

# Programmation de l'ESP32 avec Keypad Matrix

Houssem-eddine LAHMER

# Table des matières

Introduction à l'ESP32

Comprendre le Keypad Matrix

Bibliothèque Keypad pour Arduino

Schémas de câblage

Exemples pratiques

Simulation Wokwi

Exercices pratiques

Ressources supplémentaires

# Présentation de l'ESP32

- ▶ Microcontrôleur double cœur développé par Espressif Systems
- ▶ Fréquence CPU jusqu'à 240 MHz
- ▶ Wi-Fi et Bluetooth intégrés
- ▶ 520 Ko de SRAM
- ▶ 34 GPIO configurables
- ▶ Conversion analogique-numérique 12 bits
- ▶ Support pour le PWM, I<sup>2</sup>C, SPI, UART

# Avantages de l'ESP32

## Points forts

- ▶ Excellent rapport performance/prix
- ▶ Connectivité sans fil intégrée
- ▶ Grande communauté et support
- ▶ Faible consommation d'énergie

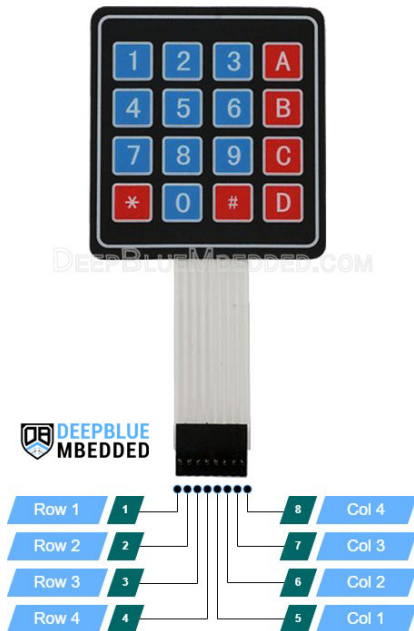
## Applications idéales

- ▶ IoT et domotique
- ▶ Contrôle à distance
- ▶ Systèmes embarqués
- ▶ Prototypage rapide

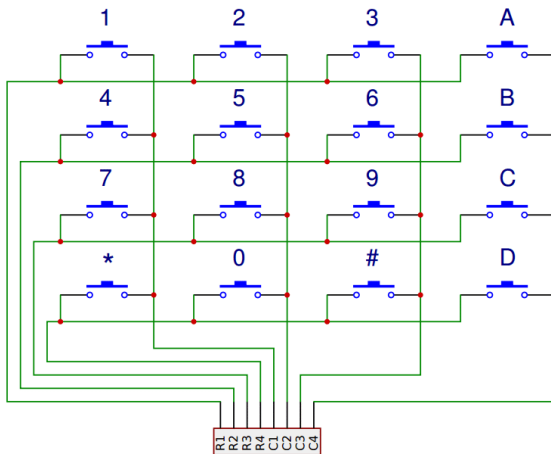
# Le Keypad Matrix - Principes

- ▶ Organisation en matrice de lignes et colonnes
- ▶ Économie de broches GPIO ( $4 \times 4 = 8$  broches au lieu de 16)
- ▶ Fonctionnement par balayage (scanning)
- ▶ Types courants :  $4 \times 4$  (16 touches) et  $4 \times 3$  (12 touches)

# Le Keypad Matrix - Principes



# Structure interne d'un Keypad Matrix



**Note :** Chaque touche est une intersection entre une ligne et une colonne

# Brochage du Keypad

## Keypad 4×4

- ▶ 8 broches au total
- ▶ 4 broches pour les lignes
- ▶ 4 broches pour les colonnes


## Keypad 4×3

- ▶ 7 broches au total
- ▶ 4 broches pour les lignes
- ▶ 3 broches pour les colonnes



# Installation de la bibliothèque Keypad

1. Ouvrir l'IDE Arduino
2. Aller à Outils ↗ Gérer les bibliothèques...
3. Rechercher "Keypad"
4. Installer la bibliothèque par Mark Stanley et Alexander Brevig

A screenshot of the Arduino IDE Libraries Manager window. The window has a title bar and a menu bar. The main area shows a list of installed libraries. The text "keypad\_library.png" is visible in the list. At the bottom right, there are navigation icons for the library manager.

keypad\_library.png

## Utilisation de base - Configuration

```
1 #include <Keypad.h>
2
3 #define ROWS 4
4 #define COLS 4
5
6 char keyMap[ROWS][COLS] = {
7     {'1', '2', '3', 'A'},
8     {'4', '5', '6', 'B'},
9     {'7', '8', '9', 'C'},
10    {'*', '0', '#', 'D'}
11 };
12
13 uint8_t rowPins[ROWS] = {14, 27, 26, 25}; //
14         GPIO pins
15
16 uint8_t colPins[COLS] = {33, 32, 18, 19}; //
17         GPIO pins
18
19 Keypad keypad = Keypad(makeKeymap(keyMap),
20                           rowPins, colPins, ROWS,
21                           COLS);
```

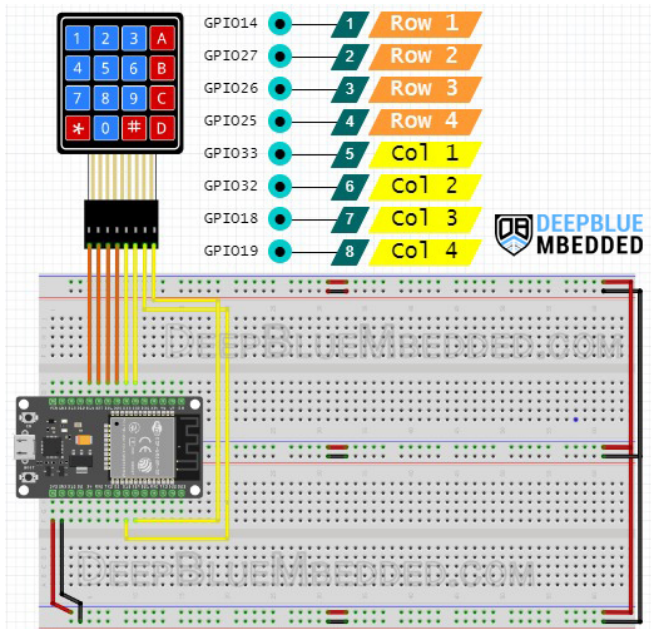
# Utilisation de base - Lecture

```
1 void setup() {  
2   Serial.begin(115200);  
3 }  
4  
5 void loop() {  
6   char key = keypad.getKey();  
7  
8   if (key) {  
9     Serial.println(key);  
10  }  
11 }
```

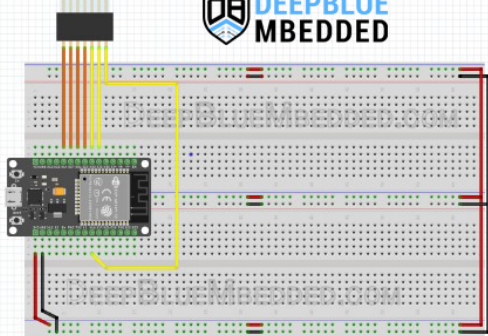
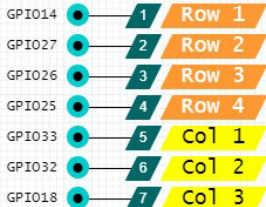
# Fonctions utiles de la bibliothèque Keypad

- ▶ `getKey()` - Renvoie la touche actuellement pressée
- ▶ `setDebounceTime(unsigned int time)` - Règle le temps d'anti-rebond
- ▶ `getState()` - Renvoie l'état actuel des touches
- ▶ `setHoldTime(unsigned int time)` - Règle le temps pour état "maintenu"
- ▶ `addEventListener(keypadEvent)` - Attache un gestionnaire d'événements
- ▶ `keyStateChanged()` - Vérifie si l'état a changé
- ▶ `waitForKey()` - Attend qu'une touche soit pressée (bloquant)

# ESP32 avec Keypad 4×4



# ESP32 avec Keypad 4x3



## Exemple1 : Démo simple (1/4)

```
1 #include <Keypad.h>
2
3 #define ROWS 4
4 #define COLS 4
```

## Exemple1 : Démo simple (2/4)

```
1 char keyMap[ROWS][COLS] = {  
2     {'1', '2', '3', 'A'},  
3     {'4', '5', '6', 'B'},  
4     {'7', '8', '9', 'C'},  
5     {'*', '0', '#', 'D'}  
6 };  
7  
8 uint8_t rowPins[ROWS] = {14, 27, 26, 25};  
9 uint8_t colPins[COLS] = {33, 32, 18, 19};
```



## Exemple1 : Démo simple (3/4)

```
1 Keypad keypad = Keypad(  
2   makeKeymap(keyMap),  
3   rowPins, colPins,  
4   ROWS, COLS  
5 );  
6  
7 void setup() {  
8   Serial.begin(115200);  
9 }
```

## Exemple1 : Démo simple (4/4)

```
1 void loop() {  
2     char key = keypad.getKey();  
3     if (key) {  
4         Serial.println(key);  
5     }  
6 }
```

## Exemple2 : Keypad + LCD I2C (1/5)

```
1 #include <Keypad.h>           // gestion du
    clavier matriciel :contentReference[oaicite
    :0]{index=0}
2 #include <Wire.h>             // bus I2C :
    contentReference[oaicite:1]{index=1}
3 #include <LiquidCrystal_I2C.h> // cran LCD via
    I2C :contentReference[oaicite:2]{index=2}
4
5 #define ROWS 4
6 #define COLS 4
```

## Exemple2 : Keypad + LCD I2C (2/5)

```
1 // disposition des touches
2 char keyMap[ROWS][COLS] = {
3     {'1', '2', '3', 'A'},
4     {'4', '5', '6', 'B'},
5     {'7', '8', '9', 'C'},
6     {'*', '0', '#', 'D'}
7 };
8
9 // broches GPIO pour le keypad : contentReference
   [oaicite:3]{index=3}
10 uint8_t rowPins[ROWS] = {14,27,26,25};
11 uint8_t colPins[COLS] = {33,32,18,19};
12
13 // position du curseur LCD
14 uint8_t LCD_CursorPosition = 0;
15
16 // instantiation des objets
17 Keypad keypad = Keypad(makeKeymap(keyMap),
   rowPins, colPins, ROWS, COLS);
18 LiquidCrystal_I2C lcd(0x27,16,2); // adresse 0
```

## Exemple2 : Keypad + LCD I2C (3/5)

```
1 void setup() {  
2   Serial.begin(115200); // pour debug  
3   lcd.begin();          // initialise l'cran  
4   I2C :contentReference[oaicite:5]{index=5}  
5   lcd.backlight();      // active le  
6   r t r o clairage  
7   lcd.clear();          // vide l'affichage  
8 }
```

## Exemple2 : Keypad + LCD I2C (4/5)

```
1 void loop() {
2   char key = keypad.getKey(); // recupere la
   touche (0 si aucune) :contentReference[
   oaicite:6]{index=6}
3
4   if (key) {
5     Serial.print(key);          // affiche sur
   moniteur s rie
6     // on fera l affichage LCD dans la diapo
   suivante
7   }
8 }
```

## Exemple2 : Keypad + LCD I2C (5/5)

```
1    lcd.setCursor(LCD_CursorPosition++, 0);
2    if (LCD_CursorPosition == 16 || key == 'D')
3    {
4        lcd.clear();                // nouvelle
        ligne ou touche      D      reset
5        LCD_CursorPosition = 0;
6    } else {
7        lcd.print(key);            // affiche le
        caract re
8    }
9 }
```

## Exemple3 : Verrouillage (1/6)

```
1 #include <Keypad.h>
2 #include <Wire.h>
3 #include <LiquidCrystal_I2C.h>
4
5 #define ROWS 4
6 #define COLS 4
```



## Exemple3 : Verrouillage (2/6)

```
1 char keyMap[ROWS][COLS] = {
2     {'1', '2', '3', 'A'},
3     {'4', '5', '6', 'B'},
4     {'7', '8', '9', 'C'},
5     {'*', '0', '#', 'D'}
6 };
7
8 uint8_t rowPins[ROWS] = {14, 27, 26, 25};
9 uint8_t colPins[COLS] = {33, 32, 18, 19};
10
11 uint8_t LCD_CursorPosition = 0;
12 String PassWord = "134679";
13 String InputStr = "";
```

## Exemple3 : Verrouillage (3/6)

```
1 Keypad keypad = Keypad(  
2   makeKeymap(keyMap),  
3   rowPins, colPins,  
4   ROWS, COLS  
5 );  
6 LiquidCrystal_I2C lcd(0x27,16,2);
```

## Exemple3 : Verrouillage (4/6)

```
1 void setup() {  
2   Serial.begin(115200);  
3   lcd.begin();  
4   lcd.backlight();  
5   lcd.clear();  
6   lcd.setCursor(0,0);  
7   lcd.print("Entrez le code:");  
8 }
```

## Exemple3 : Verrouillage (5/6)

```
1 void loop() {  
2     char key = keypad.getKey();  
3     if (key) {  
4         InputStr += key;  
5         lcd.setCursor(LCD_CursorPosition++,1);  
6         lcd.print('*'); // masque la saisie
```

## Exemple3 : Verrouillage (6/6)

```
1      if (LCD_CursorPosition == 6) {
2          if (InputStr == PassWord)
3              lcd.print("Acces Autorise!");
4          else
5              lcd.print("Code Incorrect!");
6          InputStr = "";
7          LCD_CursorPosition = 0;
8      } else if (key == 'D') {
9          InputStr = "";
10         lcd.clear();
11         LCD_CursorPosition = 0;
12         lcd.print("Entrez le code:");
13     }
14 }
15 }
```

# Simulation avec Wokwi

- ▶ Wokwi est un simulateur en ligne pour Arduino, ESP32, etc.
- ▶ Accès : <https://wokwi.com>
- ▶ Pas besoin de matériel physique pour tester
- ▶ Permet de créer des circuits virtuels
- ▶ Supporte le code Arduino et les bibliothèques populaires

# Créer un projet ESP32 sur Wokwi

1. Accéder à Wokwi et cliquer sur "Nouveau projet"
2. Sélectionner "ESP32"
3. Ajouter les composants depuis la barre latérale :
  - ▶ Keypad 4×4 ou 4×3
  - ▶ LCD I2C (si nécessaire)
4. Connecter les composants selon le schéma
5. Écrire ou coller le code Arduino
6. Cliquer sur "Démarrer la simulation"

# Exercice 1 : Calculatrice simple

**Objectif :** Créer une calculatrice simple avec le Keypad et un écran LCD

- ▶ Utiliser les touches 0-9 pour entrer des nombres
- ▶ Utiliser A, B, C, D pour les opérations (+, -, \*, /)
- ▶ Afficher l'opération et le résultat sur l'écran LCD
- ▶ Utiliser '\*' pour effacer et '' pour calculer



## Exercice 2 : Contrôle d'accès avancé

**Objectif :** Créer un système de contrôle d'accès avec plusieurs utilisateurs

- ▶ Stocker plusieurs mots de passe dans un tableau
- ▶ Permettre à l'utilisateur de choisir son ID
- ▶ Enregistrer les accès (heure et ID utilisateur)
- ▶ Bonus : Utiliser la mémoire flash pour stocker les mots de passe

## Exercice 3 : Jeu de Simon

**Objectif :** Créer un jeu de mémoire style "Simon" avec le Keypad

- ▶ Utiliser 4 LEDs de couleurs différentes
- ▶ Générer des séquences aléatoires
- ▶ Utiliser le Keypad pour que le joueur répète la séquence
- ▶ Augmenter la difficulté à chaque niveau

# Ressources pour aller plus loin

- ▶ Documentation officielle ESP32 :  
<https://docs.espressif.com/>
- ▶ GitHub de la bibliothèque Keypad :  
<https://github.com/Chris--A/Keypad>
- ▶ Tutoriels DeepBlueMbedded :  
<https://deepbluembedded.com/>
- ▶ Exemples de projets avec ESP32 :  
<https://randomnerdtutorials.com/>
- ▶ Simulateur Wokwi : <https://wokwi.com/>

# Conclusion

- ▶ L'ESP32 est un microcontrôleur puissant et polyvalent
- ▶ Le Keypad Matrix est une solution efficace pour les interfaces utilisateur
- ▶ La combinaison ESP32 + Keypad permet de nombreuses applications :
  - ▶ Systèmes de contrôle d'accès
  - ▶ Alarmes et systèmes de sécurité
  - ▶ Interfaces homme-machine (IHM)
  - ▶ Calculatrices et consoles de jeu
- ▶ N'hésitez pas à expérimenter et adapter ces exemples !

# Merci pour votre attention !

Des questions ?