

Apprendre à utiliser un clavier matriciel 16 touches avec Arduino.



Ateliers Arduino

par X. HINAULT

www.mon-club-elec.fr



Tous droits réservés – 2012.

Ce document légèrement payant est soumis au droit d'auteur et est réservé à l'usage personnel.

Afin d'encourager la production de supports didactiques de qualité, ce document est légèrement payant.

La licence d'utilisation est attribuée pour un usage personnel uniquement, dans le cercle familial. Mise en ligne et diffusion non autorisées.

Si vous n'êtes pas le détenteur de la licence attribuée pour l'usage de ce document, soyez sympa, merci d'acheter votre exemplaire personnel ici : <https://monclubelec.dpdcart.com/>

Pour tout problème lié à l'utilisation de ce document, veuillez envoyer une copie ici : support@mon-club-elec.fr

Pour obtenir tout autres types de licence d'utilisation (enseignement, commercial, etc...), veuillez contacter l'auteur ici : support@mon-club-elec.fr

Vous avez constaté une erreur ? une coquille ? N'hésitez pas à nous le signaler à cette adresse : support@mon-club-elec.fr

Truc d'utilisation : visualiser ce document en mode diaporama dans le visionneur PDF. Navigation avec les flèches HAUT / BAS ou la souris.

En mode fenêtre, activer le panneau latéral vous facilitera la navigation dans le document. Bonne lecture !

Lancer également le logiciel Arduino et connecter votre carte Arduino afin de pouvoir tester au fur et à mesure les codes d'exemples !

1. Intro

L'objectif ici est :

- de comprendre le principe de fonctionnement d'un clavier matriciel
- d'apprendre à utiliser la librairie **Keypad**
- d'écrire des programmes permettant
 - de lire les touches appuyées sur un clavier numérique
 - de mémoriser une séquence de touches saisies
 - de saisir une valeur numérique entière
 - de saisir une valeur numérique à virgule
 - de saisir une chaîne ayant un format prédéfini (adresse IP)
- de coder des applications utilisant un clavier matriciel, notamment :
 - un décodeur de codes secrets
 - une calculatrice sur nombres entiers

... afin d'être en mesure d'utiliser un clavier matriciel avec Arduino en fonction de ses besoins.

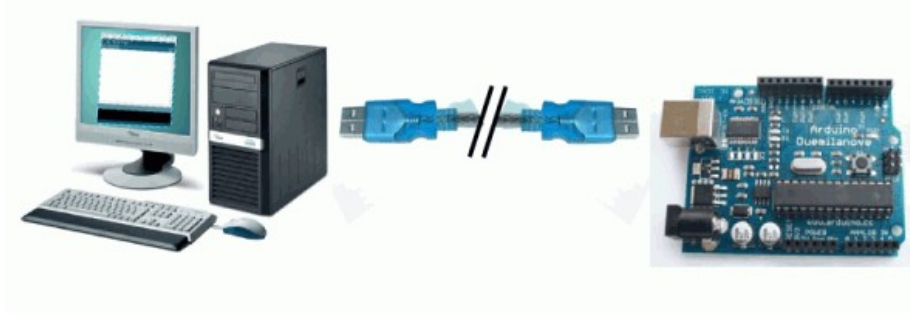


Prêt ? C'est parti !

2. Matériel nécessaire pour les ateliers Arduino

Pour cet atelier, vous aurez besoin de tout ou partie des éléments suivants pour pouvoir réaliser les exemples proposés :

De l'espace de développement Arduino

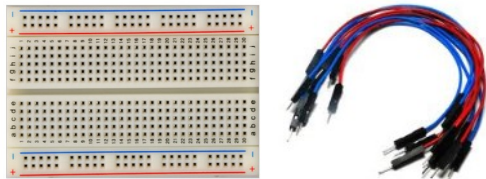


L'espace de développement Arduino associe :

- un ordinateur sous Windows, Mac Os X ou Gnu/Linux (Ubuntu)
- avec le logiciel Arduino installé (voir : <http://www.arduino.cc/>)
- un câble USB
- une carte Arduino UNO ou équivalente.

disponible chez : <http://shop.snootlab.com/> ou <http://www.gotronic.fr/>

Du nécessaire pour réaliser des montages sans soudure

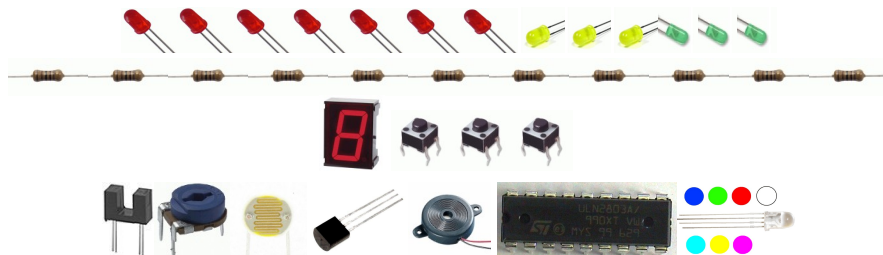


Pour réaliser des montages sans soudure, vous aurez besoin :

- d'une plaque d'essai ou breadboard moyenne (450 points)
- de quelques câbles souples (ou jumpers) mâle/mâle

disponible chez : <http://www.gotronic.fr/>

De quelques composants de base



Pour vous simplifier la vie, nous avons négocié ce kit pour vous !

Vous pouvez commander ce kit complet directement en 1 clic chez notre partenaire

<http://www.gotronic.fr/> avec le code express **701710**

GO TRONIC
ROBOTIQUE ET COMPOSANTS ÉLECTRONIQUES

Pour plus de détails, voir : http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS

Pour les ateliers Arduino niveau débutant, vous devrez idéalement disposer des composants suivants :

- des LEDs 5mm Rouges(x20), Vertes (x5) et 3 Jaunes (x5)
- digit à cathode commune rouge 13mm (x1)
- Résistances (1/4w - 5%) de 270 Ohms (x20), 4,7K Ohms (x1), 1K Ohms (x1)
- mini bouton-poussoir (x3)
- Opto-fourche (x 1)
- Résistance variable linéaire 10K (x 1)
- Photo-résistance 7mm (x 1)
- Capteur de température LM35DZ (-55/+150°C - 10mV/°C) (x 1)
- Capsule son piézoélectrique (x 1)
- ULN 2803A (CI amplificateur 8 voies, 500mA/ voie) (x 1)
- LED 5mm multicolore RVB cathode commune (x 1)

3. Matériel spécifique nécessaire pour cet atelier

Pour cet atelier vous aurez besoin également :

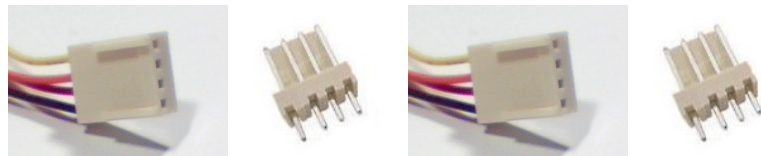
D'un clavier matriciel 16 touches (4 lignes x 4 colonnes)



Clavier 16 touches permettant la saisie de valeurs numériques, facile à utiliser avec une carte Arduino en utilisant des connecteurs 4 broches sur fils.

disponible chez : <http://www.gotronic.fr/> | Code : 07276

Du matériel nécessaire pour « préparer » un clavier matriciel pour une utilisation simplifiée avec Arduino



Pour être utilisable facilement avec Arduino, un clavier matriciel 4x4 brut nécessite une petite préparation assez simple à l'aide d'éléments de connectique simples et une résistance :

- 2 connecteur femelle sur fils – 4 contacts (code : 09827)
- 2 connecteur droit – 4 contacts (code : 09817)

disponible chez : <http://www.gotronic.fr/> avec les codes indiqués

Des outils nécessaires pour réaliser des soudures



Pour réaliser des soudures, vous aurez besoin :

- d'un fer à souder à pointe fine, 25-30W et de son support
- de soudure d'étain 60% dite « électronique » - Ø 1mm
- d'un rouleau de scotch (pratique pour faire tenir les composants)
- d'une petite pince coupante
- d'une pompe à dessouder (pour rattraper le coup au besoin)

disponible chez : <http://www.gotronic.fr/> ou en magasin de bricolage

4. Fiche technique : clavier matriciel 16 touches (4 lignes x 4 colonnes)

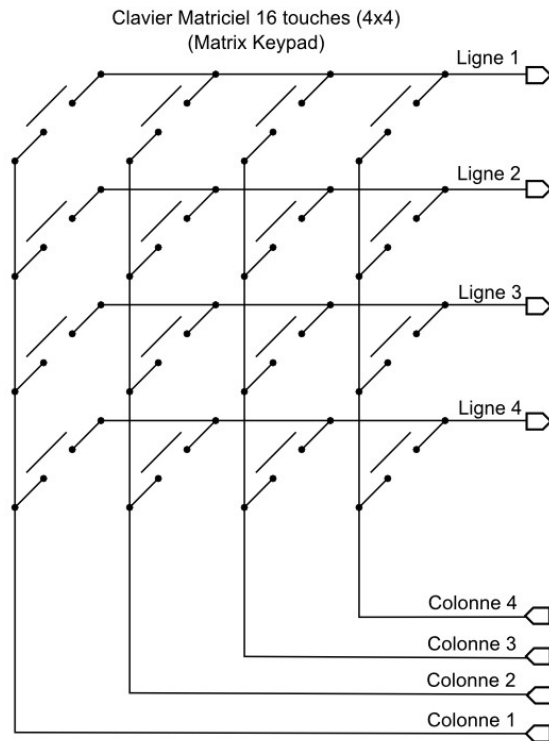
Description

Un clavier matriciel 16 touches 4 x 4 n'est ni plus ni moins qu'un boîtier plastique dans lequel 16 boutons poussoirs ont été regroupés. Le clavier dispose d'un connecteur à 8 broches.



Schéma interne

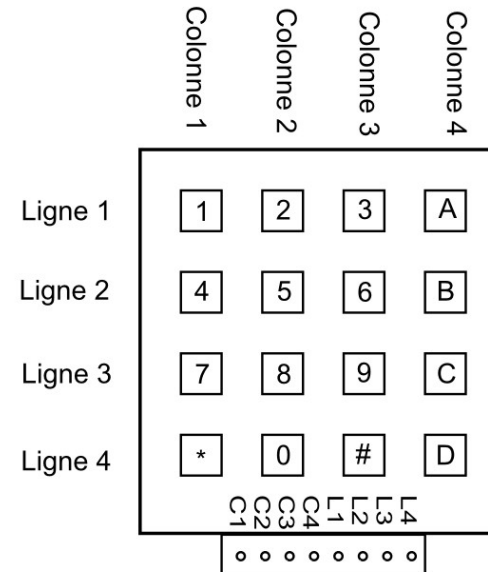
Le principe de la connexion interne consiste à mettre en contact la ligne et la colonne d'un bouton poussoir du clavier lors de l'appui sur ce bouton poussoir. Le schéma interne correspondant est le suivant :



Brochage

Un afficheur LCD standard dispose typiquement :

- de 4 broches de lignes
- de 4 broches de colonnes



Caractéristiques électriques

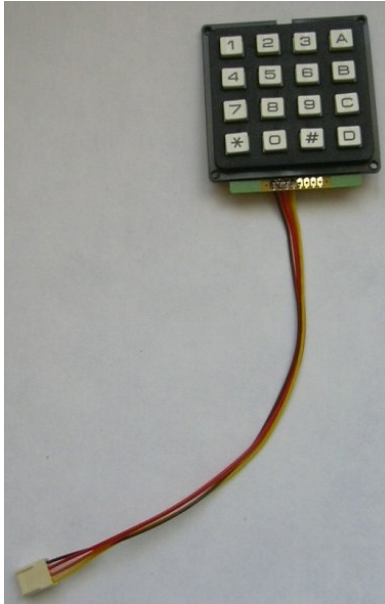
- Un clavier matriciel ne consomme rien : il se contente de mettre en contact la ligne et la colonne.

Mode de fonctionnement

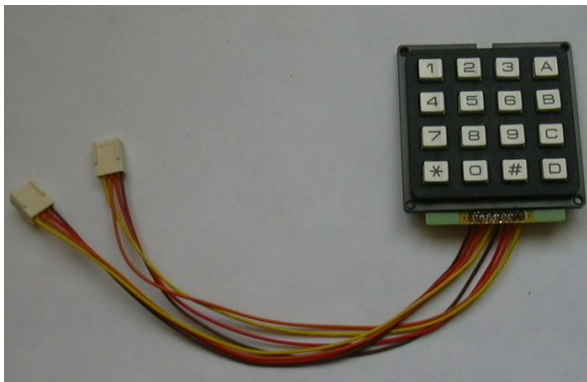
- L'utilisation avec une carte Arduino va consister à connecter directement la ligne sur une broche numérique en entrée et la colonne sur une broche numérique en sortie.
- Le programme va « scanner » l'état des broches de lignes en changeant successivement l'état des broches de colonne pour savoir quel est le bouton poussoir actuellement appuyé.

5. Préparation d'un clavier matriciel 16 touches (4 lignes x 4 colonnes) et utilisation avec une carte Arduino

- La préparation du clavier matriciel est simple à réaliser et permettra ensuite d'utiliser très facilement l'afficheur avec une carte Arduino. Voici la procédure en images.
- On commence par souder le connecteur 4 broches sur fils sur les 4 broches de colonnes :

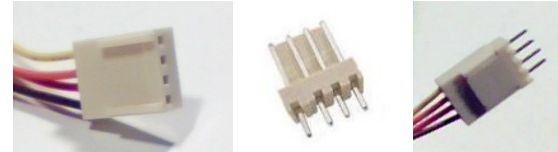


- On soude ensuite de la même façon le connecteur 4 broches sur fils sur les 4 broches de lignes :

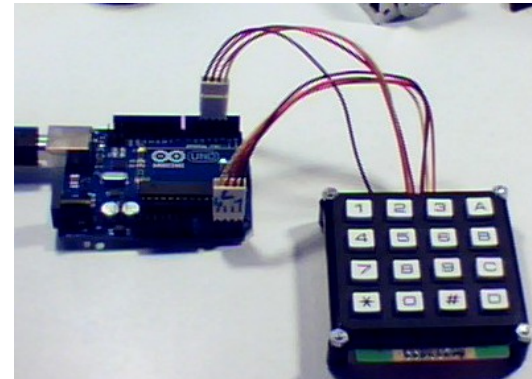


- C'est tout ! Le clavier est prêt pour être utilisé avec une carte Arduino !

- Pour pouvoir connecter facilement les connecteurs sur fils à la carte Arduino, il va falloir utiliser 2 connecteurs droits – 4 contacts que l'on va pouvoir facilement enficher dans les connecteurs femelles de la carte Arduino :



- La connexion avec la carte Arduino se réalise alors très simplement :



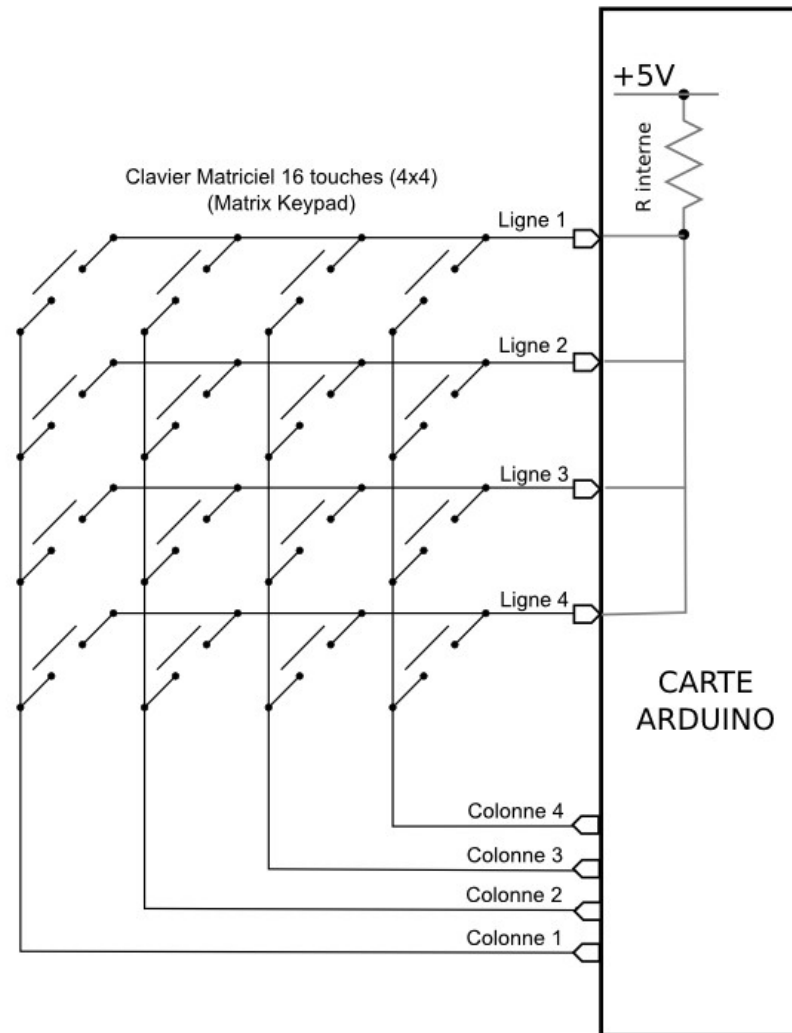
Note technique

Lors de l'utilisation du clavier, la question se pose de mettre ou non des résistances de limitation de courant sur les broches de colonnes et de lignes... En consultant plusieurs notes d'applications de chez Atmel (fabricant de l'ATmega328 et autres), on retrouve des exemples sans résistances et de plus, dans un cas où des résistances sont utilisées il est indiqué qu'elles ont pour but d'éviter les décharge électro-statiques (ESD). Ainsi, les résistances peuvent-être omises dans la grande majorité des cas.

Par contre, il faut activer le « rappel au plus » interne sur toutes les broches de lignes qui sont en entrée.... ce qui est fait par la librairie keypad lors de l'initialisation.

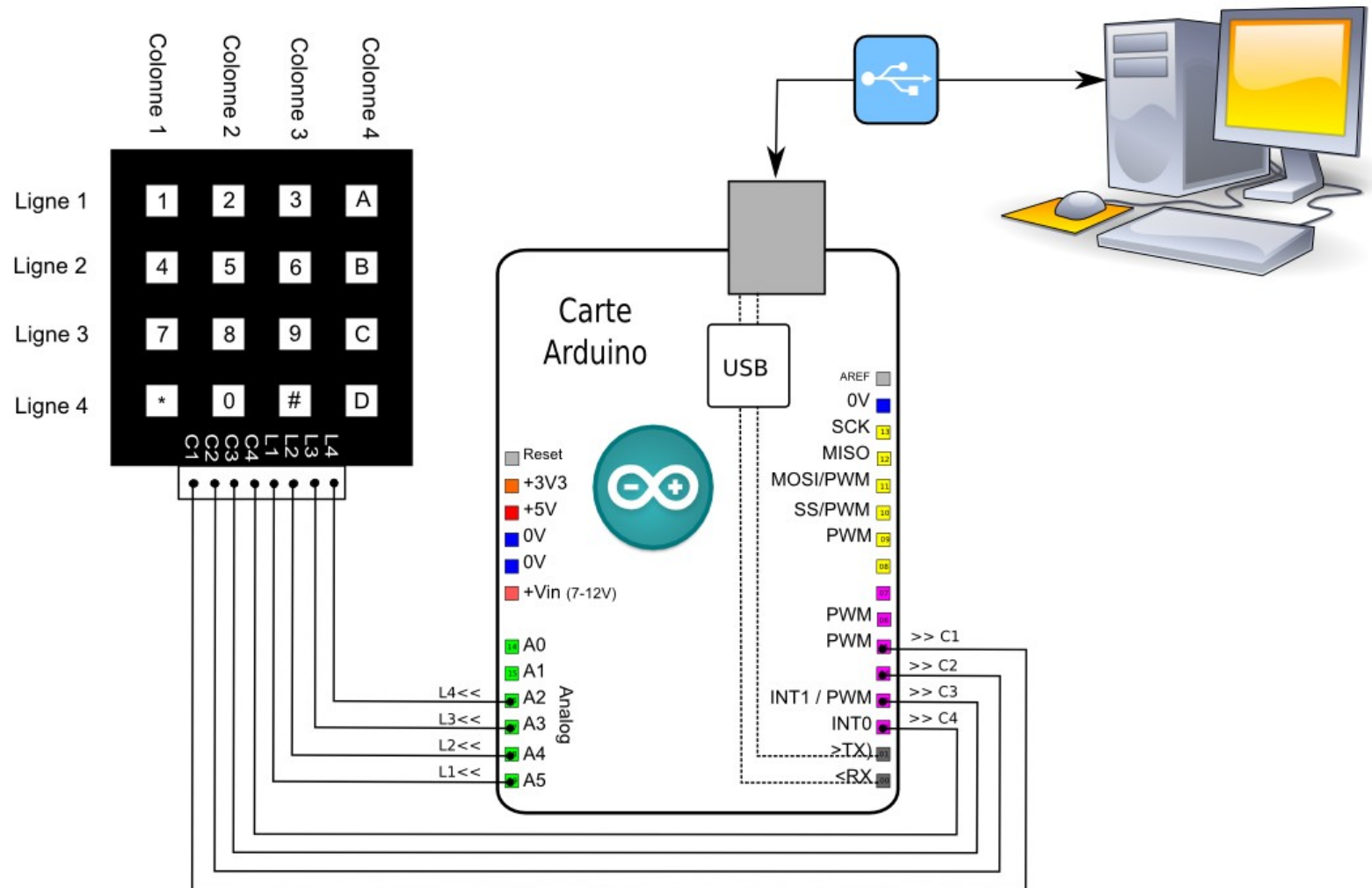
6. Utilisation d'un clavier matriciel « préparé » avec une carte Arduino : le schéma théorique

- Le principe de l'utilisation du clavier matriciel 4x4 avec une carte Arduino est simple :
 - on va connecter les 4 broches de lignes sur 4 broches numériques en entrée (avec rappel au plus interne activé).
 - on va connecter les 4 broches de colonnes sur 4 broches numériques en sortie
- Pour connaître le bouton appuyé dans une colonne, il suffira de mettre au niveau LOW la broche de colonne et à lire successivement l'état des broches de lignes. La broche de ligne qui sera au niveau LOW indiquera l'appui sur le bouton poussoir d'intersection entre la ligne et la colonne tout simplement.



7. Utilisation d'un clavier matriciel 4X4 « préparé » avec une carte Arduino : le montage

- Le montage à réaliser se superpose strictement au schéma théorique :
 - on va connecter les 4 broches de lignes sur 4 broches numériques en entrée
 - on va connecter les 4 broches de colonnes sur 4 broches numériques en sortie



8. Rappel : Langage Arduino : Introduction aux bibliothèques

C'est quoi une bibliothèque Arduino ?

Le langage Arduino comporte de nombreuses instructions comme vous avez pu le constater, une quarantaine en tout. Ces instructions sont intégrées dans ce que l'on appelle le « noyau » ou « cœur » (core en Anglais) du langage Arduino. Ces instructions sont « générales » et servent souvent.

Le langage Arduino peut cependant être étendu à la demande avec des instructions dédiées à certaines applications particulières : afin de ne pas surcharger inutilement le « cœur », ces instructions spécifiques ont été intégrées dans des « paquets d'instructions » appelés bibliothèques.

Comment ça marche ?

Par exemple, si on utilise un afficheur LCD, un servomoteur ou encore si l'on utilise un shield ethernet (réseau), on va intégrer dans notre programme la bibliothèque dédiée correspondante.

Principe général d'utilisation

Pour intégrer une bibliothèque dans un programme Arduino, c'est très simple : il suffit d'ajouter en début de programme une ligne de la forme :

```
#include <nombibliothèque.h> // bibliothèque pour servomoteur
```

ATTENTION : l'instruction include est un peu particulière : la ligne commence par un # et il n'y a pas de point virgule de fin de ligne !

Ensuite, dans le code, au niveau de l'entête déclarative, là où vous déclarez vos variables, il va falloir déclarer un objet (une sorte de super variable) représentant la bibliothèque. Cet objet est en fait une instance (= un exemplaire) d'une Classe (=le moule) qui regroupe les fonctions de la bibliothèque. On a :

```
ClasseObjet monObjet; // déclare un objet
```

Généralement ensuite :

- au niveau de la fonction `setup()`, on initialise l'objet avec les paramètres voulus
- au niveau de la fonction `draw()`, on appelle les fonctions de la bibliothèque sous la forme que vous connaissez déjà :

```
monobjet.fonction( param, param, ..);
```

Rappel : pour utiliser une fonction d'une classe du langage Arduino, on utilise le nom de la classe + un point + le nom de la fonction.

Il peut exister des variantes selon les bibliothèques, mais grosso-modo, ça fonctionne de cette façon pour la plupart des bibliothèques Arduino.

Vous avez déjà utilisé une bibliothèque !

Si vous êtes attentifs à tout ce qu'on a déjà vu, vous me direz que ça ressemble étrangement à l'utilisation de la classe **Serial...** et vous aurez raison ! En fait, la classe Serial est une bibliothèque qui est intégrée implicitement lorsque vous lancez Arduino : c'est pour ça que vous n'avez pas besoin d'utiliser **#include** pour l'utiliser.

Les bibliothèques standards Arduino

Les bibliothèques Arduino disponibles sont nombreuses, et disposent chacune de quelques fonctions à plusieurs dizaines... ce qui étend considérablement la puissance du langage Arduino et qui en fait aussi tout son intérêt. Voici la liste des bibliothèques standards du langage Arduino (**le logiciel donne la liste...**) :

- [La bibliothèque Serial](#) - pour les communications séries entre la carte Arduino et l'ordinateur ou d'autres composants
- [La bibliothèque LCD](#) - pour l'utilisation et le contrôle d'un afficheur LCD alphanumérique standard.
- [La bibliothèque Servo](#) - pour contrôler les servomoteurs.
- [La bibliothèque Stepper](#) - pour contrôler les moteurs pas à pas (nécessite une interface de commande)
- [La bibliothèque Ethernet](#) - pour se connecter à Internet en utilisant le module Arduino Ethernet
- [La bibliothèque EEPROM](#) - référence - pour lire et écrire dans la mémoire EEPROM non volatile.
- [La bibliothèque SD](#) - référence - pour utiliser une carte mémoire SD (utiliser des fichiers, stocker des données, ...)
- [La bibliothèque SoftwareSerial \(Série Logicielle\)](#) - référence - pour communication série logicielle sur n'importe quelles broches de la carte Arduino
- [La bibliothèque Wire / I2C](#) - référence - Interface "deux fils" (TWI/I2C) pour envoyer et recevoir des données sur un réseau de modules ou capteurs.
- [La bibliothèque SPI \(Serial Peripheral Interface\)](#) - pour communication série avec des modules externes supportant le protocole SPI
- [Firmata](#) - pour communiquer avec des applications sur l'ordinateur utilisant un protocole série standard.

En jaune les plus utiles. Impressionnant non ? On les étudiera pas à pas...

Les bibliothèques de la communauté

A côté de ces bibliothèques standards, il existe toute une série de bibliothèques proposées par les uns et les autres et qui concernent des matériels spécifiques, ou autre. Par exemple :

- [La bibliothèque Keypad](#) - pour l'utilisation des claviers matriciels. (**hors référence**)

Faites un tour ici pour voir ce qui existe : <http://arduino.cc/playground/Main/LibraryList>

9. Découvrir et installer les librairies Arduino fournies par la communauté Arduino

Pour comprendre...

- A côté des librairies Arduino « officielles » installées par défaut avec le langage Arduino, **il existe de nombreuses librairies fournies par la communauté** et qui peuvent être utilisées à la demande en fonction de ses besoins.
- Ces librairies sont variées et ont été créées pour diverses raisons par leurs auteurs :
 - le plus souvent **pour utiliser des matériels précis spécifiques** (par exemple un clavier de PC avec Arduino...)
 - ou **pour apporter des fonctions nouvelles** qui ne sont pas fournies en standard par le langage Arduino (écrire en mémoire Flash, utiliser les interruptions des Timers,...)
 - ou **pour permettre d'utiliser un protocole de communication particulier** avec le langage Arduino (décodage horloge atomique DCF77,...),
 - etc, etc, etc...

Ces librairies ne font pas partie « officiellement » du langage Arduino et par conséquent leur documentation est fournie (ou non !) par l'auteur. Il faut donc parfois mettre un peu les « mains dans le camboui » pour comprendre comment ça marche, mais rien de très sorcier le plus souvent... Des exemples sont presque toujours disponibles.



Où trouver des librairies Arduino de la communauté ?

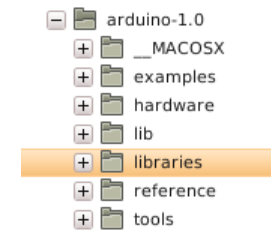
- La plupart des librairies Arduino ont été rassemblées sur le site Arduino, dans ce que l'on appelle le « playground » ou « terrain de jeu ». Allez y faire un tour, vous serez impressionné de la liste existante...
- C'est ici : <http://arduino.cc/playground/Main/LibraryList>

Il y a évidemment « à prendre et à laisser »... Lorsque des librairies de la communauté Arduino sont à connaître, nous vous en parlerons...

En résumé : vous cherchez à faire quelque chose avec Arduino ? Quelqu'un l'a probablement déjà fait avant vous et une librairie existe sûrement !

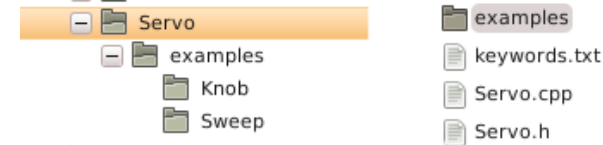
Où se trouvent les librairies Arduino déjà installées ?

- Les librairies se trouvent dans le répertoire libraries de votre répertoire Arduino :



Quelle est la structure type d'une librairie Arduino ?

- Une librairie Arduino est écrite en C++ et se présente sous la forme d'un répertoire contenant :
 - 1 ou plusieurs fichiers *.h
 - 1 ou plusieurs fichiers *.cpp
 - +/- 1 fichier keywords.txt
 - +/- 1 répertoire exemples



Comment installer une nouvelle librairie ?

- Télécharger l'archive. au format zip ou autre.
- L'extraire (dans un répertoire si l'archive ne contient pas de répertoire)
- **Vérifier que le nom du répertoire de la librairie est strictement le même que le nom du fichier *.h ou *.cpp principal. Corriger au besoin**
- Copier/coller le répertoire de la librairie dans le répertoire libraries de votre répertoire Arduino
- Relancer Arduino et vérifier que la librairie est présente dans le menu **Sketch > ImportLibrary**.
- Puis tester avec un programme d'exemple.

Envie d'écrire une librairie pour Arduino ?

- C'est un exercice intéressant mais déjà un peu avancé... Cela nécessite d'avoir quelques connaissances en C++
- Pour en savoir plus, c'est par ici : <http://arduino.cc/playground/Code/Library>

10. Installation et présentation d'une librairie de la communauté : la librairie Keypad

La librairie d'exemple utilisée

- Pour la suite, nous allons utiliser une librairie de la communauté qui permet d'utiliser facilement un clavier matriciel avec Arduino : la librairie Keypad
- Avec cette librairie, on va facilement pouvoir lire l'état du clavier matriciel.

Télécharger la librairie

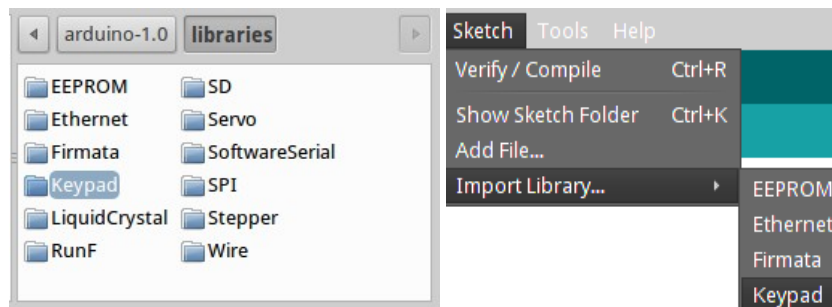
- L'archive est disponible ici :
<http://arduino.cc/playground/Code/Keypad#Download>

Documentation de la librairie

- Le site officiel de la librairie est ici :
<http://arduino.cc/playground/Code/Keypad>
- Un exemple d'utilisation est fourni ici :
<http://arduino.cc/playground/Code/Keypad#Example>

Installation

- Télécharger l'archive, au format zip ou autre. L'extraire
- **Vérifier que le nom du répertoire de la librairie est strictement le même que le nom du fichier *.h ou *.cpp principal. Corriger au besoin. Ici le nom est Keypad**
- Copier/coller le répertoire de la librairie dans le répertoire libraries de votre répertoire Arduino,
- Relancer Arduino et vérifier que la librairie est présente dans le menu **Sketch > ImportLibrary** :



Le constructeur principal

- Le constructeur principal se nomme Keypad et est de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );
```

où :

- touches est un tableau de caractères à 2 dimensions décrivant le clavier,
- brochesLignes et brochesColonnes, 2 tableaux contenant les numéros des broches de lignes et de colonnes du clavier.
- LIGNES et COLONNES, le nombre de lignes et de colonnes du clavier.

Fonctions de la librairie

- Les fonctions de la librairie Keypad sont les suivantes
 - begin()
 - waitForKey()
 - getKey()
 - getState()
 - keyStateChanged()
 - setHoldTime(unsigned int time)
 - setDebounceTime(unsigned int time)
 - addEventListener(keypadEvent)

Code d'exemple fourni avec la librairie

```
#include <Keypad.h>

const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'#','0','*'}
};

byte rowPins[ROWS] = {5, 4, 3, 2}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {8, 7, 6}; //connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup(){
  Serial.begin(9600);
}

void loop(){
  char key = keypad.getKey();

  if (key != NO_KEY){
    Serial.println(key);
  }
}
```

11. Les fonctions de la librairie Keypad et programme type utilisant un clavier 4 x 4

Les fonctions de base

void begin()

Initialise le clavier. Noter que le constructeur le fait également.

char getKey()

Renvoie la touche qui a été appuyée si c'est le cas. Cette fonction est non bloquante.

Les autres fonctions

void begin(makeKeymap(userKeymap))

Réinitialise le schéma interne du clavier.

char waitForKey()

Cette fonction attend qu'une touche soit appuyée. Attention : ceci bloque le programme jusqu'à un nouvel appui. Seules les interruptions ne sont pas bloquées. À utiliser avec précaution donc...

KeyState getState()

Renvoie l'état courant de n'importe quelle touche. Les 4 états possibles sont : IDLE, PRESSED, RELEASED et HOLD

boolean keyStateChanged()

Permet de savoir si la touche est passée d'un état vers un autre.

setHoldTime(unsigned int time)

Fixe le délai en millisecondes pendant lequel un bouton doit être relâché pour que l'état relâché soit validé.

setDebounceTime(unsigned int time)

Fixe le délai en millisecondes pendant lequel le clavier reste inactif entre 2 appuis. C'est l'équivalent d'une pause anti-rebond.

addEventListener(keypadEvent)

Génère un événement (interruption) si le clavier est utilisé.

Programme type utilisant un clavier matriciel 4x4

Voici le programme type permettant l'utilisation d'un clavier matriciel avec Arduino et la librairie Keypad. Adapter les broches à vos besoins.

```
// --- programme type utilisant un clavier matriciel 4x4 ---

#include <Keypad.h>

//--- définition nombre de lignes et de colonne
const int LIGNES = 4; // nombre de lignes
const int COLONNES = 4; // nombre de colonnes

// définition des touches
char touches[LIGNES][COLONNES] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'#','0','*','D'}
};

// tableaux de ligne et colonnes - attention : type byte obligatoire !
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de
lignes L1, L2, L3, L4
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de
colonnes C1, C2,C3, C4

// création objet Keypad représentant le clavier
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );

void setup(){
} // fin setup

void loop(){

  char touche = clavier.getKey(); // lecture de la touche appuyée

  if (touche != NO_KEY){ // si une touche a été frappée

    // ici instructions en cas d'appui d'une touche

  } // fin if touche appuyée

} // fin loop
```

12. Détecter l'appui sur une touche et affichage dans le Terminal Série : le programme

Pour commencer, je vous propose un programme très simple : afficher dans le Terminal Série la touche appuyée sur le clavier. Rien de très compliqué, mais c'est une façon simple de vérifier que tout fonctionne bien.

Entête déclarative

- On commence par importer la librairie **Keypad**
- Ensuite on configure le clavier utilisé en définissant :
 - 2 constantes correspondant au nombre de broches et lignes
 - un tableau de **char** à 2 dimensions qui définit les caractères associés à chaque touche

Note : Noter la souplesse avec laquelle on définit le tableau de touches !

- 2 tableaux de **byte** fixant les numéros des 4 broches de ligne et des 4 broches de colonnes utilisées pour le clavier matriciel.
- On déclare enfin un objet Keypad désignant le clavier à l'aide du constructeur principal de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );
```

où :

- touches est un tableau de caractères à 2 dimensions décrivant le clavier,
- brochesLignes et brochesColonnes, 2 tableaux contenant les numéros des broches de lignes et de colonnes du clavier.
- LIGNES et COLONNES, le nombre de lignes et de colonnes du clavier.

Fonction **setup()**

- On initialise simplement la connexion série à 115200 bauds.

Fonction **loop()**

- On mémorise le résultat de la fonction **getKey()** dans une variable **char** correspondant à la touche saisie.
- Ensuite, on teste si une touche a été appuyée en vérifiant que la variable de touche a une valeur différente de **NO_KEY**
- Si c'est le cas, on affiche simplement le caractère de la touche dans le Terminal Série.

Fonctionnement

Une fois la carte programmée, ouvrir le Terminal Série, fixer le débit à 115200 bauds et appuyer sur des touches du clavier : le caractère correspondant s'affiche dans le Terminal Série !

```
#include <Keypad.h>

//--- définition nombre de lignes et de colonne
const int LIGNES = 4; // nombre de lignes
const int COLONNES = 4; // nombre de colonnes

// définition des touches
char touches[LIGNES][COLONNES] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

// tableaux de ligne et colonnes - attention : type byte obligatoire !
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de
lignes L1, L2, L3, L4
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de
colonnes C1, C2,C3, C4

// création objet Keypad représentant le clavier
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );

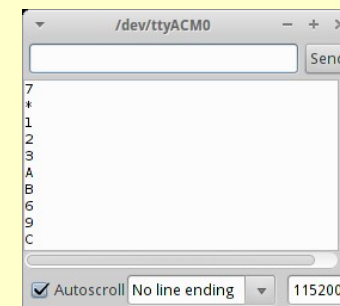
void setup(){
  Serial.begin(115200); // initialise la connexion série
} // fin setup

void loop(){

  char touche = clavier.getKey(); // lecture de la touche appuyée

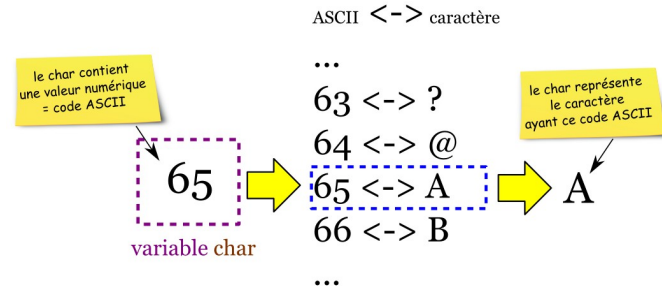
  if (touche != NO_KEY){ // si une touche a été appuyée
    Serial.println(touche); // affiche le caractère saisi
  } // fin if touche appuyée

} // fin loop
```



13. Récupérer la valeur numérique d'une touche appuyée et affichage dans le Terminal Série : le programme

Une fois que l'on est capable de récupérer la touche appuyée, il faut pouvoir récupérer la valeur numérique pour l'utiliser. Le truc consiste à retrancher la valeur 48 au code ASCII du caractère, comme cela a été vu dans l'atelier consacré à la réception de caractères sur le port série.



Rappel : Correspondance entre le code ASCII, le type char et le caractère

Entête déclarative

- On commence par importer la librairie **Keypad**
- Ensuite on configure le clavier utilisé en définissant :
 - 2 constantes correspondant au nombre de broches et lignes
 - un tableau de **char** à 2 dimensions qui définit les caractères associés à chaque touche

Note : Noter la souplesse avec laquelle on définit le tableau de touches !

- 2 tableaux de **byte** fixant les numéros des 4 broches de ligne et des 4 broches de colonnes utilisées pour le clavier matriciel.
- On déclare enfin un objet Keypad désignant le clavier à l'aide du constructeur principal de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes, brochesColonnes, LIGNES, COLONNES );
```

Fonction **setup()**

- On initialise simplement la connexion série à 115200 bauds.

Fonction **loop()**

- On mémorise le résultat de la fonction **getKey()** dans une variable **char** correspondant à la touche saisie.
- Ensuite, on teste si une touche a été appuyée en vérifiant que la variable de touche a une valeur différente de **NO_KEY**
- Si c'est le cas, on récupère la valeur décimale de la touche dans une variable de type **int** et on l'affiche. Si la valeur n'est pas comprise entre 0 et 9 un message indique que la touche appuyée n'est pas un chiffre.

```
#include <Keypad.h>

//--- définition nombre de lignes et de colonne
const int LIGNES = 4; // nombre de lignes
const int COLONNES = 4; // nombre de colonnes

// définition des touches
char touches[LIGNES][COLONNES] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

// tableaux de ligne et colonnes - attention : type byte obligatoire !
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de
lignes L1, L2, L3, L4
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de
colonnes C1, C2,C3, C4

// création objet Keypad représentant le clavier
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );

void setup(){
  Serial.begin(115200); // initialise la connexion série
} // fin setup

void loop(){

  char touche = clavier.getKey(); // lecture de la touche appuyée

  if (touche != NO_KEY){ // si une touche a été appuyée

    Serial.println(touche); // affiche le caractère saisi
    int valeur = touche-48;
    if ( (valeur>=0) && (valeur<=9) ) { // si valeur comprise entre 0 et
9

      Serial.print("Valeur de la touche = ");
      Serial.print(valeur);
      Serial.println(".");

    }
    else Serial.println("La touche n'est pas un chiffre.");

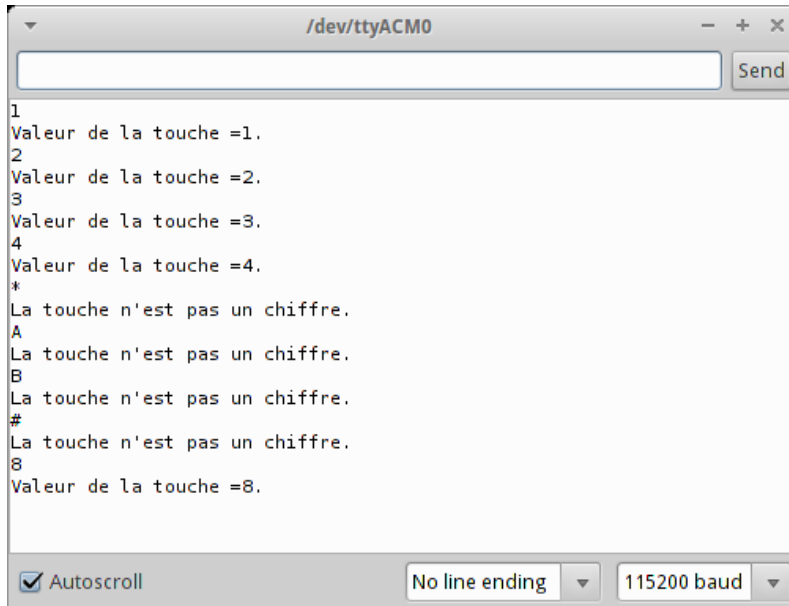
  } // fin if touche appuyée

} // fin loop
```

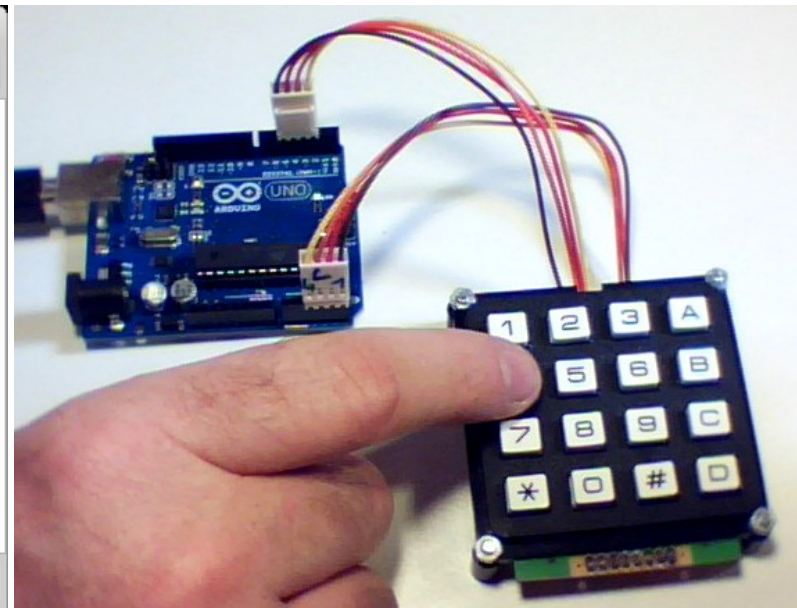

Fonctionnement

Une fois la carte programmée :

- ouvrir le Terminal Série, fixer le débit à 115200 bauds ,
- et appuyer sur des touches du clavier :
 - le caractère correspondant s'affiche dans le Terminal Série,
 - sa valeur numérique est affichée si il s'agit d'une touche numérique,
 - sinon, un message indique qu'il ne s'agit pas d'un chiffre numérique.



```
/dev/ttyACM0
1
Valeur de la touche =1.
2
Valeur de la touche =2.
3
Valeur de la touche =3.
4
Valeur de la touche =4.
*
La touche n'est pas un chiffre.
A
La touche n'est pas un chiffre.
B
La touche n'est pas un chiffre.
#
La touche n'est pas un chiffre.
8
Valeur de la touche =8.
```



Un « truc » très utile à retenir : pour les caractères 0 à 9, on obtient la valeur numérique correspondante en faisant (code ASCII – 48)

14. Mémoriser une succession de touches saisies au clavier dans une chaîne String : le programme

Ce que nous allons faire ici :

Savoir détecter l'appui sur une touche, c'est bien. Extraire la valeur numérique correspondant à cette touche également. Dans la pratique, il sera plus souvent nécessaire de mémoriser la saisie d'une succession de touches, par exemple pour la saisie d'une valeur. Ici, je vous propose de voir comment mémoriser une succession de caractères saisis dans une chaîne de caractères **String**. A vos claviers !

Entête déclarative

Initialisation du clavier matriciel

- On commence par importer la librairie **Keypad**
- Ensuite on configure le clavier utilisé en définissant :
 - 2 constantes correspondant au nombre de broches et lignes
 - un tableau de **char** à 2 dimensions qui définit les caractères associés à chaque touche

Note : Noter la souplesse avec laquelle on définit le tableau de touches !

- 2 tableaux de **byte** fixant les numéros des 4 broches de ligne et des 4 broches de colonnes utilisées pour le clavier matriciel.
- On déclare enfin un objet Keypad désignant le clavier à l'aide du constructeur principal de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );
```

Déclaration des variables globales utiles

Pour réaliser la mémorisation des touches saisies, on déclare :

- un char pour mémoriser la touche appuyée
- un int pour mémoriser la valeur numérique de la touche
- une variable de comptage du nombre de caractères saisis
- un objet String pour stocker la chaîne saisie.

```
#include <Keypad.h>

//--- définition nombre de lignes et de colonne
const int LIGNES = 4; // nombre de lignes
const int COLONNES = 4; // nombre de colonnes

// définition des touches
char touches[LIGNES][COLONNES] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

// tableaux de ligne et colonnes - attention : type byte obligatoire !
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de
lignes L1, L2, L3, L4
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de
colonnes C1, C2,C3, C4

// création objet Keypad représentant le clavier
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );

//--- variables globales utiles ---
char touche=0; // variable pour stockage touche appuyée
int valeur=0; // variable pour la valeur numérique de la touche
int compt=0; // variable comptage caractères saisis

String chaineSaisie=""; // déclare un objet String vide pour saisie
chaîne
```

Fonction setup()

- On initialise simplement la connexion série à 115200 bauds.

```
void setup(){
  Serial.begin(115200); // initialise la connexion série
} // fin setup
```

Fonction **loop()**

Boucle **while()** principale

- La saisie des touches devra se faire en boucle jusqu'à ce qu'un certain nombre de touche ou qu'une touche précise soit appuyée. Ici, la fin de la saisie de la chaîne sera provoquée par l'appui sur la touche #.
- Pour obtenir ce résultat, on utilise le « truc » suivant :
 - on crée une boucle **while(true)** qui boucle sans fin (la condition est toujours vraie) tant que l'on ne sort pas avec l'instruction **break**,
 - on place tout le code de la saisie des touches dans cette boucle ce qui aboutit à réaliser une boucle sans fin dans **loop()**
 - on sort de la boucle while au moment voulu avec l'instruction **break**

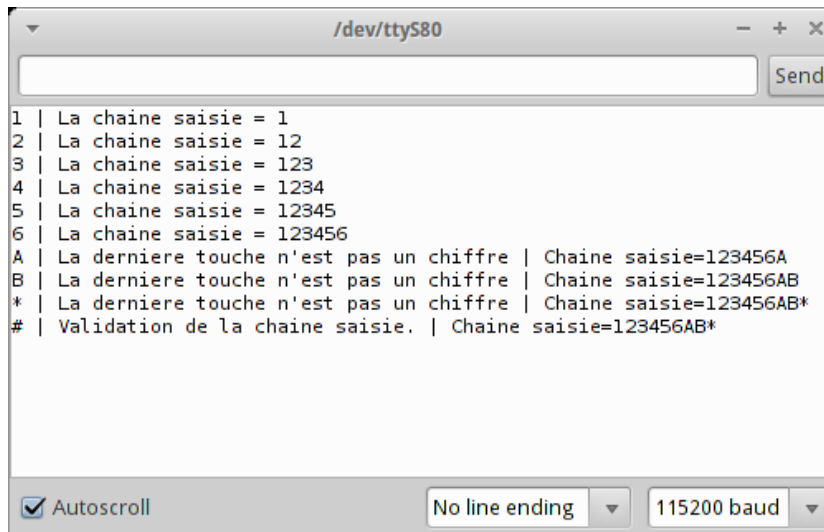
Saisie des touches

- On mémorise le résultat de la fonction **getKey()** dans une variable **char** correspondant à la touche saisie.
- Ensuite, on teste si une touche a été appuyée en vérifiant que la variable de touche a une valeur différente de **NO_KEY**
- Si c'est le cas, on récupère la valeur décimale de la touche dans une variable de type **int** et on l'affiche.
- Si la valeur est comprise entre 0 et 9 :
 - on ajoute le caractère à la chaîne **String**
 - un message indique que la touche appuyée est un chiffre.
- Si la touche appuyée est le # :
 - On valide la chaîne saisie
 - On réinitialise les variables
 - On sort de la boucle while avec l'instruction **break**
- Si la valeur n'est pas comprise entre 0 et 9 et n'est pas le # :
 - on ajoute le caractère à la chaîne **String**
 - un message indique que la touche appuyée n'est pas un chiffre.

```
void loop(){  
    while(true) { // boucle tant que l'on ne sort pas de la boucle while  
        // la sortie de la boucle while est provoquée par l'appui sur la touche # =  
        validation  
  
        touche = clavier.getKey(); // lecture de la touche appuyée  
  
        if (touche != NO_KEY){ // si une touche a été appuyée  
  
            Serial.print(touche); // affiche le caractère saisi  
  
            valeur=touche-48; // convertit l'ASCII en valeur numérique  
  
            if ( (valeur>=0) && (valeur<=9) ) { // si valeur comprise entre 0 et 9  
  
                compt=compt+1; // incrémente compteur  
                chaîneSaisie=chaîneSaisie+touche; // ajoute le caractère au String  
  
                Serial.println(" | La chaîne saisie = " + chaîneSaisie); // affiche le  
                caractère saisi  
  
                } // fin if  
  
                else if (touche=='#') { // si appui sur # = validation de la chaîne  
  
                    Serial.println (" | Validation de la chaîne saisie. | Chaîne  
saisie="+chaîneSaisie);  
                    chaîneSaisie=""; //RAZ le String de réception  
                    compt=0; // RAZ compteur  
                    delay(100); // pause  
                    break; // sort de la boucle while  
  
                } // fin else if  
  
                else { // si le caractère reçu n'est pas un chiffre ni le #  
  
                    compt=compt+1; // incrémente compteur  
                    chaîneSaisie=chaîneSaisie+touche; // ajoute le caractère au String  
  
                    Serial.print (" | La dernière touche n'est pas un chiffre");  
                    Serial.println (" | Chaîne saisie="+chaîneSaisie); // affiche la chaîne  
recue  
                } // fin else  
  
            } // fin if touche appuyée  
  
        } // fin while  
  
    } // fin loop
```

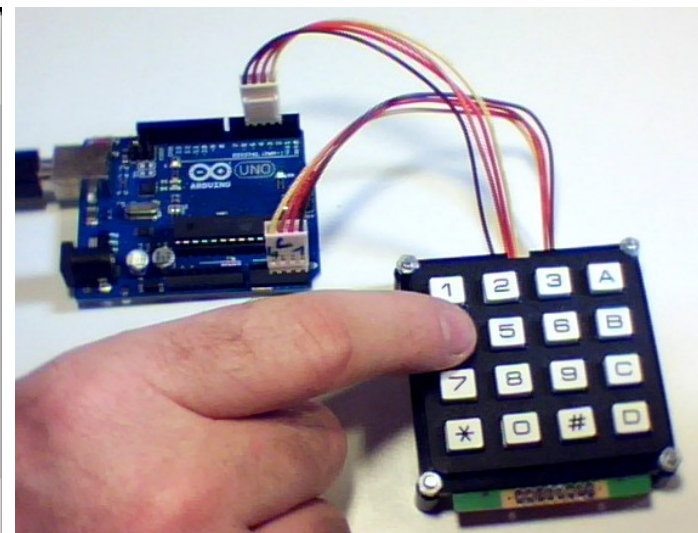
Fonctionnement du programme

- Une fois la carte programmée :
- ouvrir le Terminal Série, fixer le débit à 115200 bauds ,
- et appuyer sur des touches du clavier :
 - le caractère correspondant s'affiche dans le Terminal Série,
 - sa valeur numérique est affichée si il s'agit d'une touche numérique,
 - sinon, un message indique qu'il ne s'agit pas d'un chiffre numérique.
 - La chaîne saisie est affichée à chaque étape
 - Si on appuie sur #, la chaîne est validée.



```
/dev/ttyS80
1 | La chaîne saisie = 1
2 | La chaîne saisie = 12
3 | La chaîne saisie = 123
4 | La chaîne saisie = 1234
5 | La chaîne saisie = 12345
6 | La chaîne saisie = 123456
A | La dernière touche n'est pas un chiffre | Chaîne saisie=123456A
B | La dernière touche n'est pas un chiffre | Chaîne saisie=123456AB
* | La dernière touche n'est pas un chiffre | Chaîne saisie=123456AB*
# | Validation de la chaîne saisie. | Chaîne saisie=123456AB*

[Autoscroll checked] [No line ending] [115200 baud]
```



15. Décodeur de « code secret » : le programme

Ce que nous allons faire ici :

Une fois que l'on est capable de mémoriser une série de touches, il devient possible de vérifier si la chaîne saisie correspond à une chaîne en mémoire. On peut ainsi réaliser un simple décodeur de « code secret » qui pourra permettre de sécuriser l'utilisation d'un montage. Tout bête mais potentiellement pratique. Let's go !

Entête déclarative

Initialisation du clavier matriciel

- On commence par importer la librairie **Keypad**
- Ensuite on configure le clavier utilisé en définissant :
 - 2 constantes correspondant au nombre de broches et lignes
 - un tableau de **char** à 2 dimensions qui définit les caractères associés à chaque touche

Note : Noter la souplesse avec laquelle on définit le tableau de touches !

- 2 tableaux de **byte** fixant les numéros des 4 broches de ligne et des 4 broches de colonnes utilisées pour le clavier matriciel.
- On déclare enfin un objet Keypad désignant le clavier à l'aide du constructeur principal de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );
```

Déclaration des variables globales utiles

Pour réaliser la mémorisation des touches saisies, on déclare :

- un **char** pour mémoriser la touche appuyée
- un **int** pour mémoriser la valeur numérique de la touche
- une variable de comptage du nombre de caractères saisis
- un objet **String** pour stocker la chaîne saisie.
- un objet **String** pour stocker le code secret.

```
#include <Keypad.h>

//--- définition nombre de lignes et de colonne
const int LIGNES = 4; // nombre de lignes
const int COLONNES = 4; // nombre de colonnes

// définition des touches
char touches[LIGNES][COLONNES] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

// tableaux de ligne et colonnes - attention : type byte obligatoire !
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de
lignes L1, L2, L3, L4
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de
colonnes C1, C2,C3, C4

// création objet Keypad représentant le clavier
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );

//--- variables globales utiles ---
char touche=0; // variable pour stockage touche appuyée
int valeur=0; // variable pour la valeur numérique de la touche
int compt=0; // variable comptage caractères saisis

String chaineSaisie=""; // déclare un objet String vide pour saisie
chaine
String codeSecret="A1B2C3"; // déclare un objet String pour mémoriser
code secret
```

Fonction **setup()**

- On initialise simplement la connexion série à 115200 bauds.

```
void setup(){  
    Serial.begin(115200); // initialise la connexion série  
}  
// fin setup
```

Fonction **loop()**

- On affiche un message invitant à saisir le code
- Le résultat de la fonction saisieCode() est mémorisé
- On compare ensuite la chaîne saisie avec le code mémorisé et on affiche le message en conséquence.
- Le programme recommence...

Remarquer la souplesse d'utilisation des chaînes de caractères **String**, notamment pour la comparaison.

```
void loop(){  
    Serial.println("Saisissez votre code...");  
  
    chaineSaisie=saisieCode();  
  
    if (chaineSaisie==codeSecret) {  
        Serial.println("Code valide!");  
    } // fin if  
    else {  
        Serial.println("Code faux !");  
    } // fin else  
  
    Serial.println("----- On recommence -----");  
}  
// fin loop
```


Fonction saisieCode()

Boucle **while()** principale

- La saisie des touches devra se faire en boucle jusqu'à ce qu'un certain nombre de touche ou qu'une touche précise soit appuyée. Ici, la fin de la saisie de la chaîne sera provoquée par l'appui sur la touche #.
- Pour obtenir ce résultat, on utilise le « truc » suivant :
 - on crée une boucle **while(true)** qui boucle sans fin (la condition est toujours vraie) tant que l'on ne sort pas avec l'instruction **break**,
 - on place tout le code de la saisie des touches dans cette boucle
 - on sort de la boucle while au moment voulu avec l'instruction **break**

Saisie des touches

- On mémorise le résultat de la fonction **getKey()** dans une variable **char** correspondant à la touche saisie.
- Ensuite, on teste si une touche a été appuyée en vérifiant que la variable de touche a une valeur différente de **NO_KEY**
- Si c'est le cas, on récupère la valeur décimale de la touche dans une variable de type **int** et on l'affiche.
- Si la valeur est comprise entre 0 et 9 :
 - on ajoute le caractère à la chaîne **String**
 - un message indique que la touche appuyée est un chiffre.
- Si la touche appuyée est le # :
 - On valide la chaîne saisie
 - On réinitialise les variables
 - On sort de la boucle while avec l'instruction **break**
- Si la valeur n'est pas comprise entre 0 et 9 et n'est pas le # :
 - on ajoute le caractère à la chaîne **String**
 - un message indique que la touche appuyée n'est pas un chiffre.

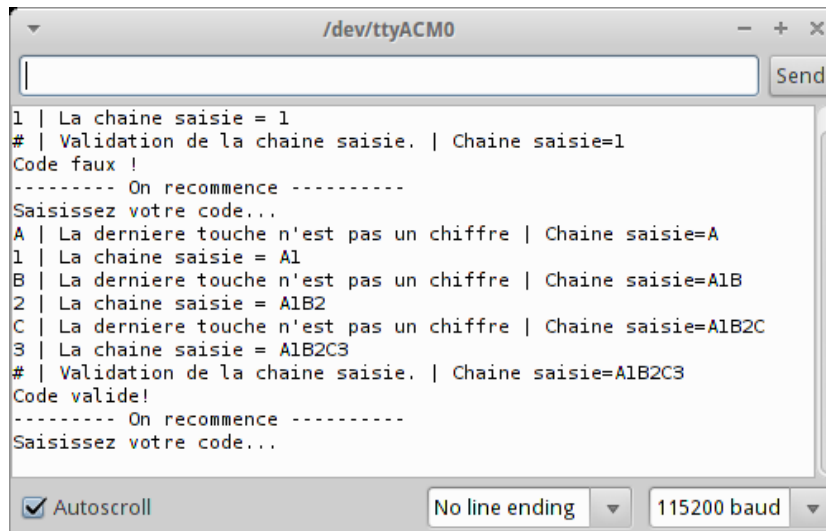
Valeur renvoyée

- La fonction renvoie l'objet **String** correspondant à la chaîne saisie.

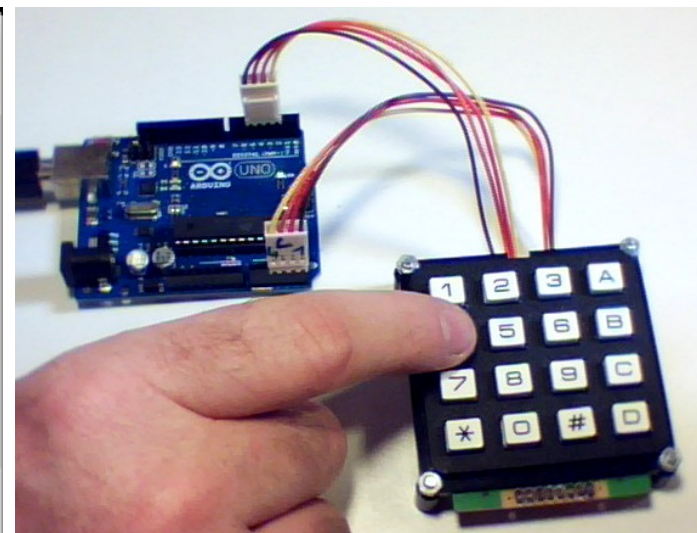
```
String saisieCode() {  
    String stringOut="";  
  
    while(true) { // boucle tant que l'on ne sort pas de la boucle while  
        // la sortie de la boucle while est provoquée par l'appui sur la touche # =  
        validation  
  
        touche = clavier.getKey(); // lecture de la touche appuyée  
  
        if (touche != NO_KEY){ // si une touche a été appuyée  
  
            Serial.print(touche); // affiche le caractère saisi  
  
            valeur=touche-48; // convertit l'ASCII en valeur numérique  
  
            if ( (valeur>=0) && (valeur<=9) ) { // si valeur comprise entre 0 et 9  
  
                compt=compt+1; // incrémente compteur  
                stringOut=stringOut+touche; // ajoute le caractère au String  
  
                Serial.println(" | La chaîne saisie = " + stringOut); // affiche le caractère  
                saisi  
  
                } // fin if  
  
                else if (touche=='#') { // si appui sur # = validation de la chaîne  
  
                    Serial.println (" | Validation de la chaîne saisie. | Chaîne  
saisie="+stringOut);  
                    //stringOut=""; //RAZ le String de réception - pas nécessaire ici  
                    compt=0; // RAZ compteur  
                    delay(100); // pause  
                    break; // sort de la boucle while  
  
                } // fin else if  
  
                else { // si le caractère reçu n'est pas un chiffre ni le #  
  
                    stringOut=stringOut+touche; // ajoute le caractère au String  
  
                    Serial.print (" | La dernière touche n'est pas un chiffre");  
                    Serial.println (" | Chaîne saisie="+stringOut); // affiche la chaîne  
                    recue  
                } // fin else  
  
                } // fin if touche appuyée  
  
            } // fin while  
  
            return(stringOut); // renvoie la chaîne saisie  
  
        } // fin saisie Code
```

Fonctionnement du programme

- Une fois la carte programmée :
- ouvrir le Terminal Série, fixer le débit à 115200 bauds ,
- et appuyer sur des touches du clavier :
 - le caractère correspondant s'affiche dans le Terminal Série,
 - sa valeur numérique est affichée si il s'agit d'une touche numérique,
 - sinon, un message indique qu'il ne s'agit pas d'un chiffre numérique.
 - La chaîne saisie est affichée à chaque étape
 - Si on appuie sur #, la chaine est validée.
- La chaine saisie est alors comparée au code mémorisé : un message indique si le code est correct.



```
/dev/ttyACM0
1 | La chaine saisie = 1
# | Validation de la chaine saisie. | Chaine saisie=1
Code faux !
----- On recommence -----
Saisissez votre code...
A | La derniere touche n'est pas un chiffre | Chaine saisie=A
1 | La chaine saisie = A1
B | La derniere touche n'est pas un chiffre | Chaine saisie=A1B
2 | La chaine saisie = A1B2
C | La derniere touche n'est pas un chiffre | Chaine saisie=A1B2C
3 | La chaine saisie = A1B2C3
# | Validation de la chaine saisie. | Chaine saisie=A1B2C3
Code valide!
----- On recommence -----
Saisissez votre code...
```



16. Saisir une valeur numérique entière de type long au clavier matriciel : le programme

Ce que nous allons faire ici :

Une fois que l'on sait recevoir une série d'appui, il n'y a qu'un pas pour obtenir la valeur numérique correspondant à l'appui successif sur plusieurs touches. C'est ce que nous allons faire ici : toutes les touches numériques appuyées seront mémorisées jusqu'à l'appui sur une touche non numérique. A ce moment là, la valeur numérique saisie sera extraite à partir de la chaîne. C'est parti... !

Entête déclarative

Initialisation du clavier matriciel

- On commence par importer la librairie **Keypad**
- Ensuite on configure le clavier utilisé en définissant :
 - 2 constantes correspondant au nombre de broches et lignes
 - un tableau de **char** à 2 dimensions qui définit les caractères associés à chaque touche

Note : Noter la souplesse avec laquelle on définit le tableau de touches !

- 2 tableaux de **byte** fixant les numéros des 4 broches de ligne et des 4 broches de colonnes utilisées pour le clavier matriciel.
- On déclare enfin un objet Keypad désignant le clavier à l'aide du constructeur principal de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );
```

Déclaration des variables globales utiles

Pour réaliser la mémorisation des touches saisies, on déclare :

- un char pour mémoriser la touche appuyée
- un int pour mémoriser la valeur numérique de la touche
- un objet String pour stocker la chaîne saisie.
- une variable **long** pour stocker la valeur numérique saisie
- une variable **boolean** qui sera utilisée pour activer ou non les messages à afficher.

Fonction **setup()**

- On initialise simplement la connexion série à 115200 bauds.

```
#include <Keypad.h>

//--- définition nombre de lignes et de colonne
const int LIGNES = 4; // nombre de lignes
const int COLONNES = 4; // nombre de colonnes

// définition des touches
char touches[LIGNES][COLONNES] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

// tableaux de ligne et colonnes - attention : type byte obligatoire !
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de
lignes L1, L2, L3, L4
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de
colonnes C1, C2,C3, C4

// création objet Keypad représentant le clavier
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );

//--- variables globales utiles ---
char touche=0; // variable pour stockage touche appuyée
int valeur=0; // variable pour la valeur numérique de la touche
String chaineSaisie=""; // déclare un objet String vide pour saisie
chaîne
long valeurSaisie =0; // valeur saisie
boolean debug=false;
```

```
void setup(){
  Serial.begin(115200); // initialise la connexion série
} // fin setup
```

Fonction **loop()**

Boucle **while()** principale

- La saisie des touches devra se faire en boucle jusqu'à ce qu'un certain nombre de touche ou qu'une touche précise soit appuyée. Ici, la fin de la saisie de la chaîne sera provoquée par l'appui sur la touche #.
- Pour obtenir ce résultat, on utilise le « truc » suivant :
 - on crée une boucle **while(true)** qui boucle sans fin (la condition est toujours vraie) tant que l'on ne sort pas avec l'instruction **break**,
 - on place tout le code de la saisie des touches dans cette boucle ce qui aboutit à réaliser une boucle sans fin dans **loop()**
 - on sort de la boucle while au moment voulu avec l'instruction **break**

Saisie des touches

- On mémorise le résultat de la fonction **getKey()** dans une variable **char** correspondant à la touche saisie.
- Ensuite, on teste si une touche a été appuyée en vérifiant que la variable de touche a une valeur différente de **NO_KEY**
- Si c'est le cas, on récupère la valeur décimale de la touche dans une variable de type **int** et on l'affiche.
- Si la valeur est comprise entre 0 et 9 :
 - on ajoute le caractère à la chaîne **String**
 - un message indique que la touche appuyée est un chiffre si debug vaut true.
- Pour toute valeur qui n'est pas un chiffre, on sort de la boucle **while()** de saisie. Le dernier caractère n'est pas ajouté à la chaîne **String**.

Extraction de la valeur numérique

- Si la chaîne String n'est pas vide, on appelle la fonction **stringToLong** qui permet d'extraire la valeur numérique correspondante. Cette fonction est décrite ci-dessous.
- La valeur est mémorisée dans une variable long et affichée.
- Si la chaîne est vide, un message est affiché.

```
void loop(){

    //----- saisie de la chaîne -----
    while(true) { // exécution tant que pas sortie de la boucle while
    //la sortie de la boucle se fait lorsque une touche non numérique est saisie

    touche = clavier.getKey(); // lecture de la touche appuyée

    if (touche != NO_KEY){ // si une touche a été appuyée

        valeur=touche-48; // convertit l'ASCII en valeur numérique

        if ( (valeur>=0) && (valeur<=9) ) { // si valeur comprise entre 0 et 9

            Serial.println(touche); // affiche le caractère saisi

            chaîneSaisie=chaîneSaisie+touche; // ajoute le caractère au String

            if (debug) Serial.println(" | La chaîne saisie = " + chaîneSaisie); //
affiche le caractère saisi

        }

        else { // si le caractère reçu n'est pas un chiffre

            if (debug) Serial.println(" | La dernière touche n'est pas un chiffre");
            if (debug) Serial.println ("Chaîne saisie="+chaîneSaisie); // affiche la
chaîne reçue

            delay(100); // pause
            break; // sort de la boucle while

        } // fin else

    } // fin if touche appuyée

    } // fin while

    //----- on se retrouve ici une fois la saisie terminée

    if (chaîneSaisie!="") {
        valeurSaisie=stringToLong(chaîneSaisie);
        Serial.print("Valeur saisie =");
        Serial.println(valeurSaisie);
        chaîneSaisie=""; //RAZ le String de réception
    }
    else {
        Serial.println("Aucune valeur saisie.");
    }

} // fin loop
```

Fonction stringToLong()

- Cette fonction :
 - reçoit un **String**
 - renvoie un **long**
- Cette fonction convertit les caractères de la chaîne String en une valeur numérique correspondante :
 - à l'aide d'une boucle, on passe en revue tous les caractères de la chaîne.
 - À l'aide de la fonction `charAt(index)` de la classe String, on récupère la valeur ASCII du caractère dont on récupère la valeur numérique.
 - On calcule la valeur numérique correspondante qui est retournée en fin de fonction

```
// ----- fonction de conversion d'un String numérique en long

long stringToLong(String chaineLong) { // fonction conversion valeur
numérique String en int

    long nombreLong=0; // variable locale
    int valeurInt=0; // variable locale

    for (int i=0; i<chaineLong.length(); i++) { // défile caractères de
la chaîne numérique

        valeurInt=chaineLong.charAt(i); // extrait le caractère ASCII à la
position voulue - index 0 est le 1er caractère
        valeurInt=valeurInt-48; // obtient la valeur décimale à partir de
la valeur ASCII

        if (valeurInt>=0 && valeurInt<=9) { // si caractère est entre 0 et
9
            nombreLong=(nombreLong*10)+valeurInt;
        } // fin si caractère est entre 0 et 9

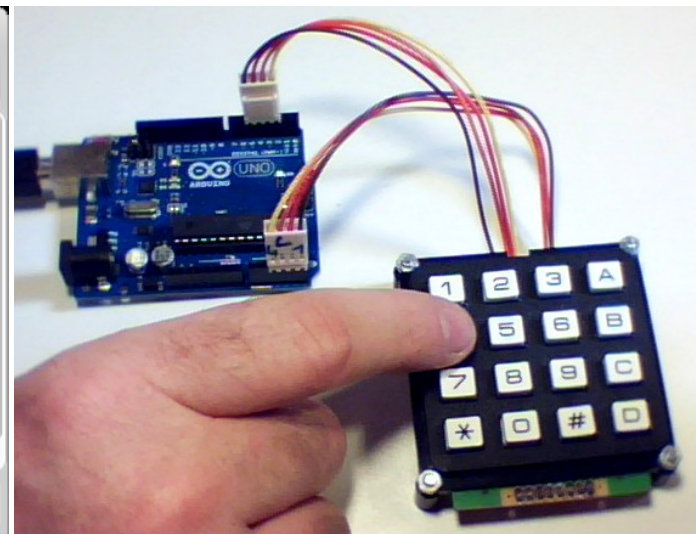
    } // fin for défile caractères

    return (nombreLong); // renvoie valeur numérique

} // ----- fin stringToLong -----
```

Fonctionnement du programme

- Une fois la carte programmée :
- ouvrir le Terminal Série, fixer le débit à 115200 bauds ,
- et appuyer sur des touches du clavier :
 - le caractère correspondant s'affiche dans le Terminal Série,
 - sa valeur numérique est affichée si il s'agit d'une touche numérique,
 - sinon, un message indique qu'il ne s'agit pas d'un chiffre numérique.
 - La chaîne saisie est affichée à chaque étape
 - Si on appuie sur #, la chaine est validée.
 - La valeur numérique correspondante est alors affichée



17. Calculatrice sur nombre entier avec un clavier matriciel : le programme.

Ce que nous allons faire ici :

A présent, nous allons essayer de saisir une première valeur, suivie d'une opération à effectuer puis d'une seconde valeur suivie d'une validation pour lancer l'opération. Le clavier matriciel va se transformer en calculatrice sur entier ! Sans prétention certes, mais un bon exercice de codage quand même... A vos neurones !

Entête déclarative

Initialisation du clavier matriciel

- On commence par importer la librairie **Keypad**
- Ensuite on configure le clavier utilisé en définissant :
 - 2 constantes correspondant au nombre de broches et lignes
 - un tableau de **char** à 2 dimensions qui définit les caractères associés à chaque touche. Ici, touches de calculatrices !

Note : Noter la souplesse avec laquelle on définit le tableau de touches !

- 2 tableaux de **byte** fixant les numéros des 4 broches de ligne et des 4 broches de colonnes utilisées pour le clavier matriciel.
- On déclare enfin un objet **Keypad** désignant le clavier à l'aide du constructeur principal de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );
```

Déclaration des variables globales utiles

Pour réaliser la mémorisation des touches saisies, on déclare :

- un **char** pour mémoriser la touche appuyée
- un **int** pour mémoriser la valeur numérique de la touche
- un objet **String** pour stocker la chaîne saisie et l'opération.
- Des **variables long** pour stocker les valeurs numériques saisies
- Une **variable float** pour le résultat
- une **variable boolean** qui sera utilisée pour activer ou non les messages à afficher.

```
// --- programme calculatrice sur entier avec un clavier matriciel 4x4 -
--

#include <Keypad.h>

//--- définition nombre de lignes et de colonne
const int LIGNES = 4; // nombre de lignes
const int COLONNES = 4; // nombre de colonnes

// définition des touches
//--- ici définition en touches de calculatrice
char touches[LIGNES][COLONNES] = {
    {'1','2','3','+'},
    {'4','5','6','-'},
    {'7','8','9','x'},
    {'.','0','=','/'}};
};

// tableaux de ligne et colonnes - attention : type byte obligatoire !
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de
lignes L1, L2, L3, L4
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de
colonnes C1, C2,C3, C4

// création objet Keypad représentant le clavier
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );

//--- variables globales utiles ---
char touche=0; // variable pour stockage touche appuyée
int valeur=0; // variable pour la valeur numérique de la touche

String chaineSaisie=""; // déclare un objet String vide pour saisie
chaîne
String operation=""; // String pour opération à réaliser

long valeurSaisie =0; // valeur saisie
long valeur1=0;
long valeur2=0;
float resultat=0.0;

boolean debug=false;
```

Fonction **setup()**

- On initialise simplement la connexion série à 115200 bauds.

```
void setup(){\n    Serial.begin(115200); // initialise la connexion série\n}\n// fin setup
```

Fonction **loop()**

Saisie de la première valeur

- On commence par appeler la fonction `saisieValeur()` décrite ci-dessous et qui renvoie le **String** correspondant aux touches numériques appuyées suivie de la touche non numérique appuyée.
- Ensuite, on extrait la valeur numérique à partir du **String** à l'aide de la fonction **substring** et de la fonction `stringToLong` décrite ci-dessous
- On extrait également le signe de l'opération qui est le dernier caractère du String, à l'aide de la fonction **substring**

Saisie de la seconde valeur

- On procède de la même façon pour la seconde valeur

Opération et résultat

- Ensuite, après avoir vérifié que le dernier caractère reçu est bien le signe =, on réalise l'opération voulue et le résultat est mémorisé dans un **float**

Remarquer tout de même la souplesse du langage Arduino et de l'objet **String** : extraction de sous-chaînes, comparaison de chaînes en toute simplicité ! Ne pas oublier que l'on programme un simple micro-contrôleur... et pas un PC ou un Mac.

```
void loop(){

  //----- première valeur -----
  chaineSaisie=saisieValeur(true); // récupère nombre saisi + dernière touche non numérique

  //-- extraction de la valeur numerique
  //Serial.println(chaineSaisie.substring(0,chaineSaisie.length()-1));
  valeur1=stringToLong(chaineSaisie.substring(0,chaineSaisie.length()-1)); // enlève le dernier
  caractère
  Serial.print("Valeur 1 = " + (String)valeur1);
  Serial.print(" ");

  //-- extraction du signe de l'opération --
  //Serial.println(chaineSaisie.charAt(chaineSaisie.length()-1));
  operation=chaineSaisie.substring(chaineSaisie.length()-1);
  Serial.print("L'operation est " );
  if (operation=="+") Serial.println("l'addition.");
  if (operation=="-") Serial.println("la soustraction");
  if (operation=="x") Serial.println("la multiplication");
  if (operation=="/") Serial.println("la division");

  //----- seconde valeur -----
  chaineSaisie=saisieValeur(true); // récupère nombre saisi + dernière touche non numérique

  //-- extraction de la valeur numerique
  //Serial.println(chaineSaisie.substring(0,chaineSaisie.length()-1));
  valeur2=stringToLong(chaineSaisie.substring(0,chaineSaisie.length()-1)); // enlève le dernier
  caractère
  Serial.print("Valeur 2 = " + (String)valeur2+ " ");
  //Serial.println();

  //----- réalisation de l'opération -----

  if (chaineSaisie.substring(chaineSaisie.length()-1)!= "="){ // si le dernier caractère n'est pas
  le =

    Serial.println("Erreur");

  }
  else { // si signe = en dernier on réalise l'opération

    if (operation=="+") resultat=valeur1+valeur2;
    if (operation=="-") resultat=valeur1-valeur2;
    if (operation=="x") resultat=valeur1*valeur2;
    if (operation=="/") resultat=valeur1/valeur2;

    //-- affiche opération réalisée
    Serial.print (valeur1);

    if (operation=="+") Serial.print(" plus ");
    if (operation=="-") Serial.print(" moins ");
    if (operation=="x") Serial.print(" fois ");
    if (operation=="/") Serial.print(" diviser par ");

    Serial.print (valeur2);
    Serial.print (" donne ");
    Serial.println (resultat);

  } // fin if

} // fin loop
```

Fonction stringToLong()

- Cette fonction :
 - reçoit un **String**
 - renvoie un **long**
- Cette fonction convertit les caractères de la chaîne String en une valeur numérique correspondante :
 - à l'aide d'une boucle, on passe en revue tous les caractères de la chaîne.
 - À l'aide de la fonction `charAt(index)` de la classe `String`, on récupère la valeur ASCII du caractère dont on récupère la valeur numérique.
 - On calcule la valeur numérique correspondante qui est retournée en fin de fonction

```
// ----- fonction de conversion d'un String numérique en long

long stringToLong(String chaineLong) { // fonction conversion valeur
numérique String en int

    long nombreLong=0; // variable locale
    int valeurInt=0; // variable locale

    for (int i=0; i<chaineLong.length(); i++) { // défile caractères de
la chaîne numérique

        valeurInt=chaineLong.charAt(i); // extrait le caractère ASCII à la
position voulue - index 0 est le 1er caractère
        valeurInt=valeurInt-48; // obtient la valeur décimale à partir de
la valeur ASCII

        if (valeurInt>=0 && valeurInt<=9) { // si caractère est entre 0 et
9
            nombreLong=(nombreLong*10)+valeurInt;
        } // fin si caractère est entre 0 et 9

    } // fin for défile caractères

    return (nombreLong); // renvoie valeur numérique

} // ----- fin stringToLong -----
```

Fonction saisieValeur()

- Cette fonction :
 - reçoit un **boolean**
 - renvoie un **String**

Boucle **while()** principale

- La saisie des touches devra se faire en boucle jusqu'à ce qu'un certain nombre de touche ou qu'une touche précise soit appuyée. Ici, la fin de la saisie de la chaîne sera provoquée par l'appui sur **n'importe quelle touche non numérique**.
- Pour obtenir ce résultat, on utilise le « truc » suivant :
 - on crée une boucle **while(true)** qui boucle sans fin (la condition est toujours vraie) tant que l'on ne sort pas avec l'instruction **break**,
 - on place tout le code de la saisie des touches dans cette boucle
 - on sort de la boucle while au moment voulu avec l'instruction **break**

Saisie des touches

- On mémorise le résultat de la fonction **getKey()** dans une variable **char** correspondant à la touche saisie.
- Ensuite, on teste si une touche a été appuyée en vérifiant que la variable de touche a une valeur différente de **NO_KEY**
- Si c'est le cas, on récupère la valeur décimale de la touche dans une variable de type **int** et on l'affiche.
- Si la valeur est comprise entre 0 et 9 :
 - on ajoute le caractère à la chaîne **String**
 - un message indique que la touche appuyée est un chiffre.
- Si la touche appuyée n'est pas une valeur numérique
 - on l'ajoute à la chaîne si le **boolean** reçu en paramètre est **true**
 - On valide la chaîne saisie
 - On sort de la boucle while avec l'instruction **break**

Valeur renvoyée

- La fonction renvoie l'objet **String** correspondant à la chaîne saisie.

```
//----- fonction de saisie de la succession des touches -----
//--- toute touche non numérique fait sortir de la saisie et est ajouté à la chaîne
String saisieValeur (boolean flagIn) {

    String chaineOut=""; // variable locale chaîne à renvoyer
    char toucheIn=0; // variable locale touche appuyée
    int valeurIn=0; // variable locale valeur numérique touche

    //----- saisie de la chaîne -----
    while(true) { // exécution tant que pas sortie de la boucle while
        //la sortie de la boucle se fait lorsque une touche non numérique est saisie

        toucheIn = clavier.getKey(); // lecture de la touche appuyée

        if (toucheIn != NO_KEY){ // si une touche a été appuyée

            valeurIn=toucheIn-48; // convertit l'ASCII en valeur numérique

            if ( (valeurIn>=0) && (valeurIn<=9) ) { // si valeur comprise entre 0 et 9

                Serial.println(toucheIn); // affiche le caractère saisi

                chaineOut=chaineOut+toucheIn; // ajoute le caractère au String

                if (debug) Serial.println(" | La chaîne saisie = " + chaineOut); // affiche le caractère saisi

            }

            else { // si le caractère reçu n'est pas un chiffre

                if (debug)Serial.print(toucheIn); // affiche le caractère saisi
                if (debug) Serial.println(" | La dernière touche n'est pas un chiffre");

                if (flagIn) chaineOut=chaineOut+toucheIn; // ajoute la touche non numérique si flagIn==true
                if (debug) Serial.println ("Chaîne saisie="+chaineOut); // affiche la chaîne reçue

                delay(100); // pause
                break; // sort de la boucle while

            } // fin else

        } // fin if touche appuyée

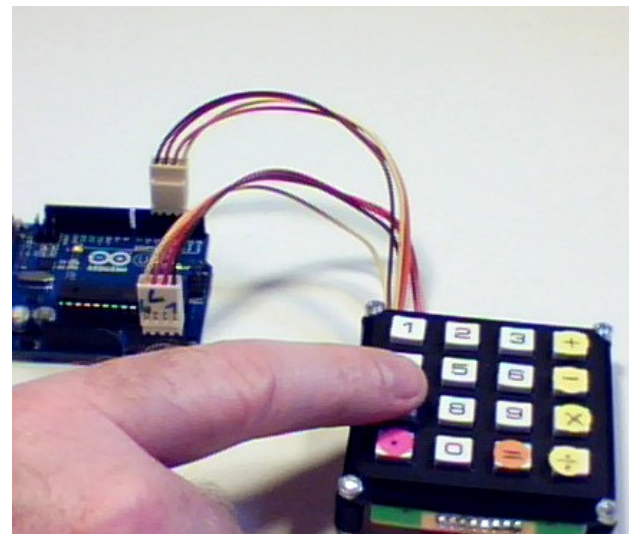
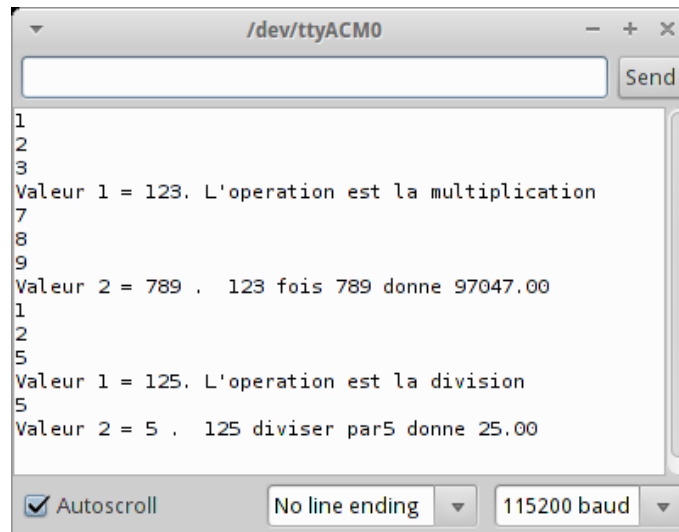
    } // fin while

    return (chaineOut);

} // fin saisie valeur
```

Fonctionnement du programme

- Une fois la carte programmée :
- ouvrir le Terminal Série, fixer le débit à 115200 bauds ,
- et appuyer sur des touches du clavier :
 - le caractère correspondant s'affiche dans le Terminal Série,
 - saisir la première valeur
 - suivie du signe de l'opération
 - puis saisir la seconde valeur
 - suivi du signe égal
 - Le résultat s'affiche



Bravo,
vous venez de programmer vous-mêmes une calculatrice pour nombres entiers !

18. Saisir une valeur décimale à virgule avec un clavier matriciel : le programme

Ce que nous allons faire ici :

Dans un certain nombre de situations, on pourra être amené à saisir une valeur décimale à virgule avec un clavier matriciel. Le principe va être le même que pour la saisie d'une valeur entière, à la différence que :

- le point ne doit pas interrompre la saisie
- la valeur décimale devra être extraite de la chaîne associant chiffres et point

Par mesure de simplification, nous associerons le point à la touche * du clavier matriciel. Y'a plus qu'à... !

Entête déclarative

Initialisation du clavier matriciel

- On commence par importer la librairie **Keypad**
- Ensuite on configure le clavier utilisé en définissant :
 - 2 constantes correspondant au nombre de broches et lignes
 - un tableau de **char** à 2 dimensions qui définit les caractères associés à chaque touche. Ici, touches de calculatrice !

Note : Noter la souplesse avec laquelle on définit le tableau de touches !

- 2 tableaux de **byte** fixant les numéros des 4 broches de ligne et des 4 broches de colonnes utilisées pour le clavier matriciel.
- On déclare enfin un objet **Keypad** désignant le clavier à l'aide du constructeur principal de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );
```

Déclaration des variables globales utiles

Pour réaliser la mémorisation des touches saisies, on déclare :

- un char pour mémoriser la touche appuyée
- un int pour mémoriser la valeur numérique de la touche
- un objet String pour stocker la chaîne saisie
- Une variable float pour stocker la valeur numérique saisie

```
#include <Keypad.h>

//--- définition nombre de lignes et de colonne
const int LIGNES = 4; // nombre de lignes
const int COLONNES = 4; // nombre de colonnes

// définition des touches
//--- ici définition en touches de calculatrice
char touches[LIGNES][COLONNES] = {
    {'1','2','3','+'},
    {'4','5','6','-'},
    {'7','8','9','x'},
    {'.','0','=','/'}};

// tableaux de ligne et colonnes - attention : type byte obligatoire !
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de
lignes L1, L2, L3, L4
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de
colonnes C1, C2,C3, C4

// création objet Keypad représentant le clavier
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,
brochesColonnes, LIGNES, COLONNES );

//--- variables globales utiles ---
char touche=0; // variable pour stockage touche appuyée
int valeur=0; // variable pour la valeur numérique de la touche

String chaineSaisie=""; // déclare un objet String vide pour saisie
chaîne
String operation=""; // String pour opération à réaliser

long valeurSaisie =0; // valeur saisie
float valeurSaisief=0.0; // valeur saisie en décimal
```

Fonction **setup()**

- On initialise simplement la connexion série à 115200 bauds.

```
void setup(){  
    Serial.begin(115200); // initialise la connexion série  
}  
// fin setup
```

Fonction **loop()**

La fonction loop est ici assez simple :

- on appelle la fonction de saisie de la valeur
- on affiche cette valeur
- on appelle la fonction de conversion de la chaîne en **float**
- on affiche le résultat avec 3 décimales

```
void loop(){  
    chaineSaisie=saisieValeur(false); // récupère nombre saisi sans  
    dernière touche non numérique  
    Serial.println("Chaîne saisie = " + chaineSaisie);  
    valeurSaisief=parseFloat(chaineSaisie);  
    Serial.print("Valeur float = ");  
    Serial.println(valeurSaisief,3); // affiche float avec 3 décimale  
}  
// fin loop
```

Fonction stringToLong()

- Cette fonction :
 - reçoit un **String**
 - renvoie un **long**
- Cette fonction convertit les caractères de la chaîne String en une valeur numérique correspondante :
 - à l'aide d'une boucle, on passe en revue tous les caractères de la chaîne.
 - À l'aide de la fonction `charAt(index)` de la classe String, on récupère la valeur ASCII du caractère dont on récupère la valeur numérique.
 - On calcule la valeur numérique correspondante qui est retournée en fin de fonction

```
// ----- fonction de conversion d'un String numérique en long

long stringToLong(String chaineLong) { // fonction conversion valeur
numérique String en int

    long nombreLong=0; // variable locale
    int valeurInt=0; // variable locale

    for (int i=0; i<chaineLong.length(); i++) { // défile caractères de
la chaîne numérique

        valeurInt=chaineLong.charAt(i); // extrait le caractère ASCII à la
position voulue - index 0 est le 1er caractère
        valeurInt=valeurInt-48; // obtient la valeur décimale à partir de
la valeur ASCII

        if (valeurInt>=0 && valeurInt<=9) { // si caractère est entre 0 et
9
            nombreLong=(nombreLong*10)+valeurInt;
        } // fin si caractère est entre 0 et 9

    } // fin for défile caractères

    return (nombreLong); // renvoie valeur numérique

} // ----- fin stringToLong -----
```

Fonction stringToFloat()

- Cette fonction :
 - reçoit un **String**
 - renvoie un **float**
- Cette fonction utilise la fonction stringToLong
- Cette fonction convertit les chiffres et point de la chaîne **String** en une valeur numérique décimale **float** correspondante :
 - on commence par extraire la position de la virgule à l'aide de la fonction **indexOf()** de la classe **String**
 - si le point n'est pas retrouvé dans la chaîne, l'index vaut -1 : donc la valeur est entière et on appelle d'emblée la fonction **stringToLong**
 - sinon :
 - on extrait tout d'abord la sous chaîne des chiffres après la virgule à l'aide de la fonction **substring** de la classe **String**
 - on fait passer ces chiffres derrière la virgule de la valeur **float** à l'aide d'une série de division par 10
 - on récupère ensuite la sous chaîne des chiffres avant la virgule que l'on additionne à la valeur précédente
 - la valeur **float** obtenue est renvoyée.

```
//----- fonction de conversion d'un String numérique avec virgule en float

float stringToFloat(String chaineFloat) { // la fonction reçoit la chaîne au format xxx.xxx et renvoie un float

    float floatOut=0.0; // variable float locale renvoyée par la fonction

    int indexPoint=chaineFloat.indexOf('.'); // récupère la position du point dans la chaîne

    if (indexPoint==-1) { // si le point n'est pas dans la chaîne...

        floatOut=stringToLong(chaineFloat); // renvoie le long et le met dans le float

    }
    else { // si le point est présent dans la chaîne on calcule la valeur correspondante float en se basant sur la position du point

        floatOut= stringToLong(chaineFloat.substring(indexPoint+1, chaineFloat.length())); // récupère valeur après la virgule
        while (abs(floatOut)>1) floatOut=floatOut/10.0; // fait passer la valeur en position derrière virgule
        floatOut=floatOut+stringToLong(chaineFloat.substring(0,indexPoint)); // récupère et ajoute valeur numérique avant virgule

    }

    return (floatOut); // renvoie le float

} // fin stringToFloat
```

Noter que les fonctions que je vous propose ici **stringToLong()** et **stringToFloat()** sont polyvalentes et pourront être réutilisées à la demande dans tout autre code : pratique !

Fonction saisieValeur()

- Cette fonction :
 - reçoit un **boolean**
 - renvoie un **String**

Boucle **while()** principale

- La saisie des touches devra se faire en boucle jusqu'à ce qu'un certain nombre de touche ou qu'une touche précise soit appuyée. Ici, la fin de la saisie de la chaîne sera provoquée par l'appui sur **n'importe quelle touche non numérique**.
- Pour obtenir ce résultat, on utilise le « truc » suivant :
 - on crée une boucle **while(true)** qui boucle sans fin (la condition est toujours vraie) tant que l'on ne sort pas avec l'instruction **break**,
 - on place tout le code de la saisie des touches dans cette boucle
 - on sort de la boucle while au moment voulu avec l'instruction **break**

Saisie des touches

- On mémorise le résultat de la fonction **getKey()** dans une variable **char** correspondant à la touche saisie.
- Ensuite, on teste si une touche a été appuyée en vérifiant que la variable de touche a une valeur différente de **NO_KEY**
- Si c'est le cas, on récupère la valeur décimale de la touche dans une variable de type **int** et on l'affiche.
- Si la valeur est comprise entre 0 et 9 **ou si c'est le .** :
 - on ajoute le caractère à la chaîne **String**
 - un message indique que la touche appuyée est un chiffre.
- Si la touche appuyée n'est pas une valeur numérique ni le .
 - on l'ajoute à la chaîne si le **boolean** reçu en paramètre est **true**
 - On valide la chaîne saisie
 - On sort de la boucle while avec l'instruction **break**

Valeur renvoyée

- La fonction renvoie l'objet **String** correspondant à la chaîne saisie.

```
//----- fonction de saisie de la succession des touches -----
String saisieValeur (boolean flagIn) {

    String chaineOut=""; // variable locale chaîne à renvoyer
    char toucheIn=0; // variable locale touche appuyée
    int valeurIn=0; // variable locale valeur numérique touche

    //----- saisie de la chaîne -----
    while(true) { // exécution tant que pas sortie de la boucle while
        //la sortie de la boucle se fait lorsque une touche non numérique est
        saisie

        toucheIn = clavier.getKey(); // lecture de la touche appuyée

        if (toucheIn != NO_KEY){ // si une touche a été appuyée

            valeurIn=toucheIn-48; // convertit l'ASCII en valeur numérique

            if ( (valeurIn>=0) && (valeurIn<=9) ) { // si valeur comprise entre
            0 et 9

                Serial.println(toucheIn); // affiche le caractère saisi

                chaineOut=chaineOut+toucheIn; // ajoute le caractère au String
            }
            else if (toucheIn=='.') {

                Serial.println(toucheIn); // affiche le caractère saisi
                chaineOut=chaineOut+toucheIn; // ajoute le caractère au String
            }
            else { // si le caractère reçu n'est pas un chiffre ni le .

                if (flagIn) chaineOut=chaineOut+toucheIn; // ajoute la touche
                non numérique si flagIn==true

                delay(100); // pause
                break; // sort de la boucle while

            } // fin else

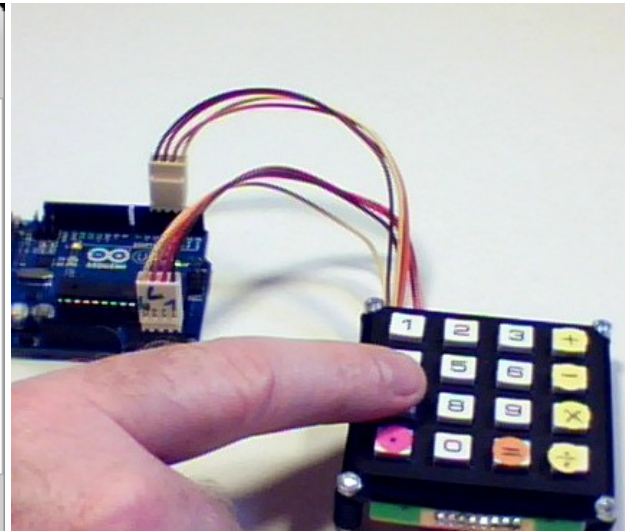
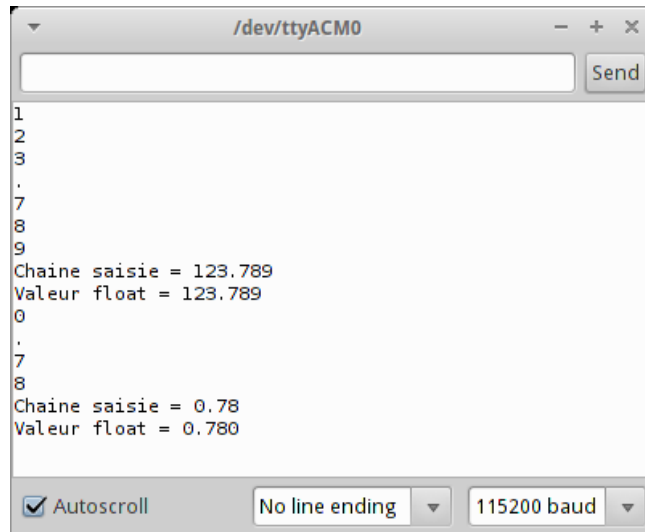
        } // fin if touche appuyée
    } // fin while

    return (chaineOut);

} // fin saisie valeur
```

Fonctionnement du programme

- Une fois la carte programmée :
- ouvrir le Terminal Série, fixer le débit à 115200 bauds ,
- et appuyer sur des touches du clavier :
 - le caractère correspondant s'affiche dans le Terminal Série,
 - saisir une valeur décimale sous la forme 123.456
 - La valeur correspondante s'affiche



A ce stade, ça ouvre quand même pas mal de possibilités : saisir une longitude, une latitude ou tout autre coefficient à virgule avec un clavier matriciel ! Ici, on utilise le port série, mais tout se passe du côté de l'Arduino et il sera donc possible de réaliser des applications autonomes avec écran LCD par exemple et clavier matriciel pour saisir les valeurs numériques entières ou décimales : à vos idées !

19. Saisir une adresse IP avec un clavier matriciel

Ce que nous allons faire ici :

« Allez, une petite dernière pour la route ! » Je suis en forme... je continue. Cette fois, je vous propose de voir comment saisir une adresse IP au clavier matriciel. Une adresse IP est une adresse de la forme xxx.xxx.xxx.xxx Une telle adresse est utilisée sur un réseau local, ethernet ou wifi, ou bien internet pour accéder à un poste du réseau. Ici, on ne réalise aucune connexion, mais je vous propose un code permettant de s'assurer de la cohérence de l'adresse saisie qui pourra servir ultérieurement pour des montages Arduino utilisant une connexion réseau (notamment via avec shield Ethernet ou wifi). Allez, encore un petit effort...

Entête déclarative

Initialisation du clavier matriciel

- On commence par importer la librairie **Keypad**
- Ensuite on configure le clavier utilisé en définissant :
 - 2 constantes correspondant au nombre de broches et lignes
 - un tableau de **char** à 2 dimensions qui définit les caractères associés à chaque touche. Ici, touches de calculatrices !

Note : Noter la souplesse avec laquelle on définit le tableau de touches !

- 2 tableaux de **byte** fixant les numéros des 4 broches de ligne et des 4 broches de colonnes utilisées pour le clavier matriciel.
- On déclare enfin un objet **Keypad** désignant le clavier à l'aide du constructeur principal de la forme :

```
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,  
brochesColonnes, LIGNES, COLONNES );
```

Déclaration des variables globales utiles

Pour réaliser la mémorisation des touches saisies, on déclare :

- un char pour mémoriser la touche appuyée
- un int pour mémoriser la valeur numérique de la touche
- un objet String pour stocker la chaîne saisie
- un autre objet String pour stocker l'adresse IP

```
// --- programme saisie d'une adresse IP avec un clavier matriciel 4x4 -  
--  
  
#include <Keypad.h>  
  
//--- définition nombre de lignes et de colonne  
const int LIGNES = 4; // nombre de lignes  
const int COLONNES = 4; // nombre de colonnes  
  
// définition des touches  
//--- ici définition en touches de calculatrice  
char touches[LIGNES][COLONNES] = {  
    {'1','2','3','+'},  
    {'4','5','6','-'},  
    {'7','8','9','x'},  
    {'.','0','=','/'},  
};  
  
// tableaux de ligne et colonnes - attention : type byte obligatoire !  
byte brochesLignes[LIGNES] = {19,18,17,16}; //tableau des broches de  
lignes L1, L2, L3, L4  
byte brochesColonnes[COLONNES] = {5, 4, 3, 2}; //tableau des broches de  
colonnes C1, C2,C3, C4  
  
// création objet Keypad représentant le clavier  
Keypad clavier = Keypad( makeKeymap(touches), brochesLignes,  
brochesColonnes, LIGNES, COLONNES );  
  
//--- variables globales utiles ---  
char touche=0; // variable pour stockage touche appuyée  
int valeur=0; // variable pour la valeur numérique de la touche  
  
String chaineSaisie=""; // déclare un objet String vide pour saisie  
chaîne  
String adresseIP=""; // déclare un objet String pour stocker l'adresse  
IP
```

Fonction **setup()**

- On initialise simplement la connexion série à 115200 bauds.

```
void setup(){  
    Serial.begin(115200); // initialise la connexion série  
}  
// fin setup
```

Fonction loop()

Ce code réutilise pas mal de chose que nous avons déjà vues et constitue une belle synthèse de la manipulation de l'objet `String()` :

- A l'aide d'une boucle for, on réalise la saisie successive des 4 groupes de valeur xxx
- pour les 3 premiers groupes, on vérifie que la chaîne se termine bien par « . », fait moins de 5 caractères et que la valeur numérique est inférieure à 255, valeur maximale autorisée pour chaque groupe de chiffre. Remarquer l'utilisation des conditions imbriquées avec l'instruction `if`
 - on affiche la chaîne si le format est correct
 - et on l'ajoute à la chaîne de stockage de l'adresse IP
- pour le 4ème groupe, on vérifie simplement que la valeur numérique est inférieure à 255.
 - on affiche l'adresse IP complète
 - on sort de la boucle avec l'instruction `break`

Remarquer l'imbrication des fonctions successives dans la condition utilisée : toute la puissance du langage Arduino apparaît ici clairement !

- Si aucune des conditions précédentes n'est vérifiée, c'est qu'il y a une erreur :
 - un message est affiché
 - et on sort de la boucle `for`

On voit ici comme il est possible de vérifier la validité du format d'une chaîne saisie au clavier matriciel. Tout ça pour seulement 8Ko de code !

```
void loop(){
    adresseIP=""; // Réinitialise adresse IP
    Serial.println("Saisir une nouvelle adresse IP au format
xxx.xxx.xxx.xxx :");

    for (int i=0; i<=3; i++) { // les 3 premiers groupes de xxx. et le
dernier xxx

        chaineSaisie=saisieValeur(true); // récupère nombre saisi avec dernière
touche non numérique

        Serial.println("Chaîne saisie = " + chaineSaisie);

        if (
            (i<3) // pour les 3ers groupes
            && (chaineSaisie.length()<=4) // et si la chaîne a moins de 5
caractères
            && (chaineSaisie.endsWith(".")) // et se termine avec .
            && (stringToLong(chaineSaisie.substring(0,chaineSaisie.length()-
1))<=255) // et si valeur < 255
        ) {

            adresseIP=adresseIP+chaineSaisie;
            //Serial.println("Adresse IP = " + adresseIP);

        } // fin if
        else if ((i==3) // 4ème groupe
            && (stringToLong(chaineSaisie.substring(0,chaineSaisie.length
()-1))<=255) // et si valeur < 255
        ){

            adresseIP=adresseIP+chaineSaisie.substring(0,chaineSaisie.lengt
h()-1); // sans le dernier caractère de la chaîne saisie
            Serial.println("Adresse IP = " + adresseIP);
            Serial.println();
            break; // sort de la boucle for

        } // fin else if

        else {
            Serial.println("Erreur de saisie - Recommencer");
            break; // sort de la boucle for
        } // fin else

    } // fin for

} // fin loop
```

Fonction stringToLong()

- Cette fonction :
 - reçoit un **String**
 - renvoie un **long**
- Cette fonction convertit les caractères de la chaîne String en une valeur numérique correspondante :
 - à l'aide d'une boucle, on passe en revue tous les caractères de la chaîne.
 - À l'aide de la fonction `charAt(index)` de la classe String, on récupère la valeur ASCII du caractère dont on récupère la valeur numérique.
 - On calcule la valeur numérique correspondante qui est retournée en fin de fonction

```
// ----- fonction de conversion d'un String numérique en long

long stringToLong(String chaineLong) { // fonction conversion valeur
numérique String en int

    long nombreLong=0; // variable locale
    int valeurInt=0; // variable locale

    for (int i=0; i<chaineLong.length(); i++) { // défile caractères de
la chaîne numérique

        valeurInt=chaineLong.charAt(i); // extrait le caractère ASCII à la
position voulue - index 0 est le 1er caractère
        valeurInt=valeurInt-48; // obtient la valeur décimale à partir de
la valeur ASCII

        if (valeurInt>=0 && valeurInt<=9) { // si caractère est entre 0 et
9
            nombreLong=(nombreLong*10)+valeurInt;
        } // fin si caractère est entre 0 et 9

    } // fin for défile caractères

    return (nombreLong); // renvoie valeur numérique

} // ----- fin stringToLong -----
```

Fonction saisieValeur()

- Cette fonction :
 - reçoit un **boolean**
 - renvoie un **String**

Boucle **while()** principale

- La saisie des touches devra se faire en boucle jusqu'à ce qu'un certain nombre de touche ou qu'une touche précise soit appuyée. Ici, la fin de la saisie de la chaîne sera provoquée par l'appui sur **n'importe quelle touche non numérique**.
- Pour obtenir ce résultat, on utilise le « truc » suivant :
 - on crée une boucle **while(true)** qui boucle sans fin (la condition est toujours vraie) tant que l'on ne sort pas avec l'instruction **break**,
 - on place tout le code de la saisie des touches dans cette boucle
 - on sort de la boucle while au moment voulu avec l'instruction **break**

Saisie des touches

- On mémorise le résultat de la fonction **getKey()** dans une variable **char** correspondant à la touche saisie.
- Ensuite, on teste si une touche a été appuyée en vérifiant que la variable de touche a une valeur différente de **NO_KEY**
- Si c'est le cas, on récupère la valeur décimale de la touche dans une variable de type **int** et on l'affiche.
- Si la valeur est comprise entre 0 et 9 :
 - on ajoute le caractère à la chaîne **String**
 - un message indique que la touche appuyée est un chiffre.
- Si la touche appuyée n'est pas une valeur numérique
 - on l'ajoute à la chaîne si le **boolean** reçu en paramètre est **true**
 - On valide la chaîne saisie
 - On sort de la boucle while avec l'instruction **break**

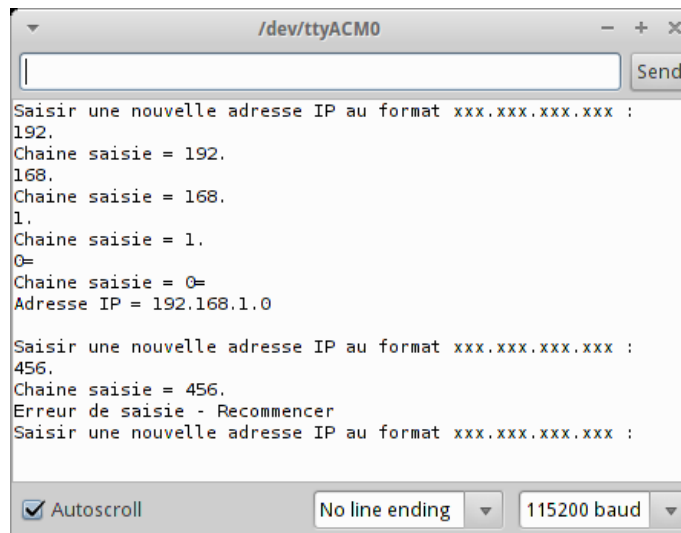
Valeur renvoyée

- La fonction renvoie l'objet **String** correspondant à la chaîne saisie.

```
//----- fonction de saisie de la succession des touches -----  
  
String saisieValeur (boolean flagIn) {  
  
    String chaineOut=""; // variable locale chaîne à renvoyer  
    char toucheIn=0; // variable locale touche appuyée  
    int valeurIn=0; // variable locale valeur numérique touche  
  
    //----- saisie de la chaîne -----  
    while(true) { // exécution tant que pas sortie de la boucle while  
        //la sortie de la boucle se fait lorsque une touche non numérique est  
        saisie  
  
        toucheIn = clavier.getKey(); // lecture de la touche appuyée  
  
        if (toucheIn != NO_KEY){ // si une touche a été appuyée  
  
            valeurIn=toucheIn-48; // convertit l'ASCII en valeur numérique  
  
            if ( (valeurIn>=0) && (valeurIn<=9) ) { // si valeur comprise entre  
            0 et 9  
  
                Serial.print(toucheIn); // affiche le caractère saisi  
  
                chaineOut=chaineOut+toucheIn; // ajoute le caractère au String  
            }  
  
            else { // si le caractère reçu n'est pas un chiffre ni le .  
  
                Serial.println(toucheIn); // affiche le caractère saisi  
  
                if (flagIn) chaineOut=chaineOut+toucheIn; // ajoute la touche  
                non numérique si flagIn==true  
  
                delay(100); // pause  
                break; // sort de la boucle while  
  
            } // fin else  
  
        } // fin if touche appuyée  
  
    } // fin while  
  
    return (chaineOut);  
  
} // fin saisie valeur
```

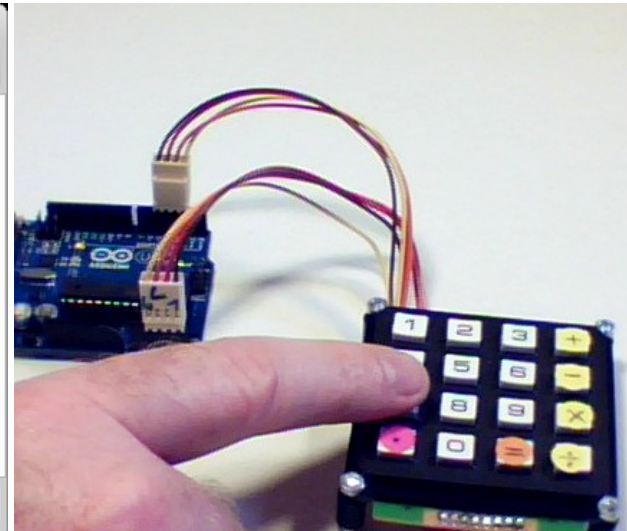
Fonctionnement du programme

- Une fois la carte programmée :
- ouvrir le Terminal Série, fixer le débit à 115200 bauds ,
- et appuyer sur des touches du clavier :
 - le caractère correspondant s'affiche dans le Terminal Série,
 - saisir une adresse IP sous la forme 192.168.1.1
 - La valeur de l'adresse IP saisie s'affiche si le format est correct, sinon, un message invite à recommencer.



```
/dev/ttyACM0
Saisir une nouvelle adresse IP au format xxx.xxx.xxx.xxx :
192.
Chaine saisie = 192.
168.
Chaine saisie = 168.
1.
Chaine saisie = 1.
0=
Chaine saisie = 0=
Adresse IP = 192.168.1.0

Saisir une nouvelle adresse IP au format xxx.xxx.xxx.xxx :
456.
Chaine saisie = 456.
Erreur de saisie - Recommencer
Saisir une nouvelle adresse IP au format xxx.xxx.xxx.xxx :
```



Si vous avez tenu jusqu'ici, bravo !
Vous êtes en passe de devenir un vrai « Arduino codeur » !

20. Les éléments du langage Arduino étudiés dans cet atelier

Les fonctions de la librairie Keypad

Nous avons présenté et étudié ici les fonctions de la librairie Keypad, notamment :

- begin()
- waitForKey()
- getKey()

La documentation complète du langage Arduino en français est disponible ici :
http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.ReferenceMaxi

21. A présent, vous devriez être capable :

- d'écrire un programme capable de lire les touches appuyées sur un clavier matriciel et d'utiliser le résultat obtenu dans vos programmes.

Table des matières

Intro |
Matériel nécessaire pour les ateliers Arduino |
Matériel spécifique nécessaire pour cet atelier |
Fiche technique : clavier matriciel 16 touches (4 lignes x 4 colonnes) |
Préparation d'un clavier matriciel 16 touches (4 lignes x 4 colonnes) et utilisation avec une carte Arduino |
Utilisation d'un clavier matriciel « préparé » avec une carte Arduino : le schéma théorique |
Utilisation d'un clavier matriciel 4X4 « préparé » avec une carte Arduino : le montage |
Rappel : Langage Arduino : Introduction aux bibliothèques |
Découvrir et installer les bibliothèques Arduino fournies par la communauté Arduino |
Installation et présentation d'une bibliothèque de la communauté : la bibliothèque Keypad |
Les fonctions de la bibliothèque Keypad et programme type utilisant un clavier 4 x 4 |
Détecter l'appui sur une touche et affichage dans le Terminal Série : le programme |
Récupérer la valeur numérique d'une touche appuyée et affichage dans le Terminal Série : le programme |
Mémoriser une succession de touches saisies au clavier dans une chaîne String : le programme |
Décodeur de « code secret » : le programme |
Saisir une valeur numérique entière de type long au clavier matriciel : le programme |
Calculatrice sur nombre entier avec un clavier matriciel : le programme. |
Saisir une valeur décimale à virgule avec un clavier matriciel : le programme |
Saisir une adresse IP avec un clavier matriciel |
Les éléments du langage Arduino étudiés dans cet atelier |
A présent, vous devriez être capable : |

Bravo !
vous avez terminé cet atelier Arduino !



Prêt pour la suite ? Retrouvez de nombreux autres thèmes d'ateliers Arduino ici :

http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS