



Matrix Keypad

Created by Kattni Rembor



<https://learn.adafruit.com/matrix-keypad>

Last updated on 2024-06-03 02:27:26 PM EDT

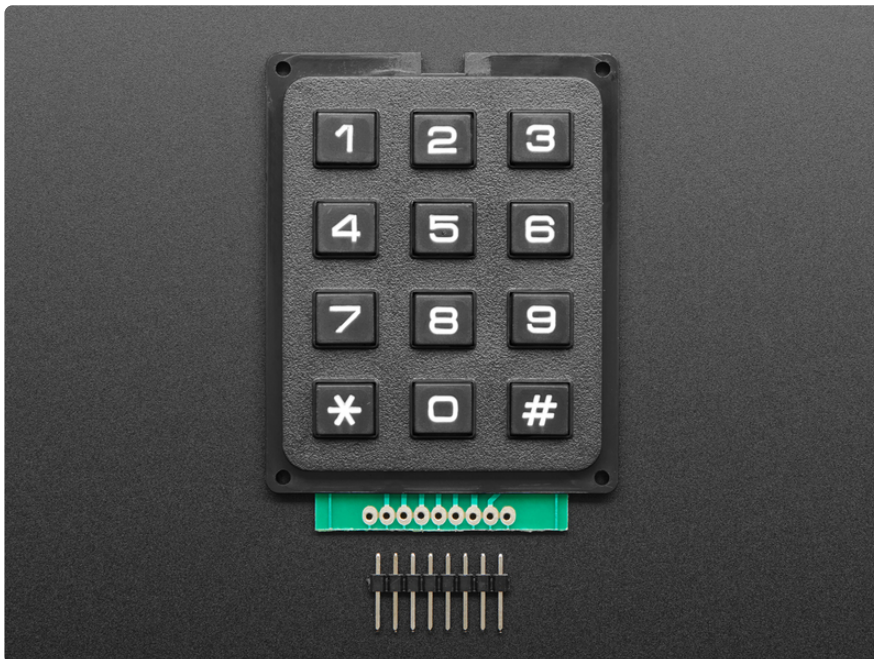
Table of Contents

Overview	3
Pinouts	4
<ul style="list-style-type: none">• General Nomenclature• 4x4 Matrix Keypad (PID 3844)• 3x4 Matrix Keypad (PID 3845)• 3x4 Phone-style Matrix Keypad (PID 1824)• Membrane 3x4 Matrix Keypad (PID 419)• Membrane 1x4 Matrix Keypad (PID 1332)	
Arduino	7
<ul style="list-style-type: none">• Install Keypad Library• Example	
Python & CircuitPython	10
<ul style="list-style-type: none">• CircuitPython Microcontroller Wiring• Python Computer Wiring• CircuitPython Installation of Matrix Keypad Library• Python Installation of Matrix Keypad Library• CircuitPython & Python Usage• Full Example Code	
Python Docs	15

Overview



Hey, Jenny, I've got your number! And I'm going to punch it into one of these nice matrix keypads. We have a bunch of different options in 3x4- and 4x4-grid styles. These are an easy way to add some input options to your project!



The keys are connected into a matrix, so for the 3x4 matrix keypads, you only need 7 microcontroller pins (3-columns and 4-rows) to scan through the pad. On the 4x4 matrix keypads, you only need 8 microcontroller pins (4-columns and 4-rows) to scan through the pad.

We have libraries in Arduino and CircuitPython to make getting started super simple and fast.



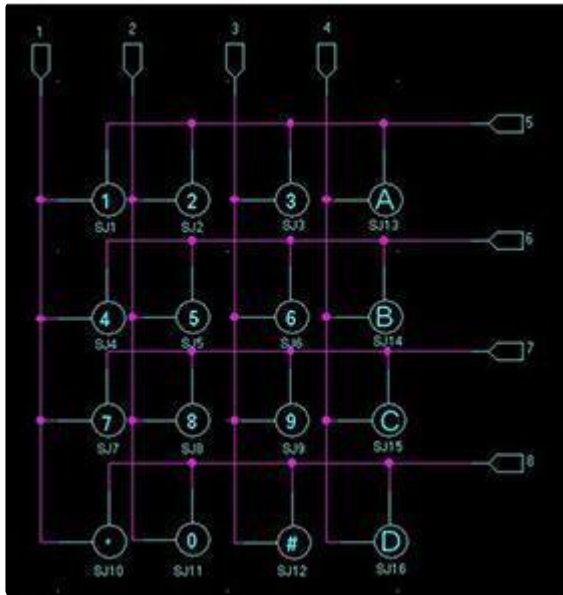
So wire it up to your project, and punch in your secret key to open all the possibilities!

Pinouts

The following covers the pinout for the various key pad products.

General Nomenclature

Each keypad is a combination of **row** and **column** circuits. The buttons themselves are simple switch like closures in these circuits. Here's an example of a basic 4x4 keypad arrangement.



There is 1 pin for each column and 1 pin for each row. So for the 4x4 keypad, there are $4 + 4 = 8$ total pins. For the 3x4 keypad, there are $3 + 4 = 7$ pins. Etc. Each of these needs to be connected to a separate digital pin on the microcontroller.

In the diagram above, the pins **1-4** correspond to the **columns** while pins **5-8** correspond to **rows**. Determining if a particular button is pressed is done by programmatically changing the digital pins between input and output and reading pin state. Doing this across all possible pin combinations is referred to as a **scan**. All of that work is typically taken care of for you in a library.

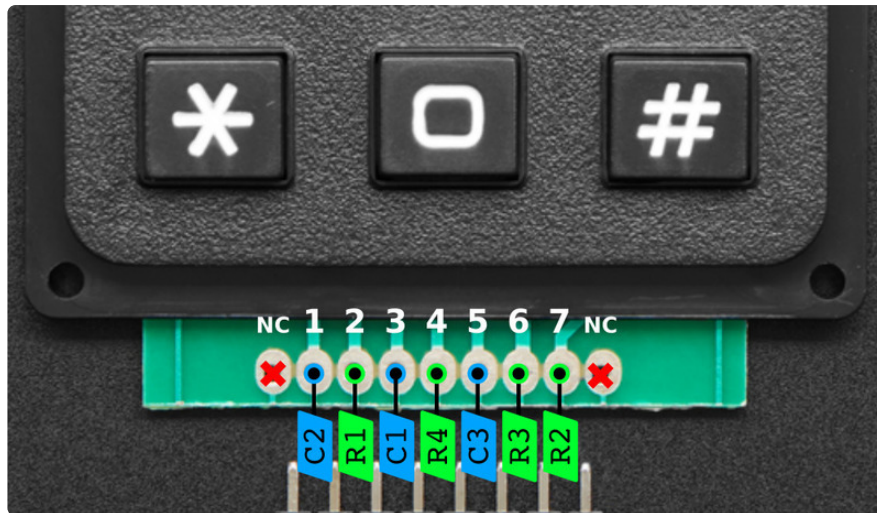
4x4 Matrix Keypad (PID 3844)

Pinout



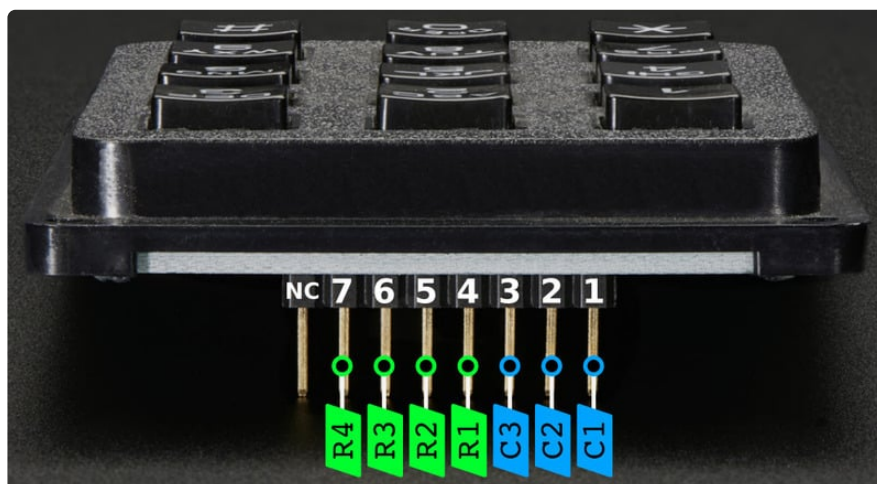
3x4 Matrix Keypad (PID 3845)

Pinout



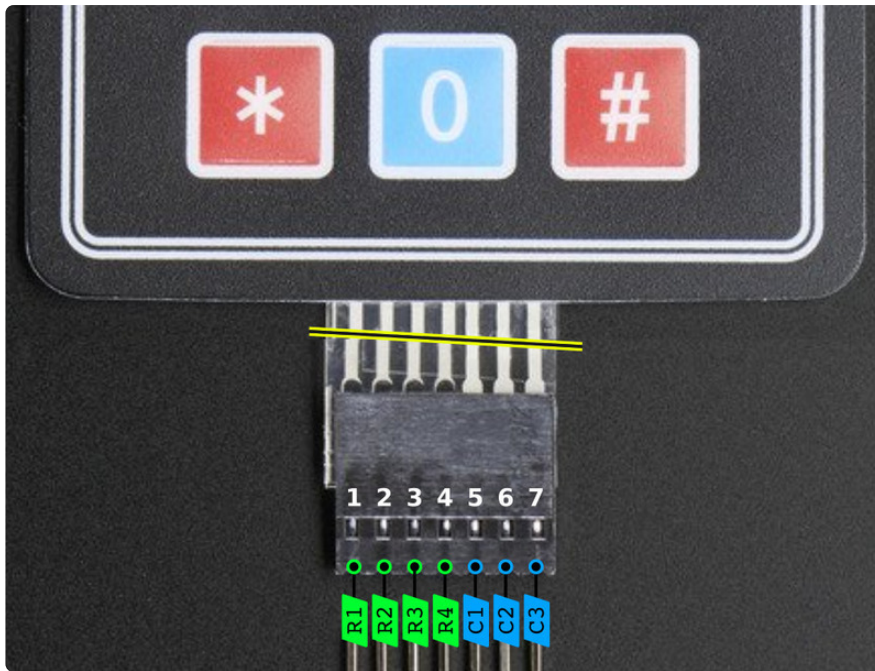
3x4 Phone-style Matrix Keypad (PID 1824)

Pinout



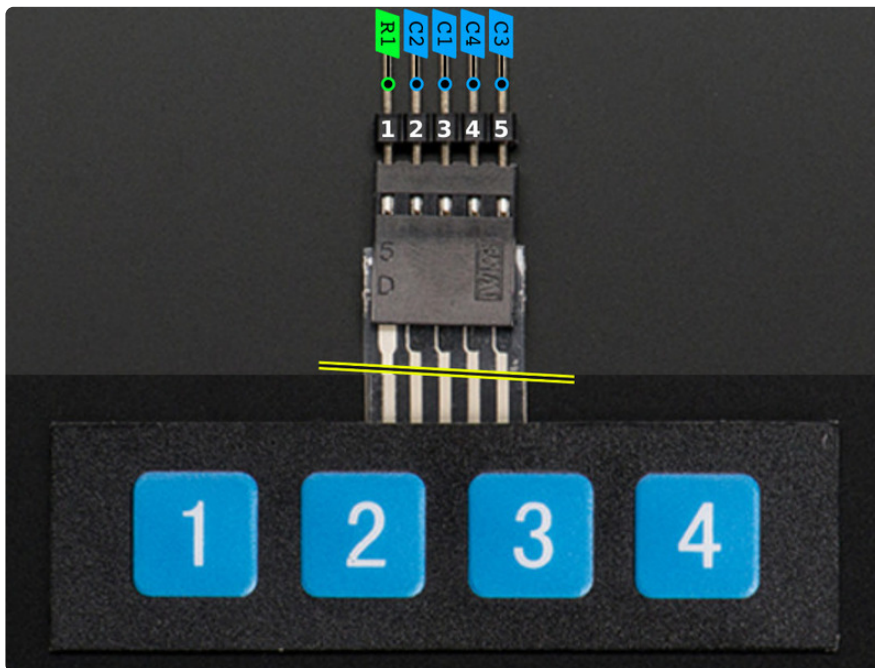
Membrane 3x4 Matrix Keypad (PID 419)

Pinout



Membrane 1x4 Matrix Keypad (PID 1332)

Pinout



Arduino

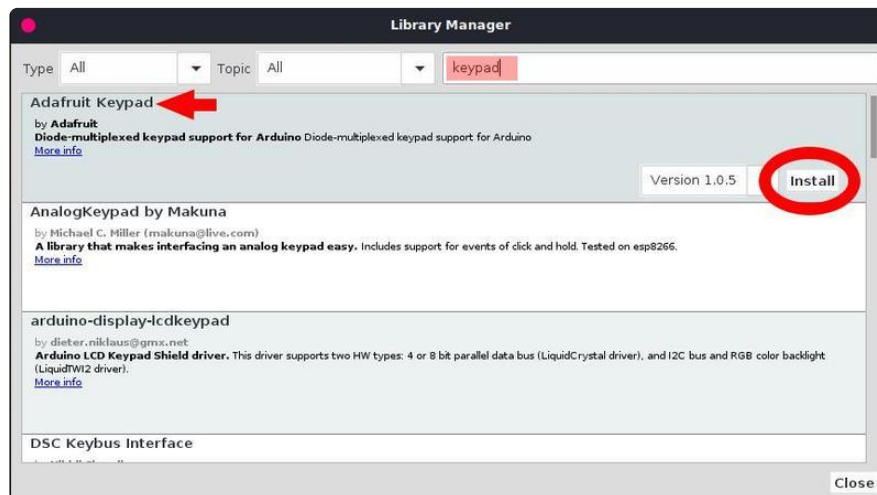
The following shows an example using the Adafruit Keypad Arduino library.

Install Keypad Library

You can install the Keypad library via the Library Manager.

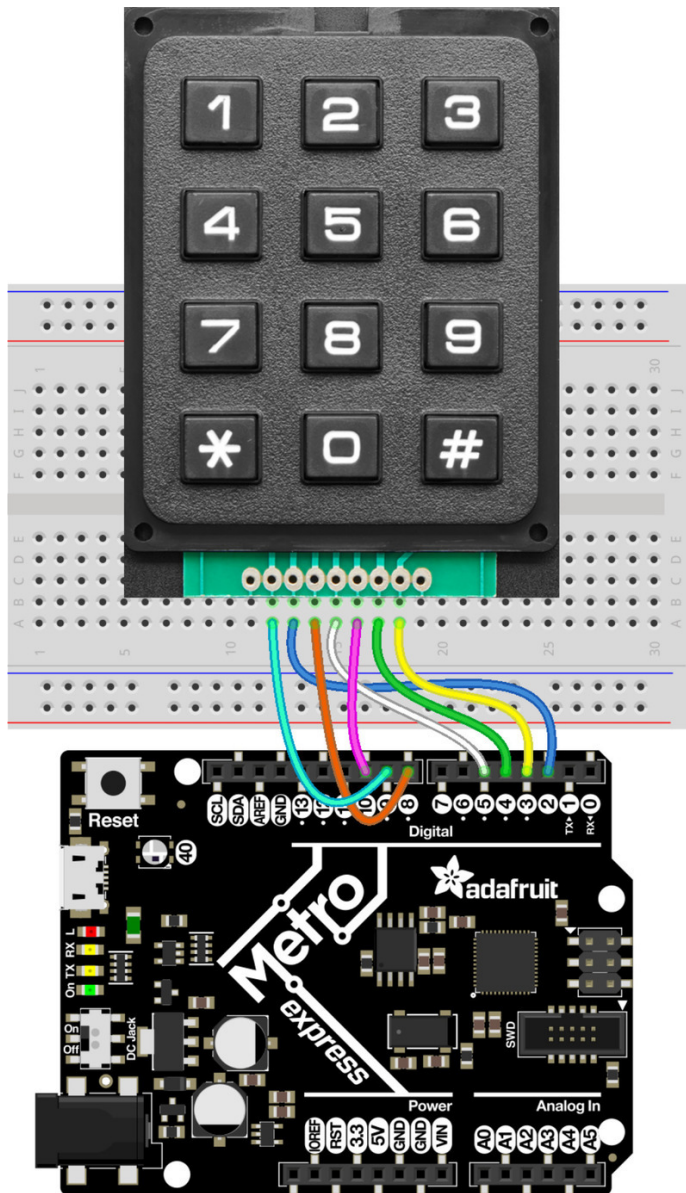
Tools -> Manage Libraries...

Open it, search for 'keypad', and click Install on the one labeled Adafruit Keypad.



Example

For this example we will use the 3x4 Matrix Keypad ([PID3845 \(http://adafru.it/3845\)](http://adafru.it/3845)). Here is the wiring.



In the Arduino IDE open the example:

File -> Examples -> Adafruit Keypad -> keypad_test

Find these lines of code up near the top:

```
// define your specific keypad here via PID
#define KEYPAD_PID3844
```

and change it to this to match the keypad being used:

```
// define your specific keypad here via PID
#define KEYPAD_PID3845
```

Upload the sketch and then open the Serial Monitor

Tools -> Serial Monitor

Try pressing the buttons. You should see text like this:



Note that each press produces two messages. One for the down (**pressed**) and one for the up (**released**).

Python & CircuitPython

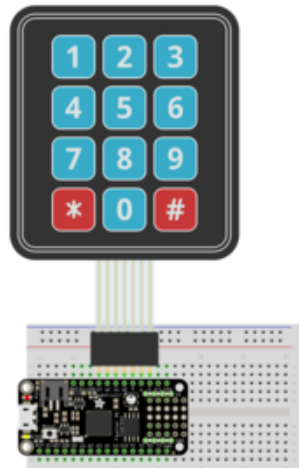
It's easy to use a matrix keypad with Python or CircuitPython and the [Adafruit CircuitPython Matrix Keypad \(https://adafru.it/CUo\)](https://adafru.it/CUo) module. This module allows you to easily write Python code that reads the button presses from the keypad.

You can use this with any CircuitPython microcontroller board or with a computer that has GPIO and Python [thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library \(https://adafru.it/BSN\)](https://adafru.it/BSN).

On most native CircuitPython boards, you can use the builtin **keypad** module instead of the `Adafruit_Matrix` library. See the **keypad** [Learn Guide \(https://adafru.it/TJF\)](https://adafru.it/TJF) and [this page \(https://adafru.it/19Ep\)](https://adafru.it/19Ep) for more details.

CircuitPython Microcontroller Wiring

First wire up a matrix keypad to your board. Here's an example of wiring a 3x4 matrix to a Feather M0. You can use any free digital I/O pins. The keypad pins are referenced left to right if the keypad is oriented upright and facing you, i.e. pin 1 is the first pin on the side of the header closest to the 1-column on the keypad.

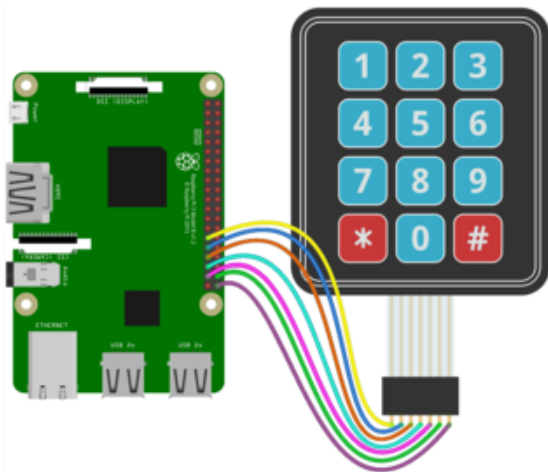


Board D13 to keypad pin 1
 Board D12 to keypad pin 2
 Board D11 to keypad pin 3
 Board D10 to keypad pin 4
 Board D9 to keypad pin 5
 Board D6 to keypad pin 6
 Board D5 to keypad pin 7

Python Computer Wiring

Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported](https://adafru.it/BSN) (<https://adafru.it/BSN>).

Here's an example of a 3x4 matrix keypad wired to the Raspberry Pi. You can use any free digital I/O pins. The keypad pins are referenced left to right if the keypad is oriented upright and facing you, i.e. pin 1 is the first pin on the side of the header closest to the 1-column on the keypad.



Pi GPIO5 to keypad pin 1
 Pi GPIO6 to keypad pin 2
 Pi GPIO13 to keypad pin 3
 Pi GPIO19 to keypad pin 4
 Pi GPIO26 to keypad pin 5
 Pi GPIO20 to keypad pin 6
 Pi GPIO21 to keypad pin 7

CircuitPython Installation of Matrix Keypad Library

You'll need to install the [Adafruit CircuitPython Matrix Keypad](https://adafru.it/CUo) (<https://adafru.it/CUo>) library on your CircuitPython board.

First make sure you are running the [latest version of Adafruit CircuitPython](https://adafru.it/Amd) (<https://adafru.it/Amd>) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from [Adafruit's CircuitPython library bundle](https://adafru.it/uap) (<https://adafru.it/uap>). Our CircuitPython starter guide has [a great page on how to install the library bundle](https://adafru.it/ABU) (<https://adafru.it/ABU>).

For non-express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- `adafruit_matrixkeypad.mpy`

Before continuing make sure your board's lib folder or root filesystem has the `adafruit_matrixkeypad.mpy` file copied over.

Next [connect to the board's serial REPL](https://adafru.it/Awz) (<https://adafru.it/Awz>) so you are at the CircuitPython >>> prompt.

Python Installation of Matrix Keypad Library

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. [Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready](https://adafru.it/BSN) (<https://adafru.it/BSN>)!

Once that's done, from your command line run the following command:

- `sudo pip3 install adafruit-circuitpython-matrixkeypad`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

CircuitPython & Python Usage

To demonstrate the usage of a matrix keypad, we'll initialise it and read the button presses from the board's Python REPL.

First run the following code to import the necessary libraries:

```
import time
import digitalio
import board
import adafruit_matrixkeypad
```

Next we're going to identify what pins are associated with the rows and columns, and set up the order of the keys:

```
cols = [digitalio.DigitalInOut(x) for x in (board.D9, board.D6, board.D5)]
rows = [digitalio.DigitalInOut(x) for x in (board.D13, board.D12, board.D11,
board.D10)]
keys = ((1, 2, 3),
        (4, 5, 6),
        (7, 8, 9),
        (*, 0, '#'))
```

Then we initialise the matrix:

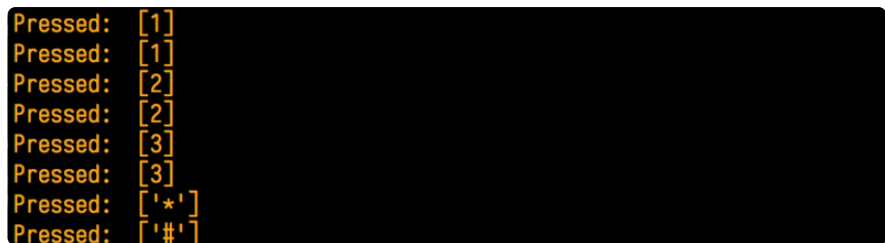
```
keypad = adafruit_matrixkeypad.Matrix_Keypad(rows, cols, keys)
```

Now you're ready to utilise the keypad with the following property:

- **pressed_keys**: An array containing all detected keys that are pressed from the initialised list-of-lists passed in during creation.

For example, to print all the key presses:

```
while True:
    keys = keypad.pressed_keys
    if keys:
        print("Pressed: ", keys)
    time.sleep(0.1)
```

A terminal window with a black background and yellow text. It shows a series of 'Pressed:' messages followed by lists of key indices or characters. The sequence is: [1], [1], [2], [2], [3], [3], ['*'], and ['#'].

```
Pressed: [1]
Pressed: [1]
Pressed: [2]
Pressed: [2]
Pressed: [3]
Pressed: [3]
Pressed: ['*']
Pressed: ['#']
```

That's all there is to using a matrix keypad with CircuitPython!

Full Example Code

For CircuitPython microcontroller boards:

```

# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import digitalio
import board
import adafruit_matrixkeypad

# Membrane 3x4 matrix keypad - https://www.adafruit.com/product/419
cols = [digitalio.DigitalInOut(x) for x in (board.D9, board.D6, board.D5)]
rows = [digitalio.DigitalInOut(x) for x in (board.D13, board.D12, board.D11,
board.D10)]

# 3x4 matrix keypad - Rows and columns are mixed up for https://www.adafruit.com/
product/3845
# Use the same wiring as in the guide with the following setup lines:
# cols = [digitalio.DigitalInOut(x) for x in (board.D11, board.D13, board.D9)]
# rows = [digitalio.DigitalInOut(x) for x in (board.D12, board.D5, board.D6,
board.D10)]

keys = ((1, 2, 3), (4, 5, 6), (7, 8, 9), ("*", 0, "#"))

keypad = adafruit_matrixkeypad.Matrix_Keypad(rows, cols, keys)

while True:
    keys = keypad.pressed_keys
    if keys:
        print("Pressed: ", keys)
        time.sleep(0.1)

```

For Raspberry Pi:

```

# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import digitalio
import board
import adafruit_matrixkeypad

# Membrane 3x4 matrix keypad on Raspberry Pi -
# https://www.adafruit.com/product/419
cols = [digitalio.DigitalInOut(x) for x in (board.D26, board.D20, board.D21)]
rows = [digitalio.DigitalInOut(x) for x in (board.D5, board.D6, board.D13,
board.D19)]

# 3x4 matrix keypad on Raspberry Pi -
# rows and columns are mixed up for https://www.adafruit.com/product/3845
# cols = [digitalio.DigitalInOut(x) for x in (board.D13, board.D5, board.D26)]
# rows = [digitalio.DigitalInOut(x) for x in (board.D6, board.D21, board.D20,
board.D19)]

keys = ((1, 2, 3), (4, 5, 6), (7, 8, 9), ("*", 0, "#"))

keypad = adafruit_matrixkeypad.Matrix_Keypad(rows, cols, keys)

while True:
    keys = keypad.pressed_keys
    if keys:
        print("Pressed: ", keys)
        time.sleep(0.1)

```

Python Docs

[Python Docs \(https://adafru.it/C97\)](https://adafru.it/C97)