SEN-KY039HS
Herzschlagsensor

INHALTSVERZEICHNIS

1. Einführung

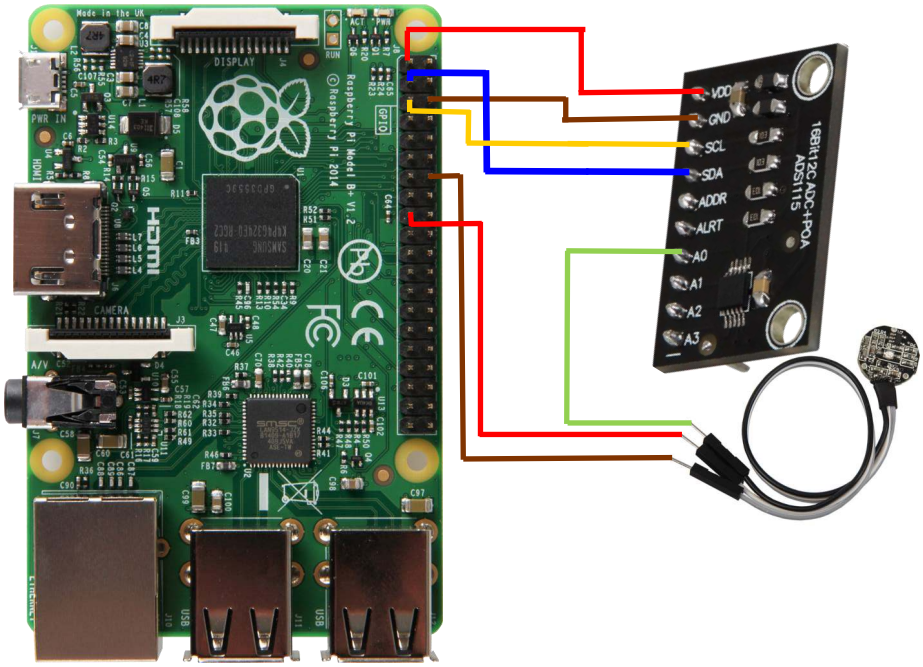
Sehr geehrter Kunde,
vielen Dank, dass sie sich für unser Produkt entschieden haben.
Im folgenden zeigen wir Ihnen, was bei der Inbetriebnahme und der Verwendung zu beachten ist.

Sollten Sie während der Verwendung unerwartet auf Probleme stoßen, so können Sie uns selbstverständlich gerne kontaktieren.

2. Verwendung Raspberry Pi

2.1 Connection

Da der Herzschlagsensor ein analoger Sensor ist und der Raspberry Pi keine analogen Eingänge hat, müssen Sie einen Analog-Digital-Wandler verwenden. Zum Beispiel unseren ADC COM-KY053ADC.



Raspberry Pi	KY039HS	ADC
3v3 (Pin1)		VDD
GND (Pin 6)		GND
SCL (Pin 5)		SCL
SDA (Pin 3)		SDA
	Schwarz	A0
GND (Pin 14)	Grau	
3v3 (Pin 17)	Weiß	

2.2 Installation

Zunächst müssen Sie die Bibliothek für den Analog-Digital-Wandler installieren.
Dies wurde unter dem folgenden Link unter der MIT-Lizenz veröffentlicht.

Zum installieren der Bibliothek geben sie einfach folgenden Befehl in die Konsole ein:

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

Zusätzlich müssen Sie I2C auf Ihrem Raspberry Pi aktivieren, geben Sie dazu folgenden Befehl ein:

```
sudo raspi-config
```

Gehen Sie nun auf **Interfacing Options** -> und aktivieren Sie **I2C.**

2.3 Programmbeispiel

```
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)

if __name__ == '__main__':

    # initialization
    GAIN = 2/3
    curState = 0
    thresh = 525  # mid point in the waveform
    P = 512
    T = 512
    stateChanged = 0
    sampleCounter = 0
    lastBeatTime = 0
    firstBeat = True
    secondBeat = False
    Pulse = False
```

2.3 Programmbeispiel

```
IBI = 600
rate = [0]*10
amp = 100


lastTime = int(time.time()*1000)

# Main loop. use Ctrl-c to stop the code
while True:
    # read from the ADC
    Signal = chan0.value   #TODO: Select the correct ADC channel
    curTime = int(time.time()*1000)

    sampleCounter += curTime - lastTime; # keep track of time in mS with this variable
    lastTime = curTime
    N = sampleCounter - lastBeatTime; # monitor the time since last beat to avoid noise


    ##  find the peak
    if Signal < thresh and N > (IBI/5.0)*3.0 :  # wait 3/5 of last IBI
        if Signal < T :                         # T is the trough
          T = Signal;                         # keep track of lowest point in pulse wave

    if Signal > thresh and  Signal > P:           # thresh condition helps avoid noise
        P = Signal;                             # P is the peak
                                              # keep track of highest point in pulse wave


      # signal surges up in value every time there is a pulse
    if N > 250 :                                   # avoid high frequency noise
        if  (Signal > thresh) and  (Pulse == False) and  (N > (IBI/5.0)*3.0)  :
          Pulse = True;               # set the Pulse flag when we think there is a pulse
          IBI = sampleCounter - lastBeatTime;         # measure time between beats in mS
          lastBeatTime = sampleCounter;               # keep track of time for next pulse

          if secondBeat :            # if this is the second beat, if secondBeat == TRUE
            secondBeat = False;                  # clear secondBeat flag
            for i in range(0,10):
              rate[i] = IBI;

          if firstBeat :
            firstBeat = False;
            secondBeat = True;
            continue


          # keep a running total of the last 10 IBI values
          runningTotal = 0;                  # clear the runningTotal variable

          for i in range(0,9):               # shift data in the rate array
            rate[i] = rate[i+1];
            runningTotal += rate[i];
```

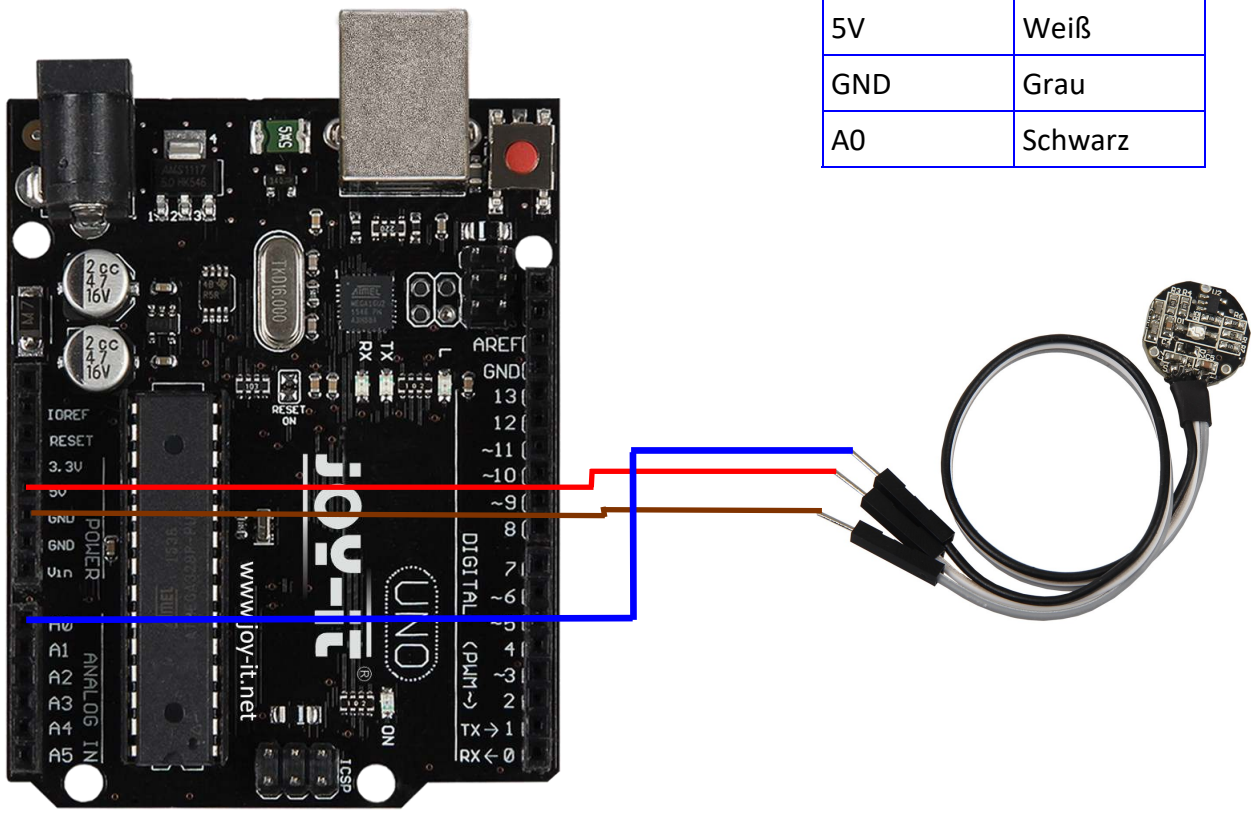
2.3 Programmbeispiel

```
            rate[9] = IBI;
            runningTotal += rate[9];
            runningTotal /= 10;
            BPM = 60000/runningTotal;
            print ('BPM: {}'.format(BPM))

        if Signal < thresh and Pulse == True :   # when the values are going down, the beat
is over
            Pulse = False;
            amp = P - T;
            thresh = amp/2 + T;
            P = thresh;
            T = thresh;

        if N > 2500 :                           # if 2.5 seconds go by without a beat
            thresh = 512;                           # set thresh default
            P = 512;                                # set P default
            T = 512;                                # set T default
            lastBeatTime = sampleCounter;           # bring the lastBeatTime up to date
            firstBeat = True;                       # set these to avoid noise
            secondBeat = False;                     # when we get the heartbeat back
            print ("no beats found")

        time.sleep(0.005)
```

3. Verwendung mit dem Arduino

3.1 Anschluss

Arduino	KY039HS
5V	Weiß
GND	Grau
A0	Schwarz



3.2 Programmbeispiel

```
////////////////////////////////////////////////////////////////////
/// Copyright (c)2015 Dan Truong
/// Permission is granted to use this software under the MIT
/// licence, with my name and copyright kept in source code
/// http://http://opensource.org/licenses/MIT
///
/// KY039 Arduino Heartrate Monitor V1.0 (April 02, 2015)
////////////////////////////////////////////////////////////////////

// German Comments by Joy-IT

int rawValue;


bool heartbeatDetected(int IRSensorPin, int delay)
{
  static int maxValue = 0;
  static bool isPeak = false;


  bool result = false;

  rawValue = analogRead(IRSensorPin);
  // Hier wird der aktuelle Spannungswert am Fototransistor ausgelesen und in der rawValue-
Variable zwischengespeichert
  rawValue *= (1000/delay);

  // Sollte der aktuelle Wert vom letzten maximalen Wert zu weit abweichen
  // (z.B. da der Finger neu aufgesetzt oder weggenommen wurde)
  // So wird der MaxValue resetiert, um eine neue Basis zu erhalten.
  if (rawValue * 4L < maxValue) {    maxValue = rawValue * 0.8;  }     // Detect new peak
  if (rawValue > maxValue - (1000/delay)) {
    // Hier wird der eigentliche Peak detektiert. Sollte ein neuer RawValue groeßer sein
    // als der letzte maximale Wert, so wird das als Spitze der aufgezeichnten Daten er-
kannt.
    if (rawValue > maxValue) {
      maxValue = rawValue;
    }
    // Zum erkannten Peak soll nur ein Herzschlag zugewiesen werden
    if (isPeak == false) {
      result = true;
    }
    isPeak = true;
  } else if (rawValue < maxValue - (3000/delay)) {
    isPeak = false;
    // Hierbei wird der maximale Wert bei jeden Durchlauf
    // etwas wieder herabgesetzt. Dies hat den Grund, dass
    // nicht nur der Wert sonst immer stabil bei jedem Schlag
    // gleich oder kleiner werden wuerde, sondern auch,
    // falls der Finger sich minimal bewegen sollte und somit
    // das Signal generell schwaecher werden wuerde.
    maxValue-=(1000/delay);
 }
  return result;
}
```

```
int ledPin=13;
int analogPin=0;

void setup()
{
  // Die eingebaute Arduino LED (Digital 13), wird hier zur Ausgabe genutzt
  pinMode(ledPin,OUTPUT);

  // Serielle Ausgabe Initialisierung
  Serial.begin(9600);
  Serial.println("Heartbeat Detektion Beispielcode.");
}

const int delayMsec = 60; // 100msec per sample

// Das Hauptprogramm hat zwei Aufgaben:
// - Wird ein Herzschlag erkannt, so blinkt die LED kurz aufgesetzt
// - Der Puls wird errechnet und auf der serriellen Ausgabe ausgegeben.

void loop()
{
  static int beatMsec = 0;
  int heartRateBPM = 0;
  if (heartbeatDetected(analogPin, delayMsec)) {
    heartRateBPM = 60000 / beatMsec;
    // LED-Ausgabe bei Herzschlag
    digitalWrite(ledPin,1);

    // Serielle Datenausgabe
    Serial.print("Puls erkannt: ");
    Serial.println(heartRateBPM);

    beatMsec = 0;
  } else {
    digitalWrite(ledPin,0);
  }
  delay(delayMsec);
  beatMsec += delayMsec;
}
```

4. Sonstige Informationen

Unsere Informations- und Rücknahmepflichten nach dem Elektrogesetz (ElektroG)

Symbol auf Elektro- und Elektronikgeräten:

Diese durchgestrichene Mülltonne bedeutet, dass Elektro- und Elektronikgeräte **nicht** in den Hausmüll gehören. Sie müssen die Altgeräte an einer Erfassungsstelle abgeben.
Vor der Abgabe haben Sie Altbatterien und Altakkumulatoren, die nicht vom Altgerät umschlossen sind, von diesem zu trennen.

Rückgabemöglichkeiten:
Als Endnutzer können Sie beim Kauf eines neuen Gerätes, Ihr Altgerät (das im Wesentlichen die gleiche Funktion wie das bei uns erworbene neue erfüllt) kostenlos zur Entsorgung abgeben.
Kleingeräte bei denen keine äußere Abmessungen größer als 25 cm sind können unabhängig vom Kauf eines Neugerätes in Haushaltsüblichen Mengen abgeben werden.

Möglichkeit Rückgabe an unserem Firmenstandort während der Öffnungszeiten:
Simac GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

Möglichkeit Rückgabe in Ihrer Nähe:
Wir senden Ihnen eine Paketmarke zu mit der Sie das Gerät kostenlos an uns zurücksenden können.
Hierzu wenden Sie sich bitte per E-Mail an Service@joy-it.net oder per Telefon an uns.

Informationen zur Verpackung:
Verpacken Sie Ihr Altgerät bitte transportsicher, sollten Sie kein geeignetes Verpackungsmaterial haben oder kein eigenes nutzen möchten kontaktieren Sie uns, wir lassen Ihnen dann eine geeignete Verpackung zukommen.

CE

5. Support

Wir sind auch nach dem Kauf für Sie da. Sollten noch Fragen offen bleiben oder Probleme auftauchen stehen wir Ihnen auch per E-Mail, Telefon und Ticket-Supportsystem zur Seite.

E-Mail: service@joy-it.net
Ticket-System: http://support.joy-it.net
Telefon: +49 (0)2845 98469 – 66 (10- 17 Uhr)

Für weitere Informationen besuchen Sie unsere Website:

www.joy-it.net