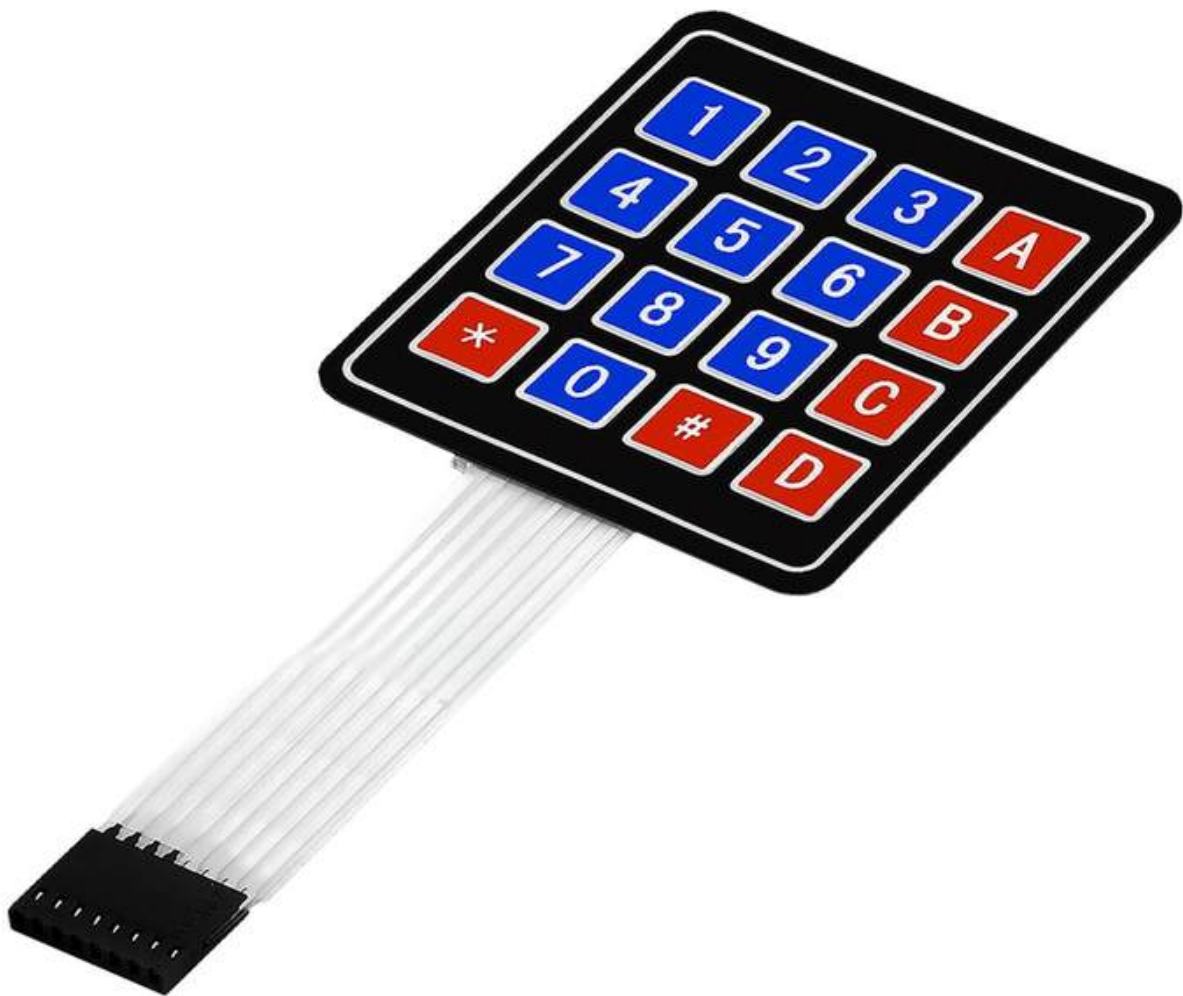


Az-Delivery

Welcome!

Thank you for purchasing our *AZ-Delivery 4x4 Matrix Keypad Module*. On the following pages, you will be introduced to how to use and set-up this handy device.

Have fun!



Areas of application

The products are intended for the support and assembly of electronic components and circuits.

Required knowledge and skills

The use of these products requires basic knowledge of electrical engineering and the handling of electronic components. Users should be able to install the products correctly and take the necessary safety precautions.

Environmental conditions

The products should be used in an environment free from moisture, dust and direct sunlight. They should not be operated near heat sources or in chemically aggressive environments to avoid damage and safety risks.

Intended Use

Passive electrical products such as heat sinks, battery holders and clips or breakout boards should be operated in environments that meet the specified temperature and voltage ranges of the respective products. These components are typically designed for indoor use.

Improper foreseeable use

Improper but foreseeable uses include use in humid or extremely hot environments or operation by untrained or disabled persons. The product must be kept away from children and pets.

disposal

Do not discard with household waste! Your product is according to the European one Directive on waste electrical and electronic equipment to be disposed of in an environmentally friendly manner. The valuable raw materials contained therein can be recycled become. The application of this directive contributes to environmental and health protection. Use the collection point set up by your municipality to return and Recycling of old electrical and electronic devices. WEEE Reg. No.: DE 62624346

safety instructions

Attention: Improper disposal of electronic components can endanger the environment and health. Note: Dispose of electronic components in accordance with local regulations and use appropriate recycling options. Attention: Chemically aggressive media can damage the materials of the products. Note: Do not use the products in corrosive or chemically aggressive environments. Attention: Improper disposal of electronic components can endanger the environment and health. Note: Dispose of electronic components in accordance with local regulations and use appropriate recycling options. Attention: Chemically aggressive media can damage the materials of the products. Note: Do not use the products in corrosive or chemically aggressive environments. Caution: Mechanical shock or bending can damage the products and connected components. Note: Avoid mechanical stress and protect the products from physical influences. Attention: Inadequate fastening can lead to malfunctions and damage. Note: Make sure all products are securely and firmly assembled. Caution: Damaged products may pose safety risks. Note: Check products regularly for visible damage and replace defective parts immediately. Attention: Overloading can lead to overheating and failure of the products. Note: Use the products only within the specified load limits. Attention: Overheating can cause damage to the products and the connected electronic components. Note: Make sure that, for example, heat sinks or components that heat up are adequately ventilated and that the specified temperature ranges are not exceeded.



Table of Contents

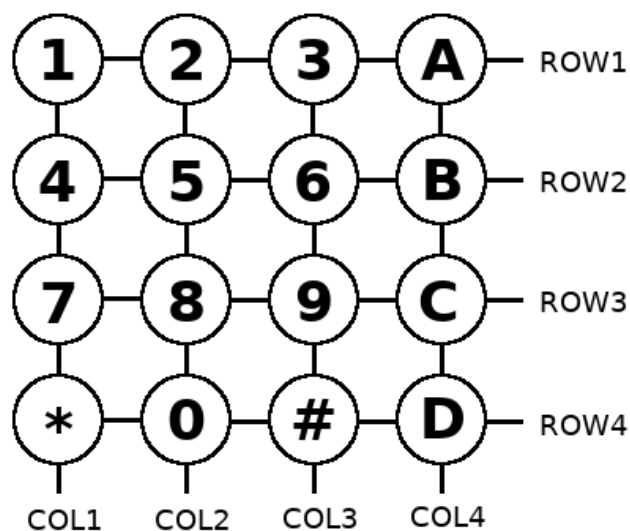
Introduction.....	3
Specifications.....	4
The pinout.....	5
How to set-up Arduino IDE.....	6
How to set-up the Raspberry Pi and the Python.....	10
Connecting the keypad with microcontroller board.....	11
Arduino IDE Library.....	13
Sketch example.....	14
Connecting the keypad with Raspberry Pi.....	18
Python script.....	19

Introduction

Keypads are a great way to make users interact with electronic devices. It can be use for navigating menus, passwords, playing games, controlling robots, etc.

The keys on the 4x4 matrix keypad module are simple momentary switches. Keys are made of thin conductive material that is stacked with an isolation foil in between. Keypad membranes are curved in the shape of the dome, so that keys do not make permanent contact when pressed.

The keys on the keypad are arranged in rows and columns. The 4x4 keypad has 4 rows and 4 columns as shown on the following image:





Specifications

- » Maximum operating voltage: 24V DC
- » Maximum operating current: 30mA
- » Operating temperature: from 0 to +50°C
- » Interface: 8pin access to 4x4 matrix
- » Dimensions: 76x9x1mm [3x2.7x0.5in]

The keypad is supplied with an adhesive backing which provides easy integration. When the white sticker on the back of the keypad is peeled off, it can be securely affixed to the desired surface.

The pinout

The 4x4 matrix keypad module has eight pins. The pinout is shown on the following image:



How to set-up Arduino IDE

If the Arduino IDE is not installed, follow the [link](#) and download the installation file for the operating system of choice.

Download the Arduino IDE



The screenshot shows the Arduino IDE download page. On the left, there is a teal circle with a white infinity symbol. To its right, the text reads: **ARDUINO 1.8.9**. Below this, it says: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions." On the right side, there is a teal sidebar with links for different operating systems: **Windows** (Installer, for Windows XP and up; ZIP file for non admin install), **Windows app** (Requires Win 8.1 or 10, with a 'Get' button), **Mac OS X** (10.8 Mountain Lion or newer), **Linux** (32 bits, 64 bits, ARM 32 bits, ARM 64 bits), **Release Notes**, **Source Code**, and **Checksums (sha512)**.

For *windows* users, double click on the downloaded .exe file and follow the instructions in the installation window.

Az-Delivery

For *Linux* users, download a file with the extension `.tar.xz`, which has to be extracted. When it is extracted, go to the extracted directory and open the terminal in that directory. Two `.sh` scripts have to be executed, the first called `arduino-linux-setup.sh` and the second called `install.sh`.

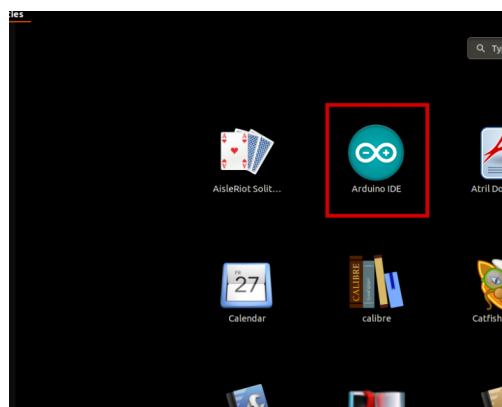
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

```
sh arduino-linux-setup.sh user_name
```

user_name - is the name of a superuser in the Linux operating system. A password for the superuser has to be entered when the command is started. Wait for a few minutes for the script to complete everything.

The second script called `install.sh` script has to be used after installation of the first script. Run the following command in the terminal (extracted directory): **sh install.sh**

After the installation of these scripts, go to the *All Apps*, where the *Arduino IDE* is installed.



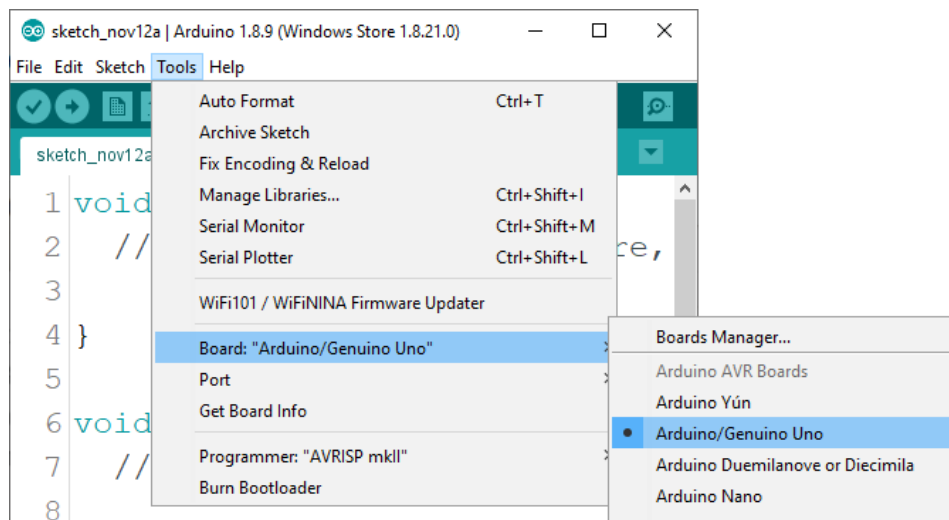
Az-Delivery

Almost all operating systems come with a text editor preinstalled (for example, *Windows* comes with *Notepad*, *Linux Ubuntu* comes with *Gedit*, *Linux Raspbian* comes with *Leafpad*, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

Next thing is to check if your PC can detect an microcontroller board. Open freshly installed Arduino IDE, and go to:

Tools > Board > {your board name here}

{your board name here} should be the *Arduino/Genuino Uno*, as it can be seen on the following image:

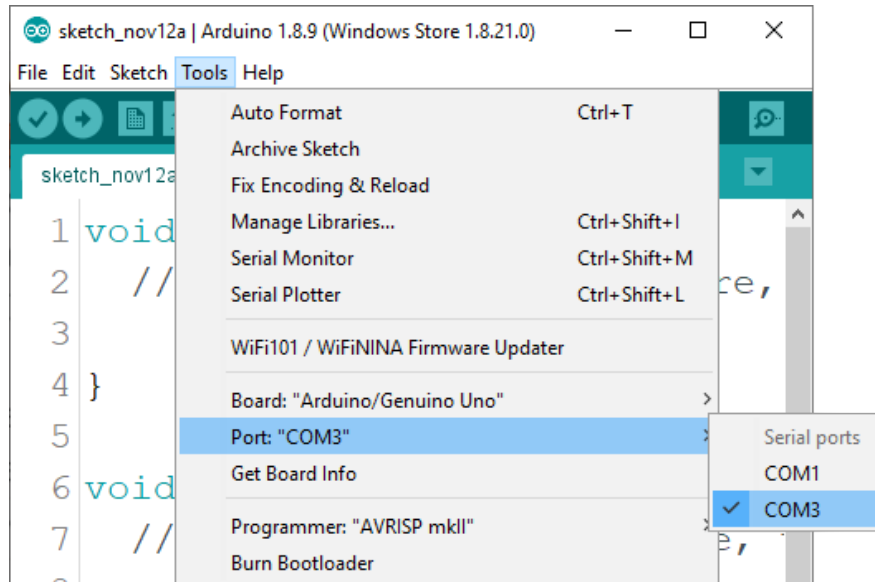


The port to which the Atmega328P board is connected has to be selected.

Go to: *Tools > Port > {port name goes here}*

and when the microcontroller board is connected to the USB port, the port name can be seen in the drop-down menu on the previous image.

If the Arduino IDE is used on Windows, port names are as follows:



For *Linux* users, for example port name is `/dev/ttyUSBx`, where *x* represents integer number between 0 and 9.



How to set-up the Raspberry Pi and Python

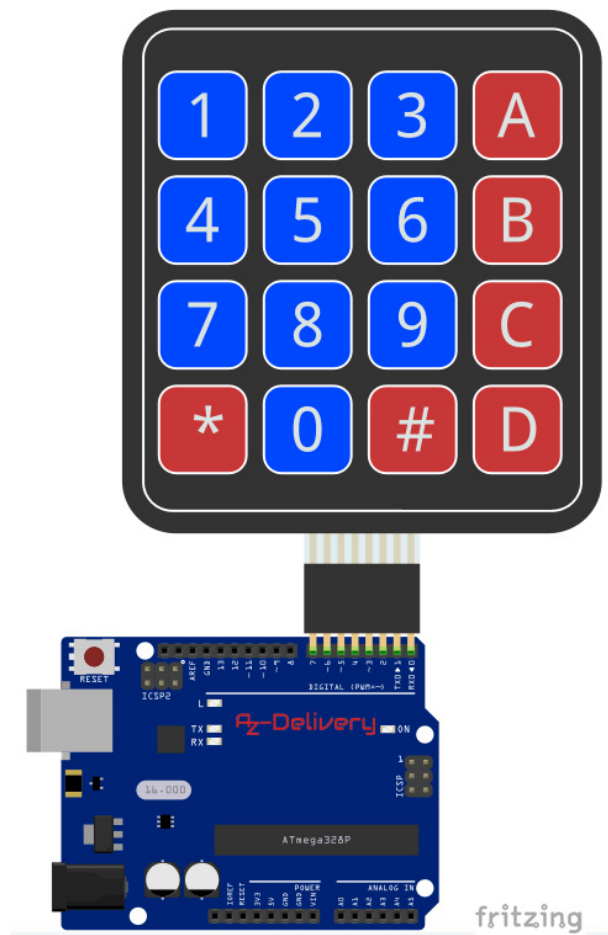
For the Raspberry Pi, first the operating system has to be installed, then everything has to be set-up so that it can be used in the *Headless* mode. The *Headless* mode enables remote connection to the Raspberry Pi, without the need for a *PC* screen Monitor, mouse or keyboard. The only things that are used in this mode are the Raspberry Pi itself, power supply and internet connection. All of this is explained minutely in the free eBook:

[Raspberry Pi Quick Startup Guide](#)

The *Raspbian* operating system comes with *Python* preinstalled.

Connecting the keypad with Atmega328P Board

Connect the 4x4 matrix keypad module with the microcontroller board as shown on the following connection diagram:



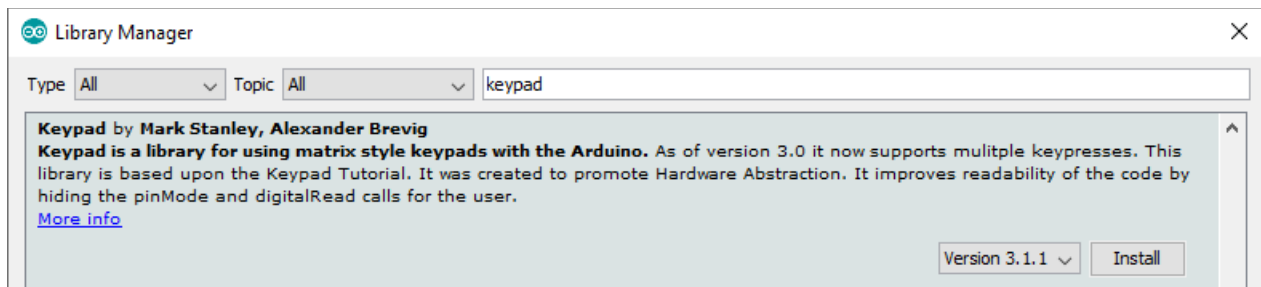
Columns		Rows	
Keypad pin	Board pin	Keypad pin	Board pin
C1	D3	R1	D7
C2	D2	R2	D6
C3	D1	R3	D5
C4	D0	R4	D4

Library for Arduino IDE

To use the keypad with microcontroller board it is recommended to download an external library for it. To download and install it, open Arduino IDE and go to:

Tools > Manage Libraries

When a new window opens, type *Keypad* in the search box and install the library called *Keypad* made by Mark Stanley and Alexander Brevig, as shown on the following image:



Click *Install* button to finish the installation of the library.

Az-Delivery

Sketch example

```
#include <Keypad.h>
#define COLS 4
#define ROWS 4

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

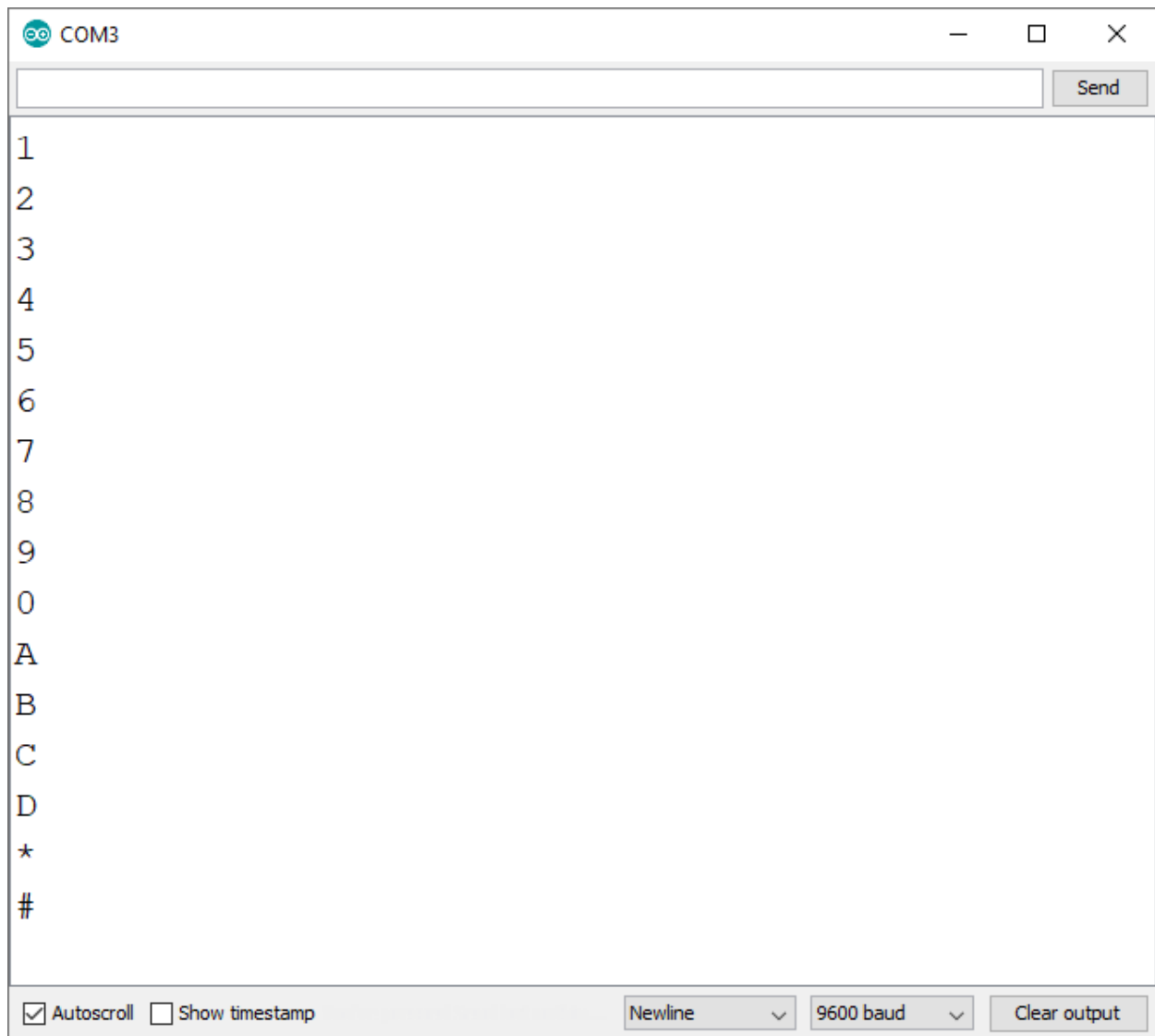
byte colPIN[COLS] = { 3, 2, 1, 0 };
byte rowPIN[ROWS] = { 7, 6, 5, 4 };
char pressedTaster;
Keypad myKeypad = Keypad(makeKeymap(KEYS), rowPIN, colPIN, ROWS, COLS);

void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.end();
  pressedTaster = myKeypad.getKey();
  if (pressedTaster) {
    Serial.begin(9600);
    Serial.print(pressedTaster);
    Serial.println();
  }
}
```

Az-Delivery

Upload the sketch to the microcontroller board and open Serial Monitor (*Tools > Serial Monitor*). The result should look like the output on the following image:



Az-Delivery

The sketch starts with including the library *Keypad*.

Two macros are created which define how many rows and columns are on the keypad. In this case, there are four rows and four columns.

Next, two-dimensional *char* array called *keys* is created, which represent characters on the keypad mapped in rows and columns.

After that, two *byte* arrays called *rowPin* and *colPin* are created and they define which pins are used on microcontroller board for row and column pin, respectively.

Next, an object *keypad* is created where previously defined macros and arrays are used to initialize it.

In the *setup()* function the serial communication is started with the baud rate of *9600bps*.

Then, in the *loop()* function, the state of the keypad is read with the following line of code: `char key = keypad.getKey();`

Az-Delivery

Next, the state of the *key* variable is checked and corresponding message is displayed in the Serial Monitor. If no key is pressed, the value saved in the *key* variable is *None*, which means that when the value is checked with *if* block, the *if* block results is *false* and the block is skipped. If one key is pressed, the value of that key is saved in the *key* variable. The *if* block results with *true* after which the value in the *key* variable is checked again. This time the value is checked for specific name, the name of specific key of the keypad. In the sketch the value that is checked is the character '4':

```
if(key == '4')
```

If the key '4' is pressed the message:

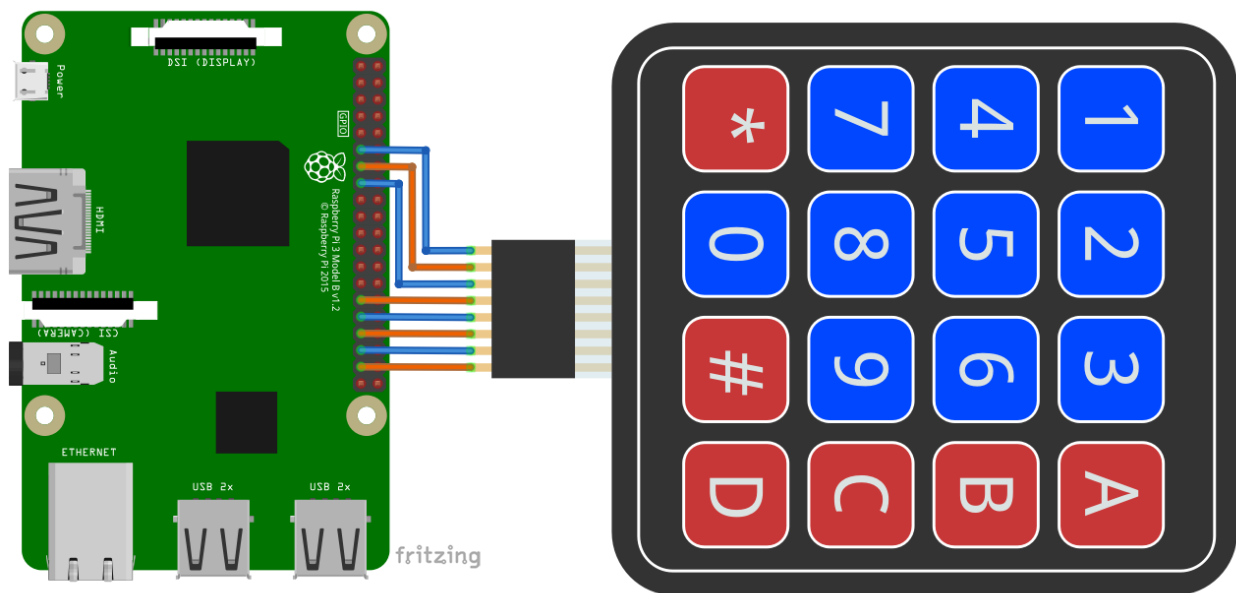
Key 4 is pressed -> Something

is displayed in the Serial Monitor.

If other pins are pressed only pin names are displayed in the Serial Monitor.

Connecting the keypad with Raspberry Pi

Connect the 4x4 matrix keypad module with the Raspberry Pi as shown on the following connection diagram:



Columns			Rows		
Keypad pin	RaspPi pin	Physical pin	Keypad pin	RaspPi pin	Physical pin
C1	GPIO6	31	R1	GPIO17	11
C2	GPIO13	33	R2	GPIO27	13
C3	GPIO19	35	R3	GPIO22	15
C4	GPIO26	37	R4	GPIO5	19



Python script

Two scripts are made. The first is the class script and the second is the main executable script. The following script code is modified from the [script](#).

The code is changed to make more beginner friendly script.

```
import RPi.GPIO as GPIO
class keypad():
    def __init__(self, C1, C2, C3, C4, R1, R2, R3, R4):
        GPIO.setmode(GPIO.BCM)
        GPIO.setwarnings(False)
        self.KEYPAD = [
            [1, 2, 3, 'A'],
            [4, 5, 6, 'B'],
            [7, 8, 9, 'C'],
            ['*', 0, '#', 'D']]
        self.ROW = [R1, R2, R3, R4]
        self.COL = [C1, C2, C3, C4]

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

Az-Delivery

```
# two tabs

# 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.COL[j], GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

# 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.COL)):
    tmpRead = GPIO.input(self.COL[j])
    if tmpRead == 1:
        colVal = j

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

Az-Delivery

```
# two tabs

if len(self.COL) == 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.COL[j], GPIO.IN, pull_up_down=GPIO.PUD_UP)

    GPIO.cleanup()
```

Save the script under name *keypad_class.py*.

Az-Delivery

The following code is the code of main script:

```
import time
import keypad_class as kpad

kp = kpad.keypad(6, 13, 19, 26, 17, 27, 22, 5)
digit = None
last_digit = None

print('[Press CTRL + C to end the script!]\n')
try: # Main program loop
    while True:
        digit = kp.getKey()
        if not last_digit == digit:
            if not digit == None:
                print('{}\n'.format(digit))

            last_digit = digit
            time.sleep(0.1)

except KeyboardInterrupt:
    print('\nScript end!\n')

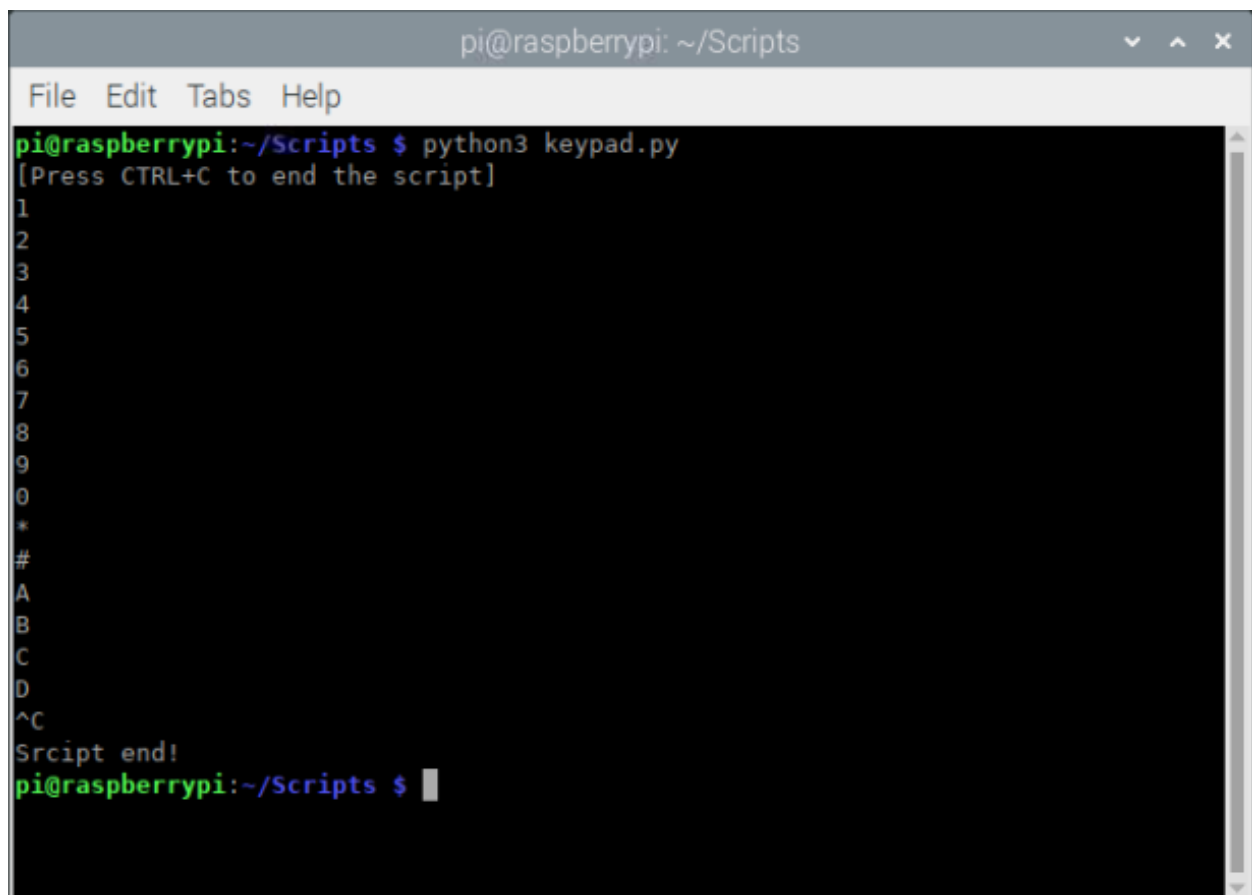
finally:
    kp.exit()
```

Az-Delivery

Save the script by the name *keypad.py* in the same directory where *keypad_class.py* script is saved. To run the main script, open the terminal in the directory where the scripts are saved and run the following command:

python3 keypad.py

The result should look like the output on the following image:

A screenshot of a terminal window titled 'pi@raspberrypi: ~/Scripts'. The window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows the command 'python3 keypad.py' being executed. The output displays a keypad layout with numbers 1-9, 0, *, and #, followed by letters A-D and a '^C' prompt. After pressing Ctrl+C, the message 'Script end!' is shown, and the prompt returns to 'pi@raspberrypi:~/Scripts \$'.

To end the script press *CTRL* + *C* on the PC keyboard.

The class script is not covered in this book, only the main script is explained.

Az-Delivery

The script *keypad.py* starts with importing one library, *time* and with importing constructor from the *keypad_class* script.

Next, the object called *kp* is created with the following line of code:

```
kp = keypad(4, 17, 27, 22, 24, 25, 12, 16)
```

where these numbers represent the pins of the Raspberry Pi to which keypad pins are connected (*C1*, *C2*, *C3*, *C4*, *R1*, *R2*, *R3*, *R4*). The *kp* object represents the keypad itself.

After this, two variables *digit* and *last_digit* are created. These two variables are used for detecting which key is pressed.

Then, the *try-except-finally* block of code is created. In the *try* block of code, the indefinite loop is created. In the indefinite loop there is the main algorithm for detecting which key is pressed.

To end the script press *CTRL + C* on the PC keyboard. This is called the keypad interrupt. When the keypad interrupt happens, the *except* block of code is executed and the message *Script end!* is displayed in the terminal.

The *finally* block of code is executed on the script end. In this block of code all used GPIO pin modes and/or interfaces are disabled.

Az-Delivery

In the indefinite block of code, first the state of the keypad is read, with the following line of code: `digit = kp.keypad()`

Then, the state is checked if the new pressed key is different from the last pressed key. If it is different, then new key state is displayed and the value of the new pressed key is saved in the last pressed key variable *last_digit*. The keypad state when the key is pressed is displayed in the terminal. The state of the keypad is displayed only when the state is changed, even if the key is pressed for a longer period of time.

```
digit = kp.getKey()
if not last_digit == digit:
    if not digit == None:
        print('{}'.format(digit))
        last_digit = digit
    time.sleep(0.1)
```

Az-Delivery

Now is the time to learn and make projects on your own. You can do that with the help of many example scripts and other tutorials, which can be found on the Internet.

If you are looking for the high quality microelectronics and accessories, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

<https://az-delivery.de>

Have Fun!

Impressum

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