Faculty of Information Technology
Brno University of Technology

SEN

Intelligent Sensors

Commissioning Heartbeat Sensor and Comparison Against Oximeter

Jiří Záleský (xzales12)
Adrián Tóth (xtotha01)

October 17, 2018

Contents

1 Project

The aim of this project is to comission a heart beat sensor. Within the project, there will be done several measurements, which will be later compared with the results of a valid device - oximeter. Prof. Ing., Dipl.-Ing. Martin Drahanský, Ph.D[1] is the project supervisor.

2 Abstract

The aim of this project is to demonstrate a method of measuring heart beat via infrared light. The upcoming sections will describe further requirements to achieve the right functionality. The results of measurement were compared with a valid heart beat sensor, which measured heart beat in other way than the infrared method.

3 Introduction

The infrared light has a key role in this project. The infrared LED represents a main source of infrared rays which are sent in a direction to phototransistor. These rays may be affected by any kind of object representing a blockade in the way to the phototransistor. Using a finger as a rays blockade, the rays are affected not only by the finger itself, but also by the blood in the finger. As the heart is pulsing inside of the person's body, the blood pressure is different inside of the finger's veins, which has also a great impact on the rays. The emitted rays from the finger are detected by the phototransistor that produces a precise values. Based on these values, there is a possibility to detect heart beat. This method of measurement is shown in Figure 1.

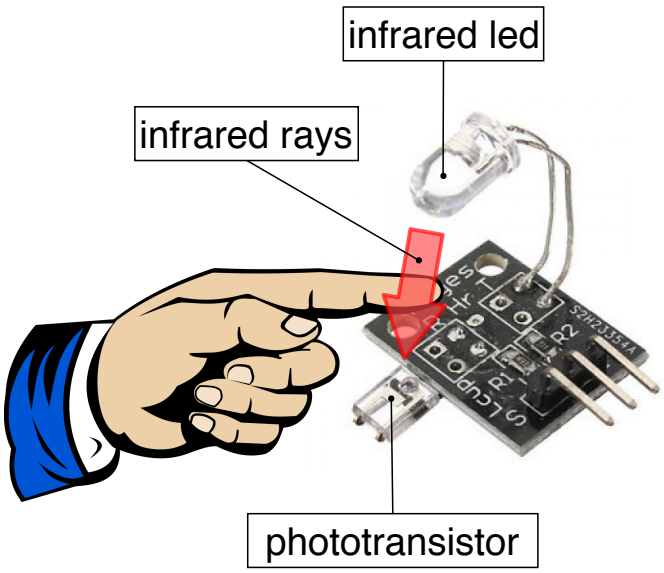

Figure 1: Method of infrared heart beat measurement.

[1] www.fit.vutbr.cz/~drahan

The necessary facilities for the implementation of the project were:

- Device shown in Figure 2.

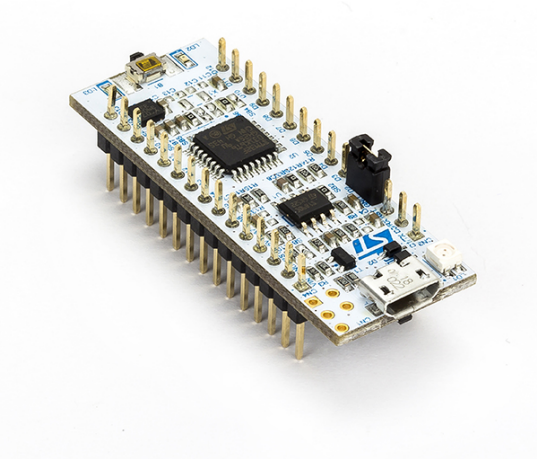

Figure 2: STM32 Nucleo board [4].

- Sensor shown in Figure 3.

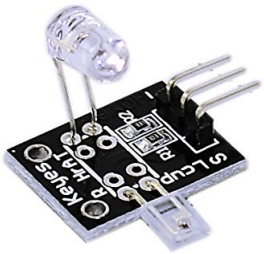

Figure 3: Keyes KY-039 Finger Heartbeat Detection Sensor [1].

- Three female–to–female jumper wires.

After we received the devices, we faced a lot of problems. The main device representing the basic platform was not working properly so we had to ask for a new one. Later, the main device (shown in Figure 2) was replaced to a new one (see Figure 4) due to wrong functionality by the technical assistant of the project - Ing. Tomáš Goldmann[2].

[2] www.fit.vutbr.cz/~igoldmann

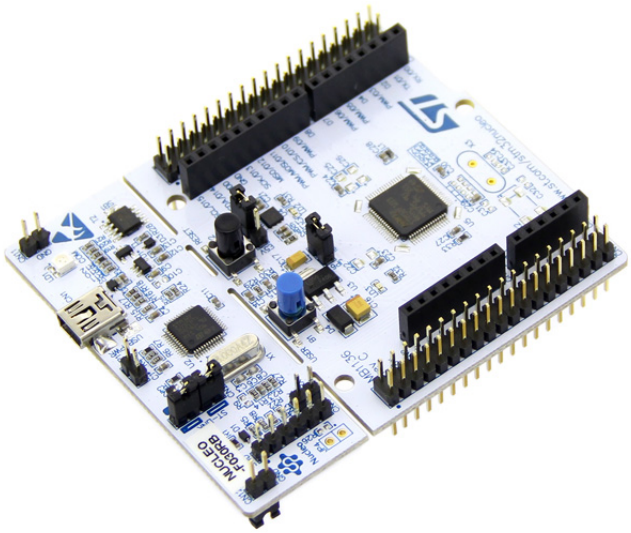

Figure 4: STM32 Nucleo board [2].

The device shown in Figure 4 - NUCLEO-F030R8 [6], was the main platform for the project. The device was equipped with a STM32F030R8 [7] microcontroller unit (MCU), which was designed and made to be suitable for a wide range of applications. MCU includes a set of peripherals through which it communicates with other devices such as the sensor shown in Figure 3. To create a communication pipeline, these units must be connected to each other via jumper wires. Wires provide a connection between the pins of units and so the pins have to be configured correctly.

4 Setup

Firstly, the components must be connected to each other in a right way. Sensor, as a slave component shown in Figure 6, is connected to the main device (master component). These two units are connected via 3 jumper wires to the corresponding pins. The Figure 5 and 6 have three common marked pins: *GND* (orange), *5V* (yellow), *PA0* and *S* (green); by which they are connected together. *PA0* indicates an analog input pin to the master component. A precise description of all the necessary pins is shown in Figure 7.

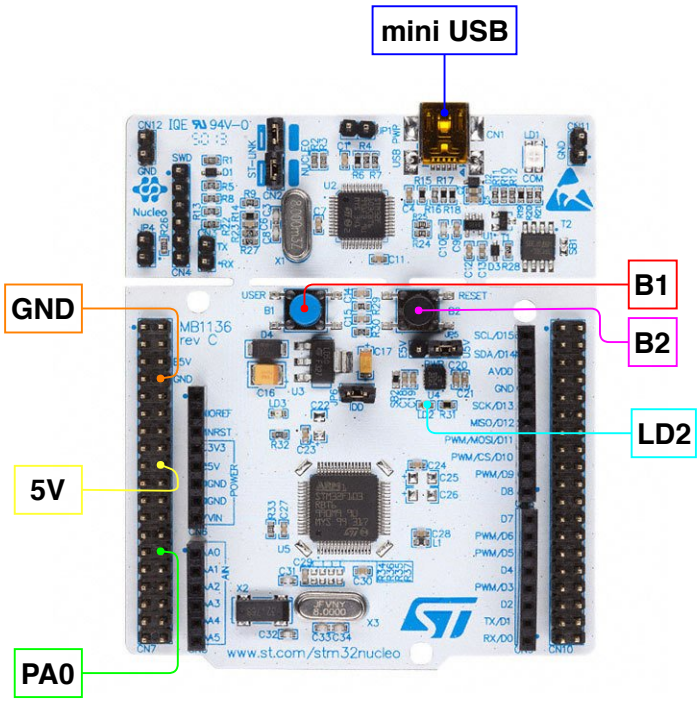

Figure 5: NUCLEO-F030R8 [3] connection scheme.

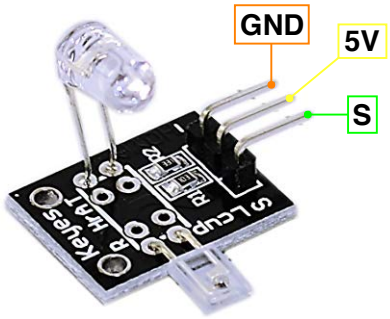

Figure 6: Keyes KY-039 [1] connection scheme relating to Figure 5.

S	sensor output
B1	button 1
B2	button 2
LD2	LED 2
GND	ground
5V	5 volt voltage supply
PA0	pin PA0
mini USB	mini USB port

Figure 7: Explanatory notes.

A connection scheme of the devices is shown in Figure 8. After the devices were correctly connected and attached to the computer via mini USB port (see Figure 9), they were ready to be programmed.

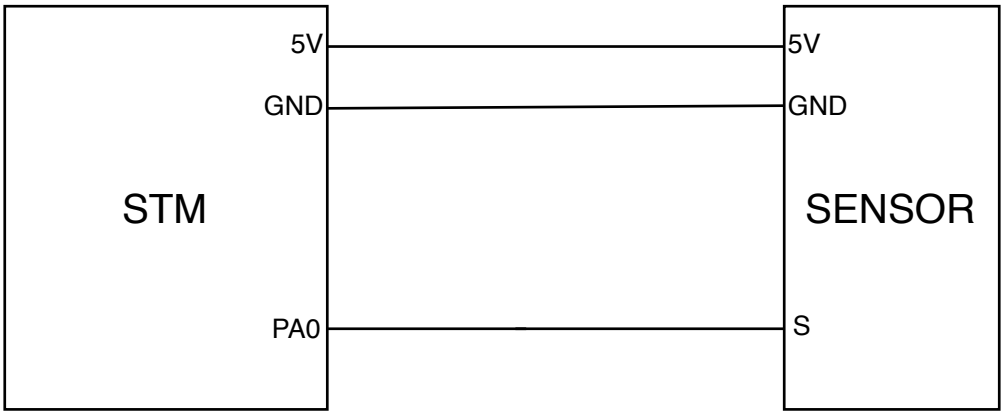

Figure 8: Device connection scheme.

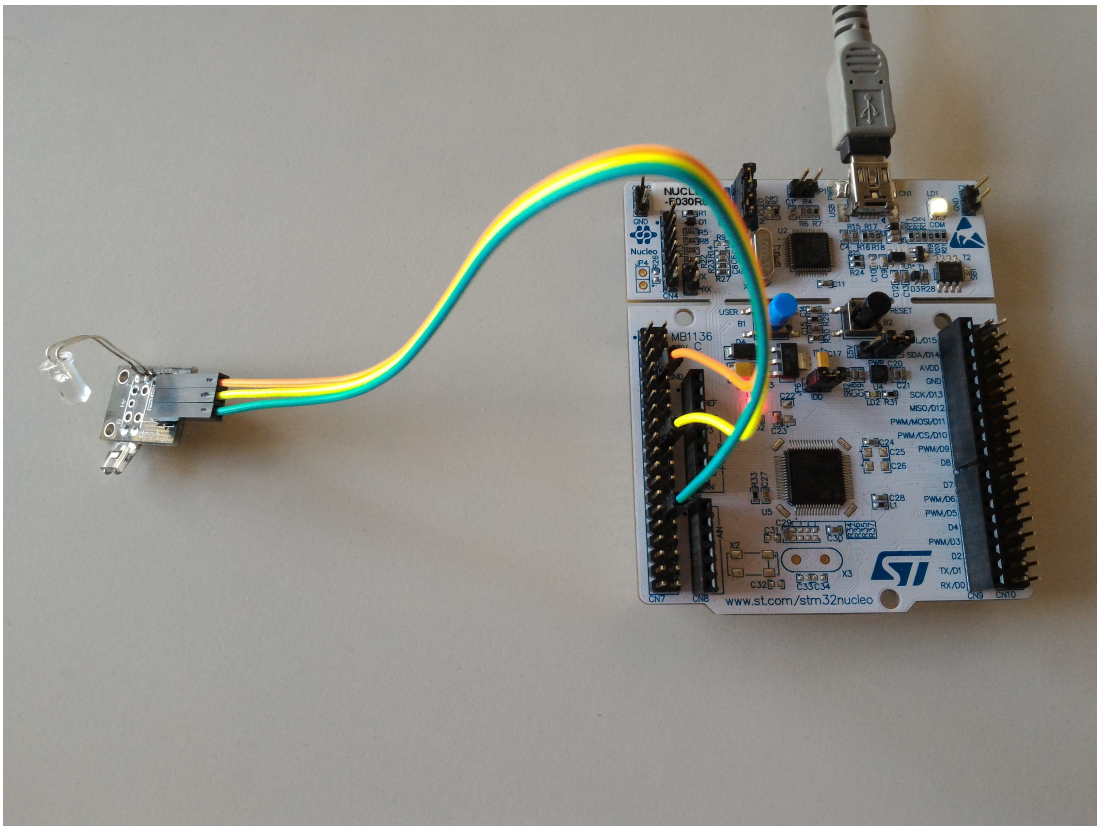

Figure 9: Correct connection of devices.

As we were working on the implementation part of the project, there was a measurement anomaly. The results were affected by different input voltage on the pin. We asked for an advice from the technical assistant. He has recommended us a different device connection which is shown in Figure 10. He suggested to use a protoboard with two 10kΩ resistors and a few additional connection wires.

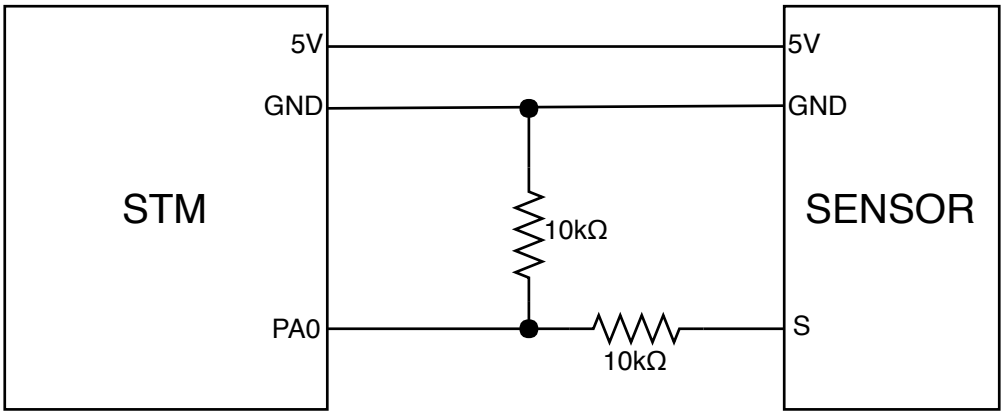

Figure 10: New device connection scheme.

The devices have been reconnected based on the given advice, which is shown in Figure 11. This setup was the proper one where the values have not been affected by voltage.

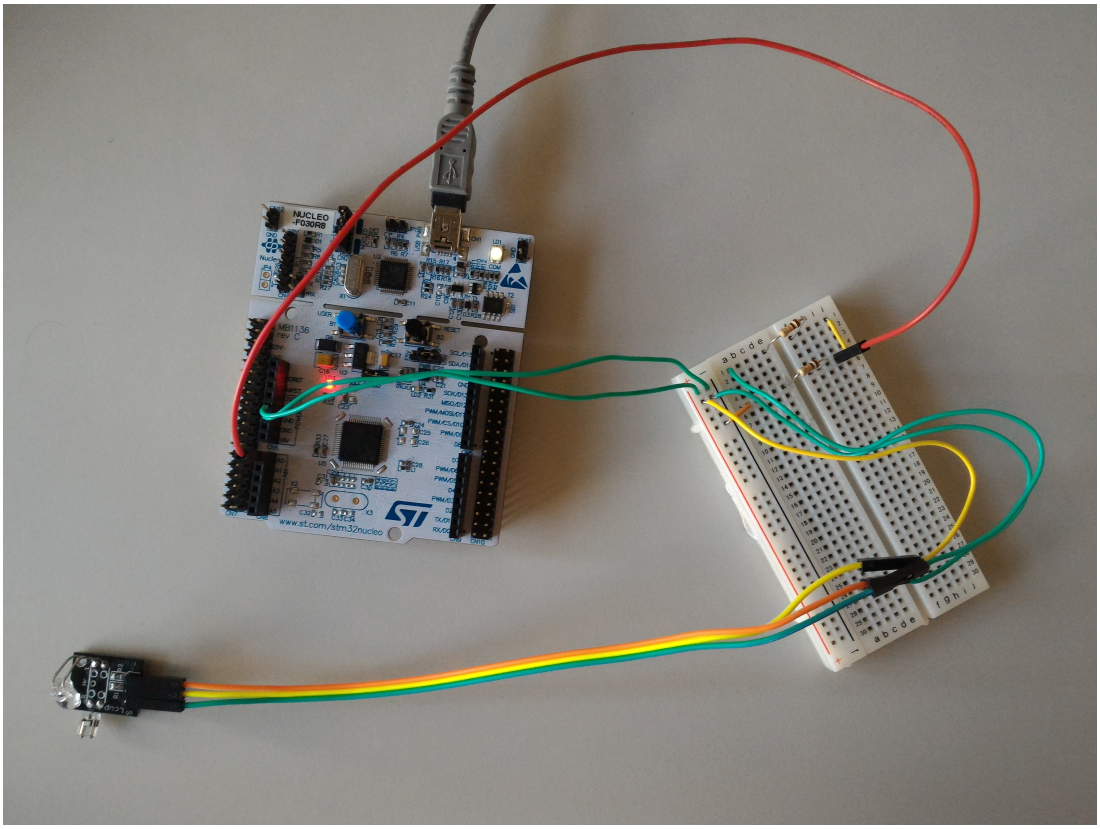

Figure 11: New connection of devices.

5 Implementation

The implementation is based on an algorithm [5], which detects deviation in a set of incoming values from the sensor. The values, incoming as a sequence of measurements, contain a few values that have a certain deviation what indicates the pulse. This is the deviation on which basis we can calculate the hear beat per minutes also know as beats per minute (BPM). The program will light up a LED (see LD2 in Figure 5) on each beat. The BPM is calculated from one period - the time between two hear beats. No history is used for the calculations.

We used *Mbed OS 5*[3] to develop the necessary code for the right functionality. It is a browser-based IDE, which also includes a build-in compiler. Based on your set device in this IDE, you can compile online and download the binary file. The device may be programmed by writing this binary file to the device, which is connected to computer by USB, via any type of file manager. The main source code is shown in Listing 1.

Algorithm itself is a little bit complicated due to dynamic range of incoming values - there is no value from a static interval. The finger position significantly affects the interval of values, but do not have any impact on the deviation between the values. To detect a deviation, it is necessary to keep a previous value from the sequence. A calculation related to comparing action is provided, which will determine divergence between the values. If this action will establish the deviation as large enough, this value will be determined as a beat at that moment.

The precision of the algorithm depends on the device configuration and sensor precision. It is nearly impossible to make a universal algorithm which can determine heart beat on inaccurate values. The more accurate the sensor values, the more precise the results. As an accuracy factor we can count with a finger motion that has the biggest impact on the BPM calculations. By examining the sensor, we can see that the phototransistor is not sealed into rubber. This fact indicates, that the sensor is receiving external infrared rays from the surrounding area. We can assume that the surrounding area can influence the results too.

6 Conclusion

...

[3] https://os.mbed.com/

7 Appendix

Name	Value
n1	v1
n2	v2
n3	v3

Figure 12: Table of measurements.

```
#include "mbed.h"

AnalogIn analog_value(A0);
DigitalOut led(LED1);
int delayTime = 10;

#define LED_ON { led = 1; }
#define LED_OFF { led = 0; }

bool measurement() {
    static int maxValue = 0;
    static bool spikeDetected = false;
    int analogValue;
    bool result = false;

    // analogValue from 0 to 65535
    analogValue = analog_value.read_u16();

    // Voltage measurement transformation
    analogValue = analogValue >> 6; // DIV 64 -> change to 0 - 1023
    analogValue *= (1000 / delayTime);

    // The last maxValue will be detected as a peak.
    if (analogValue * 4L < maxValue) {
        maxValue = analogValue * 0.8;
    }

    // Beat detection
    if (analogValue > maxValue - (1000 / delayTime)) {

        // Change maximum value for next measurement
        if (analogValue > maxValue) {
            maxValue = analogValue;
        }

        // Allocate only one heartbeat to a peak.
        if (spikeDetected == false) {
            result = true;
        }

        spikeDetected = true;
    }
    else if (analogValue < maxValue - (3000 / delayTime)) {
        spikeDetected = false;

        // Change maximum value for remove fake beats
        maxValue -= (1000 / delayTime);
    }

    return result;
}

int main() {
    int beatFrequency;
    int beatsPerMin = 0;

```

```
57      while(1) {
58          if (measurement()) {
59              LED_ON;
60
61              // Count beats per minute
62              beatFrequency = 60000 / beatsPerMin;
63
64              if (beatFrequency > 50 & beatFrequency < 200) {
65                printf("Heart␣beat␣frequency:␣%d␣BPM\r\n", beatFrequency);
66              }
67
68              beatsPerMin = 0;
69          }
70          else {
71              LED_OFF;
72          }
73
74          wait_ms(delayTime);
75          beatsPerMin += delayTime;
76      }
77  }
```

Listing 1: Source code.

8 References

[1] *Keyes KY-039 Finger Heartbeat Detection Sensor*. [Image; Online; Accessed: 2018-10-07]. Retrieved from: `https://images-na.ssl-images-amazon.com/images/I/71MfNkRMYDL._SX425_.jpg`

[2] *STM32 Nucleo board (NUCLEO-F030R8)*. [Image; Online; Accessed: 2018-10-07]. Retrieved from: `https://statics3.seeedstudio.com/images/product/NUCLEO-F030RB.jpg`

[3] *STM32 Nucleo board (NUCLEO-F030R8) - front*. [Image; Online; Accessed: 2018-10-07]. Retrieved from: `https://media.digikey.com/Photos/STMicro%20Photos/MFG_NUCLEO.jpg`

[4] *STM32 Nucleo board (NUCLEO-F042K6)*. [Image; Online; Accessed: 2018-10-11]. Retrieved from: `http://www.rhydolabz.com/images/ARD2648.jpg`

[5] Luboš M.: *Arduino senzor tepu srdce*. 2016-12-13. [Online; Accessed: 2018-09-28]. Retrieved from: `https://navody.arduino-shop.cz/navody-k-produktum/arduino-senzor-tepu-srdce.html`

[6] STMicroelectronics: *NUCLEO-F030R8*. [Online; Accessed: 2018-10-07]. Retrieved from: `https://os.mbed.com/platforms/ST-Nucleo-F030R8/`

[7] STMicroelectronics: *STM32F030R8*. [Online; Accessed: 2018-10-07]. Retrieved from: `https://www.st.com/en/microcontrollers/stm32f030r8.html`