



Enseignes et afficheurs à LED

Rubans de LED

Rubans de LED

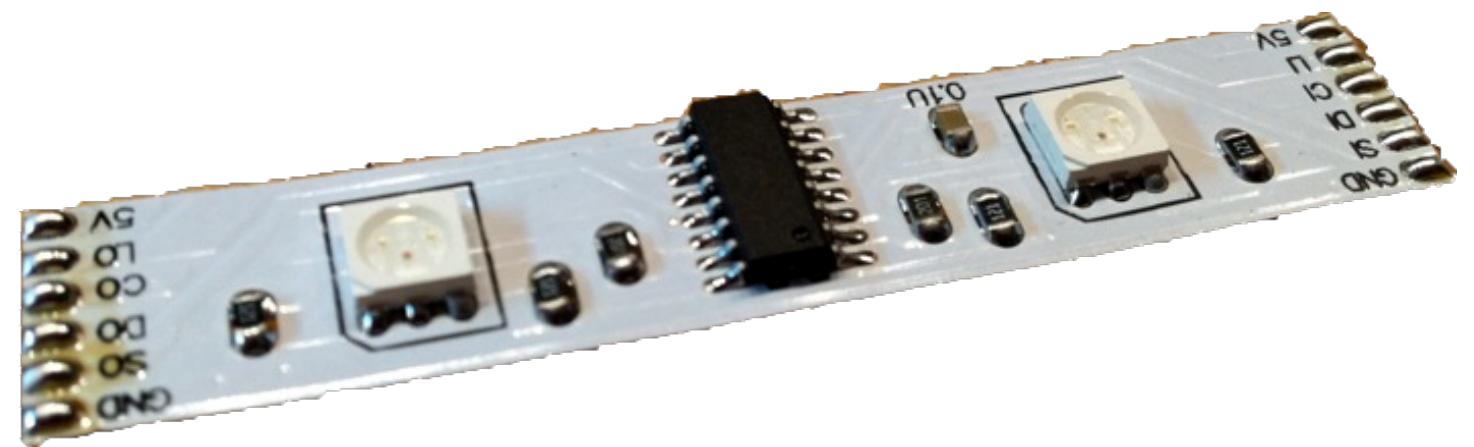
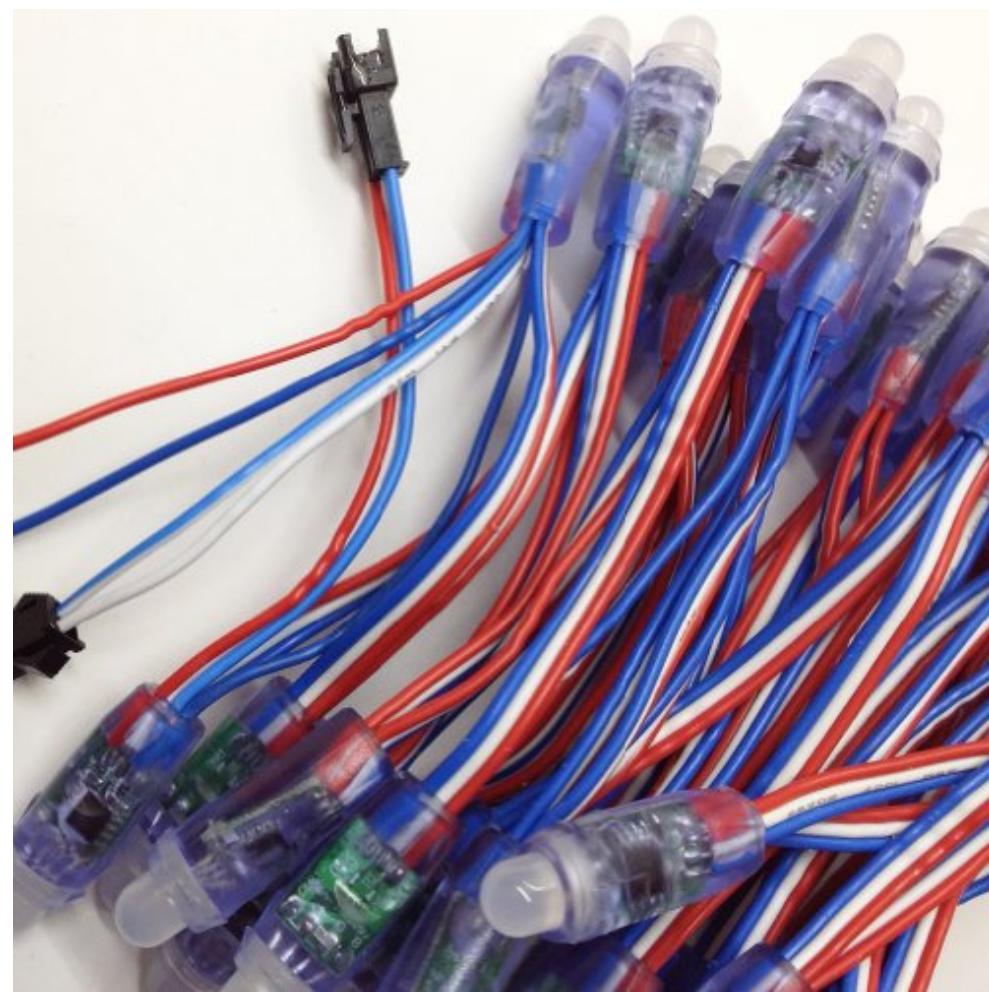


Pierre-Yves Rochat

- Rubans uniformes
- Rubans addressables
- Signaux de commande
- Programmation

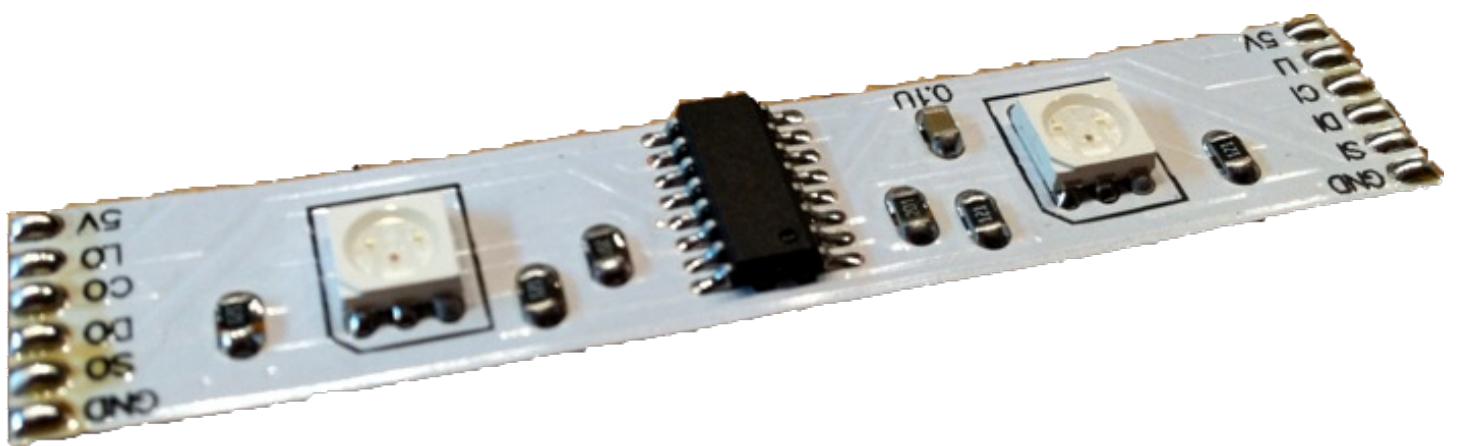
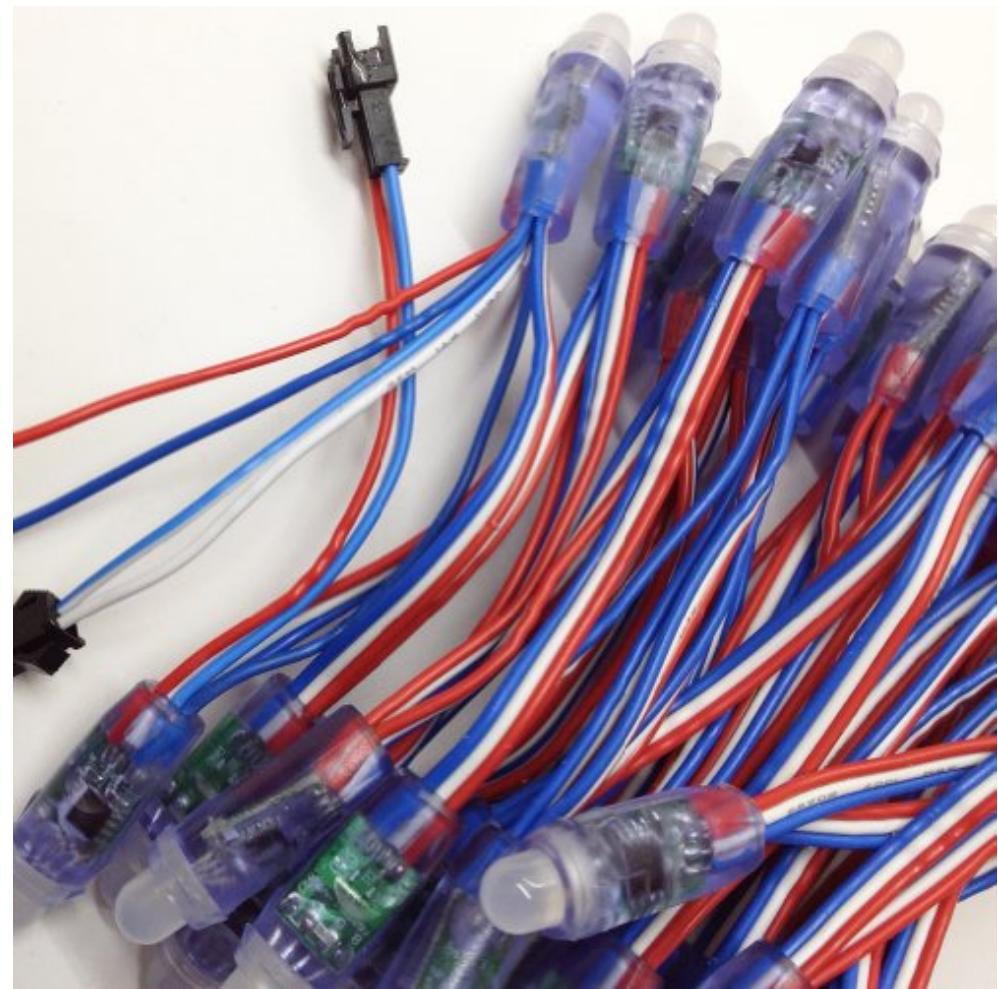
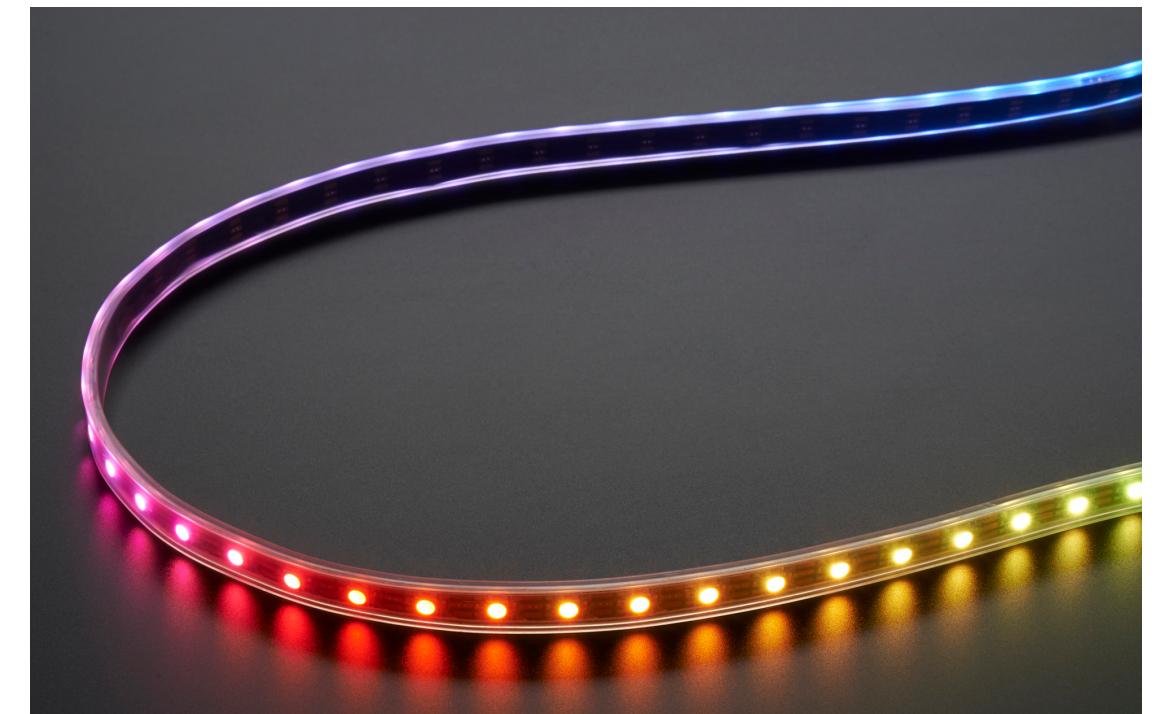
Plusieurs sortes de rubans de LED

- Des formes très différentes



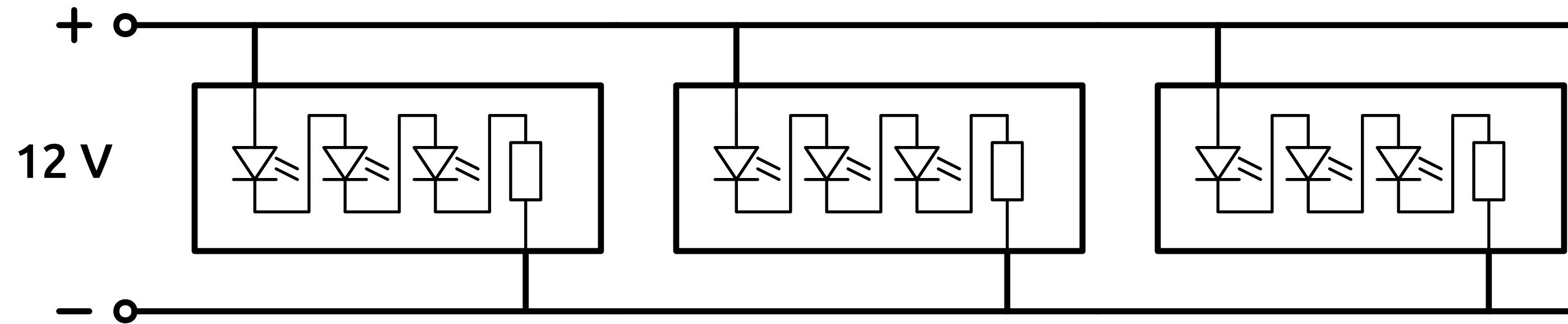
Plusieurs sortes de rubans de LED

- Des formes très différentes
- Uniformes ou adressables



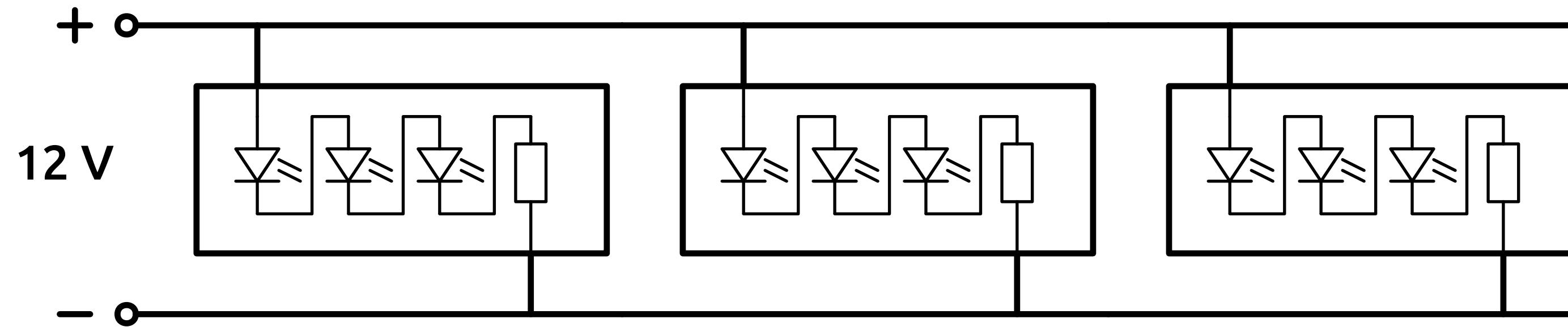
Rubans uniformes

- une seule couleur et une seule intensité à un instant donné, pour toutes les LED

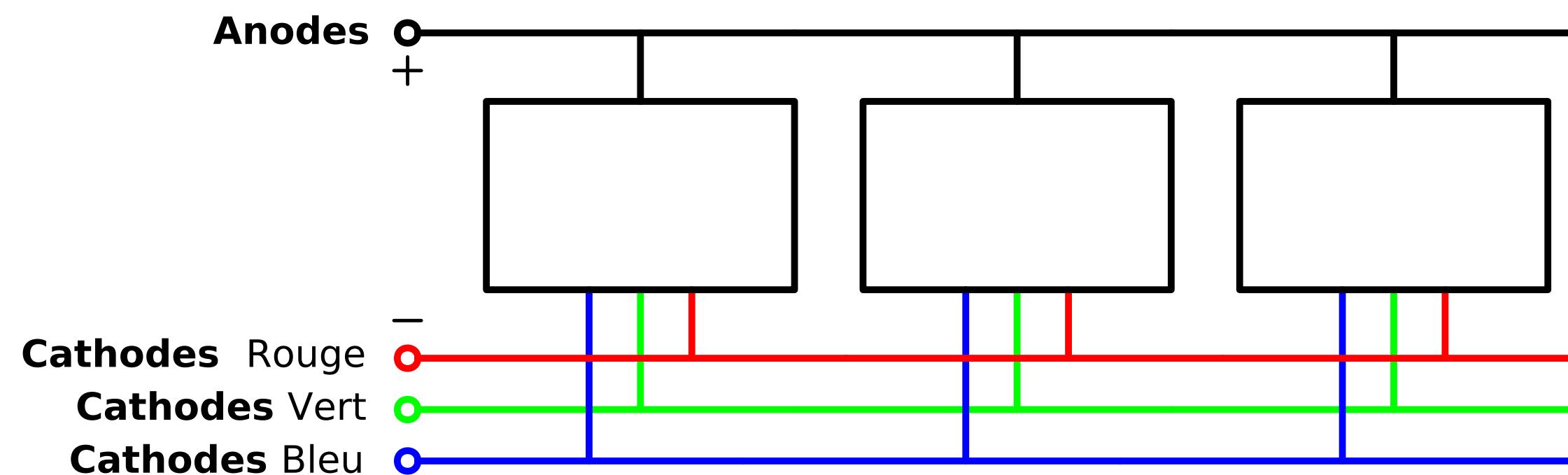


Rubans uniformes

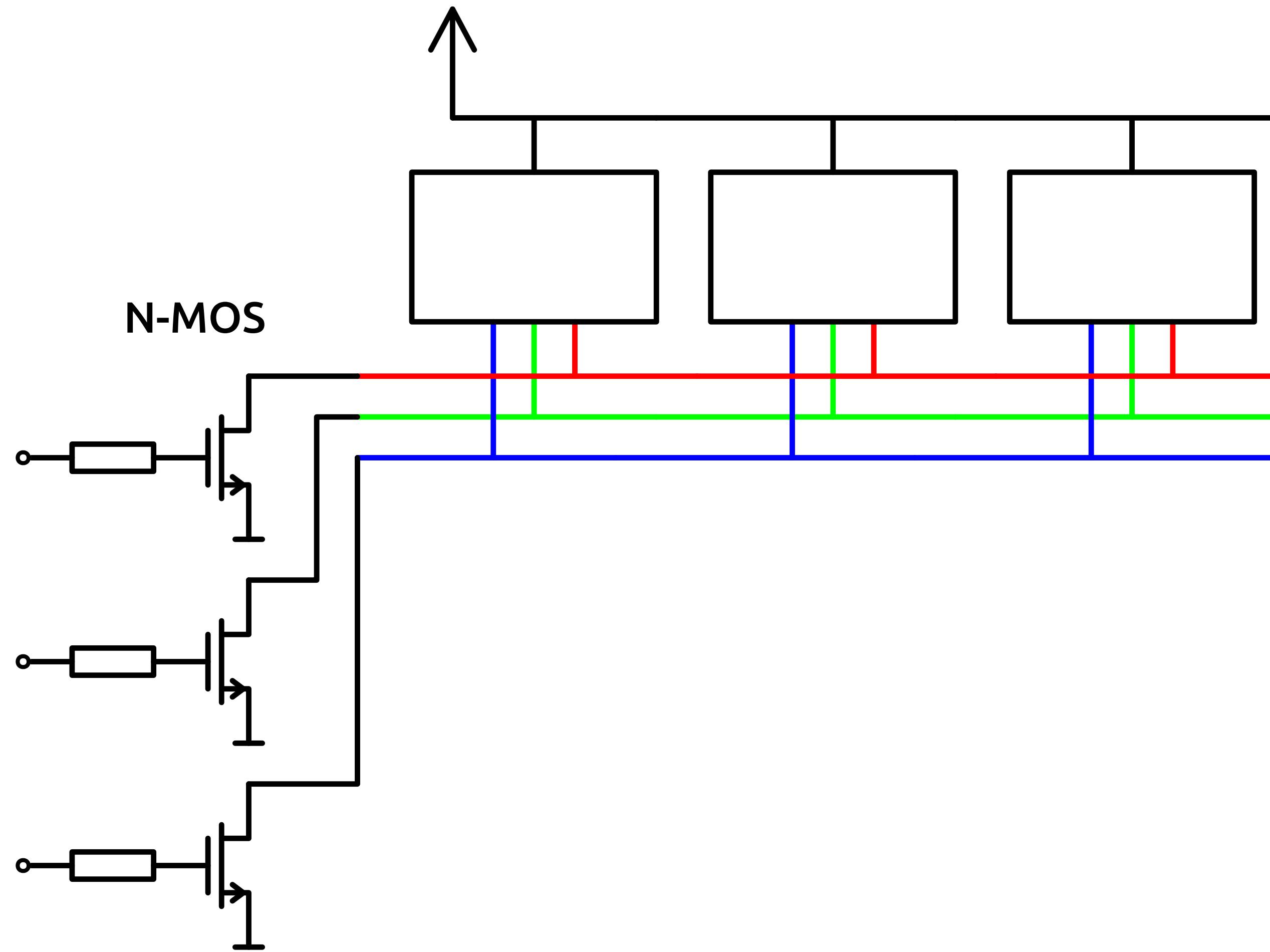
- une seule couleur et une seule intensité à un instant donné, pour toutes les LED



- monochrome ou RGB (Rouge Vert Bleu)



Commande par transistors N-MOS



Rubans adressables

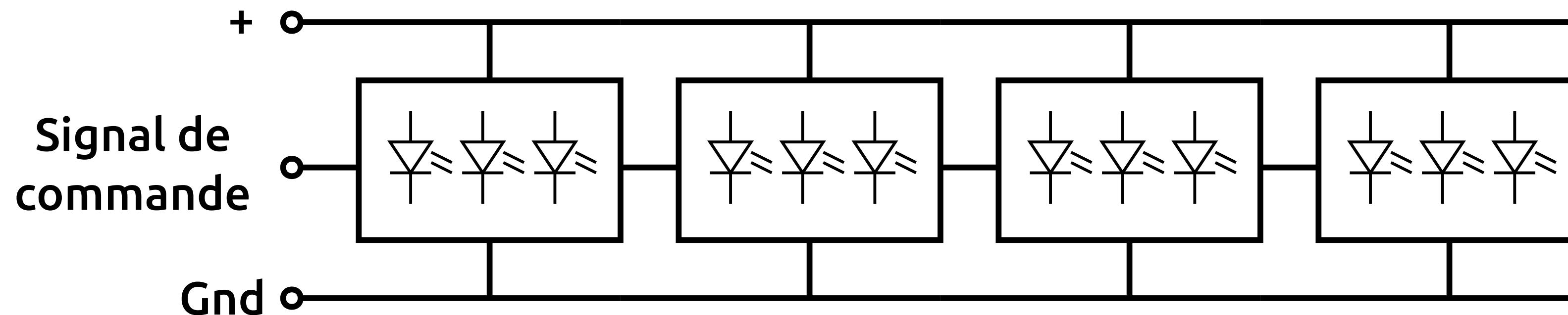
- Chaque LED est indépendante pour sa couleur et son intensité

Rubans adressables

- Chaque LED est indépendante pour sa couleur et son intensité
- Rubans adressables (*Addressable strips*)

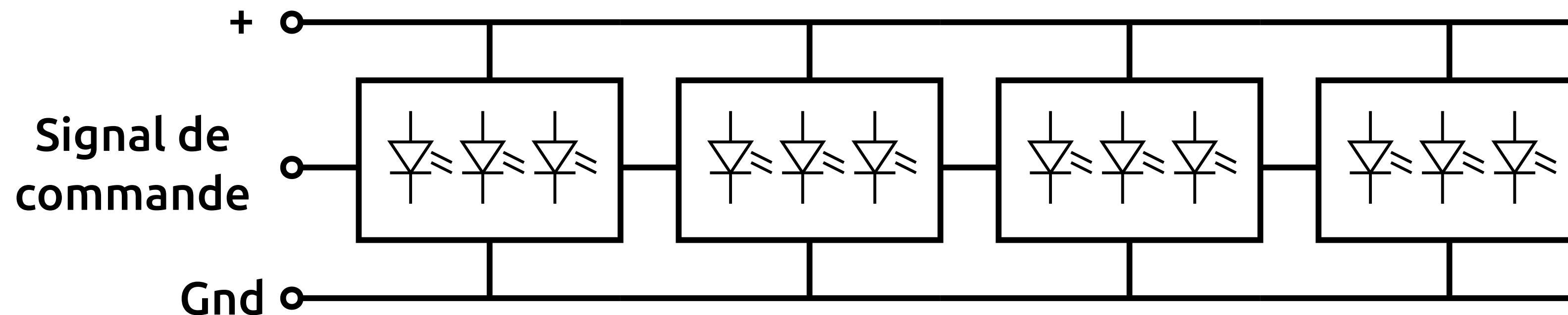
Rubans adressables

- Chaque LED est indépendante pour sa couleur et son intensité
- Rubans adressables (*Addressable strips*)

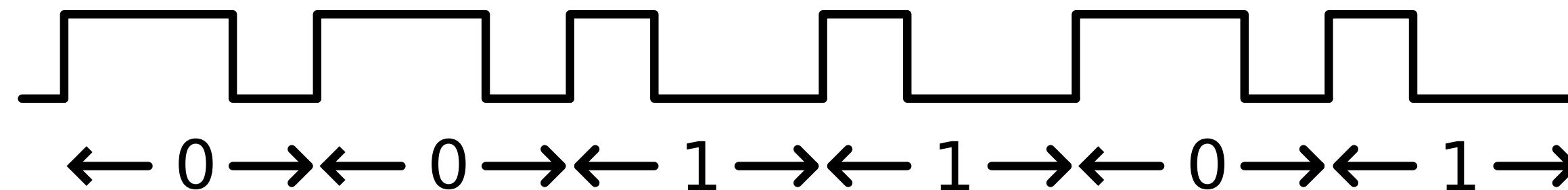


Rubans adressables

- Chaque LED est indépendante pour sa couleur et son intensité
- Rubans adressables (*Addressable strips*)

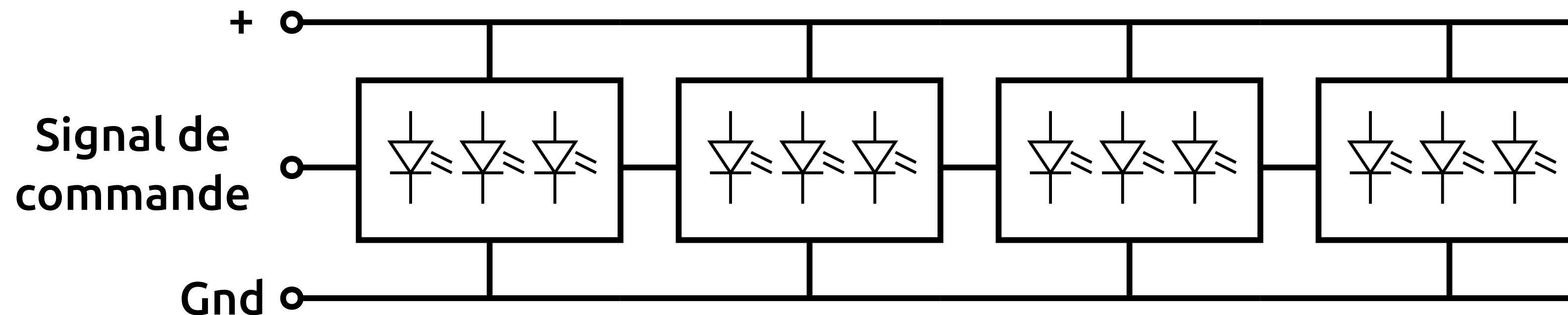


- Horloge asymétrique

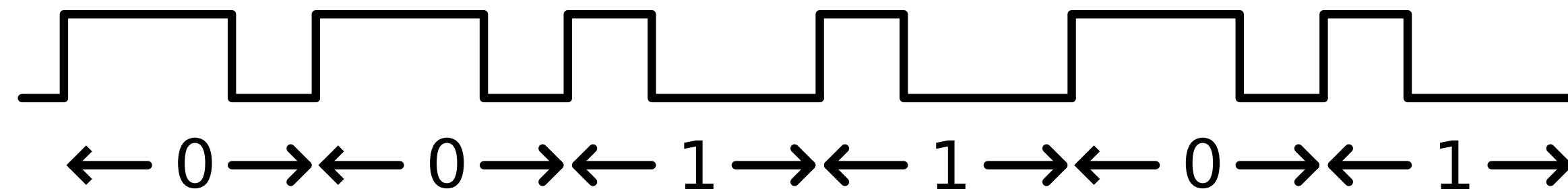


Rubans adressables

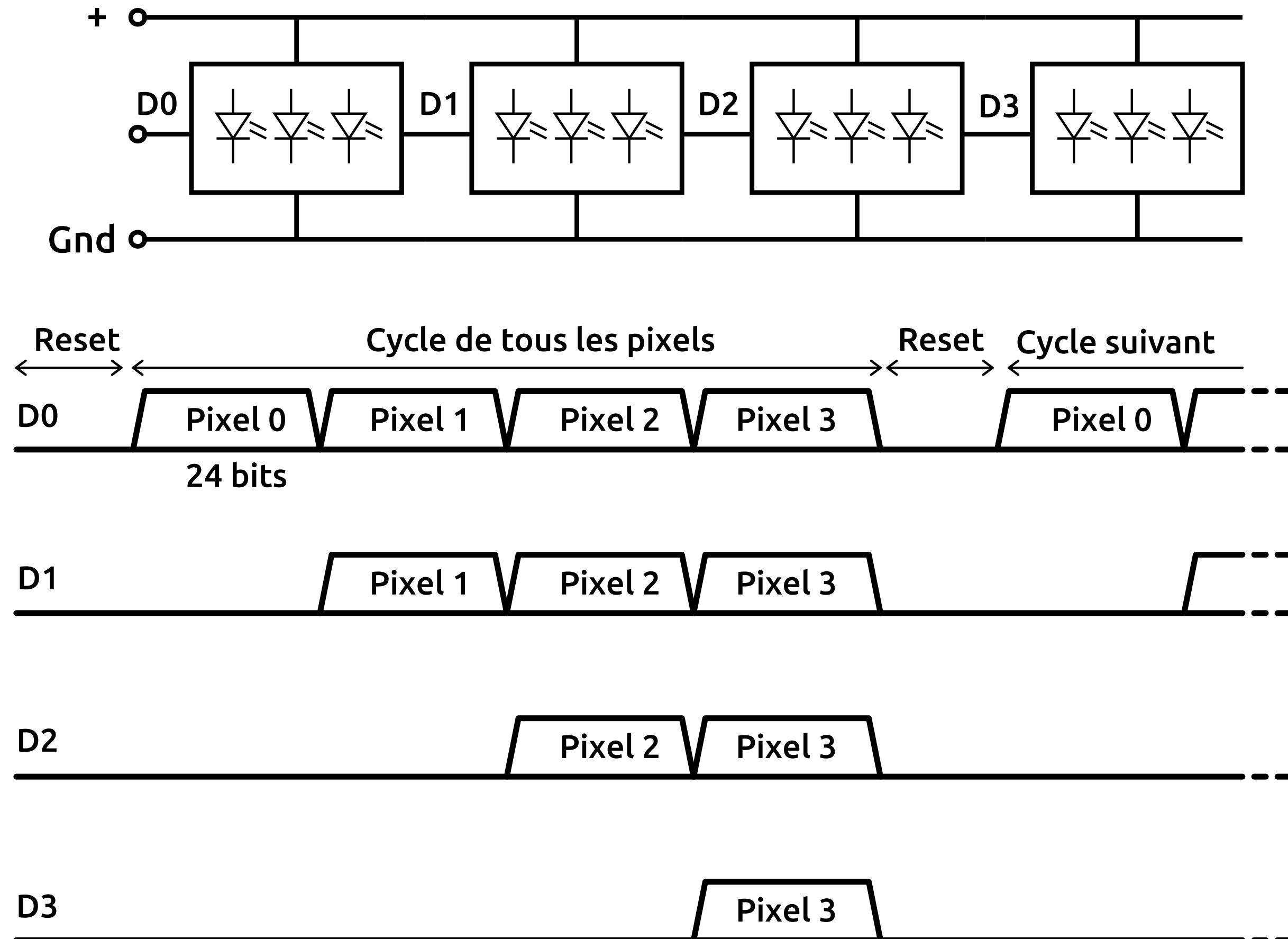
- Chaque LED est indépendante pour sa couleur et son intensité
- Rubans adressables (*Addressable strips*)



- Horloge asymétrique
- Worldsemi WS28xx



Registres série-parallèles



Chaque registre :

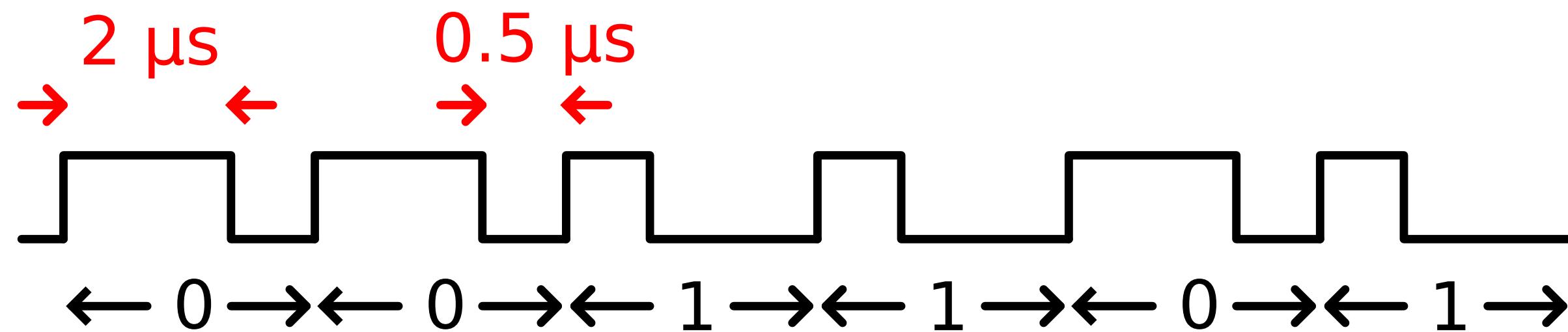
- garde la première donnée
- transmet les suivantes

Génération des signaux

- Le fabricant donne des contraintes sur le *timing* des signaux

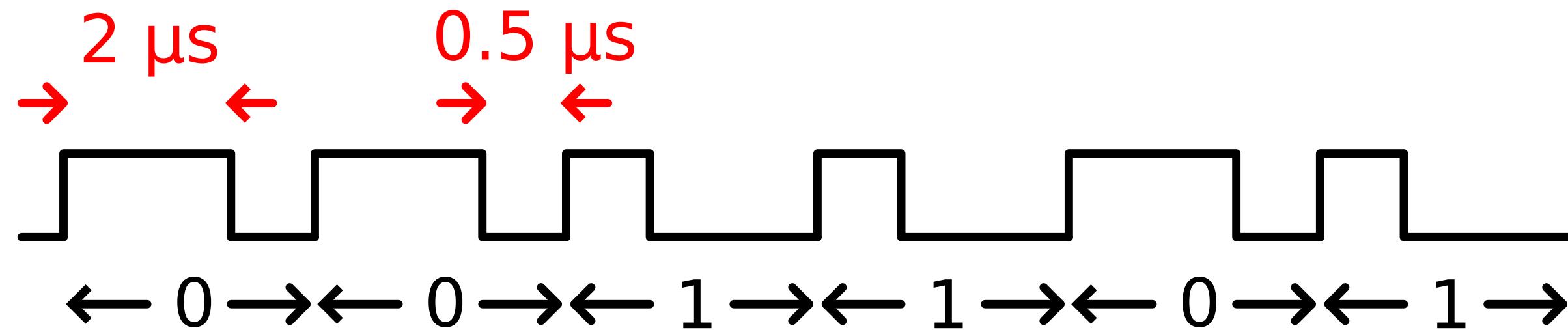
Génération des signaux

- Le fabricant donne des contraintes sur le *timing* des signaux



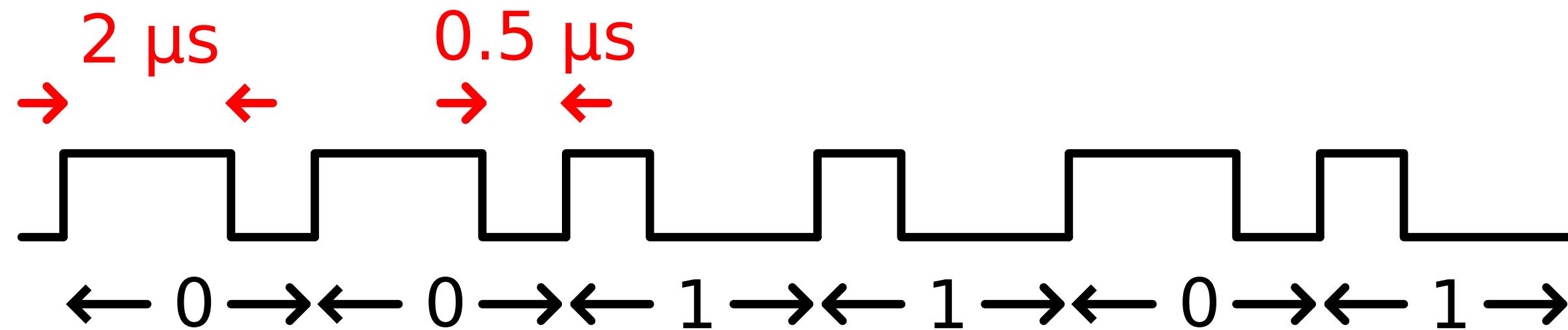
Génération des signaux

- Le fabricant donne des contraintes sur le *timing* des signaux
- Difficile de tenir les spécifications avec des processeurs dont l'horloge est à 16 MHz



Génération des signaux

- Le fabricant donne des contraintes sur le *timing* des signaux
- Difficile de tenir les spécifications avec des processeurs dont l'horloge est à 16 MHz
- Plus facile avec un processeurs dont l'horloge est de 48 MHz : **ARM**



Bit banging

```
1 #define PORT_WS2811 GPIOA
2 #define BIT_WS2811 10
3 #define WS280n (PORT_WS2811->ODR|=(1<<BIT_WS2811))
4 #define WS280ff (PORT_WS2811->ODR&=~(1<<BIT_WS2811))
5
6 #define Un WS280n;WS280n;WS280n;WS280n;WS280n;WS280n;WS280n;WS280ff;
7 #define Zero WS280n;WS280ff;WS280ff;WS280ff;WS280ff;WS280ff;WS280ff;WS280ff
8
9 #define UnCourt WS280n;WS280n;WS280n;WS280n;WS280n;WS280n;WS280n;WS280ff;
10 #define ZeroCourt WS280n;WS280ff;WS280ff;WS280ff;WS280ff;WS280ff;WS280ff;WS280ff;
```

Initialisations et variables

```

12 // Contenu du ruban :
13 #define LgRuban 50
14 uint32_t Ruban[LgRuban];
15
16 int main(void) {          // Programme principal
17     HAL_Init();           // Initialisation de la librairie Hardware Level
18     SystemClock_Config(); // Configure l'horloge système
19     MX_GPIO_Init();       // Initialise les périphériques
20     PORT_WS2811->MODER |= (0b01 << (BIT_WS2811*2)); // broche en sortie
21
22     uint32_t i;
23     volatile uint16_t j;
24     uint32_t v, idx;
25     uint32_t *pt;          // pointeur dans le tableau
26
27     // Initialisation fixe des couleurs
28     for (idx=0; idx<LgRuban; idx++) {
29         Ruban[idx] = 1<<idx;
30     }
  
```

Boucle critique

```

32 while (1) { // boucle principale
33     pt = Ruban;
34     __ASM volatile ("cpsid i"); // interrupt OFF
35
36     for (idx=0; idx<LgRuban; idx++) {
37         v = *pt;
38         if (v & (1<<23)) {Un;} else {Zero;}
39         if (v & (1<<22)) {Un;} else {Zero;}
40         if (v & (1<<21)) {Un;} else {Zero;}
41         if (v & (1<<20)) {Un;} else {Zero;}
42         if (v & (1<<19)) {Un;} else {Zero;}
43         if (v & (1<<18)) {Un;} else {Zero;}
44         if (v & (1<<17)) {Un;} else {Zero;}
45         if (v & (1<<16)) {Un;} else {Zero;}
46         if (v & (1<<15)) {Un;} else {Zero;}
47         if (v & (1<<14)) {Un;} else {Zero;}
48         if (v & (1<<13)) {Un;} else {Zero;}
49         if (v & (1<<12)) {Un;} else {Zero;}

```

Boucle critique

```

50   if (v & (1<<11)) {Un;} else {Zero;}
51   if (v & (1<<10)) {Un;} else {Zero;}
52   if (v & (1<<9)) {Un;} else {Zero;}
53   if (v & (1<<8)) {Un;} else {Zero;}
54   if (v & (1<<7)) {Un;} else {Zero;}
55   if (v & (1<<6)) {Un;} else {Zero;}
56   if (v & (1<<5)) {Un;} else {Zero;}
57   if (v & (1<<4)) {Un;} else {Zero;}
58   if (v & (1<<3)) {Un;} else {Zero;}
59   if (v & (1<<2)) {Un;} else {Zero;}
60   if (v & (1<<1)) {UnCourt;} else {ZeroCourt;}
61   pt++;
62   if (v & (1<<0)) {UnCourt;} else {ZeroCourt;}
63 }
64
65 __ASM volatile ("cpsie i"); // interrupt ON
66 for (j=0; j<500; j++) {      // reset
67 }
68 }
```

Programmer des animations

```
71 temps++; // comptage du temps
72
73 // Clignotement des LED 0 et 30 :
74 if (temps==500) {
75     Ruban[30] = Ruban[0] = 0xFFFF;
76 }
77 if (temps==1000) {
78     temps=0; Ruban[30] = Ruban[0] = 0;
79 }
80 // Changement progressif de la couleur de la LED 47 :
81 Ruban[47]++;
82 }
```

Rubans de LED

- Rubans uniformes
- Rubans addressables
- Signaux de commande
- Programmation