Energy effect of on-node processing of ECG signals

Richard Verhoeven
Diana Albu, Johan Lukkien

Vitruvius project

Content

- Introduction & motivation
- On-node processing
  - ECG signal components
  - R-peak detection algorithm
  - Quality of detection
- Energy measurements
- Conclusions

Introduction

- Health care is changing
  - Patients are released from hospital as early as possible
  - Fewer medical workers per patient
  - Assisted living

- Remote monitoring of patients
  - Detailed information about recovery status
  - Overview of the status of multiple patients
  - Prevent problems by early detection
  - *Obtain information not otherwise available*

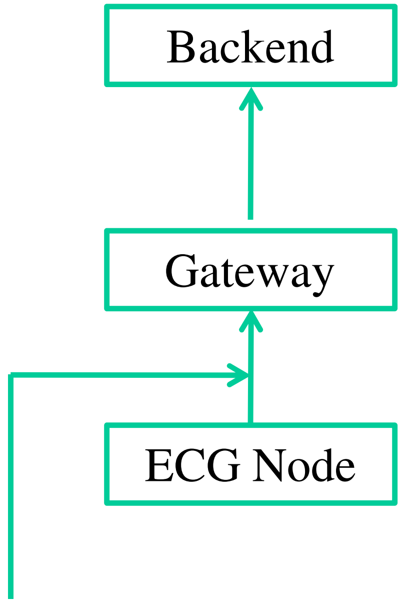
- Wireless sensor networks might help
  - Report medical information to a backend system

Vitruvius project

Motivation for on-node processing

- Information overload for a doctor
- Reduction of false positive alarms
  - take the context of the patient into account
- Privacy of the patient
- Feedback to the patient
- Mobility of the patient
  - less need for a continuous connection

- <u>Lower energy consumption</u>
  - longer lifetime or smaller form factor

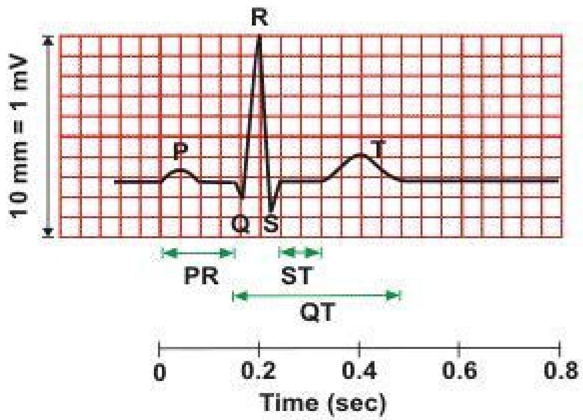


Alternatives
- Raw signals
- Processed signals, e.g. beats, heart rate
- Events, e.g. dangerous condition

Vitruvius project

ECG signal components

- QRS complex
  - RR-interval
  - Heart rate
  - Simple arrhythmias



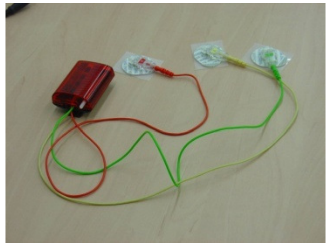
- P and T waves
  - Onset of waves
  - Complex arrhythmias
  - Specific phases in the contractions

- QRS complex or R-peak detection
  - First stage in P and T wave analysis
  - Easier to detect

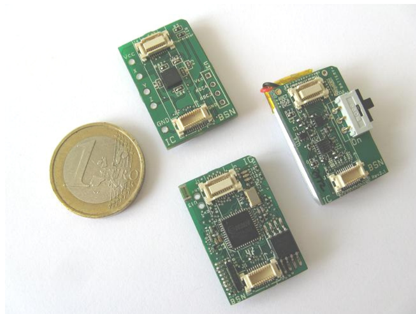
R-peak detection algorithm

- Derived from Pan-Tompkins algorithm
  - J.Pan, W.J. Tompkins, *A real-time QRS detection algorithm,* IEEE transactions on Biomedical Engineering, March 1985

- Aimed at wireless sensor nodes
  - 16-bit microprocessor, 10KB RAM, 48 KB flash

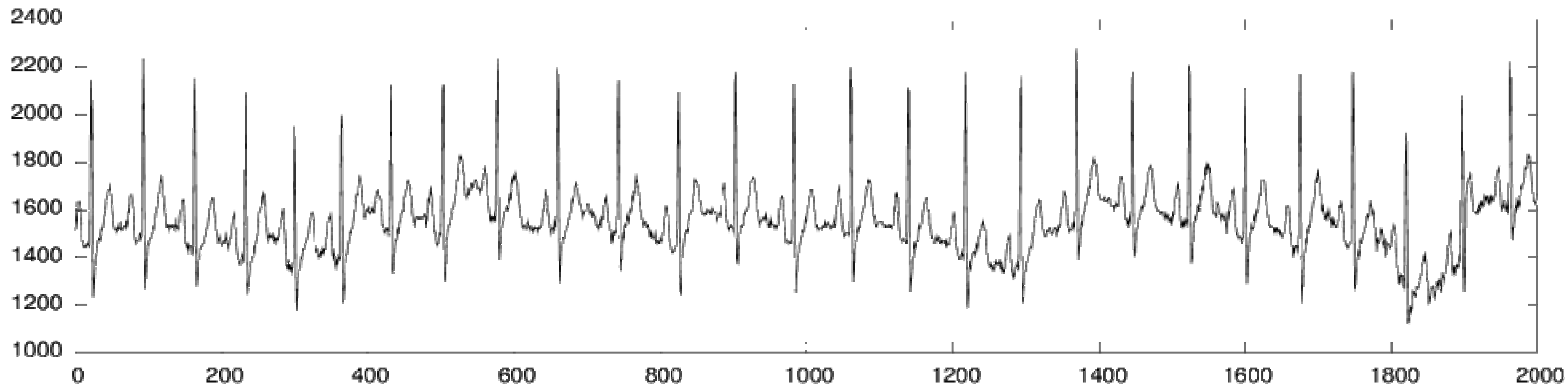


- Main adjustments to PT algorithm
  - 100Hz sample rate
  - Integer arithmetic
  - Low memory usage



Vitruvius project

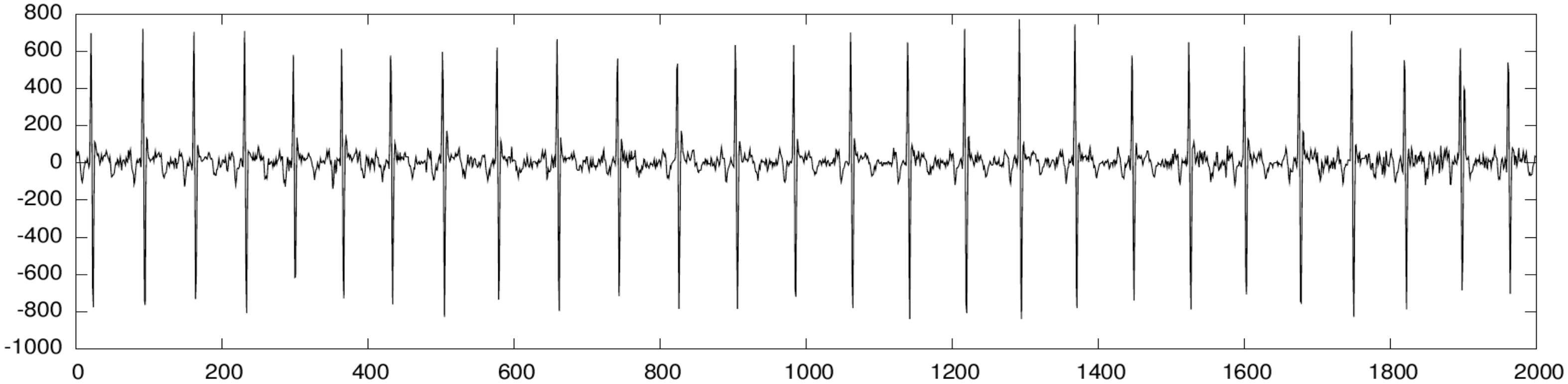
R-peak detection algorithm (1)



- Raw ECG signal *x(n),* sampled at 100Hz
- Apply bandpass filter
  - Remove 50Hz interference and baseline drift
  - Enhance R-peak, reduce P and T wave

$$y(n) = x(n) - x(n-2)$$

R-peak detection algorithm (2)



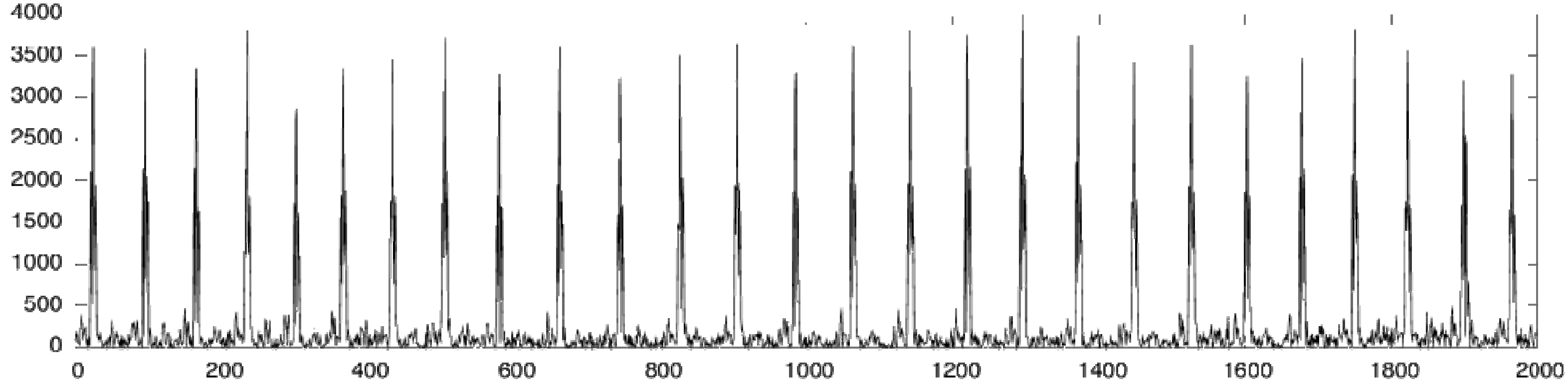
- Enhance R-peak with 1st and 2nd derivative

$$fd(n) = 2*y(n) + y(n-2) - y(n-3) - 2*y(n-4) \quad \text{(cf. PT)}$$

$$sd(n) = y(n) - 2*y(n-2) + y(n-4)$$

$$z(n) = | 3*y(n) - y(n-2) - y(n-3) - y(n-4) |$$

R-peak detection algorithm (3)

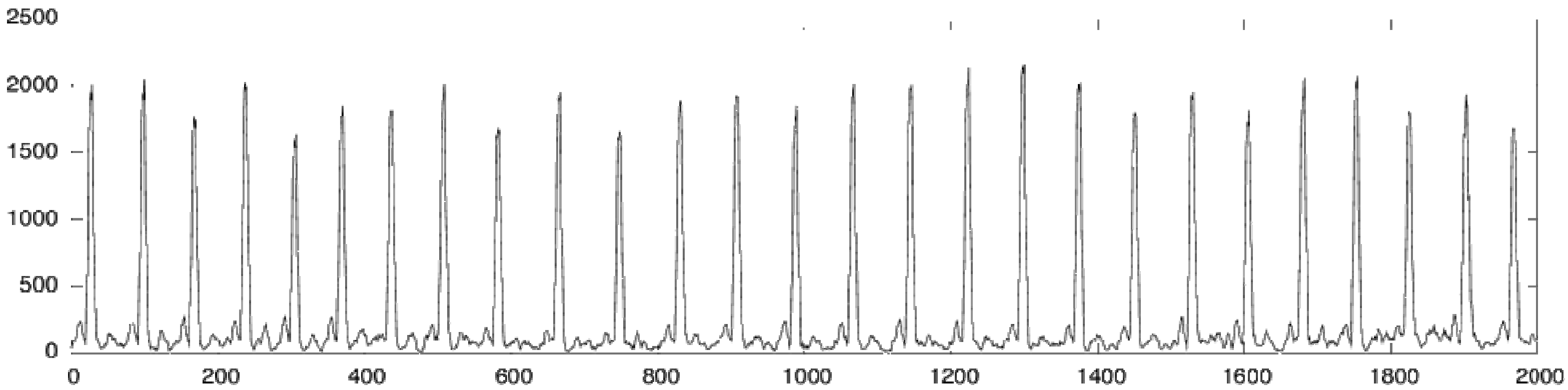


- Apply a running average filter with 6 taps

  final(n) = (z(n) + z(n-1) + ... + z(n-5)) / 6

Vitruvius project

R-peak detection algorithm (4)



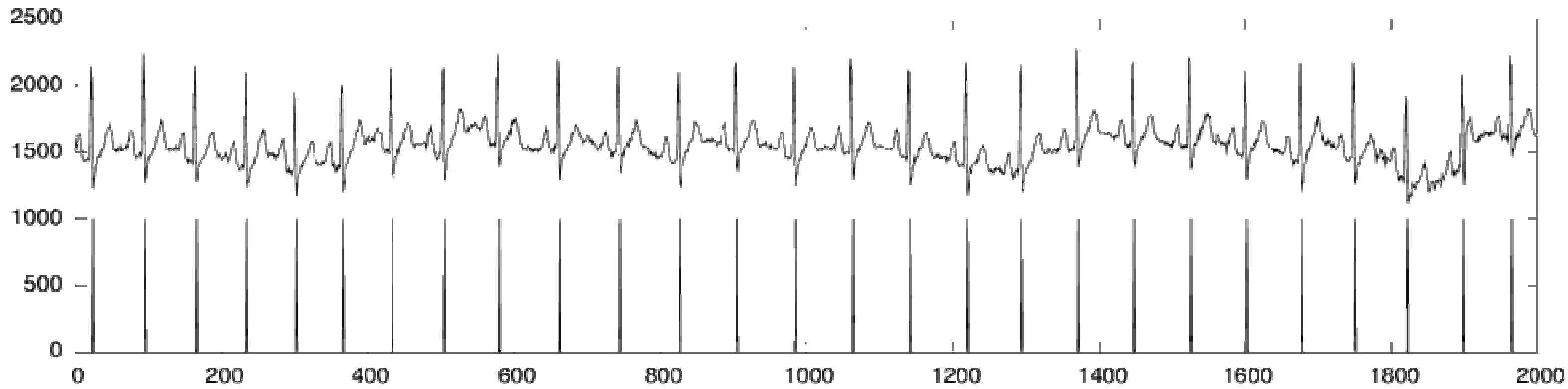
- Use threshold to detect the R-peak

$$IsPeak(n) = final(n) > Threshold \ \wedge \ \neg\exists_{1<i<20}\ IsPeak(n-i)$$

- Threshold is determined dynamically
  - Half of maximum of *final(n)* over G seconds

Vitruvius project

R-peak detection algorithm (5)



- Original signal and detected R-peaks
- Heart rate is derived from RR-interval
- Detect simple arrhythmias
  - Low and high heart rate
  - Missing heart beats

Vitruvius project

Quality of detection

- Use MIT-BIH Arrhythmia database
  - Standard reference for ECG analysis algorithms
  - 48 ECG records of 30 minutes each
    - Upper and lower signal sampled at 360Hz
  - Different patients and heart conditions
  - Annotated by experts

- Analysis
  - Down-sample upper signal from 360Hz to 90Hz
  - Run detection algorithm
  - Compare with annotations by experts

Quality of detection

- Terms
  - TP – true positive, correctly detected R-peaks
  - FP – false positive, incorrectly marked as R-peak
  - FN – false negative, R-peak which is not detected

- Sensitivity - TP / (TP+FN)
  - 98.80% average over all 48 records
  - 99.48% average over 45 records

- Positive predictability - TP / (TP+FP)
  - 99.75% average over all 48 records
  - 99.78% average over 45 records

Quality of detection

- Results comparable with reports in literature
  - Difficult due to selective analysis done by some papers
    - leaving out 'problematic' sequences

- Possible improvements
  - Adjust for 120Hz sampling and 60Hz noise
  - Different threshold adjustment

Energy model

- Hypothesis:
  energy is saved because on-node processing is much cheaper than communication

- Components of general energy model
  Energy() = Sense() + Process() + Communicate()

- Dependencies between these components
  - Input-output relation between components
  - Higher sample rate increases processing
  - Improved processing can reduce communication

Energy model

$$Sense() = Rate_{sample} \times Energy_{sample}$$

- $Energy_{sample}$ is hardware dependent
- Sensor can be turned off for low sample rates

$$Processing() = Rate_{sample} \times Energy_{filter} + Rate_{feature} \times Energy_{feature}$$

- Each sensor sample is analyzed by a filter
- Features are extracted and processed

Energy model

$$\text{Communication}() = \text{Energy}_{MAC} + \text{Rate}_{data} \times (1 + \text{Overhead} / \text{Payload}) \times \text{Energy}_{bit}$$

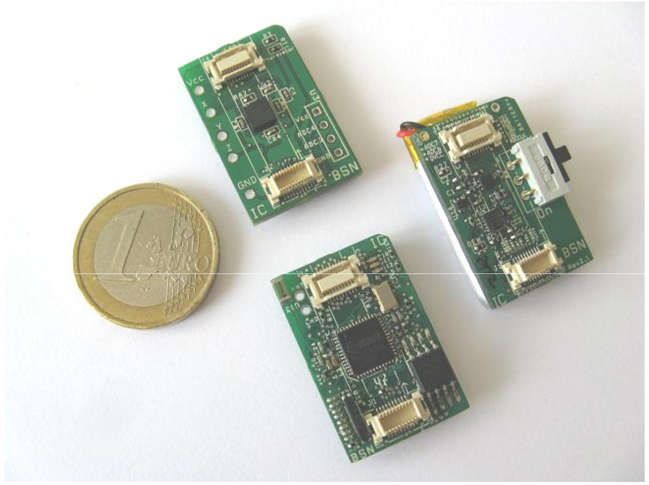
- *Energy*$_{MAC}$ includes periodic behavior of radio
- *Rate*$_{data}$ definies the bitrate of application data
- *Overhead* and *payload* are determined by the MAC protocol
- *Energy*$_{bit}$ is hardware specific

- The MAC protocol has an important effect
  - For some hardware, listening is more expensive than transmitting
  - Each MAC might have its own formula

Vitruvius project

Energy measurements

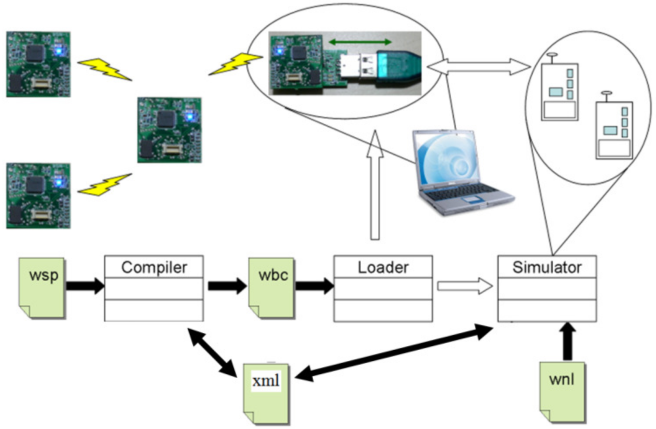
- Compare on-node processing with logging
- BSN v3 node from ICL
  - MSP430 16-bit microprocessor
  - 10KB RAM, 48 KB flash
  - CC2420 radio chipset
  - ECG sensor board



- Measure average current over 10 minutes

Implementation

- OSAS framework (Open Service Architecture for Sensors)
  - Macro-programming for sensor networks
    - One application describes the behavior of all nodes
    - Selective code upload over the air, dynamic
    - Based on periodic sampling and time-drive condition evaluation
    - Communication between nodes with active messsages
  - Run-time (interpreter) + OS combination
  - Integration with backend systems
    - Powerful device as simulated sensor node
  - Current application areas
    - Elderly / medical care, herd control, lighting



- TrawMAC protocol
  - supports traffic-aware behavior, i.e., adjusts to application-level communication frequency

Vitruvius project

Resource usage

- Operating system + OSAS framework
  - 1.5KB RAM, 27KB program flash
- R-peak detection algorithm
  - 80 bytes state, 336 bytes code
- Processor load
  - 16-bit integer arithmetic
  - Filters with few taps
  - Runs with interpreted byte code
    - Improvement possible by using machine code
    - ... however, processing contribution to total is small

Energy measurements

Contributions to the average current:

Sampling at 100Hz	1.618 mA
100Hz Logging (4 msg/sec)	12.477 mA
Heart rate (per 1 second)	4.425 mA
Heart rate (per 5 seconds)	2.375 mA
R-peak detection only	0.150 mA

- Improvement by factor 3 to 5

Energy Measurement

Energy() = Sense() + Process() + Communication()

where

Sense() = 1.618 mA

Process() = 0.150 mA

Communication() = 0.097 +
2.53 x TR +
0.00033 x Rate $_{data}$ *mA*

TR = transmission rate of the used *MAC*

The formula indicates

- Where energy can be saved
- When sensing becomes dominant

Conclusions

- R-peak detection on nodes is feasible
  - Algorithm performs well on reference data set
  - Low resource usage
- Energy improvement by factor 3 to 5
- Energy model constructed for the given application
- Future work
  - More measurements to improve the model
  - (even) better MAC protocol

Vitruvius project