VitalJacket® SDK

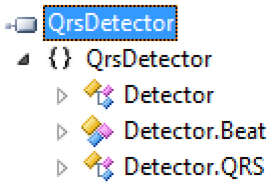
VitalJacket SDK v1.0.07 – QRS detector

LEGAL NOTICE AND DISCLAIMER

ATENTION: Although VitalJacket is a certified medical device, its developer version is NOT certified for diagnosis usage. It is intended for R&D and development purposes only. Users of VJ SDK can submit their final developments to medical certification. All contents of our product are compliant with the European Medical Device directive 93/42/EEC but, being a developer's version, it's not certified.

Index:

QRS detector


QrsDetector
{} QrsDetector
Detector
Detector.Beat
Detector.QRS

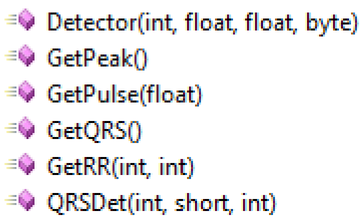
public class **Detector**
Member of **QrsDetector**

Summary:
QRS detector. The QRS detector is based on the algorithm of Pan and Tompkins [1] and was used MIT-BIH database for validate results [2].

public class **QRS**
Member of **QrsDetector.Detector**

Summary:
Class: QRS.

Class: Detector QRS



- Detector(int, float, float, byte)
- GetPeak()
- GetPulse(float)
- GetQRS()
- GetRR(int, int)
- QRSDet(int, short, int)

public **Detector**(int *sampleFrequency*, **float** *thPeakAmplitude*, **float** *th*, **byte** *windowSize*)
Member of **QrsDetector**.**Detector**

Summary:
Construtor.

Parameters:
sampleFrequency: Sample frequency (Hz)
thPeakAmplitude: Threshold of peak amplitude
th: Threshold for detect noise.
windowSize: Number of QRS for calculate mean of Heart Rate

public **short** **QRSDet**(**int** *datum*, **short** *init*, **int** *sampleCount*)
Member of **QrsDetector**.**Detector**

Summary:
Detect QRS.

Parameters:
datum: ecg sample
init: '1' for init QRS detector and '0' for detect QRS
ind: number of ecg samples

public **QrsDetector.Detector.QRS** **GetQRS**()
Member of **QrsDetector**.**Detector**

Summary:
Get last QRS detected.

Returns:
QRS detected

public QrsDetector.Detector.Beat **GetPeak**()
Member of QrsDetector.Detector

Summary:
Get last peak detected.

public static float **GetPulse**(float *rr*)
Member of QrsDetector.Detector

Summary:
Calculate bpm instantaneous (bpm).

Parameters:
rr: R-R (ms)

Returns:
value bpmi

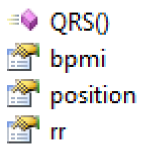
public static float **GetRR**(int *rr*, int *SampleFrequency*)
Member of QrsDetector.Detector

Summary:
Calculate R-R (ms).

Parameters:
rr: R-R in samples
SampleFrequency: Sample frequency (Hz)

Returns:
R-R (ms)

Class: QRS info


QRS()
bpmi
position
rr

public **int bpmi** { set; get; } Member of **QrsDetector.Detector.QRS** **Summary:** Bpmi (bpm).

public **int position** { set; get; } Member of **QrsDetector.Detector.QRS** **Summary:** Peak position (samples).

public **int rr** { set; get; } Member of **QrsDetector.Detector.QRS** **Summary:** R-R (ms).

Sample Code

1. Add Reference QrsDetector.dll to your project;
2. See sample code:

```
private int sampleCount = 0, windowCounter = 0;
private int sampleFrequency = 500;
private Queue<int> pulseQueue = new Queue<int>();
private int oldPeak = 0;
private float pulseValue = 0;
private Detector detector;

private void InitQrsDetector()
{
    detector = new Detector(500, 7.0f, 0.3125f, 8);
}

/// <summary>
/// Detect QRS.
/// </summary>
/// <param name="dataByte">ecg sample</param>
private void QrsDetector(short dataByte)
{
    try
    {
        sampleCount++;

        if (windowCounter > sampleFrequency * 2)
        {
            while (pulseQueue.Count > 5)
                pulseQueue.Dequeue();

            int[] cont = pulseQueue.ToArray();
            float mean = (float)(Mean(cont, cont.Length) + 0.5);
            pulseValue = mean;

            windowCounter = 0;
        }
        else
        {
            windowCounter++;
        }

        // Detect QRS
        int delay = detector.QRSDet(dataByte * 10, 0, sampleCount);
        if (delay != 0)
        {
            int DetectionTime = sampleCount - delay;

            // R-R (in samples)
            long rr = DetectionTime - oldPeak;

            // Calculate R-R (ms) and heart rate instantaneus (bpm)
            Detector.QRS qrs = new Detector.QRS();
            qrs.position = DetectionTime;
            qrs.rr = (short)Detector.GetRR((int)rr, sampleFrequency);
            qrs.bpmi = (short)Detector.GetPulse(qrs.rr);

            if (qrs.bpmi > 20 && qrs.bpmi < 255)
                pulseQueue.Enqueue(qrs.bpmi);

            oldPeak = DetectionTime;
        }
    }
    catch (Exception)
    {
        Throw;
    }
}
```

```
/// <summary>
/// Calculate mean of array values.
/// </summary>
/// <param name="value"></param>
/// <param name="nvalues"></param>
/// <returns></returns>
private float Mean(int[] value, int nvalues)
{
    float sum = 0;
    float nval = 0;

    for (int i = 0; i < nvalues; i++)
    {
        sum += value[i];

        if (value[i] > 0)
            nval++;
    }

    if (nval > 0)
        sum /= nval;

    return sum;
}
```

References

[1] Pan J and Tompkins WJ. A Real-Time QRS Detection Algorithm. IEEE Transactions on Biomedical Engineering 32(3):230-236, 1985

[2] MIT-BIH Arrhythmia Database: http://www.physionet.org/physiobank/database/mitdb/

Control versions

Version	Date	Change log
1.0.02	30-04-2013	*Get device Id*
		Send radio event to device
1.0.03	19-07-2013	*New method to send radio-event to device*
		A new too l(InfoExporter.exe) for export data to Excel and Matlab