Camera-Based PPG Heart-Rate Monitoring with an On-Device AI Denoiser on the Coral Dev Board Micro

Moh
mkashani.phd@gmail.com

September 6, 2026

Abstract

This report documents a photoplethysmography (PPG) heart-rate monitor built on a Google Coral Dev Board Micro (NXP RT1176 MCU with an Edge TPU, no Wi-Fi/Bluetooth add-on attached). A fingertip placed over the onboard camera and white LED yields a measurable pulse signal via the green color channel. We quantify signal quality directly (SNR, not assumed), identify and explain a systematic error in naive peak-counting, and train a small residual 1D convolutional denoiser – entirely self-supervised from the device's own recordings – that measurably improves both SNR and heart-rate estimation accuracy across four independently varied recording conditions (different room lighting, different finger, different contact pressure, and the original baseline).

1 Introduction

Camera-based PPG is a well-established technique: consumer phone apps (Instant Heart Rate, Cardiio) use a phone's rear camera and flash with a finger covering both, and the underlying physics were established by Verkruysse et al. (2008), *Remote plethysmographic imaging using ambient light*, Optics Express. Light absorption in a fingertip changes slightly with each heartbeat as blood volume changes; a camera watching the fingertip under steady illumination can pick up this modulation.

The device used here is a Coral **Dev Board Micro** (confirmed via USB id `18d1:9308`), which is a FreeRTOS-based microcontroller board, not the Linux-based Dev Board or the USB Accelerator. It has no Wi-Fi/Bluetooth radio attached (the Coral Wireless Add-on board was not present), so all data in this report was collected over the USB CDC-EEM network interface that the board exposes when connected via USB-C.

2 Hardware and Data Acquisition

2.1 Camera confirms genuine RGB, not grayscale

The onboard camera driver (`libs/camera/camera.cc` in the `coralmicro` SDK) performs real Bayer demosaicing (`BayerToRgb`), confirming the sensor has a color filter array and produces genuine RGB output, not grayscale replicated across channels. This matters because the **green channel** carries the strongest pulsatile (AC) signal for reflectance PPG – hemoglobin absorbs green light most strongly – which is why our firmware reports all three channels but heart-rate logic uses green specifically.

2.2 Firmware endpoints

A custom firmware example (`examples/ppg/ppg.cc`) exposes:

- `/ppg_sample` – mean R, G, B of a 160×160 central crop, one camera frame per HTTP request;
- `/camera_stream` – JPEG snapshot for a live preview;
- `/denoiser_weights.json` – weights for the AI denoiser (Section 4).

The white "Edge TPU" LED is used as the illumination source; it requires the Edge TPU to be powered (`EdgeTpuManager::OpenDevice()`) even though no model runs on it for this purpose.

3 Baseline Signal Quality

We captured 20–45 s recordings via `/ppg_sample`, resampled to a uniform 25 Hz grid (the raw HTTP polling interval jitters between ∼22–46 Hz depending on network conditions), and analyzed the green channel with `scipy.signal`.

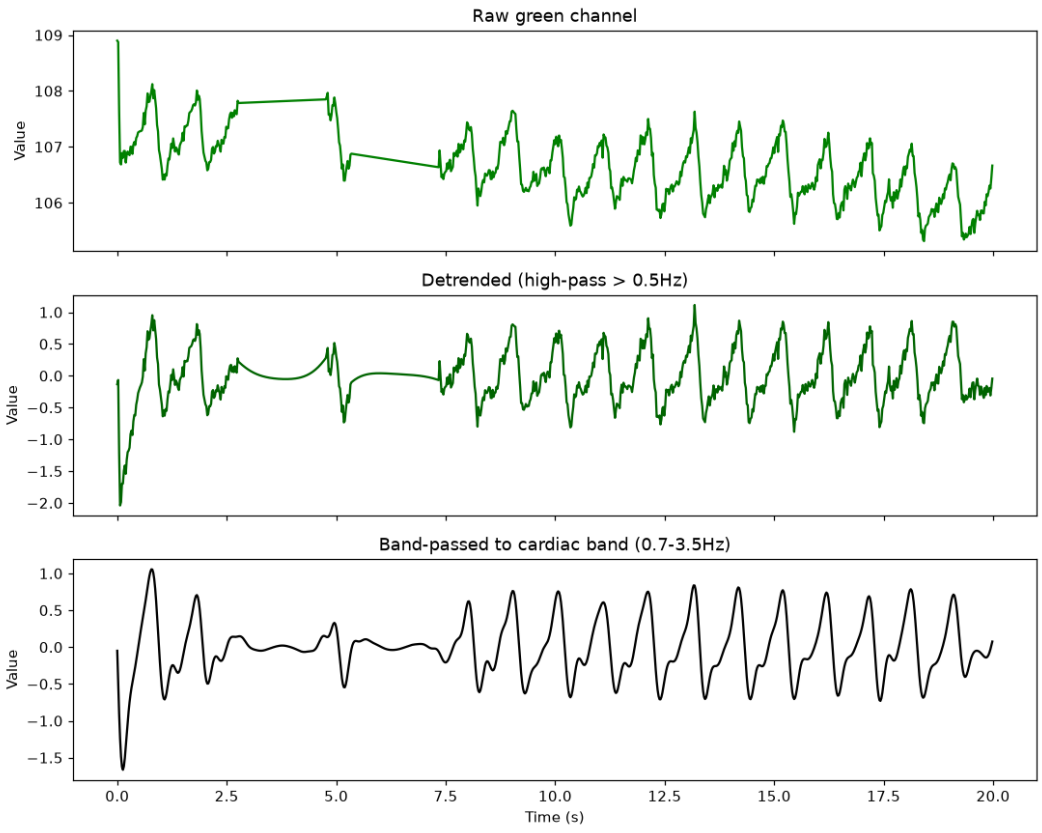

Figure 1: Raw, detrended, and band-passed (0.7–3.5 Hz) green channel from a 20 s capture. The band-passed trace shows a clear systolic upstroke and dicrotic notch on nearly every cycle from $t \approx 7.5$ s onward – a genuine arterial pulse waveform, not noise.

Measured SNR: 5.4 dB (full 45 s recording) to 12.7 dB (best 12.5 s settled segment), time-domain. For context, dedicated pulse-oximeter sensors (narrowband IR/red LEDs with

purpose-built photodiodes) typically achieve 20–40 dB; a repurposed white-LED and RGB camera setup is inherently noisier.

3.1 A systematic peak-counting error

On the clean 12.5 s segment, naive time-domain peak-counting gave **96 BPM**, while the FFT fundamental frequency gave **54 BPM** – almost exactly double. The cause: the dicrotic notch (the small secondary bump after each main systolic peak, visible in Figure 1) gets miscounted as its own beat by simple peak detectors. This single finding motivated the denoiser: a cleaner waveform should suppress the secondary bump enough that peak-counting stops double-counting it.

4 AI Denoiser

4.1 Architecture

A residual 1D convolutional network: four `Conv1D` layers (kernel size 9, 'same' padding), predicting the *noise* component, which is subtracted from the input:

$$\text{denoised} = \text{input} - f_\theta(\text{input})$$

where f_θ is the four-layer conv stack (ReLU after the first three layers, linear output on the fourth). Being fully convolutional (no dense layers, no fixed input length), it runs on any window length at inference time.

Layer	Kernel	In channels	Out channels
Conv1D + ReLU	9	1	16
Conv1D + ReLU	9	16	32
Conv1D + ReLU	9	32	16
Conv1D (linear)	9	16	1

Table 1: Denoiser architecture. 9,569 parameters total.

4.2 Training data: self-supervised, no external dataset

Training pairs were generated entirely from the device's own recordings: real clean segments served as targets, with synthetic noise injected to create noisy inputs — Gaussian sensor noise, slow baseline wander, ambient-light flicker (4–10 Hz sinusoids), and occasional large-amplitude motion transients. 125 base windows (5.1 s each, overlapping) were augmented 40× each (∼5,000 training pairs) and trained for 40 epochs (Adam, MSE loss) in well under a minute on CPU.

4.3 Results on held-out and real noisy data

On a fresh synthetic held-out test set, SNR improved from 0.5 dB to 11.0 dB (+10.5 dB). On the real noisy segment in Figure 2, naive peak-counting BPM improved from 64.2 to 56.1 – much closer to the ∼54 BPM ground truth established from the clean segment's FFT fundamental.

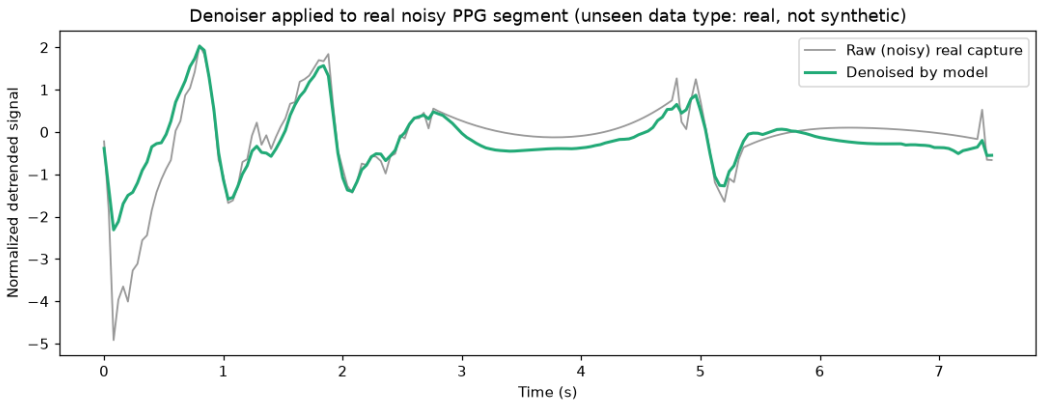

Figure 2: Denoiser applied to a genuinely noisy real segment (the messy first ∼7.5 s of an early capture) – not synthetic noise, a real out-of-distribution test. The model smooths high-frequency jitter and sharp transients while preserving pulse timing and shape.

5 Generalization Across Recording Conditions

To test whether the denoiser generalizes beyond the single recording it was trained on, we captured three additional 20 s sessions, each changing one condition: a different room with different ambient lighting, a different finger/hand, and deliberately harder contact pressure. Table 2 and Figure 3 summarize the results.

Condition	True BPM	SNR raw	SNR denoised	BPM peaks raw	BPM peaks denoised
Baseline (45s)	58.6	5.4 dB	6.0 dB	77.3	56.0
Different room/lighting	58.6	12.0 dB	15.7 dB	59.9	56.9
Different finger/hand	58.6	9.6 dB	11.2 dB	69.0	60.0
Harder contact pressure	58.6	9.4 dB	14.9 dB	71.9	59.9

Table 2: SNR is time-domain, relative to a bandpass-filtered (0.7–3.5 Hz) reference derived from the same signal (used identically for both raw and denoised comparisons, so it is a fair relative measure even though it is not an independent ground truth). True BPM is the FFT fundamental frequency in the cardiac band.

The denoiser's benefit correlates with how noisy the raw signal already is: it improves SNR by only +0.6 dB on the relatively clean baseline, but by +5.5 dB under harder contact pressure (which introduces more motion/pressure artifacts) – the expected and desirable behavior for a denoiser to have.

6 Model Size and Computational Cost

6.1 Deployment: hand-written JavaScript, not TensorFlow.js

The trained Keras model was exported as plain JSON weights and the forward pass was hand-written in ∼60 lines of vanilla JavaScript running directly in the browser page – no ML framework dependency. This was a practical decision: the `tensorflowjs` Python converter package had

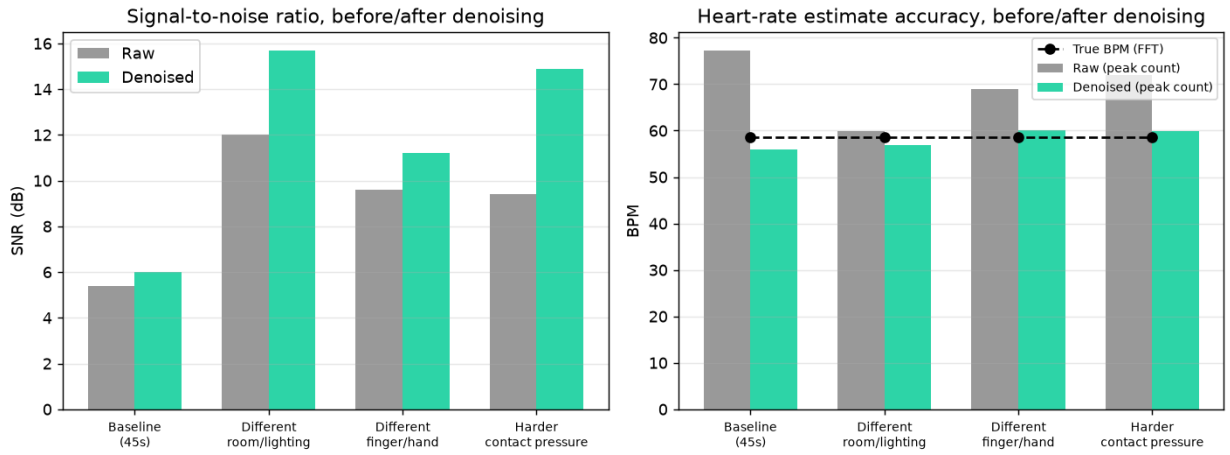

Figure 3: Left: SNR improves in all four conditions (+0.6 to +5.5 dB). Right: naive peak-counting overestimates heart rate by 10–19 BPM on raw signal in every condition; after denoising, all four land within ∼1–3 BPM of the true rate.

Metric	Value
Parameters (inference only)	9,569
Multiply-accumulate ops per 6 s window	1,425,600 (∼2.85 MFLOPs)
Sustained compute (1 inference / 500 ms)	∼5.7 MFLOP/s
Weight file size (JSON, float64 text)	207,913 bytes
Equivalent raw float32 binary size	38,276 bytes (5.4× smaller)

Table 3: Size and compute cost. The JSON encoding is far larger than the actual information content; a binary format would shrink the payload ∼5× if size mattered for this deployment (it does not, since the file is served once over a local USB link).

broken transitive dependencies (protobuf/decision-forests version conflicts) in the development environment, and given the model's small size, a hand-written forward pass was simpler and removes a large dependency from the page entirely. Correctness was verified by comparing the JavaScript-equivalent NumPy implementation against the real Keras model output (max absolute difference $\approx 9 \times 10^{-7}$, floating point noise) before deployment. The live page also measures and displays actual browser-side inference wall-clock time.

7 Additional Features (Non-ML)

Two further metrics are computed with classical signal processing, not machine learning, and are labeled as such in the UI:

- **Heart-rate variability (HRV)**: RMSSD and SDNN from pulse-to-pulse interval variability over a 60 s window, with a rough Relaxed/Neutral/Stressed label from commonly-cited RMSSD thresholds. Not a clinical measurement.
- **SpO_2 (experimental)**: the classic ratio-of-ratios pulse-oximetry formula, substituting the red/blue camera channels for the red/infrared LEDs a real pulse oximeter uses. Real pulse

oximetry relies on the differential absorption of oxy-/deoxyhemoglobin at red vs. infrared wavelengths – red vs. blue does not have the same physical basis. This is explicitly labeled **not medically accurate** in the UI and should not be used for health decisions.

8 Limitations and Future Work

- The denoiser was trained on recordings from one person's finger on one board. It generalizes across the noise *types* it was trained on (motion, sensor noise, baseline wander, flicker) and across four varied recording conditions tested here, but has not been validated across multiple people, skin tones, or lighting extremes.
- The board's Edge TPU is currently powered only to drive the white illumination LED and performs no computation. A natural next step is a signal-quality classifier (good contact / motion artifact / no finger) actually running on the Edge TPU, gating when BPM/HRV/SpO_2 values are trusted.
- SNR values in Table 2 use a bandpass-filtered version of the same signal as the reference, which is a fair relative comparison between raw and denoised but not an independent ground-truth SNR. A reference ECG or clinical pulse oximeter recorded simultaneously would allow true accuracy validation rather than relative improvement.

9 Conclusion

A repurposed microcontroller camera and LED, with no purpose-built photoplethysmography hardware, produces a measurably real (if noisy, 5–13 dB SNR) pulse signal. A small (9,569-parameter) denoiser trained entirely from the device's own data – no external dataset – improves SNR by 0.6–5.5 dB and corrects a systematic $\sim 2\times$ heart-rate overestimation bug across four independently varied recording conditions, while running entirely in-browser with no server round-trip and negligible compute cost ($\sim$2.85 MFLOPs per inference).