KY-038 Mikrofon Sound Sensor Modul

Inhaltsverzeichnis

1 Bild 1
2 Technische Daten / Kurzbeschreibung 1
3 Pin-Belegung 2
4 Funktionsweise des Sensors 2
5 Codebeispiel Arduino 3
6 Codebeispiel Raspberry Pi 4

Bild

Technische Daten / Kurzbeschreibung

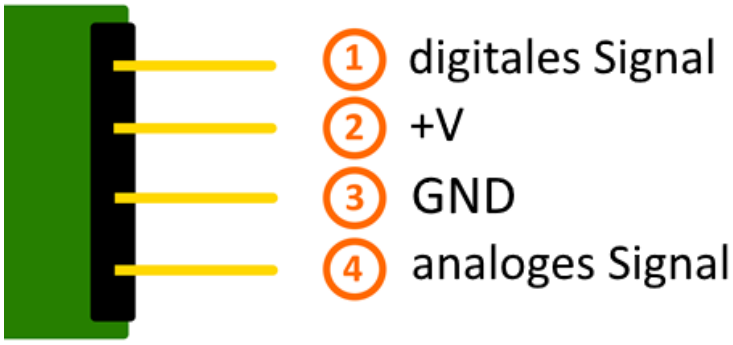
Digitaler Ausgang: Über das Potentiometer, kann ein Grenzwert für den empfangenen Schall eingestellt werden, bei dem der digitale Ausgang schalten soll.

Analoger Ausgang: Direktes Mikrofon-Signal als Spannungspegel

LED1: Zeigt an, dass der Sensor mit Spannung versorgt ist

LED2: Zeigt an, dass ein Geräusch detektiert wurde

Pin-Belegung

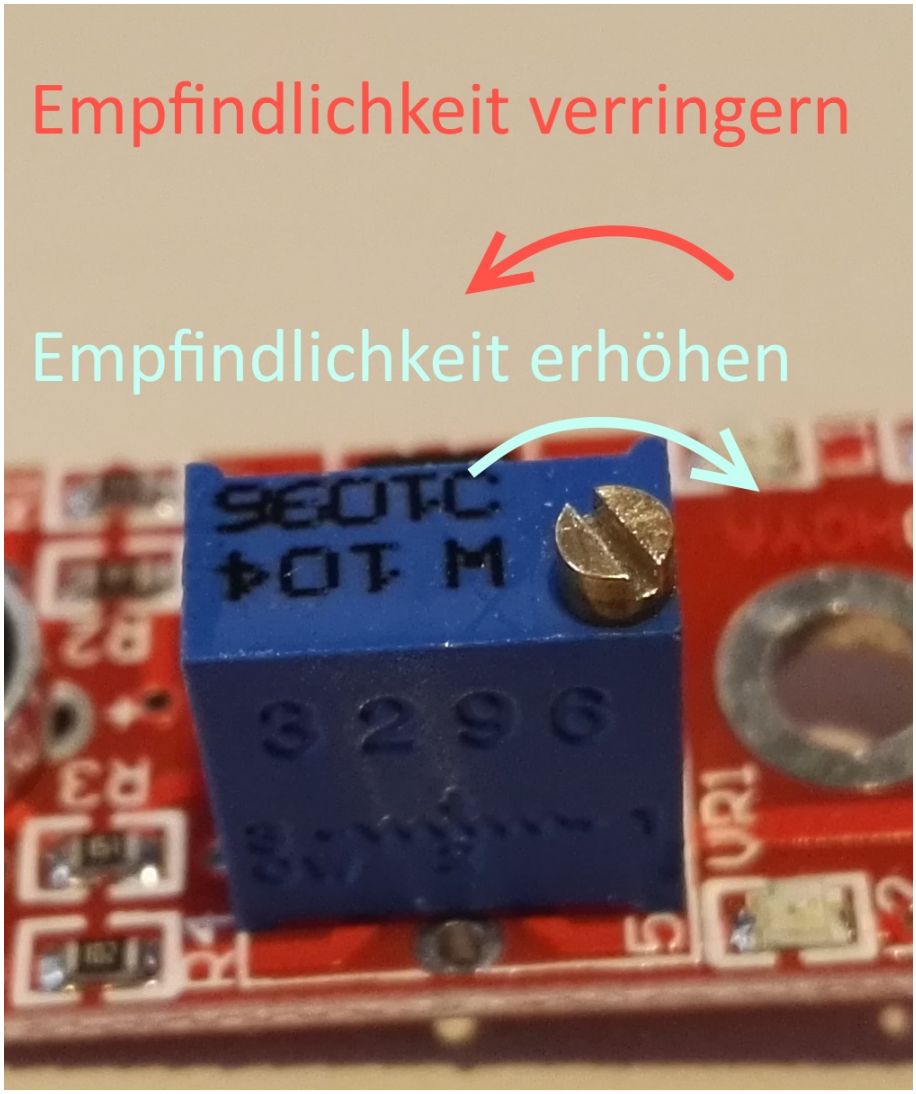


Funktionsweise des Sensors

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_CircuitPython_ADS1x15] unter der MIT-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 [V] 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
###
### KY-053 Analog Digital Converter - Raspberry Pi Python Code Example
###
##########################################################################################
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)
chan1 = AnalogIn(ads, ADS.P1)
chan2 = AnalogIn(ads, ADS.P2)
chan3 = AnalogIn(ads, ADS.P3)
```



```
delayTime = 1
Digital_PIN = 24

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

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

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

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

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 python3 RPi_AnalogSensor.py
```

