.:Quelques Bases de Programmation:.

La Programmation Arduino en Bref

Arduino est programmé en langage C. Ceci est un petit cours à destination des personnes ayant déjà une petite expérience de la programmation et qui ont besoin d'un petit briefing sur la syntaxe du C et sur l'environnement de développement (IDE) Arduino.

Structure

Chaque programme Arduino (que l'on nomme aussi sketch) requiert 2 fonctions (aussi appelées procédures).

`void setup(){ }`
Tout le code entre les deux accolades sera exécuté une fois au démarrage du programme sur votre Arduino.

`void loop(){ }`
Cette fonction s'exécute après la fin du setup. Après une première exécution, il est ré-exécuté puis encore et encore jusqu'à ce que l'alimentation soit coupée.

Syntaxe

L'un des éléments qui peut paraitre frustrant en C est son formatage nécessaire (cela le rend aussi très puissant). Si vous vous souvenez de ce qui suit vous devriez vous en sortir.

// (commente une seule ligne)
Il est souvent très utile d'écrire des notes pour soi-même au fur et à mesure de l'élaboration de votre code. Pour ce faire inscrivez deux slashs et tout ce que vous inscrirez après sera ignoré par votre programme.

/* */ (commente plusieurs lignes)
Si vous avez beaucoup à écrire, vous pouvez commenter plusieurs lignes. Tout ce qui se trouve entre ces deux symboles sera ignoré par le programme.

{ } (accolades)
On les utilise pour définir où commence et où se termine un bloc de code (utilisé dans les fonctions aussi bien que dans les boucles).

; (point-virgule)
Chaque ligne doit se terminer par un point-virgule (l'oubli d'un point-virgule est souvent la raison d'un refus du programme de compiler).

Variables

Un programme n'est rien de plus que des instructions qui manipulent des nombres de façon intelligente. Les variables sont utilisées pour stocker les valeurs.

int (entier)
Le type le plus utilisé, il contient un nombre sur deux octets (16 bits). il n'a pas de virgule et stocke des valeurs comprises entre -32 768 et 32 767.

long (long)
Variables utilisées lorsqu'un entier n'est pas assez grand. Ils prennent 4 octets (32 bits) de RAM et prennent des valeurs comprises entre -2 147 483 648 et 2 147 483 647.

boolean (booléen)
Variables qui prennent simplement les valeurs Vrai ou Faux (True et False). Très utiles, car elles ne prennent que peu de place (8 bits).

float (flottant)
Utilisés pour les variables flottantes. Ils prennent 4 octets (32 bits) de RAM et sont compris entre -3.4028235E+38 et 3.4028235E+38.

char (caractère)
Stocke un caractère en utilisant la table ASCII (exemple 'A' =65). Utilise un octet (8 bits) de RAM. Arduino stocke les chaines en tant que tableau de char.

.:Pour une description complète du langage :.
.:veuillez aller sur:.
http://ardx.org/PROG

03 PROG
Bases
de Programmation

Operateurs mathématiques

Opérateurs utilisés pour effectuer des opérations mathématiques (ils fonctionnent simplement comme en mathématiques)

- **=** (égalité) rend quelque chose égal à un autre (ex : x = 10 * 2 (x est maintenant égal à 20))
- **%** (modulo) donne le reste de la division d'un nombre par un autre (ex 12 %10 (donne 2))
- **+** (addition)
- **-** (soustraction)
- ***** (multiplication)
- **/** (division)

Comparateurs

Ces opérateurs sont utilisés pour les comparaisons logiques

- **==** (égal à) (ex : 12 == 10 est Faux et 12 == 12 est Vrai)
- **!=** (différent de) (ex : 12 != 10 est Vrai et 12 != 12 est Faux)
- **<** (inferieur à) (ex : 12 <10 est Faux et 12 < 12 est Faux et 12 < 14 est Vrai)
- **>** (supérieur à) (ex : 12 >10 est Vrai et 12>12 est Faux et 12>14 est Faux)

Structure de contrôle

Les programmes sont charges de gérer l'enchainement des instructions. Voici les éléments de contrôle basiques (vous en découvrirez beaucoup plus sur Internet).

```
if(condition){ }
else if( condition ){ }
else { }
```

Le code entre accolades sera exécuté si la condition if (si) est vrai sinon la condition else if (sinon si) sera testée et si le résultat est encore faux le code else (sinon) sera exécuté.

```
for(int i = 0; i <
#répétitions; i++){   }
```

Utilisé quand vous voulez répéter un morceau de code un certain nombre de fois (il est possible de compter i++ et de décompter i-- ou encore d'utiliser n'importe quelle variable).

Numérique

```
pinMode(patte, mode);
```

Utilisé pour changer le mode d'une patte, patte est le numéro de la patte sur laquelle vous voulez agir (0-19, les analogique 0-5 correspondent aux 14-19). Le mode peut être INPUT (entrée) et OUTPUT (sortie).

```
digitalWrite(patte, value);
```

Lorsqu'une patte est mise en OUTPUT, il est possible de changer son état. L'état peut être HIGH (mis à +5V) ou LOW (mis à la masse).

```
int digitalRead(patte);
```

Lorsqu'une patte est mise en INPUT vous pouvez utiliser cette fonction qui retourne son état, HIGH lorsqu'elle est mise à +5V et LOW lorsqu'elle est mise à la masse.

Analog

L'Arduino est une carte numérique mais elle peut aussi agir dans le domaine analogique. Voici comment utiliser des composants qui ne sont pas numériques.

```
int analogWrite(patte,
                value);
```

Certaines pattes de l'Arduino supportent les PWM (3, 5, 6, 9, 10, 11). Cela permet de changer l'état de la patte très rapidement et ainsi agir comme une sortie analogique. La valeur peut aller de 0 (0% de rapport cyclique ~0V) et 255 (100% de rapport cyclique ~5V).

```
int analogRead(patte);
```

Quand une patte analogique est mise en INPUT vous pouvez obtenir sa tension. Une valeur comprise entre 0 (pour 0V) et 1023 (pour 5V) est retournée.