KY-053 Analog Digital Converter

Inhaltsverzeichnis

Bild

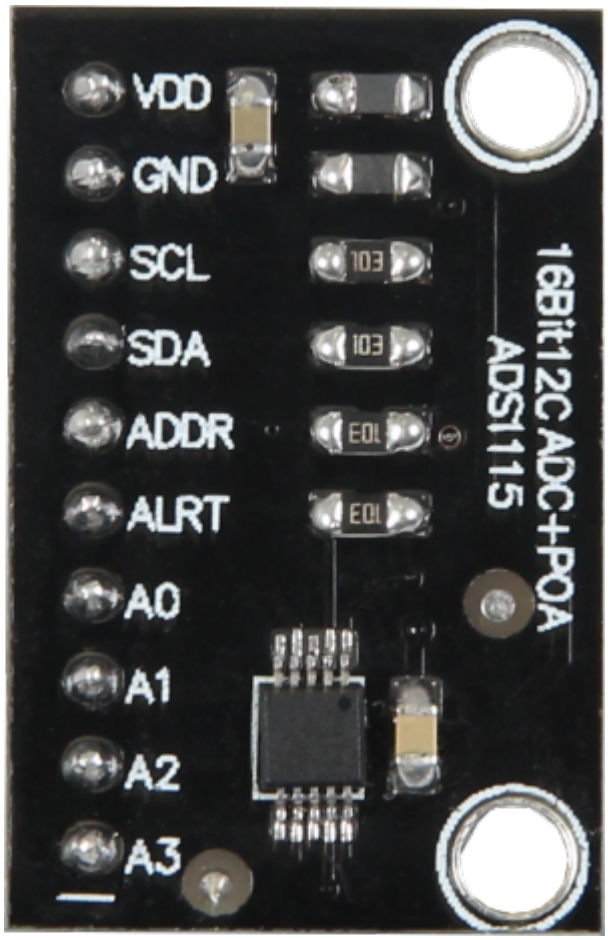
Technische Daten / Kurzbeschreibung

Per entsprechenden Befehlen auf den I2C-Bus, können auf bis zu 4 Eingängen analoge Spannungswerte mit bis zu 16-Bit Genauigkeit gemessen werden. Das Messergebnis wird kodiert auf den I2C-Bus ausgegeben.

Für dieses Modul wird eine entsprechende Software benötigt - siehe Codebeispiele unten.

Pin-Belegung

Die Pin-Belegung ist auf der Modulplatine aufgedruckt



Codebeispiel Arduino

Die Arduino-Boards besitzen von Haus aus einen 10 Bit-ADC mit 6 Kanälen. Benötigt man jedoch mehr Kanäle oder eine höhere Genauigkeit, dann kann man den Arduino mittels des KY-053 Analog Digital Converter Moduls um 4 ADC Kanäle mit 12-Bit Genauigkeit erweitern, welches per I2C an den Arduino angeschlossen wird.

Zur Ansteuerung dieses Moduls gibt es mehrere Möglichkeiten - als besonders zugänglich haben sich die ADS1X15 Libraries erwiesen, die die Firma Adafruit unter [https://github.com/adafruit/Adafruit_ADS1X15] unter der [BSD-Lizenz] veröffentlicht hat.

Das unten stehende Beispiel verwendet diese besagte Library - hierzu empfehlen wir diese von Github herunterzuladen, zu entpacken und im Arduino-Library-Ordner, welcher sich standardmäßig unter (C:\Benutzer\[Benutzername]\Dokumente\Arduino\libraries) befindet, zu kopieren, damit diese für dieses Codebeispiel und folgende Projekte zur Verfügung steht. Alternativ ist diese auch im unten stehenden Download Paket ebenfalls enthalten.


```
#include
#include

// ADS1115 Modul wird initialisiert - alle folgenden Operationen mit dem ADC
// können mit Hilfe des Objektes "ads" ausgeführt werden.
Adafruit_ADS1115 ads;

void setup(void)
{
  Serial.begin(9600);

  Serial.println("Werte der analogen Eingaenge des ADS1115 (A0..A3) werden ausgelesen und
  Serial.println("ADC Range: +/- 6.144V (1 bit = 0.1875mV)");

  // Dieses Modul besitzt an seinen analogen Eingängen Signalverstärker, deren
  // Verstärkung per Software in den unten stehenden Bereichen konfiguriert werden
  // können.
  // Dieses ist in dem Fall gewünscht, wenn ein bestimmter Spannungsbereich
  // als messergebnis erwartet wird und man so eine höhere Auflösung des Signals
  // erhält.
  // Als Standardverstärkung ist Gain=[2/3] gewählt und kann durch Auskommentieren
  // auf eine andere Verstärkung umgestellt werden.
  //                                                                ADS1115
  //                                                                -------
  ads.setGain(GAIN_TWOTHIRDS);  // 2/3x gain +/- 6.144V  1 bit = 0.1875mV
  // ads.setGain(GAIN_ONE);        // 1x gain   +/- 4.096V  1 bit = 0.125mV
  // ads.setGain(GAIN_TWO);        // 2x gain   +/- 2.048V  1 bit = 0.0625mV
  // ads.setGain(GAIN_FOUR);       // 4x gain   +/- 1.024V  1 bit = 0.03125mV
  // ads.setGain(GAIN_EIGHT);      // 8x gain   +/- 0.512V  1 bit = 0.015625mV
  // ads.setGain(GAIN_SIXTEEN);    // 16x gain  +/- 0.256V  1 bit = 0.0078125mV

  ads.begin();
}

void loop(void)
{
  uint16_t adc0, adc1, adc2, adc3;
  float voltage0, voltage1, voltage2, voltage3;
  float gain_conversion_factor;

  // Der Befehl "ads.readADC_SingleEnded(0)" ist die eigentliche Operation, die die Messun
  // die "0" als Variable für diese Funktion definiert den verwendeten Channel, der gemess
  // Soll z.B. der dritte Channel gemessen werden, so muss diese mit der "3" ausgetauscht
  adc0 = ads.readADC_SingleEnded(0);
  adc1 = ads.readADC_SingleEnded(1);
  adc2 = ads.readADC_SingleEnded(2);
  adc3 = ads.readADC_SingleEnded(3);

  // Dieser Wert wird für die Umrechnung in eine Spannung benötigt - dieser ´hängt von der
  // Der passende Wert zur Verstärkung sollte aus der oben aufgezeigten Tabelle entnommen
  gain_conversion_factor= 0.1875;

  // Umrechnung der aufgezeichneten Werte in eine Spannung
  voltage0 = (adc0 * gain_conversion_factor);
  voltage1 = (adc1 * gain_conversion_factor);
  voltage2 = (adc2 * gain_conversion_factor);
  voltage3 = (adc3 * gain_conversion_factor);

  // Ausgabe der Werte auf die serielle Schnittstelle
  Serial.print("Analog Eingang 0: "); Serial.print(voltage0);Serial.println("mV");
  Serial.print("Analog Eingang 1: "); Serial.print(voltage1);Serial.println("mV");
  Serial.print("Analog Eingang 2: "); Serial.print(voltage2);Serial.println("mV");
```



```
  Serial.print("Analog Eingang 3: "); Serial.print(voltage3);Serial.println("mV");
  Serial.println("-----------------------");

  delay(1000);
}
```

Beispielprogramm Download:

KY-053-AnalogDigitalConverter.zip

Anschlussbelegung Arduino:

VDD = [Pin 5V]
GND = [Pin GND]
SCL = [Pin SCL]
SDA = [Pin SDA]
ADDR = [N.C.]
ALRT = [N.C.]
A0 = [Messspitze Analog 0]
A1 = [Messspitze Analog 1]
A2 = [Messspitze Analog 2]
A3 = [Messspitze Analog 3]

Codebeispiel Raspberry Pi

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.

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 OpenSource-Lizenz veröffentlicht. Die benötigten Libraries sind im unteren Download-Paket enthalten.

Diese Bibliothek können Sie mit den folgenden Befehl installieren:

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

Das Programm liest die aktuelle Spannung aus, die an den 4 Kanälen des ADS1115 ADC anliegen, und zeigt diese in der Konsole an. Über die Variable "delayTime", lässt sich die Pause zwischen den Messungen einstellen.

Damit der Raspberry Pi mit dem Sensor auf dem I2C-Bus kommunizieren kann, muss vorab die I2C-Funktion beim Raspberry Pi aktiviert werden. Hierzu müssen folgende Zeilen am Ende der Datei "/boot/config.txt" hinzugefügt werden:

```
dtparam=i2c_arm=on
```

Die Datei kann mit folgenden Befehl editiert werden:

```
sudo nano /boot/config.txt
```

Führen Sie nach dem abspeichern einen Neustart des System aus, damit I²C richtig aktiviert wird:

```
sudo reboot
```

Mit der Tastenfolge [Strg+X -> Y -> Enter] kann die Datei, nach dem hinzufügen der Zeile am unteren Ende, gespeichert und geschlossen werden.

Außerdem werden zusätzliche Bibliotheken benötigt, um I2C innerhalb Python nutzen zu können. Um diese zu installieren muss folgender Befehl in die Konsole eingegeben werden:

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

Hiernach kann das folgende Python-Code Beispiel verwendet werden:

Erstellen Sie dazu eine neue Datei und kopieren Sie dann das Beispiel in die Datei:

```
sudo nano ADS1115.py
```

```
#!/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

# 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)



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)
```

Die Bibliothek bringt auch ein Code-Beispiel mit. Dieses können Sie auf der Webseite einsehen:
https://github.com/adafruit/Adafruit_CircuitPython_ADS1x15

Das Python-Skript muss mit Python 3 ausgeführt werden.

z.B.:

```
sudo python3 ADS1115.py
```

Anschlussbelegung Raspberry Pi:

VDD	=	3,3V	[Pin 01]
GND	=	Masse	[Pin 06]
SCL	=	GPIO03 / SCL	[Pin 05]
SDA	=	GPIO02 / SDA	[Pin 03]
ADDR	=	N.C.	[-]
ALRT	=	N.C.	[-]
A0	=	Messspitze Analog 0	[Zu messende Spannung | z.B. Sensorausgang]
A1	=	Messspitze Analog 1	[Zu messende Spannung | z.B. Sensorausgang]
A2	=	Messspitze Analog 2	[Zu messende Spannung | z.B. Sensorausgang]
A3	=	Messspitze Analog 3	[Zu messende Spannung | z.B. Sensorausgang]

Beispielprogramm Download

KY-053_RPi_AnalogDigitalConverter.zip

Zu starten mit dem Befehl:

```
sudo python3 KY-053_RPi_AnalogDigitalConverter.py
```


Erweiterte Funktionen des ADS1115 ADC

Die Funktion des ADS1115, die in den oben aufgezeigten Codebeispielen zur Verwendung kommt, nennt sich "Single Ended Conversion" und besagt, dass eine Messung am einzelnen ausgewählten Kanal gegen Masse erfolgt.

Neben dieser Art von Messung besitzt der ADS1115 ADC auch z.B. die Funktion der differentiellen Messung, sodass eine Differenz-Spannung zwischen zwei Eingängen gemessen wird (Beispiel: Spannung zwischen A0 und A1). Zusätzlich zur Singe-Ended Messung lässt sich auch die Comparator Funktion aktivieren, welche erst ein Messergebnis liefert, wenn eine Spannungsschwelle überschritten wird.

Diese Funktionen und Funktionen, wie z.B. die Änderung der Abstastrate (Samplerate), sind in den Adafruit Libraries zur Konfiguration einprogrammiert - Näheres finden Sie in der Dokumentation der Adafruit Libraries.

