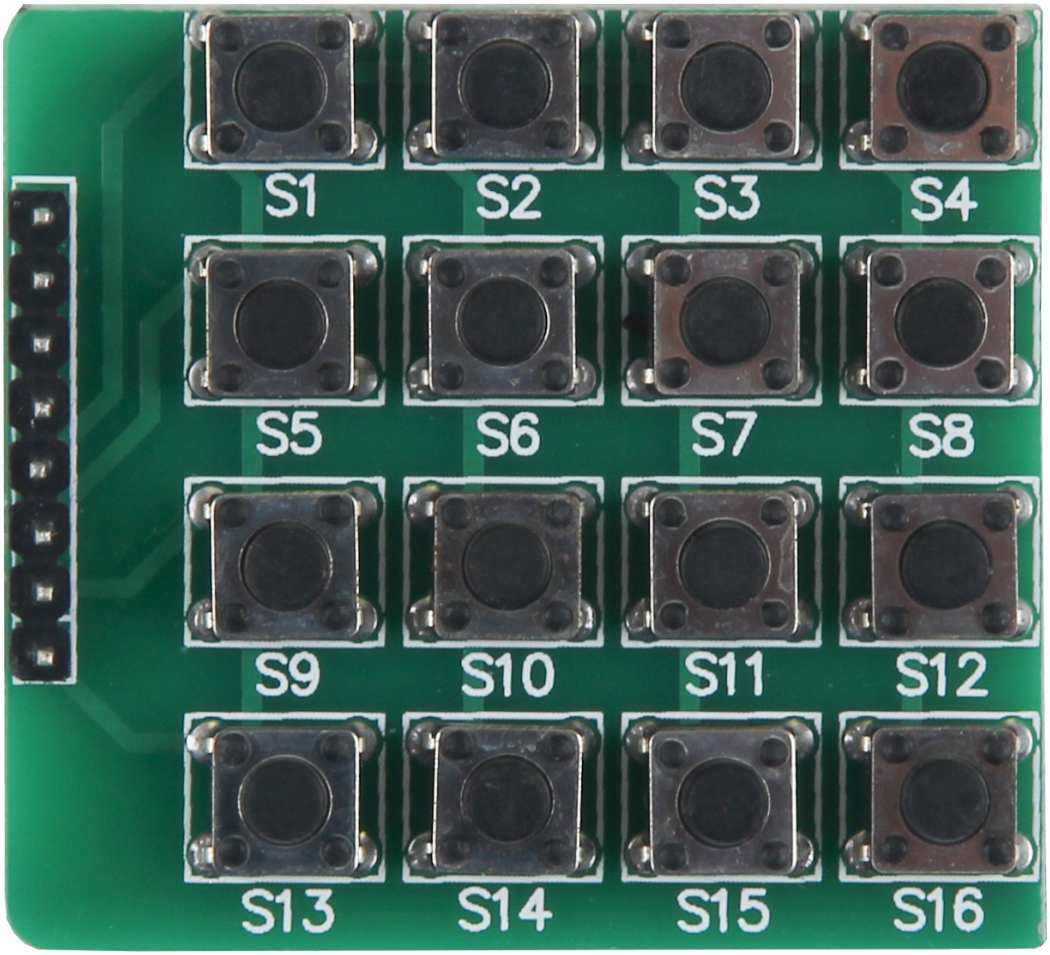
BUTTON MATRIX

4 x 4 Knöpfe



1. ALLGEMEINE INFORMATIONEN

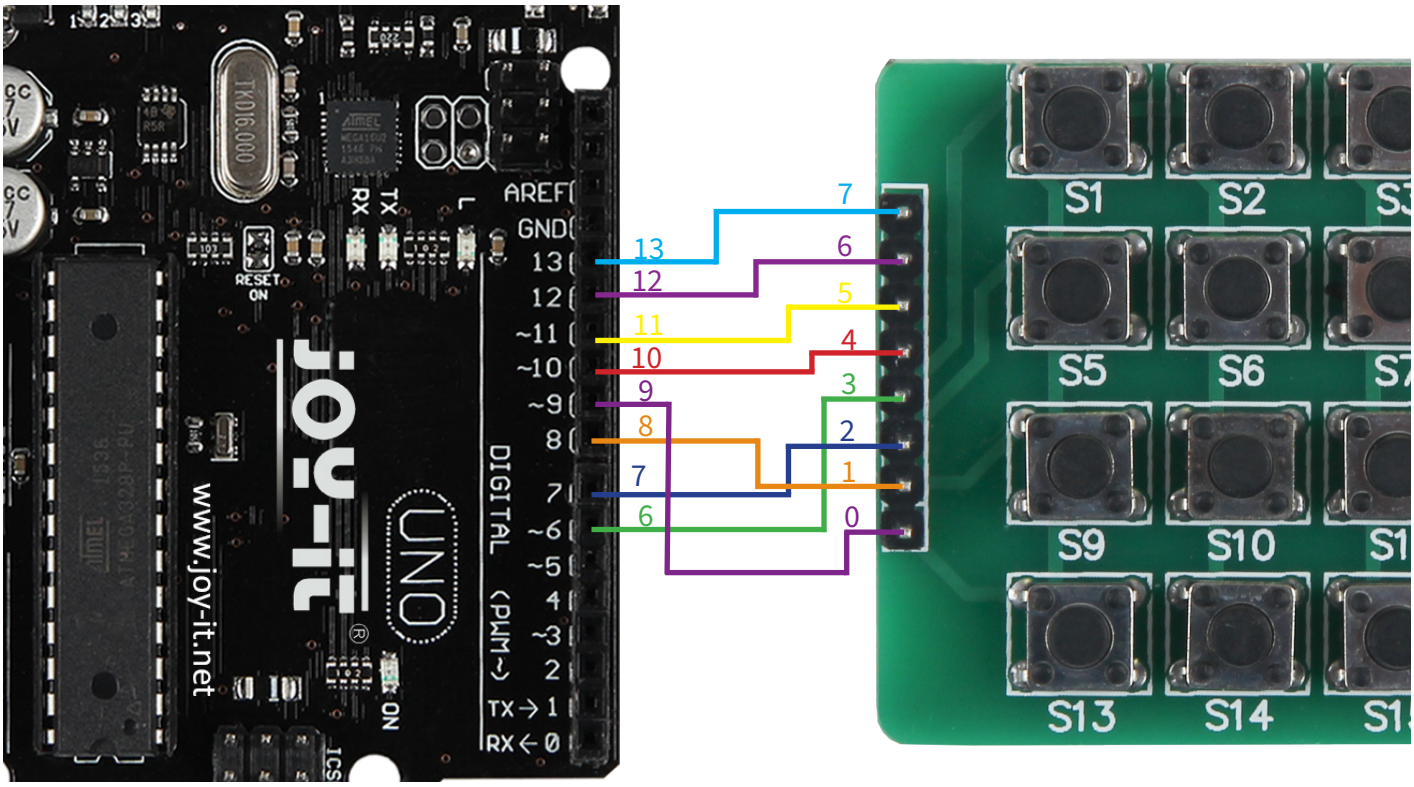
Sehr geehrter Kunde,

vielen Dank, dass Sie sich für unser Produkt entschieden haben. Im Folgenden zeigen wir Ihnen, was bei der Inbetriebnahme und der Verwendung zu beachten ist.

Sollten Sie während der Verwendung unerwartet auf Probleme stoßen, so können Sie uns selbstverständlich gerne kontaktieren.

2. VERWENDUNG MIT ARDUINO

Anschließen der Matrix



Pinnummer Arduino	Pinnummer ButtonMatrix
13	7
12	6
11	5
10	4
9	0
8	1
7	2
6	3

In diesem Beispiel wurde der Arduino Uno mit der 4 x 4 ButtonMatrix angeschlossen.

Schließen Sie die ButtonMatrix, wie in dem obigen Bild und der Tabelle zu sehen ist, an die digitalen Pins (6 - 13) des Arduinos an.

Installation der Matrix

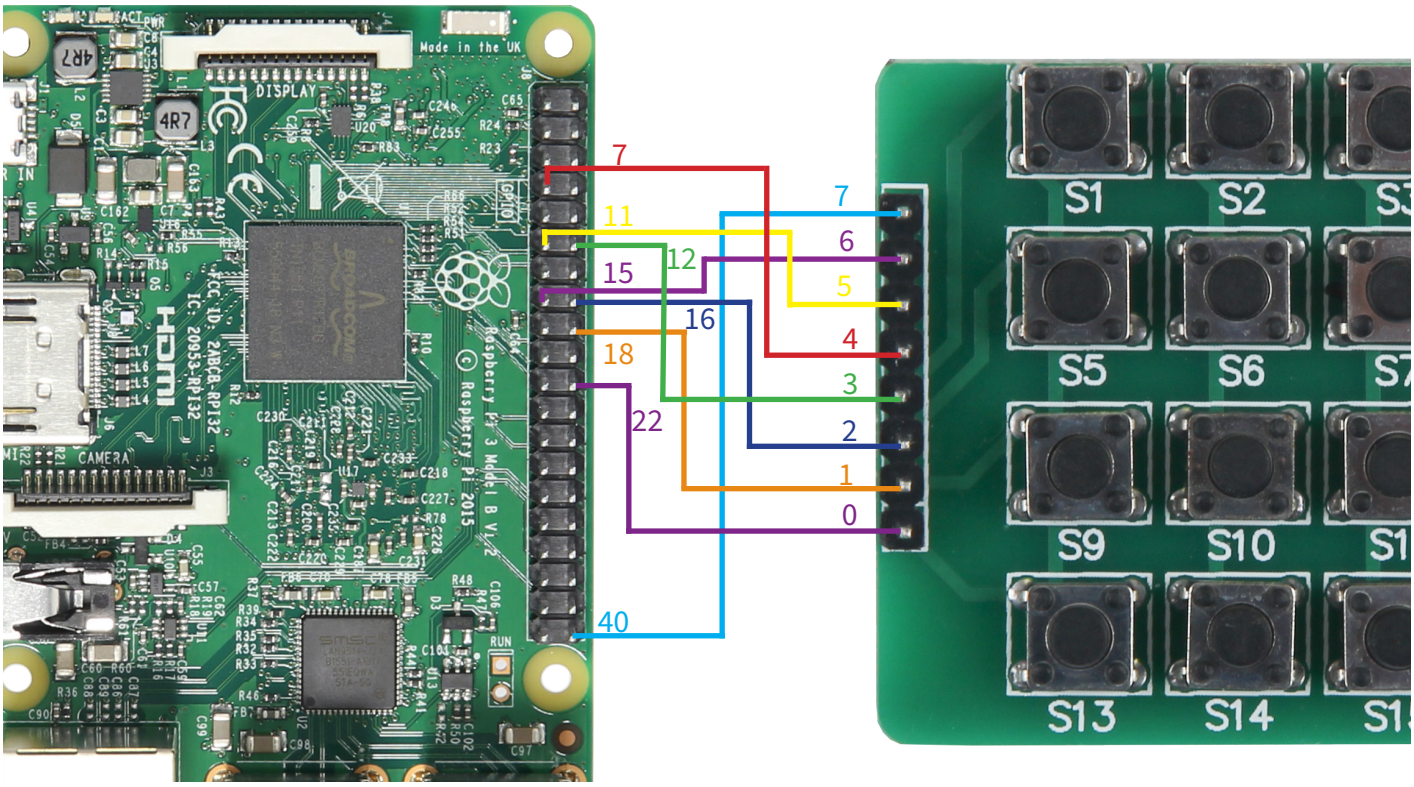
Nachfolgend können Sie ein funktionsfähiges Codebeispiel entnehmen und auf Ihren Arduino übertragen.
Die Funktion der vorhandenen Knöpfe der ButtonMatrix können Sie in der **knopfDruck**-Funktion nach Ihren Wünschen erweitern.

```
// ------------------------------
// Pin-Zuordnung
// ------------------------------
const byte rowPins[4] = {6, 7, 8, 9};      // R1 -> D6, R2 -> D7, R3 -> D8, R4 -> D9
const byte colPins[4] = {10, 11, 12, 13}; // C1 -> D10, C2 -> D11, C3 -> D12, C4 -> D13

// ------------------------------
// Tastenbelegung als 2D-Array
// (Strings, damit „10“, „11“ etc. möglich sind)
// ------------------------------
const char* keys[4][4] = {
  {"1",  "5",  "9",  "13"},
  {"2",  "6",  "10", "14"},
  {"3",  "7",  "11", "15"},
  {"4",  "8",  "12", "16"}
};

// ------------------------------
// Hilfs-Array, um den Tastenzustand
// (gedrückt / nicht gedrückt) zu merken
// ------------------------------
bool lastState[4][4] = {
  false, false, false, false,
  false, false, false, false,
  false, false, false, false,
  false, false, false, false
};

void setup() {
  Serial.begin(9600);
  Serial.println("Starte 4x4-Keypad-Scan ohne Bibliothek...");

  // Spalten-Pins als OUTPUT, HIGH = Ruhezustand
  for (byte c = 0; c < 4; c++) {
    pinMode(colPins[c], OUTPUT);
    digitalWrite(colPins[c], HIGH);
  }

  // Reihen-Pins als INPUT_PULLUP
  for (byte r = 0; r < 4; r++) {
    pinMode(rowPins[r], INPUT_PULLUP);
  }
```

```
void loop() {
  // Spalten nacheinander ansteuern
  for (byte col = 0; col < 4; col++) {
    // Aktive Spalte auf LOW setzen
    digitalWrite(colPins[col], LOW);

    // Reihen abfragen
    for (byte row = 0; row < 4; row++) {
      // Taster ist gedrückt, wenn rowPin auf LOW geht
      bool isPressed = (digitalRead(rowPins[row]) == LOW);

      // Nur ausgeben, wenn der Zustand sich von „nicht
gedrückt“ auf „gedrückt“ ändert
      if (isPressed && !lastState[row][col]) {
        Serial.print("Taste gedrueckt: ");
        Serial.println(keys[row][col]);
      }

      // Speichere aktuellen Zustand
      lastState[row][col] = isPressed;
    }

    // Spalte wieder auf HIGH setzen
    digitalWrite(colPins[col], HIGH);
  }

  // Kleine Verzögerung gegen Prellen / Entlastung der
Schleife
  delay(50);
}
```

3. VERWENDUNG MIT RASPBERRY PI

Anschließen der Matrix



Pin Raspberry Pi	Pinnummer ButtonMatrix
PIN 40 (BCM 21)	7
PIN 15 (BCM 22)	6
PIN 11 (BCM 17)	5
PIN 7 (BCM 4)	4
PIN 12 (BCM 18)	3
PIN 16 (BCM 23)	2
PIN 18 (BCM 24)	1
PIN 22 (BCM 25)	0

In diesem Beispiel wurde der Raspberry Pi 3 Modell B mit der 4 x 4 ButtonMatrix angeschlossen.

Schließen Sie die ButtonMatrix, wie in dem obigen Bild und der Tabelle zu sehen ist, an die Pins des Raspberry Pis an.

Installation der Matrix

Zunächst muss eine neue Python-Datei erstellt werden:

```
nano matrix.py
```

Schreiben Sie das folgende Codebeispiel vollständig in den Editor, der sich nun geöffnet hat.

```
import RPi.GPIO as GPIO
import time
class ButtonMatrix():
        def __init__(self):
                GPIO.setmode(GPIO.BCM)
                # Die IDs der Buttons werden festgelegt
                self.buttonIDs = [[4,3,2,1],[8,7,6,5],[12,11,10,9],[16,15,14,13]]
                # GPIO Pins fuer die Zeilen werden deklariert.
                self.rowPins = [18,23,24,25]
                # GPIO Pins fuer die Spalte werden deklariert.
                self.columnPins = [21,22,17,4]
                # Definiere Vier Inputs mit pull up Widerstaenden.
                for i in range(len(self.rowPins)):
                        GPIO.setup(self.rowPins[i],GPIO.IN,pull_up_down=GPIO.PUD_UP)
                        # Definiere Vier Outputs und setze sie auf high.
                for j in range(len(self.columnPins)):
                        GPIO.setup(self.columnPins[j], GPIO.OUT)
                        GPIO.output(self.columnPins[j], 1)
        def activateButton(self, rowPin, colPin):
        # Erhalte die Button Nummer
                btnIndex = self.buttonIDs[rowPin][colPin] - 1
                print("button " + str(btnIndex + 1) + " pressed")
                # Verhindert mehrere Knopfdruecke in zu kurzer zeit
                time.sleep(.3)
        def buttonHeldDown(self,pin):
                if(GPIO.input(self.rowPins[pin]) == 0):
                        return True
                return False
def main():
        # Initialisierung der Button Matrix
        buttons = ButtonMatrix()
        try:
                while(True):
                        for j in range(len(buttons.columnPins)):
                        # Jeder Output Pin wird auf low gesetzt.
                                GPIO.output(buttons.columnPins[j],0)
                                for i in range(len(buttons.rowPins)):
                                        if GPIO.input(buttons.rowPins[i]) == 0:
                                                buttons.activateButton(i,j)
                # Nichts tun solange der Button gedrueckt gehalten wird.
                                                while(buttons.buttonHeldDown(i)):
                                                        pass
                                GPIO.output(buttons.columnPins[j],1)
        except KeyboardInterrupt:
                GPIO.cleanup()
if __name__ == "__main__":
  main()
```

Die Datei kann mit **Strg+O** gespeichert und der Editor mit **Strg+X** verlassen werden. Anschließend kann die Datei mit folgendem Befehl ausgeführt und somit getestet werden.

```
python3 matrix.py
```

Die Datei kann mit **Strg+C** wieder verlassen werden.

4. WEITERE INFORMATIONEN

Unsere Informations- und Rücknahmepflichten nach dem Elektrogesetz (ElektroG)

Symbol auf Elektro- und Elektronikgeräten:
Diese durchgestrichene Mülltonne bedeutet, dass Elektro- und Elektronikgeräte **nicht** in den Hausmüll gehören. Sie müssen die Altgeräte an einer Erfassungsstelle abgeben. Vor der Abgabe haben Sie Alt-batterien und Altakkumulatoren, die nicht vom Altgerät umschlossen sind, von diesem zu trennen.

Rückgabemöglichkeiten:
Als Endnutzer können Sie beim Kauf eines neuen Gerätes, Ihr Altgerät (das im Wesentlichen die gleiche Funktion wie das bei uns erworbene neue erfüllt) kostenlos zur Entsorgung abgeben. Kleingeräte bei denen keine äußere Abmessungen größer als 25 cm sind können unabhängig vom Kauf eines Neugerätes in Haushaltsüblichen Mengen abgeben werden.

Möglichkeit Rückgabe an unserem Firmenstandort während der Öffnungszeiten:
SIMAC GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

Möglichkeit Rückgabe in Ihrer Nähe:
Wir senden Ihnen eine Paketmarke zu mit der Sie das Gerät kostenlos an uns zurücksenden können. Hierzu wenden Sie sich bitte per E-Mail an Service@joy-it.net oder per Telefon an uns.

Informationen zur Verpackung:
Verpacken Sie Ihr Altgerät bitte transportsicher, sollten Sie kein geeignetes Verpackungsmaterial haben oder kein eigenes nutzen möchten kontaktieren Sie uns, wir lassen Ihnen dann eine geeignete Verpackung zukommen.

CE

5. SUPPORT

Wir sind auch nach dem Kauf für Sie da. Sollten noch Fragen offen bleiben oder Probleme auftauchen stehen wir Ihnen auch per E-Mail, Telefon und Ticket-Supportsystem zur Seite.

E-Mail: service@joy-it.net
Ticket-System: https://support.joy-it.net
Telefon: +49 (0)2845 9360 – 50

Für weitere Informationen besuchen Sie unsere Website:
www.joy-it.net

Veröffentlicht: 2025.06.02