RUBANS DE LED

Pierre-Yves Rochat, EPFL

rév 2016/07/17

Document incomplet, en cours de rédaction.

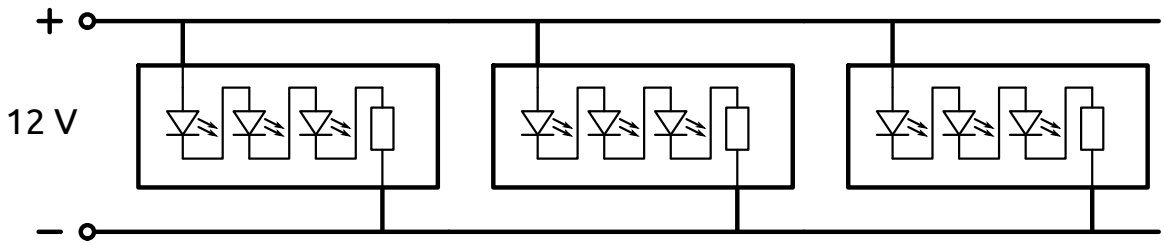
PLUSIEURS SORTES DE RUBANS DE LED

Les rubans de LED, en anglais *LED strips*, sont des LED disposées en ligne. Elles sont reliées entre elles par un circuit imprimé flexible, ou simplement par des fils. On trouve sur le marché plusieurs types de rubans de LED. On distingue principalement les rubans uniformes et les rubans adressables. Les rubans uniformes peuvent être d'une seule couleur fixe. Dans ce cas seule l'intensité peut être modifiée, pour toutes les LED du ruban en même tems, par une commande en PWM. Les rubans peuvent aussi être multicolores (RGB). Dans ce cas, tout le ruban peut changer de couleur en même temps. La commande se fait par un triple PWM, un pour chaque couleur.

Basés sur une technologie très différente, il existe aussi des rubans adressables (*addressable led strips*). Chaque pixel du ruban peut alors prendre une couleur indépendamment des autres pixels.

LES RUBANS UNIFORMES

Les rubans uniformes ne contiennent que des LED avec leurs résistances de limitation. Généralement, l'alimentation est de 12 V, ce qui signifie que plusieurs LED sont mises en série. Pour des rubans RGB, le câblage est généralement à cathod commune, avec une connexion pour les cathodes et trois connexions pour les anodes des LED rouges, vertes et bleues :

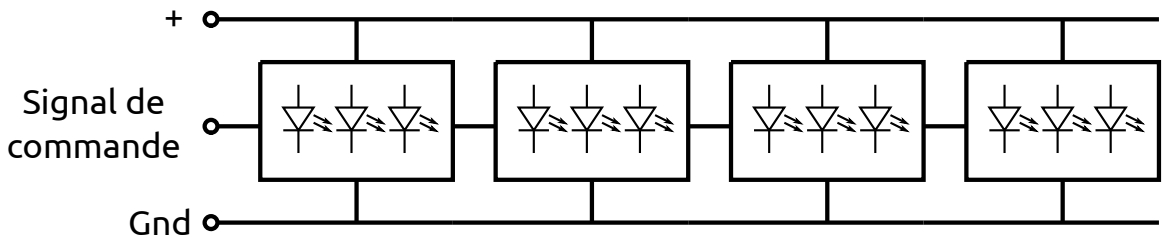


Rubans uniformes

LES RUBANS ADRESSABLES

Plusieurs solutions sont techniquement possibles pour faire varier l'intensité de chaque pixel d'un ruban. Celles qui minimisent le nombre de fils utilisé sont évidemment les plus intéressantes. Il faut forcément deux fils pour alimenter les LED, le *Gnd* et le *Vcc*. Est-ce possible d'utiliser un seul fil pour apporter à chaque pixel l'information qui le concerne ?

Une solution très souvent utilisée présente l'architecture suivante :

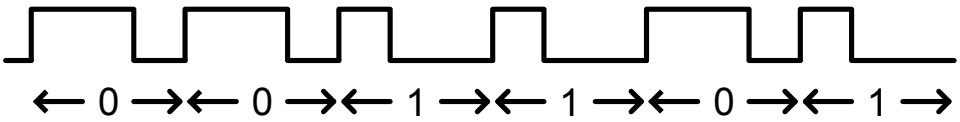


Rubans adressables à 3 fils

Alors que les deux fils d'alimentation relient chaque module de pixel, le troisième fil relie la sortie d'un module vers l'entrée d'un autre.

LE PROTOCOLE DES WS28XX

On imagine bien que les données vont être transmises en série. Mais il devient nécessaire de regrouper sur un seul fil les données et l'horloge. Le principe est une horloge asymétrique :



Horloge asymétrique

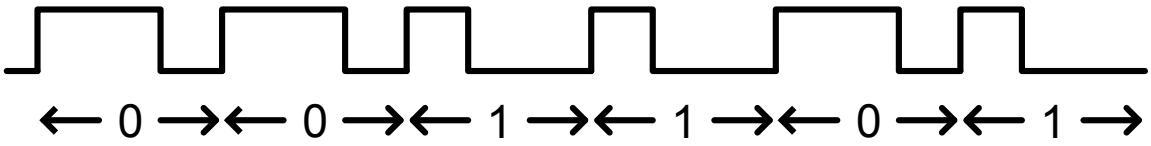
Chaque bit est transmis par un cycle d'horloge. Lorsque la durée de la partie haute du signal est plus longue que le durée de la partie basse, le bit transmis est un zéro.

Dans le cas des circuits de la famille WS28xx, un circuit intégré est utilisé pour chaque pixel, composé de trois LED de couleur rouge, verte et bleu. Souvent, les trois LED sont encapsulées dans le même boîtier. On parle alors d'une LED RGB (*Red Green Blue*).

Chaque LED est commandée par un signal PWM de 8 bits. Il est donc nécessaire d'envoyer 24 bits pour chaque pixel.

Pour synchroniser le début de l'envoi d'une nouvelle série de valeurs à tous les pixels du ruban, une attente d'au moins 50 us est nécessaire. Chaque circuit est alors prêt à recevoir 24 bits. L'astuce utilisée est alors la suivante : chaque circuit ne transmet à sa sortie les information qui se présentent à sont entrée qu'après avoir enregistré les 24 premiers bits qui succèdent au Reset.

La figure suivante explique ce principe :



Mise en cascade des modules

Signaux rapides

Les contraintes qu'imposent ce circuit sur le timing des signaux rendent difficile sa programmation avec un AVR ou un MSP430. Des solutions ont toutefois été trouvées, en programmant en assembleur ou encore en utilisant de manière astucieuse le circuit de communication série.

Avec un processeur ARM, c'est plus facile de respecter les exigence temporelles. Nous allons ici montrer un programme écrit pour un STM32, testé sur une carte Nucleo. Voici la partie avec les définitions :

```
// Signal One Wire pour WS2811 :
#define WS2811_Pin GPIO_PIN_10
#define PORT_WS2811 GPIOA
#define BIT_WS2811 10
#define WS28On (PORT_WS2811->ODR|=(1<<BIT_WS2811))
#define WS28Off (PORT_WS2811->ODR&=~(1<<BIT_WS2811))

void SystemClock_Config(void);
static void MX_GPIO_Init(void);

#define Un WS28On;WS28On;WS28On;WS28On;WS28On;WS28On;WS28On;WS28Off;
#define Zero WS28On;WS28Off;WS28Off;WS28Off;WS28Off;WS28Off;WS28Off;WS28Off

#define UnCourt WS28On;WS28On;WS28On;WS28On;WS28On;WS28On;WS28Off;
#define ZeroCourt WS28On;WS28Off;WS28Off;WS28Off;WS28Off;WS28Off;WS28Off;


// Contenu du ruban :
#define LgRuban 50
uint32_t Ruban[LgRuban];

int main(void) { // Programme principal :
  HAL_Init(); // Initialisation de la librairie Hardware Level
  SystemClock_Config(); // Configure l'horloge système
  MX_GPIO_Init(); // Initialise les périphériques

  PORT_WS2811->MODER |= (0b01 << (BIT_WS2811*2)); // broche en sortie

  uint32_t i;
  volatile uint16_t j;
  uint32_t v;
  uint32_t idx;
  uint32_t *pt; // pointeur

  // Initialisations fixes des couleurs
  for (idx=0; idx<LgRuban; idx++) {
    Ruban[idx] = 1<<idx;
  }

  while (1) { // boucle principale
    pt = Ruban;
    __ASM volatile ("cpsid i");
```

```
    for (idx=0; idx<LgRuban; idx++) {
      v = *pt;
      if (v & (1<<23)) {Un;} else {Zero;}
      if (v & (1<<22)) {Un;} else {Zero;}
      if (v & (1<<21)) {Un;} else {Zero;}
      if (v & (1<<20)) {Un;} else {Zero;}
      if (v & (1<<19)) {Un;} else {Zero;}
      if (v & (1<<18)) {Un;} else {Zero;}
      if (v & (1<<17)) {Un;} else {Zero;}
      if (v & (1<<16)) {Un;} else {Zero;}
      if (v & (1<<15)) {Un;} else {Zero;}
      if (v & (1<<14)) {Un;} else {Zero;}
      if (v & (1<<13)) {Un;} else {Zero;}
      if (v & (1<<12)) {Un;} else {Zero;}
      if (v & (1<<11)) {Un;} else {Zero;}
      if (v & (1<<10)) {Un;} else {Zero;}
      if (v & (1<<9)) {Un;} else {Zero;}
      if (v & (1<<8)) {Un;} else {Zero;}
      if (v & (1<<7)) {Un;} else {Zero;}
      if (v & (1<<6)) {Un;} else {Zero;}
      if (v & (1<<5)) {Un;} else {Zero;}
      if (v & (1<<4)) {Un;} else {Zero;}
      if (v & (1<<3)) {Un;} else {Zero;}
      if (v & (1<<2)) {Un;} else {Zero;}
      if (v & (1<<1)) {UnCourt;} else {ZeroCourt;}
      pt++;
      if (v & (1<<0)) {UnCourt;} else {ZeroCourt;}
    }

    __ASM volatile ("cpsie i");

    for (j=0; j<500; j++) {
    }
  }
}
```

Il est alors possible de créer des animations sur les LED. Une variable *temps* va compter le temps qui s'écoule et comptant les cycles de raffraîchissement du ruban. En fonction du temps, les couleurs des LED peuvent être modifiées :

```
    temps++; // comptage du temps

    // Clignotement des la LED 0 et 30 :
    if (temps==500) {
        Ruban[30] = Ruban[0] = 0xFFFFFF;
```

```
}
if (temps==1000) {
    temps=0; Ruban[30] = Ruban[0] = 0;
}
// Changement progressif de la couleur de le LED 47 :
Ruban[47]++;
```