KY-026 Détecteur de flamme

Version du 26 novembre 2016 à 18:01 (voir la source)
Sensorkit wiki admin (discuter | contributions)
(→Codebeispiel Arduino)
← Modification précédente

Ligne 3 :

==Don

− La photodiode est sensible spectre lumineux généré par une flamme.
'''
Sortie numérique:''' un analogique:''' mesure directe du capteur

---

Ligne 10 :

==Brochage==

− [[Fichier:~~4_dig_V_G_An~~.png|~~none~~|450x235px]]

==Fonctionnement du capteur==

Ligne 20 :

− Ce capteur ne ~~délivre pas~~ des valeurs absolues (par exemple, la température mesurée avec précision en ° C ~~sont relatives~~. ~~on définit~~ une valeur limite par rapport à une ~~valeur normale~~ donnée et le ~~module émet~~ un s idéal pour la surveillance de la température (KY-028), les détecteurs de proximité (KY-024, KY 025, KY-036), (KY-026).

==Exemple de code pour Arduino</spa

Ligne 28 :

<pre class="brush:cpp">// Déclaration et initialisation des broches d'entrées

− int Analog_Eingang = A0; // ~~X-Achse-~~Signal

− int Digital_Eingang = 3; // ~~Knopf~~

void setup ()

Ligne 84 :

||Signal analogique

||=

– ||[Pin 0]
|}

'''Exemple de programme à télécharger'''

– [[Média:Ard_Analoger_Sensor.zip|Ard_Analoger_Sensor.zip]]

– ==Codebeispiel Raspberry Pi==
– !! Achtung !! <span style="color: #
Achtung !!

– Der Raspberry Pi besitzt im Gegensatz zum Arduino keine analogen Eingänge bzw. es ist kein ADC (analog
den Raspberry Pi ein, wenn man Sensoren einsetzen möchte, wo nicht digital Werte ausgegeben werden [S
unterschritten -> digital AUS | Beispiel: Knopf gedrückt [EIN] Knop
sich hier um einen kontinuierlichen veränderlichen Wert handeln sollte (Beispiel: Potentiometer -> Andere F

– Um diese Problematik zu umgehen, besitzt unser "Sensorkit X40" mit dem "KY-053" ein Modul mit 16 Bit
4 analoge Eingänge erweitern zu können. Dieses wird per I2C an den Raspberry Pi angeschlossen, übernim
weiter.

– Somit empfehlen wir, bei analogen Sensoren dieses Sets das KY-053 Modul mit dem besagten ADC dazwisc
Informationsseite zum [[KY-053 Analog Digital Converter|"KY-053" Analog Digital Converter]]

– !! Achtung !! <span style="color: #
Achtung !!

– Das Programm nutzt zur Ansteuerung des ADS1115 ADC die entsprechenden ADS1x15 und I2C Python-Libra
– [[https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code https://github.com/adafruit/Adafruit-Raspbe
/licenses/BSD-3-Clause Link]] veröffentlicht. Die benötigten Libraries sind im unteren Download-Paket entha

– 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 Grenz

– <pre class="brush:py">#
– #!/usr/bin/python
– # coding=utf-8


```
  ##################################################################
  ### Copyright by Joy-IT
  ### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
  ### 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
Ligne 134 :
  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
Ligne 149 :
  # 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
Ligne 165 :
  # adc_channel = 3   # Channel 3

– # ~~Hier wird der ADC initialisiert - beim KY-053 verwendeten ADC handelt es sich um einen ADS1115 Chipsal~~
  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)
Ligne 175 :

  # ########
– # ~~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 nicht erreicht~~"
            else:
–   print "~~Analoger Spannungswert~~:", analog, "mV, ", "~~Grenzwert~~: ~~erreicht~~"
            print "--------------------------------------"

Ligne 202 :

  </pre>
– '''~~Anschlussbelegung Raspberry~~ Pi:'''

– ~~Sensor~~

{| style="height: 85px; padding-left: 30px;" width="441"
|-
– ||~~digitales~~ Signal
||=
||GPIO 24
Ligne 223 :
||[Pin 06 (RPi)]
|-
– ||~~analoges~~ Signal
||=
||Analog 0
Ligne 259 :
|}

– '''~~Beispielprogramm Download~~'''
–
– ~~[[Média:RPi_AnalogSensor.zip|RPi_AnalogSensor.zip]]~~

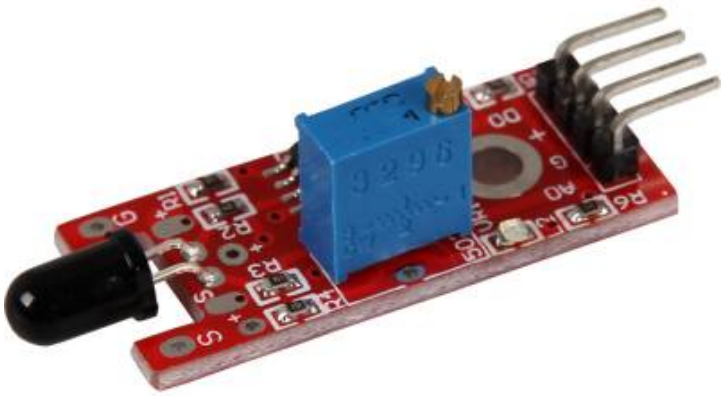
– ~~Zu starten mit dem Befehl~~:

<pre class="brush:bash">sudo python RPi_AnalogSensor.py
</pre>

Version actuelle en date du 30 décembre 2016 à 15:24

Sommaire

Photo



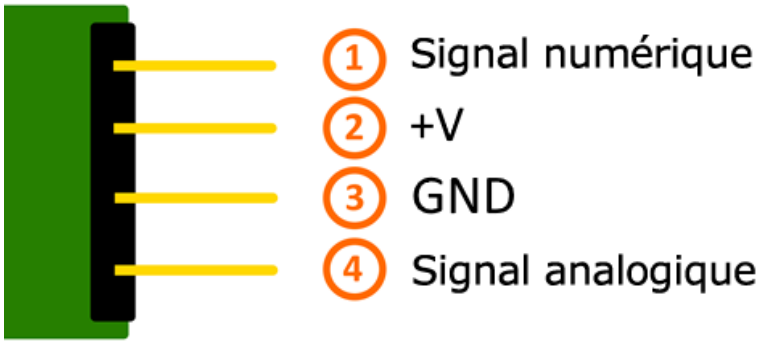
Données techniques / Description sommaire

La sortie varie en présence d'une flamme (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

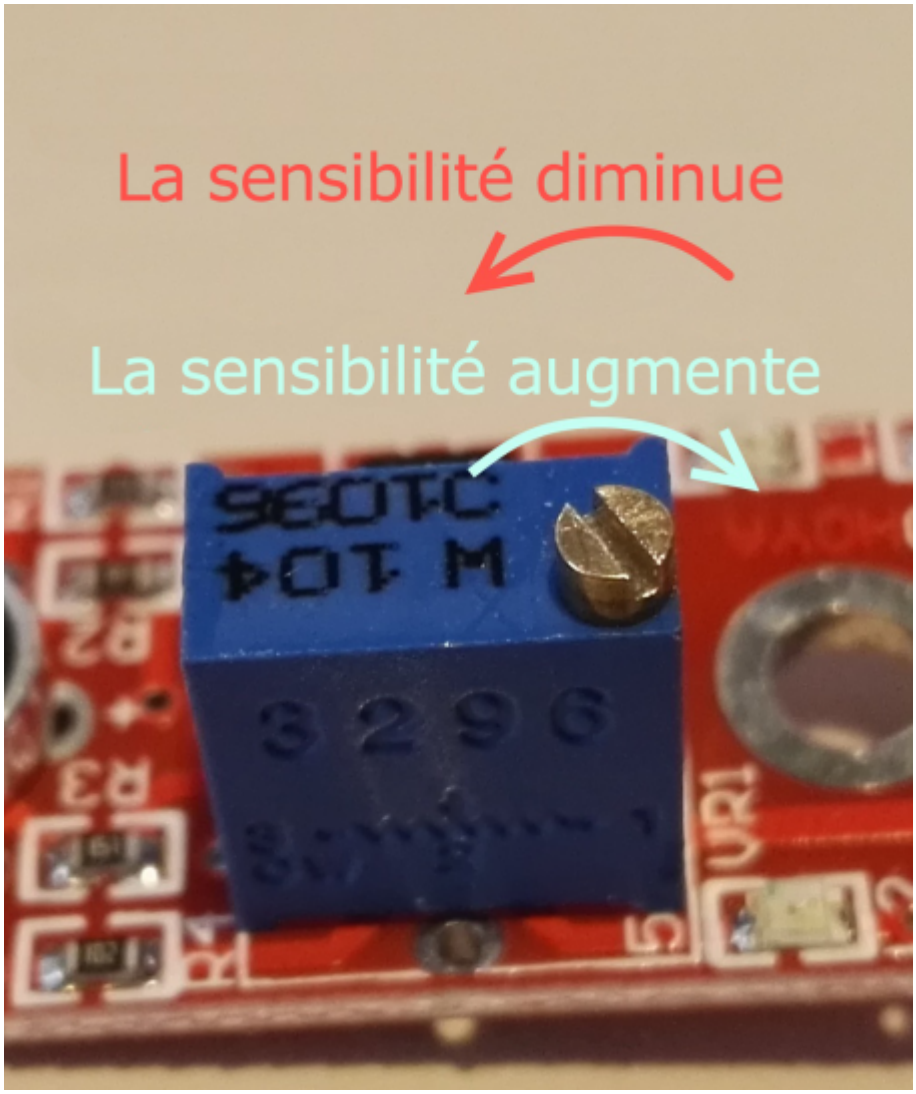


Fonctionnement du capteur

Ce module est composé de trois éléments fonctionnels. Le capteur situé à l'avant du module effectue la mesure, le signal analogique est ensuite envoyé sur l'amplificateur. Celui-ci amplifie le signal en fonction du gain déterminé par le potentiomètre et envoie le signal à la sortie analogique du module.

Il convient de noter que le signal est inversé: plus la valeur mesurée par le capteur est haute, plus la tension de sortie est faible.

La troisième partie est composée d'un comparateur qui commute la sortie numérique et la diode lorsque le signal tombe en dessous d'une certaine valeur. La sensibilité peut être ajustée au moyen du potentiomètre comme décrit ci-dessous:



Ce type de capteur ne délivre pas des valeurs absolues (par exemple, la température mesurée avec précision en ° C ou de la force du champ magnétique en mT), mais des valeurs relatives. On définit une valeur limite par rapport à une valeur normale donnée et le module émet un signal si cette limite est dépassée.

Ce fonctionnement est idéal pour la surveillance de la température (KY-028), les détecteurs de proximité (KY-024, KY 025, KY-036), la surveillance des alarmes (KY-037, KY-038) ou le détecteur de flamme (KY-026).

Exemple de code pour Arduino

Le programme lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état dela sortie numérique est également indiqué dans la console, ce qui permet de savoir su le seuil a été atteint ou pas.

```
// Déclaration et initialisation des broches d'entrées
int Analog_Eingang = A0; // Signal analogique
int Digital_Eingang = 3; // Signal numérique
```



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

  Serial.begin (9600); // Sortie série à 9600 bauds
}

// Le programme lit les valeurs des broches d'entrée et les envoie à la sortie série
void loop ()
{
  float Analog;
  int Digital;

  //Les valeurs sont lues, sont converties en tension...
  Analog = analogRead (Analog_Eingang) * (5.0 / 1023.0);
  Digital = digitalRead (Digital_Eingang);

  //... et envoyées à la sortie série.
  Serial.print ("Tension analogique:"); Serial.print (Analog, 4);  Serial.print ("V, ");
  Serial.print ("Limite:");

  if(Digital==1)
  {
      Serial.println (" atteinte");
  }
  else
  {
      Serial.println (" pas encore atteinte");
  }
  Serial.println ("-----------------------------------------------------------------");
  delay (200);
}
```

Affectation des broches Arduino:

Signal numérique	= [Pin 3]
+V	= [Pin 5V]
GND	= [Pin GND]
Signal analogique	= [Pin A0]

Exemple de programme à télécharger

KY-026.zip

Exemple de code pour Raspberry Pi

!! Attention !! Capteur analogique !! Attention !!

Contrairement à une carte Arduino, la Raspberry Pi ne dispose pas d'entrées analogiques ni de convertisseur ADC (Analog Digital Converter) intégré. Cela pose problème lorsque vous voulez utiliser des capteurs analogiques avec une carte Raspberry.

Pour contourner ce problème, Sensorkit X40 inclut le module KY-053 qui possède un module convertisseur ADC de 16 bits et qui peut être raccordé sur la Raspberry pour lui procurer 4 entrées analogiques. Ce module se raccorde à la Raspberry via le bus I2C. Il mesure la tension (analogique) et envoie une valeur numérique à la Raspberry.

Vous trouverez de plus amples informations à ce sujet dans la description du module KY-053.

!!Attention !! Capteur analogique !! Attention !!

Ce programme utilise des librairies Python de la société Adafruit pour piloter les circuits ADS1115 (ADC) et ADS1x15 (I2C).

Celles-ci se trouvent à la page https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code sous licence BSD.

Le programme mesure la tension à l'aide du convertisseur ADS1115. Il lit la valeur de la tension à la sortie analogique et l'envoie vers le port série.

L'état de la sortie numérique est également indiqué dans la console, ce qui permet de savoir su le seuil a été atteint ou pas.

```
#!/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
### Programme traduit par Go tronic
###
### Analog Sensor - Raspberry Pi Python Code Example
###
##########################################################################################

# Ce code utilise les librairies Python ADS1115 et I2C pour la Raspberry Pi
# Ces librairies sont publiées sous licence BSD sur le lien ci-dessous
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Les modules nécessaires sont importés et mis en place
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Les variables utilisées sont initialisées
delayTime = 0.2

# attribution d'adresse ADS1x15 ADC

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

# Choix du gain
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# Choix de la fréquence d'échantillonnage ADC (SampleRate)
# sps = 8    # 8 échantillons par seconde
# sps = 16   # 16 échantillons par seconde
# sps = 32   # 32 échantillons par seconde
sps = 64   # 64 échantillons par seconde
# sps = 128  # 128 échantillons par seconde
# sps = 250  # 250 échantillons par seconde
# sps = 475  # 475 échantillons par seconde
```



```
# sps = 860  # 860 échantillons par seconde

# choix du canal ADC (1-4)
adc_channel = 0    # Channel 0
# adc_channel = 1    # Channel 1
# adc_channel = 2    # Channel 2
# adc_channel = 3    # Channel 3

# initialisation du convertisseur
adc = ADS1x15(ic=ADS1115)

# Sélection de la broche d'entrée du signal numérique
Digital_PIN = 24
GPIO.setup(Digital_PIN, GPIO.IN, pull_up_down = GPIO.PUD_OFF)

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

# ########
# boucle de programme principale
# ########
# Le programme lit les tensions en entrées et les transmet à la console.

try:
        while True:
                #Les valeurs de tension sont enregistrées
                analog = adc.readADCSingleEnded(adc_channel, gain, sps)

                # Envoi vers la console
                if GPIO.input(Digital_PIN) == False:
                        print "Tension analogique:", analog,"mV, ","Limite: pas encore att
                else:
                        print "Tension analogique:", analog, "mV, ", "Limite: atteinte"
                print "---------------------------------------"

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



except KeyboardInterrupt:
        GPIO.cleanup()
```

Brochage Raspberry Pi:

Capteur

Signal numérique	= GPIO 24	[Pin 18 (RPi)]
+V	= 3,3V	[Pin 1 (RPi)]
GND	= Masse	[Pin 06 (RPi)]
Signal analogique	= 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]

Exemple de programme à télécharger

RPi_AnalogSensor.zip

Commande pour lancer le programme:

```
sudo python RPi_AnalogSensor.py
```

