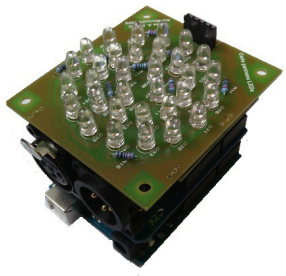
Projecteur RVB DMX / RDM avec Arduino Uno

NOM :	Date :



Objectif final :
Mettre en œuvre et programmer un shield Arduino afin de réaliser un mini-projecteur compatible DMX . Cela passe par l'analyse et l'utilisation d'une interface de puissance.

Compétences visées :

Concevoir	C3.9 : Valider une fonction à partir d'une maquette réelle : tester la conformité d'une fonction sur un dispositif de prototypage rapide. C3.10 : Élaborer les programmes de la fonction à réaliser
Réaliser	C4.2 : adapter et configurer un nouveau matériel C4.3 : adapter et configurer une structure logicielle C4.5 : tester et valider un module logiciel et matériel

Savoirs visés :

Savoir	**Description**
S4.2. Algorithmique	Structures fondamentales : enchaînements, alternatives, itérations, etc. Représentation graphique (organigrammes)
S4.4. Programmation procédurale	Manipulations de données en langage C
	Transcription d'algorithmes en langage C
S5.2. Traitement logiciel des E/S	Caractéristiques TOR
S8.1 Instruments de mesure	Oscilloscope Analyseurs logiques

Moyens :

- Ordinateur Windows + logiciel Arduino + logiciel QLC+
- Interface USB/DMX
- Arduino Uno + cordon USB alim./prog. + shield DMX RDM + librairies Conceptinetics
- Câble XLR3
- Alimentation régulée ajustable
- Câble compatible avec les panneaux de LED
- Led bicolore + fils de liaison
- Panneau de LED Dune.
- 2 Panneaux de LED didactisés avec connectiques différentes.
- Shield interface de puissance pour panneau de LED.
- Bloc secteur pour interface de puissance.

Conditions :
- Travail en binôme.
- Durée : 2 séances de 4H
- Compte rendu à la fin de la séance.

Prérequis :
- Avoir travaillé sur carte Arduino.

I. Mise en situation

On souhaite dans un premier temps faire évoluer une carte Arduino Uno en récepteur DMX en lui ajoutant un shield DMX/RDM.

Puis on souhaite transformer ce récepteur en projecteur RVB en lui associant différents types de panneaux de LEDs, et en ajoutant une configuration manuelle de son adresse DMX

Toutes les documentations nécessaires à la réalisation de ce mini-projet figurent sur le site du BTS SN, il est indispensable de les consulter pour mener à bien toutes les activités.

Sur l'ordinateur il faudra travailler en se connectant sur son compte personnel.

II. Faire évoluer une carte Arduino en récepteur DMX / RDM.

Analyse

1. Sur le site, consulter les informations concernant le shield d'adaptation DMX/RDM.

   Dans cette première partie du mini-projet seule le mode récepteur DMX *(DMX Slave)* fera l'objet de l'étude.

   Analyser le programme et les commentaires du fichier « DMX Slave » qui figure sur l'ANNEXE 1 pour répondre aux questions qui suivent.
2. Une fois la carte programmée avec ce programme quelle sera par défaut l'adresse DMX de la carte Arduino ?

   →
3. Quelle sortie de la carte Arduino sera activée à cette adresse DMX ? →
4. Comment réagira la sortie en fonction de la valeur DMX récupérée ?

   →
5. Combien de canaux seront réservés pour ce récepteur DMX ? →

Mise en œuvre

6. Dans votre espace mémoire personnel créer un sous-répertoire de travail, exemple : TP_Arduino_DMX_RDM.
7. Télécharger depuis le site la librairie : Conceptinetics.
8. Lancer le logiciel Arduino et suivre la méthode indiquée sur le site permettant d'importer la librairie Conceptinetics et ses exemples dans votre espace de travail personnel.
9. Ouvrir le programme « DMX_Slave », puis le sauvegarder sous un autre nom dans la perspective des modifications à venir. *Pour pourrez aussi par la suite créer un nouveau fichier pour chaque version.*
10. La notice accompagnant le shield DMX/RDM figure sur le site. Le mode d'utilisation du shield étant « DMX Slave » compléter ci-dessous *(colonne de gauche)* la position que devront avoir les 4 cavaliers du shield lors de la programmation de la carte Arduino. Puis faire de même *(colonne de droite)* pour configurer l'utilisation de l'ensemble Arduino + shield sur l'installation DMX.

	Arduino en mode programmation	Arduino en mode utilisation Slave DMX
Cavalier 1 *(EN ou /EN)*		
Cavalier 2 *(Slave ou DE)*		
Cavalier 3 *(TX-io ou TX-uart)*		
Cavalier 4 *(RX-io ou RX-uart)*		

11. Assembler la carte Arduino et le shield.
12. Programmer la carte Arduino.
13. Câbler une LED sur la sortie pilotée par la trame DMX *(voir photo sur le site)*.
14. Piloter l'ensemble avec le logiciel QLC+ et une interface USB/DMX et vérifier le bon fonctionnement. *Le petit interrupteur fourni permet de basculer facilement en mode programmation ou DMX.*
15. Modifier le programme pour que l'adresse DMX du récepteur passe à 7.
16. Modifier le programme pour que la sortie 8 de la carte Arduino soit utilisée à la place de la précédente.
17. Le composant mis à votre disposition contenant en fait 2 LED, modifier le programme pour que :
    - la LED rouge s'allume à l'adresse 7 pour une valeur supérieure à 100,
    - la LED verte pour une valeur inférieure à 200 à l'adresse 10.

    Voir la vidéo sur le site dans le menu : Résultats attendus.

Faire constater le bon fonctionnement, ***avant d'imprimer la partie utile du programme à remettre en fin de TP***

III. La carte Aduino peut-elle piloter directement les différents panneaux de LEDs ?

On souhaite piloter 3 structures à LEDs à partir de l'association précédente :

- Le panneau de LEDs d'un projecteur DUNE RVB *(schéma : ANNEXE 2)*.
- La version didactisée de ce panneau de LEDs *(schéma : ANNEXE 3)*.
- Une LED RGB haute luminosité *(documentation sur le site)*.

18. Quelle est l'amplitude de la tension qui doit alimenter les structures ? →
19. Les panneaux de LEDs sont-il à anode ou cathode commune ? →

Évaluation expérimentale de la consommation des LEDs du panneau didactisé.

Vous disposez du panneau didactisé, d'une alimentation régulée ajustable, d'un cordon d'adaptation.

20. Rédiger une procédure permettant de mesurer l'intensité consommée par les LEDs rouge sur ce panneau. **Faire valider votre méthode par le professeur avant toute mise en œuvre.**

    →

21. Mettre en œuvre cette méthode pour chacune des 3 couleurs.

    → Intensité des LEDs rouges du panneau didactisé =

    → Intensité des LEDs vertes du panneau didactisé =

    → Intensité des LEDs bleues du panneau didactisé =

22. Mettre en œuvre cette méthode sur le panneau du projecteur DUNE RVB

    → Intensité des LEDs rouges du panneau DUNE =

    → Intensité des LEDs vertes du panneau DUNE =

    → Intensité des LEDs bleues du panneau DUNE =

Faites constater vos mesures

23. Analyser la documentation de la LED haute luminosité pour en extraire les valeurs théoriques suivantes :

    → Intensité nominale consommée par la LED rouge =

    → Intensité nominale consommée par la LED verte =

    → Intensité nominale consommée par la LED bleue =

Intensités en sortie de la carte Arduino Uno.

24. Quel est le modèle de microcontrôleur qui équipe la carte Arduino Uno R3 mise à disposition ?

    →

25. Quelle est l'intensité qui peut être fournie ou absorbée par les sorties logiques de ce circuit ? Préciser la page de la documentation du microcontrôleur qui indique cette valeur.

    →

Conclusion

26. La carte Arduino peut-elle piloter directement les structures à LED dans les 3 cas proposés ?

    →

Analyse d'une structure d'adaptation en puissance.

Le schéma de la structure retenue figure en ANNEXE 4, elle repose sur des transistors utilisés en commutation *(fonctionnement en tout ou rien : TOR)*.

27. Entourer sur le schéma la connectique de liaison avec la carte Arduino et le shield DMX/RDM.
28. Quelle est la référence des transistors qui vont servir d'interfaces de puissance ? →
29. Consulter la documentation des transistors pour justifier qu'ils peuvent supporter l'intensité ainsi que la tension d'alimentation dans les 3 cas proposés.

    →

→

30. Quelle est la technologie des transistors utilisés ? →
31. Donner le nom *(numéro)* des sorties de la carte Arduino qui vont être utilisées pour piloter les transistors ? De quel type sont ces sorties ?

    →

    →
32. Quel sera le rôle du « Dip switch » pour le futur projecteur ?

    →

Faites constater votre analyse

IV. Programmation de la carte Arduino pour commander l'interface de puissance.

33. Sauvegarder votre version antérieure du programme sous un autre nom en prévision des modifications à venir.

    On souhaite que les sorties qui pilotent les LED soient commandées en PWM à partir des valeurs véhiculées par le signal DMX. Des informations figurent sur le site pour rappeler le principe de la commande PWM.
34. Dans l'IDE Arduino *(Integrated Development Environment)*, consulter la documentation du langage de programmation de la carte *(menu : aide → référence)* et indiquer quelle type de sortie et quelle fonction *(instruction)* il faut utiliser pour effectuer une commande en PWM. Quelle est la fréquence de cette PWM et la plage de valeurs du rapport cyclique, cette dernière est-elle directement compatible avec les données DMX ?

    →

    →

    →

    →

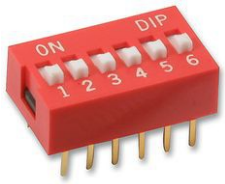
    →
35. Modifier le programme pour piloter les 3 couleurs à partir des sorties de la carte Arduino utilisées par l'interface de puissance. *Voir vidéo sur le site.*
36. Visualiser les signaux avec un oscilloscope ou un analyseur logique (LogicPort ou Saleae). Vérifier les informations sur la fréquence du signal.

    →
37. Vérifier le bon fonctionnement en pilotant l'ensemble avec une console DMX. La carte Arduino devra fonctionner de façon autonome *(si elle est connectée sur le PC ce sera uniquement pour son alimentation)*.

Faire constater le bon fonctionnement, ***avant d'imprimer la partie utile du programme à remettre en fin de TP***

38. Dessiner sur une feuille libre l'organigramme du programme réalisé et le joindre au compte-rendu.

V. Configurer l'adresse DMX par DIP switch.



On souhaite pouvoir à tout moment modifier l'adresse DMX du projecteur.

Capture de la valeur des interrupteurs

39. Récupérer sur le site le projet « *DMX_Slave_DIP_Switch.ino* » et le copier dans votre espace de travail. L'ouvrir et le sauvegarder sous *« votre_nom_dip_switch.ino ».* Le Document Annexe 5 est une version papier de ce fichier.

    Le DIP switch devra être disposé dans le sens indiqué sur la photo figurant sur le site. L'interrupteur 1 aura de poids faible et le 6 le poids fort.

40. Compléter le programme pour faire l'acquisition des 6 niveaux logiques du DIP switch, et les convertir en leur équivalent décimal qui sera stocké dans la variable « Adresse_DMX ».

    Attention pour la question suivante il ne faudra pas mettre le shield DMX/RDM en mode Enable.

41. Tester le programme. Visualiser dans le moniteur série si les valeurs entrées sur le DIP switch sont convenablement interprétées. L'utilisation du moniteur permettra de débugger le programme. Une vidéo *(Vidéo N°1)* du résultat attendu figure sur le site.

    Faites constater le bon fonctionnement

42. Reprenez votre projet de la partie 1 du mini projet et sauvegardez-le sous un autre nom : *votre_nom_proj_dip.ino*

43. Modifiez le programme pour qu'il intègre l'acquisition de l'adresse DMX.

    Une modification de l'adresse devra tout de suite être prise en compte.

    Une adresse valant 0 devra provoquer l'extinction de toutes les LED du projecteur.

    Une vidéo *(Vidéo N°2)* du résultat attendu figure sur le site.

    Faites constater le bon fonctionnement

44. Imprimer votre programme et le joindre à votre compte-rendu.

45. Imaginons un appareil tel qu'une lyre.

    - Celle-ci est-elle configurée avec un DIP switch ? →
    - La valeur du réglage est-elle perdue lors de la mise hors tension ? →
    - Comment est sauvegardée l'adresse sur ce type d'appareil ? →

VI. Ajout d'un canal dimmer/strobe au projecteur.

On souhaite rajouter un canal au projecteur.

Pour cela le but est de s'inspirer d'un vrai projecteur RVB Dune dont la documentation figure sur le site.

Mesures

46. Piloter le projecteur Dune à partir d'une console logicielle QLC+. Quel est le canal de la commande

Dimmer/Strobe ? →

47. Pour quelle plage de valeurs de la commande Dimmer/Strobe le projecteur clignote-t-il ? →

Pour simplifier la modélisation considérons que :

- lorsque le projecteur clignote la durée des éclats de lumière est de 35ms, quelle que soit la valeur de Dimmer/Strobe.
- pour Dimmer/Strobe = 0 la période de clignotement est de 55 ms
- pour Dimmer/Strobe = 125 la période de clignotement est de 2,5sec.

48. Lorsque le projecteur est en mode clignotement, pour chaque incrément Dimmer/Strobe calculer la variation sur la durée d'extinction. Cette valeur est la même pour chaque incrément. Les calculs seront arrondis à la milliseconde près.

→

49. Dans le mode clignotement calculer la durée d'extinction *(qui sera appelée D_Ext)* en fonction de la valeur de Dimmer/Strobe *(qui sera appelée DS)*. La durée sera exprimée en ms.

→

Mise en œuvre

50. Reprenez votre projet de la partie précédente *(portant sur l'acquisition de l'adresse DMX)* et sauvegardez-le sous un autre nom : *votre_nom_proj_dip_DS.ino*.
51. Modifier ce projet pour :
    - ajouter un canal,
    - obtenir un clignotement intégrant les calculs précédents,
    - dans la plage de valeurs de Dimmer/Strobe qui ne font pas clignoter le projecteur, vous ferez en sorte que l'affichage soit uniquement commandé par les canaux RVB *(même si ce n'est pas rigoureusement le cas pour le projecteur Dune).*
    - les variables DS et D_Ext seront de type entier.

    Une vidéo *(Vidéo N°3)* du résultat attendu figure sur le site.

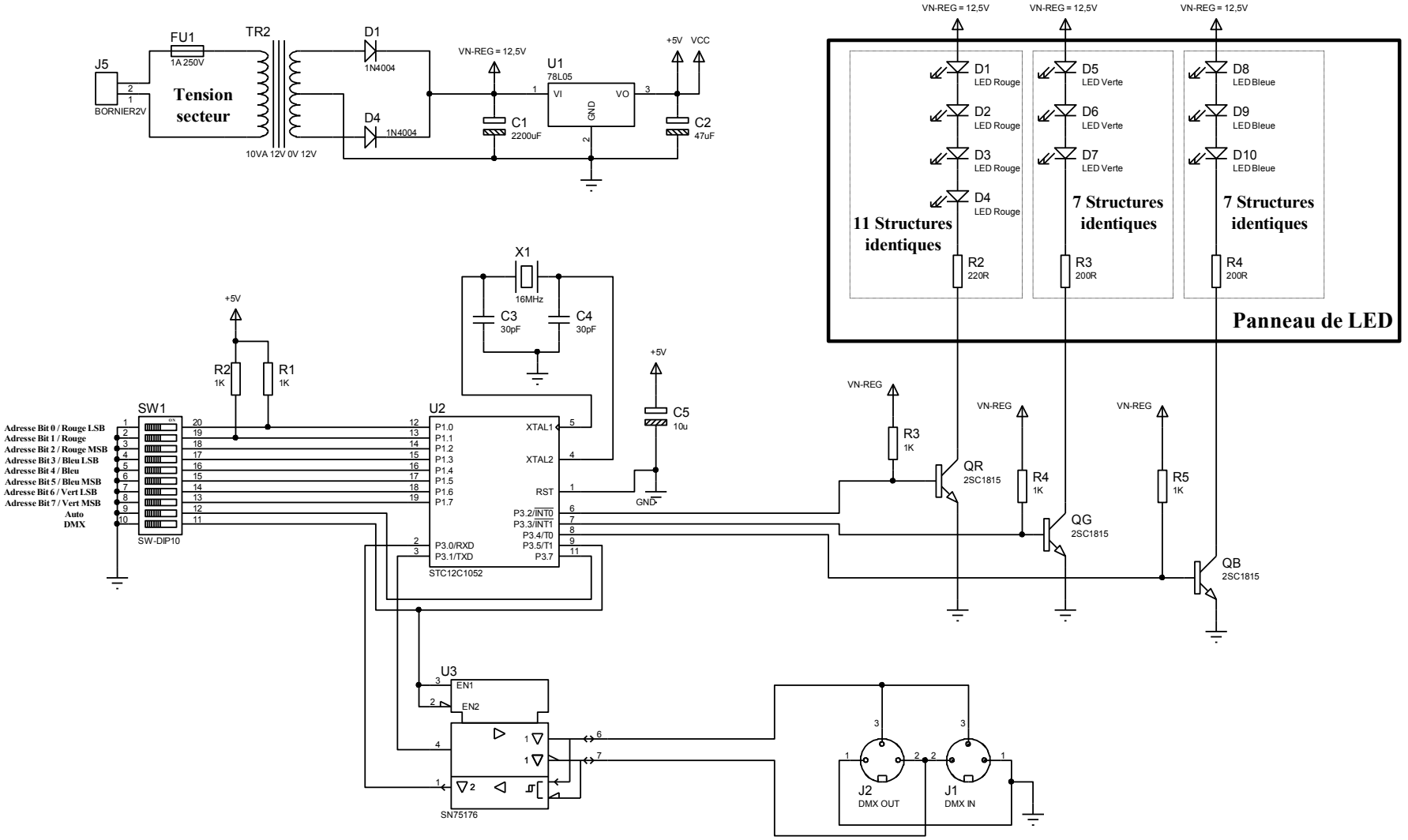
Faites constater le bon fonctionnement

52. Utiliser un oscilloscope pour vérifier que les durées obtenues sont bien celles souhaitées pour Dimmer/Stobe = 0 et pour le clignotement le plus lent. Ces chronogrammes devront figurer dans votre compte-rendu.

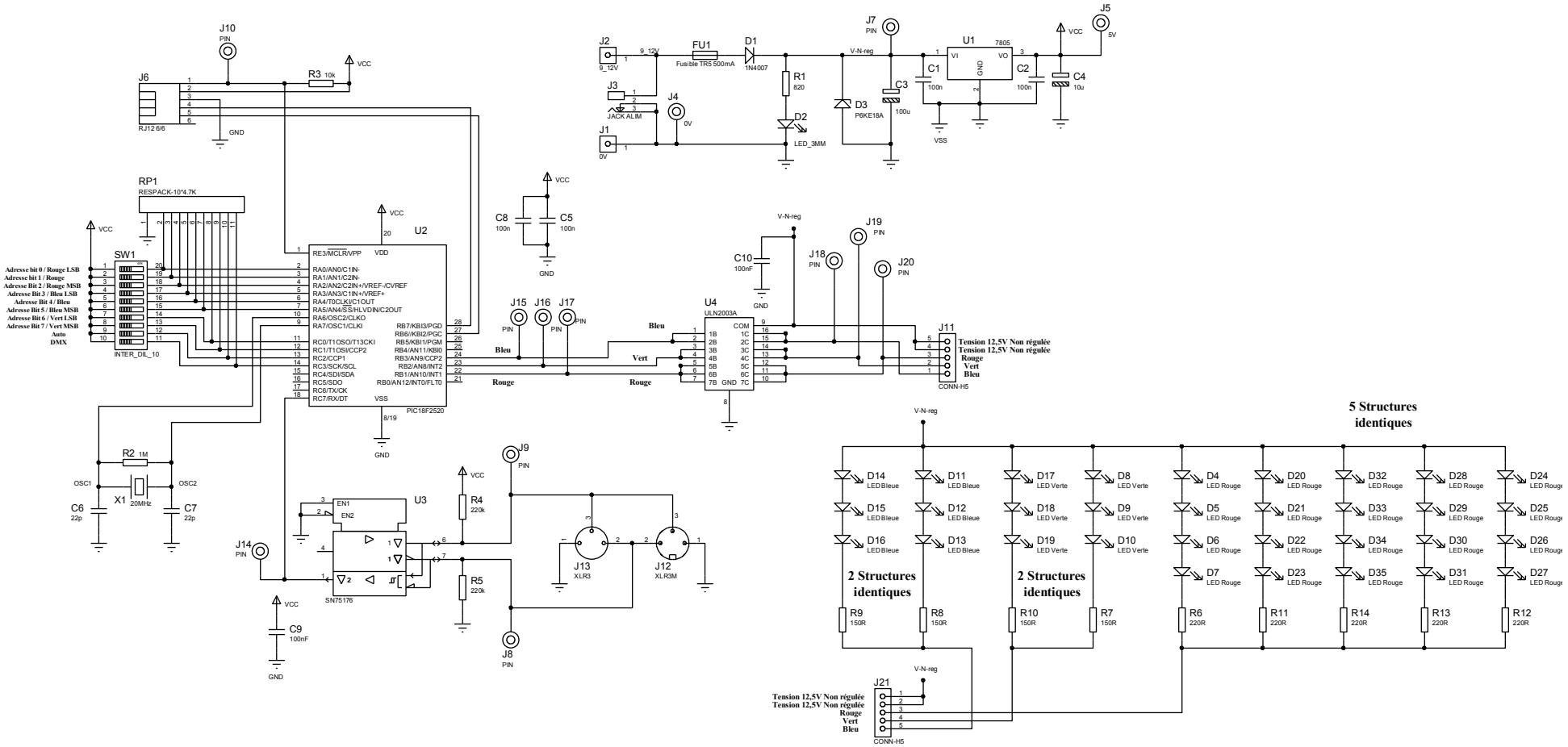
Document ANNEXE 1 : Programme DMX_Slave.ino

```
/*
  DMX_Slave.ino - Example code for using the Conceptinetics DMX library
  Copyright (c) 2013 W.A. van der Meeren <danny@illogic.nl>.  All right reserved.

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 3 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
*/

#include <Conceptinetics.h>

//
// CTC-DRA-13-1 ISOLATED DMX-RDM SHIELD JUMPER INSTRUCTIONS
//
// If you are using the above mentioned shield you should
// place the RXEN jumper towards G (Ground), This will turn
// the shield into read mode without using up an IO pin
//
// The !EN Jumper should be either placed in the G (GROUND)
// position to enable the shield circuitry
//   OR
// if one of the pins is selected the selected pin should be
// set to OUTPUT mode and set to LOGIC LOW in order for the
// shield to work
//

//
// The slave device will use a block of 10 channels counting from
// its start address.
//
// If the start address is for example 56, then the channels kept
// by the dmx_slave object is channel 56-66
//
#define DMX_SLAVE_CHANNELS   10
//
// Pin number to change read or write mode on the shield
// Uncomment the following line if you choose to control
// read and write via a pin
//
// On the CTC-DRA-13-1 shield this will always be pin 2,
// if you are using other shields you should look it up
// yourself
//
///// #define RXEN_PIN            2

// Configure a DMX slave controller
DMX_Slave dmx_slave ( DMX_SLAVE_CHANNELS );

// If you are using an IO pin to control the shields RXEN
// the use the following line instead
///// DMX_Slave dmx_slave ( DMX_SLAVE_CHANNELS , RXEN_PIN );

const int ledPin = 13;

// the setup routine runs once when you press reset:
void setup() {

  // Enable DMX slave interface and start recording
  // DMX data
  dmx_slave.enable ();

  // Set start address to 1, this is also the default setting
  // You can change this address at any time during the program
  dmx_slave.setStartAddress (1);

  // Set led pin as output pin
  pinMode ( ledPin, OUTPUT );
}

// the loop routine runs over and over again forever:
void loop()
{
  //
  // EXAMPLE DESCRIPTION
  //
  // If the first channel comes above 50% the led will switch on
  // and below 50% the led will be turned off

  // NOTE:
  // getChannelValue is relative to the configured startaddress
  if ( dmx_slave.getChannelValue (1) > 127 )
    digitalWrite ( ledPin, HIGH );
  else
    digitalWrite ( ledPin, LOW );
}
```

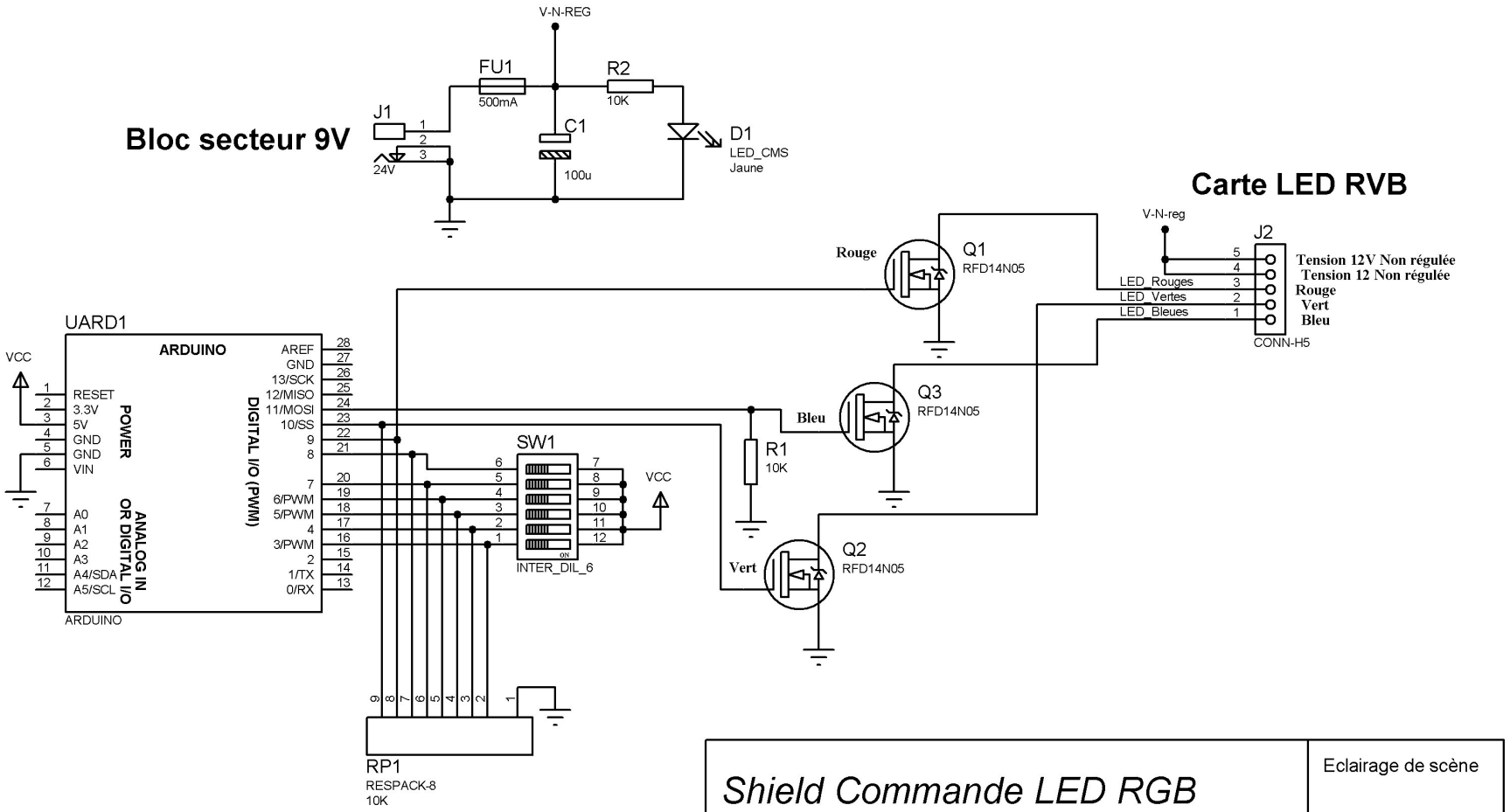
Document ANNEXE 2 : Schéma structurel du projecteur RVB DMX DUNE



Document ANNEXE 3 : Schéma structurel du projecteur didactisé



Document ANNEXE 4 : Schéma structurel de l'interface de puissance



Document ANNEXE 5 : Projet *DMX_Slave_DIP_Switch.ino*

```
const int ledPin_Rouge = 9;
const int ledPin_Vert = 10;
const int ledPin_Bleu = 11;

const int Ad1 =          ;                    // A compléter
const int Ad2 =          ;                    // A compléter
const int Ad3 =          ;                    // A compléter
const int Ad4 =          ;                    // A compléter
const int Ad5 =          ;                    // A compléter
const int Ad6 =          ;                    // A compléter

int Adresse_DMX;


// the setup routine runs once when you press reset:
void setup() {

  // Set led pin as output pin
  pinMode ( ledPin_Rouge, OUTPUT );
  pinMode ( ledPin_Vert, OUTPUT );
  pinMode ( ledPin_Bleu, OUTPUT );

  pinMode ( Ad1,          );                    // A compléter
  pinMode ( Ad2,          );                    // A compléter
  pinMode ( Ad3,          );                    // A compléter
  pinMode ( Ad4,          );                    // A compléter
  pinMode ( Ad5,          );                    // A compléter
  pinMode ( Ad6,          );                    // A compléter

  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop()
{
  Adresse_DMX=                                  ; //A compléter
  Serial.print(Adresse_DMX);
  delay(1000);

  if (Adresse_DMX!=0)
     {
     }
  else
     {
     }
}
```