

Recevoir des fonctions avec paramètres sur le port série, des fonctions avec chaînes de caractères, etc.. à l'aide de ma librairie Arduino Utils.



Ateliers Arduino

par X. HINAULT

www.mon-club-elec.fr



Tous droits réservés – 2012.

Ce document légèrement payant est soumis au droit d'auteur et est réservé à l'usage personnel.

Afin d'encourager la production de supports didactiques de qualité, ce document est légèrement payant.

La licence d'utilisation est attribuée pour un usage personnel uniquement, dans le cercle familial. Mise en ligne et diffusion non autorisées.

Si vous n'êtes pas le détenteur de la licence attribuée pour l'usage de ce document, soyez sympa, merci d'acheter votre exemplaire personnel ici : <https://monclubelec.dpdcart.com/>

Pour tout problème lié à l'utilisation de ce document, veuillez envoyer une copie ici : support@mon-club-elec.fr

Pour obtenir tout autres types de licence d'utilisation (enseignement, commercial, etc...), veuillez contacter l'auteur ici : support@mon-club-elec.fr

Vous avez constaté une erreur ? une coquille ? N'hésitez pas à nous le signaler à cette adresse : support@mon-club-elec.fr

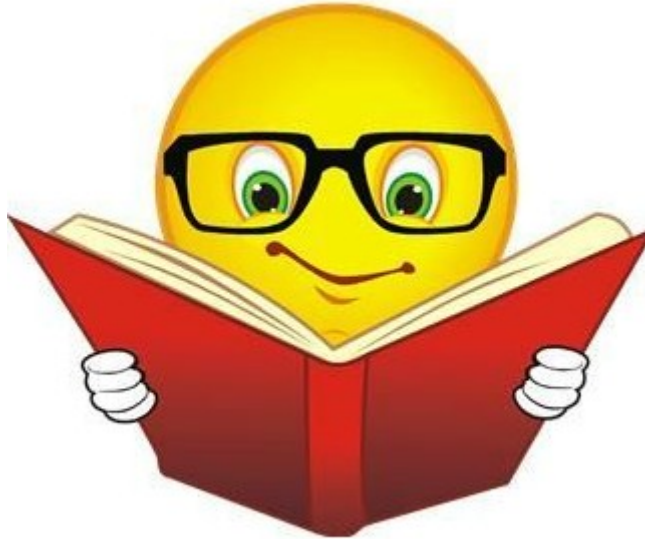
Truc d'utilisation : visualiser ce document en mode diaporama dans le visionneur PDF. Navigation avec les flèches HAUT / BAS ou la souris.

En mode fenêtre, activer le panneau latéral vous facilitera la navigation dans le document. Bonne lecture !

Lancer également le logiciel Arduino et connecter votre carte Arduino afin de pouvoir tester au fur et à mesure les codes d'exemples !

1. Intro

L'objectif ici est d'apprendre à utiliser ma librairie Arduino Utils qui simplifie la réception de chaîne avec paramètres sur le port série afin d'être en mesure de contrôler Arduino à l'aide de chaînes reçues sur le port série intégrant des paramètres numériques ou texte.



Prêt ? C'est parti !

2. Matériel nécessaire pour les ateliers Arduino

Pour cet atelier, vous aurez besoin de tout ou partie des éléments suivants pour pouvoir réaliser les exemples proposés :

De l'espace de développement Arduino

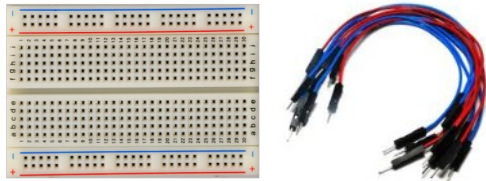


L'espace de développement Arduino associe :

- un ordinateur sous Windows, Mac Os X ou Gnu/Linux (Ubuntu)
- avec le logiciel Arduino installé (voir : <http://www.arduino.cc/>)
- un câble USB
- une carte Arduino UNO ou équivalente.

disponible chez : <http://shop.snootlab.com/> ou <http://www.gotronic.fr/>

Du nécessaire pour réaliser des montages sans soudure

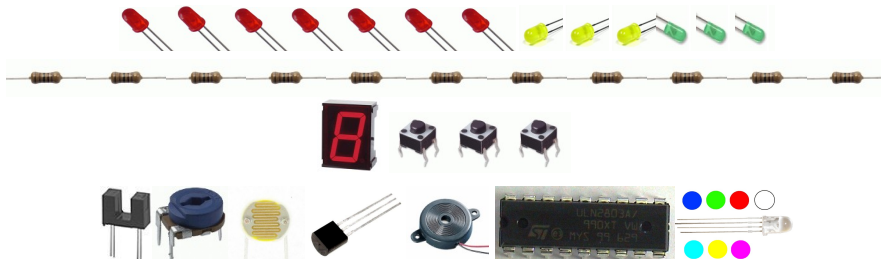


Pour réaliser des montages sans soudure, vous aurez besoin :

- d'une plaque d'essai ou breadboard moyenne (450 points)
- de quelques câbles souples (ou jumpers) mâle/mâle

disponible chez : <http://www.gotronic.fr/>

De quelques composants de base



Pour vous simplifier la vie, nous avons négocié ce kit pour vous !

Vous pouvez commander ce kit complet directement en 1 clic chez notre partenaire

<http://www.gotronic.fr/> avec le code express **701710**

GO TRONIC
ROBOTIQUE ET COMPOSANTS ÉLECTRONIQUES

Pour plus de détails, voir : http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS

Pour les ateliers Arduino niveau débutant, vous devrez idéalement disposer des composants suivants :

- des LEDs 5mm Rouges(x20), Vertes (x5) et 3 Jaunes (x5)
- digit à cathode commune rouge 13mm (x1)
- Résistances (1/4w - 5%) de 270 Ohms (x20), 4,7K Ohms (x1), 1K Ohms (x1)
- mini bouton-poussoir (x3)
- Opto-fourche (x 1)
- Résistance variable linéaire 10K (x 1)
- Photo-résistance 7mm (x 1)
- Capteur de température LM35DZ (-55/+150°C - 10mV/°C) (x 1)
- Capsule son piézoélectrique (x 1)
- ULN 2803A (CI amplificateur 8 voies, 500mA/ voie) (x 1)
- LED 5mm multicolore RVB cathode commune (x 1)

3. *Matériel mécanique utilisé ici ?*

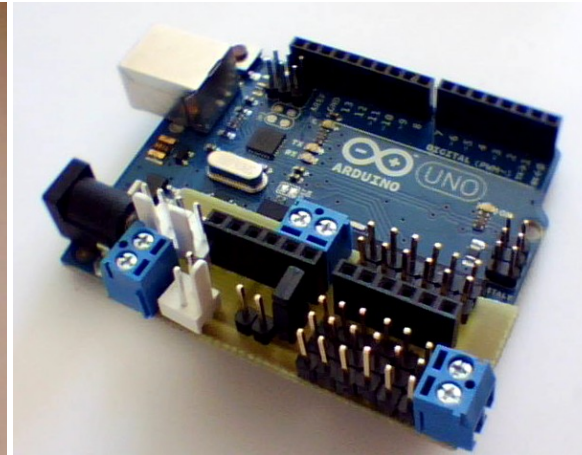
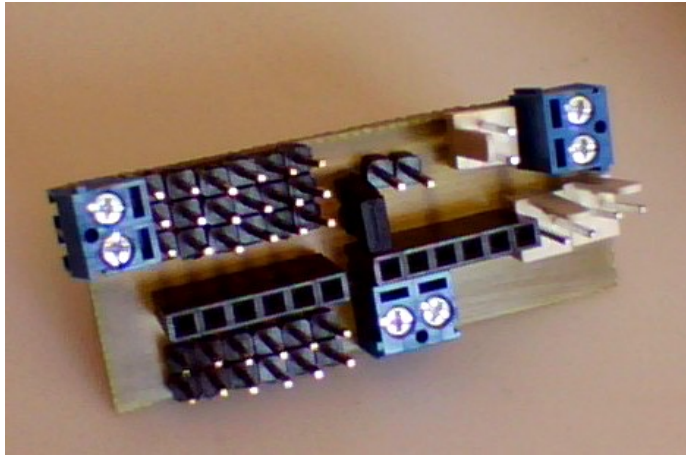
Tourelle pan-tilt

- Constituée de 2 servomoteurs standards montés l'un sur l'autre :
 - l'un assurant la rotation « pan » = panoramique
 - l'autre assurant la rotation « tilt » = inclinaison



+/- un mini-shield « bornier analogique »

- pour assurer une connexion facilitée des servomoteurs avec la carte Arduino :



4. Recevoir des chaînes avec paramètres : Installation et présentation de ma librairie Utils

La librairie Utils

- La librairie Utils, dans laquelle j'ai rassemblé plusieurs fonctions utiles, permet de recevoir et d'extraire des paramètres numériques au sein de chaînes texte reçues notamment sur le port série.
- Le très gros avantage de cette librairie est de simplifier grandement le code qui serait beaucoup plus lourd pour obtenir le même résultat uniquement avec des fonctions Arduino de base.

Télécharger la librairie

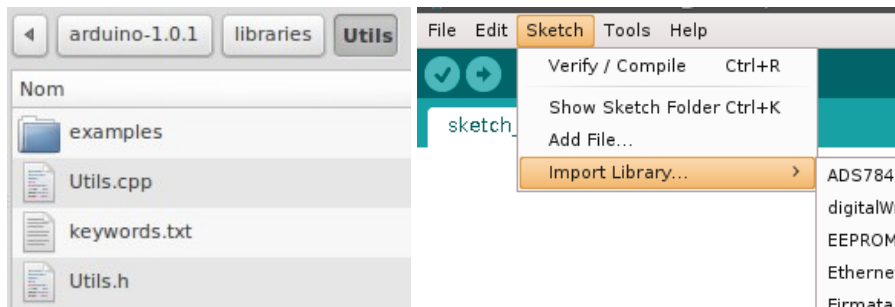
- Ma librairie Utils est disponible ici : http://www.mon-club-elec.fr/pmwiki_reference_lib_arduino_perso/pmwiki.php?n=Main.HomePage

Documentation de la librairie

- http://www.mon-club-elec.fr/pmwiki_reference_lib_arduino_perso/pmwiki.php?n=Main.HomePage

Installation

- Télécharger l'archive. au format zip ou autre. L'extraire
- **Vérifier que le nom du répertoire de la librairie est strictement le même que le nom du fichier *.h ou *.cpp principal. Corriger au besoin. Ici le nom est Utils**
- Copier/coller le répertoire de la librairie dans le répertoire libraries de votre répertoire Arduino
- Relancer Arduino et vérifier que la librairie est présente dans le menu **Sketch > ImportLibrary**.



Le constructeur principal

- Le constructeur principal se nomme Utils et est de la forme :

Utils utils;

Fonctions de la librairie

Fonctions de réception de chaîne de caractères sur le port Série :

- String [waitingString](#) (boolean debugIn) : réception d'une chaîne sur le port Série
- String [waitingString](#) () : réception d'une chaîne sur le port Série
- void [waitForString](#)(String stringForWaitIn) : attente de la réception d'une chaîne précise sur le port Série

Fonctions d'analyse de chaîne de caractères :

- String [testInstructionString](#) (String chaîneTest, String chaîneRefIn): extraction d'un paramètre texte
- boolean [testInstructionLong](#) (String chaîneReception,String chaîneTest, int nbParam, long paramsIn[]): extraction d'un ou plusieurs paramètres entiers

Code d'exemple

```
#include <Utils.h> // inclusion de la librairie
Utils utils; // déclare objet racine d'accès aux fonctions de la librairie Utils

String chaîneReception=""; // déclare un String
long params[6]; // déclare un tableau de long - taille en fonction nombre max
paramètres attendus

void setup() {

    Serial.begin(115200); // Initialisation vitesse port Série
    // Utiliser vitesse idem coté Terminal série
    Serial.println("Saisir une chaîne de la forme FONCTION(valeur)"); // message
    initial

} // fin setup

void loop() {

    chaîneReception=utils.waitingString();// sans debug

    if (chaîneReception!="") { // si une chaîne a été reçue

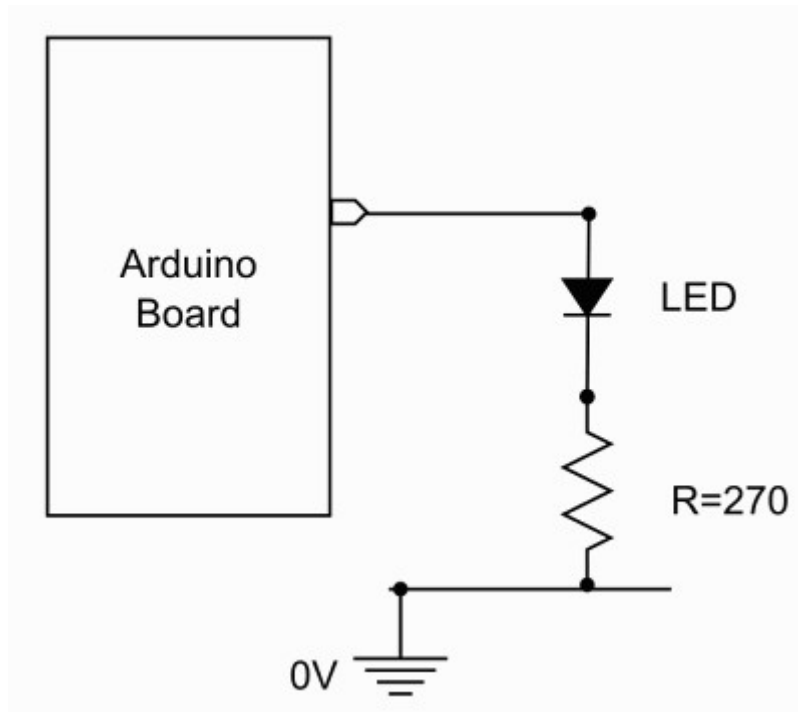
        if(utils.testInstructionLong(chaîneReception,"FONCTION(",1,params)==true)
        { // si chaîne FONCTION(valeur) bien reçue

            Serial.println("Arduino a reçu le parametre : " + (String)params[0]);

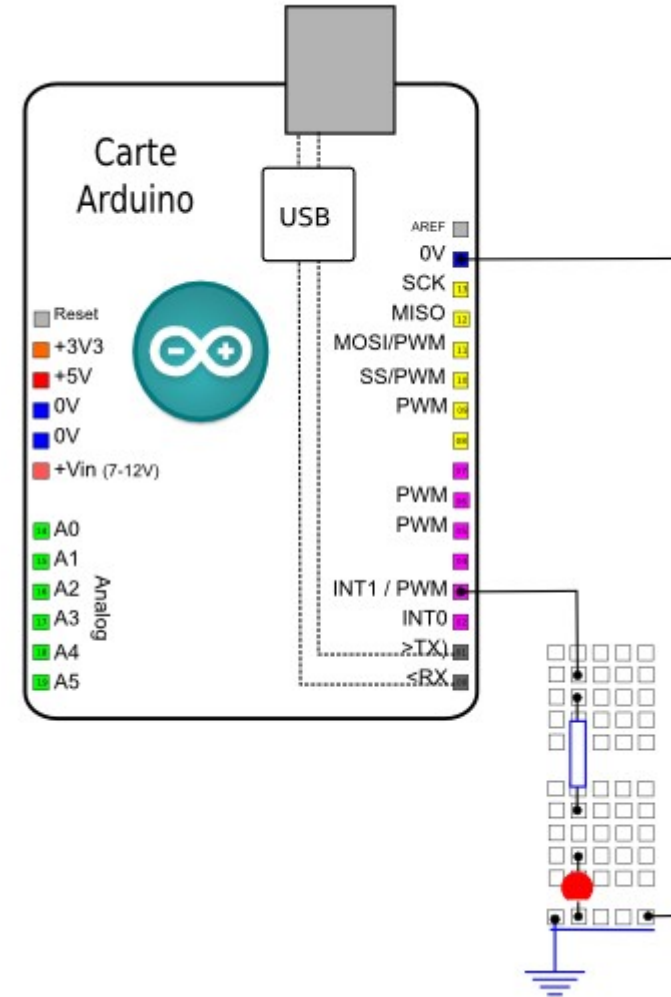
        } // fin si testInstructionLong==true
    } // fin // si une chaîne a été reçue
} // fin loop
```

5. Recevoir sur le port série une chaîne avec un paramètre numérique : le montage

On va réutiliser le premier montage réalisé avec la carte Arduino, ce qui ne devrait pas vous poser de problème, mais ici sur la broche 3 qui dispose de l'impulsion PWM.



Comme vu précédemment, si on désire une intensité de 13mA dans la LED, on utilisera, d'après la loi d'ohm, une résistance de $R = U/I = 3,5V/0,013A = 270 \text{ Ohms}$.



6. Recevoir sur le port série une chaîne avec un paramètre numérique : le programme

Ce qu'on va faire ici...

- La première chose nécessaire ici est d'être en mesure de recevoir sur le port série une chaîne au format chaîne(val1, val2) et d'extraire automatiquement les valeurs ainsi reçues.
- Plusieurs possibilités :
 - soit tout coder de zéro,
 - soit utiliser la librairie RunF déjà présentée par ailleurs,
 - soit utiliser ma librairie Arduino Utils qui permet une gestion simplifiée de la réception de chaîne avec paramètres numériques
- Les librairies proposées ici ne sont pas des librairies standards, et elles devront donc être installées au préalable. Ma librairie **Utils** est disponible ici : http://www.mon-club-elec.fr/pmwiki_reference_lib_arduino_perso/pmwiki.php?n=Main.HomePage
- Le code est disponible ici : <https://gist.github.com/sensor56/602336382609fce7ee52>

Entête déclarative

Inclusion des librairies utiles

- On commence par inclure la librairie Utils

Déclaration broche utilisée

- On déclare une constante de broche pour la LED utilisée

Variables utiles

- On déclare :
 - un objet String pour la stocker la chaîne reçue en réception
 - un tableau de long destiné à stocker les paramètres reçus en réception : **la taille du tableau fixe le nombre maximum de paramètres reçus.**

```
#include <Utils.h> // inclusion de la librairie

Utils utils; // déclare objet racine d'accès aux fonctions de la librairie Utils

String chaineReception=""; // déclare un String
long params[6]; // déclare un tableau de long - taille en fonction nombre max paramètres attendus

int LED=3; // broche de la LED
```


Fonction **setup()**

Configuration broche utilisée

- On configure la broche de LED en sortie

Initialisation série

- On initialise la communication série à 115200 bauds

Code initial

- On affiche un simple message invitant à saisir une chaîne de la forme LED(pwm)

```
void setup() {  
  pinMode(LED,OUTPUT); // met la broche en sortie  
  
  Serial.begin(115200); // Initialisation vitesse port Série  
  // Utiliser vitesse idem coté Terminal série  
  
  Serial.println("Saisir une chaîne de la forme LED(valeurPWM) avec valeurPWM entre 0 et 255 :"); // message initial  
}  
// fin setup
```


Fonction **loop()**

Réception de la chaîne sur le port série

- La librairie Utils intègre la fonction `waitingString()` qui attend sur le port série une chaîne jusqu'à l'arrivée d'un saut de ligne (« \n ») : la chaîne reçue est stockée dans un objet `String`

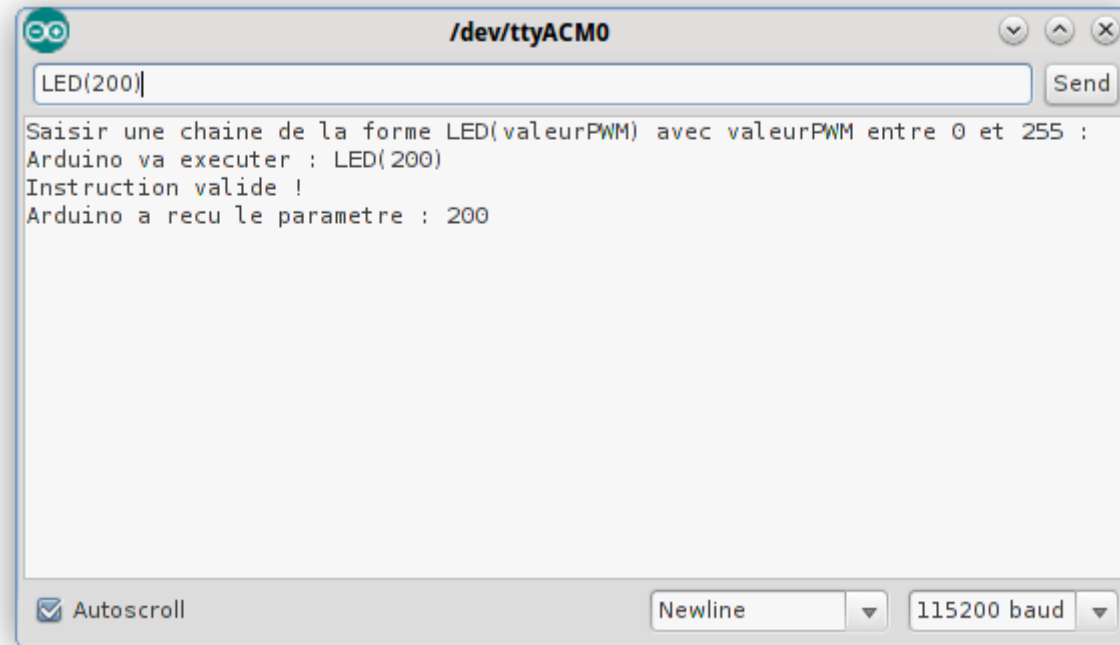
Extraction des paramètres numériques

- Ensuite, à l'aide d'une fonction, on teste si la chaîne n'est pas vide :
 - on appelle alors la fonction `testInstructionLong()` en lui passant en paramètre :
 - la chaîne reçue,
 - la racine de la chaîne à reconnaître
 - le nombre de paramètre
 - le tableau de stockage
 - la fonction renvoie `true` si l'analyse est correcte, autrement dit qu'une chaîne `LED(valeur)` a bien été reçue
- Il ne reste plus alors qu'à appliquer l'impulsion voulue sur la broche de la LED avec l'instruction `analogWrite()`

```
void loop() {  
    //chaîneReception=utils.waitingString(true);// avec debug  
    chaîneReception=utils.waitingString();// sans debug  
  
    if (chaîneReception!="") { // si une chaîne a été reçue  
        if(utils.testInstructionLong(chaîneReception,"LED(",1,params)==true) { // si chaîne LED(valeur) bine reçue  
            Serial.println("Arduino a reçu le parametre : " + (String)params[0]);  
            analogWrite(LED,params[0]); // on applique la valeur PWM sur la LED  
        } // fin si testInstructionLong==true  
    } // fin // si une chaîne a été reçue  
} // fin loop
```

Fonctionnement du programme

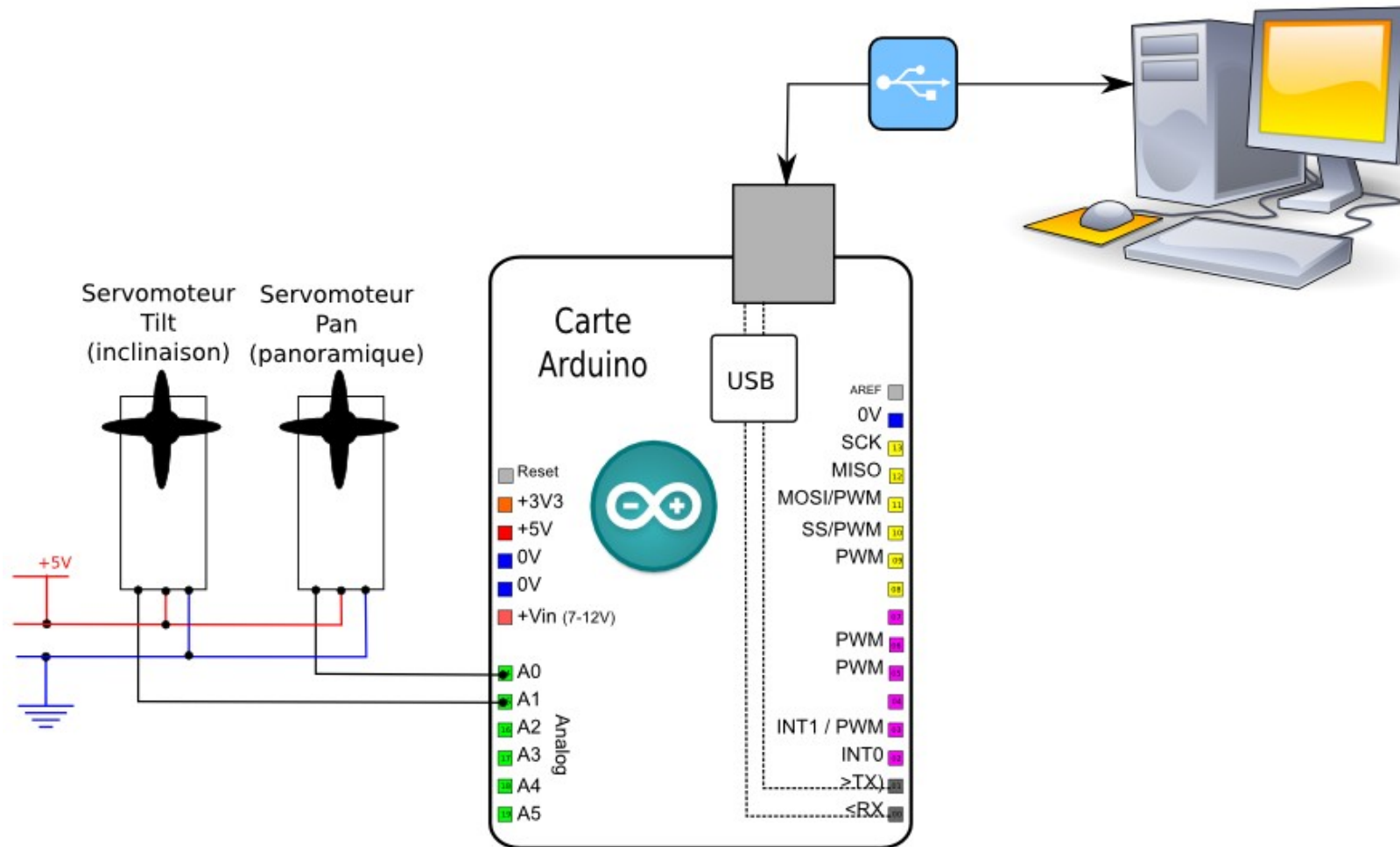
- Une fois la carte Arduino programmée, ouvrir le Terminal Série en réglant sur « newline » et « 115200 ».
- Puis saisir une chaîne de la forme LED(pwm) avec pwm une valeur comprise entre 0 et 255
- ce qui donne :



Au final, un code léger qui offre une fonctionnalité assez puissante en pratique.

7. Contrôler 2 servomoteurs par une chaîne avec valeur numérique reçue sur le port série : le montage

- Ici, on va utiliser 2 servomoteurs standards, connectés chacun sur 2 broches de la carte Arduino :



8. Contrôler 2 servomoteurs par une chaîne avec valeur numérique reçue sur le port série : le programme

Ce qu'on va faire ici...

- Une fois que l'on est en mesure de recevoir une chaîne avec des paramètres numériques, il devient possible de positionner facilement des servomoteurs. J'utilise ici à nouveau ma librairie Arduino Utils qui permet une gestion simplifiée de la réception de chaîne avec paramètres numériques
- Ici, nous allons définir la reconnaissance de 2 chaînes sous la forme chaine(val1) c'est à dire avec un seul paramètre numérique.
- Ce code est disponible ici : <https://gist.github.com/sensor56/479c5f8b8f69d5664501>

Entête déclarative

Inclusion des librairies utiles

- On commence par inclure la librairie Utils

Déclaration broche utilisée

- On déclare les constantes de broches de 2 servomoteurs

Variables et objets utiles

- On déclare :
 - un objet racine permettant l'accès aux fonctions de la librairie Utils ,
 - un objet String pour stocker la chaîne reçue en réception
 - un tableau de long destiné à stocker les paramètres reçus en réception : [la taille du tableau fixe le nombre maximum de paramètres reçus.](#)
 - les constantes de position des servomoteurs
 - les 2 objets Servo représentant les servomoteurs
 - 2 variables de position du servomoteur

```
#include <Utils.h> // inclusion de la librairie
#include <Servo.h> // inclut la librairie Servo

Utils utils; // déclare objet racine d'accès aux fonctions de la librairie Utils

String chaineReception=""; // déclare un String
long params[6]; // déclare un tableau de long - taille en fonction nombre max paramètres attendus

//----- constantes de paramétrage du servomoteur -----
const int posMin=550; // largeur impulsion en µs correspondant à la position 0° du servomoteur
const int posMax=2350; // largeur impulsion en µs correspondant à la position 180° du servomoteur
//----- valeur pour un Futaba S3003 - à adapter à votre situation

const int brocheServoPan=14; // broche du servomoteur
const int brocheServoTilt=15; // broche du servomoteur

Servo servoPan; // déclaration d'un objet servomoteur
Servo servoTilt; // déclaration d'un objet servomoteur

int angleServoPan=0; // variable de position du servomoteur
int angleServoTilt=0; // variable de position du servomoteur
```

Fonction **setup()**

Initialisation série

- On initialise la communication série à 115200 bauds

Initialisation des servomoteurs

- On initialise les servomoteurs en précisant les angles maximum et minimum, on précise les broches utilisées.

Code initial

- On affiche un simple message invitant à saisir une chaîne de la forme voulue :

```
void setup() {  
  
  Serial.begin(115200); // Initialisation vitesse port Série  
  // Utiliser vitesse idem coté Terminal série  
  
  servoPan.attach(brocheServoPan, posMin, posMax); // attache le servomoteur à la broche  
  // et initialisation des positions extremes  
  servoTilt.attach(brocheServoTilt, posMin, posMax); // attache le servomoteur à la broche  
  
  Serial.println("Saisir une chaine de la forme servoPan(angle) / servoTilt (angle) avec angle entre 0 et 180 :"); // message initial  
  
} // fin setup
```

Fonction **loop()**

Réception de la chaîne sur le port série

- La librairie **Utils** intègre la fonction **waitingString()** qui attend sur le port série une chaîne jusqu'à l'arrivée d'un saut de ligne (« \n ») : la chaîne reçue est stockée dans un objet String

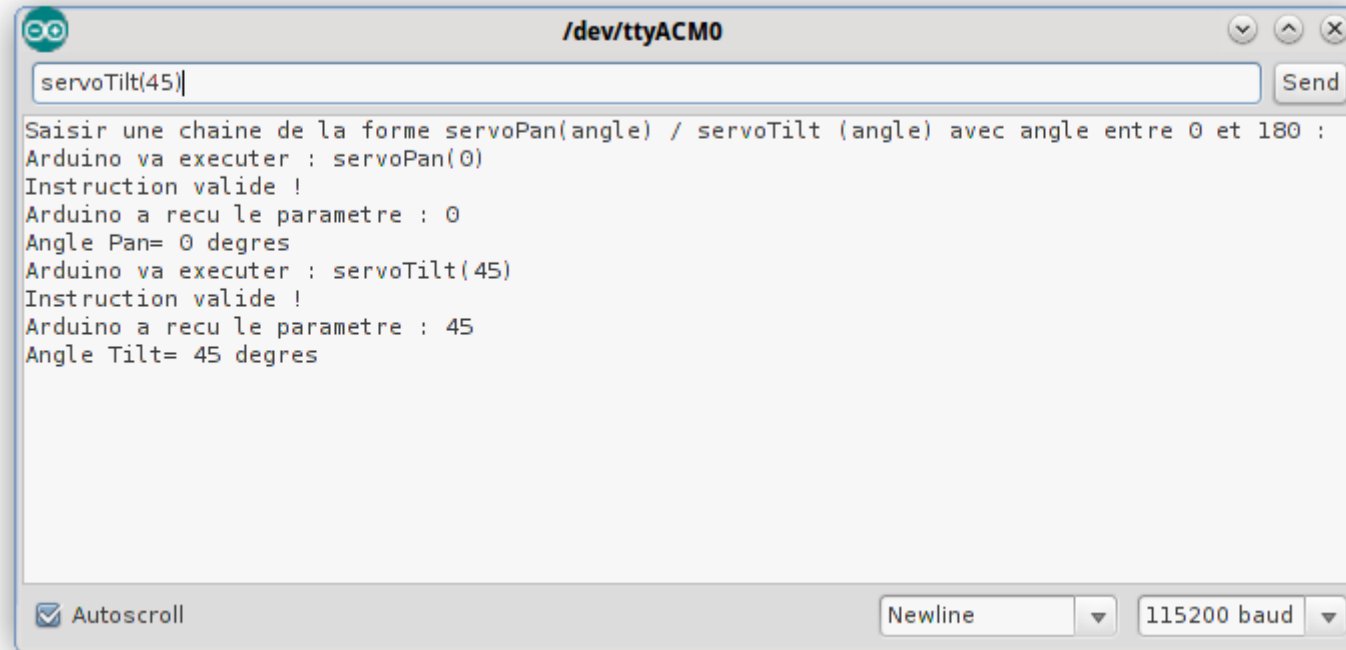
Extraction des paramètres numériques

- Ensuite, à l'aide d'une fonction, on teste si la chaîne n'est pas vide :
 - on appelle alors la fonction **testInstructionLong()** en lui passant en paramètre :
 - la chaîne reçue,
 - la racine de la chaîne à reconnaître
 - le nombre de paramètre
 - le tableau de stockage
 - la fonction renvoie **true** si l'analyse est correcte, autrement dit qu'une chaîne racine(valeur) a bien été reçue
- Il ne reste plus alors qu'à positionner le servomoteur à la position voulue :

```
void loop() {  
  
  //chaîneReception=utils.waitingString(true);// avec debug  
  chaîneReception=utils.waitingString();// sans debug  
  
  if (chaîneReception!="") { // si une chaîne a été reçue  
  
    //--- gestion servoPan(xxx)  
    if(utils.testInstructionLong(chaîneReception,"servoPan(",1,params)==true) { // si chaîne LED(valeur) bien reçue  
      Serial.println("Arduino a reçu le parametre : " + (String)params[0]);  
      angleServoPan=params[0] ;  
      servoPan.write(angleServoPan); // positionne le servomoteur à l'angle voulu  
      Serial.print ("Angle Pan= ");  
      Serial.print (angleServoPan); // --- affiche la valeur de l'angle utilisé  
      Serial.println (" degres");  
  
    } // fin si testInstructionLong==true  
  
    //--- gestion servoPan(xxx)  
    if(utils.testInstructionLong(chaîneReception,"servoTilt(",1,params)==true) { // si chaîne LED(valeur) bien reçue  
      Serial.println("Arduino a reçu le parametre : " + (String)params[0]);  
      angleServoTilt=params[0] ;  
      servoTilt.write(angleServoTilt); // positionne le servomoteur à l'angle voulu  
      Serial.print ("Angle Tilt= ");  
      Serial.print (angleServoTilt); // --- affiche la valeur de l'angle utilisé  
      Serial.println (" degres");  
  
    } // fin si testInstructionLong==true  
  
  } // fin // si une chaîne a été reçue  
  
} // fin loop
```

Fonctionnement du programme

- Une fois la carte Arduino programmée, ouvrir le Terminal Série en réglant sur « newline » et « 115200 ».
- Puis saisir une chaîne de la forme servoPan(angle) ou servoTilt(angle) avec une valeur comprise entre 0 et 255
- ce qui donne :

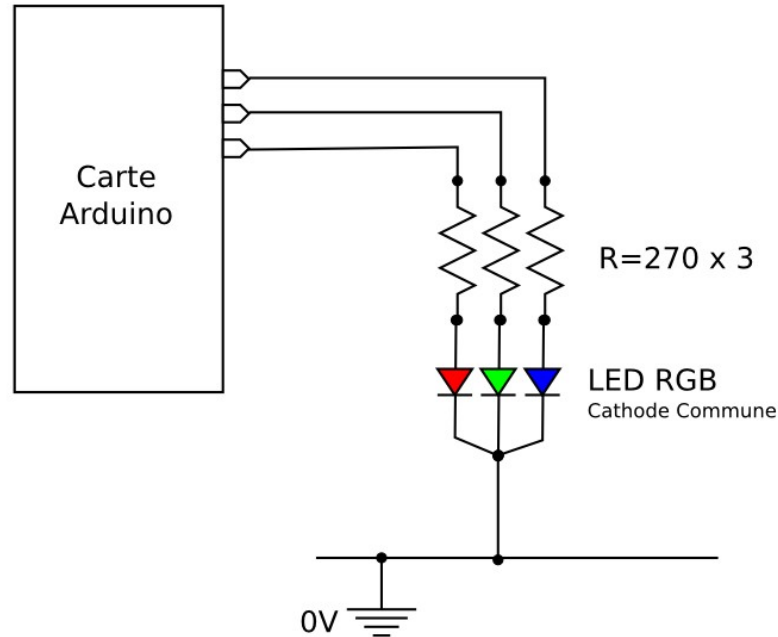


Au final, un code léger qui offre une fonctionnalité assez puissante en pratique.

9. Réception d'une chaîne avec plusieurs paramètres numériques par le port série : le montage

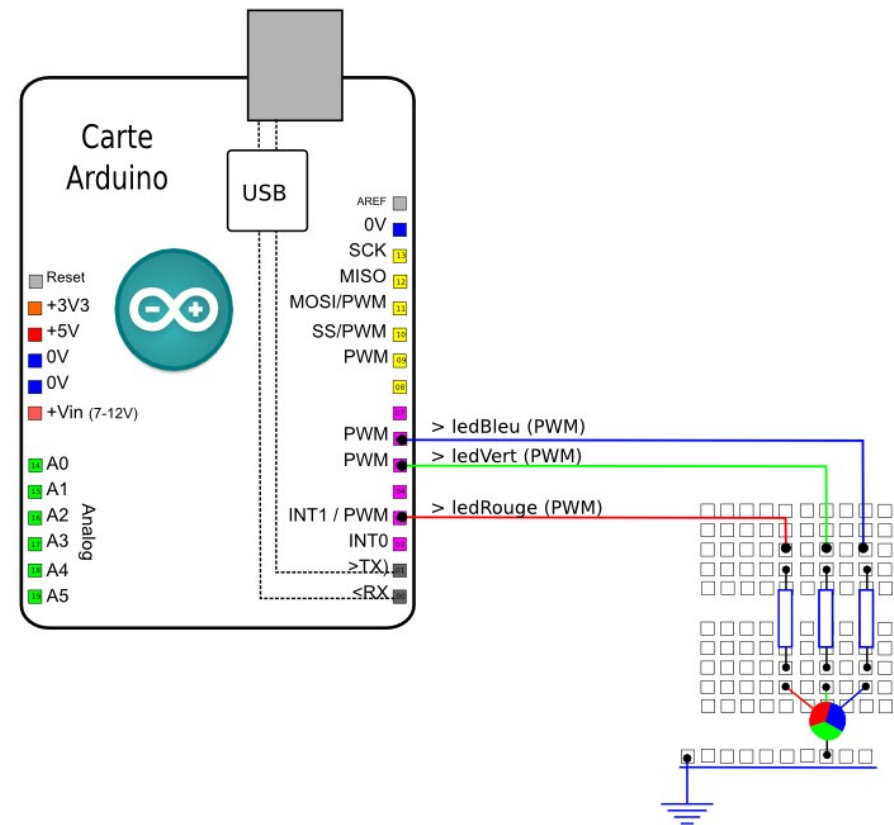
Le principe d'utilisation de la LED multicolore est le même que celui d'une LED, ou plutôt de 3 LEDs avec une broche commune :

- la broche commune sera connectée au 0V (LED RGB à cathode commune) ou au +5V (LED RGB à anode commune)
- chaque LED sera connectée sur une broche de E/S de la carte Arduino configurée en sortie via une résistance pour fixer l'intensité circulant dans la LED.



Pour les matheux (on est pas du tout obligé de savoir faire ce calcul !) :

- aux bornes de la LED, la tension vaut 1,5V environ (fixe)
- la tension aux bornes de la résistance en série avec la LED, dans le cas d'une alimentation en 5V, vaudra donc $5V - 1,5V = 3,5V$
- si on désire une intensité de 13mA dans la LED, on utilisera, d'après la loi d'ohm, une résistance de $R = U/I = 3,5V / 0,013A = 270 \text{ Ohms}$.



Montage de la LED multicolore RGB à cathode commune (0V commun)

ATTENTION :

On va utiliser 3 broches E/S particulières ayant le signe ~ devant le numéro sur la carte Arduino : ces broches ont une fonction spéciale que nous allons utiliser, la modulation de largeur d'impulsion (MLI ou PWM en anglais).

Connecter la broche **R** sur la broche **3**, la broche **V** sur la broche **5** et la broche **B** sur la broche **6**.

Voir le tuto dédié aux impulsions pour plus de détails.

10. Réception d'une chaîne avec plusieurs paramètres numériques par le port série : programme

Ce qu'on va faire ici...

- Je vous propose de poursuivre la découverte de ma librairie Arduino Utils qui permet une gestion simplifiée de la réception de chaîne avec paramètres numériques sur le port série. Ici, nous allons définir la reconnaissance d'une chaîne sous la forme chaîne(rouge, vert, bleu) c'est à dire avec 3 paramètres numériques.
- Comme vous allez le constater, cela n'est pas beaucoup plus compliqué que pour 1 seule valeur passée en paramètre.
- Ce code est disponible ici : <https://gist.github.com/sensor56/8c6a5bb3759fbb418499>

Entête déclarative

Inclusion des librairies utiles

- On commence par inclure la librairie Utils

Déclaration broche utilisée

- On déclare les constantes de broches utilisées avec la LED RGB

Variables et objets utiles

- On déclare :
 - un objet String pour la stocker la chaîne reçue en réception
 - un tableau de long destiné à stocker les paramètres reçus en réception : la taille du tableau fixe le nombre maximum de paramètres reçus.
 - un objet racine permettant l'accès aux fonctions de la librairie Utils

```
//---- inclusion de librairie
#include <Utils.h> // librairie personnelle avec plusieurs fonctions utiles

//--- variables pour réception chaîne sur port Série
String chaineReception=""; // déclare un objet String vide
long params[6]; // déclare un tableau de long - taille en fonction nombre max paramètres attendus

//---- constantes des broches utilisées
const int ledR=3; // constante désignant la broche de la LED
const int ledV=5; // constante désignant la broche de la LED
const int ledB=6; // constante désignant la broche de la LED

//--- objets utiles

Utils utils; // objet racine donnant accès aux fonctions de la librairie Utils
```

Fonction **setup()**

Initialisation série

- On initialise la communication série à 115200 bauds

Initialisation des broches

- On initialise les 3 broches utilisées pour le contrôle de la LED RGB (broches PWM obligatoires)

Code initial

- On affiche un simple message invitant à saisir une chaîne de la forme voulue :

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    //----- configuration des broches utilisées -----
    pinMode(ledR, OUTPUT); // met la broche en sortie
    pinMode(ledV, OUTPUT); // met la broche en sortie
    pinMode(ledB, OUTPUT); // met la broche en sortie

    //----- Initialisation Série -----
    Serial.begin(115200); // initialise la vitesse de la connexion série
    //-- utilise la meme vitesse dans le Terminal Série

    //-----
    Serial.println("--- Instructions reconnues: ---");
    Serial.println("rvb(rouge, vert, bleu)");
    Serial.println("Utiliser des valeurs 0 - 255 ");
    Serial.println("-----");

} // fin de la fonction setup()
```

Fonction **loop()**

Réception de la chaîne sur le port série

- La librairie **Utils** intègre la fonction **waitingString()** qui attend sur le port série une chaîne jusqu'à l'arrivée d'un saut de ligne (« \n ») : la chaîne reçue est stockée dans un objet String

Extraction des paramètres numériques

- Ensuite, à l'aide d'une fonction, on teste si la chaîne n'est pas vide :
 - on appelle alors la fonction **testInstructionLong()** en lui passant en paramètre :
 - la chaîne reçue,
 - la racine de la chaîne à reconnaître
 - le nombre de paramètres
 - le tableau de stockage des paramètres
 - la fonction renvoie **true** si l'analyse est correcte, autrement dit qu'une chaîne racine(valeur) a bien été reçue
- Il ne reste plus alors qu'à appeler la fonction **ledRVB** qui permet de mettre la LED RGB dans l'état voulu en lui passant les paramètres extraits :

```
//--- la fonction loop() : exécutée en boucle sans fin
void loop() {

    //--- reception de la chaine sur le port série - fin par saut de ligne
    //chaineReception=utils.waitingString(true);// avec debug
    chaineReception=utils.waitingString();// sans debug

    if (chaineReception!="") { // si une chaine a été reçue

        //--- gestion couleur(rouge,vert,bleu)
        if(utils.testInstructionLong(chaineReception,"rvb(",3,params)==true) { // si chaine couleur,rrr,vvv,bbb) bien reçue - 3 paramètres
            Serial.println("Arduino a reçu les paramètres : " );
            Serial.println(params[0]),Serial.println(params[1]),Serial.println(params[2]);

            ledRVB(params[0],params[1],params[2]); // passe les paramètres à la fonction ledRVB

        } // fin si testInstructionLong==true
    } // fin if chaineReception

} // fin de la fonction loop()
```

Admettez que ce code est tout de même simple pour le résultat obtenu (extraction des valeurs numériques quelque soit leur nombre de chiffre, etc..) !

Et il ne sera pas beaucoup plus compliqué de recevoir de la sorte 5, 10, 20 paramètres numériques via le port série !

Fonction ledRVB()

Pour simplifier le code, on crée une fonction pour gérer l'affichage des couleurs avec la LED RGB :

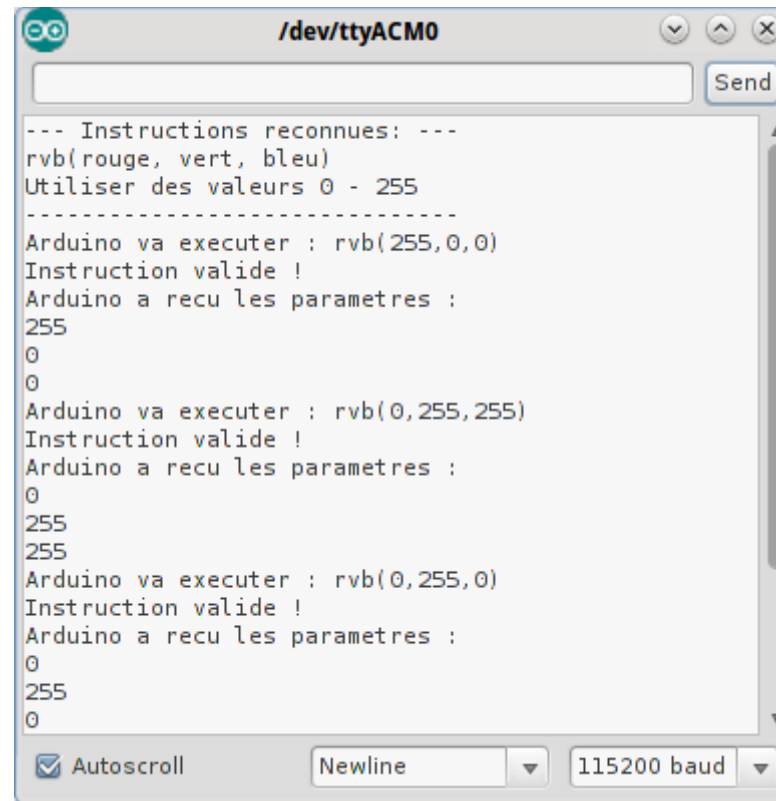
- la fonction ne renvoie rien = type void
- reçoit 3 valeurs entières (0-255) fixant la proportion des couleurs RVB
- à l'aide de 3 valeurs, on crée les impulsions PWM correspondantes
- la pause est intégrée dans cette fonction

Truc : si vous utilisez une LED à anode commune, il suffit de mettre **255-valeur** à la place de **valeur**

```
//---- fonction pour combiner couleurs ----  
void ledRVB(int Rouge, int Vert, int Bleu) {  
  analogWrite(ledR,Rouge); // proportion de rouge  
  analogWrite(ledV,Vert); // proportion de vert  
  analogWrite(ledB,Bleu); // proportion de bleu  
}  
// fin fonction ledRVB
```

Fonctionnement du programme

- Une fois la carte Arduino programmée, ouvrir le Terminal Série en réglant sur « newline » et « 115200 ».
- Puis saisir une chaîne de la forme rvb(rouge, vert, bleu) avec une valeur comprise entre 0 et 255 pour chaque couleur
- ce qui donne :



```
--- Instructions reconnues: ---
rvb(rouge, vert, bleu)
Utiliser des valeurs 0 - 255
-----
Arduino va executer : rvb(255,0,0)
Instruction valide !
Arduino a reçu les paramètres :
255
0
0
Arduino va executer : rvb(0,255,255)
Instruction valide !
Arduino a reçu les paramètres :
0
255
255
Arduino va executer : rvb(0,255,0)
Instruction valide !
Arduino a reçu les paramètres :
0
255
0
```

rvb(255,0,0) : le LED RGB s'allume en rouge, rvb(255,255,0) : le LED RGB s'allume en jaune, etc...

Au final, un code léger qui offre une fonctionnalité assez puissante en pratique.



11. Réception d'une chaîne avec un paramètre texte par le port série : le programme

Ce qu'on va faire ici...

- A présent, nous allons voir comment passer en paramètre, non pas des valeurs numériques, mais du texte. Ici, nous allons définir la reconnaissance d'une chaîne sous la forme chaîne(couleur) c'est à dire avec 1 paramètre texte. Nous utilisons le même montage que précédemment, qui nous sert ici de support pour montrer le principe, mais on pourra transposer sans difficulté à n'importe quelle autre situation : contrôle de moteurs, etc...
- Ici, nous allons définir la reconnaissance des chaînes « rouge », « vert », « bleu », « violet », « jaune », « bleclair », « blanc » passés en paramètre à la fonction rvb()
- Ce code est disponible ici : <https://gist.github.com/sensor56/ea0a1130fe71921f1a29>

Entête déclarative

Inclusion des bibliothèques utiles

- On commence par inclure la bibliothèque Utils

Déclaration broche utilisée

- On déclare les constantes de broches utilisées avec la LED RGB

Variables et objets utiles

- On déclare :
 - un objet String pour la stocker la chaîne reçue en réception
 - un objet String pour le paramètre à extraire
 - un objet racine permettant l'accès aux fonctions de la bibliothèque Utils

```
//---- inclusion de bibliothèque
#include <Utils.h> // bibliothèque personnelle avec plusieurs fonctions utiles

//--- variables pour réception chaîne sur port Série
String chaîneReception=""; // déclare un objet String vide
String param=""; // déclare un String

//---- constantes des broches utilisées
const int ledR=3; // constante désignant la broche de la LED
const int ledV=5; // constante désignant la broche de la LED
const int ledB=6; // constante désignant la broche de la LED

//--- objets utiles
Utils utils; // objet racine donnant accès aux fonctions de la bibliothèque Utils
```


Fonction **setup()**

Initialisation série

- On initialise la communication série à 115200 bauds

Initialisation des broches

- On initialise les 3 broches utilisées pour le contrôle de la LED RGB (broches PWM obligatoires)

Code initial

- On affiche un simple message invitant à saisir une chaîne de la forme voulue :

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    //----- configuration des broches utilisées -----
    pinMode(ledR, OUTPUT); // met la broche en sortie
    pinMode(ledV, OUTPUT); // met la broche en sortie
    pinMode(ledB, OUTPUT); // met la broche en sortie

    //----- Initialisation Série -----
    Serial.begin(115200); // initialise la vitesse de la connexion série
    //-- utilise la meme vitesse dans le Terminal Série

    //-----
    Serial.println("--- Instructions reconnues: ---");
    Serial.println("rvb(couleur)");
    Serial.println("Utiliser couleur parmi : rouge, vert, bleu, jaune, violet, bleuclair");
    Serial.println("-----");

} // fin de la fonction setup()
```

Fonction **loop()**

Réception de la chaîne sur le port série

- La librairie **Utils** intègre la fonction **waitingString()** qui attend sur le port série une chaîne jusqu'à l'arrivée d'un saut de ligne (« \n ») : la chaîne reçue est stockée dans un objet String

Extraction du paramètre texte

- Ensuite, à l'aide d'une condition, on teste si la chaîne n'est pas vide :
 - on appelle alors la fonction **testInstructionString()** en lui passant en paramètre :
 - la chaîne reçue,
 - la racine de la chaîne à reconnaître
 - la fonction renvoie **la chaîne extraite** si l'analyse est correcte, autrement dit qu'une chaîne racine(texte) a bien été reçue
- Il ne reste plus alors qu'à appeler la fonction **ledRVB** avec les valeurs voulues pour chaque chaîne possible, ce qui permet de mettre la LED RGB dans l'état voulu en lui passant les paramètres r,g,b adéquats :

```
//--- la fonction loop() : exécutée en boucle sans fin
void loop() {

  //--- reception de la chaîne sur le port série - fin par saut de ligne
  //chaîneReception=utils.waitingString(true);// avec debug
  chaîneReception=utils.waitingString();// sans debug

  if (chaîneReception!="") { // si une chaîne a été reçue
    //--- gestion rvb(couleur)
    param=utils.testInstructionString(chaîneReception, "rvb(", true); // extrait le paramètre de la chaîne avec debug
    //param=utils.testInstructionString("LED(", chaîneReception); // extrait le paramètre de la chaîne sans debug

    if (param!="") { // si une chaîne a été reçue en paramètre
      Serial.println("Arduino a reçu le parametre : " + param);

      if(param=="rouge") { // si le parametre vaut rouge
        ledRVB(255,0,0); // couleur RGB voulue
      } // fin if
      else if(param=="vert") { // si le parametre vaut vert
        ledRVB(0,255,0); // couleur RGB voulue
      } // fin else if
      else if(param=="bleu") { // si le parametre vaut bleu
        ledRVB(0,0,255); // couleur RGB voulue
      } // fin else if
      else if(param=="jaune") { // si le parametre vaut jaune
        ledRVB(255,255,0); // couleur RGB voulue
      } // fin else if
      else if(param=="violet") { // si le parametre vaut violet
        ledRVB(255,0,255); // couleur RGB voulue
      } // fin else if
      else if(param=="bleuclair") { // si le parametre vaut bleuclair
        ledRVB(0,255,255); // couleur RGB voulue
      } // fin else if
    } // fin if param!="
  } // fin if chaîneReception
} // fin de la fonction loop()
```

Fonction ledRVB()

Pour simplifier le code, on crée une fonction pour gérer l'affichage des couleurs avec la LED RGB :

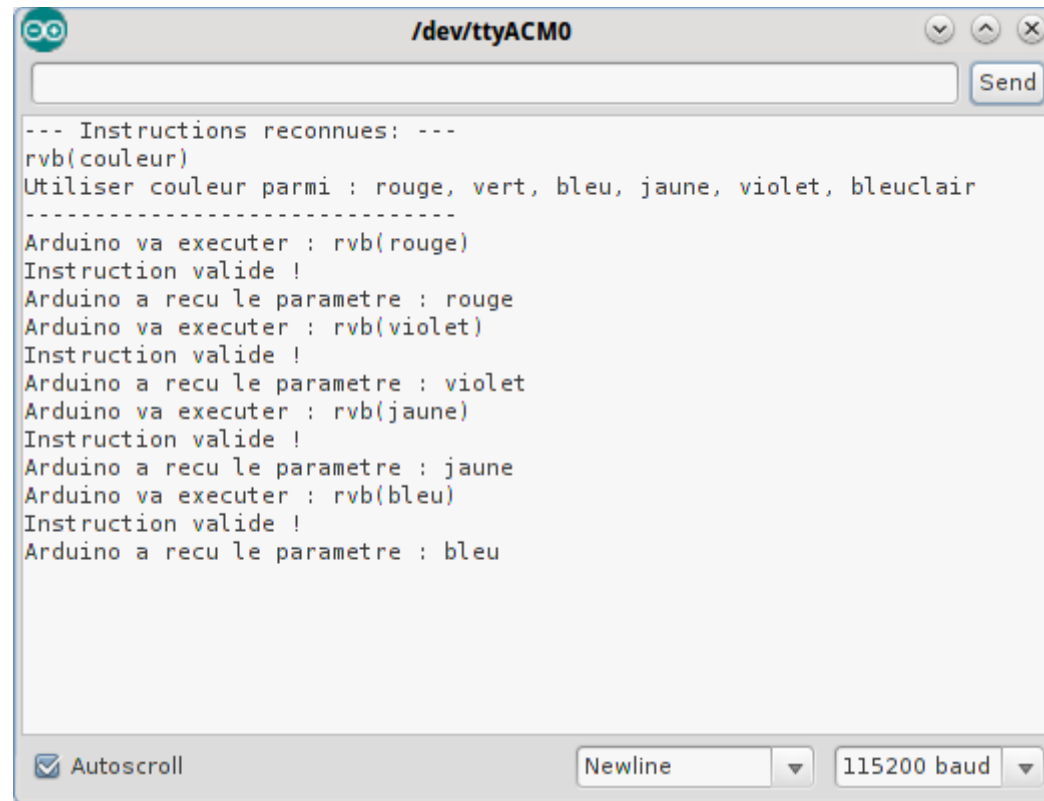
- la fonction ne renvoie rien = type void
- reçoit 3 valeurs entières (0-255) fixant la proportion des couleurs RVB
- à l'aide de 3 valeurs, on crée les impulsions PWM correspondantes
- la pause est intégrée dans cette fonction

Truc : si vous utilisez une LED à anode commune, il suffit de mettre **255-valeur** à la place de **valeur**

```
//---- fonction pour combiner couleurs ----  
void ledRVB(int Rouge, int Vert, int Bleu) {  
    analogWrite(ledR,Rouge); // proportion de rouge  
    analogWrite(ledV,Vert); // proportion de vert  
    analogWrite(ledB,Bleu); // proportion de bleu  
}  
// fin fonction ledRVB
```

Fonctionnement du programme

- Une fois la carte Arduino programmée, ouvrir le Terminal Série en réglant sur « newline » et « 115200 ».
- Puis saisir une chaîne de la forme `rvb(rouge, vert, bleu)` avec une valeur comprise entre 0 et 255 pour chaque couleur
- ce qui donne :



```
--- Instructions reconnues: ---
rvb(couleur)
Utiliser couleur parmi : rouge, vert, bleu, jaune, violet, bleuclair
-----
Arduino va executer : rvb(rouge)
Instruction valide !
Arduino a reçu le parametre : rouge
Arduino va executer : rvb(violet)
Instruction valide !
Arduino a reçu le parametre : violet
Arduino va executer : rvb(jaune)
Instruction valide !
Arduino a reçu le parametre : jaune
Arduino va executer : rvb(bleu)
Instruction valide !
Arduino a reçu le parametre : bleu
```

`rvb(rouge)` : le LED RGB s'allume en rouge, `rvb(jaune)` : le LED RGB s'allume en jaune, etc...

Remarquer avec quelle facilité vous pouvez créer vos propres « instructions » reçues sur le port série.



12. Réception d'une chaîne avec *paramètres texte et numérique* par le port série

Ce qu'on va faire ici...

- Nous allons ici combiner à la fois des paramètres texte et numérique au sein d'une même chaîne reçue sur le port série.
- Ici, nous allons définir la reconnaissance des chaînes « rouge », « vert », « bleu », passés en paramètre à la fonction `rvb()`, associé à une valeur numérique 0-100 correspondant au pourcentage d'intensité de la couleur.
- Ce code est disponible ici :

Entête déclarative

Inclusion des bibliothèques utiles

- On commence par inclure la bibliothèque Utils

Déclaration broche utilisée

- On déclare les constantes de broches utilisées avec la LED RGB

Variables et objets utiles

- On déclare :
 - un objet String pour la stocker la chaîne reçue en réception
 - un objet String pour le paramètre à extraire
 - un objet racine permettant l'accès aux fonctions de la bibliothèque Utils

```
//---- inclusion de bibliothèque
#include <Utils.h> // bibliothèque personnelle avec plusieurs fonctions utiles

//--- variables pour réception chaîne sur port Série
String chaineReception=""; // déclare un objet String vide
String param=""; // déclare un String

//---- constantes des broches utilisées
const int ledR=3; // constante désignant la broche de la LED
const int ledV=5; // constante désignant la broche de la LED
const int ledB=6; // constante désignant la broche de la LED

//--- objets utiles
Utils utils; // objet racine donnant accès aux fonctions de la bibliothèque Utils
```

Fonction **setup()**

Initialisation série

- On initialise la communication série à 115200 bauds

Initialisation des broches

- On initialise les 3 broches utilisées pour le contrôle de la LED RGB (broches PWM obligatoires)

Code initial

- On affiche un simple message invitant à saisir une chaîne de la forme voulue :

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    //----- configuration des broches utilisées -----
    pinMode(ledR, OUTPUT); // met la broche en sortie
    pinMode(ledV, OUTPUT); // met la broche en sortie
    pinMode(ledB, OUTPUT); // met la broche en sortie

    //----- Initialisation Série -----
    Serial.begin(115200); // initialise la vitesse de la connexion série
    //-- utilise la meme vitesse dans le Terminal Série
    // IMPORTANT : sélectionner l'option "No Line Ending" dans le Terminal Serie !

    //-----
    Serial.println("--- Instructions reconnues: ---");
    Serial.println("rvb(couleur, coeff)");
    Serial.println("Utiliser couleur parmi : rouge, vert, bleu");
    Serial.println("Utiliser coeff 0-100");
    Serial.println("-----");

} // fin de la fonction setup()
```

Fonction **loop()**

Réception de la chaîne sur le port série

- La librairie **Utils** intègre la fonction **waitingString()** qui attend sur le port série une chaîne jusqu'à l'arrivée d'un saut de ligne (« \n ») : la chaîne reçue est stockée dans un objet String

Extraction du paramètre texte

- Ensuite, à l'aide d'une condition, on teste si la chaîne n'est pas vide :
 - on appelle alors la fonction **testInstructionString()** en lui passant en paramètre :
 - la chaîne reçue,
 - la racine de la chaîne à reconnaître
 - la fonction renvoie **la chaîne extraite** si l'analyse est correcte, autrement dit qu'une chaîne racine(texte) a bien été reçue
- Ensuite, on recherche la présence d'une « , » et si c'est le cas, on coupe la chaîne en 2 en récupérant les 2 sous-chaînes : à partir de là, on récupère la couleur à utiliser et le coefficient à utiliser.
- Il ne reste plus alors qu'à appeler la fonction **ledRGB** avec les valeurs voulues pour chaque chaîne possible, ce qui permet de mettre la LED RGB dans l'état voulu en lui passant les paramètres r,g,b adéquats :

```
//--- la fonction loop() : exécutée en boucle sans fin
void loop() {

    //--- reception de la chaîne sur le port série - fin par saut de ligne
    //chaîneReception=utils.waitingString(true);// avec debug
    chaîneReception=utils.waitingString();// sans debug

    if (chaîneReception!="") { // si une chaîne a été reçue

        //--- gestion rvb(couleur)

        param=utils.testInstructionString(chaîneReception, "rvb(", true); // extrait le paramètre de la chaîne avec debug
        //param=utils.testInstructionString("LED(", chaîneReception); // extrait le paramètre de la chaîne sans debug

        if (param!="") { // si une chaîne a été reçue en paramètre

            Serial.println("Arduino a reçu le parametre : " + param);

            if (param.indexOf(",")>0) { // si 1 virgule a été trouvée

                String strCouleur=param.substring(0,param.indexOf(",")); // extrait du debut à la virgule = la couleur
                Serial.println("Couleur : "+ strCouleur);

                String strCoeff=param.substring(param.indexOf(",")+1); // extrait de la virgule à la fin = le coefficient
                Serial.println("Pourcentage: " + strCoeff);

                int coeff=utils.stringToLong(strCoeff);
                Serial.println(coeff); // debug

                int coeffPWM=map(coeff,0,100, 0,255); // re-échelonne valeur 0-100% en 0-255PWM
            }
        }
    }
}
```



```

    if(strCouleur=="rouge") { // si le parametre vaut rouge
        ledRVB(coeffPWM,0,0); // couleur RGB voulue
    } // fin if
    else if(strCouleur=="vert") { // si le parametre vaut vert
        ledRVB(0,coeffPWM,0); // couleur RGB voulue
    } // fin else if
    else if(strCouleur=="bleu") { // si le parametre vaut bleu
        ledRVB(0,0,coeffPWM); // couleur RGB voulue
    } // fin else if

    } // fin si une virgule a été trouvée

    } // fin if param!=""

} // fin if chaineReception

} // fin de la fonction loop()

```

Fonction ledRVB()

Pour simplifier le code, on crée une fonction pour gérer l'affichage des couleurs avec la LED RGB :

- la fonction ne renvoie rien = type void
- reçoit 3 valeurs entières (0-255) fixant la proportion des couleurs RVB
- à l'aide de 3 valeurs, on crée les impulsions PWM correspondantes
- la pause est intégrée dans cette fonction

Truc : si vous utilisez une LED à anode commune, il suffit de mettre **255-valeur** à la place de **valeur**

```

//---- fonction pour combiner couleurs ----

void ledRVB(int Rouge, int Vert, int Bleu) {

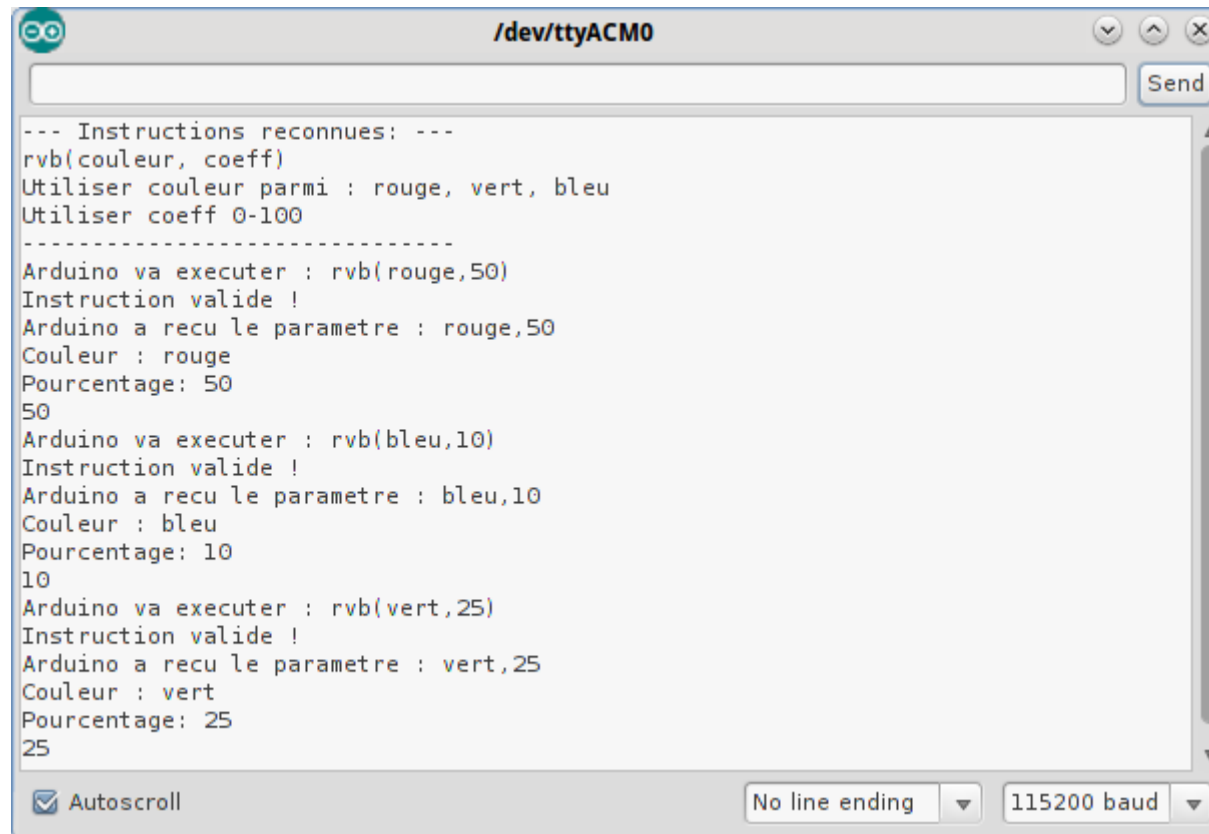
    analogWrite(ledR,Rouge); // proportion de rouge
    analogWrite(ledV,Vert); // proportion de vert
    analogWrite(ledB,Bleu); // proportion de bleu

} // fin fonction ledRVB

```

Fonctionnement du programme

- Une fois la carte Arduino programmée, ouvrir le Terminal Série en réglant sur « newline » et « 115200 ».
- Puis saisir une chaîne de la forme `rvb(rouge, vert, bleu)` avec une valeur comprise entre 0 et 255 pour chaque couleur
- ce qui donne :



```
--- Instructions reconnues: ---
rvb(couleur, coeff)
Utiliser couleur parmi : rouge, vert, bleu
Utiliser coeff 0-100
-----
Arduino va executer : rvb(rouge,50)
Instruction valide !
Arduino a reçu le parametre : rouge,50
Couleur : rouge
Pourcentage: 50
50
Arduino va executer : rvb(bleu,10)
Instruction valide !
Arduino a reçu le parametre : bleu,10
Couleur : bleu
Pourcentage: 10
10
Arduino va executer : rvb(vert,25)
Instruction valide !
Arduino a reçu le parametre : vert,25
Couleur : vert
Pourcentage: 25
25
```

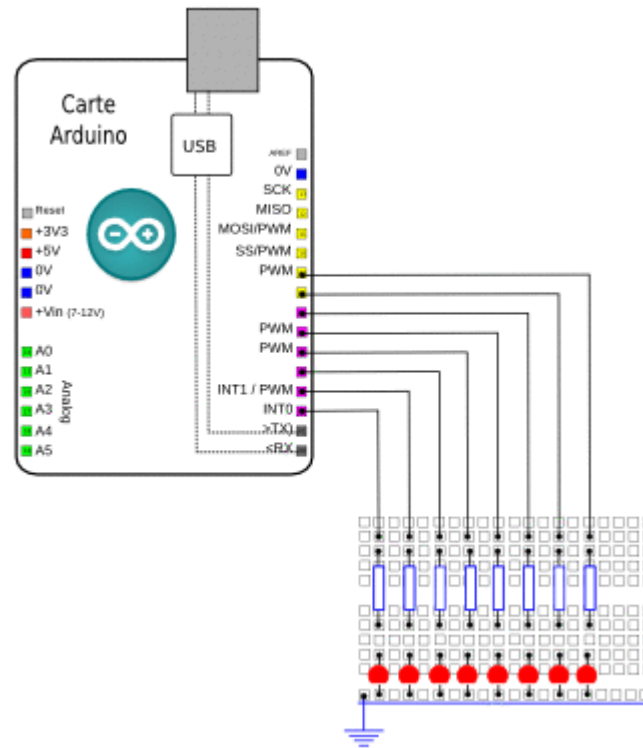
`rvb(rouge,50)` : le LED RGB s'allume en rouge à 50 %, `rvb(bleu, 10)` : le LED RGB s'allume en bleu à 10 %, etc...

A nouveau, remarquer avec quelle facilité vous pouvez créer vos propres « instructions » reçues sur le port série.

Les utilisations potentielles d'un mixage de paramètres numériques et texte sont nombreuses, notamment par exemple pour le contrôle d'un robot utilisant deux moteurs CC : il sera possible d'utiliser des instructions de la forme `xxx(droite,10)` ou `xxx(gauche,20)`, etc...

13. Arduino'live : contrôler les broches Arduino par le port série : le montage

- On va connecter le nombre voulus de LEDs, ici 8 LEDs chacune en série avec une résistance sur 8 broche E/S de la carte Arduino configurées en sortie.



14. Arduino'live : contrôler les broches Arduino par le port série : le programme

Ce qu'on va faire ici...

- Nous allons pour finir réaliser un contrôle en « live » des broches de l'Arduino à partir de chaînes reçues sur le port série, sous la forme, LED(broche, etat). Ceci pour montrer le potentiel d'utilisation et la simplicité à coder ses propres « fonctions » reconnues en réception sur le port série.
- Ce code est disponible ici : <https://gist.github.com/sensor56/9ab27e8d7291a8ac3460>

Entête déclarative

Inclusion des librairies utiles

- On commence par inclure la librairie Utils

Déclaration broche utilisée

- On déclare une constante définissant le nombre de LEDs utilisées, 8 dans notre cas, sur les broches 2 à 9 (les broches 0 et 1 restent réservées pour la communication série...)

Variables et objets utiles

- On déclare :
 - un objet String pour la stocker la chaîne reçue en réception
 - un objet String pour le paramètre à extraire
 - un objet racine permettant l'accès aux fonctions de la librairie Utils

```
//--- entete déclarative = variables et constantes globales

//---- inclusion de librairie
#include <Utils.h> // librairie personnelle avec plusieurs fonctions utiles

//--- variables pour réception chaîne sur port Série
String chaineReception=""; // déclare un objet String vide
// long params[6]; // déclare un tableau de long - taille en fonction nombre max paramètres attendus
String param=""; // déclare un String

//---- constantes des broches utilisées
const int nombreLeds=8; // nombre de broches à utiliser

//--- objets utiles
Utils utils; // objet racine donnant accès aux fonctions de la librairie Utils
```

Fonction **setup()**

Initialisation série

- On initialise la communication série à 115200 bauds

Initialisation des broches

- On initialise les broches utilisées pour le contrôle des LEDs à l'aide d'une boucle.

Code initial

- On affiche un simple message invitant à saisir une chaîne de la forme voulue :

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    //----- configuration des broches utilisées -----
    for (int i=2; i<nombreLeds+2; i++) { // défile broches à partir de la 2
        pinMode(i,OUTPUT); // met les broches en sortie
    } // fin for

    //----- Initialisation Série -----
    Serial.begin(115200); // initialise la vitesse de la connexion série
    //-- utilise la meme vitesse dans le Terminal Série
    // IMPORTANT : sélectionner l'option "No Line Ending" dans le Terminal Serie !

    //-----
    Serial.println("--- Instructions reconnues: ---");
    Serial.println("LED(numero, etat)");
    Serial.println("Utiliser numero de la broche");
    Serial.println("Utiliser etat parmi ON OFF");
    Serial.println("-----");

} // fin de la fonction setup()
```

Fonction `loop()`

Réception de la chaîne sur le port série

- La librairie **Utils** intègre la fonction `waitingString()` qui attend sur le port série une chaîne jusqu'à l'arrivée d'un saut de ligne (« \n ») : la chaîne reçue est stockée dans un objet `String`

Extraction du paramètre texte

- Ensuite, à l'aide d'une condition, on teste si la chaîne n'est pas vide :
 - on appelle alors la fonction `testInstructionString()` en lui passant en paramètre :
 - la chaîne reçue,
 - la racine de la chaîne à reconnaître
 - la fonction renvoie **la chaîne extraite** si l'analyse est correcte, autrement dit qu'une chaîne `racine(texte)` a bien été reçue
- Ensuite, on recherche la présence d'une « , » et si c'est le cas, on coupe la chaîne en 2 en récupérant les 2 sous-chaînes : à partir de là, on récupère la broche à utiliser et l'état à appliquer.
- Il ne reste plus alors qu'à appeler la fonction `digitalWrite()` avec les valeurs voulues :

```
//--- la fonction loop() : exécutée en boucle sans fin
void loop() {

  //--- reception de la chaine sur le port série - fin par saut de ligne
  //chaineReception=utils.waitingString(true);// avec debug
  chaineReception=utils.waitingString();// sans debug

  if (chaineReception!="") { // si une chaine a été reçue

    param=utils.testInstructionString(chaineReception, "LED(", true); // extrait le paramètre de la chaine avec debug
    //param=utils.testInstructionString("LED(", chaineReception); // extrait le paramètre de la chaine sans debug

    if (param!="") { // si une chaine a été reçue en paramètre

      Serial.println("Arduino a reçu le parametre : " + param);

      if (param.indexOf(",")>0) { // si 1 virgule a été trouvée

        String strBroche=param.substring(0,param.indexOf(",")); // extrait du debut à la virgule = la broche
        Serial.println("Broche: " + strBroche);

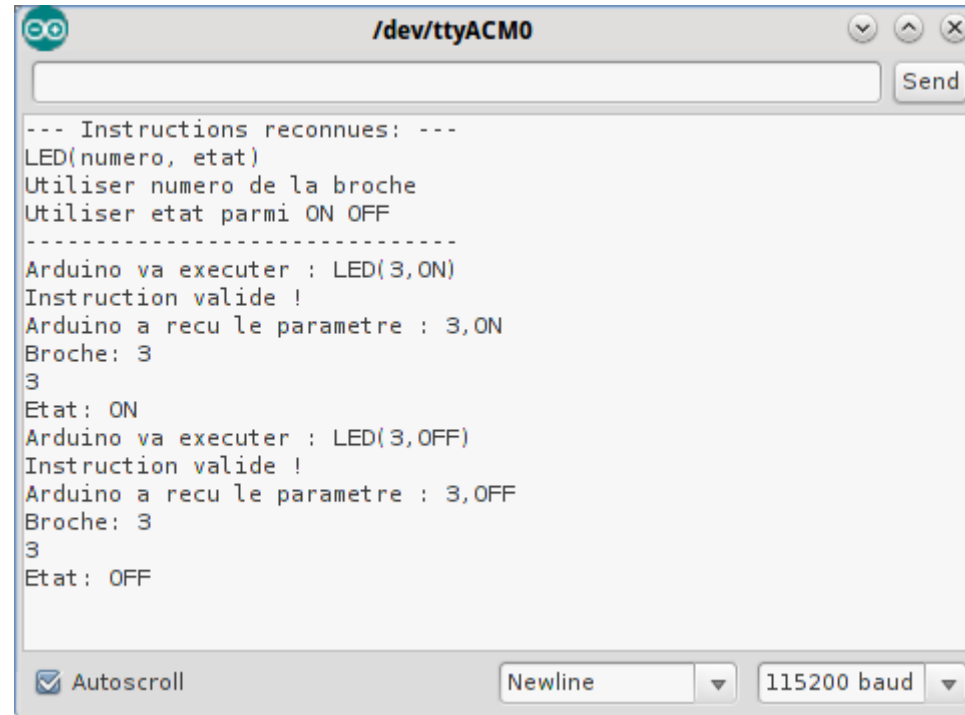
        int broche=utils.stringToLong(strBroche);
        Serial.println(broche); // debug

        String strEtat=param.substring(param.indexOf(",")+1); // extrait de la virgule à la fin = l'état
        Serial.println("Etat: " + strEtat);

        if (strEtat=="ON") digitalWrite(broche,HIGH);
        if (strEtat=="OFF") digitalWrite(broche,LOW);
      } // fin si une virgule a été trouvée
    } // fin if param!=" "
  } // fin if chaineReception
} // fin de la fonction loop()
```

Fonctionnement du programme

- Une fois la carte Arduino programmée, ouvrir le Terminal Série en réglant sur « newline » et « 115200 ».
- Puis saisir une chaîne de la forme LED(broche, etat) avec une valeur comprise entre 2 et 9 pour la broche et parmi ON/OFF pour allumer/éteindre la LED
- ce qui donne :

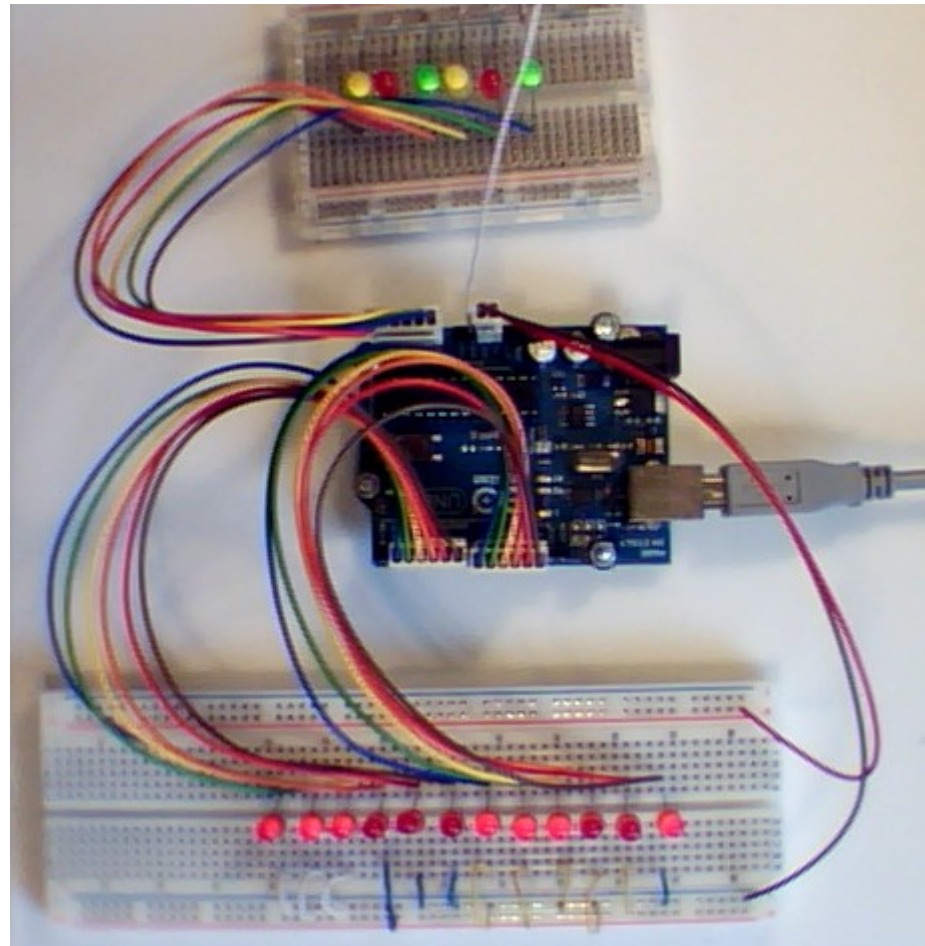


```
--- Instructions reconnues: ---
LED(numero, etat)
Utiliser numero de la broche
Utiliser etat parmi ON OFF
-----
Arduino va executer : LED(3,ON)
Instruction valide !
Arduino a reçu le parametre : 3,ON
Broche: 3
3
Etat: ON
Arduino va executer : LED(3,OFF)
Instruction valide !
Arduino a reçu le parametre : 3,OFF
Broche: 3
3
Etat: OFF
```

LED(2,ON) : la LED sur la broche 2 s'allume, LED(3,OFF) : la LED sur la broche 3 s'éteint...

Note :

Le grand intérêt de l'utilisation de chaînes texte en réception sur le port série avec Arduino réside dans la possibilité de faire communiquer assez facilement Arduino avec des interfaces graphiques côté PC.



Arduino'live : pour contrôler jusqu'à 18 LEDs ou dispositifs ON/OFF par le port série.

15. Les éléments du langage Arduino étudiés dans cet atelier

- Ma librairie Utils est disponible ici : http://www.mon-club-elec.fr/pmwiki_reference_lib_arduino_perso/pmwiki.php?n=Main.HomePage

La documentation complète du langage Arduino en français est disponible ici :
http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.ReferenceMaxi

16. A présent, vous devriez être capable :

- d'utiliser ma librairie Arduino Utils qui simplifie la réception de chaîne avec paramètres sur le port série,
- de contrôler Arduino à l'aide de chaînes reçues sur le port série intégrant des paramètres numériques ou texte.

Table des matières

Recevoir des fonctions avec paramètres sur le port série, des fonctions avec chaînes de caractères, etc.. à l'aide de ma librairie Arduino Utils.

Intro |

Matériel nécessaire pour les ateliers Arduino |

Matériel mécanique utilisé ici ? |

Recevoir des chaînes avec paramètres : Installation et présentation de ma librairie Utils |

Recevoir sur le port série une chaîne avec un paramètre numérique : le montage |

Recevoir sur le port série une chaîne avec un paramètre numérique : le programme |

Contrôler 2 servomoteurs par une chaîne avec valeur numérique reçue sur le port série : le montage |

Contrôler 2 servomoteurs par une chaîne avec valeur numérique reçue sur le port série : le programme |

Réception d'une chaîne avec plusieurs paramètres numériques par le port série : le montage |

Réception d'une chaîne avec plusieurs paramètres numériques par le port série : programme |

Réception d'une chaîne avec un paramètre texte par le port série : le programme |

Réception d'une chaîne avec paramètres texte et numérique par le port série |

Arduino'live : contrôler les broches Arduino par le port série : le montage |

Arduino'live : contrôler les broches Arduino par le port série : le programme |

Les éléments du langage Arduino étudiés dans cet atelier |

A présent, vous devriez être capable : |

Bravo !
vous avez terminé cet atelier Arduino !



Prêt pour la suite ? Retrouvez de nombreux autres thèmes d'ateliers Arduino ici :

http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS