Heart Rate Monitor with the Arduino

Jens Gansloser

Fabian Meyer

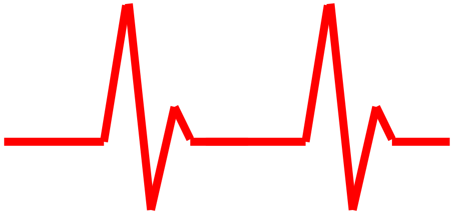
SS14 – Ubiquitos Computing Mini-Project

Project

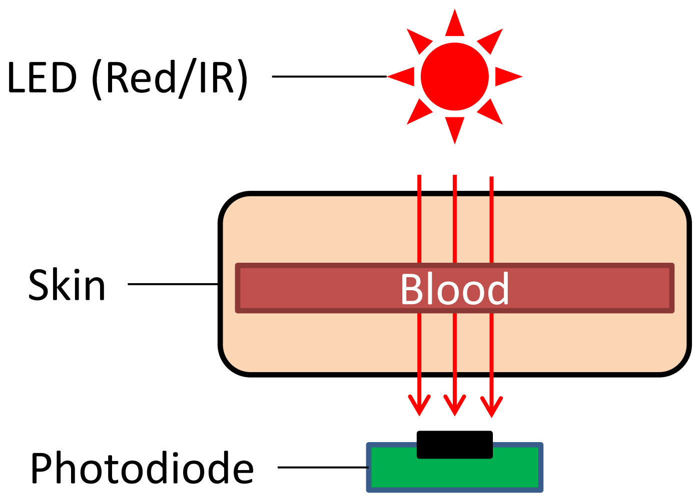
Create a Heart Rate Monitor with the Arduino.

Advantages against commercial devices:

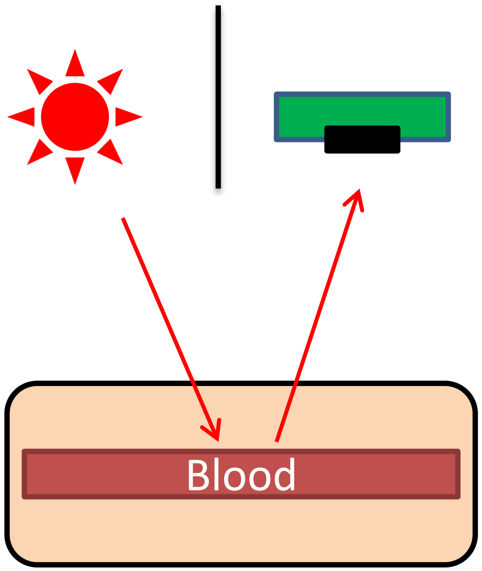
- Full source code available
- Full insight into the used technology
- Can be compared to commercial ones in precision and performance
- Higher performance/precision?



Measurement Principle



Light through skin



Light reflected

Heart Rate and Oxygen Saturation

- Red LED (660nm)
- IR LED (940nm)
- Rate: time between two peaks
- Saturation: difference between red and infrared light intensity

Detect minima and calculate time difference.

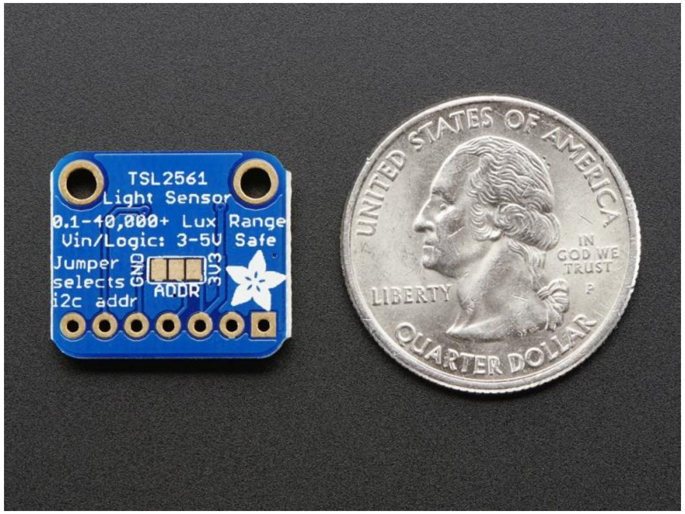
Hardware

- Adafruit TSL Light Sensor
  - Broadband and IR Photodiode

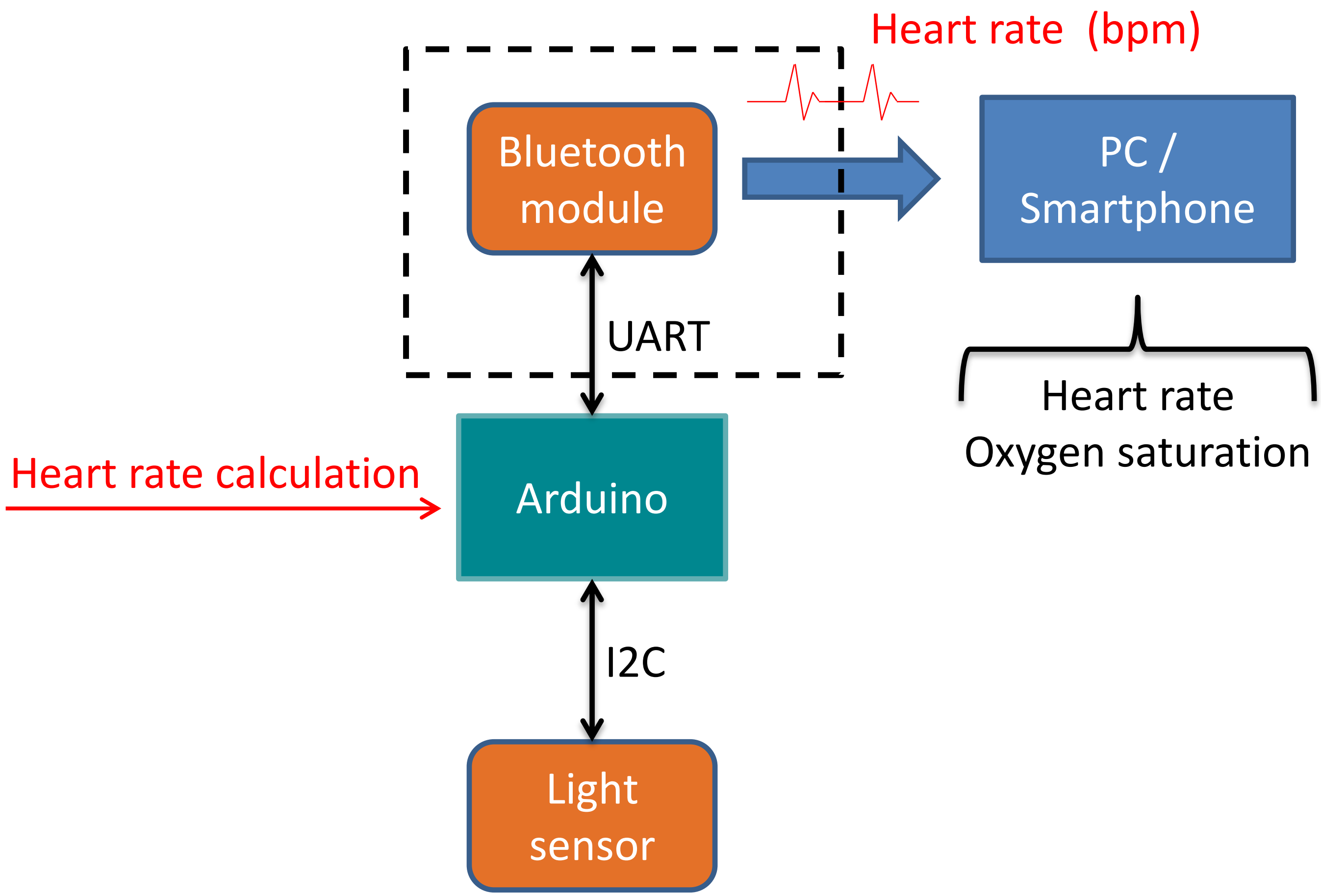
- Red and IR LED

- Bluetooth Module (not used)

- Arduino
  - Needs I2C and USART support

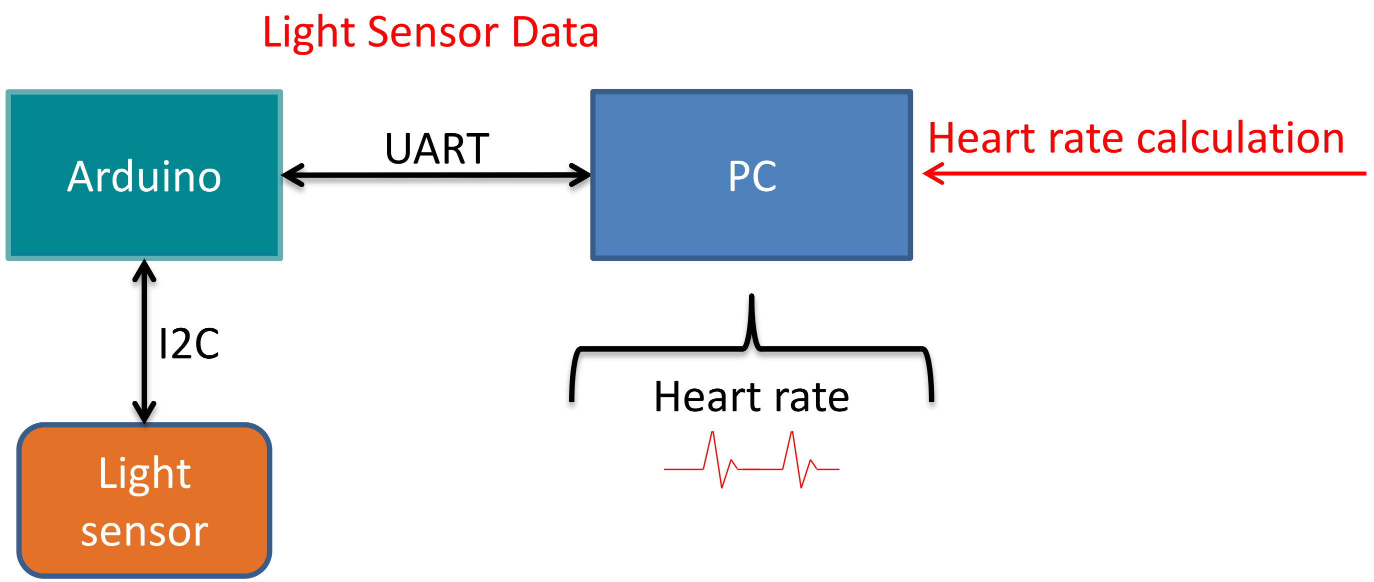


Data Flow


Heart rate (bpm)
Bluetooth module
PC / Smartphone
UART
Heart rate
Oxygen saturation
Heart rate calculation
Arduino
I2C
Light sensor

Data Flow

- Implementation not trivial: Prototype implemented on PC



Software

Libraries

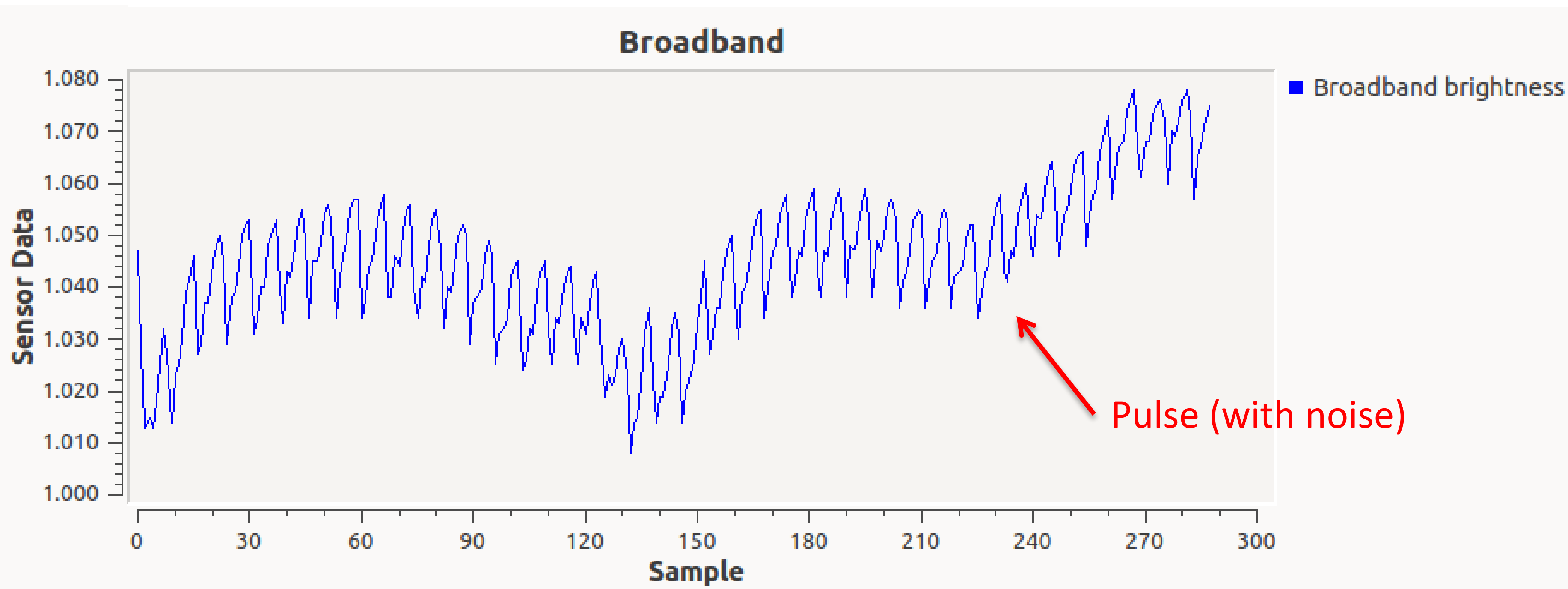
- QSerialPort
  - Serial port control
- FFTW
  - Calculation of the Fourier Transform
- Qt
  - Graphical User Interface
- Qwt
  - Graphs

Heart Rate Monitor

- GUI
  - Debugging and information display
- Serial Interface
  - Get sensor data
  - Get sensor settings
  - Set sample rate
- FFT Module
  - Signal processing
- Arduino Software
  - Read sensor light value (extended driver)

Signal Processing

Output signal from the light sensor (array of discrete light values):



How to get the Heart Rate (= Minima time difference)?

First Approach

- Determine the grade
- Is it declining or rising?
- Determine minima

First Approach

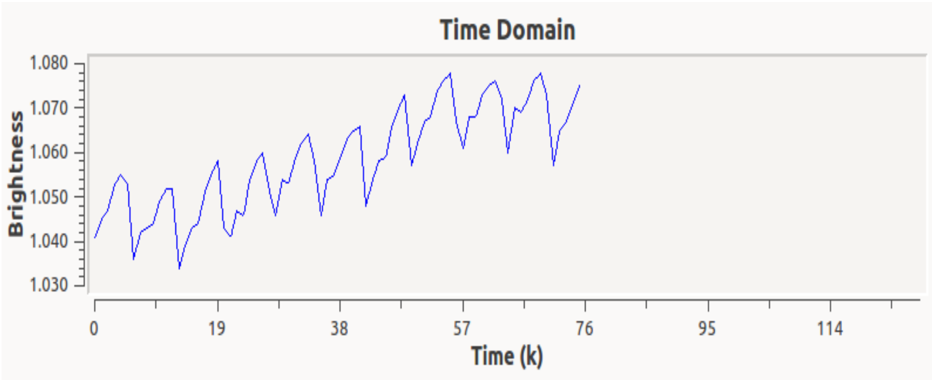
- Determine the grade
- Is it declining or rising?
- Determine minima

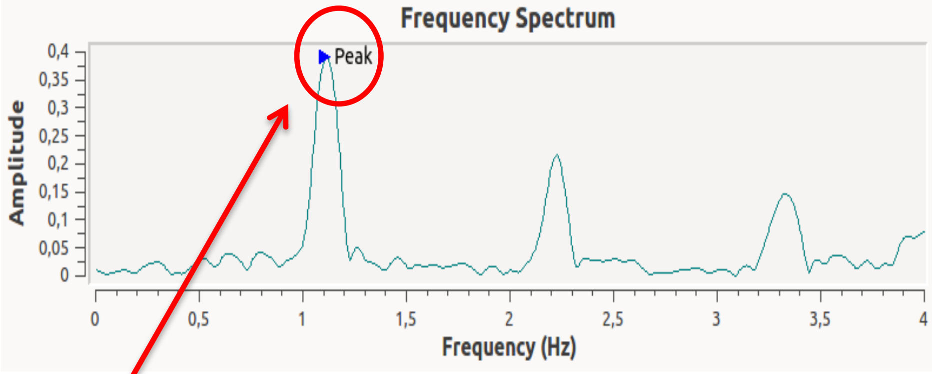
Does not work!
Input light signal contains a lot of noise, a lot of shifting in the Y-Offset, different environment light, …

Fourier Transform

Elegant way to determine the heart rate

Transform





Discrete Signal

Frequencies

Peak Frequency = Heart Rate

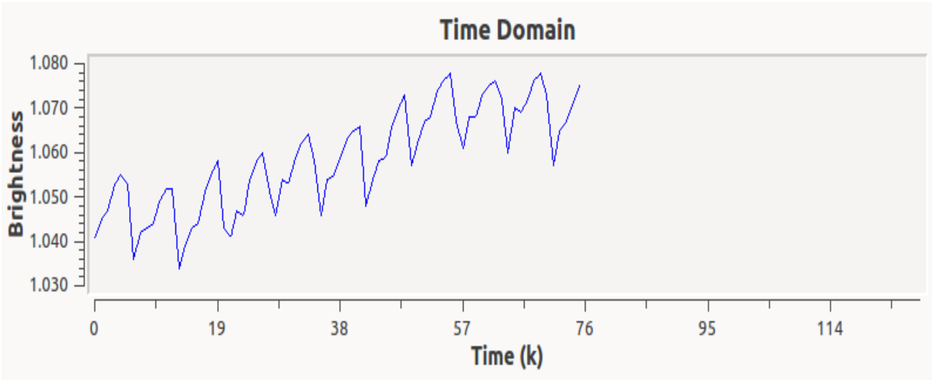
Fourier Transform

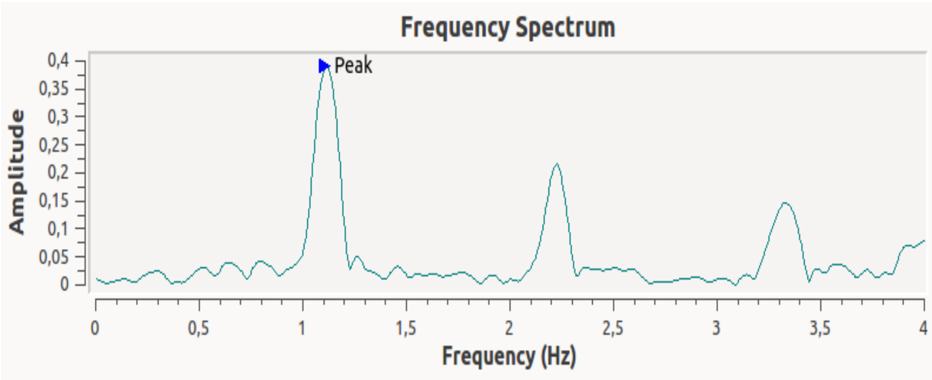
Unfortunately not that easy …

Transform

. . . .

Filtering, Window Function, Zero Padding, …





Discrete Signal

Frequencies

Which Fourier Transformation?

- Complex, Real?
- Discrete, Continues?
- Periodic, Aperiodic?

→ Different application fields

→ Complex Discrete Fourier Transform (DFT) - Forward transform synthesis (polar form)

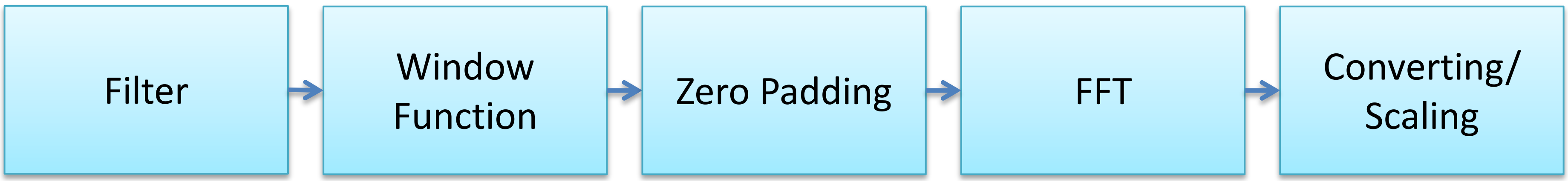
$$X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]\, e^{-j2\pi kn/N}$$

```
N samples
x[n] discrete input values
X[n] discrete output values
Calculation with complex numbers
```

Complex DFT

- Input values
  - Array of complex data (discrete input data)
  - Real part = Sensor values
  - Complex part = 0
- Output values
  - Array of complex data (discrete output data)
  - Rectangular coordinate system (complex and real values – cos and sin functions)
  - Transformation to polar coordinate system with magnitude and phase (human readable)

Signal Processing



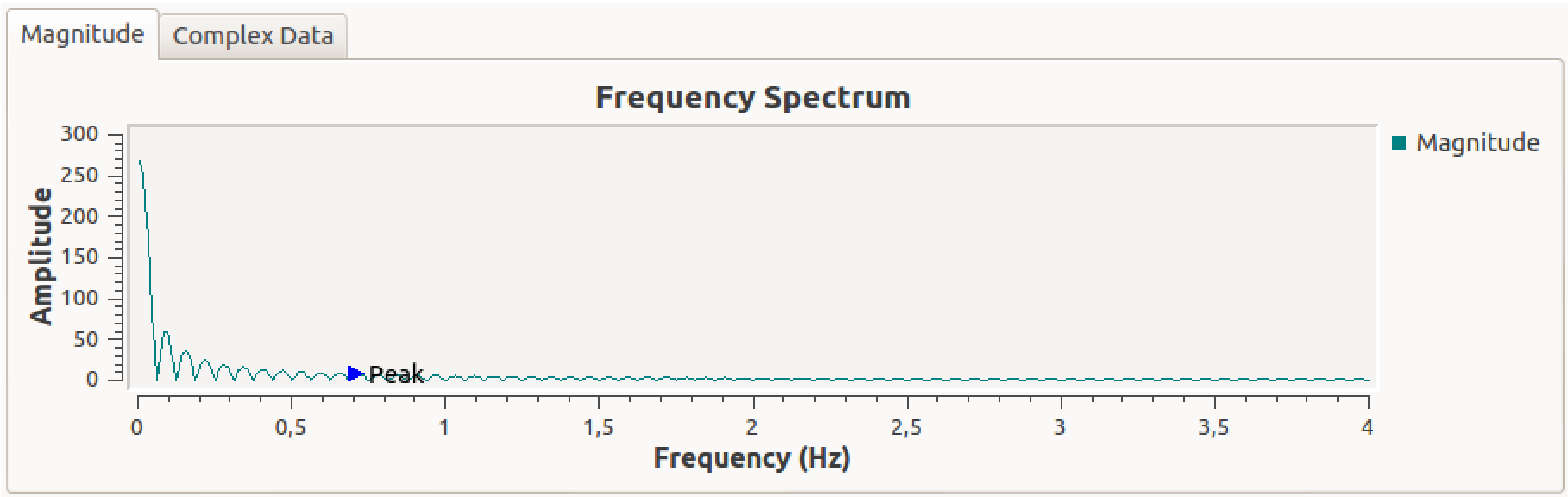
Signal Processing

Filter → Window Function → Zero Padding → FFT → Converting/ Scaling

Filter	Window Function	Zero Padding	FFT	Converting/ Scaling
Butterworth Bandpass Filter (Convolution) → Remove unwanted frequencies	Hanning-Window → Reduces Leakage Effect	Pad additional samples with zero → Interpolation (for higher frequency resolution)	Time Domain → Frequency Domain	Interpret Output Array, Prepare data, Remove DC offset, Find peak frequency, convert to bpm

Signal Processing

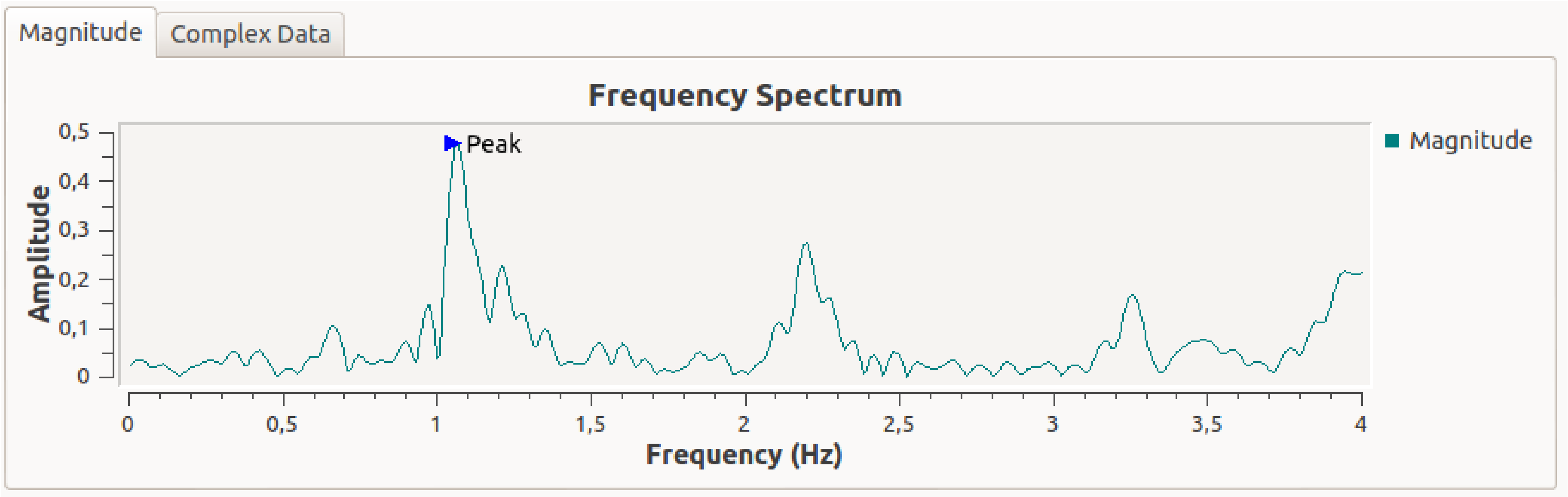
- Frequency Spectrum



No Filter and no Window Function

Signal Processing

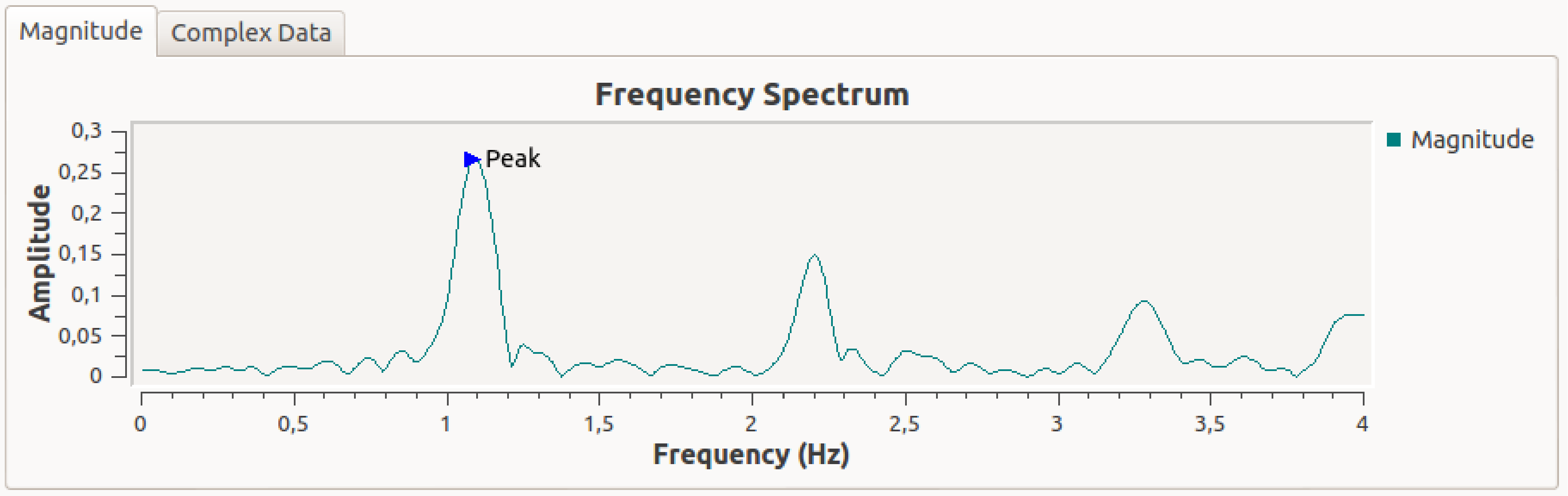
- Frequency Spectrum



No Window Function

Signal Processing

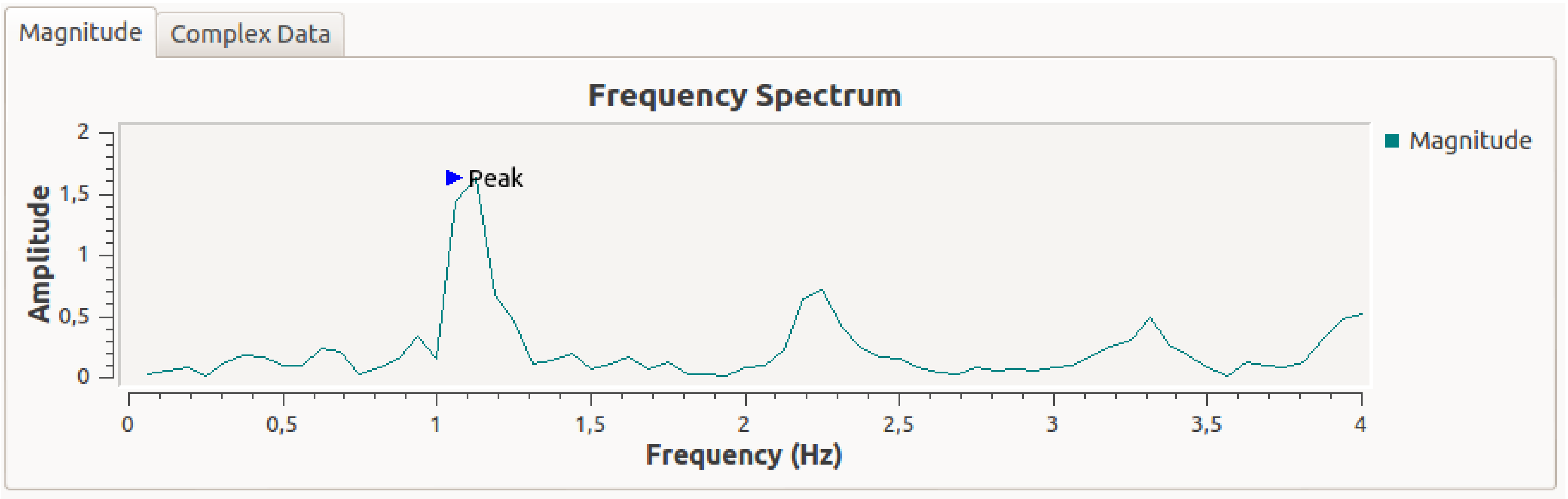
- Frequency Spectrum



With everything

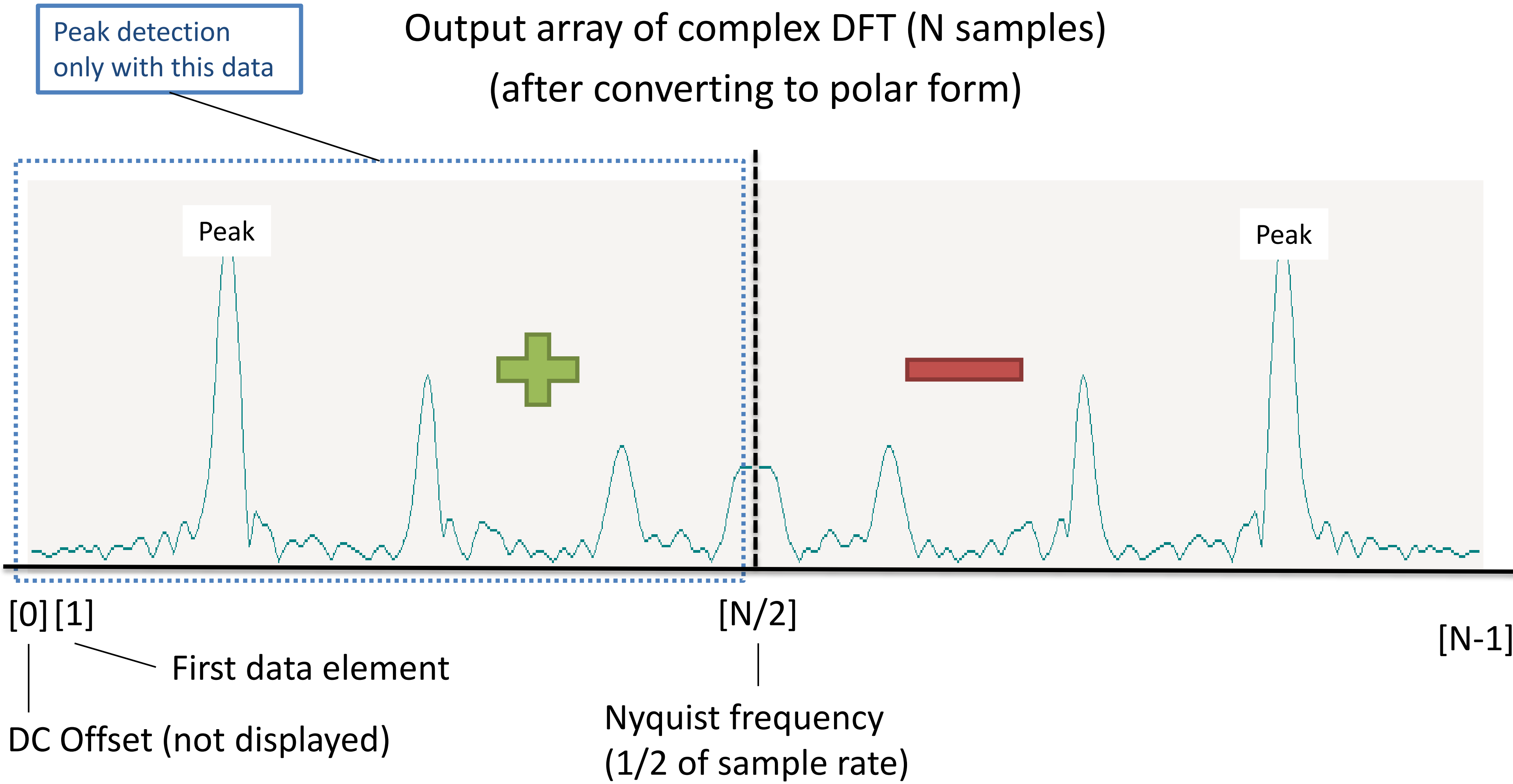
Signal Processing

- Frequency Spectrum



No Zero Padding

Signal Processing



Signal Processing

- A lot of things to consider:
  - Sample Frequency (Nyquist-Shannon theorem)
  - Segment Duration
  - Which Filter?
  - Filter Parameters
  - Target Frequency
  - Filter stabilization time
  - Correct data scaling
  - Zero Padding (interpolation) – numer of samples
  - Find correct frequencies
  - Improving frequency resolution
  - Number of samples
  - Timeinterval for doing FFT
  - …

Further Steps / Improvement

- Port to Arduino
  - FFT library is available - http://wiki.openmusiclabs.com/wiki/ArduinoFFT
  - Research needed, if performance is enough
  - Else do signal processing on smartphone
- Use smaller uC (ATtiny, …)
- Add bluetooth module (trivial)
- Add oxygen saturation measurement (easy to implement - the technology is the same as the heart rate measurement)
- Create wrist band (smaller LED)
- Brighter LED (to get greater frequecy peaks)

Sources

- *http://www.dspguide.com/*
- *Mastering the Discrete Fourier Transform in One, Two or Several Dimensions - Pitfalls and Artifacts, Isaac Amidror, Springer*
- *DFT – Diskrete Fourier-Transformation, André Neubauer, Springer*
- *Signaltheorie, Alfred Mertins, Springer*
- *http://www.thefouriertransform.com/series/fourier.php*
- *https://ccrma.stanford.edu/~jos/mdft/*
- *http://paulbourke.net/miscellaneous/dft/*
- *http://www.ignaciomellado.es/blog/Measuring-heart-rate-with-a-smartphone-camera*
- *http://www-users.cs.york.ac.uk/~fisher/mkfilter/trad.html*