SEN-KY039HS
Heartbeat sensor

TABLE OF CONTENT

1. Introduction

Dear customer,

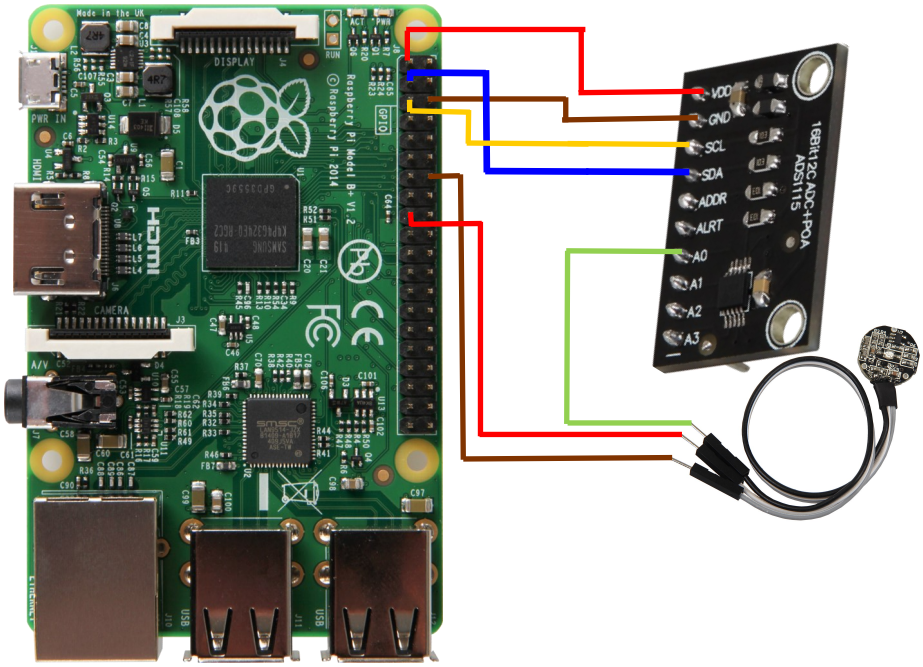
Thank you for choosing our product.
In the following, we will show you what to consider during commissioning and use.

If you experience any unexpected problems during use, you are welcome to contact us.

2. Use with the Raspberry Pi

2.1 Connection

Since the heartbeat sensor is an analog sensor and the Raspberry Pi has no analog inputs, you will need to use an analog-to-digital converter. For example our ADC COM-KY053ADC.



Raspberry Pi	KY039HS	ADC
3v3 (Pin1)		VDD
GND (Pin 6)		GND
SCL (Pin 5)		SCL
SDA (Pin 3)		SDA
	Black	A0
GND (Pin 14)	Grey	
3v3 (Pin 17)	White	

2.2 Installation

First, you need to install the library for the analog-to-digital converter.
This was published under the following link under the MIT license.

To install the library, simply enter the following command in the console:

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

In addition, you must enable I2C on your Raspberry Pi by entering the following command:

```
sudo raspi-config
```

Now go to **Interfacing Options** -> and activate **I2C**.

2.3 Program 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)

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 Program example

```
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];
```

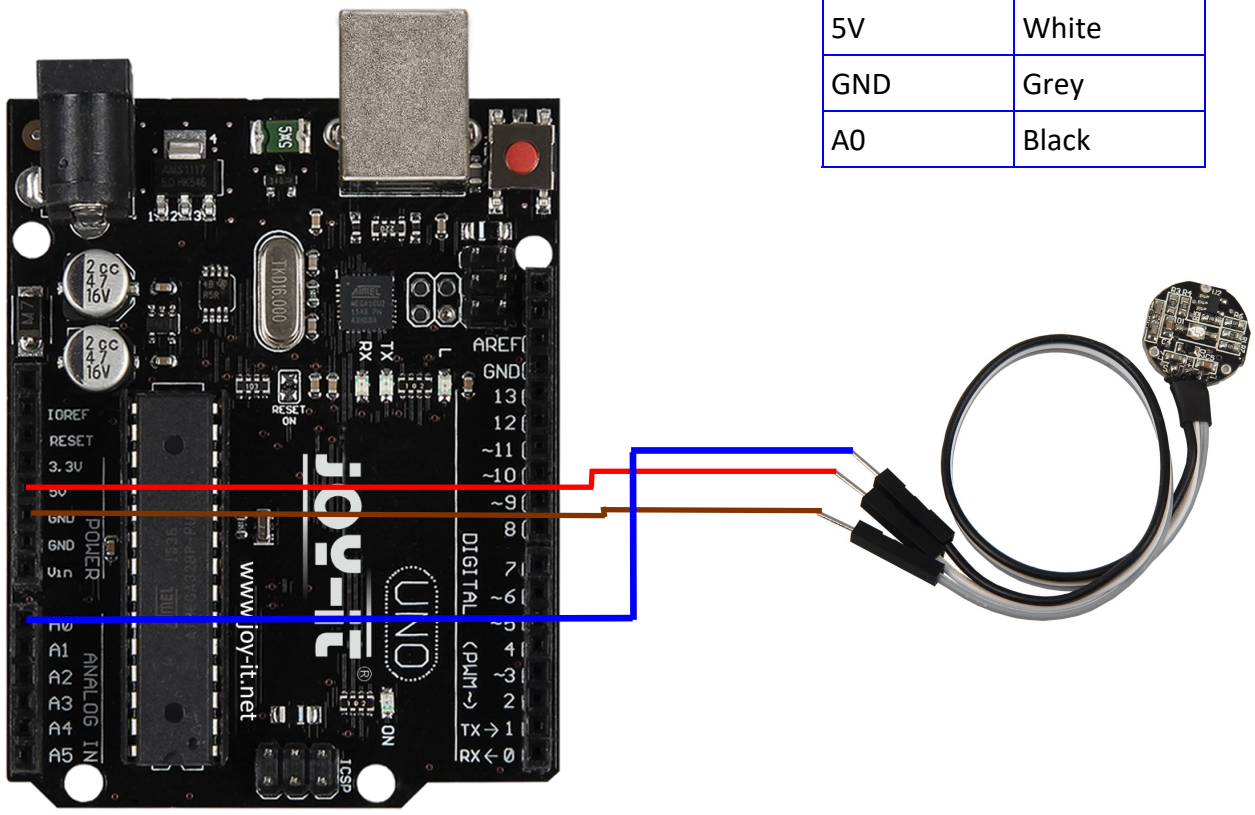
2.3 Program example

```
            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. Use with the Arduino

3.1 Connection

Arduino	KY039HS
5V	White
GND	Grey
A0	Black



3.2 Program example

```
////////////////////////////////////////////////////////////////////
/// 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. Further Information

Our information and take-back obligations according to the ElektroG

Symbol on electrical and electronic equipment

This crossed-out dustbin means that electrical and electronic equipment does not belong in the household waste. You must return the old appliances to a collection point. Before handing over waste batteries and accumulators that are not enclosed by waste equipment must be separated from it.

Return options
As an end user, you can return your old appliance (which essentially fulfils the same function as the new appliance purchased from us) free of charge for disposal when you purchase a new appliance. Small appliances with no external dimensions greater than 25 cm can be disposed of in normal household quantities independently of the purchase of a new appliance.

Possibility of return at our company location during opening hours
Simac GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

Possibility of return in your area
We will send you a parcel stamp with which you can return the device to us free of charge. Please contact us by e-mail at Service@joy-it.net or by telephone.

Information on packaging
If you do not have suitable packaging material or do not wish to use your own, please contact us and we will send you suitable packaging.

CE

5. Support

We are also there for you after your purchase. If any questions remain open or problems arise, we are also available by e-mail, telephone and ticket support system.

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

For more information visit our website:

www.joy-it.net