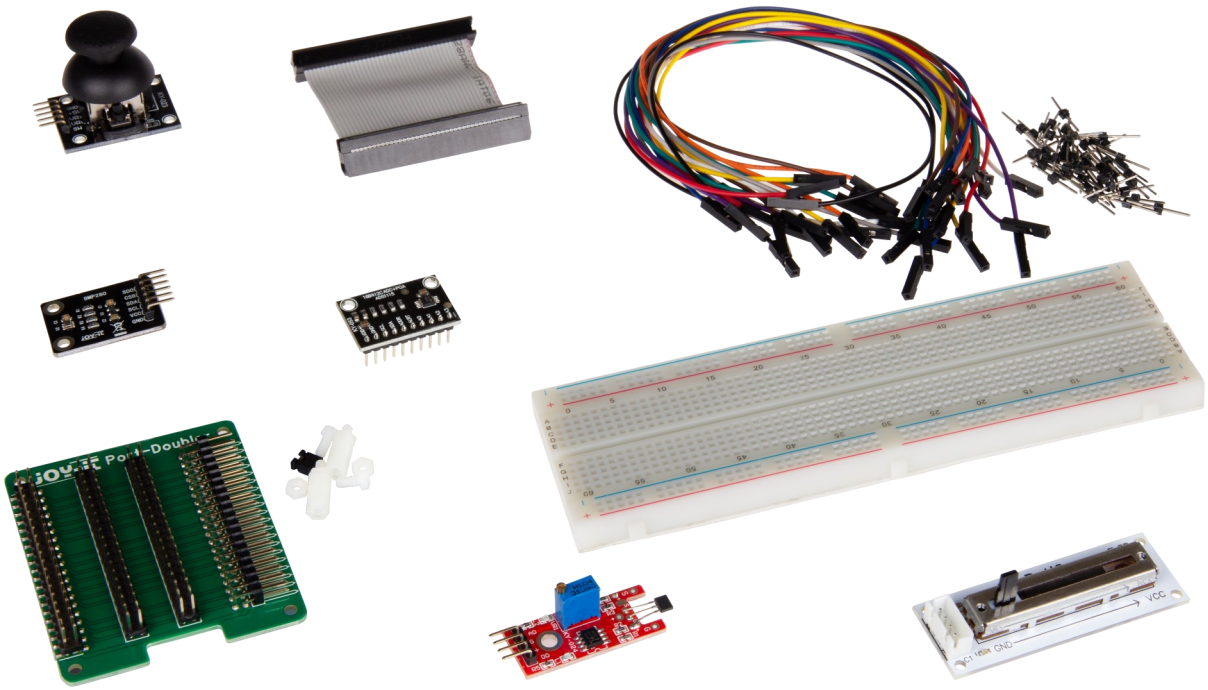
JOYPI EXPANSION KIT

RB-JoyPi-Addon



1. GENERAL INFORMATION

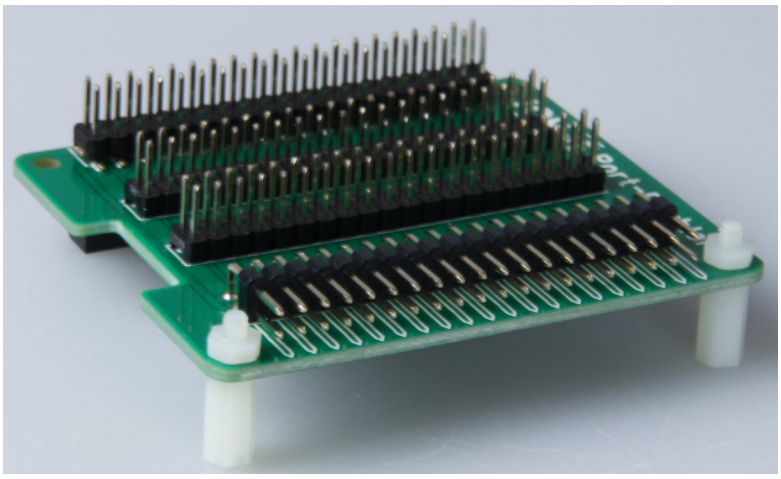
Dear Customer,
thank you for choosing our product. In the below, we will show you what you need to know about commissioning and use of the product.

If you encounter any unexpected problems during use, please feel free to contact us.

These instructions only refer to the expansion pack for the JoyPi. If you have any questions about the basic functions of the JoyPi, you can download the main JoyPi manual here.

2. ASSEMBLY

First, screw the spacers to the port doubler as seen in the picture below.



Your Raspberry Pi should already be screwed into the JoyPi.

Now you can plug the port doubler onto the Raspberry Pi.

The port doubler is not screwed to the Raspberry Pi, because the port doubler has to be removed again when you want to close the JoyPi.

Finally, you only need to connect the GPIO strip of the JoyPi, with the GPIO strip of the port doubler with the supplied cable.

3. INSTALLATION

For the commissioning of the ADC and the pressure and temperature sensor, it is necessary to install the corresponding libraries.

In our prepared image, which you can download here, they are already installed. If you use your own image, you can install the libraries with the following commands.

adafruit/**Adafruit_CircuitPython_ADS1x15**

```
sudo pip3 install adafruit-circuitpython-ads1x15
```

adafruit/**Adafruit_CircuitPython_BMP280**

```
sudo apt-get install python3-smbus i2c-tools -y
```

```
sudo pip3 install adafruit-circuitpython-bmp280
```

The sample scripts we prepared, are also already present on the image. You can also download the script directory manually here, or save it to your desktop with the following commands:

```
cd Desktop/
```

```
wget https://joy-pi.net/files/files/downloads/joypi/Joy-Pi.zip
```

```
unzip Joy-Pi.zip
```

Also, if you are using your own image, you need to enable I2C. To do this, enter the following command in the terminal:

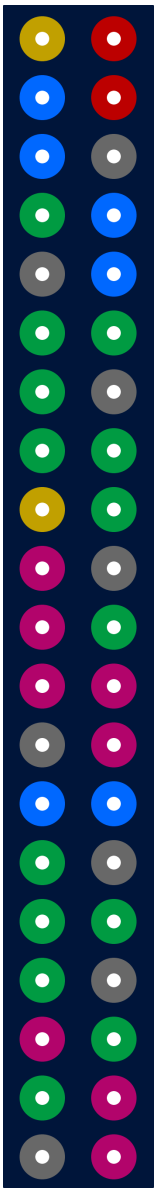
```
sudo raspi-config
```

Now go to **3Interfacing Options** and enable **I5 I2C**.

4. THE PORT-DOUBLER

The Port-Doubler has 4 GPIO-strips and allows the use of external sensors with the JoyPi.

The pinout of each of the four GPIO strips of the Port Doubler is the same as that of the Raspberry Pi.

1	3.3 V DC		**2**	5 V DC
3	GPIO 2 (SDA1, I2C)		**4**	5 V DC
5	GPIO 3 (SCL1, I2C)		**6**	Ground
7	GPIO 4		**8**	GPIO 14 (TXD0)
9	Ground		**10**	GPIO 15 (RXD0)
11	GPIO 17		**12**	GPIO 18
13	GPIO 27		**14**	Ground
15	GPIO 22		**16**	GPIO 23
17	3.3 V		**18**	GPIO 24
19	GPIO 10 (SPI, MOSI)		**20**	Ground
21	GPIO 9 (SPI, MISO)		**22**	GPIO 25
23	GPIO 11 (SPI, CLK)		**24**	GPIO 8 (SPI)
25	Ground		**26**	GPIO 7 (SPI)
27	ID_SD (I2C, EEPROM)		**28**	ID_SC
29	GPIO 5		**30**	Ground
31	GPIO 6		**32**	GPIO 12
33	GPIO 13		**34**	Ground
35	GPIO 19		**36**	GPIO 16
37	GPIO 26		**38**	GPIO 20
39	Ground		**40**	GPIO 21

Since all pins of the Raspberry Pi are used in the JoyPi, you can only use the pins that you can disconnect using the two switching units of the Joy-Pi. These are highlighted in the diagram above.

In addition, pins 3 and 5 can be used for I2C communication.

5. ANALOG-DIGITAL-CONVERTER

The analog-to-digital converter (ADC) converts an analog voltage into a digital signal, which can be retrieved from the Raspberry Pi using the I2C protocol.

The pins A0 - A3 of the ADC are the 4 analog input channels. Thus, 4 analog sensors can be operated simultaneously.

Please note that the voltage on the input channels of the ADC must not exceed 3.3 V.

The further pin assignment can be found in the table below.

ADC	Raspberry Pi
VDD	3.3 V (Pin 1)
GND	GND (Pin 6)
SCL	SCL (Pin 5)
SDA	SDA (Pin 3)

The following code example outputs the voltages applied to all 4 input channels.

```
import time
import board
import busio
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn

# Create the I2C bus
i2c = busio.I2C(board.SCL, board.SDA)

# Create the ADC object using the I2C bus
ads = ADS.ADS1115(i2c)
ads.gain = 2/3
# Create single-ended input on channels
chan0 = AnalogIn(ads, ADS.P0)
chan1 = AnalogIn(ads, ADS.P1)
chan2 = AnalogIn(ads, ADS.P2)
chan3 = AnalogIn(ads, ADS.P3)



while True:
    print("channel 0: ","{:>5}\t{:>5.3f}".format(chan0.value, chan0.voltage))
    print("channel 1: ","{:>5}\t{:>5.3f}".format(chan1.value, chan1.voltage))
    print("channel 2: ","{:>5}\t{:>5.3f}".format(chan2.value, chan2.voltage))
    print("channel 3: ","{:>5}\t{:>5.3f}".format(chan3.value, chan3.voltage))
    print("--------------------------------------------------")
    time.sleep(1)
```

You can run the file with the following command:

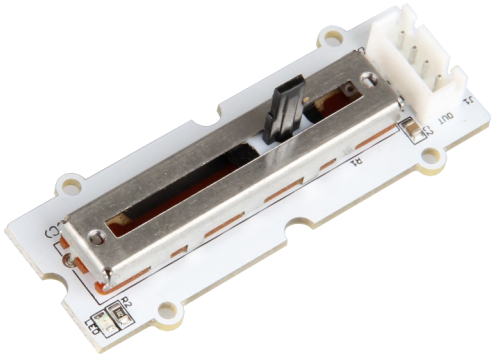
```
python3 /home/pi/Desktop/Joy-Pi/Extensions/adc.py
```

6. SLIDING POTENTIOMETER OR ROTARY POTENTIOMETER

In this expansion pack there is either a sliding potentiometer or a rotary potentiometer. These differ only in the type of operation (shifting or turning a controller).

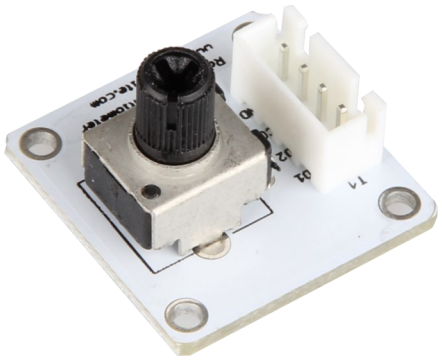
An analog-digital converter is required to use both potentiometers. You can find the pin assignment of the respective potentiometer in the following table.

The slider potentiometer is a linear variable resistor with a total resistance of 10 kΩ. Moving the slider from one side to the other changes its output voltage between 0V and the supply voltage.



Sliding potentiometer	ADC	Raspberry Pi
OUT	A0	-
LED	-	-
U	VDD	3.3 V (Pin 1)
G	GND	GND (Pin 6)
-	SCL	SCL (Pin 5)
-	SDA	SDA (Pin 3)

The rotary potentiometer is a variable resistor with a total resistance of 10 kΩ. Turning the rotary potentiometer from one side to the other changes its output voltage between 0V and the supply voltage.



Rotary potentiometer	ADC	Raspberry Pi
OUT	A0	-
VCC	VDD	3.3 V (Pin 1)
GND	GND	GND (Pin 6)
-	SCL	SCL (Pin 5)
-	SDA	SDA (Pin 3)

The following code example outputs the voltages applied to the output pin of the potentiometer.

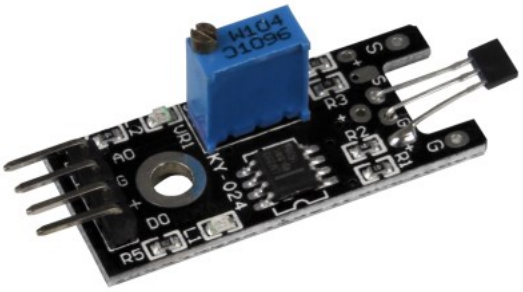
```
import RPi.GPIO as GPIO
import time
import board
import busio
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
max_limit = 3.2                 # Voltage max limit
min_limit = 0.1                 # Voltage min limit
GPIO.setmode(GPIO.BCM)
# Create the I2C bus
i2c = busio.I2C(board.SCL, board.SDA)

# Create the ADC object using the I2C bus
ads = ADS.ADS1115(i2c)
ads.gain = 2/3
# Create single-ended input on channels
chan0 = AnalogIn(ads, ADS.P0)


while True:
    if min_limit > chan0.voltage or max_limit < chan0.voltage:
      print("Potentiometer Voltage: ","{:>5.3f}".format(chan0.voltage),"V / Limit reached!")
      print("-------------------------------------------------")
      time.sleep(1)
    else:
      print("Potentiometer Voltage: ","{:>5.3f}".format(chan0.voltage),"V / Limit not reached!")
      print("-------------------------------------------------")
      time.sleep(1)
```

You can run the file with the following command:

```
python3 /home/pi/Desktop/Joy-Pi/Extensions/poti.py
```

7. MAGNETIC SENSOR

The magnetic field is measured by the sensor and output as an analog voltage value. If the limit value is exceeded, a high signal is applied to the digital pin.
This limit value can be set with the help of the blue potentiometer.



For the use of the magnetic sensor you additionally need the ADC.

The pin assignment can be found in the table below.

Magnetic sensor	ADC	Raspberry Pi
A0	A0	-
G	GND	GND
+	VDD	3.3 V (Pin 1)
D0	-	GPIO 5 (Pin 29)
-	SCL	SCL (Pin 5)
-	SDA	SDA (Pin 3)

The following code example outputs the voltages that are present at the output pin of the magnetic sensor.

It also displays whether the limit value has been reached or not.

```
import time
import board
import busio
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
# Create the I2C bus
i2c = busio.I2C(board.SCL, board.SDA)

# Create the ADC object using the I2C bus
ads = ADS.ADS1115(i2c)

# Create single-ended input on channels
chan0 = AnalogIn(ads, ADS.P0)

delayTime = 1
Digital_PIN = 5 #GPIO 5 (Pin29) Raspberry Pi

GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

while True:
                analog = '%.2f' % chan0.voltage


                if GPIO.input(Digital_PIN) == False:
                        print ("Analog Voltage:", analog,"V, ","Limit not reached")
                else:
                        print ("Analog Voltage:", analog, "V, ", "Limit reached")
                print ("---------------------------------------")

                # Reset + Delay
                button_pressed = False
                time.sleep(delayTime)
```

You can run the file with the following command:

```
python3 /home/pi/Desktop/Joy-Pi/Extensions/magnet.py
```

8. JOYSTICK

X and Y position of the joystick, are output as analog voltage on the output pins.

In this joystick for the X-axis, as well as for the Y-axis, a separate potentiometer for the X-axis and for the Y-axis.

In addition, a button is built into the joystick, which detects when the joystick is pressed down.

To use the joystick, you also need the ADC. The pin assignment can be found in the table below.

Joystick	ADC	Raspberry Pi
GND	GND	GND (Pin 6)
+5V	VCC	3.3 V (Pin 1)
VRx	A0	-
VRy	A1	-
SW	-	GPIO 6 (Pin 31)
-	SCL	SCL (Pin 5)
	SDA	SDA (Pin 3)

The following code example outputs the voltages applied to the X and Y axis pins graphically in a diagram.

```
import matplotlib.pyplot as plt
import numpy as np
import time
import board
import busio
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

Button_PIN = 6
GPIO.setup(Button_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)
# Create the I2C bus
i2c = busio.I2C(board.SCL, board.SDA)

# Create the ADC object using the I2C bus
ads = ADS.ADS1115(i2c)
ads.gain = 2/3
# Create single-ended input on channels
chan0 = AnalogIn(ads, ADS.P0)
chan1 = AnalogIn(ads, ADS.P1)

x = np.linspace(0, 6*np.pi, 100)
y = np.sin(x)

# You probably won't need this if you're embedding things in a tkinter plot...
plt.ion()

fig = plt.figure()
ax = fig.add_subplot(111)
#line1, = ax.plot(x, y, 'r-') # Returns a tuple of line objects, thus the comma

for phase in np.linspace(0, 10*np.pi, 500):
    plt.cla()
    plt.xlim(0, 3.3)
    plt.ylim(3.3, 0)
    plt.xlabel("Voltage X")
    plt.ylabel("Voltage Y")
    if GPIO.input(Button_PIN) == False:
        plt.title("Button pressed!")

    plt.plot(chan0.voltage,chan1.voltage,'ro')
    fig.canvas.draw()
    fig.canvas.flush_events()
```

It also indicates whether the joystick is pressed or not.

```
python3 /home/pi/Desktop/Joy-Pi/Extensions/joystick.py
```

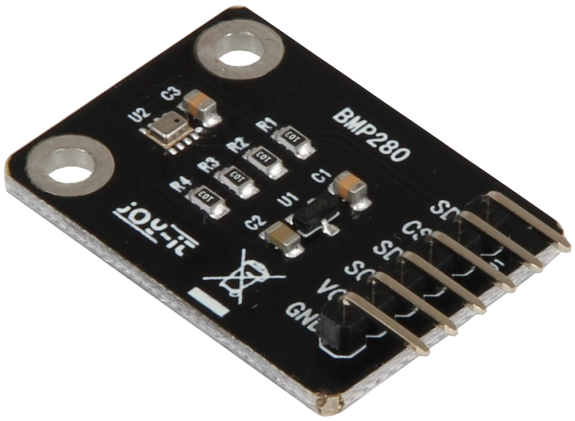
You can run the file with the following command:

If you run the program over an SSH connection use the following command:

```
sudo -E python3 /home/pi/Desktop/Joy-Pi/Extensions/joystick.py
```

9. PRESSURE AND TEMPERATURE SENSOR

This combination sensor can measure both temperature and air pressure. For the communication with the Raspberry Pi it uses, just like the ADC, the I2C protocol.



The pin assignment can be found in the table below.

BMP280	Raspberry Pi
SD0	3.3 V (Pin 1)
CSB	3.3 V (Pin 1)
SDA	SDA (Pin 3)
SCL	SCL (Pin 5)
VCC	3.3 V (Pin 1)
GND	GND (Pin 6)

The following code example outputs the current temperature and air pressure.

```
# Required modules are imported and set up
import time
import board
# import digitalio for use with SPI
import adafruit_bmp280

# Create sensor object that communicates via the RPi's standard I2C bus.
i2c = board.I2C()   # uses board.SCL and board.SDA
bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c)

# OR Create sensor object that communicates over the RPi's standard SPI bus.
# spi = board.SPI()
# bmp_cs = digitalio.DigitalInOut(board.D10)
# bmp280 = adafruit_bmp280.Adafruit_BMP280_SPI(spi, bmp_cs)

# Change this value so that it corresponds to the air pressure (hPa) at sea level at your location.
bmp280.sea_level_pressure = 1013.25

while True:
    print("\nTemperature: %0.1f C" % bmp280.temperature)
    print("Pressure: %0.1f hPa" % bmp280.pressure)
    print("Altitude = %0.2f meters" % bmp280.altitude)
    time.sleep(2)
```

You can run the file with the following command:

```
python3 /home/pi/Desktop/Joy-Pi/Extensions/bmp.py
```

10. ADDITIONAL INFORMATION

Our information and take-back obligations under the Electrical and Electronic Equipment Act (ElektroG)

Symbol on electrical and electronic equipment:
This crossed-out dustbin means that electrical and electronic appliances do not belong in the household waste. You must return the old appliances to a collection point. Before handing over waste batteries and accumulators that are not en-closed by waste equipment must be separated from it.

Return options:
As an end user, you can return your old device (which essentially fulfils the same function as the new device purchased from us) free of charge for disposal when you purchase a new device. Small appliances with no external dimensions greater than 25 cm can be disposed of in normal household quantities independently of the pur-chase of a new appliance.

Possibility return to our company location during the opening hours:
SIMAC Electronics GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

Possibility return in your area:
We will send you a parcel stamp with which you can return the device to us free of charge. Please contact us by email at Service@joy-it.net or by telephone.

Packaging information:
If you do not have suitable packaging material or do not wish to use your own, please contact us and we will send you suitable packaging.

11. SUPPORT

We are also there for you after the purchase. If any questions remain or problems arise, we are also available to assist you via email, phone and ticket support system.

E-Mail: service@joy-it.net
Ticket-System: http://support.joy-it.net
Phone: +49 (0)2845 9360 – 50 (10 - 17 Uhr)

For more information, visit our website:
www.joy-it.net

Published: 07.03.2022