RESOURCE-EFFICIENT EMBEDDED SPIROMETER FOR REAL-TIME RESPIRATORY HEALTH MONITORING

Submitted under Undergraduate Research Opportunities Programme (UROP)

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

AUTHORS: Lenin Valentine C (RA2311004010073) and Harshith Kamal R (RA231100401094)

GUIDE: Dr. S. Vasanthadev Suryakala Assistant Professor (Sr. G) Department of Electronics & Communication Engineering

Department of Electronics and Communication Engineering SRM Institute of Science and Technology, Kattankulathur

1. Introduction

A spirometer is a medical diagnostic device used to assess pulmonary function by measuring the volume of air inhaled and exhaled, along with the rate at which air flows through the respiratory airways. The clinical procedure performed using this device is known as *spirometry* and is a fundamental tool in respiratory medicine.

Spirometry is widely employed for:

- **Diagnosis** of respiratory disorders such as asthma, chronic obstructive pulmonary disease (COPD), chronic bronchitis, and pulmonary fibrosis
- **Monitoring** disease progression and evaluating response to treatment
- **Assessment** of pulmonary fitness prior to surgical procedures

A spirometer operates by measuring airflow generated during respiratory maneuvers and converting this signal into lung volume and flow-based parameters using numerical signal processing techniques.

Recent advances in wearable and portable medical technologies have shifted respiratory care from episodic, clinic-based testing toward continuous and point-of-care monitoring. As spirometry remains the gold standard for diagnosing and managing obstructive lung diseases, the development of portable, embedded spirometers is critical for early intervention and personalized healthcare. However, implementing spirometry on resource-constrained embedded platforms introduces challenges related to real-time processing, timing determinism, and efficient memory utilization.

2. Background Research

2.1 Spirometric Signal Processing Overview

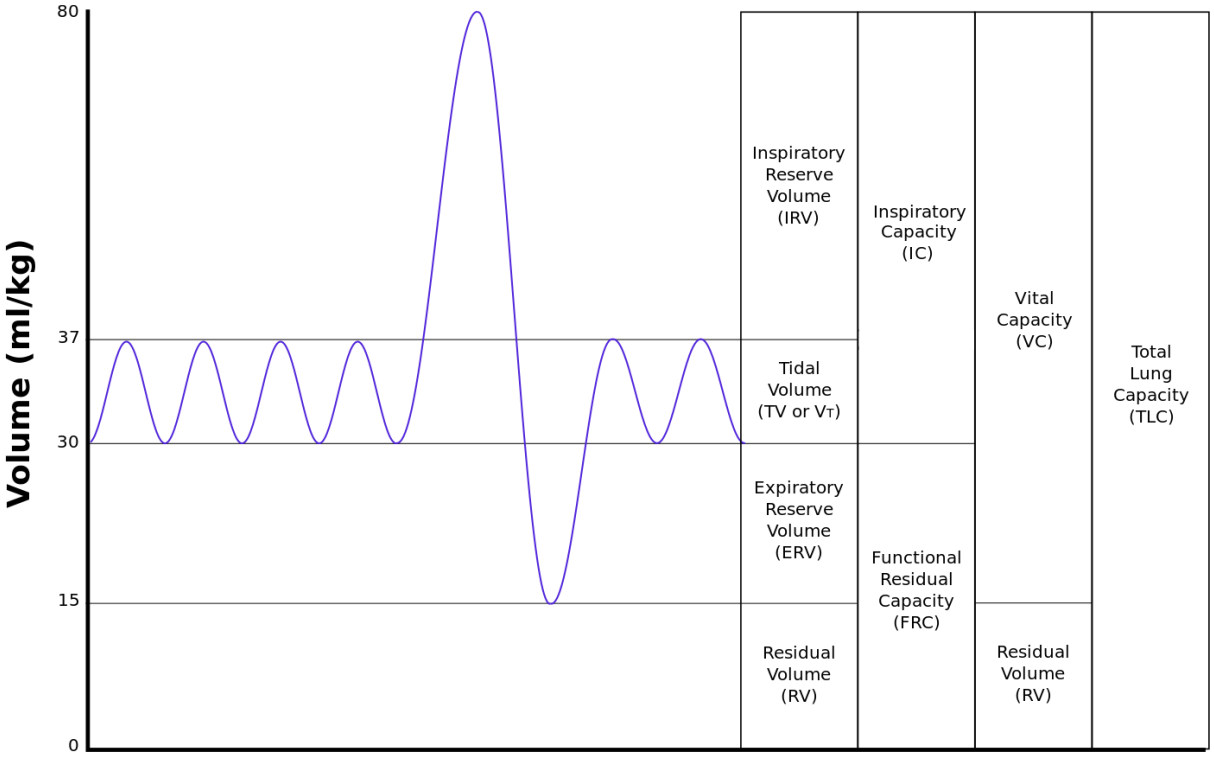
The primary objective of spirometry is the accurate computation of standardized lung function parameters from measured airflow signals. In embedded implementations, these parameters must be computed **online**, concurrently with signal acquisition, under strict timing and memory constraints. Unlike offline clinical systems, embedded spirometers must guarantee deterministic execution to preserve temporal accuracy.

2.2 Detection of Respiratory Maneuver Phases

Accurate parameter computation begins with reliable detection of key phases of the respiratory maneuver. Two critical events are identified:

- **Start of forced expiration**, detected when airflow exceeds a predefined threshold following maximum inhalation
- **End of expiration**, detected when airflow falls below a minimum threshold for a sustained duration

These events are time-critical and require precise timestamping. Any delay or jitter in detecting the onset of forced expiration directly affects the accuracy of time-dependent parameters such as Forced Expiratory Volume in one second (FEV_1).



2.3 Computation of Volumetric Parameters

2.3.1 Flow-to-Volume Conversion

The instantaneous airflow signal $Q(t)$is converted into lung volume through numerical integration:

$$V(t) = \int_0^t Q(\tau)\, d\tau$$

In discrete embedded implementations, this integration is typically performed as:

$$V[n] = V[n-1] + Q[n]\Delta t$$

where Δt is the sampling interval. Stable sampling and bounded execution latency are essential to prevent integration drift and accumulated numerical error.

2.3.2 Forced Expiratory Volume Parameters

- **Forced Expiratory Volume in 1 second (FEV_1)**

$$\mathrm{FEV}_1 = V(t = 1\ \mathrm{s})$$

This represents the volume of air exhaled during the first second of forced expiration and is a primary indicator of airway obstruction.

- **Forced Vital Capacity (FVC)**

$$\mathrm{FVC} = V(t = T_{\mathrm{end}})$$

This corresponds to the total volume of air forcibly exhaled until expiration terminates and reflects overall lung capacity.

- **Forced Expiratory Volume in 6 seconds (FEV_6)**

$$\mathrm{FEV}_6 = V(t = 6\ \mathrm{s})$$

FEV_6 is often used as a practical substitute for FVC in portable spirometers, as it reduces maneuver duration while retaining diagnostic relevance.

2.4 Computation of Flow-Based Parameters

2.4.1 Peak Expiratory Flow (PEF)

Peak Expiratory Flow represents the maximum instantaneous airflow achieved during forced expiration and is computed as:

$$\mathrm{PEF} = \max\{Q(t)\}$$

PEF is highly sensitive to sampling rate, filtering delay, and execution latency. Inadequate temporal resolution or excessive processing delay may lead to underestimation of peak flow values.

2.4.2 Flow–Volume Loop Capture

The Flow–Volume Loop (FVL) is generated by plotting airflow against lung volume:

- x-axis: Lung volume (L)
- y-axis: Airflow (L/s)

The FVL provides valuable diagnostic insight, enabling visualization of obstructive versus restrictive respiratory patterns, detection of upper airway obstruction, and shape-based clinical interpretation. Capturing the FVL requires synchronized storage of flow–volume samples with sufficient temporal resolution.

2.5 Parameters Implemented in the Current System

Category	Parameter	Computation
Volume	Total Volume	$\int Q(t)\, dt$
Time-based	FEV_1	$V(1\ s)$
Time-based	FEV_6	$V(6\ s)$
Capacity	FVC	$V(T_{\text{end}})$
Ratio	FEV_1/FVC	Division
Flow	PEF	$\max(Q)$
Graphical	FVL	Flow–volume samples

2.6 Implications for Real-Time Scheduling and Resource Design

The real-time computation of spirometric parameters imposes combined constraints on system design:

- **Latency:** Delays in task execution directly affect time-based parameters such as FEV_1 and FEV_6
- **Jitter:** Variability in sampling or processing distorts numerical integration and peak flow detection
- **Memory footprint:** Continuous buffering of flow and volume samples must be achieved without excessive SRAM usage

To meet these requirements, the firmware architecture must provide predictable task scheduling, bounded execution times, and controlled memory utilization. These constraints motivate the use of a real-time operating system (RTOS) and systematic evaluation of timing and memory behavior.

3. Literature review:

Paper	Architecture	Key Contributions	Identified Gap	Inference for This Work
"TeleSpiro: A Low-Cost Mobile Spirometer for Resource-Limited Settings" (Carspecken et al., IEEE PHT 2013)	Single-core MCU, differential pressure sensor, mobile-powered	Demonstrates feasibility of low-cost embedded spirometry meeting ATS/ERS accuracy; on-device signal acquisition	No RTOS, no latency/WCET/jitter analysis, no memory footprint evaluation	Embedded spirometry feasibility proven, but **real-time firmware behavior is uncharacterized**
"Design and Development of a Portable Spirometer for Pulmonary Function Test"	Single-core microcontroller-based spirometer	On-device airflow acquisition and spirometric parameter computation	Scheduling, timing determinism, and memory usage not analyzed	"Real-time" operation assumed, not quantitatively validated
"Development of a Low-Cost Portable	Embedded pulmonary	Continuous respiratory monitoring	No RTOS-level timing metrics or deadline analysis	Functional correctness

Spirometer for Pulmonary Disease Monitoring"	monitoring system	with parameter extraction		prioritized over determinism
"IoT-Based Smart Spirometer for Respiratory Health Monitoring"	Embedded + IoT architecture	Parameter extraction and visualization	Latency and memory footprint not quantified	Highlights need for on-device processing, but lacks system-level evaluation
"Spirometry Reference Values and Clinical Interpretation in Pulmonary Diseases" (BMC Pulmonary Medicine, 2022)	Clinical software-based analysis	Clinical validation of spirometric parameters	Not embedded-focused; no system constraints	Confirms **clinical importance of FEV_1, FVC, PEF**
"Global Trends in Spirometry Use for Pulmonary Diagnosis" (BMC Pulmonary Medicine, 2025)	Population-scale clinical study	Large-scale validation of spirometry metrics	No embedded or real-time considerations	Reinforces need for timing-accurate embedded computation
"Deep Learning–Based Spirometry Analysis for Pulmonary Disease Detection" (Scientific Reports, 2025)	ML-heavy software pipelines	Advanced interpretation accuracy	Computationally intensive; non-real-time	Unsuitable for resource-constrained embedded systems
"Real-Time Embedded Signal	Embedded signal-	Structured embedded	No biomedical real-time validation	Architectural concepts reusable, but not

Processing Pipelines for Cyber-Physical Systems” (ACM EMSOFT 2023)	processing pipelines	processing frameworks		medical-timing-aware
“Embedded Biomedical Signal Processing: Architectures and Applications” (JMIR Biomedical Engineering, 2023)	Embedded biomedical pipelines	End-to-end biosignal processing	No WCET, jitter, or deadline analysis	RTOS impact on biomedical signals not quantified
“Biomedical Signal Processing on Resource-Constrained Embedded Systems” (Springer Book Chapter, 2022)	Single-core embedded DSP	Efficient filtering and feature extraction	Single-core assumption; no scalability analysis	Algorithms transferable, architecture unstudied
“IoT-Based Embedded Healthcare Systems: Design and Implementation ” (Springer Book Chapter, 2021)	Embedded + cloud healthcare systems	System-level integration	Cloud-dependent latency	Not suitable for time-critical spirometry
“Edge Computing for Biomedical Signal Processing” (Springer Book Chapter, 2023)	Edge biomedical platforms	Embedded intelligence at the edge	Resource usage not quantified	Resource efficiency not demonstrated

"Modular Design and Optimization of Biomedical Applications for Ultralow Power Heterogeneous Platforms" (IEEE TCAD, 2020)	Heterogeneous multicore (ULP) biomedical platform	Demonstrates parallelism improves energy efficiency	No medical real-time deadlines or jitter/WCET analysis	Multicore benefits shown, but **determinism not evaluated**
"Inter-Core Communication Performance Evaluation of a Multicore Microcontroller for Edge Computing Applications"	Dual-core microcontroller	Quantifies IPC latency and throughput	Application-agnostic	Essential groundwork for dual-core spirometry design
"Exploration of Power-Savings on Multi-Core Architectures with Offloaded Real-Time Operating System"	Multicore RTOS with task offloading	Demonstrates energy reduction through offloading	No biomedical workload	Supports resource-efficiency motivation
"Resource and Performance Trade-Offs in Real-Time Embedded Control Systems" (Real-Time Systems, 2013)	RTOS-based real-time systems	Formal resource–performance trade-off framework	Control-focused, not biomedical	Provides evaluation methodology applicable to spirometry

4. Research Gap and Scope

Based on the reviewed literature, several gaps are identified:

- **Lack of RTOS-level real-time analysis:** Existing works do not quantify end-to-end latency, worst-case execution time (WCET), or jitter.
- **Absence of memory evaluation:** Prior studies rarely report SRAM usage or task stack requirements.
- **No comparative evaluation of multicore execution:** The impact of dual-core architectures on time-critical spirometric computation remains unexamined.

5. Scope of the Present Work

This work focuses on the design of a resource-efficient, RTOS-based embedded spirometer on an **STM32** platform, prioritizing:

1. **Fully on-device computation** of FEV_1, FVC, FEV_6, PEF, and FVL.
2. **Quantitative evaluation** of real-time metrics (latency, jitter, WCET).
3. **Comparative analysis** between single-core and heterogeneous dual-core execution.

6. Performance & Evaluation Parameters

Category	Parameters Evaluated
Real-Time Performance	End-to-End Latency, WCET, Deadline Miss Rate
Scheduling & Timing	Task Start-Time Jitter, Response Time, Context Switch Rate
CPU Efficiency	Utilization per Core, Idle Time/Cycles Percentage
Memory & Safety	Total SRAM Footprint, Heap Usage, Stack High Water Mark
Output Consistency	FEV_1, FVC, and PEF deviation across architectures