KY-039 Heartbeat sensor module

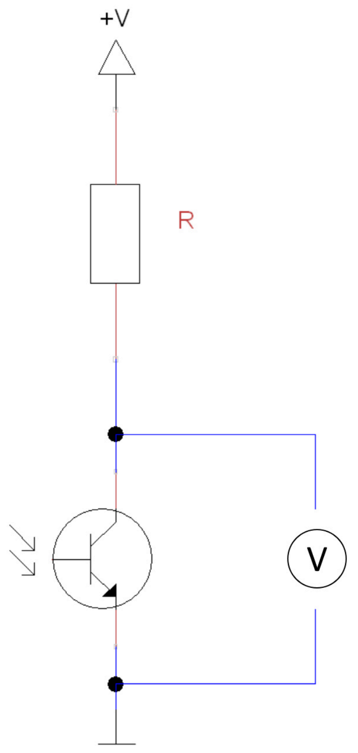
Contents

Picture

Technical data / Short description

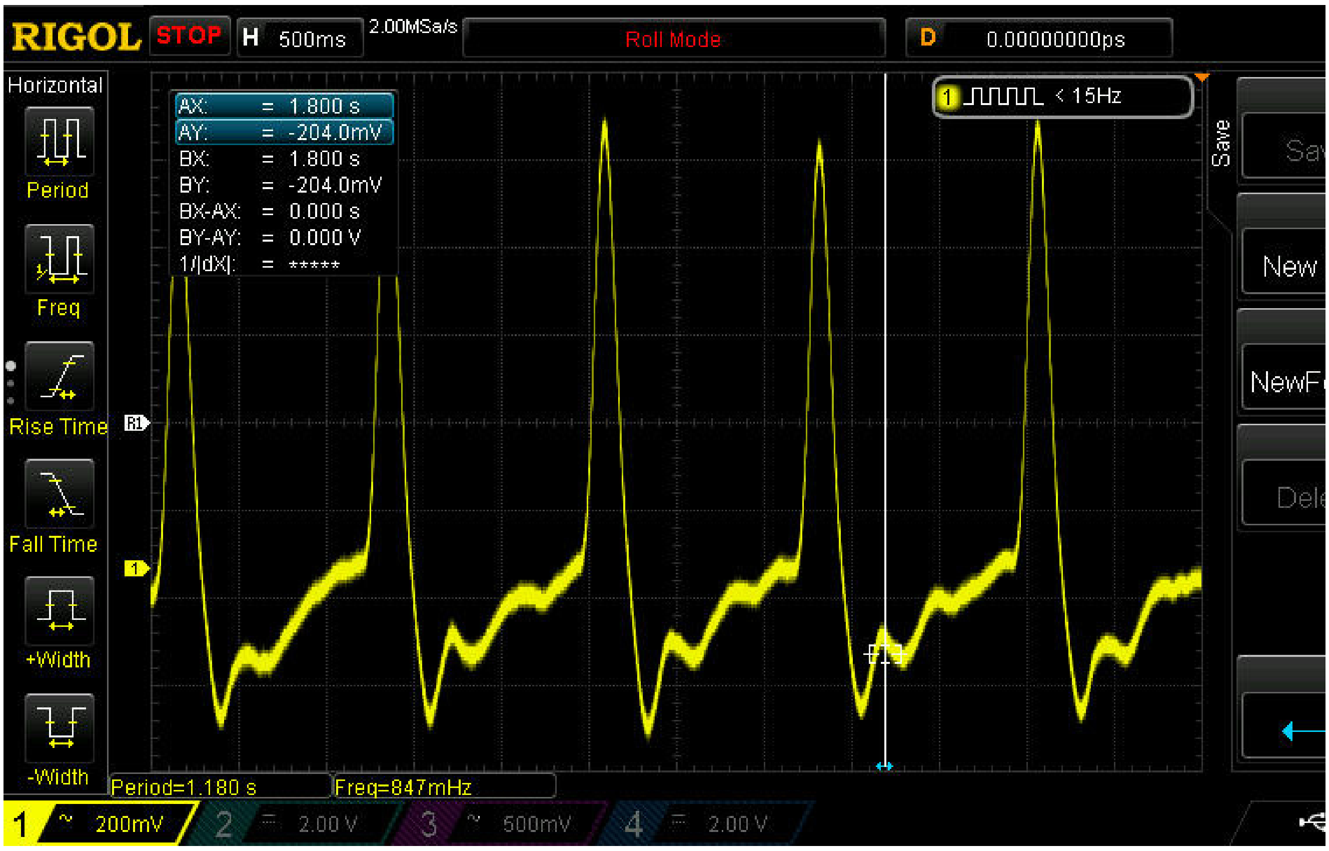
While sticking a finger between the infrared diode and the photo transistor you can detect the pulse at the signal out.

The explanation of the functionality of a photo transistor is simple: It works like a normal transistor - you will detect a higher electricity if the control voltage is higher. Instead of the control voltage, the photo transistor uses the light. If it it's a stronger light, you will measure a higher electricity.



If you switch a resistor and a transistor in row, you will see a special behavior if you measure the voltage at tl
transistor: if much light shines at the transistor, you will measure a very low voltage (near 0V).
If the transistor stays in the dark, you will measure a voltage near +V.

With this sensor module, which contains a phototransistor and an infrared diode, you can measure a pulse. Fc
you have to put a finger between the diode and the transistor. Explaination: exactly like you know it from a
flashlight, the light can shine through your skin. If you hit a vein with the light, you can see the pumping of th
a little bit. You can see it because the blood concentration is different on different areas in the vein, which me
you can see brightness differences in the blood flow. Exactly this differences can be measured with the senso
module and with that you will get the pulse. You can see it clear at the oszilloskop picture.

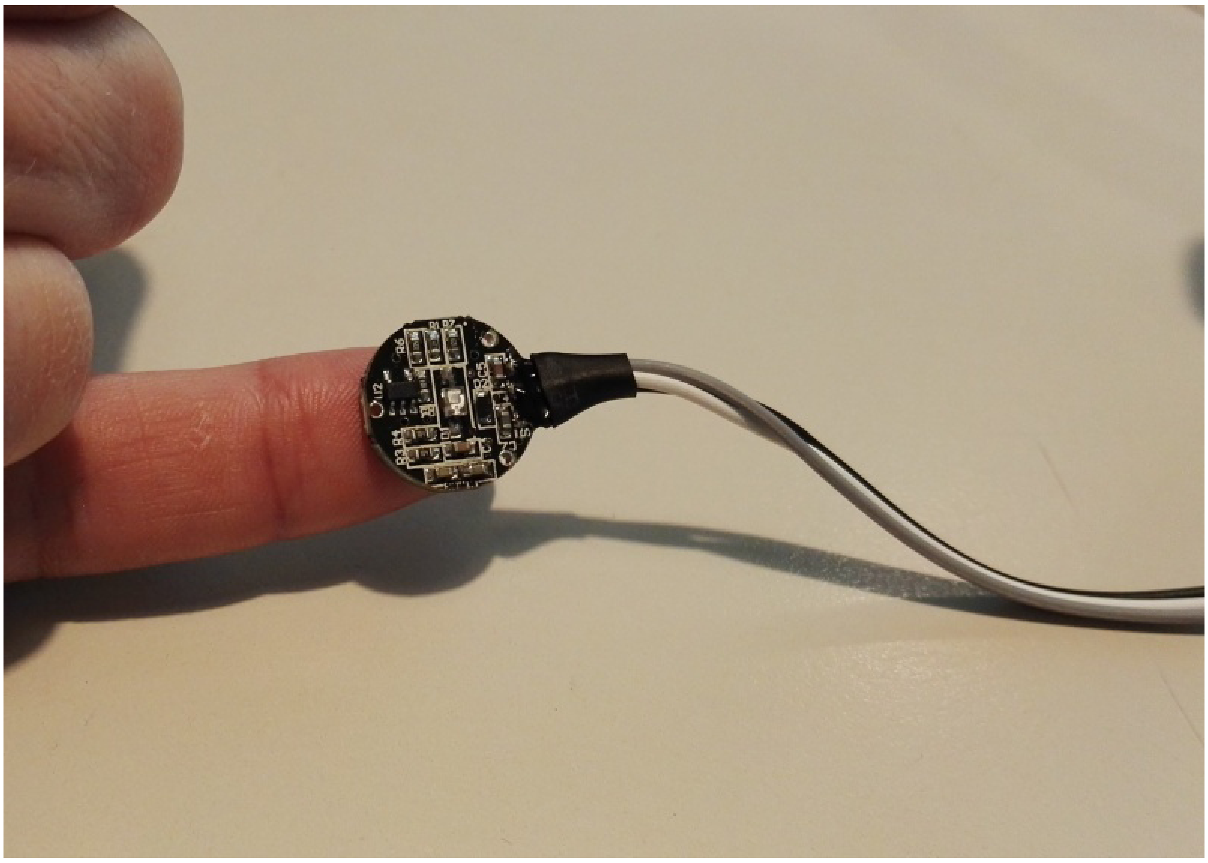


AX: = 1.800 s
AY: = -204.0mV
BX: = 1.800 s
BY: = -204.0mV
BX-AX: = 0.000 s
BY-AY: = 0.000 V
1/|dX|: = *****

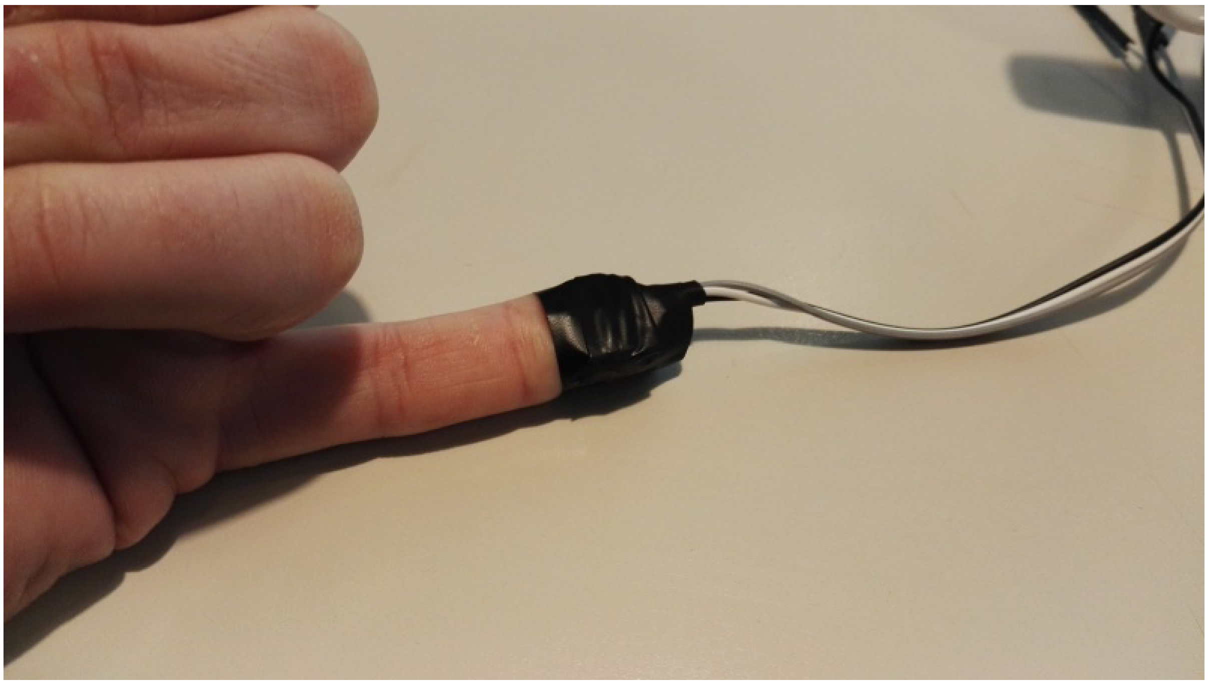
Period=1.180 s
Freq=847mHz

This shows, the changes of the voltage at the phototransistor - and with it the brightness change which comes from the flowing blood. The peaks above show the heartbeat. If you calculate the measured beats per recorded time, you will nearly get 71 beats per minute (bpm).

Additionally to that, you can see that the signal from the transistor is really low or the transistor is really sensitive to measure the small signal. For optimal results, we advice to prepare the module like it is shown in the pictures below:

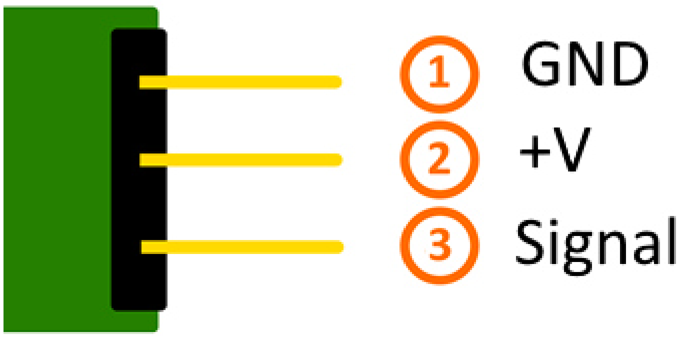


To increase the sensitivity of the sensor, we advise to fix it at your finger with tape.



The heartbeat will be measured really good if the sensor is exactly above a big vein.
To get a better signal, we advise you to change the position of the sensor, if necessary.

Pinout



Pinout:

Gray -> GND

White -> +V

Black -> Signal

Code example Arduino

The following code example was written by Dan Truong. He published it under the following [[1]] . It is available under the [|MIT OpenSource License].

The example below is a german translation, you can get the original via download.

This code shows a so called "Peak-detection". It doesn't record a heartbeat history, instead of that it looks for peaks in the recorded data, which was detected as heartbeat and shown with a LED. You can calculate the pulse with the known delay. If you move the finger to strong, it could take some time till the program reacts to the new situation and gives out the correct data.

```
////////////////////////////////////////////////////////////////////////
/// 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


////////////////////////////////////////////////////////////////////////
/// @param[in] IRSensorPin Analog PI which is connected with the sensor
/// @param[in] delay (msec) The delay between the calls of the sample function
//                     You will get the best results if take 5 sample for each heart beat
///                    Not slower than 150 mSec for 70 BPM pulse
///                    Better take 60 mSec for a pulse of 200 BPM.
///
/// @Short description
/// This code is a so called Peak-detection.
/// It doesn't record a heart beat history,
/// instead it will search for peaks in the recorded data,
/// and show them via LED. While knowing the delays you can calculate the pulse.
////////////////////////////////////////////////////////////////////////

int rawValue;


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


  bool result = false;

  rawValue = analogRead(IRSensorPin);
  // Measuring of the voltage value at the photo transistor
  rawValue *= (1000/delay);

  // If the difference of the current value and the last value is to high
  // ( maybe because you moved the finger away from the sensor)
  // maxValue will be resetted to get a new basic
  if (rawValue * 4L < maxValue) {maxValue = rawValue * 0.8;}    // Detect new peak
       if (rawValue > maxValue - (1000/delay)) {
    // The peak will be detected here. If the new rawValue is bigger than
        // the last maxValue, it will be detected as a peak.
    if (rawValue > maxValue) {
      maxValue = rawValue;
    }
    // Allocate only one heartbeat to a peak.
    if (isPeak == false) {
      result = true;
    }
    isPeak = true;
  } else if (rawValue < maxValue - (3000/delay)) {
    isPeak = false;
    // Here the maxValue will be decreased at each run
    // because if you don't do that the value would be every time lower or the same.
    // Also if you move the finger a bit the signal would be weaker without that.
    maxValue-=(1000/delay);
 }
  return result;
}

////////////////////////////////////////////////////////////////////////
// Arduino main code
////////////////////////////////////////////////////////////////////////
int ledPin=13;
int analogPin=0;

void setup()
```

```
{
  // The included LED of the Arduino (Digital 13), will be used as output here.
  pinMode(ledPin,OUTPUT);

  // Serial output initialization
  Serial.begin(9600);
  Serial.println("Heartbeat detection example code");
}

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

// The main program has two tasks:
// - The LED will light up after detecting a heart beat
// - Calculating of the pulse and outputting of it at the serial out.

void loop()
{
  static int beatMsec = 0;
  int heartRateBPM = 0;
      Serial.println(rawValue);
  if (heartbeatDetected(analogPin, delayMsec)) {
    heartRateBPM = 60000 / beatMsec;
        // LED-output for the heart beat heart beat
    digitalWrite(ledPin,1);

    // Output of the serial data
    Serial.print(rawValue);
    Serial.print(", ");
    Serial.println(heartRateBPM);

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

Connections Arduino:

Sensor Signal	=	[Pin 0]
Sensor +V	=	[5V]
Sensor -	=	[Pin GND]

Example program download

KY-039-HeartBeatDetector original by DanTruong

KY-039_Heartbeat-sensor-module with english comments by Joy-IT

Code example Raspberry Pi

!! Attention !! Analog Sensor !! Attention !!

Unlike the Arduino, the Raspberry Pi doesn't provide an ADC (Analog Digital Converter) on its Chip. This limits the Raspbery Pi if you want to use a non digital Sensor.

To evade this, use our *Sensorkit X40* with the *KY-053* module, which provides a 16 Bit ADC, which can be used with the Raspberry Pi, to upgrade it with 4 additional analog input pins. This module is connected via I2C to the Raspberry Pi.
It measures the analog data and converts it into a digital signal which is suitable for the Raspberry Pi.

So we recommend to use the KY-053 ADC if you want to use analog sensors along with the Raspberry Pi.

For more information please look at the infosite: KY-053 Analog Digital Converter

!! Attention !! Analog Sensor !! Attention !!

The program uses the specific ADS1x15 and I2C python-libraries from the company Adafruit to control the ADS1115 ADC. You can find these here: [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code] published under the BSD-License [Link]. You can find the needed libraries in the lower download package.

Normally, the function to detect a heartbeat should start after the configured delayTime (Standard: 10 ms). If it detects a heartbeat, it will output the pulse. Additional to that you can connect a LED with the LED pin (standard: GPIO24) to visualise the detected heartbeat.

If you move the finger to strong or take it back and put it to the sensor again, the program needs a few seconds (3-5 seconds) until it will give you the new, correct data.

```
#######################################################################################
### Copyright by Joy-IT
### Published under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
### Commercial use only after permission is requested and granted
###
### Parts of Code based on Dan Truong's KY039 Arduino Heartrate Monitor V1.0
### [https://forum.arduino.cc/index.php?topic=209140.msg2168654] Message #29
#######################################################################################



# This code is using the ADS1115 and the I2C python libraries for the Raspberry Pi
# It is published by BSD License under the following link
# [https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]
from Adafruit_ADS1x15 import ADS1x15
from time import sleep

# Additional needed modules will be imported and configured
import time, signal, sys, os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Used variables will be initialized
beatsPerMinute = 0
isPeak = False
result = False
delayTime = 0.01
maxValue = 0
schwelle = 25
beatTime = 0
oldBeatTime = 0

# Address allocation ADS1x15 ADC
```

```
ADS1015 = 0x00  # 12-bit ADC
ADS1115 = 0x01  # 16-bit

# Amplification (Gain) will be picked
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
# gain = 256   # +/- 0.256V

# Sample rate of the ADC (SampleRate) will be picked
sps = 8    # 8 Samples per second
# sps = 16   # 16 Samples per second
# sps = 32   # 32 Samples per second
# sps = 64   # 64 Samples per second
# sps = 128  # 128 Samples per second
# sps = 250  # 250 Samples per second
# sps = 475  # 475 Samples per second
# sps = 860  # 860 Samples per second

# ADC-Channel (1-4) will be picked
adc_channel = 0    # Channel 0
# adc_channel = 1    # Channel 1
# adc_channel = 2    # Channel 2
# adc_channel = 3    # Channel 3

# the ADC which is used by the KY-053 is an ADS1115 chipset adc = ADS1x15(ic=ADS1115)

# LED output pin declaration.
LED_PIN = 24
GPIO.setup(LED_PIN, GPIO.OUT, initial= GPIO.LOW)



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

# Buzzer output pin declaration.
def heartBeatDetect(schwelle):
        global maxValue
        global isPeak
        global result
        global oldBeatTime

        # Reading of the voltage at the photo transistor
        # saving of the voltage value into the rawValue variable
        # With "adc_channel" the channel which is connected with the ADC will be picked.
        rawValue = adc.readADCSingleEnded(adc_channel, gain, sps)

        # Reset of the result-variable
        if result == True:
            result = False

        # If the difference between the current value and the last maximum value is to big
        # (maybe bacause you moved the finger to strong for example)
        # Here you see the reset of the maxValue to get a new basic.
        if rawValue * 4 < maxValue: maxValue = rawValue * 0.8; # Detecting of the peak

              if rawValue > maxValue:
                    maxValue = rawValue
              if isPeak == False:
                    result = True

              isPeak = True

        else:

            if rawValue < maxValue - schwelle:
```

```
            if rawValue < maxValue - schwelle:
              isPeak = False
             # Here the max value will be decreased at each run
             # because if you don't do that the value would be every time lower or the same.
             # Also if you move the finger a bit the signal would be weaker without that.

            maxValue = maxValue - schwelle/2

        # If a heartbeat was detected above, the Output will start.
        if result == True:

           # Calculating of the pulse
           # Here the system time will be recorded for every heartbeat
           # At the next heartbeat, the system time will be compared with the last recorded one
           # The difference between them is the time between the heartbeats
           # with that you can also calculate the pulse
            beatTime = time.time()
            timedifference = beatTime - oldBeatTime
            beatsPerMinute = 60/timedifference
            oldBeatTime = beatTime

           # the heartbeat will light up a LED for a short time
            GPIO.output(LED_PIN, GPIO.HIGH)
            time.sleep(delayTime*10)
            GPIO.output(LED_PIN, GPIO.LOW)

            # Calculated pulse will be given to the function
            return beatsPerMinute

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


# ########
# Main program loop
# ########
# After the "delayTime" (standard: 10ms) the function to detect a heartbeat will start.
# After detecting a heartbeat, the pulse will be outputted.

try:
        while True:
                time.sleep(delayTime)
                beatsPerMinute = heartBeatDetect(schwelle)
                if result == True:
                    print "---Heartbeat detected !--- Puls:", int(beatsPerMinute),"(bpm)"



except KeyboardInterrupt:
        GPIO.cleanup()
```

Connection Raspberry Pi:

Sensor KY-039

Signal	=	Analog 0	[Pin A0 (ADS1115 - KY-053)]
+V	=	3,3V	[Pin 1]
GND	=	GND	[Pin 6]

ADS1115 - KY-053:

VDD	=	3,3V	[Pin 01]
GND	=	GND	[Pin 09]

SCL	=	GPIO03 / SCL	[Pin 05]
SDA	=	GPIO02 / SDA	[Pin 03]
A0	=	look above	[Sensor: Signal]

Example program download

KY-039_Heartbeat-sensor_RPi

To start, enter the command:

```
sudo python KY-039_Heartbeat-sensor_RPi.py
```