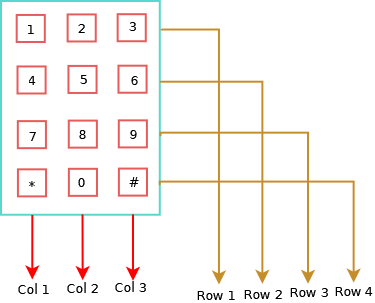
Schematics



If each col and row are connected to GPIO pins then you can detect which key is pressed by going narrowing down the row and column. for example if Col 2 and Row 2 are giving signal then Key ‘5’ is pressed. With the given information lets connect the pins to the raspberry Pi. The Pi that I used is version 1 and it has less number of GPIOs but the pinout of raspberry is partly  backward compatible so the  pin numbering and the code works fine in all the Pi versions with minor changes .

Now you  need to connect the following way.

Pin Mapping

**Key pad = Rpi pin**

**Col 1 = 17**

**Col 2 = 27**

**Col 3 = 22**

**Row 1 = 11**

**Row 2 = 23**

**Row 3 = 24**

**Row 4 = 25**

The pin number of the Rpi represents the BCM number you have a look at the pin reference. There is also an interesting website which provides Rpi pinout reference [http://pinout.xyz](http://pinout.xyz/) .

https://upload.wikimedia.org/wikipedia/commons/6/61/Raspberry-pi-gpio.png

Now that we have done the connection lets come to the coding part. I am using python for the sound board project so I want to configure that get the keypad working in python.

There are many libraries which are available to interface a keypad with a RasberryPi, I chose this one [pad4pi](https://github.com/brettmclean/pad4pi)was very handy and easy to start.

**Installation of pad4pi**

pip install pad4pi

|  |  |
| --- | --- |
| Keypad demo code | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | from pad4pi import rpi\_gpio  import time    def processKey(key):      print(key)    # Setup Keypad  KEYPAD = [       ["1","2","3"],       ["4","5","6"],       ["7","8","9"],       ["\*","0","#"]  ]    ROW\_PINS = [11,23,24,25] # BCM numbering  COL\_PINS = [17,27,22] # BCM numbering    factory = rpi\_gpio.KeypadFactory()    keypad = factory.create\_keypad(keypad=KEYPAD, row\_pins=ROW\_PINS, col\_pins=COL\_PINS)    keypad.registerKeyPressHandler(processKey)  while 1:    time.sleep(1)  keypad.cleanup() |

You can also check out the sources from [here https://github.com/codelectron/codelectron\_projects/blob/master/Rpi/keypad/keypad.py](https://github.com/codelectron/codelectron_projects/blob/master/Rpi/keypad/keypad.py)

**Code Explanation:**

**ROW\_PINS = [11,23,24,25] # BCM numbering**  
**COL\_PINS = [17,27,22] # BCM numbering**

The main connection between the hardware and the library is the pins and that is defined above.

**KEYPAD = [**  
**[“1″,”2″,”3”],**  
**[“4″,”5″,”6”],**  
**[“7″,”8″,”9”],**  
**[“\*”,”0″,”#”]**  
**]**

The keypad table is just a reference to what you have in your hardware keypad.

**factory = rpi\_gpio.KeypadFactory()**  
**keypad = factory.create\_keypad(keypad=KEYPAD, row\_pins=ROW\_PINS, col\_pins=COL\_PINS)**  
**keypad.registerKeyPressHandler(processKey)**

Here you create a keypad factory where you get a empty factory which you use to create a keypad of that table with the number of rows and columns. Finally register a interrupt handler to be called whenever a key is pressed.