AUTEUR : Leroy Jeremy	**TITRE :** Fabrication d'un contrôleur Arduino 240 pixel
VERSION : 1	**DATE :** 22/05/2018

Matériel :

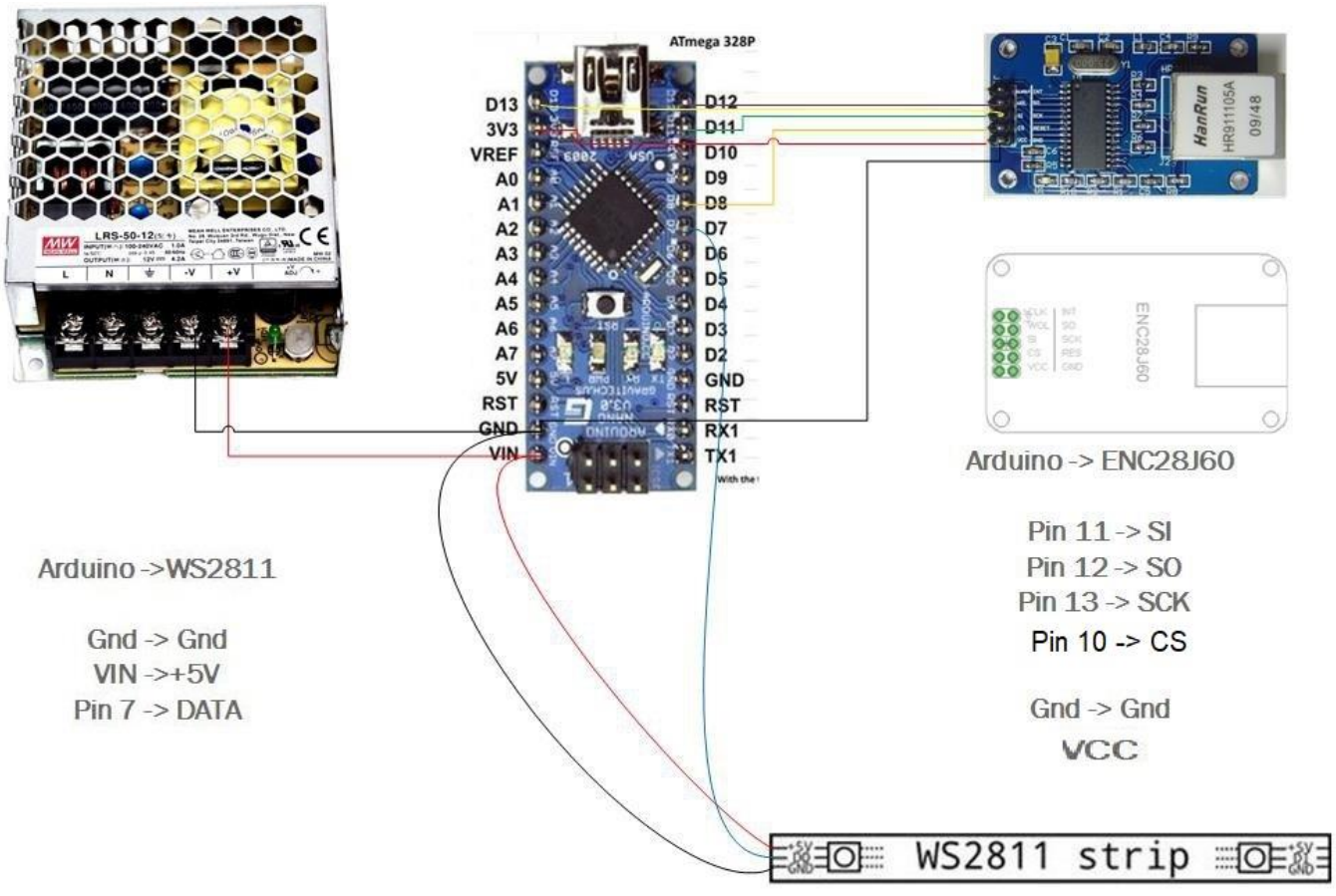
- Arduino Nano/Uno (limité a 240 pixel), Méga (limité à 480 pixel), sans ralentissements
- Module Ethernet ENC28J60

Explications :

Après de multiples recherches pour réaliser un module Ethernet pour la gestion de pixels, voici un résumé, et tuto de ce qu'il est possible de faire avec peu de matériel, et pour moins de 5€. Le code a injecter sera expliqué par la suite, il n'est pas complexe a comprendre ou modifier.

La première chose a retenir est qu'un univers sur Arduino est limité a 120 pixels, la RAM de celui-ci étant limitée. De plus, le buffer pour la gestion de l'Ethernet est a prendre en compte, ce qui limite l'utilisation.

Montage :



Brochage :

GPIO	Uno/Nano	Méga
MOSI	11	51
MISO	12	50
CS	10	53
SCK	13	52

- Ne pas oublier GND et VCC

Explication du code :

#define ETHERNET_BUFFER 540	// Ne pas toucher, Buffer du connecteur Ethernet
#define CHANNEL_COUNT 360	// Ne pas toucher, valeur de comptage de l'Arduino (120 x 3 couleurs)
#define NUM_LEDS 225	// Nombre de Pixel que vous utilisez, Uno/Nano max=240, Méga max=480.
#define UNIVERSE_COUNT 2	// Nombre d'univers, Uno/Nano max=2, Méga max=4.
#define LEDS_PER_UNIVERSE 120	// Nombre de Pixel par univers, ne pas dépasser 120 sur tous les Arduino
#define BRIGHTNESS 70	// Intensité lumineuse
#define DATA_PIN 7	// Sortie DATA de l'Arduino

Code complet à injecter dans l'Arduino :

```
#include <EtherCard.h>
#include <Adafruit_NeoPixel.h>

#define DMX_SUBNET 0
#define DMX_UNIVERSE 1

const byte mymac[] = { 0x74, 0x69, 0x69, 0x2D, 0x30, 0x17 };
static byte myip[] = { 192, 168, 0, 230 };

// OK LE 21-05-2018

#define ETHERNET_BUFFER 540 //PAS TOUCHER
#define CHANNEL_COUNT 360 //PAS TOUCHER                              ||360||
#define NUM_LEDS 225 // NOMBRE DE LED QUE VOUS VOULEZ UTILISER       ||150||225||
#define UNIVERSE_COUNT 2 // NOMBRE D'UNIVERS                        ||  2||
#define LEDS_PER_UNIVERSE 120 // NOMBRE DE LED PAR UNIVERS           || 75||120||
#define BRIGHTNESS 70 // INTENSITE LIUMINEUSE

#define DATA_PIN 7
```

```
//**************************************************************************

  Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_LEDS, DATA_PIN, NEO_RGB + NEO_KHZ800);

  byte Ethernet::buffer[ETHERNET_BUFFER]; // tcp/ip send and receive buffer


  void setup() {

    strip.begin();
    strip.setBrightness(BRIGHTNESS);
    strip.show();

    ether.begin(sizeof(Ethernet::buffer), mymac);
    ether.staticSetup(myip);
    ether.udpServerListenOnPort(&sACNPacket, 5568);

    initTest();
  }


  void loop() {
    ether.packetLoop(ether.packetReceive());
  }


  static void sACNPacket(unsigned int port, byte ip[4], unsigned int i, const char *data, unsigned int
len) {
    int count = checkACNHeaders(data, len);
    if (count) {
      // It is so process the data to the LEDS
      sacnDMXReceived(data, count);
    }
  }


  void sacnDMXReceived(const char* pbuff, int count) {
    if (count > CHANNEL_COUNT) count = CHANNEL_COUNT;
    byte b = pbuff[113]; //DMX Subnet
    if ( b == DMX_SUBNET) {
      b = pbuff[114];  //DMX Universe
      if ( b >= DMX_UNIVERSE && b <= DMX_UNIVERSE + UNIVERSE_COUNT ) {
        if ( pbuff[125] == 0 ) {  //start code must be 0
          int ledNumber = (b - DMX_UNIVERSE) * LEDS_PER_UNIVERSE;
```

```
        for (int i = 126; i < 126 + count; i = i + 3) {
          byte charValueR = pbuff[i];
          byte charValueG = pbuff[i + 1];
          byte charValueB = pbuff[i + 2];
  uint32_t c = strip.Color(int(charValueG), charValueR, charValueB);
          strip.setPixelColor(ledNumber, c);
          ledNumber++;
        }
        strip.show();
       }
      }
     }
  }

  int checkACNHeaders(const char* messagein, int messagelength) {
    if ( messagein[1] == 0x10 && messagein[4] == 0x41 && messagein[12] == 0x37) {
      int addresscount = (byte) messagein[123] * 256 + (byte) messagein[124]; // number of values
plus start code
      return addresscount - 1; //Return how many values are in the packet.
    }
    return 0;
  }


  void initTest() //runs at board boot to make sure pixels are working
  {
    colorWipe(strip.Color(255, 0, 0), 10);
    colorWipe(strip.Color(0, 0, 0, 255), 10);
  }

  void colorWipe(uint32_t c, uint8_t wait) {
    for (uint16_t i = 0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, c);
      strip.show();
      delay(wait);
    }
  }
```

Avantages de ce montage :

- Peu chère, de 3 à 8 € selon les modèles d'Arduino (Uno-Nano-Méga)
- Modification facile des programmes
- Idéale pour apprendre
- Facile à mettre en œuvre

Inconvénients :

- Limitation matérielle, 240/480 pixel max, pour un fonctionnement sans ralentissements.