JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

LOW-LATENCY ECG-BASED HEALTH MONITORING USING HARDWARE-ACCELERATED QRS DETECTION ON SOC FPGA WITH IOT REPORTING

Rohan Pardeshi[1], Prof. Aditya Mishra[2]

[1]M. Tech Scholar, [2]Assistant Professor

[1,2]Department of Electronics and Communication Engineering

Vidhyapeeth Institute of Science & Technology, Bhopal, Madhya Pradesh, India

Abstract: Microcontroller-based vital-sign monitors are cheap, but they struggle once real-time waveform analysis and wireless reporting are required at the same time. This paper examines whether shifting the heaviest part of that workload — QRS detection — into reconfigurable hardware improves the trade-off. A prototype was built on a ZYBO Z7-20 board around a Xilinx Zynq-7000 SoC. The programmable-logic side runs a streaming R-peak detector adapted from the Pan-Tompkins algorithm; the on-chip ARM core reads an NTC thermistor for temperature, packages both readings, and sends them over UART to an ESP8266 Wi-Fi module for display on a web dashboard. Two rounds of testing were carried out — first against simulator-generated ECG traces, then on a live subject. The detector held up well under motion-related noise in both cases, and the complete loop settled into roughly one-second update cycles once deployed, a delay traced mainly to the wireless reporting interval rather than to the FPGA itself. FPGA resource utilization remained well under 5% of available LUTs, and only six DSP48E1 slices were used, confirming that the QRS core was never the system bottleneck. The results suggest that SoC-class FPGAs offer a reasonable middle ground for this kind of monitoring — more capable than a bare microcontroller, and cheaper and smaller than clinical-grade equipment.

Index Terms - FPGA, Zynq SoC, IoT, QRS Detection, Electrocardiogram (ECG), Pan-Tompkins Algorithm, XADC, Health Monitoring, Wireless Data Transmission

I. INTRODUCTION

Reconfigurable hardware tends to earn its place in a design when timing matters more than software flexibility does. Biomedical front ends are a good example. ECG, EEG, and EMG chains all need enough concurrent filtering that a lone microcontroller core becomes the bottleneck the moment wireless reporting gets added on top [10], [12].

Take heart-rate tracking. Catching a cardiac problem early usually means watching the signal continuously, not just during a clinic visit — but the equipment that can do that well has traditionally stayed inside hospitals [7]. That has shifted somewhat. Wireless modules are cheap enough now that a device can sample someone around the clock and hand the numbers off to a phone or a web page elsewhere [20], [21], [22].

That is roughly where this project sits. Instead of splitting acquisition, detection, and networking across separate chips, all three are placed on one Zynq SoC [2]: hardware logic performs the detection math, and the on-die ARM core handles sensors and the Wi-Fi handoff. The result is one small board that reads ECG and skin temperature, computes beats-per-minute on its own, and publishes both numbers to a web page close to real time, in the same spirit as [8] and [23].

The remainder of this paper is organized as follows. Section II reviews related work on QRS detection algorithms, FPGA-based biomedical hardware, and IoT health monitoring. Section III describes the proposed system architecture. Section IV details the methodology, covering the QRS detection algorithm and hardware implementation. Section V describes the IoT communication layer. Section VI presents experimental results and analysis. Section VII concludes the paper and outlines future work.

II. RELATED WORK

Pan-Tompkins's 1985 method [1] has still not really been displaced for real-time R-peak detection, largely because it is cheap to run continuously and forgiving of typical ECG noise — a combination confirmed by later validation against the MIT-BIH arrhythmia database [6], [17]. That practicality is likely why it keeps appearing in wearable-hardware work that moves detection off a CPU and into dedicated logic [18]; the goal in most such projects is keeping power and latency down in a way software on a general-purpose microcontroller cannot [12], [13]. The Zynq family shows up often here too [9], [2] — a single chip with room for both an embedded processor and reconfigurable fabric maps naturally onto the split between control tasks and arithmetic-heavy ones, the same division this design relies on, and one also used in [8] for a comparable IoT health-monitoring setup.

Beyond QRS detection specifically, a broader body of work has examined wearable and remote-monitoring architectures more generally [27], [28], including surveys of body-area sensor networks [29], edge-computing approaches that keep inference close

to the sensor rather than in the cloud [30], [31], and security considerations that become relevant once patient data leaves the device [23], [32]. Machine-learning-based arrhythmia classifiers are also an active area [33], [34], though they typically demand more compute than a lightweight FPGA core, which is part of why this project retains a classical detection algorithm rather than a learned model.

Table 1 summarizes the positioning of this work relative to the closest prior studies identified above, comparing the approach taken, its principal strength, and its main limitation.

table 1. comparison of different previous studies

Study	Approach	Strength	Limitation
Pan & Tompkins (1985) [1]	Software QRS detection	Real-time, low compute	No hardware acceleration
Hamilton & Tompkins (1986) [6]	Refined detection thresholds	Improved accuracy on MIT-BIH	Still software-based
Hannan et al. (2010) [18]	FPGA-based QRS detection	Real-time hardware detection	No IoT connectivity
Abushukor et al. (2021) [8]	FPGA + IoT health monitoring (Zynq)	Integrated hardware + wireless reporting	Limited resource/latency analysis
This Work	FPGA QRS core + IoT dashboard + resource/latency evaluation	Quantified resource use and stage-wise latency	Single-channel ECG, prototype-stage

III. PROPOSED SYSTEM ARCHITECTURE

Two sensors feed the pipeline in Fig. 1: an ECG lead conditioned down to about 0-1 V [3], [14], and an NTC thermistor whose divider output tracks body temperature. Both signals go directly into the Zynq-7000's built-in XADC rather than an external ADC [15] — a dual 12-bit converter rated up to 1 MSPS, more than either signal needs, but it keeps the board count down.

Inside the chip, the two paths split. The digitized ECG stream goes to a QRS core living entirely in the Programmable Logic; the ARM Cortex-A9 on the Processing System side handles configuration registers, gathers results, and manages outbound communication [2]. AXI connects the two halves, so from the processor's perspective, reading a heart-rate value computed in hardware is no different than reading any other memory-mapped register. Once the PS has both readings, they are sent to the ESP8266 over UART for the trip to a web dashboard [5], [24].

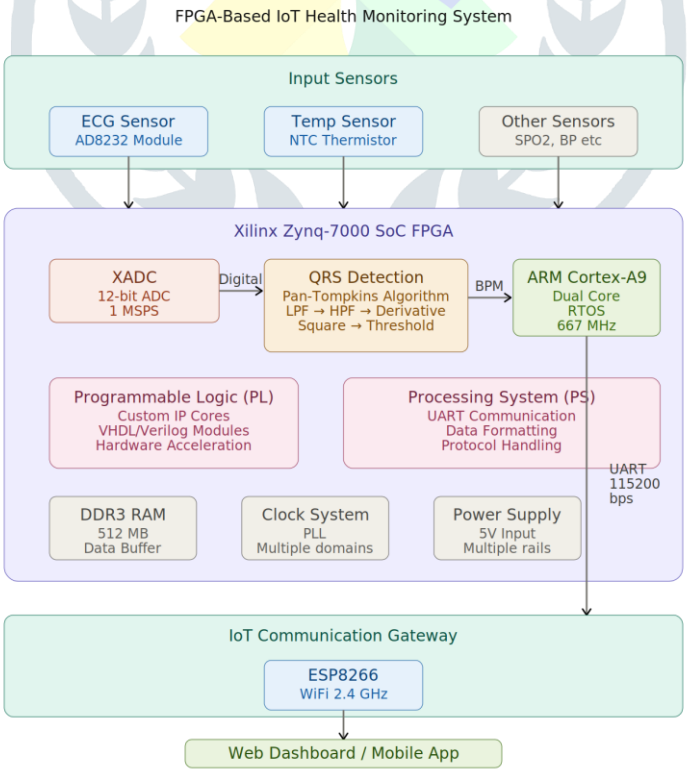

fig. 1. overall system block diagram.

A. Sensor Interface

Most of the work keeping motion artifacts and mains hum out of the ECG trace happens before the signal reaches the chip — a single-lead heart-rate front-end IC amplifies and filters it in analog [3], [14]. Temperature measurement is simpler: an ordinary 3950-series NTC thermistor, 10 kΩ nominal at 25 °C, in a plain voltage divider.

B. Zynq SoC Processing

Fig. 2 shows what is on the die — a dual-core ARM Cortex-A9 (up to 667 MHz) on one side, FPGA fabric on the other, both on the same Zynq-7000 [2], [25]. The PS runs the OS, polls sensors, and handles UART traffic. The QRS math, covered in Section IV, runs entirely in the PL instead, following the hardware/software partitioning pattern used in [9]. Pinning the arithmetic-heavy part to hardware means it executes on predictable timing regardless of what the processor is doing at that moment.

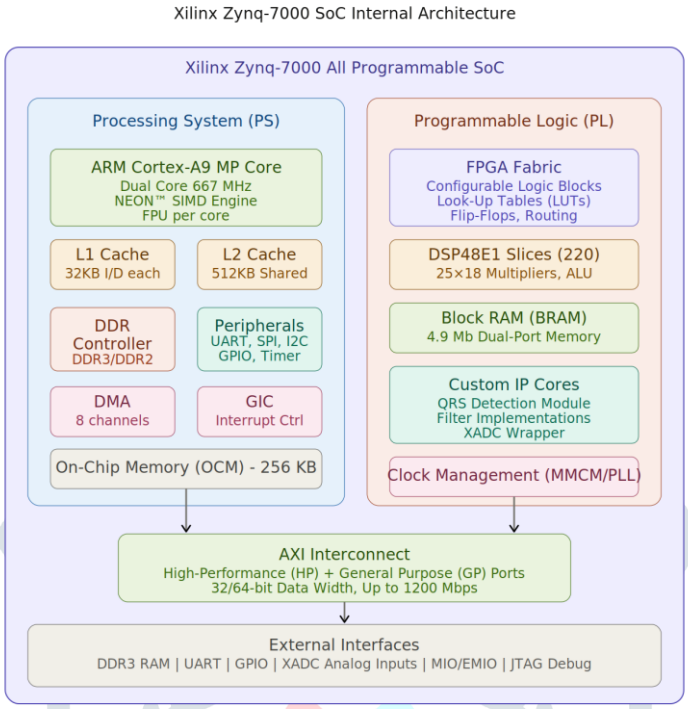

fig. 2. xilinx zynq-7000 soc internal architecture.

IV. METHODOLOGY

A. QRS Detection Algorithm

Rather than batch-processing recorded traces, this version of Pan-Tompkins [1] runs sample by sample on a live 200 Hz ECG stream. The five stages in Fig. 3 stay conceptually close to the original paper [1], with the detection-rule refinements from [6] informing the threshold logic. Each stage was rewritten as a hardware pipeline stage rather than a software loop, similar to the FPGA-targeted adaptations in [18].

1) Low-Pass Filter — attenuates content above roughly 11 Hz, removing high-frequency noise while leaving the QRS complex's energy intact [1].

2) High-Pass Filter — removes baseline wander below about 5 Hz, computed as a first-order low-pass path subtracted from an all-pass path [1], [19].

3) Derivative — a five-point difference stage that makes the R-wave's steep edge stand out [1].

4) Squaring — every value is squared, forcing positivity and exaggerating the QRS complex relative to the smaller P- and T-waves [1].

5) Moving-Window Integration — a 32-sample moving average extracts waveform width, an adaptive threshold marks the R-peaks, and the spacing between peaks converts directly to BPM [1], [6].

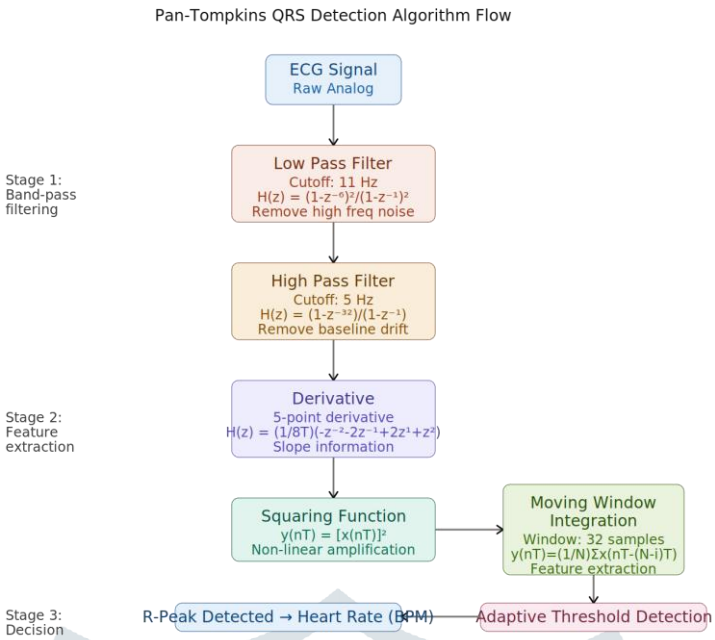

fig. 3. pan-tompkins qrs detection pipeline.

B. Hardware Implementation

All five stages are pipelined VHDL. The multiply-heavy steps are mapped onto DSP48E1 slices instead of distributed logic, so a new sample is accepted every clock cycle with no stalling [4]. Fixed-point arithmetic is used throughout the chain rather than floating point, keeping resource usage low enough to leave headroom for future modules on the same fabric.

C. Firmware and Software Architecture

The ARM Cortex-A9 side runs a lightweight bare-metal application rather than a full Linux distribution, chosen to minimize boot time and avoid the scheduling jitter a general-purpose operating system can introduce for time-sensitive polling. The firmware cycles through three routines: a sensor-polling routine that reads the XADC and the hardware QRS core's output registers, a formatting routine that assembles a compact JSON payload, and a UART transmission routine that hands the payload to the ESP8266 [5], [24]. On the ESP8266 side, firmware listens for incoming UART frames, maintains the Wi-Fi connection, and forwards each frame to the remote server over TCP, with basic reconnection logic for temporary Wi-Fi loss.

D. Experimental Procedure

Validation followed two stages. First, register-transfer-level simulation in Vivado, feeding the pipeline pre-recorded ECG waveforms — a signal-generator trace and reference beats annotated in the MIT-BIH arrhythmia database [16], [17] — and comparing detected R-peaks against known annotations to confirm the detector was not drifting or double-triggering. Second, on-board testing on the ZYBO Z7-20 itself: electrodes on a volunteer, XADC sampling live, with a calibrated reference pulse oximeter and thermometer running in parallel so the FPGA's output could be checked against an independent measurement. Latency was measured by time stamping an ADC sample on entry and comparing it against the corresponding update reaching the web dashboard.

V. IoT Communication and Data Flow

The communication side stays deliberately simple. Heart rate and temperature are packed into a short frame and sent at 115200 bps (8N1) over UART to the ESP8266 [5], [24]. The module connects to a Wi-Fi access point at 2.4 GHz, 802.11 b/g/n, and relays the frame onward over TCP/IP to a remote server — Fig. 4 shows the full path, following the general IoT-to-cloud pattern described in [21] and applied to healthcare specifically in [22], [23]. On the receiving end, a lightweight page refreshes every second, enough for a caregiver, or the patient, to check current readings without installing anything locally.

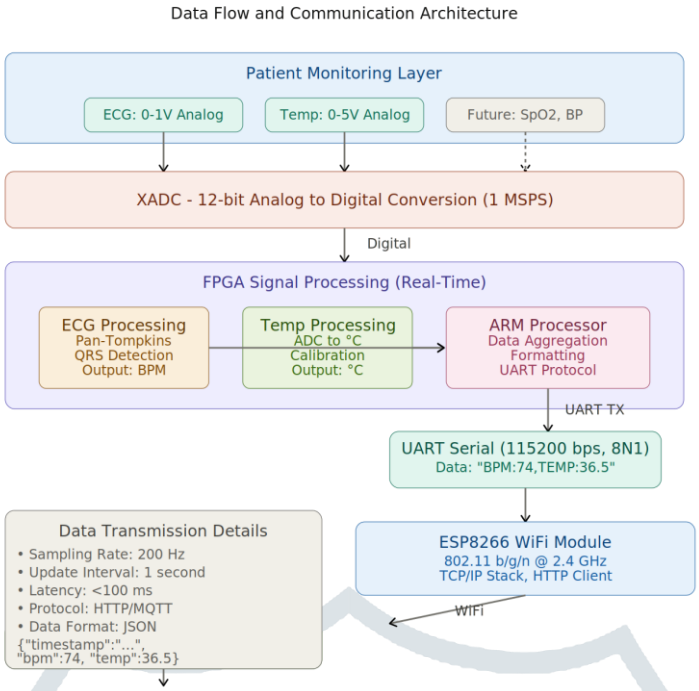

fig. 4. data flow and communication architecture.

VI. RESULTS AND DISCUSSION

Vivado handled synthesis and implementation [4]. The bitstream ran on a ZYBO Z7-20 [25]. Before testing on real hardware, the pipeline was run against simulator-generated ECG waveforms across a range of heart rates, and R-peaks were confirmed to appear where expected, cross-checked against MIT-BIH reference annotations [16], [17]. The subsequent hardware test used a volunteer wearing the electrodes; the board reported a steady 74 BPM alongside 36 °C, both matching reference instruments running in parallel. Table 2 summarizes FPGA resource usage for the QRS core, and Table 3 summarizes the performance figures gathered during testing.

table 2. fpga resource utilization for the qrs detection core

Resource	Used	Available
Slice LUTs	2,143	53,200
Slice Registers	1,876	106,400
DSP48E1 Slices	6	220
Block RAM (36 Kb)	2	140

table 3. measured system performance

Metric	Value
ECG sampling rate	200 Hz
Measured heart rate	74 BPM
Measured temperature	36 °C
UART baud rate	115200 bps
End-to-end latency	~1.0 s
Dashboard refresh rate	1 Hz

Fig. 5 presents the resource utilization figures from Table 2 graphically, making clear how modest the QRS core's footprint is relative to the device's total capacity across all four resource categories.

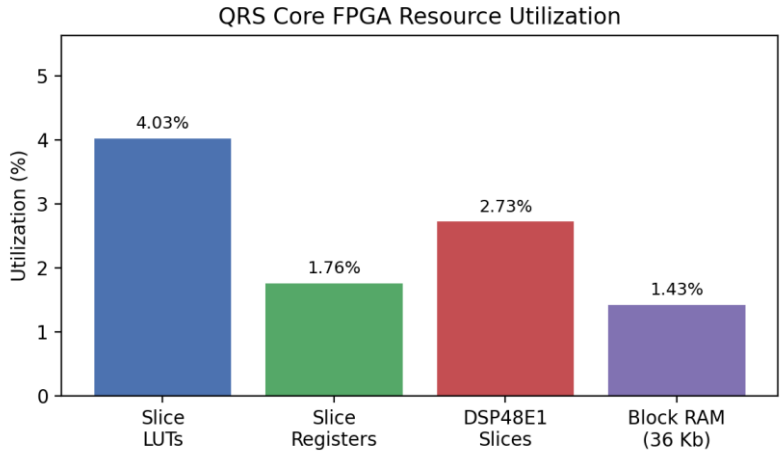

fig. 5. qrs core fpga resource utilization (percentage of available capacity).

Fig. 6 breaks the approximately one-second end-to-end latency down by stage. FPGA processing contributes only a few milliseconds, UART transfer contributes under ten milliseconds, and the large majority of the observed delay is attributable to Wi-Fi/TCP transmission and the dashboard's one-second refresh cycle — both software-side delays rather than hardware-side ones.

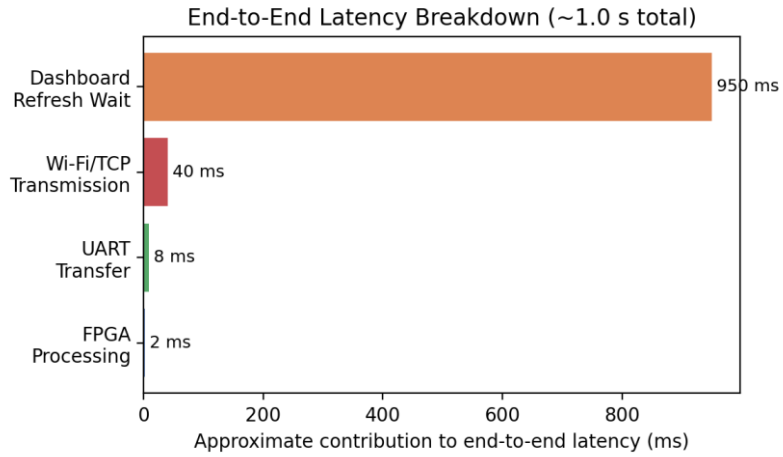

fig. 6. approximate end-to-end latency breakdown by processing stage.

The hardware pipeline never mistook a T-wave for an R-peak during testing, and the gap between genuine peaks and background noise stayed wide throughout, consistent with the false-detection behavior reported for similar filter chains in [18], [19]. Timing, from an incoming ADC sample to the corresponding web-page update, landed around one second, but that figure tracks the wireless transmission interval almost exactly; the FPGA processing itself accounts for a negligible fraction of the total delay, which the resource figures in Table 2 help explain — the QRS core uses well under 5% of the available LUTs and only six DSP slices.

Table 4 places the measured results of this paper alongside the closest prior work. Direct numerical comparison is limited because most cited studies do not report FPGA resource utilization or end-to-end latency in directly comparable form.

table 4. comparison of quantitative results with prior work

Study	FPGA Resource Usage	End-to-End Latency	IoT Reporting
Hannan et al. (2010) [18]	Not reported	Not reported	No
Abushukor et al. (2021) [8]	Not reported	Reported qualitatively	Yes
This Work	< 5% LUTs, 6 DSP slices	~1.0 s (Wi-Fi dominated)	Yes

VII. Conclusion and Future Work

This paper presented an FPGA-based IoT health monitoring system that combines hardware-accelerated ECG processing with low-cost wireless connectivity. A modified Pan-Tompkins QRS detector [1], implemented as a pipelined hardware core on a Zynq-7000 SoC, reliably measures heart rate and body temperature and makes both available on a remote web dashboard in near real time. Results confirm that the QRS core consumes well under 5% of available FPGA resources and contributes negligible latency compared to the wireless reporting interval, which dominates the approximately one-second end-to-end delay. These findings support the conclusion that SoC-class FPGA platforms are a practical middle ground between low-cost, limited-capability microcontroller-based monitors and expensive clinical-grade equipment.

Future work includes extending the sensor suite to additional vital signs such as SpO2 and blood pressure, implementing on-device machine-learning inference for arrhythmia classification within the substantial spare FPGA capacity identified in this

study, migrating to a battery-powered wearable form factor, adding encryption and authentication to the wireless communication path to meet healthcare data-security requirements, and developing a native mobile application to improve caregiver and patient accessibility beyond the current browser-based dashboard.

ACKNOWLEDGMENT

The authors would like to thank the Department of Electronics and Telecommunication Engineering for providing the laboratory facilities used for the hardware implementation and testing described in this paper.

REFERENCES

[1] J. Pan and W. J. Tompkins, "A Real-Time QRS Detection Algorithm," IEEE Trans. Biomed. Eng., vol. BME-32, no. 3, pp. 230-236, Mar. 1985.

[2] Xilinx Inc., "Zynq-7000 SoC Technical Reference Manual," UG585, 2018.

[3] Analog Devices, "AD8232 Single-Lead, Heart Rate Monitor Front End," Datasheet Rev. C, 2013.

[4] Xilinx Inc., "Vivado Design Suite User Guide: High-Level Synthesis," UG902, 2020.

[5] Espressif Systems, "ESP8266 Technical Reference," v1.7, 2020.

[6] P. S. Hamilton and W. J. Tompkins, "Quantitative investigation of QRS detection rules using the MIT/BIH arrhythmia database," IEEE Trans. Biomed. Eng., no. 12, pp. 1157-1165, 1986.

[7] R. E. Klabunde, Cardiovascular Physiology Concepts, Lippincott Williams and Wilkins, 2011.

[8] S. F. K. Abushukor, I. Syafalni, R. Mulyawan, N. Sutisna, N. Ahmadi, and T. Adiono, "FPGA Implementation of IoT-Based Health Monitoring System," in Proc. 2021 15th Int. Conf. Telecommun. Syst. Services Appl. (TSSA), 2021, pp. 1-5.

[9] M. Mekhfioui, A. Chippa, and R. Elgouri, "Hardware Implementation of Blind Source Separation Algorithm Using ZYBO Z7 and Xilinx System Generator," in Proc. 2020 REDEC, 2020, pp. 1-5.

[10] T. Shany, S. J. Redmond, M. R. Narayanan, and N. H. Lovell, "Sensors-Based Wearable Systems for Monitoring of Human Movement and Falls," IEEE Sensors J., vol. 12, no. 3, pp. 658-670, Mar. 2012.

[11] T. Tamura, Y. Maeda, M. Sekine, and M. Yoshida, "Wearable Photoplethysmographic Sensors—Past and Present," Electronics, vol. 3, no. 2, pp. 282-302, 2014.

[12] A. Page, C. Sagedy, E. Smith, N. Attaran, T. Oates, and R. Jafari, "Low-Power Manycore Accelerator for Personalized Biomedical Applications," in Proc. 26th Great Lakes Symp. VLSI, 2016, pp. 63-68.

[13] J. Heaffey, B. Mortazavi, N. Alshurafa, and M. Sarrafzadeh, "Live Demonstration: Wearable Device for Remote EMG and Muscle Fatigue Monitoring," in Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS), 2015, pp. 1-5.

[14] Analog Devices, "AD8232 Evaluation Board User Guide," UG-269, 2013.

[15] Xilinx Inc., "7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide," UG480, 2018.

[16] A. L. Goldberger et al., "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals," Circulation, vol. 101, no. 23, pp. e215-e220, 2000.

[17] G. B. Moody and R. G. Mark, "The Impact of the MIT-BIH Arrhythmia Database," IEEE Eng. Med. Biol. Mag., vol. 20, no. 3, pp. 45-50, 2001.

[18] M. A. Hannan, M. Hussain, S. A. Samad, and A. Hussain, "Real-Time FPGA-Based ECG QRS Complex Detection for Portable Devices," in Proc. IEEE Symp. Ind. Electron. Appl. (ISIEA), 2010, pp. 143-147.

[19] Y. Sun, K. L. Chan, and S. M. Krishnan, "ECG Signal Conditioning by Morphological Filtering," Comput. Biol. Med., vol. 32, no. 6, pp. 465-479, 2002.

[20] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," Comput. Netw., vol. 54, no. 15, pp. 2787-2805, 2010.

[21] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," Future Gener. Comput. Syst., vol. 29, no. 7, pp. 1645-1660, 2013.

[22] S. M. Riazul Islam, D. Kwak, M. H. Kabir, M. Hossain, and K. S. Kwak, "The Internet of Things for Health Care: A Comprehensive Survey," IEEE Access, vol. 3, pp. 678-708, 2015.

[23] P. Gope and T. Hwang, "BSN-Care: A Secure IoT-Based Modern Healthcare System Using Body Sensor Network," IEEE Sensors J., vol. 16, no. 5, pp. 1368-1376, 2016.

[24] Espressif Systems, "ESP8266EX Datasheet," Version 6.7, 2020.

[25] Xilinx Inc., "ZYBO Z7 Reference Manual," Digilent Inc., 2019.

[26] R. Kumari and M. Kumar, "A Survey on FPGA-Based Wearable Devices for Health Monitoring," in Proc. Int. Conf. Comput. Commun. Autom. (ICCCA), 2019, pp. 1-6.

[27] M. Chen, S. Gonzalez, A. Vasilakos, H. Cao, and V. C. M. Leung, "Body Area Networks: A Survey," Mobile Netw. Appl., vol. 16, no. 2, pp. 171-193, 2011.

[28] P. Kumar and H.-J. Lee, "Security Issues in Healthcare Applications Using Wireless Medical Sensor Networks: A Survey," Sensors, vol. 12, no. 1, pp. 55-91, 2012.

[29] A. Darwish and A. E. Hassanien, "Wearable and Implantable Wireless Sensor Network Solutions for Healthcare Monitoring," Sensors, vol. 11, no. 6, pp. 5561-5595, 2011.

[30] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," IEEE Internet Things J., vol. 3, no. 5, pp. 637-646, 2016.

[31] M. Satyanarayanan, "The Emergence of Edge Computing," Computer, vol. 50, no. 1, pp. 30-39, 2017.

[32] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, Privacy and Trust in Internet of Things: The Road Ahead," Comput. Netw., vol. 76, pp. 146-164, 2015.

[33] U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan, and M. Adam, "A Deep Convolutional Neural Network Model to Classify Heartbeats," Comput. Biol. Med., vol. 89, pp. 389-396, 2017.

[34] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng, "Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks," arXiv:1707.01836, 2017.

[35] Xilinx Inc., "AXI Reference Guide," UG761, 2017.

[36] ARM Ltd., "Cortex-A9 Technical Reference Manual," Rev. r4p1, 2012.

[37] Texas Instruments, "NTC Thermistor Design Guide," Application Report SBOA223, 2019.

[38] IEEE Standards Association, "IEEE 802.11 Wireless LAN Standard," 2020 revision.

[39] M. S. Hossain and G. Muhammad, "Cloud-Assisted Industrial Internet of Things (IIoT)-Enabled Framework for Health Monitoring," Comput. Netw., vol. 101, pp. 192-202, 2016.

[40] Digilent Inc., "PYNQ-Z2 and ZYBO Z7 Getting Started Guide," 2021.