Keypad on an Arduino UNO

Keypads are a great way to let users interact with your project. You can use them to navigate menus, enter passwords, and control games and robots.

In this tutorial, I’ll show you how to setup a keypad on the Arduino. First I’ll explain how the Arduino detects key presses, then I’ll show you how to find the pinout of any keypad. As a simple example, I’ll show you how to print out the key presses on the serial monitor.

I’ll be using a 4X4 matrix membrane keypad in this article, but there’s also code and wiring diagrams for 3X4 matrix keypads as well. I like membrane style keypads because they’re thin and they also have adhesive backing so you can stick them to most flat surfaces. You can also get telephone style keypads that have thicker buttons if you like that style better. Even salvaged keypads from old telephones will work with the Arduino.

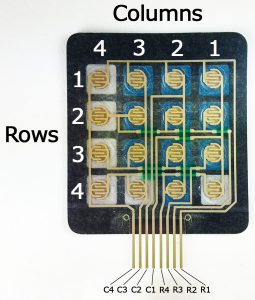
How keypads work

The buttons on a keypad are arranged in rows and columns. A 3X4 keypad has 4 rows and 3 columns, and a 4X4 keypad has 4 rows and 4 columns:



Arduino Keypad Tutorial - 3X4 and 4X4 Keypads

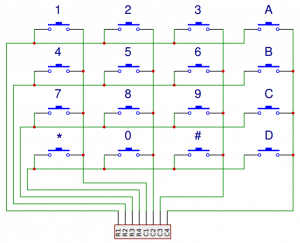
Beneath each key is a membrane switch. Each switch in a row is connected to the other switches in the row by a conductive trace underneath the pad. Each switch in a column is connected the same way – one side of the switch is connected to all of the other switches in that column by a conductive trace. Each row and column is brought out to a single pin, for a total of 8 pins on a 4X4 keypad:



How to Set Up a Keypad on an Arduino - Back Side of Keypad

Pressing a button closes the switch between a column and a row trace, allowing current to flow between a column pin and a row pin.

The schematic for a 4X4 keypad shows how the rows and columns are connected:

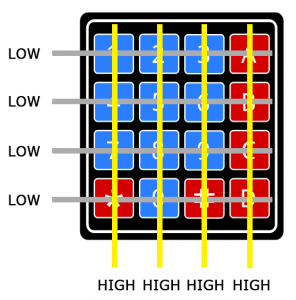


Arduino Keypad Tutorial - 4X4 Keypad Schematic

The Arduino detects which button is pressed by detecting the row and column pin that’s connected to the button.

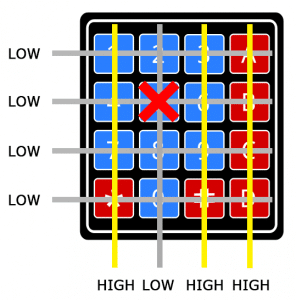
This happens in four steps:

1. First, when no buttons are pressed, all of the column pins are held HIGH, and all of the row pins are held LOW:



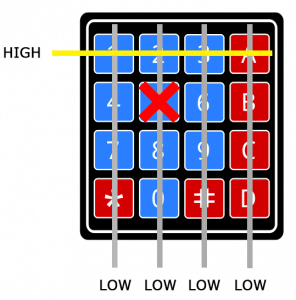
Arduino Keypad Tutorial - How the Keypad Works STEP 1

2. When a button is pressed, the column pin is pulled LOW since the current from the HIGH column flows to the LOW row pin:



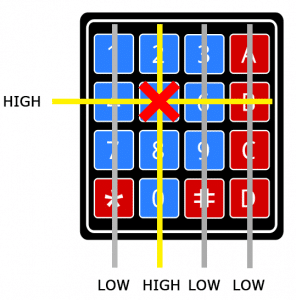
Arduino Keypad Tutorial - How the Keypad Works STEP 2

3. The Arduino now knows which column the button is in, so now it just needs to find the row the button is in. It does this by switching each one of the row pins HIGH, and at the same time reading all of the column pins to detect which column pin returns to HIGH:



Arduino Keypad Tutorial - How the Keypad Works STEP 3

4. When the column pin goes HIGH again, the Arduino has found the row pin that is connected to the button:

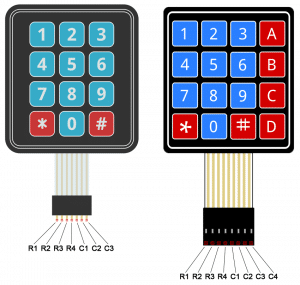


Arduino Keypad Tutorial - How the Keypad Works STEP 4

From the diagram above, you can see that the combination of row 2 and column 2 could only mean that the number 5 button was pressed.

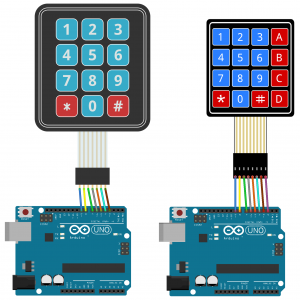
Circuit

The pin layout for most membrane keypads will look like this:



Arduino Keypad Tutorial - 4X4 and 3X4 Keypad Pin Diagram

Follow the diagrams below to connect the keypad to an Arduino Uno, depending on whether you have a 3X4 or 4X4 keypad:



Arduino Keypad Tutorial - 4X4 and 3X4 Keypad Connection Diagram

Programming the keypad

For a basic demonstration of how to setup the keypad, I’ll show you how to print each key press to the serial monitor.

Install the library

We’ll use the Keypad library by Mark Stanley and Alexander Brevig (<http://playground.arduino.cc/Code/Keypad>) . This library takes care of setting up the pins and polling the different columns and rows. To install the Keypad library, go to Sketch > Include Library > Manage Libraries and search for “keypad”. Click on the library, then click install.

Code

Once the Keypad library is installed, you can upload this code to the Arduino if you’re using a 4X4 keypad:

#include <Keypad.h>

const byte ROWS = 4;

const byte COLS = 4;

char hexaKeys[ROWS][COLS] = {

{'1', '2', '3', 'A'},

{'4', '5', '6', 'B'},

{'7', '8', '9', 'C'},

{'\*', '0', '#', 'D'}

};

byte rowPins[ROWS] = {9, 8, 7, 6};

byte colPins[COLS] = {5, 4, 3, 2};

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup(){

Serial.begin(9600);

}

void loop(){

char customKey = customKeypad.getKey();

if (customKey){

Serial.println(customKey);

}

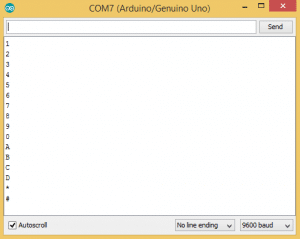
}

Lines 3 and 4 in the code above set the number of rows and columns on the keypad.

Lines 6-11 define which characters are printed when a particular button is pressed on the keypad. The characters are laid out just as they appear on the keypad. If your keypad has a different layout, you can define which characters are printed when you press a button. For example, say your keypad has a column of letters on the left instead of the right. You would just change it to this:

char hexaKeys[ROWS][COLS] = {  
{‘A’, ‘1’, ‘2’, ‘3’},  
{‘B’, ‘4’, ‘5’, ‘6’},  
{‘C’, ‘7’, ‘8’, ‘9’},  
{‘D’, ‘\*’, ‘0’, ‘#’}  
};

After you upload the code, open the serial monitor. When you press a key, the value will be printed out:

[](http://www.circuitbasics.com/wp-content/uploads/2017/04/How-to-Set-Up-a-Keypad-on-an-Arduino-Serial-Monitor-Output-of-Key-Presses.png)