Robotique

Support de Cours et Travaux pratiques

Sophie CAVASSILA

Anne Laure DEMAN

5 juin 2020

Table des matières

I. Introduction : les microcontrôleurs

Les systèmes embarqués correspondent à des équipements souvent autonomes avec une «intelligence» qui leur permet d'être en interaction directe avec l'environnement dans lequel ils sont placés.

Les systèmes embarqués sont présents dans des applications de plus en plus nombreuses:

- Systèmes mobiles communicants: robotique, téléphones mobiles, récepteurs GPS,
- Électronique grand public, électroménager, domotique
- automobile, transport aérien/maritime/fluvial.
- Santé

Ces applications embarquées nécessitent des systèmes de contrôle/commande intelligents qui intègrent le maximum de fonctions dans un même circuit (diminuer le nombre de composants des systèmes et leur coût global).

Les cartes Arduino à base de microcontrôleurs sont **largement répandues en robotique** et présentent les fonctionnalités suivantes :

Un microcontrôleur intègre les éléments suivants:

- Une unité centrale (CPU ″Central Processing Unit″) qui est le circuit effectuant les calculs et prenant les décisions logiques. L'unité centrale est cadencée par une horloge système.
- Des mémoires qui stockent les codes opératoires des programmes et éventuellement des données qui sont mises à jour pendant l'exécution du programme
- Un ensemble de périphériques tels que des ports de communications numériques assurant la communication du microcontrôleur avec le monde extérieur, convertisseur analogique numérique, des circuits de temporisation, des unités de modulation de largeurs d'impulsion (PWM), des communication série (SPI, I2C, UART,…).

A. Les cartes Arduino

La carte Arduino Uno est basée sur un ATMega328 cadencé à 16 MHz.
Elle dispose :

- de 14 broches numériques d'entrées/sorties (dont 6 peuvent être utilisées en sorties PWM (largeur d'impulsion modulée)),
- de 6 entrées analogiques (qui peuvent également être utilisées en broches entrées/sorties numériques),
- d'une horloge interne 16Mhz,
- et d'un bouton de réinitialisation (reset).

La carte Arduino Mega 2560 est basée sur un ATMega2560 cadencé à 16 MHz.
Elle dispose :

- de 54 broches numériques d'entrées/sorties (dont 14 peuvent être utilisées en sorties PWM (largeur d'impulsion modulée)),
- de 16 entrées analogiques (qui peuvent également être utilisées en broches entrées/sorties numériques),
- de 4 UART (port série matériel),
- d'une horloge interne 16Mhz,
- d'une connexion USB,

B. Bibliographie

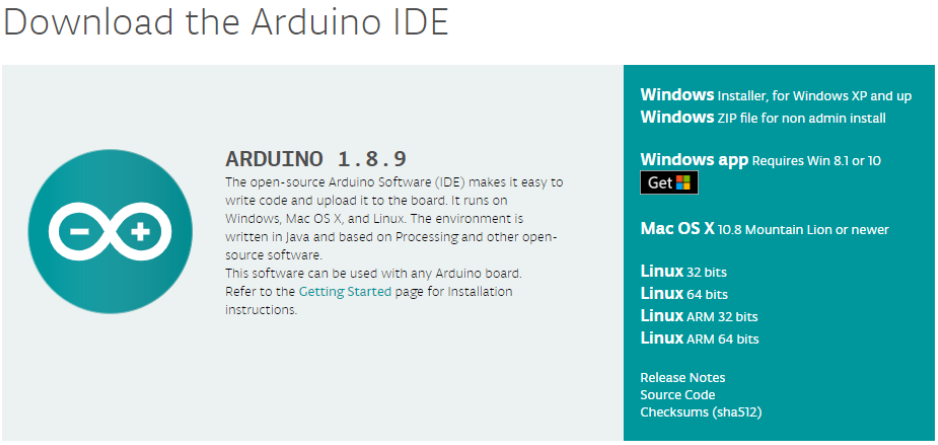
https://store.arduino.cc/arduino-uno-rev3
http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.MaterielUno
https://store.arduino.cc/mega-2560-r3

http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.MaterielMega2560)

II. L'environnement de développement Arduino IDE et éléments de programmation

L'IDE Arduino (Arduino Integrated Development Environment (IDE)) est l'environnement de développement dédié aux cartes Arduino.
Le fichier d'installation de l'environnement de développement peut être téléchargé en ligne:
https://www.arduino.cc/en/Main/Software#download

L'environnement de développement Arduino IDE est disponible pour les systèmes d'exploitation: Windows, Mac OS X et Linux.



Avec un support technique ici :
https://www.arduino.cc/en/Guide/HomePage
https://wiki.mdl29.net/lib/exe/fetch.php?media=robotsarduino:installation_arduino_ardublock_windows.pdf

A. a) Eléments de programmation

Le langage de programmation Arduino est proche du langage C. L'ensemble des éléments de programmation sont disponibles sur le site https://www.arduino.cc/reference/en/#structure

La structure minimale d'un programme est constituée de deux fonctions setup() et loop()

```
void setup() // fonction d'initialisation des périphériques utilisés
{
}

void loop()  // fonction principale décrivant l'algorithme du programme
{
}
```

La fonction **setup()** permet l'initialisation des périphériques utilisés. Elle est exécutée une seule fois lors de l'initialisation de la carte. Par exemple, les directions des ports de communication numériques seront configurés ici.

La fonction **loop**() décrit l'algorithme du programme. Elle se répète à l'infini.

Ces fonctions sont précédées d'un entête où sont déclarées les constantes et les fonctions des bibliothèques :

#define permet de définir des constantes

Exemple : #define ledPin 3 associe la valeur 3 à la constante ledPin.

#include permet de déclarer les bibliothèques de fonctions utilisées dans le programme

Exemple : #include <Servo.h> permet d'utiliser les fonctions pour la commande de servomoteurs.
De nombreuses bibliothèques sont présentées ici :https://www.arduino.cc/en/Reference/Libraries

B. Les variables et constantes

Une variable est un espace réservé en mémoire pour stocker une donnée. Une variable peut être locale (existe uniquement dans la fonction dans laquelle elle a été déclarée) ou globale (existe pour les fonctions du programme).

Type	Grandeur	Plage des données	Taille en octets
(signed) int	entière	-32768 à +32767	2
unsigned int	entière	0 à 65535	2
(signed) long	entière	-2 147 483 648 à +2 147 483 647	4
unsigned long	entière	0 à 4 294 967 295	4
(signed) char	entière	-128 à +127	1
unsigned char ou byte	entière	0 à 255	1
float	décimale	-3.4028235E+38 à +3.4028235E+38	4
double pour Arduino Uno et Mega	décimale	-3.4028235E+38 à +3.4028235E+38	4
bool	binaire	false ou true	1

Les variables peuvent être exprimées sans conséquence dans les bases binaires, décimale ou hexadécimale.

Exemple :
char valeur ;
valeur=13 ; ou valeur =0x0D ; ou valeur=0b00001101

Les constantes:
false est équivalent à '0'
true est souvent défini comme '1' mais représente en fait tout ce qui n'est pas '0'.

C. Les opérateurs et les fonctions

1. Les opérateurs arithmétiques

Opérateur	Exemples
* (multiplication)	2*3 égal 6
+ (addition)	2+3 égal 5
- (soustraction)	2-3 égal -1
/ (division entière ou réelle)	5/2 égal 2 (division entière)
% (reste de la division)	5%2 égal 1 (reste de la division entière)
= (affectation)	permet l'initialisation d'une variable

2. Les opérateurs relationnels

Opérateur
!= (not equal to)
< (less than)
<= (less than or equal to)
== (equal to)
> (greater than)
>= (greater than or equal to)

3. Les opérateurs booléens

! (logical not) && (logical and) || (logical or)

4. Les pointeurs

& (reference operator) * (dereference operator)

5. Les opérateurs de bits

& (bitwise and) << (bitshift left) >> (bitshift right)

^ (bitwise xor) | (bitwise or) ~ (bitwise not)

6. Les structures de contrôle

Les boucles de répétition avec test : do...while while
Les boucles de répétition avec compteur: for
Les choix et alternatives: if else switch...case

7. Les fonctions Arduino

Le lexique des fonctions dédiées aux microcontrôleurs Arduino est disponible ici :
https://www.arduino.cc/reference/en/#functions

- Digital I/O

digitalRead() digitalWrite() pinMode()

- Analog I/O

analogRead() analogReference() analogWrite()

- Zero, Due & MKR Family

analogReadResolution() analogWriteResolution()

- Advanced I/O

noTone() pulseIn() pulseInLong() shiftIn() shiftOut() tone()

- Time

delay() delayMicroseconds() micros() millis()

- Math

abs() constrain() map() max() min() pow() sq() sqrt()

- Trigonometry

cos() sin() tan()

- Characters

isAlpha() isAlphaNumeric() isAscii() isControl() isDigit() isGraph() isHexadecimalDigit() isLowerCase() isPrintable() isPunct() isSpace() isUpperCase() isWhitespace()

- Random Numbers

random() randomSeed()

- Bits and Bytes

bit() bitClear() bitRead() bitSet() bitWrite() highByte() lowByte()

- External Interrupts

attachInterrupt() detachInterrupt()

- Interrupts

interrupts() noInterrupts()

- Communication

Serial Stream

- USB

Keyboard Mouse

8. Les bibliothèques dédiées aux communications avec des périphériques

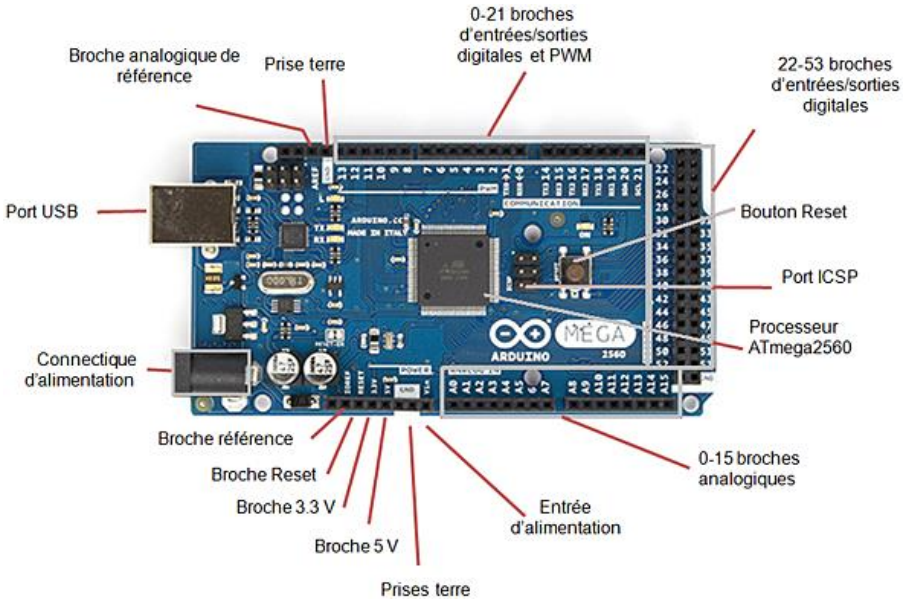
https://www.arduino.cc/en/Reference/Libraries

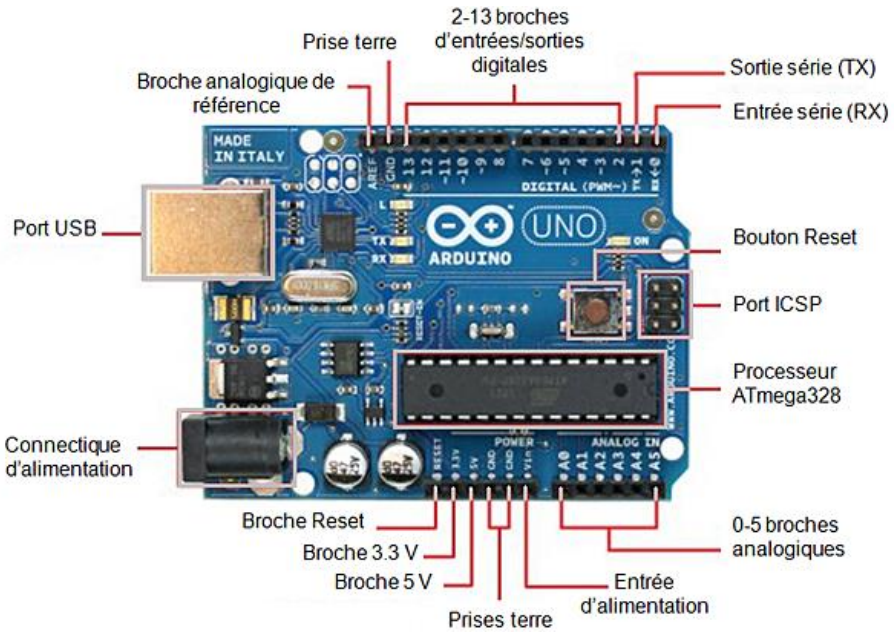
- EEPROM - reading and writing to "permanent" storage
- Ethernet - for connecting to the internet using the Arduino Ethernet Shield, Arduino Ethernet Shield 2 and Arduino Leonardo ETH
- Firmata - for communicating with applications on the computer using a standard serial protocol.
- GSM - for connecting to a GSM/GRPS network with the GSM shield.
- LiquidCrystal - for controlling liquid crystal displays (LCDs)
- SD - for reading and writing SD cards
- Servo - for controlling servo motors
- SPI - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
- SoftwareSerial - for serial communication on any digital pins. Version 1.0 and later of Arduino incorporate Mikal Hart's NewSoftSerial library as SoftwareSerial.
- Stepper - for controlling stepper motors
- TFT - for drawing text , images, and shapes on the Arduino TFT screen
- WiFi - for connecting to the internet using the Arduino WiFi shield
- Wire - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.

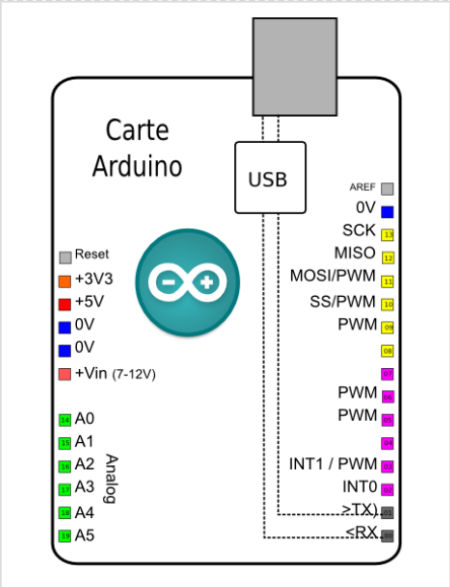
III. Les entrées et sorties numériques : Les capteurs / Les actionneurs

La carte de développement Arduino Mega dispose de: 54 broches de communications d'entrées/ sorties numériques structurées en ports (14 broches d'entrées/sortes numériques pour Uno). Ces broches peuvent être adressées en entrée ou en sortie (broches bidirectionnelles), individuellement ou par groupe de 8 (port 8 bits).

Le niveau électrique des broches peut être **LOW (0 Volt) , HIGH (5Volts).** Chaque broche fournit ou reçoit un maximum de 40 mA. Des résistances de protection (pull-up) doivent être évaluées pour chaque broche que le courant consommé ne dépasse pas cette valeur.



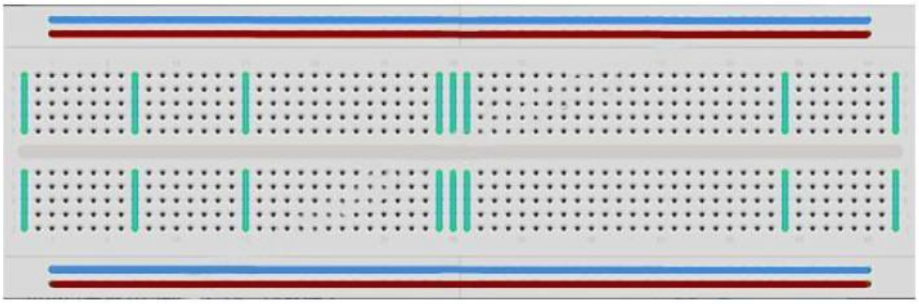




A. Platine de prototypage

Une planche prototype vous permet de réaliser des circuits très rapidement, sans avoir besoin de réaliser de soudures.

Exemple :



Il existe une grande variété de planches prototypes. La plus simple est une grille de trous dans un bloc de plastique. A l'intérieur se trouvent des lames métal permettant la connexion électrique entre les différents trous d'une même ligne. Mettre les branches de deux composants sur une même ligne les "assemble" électriquement. La ligne creusée au centre de la plaque symbolise une rupture électrique entre la partie haute et la partie basse. Cela signifie aussi que vous pouvez positionner une puce à cheval entre haut et bas. Certaines planches prototypes ont aussi deux lignes horizontales en haut et en bas. On les utilise généralement pour créer une ligne d'alimentation +5V et masse.

Attention tout de même, les planches prototypes ont comme limite d'utilisation la qualité des connexions qui ne valent pas une soudure et peuvent entraîner parfois des dysfonctionnements.

B. Configuration des broches d'entrée/sortie numériques

Quasiment toutes les broches d'un microcontrôleur Arduino peuvent être programmées en entrée ou sortie numérique (et non les deux en même temps).

Configuration en sortie: l'information est envoyée par le microcontrôleur vers le monde extérieur (actionneurs, moteurs).

Configuration en entrée: les broches sont initialisées par le monde extérieur (capteurs) et sont lues par le microcontrôleur.

Broches configurées en sortie

Broches configurées en entrée

Les broches configurées en entrée permettent de faire l'acquisition de données de capteurs alors que celles configurées en sortie permettent de commander des actionneurs.

Quel que soit la broche du microcontrôleur, on ne peut y brancher une tension supérieure à la tension d'alimentation, c'est à dire 5V ou 3,3V selon le modèle, ni une tension inférieure à la masse, le 0V, sous peine de destruction d'au moins la broche concernée si ce n'est l'Arduino en entier.

La configuration en entrée (INPUT) ou sortie (OUTPUT) est obtenue par la fonction: pinMode()

Syntax : pinMode(pin, mode)
Parameters
pin: the Arduino pin number to set the mode of.
mode: INPUT, OUTPUT, or INPUT_PULLUP. See the Digital Pins page for a more complete description of the functionality.
Returns
Nothing

C. Configuration en sortie : les actionneurs

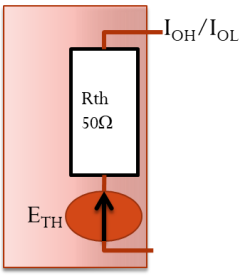
Les broches configurées en sorties numériques permettent de fixer le niveau électrique de la broche et ainsi commander l'état d'actionneurs tels que : alimentation d'une LED ou un témoin lumineux, commande d'un transistor, connexion à un afficheur LCD, commande d'interrupteurs, mais aussi permettent l'envoi de données de communications numériques (UART, SPI, I2C, USB).

Exemple :
pinMode(13, OUTPUT) permet de configurer la broche n°13 du microcontrôleur en sortie.
Les broches numérotées de 0 à 13 sont les entrées sorties numériques.

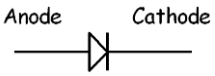
Il est préférable de ne pas dépasser une **consommation de 20 mA** sur une broche configurée en sortie et il est absolument nécessaire de ne pas dépasser **40 mA** sous peine de risque la destruction de la sortie du microcontrôleur. Il s'agit aussi de ne pas dépasser au total 200 mA.

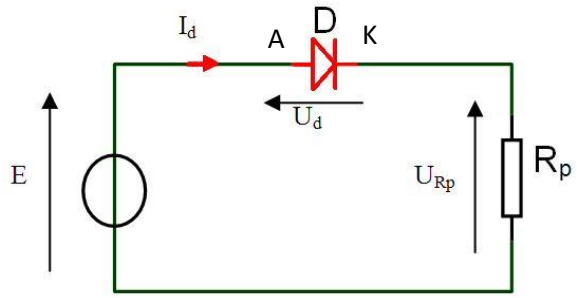
Le niveau électrique d'une **broche configurée en sortie** est fixée par le microcontroleur avec la fonction : digitalWrite()
Syntax digitalWrite(pin, value)
Parameters
pin: the Arduino pin number.
value: HIGH or LOW.
Returns Nothing

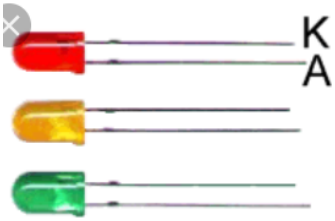
Le modèle électrique d'une broche configurée en sortie est très simple. **Le courant maximal Imax des broches est 40 mA pour un ATMega2560.**
Sortie haute: Eth=5V, Rth=50 Ω , I_{OH}<40mA
Sortie basse: Eth=0V, Rth=50 Ω, I_{OL}<40mA.



La diode :



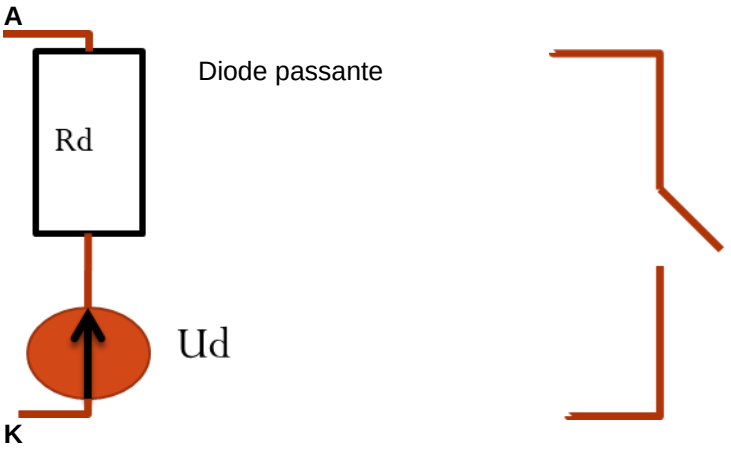




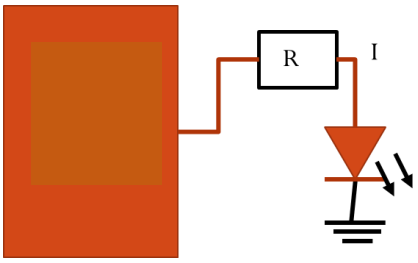
La diode est un dipôle polarisé. Elle permet la circulation d'un courant électrique dans un seul sens. Il faudra qu'un seuil de Ud aux bornes de la diode soit atteint pour que la diode soit passante. Alors, le courant traversera la diode de l'anode vers la cathode. Lorsque la diode est passante, la tension à ses bornes est égale à la tension de seuil Si le seuil de Ud n'est pas atteint, la diode sera bloquée et le courant sera nul. **(site :** http://sen.champo.free.fr/coursenligne/La%20diode.php **)**
Si le courant est trop important dans le diode, celle-ci peut se détruire.
La résistance Rp permet de fixer l'intensité du courant dans la diode. On l'appelle "résistance de protection".

Le modèle électrique d'une diode(led) est le suivant :

Diode passante: Ud=tension de seuil, Rd: résistance dynamique (valeurs typiques: Ud=0,7V, rd=0)



Exemple 1 : Le courant est fourni par le microcontrôleur. Avec un niveau HIGH sur la broche, la LED est passante.



La tension aux bornes de la résistance est : U broche= Ud+U_R=Ud+RI.
Si Ubroche= HIGH = Vcc alors RI=Ubroche-Ud et R=(Ubroche-Ud)/I.
Il faut choisir une résistance de protection de sorte que la valeur de l'intensité du courant I reste inférieure à 20mA.
Ainsi R> (Ubroche-Ud)/20.10^{-3} R>(5-0,7)/ 20.10^{-3} ainsi R> 215 Ω

TABLE INTERNATIONALE DES VALEURS NORMALISEES DES RESISTANCES A ± 10 %						
ohm		Kiloohm			mégohm	
10	100	1	10	100	1	10
12	120	1,2	12	120	1,2	12
15	150	1,5	15	150	1,5	15
18	180	1,8	18	180	1,8	18
22	220	2,2	22	220	2,2	22
27	270	2,7	27	270	2,7	
33	330	3,3	33	330	3,3	
39	390	3,9	39	390	3,9	
47	470	4,7	47	470	4,7	
56	560	5,6	56	560	5,6	
68	680	6,8	68	680	6,8	
82	820	8,2	82	820	8,2	

Fig. 13. - Valeurs des résistances avec tolérances ±10 %

Compte tenu du tableau des valeurs normalisées de résistance, **on choisit au minimum R=220 Ω**

Questions :

Une led rouge (Kingbright λ = 627nm ; Vd = 1,95 V) est montée comme sur le schéma précédent sur la broche 6 du PORTB. Comment choisir R pour avoir un courant ne dépassant pas 5 mA?

On désire commander deux leds rouge en série comme sur le schéma 1. Comment dimensionner correctement la résistance de protection?

1. Exemple BLINK

Ce programme permet de faire clignoter la led connectée à la broche 13 (visible sur la carte de développement) avec une fréquence de 0,5Hz.

Ce premier projet permet de faire clignoter la led de la carte de développement connectée à la broche 13. La temporisation du clignotement est ajustable.

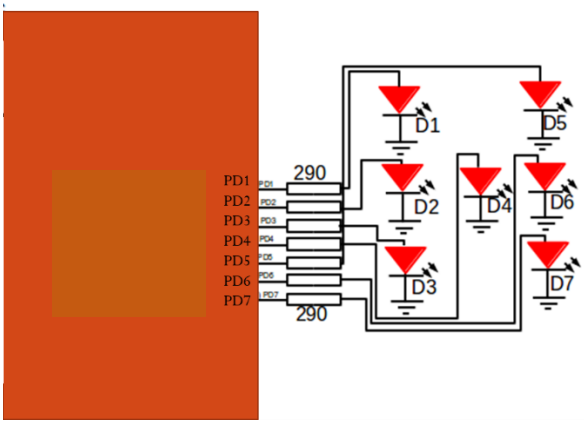

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
digitalWrite(LED_BUILTIN, HIGH);// turn the LED on (HIGH is the
                                // voltage level)
delay(1000);       // wait for a second
digitalWrite(LED_BUILTIN, LOW);// turn the LED off by making the
voltage LOW
  delay(1000);                          // wait for a second
```

Note : Dans ce programme, LED_BUILTIN est une constante qui est connue dans l'environnement de développement et qui est égale à 13, numéro de la broche à laquelle la led est connectée.

2. Projet Dé lumineux (au choix)

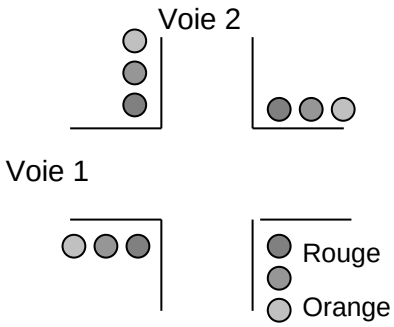
On désire réaliser un dé lumineux comme indiqué sur le schéma ci-dessous. Les 7 LEDS sont connectées aux broches du microcontrôleur. On considère que la tension de seuil de ces diodes est de 0,6 volts. Déterminer les valeurs des résistances de protection.

Rédiger un programme qui permet d'éclairer les différents états d'un dé lumineux avec une durée de 1s entre les différents états.



3. Projet Feu de croisement (au choix)

On souhaite ici simuler le fonctionnement d'un feu de croisement. Un feu est constitué de 3 leds rouge, orange et vert connectées au microcontrôleur.

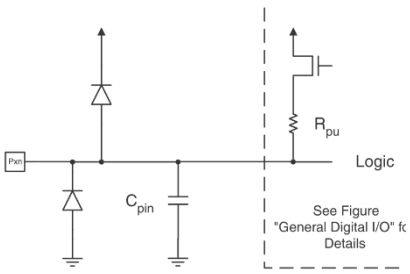


Les feux sont gérés de la façon suivant :

- La voie 1 assurant un trafic important reste au vert pendant environ **10s** alors que la durée du vert sur la voie 2 est de **5s.**
- La durée de l'état intermédiaire orange est de **1s.**
- Par mesure de sécurité, avant tout passage au vert de l'une ou l'autre des voies, on établira l'état rouge partout pendant **2s.**

Le cycle Rouge/Orange/Vert peut donc être décrit par 4 états qui s'enchaînent indéfiniment.

D. Configuration en entrée : les capteurs numériques.



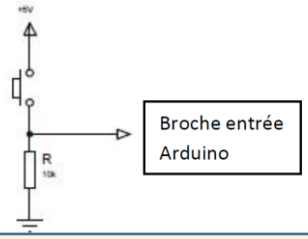
Les broches configurées en entrées numériques permettent de mesurer les états de capteurs numériques tels que capteurs de présence, de contact, boutons poussoirs, interrupteurs mais aussi permettent la réception de données de communications numériques (UART, SPI, I2C, USB).

La configuration en entrée d'un broche est obtenue par la fonction pinMode() avec l'argument INPUT ou INPUT_PULLUP.

- Avec la configuration : **INPUT_PULLUP**. Chaque broche dispose d'une résistance interne de rappel (pull-up) de 20-50kOhms. L'état de la broche configurée en entrée peut être soit **HIGH** (5 ou 3 Volt selon configuration) soit **LOW** (0Volt)
- Avec la configuration : **INPUT**. Les résistances internes de rappel (pull-up) de 20-50kOhms sont déconnectées par défaut. L'état de la broche configurée en entrée peut être : **HIGH** (5 ou 3 Volt selon configuration), **LOW** (0Volt) ou de **haute impédance** (état indéterminé)

Il est préférable de ne pas dépasser une **consommation de 20 mA** sur une broche configurée en entrée et absolument nécessaire de ne pas dépasser **40 mA** sous peine de risque la destruction du port du microcontrôleur. Il s'agit aussi de ne pas dépasser au total 200 mA.

Exemple : Branchement d'un bouton poussoir (entrée) avec une résistance de 10kΩ



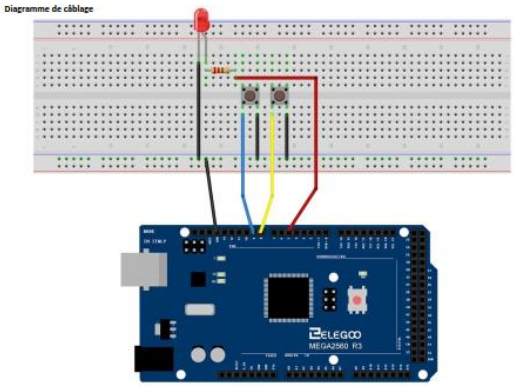
Le niveau électrique d'une **broche configurée en entrée** est lue par le microcontroleur par la fonction digitalRead()

Syntax digitalRead(pin)
Parameters : pin: the Arduino pin number you want to read
Returns : HIGH or LOW

Un niveau haut (HIGH) est mesuré si le niveau électrique est supérieur à 3.0V (5V boards) ou 2.0V volts (3.3V boards).
Un niveau bas (LOW) est mesuré si le niveau électrique est inférieur à 1.5V (5V boards) ou 1.0V volts (3.3V boards).

1. Projet avec boutons poussoirs

Le projet permet de lire l'état de 2 boutons-poussoirs connectés aux broches 8 et 9 pour allumer ou éteindre une led connectée à la broche 5.



```
int ledPin = 5;
int buttonApin = 9;
int buttonBpin = 8;

byte leds = 0;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonApin, INPUT_PULLUP);
  pinMode(buttonBpin, INPUT_PULLUP);
}

void loop()
{
  if (digitalRead(buttonApin) == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(buttonBpin) == LOW)
  {
    digitalWrite(ledPin, LOW);
  }
}
```

2. Projet avec suiveur de ligne

Le capteur suiveur de ligne est constitué de 3 couples de photodiodes associés à des photorésistances. La lumière est émise par la photodiode puis réfléchie par la surface éclairée. La lumière réfléchie est captée par la photorésistance.
La réflectance varie en fonction de la couleur de la surface éclairée. Ainsi la résistance de la photorésistance varie en fonction de l'intensité de la lumière réfléchie. Avec le capteur suiveur de ligne, il devient donc possible d'en déduire la couleur de la surface vue par celui-ci.
Le capteur présente 3 signaux (L, M, R) à l'image des tensions aux bornes des 3 photorésistances.
Proposer un programme permettant d'éclairer 2 leds verte et rouge selon les valeurs délivrées par le capteur.

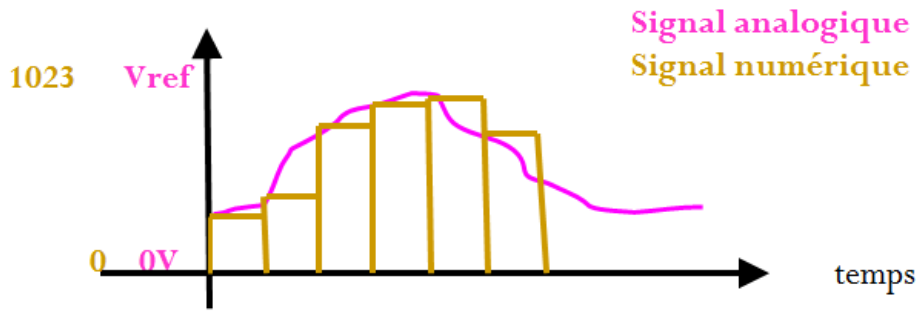
- La led verte est éclairée lorsque le signal M est LOW avec L et R HIGH.
- La led rouge est éclairée dès que L ou R sont LOW.

IV. Les Capteurs analogiques

Les microcontrôleurs intègrent un convertisseur Analogique Numérique (CAN ou ADC Analog to Digital Converter).

Un tel dispositif est chargé de convertir une tension analogique en une valeur numérique (séquence binaire codée sur 4, 8, 10,12 bits ...).

Ces valeurs numériques sont proportionnelles aux valeurs analogiques.



La plage des tensions analogiques est définie par la tension de référence. La valeur de la tension de référence doit être configurée et est choisie égale à 5V ou 1,1V pour une référence interne.

Les valeurs numériques délivrées par le convertisseur analogique numérique sont dans l'intervalle 0-15 pour un CAN 4 bits, 0-255 pour un CAN 8 bits, 0-1023 pour un CAN 10 bits.

Le quantum, q, représente la résolution du convertisseur. La relation tension analogique/valeur numérique pour un convertisseur 10 bits est la suivante:

$$ADC = \frac{Vin}{q} = \frac{Vin}{Vref} 2^{10} = \frac{Vin}{Vref} 1024$$

Vref est la tension de référence.

Vin est la tension d'entrée (gamme 0-(Vref-q)).

ADC est un nombre entier sur 10 bits.

A. Configuration de la tension de référence du convertisseur analogique-numérique broches d'entrée/sortie numériques

La tension de référence est configurée par la fonction: analogReference()

Syntax

`analogReference(type)`

Parameters

type: which type of reference to use (see list of options in the description).

Returns Nothing

Arduino AVR Boards (Uno, Mega, Leonardo, etc.)

DEFAULT: the default analog reference of 5 volts (on 5V Arduino boards) or 3.3 volts (on 3.3V Arduino boards)

INTERNAL: an built-in reference, equal to 1.1 volts on the ATmega168 or ATmega328P and 2.56 volts on the ATmega32U4 and ATmega8 (not available on the Arduino Mega)

INTERNAL1V1: a built-in 1.1V reference (Arduino Mega only)

INTERNAL2V56: a built-in 2.56V reference (Arduino Mega only)

EXTERNAL: the voltage applied to the AREF pin (0 to 5V only) is used as the reference.

B. Lecture de la valeur numérique délivrée par le convertisseur analogique numérique

La lecture de la valeur numérique résultat de la conversion analogique- numérique de la tension analogique présente sur le canal spécifié est réalisée par la fonction: analogRead()

Syntax

analogRead(pin)

Parameters : pin: the name of the analog input pin to read from (A0 to A5 on most boards, A0 to A6 on MKR boards, A0 to A7 on the Mini and Nano, A0 to A15 on the Mega).

Returns

The analog reading on the pin. Although it is limited to the resolution of the analog to digital converter (0-1023 for 10 bits or 0-4095 for 12 bits). Data type: int.

C. Exemples

Le programme ci-dessous permet la conversion de la tension présentée sur la broche A3 et affiche la valeur numérique correspondante sur l'écran LCD.

```
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

int analogPin = A3; // potentiometer wiper (middle terminal) connected to analog pin 3
                    // outside leads to ground and +5V
int val = 0;  // variable to store the value read

void setup() {
   // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
}

void loop() {
  val = analogRead(analogPin);  // read the input pin
lcd.setCursor(0, 0);
lcd.print("val= “);
lcd.print(val);
}
```

D. Projets

Nous allons nous appuyer sur le projet proposé par Arduino IDE pour illustrer la conversion analogique numérique : Exemples->Basics : ReadAnalogVoltage.

Ce programme réalise la conversion analogique numérique de la tension connectée sur la broche A0 du convertisseur et affiche dans la fenêtre Moniteur Série la valeur de la tension correspondante.

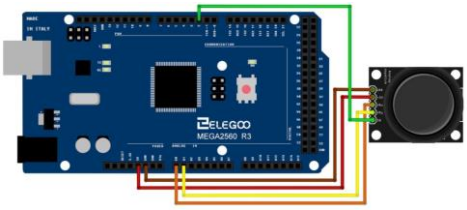
```
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.println(voltage);
}
```

Il s'agit de mettre en œuvre ce programme pour mesure les tensions délivrer au choix par plusieurs types de capteurs analogiques.

1. Projet : Joystick

Un joystick est un module à 5 broches: Vcc, ground, X, Y, Key (validation). Le câblage proposé ci-dessous prévoit :

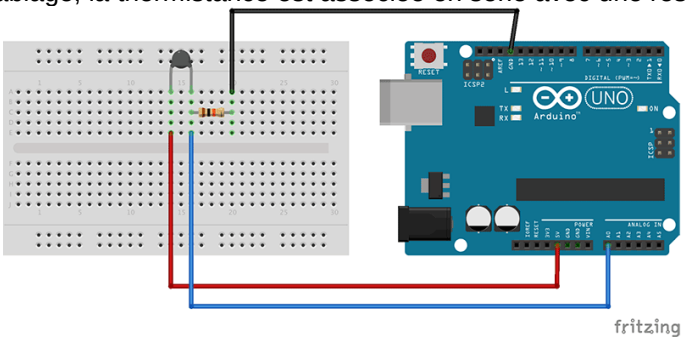
Signal de validation (key) connecté à la broche 2 (inutilisé dans le programme ci-dessus)
Axe X connecté à la broche A0
Axe Y connecté à la broche A1 (inutilisé dans le programme proposé ci-dessus).



Compléter le programme ci-dessus pour afficher également les tensions mesurées sur l'axe Y.

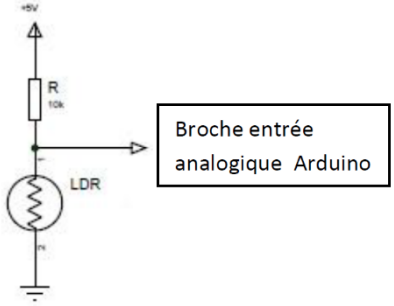
2. Projet : Thermistance (au choix)

Une thermistance est une résistance thermique: sa valeur varie en fonction de la température. Une résistance standard voit sa temperature varier, mais très faiblement, une thermistance est faite pour que cette variation soit importante et facile à mesure : 100 ohms par degré.
Il existe deux types: NTC (negative temperature coefficient) et PTC (positive temperature coefficient). En general, on utilise des NTC
Pour le câblage, la thermistance est associée en série avec une résistance de **10kOhm**.



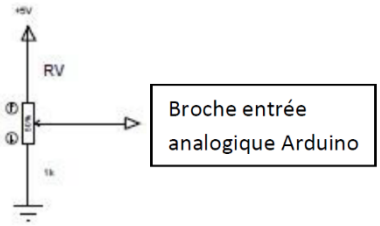
3. Projet : Photorésistance (LDR) (au choix)

Un photorésistance appelée également LDR « Light dependant resistor » est une résistance dont la valeur dépend de la quantité de umière reçue par le composant. Cette résistance prend une valeur de 50kW en obscurité et varie jusqu'à 500W en pleine lumière. Pour convertir cette variation de résistance en variation de tension, il faut bien entendu apporter une alimentation.
Pour le câblage, la photorésistance est associée en série avec une résistance de **10kOhm**.



4. Projet : Potentiomètre (au choix)

Un **potentiomètre** est une résistance variable à trois bornes, dont une est reliée à un curseur se déplaçant sur une piste résistante terminée par les deux autres bornes. Ce système permet de recueillir, entre la borne reliée au curseur (point milieu) et une des deux autres bornes, une tension qui dépend de la position du curseur et de la tension à laquelle est soumise la résistance. Mesurer la tension au point milieu en connectant le curseur à la broche A0 du microcontrôleur.



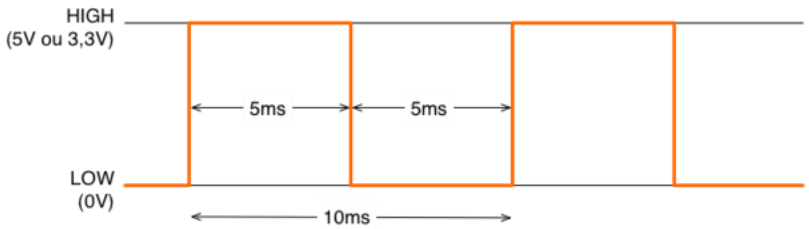
V. Signaux impulsionnels modulés en largeur d'impulsion (PWM)

La commande de nombreux actionneurs nécessite une intensité de tension variable dans la gamme 0-5V, par exemple pour faire varier la vitesse de rotation d'un moteur ou encore pour faire varier l'intensité lumineuse d'une led.

Le microcontrôleur a la possibilité de générer des signaux modulés en largeur d'impulsions (Pulse Width Modulation), ou MLI (Modulation de largeur d'impulsion). Le ***rapport cyclique*** désigne le ratio entre la durée de l'état haut et la période du signal. La modulation en largeur d'impulsion agit donc sur ce rapport cyclique. Un rapport cyclique à 0% correspond à un signal constamment à l'état LOW, et à 100% correspond à un signal constamment à l'état HIGH.

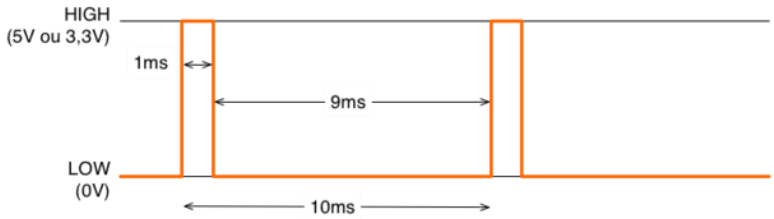
Plusieurs exemples sont donnés sur le site https://www.locoduino.org/spip.php?article47 .

Prenons par exemple un signal de période de 10ms, soit de fréquence de 100Hz. Si la led est allumée pendant 5ms et éteinte pendant 5ms, comme sur la figure ci-dessous, l'impression sera une luminosité de 50% de la luminosité maximale.



PMW à 50% : La fréquence est de 100Hz, le rapport cyclique de 50%

Si la led est allumée pendant 1ms et éteinte pendant 9ms, l'impression sera une luminosité de 10% :



PWM à 10% : La fréquence est de 100Hz et le rapport cyclique de 10%.

A. Configuration PWM

1. Fréquence du signal impulsionnel

La fréquence du signal modulé en largeur d'impulsion est imposée pour les développements Arduino. Pour les cartes les plus utilisées (uno, mega), la fréquence du signal de 490Hz ou 980Hz.

BOARD	PWM PINS	PWM FREQUENCY
Uno, Nano, Mini	3, 5, 6, 9, 10, 11	490 Hz (pins 5 and 6: 980 Hz)
Mega	2 - 13, 44 - 46	490 Hz (pins 4 and 13: 980 Hz)
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)
Uno WiFi Rev.2	3, 5, 6, 9, 10	976 Hz
MKR boards *	0 - 8, 10, A3 (18), A4 (19)	732 Hz
MKR1000 WiFi *	0 - 8, 10, 11, A3 (18), A4 (19)	732 Hz
Zero *	3 - 13, A0 (14), A1 (15)	732 Hz
Due **	2-13	1000 Hz
101	3, 5, 6, 9	pins 3 and 9: 490 Hz, pins 5 and 6: 980 Hz

a

2. Rapport cyclique du signal impulsionnel

Le fonction analogWrite() permet de fixer le rapport cyclique.

Syntax

`analogWrite(pin, value)`

Parameters

pin: the Arduino pin to write to. Allowed data types: int.

value: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: int.

Returns Nothing

B. Exemple : Modulation de l'intensité lumineuse

Un potentiomètre permet de fixer la valeur du rapport cyclique du signal impulsionnel délivré sur une broche connectée à une led.

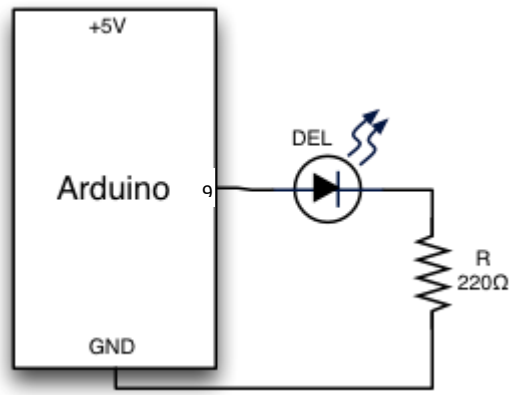
```
int ledPin = 9;      // LED connected to digital pin 9
int analogPin = 3;   // potentiometer connected to analog pin 3
int val = 0;         // variable to store the read value

void setup() {
  pinMode(ledPin, OUTPUT);  // sets the pin as output
}

void loop() {
  val = analogRead(analogPin);  // read the input pin
  analogWrite(ledPin, val / 4); // analogRead values go from 0 to 1023, analogWrite values from 0 to 255
}
```

C. Projets

1. Projet : Modulation intensité lumineuse

Pour ce projet, nous allons nous appuyer sur le projet proposé par Arduino IDE pour illustrer la modulation de largeur d'impulsions : Exemples->Basics-> **Fade.**

Ce programme permet de faire varier la luminosité de la diode en générant un signal modulé en largeur d'impulsion sur la broche 9 du microcontrôleur.




```
int led = 9;           // the pin that the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup()  {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop()  {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

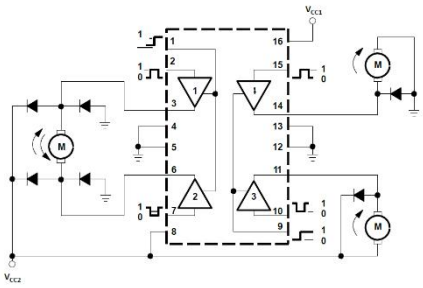
2. Projet : Modulation vitesse de rotation d'un moteur à courant continu

La vitesse d'un courant continu peut être ajustée en commandant celui-ci avec un signal impulsionnel. Un moteur nécessite une alimentation plus puissante que ce que la carte microcontrôleur peut fournir. Ainsi il sera alimenté via un module appelé pont en H permettant de délivrer une intensité de courant jusqu'à 1A. Le circuit L293D permet de piloter 2 moteurs à courant continu. Pour chaque moteur, il y a 3 signaux de commande :

- M1 –PWM qui permet de contrôler la vitesse de rotation
- Les 2 signaux M1 direction permettent de fixer le sens de rotation du moteur :

M1-INA	M1-INB	Sens
LOW	LOW	arrêt
HIGH	LOW	avant
LOW	HIGH	arrière
LOW	LOW	Arrêt

Block diagram



L293D

M1 PWM	1	16	Battery +ve
M1 direction 0/1	2	15	M2 direction 0/1
M1 +ve	3	14	M2 +ve
GND	4	13	GND
GND	5	12	GND
M1 -ve	6	11	M2 -ve
M1 direction 1/0	7	10	M2 direction 1/0
Battery +ve	8	9	M2 PWM
Motor 1			Motor 2

VI. Communications numériques: liaison série asynchrone et synchrone

Le microcontrôleur peut échanger des informations avec d'autres microcontrôleurs ou périphériques (écrans, capteurs numériques, …) via des communications série.

Lorsqu'une communication "série" a été initialisée, les données sont transmises (émises et reçues) bit à bit. Ces communications ont l'avantage de monopoliser peu de broches pour la communication. Ces communications série sont très largement utilisées.

Les protocoles série:

- **UART: Universal Asynchronous serial Receiver and Transmitter**
- **SPI: Serial Peripheral Interface**
- **TWI (I2C): Two-Wire serial Interface(jusqu'à 400kHz)**

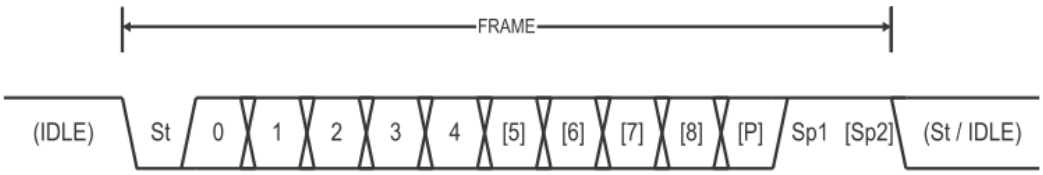
Une transmission série de données est caractérisée par :

- le sens des échanges:
  - simplex : **La liaison simplex** caractérise une liaison dans laquelle les données circulent dans **un seul sens**, c'est-à-dire de l'émetteur vers le récepteur
  - half-duplex : **La liaison half-duplex** caractérise une liaison dans laquelle les données **peuvent circuler dans un sens ou l'autre, mais pas dans les deux sens simultanément**. Avec ce type de liaison chaque extrémité de la liaison émet à son tour.
  - full-duplex : **La liaison full-duplex** caractérise une liaison dans laquelle les données circulent **de façon bidirectionnelle et simultanément**. Chaque extrémité de la ligne peut émettre et recevoir en même temps.

- le type de synchronisation entre l'émetteur et le récepteur:
  - **liaison asynchrone:** les données transmises sont mise en forme avec l'ajout de bits de début et fin de transmission. L'émetteur et le récepteur doivent être configurés de façon à émettre et lire les données à la même vitesse.

    Le récepteur reçoit l'information au rythme où l'émetteur les envoie. La liaison série est constituée de 2 supports nommés TX et RX :
    - La ligne de transmission des données TX
    - La ligne de réception des données RX.

    Chaque émission d'une information est constituée d'une séquence binaire que l'on appelle *format.*
    - une information indiquant le début de la transmission (*bit START*)
    - Les 5,6,7, 8, 9 bits d'information
    - une information permettant de contrôler d'éventuelles erreurs de transmission (bit de parité logique)
    - Suivi d'une information de fin de transmission (1 ou 2 *bit STOP*)



  - **liaison synchrone:** L'émetteur et le récepteur sont synchronisés par un signal d'horloge émis par l'émetteur. Le signal de synchronisation est actif pendant toute la durée de l'émission informant ainsi le récepteur qu'une transmission d'information est en cours. La liaison série est constituée de 3 supports:
    - La ligne de transmission des données TX

- La ligne de réception des données RX.
- L'horloge de synchronisation

A. Liaison série asynchrone : UART

1. Configuration

La liaison série est configurée par la fonction : Serial.begin() qui permet de fixer la vitesse de transmission (en bits par seconde bps) des informations.

Les vitesses typiquement utilisées : 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200.

Un second argument *optionnel* permet de configurer la taille des données, la parité et le nombre de stops bits. **Par défaut, le format de la communication est : 8 data bits, aucun bit de parité, 1 stop bit.**

Deux diodes présentes sur la carte Arduino sont associées à la liaison série :

- TX : s'allume lors d'une transmission
- RX : s'allume lors d'une réception

a) Syntax

Serial.begin(speed)
Serial.begin(speed, config)
Arduino Mega only:
Serial1.begin(speed)
Serial2.begin(speed)
Serial3.begin(speed)
Serial1.begin(speed, config)
Serial2.begin(speed, config)
Serial3.begin(speed, config)

Parameters

speed: in bits per second (baud) - *long*
config: sets data, parity, and stop bits. Valid values are :

- Aucune parité , 1 stop bit : SERIAL_5N1 ; SERIAL_6N1 ; SERIAL_7N1; SERIAL_8N1 (the default)
- Aucune parité , 2 stop bit : SERIAL_5N2 ; SERIAL_6N2 ; SERIAL_7N2 ; SERIAL_8N2
- Parité paire, 1 stop bit: SERIAL_5E1 ; SERIAL_6E1 ; SERIAL_7E1 ; SERIAL_8E1
- Parité paire, 2 stop bit: SERIAL_5E2 ; SERIAL_6E2 ; SERIAL_7E2 ; SERIAL_8E2
- Parité impaire, 1 stop bit: SERIAL_5O1 ; SERIAL_6O1 ; SERIAL_7O1 ; SERIAL_8O1
- Parité impaire, 2 stop bit: SERIAL_5O2 ; SERIAL_6O2 ; SERIAL_7O2; SERIAL_8O2

Returns *nothing*

2. Emission d'une donnée numérique sur la liaison série

Les informations sont émises par l'émetteur sur la liaison série par la fonction: Serial.write()

a) Syntax

Serial.write(val)
Serial.write(str)
Serial.write(buf, len)
Arduino Mega also supports: Serial1, Serial2, Serial3 (in place of Serial)

Parameters

val: a value to send as a single byte
str: a string to send as a series of bytes
buf: an array to send as a series of bytes
len: the length of the buffer

Returns : byte

write() will return the number of bytes written, though reading that number is optional

b) Exemple

```
void setup(){
  Serial.begin(9600);
}

void loop(){
  Serial.write(45); // send a byte with the value 45

   int bytesSent = Serial.write("hello"); //send the string "hello" and return the length of the string.
}
```

Ces informations sont visibles dans la fenêtre « Moniteur Série » de l'environnement de développement préalablement configuré avec une vitesse de transmission de 9600bps.

3. Emission d'une information de type chaine de caractères:

L'information spécifiée en argument est convertie en une chaine de caractère puis émise sur la liaison série par la fonction: Serial.print().

Ces informations sont visibles dans la fenêtre « Moniteur Série » de l'environnement de développement préalablement configuré avec une vitesse de transmission identique à la configuration du microcontrôleur.

a) Syntax

Serial.print(val)
Serial.print(val, format)

Le second paramètre *format* spécifie :

- la base numérique à utiliser lors de la conversion d'une grandeur entière en chaine de caractères : BIN(binary, or base 2), OCT(octal, or base 8), DEC(decimal, or base 10), HEX(hexadecimal, or base 16).
- le nombre de décimales à conserver pour les grandeurs décimales.

Serial.print(78, BIN) gives "1001110"
Serial.print(78, OCT) gives "116"
Serial.print(78, DEC) gives "78"
Serial.print(78, HEX) gives "4E"
Serial.print(1.23456, 0) gives "1"
Serial.print(1.23456, 2) gives "1.23"
Parameters : val: the value to print. Allowed data types: any data type.

Returns : print() returns the number of bytes written, though reading that number is optional. Data type: size_t.

b) Exemple

```
/*
  Uses a for loop to print numbers in various formats.
*/
void setup() {
  Serial.begin(9600); // open the serial port at 9600 bps:
}

void loop() {
for (int x = 0; x < 64; x++) { // only part of the ASCII chart, change to suit
    // print it out in many formats:
    Serial.print(x);       // print as an ASCII-encoded decimal - same as "DEC"
    Serial.print("\t\t");  // prints two tabs to accomodate the label lenght

    Serial.print(x, DEC);  // print as an ASCII-encoded decimal
    Serial.print("\t");    // prints a tab

    Serial.print(x, HEX);  // print as an ASCII-encoded hexadecimal
    Serial.print("\t");    // prints a tab

    Serial.println(x, BIN);  // print as an ASCII-encoded binary
    // then adds the carriage return with "println"
    delay(200);             // delay 200 milliseconds
  }
  Serial.println();        // prints another carriage return
}
```

4. Réception d'une information sur la liaison série

Le données reçues sur la liaison série sont lues par la fonction : Serial.read();

a) Syntax

Serial.read()

Returns

The first byte of incoming serial data available (or -1 if no data is available). Data type: int.

b) Exemple

```
int incomingByte = 0; // for incoming serial data

void setup() {
  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
}

void loop() {
  // send data only when you receive data:
  if (Serial.available() > 0) {
    // read the incoming byte:
    incomingByte = Serial.read();
    // say what you got:
    Serial.print("I received: ");
    Serial.println(incomingByte, DEC);
  }
}
```

5. UART : Travail demandé

Le microcontrôleur est configuré pour émettre et recevoir des informations via sa liaison série.
Les données échangées sont visibles dans la fenêtre moniteur série.

Rédiger un programme un programme qui permet d'éclairer la LED connectée à la broche 13 si l'information reçue sur la liaison série est correspond au caractère «Y » et de l'éteindre si l'information reçue correspond au caractère « N ». Les données à transmettre au microcontrôleur sont à saisir dans la fenêtre Moniteur série de l'environnement de développement préalablement configurée avec le format par défaut et une vitesse de transmission de 9600bps.

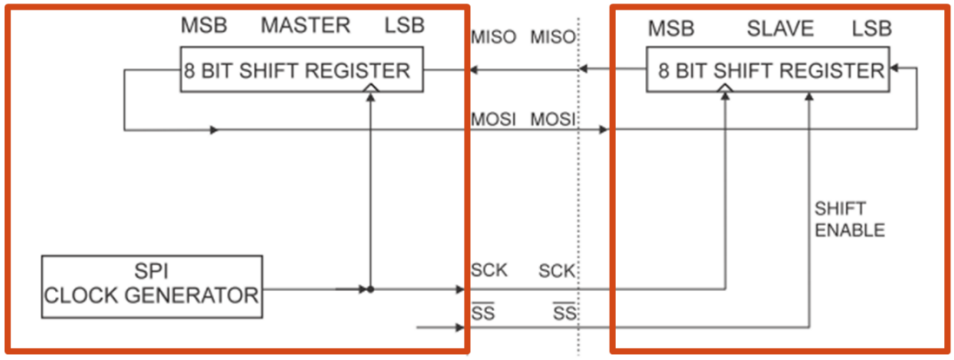
Il est également possible de reprendre le monde sur la commande d'un moteur à courant continu de façon à ce que les caractères reçus permettent de contrôleur le sens et la vitesse de rotation du moteur.

B. Liaison série synchrone : SPI

La liaison SPI (Serial Peripheral Interface) est une liaison série synchrone qui fonctionne en full duplex. La communication est réalisée selon un schéma maître-esclaves, où le maître initie et contrôle la communication. Plusieurs esclaves peuvent être reliés au même bus et la sélection du destinataire se fait par une ligne appelée Slave Select (SS).

Le bus SPI s'appuie sur quatre signaux:

- SCLK : Serial Clock, Horloge (généré par le maître).
- MOSI : Master Output, Slave Input (généré par le maître).
- MISO : Master Input, Slave Output (généré par l'esclave).
- SS : Slave Select, Actif à l'état bas (généré par le maître).

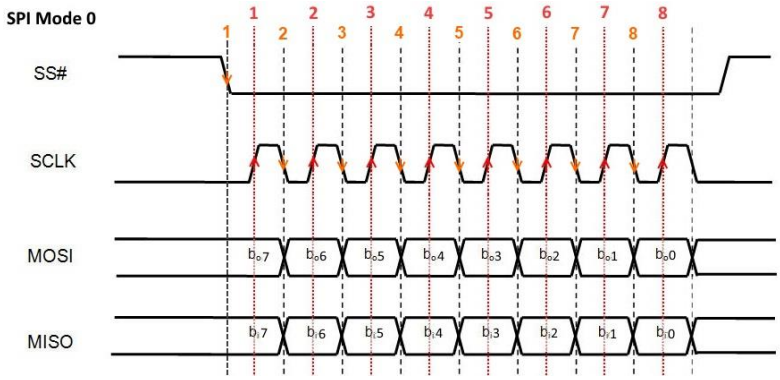


Une transmission SPI typique est une communication **simultanée** entre un maître et un esclave :

- Le module "Maitre" initie la communication en plaçant le signal SS de l'esclave au niveau bas.
- Les 2 modules (maitre et esclave) copient les données à transmettre dans leur registre à décalage respectif.
- Le module "Maitre" génère le signal d'horloge de synchronisation sur la broche SCLK.
- À chaque coup d'horloge le maître et l'esclave s'échangent un bit. Après huit coups d'horloges le maître a transmis un octet à l'esclave et vice versa. La vitesse de l'horloge est réglée selon des caractéristiques propres aux périphériques: Les données sont transférées:

- du maitre vers l'esclave via la broche MOSI (Master Out-Slave In)
- de l'esclave vers le maitre par la broche MISO (Master In-Slave Out).

- Après chaque transfert, le maitre re-synchronise l'esclave en positionnant le signal SS (Slave Select) à l'état haut.



1. SPI : Les fonctions

L'Arduino Uno possède une liaison SPI (SCLK : broche numérique N°13, MISO : broche numérique N°12, MOSI : broche numérique N°11 et SS : broche numérique N°10 et autres si nous avons plusieurs composants esclaves). Les fonctions de configuration et d'échange de données sont définies dans la bibliothèque SPI(), les prototype des fonctions de la bibliothèques sont décrits dans le fichier d'entête SPI.h (`#include <SPI.h>`) . Ses principales fonctions sont :

- **SPI.begin()** : configure les broches de communication du bus SPI en positionnant SCK, MOSI, and SS en sortie, positionnant SCK et MOSI à l'état bas, et SS à l'état haut.
- **SPI.transfer(octet)** : envoie l'octet sur la ligne MOSI ou reçoit l'octet sur la ligne MISO.
- **SPI.end()** : arrête la communication.

Les fonctions suivantes permettent de modifier la configuration définie par défaut :

- **SPI.setBitOrder(sens)** : définit le premier bit qui est transmis où « sens » peut prendre la valeur LSBFIRST si le bit de poids faible est transmis en premier ou MSBFIRST si bit de poids fort est transmis en premier.
- **SPI.setClockDivider(diviseur)** : définit la fréquence d'horloge où diviseur est un multiple de 2 et peut prendre les valeurs SPI_CLOCK_DIV2 à SPI_CLOCK_DIV128.
- **SPI.setDataMode(mode)** : définit le mode de fonctionnement du bus où mode peut prendre les valeurs SPI_MODE0 à SPI_MODE3.

2. Liaison série synchrone SPI : Projet

Plusieurs composants disponibles dans la salle de TP s'appuient sur ce protocole SPI tels que la Matrice de LED 8x8.

C. Liaison série : I2C

La liaison série I2C (*Inter Integrated Circuit*) est une communication série, synchrone bidirectionnelle. Elle permet de faire communiquer entre eux des composants électroniques très divers. Plusieurs équipements peuvent être connectés au bus de communication: un maitre et un ou plusieurs esclaves.

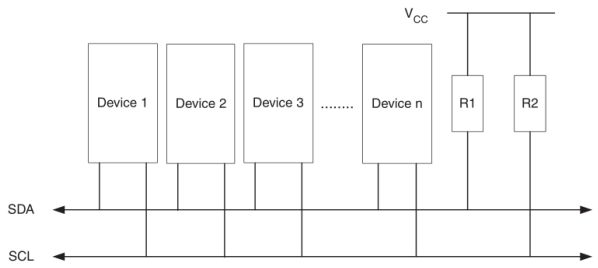
Les échanges ont toujours lieu à l'initiative du maître. Un module peut passer du statut de maître à esclave et réciproquement.

De nombreux capteurs numériques utilisent ce protocole tel que des capteurs de température, de couleur, d'humidité, accéléromètres ou gyroscopes, boussoles, potentiomètres numériques mais également des afficheurs.

La communication est réalisée par l'intermédiaire de deux lignes :

- SDA (Serial Data Line) : ligne de données bidirectionnelle
- SCL (Serial Clock Line) : ligne d'horloge de synchronisation bidirectionnelle. Le signal d'horloge est controlé par le maitre.
- Chacune des lignes (SDA, SCL) doit être connectée à la tension de référence via une résistance de "pull-up"

Chaque périphérique dispose d'une adresse unique (jusqu'128 périphériques interconnectés).



1. Fonctionnement

Seul le maître peut initier une communication. Le microcontrôleur joue le rôle de maitre et les périphériques sont ses esclaves.

Le maître commence par envoyer l'adresse du périphérique dont il désire recevoir les données. L'esclave envoie un premier signal de confirmation (acknowledgement) pour signifier qu'il a bien reçu la demande.

Puis le maître envoie l'adresse d'un registre interne du périphérique. Par exemple, pour un accéléromètre, on a trois registres stockant respectivement les données de l'accélération selon X, Y et Z. Un deuxième signal de confirmation est envoyé par le périphérique.

Enfin c'est le périphérique qui émet cette fois le message, en transférant la valeur du registre qui a été sollicité. Il termine avec une dernière confirmation, après quoi le maître envoie un signal spécifique pour mettre fin à la communication.

Le processus est similaire pour l'écriture sur un périphérique par le microcontrôleur, pour piloter un actionneur ou sauvegarder des données par exemple. Dans ce cas une commande spécifique est envoyée lors de l'adressage du registre et c'est le maître qui transmet les données.

2. Les fonctions

- **Wire.begin();** Initialisation de la communication I2C, à effectuer au début de votre programme.
- **Wire.beginTransmission(Adresse de votre périphérique);** Cette fonction permet d'initialiser une transmission avec le périphérique dont l'adresse est donnée en argument.
- **Wire.write(Adresse du registre sollicité);** Cette fonction permet de préciser l'adresse du registre du périphérique sollicité par une opération d'écriture.
- **Wire.write(Byte à écrire);** Écriture des données dans le registre sollicité au préalable par la fonction *Wire.write*.
- **Wire.requestFrom(Adresse de votre périphérique, N);** Lecture de N bits du registre du périphérique adressé au préalable.

- **Wire.endTransmission();** Fin de la transmission. Cette fonction doit être appelée à chaque fois, et avant l'envoi de toute nouvelle requête par I2C.

3. Les Projets

Plusieurs composants disponibles dans la salle de TP s'appuient sur ce protocole I2C tels que le capteur de luminosité, matrice de type bar-graph, capteur de couleur.

VII. Les IHM

Une interface homme machine est une interface utilisateur permettant de connecter une personne à une machine, à un système ou à un appareil. En théorie, il est donc possible d'utiliser ce terme pour définir n'importe quel écran permettant à un utilisateur d'interagir avec un appareil. Cependant, il est généralement utilisé pour le contexte d'un processus industriel.

Ces interfaces peuvent être :

- Interfaces Matérielles et constituées d'afficheurs et de composants permettant l'intervention de l'utilisateur tels que joysticks, interrupteurs, bouton poussoirs.
- Interfaces Logicielles et communicants avec le microcontrôleur via
    - Une liaison de type bluetooth
    - Une liaison de type wifi

A. Liaison bluetooth

Il est possible de créer une communication bluetooth entre le microcontrôleur et d'autres modules. Pour cela, le circuit HC-05 associé au microcontrôleur permet d'émettre et de recevoir des données en bluetooth. Pour cela le module HC-05 est connecté au microcontrôleur via les broches d'une liaison série asynchrone. Les échanges de données entre le microcontrôleur et le circuit se font via une liaison série asynchrone. Les données sont ensuite émises selon le liaison bluetooth vers le module appairé.

La communication bluetooth est une communication radio fréquence 2.4GHz.

Il est ainsi possible de développer une IHM sur Android d'un téléphone mobile pour communiquer avec le microcontrôleur.

Plusieurs applications android sont disponibles à cet effet dans le play store de Google :

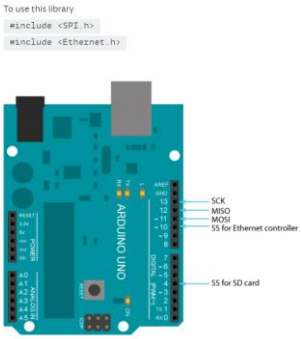
- Arduino Bluetooth controller
- Arduino Bluetooth controlr
- ArduController

B. Liaison wifi

Le shield ethernet associé permet d'autoriser le microcontroleur auquel il est associé à se connecter sur internet. Le module peut alors jouer le rôle de serveur ou de client.

La bibliothèque de fonctions associées est Ethernet.h. Les fonctions sont présentées ici : https://www.arduino.cc/en/reference/ethernet

Le microcontrôleur communique avec le shield via une liaison série synchrone SPI.



VIII. Idées de Projets

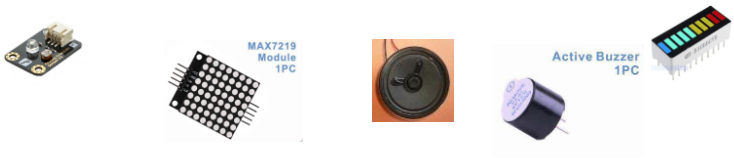
A. Alarme : Système de contrôle des accès

Les accès d'une pièce sont sécurisés. Le système de sécurité est activé par un code saisi sur un clavier matriciel ou par une carte rfid Les intrusions peuvent détectées par plusieurs capteurs de type capteur de mouvement, capteur de distance, capteur de son. Les alertes sont données par des dispositifs lumineux et sonores.
L'interface avec l'utilisateur peut être purement matérielle afficheur et clavier ou logcielle.
De nombreux éléments de présentation de capteurs et actionneurs sont à disposition (http://www.elegoo.com/download/) et libres pour le téléchargement : **Elegoo Mega 2560 The Most Complete Starter Kit, Elegoo Smart Robot Car Kit V3.0.**

B. Instrument de musique numérique

L'objectif de ce projet est de réaliser un instrument de musique capable de lire une partition musicale écrite en niveaux de gris et de l'accompagner d'une ambiance lumineuse.

- ✓ Capteur de niveau de gris
- ✓ Buzzer + haut parleur
- ✓ Matrice de led



De nombreux éléments de présentation de capteurs et actionneurs sont à disposition (http://www.elegoo.com/download/) et libres pour le téléchargement : **Elegoo Mega 2560 The Most Complete Starter Kit, Elegoo Smart Robot Car Kit V3.0.**

C. Commande des déplacements d'un robot et de son système lumineux

- Il s'agit de proposer un projet permettant la réalisation d'un robot piloté par une application android communicant en Bluetooth ou une télécommande infra-rouge. L'application permettra le contrôle des mouvements du robot ainsi que de ses signaux lumineux.
- Le robot peut être équipé d'un détecteur d'obstacle qui peut renvoyer des informations au système de pilotage.
- De nombreux éléments de présentation de capteurs et actionneurs sont à disposition (http://www.elegoo.com/download/) et libres pour le téléchargement : **Elegoo Mega 2560 The Most Complete Starter Kit, Elegoo Smart Robot Car Kit V3.0.**

D. Domotique : Contrôle de température

L'objectif de projet est de réaliser un système capable d'ajuster la température d'une pièce avec un ventilateur. L'utilisateur doit avoir le choix entre un mode automatique ou mode manuel avec une télécommande.

- ✓ Capteur de température
- ✓ Moteur + pale de ventilateur

- ✓ Télécommande IR

De nombreux éléments de présentation de capteurs et actionneurs sont à disposition (http://www.elegoo.com/download/) et libres pour le téléchargement : **Elegoo Mega 2560 The Most Complete Starter Kit, Elegoo Smart Robot Car Kit V3.0.**

E. Domotique : Contrôle de l'intensité de l'éclairage

L'objectif de ce projet est de réaliser un système capable de commander les lumières émises par des diodes RGB (couleur, intensité) en fonction de l'éclairage ambiant (mode automatique) ou en fonction d'un claquement de main (mode manuel).

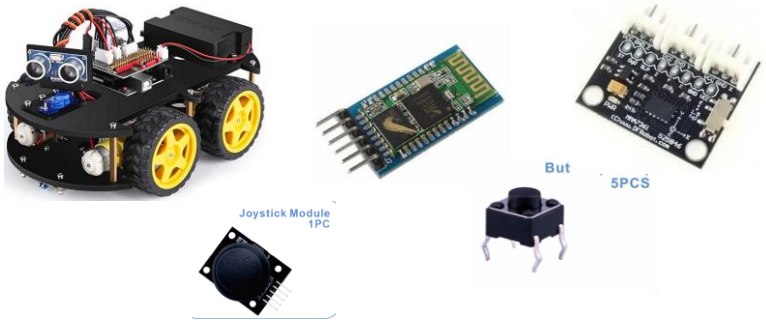
- Capteur de luminosité
- Matrice de diodes – diodes RGB
- Boutons poussoirs
- Capteur de son

De nombreux éléments de présentation de capteurs et actionneurs sont à disposition (http://www.elegoo.com/download/) et libres pour le téléchargement : **Elegoo Mega 2560 The Most Complete Starter Kit, Elegoo Smart Robot Car Kit V3.0.**

F. Domotique : Contrôle de la vitesse de rotation d'un ventilateur

L'objectif de ce projet est de concevoir une télécommande pour piloter un robot arduino à base d'un circuit accéléromètre et un joystick:

- Robot Arduino à monter
- Accéléromètre
- Joystick
- Bouton poussoir
- Module Module Bluetooth



De nombreux éléments de présentation de capteurs et actionneurs sont à disposition (http://www.elegoo.com/download/) et libres pour le téléchargement : **Elegoo Mega 2560 The Most Complete Starter Kit, Elegoo Smart Robot Car Kit V3.0.**

IX. Annexe

A. Brochage Arduino Mega

Pin Number	Pin Name	Mapped Pin Name
1	PG5 (OC0B)	Digital pin 4 (PWM)
2	PE0 (RXD0/PCINT8)	Digital pin 0 (RX0)
3	PE1 (TXD0)	Digital pin 1 (TX0)
4	PE2 (XCK0/AIN0)	
5	PE3 (OC3A/AIN1)	Digital pin 5 (PWM)
6	PE4 (OC3B/INT4)	Digital pin 2 (PWM)
7	PE5 (OC3C/INT5)	Digital pin 3 (PWM)
8	PE6 (T3/INT6)	
9	PE7 (CLKO/ICP3/INT7)	
10	VCC	VCC
11	GND	GND
12	PH0 (RXD2)	Digital pin 17 (RX2)
13	PH1 (TXD2)	Digital pin 16 (TX2)
14	PH2 (XCK2)	
15	PH3 (OC4A)	Digital pin 6 (PWM)
16	PH4 (OC4B)	Digital pin 7 (PWM)
17	PH5 (OC4C)	Digital pin 8 (PWM)
18	PH6 (OC2B)	Digital pin 9 (PWM)
19	PB0 (SS/PCINT0)	Digital pin 53 (SS)
20	PB1 (SCK/PCINT1)	Digital pin 52 (SCK)
21	PB2 (MOSI/PCINT2)	Digital pin 51 (MOSI)
22	PB3 (MISO/PCINT3)	Digital pin 50 (MISO)
23	PB4 (OC2A/PCINT4)	Digital pin 10 (PWM)
24	PB5 (OC1A/PCINT5)	Digital pin 11 (PWM)
25	PB6 (OC1B/PCINT6)	Digital pin 12 (PWM)
Pin Number	Pin Name	Mapped Pin Name
26	PB7 (OC0A/OC1C/PCINT7)	Digital pin 13 (PWM)
27	PH7 (T4)	
28	PG3 (TOSC2)	
29	PG4 (TOSC1)	

30	RESET	RESET
31	VCC	VCC
32	GND	GND
33	XTAL2	XTAL2
34	XTAL1	XTAL1
35	PL0 (ICP4)	Digital pin 49
36	PL1 (ICP5)	Digital pin 48
37	PL2 (T5)	Digital pin 47
38	PL3 (OC5A)	Digital pin 46 (PWM)
39	PL4 (OC5B)	Digital pin 45 (PWM)
40	PL5 (OC5C)	Digital pin 44 (PWM)
41	PL6	Digital pin 43
42	PL7	Digital pin 42
43	PD0 (SCL/INT0)	Digital pin 21 (SCL)
44	PD1 (SDA/INT1)	Digital pin 20 (SDA)
45	PD2 (RXDI/INT2)	Digital pin 19 (RX1)
46	PD3 (TXD1/INT3)	Digital pin 18 (TX1)
47	PD4 (ICP1)	
48	PD5 (XCK1)	
49	PD6 (T1)	
50	PD7 (T0)	Digital pin 38
51	PG0 (WR)	Digital pin 41
52	PG1 (RD)	Digital pin 40
53	PC0 (A8)	Digital pin 37
54	PC1 (A9)	Digital pin 36
55	PC2 (A10)	Digital pin 35
56	PC3 (A11)	Digital pin 34
57	PC4 (A12)	Digital pin 33
58	PC5 (A13)	Digital pin 32
59	PC6 (A14)	Digital pin 31
60	PC7 (A15)	Digital pin 30
61	VCC	VCC
62	GND	GND

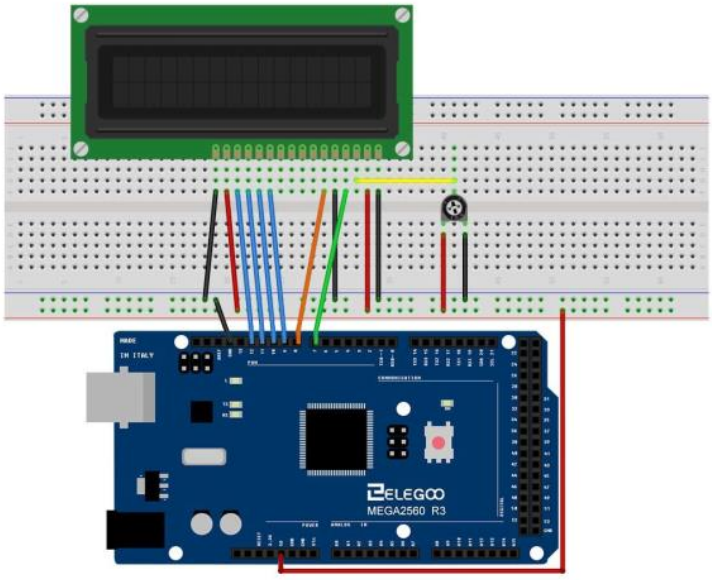
63	PJ0 (RXD3/PCINT9)	Digital pin 15 (RX3)
64	PJ1 (TXD3/PCINT10)	Digital pin 14 (TX3)
70	PG2 (ALE)	Digital pin 39
71	PA7 (AD7)	Digital pin 29
72	PA6 (AD6)	Digital pin 28
73	PA5 (AD5)	Digital pin 27
74	PA4 (AD4)	Digital pin 26
75	PA3 (AD3)	Digital pin 25
76	PA2 (AD2)	Digital pin 24
77	PA1 (AD1)	Digital pin 23
78	PA0 (AD0)	Digital pin 22
79	PJ7	
80	VCC	VCC
81	GND	GND
82	PK7 (ADC15/PCINT23)	Analog pin 15
83	PK6 (ADC14/PCINT22)	Analog pin 14
84	PK5 (ADC13/PCINT21)	Analog pin 13
85	PK4 (ADC12/PCINT20)	Analog pin 12
86	PK3 (ADC11/PCINT19)	Analog pin 11
87	PK2 (ADC10/PCINT18)	Analog pin 10
88	PK1 (ADC9/PCINT17)	Analog pin 9
89	PK0 (ADC8/PCINT16)	Analog pin 8
90	PF7 (ADC7)	Analog pin 7
91	PF6 (ADC6)	Analog pin 6
92	PF5 (ADC5/TMS)	Analog pin 5
93	PF4 (ADC4/TMK)	Analog pin 4
94	PF3 (ADC3)	Analog pin 3
95	PF2 (ADC2)	Analog pin 2
96	PF1 (ADC1)	Analog pin 1
97	PF0 (ADC0)	Analog pin 0
98	AREF	Analog Reference
99	GND	GND
100	AVCC	VCC

A. Brochage Arduino Uno

Pin Number	Mapped Pin Name
1	RESET
2	Digital pin 0 (RX)
3	Digital pin 1(TX)
4	Digital pin 2
5	Digital pin 3 (PWM)
6	Digital pin 4
7	VCC
8	GND
9	
10	
11	Digital pin 5 (PWM)
12	Digital pin 6 (PWM)
13	Digital pin 7
14	Digital pin 8
15	Digital pin 9 (PWM)
16	Digital pin 10 (PWM SS)
17	Digital pin 11 (PWM, MOSI)
18	Digital pin 12 (MISO)
19	Digital pin 13 (SCK)
20	VCC
21	AVref
22	GND
23	Analog Pin 0
24	Analog Pin 1
25	Analog Pin 2
26	Analog Pin 3
27	Analog Pin 4, SDA
28	Analog Pin 5, SCL

A. Configuration: Ecran LCD

Ce projet permet d'écrire des messages et des données numériques sur un écran LCD. Ce programme est proposé par **Elegoo Mega 2560 The Most Complete Starter Kit, l**eçon 22 LCD DISPLAY p149 (http://www.elegoo.com/download/).



Le projet fait appel à la bibliothèque de fonctions LiquidCrystal.

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("Hello, World!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```