OB1203

Pulse Oximeter Algorithm for SpO2, Heart Rate and Respiration Rate

This application note introduces the OB1203 SpO2 (saturation of peripheral blood oxygen) and heart rate detection algorithm including respiration rate (RR) detection.

Target Device

Hardware is the OB1203SD-RL-EVK Heart Rate, SpO2, and Respiration Rate Evaluation Kit with the RL78/G13 (R5F100BG) 16-bit microcontroller and OLED Display. Code is in C. Build environment is E2 Studio, with a CC-RL compiler.

Note: The algorithm is provided for reference, demonstration, and/or evaluation. No performance is guaranteed or warranted, nor is fitness or suitability for any medical device application claimed. Renesas will not indemnify customers using any part of the provided reference code. OEM is responsible for performance of products made using any of the provided reference code.

Contents

1. Algorithm Overview

1.1 Main Operation Loop

After the start button is pressed, the application starts in a low-power proximity sensing mode. When a finger is detected, the proximity sensor asserts the interrupt pin (active low). The application software then switches over to PPG (photoplethysmography – light sensing of blood in tissue), adjusts the LED intensities with an autogain control (AGC) to get within a specified target range of a setpoint, and then begins buffering data. A few seconds of data are buffered before the algorithm runs for the first time. Thereafter the algorithm runs every INTERVAL number of samples.

For respiration rate (RR) measurements `INTERVAL` is 40 (0.4sec). For heart rate (HR) and peripheral oxygen saturation (SpO2) only, INTERVAL can be 100 (1sec). From the measured data, HR and SpO2 are estimated and waveform characteristics are measured to calculate respiratory rate (RR). Respiratory rate algorithm runs on about 30 seconds of buffered HR data. It is very sensitive to motion and is only indicated for stationary subjects with good finger stability and/or with a finger clip and no motion.

The sensor is typically covered with a cover glass or surrounded by a finger support to improve stability and data quality. If the signal quality leaves a wider bound, then data collection stops and the application returns to automatic gain control (AGC) mode to re-adjust the LED current levels. If a persistent low level is reached with the LEDs maxed out, then the application reverts to proximity sensing mode.

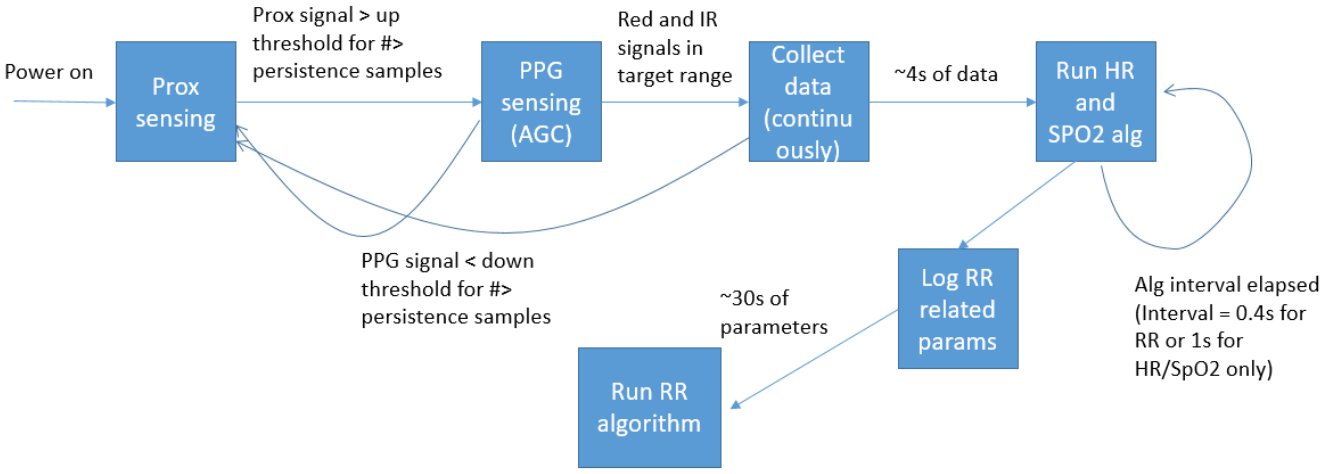

Figure 1. Main Application Operation

In the RL78 demonstration code, the following folders/libraries are provided:

- SPO2 – This code contains the heartrate, SpO2, and respiratory rate algorithms. The user will call `spo2_init()` before starting measurements and then call `add_sample()` after getting data from the OB1203. Each interval the main application must call *do_algorithm_part1()*, *do_algorithm_part2()*, `do_algorithm_part3()` and `do_algorithm_part4()`. Parts 3 and 4 are only needed if RR is calculated. Note, samples are read from the OB1203 FIFO between parts 1 and 2 and parts 2 and 3 to avoid missing samples.
- KALMAN – This code contains an outlier filter routine, referencing an array of structs where each struct contains the parameters of a different filter. There are four filters in the struct array: correlation, heartrate, spo2, and respiratory rate.
- OB1203 – This code contains the routines needed to set up OB1203 and read data.

1.2 Timing

In biosensing mode, an auto gain control (AGC) loop adjusts the LED intensity to achieve a desired signal level, about 80% of full scale. This is necessary to get high signal-to-noise (SNR) from the LED drivers and analog

digital converter. Data is read after every ADC conversion result (PPG interrupt). This is called “slow mode,” where the LED current is adjusted after each new data. Once the infrared and red signal levels are both in range, the device is switched to “fast mode” where 10 samples each of red and infrared (IR) are read out from the OB1203 data FIFO every 100ms. Algorithm steps and display updates are done between sample measurements.

The main application runs in `oximeter.c`. It sets up the OB1203 in proximity mode and then waits for an interrupt indicating a finger is present. It then calls `switch_mode()` to change to HR/SpO2 sensing PPG mode. During AGC, an PPG interrupt is used to alert the microcontroller (MCU) after each sample is acquired (so-called “fast mode”). Once IR and Red channels are both in range, operation switches to “slow mode” to read out ten samples from the FIFO each time the FIFO_A_FULL interrupt is asserted. In this mode the following sequence, depicted schematically in Figure 2, is followed:

1. Wait for interrupt. Read 10 samples each of IR and red.
2. `do_algorithm_part1()`
3. Wait for interrupt. Read 10 samples each of IR and red.
4. `do_algorithm_part2()`
5. `do_algorithm_part3()`
6. `do_algorithm_part4()`
7. Wait for interrupt. Read 10 samples each of IR and red.
8. Update the display.
9. Until `INTERVAL` expires: Wait for interrupt. Read 10 samples each of IR and red (i.e., for `INTERVAL` 40, read samples once more).

If PPG wave display is enabled in `oximeter.c`, the PPG waveform portion of the display is updated after the fourth sample measurements.

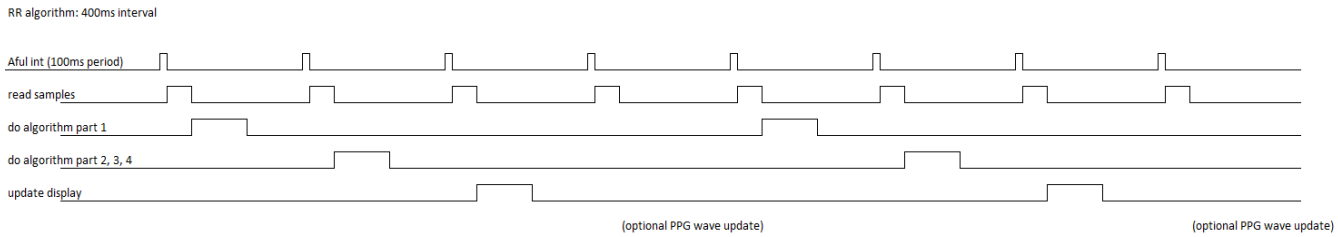

Figure 2. Sample Collection and Algorithm Timing Diagram

Recommended operation parameters for best SNR are 1600Hz sample rate for both IR and Red, with 16 on-chip averages for an output data rate of 100Hz, a setpoint value near 200,000 ADC counts, and FIFO_A_FULL interrupt-based readout of the FIFO. For more information, see Example Code.

1.3 Main Program Settings Configuration

In oximeter.c there are several user-configurable settings that are set with #define lines in the C code:

1. Operation mode. Choose between three options:
   - `UPDATE_DISPLAY` (normal operation) and debug options.
   - `PRINT_RAW` data via UART at baud rate of 230400 on RL78/G13 pin P13.
   - `DATA_FROM_UART` which uses a python script on the PC to send raw data to the evaluation kit and receive algorithm results. `DATA_FROM_UART` is useful for algorithm verification of special test cases like high heart rate, low perfusion, and metered breathing rate data and hypoxia tests.

   In `UPDATE_DISPLAY` mode, algorithm results are streamed from the UART Tx pin (RL78/G13 pin P13) for logging.

Instructions for these modes are found in the file.

2. Respiratory rate or not.

   Choose to run the algorithm every 0.4 seconds for respiratory rate (RR) calculation, or once every second for SpO2 and HR only calculation. There is a modest (~30%) MCU power consumption savings to not run RR. But most of the power is from OB1203, so this is not a large knob to turn.

3. Select power option.

   The OB1203 internal sample rate determines SNR and power consumption. The demo uses the maximum sample rate (highest power) by default. Other power options are also available for example purposes.

4. Select mode of operation.

   Generally, use `NO_TEST_MODE`. Test mode can be enabled for calibration purposes.

5. Select display mode.

   Define `SHOW_PPG_WAVE` to show the actual PPG waveform. This increases MCU power consumption. Otherwise, a blinking dot will illustrate the current heart rate.

6. Define `CHECK_PI`.

   Define `CHECK_PI` to put in more strict checks for data quality based on the measurement perfusion index (PI). For instance, if PI is too low the SpO2 output will freeze at the previous (valid) reading until the PI value increase and a valid reading is obtain. If too many low quality readings occur the algorithm will be reset. This mode also enables a descriptive comment in the UART algorithm output. Comment out `CHECK_PI` for debugging.

7. Define `SPO2_DEBUG`.

   Define `SPO2_DEBUG` when doing calibration of devices in a hypoxia lab to get SpO2 measurements that are not clipped to 100% and 60% respectively for the maximum and minimum.

1.4 Algorithm Flow

The OB1203 demo algorithm runs two complementary algorithms in parallel, as displayed in the following figure.

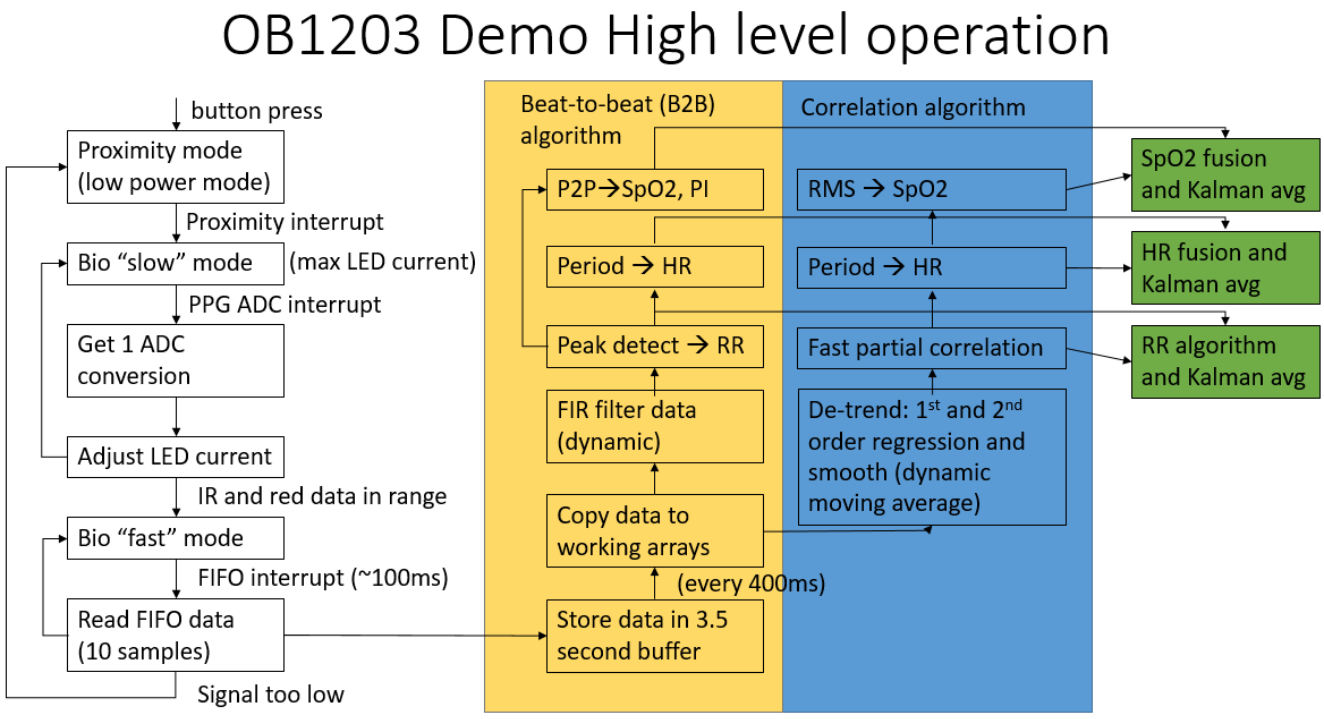

Figure 3. High-Level Algorithm Flow

The correlation algorithm excels when data quality is poor (low perfusion), but the method is susceptible to detecting subharmonics (too low of heart rate). The beat-to-beat algorithm has the highest accuracy but is susceptible to second harmonics and requires slightly better SNR (perfusion). All algorithm results use a zero-order Kalman type filter for removing outliers. The fusion method chooses the most consistent result.

Both algorithms are typically more computationally intensive than a fast Fourier transform (FFT), but they can handle more cases and respond more quickly. They are, however, far less computationally intensive than methods requiring floating point like wavelet transforms. The demo shows how to do both methods efficiently on a general-purpose 16-bit microcontroller, namely the RL78/G13. The demo code requires about 8-9kB of RAM and ~64kB of flash.

2. Algorithm Description

The algorithms implemented in the demo use tracking filters. The current estimate of heart rate is used to set a filter passband. Accurate when “on,” this method can also lock into a subharmonic, or second harmonic if care is not taken to “break out” when this occurs.

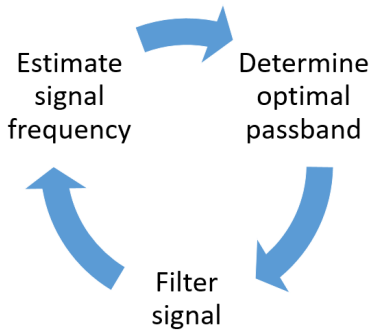

Figure 4. Tracking Filter Operation Principle

The solution implemented in the demo is to fuse multiple algorithms with complementary susceptibilities and benefits. Additionally, a zero-order Kalman (outlier) filters will reset the algorithm if too many outliers occur in a row, allowing the filter to re-acquire and track the correct frequency.

2.1 Correlation Method for Heart Rate

The correlation method scans the signal past itself to look for periodicity. The full autocorrelation of a 148 sample length signal requires 148^2 or about 22,000 multiply and add operations. Therefore, we implement a fast-partial correlation method described in detail below. This results in about 20x fewer multiplications, for a dramatic speed-up.

The correlation performs best on data that has been “detrended” or flattened. Breathing, changes in finger perfusion as finger/clip pressure pushes blood out, and motion introduce slow oscillations or drift in the signal. Much of this “bending” of the PPG signal is removed in the algorithm with DC removal, slope removal, and curvature removal (aka, second order regression), on the data in the 3.5 second buffer. This “detrending” is performed on the entire buffer and is the bulk of the correlation computational load of the correlation method. The root mean square (RMS) of the data in the detrended buffer is then calculated as an estimate of the AC or pulsile signal. This is described in the SpO2 algorithm.

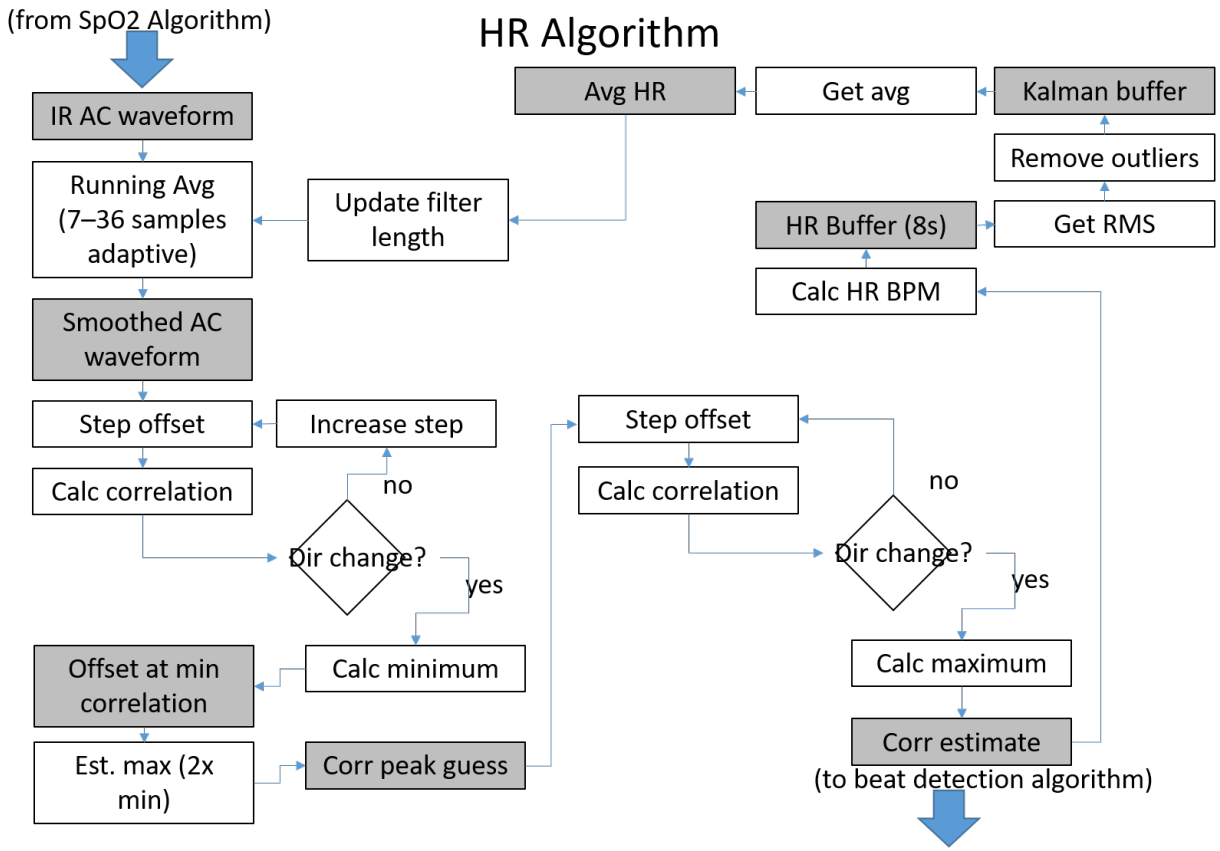

Figure 5. Simplified Correlation Method Heart Rate Algorithm Flow

2.1.1. Correlation Method Details and Limitations

In the correlation method, additional filtering is necessary to avoid detecting false peaks that show up as a result of the periodicity of the dicrotic notch.

Figure 6 shows the detrended FIFO butter data (blue) after DC and slope removal (AC data) of the IR channel, and the AC data with an additional moving average of eight samples applied (red). Another suitable method for smoothing is a Savitsky-Golay (SG) filter, which also yields the smoothed derivative. After initial smoothing, a secondary peak is often visible in the PPG data. Figure 6 shows a raw data waveform after detrending. This data has not been inverted as is customary in PPG displays, so the systole is a valley and the systole is the peak immediately prior. In the smoothed data a secondary peak after the systole is observed. This secondary peak is called the dicrotic notch (peak→notch in the inverted data waveform).

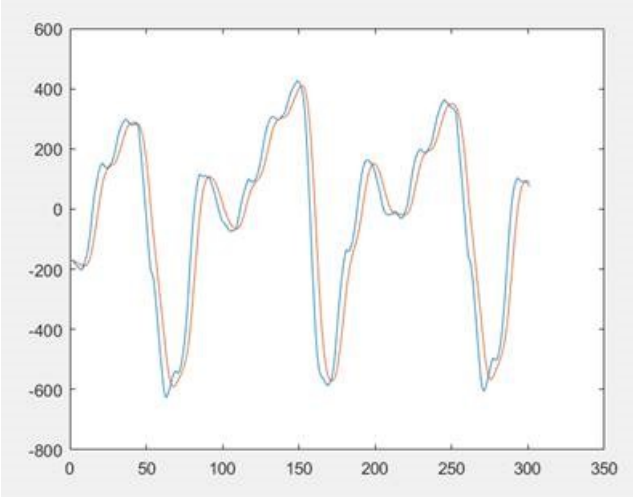

Figure 6. PPG AC Waveform with Moving Average Filter

If care is not taken, the dicrotic notch or any noise peak can be erroneously labeled as a heartbeat.

To reduce the influence of noise the algorithm performs a partial “smart” autocorrelation, multiplying the elements from 150 samples with another 150 samples with a time offset. When the two signals line up after one heart beat the dot product is maximal. Halfway between, the two waves are like out of phase sine waves and their correlation has a minimum.

Figure 7 shows an autocorrelation function of the red curve in Figure 4. A “false” peak that appears when the systole peak of one PPG waveform aligns with the small peak after the dicrotic notch of the next wave. A symptom of detection of dicrotic notches results in heart rate that is about 1.4 times too high.

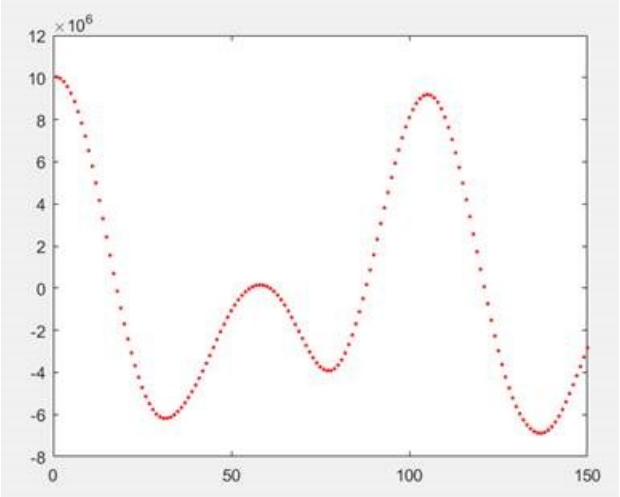

Figure 7. Autocorrelation of PPG Signal Showing False Peak from Dicrotic Notch

To remove that peak, an additional smoothing operation (running average) is applied to the data. Figure 8 plots the data after the 32 sample running average is applied. The correlation skips the first few samples where the filter is initialing, and the phase delay is not constant to avoid skewing a peak position. The algorithm implements an adaptive filter, adjusting the moving average to filter out the dicrotic notch, but not over-filter and suppress peaks. Tuning parameters are available to users in `spo2.h`. Notice the dicrotic notch is removed after averaging and the minimum occurs halfway between the zero order and first order correlation peaks.

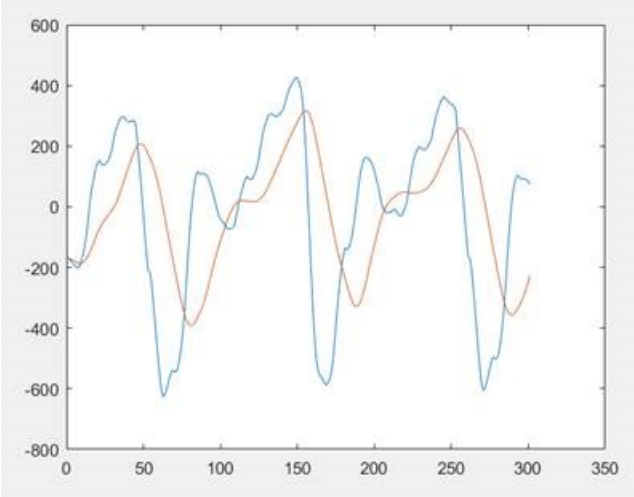

Figure 8. PPG AC Waveform (Blue) and Smoothed with 32 Sample Moving Average (Red)

2.1.2. Partial Autocorrelation Peak Search

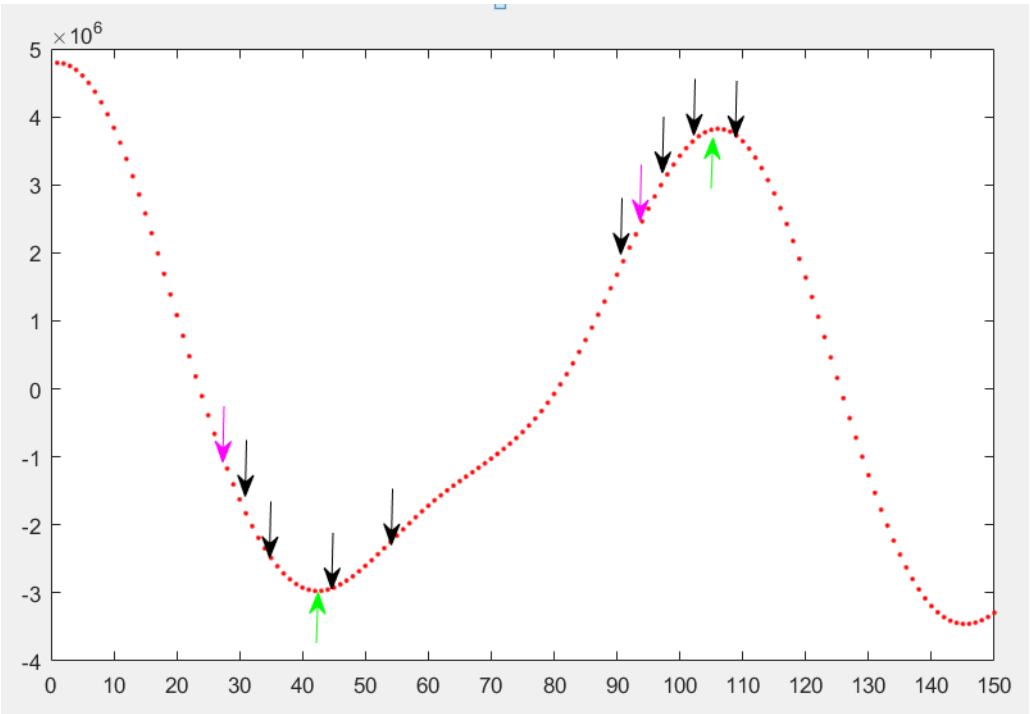

Figure 9: Autocorrelation of Smoothed AC Data and Correlation Samples

Rather than calculating the entire autocorrelation which would be processor intensive, the correlation peak search algorithm (Figure 9) starts by calculating a correlation just below the shortest allowed offset or maximum hear rate (left-side purple arrow). Using increasing steps (left side black arrows), it either seeks a minimum or maximum. The peak or valley position (left-side green arrow) is interpolated using a quadratic method. For a minimum, twice the offset is used for the start of the main peak fine search (right-side purple arrow). For a maximum, the offset nearest the peak is used for a fine search. Using small steps that scale with the start offset value of the fine search, the fine search looks to lower offsets first, then to higher offsets (right-side black arrows). When a peak is detected its position is interpolated using a quadratic method (right-side green arrow). The offset of the peak is calculated and used to estimate the heart rate. This method runs in integer math with three additional bits of fixed precision.

Peaks are typically found in eight correlations or less. In theory, the peak measured by the previous run of the algorithm can be used as a start position for the following search. This is useful for devices with very limited processing power. The disadvantage is it is possible to lock into a subharmonic or dicrotic notch peak. If processing power is available to do a blind search, this is preferred. In fact, if no peak is found using the 2 × minimum search method then the previous peak is used as a start position.

An alternative method suitable for devices with an FFT library or hardware block is to use the fast-discrete Fourier transform to find the highest frequency peak, again, using appropriate filtering to avoid getting a peak from the dicrotic notch. Care should be taken to avoid selecting the breathing rate as the heart rate if the FFT is used. Renesas provides an efficient FFT for the RL78, and example code is available upon request.

Over-filtering can cause the peak to shift to lower frequencies. To avoid this, the algorithm reduces the length of the moving average filter as a function of the measured heart rate values.

Note, the “quality” of the data can be determined by the ratio of the correlation peak to the self-correlation. Essentially, the more repeatable the signal, the more perfect the correlation. An alternate metric for detecting poor signal quality (not implemented in the example code) is “skewness”.

2.2 SpO2 Algorithm

Reflective PPG is measured against the skin with emitter and receiver on the same side, similar to an active infrared (IR) proximity sensor. Some PPG signal derives from the change in blood volume in the tissue, increasing absorption as the vessels upstream of the capillaries fill after the heartbeat. Another portion of the signal derives from agglomeration of cells at low blood flow rates and disaggregation at higher flow rates.

Reflective PPG measurement, while convenient in mobile devices, is not optimal for individuals with very soft fingers including young children and some elderly individuals.

The observed signal has a large static or DC component on which the small AC cardiovascular signal appears. The ratio of AC/DC is called perfusion index (PI).

$$PI = \frac{IR_{AC}}{IR_{DC}} * 100\%$$

Perfusion index increases when there is more blood “perfused” into the tissue. That actual number used for PI depends on several details, including wavelength and LED/receiver configuration.

For transmissive systems shining light through the finger, transmission perfusion index (tPI) is on the order of several percent, meaning the AC ripple is about 10-100 times smaller than the DC signal – if it was larger you could see your own blood pulsation in the tissue with your eye. When measuring with the light source and detector from the same side of the finger (reflective PPG), we get a shorter absorption path length so the reflective perfusion index (rPI) can be 5-10x smaller than tPI and is not equivalent to tPI. Since different tissue regions are measured using a reflective PPG compared to a transmissive PPG, even if the rPI result is scaled, it will not match tPI. Since reflective PPG has a smaller sample region, this places increased burdens on the hardware dynamic range, sensitivity, and algorithm performance. However, with suitable hardware and algorithms, overall reduction in system complexity with an integrated solution can be achieved.

2.2.1. SpO2 Mechanism

The measurement of the saturation of blood by oxygen is accomplished with PPG measurements at two wavelengths, taking advantage of the differential absorption of oxyhemoglobin and deoxyhemoglobin. The most common wavelengths are red and IR, due to the strong differential absorption, availability of light sources, relatively flat spectra (for wavelength shift tolerance).

The blood Oxygen saturation level or SpO2 level is calculated from a calibrate curve plotting SpO2 versus the “ratio of ratios” called the “R curve,” which is done by calibrating against a known instrument or blood test reference in a hypoxia lab using breathe-down testing. This calibration plot, also known as the “R curve”, depends on the mechanical details of a PPG system.

The “ratio of ratios” considers the variation in signal amplitude from person to person in the following way. The DC component has PPG-irrelevant information such as surface reflections and skin tone. However, it does set the scale for the AC signal due to hemodynamics of interest. Therefore, the red channel AC signal (measured by high-pass filtering the data) is normalized by the red DC signal. The IR channel AC signal is normalized by the IR channel DC signal and the ratio of those two normalized values contains information unique to the blood oxygen saturation level for this particular system. If anything in the mechanical or optical design changes (e.g., different enclosure color, shape, or materials), the calibration procedure must be redone.

SpO2 is calculated from the RMS or a peak-to-peak value in the following way. The ratio of ratios “R” is given by:

$$R = \frac{R_{AC}/R_{DC}}{IR_{AC}/IR_{DC}}$$

SpO2 is calculated with positive coefficients (in a linear approximation) as:

$$Spo2 = (c_0 - c_1 R) * 100\%$$

Higher order terms or a look-up table can be used. Physiologically, less oxygen means a larger red AC signal.

SpO2 requires a much higher SNR than heart rate because we are not merely trying to detect the heartbeat which is a small ripple on a big background, we are trying to quantify the size of that ripple and divide it by another very small ripple. Tight control of red LED wavelength, or wavelength-specific calibration is important. In addition, artifacts due to finger motion should be identified and removed from the data or compensated.

The AC signal can be estimated from peak-to-valley distances; however, this requires low noise data. An alternative method which is robust to moderate levels of noise, but susceptible to rapid DC changes, is to calculate the root mean square deviation (aka, standard deviation) of the signal. It also has the advantage of being numerically robust to digitization noise in smaller amplitude signals.

For the SpO2 algorithm, a special asymmetrical Kalman filter is used which allows higher jumps up than down. This is because the human body oxygenation can return to normal in a single breath after hypoxia, whereas hypoxia requires a much longer time.

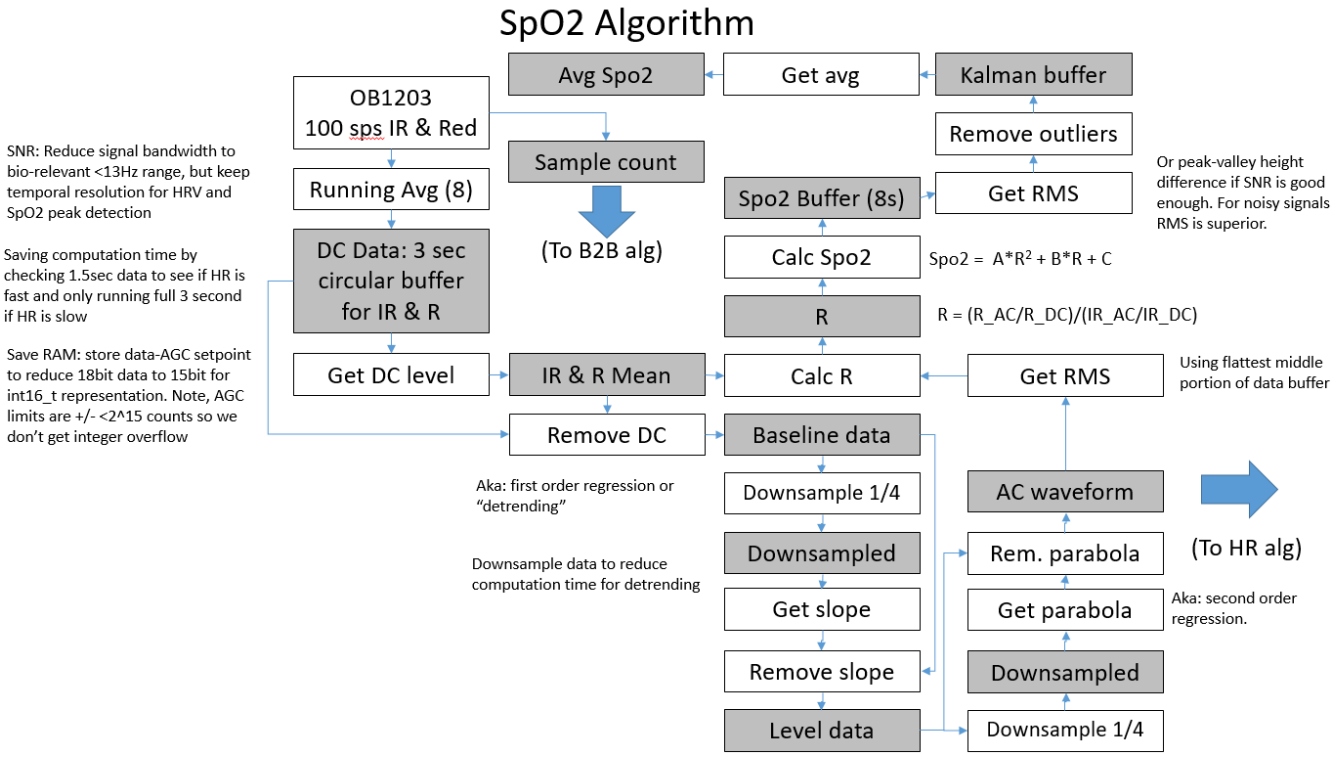

Figure 10: Regression and RMS-based SpO2 Algorithm Flow

2.2.1.1. Motion/Drift Artifacts

The second order regression detrending used in the correlation method is not able to fully flatten fast changes in the signal. If necessary, another suitable detrending method such as spline or adaptive bandpass filter can be used. In our algorithm this case is handled by an adaptive bandpass filter and is implemented in the beat-to-beat method.

In the presence of an uncompensated disturbance Δ small in comparison to the IR AC signal, but not negligible to the red AC signal we get:

$$R = \frac{(R_{AC} + \Delta)/R_{DC}}{IR_{AC}/IR_{DC}}$$

Where there is drift or motion, R increases, and measured SpO2 decreases relative to the correct value.

While the RMS method can measure SpO2 from very poor signals, it is less accurate with heavy breathing. It can also have limited accuracy for about the first 15-20 seconds of finger contact with the sensor for certain individuals, due to the perfusion in the finger tip changing when it comes into contact with the sensor. Best accuracy with low PI signals is available after about 10-30 seconds, depending on the individual.

As a result, the beat-to-beat method is used in parallel, which acquires SpO2 almost instantly, although the SNR is not as good. After the initial drift period and the signal stabilizes, the RMS result will be typically chosen for individuals with low PI based on lower measurement variance.

2.2.1.2. Alternative Methods

It is worth noting that it is not necessary to calculate RMS (which requires multiplication operations). The average absolute value can be used instead for reduced computation. Some adjustment of the PI thresholds/calibration may be necessary. Another low power method for SpO2 is adding up the differences from one sample to the next for both red and IR. The ratio of the cumulative sample differences (derivatives of the red IR signals during rising and falling portions of the waveform) is very low computation for even an 8-bit processor and yields a suitable R value. While, computationally simple, it becomes more difficult with low SNR and it works best with an analog detrending method such as an op amp that is doing DC signal removal. For a digital sensor like the OB1203, digital filtering is necessary so more powerful 16-bit methods are used.

2.3 Beat Detection Algorithm

For detection of beats with signal of poor quality, the peaks and valleys cannot always be accurately determined by simply looking at a waveform.

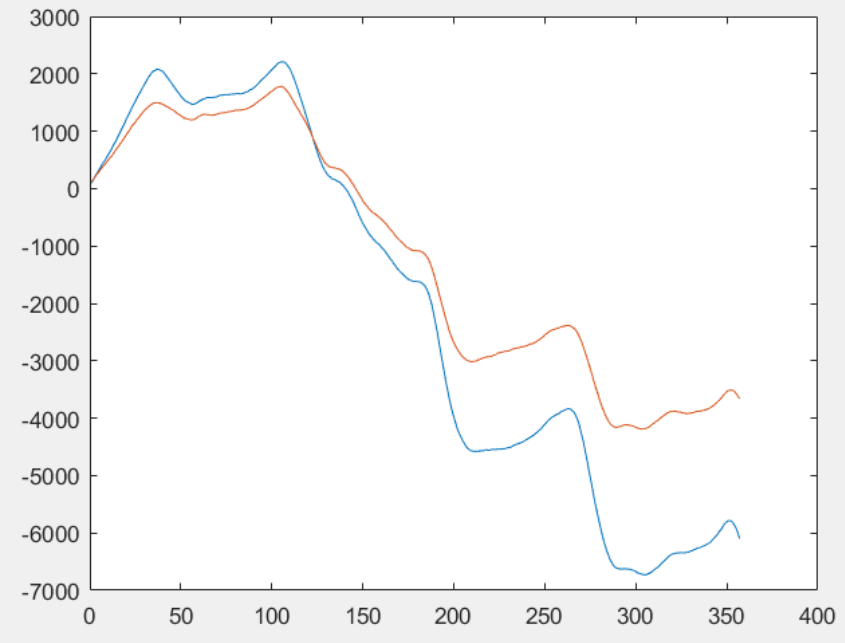

Figure 11: PPG Data with Large DC Variation (Motion, Drift, Breathing)

Figure 11 shows dc-removed data with large changes in amplitude (e.g., instance drift due to a change in finger pressure or sudden heavy breathing). Additionally there is noise in the data that would generate a large number of false peaks and valleys. Even after second order regression (Figure 12) we often find data in which true peaks are difficult to identify. In theory the user could apply strong filtering to the data, but this could erase peaks from fast heart beats.

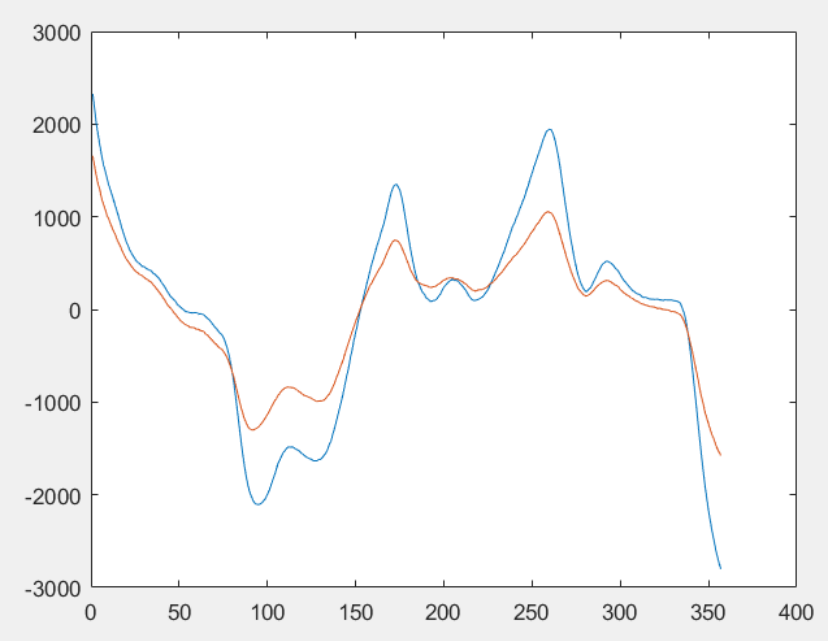

Figure 12: Tricky Data even after 2nd Order Regression Detrending

Ideally the user would choose a filter bandwidth width based on a very good guess of the heartbeat period. We have this from the correlation estimate of the heart rate. But rather than using a simple average, we create a filter with a passband shape similar to a real heartbeat PPG signal: a triangle.

The challenge is that so-called finite element response (FIR) filters are computationally intensive, involving tens of thousands of multiplications depending on the filter length. Accordingly, we implemented a computationally efficient FIR filter that allows for arbitrary filter shapes and lengths comprised of segments of slopes with powers

of two: 0, ±1, ±2, ±4, etc. This slope limitation allows the FIR filter to operate on bit shifts and addition/subtraction operations instead of multiplications. While an asymmetric triangle that looks more like a PPG waveform can be used, better suppression of dicrotic notches is achieved with a symmetric triangle (Figure 13).

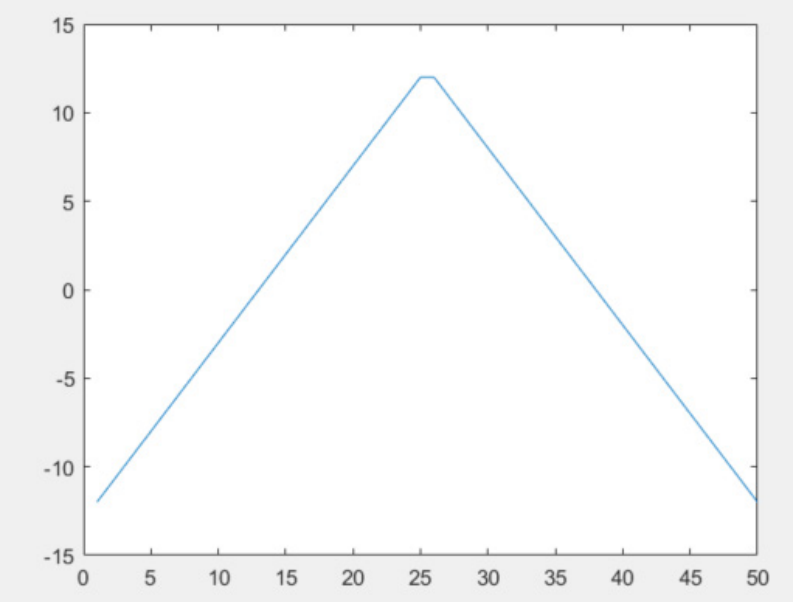

Figure 13: Triangle Filter Waveform

By convolving this trial waveform with the data, the user obtains a result very similar to a “matched filter”, which provides the theoretical best bandwidth for detection. The peak in the data corresponds to the position of best overlap of the trial filter with the data (i.e., the peak). The minimum is a valley (i.e., least like a peak). The output waveform looks more like a sine wave than a PPG wave. Note the filter average weight is zero so it removes the DC component as well as filtering out high-frequency components.

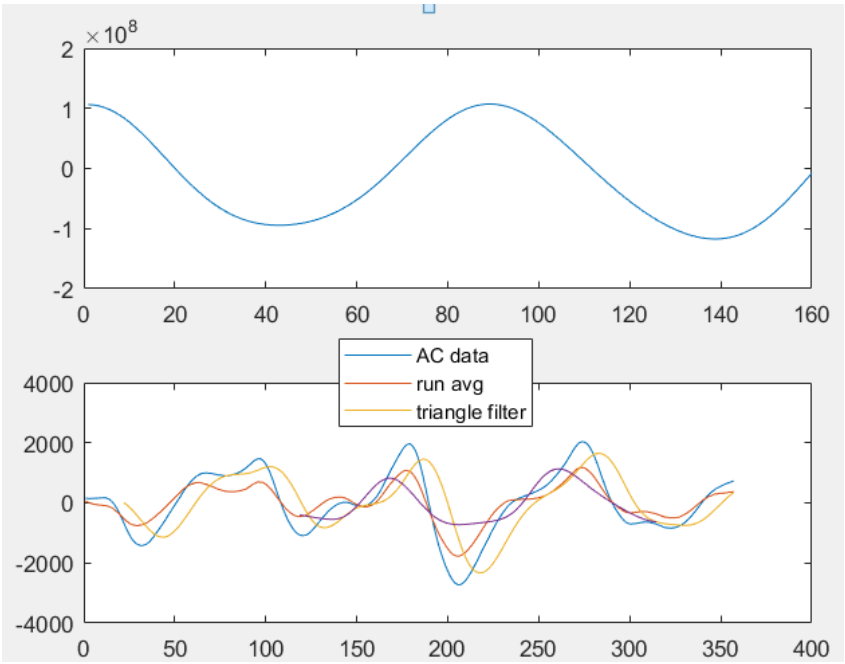

Figure 14: Triangle Filter Performance

Figure 14 compares several means of obtaining the heart rate. In the upper graph we see the correlation, which is very smooth and indicates a periodicity of about 90 samples. So, we expect a heartbeat peak every 90

samples. Looking at the running average data (red) we see that the noise added to the amplitude of the dicrotic notch visible on the original data (blue) has given us a large number of extraneous peaks. However, the triangle filter (yellow) improves the visibility of the peaks, giving us a proper estimate of the peak spacings or beat period.

In the beat-to-beat algorithm flow (Figure 15) we look only for peaks and valleys in each new increment of data (0.4sec if calculating RR) window and repeat this every interval to avoid double counting or missing peaks. In theory, a peak could be missed if the filter length changed from one measurement to the next and the peak was right on the edge due to a change in the filter lag. So, we limit the rate the filter can update and only accept beat to beat times from peak time differences measured with the same filter. A more complex method of peak counting by sampling prior to and after the window and reconciling double counting or missed samples is not warranted as the achieved performance is satisfactory.

To calculate SpO2 we need the signal amplitudes. However, the filter outputs depend on the square of the filter length, which is being dynamically updated. Note that typical FIR filter weights are normalized so that filtered output data has similar size signal as the input. This involves a lot of division. Fortunately, SpO2 is calculated using a self-normalizing ratio of ratios, so normalization is not necessary and is not implemented.

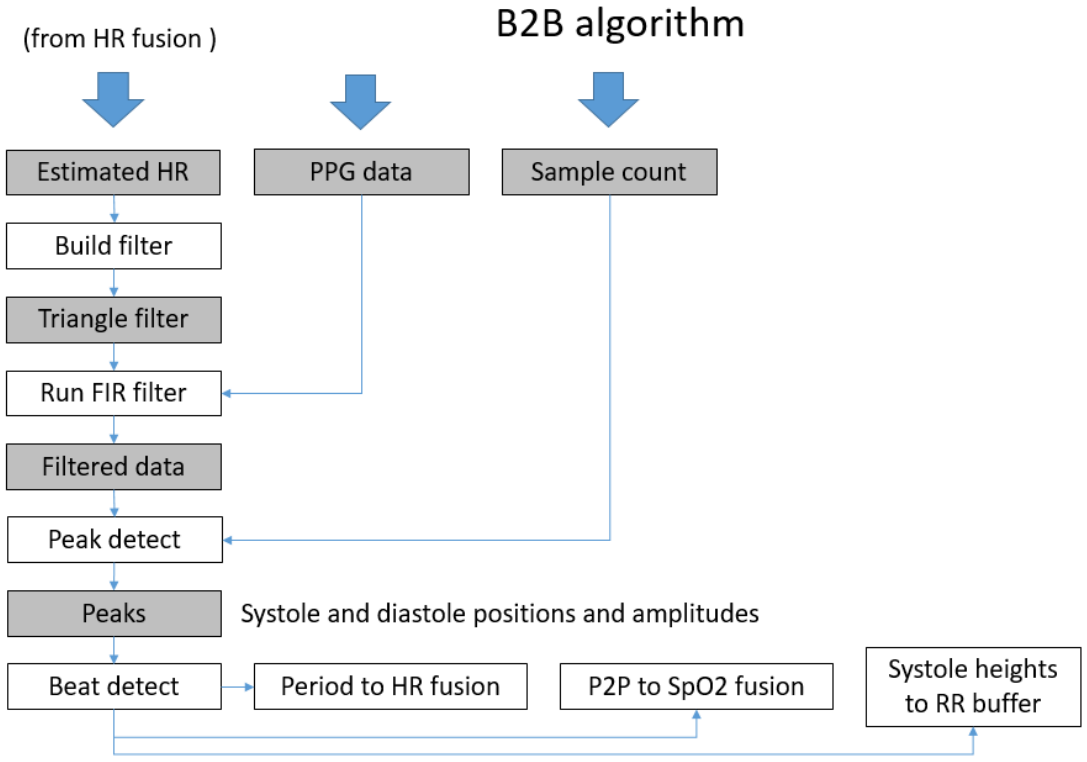

Figure 15: Beat-to-beat (B2B) Method Flow for Heart Rate, SpO2, and Respiration Rate

2.4 Respiratory Rate Algorithm

Respiratory rate can be measured by the modulation of the heartbeat according to the amount of air in the lungs, creating the “sinus rhythm” in the heartrate. When the lungs are full, baroreceptors on the heart cause it to take more frequent heart beats with lower stroke volume. The reduced stroke volume increases the DC level of the PPG signal. This is called respiration induced “baseline wander” or intensity variation (IV). Likewise, the changing stroke volume modulates the PPG amplitude (AM). The changing heart rate produces a frequency modulation (FM), as displayed in Figure 17. Other changes are apparent, including the maximum slope of the PPG signal rise to the systole which is visible in the peak of the first derivative (actually a valley in the raw data), which is a change in the PPG rise time to systole (fall time in raw data), etc.

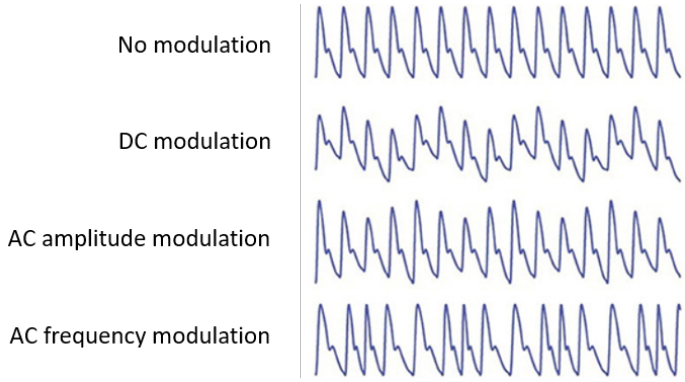

Figure 16: Modulation of PPG by Respiration

Tracking the systole peak value (valley in raw data) only includes both intensity and amplitude modulation and is a robust method of tracking respiration. Frequency variation alone is subject to autonomic variations unrelated to breathing and can have difficulty following fast breathing. Likewise, values like perfusion index or RMS which track only the amplitude can show “twinning” which complicates detection of heart rate. Since we log values at a fixed rate (INTERVAL/SAMPLE_RATE) and heart rate is not commensurate, we get significant “jitter” in addition to signal wander noise from physical changes. Therefore, correlation methods and peak detection are somewhat troublesome for RR determination. FFT typically generates multiple peaks. More advanced methods include wavelet transforms, synchro squeezing, and auto-regression; however, for our purposes a method convenient for 16-bit processors proves to be robust – provided a suitable filter is used. This sets up a “chicken and egg” problem, where lack of information about RR means we cannot select the right filter for detecting RR. Some of the key methods in the demonstrated RR detection relate to how the filters are adjusted in order to (1) acquire quickly, (2) adjust to changes, and (3) provide reasonable signal stability. Note, not all commercial device algorithms can adjust to sudden changes in RR.

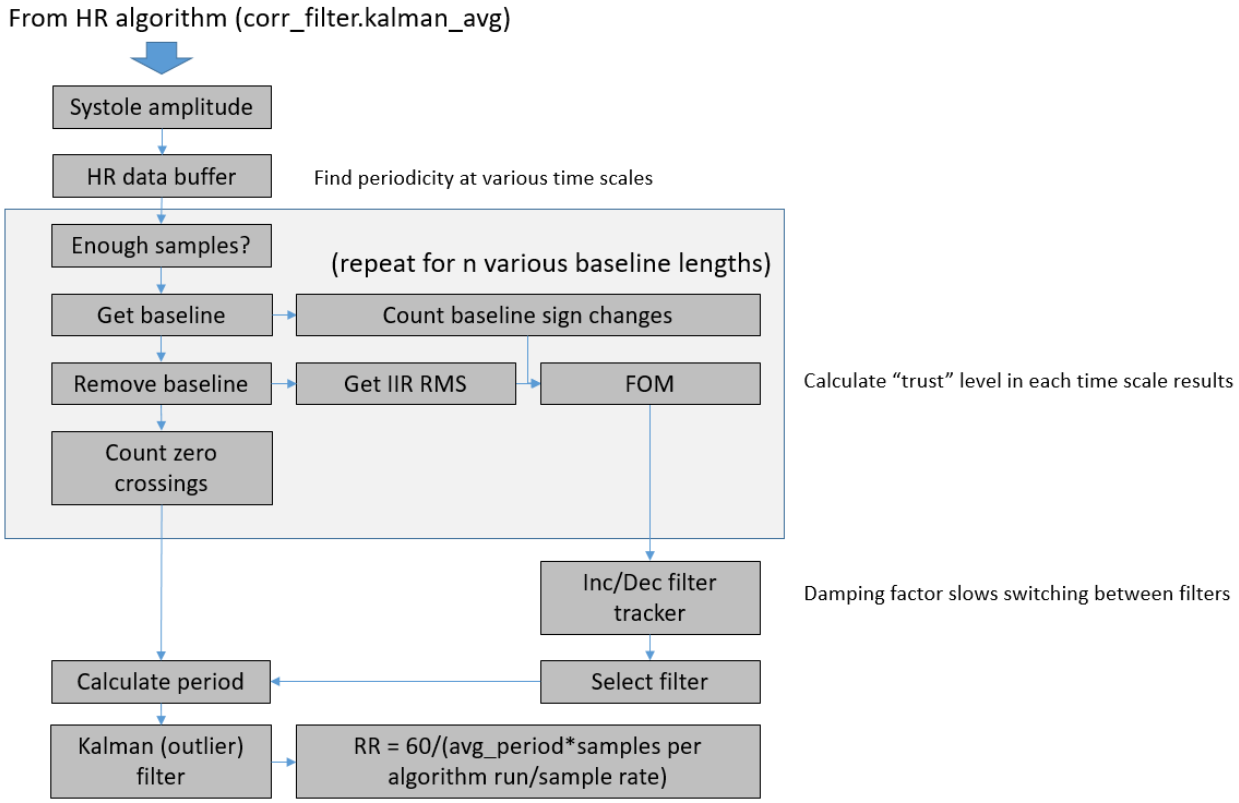

Figure 17: Respiration Rate Algorithm Flow

In addition, a variable length moving average is run by the application software to improve stability of output and adjust lag relative to preferred reference devices.

3. OB1203 Algorithm Walkthrough

The main program fetches data from the OB1203 at 100Hz for both red and IR in function `get_sensor_data()`. It arranges the bytes from the FIFO and calls `add_sample()`. The function `add_sample()` keeps a running average of eight samples (12Hz) and loads the filtered data into a circular buffer. The buffer is long enough for two beats at the slowest heart rate of ~40Hz, plus an extra sample to make the sum of squares easy, plus the maximum length of the moving average (dicrotic notch) filter. These values are defined in spo2.h.

The algorithm is run once each measurement interval (e.g., 1 second for HR/Spo2 or 0.4 seconds for RR).

The HR and SpO2 algorithm are in two pieces: `do_algorithm_part1()` (FIR filter) and `do_algorithm_part2()` (correlation). New samples are read from the FIFO between running the two portions of the algorithm. `do_algorithm_part3()` and `do_algorithm_part4()` do the respiratory rate (RR) algorithm and run immediately after part 2.

`do_algorithm_part1()` copies data using `copy_data()` from the circular buffer into linear arrays `AC1f`, obtains the DC value for PI and R calculations, then generates FIR-filtered data in arrays `ac_filtered` for IR and `r_ac_filtered` for red data. Lastly, a peak detect method `peak2peak()`is called to find peak amplitudes and beat periods for the beat detection method and PI/SpO2 calculation.

`do_algorithm_part2()` performs detrending on the `AC1f` data and calculates the RMS value for SpO2 in `get_rms()`, runs the tracking moving average filter through the `AC1f` data to remove the dicrotic notch, then calculates heart rate, PI, and SpO2 using both methods, performs the biological average (8 seconds) via respective Kalman filters, and arbitrates which method's result to use for the display out.

`get_rms()` calls `get_idx()` which figures out where in the buffer the most recent sample is and loads an array `idx[]` with indices for copying the data in the order of most recent to oldest. Note, any variable with a 1f (not LF-font can be confusing) means it has an additional 3 bits of fixed precision, so mean1f is 8 times the mean. `get_rms()` then it performs linear and quadratic regression to de-trend or flatten the buffered data using the following calculations:

- Calculate the slope: `slope1f`
- Remove the slope: `slope1f`
- Calculate the variance `var1f`
- Calculate the RMS `rms1f`

`calc_hr()`, `calc_R()`, `calc_R_p2p()`, and `calc_spo2()` are self-explanatory. The R value calculated from RMS in `calc_R()` is based on the RMS of the middle half the buffer where the signal is flattest. The R value from `calc_R_p2p()` is the peak-to-valley difference.

The spo2 calculation in `calc_spo2()` uses a polynomial in R to calculate the SpO2 level from the most stable R value.

Once the SpO2 is done we have the IR values stored in AC1f. We use those for the correlation-based heart rate calculation in `do_algorithm_part2()`. The following steps are performed:

1. Filter the data. This is necessary to flatten out the dicrotic notch that otherwise leads to false peaks (erroneously high heart rate). By default the maximum filter length is applied. If a previous valid heart rate average is available it will use less filtering for faster heart rates.
2. Run `find_max_corr()` to find the first maximum of the autocorrelation to determine signal periodicity. Note that signal quality is the ratio of the zero-offset correlation (dot product) to the first maximum. 0.5-1 is generally acceptable for stable algorithm operation. Less than 0.5 means bad. Less than 0.25 means awful. Signal quality can be used to identify motion artifacts. The correlation is filtered for outliers using a short time constant.
3. Calculate the heart rate from the correlation peak X using 60 sec/min * 100 samples/sec / X samples per period = BPM. This is retained in fixed point precision.

4. Run a Kalman filter to find and remove outliers in both SpO2 and HR and perform an 8 second (clinical) moving average.
5. Update the filter length (samples2avg).

For all peak estimations fast, quadratic fits are used to obtain resolution higher than the sample rate.

After pausing to collect more samples from the OB1203, the algorithm does `do_algorithm_part3()`. This is where the algorithm buffers the latest heart beat amplitude and calculates the approximate respiration rate (RR). RR is calculated by constructing a variety of moving averages or baselines. The algorithm calculates the number of zero crossings relative to each baseline over a length of samples `MAX_BREATH_FILTER_SPAN` is equal to 1.6 times the longest moving average. This is intended for accuracy for slow breathing. `MAX_BREATH_FILTER_SPAN` can be optimized for more accuracy or faster startup.

We characterize the baselines by counting the number of sign changes of the baseline (secret cause). More sign changes mean a better baseline. It is believed that the flatter a baseline is (i.e., it spans an integer number of breaths), the more sign changes occur, since small fluctuations can cause a sign change on a flat signal. This method is not described in literature and may be unique to our algorithm.

In `do_algorithm_part4()` we attempt to choose the best estimate of the breathing rate. We characterize the recent zero crossing calculations for each length of baseline using an IIR approximated value the RMS that is efficient for 16-bit processors.

We calculate a figure of merit (badness) based on the ratio of RMS zero crossings to baseline sign changes. We find the filter with the best figure of merit (lowest badness) and increment the filter tracker towards that filter. A `filter_damping` factor is used in the filter tracker to make it so that the filter switching takes place slowly. This enhances stability. Increasing filter damping means it can take longer to reach the correct value or adapt to a change, but there is less deviation from a correct value once we have it.

The RR value is calculated from zero crossings as breaths per minute, then the value is Kalman-filtered and displayed. An additional configurable duration moving average that can be increased for signal stability or decreased for improved response time.

3.1 Kalman Filters

Zero-order Kalman filters are used for the correlation filter result, the 8-second heart rate average, the 8-second SpO2 average, the respiratory rate, the beat-to-beat period, and the beat-to-beat amplitude.

The Kalman filter code is in `Kalman.c`.

The zero-order Kalman filter checks the RMS variation in the recent data buffer and removes outliers before adding valid samples to the Kalman buffer. The Kalman buffer is used to calculate the average.

Kalman filter parameters are defined in `Kalman.h`. Kalman filters are reset after a defined number of zero inputs (algorithm fails) or too many outliers in a row to allow the filter to more quickly recapture the correct value.

3.2 Data Quality Checks

For health related devices, it is often preferable to not display any value rather than display a potentially incorrect value. `Oximeter.c` features additional data quality checks enabled by the settings in step 6 by defining `CHECK_PI`. The following conditions result in a data “freeze” and eventual reset of the algorithm:

- PI below minimum threshold
- PI below low threshold and RMS variation in PI greater than the low PI RMS threshold
- PI above low threshold and RMS variation in PI greater the normal PI RMS threshold

4. Example Code

Download the demo source code at www.renesas.com/ob1203sd-rl-evk. Contact your helpful local sales representative for assistance.

5. Revision History

Revision	Date	Description
1.0	Jul 23, 2021	Initial release.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers skilled in the art designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only for development of an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising out of your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Rev.1.0 Mar 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.