KY-053 Convertisseur analogique digital

Sommaire

Photo

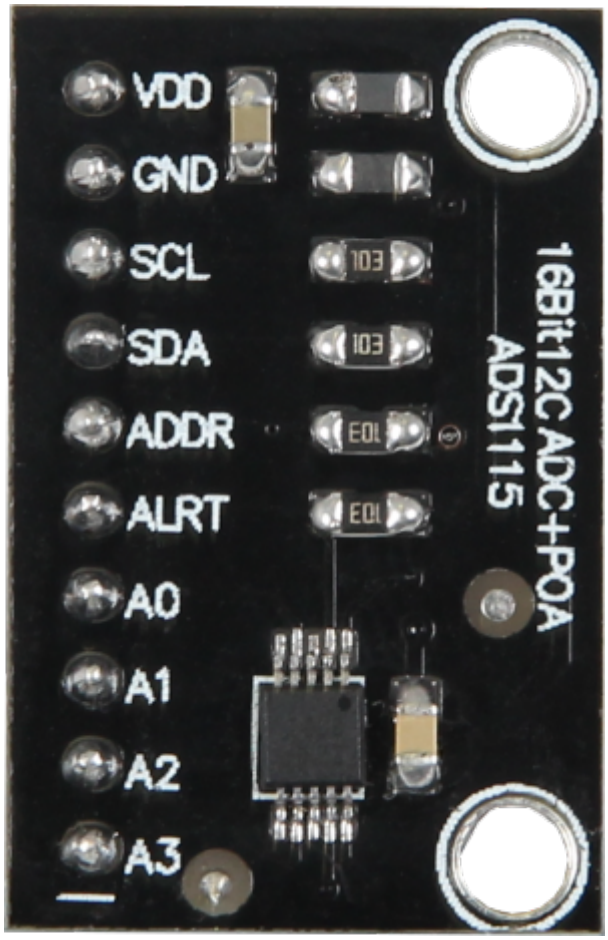
Données techniques / Description sommaire

Ce module à 4 canaux permet de mesurer des tensions analogiques avec une précision de 16 bits. Il est nécessaire d'utiliser des commandes I2C pour le faire fonctionner. Le résultat est sortie codé sur le bus I2C.

Un logiciel est nécessaire pour utiliser ce module, voir les exemples de code ci-dessous.

Brochage

L'affectation des broches est imprimée sur la carte du module.



Exemple de code pour Arduino

Les cartes Arduino sont équipées d'origine de 6 entrées analogiques 10 bits. L'utilisation du module KY-053 peut s'avérer utile si vous avez besoin d'une précision plus grande (commande via I2C).

Il existe plusieurs façons de contrôler ce module. La société Adafruit a publié des librairies du circuit ADS1X15 à la page [https://github.com/adafruit/Adafruit_ADS1X15] sous licence [BSD-Lizenz] veröffentlicht hat.

L'exemple de code ci-dessous utilise cette librairie. Nous recommandons le téléchargement de Github et l'installation après décompression dans le dossier C:\User\[Username]\Dcuments\Arduino\libraries. Cette librairie est également disponible dans le lien à télécharger ci-dessous.

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>

// Initialisation du module ADS1115. Toutes les opérations avec
// l'ADC peuvent être réalisées à l'aide de l'objet ads
Adafruit_ADS1115 ads;

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

  Serial.println("Lecture des tensions aux bornes A0 a A3 du circuit ADS1115 (A0..A3) en e
  Serial.println("Plage ADC: +/- 6.144V (1 bit = 0.1875mV)");
```



```
  // Ce module présente un amplificateur de signal à ses entrées analogiques
  // dont le gain peut être configuré comme décrit ci-dessous.
  // Ceci est utilie lorque par exemple on s'attend à obtenir un résultat dans une
  // certaine plage de mesure mais que le résultat est plus grand que prévu.
  // Le gain par défaut est Gain=[2/3] et peut être modifié simplement en enlevant les //
  //                                                                 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;

  // La commande "ads.readADC_SingleEnded(0)" est la fonction principale quifait démarrer
  // Le "0" en tant que variable pour cette fonction définit le canal à mesurer.
  // Si on voulait mesurer sur le troisième canal, on aurait chané le "0" en "2"
  adc0 = ads.readADC_SingleEnded(0);
  adc1 = ads.readADC_SingleEnded(1);
  adc2 = ads.readADC_SingleEnded(2);
  adc3 = ads.readADC_SingleEnded(3);

  // Cette valeur est nécessaire pour la conversion en une tension.
  // La valeur appropriée pour le gain doit être reprise dans la table ci-dessus
  gain_conversion_factor= 0.1875;

  // Conversion des valeurs enregistrées en tension
  voltage0 = (adc0 * gain_conversion_factor);
  voltage1 = (adc1 * gain_conversion_factor);
  voltage2 = (adc2 * gain_conversion_factor);
  voltage3 = (adc3 * gain_conversion_factor);

  // Envoi des valeurs vers l'interface série
  Serial.print("Entree analogique 0: "); Serial.print(voltage0);Serial.println("mV");
  Serial.print("Entree analogique 1: "); Serial.print(voltage1);Serial.println("mV");
  Serial.print("Entree analogique 2: "); Serial.print(voltage2);Serial.println("mV");
  Serial.print("Entree analogique 3: "); Serial.print(voltage3);Serial.println("mV");
  Serial.println("------------------------");

  delay(1000);
}
```

Exemple de programme à télécharger:

KY-053.zip

Affectation des broches Arduino:

VDD	=	[Pin 5V]
GND	=	[Pin GND]
SCL	=	[Pin SCL]
SDA	=	[Pin SDA]

ADDR = [N.C.]
ALRT = [N.C.]
A0 = [Analog 0]
A1 = [Analog 1]
A2 = [Analog 2]
A3 = [Analog 3]

Exemple de code pour Raspberry Pi

Contrairement à une carte Arduino, le Raspberry Pi ne dispose pas d'entrées analogiques. Cela limite l'utilisation de cette carte à des capteurs numériques uniquement. On ne peut pas utiliser un potentiomètre en entrée, par exemple.

Pour palier à ce problème, notre kit de capteurs est livré avec le module KY-053 qui dispose de 4 entrées analogiques avec une précision de 16 bits. Ce module se raccorde à la carte Raspberry Pi via le bus I2C.

Il existe plusieurs façons de contrôler ce module. La société Adafruit a publié des librairies du circuit ADS1X15 à la page [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] sous licence MIT OpenSource. Cette librairie est également disponible dans le lien à télécharger ci-dessous.

Le programme lit les tensions sur les 4 canaux du circuit ADS1115 et les affiche dans la console. La variable "delayTime"permet d'instaurer une pause entre les mesures.

Pour que la Raspberry Pi puisse communiquer avec le KY-053 via le bus I2C, celui-ci doit être activé au préalable. Pour ce faire, il faut ajouter la ligne ci-dessous à la fin du fichier "/boot/config.txt":

```
dtparam=i2c_arm=on
```

Le fichier peut être modifié avec la commande suivante:

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

En utilisant la séquence de touches [Ctrl + X -> Y -> Entrée], le fichier sera enregistré et fermé après avoir ajouté la ligne de commande.

Il est également nécessaire d'utiliser la librairies supplémentaires pour utiliser I2C en Python. Utilisez la commande suivante dans la console pour l'installer:

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

Vous pouvez ensuite utiliser l'exemple de code Python ci-dessous:

```
#!/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
###
### KY-053 Analog Digital Converter - 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 = 0.5 # in Sekunden

# 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 = 0    # Channel 0
adc_channel_1 = 1    # Channel 1
adc_channel_2 = 2    # Channel 2
adc_channel_3 = 3    # Channel 3

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

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

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

# ########
# 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
                adc0 = adc.readADCSingleEnded(adc_channel_0, gain, sps)
                adc1 = adc.readADCSingleEnded(adc_channel_1, gain, sps)
                adc2 = adc.readADCSingleEnded(adc_channel_2, gain, sps)
                adc3 = adc.readADCSingleEnded(adc_channel_3, gain, sps)

                # Envoi vers la console
                print "Lecture Channel 0:", adc0, "mV "
                print "Lecture Channel 1:", adc1, "mV "
                print "Lecture Channel 2:", adc2, "mV "
                print "Lecture Channel 3:", adc3, "mV "
                print "----------------------------------------"

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



except KeyboardInterrupt:
        GPIO.cleanup()
```

Brochage 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	=	Analog 0	[pour mesure de tension (capteur par ex.)]
A1	=	Analog 1	[pour mesure de tension (capteur par ex.)]
A2	=	Analog 2	[pour mesure de tension (capteur par ex.)]
A3	=	Analog 3	[pour mesure de tension (capteur par ex.)]

Exemple de programme à télécharger

KY-053_RPi_AnalogDigitalConverter.zip

Commande pour lancer le programme:

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

Note : les fichier "Adafruit_ADS1x15.py" et "Adafruit_I2C.py" doivent être placés dans le même dossier que le programme pour qu'il puisse fonctionner.

Fonctions avancées du ADS1115

Le module KY-053 étant basé sur le convertisseur ADS1115, il est possible de réaliser jusqu'à 4 mesures "Single-ended" (mesure des tensions aux entrées indépendamment les unes des autres) ou jusqu'à 2 mesures différentielles (mesure de la différence de tension entre deux entrées).

Lors d'une mesure "Single-ended", la borne est connectée à la masse.

Pour plus d'informations sur les caractéristiques et les fonction, merci de consulter les bibliothèques Adafruit.

