

AZ-Delivery

Welcome!

Thank you for purchasing our *AZ-Delivery KY-023 Joystic 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
Specification.....	4
The pinout.....	4
How to set-up Arduino IDE.....	5
How to set-up the Raspberry Pi and Python.....	9
Connecting the module with Atmega328p.....	10
Sketch example.....	11
Connecting the module with Raspberry Pi.....	13
Python script for KY-023 module.....	24



Introduction

The joystick module is an easy-to-use analog joystick for a microcontrollers. The joystick features two-axis (X and Y axis) as well as a button switch that is activated when the joystick is pressed. The joystick is basically a combination of two potentiometers. This means that when the joystick is moving along the X -axis the resistance of the potentiometer changes and when voltage is applied, resistance change results in a change of voltage. The voltage can be used to detect the X position by connecting the VRX pin of the joystick to an analog input $A0$ pin of the Atmega328p. The same applies for the Y -axis. The Y -axis position can be read by connecting VRX to analog input $A1$ pin of the Atmega328p. Press state or the state of the SW pin can be connected to the digital pin $D2$ of the Atmega328p. Because the module consist of two potentiometers, it can be used with any power supply voltage, for example it can work on both $+3.3V$ and $+5V$ DC.

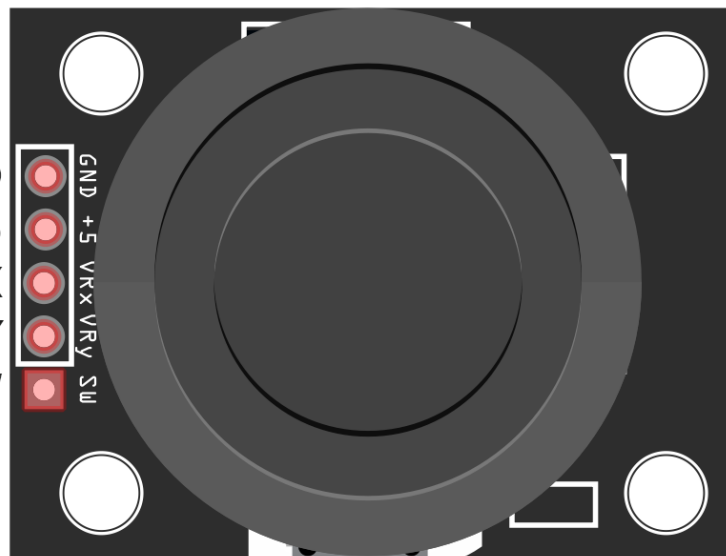
Specification

- » Operating Voltage: from 3.3V to 5V DC
- » Output: digital, with high sensitivity
- » Dimensions: 26 x 34 x 32mm [1.02 x 1.34 x 1.26in]

The pinout

The KY-023 joystic module has five pins. The pinout diagram is shown on the following image:

Ground - GND
Power supply - +5
X-axis - VRX
Y-axis - VRY
Switch - SW



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 containing a white infinity symbol with a minus sign on the left loop and a plus sign on the right loop. To the right of this icon, the text reads: **ARDUINO 1.8.9**, followed by a paragraph describing it as open-source software for writing code and uploading to boards, and a note that it can be used with any Arduino board. On the right side of the page, there is a teal sidebar with links for different operating systems: Windows (Installer and ZIP file), Windows app (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*.

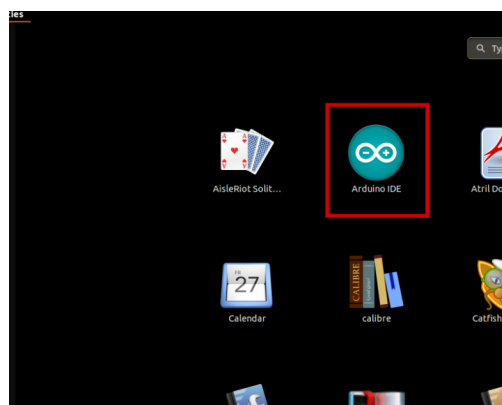
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

```
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.



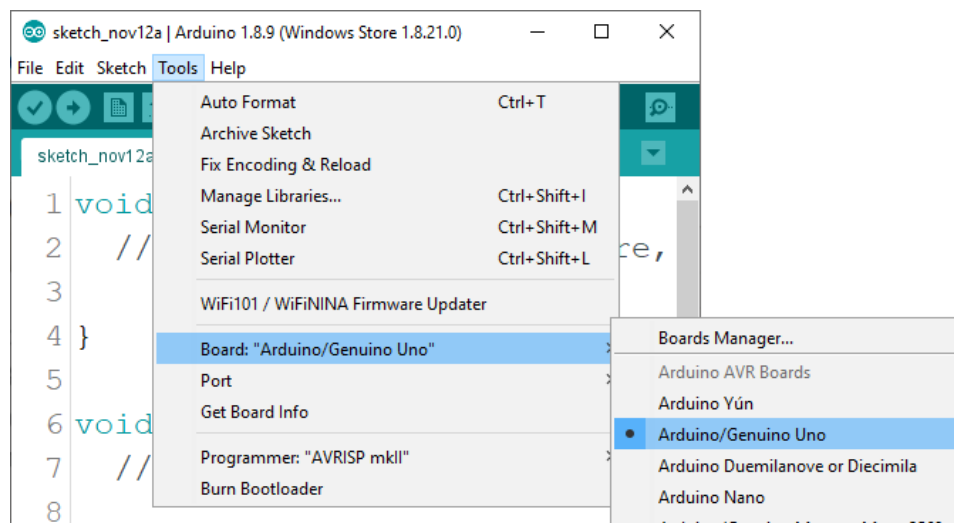
Az-Delivery

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 Atmega328p 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:

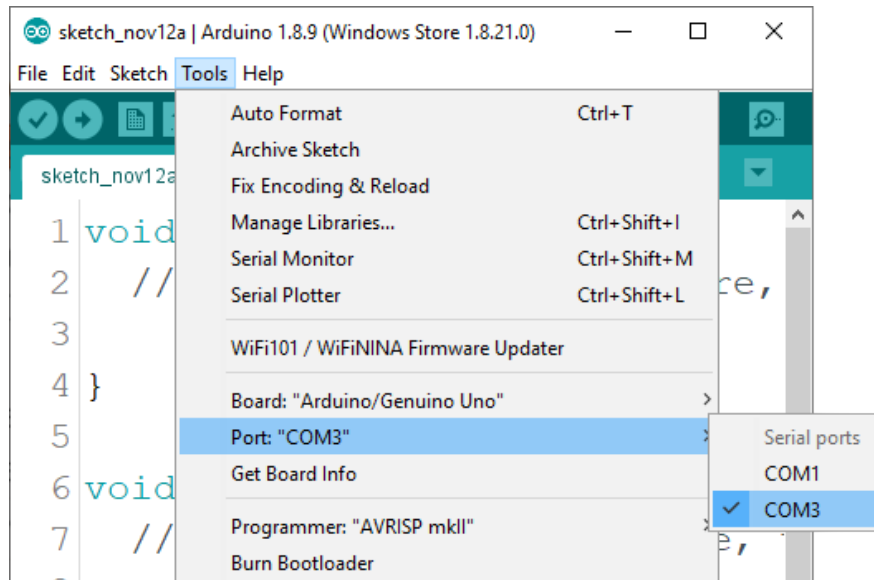


The port to which the Atmega328p board is connected has to be selected. Go to: *Tools > Port > {port name goes here}*

and when the Atmega328p board is connected to the USB port, the port name can be seen in the drop-down menu on the previous image.

Az-Delivery

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

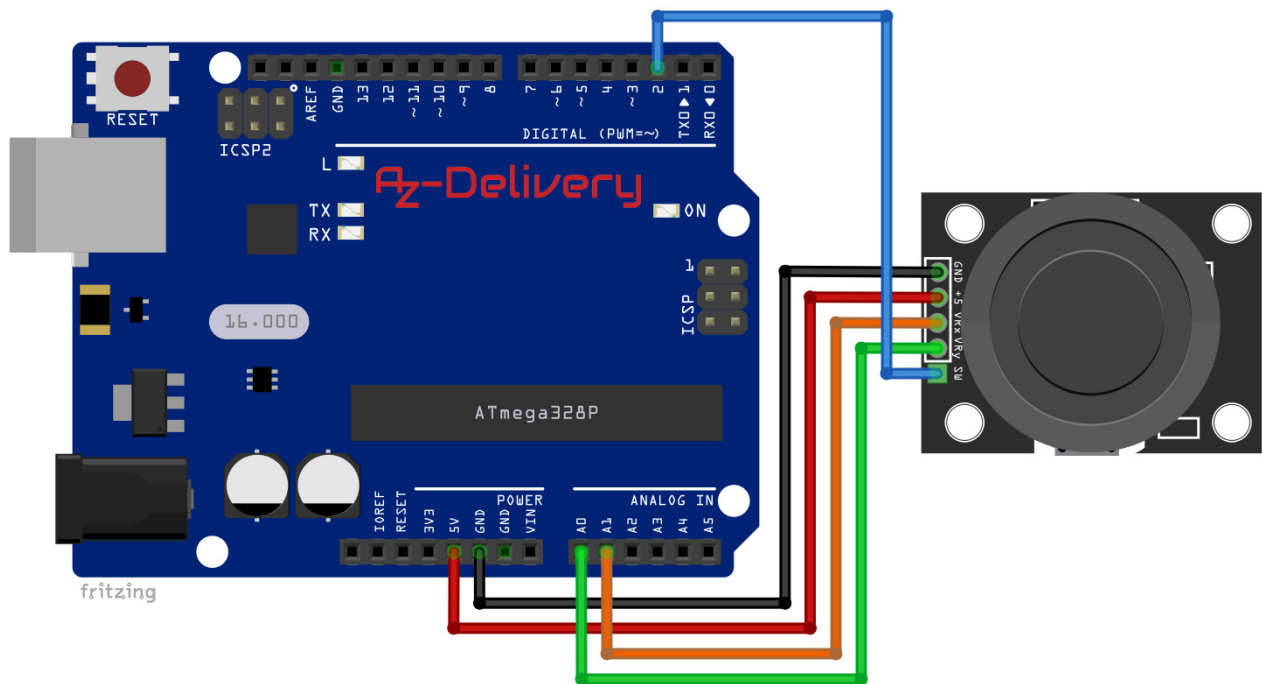
For the Raspberry Pi, first the operating system has to be installed, then everything has to be set-up so that it can be used in the *Headless* mode. The *Headless* mode enables remote connection to the Raspberry Pi, without the need for a *PC* screen Monitor, mouse or keyboard. The only things that 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 module with Atmega328p

Connect the KY-023 module with the Atmega328p as shown on the following connection diagram:



KY-023 pin	>	Mc Pin
GND	>	GND
+5V	>	5V
VRX	>	A1
VRY	>	A0
SW	>	D2

Black wire

Red wire

Orange wire

Green wire

Blue wire

Az-Delivery

Sketch example

```
#define VX_PIN 0
#define VY_PIN 1
#define BUTTON_PIN 2
uint8_t value = 0;

void setup() {
    pinMode(BUTTON_PIN, INPUT_PULLUP);
    Serial.begin(9600);
}

void loop() {
    value = analogRead(VX_PIN);
    Serial.print("X:");
    Serial.print(value, DEC);

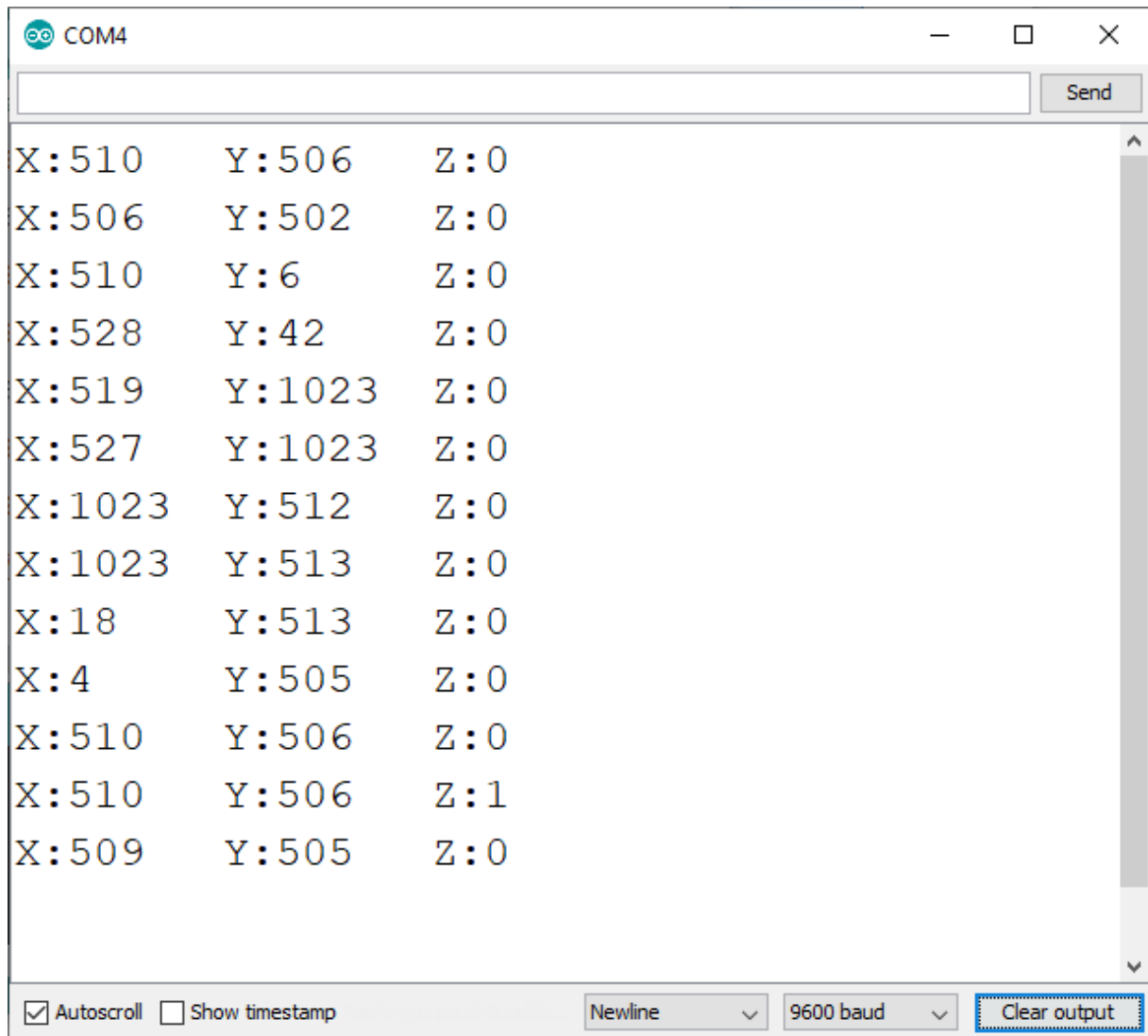
    value = analogRead(VY_PIN);
    Serial.print("\tY:");
    Serial.print(value, DEC);

    value = !digitalRead(BUTTON_PIN);
    Serial.print("\tZ:");
    Serial.println(value, DEC);

    delay(500);
}
```

Az-Delivery

Upload the sketch to the Atmega328p and open Serial Monitor (*Tools > Serial Monitor*). The result should look like the output on the following image:



To get the output from the image move or press the Joystick.



Connecting the module with Raspberry Pi

Because the Raspberry Pi does not have Analog to Digital Converter (ADC), but for purpose of using the KY-023 module with the Raspberry Pi, the Raspberry Pi has to be able to read analog voltages. The Atmega328p can be used for the purpose. In order to do so, the Atmega328p is used on the *Linux Raspbian* operating system. Atmega328p can read analog voltages, and it can use *Serial Interface* via *USB* port to send data to the Raspberry Pi.

First, the Arduino IDE has to be downloaded and installed on the Raspbian. Second, the firmware for Atmega328p has to be downloaded and uploaded to the Atmega328p, and lastly, the library for Python has to be downloaded and installed.

To do this, power on the Raspberry Pi and connect it to the internet. Start the *RealVNC* app on the remote computer and connect the app to the Raspberry Pi (like explained in the eBook for the Raspberry Pi).

First thing that has to be done when the Raspberry Pi is booted up is to update the Raspbian; open the terminal and run the following command:

```
sudo apt-get update && sudo apt-get upgrade -y
```

And wait for the command to finish its job.

Now, the Raspbian operating system is up to date.

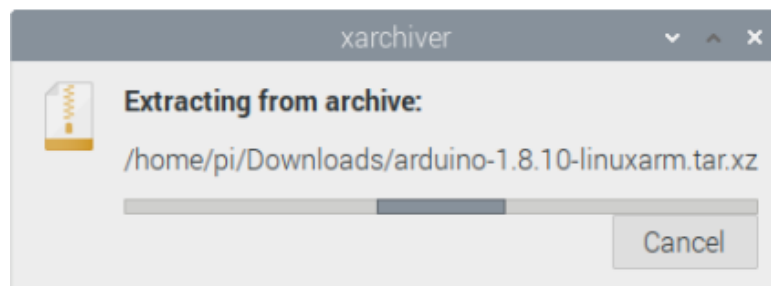
Az-Delivery

To download and install the Arduino IDE, go on the [site](#) and download the *tar.xz* file of Arduino IDE for *Linux ARM 32 bits* as shown on the following image:

Download the Arduino IDE



Then, the *tar.xz* file has to be extracted. Open the *File explorer* in directory where the *tar.xz* file is downloaded, right click on it, and run the option *Extract Here*. Wait for a few minutes for the extracting process to complete itself.

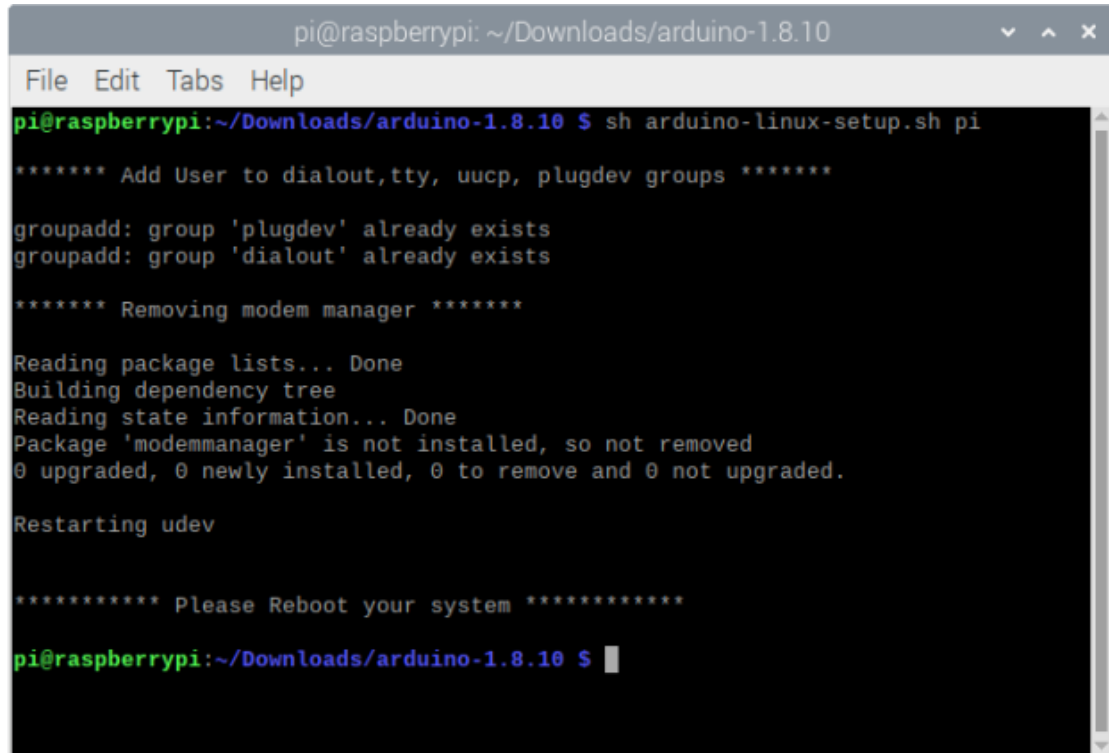


Az-Delivery

Open terminal in extracted folder and run the following command:

sh arduino-linux-setup.sh pi

where *pi* is the name of the superuser in Raspbian.



```
pi@raspberrypi: ~/Downloads/arduino-1.8.10
File Edit Tabs Help
pi@raspberrypi:~/Downloads/arduino-1.8.10 $ sh arduino-linux-setup.sh pi

***** Add User to dialout, tty, uucp, plugdev groups *****

groupadd: group 'plugdev' already exists
groupadd: group 'dialout' already exists

***** Removing modem manager *****

Reading package lists... Done
Building dependency tree
Reading state information... Done
Package 'modemmanager' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

Restarting udev

***** Please Reboot your system *****

pi@raspberrypi:~/Downloads/arduino-1.8.10 $
```

After this, to install the Arduino IDE, run the following command:

sudo sh install.sh



```
pi@raspberrypi: ~/Downloads/arduino-1.8.10
File Edit Tabs Help
pi@raspberrypi:~/Downloads/arduino-1.8.10 $ sudo sh install.sh
Adding desktop shortcut, menu item and file associations for Arduino IDE...

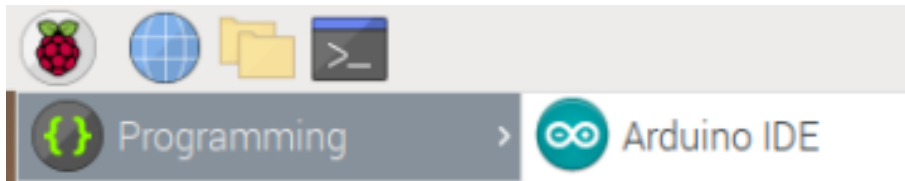
touch: cannot touch '/root/.config/mimeapps.list': No such file or directory
/usr/bin/xdg-mime: 848: /usr/bin/xdg-mime: cannot create /root/.config/mimeapps.
list.new: Directory nonexistent

done!
pi@raspberrypi:~/Downloads/arduino-1.8.10 $
```

Az-Delivery

The Arduino IDE is now installed. To run Arduino IDE, open the app:

Applications Menu > Programming > Arduino IDE



Before next steps, first the *pip3* and *git* apps have to be installed; run the following command:

```
sudo apt install python3-pip git -y
```

The library for Python is called *nanpy*. To install it, open terminal and run the following command: **pip3 install nanpy**

```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ pip3 install nanpy
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting nanpy
  Downloading https://www.piwheels.org/simple/nanpy/nanpy-0.9.6-py3-none-any.whl
(47kB)
    100% |#####| 51kB 209kB/s
Requirement already satisfied: pyserial in /usr/lib/python3/dist-packages (from
nanpy) (3.4)
Installing collected packages: nanpy
Successfully installed nanpy-0.9.6
pi@raspberrypi:~/Scripts $
```

Az-Delivery

After installing the *nanpy* library, download an Atmega328p firmware by running the following command:

```
git clone https://github.com/nanpy/nanpy-firmware.git
```

Change directory to *nanpy-firmware* by running the following command:

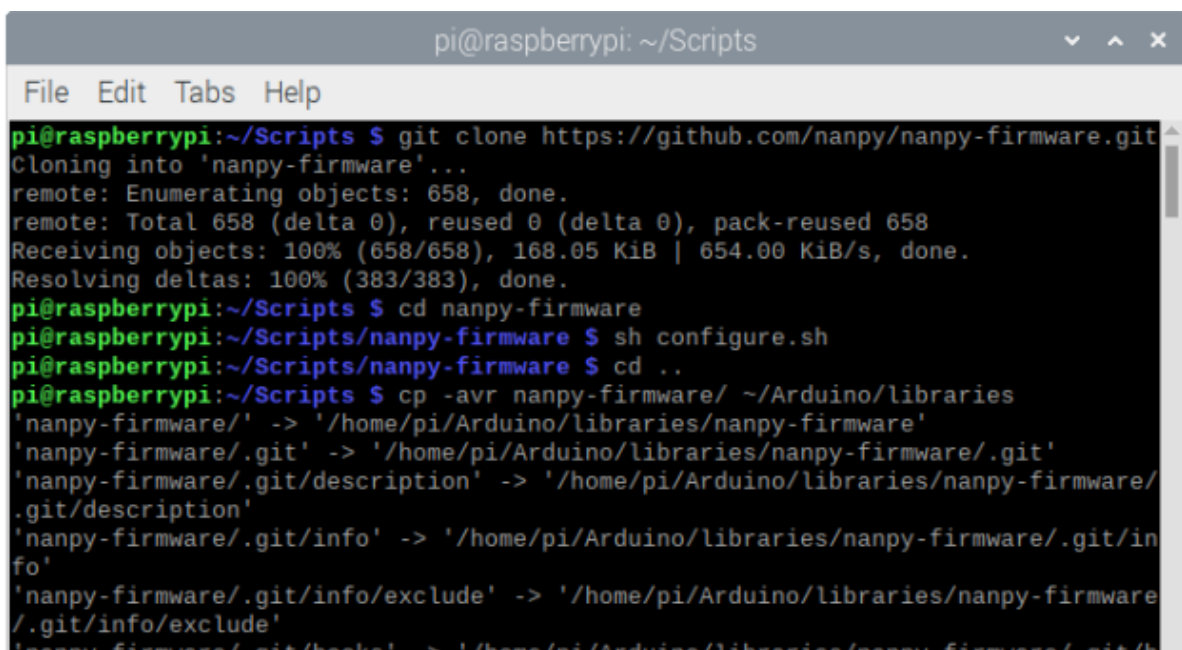
```
cd nanpy-firmware
```

And run the following command:

```
sh configure.sh
```

Next, copy the *nanpy-firmware* directory into *Arduino/libraries* directory. To do so, run the following command:

```
cp -avr nanpy-firmware/ ~/Arduino/libraries
```



```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ git clone https://github.com/nanpy/nanpy-firmware.git
Cloning into 'nanpy-firmware'...
remote: Enumerating objects: 658, done.
remote: Total 658 (delta 0), reused 0 (delta 0), pack-reused 658
Receiving objects: 100% (658/658), 168.05 KiB | 654.00 KiB/s, done.
Resolving deltas: 100% (383/383), done.
pi@raspberrypi:~/Scripts $ cd nanpy-firmware
pi@raspberrypi:~/Scripts/nanpy-firmware $ sh configure.sh
pi@raspberrypi:~/Scripts/nanpy-firmware $ cd ..
pi@raspberrypi:~/Scripts $ cp -avr nanpy-firmware/ ~/Arduino/libraries
'nanpy-firmware/' -> '/home/pi/Arduino/libraries/nanpy-firmware'
'nanpy-firmware/.git' -> '/home/pi/Arduino/libraries/nanpy-firmware/.git'
'nanpy-firmware/.git/description' -> '/home/pi/Arduino/libraries/nanpy-firmware/.git/description'
'nanpy-firmware/.git/info' -> '/home/pi/Arduino/libraries/nanpy-firmware/.git/info'
'nanpy-firmware/.git/info/exclude' -> '/home/pi/Arduino/libraries/nanpy-firmware/.git/info/exclude'
'nanpy-firmware/.git/hooks' -> '/home/pi/Arduino/libraries/nanpy-firmware/.git/hooks'
```

The *nanpy-firmware* is now installed and ready to be used.

Az-Delivery

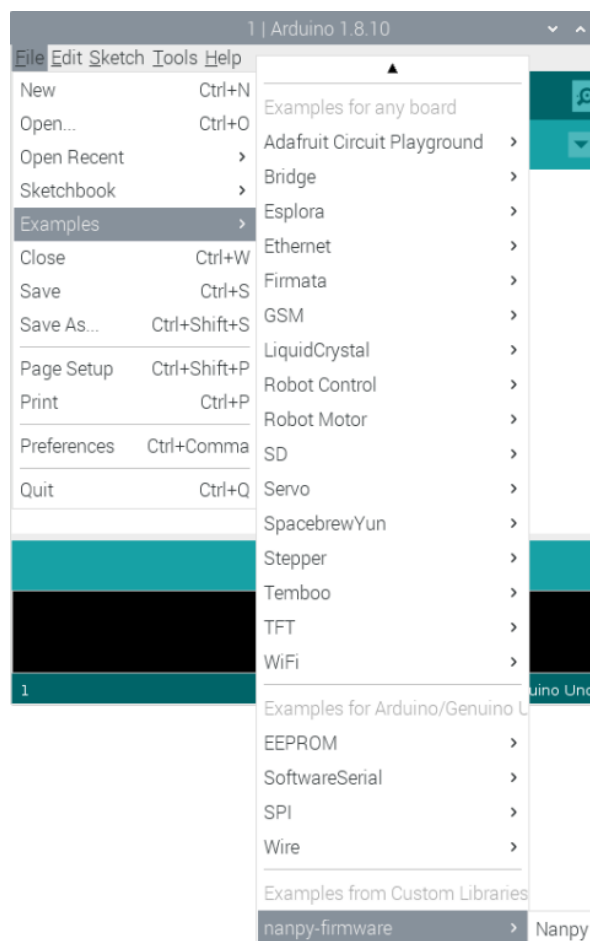
Connect the Atmega328p via the *USB* cable to the Raspberry Pi and then open the Arduino IDE in the Raspbian operating system. Check if the Arduino IDE can detect the *USB* port to which the Atmega328p is connected:

Tools > Port > dev/ttyUSB0

Then go to: *Tools > Board > {board name}*
and select *your* board.

Then, to open a sketch for the *nanpy-firmware*, go to:

File > Examples > nanpy-firmware > Nanpy

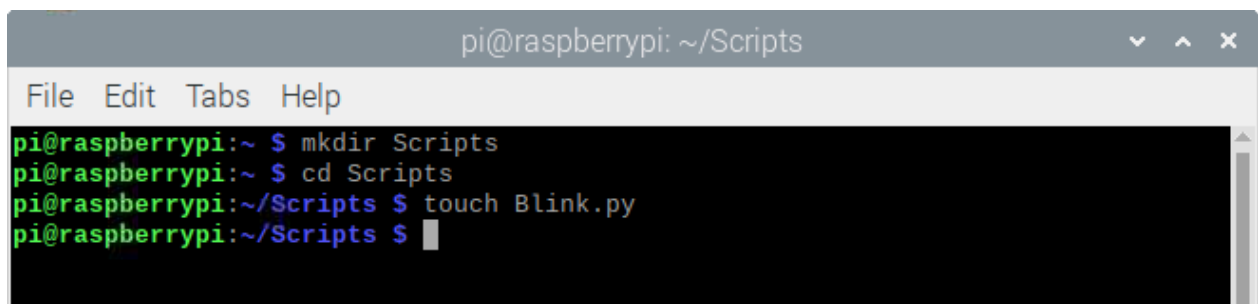


Az-Delivery

Upload the sketch to the Atmega328p. To test if everything works properly, the simple *Blink* script has to be created, where the on-board LED of the Atmega328p is blinked.

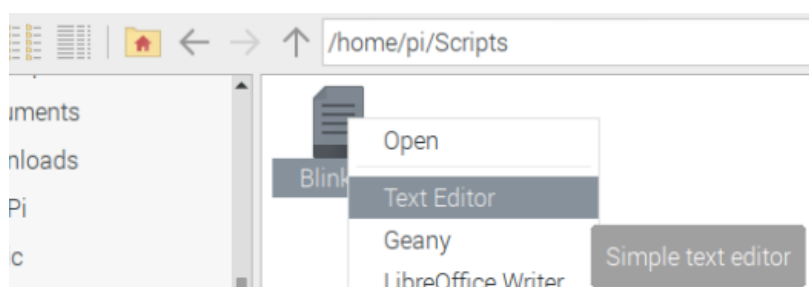
Open terminal, create the *Scripts* directory and the *Blink.py* script. To do so, run the following commands, one by one:

mkdir Scripts	- create to the <i>Scripts</i> directory
cd /Scripts	- change to the <i>Scripts</i> directory
touch Blink.py	- create new file called <i>Blink.py</i>



```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~ $ mkdir Scripts
pi@raspberrypi:~ $ cd Scripts
pi@raspberrypi:~/Scripts $ touch Blink.py
pi@raspberrypi:~/Scripts $
```

Open *File Explorer*, navigate to the *Scripts* directory and open *Blink.py* script in the default text editor:



Az-Delivery

In the *Blink.py* script write the following lines of code:

```
from nanpy import (ArduinoApi, SerialManager)
from time import sleep
```

```
ledPin = 13
```

```
try:
```

```
    connection1 = SerialManager()
```

```
    a = ArduinoApi(connection=connection1)
```

```
except:
```

```
    print('Failed to connect to the Arduino')
```

```
print('[Press CTRL + C to end the script!']')
```

```
a.pinMode(ledPin, a.OUTPUT) # Setup Arduino
```

```
try:
```

```
    while True:
```

```
        a.digitalWrite(ledPin, a.HIGH)
```

```
        print('Bulit in led HIGH')
```

```
        sleep(1)
```

```
        a.digitalWrite(ledPin, a.LOW)
```

```
        print('Bulit in led LOW')
```

```
        sleep(1)
```

```
except KeyboardInterrupt:
```

```
    print('\nScript end!')
```

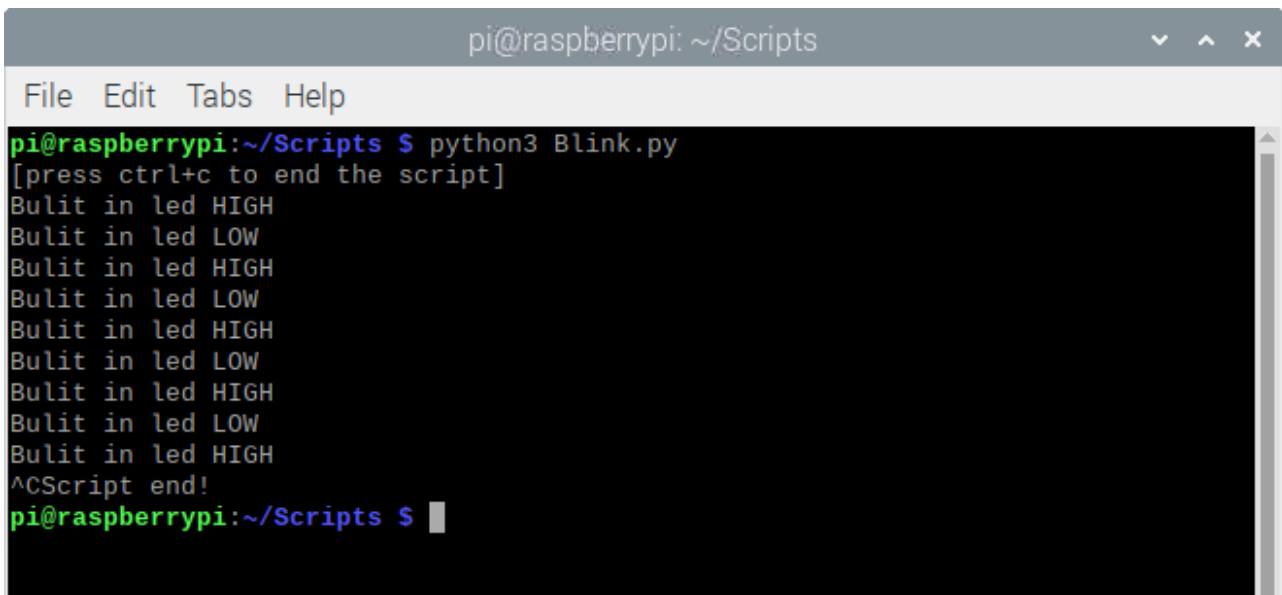
```
    a.digitalWrite(ledPin, a.LOW)
```

Az-Delivery

Save the script. To run the script, open terminal in the directory where the script is saved and run the following command:

python3 Blink.py

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



```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ python3 Blink.py
[press ctrl+c to end the script]
Bulit in led HIGH
Bulit in led LOW
Bulit in led HIGH
Bulit in led LOW
Bulit in led HIGH
Bulit in led LOW
Bulit in led HIGH
Bulit in led LOW
Bulit in led HIGH
Bulit in led LOW
Bulit in led HIGH
^CScript end!
pi@raspberrypi:~/Scripts $
```

To stop the script press *CTRL* + *C* on the keyboard.

The LED connected to the digital pin 13 of the Atmega328p should start blinking every second.

Az-Delivery

The script starts with importing two libraries, the *nanpy* library functions, and the *time*.

Then, the variable called *ledPin* is created and initialized with number *13*. The number *13* represents the number of the digital pin on which LED is connected (on-board LED of the Atmega328p).

After that, the *try-except* block of code is used to try and connect to the Atmega328p. If the connection is not successful, the message:

Failed to connect

is displayed in the terminal.

If the connection is successful, the communication object called "*a*" is created and initialized. The object "*a*" represents the Atmega328p board. Any function used in the Arduino IDE can be used with the "*a*" object, as it can be seen in the code.

With the following line of code, the pin mode of the digital pin *13* is set-up:
`a.pinMode(ledPin, a.OUTPUT)`

Then, in the indefinite loop block (*while True:*) the function *digitalWrite()* is used to set the state of the digital pin *13* (*HIGH* or *LOW* state). With *digitalWrite()* function the LED connected to the pin *13* can be turned *ON* or *OFF*.

Az-Delivery

In the indefinite loop block, the LED is first turned *ON* for a second, and then turned *OFF* for a second. This is called *blinking the LED*.

The time interval of a single blink can be changed in the following line of code: `sleep(1)`

Where the number *1* represents the number of seconds for the duration of the time interval.

To end the infinite loop press *CTRL + C* on the keyboard. This is called the keyboard interrupt, which is set in the *except* block (except *KyeboardInterrupt*). In the *expect* block the on-board LED is turned *OFF*.

Az-Delivery

Python script for KY-023 module

Connect the KY-023 module with the Atmega328p as shown on the connection diagram from the chapter *Connecting the module with Atmega328p*, and then connect the Atmega328p with the Raspberry Pi via usb cable. Next, upload *nanpy* firmware to the Atmega328p, and use the following code to controll the KY-023 module:

```
from nanpy import (ArduinoApi, SerialManager)
from time import sleep
try:
    connection1 = SerialManager()
    a = ArduinoApi(connection=connection1)
except:
    print('Failed to connect to the Arduino')

# Pin setup for the Arduino
a.pinMode(2, a.INPUT_PULLUP)
valueX, valueY, valueZ = 0, 0, 0
print('[Press CTRL + C to end the script!}')
try: # Main program loop
    while True:
        valueX = a.analogRead(0)
        valueY = a.analogRead(1)
        valueZ = not a.digitalRead(2)
        print('X:{}\tY:{}\tZ:{}'.format(valueX, valueY, valueZ))
        sleep(0.5)

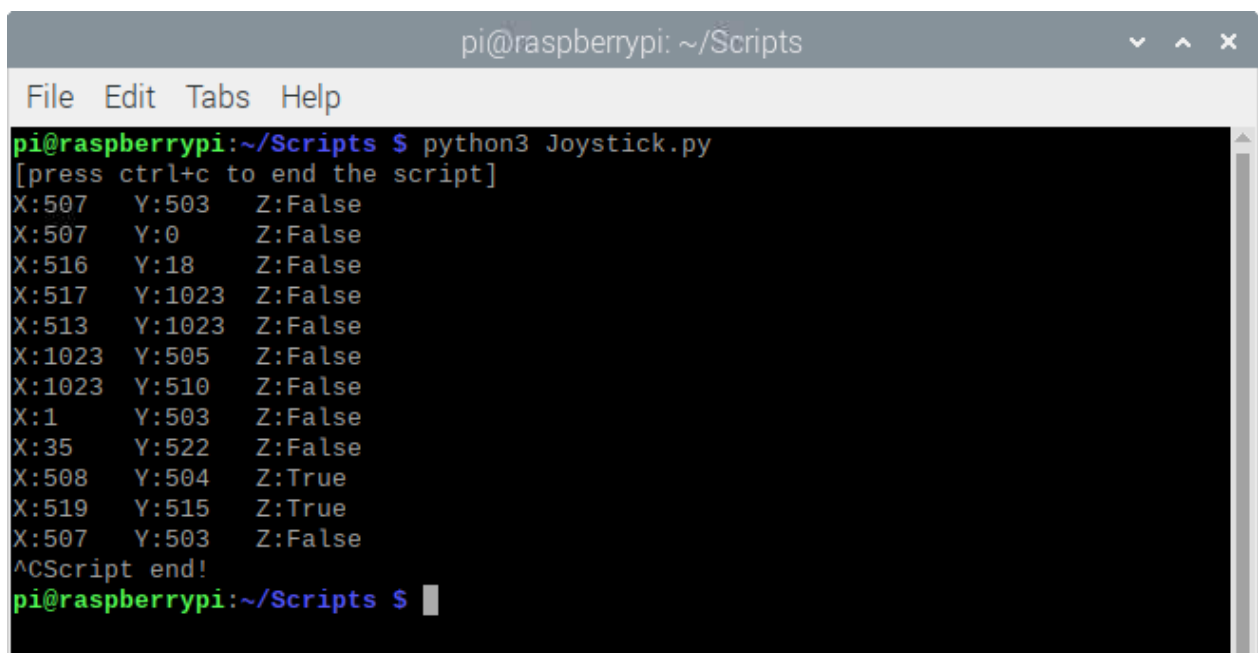
except KeyboardInterrupt:
    print('\nScript end!')
```

Az-Delivery

Save the script by the name *Joystick.py*. To run the script open terminal in the directory where the script is saved and run the following command:

python3 Joystick.py

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



```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ python3 Joystick.py
[press ctrl+c to end the script]
X:507 Y:503 Z:False
X:507 Y:0 Z:False
X:516 Y:18 Z:False
X:517 Y:1023 Z:False
X:513 Y:1023 Z:False
X:1023 Y:505 Z:False
X:1023 Y:510 Z:False
X:1 Y:503 Z:False
X:35 Y:522 Z:False
X:508 Y:504 Z:True
X:519 Y:515 Z:True
X:507 Y:503 Z:False
^CScript end!
pi@raspberrypi:~/Scripts $
```

To stop the script press *CTRL + C* on the keyboard.



Now it is the time to learn and make your own projects. 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>