un 1er trou à 11.5mm du tour intérieur puis tous les autres espacés de 23 mm

Trous de 16mm de diamètre pour bouton anti vandalisme

Avec un bouton



Pour les boutons, toutes les pattes d'un même coté sont relié au 5V. Les autres en fonction de leurs positions. Afin d'éviter tout soucis dans le temps, ne pas hésiter à souder les câbles aux connecteurs à vis. De toute façon il n'y a aucune raison pour réutiliser ces boutons (Attention à ne souder qu'une fois tout le câblage vérifié !)

Câblage

Matériel nécessaire

Câble réseau de 100m (ou 75m si disponible) permettant d'espacer 2 compteur de 50m ou plus et d'utiliser le reste pour câbler les télécommandes

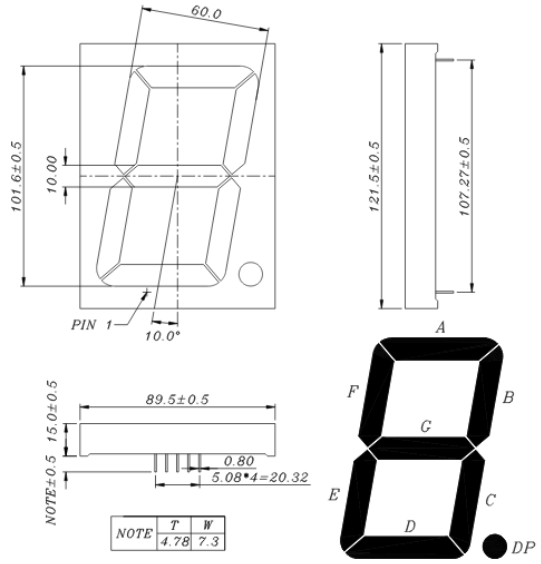
Fabrication

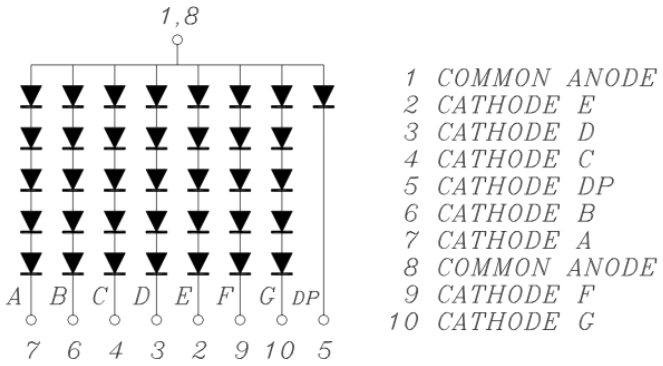
Câblage du câble Ethernet pour relier 2 boitier ensemble ou le boitier avec sa télécommande :

Blanc/Orange : 5V
Orange : 12V
Bleu : 0
Blanc/Bleu : 1
Vert : 2
Blanc/Vert : 3
Blanc/Marron : 4
Marron : GND

Part No. : CM1-4002LR00

Package Dimensions





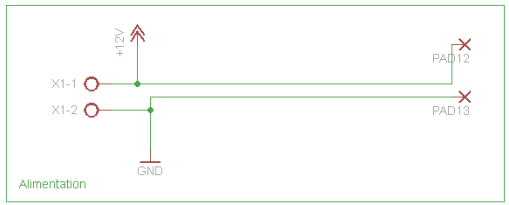
Electrical / Optical Characteristics at TA=25℃

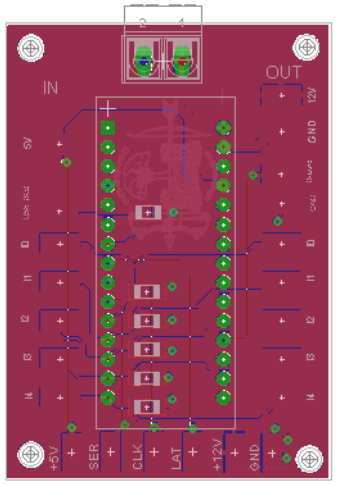
Parameter	Symbol	Min.	Typ.	Max.	Unit	Test Condition
Average Luminous Intensity	IV	30	55		mcd	IF = 20mA
Peak Emission Wavelength	λP		650		nm	IF = 20mA
Dominant Wavelength	λd		638		nm	IF = 20mA
Spectral Line Half-Width	Δλ		30		nm	IF = 20mA
Forward Voltage, any Segment or D..P.	VF		9.5	12	V	IF = 20mA
	VF(DP)		1.9	2.4		
Reverse Current, any Segment or D..P	IR			100	μA	VR = 5V
Luminous Intensity Matching Ratio	Iv-m			2:1		IF = 20mA

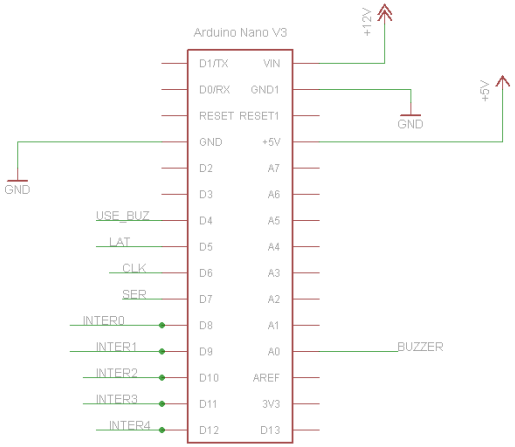
Absolute Maximum Ratings at TA=25℃

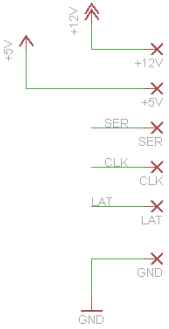
Parameter	Maximum Rating	Unit
Power Dissipation	45	mW
Peak Forward Current (1/10 Duty Cycle, 0.1ms Pulse Width)	60	mA
Continuous Forward Current	20	mA
Reverse Voltage	5	V
Operating Temperature Range	-20℃ to + 80℃	
Storage Temperature Range	-35℃ to + 100℃	
Lead Soldering Temperature [4.0mm(.157") From Body]	260℃ for 5 Seconds	
Reflow Soldering	NO	

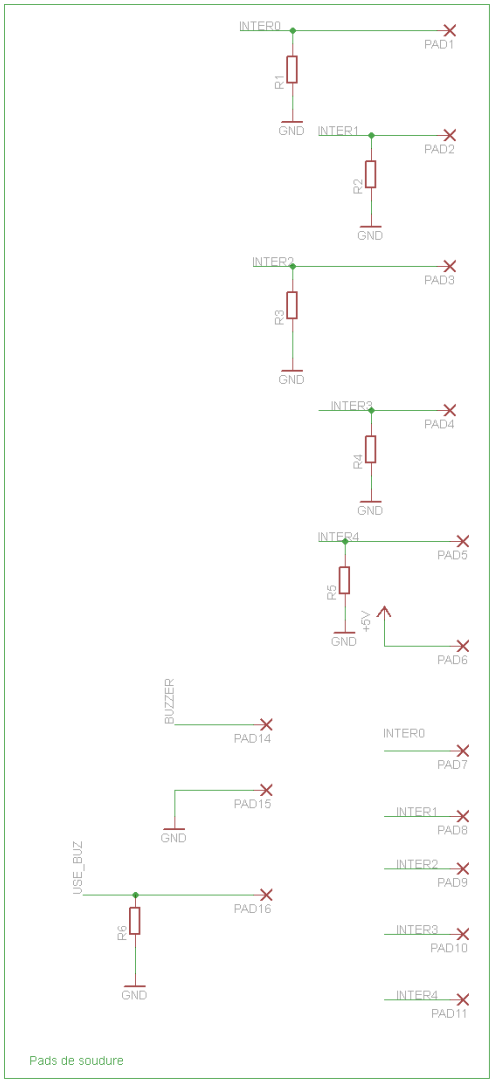
TYPICAL ELECTRON-OPTICAL CHARACTERISTIC CURVES
25°C Free Air Temperature Unless Otherwise Specified











scoreur	
30/05/2018 09:45:21	
Sheet: 1/1	

CODE ARDUINO

```
//GPIO declarations
//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
byte segmentClock = 6;
byte segmentLatch = 5;
byte segmentData = 7;
byte btn0 = 8;
byte btn1 = 9;
byte btn2 = 10;
byte btn3 = 11;
byte btn4 = 12;
byte btnBuzz = 4;
byte buzzer = A0;
//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

int currentScore = 9;
unsigned long time = 0;
#define MAX_TIME_AFFICHE 5*1000

void buzz(int nb)
{
 if(digitalRead(btnBuzz) == HIGH)
 {
  for(int i=0;i<nb;i++)
  {
   digitalWrite(buzzer,HIGH);
   delay(500);
   digitalWrite(buzzer,LOW);
   delay(500);
  }
 }
}

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

 pinMode(segmentClock, OUTPUT);
 pinMode(segmentData, OUTPUT);
 pinMode(segmentLatch, OUTPUT);

 pinMode(btn0, INPUT);
 pinMode(btn1, INPUT);
 pinMode(btn2, INPUT);
 pinMode(btn3, INPUT);
 pinMode(btn4, INPUT);
 pinMode(btnBuzz, INPUT);
 pinMode(buzzer, OUTPUT);
 digitalWrite(buzzer,LOW);

 digitalWrite(segmentClock, LOW);
 digitalWrite(segmentData, LOW);
 digitalWrite(segmentLatch, LOW);

 for(int i=4;i>=0;i--)
 {
  postNumber(i, false);
  digitalWrite(segmentLatch, LOW);
  digitalWrite(segmentLatch, HIGH); //Register moves storage register on the rising edge of RCK
  delay(500);
 }
}
```

```
void loop()
{
   byte needBuzz = 0;
   if(digitalRead(btn0) == HIGH)
   {
    time = millis();
    currentScore = 0;
    needBuzz = 1;
   }
   if(digitalRead(btn1) == HIGH)
   {
    time = millis();
    currentScore = 1;
    needBuzz = 1;
   }
   if(digitalRead(btn2) == HIGH)
   {
    time = millis();
    currentScore = 2;
    needBuzz = 1;
   }
   if(digitalRead(btn3) == HIGH)
   {
    time = millis();
    currentScore = 3;
    needBuzz = 1;
   }
   if(digitalRead(btn4) == HIGH)
   {
    time = millis();
    currentScore = 4;
    needBuzz = 1;
   }

   if(millis() < time+MAX_TIME_AFFICHE)
   {
    if(currentScore <=4)
      postNumber(currentScore, false);
   }
   else
   {
     Serial.println("TimeMax");
     for (byte x = 0 ; x < 8 ; x++)
     {
       digitalWrite(segmentClock, LOW);
       digitalWrite(segmentData, 0);
       digitalWrite(segmentClock, HIGH); //Data transfers to the register on the rising edge of SRCK
     }
   }
   digitalWrite(segmentLatch, LOW);
   digitalWrite(segmentLatch, HIGH); //Register moves storage register on the rising edge of RCK

   if(needBuzz == 1)
   {
    buzz(currentScore);
    needBuzz = 0;
   }

   delay(10);
}


//Takes a number and displays 2 numbers. Displays absolute value (no negatives)
```

```
void showNumber(float value)
{
  int number = abs(value); //Remove negative signs and any decimals

  for (byte x = 0 ; x < 2 ; x++)
  {
    int remainder = number % 10;
    postNumber(remainder, false);
    number /= 10;
  }

  //Latch the current segment data
  digitalWrite(segmentLatch, LOW);
  digitalWrite(segmentLatch, HIGH); //Register moves storage register on the rising edge of RCK
}

//Given a number, or '-', shifts it out to the display
void postNumber(byte number, boolean decimal)
{
  //    -  A
  //   / / F/B
  //    -  G
  //   / / E/C
  //    -. D/DP

#define a  1<<0
#define b  1<<6
#define c  1<<5
#define d  1<<4
#define e  1<<3
#define f  1<<1
#define g  1<<2
#define dp 1<<7
byte segments;

  switch (number)
  {
    case 1: segments = b | c; break;
    case 2: segments = a | b | d | e | g; break;
    case 3: segments = a | b | c | d | g; break;
    case 4: segments = f | g | b | c; break;
    case 5: segments = a | f | g | c | d; break;
    case 6: segments = a | f | g | e | c | d; break;
    case 7: segments = a | b | c; break;
    case 8: segments = a | b | c | d | e | f | g; break;
    case 9: segments = a | b | c | d | f | g; break;
    case 0: segments = a | b | c | d | e | f; break;
    case ' ': segments = 0; break;
    case 'c': segments = g | e | d; break;
    case '-': segments = g; break;
  }

  if (decimal) segments |= dp;
  //Clock these bits out to the drivers
  for (byte x = 0 ; x < 8 ; x++)
  {
    digitalWrite(segmentClock, LOW);
    digitalWrite(segmentData, segments & 1 << (7 - x));
    digitalWrite(segmentClock, HIGH); //Data transfers to the register on the rising edge of SRCK
  }
}
```