Bootstrapping Health Wearables Powered by Intra-Body Power Transfer

Saad Ahmed[Ψ] Eren Yildiz [†] Shashank Holla[Ψ] Noor Mohammed[⋆] Bashima Islam[‡] Kasim Sinan Yildirim[◇]
Jeremy Gummeson[⋆] Sunghoon Ivan Lee[⋆] Josiah Hester[Ψ]

[Ψ]Georgia Institute of Technology [†]Ege University [⋆]University of Massachusetts Amherst
[‡]Worcester Polytechnic Institute [◇]University of Trento

2024 IEEE 20th International Conference on Body Sensor Networks (BSN) | 979-8-3315-3014-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/BSN63547.2024.10780616

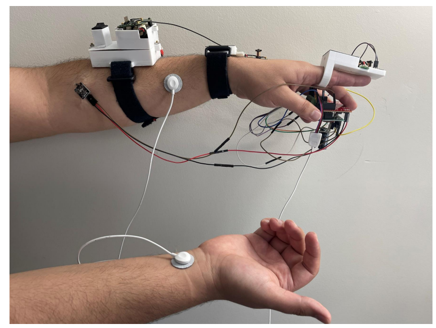
***Abstract*—Continuous health monitoring is crucial to ensuring better health and taking preventive measures just-in-time. Existing battery-powered health wearables pose a significant limitation to continuous monitoring as batteries wear out after fixed energy cycles and need replacement. Ambient energy harvesting unlocks battery-free sensing but it suffers from spatio-temporal variability, making it unfit for health sensing. Intra-body power transfer (IBPT) provides an alternative energy source for battery-free operation, however, it can only provide limited energy in order to ensure wearer's safety. Existing system support is designed to maximize computational progress in a single energy cycle, thus wasting energy on computations that become stale in the next energy cycle.**

We instantiate an IBPT-powered health wearable capable of supporting multiple health sensors. To cope with lower incoming energy, we introduce BodyOS; a system support that exposes programming constructs for domain experts to express health applications in terms of the inherent dependencies of bio-signals being monitored by the application. By avoiding unnecessary sensing operations, BodyOS allows energy-efficient application execution and faster capacitor recharge while ensuring that the data sensed by the application is always useful. We evaluate BodyOS to show that it significantly improves energy efficiency, thus increasing the on-time and number of data points collected by the device.

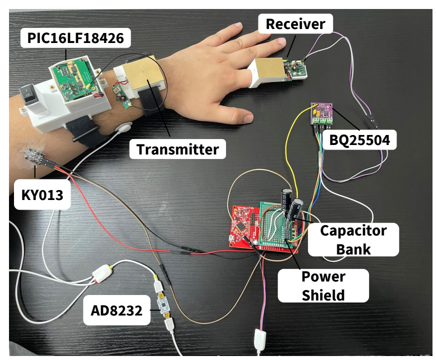
I. Introduction

Health wearables play a critical role in providing continuous and real-time monitoring essential for better management of patients with chronic illnesses such as diabetes, neurological disorders and cardiovascular diseases [1] which account for three quarters (74%) of all deaths around the world, according to the World Health Organization (WHO) [2]. Unfortunately, existing health wearables are powered by batteries that wear out over time (after a few hundred cycles) and must be periodically recharged or replaced [3], [4], thus incurring a high maintenance cost and produces e-waste that has a catastrophic environmental effect.

Several battery-free systems [5], [6] have been proposed in the literature that are capable of harvesting energy from ambient sources, paving the way for fully unobtrusive wearables that would enable continuous and sustainable health sensing. However, ambient energy sources suffer from spatiotemporal variability and are not always feasible for deployment. Not only that, all of these solutions force the wearable to be worn at a specific location on the body to maximize harvested energy that may not be the best location to measure the biosignal [7].



(a) Placement of electrodes and sensors



(b) Components used in the hardware setup

Fig. 1: Placement of sensors on the body and distance between the transmitter and receiver (Figure 1a) as well as the name of each component used for developing batteryless health wearable (Figure 1b).

In this paper, we instantiate a batteryless health wearable supporting multiple health sensors powered by IBPT mechanism, as shown in Figure 1. As the transferred power via IBPT is weak to ensure the wearer's safety, we introduce BodyOS[1], a programming language and runtime support that enables programmers to exploit interdependencies between bio-signals and efficiently manage incoming energy by avoiding wasted computations due to repeated I/O operations.

[1]Bootstrapping Health Wearables powered by Intra-Body Power Transfer

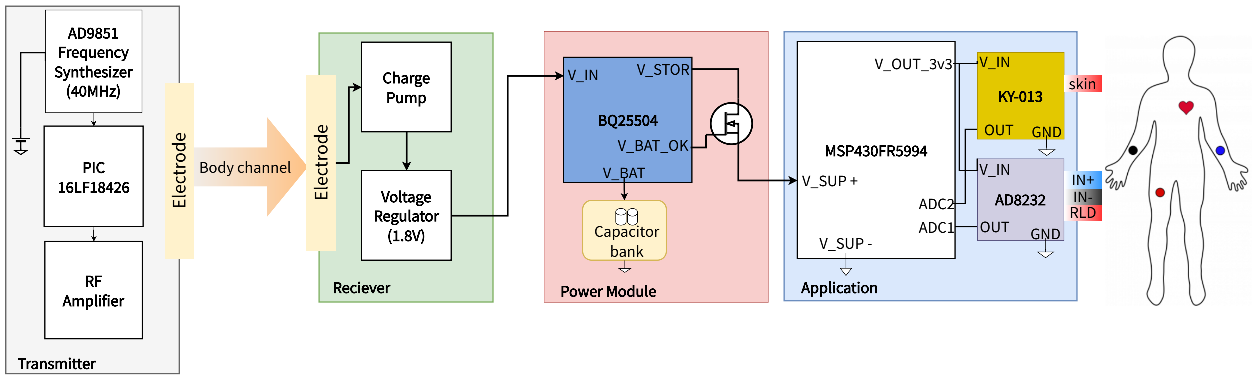


Fig. 2: Block diagram for the batteryless health wearable deployed on the body.

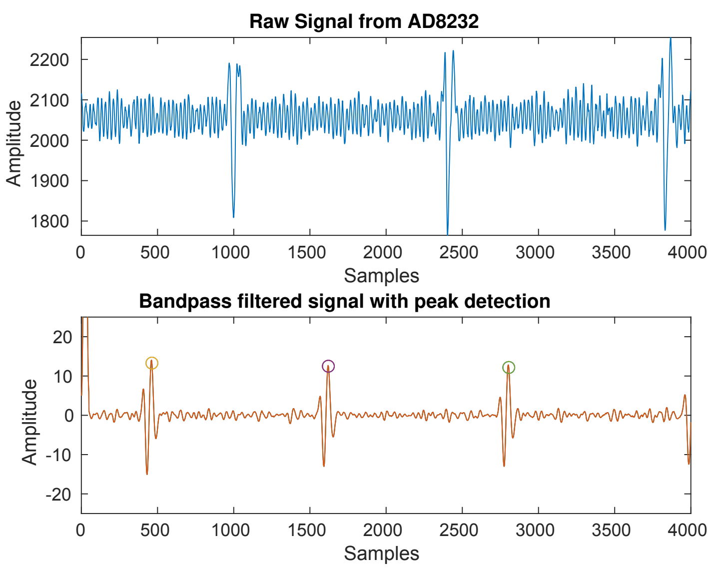


Fig. 3: On-device signal filtering on MSP430FR5994.

II. Method

A. Hardware

Figure 2 shows the block diagram of our system. IBPT setup consists of a transmitter and a wireless receiver. Both transmitter and receiver are capacitively coupled with the human body through 50-ohm wearable electrodes. The electrodes are configured as parallel plate capacitors where one of the conductive plates is in contact with the skin, and another serves as a floating ground to establish the return path. The proposed electrode setup utilizes a complex dielectric (human skin tissue) and air medium to complete the power transfer loop.

The transmitter includes a digital frequency synthesizer (AD9851) and a 50-ohm pulsed radio frequency (RF) amplifier. The synthesizer generates a 40 MHz RF carrier, and the RF amplifier amplifies the carrier to a power level of 28.5 dBm. The amplifier includes an onboard microcontroller (PIC16LF18426) that controls the duty cycle of the RF carrier. Duty cycling enables the amplifier to reduce its average power consumption. In addition, this duty cycling technique allows the amplifier to minimize heat generation without compromising the quality of the carrier.

On the receiver side, a 5-stage Dickson charge pump rectifies the RF signal harvested from the intrabody power transfer loop. It supplies a rectified 1.8V voltage; however, the amount of power that can be extracted from it is very low (in μW). The number of stages in the charge pump boosts the rectified DC level to charge a load capacitor. This load capacitor acts as an intermittent power supply for a battery manager. The battery manager serves as a power management unit for the sensors and the microcontroller to perform intermittent sensing and computation.

We use Texas Instruments' BQ25504 – an ultra-low power boost converter – for power management and is configured to charge a capacitor bank to a certain voltage (decided based on the amount of energy needed to run the application). For our application, we needed a 2.1 mF capacitor that needs 6 minutes to charge from 2.5V to 3.5V. The module outputs a $VBAT\_OK$ signal to indicate that the capacitor bank is charged to a certain voltage $VBAT\_OK\_HYST$ that is set to 3.5V in this setup. When the voltage drops below $VBAT\_OK$ voltage (set to 2.5V), the $VBAT\_OK$ signal goes low. This signal is used to drive the gate of a NMOS to supply the microcontroller (MSP430FR5994) which is also responsible for powering the sensors.

Signal Processing: Heart rate is detected by first collecting about 3000 samples from the AD8232. We smoothen this data by applying a 4-point moving average on the entire signal. Then, a bandpass FIR filter is applied with passband of 5Hz to 25Hz to remove unwanted frequencies. We invert the signal and perform peak detection to find the QRS complex. By using the average time between two peaks, heart rate can be computed. The filtered signal is shown in Figure 3.

Temperature is measured by calculating the change in resistance of the sensor using the ADC. Sensing the temperature value requires a 1.1 mA of current and a few milliseconds of computation, whereas to compute heart rate, the system requires a two-second data sample of ECG signal. Running the sensing system for 2 seconds drain the 2.1mF capacitor to 2.5V thus causing a power failure and it starts charging the capacitor.

TABLE I: BodyOS language abstraction to define task inter-dependence

Language Construct	Explanation
`INDEP_TASK(name,...)`	Declaration of an independent task
`DEP_TASK(name,...)`	Declaration of an dependent task
`INDEP_TASK_CONSTRAINTS(t1,range_max,range_min,expires...)`	Constraints for independent task
`DEP_TASK_CONSTRAINTS(t1,range_max,range_min, expires, ...)`	Constraints for dependent task
`DEPENDS_ON(t1,t2,MITD...)`	Captures dependence relation between tasks.

B. Software

Existing task-based systems [8] lack appropriate APIs to capture interdependencies between different body vitals that can help avoid unnecessary sensor sampling. However, expressing interdependencies requires domain knowledge that can only be provided by health experts. Unfortunately, existing system support for batteryless systems requires in-depth knowledge of intermittent computing challenges. BodyOS provides APIs, as shown in Table I, that hide the challenges of intermittent computing from domain experts while incorporating interdependence between body vitals in the application's control flow.

1) Task Definition & Control Block: (In)dependent tasks are the main building blocks of the BodyOS program. Each task is translated into a C function that is assigned a task control block in the non-volatile memory, which holds constraint parameters assigned by the programmer for scheduling, such as the min/max edges of the normal range and freshness time duration of the sensed data. Each (in)dependent task returns the sensed value which is a critical parameter of the application to schedule the next task in the control flow. The programmer assigns these constraints by using BodyOS's programming abstractions as shown in Table I.

2) Task Transition & Control Flow: Independent task is the first to be executed when the application starts performing computations. Based on the output of the independent task, scheduler selects the next task for transition. Finding the next task to execute plays a key role in energy-efficient execution of the application and this step starts with constraint evaluation on the dependence edges of the independent task.

3) Constraint Evaluation: BodyOS dynamic scheduler evaluates four main constraints to find the next task in control flow; `Normal-Range`, `MITD`, `Energy`, and `Data-Freshness`.

1) `Normal-Range` BodyOS evaluates if the output of an independent task falls within the normal range or not. If it does, BodyOS schedules the same task again. It saves energy by running independent tasks as much as possible since it has a higher correlation with the application's output and schedules dependent tasks only when the independent task senses an anomaly. In case of an anomaly, it inserts all dependent tasks of the current independent task in the task queue so that they can be executed next.
2) `MITD` Vital measurements have to be co-located in time to correctly represent wearer's health at that time instance. BodyOS ensures that the collected body vitals are co-located in time by evaluating the `MITD` constraint after each reboot. If the time spent to charge the capacitor exceeds `MITD`, it skips all remaining dependent tasks and schedules the independent task to collect fresh data. In this way, it saves energy by avoiding computations on stale data.
3) `Energy` Runtime checks energy availability in the capacitor to determine whether available energy is sufficient to execute the next task. Otherwise, BodyOS puts the system in sleep mode thus allowing the capacitor to be charged.
4) `Data-Freshness` This constraint checks if the data being processed by independent task is still fresh. If there is no violation, BodyOS makes the device device sleep till the capacitor is fully charged and continues running the dependent task.

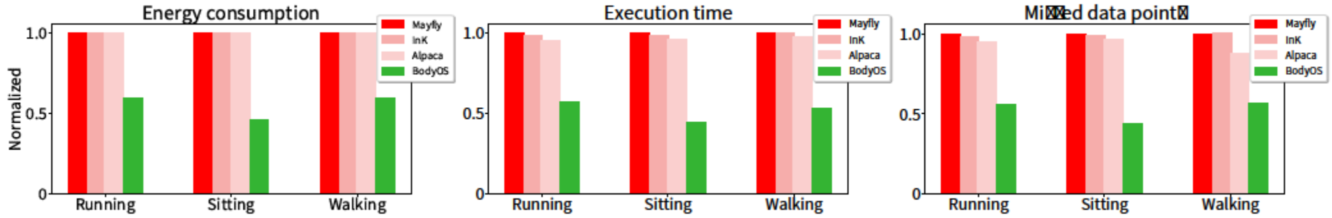
C. Evaluation

In our evaluation, we used a Patient Monitoring application that has three body vitals sensing tasks: Heart Rate (HR), Blood Pleasure (BP), and Oxygen level in blood (SpO2). The application collects HR, BP, and SpO2 vitals, respectively, as long as the capacitor has enough energy. In our evaluation, we compare the performance of BodyOS against existing intermittent computing approaches, i.e., Mayfly [8], InK [9], and Alpaca [10]. To ensure repeatability and fairness of comparison, we used a pre-recorded dataset [11] available in [12] to evaluate BodyOS's performance. The dataset contains body vitals for three types of activities which are (i) running, (ii) sitting, and (iii) walking. We developed a capacitor model to emulate application execution while using empirical measurements for each sensor's energy consumption (i.e., AD8232 and MAX30101) as well as the charging time of the capacitor when the IBPT transmitter and receiver were placed on the forearm and wrist, respectively, as shown in Figure 1. In our evaluation, we utilize four metrics to compare the systems; (i) missed data points, (ii) total execution time, (iii) average energy consumption of each data-point cycle, and (iv) memory overhead. In our evaluation setup, we observed each metric until each runtime was able to capture 600 data points in order to ensure a fair comparison. Results in Figure 4 are normalized to Mayfly's data for better understanding.

III. Results

We now share the results of our emulation.

1) BodyOS has lower energy consumption.: Figure 4a shows the energy consumption for intermittent computing runtimes. We can observe that BodyOS significantly reduces the energy consumption compared to existing approaches. `INDEP_TASK`/`DEP_TASK` interfaces allow data-aware execution to skip all dependent tasks if the independent task's output



(a) BodyOS lower energy consumption. (b) BodyOS requires shorter execution time. (c) BodyOS captures more data points.

Fig. 4: BodyOS is energy-efficient as it avoids unnecessary sensor sampling and allows the device to complete application faster while keeping the device active for a longer duration of time thus collecting more data events compared to existing intermittent computing approaches.

is in the normal range. This behavior is particularly visible when the participants were sitting. Since the body vitals remain in the normal range when sitting, BodyOS performs best by avoiding unnecessary sensing of vitals as shown in Figure 4a. On the other hand, running or walking generally increases body vitals. Therefore, BodyOS executes dependent task during these activities.

2) BodyOS has a shorter execution time.: Figure 4b shows the execution time when collecting data points for each activity. Similar behavior was observed when measuring execution time for each intermittent computing approach as shown in Figure 4a. Due to BodyOS's energy efficiency, the wearable remains active for a longer duration of time thus enabling the system to collect 600 data points in a shorter time compared to other approaches.

3) BodyOS captures more data points: BodyOS's energy-efficient execution enables the device to remain active for a longer duration of time, which has a direct impact on the number of data points captured by the application. Figure 4c shows the number of missed data points by the application for different activities. We can see that the number of data points missed by BodyOS is significantly lower compared to other approaches.

TABLE II: Memory and Code Size requirements (in B).

	InK		**Mayfly**		**Alpaca**		**BodyOS**	
	.text	*FRAM*	*.text*	*FRAM*	*.text*	*FRAM*	*.text*	*FRAM*
Required Memory	2428	4188	2190	4970	1212	326	4252	7242

4) Memory Overhead.: Table II shows the code size and memory requirement of the runtime systems. We implement BodyOS's main contributions on top of InK. BodyOS has a higher memory requirement and code size than the other runtimes as it relies on InK's memory consistency and intermittent constraint handling. However, these memory and code sizes are reasonable for intermittent hardware platforms. For instance, MSP430FR5994 which is one of the common microcontrollers for intermittent systems has a 256Kb FRAM size and can easily handle the increase in code size.

IV. Conclusion

We present BodyOS, a programming language and runtime support for an IBPT-powered health wearable that enables energy-efficient task scheduling and runtime decisions. It saves energy by allowing domain experts to express interdependencies between body vitals that help runtime avoid unnecessary task execution and schedule a task only when necessary. We evaluate BodyOS by emulating IBPT for harvesting energy and show that it significantly decreases energy consumption, execution time, and the number of missed data points compared to existing state-of-the-art systems.

References

[1] S. M. Iqbal, I. Mahgoub, E. Du, M. A. Leavitt, and W. Asghar, "Advances in healthcare wearable devices," *NPJ Flexible Electronics*, vol. 5, no. 1, p. 9, 2021.

[2] "Noncommunicable diseases," September 2022. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/noncommunicable-diseases

[3] S. Ahmed, B. Islam, K. S. Yildirim, M. Zimmerling, P. Pawełczak, M. H. Alizai, B. Lucia, L. Mottola, J. Sorber, and J. Hester, "The internet of batteryless things," *Commun. ACM*, vol. 67, no. 3, p. 64–73, feb 2024. [Online]. Available: https://doi.org/10.1145/3624718

[4] L. Wang, Y. Zhang, and P. G. Bruce, "Batteries for wearables," *National Science Review*, vol. 10, no. 1, p. nwac062, 2023.

[5] M. Afanasov, N. A. Bhatti, D. Campagna, G. Caslini, F. M. Centonze, K. Dolui, A. Maioli, E. Barone, M. H. Alizai, J. H. Siddiqui *et al.*, "Battery-less zero-maintenance embedded sensing at the mithræum of circus maximus," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 2020, pp. 368–381.

[6] J. De Winkel, V. Kortbeek, J. Hester, and P. Pawełczak, "Battery-free game boy," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 3, pp. 1–34, 2020.

[7] Y. Takahashi, K. Okura, S. Minakata, M. Watanabe, K. Hatakeyama, S. Chida, K. Saito, T. Matsunaga, and Y. Shimada, "Accuracy of heart rate and respiratory rate measurements using two types of wearable devices," *Progress in Rehabilitation Medicine*, vol. 7, p. 20220016, 2022.

[8] J. Hester, K. Storer, and J. Sorber, "Timely execution on intermittently powered batteryless sensors," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, 2017, pp. 1–13.

[9] K. S. Yıldırım, A. Y. Majid, D. Patoukas, K. Schaper, P. Pawelczak, and J. Hester, "Ink: Reactive kernel for tiny batteryless sensors," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, 2018, pp. 41–53.

[10] K. Maeng, A. Colin, and B. Lucia, "Alpaca: Intermittent execution without checkpoints," *Proceedings of the ACM on Programming Languages*, vol. 1, no. OOPSLA, pp. 1–30, 2017.

[11] P. Mehrgardt, M. Khushi, S. Poon, and A. Withana, "Pulse transit time ppg dataset," *PhysioNet*, vol. 10, pp. e215–e220, 2022.

[12] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals," *circulation*, vol. 101, no. 23, pp. e215–e220, 2000.