Raproto: An Open-Source Platform for Rapid Prototyping with Wearable Devices

Tarek Hamid*, Kimberly Helm*, Hyonyoung Choi*, Jean Park*, Claire Kendell*, Stephanie Cummings△, Steve Messe△, Stefanie Modri†, Insup Lee*, Amanda Watson*, James Weimer*
*Dept. of Computer and Information Science, University of Pennsylvania
△School of Medicine, University of Pennsylvania †School of Nursing, University of Pennsylvania
{thamid, kimhelm, hyonchoi, hlpark, ckendell, lee, aawatson, weimerj}@seas.upenn.edu, {stephanie.cummings, steven.messe}@pennmedicine.upenn.edu,{modris}@nursing.upenn.edu

***Abstract*—Advances in wearable technology have enabled ubiquitous use of wearable devices in remote patient monitoring, particularly in clinical trials. Because of the reliance on high-quality data in these endeavors, the first and often the most time-consuming step is to build a data collection system. While many systems have been developed to address this, they are often highly specific and customized to the task at hand, and are often not generalized enough to support other tasks. To remedy this, we developed Raproto, an open-source easy-to-use rapid prototyping platform that does not require the time, effort, and expertise needed for custom development. The Raproto platform consists of three components, the wearable device(s), communication protocol, and remote storage. These components support the collection, transmission, storage, analysis, and visualization of large-scale data with applications from smaller-scale research studies to large clinical trials. To reduce the burden of device and application development, we created multipurpose and customizable smartwatch applications on both the Android and Tizen operating systems. We evaluate our platform in a lab setting as well as in two real-world case studies. Overall, we find that we can collect data using our application for over 24 hours on a single charge and there is little to no data loss, thus making it an ideal tool to preface customized device development for real-world impact and commercialization.**

***Index Terms*—wearables, data collection, Android, Tizen, remote patient monitoring, clinical trials**

I. Introduction

The advent of wearable technology has enabled widespread adoption in health monitoring, particularly in clinical trials, allowing for remote, continuous data collection. While remote health monitoring has been successful thus far and is expected to continue growing [1], the implementation of these systems has introduced new challenges: they are costly to develop, and require extensive infrastructure set-up to transmit, store, and examine the gathered data [2]. As more remote health monitoring systems are developed, steps need to be taken to mitigate the cost and technical effort required for proper implementation of these systems. Many systems have been introduced to overcome these barriers, enabling the implementation of wearable devices in large-scale clinical trials. Manufacturers of popular smartwatch devices offer services for large-scale data collection such as ResearchKit and CareKit [3] for the Apple Watch. However, these applications are typically either confined to a single operating system, require extended permissions to collect certain kinds of data, or require an accompanying app that is either provided or must be developed by the study investigators. As such, a limited number of multipurpose data collection systems have been developed that use the sensors on commercially available smartwatches for their systems. A list of these systems can be found below in Table I.

TABLE I: Multipurpose Smartwatch Data Collection Systems

Platform Name	Year	Description
mCerebrum [4]	2016	Android app for multi-sensor data collection with complete storage and advanced analytics.
ROAMM [5]	2016	Tizen app for sensor data collection and remote server transmission for mobility monitoring.
WaDa [6]	2018	Android smartwatch app for sensor selection, labeling, and data export to a desktop file, limited to one watch at a time.
RADAR-Base [7]	2019	Uses wearable devices, accompanying questionnaire app, and third-party integration for data collection and storage for iOS and Android.
SenseCollect [8]	2021	Uses Android smartwatches to collect human activity data with a focus on data labeling.

Generally, while many of these data collection systems have provided great utility for smartwatch deployments with specific requirements, they are not generalizable towards diverse and broader applications where requirements may slightly change. Many of the platforms in Table I are either specific to a single data type [5], [8], are cumbersome to use with sensors spanning multiple participants at once as they require an accompanying mobile app [4], [7], do not export data that can be easily extracted and aggregated across multiple participants [6], or are not open-source [5]. This limitation becomes more pronounced in scenarios such as large-scale data collection in clinical trials or in hospitals, where the variability in data requirements such as different sensor utilization and the scale of operations in terms of interface usability and data storage expose the shortcomings of these systems.

In this paper, we present Raproto, an open-source platform enabling rapid prototyping with wearable devices. Raproto is an easy-to-use platform allowing for rapid development of data collection systems using wearable devices, decreasing the time, effort, and monetary contributions typically required for such endeavors. Our application leverages sensors already included with commercial smartwatches and allows for customization based on the needs of the project. This allows researchers to configure the watch to have extended battery life, collect more sensor data, and control how often data is transmitted for storage. The open-source code application repositories are available at [9] for the Tizen application and [10] for the Android application. This allows developers who need a more custom solution to easily extend the base application to suit their needs.

We evaluate Raproto in the lab and in two real world deployments. Firstly, we assess the Raproto applications by studying the battery life under different settings and find that data volume has the largest effect on battery life. Then, we analyze the data loss and data latency of the platform and find that by modifying our settings, we can achieve no data loss and very low data latency. Finally, we investigate how Raproto performs in two real-world deployments: in-hospital stroke detection and postpartum hemorrhage prediction. These real-world deployments illustrate system utility and usability, and provide insights for future improvements.

The contributions of this work include: (1) an open-source platform for rapid prototyping of wearable devices, (2) a customizable smartwatch application that collects data from available sensors at specified rates, and (3) an evaluation of our system in two real-world case studies: In-hospital Stroke Detection and Post-Partum Hemorrhage Prediction.

II. Raproto Platform

The Raproto platform consists of three main components: a smartwatch application, a communication protocol, and a remote server (see Fig. 1). This section provides a detailed overview of each component and their interactions.

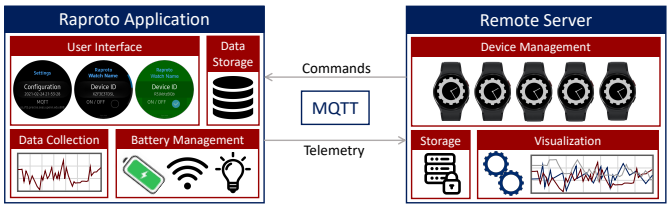

Fig. 1: Raproto Platform

A. Smartwatch Application

The Raproto application is designed to run on smartwatches with Tizen and Android operating systems. It facilitates data collection from the sensors embedded in these devices, offering a practical interface for configuration and control, without the need for a companion smartphone or mobile application. While these applications were developed across two operating systems, we have tested and evaluated them using the Samsung Galaxy Active for the Tizen operating system and the Galaxy Watch 5 for the Android operating system; functionality was also confirmed on the Google Pixel Watch 2.

(a) Android Main (b) Android Data Collect (c) Android Config

Fig. 2: User Interface of the Smartwatch Applications

a) User Interface: The Raproto application UI supports easy switching between Tizen and Android platforms, maintaining consistent functionality. It allows users to customize settings (device name, sensor configurations, battery management, data transmission intervals) manually or via a remote configuration file. Data collection is controlled by a toggle switch (Figure 2b), with green indicating success and red signaling an error. The app collects data in the background, even when the screen is off or another app is running, and can be stopped by toggling the switch off or selecting Save and Exit.

b) Watch Configuration: Raproto is designed to be fully customizable. Sensor selection, sampling rates, data transmission settings, and displayed device name are selected by the user. This customization supports targeted data collection and as-needed transmission, and can be completed remotely. In clinical applications, remotely updating the device avoids the logistical challenge of collection and distribution of devices to participants.

c) Data Collection: The Raproto applications collect de-identified sensor data from smartwatches. Both applications allow users to customize sensor combinations and sampling frequencies via the remote server's user interface, ensuring remote updates without interrupting the study.

d) Data Storage: The Raproto application support use of the smartwatch's local storage for on-device data collection. Sensor data is timestamped, labeled, and encoded in JSON format [11]. Data is collected in 10kb chunks and stored in a SQLlite database [12]. When ready, all stored chunks are transmitted to the remote server.

e) Battery Management: Raproto has been specifically configured to maximize battery life by offering customization for the display, sensor data volume, and communication radio settings, three factors that have the greatest impact on battery life in smartwatches [13]. The display settings are configured to turn off quickly, reduce brightness, and use a static black-and-white background, with a lock screen pin code to discourage interaction. Sensor data volume is managed by adjusting the number of sensors, sampling rates, and data resolution to

conserve battery life. Communication radios, particularly Wi-Fi, have three modes: Wi-Fi off, reconnect if dropped, and connect only during data transmission, balancing battery life and data transmission needs.

B. Communication Protocol

Raproto uses MQTT, a lightweight messaging protocol ideal for battery-constrained wearable devices and minimizing data loss. MQTT's small footprint and minimal bandwidth requirements ensure efficient data delivery. It offers three Quality of Service (QOS) levels: QOS 0 (low overhead, no delivery guarantee), QOS 1 (guaranteed delivery, possible duplication), and QOS 2 (exact delivery, higher overhead). Users can select the appropriate QOS level for their project, with QOS 1 being generally sufficient.

C. Remote Server

We designed Raproto to work with Thingsboard, an open-source IoT platform for device management, data collection, processing, and visualization [14]. Using SSL MQTT port 8883 for encrypted connections, our design allows compatibility with any MQTT broker/client combination. Thingsboard's web portal, which can be accessed via any commercial web browser, enables users to manage devices and data.

a) Device Management: Users can remotely configure and manage connected devices, adjusting settings such as sampling rates, device names, MQTT quality of service level, transmission rate, and Wi-Fi mode. Successful configuration is confirmed with a success message, and live data can be viewed to verify and debug configurations.

b) Data Storage: Data is stored in TimescaleDB, optimized for time-series data, allowing fast storage and efficient processing of multiple high-fidelity data streams. Users can view and export data using SQL queries through Thingsboard dashboards, which can be customized to fit specific needs.

c) Processing and Visualization: Thingsboard provides tools for data visualization and processing, aiding in quick debugging and usability for those with less technical knowledge. Customizable dashboards with widgets for time-series charts, text data, and boolean switches allow real-time data monitoring. Data filtering, enrichment, transformation, and rule chains can enhance data presentation. Alarms can alert users to real-time errors, expediting the debugging process.

III. System Evaluation

We evaluate the Raproto system to provide insight into various configurations and their impact on performance. In the following subsections, we examine factors that affect the battery life, analyze the vulnerability to data loss, and investigate the effects of data latency.

a) Baseline Performance Improvements: Raproto provides users with settings that conserve battery life. With our modifications, such as reducing screen brightness and disabling unused communication, we see a gain of approximately 4.8 hours of battery life for the Tizen smartwatch and a gain of approximately 6.2 hours of battery life for the Android smartwatch.

b) Sensor Data Volume: Testing revealed that at 10 Hz, both Tizen and Android smartwatches lasted over 24 hours. At 20 Hz, battery life dropped to around 16 hours, and at 200 Hz, it was about 12 hours. Rates above 250 Hz significantly reduced battery life. At 500 Hz, the Android watch failed to maintain the rate, and the Tizen watch struggled with transmission. At 1000 Hz, the Tizen watch experienced errors, reducing battery life to about 4.2 hours. Higher sampling rates drastically decrease battery life (Table II).

TABLE II: Expected Battery Life Using Accelerometers at Different Sampling Frequencies

Sampling Rate (Hz)	Battery Life	
	Tizen	Android
10	25.2 hrs	24.7 hrs
20	16.1 hrs	15.8 hrs
50	14.7 hrs	13.1 hrs
100	13.3 hrs	11.8 hrs
200	12.8 hrs	10.7 hrs
250	7.6 hrs	6.8 hrs
333	5.5 hrs	5.3 hrs
500	5.0 hrs	5.0 hrs
1000	4.2 hrs	5.0 hrs

To evaluate battery life across different sensors, we tested various combinations at a sampling rate of 50 Hz. HRM and PPG sensors used the least power, with Tizen lasting over 19 hours and Android over 20 hours. Accelerometer, Gyroscope, and Gravity sensors consumed more power, with Tizen lasting 18 hours and Android over 16 hours. Combining sensors further reduced battery life by 1-3 hours per additional sensor. With all five sensors active, Tizen lasted nearly 12 hours, and Android lasted 13.5 hours.

TABLE III: Sensor Combinations Expected Battery Life

Accel	Gyro	Gravity	HRM	PPG	Tizen	Android
x					18.1 hrs	16.8 hrs
	x				17.9 hrs	16.2 hrs
		x			17.8 hrs	16.3 hrs
			x		19.3 hrs	24.2 hrs
				x	19.1 hrs	n/a
x	x				17.0 hrs	15.1 hrs
x			x		17.9 hrs	16.5 hrs
x				x	17.9 hrs	n/a
x	x	x			15.6 hrs	14.8 hrs
x	x	x	x		14.1 hrs	13.4 hrs
x	x	x	x	x	11.9 hrs	n/a

c) Data Loss: Data loss can occur due to network overload or interference, such as with MRI scanners. We use MQTT's three QOS levels to mitigate this, allowing users to select the appropriate level. We evaluated data loss and duplication by sending 5000 messages per application and simulating network interference with a microwave oven [15]. Collecting accelerometer data at 50 Hz for five minutes, we found data loss only under QOS 0 (approximately 1% for both Android and Tizen). In critical scenarios, QOS 1 or 2 should be used, as they showed no data loss. QOS 2 also ensures no data duplication, though QOS 2 is generally unnecessary for our system.

d) Data Latency: Data latency, the time from smartwatch data collection to remote server storage, is influenced by transmission rate and environmental factors. The Raproto application sends data every minute by default, balancing battery life and real-time data needs. In a perfect environment with no interference, both Tizen and Android smartwatches showed less than one second of latency, which is negligible compared to other factors.

IV. Real World Deployments

A. Case Studies Overview

a) In-hospital Stroke Detection: We used Raproto to collect 4,000 hours of bi-lateral accelerometry data on 200 subjects to validate the StrokeDetectAI algorithm [16], designated a *Breakthrough Device* by the FDA. This study included 200 patients (167 control and 33 acute stroke subjects) over eight months, using twelve Samsung Galaxy Active Smartwatches connected to the Hospital of the University of Pennsylvania guest Wi-Fi network. We collected 4,169 hours of three-axis accelerometry data sampled at 5 Hz and streamed to a Thingsboard instance. Despite some issues like a subject removing a watch early and Wi-Fi connectivity problems due to an OS update, no significant data loss occurred thanks to MQTT QOS level 1.

b) Postpartum Hemorrhage Prediction: Raproto was used in a prospective study to collect 560 nm (green) PPG data on 525 patients over 4 months to validate a new PPH risk assessment algorithm [17], [18]. We used 35 Samsung Galaxy Active Smartwatches, scheduled to be worn from admission to 24 hours post-delivery. We collected 16,800 hours of PPG data sampled at 5 Hz, changing watches once during the data collection period when the battery level dropped below 20%. Overall, 16,823 hours of data were streamed to a Thingsboard instance. About 5% of subjects removed their watches early due to discomfort. The engineering team managed initial watch setup and application installation, with the clinical research team handling recruitment, watch placement, and monitoring.

B. Insights and Lessons Learned

Both deployments provided valuable insights into deploying consumer wearables for data collection in hospital environments. Raproto enabled clinical researchers to set up and operate smartwatches with minimal engineering support. Training allowed the clinical staff to manage hardware and software and confirm data collection through the Thingsboard web application. A significant challenge was the effort required to sideload the Raproto application via a command-line interface, which could be mitigated by providing engineering support or releasing Raproto in the app store. The clinical team heavily relied on Thingsboard to monitor battery life and ensure data collection, providing immediate visual feedback and enabling them to solve initial data collection issues. When data collection failed to start, restarting the watch resolved the issue. In the Postpartum Hemorrhage Prediction study, a temporary connection issue with Thingsboard caused some watches to fail to subscribe, resolved by directly encoding configuration settings into the application and updating Thingsboard. Having more watches than necessary alleviated the need for rapid responses to failures. Overall, these insights underscore the practicality and efficiency of using Raproto in clinical settings.

V. Conclusion

In this paper, we presented Raproto, an open-source platform for rapid prototyping with wearable devices. Raproto eliminates the need for extensive time, effort, and technical expertise in developing custom data collection systems. We created two customizable smartwatch applications to ease device and application development. Our evaluations showed that a smartwatch with our application collected data for over 24 hours on a single charge with minimal data loss in multiple real-world deployments. These deployments demonstrated Raproto's usability in clinical environments, providing data that would otherwise require significant hardware development and additional costs.

References

[1] "Remote patient monitoring system market growth & trends," https://www.grandviewresearch.com/press-release/global-remote-patient-monitoring-devices-market, published: January, 2021.

[2] C. Downey *et al.*, "The impact of continuous versus intermittent vital signs monitoring in hospitals: A systematic review and narrative synthesis," 2018.

[3] A. Inc., "Researchkit," https://www.apple.com/lae/researchkit/, 2024.

[4] S. Hossain *et al.*, "mcerebrum: A mobile sensing software platform for development and validation of digital biomarkers and interventions," in *SenSys*, 2017.

[5] M. Kheirkhahan *et al.*, "A smartwatch-based framework for real-time and online assessment and mobility monitoring," *Journal of biomedical informatics*, 2019.

[6] M. A. S. Mondol *et al.*, "Wada: An android smart watch app for sensor data collection," in *UBICOMP*, 2018, pp. 404–407.

[7] Y. Ranjan *et al.*, "Radar-base: Open source mobile health platform for collecting, monitoring, and analyzing data using sensors, wearables, and mobile devices," *JMIR Mhealth and Uhealth*, vol. 7, no. 8, 2019.

[8] W. Chen *et al.*, "Sensecollect: We need efficient ways to collect on-body sensor-based human activity data!" *IMWUT*, vol. 5, no. 3, 2021.

[9] A. Watson, "Raproto-tizen," https://github.com/aawatson22/Raproto-Tizen, 2022.

[10] J. Weimer, "Raproto-wearos," https://github.com/weimerj/Raproto-WearOS, 2022.

[11] F. Pezoa *et al.*, "Foundations of json schema," in *Proc. 25th Int. Conf. World Wide Web*, 2016.

[12] S. Bhosale *et al.*, "Sqlite: Light database system," *Int. J. Comput. Sci. Mob. Comput*, 2015.

[13] X. Liu and F. Qian, "Poster: measuring and optimizing android smartwatch energy consumption," in *MobiCom*, 2016.

[14] "Thingsboard: Open-source iot platform," 2022.

[15] D. Croce *et al.*, "Learning from errors: Detecting cross-technology interference in wifi networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 2, 2018.

[16] S. R. Messé *et al.*, "Derivation and validation of an algorithm to detect stroke using arm accelerometry data," *J Am Heart Assoc*, vol. 12, no. 3, p. e028819, 2023.

[17] K. Trout *et al.*, "Derivation and validation of an intrapartum algorithm to predict postpartum hemorrhage risk using photoplethysmography data," *American Journal of Obstetrics and Gynecology*, January 2024, iF 9.8.

[18] S. Modri *et al.*, "Observational clinical outcomes of a postpartum hemorrhage detection device development study," 2022.