ARDUINO

De quoi avons-nous besoin pour programmer avec Arduino ?

Un ordinateur

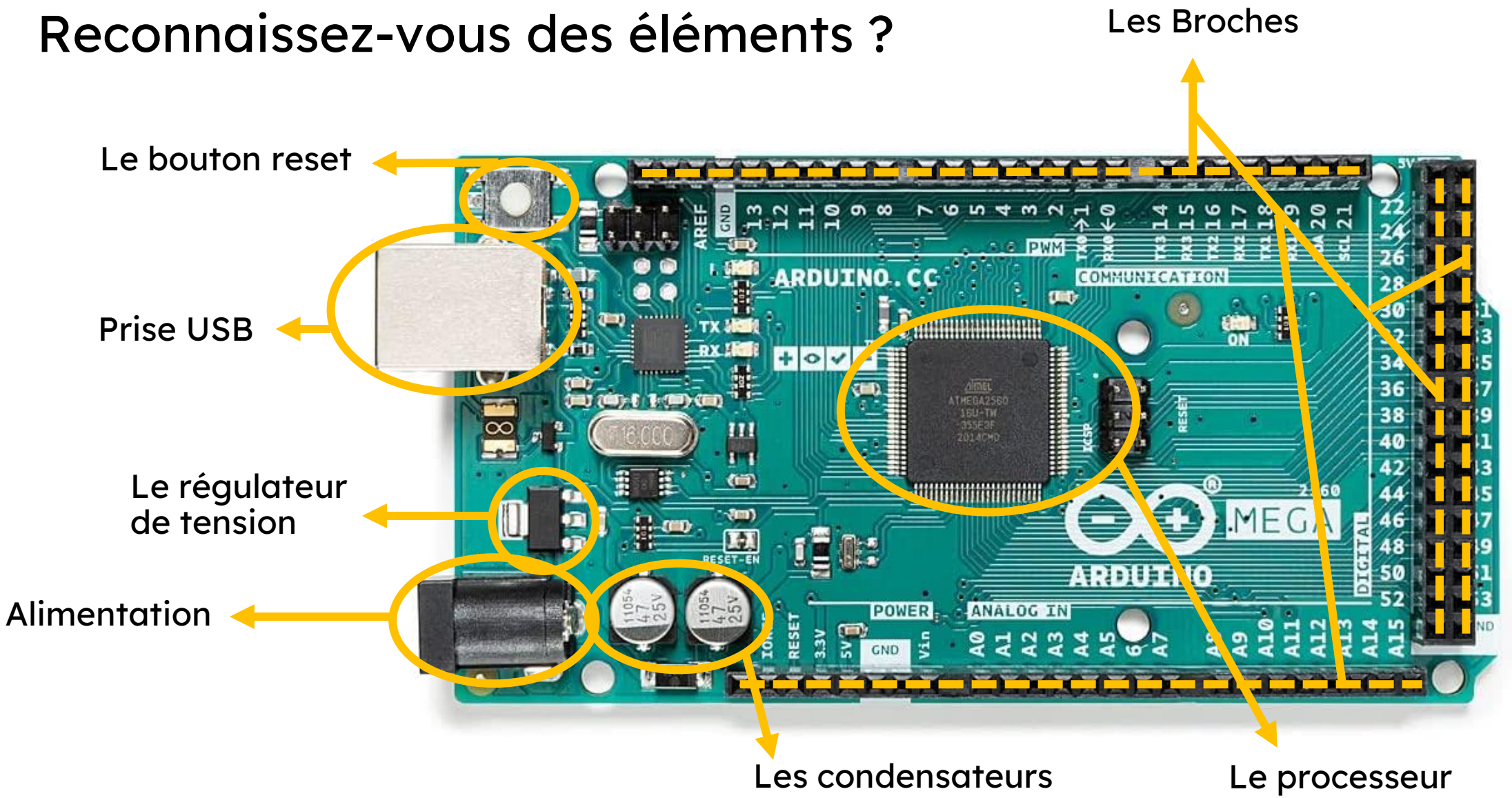
Un microcontrôleur

Quels sont les 4 principaux composants de la carte mère d'un ordinateur ?

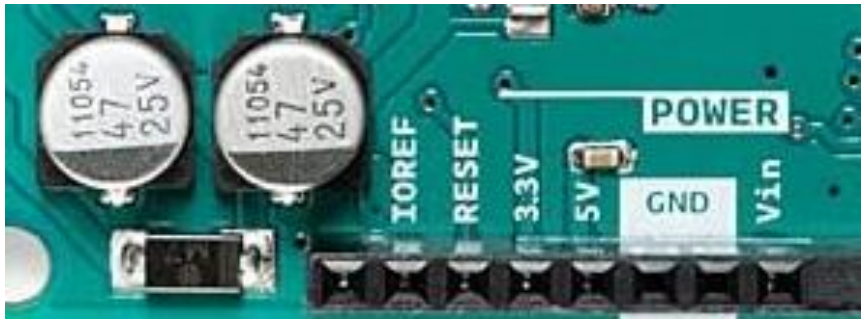
- Un processeur (la CPU) : pour traiter les données et faire les calculs.
- Un espace de stockage mémoire, pour les instructions à exécuter et les données qu'il manipule.
- Des ports d'entrée et de sortie pour communiquer avec l'extérieur.

La carte !

Reconnaissez-vous des éléments ?



Les broches POWER

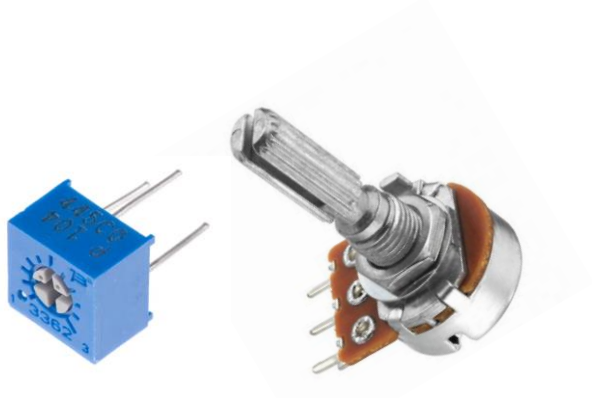


- La broche IOREF sert à distribuer à des cartes d'extension, la même tension que celle utilisée par la carte Arduino : soit 5v, soit 3,3v.
- La broche RESET reporte la fonction du bouton reset sur les cartes d'extension.
- La broche 3,3V sert à alimenter des composants avec cette tension réduite.
- La broche 5V (ou Vcc) sert à alimenter des composants en +5V.
- Les broches GND sont les pôles négatifs d'alimentation.
- Enfin, la broche VIN, sert à donner la même tension que celle en entrée, lorsque tu alimentes avec une pile ou un adaptateur secteur.

Les broches ANALOG

- Les broches ANALOG, sont les entrées analogiques. Elles sont reliées à un convertisseur analogique qui renvoie un code numérique.
- Contrairement aux broches DIGITAL que l'on va aborder juste après, le courant peut être progressif, de 0 à 5v.
- Elles sont généralement utilisées pour les éléments dont les mesures varient en continu. Des exemples ?

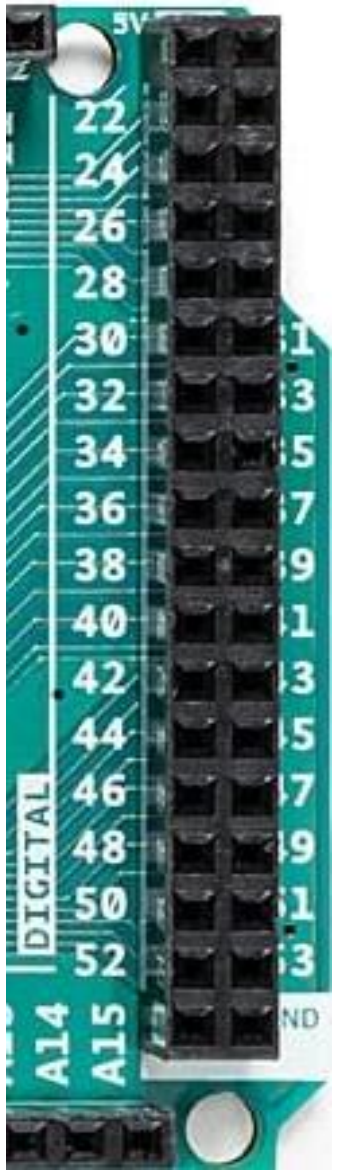
Un capteur US



Un potentiomètre

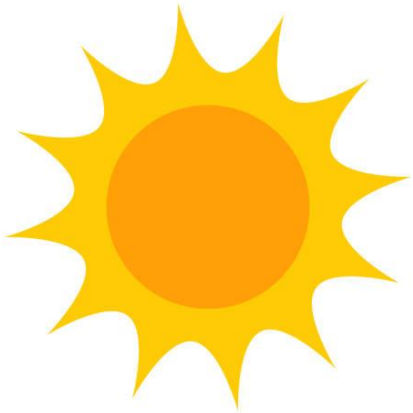
Les broches DIGITAL

- Les broches DIGITAL sont les broches numériques de sorties. C’est elles qui permettent d’envoyer les infos aux éléments extérieurs. C’est sur ces broches qu’il y aura ou pas une tension de 5 V en fonction des opérations à réaliser.



Broche DIGITAL et broche ANALOG ? Quelle différence concrète ?

- Lorsque tu allumes la lumière de ta chambre, tu utilises un interrupteur qui allume totalement la lumière, ou à l'inverse, l'éteint complètement. Ce phénomène est comparable au mécanisme numérique (broche digital). En revanche, tout au long de la journée, le soleil diffuse la lumière qui varie graduellement au cours de la journée. Ce phénomène lui est plutôt analogique (broche analog). Le soleil ne disparaît pas en un éclair au cours de la journée.



Les broches PWM

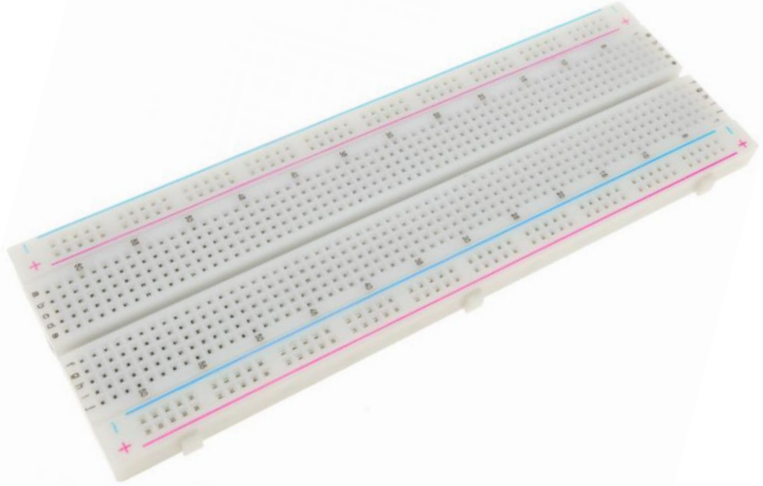
- Ces broches utilisent la technique PWM, qui permet de simuler une sortie analogique, avec une sortie numérique.
- Un ordinateur ne connaît à l'intérieur que des choses discontinues, numérique : soit il y a du courant, soit il n'y en a pas. Pourtant dans le monde réel, il y a des choses qu'on doit contrôler graduellement, de façon analogique, comme le volume sonore par exemple. C'est pour ça qu'il existe des sorties numériques pouvant être utilisées selon le principe PWM. Ce sont des broches « hybrides ».
- Lorsqu'elles sont utilisées comme entrées numériques uniquement, ces broches sont parfaitement identiques au broches DIGITAL.

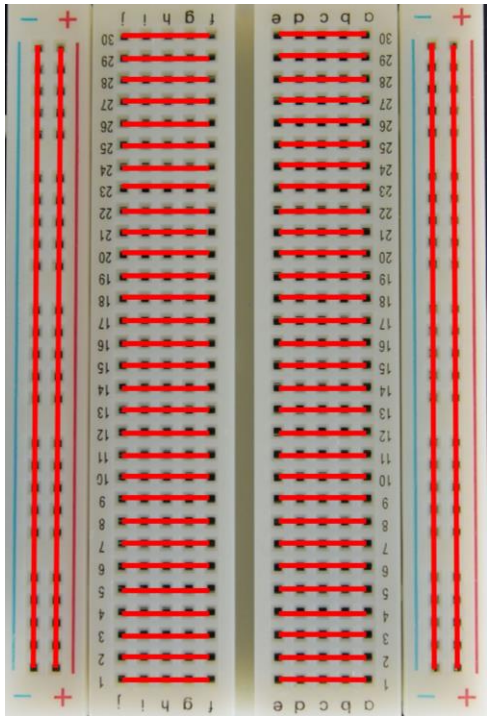
Les broches COMMUNICATION

- Les broches RX et TX servent à faire communiquer la carte avec l'ordinateur de la même manière que l'USB.
- Dans un projet, si tu as besoin d'afficher des éléments sur l'ordinateur, il ne faudra pas utiliser ces broches dans ton montage pour ne pas créer des conflits et générer des résultats bizarres.
- Ces broches sont reliées aux LEDs du même nom, qui s'allument au rythme de l'envoi (TX-transmit) et de la réception (RX-receive) de données sur la prise USB.
- Les broches SDA et SCL sont les broches de modules ICC (I2C).

Les plaques d’essai ou de prototypage

- C’est grâce à ces plaques que tu n’as pas besoin de fer à souder. Les broches et pattes des composants s’enfichent dans les trous qui sont interconnectés 5 par 5 dans chaque demi-rangée. Sur le schéma des traits rouges montrent comment se font les contacts par-dessous.





Programme ta carte Arduino

- Quand tu vas programmer ta carte avec Arduino, tu vas utiliser un langage de programmation qu'on appelle un infolangage.

- Quel est le langage le plus utilisé avec Arduino ?

  Le langage C

- Exercice :

  De la même manière que sur l'exemple, codez cette phrase : « Vous pratiquez une activité sportive tous les jours, à 18h. Lorsque vous arrivez, vous filez vous changer aux vestiaires, et vous partez vous échauffer.

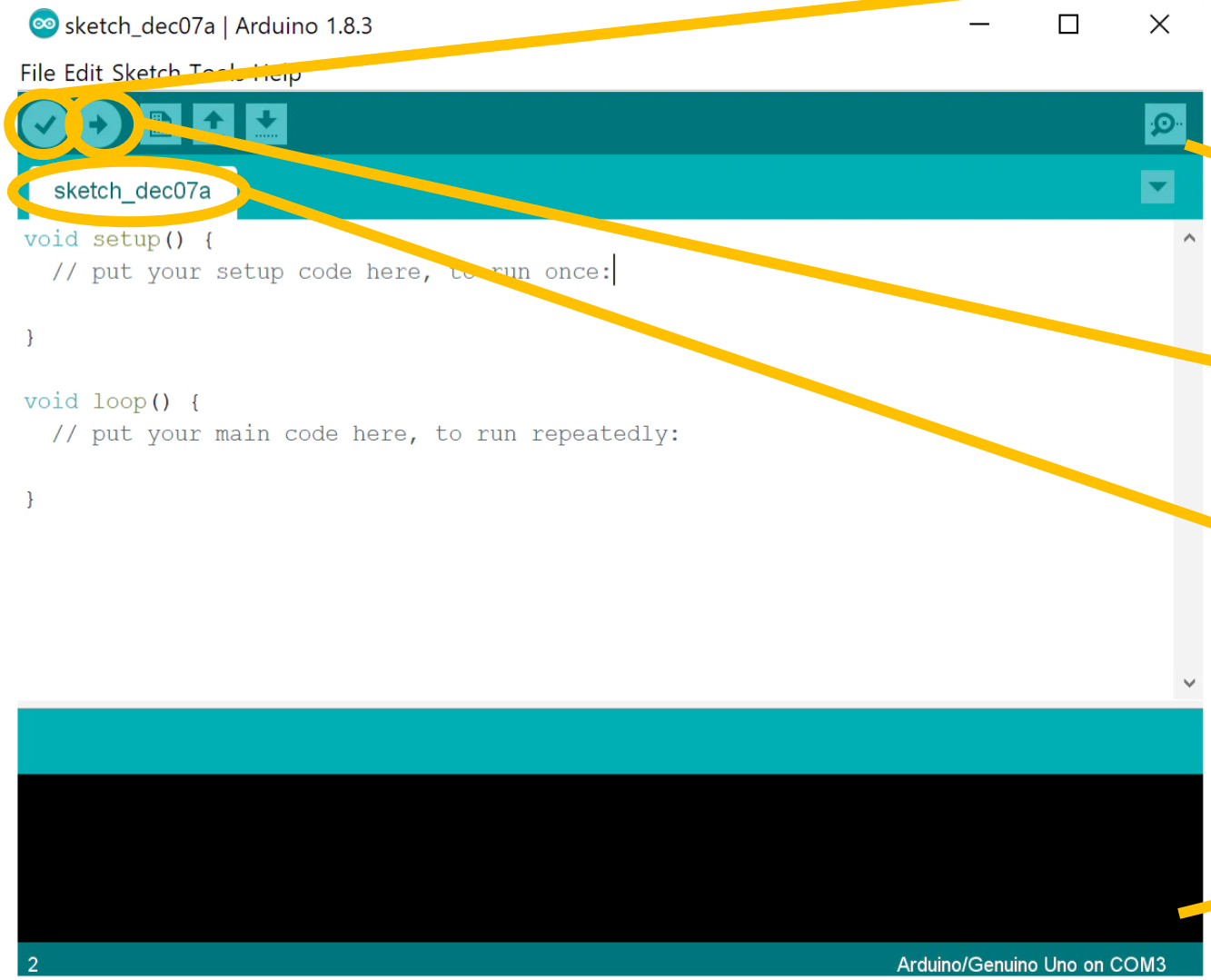
  Quel est le système qui analyse une à une chaque ligne de ton code source, de manière à le convertir en une série d'instructions machine ?

  Le compilateur

Tous les jours, je me lève à 7h, je m'habille puis je sors pour aller travailler :

```
if (heure == 7) {
  sHabiller();
  sortir();
}
```

L'IDE Arduino



Vérifier :
Tu le sais, les programmes que tu écris doivent être traduits en code exécutable avant d'être transférés vers la carte : la compilation. Cela s'opère lorsque tu appuies sur le bouton « vérifier » ou que tu téléverses.

Le moniteur série

Téléverser

Le nom du fichier : « sketch_dec07a »
sketch = croquis
dec07 = fait le 7 décembre
a = 1er programme aujourd'hui

Lorsque le programme est compilé, on nous indique à titre indicatif dans ce panneau, le nombre d'octets qu'utilise notre programme sur l'espace de stockage.

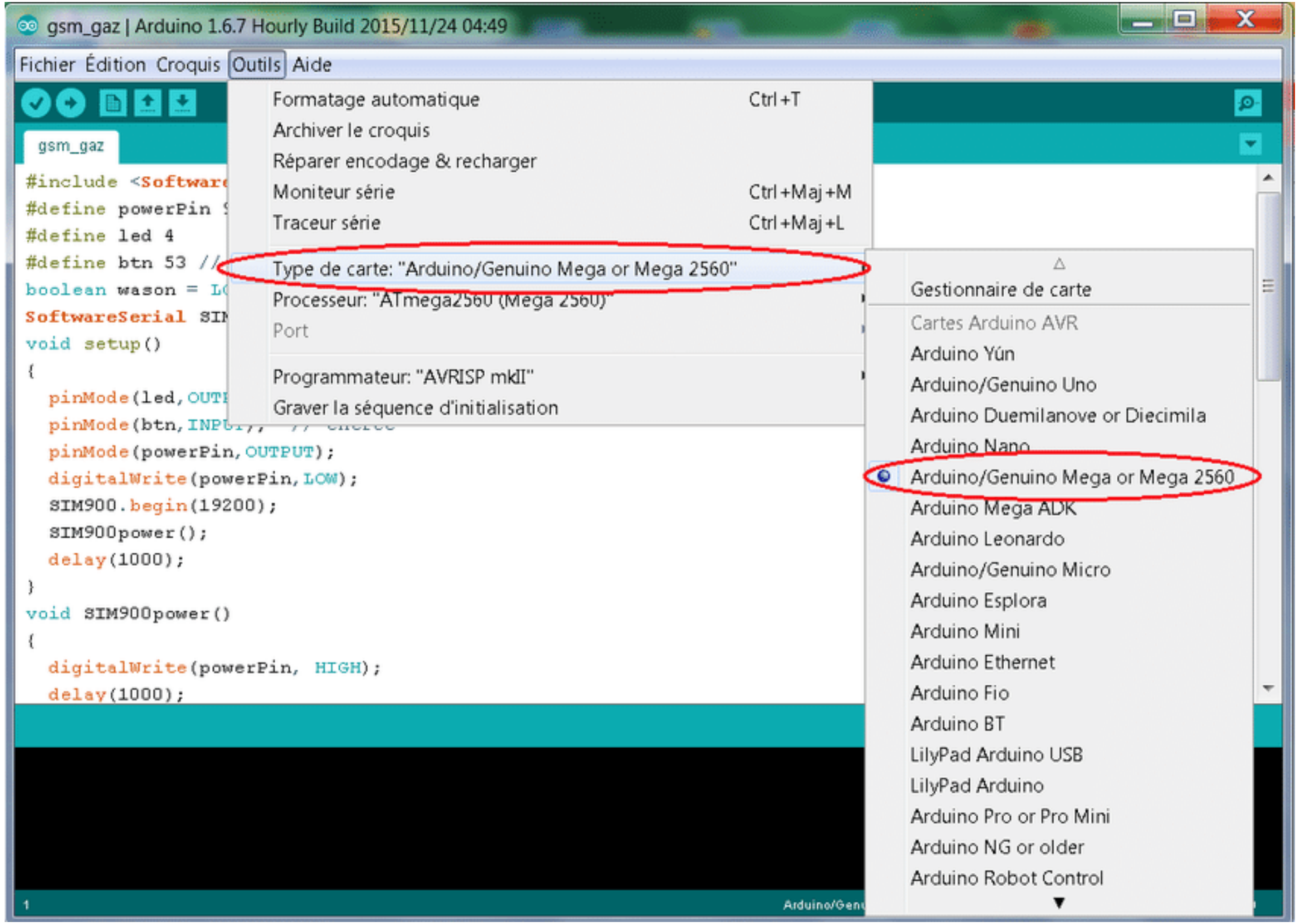
Attention !

- Si tu enregistres le programme et que tu le renommes, veille à ne pas utiliser de caractères spéciaux, majuscule et/ou d'accents. Évite aussi les espaces, et remplace-les plutôt par le signe underscore : « _ ». Cela permet d'éviter les problèmes de lecture de fichier !

  `Mon très long nom de dossier -> mon_tres_long_nom_de_dossier`

- Prenez garde à votre syntaxe ! Une mauvaise orthographe ; un point-virgule, une accolade, ou une parenthèse oubliée ; une ponctuation ou un signe mal utilisé ; et encore plein d'autres erreurs courantes, et votre programme ne sera pas compilable !

- Avant de téléverser un programme, vous devez impérativement configurer la carte sur l'IDE. C'est-à-dire que vous devez définir deux paramètres :
  - le type de carte
  - le port de communication


gsm_gaz | Arduino 1.6.7 Hourly Build 2015/11/24 04:49
Fichier Édition Croquis Outils Aide
Formatage automatique Ctrl+T
Archiver le croquis
Réparer encodage & recharger
Moniteur série Ctrl+Maj+M
Traceur série Ctrl+Maj+L
Type de carte: "Arduino/Genuino Mega or Mega 2560"
Processeur: "ATmega2560 (Mega 2560)"
Port
Programmateur: "AVRISP mkII"
Graver la séquence d'initialisation
Gestionnaire de carte
Cartes Arduino AVR
Arduino Yún
Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino/Genuino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino/Genuino Micro
Arduino Esplora
Arduino Mini
Arduino Ethernet
Arduino Fio
Arduino BT
LilyPad Arduino USB
LilyPad Arduino
Arduino Pro or Pro Mini
Arduino NG or older
Arduino Robot Control
gsm_gaz
#include <Software
#define powerPin
#define led 4
#define btn 53 //
boolean wason = L
SoftwareSerial SI
void setup()
{
pinMode(led,OUT
pinMode(btn,INPU
pinMode(powerPin,OUTPUT);
digitalWrite(powerPin,LOW);
SIM900.begin(19200);
SIM900power();
delay(1000);
}
void SIM900power()
{
digitalWrite(powerPin, HIGH);
delay(1000);

Le void setup et le void loop ?

- Dans le « void setup », viennent les instructions de préparation.
  En voici 3 principales :
  - On définit les broches et le port en entrée (input), ou sortie (output).
  - On définit la vitesse de dialogue avec l'ordinateur.
  - On prépare les variables.

  Ces instructions ne sont lues qu'une seule fois.

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

- Dans le void loop, viennent les instructions qui doivent être répétées. Toutes les instructions que tu vas ajouter à l'intérieur du bloc de cette fonction, seront exécutés de la première à la dernière, puis ainsi de suite en boucle, jusqu'à que tu coupe l'alimentation ou que tu appuies sur le bouton reset de la carte.

Un programme utilisateur Arduino est une suite d'instructions élémentaires sous forme textuelle, ligne par ligne.
La carte lit puis effectue les instructions les unes après les autres, dans l'ordre défini par les lignes de code.

Structure d'un programme

Il y a trois phases consécutives:

Commentaires multilignes pour se souvenir du patch ==>

1/La définition des constantes et des variables

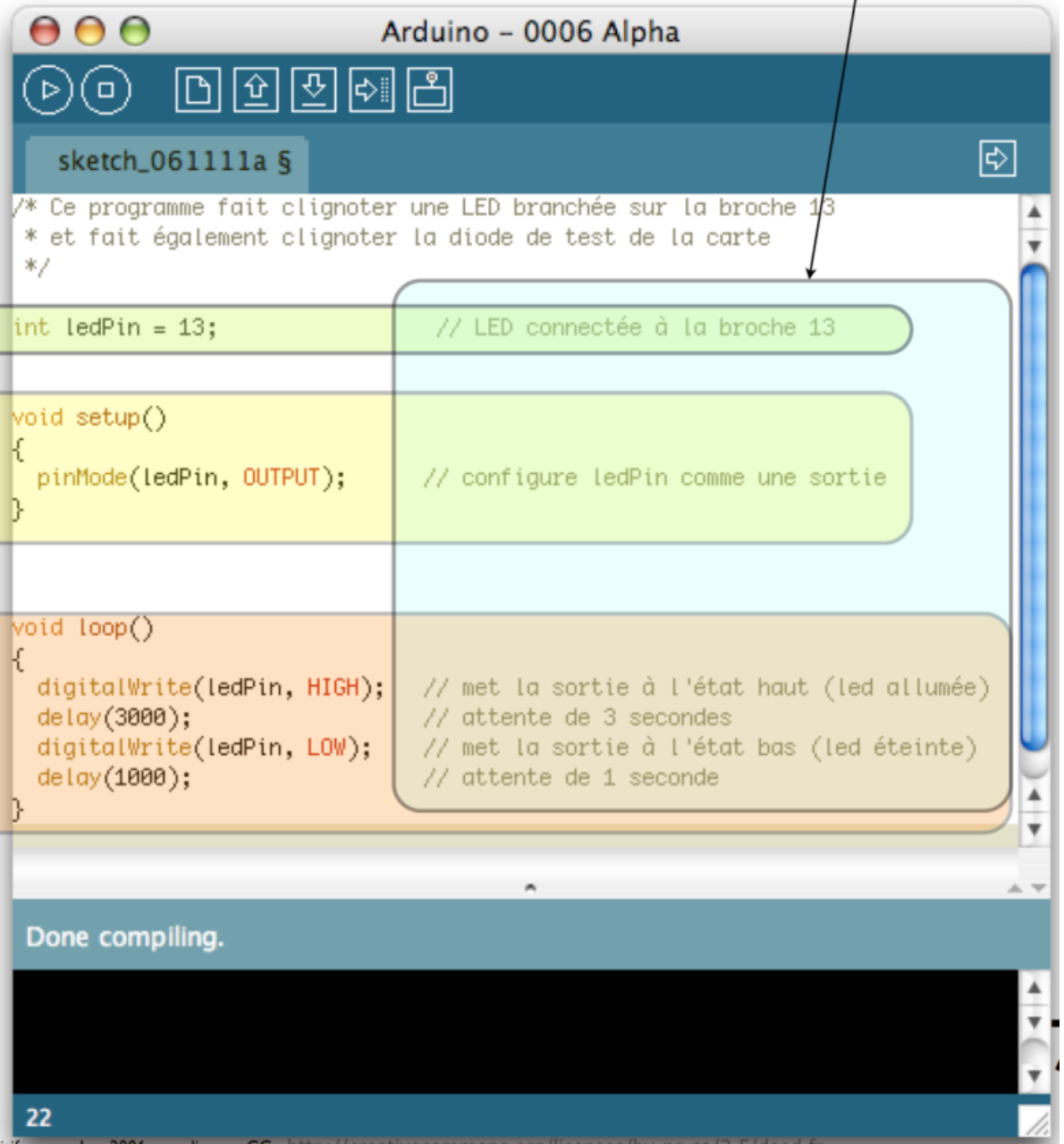
2/La configuration des entrées et sorties

void setup()

3/La programmation des interactions et comportements

void loop()

Commentaires



Une fois la dernière ligne exécutée, la carte revient au début de la troisième phase et recommence sa lecture et son exécution des instructions successives. Et ainsi de suite.

Cette **boucle** se déroule des milliers de fois par seconde et anime la carte.

Les structures de programmation

```
while(condition){};
```

La structure WHILE permet au programme d'être lu en boucle jusqu'à que la condition énoncée soit fausse.

```
if (condition) {
    // instruction(s) exécutée(s) si la condition est vraie
}
else {
    // instruction(s) exécutée(s) si la condition est fausse
}
```

La structure SI/SINON permet d'exécuter un bloc d'instruction si une condition logique est vraie, ou d'exécuter un autre bloc d'instruction si cette dernière est fausse. Cette deuxième partie est facultative.

```
for ( initialisation ; condition ; incrementation ) {
    // instructions de la boucle
}
```

La structure FOR est adaptée à l'exécution de toute action répétitive. Le but étant donc d'initialiser une variable à une certaine valeur, la faire évoluer d'un pas fixe à chaque tour pour sortir de la boucle une fois une condition atteinte, voila la syntaxe fournie par le site officiel Arduino. Exemple :

```
for(int compteur = 0 ; compteur < 10 ; compteur++)
{
                // fonction à répéter
}
```

Pour plus de précision : https://www.arduino.cc/reference/fr/

Les structures

Sketch

loop()
setup()

Structures de Contrôle

break
continue
do...while
else
for
goto
if
return
switch...case
while

Autre Syntaxe

#define (define)
#include (include)
/* */ (block comment)
// (single line comment)
; (semicolon)
{} (curly braces)

Opérateurs Arithmétiques

% (remainder)
* (multiplication)
+ (addition)
- (subtraction)
/ (division)
= (assignment operator)

Opérateurs de Comparaison

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

Opérateurs Booléens

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

Opérateurs d'accès aux Pointeurs

& (reference operator)
* (dereference operator)

Opérateurs Bit à Bit

& (bitwise and)
<< (bitshift left)
>> (bitshift right)
^ (bitwise xor)
| (bitwise or)
~ (bitwise not)

Opérateurs Composés

%= (compound remainder)
&= (compound bitwise and)
*= (compound multiplication)
++ (increment)
+= (compound addition)
-- (decrement)
-= (compound subtraction)
/= (compound division)
^= (compound bitwise xor)
|= (compound bitwise or)

Constantes

HIGH | LOW

INPUT | OUTPUT | INPUT_PULLUP

LED_BUILTIN

true | false

Floating Point Constants

Integer Constants

Conversion

(unsigned int)

(unsigned long)

byte()

char()

float()

int()

long()

word()

Types de Données

array

bool

boolean

byte

char

double

float

int

long

short

size_t

string

String()

unsigned char

unsigned int

unsigned long

void

word

Portée & Qualifieurs des Variables

const

scope

static

volatile

Utilitaires

PROGMEM

sizeof()

Les variables

E/S Numériques

digitalRead()
digitalWrite()
pinMode()

E/S Analogiques

analogRead()
analogReference()
analogWrite()

Famille Zero, Due & MKR

analogReadResolution()
analogWriteResolution()

E/S Avancées

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

Temps

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

Mathématiques

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

Trigonométrie

cos()
sin()
tan()

Caractères

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

Nombres Aléatoires

random()
randomSeed()

Bits et Octets

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

Interruptions Extern

attachInterrupt()
detachInterrupt()

Interruptions

interrupts()
noInterrupts()

Communication

Serial
Stream

USB

Keyboard
Mouse

Les fonctions