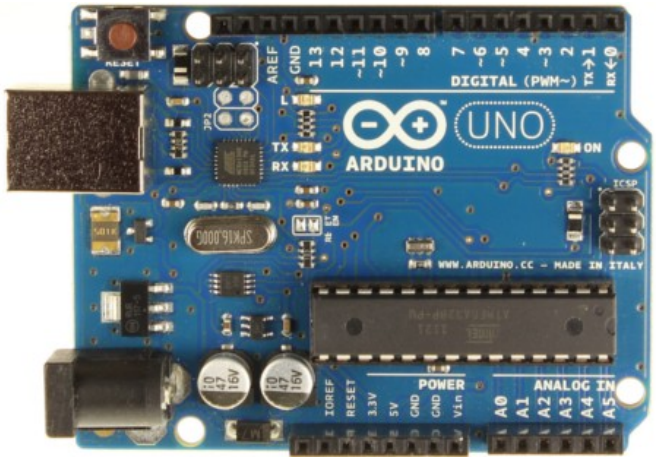
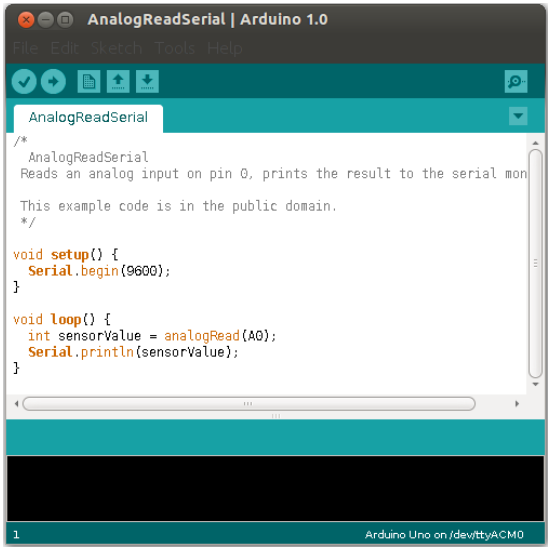
L'Arduino

Qu'est-ce que l'Arduino ?

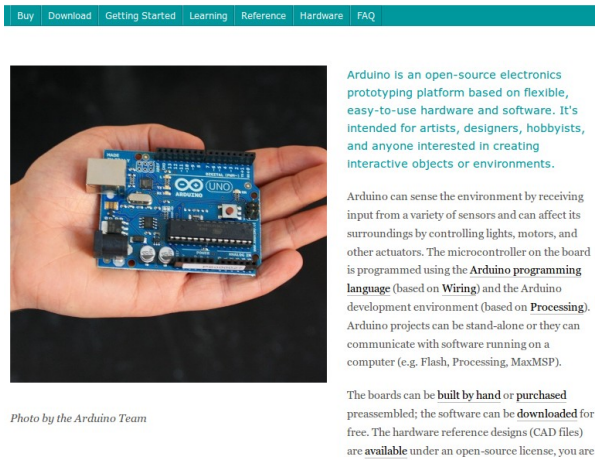
Le système **Arduino** est une plate-forme open-source d'électronique programmée qui est basée sur une simple carte à microcontrôleur (de la famille AVR), et un logiciel, véritable environnement de développement intégré, pour écrire, compiler et transférer le programme vers la carte.



Une carte électronique



Un environnement graphique



Une communauté qui échange

Le logiciel Arduino

Le logiciel de programmation des modules Arduino est une application Java, libre et multiplate-formes, servant d'**éditeur de code** et de **compilateur,** et qui peut **transférer** le firmware et le programme au travers de la liaison série (RS-232, Bluetooth ou USB selon le module).

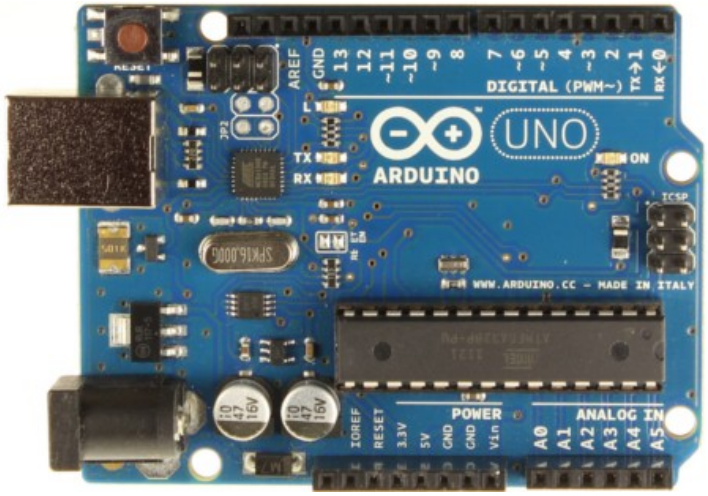
Le **langage de programmation** utilisé est le **C**++, langage standard, ce qui rend aisé le développement de programmes sur les plates-formes Arduino. De nombreuses bibliothèques de fonctions sont fournies pour la mise en œuvre des différentes fonctionnalités de la carte (CAN, PWM, liaison I^2C, ...).

La carte électronique

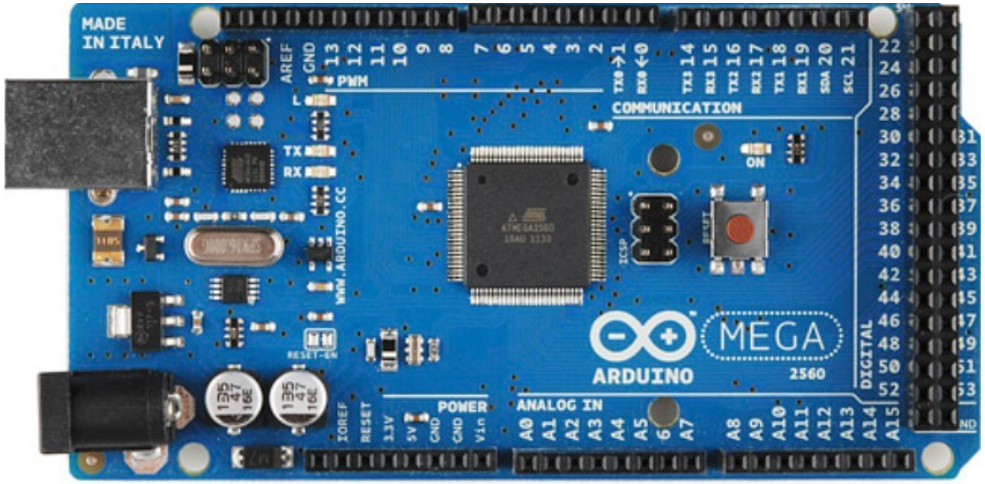
Il existe plusieurs versions de la carte Arduino, toutes **« open source »** : les schémas sont téléchargeables, on peut les copier, les modifier librement, fabriquer la carte.
Les cartes Arduino sont relativement peu coûteuses, la moins chère des versions du module Arduino peut être assemblée à la main, et même les cartes Arduino pré-assemblées coûtent moins de 25 Euros (microcontrôleur inclus...) !!!
Les cartes Arduino contiennent tout ce qui est nécessaire pour le fonctionnement du microcontrôleur; Pour pouvoir l'utiliser et se lancer, il suffit simplement de connecter la carte à un ordinateur à l'aide d'un câble USB.
Les cartes Arduino sont basées sur les microcontrôleurs Atmel ATMEGA8, ATMEGA168, ATMEGA 328, etc...

Exemples de cartes Arduino

Arduino Uno



Arduino Mega

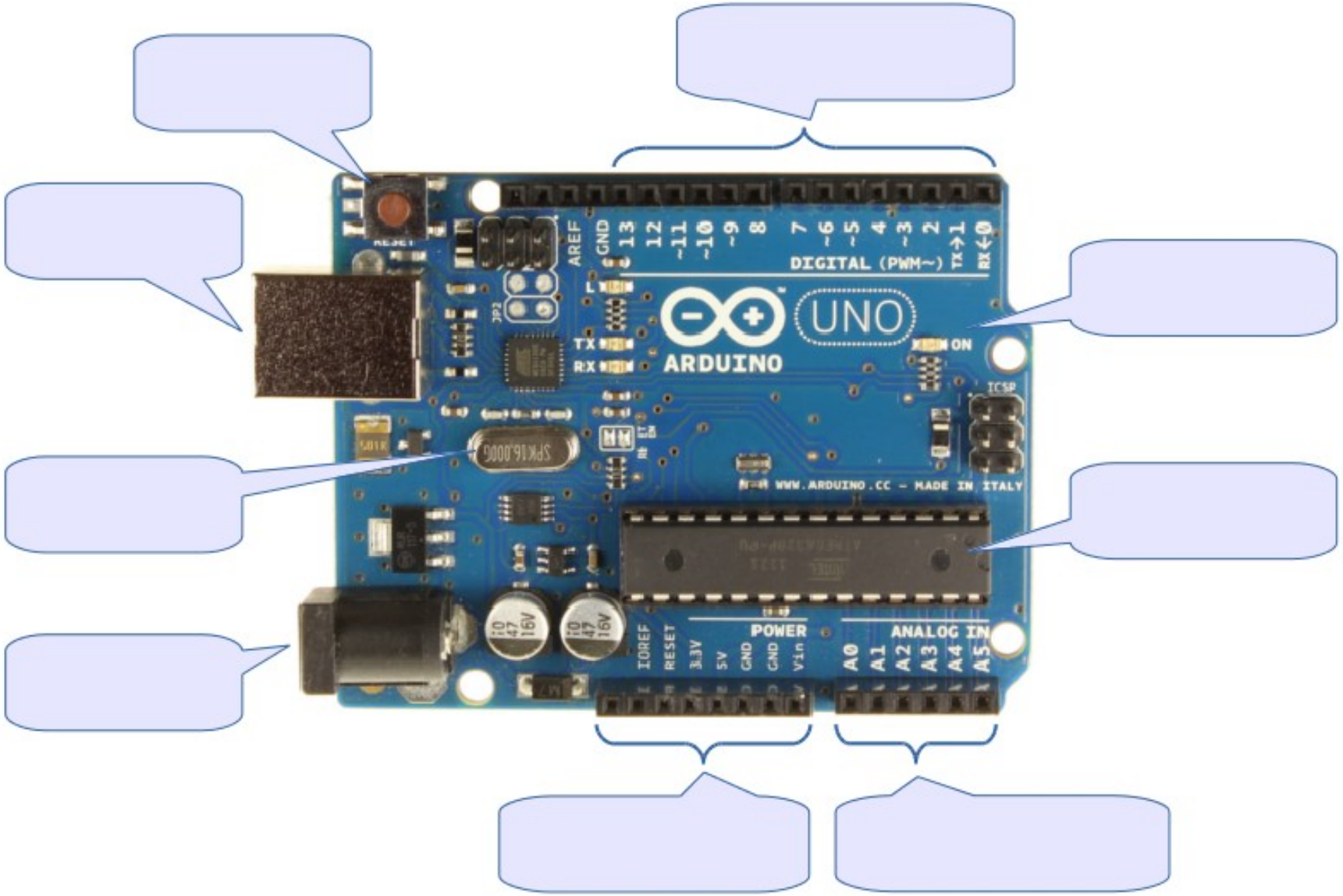


Caractéristiques de la carte Arduino Uno

Microcontrôleur	*ATmega328 (8 bits)*
Tension de fonctionnement	*5 V*
Tension d'alimentation (recommandée)	*7- 12 V*
Tension d'alimentation (limites)	*6 - 20V*
Nombre d'E/S	*14 (dont 6 pouvant générer des signaux PWM)*
Nb ports "Analogique/Numérique"	*6*
Courant max. par E/S	*40 mA sous +5V (ATTENTION : 200mA cumulé pour l'ensemble des broches E/S)*
Courant pour broches 3.3 V	*50 mA*
Intensité maxi disponible pour la sortie 5V	*Fonction de l'alimentation utilisée - 500 mA max si port USB utilisé seul*
Mémoire Flash	*32 Ko (ATmega328) dont 0.5 Ko utilisé par le bootloader*
SRAM	*2 Ko (ATmega328)*
EEPROM	*1 Ko (ATmega328)*
Vitesse horloge	*16 MHz*

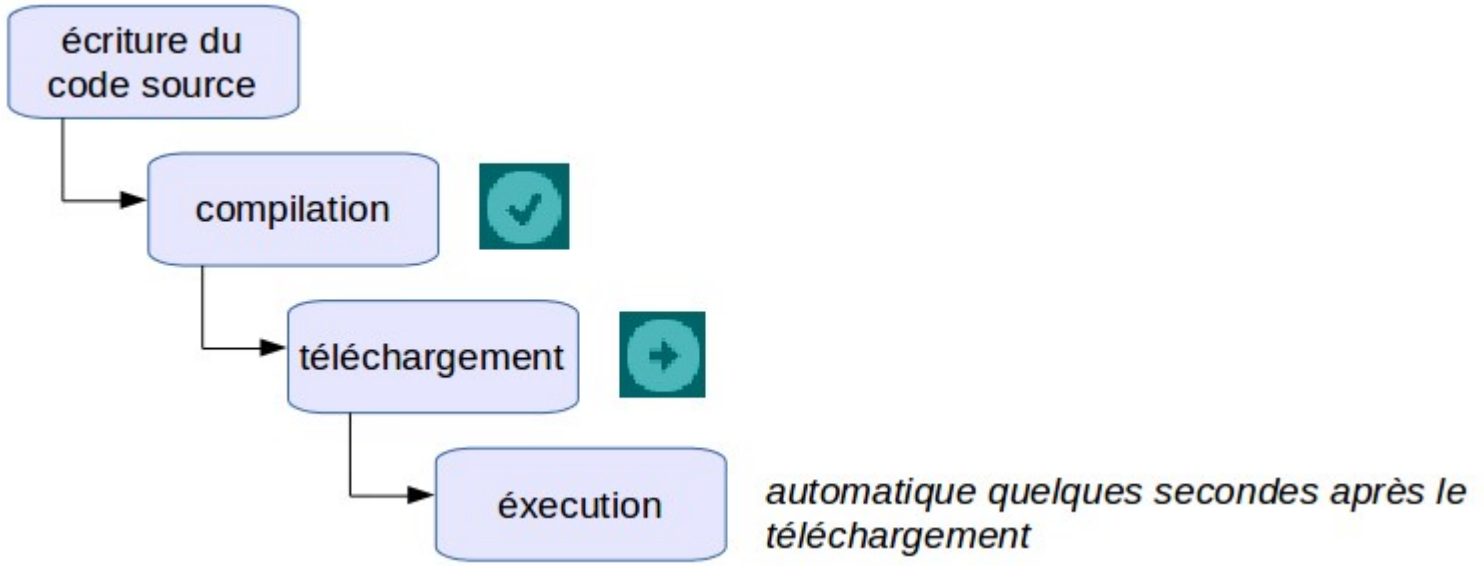
L'Arduino

Ex : nommer sur la carte Arduino Uno R3 les éléments suivants



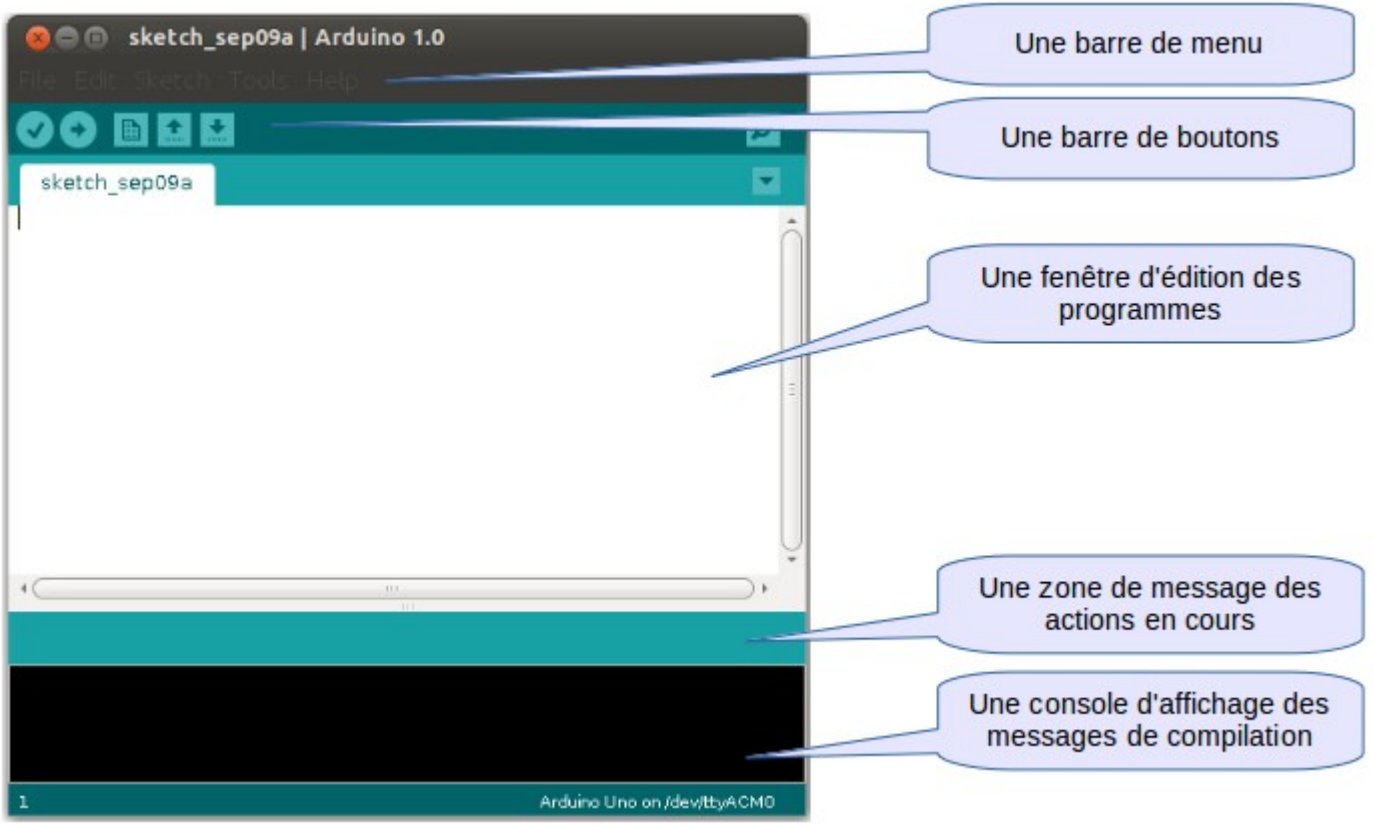
Le logiciel de programmation

Le logiciel Arduino a pour fonctions principales :
- de pouvoir écrire et compiler des programmes pour la carte Arduino ;
- de se connecter avec la carte Arduino pour y transférer les programmes ;
- de communiquer avec la carte Arduino.



Le logiciel de programmation

Le logiciel est un Environnement de Développement Intégré (*IDE*) dédié au langage Arduino et à la programmation des cartes Arduino. Il comporte :



Le logiciel de programmation

Le code écrit avec le logiciel Arduino est appelé un programme, ou une séquence (***sketch*** en anglais). Le logiciel Arduino intègre le concept d'un "***sketchbook***" (livre de programme) : un endroit réservé pour stocker vos programmes.
Les programmes que vous mettez dans votre "*sketchbook*" pourront être ouvert directement depuis le menu **File > Sketchbook** ou à l'aide du bouton **Open** (Ouvrir) dans la barre d'outils. La première fois que vous démarrer le logiciel Arduino, un chemin automatique sera créé pour votre "*sketchbook*". Vous pouvez voir ou modifier cette localisation depuis le menu **File > Preferences**.

Le langage de programmation

Le langage Arduino est basé sur les **langages C/C++**.

Avec Arduino, nous devons utiliser un code minimal lorsque l'on crée un programme. Ce code permet de diviser le programme que nous allons créer en deux grosses parties.

```
void setup()                //fonction d'initialisation de la carte
{
                            //contenu de l'initialisation
}

void loop()                 //fonction principale, elle se répète (s'exécute) à l'infini
{
                            //contenu de votre programme
}
```

Le langage de programmation

Exécutée une seule fois au démarrage du programme et après chaque reset

S'exécute en boucle sans fin

```
void setup()                //fonction d'initialisation de la carte
{
                            //contenu de l'initialisation
}

void loop()                 //fonction principale, elle se répète (s'exécute) à l'infini
{
                            //contenu de votre programme
}
```

La syntaxe de base

- **Chaque instruction se termine par un « ; »** ;
- Les accolades « {« et « } » sont les "conteneurs" du code du programme. Elles contiennent les fonctions, conditions ou boucles ;
- Les commentaires sont des lignes de texte incluses dans le programme et qui ont pour but de vous informer vous-même ou les autres de la façon dont le programme fonctionne. Ces lignes ajoutées sont ignorées par le compilateur. **Les commentaires sont précédés des caractères « // » ou bien encadrés par « /* » et « */ »** ;
- Il est formellement **interdit de mettre des accents** en programmation, sauf dans les commentaires.
- Un nombre en binaire doit être précédé de la lettre « B » ;
- Un nombre écrit en hexadécimal doit être précédé par les caractères « 0x ».

Les constantes

Constante	Description
HIGH	niveau haut logique
LOW	niveau bas logique
INPUT	entrée
OUTPUT	sortie
TRUE	vrai
FALSE	faux

Les variables

Une variable est un **nom que vous donnez à un emplacement en mémoire RAM dans lequel vous stockerez des données**.
Une variable est définie par son **nom** et son **type**.

Type	Taille en mémoire	Type de données	signe	Valeurs min/max
boolean	1 octet (8 bits)	Valeur binaire 0 ou 1	non signée	0 / 1
int	2 octets (16 bits)	Valeur entière	signée	-32 768 / +32 767
long	4 octets (32 bits)	Valeur entière	signée	-2 147 483 648 / +2 147 483 647
byte	1 octet (8 bits)	Valeur entière	non signée	0 / +255
unsigned int	2 octets (16 bits)	Valeur entière	non signée	0 / +65535
word	2 octets (16 bits)	Valeur entière	non signée	0 / +65535
unsigned long	4 octets (32 bits)	Valeur entière	non signée	0/ +4 294 967 295
float	4 octets (32 bits)	Valeur à virgule	signée	-3.4028235E+38 / +3.4028235E+38
double	4 octets (32 bits)	Valeur à virgule	signée	-3.4028235E+38 / +3.4028235E+38
char	1 octet (8 bits)	Valeur entière - Code ASCII	signée	-128 / +127

Ex : Quel est le type approprié pour les variables suivantes : la lecture d'un bouton poussoir, la variable seconde dans une montre, le montant de la prochaine cagnotte de l'Euro-millions.

Les opérations simples

- Les opérateurs mathématiques :

=	égalité
+	addition
-	soustraction
*	multiplication
/	division
%	modulo

Les opérations simples

- Les opérateurs de comparaison :

==	égal à
!=	différend de
<	Inférieur à
>	supérieur à
<=	inférieur ou égal à
>=	supérieur ou égal à

Les opérations simples

- Les opérateurs booléens :

&&	ET booléen
||	OU booléen
!	NON booléen

Les instructions conditionnelles

algorithme	Programmation en langage Arduino	Exemple
si *condition* **alors** *action* **fin si**	if (…) { … ; }	if (x==2) { y=x; }
si *condition* **alors** *action 1* **sinon** *action 2* **fin si**	if (…) { … ; } else { … ; }	if (x==2) { y=x; } else { y=x+1; }

algorithme	Programmation en langage Arduino	Exemple
	switch (var) { // début de la structure case 1 : … // cas 1 break; // l'instruction break est en option case 2 : … // cas 2 break; // l'instruction break est en option default: // cas par défaut (non -obligatoire) }	switch (entree) { case 1 : a=10;break; case 2 : a=15;break; case 3 : a=20;break; default : a=0; }

Ex : traduire l'algorithme en langage Arduino

algorithme	Programmation en langage Arduino
début de la fonction principale déclarer variable a de type byte a = 2 si a>5 alors a = 20 sinon a = 10 fin si fin de la fonction principale	

Ex : quel est le résultat des instructions suivantes pour x=4, y=0 et z=0 ?

programme 1	programme 2
if (x == 4) { y = 1;} else { y = 2; z = 3;}	{ if (x == 4) { y = 1;} else { y = 2;}} z = 3;

Les boucles

algorithme	Programmation en langage Arduino	
tant que *condition* **faire** *action* **fin tant que**	while (...) { ... ; }	while (x<10) { x++; }
répéter *action* **tant que** *condition*	do { ... ; } while (...) ;	do { x++; } while (x<10);
pour indice de Vi à Vf par pas de x **faire** *action* **fin pour**	for (indice = Vi ; indice <Vf ; indice = indice + x) { ... ; }	for (i=0;i<10;i++) { x=x+2 ; }

Ex : Quelle boucle est adaptée à l'écriture de programmes traitant les problèmes suivants :

- le calcul du total à payer à une caisse enregistreuse ;
- la recherche du jour le plus pluvieux d'une année ;
- l'attente de l'appui sur un bouton poussoir connecté à une entrée de l'Arduino.

Les fonctions prédéfinies

Fonctions	Description	Paramètres
pinMode(broche, mode)	Configure la broche spécifiée pour qu'elle se comporte soit en entrée, soit en sortie.	**broche** : le numéro de la broche de la carte Arduino dont le mode de fonctionnement (entrée ou sortie) doit être défini. Pour les entrées analogiques on utilise les références A0, A1 etc., à la place du simple numéro de la broche. **mode** : INPUT ou OUTPUT
digitalWrite(broche, valeur)	Met un niveau logique HAUT ou BAS sur une broche numérique.	**broche** : le numéro de la broche de la carte Arduino **valeur** : HIGH ou LOW (ou bien 1 ou 0)
int digitalRead(broche)	Lit le niveau logique d'une broche en entrée numérique, et renvoie la valeur HIGH ou LOW.	**broche** : le numéro de la broche numérique que vous voulez lire
int analogRead(broche)	Lit la valeur de la tension présente sur la broche spécifiée. La carte Arduino Uno comporte 6 voies analogiques connectées à un convertisseur analogique-numérique 10 bits. Cela signifie qu'il est possible de transformer la tension d'entrée entre 0 et 5V en une valeur numérique entière comprise entre 0 et 1023.	**broche** : le numéro de la broche analogique sur laquelle il faut convertir la tension analogique appliquée (0 à 5 sur la carte Arduino Uno).
analogWrite(broche, valeur)	Génère une impulsion de largeur / période voulue sur une broche de la carte Arduino (PWM - Pulse Width Modulation en anglais ou MLI - Modulation de Largeur d'Impulsion en français). Ceci peut-être utilisé pour faire briller une LED avec une luminosité variable ou contrôler un moteur à des vitesses variables.	**broche** : la broche utilisée pour "écrire" l'impulsion. Cette broche devra être une broche ayant la fonction PWM, Par exemple, sur la UNO, ce pourra être une des broches 3, 5 ,6 ,9 ,10 ou 11. **valeur** : la largeur du "duty cycle" (proportion de l'onde carrée qui est au niveau HAUT) : entre 0 (0% HAUT donc toujours au niveau BAS) et 255 (100% HAUT donc toujours au niveau HAUT).

Ex : écrire un programme complet qui configure la broche 13 en sortie et place un niveau HAUT sur cette sortie afin d'allumer la led qui y est connectée.

Ex : écrire un programme complet qui allume cette même led si un bouton poussoir connecté sur la broche 12 est appuyé, l'éteint sinon.