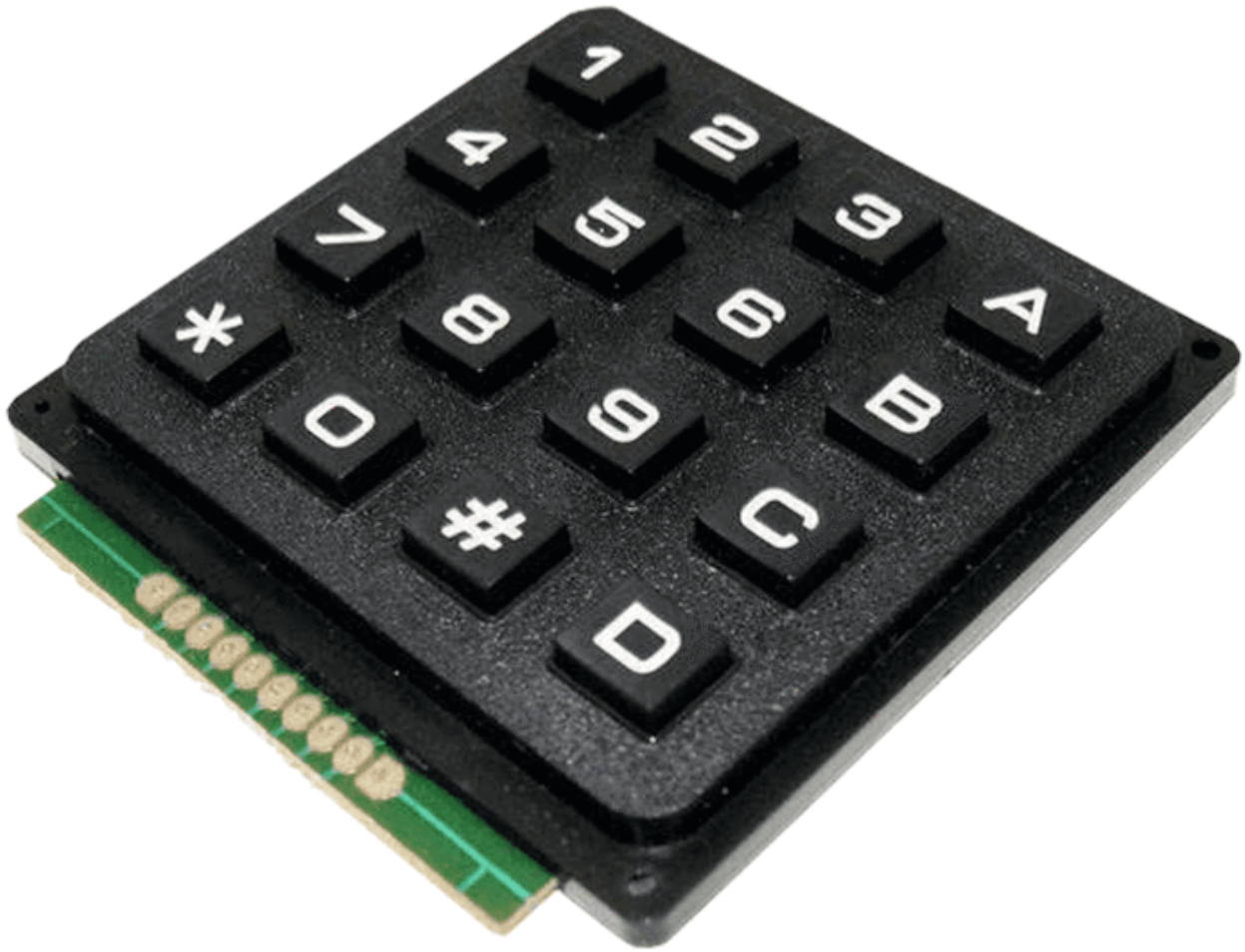
Willkommen!

Vielen Dank, dass sie sich für unser AZ-Delivery 4x4 Matrix Keyboard entschieden haben. In den nachfolgenden Seiten werden wir Ihnen erklären wie Sie das Gerät einrichten und nutzen können.

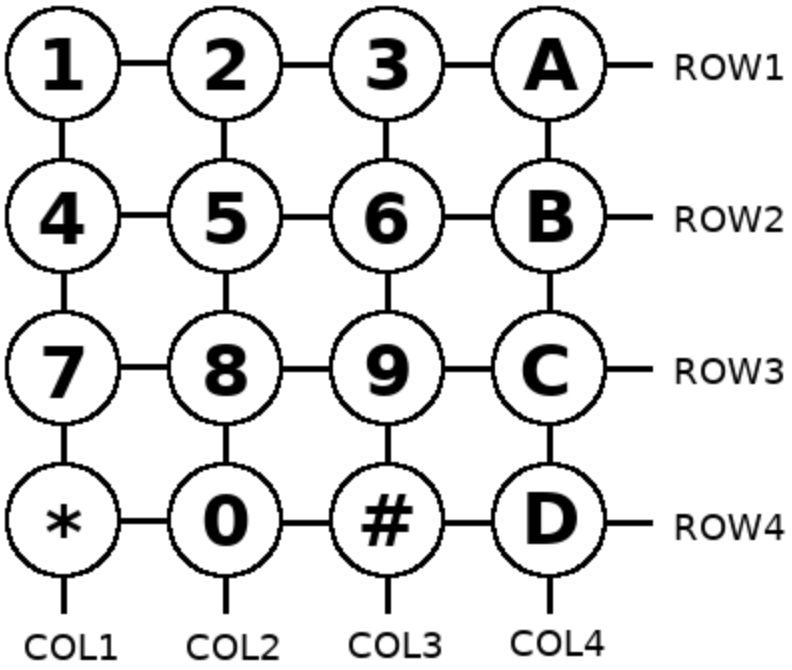
Viel Spaß!



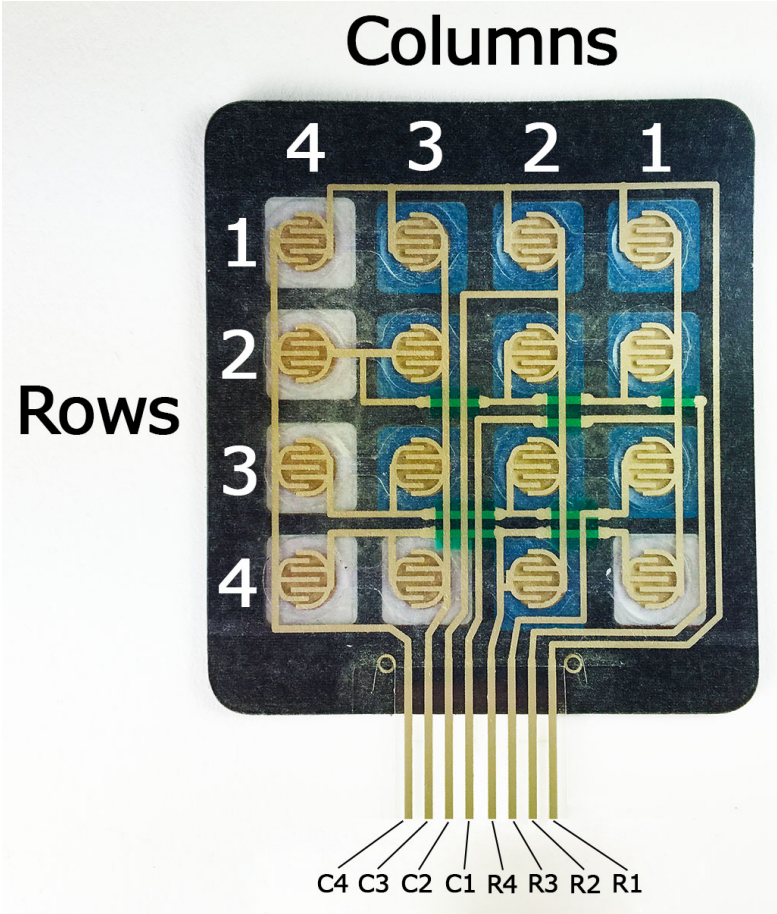
Keypads ermöglichen die Interaktion mit einer Vielzahl von Projekten . Mit ihnen können Sie Menüs navigieren, Passwörter eingeben, Spiele und Roboter steuern.

Wie funktioniert das Keypad?

Die Tasten auf dem Keypad sind in Zeilen und Spalten angeordnet. Eine 4x4-Tastatur hat 4 Zeilen und 4 Spalten:



Unter jeder Taste befindet sich ein Folientaster. Jeder Schalter einer Reihe ist mit den anderen Schaltern derselben Reihe durch eine leitfähige Spur unter dem Pad verbunden.



Jeder Schalter in einer Spalte ist auf die gleiche Weise verbunden. Eine Seite des Schalters ist mit allen anderen Schaltern dieser Spalte durch eine leitfähige Spur verbunden.

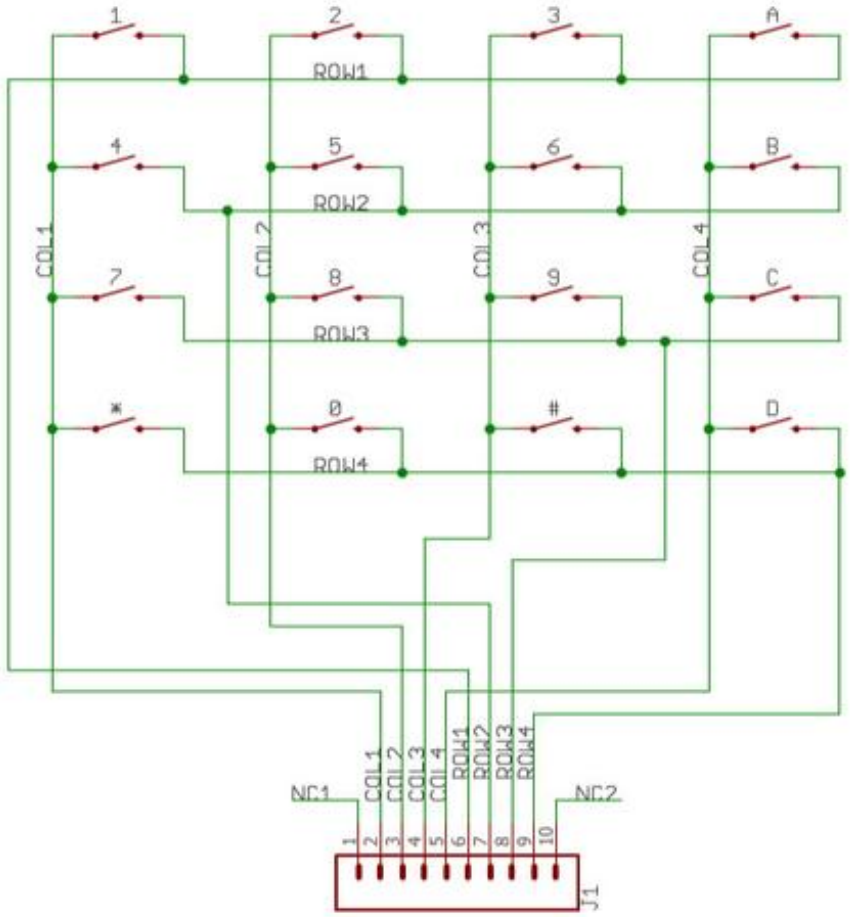
Jede Zeile und Spalte ist mit Ausgabepins verbunden.

Insgesamt sind 8 Pins auf einem 4x4-Keypad:

NC - Not Connected
C - Column
R - Row

Das Drücken einer Taste schließt den Schalter zwischen der Spalten- und Zeilenspur, so dass Strom zwischen einem Spaltenpin und einem Zeilenpin fließen kann.

Der Schaltplan des 4x4-Keypads zeigt, wie die Zeilen und Spalten verbunden sind:



Der Arduino erkennt, welche Taste gedrückt wird, indem er den Zeilen- und Spaltenpin erkennt, der mit der Taste verbunden ist. Dazu sind vier Schritte nötig:

1. Werden keine Tasten gedrückt werden, werden alle Spaltenstifte HIGH und alle Reihenstifte LOW geschaltet.

2. Wird eine Taste gedrückt, dann schaltet der Spaltenstift zu LOW, da der Strom von der HIGH-Spalte zum LOW-Reihenpin fließt.

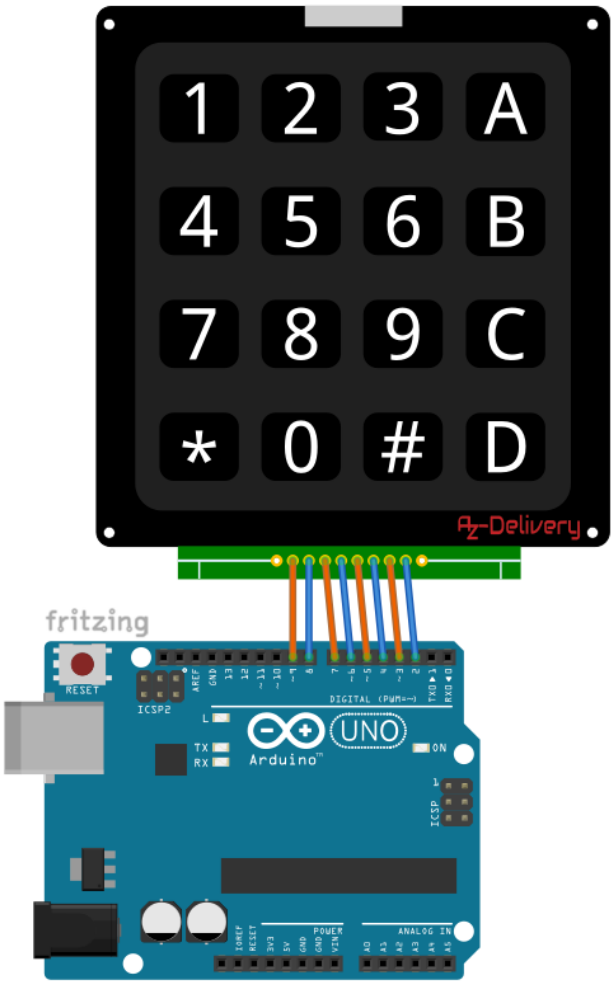
3. Der Arduino weiß nun, in welcher Spalte die Taste gedrückt wird, so dass er nun nur noch die Zeile finden muss, in der die Taste gedrückt wird. Dazu wird jeder der Reihenstifte auf HIGH geschaltet und gleichzeitig alle Spaltenstifte gelesen. So wird erkannt, welcher Spaltenstift zu HIGH zurückkehrt.

4. Wenn der Spaltenstift wieder auf HIGH steht, hat der Arduino den Reihenstift gefunden, der mit der Taste verbunden ist:

Eine Kombination von beispielsweise Reihe 2 und Spalte 2 kann nur bedeuten, dass die Taste 5 gedrückt wurde.

Verbindung des Keypads mit dem Arduino Uno

Verbinden Sie das Keypad mit dem Uno wie unten abgebildet:



Spalten			**Reihen**		
Keypad Pin	**>**	**Uno Pin**	**Keypad Pin**	**>**	**Uno Pin**
C1	>	D9	R1	>	D5
C2	>	D8	R2	>	D4
C3	>	D7	R3	>	D3
C4	>	D6	R4	>	D2

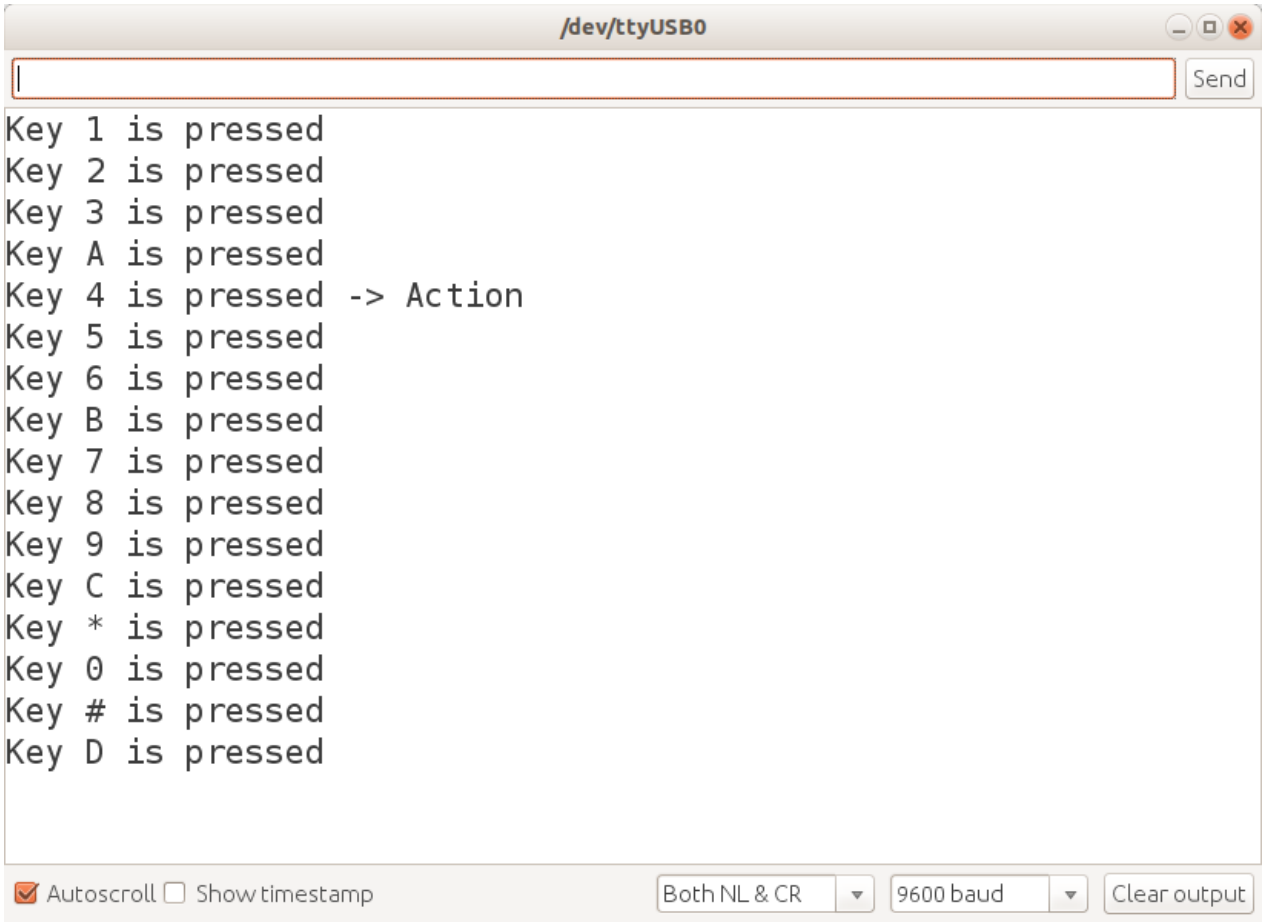
Skizze für Uno

Bevor wir eine Skizze für den Arduino Uno machen können, müssen wir die Library für dieses Keypad runterladen und importieren. Gehen Sie zu diesem Link: http://robojax.com/learn/arduino/robojax-Keypad.zip und downloaden die ".zip"-Datei. Um sie zu Ihrem Uno hinzuzufügen: *Sketch > Include Library > Add .ZIP Library* and add downloaded .zip file.

Skizzen-Code:

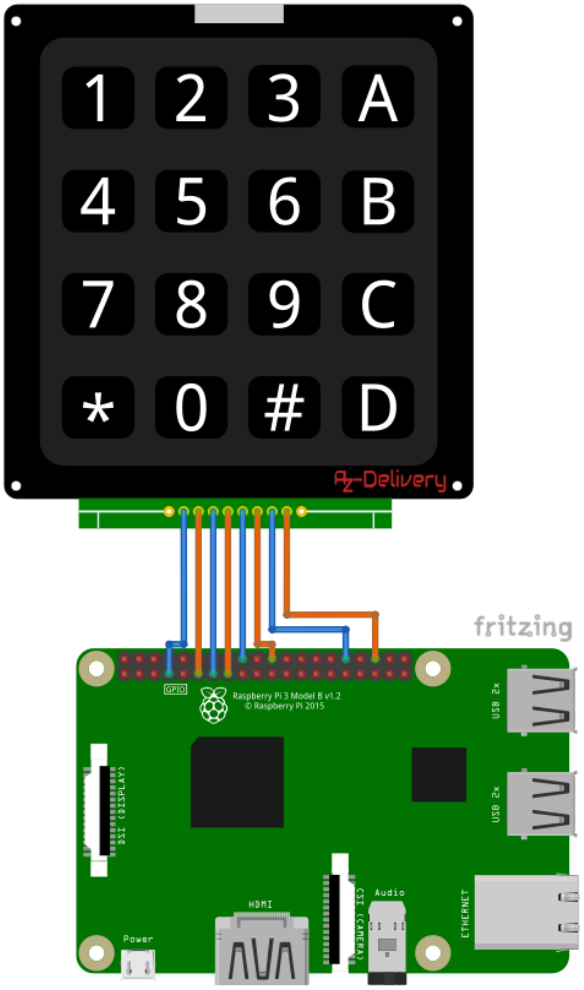
```
#include <Keypad.h>
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {5,4,3,2};
byte colPins[COLS] = {9,8,7,6};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
void setup(){
  Serial.begin(9600);
}
void loop(){
  char key = keypad.getKey();
  if(key){
    if(key == '4'){
       // do something when button 4 is pressed
       Serial.println("Key 4 is pressed -> Something");
    }
    else {
      Serial.print("Key ");
      Serial.print(key);
      Serial.print(" is pressed");
    }
  } }
```

Und wenn Sie diese Skizze in Ihren Uno hochladen, starten Sie den Serial-Monitor (Tools > Serial Monitor). Wenn Sie die Tasten auf der Tastatur drücken, sollte die Ausgabe in Serial-Monitor so aussehen:



Verbindung des keypad mit dem Raspberry Pi

Verbinden Sie das Keypad mit dem Raspberry Pi wie unten abgebildet:



Spalten			**Reihen**		
Keypad Pin	**>**	**RaspPi Pin**	**Keypad Pin**	**>**	**RaspPi Pin**
C1	>	GPIO4 [Pin 7]	R1	>	GPIO24 [Pin 18]
C2	>	GPIO17 [Pin 11]	R2	>	GPIO25 [Pin 22]
C3	>	GPIO27 [Pin 13]	R3	>	GPIO12 [Pin 32]
C4	>	GPIO22 [Pin 15]	R4	>	GPIO16 [Pin 36]

Skript für den Raspberry Pi

Wir haben das Skript modifiziert:
https://github.com/rainierez/MatrixKeypad_Python/blob/master/matrix_keypad/RPi_GPIO.py
damit es mit dem 4x4 Matrix Keypad funktioniert. Das Skript aus dem Link hat einige Fehler (in diesem eBook werden keine Korrekturen behandelt; wir zeigen Ihnen nur das korrekte Skript).

Für dieses Beispiel, werden wir zwei Skripte erstellen, Eins für das keypad-object, und ein Weiteres, um zu zeigen, wie das Keypad funktioniert (Main Skript). Code für das "class script":

```
#!/usr/bin/python
import RPi.GPIO as GPIO
class keypad():
    def __init__(self, columnCount = 3):
        GPIO.setmode(GPIO.BCM)
        GPIO.setwarnings(False)
        if columnCount is 3:
            self.KEYPAD = [ # CONSTANT
                [1,2,3,"A"],
                [4,5,6,"B"],
                [7,8,9,"C"],
                ["*",0,"#","D"]]
            self.ROW = [24, 25, 12, 16]
            self.COLUMN = [4, 17, 27]

        elif columnCount is 4:
            self.KEYPAD = [ # CONSTANTS
                [1,2,3,"A"],
                [4,5,6,"B"],
```

```
                [7,8,9,"C"],
                ["*",0,"#","D"]]
# (4 tabs)
            self.ROW = [24, 25, 12, 16]
            self.COLUMN = [4, 17, 27, 22]

        else:
            return

    def getKey(self):
        # Set all columns as output low
        for j in range(len(self.COLUMN)):
            GPIO.setup(self.COLUMN[j], GPIO.OUT)
            GPIO.output(self.COLUMN[j], GPIO.LOW)

        # Set all rows as input
        for i in range(len(self.ROW)):
            GPIO.setup(self.ROW[i], GPIO.IN, pull_up_down=GPIO.PUD_UP)

        # Scan rows for pushed key/button
        # A valid key press should set "rowVal"  between 0 and 3.
        rowVal = -1
        for i in range(len(self.ROW)):
            tmpRead = GPIO.input(self.ROW[i])
            if tmpRead == 0:
                rowVal = i

        # if rowVal is not 0 thru 3 then no button was pressed and we can exit
        if rowVal < 0 or rowVal > 3:
            self.exit()
            return

        # Convert columns to input
        for j in range(len(self.COLUMN)):
            GPIO.setup(self.COLUMN[j], GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
```

```
# (2 tabs)
        # Switch the i-th row found from scan to output
        GPIO.setup(self.ROW[rowVal], GPIO.OUT)
        GPIO.output(self.ROW[rowVal], GPIO.HIGH)

        # Scan columns for still-pushed key/button
        # A valid key press should set "colVal"  between 0 and 2.
        colVal = -1
        for j in range(len(self.COLUMN)):
            tmpRead = GPIO.input(self.COLUMN[j])
            if tmpRead == 1:
                colVal = j

        # if colVal is not 0 thru 2 or 3 then no button was pressed and we can exit
        if len(self.COLUMN) == 4:
            if colVal < 0 or colVal > 3:
                self.exit()
                return

        if len(self.COLUMN) == 3:
            if colVal < 0 or colVal > 2:
                self.exit()
                return

        # Return the value of the key pressed
        self.exit()
        return self.KEYPAD[rowVal][colVal]

    def exit(self):
        # Reinitialize all rows and columns as input at exit
        for i in range(len(self.ROW)):
            GPIO.setup(self.ROW[i], GPIO.IN, pull_up_down=GPIO.PUD_UP)
        for j in range(len(self.COLUMN)):
            GPIO.setup(self.COLUMN[j], GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

```
if __name__ == '__main__':
    # Initialize the keypad class
    kp = keypad()

    # Loop while waiting for a keypress
    digit = None
    while digit == None:
        digit = kp.getKey()

    # Print the result
    print("{}".format(digit))
```

Speichern Sie das Skript unter dem Namen “*keypad4.py*”.

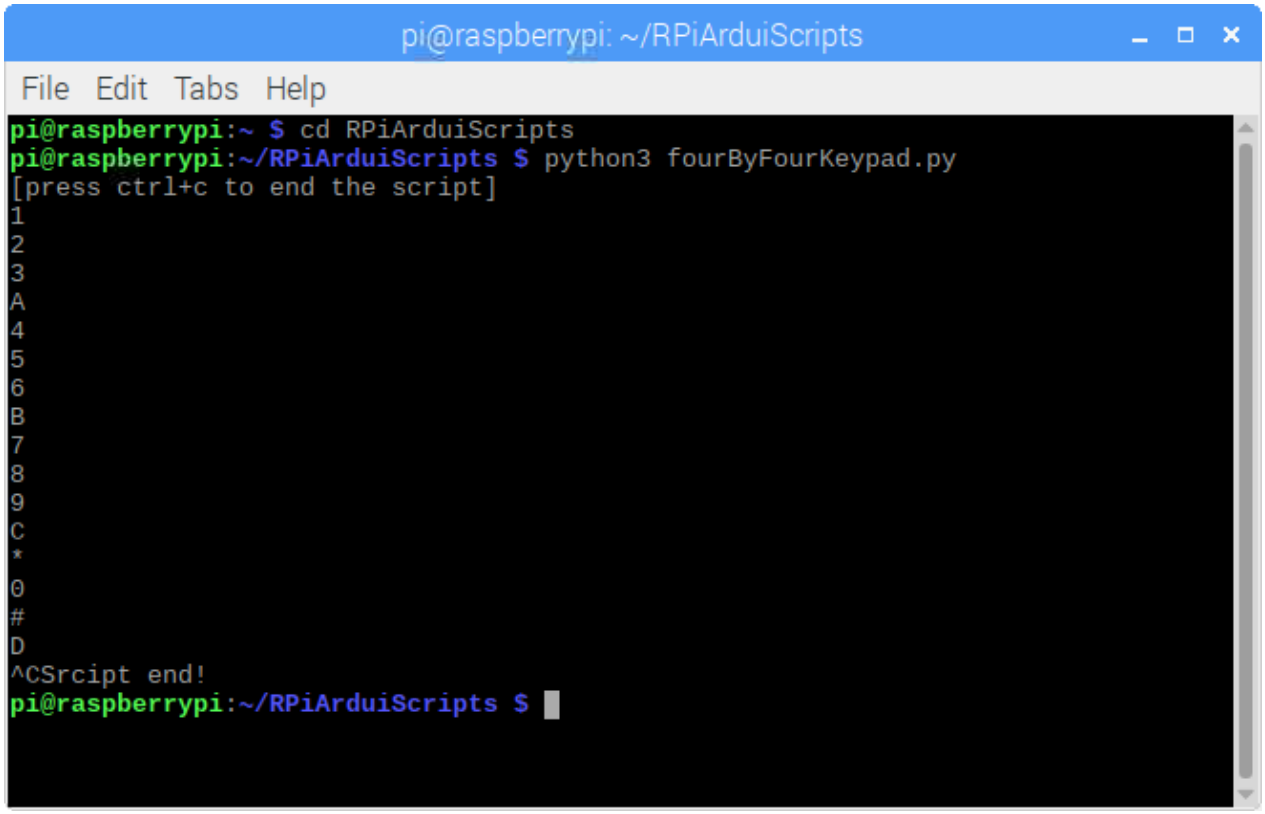
Code für das main Skript:

```
import time
import RPi.GPIO as GPIO
from keypad4 import keypad

kp = keypad(columnCount = 4)
digit = None
last_digit = None

print('[press ctrl+c to end the script]')
try: # Main program loop
   while True:
      digit = kp.getKey()
      if not last_digit == digit:
         if not digit == None:
            print("{}".format(digit))

         last_digit = digit
         time.sleep(0.5)

# Scavenging work after the end of the program
except KeyboardInterrupt:
   GPIO.cleanup()
   print('Srcipt end!')
```

Speichern Sie dieses Skript unter dem Namen “*fourByFourKeypad.py*” im selben Verzeichnis wie das “*keypad4*.py” skript.
Um das main Skript in Rasbian durchzuführen, öffnen Sie die Terminal App und führen Sie diesen Befehl aus:

`python3 foruByFourKeypad.py`

Die Ausgabe sollte so aussehen (wenn sie die Tasten auf dem Keypad drücken):



Um das Skript zu beenden, drücken Sie *CTRL* + *C*.

Sie haben es geschafft. Sie können jetzt unser Modul für Ihre Projekte nutzen.

Jetzt sind Sie dran! Entwickeln Sie Ihre eigenen Projekte und Smart-Home Installationen. Wie Sie das bewerkstelligen können, zeigen wir Ihnen unkompliziert und verständlich auf unserem Blog. Dort bieten wir Ihnen Beispielskripte und Tutorials mit interessanten kleinen Projekten an, um schnell in die Welt der Mikroelektronik einzusteigen. Zusätzlich bietet Ihnen auch das Internet unzählige Möglichkeiten, um sich in Sachen Mikroelektronik weiterzubilden.

Falls Sie nach noch weiteren hochwertigen Produkten für Arduino und Raspberry Pi suchen, sind Sie bei AZ-Delivery Vertriebs GmbH goldrichtig. Wir bieten Ihnen zahlreiche Anwendungsbeispiele, ausführliche Installationsanleitungen, E-Books, Bibliotheken und natürlich die Unterstützung unserer technischen Experten.

https://az-delivery.de

Viel Spaß!

Impressum

https://az-delivery.de/pages/about-us