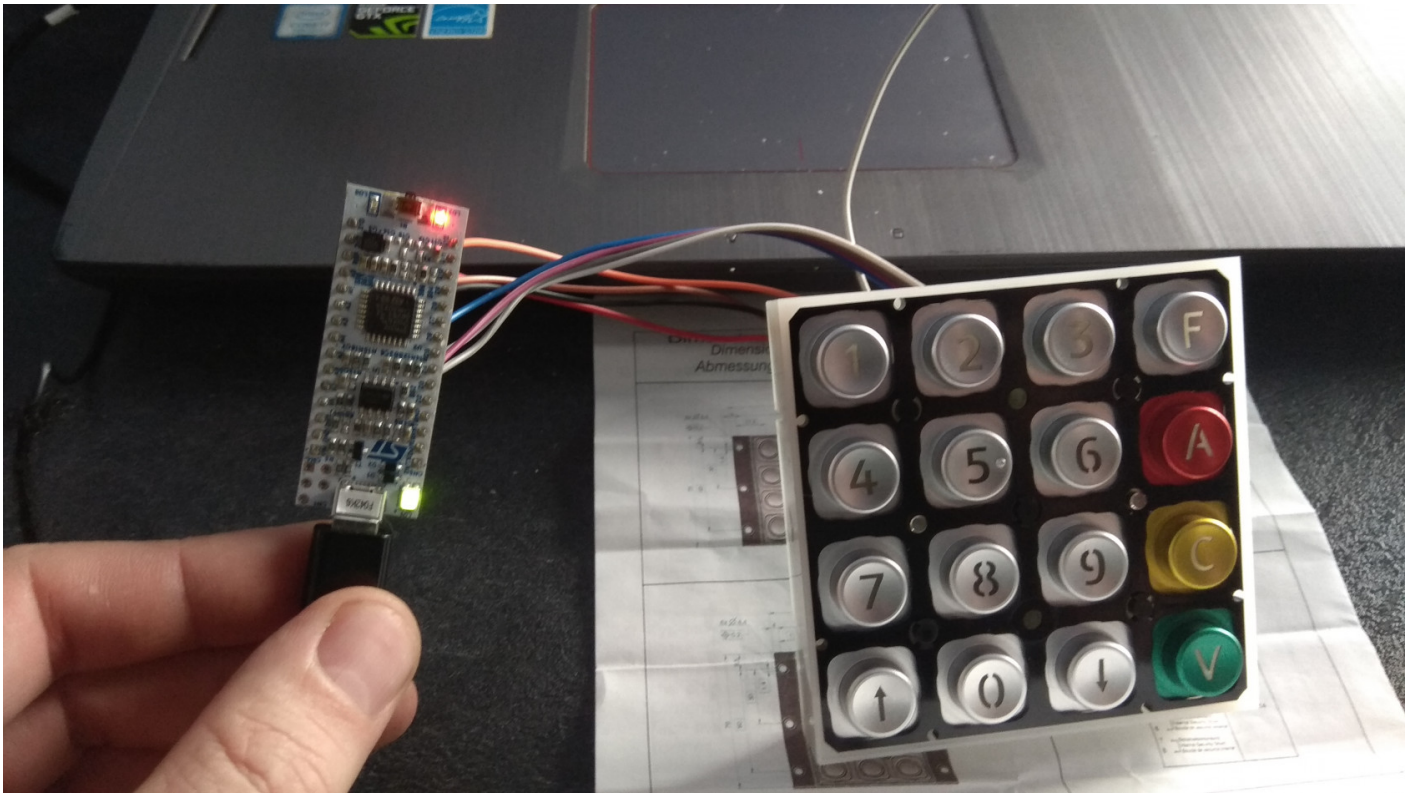
# Utilisation des keypad (débutant)

## Introduction

Au travers de ce bref tutoriel je vous propose de découvrir comment utiliser les keypads dispo au lab issu d'un don (on en a bien 50 !) :



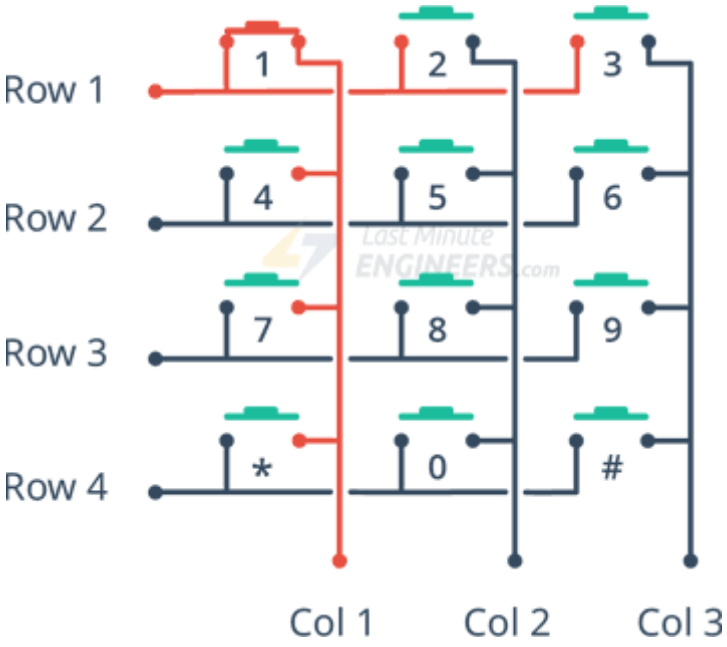

Le clavier relié à une carte STM32F042 (portage assuré par mes soins )

Il s'agit de clavier dit *matriciel* parce qu'ils utilisent un système de matrice en ligne et colonne pour multiplexer les signaux, c'est à dire que des connexions vont être mutualisées afin d'abaisser le nombre de fils à mettre entre l'Arduino et le clavier en lui-même.

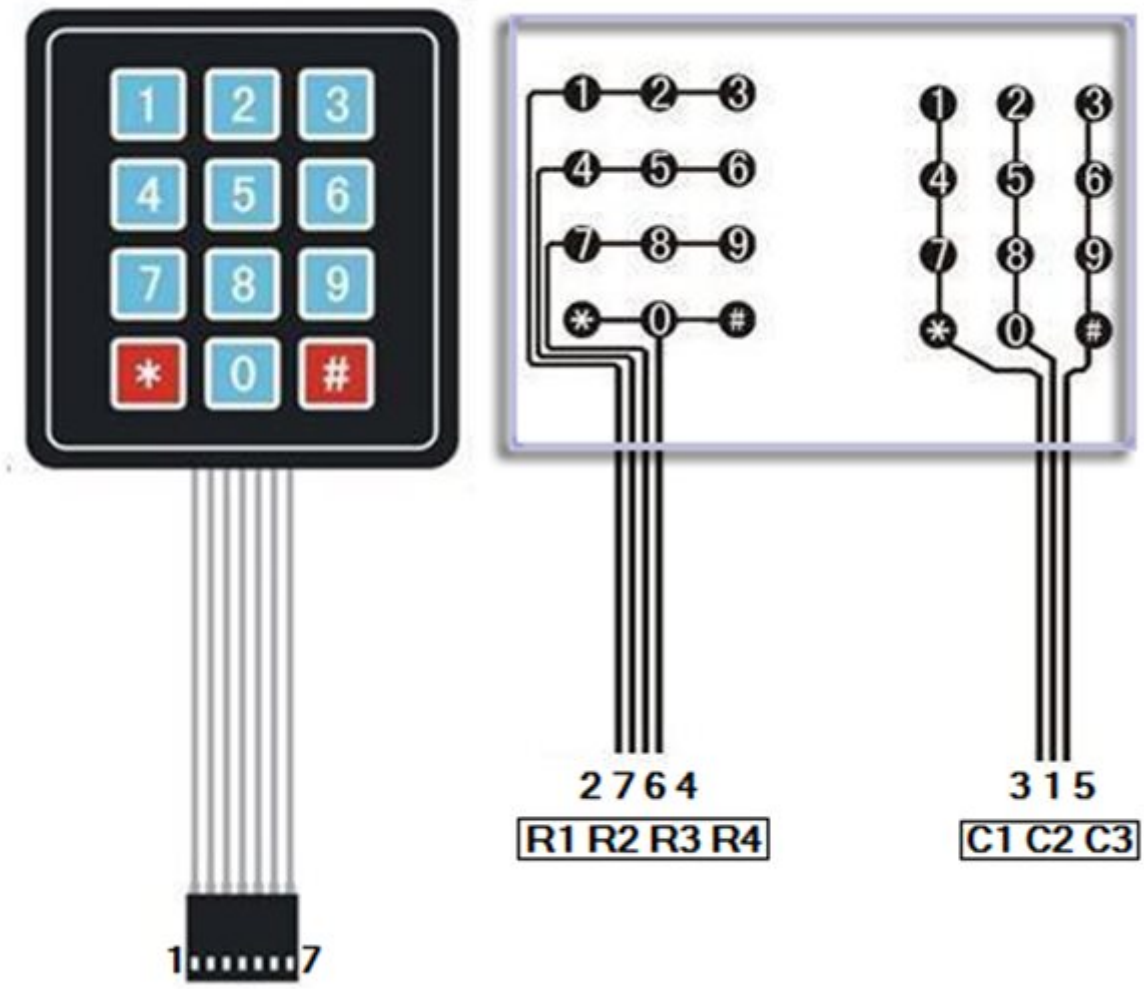
## Keypad

Un keypad c'est juste une série de bouton. En théorie il faut donc un fil par bouton pour récupérer chaque signal (et donc savoir sur quelle touche on appui). Mais pour un clavier comme ici en 4x4 ça fait donc 16 entrées à brancher : bof ...

Ici entre en jeu le matriciel : il s'agit en fait d'un genre de bataille navale. Chaque bouton, quand on appui dessus, fait contact entre une ligne et une colonne. En balayant chaque ligne on arrive donc à savoir quelle touche est pressée et lesquelles ne le sont pas. Comme une image vaut mieux qu'un long discours :




Une autre manière de le voir (plus réelle) :



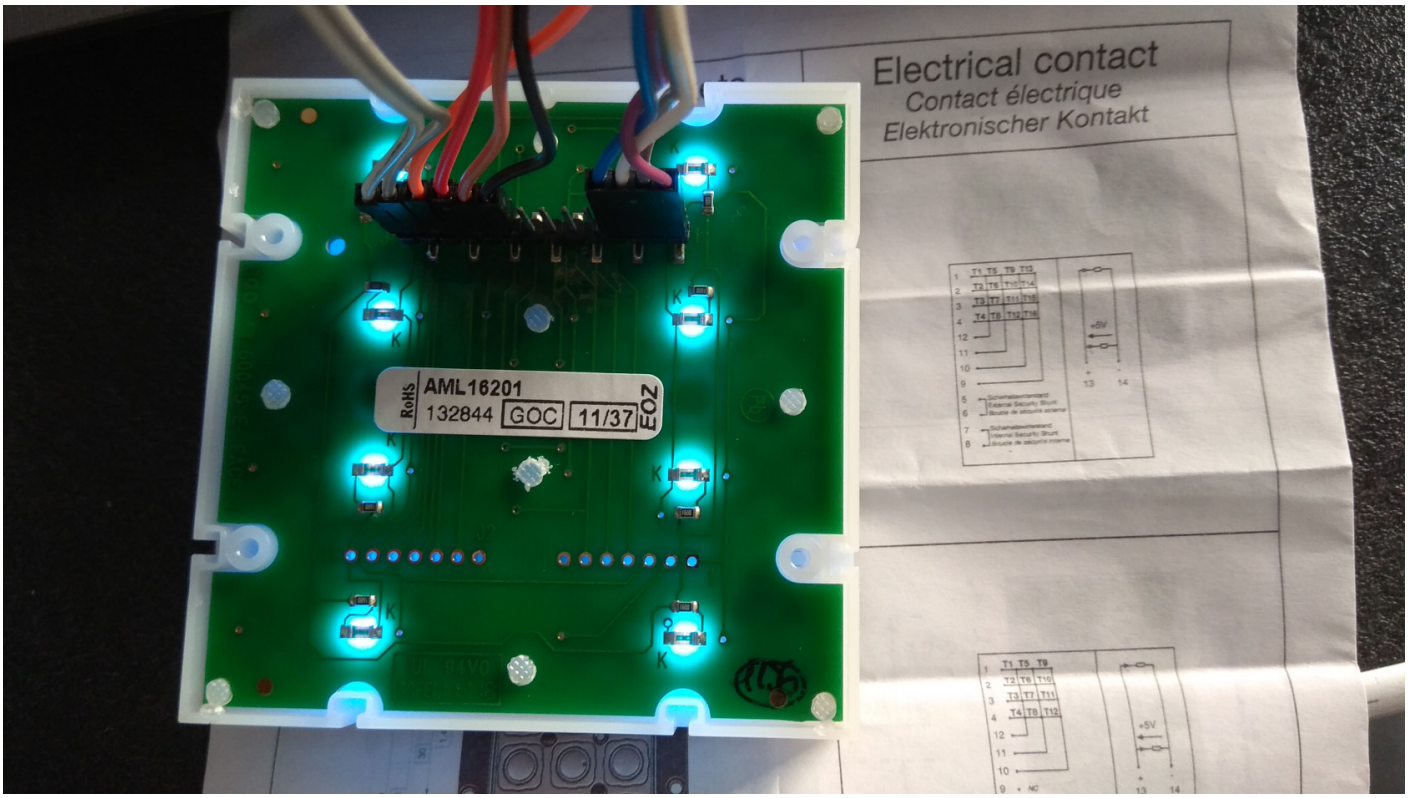

Volontairement je n'irai pas plus loin dans le détail du fonctionnement, c'est simple mais déroutant pour les débutant et si vous voulez approfondir on trouve des infos sur le net et le chat est la pour répondre à vos questions ;)

# Branchements

Les branchements sont très **simples**, surtout que la notice est filée avec le clavier (encore faut-il cependant savoir la décoder ).

**Il faut relier chaque ligne et chaque colonne à une entrée/sortie de votre Arduino** (ou compatible). C'est aussi simple que ça. Pas de résistance entre, rien ...

Sur le keypad il y a un header, sur la photo à droite sur le circuit un chiffre "**1**" et sur la gauche un chiffre "**14**" indique sa numérotation.




Quand on reprend la notice on en déduit donc :

| | |
|---|---|
| 1 | Ligne 1 |
| 2 | Ligne 2 |
| 3 | Ligne 3 |
| 4 | Ligne 4 |
| 12 | Colonne 1 |
| 11 | Colonne 2 |
| 10 | Colonne 3 |
| 9 | Colonne 4 |
| 13 | LED + |
| 14 | LED - |

**Les pins 5. 6, 7 & 8 sont juste des ponts pour détecter l'arrachement du clavier** (tentative de vol ou de piratage) nous n'en avons pas besoin.

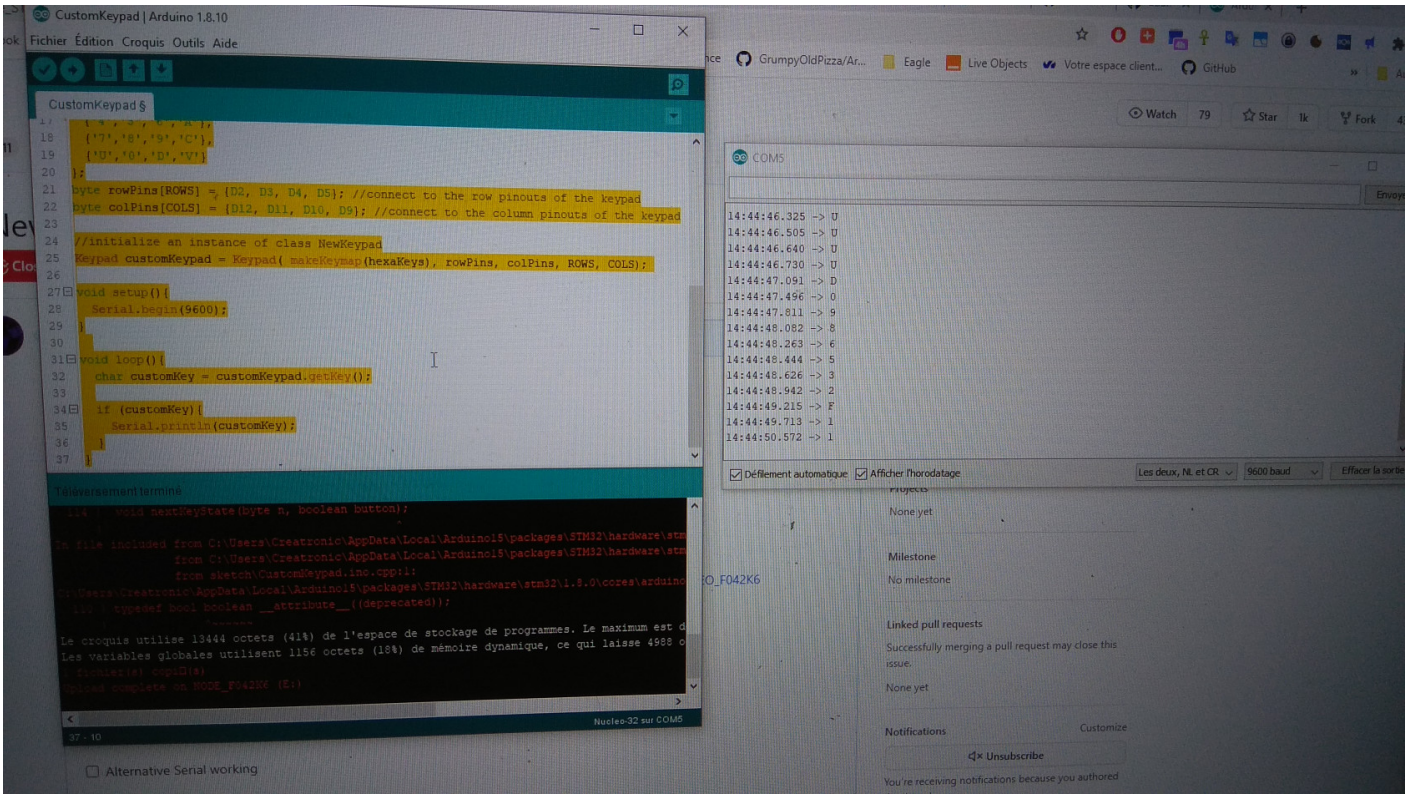
**Les deux pins d'alimentation des LED sont optionnels** (éclairera votre clavier en bleu). Il suffit de brancher le LED + au 5V et le LED - au GND.

# Le code

Ici on va utiliser une librairie très ancienne, qui est Keypad. Elle nous simplifie énormément la vie, j'ai simplement repris un des codes exemple :

```
#include <Keypad.h>

const byte ROWS = 4; //nombre de lignes
const byte COLS = 4; //nombre de colonne
//definition des touches
char hexaKeys[ROWS][COLS] = {
  {'1','2','3','F'},
  {'4','5','6','A'},
  {'7','8','9','C'},
  {'U','0','D','V'}
};
byte rowPins[ROWS] = {D2, D3, D4, D5}; //pins connectes aux lignes
byte colPins[COLS] = {D12, D11, D10, D9}; //pins connectes aux colonnes

//on initialise l'objet keypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup(){
  Serial.begin(9600); // demarrage de la liaison serie
}

void loop(){
  char customKey = customKeypad.getKey(); // recuperation de la touche pressee (ou pas)

  if (customKey){ // une touche a ete presse ?
    Serial.println(customKey); // je l'envois sur le port serie
  }
}
```

J'ai commenté le code, je pense qu'il n'y a pas grand chose à ajouter ... Il faut tester □□




Pas beau ça ?

# Les limites

Le multiplexage entraîne des limites. La principale étant que **le "mutlitouch" est impossibl**e : si j'appuie sur plusieurs bouton en même temps qui appartiennent à la même ligne ou colonne, il est impossible pour le système de savoir  qui est appuyé.

Autre point important : lors de l'appel de getKey(), le système scan toutes les entrées et c'est là qu'il sait si une touche est pressée ou pas. En d'autre terme : il n'y a qu'à ce moment la qu'un appui peut-être détecté. En d'autres termes encore : si vous faites d'autres choses à côté (par exemple qui implique un gros delay()) vous risquez de ne pas détecter un éventuel appui. **getKey() doit donc être appelée le plus souvent possible.**

Vous rentrez la dans ce qu'on appel les **problématiques de "temps réel"** c'est à dire comment je fais plusieurs choses en même temps sur un tout petit processeur qui ne sait gérer qu'une tâche.

# Conclusion

Je suis vraiment surpris par ce keypad car **il est très performant** : il génère des signaux très propres (ce qui n'est pas toujours le cas) et je n'ai expérimenté aucune erreur.

C'est un basique en Arduino, à la portée de tout le monde, rapide à faire, n'hésitez pas à en prendre un dans le carton pour essayer c'est fait pour ça

Revision #4
Created 2020-07-17 12:49:02 UTC
Updated 2020-07-17 14:04:07 UTC