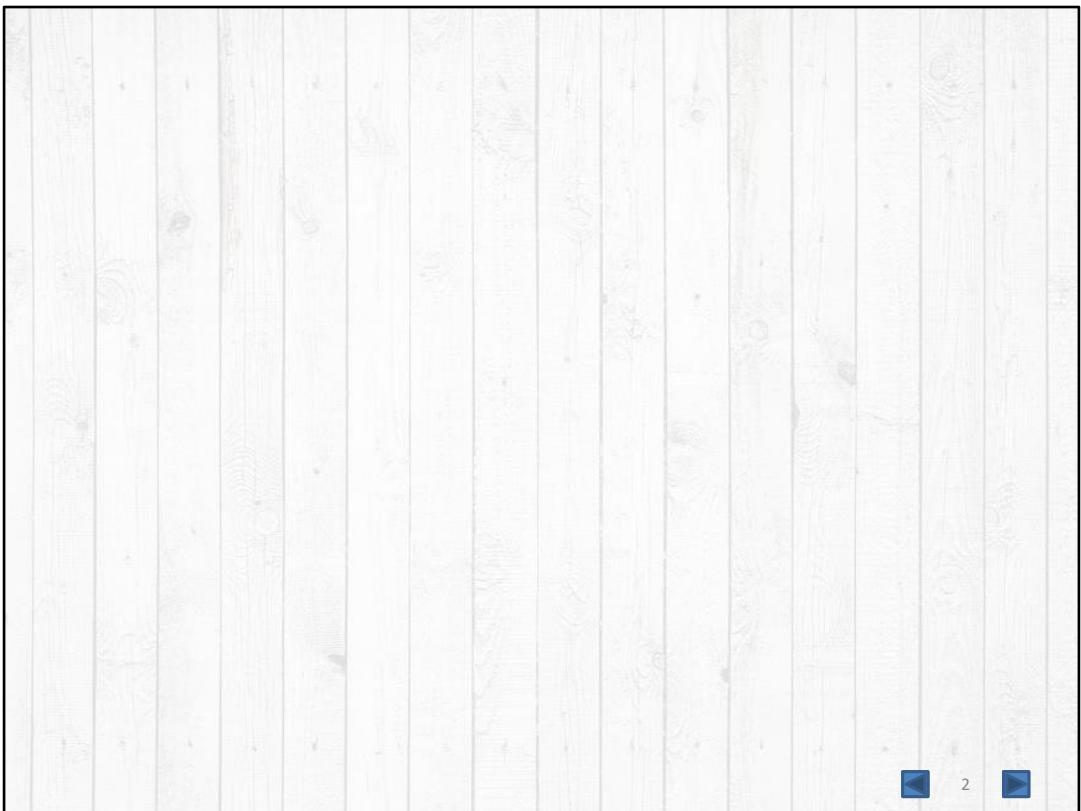


ORSYS
formation

CONCEPTION + MISE
EN ŒUVRE



Arduino



◀ 2 ▶

VOTRE FORMATION AVEC ORSYS

LOGISTIQUE



Alexandre DESORBAIX



- Culture GEEK
- Développeur informatique
- Conseil aux entreprises sur les techniques 3D
- Impression FFF au service d'entreprises et de particuliers



Orsys Formation



- Nombre de jours :
 - 4 jours
- Horaires
 - Début : 9 h
 - Fin : 17 h 30 approx.
- Pause
 - 10 h 30 approx. Matin
 - 12 h 30 approx. Midi
 - 15 h 30 approx. Après-midi



PRÉSENTATION



Sommaire

❑ Dans ce chapitre, nous aborderons :

- ❑ L'open hardware
- ❑ Wiring et arduino
- ❑ Les cartes principales de développement pour ATMEGA
- ❑ Les pinout des cartes
- ❑ Les compositions des processeurs ATMEGA
- ❑ Logique shield & modules



Open source & Electronique

- Un concept communautaire de partage
- En liaison avec Open Sources
- La licence matériel libre désigne que les informations concernant
 - leur conception,
 - les programmes utilisés,
 - ectsont **libres**
 - utilisable, modifiable et redistribuable librement.
- Open hardware mais pas que :



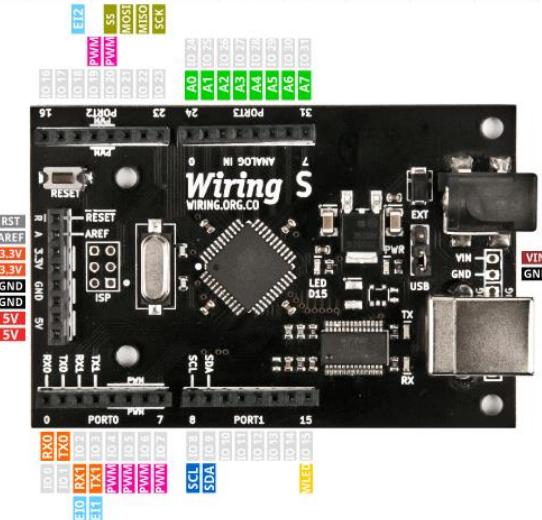
Projet wiring

- Carte wiring

- Atmega 328

- Connectiques non compatible arduino

- Alim USB ou ext.



Projet Arduino

- Bar di Re Arduino
 - Bar(débit de boisson) en Italie de rassemblement des développeurs
- Un arduino c'est une carte électronique avec
 - Un atmegaXXX*
 - Une alim 5v*
 - Un quartz 16Mhz*
 - un format standard
- Les schémas de ces cartes sont en [licence libre](#). Certaines composantes ne sont pas sous licence libre
 - exemple comme le microcontrôleur



* Plusieurs modèles d'arduino existent , certains ont de vitesses supérieures à 16MHz, certains sont sous 3.3V. Plusieurs processeurs existent sur ce type de carte ATMEGA, ATTINY, SAMX



Processeurs Atmel atmega

- Processeur 8 bits

- 16 Mhz sur arduino (Jusqu'à 20Mhz)

- Atmega **168 / 328 ***

- **flash** : 16 Kb ou 32 Kb

- **Format** : SPDIP ou LQFP ou VQFN

- Write cycle :

- » flash 10 000x

- » EEPROM 100 000x

- **EEPROM** 512 b ou 1 Kb

- **SRAM** 1 Kb ou 2 Kb

- **Hardware Interruptions** : 2

- I/O :

- » 13 numériques dont 6 PWM

- » 6 analogiques 10 bits DAC (8sur QFP/QFN)

- Protocoles :

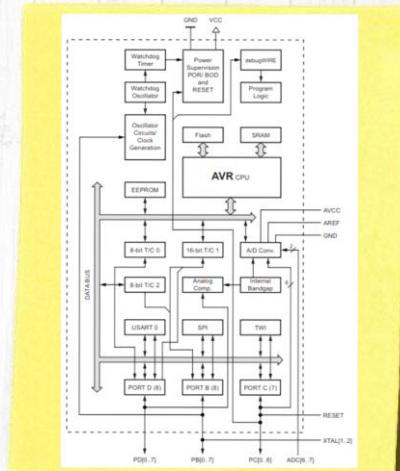
- » **UART, SPI, I²C**



datasheet : http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
<https://www.microchipdirect.com/product/search/all/atmega328>
<https://www.microchipdirect.com/product/atmega2560>

Architecture Atmega328

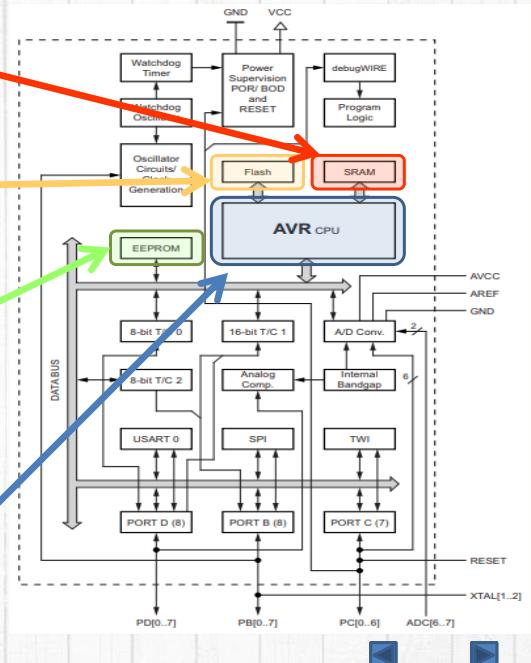
- Architecture Harvard
 - séparation physique
 - la mémoire de données
 - la mémoire programme
 - Atmega 328
 - Datasheet
 - [Liens](#)
 - Un processeur complet
 - de la mémoire, un processeur
 - un bus de données
 - » des convertisseurs
 - » des ports protocolés
 - des timers internes



http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

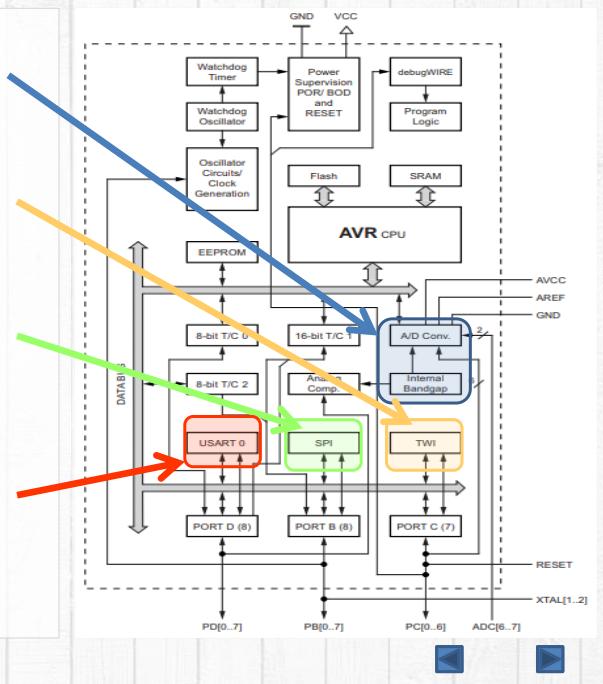
Architecture Atmega328

- Flash
 - 32 Ko
 - contient les programmes de fonctionnements avr
- SRAM
 - 2 ko
 - mémoire de travail
- EEPROM
 - 1 Ko
 - Mémoire à long terme non volatile
- Un CPU type AVR
 - Jusqu'à 20 Mhz cf. osculateur



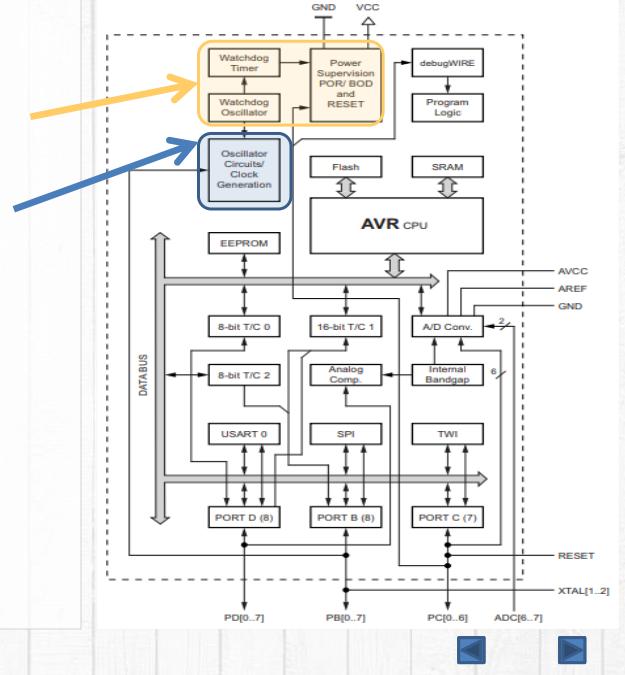
Architecture bus Atmega328

- Un CAN
 - 10 bits
- Un port I²C ou TWI
- Un port SPI
- un port serial

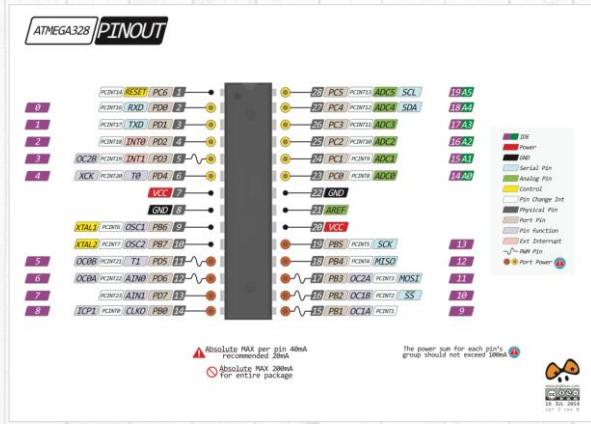


Architecture Horloge Atmega328

- Mode basse consommation
 - Watchdog
 - timer agit sur power supervision
- Génération horloge
 - Interne
 - Circuit



Atmega328 pinout



LISTE DES PINS DU 328

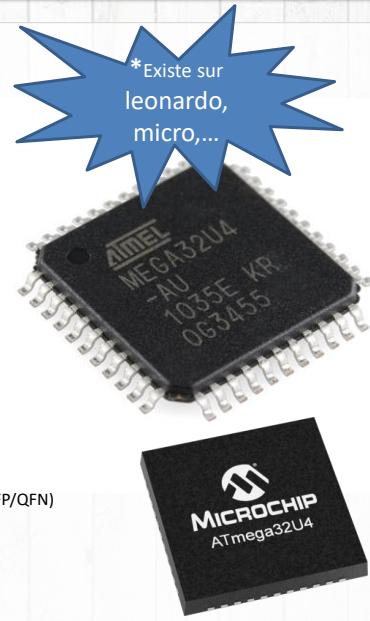
- **INTX = INTERRUPTION MATERIELLE**
- **ADCX = ANALOG ↔ DIGITAL**
- **RX/TX SERIAL PORT**
- **MISO/MOSI/SCK/SS SPI PORT**
- **SCL/SDA I2C PORT**
- **S,11,12,15,16,17 PWM PORT**
- **XTALX CRYSTAL PORT**
- **ATTENTION ZOOMA / CHIP**

Processeurs Atmel atmega 16u4/32U4

- Processeur 8bits

- 16 Mhz

- Atmega **16u4 / 32u4 ***
 - **flash** : **16Kb ou 32Kb**
 - **Format** : LQFP ou VQFN
 - Write cycle :
 - » flash 10 000x
 - » EEPROM 100 000x
 - **EEPROM** 512b ou 1Kb
 - **SRAM** 1.25 Kb ou 2.5 Kb
 - **Hardware Interruptions** : 2
 - I/O :
 - » 13 numériques dont 6 PWM
 - » 4 interruptions externes
 - » 12 analogiques 10bits DAC (8sur QFP/QFN)
 - Protocoles :
 - » UART, SPI, I²C
 - » **JTAG**
 - » **USB 2.0**

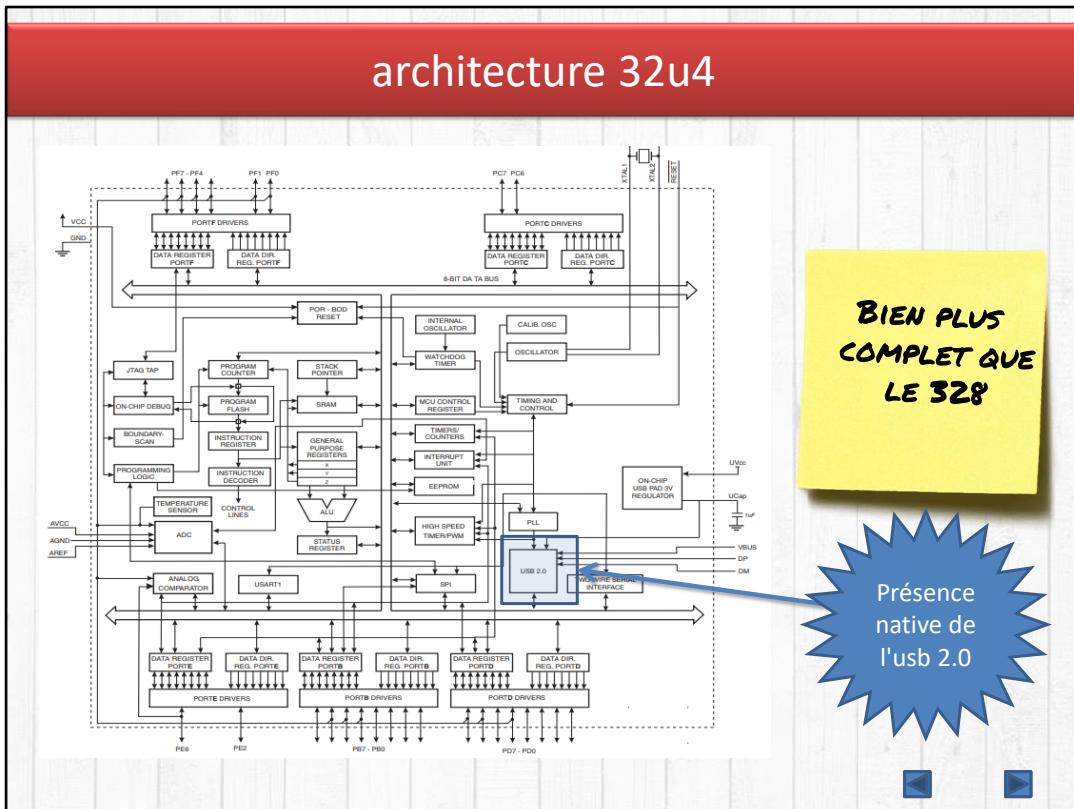


datasheet : http://ww1.microchip.com/downloads/en/devicedoc/atmel-7766-8-bit-avr-atmega16u4-32u4_datasheet.pdf

<https://www.microchipdirect.com/product/search/all/atmega328>

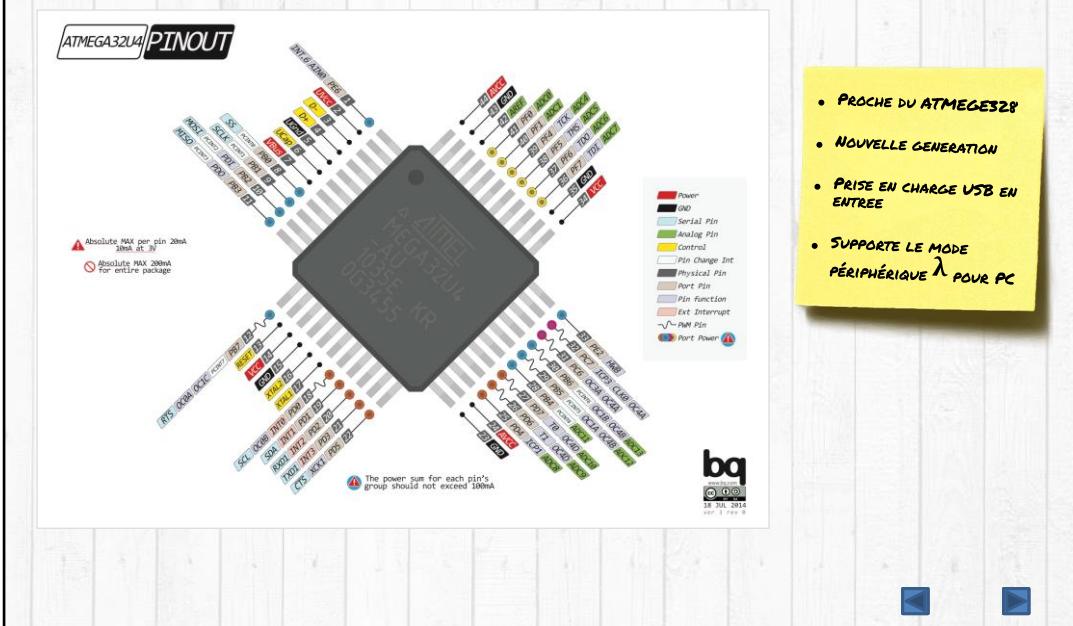
<https://www.microchipdirect.com/product/atmega2560>

architecture 32u4



http://ww1.microchip.com/downloads/en/devicedoc/atmel-7766-8-bit-avr-atmega16u4-32u4_datasheet.pdf

Atmega32u4 pinout TQFP

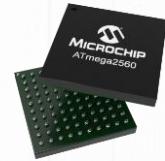


Processeurs Atmel atmega

- Processeur 8 bits

- 16 Mhz

- Atmega **1280 / 2560***
 - **flash** : **128 Kb ou 256 Kb**
 - **Format** : TQFP ou TFBGA
 - Write cycle :
 - » flash 10 000x
 - » EEPROM 100 000x
 - **EEPROM** 4Kb
 - **SRAM** 8Kb
 - **Hardware Interruptions** : 6
 - I/O :
 - » **54 numériques** dont 15(8/16bits) PWM
 - » **16 analogiques** 10 bits DAC
 - Protocoles :
 - » SPI, I²C,
 - » JTAG
 - » **4 UART**

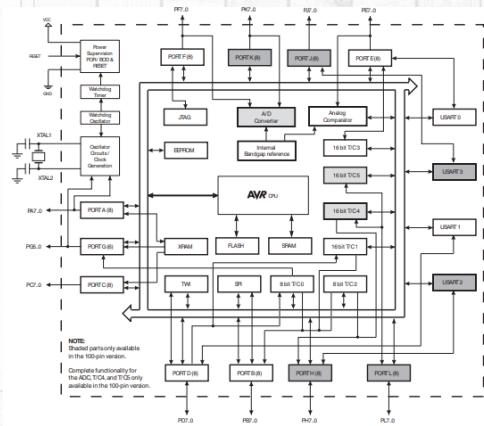


<https://www.microchipdirect.com/product/search/all/atmega328>

<https://www.microchipdirect.com/product/atmega2560>

Architecture 2560

- atmega 2560
 - Datasheet
 - [liens](#)
 - Plus complet que l'atmega 328
 - plusieurs USART serial matériel
 - toujours Watchdog avec timer
 - toujours un avr cpu



http://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf

Les kits Arduino

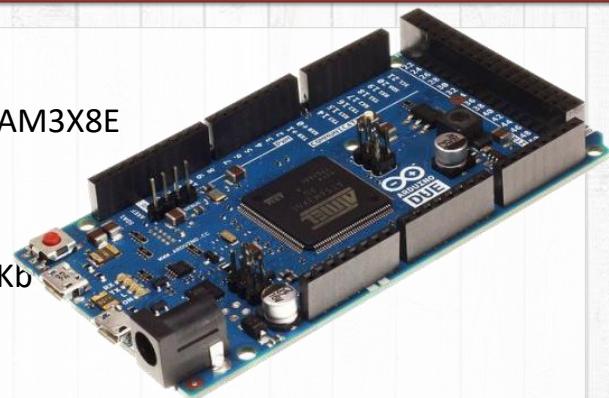
- Arduino Mega2560
 - atmega2560
- Arduino Uno & Uno R3
 - Uno atmega328 (QFP)
 - Uno R3 atmega328 (DIP)
- Arduino nano
 - atmega328 (QFP)
- Arduino mini
 - atmega328 (QFP)



Arduino & 32bits

- Arduino DUE

- processeur : AT91SAM3X8E
- **84** Mhz
- Flash : 2x256 Kb
- SRAM : 64 Kb + 32 Kb
- EEPROM
- 54 I/O numériques
- 12 analogiques 12 Bits DAC Input
- 2 analogiques Output
- USB On The Go



Les kits compatible arduino

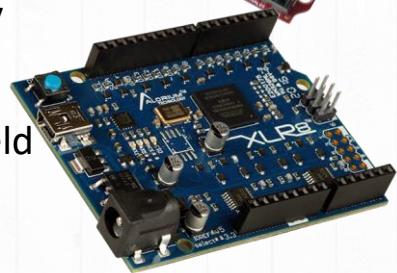
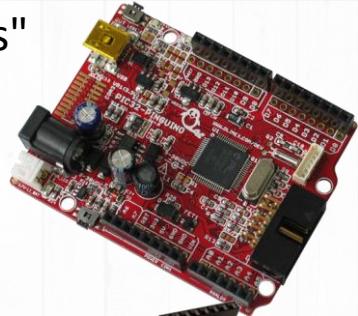
- Différentes cartes "compatibles"

- Formats

- Pinguino → PIC32MX400
 - Intel
 - ...

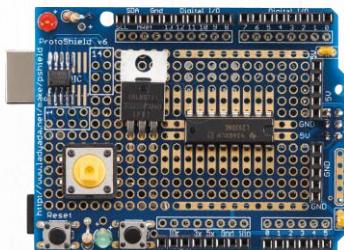
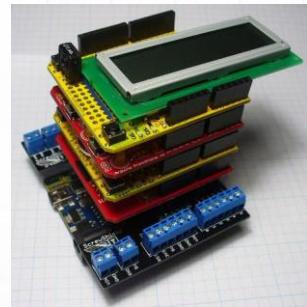
Attention aux voltages 5V vs 3.3V

compatible = enfichable sur shield



Arduino & shield

- Shield = bouclier
- Enfichable
 - multiples formats
 - nano
 - uno
 - mega
- Proto shield



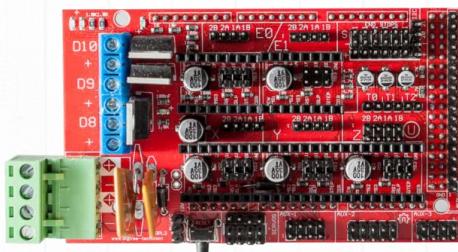
Les shields

- CNC Shield
- Wifi shield
- GPRS / LTE
- Screen LCD 16x2/20x4 Shield
- RJ45



Shield & formats

- certaines shields sont disponibles pour
 - Nano
 - Mega



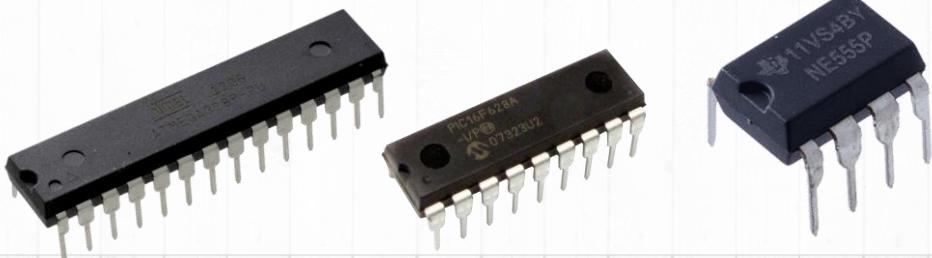
Les modules

- Les modules sont sur des pcb
 - Les composants nécessaires à la puce principale pour fonctionner(régulateur tension, résistances, ...)
 - La puce principale peut exister sans module
 - exemple :
 - bmp280
 - » composant intégrable
 - » module



Les circuits intégrés

- avec ou sans programmation
- le CI peut nécessiter un montage spécifique de composants
- Documentation nécessaire
 - ex LM393, NE555, atmel328,PIC,...



Conclusion

□ Dans ce chapitre, nous aborderons :

- ✓ L'open hardware
- ✓ Wiring et arduino
- ✓ Les cartes principales de développement pour ATMEGA
- ✓ Les pinouts des cartes
- ✓ Les compositions des processeurs ATMEGA
- ✓ Logique shield & modules



RAPPEL ELECTRONIQUES



Sommaire

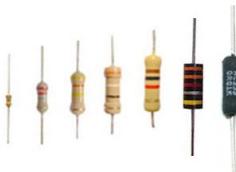
- ❑ Dans ce chapitre, nous aborderons :
 - ❑ Comment reconnaître les composants
 - ❑ Les lois basiques de l' électronique
 - ❑ Les I/O Tout ou Rien
 - ❑ Le PWM
 - ❑ Les protocoles UART, SPI I2C
 - ❑ Les 2 ports ICSP et les 2 ATMEGA présents sur les cartes



Reconnaitre les basiques

- Résistance

- Elle est exprimée en Ohm(s)

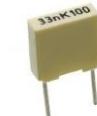


- Elles existent en CMS



- Condensateur

- Il est exprimé en décimal de farad (μf micro farad, pf pico farad...)



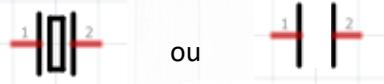
- Ils existent en CMS



Reconnaitre les basiques

- Quartz

- exprimé en Hertz(s) Hz



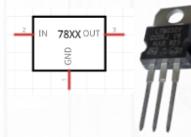
- inductance

- exprimée en Henri



- Circuit intégrés

- type gestion de puissance



- type générique

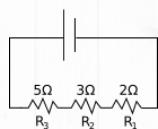


Les lois électroniques

- Remplacer un ensemble de résistance en une résistance unique équivalente
- 2 Montages série & parallèle

En Série

- La somme des résistances = Résistance équivalente
- $R_{eq} = R_1 + R_2 + R_3 \dots + R_n$

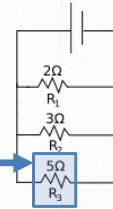


En parallèle

- L'inverse de la résistance équivalente = la somme de l'inverse des résistances

- Calculer les résistances en série en 1^{er}

$$5\Omega = 3\Omega + 2\Omega$$



$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} \dots + \frac{1}{R_n}$$



Pour les capacités la tendance s'inverse :

Serie : $\frac{1}{\text{CondensateurEq}} = \frac{1}{C_1} + \frac{1}{C_2} + \frac{1}{C_3} \dots + \frac{1}{C_n}$

Parallèle : $\text{CondensateurEq} = C_1 + C_2 + C_3 \dots + C_n$

Et pour les inductance :

Serie : $L_{eq} = L_1 + L_2 + L_3 \dots + L_n$

Parallèle : $\frac{1}{L_{eq}} = \frac{1}{L_1} + \frac{1}{L_2} + \frac{1}{L_3} \dots + \frac{1}{L_n}$

Les lois électroniques

Loi d'hom Ω

- Une formule qui fais tout

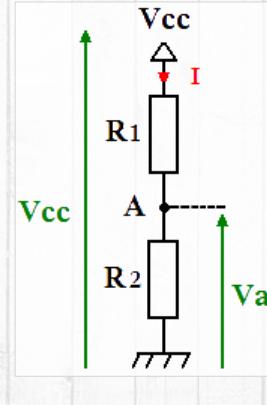
$$U_{(\text{tension } v)} = R_{(\text{résistance } \Omega)} \times I_{(\text{intensité } A)}$$

- Et qui se décline

- Pour l'intensité : $I = U/R$
- Pour la résistance : $R = U/I$

- Exemple d'application :

- $V_{CC} = 12V$
- $R_1 = 560\Omega$
- $R_2 = 560\Omega$

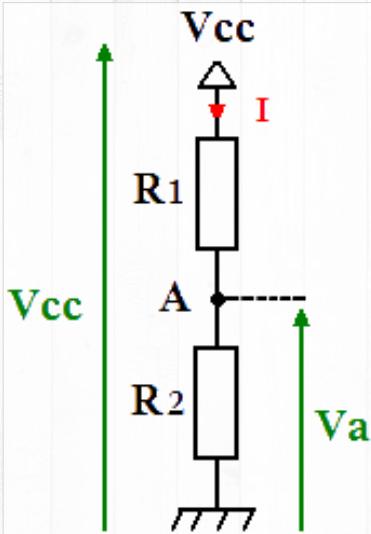


Kirchhoff a aussi mis en évidence le pont diviseur de tension souvent appelé pont de kirchhoff

loi de kirchhoff sur les ponts diviseur de tension :

https://fr.wikiversity.org/wiki/Loi_de_Kirchhoff/Pont_diviseur_de_tension

Les lois électroniques



Pont de kirchhoff

- énoncé en 1845 par le physicien Allemand Gustav Kirchhoff
- Permet d'obtenir des tensions inférieures
- R₁ permet l'écoulement de l'intensité
- Aussi appelé **pont diviseur de tension**



Kirchhoff a aussi mis en évidence le pont diviseur de tension souvent appelé pont de kirchhoff

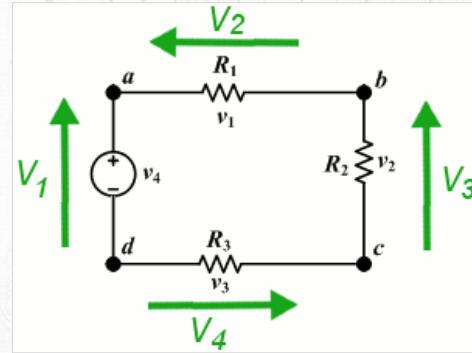
loi de kirchhoff sur les ponts diviseur de tension :

https://fr.wikiversity.org/wiki/Loi_de_Kirchhoff/Pont_diviseur_de_tension

Les lois électroniques

Loi des mailles

- énoncée en 1845 par le physicien Allemand Gustav **Kirchhoff**
- Tension = différence de potentiel électrique entre 2 points
- La somme des tensions est égale à 0V dans une maille
- $V_1 - V_2 - V_3 - v_4 = 0$

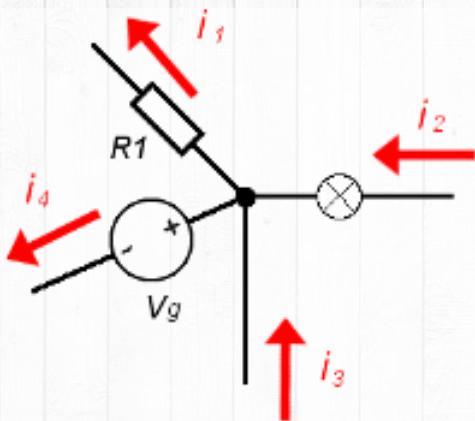


Kirchhoff a aussi mis en évidence le pont diviseur de tension souvent appelé pont de kirchhoff

loi de kirchhoff sur les ponts diviseur de tension :

https://fr.wikiversity.org/wiki/Loi_de_Kirchhoff/Pont_diviseur_de_tension

Les lois électroniques



Loi des nœuds

- énoncée en 1845 par le physicien Allemand Gustav Kirchhoff
- Somme des intensités arrivant à un point = Somme des intensités sortant d'un point
- $i_2 + i_3 = i_1 + i_4$

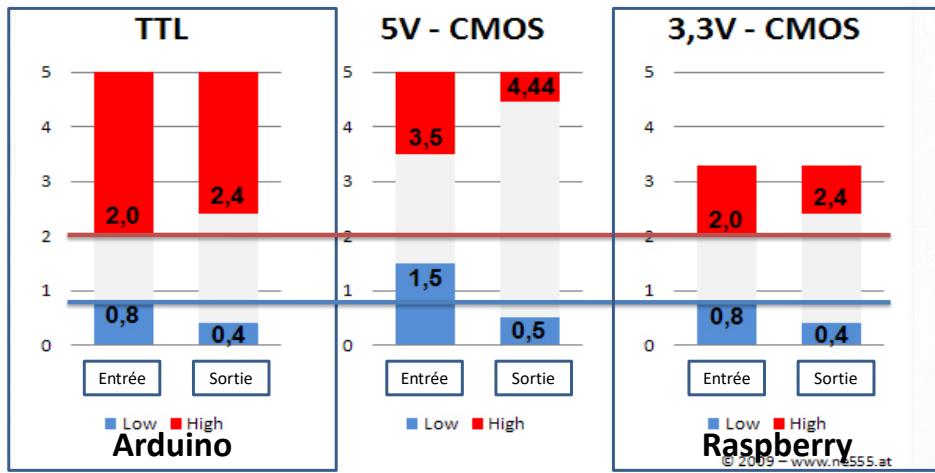
Kirchhoff a aussi mis en évidence le pont diviseur de tension souvent appelé pont de kirchhoff

loi de kirchhoff sur les ponts diviseur de tension :

https://fr.wikiversity.org/wiki/Loi_de_Kirchhoff/Pont_diviseur_de_tension

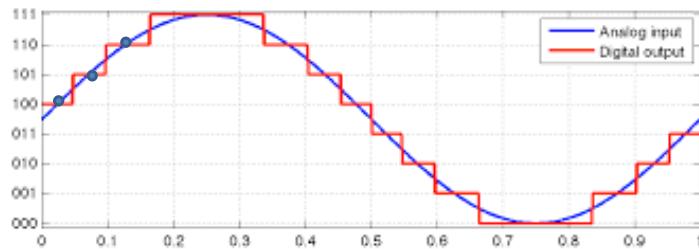
Niveau logiques

- Composants à niveaux logique TOR (Tout Ou Rien)
 - différentes normes



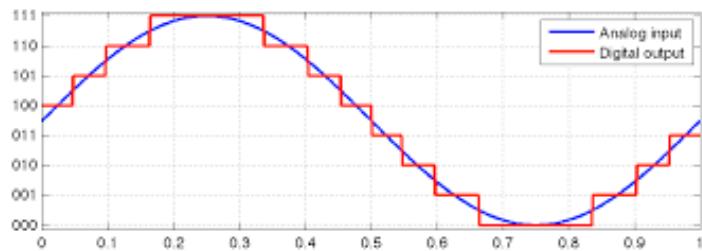
CAN ou NAC

- Convertisseur Analogique vers Numérique
 - Interprétation numérique d'une valeur analogique
 - Echantillonnage à une fréquence donnée 16MHz
max(arduino)



DAC

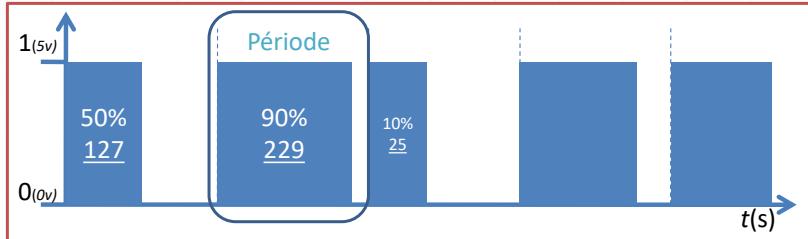
- Convertisseur **Numérique vers Analogique**
 - Définit un niveau de tension (volt) constant pendant un court lapse de temps
 - Echantillonnage à une fréquence donnée
 - Conversion avec pertes *effet palier* **ligne rouge**



PWM

- **Pulse Width Management**

- valeur sur 8bits → de 0 à 255
- Remplissage de cycles (Périodes)



- Période $t(s) = 1/f$ fréquence f (périodes par secondes)

- ex :
 - $f = 16\text{Mhz} = 16\,000\text{ Hz}$
 - $t = 1 / 16\,000 = 0,0000625\text{ s} = 0,0625\text{ ms} = 62,5\mu\text{s}$

- Sur arduino les sorties PWM sont indiquées par : ~



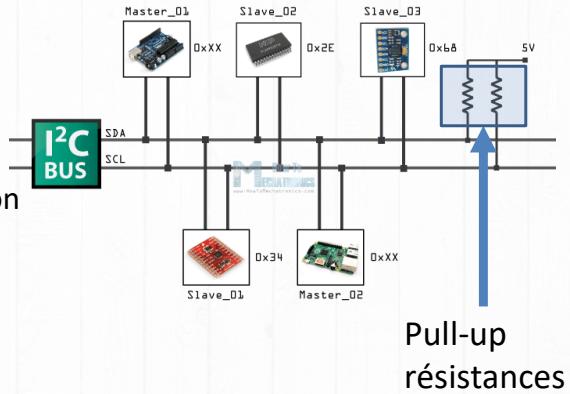
UART

- Liaison Full duplex (entrée /sortie simultanées)
- Liaison série **RS232** utilisée sur les **ports séries des ordinateurs** (prise DB9)
- 2 ports de données
 - Tx (Transmit)
 - Rx (Receive)
- **Masse commune**
- Tension de fonctionnement sur les ports sur les mêmes plages TOR



Protocole I²C

- Inter-Integrated Circuit (aussi TWI ou TWSI)
 - Développé par Philips en 1982 et maintenu par NXP
 - 6eme édition en 2014
 - Bus série synchrone bidirectionnel **half-duplex** (entrée OU sortie à la fois)
 - Sélection par adresses
 - 1 port de données
 - SDA (Serial Data Line)
 - 1 port de synchronisation
 - SCL (Serial Clock Line)



Protocole S.P.I.

- **Serial Peripheral Interface**

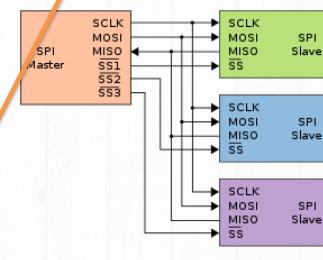
- Développé par Motorola
- **Bus série synchrone full duplex**
- Master / Slave

- 1 port de sélection périphérique
 - CS (**Cable Select**) ou SS (Slave Selector)

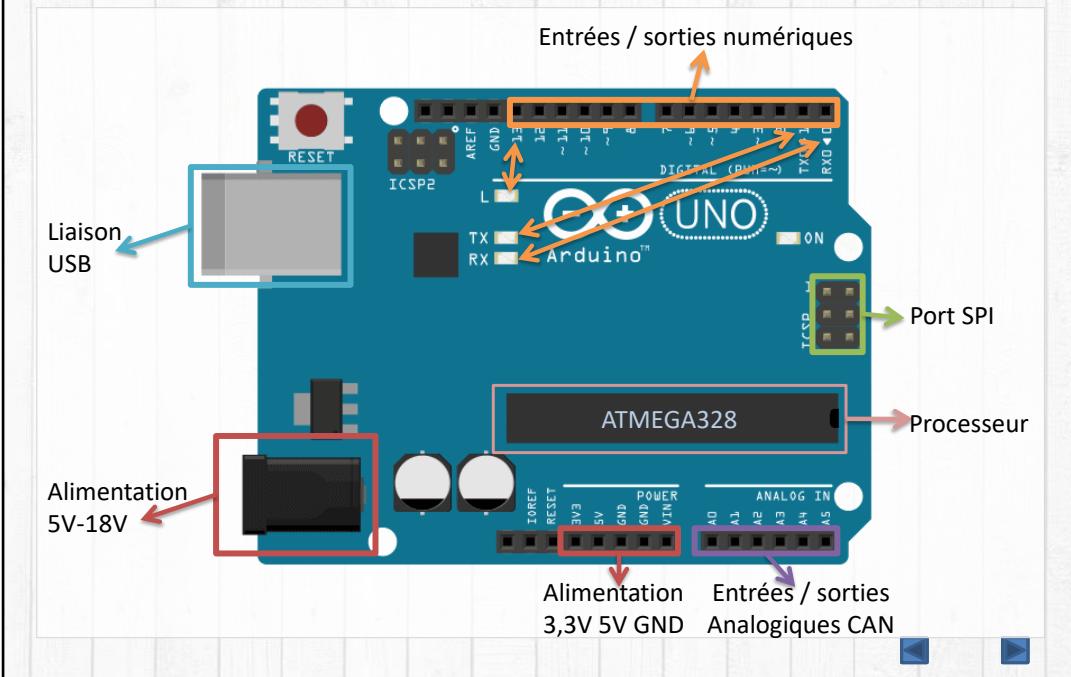
- 2 ports de données
 - MOSI (**Master Out Slave In**) ou SDA, SDO, DO, SO
 - MISO (**Master In Slave Out**) ou SDI, DI, SI
- 1 port de synchronisation d'horloge
 - CLK ou SCLK

Valeur 1 pour activer un périphérique Unique pour chaque périphérique

Ports communs à tous les périphériques



Arduino pinout

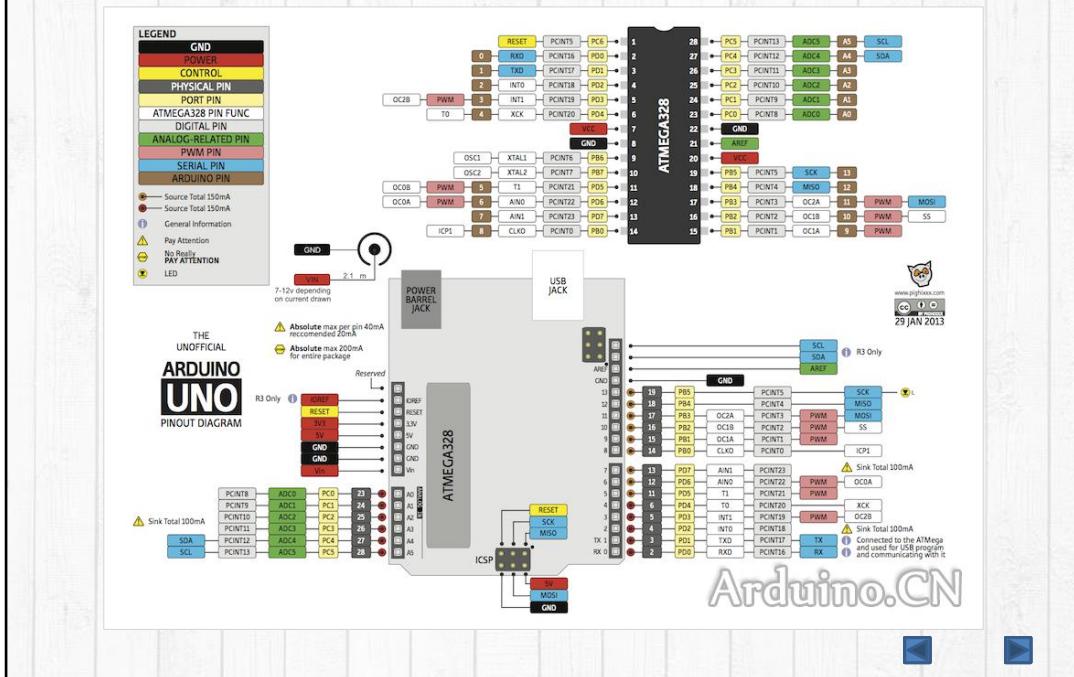


L'ensemble des ports Analogiques possèdent des *DEFINE* pour faciliter leur nommage lors du développement .

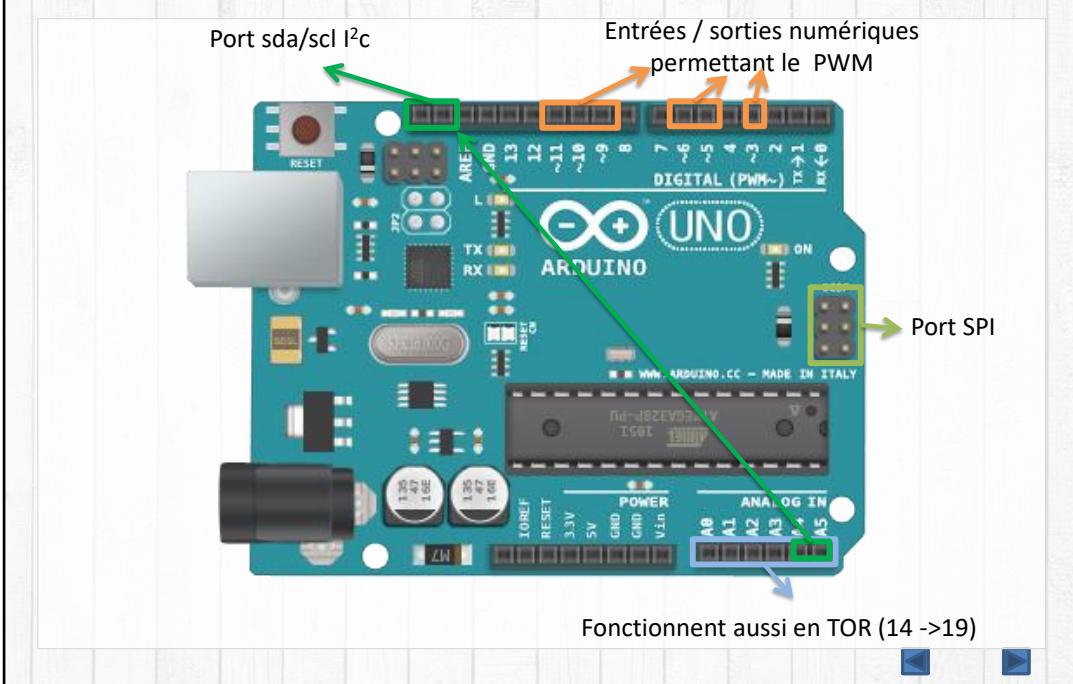
Ces define sont implicitement inclus lors de la compilation et sélectionnés en fonction du choix de la carte effectué pour la compilation du binaire à téléverser

Les IO analogique se nomment : A0, A1, A2, A5, Ax

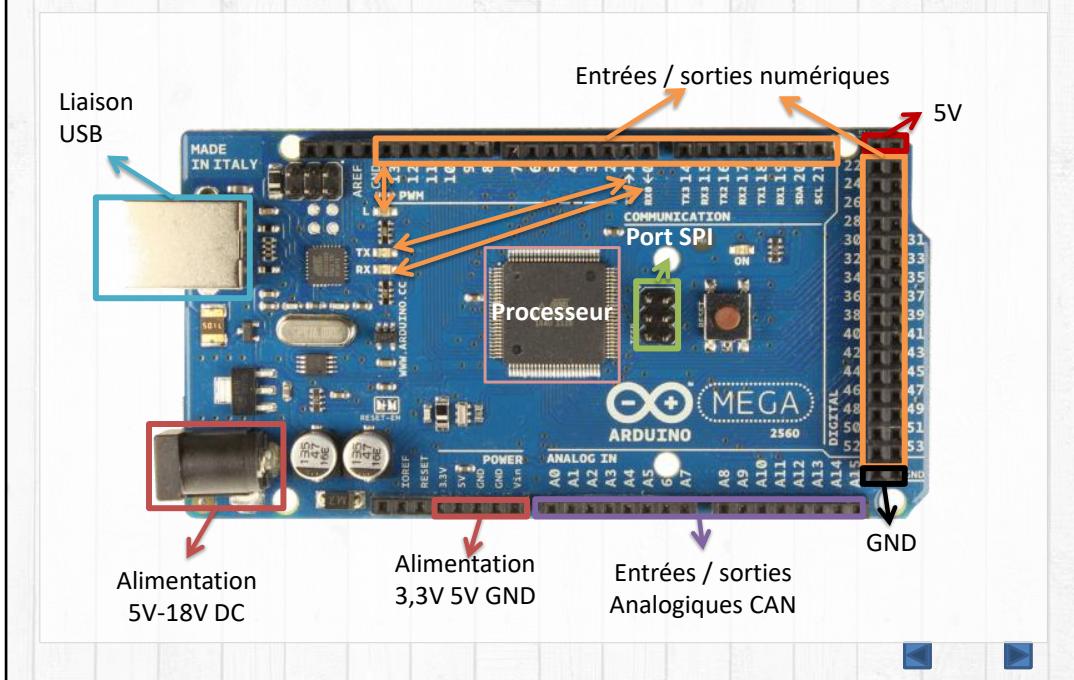
Détails techniques



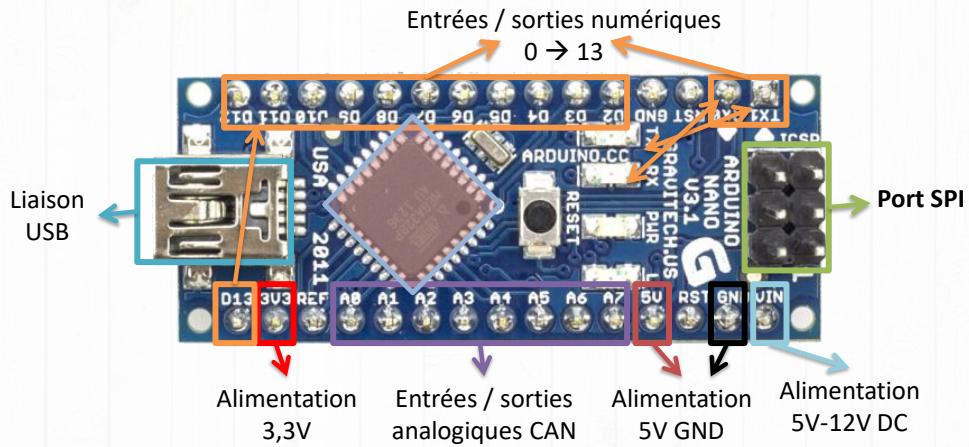
Arduino UNO pinout



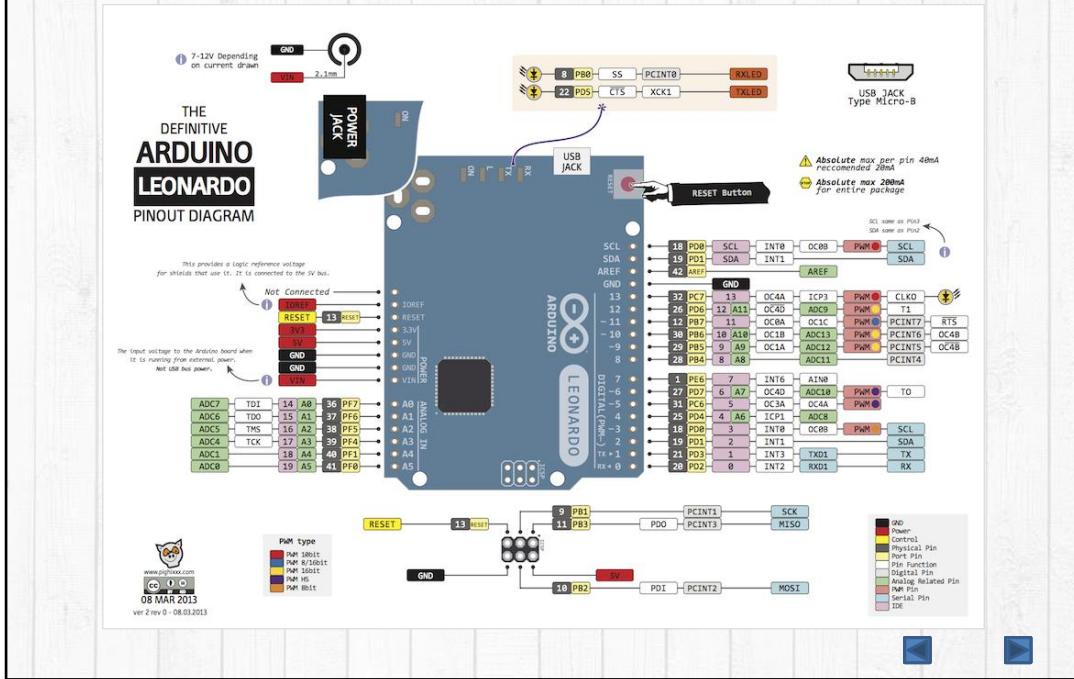
Arduino Mega2560 pinout



Arduino Nano pinout

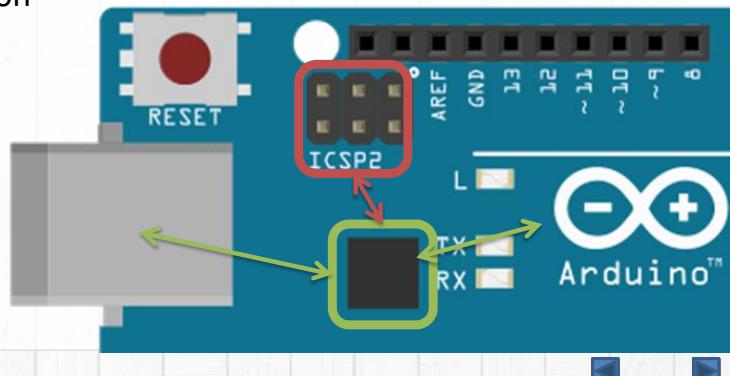


Arduino Leonardo et 32u4



arduino & ICSP2

- Rôle du Mega8
 - liaison usb ↔ serial
 - souvent remplacé par CH340* (* LOW COST)
- Rôle du 2eme port isp
 - Programmation du mega8
 - Pas dispo sur ch340 ou eq.

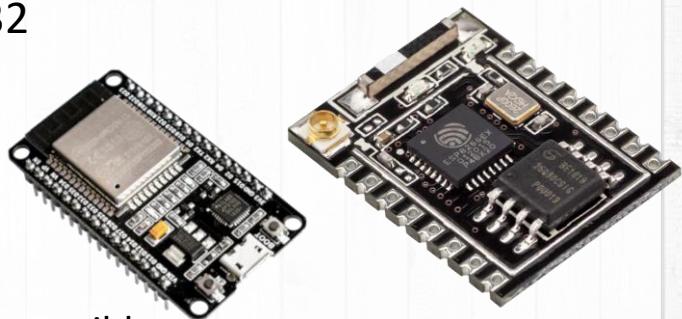


Alternatives

- ESP8266 / ESP32

- Jusqu'à 8 Mo

- espFS



- Arduino IDE compatible

- Différents formats



Alternative

- OLIMEX

- PIC32MX220F032D
 - PIC32 PINGUINO MX220
- PIC32MX440 256H
 - PIC32 PINGUINO MICRO
 - PIC32 PIGUINO

	PIC32-PINGUINO-MX220	ARDUINO-UNO
CPU	32 bit	8 bit
CLOCK	40 MHz	20 MHz
FLASH	32KB	32 KB
RAM	8KB	1 KB
USB	YES	NO*
LOW PWR	YES**	NO
GPIO MAX	40 MHz	5 MHz
ADC	1.1MSPS	15 KSPS
DMA	YES 4 CH	NO
UEXT	YES	NO

* require additional chip FT232 or ATMega with USB

** hardware allow power down to 10 uA operation for handheld battery apps



PIC32 pinguino : <https://www.olimex.com/Products/Duino/PIC32/PIC32-PINGUINO/open-source-hardware>

PIC32 pinguino MX220 : <https://www.olimex.com/Products/Duino/PIC32/PIC32-PINGUINO-MX220/open-source-hardware>

Pinguino sur arduino ide

:https://github.com/OLIMEX/Arduino_configurations/tree/master/PIC

❑ Dans ce chapitre, nous avons aborder :

- ✓ Comment reconnaître les composants
- ✓ Les lois basiques de l' électronique
- ✓ Les I/O Tout ou Rien
- ✓ Le PWM
- ✓ Les protocoles UART, SPI I2C
- ✓ Les 2 ports ICSP et les 2 ATMEGA présents sur les cartes



I/O I/O EN RENTRANT DU BOULOT...

ENTRÉES / SORTIES



Sommaire

❑ Les entrées / sorties

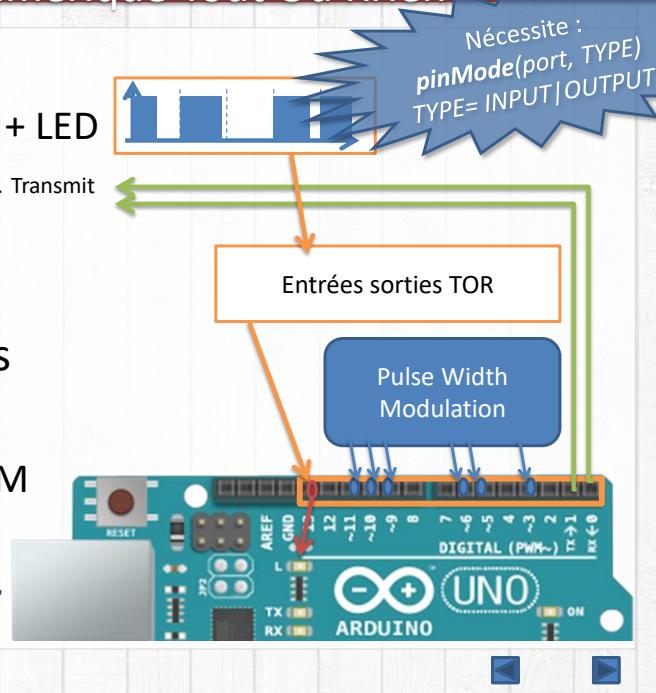
- ❑ Ports numériques
- ❑ Port analogique (CAN)
- ❑ Etats TOR
- ❑ PWM



Arduino UNO

Entrée numérique Tout Ou Rien

- Ports pré câblés
 - Hardware Serial + LED
 - ports 0 Recieve / 1 Transmit
 - Led
 - port 13
- Différents modes
 - HIGH ou LOW
 - Modulation PWM native
 - symbolisé par ~

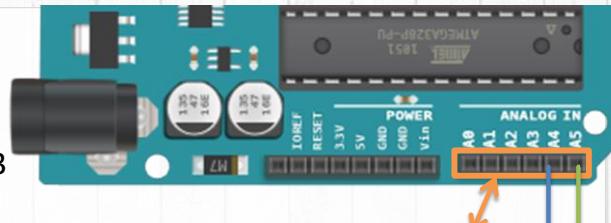


Arduino UNO Entrée analogique CAN

- **CAN 10bits**

- Plage

- $0000000000 = 0$
 - $1111111111 = 1023$



- pas :

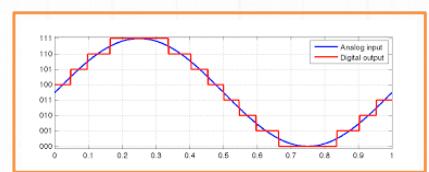
- **5v max / 1024 possibilités**
 - 0,0048828125V

- **Ports pré câblés**

- Analogique **ou** Hardware I²C

- ports
 - A4 sda
 - A5 clk

Entrées sorties CAN



sda

CLK



Conclusion

❑ Les entrées / sorties

- ✓ Ports numériques
- ✓ Port analogique (CAN)
- ✓ Etats TOR
- ✓ PWM



DÉVELOPPEMENT & IDE



Sommaire

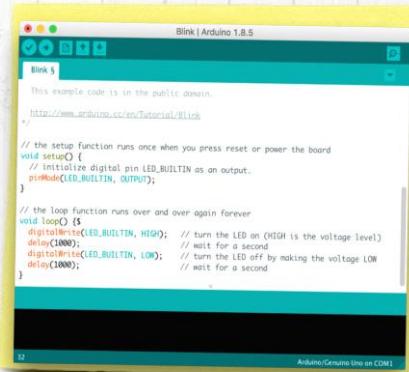
❑ Les ide

- ❑ Arduino IDE
- ❑ Code::blocks
- ❑ Scratch
- ❑ Visual studio
- ❑ Atmel studio



Arduino IDE

- Arduino IDE
 - Fournit par arduino.cc
 - Fichiers .INO
 - Envoie de bootloader
 - Compilation
 - envoie carte
 - compatible esp8266, ...



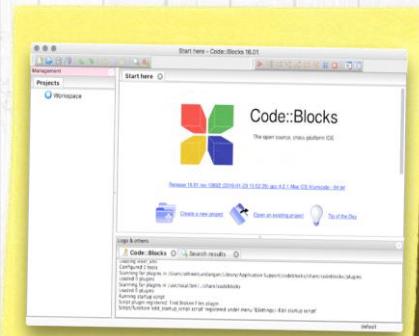
The screenshot shows the Arduino IDE interface with the title bar "Blink | Arduino 1.8.5". The code editor displays the "Blink" sketch, which blinks an LED connected to pin 13. The code is as follows:

```
This example code is in the public domain.  
http://www.arduino.cc/en/Tutorial/Blink  
  
// the setup function runs once when you press reset or power the board  
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  
}
```



Code::Blocks

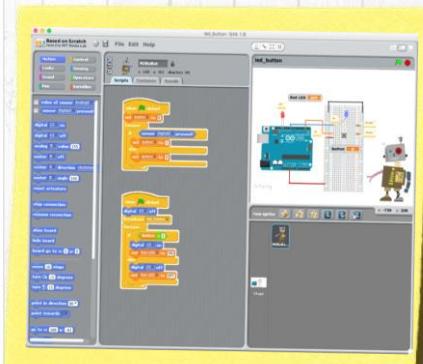
- IDE C/C++
 - Coloration syntaxique
 - Plugin arduino
 - standalone
 - dépendances nécessaires
 - » arduino.h
 - » avr.h
 - Compilation arduino
 - Simulateur Arduino*
 - Envoie carte arduino



*simule que les sorties Serial.print , analogWrite, ..., **pas les entrées** analogRead, Serial.readLine, ...

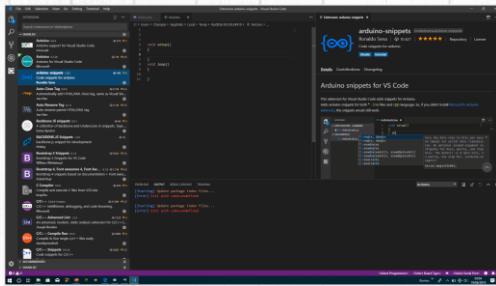
scratch

- Scratch IDE
 - Développement simplifié
 - à base de brique à assembler
 - Développement RAD
 - Génération du code arduino
 - composants limités



VS CODE

- Multiplateforme
- Edité par MS
- Prise en charge du C/C++
- Snippets C / C++ / arduino
- Accès library
 - Nécessite Arduino d'installé
- Gratuit



projets virtuels

- Fritzing propose des breadboard virtuels

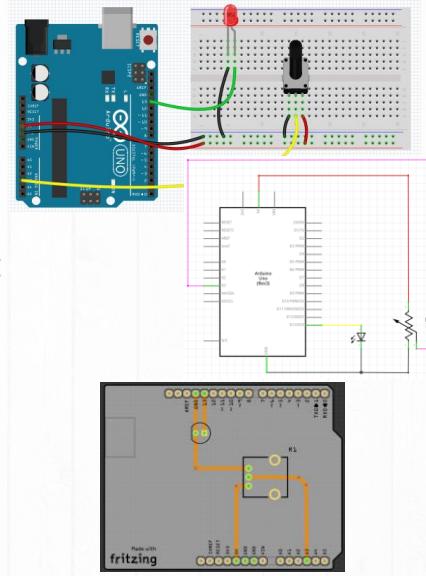
- Création schématique

- Ecoute de ports COM

- compilation & téléversement sur carte arduino
 - dépendent de arduino IDE

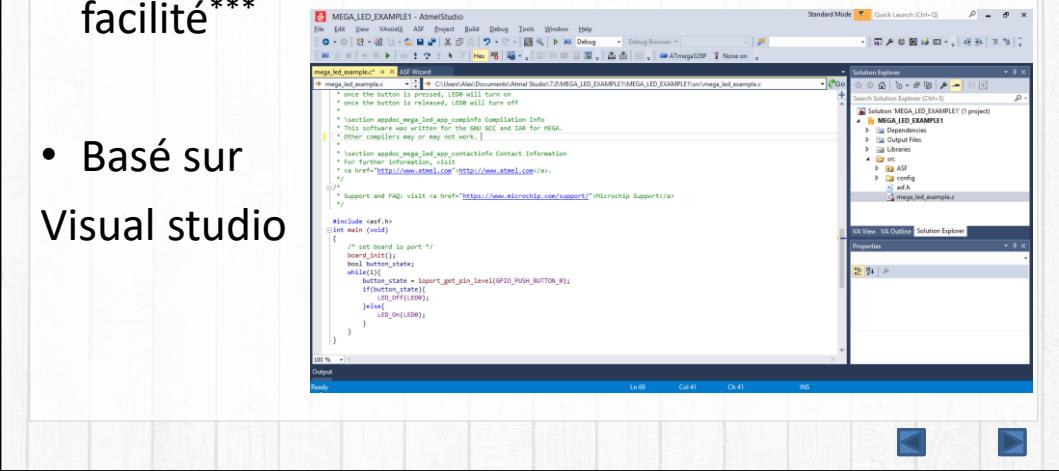
- Design de typons

- génération des fichiers *gerber*
 - service d'impression PCB



avr studio / atmel studio

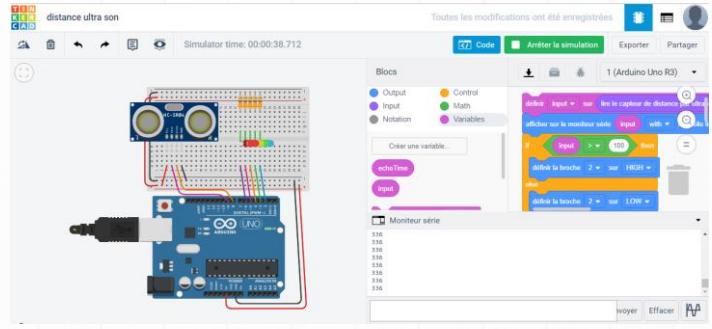
- Prend en charge toute une gamme de composants
- permet le debug & l'édition assembleur est facilité***
- Basé sur Visual studio



ThinkerCAD

- Simulation de circuits électroniques
 - gratuit
 - **simulation arduino** et certains modules
 - Ecriture du code en C/C++ arduino ou scratch

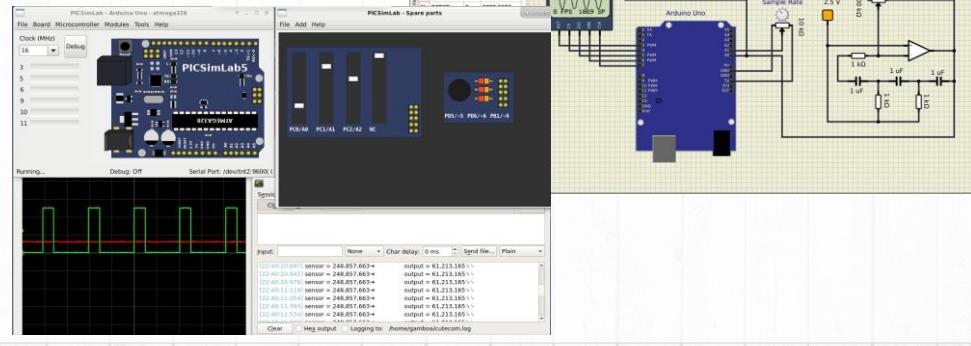
Espace personnel
avec sauvegarde &
simu des projets



Simulateur AVR

- Chargement de elf dans des puces virtuelles

- Simutron
- PicSim
- SimulIDE



Simutron : <https://sourceforge.net/projects/simutron/>

Picsim : <https://sourceforge.net/projects/picsim/>

Simulide: <https://sourceforge.net/projects/simulide/>

Visual studio & visual micro

- Microsoft Visual studio
- Plugin de visual studio
 - assez proche de avr studio mais en moins complet
 - prise en charge des compilations avr
 - prise en charge des téléchargements



Conclusion

❑ Les ide

- ✓ Arduino IDE
- ✓ Code::blocks
- ✓ Scratch
- ✓ Visual studio
- ✓ Atmel studio
- ✓ thinkerCAD



ÉCRIRE DU CODE ARDUINO & BASES DU LANGAGE



Sommaire

❑ Historique du C/C++

❑ En C

- ❑ La syntaxe,
- ❑ Les variables,
- ❑ les tableaux
- ❑ Les chaines,
- ❑ Les fonctions,
- ❑ Les pointeurs
- ❑ La doc



C l'histoire

- D'après wiki :
 - La C est un langage de programmation impératif généraliste, de bas niveau. Inventé au début des années 1970 pour réécrire UNIX
- Particularités
 - Procédurale
 - Pas d'objet, mais des structures
 - Pas de string mais des char* ou char[]
- Compilé
 - Gcc le plus souvent



C++ l'histoire

- C++ est un langage de programmation compilé permettant la programmation sous de multiples paradigmes
 - Objets, procédurale, générique,...
 - 1ere version en **1983**
 - **Syntaxe très proche du C**
 - Plus de concepts Complexes

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```



Analysons la syntaxe du C (I)

- Déclaration de fonctions

- TypeDeRetour nomDAppel (typeParam nomParam , typeParam2 nomParam2, ...)
- { } pour délimiter le contenu d'instruction de la fonction
- Valeur de retour si non **void**
- Appel d'une fonction déjà déclarer
 - Finir toutes les instructions par ;

```
#INCLUDE "MONHEADER.H"
DOUBLE MAFUNCTION (VOID)
{
    DOUBLE RETOUR = 0;

    CHAR CHAINE[32] = "HELLO";

    FLOAT ENTIERS[2] = [8.2F,0F];

    APPELFONCTION(PARAM1,PARAM2);
    SANSPARAM();

    RETURN RETOUR;
}
```

Fonctions & prototypes

- Une fonction doit être déjà connu pour être utilisée
 - Les prototypes décrivent le nom et le comportement (entrée & sortie) des fonctions pour soumettre l'existence de la déclaration principale
 - Soit dans le fichier C limite la capacité d'inclusion
 - Soit dans le fichier H pour diffuser l'existence hors du fichier C

• FICHIER .H

```
bool isNum4(int);
```

• FICHIER .C

```
#include "fichier.h"

void add(int,int,int*);
```

```
void main (void)
{
    add(1,2,p_resMain);
}
```

```
void add(int a,int b,
          int* res)
{
    isNum4(a);
    *res = a + b;
}
```

```
bool isNum4(int n)
{return n==4?true:false;}
```

Analysons la syntaxe du C (II)

- Déclaration de variables

- Type nomVariable ;
- Type nomVariable = value ;
 - Finir toutes les instructions par ;
- type en minuscule pour les primitifs
- Type Commence par majuscule en CamelCase pour les autres

- Types de variables

- int , unsigned int et uint8_t
- long, unsigned long, uint16_t
- float, unsigned float
- double
- struct votreStructure
- char, byte, ...

```
INT MAFUNCTION (VOID)
{
    INT NOMVARIABLE = 0 ;
    CHAR CHAINE[6] = "HELLO" ;
    FLOAT ENTIERS[2] = {8.2F,0F} ;

    APPELFONCTION(PARAM1,PARAM2) ;
    SANSPARAM() ;

    RETURN EXIT_SUCCESS;
}
```

Déclaration Globale vs Locale

• Globale

- La portée est pour toute l'exécution du programme et modifiable de tous
- Durée de vie constante

• Locale

- Seul le bloc dans lequel il est déclarer peut accéder a la variable
- La variable sera détruite en fin de vie du bloc

```
FLOAT ENTIERS[2] = {8.2F,0F};
```

```
INT MAFUNCTION (VOID)  
{
```

```
    INT NOMVARIABLE = 0;
```

```
    CHAR CHAINE[6] = "HELLO";
```

```
    FOR(INT I=0; I<6; I++) { .. }
```

```
    RETURN EXIT_SUCCESS;  
}
```



Les opérateurs

- Arithmétique

- +, - addition, soustraction
- ++, -- incrémentation, décrémentation
- *, / multiplication, division
- % modulo (reste de division euclidienne)

- Logique et comparaison

- ||, && ou, et
- ==, != Strictement égale ou strictement différent

- Affectation

- = affectation de valeur ou d'adresse
- =+, =-, =*, =/ affecte la destination opérer avec l'ensemble de droite



Les opérateurs de bits

- Les décalage de bits
 - Décale les états représentant la valeur
 - Décalage gauche : <<
 - Décalage droite : >>

byte a=3	0	0	1	1
a<<2	1	1	0	0
Shift left (2bits)				
byte a=12	1	1	0	0
a>>2	0	0	1	1
Shift right (2bits)				

	1	0	1	0
&(ET)	1	1	0	0
	1	0	0	0

- Les opérations booléen

- & (et) : les états doivent dans les deux termes être vrai (1) pour être maintenu à 1 en sortie
- | (ou) : au moins un des deux états doivent être vrai (1) pour être maintenu à 1 en sortie

	1	0	1	0
(OU)	1	1	0	0
	1	1	1	0



Les tableaux

- Déclaration

- Type nomVar [nbDElements];

- Le type peut être : int, float, struct maStruct, ...

- Usage

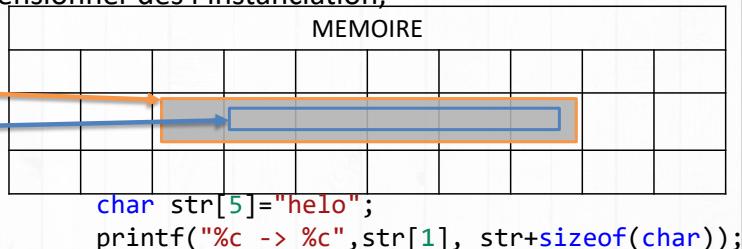
- maVariable[0]=2;

- Adresses consécutives

- Ne se redimensionne pas

- Bien dimensionner des l'instanciation;

```
int str[5];  
str+1;
```



Les chaines de char

- char est un caractère mais
- char[] un tableau de caractères
 - Prévoir un espace de plus pour le caractère de fin de chaîne '\0'
 - Se déclare de la manière suivante
`char maChaine[19]="une chaîne de char";`
- Les fonctions de chaînes
 - `strlen` taille de la chaîne jusqu'à '\0'
 - `strcat` et `strncat` concat de chaînes
 - `atof`, `atoi`, `atol` conversion alpha to float, int, long,
 - <https://devdocs.io/c-strings/>, ...



Les pointeurs

- Une fonction peut seulement éditer un paramètre d'entrée pour elle-même
- Un pointeur peut modifier une valeur pointé
- Pointer = 1 Octet d'adresse mémoire
- Un pointeur possède un type pour connaître l'organisation et la taille de l'instance pointé

Traductions de pointeur	
Nom	Fréquence ⓘ
l'aiguille	needle, hand, pointer, indicator, spire
l'index	index, index finger, forefinger, pointer, first finger, indicator
la baguette	rod, stick, baton, molding, pointer, ramrod
le conseil	board, council, counsel, tip, guidance, pointer
le tuyau	pipe, hose, tip, conduit, stem, pointer
la flèche lumineuse	pointer, streamer
le chien d'arrêt	retriever, pointer

La variable
instancié

Mémoire

Le pointeur *

Source translate.google.fr

Pourquoi les pointeurs ?

- La fonction crée une copie local de la variable en paramètre
- Le pointeur modifie7 le contenu d'un espace mémoire sans passer par la variable qui la crée

```
int main (void)
{
    int resMain=0;
    int retour=add(1,2,resMain);
    printf("%d -> %d",resMain,retour);
}

int add(int a, int b, int res){
    res = a + b;
    return res;
}

Sortie : 0 -> 3

int main (void)
{
    int resMain=0;
    int *p_resMain=&resMain;
    int retour=add(1,2,p_resMain);
    printf("%d",resMain);
}

void add(int a, int b, int* res){
    *res = a + b;
}

Sortie : 3
```



Utiliser les pointeurs

- Un jeu d'étoiles
 - & représente l'adresse d'une instance
 - * pour la valeur pointée par un pointeur
- Oubliez pas de prévoir les espaces dans la mémoire si pas affecté à une variable instancié
 - `int* ptr=(int *) calloc (1,sizeof(int));`

Déclaration	Valeur	adresse
<code>int variable;</code>	<code>variable</code>	<code>&variable</code>
<code>int* ptr=&variable;</code>	<code>*ptr</code>	<code>ptr</code>
<code>int** ptr_ptr=&ptr;</code>	<code>**ptr_ptr</code>	<code>*ptr</code>
<code>int*** ptr_ptr_ptr=&ptr_ptr;</code>	<code>***ptr_ptr_ptr</code>	<code>**ptr_ptr_ptr</code>



Conclusion

- ✓ Historique du C/C++
- ✓ En C
- ✓ La syntaxe,
- ✓ Les variables,
- ✓ les tableaux
- ✓ Les chaines,
- ✓ Les fonctions,
- ✓ Les pointeurs
- ✓ La doc



CODER POUR LES PUCES



ARDUINO ET LE C/C++



Sommaire

❑ Dans cette partie nous allons aborder :

- ❑ Structure ino
- ❑ Entrée sortie analogique
- ❑ Entrée sortie numérique
- ❑ Communication série
- ❑ Inclusion de librairie



Fichier INO minimum

- 2 sections obligatoires
 - C/C++ légères différences avec les standards
 - Méthode d'inclusion
 - setup
 - Fonction exécuter automatiquement au début de l'exécution du programme
 - loop
 - Fonction exécuter en boucle tout au long de l'exécution du programme

```
/**Fichier.ino**/
```

```
void setup()
{
//setup
}

void loop()
{
//loop
}
```



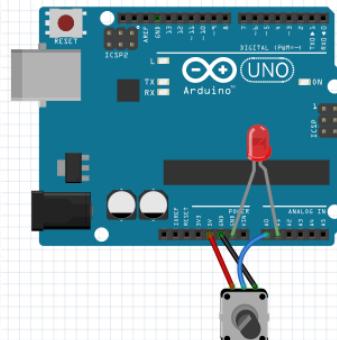
Lecture / Ecriture Analogique

- Lecture Ecriture des ports analogiques par conversion CAN
 - Sur 10bits soit de 0 à 1023
- Lecture
 - `analogRead(PinReference)`
 - Renvoie la valeur interpréter par le CAN
- Ecriture
 - `analogWrite(PinReference, value)`
 - Envoie la tension interpréter par le CAN avec la valeur de value
- Les **pin** sont identifiés **nativement** par des **constantes**
 - A1
 - A2
 - A...



TP "illumination progressive"

- Brancher une LED et un potentiomètre sur un port analogique
- Lire la valeur du potentiomètre et illuminer la LED en fonction du potentiomètre



Lecture / Ecriture Numérique / binaire / PWM

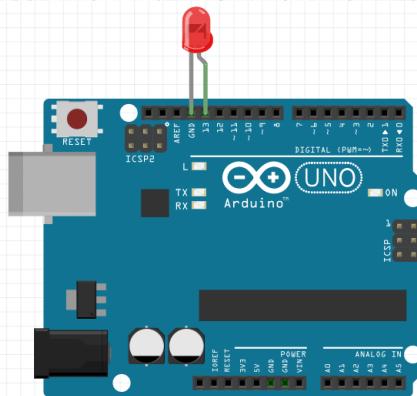
- Lecture Ecriture des ports numériques
 - Envoie d'états sur un port
 - true → HIGH
 - 5v
 - false → LOW
 - 0v
 - Valeur PWM entre 0 et 255
 - Les entrées et sortis doivent être définis pour chaque usage
 - pinMode(PinReference, IOType)
 - set le pin sur le mode d'entrée ou de sortie suivant la valeur IOType
 - » OUTPUT
 - » INPUT
 - Lecture
 - digitalRead(PinReference)
 - Renvoie la valeur PWM présente sur le pin PinReference
 - Ecriture
 - digitalWrite(PinReference, value)
 - Envoie le remplissage de cycle correspondant à la valeur PWM ou binaire

pinMode set des
résistances
internes sur 5v
ou 0v



TP "Blink"

- Brancher une LED sur le port 13
- Niveau I :Faites la s'allumé, puis s'éteindre à intervalle régulier.
- Niveau II :Faites la scintiller (PWM)



Fonction Serial

- Lot de fonction pour communiquer sur le port série matériel présent dans l'objet `Serial`
 - Démarrer le port série
 - `Serial.begin(speed [,config])`
 - Ferme le port série
 - `Serial.end()`
 - nb de bytes disponible
 - `Serial.available()`
 - Renvoi le nb de bytes disponible en lecture



Dans le cas de certaines cartes Serial1, Serial2, Serial2, Serial3, Serial4 sont disponible pour les ports serie materiel

Fonction écriture Série

- Ecriture sur le port série

- Ecrire des caractères

- `Serial.print(charString)`
 - `Serial.println(charString)`

- Ecrire des bytes

- `Serial.write(byteArray)`



Dans le cas de certaines cartes Serial1, Serial2, Serial2, Serial3, Serial4 sont disponible pour les ports serie materiel

Fonction lecture Série

- Lecture sur le port série
 - attente jusqu'à présence d'une chaîne
 - `Serial.find(charString)`
 - Lecture byte à byte
 - `Serial.peek(charString)`
 - `Serial.read()`
 - renvoie un byte du buffer série
 - Lecture chaîne
 - `Serial.readBytes(charBuffer, length)`
 - renvoie le nb de bytes placé dans `charBuffer` dans la limite de `length` & `SerialTimeout`
 - `Serial.readString()`
 - renvoie un string lu dans la limite de `SerialTimeout`
 - Interruption à réception série
 - Déclarer une fonction `void serialEvent ()`
 - Définition de timeout
 - `Serial.setTimeout (tempsEnMillis)`



Dans le cas de certaines cartes Serial1, Serial2, Serial2, Serial3, Serial4 sont disponibles pour les ports série matériel

Gestion du temps

- Faire une attente sur le processus en cours
 - en millisecondes
 - `delay();`
 - En microsecondes
 - `delayMicroseconds();`
- Connaitre le temps machine depuis le début de l'exécution
 - en millisecondes
 - `millis();`
 - en microsecondes
 - `micros();`



Fonctions trigonométriques & random

- Cosinus
 - `cos(radAngle);`
- Sinus
 - `sin(radAngle);`
- Tangente
 - `tan(radAngle);`
- Valeurs aléatoires
 - `randomSeed()`
 - set la graine pour les random
 - `random([min,]max)`
 - Renvoie une valeur aléatoire



Conclusion

❑ Dans cette partie nous avons aborder :

- ✓ Structure ino
- ✓ Entrée sortie analogique
- ✓ Entrée sortie numérique
- ✓ Communication série
- ✓ Inclusion de librairie



STARTER PACK



Sommaire

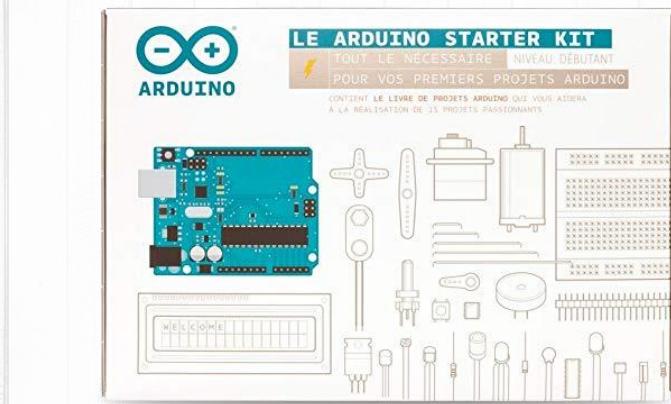
❑ Dans cette partie nous allons aborder :

 ❑ Contenu du starter Kit

 ❑ Conceptions des projets du kit



The Starter Kit

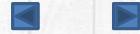


[https://www.elecrow.com/download/Starter%20Kit%20for%20Arduino\(user%20manual\).pdf](https://www.elecrow.com/download/Starter%20Kit%20for%20Arduino(user%20manual).pdf)

Arduino starter kit

Contenu du kit officiel

- | | |
|--|--|
| <ul style="list-style-type: none">• Arduino livre(170 pages)• 1 Arduino UNO rev.3 1• cable USB• 1 Breadboard• 1 base• 1 9v bat.• 70 Solid core jumper wires• 2 Stranded jumper wires• 6 Photoresistor [VT90N2 LDR]• 3 Potentiometer 10kilohm• 10 Pushbuttons• 1 Temperature sensor [TMP36]• 1 Tilt sensor• 1 LCD alphanumeric (16x2 characters)• LED (bright white)• 1 LED (RGB – common cathode)• 8 LEDs (red) 8 LEDs (green) 8 LEDs (yellow) 3 LEDs (blue)• | <ul style="list-style-type: none">• 1 Small DC motor 6/9V• 1 Small servo motor• 1 Piezo capsule [PKM17EPP-4001-B0]• 1 H-bridge motor driver [L293D]• 2 Optocouplers [4N35]• 5 Transistor [BC547]• 2 Mosfet transistors [IRF520]• 5 Capacitors 100nF 3 Capacitors 100uF 5 Capacitor 100pF• 5 Diodes [1N4007]• 3 Transparent gels (red, green, blue)• 1 Male pins strip (40x1)• 20 Resistors 220 ohm 5 Resistors 560 ohm 5 Resistors 1 kilohm 5 Resistors 4.7 kilohm 10 Resistors 10 kilohm 5 Resistors 1 megohm 5 Resistors 10 meghohm |
|--|--|



Le livre de projets

- Un ensemble de conseil technique pour découvrir l'Arduino
- Un ensemble de projets pédagogique
 - 01 Get to Know Your Tools
 - 02 Spaceship Interface
 - 03 Love-o-Meter
 - 04 Color Mixing Lamp
 - 05 Mood Cue
 - 06 Light Theremin
 - 07 Keyboard Instrument
 - 08 Digital Hourglass
 - 09 Motorized Pinwheel
 - 10 Zoetrope
 - 11 Crystal Ball
 - 12 Knock Lock
 - 13 Touchy-feely Lamp
 - 14 Tweak the Arduino Logo
 - 15 Hacking Buttons

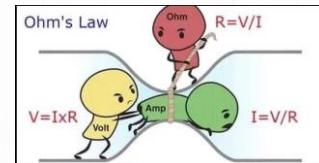
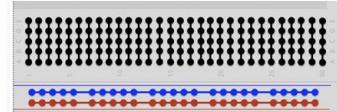
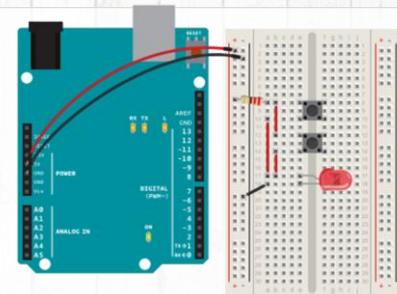


https://bastiaanvanhengel.files.wordpress.com/2016/06/arduino_projects_book.pdf

Projet

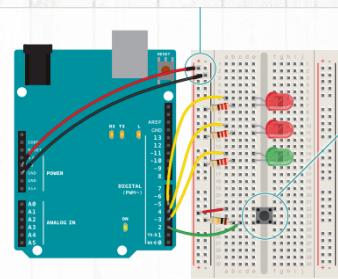
01 Get to Know Your Tools

- Page 21 (v. EN)
- Appréhension de la breadboard
- Appréhension des pins d'alims de l'arduino
- Connaissance électronique :
 - La diode électro luminescente
 - Relation U/R/I



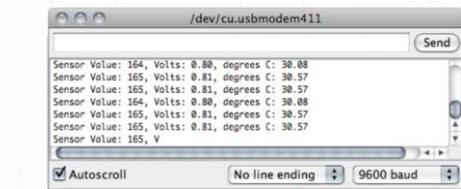
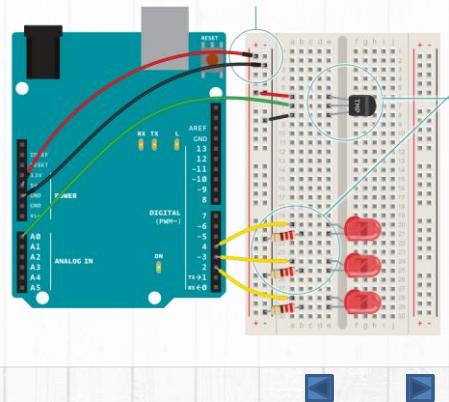
Projet 02 Spaceship Interface

- PullUp / PullDown resistor
 - pinMode
- Gestion entrées
 - DigitalRead
- Gestion sorties
 - digitalWrite
- Test logiciel
 - Niv avancé : Gestion par vecteur & debounce



Projet 3 love Ô Meter

- Page 44 (v. EN)
- Prise en main d'un capteur analogique
 - Capteur de température TMP36
 - analogRead
- Liaison serie



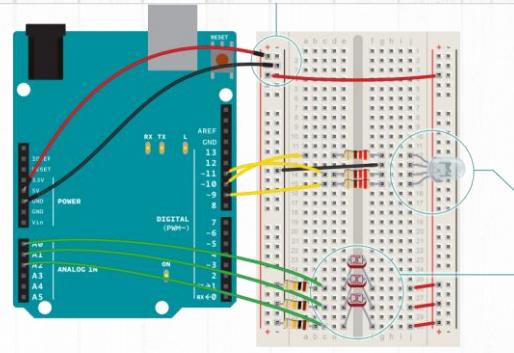
A screenshot of a terminal window titled "/dev/cu.usbmodem411". It shows a series of sensor readings being printed to the screen. The data is as follows:

```
Sensor Value: 164, Volts: 0.88, degrees C: 30.88
Sensor Value: 165, Volts: 0.81, degrees C: 30.57
Sensor Value: 165, Volts: 0.81, degrees C: 30.57
Sensor Value: 165, Volts: 0.88, degrees C: 30.88
Sensor Value: 165, Volts: 0.81, degrees C: 30.57
Sensor Value: 165, Volts: 0.81, degrees C: 30.57
Sensor Value: 165, V
```

The window includes standard terminal controls like 'Send', 'Autoscroll', and baud rate selection (9600 baud).

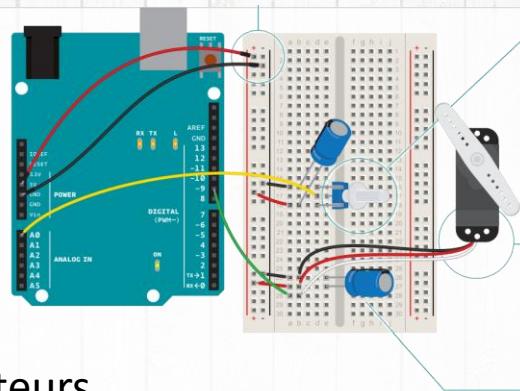
Projet 4 ColorMixingLamp

- Page 53 (v. EN)
- Lecture analogique
 - analogRead
- Modulation PWM
 - digitalWrite



Projet 5 MOOD CUE

- Page 63 (v. EN)



- Gestion des servo-moteurs

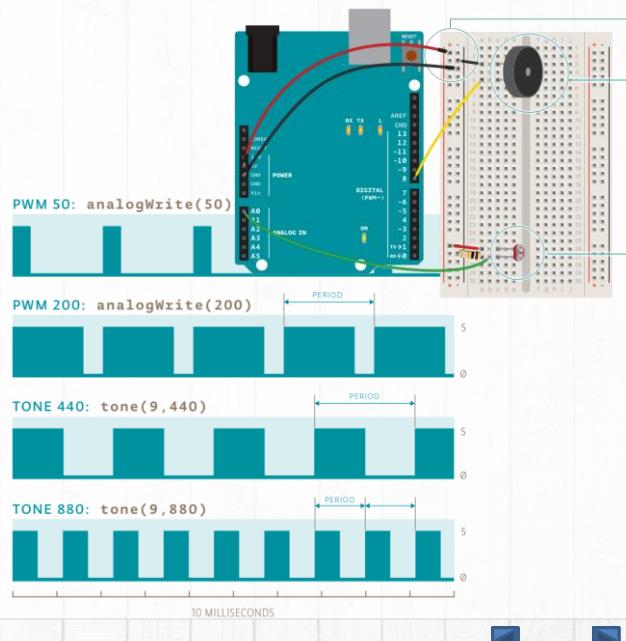
- Servo write



Projet 6

LIGHT THEREMIN

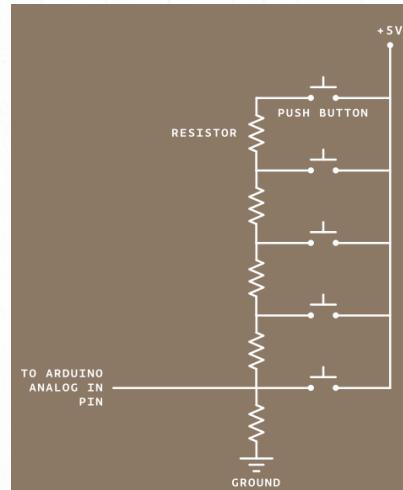
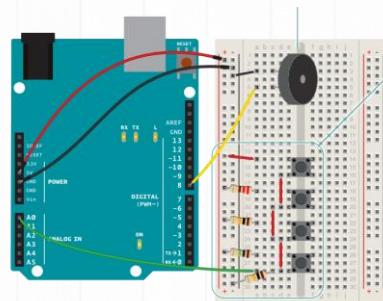
- Page 71 (v.EN)
- Buzzer
 - Tone
 - analogWrite
 - Frequency
 - delay



Projet 7

KEYBOARD INSTRUMENT

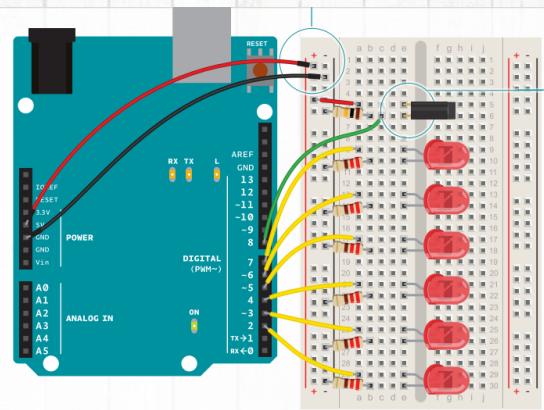
- Page 79 (v.EN)
- Pont diviseur
- Lois d'hom
- Multiplexage numérique



Projet 8

DIGITAL HOURGLASS

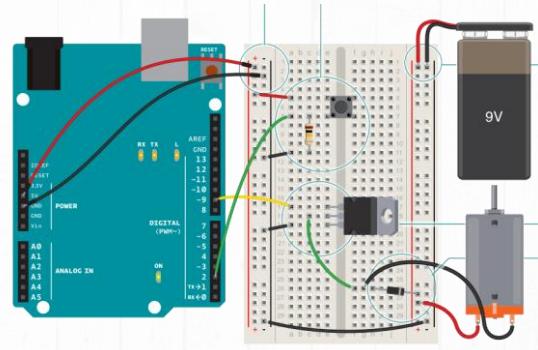
- Page 87 (v.EN)
- Tilt switch
- Gestion du temps
- Aprehension des fonctions time



Projet 9

MOTORIZED PINWHEEL

- Page 95 (v.EN)
- MOSFET
- DC motor
- Isolation basse / forte tension

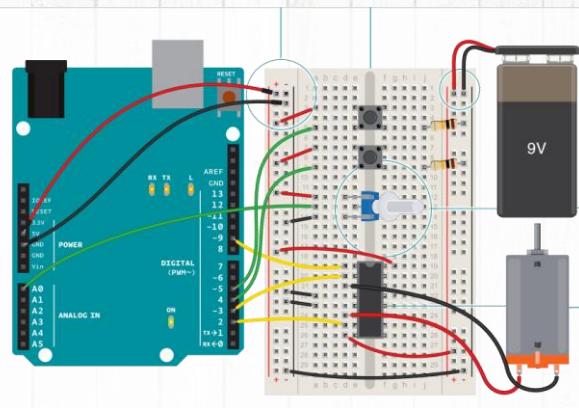


Projet 10 ZOETROPE

- Page 103 (v.EN)

- Contrôle de DC
Moteur avec CI

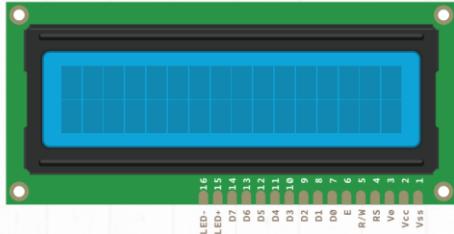
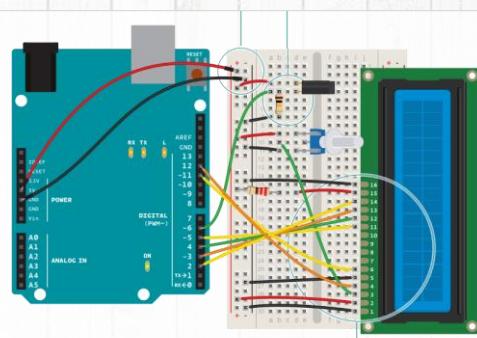
H-BRIDGE L293



Projet 11

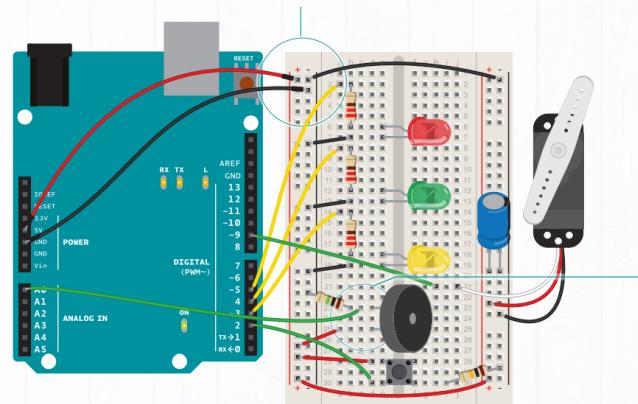
11 Crystal Ball

- Page 115 (v.EN)
- Usage du LCD
 - Lib liquidCrystal
- Exercice avancé :
 - lcd avec PCF8574



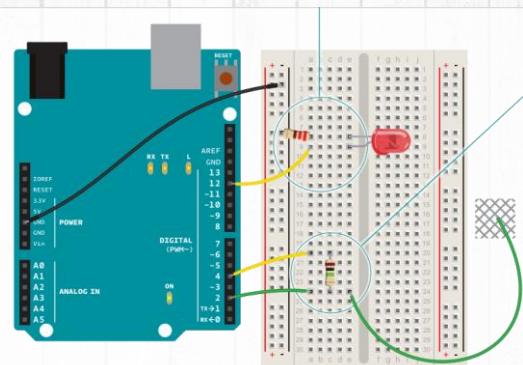
Projet 12

- Synthèse



Projet 13 Touchy-feelyLamp

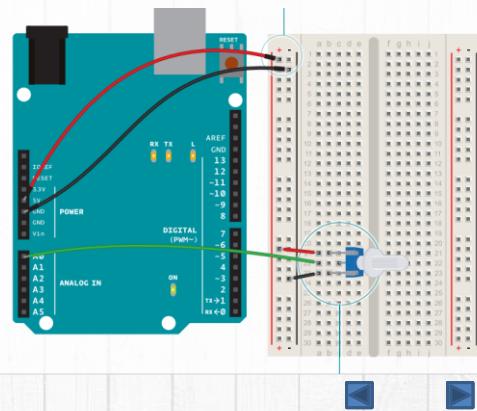
- Page 137 (v.EN)
- Pont diviseur pour sensors capacitifs
 - CapacitiveSensor



Projet 14

Tweakthe Arduino Logo

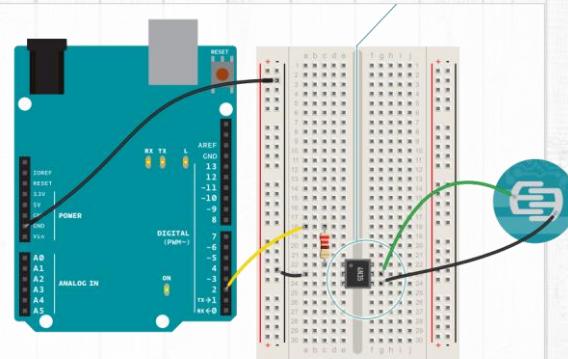
- Page 145 (v.EN)
- Manipulation serie avancé
 - Serial.write



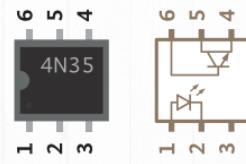
Projet 15

15 Hacking Buttons

- Page 157 (v.EN)



- Usage d'optocoupleur



Conclusion

❑ Dans cette partie nous avons aborder :

- ✓ Contenu du starter Kit
- ✓ Conceptions des projets du kit



CONDUIRE UN PROJET EMBARQUÉ



Sommaire

❑ Dans cette partie, nous allons aborder ;

 ❑ Cibler les besoins

 ❑ Dimensionner les besoins matériels

 ❑ Coder le projet

 ❑ Concevoir un prototype

 ❑ Passer de projet à produit



Cibler les besoins

• Détail des besoins type :

– Capter

• IHM :

- boutons, variations manuelle,
- clavier, touch, camera,
- biométrie, badge, Qr, barcode...

• La météo :

- T°c, Pression, humidité,...

• La mécanique :

- Contact, rayon laser,
- Rotations, pression,
- Forces, masse, ...

• Environnemental :

- luminosité, engrais,
- ph, salinité,
- profondeur, distance,
- gaz, rails, batteries ...

Cibler les besoins

- Capter
- Agir
- Réfléchir

Dimensionner les besoins matériels

Coder le projet

Concevoir un prototype

Passer de projet à produit



Cibler les besoins

• Détail des besoins type :

— Agir sur

- IHM :

- ELECTROCUTER,
- permettre l'accès,
- assurer le confort,
- avertir, ...

- L'électricité :

- allumer, éteindre,
- réguler, contrôler, ...

- La mécanique :

- Ouvrir, fermer,
- tourner, avancer,
- démarrer, stopper,
- incliner, ...

- Environnemental :

- lumières,
- engrais, ph ,
- nourrir, arroser, refroidir,
- réguler, chauffer, ...

NB : A
EVITER
😊

Cibler les besoins

- Capter
- Agir
- Réfléchir

Dimensionner les besoins matériels

Coder le projet

Concevoir un prototype

Passer de projet à produit



Cibler les besoins

• Détail des besoins type :

– Réfléchir à :

• IHM :

- assurer une présentation des données,
- comprendre des phrases, y répondre,
- Interpréter des info,
- tirer des conclusion et des alertes, ...

• Stockage :

- SD,
- structuration de l'information,
» xml, json, csv

• Sécurité :

- communiquer avec d'autres composants,
- données critiques,
- assurer un réseau,
- agir depuis un réseau, ...

Cibler les besoins

- Capter
- Agir
- Réfléchir

Dimensionner les besoins matériels

Coder le projet

Concevoir un prototype

Passer de projet à produit



Dimensionner le besoin

• Détail des besoins type :

– Capter

- Penser au vitesses d'acquisition
- Vidéo et radio

- Camera,
- dvb, FM,
- rf, wifi

• Analogique

- Couple de composants analogique,
- Modules préassemblés
- CAN >10bits, CAN I²C supplémentaire(s)
- CAN 8bits RC / PWM émulation
- Résistor array

• Numérique :

- Circuit intégré, Extension de GPIO IC,
- Modules préassemblés
- CAN SPI
- SPI

Cibler les besoins

Dimensionner les besoins matériels

- Sensors
- Actionneur
- Puces
- plaques

Coder le projet

Concevoir un prototype

Passer de projet à produit



Dimensionner le besoin

• Détail des besoins type :

– Agir sur

- Penser au vitesses d'action
- **Les mouvements**
- Penser aux forces a mettre en œuvre
 - Servo, servo continu
 - Moteur DC, pas a pas
 - Vérins, accumulateurs,
 - Relay,
 - Tapis roulant...

• L'électricité

- Relay,
- Dimmer, ...

• Environnement :

- Thermostat, Résistance chauffage
- Valve solénoïde, pompe,
- Peristaltic, ventilateurs,
- Vis sans fin, ...

Cibler les besoins

Dimensionner les besoins matériels

- Sensors
- Actionneur
- Puces
- plaques

Coder le projet

Concevoir un prototype

Passer de projet à produit



Dimensionner le besoin

Cibler les besoins

Dimensionner les besoins matériels

- Sensors
- Actionneur
- Puces
- plaques

Coder le projet

Concevoir un prototype

Passer de projet à produit

• Détail des besoins type :

– Puces sur

- Penser au vitesses d'action

• Processeur

- En fonction de la taille du contenu a manager de la vitesse et du besoin de ports et de fonctionnalité de la puce

- Atmel

- » 328, MEGA2560, tiny85/84,

- » Arduino uno, nano, micro

- ESP

- » VROOM32, MCU, ESP8266,...

- PIC

- » PIC32, PIC16

• Stockage

- EEPROM 24C, EPROM

- SD, ...

- RAM flash

• Communication Protocolées:

- RS232<->USB : AT8 FT232, CP21

- Wifi : **ESP8266, EsP32**

- RJ45: W5100, ENC28J60,...

- GSM/GPRS: SIM800, SIM900, ...

- Radio :Zigbee, LoRa 2,4GL, SigFox, Vigik,...

- Lecteur de badge, de puces,...

- GPS : NMEA, NEO6, NEO7, SIM908,...



Coder les besoins

- Type de construction

- Projet simple

- Code C / C++
 - Code en fichier ino
 - Inclusion de librairies extérieures
 - Compilation simple
 - Modularité et portabilité limitée
 - Bootloader d'origine
 - **SIMPLE**

Cibler les besoins

Dimensionner les besoins matériels

Coder le projet

- Code simple
- Code modulaire

Concevoir un prototype

Passer de projet à produit



Concevoir le prototype

Cibler les besoins

Dimensionner les besoins matériels

Coder le projet

Concevoir un prototype

- Breadboard
- Plaque maison
- Pré-série
- Boitier

Passer de projet à produit

• Détail des besoins type :

– concevoir :

- Créer et tester en breadboard

- Créer une plaque d'essais

- Fer à repasser
 - Prestataire prototype

- Faire fabriquer des plaques

- Prestataires,
 - Soudages CMS

- Trouver un boitier

- Résine, moule, 3D , imprimante,..
 - Design,...



Du prototype au produits

• En faire un produit:

- Production:
 - Prestataires,
 - Plaques en chine
 - Soudures cms
 - Test automatisé
 - Multicouches
- Soulever des fond
 - Incubateurs
 - » kickstarter
- Faire de beau « enclosures »
 - Design et incrustation des IHM

Cibler les besoins

Dimensionner les besoins matériels

Coder le projet

Concevoir un prototype

Passer de projet à produit

- Production
- Fonds
- boitier



Conclusion

□ Dans cette partie, nous allons aborder ;

- ✓ Cibler les besoins
- ✓ Dimensionner les besoins matériels
- ✓ Coder le projet
- ✓ Concevoir un prototype
- ✓ Passer de projet à produit



LIBRAIRIES



Sommaire

❑ Dans cette partie nous allons aborder :

- ❑ SoftwareSerial
- ❑ EEPROM
- ❑ LiquidCrystal
- ❑ wire

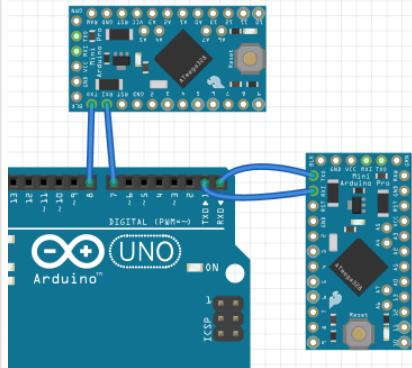


SoftwareSerial

- Librairie de gestion de communication série logiciel
 - `#include <SoftwareSerial.h>`
- définition d'un port série logiciel
 - `SoftwareSerial mySerial(rxPin, txPin);`
- Fonctionnement proche de HardwareSerial

Seul les fonctions suivantes sont disponibles

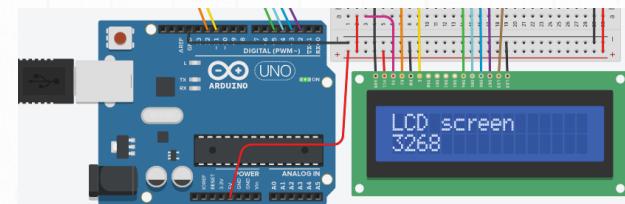
- `available()`
- `begin()`
- `isListening()`
- `overflow()`
- `peek()`
- `read()`
- `print()`
- `println()`
- `listen()`
- `write()`



pour la réception simultané sur plusieurs ports série logiciel préféré la librairie tiers AltSoftSerial : https://www.pjrc.com/teensy/td_libs_AltSoftSerial.html

liquidCrystal

- librairie pour afficheur LCD en lignes caractères
 - `#include <liquidCrystal.h>`
- définition d'un écran LCD
 - `LiquidCrystal lcd(rsPin, Epin, bit0,bit1,bit2,bit3,bit4,bit5,bit6,bit7);`
 - `LiquidCrystal lcd(rsPin, Epin,bit4,bit5,bit6,bit7);`



<https://www.arduino.cc/en/Reference/LiquidCrystal>

LiquidCrystal lib

Liste des fonctions disponibles :

- | | |
|--|---|
| <ul style="list-style-type: none">• LiquidCrystal()• begin()• clear()• home()• setCursor()• write()• print()• cursor()• noCursor() | <ul style="list-style-type: none">• blink()• noBlink()• display()• noDisplay()• scrollDisplayLeft()• scrollDisplayRight()• autoscroll()• noAutoscroll()• leftToRight()• rightToLeft()• createChar() |
|--|---|



EEPROM

- Pour l'accès à l'EEPROM interne
 - 512Bytes
 - `#include <EEPROM.h>`
- définition d'accès à un eeprom
 - `EEPROM.get (memoryAddr, dataToGet);`
 - `EEPROM.put (memoryAddr, dataToGet);`



EEPROM

- Librairie de lecture & écriture dans l'EEPROM interne

<https://www.arduino.cc/en/Reference/EEPROM>

- Examples

- [EEPROM Clear](#)
- [EEPROM Read](#)
- [EEPROM Write](#)
- [EEPROM Crc](#)
- [EEPROM Get](#)
- [EEPROM Iteration](#)
- [EEPROM Put](#)
- [EEPROM Update](#)

- Fonctions

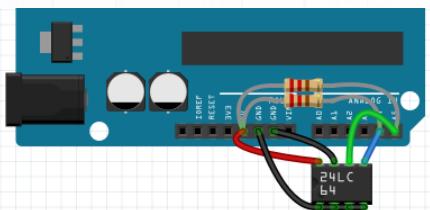
- [read\(\)](#)
- [write\(\)](#)
- [update\(\)](#)
- [get\(\)](#)
- [put\(\)](#)
- [EEPROM\[\]](#)



<https://www.arduino.cc/en/Reference/EEPROM>

Wire

- bibliothèque de base pour i²C
 - 2 adresses
 - adresse du périphérique sur le bus
 - adresse de la data dans la mémoire*
 - exemple : lire et écrire sur un EEPROM, une autre carte arduino, ... , sur un bus I²C



*uniquement dans le cas de composants à registre de stockage de valeur, certains composants tel que le GPIO EXTENDER PCF8574 sont en lecture et écriture direct et nécessite uniquement l'adresse du composant sur le bus

Les composants sur bus I2C

- Lecture d'une valeur sur un composant I²C sans registre

- Inclusion de la lib Wire.h
 - Initialisation du bus Wire
- Pour relever la valeur
 - Instancié la transmission
 - Requête un nb d'octets à une adresse de registre
 - Tant que des octets sont disponibles en lecture
 - Lire l'octet (1 par 1)
 - Clore la transmission

```
• EXEMPLE
#include <Wire.h>
#define CMP_ADR 0x27

void setup(){
    Wire.begin()
}
byte readValues()
{
    byte returnValues=0x0;
    Wire.beginTransmission(CMP_ADR);
    Wire.requestFrom(CMP_ADR, 1);
    while(Wire.available())
    {
        returnValues = Wire.read();
    }
    Wire.endTransmission();

    return returnValues;
}

void loop()
{ byte values = readValues();}
```

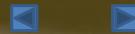
Les composants sur bus I2C

- Ecriture d'une valeur sur un composant I²C sans registre

- Inclusion de la lib Wire.h
 - Initialisation du bus Wire

- Pour relever la valeur
 - Instancié la transmission
 - Ecriture
 - Clore la transmission

```
• EXEMPLE
#include <Wire.h>
#define CMP_ADR 0x27
byte value=0;
void setup() {
    Wire.begin()
}
void loop() {
    Wire.beginTransmission(CMP_ADR);
    Wire.write(value);
    Wire.endTransmission();
    value++;
}
```



Les composants sur bus I²C

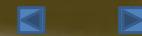
- Lecture d'une valeur dans un registre d'un composant I²C

- Positionnement dans le registre à l'adresse de la donnée
- Pour relever la valeur
 - Requête un nb d'octets à une adresse de registre
 - Tant que des octets sont disponibles en lecture
 - Lire l'octet (1 par 1)

```
• EXEMPLE
#include <Wire.h>
void readFrom(int device,
              byte address,
              int num,
              byte buff[])
{
    Wire.beginTransmission(device);
    Wire.write(address);
    Wire.endTransmission();

    Wire.beginTransmission(device);
    Wire.requestFrom(device, num);

    int i = 0;
    while(Wire.available())
    {
        buff[i] = Wire.read();
        i++;
    }
    Wire.endTransmission();
}
```



Les composants sur bus I2C

- Ecriture d'une valeur dans un registre d'un composant I²C

- Instanciation de la transmission

```
Wire.beginTransmission(device);
```

- Positionnement dans le registre à l'adresse de la donnée

```
Wire.write(address);
```

- Ecriture de la valeur
 - 1 seul octet

```
Wire.write(val);
```

- Clôture de transmission

```
Wire.endTransmission();
```

```
}
```

Conclusion

❑ Dans cette partie nous avons aborder :

- ✓ SoftwareSerial
- ✓ EEPROM
- ✓ LiquidCrystal
- ✓ wire



UTILISONS QUELQUES MODULES



Sommaire

❑ Dans cette partie nous avons aborder :

- ❑ Un ensemble de librairies
- ❑ Leur fonctionnement
- ❑ Leur docs



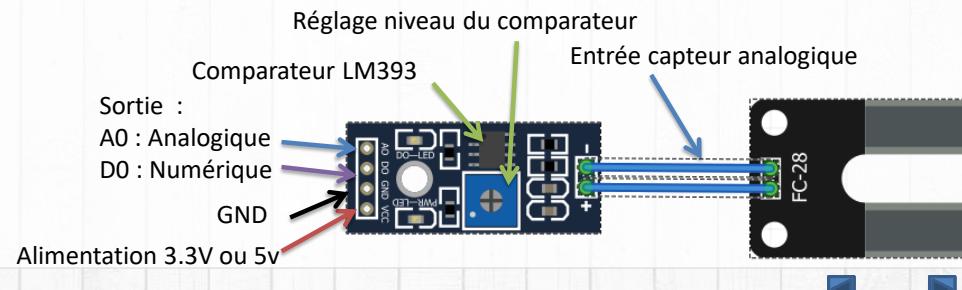
composants passifs

- Thermistance
- LDR
- Assemblage par pont de résistances



Analogique & Comparateurs Digitaux – YL-38

- Compare une tension de référence et une tension de capteur
 - à base d'ampli OP LM393
 - compatible capteur passif dipôles
 - Thermistances, résistance variables, ...
 - Exemple avec un capteur d'humidité du sol



Actionneurs Digitaux

- Utilise un relai pour isoler des circuits de puissances différentes
- Souvent utilisé en domotique (3.3V/5v pilote du 110/220V)
- 1 seule entrée / sortie par canal
- Un signal **HIGH** active la sortie
- Un signal **LOW** désactive la sortie



HC-SR04

- Télémètre à ultrasons
 - Plage : 2 cm à 400 cm
 - angle de détection : 15°
 - résolution : 0.3cm
- Rapport entre le temps de réception d'un son émis (pendant 10µs) puis reçu & la vitesse du son
- Ports :
 - Trig : émetteur
 - echo : récepteur



<https://www.gotronic.fr/pj2-hc-sr04-utilisation-avec-picaxe-1343.pdf>
lib : <https://github.com/Martinsos/arduino-lib-hc-sr04>

DHT11 / DHT22

- Sonde d'humidité résiduelle & température

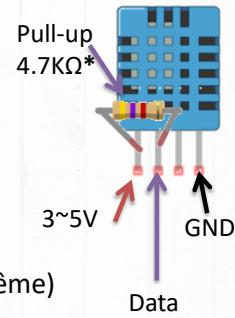
- DHT11

- » Humidité : 20% ~ 80% \pm 5%
 - » T°C : 0°C ~ 50°C \pm 2°C
 - » Fréquence : 1Hz (1/sec)
 - » Dégradation : \pm 1%/an

- DHT22

- » Humidité : 0% ~ 100% \pm 2% (5% en extrême)
 - » T°C : -40°C ~ 150°C \pm 0.5°C
 - » Fréquence : 2Hz (2/sec)
 - » Dégradation : \pm 0.5%/an

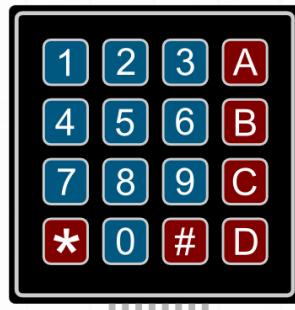
- Librairie DHT



*Pull-up de 4.7KΩ ou 10KΩ dans certains cas

Array Keypad

- Découpage en colonnes & en lignes
- Différents modèles
 - 4 x 5
 - 1 x 4
 - ...
- avec boutons
- ou avec membranes
- [librairie keypad](#)



KEYPAD PINOUT:

PIN 1: COL 4
PIN 2: COL 3
PIN 3: COL 2
PIN 4: COL 1
PIN 5: ROW 4
PIN 6: ROW 3
PIN 7: ROW 2
PIN 8: ROW 1



doc : <http://playground.arduino.cc/Code/Keypad>

Array Keypad

- inclusion de la lib
- Définition des dimensions
 - nb de lignes
 - nb de colonnes
- Définition du *keymap*
- Définition des pin
 - lignes
 - Colonnes
- Définition du keypad
 - **makeKeymap(hexaKeyMap)**
 - pins des lignes
 - pins des colonnes
 - constante ligne, constante colonnes
- Lecture des caractères saisies

```
#include <Keypad.h>
const byte ROWS = 4;
const byte COLS = 3;

char hexaKeyMap[ROWS][COLS] = {
{'1','2','3'}, {'4','5','6'},//Row 0 & 1
{'7','8','9'}, {'N','0','Y'}};//Row 2 & 3

byte rowPins[ROWS] = {3, 2, 1, 0};
byte colPins[COLS] = {7, 6, 5, 4};

Keypad arrayKeypad =
Keypad( makeKeymap(hexaKeyMap),
rowPins, colPins, ROWS, COLS);

void loop(){
char customKey = customKeypad.getKey();
}
```



Array Keypad

- `begin(makeKeymap(userKeymap))`
- `waitForKey()`
- `getKey()`
- `getState()`
- `keyStateChanged()`
- `setHoldTime(unsigned int time)`
- `setDebounceTime(time)`
- `addEventListener(keypadEvent)`



adresse de la doc : <https://www.arduino.cc/en/Reference/LiquidCrystal>

carte SD

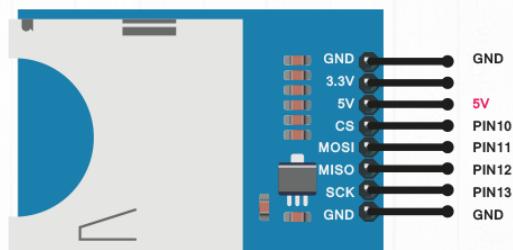
- communication SPI

- 3.3V
 - certains sous 5v

- Lecture, Ecriture

- Librairies

- SDFatLib
 - SD



SD lib

- Exemple de sd lin
- SD.begin(cs)
 - Renvoie true si ok
 - Demarre la communication
- Open
 - Ouvrir le fichier
- Close
 - Fermeture
- Read, write

```
#include <SPI.h>
#include <SD.h>

File myFile;
void setup() {
    if (!SD.begin(CSPin))
        Serial.println("init failed!");

    myFile = SD.open("test.txt",
                     FILE_WRITE);

    if (myFile) {
        myFile.println("testing 1, 2, 3.");
        myFile.close();
    }
    myFile = SD.open("test.txt");
    if (myFile) {
        while (myFile.available())
            Serial.write(myFile.read());
        myFile.close();
    }
}
```



Pour accéder plus en profondeur

- Autre exemple d'accès aux volumes des cartes SD

```
#include <SPI.h>
#include <SD.h>

Sd2Card card;
SdVolume volume;
SdFile root;

void setup()
{
if (!card.init(SPI_HALF_SPEED, CSpin))
    Serial.println("Error");

if (!volume.init(card))
    Serial.println("error partition ");

root.openRoot(volume);
root.ls(LS_R | LS_DATE | LS_SIZE);
}
```



Liquid Crystal

- Afficheur de lignes caractères

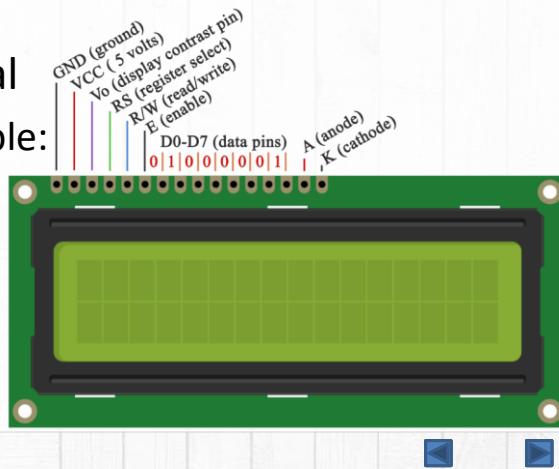
- Existe en différents gabaries

- (ex 16/20 chars sur 1, 2 ou 4 lignes)

- Librairie liquidCrystal

- contrôleur compatible:

- [Hitachi HD44780](#)



datasheet : <https://www.sparkfun.com/datasheets/LCD/HD44780.pdf>

librairie liquidCrystal

- [LiquidCrystal\(\)](#)
- [begin\(\)](#)
- [clear\(\)](#)
- [home\(\)](#)
- [setCursor\(\)](#)
- [write\(\)](#)
- [print\(\)](#)
- [cursor\(\)](#)
- [noCursor\(\)](#)
- [blink\(\)](#)
- [noBlink\(\)](#)
- [display\(\)](#)
- [noDisplay\(\)](#)
- [scrollDisplayLeft\(\)](#)
- [scrollDisplayRight\(\)](#)
- [autoscroll\(\)](#)
- [noAutoscroll\(\)](#)
- [leftToRight\(\)](#)
- [rightToLeft\(\)](#)
- [createChar\(\)](#)



adresse de la doc : <https://www.arduino.cc/en/Reference/LiquidCrystal>

LiquidCrystal

- inclusion de la lib.
- Instancier un *LiquidCrystal*
lcd en variable globale
- Démarrer le lcd avec
 - colonnes
 - lignes
- Positionner le curseur
 - colonne
 - ligne
- Ecrire du contenu

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

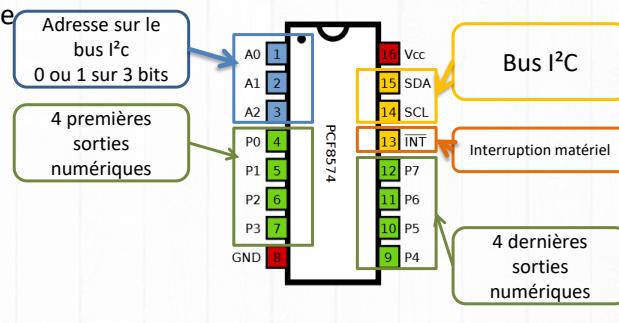
void setup() {
    lcd.begin(16, 2);
}

void loop(){
    lcd.setCursor(0, 1);
    lcd.print("Some text");
}
```



PCF8574

- I²C I/O extender
 - 8 Sorties numérique TOR
 - Reçoit un entier sur 8 bits
 - Transformé en sortie 8x1bit
 - P0~P7
 - adresse sur 3 bits
 - Ports adresse a 0V ou +5v
 - 7 pcf8574 possible sur un bus I²C
- exemple :
 - Utilisé pour un pour faire un lcd en I²C
- [librairie lcd i²c](#)



librairie lcd i²c : https://github.com/mathertel/LiquidCrystal_PCF8574

LiquidCrystal_PCF8574

- inclusion de la lib & dépendances
- Instancier un *LiquidCrystal_PCF8574* lcd en variable globale
- Démarrer le lcd avec
 - colonnes
 - lignes
- Gérer l'éclairage
 - Niveau d'intensité
- Positionner le curseur
 - colonne
 - ligne
- Ecrire du contenu

```
#include <Wire.h>
#include <LiquidCrystal_PCF8574.h>

LiquidCrystal_PCF8574 lcd(0x27);

void setup() {
    lcd.begin(16, 2);
}

void loop(){
    lcd.setBacklight(255);
    lcd.setCursor(0, 1);
    lcd.print("Some text");
}
```



tft

- Librairies propriétaires

- TFT
- UGLC
- QDTECH

- Communication SPI

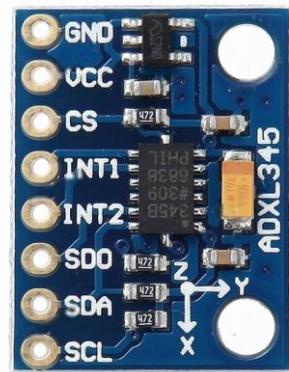
- MISO sd0
- MOSI sda
- CLK clk
- CS cs



<https://next-hack.com/index.php/2017/08/14/how-to-play-a-video-on-arduino-uno-46-playing-a-20-fps-animation-from-sd-card/>

Accéléromètres ADXL345

- Accéléromètre 3 axes
 - Plage de mesure : ± 2 , ± 4 , ± 8 , ± 16 g
 - max : 10,000 g
- SPI ou I²C
 - spi
 - CS (Cable Select), SCL (Clock), SDO (miso), SDA (mosi)
 - I²C
 - SCL, SDA
- Interruption matériels
 - INT1
 - tap, doubletap
- Librairie ADXL345
- Variante MPU6050

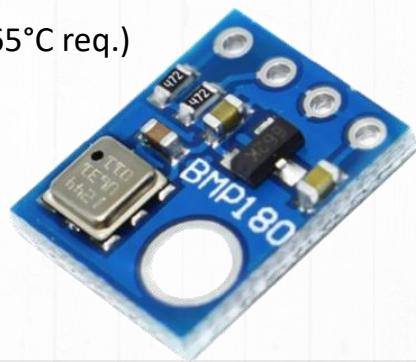


datasheet :

<https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf>

Capteur de pression BMP180

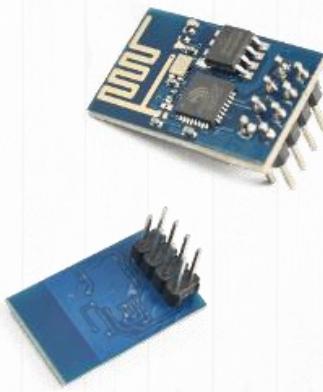
- Capteur de pression atmosphérique
 - Plage : 300 → 1100 hPa (altitude +9000 m à -500 m)
 - basse consommation 5µA/Sec
 - 0.06 hPa (0.5 m) in ultra low power mode
 - 0.02 hPa (0.17 m) advanced resolution mode
 - Usage : -40°C et +85°C (0°C à 65°C req.)
- Access I²C
 - Ports : SDA, SCL
- Librairie BMP180



datasheet : <https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf>

nrf24l01

- nRF24L01 n'est pas WIFI
- Utilisé dans les périphériques 2.4Ghz sans fil
 - ex : claviers, souris , gamepad, télécommande d'aéromodélisme, drones, ...
- faible consommation
 - 26µA en stand by
- communication jusqu'à 2Mbps
- Librairie nrf24l01



https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf

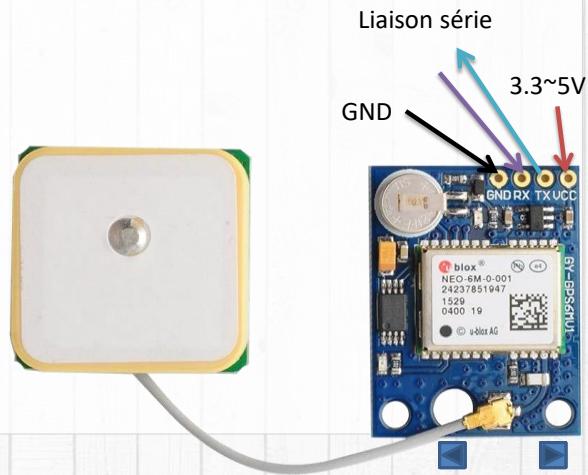
RJ45

- module de diffusion réseaux TCP/IP
 - Communication spi
 - W5100
 - Niveau OSI plus haut
 - » Couche IP
 - librairie w5100
 - Enc28j60
 - Bas niveau OSI
 - librairie enc28j60



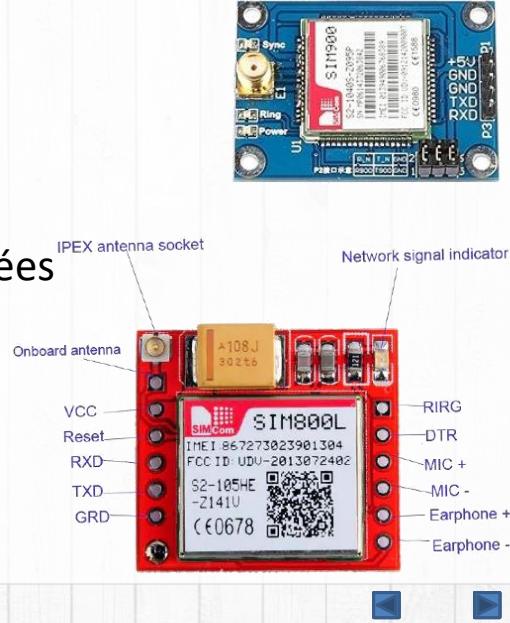
GPS & GNSS

- Communication série
- Message GNSS
- Configurable
 - Soft U-blox
 - Vitesse série
 - Fréquence
 - assistance CAP
 - type de codage
 - ...
- Librairie GNSS



GSM & commandes AT

- SIM900, SIM800
- Liaison série
- Capacité
 - appel
 - data
 - SMS
- Commande AT normalisées
 - Liste dans la doc
- Certains modèles possède un GPS
 - » SIM908
- Existe en shield



MIC : microphone

Earphone : écouteur

RIRG : déclencheur de sonnerie

GRN : GND

ESP8266

- ATTENTION
 - Très proche du nrf24l01 visuellement
- Module wifi 2.4 Ghz
 - **2 GPIO** (ou plus*) autonome
 - Capacité Access Point wifi
- Microcontrôleur ESP8266
 - **Processeur, Flash & SRAM**



<http://fab.cba.mit.edu/classes/865.15/people/dan.chen/esp8266/>

Liste des commandes AT

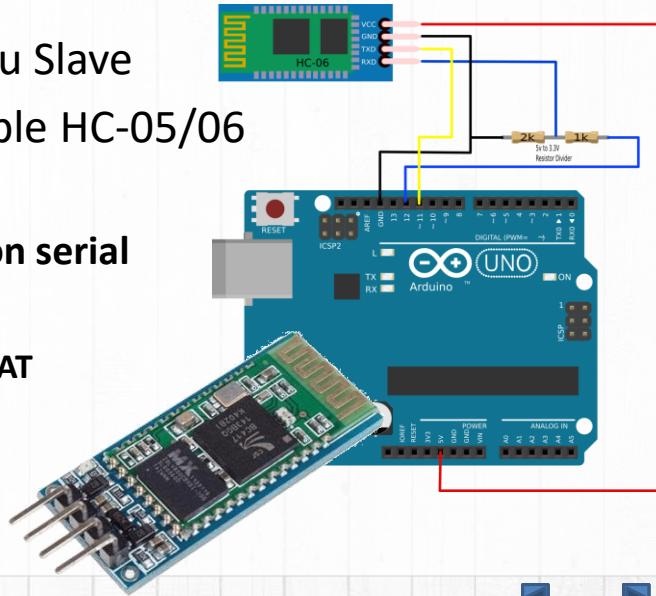
:https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf

Bluetooth HC-05 / HC-06

- HC05 slave
- HC06 Master ou Slave
- SPP-C compatible HC-05/06

— communication serial

- Commandes AT



RTC

- Real Time Clock DS1307
- I²C
- Possède un EEPROM
- Nécessite une pile
- Librairie RTC



librairie logiciel

- pour les atmel 32u4

- [MouseKeyboard](#)

- permet la saisie clavier et/ou souris par l'arduino pour les ordinateurs (vue comme un périphérique de saisie USB)

- Pris en compte comme clavier ou souris USB (OTG like)



<https://www.arduino.cc/en/Reference.MouseKeyboard>

Conclusion

❑ Dans cette partie nous avons aborder :

- ✓ Un ensemble de librairies
- ✓ Leur fonctionnement
- ✓ Leur docs



LES INTERRUPTIONS



Sommaire

❑ Dans cette partie nous allons aborder :

- ❑ Les vecteurs d'interruption
- ❑ Conception évènementiel matériel
- ❑ Les timers



interruptions

- Pause dans l'exécution de l'instance *loop()*
 - *Reprise au point où la mise en pause est survenue*
- 2 familles d'interruptions
 - Internes
 - Les timers
 - les compteurs
 - externes
 - Les ports d'interruptions matériels



Les timers

- AVR ATmega328P

- 3 timers

- timer 0

- 8 bits

- utilisé par les fonctions delay(), millis() et micros()

- pwm sur 5 et 6

- timer 1

- 16 bits

- utilisé par la lib Servo

- PWM sur 9 et 10

- timer2

- 8 bits

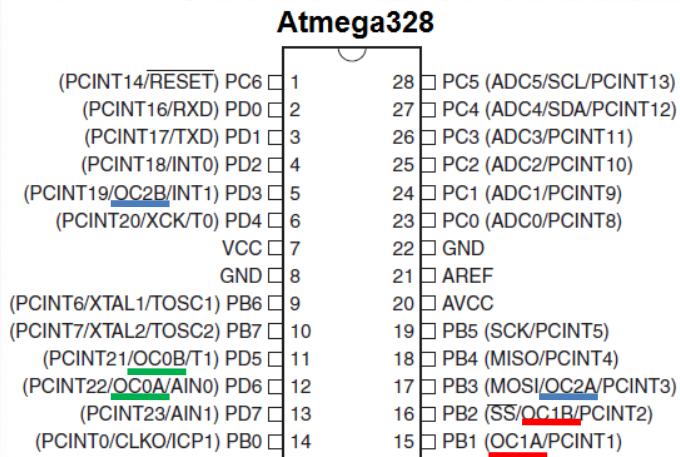
- utilisé par tone()

- PWM sur 3 et 11



timer0

- Timer 0
 - Ports PWM
 - OC0A
 - OC0B
- Timer 1
 - ports
 - OC1A
 - OC1B
- Timer 2
 - ports
 - OC2A
 - OC2B



timer & pwm

- Liens

- <https://www.locoduino.org/spip.php?article119>

<https://www.locoduino.org/spip.php?article119>



Les interruption matérielles

- Sur le UNO
 - 2 ports d'interruption matériels
 - D2 et D3
- Interruption sur les états (*mode*)
 - vers haut **RISING**
 - vers bas **FALLING**
 - changement d'état **CHANGE**
 - tant que niveau bas **LOW**
 - *tant que niveau haut HIGH **
- Attacher des fonctions au interruptions
 - `attachInterrupt(pin, functionName, mode);`
- Détacher une interruptions
 - `detachInterrupt(pin);`



<https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/>
pins des interruption matériels
Uno, Nano, Mini, other 328-based → 2, 3
Uno WiFi Rev.2 → all digital pins
Mega, Mega2560, MegaADK → 2, 3, 18, 19, 20, 21
Micro, Leonardo, other 32u4-based → 0, 1, 2, 3, 7
Zero → all digital pins, except 4
MKR Family boards → 0, 1, 4, 5, 6, 7, 8, 9, A1, A2
Due → all digital pins
101 → all digital pins (Only pins 2, 5, 7, 8, 10, 11, 12, 13 work with **CHANGE**)

Les interruption matérielles

- Les interruptions peuvent toutes être mise en sommeil et réveiller pour l'exécution de taches critique ne pouvant être interrompus
 - Désactiver toutes les interruptions
 - `noInterrupts();`
 - Réactiver les interruptions
 - `interrupts();`



Conclusion

❑ Dans cette partie nous avons aborder :

- ✓ Les vecteurs d'interruption
- ✓ Conception évènementiel matériel
- ✓ Les timers





FABRIQUEZ VOS MODULES



Sommaire

❑ Dans cette partie nous allons aborder :

- ❑ La création de modules / shield
- ❑ La confection de carte
- ❑ Le routage
- ❑ La gravure de carte



1. Préparer un projet

- Lister les besoins d'informations
 - Nomenclature capteurs
- Lister les besoin d'actions
 - Nomenclature relais, électrovannes, servos, moteurs,...
- Lister les modes de communication
 - nomenclature
 - écrans lcd ou tft
 - claviers, interface de saisie
 - interfaces réseaux rj45 ou wifi, rf, ...
- Apprécier les distances
 - nécessité de communication entre les modules ex :IR, wifi, nrf24l01, RJ45 ,...
 - gestion des alimentation des modules autonomes
- Dimensionner le processeur
 - En fonction du nombre d'entrées sorties nécessaire
 - En fonction de certaines fonctionnalité de la carte (ex usb)
 - En fonction de la taille de la flash et de la SRAM



2. Schématique / Assemblage

- Schématique et assemblage virtuel sous fritzing
- ou sur shield on empile
 - attention aux câbles select des shield
 - Brancher et coder
- ou sur breadboard on câble
 - bien vérifier avant de brancher
 - plus pratique pour porter sur une nouvelle shield
 - brancher et coder...



3 . Router une shield

- Router = relier les modules sur une carte pour l'imprimer par la suite
- fritzing contient
 - des modèles d'empreintes pour les arduinos
 - des modèles d'empreintes de modules
 - Un service d'impression de carte jusqu'à 2 couches a l'unité
 - Un module d'export GERBER pour les couches



Impression

- JLPCB
 - pas cher
 - rapide
- AllPCB
 - pas cher
- Impression Maison
 - Technique imprimante laser & fer à repasser
 - chlorure de fer & étamage à froid
 - Multicouche difficile à mettre en œuvre



Conclusion

❑ Dans cette partie nous avons aborder :

- ✓ La création de modules / shield
- ✓ La confection de carte
- ✓ Le routage
- ✓ La gravure de carte



ASSEMBLEUR AVR



Sommaire

❑ Dans cette partie, nous allons aborder :

- ❑ Les registres Arduino
- ❑ Atmel studio
- ❑ Le boot loader
- ❑ L'assembleur



Langage assembleur

- Intégration de segment de code avec asm.h dans un programme ino
 - Interrogation de registre, modification de registre
- Ou Pas besoin de bootloader arduino , *make your own*
- interprétation du code binaire
- Ecriture au plus proche de la puce
 - Un ensemble d'instructions machine
 - un ensemble de registres pour les datas



http://www.avr-asm-download.de/beginner_en.pdf

Utiliser les registres

- Nécessite `#include <avr/io.h>`
- Affectation d'une valeur dans un registre :
PORTB = val;
 - Utilisation forte des bits, flags et masques
PORTB = (B1001|PORTB);
- Des lots de GPIO arduino sont accessibles par registre
 - » Port B (broches numériques de 8 to 13)
 - » Port C (broches analogiques/numériques)
 - » Port D (broches numériques 0 to 7)



Usage des registres : http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n>Main.PortManipulation

Bootloader ou pas ?

- Bootloader démarre votre code et gère setup et loop
- Atmel studio
 - Flasher votre bootloader

http://www.vorobotics.com/wiki/index.php?title=Arduino_sans_bootloader



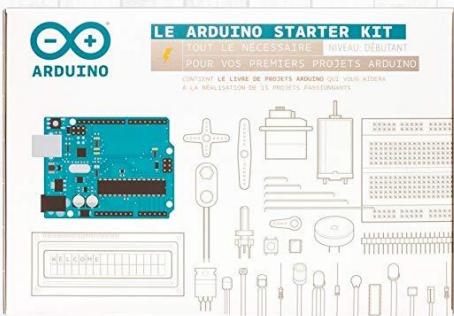
Conclusion

❑ Dans cette partie nous avons aborder :

- ✓ Les registres Arduino
- ✓ Atmel studio
- ✓ Le boot loader
- ✓ L'assembleur



Conclusion finale



ORSYS
formation

est heureux de vous offrir

**votre KIT Arduino pour
vous permettre de
continuer à progresser
chez vous**



Conclusion finale

- Remerciements et encouragements

Pendant cette formation, nous espérons :

- que vous avez appris de nombreuses choses théoriques et pratiques
- que vous y avez trouvé plus que ce que vous étiez venu chercher
- et que vous avez passé un bon moment avec votre professeur

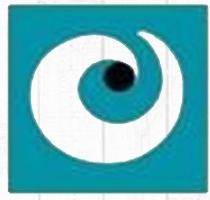
Merci d'avoir fait confiance à ORSYS, à bientôt pour d'autres formations !



Bibliographie / Références

- Livres de référence
 - « Le grand livre d'Arduino » de chez Eyrolles
- Magazines
 - <https://www.hackable.fr/>
- Sites de référence
 - GOOGLE : <http://google.fr>
 - <https://www.arduino.cc/>
(site de arduino - standards)
 - <https://zestedesavoir.com/>
 - <https://www.instructables.com/circuits>
 - <http://fritzing.org/>
(virtual breadboard, édition de typons,...)





ORSYS
formation

A BIENTÔT...

ARDUNO ET SES MODULES

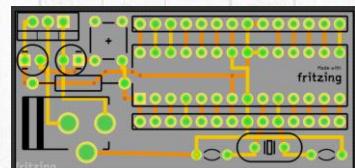
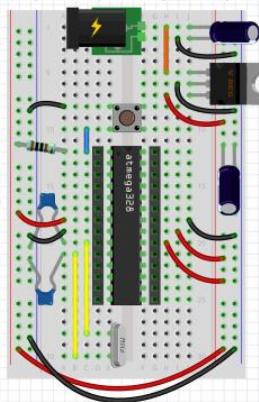
EXERCICES AVANCÉS



328p

- sur breadboard
- nécessite un usb<->rs232
- charger la puce par arduino ISP

Tout d'un arduino
routage plus complexe
rendre la carte
« compatible » shield



TP communication série

- Brancher un périphérique série sur les ports 0 TX, 1 RX
- Communiquer en entrée et sortie avec le périphérique
 - Demat le monde/hello world
- Avancé II:
 - Communiquer uniquement dans un ensemble
 - Communiquer avec son voisin
 - Se synchroniser
 - Préambule
 - Parler
 - Communiquer uniquement dans un ensemble de périphérique
 - Adressage et protocole de distribution



`Serial.read();`
`Serial.write(char); ou Serial.println(string)`

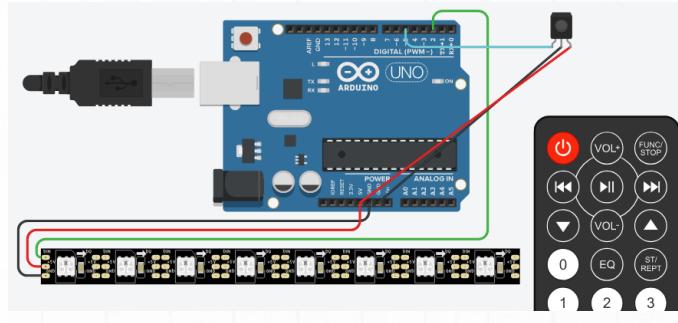
Feux tricolores

- Brancher 3 diodes
 - Sur un port numérique TOR
 - vert
 - jaune
 - rouge
- Définir les ports de sorties LED
 - pinMode
- faire une fonction qui allume chaque diode
- Version avancé
 - gérer 2 série de 3 feux tricolores
- Version avancé 2:
 - Gérer un capteur de présence



TP

- piloter une bar led neopixel à partir de la télécommande IR



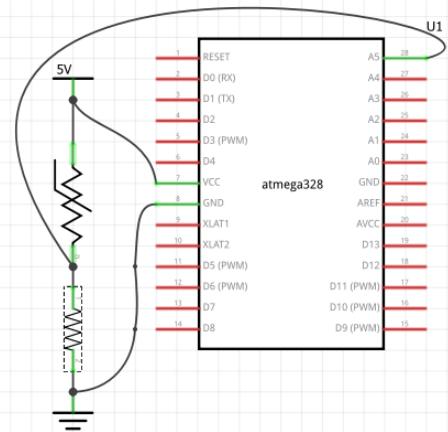
Capteurs et actionneurs numériques

- Utiliser un PIR sensor
 - Capte un état de mouvement par IR et envoie vcc si vrai
 - Allumer les lumières en cas de détection de passage
- Utiliser un LM393
 - Comparateur à seuil, vrai si supérieur à une tension de référence
 - Gérer des alertes sur un capteur analogique chaussé LM393
- Avancé II
 - Gérer les alertes par Interruption
 - Gérer 8 interruptions sur un seul câble
 - Gérer 16 interruptions puis 64
 - Usage PCF8574
 - Agir sur des actionneurs TOR sur un seul câble
 - Faire son montage LM393
 - Datasheet



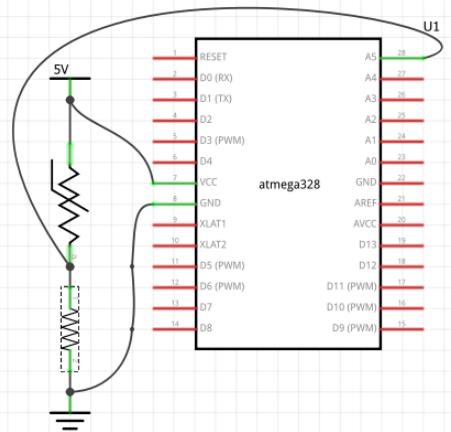
Capteur dipole à résistance variable

- Usage du CAN
 - analogRead
- Pont de kirshhoff
 - diviseur de tension
- Fabriquer un voltmètre
 - Serie , lcd, ou tft
- Utiliser une thermistance
 - Résistance variable en fonction de la T°C
 - Lire la température
 - Serie, lcd, tft
- Utiliser une LDR
 - Résistance variable en fonction de la lumière
 - Faire varier un éclairage (LED) en fonction de la luminosité ambiante



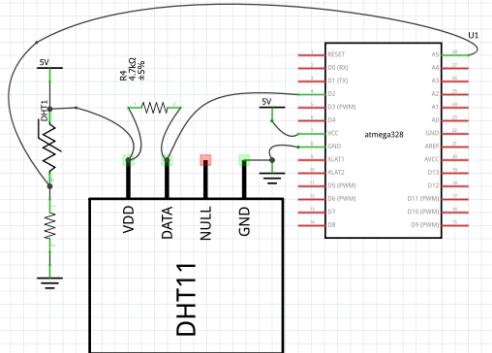
Capteur dipole à résistance variable

- Utiliser un capteur d'humidité de sol
 - Résistance variable en fonction de la Hsol%
 - Serie, lcd, tft, actionneur(pompe, ...)
- Utiliser un capteur de pluie
 - Résistance nulle à sec
 - Déetecter la pluie
 - Série, lcd, tft



Capteur d'humidité

- Utiliser DHT11/22
 - sélectionner une librairie
 - lire et écrire les valeurs vers le port série
 - la T°C
 - la T°C du DHT11
 - l'humidité ambiante



SUPER TP

- Faites en une seule puce, une station météo
- Celle-ci pourra :
 - Relever les infos météo(pluie, T°c, RH%, luminosité), agricole(humidité sol) et de présence (PIR)
 - Gérer le temps horaire
 - Afficher les données relevées
 - Liaison série
 - Gérer des seuils d'alertes
 - Actionner des éléments
 - » Pompes a eau, solénoïde
 - » Lumière
 - » Ventilateurs, machine a café, ...
- Avancé
 - Communiquer entre ensemble de capteurs
 - Et avoir qu'un afficheur central réseaux(rf, nrf, wifi, rj, ...)



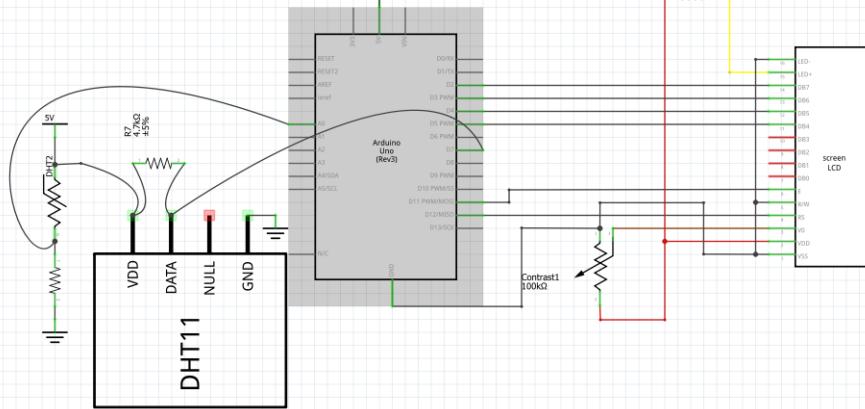
DS3231 RTC

- Connecter le DS3231
 - Mettre à l'heure la puce
 - Relever l'heure
- Avancé
 - Gérer des alarmes de temps



un écran LCD

- Afficher les données des deux capteurs
- pin
 - Les 4 derniers bits de données
 - E
 - RS



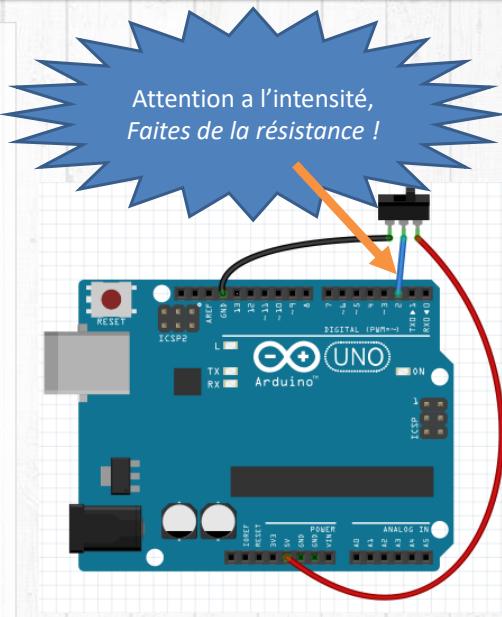
SUPER TP

- Afficher l'heure
 - Lcd, tft
- Afficher les valeurs météo, agricole, actionneurs



TP interruption

- Connecter un double switch sur 5V, GND, et sur une entrée d'interruption matériel
- Déclencher différentes fonctions suivant les changements d'états de l'entrée



Rotary encoder

- Relever des valeurs
 - bouton numérique + phototransistor et phototransistor
 - Afficher les valeurs sens + & -
 - Serie, lcd, tft
 - Relever l'état du bouton



SUPER TP

- Menu de sélection
 - Météo
 - Agricole
 - Heure
 - L'état des actionneurs



Bmp180 ou équivalent

- Lire la pression atmosphérique

- Communication avec la puce
 - Librairie

- Altitude relative

- Afficher les valeurs

- Tft, serie, lcd



SUPER TP

- Afficher l'état anticyclonique
 - Lcd, tft, serie
- Prévoir l'évolution
 - Sur 10 et 20 minutes
 - En 3 valeurs : + = -
- Avancé :
 - En déduire l'évolution des autres facteurs météo



Communication Serie

- Ecrire un client émetteur/récepteur qui souhaiterai
 - Diffuser ses x valeurs a un atmega328 ou Arduino
- Ecrire un serveur
 - Reçois et interprète les données
 - Affiche le contenu
 - Serie2 ou lcd, tft
- Avance:
 - Le serveur parle au client
 - **OK, ACTIVATE OUTPUT, FETECHDATA INPUT, DATA VALUE,**



SUPER TP

- Créer des petits module qui gère chacun 1 ou plusieurs sensor
 - Centralisation des data
 - Gestion protocolaire
- Créer un afficheur tft avec des valeurs série
- Avancé II :
 - Gestion menu des alarmes port d'action avec port d'alerte
 - Déclenchement autonome par groupe et gestion centrale



- RF433/866/915
 - Créer un émetteur / récepteur client
 - Créer un émetteur / récepteur serveur
- Gérer plusieurs périphériques
 - Gestion protocolaire
- Avancé:
 - réseau de **MESH**



SUPER TP

- Gérer des modules rf pour
 - Porte de garage : presence, bip(rf), porte, lumiere
 - Piece :Température, Humidité, luminosité, presence, Lumière, volets
 - Jardin / plante verte : Humidité, t°c, Humidité sol, lumiere, eau
- Centralisateur rf pour
 - Gestion de temps(distribué)
 - Ecran
 - Rotary encoder



SUPER TP

- Créer des modules client avec un réseaux rf qui capte les sensor, et agit sur les actionneur
 - Gérer un adressage
 - Afficher les datas secteur par secteur
 - Lcd, tft
 - Gérer les menus de sélection de secteur
- Avancé II :
 - Gérer les type de module auto ANNOUNCEMENT
 - Gérer un routage en mesh



NRF

- Demat nrf
 - Rencontre du 2eme type
- Diffuser des structures
 - protocole
- Gérer des modules
 - Adressage
- Avancé :
 - Faire des modules mesh
 - Réseaux mesh
 - Un client , un serveur



SUPER TP

- Gérer des modules nrf pour
 - Porte de garage : **presence, bip(rf), porte, lumière**
 - Piece : **Température, Humidité, luminosité, présence, Lumière, volets**
 - Jardin / plante verte : **Humidité, t°c, Humidité sol, lumière, eau**
- Centralisateur nrf pour
 - Gestion de temps (distribué)
 - Ecran
 - Rotary encoder



SD

- Lire un fichier et afficher les lignes sur écran
 - Série, lcd, tft
- Ecrire un logger de port série



SUPER TP

- Gérer des modules nrf pour
 - Porte de garage : présence, bip(rf), porte, lumière
 - Piece : Température, Humidité, luminosité, présence, Lumière, volets
 - Jardin / plante verte : Humidité, t°c, Humidité sol, lumière, eau
- Centralisateur nrf pour
 - Gestion de temps (distribué)
 - Ecran
 - Rotary encoder
 - SD logger



tft

- Demat tft
- Afficher un lot de valeur
- Avancé :
 - Afficher des Images depuis une carte SD
- Faire des fonction pour générer des composants graphique positionnable
 - Bar horizontale, bar verticale, valeur numérique, état
- Avancé :
 - Faire des composants d'état avec des images
- Avancé II:
 - Faire des composants graphiques complexes
 - Thermomètre, compteur demi cercle



SUPER TP

- Gérer des modules nrf pour
 - Porte de garage : presence, bip(rf), porte, lumière
 - Pièce :Température, Humidité, luminosité, presence, Lumière, volets
 - Jardin / plante verte : Humidité, t°c, Humidité sol, lumiere, eau
- Centralisateur nrf pour
 - Gestion de temps (distribué)
 - Ecran TFT
 - Rotary encoder
 - SD logger



SUPER TP

- Faire une interface type station météo
- Avancé II:
 - Prévoir des menus
 - gérer la synchro/adressage par menu
 - Modifier les niveaux d'alerte par écran



bluetooth

- Connecter un slave à une appli simple
 - Communiquer par Bluetooth avec un smartphone
- Remonter des infos capteurs
- Avancé III:
 - Afficher les données par une appli personnalisé codé pour android



Reseau

- Être un serveur web
- Parler au réseau
 - Faire des requête HTTP
- Répondre au réseau
 - Gérer un jeux de led
 - Envoie mail
- Avance II : Un serveur internet pour retenir les données
 - Exemple rest



SUPER TP

- Gérer des modules nrf pour
 - Porte de garage : présence, bip(rf), porte, lumière
 - Pièce :Température, Humidité, luminosité, présence, Lumière, volets
 - Jardin / plante verte : Humidité, t°c, Humidité sol, lumière, eau
- Centralisateur nrf pour
 - Gestion de temps(distribué)
 - Ecran TFT
 - Rotary encoder
 - SD logger
 - w5100



SUPER TP

- connecter votre station a internet(w5100)
 - Mettre en place un mode alarme
 - Activation code/ code réseau
 - Mettre en place des alertes
 - Mail intrusion
 - Module sécurité : Dispositif de fumé
 - Extinction des lumières, fermeture de volets, ...
 - Ouverture Portail,...



GPRS

- Piloter par 3G un jeu de led
- Envoyer des alertes
 - Mail, sms
- Avance II : Un serveur internet pour retenir les données
 - Exemple rest



SUPER TP

- Gérer des modules nrf pour
 - Porte de garage : présence, bip(rf), porte, lumière
 - Pièce : Température, Humidité, luminosité, présence, Lumière, volets
 - Jardin / plante verte : Humidité, t°c, Humidité sol, lumière, eau
- Centralisateur nrf pour
 - Gestion de temps(distribué)
 - Ecran TFT
 - Rotary encoder
 - SD logger
 - w5100
 - 3G GPRS



SUPER TP

- connecter votre station a internet GPRS
 - Mettre en place un mode alarme
 - Activation code/ code réseau
 - Mettre en place des alertes
 - Mail, sms intrusion
 - Dispositif de fumé
 - Extinction des lumières, fermeture de volets, ...
 - par sms
 - Ouverture Portail,...
 - sms
- Avancé II:
 - Gérer une communication téléphonique en cas d'alarme



SUPER TPII

- Gérer un module mouvant
 - T°c, RH%, tension batterie, vitesse moteur, ...
 - Infos GPS, suivi,
 - logger SD
 - distance du point de départ
- Avancé :
 - Nouveau écran d'affichage



SUPER TPII

- Gérer un module mouvant
 - Ajouter Altitude relative, pression
- Avancé :
 - écran d'affichage



SUPER TPII

- Gérer un module mouvant
 - Ajouter Accélération, Inclinaison X,Y,Z
- Avancé :
 - écran d'affichage



SUPER TP II

- Créer un serveur et un client rf ou nrf
 - Client
 - Capteurs véhicule
 - Serveur
 - Afficheur tft

