APPLICATION NOTE

SCA11H ADAPTIVE CALIBRATION

Introduction

This document describes a method to adaptively calibrate occupied parameters for SCA11H BCG sensor node, based on recorded BCG data. The method is described based on Python example code which is included at the end of this document.

A proper empty bed calibration is required, for example, by using embedded calibration routine phase 1 or by measuring empty bed signal strength for a period of time (for example 1 min or longer) and then setting var_level_2 to a slightly higher level.

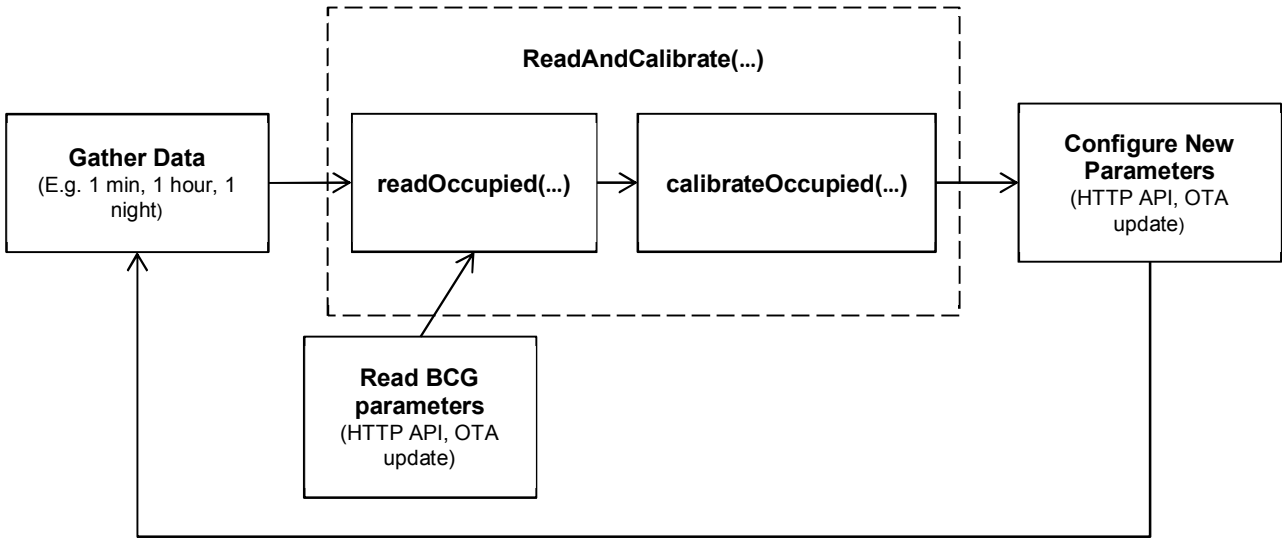

Figure 1. Flowchart of adaptive calibration

Gather data

Record BCG algorithm data over time. This BCG algorithm data can include both empty bed and occupied bed data. Generally, the longer the recording the better it represents the real use, thus for best results over night recording is recommended.
Example BCG algorithm data:

```
4,61,9,37,78,1884,1,975,0,0
5,60,9,40,75,1542,1,1045,0,0
```

Read and configure BCG parameters

Old BCG calibration parameters need to be read from the SCA11H BCG sensor node for best results. This can be done by using HTTP API or cloud OTA update methods. Also, with these methods the new calibration parameters can be configured to the SCA11H.

Find more details in

"Product Specification 1424 SCA11H Hostless WLAN HTTP API Specification"
"Product Specification 1325 SCA11H cloud server interface specification"

readAndCalibrate(…)

Combines readOccupied and occupiedCalibration functions

readOccupied(…)

Read BCG data from a list of data, and check validity of signal strengths and stroke volumes.

calibrateOccupied(…)

Main occupied calibration script.

Calculates new parameters:

var_level_1: calculated based on measured signal strengths,
signal_range: calculated based on measures stroke volumes,
stroke vol: calculated based on new signal_range

Example Python code

```
# Python 3.x (tested with Python 3.4.2)

from statistics import median
import os
import sys

# Always used parameters
EMPTY_LEVEL = 270                       # Empty bed signal strength
MOVEMENT_LEVEL = 10000                  # Maximum signal strength before "movement"
MIN_VAR_LEVEL_1 = 5000                  # Minimum possible outputted var_level_1.
                                        #   Parameter range: 3000...10000
VAR1_MULTIPLIER = 10                    # Determines multiplies median of signal range for var_level_1
                                        #   Parameter range: 5...15.
SIGNAL_RANGE_MULTIPLIER = 40            # Used to set signal range and stroke vol to correct levels.
                                        #   Parameter range: 40...50.
                                        #   - If too low noise could be picked up as heart beats
                                        #   - If too high heart beats could be missed
STROKE_VOL_MULTIPLIER = 2.6             # Sets stroke vol to right level based on signal_range.
                                        #   Parameter range: 2.5...3.0.
MIN_LINES = 15                          # Minimum number of rows of data has to be accepted to calibrate.

# Only used if old parameters are too low/high
TOO_HIGH_PARAMETERS_MULTIPLIER = 0.75   # Multiplier for to decrease stroke_vol if too high parameters detected.
                                        #   Parameter range: 0...1.
TOO_LOW_PARAMETERS_ADDITION = 4000      # Amount to increase stroke_vol if too low parameters detected.
                                        #   Parameter range: 2000...5000
ACCEPTED_RATIO_HIGH = 0.5               # Ratio of how many rows of data have to have stroke volume > 0
                                        #   Parameter range: 0...1. Has to be higher than ACCEPTED_RATIO_LOW
                                        #   Used to check if parameters are too high
ACCEPTED_RATIO_LOW = 0.75               # Ratio of how many rows of data have to have stroke volume > 0
                                        #   Parameter range: 0...1. Has to be smaller than ACCEPTED_RATIO_HIGH
                                        #   Used to check if parameters are too low
SV_DIFF_CHANGE = 0.25                   # Minimum stroke volume difference on average is required.
                                        #   Parameter range: 0.2...1
                                        #   Used to check if parameters are too low


def openFile():
    # Open file
    print('Input BCG data file')
    filename = str(input("Enter file BCG data file: "))
    fid = open(filename, 'r')
    return fid

def readDataToList(source):
    dataLine = source.readline()
    dataList = []
    while dataLine:
        dataLine = dataLine.strip()
        dataList.append(dataLine)
        dataLine = source.readline()
    return dataList

def readOccupied(dataList):
    signalStrengths = []
    strokeVolumes = []
    readLines = 0
    acceptedLines = 0
    svDiff = 0
    for row in dataList:
        row = row.split(',')
        if len(row) == 10:  # One row of BCG data has 10 values
            row = list(map(int,row))
        else:
            continue
        # Signal Strength needs to be between EMPTY_LEVEL and MOVEMENT_LEVEL
        signalStrengthIsOk = row[5] >= EMPTY_LEVEL and row[5] <= MOVEMENT_LEVEL
        if signalStrengthIsOk:
            readLines += 1;                 # Amount of lines where signal strength is ok
            strokeVolumeIsOk = row[3] > 0   # Stroke volume needs to be higher than 0
            if strokeVolumeIsOk:
                acceptedLines += 1          # Amount of lines where signal strength was ok and stroke volume is ok
                signalStrengths.append(row[5])
                strokeVolumes.append(row[3])
                if acceptedLines > 1:
                    # Calculate sum of differences, used if old are parameters too low
                    svDiff += abs(strokeVolumes[-1] - strokeVolumes[-2])
    return signalStrengths, strokeVolumes, readLines, acceptedLines, svDiff
```

```
def calibrateOccupied(oldParameters, signalStrengths, strokeVolumes, readLines, acceptedLines, svDiff):
    calcSS = median(signalStrengths)
    calcSV = median(strokeVolumes)
    acceptance = acceptedLines / readLines
    # Make sure at least MIN_LINES of rows of data with proper signal strength and stroke volume.
    if acceptedLines < MIN_LINES:
        print('Warning: Not enough occupied data. Check signal strengths.')
        parameters = oldParameters
    # Check if not enough lines have stroke volume but has signal strength
    elif acceptance < ACCEPTED_RATIO_HIGH:
        print('Warning: Old parameters likely too high. Decreasing parameters.')
        var_level_1 = MOVEMENT_LEVEL
        stroke_vol = round(oldParameters[3] * TOO_HIGH_PARAMETERS_MULTIPLIER)
        signal_range = round(stroke_vol / STROKE_VOL_MULTIPLIER)
    # Check if not enough lines have stroke volume but has signal strength and stroke volume variability is too low
    elif acceptance < ACCEPTED_RATIO_LOW and svDiff / acceptedLines < SV_DIFF_CHANGE:
        print('Warning: Old parameters likely too low. Increasing parameters.')
        var_level_1 = MOVEMENT_LEVEL
        stroke_vol = oldParameters(3) + TOO_LOW_PARAMETERS_ADDITION;
        signal_range = round(stroke_vol / STROKE_VOL_MULTIPLIER)
   # Main part of adaptive calibration.
   # Calculates new var_level_1 based on median of signal strength,
   #               signal range based on median of stroke volumes and
   #               stroke vol (parameter) based on signal range
    else:
        var_level_1 = max(MIN_VAR_LEVEL_1, VAR1_MULTIPLIER * calcSS)
        signal_range = calcSV * SIGNAL_RANGE_MULTIPLIER
        stroke_vol = round(STROKE_VOL_MULTIPLIER * signal_range)
    parameters = list(map(int,
[var_level_1, oldParameters[1], stroke_vol, oldParameters[3], signal_range, oldParameters[5]]))
    return parameters

def readAndCalibrate(dataList, oldParameters):
    signalStrengths, strokeVolumes, readLines, acceptedLines, svDiff = readOccupied(dataList)
    parameters = calibrateOccupied(oldParameters, signalStrengths, strokeVolumes, readLines, acceptedLines, svDiff)
    return parameters

def main():
    try:
        oldParameters = [7000,EMPTY_LEVEL,5000,0,1500,7]    # Default, old parameters should be read from BCG
        print("Adaptive calibration example.")
        source = openFile()
        dataList = readDataToList(source)
        parameters = readAndCalibrate(dataList, oldParameters)
        print(parameters)
    except:
        print("Unexpected error:", sys.exc_info()[0])
    finally:
        os.system("pause")  # Win only! Can be skipped

if __name__ == '__main__':
    main()
```

Rev.	Date	Change Description
1	17-May-16	First version of document.