



CELLULE ELECTRIQUE

Thème:

ARDUINO : SÉANCE 1

Plan :

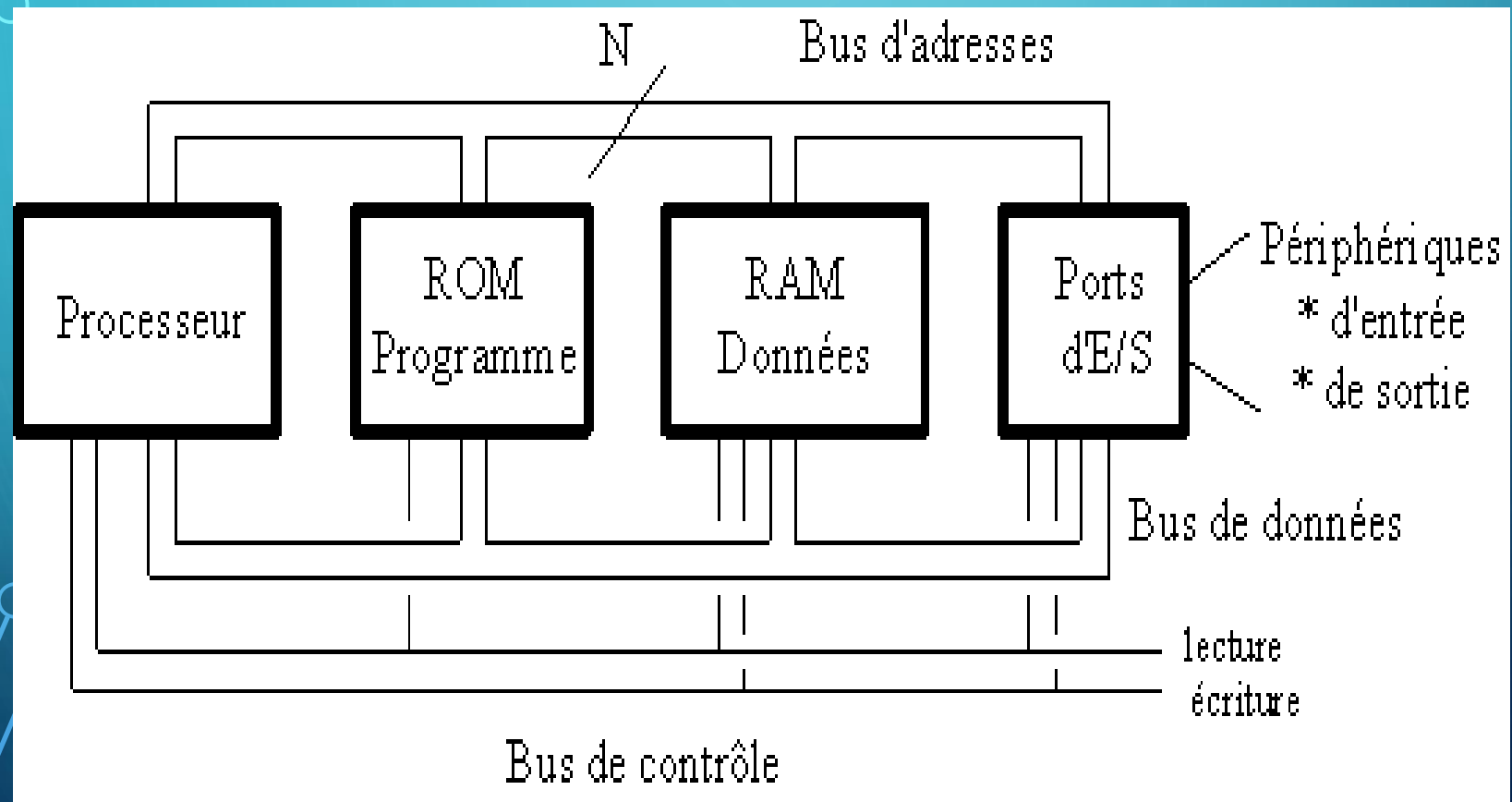
- *C'est quoi déjà un microcontrôleur?*
- *De quoi se constitue une carte Arduino?*
- *Comment programmer?*
- *Outils numériques et analogiques.*
- *Références + soutien (projet éviteur d'obstacles).*

C'est quoi un microcontrôleur :

Un **microcontrôleur** (en notation abrégée **µc**, ou **uc** ou encore **MCU** en anglais) est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte pour le programme, mémoire vive pour les données), unités périphériques et interfaces d'entrées-sorties. Les microcontrôleurs se caractérisent par un plus haut degré d'intégration, une plus faible consommation électrique, une vitesse de fonctionnement plus faible (de quelques mégahertz jusqu'à plus d'un gigahertz¹) et un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels.

Source: Wikipédia.

L'architecture d'un microcontrôleur:

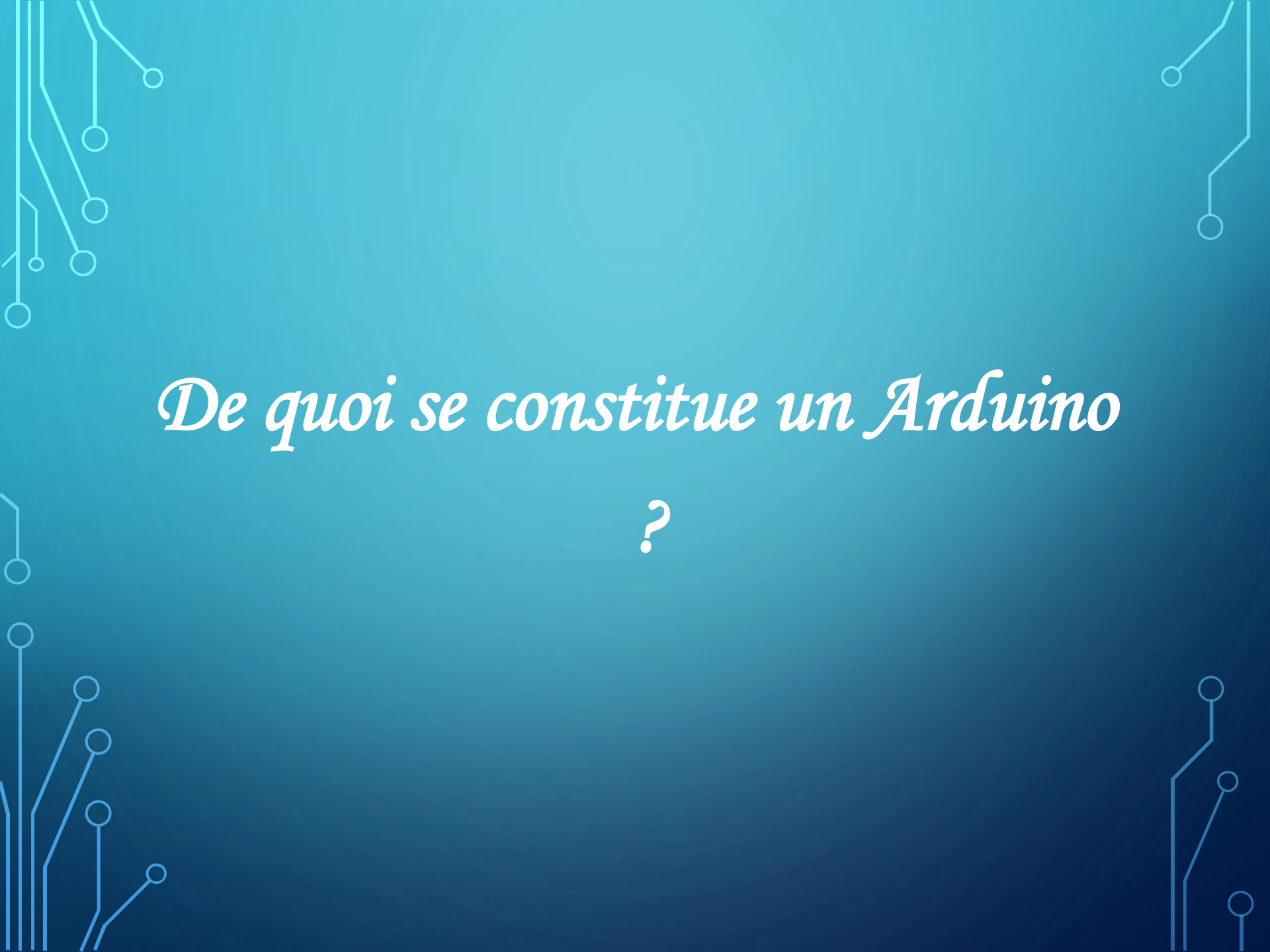


Quelques types de microcontrôleurs :

- **PIC Microcontrôleur.**
- **AVR Microcontrôleur.**
- **ARM Microcontrôleur.**

The background is a solid blue gradient. In the corners, there are white line-art illustrations of circuit traces and nodes, resembling a printed circuit board (PCB) layout. These elements are positioned in the top-left, top-right, bottom-left, and bottom-right corners, framing the central text.

Pourquoi l'Arduino?

The background is a blue gradient with decorative white circuit lines in the corners. The text is centered in a white serif font.

De quoi se constitue un Arduino
?

MODULE ARDUINO-UNO R3

CIRCUIT DE CONVERSION USB / SÉRIE

FUSIBLE 500mA DU PORT USB

EMBASE D'ALIMENTATION EXTERNE

RÉGULATEURS DE TENSIONS +5V ET +3.3V

QUARTZ 16MHz DU MICROCONTRÔLEUR
DE COMMUNICATIONMICROCONTRÔLEUR PRINCIPAL
ATMEL ATMEGA 328P
AVEC "BOOTLOADER" PRÉ-PROGRAMMÉQUARTZ 16MHz DU MICROCONTRÔLEUR
PRINCIPAL

INTERFACE USB

2 LEDS DE VISUALISATION
DE LA COMMUNICATION
SUR LE PORT USBLED DE VISUALISATION RELIÉE
A LA LIGNE D'E/S N°13

TOUCHE D'INITIALISATION

CONNECTEUR ICSP POUR LA
PROGRAMMATION DE L'ATMEL
DE COMMUNICATION

IO Réf.	→
RST (Initialisation)	→
+3.3V (50mA maxi.)	→
+5V (500mA maxi.)	→
Masse (0V)	→
Masse (0V)	→
Alim. (+8 à +12V)	→

E/S digitale 14	E analogique 0	→
E/S digitale 15	E analogique 1	→
E/S digitale 16	E analogique 2	→
E/S digitale 17	E analogique 3	→
E/S digitale 18	E analogique 4	→
E/S digitale 19	E analogique 5	→
I2C SDA		
I2C SCL		



I2C SCL		
I2C SDA		
Tension de Réf.		
Masse (0V)		
E/S digitale 13		
E/S digitale 12		
E/S digitale 11	OC2A	PWM
E/S digitale 10	OC1B	PWM
E/S digitale 9	OC1A	PWM
E/S digitale 8	CLK0	
E/S digitale 7	AIN1	
E/S digitale 6	AIN0	PWM
E/S digitale 5	T1	PWM
E/S digitale 4	T0	
E/S digitale 3	Int 1	PWM
E/S digitale 2	Int 0	
E/S digitale 1	TX ==>	
E/S digitale 0	RX <==	

CONNECTEUR ICSP POUR LA
PROGRAMMATION DU "BOOTLOADER"
DU MICROCONTROLLEUR PRINCIPALLED DE VISUALISATION DE LA
MISE SOUS TENSION

Comparaison entre Arduino Uno et Mega:

UNO		MEGA	
Microcontroller	ATmega328	Microcontroller	ATmega2560
Operating Voltage	5V	Operating Voltage	5V
Input Voltage (recommended)	7-12V	Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V	Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)	Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	6	Analog Input Pins	16
DC Current per I/O Pin	40 mA	DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA	DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader	Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	2 KB (ATmega328)	SRAM	8 KB
EEPROM	1 KB (ATmega328)	EEPROM	4 KB
Clock Speed	16 MHz	Clock Speed	16 MHz



Comment programmer?

Quelques caractéristiques :

-langage machine.

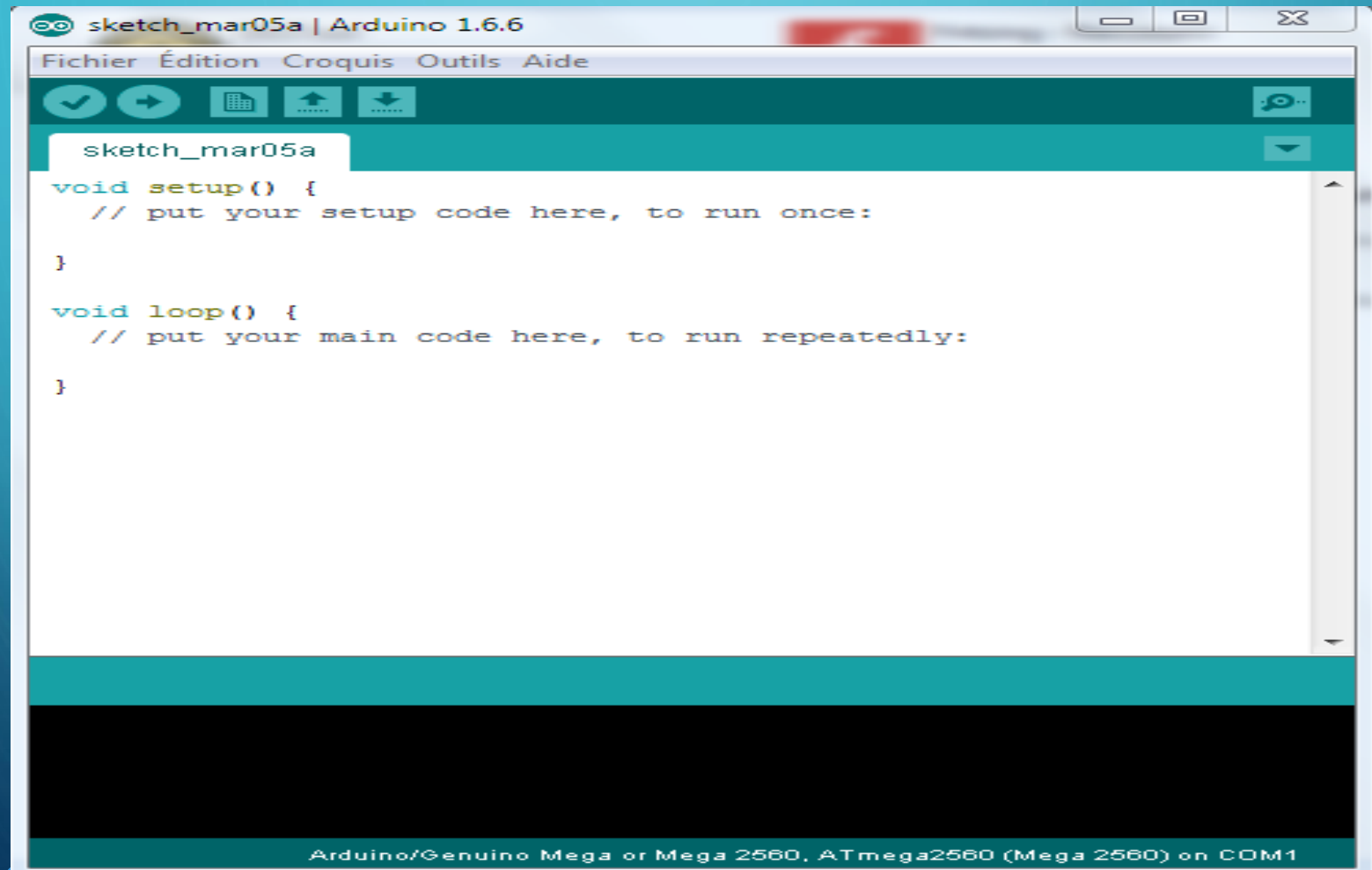
inconconvénients:

- *Il faut savoir pour chaque type son langage.*
- *c'est très difficile de travailler avec.*

solution:

-langage de programmation High level : C, C++.

L'interface du logiciel:



Les fonctions les plus utilisées:

pinMode(pin,b);	boolean	0 = LOW 1 = HIGH
digitalWrite (pin,b);	boolean	
digitalRead (pin);	boolean	
analogWrite (pin,val)	byte	0..255 pins 5,6, 3,11, 9,10
analogRead (pin);	unsigned int	0..1023 pins 14,15,16,17,18,19 A0 .. A5
delay (ms);	unsigned long	0.. $\sim 10^{10}$ millisecondes (~50 jours)
delayMicroseconds ();	unsigned int	0.. 65535 us (~0.06 sec)
millis (temps);	unsigned long	0.. $\sim 10^{10}$ millisecondes (~50 jours) depuis reset
micros(temps);	unsigned long	0.. $\sim 10^{10}$ us (~60 sec)
pulseIn (pin,b);	unsigned long	b=HIGH mesure imp à 1 HIGH attend l'impulsion et mesure sa durée
min(a,b) max(a,b)		choisit le min ou max

Les fonctions les plus utilisées(suite):

<code>constrain (x, a, b) ;</code>		garde la vaeur x entre a et b
<code>map (value, fromLow, fromHigh, toLow, toHigh) ;</code>		applique une règle de 3
<code>bitSet(var,n);</code> <code>bitClear(var,n);</code>	boolean boolean	bits numérotés depuis la gauche. s'applique aussi aux PORTs, DDRs
<code>bitRead(var,n);</code> <code>bitWrite(var,n,b);</code>	boolean boolean	if (!bitRead(PORTC,bMousD)) - vrai si MousD pressée PORTC&(1<<bMousD) préférable
<code>lowByte(var);</code> <code>highByte(var);</code>	int, long int, long	prend les 8 bits de poids faible var&0xFF plus rapide prend les 8 bits de poids fort var&0xFF00 var&0xFF000000
<code>Serial.begin(9600);</code> <code>Serial.end();</code>		setup canal série du terminal désactive, rarement utilisé

Les fonctions les plus utilisées(suite):

tone (pin, fréqHz);		démarre un son
tone (pin, fréqHz,durée);		démarre un son pour la durée spécifiée en ms
notone ();		stoppe le son
shiftOut(D,Ck,dir,val8);		décale 8 bits – très lent
randomSeed(valeur);	int	randomseed (analogRead(A0);) A0 flottant
random(max);	long	valeur rendue entre 0 et max-1
random(min,max);	long	valeur rendue entre min et max-1

Les interruptions:

Une interruption du programme est générée lors d'un événement attendu. Ceci dans le but d'effectuer une tâche, puis de reprendre l'exécution du programme.

Les interruptions:

Pour déclencher une interruption, plusieurs cas de figure sont possibles :

LOW : Passage à l'état bas de la broche.

FALLING : Détection d'un front descendant (passage de l'état haut à l'état bas).

RISING : Détection d'un front montant (pareil qu'avant, mais dans l'autre sens).

CHANGE : Changement d'état de la broche.

Les interruptions:

Pour déclencher une interruption, plusieurs cas de figure sont possibles :

LOW : Passage à l'état bas de la broche.

FALLING : Détection d'un front descendant (passage de l'état haut à l'état bas).

RISING : Détection d'un front montant (pareil qu'avant, mais dans l'autre sens).

CHANGE : Changement d'état de la broche.

Les interruptions:

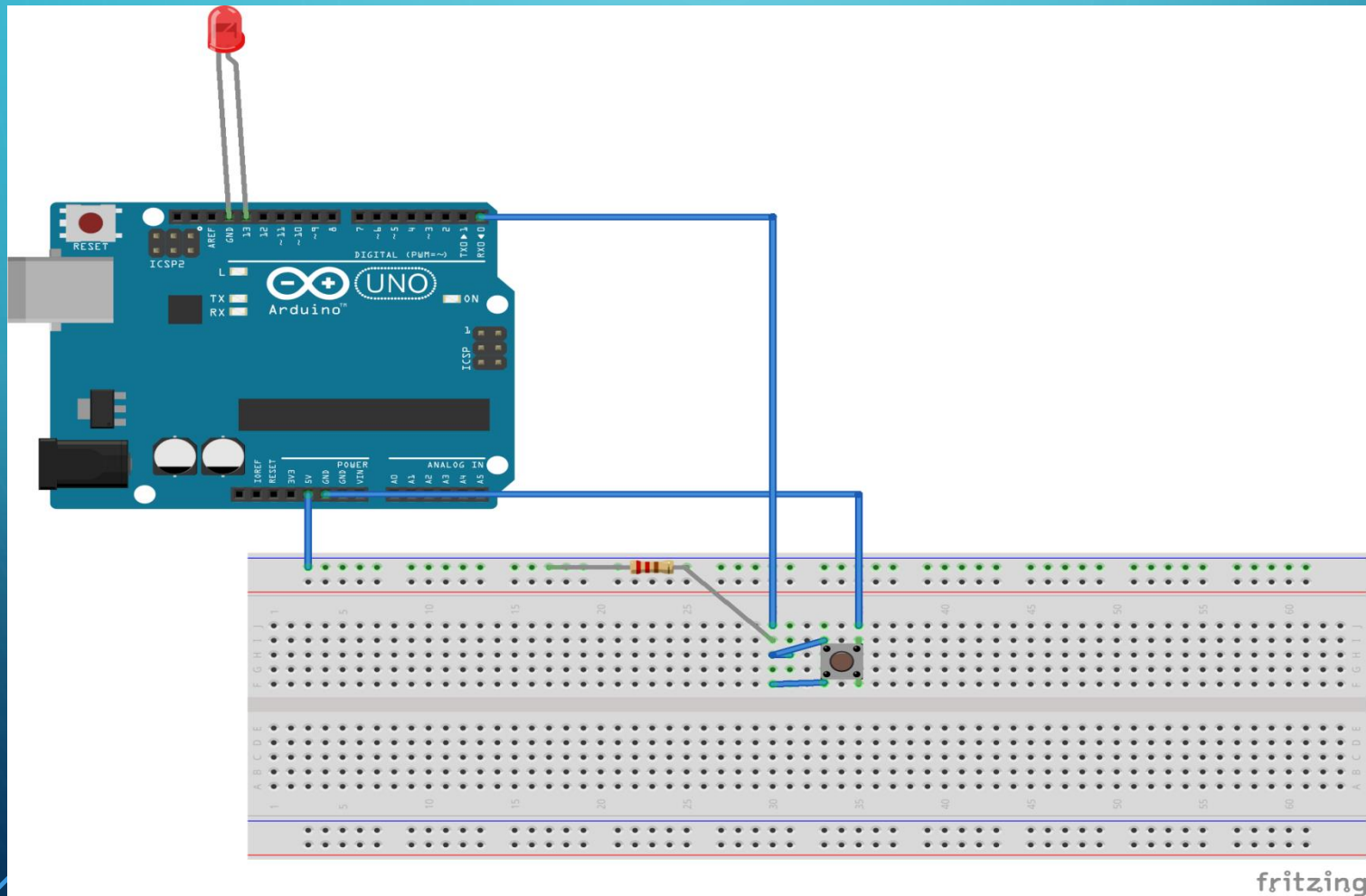
La fonction principale:

attachInterrupt(interrupt, function, mode)

Elle accepte trois paramètres :

- interrupt : qui est le numéro de la broche utilisée pour l'interruption (0 pour la broche 2 et 1 pour la broche 3)*
- function : qui est le nom de la fonction à appeler lorsque l'interruption est déclenchée*
- mode : qui est le type de déclenchement (cf. ci-dessus)*

Les interruptions (exemple):



Les interruptions (exemple):

```
int pin = 13;
volatile int state = LOW; // déclaration d'une variable volatile

void setup()
{
    pinMode(pin, OUTPUT);
    attachInterrupt(0, blink, CHANGE); // attache l'interruption externe n°0 à la fonction blink
}

void loop()
{
    digitalWrite(pin, state); // la LED reflète l'état de la variable
}

void blink() // la fonction appelée par l'interruption externe n°0
{
    state = !state; // inverse l'état de la variable
}
```

Les interruptions (autres fncts utiles):

<code>attachInterrupt(pin,fct,m);</code> <code>detachInterrupt();</code>		Appelle la fonction s'il y a transition selon mode sur la pin 2 ou 3
<code>interrupt();noInterrupt();</code>		active/désactive toutes les interruptions

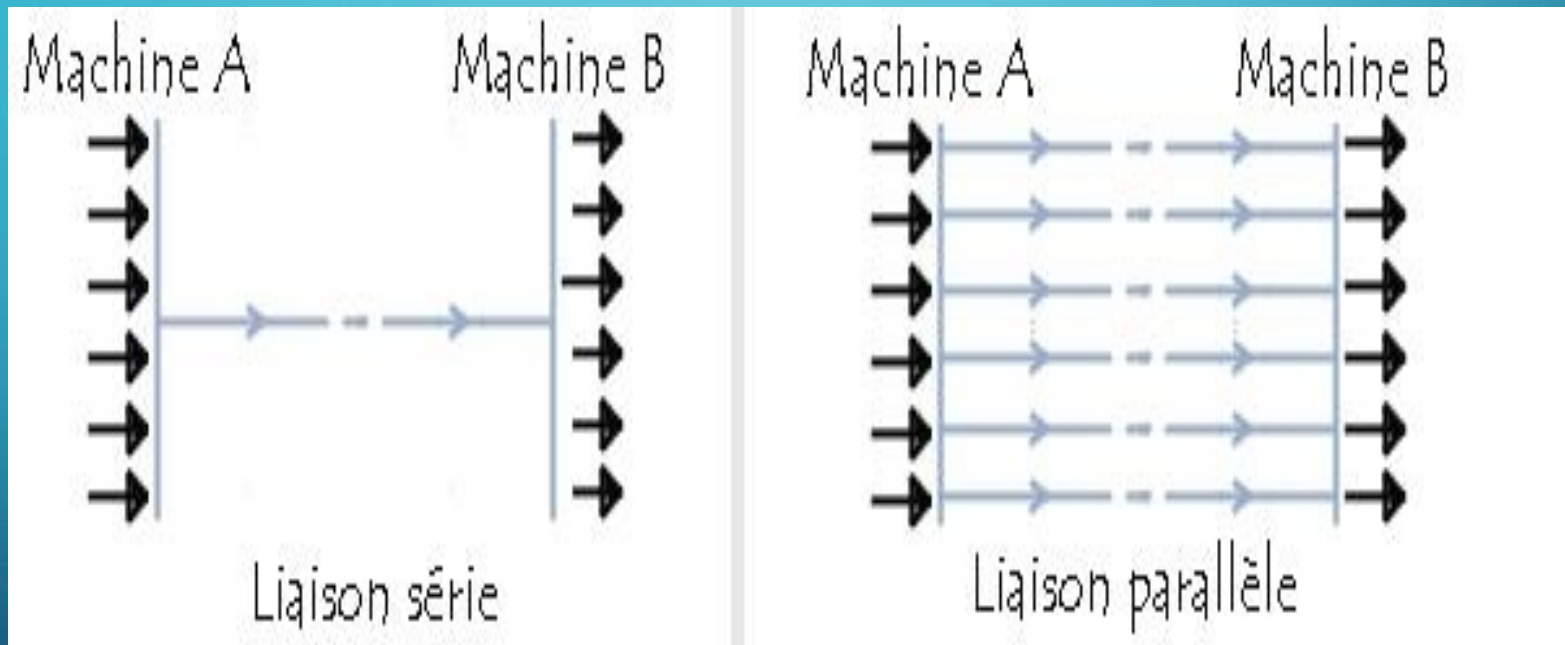
La gestion du port série:

En communication série, on découpe l'information à transmettre en petits blocs de taille fixe avant de la transmettre. La taille des blocs correspond au nombre des lignes disponibles pour la transmission des données.

Ce type de communication s'oppose à la communication parallèle. En communication parallèle, il y a une ligne par bits à transmettre. Tous les bits sont donc transmis en même temps. Pour une même fréquence de communication, la communication parallèle est donc plus rapide.

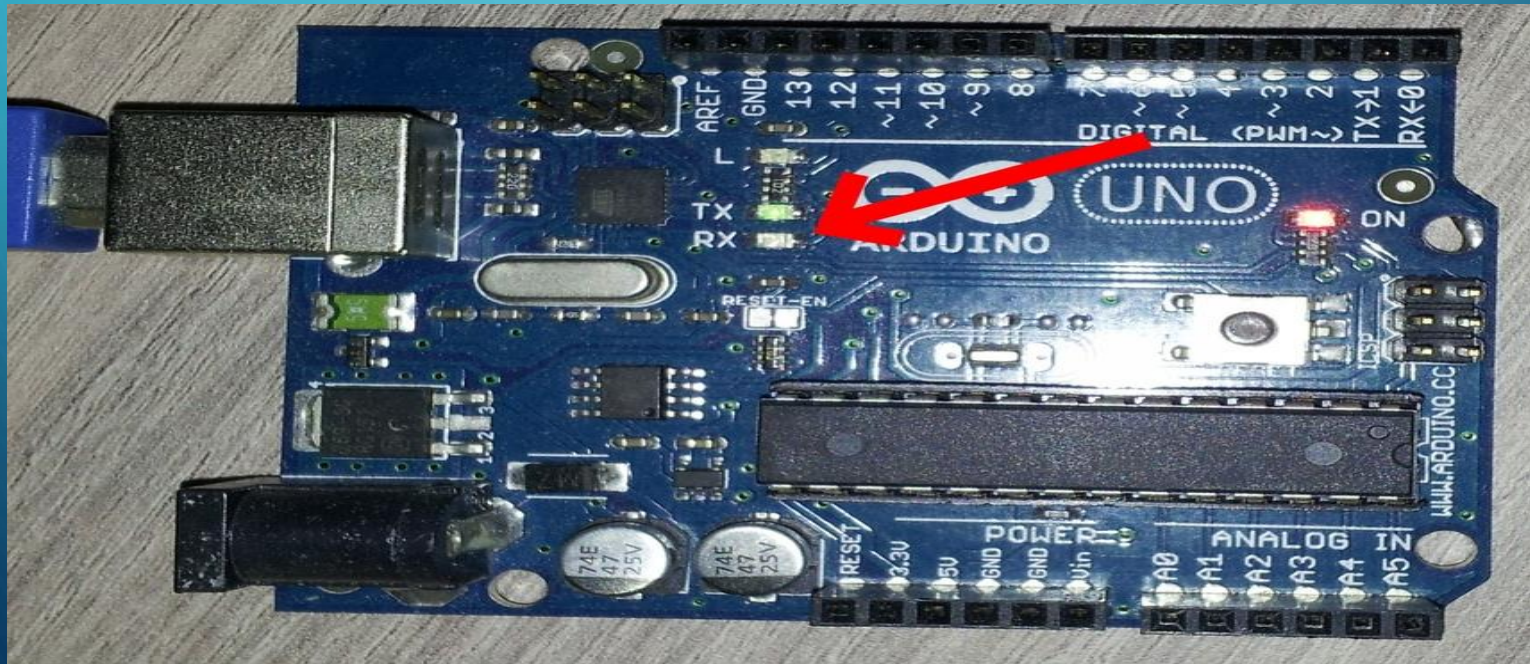
L'avantage de la communication série sur la communication parallèle est qu'elle nécessite moins de lignes, donc moins de broches, donc moins de composants. Son coût est donc plus faible.

La gestion du port série:



La gestion du port série:

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



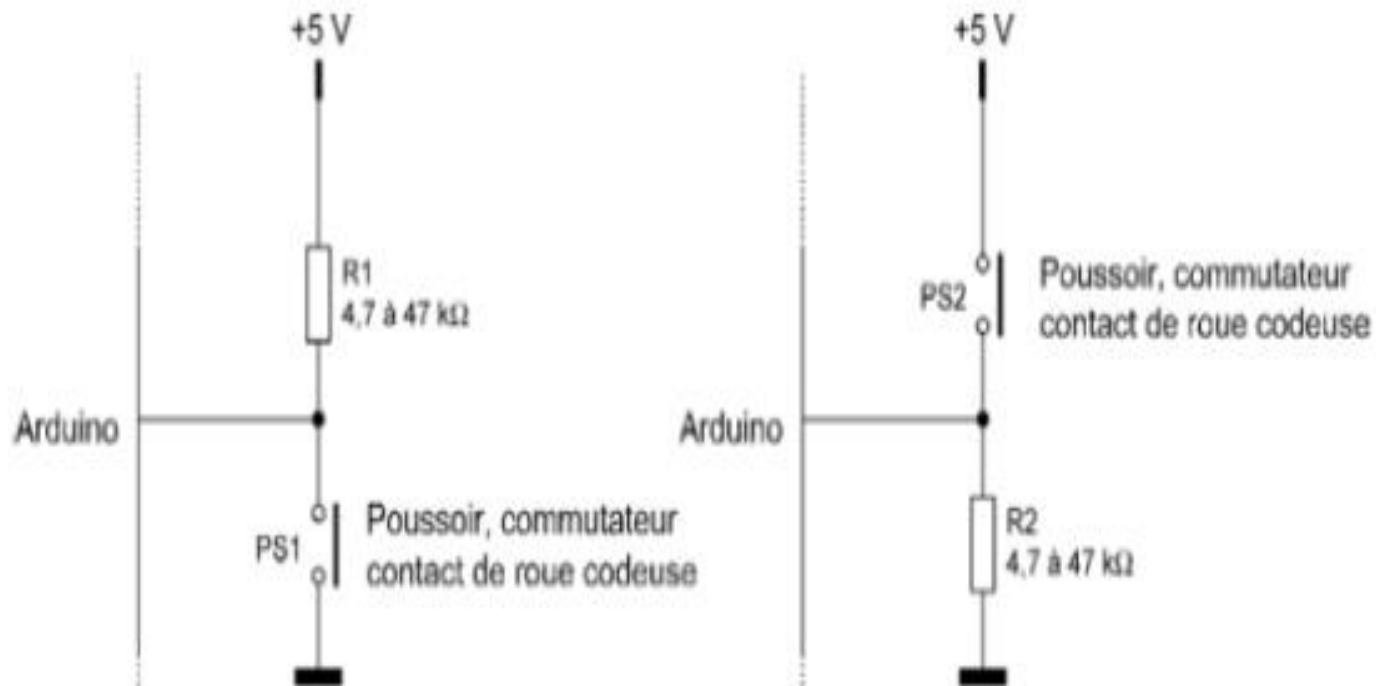
La gestion du port série(qlqs fonctions):

<code>Serial.begin(9600);</code> <code>Serial.end();</code>		setup canal série du terminal désactive, rarement utilisé
<code>Serial.print ();</code> <code>Serial.println();</code>		(75) → 75 (75,BIN) → 100101 (75,HEX) → 4B ("Texte 1""2") → Texte 1"2 '1' code Ascii de 1 = 0x31=49
<code>Serial.write();</code>		(65) → A ("abcd") → abcd
<code>Serial.available()</code> <code>Serial.read();</code> <code>Serial.flush();</code>		if (Serial.available() > 0) { octetRecu = Serial.read(); } vide le tampon

La gestion du port série(exemple):

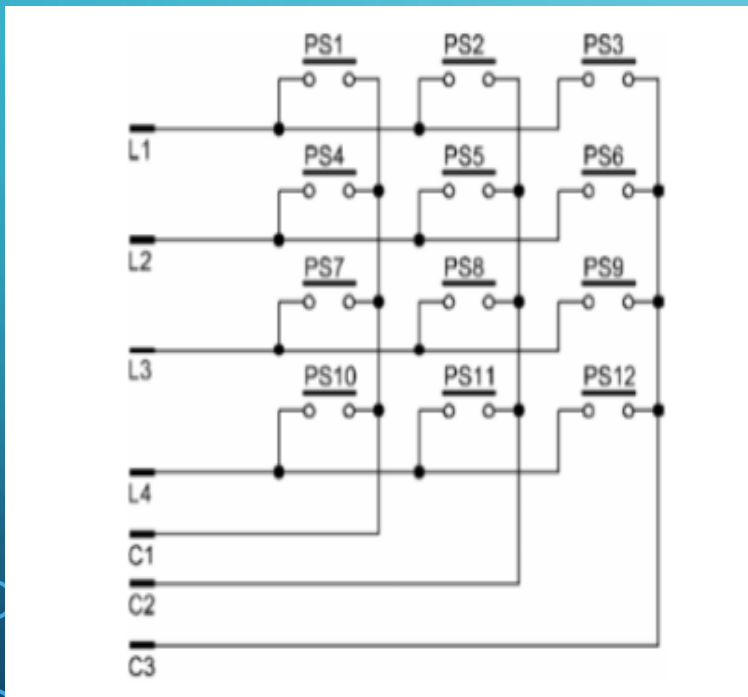
```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  if ( Serial.available() ) {  
    int lu = Serial.read();  
    Serial.println(lu);  
  } else {  
    Serial.println("Rien");  
  }  
  delay(2000);  
}
```

Les entrées numériques parallèles: les boutons poussoirs(montage pull up):



Les entrées numériques parallèles:

Les claviers en matrices:



Les entrées numériques parallèles:

Les claviers en matrices(exemple):

```
#include <Keypad.h>

const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the symbols on the buttons of the keypads
char hexaKeys[ROWS][COLS] = {
  {'0','1','2','3'},
  {'4','5','6','7'},
  {'8','9','A','B'},
  {'C','D','E','F'}
};
byte rowPins[ROWS] = {3, 2, 1, 0}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {7, 6, 5, 4}; //connect to the column pinouts of the keypad

//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup(){
  Serial.begin(9600);
}
void loop(){
  char customKey = customKeypad.getKey();

  if (customKey) {
    Serial.println(customKey);
  }
}
```



Les entrées numériques parallèles:

Les claviers en matrices:

Les fonctions : Séance pratique.





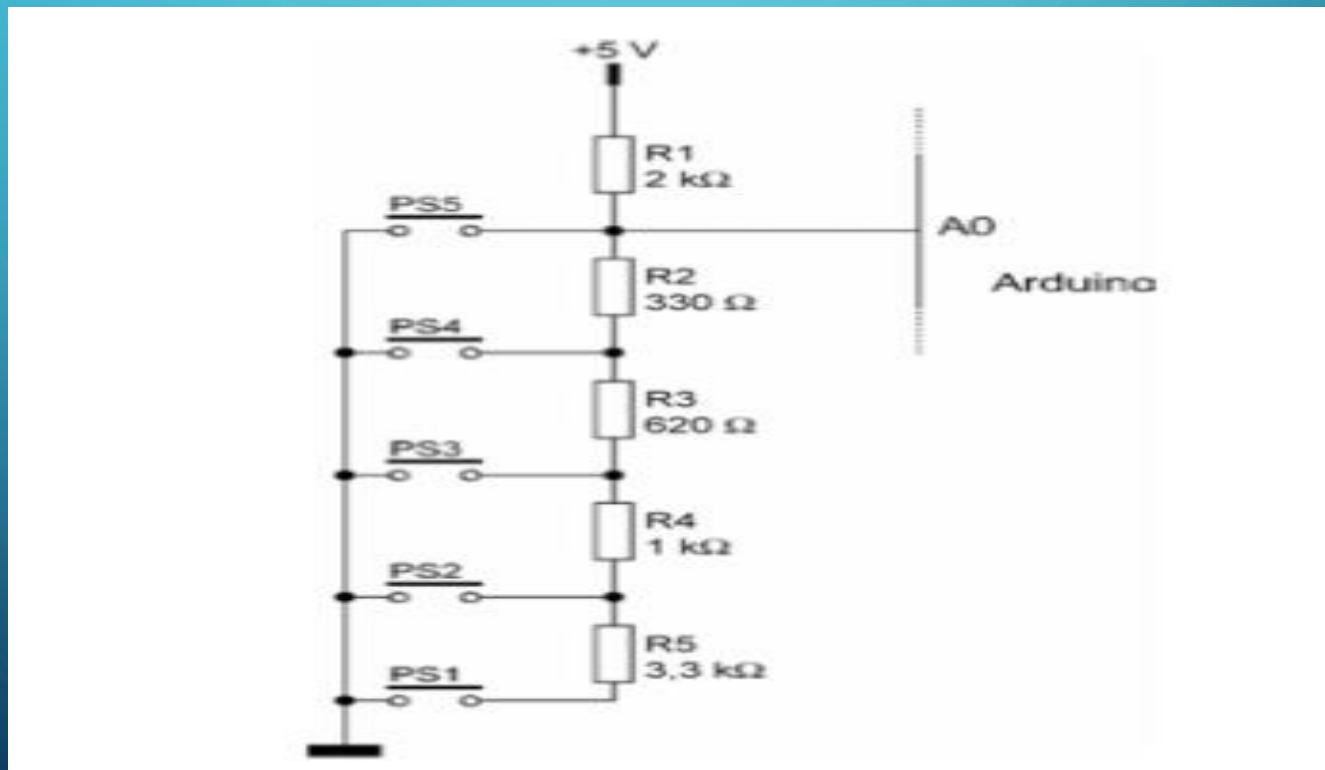
Les entrées numériques parallèles:

*Plusieurs touches avec une seule
entrée ???*



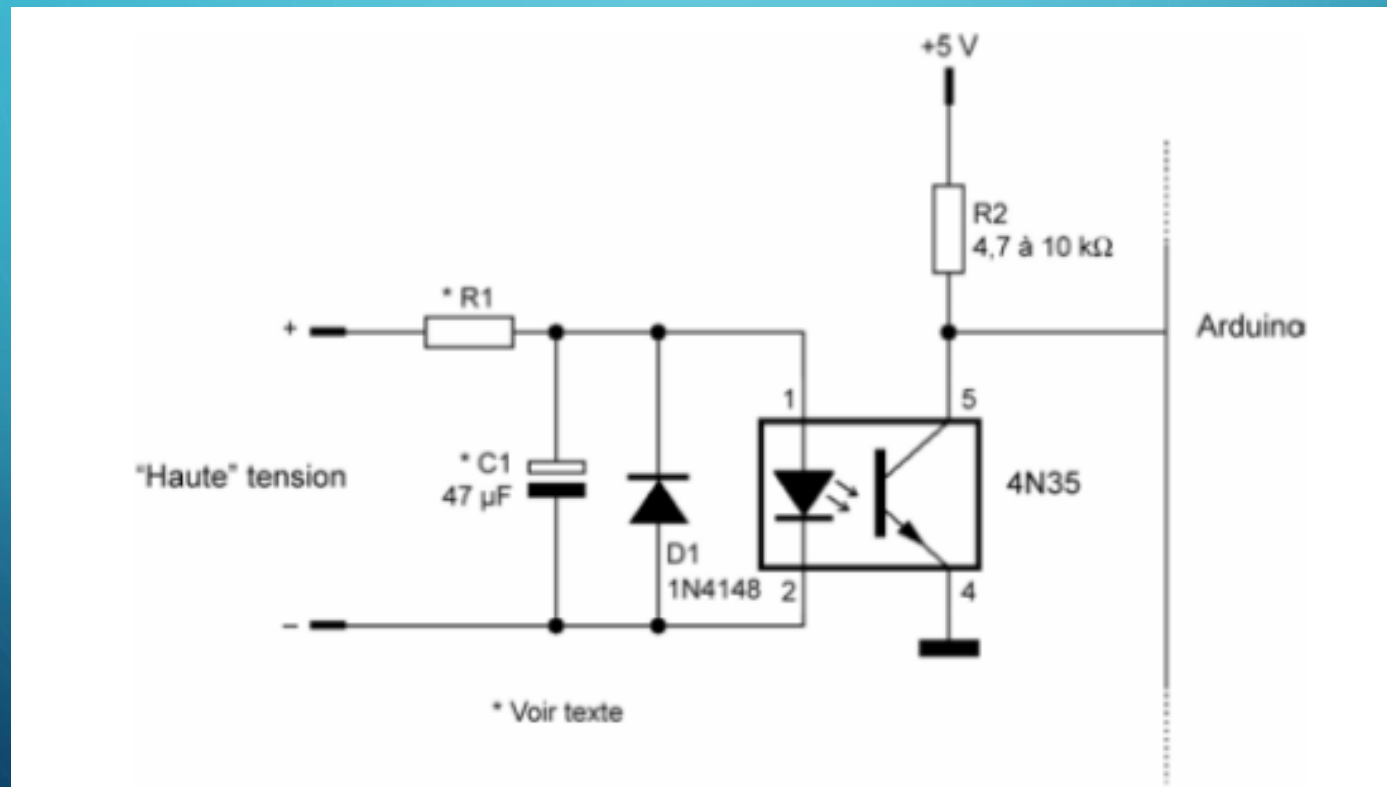
Les entrées numériques parallèles:

Plusieurs touches avec une seule entrée:



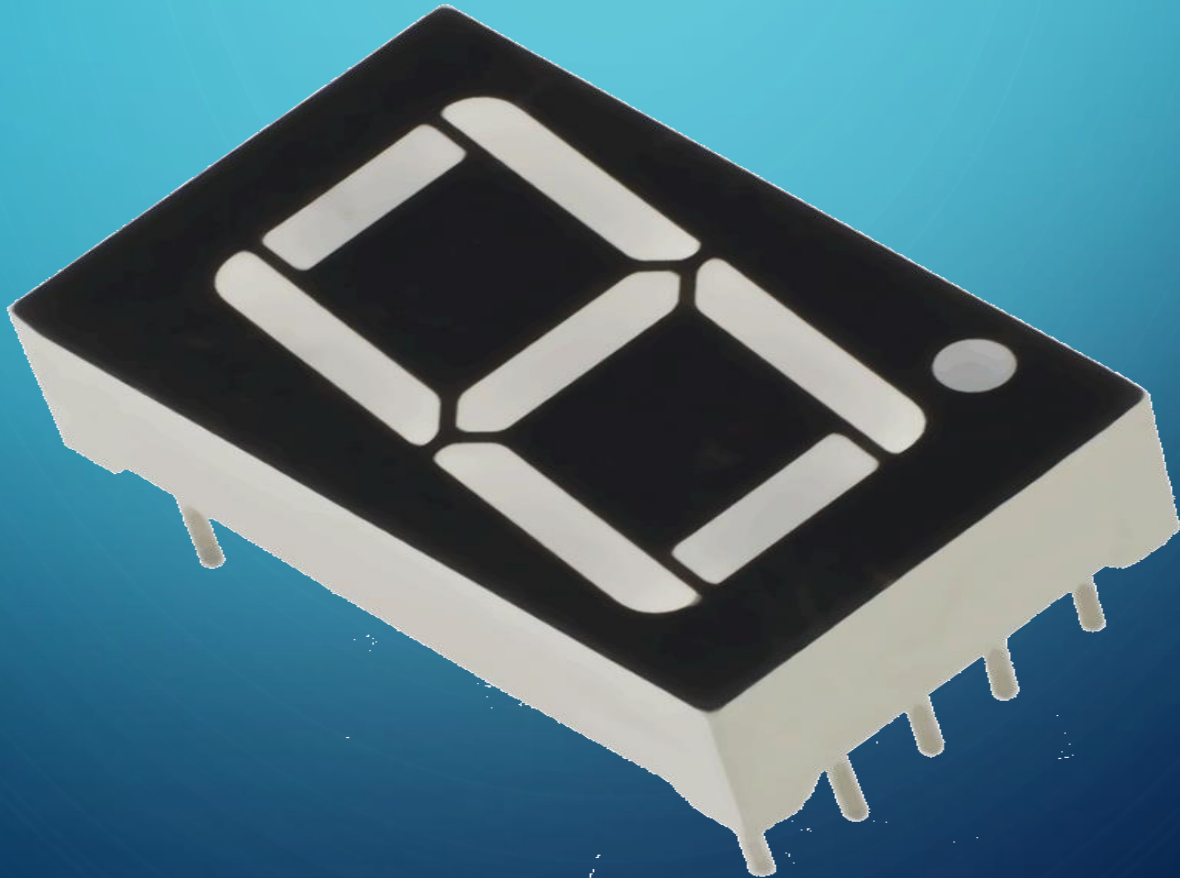
Les entrées numériques parallèles:

Entrée HAUTE tension(photocoupleur):



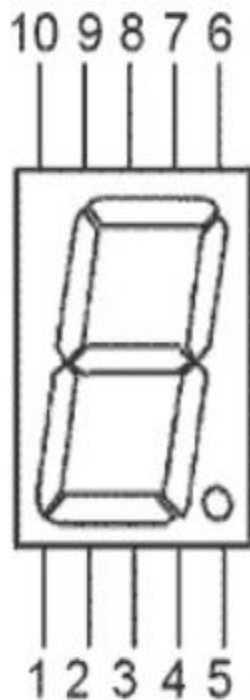
Les entrées numériques parallèles:

Les afficheurs 7 segments:

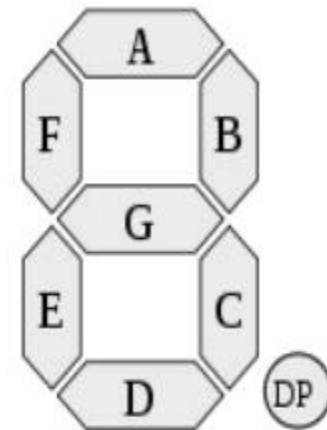


Les entrées numériques parallèles:

Les afficheurs 7 segments:

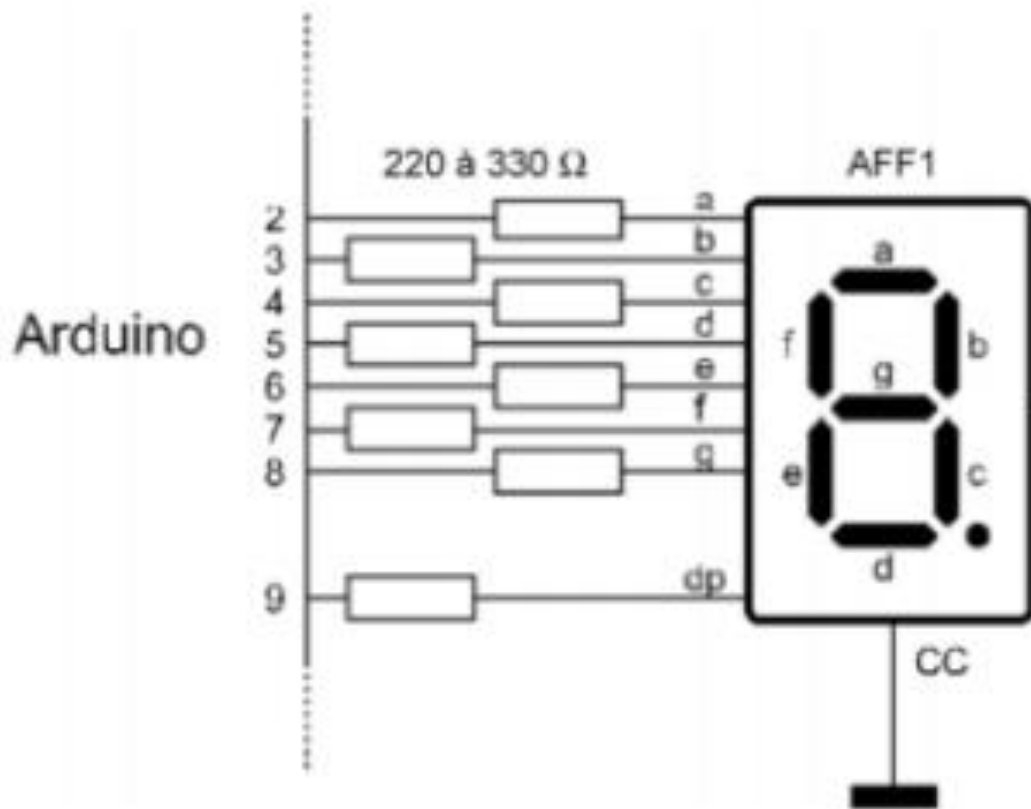


PIN NO.	E MAN6960
1	Cathode E
2	Cathode D
3	Com. Anode
4	Cathode C
5	Cathode D.P.
6	Cathode B
7	Cathode A
8	Com. Anode
9	Cathode F
10	Cathode G



Les entrées numériques parallèles:

Les afficheurs 7 segments:



Les entrées numériques parallèles:

Les afficheurs 7 segments:

```
int segmentPins[] = {2, 3, 4, 5, 6, 7, 8, 9};

byte digits[10][8] = {
  //  a  b  c  d  e  f  g  .
  { 1, 1, 1, 1, 1, 1, 0, 0}, // 0
  { 0, 1, 1, 0, 0, 0, 0, 0}, // 1
  { 1, 1, 0, 1, 1, 0, 1, 0}, // 2
  { 1, 1, 1, 1, 0, 0, 1, 0}, // 3
  { 0, 1, 1, 0, 0, 1, 1, 0}, // 4
  { 1, 0, 1, 1, 0, 1, 1, 0}, // 5
  { 1, 0, 1, 1, 1, 1, 1, 0}, // 6
  { 1, 1, 1, 0, 0, 0, 0, 0}, // 7
  { 1, 1, 1, 1, 1, 1, 1, 0}, // 8
  { 1, 1, 1, 1, 0, 1, 1, 0}  // 9
};

void setup()
{
  for (int i=0; i < 8; i++)
  {
    pinMode(segmentPins[i], OUTPUT);
  }
}

void loop()
{
  int n = 4 ; // Affichage du chiffre 4 à titre d'exemple
  for (int i=0; i < 8; i++)
  {
    digitalWrite(segmentPins[i], digits[n][i]);
  }
}
```




Les entrées numériques parallèles:

Les afficheurs 7 segments:

*Afficher dans 2 afficheurs
simultanément.*



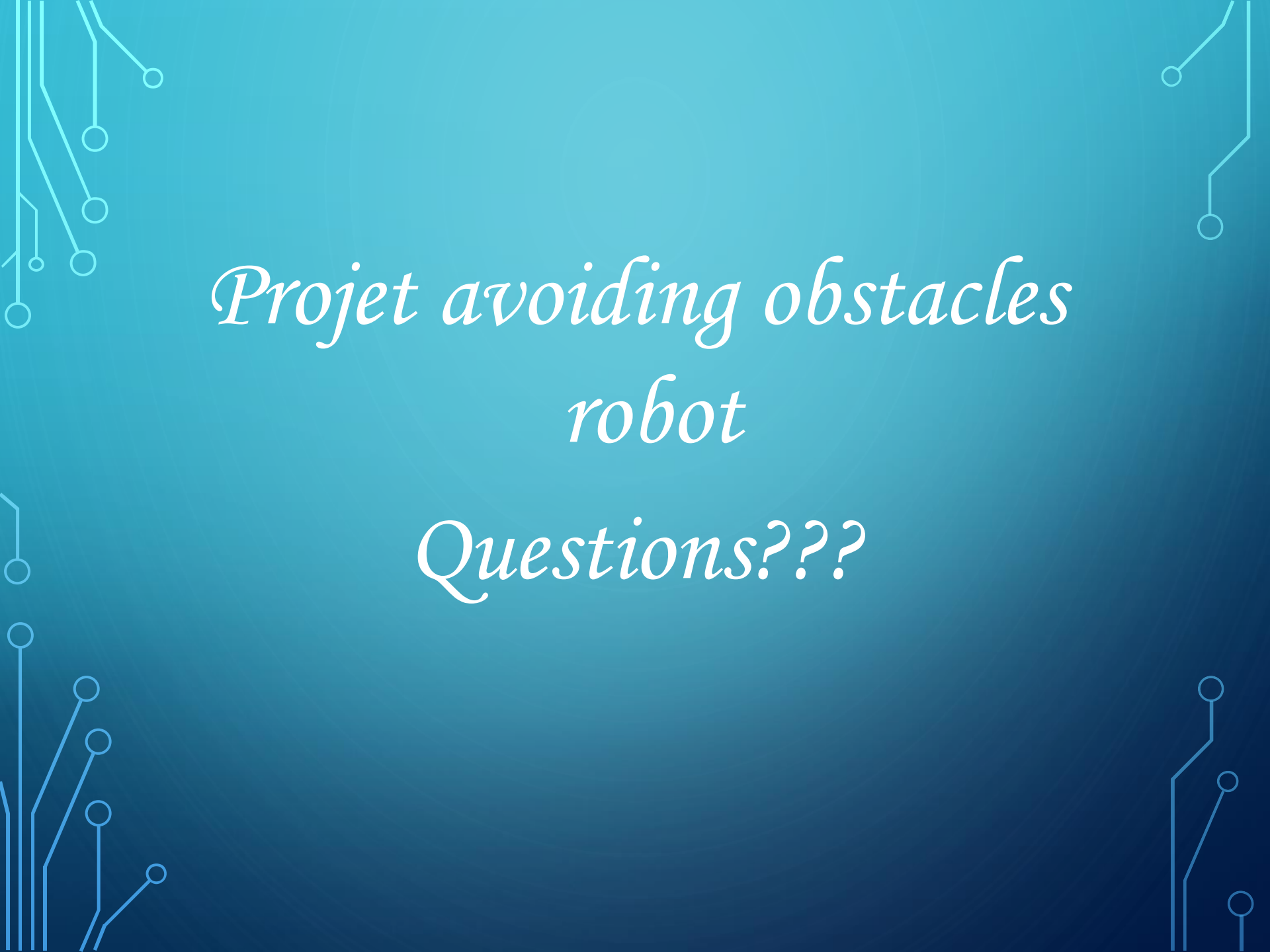
Références :

Livres:

- Arduino pour bien commencer en électronique et en programmation(open class room).**
- Arduino_maitrisez sa programmation et se cartes d'interfaces (shields).(edition DUNOD)**
- Random_nerd_tutorials_project_by_rui_santos.**

Sites:

- instructables.*
- site arduino(forum).*
- Stack_overflow(electrical field).*

The background is a solid blue gradient. In the corners, there are decorative white line art elements resembling circuit boards or neural network connections, with lines and small circles.

Projet avoiding obstacles robot

Questions???