KY-023 Joystick module (XY-Axis)

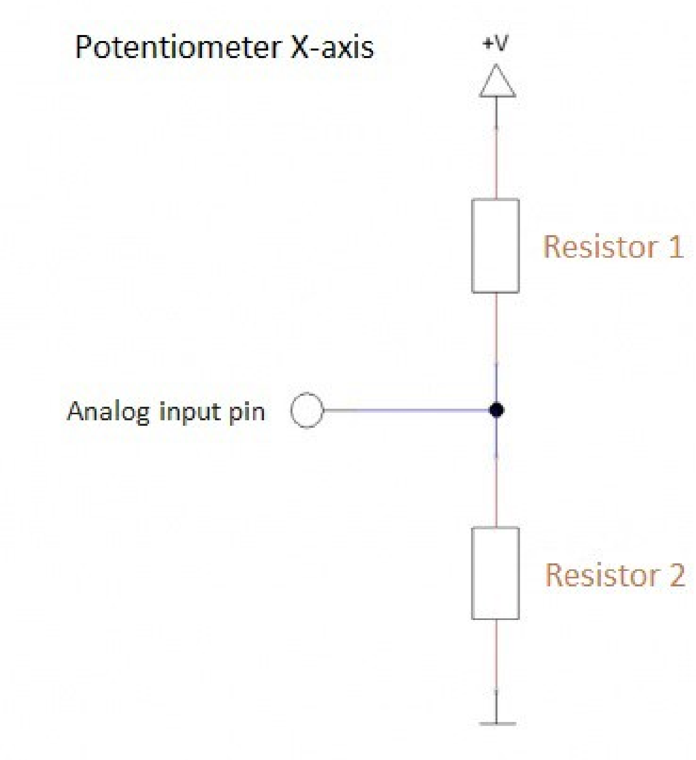
Contents

Picture

Technical data / Short description

X and Y positions of the joystick can be measured as an analog voltage at the output pin.

In this joystick, the x-axis and the y-axis have their own potentiometer. Together, they build a voltage devider like the one in the next picture.

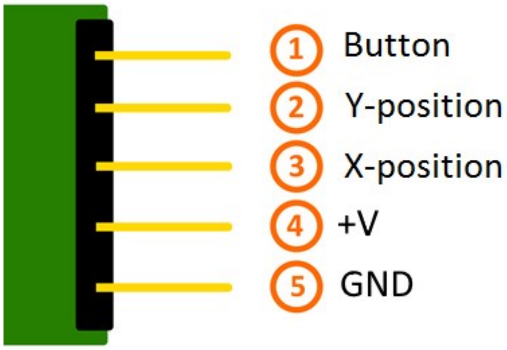


In the non-operating mode, the potentiometer is in the middle so that resistor1=resistor2, so that the voltage will equally split to both resistors - e.g. Measurement of +V=5V -> 2,5V.

If one of the axis changes, like the x-axis for example, the values of the resistors will change - e.g. value of resistor 1 will raise than the value of resistor 2 will fall or the value of resistor 1 will fall and the value of resistor 2 will raise.

According to the division of the resistor values, you can measure a specific voltage value between the resistors and locate the position of the axis.

Pinout



Code example Arduino

This program measures the value at the input pins, converts them into a voltage value (0-1023 -> 0V-5V) and prints these at the serial ouput.

```
// Declaration and initialization of the input pin
int JoyStick_X = A0; // X-axis-signal
int JoyStick_Y = A1; // Y-axis-signal
int Button = 3;

void setup ()
{
  pinMode (JoyStick_X, INPUT);
  pinMode (JoyStick_Y, INPUT);
  pinMode (Button, INPUT);

  // pushing the button leads to
  // power up the pullup-resistor
  digitalWrite(Button, HIGH);

  Serial.begin (9600); // serial output with 9600 bps
}

// The program reads the current values of the input pins
// and outputs them at the serial output
void loop ()
{
  float x, y;
  int Knopf;

  // Current values will be read and converted to the right voltage
  x = analogRead (JoyStick_X) * (5.0 / 1023.0);
  y = analogRead (JoyStick_Y) * (5.0 / 1023.0);
  Knopf = digitalRead (Button);

  //... and outputted here
  Serial.print ("X-axis:"); Serial.print (x, 4);  Serial.print ("V, ");
  Serial.print ("Y-axis:"); Serial.print (y, 4);  Serial.print ("V, ");
  Serial.print ("Button:");

  if(Knopf==1)
  {
      Serial.println ("not pushed");
  }
  else
  {
      Serial.println ("pushed");
  }
  delay (200);
}
```

Connections Arduino:

Button	=	[Pin 3]
Y-Position	=	[Pin A1]
X-Position	=	[Pin A0]
Sensor +V	=	[Pin 5V]
Sensor GND	=	[Pin GND]

Example program download:

KY-023_Joystick_Modul

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

The program reads the current values of the input pins and prints them to the terminal.

```
##########################################################################################
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported Lic
### Commercial use only after permission is requested and granted
###
### KY-053 Analog Digital Converter - Raspberry Pi Python Code Example
###
##########################################################################################


# This code is using the ADS1115 and the I2C Python Library for Raspberry Pi
# This was published on the following link under the BSD license
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# import needed modules
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# initialise variables
delayTime = 0.5 # in Sekunden

# assigning the ADS1x15 ADC

ADS1015 = 0x00  # 12-bit ADC
ADS1115 = 0x01  # 16-bit
```

```
# choosing the amplifing gain
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# choosing the sampling rate
# sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# assigning the ADC-Channel (1-4)
adc_channel_0 = 0    # Channel 0
adc_channel_1 = 1    # Channel 1
adc_channel_2 = 2    # Channel 2
adc_channel_3 = 3    # Channel 3

# initialise ADC (ADS1115)
adc = ADS1x15(ic=ADS1115)

Button_PIN = 24
GPIO.setup(Button_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

########################################################################################


# ########
# main program loop
# ########
# The program reads the current values of the input pins
# and outputs the values at the terminal

try:
        while True:
                # Current values will be recorded
                x = adc.readADCSingleEnded(adc_channel_0, gain, sps)
                y = adc.readADCSingleEnded(adc_channel_1, gain, sps)

                # Output at the terminal
                if GPIO.input(Button_PIN) == True:
                        print "X-axis:", x,"mV, ","Y-axis:", y,"mV, Button: not pushed"
                else:
                        print "X-axis:", x, "mV, ", "Y-axis:", y, "mV, Button: pushed"
                print "---------------------------------------"

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


except KeyboardInterrupt:
        GPIO.cleanup()
```

Connections Raspberry Pi:

Sensor KY-023

Button	=	GPIO24	[Pin 18 (RPi)]
Y-position	=	Analog 1	[Pin A1 (ADS1115 - KY-053)]

X-position	=	Analog 0	[Pin A0 (ADS1115 - KY-053)]
+V	=	3,3V	[Pin 1 (RPi)]
GND	=	GND	[Pin 6 (RPi)]

ADS1115 - KY-053:

VDD	=	3,3V	[Pin 01]
GND	=	GND	[Pin 09]
SCL	=	GPIO03 / SCL	[Pin 05 (RPi)]
SDA	=	GPIO02 / SDA	[Pin 03 (RPi)]
A0	=	look above	[Sensor: X-position (KY-023)]
A1	=	look above	[Sensor: Y-position (KY-023)]

Example program download

KY-023_Joystick_RPi

To start, enter the command:

```
sudo python KY-023_Joystick_RPi.py
```