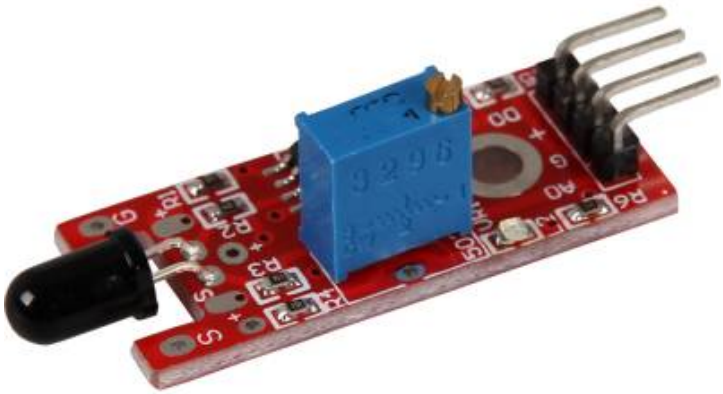
KY-026 Détecteur de flamme

Sommaire

Photo



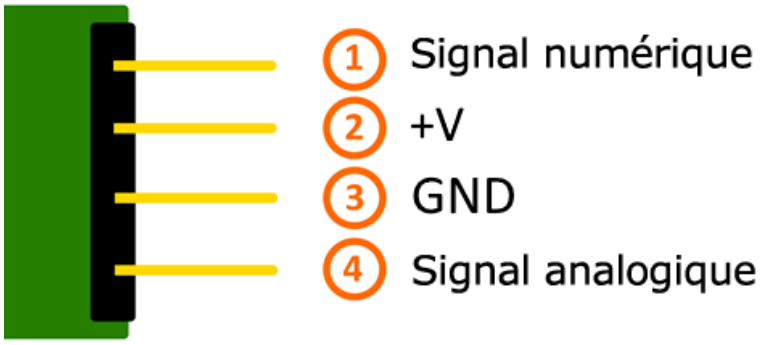
Données techniques / Description sommaire

La photodiode est sensible spectre lumineux généré par une flamme.

Sortie numérique: un signal est émis si une flamme détectée.
Sortie analogique: mesure directe du capteur

LED1: indique que le capteur est alimenté en tension
LED2: indique qu'une flamme est détectée

Brochage

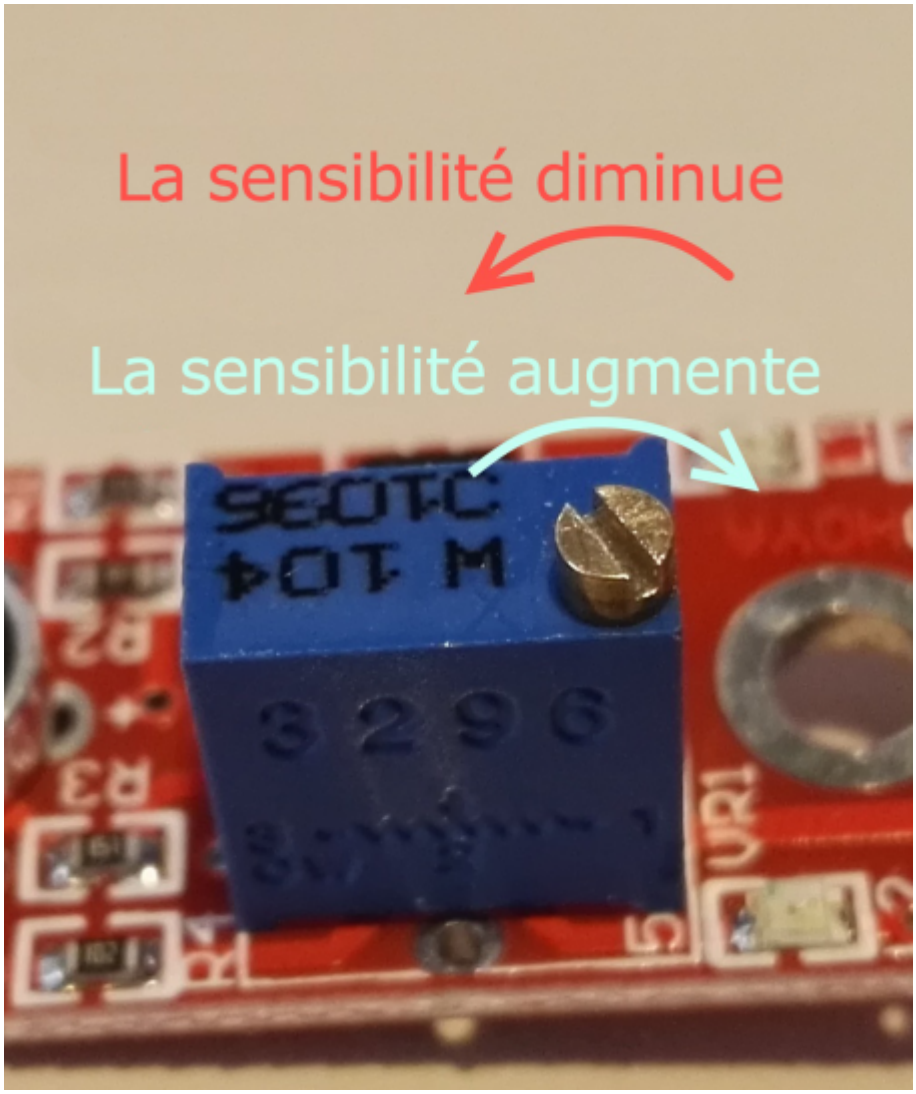


Funktionsweise des Sensors

Dieser Sensor besitzt auf seiner Platine drei funktionelle Bestandteile. Die ist die Sensoreinheit vorne am Modul, welche das aktuelle Umfeld physikalisch misst und als analoges Signal auf die zweite Einheit, dem Verstärker, ausgibt. Dieser verstärkt das Signal abhängig vom eingestellten Widerstand am Drehpotentiometer und leitet es auf den analogen Ausgang des Moduls.

Hierbei ist zu beachten: Das Signal ist invertiert; wird ein hoher Wert gemessen, so resultiert dies in einen niedrigeren Spannungswert am analogen Ausgang.

Die dritte Einheit stellt einen Komparator dar, welcher den digitalen Ausgang und die LED schaltet, wenn das Signal unter einen bestimmten Wert fällt. Mittels des Drehpotentiometers kann somit die Empfindlichkeit eingestellt werden, wie es im folgenden Bild aufgezeigt wird:



Dieser Sensor gibt somit keine absoluten Werte aus (z.B. genau gemessene Temperatur in °C oder Magnetfeldstärke in mT) , sondern es handelt sich hierbei um eine Relativ-Messung: Man definiert hierbei einen Grenzwert relativ zur gegebenen normalen Umweltsituation und es wird ein Signal ausgegeben was weiterverarbeitet werden kann, falls dieser Grenzwert überschritten bzw. ein anderer Zustand als der Normalfall eingetreten ist.

Dieses Verhalten eignet sich hervorragend zur Temperaturüberwachung (KY-028), Näherungsschalter (KY-024, KY-025, KY-036), Alarmüberwachungen (KY-037, KY-038) oder Drehgeber (KY-026).

Codebeispiel Arduino

Das Programm liest den aktuellen Spannungswert aus, der am analogen Ausgang gemessen werden kann, und gibt diesen auf der seriellen Schnittstelle aus.

Zudem wird ebenfalls der Zustand des digitalen Pins in der Konsole angegeben, was bedeutet ob der Grenzwert unterschritten wurde oder nicht.


```
// Deklaration und Initialisierung der Eingang-Pins
int Analog_Eingang = A0; // X-Achse-Signal
int Digital_Eingang = 3; // Knopf

void setup ()
{
  pinMode (Analog_Eingang, INPUT);
  pinMode (Digital_Eingang, INPUT);

  Serial.begin (9600); // Serielle Ausgabe mit 9600 bps
}

// Das Programm liest die aktuellen Werte der Eingang-Pins
// und gibt diese auf der seriellen Ausgabe aus
void loop ()
{
  float Analog;
  int Digital;

  //Aktuelle Werte werden ausgelesen, auf den Spannungswert konvertiert...
  Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
  Digital = digitalRead (Digital_Eingang);

  //... und an dieser Stelle ausgegeben
  Serial.print ("Analoger Spannungswert:"); Serial.print (Analog, 4);  Serial.print ("V, "
  Serial.print ("Grenzwert:");

  if(Digital==1)
  {
      Serial.println (" erreicht");
  }
  else
  {
      Serial.println (" noch nicht erreicht");
  }
  Serial.println ("-----------------------------------------------------------------");
  delay (200);
}
```

Anschlussbelegung Arduino:

digitales Signal	= [Pin 3]
+V	= [Pin 5V]
GND	= [Pin GND]
analoges Signal	= [Pin 0]

Beispielprogramm Download

Ard_Analoger_Sensor.zip

Codebeispiel Raspberry Pi

!! Achtung !! Analoger Sensor !! Achtung !!

Der Raspberry Pi besitzt im Gegensatz zum Arduino keine analogen Eingänge bzw. es ist kein ADC (analog digital Converter) im Chip des Raspberry Pi's integriert. Dies schränkt den Raspberry Pi ein, wenn man Sensoren einsetzen möchte, wo nicht digital Werte ausgegeben werden [Spannungswert überschritten -> digital EIN | Spannungswert unterschritten -> digital AUS | Beispiel: Knopf gedrückt [EIN] Knopf losgelassen [AUS]], sondern es sich hier um einen kontinuierlichen veränderlichen Wert handeln sollte (Beispiel: Potentiometer -> Andere Position = Anderer Spannungswert)

Um diese Problematik zu umgehen, besitzt unser *Sensorkit X40* mit dem **KY-053** ein Modul mit 16 Bit genauen ADC, welches Sie am Raspberry nutzen können, um diesen um 4 analoge Eingänge erweitern zu können. Dieses wird per I2C an den Raspberry Pi angeschlossen, übernimmt die analoge Messung und gibt den Wert digital an den Raspberry Pi weiter.

Somit empfehlen wir, bei analogen Sensoren dieses Sets das KY-053 Modul mit dem besagten ADC dazwischenzuschalten. Nähere Informationen finden Sie auf der Informationsseite zum **KY-053** Analog Digital Converter

!! Achtung !! Analoger Sensor !! Achtung !!

Das Programm nutzt zur Ansteuerung des ADS1115 ADC die entsprechenden ADS1x15 und I2C Python-Libraries der Firma Adafruit. Diese wurden unter dem folgenden Link [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] unter der BSD-Lizenz [Link] veröffentlicht. Die benötigten Libraries sind im unteren Download-Paket enthalten.

Das Programm liest die aktuellen Werte der Eingang-Pins und gibt diese in der Konsole als Wert in [mV] aus.

Zudem wird ebenfalls der Zustand des digitalen Pins in der Konsole angegeben, was bedeutet ob der Grenzwert unterschritten wurde oder nicht.

```
#
#!/usr/bin/python
# coding=utf-8

##########################################################################################
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported Lic
### Commercial use only after permission is requested and granted
###
### Analog Sensor + ADS1115 ADC - Raspberry Pi Python Code Example
###
##########################################################################################


# Dieser Code nutzt die ADS1115 und die I2C Python Library fuer den Raspberry Pi
# Diese ist unter folgendem Link unter der BSD Lizenz veroeffentlicht
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Weitere benoetigte Module werden importiert und eingerichtet
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Benutzte Variablen werden initialisiert
delayTime = 0.2

# Adresszuweisung ADS1x15 ADC

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

# Verstaerkung (Gain) wird ausgewaehlt
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V
```



```
# Abtasterate des ADC (SampleRate) wird ausgewaehlt
# sps = 8    # 8 Samples pro Sekunde
# sps = 16   # 16 Samples pro Sekunde
# sps = 32   # 32 Samples pro Sekunde
sps = 64   # 64 Samples pro Sekunde
# sps = 128  # 128 Samples pro Sekunde
# sps = 250  # 250 Samples pro Sekunde
# sps = 475  # 475 Samples pro Sekunde
# sps = 860  # 860 Samples pro Sekunde

# ADC-Channel (1-4) wird ausgewaehlt
adc_channel = 0    # Channel 0
# adc_channel = 1    # Channel 1
# adc_channel = 2    # Channel 2
# adc_channel = 3    # Channel 3

# Hier wird der ADC initialisiert - beim KY-053 verwendeten ADC handelt es sich um einen A
adc = ADS1x15(ic=ADS1115)

# Hier waehlt man den Eingangs-Pin des digitalen Signals aus
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

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

# ########
# Hauptprogrammschleife
# ########
# Das Programm liest die aktuellen Werte der Eingang-Pins
# und gibt diese in der Konsole aus

try:
        while True:
                #Aktuelle Werte werden aufgenommen
                analog = adc.readADCSingleEnded(adc_channel, gain, sps)

                # Ausgabe auf die Konsole
                if GPIO.input(Digital_PIN) == False:
                        print "Analoger Spannungswert:", analog,"mV, ","Grenzwert: noch ni
                else:
                        print "Analoger Spannungswert:", analog, "mV, ", "Grenzwert: errei
                print "---------------------------------------"

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



except KeyboardInterrupt:
        GPIO.cleanup()
```

Anschlussbelegung Raspberry Pi:

Sensor

digitales Signal	= GPIO 24	[Pin 18 (RPi)]
+V	= 3,3V	[Pin 1 (RPi)]
GND	= Masse	[Pin 06 (RPi)]
analoges Signal	= Analog 0	[Pin A0 (ADS1115 - KY-053)]

ADS1115 - KY-053:

VDD	=	3,3V	[Pin 01]
GND	=	Masse	[Pin 09]
SCL	=	GPIO03 / SCL	[Pin 05]
SDA	=	GPIO02 / SDA	[Pin 03]
A0	=	s.o.	[Sensor: analoges Signal]

Beispielprogramm Download

RPi_AnalogSensor.zip

Zu starten mit dem Befehl:

```
sudo python RPi_AnalogSensor.py
```

